Applied Biosystems SOLiD™ 3 System

SOLiD™ Analysis Tools (SAT) V3.0 User Guide

Applied Biosystems

For Research Use Only. Not for use in diagnostic procedures.

Part Number 4392959 Rev. D
04/2009

Contents

Preface

How to use this guide

Purpose of this guide

This Guide provides:

- A high-level overview of data processing with the SOLiD™ 3 System
- A description of the most important files generated
- A description of the SOLiD™ Analysis Tools (SAT) pipeline
- Locations of the files generated by analysis
- A summary of key aspects of color space

Audience

This guide is intended for advanced users of SOLiD™ Analysis Tools (SAT).

Assumptions

This guide assumes that your SOLiD™ 3 System has been installed by an Applied Biosystems technical representative, or that Global SETS (SETS and SAT) software is installed on a supported off-line cluster.

This guide also assumes that you have a working knowledge of the Microsoft® Windows® XP operating system and Linux.

Text conventions

This guide uses the following conventions:

- `Code` text indicates user action or examples of code that appear in the software. For example:

  ```
  mate.pairs.run=1
  ```

- *Italic* text indicates new or important words and is also used for emphasis. For example:

  Before analyzing, *always* prepare fresh matrix.

- A right arrow symbol (▸) separates successive commands you select from a drop-down or shortcut menu. For example:

  Select **File ▸ Open ▸ Spot Set**.

  Right-click the sample row, then select **View Filter ▸ View All Runs**.

User attention words

Two user attention words appear in Applied Biosystems user documentation. Each word implies a particular level of observation or action as described below:

Note: – Provides information that may be of interest or help but is not critical to the use of the product.

IMPORTANT! – Provides information that is necessary for proper instrument operation, accurate chemistry kit use, or safe use of a chemical.

Examples of the user attention words appear below:

Note: This Tag ID is used to describe the bead and its data in all the files.

IMPORTANT! It is important to realize that these values are relative to the reference.

How to obtain more information

Related documentation

The following related documents are shipped with the system:

- ***SOLiD™ 3 System Site Preparation Guide*** **(PN 4386998)** - Describes how to ready your location and set up for a SOLiD™ 3 System installation.
- ***SOLiD™ 3 System User Guide*** **(PN 4391587)** - Provides detailed descriptions of how to use the SOLiD™ 3 System.
- ***SOLiD™ 3 System Instrument Operation Guide*** **(PN 4407430) -**Provides detailed description of how to operate the instrument using Instrument Control Software (ICS).
- ***SOLiD™ 3 System - SETS Software Getting Started Guide*** **(PN 4389302) -** Provides brief, step-by-step procedures for SOLiD™ 3 Experiment Tracking System (SETS) software. It is designed to help you quickly learn to use the SETS software.

Send us your comments

Applied Biosystems welcomes your comments and suggestions for improving its user documents. You can e-mail your comments to:

techpubs@appliedbiosystems.com

IMPORTANT! The e-mail address above is only for submitting comments and suggestions relating to documentation. To order documents, download PDF files, or for help with a technical question, go to **http://www.appliedbiosystems.com**, then click the link for **Support**. (See “How to obtain support” below).

How to obtain support

For the latest services and support information for all locations, go to **http://www.appliedbiosystems.com**, then click the link for **Support**.

At the Support page, you can:

- Search through frequently asked questions (FAQs)
- Submit a question directly to Technical Support
- Order Applied Biosystems user documents, MSDSs, certificates of analysis, and other related documents
- Download PDF documents
- Obtain information about customer training
- Download software updates and patches

In addition, the Support page provides access to worldwide telephone and fax numbers to contact Applied Biosystems Technical Support and Sales facilities.

Data Analysis Overview 1

SOLiD™ 3 System overview

The Applied Biosystems SOLiD™ 3 System provides parallel sequencing of clonally amplified DNA fragments linked to magnetic beads. The sequencing methodology is based on sequential ligation with dye-labeled oligonucleotides. All fluorescently labeled oligonucleotide probes are present simultaneously, competing for incorporation. After each ligation, fluorescence is measured before another round of ligation takes place.

This ligation-based chemistry eliminates dephasing. The use of a two-base encoding mechanism, which interrogates each base twice for errors during sequencing, discriminates between true polymorphisms and system noise. The SOLiD™ Analyzer software allows customers to monitor the runs in real-time, and provides basic data analysis tools.

The SOLiD™ 3 System consists of:

- SOLiD™ Analyzer
- All ancillary equipment
- SOLiD™ Instrument Control Software (ICS) software, which is described in the *Applied Biosystems SOLiD™ 3 System Instrument Operation Guide* (PN 4407430)
- SOLiD™ Experiment Tracking System (SETS) software, which is described in the *Applied Biosystems SOLiD™ 3 System - SETS Software Getting Started Guide* (PN 4389302)
- SOLiD™ Analysis Tools (SAT) software, which is described in this document

Software system workflow

This section gives the workflow of the SOLiD™ 3 System software analysis. The software applications used to set and control data analysis include SOLiD™ Instrument Control Software (ICS), SOLiD™ Experiment Tracking System (SETS), and SOLiD™ Analysis Tools (SAT), as shown in Table 1.

Table 1 SOLiD™ software applications

Software application	Type	Function
ICS	Windows application	Instrument operation
SETS	Browser-based application	Reports and re-analysis
SAT	Linux command-line tool	Command-line tool for analysis

See Figure 1 on the next page for a representation of how ICS, SETS, and SAT work together. Refer to sections later in this chapter for overviews about ICS and SETS. These sections list the documents that describe ICS and SETS in detail. The rest of this document describes SAT.

SOLiD™ 3 System analysis can be organized into primary, secondary, and tertiary analysis (see Table 2).

Table 2 Types of data analysis

Type of analysis	Description
Primary	Includes universal processes for data generation, collection, and raw processing. Images for each cycle are analyzed. Data are clustered and normalized. For each tag, a sequential (sequence-ordered) set of color space calls is produced. Normalization produces a set of quality metrics.
Secondary	Analyzes application-specific data at the sequence level. Data are aligned to the reference sequence. If the experiment is a mate-paired run, the system analyzes the mate pairs.
Tertiary	Generates biological interpretation specific to an application

For additional secondary and tertiary analysis tools, visit the SOLiD™ Software Development Community website (**http://solidsoftwaretools.com**). You can integrate stand-alone tools from the SOLiD™ Software Development Community with the SAT pipeline to perform more automated analysis. Secondary analysis can be performed on an off-line analysis cluster instead of on the instrument.

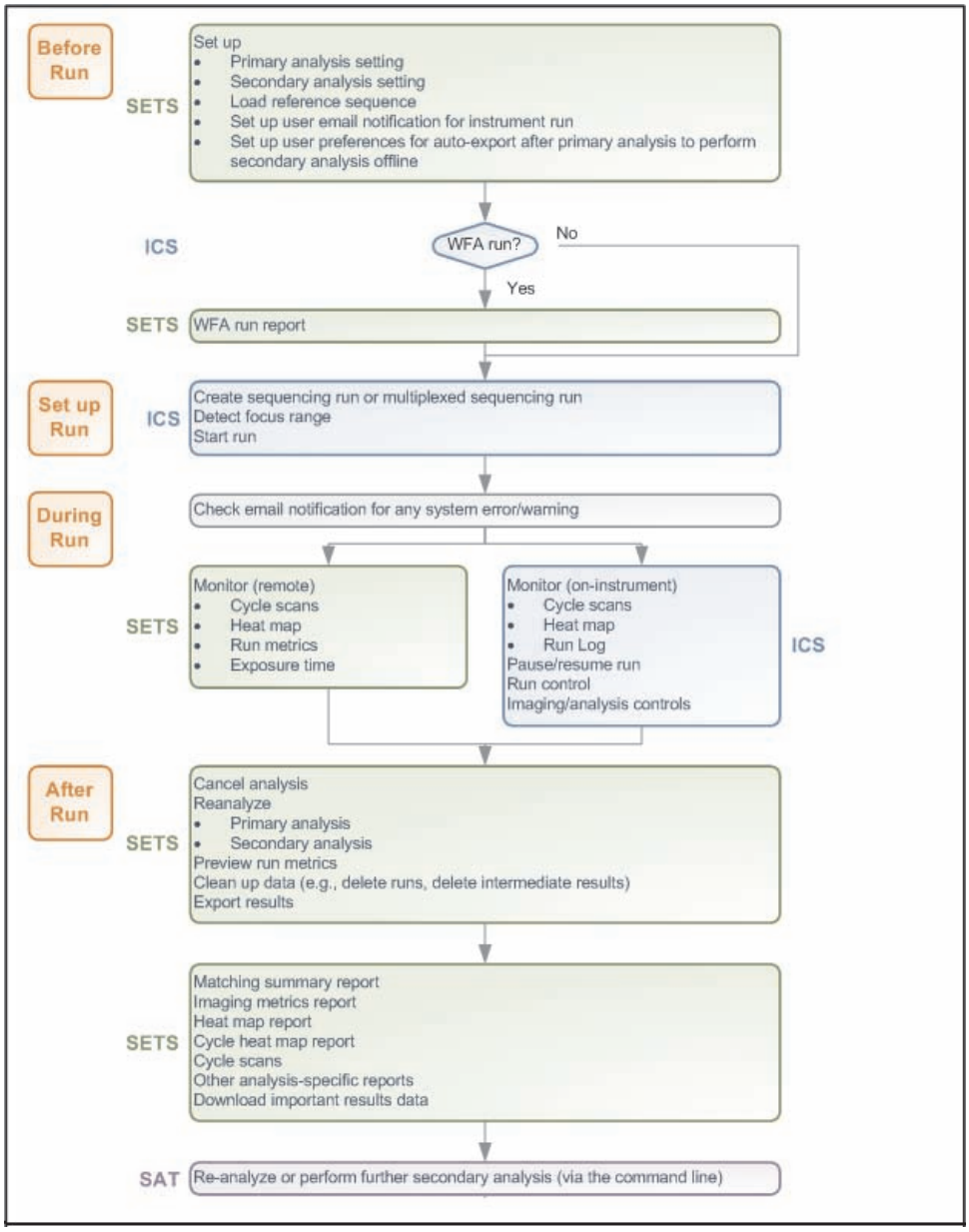


Figure 1 SOLiD™ workflow between ICS, SETS, and SAT.

Types of SOLiD™ 3 experiments

On the Applied Biosystems SOLiD™ 3 System, you can perform three types of runs: *workflow analysis (WFA), sequencing (standard)*, and *multiplexed sequencing* (see Table 3).

Table 3 Run types on the SOLiD™3 System

	WFA	Sequencing (standard)	Multiplexed sequencing
Purpose	• Assess various preparations of templated beads to determine potential quality of sequence data • Evaluate fraction of P2-positive beads • Use as a tool to determine deposition density for sequencing slides	Generate sequencing data for fragment or mate-paired libraries	Generate multiplexed sequencing data for fragment libraries
Run summary	• P1 and P2 bead counting • Single ligation cycle • Report generation	Up to 10 ligation cycles for each of 5 primers resulting in 50 bases per tag‡	Up to 10 ligation cycles for each of 5 primers resulting in 50 bases per tag *and* 1 ligation cycle for each of 5 primers resulting in 5 bases per barcode tag
Estimated run time§	~4 to 5 hours	• ~3 to 4 days for 25 bp • ~6 to 7 days for 50 bp	• ~6 to 7 days for 50 bp • ~1 day for barcode
Deposition chamber	4-well	• 1-well • 4-well • 8-well	• 1-well • 4-well • 8-well
Number of beads	15 million beads per well	• 310 million beads per well (1-well) • 60 million beads per well (4-well) • 30 million beads per well (8-well)	• 310 million beads per well (1-well) • 60 million beads per well (4-well) • 30 million beads per well (8-well)

‡ One tag for fragment libraries and 2 tags for mate-paired libraries
§ Total run time for dual slide run. Times may deviate depending on imaging time.

For more information about the nature of SOLiD™ experiments and how to run them, refer to the *Applied Biosystems SOLiD™ 3 System Instrument Operation Guide* (PN 4407430).

Workflow analysis (WFA) run

You can optimize sequencing results by performing workflow analysis (WFA) runs. A WFA run analyzes a quadrant of a slide that undergoes a single ligation cycle. The quadrant contains beads deposited at a lower density than the density of beads deposited for a sequencing run.

A WFA run determines the:

- **Optimal library concentration**: the library concentration for optimal preparation of templated beads using the library. You use this library concentration for any preparation of templated beads for that library when the scale of templated bead preparation is the same.
- **Bead enrichment efficiency**: the proportion of beads that have been successfully amplified using emulsion PCR (ePCR) as a fraction of the total number of beads prepared. You use this value to more accurately deposit successfully amplified beads for a sequencing run.

WFA runs require the same materials as those materials needed for sequencing runs. If you perform multiple WFA runs routinely, you should order additional SOLiD™ Instrument Buffer Kits.

For more information about a WFA run, refer to Applied Biosystems *SOLiD™ 3 System Instrument Operation Guide*, PN 4407430.

Sequencing run

During a SOLiD™ sequencing run, two probe sets are used to maximize the fraction of "mappable" beads, read length, and sequencing throughput. (Mappable beads are beads, amplified with template, that map to the reference genome.) This protocol must be used for sequencing 50-bp reads of both mate-paired and fragment libraries. Unlike terminator-based sequencing, SOLiD™ does not collect base-sequencing information. Instead, five rounds of primers (Primers A, B, C, D, and E) are used to sequence template by ligation of di-base labeled probes. For sequencing of fragment libraries, the set of primers used is specific to the P1 Adaptor.

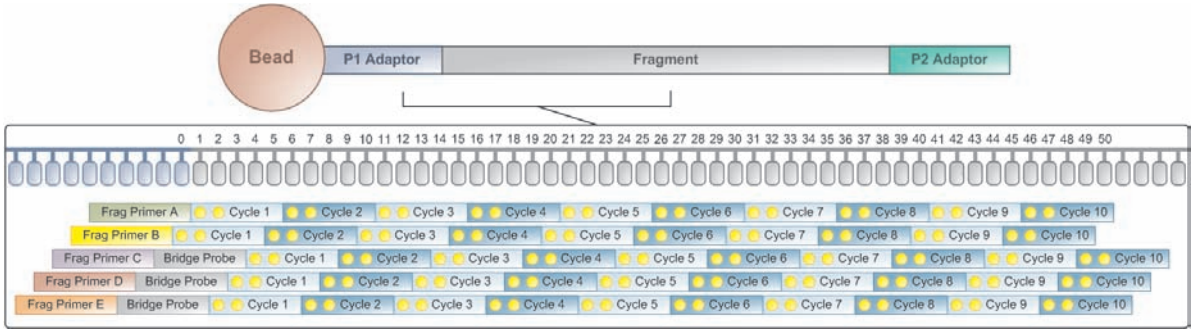


Figure 2 SOLiD™ 3 System interrogation of nucleotide positions for a 50-bp fragment sequencing run.

For more information about a sequencing run, refer to Applied Biosystems *SOLiD™ 3 System Instrument Operation Guide*, PN 4407430.

Multiplexed sequencing

For sequencing of barcoded fragment libraries, the set of primers used to sequence the fragment is specific to the P1 Adaptor, whereas the set of primers used to sequence the barcode sequence is specific to the Internal Adaptor.

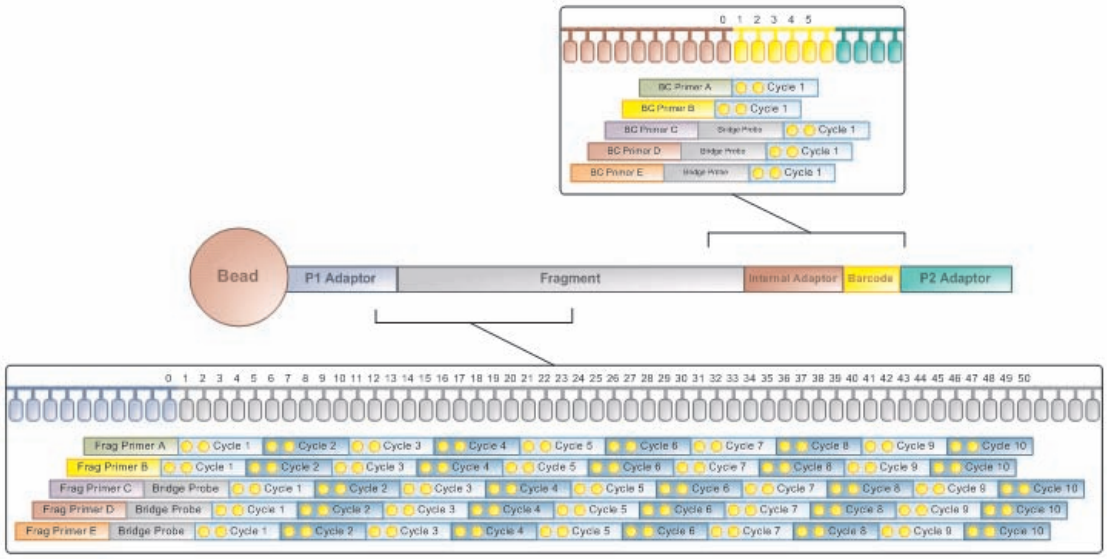


Figure 3 SOLiD™ 3 System interrogation of nucleotide positions for a 50-bp fragment and 5-bp barcode sequencing run.

For more information about a multiplexed sequencing run, refer to Applied Biosystems *SOLiD™ 3 System Instrument Operation Guide*, PN 4407430.

ICS overview

SOLiD™ 3 System ICS (Instrument Control Software) features the following enhancements:

- Wizard-based workflow for step-by-step assistance in creating runs
- Multiplexing protocol support, in addition to Sequencing and WFA run types
- Import capability to generate runs directly from data spreadsheets
- PanelBrowser™ to troubleshoot suspect or failed panels
- Overall performance and interface improvements to the software

For more information, refer to the *Applied Biosystems SOLiD™ 3 System Instrument Operation Guide* (PN 4407430) and the *Applied Biosystems SOLiD™ 3 ICS System Software Help* (PN 4404524). These describe the ICS software and provide procedures for common tasks:

- Prepare for a run
  - Load a flowcell
  - Set the focus range
- Create runs in the Run Wizard
  - Create a sequencing run
  - Create a multiplexing run
  - Create a WFA run
- Manage runs
  - Create a sequencing run by import
  - Create a multiplexing run by import
  - Create a WFA run by import
  - Assign a previous run
  - Edit a run
  - Import and export
- Monitor runs
  - Adjust a run in progress
  - View cycle scans
  - View heat maps
  - View run logs
- System status
  - Prime the buffer line
  - Set the chiller state
  - Set the door state
  - Set the lamp hours
- Troubleshoot run results, the failed panels

SETS overview

The SOLiD™ Experiment Tracking System (SETS) software is a web-based application for viewing real-time data and completed run analysis reports from a SOLiD™ Analyzer. The SOLiD™ Analyzer system uses ligation-based sequencing, then filters data to remove missing calls, non-ligations, and/or redundancies. Refer to the *Applied Biosystems SOLiD™ 3 System - SETS Software Getting Started Guide*, PN 4389302, for more detailed information about SETS. The Getting Started Guide covers the following topics and provides procedures for the following tasks:

- Prepare run settings
  - Log in to SETS
  - Set analysis parameters
  - Load a reference sequence
- Monitor the run in SETS
  - View run metrics
  - View cycle scans
  - View sample data
  - Troubleshoot failed jobs
  - View history
  - Cancel analysis midstream
  - Cluster status
  - System logs
  - RSS feed
- View reports
  - View overall run reports
  - View sample reports
  - View analysis reports
  - Master Report tool
- Perform re-analysis
  - Primary re-analysis
  - Secondary re-analysis
- Support multiplexing
  - Multiplexing and barcodes
  - View multiplexing series in SETS
  - Setup a multiplexing run in ICS
  - Multiplexing series reports
- Manage administrative tasks
  - Global SETS and export
  - Auto-export in SETS
  - Manual export in SETS
  - Event notifications and e-mail services
  - Alerts
  - Delete runs
  - Manage users

Fundamentals of color space analysis

The Applied Biosystems next-generation SOLiD™ 3 System sequencing technology is based on sequential ligation of dye-labeled oligonucleotides. This technology makes possible massively parallel sequencing of clonally amplified DNA fragments. Features of this system, such as mate-paired analysis and 2-base encoding, enable studies of complex genomes by providing a greater degree of accuracy. This section describes the principles of 2-base encoding and the benefits of performing analysis in the di-base alphabet, which is also known as *color space*.

Introduction

Until recently, most DNA sequencing was performed using the chain termination method developed by Frederick Sanger. (Refer to the paper by Sanger F., Coulson A. R., 1975, A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase. *J Mol Biol.* 94(3): 441-448.) This type of sequencing is often referred to as *Sanger sequencing*. Sanger sequencing data is also encoded in color space by the four fluorescent dyes used in the sequencing chemistry and displayed as peaks in an electropherogram. The difference between Sanger sequencing and the SOLiD™ 3 System color space data is that, in Sanger sequencing, each color represents only a single nucleotide and is automatically translated to A, C, G, or T. With the SOLiD™ 3 System, each color now represents four potential two-base combinations. See Figure 4. The conversion into nucleotide base space is usually done after the sequence is aligned to a reference genome transcribed in color space. As an alternative, translation can occur following the generation of a consensus sequence.

Construct the 2-base color code

The SOLiD™ 3 System's 2-base color-coding scheme is shown in Figure 4.

[code]	0	1	2	3
[dye]	FAM	Cy3	TXR	Cy5
$(XY)_1$	AA	AC	AG	AT
$(XY)_2$	CC	CA	GA	TA
$(XY)_3$	GG	GT	CT	CG
$(XY)_4$	TT	TG	TC	GC

Figure 4 SOLiD™ 3 System's 2-base color-coding scheme.

The column under code *i* (0, 1, 2, or 3) lists the corresponding dye and the di-base probes (adjacent nucleotides) encoded by color *i*. For example, GT is labeled with Cy3 and coded as "1".

The following steps are used to encode a DNA sequence. Consider the example ATCAAGCCTC:

1. Start at the 5′ end.

2. Replace the di-base AT at this position with its corresponding code 3 from the table.
3. Advance by one base, which shows the TC di-base (code 2).
4. Continue to advance by one base, as shown below.

```
Base Sequence: A T C A A G C C T C
Color String:   3 2 1 0 2 3 0 2 2
```

This process encodes a *k*-mer of bases as a (*k*-1)-mer of colors. Although this color string codes for four different *k*-mers, knowledge of the type and position of any of its *k* bases helps to encode the sequence. For SOLiD™ sequencing applications, prepend the leading base A in the example above to result in a *k*-mer A321023022. From this, the base sequence can be reconstructed. The SOLiD™ 3 System generates its reads in this encoded form. One way of accomplishing this is shown in Figure 5.

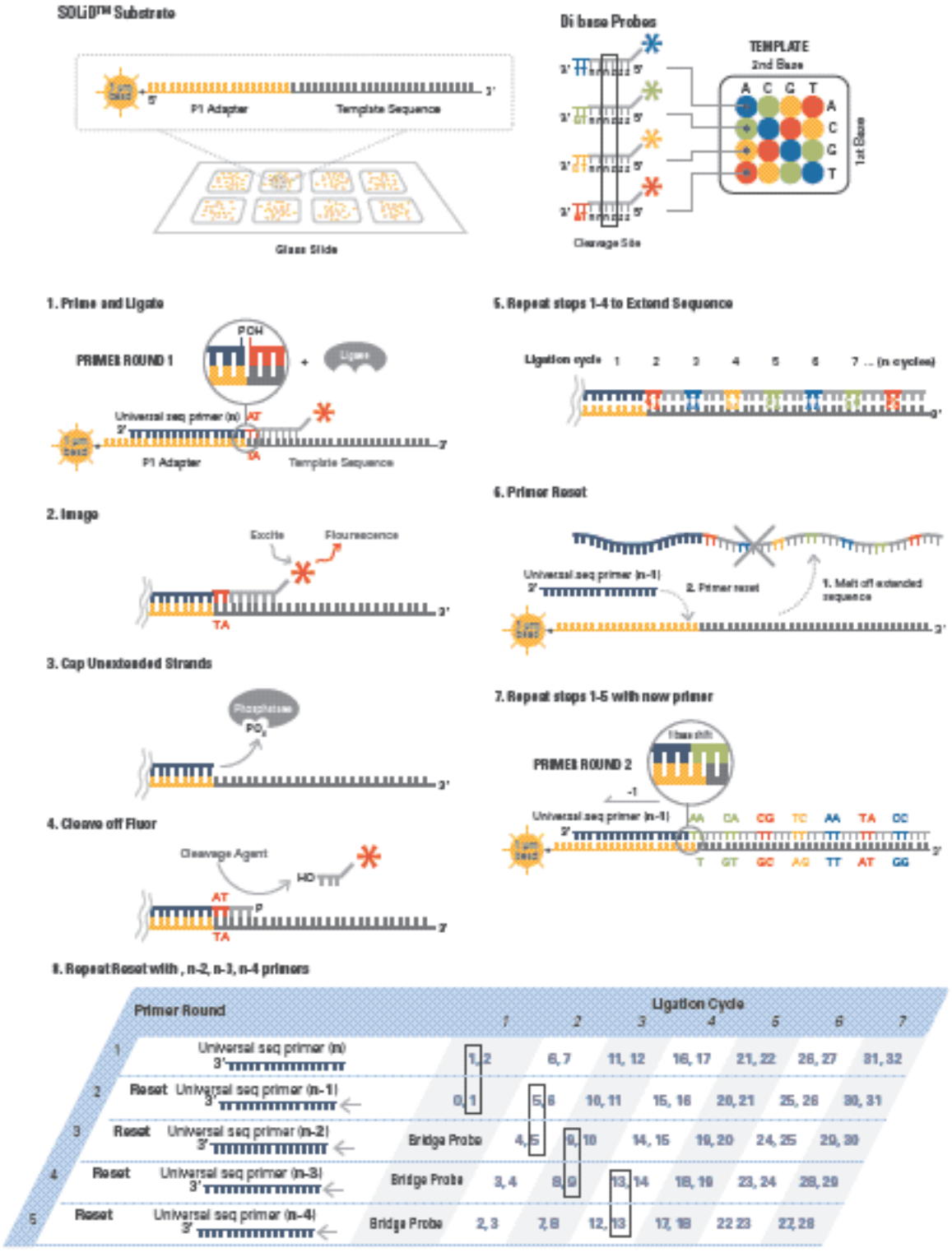


Figure 5 Sequence by ligation using di-base labeling probes on the SOLiD™ 3 System.

Requirements/properties for a two-base color code scheme

The 2-base color-coding scheme that is described above is essentially the only code that satisfies the properties in the following list. This conclusion can be observed by treating the properties as requirements and constructing the color code from them. This section addresses only bases, not other International Union of Biochemistry (IUB) codes. Let B = {A, C, G, T}.

The color code should satisfy the following requirements. For all bases b, d, e in B:

1. The available colors are 0, 1, 2, and 3.

   color (bd) $\in$ {0, 1, 2, 3}.

2. Two different di-base probes that have the same first base result in different colors.

   color (bd) $\neq$ color (be) if d $\neq$ e.

   For example, color (AC) $\neq$ color (AG).

3. A di-base probe and its opposite result in the same color.

   color (bd) = color (db).

   For example, color (AC) = color (CA).

4. Mono- and di-base probes result in the same color.

   color (bb) = color (dd).

   In other words, color (AA) = color (CC) = color (GG) = color (TT).

   The following are interesting properties that follow from the above four requirements. Property 5 follows from requirements 2 and 3 and makes constructing the color code easier.

5. Two different di-base probes that have the same second base result in different color codes: color (bd) $\neq$ color (cd), if b $\neq$ c.

   For example, color (AC) $\neq$ color (TC).

   Property 6 also follows from requirements 1-4, but it is most easily verified against the completed code (Figure 6, Panel E).

6. A di-base probe and its complement result in the same color.

   color (bcdc) = color (dcbc).

   For example, color (AC) = color (TG).

Satisfy the requirements for a two-base coding system

Figure 6 lists the colors for each di-base probe. For example, the value in row C and column T is the color 2 for di-base CT. Requirements 1 and 2 require that all colors are present in the first row. Because the system can use any one-to-one mapping between the actual dyes and the labels 0, 1, 2, and 3 (provided that requirements 1 and 2 are satisfied), the first row can be labeled as shown in Figure 6, Panel B. Requirement 3, that color (bd) = color (db), gives a unique labeling for column A (Figure 6, Panel C). Requirement 4, that color (bb) = color (AA), gives a unique labeling for the diagonal (Figure 6, Panel D). Finally, requirements 1, 2, and 5 state that every color must appear in every row and every column exactly once (Figure 6,

Panel E). The table (Figure 6, Panel E) is easy to memorize and work with because, by virtue of Property 3, di-base probes can be thought of as two-element sets for assigning colors. The di-base probes that start with A result in colors 0, 1, 2, and 3 respectively.

Therefore:

1. AA, CC, GG, TT all are assigned color 0.
2. AC and CA are assigned color 1, and so must GT and TG.
3. AG and GA are assigned color 2, and so must CT and TC.
4. AT and TA are assigned color 3, and so must CG and GC.

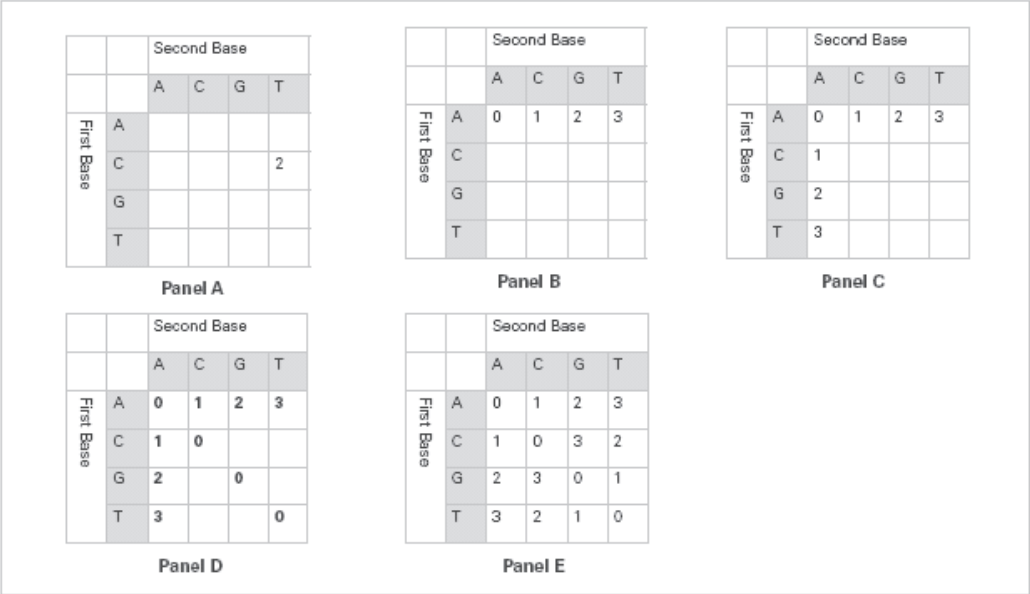


Figure 6 Requirements that assign color for a 2-base code.

Principles of ligation-based chemistry and two-base encoding

The SOLiD™ 3 System's sequencing technology is based on sequential ligation of dye-labeled oligonucleotide probes. Each probe assays two base positions at a time (Figure 5). The system uses four fluorescent dyes to encode for the sixteen possible 2-base combinations. Multiple ligation cycles of probe hybridization, ligation imaging, and analysis are performed to extend the strand from a primer hybridized to a ligated adaptor by the immobilized bead (P1 adaptor). The resulting product is then removed and the process repeated for 5 more rounds with primers hybridized to positions *n*-1, *n*-2, etc., in the P1 adaptor.

There are several fundamental properties unique to ligation-based sequencing. These properties contribute to the high accuracy inherent in the SOLiD™ 3 System. The advantages of these properties and their contribution to data quality are:

1. Two bases are interrogated in each ligation reaction, increasing specificity.

2. The primer is periodically reset for five or more independent rounds of extension reactions, improving the signal-to-noise ratio of the system.
3. Each base is interrogated twice in two independent primer rounds, increasing confidence in each call.
4. Four dyes are used to encode for sixteen possible 2-base combinations. The design of the encoding matrix enables built-in error-checking capability.

Color space and base space as applied to the SOLiD™ 3 System

The final files generated by the SOLiD™ 3 System are in base space. These are the gff, consensus, and SNP files. To maximize the built-in error correction of two-base encoding, all file analysis is conducted in color space prior to final results.

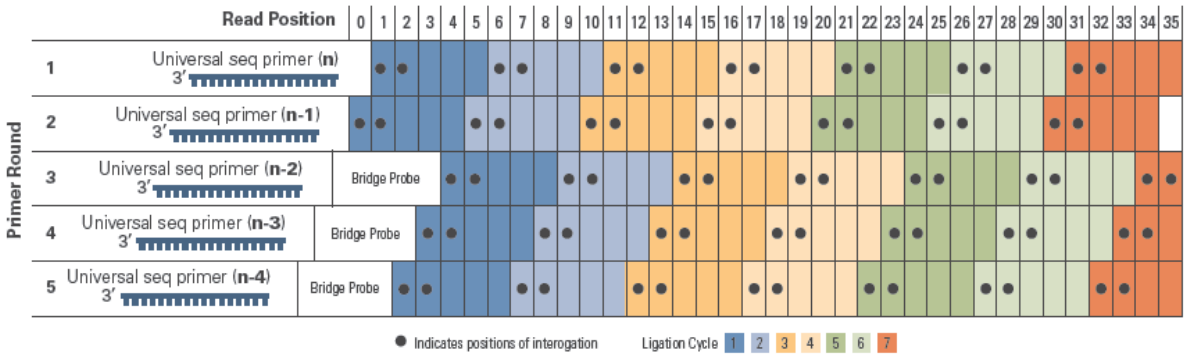
Color space data

Rather than reading one base per cycle, the software measures information on two bases simultaneously. In each cycle, the software calls one of four colors (color space call). Because each ligation measurement event measures two bases, all bases (except for the final base of a read) are interrogated twice, providing for an additional level of error correction.

Relationship between cycle and base position

The graph below shows the relationship between color space, base position, and the sequencing chemistry. *Base number* refers to base position. Base 0 is the last base of the adapter and is not part of the target sequence. The five primer lines show the order in which the data was generated (0-indexed).

Note: In all files, only processed data refers to a color space position.



Color space formats

Color space data are presented in three slightly different formats. In two of the formats, a base (A, C, G, or T) is appended to the color space calls.

Note: Color space data are self-complementary: In some situations, when you might expect to see complemented data (for example, reverse), the data appear the same. For example, AC = 1, TG = 1.

The different types of color space data are:

- **Processed color space data**: consists of a numeric string prefixed (suffixed if reversed) by a single base. The base that precedes the numeric (color code) data is the first base of the actual sequence (in base space, not color space).
- **Unprocessed color space data**: consists of a numeric string prefixed by a single base. This base is the final base of the sequencing adapter and is not part of the target sequence. It is included to disambiguate the first color call.

Complementing color space data

Color space data are self-complementary as shown in the following matrix:

		2nd Nucleotide			
		A	C	G	T
1st Nucleotide	A	0	1	2	3
	C	1	0	3	2
	G	2	3	0	1
	T	3	2	1	0

Sequence:

Base	A	G	C	T	C	G	T	C	G	T	G	C	A	G
Color Space	2	3	2	2	3	1	2	3	1	1	3	1	2	

Complemented:

Base	T	C	G	A	G	C	A	G	C	A	C	G	T	C
Color Space	2	3	2	2	3	1	2	3	1	1	3	1	2	

Two-base encoding and error recognition

The error-checking abilities of the two-base encoding schemes have *not* been used to correct any of the data provided in these files.

Example:

Reference = 2 3 2 2 **3** 1 2 3 1 1 3 1 2

Observed = 2 3 2 2 **0** 1 2 3 1 1 3 1 2

Note: Notice the single color space error.

In this example, a single color space error occurs. The most likely explanation for the observed 0 is that it is a measurement error. Because a single color space change is not allowed, a change to one of the adjacent bases is needed for a real SNP. This correction requires multiple measurement errors, leaving the most likely explanation that the 0 is a 3. The fact that the two surrounding bases are the same as the reference is further evidence that correcting the 0 to a 3 is acceptable. Any single color error can likely be corrected, especially when using multiple aligned reads.

Processing

The flow of data through the SAT pipeline is shown in Figure 7.

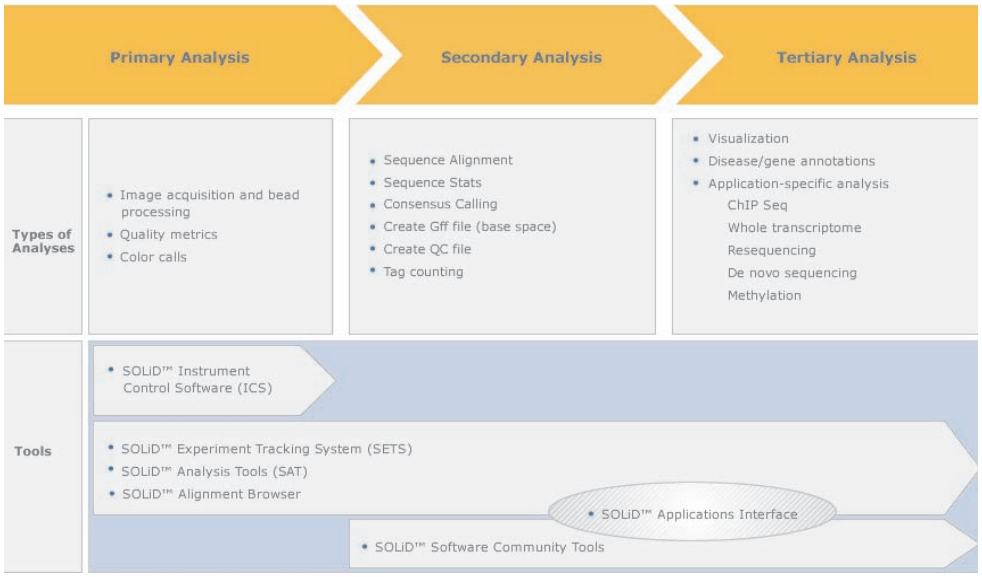


Figure 7 Data flow through the SAT pipeline.

Typically, a panel is imaged four times per cycle (once per channel). The color space calls are carried out on a by-cycle basis. Each data point represents an individual bead with four intensity values. Beads are assigned a color space call using a clustering algorithm. As part of the clustering process, data are scaled and baselined, quality values are assigned, and color call is assigned.

Results of primary analysis for each panel are stored in files with the .spch extension (Solid Panel Cache HDF5). The .spch file is used as a cache; that is, primary analysis both writes results to the file and reads results from the file as analysis proceeds on a cycle-by-cycle basis. Data within the spch file is organized by cycle. When all cycles have been analyzed, a final processing step is run to convert the data from the cycle-based format to the "read-based" ASCII formats (.csfasta and .qual).

The spch file is in the HDF5 format. Details on the format are available at http://hdf.ncsa.uiuc.edu/products/hdf5/index.html.

Tools to view the contents and extract results are available at http://hdf.ncsa.uiuc.edu/hdf-java-html/hdfview/index.html.

Prior to secondary analysis, results are filtered by removing all tags with missing data. Filtering removes all incomplete beads and reads with missing calls. The filtered data are then processed to generate the file *xxxx.csfsta*. As part of this process, a known base (from the adapter sequence) is prepended to the color space data, allowing disambiguation of the first color call. Each read is treated separately, even if it is part of a mate-paired tag. The file contains the color space read for each tag, and the color space data are used as the basis of all subsequent results. The data are then aligned in color space to the color space reference sequence. During analysis and alignment:

- Color space reference sequences are derived by converting the base-space reference sequence to color space (this process occurs internally; you supply the reference sequence in base space).
- All alignment is done in color space.
- All the alignments are with individual tags; the mate-paired information is not used in the initial alignments.
- After the single reads are analyzed, a further round of analysis is performed in which the mate-pair information is used.
- The paired tags are analyzed in a two-step process:

    1. The tags where both the forward and reverse tags passed (both tags were matched to a reference sequence) are analyzed.
    2. Those mate pairs where only one of the tags was matched to the reference sequence (in color space) are re-analyzed, and the non-matching tag is compared to the reference sequence.

  Because analysis can be constrained by the knowledge of the allowed distances, this process allows alignment with more disagreements and the use of a more cpu-intensive alignment algorithm.

Note: Where possible, all data are presented in a fasta-compatible format to facilitate use in a variety of applications. Final output files are in base space and relative to the submitted reference (for example, consensus file, variation file).

Data analysis considerations

Understand Ns and color space

In the SOLiD™ 3 System, unknown color space calls are represented as '.' or *N*. By default, the system filters out bases with a single '.' in the sequence and leaves only reads with color space calls present at all positions. You can change this behavior by masking a specific position of a read. For example, if position 21 of a 35 bp run is not good, you can mask this position to allow for missing color space calls at position 21.

Find Single Nucleotide Polymorphisms (SNPs)

SNPs are single base-pair mutations that usually consist of two alleles. Color space rules allow the software to detect only valid SNP sites. According to the color space rules, one isolated color change is always an error when you consider single nucleotide changes. Two adjacent color changes can be either an error or a valid SNP. To detect two adjacent SNPs, look for three adjacent color changes. There are very specific rules to determine which color changes are valid and which are not.

Color space rules for SNP detection

In the simplest case, where you have a single base change, the following rules apply (Figure 8):

1. For any given reference, there are only three valid double-color changes.
2. For the other 12 possibilities, the 6 single-color and 6 two-color changes are invalid, since they would require multiple changes within that genomic region. For these complicated genomic changes, the software uses a more sophisticated algorithm.

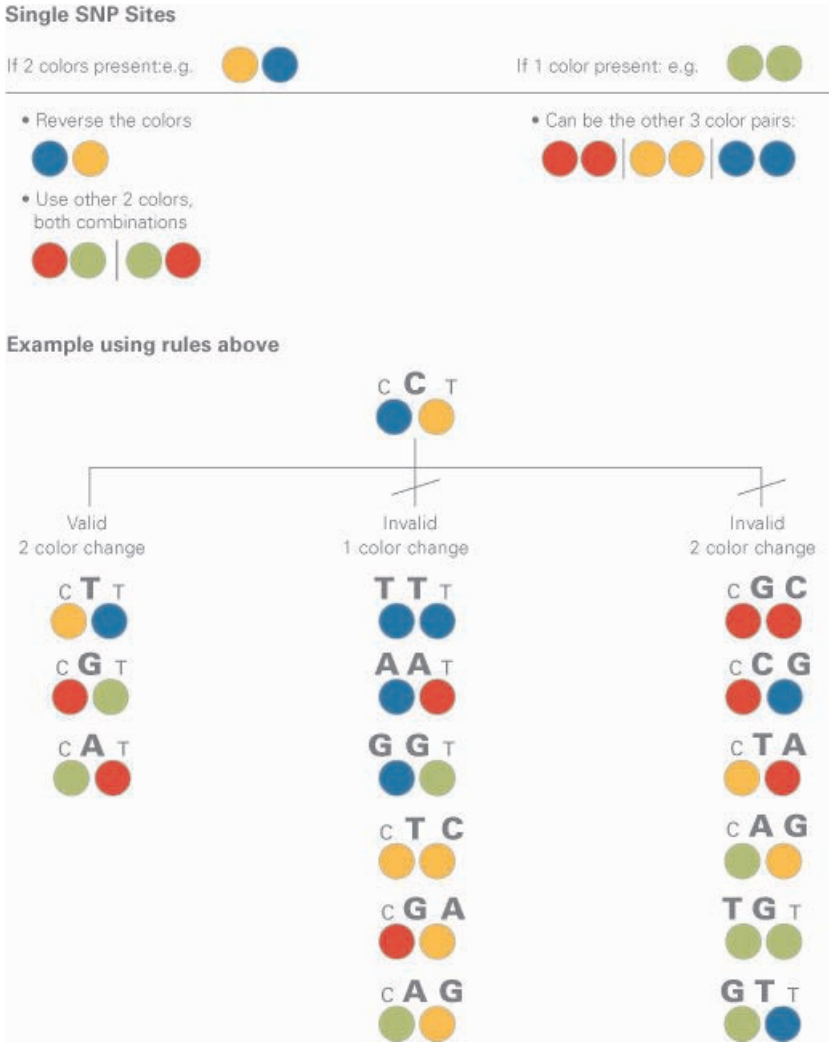


Figure 8 Color space rules for Single Nucleotide Polymorphism (SNP) sites.

Color space rules are automatically applied during data analysis using the SOLiD™ 3 System's software analysis tools. You do not need to apply these filters manually.

SNPs and errors

Because a SNP generates two adjacent and valid color space mismatches to the reference sequence, the software allows a minimum of two color space mismatches during alignment to the reference sequence. Depending on the complexity of the alignment, the optimum number of mismatches in alignment is at least three. If only two errors are allowed, then whenever there is a SNP (two mismatches) and a measurement error, the third error causes omission of that read. You can override the omission by using an alignment program setting that counts two adjacent mismatches as a single mismatch. Therefore, two adjacent mismatches and a measurement error are classified as two errors, whereas three non-adjacent errors are classified as three errors.

Higher accuracy for SNP detection

Two-base encoding provides higher system accuracy and built-in error checking because it enables discrimination between measurement errors and true sequence polymorphisms. The SOLiD™ software interrogates each base twice in two independent reactions. Information about each base is included in two adjacent pieces of color space data. Because the system uses four fluorescent dyes, there are 16 possible two-color combinations (Figure 9).

	Blue	Green	Yellow	Red
Blue	BB	BG	BG	BR
Green	GB	GG	GY	GR
Yellow	YB	YG	YY	YR
Red	RB	RG	RY	RR

Figure 9 Color space transitions.

A SNP must have two adjacent color changes. Single-color mismatches are not evidence of a SNP. This feature allows SOLiD™ to distinguish errors in measurement from true SNPs.

SNP error rates

When you use two-base encoding, any SNP in the original sequence is represented as two adjacent mismatches in color space. Only three of nine possible adjacent mismatches can correspond to a real SNP. Suppose the raw sequencing error rate for a species is 1% per base, and for the same species that is sequenced, the SNP rate is about 0.1%. For a sequencing project of M total bases, there are about $0.001M$ real SNP occurrences in the data set. Among these SNPs, $0.001M \times 0.02$ total cases appear as single mismatches or two invalid mismatches because a sequencing error happens at one of the alleles of the SNP. Only 2% of the real SNPs fail to appear as two adjacent, valid mismatches. If only two adjacent, valid mismatches are treated as candidates for SNP detection, then there is a 2% false negative rate. For any data set, two adjacent and valid mismatches can be caused by sequencing errors. However, for a total of M bps sequenced, there are $0.00003M$ total occurrences of two adjacent mismatches. Note that there are about $0.001M$ adjacent valid mismatches from real SNPs, among all two adjacent, valid mismatches observed in a particular data set. Of these, 97% are from real SNPs, and only 3% may be from sequencing errors. This is a 97% true discovery rate for that particular data set. Because there are about $0.01M$ total sequencing errors in the data set caused by two-base encoding and the software's ability to remove all single base mismatches, the error is reduced from $0.01M$ to $0.00003M$. This is a reduction of 300 times, making the effective error rate 0.003%. This calculation illustrates the power of two-base encoding in resequencing and finding SNPs.

Sampling

If coverage at a genome position is low, the second allele might often not be sampled, even if it is present. Heterozygotes are expected to be underrepresented at low coverage. (They can be called as a homozygote for one of the two alleles.) It is important to have a low false positive rate for SNP detection, because SNPs are expected to exist at only 1 in 1,000 positions for humans and some other species. The false positive rate in an individual species should be at least an order of magnitude lower than this, to avoid a high false discovery rate. An error model that incorporates qualityvalues (QV) can increase the overall accuracy of SNP detection. The SNP detection algorithms of the SOLiD™ 3 System use explicit error models generated from each run and biologically meaningful prior probabilities to evaluate evidence of a SNP. Two-base encoding provides a built-in way to distinguish errors from true alleles to manage the false positive rate.

Allele ratio

If the sample preparation method or some other cause produces results in allele ratios that are considerably different from the expected 50:50 ratio, it is more difficult to detect heterozygosity. Detecting heterozygosity then requires more sequence coverage.

Find polymorphisms in color space

The SOLiD™ 3 System has the capability to detect complicated genomic variations such as adjacent SNPs, insertions, deletions, and structural rearrangements. For these genomic variations, you can download SOLiD™ bioinformatics tools from the website http://solidsoftwaretools.com. An example of various polymorphisms in color space is shown in Figure 10.

Figure 10 Examples of polymorphisms in color space.

The Job Manager 2

Introduction

The Job Manager is a small stand-alone daemon process that picks up jobs from the SOLiD™ database and submits them in the proper order, with the proper parameters, to the proper analysis components.

The primary responsibility of Job Manager is to ensure that all of the analysis pipeline is executed in order. A dependent task is executed only after its prerequisite tasks are completed. See Figure 1. One example that you can see in it is that colorcall jobs are executed after Focalmap jobs are completed.

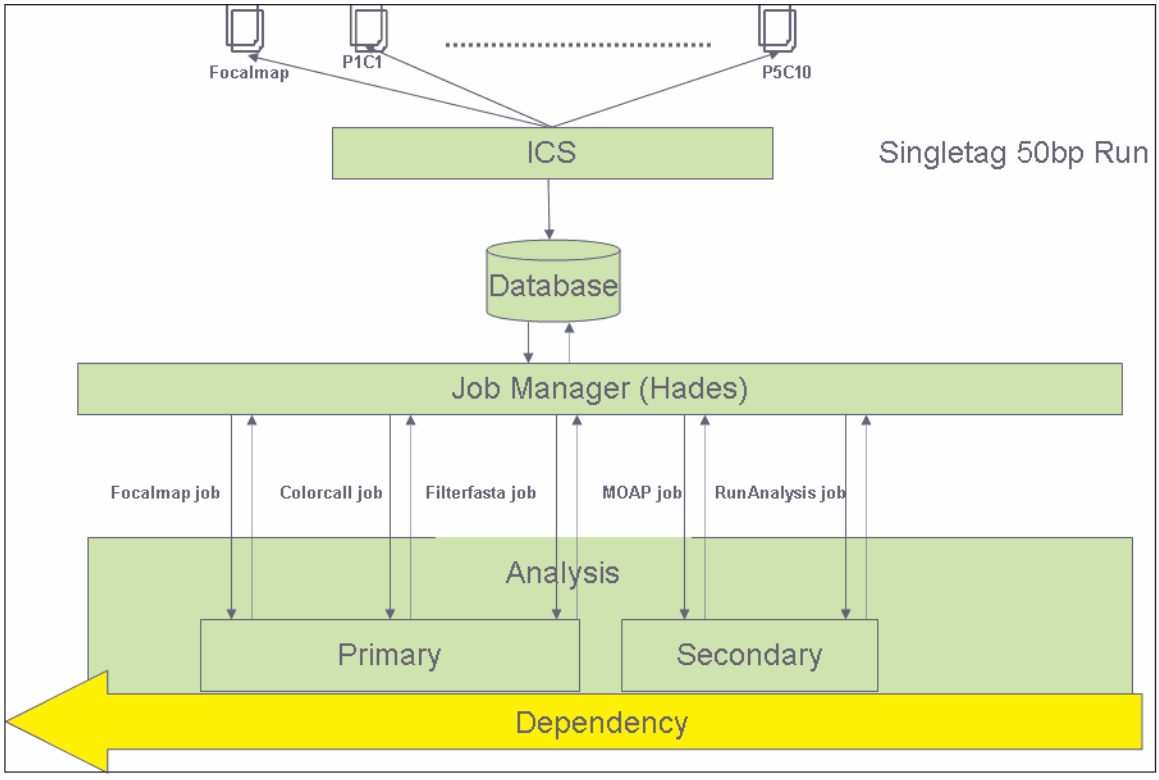


Figure 1 Job Manager workflow in the SOLiD™ 3 System.

Job Manager components

Hades

Hades is the entry point of Job Manager. Hades is launched as a daemon, so the init() and start() methods from org.apache.commons.daemon.Daemon are called instead. There is still a main() method.

Hades performs the following tasks:

1. Loads all the ControlledValues for Job Status and Job Stage into a centrally accessible Map, accessible using getControlledValue(String value).
2. Parses optional command line arguments. Legal values are:
   - enderPort - set the port (default = 8765)
   - charonBundle - read Charon properties from a different file
   - propertyFile - read all arguments from a file
   - dbeCounter - Terminate if a given number of database exceptions occur in a given time. The default value is 10,0. A value of 10,1 means that if 10 database exceptions occur within 1 minute, then terminate Hades. If the value is 0, Hades does not terminate on database exceptions.
   - verbose - log level .default value is false.
   - estimateRequiredDiskSpace=true - Hades defaults to doing disk space estimation and stops submitting jobs if there is insufficient space. The user can turn it off by setting this parameter to “false” and then restarting Hades.
   - pluginDirectory=/share/apps/corona/lib/java/hades/plugins - This is the directory to deploy Hades plug-ins. A user can change the default value to any arbitrary directory. You need to restart Hades after you change this value.
   - rsyncArgs=-arRzO -e ssh --timeout 360 - This command is for changing rsync arguments. To export large runs over a busy network, this timeout value might need to be increased.
   - sleepyTime.heartbeat - The default value is 30 seconds. Hades writes heartbeat.xml file after every specified second limit. SETS reads the heartbeat file to determine if Hades is present.
3. Creates a JMX Server.
4. Creates a Charon and launch it in its own thread.
5. Creates an Ender and launch it in its own thread.
6. Hades could be a central place for configuration data, but currently it is not.

Other than implementing various HadesMBean calls, which allow a user or developer to do some simple monitoring via a JMX web page interface, Hades essentially sits and waits for calls to stop( ). It provides one static method, handleDbException(). Its purpose is for other classes to call it when a DbException occurs. At present, after 10 DbExceptions, Hades terminates.

Charon

Charon is the main loop of the daemon. It polls the database periodically for open JobWorkflows. For each workflow, it looks for jobs that are ready to be run. (This complex logic of whether a job is ready to run was formerly in Charon in V2, but it has now moved into JobWorkflow.) If this job has not already been submitted by Charon, the following actions happen:

1. The job and its parameters are "fully loaded" from the database.
2. An AnalysisRunner is created for the job and executed.
3. When the AnalysisRunner completes, Charon's completed( ) callback is called. If there was an exception or non-zero status, the job's status is set to ANALYSIS_JOB_STATUS_FAILED. Otherwise, the job's status is set to ANALYSIS_JOB_STATUS_STARTED, unless the status is changed in the configuration file. The job also has it dateStarted property set.

The jobs are submitted based on the dependency graph shown in Figure 2.

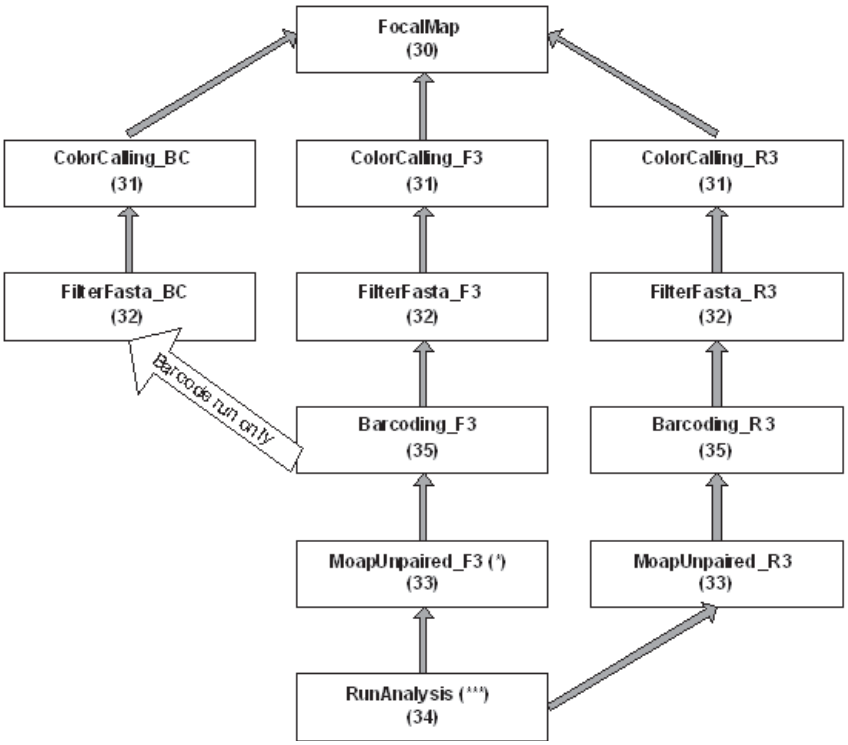


Figure 2 Charon dependency graph.

Charon is configured from a properties file (RealCharon.properties). By default, the properties file is read as a Bundle. This file must contain one entry per stage, which gives the name of the component script to be run at that stage. Each Pipeline in RealCharon.properties must be inherited from IPipeline. For example, the current file looks like:

```
#This is the real Charon command line file
postFocalMapPrimary=Class:com.apldbio.aga.analysis.primary.core.Focal
MapPrimaryStage
postScanSlidePrimary=Class:com.apldbio.aga.analysis.primary.core.Scan
SlidePrimaryStage
```

```
postPrimerSetPrimary=Class:com.apldbio.aga.analysis.primary.core.Prim
erSetPrimaryStage
postBarcodePrimary=Class:com.apldbio.aga.analysis.primary.core.Barcod
ePrimaryStage
postPrimerSetSecondary=Class:com.apldbio.aga.analysis.secondary.pipel
ine.JMOAP
postRunSecondary=Class:com.apldbio.aga.analysis.secondary.pipeline.JM
OAP
qcrpGenerate=Class:com.apldbio.aga.analysis.primary.QCTaskJobSubmitte
r
```

In addition, this file may contain additional configuration information.

By default, Charon logs any unexpected RuntimeException but continues its loop. If you do not want Charon to continue the loop, add this line:

```
keepRunningOnRTE=false
```

You can change the name of the update solid script (default = updateSOLiD.py) by adding this line:

```
update.solid.script=newname
```

Table 1 gives a summary of stage names and their functions.

Table 1 Stage names and functions

Stage	Also known as	Comments
focalmap	postFocalMapPrimary	Finds the position of all beads
colorcall	postScanslidePrimary	Finds the beads that are active during ligation cycle
Filter_fasta	postprimersetPrimary	Combines the reads from all panels
Barcoding	postBarcodePrimary	Breaks the reads into libraries. One per tag (F3, R3)
Moap_unpaired	postPrimerSetSecondary	Runs all SAT pipelines
runAnalysis	postRunSecondary	Consolidates the results. Runs pairing in a MatePair run.

Figure 3 illustrates the Domain Model.

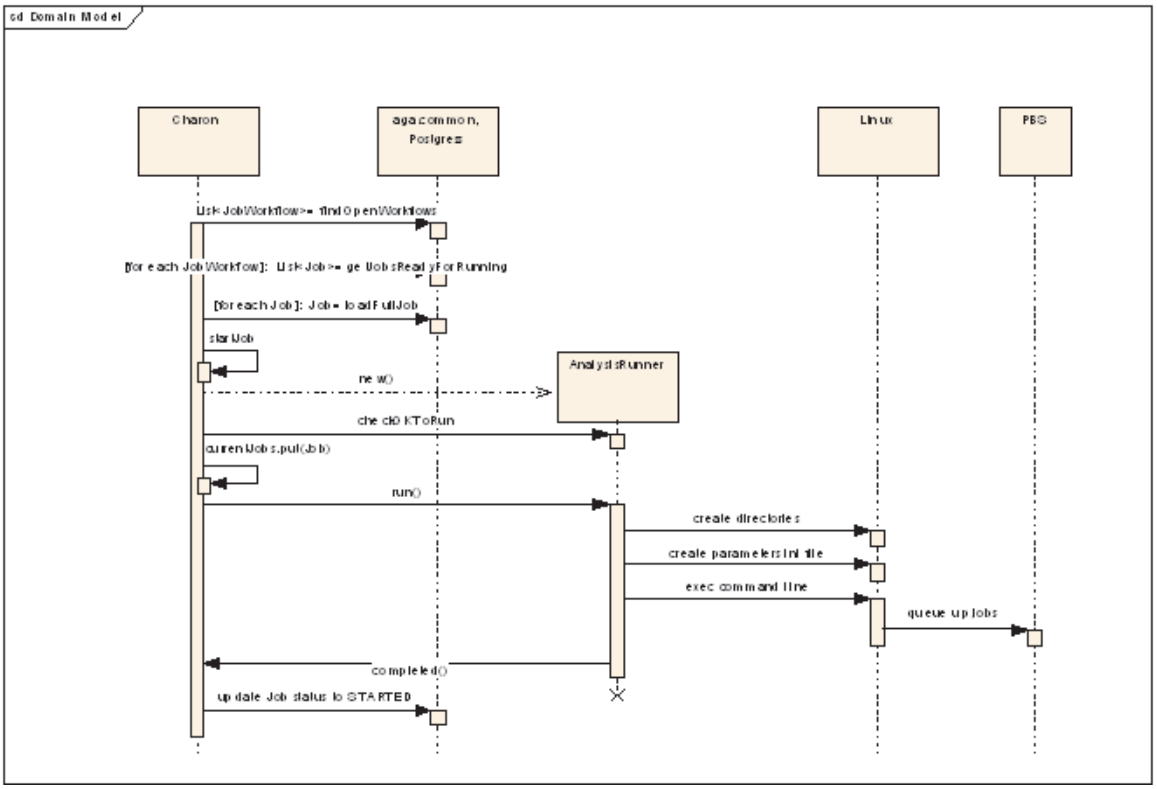


Figure 3 Domain Model.

AnalysisRunner

AnalysisRunner creates directories, parameters, and a Process for a job, then runs the Process reading output from stdout and stderr (ignoring it, to keep processes running and avoid buffer overflows). On completion, AnalysisRunner calls back to the Charon.completed() method. It is passed a bundle for configuration, although usually this is Charon's bundle. AnalysisRunner performs the following tasks:

1. The WorkingFolder for the job is created, and set as the working directory of the process. The path to the working folder is:

   /data/results/RUN_NAME/SampleX/jobs/stageName.jobID

2. A parameters.ini file is created within the working folder. This is a list of key-value pairs.

3. Several other run and results directories are created.
   - FileLayout.getSampleDir()
   - FileLayout.getSampleDir (for results)
   - FileLayout.getReadsLoc()
   - FileLayout.getSecondaryDir() only for secondary analysis jobs
   - For MOAP stage 34 some special matching directories are created/passed.

4. Hades creates an instance of Pipeline based on entry in the RealCharon.properties. After it creates an instance of IPipeline, it calls setParam method with a Map argument. It then calls run method of IPipeline.

Ender

Ender listens on a port (default 8765) for new connections. When one is received, it starts up an EnderSocketReader for that connection. When the connection is closed, completed() is called and that socket is closed. See Figure 4.

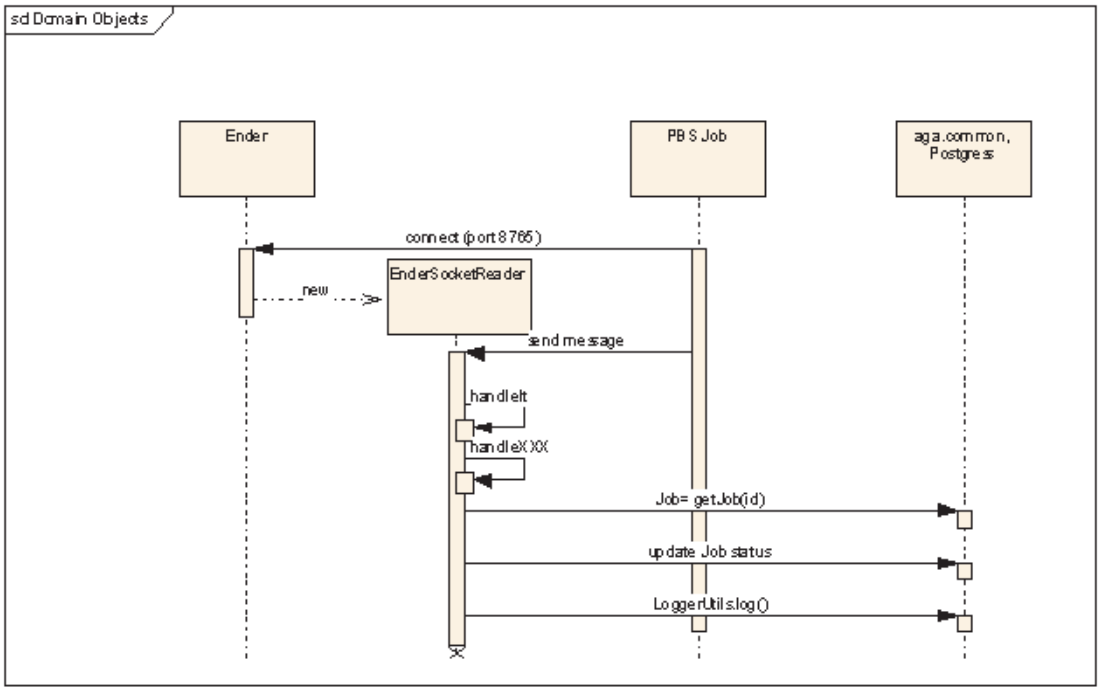


Figure 4 Ender workflow.

EnderSocket-Reader

The superclass StreamReader reads lines from a Stream and calls a do-nothing stub, handleSpecialLines(). EnderSocketReader implements handleSpecialLines(). If the line begins with "[", it tests for a message that should be logged. The format is:

```
[LEVEL] JOB_ID optionalMessage
```

The text inside the "[…]" is compared with standard logging levels "DEBUG", "INFO", "WARNING", "ERROR", "FATAL", and also against two special messages, "PROGRESS" and "COMPLETE". If the text is a logging level, the optionalMessage is logged to the database using LoggerUtils.log(). If the text is FATAL, PROGRESS or COMPLETE, the job's status is changed to the appropriate IModelConstants for failed, in_progress, or finished. For FATAL or COMPLETE, the job's DateCompleted is set.

Troubleshooting

Hades service

Hades service can be started and stopped by entering the following lines in commandline.

IMPORTANT! DO NOT STOP HADES WHILE ANALYSIS IS RUNNING.

```
Start : /etc/init.d/hadesd start
Stop: /etc/init.d/hadesd stop
```

Log files

Hades : /var/log/hades

moap: /data/results/[instrumentName]/[runName]/[sampleName]/jobs/postPrimerSetSecondary[id]

Find problems

Go to /data/results/[instrumentName]/[runName]/[sampleName]/jobs/[directoryWithId]

```
Execute  > grep -ri  '[Error | Exception | Fatal ]'
```

Check nodes

```
pbsnodes
qstat -q
qstatrunning
beostatus -c
showq
```

Lost panels

1. Check var/spool/mail/pipeline.
2. Check opt/torque/server_logs.

JMX view

To view the internal state of Hades, do the following:

1. Go to http://10.1.1.1:8082/.
2. Use corona/c0r0na.

Debug job workflow

```
-jPrintWorkflow.sh [--host=hostname] [--flags=int] [-all] [-brief]
id1 id2 ... idn
```

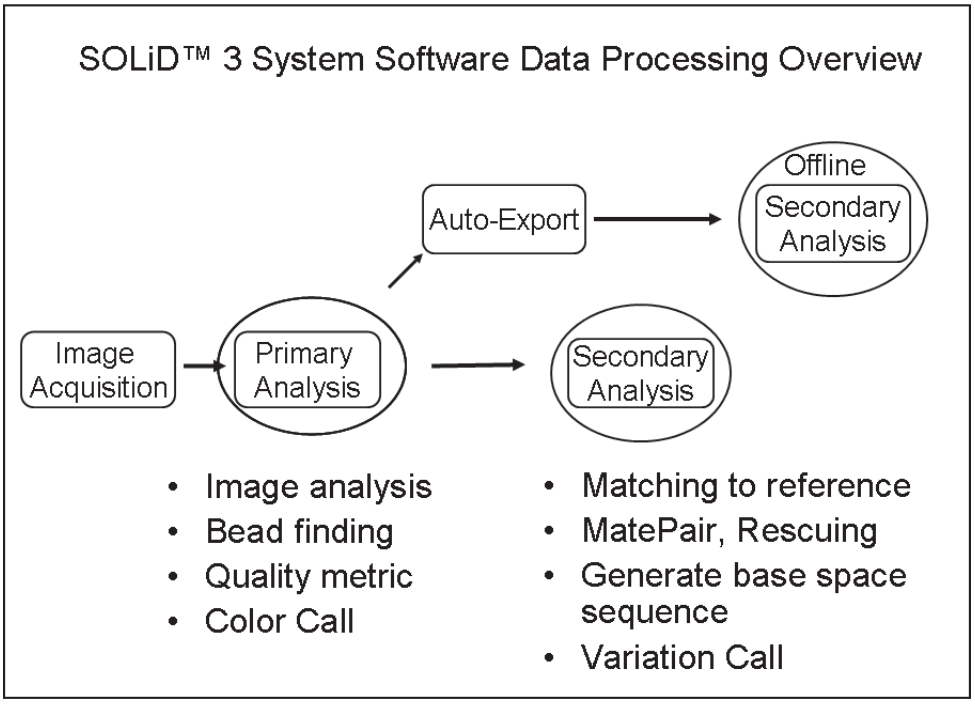
SAT Pipelines 3

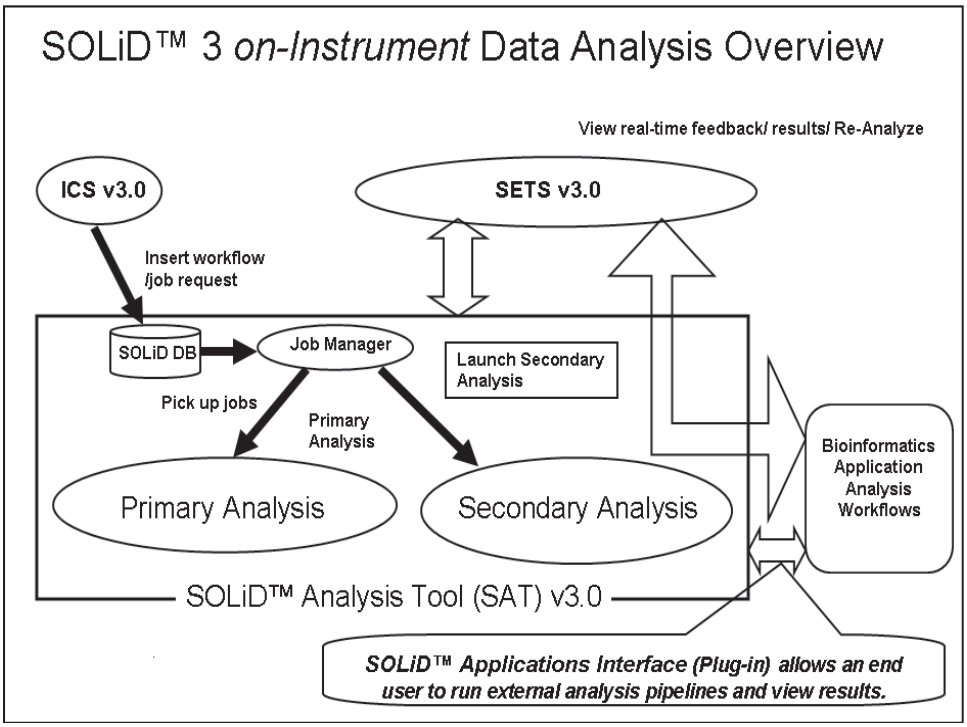
Definition of a pipeline

A *pipeline* is a framework that organizes data analysis tasks logically and functionally into sub-groups. This framework allows upstream dependencies to be assigned and validated. The current pipelines all implement a SOLiD™ pipeline interface for the Java programming language. This feature permits both coarse- and fine-grained parallelization. However, any analysis code that deploys a command-line interface can be used instead.

SOLiD™ pipeline workflow overview

The SOLiD™ instrument system contains a Windows XP computer that is used to set up runs and acquire images. The Linux computer cluster is used for data analysis, job handling, and SETS.





ICS (Instrument Control Software) initiates the experiment. It inserts workflow analysis jobs identified by users into the SOLiD™ database on the Linux headnode. As the run cycles progress, ICS acquire images and writes image files onto the Linux cluster. The Job Manager queries the SOLiD™ database to pick up open workflow jobs and to gather analysis parameters. These are used to prepare job submission to the PBS compute farm. The analysis workflow consists of two parts: primary and secondary analysis. The primary analysis focuses on color calls (similar to traditional basecalls) and generating quality values. The secondary analysis focuses on aligning reads to reference genomes and additional workflow pipelines (selectable by users through the SETS interface).

SOLiD™ database

The SOLiD™ database is a postgresql relational database. It keeps run information, sample information, analysis job workflow, and parameters associated with all the jobs. The analysis workflow can be re-created if necessary.

Primary analysis

The first step is bead registration to establish a focalmap. The focalmap serves as a reference image for aligning images from all subsequent ligation cycles for bead location and identification. The focalmap is generated from .tiff images acquired before any ligation cycle using anti-p2 hybridization. This action ensures that the beads with p2 primer are registered.

The results of primary analysis for each panel are stored in files with the .spch extension (Solid Panel Cache HDF5). The spch file is used as a cache; that is, primary analysis both writes results to the file and reads results from the file as analysis proceeds on a cycle-by-cycle basis. Data within the spch file is organized by cycle. When all cycles have been analyzed, a final processing step is run to convert the data from the cycle based format to the “read-based” ASCII formats (.csfasta and .qual).

SPCH files use the HDF5 format, which is a hierarchical binary format. Details on this format are available at http://hdf.ncsa.uiuc.edu/products/hdf5/index.html.

Tools to view the contents and extract results are available at http://hdf.ncsa.uiuc.edu/hdf-java-html/hdfview/index.html.

Chapter 5 gives a complete discussion of primary analysis.

Secondary analysis

Secondary analysis refers to application-specific analysis. It consists of a set of serial steps of modular workflow, called the *SAT analysis pipeline*. A custom analysis pipeline can be integrated with SAT so that it can be automated and can use SAT for unified job management. Refer to Chapter 4, *SOLiD™ Applications Interface*. The native SAT pipeline that comes with the software is a resequencing and variation analysis pipeline. It uses the following steps:

1. Filter reads.
2. Map or align reads to a reference genome.
3. Perform standard analysis (error profiles, coverage, .gff files, sequencing correlation, and similar tasks).
4. If the run is a mate-pair run, an additional mate-pair rescue is performed (optional).
5. Perform variation detection (SNP/DNA variation consensus calling).

You can specify additional analysis pipeline steps by the use of the SETS interface, including coverage analysis, sequencing bias analysis, sequencing chemistry correlation analysis, and similar tasks. In a manual run (command-line environment), you can do this by use of the parameter file.

Chapter 6 gives a complete discussion of secondary analysis.

Tertiary analysis

Tertiary analysis refers to analysis steps that generate biological interpretation. Examples are SNP discovery and validation analysis, structural variation analysis, ChIP-Seq analysis, and Whole Transcriptome analysis. The SOLiD™ analysis software includes SNP analysis package as part of the installation.

For more bioinformatics tools for Tertiary analysis, refer to the website http://www.solidsoftwaretools.com.

SOLiD™ Alignment Browser

The SOLiD™ Alignment Browser (SAB) is a tool modified from the Apollo genome browser to support color space reads for viewing alignment results. SAB v2.0 works with GFF v0.2 input files, which contains both base space (DNA space) and color space sequences. It is supported on Linux, Windows and Mac OS. SAB source code and executable files can be downloaded from the SOLiD™ Software Developer community, from the website http://www.solidsoftwaretools.com.

Limitation of SOLiD™ analysis software on the SOLiD™ instrument

The SOLiD™ 3 software is capable of analyzing genomes of arbitrarily large size both on and off the instrument. However, if the instrument is used constantly for sequencing, Applied Biosystems recommends that you complete the post-primary analysis steps off the instrument, on an off-line computer cluster. Use Global SETS Version 3.0 for this off-line processing. Refer to Chapter 7, *Off-line Data Analysis*.

Major pipelines

The following table lists the major pipelines and gives brief definition of each. Note that the Matching pipeline group consists of the following four sub-pipelines, given in Table 1.

- RepeatClassifier
- MatchingRandom
- MatchingRepeat
- MatchingConsolidate

Table 1 Pipeline definitions

Filtering	The Filtering pipeline performs QV filtering, in which any read is removed from further pipeline processing if it has more than a specified number of calls (“Max Failures”) that are less than a specified quality value (“QV Cutoff”).
RepeatClassifier	The RepeatClassifier pipeline, part of the Matching Pipeline Group, is new for Version 3. This is the first of four pipelines in the group that, given a set of reads and a reference sequence, acts as a predictive classifier. It separates the reads into two groups: Repeats and Random. “Repeats” are predicted to be either real repeats or to seed a high number of times during genome alignment. “Random” is the set of reads that are predicted to be randomly distributed or unique. This pipeline allows for the two subsequent matching pipelines to use different parameters, based on the types of reads expected to be processed. If this pipeline is bypassed, all reads are classified as random. (This was the behavior in Version 2.0.) This pipeline is useful for eukaryotic resequencing applications.
MatchingRandom	The MatchingRandom pipeline, part of the Matching Pipeline Group, is new for Version 3. It is an instance of the Matching pipeline that processes the reads classified as “random”. The Matching pipeline is used to search a set of reads against a reference sequence. When no classification is performed, this is the default matching pipeline used (it performs the matching for all reads). To get Version 2.0 or prior behavior in the pipeline, the Classifier is bypassed and only this pipeline is selected, with values of length equal to the read length and an upper bound for the number of mismatches.
MatchingRepeat	The MatchingRepeat pipeline, part of the Matching Pipeline Group, is new for Version 3. It is an instance of the Matching pipeline that processes the reads classified as “repeats”. The Matching pipeline is used to search a set of reads against a reference sequence. This pipeline should be executed only when the Classifier pipeline is activated.
MatchingConsolidate	The MatchingConsolidate pipeline, part of the Matching Pipeline Group, is new for Version 3. This final pipeline in the matching pipeline group is responsible for pulling together the results of the MatchingRandom and MatchingRepeat pipelines into a single results file for use by downstream pipelines. This pipeline needs to be run any time that matching is run. It provides the file system clean-up that is necessary to ensure that the all interface contracts for downstream pipeline dependencies have been satisfied. This pipeline does no new computation.

TagOutput	The TagOutput pipeline runs a series of QC and summary reports. This pipeline also creates the GFF V.2 files, which are used by other downstream pipelines. In a MatePair run, this pipeline is implicitly run for each tag by the PairedTag pipeline. The options for this pipeline are primarily used to describe how to create the GFF file.
MatePair	The MatePair pipeline produces the reports of a single tag run. This pipeline also creates reports that pertain to mate pair quality, as well as performing mate pair rescue (an additional matching process using the information from the library preparation in a mate pair sample). This pipeline also creates a GFF V.2 file with the two related tags pooled together.
ErrorModel	The ErrorModel pipeline constructs a run-specific empirical error model by looking at each *k*-mer for a bias where that model can be used by downstream pipelines.
CoverageBias	The CoverageBias pipeline looks at the sequence coverage bias in comparison to the reference sequence.
Variation Detection (diBayes SNP Detection)	The Variation Detection (diBayes SNP Detection) pipeline is new for Version 3. Given a reference sequence and a set of aligned reads, this pipeline performs polymorphism analysis and consensus sequence generation.

SOLiD™ Applications Interface 4

Introduction

This chapter describes the implementation of an application interface for SOLiD™ data analysis. This application interface uses a plug-in architecture. A *plug-in* is a small additional side task that is run at the same time as the main program. You can use this plug-in system to configure the Job Manager to run one or more plug-ins either before or after an AnalysisJob is analyzed. For example, you can use it to send an e-mail message when primary analysis is complete.

SAT now uses this application interface to define all secondary and tertiary analysis pipelines. A user can therefore completely replace or change existing secondary analysis pipelines or add new ones. Each pipeline is defined in a similar way as job control, with a small amount of additional information. This guide uses the term *pipeline plug-in* to distinguish a pipeline plug-in from a regular Job Manager (job control) plug-in.

SOLiD™ Applications Interface supports both Java classes and command lines to perform the tasks.

Note: This chapter describes how to create both a Job Manager plug-in and a pipeline plug-in. In actual practice, the vast majority of plug-ins created and used are for the SAT pipeline and are command-line plug-ins.

Create a Job Manager plug-in in Java

Use the following procedure to create a Job Manager plug-in in Java:

1. Create a class that implements com.apldbio.aga.common.model.plugins.IPlugin<T> (or extends AbstractPlugin). *T* is the class of thing that the plug-in handles. This is usually a subclass of BasePersistentObject. It is most commonly an AnalysisJob. You need to include com.apldbio.aga.common.jar in the build path, to gain access to the underlying classes.

```
public interface IPlugin<T>
{
   /**
    * Plugin interface to do something with this T.
    * For example, check status and send an e-mail
    *
    * @param t
    * @throws Exception
    */
   public void handleIt(T it) throws Exception;

   public void setProperties(Properties p) throws Exception;

   public Properties getProperties();
 }
```

   The additional properties are fed into the class from the defining .XML file. Refer to the example later in this chapter. These are frequently null.

2. Build or compile your code. Place all of the code onto the Job Manager classpath. Create a .jar file, and then place that jar file, *plus any other required jar files*, in the /lib/ext folder inside your Java Runtime Environment. You could also put the class files there, in the appropriate folder structure. You may instead add jar files and folders to the classpath dynamically.

3. Create an .XML file that describes the plug-in and place it inside the "Job Manager / plug-ins" folder. For the simplest job-control plug-in, which is a class requiring no arguments to its constructor and having no properties, the .XML file might look like this:

```
<PluginDescriptor>
  <definition>java:com.mycompany.mypackage.MyClass()</definition>
  <handles>com.apldbio.aga.common.model.objects.AnalysisJob</hand
  les>
  <when>after .*Primary</when>
</PluginDescriptor>
```

   The <definition> is a fully qualified name of your class that implements the plug-in. It begins with "java:" and ends with an (optional) "()".

   Job Manager plug-ins require two more tags:

   <handles> is the fully qualified name of the class (and subclasses thereof) that your plug-in wants to handle. To avoid this repetitive line for the most common types of BasePersistentObjects, place your .xml file into a subfolder of plug-ins named AnalysisJob, BasicWorkflow, or FlowcellRun.

<when> tells the Job Manager when the plug-in should be called. For AnalysisJobs, it should start with “after ” or “before ” and end with a regular expression that selects the stage. Current stage names are postFocalMapPrimary, postScanSlidePrimary, postPrimerSetPrimary, postPrimerSetSecondary, postRunSecondary. For other types of BasePersistentObjects the <when> tag needs more definition. For now, put in “after”.

Because “.*” is the regular expression syntax for “any characters”, the above plug-in is run after every primary analysis stage AnalysisJob. If <when> is “.*”, the plug-in is called both before and after every stage. A <when> of “NEVER” allows you to define a plug-in, but not a time for it to run. This is useful for defining a list (see below) or temporarily turning a plug-in off.

4. Restart the Job Manager.

Create a pipeline plug-in (for an SAT pipeline)

Set up the pipeline

An SAT pipeline reads its pipeline configuration information from the set of .XML files in the folder CORONAROOT/etc/plugins/pipelines. Like a Job Manager plug-in, the .XML file requires a definition. This definition may be a Java class that implements IPipeline (or extends AbstractPipeline). The <handles> and <when> tags are not necessary. An additional required tag <pipeline> names the pipeline, gives the unique run parameter name in the parameters.ini file, and defines which, if any, pipelines must be run beforehand. (The run parameter is either 0 or 1 to turn off /on the pipeline.)

Consider this example:

```
<PluginDescriptor>
  <definition>
    com.apldbio.aga.analysis.secondary.pipeline.PairedTagPipeline
  </definition>
  <pipeline name="MatePair" runParameterName="mate.pairs.run"
dependsOn="Matching"/>
</PluginDescriptor>
```

This code defines a pipeline named MatePair, which is implemented by the class PairedTagPipeline. MatePair depends on outputs from the Matching pipeline. In general, the pipeline needs only to implement run(), because AbstractPipeline now extends AbstractPlugin<Params>. The handleIt method looks like this:

```
public void handleIt(Params p) throws Exception {
  doAdditionalPrepBeforeHandleIt(p);
  setParams(p);
  validateParams();
  run();
}
```

By default, `doAdditionalPrepBeforeHandleIt()` does nothing, but a subclass could override the method.

You can provide optional information about the parameters used or required by your pipeline. Provide one or more <parameter> tags, such as the following:

```
<parameter name="Reference File" key="reference" description="What is
the name of the reference file?"/>
<parameter name="Mate Pair" key="is.mates" default="0" rule="b"
description="Is this a Mate Pair run?"/>
<parameter name="Window" key="coverage.bias.window" default="50"
rule="i" description="What is the size in bases of the Coverage Bias
window?"/>
```

Note: There is currently no connection between this .XML code and the pipeline code. If the .XML code says that a default value is 50, the Java code still needs to set the value to 50. SETS attempts to use this information to display and/or validate the input to your pipeline. The attributes are given in the following table:

Attribute	Description
name	A short, user-friendly name for the parameter
key	The unique computer key for this parameter, usually including dots
default	Optional, the default value for that parameter
rule	Optional, see below
description	An optional longer, user-friendly description of the parameter

A rule is an optional String.

- “b” means Boolean. The value must be “0”, “1”, “true” or “false”.
- “n” means nucleotide. The value must be A, C, G, T, or U.
- “i” means an integer.
- “f” means a floating point number.
- “i” and “f” may be followed by a space and bounds restriction that more or less follows standard mathematical bounds notation. Consider these examples:
  - “i [4,78]" means an integer, 4 to 78, inclusive.
  - “f (4.5,]" means a floating point number greater than 4.5 .
  - “s” (or null) means that any String is acceptable.
  - s may be followed by a space and a regular expression. “s [FR]3” means that the String must be either F3 or R3.

Java implementation details

- The plug-in system supports String, int, and double arguments to constructors. Arguments that start with a digit or minus sign are treated as a double if they include a decimal point, otherwise as an int. Note that ‘.’ is *not* a digit. If you want a double less that 1, for example, .25, you must add a leading 0 (0.25). Enclosing quotes are usually optional, but they are required if your String begins with a ‘[‘, digit or minus sign. You cannot include a comma in a string because it is the delimiter. For Java classes that expect command-line like arguments (such as InprocessCommandLine), if the first character inside the parentheses is a ‘[‘, the contents are split into a String[] on whitespace. This constructor is called taking a String[] .

  For example, if you want a String, int, and double passed to your constructor:

  ```
  <definition>java:com.blah.FooClass(“6Flags”,45,0.356)</definition
  >
  ```

- The definition is *also* used as a *unique* ID for your plug-in. There is no separate <id> field in the .XML code. If two Java plug-ins are of the same class with the same arguments, modify the definitions to make them distinct. This circumstance is unlikely, so the class name is used as a unique identifier. Class names include a company and package name and therefore are almost always unique. To make the class name unique:

  1. Add a "//" after the last ")" and add a unique comment.
  2. Add a "dummy" argument (which your constructor must support) and vary it.
  3. Add distinctive spacing between the arguments (not recommended).

- A unique ID is used so that your plug-in is only constructed once, allowing the option to maintain state across calls. However, if you do have state, and do *not* want to maintain state across calls, be sure to reset the state each time.
- Calls to each job control plug-in are made serially via a queue. All objects are processed one at a time (to avoid synchronization issues) and in the order in which they occur. Currently, each plug-in executes within its own thread, and the queue has a maximum size of 10. Thus long running calls are possible.

  Alternatively, the Job Manager can use PBS to launch your plug-in. To request this, add the attribute `distributeMe="false"` to the PluginDescriptor tag. In this scenario, the Job Manager does not wait until the PBS job completes.
- SAT pipelines are not queued. They are submitted in the order determined by the SAT pipeline and often in parallel. Therefore, to avoid any concurrency issues, a new instance of your pipeline class is instantiated every time that SAT wants to run it. This behavior currently resides in PipelineSorter.add( ).

Create a command-line plug-in

You can implement a Job Manager or a pipeline plug-in by the use of a command line instead of a java class. To do so:

1. Follow the previous procedure, except change the definition:
   ```
   <definition>cmd:foo.py<definition>
   ```
2. Place your script (and any others that it needs) somewhere on the path for UNIX.
3. Restart the Job Manager.

Command-line implementation details

- Your command line is always called with at least one argument, --id=XXX, where XXX is the database ID number of the BasePersistentObject (AnalysisJob). It may also be called with --ini=YYY, where YYY is a filename containing the appropriate parameters (e.g., for that AnalysisJob).
- Calls are made serially, one at a time, via a queue, just as for Java plug-ins.
- Call the command as written. If you want to invoke a specific shell, include it in the definition, as follows:
  ```
  <definition>cmd:bash somecommand</definition>
  ```
  It is actually a small Java wrapper class that is called, and it executes the command line.
- You can explicitly request individual parameters as follows:
  ```
  --parm=${parmname[:default]}
  ```
  Where parm is any value, parmname is the name of a known parameter for the job, and [:default] is an optional default. For example:
  ```
  <definition>cmd:Foo.py
  –myparm=${mutation.dir:mutation}</definition>
  ```

General implementation details

- The .XML definition file names must end with .xml. To turn off (or on) a plug-in temporarily, rename the file and restart the Job Manager.
- If there are multiple plug-ins with the same <handles> and <when>, the order in which they are called is undefined. If order matters, there is a special XML syntax to define an ordered list of plug-ins. It looks like this:
  ```
  <PluginDescriptor>
    <definition>list:SomeUniqueName</definition>
    <handles>com.apldbio.aga.common.model.objects.AnalysisJob</hand
    les>
    <when>after postRunSecondary</when>
    <properties>
      <property name="step.1" value="java:com.some.plugin()"/>
      <property name="step.2" value="cmd:MyScript.py"/>
    </properties>
  </PluginDescriptor>
  ```

If all that the Java plug-in needs is a definition, you do not need to define it in its own XML file. You do not need a separate XML file that defines java:com.some.plugin(). If that plug-in required properties from an XML file, define it in its own XML file. This file should consider designating a <when> of “never”.

Example – A complete Java e-mail plug-in using JavaMail

This example requires mail.jar on the classpath to run.

```
package com.apldbio.aga.plugin.email;

import java.util.Properties;
import javax.mail.*;
import com.apldbio.aga.analysis.plugin.IPlugin;
import com.apldbio.aga.analysis.plugin.PluginDescriptor;
import com.apldbio.aga.common.model.objects.AnalysisJob;

public class EMailHandleJob implements IPlugin.Job,
PluginDescriptor.HasProperties
{
   private Properties properties;
   private InternetAddress fromIA;
   private InternetAddress[] toIA;
   private Session session;

   public EMailHandleJob() {}

   public EMailHandleJob(String ignored) {}

   public void handleIt(AnalysisJob job) throws Exception
   {
      String statusRegex = properties.getProperty("status", ".*");
      if (job.getAnalysisJobStatus().getValue().matches(statusRegex
))
         sendMail(job);
   }

   public void setProperties(Properties p) throws Exception
   {
      properties = p;
      configure(p);
   }

   protected void sendMail(AnalysisJob job) throws MessagingException
{
      MimeMessage message = new MimeMessage(session);
      message.setFrom(fromIA);
      message.setRecipients(Message.RecipientType.TO, toIA);
      message.setSubject(properties.getProperty("subject"));
      message.setText(job.toString());
      Transport.send(message);
   }

   protected void configure(Properties p) throws AddressException {
      fromIA = new InternetAddress(p.getProperty("from"));
      session = Session.getInstance(p);
      toIA = parseAddresses(p.getProperty("to"));
   }

    public static InternetAddress[] parseAddresses(String toAddr)
throws AddressException
   {
      String[] splits = toAddr.split(",");
```

```
      InternetAddress[] ias = new InternetAddress[splits.length];
      for (int i=0; i<splits.length; i++) {
         ias[i] = new InternetAddress(splits[i]);
      }

      return ias;
   }
}
```

Here is the XML file:

```
<PluginDescriptor>
<definition>java:com.apldbio.aga.plugin.email.EMailHandleJob()</defin
ition>
  <handles>com.apldbio.aga.common.model.objects.AnalysisJob</handles>
  <when>after .*</when>

  <!-- You must also have mail.jar on the classpath somewhere,
typically -->
  <!-- usr/java/jdk1.6.0/jre/lib/ext -->

  <!-- PLEASE edit the following lines!!!  --!>
  <properties>
    <property name="subject" value="email subject"/>
    <property name="mail.smtp.host" value="mars.pe-c.com"/>
    <property name="from" value="fakeFromAdddress"/>
    <property name="to" value="conradMP@appliedbiosystems.com"/>
  </properties>
</PluginDescriptor>
```

An example pipeline .XML file

```
<PluginDescriptor>

<definition>com.apldbio.aga.analysis.secondary.pipeline.MutationPipel
ine</definition>
  <pipeline name="diBayes"
      runParameterName="mutation.run"
      dependsOn="TagOutput,MatePair,ErrorModel"
      description="New for V3.  Given a reference sequence and a set
      of aligned reads, this pipeline performs polymorphism analysis
      and consensus sequence generation."/>

  <parameter name="Step Size" key="mutation.stepSize"
      default="500000" rule="i"
      description="Parameter for adjusting run time performance where
      an increase is useful if there is more RAM available (2 GB per
      core default assumption) or conversely a decrease if there is
      less."/>
  <parameter name="Call Stringency" key="mutation.callStringency"
      default="default"
      rule="S:default,med_coverage,low_coverage,high_coverage"
      description="This parameter governs the false positive trade-
      off so if the desired behavior is to call more SNPs (especially
      Hets) at lower coverage with an accepted higher false positive
      rate, change setting from 'default' to 'med_coverage'. To call
      every possible SNP (at an even higher FP rate) set
      'low_coverage'.  If there is high coverage, the 'high_coverage'
      setting could be appropriate."/>
  <parameter name="Polymorphism Rate" key="mutation.polyRate"
      default="0.001" rule="f"
      description="Estimate for the population Heterozygosity rate.
      This is the prior probability of a heterozygote SNP at a
      position, and modifying this value changes the amount of
      evidence needed to call a heterozygote.  Set the value higher
      for a species with a higher population heterozygosity rate, and
      lower for a haploid genome or a species with a lower
      heterozygosity rate."/>
</PluginDescriptor>
```

Primary Analysis 5

SOLiD™ primary analysis

What is primary analysis?

Primary analysis is the process of extracting color calls or assignments (.csfasta file) from images captured from a sequencing run. In addition, a quality value for each color call is calculated and written into the .qual file. Primary analysis also performs a number of tasks on the run status to provide real-time feedback about the run.

Primary analysis overview

Analysis Stage	postFocalMap Primary	postScanMap Primary	postPrimerSet Primary
Purpose	For each panel, analyze the focal map image to identify beads.	1. Align images. 2. Extract intensities. 3. Classify colors (color calls). 4. QV calculation 5. Generate satay images, etc. 6. Accumulate run status info.	1. Assemble color space reads per panel. 2. Concatenate per-panel reads into one single .csfasta file (optional). 3. Filter reads shorter than specified read length or duplicate reads (optional).
Input	Focal map .tiff images	.tiff images from each ligation cycle Focal map .tiff images	All .spch files
Output	Bead location (in .spch files) Image statistics and metrics (in *_FOCALMAP_summary.tab files)	1. Intensities, Color calls, Quality metrics, QV, in .spch files 2. Satay .png files 3. Summary data (tab) 4. Run status info (traffic lights) (tab).	1. Colorspace raw reads (.csfasta) 2. QV (.qual) 3. Reads statistics file (.stats file) .

Primary analysis workflow

In the primary analysis workflow, each spot on the sample slide is broken down into a number of panels. Each panel is defined by the area that is imaged by the camera in one exposure. Primary analysis handles each panel independently of the others. For each panel, primary analysis comprises a sequence of steps. When ICS has finished collecting the images for each step, it creates a job workflow in the SOLiD™ database. The Job Manager spots these job workflows and prepares and submits the needed jobs to the PBS resource manager. When a job completes, the Job Manager updates the SOLiD™ database with the appropriate analysis status.

Each primary analysis step falls into one of the following three classes:

1. **Bead finding** - Focal map images are analyzed to identify and locate beads. Additionally, various image statistics and metrics are computed and recorded.
2. **Color calling** - After a ligation cycle, the four color images are registered to the focal map image. The fluorescent intensity for each color is estimated for each bead. These color intensities are analyzed to determine a color call and quality value for each bead for that cycle.
3. **Build reads** - After a complete run, the cycle-wise results are assembled into color reads. After some optional filtering, these reads are then ready for subsequent secondary analysis (e.g., mapping to a reference sequence).

See Figure 1 for the workflow. Multiple primary analysis jobs can be run on the Linux computer cluster. The analysis results are written to various files on this cluster. Upon completion, the primary analysis output includes a summary statistics file that shows a summary of the overall color calls. This file can be used to evaluate whether primary analysis completed successfully.

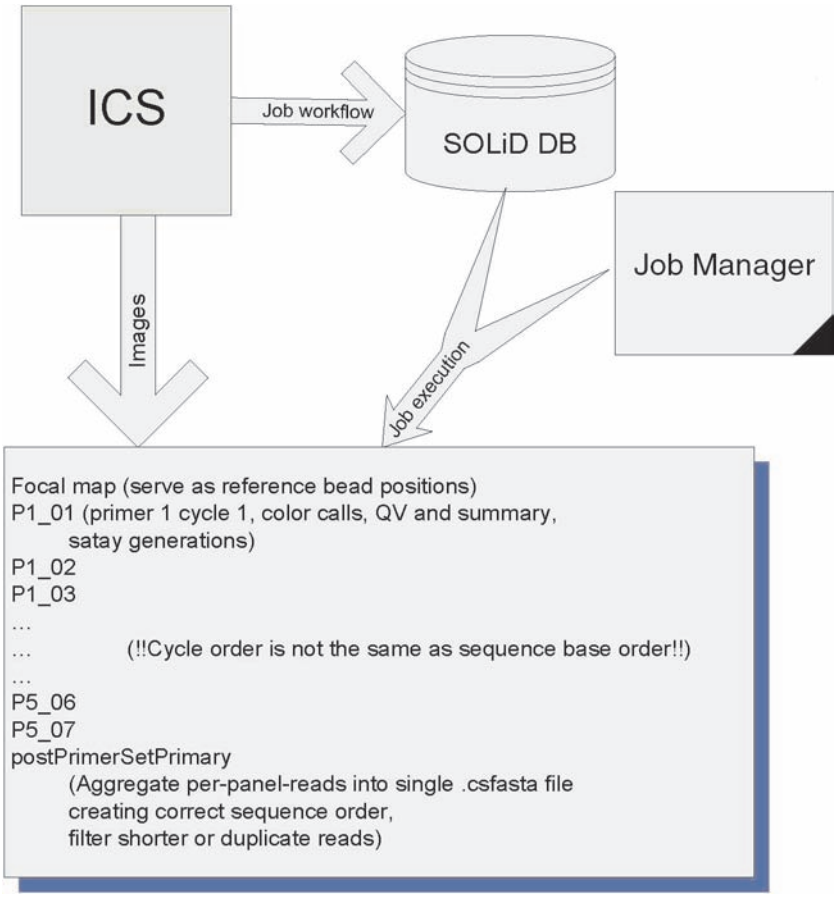


Figure 1 Primary analysis workflow.

Primary analysis inputs and outputs

The input images are in the following files:

Focal Map

```
/data/images/{run.name}/{run.name}_FOCALMAP_V1
```

Ligation Cycle

```
/data/images/{run.name}/{run.name}_F3_P1_01_V1
/data/images/{run.name}/{run.name}_F3_P1_02_V1
…
…
/data/images/{run.name}/{run.name}_R3_P1_01_V1
...
...
$sampleJobFolder =
/data/results/{instrument.name}/{run.name}/{sample.name}/jobs
$sampleResultFolder =
/data/results/{instrument.name}/{run.name}/{sample.name}/results
```

Job analysis parameter and temporary files for the Focal Map

```
$sampleJobFolder/postFocalMapPrimary.[JobID]
```

Job analysis parameters and temporary files for ligation cycle color calls

```
$sampleJobFolder/postScanSlidePrimary.[JobID]
$sampleJobFolder/postScanSlidePrimary.[JobID]
…
…
```

Job analysis parameters and temporary files for ‘postPrimerSetPrimary’

```
$sampleJobFolder/postPrimerSetPrimary.[JobID]
```

Image analysis results .spch files (SOLiD Panel Cache HDF5):

```
$sampleResultFolder/colorcalls/CACHE/{run.name}_[panelID].spch (one
full slide has 2357 panels)
```

Satay .png files

```
$sampleResultFolder/cycleplots
```

Summary data

```
$sampleResultFolder/colorcall_summary/{run.name}_FOCALMAP_V1.tab
$sampleResultFolder/colorcall_summary/{run.name}_F3_P1_01_V1.tab
..
$sampleResultFolder/colorcall_summary/{run.name}_R3_P1_01_V1.tab
..
$sampleResultFolder/colorcall_summary/{run.name}_BC_P1_01_V1.tab
```

Run status information (traffic light)

```
$sampleResultFolder/traffic_lights/
```

Color space F3 raw reads (.csfasta)

```
$sampleResultFolder/primary.[17-
digittimestamp]/reads/{run.name}_{sample.name}_F3.csfasta
```

Color space BC raw reads (.csfasta)

```
$sampleResultFolder/primary.[17-
digittimestamp]/reads/{run.name}_{sample.name}_BC.csfasta
```

Color space F3 raw reads for library (.csfasta)

```
$sampleResultFolder/libraries/{library.name}/primary.[17-
digittimestamp1]/reads/{run.name}_{sample.name}_F3.csfasta
```

Color space BC raw reads for library (.csfasta)

```
$sampleResultFolder/libraries/{library.name}/primary.[17-
digittimestamp]/reads/{run.name}_{sample.name}_BC.csfasta
```

F3 QV (.qual) file for library

```
$sampleResultFolder//libraries/{library.name}/primary.[17-
digittimestamp]/reads/{run.name}_{sample.name}_F3_QV.qual
```

BC QV (.qual) file for library

```
$sampleResultFolder//libraries/{library.name}/primary.[17-
digittimestamp]/reads/{run.name}_{sample.name}_BC_QV.qual
```

F3 Primary analysis statistics file (.stats)

```
$sampleResultFolder/primary.[17-
digittimestamp]/reads/{run.name}_{sample.name}_F3.stats
```

BC Primary analysis statistics file (.stats)

```
$sampleResultFolder/primary.[17-
digittimestamp]/reads/{run.name}_{sample.name}_BC.stats
```

F3 Primary analysis statistics file (.stats) for library

```
$sampleResultFolder/libraries/{library.name}/primary.[17-
digittimestamp]/reads/{run.name}_{sample.name}_F3.stats
```

BC Primary analysis statistics file (.stats) for library

```
$sampleResultFolder/libraries/{library.name}/primary.[17-
digittimestamp]/reads/{run.name}_{sample.name}_BC.stats
```

Reads that have less than expected read length or duplicated

```
$sampleResultFolder/primary.[17-digit timestamp]/reject
```

Reads that have less than expected read length or duplicated for library

```
$sampleResultFolder/libralies/{library.name}/primary.[17-digit
timestamp]/reject
```

Reads without barcode reads

```
$sampleResultFolder/libraries/{library.name}/missing-bc
$sampleResultFolder/libraries/{library.name}/missing-f3
```

Reads that could not be assigned to a library

```
$sampleResultFolder/libraries/{library.name}/unassigned
```

Barcode assignment statistics file/report

```
$sampleResultFolder/libraries/BarcodeStatistics. [17-
digittimestamp].txt
```

Storage requirements for primary analysis

The storage requirements values given in the table below assume a 50-base tag. If you use a tag size that is larger or smaller than 50 bases, the storage requirements differ from those given in Table 1.

Table 1 Primary analysis storage requirements

Files	Storage required	Notes about archiving
.csfasta	20 - 25 GB	Should be archived
_QV.qual	40 - 50 GB	Should be archived
.spch	975 GB	Should be archived for the lowest-level non-image backup. An estimate for the disk space needed for one .spch file (per panel) is *n**(34**m*+16) bytes, where *n* is the number of beads, and *m* is the number of cycles (F3 and R3). For 100 cycles (2x50) and 130000 beads, this yields 424 MB (per panel). A full slide has 2357 panels, which means that 975 GB are required for a full slide.
postPrimerSetPrimary.[JobID]	130 - 150 GB	Temporary files. Most of these are removed upon completion.
Intensity files	4 x 120 - 150 = 480 - 600 GB	Optional. Not generated by default

Perform primary re-analysis

To perform primary analysis manually, use the following command:

```
jprimaryanalysis.sh –-ini=PrimaryParametesr.ini
```

The ‘PrimaryParameters.ini’ file can be found in $sampleJobFolder/postPrimerSetPrimary.[JobID] with a file name ‘parameters.ini’.

To prepare for the re-analysis:

1. Create a new analysis directory (i.e., postPrimerSetPrimary.m1).

2. Copy ‘parameters.ini’ to the new directory (i.e., PrimaryParameters.ini).
3. Edit ‘PrimaryParameters.ini’ to ensure that all the paths are correct.
4. Remove parameters marked for daemon property. For example,

   analysis.job.id (default behavior does not accept non-integer)
   update.solid.script (used by daemon for update SOLiD™ database)
   logging.socket (used by daemon for logging analysis progress)
5. jprimaryanalysis.sh --ini=PrimaryParameters.ini

The following is an example of the PrimaryParameters.ini file:

```
#######################################
##- Primary Analysis Settings
#######################################

##- Instrument, sample, etc
instrument.name=Solid
run.name=Solid_20080310_1
sample.name=DH10B

panels=1-2357
primer.set=F3
read.length=35
read.prefix=T

#######################################
##- Analysis Job Parameters
focal.map.dir=Solid_20080310_1/Solid_20080310_1_FOCALMAP_V1
focal.map.stg=Solid_20080310_1/Solid_20080310_1_FOCALMAP_V1/Solid_200
80310_1_FOCALMAP_V1.STG
focal.map.version=1

##- position map. Used for generating correct sequence order
##-
images.dir.1=Solid_20080310_1/Solid_20080310_1_F3_P5_01_V1
images.dir.2=Solid_20080310_1/Solid_20080310_1_F3_P4_01_V1
images.dir.3=Solid_20080310_1/Solid_20080310_1_F3_P3_01_V2
images.dir.4=Solid_20080310_1/Solid_20080310_1_F3_P2_01_V1
images.dir.5=Solid_20080310_1/Solid_20080310_1_F3_P1_01_V1
images.dir.6=Solid_20080310_1/Solid_20080310_1_F3_P5_02_V1
images.dir.7=Solid_20080310_1/Solid_20080310_1_F3_P4_02_V1
images.dir.8=Solid_20080310_1/Solid_20080310_1_F3_P3_02_V1
images.dir.9=Solid_20080310_1/Solid_20080310_1_F3_P2_02_V1
images.dir.10=Solid_20080310_1/Solid_20080310_1_F3_P1_02_V1
images.dir.11=Solid_20080310_1/Solid_20080310_1_F3_P5_03_V1
images.dir.12=Solid_20080310_1/Solid_20080310_1_F3_P4_03_V1
images.dir.13=Solid_20080310_1/Solid_20080310_1_F3_P3_03_V1
images.dir.14=Solid_20080310_1/Solid_20080310_1_F3_P2_03_V1
images.dir.15=Solid_20080310_1/Solid_20080310_1_F3_P1_03_V1
images.dir.16=Solid_20080310_1/Solid_20080310_1_F3_P5_04_V1
images.dir.17=Solid_20080310_1/Solid_20080310_1_F3_P4_04_V1
images.dir.18=Solid_20080310_1/Solid_20080310_1_F3_P3_04_V1
images.dir.19=Solid_20080310_1/Solid_20080310_1_F3_P2_04_V1
images.dir.20=Solid_20080310_1/Solid_20080310_1_F3_P1_04_V1
images.dir.21=Solid_20080310_1/Solid_20080310_1_F3_P5_05_V1
images.dir.22=Solid_20080310_1/Solid_20080310_1_F3_P4_05_V1
```

```
images.dir.23=Solid_20080310_1/Solid_20080310_1_F3_P3_05_V1
images.dir.24=Solid_20080310_1/Solid_20080310_1_F3_P2_05_V1
images.dir.25=Solid_20080310_1/Solid_20080310_1_F3_P1_05_V1
images.dir.26=Solid_20080310_1/Solid_20080310_1_F3_P5_06_V1
images.dir.27=Solid_20080310_1/Solid_20080310_1_F3_P4_06_V1
images.dir.28=Solid_20080310_1/Solid_20080310_1_F3_P3_06_V1
images.dir.29=Solid_20080310_1/Solid_20080310_1_F3_P2_06_V1
images.dir.30=Solid_20080310_1/Solid_20080310_1_F3_P1_06_V1
images.dir.31=Solid_20080310_1/Solid_20080310_1_F3_P5_07_V1
images.dir.32=Solid_20080310_1/Solid_20080310_1_F3_P4_07_V1
images.dir.33=Solid_20080310_1/Solid_20080310_1_F3_P3_07_V1
images.dir.34=Solid_20080310_1/Solid_20080310_1_F3_P2_07_V1
images.dir.35=Solid_20080310_1/Solid_20080310_1_F3_P1_07_V1

##################################
##- Daemon Properties
##- These are optional
##-
analysis.ini=/data/results/Solid/Solid_20080310_1/DH10B/jobs/postPrim
erSetPrimary.m1/PrimaryParameters.ini
analysis.name=Filter_Fasta
analysis.pipeline=Class:com.apldbio.aga.analysis.primary.core.PrimerS
etPrimaryStage
analysis.stage.name=postPrimerSetPrimary

##################################
##- Run Directories
##- The analysis output directories. Make sure that these are correct.
##-
analysis.run.dir=/data/results/Solid/Solid_20080310_1/DH10B
analysis.results.dir=/data/results/Solid/Solid_20080310_1/DH10B/resul
ts.01
summary.dir=/data/results/Solid/Solid_20080310_1/DH10B/results.01/bas
ecall_summary
colorcallfolder=/data/results/Solid/Solid_20080310_1/DH10B/results.01
/colorcalls
cycleplots.dir=/data/results/Solid/Solid_20080310_1/DH10B/results.01/
cycleplots
analysis.sample.dir=/data/results/Solid/Solid_20080310_1/DH10B
analysis.work.dir=/data/results/Solid/Solid_20080310_1/DH10B/jobs/pos
tPrimerSetPrimary.m1
reads.result.dir.1=/data/results/Solid/Solid_20080310_1/DH10B/results
.01/primary.m1/reads
read.dir=/data/results/Solid/Solid_20080310_1/DH10B/results.01/primar
y.m1/reads
```

Redo the read-filtering process by trimming last several bases

1. Go to $sampleJobFolder/postPrimerSetPrimary.[JobID] folder, then do following modification in parameters.ini file

   ```
   read.length={your desired read.length}
   ```

2. Execute following command:

   ```
   jprimaryanalysis.sh --ini=parameters.ini
   ```

Check the completion of primary analysis

The summary of the primary analysis results is written in a .stats file. This file is called

'{run.name}_{sample.name}_F3.stats' for F3 tag analysis

or

'{run.name}_{sample.name}_R3.stats' for R3 tag analysis.

It can be found in /data/results/{instrument.name}/{run.name}/{sample.name}/results/primary.[17-digit timestamp]/reads/ .

The following is an example of the file content:

```
# Mon Mar 31 22:28:42 2008 /share/apps/corona/bin/filter_fasta.pl --
noduplicates --
output=/data/results/Solid/Solid_20080310_1/DH10B/results.01/primary.
20080328155539890 --name=Solid_20080310_1_DH10B_ --tag=F3 --
minlength=35 --mask=1111111111111111111111111111111111 --prefix=T
/data/results/Solid/Solid_20080310_1/DH10B/jobs/postPrimerSetPrimary.
847/rawseq
# Cwd: /home/pipeline
# Title: Solid_20080310_1_DH10B_
Totals (2357 p): 81935898 / 90577909 (90.5%): (Duplicates: 8085
[0.0%] (TooManyErrors: 8633926 [9.5%]
Average: 34762.8 / 38429.3 (90.6%): (Duplicates: 3.4 [0.0%])
(TooManyErrors: 3663.1 [9.4%]) Usable (2251 p) (95.5%): 81935898 /
86395269 (94.8%): (Duplicates: 8085.0 [0.0%]) (TooManyErrors:
4451286.0 [5.2%]) Usable Average: 36399.8 / 38380.8 (94.9%):
(Duplicates: 3.6 [0.0%]) (TooMnyErrors: 1977.5 [5.1%])
```

The last several lines show the summary of the primary analysis. If this file is empty or the last four lines are missing, the primary analysis has not finished or encountered some problem and terminated.

Archive primary analysis data

The image analysis results are stored in .spch files. These files are located in $sampleResultFolder/colorcalls/CACHE/ .

.spch	975 GB	Should be archived

The color call results, including colorspace raw reads and QV data, are located in $sampleResultFolder/primary.[17-digit timestamp]/reads/ .

.csfasta	20 - 25 GB	Should be archived
_QV.qual	40 - 50 GB	Should be archived
.stats	< 10 KB	Should be archived
postPrimerSetPrimary.[JobID]		Temporary files. They are removed after completion. Located in $sampleJobFolder/

There are also colorspace raw reads that were filtered in the sequence generation process. These raw reads include duplicate reads and reads with less than the expected read length, in $sampleResultFolder/primary.[17-digit timestamp]/reject/ .

.csfasta.reject	< 1 GB	Should be archived
_QV.qual.reject	< 1 GB	Should be archived

In addition, the following files can be ignored. They can be archived if needed.

Intensity files	4 x 120 - 150 = 480 - 600 GB	Optional. Not generated by default. $sampleResultFolder/primary.[17digit timestamp]/reads/ or $sampleResultFolder/primary.[17digit timestamp]/reject/

Image metrics

Image metrics are measures of the quality of the images produced by the SOLiD™ 3 instrument. The principal image metrics are the blur metric and the exposure metric.

Blur metric is a real number greater than zero that provides a measure of the average apparant radius of the beads in an image. The blur metric is computed for every image that is analyzed by primary analysis. The blur metric supports diagnosing instrument focusing and vibration issues. Figure 2 gives two examples of the blur metric:

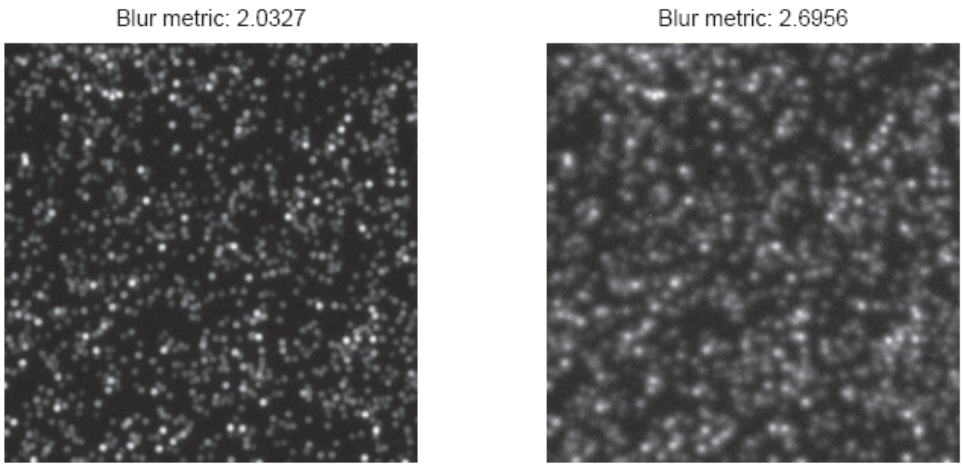


Figure 2 Two examples of the blur metric.

The *blur population* reports the population of panels that have a blur value above a given threshold. A larger average size can be caused by a small number of panels with large outlier values or a large number of panels with a large size. The two values taken together can indicate whether there is a localized problem over the whole sample (larger blur value but no outliers), a regional problem (an outlier population that spans channels), or a transient problem (outliers in a single channel).

Exposure metric is a real number that measures whether and by how much an image is over- or under-exposed, relative to a particular definition of ideal exposure. A value of zero indicates ideal exposure. A value greater than zero indicates over-exposure, and one less than zero indicates under-exposure. The scale is logarithmic with base two; a value of 1 indicates that the image is exposed twice as long as ideal.

The typical ideal exposure level of an image is one that maximizes the dynamic range of the content, with some signal occupying the lowest (darkest) range and some information extending to the saturation (highest) point. An overexposed image compresses too much of the content at the top end of the range. Overexposure creates a non-linearity in response and makes it impossible to differentiate between the highest signals. An underexposed image does not extend the content of the image across the possible range. Underexposure creates a low-end compression and fails to utilize the available range of the system.

Note: For purposes of color-calling algorithms, slightly overexposing the images yields better results.

Figure 3 gives two examples of the exposure metric:

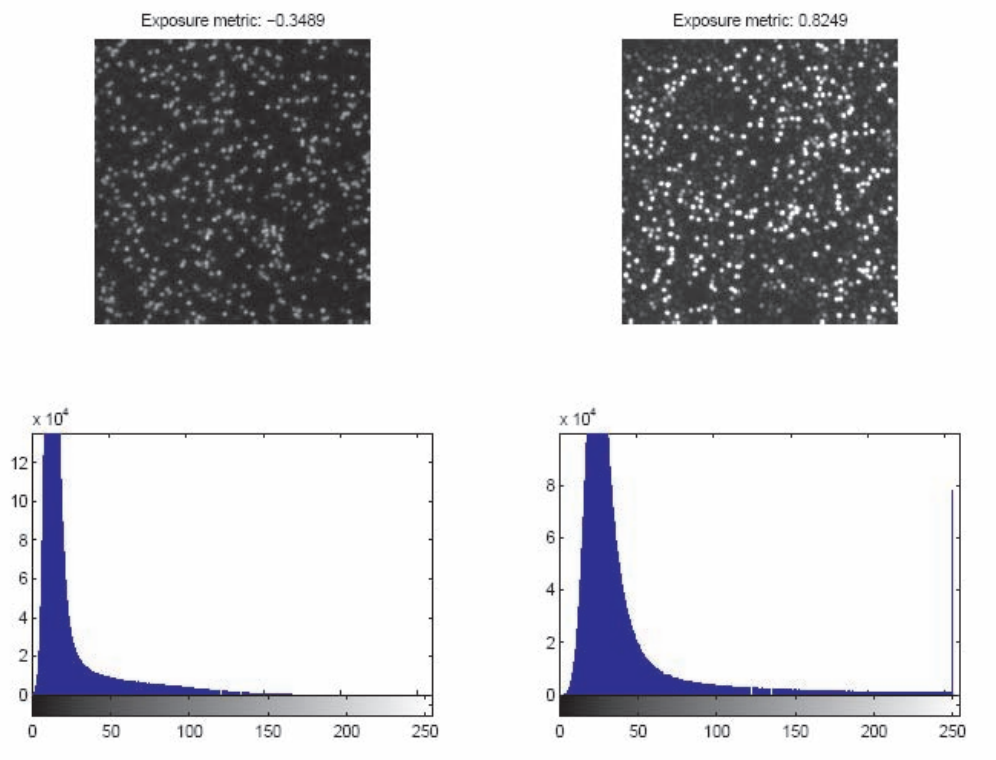


Figure 3 Two examples of the exposure metric.

The exposure value calculated is a measure of how close the system is to capturing an image that maximizes the use of the dynamic range. Looking at the tail behavior of the pixel distribution makes the determination. If the tail does not extend to the saturation point, the image is underexposed. If the tail has a peak at the far right side that is a measurement of the saturation, the image is overexposed.

The *exposure population* reports the population of panels that have an exposure quality beyond a given threshold. A large average size can be caused by a small number of panels with large outlier values or a large number of panels with a large size. The two values taken together can estimate whether there is a localized problem over the whole sample (larger deviation with no outliers), a regional problem (outlier population that spans channels), or a transient problem (outliers in a single channel). A difference with the blur metric, however, is that whereas the blur metric is constant over the number of beads present in a panel, the exposure is more sensitive to the number of beads. A baseline exposure value per channel is determined based on a sampling strategy, but panels with either significantly more or fewer beads can be affected differently. As with the blur value, the reported value is the largest mean exposure value of the four channels over all panels in the sample.

Understanding quality values in the SOLiD™ 3 System

The SOLiD™ primary analysis software analyzes the raw image data collected by the system and generates a color space read for each tag. The primary analysis software generates a quality value (QV) for each color call, which is an estimate of confidence for that color call. The SOLiD™ QVs are similar to those generated by Phred and the KB Basecaller for capillary electrophoresis sequencing. The quality value q for a particular call is mathematically related to its probability of error p.

$$q = -10\log_{10} p$$

The algorithm used to predict QVs for the SOLiD™ 3 System is similar to Phred and the KB Basecaller. Refer to the paper, Ewing B., Green P., 1998, Base-calling of automated sequencer traces using *Phred*. II. Error probabilities. *Genome Research* 8:186-194. The algorithm relies on training or calibration to a large set of control data and color calls for which the correct call is known. In the SOLiD™ 3 System, the correct call is determined by mapping the read to a known reference sequence.

Usage

Given a read with QVs, the expected number of errors in the read can be estimated. If the QV of the i-th call is q_i, then the expected number of errors in the read is

$$m = \sum_{i=1}^{n} p_i$$

where

$$p_i = 1.0022 \cdot 10^{-q_i/10}$$

is the predicted probability of error for the i-th call. This probability includes the fact that QVs are conventionally rounded to the nearest integer. The average error rate for this read is m/n, which can be converted into an overall quality score for the whole read.

$$Q = -10\log_{10}\frac{m}{n}$$

Validation and observed quality

The conventional method of validating QVs involves computing a so-called observed quality for all the calls with a particular predicted quality in a set of control data. Figure 4 shows such a validation.

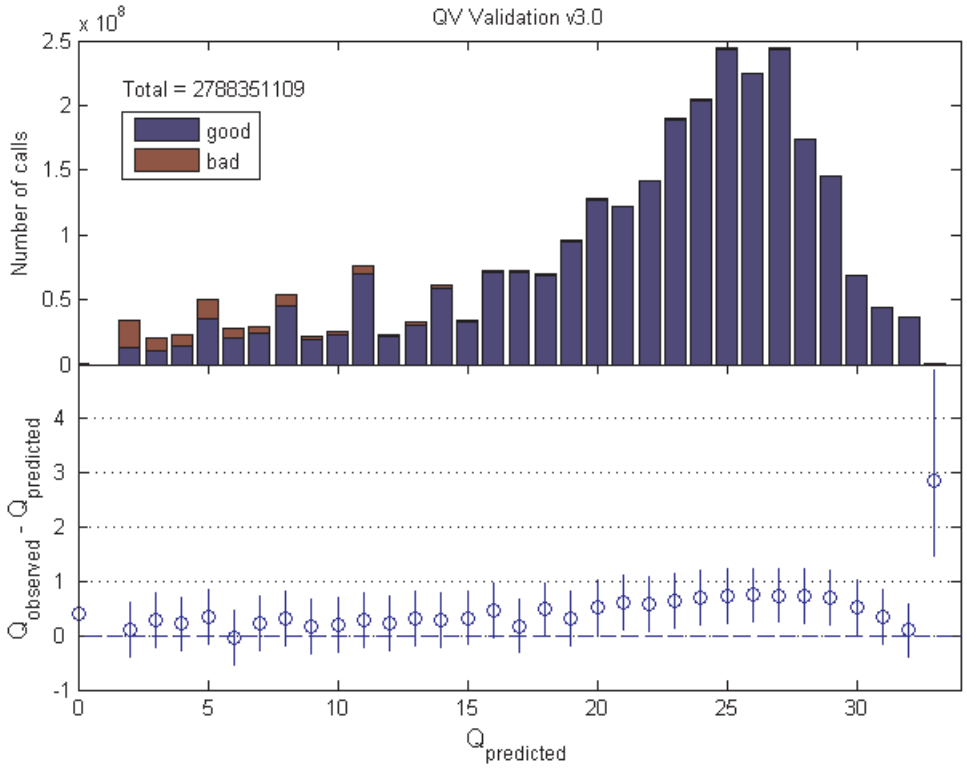


Figure 4 Statistical characteristics of quality values (QV) for the SOLiD™ 3 System.

The top panel shows distribution of QVs. The bottom panel shows deviation of observed from predicted QVs.

Key files generated by primary analysis

The key files generated by primary analysis are:

- **xxxx_sequence.csfasta** - the raw data from all beads
- **xxxx_sequence.QV.qual** - the quality values file
- **spch files** - cache files for primary analysis

xxxx_sequence.csfasta

The .csfasta file is a color space fasta file that contains the color calls generated for each tag, with the last base of the primer prepended.

Its format is:

```
>TAG_ID
Color_space
```

For example:

```
>1_88_1830_R3
G32113123201300232320
>1_89_1562_R3
G23133131233333101320
```

This file contains all the data that passed filtering, for complete reads (for example, 35 base read has 35 bases). This file is the input to the alignment tool.

xxxx_sequence.QV.qual

The QV.qual files are FASTA-like files that list the quality values in sequence order (not cycle order).

```
TAG_ID
Quality Values:
```

```
>97_2040_1850_F3
```

```
38 36 26 33 41 26 24 33 28 31 27 23 5 35 32 31 11 10 24 38 22 24 7 12
15 21 12 18 34 31 27 11 15 26 13 14 17 17 13 12 8 5 17 5 12
```

```
>97_2040_1898_F3
```

```
41 41 41 38 32 29 39 24 23 36 32 38 25 30 28 21 27 33 34 33 24 27 9 35
34 14 30 18 33 8 13 32 10 31 24 7 22 5 27 30 21 5 0 27 9
```

Note: All primary data (with exception of images) is stored in the file xxxx.spch . This file is in the hdf5 format. Tools to view and extract this data are available at http://hdf.ncsa.uiuc.edu/HDF5-FAQ.html. The data within the .spch file is organized by cycle. You can extract data in any format for your own analysis pipelines.

SPCH Files

Results of primary analysis for each panel are stored in files with the .spch extension (Solid Panel Cache HDF5). The .spch file is used as a cache; that is, primary analysis both writes results to the file and reads results from the file as analysis proceeds on a cycle-by-cycle basis. Data within the .spch file is organized by cycle. When all cycles have been analyzed, a final processing step is run to convert the data from the cycle-based format to the "read-based" ASCII formats (.csfasta and .qual).

SPCH files use the HDF5 format, which is a hierarchical binary format. Details on this format are available at http://hdf.ncsa.uiuc.edu/products/hdf5/index.html.

Tools to view the contents and extract results are available at http://hdf.ncsa.uiuc.edu/hdf-java-html/hdfview/index.html.

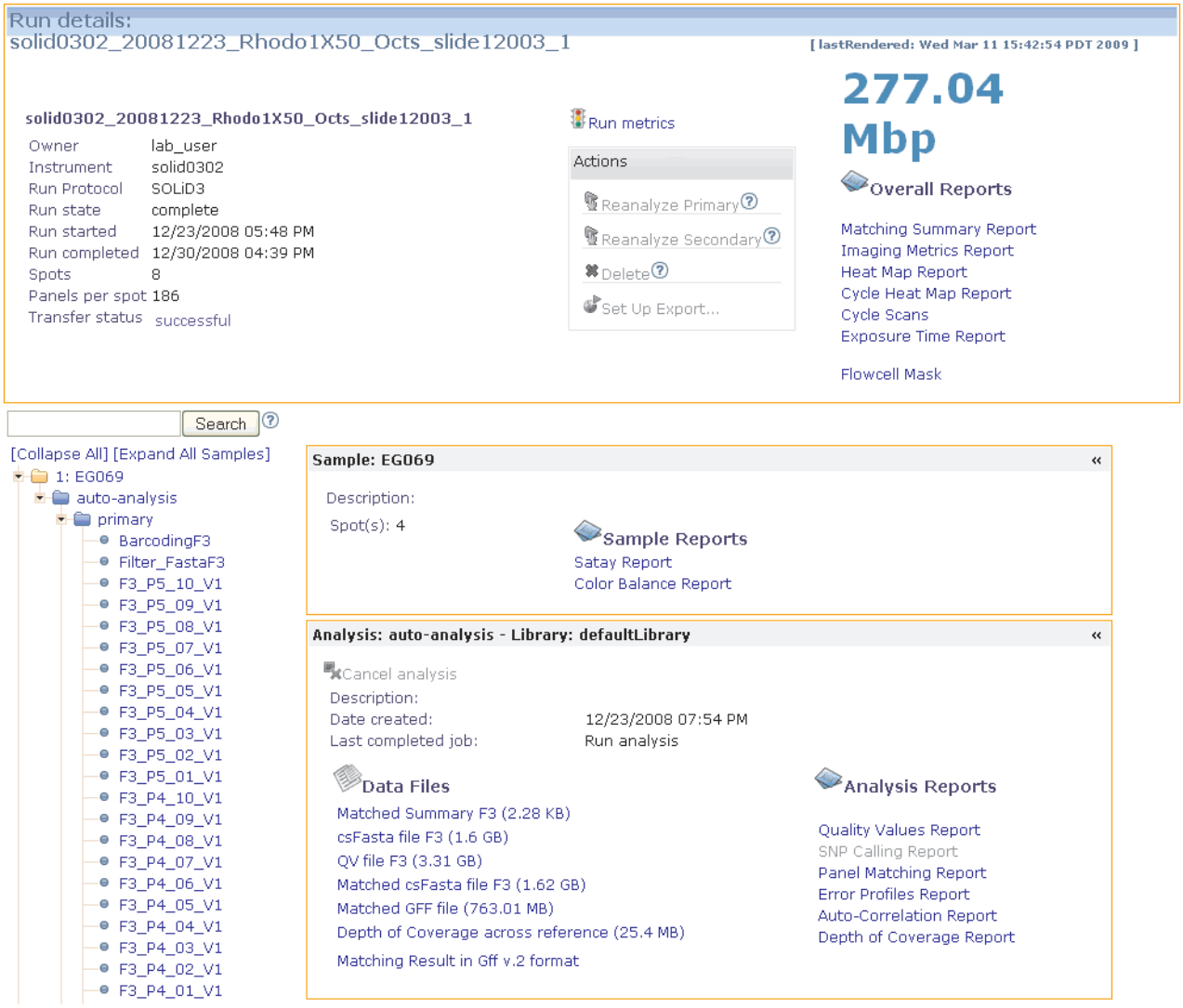
Procedures to verify primary analysis status

To verify primary analysis status:

1. From SETS, click on the History menu to go to Run History view, then click the run name of interest.

   The numbered items in the data tree on the left side of the page relate to the individual samples on the slide.

2. Click one of the data tree sample numbers on the left of the page to see the Library file name, Spot number, Sample name, and Reports information for that particular sample.



3. Click the drop-down arrow to the left of the sample folder to view individual analysis cycles for each sample.

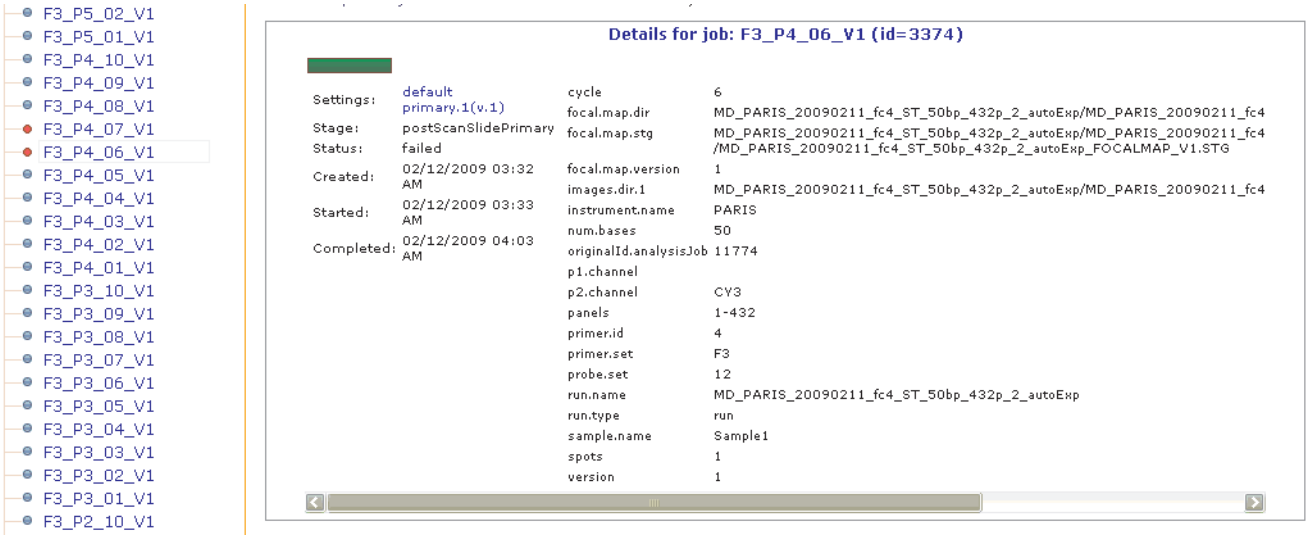
4. Check that all the analysis completed successfully. A blue dot before the analysis name means that the analysis completed successfully.

   Clicking the name of the analysis library shows additional information about the analysis.

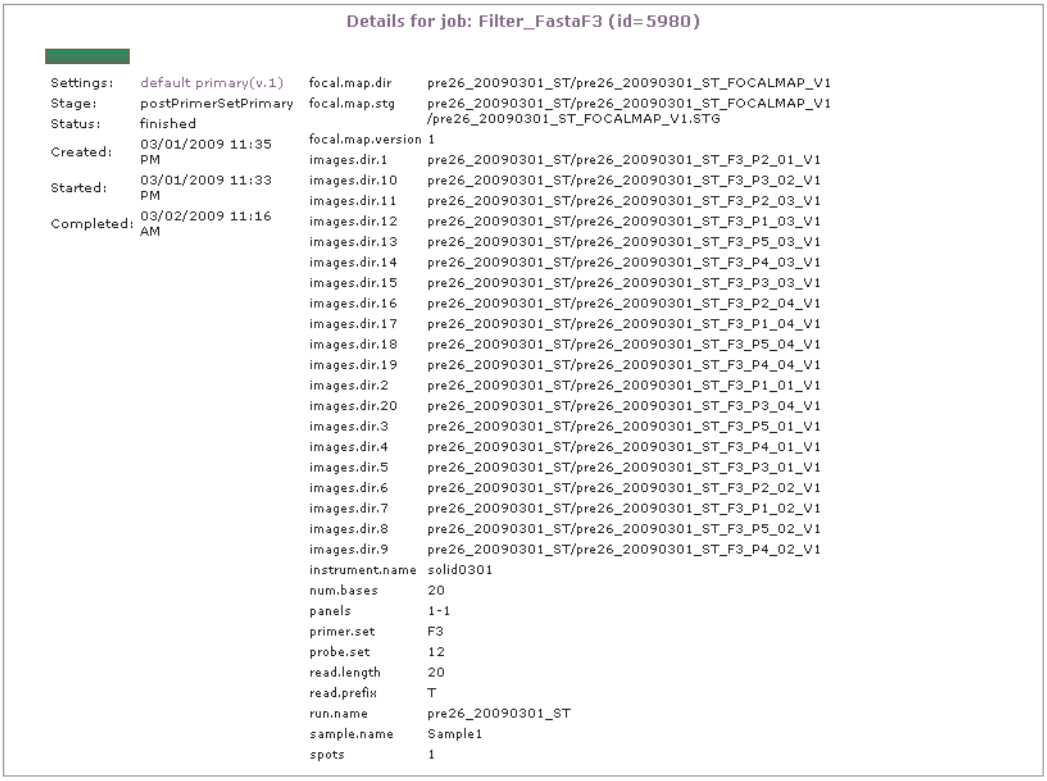
Troubleshooting analysis failure

If there is a red dot before the analysis job name, the analysis job failed. Do the following to see details:

1. Click the name of the failed analysis job. The following example shows 4 failed analysis jobs: F3_P4_02_V1, F3_P4_01_V1, F3_P3_04_V1, and F3_P3_03_V1. The right portion of the figure shows the details of analysis F3_P4_06_V1.



2. Check Filter_FastaF3 and Filter_FastaR3 (if paired-end library) to see if the analysis job was successful. Check the details for the analysis job and each cycle to ensure that they are correct.



Details for job: Filter_FastaF3 (id=5980)

Settings:	default primary(v.1)	focal.map.dir	pre26_20090301_ST/pre26_20090301_ST_FOCALMAP_V1
Stage:	postPrimerSetPrimary	focal.map.stg	pre26_20090301_ST/pre26_20090301_ST_FOCALMAP_V1 /pre26_20090301_ST_FOCALMAP_V1.STG
Status:	finished	focal.map.version	1
Created:	03/01/2009 11:35 PM	images.dir.1	pre26_20090301_ST/pre26_20090301_ST_F3_P2_01_V1
Started:	03/01/2009 11:33 PM	images.dir.10	pre26_20090301_ST/pre26_20090301_ST_F3_P3_02_V1
Completed:	03/02/2009 11:16 AM	images.dir.11	pre26_20090301_ST/pre26_20090301_ST_F3_P2_03_V1
		images.dir.12	pre26_20090301_ST/pre26_20090301_ST_F3_P1_03_V1
		images.dir.13	pre26_20090301_ST/pre26_20090301_ST_F3_P5_03_V1
		images.dir.14	pre26_20090301_ST/pre26_20090301_ST_F3_P4_03_V1
		images.dir.15	pre26_20090301_ST/pre26_20090301_ST_F3_P3_03_V1
		images.dir.16	pre26_20090301_ST/pre26_20090301_ST_F3_P2_04_V1
		images.dir.17	pre26_20090301_ST/pre26_20090301_ST_F3_P1_04_V1
		images.dir.18	pre26_20090301_ST/pre26_20090301_ST_F3_P5_04_V1
		images.dir.19	pre26_20090301_ST/pre26_20090301_ST_F3_P4_04_V1
		images.dir.2	pre26_20090301_ST/pre26_20090301_ST_F3_P1_01_V1
		images.dir.20	pre26_20090301_ST/pre26_20090301_ST_F3_P3_04_V1
		images.dir.3	pre26_20090301_ST/pre26_20090301_ST_F3_P5_01_V1
		images.dir.4	pre26_20090301_ST/pre26_20090301_ST_F3_P4_01_V1
		images.dir.5	pre26_20090301_ST/pre26_20090301_ST_F3_P3_01_V1
		images.dir.6	pre26_20090301_ST/pre26_20090301_ST_F3_P2_02_V1
		images.dir.7	pre26_20090301_ST/pre26_20090301_ST_F3_P1_02_V1
		images.dir.8	pre26_20090301_ST/pre26_20090301_ST_F3_P5_02_V1
		images.dir.9	pre26_20090301_ST/pre26_20090301_ST_F3_P4_02_V1
		instrument.name	solid0301
		num.bases	20
		panels	1-1
		primer.set	F3
		probe.set	12
		read.length	20
		read.prefix	T
		run.name	pre26_20090301_ST
		sample.name	Sample1
		spots	1

The above example shows detailed information for one Filter_FastaF3 run. This run uses 5 primers (P1-P5) and for each primer, five cycles were used. This analysis used all 5 primers (P1-P5) and for each primer, five cycles were included in the analysis job, indicating no missing images in the analysis.

Sometimes, an analysis job might be reported to be successful (no red x or warning dot next to the analysis name), but some of the primer/cycle combinations were not included in the analysis. Even if the analysis job is successful, if it is incomplete, it should be treated as a failed analysis job.

3. Log in to the instrument, and look in the directory {SampleResultsFolder}/primary.{17-digit timestamp}/reads, there should be a file named {run.name}_F3.stats for F3 tag, or {run.name}_R3.stats for R3 tag.

   This file contains information about the color space reads generation for each panel. It contains the number of reads and whether they have too many errors, too short, or duplicates. At the bottom of the file, there should be a line beginning with Totals (### p). This line summarizes the number of panels found and the percentage of error, short or duplicate reads. After this line, there should be a line beginning with Usable (### p) (##%). This line summarizes how many panels were usable and the percentage of erroneous, short, or duplicate reads among the usable panels.

   Note: If these two lines are missing, the primary analysis did not complete.

Procedure for re-analyzing the image (primary analysis)

If there was failure in the automatically generated primary analysis pipeline, there are two ways to do primary re-analysis of the image. Before doing so, look at the primary log files to find out reason of failure. The log files are in the folder:

```
/data/result/{instrument.name}/{run.name}/{sample.name}/jobs/{stage.name}.[JobId]/analysis/
```

Also, ensure that all images are present at /data/images/{run.name} .

After finding out the reason, change primary analysis parameters. If you want to re-analyze through SETS, follow the steps below:

1. Log in to SETS using an administrative account.
2. Modify used primary analysis settings. Either modify added parameters or add your new values in Additional parameter field.
3. Open run details page for that run and do primary re-analysis.

To perform primary analysis manually:

1. Go to job folder at

   ```
   /data/result/{instrument.name}/{run.name}/{sample.name}/jobs/{stage.name}.[JobId]/
   ```

2. Modify the parameters.ini file with the desired value.
3. Issue the following command:

   ```
   jprimaryanalysis.sh --dir=<parameter file directory>
   --ini=<parameter file name>
   ```

   For example:

   ```
   jprimaryanalysis.sh
   --dir=/data/results/DAEMONAC2/LAST_MP_1245_041008/Sample1/jobs/
   postPrimerSetPrimary.3793 --ini=parameters.ini
   ```

Secondary Analysis 6

SOLiD™ Secondary Analysis

What is secondary analysis?

Secondary analysis refers to application-specific analysis. It consists of a set of serial steps of modular workflow, called the *SAT analysis pipeline*. A custom analysis pipeline can be integrated with SAT so that it can be automated and can use SAT for unified job management. Refer to Chapter 4, *SOLiD™ Applications Interface*. The native SAT pipeline that comes with the software is a resequencing and variation analysis pipeline. It uses the following steps:

1. Filter reads.
2. Map or align reads to a reference genome.
3. Perform standard analysis (error profiles, coverage, .gff files, sequencing correlation, etc.).
4. If the run is a mate-pair run, an additional mate-pair rescue is performed (optional).
5. Perform Variation Detection (SNP/DNA variation consensus calling).

You can specify additional analysis pipeline steps by the use of the SETS interface, including coverage analysis, sequencing bias analysis, sequencing chemistry correlation analysis, and the like. In a manual run (command-line environment), you can do this by use of the parameter file.

Secondary analysis overview

Analysis Stage	postPrimerSetSecondary	postRunSecondary
Analysis	1. Filter reads by QV (optional) 2. Align reads to reference genomes. 3. Standard analysis (error profiles, coverage, gff, sequencing correlation, etc.) 4. diBayes SNP Detection (SNP, consensus calling) 5. Additional analysis	1. MatePairing (if a MatePair run) 2. Standard analysis (error profiles, coverage, gff, sequencing correlation, etc) 3. diBayes SNP Detection (SNP, consensus calling) 4. Additional analysis

Analysis Stage	postPrimerSetSecondary	postRunSecondary
Input	Raw sequence reads (.csfasta) Quality values (_QV.qual) Reference fasta sequences	Matched color space fasta (.csfasta.ma.[dd].[d]) Quality values (_QV.qual) Reference fasta sequences GFF V.2 files (.v2.gff)
Output	Matched color space fasta (.csfasta.ma.[dd].[d]) GFF V.2 (.v2.gff) Error profiles per color call (tab) Consensus fasta sequence in base List of differences compared to reference sequences (GFF V.3) Various analysis output (tab)	MatePairing results (mates file) GFF V.2 (.v2.gff) Error profiles per color call (tab) Consensus fasta sequence in base List of differences compared to reference sequences (GFF V.3) Various analysis output (tab)

Secondary analysis workflow

Secondary analysis starts with color space sequence reads from primary analysis and validated reference genome sequences in fasta format. You can use the Filtering pipeline to filter the QV (quality values) before analysis if you want (optional). The alignment of reads to the reference genome uses discontinuous word patterns to perform a fast and efficient search. This process is divided into several steps and is covered later in this chapter. The mapped reads are then processed through TagAnalysis, which converts each mapped read into a version .2 gff entry. This entry contains the color space read, the quality values, the base space translation, and any reference deviations that were found. The gff v.2 files are then processed to aggregate error profiles and other user feedback by the use of SETS. If you want, the reads can be processed through the diBayes SNP Detection pipeline (optional). In this pipeline, coverage is taken into account to determine aggregate SNP calls against the reference and a consensus sequence. Other optional pipelines include the ErrorModel pipeline, which performs more extensive error modeling, and the CoverageBias pipeline, which performs basic coverage analysis.

Applied Biosystems recommends that you perform complex genome analysis on a different computer system from the SOLiD instrument. Running Global SETS Version 3 on an off-line computer cluster can pick up the job automatically when a run is set for automatic data export and remote secondary analysis. Refer Chapter 7, *Off-line Data Analysis*. This chapter contains a comparison of SAT and Corona-Lite for off-line data analysis.

To run pipelines

This section discusses the input, parameters, output, and running conditions for the following pipelines:

- Filtering
- Matching group - Aligns each read to the reference sequence. This group consists of the RepeatClassifier, Matching Random, Matching Repeat, and MatchingConsolidate pipelines.
- TagOutput - Generates result files and reports for a fragment run analysis.
- MatePair - Calculates valid mates based on insert size and performs rescuing.
- ErrorModel - Looks at each k-mer for the probe bias that indicates the likelihood of having an error.
- CoverageBias - Looks at the sequence coverage bias in comparison to the reference sequence.
- diBayes SNP Detection - Detects sequence variations and generates consensus sequences.

Refer also to Chapter 3, which gives an introductory description of pipelines.

To run the secondary analysis pipeline

The SOLiD analysis pipeline system supports two different modes of use: interactive analysis from the command line and automated analysis through the auto-analysis daemon system. Both modes are accessed through a command-line interface.

A simple, direct way to run the pipeline from the command line is:

```
jmoap.sh [--debug] --dir=<working directory> --ini=<parameters.ini>
[--block]
```

The following is an example command line:

```
jmoap.sh --block --logfile= FC1_R3.log ini=parameters.ini
```

Execution of this command reads parameters from a file named “analysisParameters.ini” and writes a process log to an output file named “FC1_R3.log”.

When this command is executed, the pipeline process looks for a PBS server for submitting the analysis job. If no server can be found, the analysis is run on the current machine (this mode can be forced by using the `-- block` command-line option).

Command-line options

Table 1 Command-line options

Parameter	Optional	Description
--block	Yes	If specified, the pipeline process is run under the current shell on this computer
--logfile=...	Yes	Specifies a file to contain the output of running the pipeline processes and various useful logging output

--ini=...	Yes	This is an alternate option for specifying a parameter file to use. If this parameter is not specified, the pipeline assumes the first non-option argument is a parameter file.
--debug	Yes	This option produces verbose output at the command line.

Parameter file

For a secondary analysis job launched from ICS/SETS, the default location for parameter file and the log file is

```
/data/result/${instrument.name}/${run.name}/${sample.name}/jobs/postP
rimerSetSecondary.[jobid]
```

The parameter file specified to 'jmoap.sh' contains parameter choices that are specific to the requested analysis. The format of the file is a simple "key=value" properties file with lines of the form "parameterName=value" and comments delimited by lines that start with "#". Refer to the listing that follows:

```
####################
#Analysis Job Parameters
#
# used in auto-analysis
####################
spots=4
sample.name=[SAMPLE_NAME]
panels=1279-1704
job.stage.name=postPrimerSetSecondary
job.run.id=23
job.id=2197
update.solid.script=updateSOLiD.py
logging.socket=[MACHINE].sequencer/[IP_ADDRESS] 8765
analysis.settings.id=20
sample.id=51

#Daemon Properties
analysis.ini=/data/results/[MACHINE]/[MACHINE]_[DATE]_[RUN_NAME]/[SAM
PLE_NAME]/jobs/postPrimerSetSecondary.2197/parameters.ini
analysis.job.id=2197
analysis.name=Moap-unpaired
analysis.pipeline=Class:com.apldbio.aga.analysis.secondary.pipeline.J
MOAP
analysis.stage.name=postPrimerSetSecondary
analysis.status.name=in_progress
headnode.ip=10.1.1.1

####################
#Analysis Settings
#
#
####################

# Tag information
primer.set=F3
probe.set=12
num.bases=50
read.length=50
```

```
# Run information
library.name=defaultLibrary
instrument.name=[MACHINE]
run.name=[MACHINE]_[DATE]_[RUN_NAME]
reference.name=hurefALL_validated.fasta

# Pipeline Selection (1 = will run, 0 = will not)
mutation.run=0
coverage.bias.run=0
error.prone.run=0
mate.pairs.run=0
is.mates=0
single.tag.run=1
classifier.run=1
filtering.run=1
matching.repeat.run=1
iq.run=0
matching.random.run=1
matching.consolidate.run=1

#### Pipelines

## Tag Analysis / Single Tag
gff.clear=1000
single.tag.gff.option.correctTo=consistent
single.tag.gff.output.basespace=1
single.tag.gff.option.iubsTo=haplotype
single.tag.gff.option.trimMod=5
## End Tag Analysis

## Filtering
qv.length=20
qv.mask=
filtering.pass.all=0
qv.max.fail=3
qv.min.value=1
## End Filtering

### Mapping Group

## Mapping Shared Params
matching.use.iub.reference=0
matching.overlap=100
mismatch.level=4
matching.read.chunks=4
matching.max.hits=10

## End MappingShared

## Classifier
classifier.pass.all=0
## End Classifier

## MatchingRandom
matching.random.pass.all=0
matching.roc.plan.random=random
matching.roc.user.input.random=
roc.read.subtraction.random=1
roc.subtraction.threshold.random=-1
```

```
roc.stringency.lower.limit.random=19
roc.length.lower.bound.random=25
roc.mismatch.lower.bound.random=4
roc.cdf.recovery.upper.limit.random=1.0
## End MatchingRandom

##MatchingRepeat
matching.repeat.pass.all=0
matching.roc.plan.repeat=repeat
matching.roc.user.input.repeat=
roc.read.subtraction.repeat=1
roc.subtraction.threshold.repeat=-1
roc.stringency.lower.limit.repeat=21
roc.length.lower.bound.repeat=25
roc.mismatch.lower.bound.repeat=2
roc.cdf.recovery.upper.limit.repeat=1.0
## End MatchingRepeat

## MatchingConsolidate
## End MatchingConsolidate

### End Mapping Group

## DiBayes
mutation.callStringency=default
mutation.polyRate=0.001
mutation.stepSize=500000
## End DiBayes

## MatePair
insert.start=1800
insert.end=3200
good.mate.definition=none
mate.pairs.use.pairing=true
mate.pairs.rescue.level=10
mate.pairs.gff.option.trimMod=5
mate.pairs.gff.option.correctTo=consistent
mate.pairs.gff.option.iubsTo=haplotype
mate.pairs.gff.output.basespace=1
## End MatePair

## CoverageBias
coverage.bias.window=50
## End CoverageBias

## ErrorModel
## End ErrorModel

#Run Directories
analysis.job.dir.id=[EXT_DATE_CODE]
analysis.run.dir=/data/results/[MACHINE]/[MACHINE]_[DATE]_[RUN_NAME]/
[SAMPLE_NAME]
analysis.results.dir=/data/results/[MACHINE]/[MACHINE]_[DATE]_[RUN_NA
ME]/[SAMPLE_NAME]/results.01
summary.dir=/data/results/[MACHINE]/[MACHINE]_[DATE]_[RUN_NAME]/[SAMP
LE_NAME]/results.01/colorcall_summary
colorcall.dir=/data/results/[MACHINE]/[MACHINE]_[DATE]_[RUN_NAME]/[SA
MPLE_NAME]/results.01/colorcalls
```

```
cycleplots.dir=/data/results/[MACHINE]/[MACHINE]_[DATE]_[RUN_NAME]/[S
AMPLE_NAME]/results.01/cycleplots
secondary.result.dir=/data/results/[MACHINE]/[MACHINE]_[DATE]_[RUN_NA
ME]/[SAMPLE_NAME]/results.01/libraries/defaultLibrary/intermediate.[E
XT_DATE_CODE]
analysis.sample.dir=/data/results/[MACHINE]/[MACHINE]_[DATE]_[RUN_NAM
E]/[SAMPLE_NAME]
analysis.work.dir=/data/results/[MACHINE]/[MACHINE]_[DATE]_[RUN_NAME]
/[SAMPLE_NAME]/jobs/postPrimerSetSecondary.2197
reads.result.dir.1=/data/results/[MACHINE]/[MACHINE]_[DATE]_[RUN_NAME
]/[SAMPLE_NAME]/results.01/libraries/defaultLibrary/primary.[PR_EXT_D
ATE_CODE]/reads
read.dir=/data/results/[MACHINE]/[MACHINE]_[DATE]_[RUN_NAME]/[SAMPLE_
NAME]/results.01/libraries/defaultLibrary/primary.
[PR_EXT_DATE_CODE]/reads
filtering.qv.filtered.dir=/data/results/[MACHINE]/[MACHINE]_[DATE]_[R
UN_NAME]/[SAMPLE_NAME]/results.01/libraries/defaultLibrary/intermedia
te.[EXT_DATE_CODE]/qvfiltered
filtering.qv.filtered.dir.1=/data/results/[MACHINE]/[MACHINE]_[DATE]_
[RUN_NAME]/[SAMPLE_NAME]/results.01/libraries/defaultLibrary/intermed
iate.[EXT_DATE_CODE]/qvfiltered
filtering.qv.fail.dir=/data/results/[MACHINE]/[MACHINE]_[DATE]_[RUN_N
AME]/[SAMPLE_NAME]/results.01/libraries/defaultLibrary/intermediate.[
EXT_DATE_CODE]/qvfail
filtering.qv.fail.dir.1=/data/results/[MACHINE]/[MACHINE]_[DATE]_[RUN
_NAME]/[SAMPLE_NAME]/results.01/libraries/defaultLibrary/intermediate
.[EXT_DATE_CODE]/qvfail
rawseq.dir=/data/results/[MACHINE]/[MACHINE]_[DATE]_[RUN_NAME]/[SAMPL
E_NAME]/jobs/postPrimerSetPrimary.2195/rawseq
```

Key input and output files in secondary analysis

In all files in this chapter, *xxxx* is the file name, which consists of a run name along with extra information (for example, spot number, or number of cycles).

The following table summarizes the input and output in all secondary analysis scenarios.

	Input for SAT pipeline	Required Parameter files for stages	Output Results
Fragment Library Data Analysis	Primary analysis results in .csfasta, .QV.qual Reference sequence in .fasta	postPrimerSetSecondary	• Matching result, including mapped reads ONLY, with base sequence in .v2.gff • Matching result, including both mapped and unmapped reads, in .csfasta.ma.xx.xx • Matching stats file in .stats.txt
MatePair Library Data Analysis	Primary analysis results in .csfasta and .QV.qual for each tag Reference sequence in .fasta	For each tag analysis at postPrimerSetSecondary And for mate pair rescuing at stage postRunSecondary	• Matching result for each tag, including ONLY mapped reads, with base sequence in .v2.gff • Matching result for each tag, including both mapped and unmapped reads, in .csfasta.ma.xx.xx • Matching stats file (.stats.txt) • .v2.gff files for mates, sorted differently by genome position or mates • F3_R3.mates
If Variation Detection is turned on			• SNP list in xxx.gff.3 • Consensus file (SNPs only) in .fasta, one per chromosome • SNP report file in .txt

Input for the secondary analysis pipeline (SAT pipeline)

The following are input files that need to be specified in the parameter file.

- **SOLiD primary analysis result**: Color space fasta reads file (.csfasta) and quality value file (.QV.qual)
- **A reference sequence (base sequence) in fasta format**: It needs to be validated by the reference_validation.pl script provided in analysis pipeline.

SOLiD primary analysis results

On the instrument cluster, the default locations of the .csfasta and .QV.qual data from the primary analysis are:

For non-multiplexed (non-barcoded) libraries:

```
/data/result/${instrument.name}/${run.name}/${sample.name}/results/li
braries/defaultLibrary/primary.[17digit-time-stamp]/reads
```

For multiplexed (barcoded) libraries:

```
/data/result/${instrument.name}/${run.name}/${sample.name}/results/li
braries/${ library.name}/primary.[17digit-time-stamp]/reads
```

Reference sequence

In the resequencing application, a reference sequence file in fasta format needs to be specified in parameter file for SAT v3.0 pipeline.

For multiple chromosome reference sequences, concatenate multiple fasta files into one single fasta file and use it as reference sequence file.

To prepare reference sequences for the pipeline, the fasta sequence file must be validated by running a perl script called *reference_validation.pl*. The script can be found on the SOLiD cluster in directory */share/apps/corona/bin*. The purpose of this step is to convert non-[ACGTN] characters to N and convert lower case characters to upper case. The new script in v3 deals with multifasta inputs and IUB codes.

Run the validation script and put it into */share/reference/genomes* and it then shows up in SETS.

If the reference sequence .fasta file is imported from SETS graphical user interface, SETS validates it automatically.

Usage:

```
[corona@plcs02 solid_bin]$ ./reference_validation.pl
Usage: ./reference_validation.pl
-r <reference_file>
-o <outputfile>
```

Sample command:

```
reference_validation.pl –r chr1.fa –o chr1_validated.fa
```

Pipeline output description

The results of secondary analysis in SOLiD v3.0 are base sequence and color space calls that have been mapped to the reference sequence. These data are ready for use in an analytical context.

The secondary analysis results appear in ${secondary.result.dir}, which is specified in the analysisParameters.ini file.

On the cluster, the results appear in $sampleResultFolder/secondary.[17-digit timestamp] .

The following table shows the directory structure for secondary analysis:

Output directory	Pipeline parameter
qc_colorcallError	single.tag or mate.pairs
qc_correlation	single.tag or mate.pairs
qc_coverage	single.tag or mate.pairs; coverage.bias
qc_mates	mate.pairs
qc_panels	single.tag or mate.pairs

qc_qualityValues	single.tag or mate.pairs
qc_reports	single.tag or mate.pairs
qc_sequenceError	error.prone
qc_tagBias	single.tag or mate.pairs
{run.name}_{sample.name}_V1	pipeline report
s_matching	matching
classifier	classifier
qvfail	filtering
qvfiltered	filtering
qc_matching	single.tag or mate.pairs

Key result files and the location are indicated in the table below.

Key results	Location
Matching result (xxx.ma) (xxx .v2.gff)	$sampleResultFolder/secondary.[17 digit timestamp] /s_matching
Matching.stats file (xxx.stats.txt)	$sampleResultFolder/secondary.[17 digit timestamp] /qc_matching
F3_R3.mates	When executed from ICS/SETS: /data/result/${instrument.name}/${run.name}/${sample.name} /jobs/postRunSecondary.[jobid] When executed from command line: working directory

Filtering pipeline

Because of increased capabilities and performance of the imaging subsystems of the instrument and increases in the length of the reads, many panels that previously would have been lost are now being rescued. Sequences that would have been screened out are now being allowed through the post-primary analysis read-filtering step. The version 2 default behavior was to screen out any sequence that had any missing information in any channel in any cycle. This could be adjusted by the use of advanced parameters passed into primary analysis, that would change the parameter of maximum errors. This would allow sequences into secondary analysis with some missing information in any channel, or to pass a screening mask in which missing data in a specific channel would be ignored.

For version 3, the concept of "minimum calls" has replaced "maximum errors" in the primary analysis settings. This is thought to be a more general concept as a single primary analysis setting could be applied to sequencing runs of differing lengths. This should allow for catching "edge" beads in panels that might miss alignment on some cycles, but still yield valuable information. If a ligation cycle image is missing for a whole panel, those sequences are no longer lost. See Table 2. Mapping treats missing information (unless a mask is passed in) as an automatic mismatch, but otherwise downstream pipelines that ship with V3 software will handle missing information. Additionally, a quality value of -1 is assigned for missing calls.

Table 2 Handling of sequence reads missing information

	SOLiD™ 2 Primary Analyses	SOLiD™ 3 Primary Analyses
Handling of those sequence reads missing information in any channel in any cycle	Reads are screened out and excluded from .csfasta output	Missing colors are indicated as dots '.'in those reads and reads are included in .csfasta output QV = -1 is assigned for missing calls

Consider these examples: If 25 colorcalls are considered to be sufficient for an application to place the tag for analysis purposes, a value of 25 can be set for minimum calls. If the sequencing run is of length 25, then all colorcalls need to be present. If the sequencing run is of length 35 or 50, 25 colorcalls are still the minimum amount of information necessary. These colorcalls do not need to be contiguous. Let's look at two sequence reads (tags):

- Tag A is a tag of length 50 that has the first 25 colorcalls and misses the last 25.
- Tag B is a tag of length 50 that misses every other colorcall.

In the former case (A), progressive mapping in SAT V3.0 (see the MatchingRandom section in this chapter) could still yield a perfect 25.0 mapping result, whereas in the latter (B), the spacing of the missing data would ensure that the read would likely never match, because a 50% error rate would have to be allowed for at any mapping length.

A combination of primary analysis filtering settings and filtering pipeline settings can be utilized to yield a sufficiently screened tag set for secondary analysis without significantly skewing the read composition.

Description:

Pipeline	Description	Behavior
Filtering	The Filtering pipeline performs QV filtering, in which any read isremoved from further pipeline processing if it has more than a specified number of calls (“Max Failures”) that are less than a specified quality value (“QV Cutoff”).	Always ‘ON’ Use “Pass All = 1” to bypass

Parameters

Parameter Name and Key	Default Value	Description
Pass All filtering.pass.all	1	Boolean (0 or 1) value to govern whether reads skip any filtering.
Max Failures qv.max.fail	4	Any read is removed from further pipeline processing if it has more than Max Failures (Max Hits) calls that are less than QV Cutoff.
QV Cutoff qv.min.value	9	Single QV cutoff value to use at every position, such that an observed QV value less than this value constitutes a failure.
Number of Bases to Use qv.length	=read.length	Number of consecutive bases to examine.
QV Mask qv.mask		A space-separated list that contains one QV per position of interest, in which an observed value less than that specified in that position constitutes a failure. NOTE: Setting this value implicitly sets (and overrides) the QV Cutoff and Number of Bases To Use values.

Note: The Filtering pipeline itself always needs to be turned on because it is run as a dependency for downstream pipelines (see below). Use "Pass All"= true/false to set whether filtering is actually done.

The Filtering pipeline remains QV-based in V3, but more parameters have been exposed in the secondary analysis settings. The pipeline itself always needs to run as a dependency for downstream pipelines, even if no filtering is actually done. This is caused by filesystem locations being created that are sourced by other later-executing pipelines. To remedy this, the 'pass all' option is the first in the list. If the pipeline is 'on' but the 'pass all' is set to true, the primary analysis output is directly linked to the filtered output and no computation takes place. If this is unchecked, then the other parameters are parsed in order to filter based on the requested behavior. The three parameters of 'max hits', 'min QV' and 'length' are used, or a QV filtering mask is.

In the case where all options are set, the values in the QV mask override any other settings. Using the first three values constructs a uniform mask of values 'min QV' to screen of length 'length', in which any read having more than 'max hits' would be placed into the qvfail directory. Using the mask directly creates the screening behavior in which the mask has a fixed length but a possible different QV at each position. The 'max hits' value would still be interpreted to determine when a tag is filtered or not.

Consider the following examples: A uniform QV value of 1 with max hits of 1 with a length of 25 creates a mask of '1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1' and serves to screen out any tag with missing data in the first 25 colorcalls. This would differentiate between the two previously discussed cases of a 'min calls' value of 25, where Tag A only has the first 25 colorcalls and Tag B misses every other color call, to ensure that the Tag A made it through while the Tag B would not. A second example could be a ‘min QV’ value of 9 with ‘max hits’ of 6 with a ‘length’ of 10 which would create a mask of '9 9 9 9 9 9 9 9 9 9'. In this case, any tag with 7 QV values less than 9 in the first 10 bases would be dropped. Finally, a value of 5 for 'max hits' with a mask entry of '8 8 8 8 8 5 5 5 5 5' would put an even higher premium on the first 5 color calls and a borderline non-filter in the second set of 5 colorcalls and no screening in the final *N* colorcalls of the tag. This is summarized in the table below.

QV mask examples	‘Max Hit’	Interpretation	Filtered Sequence Reads (Tags)
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1	‘Min QV’=1, ’Length’=25 ‘Max Hit’ = 1	Any tag with more than one missing base in the 1st 25 bases e.g. Tag A would pass; while Tag B would not (see above)
9 9 9 9 9 9 9 9 9 9	6	‘Min QV’=9, ‘Length’ =10, ‘Max Hit’ =6	Any tag with more than 6 QV<9 in the first 10 bases would be filtered
8 8 8 8 8 5 5 5 5 5	5	‘Min QV’= 8 for the first 5 bases, ‘Min QV’=5 for the 2nd 5 bases ‘Max Hit’ = 5	Any tag with total failure of more than 5 positions with QV <8 in the 1st 5 bases and QV <5 in 2nd 5 bases

Composition of a run on the SOLiD™ 3 System

Any SOLiD™ 3 System run yields several classifications of bead data that can be traced back to the emulsion PCR (ePCR) stage of the sample preparation workflow. In the ePCR phase, a non-competitive PCR step takes place in a microreactor. At a given microreactor size, for DNA and bead concentration, the goal is to fill each reactor with one bead, one molecule of DNA, and sufficient reagents to drive the reaction. The concentrations are governed by a model double-Poisson selection process. After these beads are deposited on a slide and sequenced, they yield the following four classes of events:

1. Clonal beads that match a reference sequence above a given level of stringency and arise from reactors with one bead and a single DNA molecule, where amplification was sufficient to produce surface density to support sequencing at the desired length;

2. Clonal beads that do not match a reference sequence above a given level of stringency arising from reactors with:
   a. Two beads and a single piece of DNA such that the amplification yields two beads each with insufficient surface density to support sequencing at the desired length (an ePCR issue);
   b. One bead and a single piece of DNA such that the amplification limits the ability to create the necessary surface density to support sequencing at the desired length (a sequencing issue);
   c. One bead and a single piece of DNA that, based on sequence content, has insufficient surface density to sustain serial hybridization or ligation events necessary to support sequencing at the desired length (a chemistry issue);
3. Polyclonal beads that arise from reactors with one bead and two or more molecules of DNA creating a bead with a heterogeneous population. This population does not match a reference sequence beyond random probability, since it is a mix of individual reads at any point (generally 15–20% of the population) or;
4. Unloaded beads that arise from reactors with one or more beads and only molecules of DNA that were filtered in the enrichment phase of ePCR.

Notes on how quality values are derived

Some percentage of the training set is done with mate-paired data, which allows for one tag of a paired-tag sequence to have a lower stringency in the matching phase. The fact that only matched sequences can be used to discern error rates creates a selection bias in the training event. In the set of matched sequences, the correlation of called QVs to observed error rates is quite high. However, the distribution of QVs in unmatched reads is not sufficiently ‘right-shifted’ to segregate the two populations *ab-initio*. This can be inferred from the recent run shown in Figure 1.

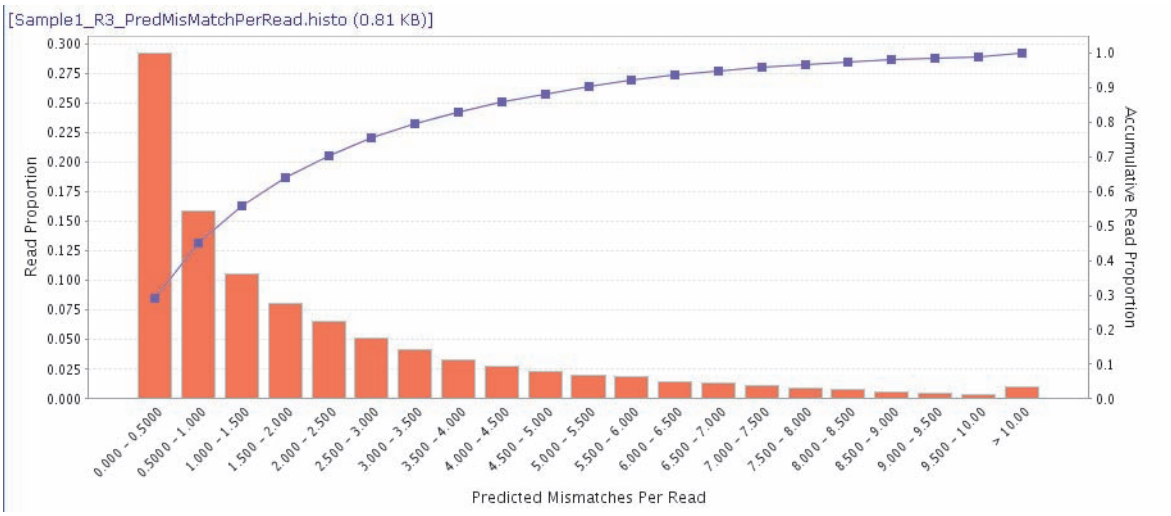


Figure 1 Predicted mismatches per read graphic using the SOLiD™ Experiment Tracking System (SETS) quality value reports.

The cumulative distribution function at the three mismatch position yields a value of 78%. The actual matching percent with less than three mismatches is 68%.

This leaves a 10% difference between the estimated and actual values. In this case, the 10% difference does not account for sequences that match the reference and have estimated error rates > 3mm (another approximately 3–5%). Until the predicted and observed values agree and the populations in the predicted and observed are the same, it is not worthwhile to prescreen sequences prior to the matching pipeline. There is also a strong sequence contextual component to a given color space call where certain sequences have a higher noise component than others. This leads to a situation where QV screening before matching does not cut sequences with a uniform composition, and it affects the sequence coverage observed.

Matching pipeline group

Basic sequence alignment

This section describes aligning SOLiD sequence reads to a reference sequence. Many sequencing applications, such as resequencing, are performed to determine an unknown sequence. In the SOLiD System, a reference sequence is used to verify the unknown sequence. The reference sequence is similar to, but not identical to, the unknown sample that is to be sequenced. An unknown sequence is determined by mapping (aligning) the set of short reads obtained from the SOLiD System to the reference sequence (Figure 2).

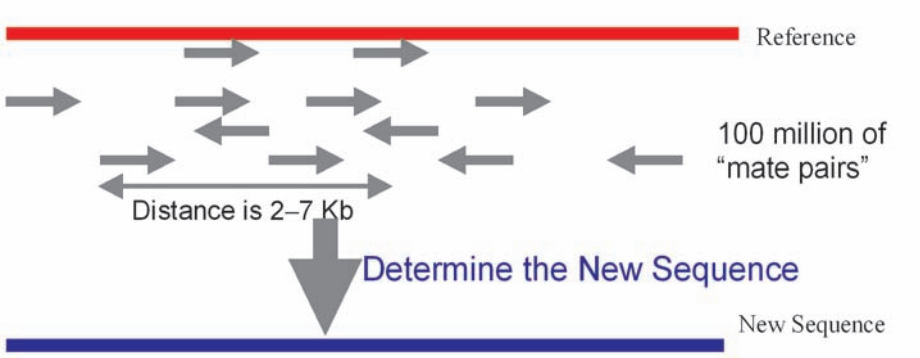


Figure 2 Map short sequences to a reference sequence.

Data obtained from the SOLiD™ System are easily aligned with a reference sequence to determine a new sequence.

Alignment overview

The alignment software Mapreads employs discontiguous word pattern search algorithms. For example, for a read length of 15, you can find all matches with 1 mismatch using the following 3 discontinuous word searches of size 10:

```
111111111100000 word pattern 1
111110000011111 word pattern 2
000001111111111 word pattern 3
```

If the mismatch occurred in the middle, then

ATTTTTTGGGTAGCCCCTTGGATGAGT reference

```
|||||||+|||||||
TTGGGTAACCCCTTG read (15mer)
111110000011111 word pattern
```

This particular word pattern (pattern 2) will ensure a mismatch within bases 6-10 can be identified. By iterating all 3 word patterns, mismatch at every single base can be identified quickly. This example shows how an alignment with read length of 15 and 1 mismatch is performed using 10 base indexing. This combination is called matching schema in SOLiD analysis pipeline. Call this schema_15_1.

SOLiD analysis pipeline has numerous preconfigured matching schemas for various read length and mismatches desired. They are located under $CORONAROOT/etc/schemas/. A partial list follows:

```
schema_25_0 schema_25_1 schema_25_2 schema_25_3 schema_25_4
schema_25_5 schema_25_6
schema_35_0 schema_35_1 schema_35_2 schema_35_3 schema_35_4
schema_35_6 schema_35_7
schema_45_0 schema_45_1 schema_45_2 schema_45_3 schema_45_4
schema_45_5 schema_45_6
schema_50_0 schema_50_1 schema_50_2 schema_50_3 schema_50_4
schema_50_6
```

Map short reads in complex genomes

The color space concept provides an increased ability to map short reads to complex genomes. Complex genomes are difficult to sequence because they contain many repetitive sequences. Mapping short reads to a reference sequence and allowing a few mismatches to account for biological sequence variation and error, often results in ambiguity in mapping. However, similar sequences that differ in a few bases in a genome would result in very different color space sequences. This enhances the ability to map. Conversely, low complexity sequences like dinucleotide repeats lose some complexity in color space and are even more difficult to map. The net effect is a greater ability to map short reads in a complex genome when alignments are done in color space.

Discontinuous word pattern

BLAST (Basic Local Alignment Search Tool) uses keyword searches to speed up alignment. However, this type of search cannot guarantee to find all hits up to a certain number of mismatches, unless a very small word size is used. For example, read lengths of 20 bases requires the word size to be 6 to find all matches up to 2 mismatches. This calculation can be very slow.

Instead of using the BLAST-type of search tool, the SOLiD Sequencing Software uses multiple discontinuous patterns. Each discontinuous word pattern is a 0/1 string of the same read length. Each 0 in the string represents a “don’t care” position in the read, and a 1 represents a position that is part of the keyword. The hash table is built on each sliding window of L bases of the reference sequence, by only taking the number 1 positions in the discontinuous pattern. For example, a pattern of 111011101 determines a sliding window of nine bases, and uses only positions 1, 2, 3, 5, 6, 7, and 9 (a total of seven bases in the window) as keywords.

For a read length of twenty bases, the following 10 discontinuous patterns can be used to find all hits up to two mismatches.

1. 11111111111100000000
2. 11111111000011110000
3. 11111111000000001111
4. 11110000111111110000

5. 11110000111100001111
6. 11110000000011111111
7. 00001111111111110000
8. 00001111111100001111
9. 00001111000011111111
10. 00000000111111111111

If 20 bases of the read are divided into 5 groups of 4 bases, the first pattern picks up hits with one mismatch in group 4 and one in group 5. The second pattern picks up hits with one mismatch in group 3 and one in group 5, and so forth. The hits with 2 mismatches in the same group can be picked up by multiple patterns above. With the SOLiD System's mapping software, 10 runs can be performed of word size 12 (each pattern has 12 ones) instead of one BLAST-type run with word size 6. This is an advantage of using a discontinuous word pattern.

Mapping group rationale

The re-implementation of the mapping pipeline for V3 was focused on scalability and speed without having to make significant changes at the lowest level of mapreads. This allows the software to treat mapreads as something of a series of transformation functions over the space of tag data. What is gained by doing this is flexibility of scheduling and allowing behavior changes based on intermediate results. See the following table.

Pipeline	Description	Behavior for mapping
RepeatClassifier	This is the first of four pipelines in the group that, given a set of reads and a reference sequence, acts as a predictive classifier. It separates the reads into two groups: Repeats and Random. "Repeats" are predicted to be either real repeats or to seed a high number of times when doing genome alignment. "Random" is the set of reads that are predicted to be randomly distributed or unique. This pipeline allows for the two subsequent matching pipelines to be parameterized differently, based on the types of reads expected to be processed. If this pipeline is bypassed (Pass All is on), all reads are classified as random. (This was the behavior in Version 2.0.) This pipeline is useful for eukaryotes resequencing applications.	Always 'ON' Use "Pass All = 1" to bypass
MatchingRepeat	This is a pipeline that processes the reads classified as "repeats". This pipeline should be executed only when the RepeatClassifier pipeline is activated.	Always 'ON' Use "Pass All =1" to bypass
MatchingRandom	This is a pipeline that processes the reads classified as "random". The Matching pipeline is used to search a set of reads against a reference sequence. When no classification is performed, this is the default matching pipeline used (i.e., it performs the matching for all reads). To get Version 2.0 or prior behavior in the pipeline, the RepeatClassifier is bypassed and only this pipeline is selected, with values of length equal to the read length and an upper bound for the number of mismatches.	Always 'ON' Use "Pass All =1" to bypass

MatchingConsolidate	This is the final pipeline in the matching pipeline group and is responsible for pulling together the results of the MatchingRandom and MatchingRepeat pipelines into a single results file for use by downstream pipelines. This pipeline needs to be run any time that matching is run. It provides the file system clean-up that is necessary to ensure that the all interface contracts for downstream pipeline dependencies have been satisfied. This pipeline does no new computation.	Always 'ON'

The inputs to the four parts of the mapping group are the same as in V2 (a .csfasta file) and the outputs are also the same (a single .ma file). The internal behaviors do not matter to the downstream pipelines. A feature of the implementation and dependencies, however, requires that if ANY reference mapping is to be done, ALL four pipelines need to be 'on' even if they do not all execute. This is similar to the filtering pipeline, in that there are assumptions made about the existence of locations in the filesystem during an analysis that each of the component pipelines of the mapping group would create. Setting a 'pass all' in any of the four would cause no computation to be performed but the necessary clean-up to be done.

Considerations for secondary analysis

Highly identical regions

There are many highly identical regions, such as transposons, which are duplicated regions in the genome. This is especially the case in eukaryotic genomes. In fragment runs, the default mapping setting is unique. This means that any reads that hit more than one place on the genome are discarded. Therefore, regions that are highly identical will not get much coverage under this setting. In general, longer and more identical regions (younger repeats and duplications) get less coverage. This can be overcome with mate pairs, especially mate pairs of different sizes (Figure 3). Also, a majority of this type of sequencing gaps can be identified by lowering the threshold of the unique mapping requirement.

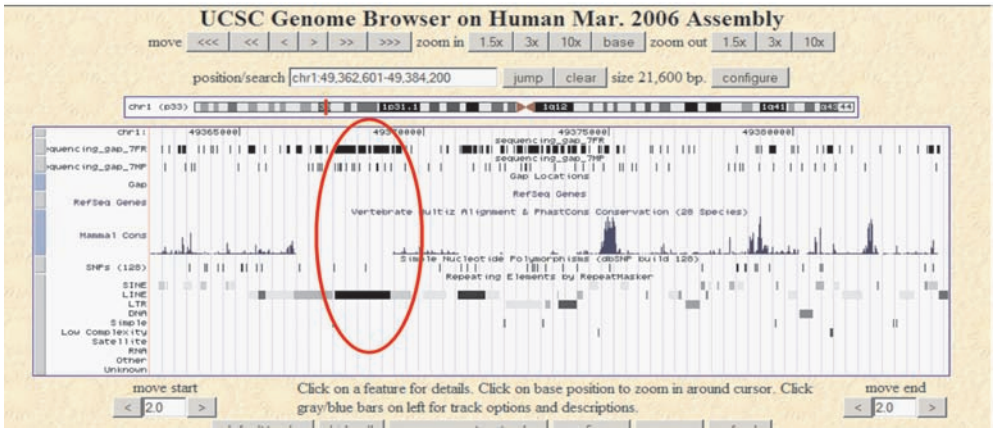


Figure 3 Repetitive elements.

Mate-pairs can identify sequencing gaps caused by repetitive elements by lowering the threshold of the unique mapping requirement.

The circled region is covered by long interspersed nucleotide elements (LINEs) and gets very little coverage from fragment runs. However, the mate-paired run identified a majority of the sequencing gap.

Centromere and telomere

The SOLiD System has an unusually high sequencing coverage close to centromere and telomere regions. These regions are usually highly enriched with satellite repeats and other types of tandem repeats. The reference sequence might have difficulty capturing all copies of these regions. If there is less copy presented in the reference than the actual copies, elevated coverage occurs. The annotation (such as discoveries of SNPs or InDels) in these regions might be less reliable than in other places.

Of course, some highly covered regions have true biological meaning. Consistent reads of over-coverage or under-coverage might represent copy number variation in the sample being sequenced.

Performance

Performance in mapping is directly correlated to the number of reference-based hash operations conducted at the global level and how many times a tag seeds at each hash operation. Wall-clock performance can additionally be gained by adding more processors to the system to allow more concurrent execution across the set of all input tags. To specifically address the first set of performance issues, as the length of the tag increases and the mismatches per tag allowed increase, the number of schemas necessary to cover the reference increases combinatorially. If the total time to place a read is a function of seed count per schema and the reads are segregated into two groups where the seed counts should be substantially different, we can order the schemas appropriately to try and satisfy a tradeoff in sequence tag length and mapping stringency versus time. To address the second set of performance issues, we use an NxM breakup of the mapping process where N is the number of read blocks (or chunks) created and M is the number of reference blocks (splits or chunks) created. Mapping is then the process of executing each schema set for all subsets of the NxM matrix.

Efficient use of resources requires splitting the reference into optimal chunks, and running matching on chunks of reads against the reference chunks. To prevent excessive run times, reads likely to be in repeat regions are segregated from likely unique reads. To fully capture reads capable of mapping, a separate regression plan is used for each class of read (repeat or unique). For each regression plan entry, the same basic pipeline flow is executed:

1. Split input sequences into groups.
2. Match to reference chunks.
3. Consolidate outputs.
4. Increment the ROC curve and decide to iterate or finish. The ROC curve is a Receiver Operating Characteristic or Relative Operating Characteristic that plots the sensitivity vs (1-sensitivity) or the true positive versus false positive level of a system. It can also be thought of as a cost-benefit tradeoff front for a decision process. In the current case, the ROC curve is a cumulative plot of fraction of reads matching the reference against matching stringency. Stringency is the combination of alignment length and maximum number of mismatches required to constitute a match.

5. Subtract aligned reads from input to generate new input, or generate final output.

Separate aligned sets are maintained for each ROC instance to enable user selection of ROC curve stringency cutoff.

Reference sequence breakup

The mapping process utilizes memory at the rate of 1 GB for the 14-mer index and 4 bytes per base of the reference, and then has overhead for each input read, results, and intermediate values. The default behavior for determining the reference chunk size should be approximately 85% of the per-core memory allocation. This is approximately 110 Megabases for the on-instrument implementation and is set by the reference.split.length parameter. Breakup by biological subunit is not used, but the results are realized by creating an index file to match the input multi-fasta file.

An index file is created with the reference name. Each chunk should have an entry in an index file that annotates the chunk number, name, start position within the fully concatenated reference, stop position within the fully concatenated reference and optionally (if known), the biological subunit (i.e., chromosome number) and the start and stop positions within that biological subunit.

The reference chunks overlap by a value that at minimum contains the longest expected read length on the SOLiD system (currently set to 100 via the parameter matching.overlap).

Read breakup

The breakup process is to enable predictability of scheduling in the on-instrument competitive scheduling environment with either real-time user feedback steps, or in the case of near real-time performance necessary for primary analysis. In the case of off-instrument analysis, the breakup generally does not happen unless the number of available processors is substantially greater than the number of reference chunks. For instance, in the case of a 400 node cluster and a reference sequence broken into 20 chunks, the read breakup could be broken up into 20 pieces to saturate the cluster.

ROC analysis

The ROC curve is a Receiver Operating Characteristic or Relative Operating Characteristic that plots the sensitivity vs (1-sensitivity) or the true positive versus false positive level of a system. It can also be thought of as a cost-benefit tradeoff front for a decision process. In the current case, the ROC curve is a cumulative plot of fraction of reads matching the reference against matching stringency. Stringency is the combination of alignment length and maximum number of mismatches required to constitute a match.

If n = the length of the sequence

And k = the number of mismatches allowed

Then the estimated effective length (or stringency) of a given pair of (n,k) can be given by the function:

n - log((n!/((n–k)! k!)) * (3k)) / log(4)

This is an estimated stringency that assumes that all combinations are both equally probable, within the reference sequence as well as within the reads, with a uniform error model over the length of the read. This is just an estimate and is not applicable for non-random genomes and non-random distributions of errors across the read.

RepeatClassifier pipeline

The RepeatClassifier pipeline is part of the Matching Pipeline Group and is new for Version 3. This is the first of four pipelines in the group that, given a set of reads and a reference sequence, acts as a predictive classifier. It separates the reads into two groups: Repeats and Random. “Repeats” are predicted to be either real repeats or to seed a high number of times when doing genome alignment. “Random” is the set of reads that are predicted to be randomly distributed or unique. This pipeline allows for the two subsequent matching pipelines to be parameterized differently, based on the types of reads expected to be processed. If this pipeline is bypassed, all reads are classified as random. (This was the behavior in Version 2.0.) This pipeline is useful for eukaryotic resequencing applications.

The 'Classifier' pipeline works to segregate the input tags into two classes: those with tags that are predicted to be distributed randomly across the reference of interest and those that are predicted to seed with a highly repetitive distribution. The classification takes place BEFORE any mapping and is not made use of by any other pipeline other than those inside of the mapping group. The read subsets and the pipelines that operate on them are treated differently in either their conditional execution or in the order at which combinations of tag length and mismatch counts are used when mapping in a progressive mode. The classification takes three schema chosen to reflect a local, semi-local and non-local tag indexing behavior and indexes the reference. It then passes the reads through the index looking only for seeding behavior in the 'long-tail' of the reference index for each schema and any tag that will seed non-randomly in different combinations of indexing will be classified as a 'repeat' while those that appear to seed with a random distribution are classified as 'random'. The term 'random' does not indicate that the mapping process will yield a randomly mapped position, only that the sampled seeding behavior followed a random distribution. The distinction is made between these two sets for purposes of run time in that tags that seed an inordinate number of times take much more time to extend and check against the reference and the decision could be made to try and map them as quickly as possible rather than place them in every position possible. If the classifier pipeline is bypassed via the 'pass all' parameter, all reads are designated as 'random'. In the case of a single tag (fragment) sequencing run, a repeat sequence may have little utility other than knowing that it occurred since its unique placement can't be made with a short read. In the case of a mate-pair sequencing run, that tag would more likely be placed accurately later when the opposite tag could be used as an anchor around a repetitive region.

MatchingRandom pipeline

The MatchingRandom pipeline is part of the Matching Pipeline Group and is new for Version 3. It is an instance of the Matching pipeline that processes the reads classified as “random”. The Matching pipeline is used to search a set of reads against a reference sequence. When no classification is performed, this is the default matching pipeline used (i.e., it performs the matching for all reads).

Random (nomenclature)

As was previously discussed in the Classifier section, if read classification occurred, the reads in the random pipeline should fit the profile of those that are distributed randomly over the to-be-mapped reference. In the case where classification did NOT occur, all reads regardless of their seeding distribution behavior are in the 'random' pipeline. The parameters in the following table are broken up into blocks that govern different aspects of the pipeline behavior.

To get Version 2.0 or prior behavior in the pipeline, the Classifier is bypassed and only this pipeline is selected, with values of length equal to the read length and an upper bound for the number of mismatches.

Progressive Mapping (nomenclature)

Progressive Mapping is a new feature in SOLiD™ 3 SAT v3.0. It performs iterative mapping to reference sequence with sequential read trimming at decreasing stringencies. The 'ROC Plan for Random Reads' (see below Parameter table) using the value 'random' generates a progressive mapping plan. It is the default behavior in SAT v3.0. The 'ROC Plan for Random Reads' using 'classic' runs the SOLiD V2 behavior of a fixed length and fixed mismatch upper bound.

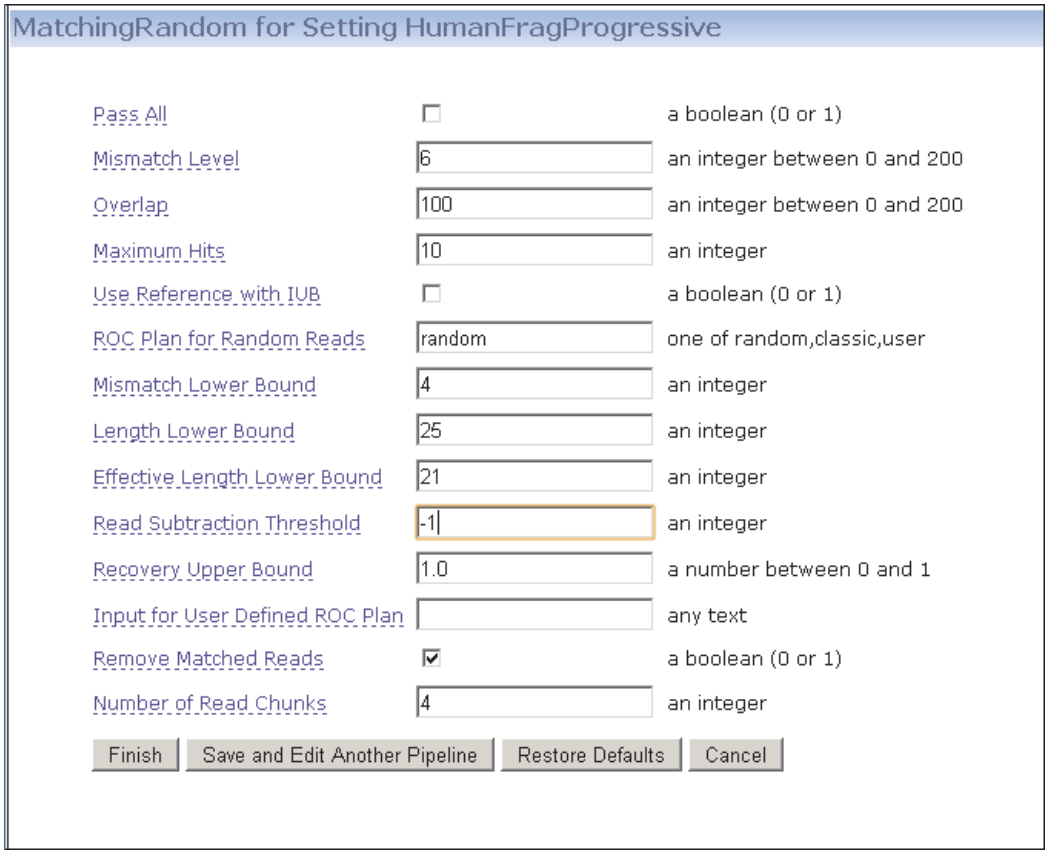
Parameters

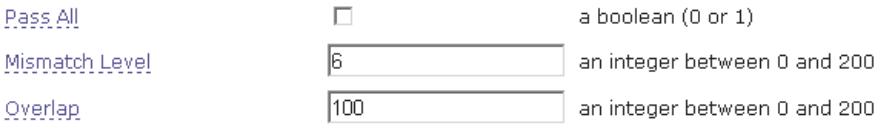
The following table describes the MatchingRandom pipeline parameters:

Para-meter #	Parameter Name and Key	Default Value	Description
1	Pass All matching.random.pass.all	0	Boolean (0 or 1) value that governs whether classified “random” tags should be compared to the reference. (It is unlikely that this should ever be set to 1.)
2	Mismatch Level mismatch.level	6	The number of mismatches allowed when using a classic ROC (Relative Operating Characteristic) Plan. This value is also used in naming the output matching results file.
3	Overlap matching.overlap	100	The overlap length used across reference breaks. Value should be at least twice the read length. For sequencing of 50 bp read length, overlap should be greater than or equal to 100 bp.
4	Maximum Hits matching.max.hits	10	Maximum number of hits per reference chunk.

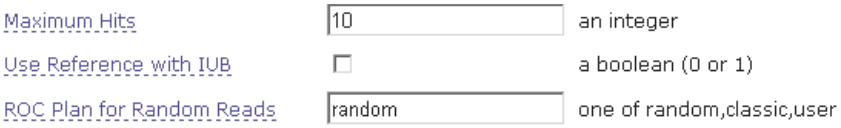
Para-meter #	Parameter Name and Key	Default Value	Description
5	Use Reference with IUB matching.use.iub.reference	false	Map allowing IUB ambiguity codes when a reference containing IUB characters at known SNPs is supplied. A reference with IUB codes does not need to be run with IUB matching, but IUB matching needs a reference with IUB codes. There is a performance decrease of approximately 30% when this is enabled. When this is enabled, the corresponding variable in TagOutput should also be set. If not, the GFF file annotations will not reflect this.
6	ROC Plan for Random Reads matching.roc.plan.random	random	One of the following (random,classic,user), in which “random” is a computed plan governed by the following 4 parameters below defining a variable length mapping that prioritizes fast exact matches over longer read length recover. “Classic” is the traditional fixed length and fixed mismatch used in V2. “User” is a series of mapping steps defined by the user.
7	Mismatch Lower Bound roc.mismatch.lower.bound.random	6	The maximum number of mismatches to allow at ANY length in a variable length mapping plan. This makes sense only if “random” is selected above.
8	Length Lower Bound roc.length.lower.bound.random	25	The shortest sequence allowed to map to the reference. Alternatively, it can be thought of as the shortest length to which a tag can be trimmed prior to mapping.
9	Effective Length Lower Bound roc.stringency.lower.limit.random	21	A computation that takes in a pair of (length,mismatches) and determines the effective length. An entry in a ROC plan that violates this constraint is dropped. A pair (25,0) has an effective length of 25, whereas a pair of (25,3) has an effective length of approximately 17.

Para-meter #	Parameter Name and Key	Default Value	Description
10	Read Subtraction Threshold roc.read.subtraction.random	-1	The minimum number of mismatches at which a sequence should be mapped, before it is subtracted from subsequent ROC plan iterations. For example, a value of 2 means that all reads are aligned at least at length.2 before being subtracted, even if there were a perfect hit at length 0.
11	Recovery Upper Bound roc.cdf.recovery.upper.limit.random	1.0	A value between 0 and 1, for which 0 is 0% and 1 is 100%. The mapping process stops when this percentage of reads has been mapped to the reference, even if there are still entries left in the mapping plan.
12	Input for User Defined ROC Plan matching.roc.user.input.random		length.mm.sub[.va] separated by a comma. It is valid only if “ROC Plan for Random Reads” is set to “user”.
13	Remove Matched Reads roc.read.subtraction.random	true	Boolean (0 or 1) value governing whether matched reads are subtracted after each iteration. For example, if a sequence maps at 50 and 0, it will not be re-attempted at 50 and 2.
14	Number of Read Chunks matching.read.chunks	4	Split the read file into multiple smaller files. The larger this number, the shorter each mapping job takes, but the higher the effort involved in putting the results back together. In an off-line environment with a large reference and no competitive scheduling requirements, this value should be 1. On the instrument with a large reference and 2 concurrent flow cells, this value should be between 10 and 20. For a small reference and concurrent flow cells, a value of 4 is sufficient.



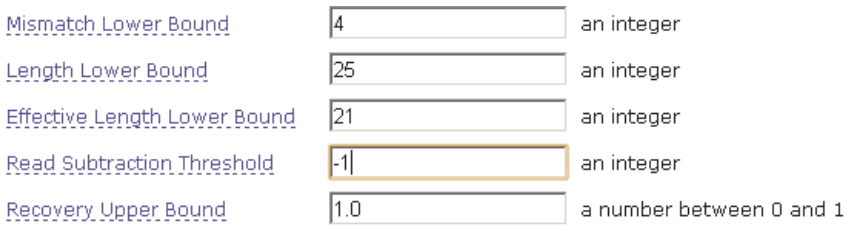


Parameters 1-3 govern pipeline setup. The 'Pass All' parameter has the pipeline execute only the filesystem preparation portions while not doing any actual reference mapping. The .ma file output in this case is the same as the .csfasta file input. The mismatch level is conditionally interpreted -- in the case of a 'classic' plan where a defined tag length and mismatch level are defined, this is the mismatch level used; in the case of a progressive plan (either random or user in this case), this value is interpreted only for purposes of output file naming and does not change any system behavior. The 'overlap' parameter defines the reference overlap when breaking up a large reference genome. This value should be approximately twice the read length.



Parameters 4-6 govern how to map. The 'Maximum Hits' parameter is also known as the 'z' value and it defines the maximum hits reported by mapreads per reference 'chunk'. If the reference is broken up into only 1 piece, this value will be the maximum hits. If the value is set to 10 and the reference is broken up into 10 component pieces, then the actual maximum z value is 100. The boolean of 'Use

Reference with IUB' turns on whether mapreads interprets bi-allelic IUB codes when doing an alignment. It defaults to 'off' because it has a roughly 30% performance hit when on. The 'ROC Plan for Random Reads' can be one of 'random', 'repeat', 'classic', or 'user'. Using the value 'random' generates a progressive mapping plan (example in Random ROC Plan section) in which tags are mapped to the reference in a decreasing stringency. The value 'repeat' generates a progressive mapping plan (example in Repeat ROC Plan in MatchingRepeat section) where tags are mapped to the reference in an increasing amount of time. The value 'classic' runs the V2 behavior of a fixed length and fixed mismatch upper bound. The value 'user' allows the user to specify any series of length and mismatch combinations. However, the user plan will be reordered by stringency if there are violations. For example, specifying a 25.6 followed by a 50.3 is re-ordered because everything that hits in 50.3 is already mapped (not necessarily accurately) at 25.6.



Parameters 7-11 govern how to halt a progressive mapping run. These parameters would not be interpreted in the case of a 'classic' mapping run. The 'mismatch lower bound' is a naming misnomer and should really be an upper bound. This value is the maximum number of mismatches allowed per tag regardless of length. Other parameters will likely determine if this upper bound value is ever actually run. The 'length lower bound' is the minimum contiguous length running from left to right and starting at the first position that will be used in mapping. A sequence will not be trimmed further back than this value. The 'effective length lower bound' is the calculated length based on equation (equation in text now). A combination of mismatch and length will yield an effective length and any combination that violates this value (is lower than this value) will not be run. The 'recover upper bound' is the fraction of the population that will be attempted to map. If 80% (0.8) has been set for this value, the first progressive iteration that maps more than 80% of the population will be the last iteration executed.

Parameters 12-14 govern the mechanics of mapping.

The User-defined ROC Plan is a series of comma-separated entries in the form:

- `[length].[mm].[subtraction boolean].[valid adjacent count]`
  - "Length" is the tag length at which to map
  - *Mm* is the number of mismatches to allow
  - "Subtraction boolean" is whether to subtract reads if they map at this level
  - "Valid adjacent count" is whether to count 'valid adjacents' as 1 or 2 mismatches

"Remove matched reads" governs whether reads are removed at each step of the progressive mapping. If this value is unchecked, the mapping pipeline defaults to the lowest stringency specified.

The 'Number of Read Chunks' is a performance-tuning issue. On-cluster, higher numbers mean that each matching substep will be faster (but there will be more) – blocking for primary analysis or opposite flowcell jobs. Off-cluster, lower numbers mean less overhead between mapping steps with a computed product of the read chunks versus the reference chunks being equal to the number of available processing cores being the most efficient in terms of wall-clock time.

Examples of parameters versus plan lines executed (Random ROC plan)

Given an input of 'Lower Bound Mismatches' of 6, minimum trim length of 25 and effective length of 17, the following 39-step **Random ROC plan** is generated and can be used as a descriptive example in modifying the parameters.

Iteration	**Length**	**MisMatch**	**Effective Length (Stringency)**
1	50	0	50
2	50	1	46.38
3	45	0	45
4	50	2	43.28
5	45	1	41.46
6	50	3	40.49
7	40	0	40
8	45	2	38.43
9	50	4	37.92
10	40	1	36.54
11	45	3	35.72
12	50	5	35.53
13	35	0	35
14	40	2	33.61
15	50	6	33.28
16	45	4	33.23
17	35	1	31.64
18	40	3	30.98
19	45	5	30.92
20	30	0	30
21	35	2	28.8
22	45	6	28.76

Iteration	Length	MisMatch	Effective Length (Stringency)
23	40	4	28.59
24	30	1	26.75
25	40	5	26.37
26	35	3	26.28
27	25	0	25
28	40	6	24.3
29	30	2	24.03
30	35	4	23.99
31	25	1	21.88
32	35	5	21.88
33	30	3	21.62
34	35	6	19.92
35	30	4	19.45
36	25	2	19.3
37	35	7	18.11
38	30	5	17.47
39	25	3	17.03

The following combination of parameters would yield the following run behavior:

Example	Combination of Parameters {mismatch bound}, {trim}, {effective length}	Run Behavior (Lines in above Random ROC plan are run)
1	{4},{30},{21}	1-11,13,14,16,17,18,20,21,23,24,26,29,30,33
2	{6}, {20}, {33}	1 - 15
3	{2},{20},{17}	1-5,7,8,10,13,14,17,20,21,24,27,29,31,36
4	{6},{50},{5}	1,2,4,6,9,12,15

Further, there are parameters that govern the minimum mismatches that need to be examined before a read is subtracted (Read Subtraction Threshold).

Taking Ex 3 above now reads as 3':

Example	Combination of Parameters {mismatch bound}, {trim}, {effective length}, {Read Subtraction Threshold}	Run Behavior (Lines in above Random ROC plan are run)
3'	{2},{20},{17},{2}	4,8,14,21,29,36

At the completion of the pipeline in a progressive run, all outputs of the *N*x*M* process are pooled together to create a single read-ordered output that becomes a final input to the MappingConsolidate pipeline. The intermediate outputs of this step no longer contain any read information. They contain only the mapping information for those reads that map.

MatchingRepeat pipeline

The MatchingRepeat pipeline is part of the Matching Pipeline Group and is new for Version 3. It is an instance of the Matching pipeline that processes the reads classified as “repeats”. The Matching pipeline is used to search a set of reads against a reference sequence. This pipeline should be executed only when the Classifier pipeline is activated.

Repeat (Nomenclature)

As was previously discussed in the Classifier section, if read classification occurred, the reads in the repeat pipeline should fit the profile of those that are distributed non-uniformly over the to-be-mapped reference. These reads show prolific seeding behavior in multiple schemas and therefore have a higher computational cost to map them. In the case where classification did NOT occur, there will be no reads input to this pipeline, however the pipeline should be bypassed ('pass all' set to '1') rather than turned off so that file system modifications can occur. The parameters for this pipeline and their usage are the same as those for the MatchRandom pipeline with minor differences. The default ROC plan is 'repeat' in this case and the default values for mismatch lower bound are lower.

Parameters

The following table describes the MatchingRepeat pipeline parameters:

Para-meter #	Parameter Name and Key	Default Value	Description
1	Pass All matching.repeat.pass.all	1	Boolean (0 or 1) value that governs whether classified “repeat” tags should be compared to the reference.
2	Mismatch Level mismatch.level	6	The number of mismatches allowed when using a classic ROC (Relative Operating Characteristic) Plan. This value is also used in naming the output matching results file.

Para-meter #	Parameter Name and Key	Default Value	Description
3	Overlap matching.overlap	100	The overlap length used across reference breaks. Value should be at least twice the read length. For sequencing of 50 bp read length, overlap should be greater than or equal to 100.
4	Maximum Hits matching.max.hits	10	Maximum number of hits.
5	Use Reference with IUB matching.use.iub.reference	false	Map allowing IUB ambiguity codes when a reference containing IUB characters at known SNPs is supplied. A reference with IUB codes does not need to be run with IUB matching, but IUB matching needs a reference with IUB codes. There is a performance decrease of approximately 30% when this is enabled. When this is enabled, the corresponding variable in TagOutput should also be set. If not, the GFF file annotations will not reflect this.
6	ROC Plan for Repetitive Reads matching.roc.plan.repeat	repeat	One of the following (repeat,classic,user), in which “repeat” is a computed plan governed by the following 4 parameters below defining a variable length mapping that prioritizes fast exact matches over longer read length recover. “Classic” is the traditional fixed length and fixed mismatch used in V2. “User” is a series of mapping steps defined by the user.
7	Mismatch Lower Bound roc.mismatch.lower.bound.repeat	6	The maximum number of mismatches to allow at ANY length in a variable length mapping plan. This makes sense only if “repeat” is selected above.
8	Length Lower Bound roc.length.lower.bound.repeat	25	The shortest sequence allowed to map to the reference. Alternatively, it can be thought of as the shortest length to which a tag can be trimmed prior to mapping.
9	Effective Length Lower Bound roc.stringency.lower.limit.repeat	21	A computation that takes in a pair of (length,mismatches) and determines the effective length. An entry in a ROC plan that violates this constraint is dropped. A pair (25,0) has an effective length of 25, whereas a pair of (25,3) has an effective length of approximately 17.

Para-meter #	Parameter Name and Key	Default Value	Description
10	Read Subtraction Threshold roc.read.subtraction.repeat	-1	The minimum number of mismatches at which a sequence should be mapped, before it is subtracted from subsequent ROC plan iterations. For example, a value of 2 means that all reads are aligned at least at length.2 before being subtracted, even if there were a perfect hit at length 0.
11	Recovery Upper Bound roc.cdf.recovery.upper.limit.repeat	1.0	A value between 0 and 1, for which 0 is 0% and 1 is 100%. The mapping process stops when this percentage of reads has been mapped to the reference, even if there are still entries left in the mapping plan.
12	Input for User Defined ROC Plan matching.roc.user.input.repeat		length.mm.sub[.va] separated by a comma. It is valid only if “ROC Plan for Repetitive Reads” is set to “user”.
13	Remove Matched Reads roc.read.subtraction.repeat	true	Boolean (0 or 1) value governing whether matched reads are subtracted after each iteration. For example, if a sequence maps at 50 and 0, it will not be re-attempted at 50 and 2.
14	Number of Read Chunks matching.read.chunks	4	Split the read file into multiple smaller files. The larger this number, the shorter each mapping job takes, but the higher the effort involved in putting the results back together. In an off-line environment with a large reference and no competitive scheduling requirements, this value should be 1. On the instrument with a large reference and 2 concurrent flow cells, this value should be between 10 and 20. For a small reference and concurrent flow cells, a value of 4 is sufficient.

Examples of parameters versus plan lines executed (Repeat ROC plan)

Given an input of 'Lower Bound Mismatches' of 2, minimum trim length of 30 and effective length of 28, the following 14-step **Repeat ROC plan** is generated and can be used as a descriptive example in modifying the parameters.

Iteration	Length	MisMatch	Effective Length (Stringency)
1	50	0	50
2	45	0	45
3	40	0	40
4	35	0	35

Iteration	Length	MisMatch	Effective Length (Stringency)
5	30	0	30
6	50	1	46.38
7	45	1	41.46
8	40	1	36.54
9	35	1	31.64
10	30	1	26.75
11	50	2	43.28
12	45	2	38.43
13	40	2	33.61
14	35	2	28.8

Combinations of parameters yield the following run results:

Example	Combination of Parameters {mismatch bound}, {trim}, {effective length}	Run Behavior (Lines in above Random ROC plan are run)
1	{1},{30},{28}	1 - 9
2	{2},{35},{30}	1,2,3,4,6,7,8,9,11,12,13

MatchingConsolidate pipeline

There are no parameters to specify for running the MatchingConsolidate pipeline.

To ensure that the output of the mapping pipeline is consistent with V2 so that other downstream consumers of the file will not break, the outputs of the two mapping sub-pipelines are consolidated into a single .ma file output.

Matching Output

The output files are placed in the directory location specified by matching.output.dir. This directory is created if it does not exist.

A file similar to fasta file format is created with the following format for file name:

{csfasta_file_name}.ma.[read.length].[num.mismatch].

Example:

```
File xxxx.ma.50.6
```

This file is the output from the alignment tool containing matching data, with tags the length of 50 bases and with up to 6 mismatches allowed.

Color space fasta file containing matches to the reference sequence:

```
>TAG_ID,LOCATION.MISMATCH
SOLiD
```

```
>1_90_1917_R3
G3113123020101303220331131230201013032203
>1_91_1943_R3,48653.1
G1311303132213331031013113031322133310310
```

The term LOCATION.MISMATCH describes the location of the read on the base space reference sequence (0-based) and the number of errors (mismatches) between the read and the reference sequence, considering the first position in base space and the remaining positions in color space. This is preprocessed data, in which the last base of the primer is prepended to the color space sequence.

Output of .ma file with Progressive Mapping

The .ma file outputs traditionally only indicate how many mismatches occurred at a reference position. The annotation of where and what the mismatches were in the alignment do not occur until the AnnotateChanges portion of 'TagAnalysis' is executed. Now complicating the issue is the progressive nature of the mapping where differing lengths could have been used. This could create the problem where a tag is annotated to have a reference mapped position with 2 mismatches, but it is unclear as to whether this mapped at full length versus a trimmed length. In order to ensure that it is clear what portion of the tag was involved in the mapping and that all downstream pipelines can operate on the assumption that every read contains read.length number of characters, the tags are shortened to have colorcalls only in the mapped positions and then dot-padded from left to right at the point where trimming occurred. This yields the following:

A full 50 color call read:

```
>1_1_1 10.0
01230123012301230123012301230123012301230123012301
>1_1_1 10.0
```

For a mapped 45 length of a 50 color call read:

```
012301230123012301230123012301230123012301230.....
>1_1_1 10.0
```

For a mapped 25 length of a 50 color call read:

```
0123012301230123012301230.........................
```

All three instances are perfect matches. However, they are perfect matches at different lengths and the dot padding reflects this. Internal dots that appear in reads are not from trimming, but are missing data.

TagOutput pipeline

The TagOutput pipeline runs a series of QC and summary reports. This pipeline also creates the GFF V.2 files, which are used by other downstream pipelines. In a MatePair run, this pipeline is implicitly run for each tag by the PairedTag pipeline. The options for this pipeline are primarily used to describe how to create the GFF file.

Parameters

The following table gives the parameters for the TagOutput pipeline.

Parameter Name and Key	Default Value	Description
Output Reads in Base Space single.tag.gff.output.basespace	1	Include corrected base space read in GFF file.
Color Correction Strategy single.tag.gff.option.correctTo	consistent	Specifies how to correct the color calls for output reads in base space. “reference” replaces all read-colors annotated inconsistent (i.e. “a” or “b”) with the corresponding reference color. “missing” replaces all inconsistent read-colors with “.”. These translates to “x” in the base space representation, attribute “b”. “singles” replaces all “single” inconsistent colors (i.e. those annotated “a” or “b” and not adjacent to another 'b') with “.”. Replaces all other inconsistent colors with reference colors. “consistent” - For each block of contiguous inconsistent colors, replaces the lowest QV-value color-call with the unique color that makes the block consistent. Breaks ties at random.
Read Trim Modulus single.tag.gff.option.trimMod	5	Requests AnnotateChanges to trim missing colors (“.”) off the 3′ end of color reads, in blocks of this length. Mapping with variable length plans defaults to trim in increments of 5. These values should remain in accord.

Parameter Name and Key	Default Value	Description
IUB Color Strategy single.tag.gff.option.iubsTo	haplotype	Specifies how to treat ambiguity codes (IUB codes other than A, C, G, or T) in the reference sequence. "missing" treats the code (e.g. "R") as if it were "X". This encodes to a missing color. "first" selects an arbitrary halplotype from the reference by treating each ambiguity as equivalent to the pure base in the set that comes first in the alphabet. This naturally has a bias towards alphabetically low bases. "haplotype" selects a haplotype that minimizes the number of mismatches with the current read.

The following global parameters are also used if available:

- reference
- read.length
- run.name
- sample.name
- f3.primer.base
- r3.primer.base
- reference.analysis.dir
- reports.deploy
- reports.deploy.dir
- colorcall.dir

Outputs

The output files are placed in the directory locations specified by the output directory parameters listed above. If the directory locations do not exist, the system creates them.

In single.tag.reports.dir:

*.html / *.png – These are the HTML and backing images which comprise the reports for single-tag analysis.

In single.tag.errors.dir (directory qc_colorcallError):

F3_positionErrors.txt – Tab-delimited text, one per primer set. Counts errors versus the reference as a function of the position in the read and the type of color substitution which caused the error.

In single.tag.correlation.dir (directory qc_correlation):

F3_autoCorrelation.txt – Tab-delimited text. This is the normalized auto-correlation (Categorical auto-correction) of the color space calls against each other across all of the unfiltered reads. The first two columns specify the two base positions which are compared, the last column is the r coefficient. Examining the auto-correlation spectrum can help inform potential chemistry-related issues such as inefficient cleavage, primer contamination, mixed primers, and upstream library prep problems.

In single.tag.panels.dir (directory qc_panels):

F3_panelStats.txt – Tab-delimited text. This file contains matching summary statistics for every panel that contained at least one read of usable sequence. The panel column is the panel number. The nomatch column is the number of beads which didn't match the genome. The 0 MM column is the number of beads that matched the genome with zero mismatches. There are MM counts for every level of mismatch up to the maximum level of mismatch allowed in this analysis (mismatch.level). The sum of matching column is the number of beads which matched the genome at any mismatch level. The fraction matching is the sum of matching divided by the total number of beads (nomatch + fraction matching).

In single.tag.tagbias.dir (directory qc_tagBias):

F3_tagSignatureBias.txt – Tab-delimited text. This is an analysis that examines the base content of the matched reads, looking for position-specific base imbalance. Such imbalance might be indicative of base-specific biases in various facets of the library preparation. The first column is the position of the read on the original reference sequence, relative to the matched position. The next four columns are the relative base composition at this position, normalized by the total occurrence of that particular base across all of the observed reads. Uniform base composition would correspond to a value of 1.0 for each of the four bases.

matching.stats file

`{run.name}_V1_[F|R]3.[read.length].[num.mismatch].stats.txt`

The text file contains a summary of the matching statistics. “Beads found” is the number of beads initially identified. Many of these are not real beads. Depending on the emulsion dilution, 5-20% of the beads that have DNA on them are doublets that are unlikely to match the genome. Likewise, current single-round enrichment provides more than 90% amplicon positive beads. The amplicon empty beads are usually still found by the bead finder because they have a slight background signal.

Adjacent mismatches (two consecutive mismatches) were split into consistent (valid - a change in color space that was permitted given the initial reference sequence marking a potential SNP) and inconsistent (invalid - a nonallowed change in color space).

The following shows an example of the stats file. You can see this file includes:

- The reference sequence
- Statistics in the context of mapped reads
- Color mismatches corrected in the base sequence
- Possible true base changes

- Adjacent color mismatches corrected in the base sequence
- Analysis parameters, which are added at the end

```
Matching Statistics
SAMPLE_STATS_ST_Sample1_F3.csfasta.ma.20.3

39,650 total tags found
Reference Sequence
ecoli_k12_MG1655, 4,639,675 bp

31,911 Mapped reads using parameter settings listed below.
Mapped Reads at Read Length 20
  0 mismatches        14,706 ( 46.08%)
  1 mismatches         6,927 ( 21.71%)        21,633 ( 67.79%)
  2 mismatches         4,760 ( 14.92%)        26,393 ( 82.71%)
  3 mismatches         5,515 ( 17.28%)        31,908 ( 99.99%)

Uniquely Placed Beads at Read Length 20
  0 mismatches         9,667 ( 30.29%)
  1 mismatches         4,662 ( 14.61%)        14,329 ( 44.90%)
  2 mismatches         3,221 ( 10.09%)        17,550 ( 55.00%)
  3 mismatches         4,186 ( 13.12%)        21,736 ( 68.11%)

Color Mismatches within Uniquely Placed Tags
 Total Mismatches                           23,662
 Non-Adjacent Mismatches                    20,498 (86.63% of Total)
 --before 2-base encoding correction; corrected in base output
 Adjacent Mismatches                         3,164 (13.37% of Total)
   Consistent Adjacent Mismatches            1,296 ( 5.48% of Total)
(40.96% of Adjacent)
   -Base Change
   Inconsistent Adjacent Mismatches          1,868 ( 7.89% of Total)
(59.04% of Adjacent)
   --before 2-base encoding correction; corrected in base output

Color Mismatch Annotations
 Isolated Non-Adjacent Color Mismatches      18,622 (78.70% of Total)
 Consistent with N Adjacent Base Changes
  One                                         1,232 ( 5.21% of Total)
  Two                                         1,028 ( 4.34% of Total)
  Three                                       1,077 ( 4.55% of Total)
  Other Mismatches                            1,703 ( 7.20% of Total)



Starting Points within Placed Tags
 Number of Starting Points  27,599 (0.30% of reference)
 Average Number of Reads per Start Point        1.16

Coverage of Uniquely Placed Tags
 MegaBases of Coverage       0.43
Coverage Per Base          Avg   (SD)
      A                    0.10 (0.36)
      T                    0.09 (0.35)
      C                    0.09 (0.35)
      G                    0.09 (0.35)
    Total                  0.09 (0.35)
Bases Not Covered   4,280,374 (92.26%)
```

```
Coverage of Uniquely and Randomly Placed Tags
 MegaBases of Coverage       0.64
Coverage Per Base          Avg   (SD)
      A                    0.14 (0.44)
      T                    0.14 (0.43)
      C                    0.14 (0.43)
      G                    0.14 (0.43)
    Total                  0.14 (0.43)
Bases Not Covered   4,130,259 (89.02%)

Solid Analysis Tools Parameters
   Run RepeatClassifier = 1
   Pass All (Classifier) = 0
   Run Matching Random = 1
   Pass All (Random) = 0
   Effective Length Lower Bound = 21
   Length Lower Bound = 25
   Mismatch Lower Bound = 6
   Remove Matched Reads = true
   Recovery Upper Bound = 1.0
   Input for User Defined ROC Plan = matchingRocUserInputRandom
   Run Matching Repeat = 1
   Pass All (Repeat) = 0
   Effective Length Lower Bound = 21
   Length Lower Bound = 25
   Mismatch Lower Bound = 6
   Remove Matched Reads = true
   Recovery Upper Bound = 1.0
   Input for User Defined ROC Plan = matchingRocUserInputRepeat
   Mismatch Level = 5
   Use Reference with IUB = false
```

xxx.gff file

Overview of .gff v2 Files

GFF (General Feature Format) is a record-based file format, where each line describes a single *feature* (in this case, a *read*) with a list of tab-delimited *fields* in a fixed order specified by the GFF specification. Because the format has been in widespread use for many years, there are many programs and bioinformatics libraries that support reading and writing to this type of file.

The default file type created by the SOLiD™ System is the .gff v2 file. The particular specialization of GFF used in the SAT pipeline was introduced to adapt the format to represent aligned color-space reads and allow for rapid visualization in a tool like SOLiD Alignment. Here is an excerpt from such a file:

```
3_6_737_F3 corona exon 180614 180632 0.000000-.
g=CGCTAcTATCATCGCGGTA;f=A;s=r6
```

- The first column is the read identifier for this read, with the suffix representing the primer set that this sequence is associated with.
- The second column is a track name for identifying what type of data is represented by this record. We use the track name "corona" to indicate reads from the SOLiD™ instrument.

- The third column is the type of feature represented by this record. The use of the feature name "exon" is a legacy implementation, to be replaced in the future – it is used to provide automatic display of mate-paired reads as multi-exon transcripts).
- The fourth and fifth columns define the start and end position of this read on the genome.
- The sixth column is an optional score, typically 0.0 in the current pipelines, which will be used in the future to denote a quality score.
- The seventh column provides which orientation of the genome was used for this read alignment (either '+' or '-').
- The eighth column represents the translational frame in the GFF format; it is not used in our implementation.
- The ninth column contains key-value data, with key-value pairs separated by ';' and denoted by "key=value".

The keys used in the current implementation are:

- g - provides the double-encoded sequence of the read. Double encoding refers to a representation of the 0,1,2,3 color-space calls as A,C,G,T. If the file has been annotated then lower-case bases represent differences from the reference.
- f - provides the first base (in base space) of this read. This is the base after the primer base, after decoding the first color-space call.
- s - provides color annotations for the SAB viewer. This is a comma-separated list of codes of the form "[ryb]##" where the first letter represents the color ('r'=red,'y'=yellow, etc...) and the number represents the position to be colored in the read sequence.
- c - provides a condensed count of this sequence. If the same read sequence appears multiple times in an experiment, this provides a means for condensing the data and conveying information for how many counts were observed.

The GFF format is convenient for its standardization and widespread integration in genomics tools but it is not an ideal format for efficient storage or I/O access for gigabase sequencing experiments.

Note: The use of double-encoding to represent the reads is a legacy feature. The A,C,G,T characters found in this file do not correspond to actual bases but to the color calls.

Convert to .gff V3 file

Use the command:

```
gffv2_module.sh MainClass
```

Where MainClass is first MaToGff and then AnnotateChanges.

MaToGff

MaToGff is called with the output of the matched .csfasta file in the following manner:

Usage:

```
MaToGff <maFile> [--convert=[unique|mapped|beads|hits]]
[--effective][--first] [--sort] [--ggg3]
[--qvs=<qvFile>] [--tempdir=<dirName>] > gffFile
```

MaToGff converts a FASTA file, maFile, of unmated reads to a SOLiD GFF v0.2 file.

`--convert=unique`

Output only reads that map uniquely to the reference (see `--clear` for definition of unique)

`--convert=mapped`

MaToGff reports only reads, at most one per bead from the ma_file, that mapped to the reference. For each read, MaToGff reports the position of the best alignment to the reference (i.e., the alignment with the least number of mismatches). If MaToGff finds more than one best position, it reports a position at random from a uniform distribution of the best candidates.

`--convert=all`

Deprecated. Use "beads" instead.

`--convert=beads`

MaToGff reports exactly one read for every bead in the ma_file, mapped or not. Mapped positions are selected as in "beads".

`--convert=hits`

MaToGff reports a GFF entry (line) for every map in the ma_file, whether they arise from the same read or not.

Note: This may result in multiple GFF lines per bead.

To uniquely identify a line it will be necessary to specify not just the bead id, but also the start, the strand, and possibly the end (if the reads have variable length). "--convert" defaults to 'unique' if it is not specified on the command line.

`--clear=<cz>`

The requested "clear zone", used when --convert=unique.

Let u_k be the number of alignments ("hits") in which the read maps to the reference with k mismatches. Let k' be the least value k for which u_k != 0. Then the read is said to map uniquely, or simply to be unique, if u_k' = 1, and u_k = 0 for all k in the closed interval [k'+1, k'+cz]. That is, u_k' = 1, and cz is the number of zeroes that must follow k' in u (up to the end of vector u).

cz defaults to 3 if it is not specified.

Consider, for example, a read that maps to its reference with 0 mismatches in just one position, but with 1 mismatch in 5 positions, and with 2 mismatches in another 5. That is, u=1,5,5. This read is unique if cz=0 because k'=0, u_k'=1, and [k'+1,k'+c] = [1,0] is empty.

On the other hand, it is NOT unique if clear > 0 because, although k'=0 and u_k'=1, [k'+1,k'+clear] = [1,1] and u_k = u_1 = 5, not 0.

For another example, if u=0,0,1,0,0,5, then u_k' = u_2 = 1 and the read is unique for cz = 0, 1, or 2, but not for 3. For a final example, u=0,2,0,0,5 is not unique for any value of cz because u_k' = u_1 = 2, not 1 as required.

`--qvs=<qvFile>`

A file of quality values for the color calls in the reads.

`--effective`

Requests MaToGff to interpret the "mismatch" field in a .ma file tag as an "effective mismatch score". It is still an integer but effectively 10 times the number of mismatches.

MaToGff uses these scores as given to determine the "best" alignment for a read, but it sets number of mismatches = round(score/10) when counting and binning.

`--first`

Overrides random selection by causing MaToGff to report the first position instead, that is, to report the alignment with the least "start" position.

`--sort`

Sorts the reads by increasing start position.

`--gff3`

Converts the .ma file to GFF3 format. The default is GFF2.

`--tempdir`

The directory to use for temporary files. MaToGff will delete any temporary files it creates when done, but not the directory.

For each read, MaToGff reports the position of the best alignment to the reference (i.e., the alignment with the least number of mismatches).

If MaToGff finds more than one best position, it reports a position at random from a uniform distribution of the best candidates.

You can override this random behavior by setting the --first option, which causes MaToGff to report the first position instead, that is, to report the alignment with the least "start" position.

Note: FASTA and GFF reads are 2-bp encoded strings.

--tempdir is the directory that MaToGff should use for any temporary files that it might create (defaults to \scratch). It will delete those files, but not the directory, upon termination.

AnnotateChanges

AnnotateChanges reads in a GFF file of color reads and a FASTA file that contains the reference base sequence to which the reads have been mapped. For each read, AnnotateChanges creates the 'r' (reference call at mismatch), 's' (mismatch annotations), and 'b' (base space representation) annotations. AnnotateChanges writes its result (a new GFF file) to Standard Out.

Usage:

```
AnnotateChanges <GFF_file> <FASTA_ref_file>
               [--tints=a[g[y[r]]]] [--b] [--cn]
               [--correctTo=reference|consistent|missing|singles]
               [--trimMod=<k>]
               [--iubsTo=[missing|first|haplotype]
```

Note: AnnotateChanges writes its result (a new GFF file) to Standard Out.

<GFF_File> is the name of the input, unannotated GFF file.

<FASTA_ref_File> is the name of the reference file.

--tints=agyr represents any number (four in this example) single-tint annotations.

- a = Isolated single-color mismatches (red)
- g = Color position that is consistent with an isolated one-base variant (e.g. a SNP)
- y = Color position that is consistent with an isolated two-base varian
- r = Color position that is consistent with an isolated three-base variant

and so on, for any number of adjacent base variants. You can use any tints for a, g, y, and r; it is their position that matters. You do not have to extend as far as 'r', nor do you have to stop there (if biologically realistic). The default is equivalent to -tints=agyr.

--b Requests the 'b' attribute, the base-sequence corresponding to the corrected color calls.

--correctTo=[reference|consistent|missing|singles]

Specifies how to correct the color calls:

- reference = Replaces all read-colors annotated inconsistent (i.e. 'a' or 'b') with the corresponding reference color.
- missing = Replaces all inconsistent read-colors with '.'. These will translate to 'x' in the base space representation, attribute 'b'.
- singles = Replaces all 'single' inconsistent colors (i.e. those annotated 'a' or 'b' and not adjacent to another 'b') with '.'. Replaces all other inconsistent colors with reference colors.
- consistent = For each block of contiguous inconsistent colors, replaces the lowest QV-value color-call with the unique color that makes the block consistent. Breaks ties at random.

The default is equivalent to --correctTo=reference.

--cn Requests AnnotatedChanges to print the names of the contigs in the reference file. Format is, for example:

```
##contig 5 The name of the fifth contig.
```

--trimMod=<k> = Requests AnnotatedChanges to trim missing colors ('.') off the 3' end of color reads, in blocks of k. For example, if k=5, the read=[3321.....] is trimmed to [3321], but [3321...] is "trimmed" to [3321...].

--iubsTo=[missing|first|haplotype]

Specifies how to treat ambiguity codes (IUB codes other than A, C, G, or T) in the reference sequence:

missing = Treats the code (e.g. 'R') as if it were 'X'. This would encode to a missing color. E.g. color(CR) = color(RC) = color(CX) = '.'

first = Selects an arbitrary halplotype from the reference by treating each ambiguity as equivalent to the alphabetically-first pure base in the set. E.g. SAAWWTCNVTH => CAAAATCAATA. This will naturally have a bias towards alphabetically low bases.

haplotype = Selects a haplotype that minimizes the number of mismatches with the current read. E.g. ref=SAAWWTCNVTH with read=G2033320312 would yield GAATATCCGTC, which encodes as G2033320312, and has no mismatches, while ref=SAAWWTCNVTH with read=G0000000000 would yield GAAAATCCCTT, which encodes as G2000320020, and has 4 mismatches.

Specification of .gff v2 Files

If a '#' character appears anywhere on a line, the rest of that line is a *comment*. Ordinarily, an application reading the file ignores comments and *comment lines* (lines starting with '#' after white space). Comment lines that begin with '##' are called *metadata*, data that apply to all of the features in the file. These metadata, defined in Section 3.1, need not be ignored by all applications.

Metadata

The GFF format allows for the use of metadata in the form of '##' comment lines, including the following standard GFF metadata tags:

##solid-gff-version

This is currently "`##solid-gff-version 0.3`".

##gff-version

This is "`##gff-version 0.3`".

##source-version <source> <version text>

A comment describing the version of the program that produced this file. There should be no spaces in the source and the version text. For an acceptable example:

```
##source-version Ma_to_gff.java v0.2
```

##date <date>

The date that the file was generated, in yyyy-mm-dd format. This example designates 1 October 2008:

```
##date 2008-10-01
```

##time <current time>

The local time, in 24-hour format, when the generating software wrote this line. For example,

```
## time 18:02:36
```

##Type <type>[<reference name>]

The type is set to **solid_read**. The reference name is the name of the reference to which all reads in this file were aligned.

##history <command line>

These comment lines allow recording of source information in addition to the program name, which is recorded by the ##source-version tag, and to record how previous processing steps lead to the data in the current SOLiD GFF v0.2 file. There is one history line for each processing step, with earlier steps appearing above later ones. It may not always be feasible to record all aspects of a command line, such as its file indirections, for example.

```
##history map Sample1_F3.csfastamyReference.fasta
##history Ma_to_gff.java --convert=unique –sort Sample1.ma
##history AnnotateChanges.java Sample1.gff myReference.fasta
##history filter_fasta.pl --noduplicates
--output=qvs.qual
##history AddQvs.java Sample1.gff qvs.qual
```

##color-code <code string>

This line specifies the color code used to generate *all* color reads in this file. The code-string is a comma-separated string of <motif>=<code> pairs. For example, the following entry specifies our current two-base encoding:

```
##color-code
AA=0,AC=1,AG=2,AT=3,CA=1,CC=0,CG=3,CT=2,GA=2,GC=3,GG=0,GT=1,TA=3,
TC=2,TG=1,TT=0
```

##primer base <code string>

A comma separated string of <primer set>=<base> pairs, each of which specifies the last base for each primer set in this file. For example,

```
##primer_base F3=T,R3=G FieldsThe feature lines specify one read
each, with the following fields, as specified in the GFF standard.
```

Fields

The feature lines specify one read each, with the following fields, as specified in the GFF standard.

`seqname` Field

The name is the bead identifier (panel_x_y) plus a suffix indicating the primer set id (either "_F3" or "_R3" currently). For example:

```
8_1727_1389_F3
```

`source` Field

This name is always `solid`.

`feature` Field

This name is always `read`.

`start` Field

The inclusive start point of the aligned read on the 1-based base-space reference sequence, indexed from 5′ to 3′. See ‘end’ for details.

`end` Field

The inclusive end point of the aligned read on the 1-based base-space reference sequence, indexed from 5′ to 3′.

By the GFF specification, start must be less than or equal to end. For example, if the read aligns to the forward strand of the reference (see strand field below), as in:

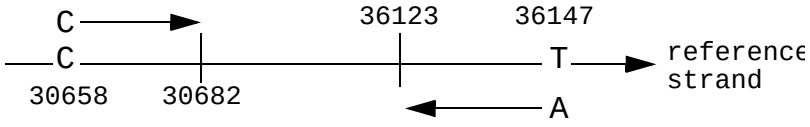
```
start    end      strand   [attributes]
30658    30682    +        g=C0103112003130213233110 32
```

The initial ‘C’ aligns with position 30658 of the reference strand, which should be a matching ‘C’; the following ‘0’, which corresponds to another ‘C’ (by the color code in §3.1.7), aligns with position 30659; and (the base corresponding to) the last ‘2’ aligns with position 30682. See the diagram below.

If, on the other hand, the read aligns to the reverse strand of the reference, as in:

```
start    end      strand   [attributes]
36123    36147    -        g=A3221210033103102321030 22
```

Then the A aligns with position 36147 of the reference strand, which should be a complementing T; the first ‘3’ after the initial ‘A’, which corresponds to a ‘T’ by the color code, aligns with position 36146 of the (forward) reference strand, and the last ‘2’ aligns with position 36123.



`score` Field

A summary quality score for the read, recorded to one decimal place:

$$score = -10 \log_{10} P$$

where P is the average probability of error at any read position:

$$P = \frac{1}{n} \sum_{i=1}^{n} 10^{-QV_i/10}$$

and QV_i is the quality value of the color at read position i (see **`q attribute`**).

Note that 0 < score < 1, and that high scores correspond to high-quality reads.

`strand` Field

Either '+', if the read aligns to the "forward," "nominal," or "given" strand (that is, the sequence in the database), or '-' if the read aligns to the other strand.

`frame` Field

SOLiD does not use the frame and therefore always sets it to ".".

`attributes` Field

This is a semi-colon separated list of key-value pairs of the form "key=value". This implementation is slightly different than GFF v2 suggests (the ACEDB format), although the implementation of this field is somewhat free.

- **`b`** Field Attribute (optional)

  The corrected base-space representation of the read. Constructed in three steps:

  1. Identify isolated and invalid mismatches (see **s**-attribute definitions) in the read, then replace them with the aligned reference color-calls.
  2. Convert the result to a DNA sequence and align it with the DNA reference.
  3. Replace mismatched nucleotides with the appropriate IUB code.

- **`g`** Field Attribute

  Required. The color space sequence for this read, written from 5′ (the bead end) to 3′. In addition, prepended to the sequence is the first (5′ most) nucleotide of the DNA target corresponding to the read. This attribute, together with the coding algorithm specified by the color-code metadata tag, allows any application to reconstruct the DNA sequence for the read.

  In addition, by referring to the 'strand' metadata tag (and assuming no errors in the data), the application can reconstruct the color-call subsequence and the DNA subsequence for either strand of the reference.

  Note: It is important to note that FASTA files generated by the instrument store their data in a different format. There, the sequence stores the last base of the primer, presumably the phase 5 primer, as its first base.

  Fortunately, given the color-code tag, it is easy for any application to convert FASTA notation to GFF notation by reconstructing the first base of the read.

  For example, the application converts the FASTA sequence T210033221 to the GFF sequence C10033221 by converting T2 to TC and dropping the primer base T.

  The first color reported by 'g' is really the second color in the read from the instrument.

- **`i`** Field Attribute

  Optional. The 1-based index of the reference sequence. For example, i=3 says that this read is aligned to the third reference sequence in the reference name (see the ##type metadata tag). If this value is not specified, it defaults to 1.

- **p** Field Attribute

  Optional. This mappability measure counts the "effective number of hits" for this read onto the reference. A small value close to 1 indicates that the read is effectively unique in the reference. A higher value indicates that the read effectively matches at multiple locations. More formally, the field attributes have these definitions:

  L Length of the read.

  k Number of mismatches in an alignment. **Note**: $k \leq L$.

  m Number of mismatches in *this* alignment.

  N_k Number of reference positions where the read aligns with exactly ***k*** mismatches. **Note**: $N_k = 0$ for all $k < m$ (for an explanation, see "end Field" on page 118).

  X_{Lk} The expected number of alignments with exactly *k* mismatches as a multiple of the expected number of alignments with 0 mismatches.

  Then

  $$X_{Lk} = 3^k \binom{L}{k} \text{ and } \textit{mappability} = X_{Lm} \sum_{k=m}^{L} \left(\frac{1}{X_{Lk}} N_k \right)$$

  In the following examples, let L=25: For a read with exactly one alignment with no mismatches, m=0, and $N_m = 1$. Then $X_{L,m} = X_{25,0} = 1$ and *mappability* = 1, indicating an ideal unique match.

  Even if the single alignment has two mismatches, then L=25, m=2, N_m=1, and mappability = 1 (the $X_{L,m}$s cancel out).

  For two perfect alignments, L=25, m=0, and $N_m = 2$. Again, $X_{L,m} = X_{25,0} = 1$. But now mappability = 2, indicating an ambiguous match. In general, if the read maps perfectly to the reference in n places, the mappability measure is also n.

  A final example shows how more complicated values arise: if $N_0 = 0$, $N_1 = 3$, and $N_2 = 25$, then mappability = 3.694. The three alignments with one mismatch contribute 3 and the 25, with two mismatches contribute the remaining 0.694 to the measure.

- **q** Field Attribute

  Optional quality values. A comma-separated list of quality values, one per color-call. Each quality value is an integer between 0 and 100, exclusive. The value -1 indicates a missing quality value for the corresponding color-call.

- **r** Field Attribute

  Optional reference call at mismatch. A comma-separated list of {position}_{ref_color} for all of the color-calls in the reference sequence that differ from the read sequence. Position is 1-based relative to the sequence specified in the "g" attribute (again, the prepended base has position 1 and the first color in 'g' has position 2.). This differs from the basechange format in that only the reference sequence call is provided and positions are 1-based and positive.

  For example, `r=18_3, 21_1` means that the reference value is 3 at position 18, and 1 at position 21.

- **s** Field Attribute

  Optional. This is a comma-separated string representing annotations on the sequence. The format is '{char}{position}' where {char} is a character representing the type of annotation (typically a formatting request to visualization software) and {position} is the position of this annotation in the read. The position is 1-based on the string recorded in the 'g' attribute. That is, the prepended base has position 1, and the first color has position 2. For example, "r5,y7,y8" means "format base 5 red, format base 7 yellow, and format base 8 yellow." SOLiD follows the convention that:

  – **r** (red) is an *isolated mismatch*; it is a mismatch, and neither the color-call on its left nor the call on its right is a mismatch.

  – **y** (yellow) is a *valid adjacent mismatch*; it is a mismatch and it, together with the adjacent mismatch on its left or right, could correspond to an isolated SNP.

  – **b** (blue) is an *invalid adjacent mismatch*; it is any mismatch other than one specified by red or yellow.

- **u** Field Attribute

  Optional. Mismatch count. This is a comma-separated list of non-negative integers. The i th number specifies the number of positions in the reference to which this read aligns with exactly i – 1 mismatches. For example, u=0,3,15 says that this read does not align to the reference anywhere with exactly 0 mismatches, but that it does align with one mismatch at 3 reference positions, and with exactly two mismatches at 15 reference positions. All unspecified mismatch counts are undefined. This example gives no information about the number of reference positions where the read aligns with exactly four mismatches.

Additional Semantics

Mate-paired data

For mate-paired reads, the reads are represented on two adjacent lines with the F3 read followed by the R3 read.

Line Order

The GFF standard allows lines in any order ("line order is not relevant"). The standard GFF files produced by the SOLiD system, however, should have metadata before feature lines, and feature lines should be sorted by increasing start position of the reads. In the case of mate-paired data, the reads should be sorted by increasing start position of the F3 read, and the F3 read and R3 read should remain adjacent.

Example

```
#gff-version 2
##source-version AnnotateChanges.java v0.2
##date 2007-08-28
##time 09:30:18
##Type solid_read DH10B_WithDup_FinalEdit
##color-code
AA=0,AC=1,AG=2,AT=3,CA=1,CC=0,CG=3,CT=2,GA=2,GC=3,GG=0,GT=1,TA=3,TC=2
,TG=1,TT=0
##generated-by "AnnotateChanges.java test-etc/modules/temp.gff test-
etc/modules/DH10B_WithDup_FinalEdit_validated.fasta
```

```
##hdr seqname     source   feature  start    end      score  strand      frame          [attributes]    [comments]
389_495_172_F3   solid    read     906843   906867   -1 +   g=T21210330303022011230133 3;r=18_0,19_1,24_2;s=yl
                                                            8,y19,r24;u=0,0,0,1
389_595_202_F3   solid    read     912290   912314   -1 -   g=T31132111321003213030012 0;u=1
```

This is an example of single-primer data which conforms to the specification described above. The reads have been annotated with reference sequence discrepancies, style annotations, and uniqueness counts. The example also shows a convenience “header” line (##hdr), which illustrates that not all comments, not even metadata, need come from this specification.

GFF v2

The GFF specification upon which this file is based can be found at http://www.sanger.ac.uk/Software/formats/GFF/GFF_Spec.shtml .

MatePair pipeline

MatePair overview

The MatePair pipeline involves a process called *pairing rescue* in addition to aligning F3 and R3 reads to reference. If the F3-R3 pair satisfies the orientation, order, and distance constraints, it is successfully paired. If the F3-R3 does not satisfy the constraints, then one of the tags is anchored to the reference, and the mapping is performed again for the other tag with more mismatches allowed in the region of the reference decided by the mate-pair insert size range. The total combined numbers of mismatches for F3 and R3 tags are still the same. If we allow mismatches up to m for F3 and m for R3, then the total combined mismatches are $2m$. In the rescue process, if the mismatch for one of the tag is n ($< m$), then the mismatches allowed for the other tag is ($2m – n$). Suppose we allow up to 3 mismatches for the alignment and one of the tags had 1 mismatch, we allow up to 5 mismatches for the other tag in the pairing rescue process within the distance, orientation and strand constraints.

MatePair analysis

Using discontinuous word patterns allows the SOLiD System to run mapping software very fast. However, because there is a fixed read length, the run time grows exponentially large in the number of mismatches, when the number of mismatches allowed is large compared to the read length. For a read length of 25 base pairs, if the software allows 0 mismatches, it requires a run time of X. If the software allows 1 mismatch, it requires a run time of 1.5X. If it allows 2 mismatches, it requires 3.5X. If it allows 3 mismatches, it takes 8.5X. If it allows 4 mismatches, it requires 35X. Therefore, the exact number of mismatches must be verified.

For mapping of a fragment library, the software considers sequences that can be mapped uniquely. In these applications, the user can set Z=2, which means finding up to 2 hits for each read. If a decision about uniqueness is made based on the difference between the most significant and the second most significant hit, then the limits for a large genome need to be set higher, to approximately 100. A read that has many hits is likely to be in the repeat region. It should be noted that setting this limit is important for the run time of the mapping algorithm.

For a mate-paired library, the 2 tags in a pair should be from the same strand of the reference sequence, and at a certain distance away from each other (see Figure 2). The Z limit in mapping can be set lower because mate-pair rescue can be performed when at least one of the reads (in the pair) is not in a repeat region. For example, if a user sets Z=10 for the MatePair library, and one of the tags (F3) in a pair is in the repeat region and the other tag (R3) is not, then the mapping software reports 10 hits for F3. However, the list does not include all possible hits. If the software reports 5 hits for the R3 tag, and these include all possible hits for that read, then the mate-pair rescue program tries to pair the hits together within the correct distance and orientation. If the hits are used to the read R3 tag and search for tag F3 in the nearby region, the correct position of F3 is found (see Figure 4). This strategy enables a faster speed for mapping MatePair data.

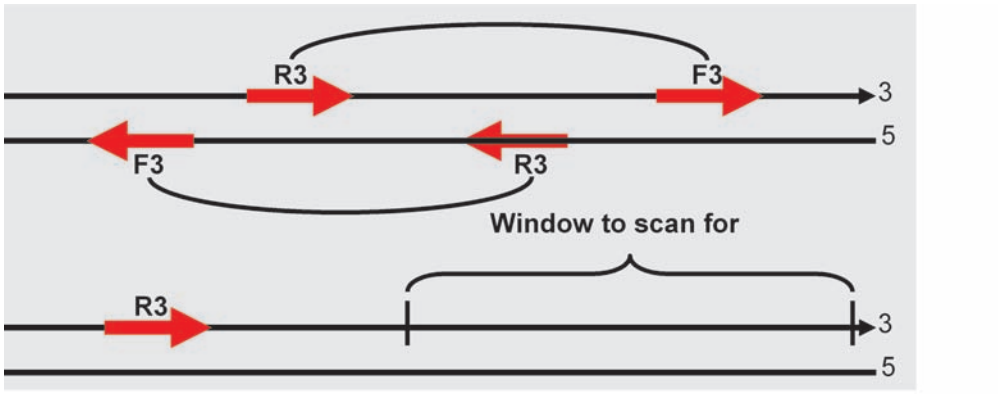


Figure 4 Schematic of MatePair data.

The tags in a pair are on the same strand on the reference sequence and are the correct distance from each other. If one of the tags (R3 in the figure) aligns to a section of the reference sequence, the position of other tag can be easily determined.

The rescue routine also allows relatively more mismatches, so that the software can locate more matches for mate-paired reads.

One of the limits of pairing/rescue is that it relies on finding correct matches for at least one tag. This means that the correct placement is in the list of reported hits for the tag. The software cannot always find all the hits for both tags. When both tags are in the repeated region, there is a possibility that the placement of both tags is not mapped correctly. The software might therefore report a wrong tag placement, not find a good pairing, or find multiple good pairings.

Applied Biosystems is working on several alternative approaches to solve this problem. A simple conservative approach is to throw away pairs of tags that have more than a certain number of hits for both tags. On average, this approach produces a small percentage of the total reads when both tags are in the repeat region. Another approach is to create a secondary mapping step specifically for these types of reads that will find all possible hits.

Parameters

The following table gives the MatePair pipeline parameters.

Parameter Name and Key	Default Value	Description
Minimum Insert insert.start	1800	Minimum insert size that defines a good mate.
Maximum Insert insert.end	3200	Maximum insert size that defines a good mate.

Parameter Name and Key	Default Value	Description
Rescue Level mate.pairs.rescue.level	10	Usually two times the mismatch level.
Use Pairing mate.pairs.use.pairing	true	
Pairing Mismatch Threshold gff.clear	1000	
Output Reads in Base Space mate.pairs.gff.output.basespace	1	Include corrected base space read in GFF file.
Color Correction Strategy mate.pairs.gff.option.correctTo	consistent	Specifies how to correct the color calls for output reads in base space. “reference” replaces all read-colors annotated inconsistent (i.e. “a” or “b”) with the corresponding reference color. “missing” replaces all inconsistent read-colors with “.”. These translates to “x” in the base space representation, attribute “b”. “singles” replaces all “single” inconsistent colors (those annotated “a” or “b” and not adjacent to another “b”) with “.”. Replaces all other inconsistent colors with reference colors. “consistent” - For each block of contiguous inconsistent colors, replaces the lowest QV-value color-call with the unique color that makes the block consistent. Breaks ties at random.
Read Trim Modulus mate.pairs.gff.option.trimMod	5	Requests AnnotateChanges to trim missing colors (“.”) off the 3′ end of color reads, in blocks of this length. Mapping with variable length plans defaults to trim in increments of 5. These values should remain in accord.
IUB Color Strategy mate.pairs.gff.option.iubsTo	haplotype	Specifies how to treat ambiguity codes (IUB codes other than A, C, G, or T) in the reference sequence. “missing” treats the code (e.g. “R”) as if it were “X”. This encodes to a missing color. “first” selects an arbitrary halplotype from the reference by treating each ambiguity as equivalent to the pure base in the set that comes first in the alphabet. This naturally has a bias towards alphabetically low bases. “haplotype” selects a haplotype that minimizes the number of mismatches with the current read.

The following global parameters are also used if available:

- reference
- reference.de
- read.length
- run.name
- sample.name
- f3.primer.base
- r3.primer.base
- reference.analysis.dir
- insert.start
- insert.end
- reports.deploy
- reports.deploy.dir
- colorcall.dir

Outputs

The outputs of the MatePair pipeline are very similar to the outputs of the FragOutput pipeline, with the addition of several analysis outputs that make sense only in the context of MatePair resequencing. These output files are placed in the directory locations specified by the output directory parameters listed above. These locations are created if they do not exist.

The outputs in the `mate.pairs.error.dir`, `mate.pairs.correlation.dir`, `mate.pairs.panels.dir`, `mate.pairs.coverage.dir`, `mate.pairs.tetrad.dir`, `mate.pairs.reports.dir`, and `mate.pairs.tagbias.dir` directories are analogous to the outputs in the FragOutput pipeline described above, with the extension that there exists a separate analysis file for each tag (primer set).

In addition, the `mate.pairs.distances.dir` directory contains:

- **`mateDistances.txt`** – Tab-delimited text. This file contains a single column of insert sizes in base pairs, one for each mate pair placed on the reference genome.

The `mate.pairs.reports.dir` directory contains additional files specific to MatePair analysis, as well as the assortment of HTML and .png files associated with both FragOutput and MatePair analysis:

- **`F3_R3_mates.report`** – Tab-delimited text. This file contains matching statistics on how many F3 reads match the genome versus R3 reads at various levels of mismatch tolerance. Missing tags and reads that do not match are also reported. For a MatePair experiment, this report provides a comprehensive summary of what happened in the matching of each tag to the genome. The census report summarizes this report in an HTML table.
- **`annotateBadMates.report`** – Tab-delimited text. This file contains statistics counting how well the aligned mate pairs fell within MatePair constraints (expected insert size, expected orientation with respect to each tag and the genome). Each mate pair is counted as a member of one of nine categories, depending on which constraints are satisfied. The census report summarizes this report in an HTML table.

F3_R3.mates

This file reports the paired tags (as a single line) and the category that the pairing falls into. The column headings are:

- Bead
- F3 Sequence
- R3 Sequence
- Number of F3 Mismatches
- Number of R3 Mismatches
- Total Mismatches
- F3 Position
- R3 Position
- Category

##beadId	F3 sequence	R3 sequence	num F3 mismatches	num R3 mismatches	total mismatches	F3 reference	R3 reference	F3 position	R3 position	category
871_155_262	T120003211020 1130012330113	G232313323120 3202120032323	4	2	6	1	1	-881697	-883861	AAA
871_162_200	T203301011011 3200213033103	G031001003102 1132331001120	1	5	6	1	1	959439	957353	AAA
871_172_339	T202032102313 0000330010031	G012032112311 2302113330223	0	2	2	1	1	-578934	-580466	AAB
871_184_319	T003221010120 0211100112220	G001301332213 2202120222131	1	2	3	1	1	1491524	1490124	AAB
871_186_359	T301203100201 0211302331320	G221121122232 2310031121312	2	3	5	1	1	-169719	-172123	AAA
871_191_59	T030120020222 2233301200202	G123003010113 0132131032230	0	6	6	1	1	-1152795	-1155407	AAA
871_194_145	T301202300112 2003332123200	G333201223232 1000330320122	3	3	6	1	1	1085862	1084574	AAB

Note: *Category* refers to the orientation and alignment of the tags. It consists of three values: XYZ.

Table 6-3 Mate-pair descriptions for the Category Column Heading

# Mate-Pair XYZ	Description
AAA	Correct orientation + ordering and acceptable insert size
AAB	Correct orientation + ordering and small insert size
AAC	Correct orientation + ordering and large insert size

Table 6-3 Mate-pair descriptions for the Category Column Heading *(continued)*

# Mate-Pair XYZ	Description
BAA	Incorrect orientation and acceptable insert size
BAB	Incorrect orientation and small insert size
BAC	Incorrect orientation and large insert size
ABA	Correct orientation, incorrect ordering, acceptable insert size
ABB	Correct orientation, incorrect ordering, small insert size
ABC	Correct orientation, incorrect ordering, large insert size
C**	Different references

- The first value refers to orientation A = correct orientation; Both tags are on the same strand and read in the same direction. B = incorrect orientation; Tags are *not* reading in the same direction relative to one another.

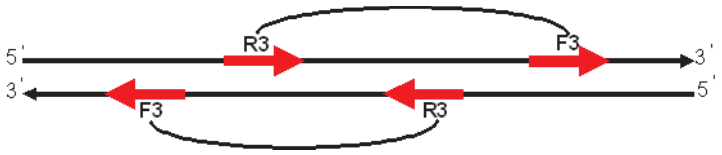


Figure 5 F3 and R3 showing correct orientations

- The second value refers to correct ordering. A = correct order; that is, reading from 5′ to 3′, the R3 read is first and the F3 is second (see above). B = incorrect order.
- The third value refers to the insert size (distance on reference genome between R3 and F3).
  - A = correct (within set parameters) insert size
  - B = smaller than expected size.
  - C = larger than expected insert size

IMPORTANT! These values are relative to the reference and therefore an individual mate-pair may not be truly good or bad. There might have been some structural variation relative to the supplied reference.

The TAG_ID of tags that match on the reverse strand are appended with _RC. The color-space data are preprocessed and therefore, the first base is the last base of the primer. Distance is the insert size between the paired tags. Bead errors is the sum of the errors in F3 and R3.

ErrorModel pipeline

The ErrorModel pipeline constructs a run-specific empirical error model by looking at each k-mer for a bias where that model can be used by downstream pipelines.

Parameters

There is one parameter that is specific to the ErrorModel pipeline. It is good.mate.definition, the Good Mate Definition, and it has a default value of “none”.

The following global parameters are also used if available:

- reference
- reference.de
- read.length
- run.name
- sample.name
- f3.primer.base
- r3.primer.base
- reference.analysis.dir
- is.mates
- insert.start
- insert.end
- reports.deploy
- reports.deploy.dir

Outputs

Output files are placed in the *error.prone.output.dir* directory. The directory is not created if it does not exist.

Three files are found in this directory. Errors for the base positions by the first 5 and 6 (5 + first inosine) positions and then errors in the inosine bases only (6,7,8).

```
qc_sequenceError/F3_123456_12encoded_errFreq.txt
qc_sequenceError/F3_12345_12encoded_errFreq.txt
qc_sequenceError/F3_678_12encoded_errFreq.txt
```

CoverageBias pipeline

The CoverageBias pipeline looks at the sequence coverage bias in comparison to the reference sequence.

Variation in coverage

A key advantage of the SOLiD™ 3 System is that it provides a high coverage over the genomic region of interest. However, the coverage is not always even across the genome. In general, mate-paired runs have a more even coverage across the genome than fragment runs because of the ability to use pairing information to map both reads correctly. If the variance is caused only by sampling, the ideal coverage would follow a binomial distribution. In reality, the binomial distribution is often not a perfect fit to the observations.

This variation in coverage can result from several factors: sample preparation, sample sequencing, the ligation chemistry during sequencing, and genomic features of the reference genome. Different sample preparation methods might result in a difference in starting concentration, which translates to differences in coverage. The samples being sequenced might also cause coverage variation. For example, more sequence coverage is observed for mitochondria and less for the X and Y human chromosomes, because of different amounts of genetic materials in the sample. Genomic features in the reference genome might result in greater-than-expected variation in coverage because of several factors.

GC content

Because of hybridization efficiency, very high GC (lower hybridization specificity) and very low GC (lower hybridization sensitivity) regions often have less coverage than normal GC content regions. In addition to the pure GC factor, a mild drop in coverage over CpG island regions has been observed (Figure 6).

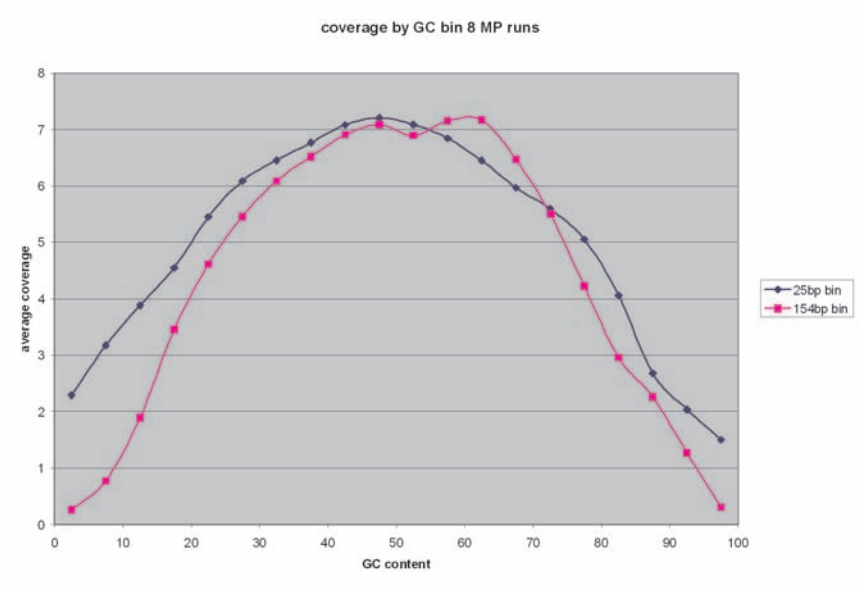


Figure 6 Average coverage binned by GC content from mate-paired runs.

Parameters

There is one parameter that is specific to the CoverageBias pipeline. It is coverage.bias.window, the size in bases of the CoverageBias window, and it has a default value of 50.

The following global parameters are also used if available:

- reference
- read.length
- run.name
- sample.name
- reference.analysis.dir
- is.mates
- reports.deploy
- reports.deploy.dir

In addition, one of the two variables – *single.tag.coverage.dir* or *mate.pairs.coverage.dir* – must be set (see above).

Outputs

Output files are placed in the *coverage.bias.output.dir* directory. This directory is created if it does not exist.

- **`binned_coverage_masked.txt`** – Tab-delimited text. This file is similar to the *fwd_rev_coverage.txt* file, but the actual coverage has been replaced by a coverage category, determined by adaptively binning the observed coverage. The adaptive binning attempts to construct ten coverage bins, each containing a

similar number of reference bases. The no-coverage bin is treated as a special case, so eleven bins are determined in the general case. The header of this file contains details on the bin choices. Coverage is only examined at locations in the genome that are sequenceable (uniquely placeable) for the given read length.

- **`[ACGT]_frequency_vs_coverage_masked.txt`** – Tab-delimited text. This file examines the forward and reverse coverage across the genome, using the coverage levels determined in the *binned_coverage_masked.txt* file, and reports a histogram of the local base composition for all of the bases covered by that coverage level. There are four files, one for each of the bases.

  Note: The sequenceability determination used in creating the coverage mask does not take into account the possibility of MatePair coverage. MatePair allows reference sequences to be significantly covered, but it is difficult to construct an automatic (and deterministic) judgment of MatePair sequenceability because of the random variables involved (mean insert size and standard deviation).

In directory *single.tag.coverage.dir*:

- **`fwd_rev_coverage.txt`** – Tab-delimited text. Each line in this file corresponds to a base in the reference sequence. The first column is the number of reads covering this base on the forward strand (5′→ 3′). The second column is the number of reads covering this base on the reverse strand (3′→ 5′).
- **`coverageHist.txt`** – Tab-delimited text. Histogram data for the coverage across the reference. The first column is the level of coverage. The second column is the number of reference bases covered with this depth of coverage.
- **`condensedCoverage.txt`** – Tab-delimited text. The same data as fwd_rev_coverage.txt but adaptively binned to fit the reference into 10,000 evenly spaced bins. The coverage in each bin is determined by averaging across the bin.

Variation Detection pipeline

Find Single Nucleotide Polymorphisms (SNPs)

SNPs are single base-pair mutations that usually consist of two alleles. Color space rules allow the software to detect only valid SNP sites. According to the color space rules, one isolated color change is always an error when you consider single nucleotide changes. Two adjacent color changes can be either an error or a valid SNP. To detect two adjacent SNPs, look for three adjacent color changes. There are very specific rules to determine which color changes are valid and which are not.

Color space rules for SNP detection

In the simplest case, where you have a single base change, the following rules apply (Figure 7):

1. For any given reference, there are only three valid double-color changes.
2. For the other 12 possibilities, the 6 single-color and 6 two-color changes are invalid, since they would require multiple changes within that genomic region. For these complicated genomic changes, the software uses a more sophisticated algorithm.

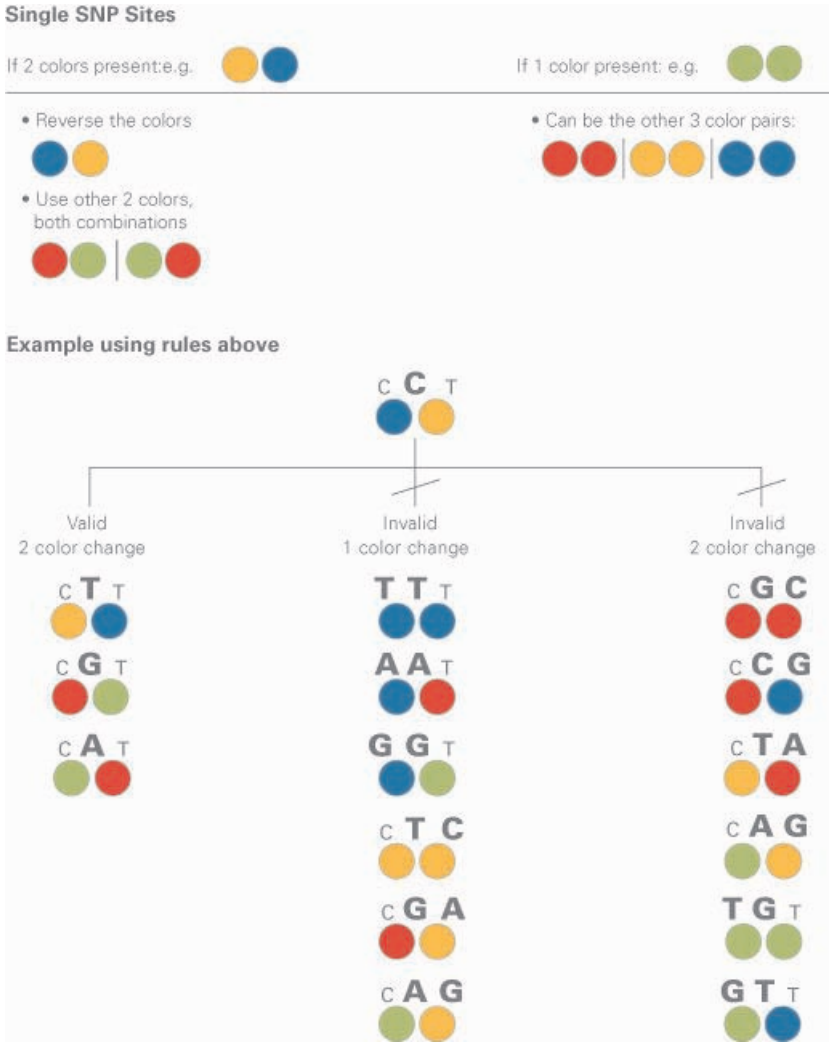


Figure 7 Color Space rules for Single Nucleotide Polymorphism (SNP) sites.

Color space rules are automatically applied during data analysis using the SOLiD System's software analysis tools. You do not need to apply these filters manually.

SNPs and errors

Because a SNP generates two adjacent and valid color space mismatches to the reference sequence, the software allows a minimum of two color space mismatches during alignment to the reference sequence. Depending on the complexity of the alignment, the optimum number of mismatches in alignment is at least three. If only two errors are allowed, then whenever there is a SNP (two mismatches) and a measurement error, the third error causes omission of that read. You can override the omission by using an alignment program setting that counts two adjacent mismatches as a single mismatch. Therefore, two adjacent mismatches and a measurement error are classified as two errors, whereas three non-adjacent errors are classified as three errors.

Higher accuracy for SNP detection

Two-base encoding provides higher system accuracy and built-in error checking because it enables discrimination between measurement errors and true sequence polymorphisms. The SOLiD™ software interrogates each base twice in two independent reactions. Information about each base is included in two adjacent pieces of color space data. Because the system uses four fluorescent dyes, there are 16 possible two-color combinations (Figure 8).

	Blue	Green	Yellow	Red
Blue	BB	BG	BG	BR
Green	GB	GG	GY	GR
Yellow	YB	YG	YY	YR
Red	RB	RG	RY	RR

Figure 8 Color Space transitions.

A SNP must have two adjacent color changes. Single-color mismatches are not evidence of a SNP. This feature allows SOLiD to distinguish errors in measurement from true SNP's.

SNP error rates

When you use two-base encoding, any SNP in the original sequence is represented as two adjacent mismatches in color space. Only three of nine possible adjacent mismatches can correspond to a real SNP. Suppose the raw sequencing error rate for a species is 1% per base, and for the same species that is sequenced, the SNP rate is about 0.1%. For a sequencing project of M total bases, there are about $0.001M$ real SNP occurrences in the data set. Among these SNPs, $0.001M \times 0.02$ total cases appear as single mismatches or two invalid mismatches because a sequencing error happens at one of the alleles of the SNP. Only 2% of the real SNPs fail to appear as two adjacent, valid mismatches. If only two adjacent, valid mismatches are treated as candidates for SNP detection, then there is a 2% false negative rate. For any data set, two adjacent and valid mismatches can be caused by sequencing errors. However, for a total of M bps sequenced, there are $0.00003M$ total occurrences of two adjacent mismatches. Note that there are about $0.001M$ adjacent valid mismatches from real SNPs, among all two adjacent, valid mismatches observed in a particular data set. Of these, 97% are from real SNPs, and only 3% may be from sequencing errors. This is a 97% true discovery rate for that particular data set. Because there are about $0.01M$ total sequencing errors in the data set caused by two-base encoding and the software's ability to remove all single base mismatches, the error is reduced from $0.01M$ to $0.00003M$. This is a reduction of 300 times, making the effective error rate 0.003%. This calculation illustrates the power of two-base encoding in resequencing and finding SNPs.

Sampling

If coverage at a genome position is low, the second allele might often not be sampled, even if it is present. Heterozygotes are expected to be underrepresented at low coverage. (They can be called as a homozygote for one of the two alleles.) It is important to have a low false positive rate for SNP detection, because SNPs are expected to exist at only 1 in 1,000 positions for humans and some other species. The false positive rate in an individual species should be at least an order of magnitude lower than this, to avoid a high false discovery rate. An error model that incorporates qualityvalues (QV) can increase the overall accuracy of SNP detection. The SNP detection algorithms of the SOLiD™ 3 System use explicit error models generated from each run and biologically meaningful prior probabilities to evaluate evidence of a SNP. Two-base encoding provides a built-in way to distinguish errors from true alleles to manage the false positive rate.

Allele ratio

If the sample preparation method or some other cause produces results in allele ratios that are considerably different from the expected 50:50 ratio, it is more difficult to detect heterozygosity. Detecting heterozygosity then requires more sequence coverage.

Find polymorphisms in color space

The SOLiD™ 3 System has the capability to detect complicated genomic variations such as adjacent SNPs, insertions, deletions, and structural rearrangements. For these genomic variations, you can download SOLiD™ bioinformatics tools from the website http://solidsoftwaretools.com. An example of various polymorphisms in color space is shown in Figure 9.

Figure 9 Examples of polymorphisms in Color Space.

diBayes SNP Detection pipeline

The SOLiD™ 3 SAT v3.0 comes with diBayes SNP detection pipeline and it is optional. The diBayes SNP Detection application performs SNP detection using filtering and a Bayesian algorithm to evaluate the probability that a heterozygote or non-reference homozygote exists at this position. It takes mapped and processed SOLiD System colorspace reads and accompanying quality values (in the form of a GFF file), along with the reference sequence, and error information on the SOliD slide, and calls SNPs. A consensus .fasta file is created, along with a list of SNPs and a list of all positions. There are two programs (scripts) included: diBayes_input, which takes the GFF file and creates a large number of input files; and diBayes, which takes the quality files, reference files, and error files, to call SNPs.

For the diBayes application, the following input files are required:

- Reference Sequence: This is the .fasta file against which the reads were aligned.

- F3-position-error-file: This is a tab-delimited text file created by the SOLiD System software during secondary analysis. There is also R3-position-error-file if this is a MatePair run.
- F3-probe-error-file: This is a tab-delimited text file created by the SOLiD System software during secondary analysis. There is also a R3-probe-error-file if this is a MatePair run.
- Quality files: These are binary files created by diBayes_input.
- Quartile-file: This is created by diBayes_input and contains information about the median coverage and the inter-quartile range of the coverage.

The following table gives the required usage parameters. The program is called automatically with the appropriate parameters from the pipeline. If you want to run diBayes SNP Detection stand-alone, the following table gives the parameters that you can use. The order of parameters in the command line is not important.

Note: You need to run the diBayes_input program BEFORE you run the diBayes program.

diBayes Required Commands	Short Command	Description	Type/Range
--reference-file	-f	Reference File	Full path to reference sequence fasta file
--f3-position-error-file	-s	F3 position error file	Full path to the F3 position error file: for example, error_position_matrix_f3.dat (Corona) or F3_positionErrors.txt (SOLiD Software).
--f3-probe-error-file	-u	F3 probe error file	Full path to the F3 probe error file: for example, F3_12encoded_6merContext .txt .
--pos-qv-file-prefix	-q	Pos quality value file prefix	Full path to “quality” files. Quality files are created by the gfftoqv program, which must be run first.
--file-index-first	-F	File index first	Integer: file number of first quality file to analyze. Quality files are created by the gfftoqv program, which must be run first.
--file-index-last	-L	File index last	Integer: file number of last quality file to analyze. Quality files are created by the gfftoqv program, which must be run first.
--output-dir	-o	Output directory	Full path to location of the directory in which to place final output files. This must be created in advance with write permission.

--working-dir	-w	Working directory	Full path to location of the directory in which to place temporary working files. This must be created in advance with write permission.
--experiment-name	-n	Experiment name	Text: output file names will have this prefix.
--read-length	-l	Read Length	SOLiD read length for the longest reads in the study.
--quartile-file-name	-r	Quartile File Name	Full path to a file containing information about the coverage quartiles for this study. Quartile files are created by the gfftoqv program, which must be run first.

The following table gives the optional usage parameters. The order of the parameters in the command line is not important.

diBayes Optional Commands	Short Command	Description	Type/Range
--call-stringency	-c	Call Stringency	Options: default, high_coverage, med_coverage, low_coverage. Med_coverage is more aggressive at calling SNPs at lower coverage. low_coverage is the most aggressive for calling SNPs.
--r3-position-error-file	-t	R3 position error file	Full path to the R3 position error file (required for a mate pair run).
--r3-probe-error-file	-v	R3 probe error file	Full path to the R3 probe error file (required for a mate pair run).
--poly-rate	-p	Polymorphism rate	Expected frequency of heterozygotes in the population: for example, 0.001 in humans.
--write-fasta	-W	Write Fasta	1= Write the Fasta file. 0= Do not write the Fasta file.
--help	-h	Help	Lists and gives an example of input parameters
--version	-V	Print Version	Displays the current version

The following table lists additional filters and algorithm tuners. These individually override filter parameters that were set by the call-stringency parameter.

diBayes Optional Commands – Filters	Description	Type/Range
--both-strands	Require that the non-reference allele be present on both strands.	0 = Do not require, 1 = Require
--min-coverage	Require at least this coverage to call a heterozygous SNP.	Integer, 1-n
--min-coverage-2nd-snp	Require at least this coverage for the less common allele.	Integer, 1-n
--check-snp-evenly-distributed	T-test to test whether both alleles are evenly distributed on both strands. The position will not be called heterozygote if alleles are distributed differently. Useful filter for high coverage data.	1= Check 0= Do not check
--half-reads-snps	Proportion of the reads containing either of the two candidate alleles. Filters positions with high raw error rate.	Real, 0-1
--min-unique-start-pos	Minimum number of unique start positions required to call a heterozgyote	Integer, 1-n
--min-allele-ratio	The less common allele must be at least this proportion of the reads of the two heterozygote alleles.	Real, 0-1 99 = no minimum

The reference sequence .fasta file might contain A, T, C, G; it might also contain heterozygous IUB codes. If it contains IUB codes, heterozygotes are called with fewer reads providing evidence at these positions, because the prior probability of a heterozygote existing at this position is higher.

Three output files are produced by diBayes. These are:

1. A list of all SNPs detected in the sample compared to the reference. This file is in GFF v3 format, a tab-delimited text format compatible with USCS browser and other programs.
2. A .fasta file containing the new consensus sequence (SNPs only), which has exactly the same dimensions as the reference.
3. A list of all positions with coverage, with information about the reads covering each position, the consensus call, the p-value, and more. Note that diBayes produces one consensus calls file per CHROMOSOME per sample.

The SNP file contains every position at which the reference contains a heterozygous IUB code, whether the consensus sequence for that position is homozygote or heterozygote.

diBayes usage example

diBayes can be called via the GUI by the software. Below is an example of how diBayes is called at the command line. Note that this example assumes that the preliminary program, diBayes_input, has already been run to create the input files from the GFF file, namely the files `MyExperiment_F3_quality_200000_1.bin to MyExperiment_F3_quality_200000_5.bin` in the example below:

```
./diBayes -f
'//home/mutation/Run Files/S_suis.fasta' –s '//home/mutation/Run
Files/F3_positionErrors.txt' –u '//home/mutation/Run
Files/123456_12encoded_F3_6merContext_err_vs_control.txt' –q
'//home/mutation/MyExperiment_F3_quality_200000' –F 1 -L 5 -w test -o
test -n 'MyExperiment' -l 50 -r 'afile'
```

qc_colorcallError

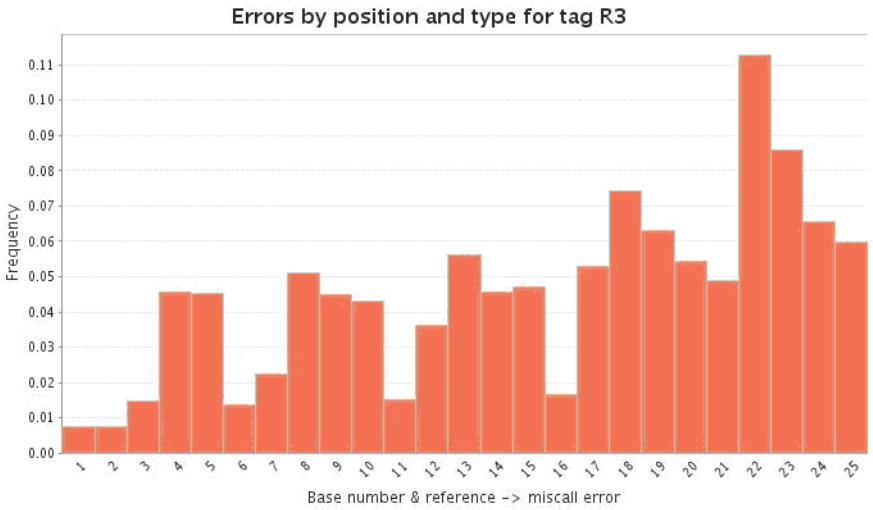
`[F3|R3]_positionErrors.txt`

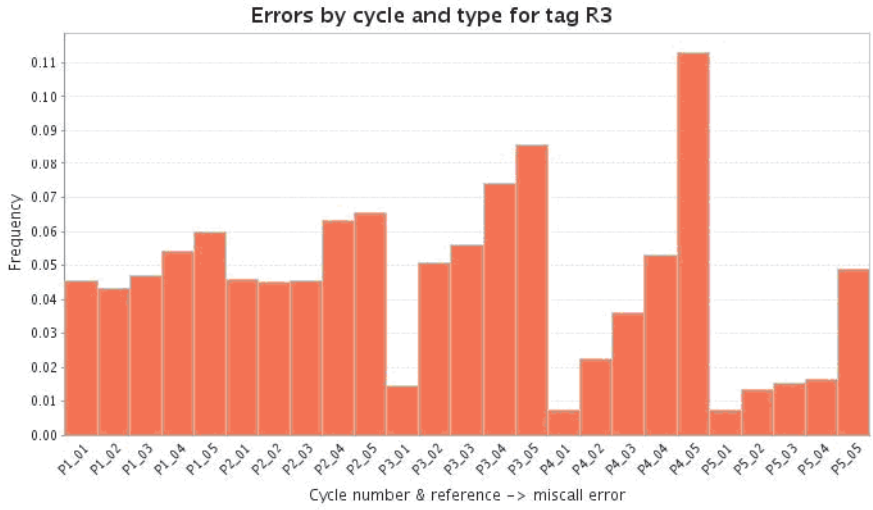
The table below shows a portion of the file. This file is a tab-delimited text with one per primer set. The errors were counted against the reference as a function of the position in the read and the type of color substitution that caused the error. The errors in the first position are counted using the base space information after the color space calls were converted to bases using the last base of the primers and 2-base encoding rules. All other positions used color space information after converting reference sequences into color space sequences.

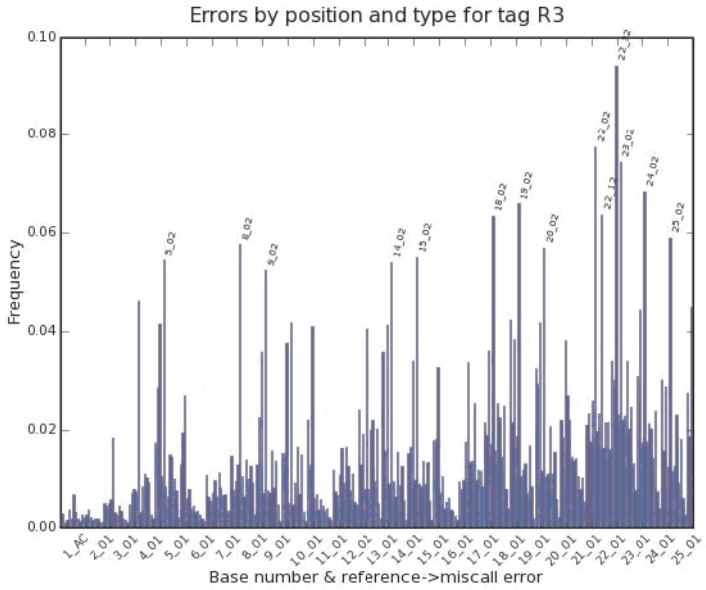
##position_[refCall][readCall]	nErrors	nOccurences	Error_frequency
1_AC	16163	14409991	0.0011
1_AG	22391	10089311	0.0022
1_AT	42772	26824498	0.0016
1_CA	30649	11142534	0.0028
1_CG	23839	10089311	0.0024
1_CT	41913	26824498	0.0016
1_GA	23536	11142534	0.0021
1_GC	25712	14409991	0.0018
1_GT	35000	26824498	0.0013
1_TA	65535	11142534	0.0059
1_TC	94061	14409991	0.0065
1_TG	95474	10089311	0.0095
2_01	25437	17046122	0.0015
2_02	34689	14731838	0.0024

2_03	14055	13974582	0.0010
2_10	24717	16713668	0.0015
2_12	28159	14731838	0.0019
2_13	35053	13974582	0.0025
2_20	35833	16713668	0.0021
2_21	11124	17046122	0.0007
2_23	27695	13974582	0.0020
2_30	63076	16713668	0.0038
2_31	31582	17046122	0.0019
2_32	19832	14731838	0.0013

There are 2 figures in SETS for this data after aggregating each position: one for base sequence order, the other for ligation cycle order. There are also figures for the error profiles that can be found in qc_reports directory, eg, R3_positionErrors.png. In this .png figure, the data are not aggregated for each position. See the following figures.


Errors by position and type for tag R3
Frequency
Base number & reference -> miscall error


Errors by cycle and type for tag R3
Frequency
Cycle number & reference -> miscall error


Errors by position and type for tag R3
Frequency
Base number & reference->miscall error

Reports

This section desribes the various reports that the SOLiD™ system produces. These reports include:

- qc_correlation
- qc_coverage
- qc_mates
- s_matching

For additional information about how to generate and interpret reports, refer to the *Applied Biosystems SOLiD™ 3 System - SETS Software Getting Started Guide* (PN 4389302).

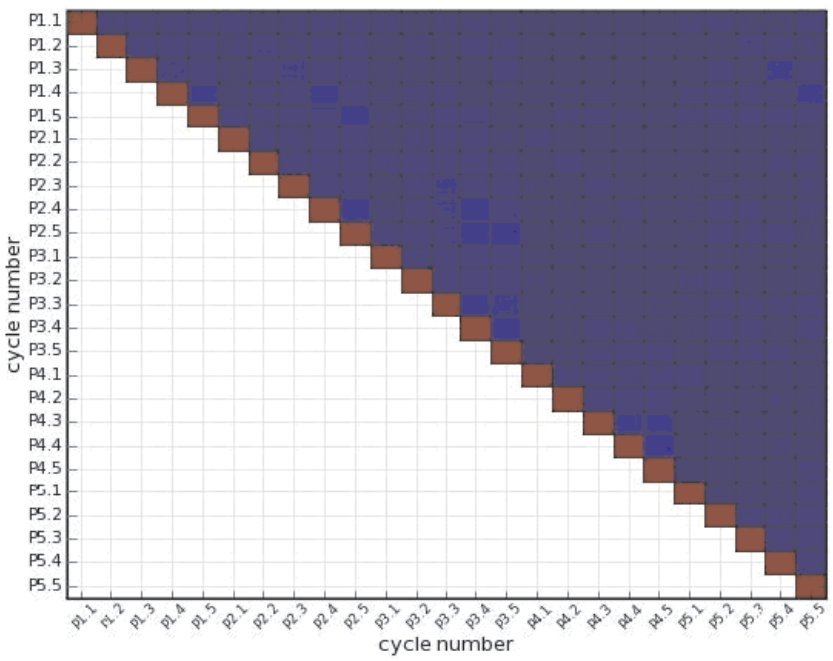
qc_correlation

[F3|R3]_autoCorrelation.txt

This is a tab-delimited text file that contains the normalized autocorrelation (Pearson's r coefficient) of the color space calls against each other across all of the unfiltered reads. The first two columns specify the two base positions which are compared. The last column is the r coefficient. Examining the auto-correlation spectrum can help with potential chemistry-related issues, such as inefficient cleavage, primer contamination, mixed primers, and upstream library prep problems.

##base i	base j	r	##base i	base j	r
1	1	1			
1	2	-0.03212	2	2	1
1	3	-0.01456	2	3	0.002179
1	4	0.009035	2	4	-0.02094
1	5	0.020838	2	5	-0.00559
1	6	0.007623	2	6	-0.00094
1	7	0.005358	2	7	0.016312
1	8	0.016748	2	8	0.012216
1	9	0.010649	2	9	0.005957
1	10	0.009629	2	10	0.006822
1	11	0.013002	2	11	0.0093
1	12	0.006531	2	12	0.016709
1	13	0.012063	2	13	0.007058
1	14	0.008445	2	14	0.008213
1	15	0.011563	2	15	0.006961

##base i	base j	r	##base i	base j	r
1	16	0.014065	2	16	0.006784
1	17	0.005943	2	17	0.018084
1	18	0.014338	2	18	0.008424
1	19	0.015109	2	19	0.00809
1	20	0.01251	2	20	0.008241
1	21	0.02167	2	21	0.004993
1	22	0.005822	2	22	0.020764
1	23	0.010909	2	23	0.01149
1	24	0.015417	2	24	0.008226
1	25	0.015648	2	25	0.007681



qc_coverage

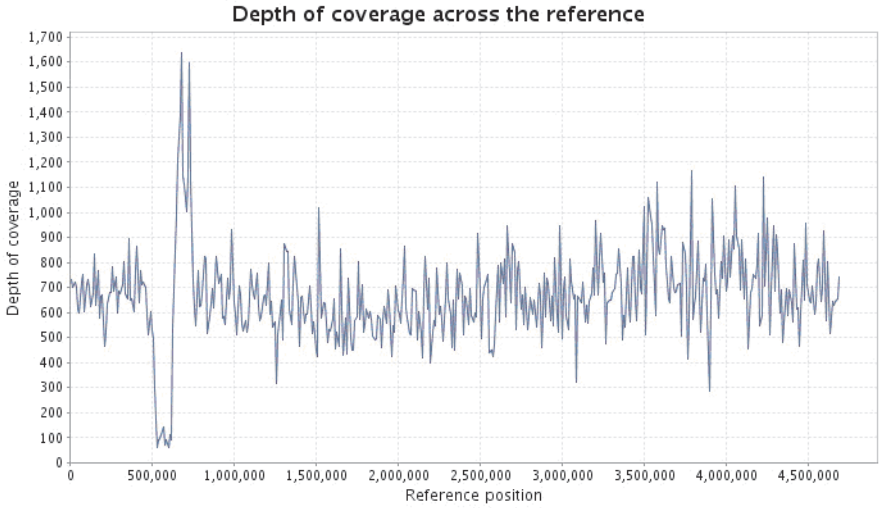
fwd_rev_coverage.txt

This is a tab-delimited text file. Each line in this file corresponds to a base in the reference sequence. The first column is the number of reads covering this base on the forward strand (5′-> 3′). The second column is the number of reads covering this base on the reverse strand (3′-> 5′).

Here is a sample portion of this file:

```
74 118
```

```
75 118
75 128
76 128
76 125
76 123
76 123
76 118
76 116
76 116
76 110
76 110
76 102
```



condensedCoverage.txt

This is a tab-delimited text file that contains the same data as fwd_rev_coverage.txt, but now adaptively binned to fit the reference into 10,000 evenly spaced bins. The coverage in each bin is determined by averaging across the bin. Here is a sample of this file:

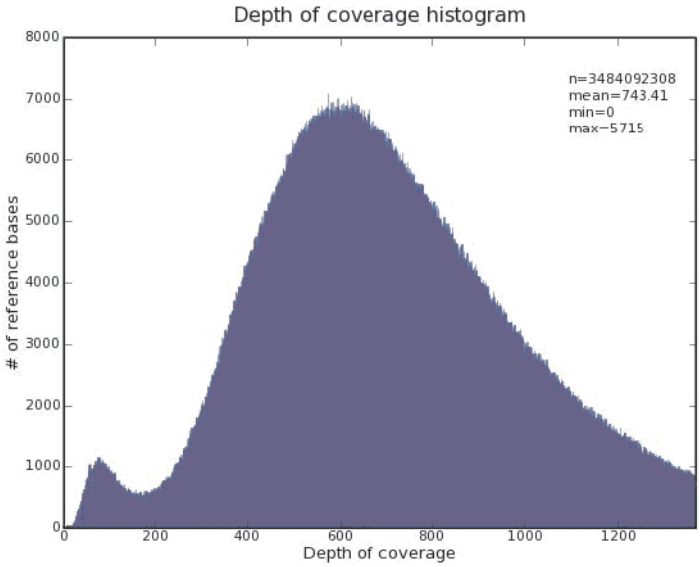
```
# 01/19/08 07:24:08
5000    361.71 368.27
15000   352.28 347.42
25000   361.45 360.79
35000   352.17 348.10
45000   305.62 306.73
```

coverageHist.txt

This is a tab-delimited text file containing the histogram data for the coverage across the reference. The first column is the level of coverage. The second column is the number of reference bases covered with this depth of coverage. Here is a sample of this file:

```
##bin count
0.000000 3.000000
1.000000 19.000000
.
.
.
```

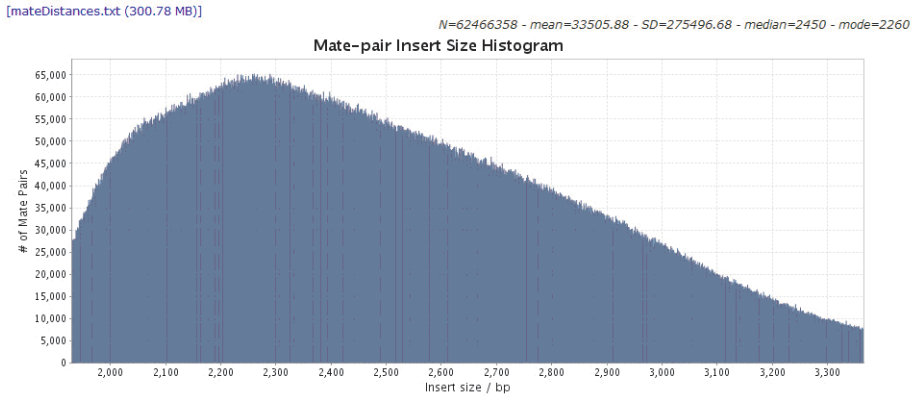
```
43.000000 796.000000
44.000000 846.000000
45.000000 866.000000
46.000000 904.000000
```



qc_mates

mateDistances.txt

This is a tab-delimited text file containing a single column of insert sizes in base pairs, one for each mate pair placed on the reference genome.



s_matching

{run.name}_{sample.name}_[F3|R3].csfasta.ma.[read.length].[num.mismatch]
e.g., S0024_20071217_2_Sample1_R3.csfasta.ma.25.3

The file contains alignment results produced by the Matching pipeline. The following is a portion of the alignment results.

```
>20_32_691_R3,-12136.0
G3132131110101112212211303
>20_32_894_R3,-2078988.3
G2331130322201132233121232
>20_32_1362_R3
G3120113012133323323131312
>20_32_1388_R3,2692764.2
G3133302123010101322233323
>20_33_242_R3,681011.2,567751.2
G3102222323301111132300211
```

Off-line Data Analysis 7

Introduction

The SOLiD™ 3 System allows primary analysis results to be exported automatically to an off-line computer cluster. This arrangement permits secondary analysis to be performed outside of the instrument. This practice allows the instrument to be used for the next experiments.

The exported data consists of the following files:

- csfasta (color space sequence in fasta format)
- _QV.qual (quality value file)
- .stats (statistics information)
- binary data file in .spch format

To provide an integrated, seamless off-line analysis, you can purchase a compatible off-line analysis cluster from Applied Biosystems.

This chapter discusses the following topics:

- Requirements to perform off-line data analysis
- Comparison of SAT and Corona_Lite for off-line data analysis
- Recommended computer hardware configuration for off-line data analysis
- Overview of off-line cluster site preparation
- Checklists for off-line cluster site preparation

For customers who use an Applied Biosystems off-line computer cluster, please refer to the document *Applied Biosystems SOLiD™ 3 System Site Preparation Guide* (PN 4386998, Rev. C). This document covers complete site preparation and safety information that is necessary for an off-line computer cluster. Some of this material is repeated in this chapter for convenience.

Requirements for off-instrument data processing and analysis

To perform automated off-line data processing and analysis, two requirements must be satisfied on the off-line cluster:

1. The off-line cluster must meet the minimum specification/requirement for both hardware and operating system software. See the off-line cluster specification given in Table 3 on page 157. This computer configuration is also given in Appendix B, *Computer Configuration*.
2. The off-line cluster must contain the stand-alone application Global SETS, which includes both SAT v3.0 and SETS v3.0. When an auto-remote analysis is sent from the instrument to the off-line cluster, Global SETS picks the job and completes it on the off-line cluster.

Data transfer

Data transfer from the SOLiD™ Offline Analysis Cluster is supported and tested on a 1-GB LAN. Applied Biosystems strongly recommends that the you set up a *dedicated* 1-GB network between the SOLiD™ 3 System and your off-line computing systems. To ensure the appropriate transfer speed, you need to configure your network according to one of the suggested solutions:

Current Network	Solution
1-GB LAN (shared)	On a shared network, the effective network speed is much lower than 1GB. Use or install a *dedicated* 1-GB network between instrument and the offline systems to ensure top transfer speeds.
No Gigabit LAN	Install a gigabit switch with the shortest path spanning tree. Connect all the SOLiD™ 3 System outgoing connections to the switch so that data transfer between the SOLiD™ 3 System and offline clusters only occurs through the extra switch at 1 Gbps.

SAT vs Corona_Lite for off-line data analysis

With the SOLiD™ 3 System release, SOLiD™ Global SETS v3.0 provides off-line data analysis pipeline similar to Corona_Lite v4.0. Table 1 describes the differences between these two software applications.

Table 1 Comparison of SAT and Corona_Lite for off-line data analysis

Off-line data analysis **Resequencing pipeline**	**SOLiD™ Global SETS v3.0** **(including SAT v3.0)** **Same as on-instrument analysis**	**Corona_Lite v4.2**
Analysis execution	Automatic through SETS GUI; Integrated command-line	Integrated command. Can run batch mode.
Programming language	Java (algorithms in C++)	Scripting languages (some algorithms in C++)
Integratable with custom pipeline	Yes (SOLiD™ Applications Interface for both GUI (SETS) and command line)	Yes, command line interface
User interface for parameter setting and analysis	Extended rich parameter setting through both GUI (browser interface of SETS) and command-line interface	Expanded rich parameter setting through command-line interface
Multiple run combination analysis	No	Yes
Integrated small indel analysis	No	Yes
Mapping algorithm	MapReads	MapReads
Default mapping setting	Progressive mapping for max throughput (User can overwrite the default to run like corona_lite.)	Full length with fixed number of mismatches
Iterative mapping (Progressive mapping)	User-configurable and automatic	No - manual
RepeatClassifier	Yes, new in v3.0	No
MatchingRepeat	Yes, new in v3.0	No
MatchingRandom	Yes, new in v3.0	No
MatchingConsolidate	Yes, new in v3.0	No
SNP algorithm	diBayes	SNP caller

Output results format	Fasta-like matching output, paired mates Automatically generates Gff v.2 (Mapped reads) including Base Sequence, When DiBayes SNP Detection is enabled, Gff v.3 SNP result and consensus base sequence with SNPs are generated.	Fasta-like matching output (inluding unique match and all match), paired mates, Optional GFF v0.2. SNP list text file
Statistics (Stats) File	New format, in the context of mapped reads and throughput	Old Stats file
Speed	Optimized compute performance for complex genome analysis	Support complex genome analysis
Warranty	Yes	No
Applied Biosystems support to end users	Yes	Yes
License Fee	Comes with SOLiD™ 3 System . Contact your Applied Biosystems representative.	Free. Open source
Supported OS	Linux CentOS v4.7, Scyld Clusterware, PBS (Torque). SGE/LSF support will be provided by the SOLiD Software community.	Linux, PBS, LSF, SGE

Overview of off-line cluster site preparation

Applied Biosystems supplies the SOLiD™ 3 Software Suite and hardware for on-instrument primary and secondary analysis.

To complete a large quantity of data analysis, to perform secondary analysis, and to free the memory used by the on-instrument cluster for the next experiment run, Applied Biosystems recommends automatic export of results to the SOLiD™ Off-line Analysis Cluster. If you have purchased the Applied Biosystems SOLiD™ Off-line Analysis Cluster, follow the guidelines in this chapter.

Before an Applied Biosystems service representative arrives to install the SOLiD Offline Analysis Cluster, prepare your site for the installation according to the instructions in this chapter. Checklists are provided later in this chapter.

IMPORTANT! If site preparation tasks are not complete when the Applied Biosystems service representative arrives, the scheduled installation of the SOLiD Off-line Analysis Cluster might be postponed.

Site preparation schedule

To minimize the time between the shipment arrival and installation of the SOLiD Off-line Analysis Cluster:

1. Complete the site preparation tasks.
2. Fill out the corresponding checklists.
3. Schedule installation before the SOLiD Offline Analysis Cluster shipment arrives.
4. Verify with an Applied Biosystems service representative, who will contact you by telephone that:
   - All checklists are complete.
   - The purchase order is complete.
   - You have considered all components and options in preparing the site.

Site preparation workflow

The general site preparation tasks and a suggested sequence for completing the tasks are summarized in Figure 1. The sequence can vary, but is always:

- Review this guide first.
- Do not uncrate the SOLiD™ Off-line Analysis Cluster until an Applied Biosystems service representative arrives.

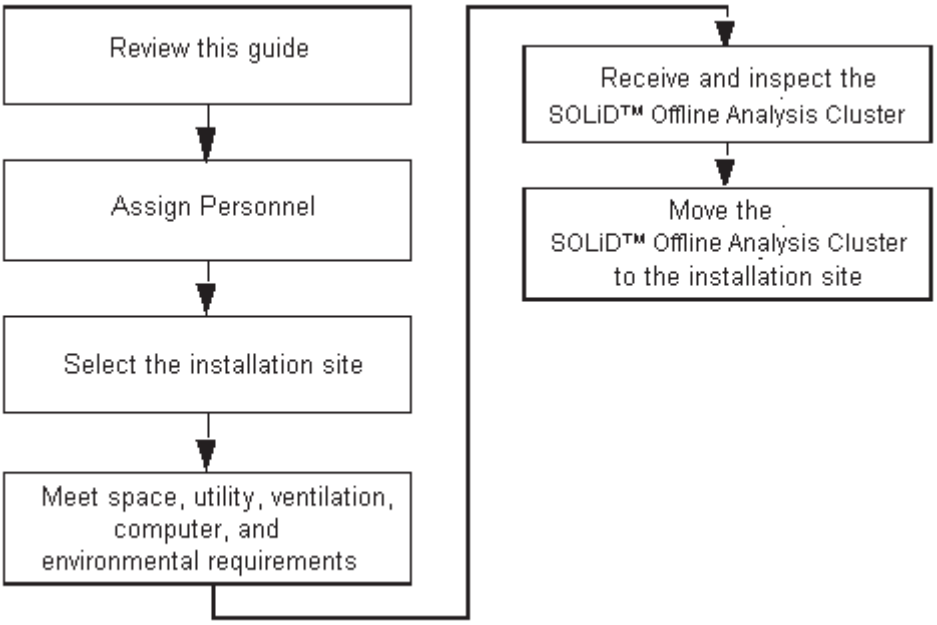


Figure 1 Site preparation tasks and their suggested sequence

Assign personnel

Laboratory safety representative

Applied Biosystems requests that a representative from your laboratory be in the vicinity and available to the Applied Biosystems service representative at all times while the service representative is at your facility. The laboratory safety representative should be familiar with laboratory safety procedures and know the location of all the safety equipment.

Tasks and personnel

Table 2 summarizes specific site-preparation tasks and suggests the personnel to accomplish the tasks. Use the table to help schedule and manage the site-preparation process.

Table 2 Suggested personnel tasks

Personnel	Tasks
Site Preparation/ Installation Coordinator	• Reviews the site preparation guide for safety information and system requirements. • Coordinates personnel and tasks. • Contacts the information technology department of your institution to meet network, power, and environmental requirements. • Chooses the site. • Reviews checklists with applicable personnel, then with the Applied Biosystems service representative to verify that the site is properly prepared. • Receives and inspects the system. • Places the SOLiD™ Off-line Analysis Cluster. • Ensures that the site is clear of unnecessary material on the installation day. • Is available to assist the service representative throughout installation.
Laboratory Safety Representative	• Reviews the site preparation guide for safety information. • Ensures that the required safety practices and equipment are in place. • Is available to assist with unpacking and setup.
Laboratory Personnel/ Primary Users	• Review safety information. • Ensure that all customer-provided materials for installation are present at the site. • Primary users (responsible for training other users) are available during the entire installation.
Facilities Personnel	• Ensure that installation requirements are met for: – Space at the installation site – Building clearances – Temperature and humidity – Ventilation and waste collection – Electrical supply – Computer – Safety and installation materials • Install the SOLiD™ Offline Analysis Cluster on the same day of arrival. • Are available to assist service representative and laboratory personnel throughout installation. • Shipping personnel move and position the system.
Network or IT Specialist (To connect the system to the network)	• Ensures that one active, tested local area network (LAN) connection is in place before the scheduled installation date. • Ensures that network hardware is compatible with a RJ45-type connector. • If necessary, supplies additional cables. • Is available during installation to connect the system to the network. **CAUTION** *Do not* attempt to connect the system components to the network before the Applied Biosystems service representative arrives. • If applicable, provides and installs a network or dedicated printer.

Selecting the site

When you decide where to install the instrument, refer to:

- "Space requirements" on page 157
- "Environmental requirements" on page 160
- "Ventilation requirements" on page 161
- "Electrical requirements" on page 162
- "Safety and materials" on page 165

IMPORTANT! The site cannot be designated BioSafety Level 3 (BSL-3) or BioSafety Level 4 (BSL-4). Applied Biosystems does not install, service, or repair Applied Biosystems instruments in areas designated BSL-3 or BSL-4.

Space requirements

SOLiD™ Off-line Analysis Cluster components

Note: Applied Biosystems might send the materials required for installation with the SOLiD™ Off-line Analysis Cluster or in a separate shipment.

The SOLiD™ Off-line Analysis Cluster components include the following. Note that this is the RECOMMENDED specification. This configuration is also given in Appendix B, *Computer Configuration*.

Table 3 SOLiD™ 3 off-line cluster components (recommended specification)

Computer components	• Racked 17-inch flat screen monitor, mouse and keyboard
	• Head node
	• Five compute nodes
	• Storage
	• Gigabit switch
	• Rack Power Distribution Units (PDU) (miinimum of 2)
	• Dual-rack UPS; power cords (2), attached
SOLiD™ 3 software	SOLiD™ 3 Global SETS:
	• SOLiD™ Experimental Tracking System (SETS)
	• SOLiD™ Analysis Tools (SAT)
Head node	• Hardware: Intel® Xeon® processors
	• PowerEdge 2950: Intel® Xeon® Dual Quad Core processors
	• Operating system: 64-bit LINUX CentOS 4.7; Scyld ClusterWare 4.2.2, Torque v2.4.0
	• Installed RAM: 16 GB
	• Hard disk storage: 6 x 1 TB SATA hard drives (RAID-5)
5 Compute nodes (each)	• PowerEdge 1950; Intel® Xeon® Dual Quad Core processors
	• Installed RAM: 16 GB
	• Hard disk storage: 2 x 1 TB SATA hard drives (RAID-0)
Storage	• Hard disk storage: 15 x 1 TB SATA hard drives (RAID-5 with hot spare)
Gigabit switch	• 16-port Gigabit Switch
Power distribution units	• Rack PDU (2): each with 8 output connectors (16 total)
Rack UPS	• Two Rack UPS

For ordering information and part numbers, contact the Applied Biosystems sales representative or go to: **www.appliedbiosystems.com**, click **SOLiD™ System Sequencing.**

IMPORTANT! Do not unpack SOLiD™ Off-line Analysis Cluster shipping containers, to protect you from liability if any damage occurred during shipping.

Required tools

Pallet jack to expedite the handling of the crate containing the SOLiD Offline Analysis Cluster.

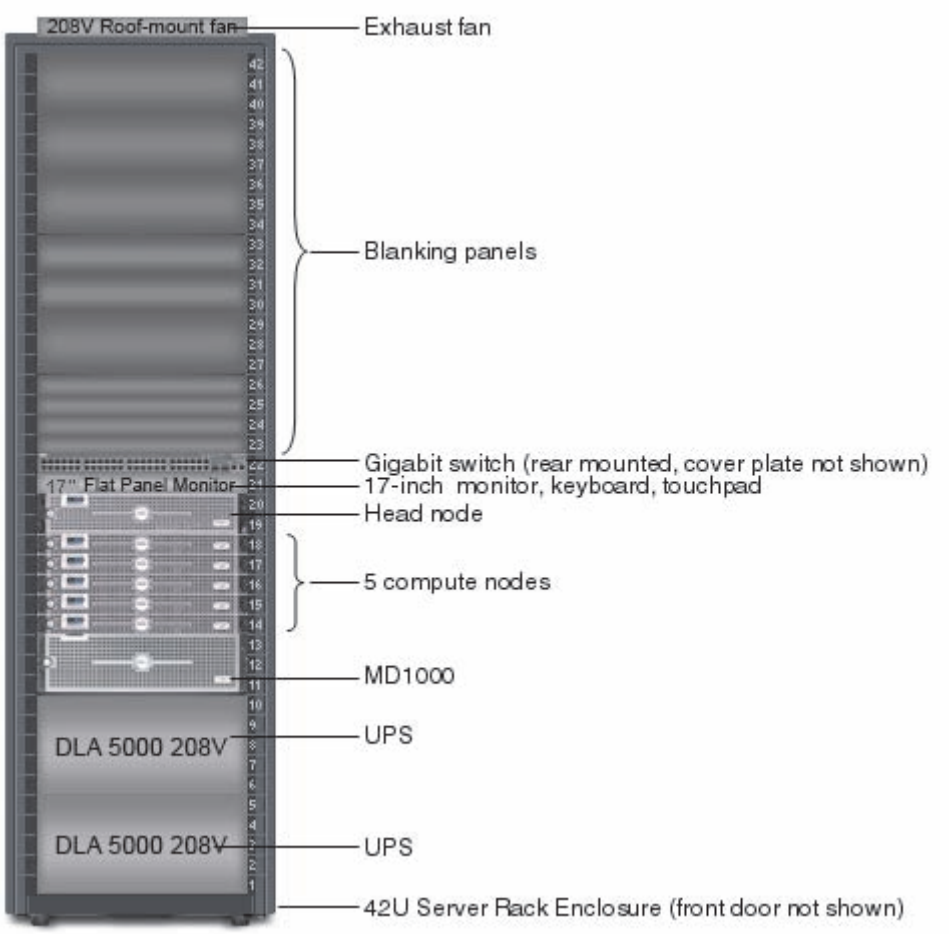


Figure 2 The SOLiD™ Offline Analysis Cluster with front door removed and 208 V components shown. Side-mounted PDU devices are not shown.

Layout requirements

See the Dimensions and weights of the SOLiD™ Off-line Analysis Cluster to help with basic layout considerations.

Dimensions and weights

Ensure that the installation site (floor space and/or bench space) can accommodate the dimensions and support the weights indicated in the table below. The following table lists the dimensions and weights.

Table 4 Dell 42U rack enclosure

Component	Width	Depth	Height	Weight
Dell 42U Rack Enclosure	60.8 cm (24 in)	107 cm (42 in)	208 cm (82 in)	~567 kg (1250 lb)
Dell 42U Rack Enclosure completely filled with server equipment	60.8 cm (24 in)	107 cm (42 in)	208 cm (82 in)	~1134 kg (~2500 lb max)
Dell 42U Rack Enclosure in crate™	~112 cm (~44 in)	~130 cm (~51 in)	~219 cm (~86 in)	~907 kg (~2000 lb max)

Required clearances

Clearance on all sides

The SOLiD™ Off-line Analysis Cluster requires clearance for ventilation, service access, and cable routing:

- Front clearance: 18.9 cm (48 in)
- Rear Clearance: 14.2 cm (36 in)
- Side Clearance: 2.54 to 5.08 cm (1 to 2 in)

Note: Clearance to the side panels is not typically required as long as the rear and front can be accessed by personnel. Example: Server racks may be installed side-by-side.

Note: Allow space for the Applied Biosystems service representative to move the SOLiD Off-line Analysis Cluster for access to the back and sides.

IMPORTANT! Access to the back and front of the SOLiD™ Off-line Analysis Cluster is required for service activities.

Vertical clearance

At least 61 cm (24 in) of unobstructed vertical clearance above the top of the SOLiD™ Off-line Analysis Cluster to allow access to exhaust fans for servicing.

Environmental requirements

Location

If you are installing the SOLiD™ Off-line Analysis Cluster in a typical room designed for a data center, then the offline cluster may be elevated on the data center floor. Standard procedures followed in data centers to blow cool air to the front of the rack and exhaust hot air from the rear or top of the rack are sufficient.

Altitude

The SOLiD™ Off-line Analysis Cluster is for indoor use only and for altitudes from ~16 m to 10,600 m (~50 ft to 35,000 ft) above sea level.

Temperature and humidity

Ensure that the installation site is maintained under the following conditions:

The following table lists the temperature and humidity requirements.

Table 5 SOLiD™ Off-line Analysis Cluster temperature and humidity requirements

Condition	Acceptable range
Temperature	10 ·C to 35 ·C (50º F to 95º F) Room temperature should not fluctuate more than 2 °C over a 2-hour period.
Humidity	20 to 80% relative humidity, noncondensing

Avoid placing the system adjacent to heaters, cooling ducts, or in direct sunlight. Fluctuations between day and night temperatures can cause system instability.

Operating vibration

To ensure optimum system performance, environmental sources of vibration within the server rack must be limited.

The SOLiD™ Off-line Analysis Cluster must be installed in a location with floor vibration intensity, for any given frequency, of 0.26 G at 5 Hz to 350 Hz for 2 minutes.

Operating shock

Six shock pulses of 71 G for up to 2 ms

Sound

The SOLiD™ Off-line Analysis Cluster generates from 7.9 to 8.4 bels A-weighted sound.

Ventilation requirements

Vent waste-fume exhaust

The SOLiD™ Offline Analysis Cluster does *not* have a waste-fume exhaust port.

Vent hot-air exhaust (no fumes or vapors) directly into the laboratory air space *only if* the room ventilation system can maintain room temperature with the additional thermal output from the SOLiD Offline Analysis Cluster.

Vent hot-air-only exhaust

Vent the hot-air exhaust from the SOLiD Offline Analysis Cluster through the ventilated rear panel and the exhaust fans on top of rack. The approximate CFM of the roof mounted fan is 425 CFM. The hot air exhaust dissipates heat produced by the SOLiD Offline Analysis Cluster equipment. A hood or similar system can also be used to vent hot exhaust above the offline cluster to an AC recovery system to maintain room temperatures.

The SOLiD™ Offline Analysis Cluster requires unobstructed air flow through the vented front panel. To dissipate heat, rack mount servers draw air from the front of the servers, then exhaust hot air from the back or top of the servers. Therefore, front to back computer server-grade ventilation is required for all server racks.

Air intake from side panels alone is typically inadequate and unsuitable for rack mount servers. Side ventilation is also not adaptable to multi-bay configuration where two or more server racks are coupled side-by-side.

Rack blanking panels (panels covering unused server locations in the rack) should not be removed to maximize airflow though the front bezels of the installed server equipment housed within the rack. Maximum cooling *through* the server equipment rather than around the server equipment is important. Maintain a minimum clearance of 1 to 1.5 feet in front of the front rack door for “front-to-rear” ventilation.

The maximum thermal output of the SOLiD™ Offline Analysis Cluster instrument “as shipped” is 3000 W (~10,236 BTU), but if the SOLiD Offline Analysis Cluster is fully equipped using all rack positions, the thermal output will reach 10,000 W (34,120 BTU).

Consult your facilities department to determine if the laboratory ventilation system can maintain room temperature with this level of thermal output. If it can maintain room temperature during instrument operation, you can vent the hot-air exhaust port directly to room air.

Connect the hot-air-only exhaust line

If the room ventilation system cannot maintain room temperature because of the heat exhaust from the instrument, connect a venting device such as a fume hood or fume duct.

If the vent is ducted, it must have negative pressure of 2.3 cmm (cubic meters per minute), 80 cfm. Ducting without negative pressure will cause problems with the performance of the SOLiD™ Off-line Analysis Cluster.

Electrical requirements

System electrical requirements

Table 6 provides electrical specifications for the SOLiD™ Off-line Analysis Cluster.

IMPORTANT! For all indicated input voltages, a 30 A circuit with dedicated ground is required.

Table 6 Electrical specifications for the SOLiD™ Off-line Analysis Cluster

Input voltage (VAC)	208	230
Frequency (Hz)	50/60 Hz ± 3 Hz (auto-sensing)	
Rated input current (A)‡	30	
Input connection type	QTY 2, Fixed NEMA L6-30P, 2.44 m (8 ft)	QTY 2, Hardwire 3-wire
Input Cords (included)	—	IEC 320 C19 to C20 Jumper cables, 2.44 m (8ft), 2 each
Input voltage range, mains operations	157 to 255 V	160 to 286 V
Input voltage adjust range	151 to 268 V	151 to 302 V

‡ The approximate current draw from both power cords.

Do not use extension cords.

Power receptacles

Applied Biosystems recommends two receptacles as defined below for 208 or 230 V applications. The receptacles should be located within 3.05 m (10 feet) from the SOLiD™ Offline Analysis Cluster to the UPS power cords. The offline cluster is equipped with two racked UPS instruments. Do not position the offline cluster and UPS(s) so it is difficult to disconnect the power cords.

CAUTION Do not unpack or plug in any components until the Applied Biosystems service representative has configured the system for the proper operating voltage.

Power line regulator

In areas where the supplied power is subject to voltage fluctuations exceeding ±10% of the nominal value, a power line regulator may be required. High or low voltages can adversely affect the electronic components of the instrument.

Power connectors

AB recommends that two receptacles, as defined below for 208 or 230 V applications, should be available within 10 feet of Off-line to connect power cords from UPS(s). Off-line will be equipped with 2 racked UPS. Do not position the equipment so it is difficult to disconnect the power cords.

CAUTION **MULTIPLE POWER CORDS.** Disconnect Both Power Cords Before Servicing Instrument.

Disconnect power

In case of emergency, you must be able to immediately disconnect the main power supply to all the instruments.

The SOLiD™ Offline Analysis Cluster is enabled to perform a safe power down. The two PDUs will be pre-wired to the two internally racked UPS devices. Each UPS device has a power cord for connection to the appropriate power outlet, as defined in “System electrical requirements” on page 162.

Backup power

Customers must use the backup power by way of the two internal UPSs provided in the SOLiD Offline Analysis Cluster.

Applied Biosystems is including two Uninterruptible Power Supplies (UPS) with the SOLiD Offline Analysis Cluster.

Uninterruptible Power Supply (UPS)

Customers must use the backup power provided by of the two internal UPSs that are in the SOLiD Offline Analysis Cluster. The specifications for the UPS are: American Power Conversion 5000 VA Rackmount 5U (see Table 7). Thermal dissipation of the UPS is included in “Vent hot-air-only exhaust” on page 161.

Table 7 Operating specifications for the American Power Conversion 5000 VA Rackmount 5U

Condition	Acceptable Range
Temperature	0 ·C to 40 ·C (32° F to 104° F) Room temperature should not fluctuate more than 2 °C over a 2-hour period.
Humidity	0 to 95%
Elevation	0 to 3000 m (0 to 10,000 ft)

Network requirements

LAN connection

A minimum of 1 RJ45 port with 1-GB speed is recommended for data transfer from the SOLiD™ Off-line Analysis Cluster to the LAN connection. The port should be within 3.1 m (10 ft) of the SOLiD™ Off-line Analysis Cluster.

Anti-virus software

When the SOLiD™ Off-line Analysis Cluster is connected to a private network, no antivirus software is needed because the Linux Head Node acts as a firewall to protect the system.

Anti-virus software, spyware, or any similar software programs should not be installed because they interfere with the SOLiD™ 3 software.

Do not connect the system to an outside or public network because it will not be protected from computer viruses.

Network cables

The computer is factory configured for the TCP/IP protocol. The product includes a fast Ethernet adapter (10/100/1000baseT) with an RJ45-type connector. A proper cable is included in the shipment.

The Ethernet cable for the SOLiD™ Off-line Analysis Cluster is a CAT6 Shielded (~4 m) cable.

Printer requirements

The SOLiD™ Off-line Analysis Cluster can use a network or dedicated printer. The printer and any necessary print drivers must be available before the scheduled installation.

Safety and materials

Safety practices

IMPORTANT! The site must not be designated BioSafety Level 3 (BSL-3) or BioSafety Level 4 (BSL-4). Applied Biosystems does not install, service, or repair Applied Biosystems instruments in areas designated BSL-3 or BSL-4.

Applied Biosystems expects that you will follow all applicable safety-related procedures at all times.

IMPORTANT! A safety representative from your facility must ensure that:

- Personnel establish and follow all applicable safety practices and policies to protect laboratory personnel from potential hazards.
- All applicable safety devices and equipment are available at all times.

Required safety equipment

The following safety protection and equipment must be available at the installation site:

- Protection from any sources of hazardous chemicals, radiation (for example, lasers, radioisotopes, radioactive wastes, and contaminated equipment), and potentially infectious biological material that may be present in the area where the Applied Biosystems service representative will work.
- Appropriate fire extinguisher:
  - You are responsible for providing an appropriate fire extinguisher for use on or near Applied Biosystems equipment.
  - The types and sizes of fire extinguishers shall be suitable for use on electrical and chemical fires as specified in current codes, regulations, and/or standards, and with approval of the Fire Marshall or other authority having jurisdiction.
  - The installation of appropriate fire extinguishers shall be in addition to other fire-protection systems and not as a substitute or alternative to them.

Receive and inspect the SOLiD™ Off-line Analysis Cluster

Shipped contents

IMPORTANT! To protect you from liability if any damage occurred during shipping, do *not* unpack the SOLiD™ Offline Analysis Cluster shipping containers.

Shipping list

Verify that the items shown on the shipping list are the same items that you ordered.

Inspect shipping containers for damage

Carefully inspect the shipping containers and report any damage to the Applied Biosystems service representative. Record any damage or mishandling on the shipping documents.

Unpack and store

IMPORTANT! Unpack only items that are temperature sensitive.

Move the crated SOLiD™ Off-line Analysis Cluster to the data center or laboratory

Moving schedule

Before the date of installation:

- Clear the installation site of all unnecessary materials.
- Make available 1.1 m ∞ 2.1 m (3.6 ft ∞ 6.9 ft) area to receive the crate before placing it in the laboratory.

IMPORTANT! Service engineers should be careful when moving the instrument across thresholds or elevator gaps with a slight angle. Cross each gap using only one one wheel at a time to minimize stress to the wheels and server frame.

To move the crate to the installation site, verify that the building clearances and local specifications are observed (see Table 8).

WARNING **PHYSICAL INJURY HAZARD.** Do not attempt to lift the crated SOLiD Offline Analysis Cluster. The SOLiD Offline Analysis Cluster is mounted on wheels. After the Applied Biosystems service representative uncrates the SOLiD Offline Analysis Cluster, it can be pushed from location to location.

Table 8 Required building clearances for the SOLiD™ Off-line Analysis Cluster

Crate Dimension	Minimum Building Clearance
Height	219 cm (86 in)
Length	112 cm (44 in)
Depth	130 cm (51 in)
Weight	~907 kg (2000 lb)

The SOLiD™ Offline Analysis Cluster weight

The Dell 42U Rack Enclosure, fully racked with the SOLiD™ Offline Analysis Cluster, weighs ~1134 kg (~2500 lb).

Move and lift the SOLiD™ Offline Analysis Cluster

CAUTION **PHYSICAL INJURY HAZARD.** The instrument is to be moved and positioned under the supervision of the Applied Biosystems service representative. If you decide to lift or move the instrument after it has been installed, do not attempt to lift or move the instrument without the assistance of others, the use of appropriate moving equipment, and proper lifting techniques. Improper lifting can cause painful and permanent back injury. Depending on the weight, moving or lifting an instrument may require two or more persons.

CAUTION Do not tip the SOLiD™ Off-line Analysis Cluster on end. Tipping damages the SOLiD™ Offline Analysis Cluster hardware and electronics.

During installation

Installation and testing

After the SOLiD™ Offline Analysis Cluster is uncrated, installation and testing of the SOLiD™ Offline Analysis Cluster by the Applied Biosystems service representative takes about 2 to 4 days.

CAUTION While the SOLiD™ Offline Analysis Cluster is being installed, avoid exposure to hazards that may be associated with the installation process.

Operator training

During and/or after installation, the Applied Biosystems service representative reviews data and provides some basic operator training. For additional training and reference information, see the user documents provided with the SOLiD™ Off-line Analysis Cluster.

IT requirements

SOLiD™ Off-line Analysis Cluster remote access

During the installation period, Applied Biosystems requires access to the system.

One or more of the following Applied Biosystems personnel will access the system:

- Field Application Specialist
- Territory Specialist
- Bioinformatics Specialist
- Professional Services Representative

Applied Biosystems requires that the customer's Network or IT technician be available to assist the Applied Biosystems representatives in establishing remote access, namely through SSH (preferred remote access method) or VPN.

Remote Access using SSH (Secure Shell)

A static IP address must be assigned to the Linux head node (see "SOLiD™ Off-line Analysis Cluster network information" below). The Applied Biosystems representatives set up port forwarding on the Field Service Engineer's laptop so that the SSH connection is forwarded to port 5900 (or 5800). Port 5900 (or 5800) is used to connect to VNC running on the Linux head node.

IMPORTANT! Contact the Professional Services representative with your concerns over connections and restrictions.

Remote Access using VPN (Virtual Private Network)

The customer's Network or IT technician must set up four VPN accounts and provide the access information to the Applied Biosystems representatives. If necessary, establish one account for each of the following:

- Field Application Specialist
- Territory Specialist
- Bioinformatics Specialist
- Professional Services Representative

SOLiD™ Off-line Analysis Cluster network information

The following sections describe the requirements from the Customer's IT Department.

Network Access

For the SOLiD™ Offline Analysis Cluster, the customer's IT department must provide one active Ethernet port that is connected to the customer's corporate network.

DNS Entry

If needed, the Applied Biosystems representatives provide the customer's IT department with the MAC address of the Linux head node. The customer's IT department adds an entry into its DNS maps for the Linux head node.

Static IP Address

To establish a Static IP Address, the Applied Biosystems representatives need from the customer's IT department the:

- IP Address to use for the Linux head node
- Subnet Mask
- IP Address of the Gateway
- IP Address of the DNS Nameserver

If you are using a 10.1.x.x subnet for network, then you need to install a gigabit router. Connect the SOLiD™ Off-line Analysis Cluster to the router, then connect the router to local network.

Data transfer

Data transfer from the SOLiD™ Offline Analysis Cluster is supported and tested on a 1-GB LAN. Applied Biosystems strongly recommends that the you set up a *dedicated* 1-GB network between the SOLiD™ 3 System and your offline computing systems. To ensure the appropriate transfer speed, you need to configure your network according to one of the suggested solutions:

Current Network	Solution
1-GB LAN (shared)	On a shared network, the effective network speed is much lower than 1GB. Use or install a *dedicated* 1-GB network between instrument and the offline systems to ensure top transfer speeds.
No Gigabit LAN	Install a gigabit switch with the shortest path spanning tree. Connect all the SOLiD™ 3 System outgoing connections to the switch so that data transfer between the SOLiD™ 3 System and offline clusters only occurs through the extra switch at 1 Gbps.

Off-line cluster checklists

Before you use these checklists, read all previous sections in this chapter.

Use the checklists in this chapter to ensure that you have made all preparations for installing the system. An Applied Biosystems service representative will contact you to verify that all checklists are complete before setting up the installation date.

In the following checklists, date each item after verifying its completion.

Assigning Personnel Checklist (SOLiD™ Off-line Analysis Cluster)

For more information, see the topic "Assign personnel" on page 154

Date Verified	Designated Personnel
	Site Preparation/Installation coordinator
	Laboratory safety representative
	Laboratory personnel: • To ensure that customer-supplied materials are on hand • Primary users to be trained during installation and to subsequently train other users
	Facilities personnel to provide environmental, electrical, and computer site-preparation requirements
	Network or IT specialist (To connect the system to the network)

Space and Layout Checklist (SOLiD™ Off-line Analysis Cluster)

For more information, see the topic "Space requirements" on page 157.

Date Verified	Requirements
	A Pallet Jack is available for moving the instrument crate.
	Location is away from: • Heating or cooling ducts. • Direct sunlight. • Object or appliance that cause vibration
	Computer workspace allows for proper ergonomics during use.

Date Verified	Requirements
	Location accommodates the “Dimensions and weights” on page 159 specified in Table 4.
	Location meets the requirements specified in Required Clearances.

Environmental Checklist (SOLiD™ Off-line Analysis Cluster)

For more information, see the topic “Environmental requirements” on page 160.

Date Verified	Requirement
	The conditions specified in “Temperature and humidity” on page 160 have been met.

Ventilation Checklist (SOLiD™ Off-line Analysis Cluster)

For more information, see the topic “Ventilation requirements” on page 161.

Date Verified	Requirement
	Instrument Waste-Fume Exhaust Venting
	Materials (including a trap) are available for connecting the instrument to the venting device through a vent line, as described in “Vent hot-air-only exhaust” on page 161.
	Instrument Hot Air Exhaust Venting
	One of the following conditions exists: • Facilities personnel have certified that the normal room ventilation system can maintain room temperature if the maximum thermal output of the system (specified in “Vent hot-air-only exhaust” on page 161) is vented directly into the room air. • A suitable venting device such as a fume hood or fume dust is available to vent the hot air exhaust from the instrument space.
	Materials are available for connecting the instrument to the venting device through a vent line, as described in “Vent hot-air-only exhaust” on page 161.

Electrical Checklist (SOLiD™ Off-line Analysis Cluster)

For more information, see the topic “Electrical requirements” on page 162.

Date Verified	Requirement
	The main power supply to the instrument can be immediately disconnected.
	Appropriate grounded power receptacles are available.

Computer Checklist (SOLiD™ Off-line Analysis Cluster)

Date Verified	Requirement
	Software
	Received a copy of the *SOLiD™ 3 Analyzer SETS Software Getting Started Guide* (PN 4389302).
	Antivirus Software
	Antivirus software, spyware, or any similar software programs are not installed because they interfere with SOLiD™ 3 software. See “Network requirements” on page 164.
	Network
	LAN Connection One active, tested LAN connection is available. A Network or IT Specialist has been assigned. See “LAN connection” on page 164.
	Network Cables Network hardware is compatible with an RJ45-type connector. See “Network requirements” on page 164.
	Printer
	A network printer or a dedicated printer and necessary print drivers are available. **Note:** Installing a printer is optional.

Safety Checklist (SOLiD™ Off-line Analysis Cluster)

For more information, see “Safety and materials” on page 165.

Date Verified	Requirement
	The site is not designated BioSafety level 3 (BSL-3) or BioSafety level 4 (BSL-4).
	Safety practices and policies to protect laboratory personnel from potential hazards are in place and are followed.
	Protection from any sources of hazardous chemicals, radiation (for example, lasers, radioisotopes, radioactive wastes, and contaminated equipment), and potentially infectious biological material is in place.
	Appropriate fire extinguisher
	Eye and hand protection
	Eyewash
	Safety shower
	Vent lines/fume hood, if applicable
	Biohazard waste container, if applicable
	First-aid equipment
	Spill cleanup equipment
	MSDSs readily available

Materials Checklist (SOLiD™ Off-line Analysis Cluster)

Equipment provided by Applied Biosystems

Date Verified	Requirement
Equipment provided by Applied Biosystems	
	Equipment provided for routine operation after the installation is available or has been ordered.

Equipment provided by customers

Date Verified	Requirement
Equipment provided by Customers	
	Equipment provided for routine operation after the installation is available or has been ordered.

System Receipt and Inspection Checklist (SOLiD™ Off-line Analysis Cluster)

For more information, see "Receive and inspect the SOLiD™ Off-line Analysis Cluster" on page 166.

Date Verified	Action
	Received the system and inspected the shipping containers for mishandling or damage. **IMPORTANT!** Do not open any shipping containers.
	Reported any damage to the shipping containers to the Applied Biosystems representative.

Moving the Crated Offline Cluster Checklist (SOLiD™ Off-line Analysis Cluster)

For more information, see “Move the crated SOLiD™ Off-line Analysis Cluster to the data center or laboratory” on page 166.

Date Verified	Item
	The measured building clearances can accommodate the SOLiD™ Off-line Analysis Cluster crate dimensions (see “Move the crated SOLiD™ Off-line Analysis Cluster to the data center or laboratory” on page 166). If the crate dimensions exceed building clearances, contact the Applied Biosystems service representative. Do not unpack the crate without authorization.
	Shipping personnel only will move the crate, uncrate the SOLiD™ Off-line Analysis Cluster, then move the offline analysis cluster to its designated position. **WARNING** **PHYSICAL INJURY HAZARD**. Do not attempt to lift or move any boxed or crated items unless you have received related training. Incorrect lifting can cause painful and sometimes permanent back injury. Use proper lifting techniques when lifting or moving items. No attempt should be made to lift the instrument.
	Cleared the installation site of all unnecessary materials.

IT Requirements Checklist (SOLiD™ Off-line Analysis Cluster)

For more information, see “IT requirements” on page 168.

Date Verified	Item
Remote Access using VPN (Virtual Private Network)‡	
	One account for the Field Application Specialist
	One account for the Territory Specialist
	One account for the Bioinformatics Specialist
	One account for Professional Services Representative
Remote Access using SSH (Secure Shell)	
	Static IP address
	Remote SSH access
SOLiD™ Offline Analysis Cluster Network Information Customer’s IT Department	
	Network Access
	DNS Entry
	Static IP address

‡ If VPN is required.

Data Management A

Saving data

This section describes data storage for the following kinds of data:

- All data
- Raw image data
- Primary analysis results data
- Secondary analysis results data

Approximate storage space for all data

Table 1 gives the size of all data for archival and storage consideration. Note that these are approximate, estimated file sizes.

Table 1 Total Data size for archival and storage consideration

50 nt tag/ 150k/panel, 2357 panels	Image data size‡	Primary analysis results size in .spch format	Primary Analysis Data size (flatfile: .csfasta, _QV.qual, .stats,)	Total size of secondary analysis data including Intermediate Results	Output result in Gff format§
1 slide -1 tag	1.84TB	573 GB	80 GB	1 TB	100 GB
1 slide - 2 tags	3.6 TB	1.15 TB	160 GB	2 TB	200 GB
2 slides - 1 tag	3.6 TB	1.15 TB	160 GB	2 TB	200 GB
2 slides - 2 tags	7.2 TB	2.3 TB	320 GB	4 TB	400 GB

‡ Image data is not needed after analysis is complete.

§ The analysis result data size directly correlates with the throughput.

Raw image data

The data should be kept until the full completion of primary analysis is verified and primary analysis results are archived. After confirmation that the image files are no longer needed, these image files can be deleted to make room for additional sequencing runs.

Primary analysis results data

This section appears also in Chapter 5, *Primary Analysis*.

The image analysis results are stored in .spch files. These files are located in $sampleResultFolder/colorcalls/CACHE/ .

.spch	975 GB	Should be archived

The color call results, including colorspace raw reads and QV data, are located in $smpleResultFolder/primary.[17-digit timestamp]/reads/ .

.csfasta	20 - 25 GB	Should be archived
_QV.qual	40 - 50 GB	Should be archived
.stats	< 10 KB	Should be archived
postPrimerSetPrimary.[JobID]		Temporary files. They are removed after completion. Located in $sampleJobFolder/

There are also colorspace raw reads that were filtered in the sequence generation process. These raw reads include duplicate reads and reads with less than the expected read length. They can be found in $sampleResultFolder/primary.[17-digit timestamp]/reject/ .

.csfasta.reject	< 1 GB	Should be archived
_QV.qual.reject	< 1 GB	Should be archived

In addition, the following files can be ignored. They can be archived if needed.

Intensity files	4 x 120 - 150 = 480 - 600 GB	Optional. Not generated by default. $sampleResultFolder/primary.[17digit timestamp]/reads/ or $sampleResultFolder/primary.[17digit timestamp]/reject/

Secondary analysis results data

The final mapped reads results in GFF Version 2 format correlate directly with the throughput. Depending on the experiment design, a GFF result can be easily copied onto external drives or through the network onto another computer server.

Data transfer procedure

Requirements

- External hard disk drive. The size of the disk drive that you need depends on how much data you decide to archive. Refer to the preceding tables for approximate file sizes.
- The drive should be formatted using UNIX systems as ext3 format.

Network

Data transfer from the SOLiD™ Off-line Analysis Cluster is supported and tested on a 1-GB LAN. Applied Biosystems strongly recommends that the you set up a *dedicated* 1-GB network between the SOLiD™ 3 System and your off-line computing systems. To ensure the appropriate transfer speed, you need to configure your network according to one of the suggested solutions:

Current Network	Solution
1-GB LAN (shared)	On a shared network, the effective network speed is much lower than 1GB. Use or install a *dedicated* 1-GB network between instrument and the offline systems to ensure top transfer speeds.
No Gigabit LAN	Install a gigabit switch with the shortest path spanning tree. Connect all the SOLiD™ 3 System outgoing connections to the switch so that data transfer between the SOLiD™ 3 System and offline clusters only occurs through the extra switch at 1 Gbps.

Procedure

1. Connect the USB drive to the USB port on the head node of Linux cluster.
2. If the drive does not mount automatically, check the log message to determine the device name for the drive:
   ```
   tail –f /var/log/messages
   ```
   or
   ```
   dmesg
   ```
3. If the drive shows up as `/dev/sdc`, then mount the drive as follows:
   ```
   mkdir/mnt/usb(to create a mount point)
   ```
   – This can be skipped to mount the drive under `/media/usbdisk mount/dev/sdc/mnt/usb` (You might need to log in as root), or
   – Mount/dev/sdc/media/usbdisk.
   If the drive mounts automatically, it usually shows up as `/media/usbdisk`

4. Use the following command to copy the directory named {run.name} on the Linux cluster to the USB drive:

   `rsync -aup {run.name} /mnt/usb/`

   or

   `rsync –aup {run.name} /media/usbdisk/` (depending on where the drive was mounted)

   Note: {run.name} = /data/results/{instrument.name}/{run.name}

   IMPORTANT! Perform the copy for each slide separately. If two slides and two tags were run, use two USB drives (700 GB each).

   The transfer rate of is approximately 10-20 MB/sec based on USB2.0 and 750 GB drive. For 300 GB of data, approximately 4 hour is needed to copy the data (150 GB / 120 min).

5. After copying is complete, verify that the USB drive contains the copied data.
6. Unmount the drive.
7. Disconnect the USB drive.

Computer Configuration B

Applied Biosystems supplies or recommends certain configurations of computer hardware, software, and peripherals for use with its instrumentation.
Applied Biosystems reserves the right to decline support for or impose extra charges for supporting nonstandard computer configurations or components that have not been supplied or recommended by Applied Biosystems. Applied Biosystems also reserves the right to require that computer hardware and software be restored to the standard configuration prior to providing service or technical support. For systems that have built-in computers or processing units, installing unauthorized hardware or software may void the Warranty or Service Plan.

This appendix contains two configurations:

- The configuration of the computer that comes with the SOLiD™ 3 System instrument
- The recommended configuration of the SOLiD™ Off-line Analysis Cluster

Both configurations are given, for comparison and completeness. The SOLiD™ Off-line Analysis Cluster recommended configuration was previously given in Chapter 7, *Off-line Data Processing*.

For ordering information and part numbers, contact the Applied Biosystems sales representative or go to: **www.appliedbiosystems.com**, click **SOLiD™ System Sequencing.**

Table 1 contains the specification of the ON-instrument compute cluster:

Table 1 SOLiD™ 3 System analysis server specification	
Computer components	• 19-inch flat screen monitor, mouse, and keyboard
	• Instrument controller
	• Head node
	• Compute nodes (3)
	• Shared storage
	• Gigabit switch
	• Power distribution units (2)
	• Power cords (2), attached
Software Suite	• Instrument Control Software v3.0
	• SOLiD™ Analysis Tools (SAT) v3.0
	• SOLiD™ Experimental Tracking System (SETS) v3.0
Instrument Controller	• Hardware: Intel® Xeon® processors
	• Operating system: Microsoft® Windows® XP Professional, Service Pack 2
	• Installed RAM: 8 GB
	• Hard disk storage: dual 250 GB SATA hard drives (RAID-1)
	• Peripheral: CD-RW/DVD ROM, 19-inch flat screen monitor, keyboard, mouse
Head Node	• Hardware: Intel® Xeon® Quad Core processors (2)
	• Operating system: 64-bit LINUX
	• Installed RAM: 16 GB
	• Hard disk storage: 6 x 1 TB SATA hard drives (RAID-5)
Compute Nodes (each)	• Hardware: Intel® Xeon® Quad Core processors (2)
	• Operating System: 64-bit LINUX
	• Installed RAM: 16 GB
	• Hard disk storage: 2 x 1 TB SATA hard drives (RAID-0)
Shared Storage	• Hard disk storage: 15 x 750 GB SATA hard drives
	• RAID-5 with hot spare
Gigabit Switch	• 16-port Gigabit Switch
Power Distribution Units	• Rack PDU (2) with 16-output connector

Table 2 lists the components of the SOLiD™ Off-line Analysis Cluster. Note that this is the RECOMMENDED specification. This configuration is given also in Chapter 7, *Off-line Data Analysis*.

Table 2 SOLiD™ 3 off-line cluster components (recommended specification)	
Computer components	• Racked 17-inch flat screen monitor, mouse and keyboard
	• Head node
	• Five compute nodes
	• Storage
	• Gigabit switch
	• Rack Power Distribution Units (PDU) (minimum of 2)
	• Dual-rack UPS; power cords (2), attached
SOLiD™ 3 software	SOLiD™ 3 Global SETS:
	• SOLiD™ Experimental Tracking System (SETS)
	• SOLiD™ Analysis Tools (SAT)
Head node	• Hardware: Intel® Xeon® processors
	• PowerEdge 2950: Intel® Xeon® Dual Quad Core processors
	• Operating system: 64-bit LINUX CentOS 4.7; Scyld ClusterWare 4.2.2, Torque v2.4.0
	• Installed RAM: 16 GB
	• Hard disk storage: 6 x 1 TB SATA hard drives (RAID-5)
5 Compute nodes (each)	• PowerEdge 1950; Intel® Xeon® Dual Quad Core processors
	• Installed RAM: 16 GB
	• Hard disk storage: 2 x 1 TB SATA hard drives (RAID-0)
Storage	• Hard disk storage: 15 x 1 TB SATA hard drives (RAID-5 with hot spare)
Gigabit switch	• 16-port Gigabit switch
Power distribution units	• Rack PDU (2): each with 8 output connectors (16 total)
Rack UPS	• Two Rack UPS

Software Warranty Information C

APPLIED BIOSYSTEMS END USER SOFTWARE LICENSE AGREEMENT

FOR INSTRUMENT OPERATING AND ASSOCIATED BUNDLED SOFTWARE AND LIMITED PRODUCT WARRANTY

Applied Biosystems SOLiD™ 3 System SAT Software

NOTICE TO USER: PLEASE READ THIS DOCUMENT CAREFULLY. THIS IS THE CONTRACT BETWEEN YOU AND LIFE TECHNOLOGIES REGARDING THE OPERATING SOFTWARE FOR YOUR APPLIED BIOSYSTEMS WORKSTATION OR OTHER INSTRUMENT AND BUNDLED SOFWARE INSTALLED WITH YOUR OPERATING SOFTWARE. THIS AGREEMENT CONTAINS WARRANTY AND LIABILITY DISCLAIMERS AND LIMITATIONS. YOUR INSTALLATION AND USE OF THE APPLIED BIOSYSTEMS SOFTWARE IS SUBJECT TO THE TERMS AND CONDITIONS CONTAINED IN THIS END USER SOFTWARE LICENSE AGREEMENT.

IF YOU DO NOT AGREE TO THE TERMS AND CONDITIONS OF THIS LICENSE, YOU SHOULD PROMPTLY RETURN THIS SOFTWARE, TOGETHER WITH ALL PACKAGING, TO APPLIED BIOSYSTEMS AND YOUR PURCHASE PRICE WILL BE REFUNDED.

This Applied Biosystems End User License Agreement accompanies an Applied Biosystems software product ("Software") and related explanatory materials ("Documentation"). The term "Software" also includes any upgrades, modified versions, updates, additions and copies of the Software licensed to you by Applied Biosystems. The term "Applied Biosystems," as used in this License, means Applied Biosystems, LLC. The term "License" or "Agreement" means this End User Software License Agreement. The term "you" or "Licensee" means the purchaser of this license to use the Software.

THIRD PARTY PRODUCTS

This Software uses third-party software components from several sources. Portions of these software components are copyrighted and licensed by their respective owners. Various components require distribution of source code or if a URL is used to point the end-user to a source-code repository, and the source code is not available at such site, the distributor must, for a time determined by the license, offer to

provide the source code. In such cases, please contact your Life Technologies representative. As well, various licenses require that the end-user receive a copy of the license. Such licenses may be found on the distribution media in a folder called "Licenses." In order to use this Software, the end-user must abide by the terms and conditions of these third-party licenses. After installation, the licenses may also be found in a folder named "Licenses" located in the Software installation's root directory.

TITLE

Title, ownership rights and intellectual property rights in and to the Software and Documentation shall at all times remain with Applied Biosystems, LLC and its subsidiaries, and their suppliers. All rights not specifically granted by this License, including Federal and international copyrights, are reserved by Life Technologies or their respective owners.

COPYRIGHT

The Software, including its structure, organization, code, user interface and associated Documentation, is a proprietary product of Life Technologies or its suppliers, and is protected by international laws of copyright. The law provides for civil and criminal penalties for anyone in violation of the laws of copyright.

LICENSE

Use of the Software

1. Subject to the terms and conditions of this Agreement, Applied Biosystems, LLC grants the purchaser of this product a non-exclusive license only to install and use the Software to operate the single product in connection with which this License was purchased and to display, analyze and otherwise manipulate data generated by the use of such product. There is no limit to the number of computers on which you may install and use the Software to display, analyze and otherwise manipulate such data.

2. If the Software uses registration codes, access to the number of licensed copies of Software is controlled by a registration code. For example, if you have a registration code that enables you to use five copies of Software simultaneously, you cannot install the Software on more than five separate computers.

3. You may make one copy of the Software in machine-readable form solely for backup or archival purposes. You must reproduce on any such copy all copyright notices and any other proprietary legends found on the original. You may not make any other copies of the Software except as permitted under Section 1 above.

Restrictions

1. You agree that you will not copy, transfer, rent, modify, use or merge the Software, or the associated documentation, in whole or in part, except as expressly permitted in this Agreement.

2. You agree that you will not reverse assemble, decompile, or otherwise reverse engineer the Software.

3. You agree that you will not remove any proprietary, copyright, trade secret or warning legend from the Software or any Documentation.

4. You agree to fully comply with all export laws and restrictions and regulations of the United States or applicable foreign agencies or authorities. You agree that you will not export or reexport, directly or indirectly, the Software into any country prohibited by the United States Export Administration Act and the regulations thereunder or other applicable United States law.

5. You agree that you will not modify, sell, rent, transfer (except temporarily in the event of a computer malfunction), resell for profit, or distribute this license or the Software, or create derivative works based on the Software, or any part thereof or any interest therein. Notwithstanding the foregoing, if this Software is instrument operating software, you may transfer this Software to a purchaser of the specific instrument in or for which this Software is installed in connection with any sale of such instrument, provided that the transferee agrees to be bound by and to comply with the provisions of this Agreement.

Trial

If this license is granted on a trial basis, you are hereby notified that license management software may be included to automatically cause the Software to cease functioning at the end of the trial period.

Termination

You may terminate this Agreement by discontinuing use of the Software, removing all copies from your computers and storage media, and returning the Software and Documentation, and all copies thereof, to Life Technologies. Life Technologies may terminate this Agreement if you fail to comply with all of its terms, in which case you agree to discontinue using the Software, remove all copies from your computers and storage media, and return the Software and Documentation, and all copies thereof, to Life Technologies.

U.S. Government End Users

The Software is a "commercial item," as that term is defined in 48 C.F.R. 2.101 (Oct. 1995), consisting of "commercial computer software" and "commercial computer software documentation," as such terms are used in 48 C.F.R. 12.212 (Sept. 1995). Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4 (June 1995), all U.S. Government End Users acquire the Software with only those rights set forth herein.

European Community End Users

If this Software is used within a country of the European Community, nothing in this Agreement shall be construed as restricting any rights available under the European Community Software Directive, O.J. Eur. Comm. (No. L. 122) 42 (1991).

Regulated Uses

You acknowledges that the Software has not been cleared, approved, registered or otherwise qualified (collectively, "Approval") by Applied Biosystems, LLC with any regulatory agency for use in diagnostic or therapeutic procedures, or for any other use requiring compliance with any federal or state law regulating diagnostic or therapeutic products, blood products, medical devices or any similar product (hereafter collectively referred to as "federal or state drug laws"). The Software may not be used for any purpose that would require any such Approval unless proper Approval is obtained. You agree that if you elect to use the Software for a purpose that would subject you or the Software to the jurisdiction of any federal or state drug laws, you will be solely responsible for obtaining any required Approvals and otherwise ensuring that your use of the Software complies with such laws.

LIMITED WARRANTY and LIMITATION OF REMEDIES

Limited Warranty. Applied Biosystems warrants that, during the same period as of the SOLiD Analyzer for which this Software is an instrument operating software, the Software will function substantially in accordance with the functions and features described in the Documentation delivered with the Software when properly installed, and that for a period of ninety days from the beginning of the applicable warranty period (as described below) the tapes, CDs, diskettes or other media bearing the Software will be free of defects in materials and workmanship under normal use.

The above warranties do not apply to defects resulting from misuse, neglect, or accident, including without limitation: operation outside of the environmental or use specifications, or not in conformance with the instructions for any instrument system, software, or accessories; improper or inadequate maintenance by the user; installation of software or interfacing, or use in combination with software or products not supplied or authorized by Applied Biosystems; intrusive activity, including without limitation computer viruses, hackers or other unauthorized interactions with instrument or software that detrimentally affects normal operations;.and modification or repair of the products not authorized by Applied Biosystems.

Warranty Period Commencement Date. The applicable warranty period for software begins on the earlier of the date of installation or three (3) months from the date of shipment for software installed by Applied Biosystems' personnel. For software installed by the purchaser or anyone other than Applied Biosystems, the warranty period begins on the date the software is delivered to you. The applicable warranty period for media begins on the date the media is delivered to the purchaser.

APPLIED BIOSYSTEMS MAKES NO OTHER WARRANTIES OF ANY KIND WHATSOEVER, EXPRESS OR IMPLIED, WITH RESPECT TO THE SOFTWARE OR DOCUMENTATION, INCLUDING BUT NOT LIMITED TO WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE OR MERCHANTABILITY OR THAT THE SOFTWARE OR DOCUMENTATION IS NON-INFRINGING. ALL OTHER WARRANTIES ARE EXPRESSLY DISCLAIMED. WITHOUT LIMITING THE GENERALITY OF THE FOREGOING, APPLIED BIOSYSTEMS MAKES NO WARRANTIES THAT THE SOFTWARE WILL MEET YOUR REQUIREMENTS, THAT OPERATION OF

THE LICENSED SOFTWARE WILL BE UNINTERRUPTED OR ERROR FREE OR WILL CONFORM EXACTLY TO THE DOCUMENTATION, OR THAT APPLIED BIOSYSTEMS WILL CORRECT ALL PROGRAM ERRORS. APPLIED BIOSYSTEMS' SOLE LIABILITY AND RESPONSIBILITY FOR BREACH OF WARRANTY RELATING TO THE SOFTWARE OR DOCUMENTATION SHALL BE LIMITED, AT APPLIED BIOSYSTEMS' SOLE OPTION, TO (1) CORRECTION OF ANY ERROR IDENTIFIED TO APPLIED BIOSYSTEMS IN A WRITING FROM YOU IN A SUBSEQUENT RELEASE OF THE SOFTWARE, WHICH SHALL BE SUPPLIED TO YOU FREE OF CHARGE, (2) ACCEPTING A RETURN OF THE PRODUCT, AND REFUNDING THE PURCHASE PRICE UPON RETURN OF THE PRODUCT AND REMOVAL OF ALL COPIES OF THE SOFTWARE FROM YOUR COMPUTERS AND STORAGE DEVICES, (3) REPLACEMENT OF THE DEFECTIVE SOFTWARE WITH A FUNCTIONALLY EQUIVALENT PROGRAM AT NO CHARGE TO YOU, OR (4) PROVIDING A REASONABLE WORK AROUND WITHIN A REASONABLE TIME. APPLIED BIOSYSTEMS SOLE LIABILITY AND RESPONSIBILITY UNDER THIS AGREEMENT FOR BREACH OF WARRANTY RELATING TO MEDIA IS THE REPLACEMENT OF DEFECTIVE MEDIA RETURNED WITHIN 90 DAYS OF THE DELIVERY DATE. THESE ARE YOUR SOLE AND EXCLUSIVE REMEDIES FOR ANY BREACH OF WARRANTY. WARRANTY CLAIMS MUST BE MADE WITHIN THE APPLICABLE WARRANTY PERIOD.

LIMITATION OF LIABILITY

IN NO EVENT SHALL APPLIED BIOSYSTEMS OR ITS SUPPLIERS BE RESPONSIBLE OR LIABLE, WHETHER IN CONTRACT, TORT, WARRANTY OR UNDER ANY STATUTE (INCLUDING WITHOUT LIMITATION ANY TRADE PRACTICE, UNFAIR COMPETITION OR OTHER STATUTE OF SIMILAR IMPORT) OR ON ANY OTHER BASIS FOR SPECIAL, INDIRECT, INCIDENTAL, MULTIPLE, PUNITIVE, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE POSSESSION OR USE OF, OR THE INABILITY TO USE, THE SOFTWARE OR DOCUMENTATION, EVEN IF APPLIED BIOSYSTEMS IS ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH DAMAGES, INCLUDING WITHOUT LIMITATION DAMAGES ARISING FROM OR RELATED TO LOSS OF USE, LOSS OF DATA, DOWNTIME, OR FOR LOSS OF REVENUE, PROFITS, GOODWILL OR BUSINESS OR OTHER FINANCIAL LOSS. IN ANY CASE, THE ENTIRE LIABILITY OF APPLIED BIOSYSTEMS' AND ITS SUPPLIERS UNDER THIS LICENSE, OR ARISING OUT OF THE USE OF THE SOFTWARE, SHALL NOT EXCEED IN THE AGGREGATE THE PURCHASE PRICE OF THE PRODUCT.

SOME STATES, COUNTRIES OR JURISDICTIONS LIMIT THE SCOPE OF OR PRECLUDE LIMITATIONS OR EXCLUSION OF REMEDIES OR DAMAGES, OR OF LIABILITY, SUCH AS LIABILITY FOR GROSS NEGLIGENCE OR WILLFUL MISCONDUCT, AS OR TO THE EXTENT SET FORTH ABOVE, OR DO NOT ALLOW IMPLIED WARRANTIES TO BE EXCLUDED. IN SUCH STATES, COUNTRIES OR JURISDICTIONS, THE LIMITATION OR

EXCLUSION OF WARRANTIES, REMEDIES, DAMAGES OR LIABILITY SET FORTH ABOVE MAY NOT APPLY TO YOU. HOWEVER, ALTHOUGH THEY SHALL NOT APPLY TO THE EXTENT PROHIBITED BY LAW, THEY SHALL APPLY TO THE FULLEST EXTENT PERMITTED BY LAW. YOU MAY ALSO HAVE OTHER RIGHTS THAT VARY BY STATE, COUNTRY OR OTHER JURISDICTION.

GENERAL

This Agreement shall be governed by laws of the State of California, exclusive of its conflict of laws provisions. This Agreement shall not be governed by the United Nations Convention on Contracts for the International Sale of Goods. This Agreement contains the complete agreement between the parties with respect to the subject matter hereof, and supersedes all prior or contemporaneous agreements or understandings, whether oral or written. If any provision of this Agreement is held by a court of competent jurisdiction to be contrary to law, that provision will be enforced to the maximum extent permissible, and the remaining provisions of this Agreement will remain in full force and effect. The controlling language of this Agreement, and any proceedings relating to this Agreement, shall be English. You agree to bear any and all costs of translation, if necessary. The headings to the sections of this Agreement are used for convenience only and shall have no substantive meaning. All questions concerning this Agreement shall be directed to: Applied Biosystems, 850 Lincoln Centre Drive, Foster City, CA 94404-1128, Attention: Legal Department.

Applied Biosystems, LLC, 850 Lincoln Centre Drive, Foster City, CA 94404.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

L

M

T

U

V

W

X

Worldwide Sales and Support

Applied Biosystems vast distribution and service network, composed of highly trained support and applications personnel, reaches 150 countries on six continents. For sales office locations and technical support, please call our local office or refer to our Web site at **www.appliedbiosystems.com**.

Applied Biosystems is committed to providing the world's leading technology and information for life scientists.

Headquarters
850 Lincoln Centre Drive
Foster City, CA 94404 USA
Phone: +1 650.638.5800
Toll Free (In North America): +1 800.345.5224
Fax: +1 650.638.5884

04/2009

Applied Biosystems

Part Number 4392959 Rev. D