

# AB diBayes SNP Tool Documentation

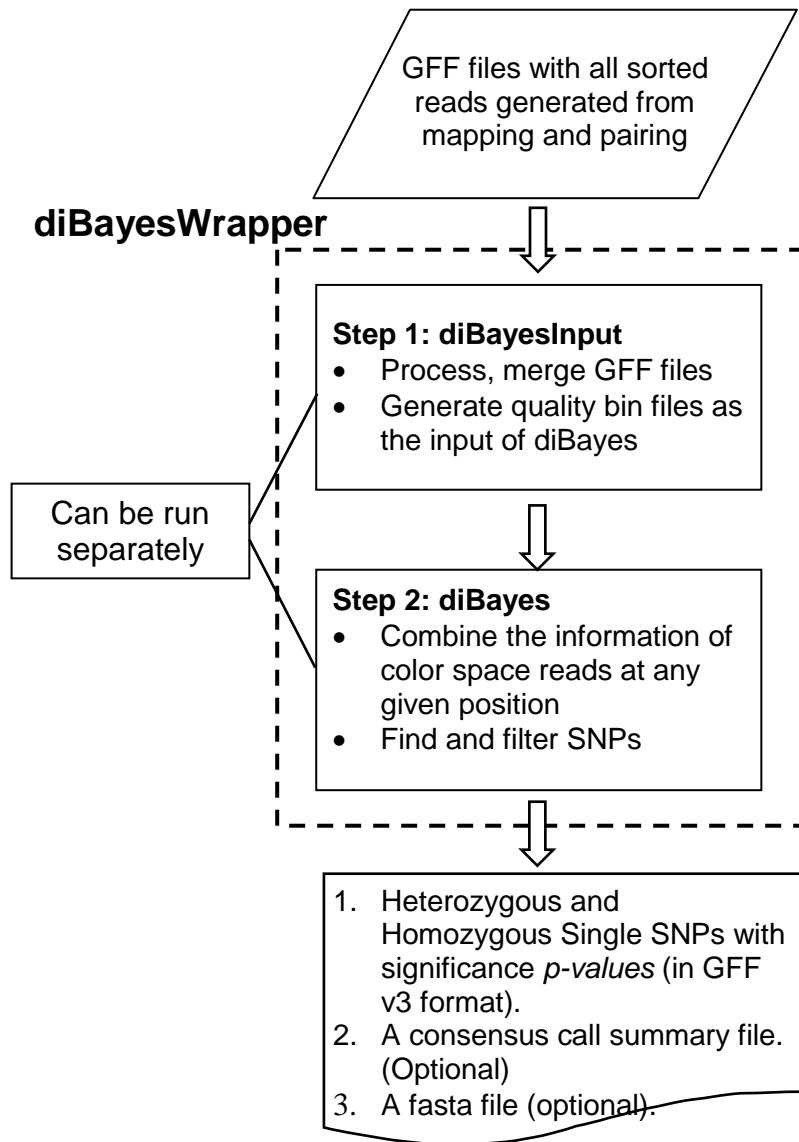
<b>1</b>	<b>INTRODUCTION .....</b>	<b>3</b>
<b>2</b>	<b>INSTALLATION .....</b>	<b>4</b>
2.1	Prerequisites .....	4
2.2	Installation Procedure.....	4
<b>3</b>	<b>DIBAYESWRAPPER PROGRAM.....</b>	<b>5</b>
3.1	Description .....	5
3.2	Algorithm.....	6
3.3	Usage .....	6
3.4	Configuration File Parameters .....	7
3.5	Other Scripts Called by this Program .....	9
3.6	System Input Files .....	9
3.6.1	Configuration file .....	9
3.6.2	SOLiD™ system reads in SOLiD GFF v2 format (files.GFF.prob.position.error parameter).....	9
3.6.3	Reference file (reference.file parameter) .....	9
3.6.4	F3 (R3) position error file (files.GFF.prob.position.error parameter) .....	10
3.6.5	F3 (R3) probe error file (files.GFF.prob.position.error parameter).....	10
3.7	Output Files.....	10
3.7.1	GFF file.....	10
3.7.2	Consensus calls .....	12
3.7.3	Base space fasta.....	14
<b>4</b>	<b>RUNNING DIBAYESINPUT AND DIBAYES.....</b>	<b>14</b>
4.1	diBayesInput .....	14
4.1.1	Usage .....	14
4.1.2	Parameters.....	14
4.2	diBayes .....	15
4.2.1	Usage .....	15
4.2.2	Parameters – Required .....	16
4.2.3	List of programs/Scripts Included.....	17
4.2.4	Other Scripts Called by this Program.....	17
4.2.5	Path Constraints.....	17
4.2.6	System Input Files.....	17

<b>5</b>	<b>FREQUENTLY ASKED QUESTIONS.....</b>	<b>18</b>
5.1	What is the work flow of the diBayes SNP detection algorithm? .....	18
5.2	How do I get the position and probe error files for diBayes?.....	18
5.3	How do I make minimum changes to a sample configuration file to run diBayes on my own data sets? .....	19
5.4	How do I know a run is successful?.....	19
5.5	How do I set the parameters for diBayes? How do I decide which call.stringency setting to use? .....	19
5.6	What is the normal running time for diBayes?.....	20
5.7	Why would I want to run the diBayesInput and diBayes steps separately? .....	20
5.8	How do I run the diBayesInput and diBayes step separately?.....	20
5.9	How do I parallelize the standalone diBayes?.....	21
5.10	What's difference between diBayes and corona SNP caller?.....	22
5.11	I seem to be finding too many SNPs: How do I troubleshoot?.....	22
5.12	I seem to be missing SNPs: How do I troubleshoot? .....	22

1.0	08/25/2009	
-----	------------	--

## 1 Introduction

The AB diBayes SNP package is the tool to identify SNPs from mapped and processed SOLiD™ System color space reads. It takes the color space reads and their quality values (in the form of GFF files from a single slide or from multiple slides) and the reference sequence and mismatch information on each SOLiD™ system slide as its input, and calls SNPs. It creates three final output files: A list of SNPs, a consensus fasta file with the same number of bases as the reference sequence (optional), and a list of all covered positions (optional).



**Figure 1: diBayes Package Flow**

The diBayes package includes three key components: **diBayesInput**, **diBayes**, and **diBayesWrapper** (See Figure 1):

- **diBayesInput** is the component that handles the input. `diBayesInput` takes the GFF file(s), merges them, and processes the reads to allow independent analysis of each genome position. It consolidates all reads per genome position and generates input files for `diBayes`. Chromosome-specific output directories are created in the user-specified output folder, based on the total number of chromosomes in all the GFF input file(s).
- **diBayes** is the program that calls SNPs. `diBayes` identifies SNPs by applying a Bayesian algorithm that evaluates the probability of the existence of a heterozygote or non-reference homozygote at any given position based on the color space reads at the given position, their quality values, and prior probabilities of being a position error or probe error. `diBayes` runs separately on each chromosome data. It creates a set of output files (SNP.GFF3, fasta file and `consensus_calls.txt`) for each individual chromosome (or contig).
- **diBayesWrapper** is the driver of the diBayes package. `diBayesWrapper` automatically runs the `diBayesInput` and `diBayes` commands based on input parameters in a configuration file. It consolidates individual results of each chromosome into final result files containing the information of all chromosomes.

Using `diBayesWrapper` and a centralized configuration file, `diBayesInput` and `diBayes` can be run automatically with a single command. These two key components can also be run separately when parallel processing multiple GFF files. If you run the components separately, you can modify some advanced parameters to fine-tune the algorithm.

## 2 Installation

### 2.1 Prerequisites

Before installation, verify that your system meets the following requirements:

- Linux CentOS 4 – The program has been built and tested on Linux CentOS 4 using GCC 3.4 and GNU Make 3.8.

### 2.2 Installation Procedure

To install the diBayes tool package:

1. Unzip the contents of the `diBayes_<version>.tar.gz` package into the desired location by entering the following:  

```
$ tar xzvf diBayes_<version>.tar.gz
```
2. Enter the subdirectory created and run make:  

```
$ cd diBayes_<version>  
$ make install PREFIX=/path/to/install
```
3. To test the installation run the `make test` command before installing the software. If the tool has run to completion, it will print “Completed diBayesAnalyzer successfully”.

- To install the diBayes binaries into an existing Corona Lite installation, run the `make install` command and set the PREFIX variable to the installation path:

```
$ make install PREFIX=$CORONAROOT
```

This will install the executables in the \$CORONAROOT/bin directory.

### 3 diBayesWrapper Program

**Program Name:** diBayesWrapper

**Program Version:** 1.1

**Development Languages:** C++

**Supports AB kit or protocol or sample prep method:** N/A

**Date:** June 15, 2009

#### 3.1 Description

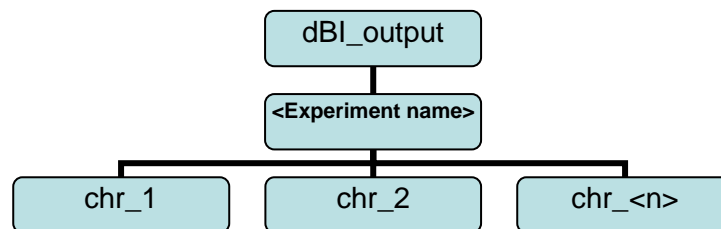
diBayesWrapper combines the two-steps, diBayesInput and diBayes, into one command.

First, diBayesWrapper reads the parameters from a centralized configuration file and executes the diBayesInput command to process the input of multiple GFF files.

Next, it runs the diBayes command to call SNPs using the results of the diBayesInput step and the parameters fetched from the configuration file.

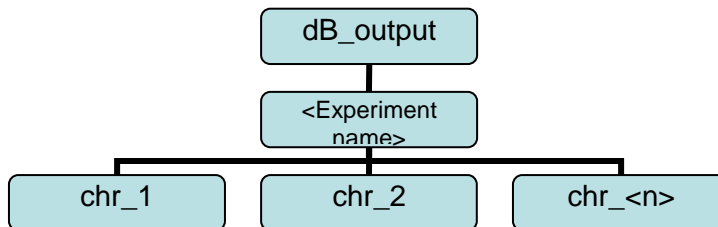
diBayesWrapper uses one centralized configuration file to set input, output files and folders and algorithm parameters. The configuration file lists all information required to run diBayesInput and diBayes. An example configuration file is available from the `config` subdirectory of the software package -diBayes\_<version>/config.

diBayesWrapper supports using multiple GFF files from multiple runs and/or multiple chromosomes. The GFF files are first processed by diBayesInput. Results of this step, including `fileranges.txt`, `positions.txt`, `quartiles.txt`, `maq` file (a temporary file), and `quality.bin` files, are generated and placed into chromosome-specific directories. Figure 2 shows the directory structure created in the user specified output folder (such as `dBI_output`) of the diBayesInput step:



**Figure 2. Directory Structure (diBayes\_intput output folder)**

Next, the `diBayes` command is repeatedly called to identify SNPs from each individual chromosome data created in the `diBayesInput` step. `diBayes` creates the set of output files (SNP report in GFF3 format, fasta file, and `consensus_calls.txt`) for each individual chromosome. The output of `diBayes` is kept in a similar directory structure as the output folder of `diBayes` input (See Figure 3.)



**Figure 3. Directory Structure (`diBayes` output folder)**

From these chromosome-specific directories, `diBayes` output files are merged/sorted in chromosome and genomic order and consolidated into final output files. These consolidated results files are placed in the `<Experiment name>` folder.

### 3.2 Algorithm

A Bayesian algorithm is used for SNP detection. It evaluates the probability that a heterozygote or non-reference homozygote exists at any given position based on the color space reads that cover the given position, their quality values, and prior probabilities of being a position error or probe error.

### 3.3 Usage

To run `diBayesWrapper`:

1. Copy the example `config.txt` file and modify the parameters for your analysis. See **Section 3.4 Configuration File Parameters** for details.

**Note:** To avoid overwriting previous results, make sure to modify `experiment.name` in the configuration file.

2. Ensure that the output and temporary directories (such as the folder for `diBayesInput.output`, `folder.for.diBayes.output`, and `working.dir`) that you specify in the configuration file actually exist.
3. Run `diBayesWrapper` by entering the following:

```
$ diBayesWrapper -c </path/to/configuration/file/config.txt>
```

As listed in Section 3.4, **all** Execution, Input, Output and Mandatory Algorithm parameters are required for the `diBayes` tool to run. `diBayes` provides four stringency settings: `low_coverage`, `med_coverage`, `default`, and `high_coverage`, with incremental SNP-calling stringency for different user needs and data sets (See Section 5.5). Each `call.stringency` setting controls multiple internal parameters. Some of these parameters are listed in the **Optional Algorithm Tuning Parameters** table.

We only recommended using these parameters when you have some special needs. Thus, these parameters are currently commented out in the `config.txt` file.

To use them, remove the comment symbol, “#”. in front of each parameter in the `config.txt` file. The values you provide will override the internal parameters set by `call.strengency` setting.

The running time for `diBayesWrapper` is approximately linear to the number of read at a given read length. For example, for 100 million 50mer mate-paired reads, it will take about 7 hours for the `diBayesInput` step to finish and another 9.5 hours for `diBayes` to complete on a single CPU machine. Advanced users may consider running these two steps separately on multiple processor clusters to achieve much shorter running time.

### 3.4 Configuration File Parameters

#### Execution Parameters:

Parameter	Description	Type/Range/Example
<code>diBayes.executable.path</code>	Full path to the <code>diBayes</code> executable. Equivalent to <code>PREFIX/bin</code> , which is usually set during the installation process.	Text Example: <code>/share/apps/diBayes/bin</code>
<code>diBayesInput.executable.path</code>	Full path to the <code>diBayesInput</code> executable. Equivalent to <code>PREFIX/bin</code> , which is usually set during the installation process.	Text Example: <code>/share/apps/diBayes/bin</code>
<code>run.diBayesInput</code>	Flag to run the <code>diBayesInput</code> step. Skip the step by setting this flag to 0 if the preparation step has already been run.	Boolean <ul style="list-style-type: none"> <li>• 0 = Skip <code>diBayesInput</code></li> <li>• 1 = Run <code>diBayesInput</code></li> </ul>
<code>run.diBayes</code>	Flag to run the <code>diBayes</code> step. Skip the step by setting this flag to 0 if only the preparation step is desired.	Boolean <ul style="list-style-type: none"> <li>• 0 = Skip <code>diBayes</code></li> <li>• 1 = Run <code>diBayes</code></li> </ul>

#### Input Parameters:

Parameter	Description	Type/Range/Example
<code>reference.file</code>	Full path to reference fasta file.	Text Example: <code>/share/reference/hgl8.fa</code>
<code>files.GFF.prob.position.error</code>	Input GFF file, position and probe error files including path.  The format is Serial No(the index of the GFF file), Mate pair flag,gff, F3 position error,F3 probe error, R3 position error,R3 probe error.  The example shows two input GFF files: The first one is from a mate-pair run with R3 error file settings, the second one from a fragment run without R3 error file settings.	Text Example: <code>1,1,/data/F3_alignment.v2.matepair.run1.gff,/data/F3_positionErrors.txt,/data/F3_123456_12encoded_errFreq.txt,/data/R3_positionErrors.txt,/data/R3_123456_12encoded_errFreq.txt</code> <code>2,0,/data/F3_alignment.v2.fragment.run2.gff,/data/F3_positionErrors.txt,/data/F3_123456_12encoded_errFreq.txt</code>

### Output Parameters:

Parameter	Description	Type/Range/Example
folder.for.diBayesInput.output	Full path to location of the directory where the output files of diBayesInput are written. This directory must already exist.	Text Example: /data/results/diBayesInput.outputdir
folder.for.diBayes.output	Full path to location of the directory where the output file of diBayes are written. This directory must already exist.	Text Example: /data/results/diBayes.outputdir
file.maq	Name of the output MAQ file for the diBayesInput step. Do not include the full path.	Text Example: MAQ
working.dir	Full path to of the temporary working directory in which some intermediate files for diBayes are placed. This file must be created in advance with write permission.	Text Example: /scratch
prefix.output.file	Name prefix for the output files.	Text Example: TestResults
experiment.name	Name of the experiment.	Text Example: Test_Results
verbose.flag	Flag to include warnings for each invalid line in a file	Boolean <ul style="list-style-type: none"> <li>• 0 = Do not print warning</li> <li>• 1 = Print warning</li> </ul>
flag.write.fasta	Flag to write fasta output file.	Boolean <ul style="list-style-type: none"> <li>• 0 = Do not write</li> <li>• 1 = Write</li> </ul>
flag.write.consensus	Flag to write consensus output file.	Boolean <ul style="list-style-type: none"> <li>• 0 = Do not write</li> <li>• 1 = Write</li> </ul>
flag.compress.consensus	Flag to compress consensus_calls.txt.	Boolean <ul style="list-style-type: none"> <li>• 0 = Do not compress</li> <li>• 1 = Compress</li> </ul>

### Algorithm Parameters:

Parameter	Description	Type/Range/Example
combine.left.right.qvs	Method used to combine left and right quality values into a single value. We recommend using 1.	Integer Values are: 1,2, 3, or 4
step.size	Step size used to break down the output into multiple files. Each file will contain up to step.size quality values, without breaking reads of the same probe.	Integer Example: 500000
maximal.read.length	Maximal read length. Note: diBayes allows combining reads from different sources with different read lengths.	Integer Example: 50
call.stringency	SNP call and filter stringency. In the order of increasing stringency: low_coverage < med_coverage < Default < high_coverage. The more stringent the setting, the less the false positives. At the same time,	Text Values are: <ul style="list-style-type: none"> <li>• Default</li> <li>• high_coverage</li> <li>• med_coverage</li> <li>• low_coverage</li> </ul>

	sensitivity may be compromised a little bit.	
poly.rate	Polymorphism rate: Expected frequency of heterozygotes in the population.	Float (< 1) Example: 0.003
flag.both.strands	Require that the novel allele be on both strands.	Boolean <ul style="list-style-type: none"> <li>• 0 = Do not require</li> <li>• 1 = Require</li> </ul>
min.coverage min.coverage.2nd.snp	Require at least this coverage to call a heterozygous SNP	Integer Example: 1
check.snp.evenly.distributed	Test whether both alleles are evenly distributed on both strands. A position will not be called heterozygote if alleles are distributed differently. This is a useful filter for high coverage data.	Boolean <ul style="list-style-type: none"> <li>• 0 = Do not check</li> <li>• 1 = Check</li> </ul>
half.reads.snps	Proportion of the reads containing either of the two candidate alleles. Filters positions with high raw error rate.	Float Example: 0.1
min.unique.start.pos	Minimum number of unique start positions required to call a heterozygote.	Integer Example: 1
min.allele.ratio	The less common allele must be at least this proportion of the reads of the two heterozygous alleles.	Float (0-1) Example: 0.1

### 3.5 Other Scripts Called by this Program

diBayesInput, diBayes.

### 3.6 System Input Files

#### 3.6.1 Configuration file

- All necessary inputs to diBayesWrapper and parameters that are required for running diBayesInput and diBayes are specified in a centralized configuration file. For more details, see **Section 3.4 Configuration File Parameters**.
- An example config.txt file is included in the source code package in diBayes\_<version>/config.

#### 3.6.2 SOLiD™ system reads in SOLiD GFF v2 format (files.GFF.prob.position.error parameter)

- The diBayesWrapper takes multiple GFF files that contain SOLiD™ system color space reads and corresponding quality values from multiple runs and/or chromosomes. These GFF files must be in SOLiD GFF v2 format. For details of the SOLiD GFF v2 format, see <http://solidsoftwaretools.com/gf/project/matoGFF/>.

#### 3.6.3 Reference file (reference.file parameter)

- The reference sequence is the sequence to which the reads were aligned and mapped, and should be in the fasta format. The reference sequence may have multiple chromosomes or contigs. Their sequences may contain IUB codes at positions of known SNPs. If it contains IUB codes,

heterozygote will be called with fewer reads providing evidence at these positions, since the prior probability of a heterozygote existing at this position is higher.

#### 3.6.4 F3 (R3) position error file (`files.GFF.prob.position.error` parameter)

- F3 (R3) Position error files are tab-delimited text files created by the SOLiD™ System software during secondary analysis. For fragment runs, only one F3 file is created and diBayes only requires an F3 position error file. Both F3 and R3 position error files are created for mate pair runs, and both are required for running diBayes. Different SOLiD™ system runs of the same sample may generate different F3 (R3) position error files for each run.

#### 3.6.5 F3 (R3) probe error file (`files.GFF.prob.position.error` parameter)

- F3 (R3) probe error files are tab-delimited text files created by the SOLiD™ System software during secondary analysis, reflecting 6-mer probe error. For fragment runs, only one F3 file is created and diBayes only requires an F3 probe error file. Both F3 and R3 probe error files are created for mate pair runs, and both are required for running diBayes. Different SOLiD™ system runs of the same sample may generate different F3 (R3) probe error files for each run.

### 3.7 Output Files

#### 3.7.1 GFF file

- The main diBayes output file is the list of all homozygous and heterozygous SNPs identified in the sample compared to the reference. This file is in GFF v3 format, a tab-delimited text format compatible with the UCSC Browser and other programs.
- For each identified SNP, the GFF v3 file reports its location, position, score, and attributes, such as genotype, coverage, dicolor encoding, and so on.
- **File name pattern:**  
<folder.for.diBayes.output>/<experiment.name>/<prefix.output.file>\_SNP.GFF3
- **Example:**

```
##GFF-version 3
##List of SNPs. Date Fri Jun 19 17:11:01 2009 Stringency: default
Mate Pair: 0 Read Length: 50 Polymorphism Rate: 0.003000 Bayes
Coverage: 60 Bayes_Single_SNP: 1 Filter_Single_SNP: 1
Quick_P_Threshold: 0.997000 Bayes_P_Threshold: 0.040000
Minimum_Allele_Ratio: 0.150000
Minimum_Allele_Ratio_Multiple_of_Dicolor_Error: 100
## Data source:
./outputs/DB_input_out/Test_Contigs/chr_1/firstTest_quality_50000
0 Experiment: Test_Contigs
#Chr Source Type Pos_Start Pos_End Score Strand Phase Attributes
```

```
chr1 SOLiD_diBayes SNP 420 420 0.000000
genotype=S;reference=C;coverage=52;refAlleleCounts=22;refAlleleStarts=15;refAlleleMeanQV=15;novelAlleleCounts=24;novelAlleleStarts=14;novelAlleleMeanQV=17;diColor1=03;diColor2=30;het=1;flag=
```

### GFF Output File Columns:

Column Name	Description	Example
##	Header comment lines.	Input files and algorithm parameters
#	Header of the results.	
seqid	The string ID of the sequence to which the start and end coordinates refer.	chr1
source	The source of the data. Always SOLiD_diBayes.	SOLiD_diBayes
type	Sequence ontology derived type for this variation. For diBayes, this is always SNP.	SNP
start	Start position of the SNP. Same as the end position.	420
end	End position of the SNP. Same as the start position.	420
score	Calculated p-value of the SNP.	0.000000
strand	Not used	.
phase	Not used	.
<b>Attributes:</b>		
genotype	Genotype in the form of IUB codes for bases observed in all the reads.	genotype=S
reference	The base of the reference sequence at the current position.	reference=C
coverage	The number of the reads that cover the current position.	coverage=52
refAlleleCounts	The number of reads of the reference allele at the current position.	refAlleleCounts=22
refAlleleStarts	The number of different start positions of reads having the reference allele at the current position.	refAlleleStarts=15
refAlleleMeanQV	The mean of quality values of all reference allele reads at the current position.	refAlleleMeanQV=15
novelAlleleCounts	The number of reads of the most abundant non-reference allele at the current position.	novelAlleleCounts=24
novelAlleleStarts	The number of different start positions of reads having the most abundant non-reference allele at the current position.	novelAlleleStarts=14
novelAlleleMeanQ	The mean of quality values of all novel allele reads.	novelAlleleMeanQV=17
diColor1	The most abundant allele in the reads (which is not necessarily the reference allele) in dicolor encoding, for example, 00, 01, ....., 32, 33 (16 possible dicolors).	diColor1=03
diColor2	The second most abundant allele in the reads in dicolor encoding, for example, 00, 01, ....., 32, 33 (16 possible dicolors).	diColor2=30
Het	Heterozygosity flag. Values are 0 for homozygous SNP, 1 for heterozygous SNP	het=1
flag	Filter summary flags for non-SNP positions. Always empty.	flag=

### 3.7.2 Consensus calls

- The `Consensus_Calls.txt` file lists information of all positions with coverage. The information includes the coverage of each position, the consensus call, the dicolor encoding, the p-value, the flags of failed filters, and more.
- **File name pattern:**  
`<folder.for.diBayes.output>/<experiment.name>/<prefix.output.file>_Consensus_Calls.txt`
- **Example:**

```
## Consensus_Calling_Alpha_2.txt. Date Fri Jun 19 17:11:01 2009
Stringency: default Mate Pair: 0 Read Length: 50 Polymorphism Rate:
0.003000 Bayes Coverage: 60 Bayes_Single_SNP: 1 Filter_Single_SNP: 1
Quick_P_Threshold: 0.997000 Bayes_P_Threshold: 0.040000
Minimum_Allele_Ratio: 0.150000
Minimum_Allele_Ratio_Multiple_of_Dicolor_Error: 100
## Data source:
./outputs/DB_input_out/Test_Contigs/chr_1/firstTest_quality_500000
Experiment: Test_Contigs
# Position Allele_DiColor1 Allele_DiColor2 Reference Genotype P-
value Flag Coverage nCounts_1st_allele nCounts_Reference_allele
nCounts_NonReference_allele Ref-Avg-QV Novel-Avg-QV Heterozygous
Algorithm Algorithm_Name
  442 03 03 C C 1.000000 m4 2 2 2 0 29 0 0 -1
  443 31 31 G G 1.000000 m4 2 2 2 0 29 0 0 -1
  444 12 12 T T 1.000000 m4 2 2 2 0 30 0 0 -1
  445 20 20 C C 1.000000 m4 2 2 2 0 28 0 0 -1
```

#### Consensus Call Output File Columns:

Column Name	Description	Example
Position	Location of the SNP on the reference sequence.	442
Allele_DiColor1	The most abundant allele in the reads (which is not necessarily the reference allele) in dicolor encoding. 00, 01, ....., 32, 33 (16 possible dicolors).	03
Allele_DiColor2	The second most abundant allele in the reads in dicolor encoding. 00, 01, ....., 32, 33 (16 possible dicolors).	03
Reference	The base of the reference sequence at the current position.	C
Genotype	Genotype in the form of IUB codes for bases observed in all the reads.	C
P-value	Calculated p-value of the SNP.	1.000000
Flag	Flag indicating why a location was <b>not</b> called a SNP. (See details in the next table).	m4
Coverage	The number of the reads that cover the current position.	2
nCounts_1st_allele	The number of the most abundant allele at the current position.	2
nCounts_Reference_allele	The number of reads having the reference allele at the current position.	2
nCounts_NonReference_allele	The number of reads having the most abundant non-reference allele at the current position.	0
Ref-Avg-QV	The mean of quality values of all reference allele reads at the current position.	29
Novel-Avg-QV	The mean of quality values of all novel allele reads.	0

Heterozygous	Heterozygosity flag. Values are: 0 for homozygous SNP, 1 for heterozygous SNP.	0
Algorithm	The algorithm used to call the current SNP. '-1': Not a SNP; bayes (Bayesian algorithm); quick (Frequentist algorithm).	-1 (Not a SNP)
Algorithm_Name	Not used.	

- Note:** If a position is **not** called as a SNP, a list of failed filter 'Flag' (the 7<sup>th</sup> column, flags are separated by commas.) are reported in the consensus\_calls.txt file. They specify the reasons why the current position is not called as a homozygous SNP or a heterozygous SNP. They may provide useful information for users to adjust different filters to achieve different sensitivities or specificities. These flags are described in the following table:

Flag	Filter Meaning
<b>Heterozygote</b>	
h1	Insufficient Coverage
h2	Not enough unique start positions
h3	Coverage is too high, filtered not a Het
h4	No evidence of a second allele
h5	Pattern of forward. Reverse reads irregular
h6	Invalid
h7	Novel allele is not on both strands
h8	Both alleles not evenly represented on both strands
h9	Second most common base not more frequent than third
h10	diColor is invalid
h11	invalid_sig1
h12	invalid_sig2
h13	Second allele has low quality value
h14	Second allele has low quality value, rare variant
h15	Genome position has low quality value
h16	Insufficient coverage of the reference allele
h17	Insufficient coverage of the non-reference allele
h18	Zero coverage
h19	Insufficient number of start positions of non-reference allele
h20	Insufficient coverage of non-reference allele
h21	Quality value of non-reference allele too low
<b>Homozygote</b>	
m1	Insufficient Coverage for a homozygous SNP
m2	Too many invalid discolors at this position
m3	Non-reference allele not on both strands
m4	Insufficient Coverage for a homozygous SNP
m5	Second most common allele too high frequency
m6	Insufficient Coverage for a homozygous call
m7	Dicolor inconsistent with reference
m8	Insufficient number of non-reference alleles
m9	Allele ratio of second allele too high for homozygous SNP
m10	Insufficient number of start positions of non-reference alleles
m11	No coverage

### 3.7.3 Base space fasta

- The Fasta file contains the new consensus sequence (SNPs included), which has exactly the same dimensions as the reference sequence.
- File name pattern:
- `<folder.for.diBayes.output>/<experiment.name>/<prefix.output.file>_Basespace2.fasta`

## 4 Running diBayesInput and diBayes

The individual executables, `diBayesInput` and `diBayes`, can be run independently to accomplish the same task as `diBayesWrapper`. This usage mode can be valuable when:

- The `diBayesInput` component takes a long time.
- Multiple `diBayes` executions are needed to evaluate different parameter settings.
- Large, multi-chromosome genomes must be analyzed in parallel using a queuing system, such as Sun Grid Engine.

`diBayesInput` must be run first to prepare files for analysis by `diBayes`.

### 4.1 diBayesInput

#### 4.1.1 Usage

- Usage:

```
diBayesInput <Method to combine qv value> <Verbose Flag>\
<Name of the output(maq) file>\
<Output directory>\
<Output name prefix>\
<Step size> <Maximum read length>\
<Input GFF file1> <Input GFF file2>
```

- Example:

```
$ diBayesInput 1 0 /data/dBI/MAQ /data/Dbi\
human_frag 500000 50\
/data/SOLiD/F3_R3_all_sorted.v2.GFF\
/data/SOLiD/B_F3_alignment.v2.GFF
```

- `diBayesInput` parameters must be presented in the exact order shown in the usage. `diBayesInput` can take multiple GFF files as its input. These GFF files should be separated by spaces in the command line. They can be generated from mate pair runs and/or fragment runs. In the example, the first GFF file listed is from the mate-pair run and the second is the fragment run. Therefore, when you set the `-v` and `-t` parameter values in the subsequent `diBayes` run, the command line for `diBayes` will contain both R3 and F3 files for the first GFF file, and no R3 error files for the second GFF file.

#### 4.1.2 Parameters

`diBayesInput` parameters are as follows:

**Note:** These parameters **must** be passed in the order listed.

- Method used to combine left and right quality values into a single value: Values are 1, 2, 3 or 4. This is a parameter used in the development stage. We recommend 1.
- Verbose Flag: Values are: 1 = Place warnings for each invalid line in a file; 0 = Do not. (We recommend 0.)
- Name of the output (maq) file, including the path.  
Example: /data/dBI/MAQ
- Directory to write the output files. (This directory must already exist).  
Example: /data/dBI
- Name to use as prefix for the output files.  
Example: human\_frag
- Step size used to break down the output into multiple files. Each file will contain up to Step size quality values, without breaking reads of the same probe.  
Example: 500000
- Maximum read length, such as 50. Note: This program allows combining reads from sources with different read lengths: use the longest read length.  
Example: 50
- Names of the input Gff files, including the paths. The Gff files must be sorted by genome positions.  
Example: /data/input/F3\_R3\_all\_sorted.v2.gff\  
/data/SOLiD/B\_F3\_alignment.v2.gff

## 4.2 diBayes

### 4.2.1 Usage

```
$ diBayes -f <reference file>\
-s <F3 position error file>[,<F3 position error file>]\
-u <F3 probe error file>[,F3 probe error file]\
-r <quartile file>\
-q <path to quality files>\
-F <first quality file index>\
-L <last quality file index>\
-l <longest read length>\
-o <output directory>\
-n <experiment name>\
-w <working directory>\
[-m <mate pair?>]\
[-t <R3 position error file>[,<R3 position error file>]]\
[-v <R3 probe error file>[,<R3 probe error file>]]\
[-p <polymorphism rate>]\
[-c <call stringency>]\
[-W <write fasta?>]\
[-C <write consensus?>]\
[-Z <zip fasta?>]
```

Example:

```

$ diBayes -f /data/references/DH10B_validated.fasta\
  -m 1\
  -s /data/SOLiD/F3_positionErrors.txt,\
  /data/SOLiD/B_F3_positionErrors_mod.txt\
  -u /data/SOLiD/F3_123456_12encoded_errFreq.txt,\
  /data/SOLiD/B_F3_123456_12encoded_errFreqmod.txt\
  -t /data/SOLiD/R3_positionErrors.txt,\
  /data/SOLiD/B_R3_positionErrors_mod.txt\
  -v /data/SOLiD/R3_123456_12encoded_errFreq.txt,\
  /data/SOLiD/B_R3_123456_12encoded_errFreqmod.txt\
  -q /data/dBI/FindSNP/chr_1/human_frag_quality_500000\
  -F 1\
  -L 1\
  -o /data/dBout/FindSNP/chr_1\
  -n FindSNP\
  -r /data/dBI/FindSNP/chr_1/human_frag_quartiles.txt\
  -l 50\
  -w /data/workingDirectory\
  -p 0.003\
  -c default

```

#### 4.2.2 Parameters - Required

**Note:** The argument order is **not** important for diBayes.

Commands	Description	Type/Range
-f, --reference-file	Full path to the reference fasta file.	Text Example: /share/reference/hg18 .fa
-s, --f3-position-error-file	Comma-separated full paths to the F3 position error files. If multiple files are specified, they must be in the same order as the GFF files from different runs passed to diBayesInput.	Text Example (multi-run case): F3_positionErrors_Run 1.txt F3_positionErrors_Run 2.txt  For a single run, only one file is required.
-u, --f3-probe-error-file	Comma-separated full paths to the F3 probe error files. This must be in the same order as the GFF files from different runs passed to diBayesInput.	Text Example (multi-run case): F3_12encoded_6merCont ext_Run1.txt,F3_12enc oded_6merContext_Run2 .txt
-q, --pos-qv-file-prefix	Full path to quality files. These files are created by the diBayesInput program.	Text Example: /data/diBayesInput Results
-F, --file-index-first	File number of the first quality file to analyze. Quality bin files are created in the diBayesInput step.	Integer Example: 1
-L, --file-index-last	File number of the last quality file to analyze. Quality bin files are created in the diBayesInput step.	Integer Example: 2
-o, --output-dir	Full path to the directory for the final output files. The directory must be	Text (valid directory)

	created in advance with write permission.	Example: /data/output/DB
-w, --working-dir	Full path to the directory for temporary working files. The directory must be created in advance with write permission.	Text (valid directory) Example: /data/workingDirectory
-n, --experiment-name	Experiment name. Output file names will have this prefix.	Text (valid path element) Example: Test_Output
-l, --read-length	Read length for the longest reads in the study.	Integer Example: 50
-r, --quartile-file-name	Full path to a file containing information about the coverage quartiles for this study. Quartile files are created by the diBayesInput program.	Text (valid directory) Example: /data/diBayesInputResults

### 4.2.3 List of programs/Scripts Included

diBayes

### 4.2.4 Other Scripts Called by this Program

None.

### 4.2.5 Path Constraints

Input arguments contain full path of expected input files.

### 4.2.6 System Input Files

#### 4.2.6.1 Reference sequence.

Fasta reference file as described in Section 3.6.3.

#### 4.2.6.2 F3 position error file.

Position error file as described in Section 3.6.4.

#### 4.2.6.3 F3 probe error file.

Probe error file as described in Section 3.6.5.

#### 4.2.6.4 Quality files.

Original GFF files in GFF v2 format are merged, converted and split into binary files with a certain size by diBayes\_input.

#### 4.2.6.5 Quartile file.

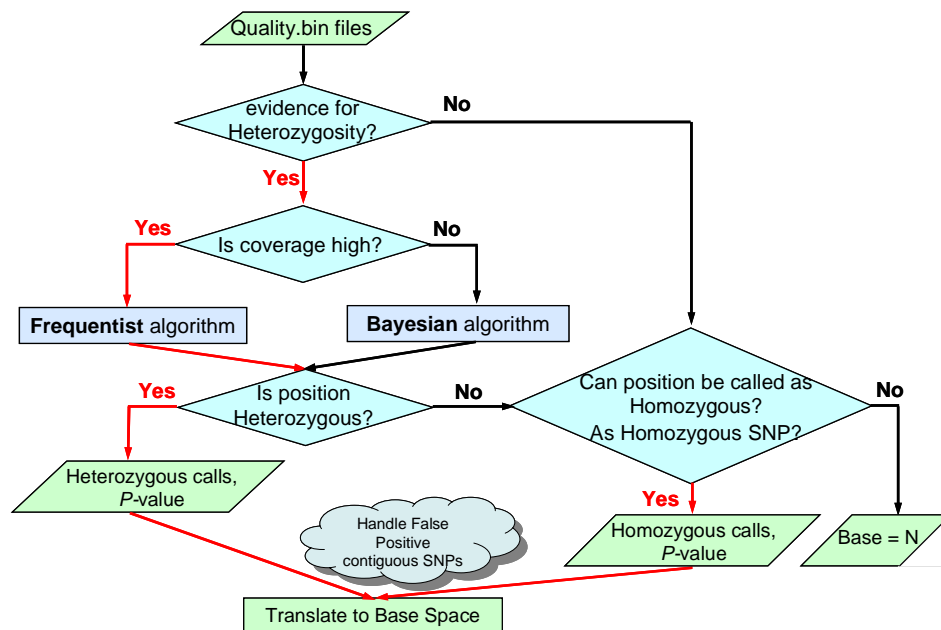
Created by diBayes\_input, contains information about the median coverage and the inter-quartile range of the coverage.

#### 4.2.6.6 Output files

Output files as described in Section 3.7.

## 5 Frequently Asked Questions

### 5.1 What is the work flow of the diBayes SNP detection algorithm?



**Figure 4. The work flow of the diBayes algorithm.**

**Answer:**

The work flow of the diBayes algorithm is shown in Figure 4.

### 5.2 How do I get the position and probe error files for diBayes?

**Answer:**

The position and probe error files are required input files for diBayes. You can run test examples using the position and probe error files provided. For a full analysis it is better to generate position and probe error files specific to each slide. To generate these error files, do the following

- Install the **corona v3** package on an offline cluster.
- Use **GFF** files (in **v2** format) from matching results or mate pairing results as input.
- Enter the following command to generate a position error file with a V2 GFF file:

```

$/share/apps/corona/bin/jmoap_module.sh CountPositionErrors -t F3 \
F3_alignment.v2.gff > error_by_positions_file
  
```

- Enter the following command to generate a 6mer context file (Probe Error) V2 GFF file:

```

$/share/apps/corona/bin/jmoap_module.sh CountErrorsByPrefix -t F3
-d -1 -k 6 --mask1=01111 --maskn=11111 \
--tmfile=/share/apps/corona/etc/analysis/6mers.Tm F3_alignment.v2.gff >
F3_123456_12encoded_errFreq.txt
  
```

### 5.3 How do I make minimum changes to a sample configuration file to run diBayes on my own data sets?

**Answer:**

You need to answer the following four questions and change the settings listed below in the configuration file:

1. Where was diBayes installed?

Set `dibayes.executable.path` and `diBayesInput.executable.path`.

2. Where will the temporary and output folders be?

Set `folder.for.dibayesinput.output`, `folder.for.dibayes.output`, and `working.dir`.

**Note:** All folders must already exist!

3. Where are the inputs for diBayes?

Set `reference.file` and `files.gff.prob.position.error`.

**Example:**

```
files.gff.prob.position.error=1,0,./in/frag1.gff,./in/F3_pstErr.r1.txt,./in/F3_prbErr.r1.txt
files.gff.prob.position.error=2,1,./in/mate2.gff,./in/F3_pstErr.r1.txt,./in/F3_prbErr.r1.txt,\
./in/R3_pstErr.r1.txt,./in/R3_prbErr.r1.txt
```

4. Did you change “experiment.name”?

Change `experiment.name` to avoid accidental replacement of the previous results.

### 5.4 How do I know a run is successful?

**Answer:**

If the run is successful, you will see `[Status][diBayesAnalyzer] Completed diBayesAnalyzer successfully` on the screen.

### 5.5 How do I set the parameters for diBayes? How do I decide which call.stringency setting to use?

**Answer:**

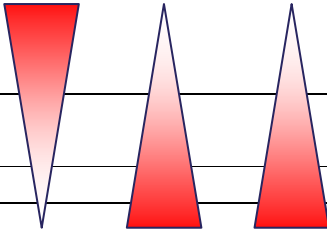
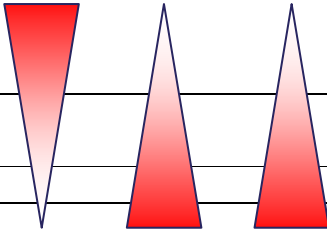
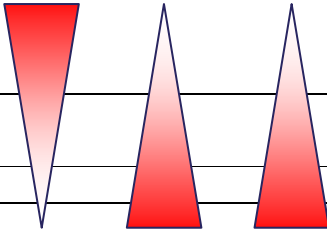
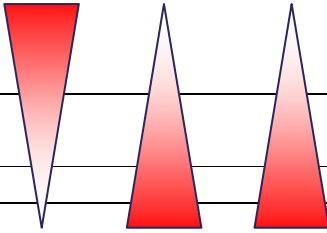
diBayes provide 4 stringency settings: **low-coverage**, **med-coverage**, **default**, and **high-coverage**, with incremental stringency for different user needs and data sets (See Figure 5).

**high-coverage** has the highest stringency, which is recommended for use on data sets with >80x coverage (very approximately), or where very low false positive tolerance is allowed.

**default** is less stringent than “high-coverage”. We recommend using this setting for data with 20-80x coverage, and also with lower coverage sets when very low false positives tolerance is allowed.

**med-coverage** usually works nicely for data sets with 1x-25x. It can also be used on data sets with higher coverage if users want to have more SNPs but with more false positives. The main difference between “default” and “med-coverage” is that it doesn't require coverage of the alleles on both strands. Med-coverage is also a good

setting to use if you expect differences in coverage of the two alleles on both strands of DNA, for example transcriptome data or certain kinds of DNA enrichment data. **low-coverage** is the setting with the least stringency. Use it only if you want very aggressive SNP detection. Using this setting, a SNP can be called even if only 1 observation of the non-reference allele is seen. As a consequence, it will have a higher false positive rate.

call.stringency	stringency	SNPs	false positive	Recommended coverage	Comments
<i>high-coverage</i>				>80x	Recommend when very low false positive tolerance is allowed
<i>default</i>				20x ~ 80x	Requires the allele on both strands
<i>med-coverage</i>				1x ~ 25x	No both-strand requirement
<i>low-coverage</i>					Very aggressive

**Figure 5. Four empirical call.stringency settings for diBayes**

diBayes also has a number of parameters that can be individually adjusted. For example, you can set the minimum coverage required to detect a SNP: whether the alleles are required to be equally represented on both strands of DNA, and many other parameters. See the rest of this document for more details. Depending on the sample type these can be useful to set individually.

### 5.6 What is the normal running time for diBayes?

**Answer:**

The running time is linear to the number of reads at a given read length. For example, for **100 million 50mer mate-paired reads**, it will take about **7 hours for the diBayesInput step** to finish and another **9.5 hours for diBayes** to complete on a single CPU machine. Parallelization of diBayes on multiple CPU clusters step greatly reduces the running time.

### 5.7 Why would I want to run the diBayesInput and diBayes steps separately?

**Answer:**

If you want to try different diBayes parameters, they do not need to rerun the diBayesInput step every time. Furthermore, you may want to parallelize the computational work to achieve more efficient running time.

### 5.8 How do I run the diBayesInput and diBayes step separately?

**Answer:**

Command line examples for each step can be found in all test case folders (For example, `cmd_diBayesInput_testcase000x`, `cmd_diBayes_testcase000x`).

There is one tricky situation that may need some extra attention, in which the GFF files are from a mixture of mate-pair runs and fragment runs. In this case, first, `-m` should be set to '1'. The R3 error files need to be specified for all runs, including the

fragment runs. To do this, use the fragment F3 position and probe error files as proxies for R3 error files (the red fonts in the following example) in `-t` and `-v`.  
 Example: diBayesInput step

```

$../../../../bin/diBayesInput 1 0 outputs/DB_input_out/MAQ
outputs/DB_input_out test_case 500000 50 inputData/F3R3_align.r1.gff
inputData/F3_align.r2.gff inputData/F3R3_align.r3.gff
diBayes step,
$../../../../bin/diBayes -m 1 -f reference/ref.fasta \ -s
inputData/F3_pstErrs.r1.txt,inputData/F3_pstErr.r2.txt,inputData/F3_pst
Err.run3.txt \
                                -u inputData/F3_prb.r1.txt,
inputData/F3_prb.r2.txt,inputData/F3_prb.r3.txt \ -t
inputData/R3_pstErrs.r1.txt,inputData/F3_pstErr.r2.txt,inputData/R3_pst
Err.run3.txt \ -v inputData/R3_prb.r1.txt,
inputData/F3_prb.r2.txt,inputData/R3_prb.r3.txt \ -q
outputs/DB_input_out/chr_1/test_case_quality_500000 -F 1 -L 1 -l 50 -o
outputs/DB_out \ -n testcase0003 -r
outputs/DB_input_out/chr_1/test_case_quartiles.txt -w
outputs/DB_working -p 0.003 \
-c default
  
```

### 5.9 How do I parallelize the standalone diBayes?

Answer:

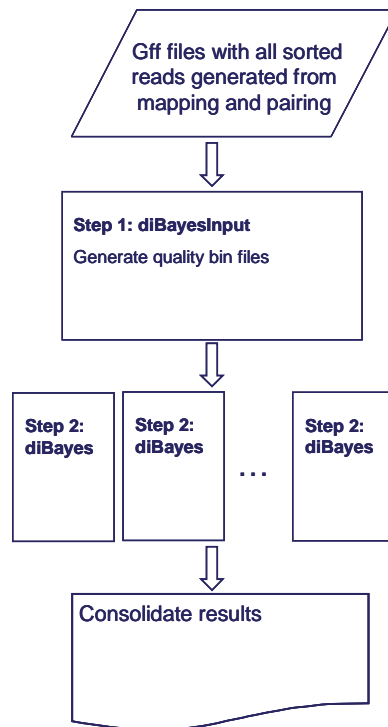


Figure 6. Parallelization Strategy for diBayes

As shown in Figure 6, we recommend **not** parallelizing the diBayesInput step but parallelizing the diBayes step for different chromosomes and/or different bin files. When setting up the parallelization jobs, users should create different output folders

for each individual job to avoid the results of different jobs being overwritten by each other.

### **5.10 What's difference between diBayes and corona SNP caller?**

**Answer:**

Normally there is a high percentage overlap between results of diBayes and those of corona SNP caller (CC). For example, on whole genome human, the overlap of SNPs detected is >95%. diBayes has a more formal statistical approach to SNP detection, which incorporates a Bayesian algorithm and a series of filters, and makes use of additional information including the color quality values for SNP detection. diBayes has been shown to have a higher sensitivity and specificity (fewer false positives and fewer false negatives) on a number of data sets.

### **5.11 I seem to be finding too many SNPs: How do I troubleshoot?**

**Answer:**

Look at the properties of the SNPs and compare them to the properties of all the positions (in the `Consensus_Calls.txt` file). Is the coverage of these SNPs much lower or much higher than average? Is the color quality value of the non-reference allele much lower than average? You may want to post-filter the results if you find these kinds of patterns. For example, you may want to remove SNPs with very low coverage or very low color quality values or p-values close to 1. You may want to repeat the analysis with a more stringent setting (such as `high-coverage`).

### **5.12 I seem to be missing SNPs: How do I troubleshoot?**

**Answer:**

First, try repeating the analysis with a lower stringency level (such as `med-coverage`). Look at the positions that you expect to be SNPs in the `Consensus_Calls.txt` file. There should usually be a list of flags that describe the reasons the position was not called as a SNP (see the documentation for details of what the flags mean). In addition, look at the properties of this position.

Visualization of the reads may be helpful here, for example with the SOLiD Alignment Browser. Is the coverage much higher than average? There is a filter to remove heterozygous SNPs at positions of extremely high coverage, since these have previously been observed to be usually variants in repeat regions rather than truly heterozygous SNPs at a single position. Are the reads strongly biased towards one strand, or is the non-reference allele missing from one strand? You probably want to run at `med-coverage`, or switch off the both strands requirement. Do all the reads have the same start position, because of the way the sample was prepared? You need to reduce the number of unique start positions required to call a SNP.

### **Licensing**

This software is being licensed to you under the OSI compliant GNU Public License (GPL V3). The license can be found at the following URL: <http://www.gnu.org/licenses/gpl.html>. Please read the license in its entirety and ensure that you understand the licensing conditions for use. Your use of this software indicates your acceptance of this licensing agreement.

© 2009 Life Technologies Corporation. All rights reserved.  
The trademarks mentioned herein are the property of Life Technologies Corporation or their respective owners.