CASAVA Software Version 1.6
User Guide
Notice

This document and its contents are proprietary to Illumina, Inc. and its affiliates (“Illumina”), and are intended solely for the contractual use of its customers and for no other purpose than to use the product described herein. This document and its contents shall not be used or distributed for any other purpose and/or otherwise communicated, disclosed, or reproduced in any way whatsoever without the prior written consent of Illumina, Inc.

For the proper use of this product and/or all parts thereof, the instructions in this document must be strictly and explicitly followed by experienced personnel. All of the contents of this document must be fully read and understood prior to using the product or any of the parts thereof.

FAILURE TO COMPLETELY READ AND FULLY UNDERSTAND AND FOLLOW ALL OF THE CONTENTS OF THIS DOCUMENT PRIOR TO USING THIS PRODUCT, OR PARTS THEREOF, MAY RESULT IN DAMAGE TO THE PRODUCT, OR PARTS THEREOF, AND INJURY TO ANY PERSONS USING THE SAME.

RESTRICTIONS AND LIMITATION OF LIABILITY

This document is provided “as is,” and Illumina assumes no responsibility for any typographical, technical or other inaccuracies in this document. Illumina reserves the right to periodically change information that is contained in this document and to make changes to the products, processes, or parts thereof described herein without notice.

Illumina does not assume any liability arising out of the application or the use of any products, component parts, or software described herein. Illumina does not convey any license under its patent, trademark, copyright, or common-law rights nor the similar rights of others. Illumina further reserves the right to make any changes in any processes, products, or parts thereof, described herein without notice. While every effort has been made to make this document as complete and accurate as possible as of the publication date, no warranty of fitness is implied, nor does Illumina accept any liability for damages resulting from the information contained in this document.

ILLUMINA MAKES NO REPRESENTATIONS, WARRANTIES, CONDITIONS, OR COVENANTS, EITHER EXPRESS OR IMPLIED (INCLUDING WITHOUT LIMITATION ANY EXPRESS OR IMPLIED WARRANTIES OR CONDITIONS OF FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, MERCHANTABILITY, DURABILITY, TITLE, OR RELATED TO THE PERFORMANCE OR NONPERFORMANCE OF ANY PRODUCT REFERENCED HEREIN OR PERFORMANCE OF ANY SERVICES REFERENCED HEREIN).

This document may contain references to third-party sources of information, hardware or software, products or services, and/or third-party web sites (collectively the “Third-Party Information”). Illumina does not control and is not responsible for any Third-Party Information, including, without limitation, the content, accuracy, copyright compliance, compatibility, performance, trustworthiness, legality, decency, links, or any other aspect of Third-Party Information. Reference to or inclusion of Third-Party Information in this document does not imply endorsement by Illumina of the Third-Party Information or of the third party in any way.

FOR RESEARCH USE ONLY

© 2009 Illumina, Inc. All rights reserved.

Illumina, illuminaDx, Solexa, Making Sense Out of Life, Oligator, Sentrix, GoldenGate, GoldenGate Indexing, DASL, BeadArray, Array of Arrays, Infinium, BeadXpress, VeraCode, Intellihyb, iSelect, CSPro, GenomeStudio, and Genetic Energy are registered trademarks or trademarks of Illumina, Inc. All other brands and names contained herein are the property of their respective owners.
## Revision History

<table>
<thead>
<tr>
<th>Part Number</th>
<th>Revision Letter</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>15009919</td>
<td>A</td>
<td>November 2009</td>
</tr>
</tbody>
</table>
## Table of Contents

Notice ................................................................. iii
Revision History .................................................... v
Table of Contents ................................................. vii
List of Tables ....................................................... xiii
List of Figures ....................................................... xv

### Chapter 1 Overview ............................................. 1

Introduction ......................................................... 2
  - Analysis of Sequencing Data ................................. 2
  - Analysis Computing Systems ................................. 3
CASAVA v1.6 ......................................................... 5
  - Demultiplexing ................................................. 5
  - Aligning Reads ................................................. 5
  - Variant Detection and Counting ............................ 6
What’s New ........................................................... 7
Technical Assistance .............................................. 8

### Chapter 2 Demultiplexing ........................................ 11

Introduction ......................................................... 12
Demultiplexing Input Files ....................................... 13
  - Basecalls Directory ........................................... 13
  - Sample Sheet .................................................. 14
Demultiplexing Methods ......................................... 15
  - Splitting _qseq Files ......................................... 15
  - Updating Statistics and Reporting ......................... 15
  - Behavior of demultiplex.pl .................................. 15
Running Demultiplexing ......................................... 17
  - Usage of Demultiplexing ..................................... 17
  - Options for Demultiplexing ................................. 17
  - Example Demultiplexing Run ............................... 17
Demultiplexing Output ........................................... 19
  - SamplesDirectories.csv File ............................... 19
GERALD Wrapper for Multiplexed Samples .................... 20
  - Running the GERALD Wrapper ............................. 20
  - Options for GERALD Wrapper .............................. 20
  - Setting up Config.template.txt ............................ 21
### Chapter 3 Using GERALD for Sequence Alignment .......................... 23

- **Introduction** .............................................. 24
- **Configuring GERALD** ...................................... 24
- **GERALD Output** ........................................... 24
- **Alignment Algorithms** ...................................... 24
- **GERALD Input Files** ......................................... 26
  - **Sequence Files** ........................................... 27
  - **Config.txt File** ......................................... 28
  - **Reference Genome** ........................................ 28
  - **BustardSummary.xml File** ................................ 28
  - **Base Calling Config.xml File** ........................... 28
  - **Additional Eland_rna Input Files** ....................... 29
  - **Repeats** .................................................. 29
  - **_qval.txt File** .......................................... 29
  - **GERALD Workflow** ........................................ 29
- **Running GERALD** ........................................... 30
  - **Standard GERALD Analysis** ............................ 30
  - **GERALD Configuration File** ............................ 31
  - **GERALD Parameters** ...................................... 34
  - **Parallelization Switch** .................................. 38
  - **Nohup Command** ......................................... 38
  - **ELANDv2 Alignments** .................................... 38
- **GERALD Output Files** ........................................ 46
  - **Visual Analysis Summary** ................................ 46
  - **Text-Based Analysis Results** ........................... 52
  - **Output File Formats** ..................................... 54
  - **Intermediate Output Data Files** ......................... 56
  - **Interpretation of Run Quality** .......................... 57

### Chapter 4 Variant Detection and Counting ................................. 63

- **Introduction** .............................................. 64
  - **Use Cases** ............................................... 64
  - **CASAVA Workflow** ....................................... 65
  - **Expected Performance** .................................... 66
- **Methods** .................................................. 67
  - **Duplicate Removal** ...................................... 67
  - **Final Set of Reads** ..................................... 67
  - **Allele Calling** ........................................... 67
  - **SNP Calling** ............................................. 67
  - **Indel Finding** ........................................... 67
  - **CASAVA Counting Methods** ............................. 68
- **Estimating Build Depth** ................................... 69
- **Variant Detection Input Files** ................................ 70
  - **Export.txt Files** ........................................ 70
  - **Run.conf.xml** ............................................ 71
  - **Pair.xml** ................................................ 71
  - **Config.xml** .............................................. 71
  - **Summary.htm.** ........................................... 71
# Table of Contents

CASAVA Genome Files .......................................................... 72
Genome Size File ................................................................. 72
Exon Coordinates Set .............................................................. 72
Running Variant Detection and Counting ....................................... 73
Configuring Multiple Runs ....................................................... 77
Examples ................................................................................. 78
Results Directory ....................................................................... 78
Variant Detection and Counting Output Files .................................. 79
Build Directory ......................................................................... 79
Build Web Page ......................................................................... 80
CASAVA Build ........................................................................... 81

## Appendix A

### Requirements and Software Installation for CASAVA

- Introduction ........................................................................ 88
- Hardware and Software Requirements ..................................... 88
  - Disk Space Requirements .................................................. 88
  - Memory Requirements ...................................................... 88
  - Analysis Computer .......................................................... 88
  - Software Requirements .................................................... 89
- Installing CASAVA ................................................................. 90
- Setting Up Email Reporting .................................................... 90

## Appendix B

### Understanding the Run Folder

- Introduction ........................................................................ 94
- Run Folder Structure .......................................................... 97
  - Images Folder .................................................................. 97
  - Data Folder ...................................................................... 98
- Demultiplexing Run Folder .................................................... 100
  - SamplesDirectories.csv File ............................................ 100
- Folder and File Naming ........................................................ 102
  - Run Folder Naming ........................................................ 102
  - File Naming ................................................................... 102
- Configuration/Parameters ....................................................... 104
  - Params File ..................................................................... 104
  - Config.xml Files ......................................................... 104
  - RunInfo.xml File .......................................................... 107
  - Run.conf.xml ................................................................ 107

## Appendix C

### Using Parallelization in GERALD

- “Make” Utilities ................................................................. 110
  - Customizing Parallelization ............................................. 110
  - Parallelization Limitations ............................................... 111
  - Memory Limitations ....................................................... 111

## Appendix D

### Base Call Calibration and Alignment Scoring

- Introduction ........................................................................ 114
- Base Call Calibration .......................................................... 114
  - Goal ............................................................................... 114
  - Method ........................................................................... 114
<table>
<thead>
<tr>
<th>Appendix</th>
<th>Reference Files CASAVA</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>Provided Reference Files</td>
<td>126</td>
</tr>
<tr>
<td></td>
<td>Custom Reference Files</td>
<td>126</td>
</tr>
<tr>
<td></td>
<td>ELAND Reference Files</td>
<td>126</td>
</tr>
<tr>
<td></td>
<td>ELAND Genome Files</td>
<td>127</td>
</tr>
<tr>
<td></td>
<td>Abundant Sequences Files</td>
<td>127</td>
</tr>
<tr>
<td></td>
<td>Splice Junction Set</td>
<td>127</td>
</tr>
<tr>
<td></td>
<td>ELAND Splice Junction Files</td>
<td>127</td>
</tr>
<tr>
<td></td>
<td>CASAVA Reference Files</td>
<td>129</td>
</tr>
<tr>
<td></td>
<td>CASAVA Genome Files</td>
<td>129</td>
</tr>
<tr>
<td></td>
<td>Genome Size File</td>
<td>129</td>
</tr>
<tr>
<td></td>
<td>Exon Coordinates Set</td>
<td>129</td>
</tr>
<tr>
<td></td>
<td>Provided CASAVA Reference Files</td>
<td>130</td>
</tr>
<tr>
<td>F</td>
<td>Generating Reference Files</td>
<td>131</td>
</tr>
<tr>
<td></td>
<td>Getting Data Files</td>
<td>131</td>
</tr>
<tr>
<td></td>
<td>Genome Sequence Files</td>
<td>132</td>
</tr>
<tr>
<td></td>
<td>Abundant Sequence Files</td>
<td>133</td>
</tr>
<tr>
<td></td>
<td>Splice Junction Set</td>
<td>133</td>
</tr>
<tr>
<td></td>
<td>Genome Size Files</td>
<td>135</td>
</tr>
<tr>
<td></td>
<td>Non-Redundant Exon Set</td>
<td>135</td>
</tr>
<tr>
<td></td>
<td>Squashing the Reference Files</td>
<td>136</td>
</tr>
</tbody>
</table>

Appendix E Reference Files CASAVA .............................................................. 125

Appendix F ELANDv2 Algorithm ................................................................. 139

Introduction ................................................................................................. 140
Appendix G  Algorithms in Variant Detection and Counting  . . . . . . . . 145

Introduction ........................................................................................................ 146
SNP Calling Algorithms .................................................................................. 146
  Allele Calling ................................................................................................. 146
  SNP Calling .................................................................................................. 147
Indel Finder Algorithm. .................................................................................... 147
  IndelFinder ...................................................................................................... 147
  AlignCandidates .......................................................................................... 148
  ClusterFinder ............................................................................................... 149
  SmallAssembler ........................................................................................... 149
  AlignContig .................................................................................................... 150
Indel Genotype Caller .......................................................................................... 153
RNA Sequencing Counting Methods ................................................................. 155
  readBases Counting Method ...................................................................... 155
  readStart Counting Method ........................................................................ 155
Advantages of Read Base Counting Method .................................................... 155
Reference ........................................................................................................... 156
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>Illumina Customer Support Contacts</td>
<td>8</td>
</tr>
<tr>
<td>Table 2</td>
<td>GERALD Configuration File Parameters</td>
<td>31</td>
</tr>
<tr>
<td>Table 3</td>
<td>GERALD Configuration File Lane-Specific Options</td>
<td>32</td>
</tr>
<tr>
<td>Table 4</td>
<td>GERALD Configuration File Optional Parameters</td>
<td>32</td>
</tr>
<tr>
<td>Table 5</td>
<td>GERALD Configuration File Paired-End Analysis Options</td>
<td>33</td>
</tr>
<tr>
<td>Table 6</td>
<td>ANALYSIS Variables</td>
<td>34</td>
</tr>
<tr>
<td>Table 7</td>
<td>Lane-by-Lane Parameters</td>
<td>35</td>
</tr>
<tr>
<td>Table 8</td>
<td>USE_BASES Options</td>
<td>36</td>
</tr>
<tr>
<td>Table 9</td>
<td>QCAL_SOURCE Variable Values</td>
<td>36</td>
</tr>
<tr>
<td>Table 10</td>
<td>Parameters for ANALYSIS eland_extended</td>
<td>40</td>
</tr>
<tr>
<td>Table 11</td>
<td>Parameters for ANALYSIS eland_pair</td>
<td>41</td>
</tr>
<tr>
<td>Table 12</td>
<td>Parameters for ANALYSIS eland_rna</td>
<td>44</td>
</tr>
<tr>
<td>Table 13</td>
<td>Example of Relative Orientation Statistics Table</td>
<td>50</td>
</tr>
<tr>
<td>Table 14</td>
<td>Example of Insert Size Statistics Table</td>
<td>50</td>
</tr>
<tr>
<td>Table 15</td>
<td>Example of Insert Statistics Table</td>
<td>50</td>
</tr>
<tr>
<td>Table 16</td>
<td>Text-Based Analysis Results</td>
<td>53</td>
</tr>
<tr>
<td>Table 17</td>
<td>Final Output File Formats</td>
<td>55</td>
</tr>
<tr>
<td>Table 18</td>
<td>Intermediate Output File Descriptions</td>
<td>56</td>
</tr>
<tr>
<td>Table 19</td>
<td>Intermediate Output File Formats</td>
<td>57</td>
</tr>
<tr>
<td>Table 20</td>
<td>Example of Lane Results Summary</td>
<td>57</td>
</tr>
<tr>
<td>Table 21</td>
<td>Example of Expanded Lane Summary</td>
<td>58</td>
</tr>
<tr>
<td>Table 22</td>
<td>Expected Performance for Typical CASAVA Projects</td>
<td>66</td>
</tr>
<tr>
<td>Table 23</td>
<td>Major File Options for Variant Detection and Counting</td>
<td>73</td>
</tr>
<tr>
<td>Table 24</td>
<td>Behavioral Options for Variant Detection and Counting</td>
<td>74</td>
</tr>
<tr>
<td>Table 25</td>
<td>Analysis Options for Variant Detection and Counting</td>
<td>75</td>
</tr>
<tr>
<td>Table 26</td>
<td>Targets for Variant Detection and Counting</td>
<td>76</td>
</tr>
<tr>
<td>Table 27</td>
<td>File Naming Components</td>
<td>103</td>
</tr>
<tr>
<td>Table 28</td>
<td>Provided Reference Files for ELAND</td>
<td>128</td>
</tr>
<tr>
<td>Table 29</td>
<td>Provided Reference Files for CASAVA for RNA Sequencing</td>
<td>130</td>
</tr>
</tbody>
</table>
List of Figures

Figure 1  Sequencing Data Analysis Workflow ........................................... 2
Figure 2  Applications in Data Analysis Workflow ...................................... 4
Figure 3  Splitting Basecalls Directory by Demultiplexing .......................... 12
Figure 4  Sample Sheet Opened in Excel .................................................... 14
Figure 5  demultiplex.pl Behavior with 3 Lanes and 3 Indexes ...................... 16
Figure 6  demultiplex.pl Behavior with 12 Different Indexes in 3 Lanes ........ 16
Figure 7  demultiplex.pl Behavior with 12 Indexes, 3, 4, and 5 per Lane ....... 16
Figure 8  SamplesDirectories.csv File Opened in Excel ............................. 19
Figure 9  Locations of GERALD Input and Output Files ............................ 26
Figure 10 CASAVA Workflow ................................................................. 65
Figure 11 CASAVA Input Files ................................................................... 70
Figure 12 CASAVA Build Directory ........................................................... 79
Figure 13 Build Web Page ......................................................................... 80
Figure 14 Summary.htm File ...................................................................... 80
Figure 15 SNPs Graphs in Home.html ..................................................... 81
Figure 16 Statistics Graphs in Home.html .................................................. 81
Figure 17 Chromosome.snp.txt File Opened in Excel ................................ 84
Figure 18 Chromosome_genes_count.txt File Opened in Excel .................. 86
Figure 19 SCS Real Time Analysis Run Folder Directory Structure ............. 94
Figure 20 OLB/CASAVA Run Folder Directory Structure ............................ 95
Figure 21 SamplesDirectories.csv File Opened in Excel ............................ 100
Figure 22 Demultiplexing Run Folder Changes ........................................ 101
Figure 23 UCSC Genome Bioinformatics Web Page ................................ 131
Figure 24 Selected Genome Web Page (Human) ........................................ 131
Figure 25 Annotation Database Files Web Page, Files .............................. 132
Figure 26 Index (Data set by chromosome) Web Page .............................. 132
Figure 27 Ungapped Versus Gapped Alignment ........................................ 141
Figure 28 Singleseed Versus Multiseed Alignment .................................... 142
Chapter 1
Overview

Topics

2 Introduction
2 Analysis of Sequencing Data
3 Analysis Computing Systems
5 CASAVA v1.6
5 Demultiplexing
5 Aligning Reads
6 Variant Detection and Counting
7 What’s New
8 Technical Assistance
Introduction

This user guide documents CASAVA v1.6 (short for "Consensus Assessment of Sequence And VAriation"). CASAVA v1.6 performs alignment of a sequencing run to a reference genome and subsequent variant analysis and read counting. The basic pieces of functionality are described below.

Analysis of Sequencing Data

After the Genome Analyzer generates the sequencing images, the data are analyzed in four steps: image analysis, base calling, sequence alignment, and variant analysis and counting (Figure 1). For multiplexed samples, an additional demultiplexing step is performed before the alignment step.
1. **Image analysis**—Uses the raw TIF files to locate clusters on the image, and outputs the cluster intensity, X,Y positions, and an estimate of the noise for each cluster. The output from image analysis provides the input for base calling. Image analysis is preformed by Sequencing Control Software real time analysis (RTA).

2. **Base calling**—Uses cluster intensities and noise estimates to output the sequence of bases read from each cluster, a confidence level for each base, and whether the read passes filtering. Base calling is preformed by RTA or the Off-Line Basecaller (OLB).

3. **Sample demultiplexing**—Uses the index read to separate multiplexed samples. The indexes are defined in a samplesheet.

4. **Sequence alignment**—Aligns samples to a reference sequence. Analysis for multiplexed samples can be set up in one step using the multiplexGERALD.pl script.

5. **Variant analysis and counting**—Calls Single Nucleotide Polymorphisms (SNPs) and indels, and performs read counting (for RNA sequencing).

After variant analysis and counting are finished, the results can be viewed and analyzed further in the GenomeStudio software, or the result files can be analyzed using third-party software.

### Analysis Computing Systems

Several analysis software products can be used for the analysis cascade. The default workflow uses these software products (Figure 2):

- Sequencing Control Software v2.6 (SCS) real time analysis (RTA), which runs on the Genome Analyzer instrument computer. RTA performs real-time image analysis and base calling.
- CASAVA v1.6 (CASAVA), which runs on a Linux analysis server. CASAVA performs demultiplexing, off-line sequence alignment, calls SNPs and indels, and performs read counting (for RNA sequencing).

Some alternative software products can be used for data analysis:

- Off-Line Base caller (OLB), which runs on a Linux analysis server. OLB performs off-line image analysis and base calling.
- Pipeline Analysis Software (Pipeline) v1.5, which runs on a Linux analysis server. Pipeline performs off-line image analysis, base calling, and sequence alignment.

### NOTE

The Pipeline alignment module GERALD is now part of CASAVA v1.6.
Figure 2 Applications in Data Analysis Workflow
CASAVA v1.6

The CASAVA v1.6 package processes sequencing reads provided by RTA or Pipeline. CASAVA provides the following output:
- Separated reads from multiplexed samples
- Aligned reads
- SNP calls
- Indel calls
- Expression levels for exons, genes and splice junctions in the RNA Sequencing analysis

In addition, CASAVA automatically generates a range of statistics, such as mean depth and percentage chromosome coverage, to enable comparison with previous builds or other individuals.

CASAVA analyzes sequencing reads in three steps:
- Demultiplexing (for multiplexed sequencing only)
- Alignment to a reference genome
- Variant detection and counting

These three steps are explained below.

Demultiplexing
Multiplexed sequencing allows you to run up to 12 individual samples in one lane, for a total of 96 samples. The samples are identified by an index sequence (barcode) that was attached to the template during sample preparation. Multiplexed sequencing is described in the Multiplexed Sequencing on the Genome Analyzer booklet.

Multiplexed sequencing runs from SCS v2.4 and later versions set the index read as a separate read. Sample demultiplexing in CASAVA creates several subdirectories to dispatch the data associated with the different barcodes. Each subdirectory has a structure similar to the original BaseCalls directory, but where each lane contains only the data associated to a single barcode.

Use and properties of demultiplexing are explained in Chapter 2, Demultiplexing.

Aligning Reads
CASAVA performs sequence alignment using the GERALD module, which is a set of utilities supplied as source code and scripts.

The output data produced by GERALD are stored in a hierarchical folder structure called the Run Folder. The Run Folder includes all data folders generated from the Genome Analyzer and the data analysis software.

Alignment Algorithms
GERALD provides two alignment algorithms: PhageAlign and ELAND.
- PhageAlign performs an exhaustive alignment and always finds the best match but is very slow.
- Efficient Large-Scale Alignment of Nucleotide Databases (ELAND) is very fast and should be used to match a large number of reads against the reference genome.
ELAND is much faster but less sensitive than PhageAlign, which will always find a best match for your reads. Consider the following points when using ELAND:

- If your reads are noisy, not all of them are going to align, and you may not get good results.
- Error rates based on ELAND output underestimate the true error rate.
- Some analysis modes eland_extended, eland_pair, and eland_rna) do not tolerate more than two mismatches in the first 32 bases.

As of CASAVA v1.6 a new version of ELAND is available, ELANDv2. The most important improvements of ELANDv2 are its ability to perform multiseed and gapped alignments. As a consequence, ELANDv2 handles indels and mismatches better.

**Multiseed and Gapped Alignment**

ELANDv2 performs multiseed alignment by aligning the first set of 16 to 32 bases and consecutive sets of bases separately. After this, ELANDv2 extends each candidate alignment to the full length of the read, using a gapped alignment method that allows for gaps (indels) of up to 20 bases. ELANDv2 then picks the best alignment based on alignment scores.

More information about ELANDv2 is available in Appendix F, *ELANDv2 Algorithm*. Use and properties of GERALD are explained in Chapter 3, *Using GERALD for Sequence Alignment*.

**Variant Detection and Counting**

During variant detection and counting, CASAVA generates a CASAVA build, which is a post-sequencing analysis of data from reads aligned to a reference genome by GERALD.

During the build process, the CASAVA collates, filters and compiles aligned reads. CASAVA then calls the genomic consensus sequence using a Bayesian algorithm and compares it to the reference sequence in order to identify homozygous or heterozygous SNPs. CASAVA also identifies consensus indels using the Indel Finder. Moreover, CASAVA provides expression levels for exons, genes and splice junctions in the RNA Sequencing analysis.

Use and properties of CASAVA are explained in Chapter 4, *Variant Detection and Counting*. More information about the algorithm for finding indels is available in *Indel Finder Algorithm* on page 147.
What’s New

Important Changes in CASAVA v1.6

- Performs demultiplexing
- Includes the GERALD alignment module
- Contains ELANDv2, which enables multiseed and gapped alignment
- Performs indel detection
- GERALD will use the default ANALYSIS: none if no analysis is provided for a lane or indexed lane.
- As of RTA v1.6, OLB v1.6, and CASAVA v1.6, the X and Y coordinates for each clusters in the qseq files are calculated in a way that makes sure the combination will be unique. The new coordinates are the old coordinates times 10, +1000, and then rounded.

For more information, see the Release Notes for CASAVA v1.6, or the Changes file in (CASAVAInstallationDirectory)/share/CASAVA-1.6.0.
Technical Assistance

For technical assistance, contact Illumina Customer Support.

Table 1  Illumina Customer Support Contacts

<table>
<thead>
<tr>
<th>Contact</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toll-free Customer Hotline</td>
<td>1-800-809-ILMN (1-800-809-4566)</td>
</tr>
<tr>
<td>International Customer Hotline</td>
<td>1-858-202-ILMN (1-858-202-4566)</td>
</tr>
<tr>
<td>Illumina Website</td>
<td><a href="http://www.illumina.com">http://www.illumina.com</a></td>
</tr>
<tr>
<td>Email</td>
<td><a href="mailto:techsupport@illumina.com">techsupport@illumina.com</a></td>
</tr>
</tbody>
</table>

MSDSs

Material safety data sheets (MSDSs) are available on the Illumina website at http://www.illumina.com/msds.

Product Documentation

If you require additional product documentation, you can obtain PDFs from the Illumina website. Go to http://www.illumina.com/documentation. When you click on a link, you will be asked to log in to iCom. After you log in, you can view or save the PDF.

If you do not already have an iCom account, then click New User on the iCom login screen and fill in your contact information. Indicate whether you wish to receive the iCommunity newsletter (a quarterly newsletter with articles about, by, and for the Illumina Community), illumINOTES (a monthly newsletter that provides important product updates), and announcements about upcoming user meetings. After you submit your registration information, an Illumina representative will create your account and email login instructions to you.

Frequently Asked Questions

Frequently asked questions are available online.

Go to http://www.illumina.com/FAQs, and click on Software.

Reporting Problems

When reporting an issue, it is critical to capture all the output and error messages produced by a run. This is done by redirecting the output using “nohup” or the facilities of a cluster management system. For an explanation of “nohup,” see Nohup Command on page 38.

Provide the following information, if available:

- Description of the error / bug / feature
- Gerald / Elandv2 Config file
- Validation report (validate.sh.out)
- Summary.htm
- List of Alignment folder contents
- Nohup
- Contents of conf folder (at the very least project.conf and run.conf.xml files)
- All files that begin with “tasks” (.txt, .current, .err.txt, and .out.txt normally)
- CASAVA.log file
- The command-line used by the user
- The makefile corresponding to the part of CASAVA that is causing the problem.
Chapter 2
Demultiplexing

Topics
12 Introduction
13 Demultiplexing Input Files
   13 Basecalls Directory
   14 Sample Sheet
15 Demultiplexing Methods
   15 Splitting _qseq Files
   15 Updating Statistics and Reporting
17 Running Demultiplexing
   17 Usage of Demultiplexing
   17 Options for Demultiplexing
   17 Example Demultiplexing Run
19 Demultiplexing Output
   19 SamplesDirectories.csv File
20 GERALD Wrapper for Multiplexed Samples
   20 Running the GERALD Wrapper
   20 Options for GERALD Wrapper
   21 Setting up Config.template.txt
   21 Example Config.template.txt
Introduction

Multiplexed sequencing allows you to run multiple individual samples in one lane. The samples are identified by an index sequence that was attached to the template during sample prep. Multiplexed sequencing is described in the *Multiplexed Sequencing on the Genome Analyzer* booklet.

Multiplexed sequencing runs from SCS v2.4 and later versions set the index read as a separate read. The sample demultiplexer in CASAVA creates a directory for each sample based on the index sequence. The directory contains the files necessary for alignment, variant analysis, and counting with CASAVA.

In CASAVA, sample demultiplexing is done before alignment. It uses a sample sheet transmitted by RTA or manually copied to split the qseq files according to the index of the samples (Figure 3).

Figure 3  Splitting Basecalls Directory by Demultiplexing
Demultiplexing needs a Basecalls directory and a sample sheet to start a run. These files are described below; see Figure 3 for their location in the run folder. For an explanation of the run folder, see Appendix B, Understanding the Run Folder.

For installation instructions, see Appendix A, Requirements and Software Installation for CASAVA).

**Basecalls Directory**

Demultiplexing requires a Basecalls directory as generated by RTA or OLB, which contains the sequence files (_qseq files). These files have the following format:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine name</td>
<td>Identifier of the sequencer.</td>
</tr>
<tr>
<td>Run number</td>
<td>Number to identify the run on the sequencer.</td>
</tr>
<tr>
<td>Lane number</td>
<td>Positive integer (currently 1-8).</td>
</tr>
<tr>
<td>Tile number</td>
<td>Positive integer.</td>
</tr>
<tr>
<td>X</td>
<td>X coordinate of the spot. Integer. As of RTA v1.6, OLB v1.6, and CASAVA v1.6, the X and Y coordinates for each clusters are calculated in a way that makes sure the combination will be unique. The new coordinates are the old coordinates times 10, +1000, and then rounded.</td>
</tr>
<tr>
<td>Y</td>
<td>Y coordinate of the spot. Integer. As of RTA v1.6, OLB v1.6, and CASAVA v1.6, the X and Y coordinates for each clusters are calculated in a way that makes sure the combination will be unique. The new coordinates are the old coordinates times 10, +1000, and then rounded.</td>
</tr>
<tr>
<td>Index</td>
<td>Index sequence or 0. For no indexing, or for a file that has not been demultiplexed yet, this field should have a value of 0.</td>
</tr>
<tr>
<td>Read Number</td>
<td>1 for single reads; 1 or 2 for paired ends or multiplexed single reads; 1, 2, or 3 for multiplexed paired ends.</td>
</tr>
<tr>
<td>Sequence</td>
<td>Called sequence of read.</td>
</tr>
<tr>
<td>Quality</td>
<td>The calibrated quality string.</td>
</tr>
<tr>
<td>Filter:</td>
<td>Did the read pass filtering? 0 - No, 1 - Yes.</td>
</tr>
</tbody>
</table>

For demultiplexing, these _qseq files should have the following structure:

- The first read is the single read sequence, or first end read sequence (for paired end sequencing).
- The index read is the second read.
For paired end runs, the paired (second) end is the third read.
The Index column (column 7) is either empty or "0".

Sample Sheet

Demultiplexing needs a sample sheet (SampleSheet.csv file) describing the samples in each lane, including the indexes used. The sample sheet is located in the Basecalls directory of the run folder. You can open and edit the sample sheet in Excel, which would like Figure 4.

The sample sheet contains the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCID</td>
<td>Flow cell ID</td>
</tr>
<tr>
<td>Lane</td>
<td>Positive integer, indicating the lane number (1-8)</td>
</tr>
<tr>
<td>SampleID</td>
<td>ID of the sample</td>
</tr>
<tr>
<td>SampleRef</td>
<td>The reference sequence for the sample</td>
</tr>
<tr>
<td>Index</td>
<td>Index sequence or 0 for no indexing</td>
</tr>
<tr>
<td>Description</td>
<td>Description of the sample</td>
</tr>
<tr>
<td>Control</td>
<td>Y indicates this lane is a control lane, N means sample</td>
</tr>
<tr>
<td>Recipe</td>
<td>Recipe used during sequencing</td>
</tr>
<tr>
<td>Operator</td>
<td>Name or ID of the operator</td>
</tr>
</tbody>
</table>

If no sample sheet has been generated yet, you can generate it using Excel or other text editing tool that allows .csv files to be saved. Enter the columns specified above for each sample, and save the Excel file in the .csv format.
Demultiplexing Methods

Demultiplexing involves splitting the _qseq files and updating statistics and reporting. This section describes these two steps.

Splitting _qseq Files

The first step of demultiplexing in CASAVA is splitting the _qseq files, based on the index sequence. This is done the following way for each cluster:

1. Get the raw index from the _qseq file.
2. Identify the appropriate directory for the index.
3. **Optional:** Detect and correct one error on the barcode.
4. For each end:
   a. Set the index field to the raw index.
   b. Append the end to the appropriate new _qseq file in the selected directory.
5. If the index cannot be identified, the data is written into the unknown sub-directory.

Updating Statistics and Reporting

The sample demultiplexer updates the following files for each index directory:

- Generates statistics
  - While splitting the _qseq files, CASAVA recalculates the base calling analysis statistics that were computed during base calling for the unsplit sample.
- Regenerates plots the analysis plots for each multiplexed sample
- Regenerates the BustardSummary.xml for each multiplexed sample
- Updates config.xml for each multiplexed sample
- Copies raw matrix and phasing files
- Updates sample sheet
  - The sample demultiplexer strips all the non-relevant indexes from the original sample sheet and places the stripped out version in the appropriate directory.
- Creates a file SampleDirectories.csv in the Demultiplexed folder to indicate in which subdirectory each barcode has been written.

Behavior of demultiplex.pl

The configuration of the Demultiplexed folder and the contents of the bin folders are dependent on the index to lane assignments in the sample sheet. The bullets below provide a brief description of how this information is organized in the Demultiplexed folder:

- If each lane contains a single different index, only one bin folder will be created (i.e., 001) and it will contain _qseq files for each lane and each read. This is because each lane is still a separate sample.
- If each lane contains two indexes, two bin folders will be created (i.e., 001 and 002) and each bin folder will contain _qseq files for each lane that are each associated with a specific index.
The max number of directories that are generated is always equal to the maximum number of indexes you have for any particular lane.

The behavior is illustrated in Figure 5, Figure 6, and Figure 7 below.

- Figure 5: demultiplex.pl Behavior with 3 Lanes and 3 Indexes
- Figure 6: demultiplex.pl Behavior with 12 Different Indexes in 3 Lanes
- Figure 7: demultiplex.pl Behavior with 12 Indexes, 3, 4, and 5 per Lane

Note that the mapping between bin folders and indexes per lane is arbitrary, so it is not guaranteed that the same index will end up in the same directory (even if they are given in the same order) as shown in Figure 5.
Running Demultiplexing

This section describes how to perform demultiplexing in CASAVA v1.6.

Usage of Demultiplexing

To perform demultiplexing, enter the following command line:

```
/path-to-CASAVA/bin/demultiplex.pl [options]
```

Options for Demultiplexing

The options for demultiplexing are described below.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Examples/Defaults</th>
</tr>
</thead>
<tbody>
<tr>
<td>--qseq-mask</td>
<td>The <code>qseq-mask</code> string specifies how to use each cycle.</td>
<td>--qseq-mask Y50nI6ny50n</td>
</tr>
<tr>
<td></td>
<td>• A lower-case “n” means ignore the cycle.</td>
<td>If no --qseq-mask is specified, the default will be inferred from</td>
</tr>
<tr>
<td></td>
<td>• An upper-case “Y” means use the cycle (for paired-end, this designates the first end).</td>
<td>the config.xml file in the base-calls directory</td>
</tr>
<tr>
<td></td>
<td>• A lower-case “y” means use the cycle for the second end of paired-end sequencing.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• An upper-case “I” means use the cycle for the index read.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• A number means that the previous character is repeated that many times.</td>
<td></td>
</tr>
<tr>
<td>-s, --sample-sheet</td>
<td>Path to sample sheet</td>
<td>Defaults to &lt;input_dir&gt;/SampleSheet.csv</td>
</tr>
<tr>
<td>-i, --input-dir</td>
<td>Path to a Basecalls directory</td>
<td>Defaults to current dir --input-dir &lt;Basecalls_dir&gt;</td>
</tr>
<tr>
<td>-o, --output-dir</td>
<td>Path to demultiplexed output</td>
<td>Defaults to &lt;input_dir&gt;/Demultiplexed</td>
</tr>
<tr>
<td>--man</td>
<td>Print a manual page for this command</td>
<td>/path-to-CASAVA/bin/demultiplex.pl --man</td>
</tr>
<tr>
<td>-h, --help</td>
<td>produce help message and exit</td>
<td>/path-to-CASAVA/bin/demultiplex.pl --h</td>
</tr>
<tr>
<td>-m, --mismatches</td>
<td>Number of mismatches allowed in the indexes (0 or 1).</td>
<td>--mismatches=1</td>
</tr>
<tr>
<td></td>
<td>Default is 0.</td>
<td></td>
</tr>
<tr>
<td>--correct-errors</td>
<td>If this option exists in the command line and --mismatches=1, mismatches found in the indexes will be corrected in the output.</td>
<td>--correct-errors</td>
</tr>
</tbody>
</table>

Example Demultiplexing Run

An example of a demultiplexing run is as follows:

```
/path-to-CASAVA/bin/demultiplex.pl --input-dir <Basecalls_dir> --output-dir <my_demultiplexed>
--sample-sheet <input_dir>/SampleSheet.csv --qseq-mask Y51l6ny51
```
This will produce a set of directories in the Basecalls directory, where all of the subdirectories are valid Basecalls directories (they have all the expected statistics and plots):

```
my_demultiplexed/001/...
  002/...
  003/...
  ...
unknown/...
```

The option `--qseq-mask Y51I6ny51` specifies the following:
- Use the first 51 bases for the first end (Y51I6ny51)
- Use the next 6 bases for the index read (Y51I6ny51)
- Do not use the next base (Y51I6ny51)
- Use the next 51 bases for the second end (Y51I6ny51)

Reads with an unresolved or erroneous index are placed in the unknown directory.
Demultiplexing Output

The demultiplexing output directories have the following characteristics:

- As many base calls directories as the maximum number of indices in each lane.
- The directory names are simply a three digit number (left padded with 0) starting from 1.
- The association of indices to specific directories is arbitrary.
- The demultiplexed output directories for each sample are specified in the SamplesDirectories.csv file in the Demultiplexed directory.
- The Unknown directory contains the reads with an unresolved or erroneous index.
- Each base calls directory is a valid base calls directory that can be used for subsequent alignment analysis in CASAVA.
- The index inserted in column 7 of the qseq files is by default the raw index.

**NOTE**

If the majority of reads end up in the 'unknown' folder, check the --qseq-mask parameter syntax and the length of the index in the sample sheet. The --qseq-mask option should be set to the length of the index in the sample sheet + the character ‘n’ to account for phasing.

**SamplesDirectories.csv File**

The SamplesDirectories.csv contains the sample information from the sample sheet, and has an additional column (Directory) that specifies the numbered demultiplexed output directory for each sample (Figure 8). The SamplesDirectories.csv file is located in the Demultiplexed directory.

When opening the file in Excel, make sure to use Excel’s Import Wizard (Data > Get External Data > From Text). Tell Excel to treat the last column (Directory) as "text" and leave the rest as "general". This will make sure the Directory numbers are displayed correctly (starting with 00).

<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>FCID</td>
<td>Lane</td>
<td>SampleID</td>
<td>SampleRef</td>
<td>Index</td>
<td>Description</td>
<td>Control</td>
<td>Recipe</td>
<td>Operator</td>
</tr>
<tr>
<td>2</td>
<td>F0612FV</td>
<td>1</td>
<td>sample1</td>
<td>hglf/ehhNis ATACG</td>
<td>desc1</td>
<td>N</td>
<td>R1</td>
<td>j</td>
<td>dec</td>
</tr>
<tr>
<td>3</td>
<td>F0612FV</td>
<td>1</td>
<td>sample2</td>
<td>E.coli TAGTTT</td>
<td>desc1</td>
<td>N</td>
<td>R1</td>
<td>j</td>
<td>dec</td>
</tr>
<tr>
<td>4</td>
<td>F0612FV</td>
<td>2</td>
<td>sample3</td>
<td>hglf/ehhNis ATACG</td>
<td>desc1</td>
<td>N</td>
<td>R1</td>
<td>j</td>
<td>dec</td>
</tr>
<tr>
<td>5</td>
<td>F0612FV</td>
<td>2</td>
<td>sample4</td>
<td>OMV CAGATG</td>
<td>desc1</td>
<td>N</td>
<td>R1</td>
<td>j</td>
<td>dec</td>
</tr>
<tr>
<td>6</td>
<td>F0612FV</td>
<td>2</td>
<td>sample5</td>
<td>E.coli CTTGTA</td>
<td>desc1</td>
<td>N</td>
<td>R1</td>
<td>j</td>
<td>dec</td>
</tr>
<tr>
<td>7</td>
<td>F0612FV</td>
<td>3</td>
<td>sample6</td>
<td>hglf/ehhNis ATACG</td>
<td>desc1</td>
<td>N</td>
<td>R1</td>
<td>j</td>
<td>dec</td>
</tr>
<tr>
<td>8</td>
<td>F0612FV</td>
<td>3</td>
<td>sample7</td>
<td>OMV CAGATG</td>
<td>desc1</td>
<td>N</td>
<td>R1</td>
<td>j</td>
<td>dec</td>
</tr>
<tr>
<td>9</td>
<td>F0612FV</td>
<td>3</td>
<td>sample8</td>
<td>E.coli CTTGTA</td>
<td>desc1</td>
<td>N</td>
<td>R1</td>
<td>j</td>
<td>dec</td>
</tr>
</tbody>
</table>

*Figure 8  SamplesDirectories.csv FileOpened in Excel*
GERALD Wrapper for Multiplexed Samples

CASAVA v1.6 has a wrapper that allows you to initiate GERALD analysis for several multiplexed samples at the same time. You only have to generate a single config.template.txt file. The multiplexedGERALD.pl script uses this template and the sample sheet to generate individual config.txt files for each GERALD analysis. Then the multiplexedGERALD.pl script starts the GERALD analysis for each sample you specified.

Running the GERALD Wrapper

The standard way to run GERALD analysis for several multiplexed samples is to set the parameters in a configuration file to create a makefile.

1. Edit the config.txt file as described below.

2. Enter the following command to create a makefile for multiplexed sequence analysis:

   ```
   /path-to-CASAVA/bin/multiplexedGERALD.pl config.template.txt --EXPT_DIR path_to_Demultiplexed_folder
   ```

3. Move into the newly created GERALD folder under the "path_to_Demultiplexed_folder". Type the "make" command for basic analysis:

   ```
   make
   ```

   You may prefer to use the parallelization option as follows:

   ```
   make -j 3 all
   ```

   The extent of the parallelization depends on the setup of your computer or computing cluster. For a description of parallelization, see Using Parallelization in GERALD on page 109.

4. After the analysis is done, review the analysis for each sample:

   a. View the analysis results of your run. See Visual Analysis Summary on page 44 and Text-Based Analysis Results on page 51.

   b. Interpret the run quality. See Interpretation of Run Quality on page 53.

Options for GERALD Wrapper

The GERALD wrapper accepts the regular GERALD options (see GERALD Parameters on page 34), but most of these will be specified in the config.template file. The options listed below are generally used in the command line.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--RUN</td>
<td>Run the makefiles immediately after analysis</td>
</tr>
<tr>
<td>--FORCE</td>
<td>Keep the analysis directories</td>
</tr>
<tr>
<td>--compression</td>
<td>Type of compression to use (gzip</td>
</tr>
</tbody>
</table>
Setting up Config.template.txt

The config.template.txt file has exactly the same syntax as the regular config.txt files used by GERALD.pl, except for two new keywords introduced in the template: SAMPLE and REFERENCE. These keywords refer to the sampleID and SampleRef specified in the SampleSheet.csv file located in the Basecalls directory of the run folder.

Lines starting with SAMPLE or REFERENCE override any default settings specified in the config.template.txt file, but only for those samples for which the SampleID or SampleRef matches the SAMPLE or REFERENCE.

For example, if the config.template.txt file describes the following analysis:

```
ANALYSIS eland_rna
REFERENCE human ANALYSIS eland_pair
```

with the following sample sheet:

```
FCID,Lane,SampleID,SampleRef,Index,Description,Control
12345AAXX,1,sample1,human,AAGTGTC,desc1,N,R1,name
12345AAXX,1,sample2,human,GGTCATA,desc2,N,R1,name
12345AAXX,2,sample3,rat,CCTGAAA,desc3,N,R1,name
12345AAXX,2,sample4,mouse,GGGAATT,desc4,N,R1,name
```

then the wrapper will initiate an eland_pair analysis for all human samples (sample1 and sample2), and use the default analysis eland_rna for all other samples (sample3 and sample4). This allows you to set the analysis, reference genome, and all other ELAND parameters reference by reference, or sample by sample.

For an overview of the GERALD parameters you can specify, see GERALD Parameters on page 34.

Example Config.template.txt

If you have a flow cell with samples needing different references, with the following sample sheet:

```
FCID,Lane,SampleID,SampleRef,Index,Description,Control
12345AAXX,1,sample1,human,AAGTGTC,desc1,N,R1,name
12345AAXX,1,sample2,human,GGTCATA,desc2,N,R1,name
12345AAXX,2,sample3,rat,CCTGAAA,desc3,N,R1,name
12345AAXX,2,sample4,mouse,GGGAATT,desc4,N,R1,name
```

and the following config.template.txt:

```
ANALYSIS eland_rna
REFERENCE human ELAND_GENOME
/path/to/human/genome/files/
REFERENCE human ELAND_RNA_GENOME_SPLICE
/path/to/human/splice/files/
REFERENCE human ELAND_RNA_GENOME_CONTAM
/path/to/human/contam/files/
REFERENCE mouse ELAND_GENOME
```
This will run GERALD with the human configuration for lanes 1 and 2 in directory 001. Then GERALD, with human configuration in lane 1 and mouse in lane 2, will be run in directory 002.

This will create the following config.txt in 001:

```
ANALYSIS eland_rna
12:ELAND_GENOME /path/to/human/genome/files/
12:ELAND_RNA_GENOME_SPLICE /path/to/human/splice/files/
12:ELAND_RNA_GENOME_CONTAM /path/to/human/contam/files/
```

And the following config.txt in 002:

```
ANALYSIS eland_rna
1:ELAND_GENOME /path/to/human/genome/files/
1:ELAND_RNA_GENOME_SPLICE /path/to/human/splice/files/
1:ELAND_RNA_GENOME_CONTAM /path/to/human/contam/files/
2:ELAND_GENOME /path/to/mouse/genome/files/
2:ELAND_RNA_GENOME_SPLICE /path/to/mouse/splice/files/
2:ELAND_RNA_GENOME_CONTAM /path/to/mouse/contam/files/
```
Chapter 3

Using GERALD for Sequence Alignment

Topics

24 Introduction
   24   Configuring GERALD
   24   GERALD Output
   24   Alignment Algorithms

26 GERALD Input Files
   27   Sequence Files
   28   Config.txt File
   28   Reference Genome
   29   Additional Eland_rna Input Files
   29   Repeats

30 Running GERALD
   30   Standard GERALD Analysis
   31   GERALD Configuration File
   34   GERALD Parameters
   38   ELANDv2 Alignments

46 GERALD Output Files
   46   Visual Analysis Summary
   52   Text-Based Analysis Results
   54   Output File Formats
   56   Intermediate Output Data Files
   57   Interpretation of Run Quality
CHAPTER 3
Using GERALD for Sequence Alignment

Introduction

GERALD is a CASAVA module that performs sequence alignments. This chapter describes running GERALD, parameters, analysis variables, configuration file options, and ELANDv2 alignments.

**NOTE**
The alignment module GERALD used to be part of the Pipeline Analysis Software, but now is part of CASAVA v1.6.

**NOTE**
For installation instructions, see Appendix A, *Requirements and Software Installation for CASAVA*.

Configuring GERALD

GERALD uses multiple analysis parameters. Therefore, it is recommended to include the parameters in a configuration file and provide that file as input to GERALD.

You can define GERALD analysis parameters in the configuration file or in the command line. Command line arguments take precedence over parameters set in the configuration file. For a full description of analysis parameters and variables, see *GERALD Parameters* on page 34.

GERALD Output

GERALD output is a flat text file containing each read and information about its alignment to the reference. In addition, GERALD produces statistics and diagnostic plots that can be used to assess data quality. These are presented in the form of html pages found in the GERALD output folder.

As a result of running the GERALD.pl script, a new directory is created in the run folder (for more information, see Appendix B, *Understanding the Run Folder*). This directory is named using the format GERALD_DD-MM-YYYY_user, where the date is the current date and user is your computer login. If you want to rerun the analysis and change parameters, you can rerun GERALD with new parameters. A new directory will be created and no information will be overwritten.

Alignment Algorithms

GERALD provides two alignment algorithms: PhageAlign and ELANDv2.

- PhageAlign performs an exhaustive alignment and always finds the best match but is very slow.
- Efficient Large-Scale Alignment of Nucleotide Databases version 2 (ELANDv2) is very fast and should be used to match a large number of reads against the human genome.

ELANDv2 is less sensitive than PhageAlign, which will always find a best match for your reads, although possibly not a unique one. Consider the following points when using ELANDv2:

- If your data is noisy, not all of it is going to align, and you may not get good results.
Error rates based on ELANDv2 output underestimate the true error rate. Since reads with many errors may not get aligned, they do not contribute to the calculation.

ELANDv2
As of CASAVA v1.6 a new version of ELAND is available, ELANDv2. The most important improvement of ELANDv2 are the following:

- Performs multiseed and gapped alignments. As a consequence, ELANDv2 handles indels and mismatches better.
- Enhanced match descriptor options to handle the gaps identified.
- Ability to split queries on a per tile basis now to allow for much greater parallelization.

More information about ELANDv2 is available in Appendix F, *ELANDv2 Algorithm*. 
GERALD Input Files

GERALD needs a Basecalls directory with _qseq files, a config file, and a reference genome for alignments. In general, GERALD will also need the files config.xml and BustardSummary.xml from the Basecalls (or Bustard) directory. For some applications, additional files are required. These files are described below.

The reference files for many builds, such as human genome build 36, mouse genome build 37, and rat genome build 3.4, are provided. If you want to use a different genome or build, you can generate reference files as described in Squashing the Reference Files on page 136.

For file locations, see Figure 9. Note that the reference files may be located in a different location, depending on your CASAVA installation.

![Figure 9 Locations of GERALD Input and Output Files](image)
GERALD Input Files

Sequence Files

GERALD needs a Basecalls (or Bustard) directory as generated by RTA or OLB, which contains the sequence files (_qseq files). These files have the following format:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine name</td>
<td>Identifier of the sequencer.</td>
</tr>
<tr>
<td>Run number</td>
<td>Number to identify the run on the sequencer.</td>
</tr>
<tr>
<td>Lane number</td>
<td>Positive integer (currently 1-8).</td>
</tr>
<tr>
<td>Tile number</td>
<td>Positive integer.</td>
</tr>
<tr>
<td>X</td>
<td>X coordinate of the spot. Integer.</td>
</tr>
<tr>
<td></td>
<td>As of RTA v1.6, OLB v1.6, and CASAVA v1.6, the X and Y coordinates for each</td>
</tr>
<tr>
<td></td>
<td>clusters are calculated in a way that makes sure the combination will be</td>
</tr>
<tr>
<td></td>
<td>unique. The new coordinates are the old coordinates times 10, +1000, and then</td>
</tr>
<tr>
<td></td>
<td>rounded.</td>
</tr>
<tr>
<td>Y</td>
<td>Y coordinate of the spot. Integer.</td>
</tr>
<tr>
<td></td>
<td>As of RTA v1.6, OLB v1.6, and CASAVA v1.6, the X and Y coordinates for each</td>
</tr>
<tr>
<td></td>
<td>clusters are calculated in a way that makes sure the combination will be</td>
</tr>
<tr>
<td></td>
<td>unique. The new coordinates are the old coordinates times 10, +1000, and then</td>
</tr>
<tr>
<td></td>
<td>rounded.</td>
</tr>
<tr>
<td>Index</td>
<td>Positive integer. No indexing should have a value of 0.</td>
</tr>
<tr>
<td>Read Number</td>
<td>1 for single reads; 1 or 2 for paired ends or multiplexed single reads; 1, 2,</td>
</tr>
<tr>
<td></td>
<td>or 3 for multiplexed paired ends.</td>
</tr>
<tr>
<td>Sequence</td>
<td>Called sequence of read.</td>
</tr>
<tr>
<td>Quality</td>
<td>The calibrated quality string.</td>
</tr>
<tr>
<td>Filter:</td>
<td>Did the read pass filtering? 0 - No, 1 - Yes.</td>
</tr>
</tbody>
</table>

Quality Scores

The default quality scoring scheme is the Phred scoring scheme, encoded as an ASCII character by adding 64 to the Phred value. A Phred score of a base is:

$$Q_{\text{phred}} = -10 \log_{10}(e)$$

where \(e\) is the estimated probability of a base being wrong.

Read Segment Quality Control Metric

A number of factors can cause the quality of base calls to be low at the end of a read. For example, phasing artifacts can degrade signal quality in some reads, and the affected portions of these reads have high error rates and unreliable base calls. Typically, the increase in phasing causes quality scores to be low in these regions, and thus these unreliable bases are scored correctly.
However, the occurrence of phasing artifacts may not always correlate with segments of high miscall rates and biased base calls, and therefore these low quality segments are not always reliably detected by our current quality scoring methods. We therefore mark all reads that end in a segment of low quality, even though not all marked portions of reads will be equally error prone.

The read segment quality control metric identifies segments at the end of reads that may have low quality, and unreliable quality scores. If a read ends with a segment of mostly low quality (Q15 or below), then all of the quality values in the segment are replaced with a value of 2 (encoded as the letter B in Illumina’s text-based encoding of quality scores). We flag these regions specifically because the initially assigned quality scores do not reliably predict the true sequencing error rate. This Q2 indicator does not predict a specific error rate, but rather indicates that a specific final portion of the read should not be used in further analyses.

This is not a read-level filter; the occurrence of consecutive Q2 values in a read does not indicate that the read itself is unreliable, but rather that only the base calls flagged with Q2 are unreliable. Note, however, that these regions are included in the Gerald error rate calculations for aligned reads. In typical sequencing runs, most reads are reliable over their entire length, and are not marked with Q2 indicators. Of the reads that are marked with the Q2 indicator, most are flagged only in the final few cycles. The number of reads marked by the quality control indicator, and the extent of the marking, can be used as an overall run quality metric.

Config.txt File

The GERALD configuration file (config.txt) specifies what analysis should be done for each lane. The requirements and options for the GERALD configuration file are described in GERALD Configuration File on page 31.

Reference Genome

PhageAlign uses a reference genome in FASTA format; all ELANDv2 modules use the same genome sequence files but in a squashed format. These files are derived from FASTA files that contain one assembled chromosome, available from UCSC or NCBI. These FASTA files must be squashed into the 2-bits-per-base format that the ELANDv2 aligner understands.

The result is one fa.2bpp file and one fa.vld file for each squashed chromosome. If a file contains multiple sequences (e.g. several scaffolds in one multi-FASTA file) there will be an extra fa.idx file.

The information contained in the former includes the start and end cycles of each read. (This is how GERALD can expand asterisks in a specified USE_BASES.)

BustardSummary.xml File

The BustardSummary.xml file in the Basecalls (or Bustard) directory contains the intensity results that are merged with GERALD’s alignment results to produce Summary.xml and Summary.htm.

Base Calling Config.xml File

The base calling configuration file (config.xml) in the Basecalls (or Bustard) directory includes the start and end cycles of each read.
Additional Eland_rna Input Files

In addition to the reference genome, eland_rna needs the following input files, in squashed format:

- A set of splice junction sequences, generated using the same genome build.
- A set of contaminant sequences for the genome—typically the mitochondrial and ribosomal sequences.

Repeats

Optional: You can provide a path to a list of repeats, in squashed format. ELANDv2 will align and mask reads aligning to the repeats. This will speed up ELANDv2 analysis.

_qval.txt File

_qval.txt files are optional files produced in Bustard when the "--with-qval" is used. They are necessary if you want to perform custom quality recalibration.

GERALD Workflow

The standard workflow for invoking the GERALD modules is as follows:

1. Navigate (via the command line) to the Run Folder location.
2. Create a configuration file that specifies what analysis should be done for each lane.
3. Check the Run Folder by running the command without --make.
4. Add command line options, generate the analysis folder, and corresponding makefiles.
5. Navigate to the analysis directory and start your analysis by executing makefiles.
Running GERALD

GERALD uses a text-based configuration file containing all parameters required for alignment, visualization, and filtering. These parameters specify the type of analysis to perform, which bases to use for alignment, and the reference files for a sequence alignment.

GERALD is a collection of Perl scripts and C++ executables, and is managed by the “make” utility. The “make” utility is commonly used to build executables from source code and is designed to model dependency trees by specifying dependency rules for files. These dependencies are stored in a file called a makefile. The GERALD.pl script is used to generate a makefile.config containing variable definitions which uses static makefiles as required. These static makefiles (including the main Makefile) have fixed content and can be included in the distribution and do not have to be regenerated for every run.

**NOTE** Static makefiles are new as of CASAVA v1.6. This does not result in any changes to the GERALD.pl and make usage below, compared to older GERALD versions.

**Standard GERALD Analysis**

The standard way to run GERALD is to set the parameters in a configuration file, create a makefile, and start the analysis with the “make” command.

1. Edit the config.txt file as described in GERALD Configuration File on page 31.
2. Enter the following command to create a makefile for sequence alignment with the desired compression option.

   ```bash
   /path-to-CASAVA/bin/GERALD.pl config.txt --EXPT_DIR path_to_SCS_Analysis_folder --make
   ```

3. Move into the newly created GERALD folder under the “path_to_SCS_Analysis_folder”. Type the “make” command for basic analysis:

   ```bash
   make
   ```

   You may prefer to use the parallelization option as follows:

   ```bash
   make -j 3 all
   ```

   The extent of the parallelization depends on the setup of your computer or computing cluster. For a description of parallelization, see Using Parallelization in GERALD on page 109.

4. After the analysis is done, review the analysis:
   a. View the analysis results of your run. See Visual Analysis Summary on page 44 and Text-Based Analysis Results on page 51.
   b. Interpret the run quality. See Interpretation of Run Quality on page 53.
GERALD Configuration File

This section describes a typical GERALD configuration file that uses the current features and parameters.

As part of the creation of the GERALD output folder, the GERALD configuration file specified is copied to the GERALD output folder using the filename config.txt. Some sites use standard configuration files, which may be stored in a central repository.

The GERALD configuration file specifies what analysis should be done for each lane, which GERALD translates into a makefile. The makefile specifies exactly what commands should be executed to carry out the requested analysis.

NOTE
If no analysis is provided for a lane or indexed lane, GERALD will assign it a value of ANALYSIS: none.

Table 2 GERALD Configuration File Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXPT_DIR data/070813_ILMN-1_0217_FC1234/Data/Intensities/Basecalls/</td>
<td>Provide the path to the experiment directory in the run folder, if not specified on the command line. For a description of the run folder, see Appendix B, Understanding the Run Folder.</td>
</tr>
<tr>
<td>USE_BASES nY*n</td>
<td>Ignore the first and last base of the read. The USE_BASES string contains a character for each cycle.</td>
</tr>
<tr>
<td></td>
<td>• If the character is “Y”, the cycle is used for alignment.</td>
</tr>
<tr>
<td></td>
<td>• If the character is “n”, the cycle is ignored.</td>
</tr>
<tr>
<td></td>
<td>• Wild cards (*) are expanded to the full length of the read.</td>
</tr>
<tr>
<td></td>
<td>For a detailed description of USE_BASES syntax, see USE_BASES Option on page 35.</td>
</tr>
<tr>
<td>ELAND_GENOME /home/user/Genomes/Eland/BAC_plus_vector/</td>
<td>Specify the genome reference for alignment with ELANDv2.</td>
</tr>
<tr>
<td>GENOME_DIR /home/user/Genomes GENOME_FILE BAC_plus_vector.fa</td>
<td>Specify the genome reference directory and file for alignment with PhageAlign.</td>
</tr>
<tr>
<td>ANALYSIS eland_extended</td>
<td>Specify the type of alignment that should be performed. Available options are:</td>
</tr>
<tr>
<td></td>
<td>• ANALYSIS eland_extended</td>
</tr>
<tr>
<td></td>
<td>• ANALYSIS eland_pair</td>
</tr>
<tr>
<td></td>
<td>• ANALYSIS eland_rna</td>
</tr>
<tr>
<td></td>
<td>• ANALYSIS default</td>
</tr>
<tr>
<td></td>
<td>• ANALYSIS sequence</td>
</tr>
<tr>
<td></td>
<td>• ANALYSIS sequence_pair</td>
</tr>
<tr>
<td></td>
<td>• ANALYSIS none</td>
</tr>
<tr>
<td></td>
<td>The default is ANALYSIS none</td>
</tr>
<tr>
<td></td>
<td>See ANALYSIS Variables on page 34 for more information.</td>
</tr>
</tbody>
</table>
Lane-Specific Options
The following table describes the lane-specific parameters in a GERALD configuration file.

Table 3  GERALD Configuration File Lane-Specific Options

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:USE_BASES nY20</td>
<td>Align only 20 cycles for lane 7, starting with the second cycle.</td>
</tr>
<tr>
<td>567:ANALYSIS eland_extended</td>
<td>Align lanes 5, 6, and 7 only against a genomic sample.</td>
</tr>
<tr>
<td>567:USE_BASES all</td>
<td></td>
</tr>
<tr>
<td>8:ANALYSIS none</td>
<td>Omit lane 8, which contains only primers.</td>
</tr>
<tr>
<td>3:QCAL_SOURCE auto8</td>
<td>Lane 3 will use the lane 8 qtable.</td>
</tr>
<tr>
<td>123:QCAL_SOURCE auto8</td>
<td>Lanes 1–3 will use the lane 8 qtable.</td>
</tr>
</tbody>
</table>

Optional Parameters
The following table describes the optional parameters in a GERALD configuration file.

Table 4  GERALD Configuration File Optional Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLOW_CELL</td>
<td>Specifies the type of flow cell and associated chemistry that was used for the sequencing run. Default is “v4” for a flow cell v4, other option is “1.4mm” for a 1.4 mm flow cell. The sequencing chemistries for the 1.4 mm flow cell and flow cell v4 are different, and GERALD has to adjust base call calibration based on the chemistry. If you used SCS v2.5 or later for sequencing, GERALD can determine the flow cell type and chemistry used from the AnalysisInfo.xml file, as long as this file is in the standard location (&lt;run-folder&gt;/Config/). If you used an earlier version of SCS, or if the AnalysisInfo.xml file is not in the standard location, you need to specify the option --flow-cell. This only needs to be done for the first analysis module used; for subsequent modules, this information will be stored in config.xml.</td>
</tr>
<tr>
<td>QCAL_SOURCE</td>
<td>This parameter specifies the base call calibration that is used. Allowed values are upstream, auto, auto&lt;n&gt;, or /path/to/qtable.txt</td>
</tr>
<tr>
<td></td>
<td>• upstream —No custom recalibration is performed at the GERALD analysis stage. The quality values produced (at the base calling stage) from a precalculated qtable supplied with the software will be used. This is the default.</td>
</tr>
<tr>
<td></td>
<td>• auto—The qtable(s) used within the lane (to reestimate the base call quality values) are the qtable(s) generated for that lane (from the quality predictor values and called and reference base values of bases in reads from that lane).</td>
</tr>
<tr>
<td></td>
<td>• auto&lt;n&gt;—The qtable(s) used in the lane are those generated for lane n. For example, “auto5” means that the qtable(s) from lane 5 are used.</td>
</tr>
<tr>
<td></td>
<td>• /path/to/qtable.txt—The qtable file at the specified path is used. See for QCAL_SOURCE Option on page 36 for more information.</td>
</tr>
<tr>
<td>BAD_TILES</td>
<td>Identify bad tiles.</td>
</tr>
<tr>
<td>s_1_0001</td>
<td></td>
</tr>
<tr>
<td>s_2_0003</td>
<td></td>
</tr>
</tbody>
</table>
Table 4  GERALD Configuration File Optional Parameters (Continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>SINGLESEED</td>
<td>If SINGLESEED is set to any value, ELANDv2 aligns only in singleseed mode. Only available for ANALYSIS eland_extended and ANALYSIS eland_pair, for which Multiseed alignment is default. See Multiseed and Gapped Alignment on page 141 for more information.</td>
</tr>
<tr>
<td>UNGAPPED</td>
<td>If UNGAPPED is set to any value, ELANDv2 aligns only in ungapped mode. Only available for ANALYSIS eland_extended and ANALYSIS eland_pair, for which gapped alignment is default. See Multiseed and Gapped Alignment on page 141 for more information.</td>
</tr>
<tr>
<td>SEQUENCE_FORMAT</td>
<td>This parameter specifies what format to use for data export in the s_N_sequence.txt file. Allowed values are --fasta, --fastq, or --scarf.</td>
</tr>
<tr>
<td></td>
<td>• FASTA—This format is widely used but does not contain quality scores.</td>
</tr>
<tr>
<td></td>
<td>• fastq—This format is an adoption of the FASTA format that contains quality scores. However, the fastq format is not completely compatible with the fastq files currently in existence, which is read by various applications (for example, BioPerl). Because a larger dynamic range of quality scores is used, the quality scores are encoded in ASCII as 64+score, instead of the standard 32+score. This method is used to avoid running into non-printable characters.</td>
</tr>
<tr>
<td></td>
<td>• scarf (Solexa compact ASCII read format)—This easy-to-parse, text-based format stores all the information for a single read in one line.</td>
</tr>
<tr>
<td>POST_RUN_COMMAND / yourPath/yourCommand yourArgs</td>
<td>Allows user-defined scripts to be run after all GERALD targets have been built.</td>
</tr>
<tr>
<td>EMAIL_LIST</td>
<td><a href="mailto:user@example.com">user@example.com</a> <a href="mailto:user2@example.com">user2@example.com</a> EMAIL_SERVER mailserver EMAIL_DOMAIN example.com</td>
</tr>
<tr>
<td></td>
<td>Send a notification to the user at the end of an analysis run. For more information on email notification, see Setting Up Email Reporting on page 90.</td>
</tr>
<tr>
<td>WEB_DIR_ROOT file:// server.example.com/share NUM_LEADING_DIRS_TO_STRIP</td>
<td>Include hyperlinks with a specific prefix to the Run Folder.</td>
</tr>
<tr>
<td></td>
<td>Specifies the number of directories to strip from the start of the full run folder path before prepending the WEB_DIR_ROOT.</td>
</tr>
<tr>
<td>ELAND_SET_SIZE</td>
<td>The maximum number of tiles analyzed by each ELAND process. The default value is 40 to ensure that the memory usage stays below 2 GB for a full 50G run (450,000 clusters/mm² x 2 x 100 paired-end run). Only available for ANALYSIS eland_extended, ANALYSIS eland_pair, and ANALYSIS eland_rna.</td>
</tr>
</tbody>
</table>

Paired-End Analysis Options

The following table describes the paired-end analysis options in a GERALD configuration file.

Table 5  GERALD Configuration File Paired-End Analysis Options

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANALYSIS eland_pair</td>
<td>Use the paired-end alignment mode of ELANDv2 to align paired reads against a target.</td>
</tr>
</tbody>
</table>
GERALD can be run in various analysis modes. Customize your analysis by specifying variables, parameters, and options.

**ANALYSIS Variables**

Set the ANALYSIS variable to define the type of analysis you want to perform for each lane. The various analysis modes include default, eland_extended, eland_pair, eland_rna, and none. You can mix and match analyses between lanes.

For all modes, except ANALYSIS none, you will get a sequence output file (s_N_sequence.txt) for each lane.

### Table 5  GERALD Configuration File Paired-End Analysis Options (Continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>USE_BASES Y*,nY*n</td>
<td>Use all bases of the first read and ignore the first and last base of the second read.</td>
</tr>
<tr>
<td>6:USE_BASES nY25</td>
<td>Ignore the first base on both the first and second read; use 25 bases each and ignore any other bases.</td>
</tr>
<tr>
<td>4:QCAL_SOURCE1 /home/illumina/ref51_qtable.txt</td>
<td>Lane 4 read 1 will use the external ref51_qtable.txt qtable</td>
</tr>
<tr>
<td>4:QCAL_SOURCE2 auto7</td>
<td>Lane 4 read 2 will use the lane 7 read 2 qtable</td>
</tr>
</tbody>
</table>

For more information on USE_BASES syntax, see **USE_BASES Option** on page 35.

---

**GERALD Parameters**

### Table 6  ANALYSIS Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Alignment Program</th>
<th>Application</th>
<th>Description</th>
</tr>
</thead>
</table>
| ANALYSIS eland_extended | ELANDv2       | Single reads | Aligns single-read data reads against a target using ELANDv2 alignments.  
  • Works well with reads > 32 bases  
  • Each alignment is given a confidence value based on its base quality scores  
  • A single file of sorted alignments is produced for each lane  
  For a detailed description, see **Using ANALYSIS eland_extended** on page 39. |
| ANALYSIS eland_pair  | ELANDv2       | Paired reads | Aligns paired-end reads against a target using ELANDv2 alignments. A single-read alignment is done for each half of the pair, and then the best-scoring alignments are compared to find the best paired-read alignment. For a detailed description, see **Using ANALYSIS eland_pair** on page 40. |
| ANALYSIS eland_rna   | ELANDv2       | Single reads | Aligns each read against a large reference genome, splice junctions, and contaminants using ELANDv2.  
  For more information on ELAND_rna, see **Using ANALYSIS eland_rna** on page 43. |
| ANALYSIS default     | Phage Align     | Single reads | Aligns each read against a reference sequence using PhageAlign. This mode is suitable only for small genome references. |
**Table 6**  ANALYSIS Variables (Continued)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Alignment Program</th>
<th>Application</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANALYSIS sequence</td>
<td>None</td>
<td>Single reads</td>
<td>A sequence.txt file is produced but no alignment is performed.</td>
</tr>
<tr>
<td>ANALYSIS sequence_pair</td>
<td>None</td>
<td>Paired reads</td>
<td>A sequence.txt file is produced for each read but no alignment is performed.</td>
</tr>
<tr>
<td>ANALYSIS none</td>
<td>None</td>
<td>Any application</td>
<td>Omits the indicated lane from the analysis. Setting the parameter 8:ANALYSIS none ignores lane 8.</td>
</tr>
</tbody>
</table>

**Lane-by-Lane Parameters**

You can set the analysis parameter and other parameters on a lane-by-lane basis. You will need to do this for any parameters specific to the analysis of a particular lane.

**Table 7**  Lane-by-Lane Parameters

<table>
<thead>
<tr>
<th>Option</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>6:ELAND_GENOME /directory/genome</td>
<td>ELAND_GENOME points to a directory of squashed genome files. Specify the name of the folder containing the reference sequence(s) for lane 6.</td>
</tr>
<tr>
<td>67:ELAND_GENOME /directory/genome</td>
<td>Specify the name of the file containing the reference sequence to use for lanes 6 and 7.</td>
</tr>
</tbody>
</table>

**NOTE**
The best way to filter out individual tiles is to set BAD_TILES to be a list of the tiles you want to filter. See Optional Parameters on page 32 for an example of the BAD_TILES parameter.

**USE_BASES Option**
The USE_BASES option identifies which bases of a full read produced by a sequencing run should be used for the alignment analysis. A fully expanded USE_BASES value is a string with one character per sequencing cycle but more compact formats can be used as described in Table 8 on page 36. Each character in the string identifies whether the corresponding cycle should be aligned. The following notation is used:

- A lower-case “n” means ignore the cycle.

**NOTE**
Prephasing correction cannot be applied to the last base, since you need to know the next base in the sequence. Thus there will be a minor error increase at the last base. Ignoring the last base from the sequence analysis can reduce alignment errors somewhat.

- An upper-case “Y” means use the cycle for the alignment.
A comma (,) denotes a read boundary used for multiple reads.
- An asterisk (*) means “fill up the read as far as possible with the preceding character.”
- A number means that the previous character is repeated that many times. Unspecified cycles are set to “n” by default. If USE_BASES is not specified at all, every cycle is used for the alignment.

The following table describes examples of USE_BASES options.

**Table 8** USE_BASES Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>USE_BASES nYYY</td>
<td>Ignore the first base and use bases 2–4.</td>
</tr>
<tr>
<td>USE_BASES Y30</td>
<td>Align the first 30 bases.</td>
</tr>
<tr>
<td>USE_BASES nY30</td>
<td>Ignore the first base and align the next 30 bases.</td>
</tr>
<tr>
<td>USE_BASES nY30n</td>
<td>Ignore the first base, align the next 30 bases, and ignore the last base.</td>
</tr>
<tr>
<td>USE_BASES nY*n</td>
<td>Ignore the first base, perform a single read alignment, and ignore the last base. The length of read is automatically set to the number of sequencing cycles minus two.</td>
</tr>
<tr>
<td>USE_BASES nY*,nY*</td>
<td>Ignore the first base of each read and perform a paired read alignment, resulting in the length of each read being set to the number of sequencing cycles associated with it minus one. The two reads do not need to be of the same length.</td>
</tr>
<tr>
<td>USE_BASES nY*</td>
<td>When used with ANALYSIS eland_pair, this is an abbreviation for USE_BASES nY*,nY*. When used with a single-read analysis mode, this means ignore the first base and perform a single-read alignment.</td>
</tr>
<tr>
<td>USE_BASES Y*n</td>
<td>When used with a single-read analysis mode, this means perform a single-read alignment and ignore the last base. When used with ANALYSIS eland_pair, this is an abbreviation for USE_BASES Y<em>n,Y</em>n.</td>
</tr>
<tr>
<td>USE_BASES all</td>
<td>Use all bases.</td>
</tr>
</tbody>
</table>

**QCAL_SOURCE Option**

The default base call quality calibration is performed using a precalculated calibration table supplied with the software.

The source of the quality table used in the quality calibration for a lane may be overridden by defining QCAL_SOURCE (or, for individual reads within a paired read analysis, QCAL_SOURCE1 and/or QCAL_SOURCE2) in the config.txt passed to GERALD.pl to configure GERALD analysis.

The supported values of the QCAL_SOURCE variables are listed below:

**Table 9** QCAL_SOURCE Variable Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>upstream</td>
<td>The reestimation of base call quality values is performed before the alignment stage ('upstream' of GERALD in the base calling stage) using a precalculated quality table supplied with the software.</td>
</tr>
</tbody>
</table>
Table 9  QCAL_SOURCE Variable Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto</td>
<td>The qtable(s) used within the lane (to reestimate the base call quality values) are the qtable(s) generated for that lane (from the quality predictor values and called and reference base values of bases in reads from that lane). QCAL_SOURCE auto will only work if Bustard has been run with the option --with-qval.</td>
</tr>
<tr>
<td>auto&lt;n&gt;, where n is the number of a lane for which alignment will be performed</td>
<td>The qtable(s) used in the lane are those generated for lane n. For example, “auto5” means that the qtable(s) from lane 5 are used. QCAL_SOURCE auto will only work if Bustard has been run with the option --with-qval.</td>
</tr>
<tr>
<td>/path/to/qtable.txt</td>
<td>The qtable file at the specified path is used. QCAL_SOURCE auto will only work if Bustard has been run with the option --with-qval.</td>
</tr>
</tbody>
</table>

**Paired-End Analysis**

For cases where paired-end analysis is in use, the following principles apply:

1. If both lanes are paired, then any specification of the source lane for read 1 of the target lane results in the read 1 qtable of the source lane being used as the read 1 qtable in the target lane, and similarly for read 2.

2. If only the target lane is paired, then there is only one qtable available in the source lane but it will be used for both reads in the target lane.

3. If only the source lane is paired, then its read 1 qtable is (arbitrarily) used.

**NOTE**

In a paired-read analysis lane, specification of QCAL_SOURCE1 and/or QCAL_SOURCE2 will override specification of QCAL_SOURCE, although the latter will be used if it has been specified and not overridden for a given read.

Do not use upstream calibration for one read and any type of custom calibration for the other read. This is not supported.

**Example**

The following example specifies the following condition:

1. Lanes 1 and 2 will not be recalibrated; the quality values derived in the base calling stage before CASAVA (using a precalculated calibration table) will be retained.

2. Lane 3 read 1 will use the lane 8 read 1 qtable; lane 3 read 2 will use the lane 8 read 2 qtable.

3. Lane 4 read 1 will use the external ref51_qtable.txt qtable

4. Lane 4 read 2 will use the lane 7 read 2 qtable

5. Lanes 5–8 reads 1 and 2 will use the ref42_qtable.txt qtable

ANALYSIS eland_pair
QCAL_SOURCE /home/illumina/ref42_qtable.txt
12:QCAL_SOURCE upstream
3:QCAL_SOURCE auto8
4:QCAL_SOURCE1 /home/illumina/ref51_qtable.txt
4:QCAL_SOURCE2 auto7

Note that even though the lane 8 qtables will not be needed in lane 8, they will still be generated for use in lane 3.

Make Option

The --make option creates GERALD directories and makefiles. Without the option, GERALD.pl will not create any directories and files and only operates in a diagnostic mode. You must specify this option to generate the GERALD analysis folder and subsequently run the analysis.

Rerunning the Analysis

The config.txt file used to generate an analysis is copied to the analysis folder so it can be used by GERALD if a reanalysis of the same data is required. To change parameters and rebuild the analysis, modify the configuration file and run the following command:

```
GERALD.pl config.txt --make
```

By adding the OUT_DIR option, you can force GERALD to overwrite an existing makefile. This way you can modify the analysis without directly editing the makefile.

Parallelization Switch

If your system supports automatic load-sharing to multiple CPUs, you can parallelize the analysis run to <n> different processes by using the “make” utility parallelization switch.

```
make recursive -j n
```

For more information on parallelization, see Appendix C, Using Parallelization in GERALD.

Nohup Command

You should use the Unix nohup command to redirect the standard output and keep the “make” process running even if your terminal is interrupted or if you log out. The standard output will be saved in a nohup.out file and stored in the location where you are executing the makefile.

```
nohup make recursive -j n &
```

The optional “&” tells the system to run the analysis in the background, leaving you free to enter more commands.

ELANDv2 Alignments

Ensure the configuration file you use to run GERALD contains the following components:

- The path to your squashed genome files:
  ELAND_GENOME /data/Genome/ELAND/hg18
- The path to your list of repeats (optional):
  ELAND_REPEAT /data/Genome/ELAND/hg18/reps30_5
  This can significantly speed up alignment against large targets.
- The analysis mode to run ELANDv2:
ANALYSIS eland_extended

- Particular lanes that you want to analyze in the analysis mode:
  34:ANALYSIS eland_extended
  This example indicates that lane 3 and 4 will be analyzed.

NOTE

ELAND_GENOME refers to a directory, not a file. The GERALD variables GENOME_DIR and GENOME_FILE are not used for ELANDv2 analysis. ELANDv2 expects a file format other than FASTA.
You can only specify one ELAND_GENOME per lane, or for multiplexing, one ELAND_GENOME per lane per multiplexed sample.

After setting up the config, you need to run GERALD.pl and then run “make”.

Missing Bases in ELANDv2

Missing bases need to be specified as “N” characters and not “.”. This conversion is managed automatically by GERALD but you need to be aware of it when running ELANDv2 as a standalone program. To attempt an alignment, ELANDv2 requires that at most half of the bases in the first 32 bases of the read consist of leading or trailing Ns.
Alternatively, at most two internal N characters are permitted.
An internal N is interpreted as a base that is present but not detected.
Read: ACNGT
Genome: ACCGT

Most N characters are due to clusters wandering off the edge of the image for a cycle or two due to imperfect re-mapping of the tile position at different cycles. Otherwise, the GERALD software will try to make a base call, even if the call is of low quality.

Using ANALYSIS eland_extended

ANALYSIS eland_extended is an improved version of the ANALYSIS eland mode that existed in Pipeline and is now deprecated. ANALYSIS eland could align reads longer than 32 bases but demanded that the first 32 bases of the read have a unique best match in the genome. The position of this match is used as a “seed” to extend the match along the full length of the read.
ANALYSIS eland_extended removes the uniqueness restriction by considering multiple 32 base matches and extending them.

Multiseed and Gapped Alignment

ANALYSIS eland_extended by default performs multiseed alignment by aligning the first set of 16 to 32 bases and consecutive sets of bases separately. ANALYSIS eland_extended uses a gapped alignment method to extend each candidate alignment to the full length that allows for gaps (indels) of up to 20 bases. For more information, see Multiseed and Gapped Alignment on page 141.
CHAPTER 3
Using GERALD for Sequence Alignment

Configuring ANALYSIS eland_extended

There are two parameters that affect the output of the alignment, ELAND_SEED_LENGTH and ELAND_MAX_MATCHES. Both parameters can be specified lane-by-lane.

The following table describes the parameters for ANALYSIS eland_extended.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELAND_SEED_LENGTH</td>
<td>By default, the first 32 bases of the read are used as a “seed” alignment. Setting ELAND_SEED_LENGTH to 25 will use 25 bases instead of the maximum of 32 for the initial seed alignment. This should increase the sensitivity since two errors per 25 bases is less stringent than two errors per 32 bases. A read is more likely to be repetitive at the 25 base level than at the 32 base level, so a decrease in ELAND_SEED_LENGTH should probably be used in conjunction with an increase in ELAND_MAX_MATCHES. Setting this to very low values will drastically slow down the alignment time and will probably result in a lot of poor confidence alignments.</td>
</tr>
<tr>
<td>ELAND_MAX_MATCHES</td>
<td>By default, ANALYSIS eland_extended will consider at most ten alignments of each read. ELAND_MAX_MATCHES allows the maximum number of alignments considered per read to be varied between 1 and 255.</td>
</tr>
</tbody>
</table>

Both ANALYSIS eland_extended and ANALYSIS eland_pair share a common export file that contains all read, quality value, and alignment information for a lane of data.

- ANALYSIS eland_extended produces a single file per lane (s_N_export.txt).
- ANALYSIS eland_pair produces two files, one for each of the two reads (s_N_1_export.txt and s_N_2_export.txt).

For a detailed description of the export.txt files, see Text-Based Analysis Results on page 51.

Using ANALYSIS eland_pair

Based heavily on ANALYSIS eland_extended, ANALYSIS eland_pair allows the analysis of a paired-read run using ELANDv2 alignments. As part of the analysis, it will:

- Align both read 1 and read 2 to the reference genome
- Determine the insert size distribution of the sample
- Use the insert size distribution to resolve repeats and ambiguities

The files s_N_1_export.txt and s_N_2_export.txt are meant to contain all information necessary for downstream processing of the alignment data. Other files produced that may be useful in some circumstances are:

- s_N_1_eland_extended.txt, s_N_2_eland_extended.txt - these contain the candidate alignments for each read 1 and read 2. The software chooses from these possibilities in attempting to pick the best alignment of the read pair.
Another output file produced is s_N_anomaly.txt, which contains reads that do not align. For some applications, reads that do not align may be of interest, since amongst those that are due to read errors may be some that represent genuine differences between the sequenced DNA and the reference.

For a detailed description of the export.txt files, see Text-Based Analysis Results on page 51.

### Multiseed and Gapped Alignment

ANALYSIS eland_pair by default performs multiseed alignment by aligning the first set of 16 to 32 bases and consecutive sets of bases separately. ANALYSIS eland_pair uses a gapped alignment method to extend each candidate alignment to the full length that allows for gaps (indels) of up to 20 bases. For more information, see Multiseed and Gapped Alignment on page 141.

### Configuring a Paired-Read Analysis

The alignments of the two reads that provide input to the pairing process may be varied by setting ELAND_SEED_LENGTH and ELAND_MAX_MATCHES. Both parameters may be set lane-by-lane, but the same values will apply to each of the two reads in a lane.

The paired-read analysis may be configured by passing options to pickBestPair. This is done by setting a parameter PAIR_PARAMS in the GERALD configuration file. For additional information, see GERALD Configuration File on page 31.

PAIR_PARAMS can be specified lane-by-lane. All of the options must be specified on a single line and space-separated, as in the following example:

```
8:PAIR_PARAMS --circular --min-percent-unique-pairs=30
```

The following table describes the parameters for ANALYSIS eland_pair.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| --circular         | This causes pickBestPair to treat each chromosome as circular and not linear, enabling it to detect valid pairings that “wrap around” when the two alignments are mapped onto the linear representation of the chromosome.  
                     | --circular=my mitochondria_file.fa  
                     | Treat alignments to my mitochondria_file.fa as circular but other chromosomes as linear (as you might want to do when e.g. aligning to the whole human genome)  
                     | --circular=chromosome1:100000,chromosome2:300000  
                     | Specify chromosomes to circularize and specify the size to “wrap around” (possibly of use when the chromosome size is uncertain) |
### Table 11 Parameters for ANALYSIS eland_pair (Continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--min-percent-unique-pairs</td>
<td>A unique pair is defined as a read pair such that its constituent reads can each be aligned to a unique position in the genome without needing to make use of the fact that they are paired. pickBestPair works in a two-pass fashion: 1. On the first pass it looks for all clusters that pass the quality filter and have a unique alignment of each of their two reads, then uses this information to determine the nominal insert size distribution and the relative orientation of the two reads. 2. On a second pass this information is used to resolve repeats and other ambiguous cases. The number of unique pairs, expressed as a percentage of the total number of clusters passing filters, must exceed a certain percentage. Otherwise, no pairing is attempted and the two reads are effectively treated as two sets of single reads. • By default, this threshold is set to 10%. • For low quality data, a pairing can be forced by setting --min-percent-unique-pairs=5. For some applications it may be useful to switch off the pairing completely. Set --min-percent-unique-pairs=101.</td>
</tr>
<tr>
<td>--min-percent-consistent-pairs</td>
<td>Of the unique pairs, the vast majority should have the same orientation with respect to each other. If they don’t, it is indicative of the following problems: • Sample prep • Circularization is not switched on • A reference sequence is extremely diverged from the sample data In such cases, no pairing is attempted and the two reads are effectively treated as two sets of single reads. By default, the threshold for this parameter is set to 70%.</td>
</tr>
<tr>
<td>--min-paired-read-alignment-score</td>
<td>For each cluster, all possible pairings of alignments between the two reads are compared. This is the score of the best one. Since we are considering the two reads as one, both reads in a cluster get the same paired-read alignment score. The alignment score is nominally on a Phred scale. However, it is probably not safe to assume the calibration is perfect. Nevertheless, it is a good discriminator between good and bad alignments. The score must exceed this threshold to go in the sorted.txt file. The default value is four.</td>
</tr>
<tr>
<td>--min-single-read-alignment-score</td>
<td>Each read is given a single-read alignment score. This is identical to the alignment score from an eland_extended analysis. If a read has a zero paired-read alignment score, but a single-read alignment score that exceeds this threshold, its alignment will still go in the sorted.txt files. If the alignments of the two reads can not be paired (resulting in a zero paired score) and only one of the reads has an alignment exceeding --min-single-read-alignment-score, the read pair is treated as a singleton. The alignment of the shadow read is unreliable enough to be ignored. The default value is four.</td>
</tr>
<tr>
<td>--add-shadow-to-singleton-threshold</td>
<td>If one read has a score exceeding --min-single-read-alignment-score but the other read either has no alignments or an alignment that does not exceed --min-single-read-alignment-score, then the non-aligning “shadow” read is added to the sorted.txt file with a zero alignment score, if the combined base quality of the shadow read (not alignment quality) exceeds this threshold. The default value of 1,000,000 indicates this feature is switched off.</td>
</tr>
</tbody>
</table>
Using ANALYSIS eland_rna

Eland_rna is the eland module built specifically for RNA Sequencing, and is required to provide the input files for CASAVA. Eland_rna delivers the following information:

- Read alignments to the genome.
- Read alignments to splice junctions.
- Read alignments to contaminants.

**NOTE**

Eland_rna does not support paired-end cDNA reads yet.

**Prerequisites**

Three sets of data files are needed:

- A genome sequence.
- A set of splice junction sequences, generated using the same genome build. This is provided for human, rat, and mouse for two read length: splice-34 to use for alignments of 35 nt sequences and splice-49 to use for alignments of 50 nt sequences.
- A set of contaminant sequences for the genome - typically the mitochondrial and ribosomal sequences.

All three of these datasets must be squashed into the 2-bits-per-base format that the ELANDv2 aligner understands. This is done by calling the program squashGerome in the Eland directory; the procedure is described in Squashing the Reference Files on page 136.

**Description of the Eland_rna Algorithm**

The algorithm aligns the reads to each of three targets:

- Contaminants
- Genome
- Splice junctions

Then a script decides which of the alignments is most likely for each read. The following steps are taken in order:

1. If a read aligns to the contaminants then the read is discarded. It is marked in the export file as 'RM' - for 'repeat masked.'

2. If the read aligns to the genome:
   - If there is a unique alignment to the genome then that alignment is printed.
   - If there are multiple possible alignments to the genome then the read is marked as 'RM' and discarded as above.
3. If there is no alignment to the genome but there is an alignment to the splice junctions:
   • If there is a unique alignment to the splice junctions then that alignment is printed.
   • If there are multiple alignments to the splice junctions then the read is marked as 'RM' and discarded.

4. If there is no alignment to either the contaminants, the genome or the splice junctions then the read is marked as 'NM' - for 'not matched.'

**Multiseed Alignment**

ANALYSIS eland_rna by default performs multiseed alignment by aligning the first set of 16 to 32 bases and consecutive sets of bases separately. For more information, see **Multiseed and Gapped Alignment** on page 141.

**Running an Eland_rna Analysis**

The GERALD configuration file specifies how the sequences from a flow cell are processed, which is described in **GERALD Configuration File** on page 31. The ANALYSIS parameter within the GERALD configuration file specifies what analysis to perform on the sequences; you will need to set up this parameter the following way (example shown):

```
ANALYSIS eland_rna
ELAND_GENOME /data/Genome/ELAND/hg18
ELAND_RNA_GENOME_SPLICE /data/Genome/ELAND_RNA/Human/human.34.splice
ELAND_RNA_GENOME_CONTAM /data/Genome/ELAND_RNA/Human/MT_Ribo_Filter
```

This tells GERALD it needs to perform eland_rna, and communicates the locations of the squashed genome, splice-junction and contaminant files.

The following table describes the parameters for ANALYSIS eland_pair.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELAND_GENOME</td>
<td>Must point to a squashed version of the reference genome, just as for a standard ELANDv2 analysis.</td>
</tr>
<tr>
<td>ELAND_RNA_GENOME_SPLICE</td>
<td>Must point to a squashed version of the splice junction file.</td>
</tr>
<tr>
<td>ELAND_RNA_GENOME_CONTAM</td>
<td>Must point to a squashed version of the files of ultra-abundant sequences (generally ribosomal and mitochondrial). Any read that hits to these is ignored.</td>
</tr>
</tbody>
</table>
**Considerations When Running Eland_rna**

When running eland_rna, bear in mind the following points:

- The above parameters may be specified on a lane-by-lane basis in the usual fashion, for example to do lanes one, two, and four, enter the following for aligning 35 bases:
  
  124:ANALYSIS eland_rna
  124:ELAND_GENOME /data/Genome/ELAND/hg18
  124:ELAND_RNA_GENOME_SPLICE /data/Genome/ELAND_RNA/
  Human/human.34.splice
  124:ELAND_RNA_GENOME_CONTAM /data/Genome/ELAND_RNA/
  Human/MT_Ribo_Filter

- The output file export.txt has the same format as those generated by eland_extended; for a description see Table 17 on page 55. The existing code ‘RM’ (‘repeat masked’) denotes all reads that hit to abundant sequences or with any other unresolvable ambiguity.
GERALD Output Files

The GERALD output files contain run information, statistitical analysis, sequence information, and alignment information. They are described below.

Visual Analysis Summary

The results of an analysis are summarized as web pages that enable a large number of graphs to be viewed as thumbnail images. This section is intended to help you interpret the various graphs that appear in an analysis directory.

As the numbers of tiles and graphs have increased, it has become impractical to generate every possible graph for every tile. Therefore, the pages should be considered as a very basic view of the data.

Results Summary

For each Run Folder, a Summary.xml and Summary.html file is produced, which contains comprehensive results and performance measures of your analysis run. It is located in the GERALD folder and provides an overview of quality metrics for a run with links to more detailed information in the form of pages of graphs. It is intended to load in a reasonable time; depending on the number of lanes and tiles used, the pages to which it links may take longer to display.

NOTE

If multiplexing is used, there are likely multiple GERALD folders per run folder, each with a Summary.xml and Summary.html.

The Summary.xml file is provided with its corresponding XSLT style sheet document (Summary.xsl). Provided that the style sheet document is in the same folder as the Summary.xml, one can open it using a browser (e.g. Firefox, Internet Explorer, etc). The XSLT transform is automatically applied to the XML document to display the data as HTML with tables, links and styled text. In addition, the users can also open the XML file in Microsoft Word and Excel (using the Open with Transform option of the Open dialog box).

In the new Summary.xml file, the information is marked by XML tags in a completely intuitive fashion. You can easily find the required information that is displayed on the ordinary Summary.htm page. Indeed, the names of the tags correspond to the different tables displayed in the HTML page (e.g ChipSummary, LaneParameterSummary, LaneResultsSummary, ExpandedLaneSummary and TileResultsByLane). Lane, read and tile numbers are identified by the corresponding tags, laneNumber, readNumber and tileNumber. The different columns in each table can also easily be identified based on their names. For instance, the corresponding XML data for "% Align (PF)" in table "Lane Results Summary: Read 1" for lane 1 would be:

```xml
<LaneResultsSummary>
<Read>
  <Lane>
    <percentUniquelyAlignedPF>
      <mean>83.72</mean>
      <stdev>5.25</stdev>
    </percentUniquelyAlignedPF>
  </Lane>
</Read>
</LaneResultsSummary>
```
In addition to Summary.xml, BustardSummary.xml and its XSLT style sheet document are generated at the end of each base calling run allowing an early inspection of the intensity related data.

In the following descriptions of the tables included in Summary.xml, the terms chip and flow cell are used interchangeably.

**Chip Summary**

The Chip Summary contains the instrument ID and the run folder. The Chip ID field is a placeholder that currently has a value of “unknown.”

**Chip Results Summary**

This table displays a summary of chip-wide performance statistics for the run. Both the original number of detected clusters and the number that passed quality filtering are shown. In addition, a chip in kilobases is presented. This is the sum over lanes of the product of number of quality-filtered clusters and number of bases per cluster.

**Lane Parameter Summary**

Lane Parameter Summary records information about the sample in each flow cell lane and the analysis that has been specified for it.

- **Sample ID**—This is a placeholder field that currently has a value of “unknown.”
- **Sample Target**—The reference sequence against which reads from the sample in this lane are to be aligned. Depending on the analysis mode, this may be the name of a folder containing one or more sequence files or the name of an individual file. The acceptable file formats also depend on the analysis mode.
- **Sample Type**—Contains the analysis mode for reads from this lane.
- **Length**—The number of bases used per read (excluding any bases masked out using USE_BASES). Where multiple reads are produced per cluster and a distinction is maintained between them during analysis, as in eland_pair analysis of paired-end reads, their respective lengths will be listed.
- **Filter**—The criterion for clusters to be selected for analysis beyond the preliminary stages. Statistics for all detected clusters and for the subset that pass filtering are annotated as “raw” and “PF,” respectively, in Summary.htm.
CHAPTER 3
Using GERALD for Sequence Alignment

- **Num Tiles**—The number of tiles from the lane that are used in the analysis.
- **Tiles**—A hyperlink for each lane to the location (within Summary.htm) of the statistics for individual tiles in that lane.

**Lane Results Summary**

This table displays basic data quality metrics for each lane. Apart from Lane Yield, which is the total value for the lane, all the statistics are given as means and standard deviations over the tiles used in the lane.

- **Clusters (raw)**—The number of clusters detected by the image analysis module.
- **Clusters (PF)**—The number of detected clusters that meet the filtering criterion listed in Lane Parameter Summary.
- **1st Cycle Int (PF)**—The average of the four intensities (one per channel or base type) measured at the first cycle averaged over filtered clusters.
- **% Intensity after 20 cycles (PF)**—The corresponding intensity statistic at cycle 20 as a percentage of that at the first cycle.
- **% PF Clusters**—The percentage of clusters passing filtering.
- **% Align (PF)**—The percentage of filtered reads that were uniquely aligned to the reference.
- **Alignment Score (PF)**—The average filtered read alignment score (reads with multiple or no alignments effectively contribute scores of 0).
- **% Error Rate (PF)**—The percentage of called bases in aligned reads that do not match the reference.

If eland_pair analysis has been specified for one or more lanes, then two Lane Results Summaries are produced, one for each read. All lanes for which analysis has been specified are represented in the Read 1 table, but only those for which eland_pair analysis has been specified contribute statistics to the Read 2 table.

**Expanded Lane Summary**

This displays more detailed quality metrics for each lane. Apart from the phasing and prephasing information, all values are tile means for the lane.

- **Clusters (tile mean) (raw)**—The number of clusters detected by the image analysis module.
- **% Phasing**—The estimated (or specified) value used for the percentage of molecules in a cluster for which sequencing falls behind the current position (cycle) within a read.
- **% Prephasing**—The estimated (specification is not recommended) value used for the percentage of molecules in a cluster for which sequencing jumps ahead of the current position (cycle) within a read.
- **% Error Rate (raw)**—The percentage of called bases in aligned reads from all detected clusters that do not match the reference.
- **Equiv Perfect Clusters (raw)**—The number of clusters in the ideal situation of read base perfectly predicting reference base that would provide the same information content (entropy of reference base given read base and a prior assumption of equiprobable reference bases) as calculated for all actual detected clusters.
% Retained — The percentage of clusters that passed filtering.

Cycle 2-4 Av Int (PF) — The intensity averaged over cycles 2, 3, and 4 for clusters that passed filtering.

Cycle 2-10 Av % Loss (PF) — The average percentage intensity drop per cycle over cycles 2–10 (derived from a best fit straight line for log intensity versus cycle number).

Cycle 10-20 Av % Loss (PF) — The average percentage intensity drop per cycle over cycles 10–20 (derived from a best fit straight line for log intensity versus cycle number).

% Align (PF) — The percentage of filtered reads that were uniquely aligned to the reference.

% Error Rate (PF) — The percentage of called bases in aligned filtered reads that do not match the reference.

Equiv Perfect Clusters (PF) — The number of clusters in the ideal situation of read base perfectly predicting reference base that would provide the same information content (entropy of reference base given read base and a prior assumption of equiprobable reference bases) as calculated for the actual clusters that passed filtering.

If eland_pair analysis has been specified for one or more lanes, then two Expanded Lane Results Summaries are produced, one for each read. All lanes for which analysis has been specified are represented in the Read 1 table, but only those for which eland_pair analysis has been specified contribute statistics to the Read 2 table.

Per-Tile Statistics

Below the two types of lane summaries are per-tile statistics, grouped into a table for each lane. The statistics are a subset of those already presented in the Lane Results Summary, but are presented in these tables as averages over the detected (raw) or filtered (PF) clusters in individual tiles.

In the event that no clusters in a tile pass filtering, all the statistics for that tile are displayed within square brackets. Such an occurrence suggests an exceptional situation (e.g., a bubble) within the tile. The brackets indicate the tile has been excluded from the calculation of lane statistics and that the values are reported only for diagnostic purposes.

Pair Summary

For lanes for which eland_pair analysis was performed, there are two per-tile summary tables (one for each read). These tables are preceded by a set of tables collectively entitled the Pair Summary. The Pair Summary tables provide statistics about the alignment outcomes of the two reads individually and as a pair, the latter including relative orientation and separation (insert size) of partner read alignments.

If the criteria for paired alignment are not met, the subset of tables reporting paired alignment results are replaced with the statement, “Paired alignment not performed.”

The following tables are displayed in Pair Summary:

- Relative Orientation Statistics
- Insert Size Statistics
- Insert Statistics (% of individually uniquely alignable pairs)
Relative Orientation Statistics—The relative orientation of a pair is the orientation of read 2 relative to the orientation of read 1, based on the definition that the read 1 orientation is forward. The relative orientation is defined as positive if the read 2 position is greater than the read 1 position.

These statistics are given only for those pairs in which both reads were individually uniquely aligned, since these are the reads used to determine the predominant relative orientation. Other orientations are considered anomalous and are filtered out.

The symbols used in the column headings are intended as a visual reminder of the definitions of the four possible relative orientations. In the example below, the nominal orientation is correctly computed as the two reads “pointing to” each other, as expected for the standard Illumina short insert paired-read sample prep.

Table 13  Example of Relative Orientation Statistics Table

<table>
<thead>
<tr>
<th></th>
<th>F-: &gt; R2 R1</th>
<th>F+: &gt; R1 R2</th>
<th>R-: &lt; R2 R1</th>
<th>R+: &gt; R1 R2</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>184 (0.0%)</td>
<td>161 (0.0%)</td>
<td>243 (0.0%)</td>
<td>2443273 (100.0%)</td>
<td>2443861</td>
</tr>
</tbody>
</table>

Insert Size Statistics—Statistics are derived from the insert sizes of those pairs in which both reads were individually uniquely aligned and have the predominant relative orientation. First, the median is determined. Then, a standard deviation value is determined independently for those values below the median and those above it. The lower and upper thresholds for acceptable insert sizes are then defined as three of the relevant standard deviations below and above the median, respectively.

Table 14  Example of Insert Size Statistics Table

<table>
<thead>
<tr>
<th></th>
<th>Median</th>
<th>Below-median SD</th>
<th>Above-median SD</th>
<th>Low Thresh.</th>
<th>High Thresh.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>214</td>
<td>10</td>
<td>11</td>
<td>184</td>
<td>247</td>
</tr>
</tbody>
</table>

Insert Statistics (% of individually uniquely alignable pairs)—This table shows the number of inserts (out of those used to calculate insert size statistics) considered acceptable in size and of those falling outside the thresholds displayed in the Insert Size Statistics table. The percentages are relative to the original number of pairs in which both reads were individually uniquely aligned.

Table 15  Example of Insert Statistics Table

<table>
<thead>
<tr>
<th></th>
<th>Too Small</th>
<th>Too Large</th>
<th>Orientation and Size OK</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3945 (0.2%)</td>
<td>1701 (0.1%)</td>
<td>2437627 (99.7%)</td>
</tr>
</tbody>
</table>

Cluster Intensity

Key web pages that illustrate cluster intensity are IVC.htm and All.htm.
ICV.htm

The IVC.htm file (Intensity versus Cycle) contains plots that display lane averages over all tiles in the lane. The plots displayed are All, Called, %Base_Calls, %All, and %Called.

- **All**—This is the lane average of the data displayed in All.htm. It plots each channel (A, C, G, T) separately as a different colored line. Means are calculated over all clusters, regardless of base calling. If all clusters are T, then channels A, C, and G will be zero. If all bases are present in the sample at 25% of total and a well-balanced matrix is used for analysis, the graph will display all channels with similar intensities. If intensities are not similar, the results could indicate either poor cross-talk correction or poor absolute intensity balance between each channel.

- **Called**—This plot is similar to All, except means are calculated for each channel using clusters that the base caller has called in that channel. If all bases are present in the sample at 25% with pure signal (zero intensity in the non-called channels), the Called intensity will be four times that of All, as the intensities will only be averaged over 25% of the clusters. For impure clusters, the called intensity will be less than four times that of All. The Called intensities are independent of base representation, so a well-balanced matrix will display all channels with similar intensities.

- **%Base_Calls**—The percentage of each base called as a function of cycle. Ideally, this should be constant for a genomic sample, reflecting the base representation of the sample. In practice, later cycles often show some bases more than others. As the signal decays, some bases may start to fall into the noise while other still rise above it. Matrix adjustments may help to optimize data.

- **%All** and **%Called**—Exactly the same as All and Called, but expressed as a percentage of the total intensities. These plots make it easier to see changes in relative intensities between channels as a function of cycle by removing any intensity decay.

For information on interpreting results in the IVC.htm file, see Interpretation of Run Quality on page 57.

All.htm

The All.htm file gives a tile-by-tile representation of the mean matrix-adjusted intensity of clusters plotted as a function of cycle. It plots each channel (A, C, G, T) separately as a different colored line. Means are calculated over all clusters, regardless of base calling.

If all clusters are T, channels A, C, and G will be at zero. If all bases are present in the sample at a rate of 25% and a well-balanced matrix is used for analysis, the graph will display all channels with similar intensities. If intensities are not similar, the results could indicate either poor cross-talk correction or poor absolute intensity balance among each channel.

A genome rich in GC content may not provide a balanced matrix for accurate cross-talk correction and absolute intensity balance.

Error Rates

For all analysis modes except sequence, Perfect.htm and Error.htm are produced, which measure sequence error rates.
**Perfect.htm**

The Perfect.htm graph shows the proportion of reads in a tile that have 0, 1, 2, 3, or 4 errors by the time they get to a given cycle.

Good data show a high proportion of reads with zero errors throughout the cycles.

**Error.htm**

The Error.htm file shows a graph of error rates for each tile on a flow cell. The red bar shows the percentage of bases at each cycle that are wrong, as calculated based on alignment to the reference sequence. Issues such as focus or fluidics problems manifest themselves as spikes in the graph.

For good data the error rate should be 1.5% or less at 36 cycles (aligned with eland_extended).

- PhageAlign allows any number of errors in an alignment and provides an accurate count of the error rate. However, it is too slow for aligning against target references larger than 2 Mb.
- ELANDv2 is capable of aligning against large genomes, such as human, in reasonable time. However, it allows only two errors per fragment. This means that error rates based on ELANDv2 alignments are underestimated. Very poor quality data has more than two errors in the first 32 aligned bases and is excluded from the calculations.

**Text-Based Analysis Results**

The output files for each lane of a flow cell are named using the format s_N_sequence.txt, where N represents a specific lane of the flow cell. For paired-read analysis, there are two parallel output files, one for each read. The files are named using the format s_N_R_sequence.txt, where N represents a specific lane of the flow cell and R represents the read number. The files are found in the GERALD folder of a finished analysis run.

The output files for each tile are named using the format s_N_TTTT_suffix.txt. For example, all files pertaining to tile 23 of lane 3 have names starting with s_3_0023.

The following table lists the files that contain the most meaningful data produced from your analysis run and the GERALD analysis mode that creates them. For descriptions of the GERALD analysis variables, see ANALYSIS VARIABLES on page 34.
### Table 16  Text-Based Analysis Results

<table>
<thead>
<tr>
<th>GERALD Analysis Mode</th>
<th>Output File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>All modes except ANALYSIS none</td>
<td>s_N_sequence.txt</td>
<td>This file contains all sequences in a single lane of a flow cell in an exportable format. The content of this file is affected by the following parameters: USE_BASES and SEQUENCE_FORMAT. For a description of each of these parameters, see ANALYSIS Variables on page 34.</td>
</tr>
<tr>
<td></td>
<td>s_N_rescore.txt</td>
<td>This file contains error rates for filtered data based on the alignments. These are used to create the graphs in the Error.htm pages. The file is located in the Stats sub-directory.</td>
</tr>
<tr>
<td></td>
<td>s_N_qcalreport.txt</td>
<td>This file reports the accuracy of the recalibrated quality values, making use of _ub_qseq.txt or _ub_custom_qseq.txt files as appropriate.</td>
</tr>
<tr>
<td>ANALYSIS eland_extended</td>
<td>s_N_export.txt</td>
<td>This file contains the results of alignment of all reads in the lane. The fields are tab separated to facilitate export to databases. This file has a line for every read, not just those that pass purity filtering. The last field on each line is a flag telling you whether or not the read passed the filter (Y or N). For file formats, see Final Output File Formats on page 55.</td>
</tr>
<tr>
<td>ANALYSIS default</td>
<td>s_N_export.txt</td>
<td>This file is in the same format as the eland_extended s_N_export.txt file. Note, however, that the score reported in the Single-Read Alignment Score is a phageAlign score and is not directly comparable to an eland_extended alignment score.</td>
</tr>
<tr>
<td>ANALYSIS eland_extended / default (deprecated)</td>
<td>s_N_sorted.txt</td>
<td>This output file is similar to s_N_export.txt, except it contains only entries for reads which pass purity filtering and have a unique alignment in the reference. These are sorted by order of their alignment position, which is meant to facilitate the extraction of ranges of reads for purposes of visualization or SNP calling.</td>
</tr>
<tr>
<td>ANALYSIS eland_pair</td>
<td>s_N_1_sequence.txt, s_N_2_sequence.txt</td>
<td>These parallel sets of files contain filtered sequences for each lane.</td>
</tr>
<tr>
<td></td>
<td>s_N_1_export.txt, s_N_2_export.txt</td>
<td>These parallel sets of files contain the results of alignment of all reads in the lane. The fields are tab separated to facilitate export to databases. Each file has a line for every read, not just those that pass purity filtering. The last field on each line is a flag telling you whether or not the read passed the filter (Y or N). For information on file format, see Final Output File Formats on page 55.</td>
</tr>
<tr>
<td></td>
<td>s_N_1_sorted.txt, s_N_2_sorted.txt (deprecated)</td>
<td>These parallel sets of files are similar to s_N_1_export.txt and s_N_2_export.txt, except they contain only entries for reads which pass purity filtering and have a unique alignment in the reference. These are sorted by order of their alignment position, which is meant to facilitate the extraction of ranges of reads for purposes of visualization or SNP calling.</td>
</tr>
<tr>
<td></td>
<td>s_N_anomaly.txt</td>
<td>This file contains one line for each read for which the two halves of the read did not align with a nominal distance and orientation from each other. This is the file to mine for structural variation information.</td>
</tr>
</tbody>
</table>
**Table 16 Text-Based Analysis Results (Continued)**

<table>
<thead>
<tr>
<th>GERALD Analysis Mode</th>
<th>Output File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANALYSIS eland_rna</td>
<td>s_N_export.txt</td>
<td>This file contains the results of alignment of all reads in the lane. The fields are tab separated to facilitate export to databases. This file has a line for every read, not just those that pass purity filtering. The last field on each line is a flag telling you whether or not the read passed the filter (Y or N). For file formats, see Final Output File Formats on page 55.</td>
</tr>
<tr>
<td></td>
<td>s_N_sorted.txt</td>
<td>This output file is similar to s_N_export.txt, except it contains only entries for reads which pass purity filtering and have a unique alignment in the reference. These are sorted by order of their alignment position, which is meant to facilitate the extraction of ranges of reads for purposes of visualization or SNP calling.</td>
</tr>
</tbody>
</table>

**Output File Formats**

The sequences and base-specific quality scores are bundled by lane and come in several configurable text formats. The currently supported formats are FASTA, fastq, and SCARF. For a description of each format, see ANALYSIS Variables on page 34.
Table 17  Final Output File Formats

<table>
<thead>
<tr>
<th>Output File</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>s_N_export.txt</td>
<td>Not all fields are relevant to a single-read analysis.</td>
</tr>
<tr>
<td>s_N_R_export.txt</td>
<td>1. Machine (Parsed from Run Folder name)</td>
</tr>
<tr>
<td>s_N_sorted.txt</td>
<td>2. Run Number (Parsed from Run Folder name)</td>
</tr>
<tr>
<td>s_N_R_sorted.txt</td>
<td>3. Lane</td>
</tr>
<tr>
<td></td>
<td>4. Tile</td>
</tr>
<tr>
<td></td>
<td>5. X Coordinate of cluster</td>
</tr>
<tr>
<td></td>
<td>6. Y Coordinate of cluster</td>
</tr>
<tr>
<td></td>
<td>7. Index value (0 for a non-indexed run)</td>
</tr>
<tr>
<td></td>
<td>8. Read number (1 or 2 for paired-read analysis, 0 for a single-read analysis)</td>
</tr>
<tr>
<td></td>
<td>9. Read</td>
</tr>
<tr>
<td></td>
<td>10. Quality string--In symbolic ASCII format (ASCII character code = quality value + 64)</td>
</tr>
<tr>
<td></td>
<td>11. Match chromosome--Name of chromosome match OR code indicating why no match resulted</td>
</tr>
<tr>
<td></td>
<td>12. Match Contig--Gives the contig name if there is a match and the match chromosome is split into contigs (Blank if no match found)</td>
</tr>
<tr>
<td></td>
<td>13. Match Position--Always with respect to forward strand, numbering starts at 1 (Blank if no match found)</td>
</tr>
<tr>
<td></td>
<td>14. Match Strand--&quot;F&quot; for forward, &quot;R&quot; for reverse (Blank if no match found)</td>
</tr>
<tr>
<td></td>
<td>15. Match Descriptor--Concise description of alignment (Blank if no match found)</td>
</tr>
<tr>
<td></td>
<td>• A numeral denotes a run of matching bases</td>
</tr>
<tr>
<td></td>
<td>• A letter denotes substitution of a nucleotide: For a 35 base read, &quot;35&quot; denotes an exact match and &quot;32C2&quot; denotes substitution of a &quot;C&quot; at the 33rd position</td>
</tr>
<tr>
<td></td>
<td>• The escape sequence &quot;^..$&quot; represents an indel. An integer in the indel escape sequence (e.g. &quot;10^2$18&quot;) indicates an insertion relative to reference of the specified size. A sequence in the indel escape sequence (e.g. &quot;10^AG$20&quot;) indicates a deletion relative to reference, with the sequence given the deleted reference sequence.</td>
</tr>
<tr>
<td></td>
<td>16. Single-Read Alignment Score--Alignment score of a single-read match, or for a paired read, alignment score of a read if it were treated as a single read. Blank if no match found; any scores less than 4 should be considered as aligned to a repeat. -1 for shadow reads.</td>
</tr>
<tr>
<td></td>
<td>17. Paired-Read Alignment Score--Alignment score of a paired read and its partner, taken as a pair. Blank if no match found; any scores less than 4 should be considered as aligned to a repeat. Note that in single-ended analyses it is always blank.</td>
</tr>
<tr>
<td></td>
<td>18. Partner Chromosome--Name of the chromosome if the read is paired and its partner aligns to another chromosome</td>
</tr>
<tr>
<td></td>
<td>19. Partner Contig</td>
</tr>
<tr>
<td></td>
<td>• Not blank if read is paired and its partner aligns to another chromosome and that partner is split into contigs.</td>
</tr>
<tr>
<td></td>
<td>• Blank for single-read analysis</td>
</tr>
<tr>
<td></td>
<td>20. Partner Offset</td>
</tr>
<tr>
<td></td>
<td>• If a partner of a paired read aligns to the same chromosome and contig, this number, added to the Match Position, gives the alignment position of the partner.</td>
</tr>
<tr>
<td></td>
<td>• If partner is a shadow read, this value is 0.</td>
</tr>
<tr>
<td></td>
<td>• If partner aligns to a different chromosome and contig, the number represents the absolute position of the partner.</td>
</tr>
<tr>
<td></td>
<td>• Blank for single-read analysis unless the record belongs to a part of a spliced RNA read.</td>
</tr>
<tr>
<td></td>
<td>21. Partner Strand--To which strand did the partner of the paired read align? &quot;F&quot; for forward, &quot;R&quot; for reverse (&quot;N&quot; if no match found, blank for single-read analysis)</td>
</tr>
<tr>
<td></td>
<td>22. Filtering--Did the read pass quality filtering? &quot;Y&quot; for yes, &quot;N&quot; for no</td>
</tr>
</tbody>
</table>
CHAPTER 3
Using GERALD for Sequence Alignment

**Table 17** Final Output File Formats (Continued)

<table>
<thead>
<tr>
<th>Output File</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>s_N_sequence.txt</td>
<td>Filtered output</td>
</tr>
<tr>
<td>s_N_R_sequence.txt</td>
<td>User-specified: FASTA, fastq, scarf (one sequence per line, not identifier)</td>
</tr>
<tr>
<td>s_N_rescore.txt</td>
<td>Various breakdowns of base mismatches within aligned reads (e.g. by cycle, called base and reference base), along with associated statistics. Tabular text format, header data included</td>
</tr>
</tbody>
</table>

**GERALD Output Files in Temp Folder**

For GERALD runs, various files are produced in Temp subfolder of the GERALD main analysis folder. Files that can be found in the Temp subfolder include:

- s_N_TTTT_align.txt
- s_N_TTTT_realign.txt
- s_N_TTTT_prealign.txt
- s_N_1_TTTT_ub_qseq.txt in Temp/Custom (deprecated)
- s_N_2_TTTT_ub_qseq.txt in Temp/Custom (deprecated)
- s_N_1_TTTT_ub_qseq.txt in Temp
- s_N_2_TTTT_ub_qseq.txt in Temp

**Intermediate Output Data Files**

Intermediate output files are found in the GERALD folder and contain data used to build the more meaningful results files described in *Pipeline Analysis Output* on page 43.

The files are named using one of the following formats:

- s_N_TTTT_name.txt, where N is the lane number, T is the tile number
- s_N_name.txt, where N is the lane number
- s_N_R_name.txt, where N is the lane number, R is the read number

**Table 18** Intermediate Output File Descriptions

<table>
<thead>
<tr>
<th>Output File</th>
<th>GERALD Analysis Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>s_N_eland_extended.txt</td>
<td>ANALYSIS eland_extended</td>
<td>Contains the corrected alignment positions and the full alignment descriptions for &gt;32 base reads. This file is not purity filtered.</td>
</tr>
<tr>
<td>s_N_R_eland_extended.txt</td>
<td>ANALYSIS eland_pair</td>
<td></td>
</tr>
<tr>
<td>s_N_calsaf.txt</td>
<td>ANALYSIS eland_extended</td>
<td>These are Calibrated Short Alignment Format files describing the selected alignment for each alignable read.</td>
</tr>
<tr>
<td>s_N_R_calsaf.txt</td>
<td>ANALYSIS eland_pair</td>
<td></td>
</tr>
<tr>
<td>s_N_qtable.txt</td>
<td>Any Analysis mode that includes alignment</td>
<td>These are produced as part of custom quality recalibration.</td>
</tr>
<tr>
<td>s_N_extended_contam.txt</td>
<td>ANALYSIS eland_rna</td>
<td>Alignments to the ELAND_RNA_GENOME_CONTAM and ELAND_RNA_GENOME_SPLICE respectively</td>
</tr>
<tr>
<td>s_N_extended_splice.txt</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Interpretation of Run Quality**

After the analysis of a run is complete, you need to interpret the data in the report summary and various graphical outputs. This section describes a standard, systematic way to examine your data.

The starting point is to know what a standard run of acceptable quality looks like. This is something of a moving target and is dependent on individual instruments, instrument configuration, genomic sample type, type of analysis, flow cell preparation, and the current state of the art. Therefore, the numbers shown in this section are for example only.

**Summary.htm**

The Summary.htm file is the first file you should review after your analysis is complete.

The following are examples of two of the tables found in Summary.htm, Lane Results Summary and Expanded Lane Summary, each truncated to a single lane of information. For a description of the tables found in Summary.htm, see Results Summary on page 46.

**Table 19 Intermediate Output File Formats**

<table>
<thead>
<tr>
<th>Output File</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>s_N_TTTT_align.txt</td>
<td>Deprecated sequence alignment format. Space-separated text values: 1. Sequence 2. Best score 3. Number of hits at that score The following columns only appear if hits equal 1 (a single, unique match) 4. Target:pos 5. Strand 6. Target sequence 7. Next best score</td>
</tr>
<tr>
<td>s_N_TTTT_realign.txt</td>
<td></td>
</tr>
<tr>
<td>s_N_TTTT_prealign.txt</td>
<td></td>
</tr>
</tbody>
</table>

**Table 20 Example of Lane Results Summary**

<table>
<thead>
<tr>
<th>Lane</th>
<th>Lane Yield (kbases)</th>
<th>Clusters (raw)</th>
<th>Clusters (PF)</th>
<th>1st Cycle Int (PF)</th>
<th>% intensity after 20 cycles (PF)</th>
<th>% PF Clusters</th>
<th>% Align (PF)</th>
<th>Alignment Score (PF)</th>
<th>% Error Rate (PF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>52456</td>
<td>63922 ± 8799</td>
<td>57518 ± 9226</td>
<td>175 ± 10</td>
<td>83.59 ± 1.61</td>
<td>90.16 ± 8.58</td>
<td>97.64 ± 1.44</td>
<td>389.39 ± 7.67</td>
<td>0.44 ± 0.17</td>
</tr>
</tbody>
</table>
**CHAPTER 3**

**Using GERALD for Sequence Alignment**

The key parameters that you should examine are listed in the following sections, along with conditions, possible causes for those conditions, and suggested actions to correct the condition.

### Clusters

This column contains the average number of clusters per tile detected in the first cycle images. For 1 Gbases of data at 35 cycles, this value needs to be greater than 20,000.

<table>
<thead>
<tr>
<th>Lane Info</th>
<th>Phasing Info</th>
<th>Raw Data</th>
<th>Filtered Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lane</td>
<td>Clusters</td>
<td>% Phasing</td>
<td>% Pre-phasing</td>
</tr>
<tr>
<td>1</td>
<td>63923</td>
<td>0.2200</td>
<td>0.2800</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Condition</th>
<th>Possible Cause</th>
<th>Suggested Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fewer clusters than expected:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Few bright clusters on the flow cell</td>
<td>Problem with cluster formation</td>
<td>Reanalyze with new default offsets in OLB.</td>
</tr>
<tr>
<td>Blurred images</td>
<td>Poor focus or dirty flow cell surface</td>
<td></td>
</tr>
<tr>
<td>Lots of clusters visible</td>
<td>Cluster density or size is too great to distinguish individual objects</td>
<td></td>
</tr>
</tbody>
</table>

| More clusters than expected: | |
| Too many clusters on the flow cell | Problem with cluster formation | |
| Very large clusters | Double counting | |

### Average First Cycle Intensity

Generally, brighter is better, but this result is instrument and sample dependent.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Possible Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low intensity</td>
<td>Problem with cluster formation or poor focus</td>
</tr>
</tbody>
</table>
Percentage of First Cycle Intensity Remaining After 20 Cycles of Sequencing

Generally, the higher, the better. Greater than 50% is acceptable, though it can be sample dependent.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Possible Cause</th>
<th>Suggested Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low value</td>
<td>A correct measure of rapid signal decay deduced from intensity plots</td>
<td>Check experiment fluidics or temperature control</td>
</tr>
<tr>
<td></td>
<td>Problem with cycle 20 deduced from intensity plots.</td>
<td>Check fluidics and focus for this cycle</td>
</tr>
<tr>
<td>Exceptionally high value</td>
<td>Low first cycle intensity</td>
<td>Check first cycle focus</td>
</tr>
</tbody>
</table>

Percentage of Clusters Passing Filters

To remove the least reliable data from the analysis, the raw data can be filtered to remove any clusters that have “too much” intensity corresponding to bases other than the called base. By default, the purity of the signal from each cluster is examined over the first 25 cycles and chastity = Highest_Intensity / (Highest_Intensity + Next_Highest_Intensity) is calculated for each cycle. The new default filtering implemented in at the base calling stage is that at most one cycle is less than the chastity threshold.

The higher the value, the better. Ideally, a value of > 70% is good, but this value is very dependent on cluster density. When there are more than 20,000 clusters per tile, the percentage starts to fall, since the major cause of an impure signal in the early cycles is the presence of another cluster within a few micrometers.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Possible Cause</th>
<th>Suggested Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very few clusters passing filter</td>
<td>Poor flow cell, perhaps unblocked DNA</td>
<td>Some of the causes may be at a single cycle. If the problem is isolated to these early cycles, it is possible that this filtering throws away very good data. Base calling errors may be limited to affected cycles, and, as early cycles are fairly resistant to minor focus and fluidics problems, even the number of errors may be few. The filtering can always be set manually to some other values. Check before assuming all the data are poor.</td>
</tr>
<tr>
<td></td>
<td>Faint clusters</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Out of focus</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Poor matrix</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A fluidics or sequencing failure</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bubbles in individual tiles</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Too many clusters</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Large clusters</td>
<td></td>
</tr>
<tr>
<td></td>
<td>High phasing or prephasing</td>
<td></td>
</tr>
</tbody>
</table>

Percentage of Clusters Passing Filters that Align Uniquely to the Reference Genome

Optimal value depends on the genome sequenced and the read-length; the higher (up to 100% max), the better. For example, for 30-mers and human genome, the optimum is less than 80%.

This result is genome specific and dependent on the completeness of the reference. A failure to align could be due to repeat or missing regions, or due to indels where sample and reference do not match.
**Percentage Error Rate of Clusters Passing Filters**

This value should be as low as possible, but it is very dependent on read-length. At 36 cycles, aligned with eland_extended, the error rate should be around 1.5%. Depending on the quality of the data, it will tend to rise at this point. If there is a sudden rise beyond cycle 32, then it is likely that ELANDv2 has effectively filtered out many clusters with more than two errors, thus suppressing the true error rate up to this point. The percentage aligning will also be low.

**Percentage of Phasing and Prephasing**

Ideally, these values should be as low as possible. Satisfactory results can be obtained with up to 1% for each. Preferably, they should be closer to 0.5%.

**Standard Deviations**

Many values have standard deviations associated with them. This can be the first indication as to the uniformity of the flow cell. If standard deviations are high, then it indicates variability from tile to tile with a lane.

---

**Condition** | **Possible Cause** | **Suggested Action**
---|---|---
Much lower than expected when using ELANDv2 | Fluidics or instrument problem | Look for an intensity dip in IVC plots. If there is a problem and it occurs after a sufficiently useful read-length, re-run ELANDv2 analysis using only the “good” cycles before the instrument problem.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Possible Cause</th>
<th>Suggested Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contamination from other genetic material resulting in an inability to align data</td>
<td>Align a few sample tiles with PhageAlign. Genomic contamination will show as early cycle error rates. If error rates remain fairly constant with cycle, then the “correct” genome has probably sequenced correctly. Non smooth error rate plots or IVC plots indicate the presence of specific tags or sequences.</td>
<td></td>
</tr>
</tbody>
</table>

**Condition** | **Possible Cause** | **Suggested Action**
---|---|---
High phasing or prephasing | Reagent issue (reagents have deteriorated) Fluidics | Check for leaks or bubbles in images or early cycle discrepancies in intensity plots. Poor flow cell | Poor blocking can be evident as intensity in all channels from cycle 1. |

<table>
<thead>
<tr>
<th>Condition</th>
<th>Possible Cause</th>
<th>Suggested Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>High standard deviations</td>
<td>Check poor tiles for: • Bubbles • Focus • Dirty flow cell surface</td>
<td>Look at the tile-by-tile statistics that appear below the flow cell-wide summary.</td>
</tr>
</tbody>
</table>

After reviewing the tables in Summary.htm, examine the thumbnails, and the output files IVC.htm, All.htm, and Error.htm.
**IVC.htm**

For a detailed description of the plots found in the IVC.htm file, see *IVC.htm* on page 51.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Possible Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intensity curves are not smooth</td>
<td>Cycle to cycle focus or fluidics problems</td>
</tr>
<tr>
<td>Called intensities are not equal</td>
<td>Poor fluidics or poorly blocked flow cell</td>
</tr>
<tr>
<td>(“% Called” may be +/- 5% out</td>
<td>If from cycle 1, initial matrix estimate may also</td>
</tr>
<tr>
<td>without major problems)</td>
<td>be in error</td>
</tr>
</tbody>
</table>

**All.htm and Error.htm**

The results in both files should show consistency from tile to tile down a lane and from lane to lane, if the results are from the same sample.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Possible Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tile variability</td>
<td>Bubbles</td>
</tr>
<tr>
<td></td>
<td>Rapid focus fluctuations</td>
</tr>
<tr>
<td></td>
<td>Dirty flow cell surface</td>
</tr>
<tr>
<td>Rising error rates</td>
<td>Low intensity at start</td>
</tr>
<tr>
<td>(Rates will always rise eventually at</td>
<td>High decay rate</td>
</tr>
<tr>
<td>high read-lengths)</td>
<td>High phasing or prephasing</td>
</tr>
<tr>
<td>High, but constant error rates from cycle 1</td>
<td>Genomic contamination</td>
</tr>
</tbody>
</table>
Chapter 4

Variant Detection and Counting

Topics

64 Introduction
73 Running Variant Detection and Counting
69 Estimating Build Depth
70 Variant Detection Input Files
73 Running Variant Detection and Counting
79 Variant Detection and Counting Output Files
79 Variant Detection and Counting Output Files
Introduction

This chapter explains how to use CASAVA v1.6 to detect Single Nucleotide Polymorphisms (SNPs) and indels, and count hits on transcripts for RNA sequencing. CASAVA generates a CASAVA build, which is a post-sequencing analysis of data from reads aligned to a reference genome by GERALD.

During the build process, CASAVA v1.6 collates, filters, and compiles aligned reads. CASAVA then calls the genomic consensus sequence using a Bayesian algorithm and compares it to the reference sequence in order to identify homozygous or heterozygous SNPs. CASAVA also identifies indels using the Indel Finder.

CASAVA automatically generates a range of statistics, such as mean depth and percentage chromosome coverage, to enable comparison with previous builds or other individuals. Moreover, CASAVA provides expression levels for exons, genes and splice junctions in the RNA Sequencing analysis.

Use Cases

The application has three basic use cases:

- DNA Sequencing for large genomes.
- DNA Sequencing for small genomes (data sets).
- RNA Sequencing.

All types of analysis take export.txt files from GERALD as input and produce a set of allele SNP calls and indels. In addition, RNA Sequencing analysis provides counts for exons, genes and splice junctions.

DNA Sequencing Analysis for Large Genomes

DNA Sequencing whole genome analysis can be used for large genomes and high coverage (like the human genome at 30x coverage), and both single-read and paired-end runs. CASAVA can take the large numbers of aligned single-read or paired-end sequences from multiple experiments, arrange them into a genome build, and describe differences from the reference sequence.

For big data sets (30x coverage human genome), the process can take between 5 hours and several days, depending on available infrastructure.

NOTE

Large projects like human genome resequencing require high-performance computer clusters; see Hardware and Software Requirements on page 88.

DNA Sequencing Analysis for Small Genome

DNA Sequencing for small genomes, such as whole genome sequencing of bacteria or targeted resequencing, is very similar to DNA Sequencing for large genomes with the only difference being that it may process data from one lane or less. Thus a single computer is enough to make the build.
RNA Sequencing Analysis

RNA Sequencing analysis supports whole transcriptome sequencing projects. In addition to allele calls there are a few more data types produced. Exon counts, splice junction counts and gene counts can be used to determine gene expression levels and expressed splice variants.

**NOTE**

RNA Sequencing only supports single-read runs.

CASAVA Workflow

The CASAVA workflow for variant detection and counting is illustrated in Figure 10, below.
**Expected Performance**

The expected run times for a number of typical CASAVA projects are listed in Table 22 on page 66:

Table 22  **Expected Performance for Typical CASAVA Projects**

<table>
<thead>
<tr>
<th>Application</th>
<th>High coverage human genome (30X)</th>
<th>Standard coverage human genome</th>
<th>High coverage <em>E. coli</em> genome (100X)</th>
<th>Human brain transcriptome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequencing Data</td>
<td>DNA Sequencing</td>
<td>DNA Sequencing</td>
<td>DNA Sequencing</td>
<td>RNA Sequencing</td>
</tr>
<tr>
<td>Sequencing Data</td>
<td>90 Gb</td>
<td>3 Gb</td>
<td>500 Mb</td>
<td>10 Gb</td>
</tr>
<tr>
<td>Processors</td>
<td>100 processors Sun Grid Engine</td>
<td>16 processor machine</td>
<td>1 processor</td>
<td>100 processors Sun Grid Engine</td>
</tr>
<tr>
<td>Total run time</td>
<td>~ 5 hours</td>
<td>~ 4.5 to 8 hours</td>
<td>~ 40 minutes</td>
<td>~ 20 minutes</td>
</tr>
</tbody>
</table>
Methods

CASAVA uses a number of methods to efficiently build the consensus, call SNPs, detect indels, and provide counts. This section explains the methods.

Duplicate Removal

Duplicate removal is performed only for paired-end runs.

Duplicate reads are removed by examining all paired reads for their 5'-most and 3'-most positions (including the insert). Any pairs that share exactly the same positions are unlikely to have occurred by chance (for sufficiently large genomes) and so it is assumed that these are duplicates caused by sample preparation and only the highest quality read-pair is kept.

Final Set of Reads

The final set of reads contains the reads matching any of the reference sequence. All reads with no match or matching ribosomal DNA (for RNA sequencing) are stored in NM and RM directories respectively.

The reads are divided into sorted.txt files by chromosome and furthermore into regions, which are 50 Mb long in the default setting. Each file contains reads matching to a given region.

For paired-end runs, files contain reads from normal pairs, anomalous pairs and orphan reads, but only reads from normal pairs are used for allele calling and SNP calling.

Allele Calling

The allele caller produces one or two allele calls and scores for each position in the genome. Only read pairs that mapped with the expected insert size (within 3 standard deviations of the median) and orientation are used for allele calling. Paired reads should have a paired read alignment score of at least 6 and single reads have a single read alignment score of at least 10 in order to be used for allele calling. The allele caller then computes an allele call score, which can be approximately translated as a Phred score divided by 10 (e.g. an allele score of 3 corresponds to a Phred score of 30).

For a detailed description, see Allele Calling on page 146

SNP Calling

Homzygous SNPs are called at positions where a non-reference allele is observed, the allele call score is ≥10, and the depth is no greater than three times the chromosomal mean. For heterozygous SNP calls, we additionally require the second-highest scoring allele to have an allele-call score ≥6 and the ratio of the highest to second-highest allele-call scores to be ≤3. The allele call score cutoff ensures that more than the equivalent of three q30 bases are needed to make a SNP call; the ratio cut-off helps to distinguish between genuine heterozygous SNPs and any residual background noise, especially for extremely high coverage (e.g. mitochondria in the human genome).

For a detailed description, see SNP Calling on page 147

Indel Finding

The Indel Finder application uses singleton/ shadow read pairs to detect indels. The indel detection works in four stages:
1. Compute clusterings of non-aligned 'shadow reads', using as a distance metric the positions of the 'singleton' reads that they pair to.

2. Assemble them into contigs.

3. Align the contigs back to the genome, using the positions of associated ‘singleton’ reads to narrow the search to a couple of thousand bp or so.

4. After candidate indels have been identified, assign a genotype to each candidate. Use both the reads which were assembled into contigs by the Indel Finder and those reads from the reference chromosome assembly which intersect the candidate indel’s position on the reference sequence, and call the candidate indel as homozygous, heterozygous or not present.

For a detailed description, see IndelFinder on page 147.

CASAVA Counting Methods

CASAVA employs two different counting methods for RNA sequencing reads. The readBases counting method is normalized better, more precise, and accounts for spliced reads bases, and is preferred over the readStart counting method, which is kept for validation and backward compatibility.

readBases Counting Method

As of version 1.6, CASAVA uses the readBases counting method. This method is for exon, splice junction and gene counts, and counts the number of bases that belong to each feature. Both reads that map to the genome and reads that map to splice junctions contribute to base coverage value. For splice junctions, the number of reads that cover the junction point is counted. The number of bases that fall into the exonic regions of each gene is summed to obtain gene level counts, and normalized according to feature size, and expressed as RPKM (Reads Per Kilotbase per Million of mapped reads).

Exons that have overlapping exons from other genes on the forward or reverse strand are excluded from counting and are also not included to compute the total gene length.

readStart Counting Method

This method is for exons, splice junctions and genes counts, and is kept unchanged from CASAVA v1.0 for validation and backward compatibility.

By default CASAVA will only count the first base of each read. The number of reads that fall into the exonic regions of each gene plus reads that map to splice junctions of the gene is summed to obtain gene level counts. These counts are then divided by the length of the gene to obtain the length-normalized gene counts. Exons that have overlapping exons from other genes on the forward or reverse stand are excluded from counting and are also not included to compute the total gene length.

For a detailed description, see RNA Sequencing Counting Methods on page 155.
Estimating Build Depth

Estimating the build depth enables you to keep track of your progress with large resequencing projects. To estimate the depth for paired 75 base reads in a short insert human project:

1. Obtain the yield of purity-filtered data from the Chip Results Summary (column: Yield) or Lane Results Summary (column: Lane Yield). These can be found in the Summary.htm file or Gerald Summary Reports link of the run_analyses.html file.

2. For paired-end runs assume 80% will uniquely align, 20% will not. Subtract 20% from the yield.

3. CASAVA will automatically remove PCR duplicates in paired-end DNA sequencing projects. To allow for this, subtract a further 10% from the yield.

4. Divide the remaining yield of PF data (in bases) by the genome size (in bases) to estimate the sequence depth.

You may want to consider the following adjustments:

- For resequencing projects on other genomes, adjust the percentage that will align (step 2) according to the results summary. The more your sample deviates from the reference genome, the lower the percentage aligned reads will be.
- A larger fragment size may give slightly more duplicates, while a longer read length will contain fewer. Adjust the percentage in step 3 accordingly.
Variant Detection Input Files

The variant detection and counting input files come from the GERALD module using the following eland modules:

- Eland_extended (page 39) for single-read DNA sequencing projects.
- Eland_pair (page 40) for paired-end DNA sequencing projects.
- Eland_rna (page 43) for single-read RNA sequencing projects (paired-end RNA sequencing projects are not supported).

The GERALD input files for CASAVA variant detection can be found in the GERALD directory of the run folder, and are described below.

In addition, CASAVA variant detection and counting uses reference files (genome sequence files, a genome size file, and an exon coordinates set). The reference files for human genome build 36, mouse genome build 37, and rat genome build 3.4 are provided. If you want to use a different genome or build, you can generate reference files as described in Appendix E, Reference Files CASAVA.

See Figure 11 for file locations.

![Image of file structure](image-url)

**Figure 11** CASAVA Input Files

**Export.txt Files**

The export.txt files contain the aligned sequences information from the GERALD module. The export.txt files are tab delimited text files; for a detailed description, see Output File Formats on page 54.
Run.conf.xml

The run.conf.xml file provides the location of the GERALD data (export.txt) for each flow cell run, and describes properties of each flow cell run. There is one run section for each flow cell run, one set section for each GERALD folder in each flow cell run, and one lane section for each lane in each set. The run.conf.xml file can be provided (created) by the user or CASAVA will generate it automatically based on command line options. run.conf.xml file should be placed in projectDirectory/conf/.

Example:
<opt><run id="080328_EAS114_0051_FC3011HAAXX"> <set setid="1" gerald="/Human/080328_EAS139_0029_FC3010UAAXX_R1/Data/Intensities/Basecalls/GERALD_09-04-2008_aladdin/" readMode="paired"> <lanes>1</lanes> <lanes>2</lanes> ... list of lanes </set> </run></opt>

NOTE
There should only be one flow cell within each <run>.

For multiple runs, see Configuring Multiple Runs on page 77.

Pair.xml

The pair.xml file provides information about pair distribution in the GERALD output (only for paired-end sequencing).

Config.xml

The GERALD config.xml file specifies what analysis GERALD was instructed to do for each lane, and contains run specifications from the recipe and the number of folders and image, intensity, and base call files.

Summary.htm

The analysis software produces a Summary.htm file for each Run Folder, which contains comprehensive results and performance measures of the analysis run. It is located in the GERALD folder and provides an overview of quality metrics for a run with links to more detailed information in the form of pages of graphs.

CASAVA can also use the Summary.xml file instead of the Summary.htm file.

For a detailed description of the Summary.htm and Summary.xml files, see Results Summary on page 44.

NOTE
The Summary.htm file is optional if CASAVA is run with the analysis parameter requireSummary = 0 in the file conf/global.conf.
CASAVA Genome Files

Variant detection and counting in CASAVA uses the same genome sequence files as GERALD (see ELAND Genome Files on page 126). The only difference is that the files should not be squashed.

Genome Size File

Variant detection and counting in CASAVA uses a genome size file that the analysis software generates, named `<species>_genomes_size.xml`. This file contains the names and sizes of the chromosomes, in XML format:

```
<SequenceSizes>
  <c2.fa>242751449</c2.fa>
  <cX.fa>154913754</cX.fa>
  ...
  <c21.fa>46944323</c21.fa>
</SequenceSizes>
```

Exon Coordinates Set

CASAVA uses a non-redundant exon coordinates set to compute gene expression counts during RNA sequencing analysis. **Exon coordinates for CASAVA and splice junction for alignments need to be from the same version of transcriptome.** For every gene, overlapping exons from the same gene are combined to form one larger region used for counting. Regions where there are overlapping exons from other genes are excluded from counting and are not present in the non-redundant exon set file.

The non-redundant exon coordinates set is generated by the `nonoverlapping_exon_coords.pl` from publicly available genome data (see Non-Redundant Exon Set on page 135). The file looks like this:

```
c21  46440018  46440098 LSS
c21  46879955  46880103 PRMT2
c21  40472724  40472891 DSCAM
```

**NOTE**

To obtain correct counts, the exon coordinates set and the junction file for the alignment must be from the same reference set of transcripts.
Running Variant Detection and Counting

**Usage**

To run CASAVA, move to the CASAVA_1.6/bin directory and enter the following in the command line:

```
./run.pl [options]
```

Set the options to define the type of analysis you want to perform for each lane. The options are listed in the next section.

**NOTE**

For RNA Sequencing, instead of `./run.pl [options]` you can use `./runRNA.pl [options]`. This implies the options `-a RNA` and `-rm SE`.

**Options**

The options that define CASAVA analysis are listed in the tables below (with SE = single end (single read), PE = paired end).

The options genomeSize and projectDir are mandatory for all analysis types. CASAVA will not run if one of these options is missing.

CASAVA will only run without exportDir, lanes, or runId if the run.conf.xml file is already setup.

CASAVA assumes that the reference sequence file organism.fa exists in the projectDir/geneomes directory or that the --refSequences parameter is provided.

**Table 23  Major File Options for Variant Detection and Counting**

<table>
<thead>
<tr>
<th>Option</th>
<th>Application</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-e --exportDir=PATH</td>
<td>SE, PE</td>
<td>Path to source directory, known also as GERALD directory or run directory.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: <code>-e TestData/GERALD</code></td>
</tr>
<tr>
<td>-g --genomeSize=PATH</td>
<td>SE, PE</td>
<td>Full PATH to xml file with chromosome/genome sizes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: <code>-g ./conf/human_genomes_size.xml</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The genomeSize file can be found in the GERALD directory</td>
</tr>
<tr>
<td>-l --lanes=NUMBER_LIST</td>
<td>SE, PE</td>
<td>List of the lanes to use.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: <code>-l 1,2,3,4</code></td>
</tr>
<tr>
<td>-p --projectDir=DIR</td>
<td>SE, PE</td>
<td>Project directory (where CASAVA keeps all intermediate files).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: <code>-p /home/user_name/data/Project_01</code></td>
</tr>
<tr>
<td>-r --runId=STRING</td>
<td>SE, PE</td>
<td>Unique identifier for each run.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: <code>-r 080408_EAS192_0027_FC20DU9AAXX</code></td>
</tr>
<tr>
<td>-ref --refSequences=PATH</td>
<td>SE, PE</td>
<td>PATH of the reference genome sequences. Default is projectDir/organisms/.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: <code>-ref /data/Genome/CASAVA/hg18</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The FASTA files should not be squashed for CASAVA.</td>
</tr>
</tbody>
</table>
Table 24  Behavioral Options for Variant Detection and Counting

<table>
<thead>
<tr>
<th>Option</th>
<th>Application</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-a</td>
<td>SE, PE</td>
<td>Type of analysis [DNA, RNA]; default is DNA. Example: <code>-a RNA</code></td>
</tr>
<tr>
<td>--applicationType=TYPE</td>
<td>SE, PE</td>
<td>Post Run Commands can be launched after CASAVA completes by including the <code>--postRunCmd</code> option, followed by the commands to be launched</td>
</tr>
<tr>
<td>--postRunCmd</td>
<td>SE, PE</td>
<td>Post Run Commands can be launched after CASAVA completes by including the <code>--postRunCmd</code> option, followed by the commands to be launched</td>
</tr>
<tr>
<td>-f</td>
<td>SE, PE</td>
<td>Ignore errors from previous CASAVA execution. Example: <code>-f</code></td>
</tr>
<tr>
<td>--force</td>
<td>SE, PE</td>
<td>Ignore errors from previous CASAVA execution. Example: <code>-f</code></td>
</tr>
<tr>
<td>-h</td>
<td>SE, PE</td>
<td>Prints on screen usage guide. If TARGET is specified, prints usage guide for the corresponding plugin target. Example: <code>-h=indels</code></td>
</tr>
<tr>
<td>--help [=TARGET]</td>
<td>SE, PE</td>
<td>Prints on screen usage guide. If TARGET is specified, prints usage guide for the corresponding plugin target. Example: <code>-h=indels</code></td>
</tr>
<tr>
<td>-rt</td>
<td>SE, PE</td>
<td>Removes all temporary data, running as soon as it is possible. Default is ON. Example: <code>-rt OFF</code></td>
</tr>
<tr>
<td>--removeTemps=ON/OFF</td>
<td>SE, PE</td>
<td>Removes all temporary data, running as soon as it is possible. Default is ON. Example: <code>-rt OFF</code></td>
</tr>
<tr>
<td>-t</td>
<td>SE, PE</td>
<td>Space-separated list of targets to run (see Table 26 on page 76). Default is all. Example: <code>-t configure</code></td>
</tr>
<tr>
<td>--targets=LIST</td>
<td>SE, PE</td>
<td>Space-separated list of targets to run (see Table 26 on page 76). Default is all. Example: <code>-t configure</code></td>
</tr>
<tr>
<td>-w</td>
<td>SE, PE</td>
<td>Instead of running CASAVA, generates the workflow definition file tasks-DATA.txt. Example: <code>-w</code></td>
</tr>
<tr>
<td>--workflow</td>
<td>SE, PE</td>
<td>Instead of running CASAVA, generates the workflow definition file tasks-DATA.txt. Example: <code>-w</code></td>
</tr>
<tr>
<td>--workflowAuto</td>
<td>SE, PE</td>
<td>Generates the workflow definition file and runs it. See <code>--jobsLimit</code>. Example: <code>--workflowAuto</code></td>
</tr>
<tr>
<td>--sgeAuto</td>
<td>SE, PE</td>
<td>Generates the workflow definition file and runs it on SGE (use with <code>--sgeQueue</code>)</td>
</tr>
<tr>
<td>--sgeQueue</td>
<td>SE, PE</td>
<td>Generates the workflow definition file and runs it on SGE (use with <code>--sgeQueue</code>)</td>
</tr>
<tr>
<td>--jobsLimit</td>
<td>SE, PE</td>
<td>Limit number of parallel jobs. Defaults: -1 (unlimited) for <code>--sgeAuto</code>. 1 for <code>--workflowAuto</code>. Do not set it to the maximum number of processors as this might cause the terminal to become unresponsive</td>
</tr>
<tr>
<td>--jobsLimit</td>
<td>SE, PE</td>
<td>Limit number of parallel jobs. Defaults: -1 (unlimited) for <code>--sgeAuto</code>. 1 for <code>--workflowAuto</code>. Do not set it to the maximum number of processors as this might cause the terminal to become unresponsive</td>
</tr>
<tr>
<td>--jobsLimit</td>
<td>SE, PE</td>
<td>Limit number of parallel jobs. Defaults: -1 (unlimited) for <code>--sgeAuto</code>. 1 for <code>--workflowAuto</code>. Do not set it to the maximum number of processors as this might cause the terminal to become unresponsive</td>
</tr>
<tr>
<td>--sgeQueue</td>
<td>SE, PE</td>
<td>SGE queue name, used with <code>--sgeAuto</code> or <code>--workflow</code> (e.g: all.q)</td>
</tr>
<tr>
<td>--verbose=NUMBER</td>
<td>SE, PE</td>
<td>Sets the verbose level (default is 0, which is the minimum). Example: <code>--verbose=1</code></td>
</tr>
<tr>
<td>--sgeAuto</td>
<td>SE, PE</td>
<td>Sets the verbose level (default is 0, which is the minimum). Example: <code>--verbose=1</code></td>
</tr>
<tr>
<td>--sgeQueue</td>
<td>SE, PE</td>
<td>SGE queue name, used with <code>--sgeAuto</code> or <code>--workflow</code> (e.g: all.q)</td>
</tr>
<tr>
<td>--version</td>
<td>SE, PE</td>
<td>Prints version information. Example: <code>--version</code></td>
</tr>
<tr>
<td>Option</td>
<td>Application</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>--spliceJunction=NAME</code></td>
<td>SE</td>
<td>Name of splice junction set relative to <code>--genesListPath</code>. Default human/splice_sites-49.fa. Example for 35 bp alignment: <code>--spliceJunction=splice_sites-34</code></td>
</tr>
<tr>
<td><code>--featureFileName=FILE</code></td>
<td>SE</td>
<td>FILE name of exons definition file relative to <code>--genesListPath</code>. Default human/exonCoords.txt. Example: <code>--featureFileName=exon_coords.txt</code></td>
</tr>
<tr>
<td><code>--genesListPath=PATH</code></td>
<td>SE</td>
<td>PATH of the reference sequences for genes. Default /opt/GOAT/CASAVA_xxxxx/share/CASAVA-xxxxx/features. Example: <code>--genesListPath=./TestData/TestProject/genes/</code></td>
</tr>
<tr>
<td><code>--snpThreshold=NUMBER</code></td>
<td>SE, PE</td>
<td>Sets the SNP caller threshold to NUMBER. Default is 10. This is the minimum allele call score required to call a SNP. See also SNP Calling on page 67. Example: <code>--snpThreshold=20</code></td>
</tr>
<tr>
<td><code>--snpThreshold2=NUMBER</code></td>
<td>SE, PE</td>
<td>Sets the SNP caller threshold for the second-highest scoring allele to NUMBER (for heterozygous SNPs). Default is 6. See also SNP Calling on page 67. Example: <code>--snpThreshold2=10</code></td>
</tr>
<tr>
<td><code>--snpMaxRatio=NUMBER</code></td>
<td>SE, PE</td>
<td>Sets the SNPCaller max ratio to NUMBER. This is used to evaluate possible heterozygous SNPs. This sets the maximum ratio between the first and second allele call scores. Situations where the first allele is much stronger than the minor allele should be called as homozygous SNPs, as the minor allele may simply be noise. Default is 3. See also SNP Calling on page 67. Example: <code>--snpMaxRatio=5</code></td>
</tr>
<tr>
<td><code>--snpCovCutoff=NUMBER</code></td>
<td>SE, PE</td>
<td>Sets the SNPCaller coverage cutoff to NUMBER (default 3: SNPs are called only at the positions where the depth is no greater than three times the chromosomal mean). This prevents SNP calling in regions with extreme depth, such as near the centromere of a human chromosome. <code>--snpCovCutoff=-1</code> turns off the filter (recommended for RNA).</td>
</tr>
<tr>
<td><code>--readMode=MODE</code></td>
<td>SE, PE</td>
<td>Run-read-mode for all runs, paired for paired end (default), single for single end (single read). To turn the filter off, for example for RNA or targeted resequencing, set it to -1. Example: <code>--readMode=single</code></td>
</tr>
<tr>
<td><code>--QVCutoff=NUMBER</code></td>
<td>PE</td>
<td>Sets the alleleCaller QVCutoff to NUMBER (default 6). See also SNP Calling on page 67. Example: <code>--QVCutoff=6</code></td>
</tr>
<tr>
<td><code>--QVCutoffSingle=NUMBER</code></td>
<td>SE, PE</td>
<td>In single-ended mode <code>QVCutoffSingle</code> (default 10) is used. <code>QVCutoffSingle</code> also affects RNA feature counting. Reads below this value (<code>QVCutoffSingle</code> as RNA is single-ended only) are not contributing to exon and gene counts. Example: <code>--QVCutoffSingle=10</code></td>
</tr>
<tr>
<td><code>--singleScoreForPE=VALUE</code></td>
<td>PE</td>
<td>Sets the alleleCaller to filter reads with single score below <code>QVCutoffSingle</code> in PE mode YESINO. Default NO. Example: <code>--singleScoreForPE=YES</code></td>
</tr>
</tbody>
</table>
CHAPTER 4
Variant Detection and Counting

**Table 25**  Analysis Options for Variant Detection and Counting

<table>
<thead>
<tr>
<th>Option</th>
<th>Application</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--baseQualityCutOff NUMBER</td>
<td>SE, PE</td>
<td>Sets minimum base quality in allele caller. Default 0.</td>
</tr>
<tr>
<td>--qualityType=VALUE</td>
<td>SE, PE</td>
<td>Quality to probability scheme Phred64</td>
</tr>
<tr>
<td>--toNMScore=NUMBER</td>
<td>SE, PE</td>
<td>Minimum SE alignment score to put a read to NM. Default=-1 (-1 means option is turned off)</td>
</tr>
<tr>
<td>--rmDup=YES</td>
<td>NO</td>
<td>PE</td>
</tr>
<tr>
<td>--dataSetSuffix=STRING</td>
<td>SE, PE</td>
<td>IndelFinder.pl, alleleCaller and SVFinder output file suffix. Default <strong>(empty)</strong></td>
</tr>
</tbody>
</table>

**Targets**

The targets that define CASAVA analysis are listed in the table below. The targets cov2bin, rnaRead, poisson, indels, and alignability are plugin targets, and have a dedicated help page with additional information. This help page can be accessed by typing:

```
Path/to/CASAVA/bin/run.pl --help=TARGET
```

**Table 26**  Targets for Variant Detection and Counting

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>Build targets that are pre-configured for the given analysis type.</td>
</tr>
<tr>
<td>configure</td>
<td>Configure the build (should be run when starting new build).</td>
</tr>
<tr>
<td>listRuns</td>
<td>Lists all runs in the configuration.</td>
</tr>
<tr>
<td>removeRun</td>
<td>Removes run from the configuration.</td>
</tr>
<tr>
<td>export</td>
<td>Loads the export files to export directory.</td>
</tr>
<tr>
<td>duplicates</td>
<td>Removes duplicate reads from export directory.</td>
</tr>
<tr>
<td>sort</td>
<td>Sorts reads and moves them to bins in the build directory (Parsed_...).</td>
</tr>
<tr>
<td>allele</td>
<td>Allele calling in the build directory.</td>
</tr>
<tr>
<td>snp</td>
<td>Finds SNP and exports them to GFF format.</td>
</tr>
<tr>
<td>exp2sra</td>
<td>Converts all export files from GERALD folders to a zipped fastq files and generates sra.xml.</td>
</tr>
<tr>
<td>clean</td>
<td>Soft clean allows to restart the build without removing project.conf and run.conf.xml.</td>
</tr>
<tr>
<td>allClean</td>
<td>Removes all data.</td>
</tr>
<tr>
<td>snpClean</td>
<td>Removes snp data.</td>
</tr>
<tr>
<td>sortClean</td>
<td>Removes sort data.</td>
</tr>
<tr>
<td>duplicatesClean</td>
<td>Removes duplicates data.</td>
</tr>
</tbody>
</table>
### Configuring Multiple Runs

There are two methods to add multiple runs:

- **Command line method.**
  
  Run the command as in "Configuring One Run", but then list all runs, export directories, and lanes, which would look like this:
  ```
  ./run.pl -p projectDir -t configure
  -r RunId1 -e exportDir1 -l laneList1
  -r RunId2 -e exportDir2 -l laneList2 ...
  -r RunIdn -e exportDirn -l laneListn
  ```
  
  Where RunId, exportDir, and laneList are respectively the run id, path to the export file from the run, and comma separated list of lanes from the run.

- **Xml file edition method.**
  
  Instead of typing multiple run ids and paths on the command line you can modify the run configuration file (run.conf.xml):
  
  a. Go to /Human/conf/run.conf.xml (see Run.conf.xml on page 71).
  
  b. Add appropriate entries to the run.conf.xml file.
  
  c. Then run the configuration again by executing:
  ```
  ./run.pl -p /Human/ -t configure
  ```

### Table 26 Targets for Variant Detection and Counting

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>alleleClean</td>
<td>Removes allele data.</td>
</tr>
<tr>
<td>cov2bin</td>
<td>Converts the sort.count files into coverage.bin files. This format is required by Resembl genome viewer.</td>
</tr>
<tr>
<td>rnaRead</td>
<td>Calculates the exon, gene and splice junction expression. The sort target must have been completed or be part of the current workflow</td>
</tr>
<tr>
<td>poisson</td>
<td>Plots the poisson statistics of the CASAVA build. The alignability and allele targets must have been completed or be part of the current workflow in order for the poisson to work. Poisson plots are not available for RNA builds</td>
</tr>
<tr>
<td>indels</td>
<td>Finds indels in a CASAVA build. The sort target must have been completed or be part of the current workflow in order for the indels to work</td>
</tr>
<tr>
<td>alignability</td>
<td>Calculates the genome alignability. Additionally it provides helper API for other workflow scripts to be able to determine the paths to alignability files. The configure target must have been completed or be part of the current workflow in order for the alignability to know the reference genome files list</td>
</tr>
</tbody>
</table>

Required standard options: refSequences, genomeSize.
Examples

The CASAVA installation provides examples of common use cases, such as:

- *E. coli* Single End
- *E. coli* Paired End
- RNA sequencing

The details of these examples are available on the `run.pl` help page. Go to the CASAVA installation directory, and type:

```
./run.pl
```

The examples are listed at the bottom of the help page.

Results Directory

Once the run is complete, all relevant information is listed in the results directory, such as:

- Build web page.
  The build web page is located in the html folder, and provides access to run information and graphs of important statistics.
- CASAVA build.
  The CASAVA build contains sequence, SNP, indels, and (for RNA Sequencing) counts information, and is located in `projectDir/Parsed_DATE`.
- Computer readable statistics.
  Computer readable statistics are located in `projectDir/stats`.
- Configuration files.
  CASAVA configuration files are located in `projectDir/conf`.

For a detailed description of the output files, see *Variant Detection and Counting Output Files* on page 79.
Variant Detection and Counting Output Files

The variant detection and counting output files contain run information, statistical analysis, sequence tags, SNP and indel information, and (for RNA Sequencing) gene counts, exon counts, and splice junction counts. They are described below.

Build Directory

An outline of the CASAVA build directory is shown below in Figure 12.

```
TestProject  project directory
    conf    configuration directory
    html    directory with html reports see html/Home.html
    stats   directory with text reports
    Parsed-xx-xx-xx current build directory (final files are here)
        NMNM  non-mapping reads
        chr1.fa build chromosome directory
            00nn chromosome bin directory
                sort.count file with allele calls for each position
                sorted.txt file with sorted sequence reads
            Chromosome.snp.txt file with SNPs
            indels.txt file with indels
            Chromosome exon_count.txt file with exon counts (RNA Sequencing only)
            Chromosome splice_count.txt file with splice counts (RNA Sequencing only)
            Chromosome gene_count.txt file with gene counts (RNA Sequencing only)
        indels.txt file with indels
```

Figure 12  CASAVA Build Directory

The most important folders for downstream analysis are listed below.

- **Html Folder**
  
The html folder contains the build web page (see Build Web Page on page 80), which provides access to run information and graphs of important statistics.

- **Parsed_xx_xx_xx folder**
  
The Parsed_xx_xx_xx folder contains most of the sequencing information, such as sequence tags, SNP information, and (for RNA Sequencing) gene counts, exon counts, and splice junction counts (see CASAVA Build on page 81). This information is organized in chromosome folders named chr1 or chr2, for example.

- **Stats Folder**
The stats folder contains statistical information in computer readable form, such as the runs_summary.xml file, which shows which lanes from which run were aggregated and called for a CASAVA build.

Conf Folder
The conf folder contains information about the configuration of the project, such as the project.conf file.

Build Web Page
The build web page is located in projectDir/html. When you open the file home.html, you will find a list of all runs, and a link to statistics (Figure 13).

If you click on a particular run, you will see the summary file for that particular run (Figure 14; for a description, see Results Summary on page 44).

The statistics link on the build web page (Figure 13) will lead you to graphs for important statistics (Figure 15, Figure 16):

- Reads per chromosome
- Duplicates statistics
- SNPs statistics
- Indel statistics
- Coverage/Depth statistics
Variant Detection and Counting Output Files

CASAVA Build

The CASAVA build, containing sequence, SNP, indels, and (for RNA Sequencing) counts information, is located in the projectDir/Parsed_xx_xx_xx folder.

Sorted.txt Files

The sorted.txt file is a tab delimited text file that looks a lot like the export.txt file. The sorted.txt contains the aligned sequence information coming from CASAVA v1.6, but is sorted by the position field. The sorted.txt file is stored in bins (default size 50 Mb) in the chromosome-specific directory under the Parsed-dd-mm-yy directory. The sorted.txt has the following features:

- The first 10 columns match the first 10 columns of the qseq files.
- The filter flag, in the last column, is a Y/N flag as opposed to the 0/1 flag in the qseq files.
- The "Partner" fields are empty for single-read analysis.
- Not all fields are relevant to a single-read analysis.

The sorted.txt contains the following columns:
1. Machine (Parsed from Run Folder name)
2. Run Number (Parsed from Run Folder name)
3. Lane
4. Tile
5. X Coordinate of cluster
6. Y Coordinate of cluster
7. Index value (0 for a non-indexed run)
8. Read number (1 or 2 for paired-read analysis, 0 for a single-read analysis)
9. Read
10. Quality string--In symbolic ASCII format (ASCII character code = quality value + 64)
11. Match chromosome--Name of chromosome match OR code indicating why no match resulted
12. Match Contig--Gives the contig name if there is a match and the match chromosome is split into contigs (Blank if no match found)
13. Match Position--Always with respect to forward strand, numbering starts at 1 (Blank if no match found)
14. Match Strand--"F" for forward, "R" for reverse (Blank if no match found)
15. Match Descriptor--Concise description of alignment (Blank if no match found)
   • A numeral denotes a run of matching bases
   • A letter denotes substitution of a nucleotide: For a 35 base read, "35" denotes an exact match and "32C2" denotes substitution of a "C" at the 33rd position
   • The escape sequence "^..$" represents an indel. An integer in the indel escape sequence (e.g. "10^2$18") indicates an insertion relative to reference of the specified size. A sequence in the indel escape sequence (e.g. "10^AG$20") indicates a deletion relative to reference, with the sequence given the deleted reference sequence.
16. Single-Read Alignment Score--Alignment score of a single-read match, or for a paired read, alignment score of a read if it were treated as a single read. Blank if no match found; any scores less than 4 should be considered as aligned to a repeat. -1 for shadow reads.
17. Paired-Read Alignment Score--Alignment score of a paired read and its partner, taken as a pair. Blank if no match found; any scores less than 4 should be considered as aligned to a repeat. Note that in single-ended analyses it is always blank.
18. Partner Chromosome--Name of the chromosome if the read is paired and its partner aligns to another chromosome
19. Partner Contig
   • Not blank if read is paired and its partner aligns to another chromosome and that partner is split into contigs.
• Blank for single-read analysis

20. Partner Offset
• If a partner of a paired read aligns to the same chromosome and contig, this number, added to the Match Position, gives the alignment position of the partner.
• If partner is a shadow read, this value is 0.
• If partner aligns to a different chromosome and contig, the number represents the absolute position of the partner.
• Blank for single-read analysis unless the record belongs to a part of a spliced RNA read.

21. Partner Strand--To which strand did the partner of the paired read align? "F" for forward, "R" for reverse ("N" if no match found, blank for single-read analysis)

22. Filtering--Did the read pass quality filtering? "Y" for yes, "N" for no

Snp.txt Files
The snp.txt file is a tab delimited text file that contains the SNP calls sorted by position. There is one snp.txt file for each chromosome (for example, chr19.snp.txt), stored in the chromosome-specific directory under the Parsed-dd-mm-yy directory. The snp.txt file contains the following fields:

1. Position—The position of the SNP on the indicated chromosome.
2. A—Number of A bases called on the reads.
3. C—Number of C bases called on the reads.
4. G—Number of G bases called on the reads.
5. T—Number of T bases called on the reads.
6. Modified_call—The genotype called. In any heterozygous call the allele with the highest score is printed first.
7. Total—Total bases called at that position.
8. Used—Bases used for making the SNP call.
9. Score—The score of the first allele, followed by the score of the second allele, if applicable.
10. Reference—The reference base at that position.
11. Type—The call type:
   • SNP_diff—homozygous SNP.
   • SNP_het1—heterozygous SNP where the reference allele has the stronger of the two allele scores
   • SNP_het2—heterozygous SNP where the non-reference allele has the stronger of the two allele scores
   • SNP_het_other—heterozygous SNP where neither allele matches the reference

An example of a c*.snp.txt file opened in Excel is shown in Figure 17.
CHAPTER 4  
Variant Detection and Counting

Figure 17  Chromosome.snp.txt File Opened in Excel

Indels.txt Files

Indels for each chromosome are summarized within each chromosome directory in a file called indels.txt. This file contains indels which have been called in each chromosomal bin segment using the Indel Finder and indel genotype caller, and filtered to remove those indels which are found at a depth greater than a multiple of the mean chromosomal depth (3 times the mean chromosomal depth is used by default). This filter is designed to remove indel calls in regions close to centromeres and other high depth regions likely to generate spurious calls.

The indels.txt file contains a block of comment lines beginning with the ‘#’ character, followed by data lines consisting of 16 tab-delimited fields. The fields are described in the table below (all information is given with respect to the forward strand of the reference sequence).

<table>
<thead>
<tr>
<th>No</th>
<th>Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>pos</td>
<td>1-indexed start position of the indel event.</td>
</tr>
<tr>
<td>2</td>
<td>CIGAR</td>
<td>String summarizing the indel in CIGAR format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The CIGAR format describes an indel using the characters D, I and M to refer to the length of deletion, insertion, and match or mismatch regions in an alignment. Thus the value “2D” refers to a 2 base deletion, and the value “3I4M1D” refers to a 3 base insertion followed by 4 aligned bases, followed by a single base deletion.</td>
</tr>
<tr>
<td>3</td>
<td>ref_upstream</td>
<td>Segment of the reference sequence 5’ of the indel event.</td>
</tr>
<tr>
<td>4</td>
<td>ref/indel</td>
<td>Equal length sequences corresponding to the reference and indel cases which span the indel event. The character ‘-’ indicates a gap in the reference or indel sequence.</td>
</tr>
<tr>
<td>5</td>
<td>ref_downstream</td>
<td>Segment of the reference sequence 3’ of the indel event.</td>
</tr>
<tr>
<td>6</td>
<td>Q(indel)</td>
<td>Phred scaled quality score of the indel. This refers to the Phred-scaled probability that no indel (homozygous or heterozygous) exists at the given position.</td>
</tr>
<tr>
<td>7</td>
<td>max_gtype</td>
<td>Most probable indel genotype {het,hom,ref}. See below for a definition of het, hom, ref.</td>
</tr>
<tr>
<td>8</td>
<td>Q(max_gtype)</td>
<td>Phred scaled quality score of the most probable indel genotype. This refers to the Phred-scaled probability that the indel genotype is not that given as “max_gtype”.</td>
</tr>
<tr>
<td>9</td>
<td>max2_gtype</td>
<td>Second most probable indel genotype {het,hom,ref}. See below for a definition of het, hom, ref.</td>
</tr>
<tr>
<td>10</td>
<td>bp1_reads</td>
<td>Number of reads crossing the 5’-most breakpoint of the indel</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<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>
<th>K</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>#position</td>
<td>A</td>
<td>C</td>
<td>G</td>
<td>T</td>
<td>modified_call</td>
<td>total</td>
<td>used</td>
<td>score</td>
<td>reference</td>
<td>type</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>18250646</td>
<td>0</td>
<td>34</td>
<td>0</td>
<td>34</td>
<td>C</td>
<td>36</td>
<td>34</td>
<td>112.79</td>
<td>T</td>
<td>SNP_diff</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>18251869</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>13</td>
<td>C</td>
<td>36</td>
<td>13</td>
<td>41.19</td>
<td>T</td>
<td>SNP_diff</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>18252422</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>26</td>
<td>TG</td>
<td>40</td>
<td>35</td>
<td>82.78:28.20</td>
<td>T</td>
<td>SNP_het1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>18252476</td>
<td>12</td>
<td>0</td>
<td>37</td>
<td>T</td>
<td>A</td>
<td>54</td>
<td>49</td>
<td>104.98:36.99</td>
<td>T</td>
<td>SNP_het1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>18252564</td>
<td>0</td>
<td>31</td>
<td>0</td>
<td>16</td>
<td>CT</td>
<td>50</td>
<td>47</td>
<td>94.33:53.20</td>
<td>C</td>
<td>SNP_het1</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>18253563</td>
<td>0</td>
<td>0</td>
<td>36</td>
<td>0</td>
<td>G</td>
<td>42</td>
<td>36</td>
<td>128.39</td>
<td>C</td>
<td>SNP_diff</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>18254404</td>
<td>0</td>
<td>28</td>
<td>0</td>
<td>30</td>
<td>C</td>
<td>30</td>
<td>28</td>
<td>83.48</td>
<td>T</td>
<td>SNP_diff</td>
<td></td>
</tr>
</tbody>
</table>

Catalog # SY-960-1601  
Part # 15009919 Rev. A
<table>
<thead>
<tr>
<th>No</th>
<th>Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>ref_reads</td>
<td>Number of reads strongly supporting the reference case. See below for a definition of strong.</td>
</tr>
<tr>
<td>12</td>
<td>indel_reads</td>
<td>Number of reads strongly supporting the indel case. See below for a definition of strong.</td>
</tr>
<tr>
<td>13</td>
<td>other_reads</td>
<td>Number of reads intersecting the indel, but not strongly supporting either the reference or the indel. See below for a definition of strong.</td>
</tr>
<tr>
<td>14</td>
<td>repeat_unit</td>
<td>The smallest repetitive sequence unit within the inserted or deleted sequence for simple insertions or deletions. For complex indels (comprised of multiple insertions and/or deletions) this field is set to the value ‘N/A’.</td>
</tr>
<tr>
<td>15</td>
<td>ref-repeat_count</td>
<td>Number of times the repeat_unit sequence is contiguously repeated starting from the indel call site in the reference case.</td>
</tr>
<tr>
<td>16</td>
<td>indel-repeat_count</td>
<td>Number of times the repeat_unit sequence is contiguously repeated starting from the indel call site in the indel case.</td>
</tr>
</tbody>
</table>

Definitions of concepts used in the indels.txt file:

- Phred-scale Q-values are derived from the corresponding probability $P$ by the relationship $Q = -10\log_{10}(P)$. The Q-values given only reflect those error conditions which can be represented in the indel calling model, which is not comprehensive.
- The genotype “hom” refers to a homozygous indel, “het” refers to a heterozygous indel and “ref” refers to no indel at this position. The “ref” state will only be observed in the indel output file as the second most probable genotype, were this state found to be the most probable genotype the candidate indel would not be reported.
- For a read to strongly support either the reference or the indel alignment, it must overlap an indel breakpoint by at least 6 bases and the probability of the read's alignment following either the reference or the indel path must be at least 0.999.

Count.txt Files

There are three different types of _count.txt files, for exon, gene, or splice junction:

- **Chromosome_exon_count.txt.** The _exon_count.txt provides counts for the number of times a particular exon has been detected in a sample.
- **Chromosome_genes_count.txt.** The _genes_count.txt provides counts for the number of times a particular gene has been detected in a sample.
- **Chromosome_splice_count.txt.** The _splice_count.txt provides counts for the number of reads that align over a particular splice junction

_count.txt files are generated by RNA Sequecing, sorted by position, and there is one of each type per chromosome (for example, chr19_exon_count.txt). The _count.txt files are stored in the chromosome-specific directory under the Parsed-dd-mm-yy directory, and contain the following columns:
1. Chromosome—starting with a c. The chromosome on which the exon resides. ChrM indicates a mitochondrial DNA alignment.

2. Start—The start of the gene.

3. End—The end of the gene.


5. Normalized counts.
   - For readStart method: Normalized count = Raw count / feature length.
   - For readBases method: Normalized count (RPKM) = (raw count x read length)/(feature length x number of mapped reads in millions)

6. Raw counts.
   - For readStart method: raw count = number of reads starting within the feature.
   - For readBases method: raw count = sum of coverages for each base within the feature.

For readStart method exon counts include only reads that map to the genome within the exon. Gene count is sum of all exon counts and all junction counts for the gene.

For readBases method exon counts are sum of base coverages from genomic and spliced reads. Therefore gene counts are sum of exon counts. And junction counts (in reads) are provided for historical reasons and for alternative splicing analysis.

An example of a chromosome_genes_count.txt file opened in Excel is shown in Figure 17. This shows an example using the readStart method. For the readBases method, raw counts would be approximately a factor read length higher.

![Figure 18 Chromosome_genes_count.txt File Opened in Excel]
Appendix A

Requirements and Software Installation for CASAVA

Topics

88 Introduction
88 Hardware and Software Requirements
  88 Disk Space Requirements
  88 Memory Requirements
  88 Analysis Computer
  89 Software Requirements
88 Hardware and Software Requirements
90 Setting Up Email Reporting
Introduction

This section describes the CASAVA v1.6 system requirements and the software installation instructions.

Hardware and Software Requirements

Disk Space Requirements

When running CASAVA without keeping temporary data (removeTemps=ON):
- Disk space needed while running = 3 x size of export files
- Disk space needed after running = 1.5 x size of export file

When running with all temporary files saved (removeTemps=Off):
- Disk space while running = 5 x size of export files

For example: to generate a build from one lane of E. coli data (1 GB with removeTemps=ON), we recommend an additional 3 GB of disc space while running CASAVA and ~1.5 GB for the final build directory.

Memory Requirements

CASAVA requires a minimum of 2GB RAM per core for a 50G run. The parameter ELAND_SET_SIZE in the GERALD config.txt specifies the maximum number of tiles aligned by each ELAND process. The default value is 40 which should keep the peak memory consumption below 2GB for a 50G run.

Analysis Computer

CASAVA may run on any 64-bit Unix variant, if all of the prerequisites described in this section are met. However, Illumina does not support any platform other than Linux.

Illumina recommends the IlluminaCompute Tier 1 data processing solution (for more information, contact Illumina). IlluminaCompute Tier 1 satisfies all the requirements listed below, but other systems satisfying these requirements are also fully supported.
- HP ProLiant DL580 G5 Rack Server
  - This system comes configured with Red Hat Linux and the full installation of the GERALD Software v1.6.
- Four quad-core 2.93GHz 64-bit Intel Xeon processors.
- 32 GB fault-tolerant RAM.
This is enough RAM to perform analysis tasks and file server tasks simultaneously. It uses high speed fault-tolerant hard drives for the operating system and applications.

- HP Modular Smart Array 20 (12 x 750 GB SATA 7,200 rpm drives, 9 TB total).

This capacity is intended to hold information from three runs, as follows:

- Last Processed Run—The results data from the last analyzed run are copied off to another storage server, where the run can be reviewed by the investigators and their staff. The raw image data is deleted.
- Currently Processed Run—The raw image data from the last completed instrument run are loaded and the GERALD is performing analysis on that run.
- Next Run for Processing—The Genome Analyzer is copying the raw data from the current run up to the server.

As data volumes increase, the storage capacity can be scaled up by adding additional MSA20 units.

Sequence alignment takes somewhere between a few hours (using our fast short-read whole-genome alignment program ELAND) and days (using more traditional alignment programs).

CASAVA parallelization is built around the multi-processor facilities of the “make” utility and scales very well to beyond eight nodes. Substantial speed increases are expected for parallelization across several hundred CPUs. For a detailed description, see Using Parallelization in GERALD on page 109.

The required software environment is described below:

- CASAVA installation may not work properly with gcc versions 3.x. If you have a gcc version 3.x, install gcc 4.0.0 or newer, with the exception of gcc version 4.0.2, which is not supported.
- Installation of CASAVA v1.6 now requires the Boost C++ library, version 1.39.0 and cmake version 2.6.4 and above. These packages are included in the CASAVA installation package, and will automatically install during the configure stage if either package is not found in the user’s environment.
Installing CASAVA

Starting with CASAVA v1.6, CASAVA must be built outside of the source directory. The installation procedure is as follows:

1. tar xvzf CASAVA_v1.6.0.tar.bz2
2. mkdir CASAVA-1.6.0-build
3. cd CASAVA-1.6.0-build
4. ../CASAVA-1.6.0/configure --prefix=(CASAVA installation directory)
5. make
6. make install

For more information on the configuration options:
   ../CASAVA-1.6.0/configure --help

For more information on the installation procedure, see the file INSTALL.

Setting Up Email Reporting

The script Gerald/runReport.pl is called at the end of a run and sends you an email when a run successfully completes.

To use email notification, set up an SMTP server and set the following parameters in the GERALD configuration file. For additional information, see GERALD Configuration File on page 31.

1. Enter a space-separated list of the email addresses that should receive the run completion notification.
   EMAIL_LIST your.name@domain.com that.name@domain.com

2. Indicate the path to the GERALD folder. The software assumes it can create a valid URL from the GERALD folder path by omitting a number of leading path elements as specified by NUM_LEADING_DIRS_TO_STRIP (by default two) and prepending WEB_DIR_ROOT.
   WEB_DIR_ROOT http://server/SHARE
   For example, if the path is /mnt/yourDrive/folder/folder/GERALD and WEB_DIR_ROOT is http://server/SHARE, the software will write the links as http://server/SHARE/folder/GERALD/File.htm.

3. Identify your domain. Your SMTP server may refuse to accept emails from or send emails to addresses that do not end in @yourdomain.com.
   EMAIL_DOMAIN yourdomain.com

4. Identify your IP address.
   EMAIL_SERVER yourserver:2525
   where yourserver is the name or IP address of a mail server that will accept SMTP email requests from you and 2525 is the port number of the SMTP service on that server.
Generally this will be 25. This is the default value if no port number is specified. The utility nmap, if installed, may help you identify which port on a server is hosting an SMTP service.

5. Test your email reporting by entering the following from the machine where you are running GERALD:

```
telnet yourserver yourPortNumber
```

If you don’t get a friendly message, then email reporting will not work. You can run runReport.pl directly in test mode by entering:

```
/runReport.pl --test yourserver:25 yourdomain.com anything your.name@yourdomain.com
```

You should receive a test email. If you do not, the transcript it generates should identify the problem.

![NOTE]

The optional email reporting feature depends on how your SMTP servers are set up locally. Email reporting is not required to run the GERALD to a successful completion.
Appendix B
Understanding the Run Folder

Topics

<table>
<thead>
<tr>
<th>Page</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>94</td>
<td>Introduction</td>
</tr>
<tr>
<td>97</td>
<td>Run Folder Structure</td>
</tr>
<tr>
<td>100</td>
<td>Demultiplexing Run Folder</td>
</tr>
<tr>
<td>102</td>
<td>Folder and File Naming</td>
</tr>
<tr>
<td>104</td>
<td>Configuration/Parameters</td>
</tr>
</tbody>
</table>
Introduction

This section describes the Run Folder, where the images and analysis output files are saved by default in a consistent hierarchical structure. A run folder containing RTA data is very similar to a Run Folder containing Pipeline analysis data.

Figure 19 illustrates a typical Run Folder after SCS image analysis and base calling, and CASAVA alignment.
Figure 20 illustrates a typical Run Folder after OLB base calling and CASAVA alignment.

The standardized structure, file naming conventions, and file formats of the Run Folder allow for the following:
- A single point of data storage, logging, and analysis output during and after a run.
Encoding sufficient information to trace the history of the data in the Run Folder back to the laboratory notebook without confusion between instruments, experiments, or sites.

Standardized input and output enabling component software to operate flawlessly, regardless of the instrument generating the data.

Capturing and encoding enough information to independently reanalyze the data at any time, in such a way that existing extractions of sequence and related data are preserved, and parameters used during any point of the extraction process are captured and related to the subsequent output data.

Subsequent analyses to be stored in the Run Folder.

The software tools and other user software to implement and enforce these structures and standards.
Run Folder Structure

The Run Folder contains the Images folder and Data folder as illustrated in Figure 19 and Figure 20 above.

- The Data folder contains Image Analysis folders and the Image Analysis folders contain BaseCalls folders which contain Sequence folders. The Data folder is created by the Genome Analyzer when a run starts. Any analysis performed on the data, including RTA or OLB analysis, is saved within the Data folder.
- The Images folder holds the images from every tile for all cycles of sequencing. The Images folder will not be present if only analysis data, not the images, are copied to the analysis server after RTA analysis. There is an option to send images to a second networked run folder apart from the main/default network destination.

Each run of the main analysis modules creates a subdirectory in the Data folder of the Run Folder as follows (see Figure 19 and Figure 20 above):

- Each run of RTA image analysis or the OLB image analysis software (Firecrest) creates a new image analysis output folder in the Data folder.
- Each run of RTA base calling or the OLB base calling software creates a new subdirectory in the image analysis subdirectory on which the base calls are based, resulting in a tree-like structure of analyses.
- Parameters and versions for any given analysis run are logged in the folder structure to make it possible to reconstruct any previous analysis run.

You can do multiple analyses of the data using different analysis parameters and the results will not be overwritten. The default naming convention for folders consists of the number of cycles run, the version of the software used for the operation, the date the analysis initiated, and the login of the user. If the user initiates a second analysis on the same day, a new folder structure is created and the results from the previous analysis are not overwritten.

Images Folder

The Images folder contains a subfolder for each lane that has been sequenced. The folders are named using the following convention where the lane number is padded to three digits:

L<lane number>

For example, L001 contains the images taken in the first lane.

Each lane folder contains a subfolder for each cycle of sequencing. Each image-cycle subfolder contains four images for every tile, one for each of the four bases.

The Image folder naming follows the naming convention C<cycle number>.<iteration number>. Cycle number is indexed and represents the nth cycle.

Within each image-cycle subfolder are four tif files for each tile. These files are named using the following convention:

<s各样e>_L<lane>_<tile>_<base>.tif

In the example, s_1_67_g.tif, the “s” is the default sample-ID.
Data Folder

The Data folder contains a hierarchical structure that consists of the image analysis output folder, then the base calling output folder, and then the sequence alignment output folder.

A new subfolder is generated each time a set of images is processed by the OLB image analysis module (Firecrest), or RTA. The data are kept in one file per tile for raw intensities and one file per tile for cluster noise. Firecrest uses the extension _int.txt and use the extension _nse.txt. RTA reports image analysis results in the binary .cif format (intensities) and .cnf format (noise).

The Data folder contains one config.xml file in each image analysis folder generated as a result of analyzing sets of images. The config.xml file explicitly records which cycle-image folders were used to generate the raw intensities and noise files, and any parameters used. For a detailed description of the parameters file, see Configuration/Parameters on page 104.

Image Analysis Folders

The image analysis folders have the following naming structure:

- The image analysis folder generated by RTA is called Intensities.
- Each image analysis folder generated by OLB is named using the following convention:
  
  C<first cycle>-<last-cycle>_<analysis module>_<version>_<date>_<user>

For example, C1-27_Firecrest1.8.20_31-07-2006_myuser.2 contains the second version of an analysis of cycles 1–27 performed using version 1.8.20 of the Firecrest analysis module, run by the user “myuser” on the 31st of July 2006.

Base Calling Folders

Each image analysis folder may hold multiple sequence folders with the output of different runs of a base caller package. The base calling folders have the following naming structure:

- The base calling folder generated by RTA is called Basecalls.
- Each base calling folder generated by OLB is named using the following convention:
  
  <analysis module>_<version>_<date>_<user>[.<version-number>]

For example, the folder name Bustard1.8.8_08-11-2005_myuser.3 represents the third run of the OLB Bustard base caller on 8th of November 2005 by the user “myuser.”

Each base calling folder also holds a config.xml that records any relevant information about the run of the base caller module.
Main Sequence Files

The main output files after base calling are the _qseq files. They have the following format:

- **Machine name**: (hopefully) unique identifier of the sequencer.
- **Run number**: (hopefully) unique number to identify the run on the sequencer.
- **Lane number**: positive integer (currently 1-8).
- **Tile number**: positive integer.
- **X**: Xcoordinate of the spot. As of RTA v1.6, OLB v1.6, and CASAVA v1.6, the X and Y coordinates for each clusters are calculated in a way that makes sure the combination will be unique. The new coordinates are the old coordinates times 10, +1000, and then rounded.
- **Y**: Ycoordinate of the spot. As of RTA v1.6, OLB v1.6, and CASAVA v1.6, the X and Y coordinates for each clusters are calculated in a way that makes sure the combination will be unique. The new coordinates are the old coordinates times 10, +1000, and then rounded.
- **Index**: positive integer. No indexing should have a value of 0.
- **Read Number**: 1 for single reads; 1 or 2 for paired ends or multiplexed single reads; 1, 2, or 3 for multiplexed paired ends.
- **Sequence**
- **Quality**: the calibrated quality string.
- **Filter**: Did the read pass filtering? 0 - No, 1 - Yes.

**NOTE**
Since Pipeline release 1.3, the quality scoring scheme is the Phred scoring scheme, encoded as an ASCII character by adding 64 to the Phred value. A Phred score of a base is:

\[ Q_{\text{phred}} = -10 \log_{10}(e) \]

where \( e \) is the estimated probability of a base being wrong.
Demultiplexing Run Folder

The demultiplexing output directories have the following characteristics:

- As many base calls directories as the maximum number of indices in each lane.
- The directory names are simply a three digit number (left padded with 0) starting from 1.
- The association of indices to specific directories is arbitrary.
- The demultiplexed output directories for each sample are specified in the SamplesDirectories.csv file in the Demultiplexed directory.
- The Unknown directory contains the reads with an unresolved or erroneous index.
- Each base calls directory is a valid base calls directory that can be used for subsequent alignment analysis in CASAVA.
- The index inserted in column 7 of the qseq files is by default the raw index.

**NOTE**

If the majority of reads end up in the 'unknown' folder, check the --qseq-mask parameter syntax and the length of the index in the sample sheet. The --qseq-mask option should be set to the length of the index in the sample sheet + the character 'n' to account for phasing.

**SamplesDirectories.csv File**

The SamplesDirectories.csv contains the sample information from the sample sheet, and has an additional column (Directory) that specifies the numbered demultiplexed output directory for each sample (Figure 21). The SamplesDirectories.csv file is located in the Demultiplexed directory.

When opening the file in Excel, make sure to use Excel’s Import Wizard (Data > Get External Data > From Text). Tell Excel to treat the last column (Directory) as "text" and leave the rest as "general". This will make sure the Directory numbers are displayed correctly (starting with 00).

![Figure 21: SamplesDirectories.csv File Opened in Excel](image)

The Run Folder changes Figure 22 illustrates a the changes to typical Run Folder after demultiplexing and before alignment.
Figure 22  Demultiplexing Run Folder Changes
Folder and File Naming

Run Folder Naming

The top level Run Folder name is generated using three fields to identify the <ExperimentName>, separated by underscores. For example, YYMMDD_machinename_NNNN. You should not deviate from the Run Folder naming convention, as this may cause the software to stop.

1. The first field is a six-digit number specifying the date of the run. The YYMMDD ordering ensures that a numerical sort of Run Folders places the names in chronological order.

2. The second field specifies the name of the sequencing machine. It may consist of any combination of upper or lower case letters, digits, or hyphens, but may not contain any other characters (especially not an underscore). It is assumed that the sequencing instrument is synonymous with the PC controlling it, and that the names assigned to the instruments are unique across the sequencing facility.

3. The third field is a four-digit counter specifying the experiment ID on that instrument. Each instrument should be capable of supplying a series of consecutively numbered experiment IDs (incremental unique index) from the onboard sample tracking database or a LIMS.

**NOTE**

It is desirable to keep Experiment-IDs (or Sample-ID) and instrument names unique within any given enterprise. You should establish a convention under which each machine is able to allocate Run Folder names independently of other machines to avoid naming conflicts.

A Run Folder named 070108_instrument1_0147 indicates experiment number 147, run on instrument 1, on the 8th of Jan 2007. While the date and instrument name specify a unique Run Folder for any number of instruments, the addition of an experiment ID ensures both uniqueness and the ability to relate the contents of the Run Folder back to a laboratory notebook or LIMS.

Additional information is captured in the Run Folder name in fields separated by an underscore from the first three fields. For example, you may want to capture the flow cell number in the Run Folder name as follows: YYMMDD_machinename_XXXX_FCYYYY.

**NOTE**

When publishing the data to a public database, it is desirable to extend the exclusivity globally, for instance by prefixing each machine with the identity of the sequencing center.

File Naming

The analysis software uses the following format for file naming:

```
<sample>_<lane>_[<tile>][<cycle>][<id>][type].<filesuffix>
```

Some files are split on a read basis, leading to the file naming:

```
<sample>_<lane>_[read]_[<tile>][<cycle>][<id>][type].<filesuffix>
```
When a given file type is split on a read basis, the read always appears in the name, even for single-read analysis.

**Table 27  File Naming Components**

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;sample&gt;</td>
<td>Alphanumeric string (always “s”)</td>
</tr>
<tr>
<td>&lt;lane&gt;</td>
<td>Single-digit number identifying a flow cell lane</td>
</tr>
<tr>
<td>&lt;read&gt;</td>
<td>Single-digit number identifying the read (starts at 1)</td>
</tr>
<tr>
<td>&lt;tile&gt;</td>
<td>Four-digit number identifying a tile location in a flow cell lane</td>
</tr>
<tr>
<td>&lt;cycle&gt;</td>
<td>Two- or three-digit number identifying a sequencing cycle</td>
</tr>
<tr>
<td>&lt;id&gt;</td>
<td>Single-digit number to distinguish files; for example, the different reads of a paired-end read</td>
</tr>
<tr>
<td>&lt;type&gt;</td>
<td>Alphabetical string identifying the type of content stored in the file</td>
</tr>
<tr>
<td>&lt;filesuffix&gt;</td>
<td>Suffix to identify the traditional file type</td>
</tr>
</tbody>
</table>

Example: s_5_1_0030_qseq.txt is a valid filename.
Exception: for image (.tif) files, the <tile> location can have less than four digits.
Configuration/Parameters

The Data Folder and subfolders, and the top level Image folder can all contain a configuration file (config.xml), and the top level Run Folder a related .params file. This is intended to contain any parameter data specific to the given level of information held in the folder.

Params File

The top level Run Folder contains a parameters file, named <Run FolderName>.params, and is written in the following format:

```xml
<experiment>
  <run>
    ...
  </run>
  ...
</run>
</experiment>
```

For each restart of the instrument, a new run tag with corresponding parameter tags is added to the parameters file. For most experiments, there will only be one run.

The XML tags in the parameters file are self-explanatory. The following shows an example of a parameters file:

```xml
<experiment>
  <run>
    <instrument>slxa-bl</instrument>
  </run>
</experiment>
```

Config.xml Files

In the top level of the Data folder you will find the config.xml file that records any information specific to the generation of the subfolders. This contains a tag-value list describing the cycle-image folders used to generate each folder of intensity and sequence files.

```xml
<?xml version="1.0"?>
<ImageAnalysis>
  <Run Name="Intensities">
    <Cycles First="1" Last="24" Number="24" />
    <ImageParameters>
      <AutoOffsetFlag>1</AutoOffsetFlag>
      <AutoSizeFlag>0</AutoSizeFlag>
      <DataOffsetFile>/data/070813_ILMN-1_0217_FC1234/Data/default_offsets.txt</DataOffsetFile>
      <Fwhm>2.700000</Fwhm>
      <InstrumentOffsetFile>/data/070813_ILMN-1_0217_FC1234/Data/default_offsets.txt</InstrumentOffsetFile>
      <Offsets X="0.000000" Y="0.000000" />
      <Offsets X="0.790000" Y="-0.550000" />
      <Offsets X="-0.240000" Y="-0.140000" />
      <Offsets X="0.190000" Y="0.650000" />
    </ImageParameters>
  </Run>
</ImageAnalysis>
```
<RemappingDistance>1.500000</RemappingDistance>
<SizeFile></SizeFile>
<Threshold>4.000000</Threshold>
</ImageParameters>
<RunParameters>
<AutoCycleFlag>0</AutoCycleFlag>
<BasecallFlag>1</BasecallFlag>
<Compression>gzip</Compression>
<CompressionSuffix>.gz</CompressionSuffix>
<Deblocked>0</Deblocked>
<DebugFlag>0</DebugFlag>
<ImagingReads Index="1">
  <FirstCycle>1</FirstCycle>
  <LastCycle>24</LastCycle>
  <RunFolder>/data/070813_ILMN-1_0217_FC1234</RunFolder>
</ImagingReads>
<Instrument>ILMN-1</Instrument>
<MakeFlag>1</MakeFlag>
<MaxCycle>-1</MaxCycle>
<MinCycle>-1</MinCycle>
<Reads Index="1">
  <FirstCycle>1</FirstCycle>
  <LastCycle>24</LastCycle>
  <RunFolder>/data/070813_ILMN-1_0217_FC1234</RunFolder>
</Reads>
</RunFolder>
<Software Name="RTA" Version="1.6.2.0" />
<TileSelection>
  <Lane Index="8">
    <Sample>s</Sample>
    <Tile>10</Tile>
    <Tile>20</Tile>
    <Tile>30</Tile>
  </Lane>
</TileSelection>
<Time>
  <Start>30-07-07 12:50:45 BST</Start>
</Time>
<User Name="user" />
</Run>
<Run Name="Intensities">
  ...
</Run>
</ImageAnalysis>

In each image analysis folder there is another config.xml file containing the meta-information about the base caller runs.
<Run Name="BaseCalls">
  <BaseCallParameters>
    <Matrix Path="">
      <AutoFlag>1</AutoFlag>
      <AutoLane>0</AutoLane>
      <Cycle>2</Cycle>
      <FirstCycle>1</FirstCycle>
      <LastCycle>24</LastCycle>
      <Read>1</Read>
    </Matrix>
    <MatrixElements />
    <Phasing Path="">
      <AutoFlag>1</AutoFlag>
      <AutoLane>0</AutoLane>
      <Cycle>1</Cycle>
      <FirstCycle>1</FirstCycle>
      <LastCycle>24</LastCycle>
      <Read>1</Read>
    </Phasing>
    <PhasingRestarts />
  </BaseCallParameters>
  <Cycles First="1" Last="24" Number="24" />
  <Input Path="Basecalls" />
  <RunParameters>
    <AutoCycleFlag>0</AutoCycleFlag>
    <BasecallFlag>1</BasecallFlag>
    <Compression>gzip</Compression>
    <CompressionSuffix>.gz</CompressionSuffix>
    <Deblocked>0</Deblocked>
    <DebugFlag>0</DebugFlag>
    <ImagingReads Index="1">
      <FirstCycle>1</FirstCycle>
      <LastCycle>24</LastCycle>
      <RunFolder>/data/070813_ILMN-1_0217_FC1234</RunFolder>
    </ImagingReads>
    <Instrument>ILMN-1</Instrument>
    <MakeFlag>1</MakeFlag>
    <MaxCycle>-1</MaxCycle>
    <MinCycle>-1</MinCycle>
    <Reads Index="1">
      <FirstCycle>1</FirstCycle>
      <LastCycle>24</LastCycle>
      <RunFolder>/data/070813_ILMN-1_0217_FC1234</RunFolder>
    </Reads>
    <RunFolder>/data/070813_ILMN-1_0217_FC1234</RunFolder>
  </RunParameters>
  <Software Name="RTA" Version="1.6.2.0" />
  <TileSelection>
    <Lane Index="5">
      ...
RunInfo.xml File

The top level Run Folder contains a RunInfo.xml file. The file RunInfo.xml (normally generated by SCS) identifies the boundaries of the reads (including index reads), and contains run-specific information such as instrument used and flow cell ID.

The XML tags in the RunInfo.xml file are self-explanatory. The following shows an example of a RunInfo.xml file:

```xml
<?xml version="1.0" ?>
<RunInfo>
  <Run Id="090903_HWD-SMANNHERI_0020" Number="20">
    <Instrument>EAS-297</Instrument>
    <FlowCellId>FC12312xx</FlowCellId>
    <Cycles Incorporation="18" Cleavage="0" ReadPrep="0" />
    <Tiles>
      <Lane Index="1" Incorporation="120" ReadPrep="0" />
      <Lane Index="2" Incorporation="120" ReadPrep="0" />
      <Lane Index="3" Incorporation="120" ReadPrep="0" />
      <Lane Index="4" Incorporation="120" ReadPrep="0" />
      <Lane Index="5" Incorporation="120" ReadPrep="0" />
      <Lane Index="6" Incorporation="120" ReadPrep="0" />
      <Lane Index="7" Incorporation="120" ReadPrep="0" />
      <Lane Index="8" Incorporation="120" ReadPrep="0" />
    </Tiles>
    <Reads>
      <Read FirstCycle="1" LastCycle="18" />
    </Reads>
    <ActualIndex />
  </Run>
</RunInfo>
```

Run.conf.xml

The run.conf.xml file provides the location of the GERALD data (export.txt) for each flow cell run, and describes properties of each flow cell run. There is one run section for each flow cell run, one set section for each GERALD folder in each flow cell run, and one lane section for each lane in each set. The run.conf.xml file can be provided (created) by the user or CASAVA will generate it automatically based on command line options. run.conf.xml file should be placed in projectDirectory/conf/

Example:

```xml
<opt>
  <run id="080328_EAS114_0051_FC3011HAAXX">
```
<set setid="1" gerald="/Human/080328_EAS139_0029_FC3010UAAXX_R1/Data/Intensities/Basecalls/GERALD_09-04-2008_aladdin/"
readMode="paired">
  <lanes>1</lanes>
  <lanes>2</lanes>
... list of lanes
</set>
</run>
</opt>

NOTE
There should only be one flow cell within each <run>.
Appendix C

Using Parallelization in GERALD

Topics

110  Introduction
110  “Make” Utilities
    110  Standard “Make”
    110  Customizing Parallelization
    110  Distributed “Make”
111  Parallelization Limitations
111  Memory Limitations
Introduction

One of the main considerations behind the current GERALD architecture is the ability to use the parallelization facilities present on almost all SMP machines and on most Linux/Unix clusters. Parallelization is scalable and makes use of all available CPU power.

“Make” Utilities

Parallelization is built around the ability of the standard “make” utility to execute in parallel across multiple processes on the same computer. GERALD also provides a series of checkpoints and hooks that enables you to customize the parallelization for your computing setup. See Customizing Parallelization on page 110 for details.

Standard “Make”

The standard “make” utility has many limitations, but it is universally available and has a built-in parallelization switch (“-j”). For example, on a dual-processor, dual-core system, running “make -j 4” instead of “make,” executes the GERALD run in parallel over four different processor cores, with an almost 4-fold decrease in analysis run time. On a 4-way SMP system, “-j 8” or more may be advisable.

Distributed “Make”

There are several distributed versions of “make” for cluster systems. Frequently used versions include “qmake” from Sun Grid Engine and “Ismake” from LSF.

To use “qmake,” a short wrapper script is required. See the grid engine documentation for details.

There are known issues with the use of “Ismake” that prevent parts of GERALD from running. Therefore, Illumina does not recommend using “Ismake” to run GERALD.

Customizing Parallelization

Many parts of GERALD are intrinsically parallelizable by lane or tile. However, some parts of GERALD cannot be parallelized completely. GERALD has a series of additional hooks and check-points for customization.

The GERALD can be divided into a series of steps with different levels of scalability where synchronization “barriers” cause GERALD to wait for each of the tasks within a step to finish before going to the next step.
You can parallelize the steps at the run level (no parallelization), the lane level (up to eight jobs in parallel), and the tile level (up to thousands of jobs in parallel). Each step is initiated by a “make” target. After completion of each of these steps, GERALD produces a file or a series of files at the lane/tile level, that determines whether all jobs belonging to the step have finished. Finally, hooks are provided upon completion of the step to issue user-defined external commands.

**Sequence Alignment**

This section lists the steps, corresponding make targets, checkfiles and hooks for sequence alignment by the GERALD module.

<table>
<thead>
<tr>
<th>Parallelization Level</th>
<th>Run</th>
<th>Lane</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td></td>
<td>s_1</td>
</tr>
<tr>
<td>Check File</td>
<td></td>
<td>s_1_finished.txt</td>
</tr>
<tr>
<td>Hook</td>
<td></td>
<td>(none)</td>
</tr>
<tr>
<td>Target</td>
<td></td>
<td>all</td>
</tr>
<tr>
<td>Check File</td>
<td></td>
<td>finished.txt</td>
</tr>
<tr>
<td>Hook</td>
<td></td>
<td>POST_RUN_COMMAND (Accessible from GERALD config file)</td>
</tr>
</tbody>
</table>

**Parallelization Limitations**

The analysis works on a per-tile basis, so the maximum degree of parallelization achievable is equal to the total number of tiles scanned during the run. However, some parts of the GERALD operate on a per-lane basis, and a few parts on a per-run basis, which means that scaling will cease to be linear at some stage for more than 8-way parallelization. The ELAND_SET_SIZE affects the maximum level of parallelism available for ELAND. Assuming that all lanes have 120 tiles, a value of 120 will lead to 8 processes, 60 to 16 processes, 40 to 24 processes, etc. These values are doubled for paired end runs.

**Memory Limitations**

GERALD requires a minimum of 2 GB RAM available per concurrent process. ELANDv2 parallelisation is artificially prevented by an non-essential make dependency. If you are certain that you cannot exhaust your available memory, you can use ELAND_SET_SIZE to the GERALD config file to remove this dependency. However, you are responsible for making sure that you have up to 16 GB of RAM at your disposal. For additional information, see *Using GERALD for Sequence Alignment* on page 23.
Appendix D

Base Call Calibration and Alignment Scoring

Topics

114  Introduction
114  Base Call Calibration
117  Extracting Reference Bases
118  Creating a Quality Table
121  Generating New Quality Values
123  Configuring Quality Table Sources in GERALD
Introduction

This section provides advanced description and usage of custom base call quality value calibration. By default, a precalculated calibration table, based upon a large sample of analysis runs and supplied with CASAVA, will be used for quality value calibration. Thus, it should not in general be necessary to apply a custom recalibration.

NOTE

Custom base call calibration is only available for ANALYSIS eland_extended and ANALYSIS eland_pair.

Base Call Calibration

The CASAVA v1.6 contains a module for quality score calibration. This is essentially a re-implementation of the method described in the Phred paper by Ewing & Green (Ewing B, Green P (1998) Base-Calling of Automated Sequencer Traces Using Phred. II. Error Probabilities. Genome Research, 8, 186).

The quality scores stored in the _qseq files are the former qcal values (based on the static qtable) used in Pipeline v1.0 and earlier. Since the recalibration uses the _qval.txt.gz files, and since these files are now optional, the command-line option --with-qval must be explicitly passed on the command line of bustard.py (or goat_pipeline.py) if you want to generate the recalibrated qtable.

The score is not perfect, particularly for high quality scores Q >> 10. Their dynamic range is artificially limited (currently to Q40). This is an arbitrary cut-off motivated by the fact that the quality estimation procedure used by the base caller is unlikely to be accurate beyond the cut-off.

Goal

The main goal of the quality calibration is to bring the quality scores and corresponding predicted error rates that the base caller generates into line with the error rates obtained from an alignment.

Method

The calibration framework uses a set of trace-specific and base-specific parameters that are indicative of the base call quality to predict a new calibrated quality score. The mapping between parameters and quality scores is encoded in a Phred table (qtable.txt). The mapping is calibrated on a set of known alignments. GERALD performs the calibration procedure as a post-processing step to the base calling.

Each lane can be calibrated separately. The alignment obtained from the lane itself can be used as a training set, resulting in a procedure we refer to as auto-calibration (e.g. training set and target data are the same). In addition, a table derived from a different data set (or a control lane) can be applied (cross-calibration).
Limitations of the Recalibration Method

The aim of recalibration in general is to improve the correlation of the scores with the error rates obtained from alignment against a known reference. There are some limitations to this method as described in this section.

Major Alignment Errors

The auto-calibration procedure that can be applied as a custom quality calibration method is based on the assumption that the alignments are more or less correct; violations of that assumption will skew the calibration and limit the accuracy of the calibration. For example, if the sample in question contains a contaminant (e.g. E. coli sequence) at the 5% level, the corresponding reads (depending on read-length and target genome) may be mistaken as aligned reads and contribute significantly to the error rates.

SNP Rate

A reference obtained for a different individual would limit the maximum quality to the rate at which SNPs are observed, since the quality scores cannot get better than the SNP rate.

Size of Data Set

The maximum alignment score is also limited by the size of the data set used for calibration. For example, in order to obtain Q40, you would need around $10^4$ base pairs at a quality of Q40 (presumably, even more because of the Poisson counting error).

Quality Value Predictors

In order to perform custom quality value recalibration, the production of quality value predictor files (_qval.txt.gz) during the Bustard analysis stage must be configured by specifying the --with-qval options of goat_pipeline.py or bustard.py.

<table>
<thead>
<tr>
<th>Predictor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gc_content</td>
<td>Fraction of (non-N) bases called as either G or C in a read</td>
</tr>
<tr>
<td>unbounded_signal_decay</td>
<td>One minus the ratio of this cycle's highest processed intensity to that of the first cycle in the read</td>
</tr>
<tr>
<td>unpurity_nhood1</td>
<td>Highest value of unpurity within 1-cycle of the current cycle, where unpurity is one minus the the ratio of the highest processed intensity to the sum of the processed intensities in all channels</td>
</tr>
<tr>
<td>shifted_unpurity</td>
<td>One minus the ratio of the highest scaled processed intensity value to the sum of these values in all channels, where the processed intensities for each cycle are scaled such that the lowest value is one</td>
</tr>
</tbody>
</table>

**NOTE**

The QVAL_COMPR definition in the GERALD makefile specifies which type of compression of the qval.txt files, if any, should be expected by the tools that use them. By default it is configured to match the gzip compression that will be applied in their production.
The format of the uncompressed qval file is as follows:

- Each row corresponds to one cycle.
- The columns are tab-separated and each contains the values for one of the quality predictors, with the exception of the last column, which contains the called base for that cycle.

**NOTE**

The naming of the predictors reflects the fact that the prediction algorithm expects higher predictor values to correspond to lower base call quality. For example, “un” is simply “1 - .”
Extracting Reference Bases

In addition to the information contained in the qval files, the calibration process also requires the reference value for each base for which it is available; this is the case when the base is within a read that has been uniquely aligned.

NOTE
In versions of the Pipeline prior to version 1.0, these reference base values were included in the qval files; the current separation reflects the stages within the Pipeline at which the two types of information become available.

Method
The called bases are derived from the combination of qseq files with saf files (or from the corresponding align files that are produced by older analysis modes) using saf2qref.pl Perl script (or align2qref.pl for older analysis modes).

Output File Format
The format of the qref file produced with either analysis mode is the same, a single column of reference bases. Within sets of bases corresponding to an aligned read, any bases corresponding to cycles masked out by USE_BASES will be represented by a dot (period). The bases corresponding to reads not uniquely aligned will all be represented in the same manner, as will any unknown bases in reference sequence to which reads have been uniquely aligned.
Creating a Quality Table

In this section, the creation of a quality table (qtable file) is explained.

NOTE
Custom base call calibration is only available for ANALYSIS eland_extended and ANALYSIS eland_pair.

Example
An example of quality table generation is (assuming the current directory is a GERALD folder):

```
/path/CASAVA/bin/QualityCalibration
--cfg /path/CASAVA/share/QualityCalibration.xml
--qval_compression gzip --qval_dir ..
--tile_list_file tiles.txt --tile_prefix s_1 --read 1
--qval_suffix _qval.txt.gz --qref_suffix _qref.txt
> s_1_1_qtable.txt
```

The set of qval and qref files to use is constructed from the subset of tiles in tiles.txt that have prefix s_1 (i.e. lane 1 tiles) by using the read number and the specified directories (here only a qval directory is specified) and suffixes.

The predictor value columns in each qval file must correspond to the predictors specified in the configuration file (here QualityCalibration.xml).

NOTE
The calibration routines are agnostic as to the number of parameters, so it would be easy to use a different set of parameters with a different extractor script. The binning for these parameters must be configured in an XML file such as QualityCalibration.xml. The path is the first argument expected by QualityCalibration. As yet, the implementation of QualityCalibration does not support the auto-generation of parameter binnings.

Predictor Bin Boundaries
Bin boundaries are used to divide predictor values into bins such that N unique bin boundaries specify N+1 bins. At least one bin boundary must be specified for each predictor. Two schemata are supported for the specification of predictor bin boundaries:

- One allows bin boundaries to be supplied as a list of increasing values.

  ```xml
  <Parameter>
  <Name>Predictor</Name>
  <BinBoundaries>-0.7 0.3 4.2 20.5 101.0</BinBoundaries>
  </Parameter>
  ```

- The other allows the specification of equidistant bin boundaries, by specifying the first and last bin boundaries in addition to the total number of equidistant bin boundaries which are to be created within this range. The minimum number of bin boundaries allowed using this schema is 2, in which case only the first and last bin boundaries are used.
QualityCalibration derives the name of a corresponding qref file (containing the reference bases that it needs) from each qval file that is specified. It does this by replacing the qval filename suffix (by default, _qval.txt) with the qref filename suffix (by default, _qref.txt). These suffix strings can be overridden if required (see Overriding Suffix Strings on page 120). The directories containing the qval and/or qref files may also be specified.

The qval files can either be listed as explicit arguments to QualityCalibration or can be derived from the contents of a tile list file, tiles.txt. In the latter case, the qval file names are generated from those tile names in the tile list file that match a specified prefix, by appending the default or specified qval file suffix.

QualityCalibration produces a qtable file, which specifies the rules for a a Phred-type quality model. The qtable file may contain any number of comment lines beginning with the # character. The remaining lines contain Phred model rules, which are tab separated columns consisting of a threshold value corresponding to each predictor in the qval files used to train the model, followed by a quality score, followed by the number of trials and errors from the training data associated with that rule.

In standard GERALD practice, the prefix corresponds to a lane (e.g. “s_1” for lane 1) and the output is written to one quality table for that lane for single read analysis.

For paired-end analysis, two per-read quality tables are produced by separate applications of QualityCalibration to each per-read set of qval files, with specification of the latter by suffix strings that include the read number (e.g. _1_qval.txt could be used for read 1 and _2_qval.txt for read 2).

QualityCalibration can read from compressed qval (and/or qref) files, if this is specified. The corresponding suffixes will then have to be specified in full as QualityCalibration does not automatically append compression-related extensions.

The detailed usage of the QualityCalibration binary is as follows:

- For information about usage:
  ```
  ./QualityCalibration -h|--help
  ```

- For creating the quality table (with available options listed):
  ```
  ./QualityCalibration
  [-D|--qval_dir QVAL_DIR]
  [-d|--qref_dir QREF_DIR]
  [-S|--qval_suffix _qval.txt]
  ```

Note that both examples specify 5 bin boundaries, in which case predictor values will be divided into 6 regions.
[-s|--qref_suffix _qref.txt]
[-z|--qval_compression none|gzip|bzip2]
[-z|--qref_compression none|gzip|bzip2]
[-r|--read READ]
[-q|--qscore_type phred|phred_cons|solexa]
[-a|--alpha value]
-c|--cfg <cfg_file.xml>
< <qvalue-file1> [<qvalue-file2> ...]
| -T|--tile_list_file <tiles.txt>
  -t|--tile_prefix <lane_prefix e.g. s_3> >

Overriding Suffix Strings

The qval_suffix and qref_suffix options (with default values as shown above) are used to generate qref filenames from the specified qval filenames. In addition, if the qval files are specified by means of a tile list file plus lane prefix filter, the qval_suffix is appended to the tile names to generate the qval filenames. Note that the specified read number will also be incorporated into the qval filenames in the latter case.
Generating New Quality Values

In standard CASAVA practice, QualityApply is run once per lane (or per read per lane) to produce a per-tile (per-read) output qseq file from all the corresponding qval files and input qseq files. The output qseq files contain custom recalibrated quality values; the other data in each input qseq file is passed through unchanged to the corresponding output qseq file.

Example

The following is an example of reestimation of qval:

```bash
CASAVA/bin/QualityApply
  --orig_read_lengths 36:36
  --tile_list_file tiles.txt
  --qval_compression gzip
  --qval_dir ..
  --qval_suffix _qval.txt.gz
  --input_qseq_dir ..
  --input_qseq_suffix _qseq.txt
  --output_qseq_dir .
  --output_qseq_suffix _custom_qseq.txt
  --tile_prefix s_5
  --read 1
  s_5_qtable.txt
```

This shows the reestimation of the quality values for base calls within the reads in one lane, based upon a quality table calculated for the whole lane.

Input Data

QualityApply takes as input a set of corresponding qval and qseq files plus a qtable (calibration table) file. The qval and input qseq files are specified by the tile list file (as filtered using the tile prefix) in conjunction with the directory, read, suffix and compression options.

Output Data

The output format for the reestimated quality values is symbolic. Specifically, the value is represented by the ASCII character for which the code is the value added to 64, allowing a range of negative as well as positive values to be compactly displayed. Each output row is a single string of such characters, corresponding to the bases of a single read.

NOTE

The output format option --numeric is deprecated in Pipeline v1.3 to increase efficiency and memory use.

These quality values are written to the output qseq files along with the other fields passed through from the corresponding input qseq files. As for the qval and input qseq files, a directory and suffix may be specified.

Compression

Once again, the qval files may be supplied and used in compressed form if this is specified; the use of any compression extension will require the explicit specification of a qval file suffix including it.
Detailed Usage of QualityApply

The detailed usage of the QualityApply binary is as follows:

- For information about usage:
  `QualityApply -h|--help`

- For generating new quality values (with available options listed):
  `QualityApply -l|--orig_read_lengths <ORIG_READ_LENGTHS>
  [-r|--read READ_NUM]
  [-T|--tile_list_file <tiles.txt>]
  -t|--tile_prefix <e.g. s_3>
  [-Z|--qval_compression none|gzip|bzip2]
  [-D|--qval_dir <qval_dir>]
  [-i|--input_qseq_dir <input_qseq_dir>]
  [-o|--output_qseq_dir <output_qseq_dir>]
  [-S|--qval_suffix _qval.txt]
  [-q|--input_qseq_suffix _qseq.txt]
  [-Q|--output_qseq_suffix _common_qseq.txt]
  <qtable-file>`
Configuring Quality Table Sources in GERALD

By default, the GERALD generates a quality table for each lane in which an analysis including alignment is performed. Then, GERALD uses this quality table (or pair of quality tables) to reestimate the base call quality values of all the tiles in that lane.

The source of the quality table(s) used in the quality calibration for a lane may be overridden by defining QCAL_SOURCE (or, for individual reads within a paired-end analysis, QCAL_SOURCE1 and/or QCAL_SOURCE2) in the config.txt passed to GERALD.pl to configure GERALD analysis.

**Supported Values**

<table>
<thead>
<tr>
<th>Supported Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>upstream</td>
<td>No custom recalibration is performed at the GERALD analysis stage. The quality values produced (at the Bustard analysis stage) from a precalculated qtable will be used.</td>
</tr>
<tr>
<td>auto</td>
<td>the qtable(s) used within the lane (to re-estimate the base call quality values) will be the qtable(s) generated for that lane (from the quality predictor values and called and reference base values of bases in reads from that lane)</td>
</tr>
<tr>
<td>auto&lt;n&gt;, where n is the number of a lane for which alignment will be performed</td>
<td>the qtable(s) used in the lane will be those generated for lane n, e.g. <code>auto5</code> means that the qtable(s) from lane 5 will be used</td>
</tr>
<tr>
<td>/path/to/qtable.txt</td>
<td>the qtable file at the specified path will be used</td>
</tr>
</tbody>
</table>

As with any config.txt variable, the QCAL_SOURCE variables may be specified on a flow cell-wide basis or for any non-overlapping subsets of the flow cell lanes, with the latter overriding the former if both are specified.

**Paired-End Analysis**

In a paired-end analysis lane, specification of QCAL_SOURCE1 and/or QCAL_SOURCE2 will override specification of QCAL_SOURCE, although the latter will be used if it has been specified and not overridden for a given read.

The interpretations for cases where paired-end analysis is in use for either or both of the source lane and the target lane are intended to be based upon the principle of least surprise:

1. If both lanes are paired, then any specification of the source lane for read 1 of the target lane results in the read 1 qtable of the source lane being used as the read 1 qtable in the target lane - and similarly for read 2.
2. If only the target lane is paired, then there is only one qtable available in the source lane but it may be used for both reads in the target lane.
3. If only the source lane is paired, then the read 1 qtable is used.

**Example**

```
ANALYSIS eland_pair
```
QCAL_SOURCE /home/illumina/ref42_qtable.txt
123:QCAL_SOURCE auto8
4:QCAL_SOURCE1 /home/illumina/ref51_qtable.txt
4:QCAL_SOURCE2 auto7

The previous example specifies the following qtable usage:

1. Lanes 1–3 read 1 will use the lane 8 read 1 qtable; lanes 1–3 read 2 will use the lane 8 read 2 qtable
2. Lane 4 read 1 will use the external ref51_qtable.txt qtable
3. Lane 4 read 2 will use the lane 7 read 2 qtable
4. Lanes 5–8 read 1 and read 2 will use the ref42_qtable.txt qtable

Even though the lane 8 qtables will not be needed in lane 8, they will still be generated for use in lanes 1–3.
Appendix E

Reference Files CASAVA

Topics

126 Introduction
  126 Provided Reference Files
  126 Custom Reference Files

126 ELAND Reference Files
  126 ELAND Genome Files
  127 Abundant Sequences Files
  127 Splice Junction Set
  127 ELAND Splice Junction Files

129 CASAVA Reference Files
  129 CASAVA Genome Files
  129 Genome Size File
  129 Exon Coordinates Set
  130 Provided CASAVA Reference Files

131 Generating Reference Files
  131 Getting Data Files
  132 Genome Sequence Files
  133 Abundant Sequence Files
  133 Splice Junction Set
  135 Genome Size Files
  135 Non-Redundant Exon Set
  136 Squashing the Reference Files
Introduction

CASAVA needs a number of special reference files to run analysis, especially for RNA sequencing.

This chapter describes the reference files that are needed to run Eland and CASAVA variant detection, and provides instructions how to generate these files for other species and builds.

Provided Reference Files

Some reference files (indicated below) are provided for the following species:

- Human genome build 36
- Mouse genome build 37
- Rat genome build 3.4

Genome sequence files are not provided, and you will need to download them from NCBI or UCSC (Getting Data Files on page 131).

Custom Reference Files

If you want to run the analysis for other species or other builds than those mentioned above, you will need to generate some of the reference files.

See Generating Reference Files on page 131 for instructions.

ELAND Reference Files

ELAND needs the following file to perform an alignment:

- Squashed genome sequence files

In addition, Eland_rna needs two types of files to analyze RNA Sequencing data:

- Abundant sequences files: mitochondrial DNA, ribosomal region sequences, 5S RNA (optional), and other contaminants
- Splice junction set files

The splice junction set files for human genome build 36, mouse genome build 37, and rat genome build 3.4 are provided. The sections below indicate where the files needed for running eland_rna are located after installation.

ELAND Genome Files

All ELAND modules use the same genome sequence files. These files are derived from FASTA files that contain one assembled chromosome, available from UCSC or NCBI. See Getting Data Files on page 131 for download instructions.

There must be one FASTA file per chromosome, with the naming scheme c1.fa, c2.fa, etc.

These FASTA files must be squashed into the 2-bits-per-base format that the ELAND aligner understands (see Squashing the Reference Files on page 136). The result is one fa.2pb file and one fa.vld file for each squashed chromosome. If a file contains multiple sequences (e.g. several scaffolds in one multi-FASTA file) there will be an extra fa.idx file.
Abundant Sequences Files

Eland_rna uses these files to mask hits to abundant or contaminant sequences. The files can be derived from the following sources:

- Mitochondrial DNA
- Ribosomal repeat region sequences
- 5S RNA (optional)
- Other contaminants, for example phiX, if phiX spikes are used

Eland_rna uses squashed FASTA files, similar to the genome sequence files (see Squashing the Reference Files on page 136). The result is one fa.2bpb file and one fa.vld file for each abundant sequence category.

Splice Junction Set

Eland_rna uses the splice junction set to align reads across splice junctions. The splice junction set is generated by the splice_sites.pl from publicly available genome data.

The splice_sites.pl generates a multi-FASTA file containing the exonic sequence upstream and downstream of all splice junctions in multi-FASTA format:

```
>WDR78_28_19_1_67100573_67109640
TGTGATAAAATCATAATGGAAATGGCATATAATGTCCTGCCGATTTG
TAGATTTAGT
```

Since the FASTA file needs to be squashed before use in Eland_rna, one splice junction set ultimately consists of three files (see Squashing the Reference Files on page 136):

- splice_sites-xx.fa.2bpb
- splice_sites-xx.fa.vld
- splice_sites-xx.fa.idx

with xx being the length of the sequence upstream and downstream of the splice junction that is retained in the FASTA sequence file.

This should be tailored to the length of the sequence reads; ideally, xx = read length - 1. For example, for a read length of 50 bases, you would like a splice junction set that contains 49 bases upstream and downstream of every splice site. For exons that are smaller than 49 bases, the entire exon is used.

ELAND Splice Junction Files

The locations of the provided ELAND splice junction files are indicated in the table below. Here we assume all genomic data are located in /data. The path mentioned below is just an example and is not hard coded. The path are defined for ELAND in the configuration file as described in page 51.

---

**NOTE**

Here we assume all genomic data are located in /data. The path mentioned below is an example and not hard coded. The path are defined for CASAVA in the command line switches as described in Options on page 73.
<table>
<thead>
<tr>
<th>Build</th>
<th>Reference Type</th>
<th>Folder</th>
<th>Reference Files</th>
</tr>
</thead>
</table>
| Human genome build 36   | Splice Junction Set, 34 bases      | /data/Genome/ELAND_RNA/Human/human.34.splice             | splice_sites-34.fa.2bpb
|                         |                                    |                                                          | splice_sites-34.fa.idx               |
|                         |                                    |                                                          | splice_sites-34.fa.vld               |
| Human genome build 36   | Splice Junction Set, 49 bases      | /data/Genome/ELAND_RNA/Human/human.49.splice             | splice_sites-49.fa.2bpb              |
|                         |                                    |                                                          | splice_sites-49.fa.idx               |
|                         |                                    |                                                          | splice_sites-49.fa.vld               |
| Mouse genome build 37   | Splice Junction Set, 34 bases      | /data/Genome/ELAND_RNA/mouse/mm9.34.splice               | splice_sites-34.fa.2bpb              |
|                         |                                    |                                                          | splice_sites-34.fa.idx               |
|                         |                                    |                                                          | splice_sites-34.fa.vld               |
| Mouse genome build 37   | Splice Junction Set, 49 bases      | /data/Genome/ELAND_RNA/mouse/mm9.49.splice               | splice_sites-49.fa.2bpb              |
|                         |                                    |                                                          | splice_sites-49.fa.idx               |
|                         |                                    |                                                          | splice_sites-49.fa.vld               |
| Rat genome build 3.4    | Splice Junction Set, 34 bases      | /data/Genome/ELAND_RNA/rn4.34.splice                     | splice_sites-34.fa.2bpb              |
|                         |                                    |                                                          | splice_sites-34.fa.idx               |
|                         |                                    |                                                          | splice_sites-34.fa.vld               |
| Rat genome build 3.4    | Splice Junction Set, 49 bases      | /data/Genome/ELAND_RNA/rn4.49.splice                     | splice_sites-49.fa.2bpb              |
|                         |                                    |                                                          | splice_sites-49.fa.idx               |
|                         |                                    |                                                          | splice_sites-49.fa.vld               |
CASAVA Reference Files

CASAVA needs three types of files to analyze RNA Sequencing data:

- Genome sequence files
- Genome size file
- Exon coordinates set. It is crucial that the exon coordinate set and splice junction file are generated from the same reference set of transcripts.

The genome size file and exon coordinates set files for human genome build 36, mouse genome build 37, and rat genome build 3.4 are on the installation DVD. The tables below indicate where the files needed for CASAVA are located after installation.

**NOTE**

CASAVA for DNA sequencing only needs the genome sequence and genome size files.

**CASAVA Genome Files**

CASAVA uses the same genome sequence files as Eland_rna (see ELAND Genome Files on page 126). The only difference is that the files should not be squashed. These files are available from UCSC or NCBI. See Getting Data Files on page 131 for download instructions.

CASAVA uses the naming scheme chr1.fa, chr2.fa, etc.

**Genome Size File**

CASAVA uses a genome size file that the analysis software generates, named <species>_genomes_size.xml. This file contains the sizes of the chromosomes, in xml format:

```xml
<SequenceSizes>
  <c2.fa>242751149</c2.fa>
  <cX.fa>154913754</cX.fa>
  ...
  <c21.fa>46944323</c21.fa>
</SequenceSizes>
```

**Exon Coordinates Set**

CASAVA uses a non-redundant exon coordinates set to compute the gene expression counts. For every gene, overlapping exons from the same gene are combined to form one larger region used for counting. Regions where there are overlapping exons from other genes are excluded from counting and are not present in the non-redundant exon set file.

The non-redundant exon coordinates set is generated by the nonoverlapping_exon_coords.pl from publicly available genome data (see Exon Coordinates File Preparation Method on page 136). The file looks like this:

```plaintext
chr21 46440018 46440098 LSS
chr21 46879955 46880103 PRMT2
chr21 40472724 40472891 DSCAM
```
Provided CASAVA Reference Files

The locations of the provided CASAVA reference files for human genome build are indicated in the table below.

**NOTE**

Here we assume all genomic data are located in /data. The path mentioned below is an example and not hard coded. The path are defined for CASAVA in the command line switches as described in Options on page 73.

<table>
<thead>
<tr>
<th>Reference Type</th>
<th>Folder</th>
<th>Reference Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human genome build 36</td>
<td>Genome Size File</td>
<td>human_genomes_size.xml</td>
</tr>
<tr>
<td>Human genome build 36</td>
<td>Exon Coordinates Set</td>
<td>exon_coords.txt</td>
</tr>
<tr>
<td>Mouse genome build 37</td>
<td>Genome Size File</td>
<td>mouse_genomes_size.xml</td>
</tr>
<tr>
<td>Mouse genome build 37</td>
<td>Exon Coordinates Set</td>
<td>mm9.exon_coords.txt</td>
</tr>
<tr>
<td>Rat genome build 3.4</td>
<td>Genome Size File</td>
<td>rat_genomes_size.xml</td>
</tr>
<tr>
<td>Rat genome build 3.4</td>
<td>Exon Coordinates Set</td>
<td>mm4.exon_coords.txt</td>
</tr>
</tbody>
</table>
Generating Reference Files

To run CASAVA, you will need to download genome and other reference files, and run some scripts on these. This is explained in this section.

Getting Data Files

Download additional genome files from UCSC GoldenPath database using the following procedure.

1. Go to: http://hgdownload.cse.ucsc.edu/downloads.html (Figure 23).

2. Click the species name of the genome you want to download; e.g., Human.
   The web page for the genome you selected appears (Figure 24).

3. Click Annotation database.
   The annotation database for your selected species appears.

4. On the Annotation Database web page, scroll down until you find files available to download (Figure 25).
5. Create a folder, and download the refFlat.txt.gz file into this folder. This file is needed to generate the exon coordinates set and splice junction set.

6. In your browser, click Back to return to the genome web page shown in Step 2 (Figure 24).

7. Click Data set by chromosome.
   The Index (Data set by chromosome) web page for the genome you selected appears (Figure 26).

8. Create a folder for the genome sequence files, and download each of the assembled sequence files (for example, chr1.fa.gz) shown on the Index web page into the folder you created.

9. Unzip the refFlat.txt.gz file and the sequence files.

Genome Sequence Files

Process the genome sequence files the following way:

1. Go to the folder containing the genome sequence files.

2. Squash the FASTA files into the 2-bits-per-base format that the ELAND aligner understands. This is described in Squashing the Reference Files on page 136. Make sure to collect the output in a new folder for the squashed genome.

   The result is one fa.2bpb file (containing compressed sequence data) and one fa.vld file (containing the start and end positions) for each squashed chromosome.
The squashed genome sequence files should be used with ELAND, and the unsquashed genome sequence files should be used with CASAVA and PhageAlign.

Abundant Sequence Files

Process the abundant sequence files the following way:

1. Generate a folder for abundant sequences.
2. Collect FASTA files for abundant sequences in the abundant sequences folder; for example, files containing these sequences:
   - The cM.fa file from the genome folder.
   - A ribosomal sequences FASTA file. You will need to find it for your genome of interest, for example, from GenBank.
   - A 5SRNA FASTA file (optional). You will need to find it for your genome of interest, for example, from GenBank.
   - A contaminants file. You can use the same the same newcontam.fa file as for human, mouse or rat.

   You do not need to have all of the files listed above, but you need at least one file for eland_rna to work properly. You can add other abundant sequences FASTA files if desired.
3. Squash the FASTA files into the 2-bits-per-base format that the ELAND aligner understands. This is described in Squashing the Reference Files on page 136. Make sure to collect the output in a new folder for the squashed abundant sequences.

   The result is one fa.2bpb file (containing compressed sequence data) and one fa.vld file (containing the start and end positions) for each FASTA file.

Splice Junction Set

The splice junction file is a FASTA file containing a non-redundant set of splice junctions from known RefSeq transcripts.

The splice junctions are defined as the boundaries between two exons. An example could be splice junction AB which is the junction between exon A and exon B. An equal number of bases are taken from the end of exon A and the start of exon B. If the total length of the exon is less that this number, the length of the exon is taken. The sequences are taken from the forward genomic strand, irrespective if the gene is encoded on the forward or reverse strand. These two sequences are then concatenated to form the splice junction AB.

This file is used by Eland RNA to map reads to the splice junctions.

Format

Splice junction sequences are stored in FASTA format, and are annotated using the following format (delimited by underscores):

ID_NumberOfBasesFrom5primeExon_NumberOfBasesFrom3primeExon_Chromosome_EndPosition5primeExon_StartPosition3primeExon

Characteristics

The splice junction annotation has the following characteristics:

- 5’ and 3’ refer to the forward genomic strand (not the mRNA).
Chromosomal positions and sequences are all based on the forward strand.

The first base of each chromosome is base at position 1.

Avoid using non-alphabetic characters in ID, if present these should be replaced by a dash.

The field Chromosomes contains the entire chromosome file name including the .fa extension.

The Chromosome positions are inclusive, meaning the base at EndPosition5primeExon belongs to the 5’ exon and same for StartPosition3primeExon

For UCSC the ID is the gene symbol or refseq ID.

The gene identifiers used in the splice junction file should match the identifiers in the non-redundant exon set file described below.

The genome build on which these positions are based should be the same as the build that was used for the alignment of the reads.

Example

UCSC

An example of a UCSC annotation:

>WDR78_28_19_1_67100573_67109640
TGTGATAAAATCATAATGGAAGATAAAGGCATAATGTCCACTGCCTGGGATTTG
TANNNNN

This junction has the following characteristics:

- Belongs to gene WDR78
- 28 bases were taken from the left exon which ends at position 67100573 on chromosome 1
- 19 bases were taken from the right exon which starts at position 67109640

Splice Junction Set Preparation Method

The splice junction set can be created by the Perl script splice_sites.pl, which is included in the installation. You need the input files:

- refFlat.txt file
- Genome folder containing FASTA files with the assembled chromosome sequences

Perform the following steps:

1. Generate a splice junction set folder.
2. Navigate to the splice junction set folder.
3. Run the script:

   perl /path/to/splice_sites.pl --refSequences=/path/to/
   genome_folder --leftFlank=i [--rightFlank=j]
   [chromosome_prefix]
   < /path/to/refFlat.txt > output_file_name

   with the following analysis parameters.
Generating Reference Files

The output files are a FASTA file containing the splice junction sequences, and a fa.idx file that annotates those sequences.

4. Squash the FASTA file into the 2-bits-per-base format that the ELANDv2 aligner understands. This is described in Squashing the Reference Files on page 136. Make sure to collect the output in a new folder for the abundant sequences.

The result is one fa.2bpb file (containing compressed sequence data), one fa.vld file (containing the start and end positions), and one fa.idx file (produced to keep track of where each splice junction starts and finishes in the .2bpb file).

The genome_size.xml file is always generated by the analysis software, so you can find it in the GERALD directory.

The non-redundant exon set is used by CASAVA to compute the gene expression counts.

For every gene, overlapping exons from the same gene are combined to form one larger region used for counting.

Regions where there are overlapping exons from other genes are excluded from counting and are not present in the non-redundant exon set file.

The format of the non-redundant exon set file consists of a tab delimited text file with four columns:

1. The first column contains the chromosome name.
2. The second column the start of a region.
3. The third column the end of a region.
4. The fourth column is the gene identifier to which the region belongs to.
Characteristics

The non-redundant exon set file has the following characteristics:

- Start and end positions are given on the forward genomic strand.
- The first base on each chromosome is at position 1.
- The bases at both the start and end positions should belong to the specified region (inclusive).
- The genome build on which these positions are based should be the same as the build that was used for the alignment of the reads.
- The gene identifiers used in the non-redundant exon set should match the identifiers in the splice junction file described above.

Examples

Example lines of the exon_coords.txt file:

```
chr17 7514652 7514758 TP53
chr17 7520037 7520315 TP53
chr21 46438833 46439003 LSS
chr21 46440018 46440098 LSS
chr21 46451014 46451110 LSS
```

Exon Coordinates File Preparation Method

The exon coordinates can be created by the Perl script nonoverlapping_exon_coords.pl, which is included in the installation. You need the input file refFlat.txt file to run this.

Perform the following steps:

1. Generate an exon coordinates set folder.
2. Navigate to the exon coordinates set folder.
3. Run the script:

   ```
   /path/to/nonoverlapping_exon_coords.pl [options]
   < /path/to/refFlat.txt >exon_coords.txt
   ```

   The output file is the exon_coords.txt.
4. Move the exon_coords.txt to the "features" subfolder of the CASAVA folder.

Squashing the Reference Files

Several of the GERALD analysis modes (namely eland, eland_extended, eland_pair, eland_rna, and elands_ms_*) make use of the ELAND alignment program to align the reads produced by the GERALD against a set of reference sequences. Before you can run an analysis that uses ELAND, you need to obtain the reference sequences you wish to align against in FASTA format and convert or “squash” them into the format that ELAND can read.

Depending on the application, a reference genome and reference files may be supplied for the read sequences to be aligned against. If not, you can squash reference files by running a program squashGenome that is provided as part of the GERALD installation.
The outcome of the squashing process is a folder containing a set of files that encode the reference sequences in a 2-bits-per-base binary format that is not human readable. Squashing needs to be done only once for each set of reference sequences you are interested in aligning against. For example, if you were doing some mouse and some human sequencing, you might create a folder containing a squashed version of the mouse genome and another folder containing a squashed version of the human genome (you probably do not want to squash both genomes into the same folder). Once created, a squashed folder can be copied between machines or placed on a shared drive, so as to be accessible from multiple machines. You specify the path of this folder as a parameter ELAND_GENOME when creating the configuration file for any analysis that involves ELAND.

The FASTA file format is widely used. Here is an example:

```plaintext
>chromosome:NCBI36:X:1:154913754:1
CTAACCCTAACCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTCTGAAAGTGG
ACCTATCAGCAG
GATGTGGTGGACGAGTAGAGAATAAAAAGCAGACTGCCTGAGC
CAGCAGTGGCAACC
```

Please note that the names of the entries in any FASTA files to be squashed cannot contain spaces. In eland_pair, eland_rna, and eland_extended, the names of the entries cannot contain spaces, commas, or colons.

1. You must first create an empty folder for the squashed files to go into.
   ```bash
   mkdir path/myGenome
   ```

2. Go to the location of the FASTA format reference sequence files and enter the following command:
   ```bash
   /path-to-CASAVA/bin/squashGenome <path>/myGenome fastaFile1.fa [fastaFile2...]
   ```
   where `<path>` denotes the full path to the folder `myGenome` you created in step 1. This will cause files `fastaFile1.fa.2bp` and `fastaFile1.fa.vld` to be created in folder `myGenome`.

Prior to GERALD version 0.3, there was a restriction of a single entry per FASTA file for the reference sequences. For GERALD version 0.3 and later, this restriction has been removed and FASTA files with multiple entries can be squashed. In this case `squashGenome` will create a third file `fastaFile1.fa.idx`

For reasons of efficiency, ELAND thinks of the reference sequence as being in “blocks” of 16 MB, of which there can be at most 240. This limits the total length of DNA that ELAND can match against in a single run.

In a single ELAND run you can match against:
- One file of at most 240 x 16 = 3824 MB
- 239 files, each up to 16 MB in size
- Something in between, such as 24 files of up to 160 MB each. (The NCBI human genome will fit.)

**NOTE**

ELAND does not check whether the directory of squashed genomes it is matching against exceeds these limits.
Each file in the reference sequence must take up at least one block, so if you have a large number of short sequences to align against, you should place them in a single large file as individual FASTA-format entries.

The resulting output is the following:
- fa.2bpb file, containing compressed sequence data.
- fa.vld file, containing the start and end positions.
- fa.idx file, keeping track of where each sequence starts and finishes in the .2bpb file (only generated if there were multiple sequences in the FASTA file).

The squashed genome directory must contain only files produced by the squashGenome program. Any other types of files will result in errors during the search. This prohibition extends to subdirectories as well.
Appendix F

ELANDv2 Algorithm

Topics

140 Introduction
141 ELANDv2 Algorithm Description
   141 Multiseed and Gapped Alignment
   142 Alignment Score Calculation
   143 Rest-of-Genome Correction
   143 Unreported Unique Alignments
Introduction

Efficient Large-Scale Alignment of Nucleotide Databases (ELAND) is very fast and should be used to match a large number of reads against a genome.

ELAND is much faster than PhageAlign but will only detect matches that have a limited number of differences between the matching region and read. This means that ELAND is less sensitive than PhageAlign, which will always find a best match for your reads, although possibly not a unique one. Consider the following points when using ELAND:

- If your reads are noisy, not all of them are going to align, and you may not get good results.
- Error rates based on ELAND output underestimate the true error rate. Since reads with many mismatches do not get aligned, they do not contribute to the calculation.

This section provides a detailed description of the ELANDv2 algorithm.

ELANDv2

As of CASAVA v1.6 a new version of ELAND is available, ELANDv2. The most important improvement of ELANDv2 is its ability to perform multi-seed and gapped alignments. As a consequence, ELANDv2 handles indels and mismatches better.

Input and Output Files

For a detailed description of the input and output files for ELANDv2, see Using ANALYSIS eland_extended on page 39 and GERALD Output Files on page 46.
ELANDv2 Algorithm Description

**Multiseed and Gapped Alignment**

ELANDv2 introduces multiseed and gapped alignments:

- Multiseed alignment works by aligning the first seed of 32 bases and consecutive seeds separately.
- Gapped alignment extends each candidate alignment to the full length of the read, using a gapped alignment method that allows for gaps up to 20 bases.

A 'match descriptor' string in the output file (see *Output File Formats* on page 54) encodes which bases in the read matched the genome and which were mismatches, and reports the gaps using the escape sequence "^..$" (see Table 17 on page 55).

The differences between gapped and ungapped alignments, and singleseed and multiseed alignments are illustrated in Figure 27 and Figure 28.

**Figure 27** Ungapped Versus Gapped Alignment
**Figure 28 Singleseed Versus Multiseed Alignment**

**Singleseed Alignment**
- First 32 base seed
- Seed spanning indel does not align properly
- Extension Alignment: No extension possible

**Multiseed Alignment**
- First 32 base seed
- Second 32 base seed
- Seed spanning indel does not align properly
- Second seed aligns properly

**Alignment Score Calculation**

The base quality values and the positions of the mismatches in a candidate alignment are used to give a probability score (p-value) to each candidate. This is the probability that the candidate position in the genome aligned to would, if its bases were sequenced at error rates that correspond to the read’s quality values, give rise to the observed read. This way the contribution of each base is weighted according to its quality.

**NOTE**

There are other binary versions of ELAND available that use 16 up to 32 bases as seed.

**NOTE**

A consequence of this is that the best alignment does not necessarily have the least number of mismatches, although an exact match will always beat any alignment containing mismatches.

The alignment score of a read is computed from the p-values of the candidate alignments. The candidate with the highest p-value is the best candidate and its alignment score is its p-value as a fraction of the sum of the p-values of all the candidates. This is also known as a Bayes’ Theorem inversion. The alignment score is expressed on the Phred scale, i.e. Q20 corresponds to 1% chance of alignment being wrong, Q30=0.1%, etc.
NOTE
The alignment score of a read and the p-values of the candidate alignments for the read are not the same. The former is computed from the latter.

Rest-of-Genome Correction
If only one candidate alignment is found, the scoring scheme above would give an infinite Phred score. MAQ deals with this by giving such cases an arbitrary high score of 255. ELANDv2 uses constant known as the ‘rest-of-genome correction’ that depends on the average base quality of the read, the read length and the size of the genome. This gives a scoring scheme with the following properties:
- Single-candidate alignments for longer reads will score more highly than single-candidate alignments of shorter reads
- Single-candidate alignments for better quality reads will score more highly than single-candidate alignments of lower quality reads
- Single-candidate alignments to shorter genomes will score more highly than single-candidate alignments to longer genomes

Unreported Unique Alignments
A line in an export file will only contain alignment information if the alignment score for that read exceeds a threshold. The primary purpose of this threshold is to retain only alignments that are markedly better than any other possible alignment for the read.

GERALD reduces alignment quality to a single confidence score and read quality, the number of mismatches in the best alignment, and the presence of other candidate alignments all contribute to the calculation of that score. Therefore, changes in any of these three variables will affect whether the alignment passes the alignment quality threshold. So even if only a single candidate alignment has been found for a read, it may still fail the alignment quality threshold for one of two reasons, and not be reported in Export.txt and Sorted.txt:
- Low base quality values.
- Excessive number of mismatches in the candidate alignment. There will be at most 2 mismatches in the seed but potentially there can be any number of mismatches in the remainder of the read.

For most applications, this is the right thing in both cases. For example, you would not want to use a read with 10 mismatches for SNP calling, even if it is the only candidate found. The same applies for a read of poor base quality.

Gapped Alignment Scoring
Given a read, ELANDv2 determines positions in the genome to which substrings of the read (seeds of length 32 bp) match with at most two errors. We then grab x additional bases before and after the hit position (default value for x is 5) to account for potential gaps in the alignment phase.

We then compute a global alignment between the read and the reference which means that the entire read is aligned to the reference. We are using affine gap penalties: opening a gap is more expensive than prolonging an existing gap. The alignment algorithm is furthermore banded, ie we restrict ourselves to a maximal length of an expected insertion/deletion (this value is set to 20).
Conditions for Opening a Gap

ELANDv2 tries to be conservative about when to open a gap. There are two main conditions that have to be satisfied to open a gap:

1. A gap corrects at least five mismatches downstream: this means that the number of mismatches between the ungapped and the gapped alignment is at least five.

2. We set the number of mismatches in the gapped and ungapped alignment in relation to each other. The reason is that we want to distinguish gaps that improve noisy ungapped alignments and real small insertions/deletions. To this end, we define the _noise ratio_ as 

\[(\text{#(mismatches ungapped.alignment) - #(mismatches gapped.alignment)}) / \text{#(mismatches gapped.alignment)}\].

If the ratio for a given alignment exceeds a certain value (set to 3.1 by default), we insert a gap.

If any of two conditions is not satisfied, we return an ungapped alignment as the result.
Appendix G

Algorithms in Variant Detection and Counting

Topics
146 Introduction
146 SNP Calling Algorithms
146 Allele Calling
147 SNP Calling
147 Indel Finder Algorithm
147 IndelFinder
148 AlignCandidates
149 ClusterFinder
149 SmallAssembler
150 AlignContig
153 Indel Genotype Caller
155 readBases Counting Method
155 readStart Counting Method
155 Advantages of Read Base Counting Method
156 Reference
Introduction

This appendix explains the algorithms used in CASAVA for the following functions:
- SNP calling
- Indel detection
- RNA sequencing counting methods

SNP Calling Algorithms

CASAVA uses two methods to call SNPs. First it calls the alleles based on base calls, alignment, and quality scores, then it calls the SNPs based on the allele calls and read depth. This section explains these methods.

Allele Calling

The base calls and their associated quality values are sent to a Bayesian allele caller, which produces one or two allele calls and scores for each position in the genome. Only read pairs that mapped with the expected insert size (within 3 standard deviations of the median) and orientation are used for allele calling. Paired reads should have a paired read alignment score of at least 6 and single reads have a single read alignment score of at least 10 in order to be used for allele calling. Those reads that do not pass this quality filter are still listed in sorted.txt and can be viewed but their bases do not contribute towards the consensus sequence.

The allele caller computes $-\log_{10}(p(\text{observed base } | \text{no } 'A' \text{'s are present})$ for A and similarly for C, G, Ts at each position. The two highest values are then converted to log-odds scores by subtracting the second-lowest value from each. These are the allele call scores for the two highest scoring alleles. They can be approximately translated as a Phred score divided by 10 (e.g. an allele score of 3 corresponds to a Phred score of 30).

As of CASAVA v1.6, a new mismatch density filter reduces noise in longer gapped reads:
- Base calls are ignored where more than 2 mismatches to the reference sequence occur within 20 bases of the call.
- If the call occurs within the first or last 20 bases of a read then the mismatch limit is applied to the 41 base window at the corresponding end of the read.
- The mismatch limit is applied to the entire read when the read length is 41 or shorter.
SNP Calling

Homozygous SNPs are called at positions where a non-reference allele is observed, the allele call score is ≥10, and the depth is no greater than three times the chromosomal mean. For heterozygous SNP calls, we additionally require the second-highest scoring allele to have an allele-call score ≥6 and the ratio of the highest to second-highest allele-call scores to be ≤3. The allele call score cutoff ensures that more than the equivalent of three q30 bases are needed to make a SNP call; the ratio cut-off helps to distinguish between genuine heterozygous SNPs and any residual background noise, especially for extremely high coverage (e.g. mitochondria in the human genome).

Indel Finder Algorithm

The Indel Finder application uses singleton/shadow read pairs to detect indels. The Indel Finder works in three stages:

1. Compute clusterings of non-aligned 'shadow reads', using as a distance metric the positions of the 'singleton' reads that they pair to.
2. Assemble them into contigs.
3. Align the contigs back to the genome, using the positions of associated 'singleton' reads to narrow the search to a couple of thousand bp or so.

To find indels you can either run the Indel Finder in default mode through CASAVA, or you can run it module by module, which provides you more flexibility to optimize indel finding. The application contains the following modules:

- IndelFinder takes a sorted.txt file from a CASAVA build and extracts 'semi-aligning' and 'shadow' reads.
- AlignCandidates does a dynamic programming alignment of each shadow read, looking within the interval within which it is expected to sit.
- ClusterFinder takes the output of AlignCandidates and tries to group them in clusters of reads that have been caused by the same indel.
- SmallAssembler takes the output of ClusterFinder and assembles clusters of reads into contigs.
- AlignContig does a dynamic programming alignment of contig to genome. This alignment allows for gaps.

This section describes the modules, example runs for running module by module, and the key parameters. Good parameters for indel finding optimization are indicated in the tables.

IndelFinder

The IndelFinder module within the Indel Finder application takes a sorted.txt file from a CASAVA build and extracts:

- Any reads containing gapped alignments
- Any reads whose alignment is much worse than expected given its quality
- Any 'shadow' reads not thought to be due just to poor base quality
It tries to exclude reads for which the bad or non-existent alignment is just a consequence of poor base quality

**IndelFinder Parameters**

The IndelFinder module can be adapted using the following parameters during your CASAVA variant detection run.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-e, --inReadsPath=PATH</td>
<td>Path to sorted.txt file.</td>
</tr>
<tr>
<td>-o, --outputFilePath=PATH</td>
<td>Path to output file.</td>
</tr>
<tr>
<td>-s, --SummaryFilePath</td>
<td>File of insert size stats generated by the IndelFinder and needed downstream.</td>
</tr>
<tr>
<td>--spReadThreshold=NUMBER</td>
<td>Spanning read score threshold. The IndelFinder assigns a spanning read score, and the higher the score, the more unlikely it is to see the pattern of mismatches given the read's quality values. For an alignment with no mismatches this metric has a value of zero. The --spReadThreshold indicates the minimum value at which reads are still used in the indel finding process. The default value for --spReadThreshold is 25. Lowering the value adds more reads into the indel finding process, at the possible expense of introducing noise.</td>
</tr>
<tr>
<td>--prasThreshold=NUMBER</td>
<td>Paired read alignment score threshold. If a read has a paired read alignment score of at least this threshold, then it is used to update the base quality stats for that sample.</td>
</tr>
</tbody>
</table>

**AlignCandidates**

The module AlignCandidates does a dynamic programming alignment of each shadow read, looking in the interval within which it is expected to sit. It takes the output of IndelFinder and does a localized alignment of each read. If this procedure finds an alignment for a read where none existed previously, or finds a better alignment than the existing one, then the previous alignment is replaced.

**AlignCandidates Parameters**

The AlignCandidates module can be adapted using the following parameters during your CASAVA variant detection run.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-e, --inReadsPath=PATH</td>
<td>Path to input file - shadows.txt file as output by IndelFinder</td>
</tr>
<tr>
<td>-o, --outputFilePath=PATH</td>
<td>Path to output file</td>
</tr>
<tr>
<td>--sampleStatsPath=PATH</td>
<td>Location of sample stats file. Can be omitted, in which case uses the start and end ranges provided in each output line from IndelFinder, which are calculated on the basis of plus or minus 3 standard deviations from the mean insert size (the 3 is currently hard coded).</td>
</tr>
<tr>
<td>--score-match=x</td>
<td>set match score in alignment to x (default is 5)</td>
</tr>
<tr>
<td>--score-mismatch=x</td>
<td>set mismatch score in alignment to x (default is 4)</td>
</tr>
<tr>
<td>--score-gap-open=x</td>
<td>set alignment gap opening score to x (default is 12)</td>
</tr>
</tbody>
</table>
Indel Finder Algorithm

ClusterFinder
This takes the output of AlignCandidates (a list of shadow and badly aligning reads) and tries to group them in clusters of reads that are thought to have been caused by the same indel

ClusterFinder Parameters
The ClusterFinder module can be adapted using the following parameters during your CASAVA variant detection run.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--score-gap-extend=x</td>
<td>set alignment gap extension score to x (default is 1)</td>
</tr>
<tr>
<td>--alignScoreThresh=NUMBER</td>
<td>If an alignment score for a read exceeds this threshold then the output file is updated to incorporate this alignment. Otherwise the read’s entry remains as per the input file. Default value is 120. A low value will cause some reads to be wrongly placed (albeit within a small interval).</td>
</tr>
<tr>
<td>--sdFlankWeight=NUMBER</td>
<td>Number of standard deviations to use when defining the genomic interval to align the read to (default: 1).</td>
</tr>
</tbody>
</table>

SmallAssembler
SmallAssembler takes the output of ClusterFinder and assembles clusters of reads into contigs. It uses an approach based on kmer-hashing and a de-Bruijn graph. If a read is successfully assembled into a contig, the read’s alignment details are updated in the output file to describe its position in the contig.

SmallAssembler Parameters
The SmallAssembler module can be adapted using the following parameters during your CASAVA variant detection run.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-e, --inReadsPath=PATH</td>
<td>Path to 'shadow' reads file</td>
</tr>
<tr>
<td>-o, --outputFilePath=PATH</td>
<td>Path to output file</td>
</tr>
<tr>
<td>-s, --summaryFilePath=PATH</td>
<td>Path to input stats file</td>
</tr>
<tr>
<td>--maxDistance=NUMBER</td>
<td>Max distance between group members. Currently the clustering works by taking the midpoint of the start/finish interval of each read, converting each read to a point. Two reads are considered to be in the same cluster if the distance between their midpoints does not exceed this.</td>
</tr>
<tr>
<td>--minGroupSize=NUMBER</td>
<td>Only output clusters if they contain at least this many reads.</td>
</tr>
<tr>
<td>--spReadThreshold=NUMBER</td>
<td>Spanning read score threshold. This is calculated in exactly the same way as --spReadThreshold in IndelFinder. However it is used in an opposite way. Here the point is to find reads with few or no mismatches, which are presumed to arise from repeats and not from indels, and exclude them from the clustering process.</td>
</tr>
</tbody>
</table>
AlignContig does a dynamic programming alignment of contig to genome.

### AlignContig Parameters

The AlignContig module can be adapted using the following parameters during your CASAVA variant detection run.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-r, --refSequence=PATH</td>
<td>Path to reference genome FASTA file (*.fa)</td>
</tr>
<tr>
<td>-c, --contigFile=PATH</td>
<td>Path to assembled contigs FASTA file (*.fa)</td>
</tr>
<tr>
<td>-o, --outputFile=PATH</td>
<td>Path to output file (contigs with alignments)</td>
</tr>
<tr>
<td>--varlingContigsPath=PATH</td>
<td>Path to indel contigs formatted for varling_caller</td>
</tr>
<tr>
<td>--report</td>
<td>Show one variant per line. If left off, shows a BLAST-style full alignment.</td>
</tr>
<tr>
<td></td>
<td>CASAVA expects to see --report format, so only leave it off if running as a</td>
</tr>
<tr>
<td></td>
<td>standalone</td>
</tr>
<tr>
<td>--variant</td>
<td>Don't show contigs that exactly match reference</td>
</tr>
<tr>
<td>--indel-only</td>
<td>Only show contigs containing indels</td>
</tr>
<tr>
<td>--snp-only</td>
<td>Only show contigs containing SNPs (no indels)</td>
</tr>
<tr>
<td>--score-match=x</td>
<td>Set match score in alignment to x (default is 5)</td>
</tr>
<tr>
<td>--score-mismatch=x</td>
<td>Set mismatch score in alignment to x (default is 4)</td>
</tr>
<tr>
<td>--score-gap-open=x</td>
<td>Set alignment gap opening score to x (default is 12)</td>
</tr>
<tr>
<td>--score-gap-extend=x</td>
<td>Set alignment gap extension score to x (default is 1)</td>
</tr>
</tbody>
</table>
Indel Finder Algorithm

CASAVA Software v1.6 User Guide

AlignContig Input and Output Files

Input #1: Contig.fa File

An output example is given below:

```
>contig=6|length=73|group=9|reads=4|start=40005768|end=40005851
AAATATACATAGATTATAAATAATATACACTATAAATAGTATATTTGTAATGTA
>contig=7|length=39|group=10|reads=3|start=40007049|end=40007123
AAAGGTTGCACCAACTTATATGTATTTTCATCAGCGAAG
```

The fields contain the following information:

1. Contig number
2. Length of contig
3. Group number
4. Number of reads
5. Start of contig
6. End of contig
7. Contig sequence

Input #2: Chromosome File

The chromosome file needs to be in FASTA format.

Output #1: align.txt file

An example of the align.txt file is given below:

```
272 6462 c20.fa 42970212 -/A -
    AAAAGAAAAATTTATGCACCTGATTTAAAAACAAACC
    AAAAAAAAAGAAAAAAACAAA 7 267
273 6463 c20.fa 42970212 -/A - TATGCACCTGATTTAAAAACAAACC
    AAAAAAAAAGAAAAAAACAAA 24 207
274 6473 c20.fa 42977855 AA/-- --
    GCCCTGGCAACAAAGGACTGAAAATCCTCTTATCAC
    AAAAAAAGAAAAAAATTTGAAA 8 256
```

The fields contain the following information, tab separated:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--min-score=x</td>
<td>Only show contigs with alignment score at least x (default is 30)</td>
</tr>
<tr>
<td>--min-context=x</td>
<td>Demand at least x exact matching bases either side of variant (default is 0??). The idea here is to ensure an indel has a minimum number of exactly matching bases on either side. Setting this to zero might be good for finding reads which align to breakpoints.</td>
</tr>
</tbody>
</table>
1. Variant number - number of entry in file contig_nr
2. Contig number this variant was obtained from (refers to number in corresponding contig_????_fa file genome
3. Chromosome aligned to start
4. Start position
5. Wild-type/variant sequence. For example, A/- means deletion of A in our data compared to the reference
7. Sequence in the contig upstream of the variant
8. Sequence in the contig downstream of the variant
9. Number of reads the contig was assembled from
10. Alignment score of the alignment of contig to genome

Output file #2: varling_contig_file
Contigs output for use by the Indel Genotype Caller module.
Indel Genotype Caller

The Indel Genotype Caller (varling) is designed to run after the Indel Finder. It does not search for indels or modify the indels predicted by the Indel Finder; rather it considers each indel predicted by the Indel Finder as a "candidate" indel to which a genotype will be assigned.

Indel Genotype Caller uses both the reads which were assembled into contigs by the Indel Finder and those reads from the reference chromosome assembly which intersect the candidate indel’s position on the reference sequence. It then calls the candidate indel as homozygous, heterozygous or not present. It does this by aligning each read to both the reference and indel sequences and determining the degree to which the read in question supports either the reference or indel sequence. It is common for reads which intersect the indel location to support the indel and reference sequences equally well, so the model is designed in such a way that these reads do not affect the genotype call.

The Indel Genotype Caller provides quality scores (Q-scores) of the indel and the most likely genotype for each indel called. The “Q(indel)” score refers to the Phred-scaled probability that no indel (homozygous or heterozygous) exists at the given position.” Q(max_gtype)” refers to the Phred-scaled probability that the indel genotype is not that given as "max_gtype". Phred-scale Q-values are derived from the corresponding probability P by the relationship Q = -10log10(P).

The predicted Q-values reflect only those error conditions which are represented in the genotype calling model, which is not comprehensive. The model accounts for basecalling error, diploid chromosome sampling and an approximation of read mapping error, but does not account for artificial indel signatures arising due to sample preparation, copy number variants, or other phenomena. The Q-score provided by the model should be interpreted with respect to these limitations.

Note that the genotype caller operates on the Indel Finder’s contigs rather than on each individual insertion or deletion event, so the call may report sequences of very close indels as single “complex” indel events which have one of the above genotypes. The caller will only consider one of a set of directly overlapping indels predicted by the Indel Finder and does not account for adjacent indel predictions made by the Indel Finder.

The script finish_variant_contigs.py reorders the contigs and the reads used to assemble those contigs by the 5’-most indel breakpoint in each contig. It also “condenses” contigs with exactly matching indels into single contigs.

Indel Genotype Caller Parameters

Indel genotype calling is implemented by the program "varling_caller". It can be adapted using the following parameters during your CASAVA variant detection run.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--sorted file</td>
<td>Analyze reads from ‘file’ in sorted export format (use &quot;-&quot; for stdin)</td>
</tr>
<tr>
<td>--indel-diploid x</td>
<td>Use Bayesian diploid genotype indel caller with heterozygosity=x</td>
</tr>
</tbody>
</table>
Output Files

1. Indel_genotype_calls_file—file containing the calls for each indel as well as the summary information for that call (such as the number of reads supporting the indel and reference alignments). The output file is in the same format as the filtered chromosome-level 'indels.txt' file which is detailed in Indels.txt Files on page 84.

2. Stdout—varling_caller prints auditing information to the stdout stream.
RNA Sequencing Counting Methods

CASAVA employs two different counting methods for RNA sequencing reads. These are explained in this section.

### readBases Counting Method

This method is for exon, splice junction and gene counts. CASAVA will count the number of bases (not the number of reads), that belong to exons, and genes. For splice junctions, the number of reads that cover the junction point is counted. The number of bases that fall into the exonic regions of each gene is summed to obtain gene level counts. The normalized values are calculated as RPKM (Reads Per KiloBase per Million of mapped reads). Since the base counts rather than read counts are used, the RPKM for exons and genes is calculated slightly differently than RPKM for splice junctions.

The normalized values for genes and exons are counted as follows:

\[
\text{Exons/genes RPKM} = 10^9 \times \frac{Cb}{NbL}
\]

With:

- \( RPKM \) = Reads Per Kilobase of exon model per Million mapped reads
- \( Cb \) = the number of bases that fall on the feature
- \( Nb \) = total number of mapped bases in the experiment
- \( L \) = the length of the feature in base pairs

The normalized values for splice junctions are counted as follows:

\[
\text{Splice junctions RPKM} = 10^9 \times \frac{Cr}{NrL}
\]

With:

- \( Cr \) = the number of reads that cover the junction point
- \( Nr \) = total number of mapped reads in the experiment
- \( L \) = the length of the feature in base pairs.

Only the reads with alignment score \( \geq \text{QVCutoffSingle} \) are considered.

Exons that have overlapping exons from other genes on the forward or reverse strand are excluded from counting and are also not included to compute the total gene length.

### readStart Counting Method

This method is for exons, splice junctions and genes counts, and is kept unchanged from CASAVA v1.0 for validation and backward compatibility.

By default CASAVA will only count the first base of each read. The number of reads that fall into the exonic regions of each gene is summed to obtain gene level counts. These counts are then divided by the length of the gene to obtain the length-normalized gene counts.

The current sample prep protocol does not keep strand information. Exons that have overlapping exons from other genes on the forward or reverse strand are excluded from counting and are also not included to compute the total gene length.

### Advantages of Read Base Counting Method

The readBases counting method has the following advantages:

- The expression values are more precise as the spliced reads contribute to both exons in proportion to the number of bases that fall on each exon.
The standard RPKM formula is used for normalized values (the readStart method does not normalize for analysis depth).

The spliced read bases are properly accounted for during the allele calling.

# Index

## A

- abundant sequences 43
- alignment analysis 3
- All.htm file 51
- allele calling 67
- analysis output 5
  - error rates 51, 52
  - file formats 54
  - file naming 52
  - lane averages 51
  - proportion of reads 52
  - tile-by-tile 51
- analysis parameters 134
- ANALYSIS variables 34

## B

- base calling 3
- Basecalls directory 13
- BaseCalls folder 98
- BustardSummary.xml file 28

## C

- CASAVA 5, 6, 64
  - build 81
  - build directory 79
  - build web page 80
  - counting method 68
  - getting data files 131
  - human reference files 130
  - installing 90
  - variant detection and counting 64
- CASAVA software 5, 6, 64
  - clusters passing filters 52, 59, 60
  - clusters per tile 58
  - config.template.txt file 21
  - config.txt file 31, 38
  - config.xml 104, 105
  - base calling folder 98
  - Data folder 98
  - configuring multiple runs 77
  - contaminants 43
  - Count.txt files 85
  - counting 6, 64
  - configuring multiple runs 77
  - examples 78
  - options 73
  - output files 79
  - results directory 78
  - running 73
- customer support 8

## D

- data folder 98
- default 34
- demultiplex.pl
  - behavior 15
  - demultiplexing 3, 5, 13
  - example 17
  - options 17
  - output 19
  - output directories 100
  - running 17
- DNA sequencing
  - large genome 64
  - small genome 64
- documentation 8
- duplicate removal 67

## E

- ELAND 5, 24
  - algorithm 140
  - analysis modes 34
  - error rates 52
  - eland_extended 34, 39
  - ELAND_MAX_MATCHES 40
  - eland_pair 34, 40
  - eland_rna 34, 43
  - getting data files 131
  - human reference files 127
  - ELAND_SEED_LENGTH 40
  - ELANDv2 25
  - email reporting 90
  - error rates 6, 25, 51, 52, 60, 140
  - Error.htm file 52
catalog # SY-960-1601
part # 15009919 rev. A
naming 102
structure 97
run quality 57
run.conf.xml file 71
runReport.pl script 90

S
SAMPLE 21
SamplesDirectories.csv 19, 100
SampleSheet.csv file 14
scarf 33
SCS real time analysis 3
sequence 35
sequence alignment 3
sequence alignments 24
sequence files 132
SEQUENCE_FORMAT 33
sequence_pair 35
SNP calling 67
Snp.txt files 83
splice junctions 43
squashGenome 136
standard deviations 60
Summary.htm file 57

T
technical assistance 8
tile variability 61

U
USE_BASES 35

V
variant detection 6, 64
configuring multiple runs 77
examples 78
input files 70
options 73
output files 79
results directory 78
running 73
Variant Detection and Counting 6

W
What's New 7