applied biosystems™

# Applied Biosystems SOLiD™ Analysis Tools (SAT) v3.5

## Reference Guide

**For Research Use Only. Not intended for any animal or human therapeutic or diagnostic use.**



Part Number 4443929 Rev.A
10/2009

# 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 Plus 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 Plus System has been installed by an Applied Biosystems technical representative.

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

## How to obtain more information

### Related documentation

The following related documents are shipped with the system:

- *Applied Biosystems SOLiD™ 3 System Plus Site Preparation Guide* (PN 4444009) - Describes how to ready your location and set up for a SOLiD™ 3 System installation.
- *Applied Biosystems SOLiD™ 3 System Plus Instrument Operation Guide* (PN 4442357) - Provides detailed description of how to operate the instrument using Instrument Control Software (ICS).
- *Applied Biosystems SOLiD™ SETS Software v3.5 Getting Started Guide* (PN 4444007) - Provides brief, step-by-step procedures for SOLiD™ 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

This chapter covers:

# SOLiD™ 3 Plus System overview

The Applied Biosystems SOLiD™ 3 Plus 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 2-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 Plus System consists of:

- SOLiD™ Analyzer
- All ancillary equipment
- SOLiD™ Instrument Control Software (ICS) software, which is described in the *Applied Biosystems SOLiD™ 3 Plus System Instrument Operation Guide* (PN 4442357) and *Applied Biosystems SOLiD™ ICS Software v3.5 Help* (PN 4404524).
- SOLiD™ Experiment Tracking System (SETS) software, which is described in the *Applied Biosystems SOLiD™ SETS Software v3.5 Getting Started Guide* (PN 4444007)
- SOLiD™ BioScope™ Software, which is described in the *Applied Biosystems SOLiD™ 3 Plus System BioScope™ Software v1.0 User Guide* (PN 4442694)
- SOLiD™ Analysis Tools (SAT) software, which is described in this document

# Software system workflow

The software applications used to set and control data analysis are 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 | 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 of ICS and SETS. The rest of this document describes SAT.

SOLiD™ 3 Plus System analysis can be organized into primary, secondary, and tertiary analysis, shown in 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 further automated analysis. Secondary analysis can be performed on an off-line analysis cluster instead of on the instrument.

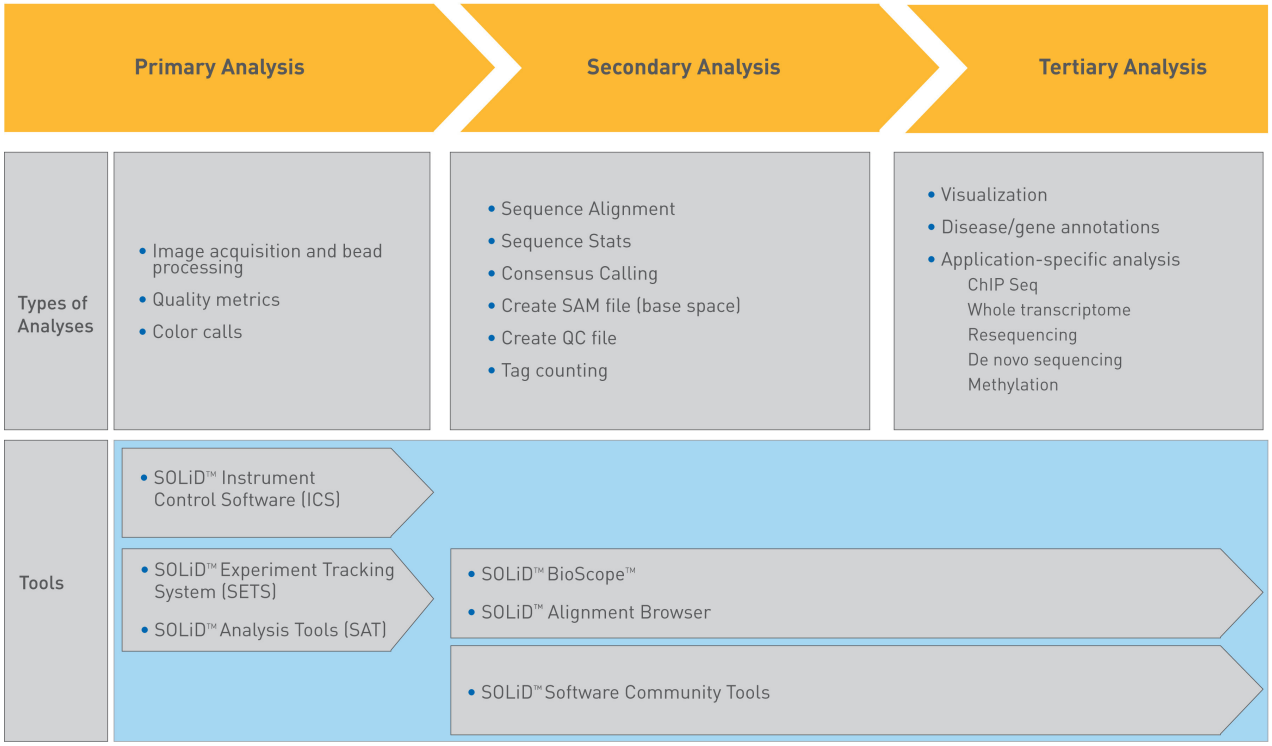



**Figure 1 SOLiD™ workflow between ICS, SETS, and SAT.**

# Types of SOLiD™ 3 Plus experiments

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

**Table 3 Types of SOLiD™ 3 Plus System experiments.**

| Workflow analysis (WFA) | Sequencing (standard) | Multiplexed sequencing |
|---|---|---|
| • Assess various preparations of templated beads to determine potential quality of sequence data<br>• Evaluate fraction of P2-positive beads<br>• 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 |

Table 4 displays detailed information on the different experiment run types:

**Table 4 Characteristics of SOLiD™ 3 Plus System experiment run types.**

| | WFA | Sequencing (standard) | Multiplexed sequencing |
|---|---|---|---|
| Run summary | • P1 and P2 bead counting<br>• Single ligation cycle<br>• 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<br>*and*<br>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<br>• ~6 to 7 days for 50 bp | • ~6 to 7 days for 50 bp<br>• ~1 day for barcode |
| Deposition chamber | 4-well | • 1-well<br>• 4-well<br>• 8-well | • 1-well<br>• 4-well<br>• 8-well |
| Number of beads | 15 million beads per well | • 310 million beads per well (1-well)<br>• 60 million beads per well (4-well)<br>• 30 million beads per well (8-well) | • 310 million beads per well (1-well)<br>• 60 million beads per well (4-well)<br>• 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.

## 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, order additional SOLiD™ Instrument Buffer Kits.

## Sequencing run

During a SOLiD™ system 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, the SOLiD™ system 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.

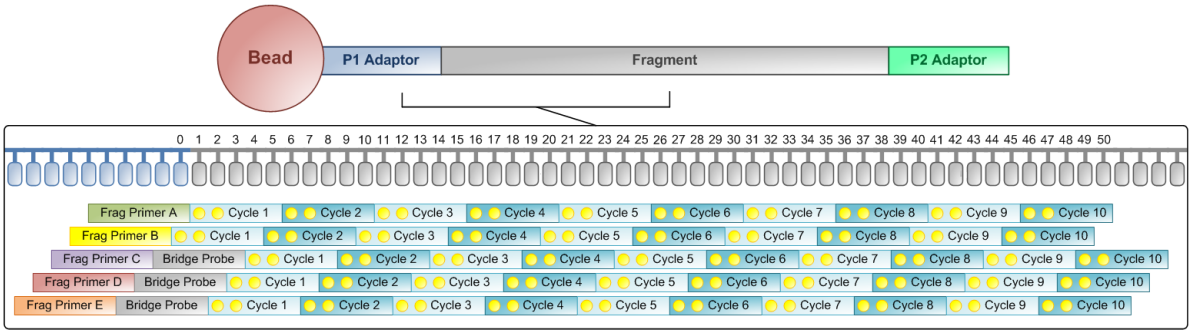



**Figure 2 SOLiD™ 3 Plus System interrogation of nucleotide positions for a 50-bp fragment sequencing run.**

## 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.

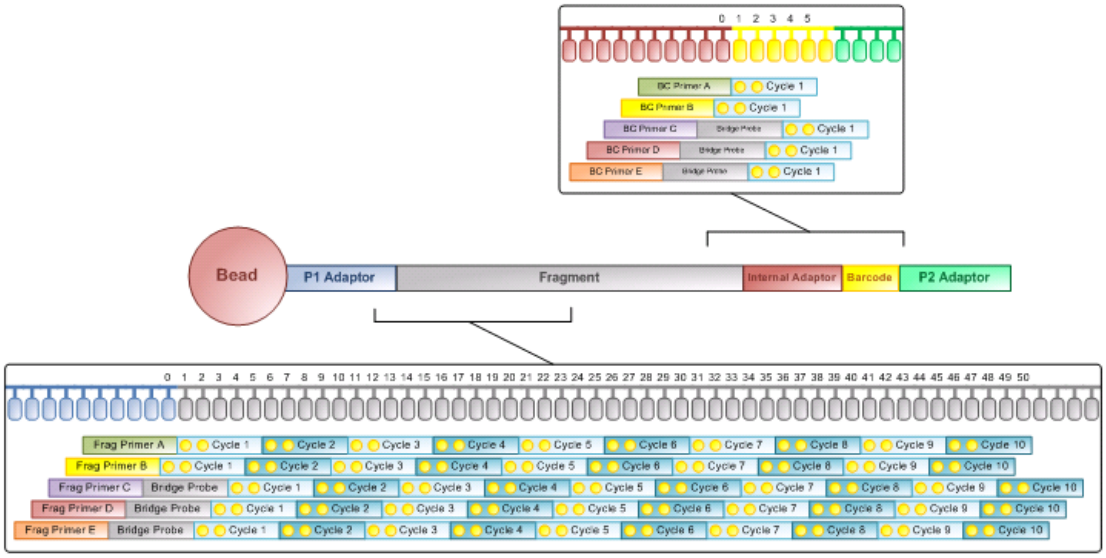



**Figure 3 SOLiD™ 3 Plus System interrogation of nucleotide positions for a 50-bp fragment and 5-bp barcode sequencing run.**

## For more information

For more information on SOLiD™ experiment types and how to run them, refer to the *Applied Biosystems SOLiD™ 3 Plus System Instrument Operation Guide* (PN 4442357).

# ICS overview

SOLiD™ ICS (Instrument Control Software) is a Windows application that you use to set up, start, and manage sequencing runs on a SOLiD™Analyzer.

For more information, refer to the *Applied Biosystems SOLiD™ ICS Software v3.5 Help* (PN 4404524).

Common ICS tasks include:

- Preparing for a run
- Creating runs
- Managing runs
- Monitoring runs
- Checking system status
- Troubleshooting run results

# SETS overview

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.

For more information, refer to the *Applied Biosystems SOLiD™ SETS Software v3.5 Getting Started Guide* (PN 4444007).

Common SETS tasks include:

- Preparing run settings
- Monitoring runs
- Viewing reports
- Performing re-analysis
- Managing administrative tasks

# Fundamentals of color-space analysis

The Applied Biosystems SOLiD™ 3 Plus System sequencing technology is based on sequential ligation of dye-labeled oligonucleotides. This technology makes possible massive 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, known as *color-space analysis*.

## The 2-base color coding scheme

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. In Sanger sequencing, each color, representing only a single nucleotide, is automatically translated to A, C, G, or T. With the SOLiD™ 3 Plus System, each color represents four potential 2-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.

| [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 Plus System's 2-base color-coding scheme.**

## Color-coding process

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 instrument takes the following steps 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™ system sequencing applications, the instrument prepends the leading base A in the example above to result in a *k*-mer A321023022 to reconstruct the base sequence. The SOLiD™ 3 Plus System generates its reads in this encoded form, as shown in Figure 5.

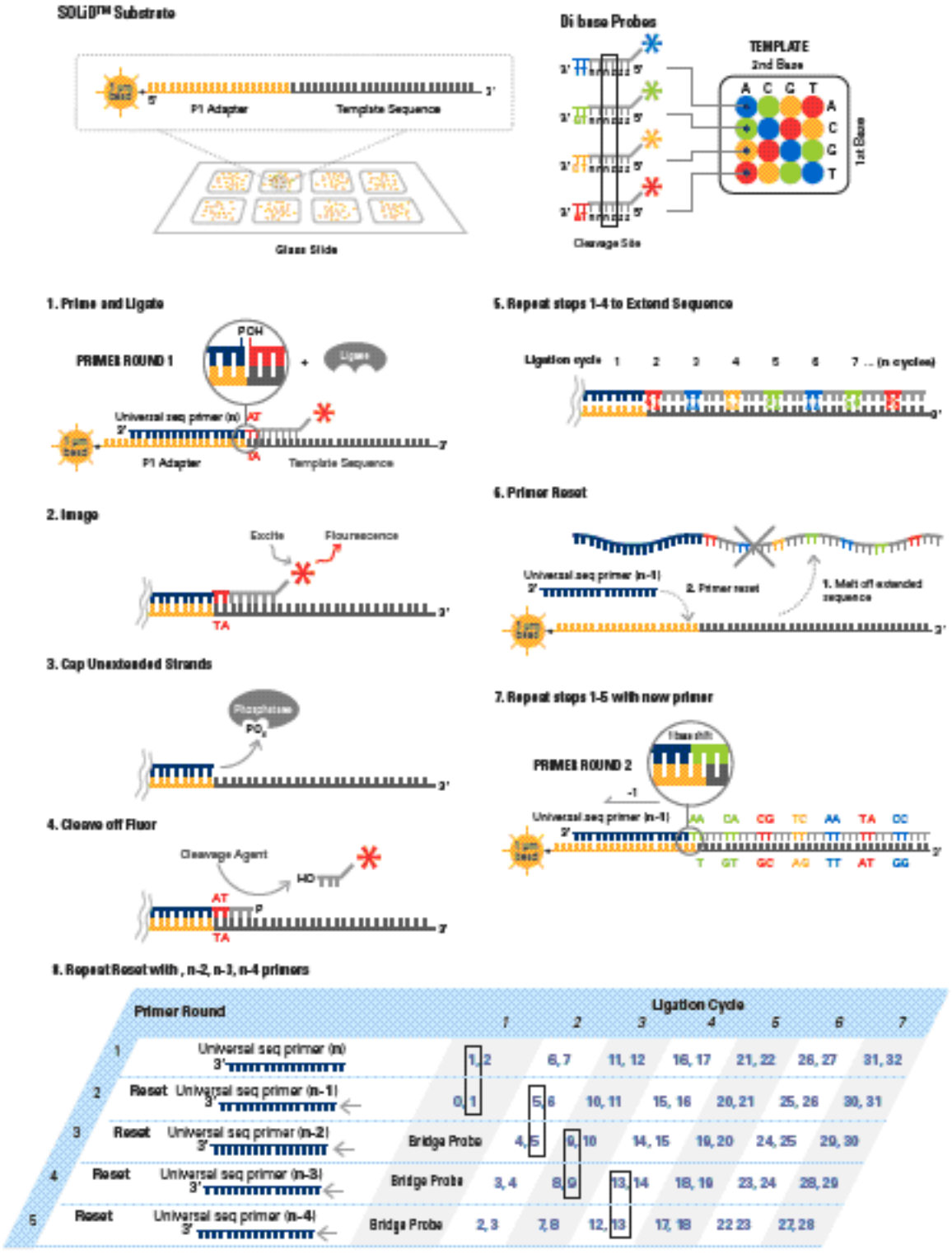



Figure 5 Sequence by ligation using di-base labeling probes on the SOLiD™ 3 Plus System.

## Required properties for a 2-base color code scheme

The 2-base color-coding scheme satisfies the requirements 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 (see 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).

## How the system satisfies 2-base coding system requirements

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 (see Figure 6, Panel C).

Requirement 4, that color (bb) = color (AA), gives a unique labeling for the diagonal (see Figure 6, Panel D).

Finally, requirements 1, 2, and 5 state that every color must appear in every row and every column exactly once (see Figure 6, Panel E).

The table (see 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:

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

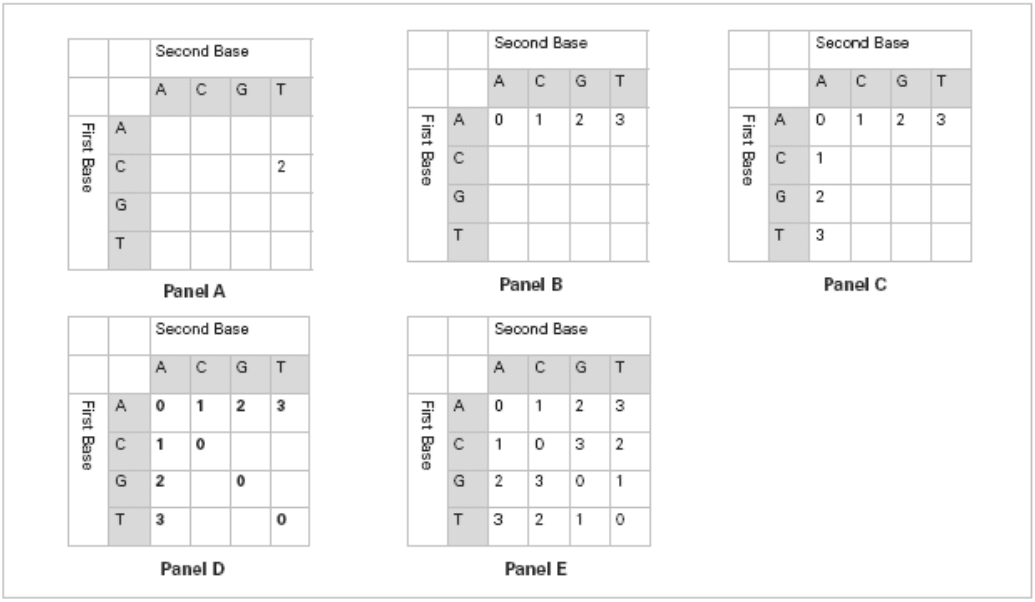



**Figure 6 Requirements that assign color for a 2-base code.**

## Principles of ligation-based chemistry and 2-base encoding

The SOLiD™ 3 Plus 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 on page 10). 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, and so on, 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 Plus System. The advantages of these properties and their contribution to data quality are:

- Two bases are interrogated in each ligation reaction, increasing specificity.
- The primer is periodically reset for five or more independent rounds of extension reactions, improving the signal-to-noise ratio of the system.
- Each base is interrogated twice in two independent primer rounds, increasing confidence in each call.
- 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 Plus System

The final files generated by the SOLiD™ 3 Plus System are in base space. These are the gff, consensus, and SNP files. To maximize the built-in error correction of 2-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

Figure 7 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.

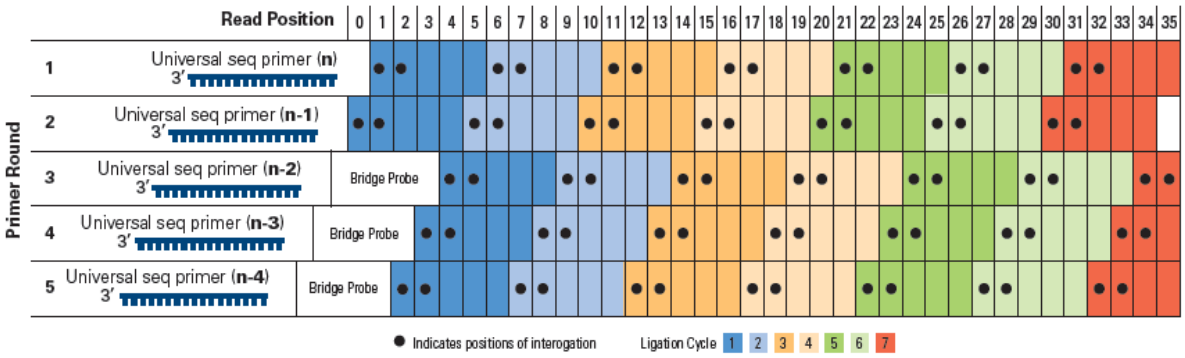



**Figure 7 Relationship between color-space, base position, and sequencing chemistry.**

## 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 Figure 8:

| | | 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 | |

**Figure 8 Color-space data.**

## 2-base encoding and error recognition

The error-checking abilities of the 2-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

Typically, a panel is imaged four times per cycle (once per channel). The color-space calls are carried out on a per-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 are 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 *xxxx.csfsta* file. 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 Plus 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 missing color-space calls, you can mask this position.

## 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 9):

- For any given reference, there are only three valid double-color changes.
- 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.

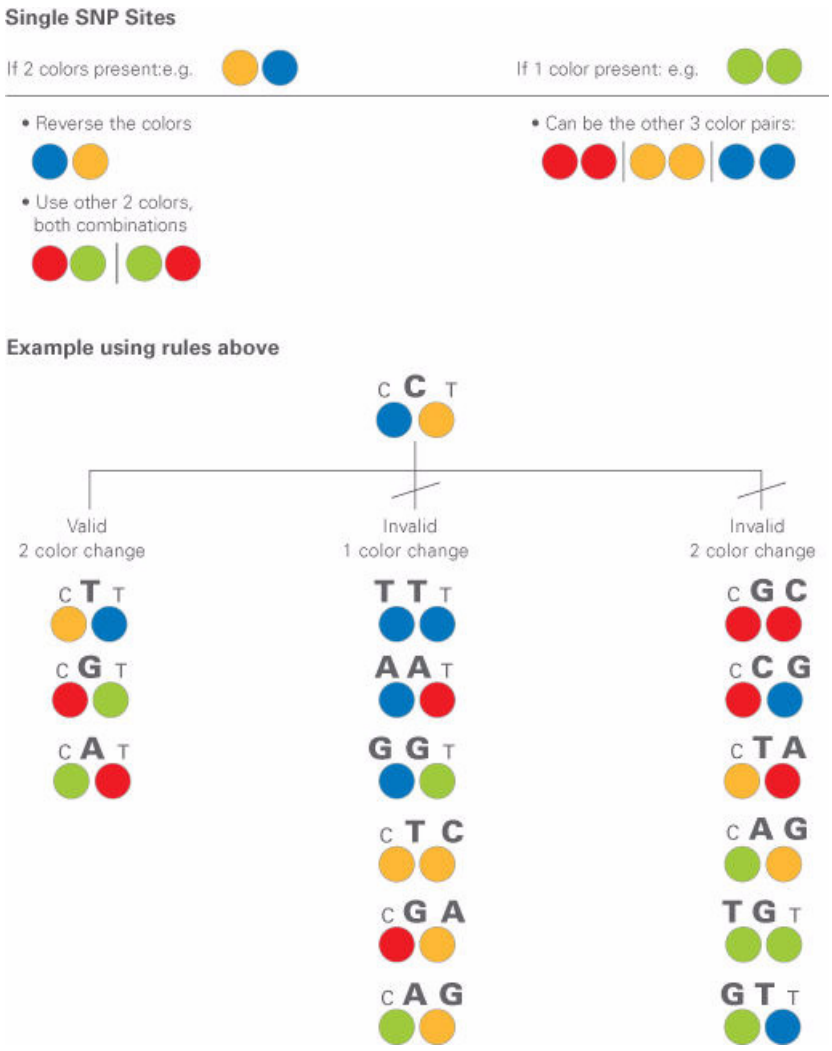



**Figure 9 Color-space rules for single nucleotide polymorphism (SNP) sites.**

Color-space rules are automatically applied during data analysis using the SOLiD™ 3 Plus 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.

## High accuracy for SNP detection

2-base encoding provides high 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 10).

| | 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 10 Color-space transitions.**

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

## SNP error rates

When you use 2-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 2-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 2-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 quality values (QV) can increase the overall accuracy of SNP detection. The SNP detection algorithms of the SOLiD™ 3 Plus System use explicit error models generated from each run and biologically meaningful prior probabilities to evaluate evidence of a SNP. 2-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 Plus System can detect complicated genomic variations such as adjacent SNPs, insertions, deletions, and structural rearrangements.

For these genomic variations, you can download SOLiD™ bioinformatics tools from http://solidsoftwaretools.com. An example of various polymorphisms in color-space is shown in Figure 11.

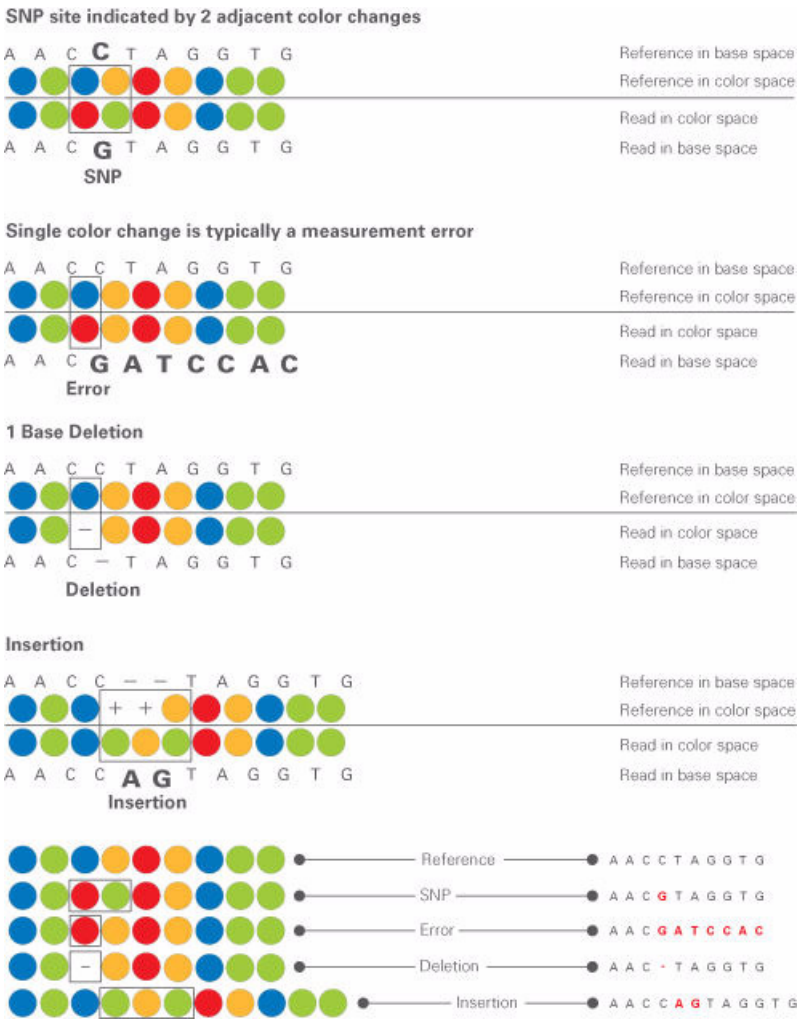



**Figure 11 Examples of polymorphisms in color-space.**

# The Job Manager 2

This chapter covers:

# Introduction

The Job Manager is a small stand-alone daemon process (named Hades) that picks up jobs from the SOLiD™ system 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 12). One example that you can see is that Colorcall jobs are executed *after* Beadfinding jobs are completed.

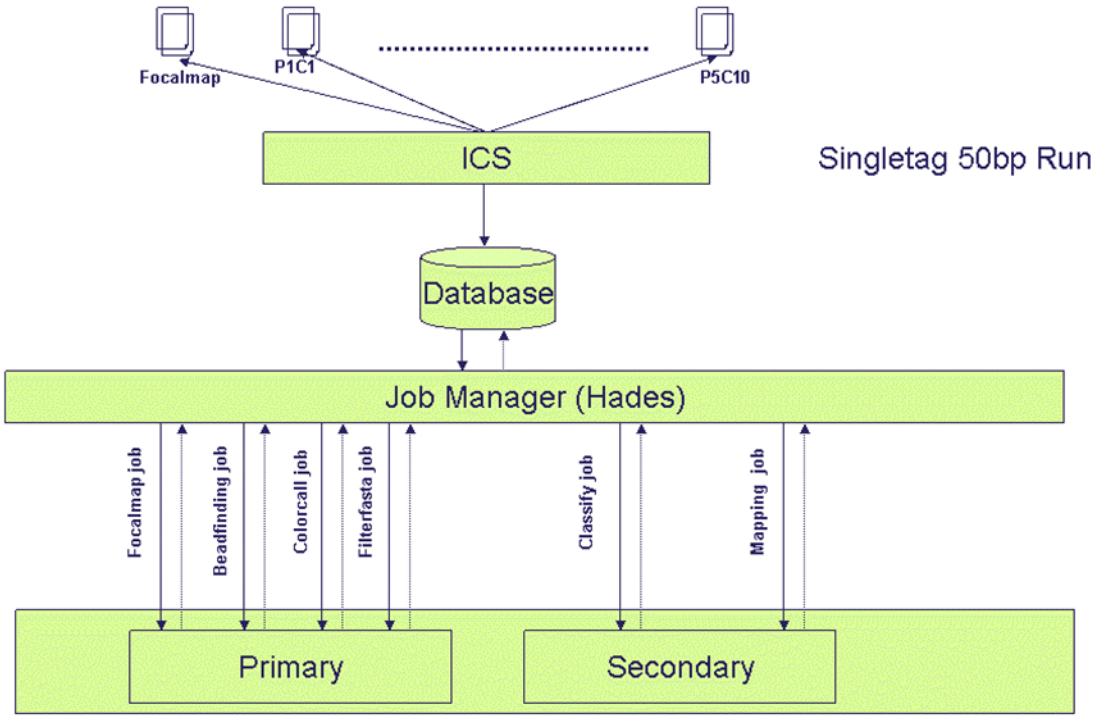



**Figure 12 Job Manager workflow in the SOLiD™ 3 Plus System.**

The SOLiD™ system analysis pipeline contains various stages. Each analysis stage has its specific tasks and Job Manager is responsible for running the pipelines in each stage. Tables 5 gives a summary of stage names and their functions.

**Table 5 Stage names and functions.**

| Stage | Also known as | Comments |
|---|---|---|
| focalmap | postFocalMapPrimary | Finds the position of all beads |
| beadfinding | postBeadFindingPrimary | Combines the P1C1 colorcall information with the focalmap information |
| colorcall | postScanslidePrimary | Finds the beads that are active during the ligation cycle |
| Filter_fasta | postPrimersetPrimary | Combines the reads from all panels |
| Barcoding | postBarcodePrimary | Breaks the reads into libraries; one per tag (F3, R3) |
| <FragmentAnalysisPlugins> | postPrimerSetSecondary | Runs all tag-specific analysis plugins, such as mapping |
| <MatePairAnalysisPlugins> | postRunSecondary | Runs all analysis plugins that go across tags, such as pairing |

With the new Beadfinding job, all Colorcall jobs wait for the Beadfinding to complete. The Beadfinding step requires that the first cycle imaging process is complete.

- **For barcoded runs**: The Beadfinging job depends on the imaging of p1c1 of the barcode primer.
- **For non-barcoded runs**: The Beadfinging job depends on the p1c1 imaging of the first p1c1. Figure 13 gives an overview of job dependency.

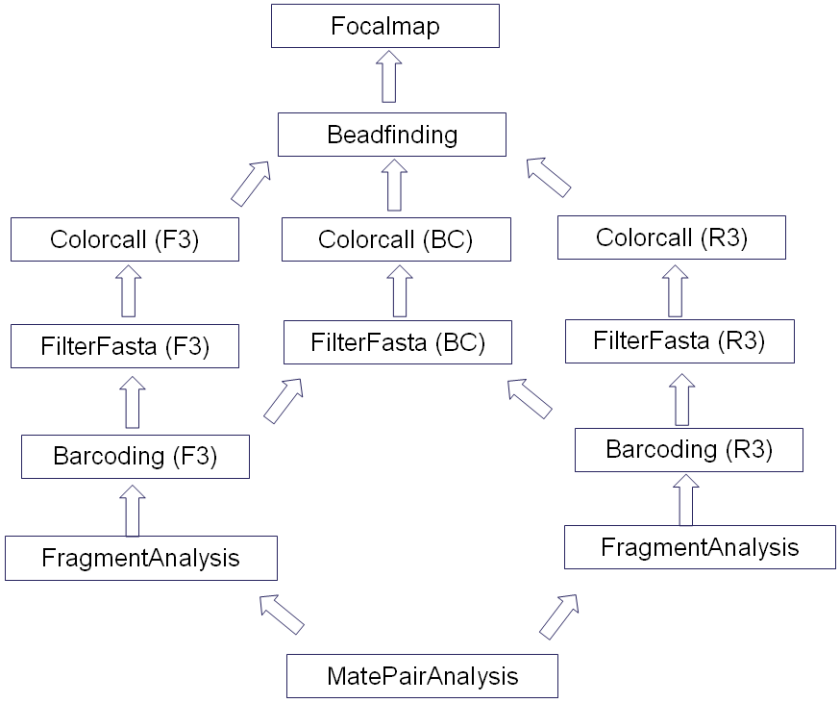



**Figure 13 Job dependency overview.**

# Java Messaging Service

Software subsystems interact with each other using the Java Messaging Service (JMS). Instrument Control Software (ICS) sends out run-specific events to the JMS broker, and primary and secondary modules sends out analysis-specific events to JMS. JobManager is responsible for coordinating these events and sending JobManager-specific messages to the JMS Broker. Figure 14 shows the role of JMS in the Job Manager workflow.

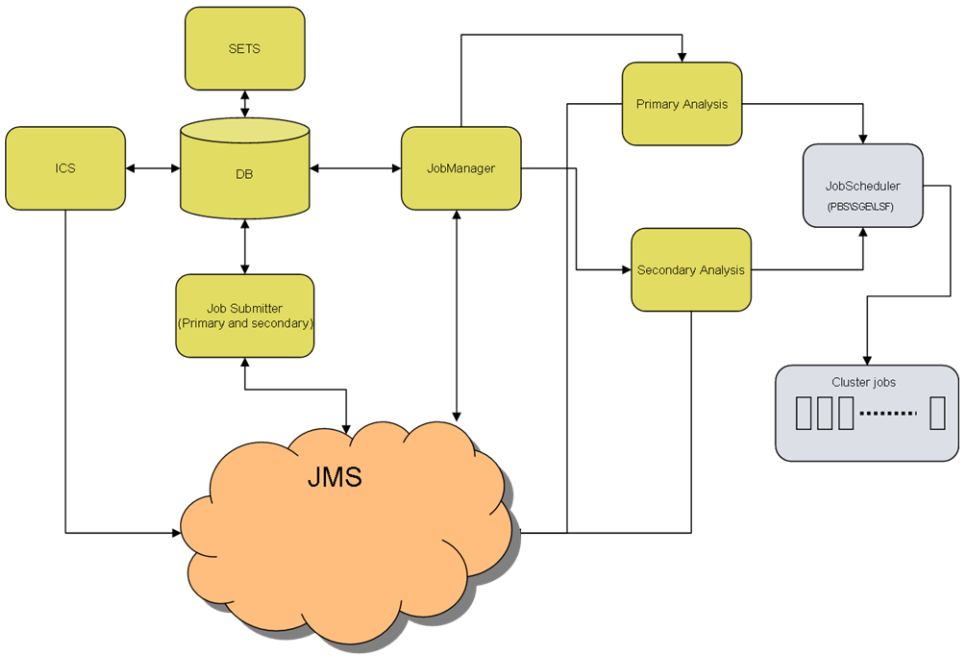



**Figure 14 JMS role in the Job Manager workflow.**

# Troubleshooting

**To start or stop Hades service:**

Enter the following commands:

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

**IMPORTANT!** Do **not** stop Hades while the analysis is running.

**To find log files:**

- Hades: `/var/log/hades`
- moap: `/data/results/[instrumentName]/[runName]/[sampleName]/jobs/postPrimerSetSecondary[id]`

**To find problems:**

Go to `/data/results/[instrumentName]/[runName]/[sampleName]/jobs/[directoryWithId]`, then enter the following command:

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

**To check nodes:**

Enter the following commands:

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

**To find lost panels:**

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

**To view the internal state of Hades (the JMX View):**

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

**To debug the job workflow:**

Enter the following command:

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

# Primary Analysis 3

This chapter covers:

# SOLiD™ system 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, calculating a quality value for each color call, and writing the color-call value to a QV file (.qual).

Primary analysis also provides real-time feedback about a sequencing run.

## Primary analysis overview

Table 6 lists the stages in primary analysis, their inputs, and their outputs.

**Table 6 Primary analysis stages, inputs and outputs.**

| Analysis stage | Inputs | Outputs | Steps |
|---|---|---|---|
| postBeadFinding Primary | Focal map and first ligation cycle .tiff images | • Bead location (in .spch files)<br>• Image statistics and metrics (in *_FOCALMAP_summary.tab files) | For each panel, analyze the focal map and first ligation cycle images to identify beads. |
| postScanSlide Primary | • .tiff images from each ligation cycle<br>• Focal map .tiff images | • Intensities, Color calls, Quality metrics, QV, in .spch files<br>• Satay .png files<br>• Summary data (tab)<br>• Run status info (traffic lights) (tab). | 1. Align images.<br>2. Extract intensities.<br>3. Classify colors (color calls).<br>4. Calculate quality value (QV).<br>5. Generate satay images.<br>6. Accumulate run status information. |
| postPrimerSet Primary | All .spch files | • color-space raw reads (.csfasta)<br>• QV (.qual)<br>• Reads statistics file (.stats file) . | 1. Assemble color-space reads per panel.<br>2. Concatenate per-panel reads into one single .csfasta file (optional).<br>3. Filter reads shorter than specified read length or duplicate reads (optional). |

# 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 the Instrument Control Software (ICS) has finished collecting the images for each step, it creates a job workflow in the SOLiD™ database. The Job Manager spots the 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 and first ligation cycle 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. **Read building**: 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 15 for the primary analysis 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.

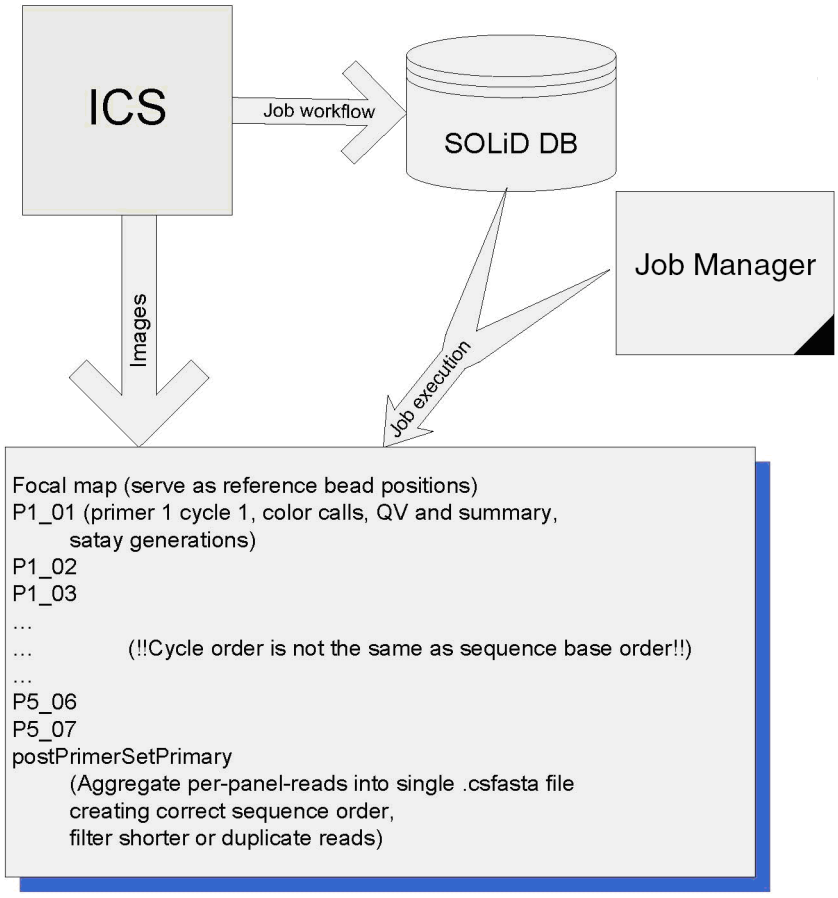



**Figure 15 Primary analysis workflow.**

## 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/postBeadFindingPrimary.[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 Table 7 assume a 50-base tag run in one flowcell. If you use a tag size that is larger or smaller than 50 bases, the storage requirements differ from those given in Table 7.

**Table 7 Primary analysis storage requirements.**

| Files | Storage required | Notes about archiving |
|---|---|---|
| .csfasta | 40 - 50 GB | Should be archived. |
| _QV.qual | 80 - 100 GB | Should be archived. |
| .spch | 476 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**(18**m*+17) bytes, where *n* is the number of beads, and *m* is the number of cycles (F3 and R3).<br>For 50 cycles (1x50) and 220K beads, this yields 201 MB (per panel).<br>A full slide has 2357 panels, which means that 476 GB are required for a full slide (50 bp single tag run). |
| postPrimerSetPrimary.[JobID] | 260 - 300 GB | Temporary files. Most of these are removed upon completion. |
| Intensity files | 4 x 240 - 300 = 960 - 1200 GB | Optional. Not generated by default. It is strongly recommended that intensity files be generated with offline ReadBuilder, which is part of the SOLiD™ BioScope™ software. |

## Perform primary re-analysis

**To perform primary analysis manually, use the following command:**

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

**Note:** Only buildReads tasks are executed for re-analysis. Neither beadFinding nor colorCalling are enabled for re-analysis in the pipeline.

The PrimaryParameters.ini file can be found in `$sampleJobFolder/postPrimerSetPrimary.[JobID]` with the file name parameters.ini.

**To prepare for the re-analysis:**

1. Create a new analysis directory, such as postPrimerSetPrimary.m1.
2. Copy `parameters.ini` to the new directory.
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 the daemon for updating the SOLiD™ database.)

   `logging.socket` (Used by the daemon for logging analysis progress.)
5. Enter `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 the last several bases

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

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

2. Enter the 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.

The file is located in:
`/data/results/{instrument.name}/{run.name}/{sample.name}/results/prim
ary.[17-digit timestamp]/reads/.`

Example of a .stats file:

```
# 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 it encountered a problem and terminated.

# Image metrics

Image metrics are measures of the quality of the images produced by the SOLiD™ 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 16 gives two examples of the blur metric:

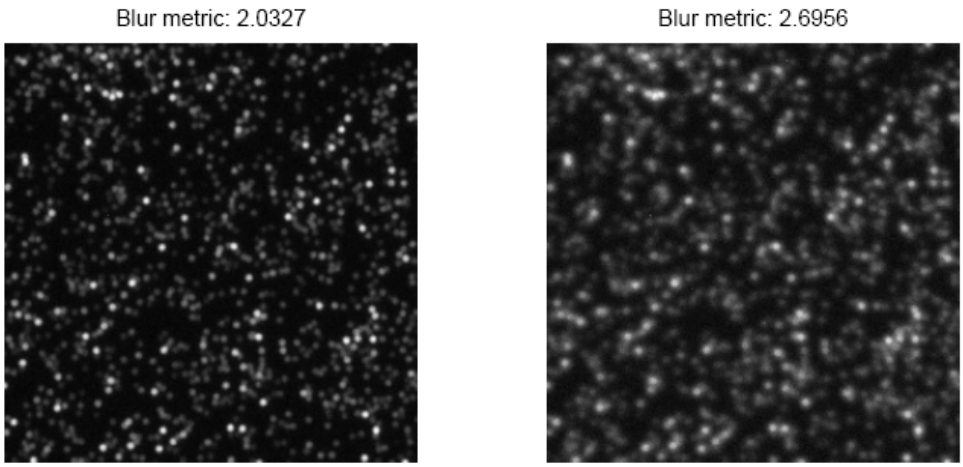



**Figure 16 Two examples of the blur metric.**

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 use the available range of the system.

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

Figure 17 gives two examples of the exposure metric:

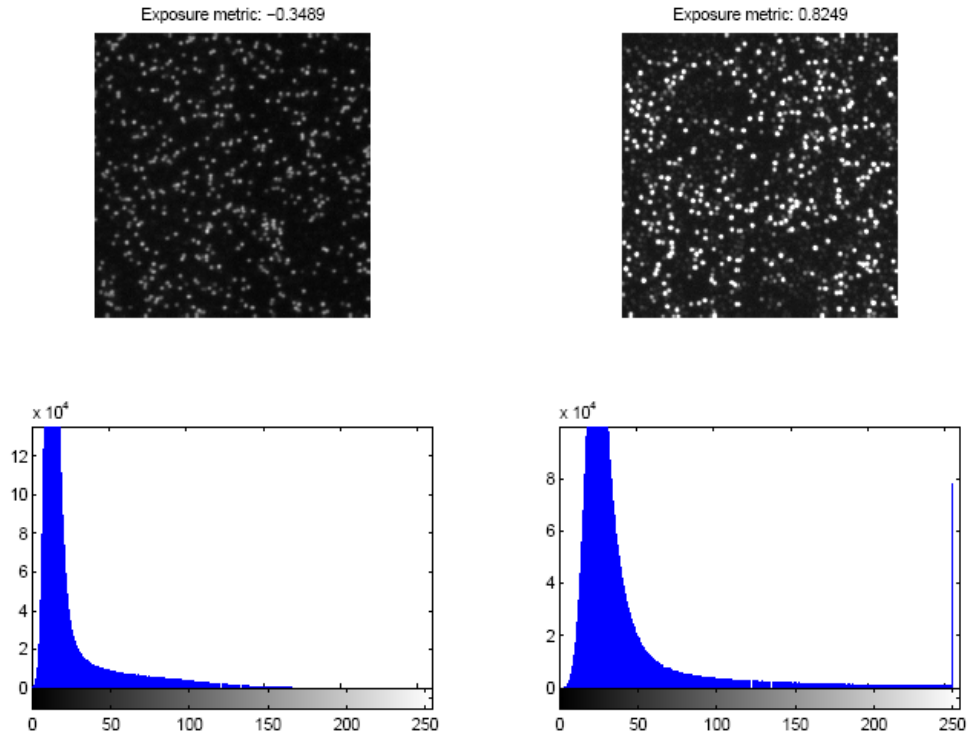



**Figure 17 Two examples of the exposure metric.**

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™ 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 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 software 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 Plus System is similar to Phred and the KB™ Basecaller software. (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 Plus 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

$$p_i = 1.00 \cdot 10^{-q_i/10}$$

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 18 shows such a validation.

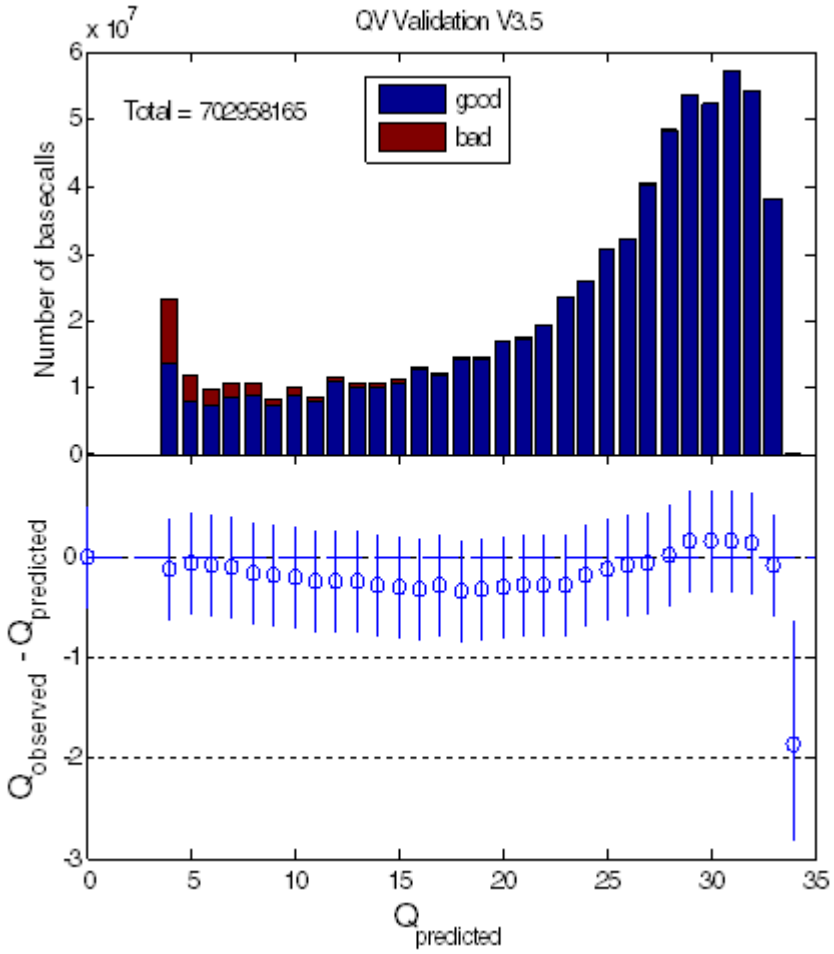



**Figure 18 Statistical characteristics of quality values (QV) for the SOLiD™ 3 Plus System.**

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

# Key files generated by primary analysis

**Table 8 Key files generated by primary analysis.**

| File Name | Description |
|---|---|
| *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.

The 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).

The format is:

```
TAG_ID
Quality Values:
```

For example:

```
>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 converts the data from the cycle-based format to the "read-based" ASCII formats (.csfasta and .qual).

.spch files use the hierarchical binary format HDF5. 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.

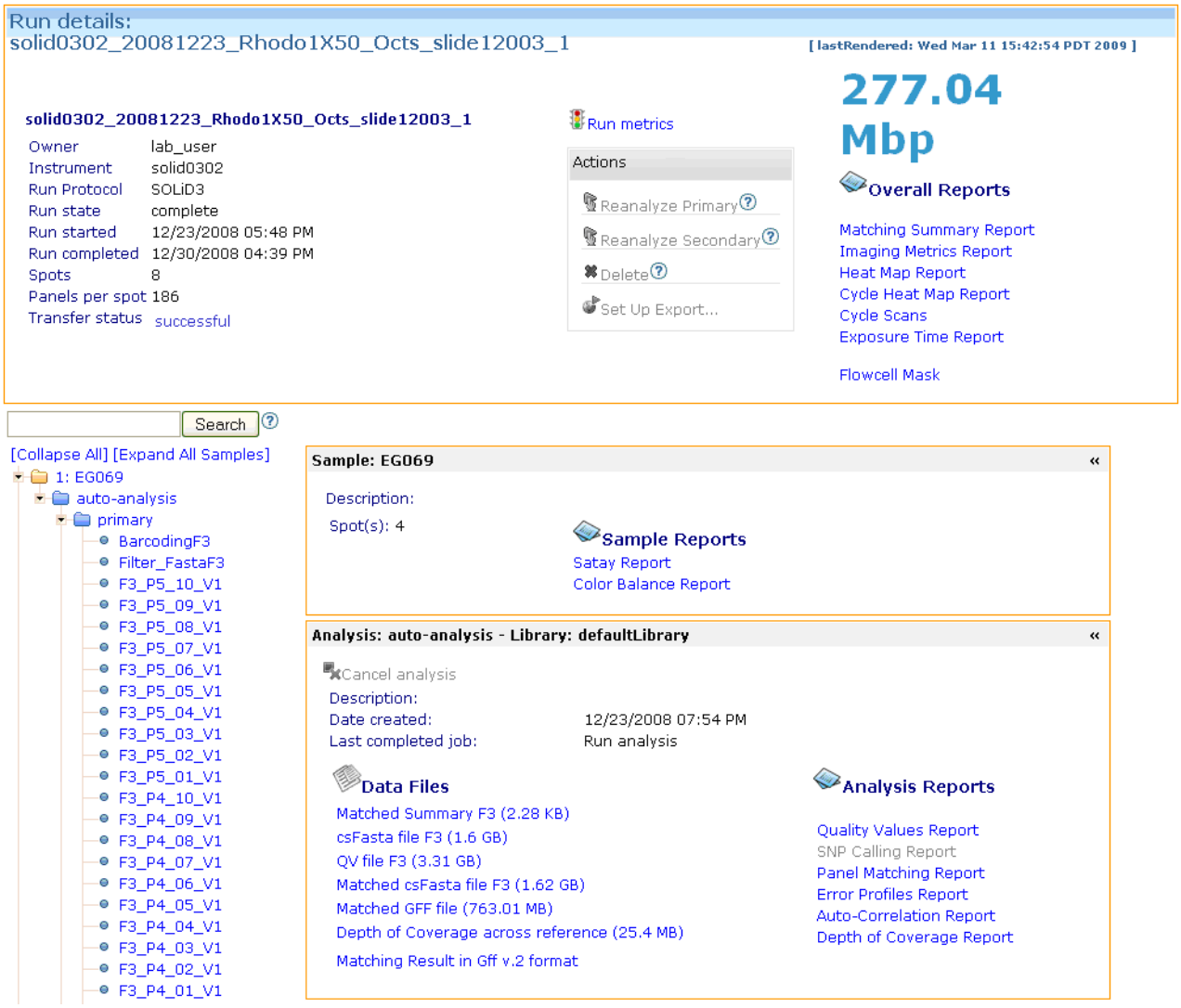
# Procedures to verify primary analysis status

To verify primary analysis status:

1. From SETS, click the History menu to access 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.

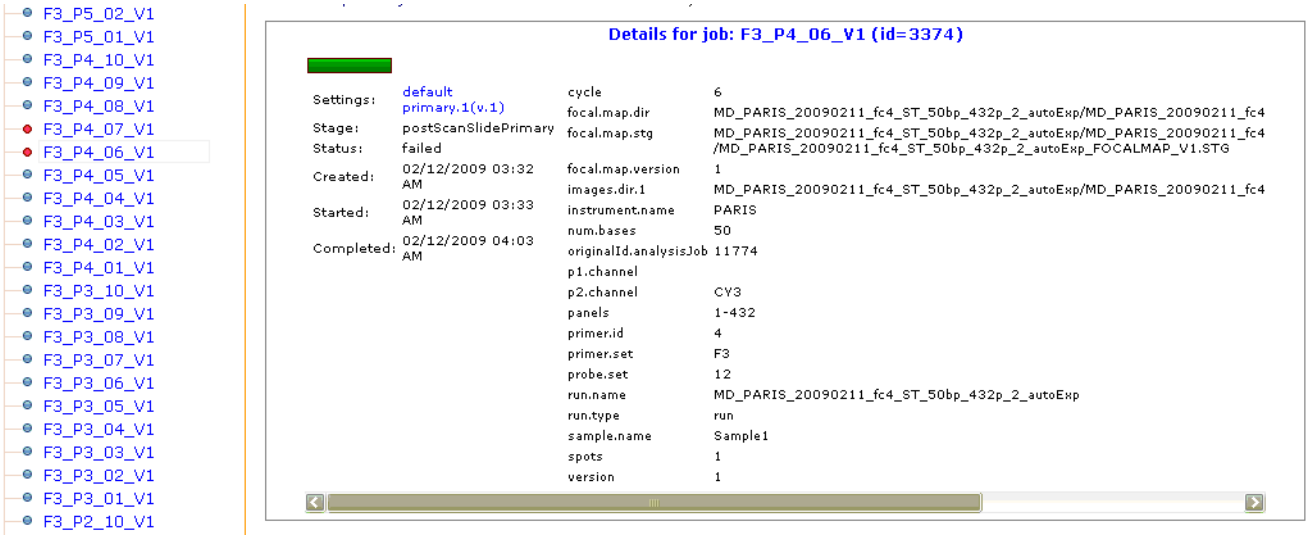
4. Check that all the analysis completed successfully. A **blue dot** before the analysis name means that the analysis completed successfully.

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

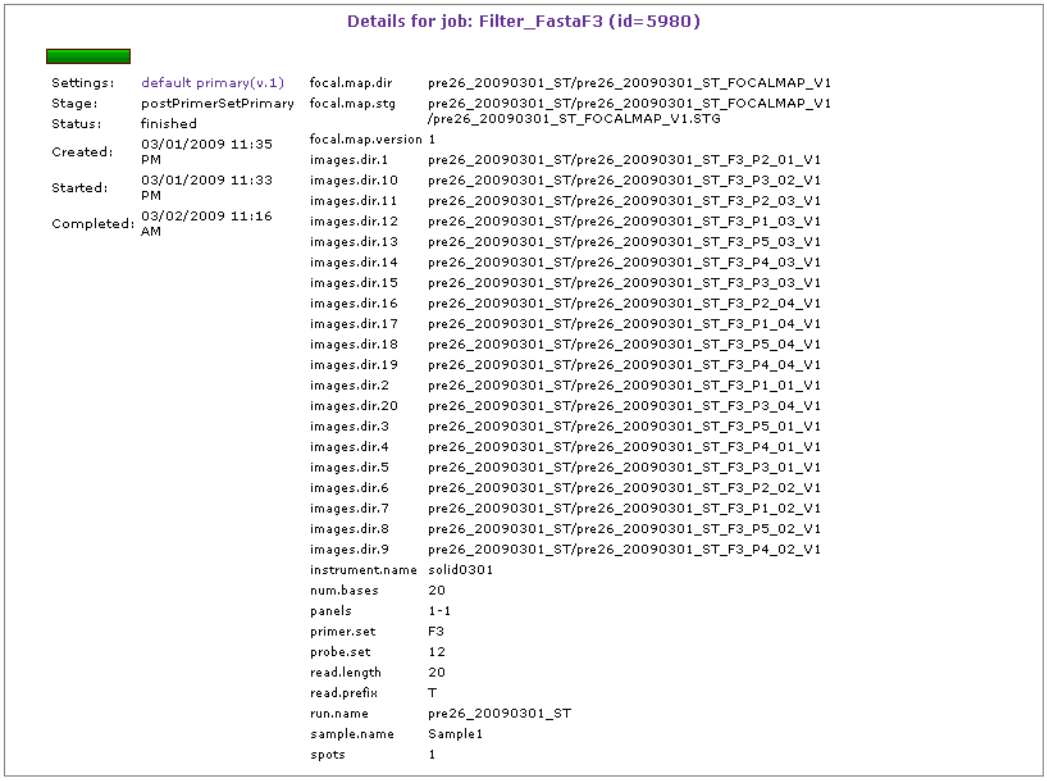
## Troubleshooting analysis failure

If there is a **red dot** before the analysis job name, the analysis job failed. 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. (See Figure 19 for an example.)



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 |

**Figure 19 Example of detailed information for one filter run.**

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

Sometimes, an analysis job might be reported as 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 following 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 an indication of whether they have too many errors, are too short, or are duplicates.

   The Totals (### p) line summarizes the number of panels found and the percentage of error, short, and duplicate reads.

   After this line is 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.

#### To re-analyze through SETS:

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

#### To perform primary analysis manually:

1. Go to the 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. Enter 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 4

This chapter covers:

# SOLiD™ system secondary analysis

## What is secondary analysis?

Secondary analysis is application-specific, and consists of a set of serial steps of modular workflow, called the *SAT analysis pipeline*. The 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 so on).
4. If the run is a mate-pair run, perform an additional mate-pair rescue (optional).

You can specify additional analysis pipeline steps by using SOLiD™ Experiment Tracking System (SETS) software. These steps can include analysis tools such as coverage analysis, sequencing bias analysis, and sequencing chemistry correlation analysis. For more information, see the *Applied Biosystems SOLiD™ SETS Software v3.5 Getting Started Guide* (PN 4444007).

You can perform sequencing analysis manually (command-line environment) using a parameter file.

## Secondary analysis overview

Table 9 lists the stages in secondary analysis, their inputs, and their outputs.

**Table 9 Secondary analysis stages, inputs and outputs.**

| Analysis Stage | Input | Steps | Output |
|---|---|---|---|
| **postPrimerSetSecondary** | • Raw sequence reads (.csfasta)<br>• Quality values (_QV.qual)<br>• Reference fasta sequences | 1. Filter reads by QV. (Optional)<br>2. Align reads to reference genomes.<br>3. Standard analysis (error profiles, coverage, gff, and sequencing correlation, for example).<br>4. Additional analysis. | • Matched color-space fasta (.csfasta.ma.[dd].[d])<br>• GFF v3 (.v3.gff)<br>• Error profiles per color call (tab)<br>• Consensus fasta sequence in base<br>• List of differences compared to reference sequences (GFF v3)<br>• Various analysis output (tab) |
| **postRunSecondary** | • Matched color-space fasta (.csfasta.ma.[dd].[d])<br>• Quality values (_QV.qual) Reference fasta sequences<br>• GFF v3 files (.v3.gff) | 1. MatePairing (if a MatePair run).<br>2. Standard analysis (error profiles, coverage, gff, and sequencing correlation, for example).<br>3. Additional analysis. | • MatePairing results (mates file)<br>• GFF v3 (.v3.gff)<br>• Error profiles per color call (tab)<br>• Consensus fasta sequence in base<br>• List of differences compared to reference sequences (GFF v3)<br>• Various analysis output (tab) |

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 reads based on their QV (quality values) before analysis (optional, off by default). 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, covered later in this chapter. The mapped reads are then processed through TagAnalysis, which converts each mapped read into a v3.gff entry. This entry contains the color-space read, the quality values, a base space translation, and any reference deviations that were found. SETS software then processes the gff v3 files to aggregate error profiles and other user feedback.

Applied Biosystems recommends that you perform complex genome analysis on a different computer system from the SOLiD™ instrument. Running SOLiD™ BioScope™ Software v1.0 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.

# Pipeline inputs, parameters, and outputs

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

- **Filtering**
- **Mapping group**: Aligns each read to the reference sequence.
- **Mapping Statistics**: Generates a file showing statistics for the mapping process.
- **MatePair**: Calculates valid mates based on insert size and performs rescuing.

## To run the secondary analysis pipeline

You run the secondary analysis pipeline through a command-line interface. Enter the following:

```
bioscope.sh [-l logfile] parameters.ini
```

or

```
bioscope.sh [-l logfile] analysis.plan
```

The first command line form, with the .ini extension, executes a single set of plug-in commands, specified in the file in the form <plugin-id> = 1. The plugin-id identifies an XML file that specifies which commands the plugin executes. A particular plugin-id can only appear once in such a file.

The second command line form, with the .plan extension, executes a list of files with the .ini extension. This command allows the same plug-in to be executed more than once in a run of bioscope.sh.

The following is an example of the first command line form:

```
bioscope.sh  F3-tag.ini
```

The .ini command reads instructions and parameters from a file named F3-tag.ini. It writes a process log to a file named bioscope.<timestamp>.log in the log subdirectory of the current directory.

The following command line

```
bioscope -l MyLogFile.log F3-tag.ini
```

will do the same, except the name of the log file will be MyLogFile.log, still in the log subdirectory:

- If the log file name is **fully qualified**, the log will be written to that file.
- If the log file name is **not fully qualified**, it is written to the directory where bioscope.sh was executed.

The bioscope.sh command creates a process on the current machine and waits for it to complete. If the system is configured to use a cluster batch manager, such as PBS/TORQUE, the plug-in steps specified in the F3-tag.ini file will be submitted to the cluster system. These steps may be submitted so that they run either in parallel or serially, depending on the plug-in design. Bioscope.sh waits for them all to complete, reporting on their completion. These sub-processes also write process logs in the log subdirectory.

The following is an example of execution of a .plan file:

```
bioscope.sh F3-R3-pair.plan
```

where the F3-R3-pair.plan file includes:

```
=F3-tag.ini
=R3-tag.ini
F3-R3-pair.ini
```

The two single tag analyses will be run in parallel. The F3-R3-pair.ini job will be run after both of the single tag jobs have completed.

When two or more parameter file entries in a plan file are preceded by an equal sign (=), those analyses are to be run in parallel. Otherwise, the analysis parameter files will be run sequentially. A single isolated = sign has no effect.

## 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 analysis proceeds similarly to the bioscope.sh case, but the SAT daemon process creates the parameter files from analysis parameters saved in the database and controls the submission of sub-jobs.

The analysis is specified by files with the .ini extension. These files contain parameter choices that are specific to the requested analysis.

The parameter files are not quite the same as "simple key=value properties files". They also support two features useful for use with bioscope.sh:

- Values in parameter files can include variables that reference the key of another parameter. For example:

  ```
  base.dir = .
  outputs.dir=${base.dir}/pairing
  ```

- Parameter files can use the keyword “import”. This keyword must be followed by a file name. The file name can be relative, but it **cannot** contain variables.

  When the file is read, the parameters in the import file are merged with the parameters in the parent file.

  If any parameters in the import file have the same key as one in the parent file, the **last one read** provides the value.

**Note:** This last point does not apply to plan files. The parameter files in a plan are independent unless they reference the same import file.

The import keyword is not active in the body of a plan file. However, the import keyword can be used in the parameter files in a plan to make them share parameter values.

Following is an example of a parameter file:

```
# Generated by ParameterFileWriter
# Date: 09/19/2009 05:15 AM
# Analysis job parameters
```

```
#1=Analysis Job Parameters
#2=Demon Properties
#3=Run Directories
#4=Solid properties
analysis.ini=/data/results/port/rjr_port_20090918_426hydra_3/Sample1/
jobs/postPrimerSetSecondary.F3.defaultLibrary.603/parameters.ini
analysis.job.dir.id=20090920074356102
analysis.job.id=34982
analysis.name=QV Filtering
analysis.pipeline=Class:com.apldbio.aga.analysis.secondary.pipeline.C
lusterPluginLauncherPipeline
analysis.results.dir=/data/results/port/rjr_port_20090918_426hydra_3/
Sample1/results.F1B1
analysis.run.dir=/data/results/port/rjr_port_20090918_426hydra_3/Samp
le1
analysis.sample.dir=/data/results/port/rjr_port_20090918_426hydra_3/S
ample1
analysis.stage.name=postPrimerSetSecondary
analysis.status.name=in_progress
analysis.work.dir=/data/results/port/rjr_port_20090918_426hydra_3/Sam
ple1/jobs/postPrimerSetSecondary.34982
app.id=single.tag
cluster.resourcemanager=TORQUE
colorcall.dir=/data/results/port/rjr_port_20090918_426hydra_3/Sample1
/results.F1B1/colorcalls
cycleplots.dir=/data/results/port/rjr_port_20090918_426hydra_3/Sample
1/results.F1B1/cycleplots
headnode.ip=solidmaster
instrument.name=port
job.id=34982
job.run.id=186
job.stage.name=postPrimerSetSecondary
library.name=defaultLibrary
logging.socket=foshtdvv09.na.ab.applera.net/167.116.6.89 8765
num.bases=50
output.dir=/data/results/port/rjr_port_20090918_426hydra_3/Sample1/re
sults/libraries/defaultLibrary/secondary.F3.20090918172655443
panels=1279-1704
plugin.id=classify.run
primer.set=F3
probe.set=12
rawseq.dir=/data/results/port/rjr_port_20090918_426hydra_3/Sample1/jo
bs/postPrimerSetPrimary.34979/rawseq
read.dir=/data/results/port/rjr_port_20090918_426hydra_3/Sample1/resu
lts.F1B1/libraries/defaultLibrary/primary.20090920074347062/reads
read.length=50
reads.result.dir.1=/data/results/port/rjr_port_20090918_426hydra_3/Sa
mple1/results.F1B1/libraries/defaultLibrary/primary.20090920074347062
/reads
reference=/share/reference/genomes/DH10B_WithDup_FinalEdit_validated.
fasta
reference.de.dir=/share/reference/genomes/doubleEncode
reference.dir=/share/reference
reference.genomes.dir=/share/reference/genomes
reference.name=DH10B_WithDup_FinalEdit_validated.fasta
run.name=rjr_port_20090918_426hydra_3
sample.id=442
sample.name=Sample1
scratch.dir=/scratch
spots=4
```

# Key input and output files in secondary analysis

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

Table 10 summarizes the input and output in all secondary analysis scenarios.

**Table 10 Key inputs and outputs for secondary analysis.**

| | Input for SAT pipeline | Required parameter files for stages | Output results |
|---|---|---|---|
| Fragment Library Data Analysis | • Primary analysis results in .csfasta, .QV.qual<br>• Reference sequence in .fasta | postPrimerSetSecondary | • Matching result, including mapped reads **only**, with base sequence in .v3.gff<br>• Matching result, including both mapped and unmapped reads, in .csfasta.ma.xx.xx<br>• Matching statistics file in .stats.txt |
| MatePair Library Data Analysis | • Primary analysis results in .csfasta and .QV.qual for each tag<br>• 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 .v3.gff<br>• Matching result for each tag, including both mapped and unmapped reads, in .csfasta.ma.xx.xx<br>• Matching statistics file (.stats.txt)<br>• v3.gff files for mates, sorted differently by genome position or mates<br>• F3_R3.mates |

## Input for the secondary analysis pipeline (SAT pipeline)

The following are input files 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**: This sequence must be validated by the reference_validation.pl script provided in the analysis pipeline.

## SOLiD™ system 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, specify a reference sequence file in fasta format in the parameter file for the SAT 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 the perl script `reference_validation.pl`. The script can be found on the SOLiD™ system cluster in the directory `/share/apps/corona/bin`. This step converts non-[ACGTN] characters to N and convert lower case characters to upper case. The script deals with multifasta inputs and IUB codes.

Run the validation script and put the validated reference sequence into `/share/reference/genomes`, to allow SETS to display the validated reference.

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

Usage:

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

Example:

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

## Pipeline output description

The results of SOLiD™ secondary analysis in are base sequences and color-space calls that have been mapped to the reference sequence. These data are ready for use in analysis.

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

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

Table 11 displays the directory structure for secondary analysis.

**Table 11 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_mapping | matching |
| classifier | classifier |
| qvfail | filtering |
| qvfiltered | filtering |
| qc_matching | single.tag or mate.pairs |

Key result files and their locations are listed in Table 12.

**Table 12 Key result files and their locations.**

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

# Filtering pipeline (Filtering step)

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).

The filtering pipeline uses the concept of minimum calls (rather than maximum errors) in the primary analysis settings to sequencing runs of differing lengths. The minimum calls approach to filtering catches 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 not lost. Mapping treats missing information (unless a mask is passed in) as an automatic mismatch. Additionally, a quality value of -1 is assigned for missing calls.

Consider these examples: If 25 colorcalls are considered 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 must 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. For example, in 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.

The filtering pipeline's progressive mapping yield a perfect 25.0 mapping result using the MatchingRandom feature (See "Mapping statistics" on page 68).

For Tag B, the spacing of the missing data ensures that the read will 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 used to yield a sufficiently screened tag set for secondary analysis without significantly skewing the read composition. (See Table 13.)

## Filtering Parameters

**Table 13 Filtering parameters.**

| Parameter name and key | Default value | Description |
|---|---|---|
| Pass All, filtering.pass.all | 1 | Boolean (0 or 1) value specifies 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. |

| Parameter name and key | Default value | Description |
|---|---|---|
| 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.<br>**Note:** Setting this value implicitly sets (and overrides) the QV Cutoff and Number of Bases To Use values. |

**Note:** Use Pass All = true/false to specify whether filtering is actually performed.

The pipeline must always be turned on and run as a dependency for downstream pipelines, even if no filtering is actually done. This requirement is caused by file system locations being created that are sourced by other later-executing pipelines.

- If the pipeline is on and Pass all is set to **true**, the primary analysis output is directly linked to the filtered output and no computation takes place.
- If the pipeline is on and Pass all is set to **false**, then the other parameters (Max hits, Min QV, Length or a QV Filtering mask) are parsed in order to filter based on the requested behavior.

If 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 are placed into the qvfail directory. Using the mask directly creates the screening behavior in which the mask has a fixed length but possibly a different QV at each position. The Max hits value is 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 differentiates between the two cases (See page 60) 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 is a min QV value of 9 with max hits value of 6 and a length of 10 which would create a mask of “9 9 9 9 9 9 9 9 9 9”. Any tag with 7 QV values less than 9 in the first 10 bases are dropped.

A value of 5 for max hits with a mask entry of “8 8 8 8 8 5 5 5 5 5” puts 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 example is summarized in Table 14.

**Table 14 Mask example.**

| 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<br>Max Hit = 1 | Any tag with more than one missing base in the 1st 25 bases<br>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<br>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 Plus System

Any SOLiD™ 3 Plus 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:

- 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.
- Clonal beads that do not match a reference sequence above a given level of stringency arising from reactors with:
  - 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. This an ePCR issue.
  - 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. This is a sequencing issue.
  - 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. This is a chemistry issue.
- 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).
- 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*. See Figure 20.

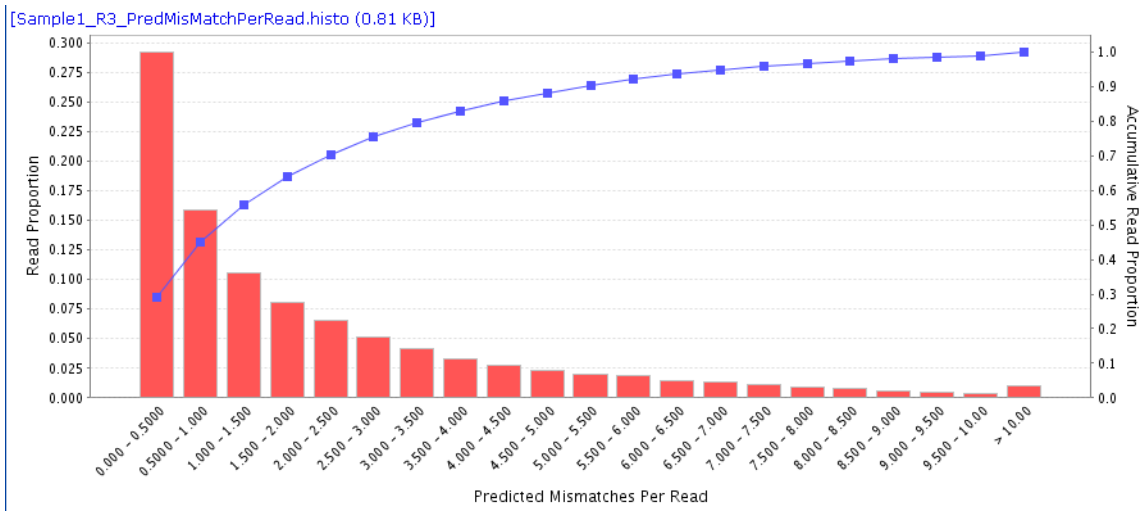



**Figure 20 Predicted mismatches per read graphic using the SOLiD™ Experiment Tracking System (SETS) quality value reports.**

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

There is a 10% difference between the estimated and actual values. In Figure 20, 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 noise leads to a situation where QV screening before matching does not cut sequences with a uniform composition, and it affects the sequence coverage observed.

# Mapping pipeline

## Basic sequence alignment

This section describes aligning SOLiD™ system 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 21).

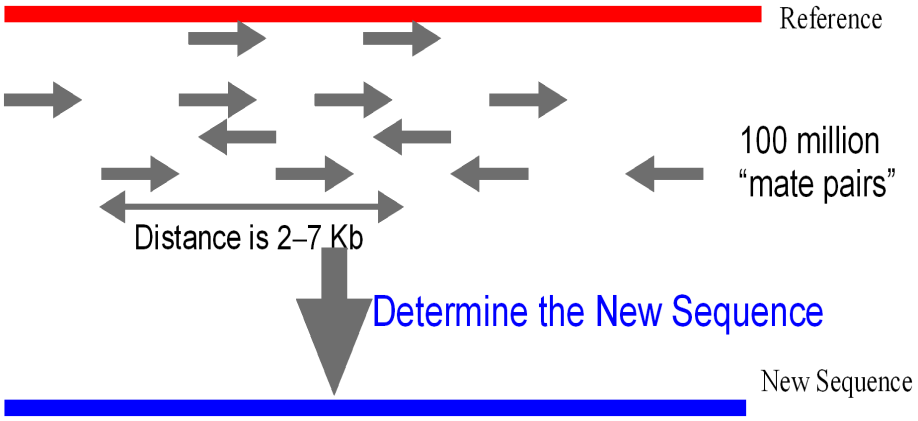



**Figure 21 Map short sequences to a reference sequence.**

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

## Alignment overview

### 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. A small number of color-space changes can lead to many changes in the translated bases, enhancing the ability to map. Conversely, low complexity sequences like dinucleotide repeats lose some complexity in color-space making these sequences 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

Programs such as BLAST (Basic Local Alignment Search Tool) use 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 at most 6 to find all hits with 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 examining only the number 1 positions in the discontinuous pattern.

For example, the pattern 111011101 determines a sliding window of nine colors, and uses only positions 1, 2, 3, 5, 6, 7, and 9 (a total of seven colors in the window) as keywords.

For a read length of twenty colors, the following 10 discontinuous patterns can be used to find all hits with 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 colors of the read are divided into 5 groups of 4 colors, 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 illustrates the advantage of using a discontinuous word pattern.

The alignment software Mapreads uses discontinuous word pattern search algorithms. For example, for a read length of 15, you can find all hits 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
```

Pattern 2 ensures that a mismatch within bases 6-10 can be identified. By iterating all 3 word patterns, a mismatch at any single color can be identified quickly. This example shows how an alignment with read length of 15 and 1 mismatch is performed using 10-base indexing. Such a combination of discontinuous words is called a matching schema in the SOLiD™ system analysis pipeline. Call this one `schema_15_1`.

The SOLiD™ system 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
```

## Mapping group rationale

The v3.5 mapping pipeline implementation focuses on scalability and speed without requiring significant changes at the lowest level of mapreads. The software can then treat mapreads as a series of transformation functions over the space of tag data, providing scheduling flexibility and allowing behavior changes based on intermediate results.

The inputs to the four parts of the mapping group are in a .csfasta file, as are the outputs (a single .ma file). The internal behavior does not matter to the downstream pipelines.

However, if **any** reference mapping is to be done, **all** four pipelines need to be on, even if they do not all execute. As with the filtering pipeline, assumptions are made about the file locations created by each of the component pipelines of the mapping group. Specifying pass all for any of the four mapping groups cause **no** computation to be performed.

## 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: Any reads that map to more than one place on the genome are discarded. Therefore, regions that are highly identical do not get much coverage using this setting. In general, longer and more identical regions (younger repeats and duplications) get less coverage. You can overcome this problem with mate-pairs, especially mate-pairs of different sizes (Figure 22). In addition, you can identify a majority of this type of sequencing gap by lowering the threshold of the unique mapping requirement.

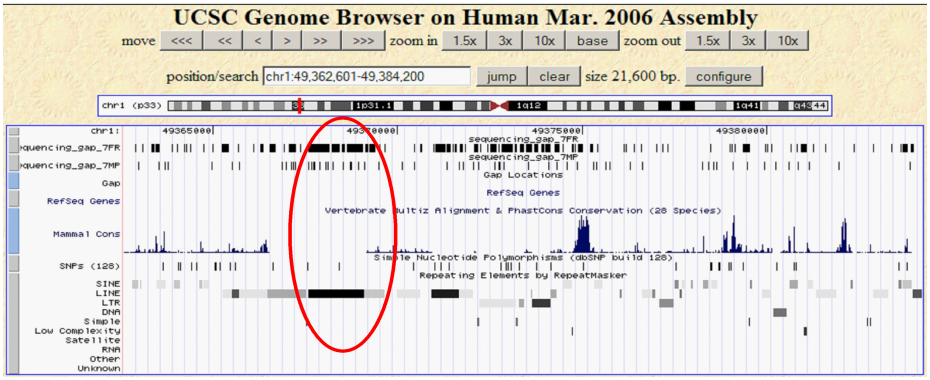



**Figure 22 Repetitive elements.**

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

In Figure 22, the circled region is covered by long interspersed nucleotide elements (LINEs). This region 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 the reference contains fewer copies than the sample, 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.

# Mapping statistics

The mapping pipeline automatically generates the file `mapping.stats.txt`, which contains a summary of the mapping statistics for the run.

Following is an example of the statistics file. (Some of the contents were left out for clarity.) The file includes:

- The reference sequence
- Statistics in the context of mapped reads

```
Mapping Statistics -- Local Alignment Method
test_S1_F3.csfasta.ma.50.6

429,162,041 total tags found
Reference Sequence
/share/reference/genomes/chromFa/hg18_validated.fasta, 3,080,436,051
bp

356,549,043 Mapped reads (83.08%) using parameter settings listed
below.
     mismatch penalty=-2.00
     clearZone=5
Mapped Reads
  Read Length 49
     0 mismatches   108,216,269 ( 30.35%)
     1 mismatches    31,793,019 (  8.92%)   140,009,288 ( 39.27%)
     2 mismatches    23,930,231 (  6.71%)   163,939,519 ( 45.98%)
     3 mismatches    10,963,550 (  3.07%)   174,903,069 ( 49.05%)
     4 mismatches     7,040,888 (  1.97%)   181,943,957 ( 51.03%)
     5 mismatches     3,729,585 (  1.05%)   185,673,542 ( 52.08%)
     6 mismatches     2,090,760 (  0.59%)   187,764,302 ( 52.66%)
     7 mismatches       920,470 (  0.26%)   188,684,772 ( 52.92%)
     8 mismatches       371,572 (  0.10%)   189,056,344 ( 53.02%)
     9 mismatches       106,244 (  0.03%)   189,162,588 ( 53.05%)
    10 mismatches        14,661 (  0.00%)   189,177,249 ( 53.06%)
  Read Length 48
     0 mismatches     6,151,048 (  1.73%)
195,328,297 (  54.78%)
     1 mismatches     3,624,505 (  1.02%) 9,775,553 (  2.74%)
198,952,802 (  55.80%)
     2 mismatches     3,082,604 (  0.86%) 12,858,157 (  3.61%)
202,035,406 (  56.66%)
     3 mismatches     2,068,511 (  0.58%) 14,926,668 (  4.19%)
204,103,917 (  57.24%)
     4 mismatches     1,507,657 (  0.42%) 16,434,325 (  4.61%)
205,611,574 (  57.67%)
     5 mismatches       922,022 (  0.26%) 17,356,347 (  4.87%)
206,533,596 (  57.93%)
     6 mismatches       524,568 (  0.15%) 17,880,915 (  5.01%)
207,058,164 (  58.07%)
     7 mismatches       237,819 (  0.07%) 18,118,734 (  5.08%)
207,295,983 (  58.14%)
     8 mismatches        90,335 (  0.03%) 18,209,069 (  5.11%)
207,386,318 (  58.16%)
```

```
     9 mismatches        21,737 (  0.01%) 18,230,806 (  5.11%)
207,408,055 (  58.17%)
  Read Length 47
     0 mismatches     6,466,180 (  1.81%)
213,874,235 (  59.98%)
     1 mismatches     3,793,888 (  1.06%) 10,260,068 (  2.88%)
217,668,123 (  61.05%)
     2 mismatches     3,220,291 (  0.90%) 13,480,359 (  3.78%)
220,888,414 (  61.95%)
     3 mismatches     2,167,468 (  0.61%) 15,647,827 (  4.39%)
223,055,882 (  62.56%)
     4 mismatches     1,571,416 (  0.44%) 17,219,243 (  4.83%)
224,627,298 (  63.00%)
     5 mismatches       947,271 (  0.27%) 18,166,514 (  5.10%)
225,574,569 (  63.27%)
     6 mismatches       511,993 (  0.14%) 18,678,507 (  5.24%)
226,086,562 (  63.41%)
     7 mismatches       213,152 (  0.06%) 18,891,659 (  5.30%)
226,299,714 (  63.47%)
     8 mismatches        72,895 (  0.02%) 18,964,554 (  5.32%)
226,372,609 (  63.49%)
     9 mismatches        13,021 (  0.00%) 18,977,575 (  5.32%)
226,385,630 (  63.49%)
<Material deleted>
3 Read Length 25
     0 mismatches     1,386,779 (  0.39%)
340,245,289 (  95.43%)
     1 mismatches     1,087,273 (  0.30%) 2,474,052 (  0.69%)
341,332,562 (  95.73%)
     2 mismatches     4,146,504 (  1.16%) 6,620,556 (  1.86%)
345,479,066 (  96.90%)
  Read Length 24
     0 mismatches     1,358,162 (  0.38%)
346,837,228 (  97.28%)
     1 mismatches     1,851,523 (  0.52%) 3,209,685 (  0.90%)
348,688,751 (  97.80%)
     2 mismatches     7,860,292 (  2.20%) 11,069,977 (  3.10%)
356,549,043 ( 100.00%)
Megabases of coverage 15,833.1661

Uniquely Placed Reads
  Read Length 49
     0 mismatches   100,879,799 ( 28.29%)
     1 mismatches    29,504,577 (  8.28%)   130,384,376 ( 36.57%)
     2 mismatches    20,843,357 (  5.85%)   151,227,733 ( 42.41%)
     3 mismatches     9,642,487 (  2.70%)   160,870,220 ( 45.12%)
     4 mismatches     5,693,098 (  1.60%)   166,563,318 ( 46.72%)
     5 mismatches     3,067,989 (  0.86%)   169,631,307 ( 47.58%)
     6 mismatches     1,593,850 (  0.45%)   171,225,157 ( 48.02%)
     7 mismatches       694,532 (  0.19%)   171,919,689 ( 48.22%)
     8 mismatches       255,951 (  0.07%)   172,175,640 ( 48.29%)
     9 mismatches        67,702 (  0.02%)   172,243,342 ( 48.31%)
    10 mismatches         9,716 (  0.00%)   172,253,058 ( 48.31%)
  Read Length 48
     0 mismatches     5,636,530 (  1.58%)
177,889,588 (  49.89%)
     1 mismatches     3,325,740 (  0.93%) 8,962,270 (  2.51%)
181,215,328 (  50.82%)
```

```
     2 mismatches     2,657,208 (  0.75%) 11,619,478 (  3.26%)
183,872,536 (  51.57%)
     3 mismatches     1,793,817 (  0.50%) 13,413,295 (  3.76%)
185,666,353 (  52.07%)
     4 mismatches     1,226,152 (  0.34%) 14,639,447 (  4.11%)
186,892,505 (  52.42%)
     5 mismatches       747,008 (  0.21%) 15,386,455 (  4.32%)
187,639,513 (  52.63%)
     6 mismatches       397,523 (  0.11%) 15,783,978 (  4.43%)
188,037,036 (  52.74%)
     7 mismatches       174,368 (  0.05%) 15,958,346 (  4.48%)
188,211,404 (  52.79%)
     8 mismatches        60,409 (  0.02%) 16,018,755 (  4.49%)
188,271,813 (  52.80%)
     9 mismatches        13,651 (  0.00%) 16,032,406 (  4.50%)
188,285,464 (  52.81%)
233,110,905 (  65.38%)
<Material Deleted>
Read Length 24
     0 mismatches       819,970 (  0.23%)
296,916,669 (  83.28%)
     1 mismatches     1,112,913 (  0.31%) 1,932,883 (  0.54%)
298,029,582 (  83.59%)
     2 mismatches     6,628,558 (  1.86%) 8,561,441 (  2.40%)
304,658,140 (  85.45%)
Megabases of coverage 13,742.7822
```

# GFF file overview

GFF (General Feature Format) is a record-based file format, where each line describes a single feature with a list of nine tab-delimited fields in a fixed order specified by the GFF3 specification. See "References" on page 83.

This section describes refinements specific to SOLiD™ GFF3 files. Each SOLiD™ system feature is a read or an alignment for a read.

- A GFF3 file consists of 9 columns or fields, separated by tabs, but not spaces or other white space.
- Undefined text fields are replaced with the "." character, as required by the GFF3 specification.
- Undefined numeric fields are replaced with -1 if there is no ambiguity.

**Table 15 GFF3 fields.**

| Column | Field name | Field description |
|---|---|---|
| 1 | seqid | The ID of the landmark used to establish the coordinate system for the current feature. The seqid will be the contig number of the alignment. |
| 2 | source | A free-text qualifier that describes the algorithm or operating procedure that generated this feature. The source will always be "solid". |
| 3 | type | The feature type, constrained to be a term from the SOFA sequence ontology. See "References" on page 83. For v3.5, this will always be "read". |
| 4 | start | The start of the feature, in 1-based integer coordinates, relative to the landmark given in column 1. Start is *always* less than or equal to end. |
| 5 | end | The end of the feature, in 1-based integer coordinates, relative to the landmark given in column 1. End is *always* greater or equal to start. |
| 6 | score | The score of the read. |
| 7 | strand | The strand of the feature, in this case the strand of the reference to which the alignment mapped:<br>• '+' for positive strand (relative to the reference)<br>• '-' for the reverse strand<br>• '.' for unmapped reads |
| 8 | phase | Applies only to features of type "CDS". Always '.' for SOLiD™ files. |
| 9 | attributes | A list of semicolon-separated feature attributes in the format key=value. Nearly all information for SOLiD™ system analysis is stored in this column. |

## Convert to .gff v3 file

MaToGff3 writes a GFF v3 file from a single-fragment match (.ma) file, and its associated quality value file.

Usage:

```
MaToGff3 <ma_file>
      [--clear=<int>]
      [--convert=[reads|hits|mapped|unique]
      [--first]
      [--mmp=<double>]
      [--out=<outputGff3filename>]
      [--qvs=<qualityValueFile>]
      [--sort]
      [--tempdir=<temporary directory>]
```

**Table 16 MaToGff3 options.**

| Options | Description |
|---|---|
| `--clear=cz` | The requested "clear zone". See `--convert=unique`. Defaults to 5 if not specified. |
| `--convert=unique` | MaToGff3 reports only reads that mapped to the reference uniquely. If a read has no hits, then it does not map anywhere, uniquely or otherwise.<br>• If it has exactly one hit, then it does map uniquely.<br>• For reads that have more than one hit, let s1 be the hits' highest local alignment score, and s2 be their second highest. Then alignment1 is deemed to map uniquely if s1 – s2 > cz, where cz is the clear zone defined by the `--clear` option.<br>Defaults to unique if not specified. |
| `--convert=mapped` | MaToGff3 reports only reads, at most one per bead in the ma_file, that mapped to the reference. |
| `--convert=reads` | MaToGff3 reports exactly one read, each on its own line, for every bead in the ma_file, mapped or not. |
| `--convert=hits` | MaToGff3 reports a GFF entry (line) for every map in the ma_file. Note that this action may result in multiple GFF lines per bead. To uniquely identify a line, you must specify not just its read id, but also its start, its strand, and possibly its end (if the alignments have variable length). |
| `--first` | Affects the behavior of `--convert=reads` and `--convert=mapped`. Overrides random selection by causing MaToGff3 to report the **first** alignment instead, that is, the alignment with the least "start" position. |
| `--mmp=<double>` | The mismatch penalty used to calculate the local alignment score in the definition of convert="unique". It must be a negative value. Ideally, it is the same value used for MapReads. Defaults to -2.0 if not specified. |
| `--out=<gff3File>` | The file to which output should be directed. Defaults to standard out if not specified. |
| `--qvs=<qvFile>` | A file of quality values for the color calls in the reads. |

| | |
|---|---|
| `--sort` | Sorts the reads by increasing numerical position: contig, then start, then end, then (by text) strand. |
| `--tempdir` | The directory to use for temporary files. MaToGff3 deletes any temporary files it creates when done, but not the directory. Defaults to "/scratch/solid" if not specified. |

For each read, MaToGff3 reports the position of the best alignment to the reference (that is, the alignment with the highest local alignment score). If MaToGff3 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 MaToGff3 to report the first alignment in the .ma file instead. FASTA and GFF3 reads are 2-bp encoded strings.

Every alignment ("hit") of a read to the reference has an associated local alignment score, s, given by:

```
s = (L - M) + p*M
```

where

`L` = length of the alignment in number of colors

`M` = number of color mismatches in the alignment

`p` = mismatch penalty from the `--mmp` option

Note that this is the same as

```
s = L - c*M
```

where `cost c = (1 - p) and c > 1`.

## AnnotateGff3Changes

AnnotateGff3Changes reads in a GFF v3 file of color reads and a FASTA file that contains the reference base sequence to which the reads have been mapped.

For each read, AnnotateGff3Changes creates and adds attributes:

- b = The base space representation
- r = The reference color call at mismatches
- rb = The reference base call at mismatches
- s = The mismatch annotations

AnnotateGff3Changes writes its result (a new GFF file) to <outputGff3filename>.

Usage:

```
AnnotateGff3Changes <GFF_file> <FASTA_ref_file>
       [--tints=a[g[y[r]]]] [--b=true|false] [--cn=true|false]
       [--correctTo=reference|consistent|missing|singles|qvThreshold]
       [--iubsTo=[missing|first|haplotype]]
       [--out=<outputGff3filename>]
       [--qvThreshold=<threshold>]
```

**Table 17 AnnotateGff3Changes parameters.**

| Parameters | Description |
|---|---|
| `<GFF_File>` | The name of the input, unannotated GFF file. |
| `<FASTA_ref_File>` | The name of the reference file. |
| `--tints=agyr` | Represents any number (four in this example) single-tint annotations.<br>• a = Isolated single-color mismatches (grAy)<br>• g = Color position that is consistent with an isolated one-base variant (e.g. a SNP)<br>• y = Color position that is consistent with an isolated 2-base variant<br>• r = Color position that is consistent with an isolated 3-base variant and so on, for any number of adjacent base variants.<br>You can use any tints for a, g, y, and r; it is the position that matters. You do not have to extend as far as r, nor do you have to stop there (if biologically realistic).<br>Defaults to agy if not specified. |
| `--b=[true\|false]` | If true, AnnotateGff3Changes prints the b attribute, which is the base-sequence corresponding to the corrected color calls. It has no effect if `--ref` is not also specified.<br>Defaults to true if not specified. |
| `--correctTo=[reference\|consistent\|missing\|singles\|qvThreshold]` | Specifies how to correct the color calls:<br>• reference: Replaces all read-colors annotated inconsistent (i.e. a or b) with the corresponding reference color.<br>• missing: Replaces all inconsistent read-colors with '.'. These will translate to x in the base space representation, attribute b.<br>• singles: Replaces all single inconsistent colors (i.e. those annotated a or b and not adjacent to another b) with the corresponding reference color. Replaces all other inconsistent colors with '.'.<br>• 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.<br>• qvThreshold: A scheme combining the four above, based on the specified qvThreshold.<br>Defaults to missing if not specified. |
| `--out=<gff3File>` | The file to which output should be directed.<br>Defaults to standard out if not specified. |
| `--qvThreshold=<k>` | Color calls with a quality value at or above this value are considered good. Used by `correctTo=qvThreshold`.<br>Defaults to 15 if not specified. |
| `--cn=[true\|false]` | Contig names. Writes a table-of-contents to the head of the file with lines such as:<br>##contig 3 name-of-the-third-contig (i=3)<br>Defaults to true if not specified. |

| | |
|---|---|
| `--iubsTo=[missing\|first\|haplotype]` | Specifies how to treat ambiguity codes (IUB codes other than A, C, G, or T) in the reference sequence:<br>• missing: Treats the code (e.g. R) as if it were X. This encodes to a missing color. E.g. color(CR) = color(RC) = color(CX) = '.'<br>• 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 selection will naturally have a bias towards alphabetically low bases.<br>• haplotype: Selects a haplotype that minimizes the number of mismatches with the current read. E.g. ref=SAAWWTCNVTH with read=G2033320312 yields GAATATCCGTC, which encodes as G2033320312, and has no mismatches, while ref=SAAWWTCNVTH with read=G0000000000 yields GAAAATCCCTT, which encodes as G2000320020 and has 4 mismatches.<br>Defaults to missing if not specified. |

## MatesToGff3

MatesToGff3 reads a SOLiD™ v3.5 (.mates) file, together with the quality value files for the reads, and a file containing the reference sequence to which they have been mapped.

From these, MatesToGff3 creates one or two SOLiD™ v3.5 GFF3 files.

The output gff3 files may be fragment-ordered or mates-ordered.

- In fragment order, the reads are listed in order of non-decreasing start position.
- In mates order, the F3 reads are listed in order of non-decreasing start position, and each R3 read immediately follows its corresponding F3 read.

```
Usage: MatesToGff3 <matesIn.mates>
[--f3qv=<F3_QV.qual>]
[--r3qv=<R3_QV.qual>]
[--ref=<ref.fasta>]
[--fout=<sortedFragments.gff]
[--mout=<sortedMates.gff]
[--tints=a[g[y[r]]]]
[--b=true|false]
[--cn=true|false]
[--tempdir=<temporaryDirectory>]
[--correctTo=reference|consistent|missing|singles|qvThreshold]
[--iubsTo=[missing|first|haplotype]
[--qvThreshold=<t>]
```

where:

**`--matesIn.mates`**
The required input SOLiD™ v3.5 mates file.

**`--f3qv`**
The quality value file for the F3 fragment color calls. Not added if not specified.

**`--r3qv`**
The quality value file for the R3 fragment color calls. Not added if not specified.

**--ref**
The file containing the reference sequence. MatesToGff3 will annotate the color calls only if `--ref` is specified.

**--fout**
The output GFF3 file for all fragments in order of increasing start point.

**--mout**
The output file for all Mates in order of increasing start point of the F3 fragment. The fragment mated to an F3 fragment appears immediately on the next line. MatesToGff3 writes these data to standard out if and only if both `--mout` and `--fout` are not specified.

**--tempdir**
The temporary directory that MatesToGff3 should use. Defaults to `/scratch`.

The remaining options `b`, `correctTo`, `cn`, `iubsTo`, `qvThreshold`, and `tints` are described in the section "AnnotateGff3Changes" on page 73.

## Specification of SOLiD™ GFF3 v3.5 files

In addition to feature lines, metadata lines and comment lines are also supported by the GFF3 format. Comment lines begin with a # character. If a # character appears anywhere on a line, then the rest of that line is a comment. A comment or comment line may appear anywhere in the file and will ordinarily be ignored by an application reading the file, with one important exception:

- Metadata lines begin with ##, and as such are a special kind of comment. They are data that apply to all of the features in the file. These metadata need not be ignored by all applications.

### Metadata

SOLiD™ GFF3 files use the following metadata:

**##gff-version**

This is "##gff-version 3".

**##solid-gff-version**

This is "##solid-gff-version 3.5".

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

A comment describing the version of the program that produced this file. The source and the version text should not include any white space. Example:

```
##source-version AnnotateGff3Changes.javav0.1
```

**##date <date>**

The date that the file was generated, in *yyyy-mm-dd* format. The following example designates 17 July 2009:

```
##date 2009-07-17
```

**##reference-file <reference name>**

The reference name is the name of the reference to which all reads in this file were aligned. Example:

```
##reference-name DH10B_WithDup_FinalEdit_validated_5contigs.fasta
```

**##history <command line>**

These comment lines allows us to record 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™ GFF3 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. Example:

```
##history map Sample1_F3.csfasta myReference.fasta
##history MaToGff3.java --convert=unique –sort Sample1.ma
##history AnnotateGff3Changes.java Sample1.gff myReference.fasta
```

**##time <current time>**

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

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

**##color-code <code string>**

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 2-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 <primer set>=<base>[,<primer set>=<base>]***

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

```
##primer-base F3=T,R3=G
```

**##max-num-mismatches <count>**

The largest number of mismatches (in color-space) for any reported hit (alignment) in the file. Example:

```
##max-num-mismatches 9
```

**##max-read-length <count>**

The largest number of positions (colors and the leading base) for any read in the file. Example:

```
##max-read-length 50
```

**##line-order <order>**

<order> has one of three values: fragment, mates, or unordered.

The GFF3 standard allows lines in **any** order. SOLiD™ GFF3 v3.5 files, however, should have metadata **before** feature lines. Feature lines should be ordered by increasing seqid (contig index), then start position, end position, strand, panel number, x coordinate, y coordinate, and finally primer set id of the reads (see Attributes below). This is fragment-ordered, indicated by:

```
##line-order fragment
```

Unmapped reads are an exception. Even though their contig index is -1, they appear at the (tail) end of the file.

**Mate-paired data**

Mate-paired data can also be fragment-ordered, one line for the forward read and a second for the reverse, again indicated by <order> = fragment. Some applications, however, may prefer an alternative, called mates-ordered, where the forward (e.g. F3) reads are fragment-ordered as before, but each reverse (e.g. R3) read appears immediately after its corresponding forward read. This ordering is indicated by:

```
##line-order mates
```

If the data are not known to be either fragment-ordered or mates-ordered, they are unordered, indicated by;

```
##line-order unordered
```

## Fields

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

**`seqid`**

The ID of the landmark used to establish the coordinate system for the current feature. This will be the contig number of the alignment. Reads without an alignment will have a -1 in this column.

**`source`**

This value is always “solid”.

**`type`**

For SOLiD™ Software v3.5, this value is always “read”. Other possible types, such as deletion or inversion, can be used when they become available.

**`start`**

The inclusive start point of the alignment. in the 1-based base-space reference sequence, indexed from 5′ to 3′. See `end` for details. The alignment may be the entire read, or more generally, a subsequence with an offset specified by the `o` attribute.

**`end`**

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

Per 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, as in:

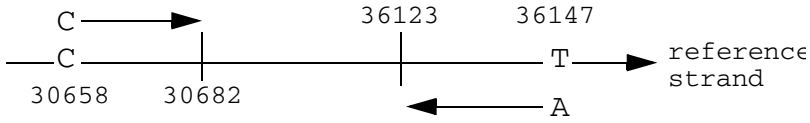
```
start    end      strand   [attributes]
30658    30682    +        g=C0103112003130213233110 32
```

then the initial C aligns with position 30658 of the reference strand, which should be a matching C; the base following 0, which is 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=A322121003310310232103022
```

then the A aligns with position 36147 of the reference strand, which should be a complementing T; the base following 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 base following the last 2 aligns with position 36123.




**score**

A summary quality score for the read, not just the alignment, 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 Attribute q). Note that high scores correspond to high quality reads.

**strand**

Either +, if the read aligns to the forward, nominal, or given strand (i.e., the sequence provided as the reference) or - (minus) if the read aligns to the other strand.

**frame**

The SOLiD™ System does not use the frame and therefore always sets it to “.”

**Attributes**

A semi-colon separated list of key-value pairs of the form “key=value”. All attributes are optional, unless marked as Required. There are only three such required attributes: keys aID, at, and g, as defined below.

**aID**

Required. This is the bead identifier (*panel_x_y*) where panel is an integer index of the panel on which the bead falls, and x and y are integer coordinates of the bead location. For example:

```
aID=8_1727_1389
```

Note that aID is **not** a unique identifier for this alignment. In fact, even aID together with the primer set ID for this read would identify only the read, not the alignment. Only the term ID is designated by the GFF3 standard to mean an identifier for this line that is unique within the file.

**at**

Required. The adaptor type, or primer set for this read. Example:

```
at=F3
```

**b**

The base-space representation of the alignment, which is not necessarily the entire read. For example, the read below has 50 bases, the alignment is 33 bases long, and starts with the first base (o=0).

```
b=GTTCTGATTgCCGCGTTAATTGCAGGTTTaGTC;
g=G10221230130333103030131201000321200000300003000 10;
o=0;
```

Lower case characters indicate base calls that differ from the reference.

**c**

The category of the mate-pair for this read. It is defined and present only for mate-pair runs. Category is written as a three-letter code that describes the relative mapping of the two reads that make up a mate-pair.

- The first letter refers to the reference strand to which the reads mapped: A same reference, same strand; B same reference, different strands; C different references (or contigs); D one read did not map; E one read is missing.
- The second letter refers to the ordering of the alignments within the strand: A correct ordering (e.g. R3 appears before F3 on its contig); B incorrect; * not applicable.
- The third letter refers to the insert size: A acceptable, B too small; C too large; * not applicable.

The following combinations are supported:

**Table 18 Supported mate-pair categories.**

| Category | Strand | Orientation on contig | Insert size |
|---|---|---|---|
| AAA | Same | Correct | Acceptable |
| AAB | Same | Correct | Too small |
| AAC | Same | Correct | Too large |
| ABA | Same | Incorrect | Acceptable |
| ABB | Same | Incorrect | Too small |
| ABC | Same | Incorrect | Too large |
| BAA | Different | Correct | Acceptable |
| BAB | Different | Correct | Too small |
| BAC | Different | Correct | Too large |

| | | | |
|---|---|---|---|
| C** | Different conduit | N/A | N/A |
| D** | One unmapped mate | N/A | N/A |
| E** | One missing mate | N/A | N/A |

**g**

Required. The color-space string for this read, written from 5′ (the bead end) to 3′. In addition, the first (5′ most) nucleotide of the DNA sequence of the read is prepended to the string. This, together with the coding algorithm specified by the color-code metadata tag, allows any application to reconstruct the DNA sequence corresponding to this read. By referring to the strand field, the application can reconstruct the color-call substring (and the corresponding DNA subsequence) for the other strand of the reference as well. This corresponding DNA sequence is identical to that of the fragment being sequenced only if all colors and the leading base are also correct, which is an attribute of the application writing this file, not of the file format. The SOLiD™ system uses the b attribute to store the algorithmically corrected base sequence.

It is important to note that FASTA files generated by the SOLiD™ instrument store their data in a different format. There, the color-string 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, simply by reconstructing the first base of the read. For example, it would convert 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.

**me**

The Mapping Quality is an integer between 0 and 150 inclusive, which describes the quality of the mapping for this alignment.

**o**

The alignment offset in the read, that is, the number of leading colors in the read sequence (the g attribute) that are **not** part of the color alignment. For example, if the read is g=G0123, and the offset is o=0, then the alignment starts at the beginning of the read; the color alignment is 0123, and the base alignment is b = GGTCG. If o=2, then the 0 and 1 are not part of the alignment; the color alignment is 23, and the base alignment is b = TCG. In all cases the length of the base alignment is given by LB = end – start + 1, and the length of the color alignment is given by
LC = end – start.

**p**

This mappability ambiguity measure gives a count of 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. Definitions:

- L - Length of the alignment.
- k - Number of mismatches in the alignment. Note $k \leq L$.
- m - Least number of mismatches over all alignments for this read.

- $N_k$ - Number of reference positions where the read aligns with exactly k mismatches. Note that $N_k = 0$ for all $k < m$, because m is the least number of mismatches for this read. Assume that $N_k = 0$ for all k greater than the largest number of mismatches reported by the u attribute. For example, if u=0,0,3,1, then N0 = 0, N1 = 0, N2 = 3, N4 = 1, and $N_k = 0$ for all $k > 4$.
- $X_{Lk}$ - The expected number of alignments with exactly k mismatches as a multiple of the expected number of alignments with 0 mismatches.

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

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 p = 1, indicating an ideal unique match.

Even if our unique alignment has two mismatches, then L=25, m=2, $N_m = 1$, and again p = 1 (the $X_{Lm}$'s cancel out).

If the read has two perfect alignments, then L=25, m=0, and $N_m = 2$. Again, $X_{L,m} = X_{25,0} = 1$. But now p = 2, indicating an ambiguous match. In general, if the read maps perfectly to the reference in n places, i.e. $N_0 = n$, then the mappability ambiguity measure p will also be n.

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

**pq**

The Pairing Quality is an integer between 0 and 300 inclusive, which describes the quality of the pairing for this alignment. It will only appear in files containing mate-pair data. Both alignments in a mate-pair will have the same value for pq.

**q**

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

**r**

The Reference color at mismatches attribute is a comma-separated list of `{position}_{ref_color}` for all color-calls in the alignment sequence that differ from the reference sequence. The 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.) Although coordinates are with respect to the entire read, only calls within the alignment will appear in r.

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

**rb**

The Reference Base at mismatches attribute is a comma-separated list of `{position}_{ref_base}` for all base-calls (attribute b) that differ from the reference sequence. Position is 1-based relative to the sequence specified in the b attribute, again, the prepended base has position 1 and the first color in g has position 2.

For example, `rb=38_A,39_C` means that the reference has base A at position 38, and base C at position 39.

**s**

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, `a5,g7,g8` means "format the color call at position 5 gray, format call 7 green, and format call 8 green". The SOLiD™ system follows this convention:

- a - (grAy) is an isolated mismatch: Neither the color-call on its left nor the call on its right is a mismatch, and it is not consistent with an isolated run of base changes.
- g - (Green) is a valid adjacent mismatch: Together with the adjacent mismatch on its left or right, it could correspond to an isolated SNP,
- y - (Yellow) is a color call that is consistent with an isolated 2-base change. In general, these will be mismatches. But a conserved color between two mismatches is also a possibility.
- r - (Red) is a color call that is consistent with an isolated three-base change.
- b - (Blue) is an invalid adjacent mismatch; it is any other mismatch.

**u**

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 within the alignment region. For example, `u=0,3,15` means 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.

## References

The SOLiD™ system GFF3 format v3.5 has been designed to adhere to the GFF3 specification, available at:

**http://www.sequenceontology.org/gff3.shtml**

and uses the sequence ontology from:

**http://www.sequenceontology.org/gff3.shtml**

There is also a publicly available GFF3 validator at:

**http://dev.wormbase.org/db/validate_gff3/validate_gff3_online**

# MatePair pipeline (Pairing step)

## 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. Allowing mismatches up to $m$ for F3 and $m$ for R3, 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 $(2m - n)$.

For example, if the pipeline allows up to 3 mismatches for the alignment and one of the tags has 1 mismatch, the pipeline allows 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.

The exact number of mismatches is critical.

To map a fragment library, the software considers sequences that can be mapped uniquely. In these applications, you 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, set the limits for a large genome higher, to approximately 100. A read that has many hits is likely to be in the repeat region. 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 21). 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 you set 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 23). This strategy enables a faster speed for mapping MatePair data.

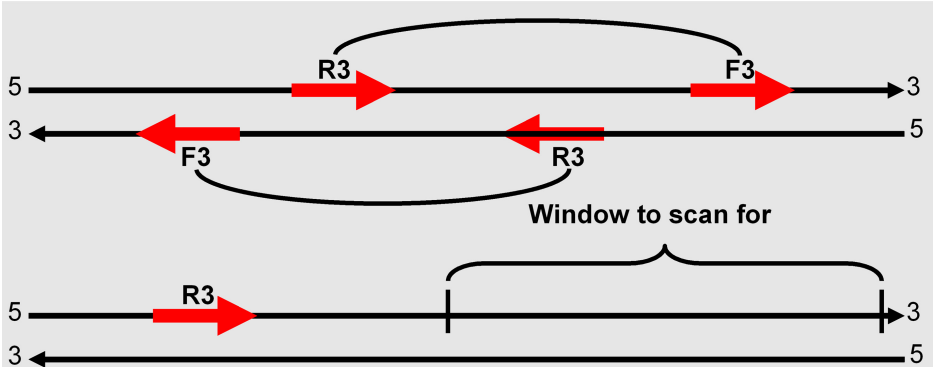



**Figure 23 Schematic of MatePair data.**

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. 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.

## Parameters

Table 19 displays the MatePair pipeline parameters.

**Table 19 MatePair pipeline parameters.**

| Parameter name and key | Default value | Description |
|---|---|---|
| Minimum Insert<br>insert.start | 1800 | Minimum insert size that defines a good mate. |
| Maximum Insert<br>insert.end | 3200 | Maximum insert size that defines a good mate. |

| Parameter name and key | Default value | Description |
|---|---|---|
| Rescue Level<br>mate.pairs.rescue.level | 10 | Usually two times the mismatch level. |
| Use Pairing<br>mate.pairs.use.pairing | true | |
| Pairing Mismatch Threshold<br>gff.clear | 1000 | |
| Output Reads in Base Space<br>mate.pairs.gff.output.basespace | 1 | Include corrected base space read in GFF file. |
| Color Correction Strategy<br>mate.pairs.gff.option.correctTo | consistent | Specifies how to correct the color calls for output reads in base space.<br>• "reference" replaces all read-colors annotated inconsistent (i.e. a or b) with the corresponding reference color.<br>• "missing" replaces all inconsistent read-colors with "." These translates to x in the base space representation, attribute b.<br>• "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.<br>• "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<br>mate.pairs.gff.option.trimMod | 5 | Requests AnnotateGff3Changes 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<br>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.<br>• "missing" treats the code (e.g. R) as if it were X. This encodes to a missing color.<br>• "first" selects an arbitrary haplotype from the reference by treating each ambiguity as equivalent to the pure base in the set that comes first in the alphabet. This action naturally has a bias towards alphabetically low bases.<br>• "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
- 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 data 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 data 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. Table 20 describes the columns and contents of the file.

**Table 20 F3_R3_mates file**

| ##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 |

| category | F3AnchorMM | R3AnchorMM | F3AlignedLen | R3AlignedLen | F3StartInRead | R3StartInRead | QV |
|---|---|---|---|---|---|---|---|
| AAA | 1 | 2 | 48 | 27 | 0 | 0 | 32 |
| AAA | 0 | 1 | 49 | 46 | 0 | 0 | 71 |
| AAB | 1 | 0 | 24 | 35 | 0 | 0 | 33 |
| AAB | 2 | 2 | 38 | 24 | 1 | 0 | 26 |
| AAA | 0 | 0 | 49 | 40 | 0 | 0 | 75 |
| AAA | 0 | 0 | 46 | 36 | 0 | 0 | 56 |
| AAB | 2 | 1 | 45 | 29 | 0 | 0 | 30 |

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

**Table 21 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 |
| 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 |
| D** | One of the tag matching lists is empty and the other has one hit. |
| E** | The pair is missing one tag. That is, the bead was not found in one of the sequencing runs and the existing tag has a single hit to the reference. |

- 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. (See Figure 24).

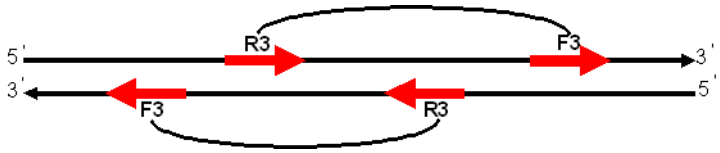



**Figure 24 F3 and R3 showing correct orientations.**

- 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.
  - B = incorrect order. (See Figure 24).
- 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.

# Reports

This section describes the reports produced by the SOLiD™ system. These reports include:

- qc_correlation
- qc_coverage
- qc_mates
- s_mapping

For additional information about how to generate and interpret reports, refer to the *Applied Biosystems SOLiD™3 SETS Software v3.5 Getting Started Guide* (PN 4444007).

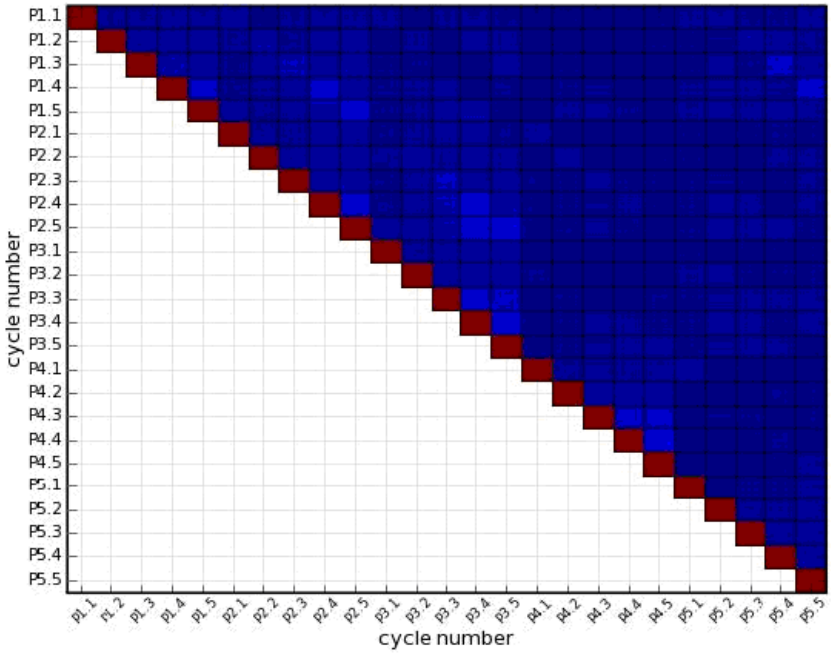
## qc_correlation

**`[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 being 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 |



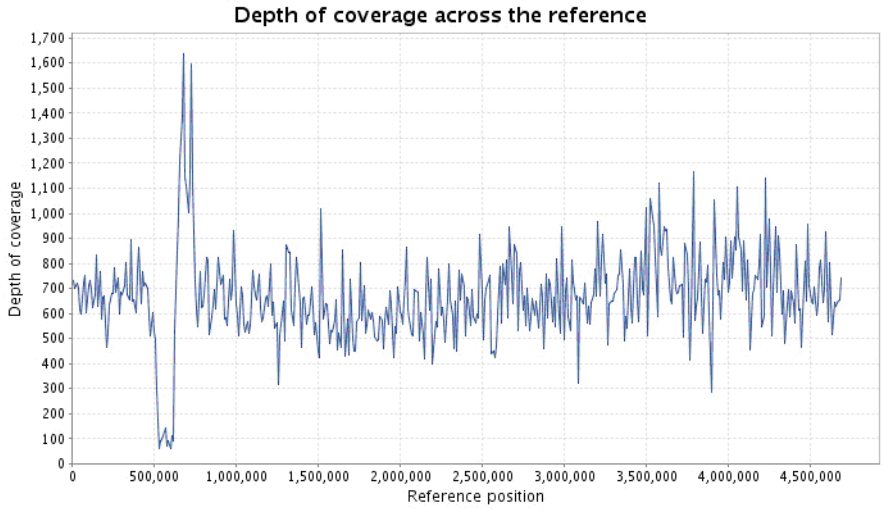

## qc_coverage

**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 adaptively binned to fit the reference into 10,000 evenly spaced bins. The coverage in each bin is determined by averaging across the bin. Following is a sample of this file:

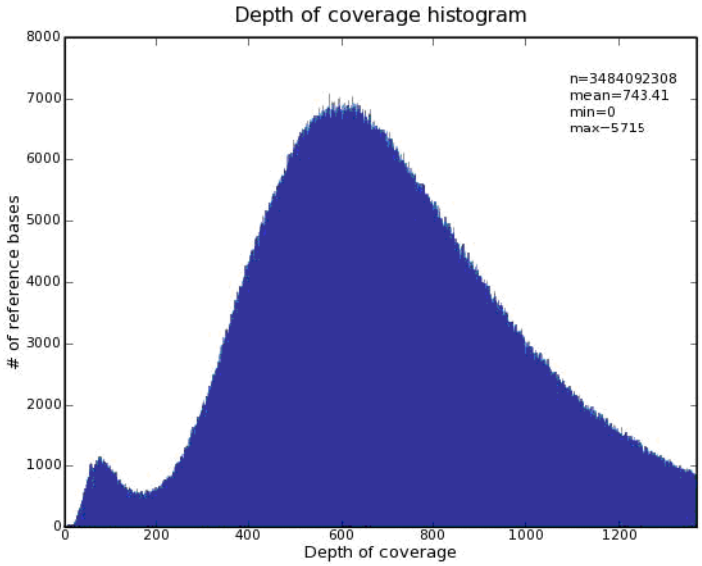
```
# 01/19/08 07:24:08
5000    361.71 368.27
15000   352.28 347.42
25000   361.45 360.79
35000   352.17 348.10
45000   305.62 306.73
```

**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. Following is a sample portion of this file:

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

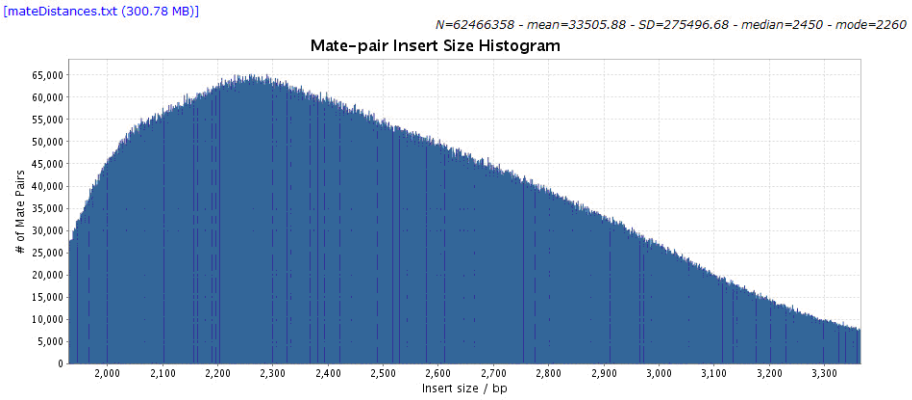
```
.
.
43.000000 796.000000
44.000000 846.000000
45.000000 866.000000
46.000000 904.000000
```




## 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
```

# Data Management A

## Saving data

This appendix describes data storage for the following:

- All data
- Raw image data
- Primary analysis results data
- Secondary analysis results data

### Approximate storage space for all data

Table 22 gives the size of all data for archival and storage consideration. Note that these are approximate, estimated file sizes.

**Table 22** 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 | 1.84 TB | 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 confirming that the image files are no longer needed, you can delete these image files to make room for additional sequencing runs.

### Primary analysis results data

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

**Table 23 Image analysis results files.**

| File type | File size | Comments |
|---|---|---|
| .spch | 476 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**(18**m*+17) bytes, where *n* is the number of beads, and *m* is the number of cycles (F3 and R3).<br>• For 50 cycles (1x50) and 220K beads, this yields 201 MB (per panel).<br>• A full slide has 2357 panels, which means that 476 GB are required for a full slide (50 bp single tag run). |

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

**Table 24 Color call results files.**

| File type | File size | Comments |
|---|---|---|
| .csfasta | 40 - 50 GB | Should be archived. |
| _QV.qual | 80 - 100 GB | Should be archived. |
| .stats | < 10 KB | Should be archived. |
| postPrimerSetPrimary.[JobID] | 260 - 300 GB | Temporary files. They are removed after completion. Located in $sampleJobFolder/. |

There are also color-space 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/`.

| File type | File size | Comments |
|---|---|---|
| .csfasta.reject | < 1 GB | Should be archived. |
| _QV.qual.reject | < 1 GB | Should be archived. |

The following files can be ignored. They can be archived if needed.

| File type | File size | Comments |
|---|---|---|
| Intensity files | 4 x 240 - 300 = 960 - 1200 GB | Optional. Not generated by default.<br>$sampleResultFolder/primary.[17digit timestamp]/reads/<br>or<br>$sampleResultFolder/primary.[17digit timestamp]/reject/ |

## Secondary analysis results data

The final mapped reads results in GFF Version 3 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

Data transfer from the SOLiD™ 3 Plus System 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 Plus System and the rest of your network. 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 the instrument and the rest of your network to ensure top transfer speeds. |
| No Gigabit LAN | Install a gigabit switch with the shortest path spanning tree. Connect all the SOLiD™ System outgoing connections to the switch so that data transfer between the SOLiD™ System and the rest of the network **only** occurs through the extra switch at 1 Gbps. |

# Exporting data

You export data from the SOLiD™ instrument using SOLiD™ Experiment Tracking System (SETS) software. You can set up SETS to auto-export data (csfasta and .spch files) for every run, or per run. You can also export files manually to a specified destination.

For detailed procedures, see Chapter 7 “Manage Administrative Tasks” in the *Applied Biosystems SOLiD™ SETS Software v3.5 Getting Started Guide* (PN 4444007).

# Software Warranty Information B

## APPLIED BIOSYSTEMS END USER SOFTWARE LICENSE AGREEMENT

FOR INSTRUMENT OPERATING AND ASSOCIATED BUNDLED SOFTWARE AND LIMITED PRODUCT WARRANTY

Applied Biosystems SOLiD™ Analysis Tools (SAT) v3.5 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

## P



## Q



## R



## S



## T



## W

Part Number 4443929 Rev. A 10/2009


applied biosystems™
part of life technologies™

**Applied Biosystems**
850 Lincoln Centre Drive | Foster City, CA 94404 USA
Phone 650.638.5800 | Toll Free 800.345.5224
www.appliedbiosystems.com

**Technical Resources and Support**
For the latest technical resources and support information for all locations, please refer to our Web site at
www.appliedbiosystems.com/support