## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>About This Guide</strong></td>
<td>21</td>
</tr>
<tr>
<td>Purpose</td>
<td>21</td>
</tr>
<tr>
<td>Prerequisites</td>
<td>21</td>
</tr>
<tr>
<td>Contents of this guide</td>
<td>21</td>
</tr>
<tr>
<td>GUI guide</td>
<td>24</td>
</tr>
<tr>
<td><strong>PART I Overview</strong></td>
<td>25</td>
</tr>
<tr>
<td><strong>CHAPTER 1 Introduction to LifeScope™ Genomic Analysis Software</strong></td>
<td>27</td>
</tr>
<tr>
<td>LifeScope™ Software Overview</td>
<td>27</td>
</tr>
<tr>
<td>Features</td>
<td>27</td>
</tr>
<tr>
<td>Data analyses</td>
<td>27</td>
</tr>
<tr>
<td>LifeScope™ Software Components</td>
<td>28</td>
</tr>
<tr>
<td>Workflows</td>
<td>28</td>
</tr>
<tr>
<td>Standard workflow</td>
<td>28</td>
</tr>
<tr>
<td>Project workflow</td>
<td>29</td>
</tr>
<tr>
<td>Analysis workflows</td>
<td>29</td>
</tr>
<tr>
<td>Primary and secondary file types</td>
<td>34</td>
</tr>
<tr>
<td>Input and output file formats</td>
<td>34</td>
</tr>
<tr>
<td><strong>PART II Installation</strong></td>
<td>35</td>
</tr>
<tr>
<td><strong>CHAPTER 2 LifeScope™ Genomic Analysis Software Installation</strong></td>
<td>37</td>
</tr>
<tr>
<td>Overview</td>
<td>37</td>
</tr>
<tr>
<td>Prerequisites</td>
<td>37</td>
</tr>
<tr>
<td>Workstation requirements</td>
<td>38</td>
</tr>
<tr>
<td>Cluster hardware requirements</td>
<td>38</td>
</tr>
<tr>
<td>Server requirements</td>
<td>38</td>
</tr>
<tr>
<td>Check firewall restrictions</td>
<td>39</td>
</tr>
<tr>
<td>Check DNS hostname</td>
<td>39</td>
</tr>
<tr>
<td>Client machines</td>
<td>39</td>
</tr>
<tr>
<td>Browser Requirements</td>
<td>39</td>
</tr>
<tr>
<td>LifeScope™ Software Administration</td>
<td>39</td>
</tr>
<tr>
<td>Installation workflow overview</td>
<td>41</td>
</tr>
</tbody>
</table>
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy data drive content</td>
<td>41</td>
</tr>
<tr>
<td>Download LifeScope™ Software</td>
<td>42</td>
</tr>
<tr>
<td>Install LifeScope™ Software</td>
<td>42</td>
</tr>
<tr>
<td><strong>Hardware configuration</strong></td>
<td>42</td>
</tr>
<tr>
<td>Authentication realm</td>
<td>42</td>
</tr>
<tr>
<td>Installation instructions</td>
<td>43</td>
</tr>
<tr>
<td>Continue installing LifeScope™ Software</td>
<td>46</td>
</tr>
<tr>
<td>Set the LifeScope™ Software environment</td>
<td>46</td>
</tr>
<tr>
<td>Start LifeScope™ Software servers and check servers’ status</td>
<td>47</td>
</tr>
<tr>
<td>Start the LifeScope™ Software server</td>
<td>47</td>
</tr>
<tr>
<td>Check the LifeScope™ Software server status</td>
<td>47</td>
</tr>
<tr>
<td>Download documentation and additional resources</td>
<td>47</td>
</tr>
<tr>
<td>Integrative Genomics Viewer (IGV)</td>
<td>48</td>
</tr>
<tr>
<td><strong>CHAPTER 3</strong> Verify the LifeScope™ Genomic Analysis Software Installation</td>
<td>49</td>
</tr>
<tr>
<td>Overview</td>
<td>49</td>
</tr>
<tr>
<td>Workflow</td>
<td>50</td>
</tr>
<tr>
<td>Verify the installation</td>
<td>50</td>
</tr>
<tr>
<td><em>(Optional)</em> Download and install demos files</td>
<td>51</td>
</tr>
<tr>
<td><em>(Optional)</em> Run demo analyses</td>
<td>51</td>
</tr>
<tr>
<td>Check results</td>
<td>52</td>
</tr>
<tr>
<td><em>(Optional)</em> Download and install performance verification scripts</td>
<td>52</td>
</tr>
<tr>
<td><em>(Optional)</em> Run the performance verification test</td>
<td>52</td>
</tr>
<tr>
<td>Monitor progress</td>
<td>53</td>
</tr>
<tr>
<td>Check results</td>
<td>53</td>
</tr>
<tr>
<td>Create user accounts</td>
<td>53</td>
</tr>
<tr>
<td>Distribute the URL and user credentials</td>
<td>54</td>
</tr>
<tr>
<td>Maintenance</td>
<td>55</td>
</tr>
<tr>
<td>Stop the LifeScope™ Software server</td>
<td>55</td>
</tr>
<tr>
<td>Upgrade</td>
<td>55</td>
</tr>
<tr>
<td>Uninstall</td>
<td>55</td>
</tr>
<tr>
<td><strong>PART III</strong> Getting Started</td>
<td>57</td>
</tr>
<tr>
<td><strong>CHAPTER 4</strong> Before You Begin</td>
<td>59</td>
</tr>
<tr>
<td>Overview</td>
<td>59</td>
</tr>
<tr>
<td>Verify the browser version</td>
<td>60</td>
</tr>
<tr>
<td>Set the LifeScope™ Software environment</td>
<td>60</td>
</tr>
<tr>
<td>BioScope™ Software users’ PATH variable</td>
<td>60</td>
</tr>
<tr>
<td>BioScope™ Software profile scripts</td>
<td>60</td>
</tr>
<tr>
<td>Clusters running both versions</td>
<td>60</td>
</tr>
</tbody>
</table>
CHAPTER 5

Understand The LifeScope™ Software Shell .......................... 63

Overview ................................................................. 63
Terminology and concepts .............................................. 64
   Projects and analyses .............................................. 66
   Naming restrictions ................................................. 68
   Reads Data .......................................................... 68
   Repositories ......................................................... 68
   The reference repository ......................................... 69
Analysis space ......................................................... 69
Find reference files for an analysis ................................. 70
Find data for an analysis ............................................. 71
View analysis information ........................................... 71
Common scenarios ..................................................... 72
   Basic ................................................................. 72
   A sample sequenced on multiple lanes ......................... 72
   Data from multiple samples .................................... 72
Naming restrictions .................................................... 73

CHAPTER 6

Run a Command Shell Analysis ................................. 75

Overview ................................................................. 75
Run a standard workflow ........................................... 76
   Example steps to run a standard workflow .................... 76
   Explanation of example steps .................................... 77
Use the command shell ................................................ 77
   Run mode .......................................................... 78
   Status mode ....................................................... 79
   Scripted mode ..................................................... 80
   Shell mode ......................................................... 81
List shell commands ................................................ 89
Options ................................................................. 89
Define input data ...................................................... 90
Example input data scenarios ...................................... 90
Index IDs ............................................................... 92
Run a grouped analysis .............................................. 93
   Example of a groupsfile ......................................... 93
Run a multi-group analysis ....................................... 93
Run a tertiary-only workflow ..................................... 95
   With add bam commands ....................................... 95
   With a groupsfile ................................................ 95
Run a multi-group tertiary analysis ............................. 96
Run an individual analysis ......................................... 97
Contents

Resume, re-run, and re-use ................................................................. 98
  Re-run an analysis in the command shell ........................................ 98
Review job status .............................................................................. 99
Review results ............................................................................... 100
Review logs .................................................................................... 100
Error behavior and error messages .................................................. 100
  Case sensitivity ............................................................................ 100
  Unrecognized commands .............................................................. 101
  Illegal usage ............................................................................... 101

CHAPTER 7
Run a Standard Workflow Analysis ............................................. 103
  Overview .................................................................................... 103
  Standard workflows .................................................................... 103
    List standard workflows ........................................................ 107
    List a workflow’s INI files and parameters ................................ 108
    Workflow directory structure .................................................. 108
    Workflow modules and parameters ........................................ 109
  Run a standard workflow ............................................................ 110
    Prepare to run a standard workflow ........................................ 110
    Order commands correctly ...................................................... 110
    Specify the regions of interest file ......................................... 111
    Run a workflow in the command shell .................................... 111
    Run a workflow in lscope.sh run mode ................................... 113
  Create a new workflow ................................................................ 113
    With the Linux cp command ................................................... 114
    With the shell get workflow command .................................... 114
  Workflow control files ................................................................. 115
    PLN files .................................................................................. 115
    INI files .................................................................................. 116
    .run parameters ....................................................................... 116
  Run the tertiary portion of a workflow ........................................... 117

PART IV
Analysis Modules ........................................................................... 119

CHAPTER 8
Run the SOLiD® Accuracy Enhancement Tool .......................... 121
  Overview .................................................................................... 121
  SAET usage guidelines ............................................................... 121
  Examples of how to run SAET .................................................... 122
    Workflows using SAET ............................................................ 122
    SAET example in a workflow .................................................. 122
    SAET usage in standard workflows .......................................... 122
  SAET input files ........................................................................... 123
### Chapter 9

**Run a Resequencing Mapping Analysis**

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>129</td>
</tr>
<tr>
<td>Examples of how to run a mapping analysis</td>
<td>130</td>
</tr>
<tr>
<td>In a standard workflow</td>
<td>130</td>
</tr>
<tr>
<td>As a demo analysis</td>
<td>131</td>
</tr>
<tr>
<td>Input files</td>
<td>131</td>
</tr>
<tr>
<td>Plan your input read-sets</td>
<td>131</td>
</tr>
<tr>
<td>Legacy data</td>
<td>131</td>
</tr>
<tr>
<td>Stages of mapping</td>
<td>131</td>
</tr>
<tr>
<td>Scatter</td>
<td>133</td>
</tr>
<tr>
<td>Fragment mapping</td>
<td>134</td>
</tr>
<tr>
<td>Mate-pair mapping</td>
<td>134</td>
</tr>
<tr>
<td>Pairing</td>
<td>134</td>
</tr>
<tr>
<td>Mark duplicates</td>
<td>136</td>
</tr>
<tr>
<td>BAM file generation</td>
<td>137</td>
</tr>
<tr>
<td>Quality alignments and unique alignments</td>
<td>140</td>
</tr>
<tr>
<td>Fragment mapping parameters</td>
<td>141</td>
</tr>
<tr>
<td>Fragment mapping parameters table</td>
<td>142</td>
</tr>
<tr>
<td>Mapping performance</td>
<td>144</td>
</tr>
<tr>
<td>Mapping algorithm</td>
<td>145</td>
</tr>
<tr>
<td>Internal mapping parameters</td>
<td>145</td>
</tr>
<tr>
<td>Mapping schemes</td>
<td>151</td>
</tr>
<tr>
<td>Pairing algorithm</td>
<td>155</td>
</tr>
<tr>
<td>Pairing parameters</td>
<td>155</td>
</tr>
<tr>
<td>Internal pairing parameters</td>
<td>155</td>
</tr>
<tr>
<td>Indel sizes</td>
<td>158</td>
</tr>
<tr>
<td>Mapping output files</td>
<td>159</td>
</tr>
<tr>
<td>Mapping statistics</td>
<td>159</td>
</tr>
<tr>
<td>Mapping statistics parameters</td>
<td>160</td>
</tr>
<tr>
<td>Summary of mapping statistics output</td>
<td>161</td>
</tr>
<tr>
<td>Mapping statistics output files</td>
<td>163</td>
</tr>
<tr>
<td>Raw number files</td>
<td>168</td>
</tr>
<tr>
<td>Mapping statistics example output</td>
<td>168</td>
</tr>
<tr>
<td>Run BAMStats standalone</td>
<td>172</td>
</tr>
</tbody>
</table>
CHAPTER 10

**Run a SNPs Analysis** ........................................... 181

Overview ............................................................. 181
SNPs input files ..................................................... 182
Examples of running a SNPs analysis ......................... 184
   Examples in a standard workflow ............................ 184
   As a demo analysis ............................................. 184
SNPs runtime parameters ........................................ 185
   The call stringency parameter ............................... 190
   The skip high coverage filter ............................... 191
   The reads mapping QV parameter .......................... 191
   The minimum base QV filters .............................. 191
   Other filtering parameters ................................. 191
SNPs internal parameters ....................................... 192
SNPs output files .................................................. 192
   VCF file ......................................................... 193
   VCF format example ...................................... 193
   GFF file format ............................................ 194
   GFF file example ........................................... 195
   Combined GFF for all contigs ............................. 196
   Consensus_Calls file format ............................... 196
   Consensus calls file example ............................. 199
   Contig base space FASTA file ............................. 200
   Combined FASTA for all contigs .......................... 200
   Quartiles file .................................................. 200
SNPs algorithm description ..................................... 201
   Frequentist algorithm ...................................... 201
   Bayesian algorithm .......................................... 201
   Data flow ....................................................... 202
   Internal module flow ....................................... 202
FAQ – SNPs .......................................................... 204

CHAPTER 11

**Run a Human CNVs Analysis** ................................. 209

Overview ............................................................. 209
Examples of running a CNV analysis ......................... 210
   In a standard workflow .................................... 210
   As a demo analysis ........................................... 210
CNV module parameter descriptions ......................... 211
Human CNVs results file format description ................ 214
OUT files .......................................................... 214
PART V  Targeted Resequecing and Low Frequency Variants  .................. 321

CHAPTER 16  Run Targeted Resequecing Analyses  ......................... 323
  Overview .................................................. 323
  Example of running targeted resequencing analyses .................. 324
  Targeted resequencing read selection algorithm .................... 325
  Run targeted resequencing modules .................................. 326

CHAPTER 17  Run Low Frequency Variant Analyses ......................... 329
  Overview .................................................. 329
  Examples of running low frequency variant analyses ................ 330
  Low frequency variant read selection algorithm ...................... 331
  SNPs optimizations ......................................... 332
  Run low frequency variant modules .................................. 334
CHAPTER 18  
Run Enrichment Analyses  ........................................ 335

Overview ................................................................. 335
Enrichment ............................................................. 335
Enrichment statistics ................................................... 336
Examples of running the enrichment module .................... 336
In a standard workflow .............................................. 336
As a demo analysis ..................................................... 337
Enrichment parameters ................................................. 337
Enrichment statistics input files ..................................... 339
The input target regions file .......................................... 339
The input aligned reads file ......................................... 340
Enrichment statistics output files .................................... 340
The output target file ................................................ 341
The output alignment file .......................................... 341
The target coverage BEDGRAPH file ............................... 341
The genome coverage frequency file ............................... 342
The target coverage frequency file ................................. 343
The target statistics report file ..................................... 343
The enrichment statistics report .................................... 344

PART VI  
Whole Transcriptome Analyses  ....................... 347

CHAPTER 19  
Run a Whole Transcriptome Mapping Analysis .......... 349

Overview ................................................................. 349
Examples of running the WT mapping module .................. 351
In a standard workflow .............................................. 351
As a demo analysis ..................................................... 352
Input files ................................................................. 352
Reads input files ...................................................... 352
Reference input files .................................................. 353
Annotations input files ............................................... 353
Stages of mapping ...................................................... 355
Scatter ................................................................. 357
Single-read mapping .................................................. 358
Filter mapping .......................................................... 358
Junction mapping ...................................................... 359
Exon mapping .......................................................... 359
Rescue method .......................................................... 359
Pairing reads ............................................................ 361
BAM file generation .................................................. 362
Quality alignments and unique alignments ....................... 364
Single-read mapping parameters .................................. 365
## Contents

- Mapping performance ...................................... 367
- WT mapping internal parameters .......................... 369
- Mapping schemes ........................................... 370
- Paired-end parameters ..................................... 371
- Mapping output files ...................................... 376
  - Overview .................................................. 376
  - BAM file differences .................................... 377
  - WT filtering stats ....................................... 378
- Mapping statistics .......................................... 379
  - Mapping statistics parameters .......................... 379
  - Summary of mapping statistics output ............... 380
- Mapping statistics output files ............................ 382
  - Raw number files ....................................... 386
  - Mapping statistics example output .................... 387
  - Run BAMStats standalone ................................ 390
- FAQ – Whole transcriptome ................................. 391
- FAQ – BAMStats ............................................ 394

### CHAPTER 20

- **Run a WT Coverage Analysis** ............................ 397
  - Overview .................................................. 397
  - WT coverage input files ................................ 397
  - WT coverage parameters ................................ 398
  - Coverage output files ................................... 399

### CHAPTER 21

- **Run a WT Count Known Genes and Exons Analysis** .... 401
  - Overview .................................................. 401
  - Input files ................................................ 402
  - WT counts parameter description ...................... 402
  - WT counts algorithm description ....................... 403
    - Filters .................................................. 403
    - Counts .................................................. 404
    - RPKM .................................................... 405
  - Output files ............................................. 405
    - GTF output ............................................ 405
    - Tab output ............................................. 405
  - FAQ – WT counts ........................................ 406

### CHAPTER 22

- **Run a WT Splice Finder Analysis** ...................... 407
  - Overview .................................................. 407
  - Splice finder parameters ................................ 408
  - SASR splice finder ....................................... 411
    - SASR splice finder module summary ................. 411
## Contents

- SASR splice finder module evidence evaluator ............................................. 412
- Call junctions and fusions with single read only ........................................... 413
- Output files ................................................................................................. 414
- WT fragment differences ............................................................................. 414
- Tabular junction files ................................................................................... 414
- Junction examples ....................................................................................... 416
- Browser Extensible Display (BED) output ..................................................... 418
- SEQ .............................................................................................................. 419
- Circos .......................................................................................................... 419
- Output format values ................................................................................... 419

### PART VII  Small RNA Analyses ................................................................. 421

#### CHAPTER 23  Run a Small RNA Mapping Analysis ............................... 423

- Overview ....................................................................................................... 423
- Example of running the small RNA mapping module .................................. 424
- Small RNA mapping input files .................................................................... 425
- Stages of mapping ......................................................................................... 426
  - Scatter ....................................................................................................... 427
  - Filter map .................................................................................................. 427
  - miRBase map ............................................................................................ 428
  - Genome map ............................................................................................. 428
  - miRBase2Genome ..................................................................................... 428
  - BAM file generation ................................................................................... 428
- Quality alignments and unique alignments ..................................................... 430
- Small RNA mapping parameters ................................................................ 431
  - Mapping performance ............................................................................... 434
  - Internal parameters ................................................................................... 435
  - Mapping schemes ..................................................................................... 437
- Small RNA mapping output files ................................................................ 438
- Mapping statistics ....................................................................................... 439
  - Mapping statistics parameters ................................................................. 439
  - Summary of mapping statistics output ..................................................... 441
  - Mapping statistics output files ................................................................. 443
  - Raw number files ...................................................................................... 446
  - Mapping statistics example output .......................................................... 447
  - Run BAMStats standalone ........................................................................ 449
- FAQ - Small RNA mapping ......................................................................... 449
- FAQ – BAMStats ......................................................................................... 454

#### CHAPTER 24  Run a Small RNA Coverage Analysis ............................... 457

- Overview ....................................................................................................... 457
**Contents**

Example of running the small RNA coverage module .............................................. 457
Small RNA coverage input files ............................................................................... 458
Small RNA coverage parameters ........................................................................... 458
Small RNA coverage output files ........................................................................... 459

**CHAPTER 25**

Run a Small RNA Counts Analysis ................................................................. 461

Overview .............................................................................................................. 461
Example of running the small RNA count module ............................................. 461
Small RNA counts .............................................................................................. 462
Small RNA counts input files .............................................................................. 462
  Alignments files .................................................................................................. 463
  Precursor sequences .......................................................................................... 463
  Mature form sequences ...................................................................................... 463
Small RNA counts parameters ............................................................................ 464
  Mapped file parameters ..................................................................................... 464
  Filtered BAM file parameters .......................................................................... 465
Small RNA counts output files ............................................................................ 465
  Mapped output .................................................................................................. 465
  Filtered counts output ....................................................................................... 467

**PART VIII**

Additional Mapping Analyses ................................................................. 469

**CHAPTER 26**

Run a ChIP-Seq Mapping Analysis ................................................................. 471

Overview .............................................................................................................. 471
Run ChIP-Seq mapping ....................................................................................... 471
  Run in the Lscope command shell ................................................................... 471
  Do not rename .................................................................................................. 473
  Turn off SAET .................................................................................................. 473
Run ChIP-Seq as an individual analysis ............................................................. 474
Mapping algorithm .............................................................................................. 474
Use results files .................................................................................................... 474

**CHAPTER 27**

Run a MethylMiner™ Mapping Analysis ...................................................... 477

Overview .............................................................................................................. 477
Run MethylMiner™ mapping .............................................................................. 478
  Run MethylMiner™ workflows ....................................................................... 479
  Run in the Lscope command shell ................................................................... 479
  Do not rename .................................................................................................. 480
  Visualize MethylMiner™ results .................................................................... 481
MethylMiner™ mapping output files .................................................................... 481
Mapping algorithm .............................................................................................. 481
Further analysis of MethylMiner™ mapping results ........................................... 481

PART IX  Appendices ......................................................... 483

APPENDIX A  FAQ .............................................................. 485

  FAQ lists for LifeScope™ Genomic Analysis Software modules ...................... 485
  General FAQ for LifeScope™ Software ......................................................... 485

APPENDIX B  Troubleshooting ................................................ 489

  Overview ......................................................................................... 489
  Logs ............................................................................................... 489
  Troubleshooting .............................................................................. 490
    Login issues .................................................................................. 490
    Analysis failures .......................................................................... 490
    The server responds with an error .................................................... 490
    Message: Read-only db connection .................................................. 490
    Sequence name not found in reference ............................................ 490
    Memory errors .............................................................................. 491
    Special characters in BAM header fields .......................................... 491

APPENDIX C  File Format Descriptions ........................................... 493

  Introduction ..................................................................................... 493
  XSQ file format ................................................................................ 494
    XSQ file content overview ............................................................. 494
    Relation to sequencing instrument ................................................. 495
    XSQ file format description .......................................................... 495
  BAM header usage ........................................................................... 498
    Sequence dictionary [@SQ] .............................................................. 498
    Read group [@RG] .......................................................................... 498
    Header [@HD] sort order ................................................................. 498
    BAM file validation ....................................................................... 499
    XSQ metadata in BAM headers ...................................................... 499
  Color-space attributes ...................................................................... 504
  Pairing information in a BAM file ..................................................... 505
    Calculation of tag names ............................................................... 505
    Proper pairs .................................................................................. 505
    Single read mapping quality ........................................................ 506
  Hard clipping of incomplete extensions ............................................. 506
  Visualize BAM output ........................................................................ 507
    Integrative Genomics View (IGV) ................................................... 507
    UC Santa Cruz (UCSC) genome browser ........................................ 507
Appendix D  XSQ Tools ................................. 519

Overview ........................................... 519
The XSQ Tools package .............................. 520
 Package components .................................. 520
 Download instructions ............................... 521
 Conversion to the XSQ format ...................... 521
 Conversion from the XSQ format .................. 525
 Conversion syntax .................................. 525
 Filter option ....................................... 525
 Options table ..................................... 526
 XSQ file splitting ................................... 526
 XSQ index reassignment ............................ 526
 Purpose ............................................ 526
 Usage ............................................. 527
 Index reassignment options table ............... 527
 HIV5 tools ......................................... 527
 Internal conversion parameters .................. 528
 Parameters table .................................. 528
 Internal INI file .................................. 530
 Links to resources ................................ 533
 FAQ - XSQ Tools .................................. 533

Appendix E  The Reference Repository ............... 539

Overview ........................................... 539
 Repository structure ............................... 539
 Repository location ............................... 540
About This Guide

Purpose

This guide is designed to help you quickly perform next-generation sequencing analyses using LifeScope™ Software. These analyses support fragment, paired-end (PE) and long mate pair (LMP) library types of analyses. Specifically, the following workflows are supported:

- Whole genome resequencing
- Targeted resequencing
- Low frequency variant resequencing
- Whole transcriptome resequencing
- Small RNA resequencing
- ChIP-Seq mapping
- MethylMiner™ mapping

Prerequisites

It is assumed that you have working knowledge of the:

- Linux® operating system
- Internet Protocol (IP) address of the LifeScope™ Software cluster.
- Linux environment and know how to:
  - Navigate to directories.
  - Edit and save files in a text editor.
  - Run Linux shell scripts.
  - Run basic Linux commands such as `chmod`, `ps`, `pwd`, `cd`, `echo`, `grep`, and other commands.

Contents of this guide

The chapters and appendices in this guide are described in the following table.

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part I Overview</td>
<td>1 &quot;Introduction to LifeScope™ Genomic Analysis Software&quot;</td>
<td>A high-level description of LifeScope™ Software, including features, workflows, and file formats.</td>
</tr>
<tr>
<td>Part II Installation</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

LifeScope™ Genomic Analysis Software Command Shell User Guide 21
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>&quot;LifeScope™ Genomic Analysis Software Installation&quot;</td>
<td>For the LifeScope™ Software administrator, describes the LifeScope™ Software installation procedure, and describes system software and hardware requirements.</td>
</tr>
<tr>
<td>3</td>
<td>&quot;Verify the LifeScope™ Genomic Analysis Software Installation&quot;</td>
<td>For the LifeScope™ Software administrator, describes the LifeScope™ Software installation procedure, and describes system software and hardware requirements.</td>
</tr>
</tbody>
</table>

### Part III Getting Started

| 4 | "Before You Begin" | Describes general prerequisites and provides procedures to complete after installation and before you run LifeScope™ Software. |
| 5 | "Understand The LifeScope™ Software Shell" | Describes the new concepts and terminology required when working with LifeScope™ Software command shell. |
| 6 | "Run a Command Shell Analysis" | Lists and explains the LifeScope™ Software shell commands, and describes how to run analyses and workflows in the shell. |
| 7 | "Run a Standard Workflow Analysis" | Lists and describes the standard workflows available with LifeScope™ Software. |

### Part IV Analysis Modules

| 8 | "Run the SOLiD® Accuracy Enhancement Tool" | Describes using SAET to improve color call accuracy prior to mapping. |
| 9 | "Run a Resequencing Mapping Analysis" | Describes secondary processing, including mapping, pairing, BAM generation, mapping statistics, parameters, and algorithms. |
| 10 | "Run a SNPs Analysis" | Describes the Single Nucleotide Polymorphisms analysis module, including command shell examples, parameters, and algorithms. |
| 11 | "Run a Human CNVs Analysis" | Describes the Human Copy Number Variation analysis module, including command shell examples, parameters, and algorithms. |
| 12 | "Run an Inversions Analysis" | Describes the inversions analysis module, including command shell examples, parameters, and algorithms. |
| 13 | "Run a Small Indels Analysis" | Describes the small indel analysis module, including command shell examples, parameters, and algorithms. |
| 14 | "Run a Large Indels Analysis" | Describes the large indel analysis module, including command shell examples, parameters, and algorithms. |
| 15 | "Add Genomic Annotations to Analysis Results" | Describes the genomic annotations analysis module, including command shell examples, parameters, annotation sources, and reports. |

### Part V Targeted Resequencing and Low Frequency Variants

| 16 | "Run Targeted Resequencing Analyses" | Describes targeted resequencing analyses within LifeScope™ Software. |
| 17 | "Run Low Frequency Variant Analyses" | Describes low frequency variant detection analyses within LifeScope™ Software. |
| 18 | "Run Enrichment Analyses" | Describes the enrichment module used with targeted resequencing and low frequency variant workflows, including command shell examples, parameters, and reports. |

### Part VI Whole Transcriptome Analyses
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>“Run a Whole Transcriptome Mapping Analysis”</td>
<td>Describes WTA secondary processing, including mapping, pairing, BAM generation, mapping statistics, mapping and pairing parameters, and mapping and pairing algorithms.</td>
</tr>
<tr>
<td>20</td>
<td>“Run a WT Coverage Analysis”</td>
<td>Describes the coverage module, which calculates read coverage per position.</td>
</tr>
<tr>
<td>21</td>
<td>“Run a WT Count Known Genes and Exons Analysis”</td>
<td>Describes the counts module, including command shell examples, parameters, and algorithms. This module counts the number of reads that align within genomic features, and counts and annotates the number of exons that are expressed in a gene region.</td>
</tr>
<tr>
<td>22</td>
<td>“Run a WT Splice Finder Analysis”</td>
<td>Describes the splice finder module, including parameters and algorithms. This module discovers splice junctions (introns from pre-messenger-RNAs to generate mature messenger-RNAs) and annotates the type and evidence for each discovered junction.</td>
</tr>
<tr>
<td></td>
<td>Part VII Small RNA Analyses</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>“Run a Small RNA Mapping Analysis”</td>
<td>Describes small RNA secondary processing, including mapping, pairing, BAM generation, mapping statistics, parameters, and algorithms.</td>
</tr>
<tr>
<td>24</td>
<td>“Run a Small RNA Coverage Analysis”</td>
<td>Describes the coverage module, which calculates read coverage per position.</td>
</tr>
<tr>
<td>25</td>
<td>“Run a Small RNA Counts Analysis”</td>
<td>Describes the counts module, including command shell examples and parameters.</td>
</tr>
<tr>
<td></td>
<td>Part VIII Additional Analyses</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>“Run a ChIP-Seq Mapping Analysis”</td>
<td>Describes using LifeScope™ Software to map data and create an output file type compatible with a variety of third-party Chromatin Immunoprecipitation Sequencing (ChIP-Seq) data analysis tools.</td>
</tr>
<tr>
<td>27</td>
<td>“Run a MethylMiner™ Mapping Analysis”</td>
<td>Describes using LifeScope™ Software for focused evaluation of methylation patterns in genome-wide studies (with the use of 5500 or SOLiD® System sequencing data).</td>
</tr>
<tr>
<td></td>
<td>Part IX Appendices</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>“FAQ”</td>
<td>Lists frequently-asked questions, and answers, for LifeScope™ Software. as a whole. FAQs for individual modules are located in the module chapters.</td>
</tr>
<tr>
<td>B</td>
<td>“Troubleshooting”</td>
<td>Describes LifeScope™ Software log files and tips for dealing with certain errors.</td>
</tr>
<tr>
<td>C</td>
<td>“File Format Descriptions”</td>
<td>Describes LifeScope™ Software’s use of BAM and XSQ files and their fields; XSQ, BED, and BEDGRAPH file formats; an overview of reference files; clipping issues; and how to visualize BAM output.</td>
</tr>
<tr>
<td>D</td>
<td>“XSQ Tools”</td>
<td>Describes XSQ conversion and manipulation tools available in the XSQ Tools package (a separate download from LifeScope™ Software).</td>
</tr>
<tr>
<td>E</td>
<td>“The Reference Repository”</td>
<td>Describes the LifeScope™ Software reference repository, including the repository structure, its initial contents [from the separate data drive], the importance and use of assembly names, adding a new species assembly, converting reference files, and reference file conflicts.</td>
</tr>
</tbody>
</table>
### Chapter Title Description

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>F</strong></td>
<td>&quot;Command Shell Control Files&quot;</td>
<td>Describes the control files used with the command shell (INI, PLN, properties, and XML files), and LifeScope™ Software framework parameters.</td>
</tr>
<tr>
<td><strong>G</strong></td>
<td>&quot;Administration&quot;</td>
<td>Describes the admin functionality available in the command shell. The GUI Admin Portal, not the command shell, is recommended for admin tasks. Refer to the <em>LifeScope™ Genomic Analysis Software User Guide</em> (Part no. 4476537).</td>
</tr>
<tr>
<td><strong>H</strong></td>
<td>&quot;Demo Analyses&quot;</td>
<td>Describes the example demos, which are used for installation verification.</td>
</tr>
<tr>
<td><strong>I</strong></td>
<td>&quot;LIFESCOPE™ GENOMIC ANALYSIS SOFTWARE v2&quot; EULA</td>
<td>The LifeScope™ Software End User License agreement.</td>
</tr>
<tr>
<td>—</td>
<td>Documentation</td>
<td>Lists other documentation related to LifeScope™ Software.</td>
</tr>
<tr>
<td>—</td>
<td>Index</td>
<td>Index to this guide.</td>
</tr>
</tbody>
</table>

---

**GUI guide**

See the companion document *LifeScope™ Genomic Analysis Software User Guide* (Part No. 4476537) for instructions to use the LifeScope™ Software graphical user interface (GUI).

The GUI guide also covers these topics that are not available in the command shell or are available with limited functionality:

- Appendix G, “Administration”
- Appendix E, “Convert 1.3 BAM Files”
PART I
Overview
1

Introduction to LifeScope™ Genomic Analysis Software

This introduction covers:
- LifeScope™ Software Overview ................................................................. 27
- Workflows ...................................................................................................... 28
- Primary and secondary file types ................................................................. 34
- Input and output file formats ........................................................................ 34

LifeScope™ Software Overview

After years of development on analysis tools for SOLiD® data, in response to customer feedback, Applied Biosystems LifeScope™ Genomic Analysis Software enables fast translation of next-generation data for biologically meaningful results. LifeScope™ Software matches the accuracy of the next generation 5500 Series Genetic Analysis Systems with Exact Call Chemistry (ECC) and streamlines your data analysis.

Features

LifeScope™ Software is part of the LifeScope™ Genomic Analysis Solution. The complete informatics solution is comprised of genomics analysis software combined with a specified hardware platform. LifeScope™ Software features:
- Seamless integration with the 5500 Series Genetic Analysis Systems
- Performance-tuned algorithms for the 5500 Series and ECC module
- Push-button workflows, intuitive user interface, and secure project management
- Optimized mapping and smaller file formats
- Annotated variant reports, numerous charts, and select visualization tools for simple data interpretation
- Graphically driven configuration of multistep analysis workflows
- Ability to save and reuse workflows
- Ability to resume a workflow without repeating completed analyses
- Secure project-based data management specific to your data analysis
- Projects can be stored and data reanalyzed

Data analyses

Complementing LifeScope™ Software features are the following types of data analyses:
- Whole genome sequencing
- Targeted resequencing
- Low frequency variant
- Whole exome sequencing
- Whole transcriptome RNA sequencing
• Small RNA sequencing
• MethylMiner™ mapping
• CHiP-Seq mapping
• Detection of wide range of genomic and RNA variation, including:
  – SNP detection
  – Large and small indel detection
  – Copy number variation detection
  – Inversion detection
  – Fusion transcript detection
  – Exon counting
  – Splice finding

LifeScope™ Software Components

LifeScope™ Software components include:

• LifeScope™ Server Software, the main software architecture that maintains the interaction of the graphical user interface (GUI) and compute engine. The server works with high performance cluster schedulers to maximize the computation demands of the finely tuned data analysis algorithms.

• The research GUI, which enables scientists to perform mapping and the detection of genomic variation from the convenience of a desktop computer, is comprised of a compute engine capable of mapping billions of reads of next-generation sequencing (NGS) data. The GUI provides auto-generated charts and plots for each genomic analysis run within the software.

• The command-line user interface, which gives bioinformaticians the power to customize workflows, and to manipulate every parameter available in each analysis module using the specific LifeScope™ command-line syntax.

• The Admin GUI Portal, which is an administration tool for managing user accounts. Refer to LifeScope™ Genomic Analysis Software User Guide (Part no. 4476537) for information on the Admin Portal.

Workflows

This section describes a standard workflow, a project workflow, and several analysis workflows.

Standard workflow

A standard workflow is a built-in series of commonly used analyses, which correspond to a common biological application. Using one of these workflows enables you to run an analysis with a minimum of setup required.

Standard workflows are provided for each of the analysis types listed above, on page 27.

In LifeScope™ Software, running a standard workflow requires the following steps:

1. Create a project and an analysis.
2. Identify your input data (read set from your 5500 Series sequencing instrument).
3. Identify the reference genome.
4. Choose the workflow to be executed on your data (from the above list).
5. Start and monitor your analysis.
6. View the results of your analysis

Project workflow

Create a project

Create an analysis

Edit an analysis

Run and monitor your analysis

View analysis results

Analysis workflows

In LifeScope™ Software, a workflow is a coordinated series of data analysis steps, with each step dependent only upon the steps that precede it. This section describes the flow of secondary and tertiary analyses in extensive detail, while primary analysis is performed on your 5500 Series sequencing instrument.

Secondary analysis

Secondary analysis begins with mapping. The input files required by mapping modules are eXtensible SeQuence files (XSQ) or converted color-space fasta (CSFASTA) files, and quality value (QV) files. Secondary analysis modules include:

<table>
<thead>
<tr>
<th>Table 1  Secondary analysis modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOLiD® Accuracy Enhancement Tool (SAET)</td>
</tr>
<tr>
<td>Mapping</td>
</tr>
<tr>
<td>Pairing</td>
</tr>
<tr>
<td>Binary Sequence Alignment Map statistics (BAMStats)</td>
</tr>
<tr>
<td>Mark duplicates</td>
</tr>
</tbody>
</table>

Tertiary analysis

The results of mapping and pairing in secondary analysis are used as input for tertiary analysis modules, which include:

<table>
<thead>
<tr>
<th>Table 2  Tertiary analysis modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNP</td>
</tr>
<tr>
<td>CNV</td>
</tr>
<tr>
<td>Small indel</td>
</tr>
</tbody>
</table>
The analysis workflows and the modules they execute are described in the following table.

### Table 2  Tertiary analysis modules

<table>
<thead>
<tr>
<th>Module Type</th>
<th>Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Large indel</td>
<td>Gene and exon count</td>
</tr>
<tr>
<td>Inversion</td>
<td></td>
</tr>
</tbody>
</table>

### Table 3  LifeScope™ Software analysis workflows and modules

<table>
<thead>
<tr>
<th>Workflow</th>
<th>Library type</th>
<th>LifeScope™ Software modules involved</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChIP-Seq</td>
<td>Fragment</td>
<td>Secondary:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Fragment mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Mapping statistics</td>
</tr>
<tr>
<td>Genomic resequencing</td>
<td>Fragment</td>
<td>Secondary:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Mapping statistics</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tertiary:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- SNP Finding</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- CNV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Small indels</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Annotations</td>
</tr>
<tr>
<td>Mate-pair</td>
<td>Secondary:</td>
<td>- Paired mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Mapping statistics</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Mark duplicates</td>
</tr>
<tr>
<td></td>
<td>Tertiary:</td>
<td>- SNP Finding</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- CNV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Inversions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Large indels</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Small indels</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Annotations</td>
</tr>
<tr>
<td>Paired-end</td>
<td>Secondary:</td>
<td>- Paired mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Mapping statistics</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Mark duplicates</td>
</tr>
<tr>
<td></td>
<td>Tertiary:</td>
<td>- SNP Finding</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Inversions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Large indels</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- CNV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Small indels</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Annotations</td>
</tr>
</tbody>
</table>
## Table 3  LifeScope™ Software analysis workflows and modules *(continued)*

<table>
<thead>
<tr>
<th>Workflow</th>
<th>Library type</th>
<th>LifeScope™ Software modules involved</th>
</tr>
</thead>
<tbody>
<tr>
<td>MethylMiner™</td>
<td>Fragment</td>
<td>Secondary:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Paired mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Mapping statistics</td>
</tr>
<tr>
<td>Paired-end</td>
<td>Secondary</td>
<td>Secondary:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fragment mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Mapping statistics</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Mark duplicates</td>
</tr>
<tr>
<td>Small RNA</td>
<td>Small RNA</td>
<td>Secondary:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Small RNA mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tertiary:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Small RNA count</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Small RNA coverage</td>
</tr>
<tr>
<td>Targeted resequencing</td>
<td>Fragment</td>
<td>Secondary:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SAET</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fragment mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Mapping statistics</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tertiary:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Enrichment</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SNP Finding</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Small indels</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Annotations</td>
</tr>
<tr>
<td>Paired-end</td>
<td>Secondary</td>
<td>Secondary:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SAET</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Paired mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Mapping statistics</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Mark duplicates</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tertiary:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Enrichment</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SNP Finding</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Small indels</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Annotations</td>
</tr>
</tbody>
</table>
### Table 3  LifeScope™ Software analysis workflows and modules (continued)

<table>
<thead>
<tr>
<th>Workflow</th>
<th>Library type</th>
<th>LifeScope™ Software modules involved</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Low frequency variant</strong></td>
<td>Fragment</td>
<td>Secondary:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fragment mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Mapping statistics</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tertiary:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Enrichment</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SNP Finding</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Annotations</td>
</tr>
<tr>
<td><strong>Paired-end</strong></td>
<td>Secondary:</td>
<td>• Paired mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Mapping statistics</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Mark duplicates</td>
</tr>
<tr>
<td><strong>Whole transcriptome</strong></td>
<td>Secondary:</td>
<td>•WT splice junction extractor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>•WT fragment mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Mapping statistics</td>
</tr>
<tr>
<td></td>
<td>Tertiary:</td>
<td>•WT counts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>•WT coverage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>•Splice finder</td>
</tr>
<tr>
<td><strong>Paired-end</strong></td>
<td>Secondary:</td>
<td>•WT exon sequence extractor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>•WT splice junction extractor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>•WT paired-end mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Mapping statistics</td>
</tr>
<tr>
<td></td>
<td>Tertiary:</td>
<td>•Splice finder</td>
</tr>
<tr>
<td></td>
<td></td>
<td>•WT counts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>•WT coverage</td>
</tr>
</tbody>
</table>
The illustration below shows the data analysis modules used in predetermined workflows.
Primary and secondary file types

The following table lists the files used in secondary and tertiary analysis.

Table 4  File types used in LifeScope™ Software analyses

<table>
<thead>
<tr>
<th>Analysis type</th>
<th>File type</th>
<th>File name extension</th>
<th>File content</th>
<th>Used in . . .</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary</td>
<td>Raw reads file</td>
<td>*.xsq</td>
<td>Extensible sequence file (XSQ)</td>
<td>Secondary analysis</td>
</tr>
<tr>
<td>Secondary</td>
<td>BAM file</td>
<td>*.bam</td>
<td>Binary Alignment sequence Map (BAM), a generic file format used to store large numbers of nucleotide sequence alignments</td>
<td>Tertiary analysis</td>
</tr>
</tbody>
</table>

Input and output file formats

All analysis modules use specific input files and produce specific output files. The input and output file requirements vary, depending on the type of analysis that you want to perform. The following table provides the names of the input and output file types.

Table 5  Input and output file formats

<table>
<thead>
<tr>
<th>Software or bioinformatics tool</th>
<th>Input file types</th>
<th>Output file type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mapping tool</td>
<td>XSQ</td>
<td>BAM</td>
</tr>
<tr>
<td>MaToBAM tool</td>
<td>MA</td>
<td>Converts a MA file to a BAM file.</td>
</tr>
<tr>
<td>Mapping statistics (BAMStats)</td>
<td>BAM</td>
<td>Position error, probe error, multiple chart and statistics files</td>
</tr>
<tr>
<td>Small indel</td>
<td>BAM</td>
<td>GFFv3</td>
</tr>
<tr>
<td>diBayes</td>
<td>BAM</td>
<td>GFFv3, CSFASTA, consensus_calls</td>
</tr>
<tr>
<td>CNV</td>
<td>BAM</td>
<td>GFFv3, *.out</td>
</tr>
<tr>
<td>Large indel</td>
<td>BAM</td>
<td>GFFv3</td>
</tr>
<tr>
<td>Inversion</td>
<td>BAM</td>
<td>GFFv3, text</td>
</tr>
<tr>
<td>WT mapping</td>
<td>XSQ, FASTA, filter reference FASTA, WT GTF reference</td>
<td>BAM</td>
</tr>
<tr>
<td>WT counts</td>
<td>BAM</td>
<td>GTF</td>
</tr>
<tr>
<td>WT coverage</td>
<td>BAM</td>
<td>WIG</td>
</tr>
<tr>
<td>WT splice junction</td>
<td>BAM</td>
<td>WIG</td>
</tr>
<tr>
<td>Small RNA mapping</td>
<td>XSQ, GTF reference</td>
<td>BAM</td>
</tr>
<tr>
<td>Small RNA counts</td>
<td>BAM</td>
<td>GTF</td>
</tr>
<tr>
<td>Small RNA coverage</td>
<td>BAM</td>
<td>WIG</td>
</tr>
</tbody>
</table>
PART II
Installation
LifeScope™ Genomic Analysis
Software Installation

This chapter covers:

- Overview .................................................. 37
- Prerequisites ............................................... 37
- LifeScope™ Software Administration ....................... 39
- Installation workflow overview ............................. 41
- Copy data drive content ................................... 41
- Download LifeScope™ Software ........................... 42
- Install LifeScope™ Software ............................... 42
- Set the LifeScope™ Software environment ............... 46
- Start LifeScope™ Software servers and check servers’ status 47
- Download documentation and additional resources ....... 47

Overview

This chapter is intended for use by the LifeScope™ Software administrator. The chapter describes the LifeScope™ Genomic Analysis Software installation procedure, and system software and hardware requirements.

Prerequisites

Some procedures in this chapter require that you:

- Know the Linux® operating system of the cluster on which LifeScope™ Software is to be installed
- Know the Internet Protocol (IP) address of the cluster
- Have a login (ID) on the cluster
- Portions of the install require root access or your system administrator's assistance
- Know how to:
  - Navigate to directories in a Linux environment
  - Edit and save files in a text editor
  - Run Linux shell scripts
  - Run basic Linux commands such as chmod, ps, pwd, cd, echo, grep, and other commands
Chapter 2  LifeScope™ Genomic Analysis Software Installation

Prerequisites

Workstation requirements

Workstations are shipped with CentOS 5.7 and are ready to run LifeScope™ Software. Customer upgrade of the CentOS software is not supported.

**WARNING!** LifeScope™ Software does not operate properly on workstations running operating systems other than CentOS 5.7. CentOS 6.0 is not supported.

Cluster hardware requirements

To successfully install and run the LifeScope™ Software application, we recommend that your cluster meets the hardware requirements listed in Table 6.

<table>
<thead>
<tr>
<th>Table 6  Cluster requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Speed</td>
</tr>
<tr>
<td>Cores</td>
</tr>
<tr>
<td>Memory</td>
</tr>
<tr>
<td>Disk Space</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

‡ You can use any type of shared storage to meet this requirement.

Note: Distributed parallel file systems such as Lustre, GlusterFS, GPFS require additional tuning to be compatible with LifeScope™ Software. For example, we have encountered instances of job failures caused by subpar small file performance.

Server requirements

Before you install LifeScope™ Genomic Analysis Software, be sure that your cluster meets the following requirements:

- LifeScope™ Genomic Analysis Software supports only the following distributions, and only on 64-bit platforms:
  - Red Hat Enterprise Linux 4.7 through 5.7
  - CentOS 4.7 through 5.7
  **Note:** LifeScope™ Software does not support CentOS 6.0 or RHEL 6.0.

- PBS/TORQUE v2.3+, SGE v6.2+, or Platform LSF 7 Update 6.
  **Note:** LifeScope™ Software requires a job submission queue on the cluster. The LifeScope™ Software installer does not create this queue on the cluster.

  **Note:** If you use SGE, you must create a symmetric multiprocessing (SMP) parallel environment.

  **Note:** If you use LSF, it is highly recommended that LifeScope™ Software jobs are non-preemptable by LSF.

Before you install LifeScope™ Software on the head node, pre-install compatible versions of these software packages on all compute nodes:

- Python 2.3 (or later version)
- Perl v.5.8.5 (or later version)
Note: Inversion module works with Perl versions up through 5.8.8. The most recent version of Perl has a backwards compatibility issue which causes the inversion module to fail.

GCC Compiler

Ensure that the GNU Compiler Collection (GCC) compiler v3.4.6 or v4.1.2 installed and configured. While LifeScope™ Software does not use the compiler itself, having the compiler installed and configured ensures that the necessary dynamically linked libraries are available.

Check firewall restrictions

Ensure that there is no firewall restriction on the host running the LifeScope™ Software server. The LifeScope™ Software UI clients and command shell clients access the LifeScope™ Software server on TCP/IP port 9998 by default, although this port can be changed during the installation. Consult your system administrator to check the firewall access for LifeScope™ Software users.

Check DNS hostname

The DNS fully-qualified domain name and the local Linux hostname must match. Also, the forward and reverse lookup on the DNS against the hostname and its public-facing IP address must be consistent. Consult the system administrator to check this requirement.

Client machines

Java v1.6 (or higher) is required on client machines.

For Mac users who do have Java v1.6, follow these steps to update:

1. Use the Mac Software Update (apple › Software Update...) to install.
2. Once you have installed, go Applications › Utilities › Java Preferences.
3. Select only Java SE 6.0.

Browser Requirements

LifeScope™ Software supports these browsers:

- Internet Explorer® versions 6 and 7
- Mozilla® Firefox 3.0.1

LifeScope™ Software Administration

This section uses the acronyms in this list to describe installation roles and locations:

- SA – The Linux system administrator for the cluster. The administrator’s root access is required for user management and copying the data drive.
- LSA – The LifeScope™ Software administrator. The installation instructions recommend creating a new user account for the LSA. The LSA must run the installation script.
- LSU – LifeScope™ Software users. These users do not have admin permissions.
- LSBF – The LifeScope™ Software Binaries Folder. The LSBF is the installation directory.
Chapter 2  LifeScope™ Genomic Analysis Software Installation

LifeScope™ Software Administration

LifeScope™ Genomic Analysis Software Command Shell User Guide

-LSDF – The LifeScope™ Software Data Folder. The LSDF is a location for LifeScope™ Software repositories, resources, and other files. The LSDF is also used temporarily to hold the installer and the installation tar ball.

To install or run LifeScope™ Software, one person must be designated as the master LifeScope™ Software user. The master user also becomes the default LifeScope™ Software Administrator (LSA). The LSA and regular LifeScope™ Software Users (LSUs) must share a common UNIX® user group.

We recommend that the LSA not share the same user account as the UNIX System Administrator (SA), because of a requirement that all LSUs be members of the LSA’s primary UNIX user group. The SA’s primary UNIX user group is usually root. Non-administrator users typically are not allowed to be members of the root user group.

We recommend creating a new Linux user account, lifescope, for the LSA. The primary UNIX user group for the lifescope user should be the same group as most LifeScope™ Software Users (for example, the group users). The Linux account for the LSA should have this shared group as its primary group when the LSA starts the LifeScope™ Software server.

After installation, LifeScope™ Software files and folders are owned by the LSA. All application jobs and processes are submitted by the LSA.

If you are not the LSA or if an LSA does not exist, please contact your System Administrator to have the master user and user group created. You must be logged in as the LSA to install LifeScope™ Software.

Identify two shared folders for holding the software and data in a suitably common location, accessible to all LifeScope™ Software users. Both locations should be accessible (mounted) from the compute nodes.

- Identify the LifeScope™ Software Binaries Folder (LSBF) to install LifeScope™ Software. This folder should be writable to LSA and readable to LSUs. Typically you may choose /opt/lifescope. If the LSBF does not exist, create it with appropriate permissions.

- Identify the location of the LifeScope™ Software Data Folder (LSDF) to hold all LifeScope™ Software resources, repositories, pre-installation tar files, installer, examples, test data, and other files. This location should have read and write permissions for the LSA and all LSUs. This location also should be sufficiently large to hold the initial data and the results of future analysis. Typically you may choose /share/lifescope. If the LSDF does not exist, create it with appropriate permissions.

Make sure that you have the appropriate read and write permissions for the LSBF and LSDF folders, before proceeding with the installation.
Installation workflow overview

IT System Administrator (SA)

Copy data drive content [page 41]

Installer (LSA)

Download LifeScope™ Genomic Analysis Software [page 42]

Install LifeScope™ Software [page 42]

Download documentation and additional resources [page 47]

Copy data drive content

(System Administrator) Copy the data drive content to the LifeScope™ Software Data Folder (LSDF) on the LifeScope™ Software server, and change owner:group to those of the LifeScope™ Software Administrator (LSA).

The data drive’s file system is ext3, which typically is supported only on Linux.

The drive content includes the following:

- Files for use with hg18 and hg19 genomes:
  - Hash tables for the main mapping schemes
  - Mappability files for use with the CNV analysis module
- LifeScope™ Software reference repository files:
  - Hg18 and hg19 reference files and filter reference files
  - Genomic annotation files (dbSNP)
  - Annotation files from the miRNA database
- Data files for installation verification

The reference files downloaded in this step become the reference repository for your LifeScope™ Software. The location of the reference files (the referenceData directory) is required during installation.

IMPORTANT! If you add your own reference files to the reference repository, you must follow the instructions in “Add new reference files” and “Prepare reference files” on page 550.
Chapter 2  LifeScope™ Genomic Analysis Software Installation

Download LifeScope™ Software

IMPORTANT! Do not download the installer and tar file to LifeScope™ Software Binary Folder (LSBF), which is the LifeScope™ Software installation directory. The directories LSBF and LSDF are described in “LifeScope™ Software Administration” on page 39.

1. (LSA) Visit the LifeScope™ Software project page. Refer to the release notes for this location.
2. On the download page, you must read and accept the LifeScope™ Software End User License Agreement.
3. Download LifeScope™ Software .tar.gz file (for example, LifeScope-2.5.1*.tar.gz) to the LifeScope™ Software Data Folder (LSDF).
4. Download the associated installer (install.sh) to the LifeScope™ Software Data Folder (LSDF).

Install LifeScope™ Software

This procedure provides typical, general instructions for installing LifeScope™ Software. The actual procedures for your site might vary, depending on the configuration of the LifeScope™ Software cluster and the installation options you select.

WARNING! LifeScope™ Software must NOT be installed as root.

Hardware configuration

This section explains the three types of installation based on hardware configuration. You must select your configuration when prompted by the installer in “Select the type of installation:”, step 6 on page 44.

- **Standalone Workstation** – LifeScope™ Software is installed completely on the local workstation. There is no access to a cluster. Resource management software is still required on the workstation.
- **Cluster** – LifeScope™ Software is installed on a cluster with a head node, a set of compute nodes, shared storage, node local storage, and resource management software, as described in “Cluster hardware requirements” on page 38.

Authentication realm

This section explains the authentication realms supported by LifeScope™ Software. You must supply your realm type when prompted by the installer in “Select an authentication realm to be used with LifeScope™ Software:”, step 14 on page 45.

- **LifeScope** – This authentication realm refers to the LifeScope™ Software application user database. If this realm is chosen, user passwords are maintained in the software database. Passwords are stored as one-way MD5 digests for security reasons. Users must be manually created in this realm. There is no relationship between the application users and Linux® host system users. User account management can be done by the LSA using the LifeScope™ Software Admin module (included in this installation).
• **LDAP** – This realm refers to an LDAP-compliant authentication server (OpenLDAP, Active Directory, etc.). If this realm is chosen, user passwords are *not* stored in LifeScope™ Software. User credentials are authenticated against the configured LDAP server. Users need not be manually created in LifeScope™ Software with this realm (though they may be). A valid LDAP server address and LDAP Bind DN must be provided in the configuration. User account management can be done by the LSA using the LifeScope™ Software Admin module (included in this installation).

• **Host** – This realm refers to the Linux host machine on which the LifeScope™ Software server is running. If this realm is chosen, user passwords are *not* stored in LifeScope™ Software. User credentials are authenticated against the local machine using SSH (therefore uses whatever authentication scheme SSH uses, for example, NIS, /etc/passwd). There is a one-to-one relationship between the application users and Linux host system users. Users need not be manually created in the LifeScope™ Software Admin Portal with this realm.

**Note:** If you change the authentication realm after installation, you must stop and restart the LifeScope™ Software server.

**Note:** Changing a realm does not remove users from the previous realm. However, users might not be able to login in the new realm, unless they both exist in the new realm and are enabled in the new realm.

**Installation instructions**

1. (LSA) Go to the downloaded folder LSDF:
   ```bash
cd /path/to/<LSDF>
```
2. Run this UNIX® command:
   ```bash
   chmod +x install.sh
   ```
   to make the downloaded install.sh file executable.
3. Run the installer:
   ```bash
   ./install.sh
   ```
   The welcome banner appears:
   
   `******************************************************************
   This is the LifeScope™ Genomic Analysis Software installation program
   ******************************************************************`

4. Installation options include:
   1) Install the LifeScope™ Genomic Analysis Software.
   2) Upgrade your current installation of LifeScope™ Genomic Analysis Software.
   
   The following options require successful installation of LifeScope:
   `------------------------------------------------------------------`
   3) Re-configure the Applied Biosystems LifeScope™ Genomic Analysis Software.
   4) Exit this application.
Please select the operation you would like to perform (Q/q to quit) [1]:

To install LifeScope™ Software, type 1.

5. At the prompt, type y to begin installation.

6. Select the type of installation:
   1) Standalone Workstation Server (default)
   2) Cluster

   Review “Hardware configuration” on page 42 to determine your installation type.

7. Enter the location where LifeScope™ Software is to be installed. The default directory is /opt/lifescope.

   Accept the default or change the directory. Provide the location you have chosen for the LifeScope™ Software Binaries Folder (LSBF):

   /path/to/<LSBF>

   If the directory you enter already exists, you are prompted “Do you still want to use this location?”

IMPORTANT! LifeScope™ Software Binary Folder (LSBF), which is the LifeScope™ Software installation directory, can not be the same as the LifeScope™ Software Data Folder (LSDF), the directory containing the installer and the tar ball. The directories LSBF and LSDF are described in “LifeScope™ Software Administration” on page 39.

8. Enter the location for reference files. Provide the destination location of the data drive contents (from the “Copy data drive content” step on page 41):

   /path/to/<LSDF>/referenceData

   Note: LifeScope reference files are required for full functionality of LifeScope™ Software. The default directory is /data/results/referenceData.

9. Enter the location for reads. This entry sets the location of the reads repository. You may choose

   /path/to/<LSDF>/reads

   The default directory is /data/results/reads.

10. Enter the location for projects. This entry sets the location of the projects repository. You may typically choose

    /path/to/<LSDF>/projects

    The default directory is /data/results/projects.

11. Enter the location for Binary Alignment Map (BAM) file repository. You may choose

    /path/to/<LSDF>/bams

    The default directory is /data/results/bams.

12. Enter the location for the scratch location for LifeScope. The default directory is /scratch.
13. Enter the IP address or Fully Qualified Domain Name (FQDN) for the LifeScope™ Software server, for example: `192.168.1.27`. The address of the system where LifeScope™ Software is installed is required to successfully access and run the software. The system address is the IP address or Fully Qualified Domain Name (FQDN) of this system or head node.

**Note:** Do not use the name `localhost` or IP address `127.0.0.1`. The address should be the public-facing name or IP address of the LifeScope™ Software server. The LSUs access the software using this name or IP address. Typically, the address is the domain name system (DNS) name of the server.

Test the connection to the IP address or FQDN to verify a connection to the system. Enter the Web URL port address to be used to access the LifeScope Server. Default is 9998.

14. Select an authentication realm to be used with LifeScope™ Software:
   1. LifeScope™ Software (default)
   2. Lightweight Directory Access Protocol (LDAP)
   3. Host

Review “**Authentication realm**” on page 42 to determine your realm type.

**Note:** If you change the authentication realm after installation, you must stop and restart the LifeScope™ Software server.

**Note:** Changing a realm does not remove users from the previous realm. However, users might not be able to login in the new realm, unless they both exist in the new realm and are enabled in the new realm.

15. Select the cluster resource manager to be used by LifeScope™ Software:
   1. PBS/Torque (default)
   2. SGE
   3. LSF

16. Enter the cluster-resource-manager-job-submission-queue-name to be used by LifeScope, for example, `lifescope`.

17. Enter the number of nodes available on the cluster. Default is 4.

18. Enter the number of cores for each compute node. Default is 8.

19. Enter the memory size (in GB) for each compute node. Default is 22.

**IMPORTANT!** Allow 2 GB for the compute nodes’ OS. For example, for compute nodes with 24 GB, enter 22; for compute nodes with 48 GB, enter 46.
Chapter 2    LifeScope™ Genomic Analysis Software Installation

Set the LifeScope™ Software environment

Continue installing LifeScope™ Software

1. At the prompt “Do you want to change any of these entries?” type n to accept the entries.
   The selected LifeScope™ Software configurations are saved.

2. At the prompt, “Do you want to continue installing the LifeScope package?”
   type y to continue the installation.

Refer to the LifeScope™ Genomic Analysis Software User Guide (Part no. 4476537)
administration appendix for instructions to make LifeScope™ Software available to users. The appendix includes this topic:

• Enabling and disabling LifeScope™ Software users.

LifeScope™ Software installation is completed. However, you are recommended to do at least one installation verification step. These steps are optional but confirm that the software is functioning properly and that the LifeScope™ Software Data Folder (LSDF) is properly configured. The optional installation verification steps are:

• The demo analyses
• The performance verification tests

See Chapter 3, “Verify the LifeScope™ Genomic Analysis Software Installation” on page 49 for instructions on optionally running example analysis workflows and optionally performing verification tests.

Set the LifeScope™ Software environment

Follow the instructions in “Set the LifeScope™ Software environment” on page 60 to set your PATH variable.
Start LifeScope™ Software servers and check servers’ status

This section describes starting the LifeScope™ Software servers and checking their status. Commands described in this section are run in the Linux shell (not in the LifeScope™ Software command shell).

Only the LifeScope™ Software administrator (LSA) must start the LifeScope™ Software server.

Start the LifeScope™ Software server

The command to start the LifeScope™ Software server is:
```
lscope-server.sh start
```

The server by default uses port 9998, but the default port number might be changed during installation. To run on a different port, use the following version:
```
lscope-server.sh start -p port
```

If you start the server on a port other than the default 9998, you must check that the firewall is also enabled for the new port number. See “Check firewall restrictions” on page 39.

If the port is busy, the server start fails.

If the server is already running, a second instance is not started. The server issues the following message:

```
Web server already running (pid=nnnnn) on port nnnn
```

You must start the LifeScope™ Software server every time the system undergoes a reboot.

Check the LifeScope™ Software server status

The following command checks if the LifeScope™ Software server is running:
```
ps -ef | grep WebServer
```

If the LifeScope™ Software server is not running, the Linux command prompt returns (with no output). Example output when the web server is running is:
```
lifescope 23521     1  0 13:26 pts/1    00:00:05 java -server -Xmx3072m -XX:PermSize=512m -XX:MaxPermSize=512m -XX:+UseParallelGC com.lifetechnologies.bioscope.webservice.WebServer -b /opt/lifescope/lifescope
```

Download documentation and additional resources

To ensure you have the latest version of LifeScope™ Software, please visit the LifeScope™ Software project page and look for the Download option. You also find user documentation and the latest information and technical resources. The full resources package includes pre-installation scripts, smaller data files, the installation wizard, and performance verification scripts. Additional links and online support resources are also available at this site.
You can optionally install the Integrative Genomics Viewer (IGV), shown on page 48, available from the Broad Institute. The IGV is a visualization tool for interactive exploration of large, integrated datasets. It directly reads BAM files, which enables you to easily view and inspect alignments against the genome.

For more information about the IGV, go to www.broadinstitute.org/igv/.
CHAPTER 3

Verify the LifeScope™ Genomic Analysis Software Installation

This chapter covers:

- Overview .......................................................... 49
- Workflow ........................................................... 50
- Verify the installation ............................................ 50
- (Optional) Download and install demos files .............. 51
- (Optional) Run demo analyses .................................. 51
- (Optional) Download and install performance verification scripts .... 52
- (Optional) Run the performance verification test .......... 52
- Create user accounts .............................................. 53
- Distribute the URL and user credentials ....................... 54
- Maintenance ........................................................ 55

Overview

This chapter describes procedures and tests that the LifeScope™ Software administrators should perform after the software has been installed. LifeScope™ Software users may skip this chapter.
Verify the installation

(LSA) Two mechanisms are provided to verify the installation:

- **Demos** – A set of individual analysis modules with small data sets. The demos take approximately 30 minutes or less to complete.
- **Performance verification** – A realistic computation scenario that takes approximately 24 hours to complete.

**IMPORTANT!** You are strongly recommended to run one or both of the verification procedures. The procedures are listed as optional because you are not required to run both of them. Run at least one of the optional verification procedures.
You may run the demo analyses and performance verification scripts as the LifeScope™ Software administrator or as a LifeScope™ Software user. See “Create user accounts” on page 53 create a LifeScope™ Software user account.

(Optional) Download and install demos files

(Optional) Download and install demos files

(LSA) This step is required in order to run the optional sample analyses, “(Optional) Run demo analyses” on page 51.

Example scripts and data for demo LifeScope™ Software analyses are available from the LifeScope™ Software product page (refer to the release notes for the project page location).

Included in this download are the required PLN and INI files for running individual LifeScope™ Software analysis modules, and scripts to run the analyses in the LifeScope™ Software command shell.

Follow these steps to install the demo files:

1. Download the .tar.gz file to the LifeScope™ Software Data Folder (LSDF).
2. Go to the LSDF directory and untar the .tar.gz file:
   
   ```
   cd <LSDF>
   tar -xzf exampleWorkflows.tar.gz
   ```

   **Note:** The demo analyses are not recommended as the first experience for users new to LifeScope™ Software. The standard workflows (Chapter 7, Run a Standard Workflow Analysis) are recommended for new users to learn LifeScope™ Software command-line analyses.

(Optional) Run demo analyses

(Optional) Run demo analyses

(LSA) Run demo analyses, which takes approximately 30 minutes.

These instructions require the following (in addition to the software installation):

- The default reference repository, which includes hg18 and hg19 assemblies. The reference repository is created during the “Copy data drive content” on page 41 and “Install LifeScope™ Software” on page 42 procedures.
- The demos files and sample data installed during the “(Optional) Download and install demos files” on page 51.

Follow these steps to execute the demos in one run:

1. Follow the instructions in “Set the LifeScope™ Software environment” on page 60 to set the LifeScope™ Software environment.
2. Go to the demos directory:
   
   ```
   cd <LSDF>/examples/demos
   ```
3. Execute this command in the Linux shell:
   
   ```
   ./runall.sh -u username
   ```

   where `username` must be for the LifeScope™ Software administrator or for a valid LifeScope™ Software user account. Provide the password when prompted.
This script runs each demo one at a time.

**Check results**

The results files are generated in the projects repository, at a location which is set during installation. Its default location is at `<LSDF>/projects`. Within the projects repository directory, the demos results are generated at the following directories:

```
username/examples/analysis_name/outputs/sample_name
```

Replace `username` with the user name used to run `runAll.sh`. The default is `lifescope`, the LSA account. The string `examples` is the project name used by all demos. `analysis_name` is the name of the analysis module used by a demo, and is for example, `cnv`, `dibayes`, `enrichment`, etc. Each `analysis_name` folder has an `outputs` subfolder. The final folder, `sample_name`, is determined by the sample in the input data.

Each of the demo analyses creates a log file named `summary.log` in the `lifescope/examples/analysis_name` directory. Check for a success message in the last line of each log file.

The logs and output files are overwritten on subsequent runs executed by the same user.

**(Optional) Download and install performance verification scripts**

(LSA) This step is required in order to run the optional performance verification tests, “(Optional) Run the performance verification test” on page 52.

The performance verification test data file are in the LSDF directory, from the “Copy data drive content” step on page 41.

The performance verification test scripts are available from the LifeScope™ Software product page (refer to the release notes for the project page location). Download the file:

```
performanceVerificationTestScripts.tar.gz
```

Follow these steps to install the performance verification test scripts:

1. Download the file `performanceVerificationTestScripts.tar.gz` to the LifeScope™ Software Data Folder (LSDF).
2. Go to the LSDF directory and untar the `.tar.gz` file:
   ```
   cd <LSDF>
   tar -xzf performanceVerificationTestScripts.tar.gz
   ```

**(Optional) Run the performance verification test**

(LSA) Run a performance verification test, which takes approximately 24 hours.

The performance verification data files are installed during the “Copy data drive content” step on page 41. The performance verification test scripts are installed during the “(Optional) Download and install performance verification scripts” step on page 52.

Follow these steps to execute the performance verification test:
Chapter 3 Verify the LifeScope™ Genomic Analysis Software Installation

Create user accounts

1. Follow the instructions in “Set the LifeScope™ Software environment” on page 60 to set the LifeScope™ Software environment.

2. Confirm that the performance verification test data is installed at this directory:
   \(<LSDF>/performanceVerificationData\)

3. Go to the performance verification test scripts directory:
   \(cd \(<LSDF>/performanceVerificationTestScripts\)\)

4. Execute this command in the Linux shell:
   \(.\/runAll.sh -u \textit{username}\)
   where \textit{username} must be for the LifeScope™ Software administrator or for a valid LifeScope™ Software user account. Provide the password when prompted.

Monitor progress

Monitor the progress of your run by checking the output log files and also through job status commands such as qstat.

Check results

The results files are generated in the projects repository, at a location which is set during installation. Its default location is at \(<LSDF>/projects\). Within the projects repository directory, the performance verification results are generated at the following results directories:

\(\textit{username}/performanceVerification/\textit{analysis_name}/outputs\)

Replace \textit{username} with the user name used to run \textit{runAll.sh}. The default is lifescience, the LSA account. The string \textit{performanceVerification} is the project name used by the performance verification test. The analysis names used by \textit{runAll.sh} are \textit{genomicResequencing.20x}, \textit{genomicResequencing.2x}, and \textit{wholeTranscriptomePE}. The outputs subfolder name is fixed.

The performance verification tests creates a log file in the \textit{analysis_name} directory. Check for a success message in the last line of each log file.

The results files are overwritten on subsequent runs executed by the same user.

Create user accounts

(LSA or SA) Create user accounts.

For instructions in this section that require using the LifeScope™ Software Admin Portal, refer to the administration appendix in the LifeScope™ Genomic Analysis Software User Guide (Part no. 4476537).

The method for user account creation depends on the authentication realm for your installation:

- **LifeScope realm** – (LSA) The LifeScope™ Software administrator creates LifeScope™ Software user accounts using LifeScope™ Software. The LifeScope™ Software Admin Portal is recommended for user administration.
  LifeScope™ Software shell commands for user administration are described in Table 13 on page 81, in Chapter 6, Run a Command Shell Analysis. Although shell commands include some user administration commands, the Admin Portal is recommended for general user administration.
Chapter 3 Verify the LifeScope™ Genomic Analysis Software Installation

Distribute the URL and user credentials

- **LDAP realm** – (SA and LSA) The Linux system administrator creates user accounts on the LDAP server. The LifeScope™ Software administrator must use the LifeScope™ Software Admin Portal to enable each LifeScope™ Software user.

- **Host realm** – (SA) The Linux system administrator creates Linux user accounts on the local operating system.

**Distribute the URL and user credentials**

(LSA) The LifeScope™ Software URL follows this pattern:

```
http://fqdn of host:<port number>/LifeScope.html
```

For example, for an installation on host `orange.intranet.company.com`, using the default port `9998` for the LifeScope™ Software server, the URL is:

```
http://orange.intranet.company.com:9998/LifeScope.html
```

With the LifeScope™ Software servers running, verify the LifeScope™ Software URL on a supported browser. The following page appears:

![LifeScope™ Genomic Analysis Software](image)

The **Analysis Portal** enables you to analyze and manage data results from your Life Technologies sequencing instrument.

![Analysis Portal](image)

The **Admin Portal** enables you to set up accounts and manage licenses.

![Admin Portal](image)

Notify LifeScope™ Software users (LSUs) of the LifeScope™ Software URL.

If new user accounts are created for LifeScope™ Software users, notify them of the username and password credentials required to access the LifeScope™ Software.
**Maintenance**

**Stop the LifeScope™ Software server**

The following cautions apply to stopping the LifeScope™ Software server:
- The LifeScope™ Software server should only be stopped for required maintenance.
- Only the LSA account can stop the LifeScope™ Software server.
- All running analyses are terminated when you stop the LifeScope™ Software server.

Follow these steps to stop the LifeScope™ Software server:
1. If you have not already done so, set the LifeScope™ Software environment. See “Set the LifeScope™ Software environment” on page 60.
2. Check for LifeScope™ Software jobs running on the cluster with a command such as `qstat`. Notify LifeScope™ Software users that their jobs are affected.
3. Execute this Linux command:
   ```bash
   lscope-server.sh stop
   ```

**Upgrade**

Upgrading to a new version of LifeScope™ Software does not require you to uninstall your existing version of LifeScope™ Software.

Follow the installation instructions for the new version of LifeScope™ Software. This procedure installs the new version in the `<LSBF>` alongside your existing version, and modifies the soft link `<LSBF>/lifescope` to point to the new version. The older version of LifeScope™ Software is not deleted during installation.

Once the new installation has been verified, you may remove the older installation at your discretion.

**Uninstall**

This section describes the steps for the LifeScope™ Software administrator (LSA) to uninstall LifeScope™ Software. If you use an authentication realm that requires the IT system administrator (SA) to create user accounts, the SA deletes those user accounts as part of the uninstall process.

Follow these steps to uninstall LifeScope™ Software:
1. (LSA or SA) Backup any required files, from the `<LSDF>` directories (these directories hold the LifeScope™ Software repositories):
   - Reads files (XSQ and BAM)
   - Reference files
   - Output files
   - Data
   - Projects and analyses
   Notify LifeScope™ Software users to backup any files they might need from the `<LSDF>` directories.

2. (LSA or SA) Delete the LifeScope™ Software installation directory, with this Linux command:
   ```bash
   /bin/rm -rf <LSBF>/lifescope
   ```
3. *(Optional) (LSA or SA) Delete the LifeScope™ Software data folder, with this Linux command:

   /bin/rm -rf <LSDF>

4. *(Optional) (SA) If you use host or LDAP authentication, delete the user accounts you created for LifeScope™ Software users.

   *(Optional) (LSA or SA) If you modified any bash profile scripts in /etc or in the local home directory to change the PATH variable, remove the lifescope location entries in the PATH variable.
PART III
Getting Started
Overview

This chapter is for LifeScope™ Genomic Analysis Software users. Some procedures described in this section require that you:

- Know the IP address of the LifeScope™ Software cluster
- Have a login ID on the LifeScope™ Software cluster
- Know how to:
  - Navigate to directories in a Linux environment
  - Edit and save files in a text editor
  - Run Linux shell scripts
  - Run Linux commands such as `ls`, `pwd`, `cd`, `echo`, `grep`, and other common utilities

This chapter describes the following procedures:

- Verify your browser version
- Set your shell environment to use LifeScope™ Software

  **Note:** In some installations, the LifeScope™ Software system administrator sets your environment for you.

- Check whether the LifeScope™ Software servers are running
Verify the browser version

LifeScope™ Genomic Analysis Software supports:
- Internet Explorer® versions 6 and 7
- Mozilla® Firefox 3.0.1

Set the LifeScope™ Software environment

To set your environment to run LifeScope™ Software, add the LifeScope™ Software bin directory to your PATH and export the new PATH variable. Enter this command every time you open a Linux shell window to run LifeScope™ Software:

```
export PATH=<installdir>/bin:$PATH
```

where `installdir` is the LifeScope™ Software installation directory.

The `installdir` is the `<LSBF>/lifescope` directory defined in the administrator instructions ("LifeScope™ Software Administration" on page 39).

If BioScope™ Software is not used on the same machine, you can make your PATH setting permanent by modifying your local `.bashrc` file or the global `/etc/bashrc` files. Contact your system administrator for more information.

BioScope™ Software users’ PATH variable

If you install LifeScope™ Software on a machine that also has BioScope™ Software installed and the BioScope™ Software is actively being used, you must follow these instructions:

1. LifeScope™ Software users must **not** use their shell profile script to update the PATH variable with the LifeScope™ Software bin directory. If the LifeScope™ Software bin directory is added to the PATH, BioScope™ Software cannot run correctly.
2. Run BioScope™ Software and LifeScope™ Software in different Linux® shell windows.
3. Set and export your PATH variable **every** time you open a Linux shell window to run LifeScope™ Software server or client software.

BioScope™ Software profile scripts

Some customers with early versions of BioScope™ Software use a profile script to set the BIOSCOPEROOT environment variable for all users. If you have the file `/etc/profile.d/bioscope_profile.sh` on your system, you must delete this file, or remove it from the `/etc/profile.d` directory, for LifeScope™ Software to run properly.

Clusters running both versions

If you currently run BioScope™ Software on your cluster, and need to support both BioScope™ and LifeScope™ users on the same cluster, follow these steps:

- Remove the file `bioscope_profile.sh` from the `/etc/profile.d` directory (the script cannot be in any `/etc/init` directory). Having this script in a system init directory prevents LifeScope™ Software from running properly. (Save the script in your own private location in order to configure the environment for BioScope™ Software users, as described below.)
• Users who run LifeScope™ Software must not have the environment variable BIOSCOPEROOT set and must not have the BioScope™ Software /bin directory in their PATH.
• Users who still run BioScope™ Software should configure their environment by copying the content of the file /etc/profile.d/bioscope_profile.sh into their own shell setup file (such as .bash_profile).

Check LifeScope™ Software servers

This section describes checking the status of LifeScope™ Software servers. Commands described in this section are run in the Linux shell (not in the LifeScope™ Software command shell).

Only the LifeScope™ Software administrator (LSA) may start the server.

The following command checks if the LifeScope™ Software server is running:

```
ps -ef | grep WebServer
```

If the LifeScope™ Software server is not running, the Linux command prompt returns (with no output). Example output when the web server is running is:

```
lifescope 23521 1 0 13:26 pts/1 00:00:05 java -server -Xmx3072m -XX:PermSize=512m -XX:MaxPermSize=512m -XX:+UseParallelGC com.lifetechnologies.bioscope.webservice.WebServer -b /opt/lifescope/lifescope
```
Chapter 4 Before You Begin
Check LifeScope™ Software servers
CHAPTER 5

Understand The LifeScope™ Software Shell

This chapter covers:

- Overview ................................................................. 63
- Terminology and concepts .......................................... 64
- Common scenarios ....................................................... 72
- Naming restrictions ..................................................... 73

Overview

This chapter describes the concepts and terminology required when working with LifeScope™ Software command shell. New concepts used with the shell include the following list. These terms are explained in more detail in this chapter.

- **Project** – A collection of analyses based on scientific experiments or computational methodologies.
- **Module** – A module is a single step in an analysis workflow. For example, mapping, SAET, and CNV are modules.
- **Analysis** – A set of modules within an analysis workflow. An analysis can contain one or more modules.
- **Analysis workflow** – A pre-defined set of modules in a workflow based on common applications areas, such as targeted resequencing or small RNA analysis.
- **Read-set** – Sequencing data associated with one XSQ file or with an indexed (barcoded) sample from one XSQ file.
- **Read-set repository** – An indexed database of XSQ files based on individual read-sets.
- **Group** – A collection of data that can be analyzed together. Grouping allows for aggregated data (for instance, data from different lanes or different runs) to be analyzed as one set of data.

The command shell presents a virtual directory hierarchy to the user. The shell operations are performed on the LifeScope™ Software server.
**Terminology and concepts**

*Table 7* defines terminology and concepts used to describe the LifeScope™ Software command shell. Review these terms before proceeding to the shell syntax in Chapter 6.

**Table 7 Concepts related to the LifeScope™ Software command shell**

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>category</td>
<td>Category refers to the various types of information containers at the root level of the LifeScope™ Software command shell. An <em>ls</em> command run at the shell root level lists the following categories: <em>bams, projects, reads, references, users, and workflow</em>.</td>
</tr>
</tbody>
</table>
| level, virtual directory | Levels refer to the virtual directory structure within the LifeScope™ Software command shell. For example:  
  The root level: /  
  The workflow level: /workflows  
  The project level: /projects  
  The analysis level: /projects/proj1/run1  
  These levels are a representation of your projects and other shell elements on the LifeScope™ Software server. |
| prompt             | The shell prompt displays your current level within the LifeScope™ Software command shell environment. The following are examples of the prompt strings at various shell levels:  
  At the root level: server:/>  
  At the workflow level: server:/workflows>  
  At the project level: server:/projects>  
  At the analysis level: server:/projects/proj1/run1> |
| projects           | A project is a collection of analyses based on scientific experiments or computational methodologies. In the LifeScope™ Software command shell, projects appear at /projects virtual directory. You create your projects at the /projects level. Within each project, you create one or more analyses.  
  You may use projects as you wish, to group your analyses in a way that is convenient and meaningful for you. Each project is private to an individual LifeScope™ Software command shell user. |
| analysis           | An analysis is a set of modules within an analysis workflow. An analysis can contain one or more LifeScope™ Software modules.  
  An analysis is defined by the following configuration:  
  • The analysis methodology (through the *set workflow* command, specifying which LifeScope™ Software modules are executed).  
  • The source (input) reads (through the *import, add xsq, and set groups*).  
  • The genome reference to be used (with the *set reference* command).  
  In addition, for tertiary-only workflows you can specify source BAM files (through the *add bam* command). |
| module             | A module is a single step in an analysis workflow. For example, mapping, SAET, and CNV are modules. |
### Table 7  Concepts related to the LifeScope™ Software command shell (continued)

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>analysis workflow</td>
<td>A pre-defined set of modules in a workflow based on common applications areas such as small RNA analysis or targeted resequencing. Also known as a standard workflow. A workflow combines a set of analysis modules into one run of LifeScope™ Software. See Chapter 7, “Run a Standard Workflow Analysis” on page 103 for more information.</td>
</tr>
<tr>
<td>analysis space</td>
<td>Your analyses can be performed in either base space or color space (if both base and color are available in your input data). You are recommended to perform secondary and tertiary analyses in the same space. See “Analysis space” on page 69.</td>
</tr>
</tbody>
</table>
| read-set                 | Sequencing data associated with either:  
  - For indexed (barcoded) data: an indexed sample from one XSQ file.  
  - For non-indexed data: the data in one XSQ file.                                                                                                                                                  |
| group                    | A collection of data that can be analyzed together. Grouping allows for aggregated data (for instance, data from different lanes or different runs) to be analyzed as one set of data.  
  The implications of using groups with your read-sets include the following:  
  - Reads that are grouped together are treated as one specimen, even if the reads are in different XSQ input files.  
  - LifeScope™ Software supports processing multiple groups in one analysis run, while treating each group’s data as an independent run.  
  Groups are created through the following methods:  
  - The group option of the `add xsq` command.  
  - The `set groups` command.  
  - The `-g` option of the LifeScope™ Software shell run command.  
  A read-set can belong to a maximum of one group.                                                                                                                                                 |
| groupsfile               | A text file optionally specified with an analysis. The groups file is a tab-separated text file with multiple lines in the following format:  
  - `groupname <path>/xsqorbamfile:indexid <path>/reference_file`  
  If no barcode `indexid` is specified, it is assumed to be ”1” (corresponding to the non-barcoded data). If a reference column entry does not exist, the reference from the previous line in the file is used.  
  All members of a group must use the same reference. A groupsfile supports only a single reference, and that reference must be specified on the first line of the groups file. |
| repository               | A LifeScope™ Software shell repositories are virtual containers that persist your reference files, reads files, and projects.  
  The projects repository, at `/projects`, contains your projects.  
  The read repository, at `/reads`, is an indexed database of XSQ files based on individual read-sets.  
  The references repository, at `/references`, contains the reference genomes used in your analyses.                                                                                                     |
### Chapter 5 Understand The LifeScope™ Software Shell

**Terminology and concepts**

#### Projects and analyses

Projects are containers for your analysis runs. You may use projects as you wish, to organize your analyses as is convenient and meaningful for you.

---

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>directory</td>
<td>The LifeScope™ Software command shell’s repositories and analyses are represented in a structure resembling directories, but these are in a virtual system that lets you browse a representation of your projects and analyses on the LifeScope™ Software server.</td>
</tr>
</tbody>
</table>
| shell | This document discusses two different shells and uses the following terms to distinguish between them:  
- LifeScope™ Software command shell and LifeScope™ Software shell—Refer to the LifeScope™ Software command-line environment that is invoked with the lscope.sh command.  
- Linux shell—Refers to the shell of the underlying operating system. |
| local file system | This document uses the phrase “on your local file system” to describe a file that is specified by a Linux absolute path, such as /data/results/example.sh. Such a file could be on a different physical disk instead of the local file system, but is available to Linux through an absolute path. The file is navigable from or available to the local file system. |
| INI and PLN files | (Optional) INI and PLN (pronounced “plan” or “PLN”) files define your LifeScope™ Software experiment. The LifeScope™ Software command shell creates these files for you, based on the projects and analyses you create and on how you have configure your analyses.  
Each INI file usually corresponds to one LifeScope™ Software module (such as mapping, SNPs, or CNV), and defines the behavior of that module through a set of key-value parameters (for example, the parameters for allowing IUB codes in references or setting the call stringency level).  
Each PLN file defines which LifeScope™ Software modules execute to perform your analysis. A PLN file lists the INI files corresponding to the LifeScope™ Software modules to execute. The PLN file list defines the order of execution and execution dependencies.  
You can define and run your analyses in the LifeScope™ Software command shell, without directly editing INI and PLN files. It is also possible to define analyses outside of the command shell, using only INI and PLN files. See Appendix F, “Command Shell Control Files” on page 553 for information on INI and PLN files. |
| run in the projects repository, or run on the local file system | When “run in the projects repository”, a LifeScope™ Software analysis creates its output files in the projects repository and the analysis [with its configuration, status, and results files] can be viewed through the LifeScope™ Software UI. When “run on the local file system” (also called “run in the current directory”), the output files are created in the current directory on the local file system, and the analysis cannot be viewed through the UI.  
See “Run mode” and Table 12 on page 79 for more information. |
You create your projects at the /projects level. A project itself does not have a configuration or definition. A project is a way of organizing your LifeScope™ Software analysis runs.

Within each project, you create one or more analyses. Through the configuration of an analysis, you define the executable run.

Table 8 lists example steps to create a run in the LifeScope™ Software shell. These steps assume you have logged into the shell. (See Chapter 6, “Run a Command Shell Analysis” on page 75 for complete instructions.)

Table 8 Setting up an analysis run in the LifeScope™ Software shell

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cd projects</td>
<td>Enter your projects area.</td>
</tr>
<tr>
<td>mk proj1</td>
<td>Create a project named proj1.</td>
</tr>
<tr>
<td>cd proj1</td>
<td>Enter the proj1 level.</td>
</tr>
<tr>
<td>mk run1</td>
<td>In your proj1 project, create an analysis named run1.</td>
</tr>
<tr>
<td>cd run1</td>
<td>Open the run1 analysis, to configure the reference and reads files, and the workflow (the analysis modules).</td>
</tr>
</tbody>
</table>

The configuration of analysis includes these areas:

- The reference genome.
- The reads files.
- The type of analysis to be performed (which LifeScope™ Software modules perform the analysis). The analysis type can be any of the following:
  - A pre-defined workflow, which is a built-in sequence of commonly-used analyses, corresponding to a common biological application.
  - An analysis sequence you define, by either creating your own or customizing one of the LifeScope™ Software pre-defined workflows.

An example of a pre-defined workflow is the targeted resequencing fragment workflow. This workflow includes the following LifeScope™ Software modules:

1. SAET
2. Mapping
3. Mapping statistics (BAMStats)
4. Enrichment
5. SNPs
6. Small indels
7. Annotations

One LifeScope™ Software shell command defines this workflow for your analysis (the set workflow targeted.resequencing.frag command).

Once an analysis is configured, it is launched with a single shell command (the run command).

Your projects are private and cannot be seen by other users or shared with other users.
Chapter 5 Understand The LifeScope™ Software Shell
Terminology and concepts

Naming restrictions
Project and analysis names must conform to Linux filename conventions:
- Contain only alphanumeric and underscore characters.
- Do not contain spaces.
- Must be unique (among siblings).
- Do not begin with an underscore character. (This requirement is a LifeScope™ Software limitation, not a Linux rule.)

In sample group names and reference file names, hyphens and spaces are not permitted.

Reads Data
The data from each lane of a 5500 Series Genetic Analysis Systems instrument results in a single XSQ output file. Different models of the 5500 support up to 6 or 12 lanes, so typically multiple XSQ files are generated per sequencing run. You can direct your LifeScope™ runs to analyze data from specific barcodes in multiple lanes, or group the output of multiple lanes or multiple sequencing runs to be processed as a single specimen.

The type of data used in LifeScope™ Software is called a read-set, which is a collection of reads belonging to one index (barcode) from one XSQ file. The smallest set of data you can specify in the shell is one index from one XSQ file.

You can optionally use groups to combine more than one index into an analysis. Reads that have been combined into a group are processed as one specimen, even if the read-sets come from different input XSQ files. Each group generates a combined set of output files.

In the LifeScope™ Software command shell, input data is added at the analysis level. Input data is associated with individual analyses, not with entire projects.

Plan your analysis input carefully. The way you define your reads input affects the behavior of your analysis. Index IDs and group names give you flexibility in handling your input data. For example:
- Combine specific read-sets from one XSQ file or multiple XSQ files into one group to be analyzed together. All read-sets within a group must be of the same library type.
- Assign read-sets to different groups, to be analyzed separately within one LifeScope™ Software run.

For more information and examples of LifeScope™ Software shell commands to use with your input data, see “Define input data” on page 90.

For information about XSQ data and the XSQ file format, see the following sections:
- “XSQ file format” on page 494, in Appendix C, “File Format Descriptions” on page 493
- Appendix D, “XSQ Tools” on page 519

For information about special character restrictions in XSQ file headers, see “Special character restrictions” on page 504.

Repositories
LifeScope™ Software repositories organize and persist your projects, your input data, and your reference data files.
The reads repository is a storage place in LifeScope™ Software for instrument data intended to be input data for LifeScope™ Software analyses. Data imported into the reads repository are in unmapped XSQ-format files.

Importing reads and references file into LifeScope™ Software repositories is optional, but preferred. The following are two advantages to using the shell’s repositories:

- The shell is aware of barcoded read-sets in the read repository. For reads files not in the read repository, you must manually specify barcodes.
- In your shell commands you can use repository names for files in the read and reference repositories. You do not have to use Linux paths to these files.

Files imported into the read and reference repositories are visible to all LifeScope™ Software users. However, each user has their own projects repository. The contents of your projects repository are private and cannot be seen by other users within the command shell.

**Note:** In the current release, one repository cannot be split across multiple partitions or file systems. (The various repositories can each be on a different partition or file system.)

### The reference repository

The reference repository is initially populated during the installation process with reference files for hg18 and hg19.

For more information (not required for most use), see Appendix E, “The Reference Repository” on page 539.

**Note:** In this release, hyphen (‘-’) and space characters are not permitted in reference file names.

### Analysis space

Mapping and select tertiary modules support the option to have your analysis done in base space or color space (if the space is present in your input data). The analysis space for your runs is controlled by these factors and parameters:

- **Input data** – Whether base space and color space are present in your input data.
- **Secondary processing** – Your setting for the `mapping.in.base` parameter. This parameter controls the analysis space for secondary processing. The run fails if the input data does not have the space specified by the `mapping.in.base` parameter.
- **Tertiary processing** – Your setting for the `analysis.space` parameter.

Use the global workflow parameter `analysis.space` to enter your preference for base space or color space tertiary analysis. Three tertiary modules – SNPs, small indel, and splice finder – use your preference, in conjunction with information about whether color space is present in the BAM file and whether mapping is done in base space or color space, to determine their analysis space.

The recommended usage scenarios for the `analysis.space` parameter are:

- For color space analysis, map in color space and set `analysis.space` to color (or use the default value of auto). The following mapping parameter settings are required (these settings are the defaults):
  - `mapping.in.base=0`
  - `bamgen.refcor.addcs=1`
For base space analysis, map in base space and set `analysis.space` to base (or use the default value of auto). The following mapping parameter setting is required:

- `mapping.in.base=1`

An example command to set the `analysis.space` parameter is:

```bash
set analysis.space color tertiary/global.ini
```

If you create your own INI files, set the `analysis.space` parameter in your `tertiary/global.ini` file. An example setting is:

```bash
set analysis.space=base
```

The possible conditions determining the analysis space are given in the following table. The **resulting analysis space** is ... column refers to the SNPs, small indel, and splice finder tertiary modules. (This table is provided for reference only. Recommended scenarios are given above.)

**Table 9** Effect of the `analysis.space` parameter

| Mapping space | Color present in BAM | `analysis.space` value | The resulting analysis space is ...
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>No</td>
<td>Base</td>
<td>Base</td>
</tr>
<tr>
<td>Base</td>
<td>No</td>
<td>Color</td>
<td>Base</td>
</tr>
<tr>
<td>Base</td>
<td>No</td>
<td>Auto</td>
<td>Base</td>
</tr>
<tr>
<td>Base</td>
<td>Yes</td>
<td>Base</td>
<td>Base</td>
</tr>
<tr>
<td>Base</td>
<td>Yes</td>
<td>Color</td>
<td>Color</td>
</tr>
<tr>
<td>Base</td>
<td>Yes</td>
<td>Auto</td>
<td>Base</td>
</tr>
<tr>
<td>Color</td>
<td>No</td>
<td>Base</td>
<td>Base</td>
</tr>
<tr>
<td>Color</td>
<td>No</td>
<td>Color</td>
<td>Base</td>
</tr>
<tr>
<td>Color</td>
<td>No</td>
<td>Auto</td>
<td>Base</td>
</tr>
<tr>
<td>Color</td>
<td>Yes</td>
<td>Base</td>
<td>Base</td>
</tr>
<tr>
<td>Color</td>
<td>Yes</td>
<td>Color</td>
<td>Color</td>
</tr>
<tr>
<td>Color</td>
<td>Yes</td>
<td>Auto</td>
<td>Color</td>
</tr>
</tbody>
</table>

**Find reference files for an analysis**

To find reference files in the references repository, follow these steps:

Log into the shell.

```bash
cd /references
ls
```

The above example lists reference files that have previously been imported (registered) into the references repository. Reference files can alternatively be specified with an absolute path to the file on the local file system (or navigable from the local file system).

See also the shell command `set reference assembly_name` in Table 13 on page 81. When used with a known assembly such as `hg18` or `hg19`, this command automates using the correct reference in your analysis.
One reference is supported per analysis run.

**Find data for an analysis**

To find data in the read-set repository, follow these steps:

Log into the shell.

```
cd /reads
ls
```

The above example lists reads files that have previously been imported (registered) into the reads repository. Reads files can alternatively be specified with an absolute path to the file on the local file system (or navigable from the local file system).

**View analysis information**

To show the status of your jobs, run a shell status command in the Linux shell. The format of the command depends on whether the analysis is in the project repository or in the current directory.

For an analysis in the project repository, run this command:

```
lscope.sh status -u user -p projectname -a analysisname
```

For an analysis in the current directory, run this command:

```
lscope.sh status
```

An alternate method is to run an `ls` command while in an analysis virtual directory, in the LifeScope™ Software command shell. For example:

```
cd /projects
cd proj1
 cd run1
 ls
```

```
--ini--
 secondary/bamstats.ini
 secondary/global.ini
 secondary/fragment.mapping.ini

--readsets--
 lung:DH10B_ColorSpaceOnly_Unbarcoded_50_Frag.xsq:1:DH10B.fasta

--jobs--
 run1 [70%]
```

The `ls` output lists the defined parameters and input read-sets. For running analyses, the output also lists the estimated percentage of completion.

**View analysis read-sets**

To show the read-sets in an analysis, `cd` to that analysis in the shell and run an `ls` command. Part of the `ls` output displays the read-sets which have been added to this analysis:

```
--readsets--(group:file:index:reference)
 Group_1:helen_20100402_2_10103F_F3_1.3.1_2.0_converted.bam::
 Group_1:helen_20100402_2_10103F_F3_1.3.1_2.0_converted.bam::
 Group_1:WTChr17_Frag.xsq:1:
```

The output includes the group name, file name, index number, and reference file, separated by colons. Except for the file name field, some fields may be empty.
Common scenarios

This section describes how LifeScope™ Software command shell handles common biologic analysis scenarios. New terms and concepts, such as project and analysis, are explained later in this chapter. Specific instructions and commands are given in Chapter 6, Run a Command Shell Analysis.

Basic

A basic scenario is to run a bioinformatics analysis, such as targeted resequencing, on sequence data. The steps to do this in LifeScope™ Software shell are:

1. Create a project.
2. In that project, create an analysis.
3. In that analysis:
   a. Specify the type of analysis to be performed on the data (set the workflow).
   b. Add the sequenced data file.
   c. Specify the reference genome for this analysis.
   d. Run the analysis.

A sample sequenced on multiple lanes

In this scenario one sample is sequenced in multiple lanes on the sequencing instrument. The output data from the sequencing instrument is contained in multiple XSQ files.

In the LifeScope™ Software command shell, the multiple XSQ files are analyzed in one run and as one sample. LifeScope™ Software treats the multiple data files as one set of data.

The steps to do this in LifeScope™ Software shell are:

1. Create a project.
2. In that project, create an analysis.
3. In that analysis:
   a. Specify the type of analysis (such as a standard workflow) to be performed on the data.
   b. Add the XSQ data files. When you add the XSQ files, specify the same group name for each file.
   c. Specify the reference genome for this analysis.
   d. Run the analysis.

Data from multiple samples

This scenario performs, in one run, the same analysis on sequence data from multiple specimens. The data can be in one or multiple XSQ files.

LifeScope™ Software keeps the data from each specimen separate, and each specimen is analyzed separately. The steps to do this in LifeScope™ Software shell are:

1. Create a project.
2. In that project, create an analysis.
3. In that analysis:
a. Specify the type of analysis (such as a standard workflow) to be performed on the data.
b. Add the sequenced data file, specifying different grouping for the data.
c. Specify the reference genome for this analysis.
d. Run the analysis.

Naming restrictions

LifeScope™ Software places certain character restrictions on file names and other names (in addition to the Linux filename conventions), as shown in the following table:

Table 10 Naming restrictions

<table>
<thead>
<tr>
<th>Use</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hyphen (‘-’)</td>
</tr>
<tr>
<td>Reference file names</td>
<td>Not allowed</td>
</tr>
<tr>
<td>Assembly names</td>
<td>Not allowed</td>
</tr>
<tr>
<td>Sample group names</td>
<td>Not allowed</td>
</tr>
<tr>
<td>Names of INI and PLN files</td>
<td>Not allowed</td>
</tr>
<tr>
<td>Names of input CSFASTA and QUAL files</td>
<td>Not allowed</td>
</tr>
<tr>
<td>Names of input XSQ files</td>
<td>Not allowed</td>
</tr>
<tr>
<td>Project names</td>
<td>—</td>
</tr>
<tr>
<td>Analysis names</td>
<td>—</td>
</tr>
</tbody>
</table>

For files listed in the table, the restrictions apply to the base file name only, not to paths up to the filename.

In addition, project names and analysis names within LifeScope™ Software must contain only alphanumeric and underscore characters. Project names must not begin with an underscore.
Chapter 5  Understand The LifeScope™ Software Shell

Naming restrictions
Run a Command Shell Analysis

This chapter covers:

- Overview ................................................. 75
- Run a standard workflow ................................ 76
- Use the command shell .................................... 77
- Run a grouped analysis ..................................... 93
- Run a tertiary-only workflow ............................... 95
- Run an individual analysis ................................. 97
- Resume, re-run, and re-use ................................ 98
- Review job status .......................................... 99
- Review results .............................................. 100
- Review logs .................................................. 100
- Error behavior and error messages ..................... 100

Overview

The LifeScope™ Software command shell provides a way to quickly run bioinformatics experiments. The shell has its own working environment, which is organized around the concepts of projects and of analyses within projects. The LifeScope™ Software command shell has elements similar to the Linux shell but with a different syntax and interpretation. If you are new to LifeScope™ Software, please review Chapter 5, “Understand The LifeScope™ Software Shell” on page 63 before reading this chapter.

New users are recommended to use the standard workflows built into LifeScope™ Software. Standard workflows are developed with defaults optimized for common analyses. The applications supported by standard workflows are:

- Genomic resequencing
- Targeted resequencing
- Low frequency variant detection
- Whole transcriptome
- Small RNA
- ChIP-Seq
- MethylMiner™
Run a standard workflow

A standard workflow is a built-in series of commonly-used analyses, which correspond to a common biological application. The pre-built standard workflows include genomic resequencing, targeted resequencing, small RNA, ChIP-Seq, MethylMiner™, and whole transcriptome. Using one of the workflows allows you to run an analysis with a minimum of setup.

In the LifeScope™ Software command shell, running a standard workflow requires the following steps:

1. Create a project and analysis.
2. Identify the workflow to be executed on your data.
3. Identify your input data (your reads).
4. Identify the reference genome.
5. For targeted resequencing and low frequency variant workflows, identify the regions of interest file.
6. Issue the run command to start your analysis.

Example steps to run a standard workflow

These steps are an example of how to run a workflow in the LifeScope™ Software command shell:

```
# log into the shell
lscope.sh shell -u username -w password
# cd to the projects repository
cd /projects
# make a project, and open it
mk ecoli
cd ecoli
# make an analysis, and open it
mk run1
cd run1
# define the analysis type
set workflow targeted.resequencing.frag
# define the input
add xsq DH10B_ColorSpaceOnly_Unbarcoded_50_Frag.xsq lung
# specify the reference to be used
set reference /data/results/references/DH10B.fasta
# define the any analysis-specific parameters
set analysis.regions.file /data/results/readsdata/dh10b.bed
# list the configuration of the analysis
ls
# start the run
run
# list progress information about the run
ls
```
Use the command shell

The LifeScope™ Software shell is a command-line tool for running LifeScope™ Software data analyses. The shell also allows you to create and manage your LifeScope™ Software data, projects, analyses, and users.

If you instead want to run the LifeScope™ Software user interface (UI), refer to the LifeScope™ Genomic Analysis Software User Guide (Part no. 4476537).

Table 11 Sample shell commands to run a workflow

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>lscope.sh shell -u username</td>
<td>Log into the LifeScope™ Software command shell.</td>
</tr>
<tr>
<td>cd /reads import /data/xsq/DH10B_ColorSpaceOnly_Unbarcoded_50_Frag.xsq</td>
<td>Go into the read repository and import an XSQ file (the input data file). This command in effect brings the reads file into the shell read repository. Once in the read repository, the reads file is available as input to your analysis runs.</td>
</tr>
<tr>
<td>cd /projects mk ecoli cd ecoli</td>
<td>In the projects repository, create a project named ecoli, and open that project.</td>
</tr>
<tr>
<td>mk run1 cd run1</td>
<td>Create an analysis named run1. Open the run1 analysis. Note: The open analysis (cd run1, in this case) command is important. The data, reference, and other configuration commands which follow apply only to the current analysis.</td>
</tr>
<tr>
<td>set workflow targeted.resequencing.frag</td>
<td>Define the workflow used with this analysis. The workflow in this example includes SAET, mapping, mapping statistics, SNPs, inversion, large indel, small indel, and annotations.</td>
</tr>
<tr>
<td>add xsq DH10B_ColorSpaceOnly_Unbarcoded_50_Frag.xsq</td>
<td>Associate the XSQ file with this analysis. This command directs LifeScope™ Software to use this XSQ file as the input reads data for this analysis. This example uses the repository name for the XSQ file. The absolute path to the XSQ file on the local file system is also supported.</td>
</tr>
<tr>
<td>set reference /data/results/results/references/DH10B.fasta</td>
<td>Define the reference file for this analysis.</td>
</tr>
<tr>
<td>set analysis.regions.file /data/results/readsdata/dh10b.bed</td>
<td>Define the regions of interest file, for targeted resequencing (also low frequency variant) workflows.</td>
</tr>
<tr>
<td>ls run</td>
<td>Display the configuration of your analysis, including input reads, reference files, workflow, and select parameter settings. Start the analysis run.</td>
</tr>
<tr>
<td>wait ls</td>
<td>Wait for the analysis run to complete. Display configuration and status information about the analysis run.</td>
</tr>
</tbody>
</table>
Chapter 6  Run a Command Shell Analysis

Use the command shell

There are four modes of operation for the LifeScope™ Software command shell:

- **Run mode** – Runs a LifeScope™ Software analysis or workflow from the Linux shell, without entering the LifeScope™ Software command shell. The LifeScope™ Software control file (plnfile) must be created and configured before executing a run mode command.
  
  Example commands:
  ```
  lscode.sh run plnfile -u marie -p proj1 -a run1 -g group1
  lscope.sh resume plnfile -u joe
  ```
  See “Run mode” on page 78.

- **Status mode** – Opens the LifeScope™ Software shell status viewer, which shows the dynamic status of running and completed jobs for specified analysis.
  
  Example commands:
  ```
  lscope.sh status -p proj1 -a analysis1
  lscope.sh status
  ```
  See “Status mode” on page 79.

- **Scripted mode** – You invoke the LifeScope™ Software shell from commands listed in a file on your local system. See “Scripted mode” on page 80 for information on running in scripted mode.

- **Shell mode** – Enters the LifeScope™ Software command shell. In this shell, only LifeScope™ Software shell commands are accepted. The LifeScope™ Software shell commands are described in Table 13 on page 81. The LifeScope™ Software shell commands allow you to create, manage, and run your LifeScope™ Software analyses.
  
  Example commands:
  ```
  • To enter the shell: lscope.sh shell -u marie
  ```
  Within the shell:
  ```
  • set workflow whole.transcriptome.frag
  • set reference DH10B.fasta
  ```
  See “Shell mode” on page 81.

**Note:** Do not confuse the LifeScope™ Software shell commands with Linux or UNIX commands. Some commands may have similar names, but the behavior of the LifeScope™ Software shell commands is entirely different from the Linux or UNIX commands. LifeScope™ Software shell commands also depend on the shell level (analysis, project, reads, references, or root) at which they are executed.

---

**Run mode**

With the run mode, you enter the LifeScope™ Software command in the Linux shell. You remain in the Linux shell, and the LifeScope™ Software command shell does not open. Run mode also includes the resume command.

The following are example run mode commands:

```bash
lscope.sh run ./splice.finder.pln -p wt_spliceFinder \ 
   -a wt_spliceFinder_PE_TC12 -u rjones
lscope.sh resume ./splice.finder.pln -p wt_spliceFinder \ 
   -a wt_spliceFinder_PE_TC12 -u rjones
```

The `-p` option specifies the project name. The `-a` option specifies the analysis name. The `-u` option specifies the username.
A \texttt{-g groupsfile} parameter is optional. When a \texttt{groupsfile} is provided, LifeScope™ Software performs multi-group analysis, which involves analyzing the data for each group independently. See “Run a grouped analysis” on page 93 for more information about multi-group analyses.

**When to use run mode**

An \texttt{lscope.sh} run mode command is appropriate when you have already configured your analysis. Prerequisites to running an \texttt{lscope.sh} run mode command include:

- The LifeScope™ Software web server must be running.
- The LifeScope™ Software user account must exist.
- The reads files and reference files required by the analysis must have been transferred to or available to the LifeScope™ Software cluster or workstation.
- Reference files, reads files, and read-sets must either have been specified in a previous LifeScope™ Software command shell session, or specified by absolute paths in the analysis PLN and INI files.
- The analysis PLN and INI files are configured correctly.

**Run “in the projects repository” or “in the local file system”**

A \texttt{run} command that includes the project and analysis options is considered to “run in the projects repository”. The shell projects repository determines the output file location, and the analysis run is visible and trackable from the LifeScope™ Software UI.

Without the project and analysis options, a \texttt{run} command is considered to “run in the local file system”. Output files are created in the current directory on the file system, and the run is tied to the current directory for checking status.

Table 12 compares the two methods.

<table>
<thead>
<tr>
<th>Category</th>
<th>Run in the projects repository</th>
<th>Run in the local file system</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command format</td>
<td>\texttt{lscope.sh run plnfile -u user -p proj -a analysis -g groupsfile}</td>
<td>\texttt{lscope.sh run plnfile -u user} Must be executed in the local file system in the same directory as plnfile.</td>
</tr>
<tr>
<td>Status command</td>
<td>\texttt{lscope.sh status -u user -p proj -a analysis}</td>
<td>\texttt{lscope.sh status} Must be executed in the same directory of the local file system.</td>
</tr>
<tr>
<td>Output file location</td>
<td>In the /projects repository</td>
<td>In the current directory of the local file system.</td>
</tr>
<tr>
<td>Visible in the UI?</td>
<td>Yes. You can view the analysis configuration, reads files, parameters, status, and results files in the UI.</td>
<td>No.</td>
</tr>
</tbody>
</table>

**Status mode**

The \texttt{lscope.sh status} command displays information about your running LifeScope™ Software analyses.

To check the status of the example run described in “Run mode”, enter this command in the Linux shell:

\begin{verbatim}
lscope.sh status -u rjones -p wt_spliceFinder \end{verbatim}
Use the command shell

-a wt_spliceFinder_WT_SF_PE_TC0012

If your run command is running in the local file system, use one of the following commands to check status. The format depends on whether or not your analysis is in the projects repository:

- For an analysis in the project repository, run this command:
  `lscope.sh status -u user -p projectname -a analysisname`
- For an analysis in the current directory, run this command:
  `lscope.sh status`

The following is an example of the `lscope.sh status` screen:

![Status screenshot]

You must be in the correct directory on your local file system (the directory from which you executed the run command).

(For another status command, see Table 13. The shell `ls` command, when executed with the shell in an analysis level, gives the status of an analysis run.)

**Scripted mode**

The LifeScope™ Software command shell accepts as input a text file containing shell commands.

1. Create a text file with one command per line. For example:
   ```bash
   cd proj1
   cd run2
   set workflow analysis.pln
   add xsq xsqfile
   set reference hg19
   run
   ls
   exit
   ```
2. Save the file, for example as `proj1script`. 
3. Then run with this command:

   lscope.sh -u user -w password < proj1script

**Note:** This example does not include the commands necessary to step up a new analysis.

**Shell mode**

In shell mode, you enter the LifeScope™ Software command shell. In this shell, only LifeScope™ Software shell commands are accepted. The LifeScope™ Software shell commands allow you to create, manage, and run your LifeScope™ Software analyses.

**Terminology**

Table 7 on page 64, in Chapter 5, Understand The LifeScope™ Software Shell, defines concepts and terminology used with the LifeScope™ Software command shell. Review these terms before proceeding to the shell commands and their syntax.

**lscope.sh commands**

Table 13 describes the commands available within the LifeScope™ Software command shell. The behavior of some shell commands depends on the level at which they are executed, as noted in Table 13.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Project and analysis commands:</strong></td>
<td></td>
</tr>
<tr>
<td>mk name</td>
<td>Creates a new project or a new analysis within the current project. Must be executed at the /projects or /projects/projname levels.</td>
</tr>
<tr>
<td></td>
<td>When executed at the /projects level, creates a new project named name.</td>
</tr>
<tr>
<td></td>
<td>When executed at the /projects/projname level, creates a new analysis named name, within your project projname.</td>
</tr>
<tr>
<td></td>
<td>Command examples:</td>
</tr>
<tr>
<td></td>
<td>• At the /projects level: mk testproj</td>
</tr>
<tr>
<td></td>
<td>• At the /projects/projname level: mk run1</td>
</tr>
<tr>
<td></td>
<td>See &quot;Project and analysis naming rules&quot; on page 89 for restrictions on name.</td>
</tr>
</tbody>
</table>
Chapter 6  Run a Command Shell Analysis

Use the command shell

---

Table 13  LifeScope™ Software shell commands (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
</table>
| `ls` | Lists information about your project or analysis. When executed at the `/projects` level, lists the names of all your projects. At the `/projects/projname` level, lists the names of all analyses contained within your project named `projname`. At the `/projects/projname/analysisname` level, provides configuration and progress reporting for the analysis:  
- Lists INI files involved in the analysis, if `set workflow` has been given.  
- Lists read-sets involved in the analysis, if read-sets have been added. The read-set listing includes the group, file name, index, and reference.  
- Lists the status of jobs for the analysis.  
Example:
  
  \`ls --ini--
  fragmentMapping.ini  
global.ini  
--readsets--(group:file:index:reference)
  Group_1:case0030_Fragment.xsq:1:chr16.validated.fasta  
--jobs--
  run1 [100%] [JOB_SUCCESS]
  fragmentMapping [100%] [JOB_SUCCESS]

Example two:
  
  \`ls --ini--
  secondary/global.ini  
  secondary/smallrnamapping.ini  
  tertiary/global.ini  
  tertiary/srCount.ini  
  tertiary/srCoverage.ini  
--readsets--(group:file:index:reference)
  HumanFrag_smRNA:reads.xsq:1:hgl_1b.fasta  
--jobs--
  smallRNAworkflow [100%] [JOB_SUCCESS]
  secondary-hgl_1b-smallrnamapping [100%] [JOB_SUCCESS]
  tertiary-HumanFrag_smRNA-srCount [100%] [JOB_SUCCESS]
  tertiary-HumanFrag_smRNA-srCoverage [100%] [JOB_SUCCESS]
### Chapter 6  Run a Command Shell Analysis

#### Use the command shell

**Table 13  LifeScope™ Software shell commands (continued)**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
</table>
| `rm name`       | Permanently deletes a project or an analysis within the current project. Must be executed at the `/projects` or `/projects/projname` levels.  
When executed at the `/projects` level, deletes the project named `name`.  
When executed at the `/projects/projname` level, creates a new analysis named `name`, within your project `projname`.  
Command examples:  
  - At the `/projects` level: `rm testproj`  
  - At the `/projects/projname` level: `rm analysis2`  
**WARNING!** This command permanently deletes the underlying project and analysis directories on the server. Recovery is **not** supported. |
| `set workflow name or path` | Sets the workflow or PLN file for the current analysis. This command is a prerequisite to running an analysis. The `set workflow` command must also precede any `set group` command or `set param value` commands.  
**Examples:**  
  - `set workflow targeted.resequencing.frag`  
  - `set workflow /share/lifescope/human/test1/my.pln`  
The `set workflow` command replaces the manual configuration of PLN and INI files used in earlier versions of BioScope™ Software. With this command the LifeScope™ Software shell automatically creates the required PLN and INI files. |
| `get workflow` | Copies the current workflow's INI files, PLN files, and their directory structure to the local file system. These files are copied to the directory you are in when you give the `lscope.sh` command.  
You must have a shell analysis open (be in the analysis virtual directory) and that analysis must have a workflow already set (with the `set workflow` command). |
| `ls`             | When executed at the level of a specific standard workflow, such as `/workflows/small.rna`, lists the INI files for the workflow. For example:  
  - `server:/workflows> cd small.rna`  
  - `server:/workflows/small.rna> ls`  
  - `secondary/bamstats.ini`  
  - `secondary/global.ini`  
  - `secondary/smallrnamapping.ini`  
  - `tertiary/filtermapping.bamstats.ini`  
  - `tertiary/global.ini`  
  - `tertiary/srCount.ini`  
  - `tertiary/srCoverage.ini`  
**Reads file (input data) commands:** |
import path [reponame]

When executed at the /reads level, imports a source XSQ file from the Linux file system into the reads repository (and updates the read repository index). The import command imports to the reads repository top folder, not to a subfolder.

When executed at the /bams level, imports a BAM input data file into the BAM repository (for tertiary analysis). The /bams virtual directory is reserved for BAM files that have been converted from earlier versions of the software or imported from another installation of the same version of LifeScope™ Software.

The path argument is one of the following:
• The absolute path to the file on the Linux file system (recommended), or
• A relative path to the file, using as the current directory the Linux directory in which the lscope.sh shell command was given.

The reponame argument is optional, and allows you to assign a repository name different from the filename. The file appears as reponame when the repository contents are displayed, and is accessed as reponame with commands such as add xsq reponame.

The import command may be one of these forms:
• import-ln – Links the file (with ln -s) given by path to the repository directory on the LifeScope™ Software server. Default.
• import-cp – Copies the file given by path to the repository directory on the LifeScope™ Software server.
• import-mv – Moves the file given by path to the repository directory on the LifeScope™ Software server.

add xsq name [id] [group]

The add xsq command specifies input for secondary analysis. It adds the read-sets with index (barcode id) id in the XSQ file specified by name into the current analysis. add xsq must be run at the analysis level.

name must match either a reads file that has been imported into the reads repository or the full path to an XSQ file on the local file system. The ls command run at the /reads level lists available read repository files.

id is optional. If no id is provided, all indices [barcodes] from the XSQ file are added. Each id refers to the data for one index.

group is optional. If group is provided, all read-sets are grouped together, and analyzed together as one specimen. Read-sets from multiple XSQ files may be added the same group name.

If group is not provided with indexed data, the following occurs:
• Read-sets are grouped by index name from the XSQ file.
• The read-sets from each index (barcode) are processed independently.

All read-sets in a group are analyzed together as one specimen. The group name is not required to match the specimen name or other information from the instrument run. Do not use spaces or special characters in a group name.

Multiple add xsq commands are supported per analysis.

del xsq filename

Removes the XSQ file specified by filename from the current analysis. filename must appear in the listing of the read repository (as listed by the LifeScope™ Software shell ls command at the /reads level). This command does not physically delete the XSQ file from your file system.
### Chapter 6 Run a Command Shell Analysis

**Use the command shell**

#### Table 13 LifeScope™ Software shell commands (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
</table>
| `rm xsq name` | Removes the XSQ file `name` from the /reads repository. `name` is the file’s repository name.  
If the XSQ file `name` was originally imported with `import-ln`, `rm` deletes the link in the repository directory on the local file system.  
If the XSQ file `name` was originally imported with `import-cp` or `import-mv`, `rm` deletes the file from the repository directory on the local file system. |
| `set groups path` | Creates read-set groups from the file given by `path`. Read-sets belonging to one group are analyzed together as one specimen. Each group generates an independent set of results files (containing data combined for the whole group).  
LifeScope™ Software can process multiple groups in one analysis run, while treating each group’s data as an independent run.  
The groups file must be a tab-separated text file with multiple lines in the following format:  
```
groupname <path>/XSQorBAMfile:indexid <path>/reference_file  
```
`path` must be an absolute path to the groups file on (or available to) the local file system.  
If no `index id` is specified, it is assumed to be “1”. All read-sets from the file are included.  
If the reference column entry does not exist, the reference from the previous line in the file is used.  
The XSQ, BAM, and reference files may be specified by their repository names, if the files have been imported into a LifeScope™ Software shell repository. If these files are specified by a path on the local file system, an absolute path is recommended. Relative paths are interpreted relative to the directory where you open the LifeScope™ Software command shell with the `lscope.sh` shell command.  
The `set workflow path` command must appear before the `set group path` command. |
| `add bam path [group]` | [Optional] Adds BAM file `path` into the current analysis. Use this command to specify input for tertiary analysis.  
`path` must match either a BAM file that has been imported into the bams repository or an XSQ file on the local file system. The `ls` command run at the `/bams` level lists available bams repository files.  
`group` is optional. If no `group` is provided, `group` is assumed to be "Default". All BAM files in a group are analyzed together as one specimen. |
| `del bam name` | Removes the BAM file `name` from read-sets to be analyzed.  
This command does not physically delete the BAM file from your file system. |
| `rm bam name` | Removes the BAM file `name` from the /bams repository. `name` is the file’s repository name.  
If the BAM file `name` was originally imported with `import-ln`, `rm` deletes the link in the repository directory on the local file system.  
If the BAM file `name` was originally imported with `import-cp` or `import-mv`, `rm` deletes the file from the repository directory on the local file system. |
| `cat bam path` | Displays header information for the BAM file given by `path`. `path` must be the absolute path to a BAM file on the file system. |
| `ls` | When executed at the `/reads` level, lists the XSQ files in the read repository. When executed at the `/bams` level, lists the files in the BAM repository. The library type and data type (base space or color space) are included in the listing. Example output at the `/reads` level is:  
```
smallindel_simulation_2INS_22_25.xsq [Fragment] [color]  
AllMapBarcodes.xsq [MatePair] [mixed] [Barcoded]  
DH10B ColorSpaceOnly_50_Frag.xsq [Fragment] [color] [Non-barcoded]  
DefaultLibrary_PE.xsq [PairedEnd] [color] [Non-barcoded]  
Fragment3panels.xsq [MatePair] [mixed] [Non-barcoded]  
```
Chapter 6  Run a Command Shell Analysis

Use the command shell

Table 13  LifeScope™ Software shell commands (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat xsqfile</td>
<td>In /reads, the cat command with an XSQ file argument displays the libraries and barcodes contained in the XSQ file. For example:</td>
</tr>
</tbody>
</table>

```
server:/reads> cat Indexing3panels.xsq
[lib:Library1] [idx:1] [MatePair] [50,50] [mixed]
[lib:Library1] [idx:13] [MatePair] [50,50] [mixed]
[lib:Library1] [idx:5] [MatePair] [50,50] [mixed]
[lib:Library1] [idx:9] [MatePair] [50,50] [mixed]
[lib:Library2] [idx:10] [MatePair] [50,50] [mixed]
[lib:Library2] [idx:14] [MatePair] [50,50] [mixed]
[lib:Library2] [idx:2] [MatePair] [50,50] [mixed]
[lib:Library2] [idx:6] [MatePair] [50,50] [mixed]
[lib:Library3] [idx:11] [MatePair] [50,50] [mixed]
```

Reference file commands:

- set reference assembly | path

**assembly** names an assembly in the LifeScope™ Software shell reference repository. The LifeScope™ Software shell sets reference-related parameters to appropriate default values. An example assembly name command is:

```
set reference hg18
```

**path** sets the genome reference to the reference file path in current analysis. path can either be an absolute filepath (on the Linux file system), or a reference name that exists in the shell reference repository (as listed by the LifeScope™ Software shell ls command at the /references level).

If **assembly** names a recognized assembly in the reference repository, then the following reference-related parameters are automatically set to default values for **assembly**:

- annotation.dbsnp.file
- annotation.gtf.file
- analysis.mirbase.precursor.file
- analysis.mirbase.mature.file
- analysis.filter.reference
- analysis.assembly.name
- analysis.species.name

The set values can be changed with the set param.key param.value command, if necessary.

A set reference command (either **assembly** or **path**) is mandatory before running an analysis. The use of **assembly** is recommended. A maximum of one reference is allowed per analysis.

Run the LifeScope™ Software shell ls command to review the parameter settings. Before starting your analysis, confirm that these parameters are set correctly.

Parameter commands:
Chapter 6  Run a Command Shell Analysis

Use the command shell

### Table 13  LifeScope™ Software shell commands (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
</table>
| **set** param.key *param.value [inifile]** | Sets a parameter for the analysis. The analysis workflow must be defined (with the **set workflow arg** command) before you set a parameter for that analysis. For example:  
```bash
set analysis.regions.file absolute_path/regions_bed_file
set annotation.dbsnp.file absolute_path/dbsnp_file
set annotation.gtf.file absolute_path/refgene_gtf_file
set analysis.filter.reference absolute_path/filter_reference_file
set analysis.mirbase.mature.file absolute_path/mature_forms_file
set analysis.mirbase.precursor.file absolute_path/precursor_forms_file
set analysis.assembly.name name_of_reference_assembly
set analysis.species.name name_of_reference_species
```

Some parameters are automatically pre-populated based on the reference chosen. See “List shell commands” on page 89.

System parameters may also be set. For example:
```bash
set scratch.dir /scratch/lifescope
```

The *ini* argument is optional. When *ini* is specified, the parameter is set for an individual INI file within the analysis. For example, the following command sets the value of the parameter `bamstats.wig.primary.only` to 0, for only the BAMStats module:
```bash
set bamstats.wig.primary.only 0 secondary/bamstats.ini
```

The *ini* value is case sensitive. The secondary or tertiary location of the INI file is required.

If the *ini* argument is not provided, the parameter is added to both the secondary and tertiary global.ini files.

The *inifile* version of the command supports turning off certain analysis modules. This method is not recommended for a module that has dependencies in the PLN file. The following example turns off the bamstats module:
```bash
set bamstats.run 0 secondary/bamstats.ini
```

**Note:** The **set workflow command** must precede any **set param value** commands.

**Note:** Do not use an equals sign ‘=’ with the **set** command.

| cat inifile | Lists parameters for the current analysis, including ones defined with the **set param value [inifile]** command. Example:  
```bash
cat pair.mapping.ini
    default.maximum.insert.size=20000
    default.minimum.insert.size=150
    pair.mapping.run=1
```

| del param inifile | Undefines a parameter from an INI file. The *inifile* value is case sensitive. If no INI file is specified, deletes the parameter from all global.ini files. If an INI file is specified, the secondary or tertiary location of the INI file is required. The following example deletes the definition of the parameter `analysis.input.readset.file` from a mapping INI file. The parameter `analysis.input.readset.file` must not be set if the SAET module is not run.
```bash
del analysis.input.readset.file secondary/fragment.mapping.ini
```

**Execution commands:**

| run | Begins execution of the current analysis. |
| wait | Waits for the currently running analysis to complete. If the current analysis not running, returns immediately. Must be run in an analysis directory. |
| stop | Stops execution of the current analysis. |
Chapter 6  Run a Command Shell Analysis
Use the command shell

### Section 6.4 Run a Command Shell Analysis

#### Command Shell User Guide

**Chapter 6**

**Use the command shell**

**Resume Resumes execution of the current analysis, if that analysis or a portion of it is in a failed or stopped state. Modules that successfully completed execution are not re-run.**

Job resumption is at the module level. A module that partially completed but then failed is restarted from the beginning by the `resume` command.

The shell resume command can also be issued in the Linux shell, as a run mode command. The syntax is:

```
lscope.sh resume plnfile -u user -p project -a analysis
```

### Navigation commands:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
</table>
| `cd /`  | Navigates to the root level of the LifeScope™ Software command shell. May be executed at any shell level.  
**Note:** `/` is the only argument allowed. You may not specify multiple levels separated by one or more `/` characters. |
| `cd ..` | Navigates up one level (one virtual directory) in the shell.  
**Note:** The `cd` command navigates only one level at a time. |
| `cd name` | Navigates down one level (one virtual directory) in the shell.  
When executed at the shell root level (`/`), opens the category named `name`. Allowed categories names are `bams`, `projects`, `reads`, `references`, `users`, and `workflow`.  
**Note:** Only admin users have access to the `users` category.  
When executed at the `/projects` level, opens your project named `name`.  
At the `/projects/projname` level, opens your analysis named `name`.  
At the `/workflows` level, opens the standard workflow named `name`.  
**Note:** The `cd` command navigates only one level at a time. |

### Administrative user commands:

**Note:** Only admin users have access to the `users` category. Must be executed at the `/users` level. These commands require admin role privilege.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ls</code></td>
<td>Lists all users.</td>
</tr>
<tr>
<td><code>mk name</code></td>
<td>Creates a new user named <code>name</code>, with the password <code>name</code> and the role <code>user</code>.</td>
</tr>
<tr>
<td><code>rm name</code></td>
<td>Deletes the LifeScope™ Software shell user named <code>name</code>.</td>
</tr>
<tr>
<td><code>loggedin</code></td>
<td>Shows logged in users.</td>
</tr>
</tbody>
</table>
| `set username role [password]` | Sets a user’s role and optionally set the user’s password.  
Valid roles are: admin, user |
| `makepasswdfile [passwordfile]` | Creates an encrypted password file for non-interactive login. By default, the file is named `.lscope.passwd`, and is created in your home directory.  
Use the optional `passwordfile` parameter to create the password file with a non-default name or location.  
When the `lscope.sh shell` command is given without a password, LifeScope™ Software looks for a `.lscope.passwd` file in the user’s directory. If the file exists, `lscope.sh` uses its password. This allows the command `lscope.sh -u username` to log the user in without asking for password.  
To use a password file of a different name or location, use the `-e` option:  
```
lscope.sh -e passwordfile
``` |
Chapter 6  Run a Command Shell Analysis

Use the command shell

Project and analysis naming rules

Project and analysis names must conform to Linux filename conventions:

- Contain only alphanumeric and underscore characters.
- Do not contain spaces.
- Must be unique (among siblings).
- Do not begin with an underscore character. (This requirement is a LifeScope™ Software limitation, not a Linux rule.)

Command completion and command history

The LifeScope™ Software command shell supports command completion, with the keyboard Tab key. The command shell supports command history, with the keyboard up arrow.

List shell commands

There are two methods to list shell commands and command syntax.

- On the Linux command line, enter this command: `lscope.sh man`
  This command lists the `lscope.sh man` page. This version is more verbose than the help output within the command shell.
- In the LifeScope™ Software command shell, enter this command: `help`
  Help lists the shell commands, with their syntax and with a brief description.

Options

The options `lscope.sh` command supports are described in Table 14.

### Table 13  LifeScope™ Software shell commands (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Other administrative commands:</strong></td>
<td>These commands require admin role privilege.</td>
</tr>
<tr>
<td>rebuild</td>
<td>In <code>/reads</code>, rebuilds the read repository index, which is used to display the list of read-sets in the repository. Recent changes (additions or deletions) to the reads repository are not reflected in the <code>ls</code> output until the index is rebuilt.</td>
</tr>
<tr>
<td><strong>Miscellaneous commands:</strong></td>
<td></td>
</tr>
<tr>
<td>config</td>
<td>Displays configuration information for the LifeScope™ Software server.</td>
</tr>
<tr>
<td>help</td>
<td>Briefly lists the LifeScope™ Software shell commands and syntax. The man page is more verbose. [The man page is access with the <code>lscope.sh man</code> command, issued in the Linux shell.]</td>
</tr>
<tr>
<td>exit</td>
<td>Logs out of the LifeScope™ Software web server, and exits the shell. Running analyses are not affected.</td>
</tr>
</tbody>
</table>
| `!cmd` | Runs a Linux shell command. Example: `!pwd`
  *Do not use `!cmd` to run a Linux `lscope.sh` command from within the command shell.* |

### Table 14  `lscope.sh` options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-a</code></td>
<td>Analysis name</td>
</tr>
</tbody>
</table>
Chapter 6  Run a Command Shell Analysis
Use the command shell

Table 14  lscope.sh options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-e passwordfile</td>
<td>The command shell uses the password in passwordfile on startup. This option is not required when you use the default password file name and location. See the makepasswdfile [passwordfile] command.</td>
</tr>
<tr>
<td>-g</td>
<td>Group name.</td>
</tr>
<tr>
<td>-p</td>
<td>Project name.</td>
</tr>
<tr>
<td>-u</td>
<td>Username.</td>
</tr>
</tbody>
</table>

For information on analysis name and project name see Table 7, “Concepts related to the LifeScope™ Software command shell” on page 64.

For information on groups, see the following:
- “Define input data”
- “Run a grouped analysis” on page 93

Define input data

This section describes adding reads input data to one analysis. The reads input can either be in the read repository or specified by absolute path to a file on the local file system.

Plan your analysis input carefully. The way you define your reads input affects the behavior of your analysis. Index IDs and group names give you flexibility in handling your input data.

Index IDs

The shell supports two methods of specifying index IDs with your reads data. Both the add xsq command and groupsfiles support index IDs. Using an index ID restricts the input to only the data for a specific index. For example:

```
add xsq xsqfile:62
```

Groups processing

Combining reads data into groups causes the data to be treated as one specimen, with one set of results files that represents the combined data for the group.

Groups are created with the optional group field of the add xsq command, or with a groupsfile and either the set groups command or the -g groupsfile option of the lscope.sh run command. See “Run a grouped analysis” on page 93 for an example of a groupsfile.

The following is an example of an add xsq command with a group field:

```
add xsq xsqfile:62 Human
```

Example input data scenarios

The following sections describe scenarios for defining your input data.
Define the input reads for a sequencing library instance

Suppose your specimen is sequenced on the instrument with three indices (barcodes). You require that LifeScope™ Software treat the data from these three indices as one specimen and generate one set of results files for the specimen. You combine your reads data into a group to achieve this effect. The following are two methods to combine these indices into one group.

These examples imply the data is in one XSQ file, but that is not required (XSQ file does not have to be the same XSQ file for all the indices).

The add xsq command

If the indices for your specimen are 3, 4, and 9, and you choose the group name Human for your specimen's data, the following commands cause your data to be analyzed as one specimen:

```
add xsq XSQfile:3 Human
add xsq XSQfile:4 Human
add xsq XSQfile:9 Human
```

A groupsfile

A groupsfile also associates the specimen's indices with a group, and is more convenient for large numbers of indices. Here are example contents of the groupsfile:

```
Human <path>/XSQfile:3 <path>/reference_file
Human <path>/XSQfile:4
Human <path>/XSQfile:9
```

Use this groupsfile with either the set groups shell command or the -g option of the lscope.sh run command:

```
set groups <path>/groupsfilename
lscope.sh run ... -g <path>/groupsfilename
```

Define reads in separate groups

Contrast the following treatment of reads data with the example in “Define the input reads for a sequencing library instance” on page 91:

```
add xsq XSQfile:3 Human
add xsq XSQfile:4 Rat
add xsq XSQfile:9 Ecoli
```

In this example the analysis is run in three groups, with each group containing the data from one index. All groups receive the same analysis modules processing (for example, mapping, SNPs, and CNV). The data from each index is processed separately, and each index generates an independent set of results files.

Non-indexed data

When you use non-indexed XSQ files, all read-sets in those files become your input. (If you use a group name that includes other read-sets, the new read-sets are combined with the other read-sets already assigned to that group.)
Both the `add xsq` command and a groupsfile can add non-indexed data to your analysis:

```bash
add xsq xsqfile1 Human
add xsq xsqfile2 Human
```

Human `<path>`/XSQfile1 `<path>`/reference_file
Human `<path>`/XSQfile2

The effect of not using the `id` index argument with indexed data

Consider what happens if you add multiple indexed XSQ files to an analysis and do not use the `id` argument to specify individual indices. The effect is very different when you assign a group and you do not assign a group.

Example with a group

```bash
add xsq xsq1 groupX
add xsq xsq2 groupX
add xsq xsq3 groupX
```

For example, the above `add xsq` commands, with a group specified, define as input all the indices in the three XSQ files. All read-sets (in all three XSQ files) are treated as one specimen, and in the output files the data from all the read-sets is combined.

Example without groups

If you do not specify the group name, the shell assigns the data to groups based on index name. For example, consider the following commands, assuming each XSQ file contains indexed data:

```bash
add xsq xsq1
add xsq xsq2
add xsq xsq3
```

The shell assigns the read-sets to groups based on index name, using group names of `Group_id`. Each index is treated as a different specimen, the data for different indices are kept separate, and a separate output directory is used for each index. If the same index name occurs in more than one XSQ file, the read-sets for those indices are combined into one group. This combination might not be what you intend. For example, if files xsq1, xsq2, and xsq3 each contains indices 1–96, the data is processed as 96 separate groups. Group 1 contains index 1 from all 3 XSQ files, group 2 contains index 2 from all three XSQ files, and so on.

Granularity

The most specific set of data that can be defined as input to an analysis run is the data from one index in one XSQ file. For example:

```bash
add xsq xsqfile:62
```

Index IDs

Using the correct index IDs is important in your data input commands. Use the LifeScope™ Software shell `cat` command in the `/reads` level to find the index ID for your data. For example:

```bash
cd /reads
```

92
Run a grouped analysis

This section describes the behavior of a grouped analysis. If an analysis has read-set groups associated with it, the following jobs are run:

- One secondary sub-analyses: one job for all read-sets.
- A number of tertiary sub-analyses: one job per read-set group.

The following mechanisms add read-set groups to an analysis:

- The group option of the add xsq command.
- The set groups command.
- The -g groupsfile option of the lscope.sh run command.

Example of a groupsfile

The following gives an example of a basic groupsfile:

```
Human /data/reads/xsqfile1.xsq:3 <path>/reference_file1
Human /data/reads/xsqfile1.xsq:21
Human /data/reads/xsqfile2.xsq:51
```

The pattern is: `groupName XSQfile:id referencefile`. These field are separated by tabs.

The `referencefile` field is optional, except for the first entry of a group. If the `referencefile` field is empty, the last reference specified is used. All read-sets within one group must use the same reference (and the groupsfile must contain only one reference).

Run a multi-group analysis

This section describes how to do a run that analyzes reads from more than one sample, using a groupsfile that specifies multiple groups. All read-sets within the analysis must use the same reference.

Multi-group groupsfile

With a multi-group analysis, a single run of LifeScope™ Software processes the input data from more than one group. For each group, LifeScope™ Software keeps the data, processing, and results separate.

The following is an example groups file for a multi-group analysis:

```
lung /data/reads/xsqfile1.xsq:1 /refs/reference_file
lung /data/reads/xsqfile1.xsq:5
```
This read-set file defines three sample groups:

- lung – barcodes 1, 5, and 13
- brain – barcodes 2, 6, 9, 10, and 14
- kidney – barcodes 3, 11, and 15

With this read-set file, LifeScope™ Software in one run performs the processing of three separate runs, one for each group. All groups in the groupsfile must use the same reference, and the reference must be specified on the first line of the file.

Example commands for a multi-group analysis

This section describes the steps to run a multi-group analysis. The groupsfile used in “Multi-group groupsfile” is suitable for this example.

Follow these steps to run the multi-group analysis:

1. Login to lscape shell:
   
   lscape.sh shell -u user

2. Create a project and analysis, and open the analysis:
   
   cd /projects
   mk proj1
   cd proj1
   mk run1
   cd run1

3. Set the workflow:
   
   set workflow workflow_name

4. Set the groups:
   
   set groups path_to_groupsfile

5. (Optional) Set these parameters:
   
   set analysis.species.name species
   set analysis.assembly.name name

   An example of a species name is ecoli. An example of an assembly name is DH10B.

   If the assembly name is not provided, an assembly name of unknown is used.
   If the species name is not provided, the name of the reference file is used.

6. Ensure that the INI files in your analysis do not set the analysis.sample.name parameter. If this parameter is set in your INI files, the LifeScope™ Software shell overrides each sample group it processes with the value from the INI file.
7. Launch the multi-sample analysis:
   run

8. Optionally check the status of the analysis:
   ls
   This alternate method to check an analysis status is entered on the Linux shell:
   lscope.sh status -u user

Run a tertiary-only workflow

When you run a standard workflow and specify only BAM input files (not XSQ files), the framework automatically runs only the tertiary portion of the workflow. The secondary (mapping) portion of the workflow is not attempted.

Two examples are given, one with add bam commands and one with the set group command and a groupsfile.

With add bam commands

The following is an outline of the shell commands to run only the tertiary portion of a workflow:

cd projects
mk tertiaryProj
cd tertiaryProj
mk tertiaryRun
cd tertiaryRun
set workflow genomic.resequencing.frag
add bam /path/to/bamfile1 group1
add bam /path/to/bamfile2 group1
add bam /path/to/bamfile3 group1
set reference /path/to/reference
run

The genomic.resequencing.frag workflow includes both secondary and tertiary analyses. The secondary analyses take as input XSQ files, and the tertiary analyses take BAM files as input. Because the above example has only BAM files as input, LifeScope™ Software automatically skips the secondary analyses. the workflow runs only the tertiary analyses.

With a groupsfile

A tertiary-only workflow can also be run using a groupsfile and the set groups command. For example, outside of the LifeScope™ Software shell, create a groupsfile named tertiarygroups with this content:

group1 /path/to/bamfile1 /path/to/reference
group1 /path/to/bamfile2
group1 /path/to/bamfile3

The following is an outline of the LifeScope™ Software shell commands to run only the tertiary portion of a workflow, using the groupsfile:

cd projects
mk tertiaryProj
cd tertiaryProj
mk tertiaryRun2
cd tertiaryRun2
Run a Command Shell Analysis

Chapter 6

Run a tertiary-only workflow

```bash
set workflow genomic.resequencing.frag
set groups ./tertiarygroups
set reference /path/to/reference
run
```

With these commands, the groupsfile tertiarygroups must be in the current directory, which is the directory you are in when you give the lscope.sh command.

Run a multi-group tertiary analysis

This section describes how to run a tertiary-only workflow using BAM file input and using groups. With a multi-group analysis, a single run of LifeScope™ Software processes the input data from more than one group. For each group, LifeScope™ Software keeps the data, processing, and results separate.

BAM files with different library types can be combined as long as the workflow and tertiary modules can process them.

Two examples are given, one with add bam commands and one with the set group command and a groupsfile.

With add bam commands

This example uses add bam commands to create 2 groups:

```bash
set workflow yourworkflow
add bam /path/to/bamfile1 group1
add bam /path/to/bamfile2 group1
add bam /path/to/bamfile3 group1
add bam /path/to/bamfile4 group2
add bam /path/to/bamfile5 group2
set reference /path/to/reference
run
```

With a groupsfile

This example uses a groupsfile to create the same 2 groups. Outside of the LifeScope™ Software shell, create a groupsfile with this content:

```bash
group1 /path/to/bamfile1 /path/to/reference
group1 /path/to/bamfile2
group1 /path/to/bamfile3
group2 /path/to/bamfile4
group2 /path/to/bamfile5
```

These example LifeScope™ Software shell commands use the groupsfile:

```bash
set workflow yourworkflow
set groups /path/to/groupsfile
set reference /path/to/reference
run
```
Run an individual analysis

The optional examples download includes examples of how to run selected LifeScope™ Software analyses individually, without being part of a standard workflow or other sequence of analyses. Users new to LifeScope™ Software are recommended to use either a standard workflow or one of the optional examples, instead of the procedures described in the rest of this section.

This section describes how to run a single LifeScope™ Software analysis module, using the command shell.

Some options required you to create or edit PLN and INI files. See Appendix F, “Command Shell Control Files” on page 553 for information about these files.

- If you have already run the analysis, and the command shell has created the underlying PLN and INI files. You can use these PLN and INI files to run the analysis.
- You can optionally configure the analysis independently outside of the shell, using INI and PLN files that you create yourself.

This section describes these methods of running a single module:

- **In shell mode** – Open the LifeScope™ Software command shell, make a project and analysis, configure the analysis, and run the analysis in the command shell.
- **In run mode** – Use the lscope.sh run command to start the analysis. To use this command the PLN and INI file must already be created and customized. Two possibilities for creating the PLN and INI files are:
  - If you have already run the analysis in the command shell, then LifeScope™ Software has created these files for you.
  - You can optionally create the PLN and INI manually, for instance with a text editor.

The run mode command you use to run the analysis depends on whether you are running in the project repository or in the local file system.

- **Run in the project repository** – The run command is in this format:
  
  lscope.sh run plnfile -u user -p project -a analysis \
  -g groupsfile

  When you run in the projects repository with the lscope.sh run command, the results files are handled by the shell project repository. Where the results files appear on your local file system is configured during installation, and is given by the lifescpe.projects parameter in the <installdir>/server/server.properties file.

- **Run in the local file system** – The run command is in this format:

  lscope.sh run plnfile -u user

  When you run in the local file system with the lscope.sh run command, the results files appear in a subdirectory to your current directory on the file system. This command is executed in the same directory as plnfile.
Resume, re-run, and re-use

This section explains analysis reuse concepts and supported functionality, for both the command shell and GUI interfaces.

- **Resume** – A stopped job is restarted, without repeating modules that have successfully completed. The command shell also supports resuming a failed job (the GUI does not).
  
  **Note:** With the `resume` command, every module that is in progress at the time the job is stopped (or fails) is processed from the beginning of that module.

- **Re-use** – An existing analysis is used as the basis for a new analysis (with a new analysis name). You can optionally change the configuration of the new analysis.

- **Re-run** – An analysis is repeated (with the same configuration). All processing is redone.

### Table 15  Comparison of resume, re-use, and re-run functionality

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Interface</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Resume</strong></td>
<td>GUI</td>
<td>Command shell</td>
</tr>
<tr>
<td></td>
<td>Use the <code>Analysis &gt; Resume...</code> command.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Only stopped jobs are supported.</td>
<td>Use the shell <code>resume</code> command.</td>
</tr>
<tr>
<td></td>
<td>Supports stopped jobs and failed jobs.</td>
<td>-</td>
</tr>
<tr>
<td><strong>Re-use</strong></td>
<td>Use the <code>Analysis &gt; Reuse...</code> command.</td>
<td></td>
</tr>
<tr>
<td><strong>Note:</strong> Reuse... is available only for analyses with status of Running, Completed, or Failed.</td>
<td>Supported indirectly: Use Linux commands to make a copy of the analysis in the project directory, under a new analysis name.</td>
<td>You may optionally change the configuration of the new analysis.</td>
</tr>
<tr>
<td><strong>Re-run</strong></td>
<td>Supported indirectly: use the <code>Analysis &gt; Reuse...</code> command, and run the same analysis content under the new analysis name.</td>
<td>Not recommended, because the new run overwrites output from the previous run. See &quot;Re-run an analysis in the command shell&quot;.</td>
</tr>
<tr>
<td></td>
<td>Supported, but <em>not recommended</em>. Restart the analysis (same as the initial run).</td>
<td>You may optionally change the analysis configuration before restarting.</td>
</tr>
</tbody>
</table>

**Note:** If all secondary processing completed successfully and the resulting BAM files are correct, use the BAM input option (not the XSQ input option) when re-using or re-running a job. If you use the XSQ option, then all secondary processing is repeated. With the BAM option, only tertiary processing is done.

### Re-run an analysis in the command shell

When you re-run an analysis in the command shell, log files are appended and generated files are overwritten:

- Log files are appended. Log entries for the different runs can be distinguished by their time stamps.
- All output files are overwritten, and the output of the previous run is lost. Overwritten files include GFF output, generated BAMs, statistics files, and others.

The mechanism (*not recommended*) to re-run an analysis in the command shell is the following (these steps assume that the project and analysis are created and configured, and that the analysis previously ran to conclusion):
1. Open the project, and cd to the analysis.

2. Enter the run command, and press Enter or Return.

Review job status

The ways to check on your run vary according to how you started your run, as shown in the following table.

<table>
<thead>
<tr>
<th>How you started your run</th>
<th>How to check the status</th>
</tr>
</thead>
<tbody>
<tr>
<td>In LifeScope™ Software command shell, with the shell run command.</td>
<td>Open the LifeScope™ Software shell, cd to your project and then to your analysis, and execute the shell ls command:</td>
</tr>
<tr>
<td></td>
<td>Log in to the LifeScope™ Software command shell.</td>
</tr>
<tr>
<td></td>
<td>cd /projects</td>
</tr>
<tr>
<td></td>
<td>cd proj1</td>
</tr>
<tr>
<td></td>
<td>cd run1</td>
</tr>
<tr>
<td></td>
<td>ls</td>
</tr>
<tr>
<td></td>
<td>Output similar to the following is displayed:</td>
</tr>
<tr>
<td></td>
<td>--ini--</td>
</tr>
<tr>
<td></td>
<td>secondary/bamstats.ini</td>
</tr>
<tr>
<td></td>
<td>secondary/fragment.mapping.ini</td>
</tr>
<tr>
<td></td>
<td>secondary/global.ini</td>
</tr>
<tr>
<td></td>
<td>tertiary/dibayes.genome.ini</td>
</tr>
<tr>
<td></td>
<td>--readsets--</td>
</tr>
<tr>
<td></td>
<td>lung:DH10B_ColorSpaceOnly_Unbarcoded_50_Frag.xsq:1:DH10B.fasta</td>
</tr>
<tr>
<td></td>
<td>--jobs--</td>
</tr>
<tr>
<td></td>
<td>run1 [70%]</td>
</tr>
<tr>
<td></td>
<td>Output for completed jobs also has a success or failure notice:</td>
</tr>
<tr>
<td></td>
<td>run1 [100%] [JOB_FAILURE], or</td>
</tr>
<tr>
<td></td>
<td>run1 [100%] [JOB_SUCCESS]</td>
</tr>
<tr>
<td>In the Linux shell, with either the lscope.sh run or lscope.sh resume command, and the analysis is running in the projects repository.</td>
<td>At the Linux prompt, enter a lscope.sh status command, with the -p project and -a analysis options:</td>
</tr>
<tr>
<td></td>
<td>lscope.sh status -u rjones -p project \</td>
</tr>
<tr>
<td></td>
<td>-a analysis</td>
</tr>
<tr>
<td>In the Linux shell, with either the lscope.sh run or lscope.sh resume command, and the analysis is running in the current directory.</td>
<td>At the Linux prompt, enter a lscope.sh status command. Do not specify the -p project and -a analysis options:</td>
</tr>
<tr>
<td></td>
<td>lscope.sh status -u rjones</td>
</tr>
</tbody>
</table>
Chapter 6  Run a Command Shell Analysis

Review results

Reviewing the output files generated by your the LifeScope™ Software analyses is performed outside of the LifeScope™ Software command shell. You can either inspect your results files on the Linux cluster or transfer the results files to your local machine to review.

If your analysis is run in the projects repository, you can view results and download results files using the LifeScope™ Software UI. Refer to the LifeScope™ Genomic Analysis Software User Guide (Part no. 4476537).

If your analysis is not run in the projects repository, you must find your results files on the local file system. The results files are under the following directory:

<projects_repo_dir>/<user_name>/<project_name>/<analysis_name>/outputs

where projects_repo_dir is the location of the shell projects repository on the local file system, and user_name, project_name, and analysis_name are your shell user id, project name, and analysis name. Contact your LifeScope™ Software administrator for the location of the shell projects repository. The projects repository directory is also given by the lifescope.projects parameter in the file <installdir>/server/server.properties.

Review logs

You can typically use the LifeScope™ Software shell status commands (see “Review job status” on page 99) to determine your jobs success or failure status and to examine any error messages.

In the event you also want to look at your job’s run logs, by default they are located in the LifeScope™ Software shell projects repository. The location of this repository is set at installtime and is specific to your installation.

If you run your analysis in the current directory, your logs are in that directory.

For successful runs, the last line of summary.log ends in:

Analysis finished successfully.

When lscope.sh finishes, it prints the status of each job. If an error occurs, then the job’s summary log is also printed.

Error behavior and error messages

Case sensitivity  LifeScope™ Software shell levels, names, and commands are case sensitive. For example, the following sequence shows that a project named D157 exists, and the command cd d157 (with incorrect case) is not successful.

~/projects> ls
D152
D157
~/projects> cd d157
Server returned error status: 404, message: Project d157 does not exist
Unrecognized commands

The LifeScope™ Software shell display an error message “Invalid command” for unrecognized commands. In the following examples, the ls command is first miss-typed and then given in the wrong case. The string mkdir proj1, while a correct Linux command, is not recognized by the LifeScope™ Software command shell. (The correct LifeScope™ Software shell command is mk proj1.)

~/projects> ls
Invalid command
~/projects> LS
Invalid command
~/projects> mkdir proj1
Invalid command
~/projects>

Illegal usage

Error: String index out of range: -1

The message is thrown when the shell command is recognized but an illegal or unrecognized argument is used. Remember that shell commands do not behave like Linux commands. For example, the LifeScope™ Software shell ls command has a specific function, to list the contents of the current level. When mistakenly used to list the contents of a child or parent level, or used with wildcards, the error message appears:

~/projects> ls
D152
D157
~/projects> ls D157
Error: String index out of range: -1
~/projects/D157/> ls ..
Error: String index out of range: -1
~/projects/D157/> ls D1*
Error: String index out of range: -1
CHAPTER 7

7

Run a Standard Workflow Analysis

This chapter covers:

- Overview ................................................................. 103
- Standard workflows ..................................................... 103
- Run a standard workflow ............................................... 110
- Create a new workflow .................................................. 113
- Workflow control files .................................................. 115
- Run the tertiary portion of a workflow .............................. 117

Overview

A standard workflow is a built-in sequence of experiments that combine multiple LifeScope™ Software modules into a single command shell run. A workflow’s analyses correspond to a common biological application, such as resequencing.

Standard workflows

The standard workflows are provided for the following analysis types:

- Targeted resequencing
- Genomic resequencing
- Low frequency variants
- Whole transcriptome
- Small RNA
- ChIP-Seq
- MethylMiner™
**Note:** The standard workflow directory also includes two file conversion utilities for XSQ and BAM conversion. The conversion utilities are for internal use and do not work on the command line or in the command shell.

The workflows and the modules they execute are described in Table 17.

<table>
<thead>
<tr>
<th>Analysis type</th>
<th>Library type</th>
<th>Workflow name</th>
<th><strong>LifeScope™ Software modules involved</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>ChIP-Seq</td>
<td>Fragment</td>
<td>chip.seq</td>
<td>Secondary:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SAET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Fragment mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Mapping statistics</td>
</tr>
<tr>
<td>Genomic resequencing</td>
<td>Fragment</td>
<td>genomic.resequencing.frag</td>
<td>Secondary:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• <em>(Optional) SAET</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Mapping statistics</td>
</tr>
<tr>
<td></td>
<td>Mate-pair</td>
<td>genomic.resequencing.lmp</td>
<td>Secondary:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• <em>(Optional) SAET</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Paired mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Mark duplicates</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Mapping statistics</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Tertiary:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SNPs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Human CNV</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Small indels</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Annotations</td>
</tr>
</tbody>
</table>

Table 17 Standard workflow libraries, locations, and modules
## Standard workflow libraries, locations, and modules (continued)

<table>
<thead>
<tr>
<th>Analysis type</th>
<th>Library type</th>
<th>Workflow name</th>
<th>LifeScope™ Software modules involved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genomic resequencing</td>
<td>Paired-end</td>
<td>genomic.resequencing.pe</td>
<td>Secondary:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• (Optional) SAET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Paired mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Mark duplicates</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Mapping statistics</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Tertiary:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SNPs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Inversions</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Large indels</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Human CNV</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Small indels</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Annotations</td>
</tr>
<tr>
<td>Low frequency variant</td>
<td>Fragment</td>
<td>low.frequency.variant.frag</td>
<td>Secondary:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Fragment mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Mapping statistics</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Tertiary:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Enrichment</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SNPs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Annotations</td>
</tr>
<tr>
<td>Low frequency variants</td>
<td>Paired-end</td>
<td>low.frequency.variant.pe</td>
<td>Secondary:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Paired mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Mark duplicates</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Mapping statistics</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Tertiary:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Enrichment</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SNPs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Annotations</td>
</tr>
<tr>
<td>Methyl Miner</td>
<td>Fragment</td>
<td>methyl.miner.frag</td>
<td>Secondary:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• (Optional) SAET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Fragment mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Mapping statistics</td>
</tr>
<tr>
<td>Methyl Miner</td>
<td>Paired-end</td>
<td>methyl.miner.pe</td>
<td>Secondary:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• (Optional) SAET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Paired mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Mark duplicates</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Mapping statistics</td>
</tr>
</tbody>
</table>
### Standard workflow libraries, locations, and modules (continued)

<table>
<thead>
<tr>
<th>Analysis type</th>
<th>Library type</th>
<th>Workflow name</th>
<th>Lifescope™ Software modules involved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small RNA</td>
<td>small.rna</td>
<td></td>
<td>Secondary:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Small RNA mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Mapping statistics</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Tertiary:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Small RNA count</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Small RNA coverage</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Filter mapping counts</td>
</tr>
<tr>
<td>Targeted resequencing</td>
<td>Fragment</td>
<td>targeted.resequencing.frag</td>
<td>Secondary:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SAET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Fragment mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Mapping statistics</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Tertiary:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Enrichment</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SNPs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Small indels</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Annotations</td>
</tr>
<tr>
<td>Targeted resequencing</td>
<td>Paired-end</td>
<td>targeted.resequencing.pe</td>
<td>Secondary:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SAET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Paired mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Mark duplicates</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Mapping statistics</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Tertiary:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Enrichment</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SNPs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Small indels</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Annotations</td>
</tr>
<tr>
<td>Whole transcriptome</td>
<td>Fragment</td>
<td>whole.transcriptome.frag</td>
<td>Secondary:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• WT exon sequence extractor</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• WT splice junction extractor</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• WT fragment mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Mapping statistics</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Tertiary:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• WT counts</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• WT coverage</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Splice finder</td>
</tr>
</tbody>
</table>
Chapter 7  Run a Standard Workflow Analysis

Standard workflows

Note: For improved accuracy in the SNPs module, turn on the BAMStats parameter bamstats.enable.probe.position, to generate position error files and probe error files:

```
set bamstats.enable.probe.position 1 secondary/bamstats.ini
```

Turn off SAET

If you turn off SAET in a workflow, make sure your mapping INI file does not contain the line:

```
analysis.input.readset.file = ${analysis.output.dir}/saet/*.rrs
```

Comment out this line if it appears in your mapping INI file.

List standard workflows

List workflows using the LifeScope™ Software command shell

This section describes two methods of listing the standard workflows available on your installation of LifeScope™ Software.

Workflows are listed in the LifeScope™ Software command shell by doing an `ls` command in the `/workflows` virtual directory. To display a list of the standard workflows,

1. Open the LifeScope™ Software command shell:
   
   `lscope.sh -u username [-w password]`

2. Enter the workflow level:

   `cd workflows`

3. List the available workflows:
   
   `ls`

   The workflows are listed.

List workflows using Linux commands

---

Table 17  Standard workflow libraries, locations, and modules (continued)

<table>
<thead>
<tr>
<th>Analysis type</th>
<th>Library type</th>
<th>Workflow name</th>
<th>LifeScope™ Software modules involved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whole transcriptome</td>
<td>Paired-end</td>
<td>whole.transcriptome.pe</td>
<td>Secondary:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• WT exon sequence extractor</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• WT splice junction extractor</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• WT paired-end mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Mapping statistics</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bam.convert</td>
<td>Tertiary:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• WT counts</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• WT coverage</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Splice finder</td>
</tr>
<tr>
<td></td>
<td></td>
<td>xsq.convert</td>
<td>For internal use only.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: For improved accuracy in the SNPs module, turn on the BAMStats parameter bamstats.enable.probe.position, to generate position error files and probe error files:
Chapter 7  Run a Standard Workflow Analysis

Standard workflows

The LifeScope™ Software standard workflows are installed under the directory `<installdir>/etc/workflows`. To list the standard workflows, run this command in the Linux shell:

```
ls <installdir>/etc/workflows
```

### List a workflow's INI files and parameters

The command shell provides commands to list a workflow's control files and parameters. The `ls` command, when run at the level of a standard workflow, lists its INI files. The `cat` command is used to list the parameter settings of one of the workflow's INI files. The shell commands are shown in bold in the following example:

```
server:/workflows> cd small.rna
server:/workflows/small.rna> ls
secondary/bamstats.ini
secondary/global.ini
secondary/smallrnamapping.ini
tertiary/filtermapping.bamstats.ini
tertiary/global.ini
tertiary/srCount.ini
tertiary/srCoverage.ini
server:/workflows/small.rna> cat secondary/bamstats.ini
bamstats.input.dir=${analysis.output.dir}/smallrnamapping
bamstats.run=1
```

### Workflow directory structure

All standard workflows follow a consistent directory structure, shown in Table 18.

<table>
<thead>
<tr>
<th>Contents of top-level workflow directory</th>
<th>Contents of sub-directories</th>
</tr>
</thead>
<tbody>
<tr>
<td>analysis.pln</td>
<td>secondary pln</td>
</tr>
<tr>
<td>secondary</td>
<td>secondary ini</td>
</tr>
<tr>
<td></td>
<td>module1.ini</td>
</tr>
<tr>
<td></td>
<td>module2.ini ...</td>
</tr>
<tr>
<td></td>
<td>moduleN.ini</td>
</tr>
<tr>
<td>tertiary</td>
<td>tertiary.pln</td>
</tr>
<tr>
<td></td>
<td>tertiary ini</td>
</tr>
<tr>
<td></td>
<td>module1.ini</td>
</tr>
<tr>
<td></td>
<td>module2.ini ...</td>
</tr>
<tr>
<td></td>
<td>moduleN.ini</td>
</tr>
</tbody>
</table>

Do not delete or rename any of the following directories or files for your workflow:

- The secondary directory
- The tertiary directory
- The secondary.pln file
- The tertiary.pln file
- The analysis.pln file
Workflow modules and parameters

The analysis parameters used in the workflow modules control the behavior of the modules’ algorithms. Table 19 lists where the module parameters are described in this guide.

Table 19 Location of parameter descriptions

<table>
<thead>
<tr>
<th>Workflow</th>
<th>Module</th>
<th>Chapter</th>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genomic resequencing</td>
<td>Mapping</td>
<td>Chapter 9</td>
<td>“Fragment mapping parameters”</td>
<td>141</td>
</tr>
<tr>
<td></td>
<td>Mapping statistics</td>
<td>Chapter 9</td>
<td>“Mapping statistics”</td>
<td>159</td>
</tr>
<tr>
<td></td>
<td>SNPs</td>
<td>Chapter 10</td>
<td>“SNPs runtime parameters”</td>
<td>185</td>
</tr>
<tr>
<td></td>
<td>CNV</td>
<td>Chapter 11</td>
<td>“CNV module parameter descriptions”</td>
<td>211</td>
</tr>
<tr>
<td></td>
<td>Small indel</td>
<td>Chapter 13</td>
<td>“Small indel parameter description”</td>
<td>237</td>
</tr>
<tr>
<td></td>
<td>Annotations</td>
<td>Chapter 15</td>
<td>“Annotation parameters”</td>
<td>297</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See also the SNPs, CNV, and small Indel chapters.</td>
<td></td>
</tr>
<tr>
<td>Targeted resequencing</td>
<td>SAET</td>
<td>Chapter 8</td>
<td>“SAET parameters”</td>
<td>123</td>
</tr>
<tr>
<td></td>
<td>Mapping</td>
<td>Chapter 9</td>
<td>“Fragment mapping parameters”</td>
<td>141</td>
</tr>
<tr>
<td></td>
<td>Mapping Statistics</td>
<td>Chapter 9</td>
<td>“Mapping statistics”</td>
<td>159</td>
</tr>
<tr>
<td></td>
<td>Enrichment</td>
<td>Chapter 16</td>
<td>“Targeted resequencing read selection algorithm”</td>
<td>325</td>
</tr>
<tr>
<td></td>
<td>SNPs</td>
<td>Chapter 10</td>
<td>“SNPs runtime parameters”</td>
<td>185</td>
</tr>
<tr>
<td></td>
<td>Small indel</td>
<td>Chapter 13</td>
<td>“Small indel parameter description”</td>
<td>237</td>
</tr>
<tr>
<td></td>
<td>Annotations</td>
<td>Chapter 15</td>
<td>“Annotation parameters”</td>
<td>297</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See also the SNPs, and small Indel chapters.</td>
<td></td>
</tr>
<tr>
<td>Low frequency variants</td>
<td>Mapping</td>
<td>Chapter 9</td>
<td>“Fragment mapping parameters”</td>
<td>141</td>
</tr>
<tr>
<td></td>
<td>Mapping Statistics</td>
<td>Chapter 9</td>
<td>“Mapping statistics”</td>
<td>159</td>
</tr>
<tr>
<td></td>
<td>Enrichment</td>
<td>Chapter 16</td>
<td>“Targeted resequencing read selection algorithm”</td>
<td>325</td>
</tr>
<tr>
<td></td>
<td>SNPs</td>
<td>Chapter 10</td>
<td>“SNPs runtime parameters”</td>
<td>185</td>
</tr>
<tr>
<td></td>
<td>Annotations</td>
<td>Chapter 15</td>
<td>“Annotation parameters”</td>
<td>297</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See also the SNPs, and small Indel chapters.</td>
<td></td>
</tr>
<tr>
<td>Small RNA</td>
<td>Small RNA mapping</td>
<td>Chapter 23</td>
<td>“Small RNA mapping parameters”</td>
<td>431</td>
</tr>
<tr>
<td></td>
<td>Mapping statistics</td>
<td>Chapter 23</td>
<td>“Mapping statistics”</td>
<td>423</td>
</tr>
<tr>
<td></td>
<td>Small RNA coverage</td>
<td>Chapter 24</td>
<td>“Small RNA coverage parameters”</td>
<td>458</td>
</tr>
<tr>
<td></td>
<td>Small RNA counts</td>
<td>Chapter 25</td>
<td>“Small RNA counts parameters”</td>
<td>464</td>
</tr>
</tbody>
</table>
Run a standard workflow

Prepare to run a standard workflow

This section explains the preparation required to run a standard workflow.

Complete the prerequisites

Complete the applicable prerequisites described in Chapter 4, “Before You Begin” on page 59.

Determine your input and reference files

Before you can run the standard workflows you must know:
- The read and references files required by your analyses
- The regions of interest file, for targeted resequencing and low frequency variant analyses
- Any non-default module parameter values

Order commands correctly

Shell commands must be ordered in the following sequence:

1. Create the project.
2. Create the analysis.
3. Set the workflow.
4. Add the read-sets.
5. Set the reference.
6. Set the assembly name, species name, and other parameters, if required.
These rules apply to the analysis configuration commands:

- The `set workflow` command must precede all other analysis configuration commands.
- Read-sets must be added before the reference is set. (These commands add read-sets to your analysis: `set groups path`, `add xsq path`, and `add bam path`.)
- The `set reference` command must precede these two commands:
  - `set analysis.assembly.name`
  - `set analysis.species.name`

**Specify the regions of interest file**

Only for targeted resequencing and low frequency variant workflows, you must provide the regions of interest file:

```
set analysis.regions.file path
```

An absolute path is recommended for `path`. A relative path is interpreted relative to the directory you are in, in the Linux shell, when you execute the `lscope.sh` command.

**Note:** Targeted resequencing and low frequency variant analyses requires the regions of interest file, but the command shell does not enforce this requirement. Be sure you specify a regions of interest file when using a targeted resequencing or low frequency variant workflow.

**Run a workflow in the command shell**

This section describes how to run the fragment targeted resequencing workflow in the LifeScope™ Software command shell.

In the LifeScope™ Software command shell, running a standard workflow requires the following steps:

1. Create a LifeScope™ Software command shell project and analysis.
2. Identify the workflow to be executed on your data.
3. Identify your input data (your reads).
4. Identify the reference genome.
5. For targeted resequencing workflows, identify the regions of interest file.
6. Issue the `run` command to start your analysis.

**Example steps to run a standard workflow**

This section shows the steps to run a workflow in the LifeScope™ Software command shell. These steps use the targeted resequencing fragment workflow as an example. A description of each step is given in Table 20.

```
# log into the shell
lscope.sh shell -u username -w password
# import your data into the reads repository
cd /reads
import /data/xsq/DH10B_ColorSpaceOnly_Unbarcoded_50_Frag.xsq
# cd to the projects repository
cd /projects
# make a project, and open it
mk ecoli
cd ecoli
```
Chapter 7  Run a Standard Workflow Analysis

Run a standard workflow

# make an analysis, and open it
mk run1
cd run1

# define the analysis type
set workflow targeted.resequencing.frag

# define your input data
add xsq DH10B_ColorSpaceOnly_Unbarcoded_50_Frag.xsq

# specify the reference to be used
set reference hg19

# optionally change parameter defaults after this line or
# set analysis-specific parameters
set analysis.regions.file /data/results/readsdata/dh10b.bed

# list the configuration of the analysis
ls

# start the run
run

# list progress information about the run
ls

Paths (/data/results) and filenames are examples only. In your actual workflow runs you use paths to your own data and reference files.

Table 20 explains the purpose of the steps shown above. Information about your

Table 20  Description of sample shell commands to run a workflow

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>lscope.sh shell -u username</td>
<td>Log into the command shell. The shell prompts for your password if you do not provide it with the -w option.</td>
</tr>
<tr>
<td>cd /reads import /data/xsq/DH10B_ColorSpaceOnly_Unbarcoded_50_Frag.xsq</td>
<td>Go into the reads repository and import an XSQ file (the input data file). This command brings the XSQ reads file into the LifeScope™ Software reads repository.</td>
</tr>
<tr>
<td>cd /projects mk ecoli cd ecoli mk run1 cd run1</td>
<td>In the projects repository, create a project named ecoli, and in that project create an analysis named run1. Open the run1 analysis. Note: The open analysis (cd run1, in this case) command is important. The data, reference, and other configuration commands which follow apply only to the current analysis.</td>
</tr>
<tr>
<td>set workflow targeted.resequencing.frag</td>
<td>Define the workflow used with this analysis. The workflow in this example includes SAET, mapping, mapping statistics, enrichment, SNPs, small indel, and annotations.</td>
</tr>
<tr>
<td>add xsq DH10B_Unbarcoded_50_Frag.xsq</td>
<td>Define the XSQ file containing the input reads data for this analysis. Multiple input files are allowed.</td>
</tr>
<tr>
<td>set reference /data/results/results/references/DH10B.fasta</td>
<td>Define the reference file for this analysis.</td>
</tr>
<tr>
<td>set analysis.regions.file /data/results/readsdata/dh10b.bed</td>
<td>Define the regions of interest file, for targeted resequencing.</td>
</tr>
<tr>
<td>ls</td>
<td>Display the configuration of your analysis, including input reads, reference files, workflow, and select parameter settings.</td>
</tr>
<tr>
<td>run</td>
<td>Start the analysis run.</td>
</tr>
</tbody>
</table>
Chapter 7  Run a Standard Workflow Analysis

Create a new workflow

Information about your workflow job

The output of the `ls` command displays the analysis’ INI files, read-sets, and run completion percentage or completion status. Example output is shown below:

```
--ini--
  fragmentMapping.ini
  global.ini
--readsets--(group:file:index:reference)
    Group_1:case0030_Fragment.xsq:1:chr16.validated.fasta
--jobs--
  run1 [100%] [JOB_SUCCESS]
    fragmentMapping [100%] [JOB_SUCCESS]
```

Run a workflow in `lscope.sh run` mode

The `lscope.sh run` mode command executes the workflow without opening the LifeScope™ Software command shell. This command is used at the Linux command prompt.

The following command runs the small RNA workflow in run mode:

```
lscope.sh run <path>/small.rna/analysis.pln -p rna \
  -a run1 -u sjones
```

where `path` is the path to the workflow’s location in the LifeScope™ Software installation directory. The project and analysis are created if they do not already exist.

As an alternative, you can copy an existing workflow to a directory of your choice (use a Linux command such as `cp -p` to preserve the directory structure, or use the command shell `get workflow` command). Then use the following version of the command:

```
lscope.sh run ./analysis.pln -p rna -a run1 -u sjones
```

See “Use the command shell” on page 77 for more information about run mode.

Create a new workflow

This section describes how to create a new workflow. The new workflow appears in the shell `ls` command (but not in the LifeScope™ Software UI list of workflows). You run the new workflow following the directions for a standard workflow. The new workflow must have a unique name that both follows Linux file naming conventions and also does not start with an underscore character.

**Note:** When you create your own workflow, you can view the analysis and results files in the LifeScope™ Software UI. But the LifeScope™ Software UI does not support other analysis activities, such as reuse, with a custom workflow created on the command line.
This section describes two methods to create a new workflow, one with a Linux `cp` command, and the other with the shell `get workflow` command.

### With the Linux `cp` command

Using the Linux `cp` command method requires that you know the location of the LifeScope™ Software installation directory on the local file system. You do not open the LifeScope™ Software command shell.

The create a new workflow, follow these steps:

1. Identify an existing workflow similar to the one you want to create.
2. Change directory to the `<installdir>/etc/workflows` directory.
3. Create a directory with the name of your new workflow:
   ```
   mkdir newworkflow
   ```
   You use this name to define your analysis type with the `get workflow` command, and the workflow appears as this name when you list the contents of the `/workflows` virtual directory.
4. Copy an existing workflow into the new directory. Use a command such as `cp -p` in order to copy the directory structure as well as the files.
   ```
   cp -p small.rna/* newworkflow
   ```
5. Edit the PLN and INI files in `newworkflow` to customize it according to the requirements of your analysis.
   Do not rename the `secondary` or `tertiary` directory, and do not rename any of the PLN files (`analysis.pln`, `secondary.pln`, or `tertiary.pln`).

### With the shell `get workflow` command

The shell `get workflow` command copies the current workflow’s INI files, PLN files, and directory structure to your current directory on the local file system. This method requires you to have (or create) a LifeScope™ Software command shell analysis configured with the shell `set workflow` command.

The create a new workflow, follow these steps:

1. Identify an existing workflow similar to the one you want to create.
2. Log into the LifeScope™ Software command shell. Cd to your project and open the analysis. Or, if the project and analysis do not exist, create them with commands such as the following:
   ```
   cd /projects
   mk myproj
   cd myproj
   mk run1
   cd run1
   ```
3. If the analysis does not already have a workflow defined, use the shell `set workflow` command and name the workflow whose files you want to copy. For example:
   ```
   set workflow genomic.resequencing.pe
   ```
4. Use the `get workflow` command. For example:
   ```
   server:/projects/demo/getwf> get workflow
   Analysis workflow saved to /home/dir/demo/getwf
This command pulls the workflow files from the server and copies them to your local file system. The destination directory is determined by the Linux current working directory when the command shell was opened (where the lscope.sh shell command was given).

The following files and directories are copied:

- analysis.pln
- secondary (including its PLN and INI files)
- tertiary (including its PLN and INI files)

5. Edit the copied PLN and INI files to customize them according to the requirements of your analysis.

Do not rename the secondary or tertiary directory, and do not rename any of the PLN files (analysis.pln, secondary.pln, or tertiary.pln).

Workflow control files

This section describes the files that define the behavior of a standard workflow. The fragment genomic resequencing workflow, genomic.resequencing.frag, is used as an example. This section describes the workflow’s directory structure and the purpose of its PLN and INI files.

Do not edit INI or PLN files for the standard workflows. Such changes affect all workflow users, including both command shell and GUI users.

PLN files

Each standard workflows contains PLN files of the same name and at the same relative location. These PLN files are described in Table 21.

<table>
<thead>
<tr>
<th>File name and relative location</th>
<th>File contents and explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;workflow_name&gt;/analysis.pln</code></td>
<td>secondary.pln</td>
</tr>
<tr>
<td></td>
<td>tertiary.pln &lt; secondary.pln</td>
</tr>
<tr>
<td></td>
<td>First, runs the modules listed in the file secondary.pln.</td>
</tr>
<tr>
<td></td>
<td>If the secondary.pln runs complete successfully, then the modules listed in the file tertiary.pln are run.</td>
</tr>
<tr>
<td><code>&lt;workflow_name&gt;/secondary/secondary.pln</code></td>
<td>Lists the INI files involved in secondary analysis for this workflow.</td>
</tr>
<tr>
<td></td>
<td>Runs the LifeScope™ Software modules specified in these INI files.</td>
</tr>
<tr>
<td><code>&lt;workflow_name&gt;/tertiary/tertiary.pln</code></td>
<td>Lists the INI files involved in tertiary analysis for this workflow.</td>
</tr>
<tr>
<td></td>
<td>Runs the LifeScope™ Software modules specified in these INI files.</td>
</tr>
</tbody>
</table>
INI files

Table 22 shows the INI files involved in the genomic resequencing standard workflow, along with the PLN files that direct the execution of the workflow. The global.ini files define parameters that are imported into one or more module INI files in the same directory. (These parameters are considered global for the single directory, not for the entire workflow.)

Table 22 PLN and INI files in a standard workflow

<table>
<thead>
<tr>
<th>File name and relative location</th>
<th>File contents and explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;workflow_name&gt;/analysis.pln</code></td>
<td><code>secondary.pln</code>&lt;br&gt;<code>tertiary.pln &lt; secondary.pln</code>&lt;br&gt;First, runs the modules listed in the file secondary.pln.&lt;br&gt;If the secondary.pln runs complete successfully, then the modules listed in the file tertiary.pln are run.</td>
</tr>
<tr>
<td><code>&lt;workflow_name&gt;/secondary/secondary.pln</code></td>
<td><code>saet.ini</code>&lt;br&gt;<code>fragment.mapping.ini &lt; saet.ini</code>&lt;br&gt;<code>bamstats.ini &lt; fragment.mapping.ini</code>&lt;br&gt;Runs SAET using the parameters in the file saet.ini.&lt;br&gt;If SAET completes successfully, runs fragment mapping using the parameters in the file fragment.mapping.ini.&lt;br&gt;If the fragment mapping module completes successfully, runs bamstats mapping statistics.</td>
</tr>
<tr>
<td><code>&lt;workflow_name&gt;/tertiary/tertiary.pln</code></td>
<td><code>dibayes.genome.ini</code>&lt;br&gt;<code>cnv.ini</code>&lt;br&gt;<code>small.indel.ini</code>&lt;br&gt;<code>annotation.dibayes.ini &lt; dibayes.genome.ini</code>&lt;br&gt;<code>annotation.cnv.ini &lt; cnv.ini</code>&lt;br&gt;<code>annotation.small.indel.ini &lt; small.indel.ini</code>&lt;br&gt;Runs, in order, the (1) SNPs, (2) CNV, and (3) small indel modules, using the parameters in the respective INI files.&lt;br&gt;If the SNPs module completes successfully, runs SNPs annotations.&lt;br&gt;If the CNV module completes successfully, runs CNV annotations.&lt;br&gt;If the small indel module completes successfully, runs small indel annotations.</td>
</tr>
</tbody>
</table>

.run parameters

The *.run parameters control which LifeScope™ Software modules execute. The following are examples of run parameters used in standard workflows:

dibayes.run=1
small.indel.run=1
cnv.run=1

The allowed values for run parameters are 0 and 1:

- 0: Do not run the module.
- 1: Run the module analysis (default).

Because mapping is a prerequisite for all other runs, fragment.mapping.run may not be set to 0 unless it has already been successfully run to completion. You may set the .run parameters other modules to 0, under these conditions:
You do not want to run that module's analysis.
• That module's analysis has already been completed (for instance, if you are restarting the workflow run and repeating a particular module is not necessary).

The run parameters are set automatically by the shell during standard workflows. Accept the default for most use.

Run the tertiary portion of a workflow

This section describes running only the tertiary part of a standard workflow. With this method, you can optionally use input data with mixed library types.

First, set a standard workflow, such as targeted.resequencing.pe. You use one or more add bam commands to define the input data for the tertiary analysis. When run, LifeScope™ Software automatically only runs the tertiary portion of the workflow, because there are no XSQ read-sets in the analysis.

The following are partial example steps:

```bash
set workflow targeted.resequencing.pe
add bam file1.bam
add bam file2.bam
add bam file3.bam
set reference referencefile1
run
```

In this scenario, you can optionally combine BAM files of different library types. The type of data you have determines which type of standard workflow you must use. Follow these guidelines for the workflow to specify:

• If your input includes mate-pair data, use a .lmp workflow. Only the genomic.resequencing.lmp workflow supports mate-pair libraries.
• If your input does not include mate-pair data, and does include paired-end data, use a .pe workflow.
• If your input does not include any paired libraries, use a .frag workflow.

Note: The large indel and inversion modules do not support input BAM files containing mixed types.
Chapter 7  Run a Standard Workflow Analysis

Run the tertiary portion of a workflow
PART IV
Analysis Modules
Run the SOLiD® Accuracy Enhancement Tool

This chapter covers:

- Overview ................................................................. 121
- SAET usage guidelines .................................................. 121
- Examples of how to run SAET ......................................... 122
- SAET input files .......................................................... 123
- SAET parameters ........................................................ 123
- SAET output files ......................................................... 126
- Algorithm description ................................................... 127

Overview

The SOLiD® Accuracy Enhancement Tool (SAET) improves color call accuracy prior to mapping. This method implements a proprietary spectral alignment error correction algorithm that uses quality values and properties of color space. Significantly improving its performance over conventional error correction methods, the use of SAET produces more accurate mapping with both an increased number of reads with zero miscalls and an increased number of mapped reads. The SNP calling on error-corrected data results in an increase in true positive calls and a decrease in the number of false positive calls. Using the de novo assembly on error-corrected reads results in an increase that is as much as three to five times higher in average contig length.

SAET performance was tested on various datasets, including a large spectrum of genome sizes and complexities, coverages, read lengths, and experiment types. SAET shows similar performance on datasets sequenced from enriched regions or genomes of 1 kbp to 3 Gbp, with coverage from 10 to 4000x and a read length of 25 to 75 bp.

On average SAET requires up to 5 GB RAM for each 1 Gbp of sequenced region. SAET can be applied for diploid organisms. SAET should not be used for rare variant detection datasets where the frequency of rare variant may be less that twice that of the color calling error rate.

SAET usage guidelines

SAET is recommended for use with targeted resequencing and is on by default in targeted resequencing standard workflows. SAET has not been validated with other experiment types and may have insufficient memory to run. If you do choose to run SAET with other experiment types, you can do the following:
Examples of how to run SAET

Workflows using SAET

The following standard workflows provide examples of running SAET:

- targeted.resequencing.frag
- targeted.resequencing.pe

The optional examples download includes an example of how to run SAET by itself, as an individual analysis that is not part of a standard workflow.

See the following sections for information on these methods:

- As part of a workflow: Chapter 7, “Run a Standard Workflow Analysis” on page 103
- As an individual analysis: Appendix H, “Demo Analyses” on page 567

SAET example in a workflow

The targeted resequencing Fragment standard workflow gives an example of running SAET on with targeted resequencing analyses. The analyses are:

- SAET
- Mapping
- Enrichment
- SNPs
- Small indels
- Annotations

SAET error corrects reads prior to mapping. This correction significantly improves mapping quality and the quality of variant calls.

Note: If you turn off SAET, make sure your mapping INI file does not contain the line:

```
analysis.input.readset.file = ${analysis.output.dir}/saet/*.*
```

Comment out this line if it appears in your mapping INI file.

SAET usage in standard workflows

SAET is on by default in the targeted resequencing workflows.

In the genomic resequencing, ChIP-Seq, and MethylMiner™ workflows, SAET is off by default. To enable SAET in these workflows, change the parameter `saet.run` from 0 to 1 with either one of the following methods:

- If you edit the INI file, change `saet.run=0` to `saet.run=1`.
- If you configure the workflow in the command shell, use the following shell command (after setting the workflow):

```
```
set saet.run 1 secondary/saet.ini

SAET input files

By default, SAET takes an input file of SOLiD® system reads in XSQ format, and an expected length of the sequenced region, such as the:

- Genome size for whole genome sequencing
- Size of enriched region in targeting resequencing
- Size of transcribed region in whole transcriptome sequencing

The required input file is in XSQ format, which is typically generated by 5500 Series Genetic Analysis Systems. The file must contain reads in color-space that require correction. SAET constructs a spectrum of all k-mers from the set of all reads. Then each read is corrected individually if necessary. (The value $k$ is the seed length. The default can be overridden with the parameter $saet.seed$.)

In targeted resequencing, the SAET module accepts as an input the BED, GFF, or TXT file with segments of target regions. In this case the $refLength$ parameter is computed from the targeted regions information.

SAET parameters

This section describes the parameters that control the runtime behavior of the SAET module. In order to change a parameter value, use the $set param$ shell command. For example, to turn off $qvupdate$, change the value of the parameter $saet.qvupdate$ with this shell command:

set saet.qvupdate 0 secondary/saet.ini

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Required parameters</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>saet.refLength</td>
<td>0</td>
<td>Expected length of sequenced (or enriched) DNA region. For example, 4,600,000 for the E.Coli 4.6 MB genome or 30,000,000 for the entire Human Transcriptome. Allowed values: Integers $\geq$ 0. The default value for $saet.refLength$ varies by workflow. For example:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Default in genomic resequencing workflows: 2800000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Default in targeted resequencing workflows: 30000000</td>
</tr>
<tr>
<td><strong>Parameters required for targeted resequencing</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>saet.on.target.ratio</td>
<td>0.5</td>
<td>The expected ratio of reads that come from the targeted region. Allowed values: Floats 0.0–1.0.</td>
</tr>
</tbody>
</table>

Optional parameters
### SAET parameters

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
</table>
| saet.qvupdate      | 1             | Whether or not to update the quality value of modified calls. Allowed values:  
|                    |               | - 0: Do not update the quality value of modified calls. This option is  
|                    |               | approximately twice as fast.  
|                    |               | - 1: Update the quality value of modified calls. Required for SNPs.  |
| saet.qvhigh        | 25            | Correction is applied to calls with a quality value below the value of this  
| saet.seed          | 0             | Size of k-mer (>=5) used in spectrum construction and error correction.  
|                    |               | (If set to 0, then the value is computed internally.)  
|                    |               | Allowed values: Integers 0, 5-28.  |
| saet.suppvotes     | 3 for targeted reseq; 2 for other analyses | The minimum number of k-mer votes required to make a correction.  
|                    |               | Allowed values: Integers >= 1.  
|                    |               | For datasets forming coverage less than 50x, the recommended value is  
|                    |               | 1 or 2. For high coverage data, the value can be increased up to k-1.  
|                    |               | Increasing this value results in a lower number of over-corrections in  
|                    |               | repetitive regions and a lower number of corrections over all. The  
|                    |               | maximum recommended value is 7.  |
| saet.trustfreq     | 0             | The lowest multiplicity of the seed to be included in the spectrum. (If set  
|                    |               | to 0, then the value is computed internally.)  
|                    |               | Allowed values: Integers >= 0.  |
| saet.localrounds   | 0             | Maximum number of allowed corrections per read. (If set to 0, then the  
|                    |               | value is set to [readLength/8]). Reduce if over-corrections are observed,  
|                    |               | or increase if under-corrections are observed.  
|                    |               | Allowed values: Integers 0–9.  |
| saet.globalrounds  | 1 for targeted reseq and reseq of large genomes; 2 for general reseq of small genome | The error correction step is repeated the provided number of times.  
|                    |               | Reduce if over-corrections are observed, or increase if under-corrections  
|                    |               | are observed. Allowed values: 1, 2, or 3.  |
| saet.trustprefix   | 0             | Position in the read at which the error rate inflates, for instance, 35-40 for  
|                    |               | 50bp long reads. If set to 0, then the value used is 0.8*readLength.  
|                    |               | Allowed values: Integers >=k, where k is the size of the k-mer (either set  
|                    |               | by saet.seed or computed internally).  |
| saet.nosampling    | 0             | Disables random sampling in spectrum building. If set to 0, then for large  
|                    |               | datasets (coverage > 300x), a subset of reads is used in spectrum building.  
|                    |               | Allowed values:  
|                    |               | - 0: Do not disable random sampling in spectrum building.  
|                    |               | - 1: Disables random sampling in spectrum building.  |

### Resource parameters

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>processors.per.node</td>
<td>8</td>
<td>Number of parallel threads used to run SAET on a node.</td>
</tr>
</tbody>
</table>
| saet.minreads.per.node  | 1000000       | If SAET is run on multiple nodes, then each node receives at least this  
|                         |               | amount of reads to process.  |
The processing time and the level of error correction aggressiveness depend mainly on
the values defined for saet.localrounds and saet.globalrounds. The
saet.localrounds parameter allows you to correct up to saet.localrounds
errors in a read by using a precomputed spectrum. The saet.globalrounds
parameter recomputes the spectrum after each global round, and allows you to correct
up to saet.localrounds errors in a read based on the precomputed spectrum.

SAET is designed to reduce the error rate in the reads generated by the SOLiD®
System. Reducing the error rate increases the number of mapped reads for
resequencing analyses, which can for example lead to an increase in true positive SNP-
calls and a decrease in false positive SNP-calls. (If the SAET is used for SNP calls, the
saet.qvupdate parameter must be set to 1.) Use the saet.qvhigh parameter to
restrict corrections to positions that have a quality value below the saet.qvhigh
threshold.

The optional parameters available in SAET allow you to modify the algorithm’s
behavior. For example, you might find that the globally computed cutoff for frequency
of trusted seeds does not meet your purpose. For example, the cutoff might be too low,
causing too many junk seeds to be considered correct. However, if the cutoff is too
high, then many correct but low-frequency seeds are inappropriately filtered out. You
can change the saet.trustfreq parameter to overwrite the estimated frequency
cutoff.

If you trust the quality of your reads more or less than the value specified in the
saet.trustprefix parameter, then make the corresponding changes to the value. If
SAET makes many corrections in the regions of reads with highly packed errors, you
can increase saet.suppvotes to improve the tendency to correct only isolated
errors. SAET provides options for reading and writing spectrum files. The options
enable you to build spectrum from better-quality reads and use the spectrum to correct
lower-quality reads. You can also build a spectrum from a reference, or build a
spectrum from multiple files, such as data from multi-run experiments.
SAET internal parameters

Table 24 lists SAET parameters that we do not recommend changing.

Table 24 SAET internal parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>saet.run</td>
<td>1</td>
<td>Whether or not to run the SAET module. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Do not run SAET.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Run SAET during this analysis.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The run parameter is set automatically by the shell during a standard</td>
</tr>
<tr>
<td></td>
<td></td>
<td>workflow. Accept the default for most use.</td>
</tr>
<tr>
<td>saet.outspecbin</td>
<td>0</td>
<td>Whether or not to output a spectrum in binary format in the file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>${task.out.dir}/analysis.input.readset.file.spect.bin</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Do not output the spectrum.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Output the spectrum.</td>
</tr>
<tr>
<td>saet.inspecbin</td>
<td>—</td>
<td>Uses pre-generated file(s) with spectrum (in binary format) for error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>correction. Use a comma to separate multiple files.</td>
</tr>
<tr>
<td>saet.maxtrim</td>
<td>0</td>
<td>Trims erroneous tails of reads up to first trusted seed or up to maxtrim.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If the remaining part of a read is shorter than seed size + 2, then the read</td>
</tr>
<tr>
<td></td>
<td></td>
<td>is discarded. Do not use this option together with the saet.qvupdate</td>
</tr>
<tr>
<td></td>
<td></td>
<td>parameter.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If this parameter is set to 0, then no trimming is done.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allowed values: Integers (\geq 0).</td>
</tr>
<tr>
<td>saet.trimqv</td>
<td>—</td>
<td>Trims erroneous tails of reads up to first trusted seed or up to a position</td>
</tr>
<tr>
<td></td>
<td></td>
<td>with quality value higher than trimqv. If the remaining part of a read is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>shorter than seed size + 2, then the read is discarded. Do not use this</td>
</tr>
<tr>
<td></td>
<td></td>
<td>option together with the saet.qvupdate parameter.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If this parameter is set to 0, then no trimming is done.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allowed values: Integers (\geq 0).</td>
</tr>
<tr>
<td>saet.log</td>
<td>saet.log</td>
<td>Log file name.</td>
</tr>
<tr>
<td>saet.progress.file</td>
<td>.progress</td>
<td>Progress file name.</td>
</tr>
<tr>
<td>saet.progress.start</td>
<td>0</td>
<td>Progress start point. Allowed values: Integers 0–100.</td>
</tr>
<tr>
<td>saet.progress.end</td>
<td>100</td>
<td>Progress end point. Allowed values: Integers 0–100.</td>
</tr>
<tr>
<td>maximum.workers</td>
<td>4</td>
<td>Maximum number of nodes that can be used by SAET.</td>
</tr>
</tbody>
</table>

SAET output files

SAET writes the corrected reads and quality values into new XSQ files.

The SAET output files are used as input to mapping analyses.
Algorithm description

SAET implementation

SAET implements a modified version of the spectral alignment error correction algorithm proposed by Pevzner et al., 2001. SAET extends the original algorithm by taking quality values and properties of color-space into account, significantly improving the performance of the original method. The method corrects reads given a set of k-mers (seeds) spectrum. Given a set of reads ($R$), and a frequency threshold ($m$), the spectrum is defined as the set of all $k$-mers that appear at least $m$ times in reads from $R$. The set of such trusted $k$-mers approximates the set of all $k$-mers in the genome. A read is defined as error-free if all of its $k$-mers are trusted. If not all $k$-mers are trusted, SAET attempts to make a read error-free by mutating a few colors in the read. SAET only considers substitution mutations since there no insertions or deletions in SOLiD® system reads. When making error corrections in the read, SAET gives priority to positions with missing calls, followed by positions that have lower-quality values. In addition, SAET avoids correcting two adjacent colors with color-call quality values above a certain threshold, in order to preserve any SNP evidence. SAET computes the $k$-mer size and the multiplicity of trusted $k$-mers, based on the estimated reference length and number of reads in the dataset. Using default parameters, the color call changes introduced by SAET are expected to be 99.99% accurate.

SAET supports multi-core runs. SAET provides a mechanism for multi-node runs by creating sub-spectra from subsets of reads. SAET merges the sub-spectra into the final spectrum that can be used for correction of any subset of reads.

Phases

The SAET algorithm has three steps as described below.

Spectrum build

In this step, a temporary file with a spectrum constructed from all of the reads is generated in the output directory. This process is very fast and does not require large RAM resources. However, SAET requires a large amount of disk space (~1 GB for each 1Mbp of genome). SAET creates temporary swap files when memory usage approaches 2 GB.

Error correction

The error correction step takes each read in the input file and attempts to correct it if any of its $k$-mers are not present in the spectrum. This step is the most time-consuming. LifeScope™ Software runs SAET on multiple cores to improve the error correction speed.

Update of quality values for corrected calls

The quality value of each corrected position is replaced with a quality value derived from spectral probability. This step is time-consuming and can be disabled if the update of quality values is not required. (To disable, set saet.qvupdate to 0.)

SAET run times

The SAET processing time depends on the input size and number of global and local rounds. With a large number of global and local rounds, an expected throughput is 1 Gbp per hour on a single core. The amount of used RAM is approximately 5 GB for 1 Gbp of genome size.
Algorithm description

Chapter 8  Run the SOLiD® Accuracy Enhancement Tool

Algorithm description
This chapter covers:

- Overview ................................................................. 129
- Examples of how to run a mapping analysis .......................... 130
- Input files ...................................................................... 131
- Stages of mapping ........................................................... 131
- Quality alignments and unique alignments .............................. 140
- Fragment mapping parameters .......................................... 141
- Mapping algorithm ......................................................... 145
- Pairing algorithm ............................................................ 155
- Indel sizes ...................................................................... 158
- Mapping output files ....................................................... 159
- Mapping statistics ............................................................ 159
- Mapping statistics output files .......................................... 163
- FAQ – Mapping ............................................................... 173
- FAQ – Pairing ................................................................. 176
- FAQ – BAMStats ............................................................. 178

Overview

This chapter describes how to perform resequencing mapping. The mapping analysis results files are required as input for tertiary analysis modules.

During secondary analysis LifeScope™ Software performs the following steps:

- Maps or aligns reads to a reference genome.
- Accepts one or more input XSQ (eXtensible SeQuence) files to generate one or more sorted BAM files containing aligned reads.
- For mate-pair or paired-end libraries, pairs alignments that have the same bead ID.
- *(Optional)* Generates mapping statistics.

The 5500 Series Genetic Analysis Systems generates XSQ files, optionally with ECC (Exact Call Chemistry) reads. If an ECC primer round has been performed, the XSQ output also includes the sequence information in base space, in addition to color space. The ECC primer round is recommended during sequencing, for significant increases in sequencing accuracy.

XSQ is an extensible binary file format for storing sequence data, and supports multiple independent reads at the same position in a fragment.
Examples of how to run a mapping analysis

In a standard workflow

The following standard workflows provide examples of running the mapping module:

- genomic.resequencing.frag
- genomic.resequencing.lmp
- genomic.resequencing.pe

The following are example shell commands using reads and references from the LifeScope™ Software repository. This example turns off tertiary modules in the workflow.

```
# log into the shell
lscope.sh shell -u username -w password
# cd to the projects repository
cd /projects
# make a project, and open it
mk ecoli
cd ecoli
# make an analysis, and open it
mk run1
cd run1
# define the analysis type
set workflow genomic.resequencing.pe
# define the input
add xsq /data/xsq/run0209a_50_PE.xsq
add xsq /data/xsq/run0209b_50_PE.xsq
# specify the reference to be used
set reference hg19
# these commands turn off the tertiary modules
set cnv.run 0 tertiary/cnv.ini
set dibayes.run 0 tertiary/dibayes.ini
set inversion.run 0 tertiary/inversion.ini
set large.indel.run 0 tertiary/large.indel.ini
set small.indel.run 0 tertiary/small.indel.ini
# optionally change mapping parameter defaults after this line
ls
# list the analysis configuration
ls
# run the analysis
run
# list progress information for the analysis
ls
```

Dummy XSQ file names are used in this example.

In order to change a mapping parameter value in your analysis, use the `set param` shell command after the line `# optionally change mapping parameter defaults after this line`. For instance, to turn on the `mapping.in.base` parameter, use this shell command:

```
    set mapping.in.base 1 secondary/pair.mapping.ini
```

This command shows how to turn on the `mapping.in.base` parameter for the `genomic.resequencing.frag` workflow:
Chapter 9  Run a Resequencing Mapping Analysis

Input files

The mapping module accepts as input one or more input XSQ files. The input reads can have different read lengths but they must be of the same library type. For example, LMP read-sets of length 50 and 60 can be processed together. The XSQ files processed in a single analysis can not have different library types. In one mapping analysis, the input must be of only one library type, either Fragment, LMP, or PE.

Plan your input read-sets

Plan your analysis input carefully. The way you define your reads input affects the behavior of your analysis. The following factors control how your input data is analyzed:

- **Index (barcode) IDs** – Using an index ID restricts your input to the reads data of one or more indices.
- **Grouping of reads** – Each group of reads is analyses together as one specimen. The output data for a group is combined into one set of results files.
- **Multiple sample runs** – Unrelated reads can be processed together in one run of LifeScope™ Software, but analyzed separately as separate input data.

See “Define input data” on page 90 for more information on designing adding input data to your analysis.

Mapping one tag of paired data is not supported.

Legacy data

If the data you want to process with LifeScope™ Software is in CSFASTA and QUAL files, these files must be converted to the XSQ file format, through one of the following methods:

- The LifeScope™ Software UI handles converting these files to the XSQ format before mapping.
- If you are using the LifeScope™ Software command shell, you must first convert the CSFASTA and QUAL files to the XSQ format. See Appendix D, “XSQ Tools” on page 519 for information on the standalone XSQ converter.

Stages of mapping

The mapping algorithm involves following stages:

```bash
set mapping.in.base 1 secondary/fragment.mapping.ini
```

See Chapter 7, “Run a Standard Workflow Analysis” on page 103 for more information on running standard workflows.

As a demo analysis

The optional examples download also includes examples of how to run the mapping module by itself, as an individual analysis that is not part of a standard workflow. Fragment and paired examples are included. The use of a standard workflow is preferred.

See Appendix H, “Demo Analyses” on page 567 for more information on running a demo analysis.
• **Scatter** – The process of distributing reads to the available compute nodes, to achieve parallelization.

• **Mapping** – Each tag is aligned to the reference.

• **Pairing** – With paired-end data, two tags are individually mapped using the same mechanism as fragment mapping. Then pairing is performed. Fragment data skips this stage.

• **BAM generation** – The process of converting the output of mapping and pairing to BAM format. For color-space data, the base translation step is also included. Due to the scattering step, the BAM files at each compute node are a sub-set of the final BAM file and are referred to as baby BAMs.

BAM generation involves these steps:
- **Ma2BAM** – A conversion step, from MA format to BAM format, for fragment data only.
- **Pa2BAM** – A conversion step, from PA format to BAM format, for paired data only.
- **Refcor** – Reference-assisted color-to-base translation.
- **Sort BAM** – Sorting the BAM file from bead-id order to coordinate order.
- **Merge BAM** – Merging the mini-BAM files created in the sort phase into the final output BAM files.

*Figure 1* shows the phases involved in mapping fragment data. *Tag1* refers to the input data in the F3 strand.

---

*Figure 1* The phases within the mapping module, when mapping fragment data
Figure 2 shows the phases involved in mapping paired data. Tag1 and Tag2 refer to the input data on the two different strands.

Figure 2 The phases within the mapping module, when mapping mate-pair or paired-end data

Scatter

The scatter stage distributes reads to the available compute nodes, in order to gain the efficiency benefits of parallel computing.

The mapping module can accept multiple read-sets, which are allowed to have different read lengths. The input set of read-sets from multiple XSQ files is first split based on read length. Fragment read-sets are partitioned into separate categories based on these read length categories:

- $0 \leq \text{Length} < 25$ (uses first|second.map.scheme.unmapped|repetitive.1)
- $25 \leq \text{Length} < 35$ (uses first|second.map.scheme.unmapped|repetitive.25)
- $35 \leq \text{Length} < 50$ (uses first|second.map.scheme.unmapped|repetitive.50)
- $50 \leq \text{Length} < 60$ (uses first|second.map.scheme.unmapped|repetitive.60)
- $60 \leq \text{Length} < 75$ (uses first|second.map.scheme.unmapped|repetitive.75)
- $75 \leq \text{Length}$ (uses first|second.map.scheme.unmapped|repetitive.75)
- Trimmed reads, with variable read lengths (uses first|second.map.scheme.unmapped|repetitive.variable)

Mate-pair read-sets are partitioned into separate categories based on the pair length. Example categories are:

- $(0.0) \leq (\text{Length1},\text{Length2}) < (25,25)$
- $(25,25) \leq (\text{Length1},\text{Length2}) < (50,50)$
- $(50,50) \leq (\text{Length1},\text{Length2}) < (60,60)$
- $(60,25) \leq (\text{Length1},\text{Length2}) < (75,35)$

LifeScope™ Genomic Analysis Software Command Shell User Guide
The reads from each category are scattered into multiple jobs based on the parameters `fragmap.minreads.per.node` and `fragmap.number.of.nodes`. These parameters are described in the performance section (see “Mapping performance” on page 144).

**Fragment mapping**

The Map phase (also known as mapreads) includes map 1, map 2, and maMerge. Map combines information from reads in XSQ-formatted input files and a FASTA-format reference file, to generate alignment information in an intermediate file format known as match file format (.ma). Unmapped or poorly mapped reads from the first stage of mapping (map 1) can enter another mapping stage (map 2). Both these mapping stages have independent controls with respect to the mapping scheme and the gapped alignment algorithm. Since a poorly-mapped read can have alignments reported from map 1 and map 2, the maMerge is invoked to report the best alignments from the two mapping stages.

In the default setting, two mapping stages are invoked. The first one does ungapped alignment only. The second one does gapped alignments only. The second stage is switched off for read-sets with lengths less than the value of the parameter `min.length.for.aggressive.gapped.mapping` (default is 50). The maMerge step is not invoked if only one stage of mapping is run.

Alignment finding and gap finding use a proprietary algorithm. The presence of a gap can not be investigated in the anchor region. In general, for read length L and anchor defined by A.B.C,

- If A is zero, mapping looks for gap from position A+1 to L.
- If C is zero, mapping looks for gap only at position 0 to C.

Each map step is a multi-threaded executable that can handle one or more XSQ files of one library type. The unmapped and poorly reads from the first mapping stage are XSQ-formatted so that they can be processed by subsequent stages of mapping. The match file format is an internal file format that stores alignment and gap information. The MA files from two stages of mapping and maMerge are all generated in the scratch folder. The parameters to control mapping and gap finding algorithms are in “Mapping algorithm” on page 145. For details of the algorithm, see “Mapping schemes” on page 151.

**Mate-pair mapping**

Each tag is mapped using a mapping protocol similar to fragment mapping. The match files (MA files) are provided to pairing stage.

**Pairing**

The pairing module matches pairs of reads from the F3 and R3 mapping results of a mate-pair run. Pairing also matches reads from the F3 and F5-P2 files of a paired-end mapping run.

Pairing is divided in two stages. In the first stage, an estimate of the insert size distribution is made. Pairing module accepts the MA files, calculates the insert size from uniquely mapped beads and writes out a BINNED file (*.binned) containing insert size distribution for each read-set. A perl script is invoked on the collection of BINNED files to calculate the insert size statistics (max, min, average, standard deviation) per read set. Output of the perl module is a tab-separated file (ISS, *.iss) containing insert size statistics per read set. This ISS file is passed to pa2BAM module.
that loads the average and standard deviation information to the appropriate BAM file. Pairing executable is invoked one more time with this ISS file and the MA files. This time pairing, rescue, and indels are called. Output of this pairing stage is a PA file, in an internal format.

The pairing algorithm performs the following steps for each pair of reads:

1. It finds all good AAA pairs based on the order, orientation, and distance between the two reads. (See the genomic classifications listed in Tables 34 and 35 on page 176 and 177 for information about AAA and other three-letter categories.)

2. If no AAA pair is found, and a reference sequence is available, the algorithm performs mate-pair rescue using hits to one tag as an anchor, and then scanning for the “sister tag” in the region predicted by the library insert size.

   **Note:** Either F3 or R3 tags can serve as the anchor, as long as the number of hits is below the Z threshold. The Z threshold is determined by the value entered for the `pairing.gapped.max.hits` parameter. If the F3 tag has x alignments, the R3 tag has y alignments, and both tags have fewer than Z hits, then the x+y anchor candidates are examined (see Figure 3, Mate-pair rescue example).

![Figure 3 Mate-pair rescue example](image)

3. It determines if a AAA pair is unique. If multiple AAA pairing candidates exist for a given pair of tags, then the pair whose best pairing candidate scores at least x higher than the second-highest scoring pairing candidate is still considered unique, and all other AAA candidates with lower scores are discarded. The value of x is determined by the setting of the `pair.uniqueness.threshold` parameter.

4. For non-AAA pairs where both tags have unique hits, the algorithm performs additional classifications based on the strand, distance, and orientation of the tags.

   If either of the two tags has multiple mapping locations, but the highest mapping score is more than half of x higher than the second-highest score, then the location with the highest score is still considered unique and all other locations are discarded.

The pairing module supports paired-end experiments in addition to mate-pair experiments. In paired-end experiments, the actual sequencing is done on the same strand in opposite directions. However, because the representation in the sequencing output file follows the convention of representing the sequence from 5’ to 3’, the matching and pairing modules attempt to match the F5 and F3 tags on different strands, facing each other. In addition, a distance constraint determined by insert size is satisfied (see Figure 4). The pairing algorithm follows the same steps as the mate-pair algorithm. Steps 1 and 2 are modified to enforce the different order and
orientation requirement. If you convert matching positions for F3 tags to the opposite strand, then paired-end pairing is performed the same way as it is with mate-pair. The three-letter classification is defined the same way in mate-pair and pair-end. (See Table 34 on page 176 and Table 35 on page 177 for the genomic classification tables.)

![Figure 4 Paired-end tags example](image)

**Mark duplicates**

For paired data only, the mark duplicate module checks for and marks duplicate bead ids. Except for the highest quality alignment among the duplicates, the duplicate alignments are marked for filtering, and are by default ignored by tertiary modules. Each duplicate alignment is marked by setting to 1 the $0 \times 400$ bit of its FLAG field.

For analyses with grouped data, the modules considers all BAM files for one sample, and finds duplicate beads across the multiple BAM files for the sample.

The mark duplicate module is enabled by default in whole genome workflows with paired data.

**Mark duplicate parameters**

Table 25 lists mark duplicate parameters.

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Optional parallelization parameters</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>process.one.bamfile.only</td>
<td>true</td>
<td>Process BAM files separately for marking duplicates. Set to false to process all BAM files of a sample together. See &quot;Mark duplicates on multiple BAM files&quot; on page 137. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• true: Process one BAM file at a time.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• false: Process all BAM files of a sample together.</td>
</tr>
<tr>
<td>markduplicates.input.dir</td>
<td>—</td>
<td>The directory containing mapping output files, which also are the BAM files input to mark duplicates. You typically do not change this parameter.</td>
</tr>
<tr>
<td><strong>Optional resource parameters</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>memory.request</td>
<td>15gb</td>
<td>The total memory, in gigabytes, that is available for map reads, per compute node. Include the units gb in the setting.</td>
</tr>
<tr>
<td>java.heap.space</td>
<td>1024</td>
<td>Dynamic memory requirement, in megabytes. Do not enter a unit with the numeric value.</td>
</tr>
<tr>
<td>markduplicates.max.nodes</td>
<td>4</td>
<td>The number of compute nodes available for this step.</td>
</tr>
</tbody>
</table>
Mark duplicates on multiple BAM files

Number of duplicates called

If you process multiple BAM files together and the same bead ID occurs in different BAMs, those beads are treated as duplicates. This is the behavior with the setting process.one.bamfile.only=false. See the mark duplicates FAQ “Why does LifeScope™ Software report more duplicate beads than Picard reports?” on page 178.

The process.one.bamfile.only parameter

The setting of this parameter affects your processing time and memory usage.

The multiple BAM option (process.one.bamfile.only=false) is appropriate for multiple barcodes from the same sample run on the same lane. This option typically takes more memory but less time, and potentially affects the number of duplicates marked. With process.one.bamfile.only=false, the mark duplicates module processes input in the following phases:

1. Scan all BAM files.
2. Only after all scanning is complete, then mark duplicates for all BAM files concurrently.

With process.one.bamfile.only=true, mark duplicates typically requires less memory but more process time. With this setting, the mark duplicates module phases are the following:

1. Scan one BAM file.
2. Only after the scan of the first BAM is complete, concurrently do these:
   a. Scan the other BAM files.
   b. Mark duplicates for only the first BAM file.
3. Only after mark duplicates for the first BAM file is complete, the module marks duplicates for the remaining BAM files, one at a time.

BAM file generation

This section describes the phases involved in BAM file generation.
Ma2BAM, pa2BAM

Ma2BAM and pa2BAM are format converters that change match files to a pro-BAM format. These pro-BAM files do not contain sequence or quality information but contain relevant BAM meta data. The pro-BAM files are sorted by incoming bead-id order. Relevant meta data from XSQ files is captured in pro-BAM files at this stage. The parameters bamgen.mqv.threshold and create.unmapped.bam.files control the functionality of the ma2BAM and pa2BAM utilities. The pro-BAM files are generated on the scratch folder.

Ma2BAM takes output of mapping (MA files) and converts them to pro-BAM format. The pa2BAM takes output of pairing (PA files) and converts them to pro-BAM format. During BAM generation, the mapping and pairing quality values are also computed.

For reads with multiple ungapped alignments, the read with the highest mapping or pairing quality value is chosen as the primary alignment for the read, and is reported to the BAM file. In cases where there are multiple alignments with the same quality value, the primary alignment is chosen at random from among the alignments with the same quality value.

Mapping and pairing quality values

This section describes mapping quality value (MQV) and pairing quality value (PQV) scores. Quality values are Phred-scaled quality scores. For any given alignment, an MQV is a measure of the confidence of a read aligning to the particular location, given all possible alignments for the read. A PQV represents the confidence in both alignments of a pair, and is a combined quality score for both reads. Quality values are within the range 0–100:

- **0** – The highest probability of error
- **100** – The lowest probability of error

The following factors increase confidence and increase an alignment’s quality value:

- Longer alignment
- Fewer possible alignments
- Fewer mismatches within the alignment

These factors reduce confidence and reduce an alignment’s quality value:

- Shorter alignment
- More possible alignments
- More mismatches within the alignment

The pairing algorithm reports multiple sets of possible alignments for any given pair of reads (F3/R3 tags for a mate-pair run and F3/F5-P2 tags for a paired-end run). The pairing quality algorithm uses a Bayesian approach to calculate the quality of a given alignment for a pair of reads and the alignment with the highest pairing quality value (PQV) is chosen as the primary alignment for the pair of reads. The PQVs represent the Phred-scaled quality score, and are useful for downstream variant detection modules such as diBayes, small indels, large indels, and CNV.

In order to be consistent with the Phred quality score (-10*log10[prob(error)]) used widely in literature, the quality is computed as the negative log odds of misaligning the read (or pairs of reads, for PQV). The resulting quality values are normalized by the maximum possible value to ensure that the quality values are within the range 0–100.
Gapped alignments

The pairing algorithm searches for gapped alignments (indels) when one of the tags (F3/R3/F5-P2/F5-BC) maps to the reference genome and the other tag does not map to the genome within the insert-size range, or it maps within the insert-range but the alignment is short. If both an ungapped and a gapped alignment are found for a given read, then, due to the low prior probability of $10^{-4}$ assigned to the gapped alignments, the PQV for gapped alignments is zero.

Figure 5 Example of a gapped alignment and a partial alignment

In calculating the PQV for gapped alignments, the alternative hypothesis tested is the probability of finding the partial unaligned alignments. The read with the gapped alignment is treated as two partial reads on either side of the indel start point. The partial read with the greater length is used as the partial alignment length for the alternate hypothesis.

Refcor

Reference Assisted Color-to-Base Translation (also known as refcor) combines information from color calls, optional ECC extra primer round calls, and the reference sequence, to enhance the quality of base-space reads by detecting and fixing color errors that convert into multiple base-space errors.

Refcor accepts a set of pro-BAM files, which contain alignments information without base or color sequence, and generates a corresponding set of BAM files containing alignments with base-space and color-space calls (on the scratch folder). These BAM files are sorted by bead-ID and contain dummy base calls and QVs, but are otherwise compatible with the BAM specification. The refcor module produces bead-ID-sorted BAM files that have valid calls and QVs.

For each alignment, the module constructs the primer-transition graph with vertices corresponding to all possible $k$-mers ($k=4$ for 2BE+4BE, $k=2$ for 2BE) in each read position. Two adjacent vertices are connected with an edge if they overlap in $k-1$ bases. A score corresponding to the $(1-Pe)$, where $Pe$ is the probability of being erroneous, is assigned to each of the vertices. Scores are derived from quality values corresponding to enumerated calls. When making a base call in a position, the current algorithm considers the cumulative probability of all sequences that have called base in that position. The evidence from the reference (base) is assigned to the vertex with $k$-mer ending in corresponding base only if the color transition between current and following reference bases matches the color call in a read. Such assignment reduces the reference over-correction in SNP positions. In order to reduce reference over-correction in positions with color errors, we assign to reference base a very low weight corresponding to QV of 8. The weight can be reduced to QV of 0, completely reducing
the reference guiding in the color-to-base translation. This results in a smaller skew between reference to non-reference allele ratio. Increasing the weight results in false positive variant calls reduction. This process restores the phase shift caused by color errors or completely eliminates the errors.

The parameters \texttt{bamgen.refcor.softclip}, \texttt{bamgen.refcor.addcs}, \texttt{refcor.reference.weight}, and \texttt{refcor.base.filter.qv} control the \texttt{refcor} program. The defaults provided are validated for a wide range of data.

- If the XSQ file contains base-space data only, the XSQ base call and quality value are transferred to the BAM file. In this case, no color calls are present in the BAM file, irrespective of the value of \texttt{bamgen.refcor.addcs}.
- If the XSQ file contains 2BE color-space data only, \texttt{refcor} uses aligned 2BE and reference to generate base translation for the aligned part of the read. The base sequence and quality value are generated using a proprietary algorithm. For reads with no alignment, bases are determined by naively translating color to base. For these unaligned reads, the QV for the base at position \( k \) is the minimum color QV for all positions \( \leq k \).
- If the XSQ file contains ECC data, then reference with 2BE and 4BE color are used to generate the new base sequence. The base sequence and quality value are generated using a proprietary algorithm. For reads with no alignment, the original base-space data from ECC is preserved.

Sort BAM

The output of \texttt{refcor} is a set of BAM files in bead-id sorted order. These files are sorted in this step using Picard sort into coordinate sort order before merge. These BAM files are referred to as mini-BAM files and are stored in the analysis temporary directory.

Merge BAM

This phase accepts a set of mini-BAM files and outputs a set of BAM files. These BAM files are stored in the analysis output directory.

One BAM file is generated per read-set. For a 96-barcoded dataset in a single XSQ file, either 96 or 97 BAM files are created. The 97th BAM contains reads that have the default barcode index.

Quality alignments and unique alignments

We strongly recommend using mapping quality values, rather than uniqueness, as the standard for recognizing or filtering good alignments. A mapping quality value estimates the probability that the alignment is correct, and takes several indications into account, including the number of alignments or uniqueness of an alignment.

The parameter \texttt{bamgen.mqv.threshold}, (introduced in release 2.5.1), is available to filter alignments written to your BAM output files. In order to limit your BAM files to good quality reads, set \texttt{bamgen.mqv.threshold} to a positive integer such as 8 or higher. This technique ensures that you do not see poor quality reads in your BAM file, and that your tertiary analysis is limited to good quality reads.

How a quality value is derived

The following factors increase confidence and increase an alignment’s quality value:
• Longer alignment
• Fewer possible alignments
• Fewer mismatches within the alignment

These factors reduce confidence and reduce an alignment’s quality value:
• Shorter alignment
• More possible alignments
• More mismatches within the alignment

How a quality value is interpreted

For any given alignment, a quality value is a measure of the confidence of a read aligning to the particular location, given all possible alignments for the read. Quality values are within the range 0–100:
• 0 – The highest probability of error
• 100 – The lowest probability of error

Quality values are Phred-scaled quality scores, logarithmically linked to error probabilities. A mapping QV of 10 means that we calculate there is a 90% chance that the alignment is correct. A mapping QV of 20 means that we calculate there is a 99% chance that the alignment is correct.

Unique alignments

Unique alignments might be confused with high quality alignments, but uniqueness is not by itself a good criteria to judge an alignment. Uniqueness only means that one step of secondary mapping, the aligner, reports one alignment. This information is a part of the total information generated during mapping, but does not contain all of the information known about the alignment. In contrast, the mapping quality value is a numerical representation of all considerations about the alignment.

For example, a read might have multiple alignments. If the quality value for only one of those multiple alignments is high, that probability of that alignment being good is high. That alignment should be used in your analysis, even though it is not unique.

For a unique alignment, its quality value must be examined to determine if the alignment should be taken or filtered out. Its uniqueness by itself is irrelevant to the decision about whether it is a good read.

Fragment mapping parameters

This section describes the runtime parameters which control the behavior of your fragment mapping analysis. Table 26 lists these parameters.

In order to change a parameter value, use the set param shell command to replace the line [optionally change mapping defaults after this line] in the example commands in “Examples of how to run a mapping analysis” on page 130. For instance, to have the unmapped BAM file generated, change the parameter create.unmapped.bam.files with this command:

```bash
set create.unmapped.bam.files 1 secondary/fragment.mapping.ini
```
**Fragment mapping parameters table**

Table 26 lists the fragment mapping parameters. See Table 30 on page 160 for mapping statistics (BAMStats) parameters.

### Table 26  Mapping parameter description

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Optional resource parameters</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wall.time</td>
<td>120</td>
<td>Total time for the process to complete.</td>
</tr>
<tr>
<td>java.heap.space</td>
<td>1500</td>
<td>Dynamic memory requirement, in megabytes.</td>
</tr>
<tr>
<td>mapping.memory</td>
<td>15gb</td>
<td>The total memory, in gigabytes, that is available for map reads, per compute node. Include the units gb in the setting. This number is set to 15 GB because the minimum hardware requirement for memory is 16 GB.</td>
</tr>
<tr>
<td><strong>Note:</strong> Whenever possible, run mapping with 24 GB. When a mapping analysis is run with less than 19 GB of memory, the mapping module splits the reference. During gapped alignment of the second half of the genome, some alignments are reported at incorrect positions. The incorrect gapped alignments cause incorrect results (missing indels) during small indel analysis.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Note:</strong> Typically, the job scheduler will allocate on nodes that have the requested memory available. For mixed hardware, it is suggested that different queues are created based on the memory available. Mapping jobs should be launched on high memory systems. For human references (hg18, hg19) and a 25.2 scheme, 45 GB is recommended for fastest performance. The smallest recommended RAM for human reference is 19 GB.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Optional parallelization parameters</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fragmap.number.of.nodes</td>
<td>4</td>
<td>The number of compute nodes available for this analysis.</td>
</tr>
<tr>
<td>fragmap.max.number.of.jobs</td>
<td>100</td>
<td>The maximum number of jobs allowed for mapping. Allowed values: Integers 24–1000.</td>
</tr>
<tr>
<td>mapping.np.per.node</td>
<td>8</td>
<td>The number of processors per node use for mapping. The reads are divided into the number of chunks specified for this parameter.</td>
</tr>
<tr>
<td>fragmap.minreads.per.node</td>
<td>4000000</td>
<td>If the total number of fragments exceeds this amount, then the analysis is distributed across the available nodes. Allowed values: Integers 1–125000000 (125 Million)</td>
</tr>
<tr>
<td>fragmap.maxreads.per.node</td>
<td>150000000</td>
<td>The maximum number of fragments that can be processed on a single node. Allowed values: Integers 1–150000000 (150 Million)</td>
</tr>
<tr>
<td><strong>Optional mapping parameters</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Fragment mapping parameters

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
</table>
| mapping.in.base              | false         | Whether to map in base space (true) or color space (false). Set to true if input data has base space available. Allowed values:  
  - false: Map in color space.  
  - true: Map in base space.  
  If only color space is available, then the module fails when base space mapping is turned on. |
| second.map.gapped.algorithm  | GLOBAL        | Whether or not to do indel finding, and controls the behavior of indel finding. Allowed values:  
  - GLOBAL: Mapping reports global alignment up to one indel.  
  - LOCAL: Mapping reports local alignment up to one indel. |
| Optional BAM generation parameters |  |  |
| bamgen.refcor.softclip       | 0             | For cases when the unaligned portion of a read needs to be presented in the BAM record, we use soft clipping in the CIGAR string. If a base-space sequence generated by ECC decoder is available, we use that sequence to represent corresponding part of the read. If the ECC decoded base-space is not available, then we naively translate using 2BE and/or +4BE evidence (if available). The soft clipped part of the read can be permanently clipped if either of these conditions is true:  
  - The ECC base QV is less than 4.  
  - The ECC base and 2+4 naive propagation have less than 50% similarity. Allowed values: 0,1. |
| bamgen.refcor.addcs          | 1             | Whether or not to add the color sequence to BAM records. If the XSQ file does not contain color space, this parameter is ignored, and the resulting BAM file output does not contain color space. See "Refcor" on page 139 for information about refcor. Allowed values: 0,1. |
| refcor.reference.weight      | 8             | Reference weight. This parameter is used during base translation. In the read reconstruction process, multiple signals are combined to generate the final base call. This parameter adds weight (in terms of Phred score) only to signals which are compatible with reference. Color combinations that result in a variant are considered compatible with reference. Additional weight helps to eliminate base propagation errors caused by color error(s) during base translation. Allowed values: Integers 0–100. |
| refcor.base.filter.qv        | 10            | Bases with a quality value below the value of this parameter provided are replaced with 'N'. Allowed values: Integers 0–1000. |
| bamgen.mqv.threshold         | 0             | Provides control over the contents written to the output BAM file depending on the quality value of the alignment. To preserve only high quality alignments, set this value to a positive integer. Allowed values: Integers 0–1000. |
Table 26  Mapping parameter description (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>create.unmapped.bam.files</td>
<td>FALSE</td>
<td>If set to TRUE, creates an additional BAM output file containing unmapped reads. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• FALSE: Do not create the unmapped reads output file.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• TRUE: Create a BAM output file containing the unmapped reads.</td>
</tr>
</tbody>
</table>

Mapping performance

Mapping speed depends on hardware properties as well as on the scattering logic. Mapping runs are split into multiple jobs for processing efficiency. The following parameters affect how a mapping run is split:

- **fragmap.number.of.nodes** – Specifies the number of jobs that are created, only if a split is necessary. The size of each split job is approximately the total number of reads divided by the number of jobs.
- **fragmap.max.number.of.jobs** – Determines the maximum number of jobs that can be launched per analysis. Setting this parameter to a very large value can cause the scheduler to behave incorrectly.
- **fragmap.minreads.per.node** – Determines the threshold of beads beyond which splitting occurs.
- **fragmap.maxreads.per.node** – Determines the largest number of beads that can be mapped per node. The default is based on the minimum scratch space requirements. If you scratch space is smaller or larger, this number can be made smaller or larger respectively.
- **fragmap.processors.per.node** – This is the number of cores available for mapping analysis on each node.
- **mapping.memory** – For human mapping and 25.2.0 scheme, there are 3 schema lines. Simultaneous handling of schema lines reduces I/O and therefore improves mapping speed.

Memory requirements for mapping jobs are the following:

- Reference: ~6 GB
- 1 schema line: ~13 GB
- 1 schema line and reference: ~19 GB
- 2 schema lines and reference: ~32 GB
- 3 schema lines and reference: ~45 GB

Below 19 GB, the reference is split up and I/O increases significantly. Pre-built hash tables are not used for memory less than 24 GB. LifeScope™ Software supports the reuse of hash tables for human genome reference files (hg18 and hg19) and the default mapping schemes 35.2, 25.2, and 20.1. Pre-calculated color-space hash tables are included in the reference directory available with LifeScope™ Software. If system RAM is 24 GB or higher, the hash tables are loaded into memory. Depending on the pipeline, this hash table reuse typically saves 2–3 hours in analysis processing time.

For genomes without pre-calculated hash tables in the reference repository, the mapping module creates hash tables but does not save them for reuse.

In general, set the mapping.memory parameter to be 1 GB less than the total system memory (RAM) available.
Mapping only scales when the number of reads that are mapped together is greater than 100 M. Splitting such that the total number of fragments per node is significantly less than 100 M is not recommended. The splitting can be explained using the following pseudo-code.

```plaintext
if (num.reads <= fragmap.minreads.per.node) { num.jobs = 1}
reads.per.job = num.reads / fragmap.number.of.nodes
if (reads.per.job > fragmap.maxreads.per.node) {
    reads.per.job = fragmap.maxreads.per.node
}
num.jobs = ceiling (num.reads / reads.per.job)
if (num.jobs > fragmap.max.number.of.jobs) {
    throw exception;
}
```

In above pseudo-code, the input read-sets (possibly from multiple XSQ files) have num.reads that fall in one read length category. Num.jobs is the total number of jobs launched for mapping analysis of num.reads beads.

In general, the mapping parameters allow mixing and matching single-anchor and multiple-anchor schemes, and also mixing and matching different seed lengths and different numbers of mismatches. For example, an unmapped scheme of 25.2.0, 35.2.0:15:25, 20.1.0, with a repetitive scheme of 35.2.0:30, 45.2.0, 60.2.0, is supported, but not recommended. The impact on performance can be significant.

### Mapping algorithm

#### Internal mapping parameters

Table 27 lists mapping parameters that we do not recommend changing. The “Mapping schemes” section which follows describes how these parameters affect the behavior of the mapping module.

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
</table>
| fragment.mapping.run       | 1             | Indicates whether or not to run the mapping module. Allowed values:
|                            |               | • 0: Do not run the mapping module.
|                            |               | • 1: Run the mapping module. The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use. |
| mapping.use.iub.reference  | FALSE         | Whether or not to support reference sequences that contain IUB codes. Allowed values:
|                            |               | • FALSE: Do not support IUB codes in the reference file.
|                            |               | • TRUE: Mapping allows matching to either alleles of bi-allelic IUB codes in the reference file. |
### Mapping algorithm

**Table 27** Internal mapping parameters (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
</table>
| bamgen.primary.output.filter.type       | primary_only  | Filters the content written to the output BAM file depending on whether the alignment is gapped or not. Note that unmapped reads are considered primary. Allowed values:  
• no_filtering: Report all alignments.  
• primary_only: Report all primary alignments (both gapped and ungapped).  
• primary_ungapped_all_gapped: Report all gapped alignments (even if not primary) and report ungapped alignments only if they are primary. Default. |
| fragmap.num.mapping.stages              | 2             | The number of mapping passes to perform in this analysis. Allowed values:  
• 1: Perform mapping in one pass. Gapped alignment finding is off by default.  
• 2: Perform mapping with two passes. Recommended for increased accuracy of indel finding, but with an increase in processing time. The second stage of mapping performs only gap alignments. When set to 2, do not set second.map.gapped.algorithm to NONE. |
| mapping.fragment.tag                    | F3            | When mapping XSQ files with more than one tag, this parameter selects the tag to map. The value must match one of the tag names in the TagDetails group of the XSQ file. Allowed values: F3, R3, F5-P2, F5-BC, F5-DNA, F5-RNA. This feature should not be used unless for data quality analysis. If a LMP/PE XSQ file is processed for fragment mapping, tertiary analysis is not supported. |
| mapping.schema.file                     | —             | Specifies the full path to the file containing the mapping schema. If this parameter is not set, the default installed file is used. |
| mapping.gap.min.non.matched.length     | 9             | The minimum number of non-matches allowed during indel finding (if performed). |
| min.length.for.aggressive.gapped.mapping | 50           | Overrides some of other parameters’ settings, depending on the relation between this value and read length. If the read length of tag1 is less than this value, the following parameters are set:  
• num.mapping.stages.tag1=1  
• first.map.gapped.algorithm.tag1=NONE  
• second.map.gapped.algorithm.tag1=NONE  
• pairing.gap.max.mismatches.tag1=2  
If the read length of tag2 is less than this value, the following parameters are set:  
• num.mapping.stages.tag2=1  
• first.map.gapped.algorithm.tag2=NONE  
• second.map.gapped.algorithm.tag2=NONE  
• pairing.gap.max.mismatches.tag2=2 |

**First map parameters**
## Table 27 Internal mapping parameters (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>first.map.gapped.algorithm</td>
<td>NONE</td>
<td>Whether or not to do indel finding, and controls the behavior of indel finding. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>NONE</strong>: Turns off indel finding.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>GLOBAL</strong>: Mapping reports global gapped alignments up to one indel.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>LOCAL</strong>: Mapping reports local gapped alignments up to one indel.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>first.map.gapped.algorithm is set to <strong>NONE</strong> if the read length is less than <strong>min.length.for.aggressive.gapped.mapping</strong>.</td>
</tr>
<tr>
<td>first.map.gapped.alignments.only</td>
<td>0</td>
<td>Whether or not the first pass outputs only gapped alignments. <strong>Allowed values</strong>: Integers 0 or 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>0</strong>: Do not restrict the output to only gapped alignments.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>1</strong>: Restrict the output to only gapped alignments.</td>
</tr>
<tr>
<td>first.map.gapped.deletion</td>
<td>19</td>
<td>The maximum deletion size in a gapped alignment in the first pass. <strong>Allowed range</strong>: 0 &lt;= deletion size &lt;= read length.</td>
</tr>
<tr>
<td>first.map.gapped.insertion</td>
<td>4</td>
<td>The maximum insertion size in a gapped alignment in the first pass. <strong>Allowed range</strong>: 0 &lt;= insertion size &lt;= read length.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The module fails if the insertion size is not within this range.</td>
</tr>
<tr>
<td>first.map.gapped.penalty</td>
<td>10</td>
<td>The penalty for a single gap in alignment, for the first mapping pass. <strong>Allowed values</strong>: Integers &gt;= 0.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When the penalty is 10, either side of the gap must have a score of at least 10 for the gapped alignment to exist. Gapped alignment is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>effectively turned off if the penalty is set greater than the read length.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For a gapped alignment, scoring is determined by three factors:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The matching score, which is 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The mismatch score, which is set by first.map.mismatch.penalty and is -2, by default.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• This parameter (first.map.gapped.penalty), which sets the score associated with a gap.</td>
</tr>
<tr>
<td>first.map.gapped.error.indel</td>
<td>3</td>
<td>The number of mismatches allowed for gap alignments on the first mapping pass. <strong>Reasonable range</strong>: 1 &lt;= value &lt;= 15% of the read length.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Allowed values</strong>: Integers, from 1 to the read length. <strong>Only used if first.map.gapped.algorithm is set to GLOBAL.</strong></td>
</tr>
<tr>
<td>first.map.gapped.edge.length.insertions</td>
<td>12</td>
<td>The minimum edge length required for insertions, on the first pass. <strong>Reasonable range</strong>: 1–15% of the read length.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lower values may lead to more false positives. <strong>Allowed values</strong>: Integers, from 1 to the read length.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Only used if first.map.gapped.algorithm is set to GLOBAL.</strong></td>
</tr>
<tr>
<td>first.map.gapped.edge.length.deletions</td>
<td>12</td>
<td>The minimum edge length required for deletions, on the first pass. <strong>Reasonable range</strong>: 1–15.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lower values may lead to more false positives. <strong>Only used if first.map.gapped.algorithm is set to GLOBAL.</strong></td>
</tr>
</tbody>
</table>
### Table 27 Internal mapping parameters (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>first.map.gapped.seed.window. left</td>
<td>40</td>
<td>The window size allowed to the left of the anchor alignment, on the first pass. Reasonable range: 5–50. Only used if first.map.gapped.algorithm is set to GLOBAL.</td>
</tr>
<tr>
<td>first.map.gapped.seed.window. right</td>
<td>80</td>
<td>The window size allowed to the right of the anchor alignment, on the first pass. Reasonable range: (5 + \text{read length}) - (50 + \text{read length}). Only used if first.map.gapped.algorithm is set to GLOBAL.</td>
</tr>
<tr>
<td>first.map.mask.positions.</td>
<td>—</td>
<td>A string of 0s and 1s that indicate the read positions to be masked during the first mapping pass. There is no penalty in mapping towards positions that are masked. Default: blank</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Leaving the parameter blank results in no masking. Leave the parameter blank for global style mapping [previously called “classic” mapping in earlier versions]. [Masking replaces read trimming.]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If set, the length of the string must match the anchor length or seed length (the first value of scheme parameter such as 25.2.0).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>With a 25-mer read and a 25.2.0 scheme, an example mask is 11111111111111111100000 (20 1s, then 5 0s). The bases corresponding to the 0s are ignored. Under the 0s portion of the mask, any mismatches do not count toward the mismatch penalty.</td>
</tr>
<tr>
<td>first.map.max.hits</td>
<td>50</td>
<td>The maximum number of alignments for a read on the first pass. This setting gives the maximum number of hits that are reported in the mapping output (even if more hits are found). Allowed values: Integers &gt;= 1.</td>
</tr>
<tr>
<td>first.map.mismatch.penalty</td>
<td>-2</td>
<td>A scoring penalty for mismatches. Used in local alignment mode. Must be a negative number or zero. Allowed values: Integers &lt;= 0. If this penalty is set to a value greater than the read length, effectively no mismatches are allowed on the first pass.</td>
</tr>
<tr>
<td>first.map.scheme.unmapped.1</td>
<td>0.0.0</td>
<td>The scheme used for unmapped reads of a length less than 25, for the first mapping stage. See “Mapping schemes” on page 151.</td>
</tr>
<tr>
<td>first.map.scheme.repetitive.1</td>
<td>—</td>
<td>The scheme used for repetitive reads of a length less than 25, for the first mapping stage. See “Mapping schemes” on page 151.</td>
</tr>
<tr>
<td>first.map.scheme.unmapped.25</td>
<td>25.2.0</td>
<td>The scheme used for unmapped reads of a length 25–34, for the first mapping stage. See “Mapping schemes” on page 151.</td>
</tr>
<tr>
<td>first.map.scheme.repetitive.25</td>
<td>—</td>
<td>The scheme used for repetitive reads with a length 25–34, for the first mapping stage. See “Mapping schemes” on page 151.</td>
</tr>
<tr>
<td>first.map.scheme.unmapped.35</td>
<td>25.2.0</td>
<td>The scheme used for unmapped reads with a length 35–49, for the first mapping stage. See “Mapping schemes” on page 151.</td>
</tr>
<tr>
<td>first.map.scheme.repetitive.35</td>
<td>—</td>
<td>The scheme used for repetitive reads with a length 35–49, for the first mapping stage. See “Mapping schemes” on page 151.</td>
</tr>
<tr>
<td>first.map.scheme.unmapped.50</td>
<td>25.2.0:15</td>
<td>The scheme used for unmapped reads with a length 50–59, for the first mapping stage. See “Mapping schemes” on page 151.</td>
</tr>
</tbody>
</table>
### Table 27  Internal mapping parameters (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>first.map.scheme.repetitive.50</td>
<td>—</td>
<td>The scheme used for repetitive reads with a length 50–59, for the first mapping stage. See “Mapping schemes” on page 151.</td>
</tr>
<tr>
<td>first.map.scheme.unmapped.60</td>
<td>25.2.0:20</td>
<td>The scheme used for unmapped reads with a length 60–74, for the first mapping stage. See “Mapping schemes” on page 151.</td>
</tr>
<tr>
<td>first.map.scheme.repetitive.60</td>
<td>—</td>
<td>The scheme used for repetitive reads with a length 60–74, for the first mapping stage. See “Mapping schemes” on page 151.</td>
</tr>
<tr>
<td>first.map.scheme.unmapped.75</td>
<td>25.2.0:20</td>
<td>The scheme used for unmapped reads with a length of 75 or larger, for the first mapping stage. See “Mapping schemes” on page 151.</td>
</tr>
<tr>
<td>first.map.scheme.repetitive.75</td>
<td>—</td>
<td>The scheme used for repetitive reads with a length of 75 or larger, for the first mapping stage. See “Mapping schemes” on page 151.</td>
</tr>
<tr>
<td>first.map.scheme.unmapped. variable</td>
<td>25.2.0:20</td>
<td>Use for unmapped reads of variable length coming from ECC trimmed base-space reads, for the first mapping stage. See “Mapping schemes” on page 151.</td>
</tr>
<tr>
<td>first.map.scheme.repetitive. variable</td>
<td>—</td>
<td>Use for repetitive reads of variable length coming from ECC trimmed base-space reads, for the first mapping stage. See “Mapping schemes” on page 151.</td>
</tr>
</tbody>
</table>

### Second map parameters

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>second.map.gapped.alignments. only</td>
<td>1</td>
<td>Whether or not the second pass outputs only gapped alignments. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Do not restrict the output to only gapped alignments.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Restrict the output to only gapped alignments.</td>
</tr>
<tr>
<td>second.map.gapped.deletion</td>
<td>19</td>
<td>The maximum deletion size in a gapped alignment in the second mapping pass.</td>
</tr>
<tr>
<td>second.map.gapped.insertion</td>
<td>4</td>
<td>The maximum insertion size in a gapped alignment in the second mapping pass. Allowed range: 0 &lt;= insertion size &lt;= read length.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The module fails if the insertion size is not within this range.</td>
</tr>
<tr>
<td>second.map.gapped.penalty</td>
<td>10</td>
<td>The penalty for a single gap in alignment, for the second mapping pass. Allowed values: Integers &gt;= 0.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When the penalty is 10, either side of the gap must have a score of at least 10 for the gapped alignment to exist. Gapped alignment is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>effectively turned off if the penalty is set greater than the read length.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For a gapped alignment, scoring is determined by three factors:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The matching score, which is 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The mismatch score, which is set by second.map.mismatch.penalty and is –2, by default.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• This parameter (second.map.gapped.penalty), which sets the score associated with a gap.</td>
</tr>
<tr>
<td>second.map.gapped.error.indel</td>
<td>3</td>
<td>The number of mismatches allowed for gap alignments on the second mapping pass. Reasonable range: 1 &lt;= value &lt;= 15% of the read length.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Maximum value: the read length.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Only used if second.map.gapped.algorithm is set to GLOBAL.</td>
</tr>
</tbody>
</table>
## Table 27 Internal mapping parameters (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>second.map.gapped.edge.length. insertions</td>
<td>12</td>
<td>The minimum edge length required for insertions, on the second mapping pass. Reasonable range: 1–15% of the read length. Lower values may lead to more false positives. Only used if second.map.gapped.algorithm is set to GLOBAL.</td>
</tr>
<tr>
<td>second.map.gapped.edge.length. deletions</td>
<td>12</td>
<td>The minimum edge length required for deletions, on the second mapping pass. Reasonable range: 1–15. Lower values may lead to more false positives. Only used if second.map.gapped.algorithm is set to GLOBAL.</td>
</tr>
<tr>
<td>second.map.gapped.seed.window. left</td>
<td>40</td>
<td>The window size allowed to the left of the anchor alignment, on the second pass. Reasonable range: 5–50. Only used if second.map.gapped.algorithm is set to GLOBAL.</td>
</tr>
<tr>
<td>second.map.gapped.seed.window. right</td>
<td>80</td>
<td>The window size allowed to the right of the anchor alignment, on the second pass. Reasonable range: (5 + read length) - (50 + read length). Only used if second.map.gapped.algorithm is set to GLOBAL.</td>
</tr>
<tr>
<td>second.map.masked.positions</td>
<td>—</td>
<td>A string of 0s and 1s that indicate the read positions to be masked during the second mapping pass. There is no penalty in mapping towards positions that are masked. Default: blank</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Leaving the parameter blank results in no masking. Leave the parameter blank for global style mapping (previously called “classic” mapping in earlier versions). (Masking replaces read trimming.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If set, the length of the string must match the anchor length or seed length (the first value of scheme parameter such as 25.2.0).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>With a 25-mer read and a 25.2.0 scheme, an example mask is 1111111111111111111100000 (20 1s, then 5 0s). The bases corresponding to the 0s are ignored. Under the 0s portion of the mask, any mismatches do not count toward the mismatch penalty.</td>
</tr>
<tr>
<td>second.map.max.hits.tag1</td>
<td>50</td>
<td>The maximum number of alignments for a read on the second mapping pass. This setting gives the maximum number of hits that are reported in the mapping output (even if more hits are found). Allowed values: Integers &gt;= 1. The module fails if the value is not within this range.</td>
</tr>
<tr>
<td>second.map.max.hits.tag2</td>
<td>50</td>
<td>The maximum number of alignments for a read on the second mapping pass. This setting gives the maximum number of hits that are reported in the mapping output (even if more hits are found). Allowed values: Integers &gt;= 1. The module fails if the value is not within this range.</td>
</tr>
<tr>
<td>second.map.mismatch.penalty</td>
<td>-2</td>
<td>A scoring penalty for mismatches. Used in local alignment mode. Must be a negative number or zero. Allowed values: Integers &lt;= 0. If the absolute of the penalty is set to greater than the read length, effectively no mismatches are allowed on this pass.</td>
</tr>
<tr>
<td>second.map.scheme.unmapped.1</td>
<td>0.0.0</td>
<td>The scheme used for unmapped reads of a length 1–24, for the second mapping stage. See “Mapping schemes” on page 151.</td>
</tr>
</tbody>
</table>
Mapping schemes

The mapping scheme parameters, listed in Table 27, determine the behavior of the mapping module. The LifeScope™ Software mapping module supports two types of mapping, global mapping and seed-and-extend mapping. These two approaches are described in more detail below.

- **Global mapping** – With this approach the seed length matches the full read length, and the seed anchors at the beginning of the read. For example, global mapping for 50-bp reads uses a scheme such as 50.6.0. (In this example, 50, the seed length, matches the read length, and 0, the seed start position, starts at the beginning of the read.)

- **Seed-and-extend** – The initial alignment step that locates short matches between a read and the reference sequence. The seed specifies the length of the attempted match. This approach is also called local alignment.
Global mapping

To map a 50 bp read using global mapping, a typical scheme to use is 50.6.0. This allows for up to 6 mismatches. Because the seed length equals the length of the read, there is no extension.

Seed-and-extend

With the seed-and-extend approach, a mapping scheme parameter value such as 25.2.0 is interpreted as follows:

- **First value** – The seed length (25).
- **Second value** – The quantity of allowed mismatches in the seed (2).
- **Third value** – The start site of the seed within the read (0).

Figure 6 Seeds for local alignments

For example, with a 25.2.0 seed extend scheme, the mapping module examines the first 25 bases of a 50 bp read, and if that portion aligns to the reference with two or fewer mismatches, then the mapping module attempts seed extension. For any alignment, we try two different extensions and report the one with higher score. The two extension approaches are described below:

- **Ungapped extension** – Extends from the anchor alignment, to find the best ungapped local alignment with a score S. The alignment length in this case can be shorter than the read length.

- **Gapped extension** – Looks for one indel, following two different algorithms that are controlled by the parameters `first.map.gapped.algorithm` and `second.map.gapped.algorithm`. Only one of the algorithms can be used at a time. These algorithms are:
  - **Local indel finding extension** – Extends from the anchor alignment, doing gapped extension with a modified Smith-Waterman-like algorithm. In this algorithm, we use a matching score (same as in ungapped extension) and gap penalty (`*.map.gapped.penalty`), to find a best local alignment within user specified size restriction (`*.map.gapped.insertion/deletion`). If this alignment has a score larger than S (the best ungapped alignment extended from the same anchor alignment), the alignment is reported. The alignment length in this case can be shorter than the read length.
  - **Global indel finding extension** – Extends from anchor alignment, does a full-length gapped extension with the allowed number of mismatches (`*.map.gapped.error.indel`), and within user-specified size restrictions (`*.map.gapped.insertion/deletion`), window restrictions (`*.map.gapped.seed.window.*`), and edge length restrictions.
("*.map.gapped.edge.length.insertions/deletions"). If this alignment has a score larger than \( S \), it is reported. This algorithm is the same as what was used with the frag indel module in earlier versions of the software.

For information on sliding periodic seeds, refer to the publication at this site:

http://bioinformatics.oxfordjournals.org/content/25/19/2514.full

Unmapped and repetitive schemes

Mapping schemes have two categories, a scheme for unmapped reads (*map.scheme.unmapped.*) and a scheme for repetitive reads (*map.scheme.repetitive.*). Either type of scheme can also be either single-anchor or multi-anchor.

A single-anchor scheme is represented by a comma-separated list of values, where each value is in the format seed_length.mismatches.seed_start_position (see Figure 6 on page 152). The following apply to both unmapped and repetitive single-anchor schemes:

- **Scheme 25.2.0,25.2.20** – First maps using scheme 25.2.0 and only for those reads that do not have an alignment, maps a second round using the scheme 25.2.20.
- **Scheme 25.2.0,25.2.20,25.2.30** – First maps using scheme 25.2.0 and only for those reads that do not have an alignment, maps a second round using the scheme 25.2.20. The reads that do not map to 25.2.20 either are then tried with scheme 25.2.30.
- When both repetitive and unmapped are specified, the repetitive schemes are applied only to the repetitive reads from first unmapped scheme. Figure 7 depicts the logic followed for an unmapped scheme of 25.2.0, 25.2.20 and a repetitive scheme of 35.2.0, 35.2.30.

Multi-anchor schemes are represented by multiple colon-separated seed start positions, in the format seed_length.mismatches.seed_start_pos1:seed_start_pos2. The following apply to both unmapped and repetitive multi-anchor schemes:

- **Scheme 25.2.0:20** – Simultaneously maps using scheme 25.2.0 and 25.2.20, and reports all alignments.
- **Scheme 25.2.0:20:30** – Simultaneously maps scheme 25.2.0, 25.2.20 and 25.2.30, and reports all alignments.
- **Figure 8** depicts the logic followed for an unmapped scheme of 25.2.0:20 and a repetitive scheme of 35.2.0:30.

The Figures 7 and 8 describe how the mapping scheme parameters are used in the mapping module. \( su1+su2 \) and \( sr1+sr2 \) are in the form of \( (L,m,b1)+(L,m,b2) \), where \( L \) is the seed length, \( m \) is number of mismatches allowed in the seed, \( b1 \) and \( b2 \) are the seed starting positions, \( H \) is the number of hits, and \( Z \) is the maximum number of alignments reported by the mapping program.

In these figures, \( su \) refers to a mapping scheme for matching unmapped reads. And \( sr \) refers to a mapping scheme for matching repetitive reads.

When \( H < Z \), the read is not in a repetitive region. When \( H >= Z \), the read matches too many locations on the reference, so it is considered a repetitive read, or sequenced from a repetitive region.
Chapter 9  Run a Resequencing Mapping Analysis

Mapping algorithm

Figure 7  Single-anchor mapping for unmapped and repetitive reads

Figure 8  Multi-anchor mapping for unmapped and repetitive reads
# Pairing algorithm

## Pairing parameters

Table 29 lists parameters for the pairing algorithm.

In order to change a parameter value, use the `set param` shell command to replace the line `optionally change mapping defaults after this line` in the example commands in “Examples of how to run a mapping analysis” on page 130. For instance, to adjust the default minimum insert size, change the parameter `default.minimum.insert.size` with this command:

```
set default.minimum.insert.size 200 secondary/pair.mapping.ini
```

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>default.minimum.insert.size</td>
<td>0</td>
<td>The minimum insert size allowed for the insert size range estimate. This value is used if this measurement fails. Allowed values: Integers 0–10000.</td>
</tr>
<tr>
<td>default.maximum.insert.size</td>
<td>20000 for LMP 2000 for PE</td>
<td>The maximum insert size allowed for the insert size range estimate. This value is used if this measurement fails. Allowed values: Integers 0–10000.</td>
</tr>
<tr>
<td>minimum.points.for.size.stat</td>
<td>1000000</td>
<td>Minimum points in insert size distribution points to do compute statistics.</td>
</tr>
<tr>
<td>memory.per.bam.merge</td>
<td>3</td>
<td>Gigabytes of memory for one BAM merge thread.</td>
</tr>
</tbody>
</table>

## Internal pairing parameters

Table 29 lists pairing parameters that we do not recommend changing.

The first stage of pairing does pairing without finding gaps. This stage is followed by 4 stages of pairing that allow gaps. Values in a format such as `0:3:14:14:0` are a list of one value for each indel pass, separated by colons (:).

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pairing.gap.on</td>
<td>1</td>
<td>Enables indel searching in pairing analysis. Allowed values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Do not include indel search.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Include indel search.</td>
</tr>
<tr>
<td>pairing.anchor.max.mismatches</td>
<td>4, if indel searching is off 4:5:5:3:5, with indel searching</td>
<td>Total mismatches for both reads of a pair. If indel searching is on, then this value is specified as a list of one number for each indel pass, separated by colons (:).</td>
</tr>
<tr>
<td>pairing.gap.min.non.matched.length</td>
<td>9</td>
<td>Used to determine whether the pair goes through an indel pass. If the regular pairing finds a good pair of alignments that are both longer than read length minus this value, then the pair does not need an indel finding run. Allowed values: Integers &gt;= 0.</td>
</tr>
</tbody>
</table>
### Table 29  Internal pairing parameters (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pairing.gap.max.mismatches.tag1</td>
<td>5</td>
<td>Number of mismatches allowed in the extension part of the tag 1. (The extension typically is 5% to 10% of the read length.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pairing.gap.max.mismatches.tag1 is set to 2 if the read length is less than the value of the parameter min.length.for.aggressive.gapped.mapping.</td>
</tr>
<tr>
<td>pairing.gap.max.mismatches.tag2</td>
<td>5</td>
<td>Number of mismatches allowed in the extension part of the tag 2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pairing.gap.max.mismatches.tag2 is set to 2 if the read length is less than the value of the parameter min.length.for.aggressive.gapped.mapping.</td>
</tr>
<tr>
<td>pairing.gapped.max.hits</td>
<td>50:10</td>
<td>For ungapped and gapped alignment determination during rescue, this value determines the maximum number of rescues attempted for each read. If a read has more than this number of hits, rescue is not performed on this bead. The first value is used for ungapped rescue. The second value is used for gapped rescue.</td>
</tr>
<tr>
<td>pairing.output.uniqueness.type</td>
<td>−1</td>
<td>Whether to output non-unique hits. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• −1: A clear zone determines uniqueness. 1 or 2 best pairs are output, depending on whether it is unique.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: All good pairs are output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Only unique good pairs are output.</td>
</tr>
<tr>
<td>pairing.annotation.type</td>
<td>0</td>
<td>Whether to output non-AAA pairs. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Do not output non-AAA pairs.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Output non-AAA pairs, only if pairing and rescue do not find good pairs.</td>
</tr>
<tr>
<td>pairing.mismatch.penalty</td>
<td>−2.0</td>
<td>A single penalty applied to alignments, to compare the significance between alignments. Helps evaluate the alignments. Allowed values: Floats &lt;0.</td>
</tr>
<tr>
<td>pair.uniqueness.threshold</td>
<td>10.0</td>
<td>This value defines the clear zone. This value is considered only when pairing.output.uniqueness.type is set to −1. If the score of the best alignment is this threshold more than the score of the second best alignment, the best alignment is considered uniquely placed. A threshold of zero means that this uniqueness is strictly adhered to, regardless of the alignment’s relative scores. Allowed values: Floats &gt;=0.</td>
</tr>
</tbody>
</table>

**General pairing indel parameters**

For these parameters each value is a colon-separated list of five numbers, which are used, in order, for the five pairing passes. The first pass of pairing does pairing without finding gaps. This pass is followed by four passes of pairing that allow gaps. These values are potentially overridden by size-specific indel parameters in other sections of this table.

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pairing.gap.max.deletion.size</td>
<td>0:11:0:0:500</td>
<td>The maximum deletion size allowed during pairing.</td>
</tr>
<tr>
<td>pairing.gap.max.insertion.size</td>
<td>0:4:15:21:0</td>
<td>The maximum insertion size allowed during pairing.</td>
</tr>
</tbody>
</table>
### Table 29: Internal pairing parameters (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pairing.gap.edge.length.deletions</td>
<td>0:3:0:0:20</td>
<td>The gap edge length for deletions during pairing.</td>
</tr>
<tr>
<td>pairing.gap.edge.length.insertion</td>
<td>0:3:14:14:0</td>
<td>The gap edge length for insertions during pairing.</td>
</tr>
</tbody>
</table>

#### Pairing indel parameters for tag lengths 75–35

For pairs of these specific lengths, parameters in this section are used instead of the general pairing indel parameters. This section applies to pairs with F3 lengths of 75 and above, and R3 or F5 lengths of 35–59.

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pairing.gap.max.deletion.size.75-35</td>
<td>0:11:0:0:500</td>
<td>The maximum deletion size allowed for these lengths.</td>
</tr>
<tr>
<td>pairing.gap.max.insertion.size.75-35</td>
<td>0:4:15:21:0</td>
<td>The maximum insertion size allowed for these lengths.</td>
</tr>
<tr>
<td>pairing.gap.edge.length.deletions.75-35</td>
<td>0:3:0:0:20</td>
<td>The gap edge length for deletions for these lengths.</td>
</tr>
<tr>
<td>pairing.gap.edge.length.insertions.75-35</td>
<td>0:3:14:14:0</td>
<td>The gap edge length for insertions for these lengths.</td>
</tr>
</tbody>
</table>

#### Pairing indel parameters for tag lengths 60–60

For pairs of these specific lengths, parameters in this section are used instead of the general pairing indel parameters. This section applies to pairs with both tags with lengths of 60–74.

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pairing.gap.max.deletion.size.60-60</td>
<td>0:11:0:0:500</td>
<td>The maximum deletion size allowed for these lengths.</td>
</tr>
<tr>
<td>pairing.gap.max.insertion.size.60-60</td>
<td>0:4:15:21:0</td>
<td>The maximum insertion size allowed for these lengths.</td>
</tr>
<tr>
<td>pairing.gap.edge.length.deletions.60-60</td>
<td>0:3:0:0:20</td>
<td>The gap edge length for deletions for these lengths.</td>
</tr>
<tr>
<td>pairing.gap.edge.length.insertions.60-60</td>
<td>0:3:14:14:0</td>
<td>The gap edge length for insertions for these lengths.</td>
</tr>
</tbody>
</table>

#### Pairing indel parameters for tag lengths 50–50

For pairs of these specific lengths, parameters in this section are used instead of the general pairing indel parameters. This section applies to pairs with both tags with lengths of 50–59.

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pairing.gap.max.deletion.size.50-50</td>
<td>0:11:0:0:500</td>
<td>The maximum deletion size allowed for these lengths.</td>
</tr>
<tr>
<td>pairing.gap.max.insertion.size.50-50</td>
<td>0:4:15:21:0</td>
<td>The maximum insertion size allowed for these lengths.</td>
</tr>
<tr>
<td>pairing.gap.edge.length.deletions.50-50</td>
<td>0:3:0:0:20</td>
<td>The gap edge length for deletions for these lengths.</td>
</tr>
<tr>
<td>pairing.gap.edge.length.insertions.50-50</td>
<td>0:3:14:14:0</td>
<td>The gap edge length for insertions for these lengths.</td>
</tr>
</tbody>
</table>

#### Pairing indel parameters for tag lengths 35–35

For pairs of these specific lengths, parameters in this section are used instead of the general pairing indel parameters. This section applies to pairs with both tags with lengths of 35–49.

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pairing.gap.max.deletion.size.35-35</td>
<td>0:11:0:0:500</td>
<td>The maximum deletion size allowed for these lengths.</td>
</tr>
<tr>
<td>pairing.gap.max.insertion.size.35-35</td>
<td>0:4:15:21:0</td>
<td>The maximum insertion size allowed for these lengths.</td>
</tr>
<tr>
<td>pairing.gap.edge.length.deletions.35-35</td>
<td>0:3</td>
<td>The gap edge length for deletions for these lengths.</td>
</tr>
<tr>
<td>pairing.gap.edge.length.insertions.35-35</td>
<td>0:3</td>
<td>The gap edge length for insertions for these lengths.</td>
</tr>
</tbody>
</table>

#### Pairing indel parameters for tag lengths 25–25

For pairs of these specific lengths, parameters in this section are used instead of the general pairing indel parameters. This section applies to pairs with both tags with lengths of 25–34.

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pairing.gap.max.deletion.size.25-25</td>
<td>0:11:0:0:500</td>
<td>The maximum deletion size allowed for these lengths.</td>
</tr>
<tr>
<td>pairing.gap.max.insertion.size.25-25</td>
<td>0:4:15:21:0</td>
<td>The maximum insertion size allowed for these lengths.</td>
</tr>
<tr>
<td>pairing.gap.edge.length.deletions.25-25</td>
<td>0:3</td>
<td>The gap edge length for deletions for these lengths.</td>
</tr>
<tr>
<td>pairing.gap.edge.length.insertions.25-25</td>
<td>0:3</td>
<td>The gap edge length for insertions for these lengths.</td>
</tr>
</tbody>
</table>
### Table 29 Internal pairing parameters (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pairing indel parameters for tag lengths 50–35</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pairing.gap.max.deletion.size.50-35</td>
<td>0:11</td>
<td>The maximum deletion size allowed for these lengths.</td>
</tr>
<tr>
<td>pairing.gap.max.insertion.size.50-35</td>
<td>0:4</td>
<td>The maximum insertion size allowed for these lengths.</td>
</tr>
<tr>
<td>pairing.gap.edge.length.deletions.50-35</td>
<td>0:3</td>
<td>The gap edge length for deletions for these lengths.</td>
</tr>
<tr>
<td>pairing.gap.edge.length.insertions.50-35</td>
<td>0:3</td>
<td>The gap edge length for insertions for these lengths.</td>
</tr>
<tr>
<td><strong>Pairing indel parameters for tag lengths 50–25</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pairing.gap.max.deletion.size.50-25</td>
<td>0:11</td>
<td>The maximum deletion size allowed for these lengths.</td>
</tr>
<tr>
<td>pairing.gap.max.insertion.size.50-25</td>
<td>0:4</td>
<td>The maximum insertion size allowed for these lengths.</td>
</tr>
<tr>
<td>pairing.gap.edge.length.deletions.50-25</td>
<td>0:3</td>
<td>The gap edge length for deletions for these lengths.</td>
</tr>
<tr>
<td>pairing.gap.edge.length.insertions.50-25</td>
<td>0:3</td>
<td>The gap edge length for insertions for these lengths.</td>
</tr>
</tbody>
</table>

### Indel sizes

The following mapping parameters affect the indel sizes that can be detected in fragment libraries during tertiary processing:

- `first.map.gapped.deletion`, **default value 19**
- `first.map.gapped.insertion`, **default value 4**
- `second.map.gapped.deletion`, **default value 19**
- `second.map.gapped.insertion`, **default value 4**

The following pairing maximum insertion and deletion parameters affect the size limits of indels that can be discovered in paired data during tertiary processing:

- `pairing.gap.max.deletion.size`
- `pairing.gap.max.insertion.size`
- `pairing.gap.max.deletion.size.75-35`, **etc**
- `pairing.gap.max.insertion.size.75-35`, **etc**

For example, the default value for `pairing.gap.max.deletion.size` is 0:11:0:0:500. This default sets to maximum supported small indel deletion to 500. The parameters ending in library sizes (`pairing.gap.max.deletion.size.75-35`, **etc**) can be used to reset the value for a specific library size.
Mapping output files

The resequencing mapping module generates a BAM file containing alignments in coordinate order. For information about the BAM file, see Appendix C, “File Format Descriptions” on page 493.

The number of BAM files generated depends on the input XSQ files. One BAM file is generated per read-set. If the input data is an XSQ file containing 96 barcodes, then the mapping module generates at least 96 output BAM files. Beads that are unclassified in any barcode are output into a separate additional BAM file. If create.unmapped.bam.files is set to TRUE, then an additional 96 output BAM files corresponding to the unmapped reads are also generated.

The filenames for the output BAM files are created using information from the input XSQ file, including: file base name, file id, index name, and index id. The BAM files are named according to the following patterns:

- Non-indexed BAM files: xsqname-fileID-1.bam
- Indexed BAM files: xsqname-fileID-idx_bcIndex-bcID.bam

The fields in the filenames are:

- **xsqname** – The XSQ file base name, without the .xsq extension.
- **fileID** – The file id for internal XSQ file tracking.
- **bcIndex** – The barcode index.
- **bcID** – The barcode identifier for internal barcode tracking.

The directory structure for mapping output is as follows:

```
outputs/${analysis.output.dir}/
  <mappingIniFileBasename>/
    <sampleName1>/bamfilename1.bam
    <sampleName2>/bamfilename2.bam
    ...
```

The directory name `<mappingIniFileBasename>` is the basename of the mapping INI file. The defaults are fragment.mapping and pair.mapping.

The string `<sampleName*>` is determined by the sample description for the particular read-set.

The mapping output files are used by the BAMStats mapping statistics module and by the LifeScope™ Software tertiary analysis modules.

Mapping statistics

Mapping statistics occur after mapping as an optional post-processing step named BAMStats. BAMStats is on by default in all standard workflows. BAMStats accepts the output of the mapping step and generates statistics files to provide an in-depth understanding of the experimental data and to better detect the presence of anomalies. LifeScope™ Software shell users can display the mapping statistics output data as a chart with a spreadsheet program or other third-party program. A subset of the output from mapping can be visualized in the LifeScope™ Software UI as a series of line and bar charts, if the analysis is run in the projects repository.
BAMStats does not accept 1.x BAM files as input. 1.3 BAM files must be converted to 2.5.1 BAMs before being used with BAMStats.

### Mapping statistics parameters

Table 30 lists the parameters which control the BAMStats output.

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Optional parameters</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bamstats.maximum.coverage</td>
<td>10000</td>
<td>Defines the maximum coverage allowed for locations in the reference. Locations with coverage more than the maximum coverage value are ignored during coverage calculations. Allowed values: Integers $\geq 1$.</td>
</tr>
<tr>
<td>bamstats.maximum.isize</td>
<td>100000</td>
<td>The maximum inset size for LMP and PE libraries. Reads with an insert size more than the specified value are ignored for Insert Range Report calculation. Allowed values: Integers 1–100000.</td>
</tr>
</tbody>
</table>
| bamstats.wig.primary.only | 1 | Use only primary alignments for coverage in WIG file format. Allowed values:  
  - 0: Do not restrict coverage in WIG file format to only primary alignments.  
  - 1: Restrict coverage in WIG file format to only primary alignments. |
| bamstats.wig.binsize | 100 | The bin size for coverage in WIG file format. Allowed values: Integers 1–100000. |
| bamstats.bin.isize | 100 | The bin size for insert range distribution. Allowed values: Integers 1–100000. |
| bamstats.wig.combined.report.both.strands | 1 | Whether to generate a WIG format coverage file with combined data from both the strands. Allowed values:  
  - 0: Do not generate wig files.  
  - 1: Generate the combined coverage file. |
| **Optional resource parameters** | | |
| wall.time | 120 | Total time for the process to complete. |
| java.heap.space | 13000 | Dynamic memory requirement, in megabytes. Do not enter a unit with the numeric value. |
Table 31 lists BAMStats parameters that we do not recommend changing.

**Table 31** BAMStats internal parameter description

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bamstats.run</td>
<td>1</td>
<td>Whether or not to run the BAMStats module. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Do not run the BAMStats module. BAMStats statistics are not generated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Run the BAMStats module during this analysis.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use.</td>
</tr>
<tr>
<td>bamstats.group.stats</td>
<td>1</td>
<td>Enable combined statistics covering all read-sets in the group or sample. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Only statistics for BAM files are generated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Also generate combined statistics for the group or sample.</td>
</tr>
<tr>
<td>bamstats.maximum.mismatches</td>
<td>100</td>
<td>The maximum mismatches allowed in the alignments. Any alignment with more then the specified number of mismatches is ignored while generating reports related to number of mismatches. Allowed values: Integers 0–100.</td>
</tr>
<tr>
<td>bamstats.maximum.baseqv</td>
<td>100</td>
<td>Max base quality values. Any base with base quality value more then the specified value is ignored while generating reports. Allowed values: Integers 0–100.</td>
</tr>
<tr>
<td>bamstats.maximum.mappingqv</td>
<td>255</td>
<td>Maximum mapping quality value. Any alignment with mapping quality value more than the specified value is ignored. Allowed values: Integers 0–255.</td>
</tr>
<tr>
<td>bamstats.wig.minimum.mappingqv</td>
<td>2</td>
<td>Defines the minimum mapping quality allowed for coverage in WIG format. Any alignment with a mapping quality value less than this value is ignored. Allowed values: Integers 0–100.</td>
</tr>
<tr>
<td>bamstats.enable.probe.position</td>
<td>0</td>
<td>Enable probe and position error reports. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Do not generate probe and position error reports.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Also generate probe and position error reports.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> Because these reports are recommended for SNPs analysis, bamstats.ini files in whole genome and targeted resequencing standard workflows have bamstats.enable.probe.position=1.</td>
</tr>
</tbody>
</table>

**Summary of mapping statistics output**

**File formats**

For every input BAM file, a set of statistics files are generated. These files are in CHT, CSV, TXT, and WIG formats. Each CHT file corresponds to one displayed chart. A CHT file specifies the type of chart, the displayed range of each axis, and the data points, without using external references.

The CHT file format is an internal file format based on the CSV file format, with addition header information included. CHT header information is the following:

```plaintext
# name:
# type: scatter2d | pie | vbar | line
# title:
# xaxisname:
# yaxisname:
```

```
```
# xrange: <min>:<tickinterval>:<max>
# yrange: <min>:<tickinterval>:<max>
XAXISNAME, SERIES1NAME, SERIES2NAME, ...
x1, y1.1, y1.2, ...
x2, y2.1, y2.2, ...
x3, y3.1, y3.2, ...

The wiggle format (.wig) is a public format typically used for coverage. Visit their site for more information:

hgdownload.cse.ucsc.edu/goldenPath/help/wiggle.html

A genome browser such as the Integrative Genomics Viewer (IGV) can be used to visualize the coverage. For information is available from their site:

www.broadinstitute.org/igv/

For a collection of input BAM files that belong to a sample, a set of cumulative statistics files are generated. The cumulative statistics files are also in CHT, CSV, TXT, and WIG formats. The cumulative statistics can be visualized in the LifeScope™ Software UI.

Directory structure

The output of the BAMStats module has the following directory structure:

bamstats/
<sampleName>/*cht, *.tab, *.tbl
<sampleName>/<bam>/*.cht, *.tab
<sampleName>/<bam>/Misc/*.csv, *.tab, *.txt, *.wig
<sampleName>/Misc/*.csv, *.tab, *.txt

The position error files and probe errors files are created in the BAM file directories.

Overview

BAMStats generates a comma-separated summary file (BAMfilename-summary.tbl) that summarizes key mapping quality statistics per input BAM file. The summary report contains a snapshot of statistics in all BAM files present in this sample. This file contains one row for each input BAM file in the sample. The summary file is displayed in the LifeScope™ Software UI. See “Summary file” on page 168.

<sampleName>/BAMfilename-summary.tbl

Following directories and reports contain the cumulative statistics from all BAM files that belong to this sample. The Misc folder is not displayed in UI.

<sampleName>/*cht, *.tab
<sampleName>/Misc/*.csv, *.wig, *.tab, *.txt

Following statistics are generated per BAM file in the mapping directory. These reports are not displayed in the UI.

<sampleName>/<BAMfilename>/*cht, *.tab
<sampleName>/<BAMfilename>/Misc/*.csv, *.tab, *.txt, *.wig

In a sample, some of the BAM files possibly represent unhealthy DNA or RNA, causing the cumulative statistics to look poor. The summary tbl file is the unified location for examining the quality of data of all BAM files in the sample.
If the data for a particular BAM file is not as expected, look at the directory-level reports for that BAM file, for details.

**Mapping statistics output files**

This section describes the mapping statistics files generated by the BAMStats module. When statistics reports are separated by tag type, the report’s file name includes the tag in the file name. The tags used in file names are:

<table>
<thead>
<tr>
<th>Library type</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fragment</td>
<td>F3</td>
</tr>
<tr>
<td>Mate-pair</td>
<td>F3</td>
</tr>
<tr>
<td></td>
<td>R3</td>
</tr>
<tr>
<td>Paired-end</td>
<td>F3</td>
</tr>
<tr>
<td></td>
<td>F5-P2</td>
</tr>
</tbody>
</table>

The output files generated by BAMStats include the name of the BAM file. The file name pattern is:

```
BAMFileName-fileID-barcodeID.StatisticsReportName.tag.extension
```

This string is referred to as `prefix` in Table 33, the mapping statistics output table. In the description of mapping statistics output file names in Table 33, the string `tag` is used to refer to the tags listed in Table 32.

A raw numbers file is generated for each chart. See “Raw number files” on page 168.

<table>
<thead>
<tr>
<th>Report</th>
<th>Description, axis information, filename</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alignment Length Distribution</strong></td>
<td>A bar plot giving the distribution of alignment lengths found in various tags used in alignment. The report is separated by tag type to provide visibility into the accuracy of individual tags. Only the primary alignment for each bead is considered in calculating the distribution. Y axis: Frequency X axis: Alignment length (from 0 to the maximum read length) Output file name: <code>prefix.Alignment.Length.Distribution.tag.cht</code></td>
</tr>
<tr>
<td><strong>Base Mismatch Distribution</strong></td>
<td>A bar plot giving the distribution of total number of mismatches found in various tags used in alignment. The bins are summed over all alignment lengths. Only the primary alignment for each bead is considered in calculating the distribution. Y axis: Frequency X axis: Number of mismatches (from 0 to the maximum mismatches allowed) Output file name: <code>prefix.BaseMismatch.Distribution.tag.cht</code></td>
</tr>
</tbody>
</table>

**Table 32  Tags in mapping statistics output file names**

<table>
<thead>
<tr>
<th>Library type</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fragment</td>
<td>F3</td>
</tr>
<tr>
<td>Mate-pair</td>
<td>F3</td>
</tr>
<tr>
<td></td>
<td>R3</td>
</tr>
<tr>
<td>Paired-end</td>
<td>F3</td>
</tr>
<tr>
<td></td>
<td>F5-P2</td>
</tr>
</tbody>
</table>
A report providing a simultaneous picture of the alignment length and number of mismatches distributions in various tags used in alignment. Only primary alignments for each tag are considered in calculating this distribution. The bins range from 0 to the maximum read length, and from 0 to the maximum number of mismatches allowed.

**Note:** This report is located in the Misc folder, and is not displayed in the UI.

**Y axis:** Alignment length (from 0 to the max read length)

**X axis:** Number of mismatches (from 0 to the maximum mismatches allowed)

**Z axis:** Frequency

Output file name: prefix.AlignmentLength.Mismatch.tag.csv

### Distribution of Mismatches by Position

A bar plot providing a distribution of errors (mismatches to reference) by position within the read. Only the primary alignments for each bead are used to generate this distribution.

Mismatch frequency is calculated for each position by dividing number of mismatches for that position by total number of bases for that specific position.

The base-space report is generated on every run.

**Y axis:** Mismatch percentage by read position

**X axis:** Position (from 0 to maximum read length)


### Color Mismatch Distribution

A bar plot providing a distribution of the total number of color mismatches in various tags used in alignment. Only the primary alignments for each bead are used to generate this distribution. This report is only generated if mapping is done in color space.

**Y axis:** Frequency

**X axis:** Number of mismatches (from 0 to the maximum mismatches allowed)


### Distribution of Alignment Length and Color Mismatches

A report providing a simultaneous picture of the alignment length and number of color mismatches distributions in various tags used in alignment. Only primary alignments for each tag are considered in calculating this distribution. The bins range from 0 to the maximum read length, and from 0 to the maximum number of mismatches allowed. This report is only generated if mapping is done in color space.

**Note:** This report is located in the Misc folder, and is not displayed in the UI.

**Y axis:** Alignment length (from 0 to the max read length)

**X axis:** Number of mismatches (from 0 to the maximum mismatches allowed)

**Z axis:** Frequency

Output file name: prefix.AlignmentLength.ColorMismatches.tag.csv

### Base QV Distribution

A bar plot providing a distribution of base quality values generated using the reference assisted error correction/base conversion algorithm. The base QV distributions are separated for each tag as quality of individual tags could be very different.

**Y axis:** Frequency

**X axis:** Base QV (from 0 to the maximum QV)

Output file name: prefix.BaseQV.tag.cht
### Table 33  Mapping statistics output files (continued)

<table>
<thead>
<tr>
<th>Report</th>
<th>Description, axis information, filename</th>
</tr>
</thead>
</table>
| **Base QVs by Position** | A report providing a distribution of base quality values by individual base positions. This report identifies if certain base positions (particularly towards the end of the read) have poor base quality values. All the tags (F3/R3/F5-P2) used in the alignment are combined into a single bin.  
Y axis: Base QV  
X axis: Base position (from 0 to read length)  
Z axis: Frequency  
Output file name: `prefix.BaseQV.by.Position.csv` |
| **Distribution of Mismatches by Base QV** | A line plot providing a distribution of errors (mismatches to reference) by base quality values bins.  
Mismatch frequency is calculated for each base quality value by dividing number of mismatches for that base quality value by total number of bases for that specific base quality value.  
Y axis: Mismatch percentage by Base QV  
X axis: Base QV (from 0 to maximum QV)  
**Note:** This report gives, for each QV, a measure of the raw mismatches at positions with that QV. This information does not measure whether the base QVs generated are well calibrated to the probability of error in that particular base position.  
| **Distribution by Mapping QVs by Tag** | A bar plot providing a distribution of mapping quality values for individual tags (F3/R3/F5-P2). Only the primary alignment for each bead is used in calculating this distribution.  
Y axis: Frequency  
X axis: Mapping QV (from 0 to maximum mapping QV)  
Output file name: `prefix.MappingQV.tag.cht` |
| **Distribution by Pairing QVs** | A bar plot providing a distribution of pairing quality values. Only the primary alignment for each bead is used in calculating this distribution.  
Y axis: Frequency  
X axis: Pairing QV (from 0 to maximum pairing QV)  
Output file name: `prefix.PairingQVs.cht` |
| **Coverage Report** | A line plot providing a distribution of coverage obtained after mapping/pairing. Only the primary alignment for each bead is used in this calculation.  
Y axis: Frequency  
X axis: Coverage (from 0 to number of reads)  
Output file name: `prefix.Coverage.cht` |
| **Coverage Report by Chromosome (Contig) and Base Windows** | |
### Table 33  Mapping statistics output files (continued)

<table>
<thead>
<tr>
<th>Report</th>
<th>Description, axis information, filename</th>
</tr>
</thead>
</table>
| A line plot providing a distribution of coverage within each reference window. The coverage is calculated within each window along a reference chromosome. The window size can theoretically be anywhere from a single base to the contig length. However the calculations of base level coverage are computationally expensive and less interpretable as the window size increases. Only the primary alignment for each bead is used in this calculation. | Y axis: Frequency  
X axis: Coverage (from 0 to number of reads)  

**Coverage by Strand**

| A line plot providing the distribution of coverage within each reference window separated by reference strand (+/-). Only the primary alignment for each bead is used in this calculation. | Y axis: Frequency  
X axis: Coverage (from 0 to number of reads)  

**Coverage files**

Coverage reports in wiggle format. For each genome position, reports the number of reads that cover (map to or span) the position. Because reporting coverage for each position results in very large files, coverage is reported for bins, with each bin spanning a user-defined number of bases. For each bin, the mean coverage of all the positions in that bin is reported. The parameter `bamstats.wig.binsize` controls the size of the bins in this file.

Each coverage file includes a header as the first line. The header lines follow this pattern:

```
track type=wiggle_0 name=<chrname> description=<coverage from positive/negative/both strand> visibility=full color=0,0,255 fixedStep chrom=<chrname> start=<startpos> step=<binsize> span=<binsize>
```

See also “Coverage files” on page 171.

Output file name: coverage_chrnn.pos.wig, coverage_chrnn.neg.wig

**Insert Range Distribution**

| A line plot providing the distribution of insert sizes for paired data (PE and LMP library types). | Y axis: Frequency  
X axis: Insert Range bins  
Output file name: prefix.Insert.Range.Distribution.cht |

**Distribution of Read Pair Types**

For paired data, this report calculates the distribution of read pair types (AAA, AAB, C**, etc.).

| Y axis: Frequency  
X axis: Read pair type  
Output file name: prefix.ReadPair.Type.cht |

**Pairing Statistics**

Table 33  Mapping statistics output files (continued)

<table>
<thead>
<tr>
<th>Report</th>
<th>Description, axis information, filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>A pie chart showing the percentages of the following F3, R3 (or F5) combinations, for paired data:</td>
<td></td>
</tr>
<tr>
<td>• Mapped, Mapped</td>
<td></td>
</tr>
<tr>
<td>• Mapped, Unmapped</td>
<td></td>
</tr>
<tr>
<td>• Unmapped, Mapped</td>
<td></td>
</tr>
<tr>
<td>• Unmapped, Unmapped</td>
<td></td>
</tr>
<tr>
<td>• Mapped, Missing</td>
<td></td>
</tr>
<tr>
<td>• Missing, Mapped</td>
<td></td>
</tr>
<tr>
<td>• Unmapped, Missing</td>
<td></td>
</tr>
<tr>
<td>• Missing, Unmapped</td>
<td></td>
</tr>
<tr>
<td>Output file name: prefix.Pairing.Stats.cht</td>
<td></td>
</tr>
</tbody>
</table>

**Unique Start Positions**

This report lists the following:

- The total number of unique start positions.
- The average number of unique start positions per reference position.

See also “Unique Start Positions” on page 170.

Output file name: BAMfilename-summary.txt, Samplename.-summary.txt

**Summary Report**

The summary file provides statistics for each BAM file in the sample. See “Summary file”.

Output file name: filename.-summary.tbl

**Position error files**

One or more tab-delimited text files generated in the directory containing the input BAM file. This file is used by the diBayes module for SNP analysis. The file records the frequencies of dicolor mismatches between reads and the reference at different positions in a read.

The following position error files are generated, based on library type:

- **Fragment runs** – An F3 position error file is created.
- **Mate-pair runs** – Both F3 and R3 position error files are created.
- **Paired-end runs** – Both F3 and F5 position error files are created.

See also “Position error file” on page 172.

Output file names: BAMfilename_positionErrors_tag_basespace.txt, BAMfilename_positionErrors_tag_colorspace.txt

**Probe error files**

One or more tab-delimited text files generated in the directory containing the input BAM file. This file is used by the diBayes module for SNP analysis. The probe error file records the frequencies of dicolor mismatches between the reads and the reference as a function of different 6-mer probes.

The following probe error files are generated, based on library type:

- **Fragment runs** – An F3 probe error file is created.
- **Mate-pair runs** – Both F3 and R3 probe error files are created.
- **Paired-end runs** – Both F3 and F5 probe error files are created.

See also “Probe error files” on page 172

Output file names: BAMfilename_probeErrors_tag_colorspace.txt
Raw number files

For each CHT and CSV file listed in Table 33, a corresponding TAB file contains the raw numbers and percentages of the distributions, and a total. The following are example contents of the `prefix.Coverage.cht` and the corresponding `prefix.Coverage.tab` file with raw number information:

`prefix.Coverage.cht`:
# name: Coverage Report (N in reference not counted as covered)
# type: line
# title: Coverage Report (N in reference not counted as covered)
# xaxisname: Coverage
# yaxisname: Percent
# xrange:
# yrange: 0:10:100
Coverage,Percent
0,95.6421
1,2.2274
2,0.1928
3,0.0570
4,0.0323
5,0.0256
6,0.0236
7,0.0234
8,0.0238
9,0.0247

`prefix.Coverage.tab`, with raw numbers in the Count column:
# name: Coverage Report (N in reference not counted as covered)
# title: Coverage Report (N in reference not counted as covered)
# total: 3095693195
Coverage   Count   Percent
0        2960786653    95.6421
1         68954650     2.2274
2         5969697      0.1928
3         1764106      0.0570
4         1000572      0.0323
5         790979       0.0256
6         731282       0.0236
7         723622       0.0234
8         736765       0.0238
9         765696       0.0247

Mapping statistics example output

This section provides example output of some mapping statistics output files.

Summary file

The summary file provides statistics for each BAM file in the sample. This list describes labels used in the summary file:

- **NumFragmentsTotal** – The total number of beads, before any filtering on the instrument. This value is also the fragment count in the input XSQ file.
- **NumFragmentsPassingFilters** – The number of beads that pass filtering on the instrument.
- **NumMapped** – The number of reads with primary alignment.
Chapter 9 Run a Resequencing Mapping Analysis

Mapping statistics output files

- **% filtered mapped** – The number of primary reads divided by the number of beads passing instrument filtering \((\text{NumMapped} / \text{NumFragmentsPassingFilters})\).
- **% total mapped** – The number of primary reads divided by the number of total beads \((\text{NumMapped} / \text{NumFragmentsTotal})\).
- **Megabases of coverage** – The sum of alignment lengths, for primary alignments only.

The number of beads filtered out in the instrument is found by subtracting the number of beads that pass filtering from the total number of beads:

\[
\text{NumFragmentsTotal} - \text{NumFragmentsPassingFilters}
\]

The following is example output of a paired summary file. The Tag2 and PredictedInsertSize are not present with fragment data. Each column of data corresponds to a BAM file.

```
#title: BAMStats Summary
BamFileName solid0054_20100203_PE_chr6P_PE.LTcolor-1-1.bam solid0107_20100514_PE_chr6P_PELTcolor-3-1.bam solid0107_20100408_PE_chr6P_PE.LTcolor-2-1.bam
IsColorInBam Y Y Y
IsBaseInXSQ N N N
LibraryType PE PE PE
ReadLength 50x25 50x25 50x25
PredictedInsertSize 165 165 165
Megabases of Coverage 517151581 504214985 555762980
NumFragmentsPassingFilters 8916035 8280061 9519990
NumFragmentsTotal 8916035 8280061 9519990
Tag1-NumMapped 8021248 7653534 8601958
Tag1- % total Mapped 89.96 92.43 90.36
Tag1- % filtered mapped 89.96 92.43 90.36
Tag2-NumMapped 5887893 5994800 6118791
Tag2- % total Mapped 66.04 72.40 64.27
Tag2- % filtered mapped 66.04 72.40 64.27
Tag1-AlignmentLength F3 F3 F3
  Min 25 25 25
  Max 50 50 50
  Avg 46.12 46.30 46.83
  Median 50 50 50
  StdDev 7.46 6.91 6.65
Tag1-MappingQV F3 F3 F3
  Min 0 0 0
  Max 60 60 60
  Avg 36.99 37.34 37.14
  Median 48 48 49
  StdDev 22.28 22.11 22.12
Tag1-BaseQV F3 F3 F3
  Min 1 1 1
  Max 41 41 41
  Avg 35.22 36.16 36.31
  Median 41 41 41
```
#### Chapter 9 Run a Resequencing Mapping Analysis

*Mapping statistics output files*

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Tag1-NumColorMismatches</th>
<th>Tag2-AlignmentLength</th>
<th>Tag2-MappingQV</th>
<th>Tag2-BaseQV</th>
<th>Tag2-NumColorMismatches</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>StdDev</td>
<td>10.69</td>
<td>10.69</td>
<td>9.83</td>
<td>9.66</td>
<td>10.69</td>
<td>10.69</td>
</tr>
<tr>
<td>Min</td>
<td>0</td>
<td>25</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Max</td>
<td>10</td>
<td>25</td>
<td>10</td>
<td>4</td>
<td>4</td>
<td>5978</td>
</tr>
<tr>
<td>Avg</td>
<td>1.38</td>
<td>25.00</td>
<td>35.59</td>
<td>33.72</td>
<td>1.36</td>
<td>0.17</td>
</tr>
<tr>
<td>Median</td>
<td>1</td>
<td>25</td>
<td>48</td>
<td>41</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>StdDev</td>
<td>1.66</td>
<td>25.00</td>
<td>23.62</td>
<td>11.85</td>
<td>1.16</td>
<td>3.15</td>
</tr>
</tbody>
</table>

**Unique Start Positions**

The following is an example of a Unique Start Position report, for fragment data:

Unique Start Positions F3

Total Number of Unique Start Positions: 16536 (16.043154 % of primary alignments)

Average Number of Unique Start Positions per Reference Position: 0.001630
Unique Start Position reports have the following naming patterns:

- `BAMfilename-summary.txt`
- `Samplename-summary.txt`

Coverage files

Example contents for the file `coverage_chr1_positive.wig` are:

```plaintext
browser position chr1:1-200000000
browser hide all
browser pack refGene encodeRegions
# minimumMapq=25, minimumCoverage=1, alignmentFilteringMode=PRIMARY, filterOrphanedMates=false,
track name="BAM Coverage positive strand" description="BAM Coverage positive strand" visibility="full color 0,0,255"
priority=10 yLineMark=0 type=wiggle_0 yLineOnOff=on
variableStep chrom=chr1 span=1
33617 1
33618 1
33619 1
33620 1
33621 1
33622 1
33623 1
33624 1
33625 1
33626 1
33627 1
...
```

Example contents for the file `coverage_chr1_negative.wig` are:

```plaintext
browser position chr1:1-200000000
browser hide all
browser pack refGene encodeRegions
# minimumMapq=25, minimumCoverage=1, alignmentFilteringMode=PRIMARY, filterOrphanedMates=false,
track name="BAM Coverage negative strand" description="BAM Coverage negative strand" visibility="full color 0,0,255"
priority=10 yLineMark=0 type=wiggle_0 yLineOnOff=on
variableStep chrom=chr1 span=1
5743 2
5744 2
5745 2
5746 3
5747 3
```
Position error file

The following is a truncated example of position error file contents:

```
# Generated by: BAM Stats
Plugin(20110415_F12_KP_FC1_1x50frag_AN_02-1-1.bam-t f3,r3 -o /
data/whist/Lifescope_Results/projects/cafe/pHRECC88020/a1/
outputs/bamstats -r /share/lifescope/Reference_Data/2.0/
lifetech/hg18/reference/human_hg18.fa ) (Base space)
# Version: 1.0
# Date: May 2, 2011
##? Total read positions = 201701987
##? Missing base calls = 34330
##? Unresolved reference positions = 4
##position_[refCall][readCall] nErrors nReadOccurrences Error frequency
1_AC 93033 1272759 0.0044
1_AG 25231 1255139 0.0059
1_AT 184621 37994592 0.0049
1_CA 66929 37196998 0.0018
```

Probe error files

The following is an example of probe error file contents:

```
# Generated by: CountErrorsByPrefix(-t,F3,-d,-1,-k,6,--
mask1=01111,--maskn=11111,--tmfile=/share/apps/corona-r33455M/
etc/analysis/6mers.Tm,/data/results/RegressionDriver/
CaseManager/results/latestBuild/2011-02-19_101137_r3
3455M3.5-trunk_sanityTest/mutationST/case0009/
F3_alignment.v2.gff)
# Version: 1.0.29869
# Date: 02/19/2011 06:41 PM
##ref context num errors num occurences err frequency
Tm
AAAAAA 0 16 0.000000 -24.04
CAAAAA 0 6 0.000000 -24.53
GAAAAA 0 18 0.000000 -25.87
TAAAAA 0 0 0.000000 -37.81
ACAAAA 0 0 0.000000 -24.77
```

Run BAMStats standalone

This section describes requirements to generate mapping statistics outside of the context of the mapping module. These requirements are:

- **Directory structure** – The input directory for BAMStats module must mimic the output directory structure of mapping:
FAQ – Mapping

1

How does the local alignment approach affect mapping?

Using the local alignment approach removes the constraint of a whole-read alignment, and mapping rate is significantly increased. It is not uncommon for a poor dataset with a 30% mapping rate at 50 bp, to reach more than 60% mapping rate with the mapping algorithm used in LifeScope™ Software.

While the majority of alignments are full length, some alignments can vary in length. If a read has many errors towards the end, the final alignment will not include these positions if a shorter alignment receives a better overall score.

2

How do I increase mapping rate?

Mapping rate increases with each additional round of mapping. However, the gain in rate comes at the cost of increased runtime and disk requirement. Testing has found sets of mapping schemes that strike a good balance between mapping rate and runtime for 25-, 35-, and 50-bp reads. These schemes are shipped as LifeScope™ Software defaults and should work well for most purposes.

For 50-bp reads, two keys in the mapping.ini file define how many rounds of mapping are run, and what happens in each round. The default values of the keys are shown below, and their meanings are further explained in Table 26 on page 142.

Note: Repetitive schemes are empty by default.

- mapping.scheme.unmapped.50 = 25.2.0, 25.2.15
- mapping.scheme.repetitive.50 = 38.3.0, 25.2.0
The repetitive scheme does not affect the number of mapped reads, and setting it only increases the chance of finding better alignments for repetitive reads. If your main goal is to improve throughput, first try to add more iterations at the unmapped scheme. For example, add another step of 25.2.25 at the end of the unmapped scheme above.

3

What is a good seed for my application?

Consider the following factors when picking seed parameters:

- Mapping is slower when more mismatches are allowed in the seed. However, allowing more mismatches improves the mapping rate.
- Shorter seeds have higher mapping rates, but also lead to more spurious alignments.
- Color-calls at the beginning of the read are more reliable than those at the end. Therefore, you have the option to anchor the seed near the front of the read. However, applications such as transcriptome sequencing are scenarios where it is an advantage to anchor the seed near the end of the read.

For 50-bp reads, the default setting of 25.2.0 for the first round, followed by 25.2.15 in the second round, delivers good results in most cases. For a single round, 30.3.0 is recommended.

4

What reads scenario achieves the best accuracy?

The best performance is achieved by considering 2BE and 4BE reads, their alignment, and reference. Typically, base-space reads generated by ECC have lower mapping throughput. Base-space reads generated from combined 2BE and 4BE calls and no reference have lower accuracy, compared to ECC calls.

5

How do I interpret a mapping quality value?

First of all, it is worth noting that the mapping quality value (QV) is not like the p-value in BLAST. In BLAST, the p-value is used to estimate the likelihood that the aligned sequences are related. In general, reads come from the same source and it is known that the two are related.

The purpose of mapping QV is to estimate the probability that the read originates from the mapped genomic location. The estimate is determined mainly by the difference in significance between the best- and second-best hits.

The numeric interpretation of mapping QV is the same as the base call QV. A mapping QV of 10 means that there is a 90% chance that the alignment is correct. A mapping QV of 20 means that there is a 99% chance that the alignment is correct.
6

How much RAM do I need to analyze human samples?

For optimal performance, 24 GB of RAM per cluster node is recommended for human samples. If the cluster node has less than 24 GB of available RAM, mapreads can split the genome into smaller segments, and align to each segment sequentially. Splitting the genome into smaller segments is implemented inside mapreads and is completely transparent to the user.

Mapping with less than the recommended amount of RAM slows down performance. If the cluster node has 16 GB of RAM, mapping human samples will be roughly 30% slower compared to a machine with 24 GB of RAM.

7

In the mapping statistics genome coverage calculation report, how are Ns in the reference counted?

Ns in the reference are counted as missing coverage. For example, in the human genome, about 7% of the reference sequence consists of Ns. This means that the coverage calculation will never be reported as higher than 93%. If the frequency distribution says 7.49% of the genome is uncovered, this means that about 0.34% of the non-N reference sequence does not have reads mapped to it, and that also the 7.15% of the genome that consists of N does not have reads mapped to it.

8

What should I do to ensure balanced allele ratios at heterozygous positions?

The base-translation algorithm, refcor, uses instrument color-space data (ECC or non-ECC) and the reference sequence to produce the most likely base calls, along with a Phred-scale quality value. The degree to which the reference influences the base call is controlled by the refcor.reference.weight parameter. A position is converted to reference only if the refcor.reference.weight setting is higher than the difference between the quality values of a correct color call and of an adjacent erroneous color call that supports reference.

The refcor.reference.weight default setting, 8, is chosen to be low enough to ensure balanced allele ratios, and also be large enough both to distinguish correct from incorrect color calls and to maximize base accuracy.

In the event an incorrect reference base call is made, the disagreement between evidence sources causes that call’s QV to be low. Incorrect calls can be filtered out based on QV threshold, and this filtering improves allele ratios.

Another way of filtering low quality base-calls is to adjust the parameter refcor.base.filter.qv, that converts all base calls with a quality value less than refcor.base.filter.qv into Ns. This filtering has the effect of setting QV to zero in the calls with low confidence. To decrease the number of N base calls, reduce this threshold (for example, from the default of 10 to 5).
**FAQ – Pairing**

1

**How is the uniqueness of a pair determined?**

A pair of reads is unique when there is exactly one good AAA pair. With the advent of local alignment, the likelihood of finding only one good pair is decreased. As a result, a different heuristic is used to determine “uniqueness”.

Consider an alignment of length $L$ with $M$ mismatches. If the local score is defined as $L + (m-1)M$, where $m \leq 0$ is the mismatch penalty, then the score of each good pair candidate is the sum of its two constituent local scores.

A pair of tags is considered unique if it has only one good pair, or if the score of the best pair is at least $X$ greater than the score of the second best pair. $X$ is specified by the `pair.uniqueness.threshold` key, and has a default value of 10.0. See Table 29 on page 155 for a description of this and all pairing parameters.

2

**What do the 3-letter pair classifications mean?**

Table 34 lists a summary of genomic code classifications for mate-pair pairing runs. See Table 35 on page 177 for a summary of genomic code classifications for paired-end runs.

<table>
<thead>
<tr>
<th>Class name</th>
<th>Strand</th>
<th>Orientation</th>
<th>Insert size</th>
<th>Preference</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAA</td>
<td>Same</td>
<td>R3 to F3</td>
<td>Normal</td>
<td>1</td>
</tr>
<tr>
<td>AAB</td>
<td>Same</td>
<td>R3 to F3</td>
<td>&lt; min</td>
<td>2</td>
</tr>
<tr>
<td>AAC</td>
<td>Same</td>
<td>R3 to F3</td>
<td>&gt; max</td>
<td>2</td>
</tr>
<tr>
<td>ABA</td>
<td>Same</td>
<td>F3 to R3</td>
<td>Normal</td>
<td>2</td>
</tr>
<tr>
<td>ABB</td>
<td>Same</td>
<td>F3 to R3</td>
<td>&lt; min</td>
<td>3</td>
</tr>
<tr>
<td>ABC</td>
<td>Same</td>
<td>F3 to R3</td>
<td>&gt; max</td>
<td>3</td>
</tr>
<tr>
<td>BAA</td>
<td>Different</td>
<td>Outward</td>
<td>Normal</td>
<td>3</td>
</tr>
<tr>
<td>BAB</td>
<td>Different</td>
<td>Outward</td>
<td>&lt; min</td>
<td>4</td>
</tr>
<tr>
<td>BAC</td>
<td>Different</td>
<td>Outward</td>
<td>&gt; max</td>
<td>4</td>
</tr>
<tr>
<td>BBA</td>
<td>Different</td>
<td>Inward</td>
<td>Normal</td>
<td>3</td>
</tr>
<tr>
<td>BBB</td>
<td>Different</td>
<td>Inward</td>
<td>&lt; min</td>
<td>4</td>
</tr>
<tr>
<td>BBC</td>
<td>Different</td>
<td>Inward</td>
<td>&gt; max</td>
<td>4</td>
</tr>
</tbody>
</table>
The first letter determines whether the two tags match the same strand:
- A – Match the expected strand.
- B – Match different strands.

The third letter determines if the distance between two hit positions is within the correct range:
- A – Within the correct range.
- B – The distance is less than the insert size measurement minimum (set with `default.minimum.insert.size` and denoted by "< min" in the table above).
- C – The distance is greater than the insert size measurement maximum (set with `default.maximum.insert.size` and denoted by "> max" in the table above).

The default minimum insert size is 0. For paired-end and long mate pair workflows, the default maximum insert sizes are 2000 and 20,000, respectively. An insert size measurement automatically determines the actual values using the defaults as the largest interval, and the defaults are used if the insert size measurement fails.

The middle letter has different meanings, depending on whether the first letter is A or B. If the first letter is A, the second letter indicates if the order of the two tags is correct:
- A – The order is correct, and R3 is upstream of F3.
- B – The order is incorrect, and F3 is upstream of R3.

When the two tags are on different strands, the second letter indicates whether the two tags are pointing toward or away from each other
- A – Pointing away from each other.
- B – Pointing toward each other.

In addition, C** classification is assigned to a pair of reads that have one hit each, but are on different chromosomes. If both F3 and R3 tags are present, but only one has a unique hit while the other one is unmapped, the pair is labeled D**. Finally, if one tag from a pair has a unique hit but the other one is missing, as opposed to unmapped, then the pair is classified as E**.

The Preference value indicates likelihood of the variation can occur in nature. Larger values indicate the variation is less likely to occur in nature.

Table 35 shows the classifications for paired-end runs. Summary statistics of these categories are calculated in the optional BAMStats Read Pair Type Statistics Report (.ReadPair.Type.cht).

**Table 35** Genomic code classifications, for paired-end pairing

<table>
<thead>
<tr>
<th>Class name</th>
<th>Strand</th>
<th>Orientation</th>
<th>Insert size</th>
<th>Preference</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAA</td>
<td>Different</td>
<td>Inward</td>
<td>Normal</td>
<td>1</td>
</tr>
<tr>
<td>AAB</td>
<td>Different</td>
<td>Inward</td>
<td>&lt; min</td>
<td>2</td>
</tr>
<tr>
<td>AAC</td>
<td>Different</td>
<td>Inward</td>
<td>&gt; max</td>
<td>2</td>
</tr>
<tr>
<td>ABA</td>
<td>Different</td>
<td>Outward</td>
<td>Normal</td>
<td>2</td>
</tr>
<tr>
<td>ABB</td>
<td>Different</td>
<td>Outward</td>
<td>&lt; min</td>
<td>3</td>
</tr>
<tr>
<td>ABC</td>
<td>Different</td>
<td>Outward</td>
<td>&gt; max</td>
<td>3</td>
</tr>
<tr>
<td>BAA</td>
<td>Same</td>
<td>F3 to F5</td>
<td>Normal</td>
<td>3</td>
</tr>
<tr>
<td>BAB</td>
<td>Same</td>
<td>F3 to F5</td>
<td>&lt; min</td>
<td>4</td>
</tr>
</tbody>
</table>
FAQ – BAMStats

Why does LifeScope™ Software report more duplicate beads than Picard reports?

LifeScope™ Software reports significantly more duplicates than does Picard. There are a number of underlying reasons why our mark duplicates module is more aggressive in marking duplicates than Picard:

- By design, LifeScope™ Software marks reads with the same read start position, even if they have soft or hard clipping at the start of the read, as duplicates. We consider this to be desirable behavior.
- There are also some differences in the handling of reads mapped to different chromosomes, and reads that are not “proper pairs” (because of orientation, strand, or distance).
- In the case of reads from multiple BAM files and with non-unique bead IDs across multiple files, LifeScope™ Software marks reads with the same bead ID as duplicates.
- LifeScope™ Software by default marks duplicates only within a single BAM file, with an option to mark duplicates across multiple BAM files. The multiple BAM option is appropriate for multiple barcodes from the same sample run on the same lane. For these runs, our mark duplicates module behaves similarly to Picard and marks around the same number of duplicates as Picard. The duplicates detected are not exactly the same, because the algorithms differ slightly.

**FAQ – BAMStats**

1

In the BAMStats summary table, why do the NumFragmentsPassingFilters and NumFragmentsTotal fields have the same totals?

NumFragmentsTotal reports the total number of fragments (read pairs or reads) passed from the instrument. NumFragmentsPassingFilters reports the number of fragments that pass the instrument filter. For the 5500 instrument, all reads pass filters when using color space, and so these two numbers are the same. If the run is performed with ECC, then the base space reads have filters applied to remove those of lowest quality. For an ECC run mapped in base space using LifeScope™ Software, the filtered reads are reported in LifeScope™ Software, and the Total and Filtered numbers are different.
What changes are required for a job with a large number of contigs?

To run BAMStats with a large number of contigs (around 500 or more) and a very high `bamstats.maximal.coverage` value (around 300,000 or more), you can increase the heap space or decrease the maximum coverage value, or do both:

- Decrease the `bamstats.maximal.coverage` value.
  You could try 100000 or 10000 for the maximum coverage value.
- Increase heap space by increasing the `memory.request` parameter for the mapping module, and increase the `java.heap.space` parameter in the `bamstats.properties` file.

  **Note:** The heap space and memory request parameters directly affect job scheduling and impact run times. The values to use also depend on your data sets and compute environment.

(It is also possible, but not recommended in most cases, to change the system property `max.head.node.java.heap.space` from the default of 64 to 128 or 256. This parameter is in the `<installdir>/etc/analysis/system.properties` file. Because changes to system properties affect all runs of all users, you could consider changing the parameter back to its original value after the large contig run completes. Setting `max.head.node.java.heap.space` to 128 or 256 could present a problem if your head node is extremely loaded.)
Chapter 9  Run a Resequencing Mapping Analysis

FAQ – BAMStats
CHAPTER 10

Run a SNPs Analysis

This chapter covers:

- Overview ................................................................. 181
- SNPs input files. ......................................................... 182
- Examples of running a SNPs analysis .............................. 184
- SNPs runtime parameters ........................................... 185
- SNPs output files ....................................................... 192
- SNPs algorithm description ......................................... 201
- FAQ – SNPs .............................................................. 204

Overview

The LifeScope™ Genomic Analysis Software SNPs analysis module uses the diBayes algorithm to find Single Nucleotide Polymorphisms (SNPs). The diBayes package performs independent SNP analysis at each position in the reference, using either a Bayesian or Frequentist algorithm.

The SNPs module is used to call Single Nucleotide Polymorphisms (SNPs) from mapped and processed SOLiD® System reads. The module takes the reads, quality values, the reference sequence, and error information of each SOLiD® System slide as its input, and calls SNPs.

The SNPs module takes the mapped and processed SOLiD® System reads, quality values, the reference sequence, and error information of each SOLiD® System slide as its input, and calls SNPs.

The SNPs module creates these results files:

- A list of SNPs.
- A quartile file with coverage as well as color and base quality value distribution.
- (Optional) A consensus FASTA file with the same number of bases as the reference sequence.
- (Optional) A consensus calls file with a list of all covered positions.
- (Optional) A collection of annotated files.
The consensus calls file and the quartile file are each generated as one file per contig. The list of SNPs and the consensus FASTA file are each generated both as one file per contig and also as a consolidated file for the entire run. Annotated files are generated for the entire run, not for individual contigs. See “SNPs output files” on page 192.

### SNPs input files

The diBayes input files include the reference file, the regions of interest file (BED), the BAM file (the output of the mapping-pairing or enrichment modules), and position error and probe error files (the output of the BAMStats mapping statistics module). When you run a standard workflow that includes a SNP analysis, the wiring of these input files is handled automatically by the LifeScope™ Software command shell. These input files are described in Table 36.

**Table 36** diBayes input file description

<table>
<thead>
<tr>
<th>Input File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM file input</td>
<td>The BAM input files are generated by LifeScope™ Software mapping or enrichment modules. The SNPs module accepts one or more BAM files as input. Each BAM file must have only one read group and only one library type. The SNPs module does not support merged BAM files with multiple read groups or with different library types (within one BAM file). For reads containing Ns, the SNPs module uses all information in the reads except for the missing bases. The SNPs module accepts both color space BAM files (from regular two base color-encoding runs) and base space BAM files (from ECC runs), but does not support combining both color space and base space BAM files in one run. The module determines what algorithms to use automatically based on the input files. Multiple base space BAM files can be processed as combined input using base space algorithms. Multiple color space BAM files are processed as combined input using color space algorithms. If color space BAM files are missing color space information (for instance when users choose not to output color information into the BAM files), then base space algorithms are used. <strong>IMPORTANT!</strong> Do not concatenate BAM files from different run types. <strong>IMPORTANT!</strong> Do not merge the BAM files before running the SNPs module. The module can accept multiple BAM files of different run types as its input. <strong>Note:</strong> If you move the input BAM file from its default location (not recommended), you must also move its corresponding position and probe error files to the same directory (the BAM file and its corresponding position and probe error files have the same file name prefix).</td>
</tr>
<tr>
<td>Reference genome</td>
<td>The sequence to which the reads are aligned and mapped. The recommended format for the reference file is the FASTA format, for example, chr20.validated.fasta. <strong>Note:</strong> When you run the SNPs module, you must use the same reference that was used for mapping. If different reference files are used, during the SNPs module analysis the coordinates in the BAM file might not correctly represent the coordinates in the reference file. The SNPs module might reject the input BAM files if the reference information in headers of BAM files is not matched to the reference provided. The reference sequence might have multiple chromosomes or contigs, and might contain IUB codes at positions of known SNPs.</td>
</tr>
</tbody>
</table>
### Table 36 diBayes input file description (continued)

<table>
<thead>
<tr>
<th>Input File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regions of interest file</td>
<td>The regions of interest file, in BED format, contains a list of defined regions. This file is required input file for target resequencing workflows and optional for whole genome resequencing workflows. If users want to use a BED file in whole genome resequencing workflows, the file must be sorted based on the start positions for defined regions. The BED files for target resequencing workflows do not have this requirement. Instead the enrichment module within the target resequencing workflow sorts the BED file and passes the sorted BED file to the SNPs module. We recommend users to choose target resequencing workflows for analyses that include a BED file.</td>
</tr>
<tr>
<td>F3 [R3/F5] position error file</td>
<td>F3 [R3/F5] position error files are tab-delimited text files created during secondary analysis. The files record the frequencies of color/base mismatches between reads and the reference at different positions in a read. For fragment runs, only an F3 position error file is needed. Both F3 and R3[F5] position error files are created for mate-pair (paired-end) runs, and both are needed to run diBayes. Position error files are automatically generated during mapping statistics if BAMStats is enabled (recommended). If position error files are not available, the SNPs module uses internal default position errors for SNP calling. Using default position error files might slightly affect the module’s accuracy. Position error files are located in the mapping results folder, and share the same name prefix as the output BAM files. You do not specify their names or location when you run SNPs in a standard workflow. If you run SNPs outside of a workflow and also move the mapping output BAM files, always copy the position and probe error files to the same directory as the BAM files.</td>
</tr>
<tr>
<td>F3 [R3/F5] probe error file</td>
<td>F3 [R3/F5] probe error files are tab-delimited text files that record the frequencies of dicolor mismatches between the reads and the reference as a function of different 6-mer probes. LifeScope™ Software calculates the probe error files. For fragment runs, only F3 probe error files are needed. Both F3 and R3[F5] probe error files are created for mate-pair (paired-end) runs, and both are needed to run diBayes on paired data. Different SOLiD® System runs of the same sample might generate different F3 [R3/F5] probe error files for each run, because of the random nature of probe errors. Probe error files are automatically generated during mapping statistics if BAMStats is enabled (recommended). However, probe error files are not generated for base space data. If the probe error file is not found, the SNPs module uses internal default probe error files for SNP calling. Using default probe error files might slightly affect the module’s accuracy. Probe error files are located in the mapping results folder, and share the same name prefix as the output BAM files. You do not need to specify their name or location when you run SNPs in a standard workflow. If you run SNPs outside of a workflow and also move the mapping output BAM files, always copy the position and probe error files to the same directory as the BAM files.</td>
</tr>
</tbody>
</table>
Examples of running a SNPs analysis

Examples in a standard workflow

The following standard workflows provide examples of running the SNPs module:

- genomic.resequencing.frag
- genomic.resequencing.lmp
- genomic.resequencing.pe
- targeted.resequencing.frag
- targeted.resequencing.pe

The following are example shell commands using reads and references from the LifeScope™ Software repository.

```bash
# log into the shell
lscope.sh shell -u username -w password
# cd to the projects repository
cd /projects
# create a project and open it
mk ecoli
cd ecoli
# create an analysis and open it
mk run1
cd run1
# define the analysis type
set workflow genomic.resequencing.pe
# define the input data
add xsq run0209a_50_PE.xsq
add xsq run0209b_50_PE.xsq
# specify the reference
set reference hg19
# optionally change SNPs defaults after this line
# list the analysis configuration
ls
# start the analysis
run
# list progress information for the run
ls
```

Dummy file names for XSQ files are shown here.

See for Chapter 7, “Run a Standard Workflow Analysis” on page 103 for more information on running standard workflows. Chapter 6, “Run a Command Shell Analysis” on page 75 for information on shell commands and syntax.

As a demo analysis

The optional examples download includes an example of how to run the SNPs module by itself, as a single analysis that is not part of a standard workflow. The demo example is not recommended for general use.

See Appendix H, “Demo Analyses” on page 567 for information on running the SNPs module as a standalone analysis.
SNPs runtime parameters

Table 38 describes the SNPs module runtime parameters.

In order to change a parameter value in your analysis, use the `set param` shell command to replace the line `# optionally change SNPs defaults after this line` in the example commands in “Examples in a standard workflow”. For instance, to change the call stringency, use this shell command:

```
set dibayes.call.stringency high tertiary/dibayes.ini
```

These workflows use the INI file name `dibayes.ini`:

- `genomic.resequencing.lmp`
- `genomic.resequencing.pe`
- `targeted.resequencing.pe`

For the workflow `targeted.resequencing.frag`, use `dibayes.targeted.frag.ini`. For the workflow `genomic.resequencing.frag`, use `dibayes.genome.ini`.

Certain parameters affect only the module’s base-space or color-space algorithm. These parameters are noted in Table 38. Changes to a base-space parameter’s setting have no effect if the module runs its color-space algorithm. Similarly, changes to a color-space parameter’s setting have no effect if the module runs its base-space algorithm.

Table 37 describes a workflow parameter, `analysis.space`, that is used in the SNPs module. The `analysis.space` parameter is set in the `tertiary/global.ini` file.

### Table 37  Analysis space parameter used in the SNPs module

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>analysis.space</code></td>
<td><code>auto</code></td>
<td>Specifies your preference for base space or color space analysis. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <code>auto</code>: The analysis space is determined based on the mapping space and the presence of color space data in the analysis BAM files.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <code>base</code>: Prefer base space analysis.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <code>color</code>: Prefer color space analysis.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>See “Analysis space” on page 69 for a description of other factors determining the analysis space used for SNPs.</td>
</tr>
</tbody>
</table>

Table 38 describes SNPs parameters.

### Table 38  SNP (diBayes) parameter description

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Optional general parameters</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>dibayes.reads.min.mapping.qv</code></td>
<td>8</td>
<td>Requires that the mapping quality value of the read be higher than this minimum mapping/pairing QV. Allowed values: Integers 0–100. See “The reads mapping QV parameter” on page 191.</td>
</tr>
<tr>
<td><code>dibayes.detect.2.adjacent.snps</code></td>
<td>0</td>
<td>Whether or not to detect two adjacent SNPs. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Do not call a second adjacent SNP as a SNP.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: If two adjacent SNPs are found, call both as SNPs.</td>
</tr>
</tbody>
</table>
### Table 38  SNP (diBayes) parameter description (continued)

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dibayes.polymorphism.rate</td>
<td>0.001</td>
<td>The expected frequency of heterozygotes in the population, for example, 0.001 in humans. Allowed values: Float 0.0–1.0.</td>
</tr>
<tr>
<td>dibayes.het.skip.high.coverage</td>
<td>1</td>
<td>Do not call SNPs when the coverage of position is too high compared to the median of the coverage distribution of all positions. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Call a position a SNP under these conditions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Do not call a position a SNP under these conditions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> Enable this filter for whole genome resequencing application. Disable it for whole transcriptome or target resequencing. See “The skip high coverage filter” on page 191.</td>
</tr>
<tr>
<td>Required parameter</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dibayes.call.stringency</td>
<td>medium</td>
<td>Specifies the SNPs call stringency. Each call stringency setting specifies a set of filters. An individual filter parameter setting overrides the call stringency setting, in case of conflict. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• highest: Recommend when a very low false positive tolerance is allowed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• high: Requires the allele on both strands.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• medium: No both-strand requirement.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• low: Very aggressive.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• lowest: Even more aggressive. Very minimal filtering.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• low_frequency: For use with the low frequency variant workflows.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lower stringency settings result in more SNP calls, but more false positives are also expected. Higher settings result in fewer SNP calls, with fewer false positives. See “The call stringency parameter” on page 190.</td>
</tr>
<tr>
<td>Optional read filters</td>
<td></td>
<td>The entire read is excluded from analysis if a read fails to pass one of these filters.</td>
</tr>
<tr>
<td>dibayes.reads.with.unmapped.mate.include</td>
<td>1 for PE data; 0 for other data</td>
<td>Whether or not to include reads that only have one tag mapped (their mate tags are either unmapped or missing). Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Do not include these reads.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Include these reads.</td>
</tr>
<tr>
<td>dibayes.reads.with.indel.exclude</td>
<td>1</td>
<td>Whether or not to exclude reads that have indels. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Do not exclude reads that have indels.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Exclude reads that have indels.</td>
</tr>
<tr>
<td>dibayes.reads.only.uniquely.mapped.allow</td>
<td>0</td>
<td>Whether or not to require that reads be uniquely mapped. A very stringent filter. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Do not require that reads be uniquely mapped.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Require uniquely mapped reads. Filter out reads that are not uniquely mapped.</td>
</tr>
<tr>
<td>dibayes.reads.max.mismatch.alignlength.ratio</td>
<td>1.0</td>
<td>The threshold of mismatch/alignment-length ratio. The reads whose mismatch/alignment length ratio is higher than this specified threshold are filtered (are ignored). Allowed values: Float 0.0–1.0.</td>
</tr>
</tbody>
</table>
Table 38  SNP (diBayes) parameter description (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>dibayes.reads.min.alignlength.readlength.ratio</code></td>
<td>0.0</td>
<td>The threshold of alignment-length/read-length ratio. The reads whose alignment-length/read-length ratio is less than this specified threshold are filtered (are ignored). Allowed values: Float 0.0–1.0.</td>
</tr>
</tbody>
</table>

Optional general position filters
The set of pile-up color/base calls at a given position is excluded from both heterozygous and homozygous SNP analysis if the position pile-up fails to pass one of these filters.

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>dibayes.snps.min.filteredreads.rawreads.ratio</code></td>
<td>0.0</td>
<td>The minimum ratio of the filtered reads and raw reads. A position whose ratio of filtered reads to raw reads is below this value is not considered. Allowed values: Float 0.0–1.0.</td>
</tr>
<tr>
<td><code>dibayes.snps.min.color.qv</code></td>
<td>7</td>
<td>Require that the candidate allele have at least this base quality value. A color call that fails this filter is excluded from the analysis, but the color pile-up is still considered for SNP calling. Allowed values: Integers 0–62. Color space only.</td>
</tr>
<tr>
<td><code>dibayes.snps.min.base.qv</code></td>
<td>26</td>
<td>Require that the candidate allele have at least this base quality value. A base call that fails this filter is excluded from the analysis, but the base pile-up is still considered for SNP calling. Allowed values: Integers 0–41. Base space only. See “The minimum base QV filters” on page 191.</td>
</tr>
<tr>
<td><code>dibayes.snps.min.nonref.base.qv</code></td>
<td>26</td>
<td>Require that the non-reference allele have at least this base quality value. A base call that fails this filter is excluded from the analysis, but the base pile-up is still considered for SNP calling. Allowed values: Integers 0–41. Base space only. See “The minimum base QV filters” on page 191.</td>
</tr>
<tr>
<td><code>dibayes.snp.both.strands</code></td>
<td>0</td>
<td>Whether or not to require that the novel allele is present on both strands and statistically similar represented on both strand for both heterozygous and homozygous SNPs. Allowed values: 0: Do not make the above requirement. 1: Make the above requirement.</td>
</tr>
</tbody>
</table>

Optional heterozygous position filters
The set of pile-up color/base calls at a given position is excluded from only heterozygous SNP analysis if the position pile-up fails to pass one of these filters.

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>dibayes.het.min.allele.ratio</code></td>
<td>0.15</td>
<td>The less-common allele must be at least this proportion of the reads of the first two most common valid alleles. Allowed values: Float 0.0–0.5.</td>
</tr>
<tr>
<td><code>dibayes.het.min.coverage</code></td>
<td>2</td>
<td>Require at least this coverage to call a heterozygous SNP. Allowed values: Integers =&gt; 0.</td>
</tr>
<tr>
<td><code>dibayes.het.min.start.pos</code></td>
<td>2</td>
<td>The minimum number of unique start positions required to call a heterozygote. Allowed values: Integers =&gt; 0.</td>
</tr>
<tr>
<td><code>dibayes.het.min.nonref.base.qv</code></td>
<td>26</td>
<td>Requires the non-reference allele to have at least this average base quality value to call a heterozygous SNP. Allowed values: Integers 0–41. Base space only.</td>
</tr>
</tbody>
</table>
### Table 38 SNP (diBayes) parameter description (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>
| dibayes.het.min.nonref.color.qv         | 7       | Requires the non-reference allele to have at least this average color quality value to call a heterozygous SNP. When you know whether your input reads contain ECC data (from ICS 1.1 or higher) or not, use one of the following recommended values:  
  - Non-ECC data: 7  
  - ECC data (from ICS 1.1 or higher): 20  
  Allowed values: Integers 0–62. Color space only. |
| dibayes.het.min.validreads.totalreads.ratio | 0.65    | The proportion of the total reads containing either of the two candidate alleles. Filters positions with high raw-error rates.  
  Allowed values: Float 0.0–1.0. |
| dibayes.het.min.adjacentsnp.allele.count | 2       | Minimum valid tricolor counts (Het).  
  Allowed values: Integers >= 0. Color space only. |

### Optional homozygous position filters

The set of pile-up color/base calls at a given position is excluded from only homozygous SNP analysis if the position pile-up fails to pass one of these filters.

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>
| dibayes.hom.min.coverage                | 1       | Requires at least this coverage to call a homozygous SNP.  
  Allowed values: Integers >= 0. |
| dibayes.hom.min.nonref.allele.count     | 2       | Requires at least this non-reference allele count to be called a homozygous SNP.  
  Allowed values: Integers >= 0. |
| dibayes.hom.min.nonref.color.qv         | 7       | Requires the non-reference allele to have at least this average color quality value to call a homozygous SNP. When you know whether your input reads contain ECC data (from ICS 1.1 or higher) or not, use one of the following recommended values:  
  - Non-ECC data: 7  
  - ECC data (from ICS 1.1 or higher): 8  
  Allowed values: Integers 0–62. Color space only. |
| dibayes.hom.min.nonref.base.qv          | 26      | Requires the non-reference allele to have at least this average base quality value to call a homozygous SNP.  
  Allowed values: Integers 0–41. Base space only. |
| dibayes.hom.min.nonref.start.pos        | 2       | Requires the non-reference allele to have at least this minimum number of unique start positions, to call a homozygote.  
  Allowed values: Integers >= 0. |

### Optional low frequency variant filters

For use with the low frequency variant (LFV) workflows. When dibayes.call.stringency is set to low_frequency, the filters in this section are set to their defaults. Other stringency settings effectively render these filters inactive.

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>
| dibayes.het.lca.both.strands            | 0       | The less common allele on both strands.  
  Allowed values: 0,1.  
  Recommended value for LFV workflows: 1. |
| dibayes.het.min.lca.start.pos           | 0       | The minimum unique start position of the less common allele.  
  Allowed values: Integers >= 0.  
  Recommended value for LFV workflows: 7. |
| dibayes.het.min.lca.color.qv            | 0       | The minimum average color QV of the less common allele.  
  Allowed values: Integers >= 0.  
  Recommended value for LFV workflows: 30. |
### Chapter 10  Run a SNPs Analysis

#### SNPs runtime parameters

##### Table 38  SNP (diBayes) parameter description (*continued*)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dibayes.het.max.diff.color.qv</td>
<td>99</td>
<td>The maximum difference of color QVs of the most common and less common alleles. Allowed values: Integers -100–100. Recommended value for LFV workflows: 10.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Output file processing</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>dibayes.write.fasta</strong> 1 Whether or not to create FASTA files. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Do not create the output FASTA files.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Create the individual and combined output FASTA files of all contigs.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>dibayes.write.consensus</strong> 1 Whether or not to create individual consensus calls files of each contig.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Do not create individual consensus calls files of each contig.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Create individual consensus calls files of each contig.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Neither setting creates a combined consensus calls file of all contigs.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>dibayes.compress.consensus</strong> 0 Whether or not to zip the generated consensus files.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Do not compress the consensus files.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Compress the individual consensus file of each contig (within the individual contig folders).</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Resource parameters</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>memory.request</strong> 7gb Memory request for process. Include the units gb in the setting.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>java.heap.space</strong> 14000 Dynamic memory requirement, in megabytes. Do not enter a unit with the numeric value.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>maximum.workers</strong> 24 Maximum number of nodes that can be used for this analysis.</td>
</tr>
</tbody>
</table>
The call stringency parameter

The `dibayes.call.stringency` parameter controls the SNP calling results. Each setting represents a different combination of filters. Table 39 explains the call stringency parameter settings. The setting `low_frequency` (not shown in the table) is for use only with low frequency variant workflows.

Table 39  Empirical `dibayes.call.stringency` settings for diBayes

<table>
<thead>
<tr>
<th>Stringency setting</th>
<th>Stringency</th>
<th>SNPs</th>
<th>False positives</th>
<th>Recommended coverage</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>highest</td>
<td></td>
<td></td>
<td></td>
<td>&gt;80x</td>
<td>Recommend when very low false positive tolerance is allowed.</td>
</tr>
<tr>
<td>high</td>
<td></td>
<td></td>
<td></td>
<td>20x – 80x</td>
<td>Requires the allele on both strands.</td>
</tr>
<tr>
<td>medium</td>
<td></td>
<td></td>
<td></td>
<td>1x – 25x</td>
<td>No both-strand requirement.</td>
</tr>
<tr>
<td>low</td>
<td></td>
<td></td>
<td></td>
<td>–</td>
<td>Very aggressive.</td>
</tr>
<tr>
<td>lowest</td>
<td></td>
<td></td>
<td></td>
<td>–</td>
<td>Very aggressive. Minimal filtering.</td>
</tr>
</tbody>
</table>

Table 40 shows the non-default filter parameter values for each level of stringency. The medium stringency level is not listed because it calls for default values of the filtering parameters. If you explicitly set one of the listed filter parameters, the value that you specify overrides the value called for by the call stringency level.

Table 40  Filter parameters affected by call stringency level

<table>
<thead>
<tr>
<th>Call stringency setting</th>
<th>Filter parameter</th>
<th>Parameter setting change</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Default value</td>
</tr>
<tr>
<td>highest</td>
<td><code>dibayes.snp.both.strands</code></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td><code>dibayes.het.min.start.pos</code></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td><code>dibayes.het.min.validreads.totalreads.ratio</code></td>
<td>0.65</td>
</tr>
<tr>
<td></td>
<td><code>dibayes.het.min.adjacentsnp.alrele.count</code></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td><code>dibayes.hom.min.nonref.start.pos</code></td>
<td>2</td>
</tr>
<tr>
<td>high</td>
<td><code>dibayes.snp.both.strands</code></td>
<td>0</td>
</tr>
<tr>
<td>low</td>
<td><code>dibayes.het.min.start.pos</code></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td><code>dibayes.hom.min.nonref.start.pos</code></td>
<td>2</td>
</tr>
<tr>
<td>lowest</td>
<td><code>dibayes.het.min.adjacentsnp.alrele.count</code></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td><code>dibayes.het.min.nonref.color.qv</code></td>
<td>7</td>
</tr>
<tr>
<td></td>
<td><code>dibayes.het.min.start.pos</code></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td><code>dibayes.het.min.validreads.totalreads.ratio</code></td>
<td>0.65</td>
</tr>
<tr>
<td></td>
<td><code>dibayes.het.min.nonref.allele.count</code></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td><code>dibayes.het.min.nonref.base.qv</code></td>
<td>26</td>
</tr>
<tr>
<td></td>
<td><code>dibayes.het.min.nonref.color.qv</code></td>
<td>7</td>
</tr>
<tr>
<td></td>
<td><code>dibayes.snps.min.nonref.base.qv</code></td>
<td>26</td>
</tr>
<tr>
<td></td>
<td><code>dibayes.reads.min.mapping.qv</code></td>
<td>8</td>
</tr>
</tbody>
</table>
The skip high coverage filter
Enable the dibayes.het.skip.high.coverage filter to skip false-positive SNP positions with high coverage. If you enable this filter, the SNP position is skipped if the coverage of a position is too high compared to the median of the coverage distribution of all positions. Enable the dibayes.het.skip.high.coverage filter for whole genome resequencing. Disable the filter for transcriptome or targeted resequencing. The filter is enabled by default.

The reads mapping QV parameter
The parameter dibayes.reads.min.mapping.qv is a read-level filter. Reads with mapping QVs lower than this value are filtered out. The empirical default of the mapping QV is 8, which gives a balanced performance between sensitivity (more SNPs to be called) and specificity (fewer false positive SNPs) on multiple known data sets.
Raising the mapping QV reduces the number of false positive SNP calls. However, because more reads are filtered out, the total number of SNP calls decreases also.

The minimum base QV filters
The dibayes.snps.min.base.qv and dibayes.snps.min.nonref.base.qv parameters work like a read filter at the position level. These parameters are very effective in removing bad base calls for SNP analysis. Increasing the value of these parameters causes only high-quality base calls to be considered, and results in higher SNP calling accuracy with a slight sacrifice of the total number of SNPs. When data coverage is high, we recommend using a higher value for higher accuracy.
Lowering the value of the dibayes.snps.min.base.qv and dibayes.snps.min.nonref.base.qv parameters results in more SNP calls.

Other filtering parameters
These other optional diBayes filters provide additional freedom for different datasets and applications. All filtering is applied before a position is considered by the module’s algorithms.
Table 44 on page 197 also describes how the optional tuning parameters are used in filtering reads.

---

Table 40 Filter parameters affected by call stringency level (continued)

<table>
<thead>
<tr>
<th>Call stringency setting</th>
<th>Filter parameter</th>
<th>Parameter setting change</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Default value</td>
</tr>
<tr>
<td>low_frequency</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>dibayes.reads.min.mapping.qv</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>dibayes.het.min.allele.ratio</td>
<td>0.15</td>
</tr>
<tr>
<td></td>
<td>dibayes.reads.min.alignlength.readlength.ratio</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>dibayes.hom.min.nonref.base.qv</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>dibayes.het.skip.high.coverage</td>
<td>1</td>
</tr>
</tbody>
</table>
SNPs internal parameters

Table 41 describes SNPs internal parameters that we do not recommend changing.

<table>
<thead>
<tr>
<th>dibayes.run</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enables the SNPs module. Allowed values:</td>
<td></td>
</tr>
<tr>
<td>• 0: Do not run a SNPs analysis.</td>
<td></td>
</tr>
<tr>
<td>• 1: Run the SNPs module during this analysis.</td>
<td></td>
</tr>
</tbody>
</table>

The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use.

SNPs output files

There are multiple output subfolders (for example, chr_1, chr_2, ..., chr_n), corresponding to individual chromosomes or chromosome/contigs. Each subfolder contains up to 4 files:

• <prefix>_SNP.gff
• <prefix>_SNP.vcf
• <prefix>_quartiles.txt
• (Optional) <prefix>_Consensus_Calls.txt
• (Optional) <prefix>_Consensus_Basespace.fasta

The remainder of this section describes these files and provides examples.

In the root of output folder, the individual GFF, VCF, and FASTA file in the chromosome subfolders are concatenated into the final GFF, VCF, and FASTA file for the whole genome. Because of the large size of individual consensus calls files, we do not consolidate them into a summarized copy to save space.

The output layout is listed below. The first two files are consolidated data for the entire run. The files in contig subfolders are data for an individual contig.

results_directory/
  <prefix>_SNP.gff3
  <prefix>_SNP.vcf
  <prefix>_Consensus_Basespace.fasta.
contig1/
  <prefix>_SNP.gff3
  <prefix>_SNP.vcf
  <prefix>_Consensus_Calls.txt
  <prefix>_Consensus_Basespace.fasta
  <prefix>_quartiles.txt
contig2/
  <prefix>_SNP.gff3
  <prefix>_SNP.vcf
  <prefix>_Consensus_Calls.txt
  <prefix>_Consensus_Basespace.fasta
  <prefix>_quartiles.txt
VCF file

A VCF file is in the Variant Call Format, which is a text format. The file contains meta-information lines, a header line, and data lines that each contain information about a position in the genome. This file format was developed as part of the 1000 Genomes project, and is described at this site:

http://vcftools.sourceforge.net/specs.html

The VCF and GFF output files contain the same contents, in a different file format.

The calculation of the genotype quality (GQ) field differs in base space and color space:

- **Base space** — GQ is calculated as the likelihood of the given genotype being correct along with a separate variant score in the QUAL field.
- **Color space** — GQ is calculated with the formula $-10 \times \log_{10}(\text{prob})$, where prob is the score field from the GFF file.

The following fields have a one-to-one correspondence in the GFF and VCF files:

<table>
<thead>
<tr>
<th>GFF field</th>
<th>VCF field</th>
</tr>
</thead>
<tbody>
<tr>
<td>rawCoverage</td>
<td>DP</td>
</tr>
<tr>
<td>coverage</td>
<td>FDP</td>
</tr>
<tr>
<td>refAlleleCounts, novelAlleleCounts</td>
<td>AD</td>
</tr>
<tr>
<td>refAlleleStarts, novelAlleleStarts</td>
<td>AST</td>
</tr>
<tr>
<td>refAlleleMeanQV, novelAlleleMeanQV</td>
<td>AMQV</td>
</tr>
</tbody>
</table>

VCF format example

The following is a truncated example of VCF file contents. Fields are described in the FORMAT lines.

```plaintext
##fileformat=VCFv4.1
##fileDate=20111011
##source="LifeScope(tm) 2.5 diBayes"
##reference=/data/shasto/Lifescope_Results/Group_1/chr1_1.fa
##contig=<ID=chr1>
##contig=<ID=chr2>
##contig=<ID=chr22>
##contig=<ID=chrX>
##INFO=<ID=DP,Number=1,Type=Integer,Description="Total Depth">  
##INFO=<ID=GT,Number=1,Type=String,Description="Genotype">  
##INFO=<ID=GQ,Number=1,Type=Float,Description="Genotype Quality">  
##INFO=<ID=DP,Number=1,Type=Integer,Description="Read Depth">  
##INFO=<ID=FDP,Number=1,Type=Integer,Description="Filtered Read Depth">  
##INFO=<ID=AD,Number=.,Type=Integer,Description="Allelic depths for the REF and ALT alleles in the order listed in the ALT field">  
##INFO=<ID=AST,Number=.,Type=Integer,Description="Allelic unique start positions for the REF and ALT alleles in the order listed in the ALT field">  
##INFO=<ID=AMQV,Number=.,Type=Integer,Description="Allelic average base qv for the REF and ALT alleles in the order listed in the ALT field">  
#CHROM POS ID REF ALT QUAL FILTER INFO FORMAT Unknown
chr1 909768 . A G . PASS DP=51 GT:GQ:DP:FDP:AD:AST:AMQV 1/1:100:51:23:0,23:0,12,0,38
```
**GFF file format**

The `<prefix>_SNP.gff` file is the list of output SNPs. This file is described in Table 42.

### Table 42  `<diBayes.output.prefix>.gff` file format description

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>##</td>
<td>Header comment lines.</td>
<td>Input files and algorithm parameters</td>
</tr>
<tr>
<td>#</td>
<td>Header of the results.</td>
<td>—</td>
</tr>
<tr>
<td>seqid</td>
<td>The string ID of the sequence to which the start and end coordinates refer.</td>
<td>chr1</td>
</tr>
<tr>
<td>source</td>
<td>The source of the data.</td>
<td>SOLiD_diBayes</td>
</tr>
<tr>
<td>type</td>
<td>Sequence ontology derived type for this variation. For diBayes, this is always SNP.</td>
<td>SNP</td>
</tr>
<tr>
<td>start</td>
<td>Start position of the SNP.</td>
<td>420</td>
</tr>
<tr>
<td>end</td>
<td>End position of the SNP.</td>
<td>420</td>
</tr>
<tr>
<td>score</td>
<td>Calculated p-value of the SNP.</td>
<td>0.000000</td>
</tr>
<tr>
<td>strand</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>phase</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

**Attributes:**

- **reference**: The base of the reference sequence at the current position. reference=c
- **allele-call**: Genotype in alleleA/alleleB (in alphabet order) format for bases observed in all valid reads that cover the position. allele-call=A/C
- **coverage**: The number of valid reads that cover the current position. coverage=52
- **rawCoverage**: The number of raw reads that cover the current position. rawCoverage=52
- **refAlleleCounts**: The number of reads of the reference allele at the current position. refAlleleCounts=22
- **refAlleleStarts**: The number of different start positions of reads having the reference allele at the current position. refAlleleStarts=15
- **refAlleleMeanQV**: The mean of quality values of all reference allele reads at the current position. refAlleleMeanQV=15
- **novelAlleleCounts**: The number of reads of the most abundant non-reference allele at the current position. novelAlleleCounts=24
- **novelAlleleStarts**: The number of different start positions of reads having the most abundant non-reference allele at the current position. novelAlleleStarts=14
- **novelAlleleMeanQV**: The mean of quality values of all novel allele reads. novelAlleleMeanQV=17
- **diColor1**: The most abundant allele in the reads (not necessarily the reference allele) in dicolor encoding (for example, one of 00, 01, ..., 32, 33 of 16 possible dicolors). Does not apply to the results of base-space BAM files. diColor1=00
### GFF file example

The following is example content of the GFF file generated by the SNPs module.

```plaintext
##gff-version 3
##source-version LifeScope(tm) 2.0 diBayes
##feature-ontology http://song.cvs.sourceforge.net/*checkout*/
song/ontology/sofa.obo?revision=1.141
##type DNA
##date 2011-03-15
##time 15:56:41
##genome-build /local/xux1/validataion/HuRef/hg18/reference/
chr22.validated.fasta
##input-files /local/xux1/validataion/HuRef/hg18/input/
solid0107_20100514_PE_launch_verification_F3.sowmi-1_0309.bam
##options call-stringency: medium; diBayes-base-space: 0;
experiment-name: Test_huref_4.0_base_newbamo309_lof3_color2.0_newout_chr22; het-
skip-high-coverage: 1; index: 1; log-dir: /local/xux1/
validataion/HuRef/hg18/log/newbamo309_lof3_color2.0; output-
dir: /local/xux1/validataion/HuRef/hg18/outputs/
```

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>diColor2</td>
<td>The second-most abundant allele in the reads. Does not apply to the results of base-space BAM files.</td>
<td>diColor2=22</td>
</tr>
<tr>
<td>zygosity</td>
<td>Heterozygosity flag. Values are: homozgyous: Homozygous SNP heterozygous: Heterozygous SNP</td>
<td>zygosity=heterozygous</td>
</tr>
<tr>
<td>flag</td>
<td>Reports why the SNP is not called as a homozygous SNP or as a heterozygous SNP, when information is available. Values are: m*: Flags beginning with the letter “m” report why the SNP is not called as a Hom. t*: Flags beginning with the letter “t” report why the SNP is not called as a Het. —: No information. The SNP can be either a Het or a Hom.</td>
<td>flag=t4,t10</td>
</tr>
</tbody>
</table>

See also the consensus calls Flag field, which reports the reason why a position is not called a SNP, and adds which filter causes this result.

Table 42 <diBayes.output.prefix>.gff file format description (continued)
newbam0309_lcf3_color2.0; poly-rate: 0.003; reference: /local/xux1/validataion/HuRef/hg18/reference/chr22.validated.fasta; working-dir: /local/xux1/validataion/HuRef/hg18/tmp/newbam0309_lcf3_color2.0;
#ChrSourceTypePos_StartPos_EndScoreStrandPhaseAttributes
chr22SOLiD_diBayesSNP14432513144325130.000984..
  reference=G;allele-call=C/G;coverage=6;rawCoverage=9;refAlleleCounts=3;refAlleleStarts=3;
  refAlleleMeanQV=22;novelAlleleCounts=3;novelAlleleStarts=3;novelAlleleMeanQV=29;diColor1=01;diColor2=32;zygosity=heterozygous;
  flag=-
chr22SOLiD_diBayesSNP14433730144337300.002050..
  reference=C;allele-call=A/C;coverage=8;rawCoverage=8;refAlleleCounts=5;refAlleleStarts=4;
  refAlleleMeanQV=20;novelAlleleCounts=3;novelAlleleStarts=3;novelAlleleMeanQV=26;diColor1=20;diColor2=31;zygosity=heterozygous;
  flag=-##

Combined GFF for all contigs

The combined GFF file is a consolidated version of individual GFF files. This file is named according to the following pattern:

${task.output.dir}/${analysis.name}_SNP.gff3

The following is an example of the summarized header of the combined GFF file:

##gff-version 3
##List of SNPs. Date Thu Oct  7 14:32:53 2010 Stringency: medium Mate Pair: 0 Read Length: 50 Polymorphism Rate: 0.001000 Bayes Coverage: 60 Bayes_Single_SNPs: 1 Filter_Single_SNPs: 1 Quick_P_Threshold: 0.997000 Bayes_P_Threshold: 0.040000 Minimum_Allele_Ratio: 0.150000 Minimum_Allele_Ratio_Multiple_of_Dicolor_Error: 100
##Data source:    Experiment: test_case0003
##chr1  1// chr${index} chromosome name
##chr2  2// chr${index} chromosome name
##chr3  3// chr${index} chromosome name

The portion in bold is the chromosome index and name table, which combine the information from individual GFF files.

Consensus_Calls file format

The <prefix>_Consensus_Calls.txt is a tab-separated file that covers all positions that have coverage and provides general information about each position (see Table 43). Its flag column shows a list of codes of filters (see Table 44 on page 197) that the position fails to pass to be called as a SNP. It is very useful for identifying the reason of false negative SNP calls (the known SNPs that are not called by diBayes). By adjusting the value of keys listed in Table 44, user can loosen the filter stringency and make the positions that originally fail the filters to be considered in the SNP analysis.

Consensus calls files are created with file names according to the following pattern:
Table 43  `<diBayes.output.prefix>_Consensus_Calls.txt` file format description

<table>
<thead>
<tr>
<th>File Name/Column</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chr</td>
<td>Chromosome/Contig number.</td>
<td>chr1</td>
</tr>
<tr>
<td>Position</td>
<td>Location of the SNP on the reference sequence.</td>
<td>442</td>
</tr>
<tr>
<td>Allele_DiColor1</td>
<td>The most abundant allele in the reads (not necessarily the reference allele) in dicolor encoding (for example 00, 01,...32, 33) of 16 possible dicolors. Color space only.</td>
<td>03</td>
</tr>
<tr>
<td>Allele_DiColor2</td>
<td>The second most abundant allele in the reads in dicolor encoding (for example 00, 01,...32, 33) of 16 possible dicolors. Color space only.</td>
<td>03</td>
</tr>
<tr>
<td>Reference</td>
<td>The base of the reference sequence at the current position.</td>
<td>C</td>
</tr>
<tr>
<td>Genotype</td>
<td>Genotype in the form of IUB codes for bases observed in all the reads.</td>
<td>C</td>
</tr>
<tr>
<td>P-value</td>
<td>Calculated p-value of the SNP. [P-values are generated from the Bayesian and Frequentist algorithms.]</td>
<td>1.00000</td>
</tr>
<tr>
<td>Coverage</td>
<td>The number of the reads that cover the current position.</td>
<td>2</td>
</tr>
<tr>
<td>Unfiltered Coverage</td>
<td>The number of all unfiltered reads that cover the current position.</td>
<td>2</td>
</tr>
<tr>
<td>nCounts_1st_allele</td>
<td>The number of the most abundant allele at the current position.</td>
<td>2</td>
</tr>
<tr>
<td>nCounts_Reference_allele</td>
<td>The number of reads having the reference allele at the current position.</td>
<td>2</td>
</tr>
<tr>
<td>nCounts_NonReference_allele</td>
<td>The number of reads having the most abundant non-reference allele reads at the current position.</td>
<td>0</td>
</tr>
<tr>
<td>Ref-Avg-QV</td>
<td>The mean of quality values of all reference allele reads at the current position.</td>
<td>29</td>
</tr>
<tr>
<td>Novel-Avg-QV</td>
<td>The mean of quality values of all novel allele reads.</td>
<td>0</td>
</tr>
<tr>
<td>Heterozygous</td>
<td>Heterozygosity flag. Values are:</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0: Homozygous SNP.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Heterozygous SNP.</td>
<td></td>
</tr>
<tr>
<td>Algorithm</td>
<td>The algorithm used to call the current SNP. Values are:</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>• –1: Not a SNP.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 0: Homozygous position.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 1: Heterozygous SNP, called by the Bayesian algorithm.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 2: Heterozygous SNP, called by the Frequentist algorithm.</td>
<td></td>
</tr>
<tr>
<td>Flag</td>
<td>Flag indicating why a location is not called a SNP.</td>
<td>m4</td>
</tr>
</tbody>
</table>

The flags in Table 44 explain why a read is filtered out.

Table 44  `<diBayes.output.prefix>_Consensus_Calls.txt` flag column description

<table>
<thead>
<tr>
<th>Flag</th>
<th>Filter meaning</th>
<th>Related parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heterozygote</td>
<td>Insufficient coverage for a heterozygous position.</td>
<td>dibayes.het.min.coverage.</td>
</tr>
</tbody>
</table>
### Table 44 <diBayes.output.prefix>_Consensus_Calls.txt flag column description (continued)

<table>
<thead>
<tr>
<th>Flag</th>
<th>Filter meaning</th>
<th>Related parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>t2</td>
<td>Insufficient unique start positions for a heterozygous position.</td>
<td>dibayes.het.min.start.pos</td>
</tr>
<tr>
<td>t3</td>
<td>The coverage is too high, compared to the coverage distribution of all positions.</td>
<td>dibayes.het.skip.high.coverage</td>
</tr>
<tr>
<td>t4</td>
<td>The fraction of the second-most common valid color (or base allele) in the total of top two valid colors (or the top two base alleles) is higher than the threshold (usually a function of raw error squared).</td>
<td>dibayes.het.min.allele.ratio</td>
</tr>
<tr>
<td>t5</td>
<td>Genome positions with sufficient coverage (20x) at which there is only 1 unique read position for all the reads. It could be a PCR error.</td>
<td>—</td>
</tr>
<tr>
<td>t6</td>
<td>The candidate SNP is evenly distributed over positions (not used).</td>
<td>—</td>
</tr>
<tr>
<td>t7</td>
<td>The non-reference allele is not on both strands (counting the reads).</td>
<td>dibayes.snp.both.strands</td>
</tr>
<tr>
<td>t8</td>
<td>Both alleles not evenly represented on both strands (doing statistical test on read distribution of the both strands).</td>
<td>dibayes.snp.both.strands</td>
</tr>
<tr>
<td>t9</td>
<td>The second-most common valid dicolor is not more frequent than the third-most. Color space only.</td>
<td>—</td>
</tr>
<tr>
<td>t10</td>
<td>There are no other valid SNPs, or the second-most common valid SNP (as a proportion of all valid SNPs) is less than half the 2-dibase error frequency. Color space only.</td>
<td>dibayes.het.min.allele.ratio</td>
</tr>
<tr>
<td>t11</td>
<td>Sum of the first and second-most common alleles must be at least this proportion, for example, 0.5, of all reads at this position.</td>
<td>dibayes.het.min.ratio.validreads.totalreads.ratio</td>
</tr>
<tr>
<td>t12</td>
<td>The dicolor alleles are inconsistent to the reference allele. Color space only.</td>
<td>—</td>
</tr>
<tr>
<td>t13</td>
<td>The difference of quality values of the non-reference allele and the reference allele is much lower the quality value of the reference allele.</td>
<td>—</td>
</tr>
<tr>
<td>t14</td>
<td>The average of p-values of color/base quality values of the non-reference allele is lower than one standard deviation from the mean of p-value distribution, when the non-reference allele has to be at low frequency (rare variant).</td>
<td>—</td>
</tr>
<tr>
<td>t15</td>
<td>The average of p-values of color/base quality values of the position is lower than one standard deviation from the mean of the p-value distribution.</td>
<td>—</td>
</tr>
<tr>
<td>t16</td>
<td>Insufficient coverage of the reference allele.</td>
<td>—</td>
</tr>
<tr>
<td>t17</td>
<td>Insufficient coverage of the non-reference allele.</td>
<td>—</td>
</tr>
<tr>
<td>t18</td>
<td>Zero coverage.</td>
<td>—</td>
</tr>
<tr>
<td>t19</td>
<td>Insufficient number of start positions of non-reference allele.</td>
<td>—</td>
</tr>
<tr>
<td>t20</td>
<td>Insufficient coverage for either of two alleles, when neither allele is same as the reference.</td>
<td>—</td>
</tr>
<tr>
<td>t21</td>
<td>Quality value of non-reference allele too low (lower than a relative threshold depend on the distribution of all color quality values).</td>
<td>—</td>
</tr>
<tr>
<td>t22</td>
<td>Quality value of non-reference allele too low (lower than an absolute threshold).</td>
<td>dibayes.het.min.nonref.color.qv</td>
</tr>
<tr>
<td>t23</td>
<td>The less common allele is not on both strands.</td>
<td>dibayes.het.lca.both.strands</td>
</tr>
<tr>
<td>t24</td>
<td>Insufficient number of start positions of the less common allele.</td>
<td>dibayes.het.min.lca.start.pos</td>
</tr>
</tbody>
</table>
Table 44 <diBayes.output.prefix>_Consensus_Calls.txt flag column description (continued)

<table>
<thead>
<tr>
<th>Flag</th>
<th>Filter meaning</th>
<th>Related parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>t25</td>
<td>Quality value of the less common allele too low (lower than an absolute color quality value).</td>
<td>dibayes.het.min.lca.color.qv</td>
</tr>
<tr>
<td>t26</td>
<td>The color QV difference between the most common and less common allele is larger than the user defined threshold.</td>
<td>dibayes.het.max.diff.color.qv</td>
</tr>
<tr>
<td>t27</td>
<td>The ratio of filtered reads and raw reads at a position is lower than a user defined threshold.</td>
<td>dibayes.snps.min.filteredreads.rawreads.ratio</td>
</tr>
</tbody>
</table>

**Homozygote**

| m1    | Insufficient coverage for a homozygous SNP.                                     | dibayes.hom.min.coverage                 |
| m2    | Insufficient proportion of the count of the most common valid dicolor (base) allele to all valid reads. |                                          |
| m3    | Non-reference allele not on both strands.                                        | dibayes.snp.both.strands                |
| m4    | Reserved for future filters [not used].                                           |                                          |
| m5    | Second-most common allele too close in coverage to the first most common allele.  |                                          |
| m6    | Insufficient coverage (as a fraction of the median of the coverage distribution of all positions) for a homozygous call. |                                          |
| m7    | Dicolor inconsistent with reference. Color space only.                             |                                          |
| m8    | Insufficient number of non-reference alleles                                      | dibayes.hom.min.nonref.allele.count      |
| m9    | Same as the m2 flag.                                                              |                                          |
| m10   | Insufficient number of start positions of non-reference alleles.                  | dibayes.hom.min.nonref.start.pos         |
| m11   | No coverage.                                                                     |                                          |
| m12   | Quality value of non-reference allele too low (lower than an absolute threshold). |                                          |
| m13   | Quality value of non-reference allele too low (lower than a relative threshold that depends on the distribution of all color quality values). |                                          |

**Consensus calls file example**

The following is example output of a SNPs consensus calls file.

```bash
##consensus-call
##source-version LifeScope(TM) 2.0 diBayes
##type DNA
##date 2011-03-10
##time 11:38:51
##genome-build /local/xux1/validataion/HuRef/hg18/reference/chr1.validated.fasta
##input-files /local/xux1/validataion/HuRef/hg18/input/solid0107 20100514_PE_launch_verification_P3.sowmi-1_0309.bam
##options call-stringency: medium; dibayes-base-space: 0;
experiment-name: Test_huref_4.0 base_newbam0309_lof3_color2.0_chr1; het-skip-high-coverage: 1; index: 1; log-dir: /local/xux1/validataion/HuRef/hg18/log/newbam0309_lof3_color2.0; output-dir: /local/
```

LifeScope™ Genomic Analysis Software Command Shell User Guide
Chapter 10  Run a SNPs Analysis

SNPs output files

```
xuxl/validataion/HuRef/hg18/outputs/newbam0309_lof3_color2.0;
poly-rate: 0.003; reference: /local/xuxl/validataion/HuRef/
hg18/reference/chrl.validated.fasta; working-dir: /local/xuxl/
validataion/HuRef/Hg18/tmp/newbam0309_lof3_color2.0;
#Chr  Position  Allele_DiColor1 Allele_DiColor2 Reference
Genotype    P-value Coverage Unfiltered_Coverage
nCounts_1st_allele nCounts_Reference_allele
nCounts_NonReference_allele Ref_Avg_QV NonRef_Avg_QV
Zygousity Algorithm Flag
chr11  50262   10     10     T    T    1.000000   1
1      1      1      0      25      0   homozygous   -
1      t4,t10,t1,t2,t21,t22,m4
chr11  50263   00     00     T    T    1.000000   1
1      1      1      0      25      0   homozygous   -
1      t4,t10,t1,t2,t21,t22,m4
```
SNPs algorithm description

LifeScope™ Software uses the diBayes package to find Single Nucleotide Polymorphisms (SNPs). The diBayes package performs independent SNP analysis at each position in the reference, using either a Bayesian or Frequentist algorithm (see Figure 9 on page 202 and Figure 10 on page 203).

Frequentist algorithm

The Frequentist algorithm is used when coverage is high for a given position, for example, 60x. Given the assumption that the errors follow a Poisson distribution, the Frequentist algorithm calculates the probability of a null hypothesis that the observed valid dicolor base mismatches are errors. If the probability of the null hypothesis is too low, the hypothesis is rejected and the position is called a SNP.

Bayesian algorithm

The SNP finding problem is to predict the most likely genotype (G) at a genome position from a list of reads (R) that are mapped to that position. To do that, for each genotype type G, we want to estimate the posterior probability of P(G|R). Using Bayesian theorem, P(G|R)=P(R|G)P(G)/P(R), where P(G) is the prior probability of the genotype G, P(R|G) is the posterior probability of observing the given read set at the position assuming the true genome type is G, and P(R) is the prior probability of the read sets. P(R) is a constant for a given set of R and can be ignored from further consideration. To estimate P(R|G), the algorithm takes into consideration the color (or base) quality values, position error profiles, and probe error profiles. (The Frequentist algorithm also uses these information.)

Define S(G)=P(R|G)P(G) and \( T = \sum_G S(G) \), the likelihood of each genotype, L(G), can be estimated by S(G)/T. To simplify the computation, our Bayesian algorithm compares two hypotheses; the first, that the true genotype is homozygous for the most common allele \( (G_0) \) in the reference, and the second, that the true genotype is heterozygous (SNP) for the top two alleles \( (G_a) \) in the reference. Therefore, \( T \) can be simplified to \( S(G_0)+S(G_a) \), and \( L(G_0)=S(G_0)/T \), and \( L(G_a)=S(G_a)/T \). The algorithm reports the genotype with the highest likelihood.

75 percentile of the coverage distribution (the third quartile): 23
0.1 percentile of the average color quality value per position distribution: 4
0.2 percentile of the average color quality value per position distribution: 4
1 percentile of the average color quality value per position distribution: 5
50 percentile of the average color quality value per position distribution (median): 19
Data flow

Figure 9 shows how diBayes can take multiple BAM files that are sorted by genome positions as input to make SNP calls at positions with coverage. Running diBayes requires the position error and probe error files of each BAM file. The position error files and probe error files are generated by the mapping statistics module. In a multi-CPU computing cluster, diBayes automatically distributes parallel jobs for individual chromosomes to different computing nodes. The final results for the whole genome are merged after the individual jobs are complete.

Figure 9  diBayes data flow

Internal module flow

The SNPs module first validates the input BAM files, then determines whether to use its internal color-space or base-space processing.

BAM file validation

The SNPs module requires the following:

• Each BAM file must have only one read group and only one library type.
• Each BAM file is sorted by genomic coordinates.
• The reference sequence lengths in BAM file header and in reference FASTA file are identical.
• The BX and BY fields in the BAM headers of all input BAM files must be consistent. (The BX and BY fields record whether the original XSQ reads file had base space present on tag 1 and tag 2, respectively.)

If BAM files that do not meet these criteria are included in the analysis, the analysis fails.

**Color-space and base-space processing**

The SNPs module uses its color-space processing algorithm if the following are both true:

• For paired data, BX and BY are both 0, or, for fragment data, BX is 0.
• Color space information is present in the BAM data (CS and CQ fields are present in the first 10 read records).

For other conditions, the SNPs module uses its base-space processing algorithm. (The exception is BAM files with inconsistent BX and BY fields. These BAM files are rejected previously during BAM file validation.)

The type of processing selected is important because certain filtering and other parameters are only effective on either base space or color space, but not both. The parameters that only apply to one type of processing are noted in the parameters table, Table 38 on page 185.

Both types of processing follow the steps shown in Figure 10.

\[\text{Figure 10  diBayes algorithm flowchart}\]
FAQ – SNPs

1

SNP calling is taking too long. What can I do?

Parallelization is the best solution for running diBayes on large data sets. LifeScope™ Software has implemented the parallel distribution of diBayes jobs for individual chromosomes. It can help the users to achieve the best running efficiency. The diBayes uses BAM files as its input and removes all large temporary files (improving run times).

2

I seem to be missing SNPs — how do I troubleshoot?

First, try repeating the analysis with a lower stringency level (for example, changing dibayes.call.stringency from medium to low). Look at the positions that you expect to be SNPs in the consensus calls file. Typically, you should find a list of flags that describe the reasons the position was not called as a SNP (see Table 42 on page 194). Look at the properties of this position. Visualizing the reads might be helpful here.

Is the coverage much higher than average? The filter dibayes.het.skip.high.coverage can remove heterozygous SNPs at positions of extremely high coverage because these have previously been observed to be variants in repeat regions rather than truly heterozygous SNPs at a single position.

Are the reads strongly biased towards one strand, or is the non-reference allele missing from one strand? You probably want to perform a run with dibayes.call.stringency=medium, or switch off the “both strands” requirement (dibayes.snp.both.strand=0).

Do all the reads have the same start position because of the way the sample was prepared? You need to reduce the number of unique start positions required to call a SNP (dibayes.het.min.start.pos and dibayes.hom.min.start.pos).

3

I seem to be finding too many SNPs — how do I troubleshoot?

Look at the properties of the SNPs and compare them to the properties of all the positions in the consensus_calls.txt file. Is the coverage of these SNPs much lower or much higher than average? Is the color quality value of the non-reference allele much lower than average? You might want to post-filter the results if you find these kinds of patterns. For example, you might want to remove SNPs with very low coverage, or very low color quality values, or p-values close to one. You might want to repeat the analysis with a more stringent setting for example, changing the parameter dibayes.call.stringency from medium to high.

Filtering by p-values is not recommended for homozygous SNPs. See also FAQ 4, “How do I interpret the SNPs p-values?” on page 205.
How do I interpret the SNPs p-values?

P-values close to 0 mean that the algorithm has strong confidence that the genotype call is correct. P-values close to 1 mean that the algorithm does not have very strong confidence that the genotype call is correct.

Note that the p-value refers to the probability that the genotype call is correct. This meaning is not the same as the probability as “there is a SNP at this position”. A homozygote SNP can have a p-value close to 1, but it is still highly likely that a SNP is present at this location.

Most commonly, a p-value close to 1 occurs when there is good evidence that a SNP is present, but it is difficult to confidently decide whether the genotype is heterozygote or homozygote. For example, take a case where there is coverage=3 and all of these are high QV non-reference allele calls (e.g., the reference is A and there are three C’s observed). In this case, it is clear there is a SNP in this location. However, it is possible this might really be a heterozygous SNP, and if we observed a fourth allele, it might be an A. So the p-value on this homozygous SNP call might be close to 1, since we are not very confident of the genotype call, even though we are confident that there is a SNP in this location. In this case, we are still confident that a SNP is present at this location.

For this reason, it is recommended not to filter out homozygous SNPs with high p-value, as this will result in removing many real SNPs.

Can the SNPs module find two adjacent SNPs?

Yes, in some circumstances. You must set the `dibayes.detect.2.adjacent.snps` parameter to 1. With this setting, diBayes detects and properly calls these scenarios:

- Two adjacent heterozygous SNPs
- Two adjacent homozygous SNPs
- A homozygous SNP adjacent to a heterozygous SNPs (base-space algorithm only)
- A heterozygous SNP adjacent to a homozygous SNPs (base-space algorithm only)

However, diBayes does not call a homozygous SNP adjacent to a heterozygous SNP in color-space BAM files (even with `dibayes.detect.2.adjacent.snps` on). In general, adjacent SNPs calls may have more false positives than single SNPs calls do.

Why does SNPs output sometimes contain a blank dicolor value?

Occasionally, the color sequence is not reported and the base sequence is reported as an ‘N’, before and after a SNP. This reporting usually occurs when the non-reference allele is more abundant than the reference allele. In these cases, the bases flanking the SNP are the normal reference sequence. (The color sequence appears as a space character in the output file. The space character may affect pattern-matching tools.)
How can I control the sensitivity and specificity of SNP calling?

Different call stringency settings and filter settings may help users to achieve different sensitivity and specificity requirements. The more stringent the filters are, the less sensitivity the SNP call and the higher specificity in general. Two filters that are very flexible for users’ different needs are the `dibayes.reads.min.mapping.qv` (MQV) and `dibayes.reads.min.alignlength.readlength.ratio` (ARR) parameters. Changing one or both of these parameters filters out low confident reads and gets SNP calls with high accuracy. The following figures show that changing MQV and ARR may affect the total number of SNP calls and the dbSNP concordance of the predictions.

Figure 11 Total SNP calls on chromosome 1 of a HuRef long mate-pair data as a function of mapping QV cutoffs and ratios of alignment length and read length (ARR)

Figure 12 dbSNP concordance of SNP calls on chromosome 1 of a HuRef long mate-pair data as a function of mapping QV cutoffs and ratios of alignment length and read length (ARR)
Why are the coverages reported by diBayes for individual genomic positions sometimes different from the coverage derived from input BAM files?

The difference happens because diBayes reports the coverage of filtered reads instead of raw coverages derived from input BAM files. The diBayes module calls SNPs only for these reads that are:

- Based on reads that are primary alignments.
- Are classified as an AAA pair (See “FAQ – Pairing” on page 176 for a description of the 3-letter genomic code classifications).
- Are not duplicates.
- Do not contain indel.
- Have high mapping quality values (default is 8).
- Pass other read level filters (such as optional tuning parameters such as \texttt{dibayes.reads.max.mismatch.alignlength.ratio} and others described in Table 38 on page 185).

The SNPs module reports both coverage and raw coverage at every position, for users' convenience.

Users can find detailed information about the numbers of reads filtered by read level filters and the total number of raw reads at the end of each algorithm log file.

Why is a SNPs analysis not recommended with whole transcriptome data?

The reverse transcription process that is part of the WT library prep intrinsically adds a percentage of sequencing errors. In addition, alternative splicing and other alignment complexities around exon junctions may cause local misalignments, which can lead to false positive SNP calls. Although the SNPs module can run successfully on BAM files generated from the WTA module, using those BAM files may cause a higher number of false positive SNPs.

SNPs reports heterozygous and homozygous zygosity for variants that occur in a single copy.

Although the VCF file reports variants as either “heterozygous” or “homozygous”, variants on chromosomes that occur in a single copy (for example, the X and Y chromosome in human males) are in fact hemizygous.

The following are examples from VCF output:
chrY SOLiD_diBayes SNP 28680488 28680488
0.068042 . . reference=G; allele-call=CG; coverage=13; rawCoverage=21; refAlleleCounts=10; refAlleleStarts=9; refAlleleMeanQV=26; novelAlleleCounts=3; novelAlleleStarts=3; novelAlleleMeanQV=26; diColor1=10; diColor2=23; zygosity=heterozygous; flag=-

chrY SOLiD_diBayes SNP 28680490 28680490
0.000000 . . reference=C; allele-call=T; coverage=13; rawCoverage=21; refAlleleCounts=1; refAlleleStarts=1; refAlleleMeanQV=31; novelAlleleCounts=12; novelAlleleStarts=11; novelAlleleMeanQV=28; diColor1=12; diColor2=12; zygosity=homozygous; flag=t4,t10
Run a Human CNVs Analysis

This chapter covers:

- Overview ................................................................. 209
- Examples of running a CNV analysis ............................... 210
- CNV module parameter descriptions .............................. 211
- Human CNVs results file format description .................... 214
- Human CNVs results file examples ................................ 216
- Algorithm for the Human CNV module .......................... 217
- FAQ – Human CNVs .................................................... 220

Overview

The LifeScope™ Software Copy Number Variation (Human CNV) module detects copy number variations in a data sample that is mapped to the reference genome. The analysis method is based on depth of coverage. The module performs these phases in the analysis:

- Samples the genome sequence into non-overlapping windows
- Identifies the regions with significantly higher or lower coverage
- Assigns copy number calls to the identified regions, based on the scale of coverage

The CNV module requires predicted mappability files, which are provide with the LifeScope™ Software reference repository. These pre-computed files are available for the human reference sequences hg18 and hg19. The module currently supports only human CNV detection. By default, the module does not call CNVs within 1 MBase of the chromosomes’ centromeres and telomeres. These regions are highly repetitive and tend to contain many apparent CNVs that are not of general interest. The size of the blackout regions is a user-configurable parameter, cnv.trim.distance. The module is designed to detect larger CNVs (which can take up whole chromosomes or large portions of chromosomes) as well as smaller CNVs, by doing either global or local normalization, respectively. By default, and if the number of valid chromosome arms is greater than 6, the module does global normalization. The parameter cnv.local.normalization is provided to switch to local normalization.

The major components of the CNV module include the following:

- Coverage calculation
Examples of running a CNV analysis

In a standard workflow

The following standard workflows provide examples of running the CNV module:
- genomic.resequencing.frag
- genomic.resequencing.lmp
- genomic.resequencing.pe

The following are example shell commands using reads and references from the LifeScope™ Software repository.

```
# log into the shell
lscope.sh shell -u username -w password
# cd to the projects repository
cd /projects
# create a project and open it
cd /projects
mk proj1
cd proj1
# create an analysis and open it
mk run1
cd run1
# define the workflow
set workflow genomic.resequencing.pe
# define the input (dummy file names are used in the example)
add xsq run0209a_50_PE.xsq
add xsq run0209b_50_PE.xsq
# define the reference
set reference hg19
# optionally change CNV defaults here
ls
# start the run
run
# list progress information about the run
ls
```

See for Chapter 7, “Run a Standard Workflow Analysis” on page 103 for more information on running standard workflows. Chapter 6, “Run a Command Shell Analysis” on page 75 for information on shell commands and syntax.

As a demo analysis

The optional examples download includes an example of how to run the CNV module by itself, as a single analysis that is not part of a standard workflow. The demo example is not recommended for general use.

See Appendix H, “Demo Analyses” on page 567 for information on running the CNV module as a standalone analysis.
Chapter 11  Run a Human CNVs Analysis

CNV module parameter descriptions

This section describes the parameters that control the runtime behavior of the CNV module. See also Table 46 on page 213, which lists the effect of the cnv.stringency.setting value on other stringency parameters.

In order to change a parameter value in your analysis, use the set param shell command to replace the line # optionally change CNV defaults here in the example commands in “In a standard workflow”. For instance, to change the call stringency, use this shell command:

```shell
set cnv.stringency.setting High tertiary/cnv.ini
```

Table 45  CNV module parameter descriptions

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CNV parameters</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cnv.window.size</td>
<td>5000</td>
<td>Size of the window block to be considered as a region.  Allowed values: Integers &gt;= 100.</td>
</tr>
<tr>
<td>cnv.trim.distance</td>
<td>1000</td>
<td>Distance in kilo bases to be trimmed from the extreme ends of the chromosome arms.  Allowed values: Integers 0–100000.</td>
</tr>
<tr>
<td>cnv.min.quality</td>
<td>2</td>
<td>Minimum mappability quality value of the alignments.  Allowed values: Integers 0–100.</td>
</tr>
<tr>
<td>cnv.ploidy</td>
<td>2</td>
<td>General ploidy of the genome.  Allowed values: Integers &gt;= 1.</td>
</tr>
<tr>
<td>cnv.ploidy.exception</td>
<td>None</td>
<td>List of all the contigs whose ploidy is different to the general ploidy of the genome.  Entries in the list are in the format (contig id: ploidy of the contig), and are separated by commas. Use the string “None” to indicate no entries.</td>
</tr>
</tbody>
</table>
| cnv.local.normalization| 0             | Whether or not genome-wide normalization or local normalization should be performed.  Allowed values:  
|                        |               | 0: Perform genome-wide normalization.  1: Perform chromosome-arm local normalization. |
| cnv.write.coverage     | 0             | Whether or not to create coverage output files.  Allowed values:  
|                        |               | 0: Do not create coverage output files.  1: Create coverage output files, in WIG format |
| cnv.coverage.wsize     | 1000          | Size of the window block to be considered as a region for writing coverage output.  The mean coverage of all bases in each of these windows is output.  Allowed values: Integers 0–100000. |
| cnv.gender             | Male          | Set the ploidy of sex chromosomes in Human correctly.  Allowed values: Male, Female. |

**CNV stringency parameters**
## CNV module parameter descriptions

### cnv.stringency.setting

**Default value:** Medium

Set the stringency setting for calling CNVs. Allowed values:
- **High**: Recommend when a very low false positive tolerance is allowed.
- **Medium**: Default values.
- **Low**: Aggressive CNV calling.

Lower settings results in more CNV calls, but with more false positives. Higher settings result in fewer CNV calls, but with fewer false positives.

See also [Table 46 on page 213](#).

**Note:** For GUI users, changing the stringency setting resets the values of certain other filtering parameters (described in [Table 46](#)). On the command line, the other parameters are not reset automatically. You must set each parameter.

### cnv.deletions.min.mappability

**Default value:** 50

Minimum mappability percentage for regions to be shown as copy number deletions. Allowed values: Floats 0.0–100.0.

If this parameter is not specified, and
- cnv.stringency.setting is set to **High**, then this parameter is set to 75.
- cnv.stringency.setting is set to **Low**, then this parameter is set to 10.

### cnv.insertions.min.mappability

**Default value:** 10

Minimum mappability percentage for the regions to be shown as copy number insertions. Allowed values: Floats 0.0–100.0.

If this parameter is not specified, and
- cnv.stringency.setting is set to **High**, then this parameter is set to 25.
- cnv.stringency.setting is set to **Low**, then this parameter is set to 0.

### cnv.deletions.min.windows

**Default value:** 2

Minimum number of windows for the regions to be shown as copy number deletions. Allowed values: Integers >= 0.

If this parameter is not specified, and
- cnv.stringency.setting is set to **High**, then this parameter is set to 4.
- cnv.stringency.setting is set to **Low**, then this parameter is set to 1.

### cnv.insertions.min.windows

**Default value:** 2

Minimum number of windows for the regions to be shown as copy number insertions. Allowed values: Integers >= 0.

If this parameter is not specified, and
- cnv.stringency.setting is set to **High**, then this parameter is set to 4.
- cnv.stringency.setting is set to **Low**, then this parameter is set to 1.

### cnv.deletions.max.pval

**Default value:** 1.0

Maximum p-value for regions to be shown as copy number deletions. Allowed values: Floats 0.0–1.0.

If this parameter is not specified, and cnv.stringency.setting is set to **High**, then this parameter is set to 0.1.

### cnv.insertions.max.pval

**Default value:** 1.0

Maximum p-value for regions to be shown as copy number insertions. Allowed values: Floats 0.0–1.0.

If this parameter is not specified, and cnv.stringency.setting is set to **High**, then this parameter is set to 0.1.

---

### Table 45 CNV module parameter descriptions (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cnv.deletions.min.mappability</td>
<td>50</td>
<td>Minimum mappability percentage for regions to be shown as copy number deletions. Allowed values: Floats 0.0–100.0. If this parameter is not specified, and cnv.stringency.setting is set to <strong>High</strong>, then this parameter is set to 75. cnv.stringency.setting is set to <strong>Low</strong>, then this parameter is set to 10.</td>
</tr>
<tr>
<td>cnv.insertions.min.mappability</td>
<td>10</td>
<td>Minimum mappability percentage for the regions to be shown as copy number insertions. Allowed values: Floats 0.0–100.0. If this parameter is not specified, and cnv.stringency.setting is set to <strong>High</strong>, then this parameter is set to 25. cnv.stringency.setting is set to <strong>Low</strong>, then this parameter is set to 0.</td>
</tr>
<tr>
<td>cnv.deletions.min.windows</td>
<td>2</td>
<td>Minimum number of windows for the regions to be shown as copy number deletions. Allowed values: Integers &gt;= 0. If this parameter is not specified, and cnv.stringency.setting is set to <strong>High</strong>, then this parameter is set to 4. cnv.stringency.setting is set to <strong>Low</strong>, then this parameter is set to 1.</td>
</tr>
<tr>
<td>cnv.insertions.min.windows</td>
<td>2</td>
<td>Minimum number of windows for the regions to be shown as copy number insertions. Allowed values: Integers &gt;= 0. If this parameter is not specified, and cnv.stringency.setting is set to <strong>High</strong>, then this parameter is set to 4. cnv.stringency.setting is set to <strong>Low</strong>, then this parameter is set to 1.</td>
</tr>
<tr>
<td>cnv.deletions.max.pval</td>
<td>1.0</td>
<td>Maximum p-value for regions to be shown as copy number deletions. Allowed values: Floats 0.0–1.0. If this parameter is not specified, and cnv.stringency.setting is set to <strong>High</strong>, then this parameter is set to 0.1.</td>
</tr>
<tr>
<td>cnv.insertions.max.pval</td>
<td>1.0</td>
<td>Maximum p-value for regions to be shown as copy number insertions. Allowed values: Floats 0.0–1.0. If this parameter is not specified, and cnv.stringency.setting is set to <strong>High</strong>, then this parameter is set to 0.1.</td>
</tr>
</tbody>
</table>
Table 45  CNV module parameter descriptions (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cnv.deletions.max.ratio</td>
<td>0.5</td>
<td>Maximum ratio between the coverage of the region and the expected coverage, for a region to be called as CNV deletion. Allowed values: Floats 0.0–1.0. If this parameter is not specified, and • cnv.stringency.setting is set to High, then this parameter is set to 0.3. • cnv.stringency.setting is set to Low, then this parameter is set to 0.6.</td>
</tr>
<tr>
<td>cnv.insertions.min.ratio</td>
<td>1.25</td>
<td>Minimum ratio between the coverage of the region and the expected coverage, for a region to be called as CNVs insertion. Allowed values: Floats &gt;= 1.0. If this parameter is not specified, and • cnv.stringency.setting is set to High, then this parameter is set to 1.5. • cnv.stringency.setting is set to Low, then this parameter is set to 1.1.</td>
</tr>
</tbody>
</table>

Resource parameters

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>memory.request</td>
<td>15gb</td>
<td>Memory request. Include the units gb in the setting.</td>
</tr>
<tr>
<td>java.heap.space</td>
<td>3000</td>
<td>Dynamic memory request, in megabytes. Do not enter a unit with the numeric value.</td>
</tr>
<tr>
<td>processors.per.node</td>
<td>8</td>
<td>Number of processors per node. The maximum number of CPUs that all simultaneous processes are expected to use to full capacity.</td>
</tr>
</tbody>
</table>

Table 46 shows the non-default filter parameter values for each level of stringency that is set with the cnv.stringency.setting parameter. For GUI users, setting the cnv.stringency.setting parameter sets the other filter parameters with the non-default values shown in this table. For command shell users, changes to the cnv.stringency.setting parameter do not affect other filtering parameters, and you must set each filtering parameter individually. The medium stringency level is not listed because it calls for default values of the filtering parameters. If you explicitly set one of the listed filter parameters, that value overrides the value called for by the CNV stringency level.

Table 46  Filter parameters affected by the CNV stringency level

<table>
<thead>
<tr>
<th>Call stringency setting</th>
<th>Filter parameter</th>
<th>Parameter setting change</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Default value</td>
</tr>
<tr>
<td>High</td>
<td>cnv.deletions.min.mappability</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>cnv.insertions.min.mappability</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>cnv.deletions.min.windows</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>cnv.insertions.min.windows</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>cnv.deletions.max.pval</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>cnv.insertions.max.pval</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>cnv.deletions.max.ratio</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>cnv.insertions.min.ratio</td>
<td>1.25</td>
</tr>
</tbody>
</table>
Table 46 Filter parameters affected by the CNV stringency level (continued)

<table>
<thead>
<tr>
<th>Call stringency setting</th>
<th>Filter parameter</th>
<th>Parameter setting change</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>cnv.deletions.min.mappability</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>cnv.insertions.min.mappability</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>cnv.deletions.min.windows</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>cnv.insertions.min.windows</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>cnv.deletions.max.ratio</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>cnv.insertions.min.ratio</td>
<td>1.25</td>
</tr>
</tbody>
</table>

Table 47 describes CNV parameters that we do not recommend changing.

Table 47 Annotation internal parameters description

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
</table>
| cnv.run          | 1             | Enables the CNV module. Allowed values:  
|                  |               | • 0: Do not run a CNV analysis.  
|                  |               | • 1: Run the CNV module during this analysis.  
|                  |               | The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use. |
| cnv.mappability.dir | ${analysis.mappability.dir} | Path to the directory of mappability files.  
|                  |               | This parameter is set automatically by the shell when you use the default hg18 or hg19 assemblies in the reference repository. |

Human CNVs results file format description

This section describes the file format of the OUT files and the GFF file created by the Human CNVs module run.

OUT files

The OUT files and their contents are:

- **AllSegments.out** – Contains all the segments, including segments with normal copy number.
- **AllCNVs.out** – Contains all the segments with copy number deletions or duplications (CN != 2).
- **OutputCNVs.out** – Contains all the segments with copy number deletions or duplications which pass the specified filtering criteria. These segments are the output CNVs, and are the same as those reported in a different format in the GFF file.

The three OUT files share a common file format, which is described in Table 48. You can view the files in a text editor or a spreadsheet application.
Chapter 11  Run a Human CNVs Analysis

Human CNVs results file format description

Wiggle file

The CNV module optionally creates, per chromosome, one wiggle format file named `Coverage_chr*.wig`. This file contains the coverage in specified window sizes. You can visualize the file in a browser such as the Integrative Genomics Viewer (IGV). You can download the IGV browser from the Broad Institute website:

www.broadinstitute.org/igv

The wiggle file specification is available from this site:

http://genome.ucsc.edu/goldenPath/help/wiggle.html

GFF file

The module creates one `OutputCNVs.gff` file. The file formats are described in Table 49. You can view the `OutputCNVs.gff` file in a text editor or a spreadsheet application. You can also visualize the file in a browser such as the Integrative Genomics Viewer (IGV), or in a browser such as the UC Santa Cruz genome browser. You can download the IGV browser from the Broad Institute website. For more information, go to these sites:

www.broadinstitute.org/igv

http://genome.ucsc.edu

Table 48  CNV *.out file format descriptions

<table>
<thead>
<tr>
<th>Column Title</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chrom</td>
<td>Chromosome number.</td>
<td>chr1</td>
</tr>
<tr>
<td>start</td>
<td>Start location of the CNV region.</td>
<td>1636395</td>
</tr>
<tr>
<td>end</td>
<td>End location of the CNV region.</td>
<td>1780200</td>
</tr>
<tr>
<td>mappability</td>
<td>Fraction of mappable bases in the CNV region.</td>
<td>83.745453</td>
</tr>
<tr>
<td>coverage</td>
<td>Mean coverage of the windows in the CNV region.</td>
<td>4.080000</td>
</tr>
<tr>
<td>log2Ratio</td>
<td>Mean of Log2Ratios of the windows in the CNV region.</td>
<td>-1.126154</td>
</tr>
<tr>
<td>copy number</td>
<td>Copy number of the region.</td>
<td>The copy number is relative to a diploid genome, with a normal copy number of 2.</td>
</tr>
<tr>
<td>numWindows</td>
<td>Number of windows in the CNV region.</td>
<td>22</td>
</tr>
<tr>
<td>p-val</td>
<td>p-value of the CNV call for the region.</td>
<td>0.00001 is very confident. 0.99 is not at all confident.</td>
</tr>
</tbody>
</table>

Table 49  OutputCNVs.gff file format

<table>
<thead>
<tr>
<th>Column Title</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chr</td>
<td>The ID of the sequence to which the start and end coordinates refer.</td>
<td>chr1</td>
</tr>
<tr>
<td>Source</td>
<td>Free text-qualifier indicating the algorithm or method that generated the feature.</td>
<td>AB_CNV_PIPELINE</td>
</tr>
<tr>
<td>Type</td>
<td>Sequence ontology derived type for this variation.</td>
<td>repeat_region</td>
</tr>
<tr>
<td>Pos_Start</td>
<td>Start position of the CNV Region.</td>
<td>1144255</td>
</tr>
<tr>
<td>Pos_End</td>
<td>End position of the CNV Region.</td>
<td>0.04223</td>
</tr>
<tr>
<td>Score</td>
<td>p-Value of the CNV Region.</td>
<td>—</td>
</tr>
<tr>
<td>Strand</td>
<td>Not used for this output.</td>
<td>—</td>
</tr>
</tbody>
</table>
Human CNVs results file examples

This section provides examples of the *.out files and the GFF file created when you run the Human CNVs module. This output can be viewed in a text editor or a spreadsheet application.

The following is example *.out file output, from an OutputCNVs.out file.

<table>
<thead>
<tr>
<th>chrom</th>
<th>start</th>
<th>end</th>
<th>mappability</th>
<th>coverage</th>
<th>logRatio</th>
<th>copynumber</th>
<th>numWindows</th>
<th>attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>chr12</td>
<td>50772857</td>
<td>50798807</td>
<td>59.6667</td>
<td>6.13873</td>
<td>-1.23401</td>
<td>1</td>
<td>3</td>
<td>copynumber=1;numWindows=3;coverage=6.13873;logRatio=-1.23401;mappability=59.6667</td>
</tr>
<tr>
<td>chr12</td>
<td>51001962</td>
<td>51067993</td>
<td>98.3077</td>
<td>23.8285</td>
<td>0.628887</td>
<td>4</td>
<td>13</td>
<td>copynumber=4;numWindows=13;coverage=23.8285;logRatio=0.628887;mappability=98.3077</td>
</tr>
</tbody>
</table>

The following is example output of the file OutputCNVs.gff:

```bash
##gff-version 3
##source-version LifeScope(tm) 2.0 {CNVcaller}
##type DNA
##date 2011-02-10
##time 15:44:31
##genome-build NCBI B36
##input-files /local/gottimrk/cnv_test_datasets/solid0122_20100412_chr12Q_chr16Q_CNVs-Paired.bam
#Chr Source Type    Pos_Start  Pos_End  Score    Strand    Phase    Attributes
chr12 AB_CNV_PIPELINE repeat_region 50772857 50798807 0.00012201 . .
copynumber=1;numWindows=3;coverage=6.13873;logRatio=-1.23401;mappability=59.6667
chr12 AB_CNV_PIPELINE repeat_region 51001962 51067993 1.53379e-16 . .
copynumber=4;numWindows=13;coverage=23.8285;logRatio=0.628887;mappability=98.3077
```

Table 49 OutputCNVs.gff file format (continued)

<table>
<thead>
<tr>
<th>Column Title</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase [attribute]</td>
<td>Not used for this output.</td>
<td>—</td>
</tr>
</tbody>
</table>

Attributes

| copynumber | Copy number of the region. | 1 |
| logRatio | Mean of Log2Ratios of the windows in the CNV region. | -1.258771 |
| numWindows | Number of windows in the Human CNV region. | 23 Usually, the larger this number is, the more confident the CNV call is. |
| coverage | Mean coverage of the windows in the CNV region. | 4.080000 |
| mappability | Fraction of mappable bases in the Human CNV region. | 91.421745 |

The maximum value is 100. A low number may indicate that this region is difficult to map, increasing the likelihood of a false positive CNV call.
Algorithm for the Human CNV module

The Human CNV module algorithm has six steps:

- Preprocessing
- Coverage calculation
- Sampling into windows
- Normalization
- Segmentation
- Post processing

**Preprocessing**
During preprocessing, user-defined configuration parameters are validated and initialized. The BAM file headers are validated against the reference sequence. Information about the chromosomes (including chromosome name, start and end locations of arms) is loaded from the mappability file. The chromosomes that are present in both the mappability file and in the reference sequence (with exactly the same name) are selected for the analysis.

**Coverage calculation**
A position’s coverage is the number of alignments spanning the position in the chromosome. The coverage at every position is computed from the BAM file. By default, all alignments with a quality value of at least 8 are used to calculate coverage. Low quality alignments can be filtered out by modifying the mapping quality threshold parameter (cnv.min.quality).

Optionally, the coverage computed for every chromosome in this step is output in WIG file format. The cnv.coverage.wsize parameter controls the window sizes for this coverage output.

**Sampling into windows**
The algorithm divides the chromosomal region into windows of variable size, depending upon the mappability of the region. The window sizes are determined dynamically so that exactly the same number of mappable positions are in each window. The program distinguishes between mappable and unmappable positions using the precomputed mappability files.

The coverage mean for every window is computed by taking the average of the coverage values from all the mappable positions in each window. The log ratios between the coverage mean of every window and the expected coverage are computed. The mean of all windows in the whole chromosome arm is used as the expected coverage.

**GC Correction**
GC content is the number of G or C bases compared to the total number of bases in a particular region. In the regions of the genome where the percentage of GC content is either high or low, the coverage is observed to be decreased (see Figure 13 on page 218).
Figure 13  Extreme GC contact reduces coverage

In Figure 13, the X-axis is the log ratio (coverage of the window or coverage of the whole chromosome arm). The Y-axis is the percentage of GC content in each window.

The algorithm normalizes this effect of GC contact by scaling the coverage of the windows with extreme GC content to the median coverage. The scaling factors for the windows are computed inline for every chromosome arm during runtime by the algorithm.

**Normalization**

The previous section explained how the log ratios computed in the previous step use the mean coverage of the local chromosomal arm as the expected value. However, if the algorithm is used with these settings, it cannot detect large CNV regions spanning more than half the length of the chromosome arm. To detect large CNVs, which can take whole chromosome arms or large portions of chromosome arms, the algorithm performs global normalization. The algorithm computes log ratios by using the median of the coverage means of all chromosome arms as the expected value (see Figure 14).

To skip the normalization step, set the local normalization parameter `cnv.local.normalization` to 1. By default, the algorithm performs global normalization if the number of valid chromosome arms provided for the analysis is nine or higher. In some cases, you might perform an initial run using Global Normalization, and then perform a second run using local normalization to detect more fine-grained Human CNVs.
Algorithm for the Human CNV module

Figure 14 Large CNV segments example

**Segmentation**

The algorithm uses the Finite First Order Bayesian Hidden Markov Model (“the model”) to take the normalized log ratios of the windows as input and convert the continuous log ratio values into discrete copy number states (see Figure 15).

The model consists of 10 states \{0, 1, 2, 3, 4, 9\}, where each state represents a copy number. For diploid species, state 2 is the normal state, states less than 2 are copy number deletions, and states greater than 2 are copy number amplifications.

The prior probabilities and transition probability matrix for the model are trained using the Baum-Welch EM algorithm.

A copy number state with a p-value is assigned to each window using the Viterbi decoding algorithm. The p-value is a measurement of statistical significance. In general, the lower the p-value, the more likely it is that the prediction will be true.

Figure 15 Segmentation example
Post processing

Adjacent windows with the same copy number are merged into a segment.

The algorithm applies filtering criteria on the Human CNV calls according to parameter settings (see Table 45 on page 211 for descriptions of these parameters). The filtering criteria can include parameters defined for minimum mappability, number of continuous blocks with copy number greater than two, and so forth. The algorithm uses the GFF format for structural variants to format the Human CNV segments that pass all of the filtering criteria. See Table 49 on page 215 for a description of the CNV GFF file format.

FAQ – Human CNVs

1

**Does the Human CNVs module work on any species?**

No. The current version of the module works only for the human hg18 and hg19 reference. The module cannot work on other species because LifeScope™ Software does not have predicted mappability files for any species other than human.

2

**How do various parameters affect the sensitivity and specificity of the CNV calls?**

The default configuration is designed such that CNV calls are made with a balance between sensitivity and specificity.

To increase the specificity, increase the `cnv.window.size` value, use only high quality alignments by filtering out the low quality ones using `cnv.min.quality`, and change the `cnv.stringency.setting` to High.

To increase the sensitivity, work at higher resolution by using a smaller value for `cnv.window.size`, by using a lower value for `cnv.min.quality`, and by changing the `cnv.stringency.setting` to Low.

3

**What do the mappability files contain?**

The mappability files are essentially a representation of reads from the reference that are mapped back to the reference. The mappability files have one row per every position, indicating whether that position is or is not uniquely mappable. A *mer* is the number of the bases per read. These files are in a binary format (HDF).

4

**What mappability files are provided?**

LifeScope™ Software provides mappability files for 50.4 fragment mapping.
**5**

**Can users generate mappability files if they have FASTA files?**

LifeScope™ Software does not provide any applications that allows users to generate their own customized mappability files. Contact your Life Technologies BioInformatics FAS for additional questions.

---

**6**

**When should we change the default value of the `cnv.window.size` parameter?**

The size of Human CNV segments detected by the module is directly dependent on the value of `cnv.window.size`. Typically, the smallest Human CNV segment that can be detected is at least twice the value of the `cnv.window.size` setting. The smaller the window size, the smaller the CNV that can be detected. However very small CNVs may be more likely to be false positives (or at least, under-represented in existing public databases).

As the `cnv.window.size` value decreases, the time taken for the Human CNV analysis and the sizes of the files generated increases significantly.

---

**7**

**When should we use the `cnv.local.normalization` parameter?**

To detect smaller Human CNVs (kbp scale) with tumor samples, normalize by the local chromosome context to detect these smaller Human CNVs. If you do not use `cnv.local.normalization`, large perturbations in ploidy across the whole genome might confuse detection. Ploidy is the number of complete sets of chromosomes in a biological cell. In humans, the somatic cells that compose the body are diploid (containing two complete sets of chromosomes, one set derived from each parent).

The module applies various user-configurable filtering criteria, such as minimum mappability, number of continuous blocks, and so forth on the Human CNV calls.

---

**8**

**How are the p-values for Human CNV segments computed?**

The p-values for Human CNV segments are computed using probabilities from the Finite First Order Bayesian Hidden Markov Model (the model). Sequence of log ratios of coverage per window are given as input to the model for segmentation. The module calculates the most probable Human CNV-state, that is, the hidden state, for each window using “Forward-Backward” Algorithm on the model. For example, if window ‘W’ is assigned Human CNV state ‘c’ with a probability ‘p’, then the p-value of that window is ‘1-p’. The calculations control Human CNV states and p-values for all windows. In the next step, the CNV algorithm merges the neighboring windows with similar copy number states into a Human CNV segment. The p-value of that Human CNV segment is given by the minimum p-value of all merged windows.
How do I interpret a CNV with a copy number value of 2?

CNVs with a copy number value of 2 can occur in male samples because these samples have only one X chromosome and one Y chromosome. Because the normal copy number is 1 for these chromosomes in males, a CNV with copy number 2 on these chromosomes is a gain in copy number. All CNVs that have a copy number value of 2 are present on either chromosome X or Y, and not on other chromosomes.
Run an Inversions Analysis

This chapter covers:

- Overview .................................................. 223
- Examples of running an inversion analysis .................. 224
- Inversion module parameters ................................ 224
- Input files .................................................. 226
- Inversion output files ....................................... 227
- Inversion algorithm ........................................ 230

Overview

An inversion is defined by its two breakpoints. The numbers of mate pairs supporting occurrence of the starting and ending inversion breakpoints are counted for each base pair. The genomic ranges corresponding to local peaks of these counts, if above a score threshold, are called as candidate breakpoint ranges. To define an inversion, its starting and ending breakpoints are paired up only if they are the reciprocal nearest neighbor of each other in the correct order. The score for the inversion is the harmonic mean of its two breakpoints. Each breakpoint range can be scanned for coverage of normal (AAA) mate pairs to identify a sub-range with the lowest normal mate pair coverage as the most probable location of a breakpoint, and to differentiate homozygous inversions from heterozygous ones. The supporting evidence of all inversions can be visually inspected using a genomic browser.
Examples of running an inversion analysis

In a standard workflow

The following standard workflows provide examples of running the inversion module:

- genomic.resequencing.lmp
- genomic.resequencing.pe

The following are example shell commands using reads and references from the LifeScope™ Software repository.

```bash
lscope.sh shell -u username -w password
cd /projects
mk ecoli
cd ecoli
mk run1
cd run1
set workflow genomic.resequencing.pe
add xsq run0209a_50_PE.xsq
add xsq run0209b_50_PE.xsq
set reference hg19
# optionally change inversion defaults here
ls
run
```

See for Chapter 7, “Run a Standard Workflow Analysis” on page 103 for more information on running standard workflows. Chapter 6, “Run a Command Shell Analysis” on page 75 for information on shell commands and syntax.

As a demo analysis

The optional examples download includes an example of how to run the inversion module by itself, as a single analysis that is not part of a standard workflow. The demo example is not recommended for general use.

See Appendix H, “Demo Analyses” on page 567 for information on running the inversion module as a standalone analysis.

Inversion module parameters

Table 50 describes the parameters that control the inversion algorithm.

In order to change a parameter value in your analysis, use the `set param shell` command to replace the line `# optionally change inversion defaults here` in the example commands in “In a standard workflow”. For instance, to change recover small inversion to on, set the `inversion.recover.tiny.inversions` parameter to 1 with this command:

```bash
set inversion.recover.tiny.inversions 1 tertiary/inversion.ini
```

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional parameters</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 50  Inversion parameter description
### Table 50 Inversion parameter description (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>library.type</td>
<td>matepair</td>
<td>The library type for alignment data. Use when the BAM LB: header field is not populated. Allowed values: matepair, pairedend.</td>
</tr>
<tr>
<td>inversion.calculate.mp.coverage</td>
<td>0</td>
<td>Whether to calculate normal mate-pair coverage around inversion breakpoints. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Do not use this calculation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Use this calculation.</td>
</tr>
<tr>
<td>inversion.number.of.chromosomes</td>
<td>25</td>
<td>The number of chromosomes used to distribute jobs on a cluster (one job per chromosome). Allowed values: Integers &gt;= 0.</td>
</tr>
<tr>
<td>inversion.abx.score</td>
<td>0</td>
<td>The ABX score. ABX is ABA/ABB/ABC, which are mates with both tags on the correct strands but in reverse order. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Do not use ABX types as candidate inversions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Use ABX types as candidate inversions.</td>
</tr>
<tr>
<td>inversion.force.update.intermediate.files</td>
<td>0</td>
<td>Whether to force updating of all intermediate files. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Do not force updating of all intermediate files.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Force updating of all intermediate files.</td>
</tr>
<tr>
<td>inversion.down.weight.mp.mismatches</td>
<td>0</td>
<td>Whether to down-weight mate-pairs with mismatches exponentially. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Do not down-weight mate-pairs with mismatches exponentially</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Down-weight mate-pairs with mismatches exponentially</td>
</tr>
<tr>
<td>inversion.max.bxx.mp.length</td>
<td>3000000</td>
<td>The maximal mapped length of BXX mate-pairs. Allowed values: Integers &gt;= 0.</td>
</tr>
<tr>
<td>inversion.max.inversion.length</td>
<td>100000</td>
<td>The maximum mapped length of inversions. Allowed values: Integers &gt;= 0.</td>
</tr>
<tr>
<td>inversion.min.pairing.quality</td>
<td>10</td>
<td>Minimum pairing quality threshold. Allowed values: Integers 0–100.</td>
</tr>
<tr>
<td>inversion.score.run.individually</td>
<td>0</td>
<td>Whether to score every run individually. The parameter is used when rerunning the inversions module with a different breakpoint score threshold, and reusing the inversion evidences and score calculated from previous run. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Do not score every run individually.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Score every run individually.</td>
</tr>
<tr>
<td>inversion.breakpoint.rescue</td>
<td>0</td>
<td>Whether to pair breakpoints with rescue. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Do not pair breakpoints with rescue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Pair breakpoints with rescue.</td>
</tr>
<tr>
<td>inversion.recover.tiny.inversions</td>
<td>0</td>
<td>Whether to recover small inversions. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Do not recover small inversions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Recover small inversions.</td>
</tr>
<tr>
<td>inversion.max.length.tiny.inversions</td>
<td>0</td>
<td>The maximum mapped length of small inversions. Allowed values: Integers &gt;= 0.</td>
</tr>
<tr>
<td>inversion.breakpoint.score.threshold</td>
<td>4</td>
<td>The breakpoint score threshold. Allowed values: Integers 0–100.</td>
</tr>
</tbody>
</table>
Table 50  Inversion parameter description (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inversion.sab.gff.score.threshold</td>
<td>0</td>
<td>The output score threshold. Allowed values: Integers 0–100.</td>
</tr>
<tr>
<td>inversion.breakpoint.peak.width</td>
<td>100</td>
<td>The breakpoint peak width. Allowed values: Integers 0–100.</td>
</tr>
</tbody>
</table>

Resource parameters

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>memory.request</td>
<td>21gb</td>
<td>Memory request. Include the units gb in the setting.</td>
</tr>
<tr>
<td>java.heap.space</td>
<td>3000</td>
<td>Dynamic memory requirement, in megabytes. Do not enter a unit with the numeric value.</td>
</tr>
</tbody>
</table>

Table 50 describes the inversion parameters that we do not recommend changing.

Table 51  Inversion internal parameter description

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inversion.run</td>
<td>1</td>
<td>Whether or not to run the inversion module.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Do not run the inversions module.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Run the inversions module during this analysis.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use.</td>
</tr>
<tr>
<td>inversion.max.anchor.mismatch</td>
<td>0</td>
<td>Filter out mates with more anchor mismatches on either tag. Allowed values: Integers 0–10.</td>
</tr>
<tr>
<td>inversion.min.alignment.length</td>
<td>0</td>
<td>Filter out mates with shorter alignment length on either tag. Allowed values: Integers 0–100.</td>
</tr>
<tr>
<td>inversion.max.alignment.start</td>
<td>0</td>
<td>Filter out mates with farther alignment start position on either tag. Allowed values: Integers 0–15.</td>
</tr>
</tbody>
</table>

Input files

The inversion module takes the BAM file output of the mapping module as input. Because the inversion module specifically selects for pairs that are in the incorrect orientation, it is important to know the correct orientation. As a result, either the library type information in the LB field or the library.type parameter must be properly set.

Libraries in the input file or files must be either:
• All mate-pair
• All paired-end

When different library types are provided as input on the same run, the inversion module fails with an error.

Read lengths

Input BAM files may contain multiple read lengths.
# Inversion output files

Table 52 describes the inversion GFF output file format.

<table>
<thead>
<tr>
<th>Column title or number</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Chromosome.</td>
<td>chr10</td>
</tr>
<tr>
<td>2</td>
<td>Method.</td>
<td>AB_SOLiD</td>
</tr>
<tr>
<td>3</td>
<td>Feature keywords.</td>
<td>inversion</td>
</tr>
<tr>
<td>4</td>
<td>Inversion start coordinate.</td>
<td>46443097</td>
</tr>
<tr>
<td>5</td>
<td>Inversion end coordinate.</td>
<td>46479578</td>
</tr>
<tr>
<td>6</td>
<td>Inversion score.</td>
<td>129.8</td>
</tr>
<tr>
<td>7</td>
<td>Not used.</td>
<td>—</td>
</tr>
<tr>
<td>8</td>
<td>Not used.</td>
<td>—</td>
</tr>
<tr>
<td>9 Attributes, see below:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>left‡</td>
<td>Starting breakpoint range.</td>
<td>left=chr10:46443097-46443161</td>
</tr>
<tr>
<td>right</td>
<td>Ending breakpoint range.</td>
<td>right=chr10:46479540-46479578</td>
</tr>
<tr>
<td>leftscore</td>
<td>Left breakpoint range score.</td>
<td>leftscore=185</td>
</tr>
<tr>
<td>rightscore</td>
<td>Right breakpoint range score.</td>
<td>rightscore=100</td>
</tr>
<tr>
<td>count_AAA_further_left</td>
<td>The number of AAA mate-pairs spanning the genomic region to the immediate left of the starting breakpoint range.</td>
<td>count_AAA_further_left=1</td>
</tr>
<tr>
<td>count_AAA_left</td>
<td>The number of AAA mate-pairs spanning the starting breakpoint range.</td>
<td>count_AAA_left=1</td>
</tr>
<tr>
<td>count_AAA_right</td>
<td>The number of AAA mate-pairs spanning the ending breakpoint range.</td>
<td>count_AAA_right=2</td>
</tr>
<tr>
<td>count_AAA_further_right</td>
<td>The number of AAA mate-pairs spanning the genomic region to the immediate right of the ending breakpoint range.</td>
<td>count_AAA_further_right=1</td>
</tr>
<tr>
<td>left_min_count_AAA</td>
<td>Sub-range within starting breakpoint range that has the minimal AAA coverage.</td>
<td>left_min_count_AAA=chr10:46443097-46443112</td>
</tr>
<tr>
<td>count_AAA_min_left</td>
<td>Minimal AAA coverage in starting breakpoint range.</td>
<td>count_AAA_min_left=0</td>
</tr>
<tr>
<td>count_AAA_max_left</td>
<td>Maximal AAA coverage in starting breakpoint range.</td>
<td>count_AAA_max_left=4</td>
</tr>
<tr>
<td>right_min_count_AAA</td>
<td>Sub-range within ending breakpoint range that has the minimal AAA coverage.</td>
<td>right_min_count_AAA=chr10:46479576-46479578</td>
</tr>
<tr>
<td>count_AAA_min_right</td>
<td>Minimal AAA coverage in ending breakpoint range.</td>
<td>count_AAA_min_right=4</td>
</tr>
<tr>
<td>count_AAA_max_right</td>
<td>Maximal AAA coverage in ending breakpoint range.</td>
<td>count_AAA_max_right=11</td>
</tr>
<tr>
<td>homozygous</td>
<td>Whether AAA coverage at both breakpoints ranges are lower than one-fifth of their neighboring ranges.</td>
<td>homozygous=YES</td>
</tr>
</tbody>
</table>

‡ This and the remaining entries are allowed values for Attributes in column 9.
Chapter 12 Run an Inversions Analysis

Inversion output files

The following is an example of the file `inversions.s2.w100.100000.GFF`, whose format is described in Table 56 on page 229:

```plaintext
##gff-version 3
##generated by SOLiD inversion tool
chr10 AB_SOLiD inversion 46442583 46479737 4 . .
  left=chr10:46442583-46443522;right=chr10:46479431-
  46479737;leftscore=6.0;rightscore=3.0;count_AAA_further_left=0;count_AAA_left
  =0;count_AAA_right=0;count_AAA_further_right=0;left_min_count_AAA=chr10:46442
  583-46443522;count_AAA_min_left=0;count_AAA_max_left=0;right_min_count_AAA=chr10:
  46479431-46479737;count_AAA_min_right=0;count_AAA_max_right=0;homozygous=YES
chr21 AB_SOLiD inversion 26295366 26297167 2.7 . .
  left=chr21:26295366-26296041;right=chr21:26296491-
  26297167;leftscore=2.7;rightscore=2.7;count_AAA_further_left=0;count_AAA_left
  =0;count_AAA_right=0;count_AAA_further_right=0;left_min_count_AAA=chr21:26295
  366-26296041;count_AAA_min_left=0;count_AAA_max_left=0;right_min_count_AAA=chr21:
  26296491-26297167;count_AAA_min_right=0;count_AAA_max_right=0;homozygous=YES
chr6  AB_SOLiD inversion 168834702 168837413 2.1 . .
  left=chr6:168834702-168835567;right=chr6:168836564-
  168837413;leftscore=2.2;rightscore=2.1;count_AAA_further_left=0;count_AAA_lef
  t=0;count_AAA_right=0;count_AAA_further_right=0;left_min_count_AAA=chr6:16883
  4702-168835567;count_AAA_min_left=0;count_AAA_max_left=0;right_min_count_AAA=chr6:
  168836564-168837413;count_AAA_min_right=0;count_AAA_max_right=0;homozygous=YES
```

Table 53 Inversion all.chr, all.chr*, pair.orphan, pair.txt file format description

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Display label. For internal use.</td>
<td>—</td>
</tr>
<tr>
<td>2</td>
<td>GFF name, type of breakpoint.</td>
<td>InvStart</td>
</tr>
<tr>
<td>3</td>
<td>GFF type.</td>
<td>exon</td>
</tr>
<tr>
<td>4</td>
<td>Range start coordinate.</td>
<td>1297</td>
</tr>
<tr>
<td>5</td>
<td>Range end coordinate.</td>
<td>1362</td>
</tr>
<tr>
<td>6</td>
<td>Breakpoint range score, number of supporting mate-pairs.</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>Strand.</td>
<td>—</td>
</tr>
<tr>
<td>8</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>9</td>
<td>—</td>
<td>g=1</td>
</tr>
<tr>
<td>10</td>
<td>Chromosome.</td>
<td>chr1</td>
</tr>
</tbody>
</table>
Table 54  inversion rank.txt file format description

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Chromosome.</td>
<td>chr1</td>
</tr>
<tr>
<td>2</td>
<td>Inversion start coordinate.</td>
<td>1632874</td>
</tr>
<tr>
<td>3</td>
<td>Inversion end coordinate.</td>
<td>1706252</td>
</tr>
<tr>
<td>4</td>
<td>Inversion score.</td>
<td>4.8</td>
</tr>
<tr>
<td>5</td>
<td>Inversion length.</td>
<td>73379</td>
</tr>
<tr>
<td>6</td>
<td>Starting breakpoint range.</td>
<td>chr1:1632874-1633166</td>
</tr>
<tr>
<td>7</td>
<td>Ending breakpoint range.</td>
<td>chr1:1705726-1706252</td>
</tr>
<tr>
<td>8</td>
<td>Left breakpoint range score.</td>
<td>6.0</td>
</tr>
<tr>
<td>9</td>
<td>Right breakpoint range score.</td>
<td>4.0</td>
</tr>
</tbody>
</table>

Table 55  Inversion AAA GFF file format description

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Mate-pair bead ID.</td>
<td>1_11_215_288</td>
</tr>
<tr>
<td>2</td>
<td>GFF name.</td>
<td>LEFT</td>
</tr>
<tr>
<td>3</td>
<td>GFF type.</td>
<td>exon</td>
</tr>
<tr>
<td>4</td>
<td>Mate-pair start coordinate.</td>
<td>3971</td>
</tr>
<tr>
<td>5</td>
<td>Mate-pair end coordinate.</td>
<td>4020</td>
</tr>
<tr>
<td>6</td>
<td>Score, number of mismatches.</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>Strand.</td>
<td>+</td>
</tr>
<tr>
<td>8</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>9</td>
<td>—</td>
<td>g=</td>
</tr>
</tbody>
</table>

Table 56  Inversion coords.inversions.s*.* and coords.orphan.* file format description

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The genomic coordinate of an inversion.</td>
<td>chr10:46443097-46479578</td>
</tr>
<tr>
<td>2</td>
<td>Inversion score.</td>
<td>129.8</td>
</tr>
</tbody>
</table>

The following is an example of the file coords.inversions.s2.w100.100000+, whose format is described in Table 56:

```
chr10:46442583-46479737   4
chr21:26295366-26297167   2.7
chr6:168834702-168837413  2.1
```
Chapter 12  Run an Inversions Analysis

Inversion algorithm

Overview

The inversion module exploits the large insert size of SOLiD® System mate-pair libraries to detect inversion polymorphisms that are important but often poorly characterized. The large insert size is critical for spanning the repeat regions that are associated with inversion break points.

Table 57  Inversion AAA/*.txt and AAA.txt file format description

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Chromosome.</td>
<td>chr10</td>
</tr>
<tr>
<td>2</td>
<td>Inversion start coordinate.</td>
<td>46443097</td>
</tr>
<tr>
<td>3</td>
<td>Inversion end coordinate.</td>
<td>46479578</td>
</tr>
<tr>
<td>4</td>
<td>Inversion score.</td>
<td>129.8</td>
</tr>
<tr>
<td>5</td>
<td>Inversion length.</td>
<td>36482</td>
</tr>
<tr>
<td>6</td>
<td>Starting breakpoint range.</td>
<td>chr10:46443097-46443161</td>
</tr>
<tr>
<td>7</td>
<td>Ending breakpoint range.</td>
<td>chr10:46479540-46479578</td>
</tr>
<tr>
<td>8</td>
<td>Left breakpoint range score.</td>
<td>185</td>
</tr>
<tr>
<td>9</td>
<td>Right breakpoint range score.</td>
<td>100</td>
</tr>
<tr>
<td>10</td>
<td>The number of AAA mate-pairs spanning the genomic region to the immediate left of the starting breakpoint range.</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>The number of AAA mate-pairs spanning the starting breakpoint range.</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>The number of AAA mate-pairs spanning the ending breakpoint range.</td>
<td>2</td>
</tr>
<tr>
<td>13</td>
<td>The number of AAA mate-pairs spanning the genomic region to the immediate right of the ending breakpoint range.</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 58  Inversion rescore.txt file format description

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-13</td>
<td>Same as AAA.txt</td>
<td>See Table 57.</td>
</tr>
<tr>
<td>14</td>
<td>Sub-range within starting breakpoint range that has the minimal AAA coverage.</td>
<td>chr10:46443097-46443112</td>
</tr>
<tr>
<td>15</td>
<td>Minimal AAA coverage in starting breakpoint range.</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>Maximal AAA coverage in starting breakpoint range.</td>
<td>4</td>
</tr>
<tr>
<td>17</td>
<td>Sub-range within ending breakpoint range that has the minimal AAA coverage.</td>
<td>chr10:46479576-46479578</td>
</tr>
<tr>
<td>18</td>
<td>Minimal AAA coverage in ending breakpoint range.</td>
<td>4</td>
</tr>
<tr>
<td>19</td>
<td>Maximal AAA coverage in ending breakpoint range.</td>
<td>11</td>
</tr>
</tbody>
</table>
In its simplest form, the algorithm collects evidence for an inversion by observing accumulations of pairs with correct relative positioning, but with an improper orientation. For SOLiD® System mate-pair libraries, this algorithm requires tags to be mapped to opposite strands. For example, if the R3 tag is to the left of an F3 tag, but the R3 tag maps to the top strand and the F3 tag maps to the bottom strand, this pair provides inversion evidence, specifically that the R3 tag is to the left of the inversion starting break point and that the F3 tag is to the right of the break point. Once the evidence is collected, candidate inversion break points are scored, paired, and ranked. Additional evidence is provided by a scan for drops in the coverage by normal mate-pairs (see Figure 16).

Figure 16 depicts the following elements:

| Thick black line | The sequenced genome. |
| Thin black lines | Mate-pairs. |
| Red and green arrows | Two ends of an inverted mate-pair. |
| Blue and orange bars | The maximum distance separating the two ends of a normal mate-pair (AAA mates). |

By definition, an inversion has two breakpoints: a starting breakpoint and an ending breakpoint. The inversion module plug-in uses SOLiD® System mate files or mate-pair BAM files as input (using paired-end BAM files is not recommended due to their small insert size). The number of mate-pairs supporting an occurrence (of both starting and ending inversion breakpoints) are counted for each base pair as breakpoint scores. Candidate breakpoint ranges are genomic ranges corresponding to local peaks of counts above a predetermined score threshold. The clone insert size constraints specify a minimum and a maximum (blue and orange horizontal bars) distance separating the two ends (small green or red arrows) of a mate-pair (thin black lines) in the sequenced
Inversion algorithm

LifeScope™ Genomic Analysis Software Command Shell User Guide

Chapter 12  Run an Inversions Analysis

Input data

The first step in the detection of inversions is the collection of inverted mates. These are filtered from the BAM file by selecting for mates of opposite orientation. In pairing category terms, this filtering includes BA*, BB*, and AB*. These records are written to an intermediate directory, so that the filtered set can be reused. It is important that either the library.type parameter be set or that the LB field be properly constructed to indicate a mate-pair library.

Non-redundant mates are selected from BAM file input using the PCR duplicates flag (0x0400). Pairing quality values can be used to select unique records via the inversion.min.pairing.quality parameter key. Values greater than 20 should be unique, but criteria may be different for different applications.

The Pairing “SV” output filter is specifically designed to capture more of the deviant pairs used for structural variant programs such as the inversion module. A lower inversion.min.qv value (~10) and a BAM file generated with the “SV” output filter can potentially find a larger number of candidate inversions.

Workflow

Figure 17 shows the inversion module workflow. The inversion module begins with a long mate-pair BAM file and proceeds through scoring, pairing, ranking, rescoring, and GFF output.
**Scoring**

Candidate breakpoints are scored by first determining the tag of a pair that is properly oriented. The inverted mate of the pair is then used to select reference locations that are in the vicinity of the inversion breakpoint. All reference locations that are between the inverted mate and the maximum possible insert size accumulate a positive score, resulting in a list of high scoring locations that represent candidate breakpoints.

**Pairing**

The high scoring candidate breakpoints are then recombined into inversion candidates by matching start and end locations that are defined by an analysis of the high scoring locations. A user-definable window size `inversion_breakpoint.peak.width` is used to determine regions with a peak score greater than the threshold `inversion_breakpoint.score.threshold`. These peaks are combined with their reciprocal nearest neighbor to create a set of possible inversions.

In some cases there will be a peak whose nearest neighbor is slightly below the threshold. A rescue analysis will retrieve those that are significant peaks, but below the threshold cutoff. This is controlled by the `inversion.breakpoint.rescue` flag.

A correct threshold for inversion breakpoint scoring depends on clone coverage of sequencing data, its nature, and its mapping quality. The default value 4 is set for 100–500X clone coverage of sequencing from a normal genome. As clone coverage increases, the inversion score threshold must be raised to filter out mismapping noise. Because more inversions can be expected in cancer samples, the threshold may be lowered when running on those samples.
Ranking

Inversions are ranked based on the harmonic mean of the scores of the start and end breakpoints. Additionally, inversions are separated out by the user specified maximum inversion length (inversion.max.inversion.length).

Inversion length thresholds

Table 59 describes how the parameters inversion.max.inversion.length and inversion.max.bxx.mp.length affect inversion calling. The parameter inversion.max.inversion.length also affects the output file name, according to these patterns:

- coords.inversions.s4.w100.<max.inversion.length>-
- coords.inversions.s4.w100.<max.inversion.length>+

The example output file names in this table are based on the parameter setting inversion.max.inversion.length=100000.

Table 59 Inversion threshold parameters

<table>
<thead>
<tr>
<th>Length</th>
<th>Inversion calling behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>From 1 to inversion.max.inversion.length</td>
<td>Inversions are called, into an output file named, for example, coords.inversions.s4.w100.100000-. Note: The minus sign (&quot;-&quot;&quot;) is part of the file name.</td>
</tr>
<tr>
<td>From inversion.max.inversion.length to inversion.max.bxx.mp.length</td>
<td>Inversions are called, into an output file named, for example, coords.inversions.s4.w100.100000+. Note: The plus sign (&quot;+&quot;) is part of the file name.</td>
</tr>
<tr>
<td>Greater then inversion.max.bxx.mp.length</td>
<td>Inversions are not called.</td>
</tr>
</tbody>
</table>

Tiny inversions

If the tiny inversions flag is set (inversion.recover.tiny.inversions), the pairing and ranking components are rerun with a window size that is sufficiently small to detect inversions with a smaller size (controlled by the parameter inversion.max.length.tiny.inversions).

Normal pair coverage

Additional evidence is provided for breakpoints by examining the relative coverage of unique proper pairs. These pairs are selected from the BAM input using the proper pair flag (0x0002), and they are used to reduce the score of reference positions covered by proper pairs.
Run a Small Indels Analysis

This chapter covers:

- **Overview** ................................................. 235
- **Examples of running a small indels analysis** .................. 236
- **Small indel parameter description** ................................ 237
- **Small indel module output file formats** .................. 241
- **Gap alignment mapping** ........................................... 253
- **Small indel calling from gap alignments** .................. 257
- **FAQ – Small indels** ............................................. 267

**Overview**

Small indel detection is a tertiary module in LifeScope™ Genomic Analysis Software. Data in BAM format from the fragment and paired mapping modules serve as direct inputs for small indel discovery. Gap alignments, those with a CIGAR string containing D or I, serve as the primary evidence for this. In the second map part of the mapping module, these evidences are found by taking partial read anchor alignments, and then either extending allowing for a gap, called LOCAL alignment, or as seed placement for a full read, called GLOBAL, very similar to what is done in the pairing phase. Gap alignments are also found separately in the pairing phase of the mapping module. Here, these evidences are found in mate pairs by using the placement of one tag such that the other tag can be used to search for the presence of a small indel. Specifically, with the unplaced tags, the algorithm finds all possible placements of the beginning and ends of the read. Then, these are joined together only if the resulting gap size is within the limits set in the pairing module. All successful joinings form the set of gap alignments.

The LifeScope™ Software small indel module allows flexible processing of these gap alignments. LifeScope™ Software takes input gap alignments from one or more input BAM files and collects them (based on their proximity from each other) to form pileups. The module, in the process of making the pileups or in evaluating these pileups, employs a number of heuristics, including map QV filtering, insertion sequence determination, and assessment of zygosity using ungapped alignment read counting and clustering of gap sizes. In the end, the small indel module produces quality indel calls in a public GFF version 3 format.
Examples of running a small indels analysis

In a standard workflow

The following standard workflows provide examples of running the small indels module:

- genomic.resequencing.frag
- genomic.resequencing.lmp
- genomic.resequencing.pe
- targeted.resequencing.frag
- targeted.resequencing.pe

The following are example shell commands using reads and references from the LifeScope™ Software repository.

```
# log into the shell
lscope.sh shell -u username -w password
# cd to the projects repository
cd /projects
# make a project, and open it
mk proj1
cd proj1
# make an analysis, and open it
mk run1
cd run1
# define the analysis type
set workflow genomic.resequencing.pe
# define the input (this example uses dummy XSQ filenames)
add xsq run0209a_50_PE.xsq
add xsq run0209b_50_PE.xsq
# specify the reference to be used
set reference hg19
# optionally change parameter defaults here
# list the analysis configuration
ls
# run the analysis
run
# list progress information for the analysis
ls
```

In order to change a parameter value in your analysis, add a set param shell command to after the line # optionally change parameter defaults here. For instance, to change the filtering parameter, use this shell command:

```
set small.indel.perform.filtering 0  tertiary/small.indel.ini
```

See for Chapter 7, “Run a Standard Workflow Analysis” on page 103 for more information on running standard workflows. Chapter 6, “Run a Command Shell Analysis” on page 75 for information on shell commands and syntax.

As a demo analysis

The optional examples download includes an sample of how to run the small indel module by itself, as a single analysis that is not part of a standard workflow. The demo example is not recommended for general use.
See Appendix H, “Demo Analyses” on page 567 for information on running the small indel module as a standalone analysis.

## Small indel parameter description

Table 60 describes a workflow parameter, `analysis.space`, that is used in the small indel module. The `analysis.space` parameter is set in the tertiary/global.ini file.

### Table 60  Analysis space parameter used in the small indel module

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>
| `analysis.space`   | auto    | Specifies your preference for base space or color space analysis. Allowed values:
|                    |         | • **auto**: The analysis space is determined based on the mapping space and the presence of color space data in the analysis BAM files.  |
|                    |         | • **base**: Prefer base space analysis.                                     |
|                    |         | • **color**: Prefer color space analysis.                                   |
|                    |         | See “Analysis space” on page 69 for a description of other factors determining the analysis space used by the small indel module. |

Table 61 describes the small indel module parameters.

### Table 61  Small indel module file parameters

<table>
<thead>
<tr>
<th>Optional input parameters</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>analysis.regions.file</code></td>
<td>—</td>
<td>BED format file indicating the target regions for the analysis. Used for targeted analysis. The parameter is managed by the framework. In targeted resequencing and low frequency variant workflows, this parameter is set in the both secondary and tertiary global.ini files.</td>
</tr>
</tbody>
</table>

**General options**

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
</table>
| `small.indel.detail.level` | 3             | For BAM file inputs, the level of detail in output:
|                            |               | • **0**: Keeps only position information about the anchor read and no information for the ungapped alignment.  |
|                            |               | • **1–8**: Keeps only some of the alignment’s anchor alignment but none of the ungapped alignment.  |
|                            |               | • **9**: Is most detailed, but also the slowest.  |
| `small.indel.zygosity.profile.name` | max-mapping | Zygosity profile name.
|                            |               | • **classic**: Profile for classically [full read] mapped reads.  |
|                            |               | • **max-mapping**: Profile for seed and extend ungapped alignments.  |
|                            |               | • **max-mapping-v2**: Reserved for future use.  |
|                            |               | • **gap-align-only**, and **no-calls**: Force all zygosity calls to be homozygous calls. |
Table 61  Small indel module file parameters (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>small.indel.genomic.region</td>
<td>—</td>
<td>Names a specific genomic region to be selected from the BAM file. Only full chromosomes are guaranteed not to alter results. Specifying partial chromosomes is allowed but may result in the loss of indels near the edges of that region. For example, chr1:2945-9659 causes a reduction of coverage for approximately a read length after position 2945.</td>
</tr>
<tr>
<td>small.indel.display.base.qvs</td>
<td>0</td>
<td>Whether or not to display base quality value scores in the GFF file. Allowed values: 0: Do not add QV scores in the output file. 1: Displays the FASTQ base QV scores for all of the reads used for each indel in the GFF file. FASTQ strings contain semi-colons, so adding these strings may produce a GFF file that is not compatible with certain applications.</td>
</tr>
<tr>
<td>small.indel.num-aligns.per.pileup</td>
<td>1000</td>
<td>For pileups with more than this number of alignments, sets the expected number of alignments per pileup. Values much higher than 1000 may result in a significant increase in computational time. After position 9659, indels may be called with partial coverage, with reads contributing to this coverage which also start at position 9659 and earlier. Allowed values: Integers &gt;= 0.</td>
</tr>
<tr>
<td>small.indel.random.seed</td>
<td>94404</td>
<td>The random seed value used to determine which pseudo random set of reads to use when there are greater than 1000 reads in a pileup. The random number generator used is the Mersenne Twister MT19937 algorithm. Allowed values: Integers &gt;= 0.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pileup options</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>small.indel.min.num.evid</td>
<td>2</td>
<td>Minimum number of evidences required for an indel call. This parameter does not have an upper limit, but a value higher than the default of 2, even if it is lower than the average coverage level, may cause a significant reduction in sensitivity.</td>
</tr>
<tr>
<td>small.indel.max.num.evid</td>
<td>-1</td>
<td>Maximum number of evidences. Use -1 for no maximum. Setting this value to some multiple of the average coverage could remove indels found in abnormally high coverage areas.</td>
</tr>
<tr>
<td>small.indel.consGroup</td>
<td>1</td>
<td>Indel grouping method. Allowed values: 1: Conservative grouping of indels with 5bp max between consecutive evidences. 2: Lax grouping. Groups indels that are at maximum the higher of 15 or 7 times the indel size. 9: No grouping. Makes every indel evidence a separate pileup.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mapping quality filtering</th>
<th></th>
<th></th>
</tr>
</thead>
</table>
### Small indel module file parameters (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>small.indel.max.reported.alignments</td>
<td>-1</td>
<td>Only uses those alignments where the NH field (the number of reported alignments; from the BAM record) is this value or lower. The range where this is effective depends on the input BAM file’s range of values of the NH tag.</td>
</tr>
<tr>
<td>small.indel.min.mapping.quality</td>
<td>8</td>
<td>Keeps only reads that have this or higher pairing qualities. For paired tags, mapping quality is for the pair (pairing quality), and for fragment, it is the single tag’s map quality. Reads that are lower than this value are filtered out. Allowed values: Integers -1 to 256.</td>
</tr>
<tr>
<td>small.indel.min.best.mapping.quality</td>
<td>10</td>
<td>For a particular indel called with a set of reads, at least one pairing quality in this set must be higher than this value. Allows for supporting evidences to have a lower mapping quality threshold than the best read. Allowed values: Integers -1 to 256.</td>
</tr>
</tbody>
</table>
| small.indel.min.anchor.mapping.quality              | -1            | Minimum mapping quality for a non-indel (anchor) tag. Effective only for paired reads, for the number of anchors queried as defined by small.indel.detail.level. Uses the SM tag of the BAM record from the mate alignment. Allowed values: Integers -1 to 256.  
**Note:** The BAM spec allows values 0—255. LifeScope™ Software mapping uses values in the range 0—100. A value of -1 or 0 filters no reads, and 256 filters all reads. |
| small.indel.ungapped.bam.flag.filter                | ProperPair,Primary | For ungapped alignments, specifies the BAM flag properties that a read must have to be included. Allowed values: A comma-separated string of one or more of these values:  
• ProperPair  
• UniqueHit (requires the BAM NH field to be set to 1)  
• NoOptDup  
• Primary  
• None  
If None is present in the string, then all filters, even those explicitly set, are turned off. |
| small.indel.gapped.bam.flag.filter                  | Primary       | For gapped alignments, specifies the BAM flag properties that a read must have to be included. Allowed values: A comma-separated string of one or more of these values:  
• ProperPair  
• UniqueHit (requires the BAM NH field to be set to 1)  
• NoOptDup  
• Primary  
• None  
If None is present in the string, then all filters, even those explicitly set, are turned off. |
### Table 61 Small indel module file parameters (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>small.indel.edge.length.deletions</td>
<td>0</td>
<td>Gap alignments that do not have this minimum length on either side of the indel will not be considered. Allowed values: Integers ( \geq 0 ).</td>
</tr>
<tr>
<td>small.indel.edge.length.insertions</td>
<td>0</td>
<td>Gap alignments that do not have this minimum length on either side of the indel will not be considered. Allowed values: Integers ( \geq 0 ).</td>
</tr>
<tr>
<td><strong>Heuristic filtering</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| small.indel.perform.filtering         | 1             | Whether or not to perform filtering in each pileup. Allowed values:  
  - 0: Do not perform filtering on pileups.  
  - 1: Perform heuristic filtering on pileups.  
  Parameters [such as small.indel.min.num.evid] that change the makeup of pileups contained in the PAS.SUM file are still active. |
| small.indel.indel.size.distribution. allowed | can-cluster   | Indel sizes in a pileup are allowed to have certain indel size distributions. Allowed values:  
  - **similar-size**: 75% of the reads of a pileup must have exactly the same size.  
  - **similar-size-any-large-deletions**: Any pileups with at least 2 large deletion alignments, the other pileups must have similar sizes.  
  - **can-cluster**: Allowed if at least one cluster of any indel size is found. [Default].  
  - **can-cluster-any-large-deletions**: Any pileups with at least 2 large deletion alignments; other pileups must be able to cluster (will have indels with two more reads with larger deletions, even if they don’t form good clusters).  
  - **any**: Can have any size distribution (same as small.indel.require.called.indel.size=false). |
| small.indel.remove.singletons         | 1             | Whether or not to remove the singletons that occur when different alignment methods are combined based on identical bead ids and read sequence. Allowed values:  
  - 0: Do not remove the singletons.  
  - 1: Remove singletons. |
| small.indel.alignment.compatibility.filter | 1             | Alignment compatibility level. Checks color space compatibility around the gap. Allowed values:  
  - 0: Disable the check.  
  - 1: Enable the check. |
| small.indel.max.coverage.ratio        | 12            | Maximum allowed value for the ratio of number of reference alignments allowed by the ungapped.bam.flag.filter over the number of non-redundant indel variant reads selected with the gapped.bam.flag.filter. Use –1 for no limit (no coverage ratio filtering). Allowed values: Integers. |
Small indel module output file formats

The main output file of this module is the produced GFF_3 file. This list describes the files generated by the small indel module:

- **.gff** – Contains the resulting small indel calls with extensive annotation in the attribute column.
- **.vcf** – Contains the resulting small indel calls. The VCF file contains the same basic content as the GFF file.
- **.pas.sum** – An internal pileup format which represents collections of gap alignments extracted from the BAM file.
- **.align** – Displays the alignments of reads of the gaps in a human readable format.
- **.ungapped** – For each pileup, contains the list of beads ids from reads that are aligned without gaps but also span the location of the indel.
- **.pas** – A legacy file produced but not used by the caller.

### Table 61 Small indel module file parameters (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>small.indel.max.nonreds-4filt</td>
<td>2</td>
<td>Maximum number of non-redundant reads where read position filtering is applied. Allowed values: Integers.</td>
</tr>
<tr>
<td>small.indel.min.from.end.pos</td>
<td>9.1</td>
<td>Minimum average number of base pairs from the end of the read required of the pileup, when there are at most a certain number of reads defined by small.indel.max.nonreds-4filt. Allowed values: Floats.</td>
</tr>
<tr>
<td>small.indel.min.insertion.size</td>
<td>0</td>
<td>Minimum insertion size to include. Allowed values: Integers.</td>
</tr>
<tr>
<td>small.indel.min.deletion.size</td>
<td>0</td>
<td>Minimum deletion size to include. Allowed values: Integers.</td>
</tr>
<tr>
<td>small.indel.max.insertion.size</td>
<td>1000000000</td>
<td>Maximum insertion size to include. Allowed values: Integers.</td>
</tr>
<tr>
<td>small.indel.max.deletion.size</td>
<td>1000000000</td>
<td>Maximum deletion size to include. Allowed values: Integers.</td>
</tr>
<tr>
<td>memory.request</td>
<td>2gb</td>
<td>Memory request. Include the units gb in the setting.</td>
</tr>
<tr>
<td>java.heap.space</td>
<td>30000</td>
<td>Dynamic memory requirement, in megabytes. Do not enter a unit with the numeric value.</td>
</tr>
<tr>
<td>maximum.workers</td>
<td>24</td>
<td>Maximum number of nodes that can be used for this analysis.</td>
</tr>
<tr>
<td>processors.per.node</td>
<td>1</td>
<td>On a node, the number of parallel threads used to run this process.</td>
</tr>
<tr>
<td>wall.time</td>
<td>336</td>
<td>The maximum total time (in hours) allowed for the process to complete. The default is equivalent to 14 days.</td>
</tr>
</tbody>
</table>
• .txt, .sql – Alternate output format that contain only a subset of the information contained in the GFF file.

Small indel GFF format

As shown in the small indel output file example below (after Table 62), the GFF file created by the small indel module begins with the General File Format Version 3 headers. The section following the GFF headers displays BAM header and read group information. The lines containing information about each indel follows. Table 62 describes each column and attribute contained in the file. Table 63 on page 249 describes the text format, which contains similar information as the GFF file.

Table 62  Small indel GFF file format descriptions

<table>
<thead>
<tr>
<th>Column/Attribute tag name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>seqid</td>
<td>The ID of the sequence to which the start and end coordinates refer, such as a chromosome number.</td>
<td>chrV</td>
</tr>
<tr>
<td>source</td>
<td>Free-text qualifier that indicates the algorithm or method that generated the feature.</td>
<td>AB_SOLiD Small Indel Tool</td>
</tr>
<tr>
<td>type</td>
<td>SOFA feature. For indels, possible values are: • insertion_site • deletion • combination</td>
<td>deletion</td>
</tr>
<tr>
<td>start</td>
<td>Start position of the feature.</td>
<td>200587060</td>
</tr>
<tr>
<td>end</td>
<td>End position of the feature. 1-based integer coordinates of the feature, relative to the sequence in column 1. For zero-length features, such as insertion sites, “start” equals “end” and the implied site is to the right of the indicated base in the direction of the landmark. For deletions, the start and end indicate the positions in the reference that are not present in the sample.</td>
<td>200587063</td>
</tr>
<tr>
<td>score</td>
<td>Floating-point value representing the quality of the evidence for the feature. “Score” is not currently calculated, so all indels are reported with a score of 1.</td>
<td>1</td>
</tr>
<tr>
<td>strand</td>
<td>The strand of the feature. The type of indels detected are not stranded, because they are found with sequence reads that are on either or both strands.</td>
<td>.</td>
</tr>
<tr>
<td>phase</td>
<td>Translation frame; relevant only for CDS features.</td>
<td>.</td>
</tr>
</tbody>
</table>

**Attributes**

<table>
<thead>
<tr>
<th>Column/Attribute tag name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Unique indel id. Non-sequential ids are due to filtering.</td>
<td>21</td>
</tr>
<tr>
<td>ins_len</td>
<td>Number of bases inserted relative to the reference.</td>
<td>7</td>
</tr>
<tr>
<td>del_len</td>
<td>Number of bases missing from the reference.</td>
<td>2</td>
</tr>
<tr>
<td>len</td>
<td>The size of the indel, if both an insertion and a deletion are detected. Deletions are indicated by a negative value.</td>
<td>7, –2</td>
</tr>
</tbody>
</table>
### Table 62 Small indel GFF file format descriptions  (continued)

<table>
<thead>
<tr>
<th>Column/Attribute tag name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>clustered-indel-sizes</td>
<td>The average indel size of each cluster rounded to the nearest integer. Deletions are indicated by a negative value.</td>
<td>4, -298</td>
</tr>
<tr>
<td>reference</td>
<td>The reference sequence at the position of the allele call. An event that is an insertion and is not a substitution, is indicated with a -.</td>
<td>ag</td>
</tr>
<tr>
<td>allele-call</td>
<td>The allele sequences detected. The reference sequence is only present if sufficient reads span the first breakpoint of the indel. The string ag/- indicates a hemizygous deletion where one allele is the reference, while -/- indicates a homozygous deletion. A third allele is possible if sufficient reads indicate its presence.</td>
<td>ag/</td>
</tr>
<tr>
<td>context-pos</td>
<td>Position of the first base of the sequences reported with additional surrounding sequence context. If the first allele is missing, it is the position immediately before the insertion.</td>
<td>713660</td>
</tr>
<tr>
<td>context-reference-seq</td>
<td>Reference sequence at the context position.</td>
<td>acagagagaag</td>
</tr>
<tr>
<td>context-variant-seq</td>
<td>A complete list of non-reference variant sequences found by all gap alignments in the pileup. Using the ambiguity information from the aligner (XA tag), this contains the context around the indel representing a possible short tandem repeat. &quot;NO_CALL&quot; indicates reads where a misalignment is likely (that is, color-space alignment is not in agreement with base-space alignment).</td>
<td>aCAGAGAAG/ acagagagaag</td>
</tr>
<tr>
<td>reference-reads</td>
<td>Number of reads detected to span the first indel breakpoint location by 5bp.</td>
<td>5</td>
</tr>
<tr>
<td>context-variant-reads</td>
<td>The number of indel reads found for each of the above variants. If the number of gap alignments found is greater than the parameter small.indel.num.aligns.per.pileup, then an estimated counts is displayed followed by the read counts of the sample in parentheses.</td>
<td>3,1 15464,1497(899,87)</td>
</tr>
<tr>
<td>gap-total-reads</td>
<td>Total number of gap alignments in the pileup.</td>
<td>4</td>
</tr>
<tr>
<td>gap-nonred-reads</td>
<td>Total number of reads that are not marked as PCR duplicates in the BAM file.</td>
<td>4</td>
</tr>
<tr>
<td>tight_chrom_pos</td>
<td>Conservative estimate of the chromosome start position range of the feature.</td>
<td>713662–713667</td>
</tr>
<tr>
<td>loose_chrom_pos</td>
<td>Estimate of the maximum chromosome start position range of the feature.</td>
<td>713662–713667</td>
</tr>
</tbody>
</table>
### Table 62 Small indel GFF file format descriptions  (continued)

<table>
<thead>
<tr>
<th>Column/Attribute tag name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>no_nonred_reads</td>
<td>Number of reads with unique start positions (non-redundant reads).</td>
<td>3</td>
</tr>
<tr>
<td>coverage_ratio</td>
<td>Clipped ungapped coverage divided by the number of non-redundant gap alignment reads.</td>
<td>1.6667</td>
</tr>
<tr>
<td></td>
<td>Clipped coverage means the last 5 bp at either end are not counted as a part of coverage.</td>
<td></td>
</tr>
<tr>
<td>zygosity</td>
<td>Experimental zygosity call. Allowed values:</td>
<td>HEMIZYGOUS-REF</td>
</tr>
<tr>
<td></td>
<td>- <strong>HEMIZYGOUS-REF</strong>: enough ungapped alignments overlap the indel break point to indicate that the reference allele is present. Also, all the indels have the same size and the same sequence.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- <strong>HOMOZYGOUS-NON-REF</strong>: occurs when there is no indication that the reference allele is present. Also, all the indels have the same size and the same sequence.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- <strong>HEMIZYGOUS-REF-MULTI-INDEL-ALLELE</strong>: occurs if the reference is detected, the variant has a single clustered indel size, but multiple variant sequences are detected.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- <strong>HETEROZYGOUS-NON-REF</strong>: occurs if the reference is not detected, the variant has a single clustered indel size, but multiple variant sequences are detected.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- <strong>MULTI-HEMIZYGOUS-REF</strong>: occurs if the reference, multiple clustered indel sizes, and multiple variant sequences are all detected.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- <strong>HEMIZYGOUS-NON-REF</strong>: occurs if the reference is not detected, but multiple clustered indel sizes and multiple variant sequences are.</td>
<td></td>
</tr>
<tr>
<td>zygosity-score</td>
<td>Experimental zygosity score. The range is 0–1. The closer to 1, the more likely the zygosity is hemizygous with the reference allele, and the closer to 0, the less likely that it is.</td>
<td>0.9656</td>
</tr>
<tr>
<td>rg_id_nums</td>
<td>The list of runIdNum's that correspond to the ##@HD header line of the GFF file. This information indicates which input BAM file the particular read came from.</td>
<td>1,1,2,1</td>
</tr>
<tr>
<td>bead_ids</td>
<td>Bead IDs for each read. The run_names and all the comma-separated attributes below are in the same order as these.</td>
<td>984_536_1054, 1431_2007_1567, 116_364_1582</td>
</tr>
<tr>
<td>overall_qvs</td>
<td>Alignment quality values for each bead.</td>
<td>26,47,66</td>
</tr>
<tr>
<td>no_mismatches</td>
<td>List of number of mismatches for each read.</td>
<td>-1,-1,-1</td>
</tr>
<tr>
<td>read_pos</td>
<td>Position in each read at which there was a gap in the alignment.</td>
<td>20,17,10</td>
</tr>
</tbody>
</table>
Table 62 Small indel GFF file format descriptions  (continued)

<table>
<thead>
<tr>
<th>Column/Attribute tag name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>from_end_pos</td>
<td>Same as above, except that the value is the number of base pairs from the end of the read.</td>
<td>30,33,40</td>
</tr>
<tr>
<td>strands</td>
<td>Strand for each read.</td>
<td>+,+,+</td>
</tr>
<tr>
<td>tags</td>
<td>Tags where the indel was found. Possible values are F3, R3, and FRA6.</td>
<td>F3,F3,F3</td>
</tr>
<tr>
<td>indel_sizes</td>
<td>List of indel sizes found for each evidence.</td>
<td>-2,-2,-2</td>
</tr>
<tr>
<td>non_indel_no_mismatches</td>
<td>Number of mismatches of the other tag that was matched without a gap. Values of NIL occur if that particular bead-id is from a fragment analysis.</td>
<td>1,3,3</td>
</tr>
<tr>
<td>unmatched-lengths</td>
<td>For a particular bead-id, the length of the read that was left unmatched (equal to the read length if no ungapped match was found). This read length is relevant in extended read alignments.</td>
<td>50,50,50</td>
</tr>
<tr>
<td>ave-unmatched</td>
<td>Average of the unmatched lengths.</td>
<td>50</td>
</tr>
<tr>
<td>anchor-match-lengths</td>
<td>Anchor tag’s match lengths in extended read alignments. If the detail level is not set to 9, some of these values may be reported as 99.</td>
<td>49,44,49</td>
</tr>
<tr>
<td>ave-anchor-length</td>
<td>Average of the anchor match lengths.</td>
<td>47.3333</td>
</tr>
<tr>
<td>read_seqs</td>
<td>The read sequence where the indel was found for each bead.</td>
<td>T302000, T120320, T232132</td>
</tr>
<tr>
<td>base_qvs</td>
<td>The color call QVs for the read sequence where the indel was found. This value is not currently displayed.</td>
<td>—</td>
</tr>
<tr>
<td>non_indel_seqs</td>
<td>Sequences of the non-indel anchor tag.</td>
<td>G100220, G313000, G203330</td>
</tr>
<tr>
<td>non_indel_qvs</td>
<td>The color call QVs for the non-indel anchor tag sequence. This value is not currently displayed.</td>
<td>—</td>
</tr>
</tbody>
</table>

Example output file  A composite example of a small indel output file is shown below. The file contents are as follows:

1. The general GFF version 3 headers.
2. The BAM header and read group information.
3. Lines containing information about each indel call.

Table 62 on page 242 describes each tag of the attributes column. The following is an example header:

```bash
##gff-version 3
##solid-gff-version 0.3
##source-version LifeScope(tm) 2.0 SmallIndel
##type DNA
##date 2011-04-09
##time 21:16:33
```
Chapter 13 Run a Small Indels Analysis
Small indel module output file formats

```bash
##feature-ontology http://song.cvs.sourceforge.net/*checkout*/song/ontology/sofa.obo
##options reference=/reference/human_hg18.fa output.dir=. output.prefix=baseFilename
bam.file=output/bam/bamA.bam,output/bam/bamB.bam,output/bam/bamC.bam
##reference /reference/human_hg18.fa
##input-files output/bam/bamA.bam,output/bam/bamB.bam,output/bam/bamC.bam
##run-path /path/of/ini/ini-filename/
##genomic-region chr1
##BAM header:
##@HD VN:1.0 SO:2
##@RG rg_id_num:1 LT:MP ID:2010062904031402 SM:HuRef LB:50x50MP
...  
##BAM-input='output/bam/bamB.bam' bam_file_num:2
##@HD VN:1.0 SO:2
##@RG rg_id_num:2 LT:MP ID:20100617182840485 SM:HuRef LB:50x50MP
...  
##BAM-input='output/bam/bamC.bam' bam_file_num:3
##@HD VN:1.0 SO:2
##@RG rg_id_num:3 LT:MP ID:20100627170940231 SM:HuRef LB:50x50MP
...  
##Annotation Apr 13, 2011; LifeScope_resources//lifetech/hg18/GTF/refGene.20090513.gtf;
LifeScope_resources/lifetech/hg18/dbSNP/dbSNP_b130.tab
##dbSNP function codes: Locus region = 1; Coding = 2; Coding-synon = 3; Coding-nonsynon = 4;
mRNA-UTR = 5; Intron = 6; Splice-site = 7; Contig-reference = 8; Coding-exception = 9;
NearGene-3 = 13; NearGene-5 = 15; Cod
ng-nonsynonymous nonsense = 41; Coding-nonsynonymous missense = 42; Coding-nonsynonymous
frameshift = 44; Coding-nonsynonymous cds-indel = 45; UTR-3 = 53; UTR-5 = 55; Splice-3 = 73;
Splice-3 = 75.
##Filtering no filtering.

<table>
<thead>
<tr>
<th>Chr</th>
<th>Source Type</th>
<th>Pos_Start</th>
<th>Pos_End</th>
<th>Strand</th>
<th>Phase</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>chr9</td>
<td>AB_SOLiD Small Indel Tool insertion_site</td>
<td>127466840</td>
<td>127466840</td>
<td>1.0000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Base space example:
Chapter 13 Run a Small Indels Analysis
Small indel module output file formats

13

nums=1_1;bead_ids=705_312_1080,585_1149_1247,486_211_1170,674_1449_36,1453_413_82;overall_qv
s=100,78,82,22;no_mismatches=3,4;read_pos=24,29,27,26,39;from_end_pos=36,31,33,34,21;strands
=-;tags=R3,F3;indel_sizes=21,15,-369;non_indel_no_mismatches=0,1;unmatched-lengths=0;ave-unm
atched=0.0000;anchor-match-lengths=59;ave-anchor-length=59.0000;read_seqs=AAACAAGGCAAGAAAAAG
AAAAGGAAAAAAAAAAAAAAAAAATCTAGAAACAAAAANAAAA,ACAAGAGAACAAGGCAAGAAAAAGAAAAGGAAAAAAATCTCCCNNNAA
AAAAAAAAAAAAA,ACAAGAGAACAAGGCAAGAAAAAGAAAAGGAAAAAAATCTCTTCGAAAAGAAAGACCAAAA,ACAAGAGAACAAGGCA
AGAAAAAGAAAAGGAAAAAAATCTCCGGGAAAAAAAAAAAAAAAA,AAATTTTTTGTAAAAATCATCTCCGTCCCAAAACAGTACAACCGGG
ATAAAAAAAAAAAAA;base_qvs=;non_indel_seqs=A22120313233321111113200002133002321231233030300300
2012032330,A030011330311130122031001300310313102220220333323012013020000,A333030221113212232
223202202123113120320222202210220210232021;non_indel_qvs=;rsID=72266485,72010320,56370851;fu

Color space example:
chr1
AB_SOLiD Small Indel Tool
deletion
4865598 4865605 1.0000 .
.
ID=3743;del_len=8;clustered-indel-sizes=-8;allele-call-pos=4865598;reference=CTCCACAGC;allel
e-call=CTCCACAGC/A;rightmost-allele-call-pos=4865598;rightmost-reference=CTCCACAGC;rightmost
-allele-call=CTCCACAGC/A;context-pos=4865596;context-reference-seq=GGCTCCACAGCT;context-vari
ant-seq=GGAT;reference-reads=17;context-variant-reads=8;context-variant-nonPCRdup-reads=8;ga
p-total-reads=8;gap-nonred-reads=7;gap-nonPCRdup-reads=8;tight_chrom_pos=4865598-4865598;loo
se_chrom_pos=4865598-4865598;coverage_ratio=2.4286;zygosity=HEMIZYGOUS-REF;zygosity-score=0.
9450;rg_id_nums=3_3,4_4,1_1,5_5,2_2;bead_ids=1043_1773_368,1779_878_646,979_899_1763,1753_11
03_184,1838_1695_1151,650_1885_304,1389_2001_1593,2219_1326_1756;overall_qvs=72,51,54,36;no_
mismatches=4,5;read_pos=11,20,21,22,32;from_end_pos=39,30,29,28,18;strands=-;tags=F3,R3;inde
l_sizes=-8;non_indel_no_mismatches=0,-1;unmatched-lengths=0;ave-unmatched=0.0000;anchor-matc
h-lengths=49,99;ave-anchor-length=67.7500;read_seqs=T020001220113200201111011100111122220003
23100111220,T10102101132000122011320020111101110011112222000323,G000102101132000122011320020
11110111001111212200032,G00010210113200012201132002011110111001111222200032,T320010210113200
01220113200201111011100111122220003,T22302021100200102101132000122011320020111101110011;base
_qvs=;non_indel_seqs=G00100020131231013223231211310101000313020212312010,,T11131231001013213
121331131333102331130110010213111,G02131022300102102201123122230102201100032000122022,G20211
023021003013123202010031022220210123101021322;non_indel_qvs=;rsID=34028970

Interpreted example output
This section gives examples of how a few output snippets are interpreted.

Example 1: A two-base insertion
The output below shows that an insertion of GT occurs after position 32658062:
chr1 AB_SOLiD Small Indel Tool insertion_site 32658062 32658062 1.0000 . .
ID=4;ins_len=2;clu stered-indel-sizes=2;allele-call-pos=32658062;reference=-;allele-call=GT/
GT;rightmost-allele -call-pos=32658062;rightmost-reference=-;rightmost-allele-call=GT/GT;con
text-pos=32658060;co ntext-reference-seq=GGG;context-variant-seq=GGGGT;reference-reads=0;con
text-variant-reads=2; context-variant-nonPCRdup-reads=2;gap-total-reads=2;gap-nonred-reads=
2;gap-nonPCRdup-reads=2 ;tight_chrom_pos=32658059-32658062;loose_chrom_pos=32658059-32658062
;coverage_ratio=0.0000; zygosity=HOMOZYGOUS-NON-REF;zygosity-score=0.0039;rg_id_nums=1_1;bea

Example 2: A four-base insertion replaces one reference base
The output below shows that an insertion ATCT replaces the reference base G at
position 34725658. Note the following fields:
• The field ins_len=3 shows a net gain of 3 bases (4 bases inserted, 1 replaced).
• The field reference=G shows the reference base that is lost due to the insertion.
chr6 AB_SOLiD Small Indel Tool insertion_site 34725658 34725658 1.0000 . .

LifeScope™ Genomic Analysis Software Command Shell User Guide

247


Example 3: A heterozygous insertion

The reference sequence AGGGGG at position 7002407 has a single base insertion of A after position 7002412. The other allele of this heterozygous indel is GAAAAAA at position 7002407. Note the following fields:

- The field ins_len=1 shows one base is inserted.
- The field allele-call=AGGGGGA/GAAAAAA shows that this insertion is heterozygous.
- Comparing the reference field with the first strand of the allele call shows that A is inserted as the 7th base:
  - reference=AGGGGG
  - allele-call=AGGGGGA
- The second strand of the allele call field shows how that strand differs from reference (GAAAAAA compared to AGGGGG).

chr18 AB_SOLiD Small Indel Tool insertion_site 7002407 7002407 1.0000 . .
ID=256;ins_len=1;clustered-indel-sizes=1;allele-call-pos=7002407;reference=AGGGGG;allele-call=AGGGGGA/GAAAAAA;rightmost-allele-call-pos=7002407;rightmost-reference=AGGGGG;rightmost-allele-call=AGGGGGA/GAAAAAA;context-pos=7002407;context-reference-seq=AGGGGG;context-variant-seq=AGGGGGA/GAAAAAA;reference-reads=1;context-variant-reads=1;context-variant-nonPCRdup-reads=1;gap-total-reads=2;gap-nonred-reads=2;gap-nonPCRdup-reads=2;tight_chrom_pos=7002412-7002412;loose_chrom_pos=7002406-7002412;coverage_ratio=0.5000;zygosity=HETEROZYGOUS-NON-REF;zygosity-score=0.0113

Small indel VCF format

A VCF file is in the Variant Call Format, which is a text format. The file contains meta-information lines, a header line, and data lines that each contain information about a position in the genome. This file format was developed as part of the 1000 Genomes project, and is described at this site:

http://vcftools.sourceforge.net/specs.html

The VCF and GFF output files contain the same contents, in a different file format.

The calculation of the genotype quality (GQ) field is calculated from the zygosity score of the indel call. The zygosity score is a multivariate descriptor of genotype quality, and has a score from 0 to 1 (see also “zygosity-score” on page 244 in Table 63). A good homozygote allele has a zygosity score close to 0, and a good heterozygote allele has a score close to 1. An ambiguous call has a score around 0.5. The following formula transforms this function trigonometrically to the 0 to 1 scale:

\[
GQ = 50 \times (1 - \sin(3.14 \times \text{Zygosity Score}))
\]

The following fields have a one-to-one correspondence in the GFF and VCF files:

<table>
<thead>
<tr>
<th>GFF field</th>
<th>VCF field</th>
</tr>
</thead>
<tbody>
<tr>
<td>reference-reads</td>
<td>REF-DP</td>
</tr>
<tr>
<td>gap-total-reads</td>
<td>GAP-DP</td>
</tr>
</tbody>
</table>
The possible Others GFF field is ignored when generating the ALT allele field in the VCF file.

**VCF format example**

The following is a truncated example of VCF file contents. Fields are described in the FORMAT lines.

```plaintext
##fileformat=VCFv4.1
##fileDate=20111011
##source="LifeScope(tm) 2.5 SmallIndel"
##reference=/share/reference/genomes/HUMAN_hg19/human_hg19.fa
##contig=<ID=chr1>
##contig=<ID=chr2>
##contig=<ID=chr3>
...
##contig=<ID=chr22>
##contig=<ID=chrX>
##INFO=<ID=DP,Number=1,Type=Integer,Description="Total Depth">
##INFO=<ID=REF-DP,Number=1,Type=Integer,Description="Reads that span reference">
##INFO=<ID=GAP-DP,Number=1,Type=Integer,Description="Reads that span a gap on the region">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Float,Description="Genotype Quality">
#CHROM POS ID REF ALT QUAL FILTER INFO FORMAT Unknown
chr1 1110990 . CG CCC . PASS DP=161;REF-DP=140;GAP-DP=21 GT:GQ 0/1:49
chr1 1276973 . G GACAC . PASS DP=14;REF-DP=3;GAP-DP=11 GT:GQ 1/1:50
chr1 1289367 . CTG C . PASS DP=18;REF-DP=0;GAP-DP=18 GT:GQ 1/1:50
chr1 6170424 . CG C . PASS DP=45;REF-DP=18;GAP-DP=27 GT:GQ 1/1:41
chr1 6252965 . CCA C . PASS DP=154;REF-DP=138;GAP-DP=16 GT:GQ 0/1:49
```

**Small indel TXT and SQL formats**

The GFF and VCF files serve as the primary output format for the small indel caller. However, the TXT output file is an alternative format of the data contained in the GFF file. The information contained in the TXT file is generally only a subset of that of the GFF. The SQL file contains a CREATE TABLE command for MySQL, and the contents of the SQL file can be imported into MySQL.

Table 63 describes specifically what information the small indel TXT file contains.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>chrom</td>
<td>Chromosome number of indel.</td>
<td>1</td>
</tr>
<tr>
<td>min-chrom-pos</td>
<td>Start position of the indel.</td>
<td>713662</td>
</tr>
<tr>
<td>max-chrom-pos</td>
<td>End position of the indel.</td>
<td>713663</td>
</tr>
<tr>
<td>called-range</td>
<td>Range of chromosome position range of the feature.</td>
<td>713661–713661</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This ambiguity is resolved in the final indel call of the GFF.</td>
<td></td>
</tr>
<tr>
<td>tight-range</td>
<td>Conservative estimate of chromosome position range of the feature.</td>
<td>713662–713667</td>
</tr>
<tr>
<td>loose-range</td>
<td>Estimate of the maximum chromosome position range of the feature.</td>
<td>713662–713667</td>
</tr>
<tr>
<td>num-pos-strand</td>
<td>Number of reads that were mapped to the positive strand.</td>
<td>3</td>
</tr>
<tr>
<td>num-neg-strand</td>
<td>Number of reads that were mapped to the negative strand.</td>
<td>0</td>
</tr>
<tr>
<td>num-r3-hits</td>
<td>Number of reads where the indel was found on the R3 or F5 tag.</td>
<td>0</td>
</tr>
<tr>
<td>num-f3-hits</td>
<td>Number of reads where the indel was found in the F3 tag.</td>
<td>3</td>
</tr>
</tbody>
</table>
### Table 63 Small indel TXT file format description (continued)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>num-frag-hits</td>
<td>Number of reads where the indel was found from a Fragment tag.</td>
<td>0</td>
</tr>
<tr>
<td>indel-type</td>
<td>The type of indel: INSERTION, DELETION, or COMBINATION. For COMBINATION, there were reads indicating both an insertion and a deletion, and no indel size was called.</td>
<td>DELETION</td>
</tr>
<tr>
<td>unique-indel-size</td>
<td>Called indel size.</td>
<td>–2</td>
</tr>
<tr>
<td>indel-size range</td>
<td>The indel sizes reported from each of the reads.</td>
<td>–2 to –2</td>
</tr>
<tr>
<td>num-uniq-align</td>
<td>Number of non-redundant alignments in the pileup.</td>
<td>3</td>
</tr>
<tr>
<td>num-tot-align</td>
<td>Total number of reads in the pileup.</td>
<td>3</td>
</tr>
<tr>
<td>average-read-position</td>
<td>Average read position where the gap occurred.</td>
<td>15.6667</td>
</tr>
<tr>
<td>ave-from-end-read-position</td>
<td>Average from end read position where the gap occurred.</td>
<td>34.3333</td>
</tr>
<tr>
<td>indel-read-pos-list</td>
<td>List of read positions where gap occurred.</td>
<td>20;17;10</td>
</tr>
<tr>
<td>dbsnp-indel</td>
<td>Legacy. Not used.</td>
<td>—</td>
</tr>
<tr>
<td>uw-hgsv-indel</td>
<td>Legacy. Not used.</td>
<td>—</td>
</tr>
<tr>
<td>read-lengths</td>
<td>Lengths of the reads (full read sequence, not the extended match length).</td>
<td>50;50;50</td>
</tr>
<tr>
<td>paired-distances</td>
<td>The clone sizes of each of the bead pairs (NIL for fragments).</td>
<td>1426;1370;1454</td>
</tr>
<tr>
<td>ave-pair-dist</td>
<td>Average of the paired distances.</td>
<td>1416.667</td>
</tr>
<tr>
<td>tags-R3-F3</td>
<td>Tags where the indel was found. Possible values are F3, R3, and FRAG.</td>
<td>F3;F3;F3</td>
</tr>
<tr>
<td>chrom-pos-s</td>
<td>Chromosome positions of the indel tag’s match location.</td>
<td>713641;713644;713651</td>
</tr>
<tr>
<td>strands</td>
<td>Strand for each bead id.</td>
<td>+;+;+</td>
</tr>
<tr>
<td>indel-sizes</td>
<td>List of indel sizes found for each read.</td>
<td>–2;–2;–2</td>
</tr>
<tr>
<td>nums-mismatches</td>
<td>List of number of mismatches for each read.</td>
<td>–1;–1;–1</td>
</tr>
<tr>
<td>ave-numb-mismatches</td>
<td>Average of the number of mismatches found in the indel tag.</td>
<td>–1</td>
</tr>
<tr>
<td>indel-lower-pos-s</td>
<td>List of lower ranges of the indel.</td>
<td>20;17;10</td>
</tr>
<tr>
<td>indel-upper-pos-s</td>
<td>List of upper ranges of the indel. The lower and upper ranges represent the ambiguity of the gap alignment, for example, AT/-- in the context of ATAT/AT.</td>
<td>25;22;15</td>
</tr>
<tr>
<td>run-names</td>
<td>Run names [one for each input file] for each read. For BAM files, the 1 in 50_1_r is the runIdNum in the ###@HD header line of the GFF file.</td>
<td>L1_1_50_1_r;L1_1_50_1_r;L1_1_50_1_r</td>
</tr>
<tr>
<td>clone-ids</td>
<td>Bead IDs for each read.</td>
<td>984_536_1054;1431_2007_1567;116_364_1582</td>
</tr>
<tr>
<td>read-seqs</td>
<td>The read sequences for each read.</td>
<td>T302000;T120320;T232132</td>
</tr>
<tr>
<td>ref-allele</td>
<td>Reference allele.</td>
<td>acagagagaag</td>
</tr>
</tbody>
</table>
Example TXT file content

The following is an example of small indel text file output. See Table 63 on page 249 for an explanation of the fields in this file.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>var-allele1</td>
<td>Most common variant allele (if it passes the color error correction, otherwise it is NULL).</td>
<td>aCAGAGAAG</td>
</tr>
<tr>
<td>other-var-alleles</td>
<td>Other alleles of the variant, if present.</td>
<td></td>
</tr>
<tr>
<td>var-counts1</td>
<td>Number of reads that have the most common indel allele.</td>
<td>3</td>
</tr>
<tr>
<td>other-var-counts</td>
<td>The list of the other allele counts.</td>
<td>NO_CALL</td>
</tr>
<tr>
<td>ungapped-unmatched-lengths</td>
<td>For a particular bead-id, the length of the read that was left unmatched (equal to the read length if no ungapped match was found or if the detail level is not set to 9).</td>
<td>50,50,50</td>
</tr>
<tr>
<td>ave-ungapped-unmatched</td>
<td>Average of the ungapped, unmatched lengths.</td>
<td>50</td>
</tr>
<tr>
<td>anchor-match-lengths</td>
<td>Anchor tags match length.</td>
<td>49,44,49</td>
</tr>
<tr>
<td>ave-anchor-length</td>
<td>The average of the anchor match lengths.</td>
<td>47.3333</td>
</tr>
<tr>
<td>coverage-ratios</td>
<td>Clipped normal coverage/number of non-redundant reads.</td>
<td>1.6667</td>
</tr>
<tr>
<td>zygosity-call</td>
<td>Experimental zygosity call.</td>
<td>HEMIZYGOUS</td>
</tr>
<tr>
<td>zygosity-p-value</td>
<td>Experimental zygosity score.</td>
<td>0.9656</td>
</tr>
</tbody>
</table>

Small indel ALIGN format

The ALIGN file contains a text visual of each pileup that makes an indel call. Below is an example record from this file, with approximately 30 base pairs trimmed from each side for clarity.

```
> 64,chr1:16765166-16765167(),DELETION,-2;,allele-call-pos=16765166;allele-call=CT;/allele-pos=16765164;alleles=CACTCTGA/CACTG
A;allele-counts=REF,15
CGAAGGGGTCAAGGACACTCTGAGTTAGTG 16765118 16765208
32020001210020211221221221032113 CACTCTGA 16765164 16765170
```

Table 63 Small indel TXT file format description (continued)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>var-allele1</td>
<td>Most common variant allele (if it passes the color error correction, otherwise it is NULL).</td>
<td>aCAGAGAAG</td>
</tr>
<tr>
<td>other-var-alleles</td>
<td>Other alleles of the variant, if present.</td>
<td></td>
</tr>
<tr>
<td>var-counts1</td>
<td>Number of reads that have the most common indel allele.</td>
<td>3</td>
</tr>
<tr>
<td>other-var-counts</td>
<td>The list of the other allele counts.</td>
<td>NO_CALL</td>
</tr>
<tr>
<td>ungapped-unmatched-lengths</td>
<td>For a particular bead-id, the length of the read that was left unmatched (equal to the read length if no ungapped match was found or if the detail level is not set to 9).</td>
<td>50,50,50</td>
</tr>
<tr>
<td>ave-ungapped-unmatched</td>
<td>Average of the ungapped, unmatched lengths.</td>
<td>50</td>
</tr>
<tr>
<td>anchor-match-lengths</td>
<td>Anchor tags match length.</td>
<td>49,44,49</td>
</tr>
<tr>
<td>ave-anchor-length</td>
<td>The average of the anchor match lengths.</td>
<td>47.3333</td>
</tr>
<tr>
<td>coverage-ratios</td>
<td>Clipped normal coverage/number of non-redundant reads.</td>
<td>1.6667</td>
</tr>
<tr>
<td>zygosity-call</td>
<td>Experimental zygosity call.</td>
<td>HEMIZYGOUS</td>
</tr>
<tr>
<td>zygosity-p-value</td>
<td>Experimental zygosity score.</td>
<td>0.9656</td>
</tr>
</tbody>
</table>
Chapter 13  Run a Small Indels Analysis

Small indel module output file formats

### Table 64 Small indel ALIGN file format

<table>
<thead>
<tr>
<th>Line</th>
<th>Name</th>
<th>Format and description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Indel</td>
<td>&gt; ID,chr#:start-end[],INDEL_TYPE,INDEL_SIZE. The rest of the line contains several tags that are in common with the GFF file.</td>
</tr>
<tr>
<td>2-5</td>
<td>Reference</td>
<td>REFERENCE_SEQUENCE START_POS END_POS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reference sequence in base and color space for entire pileup region, and region around the indel. The coordinates are given after the sequence.</td>
</tr>
<tr>
<td>6+</td>
<td>Reads</td>
<td>COLOR_SPACE_READ REF_COLORS/READ_COLORS+ REF_BASES/READ_BASES</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Contains all the reads of the pileup, extracted colors around the indel, and the corresponding bases. The sign after the extracted colors indicates color space compatibility (+) or incompatibility (-).</td>
</tr>
</tbody>
</table>

### UNGAPPED and PAS.SUM formats

The UNGAPPED file contains more details of the reads counted towards the number of ungapped alignments used for the coverage ratio. For every pileup there are as many header lines as number of BAM files as input. The header line has this format:

```
# INDEL_ID BAM_CHROM_NUM POS1 POS2
```

Each line after the header is a list of bead ids along with properties of that bead extracted from the BAM file.

The PAS.SUM file is an intermediate the pileup file extracted from the BAM file. The file starts with BAM header information from each of the inputs. Following that, each pileup has a separate line. An example and format are given in Table 65.

### Table 65 Column descriptions of the PAS SUM file

<table>
<thead>
<tr>
<th>Name</th>
<th>Components</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position, id, and</td>
<td>[CHROM_NUM]:{POS}-</td>
<td>7:315505-315512:347316:2</td>
</tr>
<tr>
<td>ungapped</td>
<td>{POS2}:{INDEL_ID}:{NUM_UNGAPPED_READS}</td>
<td></td>
</tr>
</tbody>
</table>
Gap alignment mapping

Detection of indels variants with a split-read technique is achieved by using LifeScope™ Software's small indel caller on BAM files produced from long mate-pairs, mate-pair, fragment, and pair-end library types. The combination of multiple libraries of any combination of these types is also possible. For pair-end and long mate-pair libraries, the small indel module, by default, is able to determine sizes up to 500 for deletions and 20 for insertions. For all fragment libraries, the default size range is up to 19 for deletions and up to 4 for insertions. Also, 75mers from either pair-end and fragment libraries are capable of insertions up to at least size 29. Furthermore, the module allows for detection of more complex variants such as indel-SNP combinations.
The small indels module determines high-quality calls for insertions and deletions in two stages. In the first stage, the LifeScope™ Software mapping module (see Chapter 9, “Run a Resequence Mapping Analysis” on page 129) produces gapped alignments on a bead-by-bead basis and writes them to the resulting BAM file. For all libraries, this determination occurs during the mapping phase of this using a single tag approach. For paired tag libraries, this determination additionally occurs in the pairing phase using a paired tag approach. In the second stage, the indel caller takes these gap alignments, extracts them out of the BAM file, forms pileups, filters the pileups based on certain heuristics, determines zygosity, and annotates the indel sequences. This results in concisely annotated and highly accurate indel calls.

**Figure 18** illustrates the F3 with the indel. The algorithm also determines indels in the R3/F5 tag.

![Diagram](https://via.placeholder.com/150)

**Figure 18** Gap alignment detection using the paired reads from mate-pair and paired-end data

For paired tags, the algorithm surveys only those indels with one-end anchored (OEA) mate-pairs (see Figure 18). It does so by realigning the OEA pairs using an anchor tag (that is one tag which can be aligned to the genome by itself) and by performing a more aggressive alignment with the other tag in a several kbp window (depending on the orientation of the tags and on the minimum and maximum insert sizes set in mapping’s pairing phase) around the anchored mate. A pair is considered OEA if one read of the tag fails to align or if its aligned length was not higher than a certain length threshold, specified by the parameter `pairing.gap.min.non.matched.length`. In addition, for anchor tags, only alignments above a minimum length are used (those alignments are close to full length). Finally, the anchor tag may only have a maximum number of anchor alignments (close to full length alignments) in the genome, which by default is set to 10 by the second value of `pairing.gapped.max.hits`.

Using the unanchored or non-fully extended tag, the module starts aligning both ends of the read localized by the other tag’s match location. With the region of the genome determined by this match location and insert size distribution, the module makes a catalog of end locations by extending the read starting with a minimum value (specified by `pairing.gap.edge.length.insertions` and `pairing.gap.edge.length.deletions` parameters rescue for insertions and deletions, respectively) until the alignment hits the maximum of number of mismatches allowed. These maximums are specified by these parameters:

- `pairing.anchor.max.mismatches`: the number of mismatches in both tags
• pairing.gap.max.mismatches.tag1: mismatches for the F3 tag
• pairing.gap.max.mismatches.tag2: mismatches for the R3 or F5 tag

The default value for the single tag number of mismatches is 5 for read lengths 50 and higher (50 set by min.length.for.aggressive.gapped.mapping), and 2 for shorter read lengths. Then the single tag number of mismatches for paired-end libraries is 5 and 2 by default for the F3 and F5 tags, respectively.

With this catalog, the process attempts to join the ends of the read to find a single gapped alignment within a certain size range. The maximum gap size allowed depends on parameters specified in the mapping’s pairing phase parameters. Furthermore, it identifies it as a gapped alignment if the above joining could be done with the fewest number of mismatches. Ambiguity of the location of this joining is common, mainly due to the presence of short tandem repeats. This is represented by the XA field in the BAM file. Later on downstream, the indel caller (as described later) uses the consensus of reads, produces a left most, concise representation of the variant. A rightmost representation is also produced. Furthermore, gap alignments must also satisfy a minimum edge length which is the number of base pairs before the end of the read.

For determining small indels (deletions to size 11 and insertions to size 4), the algorithm disallows indels within 3 bases from either end of the read. The module identifies if it is able to piece together both ends, allowing only for a single gap of up to 4 base pairs inserted (present in read but not in reference), or up to 11 base pairs deleted. Larger indels up to size 500 are also found using different edge length criteria. These sizes and criteria are specified in several mapping parameters as shown in the Table 66. For example, the default value for pairing.gap.max.deletion.size is 0:11:0:0:500, corresponding to the first row of Table 66. The values that effect gap alignments are the second through fourth ones, and are specified in the table as small indels, medium insertions, larger insertions, and larger deletions, respectively. s1 and s2 are specific tag lengths of the library used. For example, the parameters pairing.gap.max.deletion.size.75-35 and pairing.gap.max.insertion.size.75-35 affect a 75x35 PE library, but not other size libraries.

Larger indels are found for 60-60, 50-50, and 75-35 read length libraries, but not found for smaller 50-35, 50-25, 35-35, or 25-25 read length libraries.

Table 66  Parameters controlling gap size ranges, with default values shown

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Small indels</th>
<th>Medium insertions</th>
<th>Larger insertions</th>
<th>Larger deletions</th>
</tr>
</thead>
<tbody>
<tr>
<td>pairing.gap.max.deletion.size</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>500</td>
</tr>
<tr>
<td>pairing.gap.max.deletion.size.s1-s2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>pairing.gap.max.insertion.size</td>
<td>4</td>
<td>15</td>
<td>21</td>
<td>0</td>
</tr>
<tr>
<td>pairing.gap.max.insertion.size.s1-s2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>pairing.gap.edge.length.deletions</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>20</td>
</tr>
<tr>
<td>pairing.gap.edge.length.deletions.s1-s2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>pairing.gap.edge.length.insertions</td>
<td>3</td>
<td>14</td>
<td>14</td>
<td>0</td>
</tr>
<tr>
<td>pairing.gap.edge.length.insertions.s1-s2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
### Single tag approach (for both paired and fragment libraries)

1. Take reads that in the first stage of mapping do not align, or align only partially.
2. Align first 20 bases to the genome.
3. Gap align full read around the 20mer match.
4. Starting at read position 30, align 20 bases to the genome, and repeat step 2.

**Figure 19** Global full read gap alignment detection using the single read technique used in all libraries.

1. Take reads that in the first stage of mapping do not align, or align only partially.
2. Align first 20 bases to the genome.
3. Take initial alignment and extend requiring a gap in the alignment.
4. Starting at read position 30, align 20 bases to the genome, and repeat step 2.

**Figure 20** Local gap alignment detection using the single read technique used in all libraries. Reads here may not necessarily extended to the full read length.

Using only a single tag, gap alignments are determined in the map phase of resequencing mapping. Although gap alignments can be found along with ungapped alignments in the first stage of mapping, the process here describes the default process of finding gap alignments only in the second stage of mapping. This process is illustrated in **Figure 19** (global) and **Figure 20** (local). As with mate-pair libraries, whole genome searches for gapped alignments is prohibitively expensive. Similar to the paired tag approach, a read is only considered for this process for both local and global alignment methods if the tag fails in the first stage of mapping to align or if its alignment length did not extend past a certain length threshold, specified by the gap.
aligner parameter mapping.gap.min.non.matched.length. Then each read is localized by performing a 20 base pair ungapped alignment allowing for 1 mismatch. The 20mer taken is from both the beginning and 30 bases pairs into the read. The anchor properties are specified by the second.map.scheme.unmapped.xx mapping parameters, and for 50mers and longer, the value is 20.1.0:30.

Two algorithms are available in the system, GLOBAL (default) and LOCAL, specified by second.map.gapped.algorithm. For GLOBAL (full read) gap alignments, each 20mer alignment then defines a search region of \([A-40, A+80]\), where \(A\) is the position of the alignment. The downstream range (80) is settable using the parameter second.map.gapped.seed.window.right, and upstream (40), using second.map.gapped.seed.window.left. With this region, the same full read alignment strategy done with mate-pairs is performed: a catalog of partial begin and end read hits is formed, and an attempt is made to join them with a gap of some size. For LOCAL gap alignments, full read length matches are not required, but rather a gap is allowed during extension of the original anchor hit with a gap and mismatch penalty specified by the parameters second.map.gapped.penalty (default of 5) and second.map.mismatch.penalty (default of 2.0), respectively. Once the penalty threshold is met, the extension of the read terminates.

As in the paired approach, all gap alignments are restricted by both maximum size and minimum edge length requirements. By default, this approach finds deletions to size 19 and insertions to size 4. Larger insertions up to size 29 have been found effectively with 75mers using more restrictive parameters, which at the same time reduce sensitivity for smaller insertions and do not perform deletions. Insertions specifically at size 30 were set, but not evaluated. Both the default and larger insertion parameters are specified in the Table 67.

**Table 67** Various size and edge length requirements for the small indels and larger insertions

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Small indels (default)</th>
<th>Medium insertions (optional for 75mers)</th>
</tr>
</thead>
<tbody>
<tr>
<td>second.map.gapped.deletion</td>
<td>19</td>
<td>0</td>
</tr>
<tr>
<td>second.map.gapped.deletion.tag1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>second.map.gapped.deletion.tag2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>second.map.gapped.insertion</td>
<td>4</td>
<td>30</td>
</tr>
<tr>
<td>second.map.gapped.insertion.tag1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>second.map.gapped.insertion.tag2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>second.map.gapped.edge.length.deletions</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>second.map.gapped.edge.length.insertions</td>
<td>12</td>
<td>18</td>
</tr>
</tbody>
</table>

**Small indel calling from gap alignments**

The small indel caller uses alignments contained in the BAM file, which in the LifeScope™ Software mapper have two general types, ones that contain a single gap in the alignment (those that have an I or D in the CIGAR string) and ones that do not, which is referred to as ungapped alignments. The caller uses a variety of algorithms to produce the indel calls present in the GFF file. These algorithms are discussed in the sections below, and are summarized in this figure.
Pileup handling

The gap alignments contained in the BAM files form the basis for calling indels. The gap alignments go through a series of processes before being reported in the final GFF output. Those alignments that have a minimum overall mapping quality (`small.indel.min.mapping.quality` by default) are extracted from the BAM file. The exception to meeting this quality threshold is the class of F3 gap alignments from pair-end libraries (see "Mapping quality adjustment (paired-end libraries only)" on page 260).

Properties of the gap alignment tag (`small.indel.min.non.matched.length`) and anchor tag (`small.indel.min.anchor.mapping.quality`) are also assessed and are affected by `small.indel.detail.level`. Only the first 6 (detail level x 2) anchor reads are considered for the default setting of 3. Alternative alignments for the non-matched length filter are considered only for a detail level of 9. Alternative alignments are those that are found without a gap at the same chromosome location. Furthermore, anchor reads are present only in paired approaches, but alternative alignments could be present in either paired or single tag approaches.

Next the gap alignments are grouped together by genomic location to form pileups of reads. Because of the positional ambiguity of indels, pileups are formed by proximity; specifically, alignments that are within 5 base pairs between consecutive evidences are combined together. For pileups that have 6 or more non-redundant reads, the sixth and additional reads after that are only grouped together if the pileup is 2 or fewer base pairs from the last gap position. The default behavior is set by the parameter `small.indel.consGroup` (1 by default). A value of 9 makes every gapped alignment into a separate pileup.

Finally, the pileup is taken if the number of non-redundant reads is between `small.indel.min.num.evid` and `small.indel.max.num.evid`, inclusively. By default, pileups with two or more reads are taken at this stage. Non-redundant reads are those reads that have unique F3 and R3/F5 positions for paired tags, or unique F3 positions for single tag analysis. Mixed single and paired tag reads are...
considered unique from each other. With each pileup extracted from gap alignments, ungapped alignments that span the gap by at least 5 base pairs are counted, and are used in the normal vs. indel coverage ratio. The module stores the pileup information extracted from the BAM file into an intermediate PAS.SUM file for further analysis.

Figure 21  Small indel caller heuristics

The pileups go through additional filters, as illustrated in Figure 21, each with corresponding optional parameters in Table 61 on page 237. The best mapping quality cutoff (specified by small.indel.min.best.mapping.quality, with a default of 10) is the cutoff for the highest overall mapping quality from the pileup. For paired tags, this cutoff is the highest pairing quality value; for fragment, the highest mapping quality; and for mixed, the highest from any of the reads of either. Filtering for the maximum value allows lower quality reads to act as supportive evidence. Ambiguous indel size filtering requires that either 75% of the alignments have the same size or that they are able to form at least one valid clustered indel size. The latter method, in the case of multiple clustered sizes, allows including non-reference hemizygote indels. This method is described in more detail in “Indel size determination” on page 262.
If only two non-redundant reads are required to make an indel call, false positives can become prevalent at higher coverage levels (for instance, 100x). By using the number of ungapped alignments queried from the BAM file, the software can determine a coverage ratio of ungapped (with 5 base pairs clipped) coverage over a number of non-redundant indel supports. A high ratio is indicative of a false positive and are filtered out at values higher than the setting of `small.indel.max.coverage.ratio` (12 by default). Even in low coverage situations, there are regions of high coverage where this filter is active and further helps to reduce false positive indel calls.

Finally, the pileup must also be comprised of reads where the majority of reads have gaps that are color space compatible. The parameter `small.indel.alignment.compatibility.filter` sets filtering based on this test, where a setting of 0 indicates no color space-based filtering and 1 filters out pileups were NO_CALL is most prevalent (1 is the default for LifeScope™ Software and recommended for max mapping). NO_CALL for the allele occurs in reads where the gap is not color space compatible.

Certain paired alignments may have poor overall mapping quality because of poor mapping of one tag. However the gap alignment of the better tag may still constitute strong evidence for indels and could be used for increased sensitivity. In order to effectively use these alignments in BAM files missing the single tag mapping quality (SM), the tool performs an adjustment to the mapping quality for the sole purpose of reducing the quality required to meet the minimum overall mapping quality (`small.indel.min.mapping.quality`) for certain otherwise high quality reads. This adjustment has no effect in either the best mapping quality filter or the reported output of quality scores in the GFF3 file. The QV adjustment follows this formula:

\[ QV_{adj} = ecl \cdot EL - mmc \cdot MM + cc \]

where `ecl`, `mmc`, and `cc` are the coefficients for edge length, number of mismatches, and constant, respectively and `EL` and `MM` are the actual minimum edge match length and number of mismatches, respectively for that read. By default this is done only for the F3 tag of a pair-end library, with `ecl` = 0.25, `mmc` = 0.50, and `cc` = 5. These settings are not readily available, as it may require editing the small indel XML file to add `small.indel.mapping.quality.adjustment.libraries.tag1` (default value of PE) and `small.indel.mapping.quality.adjustment.coefficients.tag1` (default of 0.25,0.50,5) parameters (editing the file is not recommended). This adjustment has no effect on the second (F5) tag. To turn off this adjustment, specify None for the adjustment coefficients parameter.

When an indel occurs in a sequence, and that sequence is measured using color space, the color-space sequence not only has a gap of the same size of the indel, but also leaves a signature that can indicate if there is a measurement error within the gap. This is especially important in the case of insertions when you have a small number of evidences and there is a disagreement in the bases of the inserted sequence. With methods that directly measure bases, there would be no indication, based on the inserted sequence alone, on which inserted bases is more trusted.

In color space, this signature can be used to see if the color that spans the gap is compatible with the set of colors that go through the gap. For example, the alignment AACG/A-G would be 013/2-- in color space. Here the color 2 spans the gap (measuring both A and G), while 013 goes through the gap (measuring AACG). The color 2 is compatible with the sequence 013 because both would end with a G in base space. However, an alignment of 213/2-- would not be compatible because, using the
same starting base A as the above example, 213 would measure AGTA. Because the rest of the color-space sequence beyond this would be aligned, 213/2-- would be indicative of a measurement error within the insertion if 213 is the sample, or if the sequence is the reference, the color 2 that spans the deletion. The alignment’s color space compatibility can be calculated for any sequence using color space addition. This signature for color space compatibility is used to more accurately call the inserted sequence of the insertion, important if only a small number of reads are used to call an indel. Also an entire pileup is more likely to be a false positive if most of the reads indicate a gap that is not color space compatible.

For every gap alignment, the LifeScope™ Software aligner reports the position of the gap and an ambiguity of that placement using the XA BAM field. The caller takes that information and the color-space reads to determine the base-space sequence for each of the reads, reporting the reference and all the sequences found. The module takes these calls further by taking the reference and the most common allele(s) present and then making a concise representation.

**Context sequence extraction accounting for ambiguous placement**

The small indel caller has three representations of the reference and variant sequences that contain the indel:

- **Context sequences** – Represents the sequence context around the indel.
- **Leftmost sequences** – Represents the sequence after trimming all common bases between the variant reads and the reference. Trimming goes from the right to the left first, and then left to the right.
- **Rightmost sequences** – Same as leftmost, except that trimming goes from left to right first.

The context region is specified by taking the `loose_chrom_pos` (which is contained in the output GFF), expanding the start position by 1 and the end position by 2, and then trimming the first two positions and the last two positions, only if the bases at those particular positions are identical in all of the reads and the reference. In addition, if any of this trimming process produces no bases for the reference sequence, the position is represented as that of the base immediately before the insertion.

An example of this is the following indel call that has a `loose_chrom_pos=94446948-94446955` and is annotated in dbSNP as rs58864345. The reference sequence at positions 94446948 to 94446956, and the corresponding reads is:

<table>
<thead>
<tr>
<th>Ref</th>
<th>ACTCTTCTCCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1---20220200</td>
<td></td>
</tr>
<tr>
<td>Read1</td>
<td>ACTTCTTCTCCC</td>
</tr>
<tr>
<td>Read2</td>
<td>ACTTCTTCTCCC</td>
</tr>
<tr>
<td>Read3</td>
<td>ACTTCTTCTCCC</td>
</tr>
<tr>
<td>Read4</td>
<td>ACTTCTTCTCCC</td>
</tr>
</tbody>
</table>

Because the beginning A and the ending CC are all the same, these would be trimmed off. This would result in the context sequence related tags being reported as:

```
context-pos=94446949;context-reference-seq=CTTCTTCC;context-variant-seq=CTTCTTCTTC
```

The leftmost trimming of this result is represented here:

```
CTTCTTCC Reference @ 94446949
```
Chapter 13 Run a Small Indels Analysis

Small indel calling from gap alignments

Sequence found in reads

CTTCTTCTTC

This results in the following positioning and allele call:

allele-call-pos=9446948;reference=-;allele-call=-/CTT

Since the reference here was entirely trimmed, the position reported, as per the GFF standard, is the one immediately before the CTT insertion. Finally, the rightmost trimming is represented here:

CTTCTTTC Reference @ 9446949

CTTCTTCTTC Sequence found in reads

and results in a different position:

rightmost-allele-call-pos=9446955;rightmost-reference=-;rightmost-allele-call=-/TTC

All three of these representations are present to preserve the CTT repeat context and have positioning compatible with both VCF (leftmost) and dbSNP (rightmost) positioning.

Making concise allele calls

The allele-call and rightmost-allele-call tags also represent information about the sequences detected. It always contains the most common indel variant sequence. Also, if a large enough number of ungapped alignments are present, it includes the sequence taken from the reference. Finally, if the second most common variant sequence has at least 75% of the reads as the most common, it too is included.

The following indel call from chromosome 2 at position 220,378,748 in hg18 illustrates the allele calling. The sequence context of this detected indel is:

context-reference-seq=atatttttttttttttttttttttttt attatattatttttttatttttttatttttttatttttatttttatttttttttttttttt ttattttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttt
Clusters are split recursively until any resulting cluster cannot be split further, and this process forms a list of clusters. Any cluster that has a single alignment is removed. For small indels (those less than 20 in size), the algorithm uses the single size that is present in that cluster. For a cluster of larger indels, the algorithm uses an average size found within that cluster. All cluster sizes are reported by the `clustered-indel-sizes` tag in the GFF. In addition, if there is not an indel size called with 3/4 agreement in size, then the cluster size(s), if at least one exists, is reported in the GFF as a comma-separated list after `ins_len`, `del_len`, or `len`. The first two are for when all the sizes are insertions or deletions, respectively, and the last (`len`) is used if there is both an insertion and deletion, where deletions indicated by a negative value.

Indels are also filtered based on whether these methods were successful in determining a consistent indel size or set of indel sizes, depending on the `small.indel.indel.size.distribution.allowed` setting. The default setting (`can-cluster`) requires that at least one size cluster is formed, and also allows for multiple clusters of different sizes. Overall, this method allows for more variety in the types of indels called. For sample long mate-pair and targeted resequencing paired-end datasets, this method does not significantly decrease concordance to dbSNP. A more restrictive setting of `similar-size` does not allow for this variety, by requiring that the pileup has at least 75% of the reads having the exact same size. A setting of `similar-size-any-large-deletions` allows for larger deletions to have different sizes. With `similar-size-any-large-deletions`, the caller includes additional indels that are discordant in size but have at least two reads with deletions of size 20 or higher. The setting `any` turns off filtering based on indel size, but this setting usually results in greatly increasing false indel calls.

**Reference allele calling**

The coverage ratio (described in “Pileup handling” on page 258) also serves to call an indel hemizygous, specifically the type of zygosity detected here is one where the reference is present. Because the coverage ratio is the number of ungapped alignments over the number of the gapped ones, the ratio is an indicator of this localized hemizygosity. The length scale of this hemizygous call is that of the gap size range present in the input from the aligner, mostly in the small indel range. The size range is much smaller in length scale than the whole chromosome hemizygosity present in, for example, human males for chromosome X. The software calls the presence of reference alleles by utilizing a table of coverage ratios, by utilizing a table of coverage ratios located in a file located at `<installdir>/etc/small-indels/zygosity-calibration.conf`. The table was derived by matching DH10B reads to an indel introduced reference to simulate the non-hemizygous state (indels of the same length on both alleles). For the hemizygous state, reads that occurred over a simulated indel had a 50% chance of being altered to contain that simulated indel. Because indel alignments are generally less sensitive, hemizygous situations occur frequently above coverage ratios values above 1. Different situations were simulated and are available using the parameter `small.indel.zygosity.profile.name`. The DH10B data comes from a 50mer long mate-pair sequencing run, using both tags for the paired approach and using only the F3 tag for a single tag approach.

**Zygosity Calling**

Determination of the presence of the reference allele is analogous to heterozygous calling in SNPs where one allele is the reference. However, indels have greater complexity, because a single indel can have different inserted basepair sequences or it can have different gap sizes. For example, an indel can be the same with respect to size, but different with respect to the actual inserted sequence (see “Example 7, multiple inserted alleles” on page 266). Also, indels can also have different distinct sizes that are both not the reference (see “Indel size determination” on page 262).
In the end, three factors are used to call zygosity:

- Presence of the reference allele
- Presence of different instructed sequences
- Presence of multiple indel sizes

Non-diploid organisms or regions of higher than two in copy number can even have two or all three of the above factors. To take into consideration all the possible combinations of the above factors, the final zygosity call has values of HEMIZYGOUS-REF, HOMOZYGOUS-NON-REF, HEMIZYGOUS-REF-MULTI-INDEL-ALLELE, HETEROZYGOUS-NON-REF, MULTI-HOMOZYGOUS-REF, and HEMIZYGOUS-NON-REF.

**Note:** Having multiple different sizes, without having multiple different sequences, is not possible.

**Sampling of gap alignments**

Sampling of gap alignments occurs when the number of gap alignments contained in a single pileup is greater than small.indel.num-aligns.per.pileup. A larger list than this is initially kept, but that size is limited to 30 times this value. Future gap alignments coming in order from the BAM file are ignored. From this larger list, each item has a probability of being included in the sample list of small.indel.num-aligns.per.pileup divided by the size of the larger list.

This manifests itself in the context-variant-reads tag of the gff. For example,
```
context-variant-seq=GG,AT; context-variant-reads=15464,1497(899,87)
```

means that in the sample, 899 reads had GG in the actual sample, but based on the ratio of the sample size to the total size, an estimated 15,464 reads had that allele in the full sample.

**Examples**

Below are several examples that illustrate the allele calling. The positions reported here are from chromosome 1 of the human hg18 reference.

**Example 1, a simple insertion:**

```
ins_len=1; allele-call-pos=55076169; reference=-; allele-call=A/A; reference-reads=5; context-variant-reads=5; coverage_ratio=1.000; zygosity=HOMOZYGOUS-NON-REF; zygosity-score=0.1000
```

An insertion of A is reported here at 55,076,169 because it occurs between position 55,076,169 and 55,076,170. The allele call here is both alleles are variant. Even though there are as many reference reads as variants, it is called homozygous because the generally greater sensitivity of finding ungapped alignments.

```
> 4223, chr1:55076169-55076169(), INSERTION, 1, allele-call-pos=55076169; allele-call=A/A; allele-counts=REF, 5
AATCTATACATCTATCTCTTTCTTGGCATTGATCTGATCTAGCTAGTAAAGCTTCTG 55076131 55076204
032233311223213002132200002210313012210-3333221131210233102331100302032331
Reads GTATA
103
Ref GT-TA 55076168 55076170
T0333112232130021322000022103130122133333221131210
322000221031301221333332211312102331023311003022T
3220002210313012213333322113332211332102331023311003022T
000221031301221333332211312102331023311003020323OT
T0022103130122133333221131210233102331100302032331
```
Example 2, a simple deletion:

del_len=3; allele-call-pos=91763033; reference=AAA; allele-call=AAA/-; rightmost-allele-call-pos=91763033; rightmost-allele-call=AAA/-; zygosity=HEMIZYGOUS-REF;

This example shows a deletion of AAA where position 91,763,033 would be the start of the deletion and 91,763,035 the end. It is a hemizygous reference call, so the reference is present in the allele call, and there is no difference in this case between the rightmost and leftmost representations.

Example 3, an ambiguous insertion:

ins_len=3; allele-call-pos=2045476; reference=/GTA/GTA; rightmost-allele-call=AGT/AGT; context-pos=2045477; context-reference-seq=gt; context-variant-seq=GTAGT

In this example there are two possibilities, -/GTA where the insertion is after position 2,045,476 or -/AGT, where the insertion is after 2,045,478. Following the left-most rule, the allele call and allele call position are reported as the lowest chromosome position.

The full representation is reported here as g/GTAGT.

Example 4, insertion of a short tandem repeat:

ins_len=3; allele-call-pos=5274096; reference=-/TGT; rightmost-allele-call=-/GTT; context-pos=5274097; context-reference-seq=tgt; context-variant-seq=TGTTGTT

In this example there is a repeat in the sample that is not in the reference, so the full reference is TGT/TGTTGTT, where TGT in the reference starts at position 5,274,097. The small indel caller trims this and then takes the position immediately before the insertion yields a position of 5,274,096 for the leftmost, and 5274100 for the rightmost representations.
Example 5, deletion of short tandem repeats:
\[
\text{del_len=4;allele-call-pos=24115702;reference=agag;allele-call=-/-;rightmost-allele-call-pos=24115702;rightmost-reference=gaga;rightmost-allele-call=-/-;context-pos=24115700;context-reference-seq=acagagagagaac;context-variant-seq=aCAGAGAAC}
\]

The AG repeat occurs 4 times in the reference and only twice in the sample, or more concisely, AGAGAGAG/AGAG. Because it is a deletion, the left-most position of this repeat is reported here at position 24,115,702.

Example 6, an insertion/SNP combination variant:
\[
\text{allele-call-pos=5658680;reference=a;allele-call=a/CT}
\]

At position 5,658,680 the reference has an A. The sample however has a CT, so this complex variant is simultaneously a SNP and an insertion.

Example 7, multiple inserted alleles
\[
\text{ins_len=3;context-pos=60612131;context-reference-seq=A;context-variant-seq=TAGA/TTG/NO_CALL;}
\]

This example has two main inserted alleles, TAG and TTG.
FAQ – Small indels

1

Can the caller find SNP-Indel events?

Yes. These events, which are also referred to as replacements, are annotated by the allele-call tag in the GFF. For instance, allele-call=AT/G is an A/G SNP followed by a T deletion, which can equally be represented as an A deletion followed by a T/G SNP. Similarly, C/TGA is a C/T SNP followed by a GA insertion, which can also be represented as a TG insertion followed by a C/A SNP. There is no hard limit placed on the complexity of this variant (for example, double, triple, quad SNP, followed by an indel are all allowed), as long as the differences in the number of bases is within the gap size range contained in the BAM file, and the variable region is reported accurately by the aligner in the XA or XW BAM tag.

2

How can I speed up the small indel caller?

You can run the caller on different chromosomes at the same time making multiple INI files, one for every chromosome, and using the small.indel.genomic.region parameter to specify each specific chromosome.

3

Can I get the number of reference and variant reads used for each indel call?

Yes. These are directly reported as reference-reads and context-variant-reads tags in the output GFF3 file. Actual bead ids and read sequences are also reported for the variant reads in the GFF3 file using the bead_ids and read_seqs tags. Bead ids for the reference, ungapped reads are reported in the *.ungapped file.
Note that the LifeScope™ Software aligner is generally less sensitive to gap alignments than to ungapped alignments, so two or three times more ungapped alignments are to be expected for hemizygous indels. Also, this calculated number of reference alignments, along with all the other allele counts may include redundant (PCR optical duplicates) alignments. The number of non-redundant reads per allele call is currently not reported.

4

Can I perform gap alignments on all reads for greater sensitivity? Presently only those reads that do not match fully are considered for gap alignments.

The following parameters allow all alignments to be considered for gap alignments:

- For single tag: mapping.gap.min.non.matched.length=0
- For paired tags: pairing.gap.min.non-matched.length=0

However, processing with these settings could be significantly slower and may produce more false positives.

5

Can I use a legacy PAS file as input to the small indel caller?

All inputs files used with the small indel caller should be BAM files. Legacy files such as PAS and MATES files are not supported as they may give unexpected results, such as missing indel IDs or less accurate zygosity calling.

6

Why is a duplicate insertion or deletion reported?

Occasionally an indel is reported twice in the output file. This situation occurs when an indel is detected separately in two different locations, from different sets of reads. After further processing, the two locations are resolved to the same location. These two entries represent a single insertion or deletion.

7

What does "len=no_call or unknown" mean in small indel GFF output?

The message len=no_call or unknown occurs when a deletion is detected and also has reasonably good concordance with dbSNP, but the indel module is not able to determine the deletion size. This message mostly likely occurs with large (> 100bp) deletions.
Run a Large Indels Analysis

This chapter covers:

- Overview ................................................................. 269
- Examples of running a large indel analysis ....................... 270
- Large indel module parameters descriptions ..................... 270
- Large indel internal parameters .................................... 272
- Large indel output files .............................................. 272
- Large indel analysis algorithm .................................... 274
- Identify candidate indels ............................................ 275
- Assign statistical significance to candidate indels ............. 278
- Determine zygosity .................................................... 279
- FAQ – Large indels ................................................... 284

Overview

The large indel module identifies deviations in clone insert size. These deviations indicate intrachromosomal structural variations compared to a reference genome. Insertions and deletions (indels) up to 100 kbp are inferred by identifying positions in the genome in which the pairing distance between mapped mate-pairs deviates significantly from what is expected at the given level of clone coverage.

The module creates a look-up table of standard deviations at each level of clone coverage. The table produces an asymptotic curve in which the minimum size of detectable indels at a given level of significance drops rapidly as the clone coverage increases. The look-up table is used to determine the significance of the deviation in average insert size at each position in the genome.

Regions of the genome that are significantly deviated are selected as candidate indels. Hierarchical clustering is used to segregate the clones into groups in which the difference in the sizes of all clones in a group is less than a specified range. Clusters with too few clones, as specified by the user, are removed and the candidates are assessed to determine if a homozygous or heterozygous population of deviated insert sizes remains. All clones deviated by more than 100 kbp are discarded. Clones from various libraries with various insert sizes contribute to a single indel call by combining the probabilities associated with the clones from each library.
The large indel module only accepts paired data as input.

Examples of running a large indel analysis

In a standard workflow

The following standard workflows provide examples of running the large indels module:

- genomic.resequencing.lmp
- genomic.resequencing.pe

The following are example shell commands using reads and references from the LifeScope™ Software repository.

```bash
lscope.sh shell -u username -w password
cd /projects
mk ecoli
cd ecoli
mk run1
cd run1
set workflow genomic.resequencing.pe
add xsq run0209a_50_PE.xsq
add xsq run0209b_50_PE.xsq
set reference hg19
# optionally change parameter defaults here
ls
run
```

See for Chapter 7, “Run a Standard Workflow Analysis” on page 103 for more information on running standard workflows. Chapter 6, “Run a Command Shell Analysis” on page 75 for information on shell commands and syntax.

As a demo analysis

The optional examples download includes an example of how to run the large indel module by itself, as a single analysis that is not part of a standard workflow. The demo example is not recommended for general use.

See Appendix H, “Demo Analyses” on page 567 for information on running the large indel module as a standalone analysis.

Large indel module parameters descriptions

In order to change a parameter value in your analysis, use the `set param` shell command to replace the line `# optionally change parameter defaults here` in the example commands in “In a standard workflow”. For instance, to change the ploidy parameter, use this shell command:

```
set large.indel.ploidy 0 tertiary/large.indel.ini
```
### Table 68 Large indel parameter descriptions

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>library.type</td>
<td>matepair</td>
<td>Specifies the library type. Allowed values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• matepair</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• pairedend</td>
</tr>
<tr>
<td>large.indel.bas.file</td>
<td>—</td>
<td>(Optional) If you have a BAS file and specify it with this parameter, the file is not regenerated. The BAS file contains metadata information about the data source(s) and relevant mapping/pairing statistics. The BAS format is a pseudo-standard file format for storing BAM file metadata. For details see this site: <a href="">ftp://ftp-trace.ncbi.nih.gov/1000genomes/ftp/pilot_data/README.bas</a></td>
</tr>
<tr>
<td>large.indel.max.clone.cov</td>
<td>1000</td>
<td>The maximum physical [clone] coverage for analysis. Loci with clone coverage above this threshold are not analyzed. Allowed values: Integers &gt;= 3.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> You can use this parameter in combination with the parameter large.indel.high.coverage to reduce false positives in high density genomes, for example, bacteria.</td>
</tr>
<tr>
<td>large.indel.min.coverage</td>
<td>3</td>
<td>The minimum physical [clone] coverage allowed. Loci with clone coverage below this threshold are not analyzed. Allowed values: Integers &gt;= 1.</td>
</tr>
<tr>
<td>large.indel.ploidy</td>
<td>2</td>
<td>Determines zygosity and allele frequencies in eukaryotic genomes. Currently two scenarios are supported haploid (n=1) and diploid (n=2). Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Haploid.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 2: Diploid.</td>
</tr>
<tr>
<td>large.indel.max.insert.size</td>
<td>100000</td>
<td>Maximum insert size, in base pairs.</td>
</tr>
<tr>
<td>large.indel.call.stringency</td>
<td>high</td>
<td>Specifies the large indel call stringency. Automates parameter adjustment to the desired stringency level. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• highest: Recommend when a very low false positive tolerance is allowed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• high: Increased filtering.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• medium: Default values.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• low: Very aggressive.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lower settings results in more indel calls, but with more false positives. Higher settings result in fewer indel calls, but with fewer false positives.</td>
</tr>
<tr>
<td>large.indel.min.pairing.quality</td>
<td>25 (for MP) 10 (for PE)</td>
<td>Paired reads below this threshold are ignored. Allowed values: Integers 0–100.</td>
</tr>
<tr>
<td>large.indel.p.value</td>
<td>1e-10</td>
<td>P-value threshold, that is, the raw probability of committing a Type 1 error incorrectly identifying a large indel. Allowed values: Floats 0.0–1.0.</td>
</tr>
</tbody>
</table>
Chapter 14  Run a Large Indels Analysis

Large indel internal parameters

Table 68 Large indel parameter descriptions (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>large.indel.high.coverage</td>
<td>0 (off)</td>
<td>Eliminate clone coverage weighting. This option significantly reduces the number of false positives when analyzing very high coverage genomic data. (Very high coverage genomic data is typically greater than 1000x read coverage, and is common for bacterial genomes.) Allowed values: 0: No effect on the module. 1: Eliminate clone coverage weighting. The algorithm ignores coverage when calculating the structural significance of candidate indels.</td>
</tr>
</tbody>
</table>

Resource parameters

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>memory.request</td>
<td>7gb</td>
<td>Memory request. Include the units gb in the setting.</td>
</tr>
<tr>
<td>java.heap.space</td>
<td>3000</td>
<td>Dynamic memory requirement, in megabytes. Do not enter a unit with the numeric value.</td>
</tr>
</tbody>
</table>

Large indel module internal parameters

Table 69 describes parameters which we do not recommend changing.

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>large.indel.run</td>
<td>1</td>
<td>Whether or not to run the large indel module.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: Do not run the large indel module.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Run the large indel module during this analysis. The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use.</td>
</tr>
<tr>
<td>large.indel.warnings</td>
<td>0</td>
<td>Whether or not to enable warning messages. Allowed values: 0: Do not enable warning messages. 1: Enable warning messages.</td>
</tr>
</tbody>
</table>

Large indel output files

This section provides descriptions of the large.indel.gff file created by a large indels analysis run (see Table 70).

Table 70 large-indels.gff3 file format description

<table>
<thead>
<tr>
<th>Column Title1</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence ID</td>
<td>Chromosome name</td>
<td>chr11</td>
</tr>
<tr>
<td>Source</td>
<td>Tool name</td>
<td>AB SOLID Large Indel Tool</td>
</tr>
<tr>
<td>Type</td>
<td>Indel type</td>
<td>insertion, deletion, and so forth</td>
</tr>
<tr>
<td>Start Position</td>
<td>Estimated 5' breakpoint</td>
<td>1081331</td>
</tr>
<tr>
<td>End Position</td>
<td>Estimated 3' breakpoint</td>
<td>1084700</td>
</tr>
</tbody>
</table>
Chapter 14  Run a Large Indels Analysis

Large indel output files

The following describes how the values for these fields are generated:

- \( \text{avgDev} \)
- \( \text{devDev} \)
- \( \text{devVar} \)

Let \( X \) be the random variable corresponding to the entire population of reads under consideration and \( E(X) \) be the average of this population.

Consider a point on the chromosome given by \( p \). Let \( Y_p \) be the subset of reads \( R \) from the population such that \( p \) lies in the clonal span of \( R \). By clonal span of \( R \) we mean the entire region of the chromosome that gave rise to \( R \) (F3 read + R3 reads + the insert). Now do unsupervised clustering on \( Y_p \). Two scenarios are possible:

- **Homozygote case** — All \( Y_p \) forms one cluster.
- **Heterozygote case** — \( Y_p \) forms two clusters, \( Y1_p \) and \( Y2_p \).

The heterozygote case has two scenarios:
- One of the two clusters, say \( Y1_p \), has mean and variance similar to population.
- Both clusters have mean and variance deviating from the population. This scenario occurs with a site that has both an insertion and deletion (double zygosity).

---

Table 70  large-indels.gff3 file format description (continued)

<table>
<thead>
<tr>
<th>Column Title(^1)</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Score</td>
<td>Significance of the candidate Indel (p-value)</td>
<td>1e–10</td>
</tr>
<tr>
<td>Strand</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Phase</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Attributes(^2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dev</td>
<td>Indel size, measured in base pair</td>
<td>46</td>
</tr>
<tr>
<td>avgDev(^3)</td>
<td>Average deviation from the population average</td>
<td>-1.7198</td>
</tr>
<tr>
<td>Zygosity(^4)</td>
<td>Results from pair partitioning</td>
<td>Homozygous</td>
</tr>
<tr>
<td>nRef(^5)</td>
<td>Number of reference alleles</td>
<td>0</td>
</tr>
<tr>
<td>nDev</td>
<td>Number of deviated alleles</td>
<td>5</td>
</tr>
<tr>
<td>refDev(^5)</td>
<td>Average deviation of the reference-allele pairs</td>
<td>0</td>
</tr>
<tr>
<td>devDev(^3)</td>
<td>Average deviation of the deviated-allele pairs</td>
<td>-3.567</td>
</tr>
<tr>
<td>refVar(^5)</td>
<td>Variance of the reference-allele pairs</td>
<td>0</td>
</tr>
<tr>
<td>devVar</td>
<td>Variance of the deviated-allele pairs</td>
<td>0.8972</td>
</tr>
<tr>
<td>beadIds(^6)</td>
<td>Bead IDs providing support for the candidate indel</td>
<td>1806_975_1088,...</td>
</tr>
</tbody>
</table>

\(^1\)genome.ucsc.edu/goldenPath/help/customTrack.html#GFF

\(^2\)Semicolon-separated list of large indel module-specific field names followed by an equal ‘=' sign, for example dev=46

\(^3\)Insertions have negative values

\(^4\)Either homozygous, heterozygous, or double

\(^5\)Value is always zero for homozygous indels

\(^6\)Comma-separated list
The following are the calculations for the homozygote case, where \( \text{Var}() \) is the variance of the underlying random variable:

- \( \text{aveDev} = E(X) \)
- \( \text{devDev} = E(\{I_R \mid R \text{ is in } Y_p\}) \)
- \( \text{devVar} = \text{Var}(\{I_R \mid R \text{ is in } Y_p\}) \)

In the heterozygote scenario in which one of the two clusters has mean and variance similar to population, the calculations are as follows:

- \( \text{aveDev} = E(X) \)
- \( \text{devDev} = E(\{I_R \mid R \text{ is in } Y_2_p\}) \)
- \( \text{devVar} = \text{Var}(\{I_R \mid R \text{ is in } Y_2_p\}) \)

In the heterozygote double zygosity scenario, the calculations are as follows:

- \( \text{aveDev} = E(X) \)
- **Two values are reported for devDev:**
  - \( \text{devDev} = E(\{I_R \mid R \text{ is in } Y_1_p\}) / \text{devDev} = E(\{I_R \mid R \text{ is in } Y_2_p\}) \)
- **Two values are reported for devVar:**
  - \( \text{devVar} = \text{Var}(\{I_R \mid R \text{ is in } Y_1_p\}) / \text{devVar} = \text{Var}(\{I_R \mid R \text{ is in } Y_2_p\}) \)

Figure 22 shows an example of an output GFF file displayed in a spreadsheet.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>chr20</td>
<td>A0_SOLID Large Indel Tool</td>
<td>deletion</td>
<td>13022130</td>
<td>13022445</td>
<td>2.79E-11</td>
<td>.</td>
<td>.</td>
<td>dev=381, avgDev=0.000991</td>
</tr>
<tr>
<td>2</td>
<td>chr20</td>
<td>A0_SOLID Large Indel Tool</td>
<td>deletion</td>
<td>13024362</td>
<td>13034578</td>
<td>1.86E-12</td>
<td>.</td>
<td>.</td>
<td>dev=271, avgDev=1.056223</td>
</tr>
<tr>
<td>3</td>
<td>chr20</td>
<td>A0_SOLID Large Indel Tool</td>
<td>deletion</td>
<td>14030747</td>
<td>14030390</td>
<td>2.79E-12</td>
<td>.</td>
<td>.</td>
<td>dev=183, avgDev=0.735298</td>
</tr>
</tbody>
</table>

Figure 22 large-indels.gff3 file example

**Large indel analysis algorithm**

Large indel detection is a tertiary module in LifeScope™ Software. Data from the pairing phase of the mapping module serve as direct inputs for large indel discovery (see Figure 23).

The large indel module uses pairing statistics are available in BAM headers of the input BAM files. If the pairing statistics are not available, the large indel module estimates the required parameters.
The large indel module processes inputs using an alignment window to hold and analyze sets of locus-spanning pairs. Regions with pairs that have significant insert size deviations are chosen as candidate indel sites, which are processed further to determine zyosity. The output of the analysis is one file in the GFF format. The GFF file contains information about the location, size, and significance of each large indel detected by the module.

Figure 23 Large indel analysis module

This paragraph refers to Figure 23 on page 275. The large indel module accepts as input BAM files (blue) are generated during the secondary analysis module (mapping and pairing). Pairing statistics are included in the BAM headers. Pairing statistics provide more accuracy and can improve the final results of the large indel module. Final output consists of a GFF file (orange).

Identify candidate indels

Pairing distances (sometimes called insert sizes) for each pair are assigned during the mapping and pairing module and subsequently used by the large indel module to determine indel candidacy.
Chapter 14  Run a Large Indels Analysis

Identify candidate indels

Note: Insert sizes can be non-unique in the case of multiple feasible mapping/pairing combinations. When insert sizes are non-unique, the primary (optimal) pair is chosen for large indel analysis.

Clones that have been mapped and paired to a reference genome are classified as either concordant or discordant (see Figure 24 on page 276). Concordant pairs are those with insert sizes (sometimes called inter-read distances) that are not significantly deviated from the expected insert size of the library as a whole. Discordant pairs have insert sizes that deviate significantly from the expected value. Discordant pairs containing a putative deletion appear larger when mapped to the reference. Pairs containing a putative insertion appear smaller. Multiple discordant pairs in close proximity provide evidence for a candidate indel within the covered region.

Figure 24  Discordant clones used to identify candidate insertions or deletions

In Figure 24:

Pairs spanning a candidate indel (red) appear distorted when mapped to a reference genome. Insert sizes appear larger for deletions (left) and smaller for insertions (right).

The module moves across individual chromosomes in order of genomic position to generate an alignment window of overlapping locus-spanning pairs (see Figure 25 on page 277). When the window encounters the first read in a pair, the corresponding alignment is incorporated into the alignment window. Simultaneously, several moving statistics, including the number of locus-spanning pairs as well as their average insert size and variance, are updated. When the window encounters the second read in a pair, the corresponding alignment is dropped and the moving statistics are updated appropriately. A genomic region (sometimes called a window) is considered to contain a candidate indel when the average insert size of the set of locus-spanning clones is significantly deviated from the average insert size of the library as a whole.
Figure 25  Sets of locus-spanning pairs

The following sections refer to Figure 25.

Top panel

An alignment window, \( W \), contains the set of overlapping locus-spanning pairs (black loops) at a particular genomic position \( i \) (arrow) of the reference genome (black line).

Middle panel

The window advances to the next position by incorporating the next alignment \( i+1 \). The corresponding alignment is the second in the pair, so the alignment \( i \) is dropped from the alignment window and the alignment statistics are updated accordingly. When the window encounters the first alignment, the corresponding pair is added to the window.
Assign statistical significance to candidate indels

Regional insert size deviations can be calculated directly from the moving statistics associated with each clone window. The deviations that achieve statistical significance indicate relatively large structural variations compared to the reference genome (see Figure 26). Hypothesis testing determines the significance of deviations, where the null hypothesis asserts an insignificant difference between the local average insert size and the population average ($H_0: \mu = \mu$). Candidate deviations are chosen where the probability of falsely rejecting the null hypothesis in favor of the alternative ($H_1: \mu \neq \mu$) falls below a user-defined confidence threshold.

\[ z_i = \frac{\bar{x}_i - \mu}{\sigma} \]

Figure 26 Hypothesis testing example

The following paragraph refers to Figure 26.
The population average insert size $\mu$ and standard deviation $\sigma$ are calculated from the input BAM file. The alignment window contains a very small subset of pairs sorted by insert size (grey bars). Moving statistics including the number of locus-spanning pairs $n_i$ as well as the sample average insert $\bar{x}_i$ size are calculated from this subset of pairs at each genomic position $i$. The parameters are used to z-normalize insert sizes according

to $z_i = \frac{\bar{x}_i - \mu}{\sigma}$, which measures the absolute insert size deviation between the sample and the population in units of standard deviation. The normalization step allows multiple libraries with variable insert sizes to be combined into one analysis. A candidate indel is considered significant if $p(z_i | n_i) < \alpha$ where probability values (p-values) are calculated according to the standard normal distribution and is a user-defined threshold parameter, where `large.indel.p.value` default is $p=1e^{-10}$.

**Determine zygosity**

After a candidate indel is detected and deemed significant, the alignment window is partitioned to remove erroneous pairs because of mapping/pairing artifacts and to further characterize indel alleles and zygosity (see Figure 27 on page 280). Locus-spanning pairs are partitioned to remove two groups because polyploidy is currently not supported.

Each partition represents a disjoint subset of pairs from the alignment window optimally grouped by insert size so that pairs with similar insert sizes are placed in the same partition. Partitions with only one pair (sometimes called outliers) are removed from consideration. The candidate indel is also removed if the average insert size for pairs in the remaining partition are not significantly deviated from the population average. Removing the candidate indel can occur when mapping or pairing errors are responsible for observed insert-size deviations.

The number of pairs per partition and various summary statistics are calculated for each partition to determine alleles, allele frequencies, and zygosity. Candidate regions with minor allele frequencies less than one-third are removed from consideration.

The large indel module uses a heuristic method to categorize each candidate indel. If both partitions contain pairs with insert sizes that are significantly deviated from the reference but not from each other, the region is designated HOMOZYGOUS. If pairs are significantly deviated from the reference and from each other, the region is designated DOUBLE, which indicates two indel alleles of different types or sizes. DOUBLE regions can be placed into one of several indel/indel categories, for example, insertion/deletion. If one pair is deviated from the reference and the other pair is not, the region is designated HETEROZYGOUS, which indicates the presence of indel and reference alleles.
Determine zygosity

Figure 27 Partitioning the alignment window to determine zygosity

**Left panel**
Heuristic methods are used to characterize indel alleles including homozygous reference alleles (lines), which are removed from consideration; insertions (hatched boxes), and deletions (broken lines).

**Middle panel**
Each partition contains pairs with characteristic insert size distributions. Summary statistics describing the distributions are used to determine indel and reference alleles (black and red bell curves, respectively).

**Right panel**
Based on the results of the analysis, each candidate indel is assigned an appropriate zygosity category.

**Filter alignments and parameter optimization**
A user-defined pairing quality value (PQV) filtering threshold is set with the parameter `large.indel.min.pairing.quality`. The PQV threshold setting places constraints on which alignments are incorporated into the alignment window and subsequently used to determine large indel candidate regions. The default PQV threshold values are 25 for mate-pair data, and 10 for paired-end data. Adjusting the PQV threshold down (less than the default) to include lower-quality alignments generally improves sensitivity but increases the number of false positives. The opposite is true if you adjust the PQV threshold up (greater than the default).
Adjusting the p-value and PQV thresholds together might be required to optimize the module for analysis of data that differ significantly from normal HuRef samples, for example, non-human or cancer genomes. However, the default settings provide a convenient starting point for this process.

Additional optimization might be required for very high coverage samples. For example, alignments from high-density slides mapped to small prokaryotic genomes typically result in clone coverage values above 1000x. In the case of clone coverage values that are above 1000x, use the high-coverage flag (large.indel.high.coverage in LifeScope™ Software), which adjusts the p-value calculation to \( p(z) < \alpha \), eliminating the conditional coverage parameterization (weighting), and reducing the number of false positives that might otherwise result.

**Input files for large indel analysis**

The only acceptable inputs for large indel analysis are BAM files containing paired-end or mate-pair data. Other alignment types are not compatible with the large indel module.

**Note:** LifeScope™ Software does not support combining paired-end and mate-pair data into a single analysis.

**Interpreting results from the large indel module**

Large indel results are written to a GFF file, which can be uploaded directly into the UCSC Genome Browser (cbse.ucsc.edu/research/browser) or other similar visualization tools. See Table 70 on page 272 for a description of the GFF file format. See Figure 28 on page 282 and Figure 29 on page 283 for examples of GFF files generated by the large indel module.
**Figure 28** Identifying a 46bp insertion in MUC2

In Figure 28, paired-end data was analyzed using the SOLiD® mapping and pairing modules (against a human reference). The resulting BAM file was used for large indel analysis with default settings. The top panel uses the Integrated Genomics Viewer (IGV) to represent all BAM alignments covering the region chr11:1081331–1084700, which contains the partial coding region for the human mucin 2 precursor (MUC2) (bottom panel) as well as an indel previously identified by Levy et al., 2007 (not shown). Further information about IGV, including alignment color-encodings, is available at broadinstitute.org. The middle panel represents the subset of alignments used by the large indel module to identify a 46bp homozygous insertion breakpoint at the indicated position (dotted grey line). Forward and reverse strand reads are highlighted red and blue, respectively. The blue rectangle displays various alignment features, including a deviated insert size (97bp) that is significantly smaller than the average insert size of the population (170bp) providing evidence for an insertion at this site. The blue rectangle was generated by mousing over one of the reverse strand reads.
In Figure 29, the top IGV panel represents long mate-pair alignments (mapped to HuRef) covering the region chr10:6135605-6139655, which contains an intron of human interleukin 2 receptor alpha (bottom panel) as well as several indels previously identified by Ahn et al., 2009; Bentley et al., 2008; Wang et al., 2008; Wheeler et al., 2008; and Levy et al., 2007 (not shown). Several alignment features are consistent with the presence of a deletion at this site, including six gapped alignments (dotted lines) flanking the putative deletion. The region between the six gapped alignments contains very few reads with high mapping quality, indicating sequence that is present in the reference but lacking in the sample. Surrounding the gapped alignments is the set of deviated pairs identified by the large indel module as supportive evidence for a 413bp homozygous deletion (middle panel). Reads with significantly deviated insert sizes, > 2000bp compared to the population average (1575bp), are highlighted (in pink).
FAQ – Large indels

1 Why are there so many more deletions than insertions?
Insertion detection is limited by the average insert size of the library, typically around 1500 base pair for mate-pairs. The average insert size of paired-end libraries is much smaller (around 150 base pairs), and only insertions smaller than the insert size can be detected.

2 What is the resolution of large indel detection?
This depends on clone coverage, insert size variability, statistical threshold, and library type. For example, mate-pair data detects large insertions and deletions ranging from 30 base pair to 1.2 kbp and 86 bp to 100 kbp, respectively, in human hg18 (McKernan et al, 2009). For paired-end data, deletion sizes range from 100 base pairs to 2 kbp, and insertions are essentially undetectable.

3 Why are there so many large indels around 300 base pair and deletions around 6 kbp?
Alu elements (SINEs) and LINEs, respectively (McKernan et al, 2009).

4 How many large indels can I expect?
The quantity of large indels to expect depends on:
- Genome size
- Clone (read) coverage
- Average insert size
- Significance threshold
- Pairing quality value threshold
- Library type

Note: The Human hg18 has 4075 deletions and 1515 insertions.
5

**Why do pooled samples take so long to analyze?**

In areas of higher coverage, more significant candidate regions are found, causing more clustering. Analysis at CNVs can slow down as the coverage exceeds 2x.

---

6

**How long does the module take to run and how much space is required?**

Significant algorithmic improvements have cut processing time. Additional parameters and better implementation have reduced lag times associated with too much coverage and zygoisty calculation (clustering).

- Non-parallelized human genome runs consisting of approximately 500 million reads typically take two to four hours to run, depending on platform and resource load.
- Running jobs in parallel at one job per chromosome, significantly reduces run time. Intermediate file generation is negligible for BAM file inputs.
FAQ – Large indels
Add Genomic Annotations to Analysis Results

This chapter covers:

- Overview .......................................................... 287
- Workflows .......................................................... 291
- Annotation sources ................................................ 292
- Use custom reference and custom dbSNP files ................ 293
- Annotation input files .............................................. 295
- Prepare to run annotation processing .......................... 296
- Annotation parameters ............................................ 297
- Output annotation .................................................. 300
- Annotation output files .......................................... 303
- About annotations and LifeScope™ Software modules .... 318
- FAQ - Annotations ............................................... 319

Overview

The annotations module is an optional post-processing step available with select LifeScope™ Genomic Analysis Software analysis modules. The annotations processing makes a copy of the analysis module's GFF file and adds new attributes to the GFF entries in the copied file. The new attributes are taken from information in publicly available sources about the variants in the input, about features intersecting the variants in the input file, or about any biological function potentially changed by the variants.

Annotations processing does not modify the original LifeScope™ Software analysis module's output file.

Several output files are potentially generated by the annotations processing, depending on the analysis modules involved. One annotations output file contains all of the content of the GFF file generated by the modules, with added attributes for annotations. Another annotations output file is a subset of the previous file, and it contains only those entries that pass filtering requirements that you specified.

Annotation processing also produces output files containing only mutated genes, or only SNPs. Several statistics files are also produced.

Two sources of annotation are used:
Overview

- A UCSC or Ensembl GTF file, used to determine whether a variant overlaps a gene or exon.
- The National Center for Biotechnology Information (NCBI) dbSNP database, which contains information on SNPs and indels already found by other studies. We use the following fields of the dbSNP entries:
  - The reference SNP identifier (rsID)
  - The functional code
  - The corresponding locus ID (the gene)

If a conflict is found between the dbSNP data and the GTF data, both annotations are reported. The annotation module does not attempt to resolve these conflicts.

Annotation sources can be either the refGene GTF file with respect to the human hg18 reference build and the dbSNP 130 build or the human hg19 reference build and the dbSNP 132 build (both available with LifeScope™ Software), or a source that you provide. Only GTF files and dbSNP data can be used as sources. The data from these three tables in dbSNP is required:

- SNPChrPosOnRef
- SNPContigLoc
- SNPContigLocusId

Annotations processing supports these annotation types:

- DNA features: genes and protein-coding features
- Verified variants in existing database

Post-processing types include:

- Annotate genomic variants with the annotation types that apply
- Filter based on annotation
- Report a list of gene and protein-coding annotations involved the variants
- Report statistics

The annotations module is available with the following modules:

- Resequencing:
  - SNPs
  - Small indels
  - Large indels
  - Human CNV
- Targeted resequencing:
  - SNPs
  - Small indels
- Low frequency variant:
  - SNPs

Input file handling
Annotations processing uses the GFF output of an analysis module, but does not modify the GFF file. Annotations are added to a copy of the GFF file.

Filters
Annotation processing optionally generates an output file containing only those entries that fulfill all the conditions you specify with annotation filtering options. Annotation filters are described in Table 71.
By default, the filtering options report all variants (regardless of whether or not they appear in dbSNP and whether or not they occur in exons or genes).

**Table 71** Annotation filtering options

<table>
<thead>
<tr>
<th>Filtering option</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Report only variants in exons</td>
<td>Off</td>
</tr>
<tr>
<td>Report only variants in genes</td>
<td>Off</td>
</tr>
<tr>
<td>Report only variants that appear in dbSNP</td>
<td>Off</td>
</tr>
<tr>
<td>Report only variants that do not appear in dbSNP</td>
<td>Off</td>
</tr>
</tbody>
</table>

**Table 72** shows the annotation filtering supported in select LifeScope™ Software modules.

**Table 72** Support for annotation filtering in each variant type

<table>
<thead>
<tr>
<th>Annotation type</th>
<th>SNP</th>
<th>CNV</th>
<th>Small indel</th>
<th>Large indel</th>
</tr>
</thead>
<tbody>
<tr>
<td>geneID</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>exonID</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>rsID</td>
<td>Yes</td>
<td>—</td>
<td>Yes</td>
<td>—</td>
</tr>
<tr>
<td>functionCode</td>
<td>Yes</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

**Statistics**

**Table 73** lists the statistics included in the annotations output files and the LifeScope™ Software modules that support these annotations. See **Table 73** for information about SNPs transitions and transversions.

**Table 73** Statistics generated in annotations output files

<table>
<thead>
<tr>
<th>Analysis type</th>
<th>Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>All supported modules (SNPs, CNVs, Indels)</td>
<td>Number of variants&lt;br&gt;Number of variants per chromosome&lt;br&gt;Number of heterozygous variants&lt;br&gt;Number of homozygous variants&lt;br&gt;Number of heterozygous SNPs per chromosome&lt;br&gt;Number of homozygous SNPs per chromosome</td>
</tr>
<tr>
<td>SNPs</td>
<td>Number of heterozygous SNPs that are transitions, transversions&lt;br&gt;Number of homozygous SNPs that are transitions, transversions (compared to the reference)</td>
</tr>
<tr>
<td>Indels</td>
<td>Indel variant length distribution (negative for deletion, positive for insertion)</td>
</tr>
<tr>
<td>CNVs</td>
<td>Copy number distribution&lt;br&gt;CNV length distribution</td>
</tr>
</tbody>
</table>
Overview

Table 74 describes how transitions and transversions are determined based on the reference allele and the diBayes output genotype.

Table 74 Transitions and transversions

<table>
<thead>
<tr>
<th>diBayes genotype</th>
<th>Reference allele</th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>A</td>
<td>0</td>
<td>TV</td>
<td>TS</td>
<td>TV</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>TV</td>
<td>0</td>
<td>TV</td>
<td>TS</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>G</td>
<td>TS</td>
<td>TV</td>
<td>0</td>
<td>TV</td>
<td>0</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>TV</td>
<td>TS</td>
<td>TV</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>M</td>
<td>A/C</td>
<td>TV</td>
<td>TV</td>
<td>TV</td>
<td>TV</td>
<td>TV</td>
</tr>
<tr>
<td>R</td>
<td>A/G</td>
<td>TS</td>
<td>TS</td>
<td>TS</td>
<td>TS</td>
<td>TS</td>
</tr>
<tr>
<td>W</td>
<td>A</td>
<td>T</td>
<td>TV</td>
<td>TV</td>
<td>TV</td>
<td>TV</td>
</tr>
<tr>
<td>S</td>
<td>C</td>
<td>G</td>
<td>TV</td>
<td>TV</td>
<td>TV</td>
<td>TV</td>
</tr>
<tr>
<td>Y</td>
<td>C</td>
<td>T</td>
<td>TS</td>
<td>TS</td>
<td>TS</td>
<td>TS</td>
</tr>
<tr>
<td>K</td>
<td>T</td>
<td>G</td>
<td>TV</td>
<td>TV</td>
<td>TV</td>
<td>TV</td>
</tr>
<tr>
<td>Other</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Workflows

Annotation is integrated into the resequencing, targeted resequencing, and low frequency variant standard workflows. Table 75 shows the modules with integrated annotation support, in these workflows.

Table 75  Standard workflows and analysis modules with integrated annotation support

<table>
<thead>
<tr>
<th>Workflow Type</th>
<th>Library Type</th>
<th>SNP</th>
<th>CNV</th>
<th>Small indel</th>
<th>Large indel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genomic resequencing</td>
<td>Fragment</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>—</td>
</tr>
<tr>
<td>Genomic resequencing</td>
<td>Mate-pair</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Genomic resequencing</td>
<td>Paired-end</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Targeted resequencing</td>
<td>Fragment</td>
<td>Yes</td>
<td>—</td>
<td>Yes</td>
<td>—</td>
</tr>
<tr>
<td>Targeted resequencing</td>
<td>Paired-end</td>
<td>Yes</td>
<td>—</td>
<td>Yes</td>
<td>—</td>
</tr>
<tr>
<td>Low frequency variant</td>
<td>Fragment</td>
<td>Yes</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Low frequency variant</td>
<td>Paired-end</td>
<td>Yes</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

Example shell commands

The following are example shell commands to run an analysis with annotations. This example uses reads and references from the LifeScope™ Software repository.

```bash
# log into the shell
lscope.sh shell -u username -w password
# cd to the projects repository
cd /projects
# make a project, and open it
mk ecoli
cd ecoli
# make an analysis, and open it
mk run1
cd run1
# define the analysis type
set workflow genomic.resequencing.frag
# define your input data
add xsq DH10B_ColorSpaceOnly_Unbarcoded_50_Frag.xsq
# specify the reference to be used
set reference hg18
# optionally change parameter defaults after this line or
# set analysis-specific parameters
# list the configuration of the analysis
ls
# start the run
run
# list progress information about the run
ls
```

Because the set reference hg18 command uses a known assembly, reference-related parameters including annotation parameters are set automatically. (Confirm these settings with the LifeScope™ Software shell ls command before starting your analysis.)
To specify your own annotations files, use the `set param` command, as shown in these examples:

```bash
set annotation.dbsnp.file /path/to/dbsnp/file tertiary/INIfile
set annotation.gtf.file /path/to/gtf/file tertiary/INIfile
```

Place your `set param` commands after the `set reference` command (after the line `# optionally change parameter defaults after this line`). The INI file to use with the `set param` command depends on the type of LifeScope™ Software analysis you are running. Replace the string `INIfile` with one of these INI file names:

- `annotation.cnv.ini`
- `annotation.dibayes.ini`
- `annotation.large.indel.ini`
- `annotation.small.indel.ini`

### Annotation sources

**Table 76** lists common annotation sources supported by LifeScope™ Software. Files required to support certain annotations are available with LifeScope™ Software. These files support the following:

- Annotations based on the hg18 and the dbSNP 130 build
- Annotations based on the hg19 and the dbSNP 132 build

**Note:** The dbSNP 132 file used is a VCF file containing 1000 Genome data.

<table>
<thead>
<tr>
<th>Type of annotation</th>
<th>Location and notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variants in existing databases</td>
<td>NCBI Variation dbSNP resources: includes single nucleotide polymorphisms, microsatellites, and small-scale insertions and deletions. dbSNP contains population-specific frequency and genotype data, experimental conditions, molecular context, and mapping information for both neutral polymorphisms and clinical mutations. (<a href="http://www.ncbi.nlm.nih.gov/guide/variation">This text is taken from NCBI website</a>.)</td>
</tr>
<tr>
<td><strong>Note:</strong> Supported by LifeScope™ Software only if the data is downloaded in GTF format.</td>
<td>The European Bioinformatics Institute’s Alternative Splicing Database Project: (<a href="http://www.ebi.ac.uk/asd">http://www.ebi.ac.uk/asd</a>)</td>
</tr>
</tbody>
</table>

**Note:** While every effort is made to provide correct web addresses, Applied Biosystems is not responsible for the changes to these addresses, nor for the data provided by these locations.

**IMPORTANT!** Annotation files from dbSNP must be in VCF format. Other formats (such as BCP files) are *not* supported.
Use custom reference and custom dbSNP files

Note: LifeScope™ Software supports only one format for GTF files. Public sources may have a variety of formats. To use custom GTF files, it may be necessary to download and modify these files into our supported GTF format. This section contains instructions to convert files into the supported GTF format.

This section describes the steps required to use a custom reference with LifeScope™ Software. These steps are an alternative to using the reference-related files available with LifeScope™ Software.

Two main processes are required (listed here and described below):

1. Download and convert or reformat a GTF file. Choose one of the following:
   - Download and convert a UCSC GTF file.
   - Download and reformat an Ensembl GTF file.

2. Download and unzip dbSNP files.

The UCSC GTF file

Follow either the steps in this section or in the section “The Ensembl GTF file” on page 293.

After download, a conversion step is required to normalize the file by gene ID as required by LifeScope™ Software. Genome annotations that are downloaded from the UCSC Genome Browser and converted by the annotation conversion script are optimal because they contain Human Genome Organization (HUGO)-style gene names. HUGO-style gene names allow interpretation when using a genome browser or reading reports.

The annotation conversion script refgene2gff.sh works with the most recent format of refGene.txt files (as of the time of this writing). Some assemblies such as the rat genome use an alternate format for the refGene.txt file. The refgene2gff.sh script does not convert these alternate formats.

Follow these steps to download and convert the UCSC GTF file:

1. Download the file from the following URL:
   ftp://hgdownload.cse.ucsc.edu/goldenPath/hg19/database/refGene.txt.gz

2. Unzip the file with the following command:
   gunzip refGene.txt.gz

3. If the Gene ID is empty (or blank), add text entry such as “0” to the Gene ID. (Use a zero, without the quotes).

4. Use this command to convert the file to GTF format:
   refgene2gff.sh -i refGene.txt -o refGene.gtf

5. Specify the file, refGene.gtf, with the annotation.gtf.file parameter. (If you use the set reference assembly command and specify a known assembly, the annotation.gtf.file parameter is automatically set.)

The Ensembl GTF file

Follow either the steps in this section or in the section “The UCSC GTF file” on page 293.
Ensembl GTF files have the following characteristics that apply to their use with LifeScope™ Software.

- They are properly normalized by gene and transcript IDs.
- They use gene accession numbers instead of HUGO-style gene names.
- They use unprefixed sequence identifiers, such as 1, 2, 3, …X, Y, MT.
- They are incompatible with genome reference FASTA files that have UCSC-style sequence IDs with the prefix “chr”, for example, chr1, chr2, chr3, …chrX, chrY, chrM.
- In order to be used as input for annotations with LifeScope™ Software, they must be reformatted to use UCSC-style gene IDs.

Follow these steps to download and reformat the Ensembl GTF file:

1. Download the file from the following URL:
   
   http://www.ensembl.org/

2. Reformat the GTF file with the `reformat_ensembl_gtf.pl` command, and redirect the output to a new file with `.reformatted` in the file name.
   
   For Example:
   ```bash
   reformat_ensembl_gtf.pl Homo_sapiens.GRCh37.63.gtf > \\
   Homo_sapiens.GRCh37.63.reformatted.gtf
   ```

   The input GTF file name typically includes the NCBI build number and the Ensmbl release number in its name, according to the following pattern:

gtf`

3. Specify the reformatted file with the `annotation.gtf.file` parameter.
   (If you use the `set reference assembly` command and specify a known assembly, the `annotation.gtf.file` parameter is automatically set.)

### The dbSNPs files

Annotations based on dbSNPs require the data from these three dbSNPs tables: SNPChrPosOnRef, SNPContigLoc, SNPContigLocusId. LifeScope™ Software is verified to work with the format of these files as of February 2011. The dbSNP files follow this naming pattern:

`b<dbSnp_build_no>_<table_name>_<reference_ncbi_build_no>.bcp.gz`

For example, for build 131 and reference hg19 (NCBI build 37.1), the files required are shown in Table 77:

**Table 77** Example files for dbSNP

<table>
<thead>
<tr>
<th>dbSNPs table</th>
<th>File name</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNPChrPosOnRef</td>
<td>b131_SNPChrPosOnRef_37_1.bcp.gz</td>
</tr>
<tr>
<td>SNPContigLoc</td>
<td>b131_SNPContigLoc_37_1.bcp.gz</td>
</tr>
<tr>
<td>SNPContigLocusId</td>
<td>b131_SNPContigLocusId_37_1.bcp.gz</td>
</tr>
</tbody>
</table>

Follow these steps to use hg19 dbSNPs files with LifeScope™ Software annotations:

1. Download the dbSNP files from this site:
   
2. Specify the information from these files in your analysis with the annotation.dbsnp.file parameter.
   (If you use the set reference assembly command and specify a known assembly in the reference repository, the annotation.dbsnp.file parameter is automatically set.)

3. Increase the memory and heap space for your analysis, if necessary. See “Set your memory requests” on page 297 for instructions.
   “Annotation parameters” on page 297 provides more instructions on running annotations.

Annotation input files

This section describes the input files used during annotation processing.

GFFv3 variant file
This file is the output file of a previous analysis module such as SNPs, small indels, large indels, or CNV. The variant file must be in GFFv3 format.

GTF
The input genome annotation file is in GTF format. Refer to the following websites for information on the GTF file format:

   genes.cse.wustl.edu/GTF2.html
   genome.ucsc.edu/FAQ/FAQformat.html#format4

The GTF file must match the genome reference to ensure that the analysis modules work correctly.

IMPORTANT! Make sure that the GTF file is for the same genome assembly as the genome FASTA reference file, and that matching sequence identifiers are used. Gene and transcript identifiers in the GTF file must be properly normalized. Identifier normalization is a known issue in GTF files from the UCSC Genome Browser, which is a popular source of GTF-formatted annotation. The UCSC GTF files always report the same value for gene and transcript IDs.

Use the LifeScope™ Software scripts to transform a GTF file into the format required for use with LifeScope™ Software modules. (See “The UCSC GTF file” and “The Ensembl GTF file” on page 293 for more information.)

dbSNP
Data downloaded from the publicly-available dbSNP database is included in the LifeScope™ Software reference repository. This data is based on human reference builds hg18 dbSNP version 130 and hg19 dbSNP version 132.

LifeScope™ Software automatically determines your dbSNP data file from the assembly name you specify, when you use the set workflow assembly command. For example:

   set workflow hg18
   set workflow hg19

You also have the option to specify a different dbSNP database (that you must supply) for your annotations. See “Add new reference files” on page 547 in Appendix E, The Reference Repository.
Chapter 15 Add Genomic Annotations to Analysis Results

Prepare to run annotation processing

The following information is parsed from the dbSNP data:

- The refSNP identifier, rsID
- Chromosome number
- Chromosome start coordinate
- Chromosome end
- Whether or not the variant is a SNP
- Function code (consequence to transcript, if any)
- The gene name, locus_symbol (if available)

hg19
The hg19 dbSNP file is dbSNP_b132_00-All.vcf.

hg18

Table 78 describes the NCBI hg18 files used for annotations. These are a set of three tab-separated files.

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
<th>Fields used</th>
</tr>
</thead>
</table>
| b<dbSNP build number>_SNPChrPosOnRef_<reference build version>.bcp | A 4-column, tab-separated file, corresponding to the table SNPChrPosOnRef | The first 3 columns:  
  - The rsID  
  - Chromosome  
  - Position |
| b<dbSNP build number>_SNPContigLocusId_<reference build version>.bcp | A 21-column, tab-separated file, corresponding to the table SNPContigLocusId |  
  - snp_id: The rsID  
  - fxn_class: The function class code  
  - (Optional) locus_symbol: The gene name |
| b<dbSNP build number>_SNPContigLoc_<reference build version>.bcp | A 22-column, tab-separated file, corresponding to the table SNPContigLoc |  
  - snp_id: The rsID  
  - asn_from, asn_to: Used to determine the feature length  
  - loc_type: Used to determine whether an entry is a SNP or an indel |

Prepare to run annotation processing

Select the required input files

This section only applies to references not in the reference repository.

If you supply your own annotation source files, you must know the absolute path to those files.

Note: This section does not apply if you specify the human hg18 or hg19 assembly, because this option uses the annotation sources available with LifeScope™ Software.

To include gene and exon annotations, you must know the absolute path to:

- Your GTF file for gene and exon annotations
To include dbSNP annotations, you must know the absolute path to your dbSNP annotation file or files. Your dbSNP annotations might be in a single VCF file, or in a set of three BCP files:

- `{dbSnp_build_no}_SNPChrPos_{reference_ncbi_build_no}.bcp`
- `{dbSnp_build_no}_SNPContigLoc_{reference_ncbi_build_no}.bcp`
- `{dbSnp_build_no}_SNPContigLocusId_{reference_ncbi_build_no}.bcp`

You control which types of annotation are generated:

- To turn off gene and exon annotations, do not provide a GTF file.
- To turn off dbSNP annotations, do not provide the dbSNP files.

**Note:** If any one of the three required dbSNs files is missing from a set of three BCP files, no dbSNP's annotation is generated.

### Set your memory requests

The memory requirements for dbSNPs annotations vary by reference build and dbSNPs build. Table 79 lists the hardware memory requirements for dbSNPs annotations for supported hg and dbSNPs builds.

**Table 79** Hardware memory requirements for dbSNPs annotations

<table>
<thead>
<tr>
<th></th>
<th>hg18</th>
<th>hg19</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbSNP 130</td>
<td>22GB</td>
<td>Not supported</td>
</tr>
<tr>
<td>dbSNP 131</td>
<td>Not supported</td>
<td>22GB</td>
</tr>
<tr>
<td>dbSNP 132</td>
<td>Not supported</td>
<td>22GB</td>
</tr>
</tbody>
</table>

These parameters must be added to your module’s properties file. Parameters in a module properties file affect all runs of that module, for all users.

These requirements are for full annotations (dbSNPs as well as exon-based annotations). The annotation module requires considerably less memory when run with exon annotation only (that is, with a GTF file but without a dbSNP file). However, the requirement increases linearly with the number of exons in the GTF file.

### Complete the prerequisites

Additional prerequisites specific to each type of analysis are described in the instructions for each analysis module.

### Annotation parameters

Table 80 describes the annotation parameters used with LifeScope™ Software analyses.

For filtered output files, all annotation filter options that are turned on must be fulfilled by the variant entry, for the variant to appear in the output file.

**Table 80** Annotation parameters description
Table 80  Annotation parameters description (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>annotation.dbsnp.annotate.snp</td>
<td>0</td>
<td>Whether or not to annotate variants with SNP entries from the dbSNP file. Allowed values:</td>
</tr>
<tr>
<td>Note:</td>
<td></td>
<td>• 0: Do not annotate variants with SNP entries from the dbSNP file.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Annotate variants with SNP entries from the dbSNP file.</td>
</tr>
<tr>
<td>annotation.dbsnp.annotate.indel</td>
<td>0</td>
<td>Whether or not to annotate variants with indel entries from the dbSNP file. Allowed values:</td>
</tr>
<tr>
<td>Note:</td>
<td></td>
<td>• 0: Do not annotate variants with indel entries from the dbSNP file.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Annotate variants with indel entries from the dbSNP file.</td>
</tr>
<tr>
<td>annotation.indel.border.slack</td>
<td>5</td>
<td>An indel is considered to be matched to a dbSNP indel if the two overlap or if the distance between them is less than or equal to the value of annotation.indel.border.slack.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allowed values: Integers &gt;= 0.</td>
</tr>
<tr>
<td>annotation.show.only.variants.in.exons</td>
<td>0</td>
<td>When set to 1, filters the annotated output: restricts the Filtered Variant annotation output file (see “Annotation output files” on page 303) to only the variants that overlap any exon.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Do not filter the output for variants that overlap any exon.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Restrict output to variants that overlap any exon.</td>
</tr>
<tr>
<td>annotation.show.only.variants.in.genes</td>
<td>0</td>
<td>When set to 1, filters the annotated output: restricts the Filtered Variant annotation output file (see “Annotation output files” on page 303) to only the variants that overlap any gene, even if it does not overlap any exon.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Do not filter the output for variants that overlap any gene.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Restrict output to variants that overlap any gene.</td>
</tr>
<tr>
<td>Note:</td>
<td></td>
<td>if the parameter annotation.show.only.variants.in.exons is set to 1, then the setting for the parameter annotation.show.only.variants.in.genes is ignored.</td>
</tr>
<tr>
<td>annotation.show.only.variants.in.dnsnp</td>
<td>0</td>
<td>When set to 1, filters the annotated output: restricts the Filtered Variant annotation output file (see “Annotation output files” on page 303) to only SNPs or small indels that exist in dbSNP.</td>
</tr>
<tr>
<td>Note:</td>
<td></td>
<td>applies only to SNPs and small indels modules</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Do not filter the output for variants that appear in dbSNP.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Restrict output to SNPs or small indels variants that appear in dbSNP.</td>
</tr>
</tbody>
</table>
Annotations parameters are by convention defined in INI files with the following names and location:

- `<analysis>/tertiary/annotation.cnv.ini`
- `<analysis>/tertiary/annotation.dibayes.ini`
- `<analysis>/tertiary/annotation.large.indel.ini`
- `<analysis>/tertiary/annotation.small.indel.ini`

Table 81 describes annotations parameters that are automatically specified by LifeScope™ Software when run as part of a standard workflow.

### Table 80 Annotation parameters description (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
</table>
| annotation.show.only.variants.not.in.dbsnp | 0 | When set to 1, filters the annotated output: restricts the Filtered Variant annotation output file (see “Annotation output files” on page 303) to only SNPs or small indels that do not exist in dbSNP. Allowed values:  
  - 0: Do not filter the output variants that do not appear in dbSNP.  
  - 1: Restrict output to SNPs or small indels variants that do not appear in dbSNP. |

**Note:** applies only to SNPs and small indels modules

---

### Table 81 Annotations internal parameter description

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
</table>
| annotation.run            | 1             | Specifies whether or not to add annotations. Allowed values:  
  - 0: Do not generate annotations.  
  - 1: Generate annotations during this analysis. The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use. |
| annotation.dbsnp.file     | —             | The path to your input dbSNP file or files. |
| annotation.gtf.file       | —             | The path to your input genome annotation file. |
| annotation.output.file.name.stem | The base name of the input GFF file (without directory path and without file extension). | The common prefix for all output file names. Supplied by the system when run in a standard workflow. |
Output annotation

This section describes the new attributes that annotation adds to the GFF entries from the input file. The added information comes from publicly-available sources about variants in the input, features intersecting the variants in the input file, or biological function potentially changed by the variants.

Table 82 describes the new attributes added as output annotation.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
</table>
| geneID | Adds a geneID attribute to applicable entries in the output GFF file. The following apply to the geneID attribute:  
  • The attribute is valid for all variants.  
  • The attribute’s value is the gene descriptor taken from the input GTF file.  
  • If there are multiple overlapping genes, the attribute has multiple values (in a comma-separated list). |

<table>
<thead>
<tr>
<th>Supported modules</th>
<th>Requirement to qualify for annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNPs</td>
<td>The variant must be within a gene (including introns and UTRs).</td>
</tr>
<tr>
<td>Small indels</td>
<td></td>
</tr>
<tr>
<td>Large indels</td>
<td></td>
</tr>
<tr>
<td>CNVs</td>
<td></td>
</tr>
</tbody>
</table>

| exonID | Adds exonID attributes to applicable entries in the output GFF file. The following apply to the exonID attribute:  
  • The attribute is valid for all variants.  
  • The attribute’s value is of the form `<geneID>-<Exon#>`, where `Exon#` is the number of the exon counted within the gene (not within each transcript) in the transcription direction.  
  • If there are multiple overlapping exons, the attribute has multiple values (in a comma-separated list).  
  • If two exons start from the same position, the shorter one is listed first. |

<table>
<thead>
<tr>
<th>Supported modules</th>
<th>Requirement to qualify for annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNPs</td>
<td>The variant must overlap an exon.</td>
</tr>
<tr>
<td>Small indels</td>
<td></td>
</tr>
<tr>
<td>Large indels</td>
<td></td>
</tr>
<tr>
<td>CNVs</td>
<td></td>
</tr>
</tbody>
</table>

| rsID | Adds rsID attributes to applicable entries in the output GFF file. The following apply to the rsID attribute:  
  • The attribute is valid for SNPs and Indels.  
  • The attribute’s value is the rsID value of the dbSNP entry.  
  • The attribute cannot have multiple values. |

<table>
<thead>
<tr>
<th>Supported modules</th>
<th>Requirement to qualify for annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNPs</td>
<td>The SNP must appear in dbSNP.</td>
</tr>
<tr>
<td>Small indels</td>
<td>Indels must appear in dbSNP and also must be labeled as InDel in dbSNP.</td>
</tr>
</tbody>
</table>
Chapter 15  Add Genomic Annotations to Analysis Results

Output annotation

Table 82  New attributes added as annotations (continued)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>functionCode</td>
<td>Based on functional codes from dbSNP, annotate SNP entries in the output GFF file. The following apply to the SNPType attribute:</td>
</tr>
<tr>
<td></td>
<td>• The attribute is valid for only for SNPs.</td>
</tr>
<tr>
<td></td>
<td>• The attribute’s value is a pair in which the first value is the locus ID and the second the function code from dbSNP. Function codes are listed in Table 83.</td>
</tr>
<tr>
<td></td>
<td>• The attribute can have multiple values, one for each SNP type that applies to the variant.</td>
</tr>
</tbody>
</table>

Supported modules

<table>
<thead>
<tr>
<th>Requirement to qualify for annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNPs</td>
</tr>
<tr>
<td>Small indels</td>
</tr>
</tbody>
</table>

Table 83 describes function codes for refSNPs in gene features. This table is taken from the dbSNP website:


Table 83  Function codes for refSNPs in gene features

<table>
<thead>
<tr>
<th>Functional class</th>
<th>Description</th>
<th>Database code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locus region</td>
<td>Variation is within 2 kbp 5’ or 500 bp 3’ of a gene feature (on either strand), but the variation is not in the transcript for the gene. This class is indicated with an “L” in graphical summaries. Note: As of build 127, function code 1 has been modified into a two-digit code that more precisely indicates the location of a SNP. The two-digit code has function code 1 as the first digit, which keeps the meaning as described above, and 3 or 5 as the second digit, which indicates whether the SNP is 3’ or 5’ to the region of interest. See function codes 13 and 15 in this table.</td>
<td>1</td>
</tr>
<tr>
<td>Coding</td>
<td>Variation is in the coding region of the gene. This class is assigned if the allele-specific class is unknown. This class is indicated with a ’C’ in graphical summaries. Note: This code was retired as of dbSNP build 127.</td>
<td>2</td>
</tr>
<tr>
<td>Coding-synon</td>
<td>The variation allele is synonymous with the contig codon in a gene. An allele receives this class when substitution and translation of the allele into the codon makes no change to the amino acid specified by the reference sequence. A variation is a synonymous substitution if all alleles are classified as contig reference or coding-synon. This class is indicated with a ’C’ in graphical summaries.</td>
<td>3</td>
</tr>
</tbody>
</table>
Coding-nonsynon | The variation allele is nonsynonymous for the contig codon in a gene. An allele receives this class when substitution and translation of the allele into the codon changes the amino acid specified by the reference sequence. A variation is a nonsynonymous substitution if any alleles are classified as coding-nonsynon. This class is indicated with a “C” or “N” in graphical summaries.

**Note:** As of build 128, function code 4 has been modified into a two-digit code that more precisely indicates the nonsynonymous nature of the SNP. The two-digit code has function code 4 as the first digit, which keeps its original meaning, and 1, 2, or 4 as the second digit, which indicates whether the SNP is nonsense, missense, or frameshift. See function codes 41, 42, and 44 in this table.

mRNA-UTR | The variation is in the transcript of a gene but not in the coding region of the transcript. This class is indicated by a “T” in graphical summaries.

**Note:** As of build 127, function code 5 has been modified into a two-digit code that more precisely indicates the location of a SNP. The two digit code has function code 5 as the first digit, which keeps its original meaning, and 3 or 5 as the second digit, which indicates whether the SNP is 3’ or 5’ to the region of interest. See function codes 53 and 55 in this table.

Intron | The variation is in the intron of a gene but not in the first two or last two bases of the intron. This class is indicated by an “L” in graphical summaries.

Splice-site | The variation is in the first two or last two bases of the intron. This class is indicated by a “T” in graphical summaries.

**Note:** As of build 127, function code 7 has been modified into a two-digit code that more precisely indicates the location of a SNP. The two-digit code has function codes 7 as the first digit, which keeps its original meaning, and 3 or 5 as the second digit, which indicates whether the SNP is 3’ or 5’ to the region of interest. See function codes 73 and 75 in this table.

Contig-reference | The variation allele is identical to the contig nucleotide. Typically, one allele of a variation is the same as the reference genome. The letter used to indicate the variation is a “C” or “N”, depending on the state of the alternative allele for the variation.

Coding-exception | The variation is in the coding region of a gene, but the precise location cannot be resolved because of an error in the alignment of the exon. The class is indicated by a “C” in graphical summaries.

NearGene–3 | Function Code 13, where:
1: locus region [see function code 1 in this table]
3: SNP is 3’ to and 0.5 kbp away from gene

NearGene–5 | Function Code 15, where:
1: locus region [see function code 1 in this table]
3: SNP is 5’ to and 0.5 kbp away from gene

Coding-nonsynonymous nonsense | Function Code 41, where:
4: Coding-nonsynonymous [see function code 4 in this table]
1: Nonsense (changes to the Stop codon)
Table 83  Function codes for refSNPs in gene features (continued)

<table>
<thead>
<tr>
<th>Functional class</th>
<th>Description</th>
<th>Database code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding-nonsynonymous missense</td>
<td>Function Code 42, where: 4: Coding-nonsynonymous (see function code 4 in this table) 2: missense (alters codon to make an altered amino acid in the protein product)</td>
<td>42</td>
</tr>
<tr>
<td>Coding-nonsynonymous frameshift</td>
<td>Function Code 44, where: 4: Coding-nonsynonymous (see function code 4 in this table) 4 (as the second digit): frameshift (alters codon to make an altered amino acid in protein product)</td>
<td>44</td>
</tr>
<tr>
<td>UTR–3</td>
<td>Function code 53, where: 5: UTR (untranslated region: see function code 5 in this table) 3: SNP located in the 3′ untranslated region</td>
<td>53</td>
</tr>
<tr>
<td>UTR–5</td>
<td>Function code 55, where: 5: UTR (untranslated region: see function code 5 in this table) 5 (as the second digit): SNP located in the 5′ untranslated region</td>
<td>55</td>
</tr>
<tr>
<td>Splice–3</td>
<td>Function code 73, where: 7: splice site (see function code 7 in this table) 3: 3′ acceptor dinucleotide</td>
<td>73</td>
</tr>
<tr>
<td>Splice–5</td>
<td>Function code 75, where: 7: splice site (see function code 7 in this table) 5: 5′ donor dinucleotide</td>
<td>75</td>
</tr>
</tbody>
</table>

Annotation output files

These types of output files are potentially generated by the annotations processing, depending on the LifeScope™ Software modules being run:

- Annotated Variants
- Filter Variants
- Variant Statistics
- Mutated Genes
- diBayes tab-delimited output file
- diBayes annotated tab-delimited output file

Not all output files are generated on each run. Which files are produced depends on the type of analysis modules being run and on the annotation options you select. See “About annotations and LifeScope™ Software modules” on page 318 for more information on annotations available with the different modules, and “Annotation parameters” on page 297 for a description of annotation options.

The output file content is typically labeled within each file, or as described in this chapter in the sections concerning each annotation output file type.
Chapter 15  Add Genomic Annotations to Analysis Results

Annotation output files

For filtered output files, all annotation-filter options that are turned on must be fulfilled by the variant entry for the variant to appear in the output file.

The Annotated Variants and Filtered Variants output files are the input GFF files (see “Annotation input files” on page 295) with annotations added.

An Annotated Variants output file contains all the entries of the input GFF file. A Filtered Variants output file contains only those entries that fulfill all the conditions you specify with the annotation UI options.

The Annotated Variants and Filtered Variants output files are:

- In GFF v3 format
- Named according to these filename patterns:
  \(<\text{annotation.output.file.name.stem}_\text{annotated}.\text{gff}\>
  \(<\text{annotation.output.file.name.stem}_\text{annotated_filtered}.\text{gff}\>

where \(<\text{annotation.output.file.name.stem}>\) is a common prefix generated by the annotations module and used with all output files from the same run.

Table 84 describes the additional annotation attributes added to the last (ninth) column of the Annotated Variants and Filtered Variants files. Except for rsID, attributes can have multiple values listed as a comma-separated list.

**Table 84 The Annotated Variant and Filtered Variant output file added fields**

<table>
<thead>
<tr>
<th>Attribute/Metadata</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Metadata keyword</strong></td>
<td></td>
</tr>
</tbody>
</table>
| Annotation          | A metadata line beginning with the word “Annotation” and followed by these tab-separated fields:  
  • The date and the sources of annotation used  
  • The path to the flat files  
  • The dbSNP version, if available  
  • The human reference build, if available |
| Filtering           | A metadata line beginning with the word “Filtering” and followed by these tab-separated fields:  
  • The values “no filtering” or “show only”  
  • One or more of the values for “coding variants”, “variants in genes”, “variants in dbSNP”, and “variants not in dbSNP” |
| Attributes          |             |
| geneID              | Add a geneID attribute to applicable entries in the output GFF file.  
  The following apply to the geneID attribute:  
  • The attribute is valid for all variants.  
  • The attribute’s value is the gene descriptor taken from the input GTF file.  
  If there are multiple overlapping genes, the attribute has multiple values (in a comma-separated list). |
Table 84 The Annotated Variant and Filtered Variant output file added fields (continued)

<table>
<thead>
<tr>
<th>Attribute/Metadata</th>
<th>Description</th>
</tr>
</thead>
</table>
| exonID             | Adds exonID attributes to applicable entries in the output GFF file. The following apply to the exonID attribute:  
• The attribute is valid for all variants.  
• The attribute’s value is of the form `<GeneID>-<Exon#>`, where Exon# is the number of the exon counted within the gene (not within each transcript) in the transcription direction.  
• If there are multiple overlapping exons, the attribute has multiple values (in a comma-separated list).  
• If two exons start from the same position, the shorter one is listed first. |
| rsID               | Adds rsID attributes to applicable entries in the output GFF file. The following apply to the rsID attribute:  
• The attribute is valid for SNPs and Indels.  
• The attribute’s value is the rsID value of the dbSNP entry.  
• The attribute cannot have multiple values. |
| functionCode      | Based on functional codes from dbSNP, annotate SNP entries in the output GFF file. The following apply to the functionCode attribute:  
• The attribute is valid for only for SNPs.  
• The attribute’s value is a pair in which the first value is the locus ID and the second the function code from dbSNP. Function codes are listed in Table 83.  
• The attribute can have multiple values, one for each SNP type that applies to the variant. |

Variant Statistics output file for SNPs

A Variant Statistics file is a text file containing the statistics about the variants in the input GFF file. The file is named according to this filename pattern:

<annotation.output.file.name.stem>.stats

where `<annotation.output.file.name.stem>` is a common prefix generated by the annotations module and used with all output files from the same run.

The following is an example of the content of a Variant Statistics file for SNPs:

************************************************************
LIFESCOPE REPORT
************************************************************

Input File:/data/swdev/minita_test/solid0064_20101210_MP_2x60_T3_1/2.0/color_lmp/outputs/dibayes/color_lmp_SNP.gff3  
Date:Apr 14, 2011 2:17:54 PM  
Annotation file, dbSNP:/data/analysis/LifeScope_resources/lifetech/hg18/dbSNP/dbSNP_b130.tab
Annotation file, Genes and Exons, GTF:/data/analysis/LifeScope_resources/lifetech/hg18/GTF/refGene.20090513.gtf

************************************************************
Statistics Overview
************************************************************

-- Basic Statistics ----------------------------------------

Number of variants                                   3072881
Number of heterozygous variants                      1639649
Number of homozygous variants                        1433232

-- Variant-Specific Statistics (SNP) ---------------------

Number of transition SNPs                            2070418
Number of transition heterozygous SNPs               1103589
Number of transition homozygous SNPs                  966829
Number of transversion SNPs                          1002463
Number of transversion heterozygous SNPs              536060
Number of transversion homozygous SNPs                466403
Transition:Transversion ratio                  2.065 : 1.000

-- dbSNP Annotation Statistics --------------------------

Number of SNPs in dbSNP                        2913081
Number of heterozygous SNPs in dbSNP            1506740
Number of homozygous SNPs in dbSNP              1406341
dbSNP concordance                               94.80%
dbSNP heterozygous concordance                  91.89%
dbSNP homozygous concordance                    98.12%

-- Per-Chromosome Statistics --------------------------

nVar: Number of variants
nHetVar: Number of heterozygous variants
nHomVar: Number of homozygous variants

<table>
<thead>
<tr>
<th>contig</th>
<th>nVar</th>
<th>nHetVar</th>
<th>nHomVar</th>
</tr>
</thead>
<tbody>
<tr>
<td>chr1</td>
<td>233502</td>
<td>125860</td>
<td>107642</td>
</tr>
<tr>
<td>chr2</td>
<td>247682</td>
<td>132717</td>
<td>114965</td>
</tr>
<tr>
<td>chr3</td>
<td>210137</td>
<td>115661</td>
<td>94476</td>
</tr>
<tr>
<td>chr4</td>
<td>225996</td>
<td>118326</td>
<td>107670</td>
</tr>
<tr>
<td>chr5</td>
<td>185137</td>
<td>102825</td>
<td>82312</td>
</tr>
<tr>
<td>chr6</td>
<td>191793</td>
<td>104138</td>
<td>87655</td>
</tr>
<tr>
<td>chr7</td>
<td>170500</td>
<td>94092</td>
<td>76408</td>
</tr>
<tr>
<td>chr8</td>
<td>160126</td>
<td>86553</td>
<td>73573</td>
</tr>
<tr>
<td>chr9</td>
<td>128205</td>
<td>70291</td>
<td>57914</td>
</tr>
<tr>
<td>chr10</td>
<td>157345</td>
<td>85408</td>
<td>71937</td>
</tr>
<tr>
<td>chr11</td>
<td>154271</td>
<td>80792</td>
<td>73479</td>
</tr>
<tr>
<td>chr12</td>
<td>139569</td>
<td>72915</td>
<td>66654</td>
</tr>
<tr>
<td>chr13</td>
<td>118842</td>
<td>60095</td>
<td>58747</td>
</tr>
<tr>
<td>chr14</td>
<td>93526</td>
<td>48742</td>
<td>44784</td>
</tr>
</tbody>
</table>
A Variant Statistics file is a text file containing the statistics about the variants in the input GFF file. The file is named according to this filename pattern:

\<annotation.output.file.name.stem\>_stats

where \<annotation.output.file.name.stem\> is a common prefix generated by the annotations module and used with all output files from the same run.

The following is an example of the content of a Variant Statistics file for small indels:

**************************************************************
LIFESCOPE REPORT
**************************************************************

Input File:/data/swdev/minita_test/
solid0064_20101210_MP_2x60_T3_1/2.0/color_lmp/outputs/
small.indel/color_lmp.gff3
Date:Apr 14, 2011 10:01:01 AM
Annotation file, dbSNP:/data/analysis/LifeScope_resources/
lifetech/hg18/dbSNP/dbSNP_b130.tab
Annotation file, Genes and Exons, GTF:/data/analysis/
LifeScope_resources//lifetech/hg18/GTF/refGene.20090513.gtf

**************************************************************
Statistics Overview
**************************************************************

-- Basic Statistics ----------------------------------------
Number of variants                                      296183
Number of heterozygous variants                         194752
Number of homozygous variants                           100934

-- Variant-Specific Statistics (InDel) -------------------
Distribution of InDel length

<table>
<thead>
<tr>
<th>Length (bases)</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>(-Inf:-500]</td>
<td>0</td>
</tr>
<tr>
<td>(-500:-490]</td>
<td>14</td>
</tr>
<tr>
<td>(-490:-480]</td>
<td>8</td>
</tr>
<tr>
<td>(-480:-470]</td>
<td>13</td>
</tr>
<tr>
<td>(-470:-460]</td>
<td>21</td>
</tr>
</tbody>
</table>
Chapter 15  Add Genomic Annotations to Analysis Results

Annotation output files

<table>
<thead>
<tr>
<th>Region</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>(-460:-450]</td>
<td>14</td>
</tr>
<tr>
<td>(-450:-440]</td>
<td>18</td>
</tr>
<tr>
<td>(-440:-430]</td>
<td>17</td>
</tr>
<tr>
<td>(-430:-420]</td>
<td>18</td>
</tr>
<tr>
<td>(-420:-410]</td>
<td>18</td>
</tr>
<tr>
<td>(-410:-400]</td>
<td>15</td>
</tr>
<tr>
<td>(-400:-390]</td>
<td>24</td>
</tr>
<tr>
<td>(-390:-380]</td>
<td>21</td>
</tr>
<tr>
<td>(-380:-370]</td>
<td>24</td>
</tr>
<tr>
<td>(-370:-360]</td>
<td>30</td>
</tr>
<tr>
<td>(-360:-350]</td>
<td>39</td>
</tr>
<tr>
<td>(-350:-340]</td>
<td>62</td>
</tr>
<tr>
<td>(-340:-330]</td>
<td>115</td>
</tr>
<tr>
<td>(-330:-320]</td>
<td>202</td>
</tr>
<tr>
<td>(-320:-310]</td>
<td>181</td>
</tr>
<tr>
<td>(-310:-300]</td>
<td>80</td>
</tr>
<tr>
<td>(-300:-290]</td>
<td>43</td>
</tr>
<tr>
<td>(-290:-280]</td>
<td>52</td>
</tr>
<tr>
<td>(-280:-270]</td>
<td>42</td>
</tr>
<tr>
<td>(-270:-260]</td>
<td>56</td>
</tr>
<tr>
<td>(-260:-250]</td>
<td>49</td>
</tr>
<tr>
<td>(-250:-240]</td>
<td>35</td>
</tr>
<tr>
<td>(-240:-230]</td>
<td>39</td>
</tr>
<tr>
<td>(-230:-220]</td>
<td>35</td>
</tr>
<tr>
<td>(-220:-210]</td>
<td>33</td>
</tr>
<tr>
<td>(-210:-200]</td>
<td>37</td>
</tr>
<tr>
<td>(-200:-190]</td>
<td>34</td>
</tr>
<tr>
<td>(-190:-180]</td>
<td>30</td>
</tr>
<tr>
<td>(-180:-170]</td>
<td>40</td>
</tr>
<tr>
<td>(-170:-160]</td>
<td>49</td>
</tr>
<tr>
<td>(-160:-150]</td>
<td>42</td>
</tr>
<tr>
<td>(-150:-140]</td>
<td>55</td>
</tr>
<tr>
<td>(-140:-130]</td>
<td>78</td>
</tr>
<tr>
<td>(-130:-120]</td>
<td>55</td>
</tr>
<tr>
<td>(-120:-110]</td>
<td>68</td>
</tr>
<tr>
<td>(-110:-100]</td>
<td>51</td>
</tr>
<tr>
<td>(-100:-90]</td>
<td>60</td>
</tr>
<tr>
<td>(-90:-80]</td>
<td>61</td>
</tr>
<tr>
<td>(-80:-70]</td>
<td>94</td>
</tr>
<tr>
<td>(-70:-60]</td>
<td>107</td>
</tr>
<tr>
<td>(-60:-50]</td>
<td>115</td>
</tr>
<tr>
<td>(-50:-40]</td>
<td>229</td>
</tr>
<tr>
<td>(-40:-30]</td>
<td>332</td>
</tr>
<tr>
<td>(-30:-20]</td>
<td>1064</td>
</tr>
<tr>
<td>(-20:-19]</td>
<td>220</td>
</tr>
<tr>
<td>(-19:-18]</td>
<td>392</td>
</tr>
<tr>
<td>(-18:-17]</td>
<td>268</td>
</tr>
<tr>
<td>(-17:-16]</td>
<td>568</td>
</tr>
<tr>
<td>(-16:-15]</td>
<td>436</td>
</tr>
<tr>
<td>(-15:-14]</td>
<td>624</td>
</tr>
<tr>
<td>(-14:-13]</td>
<td>541</td>
</tr>
<tr>
<td>(-13:-12]</td>
<td>1126</td>
</tr>
<tr>
<td>(-12:-11]</td>
<td>1009</td>
</tr>
<tr>
<td>(-11:-10]</td>
<td>1636</td>
</tr>
</tbody>
</table>
Chapter 15  Add Genomic Annotations to Analysis Results

Annotation output files

-- dbSNP Annotation Statistics -------------------------------
Number of InDels in dbSNP 246619
Number of heterozygous InDels in dbSNP 150203
Number of homozygous InDels in dbSNP 96105
dbSNP concordance 83.27%
dbSNP heterozygous concordance 77.13%
dbSNP homozygous concordance 95.22%

-- Per-Chromosome Statistics -------------------------------

nVar: Number of variants
nHetVar: Number of heterozygous variants
nHomVar: Number of homozygous variants

<table>
<thead>
<tr>
<th>contig</th>
<th>nVar</th>
<th>nHetVar</th>
<th>nHomVar</th>
</tr>
</thead>
<tbody>
<tr>
<td>chr1</td>
<td>22753</td>
<td>15323</td>
<td>7385</td>
</tr>
<tr>
<td>chr2</td>
<td>24215</td>
<td>16211</td>
<td>7975</td>
</tr>
<tr>
<td>chr3</td>
<td>20327</td>
<td>13613</td>
<td>6686</td>
</tr>
<tr>
<td>chr4</td>
<td>22322</td>
<td>14798</td>
<td>7485</td>
</tr>
</tbody>
</table>
### Annotation output files

#### Variant Statistics output file for large indels

A Variant Statistics file is a text file containing the statistics about the variants in the input GFF file.

The file is named according to this filename pattern:

```
<annotation.output.file.name.stem>_stats
```

where `<annotation.output.file.name.stem>` is a common prefix generated by the annotations module and used with all output files from the same run.

The following is an example of the content of a Variant Statistics file for large indels:

```
************************************************************
LIFESCOPE REPORT
************************************************************

Input File:/data/swdev/minita_test/
solid0064_20101210_MP_2x60_T3_1/2.0/color_lmp/outputs/
large.indel/large-indels.gff3
Date:Apr 14, 2011 10:00:05 AM
Annotation file, dbSNP:/data/analysis/LifeScope_resources/
lifetech/hg18/dbSNP/dbSNP_b130.tab
Annotation file, Genes and Exons, GTF:/data/analysis/
LifeScope_resources//lifetech/hg18/GTF/refGene.20090513.gtf

********************************************************************
Statistics Overview
********************************************************************

-- Basic Statistics

Number of variants: 21
```

<table>
<thead>
<tr>
<th>chr</th>
<th>Start</th>
<th>End</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>chr5</td>
<td>18224</td>
<td>12212</td>
<td>5989</td>
</tr>
<tr>
<td>chr6</td>
<td>19046</td>
<td>12807</td>
<td>6213</td>
</tr>
<tr>
<td>chr7</td>
<td>16887</td>
<td>11032</td>
<td>5824</td>
</tr>
<tr>
<td>chr8</td>
<td>14530</td>
<td>9625</td>
<td>4899</td>
</tr>
<tr>
<td>chr9</td>
<td>11539</td>
<td>7926</td>
<td>3601</td>
</tr>
<tr>
<td>chr10</td>
<td>14864</td>
<td>9836</td>
<td>5000</td>
</tr>
<tr>
<td>chr11</td>
<td>14951</td>
<td>9502</td>
<td>5428</td>
</tr>
<tr>
<td>chr12</td>
<td>14000</td>
<td>9180</td>
<td>4793</td>
</tr>
<tr>
<td>chr13</td>
<td>11885</td>
<td>7685</td>
<td>4184</td>
</tr>
<tr>
<td>chr14</td>
<td>9439</td>
<td>6105</td>
<td>3323</td>
</tr>
<tr>
<td>chr15</td>
<td>8945</td>
<td>5912</td>
<td>3022</td>
</tr>
<tr>
<td>chr16</td>
<td>7705</td>
<td>5304</td>
<td>2382</td>
</tr>
<tr>
<td>chr17</td>
<td>7849</td>
<td>5323</td>
<td>2508</td>
</tr>
<tr>
<td>chr18</td>
<td>8605</td>
<td>5530</td>
<td>3066</td>
</tr>
<tr>
<td>chr19</td>
<td>5866</td>
<td>3825</td>
<td>2026</td>
</tr>
<tr>
<td>chr20</td>
<td>6107</td>
<td>4120</td>
<td>1977</td>
</tr>
<tr>
<td>chr21</td>
<td>5067</td>
<td>2977</td>
<td>2072</td>
</tr>
<tr>
<td>chr22</td>
<td>3587</td>
<td>2301</td>
<td>1280</td>
</tr>
<tr>
<td>chrX</td>
<td>6351</td>
<td>2650</td>
<td>3675</td>
</tr>
<tr>
<td>chrY</td>
<td>1118</td>
<td>954</td>
<td>151</td>
</tr>
<tr>
<td>chrM</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Number of heterozygous variants 2
Number of homozygous variants 19

-- Variant-Specific Statistics (InDel) -------------------------------
Distribution of InDel length

<table>
<thead>
<tr>
<th>Length (bases)</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>(-Inf:-500000]</td>
<td>0</td>
</tr>
<tr>
<td>(-500000:-100000]</td>
<td>0</td>
</tr>
<tr>
<td>(-100000:-500000]</td>
<td>0</td>
</tr>
<tr>
<td>(-50000:-200000]</td>
<td>0</td>
</tr>
<tr>
<td>(-20000:-100000]</td>
<td>0</td>
</tr>
<tr>
<td>(-10000:-50000]</td>
<td>0</td>
</tr>
<tr>
<td>(-5000:-10000]</td>
<td>0</td>
</tr>
<tr>
<td>(-1000:-5000]</td>
<td>0</td>
</tr>
<tr>
<td>(-500:-10000]</td>
<td>0</td>
</tr>
<tr>
<td>(-1000:-5000]</td>
<td>0</td>
</tr>
<tr>
<td>(0:1000]</td>
<td>0</td>
</tr>
<tr>
<td>(100:2000]</td>
<td>0</td>
</tr>
<tr>
<td>(200:5000]</td>
<td>0</td>
</tr>
<tr>
<td>(500:10000]</td>
<td>0</td>
</tr>
<tr>
<td>(10000:50000]</td>
<td>0</td>
</tr>
<tr>
<td>(50000:100000]</td>
<td>0</td>
</tr>
<tr>
<td>(100000:500000]</td>
<td>0</td>
</tr>
<tr>
<td>(500000:+Inf)</td>
<td>0</td>
</tr>
</tbody>
</table>

-- dbSNP Annotation Statistics -------------------------------------

Number of InDels in dbSNP 0
Number of heterozygous InDels in dbSNP 0
Number of homozygous InDels in dbSNP 0
dbSNP concordance 0.00%
dbSNP heterozygous concordance 0.00%
dbSNP homozygous concordance 0.00%

-- Per-Chromosome Statistics -------------------------------------

nVar: Number of variants
nHetVar: Number of heterozygous variants
nHomVar: Number of homozygous variants

<table>
<thead>
<tr>
<th>contig</th>
<th>nVar</th>
<th>nHetVar</th>
<th>nHomVar</th>
</tr>
</thead>
<tbody>
<tr>
<td>chr2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>chr9</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>chr10</td>
<td>7</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>chr18</td>
<td>4</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>chr19</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>chrY</td>
<td>4</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>chrM</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>
Variant Statistics output file for CNVs

A Variant Statistics file is a text file containing the statistics about the variants in the input GFF file.

The file is named according to this filename pattern:

```
<annotation.output.file.name.stem>_stats
```

where `<annotation.output.file.name.stem>` is a common prefix generated by the annotations module and used with all output files from the same run.

The following is an example of the content of a Variant Statistics file for CNVs:

```
************************************************************
LIFESCOPE REPORT
************************************************************

Input File:/data/swdev/minita_test/
solid0064_20101210_MP_2x60_T3_1/2.0/color_lmp/outputs/cnv/
OutputCNVs.gff3
Date:Apr 14, 2011 9:58:12 AM
Annotation file, dbSNP:/data/analysis/LifeScope_resources/
lifetech/hg18/dbSNP/dbSNP_b130.tab
Annotation file, Genes and Exons, GTF:/data/analysis/
LifeScope_resources//lifetech/hg18/GTF/refGene.20090513.gtf

************************************************************
Statistics Overview
************************************************************

-- Basic Statistics ------------------------------------------
Number of variants                                       120

-- Variant-Specific Statistics (CNV) ------------------------

Distribution of CNV copy number
Copy_Number      Number
0               5
1               20
2               23
3               20
4               22
5               8
6               9
7               1
8               2
9               10
>9              0

Distribution of CNV length
Length(bases)    Number
(0:1000]         0
(1000:2000]      0
```
Chapter 15 Add Genomic Annotations to Analysis Results

Annotation output files

(2000:5000] 0
(5000:10000] 0
(10000:20000] 41
(20000:50000] 54
(50000:100000] 11
(100000:500000] 10
(500000:+Inf) 4

-- Per-Chromosome Statistics -----------------------------------

nVar: Number of variants

<table>
<thead>
<tr>
<th>contig</th>
<th>nVar</th>
</tr>
</thead>
<tbody>
<tr>
<td>chr1</td>
<td>7</td>
</tr>
<tr>
<td>chr2</td>
<td>5</td>
</tr>
<tr>
<td>chr3</td>
<td>11</td>
</tr>
<tr>
<td>chr4</td>
<td>4</td>
</tr>
<tr>
<td>chr5</td>
<td>7</td>
</tr>
<tr>
<td>chr6</td>
<td>11</td>
</tr>
<tr>
<td>chr7</td>
<td>4</td>
</tr>
<tr>
<td>chr8</td>
<td>5</td>
</tr>
<tr>
<td>chr9</td>
<td>3</td>
</tr>
<tr>
<td>chr10</td>
<td>3</td>
</tr>
<tr>
<td>chr11</td>
<td>8</td>
</tr>
<tr>
<td>chr12</td>
<td>4</td>
</tr>
<tr>
<td>chr13</td>
<td>3</td>
</tr>
<tr>
<td>chr14</td>
<td>2</td>
</tr>
<tr>
<td>chr15</td>
<td>4</td>
</tr>
<tr>
<td>chr16</td>
<td>2</td>
</tr>
<tr>
<td>chr17</td>
<td>4</td>
</tr>
<tr>
<td>chr19</td>
<td>4</td>
</tr>
<tr>
<td>chr20</td>
<td>1</td>
</tr>
<tr>
<td>chr22</td>
<td>2</td>
</tr>
<tr>
<td>chrX</td>
<td>17</td>
</tr>
<tr>
<td>chrY</td>
<td>9</td>
</tr>
</tbody>
</table>

Mutated Genes output file

The Mutated Genes output file contains the list of genes whose coding regions were modified by at least one of the variants in the input file. The Mutated Genes file is a text file in tab-separated format. The file contains an entry per row, and one gene per entry. The first 12 columns are the same as the columns in the BED file (genes.bed). Additional columns are added by the annotations processing.

The file is named according to this filename pattern:

<annotation.output.file.name.stem>_genes.tab

where <annotation.output.file.name.stem> is a common prefix generated by the annotations module and used with all output files from the same run.
Table 85 lists the information included in the Mutated Genes output file.

Table 85 Description of the Mutated Genes output file

<table>
<thead>
<tr>
<th>Field number</th>
<th>Field name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Chromosome/Contig</td>
<td>The contig of the gene.</td>
</tr>
<tr>
<td>2</td>
<td>Start</td>
<td>The left-most coordinate of the gene.</td>
</tr>
<tr>
<td>3</td>
<td>End</td>
<td>The right-most coordinate of the gene.</td>
</tr>
<tr>
<td>4</td>
<td>GeneID</td>
<td>The gene identifier as provided by the GTF file.</td>
</tr>
<tr>
<td>5</td>
<td>Score</td>
<td>A score between 0 and 1000. The score is a normalized value for the number of coding variants such that all numbers scale between 0 and 1000. The score is equal to the number of coding variants for this entry divided by the maximum number of coding variants for any gene.</td>
</tr>
<tr>
<td>6</td>
<td>Strand</td>
<td>The strand of the gene.</td>
</tr>
<tr>
<td>7</td>
<td>Start</td>
<td>The left-most coordinate of the gene.</td>
</tr>
<tr>
<td>8</td>
<td>End</td>
<td>The right-most coordinate of the gene.</td>
</tr>
<tr>
<td>9</td>
<td>Variant color</td>
<td>The color depends on the type of the variant that overlaps this gene:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Red: SNP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Green: Indel (small indels, large indels)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cyan: CNV</td>
</tr>
<tr>
<td>10</td>
<td>Number of exons</td>
<td>The number of exons of the gene.</td>
</tr>
<tr>
<td>11</td>
<td>Exon start position list</td>
<td>A comma-separated list of start-coordinates for exons in the gene. All coordinates are relative to the start position of the gene.</td>
</tr>
<tr>
<td>12</td>
<td>Exon length list</td>
<td>A comma-separated list of the lengths of the exons in the gene.</td>
</tr>
<tr>
<td>13</td>
<td>Total exon length</td>
<td>The sum the length of all exons.</td>
</tr>
<tr>
<td>14</td>
<td>Number of coding variants</td>
<td>The number of variants overlapping at least one exon.</td>
</tr>
<tr>
<td>15</td>
<td>Number of variants</td>
<td>The number of variants overlapping the gene.</td>
</tr>
<tr>
<td>16</td>
<td>Variant type</td>
<td>The type of the variant. (One type is listed per line.)</td>
</tr>
<tr>
<td>17</td>
<td>Variant start list</td>
<td>A comma-separated list for start coordinates of variants overlapping the gene. All coordinates are relative to the start position of the gene.</td>
</tr>
<tr>
<td>18</td>
<td>Variant length list</td>
<td>A comma-separated list of lengths of the variants overlapping the gene.</td>
</tr>
</tbody>
</table>

Potential uses of a Mutated Genes file

You can use a spreadsheet application or scripts to remove unwanted columns. First save a backup of your output file. The following are examples of uses for a Mutated Genes file:

- The first 12 columns provide information for visualization of modified genes and exons.
- Column 4, the GeneID, provides a list of modified genes. This column provides a gene list to focus on for enrichment analysis.
- A combination of the first 10 and the last 2 fields provides information useful for visualizing variants that overlap the gene.
Fields 14 and 15 provide the number of variants that overlap exons and the number of variants that overlap the gene. With these columns you can filter by the number of variants overlapping the exons in the gene. With the GeneID in column 4, this information identifies genes to focus on for further analysis.

(Optional) View a Mutated Genes output file

Open the BED file in the genomic browser.

Example of a Mutated Genes output file

The following is a truncated example (3 lines) of a Mutated Genes output file:

Chromosome/Contig Start End GeneID Score Strand Start End Variant color Number of Exons Exon start position list Exon length list Total exon length Number of coding variants Number of variants Variant type Variant start list Variant length list

chr1 752927 779603 LOC643837 0 + 752927 779603 255,0,0 7 0,1319,19970,24243,24987,25707,25893 1543 2 15 SNP
3041,5189,6765,12362,13482,16395,16818,17512,17957,18167 60,92,182,51,125,90,186,163,116,79,500,125,111,674 2554 1 8 SNP
3041,5189,6765,12362,13482,16395,16818,17512,17957,18167 60,92,182,51,125,90,186,163,116,79,500,125,111,674 2554 1 8 SNP

The first eight columns of this output file are the same as the input GFF file. The ninth column in the input file is replaced by individual columns for each attribute, as shown in Table 86. This output file is generated only on SNP's module runs.

This is a tab-separated text file. The file is named according to this filename pattern:

<annotation.output.file.name.stem>.tab

where <annotation.output.file.name.stem> is a common prefix generated by the annotations module and used with all output files from the same run.

Table 86 Attributes in the ninth column

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seqid</td>
<td>The string ID of the sequence to which the start and end coordinates refer.</td>
</tr>
<tr>
<td>Source</td>
<td>The source of the data.</td>
</tr>
<tr>
<td>Type</td>
<td>Sequence ontology derived type for this variation. For diBayes, this is always SNP.</td>
</tr>
<tr>
<td>Start</td>
<td>Start position of the SNP.</td>
</tr>
</tbody>
</table>
Chapter 15  Add Genomic Annotations to Analysis Results

Annotation output files

Table 86  Attributes in the ninth column (continued)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>End</td>
<td>End position of the SNP.</td>
</tr>
<tr>
<td>Score</td>
<td>Calculated p-value of the SNP.</td>
</tr>
<tr>
<td>Strand</td>
<td>Not used.</td>
</tr>
<tr>
<td>Phase</td>
<td>Not used.</td>
</tr>
<tr>
<td>Genotype</td>
<td>Genotype in the form of IUB codes for bases observed in all the reads. The base of the reference sequence at the current position.</td>
</tr>
<tr>
<td>Coverage</td>
<td>The number of the reads that cover the current position.</td>
</tr>
<tr>
<td>refAlleleCounts</td>
<td>The number of reads of the reference allele at the current position.</td>
</tr>
<tr>
<td>refAlleleStarts</td>
<td>The number of different start positions of reads having the reference allele at the current position.</td>
</tr>
<tr>
<td>refAlleleMeanQV</td>
<td>The mean of quality values of all reference allele reads at the current position.</td>
</tr>
<tr>
<td>novelAlleleCounts</td>
<td>The number of reads of the most abundant non-reference allele at the current position.</td>
</tr>
<tr>
<td>novelAlleleStarts</td>
<td>The number of different start positions of reads having the most abundant non-reference allele at the current position.</td>
</tr>
<tr>
<td>novelAlleleMeanQV</td>
<td>The mean of quality values of all novel allele reads.</td>
</tr>
<tr>
<td>mostAbundantAlleleDiColor2</td>
<td>The most abundant allele in the reads (not necessarily the reference allele) in dicolor encoding (for example, 00, 01, ... 32, 33), of 16 possible dicolors.</td>
</tr>
<tr>
<td>secondAbundantAlleleDiColor3</td>
<td>The second most abundant allele in the reads.</td>
</tr>
<tr>
<td>Het</td>
<td>Heterozygosity flag. Allowed values: 0, 1</td>
</tr>
</tbody>
</table>

- **0**: Homozygous SNP
- **1**: Heterozygous SNP

The following is a truncated example of diBayes tab-delimited output file content:

```
Segid Source Type Start End Score Strand Phase genotype reference coverage refAlleleCounts refAlleleStarts refAlleleMeanQV novelAlleleCounts novelAlleleStarts novelAlleleMeanQV mostAbundantAlleleDiColor2 secondAbundantAlleleDiColor3 het
chr1 SOLiD_diBayes SNP 10007 10007 0.806886 . . A G 3 1 1 16 2 2 25 21 21 0
chr1 SOLiD_diBayes SNP 224472 224472 0.879526 . . C T 3 1 1 26 2 2 24 10 10 0
chr1 SOLiD_diBayes SNP 553488 553488 0.0625 . . C T 3 0 0 0 2 2 25 20 20 0
chr1 SOLiD_diBayes SNP 556955 556955 0.095335 . . Y T 10 7 6 14 2 2 26 13 31 1
chr1 SOLiD_diBayes SNP 558326 558326 0.093729 . . R A 8 6 6 26 2 2 22 03 21 1
chr1 SOLiD_diBayes SNP 558554 558554 0.0761 . . Y T 14 11 9 20 3 3 22 03 21 1
```
**diBayes annotated tab-delimited output file**

This output file is the same as the diBayes tab-delimited output file (see “diBayes tab-delimited output file” on page 315), with four additional columns added. As with the diBayes tab-delimited output file, the first eight columns of this output file are the same as the input file. The ninth column in the input file is replaced by individual columns for each attribute, as shown in Table 87. This output file is generated only on SNPs module runs.

This is a tab-separated text file. The file is named according to this filename pattern:

```
<annotation.output.file.name.stem>_annotated.tab
```

where `<annotation.output.file.name.stem>` is a common prefix generated by the annotations module and used with all output files from the same run.

**Table 87 Additional columns in the diBayes annotated tab-delimited output file**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>geneID</td>
<td>The gene id as provided by the GTF file.</td>
</tr>
<tr>
<td>exonID</td>
<td>The exon id formed by concatenating the GeneID and the exon index number in the list of exons of the gene sorted in transcription order. Transcription order ties are broken listing by the shortest exon first (&lt;GeneID&gt;-&lt;exon #&gt;).</td>
</tr>
<tr>
<td>rsID</td>
<td>The dbSNP id of the SNP.</td>
</tr>
<tr>
<td>functionCode</td>
<td>The dbSNP functional code.</td>
</tr>
</tbody>
</table>

The following is a truncated example of diBayes annotated tab-delimited output file content:

```
Seqid Source Type Start End Score Strand Phase genotype reference coverage refAlleleCounts refAlleleStarts refAlleleMeanQV novelAlleleCounts novelAlleleStarts novelAlleleMeanQV mostAbundantAlleleDiColor2 secondAbundantAlleleDiColor3 het geneID exonID rsID functionCode
chr1 SOLiD_diBayes SNP 10007 10007 0.806886 . . A G 3 1 1 16 2 2 25 21 21 0 WASH5P
chr1 SOLiD_diBayes SNP 224472 224472 0.879526 . . C T 3 1 1 26 2 2 24 10 10 0
chr1 SOLiD_diBayes SNP 553488 553488 0.0625 . . C T 3 0 0 0 2 2 25 20 20 0
chr1 SOLiD_diBayes SNP 556955 556955 0.095335 . . Y T 10 7 6 14 2 2 26 13 31 1 9326622
chr1 SOLiD_diBayes SNP 558326 558326 0.093729 . . R A 8 6 6 26 2 2 22 03 21 1 2153587
chr1 SOLiD_diBayes SNP 558554 558554 0.0761 . . Y T 14 11 9 20 3 3 22 03 21 1 8179256
chr1 SOLiD_diBayes SNP 658055 658055 0.0625 . . T C 3 0 0 0 2 2 22 20 20 0
```

**diBayes filtered annotated tab-delimited output file**

This file is the same as the diBayes annotated tab-delimited output file, except that it contains only those entries that fulfill all the conditions set by the your annotation filtering settings (described in “Annotation parameters” on page 297).

This is a tab-separated text file. The file is named according to this filename pattern:

```
<annotation.output.file.name.stem>_annotated_filtered.tab
```
where `<annotation.output.file.name.stem>` is a common prefix generated by the annotations module and used with all output files from the same run.

This output file is generated only on SNPs module runs.

The following is a truncated example of diBayes filtered annotated tab-delimited output file content:

```
Seqid Source Type Start End Score Strand Phase genotype reference coverage refAlleleCounts refAlleleStarts refAlleleMeanQV novelAlleleCounts novelAlleleStarts novelAlleleMeanQV mostAbundantAlleleDiColor2 secondAbundantAlleleDiColor3 het geneID exonID rsID functionCode
chr1 SOLiD_diBayes SNP 556955 556955 0.095335 . . Y T
10  7  6  14  2  2  26  13  31  1    9326622
chr1 SOLiD_diBayes SNP 558326 558326 0.093729 . . R A
  8  6  6  26  2  2  22  03  21  1  2153587
chr1 SOLiD_diBayes SNP 558554 558554 0.0761 . . Y T
14 11 9 20 3 3 22 03 21 1   8179256
chr1 SOLiD_diBayes SNP 708249 708249 0.003906 . . G A
  5  0  0  0  3  3  23  30  30  0  10900602
chr1 SOLiD_diBayes SNP 708418 708418 0.0625 . . C T
  2  0  0  0  2  2  15  13  13  0  10751453
chr1 SOLiD_diBayes SNP 710103 710103 0.003906 . . C T
  3  0  0  0  3  3  17  01  01  0  3121393
chr1 SOLiD_diBayes SNP 713754 713754 0.003906 . . C G
  4  0  0  0  3  3  21  33  33  0  2977670
```

### About annotations and LifeScope™ Software modules

This section describes which annotation functionality is available with the various LifeScope™ Software modules.

**Table 88** shows which LifeScope™ Software modules support the annotation attributes.

<table>
<thead>
<tr>
<th>Label</th>
<th>SNPs</th>
<th>CNV</th>
<th>Small indel</th>
<th>Large indel</th>
</tr>
</thead>
<tbody>
<tr>
<td>geneID, exonID</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>rsID - SNPs</td>
<td>Yes</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>rsID - Indels</td>
<td>—</td>
<td>—</td>
<td>Yes</td>
<td>—</td>
</tr>
<tr>
<td>functionCode</td>
<td>Yes</td>
<td>—</td>
<td>Yes</td>
<td>—</td>
</tr>
</tbody>
</table>

**Table 89** lists which LifeScope™ Software modules support annotation filtering.

<table>
<thead>
<tr>
<th>Label</th>
<th>SNPs</th>
<th>CNV</th>
<th>Small indel</th>
<th>Large indel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only in exons</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Only in genes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Table 90 lists which LifeScope™ Software modules support for the various annotation statistics. Heterozygous and homozygous statistics are included for applicable variants.

Table 90  Support for annotation statistics

<table>
<thead>
<tr>
<th>Label</th>
<th>SNPs</th>
<th>CNV</th>
<th>Small indel</th>
<th>Large indel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only not in dbSNP</td>
<td>Yes</td>
<td>—</td>
<td>Yes</td>
<td>—</td>
</tr>
<tr>
<td>Only in dbSNP</td>
<td>Yes</td>
<td>—</td>
<td>Yes</td>
<td>—</td>
</tr>
</tbody>
</table>

FAQ - Annotations

1

How are errors with input files handled?

If an entry in the GTF file is malformed then the entry would be skipped. If there is a problem with the file then the entire file would be skipped. If no file is provided or is skipped then the GTF annotation is turned off. Similarly, if one of the dbSNP files is missing then the dbSNP annotation is turned off. The *.stats files may still show zeros in the dbSNP fields if the dbSNP annotation is turned on.

2

Can I add annotations to my GFF file from an earlier version of the software?

Yes. Run the annotations as a standalone module. Set the parameter to point to your existing GFF file.
3

Why is the function code missing locus information in hg19 while the information is in hg18? Can I do anything to get the locus information?

While doing annotation with hg19 we use the VCF file, which does not have locus information yet. You can use the dbSNP b132 BCP files to get the locus information, but please note that the indels are not completely annotated in that file. Also, using the BCP files requires more memory than the VCF file.

4

What is the definition of dbSNP concordance?

The dbSNP concordance reported is positional concordance. The dbSNP concordance measures the percentage of SNP calls that are at the same position as a variant in dbSNP. Exact comparison of genotypes is not performed, since typically genotypes for the sample of interest are not available in dbSNP. There is no comparison of observed alleles with the alleles reported at this position in dbSNP.

By default, only variants of the same type are counted:
- SNPs are annotated against SNPs in dbSNP.
- Indels are annotated against InDels in dbSNP.
PART V
Targeted Resequencing and Low Frequency Variants
Run Targeted Resequencing Analyses

This chapter covers:
- Overview ......................................................... 323
- Example of running targeted resequencing analyses ............... 324
- Targeted resequencing read selection algorithm ....................... 325
- Run targeted resequencing modules .................................. 326

Overview

Targeted resequencing is a set of analyses designed for target-enriched sequencing data. Given a set of coordinates representing a region of interest in a genome, a library may be enriched for reads within those regions prior to sequencing.

The workflow for analysis of target-enriched data is nearly identical to a standard resequencing workflow (see Chapter 9, “Run a Resequencing Mapping Analysis” on page 129). In fact, adequate results can be obtained using the standard workflow. The targeted resequencing application consists of many of the same modules as standard resequencing analysis. However, two additional modules are enabled by default for targeted resequencing analysis, and several modules have specific parameters set that are optimized for analysis of targeted resequencing data.

The SAET module is enabled by default for enriched runs, as the smaller target size makes spectral error correction viable, even if it may not be practical for unenriched sequencing runs on a large reference sequence.

Targeted resequencing mapping produces a filtered BAM file for variant calling. The variant calling modules diBayes and small indel are run on the filtered data sets instead of the entire read population, and limit SNP and indel calls to the target region.

The input list of regions of interest (ROI) is provided by the user. Supported ROI input files include third-party files of baits, probes, or tiles. The input ROI files can be in BED format or in a text file as a list of “chr:start-end” values. The targeted resequencing mapping module sorts the input ROI file by chromosome and start position, and makes the resulting file available as output.

Figure 30 describes the targeted resequencing analyses.
The targeted resequencing analyses support data from these library types:

- Fragment
- Paired-End
- Mate-Pair
  - Mate-pair libraries are accepted, but pairing information is not considered for mate-pair libraries.
- Libraries generated with up to 96 barcodes
  - Individual sequencing and enrichment reports are provided for each barcoded library.

Targeted resequencing output files are compatible with third-party genome browsers, including the IGV browser.

The targeted resequencing mapping module provides analysis of sequencing runs that have been run through targeted enrichment.

Targeted resequencing support optional enrichment statistics, which provide a means to assess enrichment platform performance. For information about the enrichment module, see Chapter 18, “Run Enrichment Analyses” on page 335.

Example of running targeted resequencing analyses

The following standard workflows provide examples of running targeted resequencing:

- targeted.resequencing.frag
- targeted.resequencing.pe

The following are example shell commands using reads and references from the LifeScope™ Software repository.

```
lscope.sh shell -u username -w password
# cd to the projects repository
```
cd /projects
# make a project, and open it
mk ecoli
cd ecoli
# make an analysis, and open it
mk run1
cd run1
# define the analysis type
set workflow targeted.resequencing.frag
# define the input data
add xsq DH10B_ColorSpaceOnly_Unbarcoded_50_Frag.xsq
# specify the reference for the analysis
set reference hg19
# optionally change parameter defaults here
set analysis.regions.file /data/results/readsdata/dh10b.bed
# list the configuration of the analysis
ls
# start the run
run
# list progress information about the run
ls

The command to set the analysis.regions.file does not include the optional INI file argument. As written, the command sets the analysis.regions.file parameter in both the secondary and tertiary global.ini files. In this way the parameter is available to all analysis modules in the workflow. If the parameter is instead set in only one INI file, for example the enrichment.ini file, then the regions of interest file is available to the enrichment module, but not available to the SAET module and other modules in the workflow.

See for Chapter 7, “Run a Standard Workflow Analysis” on page 103 for more information on running standard workflows. Chapter 6, “Run a Command Shell Analysis” on page 75 for information on shell commands and syntax.

Targeted resequencing read selection algorithm

Reads are classified as “on-target” by a set of user-definable criteria. First, all reads are filtered for a minimum mapping quality. Reads which align to multiple places in the genome are assigned a lower mapping quality. Reads with very low mapping qualities may be false positive hits arising from repetitive sequence in the reference genome.

The default target definition criteria used in targeted resequencing mapping are:

- Reads must have a minimum mapping quality of 8. Recommended values are between 0 (in which case there may be a population of repetitive reads which inflate enrichment statistics and may contribute to false positives in variant calling), and 20 (which ensures only high quality mapping reads are used, but results in a loss of sensitivity in regions with low mappability).
- Alignments may be marked as duplicate by the alignment algorithm. These reads are typically excluded from variant calling, but are counted for purposes of assessing enrichment.
• Only primary alignments are considered. Alignments marked as secondary (alternative placements) are excluded. Note that the mapping quality threshold may excluded reads with multiple secondary hits.

• Target regions are considered as provided. Reads aligning adjacent to targets may be included in the analysis by specifying the number of bases flanking the target regions that each target should be extended by.

• Mapping to target region is defined as having at least one base overlapping the target region.

• For paired-end runs, by default the reverse read must align 100%, as these are often shorter and less accurate. The default 100% alignment requires that all bases align to coordinates that are fully contained by the target region. For this test, trimmed bases are not considered. The overlap percentage required is controlled by the `enrichment.minimum.target.overlap.reverse` parameter.

Both alignment thresholds are user-adjustable, down to \( 1 \times 10^{-4} \) (meaning at least one base, for reads up to 1000bp) or up to 1.0, meaning the read and target must completely overlap.

While only one list of target regions can be provided by the user, multiple target region sets may be analyzed by re-running the enrichment analysis, and specifying a different output folder or prefix for each target list. Similarly, to assess near-target alignments, reanalyze while specifying a larger number of bases by which to extend the target.

The targeted resequencing module by default runs the SAET utility prior to mapping.

Run targeted resequencing modules

See Chapter 18, “Run Enrichment Analyses” on page 335 for information about the enrichment module.

The instructions for other targeted resequencing modules are the same as for the resequencing tools, except that the targeted modules require an additional input file, the regions of interest (ROI) file. You must provide the ROI file, in any of the following supported formats:

• a bait, probe, tile, or other file in BED format
• a text file as a list of “chr:start-end” values

To run resequencing pairing, find SNPs, and find small indels analyses, complete the instructions in the following chapters and also specify your the regions of interest file:

• Chapter 9, “FAQ – Pairing” on page 176
• Chapter 10, “Run a SNPs Analysis” on page 181
• Chapter 13, “Run a Small Indels Analysis” on page 235

The differences when these analyses are run for targeted resequencing, compared to standard resequencing, are:

• The find SNPs and find small indels analyses are both run with an input file containing only on-target reads. Additionally, find SNPs takes the processed target list as input, and restricts SNP calls to events occurring within the target regions.
• SNPs analysis is also passed the `dibayes.het.skip.high.coverage=0` parameter, to avoid excluding regions of very high coverage, which are common in enriched regions.
The SAET module is enabled by default.

**Note:** If you turn off SAET, comment out the `analysis.input.readset.file` line, if it appears in your mapping INI file:

```plaintext
analysis.input.readset.file = ${analysis.output.dir}/saet/*.rrs
```
Chapter 16  Run Targeted Resequencing Analyses

Run targeted resequencing modules
Run Low Frequency Variant Analyses

This chapter covers:

- Overview .......................................................... 329
- Examples of running low frequency variant analyses .................. 330
- Low frequency variant read selection algorithm ...................... 331
- SNPs optimizations .............................................. 332
- Run low frequency variant modules ................................ 334

Overview

The low frequency variant detection workflow uses the diBayes (SNPs) algorithm (and highly optimized parameters) to reveal SNPs present in a targeted resequencing experiment. The detection limit can range from 1%-5% based on experimental conditions, acceptable sensitivity and specificity, and sequencing coverage. This workflow is ideal for detecting low frequency variants in cancer-related research projects.

The low frequency variant workflows use unique defaults for SNPs filters and also uses these new SNPs filters:

- The color QV of the less common allele
- The color QV difference between the most common allele and the less common allele
- The unique start positions of the less common allele
- The requirement that the less common allele be present on both strands

The workflow for analysis of target-enriched data is nearly identical to a standard targeted resequencing workflow. The low frequency variant application consists of many of the same modules as standard resequencing analysis. However, several modules have specific of target-enriched data.

Low frequency variant workflow uses the enrichment module to produce a filtered BAM file for variant calling. The variant calling module diBayes is run on the filtered data sets instead of the entire read population, and limits SNP calls to the target region. SNPs module parameters are optimized for low frequency variant detection.

The input list of regions of interest (ROI) is provided by the user. Supported ROI input files include third-party files of baits, probes, or tiles. The input ROI files can be in BED format or in a text file as a list of “chr:start-end” values. The mapping module sorts the input ROI file by chromosome and start position, and makes the resulting file available as output.

The low frequency variant analyses support data from these library types:
• Fragment
• Paired-End
• Libraries generated with up to 96 barcodes
  Individual sequencing and enrichment reports are provided for each barcoded library.

Low frequency variant output files are compatible with third-party genome browsers, including the IGV browser.

Low frequency variant workflows optionally produce enrichment statistics, which provide a means to assess enrichment platform performance. For information about the enrichment module, see Chapter 18, “Run Enrichment Analyses” on page 335.

Examples of running low frequency variant analyses

The following standard workflows provide examples of running low frequency variant analyses:
• low.frequency.variant.frag
• low.frequency.variant.pe

The following are example shell commands using reads and references from the LifeScope™ Software repositories.

lscope.sh shell -u username -w password
# cd to the projects repository
cd /projects
# make a project, and open it
mk liversample1
cd liversample1
# make an analysis, and open it
mk run1
cd run1
# define the analysis type
set workflow low.frequency.variant.frag
# define the input data
add xsq Sample_TarReseq_Unbarcoded_50_Frag.xsq
# specify the reference for the analysis
set reference hg19
# optionally change parameter defaults here
set analysis.regions.file /data/results/readsdata/dh10b.bed
# list the configuration of the analysis
ls
# start the run
run
# list progress information about the run
ls
The command to set the `analysis.regions.file` does not include the optional INI file argument. As written, the command sets the `analysis.regions.file` parameter in both the secondary and tertiary `global.ini` files. In this way the parameter is available to all analysis modules in the workflow. If the parameter is instead set in only one INI file, for example the `enrichment.ini` file, then the regions of interest file is available to the enrichment module, but not available to the SAET module and other modules in the workflow.

See for Chapter 7, “Run a Standard Workflow Analysis” on page 103 for more information on running standard workflows. Chapter 6, “Run a Command Shell Analysis” on page 75 for information on shell commands and syntax.

## Low frequency variant read selection algorithm

Reads are classified as “on-target” by a set of user-definable criteria. First, all reads are filtered for a minimum mapping quality. Reads which align to multiple places in the genome are assigned a lower mapping quality. Reads with very low mapping qualities may be false positive hits arising from repetitive sequence in the reference genome.

The default target definition criteria used in low frequency variant mapping are:

- Reads must have a minimum mapping quality of 8. Recommended values are between 0 (in which case there may be a population of repetitive reads which inflate enrichment statistics and may contribute to false positives in variant calling), and 20 (which ensures only high quality mapping reads are used, but results in a loss of sensitivity in regions with low mappability).
- Alignments may be marked as duplicate by the alignment algorithm. These reads are typically excluded from variant calling, but are counted for purposes of assessing enrichment.
- Only primary alignments are considered. Alignments marked as secondary (alternative placements) are excluded. Note that the mapping quality threshold may excluded reads with multiple secondary hits.
- Target regions are considered as provided. Reads aligning adjacent to targets may be included in the analysis by specifying the number of bases flanking the target regions that each target should be extended by.
- Mapping to target region is defined as having at least one base overlapping the target region.
- For paired-end runs, by default the reverse read must align 100%, as these are often shorter and less accurate. The default 100% alignment requires that all bases align to coordinates that are fully contained by the target region. For this test, trimmed bases are not considered. The overlap percentage required is controlled by the `enrichment.minimum.target.overlap.reverse` parameter.

Both alignment thresholds are user-adjustable, down to $1 \times 10^{-4}$ (meaning at least one base, for reads up to 1000bp) or up to 1.0, meaning the read and target must completely overlap.

While only one list of target regions can be provided by the user, multiple target region sets may be analyzed by re-running the enrichment analysis, and specifying a different output folder or prefix for each target list. Similarly, to assess near-target alignments, reanalyze while specifying a larger number of bases by which to extend the target.
SNPs optimizations

SNP Finding parameters are optimized for Low Frequency Variant analysis. Parameters specific to LFV workflows are used, and parameters shared with other analyses have different default values for LFV workflows.

The next sections describes optimizations for Low Frequency Variant analysis.

Set the minimum allele ratio

Use the parameter dibayes.het.min.allele.ratio to set the allele ratio to a desired allele frequency.

In most experiments, setting the minimum allele ratio to be one-fifth of the lowest desired variant frequency ensures high sensitivity. For example, if you want to detect >90% of variants down to 10% frequency, set the minimum allele ratio to 2%. (This setting, of course, also generates variants with lower frequency, for example about half of variants at frequency in the sample of 2% are detected.) If you want to detect all variants down to 1% frequency, set the minimum allele ratio to 0.2%. If your coverage is not extremely high, or to increase specificity with some reduction in sensitivity, the minimum allele ratio may be set to one-third or one-half of the desired variant frequency. For example, if you are looking for variants at 1% frequency, you can set the minimum allele ratio to 0.5% or 0.33%.

Adjust sensitivity and specificity

To increase the sensitivity (detect more low frequency variants), or if the coverage is not extremely high, change the following parameters:

• *Reduce* the minimum number of unique start points of the less common allele (for example, to 3 or 4), with dibayes.het.min.lca.start.pos.
• *Reduce* the threshold for the difference of color QVs of the most common and less common alleles (for example, to 4), with dibayes.het.max.diff.color.qv.

Other parameters that can be changed to increase sensitivity include:

• *Remove* the requirement to observe the less common allele on both strands, with dibayes.het.lca.both.strands.
• *Reduce* the minimum average color QV of the less common allele (for example, to 20), with dibayes.het.min.lca.color.qv.
• *Reduce* the minimum map QV (for example, to 8), with dibayes.reads.min.mapping.qv.

To increase the specificity (call fewer false positives), parameters that can be changed include:

• *Increase* the minimum allele ratio, with dibayes.het.min.allele.ratio.
• *Increase* the minimum number of unique start points of the less common allele, with dibayes.het.min.lca.start.pos.
• *Include* the requirement to observe the less common allele on both strands, with dibayes.het.lca.both.strands.
• *Increase* the minimum average color QV of the less common allele, with dibayes.het.min.lca.color.qv.
**Low frequency variant detection SNPs parameter table**

The following table lists SNP Finding parameters and their default values in low frequency variant detection workflows.

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>General parameters</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Call stringency</td>
<td>low_frequency</td>
<td>The default for Targeted Resequencing is medium. Always use low_frequency with LFV workflows.</td>
</tr>
<tr>
<td>Skip high coverage positions (Het)</td>
<td>0</td>
<td>—</td>
</tr>
<tr>
<td>Minimum mapping quality value</td>
<td>16</td>
<td>The default for Targeted Resequencing is 8.</td>
</tr>
<tr>
<td>Detect adjacent SNPs</td>
<td>0</td>
<td>—</td>
</tr>
<tr>
<td>Polymorphism rate</td>
<td>0.001</td>
<td>—</td>
</tr>
<tr>
<td><strong>Low frequency variant filters</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minimum average color qv of the less common allele</td>
<td>30</td>
<td>The default for Targeted Resequencing is 0.</td>
</tr>
<tr>
<td>Minimum unique start position of the less common allele</td>
<td>7</td>
<td>The default for Targeted Resequencing is 0.</td>
</tr>
<tr>
<td>The less common allele on both strands</td>
<td>1</td>
<td>The default for Targeted Resequencing is 0.</td>
</tr>
<tr>
<td>Maximum difference of color qvs of the most common and less common alleles</td>
<td>10</td>
<td>The default for Targeted Resequencing is 0.</td>
</tr>
<tr>
<td><strong>Read filters</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Include reads with unmapped mate</td>
<td>True (Paired)</td>
<td>False (Frag)</td>
</tr>
<tr>
<td>Exclude reads with indels</td>
<td>True</td>
<td>—</td>
</tr>
<tr>
<td>Require only uniquely mapped reads</td>
<td>0</td>
<td>—</td>
</tr>
<tr>
<td>Ignore reads with a higher mismatch count to alignment length ratio</td>
<td>1.0</td>
<td>—</td>
</tr>
<tr>
<td>Ignore reads with a lower alignment length to read length ratio</td>
<td>0.9</td>
<td>The default for Targeted Resequencing is 1.0.</td>
</tr>
<tr>
<td><strong>General position filters</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Require alleles to be present in both strands</td>
<td>False</td>
<td>—</td>
</tr>
<tr>
<td>Minimum base quality value for a position</td>
<td>28</td>
<td>—</td>
</tr>
<tr>
<td>Minimum base quality value of the non-reference allele of a position</td>
<td>28</td>
<td>—</td>
</tr>
<tr>
<td><strong>Heterozygous position filters</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minimum allele ratio (Het)</td>
<td>0.01</td>
<td>The default for Targeted Resequencing is 0.15.</td>
</tr>
<tr>
<td>Minimum coverage (Het)</td>
<td>2</td>
<td>—</td>
</tr>
<tr>
<td>Minimum unique start position (Het)</td>
<td>2</td>
<td>—</td>
</tr>
<tr>
<td>Minimum non-reference color QV (Het)</td>
<td>7</td>
<td>—</td>
</tr>
<tr>
<td>Minimum non-reference base QV (Het)</td>
<td>28</td>
<td>—</td>
</tr>
<tr>
<td>Minimum ratio of valid reads (Het)</td>
<td>0.65</td>
<td>—</td>
</tr>
</tbody>
</table>
Table 91  SNPs parameter changes for low frequency variant detection workflows

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum valid tricolor counts [Het]</td>
<td>2</td>
<td>—</td>
</tr>
<tr>
<td><strong>Homozygous position filters</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minimum coverage [Hom]</td>
<td>1</td>
<td>—</td>
</tr>
<tr>
<td>Minimum count of the non-reference allele [Hom]</td>
<td>2</td>
<td>—</td>
</tr>
<tr>
<td>Minimum average non-reference base QV [hom]</td>
<td>28</td>
<td>—</td>
</tr>
<tr>
<td>Minimum average non-reference color QV [hom]</td>
<td>7</td>
<td>—</td>
</tr>
<tr>
<td>Minimum unique start position of the non-reference allele [Hom]</td>
<td>2</td>
<td>—</td>
</tr>
</tbody>
</table>

**Run low frequency variant modules**

See Chapter 18, “Run Enrichment Analyses” on page 335 for information about the enrichment module.

The instructions for other low frequency variant modules are the same as for the resequencing tools, except that the targeted modules require an additional input file, the regions of interest (ROI) file. You must provide the ROI file, in any of the following supported formats:

- a bait, probe, tile, or other file in BED format
- a text file as a list of “chr:start-end” values

To run resequencing pairing and find SNPs, complete the instructions in the following chapters and also specify your the regions of interest file:

- Chapter 9, “FAQ – Pairing” on page 176
- Chapter 10, “Run a SNPs Analysis” on page 181

The differences when these analyses are run for low frequency variants, compared to standard resequencing, are:

- The low frequency variant workflows use these SNPs parameters (which are not used in other SNPs applications):
  - dibayes.het.lca.both.strands
  - dibayes.het.min.lca.start.pos
  - dibayes.het.min.lca.color.qv
  - dibayes.het.max.diff.color.qv

  See Table 38 on page 185 for a description of these parameters.

- The find SNPs analyses are run with an input file containing only on-target reads. Additionally, find SNPs takes the processed target list as input, and restricts SNP calls to events occurring within the target regions.

- SNPs analysis is also passed the dibayes.het.skip.high.coverage=0 parameter, to avoid excluding regions of very high coverage, which are common in enriched regions.
Run Enrichment Analyses

This chapter covers:

- Overview ................................................................. 335
- Examples of running the enrichment module ................. 336
- Enrichment parameters ............................................. 337
- Enrichment statistics input files ................................... 339
- Enrichment statistics output files ................................. 340

Overview

Enrichment

The targeted resequencing and low frequency variant workflows use the enrichment module to produce target-enriched sequencing data from the input BAM files. Given a set of coordinates representing a region of interest in a genome, a library may be enriched for reads within those regions prior to tertiary analysis. The variant calling modules are run on the filtered data sets instead of the entire read population, and limit variant calls to the target region.

Figure 31 shows the use of enrichment analysis in the targeted resequencing workflow. In low frequency variant workflows, the use of the enrichment module is similar.

Figure 31  The role of enrichment in targeted resequencing analyses
The input list of regions of interest (ROI) is provided by the user. Supported ROI input files include third-party files of baits, probes, or tiles. The input ROI files can be in BED format or in a text file as a list of “chr:start-end” values. The mapping module sorts the input ROI file by chromosome and start position, and makes the resulting file available as output.

The tertiary analysis modules that support enriched input are the following:

- SNPs, by both targeted resequencing and low frequency variant workflows
- Small indels, by targeted resequencing workflows

### Enrichment statistics

Targeted resequencing and low frequency variant workflows support optional enrichment statistics, which provide a means to assess enrichment platform performance by:

- Looking at variations in coverage, both across all targets and on a per-target basis.
- Addressing the uniformity and completeness of coverage within the target region.
- Calculating the degree of enrichment.

### Examples of running the enrichment module

#### In a standard workflow

The following standard workflows provide examples of running the enrichment module:

- targeted.resequencing.frag
- targeted.resequencing.pe

The following are example shell commands using reads and references from the LifeScope™ Software repository.

```
lscope.sh shell -u username -w password
# cd to the projects repository
cd /projects
# make a project, and open it
mk ecoli
cd ecoli
# make an analysis, and open it
mk run1
cd run1
# define the analysis type
set workflow targeted.resequencing.frag
# define the input data
add xsq DH10B_ColorSpaceOnly_Unbarcoded_50_Frag.xsq
# specify the reference for the analysis
set reference hg19
# optionally change parameter defaults here
set analysis.regions.file /data/results/readsdata/dh10b.bed
# list the configuration of the analysis
ls
# start the run
run
# list progress information about the run
```
The command to set the `analysis.regions.file` does not include the optional INI file argument. As written, the command sets the `analysis.regions.file` parameter in both the secondary and tertiary global.ini files. In this way the parameter is available to all analysis modules in the workflow. If the parameter is instead set in only one INI file, for example the `enrichment.ini` file, then the regions of interest file is available to the enrichment module, but not available to the SAET module and other modules in the workflow.

See for Chapter 7, “Run a Standard Workflow Analysis” on page 103 for more information on running standard workflows. Chapter 6, “Run a Command Shell Analysis” on page 75 for information on shell commands and syntax.

### As a demo analysis

The optional examples download includes an example of how to run the enrichment module by itself, as a single analysis that is not part of a standard workflow. The demo example is not recommended for general use.

See Appendix H, “Demo Analyses” on page 567 for information on running the enrichment module as a standalone analysis.

### Enrichment parameters

Table 92 describes the parameters used with the enrichment statistics module.

In order to change a parameter value in your analysis, use the `set param shell` command to replace the line `# optionally change parameter defaults here` in the example commands in “In a standard workflow”. For instance, to turn off the summary report, use this shell command:

```
set enrichment.summary.report 0  tertiary/enrichment.ini
```

**Note:** Targeted resequencing analysis requires the regions of interest file, but the user interfaces do not enforce this requirement. Be sure you specify a regions of interest file when using a targeted resequencing workflow. An example command is:

```
set analysis.regions.file /data/results/readsdata/dh10b.bed.
```

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Required parameter</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>analysis.regions.file</td>
<td>—</td>
<td>The full path to your regions of interest file. You must supply the regions of interest file and you must explicitly set this parameter. See “Examples of running the enrichment module” on page 336 for an example command. Set this parameter in the global INI files, not in the analysis INI files.</td>
</tr>
<tr>
<td><strong>Optional module parameters</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| enrichment.run    | 1             | Whether or not to run the enrichment module. Allowed values:  
  * 0: Do not run the enrichment module.  
  * 1: Run the enrichment module during this analysis.  

The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use.
### Enrichment parameters

#### Table 92 Enrichment statistics parameters (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrichment.extend.bases</td>
<td>0</td>
<td>The number of bases on either side of the target region in which alignments may be captured. The target sequence is extended by this number of bp on either side. Used by both target capture and target coverage statistics. Example: enrichment.extend.bases=0 Allowed values: Integers &gt;= 0.</td>
</tr>
<tr>
<td>enrichment.minimum.mapping.score</td>
<td>8</td>
<td>Minimum mapping quality value (MAPQ) allowed for aligned reads. Reads below this threshold are not used. Allowed values: Integers &gt;= 0–80.</td>
</tr>
<tr>
<td>enrichment.minimum.target.overlap</td>
<td>0.0001</td>
<td>The fraction of an alignment that must be overlapped by a target in order to be classified as on target. “0.50” is interpreted as 50%, and implies that the midpoint of the read must fall within target region. Allowed values: 0.0001 &lt; X &lt;= 1.0 Example: enrichment.minimum.target.overlap=0.50</td>
</tr>
<tr>
<td>enrichment.minimum.target.overlap.reverse</td>
<td>1.0</td>
<td>The fraction of a Reverse Read (F5) alignment which must be overlapped by the target in order to be classified as on-target. To require 100% alignment, use 1.0 (the default). Example: enrichment.minimum.target.overlap.reverse=.9 Allowed values: 0.0001 &lt; X &lt;= 1.0</td>
</tr>
</tbody>
</table>

**Required output parameters**

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrichment.summary.report</td>
<td>true</td>
<td>Whether or not to create the summary statistics file. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• false, 0: Do not create the summary statistics.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• true, 1: Create the summary statistics file.</td>
</tr>
<tr>
<td>enrichment.target.coverage.stats</td>
<td>true</td>
<td>Whether or not to output per-target coverage statistics (min, max, mean) in tabular format. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• false, 0: Do not create per-target coverage statistics.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• true, 1: Create per-target coverage statistics.</td>
</tr>
<tr>
<td>enrichment.target.coverage.frequency</td>
<td>false</td>
<td>Whether or not to create the per-target coverage frequency histogram. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• false, 0: Do not create the per-target coverage frequency histogram.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• true, 1: Create the per-target coverage frequency histogram.</td>
</tr>
<tr>
<td>enrichment.target.coverage.bedgraph</td>
<td>false</td>
<td>Whether or not to create a BEDGRAPH format coverage file for on-target reads. Allowed values: true, false, 1, 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• false, 0: Do not create this file.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• true, 1: Create a BEDGRAPH format coverage file for on-target reads.</td>
</tr>
<tr>
<td>enrichment.genome.coverage.frequency</td>
<td>false</td>
<td>Whether or not to create a per-chromosome coverage frequency histogram. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• false, 0: Do not create this histogram.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• true, 1: Output a per-chromosome coverage frequency histogram.</td>
</tr>
</tbody>
</table>

**Resource parameters**
Enrichment statistics input files

This section describes the input files required for by the enrichment statistics module. Parameters mentioned in this section are described in Table 92 on page 337.

Table 93 lists the two input files for the enrichment statistics module.

Table 93  Enrichment Statistics required input files

<table>
<thead>
<tr>
<th>File</th>
<th>File format</th>
<th>Associated parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>target regions</td>
<td>BED, GFF v3, or text</td>
<td>analysis.regions.file</td>
<td>A list of target regions, either as regions of interest (ROI) or as target lists provided by vendors of enrichment platforms (such as baits, probes, or tiles files). Required, and supplied by the user.</td>
</tr>
<tr>
<td>aligned reads</td>
<td>BAM</td>
<td>analysis.input.bams</td>
<td>Alignments generated by the sequencing run after being matched to the entire reference genome. Managed by the framework, in the tertiary global.ini file.</td>
</tr>
</tbody>
</table>

The input target regions file

The target regions file may be either in BED format or a list of chr:start-end elements. An example of partial content for a target region file is shown in Figure 32.

<table>
<thead>
<tr>
<th>chr13</th>
<th>31787000</th>
<th>31788297</th>
</tr>
</thead>
<tbody>
<tr>
<td>chr13</td>
<td>31788404</td>
<td>31788957</td>
</tr>
<tr>
<td>chr13</td>
<td>31789616</td>
<td>31789695</td>
</tr>
<tr>
<td>chr13</td>
<td>31790390</td>
<td>31790601</td>
</tr>
<tr>
<td>chr13</td>
<td>31790852</td>
<td>31791495</td>
</tr>
<tr>
<td>chr13</td>
<td>31791644</td>
<td>31791981</td>
</tr>
</tbody>
</table>

Figure 32  Example content for an input target region file

The following information applies to the input target region file:

- Target region coordinates must be specified using the same reference used in mapping.
- Target regions should be sorted by chromosome/position. If not sorted, they will be sorted as the first step in the enrichment statistics module.
- After the first 3 required fields per line, any additional fields are ignored.
Chapter 18  Run Enrichment Analyses

Enrichment statistics output files

- The first field, “chrom”, must have identifiers which match the sequence names used in the BAM file. For example, for the hg18 reference:
  - Use “chr1”, not “1”.
  - Use “chrX”, not “23” or “chr23”.
- This file is in standard BED format, using zero-based indexing and a half-open interval (meaning the specified range is up to, but not including, the last position).

The input can optionally be a list of target regions in the “chr:start-end” format. This format uses one-based, closed intervals to represent genomic regions. An example of this format is:

```
chr1:1234-4321
```

**Note:** The input target region file and the chr:start-end list use different indexing and different interval formats.

The input aligned reads file

This file is a BAM-formatted file of aligned reads generated by the sequencing run after being matched to the entire reference genome.

The following information applies to the input aligned reads file:

- The BAM file must have a header listing sequence identifiers.
  The header is mandatory in the SAM specification for the binary BAM files.
- These sequence identifiers must be consistent with those used in the target list.
- The BAM file must be sorted by coordinate. The SO annotation in the header is not examined. All alignments within a chromosome must be contiguous. For sequential records, start coordinates within a chromosome must be equal or increasing.
- The BAM file may be from a fragment, paired-end, or mate-pair library.
  All reads are treated independently. No special handing is performed based on library type.
- Reads from paired libraries tend to have improved mapping scores based on unique paired hits, and are more likely to pass uniqueness filtering than unpaired reads.

Enrichment statistics output files

This section describes the enrichment statistics module’s output files. The output files are used in downstream analysis with LifeScope™ Software and are also compatible with external viewers. Genomic reads and annotations in GFFv3, BED, BEDGRAPH, and BAM format can be displayed with third-party graphical genome browsers. Targeted resequencing reports and tabular data can be viewed and graphed in third-party spreadsheet applications.
Table 94 lists the output files for the enrichment statistics module. These files’ purpose and content are described in the following pages. The enrichment statistics parameters mentioned in this section are described in Table 92 on page 337.

### Table 94  Enrichment Statistics output files

<table>
<thead>
<tr>
<th>File</th>
<th>File format</th>
<th>Optional</th>
<th>Controlling parameter (if optional)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output target file</td>
<td>BED</td>
<td>No</td>
<td>—</td>
</tr>
<tr>
<td>Output alignment file</td>
<td>BAM</td>
<td>No</td>
<td>—</td>
</tr>
<tr>
<td>Target coverage BEDGRAPH file</td>
<td>BEDGRAPH</td>
<td>Yes</td>
<td>enrichment.target.coverage.bedgraph</td>
</tr>
<tr>
<td>Genome coverage frequency file</td>
<td>Text</td>
<td>Yes</td>
<td>enrichment.genome.coverage.frequency</td>
</tr>
<tr>
<td>Target coverage frequency file</td>
<td>Text</td>
<td>Yes</td>
<td>enrichment.target.coverage.frequency</td>
</tr>
<tr>
<td>Target statistics report file</td>
<td>Text</td>
<td>Yes</td>
<td>enrichment.target.coverage.statistics</td>
</tr>
<tr>
<td>Enrichment statistics reports</td>
<td>Text</td>
<td>No</td>
<td>—</td>
</tr>
</tbody>
</table>

**The output target file**

This file is a BED file containing the sorted target list for use by later modules.

**The output alignment file**

This file is a filtered BAM file which includes only those alignments that overlap target regions. The file is a filtered version of the input file and retains the input file’s original characteristics. The BAI index file is also created.

The default file names for the BAM and index file are:

```bash
<input BAM file name>.ontarget.<extend_bases>.bam
<input BAM file name>.ontarget.<extend_bases>.bam.bai
```

where `<input BAM file name>` is the name of the input file (without path or file name extension), and `<extend_bases>` is optional and taken from the `enrichment.extend.bases` parameter, if that parameter is non-zero.

**The target coverage BEDGRAPH file**

This target coverage BEDGRAPH file is a text file in BEDGRAPH format and describes the depth of coverage within targets by on-target alignments. The target coverage file is generated, as determined by the `enrichment.target.coverage.bedgraph` parameter.

By default the target coverage file is created with the name

`${enrichment.output.prefix}_target_coverage.bedgraph`

Example content for a target coverage BEDGRAPH file is shown below:

```plaintext
track type=bedGraph name="On-Target Coverage" description="Depth of Coverage of On-Target Reads"
chr1 2974349 2974349 1
chr1 2974350 2974351 2
chr1 2974352 2974352 5
chr1 2974353 2974353 18
chr1 2974354 2974354 40
```
Table 95 describes the file content. These are tab-separated text fields.

Table 95 Description of the target coverage BEDGRAPH file fields

<table>
<thead>
<tr>
<th>Field name</th>
<th>Example content</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Track annotation</td>
<td>track type=bedGraph name=&quot;Coverage&quot;</td>
<td>Annotations used for UCSC browser display purposes.</td>
</tr>
<tr>
<td>Contig name</td>
<td>chr1</td>
<td>Contig name of the feature. This value comes from the sequence name [SN] in the BAM header. However, in order to be viewed in the UCSC browser, these values must be of the form chr1, chr2, etc.</td>
</tr>
<tr>
<td>Feature start</td>
<td>148</td>
<td>Start location of the feature. Zero-based, inclusive.</td>
</tr>
<tr>
<td>Feature end</td>
<td>150</td>
<td>End location of the feature. Zero-based, exclusive (half-open ranges).</td>
</tr>
<tr>
<td>Coverage</td>
<td>1</td>
<td>Depth of coverage over the given [start-end] range.</td>
</tr>
</tbody>
</table>

The genome coverage frequency file

The genome coverage frequency file is a text file which reports frequency of the coverage depth for each reference sequence contig listed in the BAM file header. The genome coverage frequency file is optionally generated, as determined by the `enrichment.genome.coverage.frequency` parameter.

By default the genome coverage frequency file is named

`${enrichment.output.prefix}_genome_coverage.frequency.tab`

Example content for a genome coverage frequency file is shown below:

```
track type=bedGraph name="On-Target Coverage" description="Depth of Coverage of On-Target Reads"
chr1 0 233314335 247249719 0.943638
chr1 1 10264341 247249719 0.0415141
chr1 2 2195560 247249719 0.00887993
chr1 3 581693 247249719 0.00235265
chr1 4 216736 247249719 0.000876587
chr1 5 114047 247249719 0.000461262
```

Table 96 describes the content of the genome coverage frequency file.

Table 96 Description of the genome coverage frequency file fields

<table>
<thead>
<tr>
<th>Column</th>
<th>Field name</th>
<th>Example content</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Contig name</td>
<td>chr1</td>
<td>Text</td>
<td>Contig name of the feature. This value comes from the sequence name [SN] in the BAM header.</td>
</tr>
<tr>
<td>2</td>
<td>Coverage</td>
<td>0</td>
<td>Text, Integer</td>
<td>Coverage depth value.</td>
</tr>
<tr>
<td>3</td>
<td>Coverage count</td>
<td>211562282</td>
<td>Text, Integer</td>
<td>Number of positions with the given depth of coverage.</td>
</tr>
<tr>
<td>4</td>
<td>Total contig size</td>
<td>247249719</td>
<td>Text, Integer</td>
<td>Total number of base pairs in the contig.</td>
</tr>
<tr>
<td>5</td>
<td>Frequency of depth in contig</td>
<td>0.855662</td>
<td>Text, Float</td>
<td>Fraction of contig covered at given depth. Count / Total = Frequency</td>
</tr>
</tbody>
</table>
The target coverage frequency file

The target coverage frequency file is a text file which reports the frequency of coverage depth for each target. The target coverage frequency file is optionally generated, as determined by the `enrichment.target.coverage.frequency` parameter.

By default the target coverage frequency file name ends in `_target_coverage.frequency.tab`.

Example content for a target coverage frequency file is shown below:

```
chr1_2974357_2974879    33      4       522     0.00766284
chr1_2974357_2974879    34      3       522     0.00574713
chr1_2974357_2974879    35      2       522     0.00383142
chr1_2974357_2974879    36      2       522     0.00383142
chr1_2974357_2974879    37      4       522     0.00766284
chr1_2974357_2974879    38      5       522     0.00957854
```

Table 97 describes the file content. The target coverage frequency fields are the same as for the genome coverage frequency file, except that location are given as target ranges.

Table 97 Description of the target coverage frequency file fields

<table>
<thead>
<tr>
<th>Column</th>
<th>Field name</th>
<th>Example content</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Target range</td>
<td>chr1_2974357_2974879</td>
<td>Text</td>
<td>One of the input target ranges. The contig name and the start and end positions from a single input target file line are concatenated.</td>
</tr>
<tr>
<td>2</td>
<td>Coverage</td>
<td>33</td>
<td>Text, Int</td>
<td>Coverage depth value.</td>
</tr>
<tr>
<td>3</td>
<td>Coverage count</td>
<td>4</td>
<td>Text, Int</td>
<td>Number of positions with the given depth of coverage.</td>
</tr>
<tr>
<td>4</td>
<td>Total target size</td>
<td>522</td>
<td>Text, Int</td>
<td>Total number of base pairs in the target.</td>
</tr>
<tr>
<td>5</td>
<td>Frequency of depth in target</td>
<td>0.00766284</td>
<td>Text, Float</td>
<td>Fraction of target covered at given depth. Count / Total = Frequency</td>
</tr>
</tbody>
</table>

The target statistics report file

The target statistics report file is a text file which reports per-target coverage and enrichment details in tab-delimited format, with one line per target. The target statistics report file is optionally generated, as determined by the `enrichment.target.coverage.statistics` parameter.

By default the target statistics report file name ends in `_target_statistics.txt`.

Example content for a target statistics report file is shown below:

```
Target Name     Target Size     Minimum Coverage        Maximum Coverage        Average Coverage        Total BP Covered at least 1X
chr1_2974357_2974879    522     33      2334    605.515 316079  522
chr1_2974634_2975198    564     0       892    210.183 118543  547
chr1_2975115_2975633    518     22      892     295.367 153000  518
chr1_2975421_2975989    568     0       571  110.783 62925   218
chr1_3092354_3092941    587     347    1875  991.16 581811  587
chr1_3150338_3150930    592     75      1301  546.51 323534  592
```
Chapter 18  Run Enrichment Analyses

Enrichment statistics output files

Table 98 describes the content of a target statistics report file.

Table 98  Description of the target statistics report file fields

<table>
<thead>
<tr>
<th>Column</th>
<th>Field name</th>
<th>Example content</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Target name (target range)</td>
<td>chr1_2974357_2974879</td>
<td>Text</td>
<td>One of the input target ranges. The contig name and the start and end positions from a single input target file line are concatenated with underscores.</td>
</tr>
<tr>
<td>2</td>
<td>Target Size</td>
<td>522</td>
<td>Text, Integer</td>
<td>Total size of this target range, in base pairs.</td>
</tr>
<tr>
<td>3</td>
<td>Min</td>
<td>33</td>
<td>Text, Integer</td>
<td>Minimum fold coverage of position in target.</td>
</tr>
<tr>
<td>4</td>
<td>Max</td>
<td>2334</td>
<td>Text, Integer</td>
<td>Maximum fold coverage of position in target.</td>
</tr>
<tr>
<td>5</td>
<td>Avg</td>
<td>605.515</td>
<td>Text, Number</td>
<td>Average fold coverage of all positions in target.</td>
</tr>
<tr>
<td>6</td>
<td>Total BP Coverage</td>
<td>316079</td>
<td>Text, Number</td>
<td>Enrichment fold of the target range.</td>
</tr>
<tr>
<td>7</td>
<td>BP Covered at least 1X</td>
<td>522</td>
<td>Text, Number</td>
<td>Number of bases with at least 1X coverage.</td>
</tr>
</tbody>
</table>

The enrichment statistics report

These reports provide data on the targeted regions in your analysis. The percent of coverage data is for the regions of interest, not for the whole genome reference. The end of the report summarizes the number of target bases covered at various depths. The file name pattern is *enrichment_report.stats. Example content is given below.

Run  Reads On  Percent On  Reads Off  Percent Off
 Off  Enrichment Fold
./outputs/enrichment/output.bam  69260675  88.6216%  8892585  11.3784%  1338.33

Reads In Targets:  69260675  88.6216%
Reads Off Targets:  8892585  11.3784%
Ratio of Percent on/off Target:  7.78859
Total BP:  2039803
Total Genome Size:  3080436051
Ratio of target to genome:  0.00066218
Enrichment fold relative to target size:  1338.33

<table>
<thead>
<tr>
<th># Targets Not Covered</th>
<th>Target Bases Not Covered</th>
<th>Percent of Target Bases Not Covered</th>
<th>Percent of Target Covered &gt;= 1X</th>
<th>Percent of Target Covered &gt;= 5X</th>
<th>Percent of Target Covered &gt;= 10X</th>
<th>Percent of Target Covered &gt;= 20X</th>
<th>Average Depth of Target Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12061</td>
<td>0.59%</td>
<td>99.41%</td>
<td>98.98%</td>
<td>98.63%</td>
<td>98.22%</td>
<td>2351.24</td>
</tr>
</tbody>
</table>

Number of target regions with no coverage: 1
Percent of target bp not covered: 0.59% (12061 bp)
Percent of target bp covered at >= 1X: 99.41%
Percent of target bp covered at >= 5X: 98.98%
Percent of target bp covered at >= 10X: 98.63%
Percent of target bp covered at >= 20X: 98.22%
Maximum Depth of Coverage within targets: 49453
Average Depth of Coverage within targets: 2351.24
Chapter 18  Run Enrichment Analyses

Enrichment statistics output files
PART VI

Whole Transcriptome Analyses
Run a Whole Transcriptome Mapping Analysis

This chapter contains:

- Overview ................................................................. 349
- Examples of running the WT mapping module ................ 351
- Input files ............................................................... 352
- Stages of mapping ...................................................... 355
- Quality alignments and unique alignments ..................... 364
- Single-read mapping parameters ................................ 365
- Paired-end parameters ............................................... 371
- Mapping output files ............................................... 376
- Mapping statistics ..................................................... 379
- FAQ – Whole transcriptome ........................................ 391
- FAQ – BAMStats ........................................................ 394

Overview

High-throughput sequencing of the transcriptome using the SOLiD® System enables genome-wide expression profiling with high sensitivity and a wider dynamic range than microarray technology. The whole transcriptome library preparation also preserves the strandedness of the RNA transcripts. Preserving the strandedness simplifies data analysis, allows determination of the directionality of transcription and gene orientation, and facilitates detection of opposing and overlapping transcripts.

You can use the 5500 Series Genetic Analysis Systems to sequence RNA prepared with RNA-Seq sample preparation kits. RNA sequencing produces high-depth, short-read sequencing data that can be used to measure RNA expression. Like microarray analysis, RNA-Seq measures expression intensity across many genomic features. Unlike microarray analysis, RNA-Seq can be used to identify novel transcriptome features in a sample.

Like other SOLiD® System applications, RNA-Seq produces short reads in the form of XSQ files. Whole Transcriptome Analysis (WTA) may be applied to these reads to produce alignment, coverage, count, and splice-finding results. WTA is a reference-based analytical method, which means a reference genome is required as input. Much WTA functionality also requires gene annotation associated with the reference genome sequence.
WTA analysis can be divided into secondary and tertiary analysis. Secondary analysis produces the aligned reads using sequences and quality values as input. Tertiary analyses produce biologically meaningful results (gene expression level, feature discovery, etc.) from aligned reads. This chapter describes how WTA is used to align reads to a reference genome, the methods used by WTA, and the configurable parameters of WTA.

The 5500 Series Genetic Analysis Systems produces RNA-Seq reads in both single-read and paired-end configurations. While there is overlap in the analytical techniques, the analysis of single-read and paired-end data is performed with two distinct pipelines. The single-read pipeline consists of alignment, counting, and coverage steps. The paired-end pipeline extends single-read analysis with an alternative alignment step and an additional junction finding step.

The aligned reads produced by WTA can be used with the following tertiary modules:

- **Count annotations** – Computes the number of reads that align within genomic features.
- **Coverage** – Computes the read coverage at each genomic position.
- **Splice finder** – Identifies splice junctions of various types, including fusion junctions from paired-end reads.

The 5500 Series Genetic Analysis Systems™ System produces RNA-Seq reads in both single-read and paired-end configurations. While there is overlap in the analytical techniques, the analysis of single-read and paired-end data is performed with two distinct pipelines. The single-read pipeline consists of alignment, counting, and coverage steps. The paired-end pipeline extends single-read analysis with an alternative alignment step and an additional splice finding step.

![Figure 33 WTA single-read and paired-end internal workflows](image-url)
Examples of running the WT mapping module

In a standard workflow

The following standard workflows provide examples of running the WT mapping module:

- whole.transcriptome.frag
- whole.transcriptome.pe

The following are example shell commands using reads and references from the LifeScope™ Software repository. This example turns off tertiary modules in the workflow.

# log into the shell
lscope.sh shell -u username -w password
# cd to the projects repository
cd /projects
# make a project, and open it
mk ecoli
cd ecoli
# make an analysis, and open it
mk run1
cd run1
# define the analysis type
set workflow genomic.resequencing.pe
# define the input
add xsq run0209a_50_PE.xsq
add xsq run0209b_50_PE.xsq
# specify the reference to be used
set reference hg19
# these commands turn off the tertiary modules
set coverage.run 0 tertiary/coverage.ini
set splice.finder.run 0 tertiary/splice.finder.ini
set wtcounts.run 0 tertiary/wtcounts.ini
# optionally change mapping parameter defaults after this line
# list the configuration of the analysis
ls
# start the run
run
# list progress information about the run
ls

In order to change a mapping parameter value in your analysis, use the set param shell command after the line # optionally change mapping parameter defaults after this line. For instance, to turn on the mapping.in.base parameter, use this shell command:

    set mapping.in.base 1 secondary/wt.pe.map.ini

This command shows how to turn on the mapping.in.base parameter for the whole.transcriptome.frag workflow:

    set mapping.in.base 1 secondary/wt.frag.map.ini

See Chapter 7, “Run a Standard Workflow Analysis” on page 103 for more information on running standard workflows.
As a demo analysis

The optional examples download includes an example of how to run the whole transcriptome analyses outside of a standard workflow. A standard workflow is recommended.

See Appendix H, “Demo Analyses” on page 567 for more information on running a demo analysis.

Input files

Reads input files

The WT mapping module accepts as input one or more input XSQ files. The input reads can have different read lengths but they must be of the same library type (fragment or paired-end). For example, paired-end read-sets of length 50x25 and 60x25 can be processed together, but a paired-end read-set cannot be processed with a fragment read-set.

RNA-Seq reads

The RNA-Seq reads used in WTA are different from the genomic reads. For RNA-Seq reads:

- Only transcribed sequences are measured by the system.
- Genome coverage is non-uniform due to variation in transcriptional intensity.
- A large subset of reads originate from uninteresting sequences such as ribosomal RNA (rRNA).
- A subset of the reads originates from splice junctions and do not align contiguously on the genome.

Reads input specification

Plan your analysis input carefully. The way you define your reads input affects the behavior of your analysis. The following factors control how your input data is analyzed:

- **Index (barcode) IDs** – Using an index ID restricts your input to the reads data of one or more indices.
- **Grouping of reads** – Each group of reads is analyses together as one specimen. The output data for a group is combined into one set of results files.
- **Multiple sample runs** – Unrelated reads can be processed together in one run of LifeScope™ Software, but analyses separately as separate input data.

See “Define input data” on page 90 for more information on designing how to add input data to your analysis.

Mapping one tag of paired data is not supported.

Legacy data

If the data you want to process with LifeScope™ Software is in CSFASTA and QUAL files, these files must be converted to the XSQ file format, through one of the following methods:

- The LifeScope™ Software UI handles converting these files to the XSQ format before mapping.
• If you are using the LifeScope™ Software command shell, you must first convert the CSFASTA and QUAL files to the XSQ format. See Appendix D, “XSQ Tools” on page 519 for information on the standalone XSQ converter.

### Reference input files

WTA uses several files defining reference information: filter-reference, genome-reference and annotation files. The filter-reference file is a FASTA-formatted file defining uninteresting reference sequences such as Ribosomal RNA, tRNA, or vector sequences. To increase the speed of analysis, reads that align to sequences in the filter reference are excluded from downstream alignment steps. Many of the entries in the filter reference are species-specific. The filter reference file must be appropriate for the species being studied.

The principal output of alignment analysis is a set of alignments between the reads and the genome-reference in BAM format. The genome-reference file is a FASTA-formatted file defining sequences in the reference genome.

The annotation file is a GTF-formatted file defining known genes, transcripts, and exons in the genome reference. See “Annotations input files”.

Junction-reference and exon-reference files are generated internally by the software. These files are entirely derived from the GTF and genome-reference files. A user is not required to provide these references. The junction-reference file is a FASTA-formatted file with an entry for every potential intragenic splice-junction defined in the annotation file. The entry contains the position of the junction and the associated flanking sequence. The exon-reference is a FASTA-formatted file with an entry for every exon defined in the annotation file.

### Annotations input files

While genomic resequencing relies only on alignment to a reference sequence, WTA also makes use of gene annotations. Gene annotations define the exons, genes, and transcripts used to improve alignment.

WTA modules use genome annotation files in GTF format. The following sites explain the GTF format:

```plaintext
genes.cse.wustl.edu/GTF2.html
genome.ucsc.edu/FAQ/FAQformat.html#format4
```

The GTF file must match the genome reference to ensure that the WTA modules work correctly.

**IMPORTANT!** Make sure the GTF file is for the same genome assembly as the FASTA file, and that matching sequence identifiers are used. Gene and transcript identifiers in the GTF file must be properly normalized. Identifier normalization is a known issue in GTF files from the UCSC Genome Browser, which is a popular source of GTF-formatted annotation. The UCSC GTF files always report the same value for gene and transcript IDs.

LifeScope™ Software includes a script `refgene2gff.sh` that produces a compliant GTF file from the UCSC Genome browser database.
Chapter 19  Run a Whole Transcriptome Mapping Analysis

Input files

UCSC genome annotations

Format UCSC Genome Browser Database annotations for WT analyses

The UCSC Genome Browser Database has genome annotations available for many assemblies at

`hgdownload.cse.ucsc.edu/goldenPath/`

The GTF-formatted annotations available for download are not properly normalized by gene ID. The required content is present for each assembly in the file export of the refGene database table `database/refGene/txt.gz`.

For example, annotation for human genome build 18 is available at:

`hgdownload.cse.ucsc.edu/goldenPath/hg18/database/refGene.txt.gz`

Note: The `refGene.txt.gz` file is not in GTF format. You must convert the annotation before using it in WTA.

Convert the `refGene.txt.gz` file

Run the script `bin/refgene2gff.sh` to convert the `refGene.txt.gz` file:

```
gunzip refGene.txt.gz
refgene2gff.sh -i refGene.txt -o refGene.gff
```

Genome annotations that are downloaded from the UCSC Genome Browser and converted by the annotation conversion script are preferable because they contain Human Genome Organization (HUGO)-style gene names. HUGO-style gene names allow interpretation when using a genome browser or reading reports.

The annotation conversion script works with the latest format of `refGene.txt` files. Assemblies, such as the rat genome, use an alternative format for the `refGene.txt` file. The `refgene2gff.sh` script does not convert alternative formats.

Ensembl GTF files

Format Ensembl GTF files for WTA pipelines

The Ensembl website `ensembl.org` has GTF-formatted genome annotations available for many popular assemblies. Unlike the GTF files directly downloadable from UCSC (see “UCSC genome annotations” on page 354), Ensembl GTF files are properly normalized by gene and transcript IDs. Visit the following site to download an Ensembl GTF file:

`http://www.ensembl.org/`

Ensembl GTF files use gene accession numbers instead of HUGO-style gene names. Ensembl GTF files also use unprefixed sequence identifiers, such as 1,2,3….X,Y,MT. The Ensembl GTF files are incompatible with genome reference FASTA files that have UCSC-style sequence IDs with the prefix “chr”, for example, chr1, chr2, chr3….chrX, chrY, chrM.
Reformat the Ensembl GTF file

To reformat the Ensembl GTF file to use UCSC-style gene IDs, run the `reformat_ensembl_gtf.pl` script, and redirect the output to a new file with .reformatted in the file name.

For Example:
```
reformat_ensembl_gtf.pl Homo_sapiens.GRCh37.63.gtf > Homo_sapiens.GRCh37.63.reformatted.gtf
```

**Stages of mapping**

A WT mapping run consists of the following phases:

- **Scatter** – The entire set of reads is divided into small non-overlapping subsets and defines one mapping job per subset.
- **Mapping** – A single computer processes a mapping job, which takes reads as input and produces aligned reads as output.
- **Gather** – The results of multiple mapping jobs are combined to produce final mapping result.

![WT mapping phases](image)

**Figure 34** WT mapping phases

The Mapping phase alone computes the alignment records that compose the results. This phase can be broken down into several steps:

- **Filter mapping** – Reads that align to the filter reference are counted and excluded from further analysis.
- **Genome mapping** – Reads are aligned to the genome reference.
- **Junction mapping** – Reads are aligned to the junction reference (F3 reads only).
- **Exon mapping** – Reads are aligned to the exon reference (F5 reads only).
- **Alignment merge** – Results from genome and junction or exon mapping are combined to produce a single set of alignments for each read type (F3 or F5).
• **Exon table rescue** – Supplements the set of alignments by conducting a more sensitive search for a alignments of a read (target) near the alignments found for the other member of the pair (anchor). These regions are selected using the gene structure of the genome. (Paired-end only).

• **Pairing** – Selection of most likely alignments based on alignment quality and the distance between alignments of each member of a pair. (Paired-end only).

• **BAM file generation** – The process producing results in BAM format. For color-space data, the base translation step is also included. Due to the scattering step, the BAM files at each compute node are a sub-set of the final BAM file and are referred to as baby BAMs.

BAM generation is composed of several substeps.
- **Ma2BAM, Pa2BAM** – A BAM file (lacking actual sequences or quality values) is written. Mapping quality values are computed (MAPQ).
- **Refcor** – Reference-assisted color-to-base translation. Sequences and quality values are added to the BAM records.

• **Sorting** – Sorting the BAM file by genome coordinate.

![Diagram](image)

*Figure 35 The phases of mapping WT fragment data*
**Scatter**

The scatter stage distributes reads to the available compute nodes, in order to gain the efficiency benefits of parallel computing.

The mapping module can accept multiple read-sets, which are allowed to have different read lengths. The input set of read-sets from multiple XSQ files is first split based on read length. Fragment read-sets are partitioned into separate categories based on these read length categories:

- $0 \leq \text{Length} < 25$
- $25 \leq \text{Length} < 35$
- $35 \leq \text{Length} < 50$
- $50 \leq \text{Length} < 75$
- $75 \leq \text{Length}$
- Trimmed reads, with variable read lengths

The reads from each category are scattered into multiple jobs based on the parameters `fragmap.minreads.per.node` and `fragmap.number.of.nodes`. These parameters are described in the section “Mapping performance” on page 367.
Single-read mapping

WT Single Read mapping produces a set of aligned reads. Stated briefly, the method performs the following steps (see Figure 35 on page 356).

- Remove reads that align to the filter reference.
- Map reads to genome and junctions.
- Merge results.
- Convert to BAM format.

Alignment finding uses a proprietary algorithm (see Chapter 9, “Run a Resequencing Mapping Analysis” on page 129). The filter mapping step identifies and counts the reads that align to the filter reference. The counts are written to the filter report. Reads that align to entries in the filter reference are omitted from further analysis and do not appear in the set of aligned reads produced by the module.

The genome mapping step finds alignments between the reads and the genome reference. Likewise the junction mapping step finds alignments with the junction reference. Alignments of a read to a genome and to junctions can produce multiple similar alignments at the same location. When multiple similar alignments occur, the alignment with the highest alignment score replaces all others. If there is a tie, the genomic alignment is used. The alignment score is calculated as follows:

\[
Score = len - nm \times 1(1+mp) - jp
\]

where:
- \( len \) = number of colors in the alignment
- \( nm \) = number of color mismatches
- \( mp \) = mapping mismatch penalty
- \( jp \) = penalty for alignment to a junction

The parameters to control mapping algorithm are in Table 101 on page 366. For details of the mapping algorithm, see “Mapping schemes” on page 370.

Filter mapping

The reads are mapped to the filter reference in the filter mapping step. Reads that align to the filter reference are called filtered reads. The result of this mapping step is used to populate the filter report. In the single-read pipeline, filtered reads are populated in a separate BAM file. Filtered reads are omitted from the BAM file. A filter reference for use with human reads is available with LifeScope™ Software. This reference contains:

- SOLiD® adapter sequences
- Human ribosomal RNAs (rRNAs)
- Human transfer RNAs (tRNAs)
- Other human sequences
- Single-base-repeat sequences, including Poly-A, Poly-T, and more

Complete the following steps to construct a filter reference for another species.

2. Append FASTA-formatted species-specific ribosomal, transfer RNA, and other sequences to the filter reference.
Junction mapping

Reads are mapped to the flanking sequence of junctions defined by the genome annotation in the junction mapping step. Junctions are inferred from the exon, gene, and transcript definitions in the genome annotation. Junction sequences are produced for all splices between pairs of exons within a gene. Junctions present in annotated transcripts are called known junctions. Junctions not present in annotated transcripts are called putative junctions.

Exon mapping

In the exon mapping step, paired-end reads are mapped to the set of exon sequences defined by the genome annotation. This step maps the shorter F5 reads in the paired-end pipeline. By default, more mismatches are allowed in exon mapping of F5 reads in the paired-end pipeline. Single-read mapping skips this stage.

Table 99 describes a human exon sequence database and a human junctions database.

Table 99  Comparison of alternative human references

<table>
<thead>
<tr>
<th>Reference</th>
<th>Number of references</th>
<th>Bp size</th>
<th>Mappability</th>
<th>Novelty</th>
<th>Junctions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>F3</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Genome</td>
<td>25</td>
<td>3 billion</td>
<td>OK</td>
<td>OK</td>
<td>—</td>
</tr>
<tr>
<td>Junctions</td>
<td>2,012,075</td>
<td>2,012,075,000</td>
<td>OK</td>
<td>—</td>
<td>OK</td>
</tr>
<tr>
<td>Refseq</td>
<td>32,699</td>
<td>99,027,499</td>
<td>OK</td>
<td>—</td>
<td>Partial</td>
</tr>
<tr>
<td><strong>F5</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Genome</td>
<td>25</td>
<td>3 billion</td>
<td>—</td>
<td>OK</td>
<td>—</td>
</tr>
<tr>
<td>Exons</td>
<td>216,884</td>
<td>99,026,749</td>
<td>OK</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Refseq</td>
<td>32,699</td>
<td>99,026,749</td>
<td>OK</td>
<td>—</td>
<td>Partial</td>
</tr>
</tbody>
</table>

In a paired-end alignment, the F3 and F5 paired-ends are initially processed through independent mapping paths. Similar to single-read-mapping, the F3 reads are mapped to filter, genome, and junction sequences. The F5 reads are also mapped to filter, genome, and exon sequences. Reads that map to filter sequences are discarded and do not proceed further in the pipeline.

The F3 genomic and junction mapping results are merged into a non-redundant set of alignments. The F5 genomic and exon mappings are merged into a non-redundant set of alignments. At this point in the analysis, the output consists of a set of independently processed F3 and F5 genomic alignments.

Rescue method

You can optionally use the rescue method to find additional alignments. Rescue is an alignment method that is applied to read pairs that have at least one alignment, but no pair of alignments occurring within an expected range (see Figure 37 on page 360). The expected range is set to 100,000 bases by default.
Chapter 19  Run a Whole Transcriptome Mapping Analysis

Stages of mapping

Figure 37  Annotation-aided rescue for WT

The next section refers to Figure 37.

Row A

For alignments that fall on an exon of a gene, and do not have a mate alignment above a certain score threshold, a special exon rescue is performed. Rescue region (rr) is defined as a rescue distance that is downstream of the alignment. The rescue distance is determined by user-defined thresholds. The rescue distance starts from the left-most position of the alignment for overlapping mates. The rescue region also includes certain position range in downstream exons of the same gene. The approach defined in the previous sentences helps rescue mates on different exons.

Row B

A rescue is still performed on downstream exons if a read is mapped on the intron of a given gene.

Row C

No special exon rescue is performed if an alignment falls on an intergenic region. Instead, regular rescue within the defined pairing distance is performed.

Alignments that lack a sibling read aligned nearby are called anchors. If anchors are found, the rescue tool conducts a more sensitive search for the sibling near the anchor, within a limited region of the genome. In the WT paired-end pipeline, the search is limited to anchors that occur within the introns or exons of annotated genes, with an allowance for a few overhanging bases.

Rescue is performed only within a set of expected rescue distances determined by a gene’s exon structure and the insert size distribution. Rescue distances are a function of the transcript rescue distance:

\[
\text{transcript rescue distance} = \text{mean insert size} + 3 \text{ standard deviations}
\]
This distance is longer than the majority (99.7%) of inserts. The formula above describes the rescue distance without taking into account the presence of introns. Because inserts are very likely to contain introns, a rescue distance is calculated for each potential splice configuration of the gene. For each configuration, the rescue distance is the transcript rescue distance + length of introns within the relevant transcribed sequence interval. Alternatively, rescue can be performed on all exons to improve sensitivity, but with a substantial increase in both false positives and run time. Rescue can be applied using F3 anchors only, F5 anchors only, or both.

Regardless of the rescue steps employed, the rescue results supplement the alignments detected in individual mapping steps. The end result of rescue is a set of F3 and F5 alignments in the same format as that produced by mapping.

**Pairing reads**

In the pairing step, pairs of reads are evaluated, assigned a mapping quality value, and written to a BAM file. The pairing range is set to 100,000 so that reads in adjacent exons are tagged as proper pairs. Unlike the pairing of genomic resequencing data, the mapping quality of read pairs is a function of genome annotation. Alignment pairs that do not occur within the same gene are penalized. For each pair, the alignment with the highest quality value is designated as the primary alignment. In the case of multiple highest quality, a single-pair alignment is selected randomly. Figure 38 shows an example of pairing range calculation with junction alignments.

![Figure 38](image)

**Figure 38** WT annotation-aided alignment Pairing Quality Value (PQV)

See Table 100 for details about rows A to F in Figure 38.

<table>
<thead>
<tr>
<th>Row(s)</th>
<th>Description</th>
<th>PVQ penalized?</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Mates fall on the same exon.</td>
<td>No</td>
</tr>
<tr>
<td>B</td>
<td>One mate falls on an exon and the other falls on an intron.</td>
<td>Yes</td>
</tr>
<tr>
<td>C and D</td>
<td>Mates fall on separate exons of the same gene.</td>
<td>No</td>
</tr>
<tr>
<td>E</td>
<td>Mates fall on exons of different genes.</td>
<td>Yes</td>
</tr>
<tr>
<td>F</td>
<td>Spliced alignment where one mate partially falls on a known gene and the other falls on the exon of the same gene.</td>
<td>No</td>
</tr>
</tbody>
</table>
Chapter 19 Run a Whole Transcriptome Mapping Analysis
Stages of mapping

BAM file generation

Ma2BAM, Pa2BAM

Ma2BAM and pa2BAM are format converters that change match files to a pro-BAM format. These pro-BAM files do not contain sequence or quality information but contain relevant BAM meta data. The pro-BAM files are sorted by incoming bead-id order. Relevant meta data from XSQ files is captured in pro-BAM files at this stage. The parameters bamgen.primary.output.filter.type, bamgen.mqv.threshold, and create.unmapped.bam.files control the functionality of the ma2BAM and pa2BAM utilities. The pro-BAM files are generated on the scratch folder.

Ma2BAM takes output of mapping (MA files) and converts them to pro-BAM format. The pa2BAM takes output of pairing (PA files) and converts them to pro-BAM format. During BAM generation, the mapping and pairing quality values are also computed.

For reads with multiple ungapped alignments, the read with the highest mapping or pairing quality value is chosen as the primary alignment for the read, and is reported to the BAM file. In cases where there are multiple alignments with the same quality value, the primary alignment is chosen at random from among the alignments with the same quality value. The details of calculating mapping and pairing quality values are described below.

Mapping and pairing quality values

This section describes mapping quality value (MQV) and pairing quality value (PQV) scores. Quality values are Phred-scaled quality scores. For any given alignment, an MQV is a measure of the confidence of a read aligning to the particular location, given all possible alignments for the read. A PQV represents the confidence in both alignments of a pair, and is a combined quality score for both reads. Quality values are within the range 0–100:

- 0 – The highest probability of error
- 100 – The lowest probability of error

The following factors increase confidence and increase a read’s quality value:

- Longer alignment
- Fewer possible alignments
- Fewer mismatches within the alignment

These factors reduce confidence and reduce a read’s quality value:

- Shorter alignment
- More possible alignments
- More mismatches within the alignment

The pairing algorithm reports multiple sets of possible alignments for any given pair of reads (F3/R3 tags for a mate-pair run and F3/F5-P2 tags for a paired-end run). The pairing quality algorithm uses a Bayesian approach to calculate the quality of a given alignment for a pair of reads and the alignment with the highest pairing quality value (PQV) is chosen as the primary alignment for the pair of reads. The PQVs represent the Phred-scaled quality score, and they are useful for downstream variant detection modules such as diBayes, small indels, large indels, and CNV.
In order to be consistent with the Phred quality score (-10*\log_{10}\text{[prob(error)]}) used widely in literature, the quality is computed as the negative log odds of misaligning the read (or pairs of reads, for PQV). The resulting quality values are normalized by the maximum possible value to ensure that the quality values are within the range 0–100.

Gapped alignments

The pairing algorithm searches for gapped alignments (indels) when one of the tags (F3/R3/F5-P2/F5-BC) maps to the reference genome and the other tag does not map to the genome within the insert-size range, or it maps within the insert-range but the alignment is short. If both an ungapped and a gapped alignment are found for a given read, then, due to the low prior probability of $10^{-4}$ assigned to the gapped alignments, the PQV for gapped alignments is zero.

![Figure 39 Example of a gapped alignment and a partial alignment](image)

In calculating the PQV for gapped alignments, the alternative hypothesis tested is the probability of finding the partial ungapped alignments. The read with the gapped alignment is treated as two partial reads on either side of the indel start point. The partial read with the greater length is used as the partial alignment length for the alternate hypothesis.

Refcor

Reference Assisted Color-to-Base Translation (also known as refcor) combines information from color calls, optional ECC extra primer round calls, and the reference sequence, to enhance the quality of base-space reads by detecting and fixing color errors that convert into multiple base-space errors.

Refcor accepts a set of pro-BAM files, which contain alignments information without base or color sequence, and generates a corresponding set of BAM files containing alignments with base-space and color-space calls (on the scratch folder). These BAM files are sorted by bead-ID and contain dummy base calls and QVs, but are otherwise compatible with the BAM specification. The refcor module produces bead-ID-sorted BAM files that have valid calls and QVs.

For each alignment, the module constructs the primer-transition graph with vertices corresponding to all possible k-mers (k=4 for 2BE+4BE, k=2 for 2BE) in each read position. Two adjacent vertices are connected with an edge if they overlap in k–1 bases. A score corresponding to the (1–Pe), where Pe is the probability of being erroneous, is assigned to each of the vertices. Scores are derived from quality values corresponding to enumerated calls. When making a base call in a position, the current algorithm considers the cumulative probability of all sequences that have called base in that
position. The evidence from the reference (base) is assigned to the vertex with \( k \)-mer ending in corresponding base only if the color transition between current and following reference bases matches the color call in a read. Such assignment reduces the reference over-correction in SNP positions. In order to reduce reference over-correction in positions with color errors, we assign to reference base a very low weight corresponding to QV of 8. The weight can be reduced to QV of 0, completely reducing the reference guiding in the color-to-base translation. This results in a smaller skew between reference to non-reference allele ratio. Increasing the weight results in false positive variant calls reduction. This process restores the phase shift caused by color errors or completely eliminates the errors.

The parameters \( \text{bamgen.refcor.addcs}, \text{refcor.reference.weight}, \) and \( \text{refcor.base.filter.qv} \) control the refcor program. The defaults provided are validated for a wide range of data.

- If the XSQ file contains base-space data only, the XSQ base call and quality value are transferred to BAM file. In this case, no color calls are present in BAM file irrespective of the value of \( \text{bamgen.refcor.addcs} \).
- If the XSQ file contains 2BE color-space data only, refcor uses aligned 2BE and reference to generate base translation for the aligned part of the read. The base sequence and quality value are generated using a proprietary algorithm. For reads with no alignment, bases are determined by naively translating color to base. For these unaligned reads, the QV for the base at position \( k \) is the minimum color QV for all positions \( \leq k \).
- If the XSQ file contains ECC data, then reference with 2BE and 4BE color are used to generate the new base sequence. The base sequence and quality value are generated using a proprietary algorithm. For reads with no alignment, the original base-space data from ECC is preserved.

**Sort BAM**

The output of refcor is a set of BAM files in bead-id sorted order. These files are sorted in this step using Picard sort into coordinate sorted order before merge. These BAM files are referred to as mini-BAM files and are stored in the analysis temporary directory.

**Merge BAM**

This phase accepts a set of mini-BAM files and outputs a set of BAM files. These BAM files are stored in the analysis output directory (which is typically a network-attached storage device).

One BAM file is generated per read-set. For a 96-barcoded dataset in a single XSQ file, either 96 or 97 BAM files are created. The 97th BAM contains reads that have the default barcode index.

**Quality alignments and unique alignments**

We strongly recommend using mapping quality values, rather than uniqueness, as the standard for recognizing or filtering good alignments. A mapping quality value estimates the probability that the alignment is correct, and takes several indications into account, including the number of alignments or uniqueness of an alignment.
The parameter `bamgen.mqv.threshold`, new in release 2.5.1, is available to filter alignments written to your BAM output files. In order to limit your BAM files to good quality reads, set `bamgen.mqv.threshold` to a positive integer such as 8 or higher. This technique ensures that you do not see poor quality reads in your BAM file, and that your tertiary analysis is limited to good quality reads.

**How a quality value is derived**

The following factors increase confidence and increase an alignment's quality value:

- Longer alignment
- Fewer possible alignments
- Fewer mismatches within the alignment

These factors reduce confidence and reduce an alignment's quality value:

- Shorter alignment
- More possible alignments
- More mismatches within the alignment

**How a quality value is interpreted**

For any given alignment, a quality value is a measure of the confidence of a read aligning to the particular location, given all possible alignments for the read. Quality values are within the range 0–100:

- 0 – The highest probability of error
- 100 – The lowest probability of error

Quality values are Phred-scaled quality scores, logarithmically linked to error probabilities. A mapping QV of 10 means that we calculate there is a 90% chance that the alignment is correct. A mapping QV of 20 means that we calculate there is a 99% chance that the alignment is correct.

**Unique alignments**

Unique alignments might be confused with high quality alignments, but uniqueness is not by itself a good criteria to judge an alignment. Uniqueness only means that one step of secondary mapping, the aligner, reports one alignment. This information is a part of the total information generated during mapping, but does not contain all of the information known about the alignment. In contrast, the mapping quality value is a numerical representation of all considerations about the alignment.

For example, a read might have multiple alignments. If the quality value for only one of those multiple alignments is high, that probability of that alignment being good is high. That alignment should be used in your analysis, even though it is not unique.

For a unique alignment, its quality value must be examined to determine if the alignment should be taken or filtered out. Its uniqueness by itself is irrelevant to the decision about whether it is a good read.

**Single-read mapping parameters**

Table 101 describes the algorithm parameters that you can configure for single-read whole transcriptome mapping.
This table lists mapping, resource, parallelization, and BAM-generation parameters. WT tertiary parameters are listed in their chapters:

- Chapter 20, Run a WT Coverage Analysis
- Chapter 21, Run a WT Count Known Genes and Exons Analysis
- Chapter 22, Run a WT Splice Finder Analysis.

Table 101 Single-read WTA mapping parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mapping parameters</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>enable.filter.mapping</td>
<td>1</td>
<td>Enables filter mapping.</td>
</tr>
<tr>
<td>enable.junction.mapping</td>
<td>1</td>
<td>Enables junction mapping.</td>
</tr>
</tbody>
</table>
| mapping.in.base                    | false   | Whether to map in base space (true) or color space (false). Set to true if input data has base space available. Allowed values:  
  - false: Map in color space.  
  - true: Map in base space. If only color space is available, then the module fails when base space mapping is turned on. |
| wt.merge.min.junction.overhang     | 8       | Alignments to junctions are not reported if they have fewer than this number of bases unaligned on either side of the splice. Allowed values: Integers 0–100. |
| **Optional resource parameters**  |         |                                                                            |
| java.heap.space                    | 1500    | Dynamic memory requirement, in megabytes. Do not enter a unit with the numeric value. |
| mapping.memory                     | 15gb    | The total memory, in gigabytes, that is available for map reads, per compute node. Include the units gb in the setting. This number is set to 15 GB because the minimum hardware requirement for memory is 16 GB.  
  **Note:** Typically, the job scheduler will allocate on nodes that have the requested memory available. For mixed hardware, it is suggested that different queues are created based on the memory available. Mapping jobs should be launched on high memory systems. For human references (hg18, hg19) and a 25.2 scheme, 45 GB is recommended for fastest performance. The smallest recommended RAM for human reference is 19 GB. |
| mapping.schema.file                | —       | The mapping schemas file.                                                  |
| **Optional parallelization parameters** |     |                                                                            |
| fragmap.number.of.nodes            | 4       | The number of compute nodes available for this analysis.                  |
| fragmap.max.number.of.jobs         | 100     | The maximum number of nodes allowed for mapping.                          |
| fragmap.minreads.per.node          | 4000000 | If the total number of fragments exceeds this amount, then the analysis is distributed across the available nodes. Allowed values: Integers 1–125000000 (125 Million) |
| fragmap.maxreads.per.node          | 15000000| The maximum number of fragments that can be processed on a single node. Allowed values: Integers 1–150000000 (150 Million) |
Mapping performance

Mapping speed depends on hardware properties as well as on the scattering logic. Mapping runs are split into multiple jobs for processing efficiency. The following parameters affect how a mapping run is split:

- **fragmap.number.of.nodes** – Specifies the number of jobs that are created, only if a split is necessary. The size of each split job is approximately the total number of reads divided by the number of jobs.
- **fragmap.max.number.of.jobs** – Determines the maximum number of jobs that can be launched per analysis. Setting this parameter to a very large value can cause the scheduler to behave incorrectly.
- **fragmap.minreads.per.node** – Determines the threshold of beads beyond which splitting occurs.

---

**Table 101 Single-read WTA mapping parameters (continued)**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>processors.per.node</td>
<td>8</td>
<td>The number of processors per node use for mapping. The reads are divided into the number of chunks specified for this parameter.</td>
</tr>
<tr>
<td>wt.filter.mapping.np.per.node</td>
<td>8</td>
<td>The number of processors per node use for filter mapping.</td>
</tr>
<tr>
<td>wt.genomic.mapping.np.per.node</td>
<td>8</td>
<td>The number of processors per node use for genomic mapping.</td>
</tr>
<tr>
<td>wt.junction.mapping.np.per.node</td>
<td>8</td>
<td>The number of processors per node use for junction mapping.</td>
</tr>
<tr>
<td>Bamgen.refcor.addcs</td>
<td>1</td>
<td>Whether or not to add the color sequence to BAM records. If the XSO file does not contain color space, this parameter is ignored, and the resulting BAM file output does not contain color space. See “Refcor” on page 363 for information about refcor. Allowed values: 0,1.</td>
</tr>
<tr>
<td>create.unmapped.bam.files</td>
<td>FALSE</td>
<td>If set to TRUE, creates an additional BAM output file containing unmapped reads. Allowed values: 0,1.</td>
</tr>
<tr>
<td>Bamgen.mqv.threshold</td>
<td>0</td>
<td>Provides control over the contents written to the output BAM file depending on the quality value of the alignment. To preserve only high quality alignments, set this value to a positive integer. Allowed values: Integers 0–255.</td>
</tr>
<tr>
<td>Refcor.base.filter.qv</td>
<td>10</td>
<td>Bases with a quality value below the value of this parameter provided are replaced with ‘N’. Allowed values: Integers 0–255.</td>
</tr>
<tr>
<td>Refcor.reference.weight</td>
<td>8</td>
<td>Reference weight. This parameter is used during base translation. In the read reconstruction process, multiple signals are combined to generate the final base call. This parameter adds weight (in terms of Phred score) to the signals which are compatible with reference. Color combinations that result in a variant are considered compatible with reference. Additional weight helps to eliminate base errors caused by color error(s) during base translation. Allowed values: Integers 0–100.</td>
</tr>
</tbody>
</table>
**Single-read mapping parameters**

- **fragmap.maxreads.per.node** – Determines the largest number of beads that can be mapped per node. The default is based on the minimum scratch space requirements. If you scratch space is smaller or larger, this number can be made smaller or larger respectively.

- **processors.per.node** – This is the number of cores available for mapping analysis on each node.

- **mapping.memory** – For human mapping and 25.2.0 scheme, there are 3 schema lines. Simultaneous handling of schema lines reduces I/O and therefore improves mapping speed.

Memory requirements for mapping jobs are the following:

- Reference: ~6 GB
- 1 schema line: ~13 GB
- 1 schema line and reference: ~19 GB
- 2 schema lines and reference: ~32 GB
- 3 schema lines and reference: ~45 GB

Below 19 GB, the reference is split up and I/O increases significantly. Pre-built hash tables are not used for memory less than 24 GB. LifeScope™ Software supports the reuse of hash tables for human genome reference files (hg18 and hg19) and the default mapping schemes 35.2, 25.2, and 20.1. Pre-calculated color-space are included in the reference directory available with LifeScope™ Software. If system RAM is 24 GB or higher, the hash tables are loaded into memory. Depending on the pipeline, this hash table reuse typically saves 2–3 hours in analysis processing time.

For genomes without pre-calculated hash tables in the reference repository, the mapping module creates hash tables but does not save them for reuse.

In general, set the **mapping.memory** parameter to be 1 GB less than the total system memory (RAM) available.

Mapping only scales when the number of reads that are mapped together is greater than 100 M. Splitting such that the total number of fragments per node is significantly less than 100 M is not recommended. The splitting can be explained using the following pseudo-code.

```java
if (num.reads <= fragmap.minreads.per.node) { num.jobs = 1;
reads.per.job = num.reads / fragmap.number.of.nodes
if (reads.per.job > fragmap.maxreads.per.node) {
    reads.per.job = fragmap.maxreads.per.node
}
num.jobs = ceiling (num.reads / reads.per.job)
if (num.jobs > fragmap.max.number.of.jobs) {
    throw exception;
}
```

In above pseudo-code, the input read-sets (possibly from multiple XSQ files) have **num.reads** that fall in one read length category. **Num.jobs** is the total number of jobs launched for mapping analysis of **num.reads** beads.

In general, the mapping parameters allow mixing and matching single-anchor and multiple-anchor schemes, and also mixing and matching different seed lengths and different numbers of mismatches. For example, an unmapped scheme of 25.2.0, 35.2.0:15:25, 20.1.0, with a repetitive scheme of 35.2.0:30, 45.2.0, 60.2.0, is supported, but not recommended. The impact on performance can be significant.
Table 102 describes the single-read algorithm parameters that we do not recommend changing.

Table 102  Single-read WTA internal parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wt.frag.map.run</td>
<td>1</td>
<td>Enables fragment mapping. The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use.</td>
</tr>
<tr>
<td>wtexonsequenceextractor.secondary.run</td>
<td>1</td>
<td>Enables exon extraction. The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use.</td>
</tr>
<tr>
<td>wtssplicejunctionextractor.run</td>
<td>1</td>
<td>Enables splice junction extraction, which is required for junction mapping. The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use.</td>
</tr>
</tbody>
</table>
| mapping.use.iub.reference | FALSE | Whether or not to support reference sequences that contain IUB codes. Allowed values:  
  • FALSE: Do not support IUB codes in the reference file.  
  • TRUE: Mapping allows matching to either alleles of bi-allelic IUB codes in the reference file. |
| bamgen.primary.output.filter.type | primary_only | Filters the content written to the output BAM file depending on whether the alignment is gapped or not. Note that unmapped reads are considered primary. Allowed values:  
  • no_filtering: Report all alignments.  
  • primary_only: Report all primary alignments (both gapped and ungapped). |
| wt.filter.map.max.hits | 10 | The maximum number of alignments for a read, during filter mapping. Allowed values: Integers 1–100000. |
| wt.genomic.map.max.hits | 10 | The maximum number of alignments for a read, during genomic mapping. Allowed values: Integers 1–100000. |
| wt.junction.map.max.hits | 10 | The maximum number of alignments for a read, during junction mapping. Allowed values: Integers 1–100000. |
| junction.reference.file | — | The junction reference file. Must be a FA or FASTA file. |
| wt.mask.positions | — | A string of 0s and 1s that show positions to be masked or not. If blank [default], no masking is performed. There is no penalty in mapping towards positions that are masked. |
| wt.map.mismatch.penalty | -2.0 | Specifies a negative scoring penalty for mismatch that is used in local alignment mode. Setting this value to the negative of a number greater than the read lengths has the effect of disallowing mismatches. Allowed values: Floats. |

Mapping scheme parameters
Chapter 19 Run a Whole Transcriptome Mapping Analysis

Single-read mapping parameters

The mapping scheme parameters, listed in Table 102, determine the behavior of the mapping module. The LifeScope™ Software mapping module supports two types of mapping, global mapping and seed-and-extend mapping.

- **Global mapping** – With this approach the seed length matches the full read length, and the seed anchors at the beginning of the read. For example, global mapping for 50-bp reads uses a scheme such as \(50.6.0\). (In this example, 50, the seed length, matches the read length, and 0, the seed start position, starts at the beginning of the read.)

- **Seed-and-extend** – The initial alignment step that locates short matches between a read and the reference sequence. The seed specifies the length of the attempted match. This approach is also called local alignment.

### Global mapping

To map a 50 bp read using global mapping, a typical scheme to use is 50.6.0. This allows for up to 6 mismatches. Because the seed length equals the length of the read, there is no extension.

### Seed-and-extend

With the seed-and-extend approach, a mapping scheme parameter value such as \(25.2.0\) is interpreted as follows:

- **First value** – The seed length (25).
- **Second value** – The quantity of allowed mismatches in the seed (2).
- **Third value** – The start site of the seed within the read (0).

**Figure 40** Seeds for local alignments

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>map.scheme.unmapped.1</td>
<td>0.0.0</td>
<td>Mapping scheme used for reads with lengths less than 25.</td>
</tr>
<tr>
<td>map.scheme.unmapped.25</td>
<td>25.2.0</td>
<td>Mapping scheme used for reads with lengths 25–34.</td>
</tr>
<tr>
<td>map.scheme.unmapped.35</td>
<td>25.2.0:10</td>
<td>Mapping scheme used for reads with lengths 35–49.</td>
</tr>
<tr>
<td>map.scheme.unmapped.50</td>
<td>25.2.0:20</td>
<td>Mapping scheme used for reads with lengths 51–74.</td>
</tr>
<tr>
<td>map.scheme.unmapped.75</td>
<td>25.2.0:20</td>
<td>Mapping scheme used for reads with lengths of 75 and greater.</td>
</tr>
<tr>
<td>map.scheme.unmapped.variable</td>
<td>25.2.0:20</td>
<td>Mapping scheme used for trimmed reads.</td>
</tr>
</tbody>
</table>

**Figure 40** Seeds for local alignments
For example, with a 25.2.0 seed extend scheme, the mapping module examines the first 25 bases of a 50 bp read, and if that portion aligns to the reference with two or fewer mismatches, then the mapping module attempts seed extension. Seed extension begins at the anchor alignment and extends along the read, to find the best ungapped local alignment with a score $S$. The alignment length can be shorter than the read length.

For information on sliding periodic seeds, refer to the publication at this site:

http://bioinformatics.oxfordjournals.org/content/25/19/2514.full

### Paired-end parameters

Table 103 describes the algorithm parameters that you can configure for paired-end whole transcriptome mapping.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mapping parameters</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>enable.exon.mapping</td>
<td>1</td>
<td>Enables exon mapping.</td>
</tr>
<tr>
<td>enable.filter.mapping</td>
<td>1</td>
<td>Enables filter mapping.</td>
</tr>
<tr>
<td>enable.junction.mapping</td>
<td>1</td>
<td>Enables junction mapping.</td>
</tr>
<tr>
<td>wt.f3.rescue.enabled</td>
<td>true</td>
<td>Whether or not to attempt rescue of F3 reads.</td>
</tr>
<tr>
<td>wt.f5.rescue.enabled</td>
<td>true</td>
<td>Whether or not to attempt rescue of F5 reads.</td>
</tr>
<tr>
<td>std.insert.size</td>
<td>60</td>
<td>The standard deviation of the insert size. Allowed values: Integers 1–100000.</td>
</tr>
</tbody>
</table>
| mapping.in.base            | FALSE   | Whether to map in base space (TRUE) or color space (FALSE). Set to TRUE if input data has base space available. Allowed values:  
• **FALSE**: Map in color space.  
• **TRUE**: Map in base space. If only color space is available, then the module fails when base space mapping is turned on. |
| wt.merge.min.junction.overhang | 8    | Alignments to junctions are not reported if they have fewer than this number of bases unaligned on either side of the splice. Allowed values: Integers 0–100. |
| **Splice junction extractor parameters** |         |                                                                              |
| read.length                | 75      | Read length. Allowed values: Integers 25–100.                                 |
| **Rescue parameters**      |         |                                                                              |
| avg.insert.size            | 120     | The average the insert size. Used to compute the rescue distance:  
rescueDistance = avgInsertSize + (3 * stdInsertSize)  
Allowed values: Integers 1–100000. |
| wt.f3.rescue.max.mismatches.allowed | 8 | The maximum number of mismatches allowed in a rescued F3 alignment.  
Allowed values: Integers 0–15. |
Chapter 19  Run a Whole Transcriptome Mapping Analysis

**Paired-end parameters**

Table 103  Paired-end WTA mapping parameters *(continued)*

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wt.f5.rescue.max.mismatches.allowed</td>
<td>6</td>
<td>The maximum number of mismatches allowed in a rescued F5 alignment. Allowed values: Integers 0–10.</td>
</tr>
</tbody>
</table>

**Optional resource parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.heap.space</td>
<td>1500</td>
<td>Dynamic memory requirement, in megabytes. Do not enter a unit with the numeric value.</td>
</tr>
<tr>
<td>mapping.memory</td>
<td>15gb</td>
<td>The total memory, in gigabytes, that is available for map reads, per compute node. This number is set to 15 GB because the minimum hardware requirement for memory is 16 GB. <strong>Note:</strong> Typically, the job scheduler will allocate on nodes that have the requested memory available. For mixed hardware, it is suggested that different queues are created based on the memory available. Mapping jobs should be launched on high memory systems. For human references (hg18, hg19) and a 25.2 scheme, 45 GB is recommended for fastest performance. The smallest recommended RAM for human reference is 19 GB.</td>
</tr>
</tbody>
</table>

**Optional parallelization parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fragmap.number.of.nodes</td>
<td>4</td>
<td>The number of compute nodes available for this analysis.</td>
</tr>
<tr>
<td>fragmap.max.number.of.jobs</td>
<td>100</td>
<td>The maximum number of nodes allowed for mapping.</td>
</tr>
<tr>
<td>fragmap.minreads.per.node</td>
<td>4000000</td>
<td>If the total number of fragments exceeds this amount, then the analysis is distributed across the available nodes. Allowed values: Integers 1–125000000 (125 Million).</td>
</tr>
<tr>
<td>fragmap.maxreads.per.node</td>
<td>150 M</td>
<td>The maximum number of fragments that can be processed on a single node. Allowed values: Integers 1–150000000 (150 Million).</td>
</tr>
<tr>
<td>processors.per.node</td>
<td>8</td>
<td>The number of processors per node use for mapping. The reads are divided into the number of chunks specified for this parameter.</td>
</tr>
</tbody>
</table>

Table 104 describes WTA paired-end algorithm parameters that we do not recommend changing.

Table 104  Paired-end WTA internal parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wt.pe.map.run</td>
<td>1</td>
<td>Enables paired-end mapping. The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use.</td>
</tr>
</tbody>
</table>

**Splice junction parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wt.f3.splext.output.reference</td>
<td>—</td>
<td>The name of the reference file output by the splice junction extractor stage.</td>
</tr>
<tr>
<td>wt.gtf.file</td>
<td>—</td>
<td>The GTF file used to create the genemodel. The default is $(annotation.gtf.file).</td>
</tr>
</tbody>
</table>

**Splice junction extractor parameters**
Table 104  Paired-end WTA internal parameters (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| wtspljunctionextractor.run                    | 1     | Enables splice junction extraction, which is required for junction mapping.  
|                                                |       | The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use. |

**Exon extractor parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| wtexonsequenceextractor.secondary.run         | 1     | Enables exon extraction, which is required for F5 pairing.                  
|                                                |       | The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use. |
| wt.f5.exseqext.output.reference              |       | The name of the reference file output by the exon extractor stage.          |

**Algorithm parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| mapping.use.iub.reference                     | FALSE | Whether or not to support reference sequences that contain IUB codes.       
|                                                |       | Allowed values:                                                           |
|                                                |       | • FALSE: Do not support IUB codes in the reference file.                   |
|                                                |       | • TRUE: Mapping allows matching to either alleles of bi-allelic IUB codes in the reference file. |
| wt.map.mismatch.penalty                       | -2.0  | Specifies a negative scoring penalty for mismatch that is used in local alignment mode. Setting this value to the negative of a number greater than the read lengths has the effect of disallowing mismatches. 
|                                                |       | Allowed values: Floats.                                                   |
| wt.f3.filter.map.max.hits                     | 10    | The maximum number of alignments for an F3 read, during filter mapping.    
|                                                |       | Allowed values: Integers 1–100000.                                         |
| wt.f3.genome.map.max.hits                     | 10    | The maximum number of alignments for an F3 read, during genomic mapping.   
|                                                |       | Allowed values: Integers 1–100000.                                         |
| wt.f3.junction.map.max.hits                   | 10    | The maximum number of alignments for an F3 read, during junction mapping.  
|                                                |       | Allowed values: Integers 1–100000.                                         |
| wt.f3.map.mask.positions                      | —     | A string of 0s and 1s that show F3 positions to be masked or not. If blank (default), no masking is performed. 
|                                                |       | There is no penalty in mapping towards positions that are masked.          |
| wt.f3.mapping.schema.file                     | —     | Mapping schema file for F3 reads.                                          |
| wt.f5.filter.map.max.hits                     | 10    | The maximum number of alignments for an F5 read, during filter mapping.    
|                                                |       | Allowed values: Integers 1–100000.                                         |
| wt.f5.genome.map.max.hits                     | 10    | The maximum number of alignments for an F5 read, during genomic mapping.   
|                                                |       | Allowed values: Integers 1–100000.                                         |
| wt.f5.exon.map.max.hits                       | 10    | The maximum number of alignments for an F5 read, during exon mapping.      
|                                                |       | Allowed values: Integers 1–100000.                                         |
### Paired-end parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wt.f5.map.mask.positions</td>
<td>—</td>
<td>A string of 0s and 1s that show F5 positions to be masked or not. If blank (default), no masking is performed. There is no penalty in mapping towards positions that are masked.</td>
</tr>
<tr>
<td>wt.f5.mapping.schema.file</td>
<td>—</td>
<td>Mapping schema file for F5 reads.</td>
</tr>
<tr>
<td>exon.reference.file</td>
<td>exons.fasta</td>
<td>Path to the exon reference file. Must be a FA or FASTA file.</td>
</tr>
<tr>
<td>junction.reference.file</td>
<td>—</td>
<td>Path to the junction reference file. Must be a FA or FASTA file.</td>
</tr>
</tbody>
</table>

### Pairing parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pair.uniqueness.threshold</td>
<td>10.0</td>
<td>This value defines the clear zone. This value is considered only when pairing.output.uniqueness.type is set to -1. If the score of the best alignment is this threshold more than the score of the second best alignment, the best alignment is considered uniquely placed. A threshold of zero means that this uniqueness is strictly adhered to, regardless of the alignment’s relative scores. Allowed values: Floats 0.0–100.0.</td>
</tr>
<tr>
<td>pairing.anchor.max.mismatches</td>
<td>4</td>
<td>Total mismatches for both reads of a pair. Allowed values: Integers 0–100.</td>
</tr>
<tr>
<td>pairing.gapped.max.hits</td>
<td>100:0</td>
<td>For gapped and ungapped alignment determination during rescue, this value determines the maximum number of rescues attempted for each read. If a read has more than this number of hits, rescue is not performed on this bead. The first value is for ungapped rescue, and second, gapped.</td>
</tr>
<tr>
<td>pairing.mismatch.penalty</td>
<td>-2.0</td>
<td>A single penalty applied to alignments, to compare the significance between alignments. Helps evaluate the alignments. Allowed values: Floats -100 to 0.</td>
</tr>
</tbody>
</table>
| pairing.output.uniqueness.type| -1    | Whether to output non-unique hits. Allowed values:  
|                                |       | • -1: A clear zone determines uniqueness. One or two best pairs are output, depending on whether it is unique.  
|                                |       | • 0: All good pairs are output.  
|                                |       | • 1: Only unique good pairs are output.  
|                                |       | Allowed values: Integers -1, 1.                                                                 |

### Rescue parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wt.rescue.input.generation.exon.fuzzy.border.width</td>
<td>10</td>
<td>The amount of slack allowed around exon borders, for doing rescue with fuzzy exon borders. Allowed values: Integers 0–2147483647.</td>
</tr>
<tr>
<td>wt.rescue.input.generation.min.alignment.distance.for.rescue</td>
<td>100000</td>
<td>When rescuing reads that have already been mapped to the reference, this value is the minimum distance between the anchor alignment and any of the alignments of the rescued read downstream of the anchor alignment. Allowed values: Integers 0–2147483647.</td>
</tr>
</tbody>
</table>
### Paired-end parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| wt.rescue.input.generation.rescue.anchor.alignments.not.overlapping.exons | 1 | Enables the additional rescue downstream and upstream of alignments that do not overlap exons but do overlap genes. Allowed values:  
  • 0: No change in behavior.  
  • 1: Enables the rescue described above. |
| wt.rescue.input.generation.rescue.fuzzy.exon.borders | 1 | Enables allowing some slack around exons borders. Allowed values:  
  • 0: No change in behavior.  
  • 1: Enables slack around exons borders. |
| wt.rescue.input.generation.rescue.only.for.the.best.anchor.alignment | 0 | Limits rescue to only for the best anchor alignment. Allowed values:  
  • 0: Rescue for all anchor alignments.  
  • 1: Rescue only for the best anchor alignment. |
| wt.rescue.input.generation.rescue.only.unaligned.reads | 0 | Rescue ignores target reads that already have an alignment. Allowed values:  
  • 0: Rescue all the target reads.  
  • 1: Rescue ignores target reads that already have an alignment. |
| wt.rescue.input.generation.rescue.only.within.rescue.distance | 1 | Rescue only within rescue distance. Rescue only within the rescue region starting from the exon border, to a limit inside the exon. The limit is computed based on the insert size and the location of the anchor alignment inside the previous exon. Allowed values:  
  • 0: Rescue on the entire exon.  
  • 1: Rescue only within rescue distance. |
| wt.rescue.valid.adjacent.mismatches.count.as.one | 0 | Two adjacent mismatches that can account for a SNP count only as one mismatch.  
  • 0: No change in behavior.  
  • 1: Two adjacent mismatches that can account for a SNP count only as one mismatch. |
| wt.rescue.input.generation.rescue.short.range | 1 | Limits rescue to all the target reads in the vicinity of the anchor read. Allowed values:  
  • 0: No change in behavior.  
  • 1: Limits rescue to all the target reads in the vicinity of the anchor read. |
| wt.rescue.input.generation.zLimit | 100 | The Z limit. Allowed values: Integers 0–2147483647. |
| wt.rescue.standalone.rescue.executable | rescue | The name of the executable file used for rescue. |

### Mapping scheme parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wt.f3.mapping.scheme.unmapped.1</td>
<td>0.0.0</td>
<td>Mapping scheme for F3 reads with lengths less than 25.</td>
</tr>
</tbody>
</table>
Mapping output files

Overview

The mapping module generates a BAM file containing alignments in coordinate order. For information about the BAM file, see Appendix C, “File Format Descriptions” on page 493.

The number of BAM files generated depends on the input XSQ files. One BAM file is generated per read-set. If the input data is an XSQ file containing 96 barcodes, then the mapping module generates at least 96 output BAM files. Beads that are unclassified in any barcode are output into a separate additional BAM file. If create.unmapped.bam.files is set to TRUE, then an additional 96 output BAM files corresponding to the unmapped reads are also generated.

The filenames for the output BAM files are created using information from the input XSQ file, including: file base name, file id, index name, and index id. The BAM files are named according to the following patterns:

- Non-indexed BAM files: xsqname-fileID-1.bam
- Indexed BAM files: xsqname-fileID-idx_bcIndex-bcID.bam

---

Table 104  Paired-end WTA internal parameters (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wt.f3.mapping.scheme.unmapped.35</td>
<td>25.2.0:10</td>
<td>Mapping scheme for F3 reads with lengths 35–49.</td>
</tr>
<tr>
<td>wt.f3.mapping.scheme.unmapped.50</td>
<td>25.2.0:20</td>
<td>Mapping scheme for F3 reads with lengths 51–74.</td>
</tr>
<tr>
<td>wt.f3.mapping.scheme.unmapped.75</td>
<td>25.2.0:20</td>
<td>Mapping scheme or F3 reads with lengths of 75 and greater.</td>
</tr>
<tr>
<td>wt.f3.mapping.scheme.unmapped.variable</td>
<td>25.2.0:20</td>
<td>Mapping scheme for F3 reads with variable lengths (trimmed reads).</td>
</tr>
<tr>
<td>wt.f5.mapping.scheme.unmapped.1</td>
<td>0.0.0</td>
<td>Mapping scheme for F5 reads with lengths less than 25.</td>
</tr>
<tr>
<td>wt.f5.mapping.scheme.unmapped.25</td>
<td>25.2.0</td>
<td>Mapping scheme for F5 reads with lengths 25–34.</td>
</tr>
<tr>
<td>wt.f5.mapping.scheme.unmapped.35</td>
<td>25.2.0:10</td>
<td>Mapping scheme for F5 reads with lengths 35–49.</td>
</tr>
<tr>
<td>wt.f5.mapping.scheme.unmapped.50</td>
<td>25.2.0:20</td>
<td>Mapping scheme for F5 reads with lengths 51–74.</td>
</tr>
<tr>
<td>wt.f5.mapping.scheme.unmapped.75</td>
<td>25.2.0:20</td>
<td>Mapping scheme for F5 reads with lengths of 75 and greater.</td>
</tr>
<tr>
<td>wt.f5.mapping.scheme.unmapped.variable</td>
<td>25.2.0:20</td>
<td>Mapping scheme for F5 reads with variable lengths (trimmed reads).</td>
</tr>
<tr>
<td>wt.f5.exon.mapping.scheme.unmapped.1</td>
<td>0.0.0</td>
<td>Mapping scheme for F5 reads with lengths less than 25.</td>
</tr>
<tr>
<td>wt.f5.exon.mapping.scheme.unmapped.25</td>
<td>25.3.0</td>
<td>Mapping scheme for F5 reads with lengths 25–34.</td>
</tr>
<tr>
<td>wt.f5.exon.mapping.scheme.unmapped.35</td>
<td>25.3.0:10</td>
<td>Mapping scheme for F5 reads with lengths 35–49.</td>
</tr>
<tr>
<td>wt.f5.exon.mapping.scheme.unmapped.50</td>
<td>25.3.0:20</td>
<td>Mapping scheme for F5 reads with lengths 51–74.</td>
</tr>
<tr>
<td>wt.f5.exon.mapping.scheme.unmapped.75</td>
<td>25.3.0:20</td>
<td>Mapping scheme for F5 reads with lengths of 75 and greater.</td>
</tr>
<tr>
<td>wt.f5.exon.mapping.scheme.unmapped. variable</td>
<td>25.3.0:20</td>
<td>Mapping scheme for F5 reads with variable lengths (trimmed reads).</td>
</tr>
</tbody>
</table>
The fields in the filenames are:

- **xsqname** – The XSQ file base name, without the .xsq extension.
- **fileID** – The file ID for internal XSQ file tracking.
- **bcIndex** – The barcode index.
- **bcID** – The barcode identifier for internal barcode tracking.

The directory structure for mapping output is as follows:

```plaintext
outputs/${analysis.output.dir}/
<mappingIniFileBasename>/
<sampleName1>/*.bam
<sampleName2>/*.bam
```

The directory name `<mappingIniFileBasename>` is the basename of the mapping INI file. The defaults are `wt.frag.map` and `wt.pe.map`.

The string `<sampleName*>` is determined by the sample description for the particular read-set.

The mapping output files are used by the BAMStats mapping statistics module and by the whole transcriptome tertiary analysis modules: WT counts, WT coverage, and WT splice junctions.

### BAM file differences

This section describes two types of differences seen in WT BAM files, compared to other LifeScope™ Software BAM files.

#### Single-read optional fields

A BAM file produced by the WT paired-end pipeline is identical to that produced by the resequencing pipeline. However, the BAM format from the WT single-read pipeline differs from BAM files produced elsewhere in LifeScope™ Software. The single-read pipeline produces separate BAM files for mapped and unmapped reads.

**Note:** The WTA single-read pipeline produces a BAM file that uses the optional fields described in **Table 105**.

<table>
<thead>
<tr>
<th>Optional field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IH:i:</td>
<td>Number of stored alignments containing the current query.</td>
</tr>
<tr>
<td>HI:i:</td>
<td>Query hit index.</td>
</tr>
<tr>
<td>NH:i:</td>
<td>Number of reported alignments containing the current query.</td>
</tr>
<tr>
<td>CS:z:</td>
<td>Color read sequence.</td>
</tr>
<tr>
<td>CQ:z:</td>
<td>Color quality sequence.</td>
</tr>
<tr>
<td>CC:z:</td>
<td>Reference name of the next hit.</td>
</tr>
<tr>
<td>CP:z:</td>
<td>Coordinate of the next hit.</td>
</tr>
<tr>
<td>AS:i:</td>
<td>locationAlignment Score generated by the aligner.</td>
</tr>
<tr>
<td>XN:i:</td>
<td>Alignment score of the best non-primary alignment for query in the current record.</td>
</tr>
<tr>
<td>XF:z:</td>
<td>T for true or F for false. Set to &quot;T&quot; if this read is filtered.</td>
</tr>
</tbody>
</table>
Table 105  WT single-read pipeline BAM file optional fields (continued)

<table>
<thead>
<tr>
<th>Optional field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XJ:z:</td>
<td>“K” for Known Junction or “P” for Putative Junction.</td>
</tr>
</tbody>
</table>

**Clipped records**

By design, BAM records produced in WT modules may have soft-clips in the CIGAR string. For records that have soft-clips at the beginning, the correct start position of reported sequence is interpreted as the start position minus the soft-clipped length. However, BAM-consuming external tools that are not BAM-specification complaint may present soft-clipped records as shifted or missed alignments.

**WT filtering stats**

An example whole transcriptome human filter reference is provided under the `<examplesdir>/demos/WholeTranscriptome/references` folder in the optional LifeScope™ Software examples distribution. This example reference fasta includes contigs from barcode primers, human ribosomal RNAs, tRNAs, and other known targets for filtering. The filter reference may be expanded or removed with more csfasta reference records at the discretion of the user.

The stats for filtered reads are generated after a LifeScope™ Software run. First two lines of the stats file states the number of reads processed, and the number of reads mapped to the filtered reference followed by its percentage. Each following line reports a contig name and the count of reads that aligned to that contig. Paired-end filtering stats may be found in the Intermediate folder. The following is a truncated example of filtering stats output:

```plaintext
countOfReadsProcessed: 175,839,941
countOfReadsMapping:  18,783,220 (10.7%)

... Barcode-047-3-end reverse  83
Barcode-048-3-end reverse  2,350
gi|124517659|ref|NR_003286.1| Homo sapiens 18S ribosomal RNA (LOC100008588)  5,175,899
gi|142372596|ref|NR_003285.2| Homo sapiens 5.8S ribosomal RNA (LOC100008587)  78,936
gi|124517661|ref|NR_003287.1| Homo sapiens 28S ribosomal RNA (LOC100008589)  7,616,212
chr6.trna95-AlaAGC (58249908-58249836) Ala (AGC) 73 bp Sc: 42.26   5
chr6.trna25-AlaAGC (26859897-26859969) Ala (AGC) 73 bp Sc: 46.89   2
chr6.trna94-AlaAGC (58250620-58250548) Ala (AGC) 73 bp Sc: 54.62   4
chr6.trna160-AlaAGC (26881822-26881750) Ala (AGC) 73 bp Sc: 54.69   35
```
Mapping statistics

Mapping statistics occur after mapping as an optional post-processing step named BAMStats. BAMStats accepts the output of the mapping step and generates statistics files to provide an in-depth understanding of the experimental data and to better detect the presence of anomalies. LifeScope™ Software shell users can display the mapping statistics output data as a chart with a spreadsheet program or other third-party program. A subset of the output from mapping can be visualized in the LifeScope™ Software UI as a series of line and bar charts, if the analysis is run in the projects repository.

**Note:** BAMStats does not accept 1.x BAM files as input. 1.3 BAM files must be converted to 2.5.1 BAMs before being used with BAMStats.

### Mapping statistics parameters

Table 106 lists the parameters which control the BAMStats output.

**Table 106**  
BAMStats parameter description

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mandatory parameters</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bamstats.maximum.coverage</td>
<td>500000</td>
<td>Defines the maximum coverage allowed for locations in the reference. Locations with coverage more than the maximum coverage value are ignored during coverage calculations. Allowed values: Integers &gt;= 1.</td>
</tr>
<tr>
<td>bamstats.maximum.isize</td>
<td>100000</td>
<td>The maximum inset size for LMP and PE libraries. Reads with an insert size more then the specified value are ignored for Insert Range Report calculation. Allowed values: Integers 1–100000.</td>
</tr>
</tbody>
</table>
| bamstats.wig.primary.only           | 1             | Use only primary alignments for coverage in WIG file format. Allowed values:  
• 0: Do not restrict coverage in WIG file format to only primary alignments.  
• 1: Restrict coverage in WIG file format to only primary alignments. |
| bamstats.wig.binsize                | 100           | The bin size for coverage in WIG file format. Allowed values: Integers 1–100000. |
| bamstats.bin.isize                  | 100           | The bin size for insert range distribution. Allowed values: Integers 1–100000. |
| bamstats.wig.combined.report.both.strands | 0         | Whether to generate a WIG format coverage file with combined data from both the strands. Allowed values:  
• 0: Do not generate wig files.  
• 1: Generate the combined coverage file. |
| Optional resource parameters        |               |                                                                             |
| wall.time                           | 120           | Total time for the process to complete.                                      |
Table 106  BAMStats parameter description (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.heap.space</td>
<td>13000</td>
<td>Dynamic memory requirement, in megabytes. Do not enter a unit with the numeric value.</td>
</tr>
<tr>
<td>number.of.nodes</td>
<td>4</td>
<td>The number of compute nodes available for this analysis.</td>
</tr>
<tr>
<td>memory.request</td>
<td>14gb</td>
<td>Memory request. Include the units gb in the setting.</td>
</tr>
</tbody>
</table>

Table 107 lists BAMStats parameters that we do not recommend changing.

Table 107  BAMStats internal parameter description

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bamstats.run</td>
<td>1</td>
<td>Whether or not to run the BAMStats module. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Do not run the BAMStats module. BAMStats statistics are not generated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Run the BAMStats module during this analysis.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use.</td>
</tr>
<tr>
<td>bamstats.group.stats</td>
<td>1</td>
<td>Enable combined statistics covering all read-sets in the group or sample. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Only statistics for BAM files are generated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Also generate combined statistics for the group or sample.</td>
</tr>
<tr>
<td>bamstats.maximum.mismatches</td>
<td>100</td>
<td>The maximum mismatches allowed in the alignments. Any alignment with more then the specified number of mismatches is ignored while generating reports related to number of mismatches. Allowed values: Integers 0–100.</td>
</tr>
<tr>
<td>bamstats.maximum.baseqv</td>
<td>100</td>
<td>Max base quality values. Any base with base quality value more then the specified value is ignored while generating reports. Allowed values: Integers 0–100.</td>
</tr>
<tr>
<td>bamstats.maximum.mappingqv</td>
<td>255</td>
<td>Maximum mapping quality value. Any alignment with mapping quality value more than the specified value is ignored. Allowed values: Integers 0–255.</td>
</tr>
<tr>
<td>bamstats.wig.minimum.mappingqv</td>
<td>2</td>
<td>Defines the minimum mapping quality allowed for coverage in WIG format. Any alignment with a mapping quality value less than this value is ignored. Allowed values: Integers 0–100.</td>
</tr>
<tr>
<td>bamstats.enable.probe.position</td>
<td>0</td>
<td>Enable probe and position error reports. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Do not generate probe and position error reports. Off is recommended for WT.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Also generate probe and position error reports.</td>
</tr>
</tbody>
</table>

Summary of mapping statistics output

File formats

For every input BAM file, a set of statistics files are generated. These files are in CHT, CSV, TXT, and WIG formats. Each CHT file corresponds to one displayed chart. A CHT file specifies the type of chart, the displayed range of each axis, and the data points, without using external references.
The CHT file format is an internal file format based on the CSV file format, with addition header information included. CHT header information is the following:

- **# name:**
- **# type:** scatter2d | pie | vbar | line
- **# title:**
- **# xaxisname:**
- **# yaxisname:**
- **# xrange:** <min>:<tickinterval>:<max>
- **# yrange:** <min>:<tickinterval>:<max>

The *CHT* file format is an internal file format based on the CSV file format, with addition header information included. CHT header information is the following:

- **# name:**
- **# type:** scatter2d | pie | vbar | line
- **# title:**
- **# xaxisname:**
- **# yaxisname:**
- **# xrange:** <min>:<tickinterval>:<max>
- **# yrange:** <min>:<tickinterval>:<max>

The wiggle format (.wig) is a public format typically used for coverage. Visit their site for more information:

[hgdownload.cse.ucsc.edu/goldenPath/help/wiggle.html](http://hgdownload.cse.ucsc.edu/goldenPath/help/wiggle.html)

A genome browser such as the Integrative Genomics Viewer (IGV) can be used to visualize the coverage. For information is available from their site:

[www.broadinstitute.org/igv/](http://www.broadinstitute.org/igv/)

For a collection of input BAM files that belong to a sample, a set of cumulative statistics files are generated. The cumulative statistics files are also in CHT, CSV, TXT, and WIG formats. The cumulative statistics can be visualized in the LifeScope™ Software UI.

**Summary information**

A comma-separated summary file (*BAMfilename*-summary.tbl) is generated that summarizes key mapping quality statistics per input BAM file. This file contains one row for each input BAM file in the sample. The summary file is displayed in the LifeScope™ Software UI. See “Summary file” on page 387.

**Directory structure**

The output of the BAMStats module has the following directory structure:

```
bamstats/
  <sampleName>/*.cht, *.tbl
  <sampleName>/*<bam>/*.cht
  <sampleName>/bam/Misc/*.csv, *.txt, *.wig
  <sampleName>/Misc/*.csv, *.txt
```

The position error files and probe errors files are created in the BAM file directories.

**Overview**

The summary report contains a snapshot of statistics in all BAM files present in this sample. This report is displayed in the LifeScope™ Software UI.

```
<sampleName>/BAMfilename-summary.tbl
```

Following directories and reports contain the cumulative statistics from all BAM files that belong to this sample. The Misc folder is not displayed in UI.

```
<sampleName>/*.cht
```
Following statistics are generated per BAM file in the mapping directory. These reports are not displayed in the UI.

In a sample, some of the BAM files possibly represent unhealthy DNA or RNA, causing the cumulative statistics to look poor. The summary tbl file is the unified location for examining the quality of data of all BAM files in the sample.

If the data for a particular BAM file is not as expected, look at the directory-level reports for that BAM file, for details.

### Mapping statistics output files

This section describes the mapping statistics files generated by the BAMStats module. When statistics reports are separated by tag type, the report’s file name includes the tag in the file name. The tags used in file names are:

<table>
<thead>
<tr>
<th>Library type</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fragment</td>
<td>F3</td>
</tr>
<tr>
<td>Mate-pair</td>
<td>F3</td>
</tr>
<tr>
<td></td>
<td>R3</td>
</tr>
<tr>
<td>Paired-end</td>
<td>F3</td>
</tr>
<tr>
<td></td>
<td>F5-P2</td>
</tr>
</tbody>
</table>

The output files generated by BAMStats include the name of the BAM file. The file name pattern is:

\[ \text{BAMFileName}-\text{fileID}-\text{barcodeID}.\text{StatisticsReportName}.\text{tag}.extension \]

This string is referred to by \textit{prefix} in Table 109, the mapping statistics output table. In the description of mapping statistics output file names in Table 109, the string \textit{tag} is used to refer to the tags in Table 108.

A raw numbers file is generated for each chart. See “Raw number files” on page 386.

<table>
<thead>
<tr>
<th>Report</th>
<th>Description, axis information, filename</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alignment Length Distribution</strong></td>
<td>A bar plot giving the distribution of alignment lengths found in various tags used in alignment. The report is separated by tag type to provide visibility into the accuracy of individual tags. Only the primary alignment for each bead is considered in calculating the distribution.</td>
</tr>
<tr>
<td>Y axis: Frequency</td>
<td>X axis: Alignment length (from 0 to the maximum read length)</td>
</tr>
<tr>
<td>Output file name: \textit{prefix}Alignment.Length.Distribution.\textit{tag}.cht</td>
<td></td>
</tr>
</tbody>
</table>

| **Base Mismatch Distribution**     | |
|------------------------------------| |

---

<sampName>/Misc/*.csv, *.wig, *.txt

<sampName>/<BAMfilename>/.*cht
<sampName>/<BAMfilename>/Misc/*.csv, *.txt, *.wig

<sampName>/BAMfilename>/.*cht
<sampName>/BAMfilename>/Misc/*.csv, *.txt, *.wig
### Table 109  Mapping statistics output files (continued)

<table>
<thead>
<tr>
<th>Report</th>
<th>Description, axis information, filename</th>
</tr>
</thead>
</table>
| **Distribution of Alignment Length and Number of Mismatches in Tags** | A report providing a simultaneous picture of the alignment length and number of mismatches distributions in various tags used in alignment. Only primary alignments for each tag are considered in calculating this distribution. The bins range from 0 to the maximum read length, and from 0 to the maximum number of mismatches allowed. This report is located in the **Misc** folder, and is not displayed in the UI.  
  
  Z axis: Frequency  
  Y axis: Alignment length (from 0 to the maximum mismatches allowed)  
  X axis: Number of mismatches (from 0 to the maximum mismatches allowed)  
  
  Output file name: `prefix.AlignmentLength.Mismatch.tag.csv` |
| **Distribution of Mismatches by Position** | A bar plot providing a distribution of errors (mismatches to reference) by position within the read. Only the primary alignments for each bead are used to generate this distribution. Mismatch frequency is calculated for each position by dividing number of mismatches for that position by total number of bases for that specific position. The base-space report is generated on every run.  
  
  Y axis: Mismatch percentage by read position  
  X axis: Position (from 0 to maximum read length)  
  
| **Color Mismatch Distribution** | A bar plot providing a distribution of the total number of color mismatches in various tags used in alignment. Only the primary alignments for each bead are used to generate this distribution. This report is only generated if mapping is done in color space.  
  
  Y axis: Frequency  
  X axis: Number of mismatches (from 0 to the maximum mismatches allowed)  
  
| **Distribution of Alignment Length and Color Mismatches** | |
Table 109  Mapping statistics output files (continued)

<table>
<thead>
<tr>
<th>Report</th>
<th>Description, axis information, filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>A report providing a simultaneous picture</td>
<td>A report providing a simultaneous picture of the alignment length and number of color mismatches of the alignment length and number of color mismatches distributions in various tags used in alignment. Only primary alignments for each tag are considered in calculating this distribution. The bins range from 0 to the maximum read length, and from 0 to the maximum number of mismatches allowed. This report is only generated if mapping is done in color space.</td>
</tr>
<tr>
<td>Y axis: Alignment length (from 0 to the max read length)</td>
<td></td>
</tr>
<tr>
<td>X axis: Number of mismatches (from 0 to the maximum mismatches allowed)</td>
<td></td>
</tr>
<tr>
<td>Z axis: Frequency</td>
<td></td>
</tr>
<tr>
<td>Output file name: <code>prefix.AlignmentLength.ColorMismatches.tag.csv</code></td>
<td></td>
</tr>
</tbody>
</table>

**Base QV Distribution**

A bar plot providing a distribution of base quality values generated using the reference assisted error correction/base conversion algorithm. The base QV distributions are separated for each tag as quality of individual tags could be very different.

Y axis: Frequency
X axis: Base QV (from 0 to the maximum QV)
Output file name: `prefix.BaseQV.tag.cht`

**Base QVs by Position**

A report providing a distribution of base quality values by individual base positions. This report identifies if certain base positions (particularly towards the end of the read) have poor base quality values. All the tags (F3/R3/F5-P2) used in the alignment are combined into a single bin.

Y axis: Base QV
X axis: Base position (from 0 to read length)
Z axis: Frequency
Output file name: `prefix.BaseQV.by.Position.csv`

**Distribution of Mismatches by Base QV**

A line plot providing a distribution of errors (mismatches to reference) by base quality values bins.

Y axis: Mismatch percentage by Base QV
X axis: Base QV (from 0 to maximum QV)

This report gives, for each QV, a measure of the raw mismatches at positions with that QV. This information does not measure whether the base QVs generated are well calibrated to the probability of error in that particular base position.


**Distribution by Mapping QVs by Tag**

A bar plot providing a distribution of mapping quality values for individual tags (F3/R3/F5-P2). Only the primary alignment for each bead is used in calculating this distribution.

Y axis: Frequency
X axis: Mapping QV (from 0 to maximum mapping QV)
Output file name: `prefix.MappingQV.tag.cht`

**Distribution by Pairing QVs**
### Table 109  Mapping statistics output files (continued)

<table>
<thead>
<tr>
<th>Report</th>
<th>Description, axis information, filename</th>
</tr>
</thead>
</table>
| **A bar plot providing a distribution of pairing quality values for individual tags (F3/R3/F5-P2). Only the primary alignment for each bead is used in calculating this distribution.**<br><br>Y axis: Frequency<br>X axis: Pairing QV (from 0 to maximum pairing QV)<br>Output file name: `prefix.PairingQVs.tag.cht`<br><br>**Coverage Report**<br><br>A line plot providing a distribution of coverage obtained after mapping/pairing. Only the primary alignment for each bead is used in this calculation.<br><br>Y axis: Number of bases<br>X axis: Coverage (from 0 to number of reads)<br>Output file name: `prefix.Coverage.tag.cht`<br><br>**Coverage Report by Chromosome (Contig) and Base Windows**<br><br>A line plot providing a distribution of coverage within each reference window. The coverage is calculated within each window along a reference chromosome. The window size can theoretically be anywhere from a single base to the contig length. However the calculations of base level coverage are computationally expensive and less interpretable as the window size increases. Only the primary alignment for each bead is used in this calculation.<br><br>Y axis: Coverage (from 0 to number of reads)<br>X axis: Frequency<br>Output file name: `prefix.Coverage.By.Chromosome.contignn.tag.cht`<br><br>**Coverage by Strand**<br><br>A line plot providing the distribution of coverage within each reference window separated by reference strand (+/-). Only the primary alignment for each bead is used in this calculation.<br><br>Y axis: Frequency<br>X axis: Coverage (from 0 to number of reads)<br>Output file name: `prefix.Coverage.By.Strand.tag.cht`<br><br>**Coverage files**<br><br>Coverage reports in wiggle format. For each genome position, reports the number of reads that cover (map to or span) the position. Because reporting coverage for each position results in very large files, coverage is reported for bins, with each bin spanning a user-defined number of bases. For each bin, the mean coverage of all the positions in that bin is reported. The parameter `bamstats.wig.binsize` controls the size of the bins in this file.<br><br>Each coverage file includes a header as the first line. The header lines follow this pattern:<br><br>`track type=wiggle_0 name=<chrname> description=<coverage from positive/negative/both strand> visibility=full color=0,0,255 fixedStep chrom=<chrname> start=<startpos> step=<binsize> span=<binsize>`<br><br>See also "Coverage files" on page 389.<br><br>Output file name: `coverage_chrnn.POS.wig, coverage_chrnn.NEG.wig`<br><br>**Insert Range Distribution**<br><br>A line plot providing the distribution of insert sizes for paired data (PE and LMP library types).<br><br>Y axis: Frequency<br>X axis: Insert Range bins<br>Output file name: `prefix.Insert.Range.Distribution.cht`
### Table 109  Mapping statistics output files (continued)

<table>
<thead>
<tr>
<th>Report</th>
<th>Description, axis information, filename</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Distribution of Read Pair Types</strong></td>
<td>For paired data, this report calculates the distribution of read pair types (AAA, AAB, C**, etc.).</td>
</tr>
<tr>
<td></td>
<td>Y axis: Frequency</td>
</tr>
<tr>
<td></td>
<td>X axis: Read pair type</td>
</tr>
<tr>
<td></td>
<td>Output file name: prefix.ReadPair.Type.cht</td>
</tr>
<tr>
<td><strong>Pairing Statistics</strong></td>
<td>A pie chart showing the percentages of the following F3, R3 (or F5) combinations, for paired data:</td>
</tr>
<tr>
<td></td>
<td>• Mapped, Mapped</td>
</tr>
<tr>
<td></td>
<td>• Mapped, Unmapped</td>
</tr>
<tr>
<td></td>
<td>• Unmapped, Mapped</td>
</tr>
<tr>
<td></td>
<td>• Unmapped, Unmapped</td>
</tr>
<tr>
<td></td>
<td>• Mapped, Missing</td>
</tr>
<tr>
<td></td>
<td>• Missing, Mapped</td>
</tr>
<tr>
<td></td>
<td>• Unmapped, Missing</td>
</tr>
<tr>
<td></td>
<td>• Missing, Unmapped</td>
</tr>
<tr>
<td></td>
<td>Output file name: prefix.Pairing.Stats.cht</td>
</tr>
<tr>
<td><strong>Unique Start Position</strong></td>
<td>This report lists the following:</td>
</tr>
<tr>
<td></td>
<td>• The total number of unique start positions.</td>
</tr>
<tr>
<td></td>
<td>• The average number of unique start positions per reference position.</td>
</tr>
<tr>
<td></td>
<td>See also &quot;Unique Start Position&quot; on page 389.</td>
</tr>
<tr>
<td></td>
<td>Output file name: BAMfilename-summary.txt, Samplename.-summary.txt</td>
</tr>
<tr>
<td><strong>Summary Report</strong></td>
<td>The summary file provides statistics for each BAM file in the sample. See &quot;Summary file&quot;.</td>
</tr>
<tr>
<td></td>
<td>Output file name: filename.-summary.tbl</td>
</tr>
</tbody>
</table>

### Raw number files

For each CHT and CSV file listed in Table 109, a corresponding TAB file contains the raw numbers and percentages of the distributions, and a total. The following are example contents of the prefix.Coverage.cht and the corresponding prefix.Coverage.tab file with raw number information:

```plaintext
prefix.Coverage.cht:
# name: Coverage Report (N in reference not counted as covered)
# type: line
# title: Coverage Report (N in reference not counted as covered)
# xaxisname: Coverage
# yaxisname: Percent
# xrange:
# yrange: 0:10:100
Coverage,Percent
0,95.6421
1,2.2274
2,0.1928
3,0.0570
```
This section provides example output of some mapping statistics output files.

**Summary file**

The summary file provides statistics for each BAM file in the sample. This list describes labels used in the summary file:

- **NumFragmentsTotal** – The total number of beads, before any filtering on the instrument. This value is also the fragment count in the input XSQ file.
- **NumFragmentsPassingFilters** – The number of beads that pass filtering on the instrument.
- **NumMapped** – The number of reads with primary alignment.
- **% filtered mapped** – The number of primary reads divided by the number of beads passing instrument filtering ($\text{NumMapped} / \text{NumFragmentsPassingFilters}$).
- **% total mapped** – The number of primary reads divided by the number of total beads ($\text{NumMapped} / \text{NumFragmentsTotal}$).
- **Megabases of coverage** – The sum of alignment lengths, for primary alignments only.

The number of beads filtered out in the instrument is found by subtracting the number of beads that pass filtering from the total number of beads:

$$\text{NumFragmentsTotal} - \text{NumFragmentsPassingFilters}$$

The following is example output of a paired summary file. The Tag2 and PredictedInsertSize are not present with fragment data. Each column of data corresponds to a BAM file.

#title: BAMStats Summary

```
4, 0.0323
5, 0.0256
6, 0.0236
7, 0.0234
8, 0.0238
9, 0.0247
```
## Chapter 19 Run a Whole Transcriptome Mapping Analysis

### Mapping statistics output files

<table>
<thead>
<tr>
<th>BamFileName</th>
<th>solid0054_20100203_PE_chr6P_PE.LTcolor-1-1.bam</th>
<th>solid0107_20100203_PE_chr6P_PE.LTcolor-3-1.bam</th>
<th>solid0107_20100408_PE_chr6P_PE.LTcolor-2-1.bam</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsColorInBam</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>IsBaseInXSQ</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>LibraryType</td>
<td>PE</td>
<td>PE</td>
<td>PE</td>
</tr>
<tr>
<td>ReadLength</td>
<td>50x25</td>
<td>50x25</td>
<td>50x25</td>
</tr>
<tr>
<td>PredictedInsertSize</td>
<td>165</td>
<td>165</td>
<td>165</td>
</tr>
<tr>
<td>Megabases of Coverage</td>
<td>517151581</td>
<td>504214985</td>
<td>555762980</td>
</tr>
<tr>
<td>NumFragmentsPassingFilters</td>
<td>8916035</td>
<td>8280061</td>
<td>9519990</td>
</tr>
<tr>
<td>NumFragmentsTotal</td>
<td>8916035</td>
<td>8280061</td>
<td>9519990</td>
</tr>
<tr>
<td>Tag1-NumMapped</td>
<td>8021248</td>
<td>7653534</td>
<td>8601958</td>
</tr>
<tr>
<td>Tag1- % total Mapped</td>
<td>89.96</td>
<td>92.43</td>
<td>90.36</td>
</tr>
<tr>
<td>Tag1- % filtered mapped</td>
<td>89.96</td>
<td>92.43</td>
<td>90.36</td>
</tr>
<tr>
<td>Tag2-NumMapped</td>
<td>5887893</td>
<td>5994800</td>
<td>6118791</td>
</tr>
<tr>
<td>Tag2- % total Mapped</td>
<td>66.04</td>
<td>72.40</td>
<td>64.27</td>
</tr>
<tr>
<td>Tag2- % filtered mapped</td>
<td>66.04</td>
<td>72.40</td>
<td>64.27</td>
</tr>
<tr>
<td>Tag1-AlignmentLength</td>
<td>F3</td>
<td>F3</td>
<td>F3</td>
</tr>
<tr>
<td>Min</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>Max</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>Avg</td>
<td>46.12</td>
<td>46.30</td>
<td>46.83</td>
</tr>
<tr>
<td>Median</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>StdDev</td>
<td>7.46</td>
<td>6.91</td>
<td>6.65</td>
</tr>
<tr>
<td>Tag1-MappingQV</td>
<td>F3</td>
<td>F3</td>
<td>F3</td>
</tr>
<tr>
<td>Min</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Max</td>
<td>60</td>
<td>60</td>
<td>60</td>
</tr>
<tr>
<td>Avg</td>
<td>36.99</td>
<td>37.34</td>
<td>37.14</td>
</tr>
<tr>
<td>Median</td>
<td>48</td>
<td>48</td>
<td>49</td>
</tr>
<tr>
<td>StdDev</td>
<td>22.28</td>
<td>22.11</td>
<td>22.12</td>
</tr>
<tr>
<td>Tag1-BaseQV</td>
<td>F3</td>
<td>F3</td>
<td>F3</td>
</tr>
<tr>
<td>Min</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Max</td>
<td>41</td>
<td>41</td>
<td>41</td>
</tr>
<tr>
<td>Avg</td>
<td>35.22</td>
<td>36.16</td>
<td>36.31</td>
</tr>
<tr>
<td>Median</td>
<td>41</td>
<td>41</td>
<td>41</td>
</tr>
<tr>
<td>StdDev</td>
<td>10.69</td>
<td>9.83</td>
<td>9.66</td>
</tr>
<tr>
<td>Tag1-NumColorMismatches</td>
<td>F3</td>
<td>F3</td>
<td>F3</td>
</tr>
<tr>
<td>Min</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Max</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Avg</td>
<td>1.38</td>
<td>1.48</td>
<td>1.33</td>
</tr>
<tr>
<td>Median</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>StdDev</td>
<td>1.66</td>
<td>1.67</td>
<td>1.66</td>
</tr>
<tr>
<td>Tag2-AlignmentLength</td>
<td>F5-P2</td>
<td>F5-P2</td>
<td>F5-P2</td>
</tr>
<tr>
<td>Min</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>Max</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>Avg</td>
<td>25.00</td>
<td>25.00</td>
<td>25.00</td>
</tr>
<tr>
<td>Median</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>StdDev</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Tag2-MappingQV</td>
<td>F5-P2</td>
<td>F5-P2</td>
<td>F5-P2</td>
</tr>
</tbody>
</table>
Mapping statistics output files

<table>
<thead>
<tr>
<th></th>
<th>F5-P2</th>
<th>F5-P2</th>
<th>F5-P2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Max</td>
<td>60</td>
<td>60</td>
<td>60</td>
</tr>
<tr>
<td>Avg</td>
<td>35.59</td>
<td>36.83</td>
<td>34.41</td>
</tr>
<tr>
<td>Median</td>
<td>48</td>
<td>49</td>
<td>46</td>
</tr>
<tr>
<td>StdDev</td>
<td>23.62</td>
<td>22.99</td>
<td>23.63</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tag2-BaseQV</th>
<th>F5-P2</th>
<th>F5-P2</th>
<th>F5-P2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Max</td>
<td>41</td>
<td>41</td>
<td>41</td>
</tr>
<tr>
<td>Avg</td>
<td>33.72</td>
<td>34.78</td>
<td>33.52</td>
</tr>
<tr>
<td>Median</td>
<td>41</td>
<td>41</td>
<td>41</td>
</tr>
<tr>
<td>StdDev</td>
<td>11.85</td>
<td>11.37</td>
<td>11.98</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tag2-NumColorMismatches</th>
<th>F5-P2</th>
<th>F5-P2</th>
<th>F5-P2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Max</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Avg</td>
<td>1.36</td>
<td>1.31</td>
<td>1.49</td>
</tr>
<tr>
<td>Median</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>StdDev</td>
<td>1.16</td>
<td>1.19</td>
<td>1.20</td>
</tr>
</tbody>
</table>

Coverage

<table>
<thead>
<tr>
<th></th>
<th>F5-P2</th>
<th>F5-P2</th>
<th>F5-P2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Max</td>
<td>5978</td>
<td>5922</td>
<td>7072</td>
</tr>
<tr>
<td>Avg</td>
<td>0.17</td>
<td>0.16</td>
<td>0.18</td>
</tr>
<tr>
<td>Median</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>StdDev</td>
<td>3.15</td>
<td>3.14</td>
<td>3.48</td>
</tr>
</tbody>
</table>

Unique Start Position

The following is an example of a Unique Start Position report, for fragment data:

```
Unique Start Positions F3
Total Number of Unique Start Positions: 16536 (16.043154 % of primary alignments)
Average Number of Unique Start Positions per Reference Position: 0.001630
```

Unique Start Position reports have the following naming patterns:

- `BAMfilename-summary.txt`
- `Samplename-summary.txt`

Coverage files

Example contents for the file `coverage_chr1_positive.wig` are:
```
browser position chr1:1-200000000
browser hide all
browser pack refGene encodeRegions
# minimumMapq=25, minimumCoverage=1, alignmentFilteringMode=PRIMARY, filterOrphans=false,
track name="BAM Coverage positive strand" description="BAM Coverage positive strand" visibility="full color 0,0,255"
priority=10 yLineMark=0 type=wiggle_0 yLineOnOff=on
```
variableStep chrom=chr1 span=1
33617 1
33618 1
33619 1
33620 1
33621 1
33622 1
33623 1
33624 1
33625 1
33626 1
33627 1
...

Example contents for the file coverage_chr1_negative.wig are:

```plaintext
browser position chr1:1-200000000
browser hide all
browser pack refGene encodeRegions
# minimumMapq=25, minimumCoverage=1,
alignmentFilteringMode=PRIMARY, filterOrphanedMates=false,
track name="BAM Coverage negative strand" description="BAM Coverage negative strand" visibility="full color 0,0,255"
priority=10 yLineMark=0 type=wiggle_0 yLineOnOff=on
variableStep chrom=chr1 span=1
5743 2
5744 2
5745 2
5746 3
5747 3
5748 3
5749 3
5750 3
5751 3
5752 3
5753 3
...
```

Run BAMStats standalone

This section describes requirements to generate mapping statistics outside of the context of the mapping module. These requirements are:

- **Directory structure** – The input directory for BAMStats module must mimic the output directory structure of mapping:
  
  ```plaintext
  Input.BAMStat.dir/
  <sampleName1>/*.bam, *.bai
  <sampleName2>/*.bam, *.bai
  ```
• **Index files** – For every BAM file, there must be a corresponding BAI-formatted indexed file. BAI files can be generated using the following command:

```bash
samtools index <bamfile>
```

This command generates index sorted alignment for fast random access. The index file `<bamfile>.bai` is created. For details, please refer to the samtools site:


• **Write permission** – The user must also have write permission for the input directory, because the BAMStats module generates position and probe error files in that directory.

### FAQ – Whole transcriptome

1

**Why is SNPs analysis not included in the whole transcriptome pipeline?**

A SNPs analysis is not offered for WTA pipelines. The reverse transcription process that is part of the WT library prep intrinsically adds a percentage of sequencing errors. In addition, alternative splicing and other alignment complexities around exon junctions may cause local misalignments, which can lead to false positive SNP calls. Using that data in a SNPs analysis may cause a high number of false positives.

2

**What reads scenario achieves the best accuracy?**

The best performance is achieved by considering 2BE and 4BE reads, their alignment, and reference. Typically, base-space reads generated by ECC have lower mapping throughput. Base-space reads generated from combined 2BE and 4BE calls and no reference have lower accuracy, compared to ECC calls.

3

**In the mapping statistics genome coverage calculation report, how are Ns in the reference counted?**

Ns in the reference are counted as missing coverage. For example, in the human genome, about 7% of the reference sequence consists of Ns. This means that the coverage calculation will never be reported as higher than 93%. If the frequency distribution says 7.49% of the genome is uncovered, this means that about 0.34% of the non-N reference sequence does not have reads mapped to it, and that also the 7.15% of the genome that consists of N does not have reads mapped to it.
What should I do to ensure balanced allele ratios at heterozygous positions?

The base-translation algorithm, refcor, uses instrument color-space data (ECC or non-ECC) and the reference sequence to produce the most likely base calls, along with a Phred-scale quality value. The degree to which the reference influences the base call is controlled by the refcor.reference.weight parameter. A position is converted to reference only if the refcor.reference.weight setting is higher than the difference between the quality values of a correct color call and of an adjacent erroneous color call that supports reference.

The refcor.reference.weight default setting, 8, is chosen to be low enough to ensure balanced allele ratios, and also be large enough both to distinguish correct from incorrect color calls and to maximize base accuracy.

In the event an incorrect reference base call is made, the disagreement between evidence sources causes that call’s QV to be low. Incorrect calls can be filtered out based on QV threshold, and this filtering improves allele ratios.

Another way of filtering low quality base-calls is to adjust the parameter refcor.base.filter.qv, that converts all base calls with a quality value less than refcor.base.filter.qv into Ns. This filtering has the effect of setting QV to zero in the calls with low confidence. To decrease the number of N base calls, reduce this threshold (for example, from the default of 10 to 5).

Note: Setting the refcor.reference.weight parameter to zero completely eliminates the reference as a guide for base-translation (completely eliminates reference bias), but also considerably reduces the quality of base-translation.

How is the uniqueness of a pair determined?

A pair of reads is unique when there is exactly one good AAA pair. With the advent of local alignment, the likelihood of finding only one good pair is decreased. As a result, a different heuristic is used to determine “uniqueness”.

Consider an alignment of length L with M mismatches. If the local score is defined as \(L+(m-1)M\), where \(m=0\) is the mismatch penalty, then the score of each good pair candidate is the sum of its two constituent local scores.

A pair of tags is considered unique if it has only one good pair, or if the score of the best pair is at least \(X\) greater than the score of the second best pair. \(X\) is specified by the pair.uniqueness.threshold key, and has a default value of 10.0. See Table 104 on page 372 and Table 103 on page 371 for a description of this and all pairing parameters.

What do the 3-letter pair classifications mean?

Table 110 lists a summary of genomic code classifications for mate-pair pairing runs. See Table 111 on page 394 for a summary of genomic code classifications for paired-end runs.
The first letter determines whether the two tags match the same strand:
- A – Match the expected strand.
- B – Match different strands.

The third letter determines if the distance between two hit positions is within the correct range:
- A – Within the correct range.
- B – The distance is less than the insert size measurement minimum (denoted by "< min" in the table above).
- C – The distance is greater than the insert size measurement maximum (denoted by "> max" in the table above).

The default minimum insert size is 0. For paired-end and long mate pair workflows, the default maximum insert sizes are 2000 and 20,000, respectively. An insert size measurement automatically determines the actual values using the defaults as the largest interval, and the defaults are used if the insert size measurement fails.

The middle letter has different meanings, depending on whether the first letter is A or B. If the first letter is A, the second letter indicates if the order of the two tags is correct:
- A – The order is correct, and R3 is upstream of F3.
- B – The order is incorrect, and F3 is upstream of R3.

When the two tags are on different strands, the second letter indicates whether the two tags are pointing toward or away from each other
- A – Pointing away from each other.
- B – Pointing toward each other.

In addition, C** classification is assigned to a pair of reads that have one hit each, but are on different chromosomes. If both F3 and R3 tags are present, but only one has a unique hit while the other one is unmapped, the pair is labeled D**. Finally, if one tag from a pair has a unique hit but the other one is missing, as opposed to unmapped, then the pair is classified as E**.
The Preference value indicates likelihood of the variation can occur in nature. Larger values indicate the variation is less likely to occur in nature.

Table 111 shows the classifications for paired-end runs. Summary statistics of these categories are calculated in the optional BAMStats Read pair Type Statistics Report (.ReadPair.Type.cht).

Table 111 Genomic code classifications, for paired-end pairing

<table>
<thead>
<tr>
<th>Class name</th>
<th>Strand</th>
<th>Orientation</th>
<th>Insert size</th>
<th>Preference</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAA</td>
<td>Different</td>
<td>Inward</td>
<td>Normal</td>
<td>1</td>
</tr>
<tr>
<td>AAB</td>
<td>Different</td>
<td>Inward</td>
<td>&lt; min</td>
<td>2</td>
</tr>
<tr>
<td>AAC</td>
<td>Different</td>
<td>Inward</td>
<td>&gt; max</td>
<td>2</td>
</tr>
<tr>
<td>ABA</td>
<td>Different</td>
<td>Outward</td>
<td>Normal</td>
<td>2</td>
</tr>
<tr>
<td>ABB</td>
<td>Different</td>
<td>Outward</td>
<td>&lt; min</td>
<td>3</td>
</tr>
<tr>
<td>ABC</td>
<td>Different</td>
<td>Outward</td>
<td>&gt; max</td>
<td>3</td>
</tr>
<tr>
<td>BAA</td>
<td>Same</td>
<td>F3 to F5</td>
<td>Normal</td>
<td>3</td>
</tr>
<tr>
<td>BAB</td>
<td>Same</td>
<td>F3 to F5</td>
<td>&lt; min</td>
<td>4</td>
</tr>
<tr>
<td>BAC</td>
<td>Same</td>
<td>F3 to F5</td>
<td>&gt; max</td>
<td>4</td>
</tr>
<tr>
<td>BBA</td>
<td>Same</td>
<td>F5 to F3</td>
<td>Normal</td>
<td>3</td>
</tr>
<tr>
<td>BBB</td>
<td>Same</td>
<td>F5 to F3</td>
<td>&lt; min</td>
<td>4</td>
</tr>
<tr>
<td>BBC</td>
<td>Same</td>
<td>F5 to F3</td>
<td>&gt; max</td>
<td>4</td>
</tr>
</tbody>
</table>

How do I run an WT analysis with ERCC controls?

Use the following ERCC files:
- ERCC_Controls.fa
- ERCC_Controls.gtf

You can append the file ERCC_Controls.fa to either your filter reference or your genomic reference. If you append the file ERCC_Controls.fa to your genomic reference and also append the file ERCC_Controls.gtf to the annotation GFF file, then WT counts include the ERCC information.

Check the LifeScope™ Software project page for the location of the ERCC_Controls files.

FAQ – BAMStats

1

In the BAMStats summary table, why do the NumFragmentsPassingFilters and NumFragmentsTotal fields have the same totals?
Chapter 19  Run a Whole Transcriptome Mapping Analysis

FAQ – BAMStats

What changes are required for a job with a large number of contigs?

To run BAMStats with a large number of contigs (around 500 or more) and a very high bamstats.maximum.coverage value (around 300,000 or more), you can increase the heap space or decrease the maximum coverage value, or do both:

- Decrease the bamstats.maximum.coverage value.
  You could try 100000 or 10000 for the maximum coverage value.
- Increase heap space by increasing the memory.request parameter for the mapping module, and increase the java.heap.space parameter in the bamstats.properties file.

Note: The heap space and memory request parameters directly affect job scheduling and impact run times. The values to use also depend on your data sets and compute environment.

(It is also possible, but not recommended in most cases, to change the system property max.head.node.java.heap.space from the default of 64 to 128 or 256. This parameter is in the <installdir>/etc/analysis/system.properties file. Because changes to system properties affect all runs of all users, you could consider changing the parameter back to its original value after the large contig run completes. Setting max.head.node.java.heap.space to 128 or 256 could present a problem if your head node is extremely loaded.)

NumFragmentsTotal reports the total number of fragments (read pairs or reads) passed from the instrument. NumFragmentsPassingFilters reports the number of fragments that pass the instrument filter. For the 5500 instrument, all reads pass filters when using color space, and so these two numbers are the same. If the run is performed with ECC, then the base space reads have filters applied to remove those of lowest quality. For an ECC run mapped in base space using LifeScope™ Software, the filtered reads are reported in LifeScope™ Software, and the Total and Filtered numbers are different.
Chapter 19  Run a Whole Transcriptome Mapping Analysis

FAQ – BAMStats
Run a WT Coverage Analysis

This chapter covers:
- Overview ................................................................. 397
- WT coverage input files ............................................. 397
- WT coverage parameters ......................................... 398
- Coverage output files .............................................. 399

Overview

The whole transcriptome coverage module calculates read coverage per position.

WT coverage input files

The WT coverage module takes as input one or more BAM files containing mapped data. The following describe the alignment input accepted. This module:
- Accepts only fragment data for RNA libraries.
- Accepts multiple BAM files as input.
- Accepts BAM files with different read lengths.
- Accepts both color space and base-space files, and a combination of color and base-space data.
## WT coverage parameters

Table 112 lists the WT coverage parameters.

### Table 112  WT coverage parameter description

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
</table>
| coverage.run                    | 1             | Whether or not to run the WT coverage module. Allowed values:  
|                                 |               |  
|                                 |               | • 0: Do not run the coverage module.  
|                                 |               | • 1: Run the coverage module during this analysis.  
|                                 |               | The run parameter is set automatically by the shell during a  
|                                 |               | standard workflow. Accept the default for most use.  
| RNA.bam.file                    | —             | A comma-separated list of paths to the input BAM files. Relative  
|                                 |               | and absolute paths are accepted.  
|                                 |               | Generated automatically by LifeScope™ Software during a  
|                                 |               | workflow. Use this parameter only when running this module  
|                                 |               | standalone.  

### Optional parameters

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
</table>
| RNA.coverage.min.quality        | 2             | (Optional) The mapping quality value threshold for selecting the  
|                                 |               | alignments. Alignments with a mapping quality value lower than  
|                                 |               | this threshold are not written to output.  
|                                 |               | Allowed values: Integers 0–100.  
| RNA.coverage.min.value          | 0             | (Optional) The mapping quality threshold value for selecting  
|                                 |               | alignments from the input BAM. In order to be included in the  
|                                 |               | output, a position’s coverage must be greater than or equal to this  
|                                 |               | value. Allowed values: Integers >= 0.  
| RNA.coverage.per.chromosome     | true          | (Optional) Whether coverage output is generated as one file per  
|                                 |               | every chromosome per strand, or as a single file with coverage of  
|                                 |               | all chromosomes per strand. Allowed values:  
|                                 |               | • false: Generate a single file with coverage of all chromosomes  
|                                 |               | per strand.  
|                                 |               | • true: Generate one file per every chromosome per strand.  
| RNA.coverage.primary.only       | true          | (Optional) Whether only primary alignments or all the alignments  
|                                 |               | from the input BAM file are used as input. Allowed values:  
|                                 |               | • false: Consider all alignments.  
|                                 |               | • true: Consider only primary alignments (both gapped and  
|                                 |               | ungapped).  

### Resource parameters

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
</table>
| memory.request                  | 4gb           | Memory request. Include the units gb in the setting.  
| java.heap.space                 | 4000          | Dynamic memory requirement, in megabytes. Do not enter a unit with the numeric value.  

Coverage output files

The WT coverage module's output files are in the wiggle (*.wig) file format. Wiggle files are visualized in genomics browsers such as the Integrative Genomics Viewer (IGV). You can download the IGV browser from the Broad Institute website:

www.broadinstitute.org/igv

The wiggle file specification is available from this site:

http://genome.ucsc.edu/goldenPath/help/wiggle.html

If the parameter RNA.coverage.per.chromosome is set to false, the WT coverage module generates two output files with coverage of all chromosomes per strand:

- coverage_positive.wig
- coverage_negative.wig

If the parameter RNA.coverage.per.chromosome is set to true, the WT coverage module generates one coverage file per chromosome per strand:

- coverage_chr**_positive.wig
- coverage_chr**_negative.wig
Chapter 20  Run a WT Coverage Analysis
Coverage output files
Run a WT Count Known Genes and Exons Analysis

This chapter contains:

- Overview ................................................................. 401
- Input files ............................................................... 402
- WT counts parameter description ................................. 402
- WT counts parameter description ................................. 402
- Output files ............................................................ 405
- FAQ – WT counts ...................................................... 406

Overview

The module performs counting of features provided in a GTF file. It counts:

- Exons
- Genes

The module allows filters, generates RPKM, and annotates how many exons are expressed in a gene. This tool annotates only the Exon features in a GTF file.

**IMPORTANT!** Alignments must come from the same strand as the feature to contribute to the count. A non-gapped tag contributes to a feature’s count if it overlaps the feature and has no more than three bases outside the feature. A gapped tag contributes to a features count if one of its match regions terminates at a feature boundary.
Chapter 21  Run a WT Count Known Genes and Exons Analysis

Input files

Use the WT counts module to count the number of reads that align within genomic features. The WT counts module takes the following as input:

- A set of read alignments (BAM file)
- A set of user-defined (or default) stringency parameters
- A set of genome annotations (GTF file)

Paired-end and fragment libraries may be combined in the BAM input. Unmapped reads in the BAM input are not used.

WT counts parameter description

Table 113 describes parameters that you can configure for the WT counts module.

Table 113  Counts module algorithm parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm parameters</td>
<td></td>
<td></td>
</tr>
<tr>
<td>wt.gtf.file</td>
<td>${annotation.gtf.file}</td>
<td>The GTF file used to create the genemodel.</td>
</tr>
<tr>
<td>wt.count.features.gene.count</td>
<td>1</td>
<td>Generate whole transcriptome gene count. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Do not generate whole transcriptome gene count.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Generate whole transcriptome gene count.</td>
</tr>
<tr>
<td>wt.count.features.primary</td>
<td>1</td>
<td>Whole transcriptome counts primary. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Count all alignments.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Count only primary alignments. Recommended.</td>
</tr>
<tr>
<td>wt.count.features.min.mapq</td>
<td>10</td>
<td>The minimum alignment quality for an alignment to be counted. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Integers 0–255.</td>
</tr>
<tr>
<td>wt.count.features.overflow.limit</td>
<td>3</td>
<td>Whole transcriptome overflow limit. An alignment may align outside the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>boundaries of a given exon feature (called overflowing). If an alignment</td>
</tr>
<tr>
<td></td>
<td></td>
<td>overflows the feature more than the value specified in this parameter, then</td>
</tr>
<tr>
<td></td>
<td></td>
<td>that read is not counted towards that feature. Both sides of the exon are</td>
</tr>
<tr>
<td></td>
<td></td>
<td>checked for overflows, and the sum of all overflowing bases (on both sides)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>is compared to this threshold.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allowed values: Integers 0–100.</td>
</tr>
<tr>
<td>Optional resource parameters</td>
<td></td>
<td></td>
</tr>
<tr>
<td>java.heap.space</td>
<td>8000</td>
<td>Dynamic memory requirement, in megabytes. Do not enter a unit with the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>numeric value.</td>
</tr>
<tr>
<td>memory.request</td>
<td>8gb</td>
<td>Memory request. Include the units gb in the setting.</td>
</tr>
</tbody>
</table>
Table 114 lists WT counts parameters that we do not recommend changing.

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wt.exon.reference</td>
<td>The exon FASTA reference file. Default is <code>${analysis.output.dir}/wtexonsequenceextractor/exons.fasta</code>.</td>
</tr>
</tbody>
</table>

Table 115 lists parameters that you do not specify when you run the module in a standard workflow.

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Description</th>
</tr>
</thead>
</table>
| wt.counts.run | Enable the WT counts module. Allowed values: 
- 0: Do not run the module. 
- 1: Run the WT counts module. Default. |
| wt.exon.reference | The exon FASTA reference file. Default is `${analysis.output.dir}/wtexonsequenceextractor/exons.fasta`. This file is defined by the wt.f5.exseqext.output.reference parameter in the WT exon sequence extractor. |
| wt.gtf.file | The GTf file used to create the genemodle. Default: `${annotation.gtf.file}`. |

### WT counts algorithm description

**Filters**

Stringency parameters govern the set of alignments in a BAM file. These alignments are considered in counting. Reads that do not meet these stringency parameters are filtered out and do not contribute to the result.

These parameters govern stringency:

- Filter alignment mode (`wt.count.features.primary`)
- Minimum mapping quality (`wt.count.features.min.mapq`)

The filter alignment mode governs how multi-mapping reads are handled. The option settings for filter alignment mode are all and primary (see Table 116). Both options are supported for single-read and for paired-end data.

Table 116 Alignment filter modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>All alignments are considered.</td>
</tr>
<tr>
<td>primary</td>
<td>Only primary alignments are considered.</td>
</tr>
</tbody>
</table>

If a minimum mapping quality is selected, only alignments with the minimum quality or greater are considered. Likewise, if a minimum score is selected, only alignments having at least the minimum score are considered.

Filtering by unique or top criteria is included mainly for legacy reasons. We recommend that you use only default setting of mapping quality to filter reads.
Half (0.5) a count is assigned for each exon of a read that spans an exon-exon junction.

Mapping quality incorporates an assessment of score and uniqueness. To filter using only mapping quality specify a minimum mapping quality, and set the alignment filter mode to all, and specify a minimum mapping quality.

In previous releases of LifeScope™ Software, filtering single-read alignments using the unique alignment filter mode was recommended. When unique filtering is enabled, only alignments that satisfy the following criteria pass the filter:

- The total number of alignments for the read is less than the maximum allowed to be reported in mapping (the Z parameter).
- The alignment must be the single best alignment of the set of alignments for the read.
- The score of the alignment must be sufficiently better than that of any suboptimal hits. If no suboptimal hits are present, a suboptimal hit is assumed to exist with a score one less than the minimum possible score for the anchor size and mismatch level used in mapping. The difference between the score of the alignment score and the best suboptimal alignment must be greater than or equal to the clear zone.

Counts

Each feature defined in the genome annotation is assigned a tag count. A tag count is the total number of alignments that pass the stringency filters and are consistent with the annotation. The criteria for counting an alignment toward a feature are:

- The alignment must overlap the feature’s numeric coordinates (contig, start, end).
- The alignment strand orientation must be consistent with the feature strand orientation.

**Note:** F3 reads align to the sense genomic strand; F5 reads align to the antisense genomic strand.

- If the alignment does not span an intron, the alignment must include no more than three bases outside the feature.
- If the alignment spans an intron, the position of the exon-intron boundary must match in the alignment and the feature.

The read count for a given gene is defined as number of reads that map to the maximal region (features) of the merged gene. Maximal features are calculated from the transcripts provided in the GTF file by using the gene model. Reads that map to overlapping exons or features of a given gene are not counted multiple times.

The following figure describes the component flow during exon counting.
Chapter 21  Run a WT Count Known Genes and Exons Analysis

Output files

This section describes the output of the WT counts module.

GTF output

The output format of the WT counts module follows the same GTF file format as the supplied genome annotation. The score field of the GTF file contains the tag count. RPKM is reported in the attributes field.

The following is an example GTF output.

```
# filterOrphanedMates=false, minMapq=10, minScore=0, scoreClearZone=5,
# number_of_mapped_reads=190375154
chr1  BioScope exon  4225  4692  86 - .  gene_id "WASH5P"; transcript_id "NR_024540"; RPKM "0.97";
chr1  BioScope exon  4833  4901  5 - .  gene_id "WASH5P"; transcript_id "NR_024540"; RPKM "0.38";
chr1  BioScope exon  5659  5810  18 - .  gene_id "WASH5P"; transcript_id "NR_024540"; RPKM "0.62";
```

Tab output

The following fields are in the `gene_counts.tab` output file:

- Gene name
- Chr
- Start
- End
- Strand
- ReadCount
- Length (the maximal gene region length)
- RPKM
- Number of features (exons)
- Number of expressed features (>10 reads)
- Transcript names

Every gene is reported whether it is expressed or not.

RPKM

The RPKM of exons (the Reads Per Kilobase of exon model per Million mapped reads), which is a normalized measure of expression, is also reported. This metric is calculated with the formula:

\[
\text{RPKM} = 10^9 \times \frac{\text{ExonReadCount}}{\text{TotalReadCount} \times \text{ExonLength}}
\]

The number of total uniquely mapped reads on exons is used as the normalization factor. The RPKM value is also reported by the WT splice finder module (see “Splice finder parameters” on page 408).
FAQ – WT counts

1

Does the whole transcriptome module count invalid reads?

A: No. If a read is invalid (that is, considered invalid by Picard software), that read is not included in the count of mapped reads.
Run a WT Splice Finder Analysis

This chapter contains:
- Overview .................................................. 407
- Splice finder parameters .................................. 408
- SASR splice finder ........................................... 411
- Output files .................................................. 414

Overview

Transcripts of eukaryotic genomes are transcribed from exons which are usually separated by one or more introns. The spliceosome machinery removes the introns from pre-messenger-RNAs to generate mature messenger-RNAs. The removed introns are called splice junctions. Alignments from an RNA-Seq experiment may map on these splice junctions, meaning the alignment has two parts that are separated by thousands of bases on the genome. The splice finder software is designed to discover these splice junctions efficiently and annotate the type and evidence for each discovered junction.

A fusion junction is a section of transcribed RNA that maps to an exon from one gene followed by an exon from another gene. It can occur as the result of a translocation, deletion, or chromosom al inversion. A splice junction excludes exon-to-exon boundaries that arise from alternative splicing for a gene.

The module takes as input:
- A BAM file of either paired-end or fragment libraries reads sequenced from transcribed RNA
- A list of exons. Each exon includes its color sequence and the reverse of that sequence.
- A GTF file
- A FASTA file of the reference genome

Unmapped reads in the BAM input are not used. Partially mapped reads are used to find gene fusions.
Splice finder parameters

Table 117 describes a workflow parameter, analysis.space, that is used in the splice finder module. The analysis.space parameter is set in the tertiary/global.ini file.

Table 117 Analysis space parameter used in the splice finder module

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>
| analysis.space      | auto    | Specifies your preference for base space or color space analysis. Allowed values:  
|                     |         | • auto: The analysis space is determined based on the mapping space and the presence of color space data in the analysis BAM files.  
|                     |         | • base: Prefer base space analysis.                                           
|                     |         | • color: Prefer color space analysis.                                        |
|                     |         | See “Analysis space” on page 69 for a description of other factors determining the analysis space used by the splice finder module. |
Chapter 22 Run a WT Splice Finder Analysis

Splice finder parameters

Table 118 Splice finder parameter description (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wt.splice.finder.single.read.clip.total</td>
<td>10</td>
<td>Total size for clipping at the end of read. Allowed values: Integers 0–25.</td>
</tr>
</tbody>
</table>
| wt.splice.finder.single.read.ReportMultihit         | 0             | How to count single reads that align to multiple splice junctions. Allowed values:  
|                                                   |               | • 0: Do not count them as evidence to any of the junctions.  
|                                                   |               | • 1: Count them as evidence to all of the junctions.  
|                                                   |               | • 2: Count them as evidence to the first of the junctions.  
|                                                   |               | Option 0 provides uniqueness. |
| wt.splice.finder.single.read.remap                  | 0             | Single read remap. 0=false, 1=true. Allowed values:  
|                                                   |               | • 0: SASR registers the evidence, but does not remap.  
|                                                   |               | • 1: SASR remaps reads that have already been mapped to a junction, and registers evidence for that remap. |
| wt.splice.finder.single.read.clip.5.prime           | 1             | Single read clip 5 prime. Allowed values:  
|                                                   |               | • 0: SASR clips only the 3’ end.  
|                                                   |               | • 1: SASR clips both the 5’ and the 3’ end of the read. |
| wt.splice.finder.single.read.min.read.length        | 37            | SASR considers shorter reads (in number of colors) to be inadmissible as evidence. Allowed values: Integers 0–100. |
| Optional paired read junction evidence collection    |               |            |
| wt.splice.finder.paired.read                        | 1             | Enable paired-end fusion finding. Allowed values:  
|                                                   |               | • 0: Do not run paired-end junction finding.  
|                                                   |               | • 1: Enable paired-end junction finding. |
| wt.splice.finder.paired.read.min.mapq                | 10            | A read-pair that has mapping quality lower than this is inadmissible as evidence. Allowed values: Integers 0–225. |
| wt.splice.finder.paired.read.avg.insert.size        | 120           | Average insert size of the paired-end library. Allowed values: Integers 0–1000. |
| wt.splice.finder.paired.read.std.insert.size        | 60            | Standard deviation of the insert size for the paired-end library. Allowed values: Integers 0–1000. |
| Combined caller for junction                        |               |            |
| The parameters below allow splice finder to combine evidence from single-reads and from paired-reads to call junctions, alternative splices, and fusions, respectively. In each case, an event (for example, a fusion) is reported if SRE>x and PRE>y and (|SRE+PRE|>z or MAX(SRE,PRE) > w), where  
|                                                   |               | SRE = unique single-read evidence,  
|                                                   |               | PRE = unique paired-read evidence, and  
|                                                   |               | x, y, z, and w are specified by the parameters below.  
| wt.splice.finder.junction.split.read.min.evidence  | 1             | Minimum single read evidence for junction. Allowed values: Integers 0–50. |
### Table 118 Splice finder parameter description (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
</table>
| `wt.splice.finder.junction.paired.end.min.evidence` | 1 | y in PRE>y  
Minimum paired-end read evidence for junction.  
Allowed values: Integers 0–50. |
| `wt.splice.finder.junction.combined.min.evidence` | 2 | z in (SRE+PRE)>z  
Minimum combined evidence for junction.  
Allowed values: Integers 0–50. |

**Combined caller for (same gene) alternative splicing**

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
</table>
| `wt.splice.finder.alt.splicing.split.read.min.evidence` | 1 | x  
Minimum single read evidence for alternative splicing.  
Allowed values: Integers 0–50. |
| `wt.splice.finder.alt.splicing.paired.end.min.evidence` | 1 | y  
Minimum paired-end evidence for alternative splicing.  
Allowed values: Integers 0–50. |
| `wt.splice.finder.alt.splicing.combined.min.evidence` | 2 | z  
Minimum combined evidence for alternative splicing.  
Allowed values: Integers 0–50. |

**Combined caller for gene fusion**

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
</table>
| `wt.splice.finder.fusion.split.read.min.evidence` | 2 | x  
Minimum split read evidence for junction.  
Allowed values: Integers 0–50. |
| `wt.splice.finder.fusion.paired.end.min.evidence` | 2 | y  
Minimum paired-end evidence for junction.  
Allowed values: Integers 0–50. |
| `wt.splice.finder.fusion.combined.evidence` | 4 | z  
Minimum combined evidence for junction.  
Allowed values: Integers 0–50. |

**Other options**

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
</table>
| `wt.splice.finder.output.format` | 3 | Output format parameter. Allowed values:  
• 1: Tabular output format  
• 2: BED output format  
• 3: BED, tabular, and SEQ output formats  
• 4: All formats, including Circos.  
With option 3, both tabular, BED, and also additional SEQ files are created separately. SEQ files contain 50 base pairs of sequences from each end of a junction exon and are useful for validation. |
**SASR splice finder**

**SASR splice finder module summary**

1. Read in and process the exon list.
2. For each admissible read:
   a. Let $L$ be the set of exons that map to the prefix of the read.
   b. Let $R$ be the set of exons that map to the suffix of the read.
   c. For each exon $e$ in $L$, and each exon $f$ in $R$,
      \[
      \text{if } \text{overlap}(e, r) + \text{overlap}(f, r) + 1 = \text{length}(r),
      \text{ then register read } r \text{ as evidence for junction } e-f.
      \]
3. Output the list of junctions and evidence.

---

**Table 19**  Splice finder internal parameters

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
</table>
| wt.splice.finder.base.space     | 0             | Call junctions in base space. Allowed values:  
   • 0: Call junctions in color space.  
   • 1: Call junctions in base space. |

**Optional resource parameters**

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.heap.space</td>
<td>14000</td>
<td>Dynamic memory requirement, in megabytes. Do not enter a unit with the numeric value.</td>
</tr>
<tr>
<td>memory.request</td>
<td>15gb</td>
<td>Memory request. Include the units gb in the setting.</td>
</tr>
</tbody>
</table>

**Table 20**  Splice finder parameters supplied by LifeScope™ Software

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Description</th>
</tr>
</thead>
</table>
| splice.finder.run      | Enable the splice finder module. Allowed values:  
   • 0: Do not run the splice finder module.  
   • 1: Run the splice finder module. Default. |
| wt.exon.reference      | The exon FASTA reference file. Default is `${analysis.output.dir}/wtexonsequenceextractor/exons.fasta`. This file is defined by the `wt.f5.exseqext.output.reference` parameter in the WT exon sequence extractor. |
| wt.gtf.file            | The GTf file used to create the genemodel. Default: `${annotation.gtf.file}`. |
After the single-read and paired-end splice finders fully process the BAM file, the evidence evaluator evaluates the existing evidence and makes junction calls based on user-defined thresholds.

Pairs mapped less than a confidence threshold (set with the parameter `wt.splice.finder.paired.read.min.mapq`) are not used. By default, reads with an original length of less than 37 are not considered for splicing evidence. The parameter `wt.splice.finder.single.read.min.read.length` changes the default. Spliced reads are used as evidence unless remap is enabled (with the parameter `wt.splice.finder.single.read.remap`). When remap is enabled, the spliced read is remapped and then used as evidence if it falls on a junction.

The default thresholds require at least one single-read and one paired-end evidence to call a junction. The calling of special junctions, such as alternative splicing and fusion, is described in the following three paragraphs.

The evidence evaluator also counts the number of unique pieces of evidence. In the case of paired-end evidence, this count is the number of unique mate-alignment start positions, and in the case of single-read evidence, it is the number of unique overlap lengths for the “left” exon.

The candidate’s data structure stores the evidence for junctions as properties of its exons. Each exon points to other exons for which there is evidence of a junction, where the current exon is a source, and the pointed-to exon is the destination. The data structure of the candidate corresponds to a directed exon evidence graph (shown in Figure 41 on page 412).

In a junction evidence graph, nodes are exons, and each directed edge corresponds to junction evidence from a source exon to a destination exon. Each edge has two values:

- Unique SASR evidence
- Unique PR evidence
Referring to Figure 41, the black arrows correspond to normal junctions, blue arrows correspond to alternative splice junctions, and red arrows correspond to fusions.

Most junctions found are regular junctions, meaning junctions between exons of the same gene in the RefSeq expected order. However, more interesting special junctions are also detected during this evaluation step. Alternatively, spliced junctions are defined as multiple junctions from a given exon to two other exons of the same gene within the given sample. Fusion junctions are defined as junctions between exons of different genes.

**Call junctions and fusions with single read only**

LifeScope™ Software includes user-defined parameters that allow the software to call junctions and fusions with fragment (F3 only) datasets. However, LifeScope™ Software’s fusion caller is designed to work with paired-end reads. Calling fusions with only fragment reads likely results in a large number of false positives. We observed that on the same dataset, LifeScope™ Software calls ten times or more fusions using only single-read evidence compared with calls using both paired-end and single-read evidence. We suggest rigorous post-filtering and sorting by total unique single-read evidence for validating those fusion calls.

For detecting known same-gene junctions we observed that using only single-reads calls 80% of the junctions called by paired-end reads (see Figure 42). However, the remaining 20% usually contains lowly-expressed or harder to detect junctions, and we observed a need to double the total number of reads in order to achieve the same sensitivity. As a result, detection of exon junctions with fragment reads is not recommended due to lower specificity and sensitivity.

**Figure 42** compares single-read (SR) vs. paired-end (SR+PE) junction calling sensitivity. The left two bars show SR and SR+PE number of junction calls when using UHR barcode 1. The middle two bars are with UHR barcode 2. The right two bars show the increase in number of calls when the two barcodes are merged and calls are repeated with effectively double the amount of reads from the same library. For both SR and SR+PE, two unique evidences are required to call a junction. For SR+PE, at least one SR and at least one PE evidence is required as well.

![Image](chart.png)

**Figure 42** Comparison of single-read [SR] vs. paired-end [SR+PE] junction calling sensitivity with two barcodes
Output files

The following types of output are generated, depending on the parameter `wt.splice.finder.output.format`:

- Tabular (a tab-delimited junction database) files with the following contents:
  - All detected junctions
  - Junctions interpreted as arising from alternate gene splicing
  - Junctions interpreted as arising from inter-gene fusion
- BED
- SEQ
- Circos
- Statistics files, which contain the following parts:
  - Paired-end splice finder statistics
  - SASR splice finder statistics
  - Evidence matrix

These files are described in the remainder of this section.

WT fragment differences

Exon counts and Junction confidence values (JCV) are not calculated for WT fragment analyses. In the output of the WT fragment splice_finder module, exon counts are reported as 0, and JCV as 100, and those columns should be ignored. For WT fragment analyses, exon counts can be found in the WT counts module output.

Tabular junction files

Three output files are generated with the `.tab` extension for regular junctions, alternatively-spliced junctions, and fusion junctions. Table 121 describes the fields for these files.

<table>
<thead>
<tr>
<th>Field name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exon-1</td>
<td>The gene id followed by the exon order on the gene.</td>
</tr>
<tr>
<td>Exon-1-reference</td>
<td>The reference FASTA file.</td>
</tr>
<tr>
<td>Exon-1-strand</td>
<td>± strand.</td>
</tr>
<tr>
<td>Exon-1-start</td>
<td>The start position.</td>
</tr>
<tr>
<td>Exon-1-end</td>
<td>The end position.</td>
</tr>
<tr>
<td>Exon-2</td>
<td>The gene id followed by the exon order on the gene.</td>
</tr>
<tr>
<td>Exon-2-reference</td>
<td>The reference FASTA file.</td>
</tr>
<tr>
<td>Exon-2-strand</td>
<td>± strand.</td>
</tr>
<tr>
<td>Exon-2-start</td>
<td>The start position.</td>
</tr>
<tr>
<td>Exon-2-end</td>
<td>The end position.</td>
</tr>
<tr>
<td>Exon-1-size</td>
<td>The size of the first exon.</td>
</tr>
<tr>
<td>Exon-2-size</td>
<td>The size of the second exon.</td>
</tr>
<tr>
<td>Exon-1-readcount</td>
<td>The number of reads that map on the first exon.</td>
</tr>
<tr>
<td>Exon-2-readcount</td>
<td>The number of reads that map on the second exon.</td>
</tr>
</tbody>
</table>
The columns that are most relevant to the splice finder are Unique-PR-evidence and Unique-SR-evidence. Also of interest are the two metrics RPKM and JCV.

The purpose of the RPKM (Reads Per Kilobase of exon sequence, per Million reads) metric is to provide a normalized exon expression level. This metric is calculated with the formula:

\[
\text{RPKM} = 10^9 \times \frac{\text{ExonReadCount}}{\text{TotalReadCount} \times \text{ExonLength}}
\]

The RPKM value is also reported by the WT counts module (see “WT counts parameter description” on page 402).

The purpose of the Junction Confidence Value (JCV) metric is to aid in the detection of false positives and in other decisions that depend on a confidence level. Depending on the coverage of the sequencing experiment and exons being tested, the algorithms might generate results that require different user-defined thresholds to detect the most likely fusions. The major contributor of false positives possibly is highly-expressed genes, which have a considerable random chance of encountering a fusion event due to sequencing errors and mapping to homologous exons. A statistical confidence metric (see below) was developed to detect such false positives and assign a confidence level to the evidenced junction.

\[
JCV_{j_{x-y}} = \sum_{i=1}^{n} PQV_i - 10\log_{10} \left( EEM_{j_{x-y}} \right)
\]

**Equation 1. Junction Confidence Value (JCV)**

\[
EEM_{j_{x-y}} = \frac{RC_x}{I_x} \times \frac{RC_y}{I_y} \frac{\mu_T + 3\times \sigma_T}{\mu_T + 3\times \sigma_T}
\]
Equation 2. Error expectation metric (EEM)

where $PQV_i$ is the Phred-scale pairing quality value for the $i$'th unique paired read evidence for a candidate junction $j_{x,y}$ and $x$ and $y$ are the junction exons. For each unique single read evidence, the $PQV_i$ is set to 10. If there are multiple alignments for a given unique start point, take the $PQV$ of the first such alignment. $RC$ is the absolute proper mapped read count for the corresponding exon and is the length of the exon; $\mu_T$ and $\sigma_T$ are the mean and standard deviation of the insert size for the current experiment, $T$.

Error expectation metric (EEM) is used to quantify highly expressed junctions. This metric is hard to calculate due to genome complexity and homology of exons. Our estimation considers the number of reads mapped to the exons, the length of exons, and a conservative insert range.

After the equation is calculated, a JCV that is larger than 100 is set to 100, and those smaller then 0 are set to 0. If a JCV approaches 100, it is more likely to be a real junction.

Junction examples

For a junction detected between exon-1 of size 5,000 and exon-2 of size 400, with mean insert 100 and standard deviation 33.3 bp, assume in case-1, there were three unique junction evidences with PQV 30, 20 and 40 respectively and in case-2 only a single unique evidence with PQV 20. There were 900 properly mapped alignments to exon-1 and 100 such alignments for exon-2. In case-3 has 3 unique evidences similar to case-1, but the exons are assumed to have 100 times more coverage each. The junction confidence value for a junction between these exons would be:

$$30 + 20 + 40 - 10 \times \log_{10} \left( \frac{900}{5000} \times \frac{100}{400} \times 200^2 \right) = 90 - 10 \times \log_{10}(1800) \approx 50 \text{ to } 60$$

Equation 3. Case-1: 3 unique junction evidence between 9x and 12x exons

$$30 + 20 - 10 \times \log_{10} \left( \frac{900}{5000} \times \frac{100}{400} \times 200^2 \right) = 90 - 10 \times \log_{10}(1800) = 0$$

Equation 4. Case-2: 1 unique junction evidence between 9x and 12x exons

$$30 + 20 - 10 \times \log_{10} \left( \frac{90000}{5000} \times \frac{10000}{400} \times 200^2 \right) = 90 - 10 \times \log_{10}(18,000) \approx 10 \text{ to } 20$$

Equation 5. Case-3: 3 unique junction evidence between 900x and 1200x exons
A simplified table of outputs is shown in Table 122.

<table>
<thead>
<tr>
<th>E1</th>
<th>E1-reference</th>
<th>s</th>
<th>E1-start/ E1-end</th>
<th>E2</th>
<th>E2-reference</th>
<th>s</th>
<th>E2-Start/ E2-End</th>
<th>Unique-PR</th>
<th>Unique-SR</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGRN-1</td>
<td>chr1</td>
<td>+</td>
<td>886872/886993</td>
<td>KLHL17-4</td>
<td>chr1</td>
<td>+</td>
<td>887069/887290</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>KLHL17-8</td>
<td>chr1</td>
<td>+</td>
<td>888580/888747</td>
<td>KLHL17-9</td>
<td>chr1</td>
<td>+</td>
<td>889163/889251</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>AGRN-11</td>
<td>chr1</td>
<td>+</td>
<td>969352/969500</td>
<td>AGRN-12</td>
<td>chr1</td>
<td>+</td>
<td>969577/969682</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>AGRN-14</td>
<td>chr1</td>
<td>+</td>
<td>970602/970766</td>
<td>AGRN-15</td>
<td>chr1</td>
<td>+</td>
<td>970976/971119</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>AGRN-17</td>
<td>chr1</td>
<td>+</td>
<td>971403/971508</td>
<td>AGRN-15</td>
<td>chr1</td>
<td>+</td>
<td>971640/971978</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>AGRN-20</td>
<td>chr1</td>
<td>+</td>
<td>972570/972697</td>
<td>AGRN-21</td>
<td>chr1</td>
<td>+</td>
<td>972816/972930</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>AGRN-21</td>
<td>chr1</td>
<td>+</td>
<td>972816/972930</td>
<td>AGRN-22</td>
<td>chr1</td>
<td>+</td>
<td>973019/973138</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>AGRN-26</td>
<td>chr1</td>
<td>+</td>
<td>974809/975038</td>
<td>AGRN-27</td>
<td>chr1</td>
<td>+</td>
<td>975146/975280</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>AGRN-27</td>
<td>chr1</td>
<td>+</td>
<td>975146/975280</td>
<td>AGRN-28</td>
<td>chr1</td>
<td>+</td>
<td>975476/975572</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>PUSL1-4</td>
<td>chr1</td>
<td>+</td>
<td>1234697/1234846</td>
<td>PUSL1-5</td>
<td>chr1</td>
<td>+</td>
<td>1234924/1235094</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>PUSL1-6</td>
<td>chr1</td>
<td>+</td>
<td>1235877/1235931</td>
<td>PUSL1-7</td>
<td>chr1</td>
<td>+</td>
<td>1236152/1236314</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>VWA1-0</td>
<td>chr1</td>
<td>+</td>
<td>1362170/1362727</td>
<td>VWA1-3</td>
<td>chr1</td>
<td>+</td>
<td>1364324/136600</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>MIB2-3</td>
<td>chr1</td>
<td>+</td>
<td>1548632/1548942</td>
<td>MIB2-4</td>
<td>chr1</td>
<td>+</td>
<td>1549017/1549188</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>MIB2-4</td>
<td>chr1</td>
<td>+</td>
<td>1549017/1549188</td>
<td>MIB2-5</td>
<td>chr1</td>
<td>+</td>
<td>1550038/1550144</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>MIB2-6</td>
<td>chr1</td>
<td>+</td>
<td>1550234/1550428</td>
<td>MIB2-7</td>
<td>chr1</td>
<td>+</td>
<td>1550529/1550671</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>MIB2-7</td>
<td>chr1</td>
<td>+</td>
<td>1550529/1550671</td>
<td>MIB2-8</td>
<td>chr1</td>
<td>+</td>
<td>1550789/1550896</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>MIB2-9</td>
<td>chr1</td>
<td>+</td>
<td>1551893/1551997</td>
<td>MIB2-10</td>
<td>chr1</td>
<td>+</td>
<td>1552080/1552242</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>MIB2-10</td>
<td>chr1</td>
<td>+</td>
<td>1552080/1552242</td>
<td>MIB2-11</td>
<td>chr1</td>
<td>+</td>
<td>1552317/1552450</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>MIB2-11</td>
<td>chr1</td>
<td>+</td>
<td>1552317/1552450</td>
<td>MIB2-12</td>
<td>chr1</td>
<td>+</td>
<td>1552539/1552687</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>MIB2-14</td>
<td>chr1</td>
<td>+</td>
<td>1553262/1553422</td>
<td>MIB2-15</td>
<td>chr1</td>
<td>+</td>
<td>1553516/1553642</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>
The BED format was developed to extend the UCSC Genome Browser with user-defined tracks. BED is used to visualize the splice and fusion junctions in the UCSC Genome browser and in the IGV browser (see Figure 43 on page 419). For general documentation about the BED format, including information about all of the BED fields, go to:

genoome.ucsc.edu/FAQ/FAQformat.html

For information about visualization software, see “Pairing information in a BAM file” on page 505.

Each line in the track defines a junction where chromStart is the smaller of the coordinates and chromEnd is the greater.

There are two blocks because a junction typically contains two exons. BlockSizes are the lengths of the exons. The block starts the beginning of the exons. When fusions on different strands or chromosomes, two lines are added to the output, with each line representing one chromosome. Different colors are used to color-code different types:

Figure 43 on page 419 shows the Upstream Hypersensitive Region (UHR) gene region displayed with the Integrative Genomics Viewer (IGV) for positions 3,530,193 to 3,548,355 of Human Chr-1. The following sections describe the tracks in Figure 43.

**WIG (x2) tracks**

The top two tracks show the genomic coverage using the negative strand and positive strand generated by the Bam2Wig tool (Max: 100 coverage).

**BAM track**

The middle track shows the alignments from the BAM file. For display purposes, reads are filtered with MAPQ threshold of 45 (a stringent filter). Bases with quality value 5 to 20 are shaded.

**BED track**

The fifth track shows the junctions detected by the splice finder (BED file). As shown in the figure, all junctions detected are “known” and so are shaded in green.
The optional SEQ output files are in FASTA-format and list regular junctions, alternatively-spliced junctions, and fusion junctions. The entries are 100bp from each end of putative junctions. If any of the exons is less than 100bp in size, the total exon sequence is added.

The Circos-ready format is generated when the `wt.splice.finder.output.format` parameter is set to 4. Gene fusion calls are output in Circos-compatible format, to enable easy visualization with Circos. Follow these steps to use this file with Circos:

1. Copy the folder splice-finder output directory `circus-fusion` and paste it under the Circos installation directory.
2. By default, only the human karyotype file is distributed. If you are working with another species, you must replace the karyotype file.
3. Run this command to start Circos:
   ```
   bin/Circos -conf circus-fusion/circos.conf
   ```

The value of the `wt.splice.finder.output.format` parameter determines which file formats are included in the splice-finder output, as shown in the following list:

- `wt.splice.finder.output.format=1`
  - `<bam_prefix>.alternative_splicing.tab`
  - `<bam_prefix>.fusion.tab`
  - `<bam_prefix>.junctions.tab`
  - `<bam_prefix>.stats`
- `wt.splice.finder.output.format=2`
  - `<bam_prefix>.alternative_splicing.tab`
  - `<bam_prefix>.fusion.tab`
  - `<bam_prefix>.junctions.bed`
Chapter 22  Run a WT Splice Finder Analysis

Output files

- `<bam_prefix>.junctions.tab`
- `<bam_prefix>.stats`

* `wt.splice.finder.output.format=3`
- `<bam_prefix>.alternative_splicing.seq`
- `<bam_prefix>.alternative_splicing.tab`
- `<bam_prefix>.fusion.seq`
- `<bam_prefix>.fusion.tab`
- `<bam_prefix>.junctions.bed`
- `<bam_prefix>.junctions.seq`
- `<bam_prefix>.junctions.tab`
- `<bam_prefix>.stats`

* `wt.splice.finder.output.format=4`
- `<bam_prefix>.alternative_splicing.seq`
- `<bam_prefix>.alternative_splicing.tab`
- `<bam_prefix>.fusion.seq`
- `<bam_prefix>.fusion.tab`
- `<bam_prefix>.junctions.bed`
- `<bam_prefix>.junctions.seq`
- `<bam_prefix>.junctions.tab`
- `<bam_prefix>.stats`
- `circos-fusion/fusion.genes`
- `circos-fusion/fusion.links`
- `circos-fusion/circos.conf`
- `circos-fusion/ideogram.conf`
- `circos-fusion/ticks.conf`
- `circos-fusion/karyotype.human.txt`
PART VII

Small RNA Analyses
Run a Small RNA Mapping Analysis

Overview

The LifeScope™ Software small RNA analysis modules are used to analyze high throughput small RNA sequencing raw data. These modules encapsulate a workflow to map reads to the target genome, to detect small RNAs in the genome, and to determine the expression of the detected small RNAs.

The small RNA mapping module maps small RNA (also known as microRNA or miRNA) reads using the mapping program mapReads. This module maps the small RNA reads to three different references in three steps:

- In the first step, the given set of reads are mapped to filter sequences in order to eliminate the reads generated from irrelevant sources (such as tRNA, adaptor sequences, or others).
- In the second step, the remaining reads from the first step are mapped to the list of known small RNA precursor sequences (miRBase annotations). This list is downloaded from Sanger’s website. This lists for human hg18 and hg19 references are available in the LifeScope™ Software reference repository.
- In the third step, the unmapped reads from the second step are mapped to the genome reference sequence, in order to find novel small RNAs in the sample.
The mapped reads outputs from the second and third steps are merged, and the merged file is the primary output of the small RNA mapping module.

The major components of the small RNA mapping module are:

1. Filter reference mapping
2. miRBase reference mapping
3. Genome reference mapping
4. Merge of mapped reads output (MA files)
5. BAM generation (including ma2BAM, refcor, and sort steps) and BAM merge

Parameters are provided to skip any of the three mapping steps.

Usual read lengths from small RNA sequencing are 35–50 bases, and the length of the miRNA fragments is only 18–28 bp. The reads contain both the miRNA sequence and a P2 adapter sequence at the end. The mapReads phase does an extra step of extension onto adaptor, to correctly remove the adaptor sequence from the read and to report only small RNA sequences in the BAM file.

After the miRBase mapping step, the mapping output file (a MA file) has alignments with respect to the smaller miRBase reference locations. Before merging the MA files, the program converts these miRBase alignments to genomic reference locations, using the input GFF file (described in “Small RNA mapping input files” on page 425).

Example of running the small RNA mapping module

The small.rna standard workflow provides an example of how to run the small RNA mapping module. Here are example LifeScope™ Software shell commands for this workflow, which includes small RNA coverage and small RNA counts:

```
# log into the shell
lscope.sh shell -u username -w password
# cd to the projects repository
cd projects
# make a project, and open it
mk mirna
cd mirna
# make an analysis, and open it
mk run1
cd run1
# define the analysis type
set workflow small.rna
# define the input
add xsq rna.xsq
add xsq rna2.xsq
# specify the reference to be used
set reference hg18
# optionally change parameter defaults after this line
# list the analysis configuration
ls
# run the analysis
run
```
# display the run progress
ls
# exit the command shell
exit

Dummy XSQ file names are used in this example.

In order to change a parameter value in your analysis, use the set param shell command after the line # optionally change parameter defaults after this line in the example commands. For instance, to change the create.unmapped.bam.files parameter, use this shell command:

```
set create.unmapped.bam.files 0 secondary/smallrnamapping.ini
```

You must know which INI file to use for set param command. The ls command, executed in an analysis virtual directory after the workflow has been defined, lists the INI files used in the analysis.

## Small RNA mapping input files

The small RNA mapping module requires the following input:

- One or more XSQ files containing sequencing data.
- A Genome Coordinates GFF file containing precursor sequences with genomic locations. This file is available the following link:
  
  ![http://www.mirbase.org/ftp](http://www.mirbase.org/ftp)

- A single multi-fasta reference file for the filter mapping step.
- A single multi-fasta reference file for the genome mapping step.

The mapping modules accepts as input one or more input XSQ files. The input reads can have different read lengths but they must be of the same library type.

### RNA-Seq reads

The RNA-Seq reads used in the small RNA modules are different from the genomic reads. For RNA-Seq reads:

- Only transcribed sequences are measured by the system.
- Genome coverage is non-uniform due to variation in transcriptional intensity.
- A large subset of reads originate from uninteresting sequences such as ribosomal RNA (rRNA).

### Reads input specification

Plan your analysis input carefully. The way you define your reads input affects the behavior of your analysis. The following factors control how your input data is analyzed:

- **Index (barcode) IDs** – Using an index ID restricts your input to the reads data of one or more indices.
- **Grouping of reads** – Each group of reads is analysed together as one specimen. The output data for a group is combined into one set of results files.
- **Multiple sample runs** – Unrelated reads can be processed together in one run of LifeScope™ Software, while analyzed separately as separate input data.
Stages of mapping

The mapping algorithm involves following stages:

- **Scatter** – The process of distributing reads to the available compute nodes, to achieve parallelization.
- **Mapping** – Each tag is aligned to the reference.
- **BAM file generation** – The process of converting the output of mapping and pairing to BAM format. For color-space data, the base translation step is also included. Due to the scattering step, the BAM files at each compute node are a sub-set of the final BAM file and are referred to as baby BAMs.

BAM generation involves these steps:
  - **Ma2BAM** – A conversion step, from MA format to BAM format, for fragment data only.
  - **Refcor** – Reference-assisted color-to-base translation.
  - **Sort BAM** – Sorting the BAM file from bead-id order to coordinate order.
  - **Merge BAM** – Merging the mini-BAM files created in the sort phase into the final output BAM files.

**Figure 44** shows the phases involved in small RNA mapping.
Chapter 23  Run a Small RNA Mapping Analysis

Stages of mapping

Figure 44  The phases within the small RNA mapping module

Scatter

The scatter stage distributes reads to the available compute nodes, in order to gain the efficiency benefits of parallel computing.

The mapping module can accept multiple read-sets, which are allowed to have different read lengths. The input set of read-sets from multiple XSQ files is first split based on read length. Fragment read-sets are partitioned into separate categories based on these read length categories:

- $0 \leq \text{Length} < 25$
- $25 \leq \text{Length} < 35$
- $35 \leq \text{Length} < 50$
- $50 \leq \text{Length} < 60$
- $60 \leq \text{Length} < 75$
- $75 \leq \text{Length}$
- Trimmed reads, with variable read lengths

The reads from each category are scattered into multiple jobs based on the parameters `fragmap.minreads.per.node` and `fragmap.number.of.nodes`. These parameters are described in the performance section.

Filter map

The reads are mapped to the filter reference in the filter mapping step. Reads that align to the filter reference are skipped from mapping to miRBase and genome references. The filtered reads are populated in separate BAM file inside the `outputs/filtered` directory. A filter reference for use with human reads is available with LifeScope™ Software. This reference contains:

- SOLiD® adapter sequences
- Human ribosomal RNAs (rRNAs)
- Human transfer RNAs (tRNAs)
- Other repetitive human sequences
- Single-base-repeat sequences, including Poly-A and Poly-T

Complete the following steps to construct a filter reference for another species.

2. Append FASTA-formatted species-specific ribosomal, transfer RNA, and other sequences to the filter reference.

• *(Optional)* Add the new filter reference file to the reference repository.
  
  – If you added the assembly for this species to the repository, copy the new filter file to the assembly directory. See “Add new reference files” on page 547. The filter reference is then set automatically when you use the follow shell command: `set reference assembly_name`.
  
  – If you do not have the assembly for this species in the reference repository, use the `analysis.filter.reference` parameter to include the new filter reference in your analysis. For example:

    ```
    set analysis.filter.reference filterfile secondary/global.ini
    ```

    An absolute path is recommended for `filterfile`. A relative path is interpreted from the Linux directory in which the shell was started (when the `lscope.sh` shell command was given). *(The `secondary/global.ini` INI file is the correct file for the `analysis.filter.reference` parameter.)*

**miRBase map**

This phase maps to known miRBase sequences. In a pre-process step miRBase database is converted from GFF to FASTA format to create the miRBase reference. Then the map module is invoked with the unmapped reads from the filter map step and the miRBase reference. The outputs of this step are the miRBase match file (or files) and unmapped reads in XSQ format.

**Genome map**

The unmapped reads from the miRBase map step are mapped to the genome reference in the genome mapping step to produce genome match files.

**miRBase2Genome**

The output of the miRBase mapping step has alignments with respect to the smaller miRBase reference locations. This phase converts alignments from the miRBase coordinate system to the genome coordinate system required with LifeScope™ Software BAM files.

**BAM file generation**

**Ma2BAM**

Ma2BAM is a format converter that changes match files to a pro-BAM format. These pro-BAM files do not contain sequence or quality information but contain relevant BAM meta data. The pro-BAM files are sorted by incoming bead-id order. Relevant meta data from XSQ files is captured in pro-BAM files at this stage. The parameters `smallRNA.bam.primary.align.only`, `smallrna.bam.mqv.threshold`, and `smallrna.create.unmapped.bam.files` control the functionality of the ma2BAM utility. The pro-BAM files are generated on the scratch folder.

Ma2BAM takes output of mapping (MA files) and converts them to pro-BAM format. During BAM generation, mapping quality values are also computed.

For reads with multiple ungapped alignments, the read with the highest mapping quality value is chosen as the primary alignment for the read, and is reported to the BAM file. In cases where there are multiple alignments with the same quality value, the primary alignment is chosen at random from among the alignments with the same quality value. The details of calculating mapping quality values are described below.
Mapping quality values

Quality values are Phred-scaled quality scores. For any given alignment, an MQV is a measure of the confidence of a read aligning to the particular location, given all possible alignments for the read. Quality values are in the range 0–100:

- **0** – The highest probability of error
- **100** – The lowest probability of error

The following factors increase confidence and increase a read’s quality value:

- Longer alignment
- Fewer possible alignments
- Fewer mismatches within the alignment

These factors reduce confidence and reduce a read’s quality value:

- Shorter alignment
- More possible alignments
- More mismatches within the alignment

In order to be consistent with the Phred quality score (-10*log10[prob(error)]) used widely in literature, the quality is computed as the negative log odds of misaligning the read. The resulting quality values are normalized by the maximum possible value to ensure that the quality values are within the range 0–100.

Refcor

Reference Assisted Color-to-Base Translation (also known as refcor) combines information from color calls, optional ECC extra primer round calls, and the reference sequence, to enhance the quality of base-space reads by detecting and fixing color errors that convert into multiple base-space errors.

Refcor accepts a set of pro-BAM files, which contain alignments information without base or color sequence, and generates a corresponding set of BAM files containing alignments with base-space and color-space calls (on the scratch folder). These BAM files are sorted by bead-ID and contain dummy base calls and QVs, but are otherwise compatible with the BAM specification. The refcor module produces bead-ID-sorted BAM files that have valid calls and QVs.

For each read, the module constructs the primer-transition graph with vertices corresponding to all possible k-mers ($k=4$ for 2BE+4BE, $k=2$ for 2BE) in each read position. Two adjacent vertices are connected with an edge if they overlap in $k-1$ bases. A score corresponding to the $(1-P_e)$, where $P_e$ is the probability of being erroneous, is assigned to each of the vertices. Scores are derived from quality values corresponding to enumerated calls. When making a base call in a position, the current algorithm considers the cumulative probability of all sequences that have called base in that position. The evidence from the reference (base) is assigned to the vertex with $k$-mer ending in corresponding base only if the color transition between current and following reference bases matches the color call in a read. Such assignment reduces the reference over-correction in SNP positions. In order to reduce reference over-correction in positions with color errors, we assign to reference base a very low weight corresponding to QV of 8. The weight can be reduced to QV of 0, completely reducing the reference guiding in the color-to-base translation. This results in a smaller skew between reference to non-reference allele ratio. Increasing the weight results in false positive variant calls reduction. This process restores the phase shift caused by color errors or completely eliminates the errors.
Chapter 23 Run a Small RNA Mapping Analysis

Quality alignments and unique alignments

The parameters bamgen.refcor.softclip, smallRNA.bamgen.refcor.addcs, refcor.reference.weight, and refcor.base.filter.qv control the refcor program. The defaults provided are validated for a wide range of data.

- If the XSQ file contains base-space data only, the XSQ base call and quality value are transferred to BAM file. In this case, no color calls are present in BAM file irrespective of the value of bamgen.refcor.addcs.
- If the XSQ file contains 2BE color-space data only, refcor uses aligned 2BE and reference to do base translation for the aligned part of the read. The base sequence and quality value are generated using a proprietary algorithm. For reads with no alignment, bases are determined by naively translating color to base. For these unaligned reads, the QV for the base at position \( k \) is the minimum color QV for all positions \( \leq k \).
- If the XSQ file contains ECC data, then reference with 2BE and 4BE color are used to generate the new base sequence. The base sequence and quality value are generated using a proprietary algorithm. For reads with no alignment, the original base-space data from ECC is preserved.

Sort BAM

The output of refcor is a set of BAM files in bead-id sorted order. These files are sorted in this step using Picard sort into coordinate sorted order before merge. These BAM files are referred to as mini-BAM files and are stored in the analysis temporary directory.

Merge BAM

This phase accepts a set of mini-BAM files and outputs a set of BAM files. These BAM files are stored in the analysis output directory (which is typically a network-attached storage device).

One BAM file is generated per read-set. For a 96-barcoded dataset in a single XSQ file, either 96 or 97 BAM files are created. The 97th BAM contains reads that have the default barcode index.

Quality alignments and unique alignments

We strongly recommend using mapping quality values, rather than uniqueness, as the standard for recognizing or filtering good alignments. A mapping quality value estimates the probability that the alignment is correct, and takes several indications into account, including the number of alignments or uniqueness of an alignment.

The parameter bamgen.mqv.threshold, new in release 2.5.1, is available to filter alignments written to your BAM output files. In order to limit your BAM files to good quality reads, set bamgen.mqv.threshold to a positive integer such as 8 or higher. This technique ensures that you do not see poor quality reads in your BAM file, and that your tertiary analysis is limited to good quality reads.

How a quality value is derived

The following factors increase confidence and increase an alignment’s quality value:
• Longer alignment
• Fewer possible alignments
• Fewer mismatches within the alignment

These factors reduce confidence and reduce an alignment’s quality value:
• Shorter alignment
• More possible alignments
• More mismatches within the alignment

How a quality value is interpreted

For any given alignment, a quality value is a measure of the confidence of a read aligning to the particular location, given all possible alignments for the read. Quality values are within the range 0–100:
• 0 – The highest probability of error
• 100 – The lowest probability of error

Quality values are Phred-scaled quality scores, logarithmically linked to error probabilities. A mapping QV of 10 means that we calculate there is a 90% chance that the alignment is correct. A mapping QV of 20 means that we calculate there is a 99% chance that the alignment is correct.

Unique alignments

Unique alignments might be confused with high quality alignments, but uniqueness is not by itself a good criteria to judge an alignment. Uniqueness only means that one step of secondary mapping, the aligner, reports one alignment. This information is a part of the total information generated during mapping, but does not contain all of the information known about the alignment. In contrast, the mapping quality value is a numerical representation of all considerations about the alignment.

For example, a read might have multiple alignments. If the quality value for only one of those multiple alignments is high, that probability of that alignment being good is high. That alignment should be used in your analysis, even though it is not unique.

For a unique alignment, its quality value must be examined to determine if the alignment should be taken or filtered out. Its uniqueness by itself is irrelevant to the decision about whether it is a good read.

## Small RNA mapping parameters

LifeScope™ Software sets reference-related parameters for you when you use the set reference command with a recognized assembly name and also have the reference files properly added to the reference repository. Table 123 lists miRBase parameters you must either set manually or cause to be set with the set reference command.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>analysis.mirbase.mature.file</td>
<td>The file containing mature form microRNA database annotations. This file is a tab-delimited text file version of miRBase’s mirRNA.xls (not miRBase’s mirRNA.txt).</td>
</tr>
</tbody>
</table>
Table 123  Small RNA miRBase parameters

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>analysis.mirbase.precursor.file</td>
<td></td>
<td>The file containing precursor microRNA database annotations. Set this parameter to the miRBase GFF file for your organism of choice.</td>
</tr>
</tbody>
</table>

Table 124 lists small RNA mapping parameters.

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>smallrnamapping.run</td>
<td>1</td>
<td>Whether or not to run the small RNA mapping module. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Do not run the small RNA mapping module.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Run the small RNA mapping module during this analysis.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use.</td>
</tr>
<tr>
<td>fragmap.number.of.nodes</td>
<td>4</td>
<td>The number of compute nodes available for this analysis.</td>
</tr>
<tr>
<td>fragmap.max.number.of.jobs</td>
<td>100</td>
<td>The maximum number of jobs allowed for mapping. Allowed values: Integers 24–1000.</td>
</tr>
<tr>
<td>mapping.np.per.node</td>
<td>8</td>
<td>The number of processors per node use for mapping. The reads are divided into the number of chunks specified for this parameter.</td>
</tr>
<tr>
<td>fragmap.minreads.per.node</td>
<td>4000000</td>
<td>If the total number of fragments exceeds this amount, then the analysis is distributed across the available nodes. Allowed values: Integers 1–125000000 (125 Million).</td>
</tr>
<tr>
<td>fragmap.maxreads.per.node</td>
<td>150000000</td>
<td>The maximum number of fragments that can be processed on a single node. Allowed values: Integers 1–150000000 (150 Million).</td>
</tr>
<tr>
<td>wall.time</td>
<td>120</td>
<td>Total time for the process to complete.</td>
</tr>
<tr>
<td>java.heap.space</td>
<td>1500</td>
<td>Dynamic memory requirement, in megabytes. Do not enter a unit with the numeric value.</td>
</tr>
</tbody>
</table>
### Small RNA mapping parameters

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mapping.memory</td>
<td>15gb</td>
<td>The total memory, in gigabytes, that is available for map reads, per compute node. Include the units gb in the setting. This number is set to 15 GB because the minimum hardware requirement for memory is 16 GB. <strong>Note:</strong> Typically, the job scheduler allocates on nodes that have the requested memory available. For mixed hardware, it is suggested that different queues are created based on the memory available. Mapping jobs should be launched on high memory systems. For human references (hg18, hg19) and a 25.2 scheme, 45 GB is recommended for fastest performance. The smallest recommended RAM for human reference is 19 GB.</td>
</tr>
</tbody>
</table>

**Optional mapping parameters**

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
</table>
| smallRNA.filter.mapping     | true          | Whether to turn on or off the filter mapping step. Allowed values:  
  • **false:** Turn off the filter mapping step.  
  • **true:** Turn on the filter mapping step. |
| smallRNA.genome.mapping     | true          | (Optional) Whether to turn on or off the genome mapping step. Allowed values:  
  • **false:** Turn off the genome mapping step.  
  • **true:** Turn on the genome mapping step. |
| mapping.in.base             | false         | Whether or not to include base-space data in the mapping. Allowed values:  
  • **false:** Do not include base-space data in mapping.  
  • **true:** Include base-space data in mapping. |
| smallRNA.mirBase.mapping    | true          | (Optional) Whether or not to turn on the small RNA miRBase mapping step. Allowed values:  
  • **false:** Turn off the small RNA miRBase mapping step.  
  • **true:** Turn on the small RNA miRBase mapping step. |

**Optional BAM file generation parameters**

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
</table>
| bamgen.refcor.addcs          | true          | Whether or not to add the color sequence to BAM records. If the XSQ file does not contain color space, this parameter is ignored, and the resulting BAM file output does not contain color space. See “Refcor” on page 429 for information about refcor. Allowed values:  
  • **false:** Do not add color sequence to the output.  
  • **true:** Add color sequence, if color space is present. |
Mapping speed depends on hardware properties as well as on the scattering logic. Mapping runs are split into multiple jobs for processing efficiency. The following parameters affect how a mapping run is split:

- `fragmap.number.of.nodes` – Specifies the number of jobs that are created, only if a split is necessary. The size of each split job is approximately the total number of reads divided by the number of jobs.

- `fragmap.max.number.of.jobs` – Determines the maximum number of jobs that can be launched per analysis. Setting this parameter to a very large value can cause the scheduler to behave incorrectly.

- `fragmap.minreads.per.node` – Determines the threshold of beads beyond which splitting occurs.

- `fragmap.maxreads.per.node` – Determines the largest number of beads that can be mapped per node. The default is based on the minimum scratch space requirements. If you scratch space is smaller or larger, this number can be made smaller or larger respectively.

- `processors.per.node` – This is the number of cores available for mapping analysis on each node.

- `mapping.memory` – For human mapping and 25.2.0 scheme, there are 3 schema lines. Simultaneous handling of schema lines reduces I/O and therefore improves mapping speed.

Memory requirements for mapping jobs are the following:

### Table 124 Small RNA mapping parameters (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>refcor:reference.weight</code></td>
<td>8</td>
<td>Reference weight. This parameter is used during base translation. In the read reconstruction process, multiple signals are combined to generate the final base call. This parameter adds weight (in terms of Phred score) to the signals which are compatible with reference. Color combinations that result in a variant are considered compatible with reference. Additional weight helps to eliminate base errors caused by color error(s) during base translation. Allowed values: Integers 0–100.</td>
</tr>
<tr>
<td><code>refcor:base.filter.qv</code></td>
<td>10</td>
<td>Bases with a quality value below the value of this parameter provided are replaced with 'N'. Allowed values: Integers 0–255.</td>
</tr>
<tr>
<td><code>bamgen.mqv.threshold</code></td>
<td>0</td>
<td>Provides control over the contents written to the output BAM file depending on the quality value of the alignment. To preserve only high quality alignments, set this value to a positive integer. Allowed values: Integers 0–100.</td>
</tr>
<tr>
<td><code>create.unmapped.bam.files</code></td>
<td>false</td>
<td>If set to TRUE, creates an additional BAM output file containing unmapped reads. Allowed values: <code>false</code>: Do not create the unmapped reads output file. <code>true</code>: Create a BAM output file containing the unmapped reads.</td>
</tr>
</tbody>
</table>
- Reference: ~6 GB
- 1 schema line: ~13 GB
- 1 schema line and reference: ~19 GB
- 2 schema lines and reference: ~32 GB
- 3 schema lines and reference: ~45 GB

Below 19 GB, the reference is split up and I/O increases significantly. Pre-built hash tables are not used for memory less than 24 GB. LifeScope™ Software supports the reuse of hash tables for human genome reference files (hg18 and hg19) and the default mapping schemes 35.2, 25.2, and 20.1. Pre-calculated color-space hash tables are included in the reference directory available with LifeScope™ Software. If system RAM is 24 GB or higher, the hash tables are loaded into memory. Depending on the pipeline, this hash table reuse typically saves 2–3 hours in analysis processing time.

For genomes without pre-calculated hash tables in the reference repository, the mapping module creates hash tables but does not save them for reuse.

In general, set the mapping.memory parameter to be 1 GB less than the total system memory (RAM) available.

Mapping only scales when the number of reads that are mapped together is greater than 100 M. Splitting such that the total number of fragments per node is significantly less than 100 M is not recommended. The splitting can be explained using the following pseudo-code.

```java
if (num.reads <= fragmap.minreads.per.node) { num.jobs = 1}
reads.per.job = num.reads / fragmap.number.of.nodes
if (reads.per.job > fragmap.maxreads.per.node) {
    reads.per.job = fragmap.maxreads.per.node
}
num.jobs = ceiling (num.reads / reads.per.job)
if (num.jobs > fragmap.max.number.of.jobs) {
    throw exception;
}
```

In above pseudo-code, the input read-sets (possibly from multiple XSQ files) have num.reads that fall in one read length category. Num.jobs is the total number of jobs launched for mapping analysis of num.reads beads.

In general, the mapping parameters allow mixing and matching single-anchor and multiple-anchor schemes, and also mixing and matching different seed lengths and different numbers of mismatches. For example, an unmapped scheme of 25.2.0, 35.2.0:15:25, 20.1.0, with a repetitive scheme of 35.2.0:30, 45.2.0, 60.2.0, is supported, but not recommended. The impact on performance can be significant.

**Internal parameters**

Table 125 lists small RNA mapping parameters that we do not recommend changing.

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>smallRNA.adaptor.sequence</td>
<td>CGCCTTGGCGCTACAGCAG</td>
<td>(Optional) The small RNA adaptor sequence. Used by mapreads in the extension step to identify the adaptor in the read.</td>
</tr>
</tbody>
</table>
### Table 125: Small RNA internal mapping parameters (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>smallRNA.filter.mapping.max.hits</td>
<td>10</td>
<td>The maximum number of hits for a read in the filter mapping step. Allowed values: Integers.</td>
</tr>
<tr>
<td>smallRNA.filter.mapping.mismatch.penalty</td>
<td>-2.0</td>
<td>Specifies a negative scoring penalty for mismatch that is used in local alignment mode. Setting this value to a negative number greater than the read lengths has the effect of disallowing mismatches. Allowed values: Floats.</td>
</tr>
<tr>
<td>smallRNA.filter.mapping.scheme</td>
<td>20.1.0</td>
<td>The mapping scheme for the filter mapping step. The values are: the size of the seed, number of mismatches allowed in the seed, and the start position of the seed in the read. See “Mapping schemes” on page 437 for information about mapping scheme notation.</td>
</tr>
<tr>
<td>smallRNA.genome.mapping.max.hits</td>
<td>30 (Optional)</td>
<td>The maximum number of hits allowed for a read in the genome mapping step. Allowed values: Integers &lt;= 0.</td>
</tr>
</tbody>
</table>
| smallRNA.genome.mapping.mismatch.penalty            | -2.0          | Specifies a negative scoring penalty for a mismatch. Used in local alignment mode. Allowed values: Floats <= 0.0.  
If the absolute value of this penalty is set to greater than the read length, effectively no mismatches are allowed on this pass. |
| smallRNA.genome.mapping.scheme                      | 20.1.0 (Optional) | The mapping scheme for the filter mapping step. The values are: the size of the seed, number of mismatches allowed in the seed, and the start position of the seed in the read. See “Mapping schemes” on page 437 for information about mapping scheme notation. |
| smallRNA.mapping.fragment.tag                       | F3            | Small RNA mapping fragment tag name. |
| smallRNA.mapping.schemas.file                       | —             | Location of the schemas file for small RNA mapping. |
| smallRNA.mirBase.mapping.max.hits                   | 30 (Optional) | The maximum number of hits per read in the miRBase mapping step. |
| smallRNA.mirBase.mapping.mismatch.penalty           | -2.0          | Specifies a negative scoring penalty for a mismatch. Used in local alignment mode. Allowed values: Floats <= 0.0.  
If the absolute value of this penalty is set to greater than the read length, effectively no mismatches are allowed on this pass. |
| smallRNA.mirBase.mapping.scheme                     | 18.2.0 (Optional) | Mapping scheme for the miRBase mapping step. The values are: the size of the seed, number of mismatches allowed in the seed, and the start position of the seed in the read. |
| smallRNA.valid.adjacent.mismatch                    | false (Optional) | Indicates how to count adjacent mismatches. Allowed values:  
• false: Treat two valid adjacent mismatches as two.  
• true: Treat two valid adjacent mismatches as one.  
Required if smallRNA.mirBase.mapping is set to true. |
Table 125 Small RNA internal mapping parameters (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>task.scratch.dir.delete</td>
<td>true</td>
<td>Whether to keep the scratch files generated during small RNA. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• false: Do not keep the scratch files.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• true: Keep the scratch files.</td>
</tr>
<tr>
<td>analysis.clean.temp.files</td>
<td>1</td>
<td>Whether or not to keep temporary files generated during small RNA mapping. Allowed values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Do not keep temporary files.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Keep the temporary files.</td>
</tr>
</tbody>
</table>

**BAM file generation parameters**

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bamgen.primary.align.only</td>
<td>true</td>
<td>Whether only primary alignments or all the alignments are reported in the BAM files. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• false: Report all alignments.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• true: Report only primary alignments (both gapped and ungapped).</td>
</tr>
<tr>
<td>bamgen.refcor.softclip</td>
<td>0</td>
<td>Whether to try partial recovering of hard-clipped regions. For cases when the unaligned portion of a read needs to be presented in the BAM record, we use soft clipping in the CIGAR string. If a base-space sequence generated by ECC decoder is available, we use that sequence to represent corresponding part of the read. If the ECC decoded base-space is not available, then we naively translate using 2BE and/or +4BE evidence (if available). The soft clipped part of the read can be permanently clipped if either of these conditions is true:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The ECC base QV is less than 4.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The ECC base and 2+4 naive propagation have less than 50% similarity.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• false: Do not try partial recovery.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• true: Attempt partial recovery.</td>
</tr>
</tbody>
</table>

**Mapping schemes**

The mapping scheme parameters listed in Table 110 on page 385 determine the behavior of the mapping module. The LifeScope™ Software mapping module is based on seed-and-extend mapping, where the initial alignment step locates short matches between a read and the reference sequence. The seed specifies the length of the attempted match. With the seed-and-extend approach, a mapping scheme parameter value such as 25.2.0 is interpreted as follows:

- **First value** – The seed length (25).
- **Second value** – The quantity of allowed mismatches in the seed (2).
- **Third value** – The start site of the seed within the read (0).
Small RNA mapping output files

The small RNA mapping module generates two BAM files containing alignments in coordinate order:

- A filtered BAM file, from the filter mapping step.
- A mapped BAM file with genomic locations. This content is merged output from the miRBase mapping and genomic mapping steps.

For information about the BAM file format, see Appendix C, “File Format Descriptions” on page 493.

The number of BAM files generated depends on the input XSQ files. One BAM file is generated per read-set. So if the input data is an XSQ file containing 96 barcodes, then the mapping module generates at least 96 output BAM files. Beads that are unclassified in any barcode are output into a separate additional BAM file. If the parameter `smallRNA.create.unmapped.bam.files` is set to TRUE, then an additional 96 or 97 output BAM files corresponding to the unmapped reads are also generated.

The filenames for the output BAM files are created using information from the input XSQ file, including: file base name, file id, index name, and index id. The BAM files are named according to the following patterns:

- Non-indexed BAM files: `xsqname-fileID-1.bam`
- Indexed BAM files: `xsqname-fileID-idx_bcIndex-bcID.bam`

The fields in the filenames are:

- `xsqname` – The XSQ file base name, without the .xsq extension.
- `fileID` – The file ID for internal XSQ file tracking.
- `bcIndex` – The barcode index.
• **bcID** – The barcode identifier for internal barcode tracking.

The directory structure for mapping output is as follows:

```
outputs/${analysis.output.dir}/
  <mappingIniFileBasename>/
    <sampleName1>/bamfilename1.bam
    <sampleName2>/bamfilename1.bam
```

The directory name `<mappingIniFileBasename>` is the basename of the mapping INI file. The default is `smallrnamapping`.

The string `<sampleName*>` is determined by the sample description for the particular read-set.

The small RNA mapping output files are used by the BAMStats mapping statistics module and by the small RNA tertiary analysis modules.

## Mapping statistics

Mapping statistics occur after mapping as an optional post-processing step named BAMStats. BAMStats accepts the output of the mapping step and generates statistics files to provide an in-depth understanding of the experimental data and to better detect the presence of anomalies. LifeScope™ Software shell users can display the mapping statistics output data as a chart with a spreadsheet program or other third-party program. A subset of the output from mapping can be visualized in the LifeScope™ Software UI as a series of line and bar charts, if the analysis is run in the projects repository.

**Note:** BAMStats does not accept 1.x BAM files as input. 1.3 BAM files must be converted to 2.5.1 BAMs before being used with BAMStats.

## Mapping statistics parameters

Table 126 lists the parameters which control the BAMStats output.

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Optional parameters</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bamstats.maximum.coverage</td>
<td>10000</td>
<td>Defines the maximum coverage allowed for locations in the reference.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Locations with coverage more than the maximum coverage value are ignored</td>
</tr>
<tr>
<td></td>
<td></td>
<td>during coverage calculations.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allowed values: Integers &gt;= 1.</td>
</tr>
<tr>
<td>bamstats.maximum.isize</td>
<td>100000</td>
<td>The maximum insert size for LMP and PE libraries. Reads with an insert size</td>
</tr>
<tr>
<td></td>
<td></td>
<td>more than the specified value are ignored for Insert Range Report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>calculation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allowed values: Integers 1–100000.</td>
</tr>
</tbody>
</table>
### Table 126  BAMStats parameter description (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bamstats.wig.primary.only</td>
<td>1</td>
<td>Use only primary alignments for coverage in WIG file format. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Do not restrict coverage in WIG file format to only primary alignments.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Restrict coverage in WIG file format to only primary alignments.</td>
</tr>
<tr>
<td>bamstats.wig.binsize</td>
<td>100</td>
<td>The bin size for coverage in WIG file format. Allowed values: Integers 1–100000.</td>
</tr>
<tr>
<td>bamstats.bin.isize</td>
<td>100</td>
<td>The bin size for insert range distribution. Allowed values: Integers 1–100000.</td>
</tr>
<tr>
<td>bamstats.wig.combined.report.both.strands</td>
<td>0</td>
<td>Whether to combine data from both the strands for coverage in WIG format. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Do not combine data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Combine data from both the strands for coverage in WIG format.</td>
</tr>
</tbody>
</table>

### Resource parameters

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wall.time</td>
<td>120</td>
<td>Total time for the process to complete.</td>
</tr>
<tr>
<td>java.heap.space</td>
<td>13000</td>
<td>Dynamic memory requirement, in megabytes. Do not enter a unit with the numeric value.</td>
</tr>
<tr>
<td>number.of.nodes</td>
<td>4</td>
<td>The number of compute nodes available for this analysis.</td>
</tr>
<tr>
<td>memory.request</td>
<td>14gb</td>
<td>Memory request. Include the units gb in the setting.</td>
</tr>
</tbody>
</table>

Table 127 lists BAMStats parameters that we do not recommend changing.

### Table 127  BAMStats internal parameter description

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bamstats.run</td>
<td>1</td>
<td>Whether or not to run the BAMStats module. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Do not run the BAMStats module. BAMStats statistics are not generated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Run the BAMStats module during this analysis. The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use.</td>
</tr>
<tr>
<td>bamstats.group.stats</td>
<td>1</td>
<td>Enable combined statistics covering all read-sets in the group or sample. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Only statistics for BAM files are generated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Also generate combined statistics for the group or sample.</td>
</tr>
<tr>
<td>bamstats.maximum.mismatches</td>
<td>100</td>
<td>The maximum mismatches allowed in the alignments. Any alignment with more then the specified number of mismatches is ignored while generating reports related to number of mismatches. Allowed values: Integers 0–100.</td>
</tr>
</tbody>
</table>
Table 127  BAMStats internal parameter description (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bamstats.maximum.baseqv</td>
<td>100</td>
<td>Max base quality values. Any base with base quality value more than the specified value is ignored while generating reports. Allowed values: Integers 0–100.</td>
</tr>
<tr>
<td>bamstats.maximum.mappingqv</td>
<td>255</td>
<td>Maximum mapping quality value. Any alignment with mapping quality value more than the specified value is ignored. Allowed values: Integers 0–100.</td>
</tr>
<tr>
<td>bamstats.wig.minimum.mappingqv</td>
<td>2</td>
<td>Defines the minimum mapping quality allowed for coverage in WIG format. Any alignment with a mapping quality value less than this value is ignored. Allowed values: Integers 0–100.</td>
</tr>
</tbody>
</table>
| bamstats.enable.probe.position      | 0             | Enable probe and position error reports. Allowed values:  
  * 0: Do not generate probe and position error reports. Off is recommended for small RNA. 
  * 1: Also generate probe and position error reports. |

Summary of mapping statistics output

File formats

For every input BAM file, a set of statistics files are generated. These files are in CHT, CSV, TXT, and WIG formats. Each CHT file corresponds to one displayed chart. A CHT file specifies the type of chart, the displayed range of each axis, and the data points, without using external references.

The CHT file format is an internal file format based on the CSV file format, with addition header information included. CHT header information is the following:

```
# name:
# type: scatter2d | pie | vbar | line
# title:
# xaxisname:
# yaxisname:
# xrange: <min>:<tickinterval>:<max>
# yrange: <min>:<tickinterval>:<max>
XAXISNAME, SERIES1NAME, SERIES2NAME, ...
x1, y1.1, y1.2, ...
x2, y2.1, y2.2, ...
x3, y3.1, y3.2, ...
```

The wiggle format (.wig) is a public format typically used for coverage. Visit their site for more information:

```
hgdownload.cse.ucsc.edu/goldenPath/help/wiggle.html
```

A genome browser such as the Integrative Genomics Viewer (IGV) can be used to visualize the coverage. For information is available from their site:

```
www.broadinstitute.org/igv/
```

For a collection of input BAM files that belong to a sample, a set of cumulative statistics files are generated. The cumulative statistics files are also in CHT, CSV, TXT, and WIG formats. The cumulative statistics can be visualized in the LifeScope™ Software UI.

Table 127  BAMStats internal parameter description (continued)
Chapter 23  Run a Small RNA Mapping Analysis

Mapping statistics

Directory structure

The output of the BAMStats module has the following directory structure:

```
bamstats/
   <sampleName1>/*.cht, *.tbl
   <sampleName1>/<bam1>/*.cht
   <sampleName1>/<bam1>/misc/*.csv, *.txt, *.wig
```

Overview

BAMStats generates a comma-separated summary file (BAMfilename-summary.tbl) that summarizes key mapping quality statistics per input BAM file. The summary report contains a snapshot of statistics in all BAM files present in this sample. This file contains one row for each input BAM file in the sample. The summary file is displayed in the LifeScope™ Software UI.

```
<sampleName1>/BAMfilename-summary.tbl
```

Following directories and reports contain the cumulative statistics from all BAM files that belong to this sample. The Misc folder is not displayed in UI.

```
<sampleName1>/*.cht, *.tab
<sampleName1>/Misc/*.csv, *.wig, *.tab, *.txt
```

Following statistics are generated per BAM file in the mapping directory. These reports are not displayed in the UI.

```
<sampleName1>/<BAMfilename>/*.cht
<sampleName1>/<BAMfilename>/Misc/*.csv, *.tab, *.txt, *.wig
```

In a sample, some of the BAM files possibly represent unhealthy DNA or RNA, causing the cumulative statistics to look poor. The summary tbl file is the unified location for examining the quality of data of all BAM files in the sample.

If the data for a particular BAM file is not as expected, look at the directory-level reports for that BAM file, for details.


**Mapping statistics output files**

This section describes the mapping statistics files generated by the BAMStats module. The output files generated by BAMStats include the name of the BAM file. The file name pattern is:

```
BAMFileName-fileID-barcodeID.StatsReportName.tag.extension
```

This string, without the extension, is referred to by `prefix` in Table 128, the mapping statistics output table. For small RNA reports, `tag` is F3.

A raw numbers file is generated for each chart. See “Raw number files” on page 446.

<table>
<thead>
<tr>
<th>Report</th>
<th>Description, axis information, filename</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alignment Length Distribution</strong></td>
<td>A bar plot giving the distribution of alignment lengths found in various tags used in alignment. The report is separated by tag type to provide visibility into the accuracy of individual tags. Only the primary alignment for each bead is considered in calculating the distribution. Y axis: Frequency X axis: Alignment length (from 0 to the maximum read length) Output file name: <code>prefix.Alignment.Length.Distribution.tag.cht</code></td>
</tr>
<tr>
<td><strong>Base Mismatch Distribution</strong></td>
<td>A bar plot giving the distribution of total number of mismatches found in various tags used in alignment. The bins are summed over all alignment lengths. Only the primary alignment for each bead is considered in calculating the distribution. Y axis: Frequency X axis: Number of mismatches (from 0 to the maximum mismatches allowed) Output file name: <code>prefix.BaseMismatch.Distribution.tag.cht</code></td>
</tr>
<tr>
<td><strong>Distribution of Alignment Length and Number of Mismatches in Tags</strong></td>
<td>A report providing a simultaneous picture of the alignment length and number of mismatches distributions in various tags used in alignment. Only primary alignments for each tag are considered in calculating this distribution. The bins range from 0 to the maximum read length, and from 0 to the maximum number of mismatches allowed. <strong>Note:</strong> This report is located in the Misc folder, and is not displayed in the UI. Y axis: Alignment length (from 0 to the max read length) X axis: Number of mismatches (from 0 to the maximum mismatches allowed) Z axis: Frequency Output file name: <code>prefix.AlignmentLength.Mismatch.tag.csv</code></td>
</tr>
<tr>
<td><strong>Distribution of Mismatches by Position</strong></td>
<td>A bar plot providing a distribution of errors (mismatches to reference) by position within the read. Only the primary alignments for each bead are used to generate this distribution. Mismatch frequency is calculated for each position by dividing number of mismatches for that position by total number of bases for that specific position. The base-space report is generated on every run. Y axis: Mismatch percentage by read position X axis: Position (from 0 to maximum read length) Output file name: <code>prefix.Mismatches.By.Position.BaseSpace.tag.cht, and (only if color-space data is present in the BAM file) prefix.Mismatches.By.Position.ColorSpace.tag.cht</code></td>
</tr>
</tbody>
</table>
### Table 128  Mapping statistics output files (continued)

<table>
<thead>
<tr>
<th>Report</th>
<th>Description, axis information, filename</th>
</tr>
</thead>
</table>
| **Color Mismatch Distribution** | A bar plot providing a distribution of the total number of color mismatches in various tags used in alignment. Only the primary alignments for each bead are used to generate this distribution. This report is only generated if mapping is done in color space.  
Y axis: Frequency  
X axis: Number of mismatches (from 0 to the maximum mismatches allowed)  
| **Distribution of Alignment Length and Color Mismatches** | A report providing a simultaneous picture of the alignment length and number of color mismatches distributions in various tags used in alignment. Only primary alignments for each tag are considered in calculating this distribution. The bins range from 0 to the maximum read length, and from 0 to the maximum number of mismatches allowed. This report is only generated if mapping is done in color space.  
**Note:** This report is located in the **Misc** folder, and is not displayed in the UI.  
Y axis: Alignment length (from 0 to the max read length)  
X axis: Number of mismatches (from 0 to the maximum mismatches allowed)  
Z axis: Frequency  
Output file name: `prefix.AlignmentLength.ColorMismatches.tag.csv` |
| **Base QV Distribution** | A bar plot providing a distribution of base quality values generated using the reference assisted error correction/base conversion algorithm. The base QV distributions are separated for each tag as quality of individual tags could be very different.  
Y axis: Frequency  
X axis: Base QV (from 0 to the maximum QV)  
Output file name: `prefix.BaseQV.cht` |
| **Base QVs by Position** | A bar plot providing a distribution of base quality values by individual base positions. This report identifies if certain base positions (particularly towards the end of the read) have poor base quality values. All the tags (F3/R3/F5-P2) used in the alignment are combined into a single bin.  
Y axis: Base QV  
X axis: Base position (from 0 to read length)  
Z axis: Frequency  
Output file name: `prefix.BaseQV.by.Position.csv` |
| **Distribution of Mismatches by Base QV** | A line plot providing a distribution of errors (mismatches to reference) by base quality values bins. Mismatch frequency is calculated for each base quality value by dividing number of mismatches for that base quality value by total number of bases for that specific base quality value.  
Y axis: Percentage of total mismatches  
X axis: Base QV (from 0 to maximum QV)  
This report gives, for each QV, a measure of the raw mismatches at positions with that QV. This information does not measure whether the base QVs generated are well calibrated to the probability of error in that particular base position.  
Table 128  Mapping statistics output files (continued)

<table>
<thead>
<tr>
<th>Report</th>
<th>Description, axis information, filename</th>
</tr>
</thead>
</table>
| **Distribution by Mapping QVs** | A bar plot providing a distribution of mapping quality values for the F3 tag. Only the primary alignment for each bead is used in calculating this distribution.  
Y axis: Frequency  
X axis: Mapping QV (from 0 to maximum mapping QV)  
Output file name: prefix.MappingQV.tag.cht |
| **Coverage Report** | A line plot providing a distribution of coverage obtained after mapping/pairing. Only the primary alignment for each bead is used in this calculation.  
Y axis: Frequency  
X axis: Coverage (from 0 to number of reads)  
Output file name: prefix.Coverage.tag.cht |
| **Coverage Report by Chromosome [Contig] and Base Windows** | A line plot providing a distribution of coverage within each reference window. The coverage is calculated within each window along a reference chromosome. The window size can theoretically be anywhere from a single base to the contig length. However the calculations of base level coverage are computationally expensive and less interpretable as the window size increases. Only the primary alignment for each bead is used in this calculation.  
Y axis: Frequency  
X axis: Coverage (from 0 to number of reads)  
| **Coverage by Strand** | A line plot providing the distribution of coverage within each reference window separated by reference strand (+/-). Only the primary alignment for each bead is used in this calculation.  
Y axis: Frequency  
X axis: Coverage (from 0 to number of reads)  
| **Coverage files** | Coverage reports in wiggle format. For each genome position, reports the number of reads that cover (map to or span) the position. Because reporting coverage for each position results in very large files, coverage is reported for bins, with each bin spanning a user-defined number of bases. For each bin, the mean coverage of all the positions in that bin is reported. The parameter bamstats.wig.binsize controls the size of the bins in this file.  
Each coverage file includes a header as the first line. The header lines follow this pattern:  
track type=wiggle_0 name=<chrname> description=<coverage from positive/negative/both strand> visibility=full color=0,0,255 fixedStep chrom=<chrname> start=<startpos> step=<binsize> span=<binsize>  
See also "Coverage files" on page 448.  
Output file name: coverage_chrnn.POS.wig, coverage_chrnn.NEG.wig |

**Unique Start Positions**
Table 128  Mapping statistics output files (continued)

<table>
<thead>
<tr>
<th>Report</th>
<th>Description, axis information, filename</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This report lists the following:</td>
</tr>
<tr>
<td></td>
<td>• The total number of unique start positions.</td>
</tr>
<tr>
<td></td>
<td>• The average number of unique start positions per reference position.</td>
</tr>
<tr>
<td></td>
<td>See also &quot;Unique Start Position&quot; on page 448.</td>
</tr>
<tr>
<td></td>
<td>Output file name: BAMfilename-summary.txt, Samplename.-summary.txt</td>
</tr>
</tbody>
</table>

**Summary Report**

The summary file provides statistics for each BAM file in the sample. See "Summary file".

Output file name: filename.-summary.tbl

**Raw number files**

For each CHT and CSV file listed in Table 128, a corresponding TAB file contains the raw numbers and percentages of the distributions, and a total. The following are example contents of the prefix.Coverage.cht and the corresponding prefix.Coverage.tab file with raw number information:

**prefix.Coverage.cht:**

```
# name: Coverage Report (N in reference not counted as covered)
# type: line
# title: Coverage Report (N in reference not counted as covered)
# xaxisname: Coverage
# yaxisname: Percent
# xrange: Coverage,Percent
# yrange: 0:10:100
Coverage,Percent
0,95.6421
1,2.2274
2,0.1928
3,0.0570
4,0.0323
5,0.0256
6,0.0236
7,0.0234
8,0.0238
9,0.0247
```

**prefix.Coverage.tab, with raw numbers in the Count column:**

```
# name: Coverage Report (N in reference not counted as covered)
# title: Coverage Report (N in reference not counted as covered)
# total: 3095693195
Coverage Count Percent
0 2960786653  95.6421
1  68954650  2.2274
2  5969697  0.1928
3  1764106  0.0570
4  1000572  0.0323
5  790979  0.0256
6  731282  0.0236
7  723622  0.0234
```
This section provides example output of some mapping statistics output files.

**Summary file**

The summary file provides statistics for each BAM file in the sample. This list describes labels used in the summary file:

- **NumFragmentsTotal** – The total number of beads, before any filtering on the instrument. This value is also the fragment count in the input XSQ file.
- **NumFragmentsPassingFilters** – The number of beads that pass filtering on the instrument.
- **NumMapped** – The number of reads with primary alignment.
- **% filtered mapped** – The number of primary reads divided by the number of beads passing instrument filtering \((\text{NumMapped} / \text{NumFragmentsPassingFilters})\).
- **% total mapped** – The number of primary reads divided by the number of total beads \((\text{NumMapped} / \text{NumFragmentsTotal})\).
- **Megabases of coverage** – The sum of alignment lengths, for primary alignments only.

The number of beads filtered out in the instrument is found by subtracting the number of beads that pass filtering from the total number of beads:

\[
\text{NumFragmentsTotal} - \text{NumFragmentsPassingFilters}
\]

The following is example output of a summary file. Each column of data corresponds to a BAM file.

```
#title: BAMStats Summary
BamFileName           solid0054_20100203_chr6P.LTcolor-1-1.bam
                      solid0107_20100514_chr6PLTcolor-3-1.bam
                      solid0107_20100408_chr6P.LTcolor-2-1.bam
IsColorInBam         Y        Y        Y
IsBaseInXSQ          N        N        N
LibraryType          FRAG     FRAG     FRAG
ReadLength           50x25     50x25     50x25
Megabases of Coverage 517151581 504214985 555762980
NumFragmentsPassingFilters 8916035   8280061   9519990
NumFragmentsTotal    8916035   8280061   9519990
Tag1-NumMapped       8021248   7653534   8601958
Tag1- % total Mapped 89.96     92.43     90.36
Tag1- % filtered mapped 89.96     92.43     90.36
Tag1-AlignmentLength F3       F3       F3
  Min      25     25     25
  Max      50     50     50
  Avg  46.12   46.30   46.83
  Median  50     50     50
  StdDev  7.46    6.91    6.65
```
Chapter 23  Run a Small RNA Mapping Analysis

Mapping statistics

Unique Start Position

The following is an example of a Unique Start Position report, for fragment data:

Unique Start Positions F3
Total Number of Unique Start Positions: 16536 (16.043154 % of primary alignments)
Average Number of Unique Start Positions per Reference Position: 0.001630

Unique Start Position reports have the following naming patterns:
- BAMfilename-summary.txt
- Samplename-summary.txt

Coverage files

Example contents for the file coverage_chr1_positive.wig are:

```
browser position chr1:1-200000000
browser hide all
browser pack refGene encodeRegions
# minimumMapq=25, minimumCoverage=1, alignmentFilteringMode=PRIMARY, filterOrphanedMates=false,
track name="BAM Coverage positive strand" description="BAM Coverage positive strand" visibility="full color 0,0,255"
priority=10 yLineMark=0 type=wiggle_0 yLineOnOff=on
variableStep chrom=chr1 span=1
33617 1
33618 1
33619 1
33620 1
33621 1
33622 1
33623 1
33624 1
33625 1
33626 1
33627 1
...
```

Example contents for the file coverage_chr1_negative.wig are:

```
browser position chr1:1-200000000
browser hide all
browser pack refGene encodeRegions
# minimumMapq=25, minimumCoverage=1, alignmentFilteringMode=PRIMARY, filterOrphanedMates=false,
```
This section describes requirements to generate mapping statistics outside of the context of the mapping module. These requirements are:

- **Directory structure** – The input directory for BAMStats module must mimic the output directory structure of mapping:
  ```
  Input.BAMStat.dir/
  <sampleName1>/*.bam, *.bai
  <sampleName2>/*.bam, *.bai
  ```

- **Index files** – For every BAM file, there must be a corresponding BAI-formatted indexed file. BAI files can be generated using the following command:
  ```
  samtools index <bamfile>
  ```
  This command generates index sorted alignment for fast random access. The index file <bamfile>.bai is created. For details, please refer to the samtools site:
  ```
  http://samtools.sourceforge.net/samtools.shtml
  ```

- **Write permission** – The user must also have write permission for the input directory, because the BAMStats module generates position and probe error files in that directory.

**FAQ - Small RNA mapping**

1

**How does the local alignment approach affect mapping?**

Using the local alignment approach removes the constraint of a whole-read alignment, and mapping rate is significantly increased. It is not uncommon for a poor dataset with a 30% mapping rate at 50_6, to reach more than a 60% mapping rate with the mapping algorithm used in LifeScope™ Software.
While the majority of alignments are full length, some alignments can vary in length. If a read has many errors towards the end, the final alignment will not include these positions if a shorter alignment receives a better overall score.

2

How do I increase mapping rate?

Mapping rate increases with each additional round of mapping. However, the gain in rate comes at the cost of increased runtime and disk requirement. Testing has found sets of mapping schemes that strike a good balance between mapping rate and runtime for 25-, 35-, and 50-bp reads. These schemes are shipped as LifeScope™ Software defaults and should work well for most purposes.

For 50-bp reads, two mapping parameters keys define how many rounds of mapping are run, and what happens in each round. The default values of the keys are shown below, and their meanings are further explained in Table 9 on page 76.

Note: Repetitive schemes are empty by default.

- mapping.scheme.unmapped.50 = 25.2.0,25.2.15
- mapping.scheme.repetitive.50 = 38.3.0,25.2.0

The repetitive scheme does not affect the number of mapped reads, and setting it only increases the chance of finding better alignments for repetitive reads. If your main goal is to improve throughput, first try to add more iterations at the unmapped scheme. For example, add another step of 25.2.25 at the end of the unmapped scheme above.

3

What is a good seed for my application?

Consider the following factors when picking seed parameters:

- Mapping is slower when more mismatches are allowed in the seed. However, allowing more mismatches improves the mapping rate.
- Shorter seeds have higher mapping rates, but also lead to more spurious alignments.
- Color-calls at the beginning of a read are more reliable than those at the end. Therefore, you have the option to anchor the seed near the front of the read. However, applications such as transcriptome sequencing are scenarios where it is an advantage to anchor the seed near the end of the read.

For 50-bp reads, the following default settings are recommended:

- 18.2.0 for Sanger alignment
- 20.1.0 for genome alignment

4

What reads scenario achieves the best accuracy?
The best performance is achieved by considering 2BE and 4BE reads, their alignment, and reference. Typically, base-space reads generated by ECC have lower mapping throughput. Base-space reads generated from combined 2BE and 4BE calls and no reference have lower accuracy, compared to ECC calls.

5

How do I interpret a mapping quality value?

First of all, it is worth noting that the mapping quality value (QV) is not like the p-value in BLAST. In BLAST, the p-value is used to estimate the likelihood that the aligned sequences are related. In general, reads come from the same source and it is known that the two are related. The purpose of mapping QV is to estimate the probability that the read originates from the mapped genomic location. The estimate is determined mainly by the difference in significance between the best- and second-best hits.

The numeric interpretation of mapping QV is the same as the base call QV. A mapping QV of 10 means that there is a 90% chance that the alignment is correct. A mapping QV of 20 means that there is a 99% chance that the alignment is correct.

6

How much RAM do I need to analyze human samples?

For optimal performance, 24 GB of RAM per cluster node is recommended for human samples. If the cluster node has less than 24 GB of available RAM, mapreads can split the genome into smaller segments, and align to each segment sequentially. Splitting the genome into smaller segments is implemented inside mapreads and is completely transparent to the user.

Mapping with less than the recommended amount of RAM slows down performance. If the cluster node has 16 GB of RAM, mapping human samples will be roughly 30% slower compared to a machine with 24 GB of RAM.

7

In the mapping statistics genome coverage calculation report, how are Ns in the reference counted?

Ns in the reference are counted as missing coverage. For example, in the human genome, about 7% of the reference sequence consists of Ns. This means that the coverage calculation will never be reported as higher than 93%. If the frequency distribution says 7.49% of the genome is uncovered, this means that about 0.34% of the non-N reference sequence does not have reads mapped to it, and that also the 7.15% of the genome that consists of N does not have reads mapped to it.
What should I do to ensure balanced allele ratios at heterozygous positions?

The base-translation algorithm, refcor, uses instrument color-space data (ECC or non-ECC) and the reference sequence to produce the most likely base calls, along with a Phred-scale quality value. The degree to which the reference influences the base call is controlled by the refcor.reference.weight parameter. A position is converted to reference only if the refcor.reference.weight setting is higher than the difference between the quality values of a correct color call and of an adjacent erroneous color call that supports reference.

The refcor.reference.weight default setting, 8, is chosen to be low enough to ensure balanced allele ratios, and also be large enough both to distinguish correct from incorrect color calls and to maximize base accuracy.

In the event an incorrect reference base call is made, the disagreement between evidence sources causes that call’s QV to be low. Incorrect calls can be filtered out based on QV threshold, and this filtering improves allele ratios.

Another way of filtering low quality base-calls is to adjust the parameter refcor.base.filter.qv, that converts all base calls with a quality value less than refcor.base.filter.qv into Ns. This filtering has the effect of setting QV to zero in the calls with low confidence. To decrease the number of N base calls, reduce this threshold (for example, from the default of 10 to 5).

Note: Setting the refcor.reference.weight parameter to zero completely eliminates the reference as a guide for base-translation (completely eliminates reference bias), but also considerably reduces the quality of base-translation.

Can I run BAMStats (mapping statistics) on filtered BAM files?

Yes. You can run BAMStats on the filtered BAM files using the command shell. This scenario is not supported in the LifeScope™ Software UI.

These instructions describe creating your own version of the small RNA workflow, and modifying your version to include BAMStats on the filtered BAM file.

1. Follow the instructions in “Create a new workflow” on page 113 to create a workflow based on the small.rna standard workflow.

2. Create a new INI file named filtermapping.bamstats.ini, with these contents:

   import global.ini

   ###Run parameter

   bamstats.run=1

   ###Bin size for insert range distribution.
   #bamstats.bin.isize=100
## The input directory for BamStats. There should be one directory per sample containing the BAM files for that sample.

```sh
bamstats.input.dir=${analysis.output.dir}/bam/filtered
```

## Maximum coverage of a location in the reference. Locations with coverage more than the maximum coverage values will be ignored during coverage calculations.

```
#bamstats.maximum.coverage=10000
```

## Maximum insert size for LMP and PE libraries. Reads with insert size more than the specified value will be ignored for Insert Range Report calculation.

```
#bamstats.maximum.isize=100000
```

## The path to the output directory where BamStats will write its .cht files.

```
#bamstats.output.dir= ${task.output.dir}
```

## Bin size for Coverage in WIG file format

```
#bamstats.wig.binsize=100
```

## Whether to combine data from both the strands for coverage in WIG format

```
#bamstats.wig.combined.report.both.strands=0
```

## Use only primary alignments for Coverage in WIG file format.

```
#bamstats.wig.primary.only=1
```

analysis.genome.reference = ${analysis.filter.reference}

10. **Edit the file secondary/secondary.pln in your workflow to add the filtermapping.bamstats step.** The contents of the revised PLN file are:

```sh
smallrnamapping.ini
bamstats.ini < smallrnamapping.ini
```

This PLN file runs the BAMStats module on filtered BAM files also as part of workflow.

---

**What is the difference between the small RNA alignment method and the other LifeScope™ Software alignment methods?**

Genome alignment algorithm is modified to accommodate reads particularities of small RNA libraries. The main difference between reads generated from DNA or RNA libraries and small RNA libraries is that only the beginning of the reads is represented in reference genome (mature forms of miRNA with variable length), followed by the P2 adaptor sequence.

Alignment (against Sanger or genome reference) comprises two steps:

1. **Global alignment** – Read seed alignment.
Global alignment is identical to the one used in all other LifeScope™ Software alignment methods (that is, genomic resequencing and whole transcriptome mapping).

2. **Local alignment** – Search for adaptor sequence start position (within read) using seed alignment location.

   During local alignment step, we use the following to estimate the adaptor start position:
   - For every possible adaptor start position \( n \), we compare the actual read with the hypothetical one obtained by concatenating \( n-1 \) bases extracted from the genomic location of the read seed and read length \( -n \) bases of the adaptor sequence.
   - All adaptor start locations are tested. The adaptor start location giving the smallest number of mismatches is considered to be the beginning of the adaptor.
   - The number of mismatches for the full length of the read is used to generate the alignment confidence metric (the mapping QV).

---

**FAQ – BAMStats**

**11 Are mismatches in the adapter counted in the small RNA module?**

In the small RNA module, the number of mismatches reported in the alignment include the number of mismatches in the adapter. When the BAM file is generated, the adaptor is trimmed from the alignment, but the count of mismatches is not adjusted.

**FAQ – BAMStats**

**1 In the BAMStats summary table, why do the NumFragmentsPassingFilters and NumFragmentsTotal fields have the same totals?**

NumFragmentsTotal reports the total number of fragments (read pairs or reads) passed from the instrument. NumFragmentsPassingFilters reports the number of fragments that pass the instrument filter. For the 5500 instrument, all reads pass filters when using color space, and so these two numbers are the same. If the run is performed with ECC, then the base space reads have filters applied to remove those of lowest quality. For an ECC run mapped in base space using LifeScope™ Software, the filtered reads are reported in LifeScope™ Software, and the Total and Filtered numbers are different.

**2 What changes are required for a job with a large number of contigs?**
To run BAMStats with a large number of contigs (around 500 or more) and a very high `bamstats.maximum.coverage` value (around 300,000 or more), you can increase the heap space or decrease the maximum coverage value, or do both:

- **Decrease the `bamstats.maximum.coverage` value.**
  
  You could try 100000 or 10000 for the maximum coverage value.

- **Increase heap space by increasing the `memory.request` parameter for the mapping module, and increase the `java.heap.space` parameter in the `bamstats.properties` file.**

  **Note:** The heap space and memory request parameters directly affect job scheduling and impact run times. The values to use also depend on your data sets and compute environment.

(It is also possible, but not recommended in most cases, to change the system property `max.head.node.java.heap.space` from the default of 64 to 128 or 256. This parameter is in the `<installdir>/etc/analysis/system.properties` file. Because changes to system properties affect all runs of all users, you could consider changing the parameter back to its original value after the large contig run completes. Setting `max.head.node.java.heap.space` to 128 or 256 could present a problem if your head node is extremely loaded.)
Run a Small RNA Coverage Analysis

This chapter covers:

- Overview ............................................................. 457
- Example of running the small RNA coverage module ............ 457
- Small RNA coverage input files .................................... 458
- Small RNA coverage parameters ................................... 458
- Small RNA coverage output files .................................. 459

Overview

The small RNA coverage module calculates read coverage per position.

Example of running the small RNA coverage module

The small.rna standard workflow provides an example of how to run the small RNA coverage module. Here are example LifeScope™ Software shell commands for this workflow, which includes small RNA mapping and small RNA counts:

```
lscope.sh shell -u username -w password
cd projects
mk mirna
cd mirna
mk run1
cd run1
set workflow small.rna
add xsq rna.xsq
set reference hg18
# optionally change parameter defaults here
run
ls
exit
```

In order to change a parameter value in your analysis, use the `set param shell` command after the line `# optionally change parameter defaults here` in the example commands. For instance, to change the `RNA.coverage.min.value` parameter, use this shell command:

```
set RNA.coverage.min.value 0  secondary/srCoverage.ini
```
Small RNA coverage input files

The small RNA coverage module takes as input one or more BAM files containing mapped data. The following describe the alignment input accepted. This module:

- Accepts only fragment data for RNA libraries.
- Accepts multiple BAM files as input.
- Accepts BAM files with different read lengths.
- Accepts both color space and base-space files, and a combination of color and base-space data.

Small RNA coverage parameters

Table 129 lists the small RNA coverage parameters.

Table 129  Small RNA coverage parameter description

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Optional parameters</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RNA.coverage.min.quality</td>
<td>2</td>
<td>The mapping quality value threshold for selecting the alignments. Alignments with a mapping quality value lower than this threshold are not written to output. Allowed values: Integers 0–100.</td>
</tr>
<tr>
<td>RNA.coverage.min.value</td>
<td>0</td>
<td>The mapping quality threshold value for selecting alignments from the input BAM. In order to be included in the output, a position’s coverage must be greater than or equal to this value. Allowed values: Integers &gt;= 0.</td>
</tr>
</tbody>
</table>
| RNA.coverage.per.chromosome     | true          | Whether coverage output is generated as one file per every chromosome per strand, or as a single file with coverage of all chromosomes per strand. Allowed values: 
  - false: Generate a single file with coverage of all chromosomes per strand.
  - true: Generate one file per every chromosome per strand. |
| RNA.coverage.primary.only       | true          | Whether only primary alignments or all the alignments from the input BAM file are used as input. Allowed values: 
  - false: Consider all alignments.
  - true: Consider only primary alignments (both gapped and ungapped). |
| **Resource parameters**         |               |             |
| memory.request                  | 4gb           | Memory request. Include the units gb in the setting. |
| java.heap.space                 | 4000          | Dynamic memory requirement, in megabytes. Do not enter a unit with the numeric value. |
Table 130 lists the small RNA coverage parameters that we do not recommend changing.

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>coverage.run</td>
<td>1</td>
<td>Whether or not to run the small RNA coverage module. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Do not run the coverage module.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Run the coverage module during this analysis.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use.</td>
</tr>
<tr>
<td>analysis.input.bams</td>
<td>—</td>
<td>A comma-separated list of paths to the input BAM files. Relative and absolute paths are accepted. Generated automatically by LifeScope™ Software during a workflow.</td>
</tr>
</tbody>
</table>

Small RNA coverage output files

The small RNA coverage module's output files are in the wiggle (*.wig) file format. Wiggle files are visualized in genomics browsers such as the Integrative Genomics Viewer (IGV). You can download the IGV browser from the Broad Institute website:

www.broadinstitute.org/igv

The wiggle file specification is available from this site:

http://genome.ucsc.edu/goldenPath/help/wiggle.html

If the parameter RNA.coverage.per.chromosome is set to false, the small RNA coverage module generates two output files with coverage of all chromosomes per strand:

- coverage_positive.wig
- coverage_negative.wig

If the parameter RNA.coverage.per.chromosome is set to true, the small RNA coverage module generates one coverage file per chromosome per strand:

- coverage_chr**_positive.wig
- coverage_chr**_negative.wig
Run a Small RNA Counts Analysis

This chapter covers:

- Overview ................................................................. 461
- Example of running the small RNA count module. .................... 461
- Small RNA counts ....................................................... 462
- Small RNA counts input files ........................................ 462
- Small RNA counts parameters ....................................... 464
- Small RNA counts output files ...................................... 465

Overview

This module generates tag counts for precursor and mature miRNA sequences. Run this analysis module to generate miRNA expression profiles.

IMPORTANT! Alignments must come from the same strand as the feature to contribute to the count. A non-gapped tag contributes to a feature’s count if it overlaps the feature and has no more than three bases outside the feature. A gapped tag contributes to a feature’s count if one of its match regions terminates at a feature boundary.

Example of running the small RNA count module

The small.rna standard workflow provides an example of how to run the small RNA counts module. Here are example LifeScope™ Software shell commands for this workflow, which includes small RNA mapping and small RNA coverage:

```bash
# log into the shell
lscope.sh shell -u username -w password
# cd to the projects repository
cd /projects
# create a project and open it
mk mirna
cd mirna
# create an analysis and open it
```
mk run1
cd run1
# define the analysis type
set workflow small.rna
# define the input data
add xsq rna.xsq
add xsq rna2.xsq
# specify the reference
set reference hg18
# optionally change parameter defaults here
# list the analysis configuration
ls
# start the analysis
run
# list progress information about the run
ls
# exit the command shell
exit

Dummy XSQ file name are given in the example. In order to change a parameter value in your analysis, use the \texttt{set param} shell command after the line \# optionally change parameter defaults here in the example commands. For instance, to change the parameter \texttt{SmallRNA.counts.min.quality}, use this shell command:

\begin{verbatim}
set SmallRNA.counts.min.quality 0 secondary/srCount.ini
\end{verbatim}

You must know which INI file to use for the \texttt{set param} command. The \texttt{ls} command, executed in an analysis virtual directory after the workflow has been defined, lists the INI files used in the analysis.

\textbf{Note:} The shell command \texttt{set workflow hg18} (or \texttt{hg19}) incorrectly sets the \texttt{analysis.mirbase.mature.file parameter} to \texttt{README.txt}. You must manually set the parameter correctly after the \texttt{set workflow} command.

### Small RNA counts

The LifeScope™ Software small RNA counts module calculates counts per precursor or mature sequence, where counts are the number of reads mapped. This module is similar to the WT counts module, which computes counts per exon.

This module requires an annotation features file, which contains either precursor sequences information in a GFF file, or mature sequences information in a text file. The counts module uses the annotation features file and alignments in the form of one or more BAM files, and computes the number of alignments that match every feature. Non-primary and low quality alignments can be filtered by setting parameters appropriately.

### Small RNA counts input files

The small RNA mapping module generates two BAM files containing alignments in coordinate order. These two BAM files are input to the small RNA counts module for separate counts processing:
The filtered BAM file, from the filter mapping step. The small RNA counts module uses this file to provide counts for the number of reads mapped to the sequences in the filter reference. See “Filtered BAM file parameters” on page 465 and “Filtered counts output” on page 467.

One or more mapped BAM files with genomic locations. The module uses these files as input with the mapped BAM files:

- A Genome Coordinates GFF file containing precursor sequences with genomic locations.
- (Optional) A file containing mature form sequences with locations relative to precursor sequences. If the mature sequences file is not provided as input, the module computes counts per precursor sequence.

These files are described in the following sections.

### Alignments files

The following describe the alignment input accepted by the small RNA counts module. This module:

- Accepts only fragment data for RNA libraries.
- Accepts multiple BAM files as input.
- Accepts BAM files with different read lengths.
- Accepts both color space and base space files, and a combination of color-space and base-space data.

### Precursor sequences

This file is a Genome Coordinates GFF file containing precursor sequences with genomic locations. You must use the same file that was used during the small RNA mapping step.

This file is included in the LifeScope™ Software reference repository and is also available the following link:

http://www.mirbase.org/ftp

### Mature form sequences

The mature form sequences file provides locations relative to precursor sequences. This tab-separated file is included in the LifeScope™ Software reference repository and is also available the following link:

ftp://mirbase.org/pub/mirbase/CURRENT/miRNA.xls

If you download a new file, you must save the file locally in tab-delimited format. Only this specific mature forms file is supported.

If the mature sequences file is not provided as input, the module computes counts per precursor sequence.
**Small RNA counts parameters**

**Mapped file parameters**

Table 131 lists the small RNA counts parameters used to process the mapped BAM file.

### Table 131 Small RNA counts parameter description

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SmallRNA.counts.min.quality</td>
<td>2</td>
<td>The mapping quality value threshold for selecting the alignments. Alignments with a mapping quality value lower than this threshold are not written to output. Allowed values: Integers 0–100.</td>
</tr>
<tr>
<td>SmallRNA.counts.overflow.limit</td>
<td>3</td>
<td>The maximum allowed offset on either side of an alignment or feature, when looking for overlap between an alignment and a feature. If this limit is set to 0, then the start and end coordinates of the alignment record and feature must match exactly. Allowed values: Integers 0–20.</td>
</tr>
</tbody>
</table>
| SmallRNA.counts.per.feature           | true          | Controls the aggregated count output. Allowed values:  
  - false: The aggregated count of all the reads mapped to a feature are output on the same line.  
  - true: Reads of different starts and ends are counted separately for every feature and are output on separate lines.  
  The primary key in the output GTF file is: \{Feature, start, end\}. |
| SmallRNA.counts.primary.only          | true          | Whether only primary alignments or all the alignments from the input BAM file are used as input. Allowed values:  
  - false: Consider all alignments.  
  - true: Consider only primary alignments (both gapped and ungapped). |

**Resource parameters**

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>memory.request</td>
<td>4gb</td>
<td>Memory request. Include the units gb in the setting.</td>
</tr>
<tr>
<td>java.heap.space</td>
<td>3000</td>
<td>Dynamic memory requirement, in megabytes. Do not enter a unit with the numeric value.</td>
</tr>
</tbody>
</table>

Table 132 lists the small RNA counts parameters that we do not recommend changing.

### Table 132 Small RNA counts internal parameter description

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
</table>
| counts.run                            | 1             | Whether or not to run the small RNA counts module. Allowed values:  
  - 0: Do not run the small RNA counts module.  
  - 1: Run the small RNA counts module during this analysis.  
  The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use. |
| SmallRNA.counts.mismatches            | 3             | The maximum number of mismatches used when computing statistics. Allowed values: Integers 0–10. |
| SmallRNA.matureForms.file             | —             | Path to the tab-separated file with mature form sequences in every precursor sequence. Relative and absolute paths are supported. |
Chapter 25 Run a Small RNA Counts Analysis

Small RNA counts output files

Mapped output

This section describes the output files the small RNA counts module creates from the mapped BAM input.

GTF files

The small RNA counts module creates a GTF file for precursor counts and one for mature counts (if mature parameter is provided). Each input GFF file contains the counts per feature in column 5. Table 134 describes the counts file fields. The names of these files are:

- precursor_counts.gtf
- matureForms_counts.gtf

The following is example content from a counts.gtf file:

```
chr1 . mirBase 3044539 3044599 0 - . Precursor_ID "hsa-mir-4251"; RPM "0.00";
```
Table 134 Small RNA counts GTF output file format description

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>seqname</td>
<td>The name of the sequence. Must be a chromosome or a scaffold.</td>
<td>chr1</td>
</tr>
<tr>
<td>source</td>
<td>The program that generated this feature.</td>
<td>–</td>
</tr>
<tr>
<td>feature</td>
<td>The name of this type of feature.</td>
<td>*miRNA</td>
</tr>
<tr>
<td>start</td>
<td>The starting position of the feature in the sequence. The first base is numbered 1.</td>
<td>3044539</td>
</tr>
<tr>
<td>end</td>
<td>The ending position of the feature (inclusive).</td>
<td>3044599</td>
</tr>
<tr>
<td>score</td>
<td>The tag count for the feature.</td>
<td>0</td>
</tr>
<tr>
<td>strand</td>
<td>The strand containing the feature. Values: • ‘+’: The feature is on the 3’ strand. • ‘-’: The feature is on the 5’ strand. • ‘.’: Unknown or not required.</td>
<td>–</td>
</tr>
<tr>
<td>frame</td>
<td>[Not applicable to miRNAs.]</td>
<td>–</td>
</tr>
<tr>
<td>attributes</td>
<td>A list of attributes delimited by semi-colons. Each attribute is a type/value pair, with type and value separated by a single space character. Type is a string matching the pattern [A-Za-z1-9_]+. Value is a number or a double-quoted string. Every attribute must end with a semi-colon.</td>
<td>ACC=&quot;MI000000&quot;: ID &quot;hsa-mir-4251&quot;;</td>
</tr>
</tbody>
</table>

**contig_counts.txt file**

This section describes the output file, `contig_counts.txt`, generated from the mapped BAM file. This output file is a tab-separated file listing the following for each the sequence id in the filter reference:

- The sequence id
- The number of reads on the positive strand that map to that sequence
- The number of reads on the negative strand that map to that sequence
- The size of the reference sequence

The following is an example of a few lines from the mapped `contig_counts.txt` file:

```
ID       Counts_Pos_Strand Counts_Neg_Strand RefSeq_size
chr20    2932    1       63025520
chr21    1312    0       48129895
chr22    159     2       51304566
```

**Mismatches stats files**

The small RNA module generates two stats files based on the mapped BAM input:

- `precursor_mismatch_stats.txt`
- `matureForms_mismatch_stats.txt`
These files list the sequences from either the input precursor and or the mature forms file, and provide the number of alignments that map to each sequence with 0, 1, 2, and 3 mismatches. The total number of alignments that map to the sequence is also provided. The following is an example of a few lines from a mismatch_stats.txt file:

<table>
<thead>
<tr>
<th></th>
<th>0MM</th>
<th>1MM</th>
<th>2MM</th>
<th>3MM</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>hsa-let-7a</td>
<td>48 (64.00%)</td>
<td>15 (20.00%)</td>
<td>3 (4.00%)</td>
<td>5 (6.67%)</td>
<td>75 (100.00%)</td>
</tr>
<tr>
<td>hsa-let-7a*</td>
<td>1 (16.67%)</td>
<td>0 (0.00%)</td>
<td>4 (66.67%)</td>
<td>0 (0.00%)</td>
<td>6 (100.00%)</td>
</tr>
<tr>
<td>hsa-let-7a-2*</td>
<td>0 (0.00%)</td>
<td>0 (0.00%)</td>
<td>0 (0.00%)</td>
<td>0 (0.00%)</td>
<td>0 (-)</td>
</tr>
<tr>
<td>hsa-let-7b</td>
<td>17 (33.33%)</td>
<td>7 (13.73%)</td>
<td>16 (31.37%)</td>
<td>7 (13.73%)</td>
<td>51 (100.00%)</td>
</tr>
<tr>
<td>hsa-let-7b*</td>
<td>1 (33.33%)</td>
<td>0 (0.00%)</td>
<td>1 (33.33%)</td>
<td>1 (33.33%)</td>
<td>3 (100.00%)</td>
</tr>
</tbody>
</table>

Filtered counts output

This section describes the output file, contig_counts.txt, generated from the filtered BAM file. This output file is tab-separated and lists the following for each the sequence id in the filter reference:

- The sequence id
- The number of reads on the positive strand that map to that sequence
- The number of reads on the negative strand that map to that sequence
- The size of the reference sequence

The following is an example of a few lines from the filtered contig_counts.txt file:

<table>
<thead>
<tr>
<th>ID</th>
<th>Counts_Pos_Strand</th>
<th>Counts_Neg_Strand</th>
<th>RefSeq_size</th>
</tr>
</thead>
<tbody>
<tr>
<td>3000072055953=Tigger4a#DNA/MER2_type</td>
<td>0</td>
<td>0</td>
<td>236</td>
</tr>
<tr>
<td>chr16.trna19-GlyGCC</td>
<td>69</td>
<td>0</td>
<td>71</td>
</tr>
<tr>
<td>chr17.trna14-ThrCGT</td>
<td>2</td>
<td>0</td>
<td>72</td>
</tr>
<tr>
<td>3000072056044=tRNA-Tyr-TAC#tRNA</td>
<td>0</td>
<td>0</td>
<td>76</td>
</tr>
<tr>
<td>chr2.trna27-GlyCCC</td>
<td>128</td>
<td>0</td>
<td>71</td>
</tr>
</tbody>
</table>
Chapter 25  Run a Small RNA Counts Analysis

Small RNA counts output files
PART VIII

Additional Mapping Analyses
Run a ChIP-Seq Mapping Analysis

This chapter covers:
- Overview .................................................. 471
- Run ChIP-Seq mapping .................................. 471
- Run ChIP-Seq as an individual analysis .............. 474
- Mapping algorithm ...................................... 474
- Use results files ........................................... 474

Overview

LifeScope™ Genomic Analysis Software provides the ability to map data and create an output file type compatible with a variety of third-party Chromatin Immunoprecipitation Sequencing (ChIP-Seq) data analysis tools. The ChIP-Seq application has publicly available analysis software that can be used with LifeScope™ Software output.

The ChIP assay is a method for analyzing epigenetic modifications and genomic DNA sequences bound to specific regulatory proteins. ChIP-Seq is a combined assay and sequencing technique for identifying and characterizing elements in protein-DNA interactions. It typically examines transcription factors (TF) bound to DNA and finds DNA sequence motifs common to binding sites.

Using the MAGnify™ ChIP-Seq kit with the SOLiD® sequencing system enables you to generate sequence read data from a ChIP-Seq experimental approach. LifeScope™ Software gives you the option to map the read data.

Run ChIP-Seq mapping

ChIP-Seq analysis is available as workflow in the LifeScope™ Software command shell.

Run in the lscope command shell
This section describes how to run the fragment ChIP-Seq mapping workflow in the LifeScope™ Software command shell.
In the LifeScope™ Software command shell, running the ChIP-Seq workflow requires the following steps:

- Create a LifeScope™ Software command shell project and analysis.
- Identify your input data (your reads).
- Identify the reference genome.
- Select the ChIP-Seq the workflow as the analysis to be executed on your data.
- Issue the run command to start your analysis.

**Example steps to run ChIP-Seq mapping as standard workflow**

This section shows the steps to run the ChIP-Seq workflow in the LifeScope™ Software command shell. A description of each step is given in Table 135.

```
# log into the shell
lscope.sh shell -u username -w password
# import your data into the reads repository
cd /reads
import /data/xsq/xsq1.xsq
# cd to the projects repository
cd /projects
# make a project, and open it
cd /projects
mk chip_seq
cd chip_seq
# make a analysis, and open it
mk run1
cd run1
# define the analysis type (the ChIP-Seq workflow)
set workflow chip.seq
# define the input
add xsq xsq1.xsq lung
# specify the reference to be used
set reference /data/results/references/DH10B.fasta
# list the analysis configuration
ls
# run the analysis
run
# display progress information on your run
ls
```

Dummy XSQ file names are used in the example.

**Information about your workflow job**

Output of the `ls` command displays the analysis’ parameters, INI files, and run completion percentage or completion status.

Table 135 explains the purpose of the steps shown above.

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>lscope.sh shell -u username</td>
<td>Log into the LifeScope™ Software command shell.</td>
</tr>
</tbody>
</table>
### Table 135 Description of sample shell commands to run the ChIP-Seq workflow (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>cd /reads import /data/xsq/DH10B_ColorSpaceOnly_Unbarcoded_50_Frag.xsq</td>
<td>Go into the reads repository and import an XSQ file (the input data file). This command in effect brings the reads file into the shell reads repository. Once in the reads repository, the reads file is available as input to your analysis runs.</td>
</tr>
<tr>
<td>cd /projects mk chip_seq cd chip_seq mk run1 cd run1</td>
<td>In the projects repository, create a project named <code>ecoli</code>, and in that project create an analysis named <code>run1</code>. Open the <code>run1</code> analysis. <strong>Note:</strong> A project name cannot include a dash character (&quot;-&quot;). <strong>Note:</strong> The open analysis (<code>cd run1</code>, in this case) command is important. The data, reference, and other configuration commands which follow apply only to the current analysis.</td>
</tr>
<tr>
<td>set workflow chip.seq</td>
<td>Define the workflow used with this analysis. The ChIP-Seq workflow includes mapping, mapping statistics, and duplicate bead finding.</td>
</tr>
<tr>
<td>add xsq DH10B_ColorSpaceOnly_Unbarcoded_50_Frag.xsq</td>
<td>Define the XSQ file containing the input reads data for this analysis.</td>
</tr>
<tr>
<td>set reference /data/results/results/references/DH10B.fasta</td>
<td>Define the reference file for this analysis.</td>
</tr>
<tr>
<td>set saet.run 0 secondary/saet.ini</td>
<td>Turns off the SAET module. SAET IS NOT recommended for ChIP-Seq.</td>
</tr>
<tr>
<td>ls</td>
<td>Display the configuration of your analysis, including input reads, reference files, workflow, and select parameter settings.</td>
</tr>
<tr>
<td>run</td>
<td>Start the analysis run.</td>
</tr>
<tr>
<td>ls</td>
<td>Display status information about the analysis run.</td>
</tr>
</tbody>
</table>

See Chapter 7, “Run a Standard Workflow Analysis” on page 103 for more information on running workflows, including the ChIP-Seq workflow.

**Do not rename**

Do not delete or rename any of the following directories or files for your workflow:

- The `analysis.pln` file
- The `secondary` directory
- The `tertiary` directory
- The `secondary.pln` file

**Turn off SAET**

We recommend not running SAET with the ChIP-Seq workflow. When you turn off SAET in a workflow, make sure your mapping INI file does not contain the line:

```
analysis.input.readset.file = ${analysis.output.dir}/saet/*.rrs
```

Comment out this line if it appears in your mapping INI file.
Run ChIP-Seq as an individual analysis

The optional examples download includes an example of how to run the ChIP-Seq module by itself, as an individual analysis that is not part of a standard workflow.

See Appendix H, “Demo Analyses” on page 567 for information on the examples.

Mapping algorithm

The ChIP-Seq module uses the resequencing mapping and pairing algorithm. See the following sections for more information:

- “Stages of mapping” on page 131
- “Mapping algorithm” on page 145
- “Mapping statistics” on page 159
- “Mapping output files” on page 159

Use results files

After the mapping steps are complete, the resulting BAM file can be used with compatible third-party commercial and academic ChIP-Seq analysis software tools (see Figure 46 on page 475).

As of this writing, you can download a BAM-to-BED format converter from third-party tools sites, for example:

code.google.com/p/bedtools
Figure 46 Compatible ChIP-Seq analysis software tools
Chapter 26  Run a ChIP-Seq Mapping Analysis

Use results files
CHAPTER 27

Run a MethylMiner™ Mapping Analysis

This chapter covers:

- Overview ................................................................. 477
- Run MethylMiner™ mapping ........................................ 478
- MethylMiner™ mapping output files ........................... 481
- Mapping algorithm ..................................................... 481
- Further analysis of MethylMiner™ mapping results .......... 481

Overview

DNA methylation is an epigenetic modification crucial for organism development and normal gene regulation. Life Technologies have introduced a versatile methyl-CpG binding protein-based system, the MethylMiner™ Kit, for the enrichment of methylated sequences from genomic DNA, that, with the use of SOLiD® System sequencing, allows for focused evaluation of methylation patterns in genome-wide studies. The enriched reads can also be bisulfite-converted, which would additionally allow determination of methylation status of individual cytosines.

SOLiD® System sequencing can also, of course, be used to study methylation in unenriched, whole-genome bisulfite-converted read data.

The purpose of LifeScope™ Genomic Analysis Software MethylMiner™ module is to provide a data analysis pipeline for mapping and analyzing MethylMiner™ enriched and unenriched fractions of genomic DNA as well as bisulfite-converted and unconverted reads sequenced on the SOLiD® System.

The MethylMiner™ analysis module functionality currently includes:

- Mapping of unconverted and bisulfite-converted reads.
- Mapping statistics and statistics on read coverage and depth.
- Mapped reads output in BAM-format files.
- Visualization of mapped reads on publicly available genome browsers.
Figure 47 shows the MethylMiner™ analysis workflow with SOLiD® software and third-party browsers.

### Run MethylMiner™ mapping

Unconverted MethylMiner™ reads must be mapped to a regular (unconverted) reference genome sequence.

Bisulfite-converted reads must be mapped to a pair of appropriately converted reference sequences (forward and reverse conversions), as recommended below.

**Note:** MethylMiner™ supports fragment and paired-end libraries. Do not use this module with mate-pair libraries.

Follow these recommendations to map MethylMiner™ bisulfite-converted reads:

1. For mapping bisulfite-converted reads, the converted reference sequence pairs below are recommended. Use either Pair 1 or Pair 2 below.
   - Pair 1:
     - Reference with all non-CpG C’s converted to T’s (the forward conversion)
     - Reference with all non-CpG G’s converted to A’s (the reverse conversion)
   
   We have found that Pair 1 is most appropriate for bisulfite-converted reads from MethylMiner™ samples.

   - Pair 2:
     - Reference with all C’s converted to T’s
- Reference with all G’s converted to A’s

Pair 2 is most appropriate for bisulfite-converted reads from whole-genome data (not MethylMiner™ enriched).

2. The two sets of reference sequences within a given pair must be saved as separate reference files, and the mapping must be performed separately with each of them.

**Run MethylMiner™ workflows**

MethylMiner™ analysis is available as two standard workflows in the LifeScope™ Software command shell:

- For fragment data: methyl.miner.frag
- For paired-end data: methyl.miner.pe

These workflows use the resequencing mapping modules to perform mapping to a selected reference and generate standard BAM files.

The additional consolidation of the results from the forward and reverse conversions must be done by the user, and further analysis can be done with third-party tools.

**Run in the lscope command shell**

This section describes how to run the fragment MethylMiner™ mapping workflow in the LifeScope™ Software command shell.

In the LifeScope™ Software command shell, running the MethylMiner™ workflow requires the following steps:

- Create a LifeScope™ Software command shell project and analysis.
- Identify your input data (your reads).
- Identify the reference genome.
- Specify the MethylMiner™ fragment workflow as the analysis type to be executed on your data.
- Issue the run command to start your analysis.

**Example steps to run the MethylMiner™ fragment mapping standard workflow**

This section shows the steps to run the MethylMiner™ fragment mapping workflow in the LifeScope™ Software command shell. A description of each step is given in Table 136.

```
lscope.sh shell -u username -w password
cd /reads
import /data/xsq/DH10B_ColorSpaceOnly_Unbarcoded_50_Frag.xsq
cd /projects
mk methyl
cd methyl
mk run1
cd run1
set workflow methyl.miner.frag
add xsq DH10B_ColorSpaceOnly_Unbarcoded_50_Frag.xsq  lung
set reference /data/results/references/DH10B.fasta
set saet.run 0 secondary/saet.ini
ls
run
ls
```

Dummy XSQ file names are used in the example.
Chapter 27  Run a MethylMiner™ Mapping Analysis

Run MethylMiner™ mapping

Information about your workflow job

Output of the `ls` command displays the analysis’ parameters, INI files, and run completion percentage or completion status.

Table 136 explains the purpose of the steps shown above.

Table 136  Description of sample shell commands to run the MethylMiner™ workflow

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>lscope.sh shell -u username</code></td>
<td>Log into the LifeScope™ Software command shell.</td>
</tr>
<tr>
<td><code>cd /reads</code>&lt;br&gt;<code>import /data/xsq/DH10B_ColorSpaceOnly_Unbarcoded_50_Frag.xsq</code></td>
<td>Go into the reads repository and import an XSQ file (the input data file). This command in effect brings the reads file into the shell reads repository. Once in the reads repository, the reads file is available as input to your analysis runs.</td>
</tr>
<tr>
<td><code>cd /projects</code>&lt;br&gt;<code>mk methyl</code>&lt;br&gt;<code>cd methyl</code>&lt;br&gt;<code>mk run1</code>&lt;br&gt;<code>cd run1</code></td>
<td>In the projects repository, create a project named <code>ecoli</code>, and in that project create an analysis named <code>run1</code>. Open the <code>run1</code> analysis. <strong>Note:</strong> The open analysis (<code>cd run1</code>, in this case) command is important. The data, reference, and other configuration commands which follow apply only to the current analysis.</td>
</tr>
<tr>
<td><code>set workflow methyl.miner.frag</code></td>
<td>Define the workflow used with this analysis. The MethylMiner™ workflow includes SAET, mapping, mapping statistics, and duplicate bead finding.</td>
</tr>
<tr>
<td><code>add xsq DH10B_ColorSpaceOnly_Unbarcoded_50_Frag.xsq</code></td>
<td>Define the XSQ file containing the input reads data for this analysis.</td>
</tr>
<tr>
<td><code>set reference /data/results/results/references/DH10B.fasta</code></td>
<td>Define the reference file for this analysis.</td>
</tr>
<tr>
<td><code>set saet.run 0 secondary/saet.ini</code></td>
<td>Turns off the SAET module. SAET IS NOT recommended for the MethylMiner™ workflow.</td>
</tr>
<tr>
<td><code>ls</code></td>
<td>Display the configuration of your analysis, including input reads, reference files, workflow, and select parameter settings.</td>
</tr>
<tr>
<td><code>run</code></td>
<td>Start the analysis run.</td>
</tr>
<tr>
<td><code>ls</code></td>
<td>Display status information about the analysis run.</td>
</tr>
</tbody>
</table>

See Chapter 7, “Run a Standard Workflow Analysis” on page 103 for more information on running workflows, including the MethylMiner™ workflows.

Do not rename

When using a standard workflow, do not delete or rename any of the following directories or files:

- The `analysis.pln` file
- The `secondary` directory
- The `tertiary` directory
- The `secondary.pln` file
Chapter 27  Run a MethylMiner™ Mapping Analysis

MethylMiner™ mapping output files

Mapped reads are output in BAM files.

The output mapping statistics reports generated by MethylMiner™ mapping include:

- The number and percent of all reads mapped and of reads mapped uniquely, in the file mapping-stats.txt and also in files ending in F3.stats and F5-P2.stats.
- Genome coverage tables for at least read depths 0X to 50X, in output files ending in coverage-histogram-F3.txt and coverage-histogram-F5-P2.txt.

Mapping algorithm

The MethylMiner™ module uses the resequencing mapping and pairing algorithm. See the following sections for more information:

- “Stages of mapping” on page 131
- “Mapping algorithm” on page 145
- “Mapping statistics” on page 159
- “Mapping output files” on page 159

Further analysis of MethylMiner™ mapping results

Besides viewing mapped reads in genome browsers, you can further analyze mapped reads outside of LifeScope™ Software with software available in the SOLiD® development community or with other third-party tools. MethylMiner™ unconverted mapped reads can be processed with peak-finding programs like MACS to identify genome regions of significant methylation.

Similarly, MethylMiner™ bisulfite-converted mapped reads can be processed with peak-finding programs to identify regions of significant methylation. These reads can also be processed at nucleotide resolution to report the methylation status of individual C bases, for bases covered at sufficient read depth.

Visualize MethylMiner™ results

After the mapping steps are complete, the resulting BAM file with mapped reads can be visualized with compatible third-party commercial software tools and publicly-available genome browsers (see Figure 47 on page 478).

To visualize the mapped reads, launch your UCSC genome browser or IGV browser to import and view the mapped read BAM file generated by MethylMiner™ mapping.
Chapter 27  Run a MethylMiner™ Mapping Analysis

Further analysis of MethylMiner™ mapping results
PART IX
Appendices
This appendix covers:

- FAQ lists for LifeScope™ Genomic Analysis Software modules .......... 485
- General FAQ for LifeScope™ Software ........................................... 485

FAQ lists for LifeScope™ Genomic Analysis Software modules

Frequently-asked questions for modules are available in the analysis module chapters. See the following sections for module-specific FAQ lists:

- “FAQ – Mapping”
- “FAQ – Pairing”
- “FAQ – BAMStats”
- “FAQ – SNPs”
- “FAQ – Human CNVs”
- “FAQ – Small indels”
- “FAQ – Large indels”
- “FAQ - Annotations”
- “FAQ – Whole transcriptome”
- “FAQ - Small RNA mapping”
- “FAQ - XSQ Tools”

General FAQ for LifeScope™ Software

1

What is the performance increase vs. RAM increase?

For the human genome, 24GB of RAM per node provides approximately 30% to 40% performance improvement over systems that have 16GB of RAM per node.
2

Does LifeScope™ Software support hyper-threading?
Yes, some LifeScope™ Software components can take advantage of hyper-threading if it is present.

3

Can I configure LifeScope™ Software to submit jobs to the queues that are already set up on my cluster?
You can use pre-existing queues. During installation you will be asked which existing queues you would like to use with LifeScope™ Software.

4

What is the expected time for my run to complete?
This depends on many factors, like the number of beads, the reference, the hardware of the cluster in use, the traffic condition if it's a shared cluster, the speed of storage system, etc.

In a typical dedicated three to five nodes cluster, with 8 cores per node and 24GB memory per node, for one (1) full slide in a mate-pair run, it takes about two days to do the mapping, pairing, and generate BAM files for 1 billion reads. Most analyses can be done within one or two days.

5

How do I know if my run is successfully complete?
Use the LifeScope™ Software command shell ls command in your analysis, or the lscope.sh status command at the Linux prompt.

See “Review job status” on page 99 and “Review logs” on page 100.

6

What should I do if my run never seems to complete?
Verify that your run partition is not full and that the queueing system is working correctly. If your run directory’s partition is 100% full, LifeScope™ Software may report that your job has started, but LifeScope™ Software is not able to process the job.

7

Do you have example data to test my install with?
Ask your LifeScope™ Software administrator if the optional Examples directory was installed.

8

Can the “scratch” directory be on a shared NFS storage?

Scratch space can be a local disk or shared storage mounted via NFS, but there will be high I/O activity on the scratch disk. Ensure the storage has high throughput and will not degrade the overall performance of the system. LifeScope™ Software is not tested in an environment that is configured with scratch space that is mounted via NFS.

9

Can multiple users try to install LifeScope™ Software in different locations?

No. There is no need to install multiple copies. You just need to set the environment to point to LifeScope™ Software. See “Set the LifeScope™ Software environment” on page 60.

10

Can I continue to run my BioScope™ Software after I install LifeScope™ Software?

Yes. You must follow the instructions in “BioScope™ Software users’ PATH variable” on page 60.

11

What changes are required for a job with a large number of contigs?

For a job with a large (more than a few thousand) number of contigs, change the system property max.head.node.java.heap.space from the default of 64 to 128 or 256.

For example, a run with 40,000 contigs required 128 for BAMStats and 256 for SNPs. Modify this parameter in the <installdir>/etc/analysis/system.properties file.

Because changes to system properties affect all runs of all users, you could consider changing the parameter back to its original value after the large contig run completes. Setting max.head.node.java.heap.space to 128 or 256 could present a problem if your head node is extremely loaded.
Troubleshooting

This appendix covers:
- Overview ................................................................. 489
- Logs ............................................................................. 489
- Troubleshooting ......................................................... 490

Overview

This appendix describes job log files and troubleshooting tips.

Logs

An analyses’ log files are located in the results directory (set during the installation) under the following directory naming scheme:

username/project_name/analysis_name

The following logs contain messages related to your analysis:
- analysis.log – A full log of entire experiment
- summary.log – A log containing milestone messages
- log/module/* .log – Log files generated by the analysis modules in your run (modules such as mapping, SNPs, CNV, etc.)
- log/TORQUE/module/* .log – Log files from the compute node

The TORQUE logs contain the following information of interest:
- The node the job is run on
- Job kill notification for timeout failures

Server log messages and errors are captured in the following files:
- <installdir>/server/server.log
- <installdir>/server/server.err.
Troubleshooting

Login issues

- Check that the LifeScope™ Software server is running.
  - The Linux command `ps -ef | grep WebServer` displays the server process, if it is running.
  - The file `<installdir>/server/server.pid` lists the process id and port number for the server.

Analysis failures

- Use the `lscope.sh status` command or the GUI status information to identify which analysis module failed and to check the error message.
- If the error message is not sufficient to diagnose the problem, use the summary log and the analysis module log to identify more details on the problem.

The server responds with an error

- Check the files `<installdir>/server/server.log` and `<installdir>/server/server.err`.
- Check the analysis configuration for sample grouping errors. In the command shell read-sets are grouped with `add xsq` commands options or with a groupsfile. In the GUI, groups are created in the Add Data to Project and Add Group to Project windows.

Note: In this release, sample group names and reference file names containing a hyphen or a space are not permitted.

Message: Read-only db connection

If power was lost, causing the LifeScope™ Software server to shut down, and the server generates the message “read-only db connection,” type the following commands:

1. `lscope-server.sh stop`
2. `rm /share/apps/lifescope/server/UserDB/db*.lck`
3. `lscope-server.sh start`

If the problem persists, or if the error message is different (indicating a corrupt db), type the following commands to delete the UserDB, which will remove all the current users in the system. Then recreate the users.

1. `lscope-server.sh stop`
2. `rm -rf /share/apps/lifescope/server/UserDB`
3. `lscope-server.sh start`

Sequence name not found in reference

An analysis fails with a reference error, but the reference file is in the reference repository. The error message is similar to the following:

12 Apr 2011 16:17:41,750 ERROR [0x2a96825860] reference sam_check: 49 - sequence name chr6 was not found in reference file;

If the error is seen with an analysis that previously ran without error, a conflict in the reference repository is possibly the cause. LifeScope™ Software first looks in the reference repository by assembly name, and then in the assembly folder searches for a reference file that includes the assembly name in its file name. If a file with the assembly name is not found, the software then uses the first file in that folder (alphabetically). The reference selected may or may not contain the required sequence.
The avoid this type of error, when adding a new reference file to the repository, always add it under the subfolder for the correct assembly name. Create a new assembly name subfolder under referenceData/internal when adding a new genome. See Appendix E, “The Reference Repository” on page 539 for more information.

To recover from this error, you could use the set reference command with an absolute path to the reference file on the Linux file system.

**Memory errors**

If a job is killed for exceeding its requested memory limit, an error similar to the following is seen (the example memory numbers are unimportant):

```
PBS: job killed: mem job total 4203328 kb exceeded limit 4194304 kb
```

Job schedulers can be configured to kill jobs that exceed their requested memory or to be tolerant as long as memory is available.

Use the memory.request or mapping.memory parameters to configure the memory request for each analysis module.

**Special characters in BAM header fields**

The following special characters are not allowed in XSQ file fields that mapped to BAM header fields:

- **Slashes** – /, \  
- **Dollar sign** – $  
- **Greater than and less than signs** – >, <

Examples of BAM header fields in which these characters are not allowed are the following:

- AT in @CO  
- LD in @CO  
- PJ in @CO  
- SP in @CO  
- SO in @CO  
- CO in @CO  
- SD in @CO  
- ID in @PG  
- SM in @RG

The presence of unsupported special characters in these fields can cause errors such as null pointer exceptions.
File Format Descriptions

This appendix covers:
- Introduction ................................................................. 493
- XSQ file format ............................................................... 494
- BAM header usage ........................................................... 498
- Color-space attributes ...................................................... 504
- Pairing information in a BAM file ...................................... 505
- Hard clipping of incomplete extensions .............................. 506
- Visualize BAM output ...................................................... 507
- Indel alignments ............................................................. 508
- BED file format ............................................................... 510
- BEDGRAPH file format .................................................... 511
- Reference file data overview ............................................. 512
- Read-set file format ........................................................ 513
- Legacy format translation ................................................ 515
- Legacy CMAP file format description ............................... 516

Introduction

This appendix describes file formats used with LifeScope™ Software.

Before reading the section about the SOLiD® System BAM file contents, you should be familiar with the general SAM specification and with the SAM specification field definitions.

Before SOLiD® 4.0, the primary SOLiD® read and alignment format was based on the public GFF specification. The specification defined the location of an aligned read on a reference sequence and provided for arbitrary name-value pair attributes. For the SOLiD® GFF, attributes are used to detail color reads and quality values, base space translations, mate-pair information, and other aspects of the aligned read. Increased throughput of second-generation sequencing technologies resulted in the definition of the SAM file format and a compressed, indexed binary format (BAM) that expands the basic alignment information of a GFF file to include paired-read information and a more structured attribute set. A number of tools exist for basic manipulation of BAM files, and an array of viewers and analysis tools are available.
LifeScope™ Software secondary analysis (mapping and pairing) now produces a BAM file as the main alignment format. Mate-pair and paired-end analysis directly produces a BAM file, while a single file conversion is needed for fragment libraries. Depending on the output filter selected, unmapped and secondary alignments can be included.

LifeScope™ Software supports the XSQ sequence data file format introduced with the 5500 Series Genetic Analysis Systems.

**XSQ file format**

XSQ (eXtensible SeQuence) is an extensible file format for storing sequence data. The XSQ format supports multiple independent reads at the same position in a fragment. XSQ is a binary format based on the open HDF format.

Sequencing run data are automatically exported from the 5500 Series Genetic Analysis Systems in XSQ binary file format. If an ECC primer round has been performed, the XSQ output also includes the sequence information in base space, in addition to color space.

### XSQ file content overview

**5500 Series Genetic Analysis Systems data**

XSQ files generated by the 5500 Series Genetic Analysis Systems contain:

- **Base-space data** – Reference-free base-space data from primary analysis (if the optional ECC round is performed).
- **Color-space data** – Referred to as 2+4 color, as follows:
  - The 2BE (Base Encoding) data is the data from first 5 primer rounds of regular 2BE probes. This data is in the same format as a data created by a SOLiD® 4 System non-ECC run.
  - The 4BE data is the data from the 6th primer round, the optional ECC round.

Data for optional ECC primer rounds is supported in an XSQ file.

Paired-end libraries do not support ECC data. Fragment libraries support ECC data, and mate-pair libraries support ECC data on both tags.

Trimming is allowed with base-space reads, and can occur at the beginning as well as the end of a read. The trimming is typically an output of an ECC run. The mapping of trimmed reads requires that the untrimmed portion be at least as long as the mapping seed/index length.

**SOLiD® 4 System data**

SOLiD® 4 System CSFASTA and QUAL files can be used with LifeScope™ Software, but first must be converted to XSQ format. Converted XSQ files contain only:

- Color-space 2BE data
- Color for both tags (one tag for fragment and both tags for mate-pair and paired-end data).

See Appendix D, “XSQ Tools” on page 519 for information on the XSQ converter.
Relation to sequencing instrument

This section describes how physical aspects of the 5500 Series Genetic Analysis Systems instrument affect the output XSQ files.

Each lane of the 5500 instrument results in a single XSQ output file. Different models of the 5500 support from 6 to 12 lanes, so typically multiple XSQ files are generated per sequencing run. You can direct your LifeScope™ Software analyses to data from specific barcodes run in different lanes, or group the output of multiple lanes or multiple sequencing runs to be processed as a single specimen.

The most granularity possible for a LifeScope™ Software analysis is the data from one barcode in one lane.

For information on specifying barcodes and specimen groups, see the following material:

- The add xsq command in Table 13 “LifeScope™ Software shell commands” on page 81.
- “Run a grouped analysis” on page 93 in Chapter 6, “Run a Command Shell Analysis” on page 75

XSQ file format description

This section describes several tables within an XSQ file. The complete XSQ file format specification is available on the SOLiD® Software Community website.

The LibraryDetails table describes the properties of each library used in the XSQ file. String fields are limited in length to 255 characters.

Table 137  XSQ library details table

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LibraryName</td>
<td>Describes the name of one library preparation, which may be used across several indexes in an indexing run.</td>
</tr>
<tr>
<td>Application</td>
<td>Describes how this sequence is intended to be analyzed. Allowed values are: whole genome resequencing, targeted resequencing, whole transcriptome (Fragment), whole transcriptome (PairedEnd), small RNA, ChIP-Seq, methylation</td>
</tr>
<tr>
<td>ProjectName</td>
<td>Describes the project within which this sample is being run.</td>
</tr>
<tr>
<td>SampleOwner</td>
<td>Names the owner of this sample.</td>
</tr>
<tr>
<td>SampleIdentifier</td>
<td>A unique name for this sample.</td>
</tr>
<tr>
<td>Description</td>
<td>Describes the sample.</td>
</tr>
<tr>
<td>Comments</td>
<td>Optional additional information about this sample or its preparation.</td>
</tr>
<tr>
<td>ERCC</td>
<td>Whether internal controls have been used. Reserved for future use.</td>
</tr>
<tr>
<td>Species</td>
<td>The intended species for mapping. Example values include: homo sapiens, mus musculus, other</td>
</tr>
</tbody>
</table>
Appendix C  File Format Descriptions

XSQ file format

Table 137  XSQ library details table (continued)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assembly</td>
<td>The intended assembly for mapping (may be blank if unknown). Example values include:</td>
</tr>
<tr>
<td></td>
<td>• hg18</td>
</tr>
<tr>
<td></td>
<td>• hg19</td>
</tr>
<tr>
<td></td>
<td>• mm9</td>
</tr>
<tr>
<td></td>
<td>Other valid assembly names are accepted. Spaces are not allowed.</td>
</tr>
<tr>
<td>LibraryInsertSize Minimum</td>
<td>The expected minimum insert size in nucleotides. Used with paired-end and mate-pair libraries. Allowed values: Integers &gt;= 0. Set to 0 for fragment libraries, or when the size is unknown.</td>
</tr>
</tbody>
</table>

Each read type (tag) has a tag-level properties in the XSQ file. Table 138 lists the attributes describing the base-space and color-space data for a single type of read. Table 139 on page 497 lists tag names for current library types.

Table 138  XSQ tag details table

<table>
<thead>
<tr>
<th>Column name</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TagSequence</td>
<td>—</td>
<td>The full sequence of the tag used for primer annealing. It is used in conjunction with the read offset to determine the nucleotide or nucleotides for phasing the color-space reads. When the full sequence is not available, the last base(s) of the tag are sufficient as long as they cover the largest offset in the DatasetColorEncoding table. Conditional: Required when color space is present.</td>
</tr>
<tr>
<td>IsColorPresent</td>
<td>0</td>
<td>Whether color-space call data is present. Values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: No calls are reported in color space (causing the ColorCallQV dataset to be missing).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Non-zero: Color calls are reported.</td>
</tr>
<tr>
<td>IsBasePresent</td>
<td>1</td>
<td>Whether base call data is present. Values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: No calls are reported in base space (causing the BaseCallQV dataset to be missing).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Non-zero: Base calls are reported.</td>
</tr>
<tr>
<td>NumBaseCalls</td>
<td>—</td>
<td>The maximum untrimmed read length in base space across all reads for this tag. This length is the width of the BaseCallQV dataset. This field is not used when the BaseCallQV dataset is not present, but is required when BaseCallQV is present.</td>
</tr>
<tr>
<td>MinTrimmedReadLength</td>
<td>—</td>
<td>The minimum trimmed read length across all unfiltered reads for this tag. Allowed values: Integers &gt;= 0.</td>
</tr>
<tr>
<td>DoesRepresentedStrandMatchSource</td>
<td>1</td>
<td>Whether the reads are represented on the same strand as the source. Values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: The synthesis strand is not the same as the source strand.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Non-zero: The synthesis strand is the same as the source strand.</td>
</tr>
<tr>
<td>RelativeOrder</td>
<td>—</td>
<td>Describes the relative order of a read on the sense strand of the source nucleic acid, in relation to all of the other reads from the same fragment. 1 based. Required only when there is more than one tag per fragment.</td>
</tr>
</tbody>
</table>
Table 138  XSQ tag details table (continued)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>
| SynthesisFromThreePrime      | 0       | (Optional) Whether the synthesis direction of the synthesized strand is from the 3’ end. Values:  
|                              |         | • 0 or missing: synthesis from the 5’ end to the 3’ end.  
|                              |         | • Non-zero: The synthesis direction is from the 3’ end to the 5’ end. |

Table 139 lists tag names for current library types

Table 139  Tag information for current library types

<table>
<thead>
<tr>
<th>Instrument</th>
<th>Library type</th>
<th>Indexing</th>
<th>Tag name</th>
<th>Base</th>
<th>Does-Represented-StrandMatch-Source</th>
<th>Relative-Order</th>
<th>Synthesis-FromThree-Prime</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOLiD®</td>
<td>Fragment</td>
<td>Yes/No</td>
<td>F3</td>
<td>T</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SOLiD® and 5500</td>
<td>Mate-pair</td>
<td>No</td>
<td>F3</td>
<td>T</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>SOLiD®</td>
<td>Paired-end</td>
<td>No</td>
<td>F3</td>
<td>T</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>F5-P2</td>
<td>T</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>SOLiD®</td>
<td>Paired-end</td>
<td>Yes</td>
<td>F3</td>
<td>T</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>F5-BC</td>
<td>G</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>5500</td>
<td>Paired-end (DNA)</td>
<td>Yes/No</td>
<td>F3</td>
<td>T</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>F5-DNA</td>
<td>T</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>5500</td>
<td>Paired-end (RNA)</td>
<td>Yes/No</td>
<td>F3</td>
<td>T</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>F5-RNA</td>
<td>G</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 139 lists tag names for current library types

The DatasetColorEncoding table in the XSQ file is required when color space is present. The table describes how reads are encoded as colors by the instrument. Indexing reads are not included in this table. The DatasetColorEncoding table is described in Table 140.

Table 140  The DatasetColorEncoding table

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DataSetName Must match the name of a read dataset under a tag group. This table describes the encoding scheme used to interpret that read dataset.</td>
</tr>
<tr>
<td>2</td>
<td>Offset</td>
</tr>
<tr>
<td>3</td>
<td>Encoding</td>
</tr>
<tr>
<td>4</td>
<td>Stride</td>
</tr>
</tbody>
</table>
BAM header usage

The BAM file generates all of the header information required by the SAM format specification, including @HD, @SQ, and @RG lines.

To view the content of the BAM file header, use the following command:

```
samtools view -H <BAMfilename>
```

Sequence dictionary (@SQ)

Sequence header lines include the reference file URL, for example, `file://share/reference/genomes/hg18.fa` in the optional UR field of the reference file. This value might become invalid if you relocate files.

Read group (@RG)

Read groups receive an arbitrary ID and sample name. The library field (LB) contains information that is important to downstream algorithms that use pairing information. A library name, which is specified by the tool parameter `library.name`, and the library type, are separated by a dash in the LB field. The library type is a structured value that details the nominal length of the two tags and the protocol used as shown in the following syntax example:

```
11(x12) [F|MP|RR|RRBC]
```

In the syntax example, 11 is the nominal length of the first read and l2 is the nominal length of the second read. There will only be one number for fragment libraries. The library types correspond to fragment (F), mate-pair (MP), reverse read (RR), and reverse read-bar coded (RRBC).

Detecting structural variations, particularly large insertions and deletions, depends on the range of pairing insert (PI) sizes. The pairing tool generates the information about PI sizes. The information has been used to define the three-letter pairing category, specifically the third letter. (See Table 34 on page 176 and Table 35 on page 177 for descriptions of the three-letter categories.) The PI field in the read group captures the range of pairing insert sizes with a range of the form shown in the following example:

```
PI:low-high
```

In the PI example, low is the lower bound of the pairing range in nucleotides, and high is the upper bound.

Header (@HD) sort order

The LifeScope™ Software SNPs module requires that BAM headers of its input files contain an @HD line with a SO sort-order field, even if the BAM file is sorted properly. The required header line is:

```
@HD VN:1.0 GO:none SO:coordinate
```

This header contains the information that the BAM file is sorted by genomic coordinates. BFAST mapping, for example, by default sorts the BAM file by coordinates, but does not update the header to reflect the sort order.
This section describes how to update the @HD SO field. If your BAM file is sorted properly but does not contain the SO sort-order field, follow these instructions before using a BAM file generated outside of LifeScope™ Software with the SNPs module:

1. Write the BAM file header to a file
   
   ```bash
   samtools view -H input.bam > input_header.sam
   ```

2. Edit the BAM header file (input_header.sam created in the previous step). Insert the following as the first line of the header:
   
   ```
   @HD    VN:1.0    GO:none SO:coordinate
   ```
   Make sure that the fields are tab-separated.

3. Update your BAM file with this new header. This command requires samtools-0.1.8 or later.
   
   ```bash
   samtools reheader input_header.sam input.bam
   ```

These instructions are not required with BAM files generated by LifeScope™ Software mapping modules.

**BAM file validation**

In order to be used by LifeScope™ Software tertiary analysis tools, the BAM file must have the @RG field populated in the BAM header. If you are using BFAST mapping analysis, you must run it with the -R option in the post-processing step to create the @RG field in the BAM file header. BAM files created outside of LifeScope™ Software must pass Picard validation in order to be used with LifeScope™ Software tertiary analysis tools.

**Note:** Picard validation and samtools are supported by the open source project SourceForge.net. Applied Biosystems is not responsible for the availability or functionality of the Picard utility.

At the time of this writing the syntax for Picard validation is available at the following URL:


The Picard utility, `picard-tools-version.zip` (where version is the current version number) can be downloaded at:

[http://sourceforge.net/projects/picard/files](http://sourceforge.net/projects/picard/files)

The suggested Picard validation command is:

```bash
java -Xmx8g -jar ./ValidateSamFile.jar I=./input.bam \
   IGNORE=MISSING_TAG_NM
```

where `input.bam` is the name of the BAM file to be validated.

**XSQ metadata in BAM headers**

LifeScope™ Software secondary analysis modules write metadata contained in XSQ reads files to their mapping output files, as metadata in BAM header comment (@CO) lines. The new metadata is not in the BAM specification, but BAM files with this metadata conform to the BAM specification. Table 141 lists the XSQ file fields with both the new @CO field names and the existing BAM header fields that contain the XSQ metadata.
Tie to existing BAM header lines

Each @CO metadata line with metadata information references one of the following types of BAM header lines:

- **Program group** – @PG lines capture information about the program that generated the data originally. The sequencing instrument and primary analysis software are considered to be the initial program. An example @PG line is:
  ```
  @PG ID:Mordor_201012161921_0_3 [...] 
  ```
- **Read group** – @RG lines contain information on library type and read length. (See “Read group (@RG)” on page 498 for more information.) An example @RG line is:
  ```
  @RG ID:1 SM:Stooges LB:Larry DS:”75x35PE” PI:225 [...] 
  ```
- **General header** – @HD lines are general header lines.

All the attributes on the @CO metadata line then apply to referenced line.

The following example shows how @CO lines containing BAM metadata information refer to read group and program group lines. Except for @CO HD lines, the second field of the @CO metadata lines tie its attributes to either @RG or @PG lines. In the example, the RG:1 fields tie the metadata in those @CO lines to the @RG read group with an ID of 1. The PG:Mordor_201012161921_0_3 field ties the metadata in that @CO line to the @PG program group with an ID of Mordor_201012161921_0_3.

```plaintext
@CO RG:1 IA:215 IS:5.232 IN:200 IM:250
@CO RG:1 TN:50 TX:75 TB:0 TC:1
@CO RG:1 UN:25 UX:35 TB:0 TC:1
@CO RG:1 BX:0 EC:0
@CO PG:Mordor_201012161921_0_3 AS:”primary5500 1.0.7”
@CO PG:Mordor_201012161921_0_3 \
   CL:”/usr/bin/primary5500 <with full arguments>”
@CO HD UF:”f47ac10b-58cc-4372-a567-0e02b2c3d479”
```

Metadata table

Table 141 lists the existing BAM header fields that contain the XSQ metadata, the new @CO metadata field names, and the XSQ file fields. See also “Input read count fields” on page 502.

<table>
<thead>
<tr>
<th>Field name</th>
<th>Public BAM 1.3 spec field</th>
<th>New field in @CO line</th>
<th>XSQ field name</th>
<th>Optional or mandatory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference</td>
<td>@SQ SP</td>
<td>—</td>
<td>Species</td>
<td>Mandatory</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>{ ‘Homo sapiens’</td>
<td>‘Mus musculus’</td>
</tr>
<tr>
<td>Assembly</td>
<td>@SQ AS</td>
<td>—</td>
<td>Assembly</td>
<td>Mandatory</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>{ ‘hg18’</td>
<td>‘hg19’</td>
</tr>
<tr>
<td>Program</td>
<td>@PG ID</td>
<td>—</td>
<td>InstrumentName_Date_FlowcellAssignment_LaneNumber</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Analysis Software</td>
<td>@CO PG:x AS</td>
<td>AnalysisSoftware</td>
<td>Optional</td>
<td></td>
</tr>
<tr>
<td>InstrumentSerial</td>
<td>@CO PG:x PS</td>
<td>InstrumentSerial</td>
<td>Optional</td>
<td></td>
</tr>
<tr>
<td>InstrumentName</td>
<td>@CO PG:x PN</td>
<td>InstrumentName</td>
<td>Optional</td>
<td></td>
</tr>
</tbody>
</table>
### File Format Descriptions

#### BAM header usage

**Table 141** XSQ-related BAM metadata fields

<table>
<thead>
<tr>
<th>Field name</th>
<th>Public BAM 1.3 spec field</th>
<th>New field in @CO line</th>
<th>XSQ field name</th>
<th>Optional or mandatory</th>
</tr>
</thead>
<tbody>
<tr>
<td>InstrumentVendor</td>
<td>@CO PG:x PV</td>
<td>@RG ID</td>
<td>InstrumentVendor</td>
<td>Optional</td>
</tr>
<tr>
<td>InstrumentModel</td>
<td>@CO PG:x PM</td>
<td>@RG ID</td>
<td>InstrumentModel</td>
<td>Optional</td>
</tr>
<tr>
<td>ReadGroupID</td>
<td>@CO RG:x IX</td>
<td>LibraryName or LibraryName_IndexName [if present]</td>
<td>Mandatory</td>
<td></td>
</tr>
<tr>
<td>Sequencing Center</td>
<td>@CO RG:x IX</td>
<td>LibraryName or LibraryName_IndexName [if present]</td>
<td>Optional</td>
<td></td>
</tr>
<tr>
<td>Application Type</td>
<td>@CO RG:x AT</td>
<td>ApplicationType { 'Whole Genome Resequencing'</td>
<td>Mandatory</td>
<td></td>
</tr>
<tr>
<td>Library Description</td>
<td>@CO RG:x LD</td>
<td>LibraryDetails.Description</td>
<td>Optional</td>
<td></td>
</tr>
<tr>
<td>Library Type</td>
<td>@CO RG:x LT</td>
<td>LibraryType { 'MatePair'</td>
<td>Mandatory</td>
<td></td>
</tr>
<tr>
<td>Predicted Insert Size</td>
<td>@CO RG:x PI</td>
<td>PredictedInsertSizeMin + PredictedInsertSizeMax / 2</td>
<td>Conditional</td>
<td></td>
</tr>
<tr>
<td>Calculated Average Insert Size</td>
<td>@CO RG:x IA</td>
<td>&lt;calculated&gt;</td>
<td>Mandatory</td>
<td></td>
</tr>
<tr>
<td>Calculated Insert Size Std Dev</td>
<td>@CO RG:x IS</td>
<td>&lt;calculated&gt;</td>
<td>Mandatory</td>
<td></td>
</tr>
<tr>
<td>Min Insert Size from User Input</td>
<td>@CO RG:x IN</td>
<td>LibraryInsertSizeMinimum</td>
<td>Conditional</td>
<td></td>
</tr>
<tr>
<td>Max Insert Size from User Input</td>
<td>@CO RG:x IM</td>
<td>LibraryInsertSizeMaximum</td>
<td>Conditional</td>
<td></td>
</tr>
<tr>
<td>Input Read Count Passing All Filtering Steps</td>
<td>@CO RG:x CU</td>
<td>The sum across all ImageUnits of ImageUnitId.Fragments.NumFragmentsPassed [passes all filtering steps]</td>
<td>Optional</td>
<td></td>
</tr>
<tr>
<td>Input Read Count Total</td>
<td>@CO RG:x CT</td>
<td>The sum across all ImageUnits of ImageUnitId. FragmentCount [filtered and unfiltered]</td>
<td>Mandatory</td>
<td></td>
</tr>
<tr>
<td>Specimen</td>
<td>@CO RG:x SP</td>
<td>SampleIdentifier</td>
<td>Optional</td>
<td></td>
</tr>
<tr>
<td>Platform name</td>
<td>@RG PL</td>
<td>&quot;SOLID&quot;</td>
<td>Mandatory</td>
<td></td>
</tr>
<tr>
<td>Lane Id</td>
<td>@RG PU</td>
<td>FlowcellAssignment_LaneNumber</td>
<td>Optional</td>
<td></td>
</tr>
<tr>
<td>Sample (Pool)</td>
<td>@RG SM</td>
<td>SequencingSampleName</td>
<td>Optional</td>
<td></td>
</tr>
<tr>
<td></td>
<td>@CO RG:x SD</td>
<td>SequencingSampleDescription</td>
<td>Optional</td>
<td></td>
</tr>
</tbody>
</table>
Appendix C  File Format Descriptions

BAM header usage

### File Format Descriptions

#### BAM header usage

This section describes aspects of the Input Read Count fields from the viewpoint of the sequencing instrument and the viewpoint of secondary analysis.

The Input Read Counts fields are:

- **CU** – Input Read Count Passing All Filtering Steps, @CO RG:x CU
- **CT** – Input Read Count Total, @CO RG:x CT

The number of *total beads*, before any filtering on the instrument, is given by these fields:

- **NumFragmentsTotal** – In mapping statistics output
- **FragmentCount** – In an XSQ file
- **CT** – In BAM header metadata

The number of *beads that pass filtering* on the instrument is given by these fields:

- **NumFragmentsPassingFilters** – In mapping statistics output
- **NumFragmentsPassed** – In an XSQ file
- **CU** – In BAM header metadata

The number of *beads filtered out* on the instrument is given by:

\[
\text{NumFragmentsTotal} - \text{NumFragmentsPassingFilters}
\]

---

### Table 141  XSQ-related BAM metadata fields

<table>
<thead>
<tr>
<th>Field name</th>
<th>Public BAM 1.3 spec field</th>
<th>New field in @CO line</th>
<th>XSQ field name</th>
<th>Optional or mandatory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag1 Was Base Present In XSQ</td>
<td>—</td>
<td>@CO RG:x BX</td>
<td>Tag1.IsBasePresent</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Tag1 Min Read Len</td>
<td>—</td>
<td>@CO RG:x TN</td>
<td>Tag1.MinTrimmedReadLength</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Tag1 Max Read Len</td>
<td>—</td>
<td>@CO RG:x TX</td>
<td>Tag1.NumColorCalls or Tag1.NumBaseCalls</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Tag2 Was Base Present In XSQ</td>
<td>—</td>
<td>@CO RG:x BY</td>
<td>Tag2.IsBasePresent</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Tag2 Min Read Len</td>
<td>—</td>
<td>@CO RG:x UN</td>
<td>Tag2.MinTrimmedReadLength</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Tag2 Max Read Len</td>
<td>—</td>
<td>@CO RG:x UX</td>
<td>Tag2.NumColorCalls or Tag2.NumBaseCalls</td>
<td>Mandatory</td>
</tr>
<tr>
<td>ECC run</td>
<td>—</td>
<td>@CO RG:x EC</td>
<td>&lt;whether this was an ECC run (inferred)&gt;</td>
<td>Mandatory</td>
</tr>
<tr>
<td>ERCC</td>
<td>—</td>
<td>@CO RG:x ER</td>
<td>ERCC</td>
<td>Optional</td>
</tr>
<tr>
<td>—</td>
<td>—</td>
<td>@CO RG:x CO</td>
<td>Operator</td>
<td>Optional</td>
</tr>
<tr>
<td>—</td>
<td>—</td>
<td>@CO RG:x UU</td>
<td>LibraryIndexUUID</td>
<td>Optional</td>
</tr>
<tr>
<td>—</td>
<td>—</td>
<td>@CO RG:x PN</td>
<td>Application</td>
<td>Optional</td>
</tr>
<tr>
<td>—</td>
<td>—</td>
<td>@CO RG:x PJ</td>
<td>ProjectName</td>
<td>Optional</td>
</tr>
<tr>
<td>—</td>
<td>—</td>
<td>@CO RG:x SO</td>
<td>SampleOwner</td>
<td>Optional</td>
</tr>
</tbody>
</table>
Example of @CO metadata in a BAM header

The following is an example of a BAM header with an @CO comment line containing metadata information. The @CO line applies to the @RG ID:12345678 line below it. (The @SQ lines for chromosomes 2–22 are deleted.)

```
@HD VN:1.0 GO:none SO:coordinate
@SQ SN:chr1 LN:247249719 UR:/ref/hg18.fasta
  SP:Homosapiens AS:hg18 ...
@SQ SN:chrX LN:154913754 UR:/ref/hg18.fasta
  SP:Homosapiens AS:hg18
@SQ SN:chrY LN:57772954 UR:/ref/hg18.fasta
  SP:Homosapiens AS:hg18
@SQ SN:chrM LN:16571 UR:/ref/hg18.fasta
  SP:Homosapiens AS:hg18
@PG ID:SOLiD_12321_1
@CO RG:12345678 LT:MatePair AT:WholeGenomeResequencing LN:<50X50MP> PI:1500 IA:1500 IS:300 IN:1200 IM:1800 PU:1234 SM:sample BX:0 TN:50 TX:50 BY:1 UN:50 UX:60 EC:0
@RG ID:12345678 DS:50x50MP LB:MP PI:1500 PU:1234 SM:sample
```

@CO syntax and rules

This section describes the syntax rules and conventions for @CO metadata lines. The @CO metadata rules are:

- The second field of each metadata @CO line either notes that the line is a general header (HD), or refers to a single @RG ID or @PG ID.
- Each metadata @CO line contains one or more tab-separated attributes.
- For a metadata @CO line that refers to either a @RG or @PG line, all of the @CO attributes must apply to the same @RG or @PG.
- The beginning of metadata @CO lines must match the regular expression “^@CO\t\{PR\}G:” and contain no other non-metadata text. LifeScope™ Software ignores and does not change @CO lines that do not match the regular expression.
- The @CO PG:id line describing the instrument is required and must refer to a valid @PG ID:id program group line. This @CO line contains information about the primary analysis done on the instrument.

According to the BAM specification, the @HD line is optional. If present, the @HD line is required to be the first line of the header. The order of all other header lines is not defined. The following order of @CO metadata lines is preferred when possible (but not required):

- Place the @CO HD:UF comment immediately after the @HD line.
- Place @CO RG: and @CO PG: comment lines before other @CO lines for ease of visibility.
- Place @CO RG: and @CO PG: comment lines before the @RG and @PG blocks that the comment lines refer to.
Special character restrictions

Table 142 lists special characters that are not allowed in the LibraryName (@RG LB) and the LibraryType (LT in @CO) BAM header fields. Program failures are observed when either the LibraryName or the LibraryType fields contain one of these characters.

Table 142 Special characters not supported in the LibraryName and LibraryType fields

<table>
<thead>
<tr>
<th>Character name</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forward slash</td>
<td>\</td>
</tr>
<tr>
<td>Backwards slash</td>
<td>/</td>
</tr>
<tr>
<td>Period</td>
<td>.</td>
</tr>
<tr>
<td>Comma</td>
<td>,</td>
</tr>
<tr>
<td>Underscore</td>
<td>_</td>
</tr>
<tr>
<td>Hyphen</td>
<td>-</td>
</tr>
<tr>
<td>Space</td>
<td>“ &quot;</td>
</tr>
<tr>
<td>Dollar sign</td>
<td>$</td>
</tr>
<tr>
<td>Less than sign</td>
<td>&lt;</td>
</tr>
<tr>
<td>Greater than sign</td>
<td>&gt;</td>
</tr>
<tr>
<td>Number sign</td>
<td>#</td>
</tr>
</tbody>
</table>

The following BAM header fields do allow the characters listed in Table 142:

- AT in @CO
- LD in @CO
- PJ in @CO
- SP in @CO
- SO in @CO
- CO in @CO
- SD in @CO
- ID in @PG
- SM in @RG

Color-space attributes

The SAM format specification includes the attribute tags CS, CQ and CM. All BAM files support color-space reads (see Table 143).

Table 143 Color attribute tag description

<table>
<thead>
<tr>
<th>Attribute tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS</td>
<td>Color-space (CS) read. The CS field contains the original color-space read, which includes the primer base, in the orientation of the CSFASTA file. CS entries are not manipulated to be top-strand relative.</td>
</tr>
<tr>
<td>CQ</td>
<td>Color qualities. Color qualities are encoded according to the ascii-33 scheme used for the QUAL field. The orientation is the same as the orientation used for the CSFASTA file.</td>
</tr>
</tbody>
</table>
**Pairing information in a BAM file**

The BAM file that is produced by the pairing tool supports both mate-pair and paired-end protocols using the standard SAM format fields, in particular the ISIZE and FLAG fields.

### Calculation of tag names

The paired libraries use tag names to refer to members of the pair. The mate-pair libraries use F3 and R3 as the tag names. The paired-end libraries use F3 and F5 as the tag names. Use the FLAG field and information from the LB field of the read group to recapitulate tag names (see Table 144).

**Table 144 Calculating tag names**

<table>
<thead>
<tr>
<th>FLAG bit</th>
<th>Library type</th>
<th>Tag name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0040</td>
<td>LMP</td>
<td>F3</td>
</tr>
<tr>
<td>0x0080</td>
<td>LMP</td>
<td>R3</td>
</tr>
<tr>
<td>0x0040</td>
<td>PE</td>
<td>F3</td>
</tr>
<tr>
<td>0x0080</td>
<td>PE</td>
<td>F5</td>
</tr>
<tr>
<td>0x040</td>
<td>F</td>
<td>F3</td>
</tr>
</tbody>
</table>

### Proper pairs

Legacy file formats, such as *.mates, and GFF, described pairs using a three-letter category. Pairs in the AAA category correspond to the “proper pair” concept in the SAM format. The pairs reflect pairings that are not altered by a structural variation such as an inversion or deletion (see Figure 48). The BAM file field values for proper pairs are different for mate-pair and paired-end libraries:

**Mate-pair libraries**

- Strand flag is equal for both mates (both 0 or both 1).
- ISIZE is between the lower and upper limit of the insert range.
- For forward strand hits R3 POS < F3 POS.
- For reverse strand hits F3 POS < R3 POS.

**Paired-end**

- Strand flag is opposite for the mates.
- ISIZE is between the lower and upper limit of the insert range. In the case of paired-end libraries, the ISIZE might be smaller than the sum of the alignment lengths.
- F3 POS < F5 POS if F3 is on the forward strand.
- F5 POS < F3 POS if F5 is on the forward strand.
LifeScope™ Genomic Analysis Software Command Shell User Guide

Appendix C File Format Descriptions

Hard clipping of incomplete extensions

LifeScope™ Software mapping uses a seed-extend algorithm. The algorithm increases mapping throughput by matching a seed, usually 25 bp, and extending the alignment until mismatches drive down the alignment score. Many alignments do not completely cover the color-space read. Because the base-space sequence of color reads cannot be precisely known in the absence of alignment, incomplete extensions are represented as a hard-clip (H) operation in the BAM CIGAR string (see Figure 49).

Figure 48 Example of proper pairs for mate-pair and paired-end protocols

Figure 49 Example of hard clipping from a color alignment

Figure 34 shows the read in normal orientation (see the top section) and aligned in reverse orientation to the reference top strand (see the middle section). The lines below the alignment show the extent of the seed (top horizontal line) and extension (bottom horizontal line) phases of mapping. The extension only results in 42 bases of
alignment. The remaining portion of the color alignment has a number of mismatches that prevent extension. These are coded as hard-clipped regions. The CIGAR field in the BAM file is top-strand relative, so even though the hard clipping is on the end of the reversed color read, it is on the beginning of the CIGAR string.

Visualize BAM output

You can use third-party software visualization tools to view BAM files in a browser.

The Integrative Genomics Viewer (IGV) available from the Broad Institute is a visualization tool for interactive exploration of large, integrated datasets. The IGV reads BAM files directly, which allows for easy viewing and inspection of alignments against the genome (see Figure 49).

For more information, go to the following site:

www.broadinstitute.org/igv/

If you use IGV to visualize the BAM files, verify that the BAI file is present. The BAI file is the index that is built for BAM files and is a standard part of the public SAM specification. If the pairing and MaToBam tools do not automatically create the BAI file:

1. Log in to the LifeScope™ Software cluster.
2. At a command prompt, enter:
   
   samtools index <BAM file name>.bam

Indexing only works if the file is sorted in coordinate order. If the file is not sorted in coordinate order, at a command prompt, run the following command to sort the file in coordinate order:

   samtools sort <unsorted BAM name>.bam <sorted BAM name>

The UCSC Genome Browser serves as an interactive web-based microscope that allows researchers to view all 23 chromosomes of the human genome at any scale, from a full chromosome down to an individual nucleotide.

For more information, go to the Genome Browser website:

www.cbse.ucsc.edu/research/browser
Indel alignments

The alignments are included in the secondary analysis BAM file. Indel alignments are included in quality calculations and the selections of primary alignments and pairs.

Nearly all of the fields in the PAS file are represented in SAM format. However, there is no allowance for ambiguous locations in the CIGAR string or elsewhere. Ambiguity occurs when a repeat element is inserted or deleted with respect to the reference sequence. Indel alignments include a user-defined attribute (XW:Z) to specify the range of possible locations within the read for an indel in a repeat region (see Figure 51).

Referring to Figure 51, in the alignment shown of a reference sequence stretch with a color read, the deletion of a single T, which is represented as a dot in the color sequence, cannot be placed precisely because of the repeated Ts. The XW attribute would span the homopolymer region (30_35).

Table 145 PAS file column descriptions

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
</table>
| Genome position | 4294973387 | The genome position given by this formula: 
C * 232 + P – 1  
where C is the chromosome number and P is the position on that chromosome. |
The PAS file is a legacy format containing four tab-separated columns as described in Table 145. The fourth column contains the full alignment information. One example of an alignment is illustrated as follows.

The following is an example of PAS file content:

```
>600_16_579_14_Lib1_1_50_1_runName_sampleName,1_6074.26.2(17:17_17)[G20]| 1_7297.1:(44.8.0)[T02] !8B52/:
B;*%=7+'539&)4>455++60+0<9.(2
```

The alignment is in the form of a number of concatenated fields. The text represents:

```
<data source>,<R3 tag details>|<F3 tag details> R3_FASTQ F3_FASTQ
```

where the indel may be either F3 or R3, and the other tag would be the anchor, ungapped alignment. The R3_FASTQ and F3_FASTQ is the color quality strings for those tags in FASTQ base quality format. For fragment, there is only one tag:

```
<data source>,<tag details> FASTQ
```

Table 146 explains the fields in the example PAS file content.

<table>
<thead>
<tr>
<th>Field</th>
<th>Meaning</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data source fields</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>600_16_579</td>
<td>The bead ID.</td>
<td>—</td>
</tr>
<tr>
<td>14</td>
<td>the unmatched length for that bead if an ungapped extended alignment was found for that bead.</td>
<td>The value 99999 is used if there was no ungapped alignment found.</td>
</tr>
<tr>
<td>Lib1_1</td>
<td>A library indicator.</td>
<td>Deprecated.</td>
</tr>
<tr>
<td>50</td>
<td>The full read-length.</td>
<td>—</td>
</tr>
<tr>
<td>1</td>
<td>The run ID.</td>
<td>—</td>
</tr>
<tr>
<td>runName_sampleName</td>
<td>The concatenation of the run name and the sample name.</td>
<td>—</td>
</tr>
<tr>
<td><strong>Indel tag fields</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1_6074.26.2</td>
<td>A sequence 1 hit at position 6074.</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>26 is the match length with 2 mismatches.</td>
<td></td>
</tr>
</tbody>
</table>

Table 145 PAS file column descriptions

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indel size</td>
<td>–7</td>
<td>Number of bases in the indel. A negative value means a deletion, a positive value is an insertion, for example, –8 is a deletion of size 8, and 3 is an insertion of size 3.</td>
</tr>
<tr>
<td>Number of errors</td>
<td>2</td>
<td>Number of errors in the tag where the indel was found.</td>
</tr>
<tr>
<td>Alignment</td>
<td>See below.</td>
<td>Details of the alignment in the form of a number of concatenated fields.</td>
</tr>
</tbody>
</table>

Table 146 PAS file field descriptions
Bed file format

The browser extensible display (Bed) format was developed to extend the UCSC Genome Browser with user-defined tracks. Bed is used to visualize the splice and fusion junctions in the UCSC Genome browser and in the IGV browser (see figure 52 on page 511). For general documentation about the Bed format, including information about all of the Bed fields, go to:

genome.ucsc.edu/FAQ/FAQformat.html

For information about visualization software, see “Pairing information in a BAM file” on page 505.

Each line in the track defines a junction where chromStart is the smaller of the coordinates and chromEnd is the greater.

There are two blocks because a junction typically contains two exons. BlockSizes are the lengths of the exons. The block starts the beginning of the exons. When fusions on different strands or chromosomes, two lines are added to the output, with each line representing one chromosome. Different colors are used to color-code different types:
Figure 52 on page 511 shows the Upstream Hypersensitive Region (UHR) gene region displayed with the Integrative Genomics Viewer (IGV) for positions 3,530,193 to 3,548,355 of Human Chr-1. The following sections describe the tracks in Figure 52.

WIG (x2) tracks
The top two tracks show the genomic coverage using the negative strand and positive strand generated by the Bam2Wig tool (Max: 100 coverage).

BAM track
The middle track shows the alignments from the BAM file. For display purposes, reads are filtered with MAPQ threshold of 45 (a stringent filter). Bases with quality value 5 to 20 are shaded.

BED track
The fifth track shows the junctions detected by the Junction Finder (BED file). As shown in the figure, all junctions detected are “known” and so are shaded in green.

**Figure 52**  BED-IGV extended track example

**BEDGRAPH file format**

Table 147 describes the BEDGRAPH format fields. These are tab-separated text fields.

<table>
<thead>
<tr>
<th>Field name</th>
<th>Example content</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Track annotation headers</td>
<td>track type=bedGraph name=&quot;Coverage&quot;</td>
<td>Annotations used for UCSC browser display purposes.</td>
</tr>
</tbody>
</table>
**Reference file data overview**

Nearly all LifeScope™ Software analysis modules use reference sequence data and, in some cases, the modules also use reference annotations and metadata.

Reference data takes a number of forms, which are described in the following sections.

**Contig multi-fasta file**

A single, multiple-entry FASTA file where each entry corresponds to a genomic contig or chromosome.

**Single contig FASTA file**

A single-entry FASTA file containing one contig or chromosome.

**CMAP file**

A tab-delimited text file used in previous versions of the software. Each line specifies the path to a single contig FASTA file. Columns in the CMAP file provide annotation and paths to other chromosome specific support files.

**GTF file**

A genome annotation file. This file describes gene models used by the whole transcriptome and small RNA modules. It is similar in form to the GTF files downloaded from the UCSC website, but includes some changes for WT or small RNA processing.

Genomic reference files downloaded from UCSC; for example, you can download hg18 from:

http://hgdownload.cse.ucsc.edu/goldenPath/hg18/bigZips/chromFa.zip

The file can be used as a basis for the reference sequence data in LifeScope™ Software tools. A few small transformations are required to prepare the GTF files for use by LifeScope™ Software applications. See “Add new reference files” on page 547 and “Convert a GTF file” on page 550.

---

**Table 147** Description of BEDGRAPH fields

<table>
<thead>
<tr>
<th>Field name</th>
<th>Example content</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contig name</td>
<td>chr1</td>
<td>Contig name of the feature. This value comes from the sequence name [SN] in the BAM header. However, in order to be viewed in the UCSC browser, these values must be of the form chr1, chr2, etc.</td>
</tr>
<tr>
<td>Feature start</td>
<td>148</td>
<td>Start location of the feature. Zero-based, inclusive.</td>
</tr>
<tr>
<td>Feature end</td>
<td>150</td>
<td>End location of the feature. Zero-based, exclusive (half-open ranges).</td>
</tr>
<tr>
<td>Coverage</td>
<td>1</td>
<td>Depth of coverage over the given [start-end] range.</td>
</tr>
</tbody>
</table>
A GTF file is an ASCII text file composed of lines of text separated by line feed characters (UNIX-style new lines). Each line of text represents a comment or a tab delimited content about a genomic feature. A comment line always starts with a “#” character. GTF does not specify any format of structured content in comments. Comments containing parsable meta-data should precede any lines containing genomic features. For more detail about the format, refer to the following site:

http://genome.ucsc.edu/FAQ/FAQformat.html#format4

**VCF file**

The Variant Call Format, in a text format. The file contains meta-information lines, a header line, and data lines that each contain information about a position in the genome. This file format was developed as part of the 1000 Genomes project, and is described here:

http://vcftools.sourceforge.net/specs.html

Recent releases of dbSNP are available in as VCF files. An example is:


**Reference sequence data validation**

The `reference_validation.pl` script provided with LifeScope™ Software checks and corrects common issues with FASTA sequence files that can cause errors in LifeScope™ Software runs. Examples of errors generated by an uncorrected GTF file can include extra spaces and mixed case.

**Concatenation**

The contig multi-FASTA reference file can be generated by concatenating the individual chromosome files together.

**Select a reference file**

Be sure to use the correct reference file type for your requirements. The identifiers in the FASTA definition line are carried through many tools, including the BAM file generated by mapping and pairing, and the GFF files typically written by tertiary analysis tools.

For some downstream viewing and integration tools, for example, the UCSC genome browser, the identifiers can be used to connect sequence references with other sources of annotation.

**Read-set file format**

An RRS (Read Range Specifier) file, also called a read-set file, is an internal file that defines which reads data is processed in a mapping run. LifeScope™ Software automatically creates the RRS file for your mapping analysis. This format is considered an internal file format and specification for this file format can change in future versions of LifeScope™ Software.

An RRS file allows a mapping analysis to process, in a single run:

- Data from a single XSQ reads file.
- Data from multiple XSQ reads files.
- Data for a range of barcodes (a subset of XSQ file data).
- Data for a range of panels within a barcode (also a subset of XSQ file data).
The RRS file is a tab-separated file with three columns which describe the input XSQ file, the sample name, and optionally a barcode range and panel range (see Table 148 on page 514). The following rules apply to RRS files:

- One read-set (barcode) cannot be part of two samples.
- For a single analysis, mapping can be in color or base. If the mapping is in color, all XSQ files in the RRS file must have color available. If the mapping is in base, all XSQ files in the RRS file must have base available. Mapping fails if these restrictions are not met.
- For barcoded XSQ files, the RRS file can specify a range of barcodes. If some of specified barcodes are missing, mapping will log a warning for the missing barcode IDs.
- One read-set is specified per content line.
- At least one of the read-sets needs to be valid for mapping module to complete.

The requirements for RRS fields are described in Table 148. The read-set file also accepts comments, which contain a ‘#’ character at the beginning of the line.

**Table 148  Read-set file description**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
</table>
| Sample name     | Multiple read-sets with the same sample name are treated as a single sample (and are processed together).  
Allowed values: A string of alpha-numeric characters up to 255 chars longs. Spaces are not allowed.  
Examples: Human_101, Bob, Sample19  
(This sample name does not need to match any information provided on the instrument.) |
| File id         | A positive integer that distinguishes between XSQ files according to these rules:  
- Two different XSQ files must have different file IDs.  
- Two identical XSQ files must have same file ID.  
(For read-set files created with the UI, LifeScope™ Software assigns this number.) |
| read-set range  | A string specifying one or more XSQ data files and optionally also barcode and panel specifiers. The format is a URI scheme:  
(fileName?start=barcodeID.panelstart&end=barcodeID.panelend)  
fileName: either a read in the reads repository or the absolute path to an XSQ file on the Linux file system (and that file must be a valid XSQ file for which the user has read permission).  
barcodeID: integers from 1–96.  
panelstart, panelend: integers from 1–4000. These fields correspond to panels on the sequencing instrument.  
The start= section must appear before the end= section. Start barcodeID value must be <= the end barcodeID value. The panelstart value must be <= the panelend value. |
Example RRS file

The following is a simple example of an RRS file which defines an entire XSQ file as the read-set. The XSQ file in this example has been imported into the reads repository, so an absolute path to the local file system is not required.

```
#SampleName FileID ReadSets
Huref 1 reads/solid0054_20110102_RD_HuRef100_F3.xsq
```

Legacy format translation

**Table 149** and **Table 150** map the GFF file field name to the corresponding BAM file field name for header fields and alignment data.

**Table 149** GFF file header fields

<table>
<thead>
<tr>
<th>GFF field</th>
<th>Corresponding BAM field</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>reference-name</td>
<td>@SQ UR field</td>
<td>This header field contains the reference file name in the UR field.</td>
</tr>
<tr>
<td>history</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>color-code</td>
<td>—</td>
<td>Only a single color encoding is used.</td>
</tr>
<tr>
<td>primer-base</td>
<td>—</td>
<td>This field is not needed. The primer base is stored with the color read attribute.</td>
</tr>
<tr>
<td>max-num-mismatches</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>max-read-length</td>
<td>@RG LB field</td>
<td>The read lengths in the file are determined from the library type information in the LB field of the read group.</td>
</tr>
<tr>
<td>line-order</td>
<td>@HD SO field</td>
<td>The sort order field in the BAM header indicates the line order. Valid values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• None</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Coordinate</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Qname</td>
</tr>
</tbody>
</table>

**Table 150** Alignment data

<table>
<thead>
<tr>
<th>GFF field</th>
<th>Corresponding BAM field</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>seqid</td>
<td>RNAME</td>
<td>GFF files use an ordinal number to indicate the reference contig. The ordinal number could be translated to names by a header table. The BAM file uses the @SQ lines to perform a similar lookup.</td>
</tr>
<tr>
<td>source</td>
<td>@RG PL</td>
<td>The read group platform (@RG PL field) is used in a manner similar to the SOLID® GFF source.</td>
</tr>
<tr>
<td>type</td>
<td>—</td>
<td>All records are alignments.</td>
</tr>
<tr>
<td>start</td>
<td>POS</td>
<td>The 1-based, left-most, top-strand position of the alignment.</td>
</tr>
<tr>
<td>end</td>
<td>Calculated from CIGAR</td>
<td>The BAM file does not support an explicit end point. An explicit end point can be calculated from the CIGAR string.</td>
</tr>
<tr>
<td>score</td>
<td>—</td>
<td>A read quality is not stored in the BAM alignment record. Mapping and pairing quality is stored in the BAM alignment record.</td>
</tr>
</tbody>
</table>
### Legacy CMAP file format description

CMAP is a legacy file format normally used to describe the chromosomes of an organism. A CMAP file provides a mapping between:

- Individual FASTA reference files
- An integer index
- A short name

#### GFF attributes

<table>
<thead>
<tr>
<th>GFF field</th>
<th>Corresponding BAM field</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>strand</td>
<td>5th FLAG bit (query strand)</td>
<td>The fifth bit of the FLAG field indicates the strand of the alignment.</td>
</tr>
</tbody>
</table>

#### Table 150  Alignment data (continued)

<table>
<thead>
<tr>
<th>GFF field</th>
<th>Corresponding BAM field</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>at</td>
<td>QNAME</td>
<td>The adaptor type or primer set field can be calculated as described in &quot;Calculation of tag names&quot; on page 505.</td>
</tr>
<tr>
<td>b</td>
<td>SEQ</td>
<td>Base space representation of the aligned read. Bases are all uppercase. Equal signs &quot;=&quot; are not used.</td>
</tr>
<tr>
<td>c</td>
<td>2nd FLAG bit (proper pair) and calculation</td>
<td>The most commonly-used category value, AAA, corresponds to the proper pair bit in the FLAG field. Use POS, MPOS, and strand bits in the FLAG field to calculate other categories. You can find C** pairs (different contigs) by interrogating the mate reference (MRNM). The D** is indicated by the fourth FLAG bit.</td>
</tr>
<tr>
<td>g</td>
<td>CS attribute</td>
<td>The color read is stored in the CS attribute. However, unlike the 'g' GFF attribute, the color read is stored in an unaltered CSFASTA form that includes the last primer base.</td>
</tr>
<tr>
<td>mq</td>
<td>MAPQ (or SM for pairs)</td>
<td>The mapping quality is stored in the MAPQ field for fragment libraries and in the SM attribute for paired results.</td>
</tr>
<tr>
<td>o</td>
<td>CIGAR (hard-clipping)</td>
<td>The offset for an alignment can be computed by checking the number of hard clip (H) operations on the CIGAR string.</td>
</tr>
<tr>
<td>p</td>
<td>—</td>
<td>Use mapping quality for uniqueness determinations.</td>
</tr>
<tr>
<td>pq</td>
<td>MAPQ (for pairs)</td>
<td>The pairing quality is stored in the MAPQ field for paired results.</td>
</tr>
<tr>
<td>q</td>
<td>CQ</td>
<td>The CQ attribute stores color quality. The values are not represented as integers, but are represented by ascii-33 encoding.</td>
</tr>
<tr>
<td>r</td>
<td>Calculate reference color at mismatches using the samtools fillmd command.</td>
<td>CIGAR strings do not distinguish between matches and mismatches. Calculate this information using the samtools fillmd command, though in a form that is different from the rb GFF attribute.</td>
</tr>
<tr>
<td>s</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>u</td>
<td>Per-alignment mismatches supported with CM attribute</td>
<td>The CM attribute specifies the number of color mismatches for the current alignment. You can use a combination of the number of color mismatches across alignments in the BAM file to reconstruct values like the ‘u’ attribute.</td>
</tr>
</tbody>
</table>
Additional annotation is added in subsequent columns. File paths can be relative or absolute. FASTA reference files must be single entry. See Table 133 for a description of CMAP file fields.

CMAP files generated by the CNV tool contain additional information, pointing to the mappability files for each chromosome arm. See Table 134 on page 445 for a description of the Human CNV CMAP file fields.

The CMAP file may begin with one or more comment lines beginning with a pound sign (#). The comments can include file variables. Variable names use dot separated syntax and must appear immediately after the pound sign. Variable values are separated from the name by an equal sign (=) which may not be surrounded by whitespace. The following fields are currently supported:

- data.dir – You can use this optional field to define a root directory. If defined, relative file paths used in the data columns are relative to the root directory.
- Header – This optional field may be used to define column headers for the columns. Column headers should be tab-separated.

### Table 151 CMAP file parameters

<table>
<thead>
<tr>
<th>Column</th>
<th>Field name</th>
<th>Type</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Seq id</td>
<td>Non-negative integer</td>
<td>2</td>
<td>The seq id is an ordinal index of the sequence.</td>
</tr>
<tr>
<td>2</td>
<td>Seq name</td>
<td>String</td>
<td>2</td>
<td>The name of the sequence. The name is typically a chromosome name and might include X, Y, M, etc. The *.chr suffix is added to the sequence name in a number of tools to support the UCSC browser function.</td>
</tr>
<tr>
<td>3</td>
<td>Fasta file path</td>
<td>String (path)</td>
<td>—</td>
<td>The path to a FASTA record containing a single sequence entry. The path can be relative or absolute. If the path is relative, you must define the data.dir field in the comments.</td>
</tr>
<tr>
<td>4</td>
<td>Double-encoded file path</td>
<td>String (path)</td>
<td>—</td>
<td>If there is a double-encoded version of the FASTA reference file, the path is specified here. Like the FASTA file path, if a relative path is provided, data.dir must be specified.</td>
</tr>
<tr>
<td>5, 6, 7, etc.</td>
<td>Additional data</td>
<td>Any</td>
<td>—</td>
<td>Any number of additional fields can be specified after the first four columns. These fields can be integers, strings, file paths, etc. If additional fields are file paths, the rules for relative file paths apply.</td>
</tr>
</tbody>
</table>
Appendix C  File Format Descriptions
Legacy CMAP file format description
This appendix covers:

- Overview ................................................................. 519
- The XSQ Tools package ............................................. 520
- Conversion to the XSQ format .................................... 521
- Conversion from the XSQ format ............................... 525
- XSQ file splitting ................................................. 526
- XSQ index reassignment ........................................ 526
- HDF5 tools .......................................................... 527
- Internal conversion parameters .............................. 528
- Links to resources .................................................. 533
- FAQ - XSQ Tools ................................................... 533

Overview

As sequencing technologies evolve and their output capacity increases, large amounts of new sequence data are straining current data storage and analysis systems in the genomics community. One approach to mitigating this pain is to store the instrument data in a more compact format, replacing the older text-based formats — FASTQ, CSFASTA, and QUAL — with more efficient binary encoding. The Extensible Sequence (XSQ) file format has been developed to store each call and quality value in a single byte, which results in file sizes that are up to 75% smaller. There are many other benefits of this format, including embedded metadata, hierarchical structure of the file allowing simplified access to its contents, and the ability to store parallel datasets for easier integration of results.

Each new data format, however, comes with costs. Learning the intricacies of a new format can be easier with full descriptions and simplified examples. Integrating the new format with legacy data or legacy workflows can be more challenging, but this can be mitigated through the use of appropriate data format converters. The goal of the XSQ Tools package is to lower the activation energy associated with adopting the XSQ format by providing documentation, examples, and converters.

For command-line users, the XSQ Tools package is downloaded and run separately from LifeScope™ Software. The LifeScope™ Software UI handles the file conversion automatically.
Utilities in the XSQ Tools package perform these functions:

- For SOLiD® 4 System users, convert CSFASTA format files into the XSQ format required by LifeScope™ Software.
- For 5500 Series Genetic Analysis Systems users running BioScope™ Software or third-party tools, convert XSQ files into the CSFASTA format required by BioScope™ Software.
- For LifeScope™ Software users, optionally split an XSQ file into multiple XSQ files, each containing the data for one library.

More information about XSQ files and the XSQ file format is available on the SOLiD® Software Community website.

This website provides:

- The XSQ file format specification document
- Slides describing XSQ and its relation to the 5500 Series Genetic Analysis Systems
- A link to the XSQ Tools package

Table 152 lists the platforms supported by the XSQ Tools package.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Conversion type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>To XSQ</td>
</tr>
<tr>
<td>Linux® CentOS 4.7</td>
<td>Yes</td>
</tr>
<tr>
<td>Linux® CentOS 5.5</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Note: An XSQ script is included in the files installed with LifeScope™ Software. This script is for internal use only, and cannot be used convert your XSQ files.

The XSQ Tools package

Package components

The following components are included in the XSQ Tools package:

- The `convertToXSQ.sh` wrapper script, which sets paths and executes tools for converting from CSFASTA and QUAL to XSQ.
- The `convertFromXSQ.sh` wrapper script, which sets paths and executes tools for the following:
  - Convert from XSQ to CSFASTA and QUAL.
  - Convert from XSQ to FASTQ.
  - Split an indexed (barcoded) XSQ file by library into new XSQ files.
  - Filter an XSQ file.
- The `indexReassignment.sh` wrapper script, which supports library index reassignment for an XSQ file.
- Operating system-specific software libraries to support standalone executables.
- Example data sets, which show the data format using simulated reads. (These do not represent data generated by an instrument). We recommend you use the example data to verify your setup and to practice a conversion. This data is based on chromosome 6.
- Example usage scripts.
Download instructions

Refer to the release notes for the download location.

You must accept the EULA and follow the download directions on the site. After downloading the file, follow these steps to install the XSQ Tools package:

1. Move the file to an appropriate location.
2. Change directory to the tarball location.
3. Execute the command `tar-xvfz XSQ_Tools.tgz`. This creates a directory named `XSQ_Tools` with the package contents.
4. Add the `XSQ_Tools` directory to your path, with one of the following commands:
   - `bash`: `export PATH=$PATH:$tarlocation/XSQ_Tools`
   - `csh/tcsh`: `set PATH=$( $PATH:$tarlocation/XSQ_Tools )`
   where `tarlocation` is the directory containing the package tarball.

Conversion to the XSQ format

This section describes converting CSFASTA and QUAL files from earlier SOLiD® Systems to the XSQ format. This conversion is also supported in the LifeScope™ Software UI.

The `convertToXSQ.sh` script converts a pair of CSFASTA and QUAL files to an XSQ file. The command options vary by library type, as shown below:

- **To convert SOLiD® fragment data:**
  ```bash
  sh convertToXSQ.sh \
  --mode=Fragment \
  --c1=<csfastafilename> --q1=<qualfilename> \ 
  --xsqfile=<output XSQ path and filename> \ 
  --libraryName=<LibraryName> \ 
  --laneNumber=<LaneNumber> \ 
  --runStartTime="yyyy-mm-dd hh:mm:ss"
  ```

- **To convert SOLiD® mate-pair or paired-end data:**
  ```bash
  sh convertToXSQ.sh \
  --mode=LMP OR --mode=PE \ 
  --c1=<csfastafilename> --q1=<qualfilename> \ 
  --c2=<csfastafilename> --q2=<qualfilename> \ 
  --libraryInsertSizeMinimum=<Minimum insert size> \ 
  --libraryInsertSizeMaximum=<Maximum insert size> \ 
  --xsqfile=<output XSQ path and filename> \ 
  --libraryName=<LibraryName> \ 
  --laneNumber=<LaneNumber> \
  ```
To convert SOLiD® barcoded fragment data (a single barcoded fragment data set):

```
sh convertToXSQ.sh \
  --mode=BC \
  --c1=<csfastafile> --q1=<qualfile> \
  --bc1=<index csfastafile> --bq1=<index qualfile> \
  --xsqfile=<output XSQ path and filename> \
  --libraryName=<LibraryName> \
  --laneNumber=<LaneNumber> \
  --runStartTime="yyyy-mm-dd hh:mm:ss"
```

To convert SOLiD® barcoded paired-end data (a single barcoded paired-end data set):

```
sh convertToXSQ.sh \
  --mode=BCPE \
  --c1=<csfastafile> --q1=<qualfile> \
  --c2=<csfastafile> --q2=<qualfile> \
  --bc1=<index csfastafile> --bq1=<index qualfile> \
  --libraryInsertSizeMinimum=<Minimum insert size> \
  --libraryInsertSizeMaximum=<Maximum insert size> \
  --xsqfile=<output XSQ path and filename> \
  --libraryName=<LibraryName> \
  --laneNumber=<LaneNumber> \
  --runStartTime="yyyy-mm-dd hh:mm:ss"
```

Table 153 lists the options required by the `convertToXSQ.sh` script. Table 154 on page 523 lists the remaining options.

### Table 153 Required arguments for the `convertToXSQ.sh` script (to create an XSQ file)

<table>
<thead>
<tr>
<th>Option</th>
<th>Required with...</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--mode</td>
<td>All</td>
<td>Allowed values are Fragment, LMP, PE, BC, BCPE:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>Fragment</strong>: Convert fragment CSFASTA and QUAL files to XSQ format.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>LMP</strong>: Convert mate-pair CSFASTA and QUAL files to XSQ format.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>PE</strong>: Convert paired-end CSFASTA and QUAL files to XSQ format.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>BC</strong>: Convert single barcoded fragment CSFASTA/QUAL with index information to XSQ format.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>BCPE</strong>: Convert single barcoded paired-end CSFASTA/QUAL with index information to XSQ format.</td>
</tr>
<tr>
<td>--libraryName</td>
<td>All</td>
<td>Name of the library preparation.</td>
</tr>
<tr>
<td>--laneNumber &lt;arg&gt;</td>
<td>All</td>
<td>Lane number in which this sequencing sample was loaded on the flowchip.</td>
</tr>
</tbody>
</table>
## Table 153 Required arguments for the `convertToXSQ.sh` script (to create an XSQ file) (continued)

<table>
<thead>
<tr>
<th>Option</th>
<th>Required with...</th>
<th>Description</th>
</tr>
</thead>
</table>
| --runStartTime <arg>  | All              | Time when the instrument run was executed. This information is entered into the output XSQ file. Use one of these formats:  
|                       |                  | • ‘yyyy-mm-dd hh:mm:ss’ (including the quotes, either double or single).  
|                       |                  | • yyyy-mm-dd\ hh:mm:ss (escape the space with a backslash).                |
| --xsqfile <arg> or -x <arg> | All | Requested name of output XSQ file. The name must include the extension .xsq. |
| --c1 <arg>            | All SOLiD® System data | The CSFASTA input file for the first tag.                                    |
| --q1 <arg>            | All SOLiD® System data | The QUAL input file for the first tag.                                       |
| --c2 <arg>            | LMP, PE, or BCPE SOLiD® System data | The CSFASTA input file for the second tag.                                    |
| --q2 <arg>            | LMP, PE, or BCPE SOLiD® System data | The QUAL input file for the second tag.                                       |
| --bc1 <arg>           | BC or BCPE SOLiD® System data | The index (barcoded) CSFASTA input file.                                     |
| --bq1 <arg>           | BC or BCPE SOLiD® System data | The index (barcoded) QUAL input file.                                        |
| libraryInsertSizeMinimum | LMP or PE SOLiD® System data | The expected minimum insert size, in nucleotides.                           |
| libraryInsertSizeMaximum | LMP or PE SOLiD® System data | The expected maximum insert size, in nucleotides.                           |

Table 154 lists options for the `convertToXSQ.sh` script. See Table 153 on page 522 for the required options.

## Table 154 Optional arguments for the `convertToXSQ.sh` script (to create an XSQ file)

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| -a <arg> --application <arg> | The type of analysis intended for this input data. Allowed values are listed below. Quotes are required around values that contain spaces.  
|                     | • Whole genome resequencing  
|                     | • Targeted resequencing  
|                     | • Whole transcriptome (Fragment)  
|                     | • Whole transcriptome (PairedEnd)  
|                     | • Small RNA  
|                     | • ChiP-Seq  
|                     | • Methylation                                                             |
Table 154 Optional arguments for the `convertToXSQ.sh` script (to create an XSQ file) (continued)

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--assembly &lt;arg&gt;</code></td>
<td>The intended assembly for mapping. May be blank if unknown. Examples:</td>
</tr>
<tr>
<td></td>
<td>• hg18</td>
</tr>
<tr>
<td></td>
<td>• hg19</td>
</tr>
<tr>
<td></td>
<td>• mm9</td>
</tr>
<tr>
<td></td>
<td>Other assembly names are allowed. Spaces are not allowed in these names.</td>
</tr>
<tr>
<td><code>-c &lt;arg&gt;</code></td>
<td>Use the configuration file <code>&lt;arg&gt;</code> as a list of conversion parameters. The</td>
</tr>
<tr>
<td></td>
<td>parameters are specified as nested key value pairs. Example file content:</td>
</tr>
<tr>
<td></td>
<td>q1=/data/results/reads/uhr.150.F3.qual</td>
</tr>
<tr>
<td></td>
<td>cl=/data/results/reads/uhr.150.F3.csfasta</td>
</tr>
<tr>
<td></td>
<td>x=./out.xsq</td>
</tr>
<tr>
<td></td>
<td>mode=Fragment</td>
</tr>
<tr>
<td></td>
<td>runStartTime=2011-04-01 \ 12:01:02</td>
</tr>
<tr>
<td></td>
<td>libraryName=150_UHR</td>
</tr>
<tr>
<td></td>
<td>Option hyphens ('-', '--') are not used in the configuration file.</td>
</tr>
<tr>
<td><code>--comments</code></td>
<td>Additional information about this sample or its preparation.</td>
</tr>
<tr>
<td><code>-d &lt;arg&gt;</code></td>
<td>--description &lt;arg&gt;</td>
</tr>
<tr>
<td></td>
<td>A description of the sample.</td>
</tr>
<tr>
<td><code>-e</code></td>
<td>--ercc</td>
</tr>
<tr>
<td></td>
<td>Whether internal controls have been used. Reserved for future use.</td>
</tr>
<tr>
<td><code>--flowCellAssignment &lt;arg&gt;</code></td>
<td>The position of the instrument flowcell in which this flowchip was placed.</td>
</tr>
<tr>
<td><code>-h</code></td>
<td>--help</td>
</tr>
<tr>
<td></td>
<td>Display command usage and options.</td>
</tr>
<tr>
<td><code>--instrumentName &lt;arg&gt;</code></td>
<td>The name of the instrument.</td>
</tr>
<tr>
<td><code>--instrumentSerial &lt;arg&gt;</code></td>
<td>The unique serial number identifying the instrument.</td>
</tr>
<tr>
<td><code>-k &lt;arg&gt;</code></td>
<td>--indexKitName &lt;arg&gt;</td>
</tr>
<tr>
<td></td>
<td>The name of the kit used to add the indexing nucleotides.</td>
</tr>
<tr>
<td><code>--laneNumber &lt;arg&gt;</code></td>
<td>Lane number in which this sequencing sample was loaded within the</td>
</tr>
<tr>
<td></td>
<td>flowchip layout.</td>
</tr>
<tr>
<td><code>--libraryName &lt;arg&gt;</code></td>
<td>The name of library preparation.</td>
</tr>
<tr>
<td><code>-n</code></td>
<td>--noqualfile</td>
</tr>
<tr>
<td></td>
<td>Use this attribute to specify that the QUAL file is not available.</td>
</tr>
<tr>
<td><code>-o &lt;arg&gt;</code></td>
<td>--operator &lt;arg&gt;</td>
</tr>
<tr>
<td></td>
<td>The name or ID of person who set up the instrument run.</td>
</tr>
<tr>
<td><code>-p &lt;arg&gt;</code></td>
<td>--projectName &lt;arg&gt;</td>
</tr>
<tr>
<td></td>
<td>Project within which this sample is being run. This value is entered into</td>
</tr>
<tr>
<td></td>
<td>the projectName field in the libraryDetails table, and may be different from</td>
</tr>
<tr>
<td></td>
<td>the LifeScope™ Software project name.</td>
</tr>
<tr>
<td><code>--runEndTime &lt;arg&gt;</code></td>
<td>Time when the instrument run completed. This information is entered into</td>
</tr>
<tr>
<td></td>
<td>the output XSQ file. Use one of these formats:</td>
</tr>
<tr>
<td></td>
<td>• 'yyyy-mm-dd hh:mm:ss' (including the quotes, either double or single).</td>
</tr>
<tr>
<td></td>
<td>• yyyy-mm-dd\ hh:mm:ss (escape the space with a backslash).</td>
</tr>
<tr>
<td><code>--runName &lt;arg&gt;</code></td>
<td>The name of a flowchip run.</td>
</tr>
<tr>
<td><code>--sampleIdentifier &lt;arg&gt;</code></td>
<td>The unique name for this sample.</td>
</tr>
<tr>
<td><code>--sampleOwner &lt;arg&gt;</code></td>
<td>The owner of this sample.</td>
</tr>
<tr>
<td><code>-sequencingSampleDescription &lt;arg&gt;</code></td>
<td>Description of the sample loaded in a lane on a flowchip.</td>
</tr>
<tr>
<td><code>--sequencingSampleName &lt;arg&gt;</code></td>
<td>The name of the sample loaded in a lane on a flowchip.</td>
</tr>
</tbody>
</table>

LifeScope™ Genomic Analysis Software Command Shell User Guide
Conversion from the XSQ format

This section describes converting XSQ format files from the 5500 Series Genetic Analysis Systems into CSFASTA and QUAL files, in the format required by earlier version of BioScope™ Software. This section does not apply to LifeScope™ Software users who are analyzing XSQ data files generated by the 5500 Series Genetic Analysis Systems.

The conversion program is available as a standalone script to be run on the Linux shell. This conversion is not available within the LifeScope™ Software UI or the LifeScope™ Software command shell.

If the input XSQ file includes base-space data, the conversion also exports the base-space data into a FASTQ file.

Conversion syntax

The convertFromXSQ.sh script converts an XSQ file into a pair of CSFASTA and QUAL files. The usage for this script is:

```
convertFromXSQ.sh -c xsqfile
```

where xsqfile gives an existing XSQ file on the file system.

The converted files are created in the following directory:

```
/Library/<LibraryName>/<TagName>/reads/
```

The output files are named according to the following pattern:

```
<xsqfile>_<LibraryName>_<TagName>.ext
```

where ext has the following values:

- .csfasta
- .QV.qual
- .fastq

FASTQ files are created only if the input XSQ file contains base-space data.

Filter option

Use the -f option to filter (remove) marked reads during conversion, with this version of the command:

```
convertFromXSQ.sh -c -f xsqfile
```

In the XSQ file, each read contains a field of filtering flags, which may mark the read for filtering. Only when the -f option is used, reads that are marked for filtering are not written to the CSFASTA file.
By default `convertFromXSQ.sh` ignores the filtering flags and writes all reads to the converted output files. With the `-f` option, the converter applies the filtering flags during conversion, which suppresses the filtered reads and quality values from being written in the output files.

### Options table

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-c</code></td>
<td>Converts the input XSQ file into CSFASTA, QUAL, and FASTQ files.</td>
</tr>
<tr>
<td><code>-f</code></td>
<td>(Optional) During conversion, removes reads that are marked for filtering.</td>
</tr>
<tr>
<td><code>-o</code></td>
<td>(Optional) Specifies an output directory partial path.</td>
</tr>
<tr>
<td><code>-s</code></td>
<td>Splits the input XSQ file into multiple output files, one for each library in the parent XSQ file.</td>
</tr>
<tr>
<td><code>-v</code></td>
<td>Displays the package version.</td>
</tr>
</tbody>
</table>

### XSQ file splitting

The `convertFromXSQ.sh` script with the `-s` option splits an input XSQ file into multiple XSQ files, one file per library. For example, if a paired-end library is created with indices for each of lib1, lib2, and lib3 and run in the same lane, a single XSQ file will be created. Using the `-s` option creates three separate XSQ files - one for each of lib1, lib2, and lib3. Each XSQ file contains data for both the F3 and F5 tags.

To split an XSQ file, use the following command:

```
convertFromXSQ.sh -s xsqfile -o outputpath
```

The following output files are produced:

```
outputpath/
  lib1.xsq
  lib2.xsq
  ...
```

Only one input XSQ file is allowed per run.

### XSQ index reassignment

#### Purpose

The `indexReassignment.sh` script supports correcting the index information in an XSQ file (for example, in case incorrect information is entered on the instrument). The process involves these steps:

1. Run the `indexReassignment.sh` script to write a comma-separated values (CSV) file.
2. Manually edit the CSV file to enter the correct index name.
3. Run the `indexReassignment.sh` script to read the CSV file and create an XSQ file with the corrected index information.
The `indexReassignment.sh` script also supports optionally changing the mismatch level.

### Usage

This section describes the usage of the `indexReassignment.sh` script.

- **To extract** the index information:
  
  ```
  indexReassignment.sh -e -xsq example.xsq
  ```
  
  This command creates a CSV file with the same file name as the input XSQ file but with a `.csv` extension. (You then manually edit the CSV file to correct the index information, before running `indexReassignment.sh` to update the information.)

- **To update** with the corrected index information:
  
  ```
  indexReassignment.sh -csv example.csv -xsq example.xsq \
  -o example-fixed.xsq
  ```
  
  This command creates a new XSQ file with the corrected index information. The options have the following meaning:
  
  - **-csv <arg>**: The CSV file edited to correct the index.
    
    **Note**: You create the CSV file during the `indexReassignment.sh` extraction step above, and manually edit it to correct the index.
  
  - **-xsq <arg>**: The original XSQ file.
  
  - **-o <arg>**: The output XSQ file, newly-created with the corrected index information.

- **To change** the mismatch level while updating the index:
  
  ```
  indexReassignment.sh -csv example.csv -xsq example.xsq \
  -o example-fixed.xsq -mm n
  ```
  
  The `-mm n` option changes the update command to also update the mismatch level. This option redoes the entire indexing assignment and rewrites all library groups. (Allowed values with the `-mm` option are 0 and 1.)

### Index reassignment options table

Table 156 lists the options supported by the `indexReassignment.sh` script.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-csv &lt;arg&gt;</code></td>
<td>Update the XSQ file index, reading index information from the CSV file <code>&lt;arg&gt;</code>.</td>
</tr>
<tr>
<td><code>-e</code></td>
<td>Extract current index information.</td>
</tr>
<tr>
<td><code>-mm &lt;arg&gt;</code></td>
<td>Update the mismatch level for index classification.</td>
</tr>
<tr>
<td><code>-o &lt;arg&gt;</code></td>
<td>The output XSQ file, created with the corrected index information.</td>
</tr>
<tr>
<td><code>-xsq &lt;arg&gt;</code></td>
<td>The input XSQ file.</td>
</tr>
</tbody>
</table>

### HDF5 tools

The XSQ format conforms to the HDF5 specification, and HDF5 tools work on XSQ files. Figure 53 is an example of how HDFView displays an XSQ file, showing Groups, Tables, and Datasets.
Internal conversion parameters

This section describes the parameters used internally by the LifeScope™ Software UI when converting a file to the XSQ format from the older CSFASTA and QUAL formats used in earlier versions of the software. These parameters are for internal use.

Parameters table  Table 157 lists and describes the conversion parameters used internally by the LifeScope™ Software UI.

Table 157  Internal XSQ conversion parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xsqconverter.run</td>
<td>—</td>
<td>For internal use.</td>
</tr>
</tbody>
</table>
Table 157  Internal XSQ conversion parameters (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>
| xsq.application         | —       | Describes how this sequence is intended to be analyzed. Allowed values are:  
|                         |         | • Whole genome resequencing  
|                         |         | • Targeted resequencing  
|                         |         | • Whole transcriptome (Fragment)  
|                         |         | • Whole transcriptome (PairedEnd)  
|                         |         | • Small RNA  
|                         |         | • ChiP-Seq  
|                         |         | • Methylation  |
| xsq.assembly            | —       | The intended assembly for mapping. May be blank if unknown. Example values:  
|                         |         | • hg18  
|                         |         | • hg19  
|                         |         | • mm9  |
| xsq.comments            | —       | Additional information about this sample or its preparation.  |
| xsq.description         | —       | Sample description.  |
| xsq.ercc                | 0       | Whether or not internal controls have been used. Allowed values:  
|                         |         | • 0: No  
|                         |         | • 1: Yes  |
| xsq.flowCellAssignment  | 255     | The position of the instrument flowcell in which this flowchip was placed.  |
| xsq.indexKitName        | —       | Name of the kit used to add the indexing nucleotides.  |
| xsq.input.barcedCsfasta1| —     | Barcoded CSFASTA file (used only in the case of indexed runs).  |
| xsq.input.barcedQual1   | —       | Barcoded QUAL file (used only in the case of indexed runs).  |
| xsq.input.csfasta1      | —       | Input CSFASTA file.  |
| xsq.input.csfasta2      | —       | Second CSFASTA file (use only in the case of LMP or PE runs).  |
| xsq.input.qual1         | —       | Input QUAL file.  |
| xsq.input.qual2         | —       | Second QUAL file (use only in the case of LMP or PE).  |
| xsq.instrumentName      | —       | Name of the sequencing instrument.  |
| xsq.instrumentSerial    | —       | Unique serial number to identify a particular sequencing instrument.  |
| xsq.laneNumber          | 255     | Lane number within the flowchip layout in which this sequencing sample was loaded.  |
| xsq.libraryInsertSizeMaximum| — | Expected maximum insert size in nucleotides. (Required for LMP and PE conversion).  |
| xsq.libraryInsertSizeMinimum| — | Expected minimum insert size in nucleotides. (Required for LMP and PE conversion).  |
| xsq.libraryName          | —       | The name of library preparation.  |
## Run parameter

```
xsqconverter.run=1
```

## How this sequences is intended to be analyzed

```
xsq.application=Unknown
```

## Intended assembly for mapping

```
xsq.assembly=Unknown
```

### Internal XSQ conversion parameters

Table 157 Internal XSQ conversion parameters *(continued)*

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xsq.mode</td>
<td>—</td>
<td>XSQ mode of operation. Allowed values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fragment: Convert fragment CSFASTA and QUAL formats to XSQ.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• LMP: Convert mate-pair CSFASTA and QUAL formats to XSQ.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• PE: Convert paired-end CSFASTA and QUAL formats to XSQ.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• BC: Converting single barcoded fragment CSFASTA and QUAL formats with index information to XSQ.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• BCPE: Convert single barcoded paired-end CSFASTA and QUAL formats with index information to XSQ.</td>
</tr>
<tr>
<td>xsq.noqualfile</td>
<td>false</td>
<td>Specify that the input QUAL file is not available. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• false: The QUAL file is available.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• true: The QUAL file is missing.</td>
</tr>
<tr>
<td>xsq.operator</td>
<td>—</td>
<td>The name of person or ID who set up the instrument run.</td>
</tr>
<tr>
<td>xsq.output.xsqfile</td>
<td>—</td>
<td>The requested name for output XSQ file. Must use the &quot;*.xsq&quot; extension.</td>
</tr>
<tr>
<td>xsq.projectName</td>
<td>—</td>
<td>Project within which this sample is being run.</td>
</tr>
<tr>
<td>xsq.runEndTime</td>
<td>—</td>
<td>Time when the instrument run completed. Required format: 'yyyy-mm-dd hh:mm:ss' (including quotes).</td>
</tr>
<tr>
<td>xsq.runName</td>
<td>—</td>
<td>Name of a flowchip run.</td>
</tr>
<tr>
<td>xsq.runStartTime</td>
<td>—</td>
<td>Time when the instrument run started. Required format: 'yyyy-mm-dd hh:mm:ss' (including quotes).</td>
</tr>
<tr>
<td>xsq.sampleIdentifier</td>
<td>—</td>
<td>Unique name for this sample.</td>
</tr>
<tr>
<td>xsq.sampleOwner</td>
<td>—</td>
<td>Owner of this sample.</td>
</tr>
<tr>
<td>xsq.sequencingSampleDescription</td>
<td>—</td>
<td>Description of the sample loaded in a lane on a flowchip.</td>
</tr>
<tr>
<td>xsq.sequencingSampleName</td>
<td>—</td>
<td>Name of sequencing sample loaded in a lane on a flowchip. This value is the bead emulsion loaded on the instrument. The emulsion may be a collection of barcoded libraries from multiple samples.</td>
</tr>
<tr>
<td>xsq.species</td>
<td>—</td>
<td>The Intended species for mapping. Example values include:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Homo sapiens</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Mus musculus</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Other</td>
</tr>
</tbody>
</table>

### Internal INI file

The following is an example of an INI file used internally by the LifeScope™ Software UI during a conversion to the XSQ format from the older CSFASTA and QUAL formats used in earlier versions of the software. Refer to the LifeScope™ Genomic Analysis Software User Guide (Part no. 4476537) for information about the UI. This INI file is for internal use.

```
##Run parameter
xsqconverter.run=1

##How this sequences is intended to be analyzed
xsq.application=Unknown

##Intended assembly for mapping
xsq.assembly=Unknown
```
## Additional information about this sample or its preparation
xsq.comments=Unknown

## Sample description
xsq.description=Unknown

## Whether internal controls have been used (0=No, 1=Yes)
xsq.ercc=0

## The position of the instrument flowcell in which this flowchip was placed
xsq.flowCellAssignment=255

## Name of the kit used to add the indexing nucleotides
xsq.indexKitName=Unknown

## Barcoded csfasta file (should be used only in case of indexed runs)
xsq.input.barcodedCsfastal=

## Barcoded qual file (should be used only in case of indexed runs)
xsq.input.barcodedQual1=

## Csfasta file
xsq.input.csfastal=

## Second csfasta file (should be used only in case of LMP or PE)
xsq.input.csfastac2=

## Qual file
xsq.input.qual1=

## Second qual file (should be used only in case of LMP or PE)
xsq.input.qual2=

## Name of instrument
xsq.instrumentName=Unknown

## Unique serial number to identify a particular instrument
xsq.instrumentSerial=Unknown

## Lane number within the flowchip layout in which this sequencing sample was loaded
xsq.laneNumber=255

## Expected maximum insert size in nucleotides. (Required for LMP and PE conversion)
xsq.libraryInsertSizeMaximum=

## Expected minimum insert size in nucleotides. (Required for LMP and PE conversion)
xsq.libraryInsertSizeMinimum=
# The name of library preparation
xsq.libraryName=DefaultLibrary

# Allowed values are: Fragment, LMP, PE, BC, BCPE. Use 'Fragment' for converting fragment CSFASTA/QUAL to XSQ. Use 'LMP' for converting mate pair CSFASTA/QUAL to XSQ. Use 'PE' for converting paired end CSFASTA/QUAL to XSQ. Use 'BC' for converting single barcoded fragment CSFASTA/QUAL with index information to XSQ. Use 'PE' for converting single barcoded paired end CSFASTA/QUAL with index information to XSQ
xsq.mode=Fragment

##Specify that qual file is not available (Use 'true' - to specify qual file missing)
xsq.noqualfile=false

##The name of person/ID who set up the instrument run
xsq.operator=Unknown

##Desired name for output XSQ file
xsq.output.xsqfile=pluginfrag.xsq

##Project within which this sample is being run
xsq.projectName=Unknown

##Time when run completed. The run end time should be in 'yyyy-mm-dd hh:mm:ss' format
xsq.runEndTime=

##Name of a flowchip run
xsq.runName=Unknown

##Time when run was executed. The run start time should be in 'yyyy-mm-dd hh:mm:ss' format
xsq.runStartTime=2011-01-11 20:10:20

##Unique name for this sample
xsq.sampleIdentifier=Unknown

##Owner of this sample
xsq.sampleOwner=Unknown

##Description of the sample loaded in a lane on a flowchip
xsq.sequencingSampleDescription=Unknown

##Name of sample loaded in a lane on a flowchip
xsq.sequencingSampleName=Unknown

##Intended species for mapping
xsq.species=Unknown
Links to resources

This section lists resources related to XSQ file format and converters.

For the XSQ Tools package and documentation, XSQ file format specification, XSQ webinar slides, and example XSQ files, refer to the release notes for the current location.

HDFView is a Java® tool for browsing HDF-based files, including XSQ files. For information about HDFView, visit this site:

http://www.hdfgroup.org/hdf-java-html/hdfview/

HDF5 is a technology suite for managing of large, complex data collections. HDF5 APIs are available at this site:

http://www.hdfgroup.org/HDF5/release/obtain5.html

Exact Call Chemistry (ECC) is a unique ligation-based sequencing methodology, introduced with the 5500 Series Genetic Analysis Systems. The Exact Call Chemistry white paper is available through this link and the instructions that follow:


1. Under the Publication & Literature section, click the Product Literature tab.
2. Scroll down to the White Papers section.
3. Click on the “SOLiD® System Accuracy with the Exact Call Chemistry Module” link.

FAQ - XSQ Tools

1. What software accepts the XSQ file format?

Initially, only LifeScope™ Software supports the new format. Life Technologies Corporation is working with third-party developers to adapt their workflows to support the new chemistry and data format.

2. I have pipelines that require CSFASTA and QUAL files as input. What should I do?

Life Technologies Corporation provides tools on the SOLiD® Software Tools website to convert XSQ files into CSFASTA and QUAL files. See “Links to resources” on page 533, and download the XSQ Tools package.
3. I have pipelines that require FASTQ files as input. What should I do?

When the ECC module is used during sequencing, base-space data is available in the XSQ file and can be exported into a FASTQ file. See “Conversion from the XSQ format” on page 525.

4. Can I use data from both a SOLiD® 4 System and a 5500 Series Genetic Analysis Systems for data analysis?

Yes. There are two options and both involve converting the data into a common format, either XSQ or CSFASTA+QUAL. One way is to convert the SOLiD® 4 System CSFASTA and QUAL output to the XSQ format (see “Conversion to the XSQ format” on page 521). Then supply both the converted XSQ file and the 5500 XSQ file as input to your LifeScope™ Software analysis.

Another method is to convert the XSQ data into CSFASTA+QUAL for use with older analysis tools.

5. Is there a converter to change CSFASTA and QUAL files to the XSQ format?

Yes, standalone converters are provided in the XSQ Tools package. Refer to the release notes for the location of the XSQ package.

6. How does the XSQ format handle multiplexing?

Multiple libraries may be included in a single XSQ file, such as are generated for each lane of a SOLiD® 5500 Series Sequencer run. When multiple different users are providing samples to be run in the same lane, the resulting file may be split by library for distribution to individual users, without sharing all of the data with each user. The `convertFromXSQ.sh` script provides the `-s` splitting option to separate libraries into separate XSQ files when necessary. See “XSQ file splitting” on page 526.

**Note:** The `-s` splitting option only works for indexing files. Non-indexing files already contain only a single library (and are not split with the `-s` option).

The `convertToXSQ.sh` script is run once for each barcode index, and each run results in a single XSQ file. Multiple barcode indices cannot be joined in a single XSQ file with the current converter.

7. Are APIs available for accessing XSQ data?
Yes. HDF5 APIs work with the XSQ format, and are available at this site:

http://www.hdfgroup.org/HDF5/release/obtain5.html

8

Is an XSQ file viewer available?

Yes. XSQ data is can be viewed with the HDFView browser, which can be obtained at this site:

http://www.hdfgroup.org/hdf-java-html/hdfview/

9

Can XSQ files be converted in the LifeScope™ Software UI and command shell?

Yes. The LifeScope™ Software UI supports the conversion of CSFASTA and QUAL files to the XSQ format. If you are running the LifeScope™ Software command shell, use the standalone XSQ conversion tool that is part of the XSQ Tools package.

10

Is library index correction supported?

Not in the current version. An XSQ library index reassignment tool is under consideration.

11

Why does the converter not work on my variant of Linux.

The table in the “Overview” on page 519 section describes that conversion to XSQ requires CentOS 4.7 or later, and conversion from XSQ requires CentOS 5.5 or later. Other versions of Linux are not supported.

12

What are the reserved quality values (QVs) in the XSQ file?

From the XSQ File Format Specification:

For the QV, the range of the 6 bits is between 0 and 63. Valid quality values will be from 3 to 62, represented as PHRED scores \[QV = -10\log_{10}(P(error))\]. 0 and 1 are reserved for future usage; 2 indicates a missing quality value (not trimming); 63 refers to an uninformative or missing call.
13

What values should I use for libraryInsertSizeMinimum and libraryInsertSizeMaximum when they are not known during the conversion of mate-pair and paired-end files?

Any values will work as long as libraryInsertSizeMinimum is less than or equal to libraryInsertSizeMaximum and both values are nonnegative.

14

How can I collect statistics on an XSQ file?

Not supported currently. An XSQ Statistics utility to extract the number of reads by file, barcode, tag, etc., in addition to library type and read length, is under consideration.

15

How can I tell if my XSQ file is valid or corrupt?

Not supported currently. An XSQ validator utility is under consideration.

16

I have a CSFASTA file to convert to XSQ format, but I do not have the QUAL file. Can I create the XSQ file?

When converting CSFASTA to XSQ, if the QUAL file is missing or not found, all reads are marked for filtering in the output XSQ file. As a result, the reads are ignored (not processed) when used as input into LifeScope™ Software analyses.

The workaround is to create a dummy QUAL file to use as input with the convertToXSQ.sh script. Create a QUAL file with the same bead ids (and in the same order) as the CSFASTA file. Use QV values at least high enough to pass default thresholds.

18

Where does the tool get the indexing kit information? Is it stored in the XSQ file or does it come from the XSQ_Tool package?

This information is distributed within XSQ Tools. See XSQ_Tools/xsqreindexer/conf for details.
19

If a wrong kit was used to assign the original XSQ file, can the tool reassign it to a different, correct kit? For example, does the tool support reassignment from a 16-index kit to a 96-index kit?

No. Kit reassignment is not supported here because the index calls are of a different length (5 vs. 10 calls).

20

The 5500 ICS v1.0 and v1.1 only uses 4 colors of the 5-color 16 barcode kit. Can the index reassignment tool use all 5 colors for reassignment?

Currently the index reassignment tool is using the same underlying library, so for now the same issue is present for the 16 index library. This is not a problem with the 96 index libraries.

21

How can I tell if an XSQ file was generated from an ECC run?

As this is a property of each tag and may be different between tags, the IsBasePresent attribute is most informative. See /RunMetadata/TagDetails/<TagName>/IsBasePresent in the XSQ file.
This appendix covers:

- Overview .................................................. 539
- Repository structure ....................................... 539
- Repository location ......................................... 540
- Reference file conflicts .................................... 540
- Initial repository contents ................................ 541
- Add new reference files .................................... 547
- Prepare reference files ..................................... 550

Overview

This appendix describes the LifeScope™ Genomic Analysis Software reference repository. The repository is created during the pre-installation and installation process, using files shipped on a separate data drive.

You can add new reference files to the repository at any time after installation, but you must follow restrictions and conventions described in this appendix.

Repository structure

The files on disk that make up the reference repository have the following top-level organization:

- lifetech – Contains files that have been prepared or formatted to support customizations within LifeScope™ Software
- external – Contains files that have been downloaded and placed here verbatim, such as annotations from public repositories.
- internal – Intended to hold locally derived files to support new genomes and annotations. The internal structure mirrors the lifetech directory.

The next directory level are assembly names such as hg18 and hg19.

For more detail see “Initial repository contents” on page 541.
Repository location

The location (path) of the reference repository is set during the pre-installation and installation process. The location is determined by where the data drive contents are copied, and must match the location answered to the installer question “Enter the location for reference files.”

The reference repository location after installation is given by the parameter lifescape.references in the file <installdir>/server/server.properties. For example:

```bash
# Define reference directory
lifescape.references=/data/results/referenceData
```

The location /data/results is an example only. The reference repository location is set during “Copy data drive content” on page 41 in Chapter 2, LifeScope™ Genomic Analysis Software Installation, and is the location <LSDF>/referenceData, where <LSDF> is the LifeScope™ Software Data Folder that the LifeScope™ Software administrator creates during the “Copy data drive content” step.

Reference file conflicts

In order to avoid conflicts with reference files, follow the instructions in “Add new reference files” to create a new folder when adding references for a new genome to the repository.

The behavior for a set reference assembly_name command, such as set reference hgl9, is the following:

- LifeScope™ Software first looks in the reference repository for the assembly_name folder, in the lifetech, internal, and external directories, in that order. The search stops when the first match is found.
- The software searches in the assembly_name/reference folder for a genomic reference files that includes the assembly name in its file name.
- In the assembly_name folder, the software searches for reference files that include the assembly name in its file name.
- If a file with the assembly name is not found, or if multiple files are found that both have the assembly name and have the expected file extension, the software then uses the first file (alphabetically) in that folder.

If you do not follow the recommended conventions for adding reference files to the repository, the reference selected may or may not be the one intended.

Assembly names and reference parameters

Users are recommended to use the assembly name option with the set reference command, when possible. Using a recognized assembly name automates the specification of reference-related parameters. See Table 13 on page 81 for more information on the set reference command.

Before starting your analysis, review the reference-related parameters to confirm that they are correctly set. (Run the LifeScope™ Software shell ls command in the analysis directory to list the parameter settings.)
Initial repository contents

The initial contents of the repository are created from the data drive during the installation process (“Copy data drive content” on page 41).

The following tables describe the files and directories that make up the initial reference repository:

- lifetech/hg19 – Table 158
- lifetech/hg18 – Table 159
- external/hg18 and hg19 – Table 160
- external/uc01 – Table 161
- internal/uc01 – Table 162

See “Add new reference files” on page 547 to add new reference files to the repository.

**Note:** Modifying files under the lifetech directory of the reference repository is not supported. Header and other information in the reference file is case sensitive. For example, changing the case of a contig name in a reference file causes program failures. Adding files under the lifetech directory also is not supported. Changing the information in any respect is not supported.

Table 159 describes the reference files in the lifetech/hg19 directory.

<table>
<thead>
<tr>
<th>lifetech/hg19 file or directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hg19/human_filter_reference.fasta</td>
<td>The file human_filter_reference.fasta is a collection of sequences that are used to filter out commonly occurring motifs and contaminating sequences prior to mapping.</td>
</tr>
</tbody>
</table>

See “Add new reference files” on page 547 to add new reference files to the repository.
### The Reference Repository

**Initial repository contents**

<table>
<thead>
<tr>
<th>lifetech/hg19 file or directory</th>
<th>Description</th>
</tr>
</thead>
</table>
| **hg19/reference**               | This directory contains a validated FASTA file describing the reference sequence for human hg19: human_hg19.fa  
An index file with the following name has also been created to describe each entry: human_hg19.fa.fai  
This file contains five columns, with one row for each entry:  
1. Chromosome or contig name (no spaces)  
2. Chromosome or contig length  
3. Cumulative chromosome or contig length  
4. Number of characters per line  
5. Line width (including carriage return)  
The other files are all derivatives of the original file and contain the hash tables used for mapping:  
human_hg19.fa.10011101011110110100111101  
human_hg19.fa.101100111110111101  
human_hg19.fa.101101110111011101  
human_hg19.fa.110100111011011001111  
human_hg19.fa.1110100111010011101001  
human_hg19.fa.11101001110100111010011  
human_hg19.fa.111011101101111011  
human_hg19.fa.1111110101110101101 |
| **hg19/refGene**                 | RefGene files are downloaded directly from UCSC, from this website:  
[http://hgdownload.cse.ucsc.edu/goldenness/hg19/database/refGene.txt.gz](http://hgdownload.cse.ucsc.edu/goldenness/hg19/database/refGene.txt.gz)  
These files have this filename pattern: refGene.\$(ORGAMISM).\$(DATE).gtf.  
RefGene files are converted to GTF (preserving the filename pattern):  
refGene.hg19.20101221.gtf  
A refGene file is then used, in conjunction with the reference FASTA file,  
referenceData/lifetech/hg19/reference/human_hg19.fa  
to generate new FASTA files for mapping transcripts to exons and junctions:  
referenceData/lifetech/hg19/refGene/refGene.hg19.20101221.exon.fa  
referenceData/lifetech/hg19/refGene/refGene.hg19.20101221.junctions.fa |
| **hg19/dbSNP/dbSNP_b132_00-All.vcf** | This file is from the National Center for Biotechnology Information (NCBI) dbSNP database, which contains information on SNPs and indels already found by other studies. Recent releases of dbSNP are available in VCF (Variant Call Format), such as this file for Human:  
The VCF file format was developed as part of the 1000 Genomes project. The format is described at this website:  
[http://vcftools.sourceforge.net/specs.html](http://vcftools.sourceforge.net/specs.html) |
Table 158 Contents and purpose of lifetech/hg19 reference files and directories (continued)

<table>
<thead>
<tr>
<th>lifetech/hg19 file or directory</th>
<th>Description</th>
</tr>
</thead>
</table>
| hg19/mirbase/hsa.gff            | The microRNA database annotations are derived directly from the miRBase website: [http://www.mirbase.org/ftp](http://www.mirbase.org/ftp)  
The species-specific precursor GFF file is downloaded from this page: [http://www.mirbase.org/ftp.shtml](http://www.mirbase.org/ftp.shtml)  
which resolves into a file such as this (for human): [ftp://mirbase.org/pub/mirbase/CURRENT/genomes/hsa.gff](ftp://mirbase.org/pub/mirbase/CURRENT/genomes/hsa.gff) |
| hg19/CNV/mappability/fragment_50.h5 | The CNV hg19 mappability file. |

Table 159 describes the reference files in the lifetech/hg18 directory.

Table 159 Contents and purpose of lifetech/hg18 reference files and directories

<table>
<thead>
<tr>
<th>lifetech/hg18 file or directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hg18/human_filter_reference.fasta</td>
<td>The file human_filter_reference.fasta is a collection of sequences that are used to filter out commonly occurring motifs and contaminating sequences prior to mapping.</td>
</tr>
</tbody>
</table>
| hg18/reference | This directory contains a validated FASTA file describing the reference sequence for human hg18: human_hg18.fa  
An index file with the following name has also been created to describe each entry: human_hg18.fa.fai  
This file contains five columns, with one row for each entry:  
1. Chromosome or contig name (no spaces)  
2. Chromosome or contig length  
3. Cumulative chromosome or contig length  
4. Number of characters per line  
5. Line width (including carriage return)  
The other files are all derivatives of the original file and contain the hash tables used for mapping:  
human_hg18.fa.111011101101111011  
human_hg18.fa.10110111011101101  
human_hg18.fa.11010011101001110100111  
human_hg18.fa.101100111110111101  
human_hg18.fa.1110100111010011101001  
human_hg18.fa.11101001110100111010011  
human_hg18.fa.1001110100111010011101  
human_hg18.fa.1111110101110101101 |
## Initial repository contents

**Appendix E**  
The Reference Repository

RefGene files are downloaded directly from UCSC, from this website:  
http://hgdownload.cse.ucsc.edu/goldenPath/hg18/database/refGene.txt.gz  
These files have this filename pattern: refGene.$(ORGANISM).$(DATE).gtf.  
RefGene files are converted to GTF (preserving the filename pattern):  
refGene.hg18.20090513.gtf  
A refGene file is then used, in conjunction with the reference FASTA file,  
referenceData/lifetech/hg18/reference/human_hg18.fa  
to generate new FASTA files for mapping transcripts to exons and junctions:  
referenceData/lifetech/hg18/refGene/refGene.hg18.20090513.exon.fa  
referenceData/lifetech/hg18/refGene/refGene.hg18.20090513.junctions.fa

**hg18/dbSNP/dbSNP_b130.tab**  
This file is from the National Center for Biotechnology Information (NCBI) dbSNP database, which contains information on SNPs and indels already found by other studies. This file contains the data from dbSNP version 130, and has the following tab-separated fields:  
1. rsID, the refSNP identifier  
2. Chromosome number  
3. Chromosome start coordinate  
4. Chromosome end  
5. Is SNP?  
6. Function code (consequence to transcript if any)  
The function code is a comma separated list of pairs: locus_id:function code. The FunctionCode attribute is valid only for SNPs and small indels, and are used to annotate SNPs. Codes are based on functional codes from dbSNP, or refSNPs in gene features (from the dbSNP handbook).

**hg18/mirbase/hsa.gff**  
The microRNA database annotations are derived directly from the miRBase website:  
http://www.mirbase.org/ftp  
The species-specific precursor GFF file is downloaded from this page:  
http://www.mirbase.org/ftp.shtml  
which resolves into a file such as this (for human):  
ftp://mirbase.org/pub/mirbase/CURRENT/genomes/hsa.gff

**hg18/CNV/mappability/**  
fragment_25.h5  
fragment_50.h5  
matepair_50x50.h5  
The CNV hg18 mappability files.
Table 161 describes the hg18 and hg19 reference files in the external directory.

<table>
<thead>
<tr>
<th>external file or directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hg18/ucsc/refGene.txt.gz,</td>
<td>RefGene files are downloaded directly from UCSC, from these websites:</td>
</tr>
<tr>
<td>hg19/ucsc/refGene.txt.gz</td>
<td><a href="http://hgdownload.cse.ucsc.edu/goldenPath/hg18/database/refGene.txt.gz">http://hgdownload.cse.ucsc.edu/goldenPath/hg18/database/refGene.txt.gz</a></td>
</tr>
<tr>
<td></td>
<td><a href="http://hgdownload.cse.ucsc.edu/goldenPath/hg19/database/refGene.txt.gz">http://hgdownload.cse.ucsc.edu/goldenPath/hg19/database/refGene.txt.gz</a></td>
</tr>
<tr>
<td>hg18/targetedEnrichment/</td>
<td>Targeted Regions files are specific to each target capture procedure used during the</td>
</tr>
<tr>
<td>targetRegionsHg18.bed</td>
<td>sample preparation process before the library generation step. As there are many</td>
</tr>
<tr>
<td>hg19/targetedEnrichment/</td>
<td>ways to enrich for targeted regions, including custom preparations, these files are to</td>
</tr>
<tr>
<td>targetRegionsHg19.bed</td>
<td>be provided by the user. This directory is the default location for such files, and BED</td>
</tr>
<tr>
<td></td>
<td>format is typically used here.</td>
</tr>
<tr>
<td></td>
<td>Here are example locations and filenames:</td>
</tr>
<tr>
<td></td>
<td>external/hg18/targetedEnrichment/targetRegionsHg18.bed</td>
</tr>
<tr>
<td></td>
<td>external/hg19/targetedEnrichment/targetRegionsHg19.bed</td>
</tr>
<tr>
<td></td>
<td>The BED file format was developed to support the UCSC Genome Browser, and the</td>
</tr>
<tr>
<td></td>
<td>format is described here:</td>
</tr>
<tr>
<td></td>
<td><a href="http://genome.ucsc.edu/FAQ/FAQformat.html#format1">http://genome.ucsc.edu/FAQ/FAQformat.html#format1</a></td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The target regions BED files on the data drive are 0-length placeholders.</td>
</tr>
<tr>
<td>hg18/mirbase/miRNA.txt,</td>
<td>The microRNA database annotations are derived directly from the miRBase website:</td>
</tr>
<tr>
<td>hg19/mirbase/miRNA.txt</td>
<td><a href="http://www.mirbase.org/ftp">http://www.mirbase.org/ftp</a></td>
</tr>
<tr>
<td></td>
<td>The species-specific mature forms are downloaded together as an Excel document:</td>
</tr>
<tr>
<td></td>
<td>ftp://mirbase.org/pub/mirbase/CURRENT/miRNA.xls</td>
</tr>
<tr>
<td></td>
<td>and written as text files on the data drive:</td>
</tr>
<tr>
<td></td>
<td>referenceData/external/hg18/mirbase/miRNA.txt</td>
</tr>
<tr>
<td></td>
<td>referenceData/external/hg19/mirbase/miRNA.txt</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The shell command set workflow hg18 (or hg19) incorrectly sets the</td>
</tr>
<tr>
<td></td>
<td>analysis.mirbase.mature.file parameter. You must manually set the</td>
</tr>
<tr>
<td></td>
<td>parameter correctly after the set workflow command.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The hsa.gff files are included in the lifetech/hg18/mirbase and</td>
</tr>
<tr>
<td></td>
<td>lifetech/hg19/mirbase directories.</td>
</tr>
</tbody>
</table>

Table 161 describes the uc01 example reference files in the external directory.

<table>
<thead>
<tr>
<th>external uc01 file or directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbSNP/b131_SNPChrPosOnRef_37_1.bcp</td>
<td>This set of three files is available from the NCBI dbSNP database, which</td>
</tr>
<tr>
<td>dbSNP/b131_SNPCntigLoc_37_1.bcp</td>
<td>contains information on SNPs and indels already found by other studies.</td>
</tr>
<tr>
<td>dbSNP/b131_SNPCntigLocusId_37_1.bcp</td>
<td>These files contain the data from dbSNP version 131, and are available from</td>
</tr>
<tr>
<td></td>
<td>this website:</td>
</tr>
<tr>
<td></td>
<td>Three files are used together and all three are needed.</td>
</tr>
</tbody>
</table>
Table 161 Contents and purpose of external uc01 reference files and directories (continued)

<table>
<thead>
<tr>
<th>external uc01 file or directory</th>
<th>Description</th>
</tr>
</thead>
</table>
| uc01/targetedEnrichment/ targetRegionsUc01.bed | Targeted Regions files are specific to each target capture procedure used during the sample preparation process before the library generation step. As there are many ways to enrich for targeted regions, including custom preparations, these files are to be provided by the user. This directory is the default location for such files, and BED format is typically used here.

Here is an example location and filename:

external/hg18/targetedEnrichment/targetRegionsHg18.bed

The BED file format was developed to support the UCSC Genome Browser, and the format is described here:

http://genome.ucsc.edu/FAQ/FAQformat.html#format1

**Note:** The target regions BED files on the data drive are 0-length placeholders.

| uc01/mirbase/ | The microRNA database annotations are derived directly from the miRBase website:
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><a href="http://www.mirbase.org/ftp">http://www.mirbase.org/ftp</a></td>
</tr>
<tr>
<td></td>
<td>which resolves into a file such as this (for human):</td>
</tr>
<tr>
<td></td>
<td>ftp://mirbase.org/pub/mirbase/CURRENT/genomes/hsa.gff</td>
</tr>
<tr>
<td></td>
<td>The mature forms are downloaded together as an Excel document:</td>
</tr>
<tr>
<td></td>
<td>ftp://mirbase.org/pub/mirbase/CURRENT/miRNA.xls</td>
</tr>
</tbody>
</table>

Table 162 describes the uc01 example reference files in the internal directory. The uc01 is an example assembly. Copy this directory structure to add your own locally-derived assembly to the reference repository, and replace the string uc01 with the name of your new assembly.

Table 162 Contents and purpose of internal uc01 reference files and directories

<table>
<thead>
<tr>
<th>internal uc01 file or directory</th>
<th>Description</th>
</tr>
</thead>
</table>
| reference/unicorn_uc01.fa       | This directory is for a validated FASTA file describing the genomic reference sequence for uc01: uc01.fa
| reference/unicorn_uc01.fa.fai   | (Optional) An index file with the following name can be created to describe each entry: uc01.fa.fai
|                                 | This file contains five columns, with one row for each entry:
|                                 | 1. Chromosome or contig name (no spaces)
|                                 | 2. Chromosome or contig length
|                                 | 3. Cumulative chromosome or contig length
|                                 | 4. Number of characters per line
|                                 | 5. Line width (including carriage return) |
Add new reference files

You use the Linux file system, not LifeScope™ Software shell commands, to add new reference files. Depending on the local Linux file system configuration, you may need admin permissions to add new reference files to the repository.

WARNING! Do not add new files to the lifetech directory or modify files under lifetech directory. Instead use the internal and external folders, as described below.

Add by assembly name

The reference repository is organized by subfolders for each assembly name. The choice of folder name is important because the folder name matches the assembly name and is used with the set reference assembly command to automate setting reference-related parameters.

The internal and external folders are available for your new reference files. Follow this convention:

- external – Use for files that have been downloaded from public repositories and placed here verbatim.
- internal – Use for locally-derived files to support new genomes and annotations.

Example references at internal/uc01 and external/uc01, in the reference repository location, provide a structure to be followed for new references.

Follow these steps to add a new reference file to the reference repository:

1. Find the location of the reference repository on the local Linux file system.
2. If necessary, create a new subfolder under referenceData/internal (or external) for the new reference being added. Use a unique assembly name as the name for the new subfolder.
   
   Mimic the structure of the example references at referenceData/internal/uc01, or referenceData/external/uc01, in the reference repository location, for your new references.
   
   For example, to add references for an example assembly named uc02, create the following directory: referenceData/internal/uc02 or referenceData/external/uc02.
   
3. Follow the instructions in “Prepare reference files” to validate and convert your new reference files.

Example steps to add a new assembly

This section lists example steps to add a new assembly or a new species to the reference repository. This example describes how to add the rat assembly rn4 to the repository. In most installations, file permissions require that the LifeScope™ Software administrator perform these steps.

Repository location and directories

1. Identify the reference repository location, which in the installation instructions is <LSDF>/referenceData.
2. Under <LSDF>/referenceData, create these directories for the new assembly:
**FASTA file**

1. Find a FASTA file describing the reference sequence for rat, for instance from an NCBI, UCSC, or Ensembl site.
2. Copy the rat FASTA file to a local directory.
3. Validate the FASTA file with the `reference_validation.pl` script (see “Validate new reference files” on page 550).
4. Copy or move the validated FASTA file to `external/rn4/reference`, for instance, as the file: `external/rn4/reference/rat_rn4.fa`

**RefGene file**

1. Find a RefGene file for the rat assembly. RefGene files are downloaded directly from UCSC, from this website: [http://hgdownload.cse.ucsc.edu](http://hgdownload.cse.ucsc.edu)
   
   The files have this filename pattern: `refGene.$\{ASSEMBLY\}.{$\{DATE\}}.gtf`. Convert the RefGene file to GTF, using the `refgene2gff.sh` described in “Convert a GTF file” on page 550. The output is in the same filename pattern: `refGene.rn4.20110517.gtf`

2. This step is only required for whole transcriptome analyses. In conjunction with the reference FASTA file, the WT exon extractor and junction extractor modules generate new FASTA files for mapping transcripts to exons and junctions: `external/rn4/reference/rat_rn4.fa`

   Copy the new FASTA files to:
   - `external/rn4/refGene/refGene.rn4.20110517.exon.fa`
   - `external/rn4/refGene/refGene.rn4.20110517.junctions.fa`

**Regions of interest file**

This section is required for targeted resequencing, low frequency variant, and other enrichment analyses.

1. Find or develop a regions of interest file, in BED format (see “BED file format” on page 510).
2. Copy the regions of interest file to the following location and file name:
   - `external/rn4/targetedEnrichment/targetRegionsRn4.bed`
miRBase files
This section is only required for small RNA analyses.

1. Find the species-specific microRNA database annotations (precursor file) on the miRBase website:
   http://www.mirbase.org/ftp.shtml

2. Download the mature forms together as an Excel document:
   ftp://mirbase.org/pub/mirbase/CURRENT/miRNA.xls

3. Save the Excel document as a text file in this location and file name:
   external/rn4/mirbase/miRNA.txt

Filter reference file

1. Find or develop a filter reference file.

2. Copy the filter reference file to a local directory.

3. Validate the filter reference file with the reference_validation.pl script (see “Validate new reference files” on page 550).

4. Copy or move the validated FASTA file to external/rn4, for instance, as the file:
   external/rn4/rat_filter_reference.fa

Genomic annotations

Annotation source files such as dbSNP are only required to run the LifeScope™ Software genomic annotations module.

1. See “Annotation sources” on page 292 in Chapter 15, “Add Genomic Annotations to Analysis Results” on page 287 for information on obtaining these files.

2. Copy the dbSNP and other annotation source files to the following directory:
   external/rn4/dbSNP

Shell command

When you configure your LifeScope™ Software command shell analysis, use the following shell command to specify your new reference files:

```
set reference rn4
```

This command sets the following parameters with default values for rn4:

- annotation.dbsnp.file
- annotation.gtf.file
- analysis.mirbase.precursor.file
- analysis.mirbase.mature.file
- analysis.filter.reference
- analysis.assembly.name
- analysis.species.name
Appendix E  The Reference Repository

Prepare reference files

You must perform these procedures on new reference files that you add to the reference repository, before you can use them in a LifeScope™ Software analysis. Each of these steps is described below.

1. Validate the reference file to a format that complies with LifeScope™ Software.
2. Convert new GTF files and refGene files that are required for your analyses.
3. Concatenate the reference files into a single file, if you have more than one, such as when the sequence is separated into single chromosome files.

Prepare reference files

Note: If an assembly of the same name exists in the reference repository lifetech directory, the software finds the lifetech assembly first. In this case you must set the reference parameters individually.

Validate new reference files

Reference validation is not performed automatically. You must validate the reference files that you add to the reference repository so that reference files are presented in the correct format to each LifeScope™ Software module that uses a reference file.

To validate a reference file, follow these steps:

1. Log in to the LifeScope™ Software cluster. Be sure that you log in to Linux with a user name that has execute (‘x’) privileges on the directory (<installdir>/bin) that contains the reference_validation.pl script. The script is under the bin folder where the LifeScope™ Software is installed.
2. Navigate to the directory that contains the reference_validation.pl file.
3. At a command prompt, enter:
   `<installdir>/bin/reference_validation.pl -f <InputFasta.ext>
   `If the input FASTA file contains IUB codes, the default is to replace the IUBs with Ns. Preserving IUBs is not recommended for use with some tertiary modules. To keep IUBs unchanged, use the -i (keep IUBs) option:
   `reference_validation.pl -i TRUE -f <InputFasta.ext>
   `The output file is named `<InputFasta>_validated.fa`.

Convert a GTF file

Not all species have exactly the same format of GTF file. Before using a GTF file from a public source with a LifeScope™ Software module, you must convert the GTF file into a format that is compatible with LifeScope™ Software.

Convert an Ensembl GTF file

The Ensembl website has GTF-formatted genome annotations available for many popular assemblies. Ensembl GTF files are properly normalized by gene and transcript IDs. The site is at:

www.ensembl.org
Ensembl GTF files use gene accession numbers instead of HUGO-style gene names. Ensembl GTF files also use unprefixed sequence identifiers, such as 1,2,3,...,X,Y,MT. The Ensembl GTF files are incompatible with genome reference FASTA files that have UCSC-style sequence IDs with the prefix “chr”, for example, chr1, chr2, chr3,...,chrX, chrY, chrM.

**Reformat a Ensembl GTF file**

1. Log in to the cluster where LifeScope™ Software is installed.
2. Reformat the GTF file with the reformat_ensembl_gtf.pl command, and redirect the output to a new file with .reformatted in the file name.
   
   For Example:
   ```
   reformat_ensembl_gtf.pl Homo_sapiens.GRCh37.63.gtf > Homo_sapiens.GRCh37.63.reformatted.gtf
   ```

   The input GTF file name typically includes the NCBI build number and the Ensmbl release number in its name, according to the following pattern:
   ```
   Homo_sapiens.<reference_ncbi_build_no>.<Ensembl_release_no>.gtf
   ```

**Convert a refGene.txt.gz file**

The UCSC Genome Browser has genome annotations available for many assemblies at

```
hgdownload.cse.ucsc.edu/goldenPath/
```

The GTF-formatted annotations available for download are not properly normalized by gene ID. The required content is present for each assembly in the file export of the refGene database table `database/refGene.txt.gz`.

For example, annotation for human genome build 18 is available at:

```
hgdownload.cse.ucsc.edu/goldenPath/hg18/database/refGene.txt.gz
```

**Note:** The GTF-formatted annotation is not in the GTF format required by LifeScope™ Genomic Analysis Software. You must convert the annotation before using it in WTA.

Run the `bin/refgene2gff.sh` script to convert the refGene.txt.gz file:

```
gunzip refGene.txt.gz
refgene2gff.sh -i refGene.txt -o refGene.gff
```

Genome annotations that are downloaded from the UCSC Genome Browser and converted by the annotation conversion script are optimal because they contain Human Genome Organization (HUGO)-style gene names. HUGO-style gene names allow interpretation when using a genome browser or reading reports.

The annotation conversion script works with the latest format of `refGene.txt` files. Assemblies, such as the rat genome, use an alternative format for the `refGene.txt` file. The `refgene2gff.sh` script does not convert alternative formats.
Appendix E  The Reference Repository

Prepare reference files
Command Shell Control Files

This appendix covers:

- Overview ................................................................. 553
- Configuration files ...................................................... 554
- Set up your own analysis run ........................................ 558
- LifeScope™ Software parameters ................................... 559
- (Optional) The Linux mail command ............................... 561

Overview

This chapter describes the configuration files that LifeScope™ Genomic Analysis Software uses to control its execution. These files control which LifeScope™ Software modules are executed and their order of execution, and also define the run-time parameters that customize each module’s behavior.

When you use the LifeScope™ Software UI or the LifeScope™ Software command shell, the configuration files are automatically generated for you.

The following types of control files are used with your LifeScope™ Software analyses:

- **Read-only module XML files** – Contain the parameters accepted for each module and default settings for those parameters. The XML files, one per module, are in the `<installdir>/etc/plugins/pipelines` directory.
  
  Example: `<installdir>/etc/plugins/pipelines/dibayes.xml`

- **Module properties files** – Define settings used in all runs for a particular module. These files are located in `<installdir>/etc/plugins/properties`.
  
  Example: `<installdir>/etc/plugins/properties/ dibayes.properties`

- **A global.ini file** – Defines settings which by convention are imported into other INI files in the current directory only. For example, in a workflow tertiary directory, the other INI files in that tertiary directory import `global.ini` and all its parameters. In the case of conflicting settings, the setting in the `global.ini` files takes precedence over settings in the `<installdir>` module properties file.
  
  Example: `<installdir>/etc/workflows/small.rna/tertiary/ global.ini`

- **LifeScope™ Software modules' INI files** – Define settings for a single module run. In the case of conflicting settings, the module INI file setting takes precedence over settings in both your `global.ini` file and the `<installdir>` module properties file.
  
  Example: `<installdir>/etc/workflows/ targeted.resequencing.frag/tertiary/dibayes.ini`
Configuration files

PLN files

A PLN file defines your analysis through a list of tasks, which is a list of one or more INI or PLN files. For each listed INI file, LifeScope™ Software executes an analysis run of the corresponding module.

The following is an example of a simple PLN file, which causes a SNPs module analysis to be run:

diBayes.ini

Tasks (INI or PLN files) are listed one per line. When there are multiple tasks listed, every task without a dependency is executed in parallel (assuming there are sufficient compute nodes).

Dependencies

A PLN file specifies a directed acyclic graph of module dependencies. The less-than sign (\(<\)\) signifies a dependency. A dependency means that task on the left of the less-than sign is executed only if the task on the right of the sign completes successfully.

For example, the following is a typical workflow PLN file:

secondary.pln
tertiary.pln < secondary.pln

This PLN file specifies that tertiary.pln is dependent on secondary.pln. First the modules specified in secondary.pln are executed. Only if those modules completed successfully, then the modules specified in tertiary.pln are executed.

A task with no dependencies takes this form in a PLN file:

x.ini

Multiple dependencies are added after the less-than sign, separated by commas:

x.ini < a.ini, b.ini, c.ini

Forward references are not allowed in the dependency list, and a task cannot list itself as a dependency.

Other syntax

Blank lines and comment lines are ignored in PLN files. Comment lines must begin with a “#” as the first character.

On error

When one module in a sequence fails, the remaining modules specified in the PLN file continue. However, any module that depends in the failed module is not attempted.
INI files

Module INI files

Module INI files are the main source of configuration parameters requiring your customization for LifeScope™ Software runs. A module INI file defines run-time parameters which control the behavior of the module’s algorithm.

Many examples of module INI files are included in the LifeScope™ Software standard workflows, under the following directories:

- `<installdir>/etc/workflows/*/secondary`
- `<installdir>/etc/workflows/*/tertiary`

The following is example content of a module INI file, in this case the CNV INI file:

```ini
import global.ini

##Run parameter
cnv.run=1

cnv.mappability.dir=${analysis.mappability.dir}

##Size of the window block to be considered as a region for writing coverage output. Mean coverage of all bases in each of these windows will be written
#cnv.coverage.wsize=1000

##The maximum p-value for the regions to be shown as copy number deletions
#cnv.deletions.max.pval=1.0

##The maximum ratio between the coverage of the region and the expected coverage, for the region to be called as cnv deletion
#cnv.deletions.max.ratio=0.5

##The minimum mappability percentage for the regions to be shown as copy number deletions
#cnv.deletions.min.mappability=50

##The minimum number of windows for the regions to be shown as copy number deletions
#cnv.deletions.min.windows=2

##To set the ploidy of all chromosomes for Humans
#cnv.gender=Male

##The maximum p-value for the regions to be shown as copy number insertions
#cnv.insertions.max.pval=1.0

##The minimum mappability percentage for the regions to be shown as copy number insertions
#cnv.insertions.min.mappability=10

##The minimum ratio between the coverage of the region and the expected coverage, for the region to be called as cnv insertion
#cnv.insertions.min.ratio=1.25
```
## The minimum number of windows for the regions to be shown as copy number insertions
#cnv.insertions.min.windows=2

## To indicate whether genome wide normalization or local normalization should be performed. false for genome wide and true for chromosome-arm local normalization
#cnv.local.normalization=0

## The minimum quality value of the alignments
#cnv.min.quality=2

## The general ploidy of the genome
#cnv.ploidy=2

## List of {contig id: ploidy of the contig} of all the contigs whose ploidy is different to the general ploidy of the genome
#cnv.ploidy.exception=None

## To set the stringency setting for calling CNVs
#cnv.stringency.setting=Medium

## The distance in kilo bases to be trimmed from the extreme ends of the chromosome arms
#cnv.trim.distance=1000

## The size of the window block to be considered as a region
#cnv.window.size=5000

## To indicate whether coverage files should be output or not. Enter 1 to write coverage output in *.wig format. Keep the default value if you do not want to output coverage
#cnv.write.coverage=0

**Note:** The parameters for CNV are described in Table 45 on page 211 in Chapter 11, Run a Human CNVs Analysis.

### Import command

The module INI file typically imports the global.ini file. The effect of the import statement is to include the content of the global.ini file, with all its parameters and defined values, into the module INI file.

**Note:** An imported file must be in the same directory as the INI file. No pathnames are allowed with INI file import statement.

### .run parameters

Within a INI file, the *.run parameters control which modules are run. For example, these .run parameters appear in INI files:

```
saet.run
bamstats.run
```
The run parameters are set automatically by the shell during standard workflows. Accept the default for most use.

The allowed values for .run parameters are 0 and 1:

- 0: Do not run the module.
- 1: Run the module analysis (default).

Because mapping is a prerequisite for all other runs, mapping.run may not be set to 0 unless it has already been successfully run to completion. You may set the .run parameters for other modules to 0, under these conditions:

- You do not want to run that module analysis.
- That module analysis has already been completed (for instance, if you are restarting the workflow run and repeating a particular module is not necessary).

When you run in the project repository, the LifeScope™ Software command shell resume command does not repeat modules that have successfully completed.

The global.ini file

The global.ini file defines parameters for one or more INI files.

If a parameter in the global.ini also appears in an analysis module's INI file, then the value set in the global.ini file is ignored and the analysis module's INI's definition is used. The import global.ini statement must be placed at the top of the an analysis module's INI file.

Paths are not allowed in the import global.ini statement. The global.ini file must be in the same directory as the analysis module INI file that imports it.

Properties files

A properties file sets the default parameters used by every run of its module type, for all users of the LifeScope™ Software installation. Properties files are located in the directory <installdir>/etc/plugins/properties.

For example, the file fragment.mapping.properties contains the settings for genomic mapping runs. Example contents for the fragment.mapping.properties file are shown below.

```
wall.time=120
fragmap.number.of.nodes=4
mapping.np.per.node=8
fragmap.minreads.per.node=4000000
fragmap.maxreads.per.node=150000000
fragmap.max.number.of.nodes=100
mapping.memory=15gb
java.heap.space=1500
```

Parameter values defined in a .properties file apply to all runs of that type of module, except that parameter definitions in a INI file override the value set in the properties file. The relationship between the two types of files is explained in “Property files vs. INI files”.
Property files vs. INI files

Parameter values set in a properties file affect all runs of that module, but may be reset in the module INI file.

Modifying a setting in the properties file affects all future runs of that module, by all users. Modifying the setting in the fragment.mapping.ini file affects only the current run and any subsequent runs using the INI file. Settings present in the fragment.mapping.ini file take precedence over settings in the properties file. Table 163 compares the influence of parameters in a .properties file and an INI file. Mapping module files are used as an example to demonstrate the general principle relating .properties files and INI files.

Table 163 Comparing module properties and INI files (using mapping as an example)

<table>
<thead>
<tr>
<th>File and location</th>
<th>Coverage</th>
<th>Behavior on parameter conflicts</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;installdir&gt;/etc/plugins/properties/</td>
<td>Affects all mapping runs.</td>
<td>Values may be overridden by settings in</td>
</tr>
<tr>
<td>mapping.properties</td>
<td></td>
<td>fragment.mapping.ini.</td>
</tr>
<tr>
<td>your global.ini file</td>
<td>Affects all runs (of all modules) if an</td>
<td>Overrides settings in mapping.properties.</td>
</tr>
<tr>
<td></td>
<td>INI file imports this global.ini file.</td>
<td>Values may be overridden by settings in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>fragment.mapping.ini.</td>
</tr>
<tr>
<td>your fragment.mapping.ini file</td>
<td>Affects all fragment mapping runs that use</td>
<td>The parameter value defined in</td>
</tr>
<tr>
<td></td>
<td>this INI file.</td>
<td>fragment.mapping.ini is used, overriding</td>
</tr>
<tr>
<td></td>
<td></td>
<td>settings from the mapping.properties and</td>
</tr>
<tr>
<td></td>
<td></td>
<td>global.ini files.</td>
</tr>
</tbody>
</table>

Analysis module XML files

In the directory <installdir>/etc/plugins/pipelines are module XML files, one per module, which list all parameters accepted by each module, along with the parameters’ default values. Use the XML files as a reference for allowed parameters and their default values.

Do not edit these XML files in order to change a parameter’s value. Instead, modify one of these two files, depending on your purpose:

- **The module’s properties file** – To change a parameter for all subsequent module runs for all users, edit the parameter in the module’s properties file under <installdir>. See “Properties files” on page 557.
- **Your module INI file** – To change a parameter for only your own runs of a module, edit the parameter in your module’s INI file. This change affects only analyses that use that INI file. See “PLN files” on page 554.

In either of the two cases above, you may add a parameter that did not previously exist in the file, only if that parameter appears in the module’s XML file.

Set up your own analysis run

The most direct way to customize your own LifeScope™ Software analysis is to copy an existing workflow. You can then change the workflow to execute your preferred module or sequence of modules. See “Create a new workflow” on page 113 and “Run a standard workflow” on page 110 for more information.

The optional examples download provides single-module analyses. See Appendix H, “Demo Analyses” on page 567 for information on these analyses.
The shell `get workflow` command copies a workflow’s INI files, PLN files, and their directory structure to the local file system. These files are copied to the directory you are in when you open the shell with the `lscope.sh` command. You can modify the INI and PLN files to customize your own analysis. See Table 13 on page 81 and “Create a new workflow” on page 113.

### LifeScope™ Software parameters

This section describes parameters the software uses to control your analysis runs. When you run an analysis in the command shell, these parameters can be set automatically by the software. If you create your own INI files, you often need to specify some of these parameters (depending on the type of analysis you are running).

See also “Analysis space” on page 69 for a description of the `analysis.space` parameter.

#### General parameters

Table 164 lists parameters that LifeScope™ Software sets when you run an analysis in the command shell.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>analysis.input.readset.file</code></td>
<td>An RRS [Read Range Specifier] file built by the system specifying the XSQ read-sets being analyzed (only applies to secondary analysis). RRS is an internal file format.</td>
</tr>
<tr>
<td><code>analysis.genome.reference</code></td>
<td>Path to the genome reference file.</td>
</tr>
<tr>
<td><code>analysis.sample.name</code></td>
<td>User-defined sample name.</td>
</tr>
<tr>
<td><code>analysis.input.bams</code></td>
<td>Comma-separated paths to input BAM files (for tertiary analysis).</td>
</tr>
</tbody>
</table>

#### Reference parameters

Table 165 lists reference-related parameters that LifeScope™ Software sets for you when you use the `set reference` command with a recognized assembly name.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>annotation.dbsnp.file</code></td>
<td>The file used to annotate SNPs and small indels. The initial reference repository includes dbSNP_b130.tab [hg18] and dbSNP_b132_00-All.vcf [hg19].</td>
</tr>
<tr>
<td><code>annotation.gtf.file</code></td>
<td>The file containing gene and exon annotations corresponding to the genome assembly used in the analysis.</td>
</tr>
<tr>
<td><code>analysis.filter.reference</code></td>
<td>The file containing a collection of sequences that are used to filter out commonly-occurring motifs and contaminating sequences prior to mapping.</td>
</tr>
<tr>
<td><code>analysis.mirbase.mature.file</code></td>
<td>The file containing mature form microRNA database annotations.</td>
</tr>
<tr>
<td><code>analysis.mirbase.precursor.file</code></td>
<td>The file containing precursor microRNA database annotations.</td>
</tr>
<tr>
<td><code>analysis.assembly.name</code></td>
<td>The name of the genome assembly used in current analysis. Examples: hg18, hg19</td>
</tr>
</tbody>
</table>
Before starting your analysis, confirm that the reference-related parameters are correctly set. After the set reference command, run the LifeScope™ Software shell ls command to review the parameter settings (both commands are executed in the LifeScope™ Software command shell, in the analysis virtual directory).

**Note:** In the default reference repository, the precursor file hsa.gff is not located under the external directories. The file is included as referenceData/lifetech/hg19/mirbase/hsa.gff.

Table 166 lists a framework parameter, analysis.regions.file, that you must explicitly set (in both the GUI and the command shell). The reference repository has placeholders for the targeted regions of interest files, but you must both supply the regions of interest file and explicitly set the analysis.regions.file parameter. The framework does not set this parameter automatically with the set workflow command. The GUI does not enforce that this parameter be set.

The regions is interest file is required with targeted resequencing and low frequency variant workflows, and with any other analysis that involves the enrichment module.

### Table 166 Regions of interest file parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>analysis.regions.file</td>
<td>Full path to your regions of interest file.</td>
</tr>
</tbody>
</table>

### Execution control parameters

Table 167 lists parameters used for job management.

### Table 167 Job management parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>task.temp.dir.delete</td>
<td>Toggles cleanup of intermediate files created in the temp directory.</td>
</tr>
<tr>
<td>task.scratch.dir.delete</td>
<td>Toggles cleanup of intermediate files created in the scratch directory.</td>
</tr>
<tr>
<td>scratch.dir</td>
<td>Location of the scratch directory.</td>
</tr>
<tr>
<td><em>.</em>.run</td>
<td>Parameters that toggle the execution of analysis modules. Examples:</td>
</tr>
<tr>
<td></td>
<td>• cnv.run</td>
</tr>
<tr>
<td></td>
<td>• dibayes.run</td>
</tr>
</tbody>
</table>

### Read-only parameters

Table 168 lists read-only parameters that you can use in your INI files to set other parameters. Do not set any of these parameters.

### Table 168 Read-only parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>analysis.dir</td>
<td>The base analysis directory.</td>
</tr>
<tr>
<td>analysis.name</td>
<td>The name of the base analysis.</td>
</tr>
</tbody>
</table>
The LifeScope™ Software command shell protects your currently running analyses as well as the configuration and definition of both your projects and your analyses. The running jobs and the configuration are not affected if you close your LifeScope™ Software shell and exit your Linux session. You do not need to use Linux commands such as screens or nohup in order to keep your analysis or session window open.

The UNIX mail command can be used to send an email notification of a script’s completion. Your UNIX mail system must be configured properly. Add the following to your lscope.sh run or lscope.sh resume command:

```
&& mail -s "subject" <email_address>
```

where subject is the subject of the email message to be sent and email_address is the email account to be sent the completion message. You must follow UNIX command conventions. Keep the subject string short and avoid special characters.

An example of the complete command is:

```
lscope.sh run my.pln -u user && mail -s "done" me@xyz.com
```
(Optional) The Linux mail command
This appendix covers:
- Overview .............................................................. 563
- Admin tasks ............................................................. 563
- Admin interfaces ...................................................... 565

Overview

This appendix describes administrative functions that can be performed on either in the Linux shell or the LifeScope™ Software command shell.

Most administrative functions performed in the LifeScope™ Software UI, with the GUI Admin Portal. Refer to the LifeScope™ Genomic Analysis Software User Guide (Part no. 4476537) for more information.

Under some conditions, you can use the LifeScope™ Software command shell (or Linux shell) for all administrative functions, and avoid using the GUI Admin Portal. You do not need the GUI Admin Portal if both of these are true for your installation:

- You select the “host” authentication realm during installation.

See “Authentication realm” on page 42, in Chapter 2, LifeScope™ Genomic Analysis Software Installation, for information on authentication realms.

Admin tasks

This section describes typical admin tasks such as user management, scratch space cleanup, reference repository management, and resetting the admin password.

User management

User management tasks vary with the authentication realm used for your installation:

- LifeScope – Users are created in the LifeScope™ Software user repository. User account management (including account creation) is performed with the LifeScope™ Software Admin Portal (included in this installation). There is no relationship between the LifeScope™ Software users accounts and Linux® host system users or LDAP users.
• **LDAP** – User accounts on LDAP-compliant authentication server (OpenLDAP, Active Directory, etc.) are eligible to be LifeScope™ Software users. User account management is performed with the LifeScope™ Software Admin Portal (included in this installation). As admin, you select the LDAP users to be allowed LifeScope™ Software access. You do not create user accounts.

• **Host** – Local (on the installation machine or cluster) Linux users are all LifeScope™ Software users. You do not create user accounts or select LifeScope™ Software users.

As admin, in the Admin Portal you can disable a LifeScope™ Software user and later re-enable the user.

---

### Scratch space cleanup

Under normal operation, LifeScope™ Software removes the files and directories it created in scratch space. However, as admin you must periodically check the scratch directory and manually delete old files that LifeScope™ Software did not remove.

The scratch directory is setting during installation. The default is `/scratch/solid/lifescope`.

The files and directories to delete have various names, but contain the name of an analysis module and end in a time stamp. Example names are:

- `wholeTranscriptomeFrag_secondary-human_chr17_6-wt.frag.map.main.20110430091312676`
- `test45_secondary-hg18-saet.main.20110702013417478`
- `test52_20110707030127_secondary-hg18-saet.main.20110707100009799`
- `saet_secondary-hg18-saet.main.20110427011039729`

**Note:** Be careful to not remove files or directories for a currently-running LifeScope™ Software process.

### Reference file conflicts

Depending on the local file permissions, the LifeScope admin might be responsible for adding the reference files to the reference repository. In order to avoid conflicts with other reference files in the reference repository, always add new reference files to the correct assembly subfolder under the `internal` directory. See “Add new reference files” on page 547 in Appendix E, “The Reference Repository” on page 539.

### Reset the admin password

The `resetpwd.sh` script is provided as an emergency way to reset the administrator’s password to its default value. To reset the admin password, follow these steps (at a Linux prompt):

1. Stop the LifeScope™ Software server:
   ```bash
   lscope-server.sh stop
   ```

2. Reset the admin password:
   ```bash
   resetpwd.sh lifescope
   ```

3. Restart the LifeScope™ Software server.

4. In the LifeScope™ Software UI, change the admin password to a secure password, according to your local policy.
The use of the `resetpwd.sh` script is not recommended for any user account except the LifeScope™ Software admin. Passwords reset for other users may become out of sync the authenticating realm.

If you change the path to the read-set repository, notify users to restart LifeScope™ Software so that the repository can be updated to the new path.

### Admin interfaces

#### GUI Admin Portal

The main administrative interface is the GUI Admin Portal, described in the LifeScope™ Genomic Analysis Software User Guide (Part no. 4476537). The Admin Portal supports all administrative tasks required for LifeScope™ Software.

The following administrative function cannot be done in the command shell and can only be done in the GUI Admin Portal:

- Enabling and disabling LifeScope™ Software users.

However, with the “host” authentication realm (selected during installation), all Linux user accounts are enabled as LifeScope™ Software users. With “host” authentication, you avoid the need to enable LifeScope™ Software users.

#### Shell admin commands

Limited admin functionality is also available in the command shell. Table 169 lists the LifeScope™ Software shell admin commands, which cover user administration and read repository index. These commands require the admin role privilege.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Administrative user commands:</strong></td>
<td></td>
</tr>
<tr>
<td><code>ls</code></td>
<td>Lists all users.</td>
</tr>
<tr>
<td><code>mk name</code></td>
<td>Creates a new user named <code>name</code>, with the password <code>name</code> and the role <code>user</code>.</td>
</tr>
<tr>
<td><code>rm name</code></td>
<td>Deletes the LifeScope™ Software shell user named <code>name</code>.</td>
</tr>
<tr>
<td><code>loggedin</code></td>
<td>Shows logged in users.</td>
</tr>
<tr>
<td><code>set username role [password]</code></td>
<td>Sets a user’s role and optionally set the user’s password. Valid roles are: admin, user</td>
</tr>
<tr>
<td><code>makepasswdfile</code></td>
<td>Creates an encrypted password file for non-interactive login. The file is named <code>lscope.passwd</code>, and is created in the directory where your <code>lscope.sh</code> shell command is originally issued.</td>
</tr>
<tr>
<td><strong>Other administrative command:</strong></td>
<td></td>
</tr>
<tr>
<td><code>rebuild</code></td>
<td>In <code>/reads</code>, rebuilds the read repository index, which is used to display the list of read-sets in the repository. Recent changes (additions or deletions) to the read repository are not reflected in the <code>ls</code> output until the index is rebuilt.</td>
</tr>
</tbody>
</table>

Only admin users have access to the `/users` category. The admin user commands must be executed at the `/users` level.

### Other information

The `<installdir>/server` directory contains the following files of interest:
• **server.properties** – Contains important LifeScope™ Software configuration information and locations, such as the authentication realm, port numbers, projects repository directory, results directory, and others.

• **server.err** – Contains error messages generated by the LifeScope™ Software server.

• **server.log** – Contains log messages generated by the LifeScope™ Software server.

• **server.pid** – Contains the process id and port number used by the LifeScope™ Software server.

See also the following sections:

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
<th>Chapter or appendix</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Logs”</td>
<td>489</td>
<td>“Troubleshooting”</td>
</tr>
<tr>
<td>“Troubleshooting”</td>
<td>490</td>
<td>“Troubleshooting”</td>
</tr>
<tr>
<td>“Naming restrictions”</td>
<td>73</td>
<td>“Understand The LifeScope™ Software Shell”</td>
</tr>
<tr>
<td>“Error behavior and error messages”</td>
<td>100</td>
<td>“Run a Command Shell Analysis”</td>
</tr>
<tr>
<td>“Check LifeScope™ Software servers”</td>
<td>61</td>
<td>“Before You Begin”</td>
</tr>
</tbody>
</table>
Demo Analyses

This appendix covers:

- Overview ................................................................. 567
- List of demo modules .................................................. 567
- Demos location .......................................................... 568
- How a demo is setup .................................................... 568
- Run all demos ............................................................ 569
- Run a single demo analysis ........................................... 570
- Output files and logs .................................................... 570

Overview

This appendix describes the demo modules included in an optional download. The demos show how to run an individual LifeScope™ Software analysis module, such as mapping or SNPs, using the LifeScope™ Software command shell.

The demo analyses are mostly self-contained. Input data and references files are provided. The exception is that some demos use hg18 references files in the LifeScope™ Software reference repository.

The standard workflows, which are sequences of analyses combined into one run of LifeScope™ Software, are recommended for general use instead of the demos. See Chapter 7, “Run a Standard Workflow Analysis” on page 103 for information on standard workflows. The demo analyses are based on execution methods used in earlier versions of the software, and have been adapted for the LifeScope™ Software command shell.

List of demo modules

The following analyses are included in the demos download:

- Mapping – fragmentMapping
- Paired mapping – pairMapping
- SNPs – diBayes
- Human CNV – cnv
- Inversion – inversion
- Large indel – largeIndel
- Small indel – smallIndel
- SAET – saet
- Enrichment – enrichment
Appendix H  Demo Analyses

Demos location

The LifeScope™ Software demos are a separate download not included in the main installation. Ask your LifeScope™ Software administrator for the location of the demos.

How a demo is setup

This section describes the mechanics of how each demo and its files and directories are organized.

Each demo analysis directory contains a run.sh script, which invokes lscope.sh in scripted mode. The scripted lscope.sh shell commands are contained in a file named run.txt. Here are example run.txt contents, for SNPs:

```bash
cd projects
mk examples
cd examples
rm diBayes
mk diBayes
cd diBayes
set workflow ./analysis.pln
add bam ./bam/F3-R3-Paired.twoSNPs.bam
add bam ./bam/F3-R3-Paired.twoSNPsmerged.bam
set reference hg18
run
wait
ls
exit
```
Table 170 explains the purpose of the steps shown above.

**Table 170  Description of sample shell commands to run a workflow**

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>cd /projects</td>
<td>In the projects repository, create a project named examples, and in that project create an analysis named dibayes. Open the dibayes analysis.</td>
</tr>
<tr>
<td>mk examples</td>
<td></td>
</tr>
<tr>
<td>cd examples</td>
<td></td>
</tr>
<tr>
<td>rm dibayes</td>
<td>Note: The open analysis (cd dibayes, in this case) command is important. The data, reference, and other configuration commands which follow apply only to the current analysis.</td>
</tr>
<tr>
<td>mk dibayes</td>
<td>Note: The demos all delete the current demo analysis if it exists (the rm dibayes command in this case).</td>
</tr>
<tr>
<td>set workflow ./analysis.pln</td>
<td>Define the workflow used with this analysis.</td>
</tr>
<tr>
<td>add bam ./bam/F3-R3-Paired.twoSNPs.bam</td>
<td>Define the BAM input files containing data for this analysis.</td>
</tr>
<tr>
<td>add bam ./bam/F3-R3-Paired.twoSNPsmerged.bam</td>
<td></td>
</tr>
<tr>
<td>set reference hg18</td>
<td>Define the reference for this analysis. The string hg18 matches an assembly name in the reference repository.</td>
</tr>
<tr>
<td>run</td>
<td>Start the analysis run.</td>
</tr>
<tr>
<td>wait</td>
<td>Wait until the current analysis run completes.</td>
</tr>
<tr>
<td>ls</td>
<td>Display status information about the analysis run.</td>
</tr>
</tbody>
</table>

**Workflow**

The workflow defined in this demo is a PLN file, analysis.pln. A PLN file defines your analysis through a list of tasks, which is a list of one or more INI or PLN files. For each listed INI file, LifeScope™ Software executes an analysis run of the corresponding module.

In the SNPs demo, analysis.pln calls tertiary.pln, which contains the entry dibayes.ini to invoke the SNPs analysis module.

**Input files**

The demo uses BAM input, because SNPs is a tertiary analysis. Demos which do mapping analysis instead take XSQ files as input.

**Run all demos**

This section describes how to run all the demo analyses in one run. The combined run should complete in under 30 minutes.

The demos require the default reference repository, which includes hg18 and hg19 assemblies.

Follow these steps to execute all the demos in one run:

1. Cd to the demo directory.
2. Execute this command in the Linux shell:
   ```bash
   runall.sh
   ```
   This script runs each demo one at a time.
The `runall.sh` script by default executes the run under the `lifescop`e user account. To use a different user name, use the `-u` and `-w` options:

```
runall.sh -u username -w password
```

### Run a single demo analysis

Each demo is executed with the same command. To execute any demo, follow these steps:

1. Cd to the general demo directory, and then cd to the specific analysis directory.
2. Execute this command in the Linux shell:
   ```
   run.sh
   ```
   The `run.sh` invokes `lscope.sh` in scripted mode. The scripted `lscope.sh` shell commands are contained in a file named `run.txt`, and are similar to the commands in Table 170.

   The major differences in `run.txt` commands are the following:
   - The analysis name
   - The input commands, add xsq or add bam
   - The set reference command

The `runall.sh` script by default executes the run under the `lifescop`e user account. To use a different user name, use the `-u` and `-w` options:

```
runall.sh -u username -w password
```

### Output files and logs

The results files are generated in the projects repository, which is set during installation. Within the projects repository directory, the demos results are generated in the following directories:

```
user_name/examples/analysis_name(outputs/sample_name)
```

This list describes the folder names

- `user_name` – The LifeScope™ Software user name of the person running the demo.
- `examples` – The project name used by all demos.
- `analysis_name` – The name of the analysis module used by a particular demos. For example: cnv, dibayes, enrichment.
- `outputs` – A hard-coded subfolder name.
- `sample_name` – Determined by the sample name in the input data.

In its results directory, each of the demos analyses creates an output file named `summary.log`. Check for a success message in the last statement of the log.

The results files are overwritten on subsequent runs executed by the same user. You can view the results of your own demo runs in the LifeScope™ Software UI.
END USER LICENSE AGREEMENT

This is a legal agreement between you, the person or entity receiving software products and/or software support ("Licensee"), and Life Technologies Corporation, having offices at 5791 Van Allen Way, Carlsbad California 92008 USA ("Licensor"). This agreement is part of a package that includes one or more software products and certain electronic and/or written materials. This agreement covers your licensing of such software and/or purchase of support.

You must agree to the terms in this End User License Agreement ("EULA") in order to access the software and/or receive support.

BY CLICKING YOUR ACCEPTANCE OF THIS EULA, OR BY INSTALLING OR USING THE SOFTWARE (defined below) OR ANY OTHER COMPONENT OF THE PACKAGE, YOU ACKNOWLEDGE THAT YOU HAVE READ ALL OF THE TERMS AND CONDITIONS OF THIS EULA, UNDERSTAND THEM, AND AGREE TO BE LEGALLY BOUND BY THEM. If you do not agree to the terms of this EULA, you may not install or use the Software. In addition to the restrictions imposed under this EULA, Software installation instructions or release notes, and Support policies (defined below) shall apply to your use of the Software and receipt of Support.

As used in this EULA: "Authorized Users" means, collectively, the personnel authorized by you to use the Software for your benefit. Unless otherwise expressly allowed by this EULA, Authorized Users may include only your employees and agents having a need to know, and Authorized Users may not be entities or persons in the business of licensing or otherwise providing products or services competitive with the Software. "Software" means the software product(s) accompanying this EULA and the content therein; including the associated data, user manuals, user documentation and application program interfaces (if any), and any patches, updates, upgrades, improvements, enhancements, fixes and revised versions of any of the foregoing that may be provided to you from time to time, and any combination of the foregoing. "Affiliate" means any entity Controlling, Controlled by, or under common Control with the referenced entity, where the term "Control" means the possession, direct or indirect, of the power to direct or cause the direction of the management and policies of an entity, whether through the ownership of voting securities, by contract, or otherwise.

1. GRANT OF SOFTWARE LICENSE. Subject to the terms and conditions of this EULA, Licensor grants to you a non-exclusive, non-transferable license ("License") for an unlimited number of Authorized Users to use the Software and Support for your internal operations and internal data processing purposes. Use of the Software on a network is authorized. This EULA and the License granted hereunder shall be effective until terminated in accordance with Section 7 below. The Software may be provided under a "Trial License" for your internal evaluation purposes only. Depending on the mode of delivery of the Trial License, you may (i) access the Software though an online account mechanism, or (ii) you may install and operate the
Software on computer hardware owned or controlled by you. Under a Trial License you may not network the Software for use at any other geographic location. A Trial License may not be used for any commercial purposes. Software provided under a Trial License is made available to you "AS IS", AND LICENSOR MAKES NO REPRESENTATION OR WARRANTY REGARDING SUCH SOFTWARE.

2. RESTRICTIONS ON USE. You acknowledge that you are receiving LICENSED RIGHTS only. The Software may only be used internally, by your Authorized Users. If any Software is provided on separate media (e.g., a CD-ROM), you may make a single copy solely for your internal backup purposes. You shall not directly or indirectly: (i) sell, rent, lease, distribute, redistribute or transfer any of the Software or any rights in any of the Software, including without limitation through "charge back" or any other selling, reselling, distributing or redistributing within your organization of any usage capacity you have licensed, without the prior express written approval of Licensor, (ii) modify, translate, reverse engineer (except to the limited extent permitted by law), decompile, disassemble, attempt to discover the source code for, create derivative works based on, or sublicense any of the Software, (iii) use any Software for the benefit of any third parties (e.g., in an ASP, outsourcing or service bureau relationship), or in any way other than in its intended manner, without the prior express written approval of Licensor, (iv) remove any proprietary notice, labels, or marks on or in the Software, or (v) disable or circumvent any access control or related device, process or procedure established with respect to the Software. If the Software design permits modification, then you may only use such modifications or new software programs for your internal purposes and otherwise consistent with the License and Authorized Scope. You are responsible for all use of the Software and for compliance with this EULA; any breach by you or any user of the Software shall be deemed to have been a breach by you. Licensor reserves all rights not expressly granted; no right or license is granted hereunder, express or implied or by way of estoppel, to any intellectual property rights other than as expressly set forth herein; and this License does not by itself convey or imply the right to use the Software in combination with any other product(s). As between you and Licensor, Licensor retains all right, title, and interest in and to the Software, which rights include, but are not limited to, patent, copyright, moral, trademark, trade secret and all other intellectual property rights. You agree and acknowledge that you have been provided sufficient information such that you do not need to reverse engineer the Software in any way to permit other products or information to interoperate with the Software.

3. NO SEPARATION OF COMPONENTS. The Software is licensed as a single product. Some Licensor software products combine separately available components into a single product (e.g., a software suite product may be comprised of multiple component products). When licensed as a combination product, the component parts may not be separated for use independently of the combination product. You must first purchase a license to each component of the combination product before you may use it independently of the combination product.

4. TERM AND TERMINATION. Unless otherwise agreed, the term of this EULA shall continue until it expires or is terminated; however, if you are receiving a Trial License, the term shall expire thirty (30) days following the date you receive the Software. Termination or expiration of this EULA shall concurrently terminate all Licenses granted under this EULA. Licensor shall not refund any amounts paid by you hereunder in the event of expiration or termination of this EULA unless expressly provided in this EULA. Licensor may terminate this EULA (or any License) upon fifteen (15) days written notice if you breach this EULA and do not cure the breach within fifteen (15) days following receipt of written notice of breach. Immediately upon any termination or expiration of this EULA, you agree to: (a) un-install and cease
use of the Software for which your rights have been terminated; (b) upon request, return to Licensor (or destroy) all copies of the Software and any other Confidential Information or proprietary materials in your possession for which your rights have been terminated; and (c) upon request, certify in writing your compliance with (a) and (b), above.

5. **CONFIDENTIALITY.** You agree to protect Licensor’s Confidential Information with the same degree of care used to protect your own confidential information (but in no event less than a reasonable standard of care), and not to use or disclose any portion of such Confidential Information to third parties, except as expressly authorized in this EULA. You acknowledge that the Software, including its content, structure, organization and design constitute proprietary and valuable trade secrets (and other intellectual property rights) of Licensor and/or its licensors. The term “Confidential Information” means, collectively, non-public information that Licensor (and its licensors) provide and reasonably consider to be of a confidential, proprietary or trade secret nature, including but not limited to (i) the Software, and (ii) confidential elements of the Software and Licensor’s (and its licensors’) technology and know-how, whether in tangible or intangible form, whether designated as confidential or not, and whether or not stored, compiled or memorialized physically, electronically, graphically, photographically, or in writing. Confidential Information does not include any information which you can demonstrate by credible evidence: (a) is, as of the time of its disclosure, or thereafter becomes part of the public domain through no fault of yours; (b) was rightfully known to you prior to the time of its disclosure, or to have been independently developed by you without use of Confidential Information; and/or (c) is subsequently learned from a third party not under a confidentiality obligation with respect to such Confidential Information. Confidential Information that is required to be disclosed by you pursuant to a duly authorized subpoena, court order, or government authority shall continue to be Confidential Information for all other purposes and you agree, prior to disclosing pursuant to a subpoena, court order, or government authority, to provide prompt written notice and assistance to Licensor prior to such disclosure, so that Licensor may seek a protective order or other appropriate remedy to protect against disclosure.

6. **WARRANTY AND DISCLAIMER.** Licensor warrants that (i) except with respect to Software provided under a Trial License (in respect of which no warranty is made, as described in Section 1), for a period of twenty (20) days from the date of acceptance of the Software as described in Section 1, the Software will, under normal use and as unmodified, substantially perform the functions described in its accompanying documentation; and (ii) Licensor will perform Support services during the License term in a professional and workmanlike manner. No warranty is provided for uses beyond the Authorized Scope. THE FOREGOING EXPRESS WARRANTIES REPLACE AND ARE IN LIEU OF ALL OTHER WARRANTIES AND REPRESENTATIONS BY LICENSOR, WHETHER EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO ANY IMPLIED OR OTHER WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, NON-INFRINGEMENT OR NON-MISAPPROPRIATION OF INTELLECTUAL PROPERTY RIGHTS OF A THIRD PARTY, CUSTOM, TRADE, QUIET ENJOYMENT, ACCURACY OF INFORMATIONAL CONTENT, OR SYSTEM INTEGRATION. NO WARRANTY IS MADE THAT ANY SOFTWARE WILL OPERATE IN AN ERROR FREE, UNINTERRUPTED OR COMPLETELY SECURE MANNER, IN COMBINATION WITH THIRD PARTY HARDWARE OR SOFTWARE PRODUCTS, OR THAT ALL DEFECTS CAN BE CORRECTED. YOU ACKNOWLEDGE THAT LICENSOR HAS NO CONTROL OVER THE SPECIFIC CONDITIONS UNDER WHICH YOU USE THE SOFTWARE. ACCORDINGLY, EXCEPT FOR THE FOREGOING EXPRESS WARRANTY, LICENSOR CANNOT AND
DOES NOT WARRANT THE PERFORMANCE OF THE SOFTWARE OR ANY PARTICULAR RESULTS THAT MAY BE OBTAINED BY THE USE OF THE SOFTWARE. THE SOFTWARE AND SUPPORT DO NOT REPLACE YOUR OBLIGATION TO EXERCISE YOUR INDEPENDENT JUDGMENT IN USING THE SOFTWARE. The warranties made by Licensor may be voided by abuse or misuse of the Software and/or Support.

7. EXCLUSIVE REMEDY. Licensor’s sole obligation and liability, and your sole and exclusive remedy under the warranties set forth in Section 6, shall be for Licensor to use commercially reasonable efforts to have the problem remedied, to re-perform Support services or to deliver to you a replacement for the defective Software (in each case, as determined by Licensor and as applicable), provided that Licensor is notified in writing of all warranty problems during the applicable warranty period.

8. THIRD PARTY SOFTWARE AND DATABASES. This Software uses third-party software components from several sources. Portions of these software components are copyrighted and licensed by their respective owners. Various components require distribution of source code or if a URL is used to point the end-user to a source-code repository, and the source code is not available at such site, the distributor must, for a time determined by the license, offer to provide the source code. In such cases, please contact your Life Technologies representative. In addition, various licenses require that the end user receive a copy of the license. Such licenses may be found on-line as supporting files on the download page for the Software. In order to use this Software, the end-user must abide by the terms and conditions of these third-party licenses. You understand that third party products integrated into the Software or provided for use with the Software may be subject to additional terms and conditions and/or license agreements from the applicable third party vendor, which shall govern over conflicting terms of this EULA for purposes of your relationship with the third party vendor. You agree not to use any such third party product on a stand-alone basis independent of the Software, unless you have purchased the appropriate license from the third party vendor for use of such products. NOTWITHSTANDING ANYTHING TO THE CONTRARY IN THIS AGREEMENT, ALL THIRD PARTY SOFTWARE, DATABASES AND OTHER PROGRAMS AND SOFTWARE COMPONENTS ARE PROVIDED "AS IS" WITHOUT ANY WARRANTY WHATSOEVER FROM LICENSOR. ANY DATABASES OR OTHER INFORMATION PROVIDED BY LICENSOR ARE DESIGNED TO SUPPLEMENT OTHER SOURCES OF INFORMATION, ARE NOT INTENDED TO REPLACE YOUR PROFESSIONAL DISCRETION AND JUDGMENT AND LICENSOR MAKES NO REPRESENTATIONS OR WARRANTIES REGARDING SUCH DATABASES OR INFORMATION, THEIR ACCURACY, COMPLETENESS OR OTHERWISE. Licensor agrees, upon request and as Licensor’s sole liability and obligation, and for your convenience only, to have passed through to you (to the extent it may reasonably do so) any warranties and indemnifications provided by the applicable third party vendor of any third party products provided to you. To the extent any problem or liability arises from a third party product, you agree to seek recourse solely from the applicable third party vendor and not Licensor.

9. LICENSOR INDEMNIFICATION. Subject to the limitations set forth herein, Licensor agrees to defend you against any claims, actions, suits and proceedings brought against you by unaffiliated third parties arising from or related to a claim that the Software (other than any third party or open source components or elements) infringes upon such third party’s copyrights, and Licensor agrees to pay all damages that a court finally awards to such third party, and all associated settlement amounts agreed to by Licensor in writing; provided that, Licensor receives from you (i) prompt written notice of the claim; (ii) all necessary assistance, information and authority
necessary for Licensor to defend the claim and perform Licensor’s obligations under this Section; and (iii) sole control of the defense of such claim and all associated settlement negotiations. If such a claim is made or appears likely to be made, you agree to permit Licensor to enable you to continue to use the affected Software, or to have the Software modified to make it non-infringing, or to have the Software replaced with a substantially functional equivalent. If Licensor determines that none of these options is reasonably available, then Licensor may terminate this EULA in whole or with respect to the affected Software product. THIS SECTION STATES LICENSOR’S AND ITS AFFILIATES’ ENTIRE OBLIGATION AND LIABILITY REGARDING INFRINGEMENT OF THIRD PARTY RIGHTS OF ANY KIND OR CLAIMS OF ANY SUCH INFRINGEMENT. Licensor will have no responsibility for (v) any use of any product after you have been notified to discontinue use because of a third party claim of infringement, (v) the alteration of the Software or the combination of the Software with third party materials, products or software, (x) use of the Software by any person or entity other than an Authorized User, (y) any misuse or unauthorized use of the Software, or (z) failure to use provided updated or modified Software to avoid a claim of infringement or misappropriation.

10. **INDEMNIFICATION BY YOU.** You agree to indemnify and defend Licensor, its licensors, and its affiliates, against any third party claims arising from or related to your use or misuse of the Software or any breach of the terms and conditions of this EULA, and you agree to pay all costs, losses, damages, and attorneys’ fees that a court finally awards, and all associated settlements.

11. **LIMITATION OF LIABILITY.** EXCEPT TO THE EXTENT PROHIBITED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR’S OR ITS AFFILIATES’ TOTAL, AGGREGATE LIABILITY ARISING FROM OR RELATED TO THIS AGREEMENT, THE SOFTWARE AND/OR SUPPORT (INCLUDING FOR NEGLIGENCE, STRICT LIABILITY, BREACH OF CONTRACT, MISREPRESENTATION, AND OTHER CONTRACT OR TORT CLAIMS), EXCEED THE AMOUNT OF YOUR DIRECT DAMAGES ACTUALLY INCURRED, UP TO THE AMOUNT OF FEES PAID TO LICENSOR UNDER THIS AGREEMENT FOR THE SOFTWARE PRODUCT OR SUPPORT PRODUCT THAT IS THE SUBJECT OF THE CLAIM UNDERLYING THE DAMAGES; OR, IN THE CASE OF SOFTWARE PROVIDED UNDER A TRIAL OR DEMONSTRATION LICENSE, ONE HUNDRED DOLLARS ($100.00), WHICHER IS LESS.

12. **EXCLUSION OF DAMAGES.** EXCEPT TO THE EXTENT PROHIBITED BY APPLICABLE LAW, UNDER NO CIRCUMSTANCES SHALL LICENSOR, ITS AFFILIATES, OR ANY OF THEIR SUPPLIERS OR LICENSORS BE LIABLE HEREUNDER FOR ANY OF THE FOLLOWING: (I) THIRD PARTY CLAIMS, EXCEPT AS PROVIDED IN SECTION 9, (II) LOSS OR DAMAGE TO ANY SYSTEMS, RECORDS OR DATA, (III) DIRECT DAMAGES FOR BREACH OF WARRANTY (IN RESPECT OF WHICH ANY LIABILITY SHALL BE LIMITED TO RE-PERFORMANCE OR REFUND AS SPECIFIED IN SECTION 7), AND/OR (IV) INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL, PUNITIVE, RELIANCE, OR COVER DAMAGES (INCLUDING WITHOUT LIMITATION FOR LOST PROFITS, LOST SAVINGS AND DAMAGE TO ANY DATA OR SYSTEMS); EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES AND EVEN IF A LIMITED REMEDY SET FORTH HEREIN FAILS OF ITS ESSENTIAL PURPOSE. YOU ARE SOLELY RESPONSIBLE AND LIABLE FOR VERIFYING THE ACCURACY AND ADEQUACY OF ANY OUTPUT FROM THE SOFTWARE, AND FOR ANY RELIANCE THEREON.

13. **REGULATED USES.** FOR RESEARCH USE ONLY (RUO): NOT INTENDED FOR ANY ANIMAL OR HUMAN THERAPEUTIC OR DIAGNOSTIC USE

You acknowledge that the Software has not been cleared, approved, registered or otherwise qualified (collectively, "Approval") by Life Technologies Corporation with any regulatory agency for use in diagnostic or therapeutic procedures, or for any other use requiring compliance with any federal or state law regulating diagnostic or therapeutic products, blood products, medical devices or any similar product (hereafter collectively referred to as "federal or state drug laws"). The Software may not be used for any purpose that would require any such Approval unless proper Approval is obtained. You agree that if you elect to use the Software for a purpose that would subject you or the Software to the jurisdiction of any federal or state drug laws, you will be solely responsible for obtaining any required Approvals and otherwise ensuring that your use of the Software complies with such laws.


15. **LEGAL COMPLIANCE; RESTRICTED RIGHTS.** The Software is provided solely for internal research and solely for lawful purposes and use. You shall be solely responsible for, and agree to comply with, all applicable laws, statutes, ordinances, and other governmental authority, however designated. Without limiting the foregoing, this EULA is expressly made subject to any United States government laws, regulations, orders or other restrictions regarding export from the United States and re-export from other jurisdictions of equipment, computer hardware, software, technical data and information or derivatives of such equipment, hardware, software or technical data and information. You agree to comply with all applicable export and re-export control laws and regulations in regard to products (including computer hardware, software, deliverables, technical data, source code, or any other technology, equipment, and/or derivatives of such hardware, software, deliverables, technical data, source code, equipment, or any other technology) received from Licensor. You further certify that you will not, directly or indirectly, without obtaining prior authorization from the competent government authorities as required by those laws and regulations: (1) sell, export, re-export, transfer, divert, or disclose technical data or dispose of any product or technology received from Licensor to any prohibited person, entity, or destination; or (2) use the product or technology for any use prohibited by the laws or regulations of the United States. You will reasonably cooperate with Licensor and will provide to Licensor promptly upon request any certificates or documents as are reasonably requested to obtain approvals, consents, licenses and/or permits required for any payment or any export or import of products or services under this EULA, at Licensor’s expense. Your breach of this provision shall constitute cause for immediate termination of this EULA. You agree to indemnify and hold harmless Licensor, its affiliates, and their respective officers, directors, employees and agents for your noncompliance with this Section. The Software is a "commercial item," as that term is defined in 48 C.F.R. 2.101, consisting of "commercial computer software" and

16. GOVERNING LAW; SEVERABILITY. This EULA shall be governed in all respects by the laws of the State of California, USA, without regard to its conflicts of law rules or principles. Any dispute arising out of or related to this EULA shall be resolved only in the state or federal courts having subject matter jurisdiction in California. This Agreement shall not be governed by the United Nations Convention on Contracts for the International Sale of Goods. Each party hereby consents to the exclusive jurisdiction and venue of such courts. If any provision of this EULA is held to be illegal or unenforceable for any reason, then such provision shall be deemed to be restated so as to be enforceable to the maximum extent permissible under law; the remainder of this EULA shall remain in full force and effect.

17. SOFTWARE MAINTENANCE AND SUPPORT. Licensor offers certain software maintenance and technical support service programs documented in Licensor’s then-current Support policies (“Support”). Subscription to Support shall be governed by this EULA and by the terms and conditions set forth in such policies. Licensor shall have no obligation to provide Support if (a) the Software is not used in accordance with the Documentation or Authorized Scope; (b) the Software was modified by you; (c) you have not implemented all upgrades that would otherwise correct the problem; or (d) the problem is caused by your misuse, negligence or other cause within your control. Licensor may change its Support policies and prices at any time. Licensor reserves the right to discontinue Support services for any Software where Licensor generally discontinues such services to all Licensees of such Software, in which case such discontinuation shall not automatically terminate this EULA and the License. If you terminate Support and then re-enroll, Licensor may charge you a reinstatement fee.

18. GENERAL. This EULA, including any Support policies, (all of which are incorporated herein), are collectively the parties’ complete agreement regarding its subject matter, superseding any prior oral or written communications, representations or agreements. In the event that any prior oral or written communication is in direct conflict with the terms of this EULA, this EULA shall control. You understand and agree that, to the extent Licensor permits you to use a non-Licensor purchase order or other form to order Software and/or Support, Licensor does so solely for your convenience. Any terms in any such forms that purport to vary or are in addition to or inconsistent with any terms in this EULA shall be deemed to be void and of no effect. Amendments or changes to this EULA must be in mutually executed writings to be effective. Sections 1, 4, 5, and 7 through 22, inclusive, shall survive the termination or expiration of this EULA. The parties are independent contractors for all purposes under this EULA. Neither party shall be liable for any delay or failure due to force majeure and other causes beyond its reasonable control; provided that the foregoing shall not apply to any of your payment obligations. Any notices under this EULA to Licensor must be personally delivered or sent by certified or registered mail, return receipt requested, or by nationally recognized overnight express courier, to the address specified herein or such other address as Licensor may specify in writing. Such notices will be effective upon receipt, which may be shown by confirmation of delivery. All such notices shall be sent to the attention of the Chief Legal Officer of Life Technologies Corporation (unless otherwise specified by Licensor). You may not assign or otherwise transfer this EULA or any License without Licensor’s prior written consent. This EULA shall be binding upon and inure to the benefit of the parties’ successors and permitted assigns. You agree, at Licensor’s request and reasonable
expense, to provide reasonable assistance and cooperation to Licensor and its
designees, and to give testimony and execute documents and to take such further acts
reasonably requested by the other to acquire, transfer, maintain, perfect, and enforce
Licensor’s intellectual property rights as described in this EULA. To the extent you fail
to do so, you appoint Licensor’s or its affiliates’ officers as your attorney in fact to
execute documents on your (and your personnel’s), successors’ and assigns’ behalf for
this limited purpose.
<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>alignment</td>
<td>The process of mapping sequencing reads to a reference genome or sequence.</td>
</tr>
<tr>
<td>alignment browser</td>
<td>An interactive software in which to view alignments of sequencing reads with the reference genome or sequence.</td>
</tr>
<tr>
<td>alignment score</td>
<td>Matching score; an optimal alignment is an alignment giving the highest score of matches.</td>
</tr>
<tr>
<td>allele</td>
<td>One of two or more alternative nucleotide sequences at the same location on homologous chromosomes.</td>
</tr>
<tr>
<td>analysis</td>
<td>A data analysis run in LifeScope™ Software; may include secondary analysis on data from multiple sequencer runs.</td>
</tr>
<tr>
<td>annotated gene</td>
<td>Within one of several reference databases, a gene sequence that has biological attributes attached that describe structure or function, such as coding regions or biochemical function.</td>
</tr>
<tr>
<td>annotation</td>
<td>Biological attributes or metadata that are attached to sequence data or files. Examples include: genes and protein-coding features, and verified variants.</td>
</tr>
<tr>
<td>BAM file</td>
<td>The compressed binary version of the Sequence Alignment/Map (SAM) format, a compact and index-able representation of nucleotide sequence alignments. The BAM file is generated from the mapping step and has special attributes to support SOLiD® data.</td>
</tr>
<tr>
<td>barcode</td>
<td>A short, unique sequence that is incorporated into a library that enables identification of the library during multiplex sequencing.</td>
</tr>
<tr>
<td>barcode group, barcode pool</td>
<td>The conceptual grouping used during data analysis if more than one barcode is used during libration preparation from a single biological specimen.</td>
</tr>
<tr>
<td>barcoded library</td>
<td>A library that has a unique barcode sequence incorporated that enables identification of the library during multiplex sequencing.</td>
</tr>
<tr>
<td>base-space analysis</td>
<td>Analysis of SOLiD® sequencing data that has been converted from color space to a nucleotide sequence.</td>
</tr>
</tbody>
</table>
### Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bayesian</td>
<td>An algorithm that is used to evaluate the probability of the existence of a heterozygote or a non-reference homozygote when the coverage at the position is not high.</td>
</tr>
<tr>
<td>bead</td>
<td>The substrate to which the individual strands of DNA molecules in a SOLiD® library are attached. The covalently attached strand serves as the template for SOLiD® sequencing after the bead is immobilized on the flowchip surface.</td>
</tr>
<tr>
<td>BED file</td>
<td>BED format provides a flexible way to define the data lines that are displayed in an annotation track. The required BED fields are:</td>
</tr>
<tr>
<td></td>
<td>• chrom - name of the chromosome</td>
</tr>
<tr>
<td></td>
<td>• chromStart - starting position of the feature in the chromosome (the first base in a chromosome is numbered 0).</td>
</tr>
<tr>
<td></td>
<td>• chromEnd - ending position of the feature in the chromosome</td>
</tr>
<tr>
<td>BEDGRAPH</td>
<td>In targeted resequencing, a text file in BEDGRAPH format that describes the depth of coverage within targets by on-target alignments.</td>
</tr>
<tr>
<td>bisulfite read</td>
<td>A sequence read of bisulfite-converted DNA, in which unmethylated cytosine residues have been converted to uracil. Used in methylation analysis of genomic DNA.</td>
</tr>
<tr>
<td>BLAST</td>
<td>In bioinformatics, Basic Local Alignment Search Tool, or BLAST, is an algorithm for comparing primary biological sequence information, such as the amino-acid sequences of different proteins or the nucleotides of DNA sequences.</td>
</tr>
<tr>
<td>call stringency</td>
<td>The criteria for calling a nucleic acid species or genetic variant present in the biological sample.</td>
</tr>
<tr>
<td>calling threshold</td>
<td>The criteria for calling a nucleic acid species or genetic variant present in the biological sample.</td>
</tr>
<tr>
<td>ChIP-Seq</td>
<td>Sequence analysis of genomic regions associated with DNA-binding proteins.</td>
</tr>
<tr>
<td>classic mapping</td>
<td>An algorithm that searches for matches at the full read length, allowing up to a user-specified number of mismatches. Now called global mapping.</td>
</tr>
<tr>
<td>clear zone</td>
<td>The threshold to decide whether a read is mapped uniquely in the reference. Clear zones are used in previous versions of the software, but are not used with LifeScope™ Software.</td>
</tr>
<tr>
<td>clipping, hard clipping</td>
<td>An operation that codes a portion of a color alignment with a number of mismatches that prevent seed or anchor extension.</td>
</tr>
<tr>
<td>clone</td>
<td>A single bead with a clonal population of templates for sequencing, generated during templated bead preparation/emulsion PCR.</td>
</tr>
<tr>
<td>color balance</td>
<td>The relative proportion of beads in a given sequencing cycle that are called as each of the four colors.</td>
</tr>
</tbody>
</table>
| color call            | For each SOLiD® chemistry cycle, the color that is determined for each bead. }
color space, color-space analysis

Color space data is a sequence of colors obtained from 2-base encoded probes in SOLiD® System sequencing, representing a series of overlapping dinucleotides. Color space data is converted to a nucleotide sequence by aligning to a reference.

complexity

The size of the genome that is composed of unique sequences. With respect to SOLiD® libraries, the number of independent molecules derived from the original DNA input, not from amplification.

consensus calls

A file that lists the SNPs called with the SNP Finding algorithm, and provides general information about each position.

contig

A nucleotide sequence that has been assembled from shorter, overlapping sequences.

core

A single core is equivalent to a processor on a traditional computer. Multiple cores increase the performance and efficiency of a computer that is running multiple programs.

counting

Expression analysis that focuses on relative or absolute quantification of RNA molecules in a biological specimen.

copy number variation [CNV]

A variation in the number of copies of DNA segments among individuals in a population. The length of the segment can vary, and it is typically more than a few base pairs. Some use a broader definition that includes any insertion, deletion, or duplication longer than a few base pairs.

CountTags

A module to count the number of reads that align within genomic features.

counting

Sequence analysis that generates read or tag counts for annotated regions of a reference sequence.

coverage

- In genomic analysis, the number of aligned (or mapped) sequencing reads that span a position in the reference genome.
- In RNA analysis, this term is sometimes used to describe the fraction of the reference sequence that has sequencing reads aligned (or mapped), at a certain calling threshold.

csfasta file

The SOLiD® System produces color-space sequence reads in a fasta-format labeled as CSFASTA. These reads can be retained and analyzed in color space.

dbSNP

A Single Nucleotide Polymorphism database repository for single nucleotide polymorphisms and short insertion and deletion polymorphisms, hosted by the NCBI.

de novo sequencing

The initial generation of the primary genetic sequence of a particular organism, without use of a reference genome or sequence.

deletion

A gap in a nucleotide sequence with respect to a reference genome or sequence.

dibayes

Analysis module to identify SNPs from mapped and processed SOLiD® System color space reads. Also called diBayes and SNP Finding.
discordant  
In large indel analysis, mate-paired reads that have been mapped to a reference genome, and whose inter-read distance deviates significantly from the expected insert size of the mate-paired library as a whole.

Discordant pairs have insert sizes that deviate significantly from the expected value. Discordant pairs containing a putative deletion appear larger when mapped to the reference.

EBI  
European Bioinformatics Institute.

enriched data  
Sequencing data that is associated with a region of interest in a targeted resequencing experiment or in a low frequency variant detection workflow.

enrichment statistics  
Calculated results derived from the targeted resequencing enrichment module in LifeScope™ Software. Also generated in low frequency variant detection workflows.

ENSEMBL  

error correction  
(SAET) This step takes each read in the input file and attempts to correct it if any of its k-mers are not present in the spectrum. Use of multi-threaded versioning may be recommended for speed of process.

Error expectation metric (EEM)  
A whole transcriptome pipeline, a parameter that is used in the calculation of the Junction Confidence Value metric, to determine the statistical significance of detected variant fusion transcripts.

evidence graph  
A visual representation of the data structure of a candidate fusion transcript.

exon  
In eukaryotic organisms, a segment of a gene that encodes part or all of a protein. Exons may be separated by introns that are spliced out of the primary transcript to produce a mature mRNA for translation into protein.

exon mapping  
A whole transcriptome pipeline, a step in which reads are mapped to the set of exon sequences defined by the genome annotation.

F3 tag  
Sequencing data derived from the P1 end of the template in the SOLiD® templated bead, using forward ligation chemistry. The F3 tag is generated using the SOLiD® FWD1 Seq. Primers or the SOLiD® Small RNA Seq. Primers. See the 5500 Series SOLiD® Sequencers: Reagents and Consumables Ordering Guide (Part no. 4465650) for an illustration.

F5 tag  
Sequencing data derived from the P2 end of the template in the SOLiD® templated bead, using reverse ligation chemistry. The F5 tag is generated using the SOLiD® REV1 (DNA) Seq. Primers or the SOLiD® REV1 (RNA) Seq. Primers. See the 5500 Series SOLiD® Sequencers: Reagents and Consumables Ordering Guide (Part no. 4465650) for an illustration.

false positive  
Sequence variants that are present in the output data from a sequencing run but are not present in the biological source.
<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter</td>
<td>A reference sequence of interest that is used to select reads that align with or map to the reference sequence for further analysis.</td>
</tr>
<tr>
<td>filter mapping</td>
<td>Alignment of sequencing reads to a reference sequence of interest, to select reads for further analysis.</td>
</tr>
<tr>
<td>Frequentist</td>
<td>An algorithm for finding SNPs (alternative to SNP Finding algorithm).</td>
</tr>
<tr>
<td>function code [dbSNPs]</td>
<td>A code in the dbSNP database that indicates the consequence, if any, of a SNP to the transcript in which it is located.</td>
</tr>
<tr>
<td>fusion transcript</td>
<td>An RNA molecule that results from transcription of a gene fusion. See also gene fusion</td>
</tr>
<tr>
<td>gapped alignment</td>
<td>A read alignment to the reference sequence that indicates an insertion or deletion. The pairing algorithm searches for gapped alignments (indels) when one of the tags (F3/R3/F5-P2) maps to the reference genome and the other tag does not map to the genome within the insert-size range. Small indel analysis calls indels from a consensus of gapped alignments, using the BAM file as input.</td>
</tr>
<tr>
<td>gene fusion</td>
<td>A section of the genome that maps to an exon from one gene followed by an exon from another gene. It can occur as the result of a translocation, deletion, or chromosomal inversion. A gene fusion junction excludes exon-to-exon boundaries that arise from alternative splicing of a transcript.</td>
</tr>
<tr>
<td>genomic classification table</td>
<td>For mate-paired or paired-end reads, a code that describes the strandedness, distance, and orientation of the two reads.</td>
</tr>
<tr>
<td>genomic mapping</td>
<td>See mapping.</td>
</tr>
<tr>
<td>genomics</td>
<td>Global analysis of the genome to discern elements involved in regulation of gene activity or expression, with an emphasis on genetic variation such as single nucleotide polymorphisms, small and large insertions and deletions, and other structural variants such as translocations and inversions. Some use the term genomics to as an umbrella term that includes transcriptomics, epigenomics, and analysis of the genome.</td>
</tr>
<tr>
<td>group</td>
<td>A collection of read-sets that are required to be analyzed together, in groups that you define. Reads that are grouped together are treated as one specimen, even if the reads are in different *.xsq input files.</td>
</tr>
<tr>
<td>hemizygous</td>
<td>A small indels tool, a parameter that indicates that the reference allele and the small indel are both present.</td>
</tr>
<tr>
<td>hg18, hg19</td>
<td>Reference sequence assemblies of the human genome using the nomenclature used by the UCSC Genome Browser.</td>
</tr>
<tr>
<td>highly expressed junctions</td>
<td>A whole transcriptome pipeline, high-count sequencing reads that span an exon-exon junction.</td>
</tr>
<tr>
<td>homozygous</td>
<td>In diploid organisms, having two identical alleles in the corresponding genes.</td>
</tr>
<tr>
<td>Term</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>HUGO-style gene names</td>
<td>Gene names following the Human Genome Organization style.</td>
</tr>
<tr>
<td>HuRef</td>
<td>A diploid human genome sequence of one individual (J. Craig Venter).</td>
</tr>
<tr>
<td>indel</td>
<td>A difference in sequence due to either an insertion or a deletion event; especially used when the evolutionary direction of the change is unspecified.</td>
</tr>
<tr>
<td>insert size</td>
<td>The physical size of the genomic DNA segments or RNA molecules represented in a SOLiD® library.</td>
</tr>
<tr>
<td></td>
<td>• Fragment libraries: the size of the sheared DNA fragments.</td>
</tr>
<tr>
<td></td>
<td>• Mate-paired libraries: the length of the genomic DNA segment spanned by the corresponding mate-pair tags.</td>
</tr>
<tr>
<td></td>
<td>• Whole transcriptome libraries: the size of the RNA fragments.</td>
</tr>
<tr>
<td>insertion</td>
<td>An insertion of nucleotide sequence with respect to the reference genome or sequence.</td>
</tr>
<tr>
<td>intron</td>
<td>The genomic sequence between two exons that is spliced out of a primary transcript prior to translation. See <em>exon</em>.</td>
</tr>
<tr>
<td>inversion</td>
<td>A segment of DNA that is in its native location but is in the reverse orientation.</td>
</tr>
<tr>
<td>junction</td>
<td>A place where two regions that are not contiguous in the genomic sequence are joined in a single sequenced region under consideration.</td>
</tr>
<tr>
<td>Junction Confidence Value (JCV) metric</td>
<td>Whole transcriptome pipeline, a statistical confidence metric to detect false positives and assign a confidence level to a detected junction.</td>
</tr>
<tr>
<td>junction mapping</td>
<td>Whole transcriptome pipeline, a mapping step that incorporates exon, gene, and transcript definitions in the genome annotation for that region.</td>
</tr>
<tr>
<td>junk seed</td>
<td>An initial alignment in a seed-and-extend algorithm that is not aligned with the correct sequence</td>
</tr>
<tr>
<td>large indel</td>
<td>An insertion or deletion, with respect to a reference sequence, of more than 200 bp.</td>
</tr>
<tr>
<td>LB field</td>
<td>A field in a BAM file that contains library type information.</td>
</tr>
<tr>
<td>library</td>
<td>A set of DNA or cDNA molecules prepared from the same biological specimen and prepared for sequencing on the SOLiD® System.</td>
</tr>
<tr>
<td>local mapping</td>
<td>The initial alignment step that starts with locating short matches between a read and the reference sequence.</td>
</tr>
<tr>
<td>locus-spanning</td>
<td>Pairs of reads derived from mate pairs or paired-end reads that span a relatively large distance in the reference sequence.</td>
</tr>
<tr>
<td><strong>Map Fragment data tool</strong></td>
<td>Tool that maps sequence data from fragment libraries.</td>
</tr>
<tr>
<td>----------------------------</td>
<td>------------------------------------------------------</td>
</tr>
<tr>
<td><strong>mappability</strong></td>
<td>In the CNVs analysis module, the fraction of mappable bases in the candidate CNV region.</td>
</tr>
<tr>
<td><strong>mapping</strong></td>
<td>The process of aligning sequencing reads to a reference genome or sequence.</td>
</tr>
<tr>
<td><strong>mate alignment</strong></td>
<td>The process of mapping paired-end or mate-paired reads to the reference sequence, taking into account the proximity of the reads on a template clone.</td>
</tr>
<tr>
<td><strong>mate-paired library</strong></td>
<td>Library consisting of two DNA segments that reside a known distance apart in the genome, linked by an internal adaptor, and with P1 and P2 Adaptors ligated to the 5' and 3' ends of the template strand, respectively.</td>
</tr>
<tr>
<td><strong>mates</strong></td>
<td>Sequencing reads that are linked through their origin in a mate-paired library or a paired-end sequencing run.</td>
</tr>
<tr>
<td><strong>mer</strong></td>
<td>The number of nucleotides in a sequencing read.</td>
</tr>
<tr>
<td><strong>merging (WTA)</strong></td>
<td>Algorithm parameter in paired-end WTA analysis that enables the merging of genomic and junction mappings.</td>
</tr>
<tr>
<td><strong>methylation analysis</strong></td>
<td>The study of how methylation of nucleic acids is involved in DNA structure and control of gene expression.</td>
</tr>
<tr>
<td><strong>module</strong></td>
<td>A virtual piece of LifeScope™ Software, such as mapping, SNP Finding, or CNV.</td>
</tr>
<tr>
<td><strong>NCBI</strong></td>
<td>National Center for Biotechnology Information.</td>
</tr>
<tr>
<td><strong>node</strong></td>
<td>A node is a single computer that is connected to the network. A multi-node cluster has several nodes (computers).</td>
</tr>
<tr>
<td><strong>normalization</strong></td>
<td>In general, the process of comparing an experimental measurement to a reference measurement.</td>
</tr>
<tr>
<td></td>
<td>In the CNV analysis module, the process of comparing the relative coverage of a region of interest to the global coverage, based on the human reference genome.</td>
</tr>
<tr>
<td></td>
<td>In targeted resequencing, the process of adjusting the amplicon amounts after PCR enrichment and before library preparation, so that amplicons are represented equally in the library.</td>
</tr>
<tr>
<td></td>
<td>In the whole transcriptome pipeline, the parameter RPKM is a normalized measure of gene expression.</td>
</tr>
<tr>
<td><strong>paired-end sequencing</strong></td>
<td>Sequencing runs that acquire sequence from each end of the insert in a DNA fragment or whole transcriptome library, using both forward and reverse reads.</td>
</tr>
<tr>
<td><strong>pairing</strong></td>
<td>An algorithm that occurs after mapping in the LifeScope™ Software workflow.</td>
</tr>
<tr>
<td>Term</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>pairing distances</td>
<td>Pairing distances (sometimes called insert sizes) for each pair are assigned during the mapping/pairing pipelines and subsequently used by the large indel tool to determine indel candidacy.</td>
</tr>
<tr>
<td>parallelization</td>
<td>A tertiary analysis tool by which you can split spectrum generation into multiple jobs, where each job generates a subspectrum from the subset of reads.</td>
</tr>
<tr>
<td>pipeline</td>
<td>Sets of steps or modules that are used in sequence for different data analysis goals, such as whole transcriptome analysis or methylation analysis.</td>
</tr>
<tr>
<td>ploidy</td>
<td>The number of chromosome sets in a cell.</td>
</tr>
<tr>
<td>polyploidy</td>
<td>The number of chromosome sets in a cell.</td>
</tr>
<tr>
<td>polymorphism</td>
<td>A genetic variant in a population of individuals that may or not may be associated with an observable (phenotypic) trait.</td>
</tr>
<tr>
<td>primer set</td>
<td>In the SOLiD® System, the set of primers that are used sequentially to initiate ligation sequence chemistry.</td>
</tr>
<tr>
<td>project</td>
<td>A project is a container for your analysis runs. You create your projects. Within each project, you create one or more analyses. Each project is private to an individual LifeScope™ Software user.</td>
</tr>
<tr>
<td>p-value</td>
<td>The statistical significance of an alignment score is frequently assessed by its p-value, which is the probability that this score or a higher one can occur simply by chance, given the probabilistic models for the sequences.</td>
</tr>
<tr>
<td>quality value</td>
<td>An empirically defined value based on a phred-like score equating to the confidence that the color called for that cycle is the correct one. In general, the brighter the bead, the greater the difference in signal between the primary and secondary colors, the higher the quality value (QV).</td>
</tr>
<tr>
<td>R3 tag</td>
<td>The R3 tag applies only to mate-paired libraries; sequencing data derived from the mate-pair tag closer to the P2 end of the SOLiD® templated bead, using forward ligation chemistry. The R3 tag initiates in the IA sequence using the SOLiD® FWD2 Seq. Primers. See the 5500 Series SOLiD® Sequencers: Reagents and Consumables Ordering Guide (Part no. 4465650) for an illustration.</td>
</tr>
<tr>
<td>raw reads</td>
<td>A format in which the nucleotide sequence appears without headers or comments.</td>
</tr>
<tr>
<td>read, sequencing read</td>
<td>Sequencing data from a single bead with a single primer set.</td>
</tr>
<tr>
<td>read-set</td>
<td>A group of reads belonging either to one .xsq file (in the case of non-indexed data) or to one barcode from one *.xsq file (in the case of indexed data).</td>
</tr>
<tr>
<td>read-set group</td>
<td>A collection of similar read-sets designated by a user.</td>
</tr>
</tbody>
</table>
read-set repository A storage place in LifeScope™ Software for instrument data intended to be input data for LifeScope™ Software analyses.

reference, reference genome, reference sequence A sequence against which sequencing reads are aligned before further bioinformatics analysis.

reference file A file containing the reference sequence information, usually in the *.fasta format.

RefSeq A multi-organism database archive of DNA, RNA, and protein sequences, hosted by the NCBI.

repository A virtual container with your reference files, reads files, and projects. Repositories include:
- Projects, which contains your projects.
- Reads (read-sets), which contains your *.xsq data files.
- References, which contains the reference genomes used in your analyses.

rescue An alignment method that is applied to read pairs that have at least one alignment, but no pair of alignments occurring within an expected range. Use the rescue method to find additional alignments.

rescue distance The maximum distance x for a pair considered to be properly paired (SAM flag 0x2) is calculated by solving equation Phil((x-mu)/sigma)=x/L*p0, where mu is the mean, sigma is the standard error of the insert size distribution, L is the length of the genome, p0 is prior of anomalous pair and Phi() is the standard cumulative distribution function.

resequencing The process of genomic sequencing in cases where a reference sequence already exists, and the new sequence is compared to the reference.

resource manager Software for executing batch and interactive jobs on a cluster of networked computers.

RNA-Seq Gene expression analysis using sequence-based approaches. RNA-Seq can include whole transcriptome analysis, small RNA analysis, and SOLiD® SAGE™ analysis.

RPKM The number of reads mapping to a transcript per kilobase of transcript length per million mappable reads. RPKM is used to set a threshold for calling a transcript or new RNA species or isoform “present.” 1 RPKM is equivalent to 20 reads mapping to a 1 kb transcript per 20 \( \times 10^6 \) mappable reads.

SAET SOLiD® Accuracy Enhancer Tool for correcting spectral-alignment errors in raw data; reduces color-calling error rate without alignment to reference genome. SAET takes as input a file with SOLiD® reads in .csfasta format and outputs corrected reads and quality values into new .csfasta and .qual files, respectively.
<table>
<thead>
<tr>
<th>Glossary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SAGE™ analysis</strong></td>
</tr>
<tr>
<td><strong>sample, barcoded sample</strong></td>
</tr>
<tr>
<td><strong>SAMtools</strong></td>
</tr>
<tr>
<td><strong>Satay plot</strong></td>
</tr>
<tr>
<td><strong>secondary analysis</strong></td>
</tr>
<tr>
<td><strong>seed</strong></td>
</tr>
<tr>
<td><strong>seed-and-extend</strong></td>
</tr>
<tr>
<td><strong>sense strand</strong></td>
</tr>
<tr>
<td><strong>sequence trimming</strong></td>
</tr>
<tr>
<td><strong>sequencing sample</strong></td>
</tr>
<tr>
<td><strong>single read</strong></td>
</tr>
<tr>
<td><strong>small indel</strong></td>
</tr>
<tr>
<td><strong>small RNA analysis</strong></td>
</tr>
<tr>
<td><strong>small RNA library</strong></td>
</tr>
<tr>
<td><strong>SNP</strong></td>
</tr>
</tbody>
</table>
SNP Finding: An analysis module to identify SNPs from mapped and processed SOLiD® System color space reads. Also called diBayes.

spectrum building: (SAET) When a temporary file with spectrum constructed from all reads generates in the output directory. This process is fast, resource less, and creates temporary swap files when memory usage is above 2GB.

splice junction: Exon-to-exon boundaries that arise from alternative splicing of a transcript. See also gene fusion.

splicing: The process whereby introns are removed from a primary mRNA, resulting in a mature mRNA that is ready for translation into protein.

strandedness: The polarity or orientation of a nucleic acid strand with respect to being sense or antisense. Libraries prepared using the SOLiD® Total RNA-Seq Kit preserve the strandedness of the original RNA molecule such that F3 tag reads align to the sense strand and F5 tag reads align to the antisense strand.

stringency: Stringency is one criterion used to filter out junk reads in ReadBuilder stage. Specifically, in the SNP Finding tool, the stringency parameter (call.stringency) defines the criteria by which to report SNPs in analyzing genomic data. There are several main empirical stringency settings (low_frequency, low, medium, high, highest).

tag: A tag is:
- Sequencing data from a single bead with a single primer set; sometimes used interchangeably with read, sequencing read.
- A length of DNA or cDNA to be sequenced; especially, a relatively short stretch of DNA or cDNA that is used to infer information about the longer native molecule from which it is derived, such as in mate-paired library sequencing and SAGE™ analysis, respectively.

tags: BC, F3, F5, R3: Sequencing data derived from specific locations on the SOLiD® templated bead. See the 5500 Series SOLiD® Sequencers: Reagents and Consumables Ordering Guide (Part no. 4465650) for an illustration.

targeted resequencing: Comparative sequence analysis of selected candidate genes or regions to discover genetic variants and mutations.

template strand: The strand of each DNA molecule that is covalently attached to SOLiD® P1 beads during templated bead preparation and that serves as a template for SOLiD® System sequencing.

tertiary analysis: Data analysis that takes place after mapping and alignment.

threading: The ability of an operating system to execute different parts of a program, called threads, simultaneously. A multi-threaded software program is designed so that all the threads can execute at the same time without interfering with each other.
<table>
<thead>
<tr>
<th>Glossary Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>transcript rescue distance</td>
<td>See rescue distance.</td>
</tr>
<tr>
<td>transcriptome</td>
<td>The compilation of all transcribed sequences from a genome, both coding and non-coding.</td>
</tr>
<tr>
<td>transition</td>
<td>A single nucleotide (point) mutation that changes a purine nucleotide to another purine nucleotide (for example, A to G), or a pyrimidine nucleotide to another pyrimidine nucleotide (for example T to C).</td>
</tr>
<tr>
<td>translocation</td>
<td>A mutation in which a chromosomal segment changes position, usually moving from one chromosome to a different, nonhomologous chromosome.</td>
</tr>
<tr>
<td>transversion</td>
<td>A single nucleotide (point) mutation that changes a purine nucleotide to a pyrimidine nucleotide (for example, A to C), or a pyrimidine nucleotide to a purine nucleotide.</td>
</tr>
<tr>
<td>trimming</td>
<td>Filtering noise algorithmically which removes 5’ overhangs and shortens 3’ overhangs, typically to 4-5 bases. Trimming reads requires both modifying the .csfasta reads file to replace the color to be trimmed with a dot (&quot;.&quot;), and modifying the mapping parameters to account for the alignment change.</td>
</tr>
<tr>
<td>trusted frequency [SAET]</td>
<td>(SAET) Use this developer option (-trustfreq freq) to overwrite estimated frequency cutoff of trusted seeds. All seeds with frequency &lt; “freq” are filtered out of spectrum.</td>
</tr>
<tr>
<td>trusted seed [SAET]</td>
<td>(SAET) If globally computed cutoff for frequency of trusted seeds does not meet your purpose, e.g., it is too low and too many junk seeds are considered correct or it is too high and many correct but low frequency seeds are filtered out, then use -trustfreq option to overwrite estimated frequency cutoff.</td>
</tr>
<tr>
<td>ungapped alignment</td>
<td>See gapped alignment.</td>
</tr>
<tr>
<td>uniquely placed reads</td>
<td>A read that is mapped only once in a genome with a given number of mismatches.</td>
</tr>
<tr>
<td>variant</td>
<td>A difference in the nucleotide sequence of interest, with respect to the reference sequence.</td>
</tr>
<tr>
<td>visualization</td>
<td>Viewing mapped reads on any number of publicly available genome browsers.</td>
</tr>
<tr>
<td>weighting</td>
<td>A weighting scheme can outperform a simple binary scheme traditionally applied in genomic analysis, independent of the organism, and used to improve the quality of the data.</td>
</tr>
<tr>
<td>whole transcriptome analysis</td>
<td>Global sequence analysis of coding and non-coding RNA transcripts along their entire length.</td>
</tr>
<tr>
<td>whole transcriptome library</td>
<td>A SOLiD® System-compatible library that is prepared from total or poly(A) RNA that enables sequence analysis of the transcripts along their entire length.</td>
</tr>
<tr>
<td>Term</td>
<td>Definition</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>workflow</td>
<td>Configuration files that are used for common end-to-end analysis runs, including multiple modules. LifeScope™ Software runs these common workflows: resequencing, targeted resequencing, low frequency variant, whole transcriptome, small RNA, Methyl Miner, and ChIP-Seq.</td>
</tr>
<tr>
<td>WTA</td>
<td>See <em>whole transcriptome analysis</em>.</td>
</tr>
<tr>
<td>*.xsq</td>
<td>eXtensible SeQuence, an extensible file format for storing sequence data. A binary sequence output file generated by the instrument. This file contains primary analysis results for a single lane in a flowchip.</td>
</tr>
<tr>
<td>z-normalize</td>
<td>Parameters called moving statistics, such as the number of locus-spanning pairs, and the sample average insert size, are calculated from a subset of pairs at each genomic position, yielding the absolute insert size deviation between the sample and the population in units of standard deviation. This normalization step allows multiple libraries with variable insert sizes to be combined into one analysis.</td>
</tr>
<tr>
<td>zygosity</td>
<td>The combination of alleles at a site in a nucleotide sequence; for example, homozygous reference allele, homozygous non-reference allele, or heterozygous.</td>
</tr>
</tbody>
</table>
Glossary
Related documentation

For the latest documentation on LifeScope™ Genomic Analysis Software, go to:


<table>
<thead>
<tr>
<th>Document</th>
<th>Part number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LifeScope™ Genomic Analysis Software User Guide</td>
<td>4476537</td>
<td>Describes the bioinformatics analysis framework for flexible application analysis (data-generated mapping, SNPs, count reads, etc.) from sequencing runs.</td>
</tr>
</tbody>
</table>

Obtaining support

For the latest services and support information for all locations, go to:

www.appliedbiosystems.com

At the Applied Biosystems website, you can:

• Access worldwide telephone and fax numbers to contact Applied Biosystems Technical Support and Sales facilities.
• Search through frequently asked questions (FAQs).
• Submit a question directly to Technical Support.
• Order Applied Biosystems user documents, SDSs, certificates of analysis, and other related documents.
• Download PDF documents.
• Obtain information about customer training.
• Download software updates and patches.
### Numerics

<table>
<thead>
<tr>
<th>Character</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2BE</td>
<td>494</td>
</tr>
<tr>
<td>4BE</td>
<td>494</td>
</tr>
</tbody>
</table>

### A

<table>
<thead>
<tr>
<th>Administrative shell commands</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>88</td>
<td>565</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Alignment</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>579</td>
<td></td>
</tr>
<tr>
<td>Alignment browser</td>
<td>579</td>
</tr>
<tr>
<td>Alignment format</td>
<td>493</td>
</tr>
<tr>
<td>Alignment score</td>
<td>579</td>
</tr>
<tr>
<td>Annotation-aided alignment</td>
<td>361</td>
</tr>
<tr>
<td>Basic alignment information</td>
<td>493</td>
</tr>
<tr>
<td>Color</td>
<td>506, 507</td>
</tr>
<tr>
<td>Format</td>
<td>494</td>
</tr>
<tr>
<td>Gapped</td>
<td>139, 363</td>
</tr>
<tr>
<td>Gapped alignment</td>
<td>583</td>
</tr>
<tr>
<td>Mate</td>
<td>585</td>
</tr>
<tr>
<td>Nucleotide sequence</td>
<td>34</td>
</tr>
<tr>
<td>Score</td>
<td>506</td>
</tr>
<tr>
<td>Unmapped and secondary</td>
<td>494</td>
</tr>
<tr>
<td>Viewing and inspection</td>
<td>48, 507</td>
</tr>
</tbody>
</table>

| Allele                       | 579  |

<table>
<thead>
<tr>
<th>Analysis</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>579</td>
<td></td>
</tr>
<tr>
<td>Methylation</td>
<td>585</td>
</tr>
<tr>
<td>SAGE</td>
<td>588</td>
</tr>
<tr>
<td>Secondary</td>
<td>588</td>
</tr>
</tbody>
</table>

| Analysis groups              | 96   |

| Analysis type                | 579  |

| Analysis.regions.file        | 560  |
| Example command              | 76   |
| With small indel module      | 237  |

| Annotated gene               | 579  |

| Annotation                   | 287, 579 |
| CNV output file              | 312     |
| Conflicts                    | 288     |
| Custom dbSNP build           | 293     |
| Custom reference             | 293     |
| dbSNP 130 build              | 288     |
| dbSNP 131 build              | 288     |
| dbSNP files                  | 295     |

| dbsNP tables                 | 288   |
| Download dbSNP files         | 294   |
| Ensembl GTF files            | 294   |
| Error handling               | 319   |
| Filtered variants output files | 304   |
| Filtering options            | 288   |
| Function codes               | 301   |
| Gene features                | 301   |
| HG18 reference build         | 288   |
| HG19 reference build         | 288   |
| Input files                  | 295   |
| Large indels output file     | 310   |
| Memory requirements          | 297   |
| Mutated genes output file    | 313   |
| Optional annotation sources  | 292, 293 |
| Options                      | 288   |
| Output annotation            | 300   |
| Parameters table             | 297   |
| RefGene.txt                  | 293   |
| Small indels annotated output file | 307 |
| SNPs output file             | 305, 315, 317 |
| Sources                      | 295   |
| Supported attributes         | 318   |
| Transitions and versions     | 290   |
| UCSC GTF file                | 293   |
| Variant overlapping gene or exon | 288 |
| Variant statistics output file | 305, 310, 312 |
| Variants appearing in dbSNP  | 289   |
| Variants in exons            | 289   |
| Variants in genes            | 289   |

| Annotation parameters        | 297   |
| Annotation parameters table  | 297   |
| Filtering parameters         | 298   |
| Source parameters            | 297   |

| Authentication                | 42    |

### B

<p>| BAM file                     | 579   |
| @RG header field required    | 499   |
| Header requirement           | 498   |
| Indel alignments             | 508   |</p>
<table>
<thead>
<tr>
<th>term</th>
<th>page numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>pairing information</td>
<td>505</td>
</tr>
<tr>
<td>Picard validation</td>
<td>499</td>
</tr>
<tr>
<td>position errors</td>
<td>202</td>
</tr>
<tr>
<td>with samtools command</td>
<td>498</td>
</tr>
<tr>
<td>WT format differences</td>
<td>377</td>
</tr>
<tr>
<td>WT optional fields</td>
<td>377</td>
</tr>
<tr>
<td>bamgen parameters</td>
<td>140, 367, 430</td>
</tr>
<tr>
<td>bamgen.mqv.threshold</td>
<td>140, 356, 430</td>
</tr>
<tr>
<td>BAMStats</td>
<td>452</td>
</tr>
<tr>
<td>megabases of coverage</td>
<td>169, 387, 447</td>
</tr>
<tr>
<td>NumFragmentsPassingFilters</td>
<td>168</td>
</tr>
<tr>
<td>NumFragmentsTotal</td>
<td>168</td>
</tr>
<tr>
<td>NumMapped</td>
<td>168</td>
</tr>
<tr>
<td>output</td>
<td>161, 163, 380, 441</td>
</tr>
<tr>
<td>output description</td>
<td>168, 387, 447</td>
</tr>
<tr>
<td>output files</td>
<td>382, 443</td>
</tr>
<tr>
<td>overview</td>
<td>159, 379, 439</td>
</tr>
<tr>
<td>parameters table</td>
<td>160, 161, 379, 380, 439, 440</td>
</tr>
<tr>
<td>report files</td>
<td>163</td>
</tr>
<tr>
<td>barcode</td>
<td>86, 140, 579</td>
</tr>
<tr>
<td>barcode group</td>
<td>579</td>
</tr>
<tr>
<td>barcode pool</td>
<td>579</td>
</tr>
<tr>
<td>barcoded library</td>
<td>579</td>
</tr>
<tr>
<td>barcoded sample</td>
<td>588</td>
</tr>
<tr>
<td>See also index ID</td>
<td></td>
</tr>
<tr>
<td>base calls</td>
<td>452</td>
</tr>
<tr>
<td>base-space analysis</td>
<td>579</td>
</tr>
<tr>
<td>base-translation algorithm</td>
<td>See refcor</td>
</tr>
<tr>
<td>Bayesian</td>
<td>580</td>
</tr>
<tr>
<td>bead</td>
<td>580</td>
</tr>
<tr>
<td>BED file</td>
<td>580</td>
</tr>
<tr>
<td>BED format</td>
<td>418, 474, 510</td>
</tr>
<tr>
<td>bedgraph</td>
<td>338, 341, 580</td>
</tr>
<tr>
<td>Binary Alignment Map</td>
<td>34</td>
</tr>
<tr>
<td>bisulfite reads</td>
<td>478, 580</td>
</tr>
<tr>
<td>BLAST</td>
<td>580</td>
</tr>
<tr>
<td>browser, supported versions</td>
<td>60</td>
</tr>
</tbody>
</table>

**C**

<table>
<thead>
<tr>
<th>term</th>
<th>page numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>call stringency</td>
<td>580</td>
</tr>
<tr>
<td>calling threshold</td>
<td>580</td>
</tr>
<tr>
<td>character restrictions</td>
<td>73</td>
</tr>
<tr>
<td>ChIP-Seq</td>
<td>471, 580</td>
</tr>
<tr>
<td>analysis tools</td>
<td>471</td>
</tr>
<tr>
<td>BAM-to-BED format converter</td>
<td>474</td>
</tr>
<tr>
<td>ChIP-Seq analysis software tools</td>
<td>475</td>
</tr>
<tr>
<td>ChIP-Seq Map Data</td>
<td>471</td>
</tr>
</tbody>
</table>

**instructions** 471

**chromosome files** 513

**CIGAR string** 378

**Circos** 410, 414, 419

**clear zone** 580

**clipping** 580

<table>
<thead>
<tr>
<th>term</th>
<th>page numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>H operation</td>
<td>506</td>
</tr>
<tr>
<td>hard</td>
<td>506</td>
</tr>
<tr>
<td>hard clipping</td>
<td>506, 507</td>
</tr>
<tr>
<td>hard clipping example</td>
<td>506</td>
</tr>
<tr>
<td>refcor parameter</td>
<td>430</td>
</tr>
<tr>
<td>splice finder parameter</td>
<td>409</td>
</tr>
<tr>
<td>WT clipped records</td>
<td>378</td>
</tr>
<tr>
<td>WT soft-clips</td>
<td>378</td>
</tr>
<tr>
<td>clone</td>
<td>271, 276, 580</td>
</tr>
</tbody>
</table>

**CNV**

<table>
<thead>
<tr>
<th>term</th>
<th>page numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>algorithm description</td>
<td>217</td>
</tr>
<tr>
<td>CG correction</td>
<td>217</td>
</tr>
<tr>
<td>coverage</td>
<td>218</td>
</tr>
<tr>
<td>FAQ</td>
<td>220</td>
</tr>
<tr>
<td>output file examples</td>
<td>216</td>
</tr>
<tr>
<td>output file format</td>
<td>215</td>
</tr>
<tr>
<td>output files</td>
<td>214</td>
</tr>
<tr>
<td>CNV parameters</td>
<td>211</td>
</tr>
<tr>
<td>CNV parameters table</td>
<td>211</td>
</tr>
<tr>
<td>CNV stringency settings</td>
<td>211</td>
</tr>
<tr>
<td>color balance</td>
<td>580</td>
</tr>
<tr>
<td>color call</td>
<td>580</td>
</tr>
<tr>
<td>color space</td>
<td>260, 581</td>
</tr>
<tr>
<td>analysis in color space</td>
<td>581</td>
</tr>
<tr>
<td>color attribute tags</td>
<td>504</td>
</tr>
<tr>
<td>color quality value</td>
<td>581</td>
</tr>
<tr>
<td>refcor color sequence parameter</td>
<td>433</td>
</tr>
</tbody>
</table>

**command shell**

<table>
<thead>
<tr>
<th>term</th>
<th>page numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>commands table</td>
<td>81</td>
</tr>
<tr>
<td>common analysis scenarios</td>
<td>72</td>
</tr>
<tr>
<td>concepts</td>
<td>63, 64</td>
</tr>
<tr>
<td>grouped analysis</td>
<td>93</td>
</tr>
<tr>
<td>groupsfile</td>
<td>93</td>
</tr>
<tr>
<td>help</td>
<td>89</td>
</tr>
<tr>
<td>man page</td>
<td>89</td>
</tr>
<tr>
<td>modes</td>
<td>77</td>
</tr>
<tr>
<td>modes of operation</td>
<td>78</td>
</tr>
<tr>
<td>naming restrictions</td>
<td>68</td>
</tr>
<tr>
<td>options</td>
<td>89</td>
</tr>
<tr>
<td>reference files</td>
<td>70</td>
</tr>
<tr>
<td>repositories</td>
<td>68</td>
</tr>
<tr>
<td>review job status</td>
<td>99</td>
</tr>
<tr>
<td>review results</td>
<td>100</td>
</tr>
</tbody>
</table>
run mode 78
run mode commands 79
scripted mode 80
shell mode 81
status mode 79
terminology 63, 64
complexity 581
concordant 581
configuration files 553
consensus calls 581
consensus_calls file 34, 192, 197, 204, 207
contig 581
cvtColors.sh 520
copy number variation (CNV) 581
Count Known Exons 402
  algorithm description 403
  parameters 402
counting 581
Counts tool 402, 405, 415
CountTags 581
CountTags tool. See also Count Known Exons
coverage 581
csfasta 34, 504
csfasta file 581

dbSNP 581, 583
  annotation files 297
  build 131 297
de novo sequencing 581
deletion 581
demo examples
  instructions 51
  not recommended for new users 51
  optional 51
diBayes 182, 201, 202, 204
dibayes.call.stringency 190
discordant 582
distance rescue 587
documentation, related 593

E
EBI 582
ECC data 494
  not supported on PE R3 tag 494
  with SAET 429
  email notification 561
  enriched data 582
  enrichment
    in the targeted resequencing workflow 335
    regions of interest file 336
  enrichment parameters
    enrichment parameters table 337
    enrichment statistics parameters 338
  enrichment statistics 582
    aligned reads input 340
    genome coverage frequency file 342
    input files 339
    output alignment file 341
    output bedgraph file 341
    output files 340
    target coverage frequency file 343
    target regions file 339
    target statistics report file 343
Ensembl 582
  Ensembl GTF file 293, 294, 355, 550
  reformat_ensembl_gtf.pl 294
error correction 127, 582
  See also SAET
Error expectation metric (EEM) 416, 582
evidence graph 582
Example steps to run a standard workflow 76
examples of INI files 555
exon 407, 582
  exon mapping 359
  Exon-1 417
  Exon-2 417
  exon-exon boundaries 407
exon mapping 582

F
F3 tag 582
F5 tag 582
false positive 582
FAQ 485
  CNV 220
  general 485
  large indels 284
  mapping 173
  pairing 176
  small indels 267
  SNPs 204
  file formats 34, 192, 493, 505
  file types 34
<table>
<thead>
<tr>
<th>Term</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter</td>
<td>583</td>
</tr>
<tr>
<td>filter mapping</td>
<td>583</td>
</tr>
<tr>
<td>filter reference fasta</td>
<td>34</td>
</tr>
<tr>
<td>fragmap.max.number.of.jobs</td>
<td>144, 367, 434</td>
</tr>
<tr>
<td>fragmap.maxreads.per.node</td>
<td>144, 368, 434</td>
</tr>
<tr>
<td>fragmap.minreads.per.node</td>
<td>144, 367, 434</td>
</tr>
<tr>
<td>fragmap.processors.per.node</td>
<td>144</td>
</tr>
<tr>
<td>framework parameters</td>
<td></td>
</tr>
<tr>
<td>general parameters</td>
<td>559</td>
</tr>
<tr>
<td>read-only parameters</td>
<td>560</td>
</tr>
<tr>
<td>reference-related parameters</td>
<td>431, 559</td>
</tr>
<tr>
<td>with the set reference command</td>
<td>431, 559</td>
</tr>
<tr>
<td>Frequentist</td>
<td>583</td>
</tr>
<tr>
<td>function code (dbSNPs)</td>
<td>583</td>
</tr>
<tr>
<td>fusion junction</td>
<td>350, 414</td>
</tr>
<tr>
<td>fusion transcript</td>
<td>583</td>
</tr>
<tr>
<td>fusion, gene</td>
<td>583</td>
</tr>
<tr>
<td>G</td>
<td></td>
</tr>
<tr>
<td>gap alignment</td>
<td>139, 256, 258, 261, 363</td>
</tr>
<tr>
<td>detection</td>
<td>254, 256</td>
</tr>
<tr>
<td>gapped alignment</td>
<td>583</td>
</tr>
<tr>
<td>gene</td>
<td>550</td>
</tr>
<tr>
<td>gene annotations</td>
<td>353</td>
</tr>
<tr>
<td>gene orientation</td>
<td>349</td>
</tr>
<tr>
<td>HUGO-style gene names</td>
<td>551</td>
</tr>
<tr>
<td>gene fusion</td>
<td>583</td>
</tr>
<tr>
<td>genome</td>
<td></td>
</tr>
<tr>
<td>annotations</td>
<td>354, 512</td>
</tr>
<tr>
<td>reference</td>
<td>587</td>
</tr>
<tr>
<td>genomic browser</td>
<td></td>
</tr>
<tr>
<td>IGV 481, 507</td>
<td></td>
</tr>
<tr>
<td>UCSC Genome Browser 507</td>
<td></td>
</tr>
<tr>
<td>genomic classification</td>
<td>583</td>
</tr>
<tr>
<td>classification table</td>
<td></td>
</tr>
<tr>
<td>genomic classifications</td>
<td>176, 392</td>
</tr>
<tr>
<td>genomic mapping</td>
<td>583</td>
</tr>
<tr>
<td>genomics</td>
<td>583</td>
</tr>
<tr>
<td>get INI files</td>
<td>83, 115</td>
</tr>
<tr>
<td>get PLN files</td>
<td>83, 115</td>
</tr>
<tr>
<td>group</td>
<td>583</td>
</tr>
<tr>
<td>BAM @RG field</td>
<td>498</td>
</tr>
<tr>
<td>described</td>
<td>63, 65</td>
</tr>
<tr>
<td>group BAM files</td>
<td>96</td>
</tr>
<tr>
<td>read-set groups</td>
<td>85</td>
</tr>
<tr>
<td>running a grouped analysis</td>
<td>93</td>
</tr>
<tr>
<td>set group command</td>
<td>85</td>
</tr>
<tr>
<td>groupsfile</td>
<td>65, 95, 96</td>
</tr>
<tr>
<td>GTF files</td>
<td>512</td>
</tr>
<tr>
<td>annotation source</td>
<td>288</td>
</tr>
<tr>
<td>converting</td>
<td>551</td>
</tr>
<tr>
<td>GUI guide</td>
<td>24, 593</td>
</tr>
<tr>
<td>H</td>
<td></td>
</tr>
<tr>
<td>hard clipping</td>
<td>580</td>
</tr>
<tr>
<td>hemizygous</td>
<td>583</td>
</tr>
<tr>
<td>hg18</td>
<td></td>
</tr>
<tr>
<td>and the CNV tool</td>
<td>220</td>
</tr>
<tr>
<td>annotations based on hg18</td>
<td>292</td>
</tr>
<tr>
<td>genomic reference download from UCSC</td>
<td>512</td>
</tr>
<tr>
<td>hg18 reference build</td>
<td>288</td>
</tr>
<tr>
<td>human hg18 reference</td>
<td>284</td>
</tr>
<tr>
<td>hg18, hg19</td>
<td>583</td>
</tr>
<tr>
<td>hg19</td>
<td></td>
</tr>
<tr>
<td>annotations based on hg19</td>
<td>292</td>
</tr>
<tr>
<td>memory requirements</td>
<td>297</td>
</tr>
<tr>
<td>highly expressed junctions</td>
<td>583</td>
</tr>
<tr>
<td>homozygous</td>
<td>583</td>
</tr>
<tr>
<td>HUGO-style gene names</td>
<td>584</td>
</tr>
<tr>
<td>HuRef 584</td>
<td></td>
</tr>
<tr>
<td>hyper-threading</td>
<td>486</td>
</tr>
<tr>
<td>hyphen, restrictions</td>
<td>68</td>
</tr>
<tr>
<td>I</td>
<td></td>
</tr>
<tr>
<td>IGV viewer</td>
<td>418, 481, 507, 510</td>
</tr>
<tr>
<td>import statement restriction for global.ini files</td>
<td>556</td>
</tr>
<tr>
<td>increase Java heap space</td>
<td>487</td>
</tr>
<tr>
<td>indel</td>
<td>584</td>
</tr>
<tr>
<td>large</td>
<td>584</td>
</tr>
<tr>
<td>small</td>
<td>588</td>
</tr>
<tr>
<td>indel alignments</td>
<td>508</td>
</tr>
<tr>
<td>index ID</td>
<td>92</td>
</tr>
<tr>
<td>in a groupsfile</td>
<td>91, 93</td>
</tr>
<tr>
<td>in BAM file name</td>
<td>159</td>
</tr>
<tr>
<td>shown by the shell cat command</td>
<td>86</td>
</tr>
<tr>
<td>shown by the shell ls command</td>
<td>82</td>
</tr>
<tr>
<td>when specifying input data</td>
<td>90, 131</td>
</tr>
<tr>
<td>with the add xsq command</td>
<td>84</td>
</tr>
<tr>
<td>with the set groups command</td>
<td>85</td>
</tr>
<tr>
<td>INI files</td>
<td></td>
</tr>
<tr>
<td>analysis module INI files</td>
<td>555</td>
</tr>
<tr>
<td>examples</td>
<td>555, 558</td>
</tr>
<tr>
<td>explained</td>
<td>555, 558</td>
</tr>
<tr>
<td>import statement</td>
<td>556</td>
</tr>
<tr>
<td>import statement restriction</td>
<td>556</td>
</tr>
</tbody>
</table>


parameters 558
run parameters 556
insert size 584
insertion 584
installation
  authentication realms 53
  cluster 42
  data drive 540
  demos as installation verification 51
  hardware configuration 42
  instructions 43
  uninstall 55
  user accounts 53
  user authentication realms 42
  verification 50
  verification tests 52
  workstation 42
inversion 584
  ambiguous tag placement 232
  breakpoints 231
  candidate breakpoints 233
  GFF file format 229
  GFF output file format 227
  inversion polymorphisms 230
  inversion thresholds 234
  inversion tool workflow 232
  inverted mate-pair 231
  inverted mates 232
  output files 229
  rank.txt file format 229
  ranking 234
  rescore.txt file format 230
  scoring 233
  small inversion detection 232
  tiny inversions 234
  window size 233
inversion parameters
  inversion threshold parameters 234
  inversion.breakpoint.peak.width 233
  inversion.breakpoint.rescue 233
  inversion.max.bxx.mp.length 234
  inversion.max.inversion.length 234
  inversion.max.length.tiny.inversions 234
  inversion.min.pairing.quality 232
  inversion.min.qv 232
  inversion.recover.tiny.inversions 232, 234
J
  JCV metric 584
  job queues 486
  junction 584
  splice 589
  Junction Confidence Value (JCV) metric 415, 584
  junction mapping 584
  junk seed 584
L
  large indel 584
    candidate deviations 278
    candidate indels 278
    detection 274
    determining zygosity 279, 280
    FAQ 284
    module flow 275
    output file formats 272
    pairing distances 275
    parameter optimization 280
    parameter table 271
    run time 285
    space requirements 285
    zygosity 271
  large indel parameters 270
  parameter table 271
  resource parameters 272
LB field 584
  legacy format translation
    @HD SO field 515
    @RG LB field 515
    @SQ UR field 515
    ##color-code 515
    ##history 515
    ##line-order 515
    ##max-num-mismatches 515
    ##max-read-length 515
    ##primer-base 515
    ##reference-name 515
library 584
  mate-paired 585
  small RNA 588
  LifeScope™ Software URL 54
  local mapping 584
  locus-spanning 584
  log files 489
  low frequency variant detection 329
Index

read selection algorithm 331
running analyses 330
SNPs optimizations 332
SNPs parameters 188
supported analyses 334
lscope.sh 77
commands 81
options 89
usage 78

M
mail command, for completion notification 561
Map Fragment data tool 585
mappability 585
mapping 585
FAQ 173
filter 583
frag mapping of paired data 146
genomic 583
global mapping 580
junction 584
local 584
local alignment 143, 148, 150, 151, 173, 176
mapping quality value 138
performance 144, 367, 434
quality 403, 516
repetitive scheme 174, 450
schema lines 144, 368, 434
split into multiple jobs 144, 367, 434
mapping parameters
fragmap.mi.reads.per.node 144, 367, 434
fragmap.minreads.per.node 144, 367, 434
fragmap.number.of.nodes 144, 367, 434
mapping parameters table 142, 145
mapping.memory 142, 144, 368, 433, 434
mapping.min.reads 142, 366, 372, 432
parallelization 144
mapping statistics
See BAMStats
mark duplicates
compared to Picard 178
module 136
process.one.bamfile.only parameter 136
matching file,.ma 585
mate alignment 585
mate-paired library 585
mates 585
MaToBam 34

max.head.node.java.heap.space property 487
memory.request 125, 189, 213, 226, 241, 272, 299,
339, 380, 398, 402, 411, 491
annotations 299
CNV 213
inversions 226
large indel 272
SAET 125
small indel 241
small RNA count 464
small RNA coverage 458
small RNA mapping 440
SNPs 189
targeted resequencing 339
whole transcriptome count 402
whole transcriptome coverage 398
whole transcriptome mapping 380
whole transcriptome splice finder 411
mer 585
merging (WTA) 585
methylation analysis 585
MethylMiner™ 477
Methyl Map Data 477, 479
output files 481
sample workflow commands 479
mismatches
allowed in the seed 152, 370, 437
color space 505
dicolor 183, 201
mismatch penalty 176, 392
number of mismatches 229, 244, 245
mixed library types, tertiary only 117
module 585
module properties files 557
multi-group tertiary analysis 96

N
naming restrictions 73
NCBI 288, 542, 544, 545, 585
annotation sources 296
dbSNP data 295
node 585
normalization 585

O
order requirements for shell commands 111
output files location 53
paired-end sequencing 585
pairing 134, 585
"SV" output filter 232
distances 586
FAQ 176
mate-pair rescue 135
paired-end tags example 136
pairing quality 138, 362
pairing quality value 416
quality 516
uniqueness 176, 392
pairing quality value 138
parallelization 586
parameter table 270
PAS files 241, 268
   legacy format, not used 241
   PAS file format 509
PAS.SUM file 241, 252, 259
penalty 586
Phred-scale 362, 416
pipeline 586
ploidy 586
polymorphism 586
polyploidy 586
position error 202
PQV 138, 139, 280, 361, 362, 363, 416
primary alignment 138
primer set 586
probe error 202
processors.per.node 368, 434
project 586

quality
   bamgen.mqv.threshold parameter 140, 365, 430
   mapping quality values 140, 364, 430
   quality alignments 140, 364, 365, 430
   QV preferred over uniqueness 140, 364, 430
   quality value 586

R3 tag 586
raw reads 586
read 586
   sibling 588
   single 588
read repository index 89, 565
read-set 586
read-set group 586
read-set repository 587
rebuild command 89, 565
reftcor 139, 363, 429
   clipping parameter 430, 437
   reference weight 430, 434, 452
reftcor parameters 140, 364, 430
Reference 434
reference
   files 587
   fasta 34
   one maximum per analysis run 86
   sequence 587
Reference Assisted Base Translation. See reftcor
reference files 34, 70
   conflicts 540
   fasta 34
   filter reference FASTA 34
   multi-fasta 513
   overview 512
   reference file types 513
   validation 550
reference repository 540
   add new reference files 41
   initial content 41
   location 540
reference SNP identifier 288, 296
refGene.txt 293
refgene2gff.sh 293
reformat_ensembl_gtf.pl 551
RefSeq 587
refSNP identifier 288, 296
regions of interest file 111
repository 587
requirements 38
rescue 587
distance 587
resequencing 587
   mapping overview 129
   standard workflows 103
resource manager 587
restart an analysis 98
results directory location 97
results files location 53
Index

resume 98
reuse an analysis 98
RNA library, small 588
RNA sequencing 349
RNA-Seq 587
RPKM 587
RPKM metric 405, 415
rsID 288, 296
run completion 486, 561

S
SAET
error correction 127
spectrum building 127
usage 121
SAET parameters
SAET parameters table 123
saet.trustfreq 125
SAGE analysis 588
sample 588
sample, sequencing 588
samtools 588
fillmd command 516
index command 507
sort command 507
view command 498
SASR. See splice finder
Satay plot 588
scratch directory 487
secondary analysis 588
secondary processing
quality values 140, 364, 430
unique alignments 141, 365, 431
seed 127, 506, 588
allowed mismatches 152, 370, 437
anchoring 174, 450
for an application 174, 450
junk 125, 584
low-frequency 125
picking seed parameters 174, 450
seed-extend 506
shorter seeds 174, 450
start site of 152, 370, 437
trusted 125
seed-and-extend 588
sense strand 588
sequence
trimming 588
sequence, reference 587
sequencing
paired-end 585
sequencing read 586
sequencing sample 588
Serial Analysis of Gene Expression 588
server
check status 47, 61
how to start 47
how to stop 55
set reference command 431, 559
shell commands
concepts 64
example 76
syntax table 81
table 81
sibling read 588
Single Nucleotide Polymorphisms 588
Single Nucleotide Polymorphisms. See also SNPs
single read 588
small indel 253, 588
algorithm 254
ALIGN file 251, 252
allele calling examples 264
allele calls 262
ambiguous insertion example 265
anchor 254
caller heuristics 259
clipped coverage 244
color space 260
coverage ratio 263
detection 235, 253
deed and insertion ranges 255
example commands 236
FAQ 267
gap alignment detection 254, 256
GFF file format 242
hemizygous 244, 263
hemizygous call 263
indel size determination 262
local alignment strategy 257
multiple inserted alleles example 266
one-end anchored 254
output file example 245
output file formats 241
overview 235
pileups 251, 258, 260
TXT file format 249
with targeted resequencing 326
zygosity call 244
small indel parameters 237
indel size filtering 241
indel.max.mismatches 254
indel.min.non-matched.length 254
pairing.indel.max.mismatch.tag1 255
pairing.indel.max.mismatch.tag2 255
resource parameters 241
small indel parameters table 237
small.indel.colorspace.compatibility.level 260
small.indel.consGroup 258
small.indel.detail.level 258
small.indel.genomic.region 267
small.indel.indel.size.distribution.allowed 263
small.indel.max.coverage.ratio 260
small.indel.max.num.evid 258
small.indel.min.map.qv 258
small.indel.min.mapping.quality 258
small.indel.min.non.matched.length 258
small.indel.min.num.evid 258
small.indel.zygosity.profile.name 263
small RNA
FAQ 449
mapping stages of mapping 426
overview 423
small RNA counts 462
gff output file description 466
input files 462
internal parameters table 464
output files 465
parameters table 464
precursor sequence 463
small RNA coverage 457
BAM metadata 398
coverage per chromosome parameter 399
input files 397, 458
output files 399, 459
parameters table 398, 458, 459
RNA.coverage.per.chromosome parameter 459
small RNA library 588
small RNA mapping 423
components 424
input files 425
miRBase2Genome 428
output files 438
parameters table 432
to known miRBase sequences 428
SNP 588
SNP Finding 589
SNPChrPosOnRef 294
SNPContigLoc 294
SNPContigLocusId 294
SNPs
algorithm 201
call stringency settings 206
consensus calls file 196
consensus calls file flags 197
Consensus_Calls.txt output file format 197
FAQ 204
filtered reads 207
filtering reads 191
frequentist algorithm 201
gff3 output file format 194
multiple BAM files as input 182
output file formats 192
with low frequency variant detection 334
with targeted resequencing 326
SNPs parameters 185
call stringency 190
dibayes.call.stringency 186, 190
dibayes.het.skip.high.coverage 191
dibayes.reads.min.mapping.qv 191
het.skip.high.coverage 326, 334
SNPs parameters table 185
stringency and low frequency variant workflows 188
SNPs parameters table 185
stringency and low frequency variant workflows 188
SOLiD® Accuracy Enhancer Tool. See SAET 587
spectrum building 589
splice finder
evidence evaluator 412
evidence graph 412
output 414, 419
overview 407
parameters 408
parameters table 408
remaps 409
statistics 414
splice finder parameters 408
splice junction 589
splicing 589
standard workflows 30, 103, 104
example commands 76
strandedness 589
stringency 589
system requirements 38
Index

T

tag 589
F3 582
F5 582
R3 586
targeted resequencing
  algorithm 325, 331
  analysis.regions.file required 337, 560
  barcode support 324, 330
  coverage data 344
  overview 323, 335
  regions of interest file required 337
  SAET 326, 327
  supported analyses 323, 326
  target coverage bedgraph file 341
termology 63, 64
tertiary analysis
  example commands 95
  tertiary-only analysis 95
  with a groupsfile 95
  with grouped BAM files 96
TORQUE 38, 489
trimming 494
troubleshooting 490

U

UCSC Genome Browser 354, 418, 507, 510
unique alignments 141, 365, 431
Unique-PR 417
Unique-SR 417
UNIX commands, mail 561
user accounts 42, 53
user authentication realms 42

V

Variant Call Format (VCF) 193, 248, 513

W

whole transcriptome 349, 401, 407
  exon mapping 359
  pairing 361
  rescue 359
  splice finder 407
  whole transcriptome analysis 349
  whole transcriptome coverage 397
WTA merging 585

X

XSQ converter 520
XSQ file format 494, 495
  library details table 495
tag details table 496
XSQ files
  5500 instrument data 68
  in reads repository 63
  input to mapping 29
  the add xsq command 84

Z

zygosity 237, 244, 251, 254, 263, 268, 273, 275, 279, 280, 285