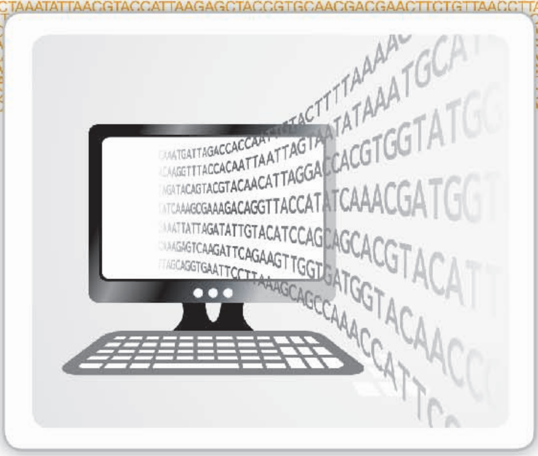
illumina®

CASAVA v1.8
User Guide



FOR RESEARCH USE ONLY

ILLUMINA PROPRIETARY
Part # 15011196 Rev B
May 2011

This document and its contents are proprietary to Illumina, Inc. and its affiliates ("Illumina"), and are intended solely for the contractual use of its customer in connection with the use of the product(s) described herein and for no other purpose. 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. Illumina does not convey any license under its patent, trademark, copyright, or common-law rights nor similar rights of any third parties by this document.

The instructions in this document must be strictly and explicitly followed by qualified and properly trained personnel in order to ensure the proper and safe use of the product(s) described herein. All of the contents of this document must be fully read and understood prior to using such product(s).

FAILURE TO COMPLETELY READ AND EXPLICITLY FOLLOW ALL OF THE INSTRUCTIONS CONTAINED HEREIN MAY RESULT IN DAMAGE TO THE PRODUCT(S), INJURY TO PERSONS, INCLUDING TO USERS OR OTHERS, AND DAMAGE TO OTHER PROPERTY.

ILLUMINA DOES NOT ASSUME ANY LIABILITY ARISING OUT OF THE IMPROPER USE OF THE PRODUCT(S) DESCRIBED HEREIN (INCLUDING PARTS THEREOF OR SOFTWARE) OR ANY USE OF SUCH PRODUCT(S) OUTSIDE THE SCOPE OF THE EXPRESS WRITTEN LICENSES OR PERMISSIONS GRANTED BY ILLUMINA IN CONNECTION WITH CUSTOMER'S ACQUISITION OF SUCH PRODUCT(S).

FOR RESEARCH USE ONLY

Revision History

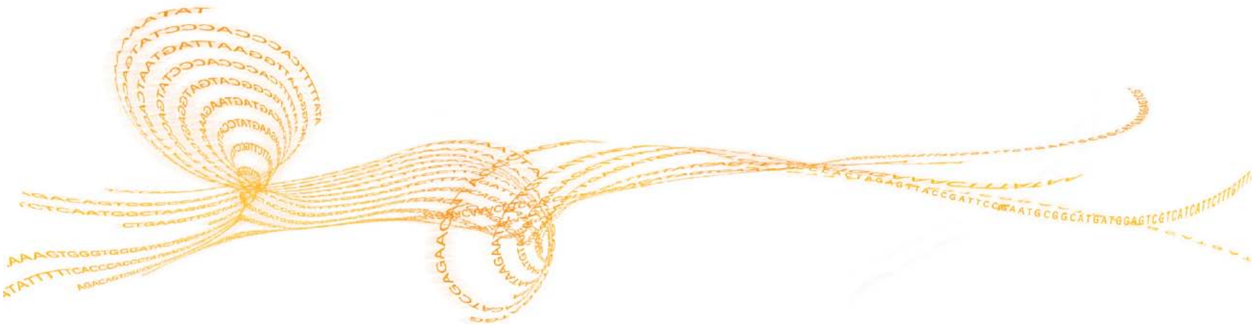
Part #	Revision	Date	Description of Change
15011196	B	May 2011	
15011196	A	March 2010	
1509919	A	November 2009	

Table of Contents

List of Tables

Chapter 1

Overview



Introduction

This user guide documents CASAVA 1.8 (short for "Consensus Assessment of Sequence And VAriation"). CASAVA is the part of Illumina's sequencing analysis software that performs alignment of a sequencing run to a reference genome and subsequent variant analysis and read counting. The basic pieces of functionality of Illumina's sequencing analysis cascade are described below.

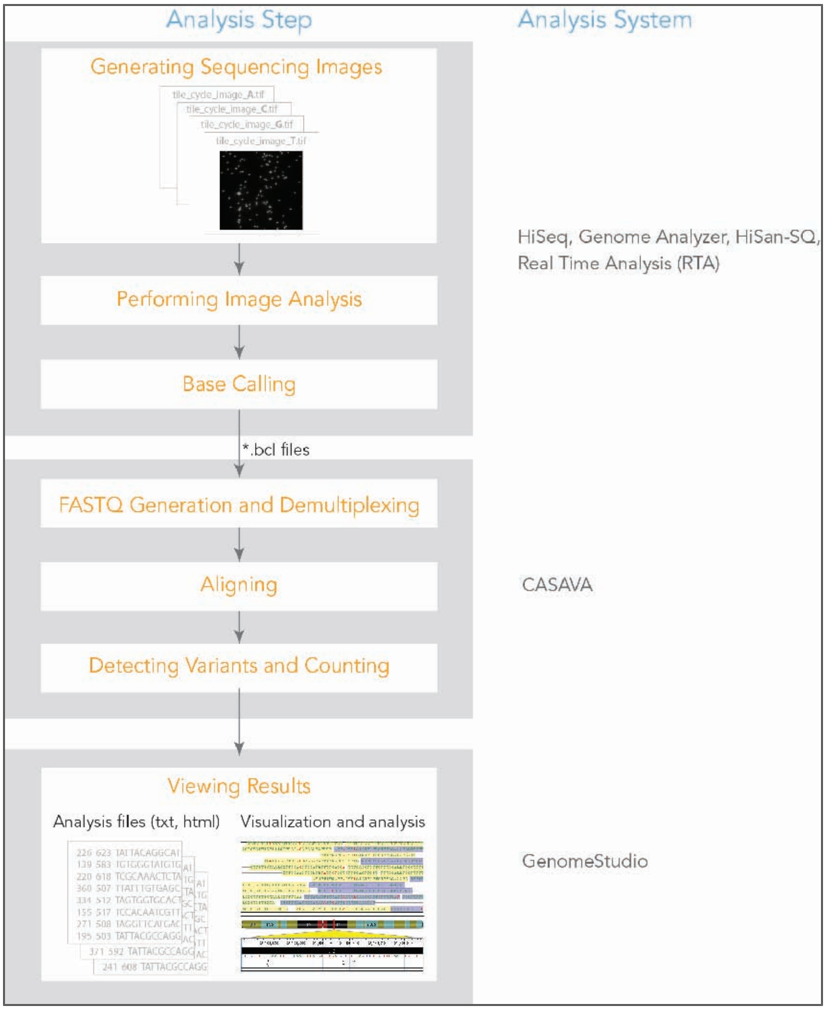
Analysis of Sequencing Data

After the sequencing platform generates the sequencing images, the data are analyzed in five steps: image analysis, base calling, bcl conversion, sequence alignment, and variant analysis and counting. CASAVA performs the bcl conversion, sequence alignment, and variant analysis and counting steps, demultiplexes multiplexed samples during the bcl conversion step.

1. **Image analysis**—Uses the raw images to locate clusters, 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 performed by the instrument control software.
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 performed by the instrument control software's Real Time Analysis (RTA) or the Off-Line Basecaller (OLB).
3. **Bcl conversion**—Converts *.bcl files into *.fastq.gz files (compressed FASTQ files) in CASAVA. Multiplexed samples are demultiplexed during this step.
4. **Sequence alignment**—Aligns samples to a reference sequence using the compressed FASTQ files.
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.

Figure 1 Sequencing Data Analysis Workflow


Analysis Step
Analysis System
Generating Sequencing Images
tile_cycle_image_A.tif
tile_cycle_image_C.tif
tile_cycle_image_G.tif
tile_cycle_image_T.tif
Performing Image Analysis
Base Calling
HiSeq, Genome Analyzer, HiSan-SQ, Real Time Analysis (RTA)
*.bcl files
FASTQ Generation and Demultiplexing
Aligning
Detecting Variants and Counting
CASAVA
Viewing Results
Analysis files (txt, html)
Visualization and analysis
GenomeStudio

Default Analysis Workflow

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

- HiSeq Control Software (HCS) 1.3 and Real Time Analysis (RTA) 1.10, or Genome Analyzer's Sequencing Control Software (SCS) 2.9 and RTA 1.9. The instrument computer running this software performs the following in real time:
  - Image analysis
  - Base calling
- CASAVA 1.8, running on a Linux analysis server, performs:
  - Bcl conversion and demultiplexing
  - Off-line sequence alignment
  - SNP calling and indel detection, read counting (for RNA sequencing)

NOTE
As of 1.8, CASAVA uses *.bcl as primary input, and does not support the _ qseq.txt format. For *_qseq.txt files, use an older version of CASAVA, or convert the _qseq.txt format as described in *Qseq Conversion* on page 159.

Supporting Software

There are a number of software applications that support CASAVA:

- The Off-Line Base caller (OLB) is an alternative for the on-instrument base calling by RTA.
- The Analysis Visual Controller (AVC) provides a GUI interface for running CASAVA, and is especially convenient for users not proficient with running applications through the Linux command line.
- GenomeStudio contains modules for viewing the data analysis results in the genomic context. These modules are the GenomeStudio ChIP Sequencing Module, DNA Sequencing Module, and RNA Sequencing Module.
- The Sequencing Analysis Viewer (SAV) allows you to view primary analysis metrics from the sequencing instrument.

To download these applications and their documentation, go to http://www.illumina.com or https://icom.illumina.com.

CASAVA Features

The CASAVA 1.8 package processes sequencing reads provided by RTA or OLB. CASAVA can generate the following data:

- Sample-specific reads from multiplexed flow cells
- Aligned reads
- SNP calls
- Indel calls
- Expression levels for exons, genes and splice junctions in the RNA Sequencing analysis

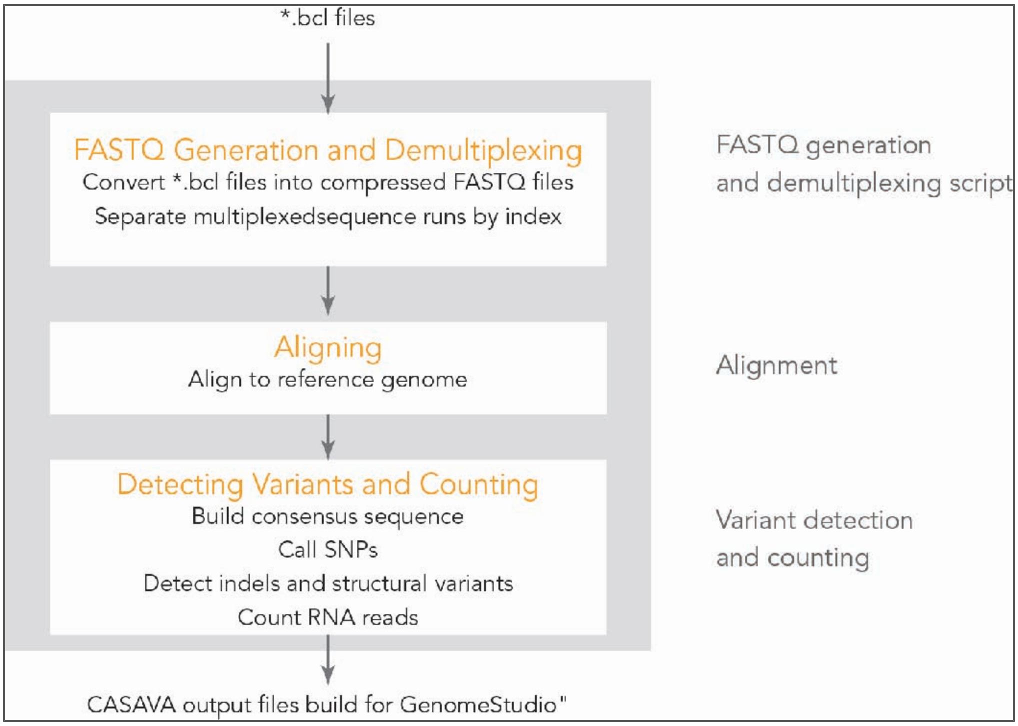
In addition, CASAVA automatically generates a range of statistics, such as mean depth and percentage chromosome coverage, to enable comparison with previous builds or other samples.

CASAVA analyzes sequencing reads in three stages:

- FASTQ file generation and demultiplexing
- Alignment to a reference genome
- Variant detection and counting

These three stages are explained below.

Figure 2 CASAVA Workflow


*.bcl files
FASTQ Generation and Demultiplexing
Convert *.bcl files into compressed FASTQ files
Separate multiplexedsequence runs by index
Aligning
Align to reference genome
Detecting Variants and Counting
Build consensus sequence
Call SNPs
Detect indels and structural variants
Count RNA reads
CASAVA output files build for GenomeStudio"
FASTQ generation
and demultiplexing script
Alignment
Variant detection
and counting

Bcl Conversion

CASAVA 1.8 uses *.bcl files as primary sequence input. The first step, bcl conversion, performs the following:

- Generates compressed FASTQ files that can be used by configureAlignment.

- Organizes the output in Project and Sample folders (based on the sample sheet, if provided).
- Demultiplexes samples into that same run folder organization, based on the sample sheet.

NOTE
The separate demultiplexing step present in CASAVA 1.7 has been integrated in the bcl conversion step.

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 runs from SCS 2.4 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..

Aligning Reads

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

The output data produced by configureAlignment are stored in a hierarchical folder structure called the run folder. The run folder includes all data folders generated from the sequencing platform and the data analysis software.

For the alignment step, the standard input files for reads are the compressed FASTQ files (<sample name>_<barcode sequence>_L<lane>_R<read number>.<0-padded 3-digit set number>.fastq.gz). The standard output files for reads are the export files (<sample name>_<barcode sequence>_L<lane>_R<read number>.<0-padded 3-digit set number>_export.gz).

Alignment Algorithms

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

ELAND has been improved a number of times:

- CASAVA 1.6 introduced a new version of ELAND, ELANDv2. The most important improvements of ELANDv2 are its ability to perform multiseed and gapped alignments.
- As of CASAVA 1.8 a new version of ELANDv2 is available, ELANDv2e. The most important improvements of ELANDv2e are improved repeat resolution and implementation of orphan alignment.

A short description of these improvements is provided below; more information about ELANDv2 is available in *Algorithm Descriptions* on page 129.

Multiseed and Gapped Alignment

ELANDv2e performs multiseed alignment by aligning consecutive sets of 16 to 32 bases separately. After this, ELANDv2e extends each candidate alignment to the full length of the read, using a gapped alignment method that allows for gaps (indels) of up to 10 bases. ELANDv2e then picks the best alignment based on alignment scores.

Repeat Resolution

ELANDv2e aligns reads in repeat regions using two new modes: semi-repeat resolution and full repeat resolution. Both modes take also repetitive hits into account for the multiseed pass of ELAND. Full repeat resolution is more sensitive and places more reads in repeat regions, but will result in longer run time.

By default, ELANDv2e runs in semi-repeat resolution mode. Full repeat resolution can be turned on with the option INCREASED_SENSITIVITY.

Orphan Alignment

ELANDv2e performs orphan alignment by identifying read pairs for which only one of the reads aligns. ELANDv2e tries to align the other read in a defined window (by default 450 bp). If the number of mismatches is <10% of the read length, ELANDv2e reports the 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 configureAlignment.

The CASAVA build process is divided into several modules (or targets), each of which completes a major portion of the post-alignment analysis pipeline:

1. The first module, 'sort', bins aligned reads into separate regions of the reference genome, sorts these reads by alignment position and optionally removes PCR duplicates (for paired-end reads) and finally converts these reads into BAM format.
2. In a paired-end analysis the next module, 'assembleIndels', is used to search for clusters of poorly aligned and anomalous reads. These clusters of reads are de-novo assembled into contigs which are aligned back to the reference to produce candidate indels.
3. Subsequently, the 'callSmallVariants' module uses the sorted BAM files and the candidate indels predicted by the assembleIndels module to perform local read realignment and genotype SNPs and indels under a diploid model.
4. In an RNA-Seq build the 'rnaCounts' module will also be run to calculate gene and exon counts. Other optional modules can be added to the build process to perform additional functions.

For the variant discovery and counting step, the standard input file format for reads is the export format (<sample name>_<barcode sequence>_L<lane>_R<read number>.<0-padded 3-digit set number>_export.gz). The standard output file format for reads is the BAM format. The sorted.bam files are stored in chromosome-specific directories under the output directory.

Use and properties of CASAVA's post-alignment modules are explained in *Variant Detection and Counting* on page 85. More information about the algorithms is available in *Variant Detection* on page 139.

Capabilities and Limitations

This section explains the capabilities and limitations of CASAVA when performing data analysis.

Demultiplexing

Demultiplexing is required for downstream analysis when a run is indexed. Demultiplexing processes the read data so that the indexing read or barcodes are parsed and removed. The reads are segregated and copied into separate directories.

Alignment

Alignment is controlled by the configureAlignment.pl wrapper script, which includes several analysis modes that initiate single-end (eland_extended), paired-end (eland_pair), and single-end RNA (eland_rna) analysis. The default behavior of configureAlignment.pl is to perform a multi-seeded, gapped alignment. This allows for the identification of small indels (<= 10 nt) during alignment; a gap of up to 10 bases can be opened during seed extension.

- **DNA**: The eland_extended and eland_pair analysis modes can be used to align reads to a genome. The types of experiments supported include genome resequencing, exome-capture, targeted capture, and ChIP-Seq data.
- **Methylation**: There is currently no support for aligning Bisulfite-Seq data with Eland.
- **RNA**:Eland_rna will align transcriptome data. Transcript data is limited to single reads that cross at most one splice junction. Eland_rna cannot align paired-end data. For paired-end read transcriptome data, it is recommended that a third party tool such as BowTie/TopHat be used.

Variant Analysis

Variant analysis and RNA counting are controlled by the configureBuild.pl script. The script can be used to describe the following types of variation:

- **Site genotypes and SNPs**: Homozygous and heterozygous single nucleotide variants (SNPs) are called using a Bayesian site genotyping model, which takes into account base calls, quality scores, and alignment scores of the reads at the given position.
- **Indels**: Indels are called using a two-stage process. First, contigs are assembled from poorly aligned/anomolous reads and aligned back to the reference genome to produce indel candidates, and then the variant caller consolidates these candidates, performs local realignment, and genotypes the indel. Indels of up to 300 bases in length can be genotyped using this process. Small indels (up to 10 bases) can be detected directly from the gapped alignment.
- **RNA counting**: The number of bases that fall into the exonic regions of each gene are summed to obtain gene level counts, normalized according to feature size, and expressed as RPKM (Reads Per Kilobase per Million of mapped reads). Only splice sites from known splice variants are reported, one at a time. If a read represents a new splice variant or spans multiple splice junctions it will not be counted.

What's New

Important Changes in CASAVA 1.8

Bcl Conversion and Demultiplexing

- Uses *.bcl files as input which are converted into the compressed FASTQ format.
- No direct support for *_qseq.txt files.
- Demultiplexing performed during the BCL to FASTQ conversion step.
- Output directory structure divided in Project and Sample folders, based on the sample sheet.
- Script renamed to configureBclToFastq.pl
- A Qseq Converter has been added, which converts *_qseq.txt files into FASTQ files for configureAlignment.
- FASTQ files use the standard 33 offset in ASCII Qscore encoding.

Alignment

- Implementation of ELANDv2e, which provides better alignment in repeat regions.
- Script renamed to configureAlignment.pl
- Supports multi-FASTA reference files. Reference genome files should be provided in the FASTA format.
- Code changes to increase ELAND performance.
- Uses FASTQ input files instead of *_qseq.txt files.
- Align as you go: CASAVA 1.8 can start analysis as soon as the first read has been sequenced completely.

Variant Detection and Counting

- Improved indel and SNP calling.
- Script renamed to configureBuild.pl and configureRnaBuild.pl
- Archival option to preserve all data in post-alignment BAM files.
- Indel assembler has much higher sensitivity to large insertions and deletions.
- Indel caller can now handle complex and overlapping indels.
- Indels can be called directly from gapped alignments, in addition to realigned orphan contigs.
- Indels can be called from RNA-Sequencing builds.
- Indel output now includes genotyping of "open-ended" breakpoints.
- Local read realignment is now used to improve SNP and indel calls.
- New SNP-caller with significantly improved genotyping accuracy.

For more information, see the Release Notes for CASAVA 1.8, or the Changes file in {CASAVAInstallationDirectory}/share/CASAVA-1.8.0.

New Options

The new options for release 1.8 are listed below.

Bcl Conversion and Demultiplexing

For descriptions, see *Options for Bcl Conversion and Demultiplexing* on page 35.

```
--fastq-cluster-count
```

```
--positions-dir
--positions-format
--filter-dir
--use-bases-mask
--no-eamss
```

makefile variable (for descriptions, see *Makefile Options for Bcl Conversion and Demultiplexing* on page 36.):

```
r1
POST_RUN_COMMAND_R1
```

Alignment

Optional parameters (for a description, see *Config File Parameter List* on page 55.):

```
INCREASED_SENSITIVITY
ELAND_FASTQ_FILES_PER_PROCESS
ELAND_RNA_GENOME_ANNOTATION
ELAND_RNA_GENOME_CONTAM
ELAND_RNA_GENE_MD_GROUP_LABEL
```

makefile variable (for a description, see *Starting Alignment for Read 1* on page 63):

```
r1
```

Variant Detection and Counting

Options on page 95

Workflow options:

```
--variantsSkipContigs
--variantsNoSitesFiles
--variantsNoReadTrim
--variantsWriteRealigned
--sortKeepAllReads
```

Read mapping options:

```
--variantsSEMapScoreRescue
--variantsIncludeSingleton
--variantsIncludeAnomalous
--ignoreUnanchored
```

SNP and indel options:

```
--variantsNoCovCutoff
```

SNP options:

```
--variantsSnpTheta=FLOAT
--variantsSnpCovCutoff=FLOAT
--variantsSnpCovCutoffAll
--variantsMDFilterCount=INTEGER
--variantsMDFilterFlank=INTEGER
--variantsIndependentErrorModel
--variantsMinQbasecall=INTEGER
--variantsSummaryMinQsnp=INTEGER
```

Indel options:

```
--variantsIndelTheta=FLOAT
--variantsIndelCovCutoff=FLOAT
--variantsCanIndelMin=INTEGER
```

```
--variantsCanIndelMinFrac=FLOAT
--variantsSmallCanIndelMinFrac=FLOAT
--variantsIndelErrorRate=FLOAT
--variantsSummaryMinQindel=INTEGER
--variantsMaxIndelSize=INTEGER
```

RNA-specific options

```
--seqGeneMdGroupLabel
```

Targets

For descriptions, see *Targets* on page 94.

```
assembleIndels
callSmallVariants
gsIndex
```

Qseq Converter

For descriptions, see *Qseq Converter Parameters* on page 164.

```
--input-dir
--output-dir
--fastq-cluster-count
--config-file
--summary-file
--flowcell-id
```

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 24.

Provide a description of the error / bug / feature, along with the following information, if available:

Demultiplexing/Bcl Conversion

- The configureBclToFastq.pl command-line
- Nohup.out from the `make` execution
- SampleSheet.csv
- support.txt file in the Unaligned folder

Alignment

- The configureAlignment.pl command-line
- Nohup.out from the `make` execution
- Config.txt
- support.txt file in the Aligned folder

Variant Detection/Counting

- The command-line
- CASAVA.log
- conf/project.conf

Chapter 2

Interpretation of Run Quality

Introduction

Before beginning a secondary analysis, you should do an assessment of a sequencing run's performance metrics. This can help reveal any issues which may affect the secondary analysis.

To assess a run, you can use either the Status.htm file from the Run\Data directory, or the Sequence Analysis Viewer (SAV). The SAV is an Illumina software package available on iCom which can be used to view the performance metrics of a sequencing run.

In general, using a PhiX or other balanced, suitable control sample (such as human genomic DNA sequencing) as guide helps when interpreting these graphs.

Quality Tables and Graphs

Before beginning an analysis run, you should check the following tables and graphs in status.htm or SAV:

- **Run Info**: You can view basic information on the run's configuration, read length, and control specifications on the Run Info tab of the Status.htm output or the Summary tab of the SAV window.
- **Data by Cycle**: These graphs help you examine intensities, focus metrics (FWHM), percent base, qscores, error rates and other metrics per cycle and per lane. You can identify sample properties or instrument related events that affect the data.
- **Data by Tile Charts**: These graphics show run metrics by cycle and by lane and tile. These can be used to identify any issues which may be specific to a certain lane, or group of tiles.
- **Cluster Density Box Plots**: These plots show the raw cluster densities per lane, and the clusters passing filter.

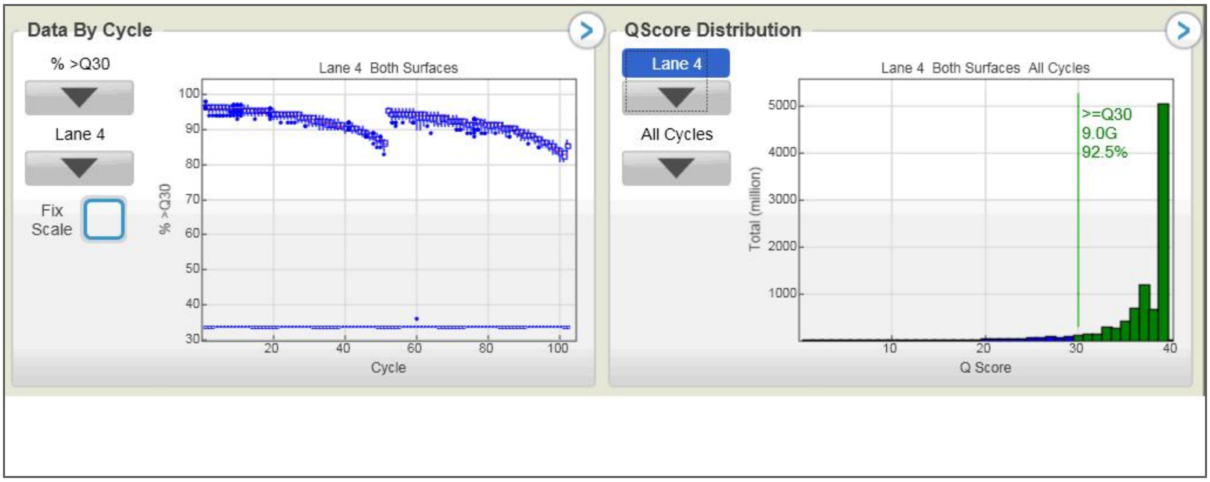
NOTE
Many of the run quality metrics are depicted as box plots. In these plots, the red line shows the median, the box delimits the middle 50% of the data (interquartile range), and the error bars indicate the sample minimum and maximum.

The sections below describe a number of examples of good runs and bad runs.

Excellent Quality Metrics

The figure below shows a screen shot from SAV displaying a run with excellent quality metrics. Note the trend of high Q-scores (%>Q30) across each cycle (left side) and the cumulative distribution of %>Q30 among the reads (right side).

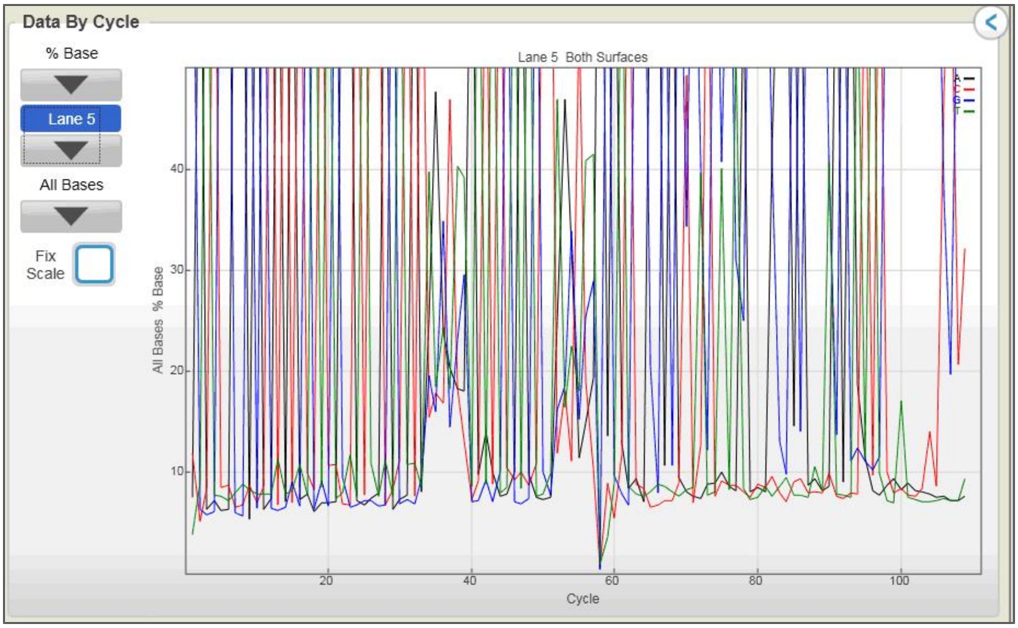
Figure 3 SAV Screenshot Showing Excellent Quality Metrics



Low Diversity Samples

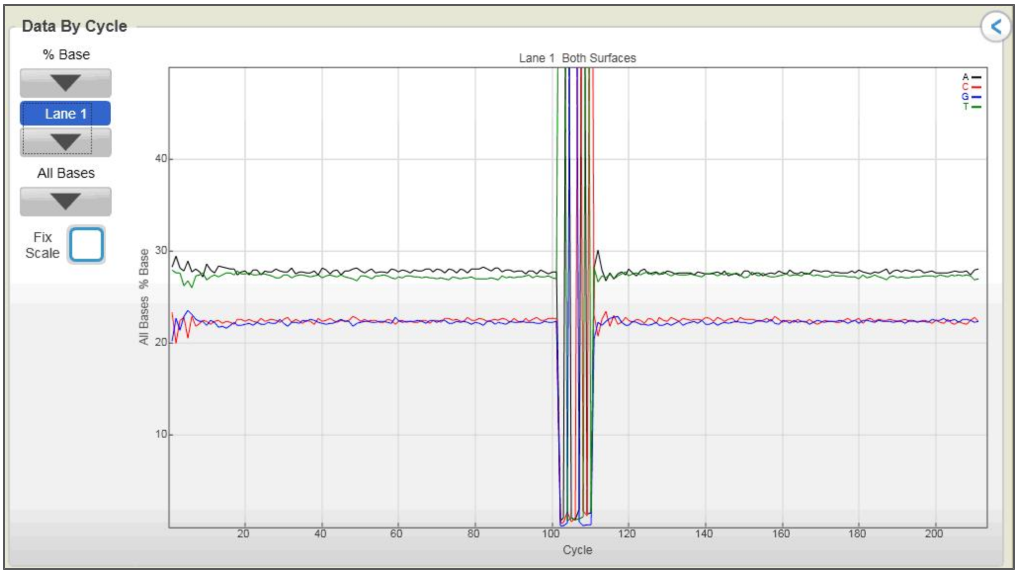
The figure below shows a screen shot from SAV displaying the percent base per cycle for a low diversity sample, which might result from sequencing a small number of PCR artifacts.

Figure 4 Low Diversity Samples



In contrast, the figure below shows the percent base per cycle graph for a more diverse sample. Note the low diversity for cycles 102-109: this was a multiplexed sample and these are the index read cycles, so this is normal.

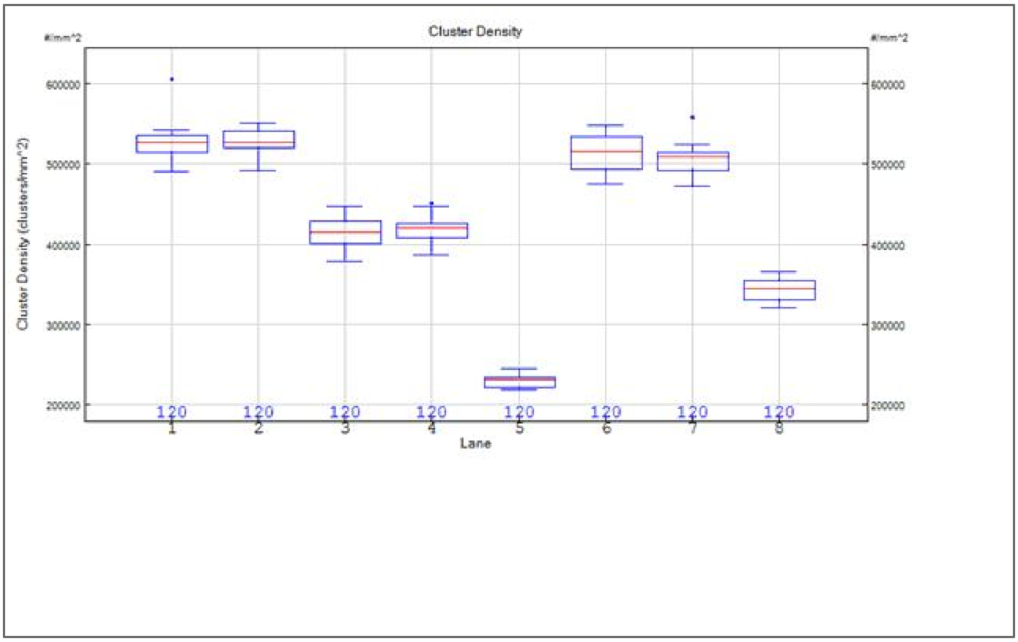
Figure 5 Proper Diversity Samples



Cluster Density

The figure below shows a screen shot from SAV displaying raw cluster densities for lanes 1-8 of a flow cell. Note the very low cluster density of lane 5: if this is set as the control lane for the run, you might need to repeat basecalling (using OLB) with a more successful control lane.

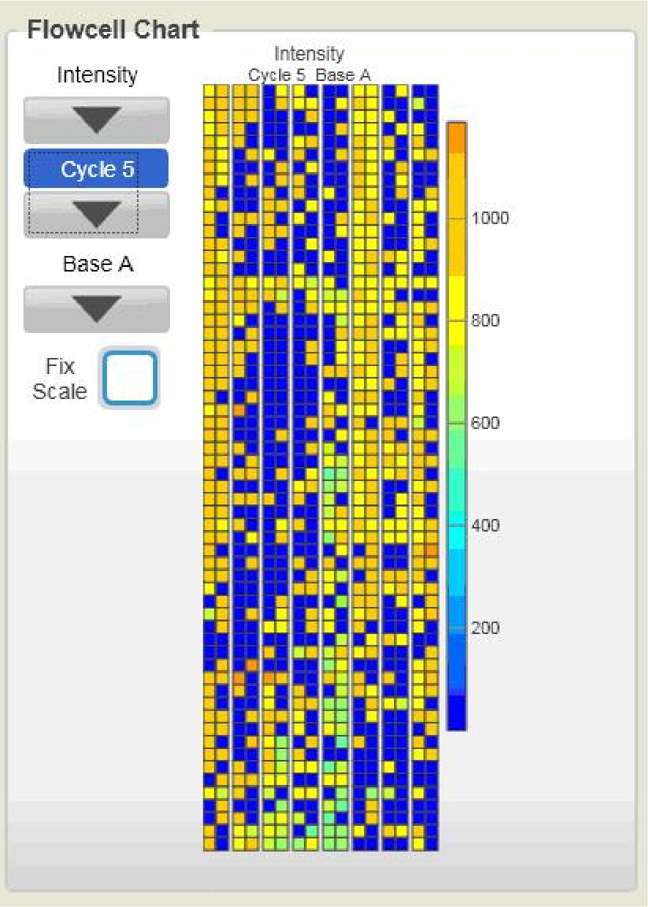
Figure 6 Cluster Density



Fluidics Leak

The figure below depicts a flow cell with spatial variability in intensity. Typically, we would expect intensity to be nearly even within each lane. This variability might indicate a fluidics issue such as a large volume of bubbles moving through the flow cell

Figure 7 Fluidics Leak



SummaryTab

Another tab in the status.htm page or SAV that you should examine is the Summary tab. The key parameters 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.

Condition	Possible Cause	Suggested Action
Fewer clusters than expected:		Reanalyze with new default offsets in OLB. You will need *.cif files for that.
Few bright clusters on the flow cell	Problem with cluster formation	
Blurred images	Poor focus or dirty flow cell surface	
Lots of clusters visible	Cluster density or size is too great to distinguish individual objects	
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.

Condition	Possible Cause
Low intensity	Problem with cluster formation or poor focus

Percentage of First Cycle Intensity Remaining After 20 Cycles of Sequencing

Generally, the higher, the better. The intensity remaining can be sample dependent.

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

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 calculated as Chastity = Highest_Intensity / (Highest_Intensity + Next_Highest_Intensity) for each cycle. The new default filtering implemented at the base calling stage allows at most one cycle that is less than the Chastity threshold.

The higher the value, the better. This value is very dependent on cluster density, since the major cause of an impure signal in the early cycles is the presence of another cluster within a few micrometers.

Condition	Possible Cause	Suggested Action
Very few clusters passing filter	Poor flow cell, perhaps unblocked DNA Faint clusters Out of focus Poor matrix A fluidics or sequencing failure Bubbles in individual tiles Too many clusters Large clusters High phasing or prephasing	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.

Percentage of Phasing and Prephasing

Ideally, these values should be as low as possible.

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.

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
High standard deviations	Check poor tiles for: • Bubbles • Focus • Dirty flow cell surface	Look at the tile-by-tile statistics that appear below the flow cell-wide summary.

After reviewing the tables in Summary.htm, examine the thumbnails.

Chapter 3

Bcl Conversion and Demultiplexing

Introduction

As of CASAVA 1.8, configureAlignment uses FASTQ files as input. Since Illumina sequencing instruments generate *.bcl files as primary sequencing output, CASAVA contains a BCL to FASTQ converter that combines these per-cycle *.bcl files from a run and translates them into FASTQ files.

NOTE
As of 1.8, CASAVA uses *.bcl as primary input, and does not support the _qseq.txt format. For *_qseq.txt files, use an older version of CASAVA, or convert the _qseq.txt format as described in *Qseq Conversion* on page 159.

CASAVA 1.8 can start with bcl conversion and alignment as soon as the first read has been sequenced completely.

In addition to generating FASTQ files, CASAVA uses a user-created sample sheet to divide the run output in projects and samples, and stores these in separate directories. Each directory can be independently analyzed (alignment, variant analysis, and counting) with CASAVA.

NOTE
Some of the files needed for the alignment are at the top level of the Unaligned directory.

contains the files necessary for alignment, variant analysis, and counting with CASAVA.

At the same time, CASAVA also separates multiplexed samples (demultiplexing). 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. The multiplexed samples are assigned to projects and samples based on the sample sheet, and stored in corresponding project and sample directories as described above.

WARNING
The CASAVA 1.8 directory organization differs considerably from the directory organization used in CASAVA 1.7.

NOTE
You cannot start Bcl conversion, demultiplexing, and alignment in one step using CASAVA.

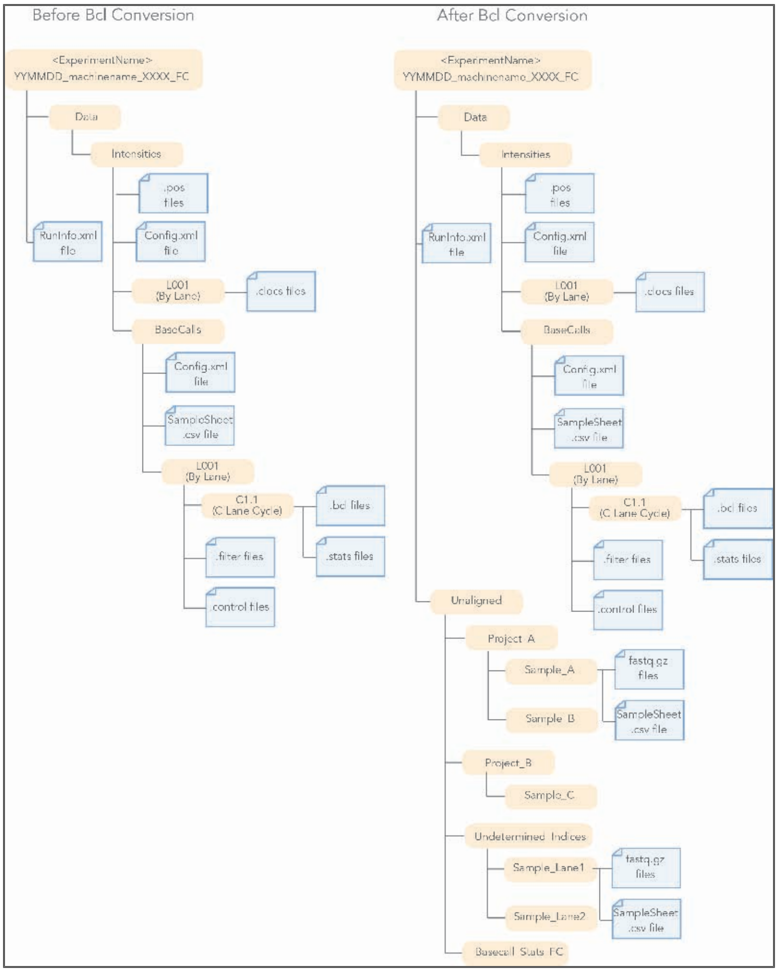
Bcl Conversion/Demultiplexing Directory Structure

Bcl conversion and demultiplexing is done in a single step, and generates a new directory in the Run folder called **Unaligned**, which contains all of the demultiplexed compressed FASTQ files. One level down from the Unaligned directory are the **project directories** and within each project directory are the **sample directories**.

Reads with undetermined index will be placed in the directory Undetermined_indices, unless the sample sheet specifies a specific sample and project for reads without index in that lane.

NOTE
CASAVA 1.8 introduces samples and projects as organizing principle, which differs from CASAVA 1.7, which organized output by lanes or index.

Figure 8 Typical Run Folder Structure after Bcl Conversion and Demultiplexing



Sample Sheet

The sample sheet (SampleSheet.csv file) directs the software how to assign reads to samples, and samples to projects. The sample sheet specifies for every index in every lane which sample and which project it belongs to. Lanes with samples that were not indexed can also be assigned to samples and projects using the sample sheet. Projects can consist of multiple samples, and samples can consist of multiple lanes and multiple indexes.

The sample sheet contains the following columns:

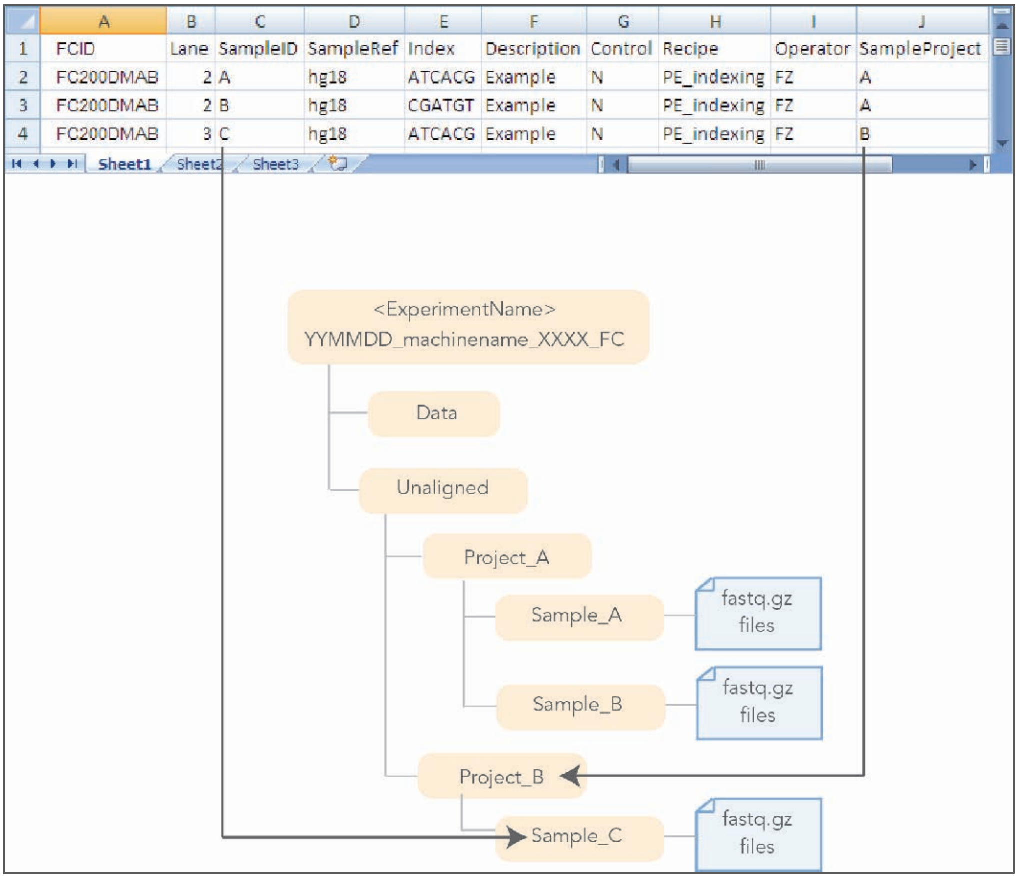
Column	Description
FCID	Flow cell ID
Lane	Positive integer, indicating the lane number (1-8)
SampleID	ID of the sample
SampleRef	The name of the reference
Index	Index sequence
Description	Description of the sample
Control	Y indicates this lane is a control lane, N means sample

Column	Description
Recipe	Recipe used during sequencing
Operator	Name or ID of the operator
SampleProject	The project the sample belongs to

NOTE
The column SampleProject is new in CASAVA 1.8 and links samples to projects.

Every project in the sample sheet is linked to a corresponding project directory. Each sample belonging to that project is linked to a corresponding sample directory within that project directory. Reads are stored in the FASTQ files located in the project and sample directories specified in the sample sheet, as illustrated below for the sample in line 5 of the sample sheet.

Figure 9 Relation between Sample Sheet and Directory Structure



	A	B	C	D	E	F	G	H	I	J
1	FCID	Lane	SampleID	SampleRef	Index	Description	Control	Recipe	Operator	SampleProject
2	FC200DMAB	2	A	hg18	ATCACG	Example	N	PE_indexing	FZ	A
3	FC200DMAB	2	B	hg18	CGATGT	Example	N	PE_indexing	FZ	A
4	FC200DMAB	3	C	hg18	ATCACG	Example	N	PE_indexing	FZ	B

Bcl Conversion/Demultiplexing Examples

Bcl conversion and demultiplexing support four scenarios:

- Multiplexed samples present, with sample sheet.
  Reads are placed within the directory structure specified by the sample sheet, based on the index and lane information. Reads for which the index sequence was ambiguous will be placed in a project directory called Undetermined_indices,

unless the sample sheet specifies a specific sample and project for reads without index in that lane.

- Multiplexed and non-multiplexed samples present, with sample sheet.
  Reads are placed within the directory structure specified by the sample sheet, based on the index and lane information. Reads containing ambiguous or no barcodes will be placed in a project directory called Undetermined_indices, unless the sample sheet specifies a specific sample and project for reads without index in that lane.
- No multiplexed samples present, with sample sheet.
  Reads are placed within the directory structure directed by the sample sheet, based on the lane information.
- No multiplexed samples present, without sample sheet.
  Reads are placed in a project directory named after the flow cell, and sample directories based on the lane number.

Bcl Conversion As You Go

Bcl conversion supports alignment of the first read of a paired-end run before completion of the run.(align as you go). You can kick off Bcl conversion for read 1 using the target `r1` when running `make` at any time after the last read has started (for multiplexed runs, this is after completion of the indexing read). You can also start alignment using the target `r1` when running `make` for configureAlignment, or you can use the `POST_RUN_COMMAND_R1` variable to automatically start the alignment of read 1 at the end of the Bcl conversion.

For instructions, see *Starting Bcl Conversion for Read 1* on page 37.

Demultiplexing Methods

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

Splitting FASTQ Files

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

1. Get the raw index from the .bcl file.
2. Identify the appropriate directory for the index based on the sample sheet.
3. ***Optional***: Detect and correct up to one error on the barcode, and identify the appropriate directory.
4. For each read:
   a. Write the index sequence into the index field.
   b. Append the end to the appropriate new FASTQ file in the selected directory.
5. If the index cannot be identified, the data is written into the Undetermined_indices directory, unless the sample sheet specifies a project and sample for reads without index.

Updating Statistics and Reporting

The sample demultiplexer updates the following files:

- Generates statistics

While splitting the FASTQ files, CASAVA recalculates the base calling analysis statistic that were computed during base calling for the unsplit lanes. These files (Demultiplex_Stats.htm, BustardSummary.xml, and IVC.htm) are stored in the Unaligned/Basecall_Stats_FCID folder.

- Regenerates 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 the Demultiplex_Stats_FCID.csv file in the Unaligned folder to indicate in which subdirectory each index has been written.

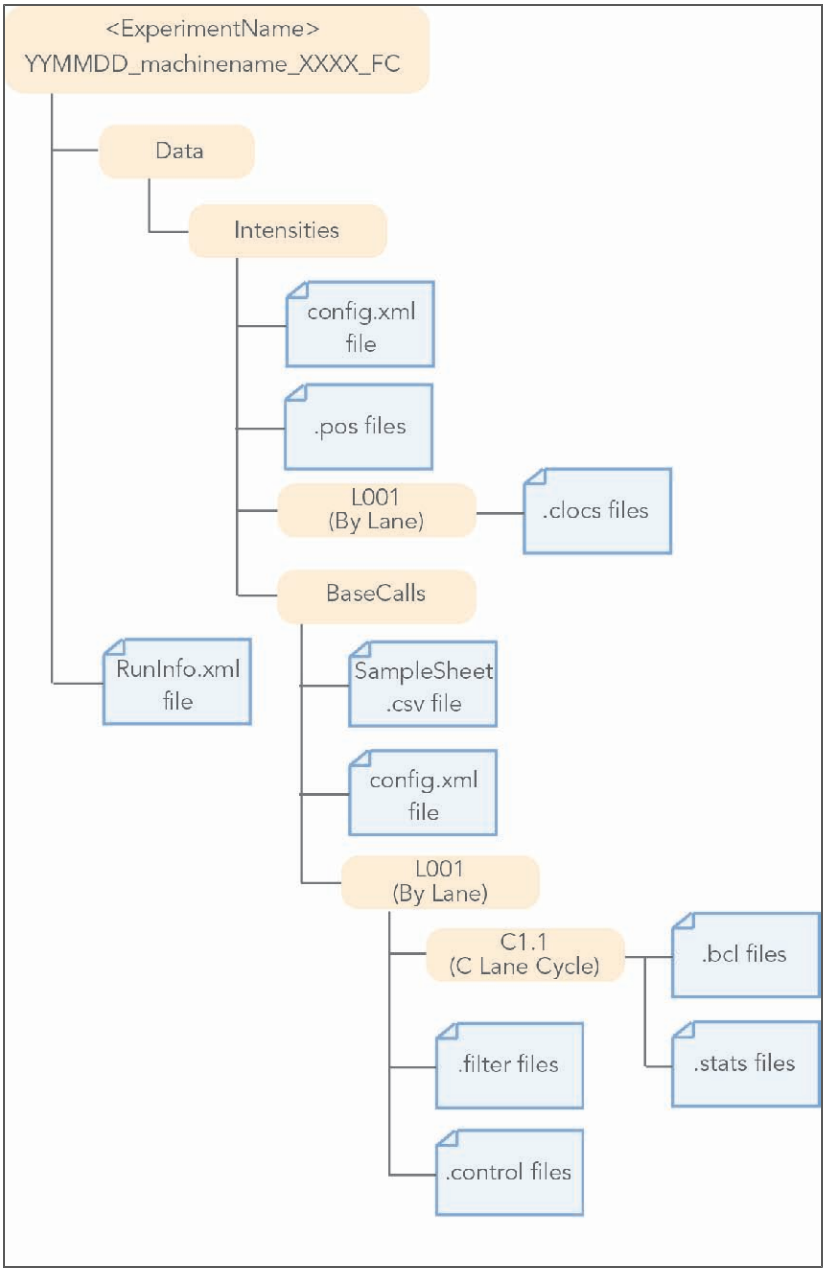
For a description of these files, see *Bcl Conversion Output Folder* on page 38.

Bcl Conversion Input Files

Demultiplexing needs a BaseCalls directory and a sample sheet to start a run. These files are described below. See also image below.

NOTE
For installation instructions, see *Requirements and Software Installation* on page 109.

Figure 10 Bcl Conversion Input Files



Folder and File 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 platform 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_FCYYY.

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.

BaseCalls Directory

Demultiplexing requires a BaseCalls directory as generated by RTA or OLB (Off-Line Basecaller), which contains the binary base call files (*.bcl files).

NOTE
As of 1.8, CASAVA does not use *_qseq.txt files as input anymore.

The BCL to FASTQ converter needs the following input files from the BaseCalls directory:

- *.bcl files.
- *.stats files.

- *.filter files.
- *.control files
- *_pos.txt, *.locs, or *.clocs files. The BCL to FASTQ converter determines which type of position file it looks for based on the RTA version that was used to generate them.
- RunInfo.xml file. The RunInfo.xml is at the top level of the run folder.
- config.xml file

RTA is configured to copy these files off the instrument computer machine to the BaseCalls directory on the analysis server. The files are described below.

Bcl Files

The *.bcl files can be found in the BaseCalls directory:

```
<run directory>/Data/Intensities/BaseCalls/L<lane>/C<cycle>.1
```

They are named as follows:

```
s_<lane>_<tile>.bcl
```

The *.bcl files are binary base call files with the format described below.

Bytes	Description	Data type
Bytes 0–3	Number N of cluster	Unsigned 32bits little endian integer
Bytes 4–(N+3) Where N is the cluster index	Bits 0-1 are the bases, respectively [A, C, G, T] for [0, 1, 2, 3]: bits 2-7 are shifted by two bits and contain the quality score. All bits '0' in a byte is reserved for no-call.	Unsigned 8bits integer

Stats Files

The stats files can be found in the BaseCalls directory:

```
<RunDirectory>/Data/Intensities/BaseCalls/L00<lane>/C<cycle>.1
```

They are named as follows:

```
s_<lane>_<tile>.stats
```

The Stats file is a binary file containing base calling statistics; the content is described below. The data is for clusters passing filter only.

Start	Description	Data type
Byte 0	Cycle number	integer
Byte 4	Average Cycle Intensity	double
Byte 12	Average intensity for A over all clusters with intensity for A	double
Byte 20	Average intensity for C over all clusters with intensity for C	double
Byte 28	Average intensity for G over all clusters with intensity for G	double
Byte 36	Average intensity for T over all clusters with intensity for T	double
Byte 44	Average intensity for A over clusters with base call A	double
Byte 52	Average intensity for C over clusters with base call C	double
Byte 60	Average intensity for G over clusters with base call G	double
Byte 68	Average intensity for T over clusters with base call T	double

Start	Description	Data type
Byte 76	Number of clusters with base call A	integer
Byte 80	Number of clusters with base call C	integer
Byte 84	Number of clusters with base call G	integer
Byte 88	Number of clusters with base call T	integer
Byte 92	Number of clusters with base call X	integer
Byte 96	Number of clusters with intensity for A	integer
Byte 100	Number of clusters with intensity for C	integer
Byte 104	Number of clusters with intensity for G	integer
Byte 108	Number of clusters with intensity for T	integer

Filter Files

The filter files can be found in the BaseCalls directory:

```
<run directory>/Data/Intensities/BaseCalls/L00<lane>/
```

They are named as follows:

```
s_<lane>_<tile>.filter
```

The *.filter files are binary files containing filter results; the format is described below.

Bytes	Description
Bytes 0–3	Zero value (for backwards compatibility)
Bytes 4–7	Filter format version number
Bytes 8–11	Number of clusters
Bytes 12–(N+11) Where N is the cluster number	unsigned 8-bits integer: • Bit 0 is pass or failed filter

Control Files

The control files can be found in the BaseCalls directory:

```
<run directory>/Data/Intensities/BaseCalls/L00<lane>/
```

They are named as follows:

```
s_<lane>_<tile>.control
```

The *.control files are binary files containing control results; the format is described below.

Bytes	Description
Bytes 0–3	Zero value (for backwards compatibility)
Bytes 4–7	Format version number
Bytes 8–11	Number of clusters
Bytes 12–(2xN+11) Where N is the cluster index	The bits are used as follows: • Bit 0: always empty (0) • Bit 1: was the read identified as a control? • Bit 2: was the match ambiguous? • Bit 3: did the read match the phiX tag? • Bit 4: did the read align to match the phiX tag? • Bit 5: did the read match the control index sequence? • Bits 6,7: reserved for future use • Bits 8..15: the report key for the matched record in the controls.fasta file (specified by the REPORT_KEY metadata)

Position Files

The BCL to FASTQ converter can use different types of position files (*.pos, *.locs, or *.clocs), and will expect a type based on the version of RTA used.

Locs files

The locs files can be found in the Intensities directory:

```
<run directory>/Data/Intensities/L00<lane>
```

They are named as follows:

```
s_<lane>_<tile>.locs.
```

Clocs files

The clocs files are compressed versions of locs file and can be found in the Intensities directory:

```
<run directory>/Data/Intensities/L00<lane>
```

They are named as follows:

```
s_<lane>_<tile>.clocs.
```

Pos Files

The *_pos.txt files can be found in the Intensities directory:

```
<run directory>/Data/Intensities
```

They are named as follows:

```
s_<lane>_<tile>_pos.txt
```

The *_pos.txt files are text files with 2 columns and a number of rows equal to the number of clusters. The first column is the X-coordinate and the second column is the Y-coordinate. Each line has a <cr><lf> at the end.

NOTE
The `<tile>` value for *._pos.txt files are padded, while those for *.locs and *.clocs files are unpadded.

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).

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

```
<?xml version="1.0"?>
<RunInfo>
  <Run Id="071112_SLXA-EAS1_0089_FC20120_R1" Number="89" >
    <Instrument>SLXA-EAS1</Instrument>
    <Reads>
      <Read FirstCycle="1" LastCycle="30" />
      <Read FirstCycle="31" LastCycle="37" >
        <Index Name="xxx" FirstCycle="31" LastCycle="37" />
      </Read>
    </Reads>
    <SecondEnd FirstCycle="38" />
    <ActualIndex>
```

```
        <Cycle>31</Cycle>
        <Cycle>32</Cycle>
        <Cycle>33</Cycle>
        <Cycle>34</Cycle>
        <Cycle>35</Cycle>
        <Cycle>36</Cycle>
      </ActualIndex>
    </Run>
  </RunInfo>
```

config.xml Files

In the Intensities 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.

In the BaseCalls folder there is another config.xml file containing the meta-information about the base caller runs.

Generating the Sample Sheet

The user generated sample sheet (SampleSheet.csv file) describes the samples and projects in each lane, including the indexes used. The sample sheet should be located in the BaseCalls directory of the run folder. You can create, open, and edit the sample sheet in Excel.

The sample sheet contains the following Column:

Column Header	Description
FCID	Flow cell ID
Lane	Positive integer, indicating the lane number (1-8)
SampleID	ID of the sample
SampleRef	The reference sequence for the sample
Index	Index sequence
Description	Description of the sample
Control	Y indicates this lane is a control lane, N means sample
Recipe	Recipe used during sequencing
Operator	Name or ID of the operator
SampleProject	The project the sample belongs to

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. If the sample you want to specify does not have an index sequence, leave the Index field empty.

Illegal Characters

Project and sample names in the sample sheet cannot contain illegal characters not allowed by some file systems. The characters not allowed are the space character and the following:

```
? ( ) [ ] / \ = + < > : ; " ' , * ^ | &
```

Samples Without Index

As of CASAVA 1.8, you can assign samples without index to projects, sampleIDs, or other identifiers by leaving the Index field empty.

Running Bcl Conversion and Demultiplexing

Bcl conversion and demultiplexing is performed by one script, demultiplex.pl. This section describes how to perform Bcl conversion and demultiplexing in CASAVA 1.8.

Usage of configureBclToFastq.pl

The standard way to run bcl conversion and demultiplexing is to first create the necessary Makefiles, which configure the run. Then you run `make` on the generated files, which executes the calculations.

1 Enter the following command to create a makefile for demultiplexing:

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

> NOTE
> The options have changed significantly between CASAVA 1.7 and 1.8. See *Options for Bcl Conversion and Demultiplexing* on page 35.

2 Move into the newly created Unaligned folder specified by `--output-dir`.

3 Type the “make” command. Suggestions for “make” usage, depending on your workflow, are listed below.

Make Usage	Workflow
`nohup make -j N`	Bcl conversion and demultiplexing (default).
`nohup make -j N r1`	Bcl conversion and demultiplexing for read 1.

- The -j option specifies the extent of parallelization, with the options depending on the setup of your computer or computing cluster (see *Using Parallelization* on page 117).
- The Unix nohup command redirects the standard output and keeps the “make” process running even if your terminal is interrupted or if you log out.

See *Makefile Options for Bcl Conversion and Demultiplexing* on page 36 for explanation of the options.

> NOTE
> The ALIGN option, which kicked off configureAlignment after demultiplexing was done in CASAVA 1.7, is no longer available.

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

See *Demultiplex_Stats File* on page 41 and *DemultiplexedBustardSummary.xml File* on page 42.

Example Bcl Conversion and Demultiplexing

An example of a demultiplexing run is as follows:

1 Enter:

```
/path-to-CASAVA/bin/configureBclToFastq.pl --input-dir
   <BaseCalls_dir> --output-dir <Unaligned> --sample-sheet
   <input_dir>/SampleSheet.csv
```

2 Go to the <Unaligned> folder.

3 Run:

```
nohup make -j 3
```

Step one will produce a set of directories in the Unaligned directory. Reads with an unresolved or erroneous index are placed in the Undetermined_indices directory.

Options for Bcl Conversion and Demultiplexing

The options for demultiplexing are described below.

Option	Description	Examples
`--fastq-cluster-count`	Maximum number of clusters per output FASTQ file. Do not go over 16000000, since this is the maximum number of reads we recommend for one ELAND process. Defaults to 4000000.	`--fastq-cluster-count 6000000`
`-i, --input-dir`	Path to a BaseCalls directory.\ Defaults to current dir	`--input-dir <BaseCalls_dir>`
`-o, --output-dir`	Path to demultiplexed output. Defaults to <run_folder>/Unaligned Note that there can be only one Unaligned directory by default. If you want multiple Unaligned directories, you will have to use this option to generate a different output directory.	`--output-dir <run_folder>/Unaligned`
`--positions-dir`	Path to a directory containing positions files. Defaults depends on the RTA version that is detected.	`--positions-dir <positions_dir>`
`--positions-format`	Format of the input cluster positions information. Options: • .locs • .clocs • _pos.txt Defaults to .clocs.	`--positions-format .locs`
`--filter-dir`	Path to a directory containing filter files. Defaults depends on RTA version that is detected.	`--filter-dir <filter_dir>`
`--intensities-dir`	Path to a valid Intensities directory. Defaults to parent of base_calls_dir.	`--intensities-dir <intensities_dir>`
`-s,--sample-sheet`	Path to sample sheet file. Defaults to <input_dir>/SampleSheet.csv	`--sample-sheet <input_dir>/SampleSheet.csv`
`--tiles`	--tiles option takes a comma-separated list of regular expressions to match against the expected "s_<lane>_<tile>" pattern, where <lane> is the lane number (1-8) and <tile> is the 4 digit tile number (left-padded with 0s).	`--tiles=s_[2468]_[0-9][0-9][02468]5,s_1_0001`

`--use-bases-mask`	The `--use-bases-mask` string specifies how to use each cycle.
	• An "n" means ignore the cycle.
	• A "Y" means use the cycle (for paired-end, this designates the first end).
	• An "I" means use the cycle for the index read.
	• A number means that the previous character is repeated that many times.
		`--use-bases-mask y50n,I6n,Y50n`
		`This means:`
		• Use first 50 bases for

Option	Description	Examples
	• The reads are defined by commas "," If this option is not specified, the mask will be determined from the 'RunInfo.xml' file in the run directory. If no information is present in the 'RunInfo.xml', the mask will be determined from the 'config.xml' file in the base-calls directory. If no information is present in the 'config.xml' file, the default 'Y*' will be used.	first read (Y50) • Ignore the next (n) • Use 6 bases for index (I6) • Ignore next (n) • Use 50 bases for second read (Y50) • Ignore next (n)
`--no-eamss`	Disable the masking of the quality values with the EAMSS filter.	`--no-eamss`
`--mismatches`	Number of mismatches allowed in the indexes (0 or 1). Default is 0.	`--mismatches 1`
`--flowcell-id`	Use the specified string as the flowcell id. (default value is parsed from the config-file)	`--flowcell-id flow_cell_id`
`--ignore-missing-stats`	Fill in with zeros when *.stats files are missing	`--ignore-missing-stats`
`--ignore-missing-bcl`	Interpret missing *.bcl files as no call	`--ignore-missing-bcl`
`--ignore-missing-control`	Interpret missing control files as not-set control bits	`--ignore-missing-control`
`--man`	Print a manual page for this command	`--man`
`-h, --help`	Produce help message and exit	`-h`

Makefile Options for Bcl Conversion and Demultiplexing

The options for `make` usage in demultiplexing/analysis are described below.

Parameter	Description
nohup	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 -j n & The optional "&" tells the system to run the analysis in the background, leaving you free to enter more commands. We suggest always running nohup to help troubleshooting if issues arise.
-j N	The -j option specifies the extent of parallelization, with the options depending on the setup of your computer or computing cluster. For a description of parallellization, see *Using Parallelization* on page 117.
r1	Runs Bcl conversion for read 1. Can be started once the last read has started sequencing.
POST_RUN_COMMAND_R1	A Makefile variable that can be specified either on the make command line or as an environment variable to specify the post-run commands after completion of read one, if needed. Typical use would be triggering the alignment of read 1.
POST_RUN_COMMAND	A Makefile variable that can be specified on the make command line to specify the post-run commands after completion of the run.
KEEP_INTERMEDIARY	The option KEEP_INTERMEDIARY tells CASAVA not to delete the intermediary files in the Temp dir after Bcl conversion is complete. Usage: `KEEP_INTERMEDIARY:=yes`

NOTE

If you specify one of the more specific workflows and then run a more general one, only the difference will get processed. For instance:

```
make -j N r1
```

followed by:

```
 make -j N
```

will do read 1 in the first step, and read 2 the second one.

Starting Bcl Conversion for Read 1

If you want to start Bcl to FASTQ conversion before completion of the run, use the makefile target `r1` at any time after the last read has started (for multiplexed runs, this is after completion of the indexing read).

1 Enter the following command to create a makefile for Bcl conversion:

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

2 Move into the newly created Unaligned folder specified by --`output-dir`.

3 Type the "make r1" command:

```
make -j 8 r1
```

> NOTE
> the `-j <n>` command line option is supported to indicate up to <n> processes in parallel. However, for Bcl conversion the maximum level of parallelization is 8.

4 After the analysis is done, review the analysis for each sample.\
See *Demultiplex_Stats File* on page 41 and *DemultiplexedBustardSummary.xml File* on page 42.

Starting Alignment

You can also start alignment before completion of the run using the target `r1` when running `make` for configureAlignment.

See *Starting Alignment for Read 1* on page 63.

Alternatively, you can use the `POST_RUN_COMMAND_R1` variable to automatically start the alignment of read 1 at the end of the Bcl conversion. For example:

```
make -j 8 r1 POST_RUN_COMMAND_R1="cd ../Aligned ; make -j 16
   r1"
```

Starting the Second Read

To start Bcl conversion of the second read, use the regular make command in the Unaligned folder. Perform the following:

1 Move into the Unaligned folder specified by --`output-dir`.

2 Type the regular "make" command:

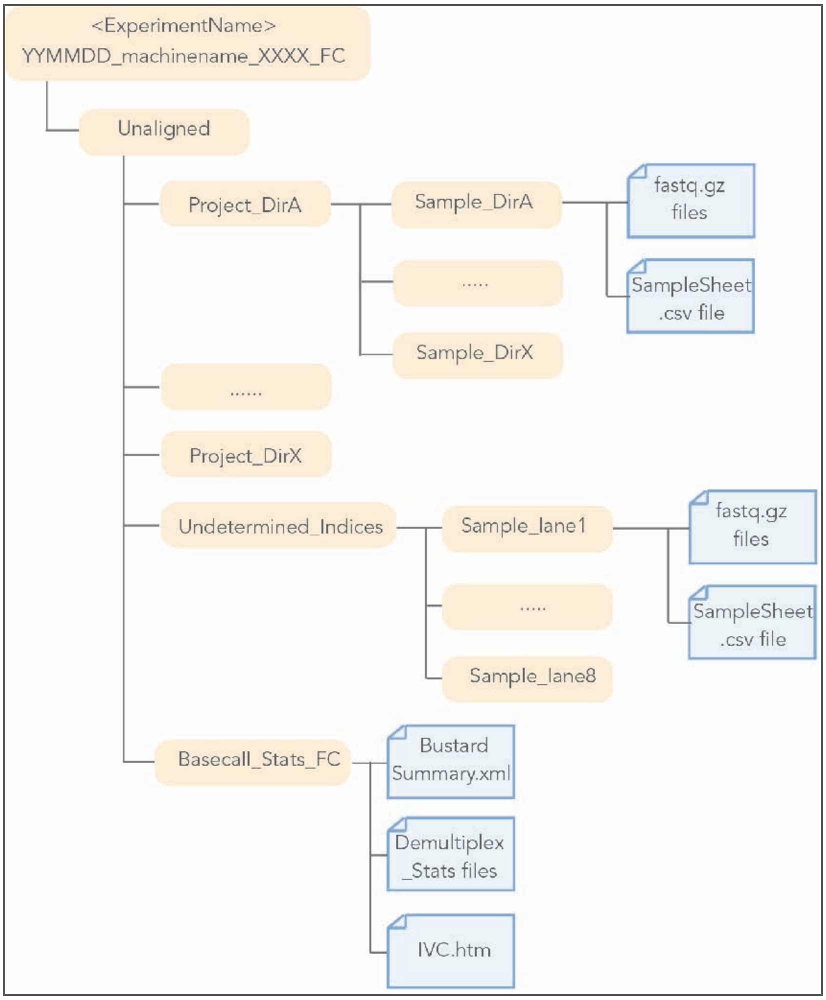
```
make -j 8
```

Bcl Conversion Output Folder

The Bcl Conversion output directory has the following characteristics:

- The project and sample directory names are derived from the sample sheet.
- The Demultiplex_Stats file shows where the sample data are saved in the directory structure.
- The Undetermined_indices directory contains the reads with an unresolved or erroneous index.
- If no sample sheet exists, CASAVA generates a project directory named after the flow cell, and sample directories for each lane.
- Each directory is a valid base calls directory that can be used for subsequent alignment analysis in CASAVA.

NOTE
If the majority of reads end up in the 'Undetermined_indices' folder, check the --use-bases-mask parameter syntax and the length of the index in the sample sheet. It may be that you need to set the --use-bases-mask option to the length of the index in the sample sheet + the character 'n' to account for phasing



NOTE
There can be only one Unaligned directory by default. If you want multiple Unaligned directories, you will have to use the option `--output-dir` to generate a different output directory.

FASTQ Files

As of 1.8, CASAVA converts *.bcl files into FASTQ files, and uses these FASTQ files as sequence input for configureAlignment. The files are located in the Unaligned/Project_<ProjectName>/Sample_<SampleName> directories.

NOTE
Reads that were identified as sample prep controls in the control files are not saved in the FASTQ files.

Naming

Illumina FASTQ files use the following naming scheme:

```
<sample name>_<barcode sequence>_L<lane (0-padded to 3
   digits)>_R<read number>_<set number (0-padded to 3
   digits>.fastq.gz
```

For example, the following is a valid FASTQ file name:

```
NA10831_ATCACG_L002_R1_001.fastq.gz
```

In the case of non-multiplexed runs, <sample name> will be replaced with the lane numbers (lane1, lane2, ..., lane8) and <barcode sequence> will be replaced with "NoIndex".

Set Size

The FASTQ files are divided in files with the file size set by the --fastq-cluster-count command line option of configureBclToFastq.pl.The different files are distinguished by the 0-padded 3-digit set number.

TIP
If you need to combine the divided fastq gzipped files into one unique fastq gzipped file for use in a third-party tool, you can use the `cat` command, for example, like this:

```
cd Unaligned/Project_ID/sample_ID
cat NA10831_ATCACG_L002_R1_001.fastq.gz NA10831_ATCACG_
L002_R1_002.fastq.gz > NA10831_ATCACG_L002_R1_
combined.fastq.gz
```

Compression

FASTQ files are saved compressed in the GNU zip format, an open source file compression program. This is indicated by the .gz file extension. CASAVA automatically unzips the files before using them.

Format

Each entry in a FASTQ file consists of four lines:

- Sequence identifier
- Sequence
- Quality score identifier line (consisting of a +)
- Quality score

Each sequence identifier, the line that precedes the sequence and describes it, needs to be in the following format:

```
@<instrument>:<run number>:<flowcell ID>:<lane>:<tile>:<x-
   pos>:<y-pos> <read>:<is filtered>:<control number>:<index
   sequence>
```

The elements are described below.

Element	Requirements	Description
@	@	Each sequence identifier line starts with @
`<instrument>`	Characters allowed: a-z, A-Z, 0-9 and underscore	Instrument ID
`<run number>`	Numerical	Run number on instrument
`<flowcell ID>`	Characters allowed: a-z, A-Z, 0-9	
`<lane>`	Numerical	Lane number
`<tile>`	Numerical	Tile number
`<x_pos>`	Numerical	X coordinate of cluster
`<y_pos>`	Numerical	Y coordinate of cluster
`<read>`	Numerical	Read number. 1 can be single read or read 2 of paired-end
`<is filtered>`	Y or N	Y if the read is filtered, N otherwise
`<control number>`	Numerical	0 when none of the control bits are on, otherwise it is an even number
`<index sequence>`	ACTG	Index sequence

An example of a valid entry is as follows; note the space preceding the read number element:

```
@EAS139:136:FC706VJ:2:5:1000:12850 1:Y:18:ATCACG
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
+
BBBBCCCC?<A?BC?7@@???????DBBA@@@@A@@
```

NOTE
Be aware that the CASAVA 1.8 FASTQ files contain all reads, both the reads that passed filtering, as well as the reads that did not pass filtering. If you use third-party software that cannot handle this, we recommend that clean up the FASTQ files first using the `<is filtered>` field described above. you can use the

```
cd /path/to/project/sample
mkdir filtered
for fastq in *.fastq.gz ; do zcat $fastq | grep
   -A 4 '^@.* [^:]*:N:[^:]*:' > filtered/$fastq
   ; done
```

Quality Scores

A quality score (or Q-score) expresses an error probability. In particular, it serves as a convenient and compact way to communicate very small error probabilities.

Given an assertion, A, the probability that A is not true, P(~A), is expressed by a quality score, Q(A), according to the relationship:

$$Q(A) =-10 \log_{10}(P(\sim A))$$

where *P(~A)* is the estimated probability of an assertion A being wrong.

The relationship between the quality score and error probability is demonstrated with the following table:

Quality score, Q(A)	Error probability, P(~A)
10	0.1
20	0.01
30	0.001

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 # in Illumina's text-based encoding of quality scores), while the rest of the quality values within the read remain unchanged. 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.

Demultiplex_Stats File

The Demultiplex_Stats.htm file provides stats about demultiplexing and shows where samples are saved in the directory structure. The Demultiplex_Stats file is located in the Unaligned/Basecall_Stats_FCID directory.

The file contains the sample information from the sample sheet, with added rows for reads that end up in the Undetermined_indices directory. If no sample sheet exists, CASAVA generates rows for each lane. The Demultiplex_Stats file has a number of additional columns that display demultiplexing stats and show the directory the samples are saved in. The Demultiplex_Stats file contains the following fields:

Field	Description
Lane	Positive integer, indicating the lane number (1-8)

Field	Description
SampleID	ID of the sample
SampleRef	The reference sequence for the sample
Index	Index sequence
Description	Description of the sample
Control	Y indicates this lane is a control lane, N means sample
Project	The project the sample belongs to
# Reads	Number of reads, equals (total number of lines in fastq files)/4
Yield	The sum of all bases in clusters that passed filtering for the entire project.
% PF	The percentage of clusters that passed filtering.
% of Lane	Percentage of reads in the sample compared to total number of reads in that lane.
% Perfect Index Reads	Percentage of index reads in this sample which perfectly matched the given index.
% One Mismatch Reads (Index)	Percentage of index reads in this sample which had 1 mismatch to given index.
% of >=Q30 Bases	Yield of bases with Q30 or higher from clusters passing filter divided by total yield of clusters passing filter.
Mean Quality Score	The total sum of quality scores of clusters passing filter divided by total yield of clusters passing filter.
Recipe	Recipe used during sequencing
Operator	Name or ID of the operator
Directory	Full path to the directory.

Below the sample information are links to the IVC plots and BustardSummary.xml file. For a description of the data, see *DemultiplexedBustardSummary.xml File* on page 42.

Finding Demultiplexed Samples

The key to finding the location of demultiplexed data is looking at the Demultiplex_Stats.htm file in the BaseCalls_Stats directory. The Directory column will indicate the project/sample output directory. The FASTQ files within the directory contain the index and lane as part of the name. Alternatively it can be inferred from the project name and the sample id as described in *FASTQ Files* on page 39.

DemultiplexedBustardSummary.xml File

The DemultiplexedBustardSummary.xml file contains five types of tables with overviews of quality metrics (described below).

NOTE
In the descriptions of the tables included in the BustardSummary.xml file, 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:

- The original number of detected clusters.
- The number of clusters that passed quality filtering.

- The flow cell (chip) yield in Mb. This is the sum of the quality-filtered bases used for analysis over analyzed lanes.

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 RTA image analysis.
- **Clusters (PF)**—The number of detected clusters that meet the filtering criterion.
- **First 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.

Expanded Lane Summary

This displays more detailed quality metrics for each lane. Apart from the phasing and prephasing information (which display values for an entire lane), all values are tile means for the lane.

- **Clusters (tile mean) (raw)**—The number of clusters detected by RTA of images.
- **% Phasing**—The estimated (or specified) value used by RTA for the percentage of molecules in a cluster for which sequencing falls behind the current position (cycle) within a read.
- **% Prephasing**—The estimated value used by RTA for the percentage of molecules in a cluster for which sequencing jumps ahead of the current position (cycle) within a read.
- **% 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.

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 in the Lane Results Summary, but are presented as averages over the detected (raw) or passing-filter (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. This suggests an exceptional situation (e.g., a bubble) within the tile. The brackets indicate that the tile has been excluded from the calculation of lane statistics and that the values are reported only for diagnostic purposes.

IVC Plots

The BustardSummary.xml file provides a link to the Intensity versus Cycle (IVC) Plots at the bottom of the page. These plots display lane averages over all tiles in a lane. The plots displayed are All, Called, %Base_Calls, %All, and %Called.

- **All**—This displays the mean intensity of clusters after adjustment for cross-talk between channels. The data is plotted as a function of cycle, and each channel (A, C, G, T) is plotted separately as a different colored line. Means are calculated over all clusters, regardless of base calling.

  If all bases are present in the sample at 25% of total and a well-balanced matrix is used for analysis, the graph displays all channels with similar intensities. If intensities are not similar, the results could indicate 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 is four times that of All, as the intensities are only averaged over 25% of the clusters. For impure clusters, the difference in intensity is less than four times that of All.

  The Called intensities are independent of base representation, so a well-balanced matrix displays 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 others 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.

All Intensity Plots

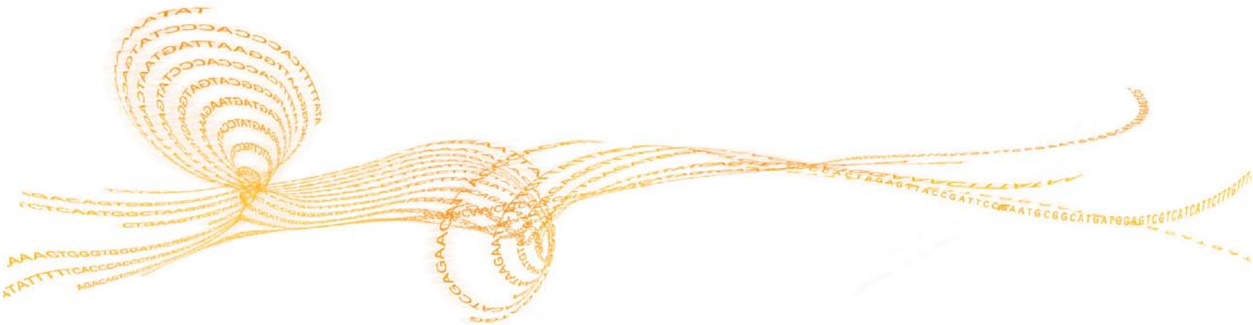
The BustardSummary.xml file provides a link to the All Intensity Plots at the bottom of the page. These plots give a tile-by-tile representation of the mean matrix-adjusted intensity of clusters plotted as a function of cycle. Each channel (A, C, G, T) is represented as a different colored line. Means are calculated over all clusters, regardless of base calling.

If all bases are present in the sample at 25% of total and a well-balanced matrix is used for analysis, the graph displays all channels with similar intensities. Dissimilar intensities could indicate poor cross-talk correction or poor absolute intensity balance between each channel. A genome rich in GC content may not provide a balanced matrix for accurate cross-talk correction and absolute intensity balance.

NOTE
For large experiments, All.htm only shows a subset of tiles. However, each file contains links to the full output results. The full output files may take some time to open.

Chapter 4

Sequence Alignment



Introduction

configureAlignment is a CASAVA module that performs sequence alignments. This chapter describes running configureAlignment, parameters, analysis variables, configuration file options, and ELANDv2e alignments.

NOTE
For installation instructions, see *Requirements and Software Installation* on page 109.

Configuring configureAlignment

You can define configureAlignment analysis parameters in a 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 *configureAlignment Parameters Detailed Description* on page 60.

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

configureAlignment and Align As You Go

Bcl conversion supports alignment of the first read of a paired-end run before completion of the run.(align as you go). You can kick off alignment for read 1 using the target `r1` when running `make` at any time after Bcl conversion for read 1 is complete.

For instructions, see *Starting Alignment for Read 1* on page 63.

configureAlignment Output

configureAlignment output is a flat text file called *_export.txt.gz containing each read and information about its alignment to the reference. In addition, configureAlignment 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 Aligned output folder.

As a result of running the configureAlignment.pl script, a new directory is created in the run folder. This directory is named using the format Aligned. If you want to rerun the analysis and change parameters, you can rerun configureAlignment with new parameters if you specify a new alignment directory (OUT_DIR).

CASAVA 1.8 also contains a script that converts *_export.txt files to SAM files (see *Introduction* on page 168 and *SAM Format* on page 169).

Alignment Algorithms

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

ELAND has been improved a number of times:

- CASAVA 1.6 introduced a new version of ELAND, ELANDv2. The most important improvements of ELANDv2 are its ability to perform multiseed and gapped alignments.

- As of CASAVA 1.8 a new version of ELANDv2 is available, ELANDv2e. The most important improvements of ELANDv2e are improved repeat resolution and implementation of orphan alignment.

A short description of these improvements is provided below; more information about ELANDv2 is available in *Algorithm Descriptions* on page 129.

ELANDv2

The most important improvement of ELANDv2 are the following:

- Handles indels and mismatches better by performing multiseed and gapped alignments.
- Enhanced match descriptor options to handle the gaps identified (see *Export.txt.gz* on page 77).
- Ability to split queries on a per tile basis now to allow for much greater parallelization.

The hashing method in ELANDv2 has been optimized in CASAVA 1.7 for performance. This leads to a significant improvement in running times for the seed-matching step of the alignment in CASAVA. More information about ELANDv2 is available in on page 129.

ELANDv2e Alignment Improvements

CASAVA 1.8 launches ELANDv2e, and contains the following new features for alignment: ELANDv2e has better repeat resolution ELANDv2e introduces the orphan aligneralignmentResolver is rewritten, resulting in shorter run times

configureAlignment Input Files

The folder structure and format of configureAlignment input has changed significantly in CASAVA 1.8. The major changes are as follows:

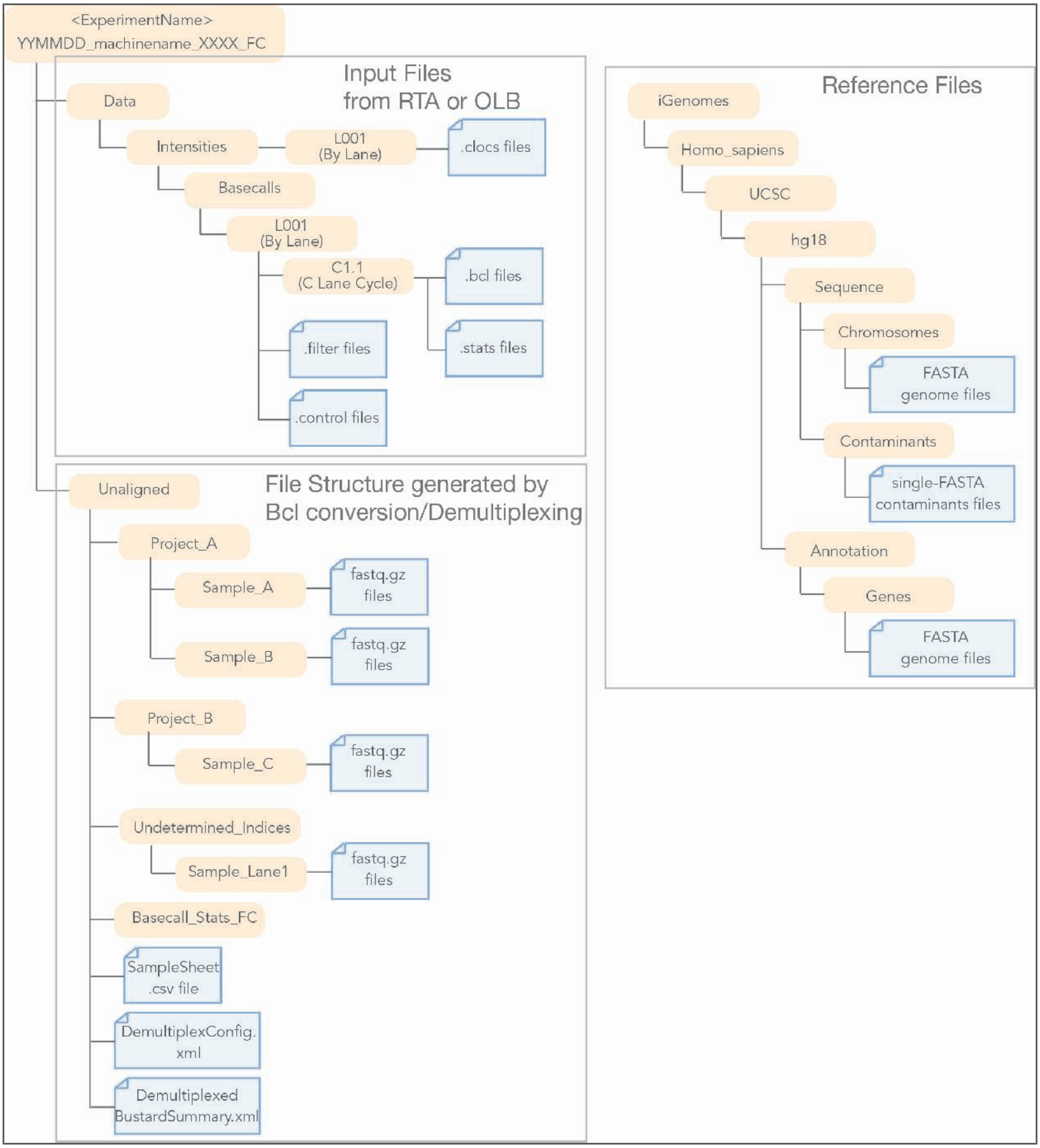
- configureAlignment uses FASTQ files as sequence input.
- Bcl conversion and demultiplexing are merged in one step, and both multiplexed and non-multiplexed samples need to be processed.
- The FASTQ files for both multiplexed and non-multiplexed samples are organized using the Project and Sample concepts, as governed by the sample sheet.
- configureAlignment uses the sample sheet to identify projects and samples, and the sample organization as described in the sample sheet should always match the actual Unaligned folder organization.

As a result of these changes, configureAlignment expects the following input files:

- A Unaligned directory with fastq.gz files (even for cases where only one project exists).
- A config.txt file, which specifies the analysis.
- A base calling summary.xml file (DemultiplexedBustardSummary.xml).
- A base calling config.xml file (DemultiplexedBustardConfig.xml).
- A FASTA reference genome for alignments.
- For RNA applications, additional files are required.

This section explains these files. For file locations, see figure below. Note that the reference files may be located in a different location, depending on your CASAVA installation.

Figure 11 Locations of configureAlignment Input Files



Sequence Files

configureAlignment needs a Unaligned directory as generated by BCL to FASTQ converter, which contains the gzipped sequence files (*.fastq.gz files).

NOTE
As of CASAVA 1.8, configureAlignment uses FASTQ input files instead of *_ qseq.txt files.

For a description of the FASTQ files, see *FASTQ Files* on page 39.

Configuration File

The configureAlignment configuration file (generally named config.txt) specifies what analysis should be done for each lane. The requirements and options for the configureAlignment configuration file are described in *configureAlignment Configuration File* on page 55.

Sample Sheet

The SampleSheet.csv file describes the samples and projects in each lane, including the indexes used. It is derived from the user generated sample sheet that is required for bcl conversion and demultiplexing. The sample sheet should be located in the Unaligned directory of the run folder.

The sample sheet has to match the directory structure created during the bcl conversion and demultiplexing. If you need to change the sample sheet, it is best to rerun the bcl conversion and demultiplexing.

DemultiplexedBustardSummary.xml

The DemultiplexedBustardSummary.xml file in the Unaligned directory contains the intensity summary results that are merged with configureAlignment's alignment results to produce Summary.xml and Summary.htm. The DemultiplexedBustardSummary.xml file is derived from the BustardSummary.xml file generated during base calling, but renamed and moved by the BCL to FASTQ converter.

DemultiplexedBustardConfig.xml File

The base calling configuration file (DemultiplexedBustardConfig.xml) in the demultiplexed directory includes the start and end cycles of each read. The DemultiplexedBustardConfig.xml file is derived from the Config.xml file generated during base calling, but renamed and moved by the BCL to FASTQ converter.

Reference Genome

CASAVA uses a reference genome in FASTA format. Both single sequence FASTA and multi-sequence FASTA genome files are supported.

Genome sequence files for most commonly used model organisms are available through iGenome (*Getting Reference Files* on page 126).

NOTE
As of CASAVA 1.8, you do not need to squash the reference genome anymore.

Single Sequence FASTA Files

CASAVA accepts single-sequence FASTA files as genome reference, which should be provided unsquashed for both alignment and post-alignment steps. The chromosome name is derived from the file name.

Direct CASAVA to a folder containing the FASTA files using the option `ELAND_GENOME` for configureAlignment.

Multi-Sequence FASTA Files

As of version 1.8, CASAVA accepts a multi-sequence FASTA file as genome reference. This should be provided as a single genome, SAM compliant, unsquashed file, for both alignment and post-alignment steps. The chromosome name is derived directly from the first word in the header for each sequence.

Direct CASAVA to a multi-sequence FASTA file using the option `SAMTOOLS_GENOME` for configureAlignment.

WARNING
GenomeStudio does not support the use of multi-sequence FASTA files. Therefore, if you want to analyze your output in GenomeStudio, we recommend using single sequence FASTA reference files.

Chromosome Naming Restrictions

CASAVA does not accept the following characters in the FASTA chromosome name header:

```
- ? ( ) [ ] / \ = + < > : ; " ' , * ^ | &
```

This validation can be disabled in configureAlignment using the following option:

```
CHROM_NAME_VALIDATION off
```

WARNING
You may run into problems with downstream analysis if you disable chromosome name validation.

NOTE
If ELAND finds two alignments with identical alignment scores, ELAND will pick the first alignment (in the single-end case) or combination of alignments (in the paired-end case) that exhibit the highest observed alignment quality. These are the alignments that make it into the export files (which only contain the best alignment for each read). In practice, post-alignment CASAVA ignores these reads because of the low alignment qualities.Using a reference with lexicographic chromosome names (like chr1) will yield slightly different results compared to a reference with numerical chromosome names (like 1) for these reads, since the hits are sorted in a different way.

Reference Sequence Blocks

For reasons of efficiency, ELAND treats 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.)

Additional eland_rna Input Files

The following additional files are needed for eland_rna:

- **refFlat.txt.gz** or **seq_gene.md.gz** file—as of CASAVA 1.7, eland_rna uses the refFlat.txt.gz or seq_gene.md.gz file to generate the splice junction set automatically. The refFlat.txt.gz file is available from UCSC, while the seq_gene.md.gz file is from NCBI. They should be provided gzip compressed, and should be from the same build as the reference files you are using for alignment. This negates the need to provide separate splice junction sets as in earlier versions of CASAVA. The parameter to use for either one is ELAND_RNA_GENOME_ANNOTATOTION.

  WARNING
  Do not change the names of the **refFlat.txt.gz** or **seq_gene.md.gz** file. CASAVA uses the name to determine the type of file.

- A set of **contaminant sequences** for the genome—typically the mitochondrial and ribosomal sequences. These must be in single FASTA format. The parameter to use to direct to the contaminant sequence files is ELAND_RNA_GENOME_CONTAM.

Running configureAlignment

When running configureAlignment, two concepts are important to understand: the configureAlignment configuration file that specifies analysis, and the make utility that manages the analysis.

configureAlignment Configuration File

configureAlignment 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. Analysis can be specified by lane, index (barcode), sample, reference, or project.

Make Utility

configureAlignment 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 configureAlignment.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 so can be included in the distribution and do not have to be regenerated for every run.

Standard configureAlignment Analysis

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

1 Edit the configureAlignment configuration file as described in *configureAlignment Configuration File* on page 55.

2 Check the analysis by running the configureAlignment.pl command without `--make`.

```
/path-to-CASAVA/bin/configureAlignment.pl config.txt
   --EXPT_DIR path_to_Unaligned_folder
```

3 Enter the configureAlignment.pl command, but now with `--make`. This creates the makefile for sequence alignment.

```
/path-to-CASAVA/bin/configureAlignment.pl config.txt
   --EXPT_DIR path_to_Unaligned_folder --make
```

4 Move into the newly created Aligned folder under the Run folder (see *configureAlignment Output Files* on page 71). Type the "make" command for basic analysis:

```
make
```

> NOTE
> 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 parallellization, see *Using Parallelization* on page 117.

5 After the analysis is done, review the analysis.

a View the analysis results of your run. See *Analysis Summary* on page 72 and *Analysis Results* on page 77.

b Interpret the run quality. See *Interpretation of configureAlignment Run Quality* on page 81.

ELAND_FASTQ_FILES_PER_PROCESS

CASAVA requires a minimum of 2 GB RAM per core. The parameter ELAND_FASTQ_FILES_PER_PROCESS in the configureAlignment config.txt specifies the maximum number of FASTQ files aligned by each ELAND process, to limit the per-core memory consumption.

NOTE
`ELAND_FASTQ_FILES_PER_PROCESS` is equivalent to the `ELAND_SET_SIZE` parameter used in CASAVA 1.7 and earlier.

The optimal value leads to approximately 10 to 13 million clusters in one set. Since the FASTQ file size (in reads) is determined by the Bcl conversion option --fastq-cluster-count , while the maximum number of files per process is determined by ELAND_FASTQ_FILES_PER_PROCESS, the product of these options should not exceed 16 million:

(`ELAND_FASTQ_FILES_PER_PROCESS` value) × (`--fastq-cluster-count` value) ≤ 16 million

NOTE
The --fastq-cluster-count used during Bcl conversion can be found in Unaligned/Makefile.

See the table below for set size - cluster count combinations.

CAUTION
Setting the right value for the ELAND_FASTQ_FILES_PER_PROCESS is very important. Too high may result in silent crashes due to too high memory utilization, and should be avoided. Too low may result in a decreased performance.

--fastq-cluster-count	ELAND_FASTQ_FILES_PER_PROCESS	Reads per process	Comment
12 000 000	1	12000000	
6 000 000	2	12000000	
4 000 000	3	12000000	Default values
3 000 000	4	12000000	
2 000 000	6	12000000	
1 000 000	12	12000000	

NOTE
Slightly differences can be expected when using different combinations of `--fastq-cluster-count` and `ELAND_FASTQ_FILES_PER_PROCESS`. We recommend using default parameters.

The --fastq-cluster-count used during Bcl conversion can be found in Unaligned/Makefile.

configureAlignment Configuration File

This section describes the features and parameters of the configureAlignment configuration text file.

The configureAlignment configuration file specifies the analysis for each lane, sample, project, reference, or index (barcode). The configureAlignment configuration file is a text file, and the path to the file should be the first argument after the configureAlignment.pl command. configureAlignment translates the analysis in the configuration file into a makefile. The makefile specifies exactly what commands will be executed to carry out the requested analysis.

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

Config File Parameter List

The following tables list the parameters that can be specified in a configureAlignment configuration file.

The section *configureAlignment Parameters Detailed Description* on page 60 provides a detailed description of these parameters.

Core Parameters

Table 1 GERALD Configuration File Core Parameters

Parameter	Definition
`EXPT_DIR data/110113_ILMN-1_0217_FC1234/Unaligned`	Provide the path to the experiment (demultiplexed) directory in the run folder, if not specified on the command line. Usually the output folder from the BCL to FASTQ converter. The path should always be to the Unaligned directory, even when the run only contains one project For a description of the run folder, see *Bcl Conversion Output Folder* on page 38.
`USE_BASES nY*n`	Ignore the first and last base of the read. The USE_BASES string contains a character for each cycle. • If the character is "Y", the cycle is used for alignment. • If the character is "n", the cycle is ignored. • Wild cards (*) are expanded to the full length of the read. USE_BASES should not be used for masking custom index cycles; use the `--use-bases-mask` option (*Options for Bcl Conversion and Demultiplexing* on page 35). Default is `USE_BASES Y*n`, which means perform a single-read alignment and ignore the last base. For a detailed description of USE_BASES syntax, see *USE_BASES Option* on page 61.
`ELAND_GENOME /home/user/Genomes/Eland/BAC_plus_vector/`	Specify the single FASTA files that you want to use as genome reference for alignment with ELANDv2e.
`SAMTOOLS_GENOME`	Direct CASAVA to a multi-sequence FASTA reference file.
`ANALYSIS eland_extended`	Specify the type of alignment that should be performed. Available options are: • `ANALYSIS eland_extended` • `ANALYSIS eland_pair`

Parameter	Definition
	• ANALYSIS eland_rna • ANALYSIS none The default is ANALYSIS none See *ANALYSIS Variables* on page 61 for more information.
ELAND_FASTQ_FILES_PER_PROCESS N	The maximum number of files analyzed by each ELAND process, needed to ensure that the memory usage stays below 2 GB. The optimal value is such that there are approximately 10 to 13 million lines (reads) in one set. Only available for ANALYSIS eland_extended, ANALYSIS eland_pair, and ANALYSIS eland_rna. See *ELAND_FASTQ_FILES_PER_PROCESS* on page 54 for more information. Default value is 3.*ANALYSIS Variables* on page 61

WARNING
Default for USE_BASES is Y*n, which means perform a single-read alignment and ignore the last base. If running ANALYSIS eland_pair, make sure to specify the USE_BASES option for two reads (for example USE_BASES Y*n,Y*n).

Optional Parameters

Table 2 configureAlignment Configuration File Optional Parameters

Parameter	Definition
SINGLESEED	If SINGLESEED is set to --singleseed, ELANDv2e aligns only in singleseed mode. Only available for ANALYSIS eland_extended and ANALYSIS eland_pair, for which multiseed alignment is default. See *ELANDv2 Algorithm Description* on page 131 for more information.
UNGAPPED	If UNGAPPED is set to --ungapped, ELANDv2e aligns only in ungapped mode. See *ELANDv2 Algorithm Description* on page 131 for more information.
INCREASED_SENSITIVITY	If you specify INCREASED_SENSITIVITY --sensitive, ELANDv2e aligns in full repeat mode. Semi-repeat resolution alignment is default. You can also use --INCREASED_SENSITIVITY=--sensitive on the command line. See *Repeat Resolution* on page 135 for more information.
OUT_DIR	Path to configureAlignment output. Defaults to <run_folder>/Aligned Note that there can be only one Aligned directory by default. If you want multiple Aligned directories, you will have to use this option to generate a different output directory.
DATASET_POST_RUN_COMMAND /yourPath/yourCommand yourArgs	Allows user-defined scripts to be run after all configureAlignment targets have been built. Invoked per barcode-lane for multiplexed samples, per lane for non-multiplexed samples. See also *Using DATASET_POST_RUN_COMMAND* on page 65.
EMAIL_LIST user@example.com user2@example.com EMAIL_SERVER mailserver	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 114.

Parameter	Definition
`EMAIL_DOMAIN example.com`	
`WEB_DIR_ROOT file://server.example.com/share`	Include hyperlinks with a specific prefix to the run folder.
`NUM_LEADING_DIRS_TO_STRIP`	Specifies the number of directories to strip from the start of the full run folder path before prepending the WEB_DIR_ROOT
`ELAND_RNA_GENOME_CONTAM`	Points to the folder containing a set of **contaminant sequences** for the genome—typically the mitochondrial and ribosomal sequences. The files must be in single FASTA format.
`ELAND_RNA_GENOME_ANNOTATION`	Path to transcripts mapping to the genome (refFlat.txt.gz or seq_gene.md.gz). See also *Using ANALYSIS eland_rna* on page 68.
`ELAND_RNA_GENE_MD_GROUP_LABEL`	The group label above specifies which assembly to use in the seq_gene file, and is found in column 13 of the file. seq_gene files can hold entries for multiple assemblies. Example: `ELAND_RNA_GENE_MD_GROUP_LABEL GRCh37.p2-Primary Assembly`.
`KAGU_PARAMS`	KAGU_PARAMS passes options to the alignmentResolver through the configureAlignment configuration file. For additional information, see *KAGU_PAIR_PARAMS and KAGU_PARAMS* on page 63.

Paired-End Analysis Options

Table 3 configureAlignment Configuration File Paired-End Analysis Options

Parameter	Definition
`ANALYSIS eland_pair`	Use the paired-end alignment mode of ELANDv2e to align paired reads against a target.
`USE_BASES Y*,nY*n`	Use all bases of the first read and ignore the first and last base of the second read.
`6:USE_BASES nY25`	Ignore the first base on both the first and second read; use 25 bases each and ignore any other bases for lane 6 only.
`KAGU_PAIR_PARAMS`	KAGU_PAIR_PARAMS passes options for paired-end runs to the alignmentResolver through the configureAlignment configuration file. For additional information, see *KAGU_PAIR_PARAMS and KAGU_PARAMS* on page 63.

For more information on USE_BASES syntax, see *USE_BASES Option* on page 61.

Specifying Analysis

Analysis can be specified by project, reference, sample, index, or lane, which is explained in this section.

Lane-Specific Analysis

By adding the lane number(s) followed by colon in front of an analysis option, you state that the analysis option is only for samples from that lane. The lane number is only valid for the configureAlignment settings on that same line.

For example, `567:ANALYSIS eland_extended` tells configureAlignment that eland_extended should be run on samples from lane 5, 6, and 7.

Sample-Specific Analysis

The config.txt file has some keywords that enable you to specify analysis for project, reference, sample, or index: PROJECT, REFERENCE, SAMPLE, and BARCODE. These keywords refer to the SampleProject, SampleRef, SampleID, and Index specified in the samplesheet.csv file located in the Unaligned directory of the run folder.

Lines starting with PROJECT, REFERENCE, SAMPLE, and BARCODE override any default settings specified in the config.txt file, but only for those samples for which the SampleProject, SampleRef, SampleID, or Index matches the PROJECT, REFERENCE, SAMPLE, or BARCODE. The override is only valid for the configureAlignment settings on that same line.

Example Sample-Specific Analysis

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

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

`with the following sample sheet:`

FCID	Lane	Sample ID	Sample Ref	Index	Descrip-tion	Control	Recipe	Operator	Sample Project
12345AAXX	1	sample1	human	ATCACG	desc1	N	R1	name	Proj1
12345AAXX	1	sample2	human	CGATGT	desc2	N	R1	name	Proj1
12345AAXX	2	sample3	rat	TTAGGC	desc3	N	R1	name	Proj2
12345AAXX	2	sample4	mouse	TGACCA	desc4	N	R1	name	Proj3

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

Combining Specificity

It is also possible to combine specific analyses, like in this example:

```
12: REFERENCE human ANALYSIS eland_pair
```

or

```
REFERENCE human 12: ANALYSIS eland_pair
```

Both of which tells configureAlignment to perform eland_pair analysis on the human reference samples from lanes 1 and 2.

Priority

If multiple specific settings conflict, configureAlignment uses the following order of priority:

1 PROJECT

2 REFERENCE

3 SAMPLE

4 BARCODE

5 Lane

6 Global settings

This means, PROJECT settings override any other settings, while REFERENCE settings can only be overruled by PROJECT settings, and so on.

WARNING
The attribute cannot be set for more than one scope at a time. In other words the following is not allowed:

```
PROJECT test BARCODE ACGT ANALYSIS eland_extended
```

Additional Examples

Some more examples are listed below:

- Standard flow cell-level variables

```
USE_BASES y*,y*
CHROM_NAME_VALIDATION off
ANALYSIS eland_rna
ELAND_FASTQ_FILES_PER_PROCESS 2
```

- Flow cell-level ELAND_GENOME variable set for all datasets with Reference HumanNCBI37ELAND

```
REFERENCE HumanNCBI37ELAND ELAND_GENOME
   /home/user/genomes/archive/UCSChg18/fasta
```

- Flow cell-level SAMTOOLS_GENOME variable set for all datasets with Reference AMPLICONS180111JRB_ELAND

```
REFERENCE AMPLICONS180111JRB_ELAND SAMTOOLS_GENOME
   /home/user/genomes/AMPLICONS180111JRB/AMPLICONS180111JRB.fa
```

- Flow cell-level SAMTOOLS_GENOME variable set for all datasets with Reference TSC1_ELAND

```
REFERENCE TSC1_ELAND SAMTOOLS_GENOME
   /illumina/user/TSC1/TSC1.fa
```

- Overrides global eland_rna ANALYSIS with eland_extended if the reference is not human

```
REFERENCE AMPLICONS180111JRB_ELAND ANALYSIS eland_extended
REFERENCE TSC1_ELAND ANALYSIS eland_extended
```

- If the reference is unknown (default for Undetermined barcode datasets), sets the analysis to none. Only affects lanes 1,2,3 and 4

```
1234:REFERENCE unknown ANALYSIS none
```

- Alternative way of ensuring Undetermined barcode datasets do not get aligned. Only affects lanes 5,6,7 and 8

```
5678:BARCODE Undetermined ANALYSIS none
```

Specific Scenarios

Below a number of scenarios are written out, assuming SampleSheet.csv has two projects: idxProj and noIdxProj

- Analyze only data for idxProj, not noIdxProj.
  - Disable analysis by default:

```
ANALYSIS none
```

  - Then the following analysis specifiations only affect sample sheet entries that have idxProj as their project field:

```
PROJECT idxProj ANALYSIS eland_pair
PROJECT idxProj USE_BASES Y*n,Y*n
PROJECT idxProj ELAND_GENOME x/y/z/G1
```

- Align only PhiX of idxProj, assuming there are 2 references for idxProj (hum and PhiX).
  - Disable analysis by default so that anything not explicity described is not analysed:

```
ANALYSIS none
```

  - Disable analysis for noIdxProj. This will take priority over REFERENCE-scope attributes below:

```
PROJECT noIdxProj ANALYSIS none
```

  - Set REFERENCE-scope variables so that when the data belongs to PhiX, they have an effect (noIdxProj will not be analysed as PROJECT-scope has higher priority):

```
REFERENCE phix ANALYSIS eland_pair
REFERENCE phix USE_BASES Y*n,Y*n
REFERENCE phix ELAND_GENOME x/y/z/GP
```

- Align only human for Lane 2, assuming 2 references for idxProj (human, PhiX).
  - Disable analysis by default so that anything not explicitly described is not analysed

```
ANALYSIS none
```

  Notice that everything below is set only for lane 2 so the rest of the data has 'ANALYSIS none' from above.

  - Disable analysis for noIdxProj. This will take priority over REFERENCE - scope attributes below

```
2: PROJECT noIdxProj ANALYSIS none
```

  - Set REFERENCE-scope variables so that when the data belongs to PhiX, they have an effect(noIdxProj will not be analysed as PROJECT-scope has higher priority).

```
2: REFERENCE hum ANALYSIS eland_pair
2: REFERENCE hum USE_BASES Y*n,Y*n
2: REFERENCE hum ELAND_GENOME x/y/z/GH
```

Samples Without Index

Unless otherwise specified in the sample sheet, samples without index will end up in the project folder Undetermined_indices, and in a sample folder named after the lane (e.g. Sample_lane1).

If you want to specify analysis for these samples without index other than the global analysis, you can use identifiers PROJECT Undetermined_indices or SAMPLE lane1.

NOTE
Normally you would want to use:

```
PROJECT Undetermined_indices ANALYSIS none
```

or

```
REFERENCE unknown ANALYSIS none
```

to avoid wasting CPU time on the Undetermined_indices data, which often is of poor quality.

configureAlignment Parameters Detailed Description

configureAlignment 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.

Table 4 ANALYSIS Variables

Variable	Alignment Program	Application	Description
`ANALYSIS eland_extended`	ELANDv2	Single reads	Aligns single-read data reads against a target using ELANDv2e 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 *configureAlignment Input Files* on page 48.
`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 67.
`ANALYSIS eland_rna`	ELANDv2	Single reads	Aligns each read against a large reference genome, splice junctions, and contaminants using ELANDv2e. For more information on ELAND_rna, see *Using ANALYSIS eland_rna* on page 68.
`ANALYSIS none`	None	Any application	Omits the indicated lane from the analysis. Setting the parameter 8:ANALYSIS none ignores lane 8.

WARNING
Default for `USE_BASES` is `Y*n`, which means perform a single-read alignment and ignore the last base. If running ANALYSIS eland_pair, make sure to specify the USE_BASES option for two reads (for example Y*n,Y*n).

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 *USE_BASES Option* on page 61. 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.

For this reason Illumina recommends that if 'n' bases of sequence are desired, 'n+1' cycles should be run.

- 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.
- Note that the symbol "I" for indexing is no longer accepted syntax for USE_BASES.

NOTE
Default is `USE_BASES Y*n`, which means perform a single-read alignment and ignore the last base. If running ANALYSIS eland_pair, make sure to specify the USE_BASES option for two reads (for example USE_BASES Y*n,Y*n).

The following table describes examples of USE_BASES options.

Table 5 USE_BASES Options

Option	Definition
`USE_BASES nYYY`	Ignore the first base and use bases 2–4.
`USE_BASES Y30`	Align the first 30 bases.
`USE_BASES nY30`	Ignore the first base and align the next 30 bases.
`USE_BASES nY30n`	Ignore the first base, align the next 30 bases, and ignore the last base.
`USE_BASES nY*n`	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.
`USE_BASES Y*n`	This means perform a single-read alignment and ignore the last base. Default for single-read alignment.
`USE_BASES Y*n,Y*n`	Perform a paired read alignment but ignore the last base of each read, 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.
`USE_BASES nY*,nY*`	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.
`USE_BASES nY*`	This means ignore the first base and perform a single-read alignment.
`USE_BASES n*,Y*n`	Ignore the first read and perform a single-read alignment with the second read, ignoring the last base.
`USE_BASES Y*n,n*`	Perform a single-read alignment with the first read, ignoring the last base, and ignore the second read.

Make Option

The --make option creates Aligned output directories and makefiles. Without the option, configureAlignment.pl will not create any directories and files and only operates in a diagnostic mode. You must specify this option to generate the Aligned 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 configureAlignment if a reanalysis of the same data is required.

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 all -j n
```

For more information on parallelization, see *Using Parallelization* on page 117.

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.out can be used by Illumina Technical Support for troubleshooting should problems arise

```
nohup make all -j n &
```

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

Starting Alignment for Read 1

If you want to start alignment before completion of the run, use the makefile target `r1`. This can be started once Bcl conversion for read 1 has finished (*Starting Bcl Conversion for Read 1* on page 37).

Set up a regular configureAlignment analysis, but run `make` using the `r1` target, for example:

```
nohup make -j 16 r1
```

NOTE
the `-j <n>` command line option is supported to indicate up to <n> processes in parallel.

Starting the Second Read

To start alignment of the second read, use the regular "make" command in the Aligned folder. Perform the following:

1 Move into the Aligned folder.

2 Type the regular "make" command:

```
make -j n
```

KAGU_PAIR_PARAMS and KAGU_PARAMS

The parameters KAGU_PARAMS (for all runs) and KAGU_PAIR_PARAMS (for paired-end runs) pass options to the alignmentResolver through the configureAlignment configuration file. For additional information, see *configureAlignment Configuration File* on page 55.

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

```
8:KAGU_PAIR_PARAMS --circular --muf 0
```

or

```
8:KAGU_PARAMS --mmaq 4
```

The following table describes the parameters.

Table 6 Parameters for KAGU_PAIR_PARAMS and KAGU_PARAMS

Parameter	Description
`--circular`	This causes alignmentResolver 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)
`--muf`	Minimum percentage of Unique Fragments. 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. alignmentResolver 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 non-orphaned 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 specifying --muf 0 For some applications it may be useful to switch off the pairing completely by specifying --muf 1.0
`--mcf`	Minimum percentage of Consistent Fragments. 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 • 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%.
`--mfaq`	Minimum Fragment Alignment Quality. 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 fragment, 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 export.txt.gz file. The default value is 4.
`--mmaq`	Minimum Mate Alignment Quality. 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 export.txt.gz 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 --

Parameter	Description
	min-single-read-alignment-score, the read pair is treated as a singleton. The alignment of the orphan read is unreliable enough to be ignored. The default value is 4.

Using DATASET_POST_RUN_COMMAND

DATASET_POST_RUN_COMMAND will be invoked at completion of DATASET alignment, and may be constructed of a single or multiple shell calls (for latter, separated by semicolon ;). Following variables, derived from SampleSheet, will be available (please use brackets properly): $(project) $(sample) $(barcode) $(lane).

Assuming we use the following SampleSheet:

```
FCID,Lane,S-
   ampleID,SampleRef,Index,Description,Control,Recipe,Operator,Project
B809UWABXX-1TILE-DMX,1,human1,human,GCCAAT,myTest,N,32+7,CB,testPRC1
B809UWABXX-1TILE-DMX,1,human1,human,CTTGTA,myTest,N,32+7,CB,testPRC2
B809UWABXX-1TILE-DMX,1,phix1,phix,TTAGGC,myTest,N,32+7,CB,testPRC1
B809UWABXX-1TILE-DMX,2,human1,human,GCCAAT,myTest,N,32+7,CB,testPRC1
B809UWABXX-1TILE-DMX,2,phix2,phix,TTAGGC,myTest,N,32+7,CB,testPRC2
B809UWABXX-1TILE-DMX,3,human1,human,TGACCA,myTest,N,32+7,CB,testPRC1
B809UWABXX-1TILE-DMX,4,phix4,phix,TTAGGC,myTest,N,32+7,CB,testPRC3
B809UWABXX-1TILE-DMX,5,human1,human,TGACCA,myTest,N,32+7,CB,testPRC1
B809UWABXX-1TILE-DMX,5,human5,human,GCCAAT,myTest,N,32+7,CB,testPRC2
B809UWABXX-1TILE-DMX,6,human1,human,CGATGT,myTest,N,32+7,CB,testPRC1
B809UWABXX-1TILE-DMX,7,human1,human,CGATGT,myTest,N,32+7,CB,testPRC1
B809UWABXX-1TILE-DMX,8,human1,human,CGATGT,myTest,N,32+7,CB,testPRC1
B809UWABXX-1TILE-DMX,8,human8,human,TGACCA,myTest,N,32+7,CB,testPRC2
```

Examples below illustrate use of DATASET_POST_RUN_COMMAND:

DATASET_POST_RUN_COMMAND limited to a PROJECT

Following config file for PROJECT selection:

```
ANALYSIS none
PROJECT testPRC1 ANALYSIS eland_extended
PROJECT testPRC1 USE_BASES Y*n
PROJECT testPRC1 ELAND_GENOME
   /illumina/scratch/iGenomes/PhiX/Illumina/RTA/Sequence/Squashed-
   PhiX-Illumina-RTA
PROJECT testPRC1 DATASET_POST_RUN_COMMAND echo $(project)
   $(sample) $(barcode) $(lane) >> out.DPRC.txt
```

will generate out.DPRC.txt in /Aligned folder:

```
testPRC1 phix1 TTAGGC 1
testPRC1 human1 CGATGT 7
testPRC1 human1 CGATGT 8
testPRC1 human1 CGATGT 6
testPRC1 human1 TGACCA 3
testPRC1 human1 TGACCA 5
testPRC1 human1 GCCAAT 1
testPRC1 human1 GCCAAT 2
```

DATASET_POST_RUN_COMMAND limited to a LANE

Following config file for LANE selection:

```
ANALYSIS none
1:ANALYSIS eland_extended
1:USE_BASES Y*n
1:ELAND_GENOME
   /illumina/scratch/iGenomes/PhiX/Illumina/RTA/Sequence/Squashed-
   PhiX-Illumina-RTA
1:DATASET_POST_RUN_COMMAND echo $(project) $(sample)
   $(barcode) $(lane) >> out.DPRC.txt
```

will generate following out.DPRC.txt in /Aligned folder:

```
testPRC1 phix1 TTAGGC 1
Undetermined_indices lane1 Undetermined 1
testPRC1 human1 GCCAAT 1
testPRC2 human1 CTTGTA 1
```

POST_RUN_COMMAND

You can also run POST_RUN_COMMAND from the make command lane, for example:

```
make all POST_RUN_COMMAND:='echo everything is done'
```

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, Gapped, Repeat Alignment

ANALYSIS eland_extended performs the following alignment features implemented in ELANDv2 and ELANDv2e:

- By default performs multiseed alignment by aligning consecutive sets of 16 to 32 bases separately.
- Uses a gapped alignment method to extend each candidate alignment to the full length that allows for gaps (indels) of up to 10 bases.
- Aligns reads in repeat regions using two new modes: semi-repeat resolution and full repeat resolution. Full repeat resolution is more sensitive and places more reads in repeat regions, but will result in longer run time. By default, ELANDv2e runs in semi-repeat resolution mode. Full repeat resolution can be turned on with the option INCREASED_SENSITIVITY.

Configuring ANALYSIS eland_extended

There are three parameters that affect the output of the alignment, ELAND_SEED_LENGTH1, ELAND_SEED_LENGTH2, and ELAND_MAX_MATCHES. Both parameters can be specified lane-by-lane.

The following table describes the parameters for ANALYSIS eland_extended.

Table 7 Parameters for ANALYSIS eland_extended

Parameter	Description
`ELAND_SEED_LENGTH1ELAND_SEED_LENGTH2`	By default, the first 32 bases of the read are used as a "seed" alignment. Setting ELAND_SEED_LENGTH1 to 25 will use 25 bases in read 1 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.
`ELAND_MAX_MATCHES`	By default, ANALYSIS eland_extended will consider at most ten alignments of each read. This can ELAND_MAX_MATCHES allows the maximum number of alignments considered per read to be varied between 1 and 255.

Both ANALYSIS eland_extended and ANALYSIS eland_pair produce export files that contain all read, quality value, and alignment information for the analysis.

For a detailed description of the export.txt.gz 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 ELANDv2e 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 export.txt.gz files 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, Gapped, Repeat, Orphan Alignment

ANALYSIS eland_pair performs the following alignment features implemented in ELANDv2 and ELANDv2e:

- By default performs multiseed alignment by aligning consecutive sets of 16 to 32 bases separately.
- Uses a gapped alignment method to extend each candidate alignment to the full length that allows for gaps (indels) of up to 10 bases.

- Aligns reads in repeat regions using two new modes: semi-repeat resolution and full repeat resolution. Full repeat resolution is more sensitive and places more reads in repeat regions, but will result in longer run time. By default, ELANDv2e runs in semi-repeat resolution mode. Full repeat resolution can be turned on with the option INCREASED_SENSITIVITY.
- Performs orphan alignment by identifying read pairs for which only one of the reads aligns. ELANDv2e then tries to align the other read in a defined window (by default 450 bp).

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 alignmentResolver. This is done by setting a parameter KAGU_PAIR_PARAMS in the configureAlignment configuration file. For additional information, see *KAGU_PAIR_PARAMS and KAGU_PARAMS* on page 63.

KAGU_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:KAGU_PAIR_PARAMS --circular --muf 0
```

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

Four sets of data files are needed:

- A genome sequence file.
- Fasta files of all chromosomes for on-fly splice junction generation.
- refFlat .txt.gz (from UCSC) or seq_gene.md.gz file (from NCBI) —as of CASAVA 1.7, eland_rna uses the refFlat.txt.gz or seq_gene.md.gz file to generate the splice junction set automatically. These files come from the following sources.
    - The refFlat.txt.gz file is available from UCSC
    - The seq_gene.md.gz file is available from NCBI.

  They should be provided gzip compressed, and should be from the same build as the reference files you are using for alignment. This negates the need to provide separate splice junction sets as in previous version of CASAVA.
- A set of contaminant sequences for the genome - typically the mitochondrial and ribosomal sequences.

Description of the eland_rna Algorithm

The algorithm aligns the reads to each of three targets:

- Contaminants
- Genome
- Splice junctions; alignments need to span splice junction

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 and/or splice junctions:
   - If there is a unique alignment to the genome or splice junctions then that alignment is printed.
   - If there are multiple possible alignments to the genome and splice junctions then the read is marked as 'RM' and discarded as above.
3. 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, Repeat Alignment

ANALYSIS eland_rna performs the following alignment features implemented in ELANDv2 and ELANDv2e:

- By default performs multiseed alignment by aligning consecutive sets of 16 to 32 bases separately.
- Aligns reads in repeat regions using two new modes: semi-repeat resolution and full repeat resolution. Full repeat resolution is more sensitive and places more reads in repeat regions, but will result in longer run time. By default, ELANDv2e runs in semi-repeat resolution mode. Full repeat resolution can be turned on with the option INCREASED_SENSITIVITY.

Running an eland_rna Analysis

The configureAlignment configuration file specifies how the sequences from a flow cell are processed, which is described in *configureAlignment Configuration File* on page 55. The ANALYSIS parameter within the configureAlignment 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_ANNOTATION /data/Genome/ELAND_
   RNA/Human/refFlat.txt.gz
ELAND_RNA_GENOME_CONTAM /data/Genome/ELAND_RNA/Human/MT_Ribo_
   Filter/
```

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

The following table describes the parameters for ANALYSIS eland_rna.

Table 8 Parameters for ANALYSIS eland_rna

Parameter	Description
ELAND_GENOME	Must point to the reference genome, just as for a standard ELANDv2e analysis.
ELAND_RNA_GENOME_ANNOTATION	Must point to the refFlat.txt.gz file (gzip compressed) or seq_gene.md.gz file (gzip compressed).

Parameter	Description
`ELAND_RNA_GENOME_CONTAM`	Must point to the files of ultra-abundant sequences (generally ribosomal and mitochondrial). Any read that hits to these is ignored.

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:

```
124:ANALYSIS eland_rna
124:ELAND_GENOME /data/Genome/ELAND/hg18/
124:ELAND_RNA_GENOME_ANNOTATION /data/Genome/ELAND_
   RNA/Human/refFlat.txt.gz
124:ELAND_RNA_GENOME_CONTAM /data/Genome/ELAND_RNA/Human/MT_
   Ribo_Filter/
```

- The output file export.txt.gz has the same format as those generated by eland_extended; for a description see *Export.txt.gz* on page 77. The existing code 'RM' ('repeat masked') denotes all reads that hit to abundant sequences or with any other unresolvable ambiguity.

KEEP_INTERMEDIARY Option

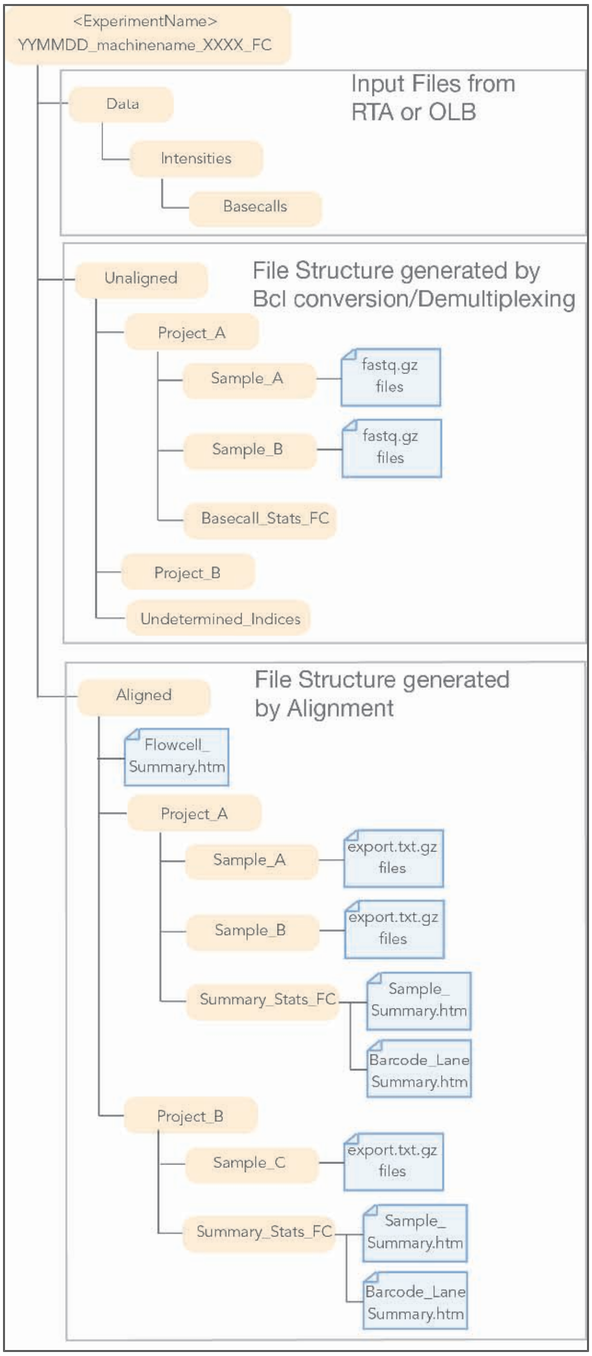
The option KEEP_INTERMEDIARY tells CASAVA not to delete the intermediary alignment files in the alignment Temp dir after alignment is complete. This is a `make` option, and needs to be used when you run make. For example:

```
nohup qmake -cwd -v PATH -q genexpr.q -- -j32 all KEEP_
   INTERMEDIARY:=yes&
```

configureAlignment Output Files

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

Figure 12 Run Folder after configureAlignment Analysis



NOTE

There can be only one Aligned directory by default. If you want multiple Aligned directories, you will have to use the option `OUT_DIR` to generate a different output directory.

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.

- For each project, a **Sample_Summary.htm** file is produced, which contains comprehensive results and performance measures of your analysis run for a project per sample. It is located in the Aligned/project folder and provides an overview of quality metrics for a project with links to more detailed information in the form of pages of graphs.
- For each project, a **Barcode_Lane_Summary.htm** file is produced, which contains comprehensive results and performance measures of your analysis run for a project per barcode and lane. It is located in the Aligned/project folder and provides an overview of quality metrics for a project with links to more detailed information in the form of pages of graphs.
- For each run, a **FlowCellSummary.htm** file is produced, which contains comprehensive results and performance measures of your entire analysis run across all projects. It is located in the Aligned folder.

Sample_Summary Page

For each sample, a Sample_Summary.htm file and Barcode_Lane_Summary.htm file is produced, which contains comprehensive results and performance measures of your analysis run for a project. It is located in the Aligned/Project_<ProjectName>/Summary_Stats folder and provides an overview of quality metrics for a run per sample with links to more detailed information in the form of pages of graphs.

The metrics are described below.

Project Summary

The Project Summary contains general project information:

- **Project Name**
- **Machine** name
- **Run Folder**—full path to the run folder
- **Flow Cell ID**
- **Platform**—instrument type
- **Control Software** and version,
- **Primary Analysis** software and version
- **Secondary Analysis** software and version

Project Results Summary

This table displays a summary of project-wide performance statistics for the run:

- **Clusters**—Original number of detected clusters.
- **Clusters (PF)**—Number of clusters that passed quality filtering.
- **Yield**—The sum of all bases (in Mb) in clusters that passed filtering for the entire project.

Barcode-Lane Summary

The Barcode-Lane Summary records information about the barcoded samples in each flow cell lane and the analysis that has been specified for it.

- **Barcode-Lane**—The identity of the barcoded sample in a lane. The identity follows the following format: `<SampleName>_<Barcode>_<Lane>`.
- **Sample**—Sample name.
- **Barcode**—The sequence of the barcode (index).
- **Lane**
- **Species**—The reference sequence against which was 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.
- **Analysis 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.
- **Num Tiles**—The number of tiles from the lane that are used in the analysis.
- **Genome Directory**— Full path to the genome directory.

Sample Results Summary

This table displays basic data quality metrics for each sample, displayed on the Summary - sample page.

- **Sample Yield**—The sum of all bases (in Mb) in clusters that passed filtering for the sample.
- **Clusters (raw)**—The number of clusters detected by the image analysis module.
- **Clusters (PF)**—The number of detected clusters that meet the filtering criterion.
- **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 reads passing filter that were uniquely aligned to the reference. For eland_rna it is number of PF reads aligned to the genome and splice junctions. Reads aligned to abundant sequences and masked by eland_rna do not participate in this number.
- **Alignment Score (PF)**—The average filtered read alignment score (reads with multiple or no alignments effectively contribute scores of 0). For phiX spikes, the number of reads aligning to PhiX is small and therefore the reported alignment score (small number of aligned reads divided by total number of PF reads) is usually small.
- **Mismatch Rate (PF)**—The percentage of called bases in aligned reads that do not match the reference.
- **% >=Q30 bases (PF)**—Yield of bases with Q30 or higher from clusters passing filter divided by total yield of clusters passing filter.
- **Mean Quality Score (PF)**—The total sum of quality scores of clusters passing filter divided by total yield of clusters passing filter.

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 Sample Summary

This displays more detailed quality metrics for each sample.

- **Clusters (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.
- **% Mismatch Rate (raw)**—The percentage of called bases in aligned reads from all detected clusters that do not match the reference.
- **% PF Clusters**—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 reads passing-filter that were uniquely aligned to the reference.
- **% Mismatch Rate (PF)**—The percentage of called bases in aligned reads passing-filter that do not match the reference.
- **% >=Q30 bases (PF)**—Yield of bases with Q30 or higher from clusters passing filter divided by total yield of clusters passing filter.
- **Mean Quality Score (PF)**—The total sum of quality scores of clusters passing filter divided by total yield of clusters passing filter.

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.

Bustard Summary File

Below the expanded sample summaries is a link to the BustardSummary.xml file, generated during bcl conversion. For a description of the data, see *DemultiplexedBustardSummary.xml File* on page 42.

Per-Tile Statistics

Below the link to the BustardSummary.xml file is a link to a file containing per-tile statistics. The displayed metrics are similar to the expanded Lane 1 : Read 1 tables in the CASAVA 1.8 configureAlignment summary files.

IVC Plots

Next is a link to IVC plots. The IVC.htm file (Intensity versus Cycle) contains plots that display lane averages for samples:

- **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.

All Intensity Plots

The link to All.htm file gives a 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.

Mismatch Graphs

The Mismatch Graphs link leads to a file with graphs of error rates 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.

ELANDv2e is capable of aligning against large genomes, such as human, in reasonable time. However, it allows only two errors per seed. This means that error rates based on ELANDv2e alignments are underestimated.

Mismatch Curves

The Mismatch Curves link leads to a file with graphs of 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.

Additional Paired Statistics

For samples for which eland_pair analysis was performed, there is a table called Additional Paired Statistics. This table provides 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." When this happens, CASAVA builds for these paired reads cannot be performed without first rerunning configureAlignment.pl and adjusting parameters such as `--min-percent-unique-pairs` and `--min-percent-consistent-pairs` to produce acceptable paired data and summaries."

The following sections are displayed in Additional Paired Statistics:

- **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. Unlike these short insert pairs that have a predominance in opposite and inwardly facing read pairs (R+: > R1 R2 <), the large insert mate pair libraries expect to produce a predominance in opposite and outwardly facing read pairs (R-: < R2 R1 >). High frequencies of paired reads having the same orientation (F-: > R2 R1 > or F+: > R1 R2 >) may be indicative of a sample preparation problem, or evidence of an adapter read through problem found when the read lengths are long relative to the library insert size.
- **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.
- **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.

Barcode_Lane_Summary Page

The Barcode_Lane_Summary.htm file provides similar metrics as the Sample_Summary page, with the following differences:

- The results are displayed for each barcoded sample in a lane, instead of for samples.
- Tables are named accordingly: the equivalents for the Sample Results Summary and Expanded Sample Summary are named Barcode Lane Results Summary and Expanded Barcode Lane Summary
- The Barcode_Lane_Summary page contains a Barcode-Lane Summary, described below.

For a description, see the equivalent section in the Sample_Summary Page description (*Barcode-Lane Summary* on page 73).

Flow Cell Summary

For each run a FlowCellSummary_FCID.htm file is produced, which contains the Project Summaries and Sample Results Summaries of all projects. This provides an overview of the most relevant metrics for the entire run. It is located in the Aligned folder.

For a description of Project Summaries and Sample Results Summaries, see *Sample_Summary Page* on page 72.

Analysis Results

The output files for each lane of a flow cell are named using the format export.txt.gz. For paired-read analysis, there are two parallel output files, one for each read. The files are named using the format : <sample name>_<barcode sequence>_L<lane>_R<read number>.<0-padded 3-digit set number>_export.gz. The files are found in the Aligned/Project_ID/SAmple_ID folder of a finished analysis run.

Export.txt.gz

The standard naming format for *_export.txt.gz files is <sample name>_<barcode sequence>_L<lane>_R<read number>.<0-padded 3-digit set number>_export.gz, like in: NA10831_ATCACG_L001_R1_001_export.txt.gz. The *_export.txt.gz files are saved as compressed gzipped files. The content of the *_export.txt.gz files is described below; not all fields are relevant to a single-read analysis.

> NOTE
> The old Illumina-specific transformation (ASCII offset of 64) will still be used in the export files, but export.txt.gz is meant to be an internal file format.

1. Machine (Parsed from run folder name)
2. Run Number (Parsed from run folder name)
3. Lane
4. Tile
5. X Coordinate of cluster. As of RTA 1.6, OLB 1.6, and CASAVA 1.6, 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.

6 Y Coordinate of cluster. As of RTA 1.6, OLB 1.6, and CASAVA 1.6, 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.

7 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.

8 Read number (1 for single reads; 1 or 2 for paired ends or multiplexed single reads; 1, 2, or 3 for multiplexed paired ends)

9 Called sequence of 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.

- "ee:ss:dd": too many hits, where ee is the number of exact hits, ss is the number of hits with a single mismatch and dd is the number of hits with a double mismatch
- NM: no match
- QC: QC failure
- RM: repeat masked, for example match against abundant sequences

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 orphan 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 orphan read, this value is 0.
- If partner aligns to a different chromosome and/or 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 filtering? N - No, Y - Yes.

Additional configureAlignment Output Files

s_N_TTTT_rescore.txt

The "txt" score and rescore files are produced by tile. The corresponding XML summaries are by lane. 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.

rnaqc.txt

The output file rnaqc.txt files in the Aligned folder provides the following information on alignment distribution for eland_rna:

1 totalClusters—number of total clusters.

2 PFClusters—number of clusters passing purity filter.

3 Usable—number of reads passing filter and aligned uniquely to the genome plus splice junction.

4 QC—number of reads passing filter that were not aligned due to too many bases not called (QC in the 11th field of the export file).

5 noMatch—number of reads passing filter that did not match anything (including repeat-masked); these reads have NM label in the 11th field of the export file.

6 repeatMasked—number of reads passing filter that were masked by eland_rna (RM label in the 11th field of the export file). These are reads mapping to abundant sequences and reads that do not have unique alignments to the genome or splice junctions.

> NOTE
> Sum of Usable, QC, noMatch, and repeatMasked reads is equal to number of reads reported in PFClusters.

7 spliceUsable—number of reads passing filter aligned uniquely to the splice junctions;

8 genomeUsable—number of reads passing filter aligned uniquely to the genome;

NOTE
Sum of spliceUsable and genomeUsable is equal to Usable.

9 In the last rows numbers are provided for number of passing filter reads aligned to each reference sequence file within the AbundantSequences directory. The names are derived from the fasta headers (up to first space) used to list each reference in the multifasta abundant sequences file. If you want a more descriptive names, like ribosomal, E.coli, or phiX, you should modify fasta headers in the abundant sequences file.

NOTE
Difference between repeatMasked and sum of all abundant sequences gives the number of reads that do not have unique alignments.

contam_export.txt.gz

Contains unique alignments to sequences in the CONTAM directory, in the export format (see *Export.txt.gz* on page 77).

Intermediate Output Data Files

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

CAUTION
Do not use the intermediate files as input for custom scripts. These files may not be generated anymore in future CASAVA versions.

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
- <sample name>_<barcode sequence>_L<lane>_R<read number>.<0-padded 3-digit set number>_name.txt

Table 9 Intermediate Output File Descriptions

Output File	configureAlignment Analysis Mode	Description
*_eland_extended.txt	ANALYSIS eland_extended	Contains the corrected alignment positions and the full alignment descriptions for >32 base reads. This file is not purity filtered.
*_eland_extended.txt	ANALYSIS eland_pair	
*_extended_contam.txt	ANALYSIS eland_rna	Alignments to the ELAND_RNA_GENOME_CONTAM.
*_extended_splice.txt	ANALYSIS eland_rna	Alignments to the splice junctions.

Table 10 Intermediate Output File Formats

Output File	Format
s_N_TTTT_align.txt s_N_TTTT_realign.txt s_N_TTTT_prealign.txt	Deprecated sequence alignment format. Space-separated text values: 1. Sequence 2. Best score 3. Number of hits at that score 4. The following columns only appear if hits equal 1 (a single, unique match) 5. Target:pos

Output File	Format
	6. Strand 7. Target sequence 8. Next best score

Interpretation of configureAlignment 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 Pages

After analysis is complete, check the FlowCellSummary_FCID.htm file, Sample_Summary.htm, and Barcode_Lane_Summary.htm files. These provide metrics per flow cell, sample, and barcode-lane, respectively. For a description of the tables found, see *Flow Cell Summary* on page 77., *Sample_Summary Page* on page 72, and *Barcode_Lane_Summary Page* on page 77.

The key parameters that you should examine are listed in *SummaryTab* on page 19 and in the following sections.

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.

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.

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 ELANDv2e analysis using only the “good” cycles before the instrument problem.
	Contamination from other genetic material resulting in an inability to align data	Align a few sample tiles. 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.

Percentage Mismatch Rate of Clusters Passing Filters

This value should be as low as possible, but it is very dependent on read-length. If there is a sudden rise beyond cycle 32, then it is likely that ELANDv2e 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.

IVC.htm

For a detailed description of the plots found in the IVC.htm file, see *IVC Plots* on page 75.

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

All.htm and Mismatch.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.

Condition	Possible Cause
Tile variability	Bubbles Rapid focus fluctuations Dirty flow cell surface
Rising mismatch rates (Rates will always rise eventually at high read-lengths)	Low intensity at start High decay rate High phasing or prephasing Adapter read through
High, but constant mismatch rates from cycle 1	Genomic contamination

Running ELAND as a Standalone Program

You can run ELAND without the rest of configureAlignment as a post-analysis step. ELAND can be run as a standalone program for the following reasons:

- To test the effect of different filter parameters
- To test alignment targets
- To test applications that read export files

To run ELAND as a standalone program, use the script Path/to/CASAVA1.8/bin/ELAND_standalone.pl.

```
Path/to/CASAVA1.8/bin/ELAND_standalone.pl -if read1.fastq -if
   read2.fastq
-ref /lustre/data01/Mondas_software/Genomes/E_coli_ELAND
```

Table 11 Required Parameters for ELAND_standalone.pl

Option	Short Form	Description
--input-file <input file>	-if	Specify at least one file for single-reads and two files for paired-reads (mandatory)
--ref-sequences <path to genome dir>	-ref	Full path of a genome directory (mandatory)

Table 12 Options for ELAND_standalone.pl

Option	Short Form	Description
--bam		Enables BAM output.
--base-quality <value>	-bq	Assumes all bases have this quality when in fasta mode (default is set to 30)
--copy-references	-cr	Copies the references to the output directory. Use this option if your reference sequence directory is write-protected.
--force		Forces existing output files to be overwritten.
--input-type <input format>	-it	Type of input file (FASTQ, FASTA, export, or qseq).
--log <path to log>	-l	The path to the log file. Default =ELAND_standalone.log.
--output-directory<output dir>	-od	The output directory.
--output-prefix <prefix>	-op	Produces a set of output files with a prefix of this value (default value is “reanalysis”)
--kagu-options <"options">	-ko	Indicates paired-read analysis parameters to pass to alignmentResolver. e.g. -ko "-c" enables circular reference sequence support. Multiple arguments must be contained in quotation marks.
--remove-temps	-rt	removes all files except exports, BAM files, and log files upon successful completion.
--seed-length <value>	-sl	Length of read substring (seed) used for ELAND alignment (defaults to the lower of read-length and 32). Use twice for paired-end data sets.
--use-bases <value>	-ub	Expanded mask to apply to the FASTQ file two values if paired analysis Defaults to Y*n
--help	-h	Shows help text.

NOTE
The orphan aligner is always enabled when performing paired-end analysis with ELAND standalone – just like configureAlignment.

The orphan aligner is always enabled when performing paired-end analysis with ELAND standalone – just like GERALD.

Running ELAND as a standalone program does not perform all of the various steps that are included during a configureAlignment run. The most important differences are:

- ELAND standalone does not generates many of the statistics
- ELAND standalone is not massively parallel like configureAlignment

If you require any or all of the above, it is best to create a modified config file to align to a different genome, and rerun configureAlignment. For more information, see *Running configureAlignment* on page 53.

FASTQ Format

Any FASTQ file will be supported, but the CASAVA FASTQ file format is optimal for populating the appropriate fields. The format is: (note the space between y-pos and read number:

```
@<instrument-name>:<run ID>:<flowcell ID>:<lane>:<tile>:<x-
   pos>:<y-pos> " " <read number>:<is filtered>:<control
   number>:<barcode sequence>
```

The elements are described below.

Element	Requirements	Description
@	@	Each sequence identifier line starts with @
<instrument-name>	Characters allowed: a-z, A-Z, 0-9	Instrument-name
<RunID>	Characters allowed: a-z, A-Z, 0-9	Run ID
<flowcell ID>	Characters allowed: a-z, A-Z, 0-9	flowcell ID
<lane>	Numerical	Lane number
<tile>	Numerical	Tile number
<x-pos>	Numerical	X coordinate of cluster
<y-pos>	Numerical	Y coordinate of cluster
<read number>	Numerical	Is usually 1 or 2 for paired-end reads or 1 for single-end reads. There field can support more than two reads.
<is filtered>	Y or N	is Y if the read is filtered, N otherwise
<control number>	0	Is 0 when none of the control bits are on (reserved for future use)
<barcode sequence>	ACGT	Represents the USE_BASES masked barcode sequence, empty otherwise

An example is shown below

```
@EAS139:136:FC706VJ:2:5:1000:12850 1:Y:18:ATCACG
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
+
BBBBCCCC?<A?BC?7@@???????DBBA@@@@A@@
```

Chapter 5

Variant Detection and Counting

Introduction

This chapter explains how to use CASAVA1.8 to detect Single Nucleotide Polymorphisms (SNPs) and insertions/deletions (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 configureAlignment.

The CASAVA build process is divided into several modules (or targets), each of which completes a major portion of the post-alignment analysis pipeline. The first module, 'sort', bins aligned reads into separate regions of the reference genome, sorts these reads and optionally removes PCR duplicates (for paired-end reads) and finally converts these reads into BAM format. In a paired-end analysis the next module, 'assembleIndels', is used to search for clusters of poorly aligned and anomalous reads. These clusters of reads are de-novo assembled into contigs which are aligned back to the reference to produce candidate indels. Subsequently, the 'callSmallVariants' module uses the sorted BAM files and the candidate indels predicted by the assembleIndels module to perform local read realignment and genotype SNPs and indels under a diploid model. In an RNA-Seq build the 'rnaCounts' module will also be run to calculate gene and exon counts. Other optional modules can be added to the build process to perform additional functions.

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 files from configureAlignment as input and produce SNP and indel calls, but note that gapped alignments are required for indel calls in RNA and single-ended DNA builds. 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 110.

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 SNP and indel 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.

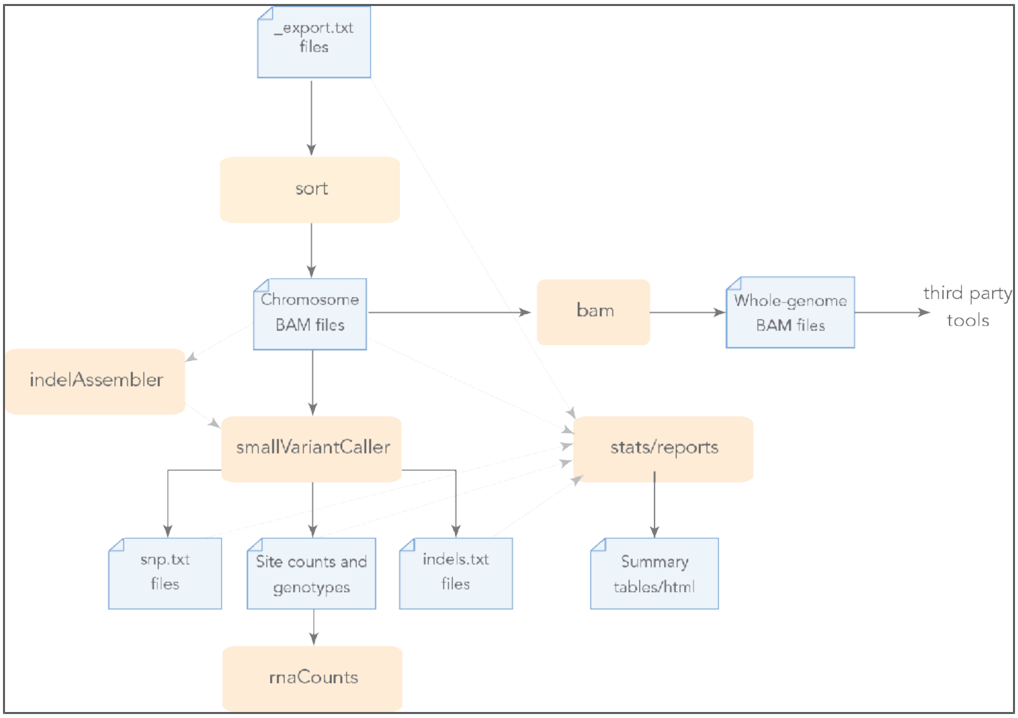
TIP
As long as a gapped alignment is performed, small indels (up to 10 nucleotides) can be called from RNA-Sequencing builds.

NOTE
RNA Sequencing only supports single-read runs.

Post-Alignment Workflow

The CASAVA workflow for variant detection and counting is illustrated below

Figure 13 CASAVA Variant Detection and Counting Workflow



Post-sort BAM

CASAVA has a number of changes in the way files are handled in the post-alignment workflow:

- CASAVA 1.8 operates entirely on BAM files after the sort module in the post-alignment workflow has completed, sorted.txt files are no longer created or stored.. This significantly reduces the build size: the combined changes in the new variant caller and BAM files for CASAVA reduce the human DNA Sequencing post-alignment builds size by 75-80%.
- Archival mode: CASAVA can be run so that all input reads are retained in the build in their entirety. Variant calling and RNA counting results are identical in archival and non-archival versions of the build.
- Fast creation of whole genome BAM files: After the sort module has completed, 30-40x whole genome BAM files can now be created and indexed in approximately 1 hour.
- Spliced alignments are now represented in BAM using the same format as TopHat, allowing visualization of splice junctions in IGV.

Archival Build

Archival builds, turned on with the option `--sortKeepAllReads`, include all reads given as input to the build in their entirety:

- Purity filtered and duplicate reads are stored in the primary BAM files with the appropriate bit settings to identify them. These will be ignored by variant calling and RNA read counting.
- To handle various types of unmapped reads, the CASAVA 1.7 "NMNM" directory has been renamed as "notMapped". Reads within this directory are classified into separate BAM files for the following categories: noMatch, qcFail, nonUnique, repeatMasked, mixed.
- In any situation where reads were trimmed in CASAVA 1.7 they are now soft-clipped. In some cases, where a read would be removed in non-archival mode due to some anomalous condition, that read is now marked as unmapped and stored in the build instead. Note that the small variant caller is designed to preserve any soft-clip regions from an input read (though it may expand them as part of local realignment).

NOTE
This is independent of the bam files produced by the target bam,which aggregates all reads into a single BAM file with chromosome re-labeling (see *Targets* on page 94).

- This is independent of the archival bam file, which can be produced using the option `--sortKeepAllReads`(see *Archival Build* on page 88).

Methods

CASAVA uses a number of methods to efficiently assemble indel candidates, call SNPs and indels, and provide counts. This section explains the methods.

Variant Detection

Post-alignment CASAVA performs variant detection using two modules:

- The assembleIndels module (Grouper) detects candidate indels using singleton/orphan and anomalous read pairs. The assembleIndels module works well for detecting larger indels. The candidate indels detected by the assembleIndels module are passed on to the small variant caller for consolidation and genotyping.
- The callSmallVariants module genotypes and provides quality scores for SNPs and indels. Indels can be called from candidate indel evidence provided by both ELAND gapped-read alignments (for smaller indels) and from the assembleIndels module (for larger indels).

For each SNP or indel call the probability of both the called genotype and any non-reference genotype is provided as a quality score (Q-score). Reads are re-aligned around candidate indels to improve the quality of SNP calls and site coverage summaries.

The callSmallVariants module also generates files which summarize the depth and genotype probabilities for every site in the genome. As a final step it produces tables and html-formatted reports of SNP and indel calls .

assembleIndels Algorithm

The assembleIndels module (Grouper) runs only during paired read DNA CASAVA builds. It uses orphan reads and, as of CASAVA 1.8, anomalous read pairs to detect indels.

Grouper detects indels in five stages:

1. Compute clusterings of non-aligned 'orphan reads'.
2. Compute clusterings of anomalous read pairs, with an insert size that is anomalously large (possible deletion) or small (possible insertion).
3. Combine clusters that appear to correspond to the same event.
4. Assemble them into contigs.
5. 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.

Variant Caller Methods

The callSmallVariants module calls SNPs and small indels from both the sorted alignment files (sorted.bam) and optionally also from the candidate indel contigs produced by assembleIndels. The procedure is outlined below.

- Read in read alignments and candidate indel contigs. Filter out read alignments based on quality checks, paired-end anomalies, or ELAND alignment score. Filter out contig alignments containing adjacent insertion/deletion events.
- Consolidate indel evidence from read and contig alignments to produce a set of candidate indels.
- Perform local read realignment using candidate indels.

- Call indels based on the set of alignments for each read which intersect/include a candidate indel.
- Select most likely read realignment for subsequent site counting and genotyping.
- Further filter individual basecalls based on mismatch density or ambiguity ('N').
- Use all remaining base calls to predict site genotypes and SNPs.
- Filter to remove SNP and indel calls near the centromeres and within high-copy number regions.

readBases Counting Method

As of version 1.6, CASAVA uses the readBases counting method. This method is for exon 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 exon base coverage value.

NOTE
Before counting CASAVA split alignments to the splice junction to two shorter genomic reads.

Counts for splice junctions are provided for convenience and correspond to the number of reads that cover the junction point. Bases within reads aligned to the junction are counted only once in the exon counts.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 Kilobase 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.

Variant Caller and Counting Detailed Description

For a detailed description of the variant caller algorithm, see *Variant Detection* on page 139.

Variant Detection Input Files

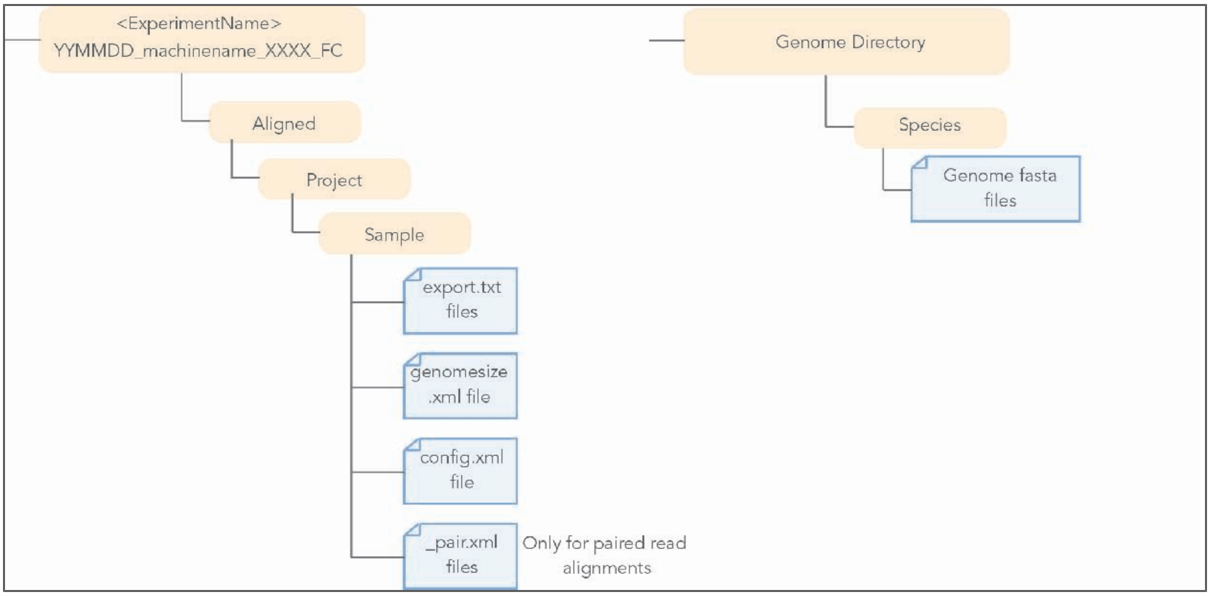
The variant detection and counting input files come from the configureAlignment module using the following eland modules:

- eland_extended (*configureAlignment Input Files* on page 48) for single-read DNA sequencing projects.
- eland_pair (*Using ANALYSIS eland_pair* on page 67) for paired-end DNA sequencing projects.
- eland_rna (*Using ANALYSIS eland_rna* on page 68) for single-read RNA sequencing projects (paired-end RNA sequencing projects are not supported).

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

In addition, CASAVA variant detection and counting uses annotation files (genome sequence files and refFlat.txt.gz or seq_gene.md.gz file) .

Figure 14 CASAVA Input Files



export.txt.gz Files

The export.txt.gz files contain the aligned sequence information from the configureAlignment module, and are required.

The export.txt.gz files are tab delimited text files; for a detailed description, see See "Output File Formats".

Run.conf.xml

The run.conf.xml file provides the location of the configureAlignment data (export.txt.gz) for each flow cell run, and describes their properties of each flow cell run. There is one run section for each flow cell run, one set section for each Aligned 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 buildDirectory/conf/

Pair.xml

The pair.xml file provides information about pair distribution in the configureAlignment output (only for paired-end sequencing). Pair.xml is required for paired samples to be treated as paired. You do not need to point to it specifically, since it should have been placed in the Aligned/Project/Sample folder for your sample by configureAlignment.

Genomesize.xml

The *_genomesize.xml file contains names of reference genomes, and is required for variant detection. You do not need to point to it specifically, since it should have been placed in the Aligned/Project/Sample folder for your sample by configureAlignment.

Reference Genome

CASAVA uses a reference genome in FASTA format. Both single sequence FASTA and multi-sequence FASTA genome files are supported.

Genome sequence files for most commonly used model organisms are available through iGenome (*Getting Reference Files* on page 126).

Single Sequence FASTA Files

CASAVA accepts single-sequence FASTA files as genome reference, which should be provided unsquashed for both alignment and post-alignment steps. The chromosome name is derived from the file name.

Direct CASAVA to a folder containing the FASTA files using the option `--refSequences=PATH` for variant detection and counting.

Multi-Sequence FASTA Files

As of version 1.8, CASAVA accepts a multi-sequence FASTA file as genome reference. This should be provided as a single genome, SAM compliant, unsquashed file, for both alignment and post-alignment steps. The chromosome name is derived directly from the first word in the header for each sequence.

Direct CASAVA to multi-sequence FASTA file using the option `--samtoolsRefFile=FILE` for variant detection and counting.

WARNING
GenomeStudio does not support the use of multi-sequence FASTA files. Therefore, if you want to analyze your output in GenomeStudio, we recommend using single sequence FASTA reference files.

Chromosome Naming Restrictions

CASAVA does not accept the following characters in the chromosome name:

```
  - ? ( ) [ ] / \ = + < > : ; " ' , * ^ | &
```

refFlat.txt.gz or seq_gene.md.gz File

CASAVA 1.8 generates the non-overlapping exon coordinates set automatically using the refFlat.txt.gz file (from UCSC) or seq_gene.md.gz file (from NCBI). They should be from the same build as the reference files you are using for alignment, and are available from iGenome for the most common model organisms (*Getting Reference Files* on page 126).

Running Variant Detection and Counting

The major use cases for running CASAVA variant detection and counting are listed below.

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

Major Use Cases

- SNP and Indel Calling

  To run CASAVA with callSmallVariants and assembleIndels, enter:

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

- SNP and Indel calling without large-indel assembly

  To run CASAVA with callSmallVariants, but without assembleIndels, enter:

  ```
  /path-to-CASAVA/bin/configureBuild.pl --targets all
     noassembleIndels --variantsSkipContigs [options]
  ```

- SNP and Indel calling, Single-end Build

  To run CASAVA with callSmallVariants for a single-end build, enter:

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

- RNA Sequencing

  To run CASAVA for RNA Sequencing, enter:

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

Other Use Cases

- Help

  To get the CASAVA Help for callSmallVariants, enter:

  ```
  /path-to-CASAVA/bin/configureBuild.pl --help callSmallVariants
  ```

- Rerun callSmallVariants

  In any pre-existing build in which the sort module was previously completed (and the assembleIndels module for a paired end build), Small variant calling may be rerun using:

  ```
  /path-to-CASAVA/bin/configureBuild.pl -od $PROJECT_DIR --
     targets callSmallVariants
  ```

  NOTE
  We only support datasets originated from the same version of the software.

- Generate BAM File with Altered Alignments

  An advanced option useful for variant diagnosis is to create BAM files for those reads which had their alignments altered by the variant caller during local realignment. This may be done by adding the command `--variantsWriteRealigned` to any command-line which runs the variant caller.

Targets

The targets that define CASAVA analysis are listed in the tables below.

Table 13 Targets for Variant Detection and Counting

Option	Description
all	Run all pre-configured targets for the given analysis type (default), except for

Option	Description
	target bam.
sort	Bin reads and sort by position; Remove PCR duplicates for paired-end data.
assembleIndels	Search for candidate indels from paired-end reads via de-novo assembly of contigs which are aligned back to the reference.
callSmallVariants	Call SNPs and indels from locally re-aligned reads. Candidate indels from the assembleIndels target can be used to improve indel results. See also *Target callSmallVariants Usage* on page 95.
rnaCounts	Calculate gene and exon counts in an RNA-Seq build.
bam	Aggregate all reads into a single BAM file with chromosome re-labeling. This target is not part of target all, and is therefore not done by default. This BAM file is independent of the archival bam file, which can be produced using the option `--sortKeepAllReads`(see *Archival Build* on page 88).
gsIndex	Pre-compute Genome Studio linear index for all reads in the build.

If you run a target other than the default target (`all`), make sure to read the help written for the target. This will help you identify any dependencies for the target you want to run.

Target help can be accessed by typing:

```
Path/to/CASAVA/bin/configureBuild.pl --help <target>
```

NOTE
Prefixing any target name with `no` will exclude it from the targets list. Example:

```
path-to-CASAVA/bin/configureBuild.pl --targets all
noassembleIndels --variantsSkipContigs [options]
```

Target callSmallVariants Usage

The callSmallVariants module is designed to use the results of the assembleIndels module if available, so a new paired-end build could be run with the following minimum set of targets:

```
--targets sort assembleIndels callSmallVariants
```

If assembleIndels (Grouper) cannot be run, an alternative workflow is:

```
--targets sort callSmallVariants --variantsSkipContigs
```

NOTE
To have the plugin provide a BAM file containing all reads which have had their alignments altered during realignment, add the following to the configuration command line:

```
--variantsWriteRealigned
```

These reads will appear in the file "sorted.realigned.bam" in the chromosome realigned bam directory.

Options

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

Advanced options for fine-tuning the variant calling are listed in *Advanced Options for Variant Detection* on page 151.

NOTE
The option --outDir is mandatory for all analysis types. CASAVA will not

run if this option is missing.

CASAVA will only run without --inSampleDir if the build has been already configured with --inSampleDir before.

Global Options

The options described below are global options used to specify analysis across different targets.

Table 14 Major File Options for Variant Detection and Counting

Option	Application	Description

`-id,`
`--inSampleDir=PATH`
	SE, PE	PATH to the aligned sample input directory.
		Example: `-id TestData/Aligned/Project_<SampleProject>/Sample_<SampleID>`

`-od,`
`--outDir=PATH`
	SE, PE	PATH to the build sample output directory.
		Example: `-od /home/user_name/data/Project_01`

`-ref,`
`--refSequences=PATH`
	SE, PE	PATH of the reference genome sequences. Default is buildDir/genomes/.
		Example: `-ref /data/Genome/CASAVA/hg18`
		The FASTA files should not be squashed for CASAVA.

`--samtoolsRefFile=FILE`	SE, PE	PATH to a single samtools-style reference file

Table 15 Behavioral Options for Variant Detection and Counting

Option	Application	Description

`-a,`
`--applicationType=TYPE`
	SE, PE	Type of analysis [DNA, RNA]; default is DNA.
		Example: `-a RNA`

`-f,`
`--force`
	SE, PE	Ignore errors from previous CASAVA execution.
		Example: `-f`

`-h,`
`--help [TARGET]`
	SE, PE	Prints on screen usage guide. If TARGET is specified, prints usage guide for the corresponding plugin target
		Example: `--help bam`

`-j, --jobsLimit`	SE, PE	Limit number of parallel jobs. Defaults: -1 (unlimited) for --sgeAuto. 1 for --workflowAuto. Do not set it to the maximum number of processors as this might cause the terminal to become unresponsive
`--postRunCmd=CMDLINE`	SE, PE	Post Run Commands can be launched after CASAVA completes by including the --postRunCmd option, followed by the commands to be launched
`-sa, --sgeAuto`	SE, PE	Generates the workflow definition file and runs it on SGE (use with --sgeQueue)
`--sgeQsubFlags`	SE, PE	Extra parameters to be passed to SGE qsub by the taskServer.pl
`--sgeQueue`	SE, PE	SGE queue name, used with --sgeAuto or --workflow (e.g: all.q)
`--targets=LIST`	SE, PE	Space-separated list of targets to run (see *Targets* on page 94). Default is `all`. Example: `--targets sort bam`
`--tempDir`	SE, PE	Overrides default path for local temporary files
`--verbose=NUMBER`	SE, PE	Sets the verbose level (default is 0, which is the minimum). Example: `--verbose=1`
`--version`	SE, PE	Prints version information. Example: `--version`

`-w,`
`--workflow`
	SE, PE	Instead of running CASAVA , generates the workflow definition file

Option	Application	Description
		tasks-DATA.txt Example: `-w`

`-wa,`
`--workflowAuto`
	SE, PE	Generates the workflow definition file and runs it. See --jobsLimit.
		Example: `--workflowAuto`

`--workflowFile=FILE`	SE, PE	Overrides workflow file name. Default is tasks.<date>.txt Example: `--workflowFile=`FILENAME.txt

Table 16 Global Analysis Options for Variant Detection and Counting

Option	Application	Description
`--QVCutoff=NUMBER`	PE	Sets the paired-end alignment score threshold to NUMBER (default 90). Example: `--QVCutoff=60`
`--QVCutoffSingle=NUMBER`	SE, PE	Sets the single-read alignment score threshold to NUMBER (default 10). Example: `--QVCutoffSingle=60`
`--read=NUMBER`	PE	Limit input to the specified read only. Forces single-ended analysis on one read of a double-ended dataset. Example: `--read=1`
`--singleScoreForPE=VALUE`	PE	Sets the variant caller to filter reads with single score below QVCutoffSingle in PE mode YES|NO. Default NO. Example: `--singleScoreForPE=YES`
`--sortKeepAllReads`	SE, PE	Generate an archive BAM file. Keep all purity filtered, duplicate and unmapped reads in the build. These reads will be ignored during variant calling. Example: `--sortKeepAllReads`
`--toNMScore=NUMBER`	SE, PE	Minimum SE alignment score to put a read to NM. Default=-1 (-1 means option is turned off)
`--ignoreUnanchored`	PE	Ignore unanchored read pairs in indel assembly and variant calling. Unanchored read pairs have a single-read alignment score of 0 for both reads. Example: `--ignoreUnanchored`

Options for Target sort

The options described below are used to specify analysis for target `sort`.

Table 17 Analysis Options for sort

Option	Application	Description
`--rmDup=YES|NO`	PE	Turn On/Off PCR duplicate marking/removal for paired-end reads (default YES).
`--sortBufferSize=INTEGER`	SE, PE	Buffer size used by the read sorting process, in megabytes (default: 1984).
`--sortKeepAllReads`	SE, PE	Run the sort module in archival mode instead of the default filtered mode. See *Archival Build* on page 88). Example: `--sortKeepAllReads`

Options for Target rnaCounts

The options described below are used to specify analysis for target `rnaCounts`.

Table 18 Analysis Options for rnaCounts

Option	Application	Description
`--refFlatFile`	SE	Name and location of UCSC refFlat.txt.gz file. The file must be gz-compressed. Example: `--refFlatFile=/data/Genome/ELAND_RNA/Human/refFlat.txt.gz`
`--seqGeneMdFile`	SE	Name and location of NCBI seq_gene.md.gz file. Example: `--seqGeneMdFile=/data/Genome/ELAND_RNA/Human/seq_gene.md.gz`
`--seqGeneMdGroupLabel`	SE	The group label specifies which assembly to use in the seq_gene file, and is found in column 13 of the file. seq_gene files can hold entries for multiple assemblies. **Required** for RNA counting when you use the annotation seqGeneMd file from NCBI. Example: `--seqGeneMdGroupLabel 'GRCh37.p2-Primary Assembly'`

Options for Target bam

The options described below are used to specify analysis for target `bam`.

Table 19 Analysis Options for bam

Option	Application	Description
`--bamChangeChromLabels=OFF/NOFA/UCSC`	SE, PE	Change chromosome labels in the bam plugin output. The available behaviors are: `OFF` Use unmodified CASAVA chromosome labels (default behavior). `NOFA` Remove any ".fa" suffix found on each chromosome label. For example "c11.fa" is changed to "c11". `UCSC` Remove any ".fa" suffix found on each chromosome label and attempt to map the result to the corresponding UCSC human chromosome label. For example "c11.fa" is changed to "chr11".
`--bamSkipRefSeq`	SE, PE	Do not generate a reference sequence file with each bam file. The default behavior can be restored with `--no-bamSkipRefSeq`.

Configuring Multiple Runs

To add multiple runs you can modify the run configuration file (run.conf.xml):

a Go to /Human/conf/run.conf.xml (see *Run.conf.xml* on page 91).

b Add the additional entries to the run.conf.xml file.

c Then run the configuration again by executing:

```
./configureBuild.pl -p /Human/
```

Targeted Resequencing

Since targeted resequencing only sequences part of a genome, we recommend using the option `--variantsNoCovCutoff` to turn off high-coverage filtration of SNPs and indels.

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 `configureBuild.pl` help page. Go to the CASAVA installation directory, and type:

```
./configureBuild.pl
```

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

Variant Detection and Counting Output Files

Once the post-alignment build is complete, all relevant information is listed in the build directory, such as:

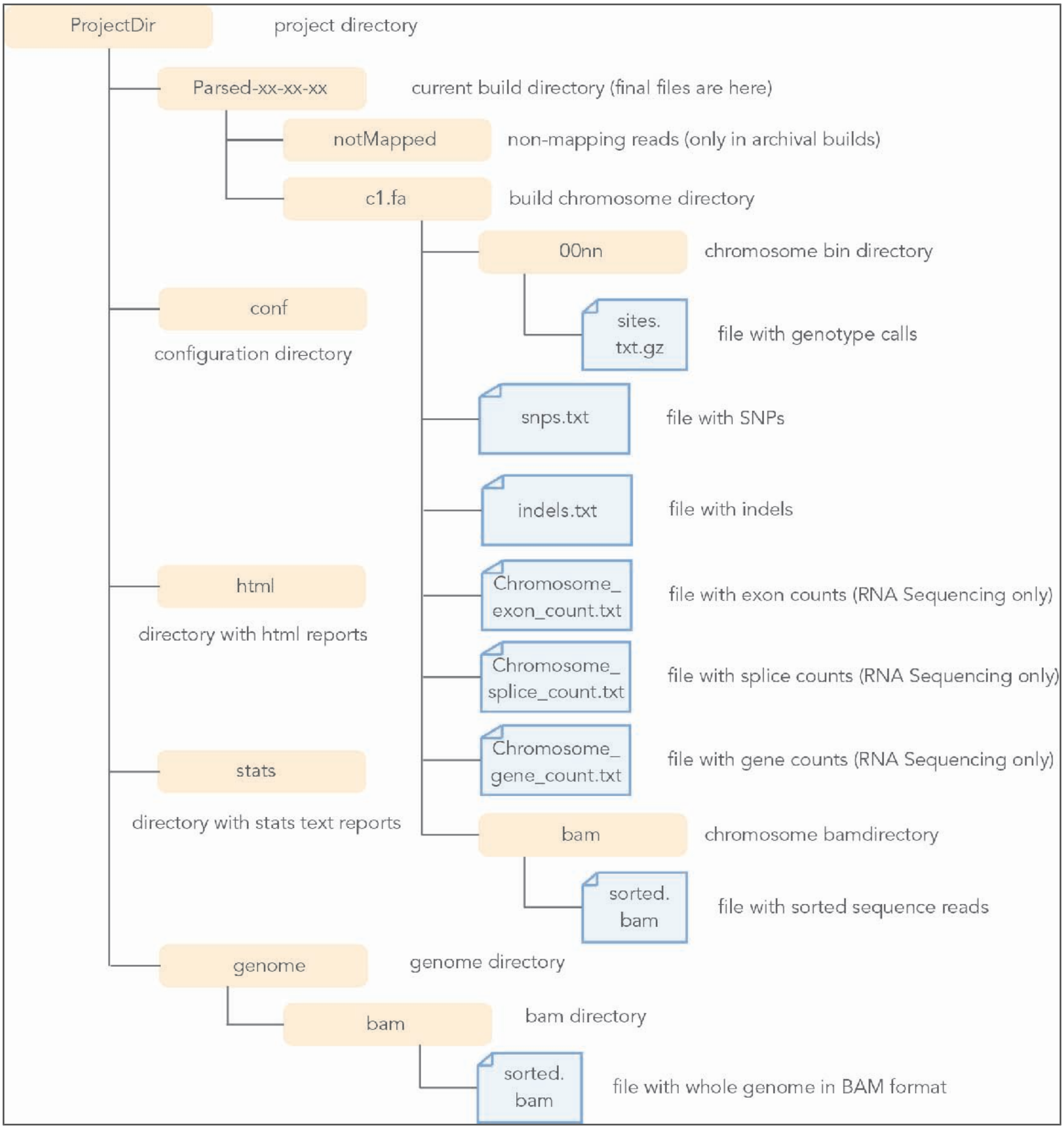
- Build summary html pages.
  The build summary html pages are located in the buildDir/html folder, and provides access to run information and graphs of important statistics.
- Variant calls and counts.
  The CASAVA build contains sequence, SNP, indels, and (for RNA Sequencing) counts information, and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

These files are described below.

Build Directory

An outline of the CASAVA build directory is shown below.

Figure 15 CASAVA Build Directory


ProjectDir
project directory
Parsed-xx-xx-xx
current build directory (final files are here)
notMapped
non-mapping reads (only in archival builds)
c1.fa
build chromosome directory
00nn
chromosome bin directory
conf
configuration directory
sites.txt.gz
file with genotype calls
snps.txt
file with SNPs
indels.txt
file with indels
html
directory with html reports
Chromosome_exon_count.txt
file with exon counts (RNA Sequencing only)
Chromosome_splice_count.txt
file with splice counts (RNA Sequencing only)
stats
directory with stats text reports
Chromosome_gene_count.txt
file with gene counts (RNA Sequencing only)
bam
chromosome bamdirectory
sorted.bam
file with sorted sequence reads
genome
genome directory
bam
bam directory
sorted.bam
file with whole genome in BAM format

The most important folders for downstream analysis are listed below.

- Html Folder

  The html folder contains the build summary html pages (see *Build Html Page* on page 102), 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 sorted alignments, SNP and indel calls, and (for RNA Sequencing) gene counts, exon counts, and splice junction counts (see *CASAVA Build* on page 103). This information is organized in chromosome folders named c1 or c2, 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.

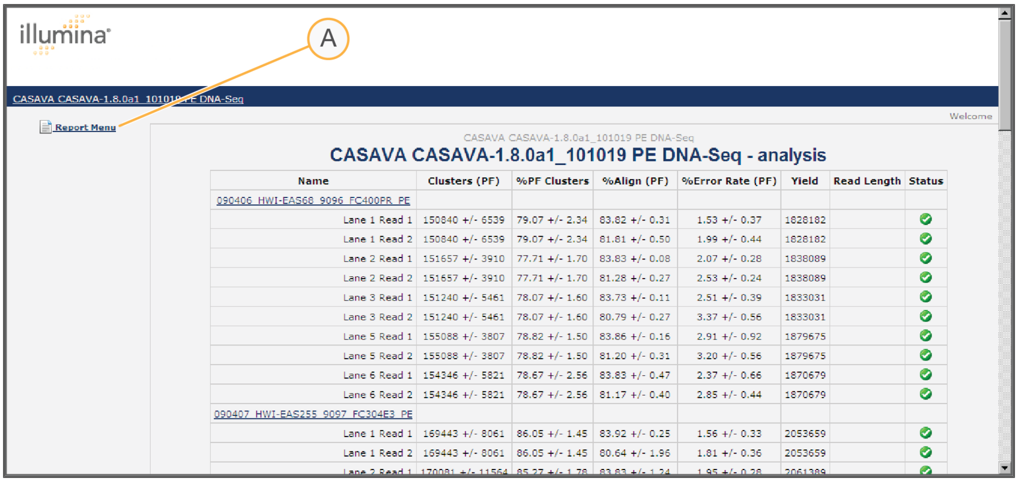
- Conf Folder

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

Build Html Page

The build html page is located in buildDir/html. When you open the file Home.html, you will find a list of all runs, and a link to statistics.

Figure 16 Build Html Page



CASAVA CASAVA-1.8.0a1_101019 PE DNA-Seq

CASAVA CASAVA-1.8.0a1_101019 PE DNA-Seq

CASAVA CASAVA-1.8.0a1_101019 PE DNA-Seq - analysis

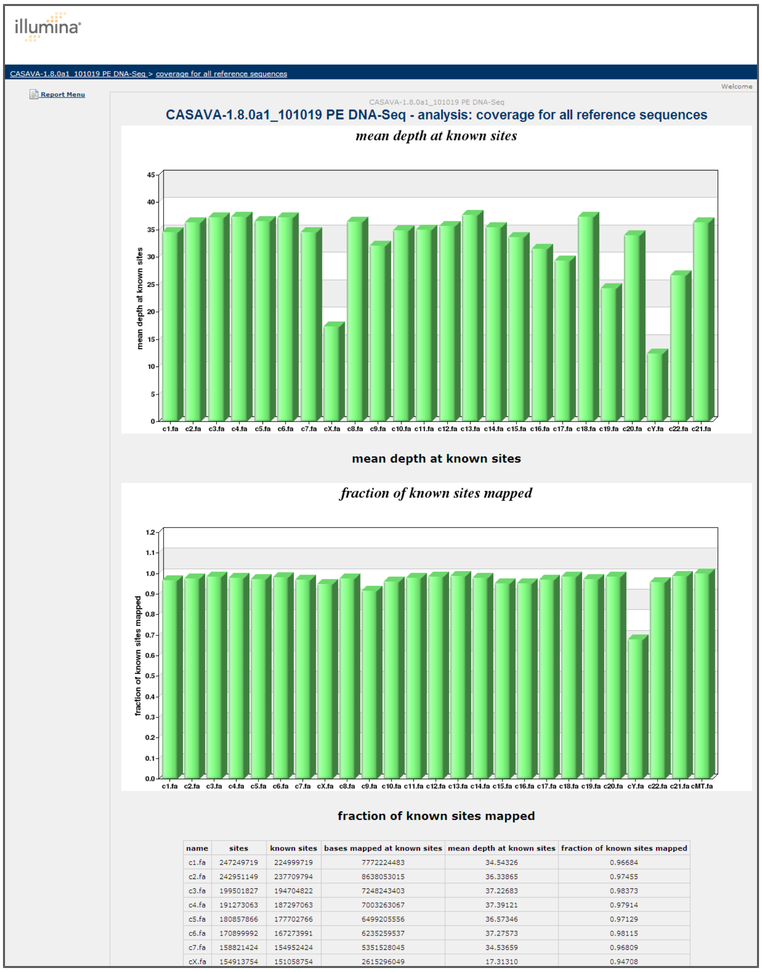
Name	Clusters (PF)	%PF Clusters	%Align (PF)	%Error Rate (PF)	Yield	Read Length	Status
090406_HWI-EAS68_9096_FC400PR_PE							
Lane 1 Read 1	150840 +/- 6539	79.07 +/- 2.34	83.82 +/- 0.31	1.53 +/- 0.37	1828182		✓
Lane 1 Read 2	150840 +/- 6539	79.07 +/- 2.34	81.81 +/- 0.50	1.99 +/- 0.44	1828182		✓
Lane 2 Read 1	151657 +/- 3910	77.71 +/- 1.70	83.83 +/- 0.08	2.07 +/- 0.28	1838089		✓
Lane 2 Read 2	151657 +/- 3910	77.71 +/- 1.70	81.28 +/- 0.27	2.53 +/- 0.24	1838089		✓
Lane 3 Read 1	151240 +/- 5461	78.07 +/- 1.60	83.73 +/- 0.11	2.51 +/- 0.39	1833031		✓
Lane 3 Read 2	151240 +/- 5461	78.07 +/- 1.60	80.79 +/- 0.27	3.37 +/- 0.56	1833031		✓
Lane 5 Read 1	155088 +/- 3807	78.82 +/- 1.50	83.86 +/- 0.16	2.91 +/- 0.92	1879675		✓
Lane 5 Read 2	155088 +/- 3807	78.82 +/- 1.50	81.20 +/- 0.31	3.20 +/- 0.56	1879675		✓
Lane 6 Read 1	154346 +/- 5821	78.67 +/- 2.56	83.83 +/- 0.47	2.37 +/- 0.66	1870679		✓
Lane 6 Read 2	154346 +/- 5821	78.67 +/- 2.56	81.17 +/- 0.40	2.85 +/- 0.44	1870679		✓
090407_HWI-EAS255_9097_FC304E3_PE							
Lane 1 Read 1	169443 +/- 8061	86.05 +/- 1.45	83.92 +/- 0.25	1.56 +/- 0.33	2053659		✓
Lane 1 Read 2	169443 +/- 8061	86.05 +/- 1.45	80.64 +/- 1.96	1.81 +/- 0.36	2053659		✓
Lane 2 Read 1	170081 +/- 11564	85.27 +/- 1.78	83.83 +/- 1.24	1.95 +/- 0.28	2061389		✓

A **Report Menu** link

The **Report Menu** link on the build html page will lead you to graphs and tables for important statistics:

- Coverage
- Duplicates
- Indels statistics
- SNPs statistics

Figure 17 Coverage Graph in Home.html


illumina
CASAVA-1.8.0a1_101019 PE DNA-Seq > coverage for all reference sequences
Report Menu
Welcome
CASAVA-1.8.0a1_101019 PE DNA-Seq
mean depth at known sites
45
40
35
30
25
20
15
10
5
0
mean depth at known sites
c1.fa c2.fa c3.fa c4.fa c5.fa c6.fa c7.fa cX.fa c8.fa c9.fa c10.fa c11.fa c12.fa c13.fa c14.fa c15.fa c16.fa c17.fa c18.fa c19.fa c20.fa cY.fa c22.fa c21.fa
fraction of known sites mapped
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
fraction of known sites mapped
c1.fa c2.fa c3.fa c4.fa c5.fa c6.fa c7.fa cX.fa c8.fa c9.fa c10.fa c11.fa c12.fa c13.fa c14.fa c15.fa c16.fa c17.fa c18.fa c19.fa c20.fa cY.fa c22.fa c21.fa cMT.fa

CASAVA-1.8.0a1_101019 PE DNA-Seq - analysis: coverage for all reference sequences

mean depth at known sites

fraction of known sites mapped

name	sites	known sites	bases mapped at known sites	mean depth at known sites	fraction of known sites mapped
c1.fa	247249719	224999719	7772224483	34.54326	0.96684
c2.fa	242951149	237709794	8638053015	36.33865	0.97455
c3.fa	199501827	194704822	7248243403	37.22683	0.98373
c4.fa	191273063	187297063	7003263067	37.39121	0.97914
c5.fa	180857866	177702766	6499205556	36.57346	0.97129
c6.fa	170899992	167273991	6235259537	37.27573	0.98115
c7.fa	158821424	154952424	5351528045	34.53659	0.96809
cX.fa	154913754	151058754	2615296049	17.31310	0.94708

CASAVA Build

The CASAVA build, containing sequence, SNP, indels, and (for RNA Sequencing) counts information, is located in the buildDir/Parsed_xx-xx-xx folder.

Sorted.bam Files

The sorted.bam file is a binary file that contains sorted sequence alignments. There is one sorted.bam file for each chromosome, stored in the 'bam' subdirectory under each chromosome specific directory.

BAM Format

The Binary Alignment/Map (BAM) file is the binary equivalent of SAM files and is compressed in the BGZF format. Each BAM file is much smaller than its SAM equivalent, yet it can be easily converted to SAM, e.g. with samtools using "samtools view -h file.bam".

When a BAM file is created for each chromosome, these files are placed in the bam directory immediately under the Parsed chromosome directory. For example the BAM file for chromosome 1 in a human build would be located here:

```
Project_Dir/Parsed_NN-NN-NN/c1.fa/bam/
```

When one BAM file is created for the entire genome (using the target `bam`) it can be found in:

```
Project_Dir/genome/bam/
```

A set of auxillary files is created with the whole genome BAM file to facilitate use in downstream packages such as SAMtools or the Broad IGV. These files are:

- sorted.bam—the bam file itself
- sorted.bam.bai—index of the bam file
- sorted.bam.fa.gz—gzipped fasta file containing the reference sequence(s)

For a description of the BAM format, see samtools.sourceforge.net.

The format of SAM files is described in *SAM Format* on page 169. BAM files are the binary equivalent of SAM files, and Illumina's BAM convention has the following features:

- The new private optional tag "XC" has been added to provide read status information normally conveyed in the chromosome field of the export.txt file for unmapped reads. Specifically, "XC:Z:QC" is used to mark an ELAND QC failure read, "XC:Z:RM" is used to mark an ELAND repeat mask read, and "XC:Z:CONTROL" is used to mark a control read. No optional field is added to reads which are marked as no match ("NM") in the export file – it is understood that this is the default status of an unmapped read..
- Reads which cross a splice junction are annotated as a single record using the SAM CIGAR "N" (SKIP) character. For example, a 75-base read spanning a 1000-base intron may have the cigar string: "35M1000N40M".
- Chromosome names cannot be changed in the chromosome BAM files on which CASAVA operates. The bam module may be used to create a whole genome BAM file with translated chromosome labels. Note that whole genome bam files are now the only option for the bam module.

Converting Sam and Bam Files

If you want to convert Sam files into Bam files, enter the following:

```
samtools view -b -h -S -o output.bam. <in.sam>
```

Where:

`-b` Output in the BAM format.
`-h` Include the header in the output.
`-S` Input is in SAM. If @SQ header lines are absent, the ‘-t’ option is required.
`-o FILE` Output file [stdout]

If you want to convert Bam files into Sam files, enter the following:

```
samtools view -h -o output.sam <in.bam>
```

Where:

`-h` Include the header in the output.
`-o FILE` Output file [stdout]

For more information, samtools.sourceforge.net.

Variant Detection Output Files

All variant caller output files are written in a text format composed of one header segment followed by one data segment.

All lines in the header segment begin with the '#' character. Header lines beginning with the sequence '#$' contain a key,value pair. The reserved key 'COLUMNS' has an associated value containing the set of column labels in the following data segment.

The data segment contains one entry per line, where each line is a set of tab-delimited columns. Wherever appropriate, columns for sequence name and position number are included such that the files are tabix compatible.

The following files are generated by CASAVA variant detection and counting:

- Depth and single position genotype call scores for every mapped site in the reference genome are saved in each bin directory in the gzipped file "sites.txt.gz":
  `Project_Dir/Parsed_NN-NN-NN/c1/0000/sites.txt.gz`
  Note that this output can be omitted with the `--variantsNoSitesFiles` option.
- The SNPs for each reference sequence are aggregated and filtered according to the `--variantsSnpCovCutoff` setting and summarized in the chromosome-level file snps.txt:
  `Project_Dir/Parsed_NN-NN-NN/c1/snps.txt`
- The indels for each reference sequence are aggregated and filtered according to the `--variantsIndelCovCutoff` setting and summarized in the chromosome-level file indels.txt:
  `Project_Dir/Parsed_NN-NN-NN/c1/indels.txt`
- If any SNPs and indels are removed by the high-depth filter, they can be found in their corresponding bin directory as:
  `Project_Dir/Parsed_NN-NN-NN/c1/0000/snps.removed.txt`
  `Project_Dir/Parsed_NN-NN-NN/c1/0000/indels.removed.txt`
- When the `--variantsWriteRealigned` option is selected, there will alse be a BAM file written to each reference sequence realigned bam directory containing only those reads aligned to an alternate alignment by the variant caller. The BAM filename is sorted.realigned.bam:

```
Project_Dir/Parsed_NN-NN-NN/
   c1/bam/realigned/sorted.realigned.bam
```

- Statistics for coverage, as well as snp and indel calls for all reference sequences are found in the 'stats' directory:

```
Project_Dir/stats/coverage.summary.txt
Project_Dir/stats/snps.summary.txt
Project_Dir/stats/indels.summary.txt
```

- A summary of the same information is also available on the following html pages:

```
Project_Dir/html/coverage.html
Project_Dir/html/snps.html
Project_Dir/html/indels.html
```

To summarize the snps and indels in the stats and html directories above, quality thresholds are used to select a subset of snps and indels for summary reporting. The default thresholds are Q(snp) >= 20 and Q(indel) >= 20. These values may be changed using the options `--variantsSummaryMinQsnp` and `--variantsSummaryMinQindel`.

snps.txt and sites.txt Files

The snps.txt files contain the SNP calls sorted by position, while the sites.txt files provide depth and single position genotype call scores for every mapped site. There is one snp.txt file for each chromosome, stored in the chromosome-specific directory under the Parsed-dd-mm-yy directory. The snps.txt and sites.txt files are tab delimited text files contain the same columns, which are the following:

No	Label	Description
1	seq_name	Reference sequence label
2	Pos	Sequence position of the site/snp
3	bcalls_used	Basecalls used to make the genotype call for this site
4	bcalls_filt	Basecalls mapped to the site but filtered out before genotype calling
5	Ref	Reference Base
6	Q(snp)	A Q-value expressing the probability of the homozygous reference genotype, subject to the expected rate of haplotype difference as expressed by the (Watterson) theta parameter (see *New Variant Calling Parameter: Theta* on page 149).
7	max_gt	The most likely genotype (subject to theta, as above).
8	Q(max_gt)	A Q-value expressing the probability that the genotype is not the most likely genotype above (subject to theta).
9	max_gt|poly_site	The most likely genotype assuming this site is polymorphic with an expected allele frequency of 0.5 (theta is still used to calculate the probability of a third allele -- i.e. the chance of observing two non-reference alleles).
10	Q(max_gt|poly_site)	A Q-value expressing the probability that the genotype is not the most likely genotype above assuming this site is polymorphic.
11	A_used	'A' basecalls used
12	C_used	'C' basecalls used
13	G_used	'G' basecalls used
14	T_used	'T' basecalls used

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 reference sequence by the small variant 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, which can be changed using the `--variantsIndelCovCutoff` option). This filter is designed to remove indel calls in regions close to centromeres and other high depth regions likely to generate spurious calls.

NOTE
This filter is off for RNA variant calling, and we recommend to turn it off for targeted resequencing.

The indels.txt file follows the general variant caller output file structure. The data segment of this file consists of 16 tab-delimited fields. The fields are described in the Table below (note that all information is given with respect to the forward strand of the reference sequence):

No	Label	Description
1	seq_name	Reference sequence label
2	pos	Except for right-side breakpoints, the reported start position of the indel is the first (left-most) reference position following the indel breakpoint. For right-side breakpoints the reported position is the right-most position preceding the breakpoint. Also note that wherever the same indel could

No	Label	Description
		be represented in a range of locations, the caller attempts to report it in the left-most position possible.
3	type	String summarizing the indel type. One of: • *n*I—Insertion of length n (e.g. 10I is a 10 base insertion) • *n*D—Deletion of length n (e.g. 10D is a 10 base deletion) • BP_LEFT—Left-side breakpoint • BP_RIGHT—Right-side breakpoint
4	ref_upstream	Segment of the reference sequence 5′ of the indel event. For right-side breakpoints this field is set to the value 'N/A'.
5	ref/indel	Equal length sequences corresponding to the reference and indel alleles which span the indel event. The character '-' indicates a gap sequence of the reference or the indel allele.
6	ref_ downstream	Segment of the reference sequence 3′ of the indel event. For left-side breakpoints this field is set to the value 'N/A'.
7	Q(indel)	Phred scaled quality score of the indel, which refers to probability that this indel does not exist at the given position. The Q-values given only reflect those error conditions which can be represented in the indel calling model, which is not comprehensive. See also *Quality Scores* on page 146. By default the variant caller reports all indels with Q(indel) > 0.
8	max_gtype	Most probable indel genotype. The indel genotype categories are as follows: `hom` refers to a homozygous indel. `het` refers to a heterozygous indel. `ref` refers to no indel at this position. Note that these do refer to true genotypes where indels overlap because the model is not capable of jointly calling overlapping indels. In the case of overlapping indels, max_gtype refers to the most likely copy-number of the indel. Note that indel calls where `ref` is the most-likely genotype will be reported. These correspond to indels with very low Q(indel) values.
9	Q(max_gtype)	Phred scaled quality score of the most probable indel genotype, which refers to the probability that the genotype of the indel is not that given as "max_gtype". The Q-values given only reflect those error conditions which can be represented in the indel calling model, which is not comprehensive. See also *Quality Scores* on page 146.
10	depth	Except for right-side breakpoints, this field reports the depth of the position preceding the left-most indel breakpoint. For right-side breakpoints this is the depth of the position following the breakpoint.
11	alt_reads	Number of reads strongly supporting either the reference path or an alternate indel path.
12	indel_reads	Number of reads strongly supporting the indel path.
13	other_reads	Number of reads intersecting the indel, but not strongly supporting either the reference or any one indel path.
14	repeat_unit	The smallest repeating sequence unit within the inserted or deleted sequence. For breakpoints this field is set to the value 'N/A'.
15	ref_repeat_ count	Number of times the repeat_unit sequence is contiguously repeated starting from the indel start position in the reference case.
16	indel_repeat_ count	Number of times the repeat_unit sequence is contiguously repeated starting from the indel start position in the indel case.

Note that 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 Sequencing, sorted by position, and there is one of each type per chromosome (for example, c19_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. “cM” indicates a mitochondrial DNA alignment.

2 Start—The start of the gene.

3 End—The end of the gene.

4 Genes—The gene symbol.

5 Normalized count (RPKM)=(10^9 x raw count)/(feature length x number of mapped bases)

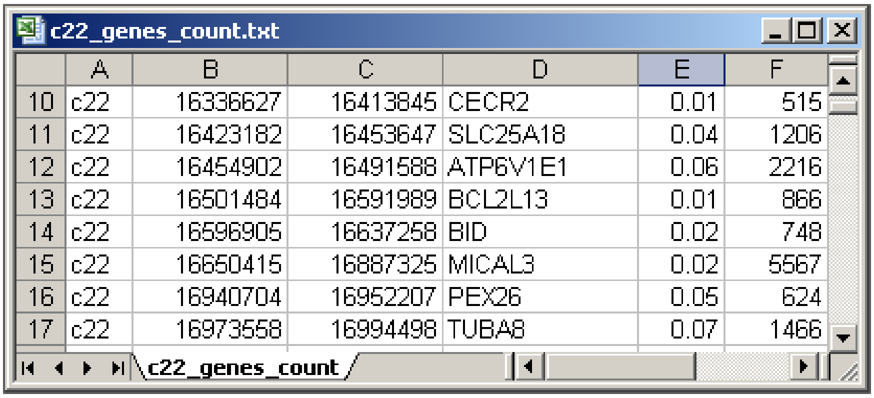
6 Raw count=sum of coverages for each base within the feature.

NOTE
For overlapping genes with different gene names only the non-overlapping portions for each gene participate in count generation.

Exon counts are sum of base coverages from genomic and spliced reads. Therefore gene counts are the 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 below.

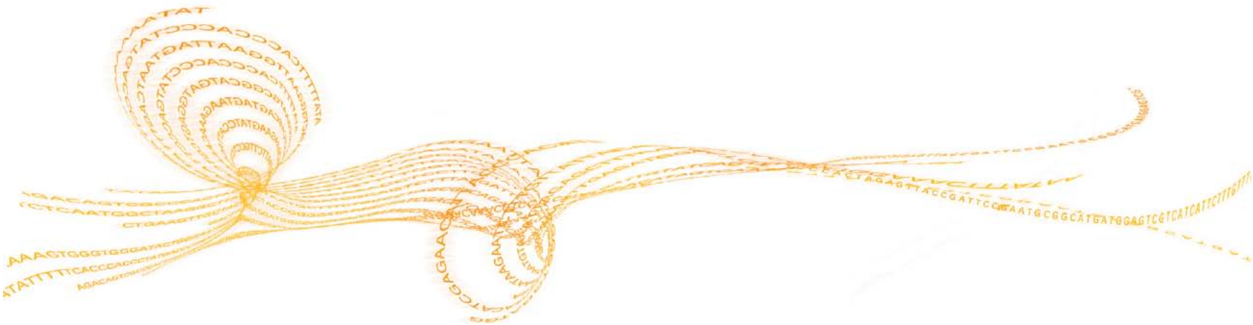
Figure 18 Chromosome_genes_count.txt File Opened in Excel



	A	B	C	D	E	F
10	c22	16336627	16413845	CECR2	0.01	515
11	c22	16423182	16453647	SLC25A18	0.04	1206
12	c22	16454902	16491588	ATP6V1E1	0.06	2216
13	c22	16501484	16591989	BCL2L13	0.01	866
14	c22	16596905	16637258	BID	0.02	748
15	c22	16650415	16887325	MICAL3	0.02	5567
16	c22	16940704	16952207	PEX26	0.05	624
17	c22	16973558	16994498	TUBA8	0.07	1466

Appendix A

Requirements and Software Installation



Hardware and Software Requirements

Network Infrastructure

The large data volumes generated and moved when running CASAVA mean that you will need the following:

1. A high-throughput ethernet connection (1 Gigabit recommended) or other data transfer mechanism.
2. A suitably large holding area for the analysis output (1 TB per run). As there will almost certainly be some overlap between copying, analysis, possible reanalysis, 2–3 TB is an absolute minimum.
3. You need to consider which parts of the data you want to store long-term and what storage infrastructure you want to provide. CASAVA provides the option to perform loss-less data compression.

Storage Configurations

You can configure your analysis server with either local storage or external network storage.

- Local server storage can be internal to the server, or Direct Attached Storage (DAS), which is a separate chassis attached to the server.
  - **Internal**—Simple but not scalable. Results data must be moved off to network storage at some point to make room for subsequent runs.
  - **DAS**—External chassis that is scalable since more than one DAS can be connected to the server. The server is an application server running CASAVA and a file server providing access to results and receiving incoming raw data files.
- External network storage is either Network Attached Storage (NAS) or Storage Area Network (SAN). NAS and SAN are functionally equivalent, but SAN is larger, with higher performance, more connections, and more management options.
  - **NAS**—External chassis connected via an Ethernet to the server, instrument PC, and other clients on the network. NAS devices are scalable and highly optimized.
  - **SAN**—The most scalable with the highest performance. They have a very high bandwidth and support many simultaneous clients, but are complex to manage and significantly more expensive.

Server Configurations

You can use either a single multi-processor, multi-core computer running Linux, or a cluster of Linux servers with a head node. CASAVA can take advantage of clustered and multi-processing servers.

- **Single multi-processor, multi-core server**—Simple but not scalable. It can only analyze data from one sequencing platform, or two depending on power and your turn-around requirements.
- **Linux Cluster**—Highly scalable and capable of running multiple jobs simultaneously. It requires one server as a management node and a minimum number of computational notes to be as efficient as a standalone server. By adding computational nodes, the cluster can service more instruments.

NOTE
We test our software with SGE; other cluster configurations (like LSF or PBS) are not recommended.

Analysis Computer

Illumina supports running CASAVA only on Linux operating systems. It may be possible to run CASAVA on other 64-bit Unix variants, if all of the prerequisites described in this section are met.

Illumina recommends the IlluminaCompute data processing solution for CASAVA. IlluminaCompute is available as a multi-tier option, with the volume of instrument data output per week determining the recommended Tier level. For more information, contact Illumina Support.

For example, for a laboratory generating 200 GB of sequence per week, the Tier 1 IlluminaCompute solution is recommended, for which the specifications are listed below (non-IlluminaCompute systems satisfying these requirements are also fully supported):

- 1 APC Netshelter: 40U Rack with 1U KMM console
- 3 Dell R610 Server: 8 CPU cores, 48 GB RAM
- 3 Isilon IQ12000x storage modules
- 1 Serial MGT Console 16
- 2 Cisco 3750e switches

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* on page 117.

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.

NOTE
For large projects (such as human genome resequencing) we recommend using highly distributed disk storage (like Lustre or Isilon).

The space requirements for ELAND temporary files inside the Aligned directory, as long as you stay at <13M reads per eland set size, are as follows:

- Eight bytes per match
- This should equate to less than 0.6 GB per process
- This is less than 5GB for 8 ELAND processes

If /tmp space is an issue, perform the following:

- Increase space for /tmp.
- Decrease ELAND_FASTQ_FILES_PER_PROCESS (see *ELAND_FASTQ_FILES_PER_PROCESS* on page 54). Setting the right value for the ELAND_FASTQ_FILES_PER_PROCESS is very important, because too low may result in a decreased performance.

Memory Requirements

CASAVA requires a minimum of 2 GB RAM per core. The parameter `ELAND_FASTQ_FILES_PER_PROCESS` in the configureAlignment config.txt specifies the maximum number of files aligned by each ELAND process. The optimal value is such that there are approximately 10 to 13 million lines (reads) in one set.

NOTE
Peak memory usage occurs during the ELANDv2e portion of configureAlignment.

Software Requirements

CASAVA has been primarily developed and tested on CentOs 5, Illumina's recommended and supported platform. It may be possible to install and run CASAVA on other 64-bit Linux distributions (particularly on similar distributions such as RedHat and Fedora) or on other Unix variants, if all of the prerequisites described in this section are met.

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 up to and including version gcc 4.5.2, with the exception of gcc version 4.0.2, which is not supported.
- Installation of CASAVA 1.8 now requires the Boost C++ library, version 1.44.0 and cmake version 2.8.0 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.

The following software is required to run the CASAVA 1.8; check whether it has been installed:

- GNU make (3.81 recommended)
- Perl (>= 5.8)
- Python (>=2.3 and <=2.6)
- PyXML
- gnuplot (>= 3.7, 4.0 recommended)
- ImageMagick (>= 5.4.7)
- ghostscript
- libxslt
- libxslt-devel
- libxml2
- libxml2-devel
- libxml2-python
- ncurses
- ncurses-devel
- gcc (4.0.0 or newer up to and including version gcc 4.4.x, except 4.0.2), with c++
- libtiff

- libtiff-devel
- bzip2
- bzip2-devel
- zlib
- zlib-devel
- Perl modules
    - perl-XML-Dumper
    - perl-XML-Grove
    - perl-XML-LibXML
    - perl-XML-LibXML-Common
    - perl-XML-NamespaceSupport
    - perl-XML-Parser
    - perl-XML-SAX
    - perl-XML-Simple
    - perl-XML-Twig
    - perldoc

InstallingCASAVA

Starting with CASAVA 1.8, CASAVA must be built outside of the source directory.

NOTE
For more information on the installation procedure, see the file CASAVA-1.8.0/install/CASAVA-1.8.0/src/INSTALL .

The installation procedure is as follows:

1 The Boost library 1.44.0 is bundled in the CASAVA distribution and will be automatically built when necessary. If you want to use a preinstalled Boost library, declare the `BOOST_ROOT` bash variable by typing the following at the command prompt prior to running the CASAVA./configure script:

```
export BOOST_ROOT=/path_to_compiled_boost_directory/boost_1_
   44_0
```

2 Download CASAVA v1.8 and copy it in a temporary directory (you will not need to keep it once the installation is done), like /tmp for example.

3 download and untar CASAVA v1.8 using the following commands:

```
cd /tmp
tar xvjf CASAVA_1.8.0.tar.bz2
```

4 Prepare to build CASAVA:

```
mkdir CASAVA-1.8.0-build
cd CASAVA-1.8.0-build
```

5 Configure CASAVA so it will be first built and then install where you want (in this example we want to install it in /illumina/software/CASAVA 1.8.0):

```
../CASAVA-1.8.0/src/configure --
   prefix=/illumina/software/CASAVA-1.8.0
```

6 Build CASAVA:

```
make
```

7 `Finally install it:`

```
make install
```

NOTE
For more information on the configuration options:

```
../CASAVA-1.8.0/src/configure --help
```

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 configureAlignment configuration file. For additional information, see *configureAlignment Configuration File* on page 55.

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 Aligned folder. The software assumes it can create a valid URL from the Aligned 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/Aligned and WEB_DIR_ROOT is http://server/SHARE, the software will write the links as http://server/SHARE/folder/Aligned/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:25
```

where yourserver is the name or IP address of a mail server that will accept SMTP email requests from you and 25 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 configureAlignment:

```
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
```

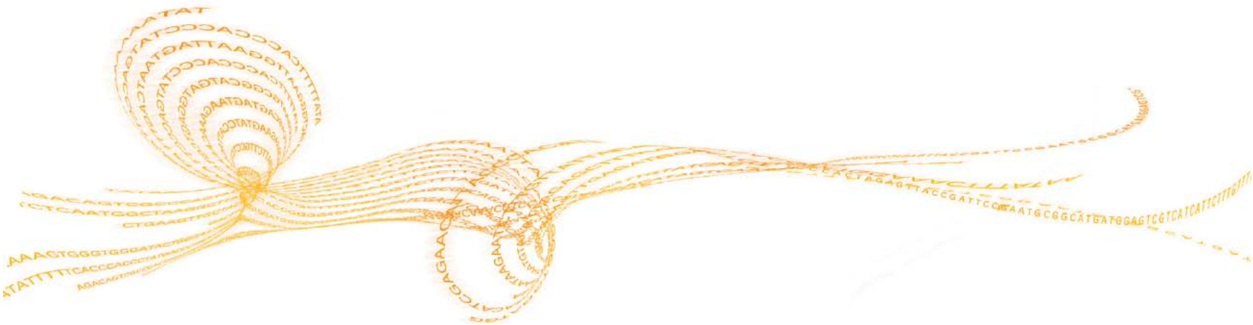
You should receive a test email. If you do not, the transcript it generates should identify the problem.

NOTE

The optional email reporting feature depends on how your SMTP servers are set up locally. Email reporting is not required to run the configureAlignment to a successful completion.

Appendix B

Using Parallelization



"Make" Utilities

Parallelization is built around the ability of the standard "make" utility to execute in parallel across multiple processes on the same computer. configureAlignment also provides a series of checkpoints and hooks that enables you to customize the parallelization for your computing setup. See *Customizing Parallelization* on page 119 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 configureAlignment run in parallel over four different processor cores, with an almost 4-fold decrease in analysis run time. On a system with more sockets or more cores per socket, "-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 (SGE).

To use "qmake," a short wrapper script is required. See below for details.

There are known issues with the use of "lsmake" that prevent parts of CASAVA from running. Therefore, Illumina does not recommend using "lsmake" to run CASAVA.

NOTE

Distributed cluster computing may require significant system administration expertise.

Illumina does not support external installations.

Using qmake

SGE has the utility qmake, which can run the tasks of a make across a cluster in parallel. There are two possible ways to run this.

Separate Jobs on Queuing System

The first generates each make tasks as a separate job run on the queuing system:

6 Move into the output folder .

7 Create a script file which contains the following:
```
qmake -cwd -v PATH -- -j 32
```

8 Submit the jobs to the SGE:
```
qsub -cwd -v PATH <script file>
```

The options convey the following information:

- -cwd tells the job to run in the current directory
- -v PATH passes the job the path to the executables needed for CASAVA.
- -- tells the job to pass everything after the -- to the make command
- -j tells qmake how many tasks to run at the same time. Qmake will then submit this number of tasks to the SGE queue. As tasks finish more tasks will be submitted.

The number after the `-j` should be adjusted depending on the size of the system and the number of users sharing it.

This method uses resources efficiently, but job monitoring and management is harder. If you need to kill a job you have to kill each of these tasks individually.

Slots Dedicated Upfront

The second method uses a parallel environment where a number of slots are dedicated upfront and the tasks are run on these slots.

1. Move into the output folder.
2. Create a script file which contains the following::
   ```
   qmake -cwd -v PATH -inherit -- all
   ```
3. Submit the jobs to the SGE:
   ```
   qsub -cwd -v PATH -pe make 32 <script file>
   ```

In addition to the options described above, this method uses the following options:

- `-pe make` says to run in the parallel environment, `make` is the default one
- The number after the word `make` says how many slots the job needs to run. If you set this number too high you may have to wait a long time for them all to become free. It will never run if you set it to more slots than you have on your system. The more slots you use the quicker your job will run (up to the parallelization limit) but the correct number to use depends on how big the system is, the number of other users, and the number of jobs you want to run at any one time.

This method can have some inefficiency if there are fewer tasks than slots at any point, but it allows easy job monitoring and management. If you need to kill your job then this is much easier with this method.

When you submit the job the command will return the SGE job id. You can get information about the state of your job with `qstat -j <job id>` or viewing it with `qmon`.

Customizing Parallelization

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

The configureAlignment can be divided into a series of steps with different levels of scalability where synchronization "barriers" cause configureAlignment 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, configureAlignment 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.

Parallelization Limitations

The analysis works on a per-file basis, so the maximum degree of parallelization achievable is equal to the total number of files generated during demultiplexing.

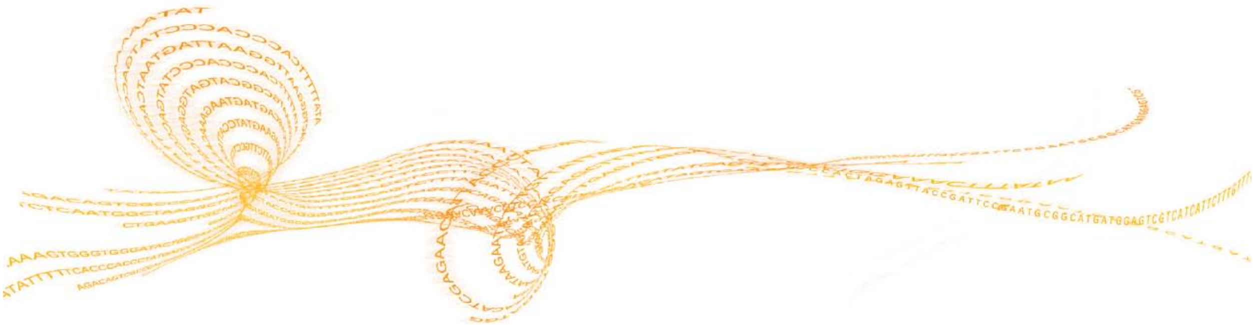
However, some parts of configureAlignment operate on a per-lane basis, and a few parts on a per-run basis, which means that scaling will cease to be linear at some stage for more than 8-way parallelization. The ELAND_FASTQ_FILES_PER_PROCESS affects the maximum level of parallelism available for ELAND. If all sequence information is stored in a total of 64 files, a value of 32 will lead to 2 processes, 8 to 8 processes, 4 to 16 processes, etc. These numbers of processes are doubled for paired end runs.

Memory Limitations

CASAVA requires a minimum of 2 GB RAM per core. The parameter `ELAND_FASTQ_FILES_PER_PROCESS` in the configureAlignment config.txt specifies the maximum number of tiles aligned by each ELAND process. The optimal value is such that there are approximately 10 to 13 million lines (reads) in one set. For additional information, see *Sequence Alignment* on page 45.

Reference Files CASAVA

Appendix C



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 Elandv2e and CASAVA variant detection, and provides instructions how to generate these files for other species and builds. As of CASAVA 1.8, ELAND squashes genome files automatically when it starts.

Genome sequence files for most commonly used model organisms are available through iGenome (*Getting Reference Files* on page 126).

ELAND Reference Files

ELAND needs the following file to perform an alignment:

- Unsquashed genome sequence files. As of CASAVA 1.8, ELAND squashes genome files automatically when it starts.

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
- RefFlat.txt.gz file (UCSC type) or seq_gene.md.gz file (NCBI type).

Reference Genome

CASAVA uses a reference genome in FASTA format. Both single sequence FASTA and multi-sequence FASTA genome files are supported.

Genome sequence files for most commonly used model organisms are available through iGenome (*Getting Reference Files* on page 126).

NOTE
As of CASAVA 1.8, you do not need to squash the reference genome anymore.

Single Sequence FASTA Files

CASAVA accepts single-sequence FASTA files as genome reference, which should be provided unsquashed for both alignment and post-alignment steps. The chromosome name is derived from the file name.

Direct CASAVA to a folder containing the FASTA files using the option `ELAND_GENOME` for configureAlignment.

Multi-Sequence FASTA Files

As of version 1.8, CASAVA accepts a multi-sequence FASTA file as genome reference. This should be provided as a single genome, SAM compliant, unsquashed file, for both alignment and post-alignment steps. The chromosome name is derived directly from the first word in the header for each sequence.

Direct CASAVA to a multi-sequence FASTA file using the option `SAMTOOLS_GENOME` for configureAlignment.

WARNING
GenomeStudio does not support the use of multi-sequence FASTA files. Therefore, if you want to analyze your output in GenomeStudio, we recommend using single sequence FASTA reference files.

Chromosome Naming Restrictions

CASAVA does not accept the following characters in the FASTA chromosome name header:

```
- ? ( ) [ ] / \ = + < > : ; " ' , * ^ | &
```

This validation can be disabled in configureAlignment using the following option:

```
CHROM_NAME_VALIDATION off
```

WARNING
You may run into problems with downstream analysis if you disable chromosome name validation.

NOTE
If ELAND finds two alignments with identical alignment scores, ELAND will pick the first alignment (in the single-end case) or combination of alignments (in the paired-end case) that exhibit the highest observed alignment quality. These are the alignments that make it into the export files (which only contain the best alignment for each read). In practice, post-alignment CASAVA ignores these reads because of the low alignment qualities.Using a reference with lexicographic chromosome names (like chr1) will yield slightly different results compared to a reference with numerical chromosome names (like 1) for these reads, since the hits are sorted in a different way.

Reference Sequence Blocks

For reasons of efficiency, ELAND treats 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.)

Abundant Sequences Files (eland_rna)

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 squashes the provided FASTA files at the start automatically, similar to the genome sequence files.

refFlat.txt.gz or seq_gene.md.gz File (eland_rna)

As of CASAVA 1.7,eland_rna uses the refFlat.txt.gz or seq_gene.md.gz file to generate the splice junction set automatically. The refFlat.txt.gz file comes from UCSC, while the seq_gene.md.gz file comes from NCBI, and are available through iGenomes. They should be provided gzip compressed, and should be from the same build as the reference files you are using for alignment. This negates the need to provide separate splice junction sets as in previous versions of CASAVA.

Variant Detection and Counting Reference Files

CASAVA variant detection and counting needs two types of files to analyze RNA Sequencing data:

- Genome sequence files
- refFlat.txt.gz or seq_gene.md.gz File (RNA Seq)

NOTE
CASAVA for DNA sequencing only needs the genome sequence files.

Reference Genome

CASAVA uses a reference genome in FASTA format. Both single sequence FASTA and multi-sequence FASTA genome files are supported.

Genome sequence files for most commonly used model organisms are available through iGenome (*Getting Reference Files* on page 126).

Single Sequence FASTA Files

CASAVA accepts single-sequence FASTA files as genome reference, which should be provided unsquashed for both alignment and post-alignment steps. The chromosome name is derived from the file name.

Direct CASAVA to a folder containing the FASTA files using the option `--refSequences=PATH` for variant detection and counting.

Multi-Sequence FASTA Files

As of version 1.8, CASAVA accepts a multi-sequence FASTA file as genome reference. This should be provided as a single genome, SAM compliant, unsquashed file, for both alignment and post-alignment steps. The chromosome name is derived directly from the first word in the header for each sequence.

Direct CASAVA to multi-sequence FASTA file using the option `--samtoolsRefFile=FILE` for variant detection and counting.

WARNING
GenomeStudio does not support the use of multi-sequence FASTA files. Therefore, if you want to analyze your output in GenomeStudio, we recommend using single sequence FASTA reference files.

Chromosome Naming Restrictions

CASAVA does not accept the following characters in the chromosome name:

```
- ? ( ) [ ] / \ = + < > : ; " ' , * ^ | &
```

refFlat.txt.gz or seq_gene.md.gz File

CASAVA 1.8 generates the non-overlapping exon coordinates set automatically using the refFlat.txt.gz file (from UCSC) or seq_gene.md.gz file (from NCBI). They should be from the same build as the reference files you are using for alignment, and are available from iGenome for the most common model organisms (*Getting Reference Files* on page 126).

Getting Reference Files

To run CASAVA, you will need to download genome and other reference files. You can use iGenome for most commonly used model organisms. This is explained in this section.

Illumina Provided Genomes

Illumina provides a number of commonly used genomes at ftp.illumina.com along with a reference annotation:

- Arabidopsis_thaliana
- Bos_taurus
- Caenorhabditis_elegans
- Canis_familiaris
- Drosophila_melanogaster
- Equus_caballus
- Escherichia_coli_K_12_DH10B
- Escherichia_coli_K_12_MG1655
- Gallus_gallus
- Homo_sapiens
- Mus_musculus
- Mycobacterium_tuberculosis_H37RV
- Pan_troglodytes
- PhiX
- Rattus_norvegicus
- Saccharomyces_cerevisiae
- Sus_scrofa

You can login using the following credentials:

- Username: igenome
- Password: G3nom3s4u

For example, download the FASTA, annotation, and bowtie index files for the human hg18 genome with the following commands:

```
>wget --ftp-user=igenome --ftp-password=G3nom3s4u
   ftp://ftp.illumina.com/Homo_sapiens/UCSC/hg18/Homo_sapiens_
   UCSC_hg18.tar.gz
```

Unpack the tar file:

```
tar xvzf Homo_sapiens_UCSC_hg18.tar.gz
```

Unpacking will make its own folder

```
Homo_sapiens/UCSC/hg18
```

Abundant Sequence Files (RNA Seq)

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.

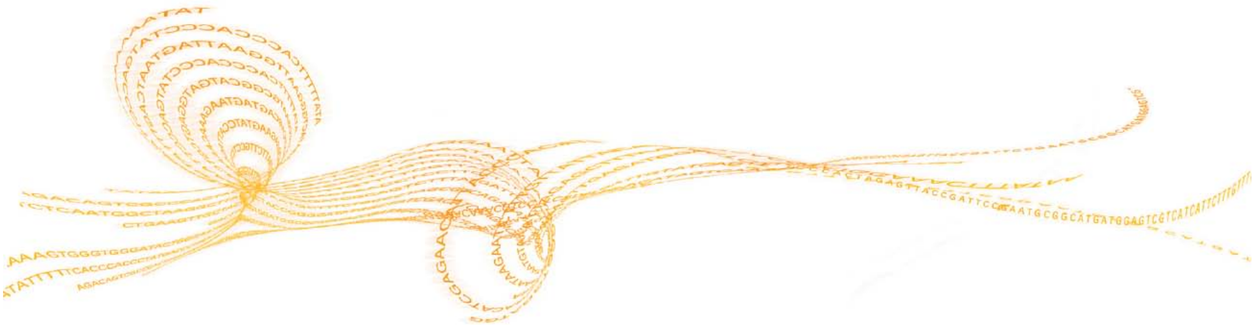
- A ribosomal sequences FASTA file. You will need to find it for your genome of interest, for example, from GenBank.
- A 5SRNA FASTA file (optional). You will need to find it for your genome of interest, for example, from GenBank.
- A contaminants file. You can use the same newcontam.fa file as for human, 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.

NOTE
Abundant sequence files need to be single-FASTA files, no multi-FASTA allowed.

Appendix D

Algorithm Descriptions



Introduction

This appendix explains the algorithms used in CASAVA for the following functions:

- Alignment using ELAND
- Indel detection and small variant genotyping
- RNA sequencing counting methods

ELANDv2 and ELANDv2e

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

As of CASAVA 1.6 a new version of ELAND is available, ELANDv2. The most important improvement of ELANDv2 is its ability to perform multiseed and gapped alignments. As a consequence, ELANDv2 handles indels and mismatches better.

CASAVA 1.8 also contains a new version of ELAND (ELANDv2e), with an orphan aligner, repeat resolution, and performance enhancements.

Input and Output Files

For a detailed description of the input and output files for ELANDv2e, see *configureAlignment Input Files* on page 48 and *configureAlignment Output Files* on page 71.

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 10 bases.

A 'match descriptor' string in the output file (see *Output File Formats* on page 1) encodes which bases in the read matched the genome and which were mismatches, and reports the gaps using the escape sequence "^..$" (see *Export.txt.gz* on page 77).

The differences between gapped and ungapped alignments, and singleseed and multiseed alignments are illustrated below.

Figure 19 Ungapped Versus Gapped Alignment

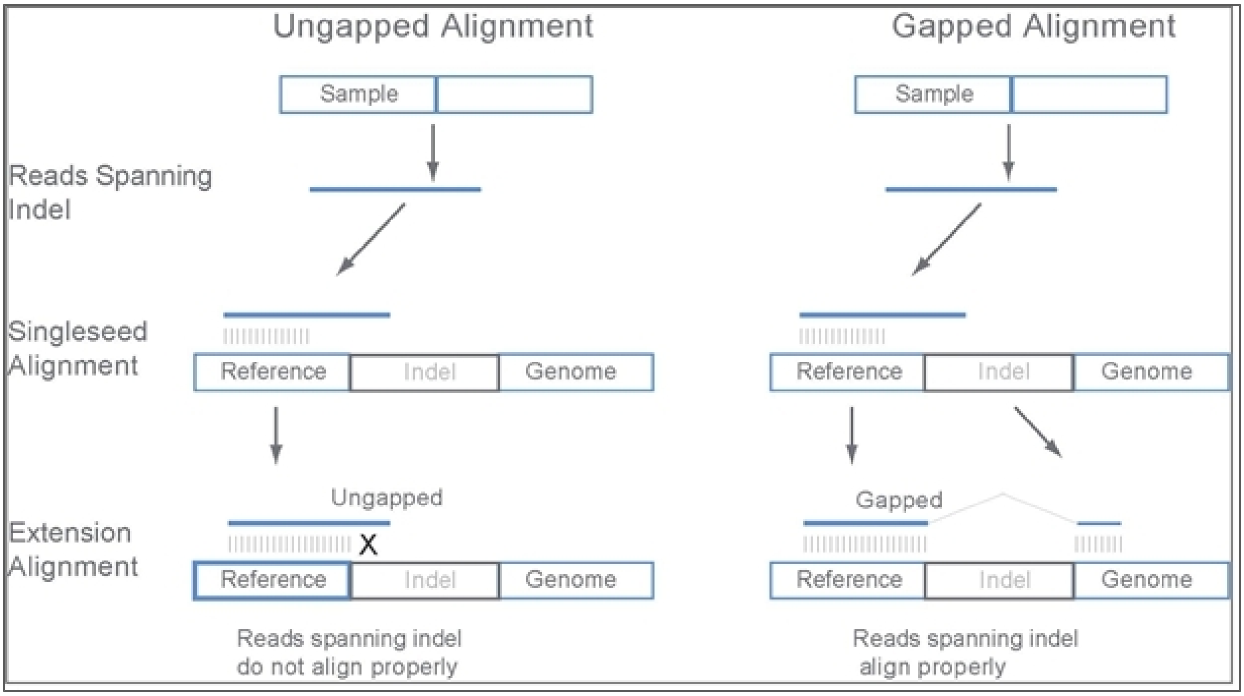
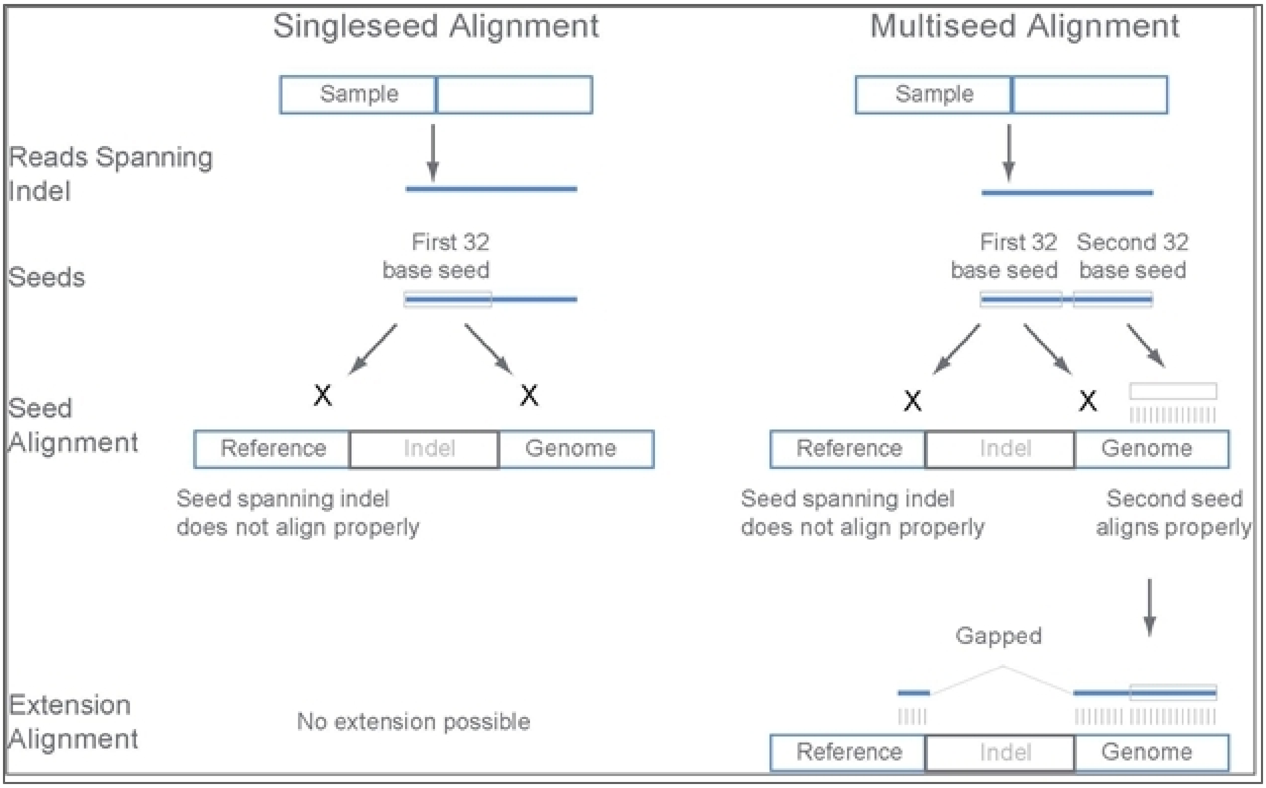


Figure 20 Singleseed Versus Multiseed Alignment



Note that a read has to have at least one seed that matches with at most 2 mismatches, and for that seed no gaps are allowed. For the whole read we allow any number of gaps, as long as they correct at least five mismatches downstream.

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
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.

For example, if there are two candidates for a read with p-values 0.9 and 0.3, the alignment score calculation would be as follows:

0.9/(0.9+0.3) = 0.75, chance of highest scoring alignment being right

1- 0.75 = 0.25, chance of highest scoring alignment being wrong

Expressed on the Phred scale:

Alignment score = -10 log (0.25) = 6.

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. ELANDv2e uses a 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.

configureAlignment 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.gz:

- 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, ELANDv2e 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, i.e. we restrict ourselves to a maximal length of an expected insertion/deletion (this value is set to 10).

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 (#(mismatches ungapped_alignment)-#(mismatches gapped_alignment)/#(mismatches gapped_alignment). If the ratio for a given alignment exceeds a certain value (set to 3.1 by default), we insert a gap.

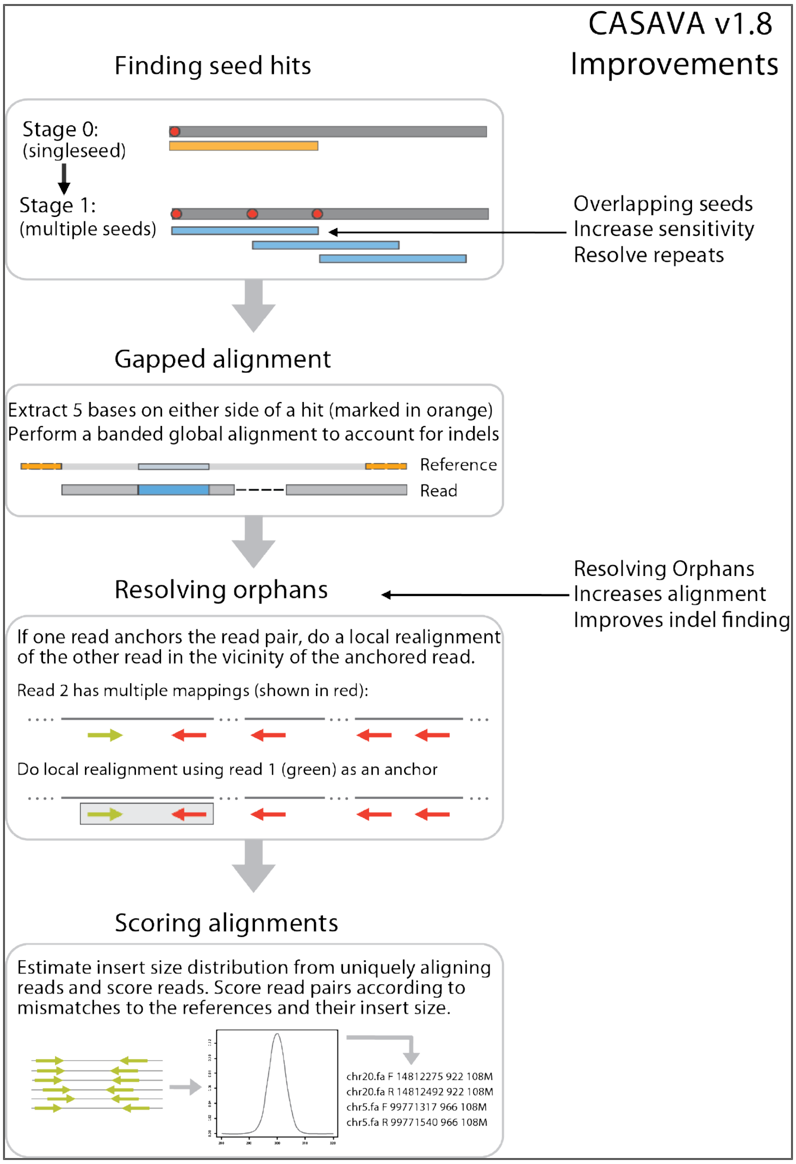
If any of two conditions is not satisfied, we return an ungapped alignment as the result.

ELANDv2e Alignment Improvements

CASAVA 1.8 launches ELANDv2e, and contains the following new features for alignment:

- ELANDv2e has better repeat resolution
- ELANDv2e introduces the orphan aligner
- alignmentResolver is rewritten, resulting in shorter run times

Figure 21 ELANDv2e Workflow



Extract 5 bases on either side of a hit (marked in orange)
Perform a banded global alignment to account for indels

If one read anchors the read pair, do a local realignment of the other read in the vicinity of the anchored read.

Read 2 has multiple mappings (shown in red):

Do local realignment using read 1 (green) as an anchor

Estimate insert size distribution from uniquely aligning reads and score reads. Score read pairs according to mismatches to the references and their insert size.

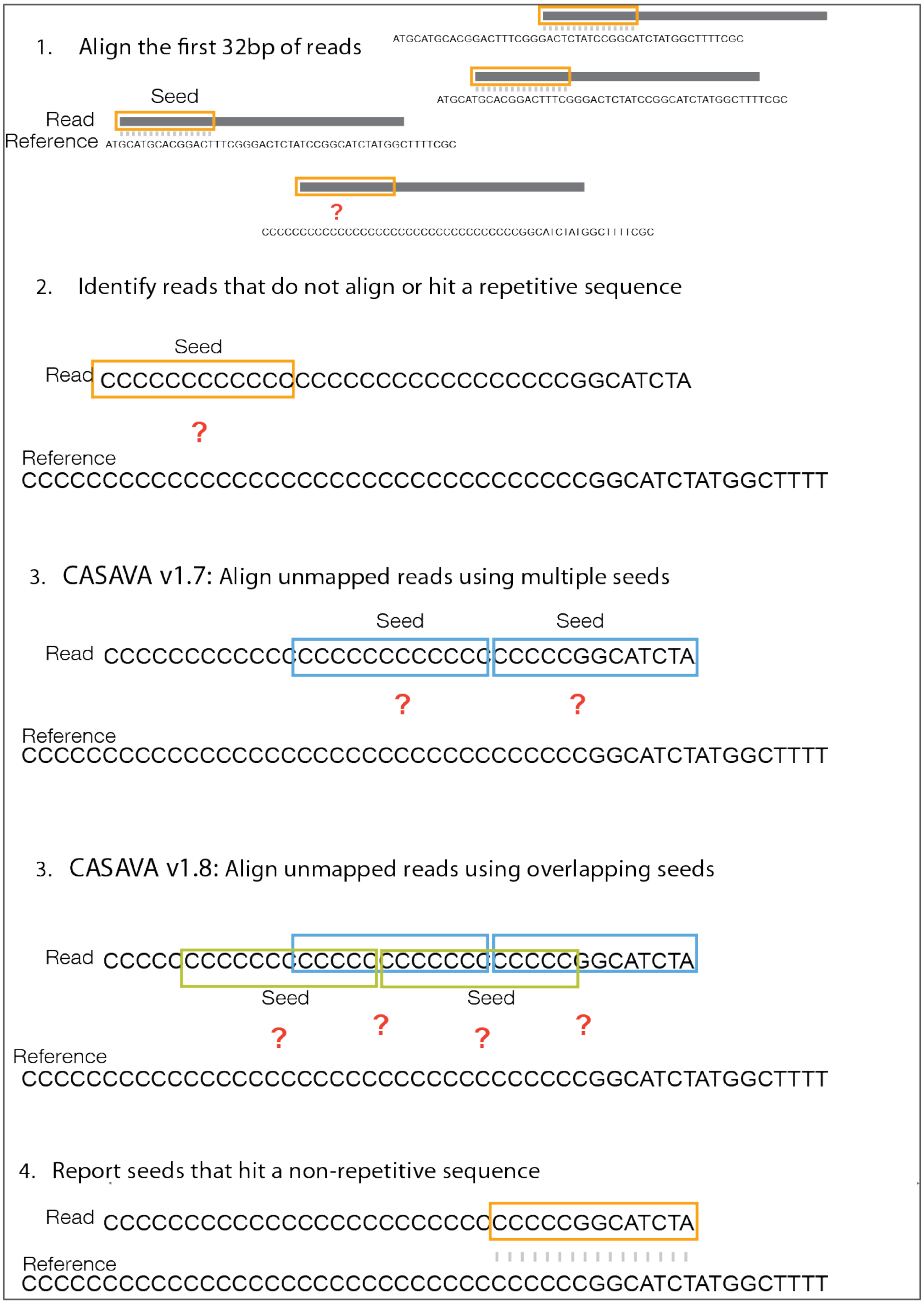
```
chr20.fa F 14812275 922 108M
chr20.fa R 14812492 922 108M
chr5.fa F 99771317 966 108M
chr5.fa R 99771540 966 108M
```

Repeat Resolution

ELANDv2e aligns reads in repeat regions using two new modes: semi-repeat resolution and full repeat resolution. Both modes take also repetitive hits into account for the multiseed pass of ELAND. Full repeat resolution is more sensitive and places more reads in repeat regions, but will result in longer run time.

By default, ELANDv2e runs in semi-repeat resolution mode. Full repeat resolution can be turned on with the option INCREASED_SENSITIVITY.

Figure 22 Changes between CASAVA 1.7 and 1.8 in multiseed ELAND alignment

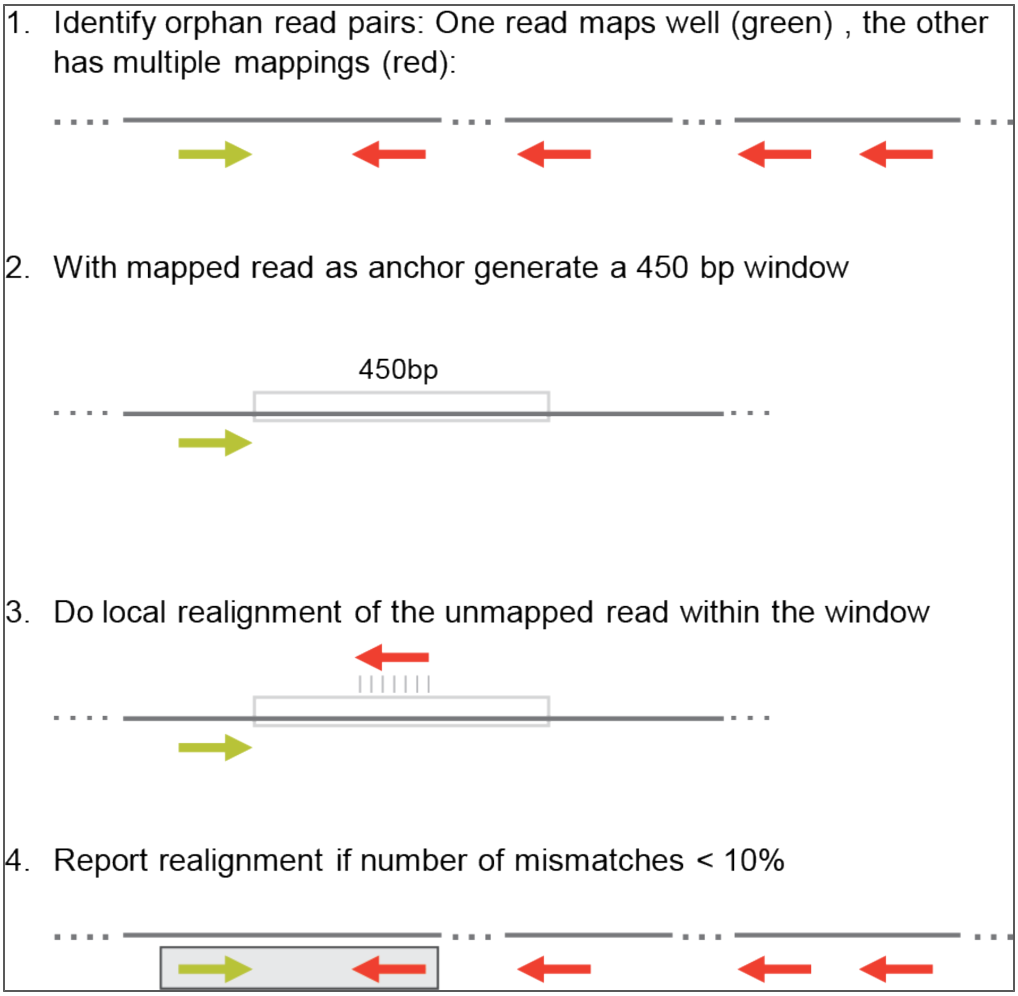

1. Align the first 32bp of reads
ATGCATGCACGGACTTTCGGGACTCTATCCGGCATCTATGGCTTTTCGC
ATGCATGCACGGACTTTCGGGACTCTATCCGGCATCTATGGCTTTTCGC
Seed
Read
Reference ATGCATGCACGGACTTTCGGGACTCTATCCGGCATCTATGGCTTTTCGC
?
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCGGCATCTATGGCTTTTCGC
2. Identify reads that do not align or hit a repetitive sequence
Seed
Read CCCCCCCCCCCCCCCCCCCCCCCCCCCCCGGCATCTA
?
Reference
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCGGCATCTATGGCTTTT
3. CASAVA v1.7: Align unmapped reads using multiple seeds
Seed
Seed
Read CCCCCCCCCCCCCCCCCCCCCCCCCCCCGGCATCTA
?
?
Reference
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCGGCATCTATGGCTTTT
3. CASAVA v1.8: Align unmapped reads using overlapping seeds
Read CCCCCCCCCCCCCCCCCCCCCCCCCCCCGGCATCTA
Seed
Seed
?
?
?
?
Reference
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCGGCATCTATGGCTTTT
4. Report seeds that hit a non-repetitive sequence
Read CCCCCCCCCCCCCCCCCCCCCCCCCCCCGGCATCTA
Reference
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCGGCATCTATGGCTTTT

Orphan Alignment

ELANDv2e performs orphan alignment by identifying read pairs for which only one of the reads aligns. ELANDv2e tries to align the other read in a defined window (by default 450 bp). If the number of mismatches is <10% of the read length, ELANDv2e reports the alignment.

The orphan alignment workflow is illustrated below.

Figure 23 Orphan Alignment


1. Identify orphan read pairs: One read maps well (green) , the other has multiple mappings (red):
2. With mapped read as anchor generate a 450 bp window
450bp
3. Do local realignment of the unmapped read within the window
4. Report realignment if number of mismatches < 10%

Alignment Performance Improvements

Many components of CASAVA have changed, which affects alignment performance. We therefore compared alignment percentage, mismatch rates, and CPU run times for three different configurations: CASAVA v1.7, v1.8 with semi-repeat resolution, and v1.8 with full repeat resolution. The dataset consisted of three lanes of HiSeq data from a single sample sequenced with TruSeq v3 chemistry. The analysis was performed on an iCompute cluster with j = 32. The results are summarized in the tables below; metrics are for reads passing filtering.

Table 20 Alignment and Mismatch Rate Comparison,

	v1.7		v1.8, semi-repeat		v1.8, full repeat	
	% Align	% Mismatch	% Align	% Mismatch	% Align	% Mismatch
Read 1	84.56	0.70	88.29	0.72	90.17	0.73
Read 2	81.92	1.39	85.81	1.44	87.56	1.44

Clearly CASAVA v1.8 aligns a higher percentage of reads, with full repeat alignment performing best. This higher alignment rate is mainly due to CASAVA v1.8's ability to align in more challenging repeat regions. Remarkably, even with more reads aligned in repeat regions, mismatch rates are still very similar.

Table 21 CPU Run Time Comparison

	v1.7 (CPU hours)	v1.8, semi-repeat (CPU hours)	v1.8, full repeat (CPU hours)
ELAND	523.28	518.40	855.40
orphanAligner	N/A	54.17	31.20
PickBestPair/ alignmentResolver	200.77	14.67	14.97
produceAlignStats	21.65	12.43	14.55
Other Processes	25.99	0.17	0.20
Total	771.72	599.85	916.33

Two observations jump out when looking at the CPU run time comparison:

- The module alignmentResolver (previously called PickBestPair) has been rewritten. This clearly results in much faster run times (200 hours for v1.7, versus 15 for v1.8).
- v1.8 full repeat resolution takes quite a bit longer than semi-repeat resolution (520 hours versus 855 hours for the ELAND step). However, it does result in a higher number of aligned reads. The best analysis type therefore depends on the project: is a shorter run time more important, or the highest number of aligned reads.

Variant Detection

Post-alignment CASAVA performs variant detection using two modules:

- The assembleIndels module (Grouper) detects candidate indels using singleton/orphan and anomalous read pairs. The assembleIndels module works well for detecting larger indels. The candidate indels detected by the assembleIndels module are passed on to the small variant caller for consolidation and genotyping.
- The callSmallVariants module genotypes and provides quality scores for SNPs and indels. Indels can be called from candidate indel evidence provided by both ELAND gapped-read alignments (for smaller indels) and from the assembleIndels module (for larger indels).

For each SNP or indel call the probability of both the called genotype and any non-reference genotype is provided as a quality score (Q-score). Reads are re-aligned around candidate indels to improve the quality of SNP calls and site coverage summaries.

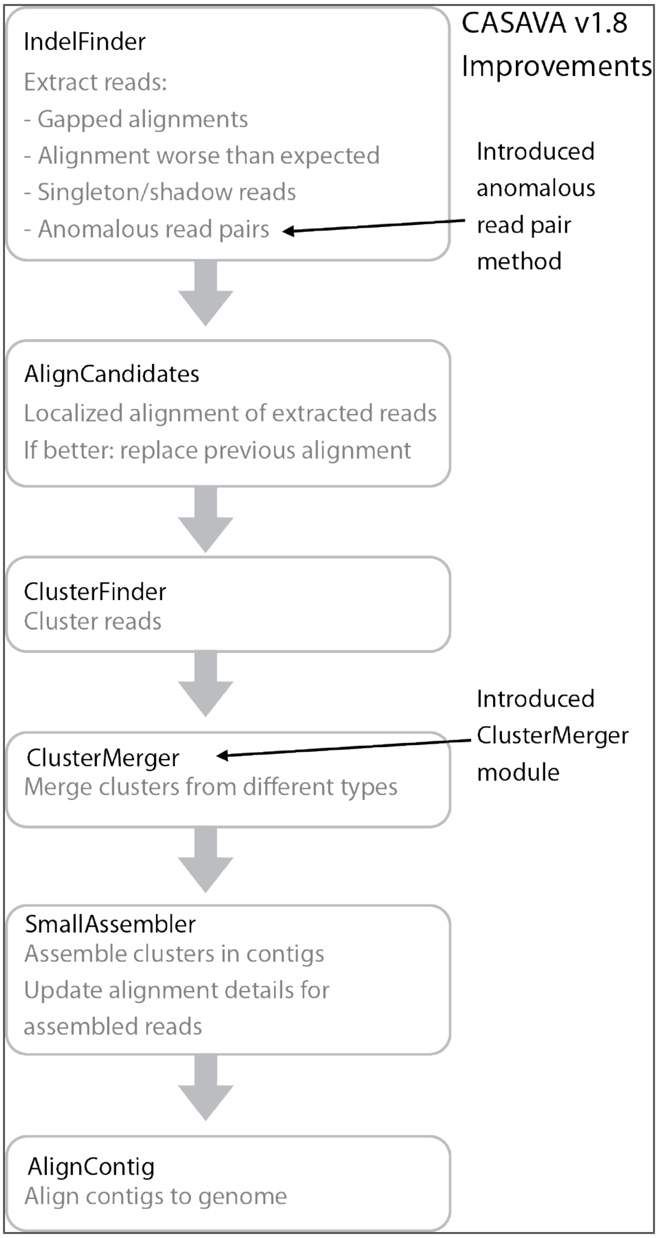
The callSmallVariants module also generates files which summarize the depth and genotype probabilities for every site in the genome. As a final step it produces tables and html-formatted reports of SNP and indel calls .

assembleIndels Module Improvements

The major changes for the assembleIndels module (Grouper) are:

- assembleIndels uses an additional method to identify indels. It finds read pairs that map anomalously (for example, with unexpected insert size), and identifies potential indels.
- assembleIndels merges indel calls detected through anomalous read pairs with those identified through singleton/orphan reads, and combines clusters that appear to correspond to the same event.

Figure 24 Changes to the assembleIndels Workflow



assembleIndels Algorithm

The assembleIndels module (Grouper) runs only during paired read DNA CASAVA builds. It uses orphan reads and, as of CASAVA 1.8, anomalous read pairs to detect indels.

Grouper detects indels in five stages:

1 Compute clusterings of non-aligned 'orphan reads'.

2 Compute clusterings of anomalous read pairs, with an insert size that is anomalously large (possible deletion) or small (possible insertion).

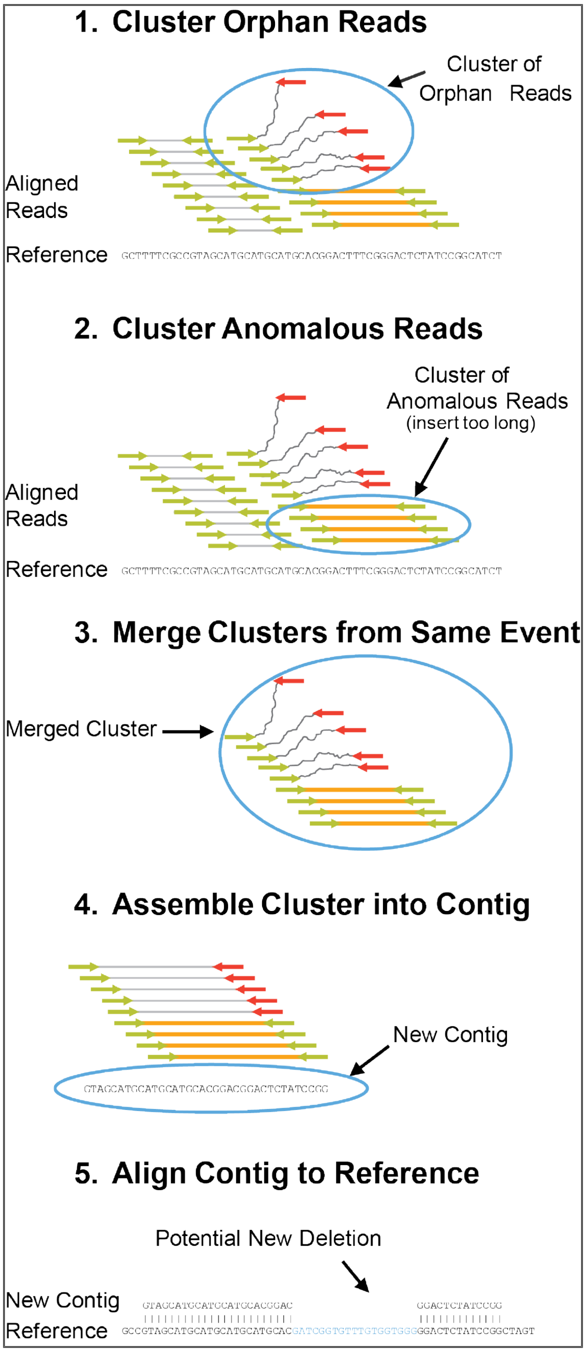
3 Combine clusters that appear to correspond to the same event.

4 Assemble them into contigs.

5 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.

The algorithm is illustrated in the figure below.

Figure 25 assembleIndels Algorithm


1. Cluster Orphan Reads
Cluster of Orphan Reads
Aligned Reads
Reference
GCTTTTCGCCGTAGCATGCATGCATGCACGGACTTTCGGGACTCTATCCGGCATCT
2. Cluster Anomalous Reads
Cluster of Anomalous Reads
(insert too long)
Aligned Reads
Reference
GCTTTTCGCCGTAGCATGCATGCATGCACGGACTTTCGGGACTCTATCCGGCATCT
3. Merge Clusters from Same Event
Merged Cluster
4. Assemble Cluster into Contig
New Contig
GTAGCATGCATGCATGCACGGACGGACTCTATCCGG
5. Align Contig to Reference
Potential New Deletion
New Contig
GTAGCATGCATGCATGCACGGAC
GGACTCTATCCGG
Reference
GCCGTAGCATGCATGCATGCATGCACGATCGGTGTTTGTGGTGGGGGACTCTATCCGGCTAGT

assembleIndels Components

The assembleIndels module contains the following components:

- **IndelFinder**—The IndelFinder component takes a sorted.bam 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 'orphan' reads not thought to be due just to poor base quality
- Reads from read pairs mapped anomalously. The expected relative orientation of read partners and the insert size statistics required to detect the anomalies are read per-lane from the s_?_pair.xml files produced during the alignment phase by alignmentResolver. An anomalously large insert size is defined as 3 standard deviations above the median; an anomalously small one as 5 standard deviations below the median.Two types of anomalous mapping are used:
  - Insert size anomalously large : Possible deletion
  - Insert size anomalously small : Possible insertion

IndelFinder tries to exclude reads for which the bad or non-existent alignment is just a consequence of poor base quality

- **AlignCandidates**—The component AlignCandidates does a dynamic programming alignment of each orphan 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.
- **ClusterFinder**—This takes the output of AlignCandidates (a list of orphan and badly aligning reads) and tries to group them in clusters of reads that are thought to have been caused by the same indel, based on genomic location.

  ClusterFinder separately clusters each type of anomalous read. The resulting clusters are labeled in the output file as:
  - Shadow/SemiAligned (orphan/semi aligned)
  - DeletionPair (insert size anomalously large)
  - InsertionPair (insert size anomalously small)
- **ClusterMerger**—This stage combines clusters of different types above that appear to correspond to the same event. One anticipated case is that of two Shadow/SemiAligned clusters and a DeletionPair cluster corresponding to the same (medium or larger scale) deletion. The currently supported merging mechanism is the combination of clusters of different types that share reads. This is possible as a read may be detected as being both SemiAligned as one partner in an anomalously mapped read pair. Apart from its role in merging related clusters, this step also ensures that reads are not multiply represented in the subsequent assembleIndels stages and downstream analysis.
- **SmallAssembler**—SmallAssembler takes the output of ClusterMerger 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 to describe its position in the contig.
- **AlignContig**—AlignContig does a dynamic programming alignment of contig to genome.

Variant Caller Methods

The callSmallVariants module calls SNPs and small indels from both the sorted alignment files (sorted.bam) and optionally also from the candidate indel contigs produced by assembleIndels. The procedure is outlined below.

- Read in read alignments and candidate indel contigs. Filter out read alignments based on quality checks, paired-end anomalies, or ELAND alignment score. Filter out contig alignments containing adjacent insertion/deletion events.
- Consolidate indel evidence from read and contig alignments to produce a set of candidate indels.
- Perform local read realignment using candidate indels.
- Call indels based on the set of alignments for each read which intersect/include a candidate indel.
- Select most likely read realignment for subsequent site counting and genotyping.
- Further filter individual basecalls based on mismatch density or ambiguity ('N').
- Use all remaining base calls to predict site genotypes and SNPs.
- Filter to remove SNP and indel calls near the centromeres and within high-copy number regions.

Read Filtering

The variant caller performs an initial read filtering to remove reads from both SNP and indel calling based on the following criteria:

- Any reads marked as failing primary analysis quality checks (e.g failing the purity filter) or marked as PCR or optical duplicates are removed from consideration.
- For paired-end reads, any reads which are not marked as being part of a ‘proper pair’ are removed from consideration. This is intended to remove any reads from chimeric pairs, with unmapped mates or with an anomalous pair insert size.
- Reads are filtered on ELAND alignment score. For paired-end reads the variant-caller removes by default any read with a paired-end alignment score less than 90, and for single-end reads, those with a single-end alignment score less than 10 are removed.

Detecting Indels and Realigning

The variant caller proceeds with candidate indel discovery and generation of alternate read alignments based on these candidate indels.

As part of this re-alignment process the variant caller selects a representative alignment to be used for site genotype calling and depth summarization by the SNP caller. This alignment is selected to be within a certain threshold of the most-likely of all alignments for a read, and any leading or trailing portions of the read with ambiguous support for 2 or more different alignments are marked as clipped. This representative alignment does not affect the indel caller – the indel calling process considers all alignments for each read without end-clipping. For diagnostic purposes, the set of reads which have their alignments altered during re-alignment may be written out to a separate BAM file for each chromosome using the `--variantsWriteRealigned` flag.

Indel Caller

The indel caller finds indels using a two stage process: In the first stage an indel must be identified as a candidate indel. In the second stage, after indel candidates have been identified, all intersecting reads are aligned to each indel, to the reference and to any alternate indels at the same site. The relative probabilities of these alignments for each read are used to call the indel's genotype and calculate the associated quality score.

Candidate Indel search

For the first stage of indel calling, candidate indels are identified from two sources of evidence:

- The first of these are from small indels already present in the input reads in the form of gapped alignments.
- The second source are alignments of locally assembled contigs to the reference provided by the assembleIndels module.

Every indel present in a conventional read alignment or assembleIndels contig is stored in a pool of potential indels.

Support for each one of these potential indels is measured as the number of read alignments which contain the indel. These alignments may be from the primary alignment or from reads used by Grouper to assemble each contig. If the number of reads supporting a potential indel is less than 3 or less than 2% of the total depth at the indel site, the indel cannot become a candidate. Additionally for short indels (of length 4 or less), if the number of supporting read is less than 10% of the total depth the indel cannot become a candidate. These cases are retained as 'private' indels in the reads alignments which support them. All other potential indels become candidate indels, subject to realignment and indel calling.

Note that as a consequence of the candidate indel discovery process, indels can be called using either gapped alignments or Grouper contig alignments as input, and the evidence from these two sources will be combined if both are available. Typically gapped alignments can be used to efficiently identify relatively small indels (roughly 1-10 bases in length), whereas local contig assembly can efficiently identify much larger indels. The greatest indel sensitivity can be achieved by generating candidate indels from both of these sources.

The parameters described for candidate indel filtration above are configurable as described in the *CASAVA User Guide*. Accepting too many candidate indels increases runtime and can lead to occasional spurious indel calls or poorly realigned reads in noisy regions of the genome.

Realignment and Indel Calling

For the second stage of indel calling, the variant caller realigns all intersecting reads to each candidate indel, in addition to aligning the read to the reference and any alternate indel candidates at the same site. It is common for reads which intersect the indel location to support the indel and reference alignments equally well, so the model is designed in such a way that these reads do not affect the genotype call.

The relative likelihoods of all alignments for each read are used to assign probabilities to each of three possible indel genotypes: homozygous, heterozygous or not present. The result of this calculation is reported as two quality scores.

- The first of these scores, Q(indel), expresses the probability that the indel is present in the sample as either a heterozygous or homozygous variant.
- The second score, Q(max_gt), expresses the probability that the most probable indel genotype, reported as the value max_gt, is correct.

To accommodate diverse applications, the CASAVA variant caller does not filter out low-confidence calls and thus prints out all indels with a Q(indel) value of 1 or greater. Summary statistics for indels are generated for a subset of higher confidence indels -- by default any indel with Q(indel) of 20 or greater is summarized in CASAVA's reports. Note that for calls with a very low Q(indel) score, it is possible that the most likely genotype will be ‘ref’, indicating that the indel is not present. This should be interpreted to mean that there is a non-trivial probability of the indel existing as a heterozygous variant at this site, but that the indel is more likely to be absent from the sample than present.

The predicted Q-scores 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, a spurious indel rate and an approximation of read mapping error. However note that artifactual indel signatures could still arise due to complex overlapping variants, atypical sample preparation, large-scale structural variants or other phenomena not accounted for by the model. The Q-score provided by the model should be interpreted with respect to these limitations.

Homopolymers

The indel calling model accounts for the probability of a spurious indel error as a function of homopolymer length and indel type. This spurious indel correction causes simple expansions and contractions of homopolymers to be predicted as less likely as homopolymer length increases. The spurious indel error probabilities are calculated from empirical observations. There is an option available in the small variant caller to replace these values with a single constant indel error probability to be used for all homopolymer lengths and indel types.

Overlapping Indels

Note that the model handles overlapping indels in an approximate fashion, by evaluating the probability of each indel allele compared to either the reference or any other indel allele at the same site. Thus it does not explicitly enumerate all possible pairs of alignment paths at the site to calculate the joint probability of the path for both haplotypes of a diploid sample—instead the method considers the current indel allele compared to all other possible alignment paths at the site.

This approximation effectively handles most simple overlapping indels, but will tend to undercall indels in regions with very high indel error rates. A consequence of this model is that where overlapping indels occur, the model does not strictly report a genotype, but rather the max_gt call reflects the copy number for each of the two indel alleles, and the probability of that copy number. Each indel allele of the two overlapping indels are reported on separate lines by the model. Due to the approximate nature of this model and the independent evaluation of each overlapping indel allele, it is possible that the most likely copy number for each allele could conflict (e.g. max_gt will not be het for both indel alleles), in the rare cases where this occurs the associated Q(max_gt) scores are typically very low.

Calling SNPs

Once the indels are called, and the reads are re-aligned to take into account the discovered indels, site genotyping and SNP-calling is conducted using the following steps:

6 Given the set of filtered and realigned reads, the variant caller next runs certain types of filtration on base calls within these reads:

- First, any contiguous trailing sequence of ‘N’ base calls are effectively treated as trimmed off of the ends of reads for the purpose of genotyping and depth calculation.

- Next the mismatch density filter is run on all reads to mask out sections of the read having an unexpectedly high number of disagreements with the reference. The current default mismatch density filter behavior is as follows:
  - Base calls are ignored where more than 2 mismatches to the reference sequence occur within 20 bases of the call. Note that this filter treats each insertion or deletion as a single mismatch.
  - 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.

All bases marked by the mismatch density filter, together with any 'N' base calls which remain after the end-trimming step, are filtered out by the variant caller. These filtered base calls are not used for site-genotyping but appear in the filtered base call counts in the variant caller's output for each site.

7 All remaining base calls are used for site-genotyping. The genotyping method heuristically adjusts the joint error probability calculated from multiple observations of the same allele on each strand of the genome to account for the possibility of error dependencies between these observations. The method accomplishes this by treating the highest quality base call of each allele from each strand as independent observations, leaving their associated base call quality scores unmodified. However, subsequent base calls for each allele and strand have their qualities adjusted to increase the joint error probability of that allele above the error expected from independent base call observations.

After running the site-genotyper on all positions, a set of unfiltered SNP sites is produced, consisting of all sites with Q(snp) > 0.

8 A final filtration step is taken to remove potentially spurious SNP calls near the centromeres and within high-copy number regions. This is done by calculating the mean used depth for each chromosome, and filtering out all SNP calls which occur at a used depth which is greater than 3 times this chromosomal mean.

Variant Detection Q-Scores

Quality Scores

A quality score (or Q-score) expresses an error probability. In particular, it serves as a convenient and compact way to communicate very small error probabilities.

Given an assertion, A, the probability that A is not true, P(~A), is expressed by a quality score, Q(A), according to the relationship:

$$Q(A) = -10 \log_{10}(P(\sim A))$$

where *P(~A)* is the estimated probability of an assertion A being wrong.

The relationship between the quality score and error probability is demonstrated with the following table:

Quality score, Q(A)	Error probability, P(~A)
10	0.1
20	0.01
30	0.001

Quality Scores Encoding

Quality scores are encoded into a compact form in FASTQ files which uses only one byte per quality value. In this encoding the quality score is represented as the character with an ASCII code equal to its value + 33, as of CASAVA 1.8. The following table demonstrates the relationship between the encoding character, the character's ASCII code, and the quality score represented.

WARNING
Quality score encoding schemes in previous version of CASAVA used an Illumina-specific offset value of 64.

Table 22 ASCII Characters Encoding Q-scores 0–40

Symbol	ASCII Code	Q-Score	Symbol	ASCII Code	Q-Score	Symbol	ASCII Code	Q-Score
!	33	0	/	47	14	=	61	28
"	34	1	0	48	15	>	62	29
#	35	2	1	49	16	?	63	30
$	36	3	2	50	17	@	64	31
%	37	4	3	51	18	A	65	32
&	38	5	4	52	19	B	66	33
'	39	6	5	53	20	C	67	34
(	40	7	6	54	21	D	68	35
)	41	8	7	55	22	E	69	36
*	42	9	8	56	23	F	70	37
+	43	10	9	57	24	G	71	38
,	44	11	:	58	25	H	72	39
-	45	12	;	59	26	I	73	40
.	46	13	<	60	27			

Variant Genotypes

In the context of resequencing a diploid individual, a genotype for a single site or indel indicates the two alleles that are present.

The set of diploid site genotypes considered by the CASAVA1.8 model for SNPs are {AA,CC,GG,TT,AC,AG,AT,CG,CT,GT}. If we consider an example site in the genome with a reference base of ‘C’, then the homozygous reference genotype is ‘CC’. A prediction of a SNP at that site is an assertion that the predicted genotype is not ‘CC’.

The CASAVA1.8 model for indels comprises three possible indel genotypes: homozygous, heterozygous or not present.

NOTE
It is possible to have high confidence that the genotype is not the reference without having high confidence in exactly what the genotype is at the site. In this situation there is strong evidence of a SNP but the exact genotype at the site is less certain.

Q(snp)

The SNP caller's site-genotyping methods take a set of base calls and associated qualities for each site, and produce a probability distribution over the 10 diploid genotype states {AA,CC,GG,TT,AC,AG,AT,CG,CT,GT}. Given this probability distribution, the value Q(snp) is a Q-score expressing the probability that the site genotype is not that of the homozygous reference.

NOTE
The diploid genotypes are printed out as two letter codes representing two unphased (single-base) alleles. For each heterozygous genotype the two alleles are provided in alphabetical order (e.g. 'CT' will be used and not 'TC').

For example, if the reference base is 'C' and the probability of the reference genotype 'CC' is 0.001, the value for Q(snp) is 30, reflecting a relatively high confidence that at least one non-reference allele exists at this site.

Prior Probabilities

An important component of the SNP-calling model is the prior probability distribution over diploid genotypes. The prior distribution expresses the information available about the genotype distribution at the site before sequencing. For instance, we expect that at any site in the genome, the sample genotype is likely be homozygous for an allele found at the same location in a reference sequence sampled from the same population.

The CASAVA 1.8 SNP-caller expresses this notion of prior expectation based on a reference sequence using its 'genomic' prior distribution, which is used to calculate Q(snp) and Q(max_gt). A specialized 'polymorphic' prior distribution is also used to compute Q(max_gt|poly_site), which is applicable to sites where there is a greater prior expectation of polymorphism, such as a set of sites from dbSNP.

The sections below provide an overview of both of these distributions.

Genomic Prior

When resequencing an individual from a population, there is a strong prior expectation that a randomly selected site in the sample assembly will be homozygous for the allele found in a reference chromosome from the same population. This expectation of similarity to a reference sequence in most portions of the genome is referred to below as the 'genomic' prior for the model.

For a simple illustration of a genomic prior, suppose that on average we expect 1 in 1000 sites in a sample chromosome to differ from a reference chromosome. If the reference at a particular site is 'A', then the Q-score for the reference genotype 'AA' in the absence of any sample observations will be approximately 30. Additionally, because of this prior we would expect the most likely genotype to be 'AA' even after observing a single non-reference basecall of modest quality. Thus the genomic prior has the effect of increasing the evidence required for a site to be called a SNP, and acts to reduce the rate of any false positive SNP predictions made by the model. For this reason the genomic prior is used to calculate the genotype probability distribution used for Q(snp) and Q(max_gt).

Polymorphic Prior

When considering a subset of sites from a genome which are already known to be polymorphic in a population, there is a much different prior expectation of the genotype distribution than in the scenario described above for all sites in the genome. A principal difference in this scenario is that our expectation that each site will be homozygous for the reference allele is much lower. We may also need to be able to examine these sites to distinguish strong evidence for the homozygous reference genotype from a site where no observations have been made. The polymorphic prior is used to compute the polymorphic-site genotype quality score: Q(max_gt|poly_site).

New Variant Calling Parameter: Theta

The parameter theta as used in the variant calling model refers to the expected proportion of differing sites between two chromosomes sampled from the population.

For site-genotyping, it is set by default to 1/1000, a value appropriate for human re-sequencing. Raising this value, to e.g. 1/100, would have the effect of increasing the prior expectation of a non-reference genotype and increase Q(snp) values.

The parameter theta for indels is set to a default value of 1/10,000.

Q(Indel)

Once the candidate indels are identified, the variant caller realigns all intersecting reads to each candidate indel, in addition to aligning the read to the reference and any alternate indel candidates at the same site. The relative likelihoods of all alignments for each read are used to assign probabilities to each of three possible indel genotypes: homozygous, heterozygous or not present.

The associated quality score Q(indel) expresses the probability that the non-reference indel allele referred to in the indel call exists in the sample as either a heterozygous or homozygous variant, analogous to Q(snp).

SNP Caller Reporting

The SNP caller reports the following files:

- **snps.txt**. SNPs for each chromosome are summarized within each chromosome directory in a file called snps.txt. This file contains SNPs which have been called by CASAVA’s callSmallVariants module.
- **sites.txt.gz**. As part of the SNP calling process, the variant caller also outputs information on coverage and consensus genotype for every mapped site in the genome. These results are found in each chromosome bin directory in a gzip compressed file called sites.txt.gz.
- **snps.removed.txt**. As a final noise filtration step, the SNP calls in the snps.txt files have been filtered to remove SNP calls in regions close to centromeres and other high copy number regions. The SNP calls filtered out by this procedure can be found in the file snps.removed.txt in each chromosomal bin directory.

The SNP-caller implemented in this module employs a probabilistic model which ultimately produces probability distributions over all diploid genotypes for each site in the genome. The primary values summarized from these distributions are a set of quality scores.

- **Q(snp)**. The value of Q(snp) expresses the probability that the genotype at this site is not the homozygous reference state.
- **Q(max_gt)**. The value of Q(max_gt) expresses the probability of the most-likely genotype state at this site, reported as the value "max_gt". Note that the value Q(max_gt) corresponds to a value referred to as "consensus quality" in SNP-calling methods such as samtools pileup.

- **Q(max_gt|poly_site)**.One additional score is provided by the SNP-caller which can be used to look at sites for which there is a strong expectation that the site is polymorphic. This value is Q(max_gt|poly_site), which expresses the probability of the most-likely genotype state at the site, assuming the site is polymorphic. This state is separately reported as the value "max_gt|poly_site". This genotype value and quality score provides greater sensitivity when looking at, for example, a particular set of polymorphic sites from dbSNP. This value should not be used to evaluate the genotype for every position in the genome as this would result in a high number of false positive SNP predictions.

To accommodate diverse applications, the CASAVA variant caller does not filter out low-confidence calls and thus prints all sites where Q(snp) is greater than zero to the snps.txt file. Summary statistics for SNPs are generated for a subset of higher confidence SNPs -- by default any SNP with Q(snp) of 20 or greater is summarized in CASAVA's reports. Note that for calls with a very low Q(snp) score, it is possible that the most likely genotype will be that of the homozygous reference, e.g. max_gt will be 'CC' for a position with a reference value of 'C'. This can be interpreted to mean that there is a non-trivial probability of a heterozygous SNP existing at this site, but that the homozygous reference genotype is still more likely than that of any non-reference variant.

Indel Caller Reporting

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 small variant caller from CASAVA's callSmallVariants module.

These indel calls have been filtered to remove those calls which are found at a depth greater than a certain multiple of the mean chromosomal depth. By default this multiple is set to 3. The purpose of this filtration is to remove indels calls in regions close to centromeres and other high copy number regions.

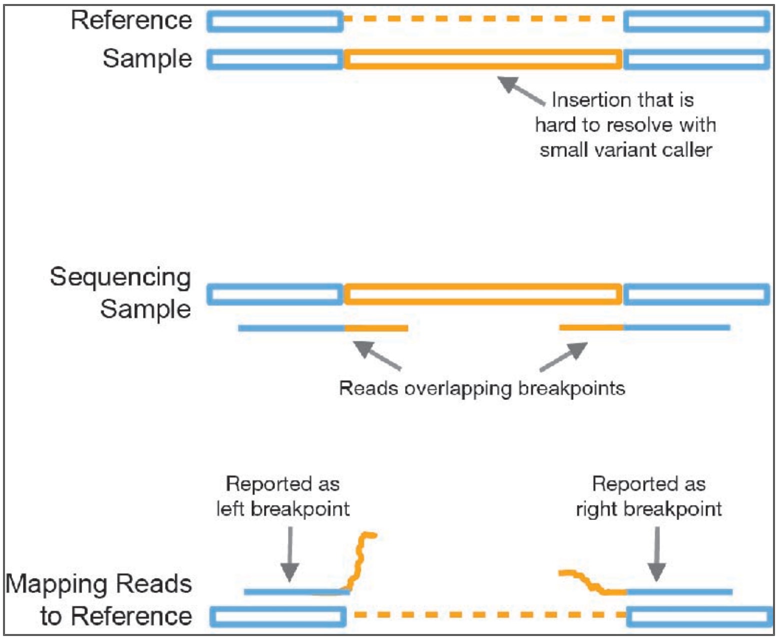
Three categories of indels are reported:

- Insertions
- Deletions
- Breakpoints. Breakpoint calls correspond to one of the two "ends" of a large indel or structural variant, for which the complete variant is either unknown or cannot be represented by the small variant caller. Breakpoint events are reported as either left or right breakpoints.

  A left breakpoint corresponds to a haplotype which can be mapped on the left side of the breakpoint location but not on the right. A right breakpoint indicates that a haplotype can be mapped to the right of the breakpoint location but not to the left. If a simple insertion or deletion were represented as two breakpoint calls, then they would occur on the forward strand as a left breakpoint followed by a right breakpoint.

  The figure below illustrates how two breakpoint calls could potentially be called corresponding to a large insertion in the sample relative to a population reference.

Figure 26 Left and Right Breakpoints


Reference
Sample
Insertion that is hard to resolve with small variant caller
Sequencing Sample
Reads overlapping breakpoints
Reported as left breakpoint
Reported as right breakpoint
Mapping Reads to Reference

Advanced Options for Variant Detection

This section lists advanced options for variant detection, which will help you fine-tune the variant calling.

Global Analysis Options

The options described below are global options used to specify analysis across different targets.

Table 23 Global Analysis Options for Variant Detection and Counting

Option	Application	Description
`--QVCutoff=NUMBER`	PE	Sets the paired-end alignment score threshold to NUMBER (default 90). Example: `--QVCutoff=60`
`--QVCutoffSingle=NUMBER`	SE, PE	Sets the single-read alignment score threshold to NUMBER (default 10). Example: `--QVCutoffSingle=60`
`--read=NUMBER`	PE	Limit input to the specified read only. Forces single-ended analysis on one read of a double-ended dataset. Example: `--read=1`
`--singleScoreForPE=VALUE`	PE	Sets the variant caller to filter reads with single score below QVCutoffSingle in PE mode YES|NO. Default NO. Example: `--singleScoreForPE=YES`
`--sortKeepAllReads`	SE, PE	Generate an archive BAM file. Keep all purity filtered, duplicate and unmapped reads in the build. These reads will be ignored during variant calling. Example: `--sortKeepAllReads`
`--toNMScore=NUMBER`	SE, PE	Minimum SE alignment score to put a read to NM. Default=-1 (-1 means option is turned off)
`--ignoreUnanchored`	PE	Ignore unanchored read pairs in indel assembly and variant

Option	Application	Description
		calling. Unanchored read pairs have a single-read alignment score of 0 for both reads. Example: `--ignoreUnanchored`

Options for Target assembleIndels

The options described below are used to specify analysis for target `assembleIndels`.

Table 24 Options for assembleIndels

Option	Application	Description
`--indelsSpReadThresholdIndels= NUMBER`	PE	Spanning read score threshold. The higher the single read alignment score before realignment, the more unlikely it is to see this pattern of mismatches given the read's quality values. Default threshold value is 25. Drop this value to add more reads into the indel finding process, at the possible expense of introducing noise. For an alignment with no mismatches this option should be set at zero.
`--indelsPrasThreshold=NUMBER`	PE	Paired read alignment score threshold. If a read has a paired read alignment score of at least this, then it is used to update the base quality stats for that sample prep. Default is calculated based off the data.
`--indelsAlignScoreThresh= NUMBER`	PE	If an alignment score for a read exceeds this threshold after realignment then the output file is updated to incorporate this new 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).
`--indelsSdFlankWeight=NUMBER`	PE	Number of standard deviations to use when defining the genomic interval to align the read to (default: 1).
`--indelsMinGroupSize=NUMBER`	PE	Only output clusters if they contain at least this many reads.
`--indelsSpReadThresholdClusters= NUMBER`	PE	Spanning read score threshold. This is calculated in exactly the same way as `--indelsSpReadThresholdIndels`. However it is used in the opposite way. Here the point 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.
`--indelsMinCoverage=NUMBER`	PE	Minimum coverage to extend contig (default 3).
`--indelsMinContext=NUMBER`	PE	Demand at least x exact matching bases either side of variant (default is 6). The idea here is to ensure that 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.
`--indelsSaveTempFiles`	PE	Add this flag to save intermediate output files from each stage of the indel assembly process.

Options for Target callSmallVariants

The options described below are used to specify analysis for target `callSmallVariants`.

Table 25 Workflow Options for callSmallVariants

Option	Application	Description
`--variantsSkipContigs`	PE	By default information from the assembleIndels module is used (and required) in paired-end DNA Sequencing analysis. This option disables use of indel contigs during variant calling, and only uses gapped alignment to find indels. Example: `--variantsSkipContigs`
`--variantsNoSitesFiles`	SE, PE	Do not write out the sites.txt.gz files. Example: `--variantsNoSitesFiles`
`--variantsNoReadTrim`	SE, PE	By default, the ends of reads can be trimmed if the alignment path through an indel is ambiguous. This option disables read trimming and chooses the ungapped sequence alignment for any ambiguous read segment. Note that this can trigger spurious SNP calls near indels. Example: `--variantsNoReadTrim`
`--variantsWriteRealigned`	SE, PE	Write only those reads which have been realigned to bam file: "sorted.realigned.bam" for each reference sequence. Example: `--variantsWriteRealigned`

Table 26 Read Mapping Options for callSmallVariants

Option	Application	Description
`--variantsIncludeAnomalous`	PE	Include paired-end reads which have anomalous insert-size or orientation. Note that "--variantsSEMapScoreRescue" must also be specified because ELAND gives anomalous reads a PE mapping score of zero.
`--variantsIncludeSingleton`	PE	Include paired-end reads which have unmapped mate reads. Note that "--variantsSEMapScoreRescue" must also be specified because ELAND gives singleton reads a PE mapping score of zero.
`--variantsSEMapScoreRescue`	PE	Include reads if they have an SE mapping score equal to or above that set by the "--QVCutoffSingle" option, even if the read pair fails the PE mapping score threshold.

Table 27 SNP and Indel Options for callSmallVariants

Option	Application	Description
`--variantsNoCovCutoff`	SE, PE	Disables the SNP and indel coverage filters detailed below for the options: `--variantsSnpCovCutoff` and `--variantsIndelCovCutoff`. This setting is recommended for targeted resequencing and RNA-Seq (Note it is already set by default for RNA-Seq). Example: `--variantsNoCovCutoff`

Table 28 SNP Options for callSmallVariants

Option	Application	Description
`--variantsSnpTheta=FLOAT`	SE, PE	The frequency with which single base differences are

Option	Application	Description
		expected between two unrelated haplotypes (default is 0.001). Example: `--variantsSnpTheta=0.002`
`--variantsSnpCovCutoffAll`	SE, PE	By default the mean chromosomal depth filter is based on "used-depth" (the number of basecalls used by the snp-caller after filtration) calculated from all known sites (non-N) in the reference sequence. When this option is set, the threshold and the filtration use the full depth at all known sites in the reference sequence. Example: `--variantsSnpCovCutoffAll`
`--variantsSnpCovCutoff=FLOAT`	SE, PE	SNPs are filtered out of the final output if the depth of reads used for that site is greater than this value times the mean chromosomal used-depth. (default 3.0) The filter may be disabled for targeted resequencing or other applications by setting this value to -1 (or any negative number). Example: `--variantsSnpCovCutoff=4`
`--variantsMDFilterCount=INTEGER`	SE, PE	The mismatch density filter removes all basecalls from consideration during SNP calling where greater than 'variantsMDFilterCount' mismatches to the reference occur on a read within a window of 1+2*'variantsMDFilterFlank' positions encompassing the current position. The default value for 'variantsMDFilterCount' is 2 and for 'variantsMDFilterFlank is 20. Set either value to less than 0 to disable the filter. Example: `--variantsMDFilterCount=3`
`--variantsMDFilterFlank=INTEGER`	SE, PE	The mismatch density filter removes all basecalls from consideration during SNP calling where greater than 'variantsMDFilterCount' mismatches to the reference occur on a read within a window of 1+2*'variantsMDFilterFlank' positions encompassing the current position. The default value for 'variantsMDFilterCount' is 2 and for 'variantsMDFilterFlank is 20. Set either value to less than 0 to disable the filter. Example: `--variantsMDFilterFlank=25`
`--variantsIndependentErrorModel`	SE, PE	This switch turns off all error dependency terms in the SNP calling model, resulting in a simpler model where each basecall at a site is treated as an independent observation. Example: `--variantsIndependentErrorModel`
`--variantsMinQbasecall=INTEGER`	SE, PE	The minimum basecall quality used for SNP calling. (default is 0). Example: `--variantsMinQbasecall=10`
`--variantsSummaryMinQsnp=INTEGER`	SE, PE	The snps.txt files contain all positions where Q(snp) > 0, however it is expected that only a higher Q(snp) subset of these will be used dependent upon the false positive tolerance of a user's workflow. For this reason summary statistics about the called SNPs are created at a higher "average-application" threshold, which can be set using this option (default is 20). Example: `--variantsSummaryMinQsnp=25`

Table 29 Indel Options for callSmallVariants

Option	Application	Description
`--variantsIndelTheta=FLOAT`	SE, PE	The frequency with which indels are expected between two unrelated haplotypes (default is 0.0001). See *New Variant Calling Parameter: Theta* on page 149 for more explanation. Example: `--variantsIndelTheta=0.0002`
`--variantsIndelCovCutoff=FLOAT`	SE, PE	Indels are filtered out of the final output if the local sequence depth is greater than this value times the mean chromosomal depth. The sequence depth of the indel is approximated by the depth of the site 5' of the indel. (default 3.0) The filter may be disabled for targeted resequencing or other applications by setting this value to -1 (or any negative number). Example: `--variantsIndelCovCutoff=4`
`--variantsCanIndelMin=INTEGER`	SE, PE	Unless an indel is observed in at least this many gapped or assembleIndels reads, the indel cannot become a candidate for realignment and genotype calling. (default: 3) Example: `--variantsCanIndelMin=4`
`--variantsCanIndelMinFrac= FLOAT`	SE, PE	Unless an indel is observed in at least this fraction of intersecting reads, the indel cannot become a candidate for realignment and genotype calling. (default: 0.02) Example: `--variantsCanIndelMinFrac=0.01`
`--variantsSmallCanIndelMinFrac= FLOAT`	SE, PE	In addition to the above filter for all indels, for indels of size 4 or less, unless the indel is observed in at least this fraction of intersecting reads, the indel cannot become a candidate for realignment and genotype calling. (default: 0.1) Example: `--variantsSmallCanIndelMinFrac=0.2`
`--variantsIndelErrorRate=FLOAT`	SE, PE	Set the indel error rate used in the indel genotype caller to a constant value of f (0<=f<=1). The default indel error rate is taken from an empirical function accounting for homopolymer length and indel type (i.e. insertion or deletion). This flag overrides the default behavior with a constant error rate for all indels. Example: `--variantsIndelErrorRate=0.5`
`--variantsSummaryMinQindel= INTEGER`	SE, PE	The indels.txt files contain all positions where Q(indel) > 0, however it is expected that only a higher Q(indel) subset of these will be used dependent upon the false positive tolerance of a user's workflow. For this reason summary statistics about the called snps are created at a higher "averege-application" threshold, which can be set using this option (default is 20). Example: `--variantsSummaryMinQindel=25`
`--variantsMaxIndelSize= INTEGER`	SE, PE	Sets the maximum indel size for realignment and indel genotype calling. Whenever an indel larger than this size is nominated by a de-novo assembly contig it is handled as two independent breakpoints. Note that increasing this value should lead to an approximately linear increase in variant caller memory consumption. The default value is 300 for paired-end builds and 50

Option	Application	Description
		for single-end builds. Example: `--variantsMaxIndelSize=200`

readBases Counting Method

This method is for exon and gene counts. Before counting CASAVA converts the alignments to splice junction into two shorter genomic alignments. Then CASAVA will count the number of bases (not the number of reads), that belong to exons, and genes. Bases within both original genomic and shorter genomic reads derived from spliced alignments participate in the exon and gene counts.

NOTE
Junction counts (in reads, not bases) are provided for convenience. Because alignments to the junctions are converted to the genomic reads before the counting, bases within reads aligned to splice junction are counted only once for exon and gene counts.

For splice junctions, counts are provided as the number of reads that cover the junction point. 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:

Exons/genes RPKM = 10^9 x 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:

Splice junctions RPKM = 10^9 x 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 >= 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.

Reference

Mortazavi A, Williams BA, McCue K, Schaeffer L, Wold (2008) Mapping and quantifying mammalian transcriptomes by RNA-Seq. Nature Methods. 5:585-7.

Appendix E

Qseq Conversion

Introduction

As of CASAVA 1.8, configureAlignment uses FASTQ files as input. If you have *_qseq.txt files that you want to analyze using CASAVA 1.8, use the Qseq Converter that converts *_qseq.txt files into FASTQ files.

The script has the following features:

- Creates a makefile to convert a directory of *_qseq.txt files to a directory tree of compressed FASTQ files following CASAVA 1.8 filename and directory structure conventions.
- If detected, configuration data used by configureAlignment are also transferred to the output directory.
- This script will not configure demultiplexing. The input directory must contain *_qseq.txt files which are either non-demultiplexed or already demultiplexed by another utility.

This appendix provides instructions to run the Qseq Converter.

Qseq Converter Input Files

The Qseq Converter needs the following input files:

- A BaseCalls directory with *_qseq.txt files. The Qseq Converter is specifically designed to convert *_qseq.txt files produced by OLB. It expects the *_qseq.txt files to follow the OLB naming conventions:
  `s_<lane>_<read>_<tile>_qseq.txt`
  With:
  `<lane>`: the lane number on the flow cell (1-8)
  `<read>`: the read number (1 or 2)
  `<tile>`: the tile number, left padded with '0', to 4 digits
  For example: s_1_1_0001_qseq.txt.

These files have the following format:

Field	Description
Machine name	Identifier of the sequencer.
Run number	Number to identify the run on the sequencer.
Lane number	Positive integer (currently 1-8).
Tile number	Positive integer.
X	X coordinate of the spot. Integer. As of RTA 1.6, OLB 1.6, and CASAVA 1.6, 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	Y coordinate of the spot. Integer. As of RTA 1.6, OLB 1.6, and CASAVA 1.6, 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	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.
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	Called sequence of read.
Quality	The calibrated quality string.
Filter:	Did the read pass filtering? 0 - No, 1 - Yes.

The Qseq Converter also looks for files that configureAlignment needs, and transfers them to its output directory. These files are:

- Config.xml file in the Basecalls folder.
- BustardSummary.xml.

Requirements to Run configureAlignment

To run CASAVA 1.8 configureAlignment on FASTQ files generated by the Qseq Converter, the following is required:

- The input *_qseq.txt files must be non-multiplexed or already demultiplexed into separate directories. If the converter finds reads 1, 2, and 3 from a multiplexed run, it will convert all three to FASTQ, but configureAlignment cannot run on these files.

- A BustardSummary.xml file must be found in the Qseq Basecalls folder, or the `--summary-file` argument to the Qseq Converter must point to an equivalent file.
- The BustardSummary.xml file must be copied to the FASTQ root folder and renamed DemultiplexedBustardSummary.xml. If the script can find the BustardSummary.xml, this will be done automatically.
- A config.xml file must be found in the Qseq Basecalls folder, or the `--config-file` argument to the Qseq Converter must point to an equivalent file. The config.xml file must be copied to the FASTQ root folder and renamed DemultiplexedBustardConfig.xml.

NOTE
configureAlignment requires SampleSheet.csv and SampleSheet.xml files but default versions of both files are created by the Qseq Converter

Running Qseq Converter

To convert *_qseq.txt files, you need to run the `configureQseqToFastq.pl` script. This sets up the run by generating a makefile and metadata. Running `make` or `qmake` then converts the *_qseq.txt files into FASTQ files.

9 Enter the following command to create a makefile for sequence alignment with the desired compression option.

```
/path-to-CASAVA/bin/configureQseqToFastq.pl --input-dir DIR
    [options]
```

10 Move into the newly created output folder. Type the “make” command for basic analysis:

```
make
```

NOTE
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.

Qseq Converter Parameters

The Qseq Converter parameters that can be entered are listed below.

Parameter	Description
`--input-dir DIRECTORY`	Path to *_qseq.txt directory No default.
`--output-dir DIRECTORY`	Path to root of CASAVA 1.8 unaligned directory structure. Directory will be created if it does not exist. Default: <input-dir>/QseqToFastq/Unaligned
`--fastq-cluster-count INTEGER`	Maximum number of fastq records per fastq file. Default: 4 000 .
`--config-file FILENAME`	Specify the Bustard config file to be copied to the fastq directory. Default: <input-dir>/config.xml
`--summary-file FILENAME`	Specify the Bustard summary file to be copied to the fastq directory. Default: <input-dir>/BustardSummary.xml
`--flowcell-id STRING`	Use the specified string as the flow cell id. Default value is parsed from the config-file.

Qseq Converter Output Data

The Qseq Converter generates the following output:

- gzipped FASTQ files in the directory structure configureAlignment expects (*configureAlignment Input Files* on page 48).
- If found, Qseq Converter copies the basecalling config.xml to the root of the FASTQ directory structure and renames it DemultiplexedBustardConfig.xml, which is the file expected by configureAlignment.
- If found, Qseq Converter copies the BustardSummary.xml to the root of the FASTQ directory structure and renames it DemultiplexedBustardSummary.xml, which is the file expected by configureAlignment.
- Qseq Converter also creates a default sample-sheet in the destination directory.
- IVC.htm and corresponding plots are in the same directory where the qseq files are.

NOTE
configureAlignment in CASAVA 1.8 will fail if you try to run it and the DemultiplexedBustardConfig.xml or the DemultiplexedBustardSummary.xml files are not there.

Appendix F

Export to SAM Conversion

Introduction

CASAVA 1.8 provides two SAM/BAM conversion pathways:

- Running the post-alignment 'sort' and 'bam' modules; see *Targets* on page 94
  Running the post-alignment sort and bam modules together offers sorting, PCR duplicate removal, indexing, and automatic chromosome renaming options, and by default it will write out a reference sequence file with chromosome labels that have been synchronized to the labels used in the BAM file. If the sort module is run in "archival" mode, the BAM file created will contain all of the reads provided in export.txt.gz files given as input.
- The illumina_export2sam.pl script.
  The illumina_export2sam.pl script provides basic conversion from export to SAM format without sorting, duplicate removal, conversion to BAM format or indexing. This script is intended to be used as one component in a custom post-alignment pipeline. Users desiring a 'turn-key' BAM creation method (e.g. to rapidly view reads in IGV) are encouraged to use the post-alignment sort and bam modules instead.

This section describes the usage of the illumina_export2sam.pl script . The script is located in CASAVA's bin directory and is an update to the SAMtools script export2sam.pl, redistributed in CASAVA under the MIT license (see http://sourceforge.net/projects/samtools/develop).

NOTE
Use CASAVA's illumina_export2sam.pl script instead of the SAMtools script. The illumina_export2sam.pl script has a number of updates that are important for proper conversion of ELANDv2e alignments. See the script header for a full list of these updates.

SAM Format

The Sequence Alignment/Map (SAM) format is a generic format for storing large nucleotide sequence alignments. SAM files have a .sam extension. and consist of one header section and one alignment section. The whole header section can be absent, but keeping the header is recommended.

This section provides the information relevant for the SAM files generated by CASAVA; a detailed description of the generic SAM format is available from samtools.sourceforge.net.

To generate a SAM file, see *Introduction* on page 168.

Header Section

The Illumina SAM files start with @PG, which indicates that the first line is a header line (@) of the program type (PG). The line is TAB-delimited and each data field has an explicit field tag, which is represented using two ASCII characters, as described below.

Tag	Description
ID	Program name
VN	Program version used to generate the file
CL	The configureBuild.pl command line used to execute or create the workflow for the SAM target

An example of a header line is shown below:

```
@PG  ID:CASAVA  VN:CASAVA-1.8.0  CL:/home/user1/CASAVA_
   20091209/bin/configureBuild.pl -p testBaseMiniBAM --targets
   bam
```

Alignment Section

The alignment section consists of multiple TAB-delimited lines with each line describing an alignment. Each line is:

```
<QNAME> <FLAG> <RNAME> <POS> <MAPQ> <CIGAR> <MRNM> <MPOS>
   <ISIZE> <SEQ> <QUAL> \ [<TAG>:<VTYPE>:<VALUE> [...]]
```

An example of a line in an alignment section is shown below:

```
HWI-EAS68_9096:2:15:512:204  99  c22.fa  14483804  29
 76M6I18M  =  14484254  550
AGAAATGTTCTAAAATTAAATTGTAGTGATGTCTGCACAACTTTGTAAGT
TTATAAAAAATAATTGACTTGTACACTTAATATTAATGAGTTGTATGGCA
HGFGHGHHGHHIHEGHHHEHHHFEECBBFBGFHHHEHHHEHHGHHHDHHD
HEH>EFHH=?CC?C6HHHEED?FFFHHHF=HEHH?HHH;HGHHGBHFBD?
XD:Z:76^6$18 SM:i:1 AS:i:29
```

The format of each field is explained in the following table.

Field	Description
QNAME	Query pair name if paired; or Query name if unpaired. This consists of the following sequence: <Machine>_<Run number>:<Lane>:<Tile>:<X coordinate of cluster>:<Y coordinate of cluster>
FLAG	Bitwise flag. For a description, see *Bitwise Flag Values* on page 170.

Field	Description
RNAME	Reference sequence name. The contains the export.txt.gz file match chromosome value, and if the export.txt.gz file "Match contig" field is not empty, the SAM RNAME field will be appended with a "/" character followed by the match contig name. See *Export.txt.gz* on page 77.
MAPQ	Mapping quality (Phred-scaled posterior probability that the mapping position of this read is incorrect)
CIGAR	Extended CIGAR string. For a description, see *Extended CIGAR Format* on page 170.
MRNM	Mate Reference sequence NaMe; "=" if the same as <RNAME>
MPOS	1-based leftmost mate position of the clipped sequence
ISIZE	Inferred insert size
SEQ	query SEQuence; "=" for a match to the reference; n/N/. for ambiguity; cases are not maintained
QUAL	query QUALity; ASCII-33 gives the Phred base quality
TAG	TAG for an optional field. For a description, see *Optional Fields* on page 171.
VTYPE	Value TYPE for an optional field. For a description, see *Optional Fields* on page 171.
VALUE	Match <VTYPE> for an optional field. For a description, see *Optional Fields* on page 171.

Bitwise Flag Values

The FLAG field in the alignment section is a bitwise flag. The meaning of predefined bits is shown in the following table:

Hexadecimal Value	Decimal Value	Description
0x0001	1	The read is paired in sequencing, no matter whether it is mapped in a pair
0x0002	2	The read is mapped in a proper pair (depends on the protocol, normally inferred during alignment)
0x0004	4	The query sequence itself is unmapped
0x0008	8	The mate is unmapped
0x0010	16	Strand of the query (0 for forward; 1 for reverse strand)
0x0020	32	Strand of the mate
0x0040	64	The read is the first read in a pair
0x0080	128	The read is the second read in a pair
0x0100	256	The alignment is not primary (a read having split hits may have multiple primary alignment records)
0x0200	512	The read fails platform/vendor quality checks
0x0400	1024	The read is either a PCR duplicate or an optical duplicate

NOTE
The bitwise flag means that if multiple conditions are true, the values are added, and only the total value is reported. For example, if a read is paired in sequencing (value 1), the mate is unmapped (value 8), and the read is the first read in a pair (value 64) a total of 1 + 8 + 64 = 73 is reported).

Extended CIGAR Format

A CIGAR string is comprised of a series of operation lengths plus the operations. The conventional CIGAR format allows for three types of operations: M for match or mismatch, I for insertion and D for deletion. The extended CIGAR format further allows four more operations, as is shown in the following table, to describe clipping, padding and splicing:

Operation	Description
M	Alignment match (can be a sequence match or mismatch)
I	Insertion to the reference
D	Deletion from the reference
N	Skipped region from the reference
S	Soft clip on the read (clipped sequence present in <seq>)
H	Hard clip on the read (clipped sequence NOT present in <seq>)
P	Padding (silent deletion from the padded reference sequence)

For example, the CIGAR string 30M1I69M means 30 bases aligning to the reference (30M), 1 base insert (1I), and 69 bases aligning (69M).

Optional Fields

Optional fields are in the format: <TAG>:<VTYPE>:<VALUE>, for example: XD:Z:73T26.

Each tag is encoded in two alphanumeric characters and appears only once for an alignment. Illumina SAM files may use some or all of the following optional fields:

Tag	Description
SM	ELAND single-read alignment score
AS	ELAND paired-read alignment score
XD	String for mismatching positions
XC	Provides information to distinguish different unmapped read types

The <VTYPE> describes the value type in the optional field. Valid types in SAM are described in the following table.

Type	Description
A	Printable character
i	Signed 32-bit integer
f	Single-precision float number
Z	Printable string
H	Hex string (high nybble first)

The <VALUE> field format is defined by the tag:

Tag	Value Field
SM	The <VALUE> field contains the ELAND single-read alignment score
AS	The <VALUE> field contains the ELAND paired-read alignment score
XD	The <VALUE> field contains the string for mismatching positions. • Matching bases are numbered. For example, a value of 100 means that 100 consecutive bases match the reference. • Mismatched bases are indicated by a base (ACGTN), where the letter indicates the reference base.

Tag	Value Field
	• Insertions and deletions start with a "^" character and are closed with a "$" character. A number indicates an insertion in the read of that size, a base (or number of bases) indicate the sequence of the reference that was deleted in the read. For example, the string 30^1$28G means the following: • 30: 30 bases matching reference • ^1$: one base insertion in read • 28: 28 bases matching reference • G: reference base G is mismatched in read
XC	Provides read status information normally conveyed in the chromosome field of the export.txt file for unmapped reads. Specificially, "XC:Z:QC" is used to mark an ELAND QC failure read, "XC:Z:RM" is used to mark an ELAND repeat mask read, and "XC:Z:CONTROL" is used to mark a control read. No optional field is added to reads which are marked as no match ("NM") in the export file – it is understood that this is the default status of an unmapped read

Usage

For export to SAM conversion, enter the following:

```
path-to-CASAVA/bin/ illumina_export2sam.pl --read1=FILENAME
    [options] > outputfile.sam
```

Make sure to specify an output file, else the output gets written to the screen.

The options are described in the table below.

Option	Description
`--read1=FILENAME`	Read1 export file (mandatory). File may be gzipped with ".gz" extension.
`--read2=FILENAME`	Read2 export file. File may be gzipped with ".gz" extension.
`--nofilter`	Include reads that failed the basecaller purity filter.
`--qlogodds`	Assume export file(s) use logodds quality values as reported by Pipeline prior to 1.3.
`--version`	Prints version information.
`--help`	Prints on screen usage guide.

Example

An example of illumina_export2sam.pl use is as follows:

```
path-to-CASAVA/bin/illumina_export2sam.pl --read1=NA10831_
    ATCACG_L001_R1_001_export.txt.gz --read2=NA10831_ATCACG_
    L001_R2_001_export.txt.gz > s_2_converted.sam
```

This will write an output file s_2_converted.sam that contains the paired-end reads from s_2_1_export.txt and s_2_2_export.txt.

Glossary

"

"Density" Probe Interval

A user defined value (normal or dense).

#

# of Probes

The total number of probes for the specified targets. This depends on sequence characteristics of the defined regions. Some regions are more amenable to probe design than others.

A

Added

The time at which the target was added to the project.

Average Quality

A bioinformatic prediction of how well probes will be able to pull down target sequences. This is based on CG content, the uniqueness of the sequence, and probe uniformity.

B

Bayesian model

A Bayesian model provides a means to update a prior hypothesis based on evidence. As an example, in a Bayesian genotype model we may have a prior hypothesis that our sample genotype matches that from a reference sample with probability q. After accounting for evidence of the sample genotype in the form of sequencing reads which are inconsistent with the reference genotype, our hypothesis is updated such that the probability of the sample genotype matching the reference has been reduced to a value less than q.

C

Chromosome (Chr.) Start-Stop

The beginning and ending coordinates of the target region in base pairs. You define these values.

D

De Bruijn graph

A De Bruijn graph of m symbols is a graph representing overlaps between sequences. De Bruijn graphs are used for de novo assembly of short read sequences into a genome.

deprecated

Deprecated refers to software features that are superseded and should be avoided. Although deprecated features remain in the current version, their use may raise warning messages, and deprecated features may be removed in the future. Features are deprecated—rather than being removed—in order to provide backward compatibility and give programmers who have used the feature time to bring their code into compliance with the new standard.

Design Region

When specified, the region of focus within the target region. Choosing Exon Only will target probe design to just the exons within the target region. Choosing Full Region will target probes across the entire region in both introns, exons, and intergenic regions.

G

Gap

A gap is any base positions defined as a region of interest (exon or otherwise), that is intended for enrichment, but fails to be theoretically or physically enriched by the probe set. Gaps are one or more bases long and tend to show up more frequently at the ends of targeted regions when "standard" spacing is used rather than "dense" spacing.

I

Insert Size

The average length of the genome DNA insert in the prepared Library (library – adapters).

K

kmer-hashing

Hashing refers to the use of subfragments of a particular read to find matching pieces of DNA in a hash table. k-mer means the size of the fragment used for hashing.

L

Lebel

Text tags that can be used for filtering or annotation of target regions.

M

My Term

My definition

N

nybble

A nybble, sometimes spelled "nibble," is a set of four bits. Since there are eight bits in a byte, a nybble is half of one byte.

O

orphan reads

An orphan read is the unaligned part of paired reads for which only one read aligned. Identical to shadow read.

P

Pulldown region

The region of the genome that a probe or set of probes will be able to enrich.

Q

Quality

The quality score for the entire target.

S

shadow read

A shadow read is the unaligned part of paired reads for which only one read aligned. Identical to orphan read.

singleton

A singleton is the aligned part of paired reads for which only one read aligned.

T

Target

The region you specified for probe design. A project can can contain numerous targets, and these targets may be discontiguous.

U

Uncovered Target Distance

Target regions for which probes cannot be designed.

W

wrapper script

A wrapper script is a script whose main purpose is to call a second function in a computer program with little or no additional computation.

Index

Technical Assistance

For technical assistance, contact Illumina Customer Support.

Table 30 Illumina General Contact Information

Illumina Website	http://www.illumina.com
Email	techsupport@illumina.com

Table 31 Illumina Customer Support Telephone Numbers

Region	Contact Number	Region	Contact Number
North America	1.800.809.4566	Italy	800.874909
Austria	0800.296575	Netherlands	0800.0223859
Belgium	0800.81102	Norway	800.16836
Denmark	80882346	Spain	900.812168
Finland	0800.918363	Sweden	020790181
France	0800.911850	Switzerland	0800.563118
Germany	0800.180.8994	United Kingdom	0800.917.0041
Ireland	1.800.812949	Other countries	+44.1799.534000

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/support/documentation.ilmn. 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. To register for an iCom account, please visit https://icom.illumina.com/Account/Register.

Illumina, Inc.
9885 Towne Centre Drive
San Diego, CA 92121-1975
+1.800.809.ILMN (4566)
+1.858.202.4566 (outside North America) techsupport@illumina.com
www.illumina.com