# snppipeline Documentation

***Release 2.2.1***

**Errol Strain, Yan Luo, James Pettengill, Hugh A. Rand, Steve Davi**

**Sep 23, 2020**

# Contents


**1 CFSAN SNP Pipeline** **3**
- 1.1 Introduction . . . 3
- 1.2 Citing SNP Pipeline . . . 4
- 1.3 License . . . 4

**2 SNP Pipeline Processes** **5**

**3 Installation** **7**
- 3.1 Step 1 - Operating System Requirements . . . 7
- 3.2 Step 2 - Executable Software Dependencies . . . 7
- 3.3 Step 3 - Environment Variables . . . 8
- 3.4 Step 4 - Python . . . 8
- 3.5 Step 5 - Pip . . . 8
- 3.6 Step 6 - Install the SNP Pipeline Python Package . . . 8
- 3.7 Upgrading SNP Pipeline . . . 9
- 3.8 Uninstalling SNP Pipeline . . . 9
- 3.9 Tips . . . 9

**4 Usage** **11**
- 4.1 Inputs . . . 12
- 4.2 Outputs . . . 12
- 4.3 All-In-One SNP Pipeline Script . . . 13
- 4.4 Logging . . . 13
- 4.5 Mirrored Inputs . . . 14
- 4.6 High Performance Computing . . . 15
- 4.7 Tool Selection . . . 16
- 4.8 All-In-One SNP Pipeline Workflows . . . 16
- 4.9 Step-by-Step Workflows . . . 21
- 4.10 Duplicate Read Removal . . . 29
- 4.11 Local Realignment . . . 30
- 4.12 SNP Filtering . . . 30
- 4.13 Excessive SNPs . . . 32
- 4.14 Metrics . . . 32
- 4.15 Error Handling . . . 34

**5 Correct and Reproducible Results** **35**
- 5.1 Reproducible Results . . . 35

5.2 Test Data Sets . . . 37

**6 FAQ / Troubleshooting Guide 39**
6.1 Installation . . . 39
6.2 Running the Pipeline . . . 40
6.3 Performance . . . 41
6.4 Developer Questions . . . 42

**7 Configuration 45**
7.1 StopOnSampleError . . . 45
7.2 MaxCpuCores . . . 46
7.3 CpuCoresPerProcessOnHPC . . . 46
7.4 CpuCoresPerProcessOnWorkstation . . . 46
7.5 MaxSnps . . . 47
7.6 SnpPipeline_Aligner . . . 47
7.7 Bowtie2Build_ExtraParams . . . 47
7.8 SmaltIndex_ExtraParams . . . 47
7.9 SamtoolsFaidx_ExtraParams . . . 48
7.10 CreateSequenceDictionary_ExtraParams . . . 48
7.11 Bowtie2Align_ExtraParams . . . 48
7.12 SmaltAlign_ExtraParams . . . 49
7.13 SamtoolsSamFilter_ExtraParams . . . 49
7.14 SamtoolsSort_ExtraParams . . . 50
7.15 SamtoolsIndex_ExtraParams . . . 50
7.16 RemoveDuplicateReads . . . 51
7.17 PicardJvm_ExtraParams . . . 51
7.18 PicardMarkDuplicates_ExtraParams . . . 51
7.19 EnableLocalRealignment . . . 51
7.20 GatkJvm_ExtraParams . . . 52
7.21 RealignerTargetCreator_ExtraParams . . . 52
7.22 IndelRealigner_ExtraParams . . . 52
7.23 SamtoolsMpileup_ExtraParams . . . 53
7.24 VarscanMpileup2snp_ExtraParams . . . 53
7.25 VarscanJvm_ExtraParams . . . 53
7.26 FilterRegions_ExtraParams . . . 54
7.27 MergeSites_ExtraParams . . . 54
7.28 CallConsensus_ExtraParams . . . 54
7.29 SnpMatrix_ExtraParams . . . 55
7.30 SnpReference_ExtraParams . . . 55
7.31 MergeVcfs_ExtraParams . . . 55
7.32 BcftoolsMerge_ExtraParams . . . 56
7.33 CollectSampleMetrics_ExtraParams . . . 56
7.34 CombineSampleMetrics_ExtraParams . . . 57
7.35 Torque_StripJobArraySuffix . . . 57
7.36 GridEngine_StripJobArraySuffix . . . 57
7.37 GridEngine_PEname . . . 57
7.38 Slurm_SbatchExtraParams . . . 58
7.39 GridEngine_QsubExtraParams . . . 58
7.40 Torque_QsubExtraParams . . . 58

**8 Command Reference 59**
8.1 cfsan_snp_pipeline . . . 59
8.2 run . . . 60
8.3 data . . . 61

8.4 index_ref . . . 62
8.5 map_reads . . . 63
8.6 call_sites . . . 63
8.7 filter_regions . . . 63
8.8 merge_sites . . . 64
8.9 call_consensus . . . 65
8.10 merge_vcfs . . . 67
8.11 snp_matrix . . . 67
8.12 distance . . . 68
8.13 snp_reference . . . 68
8.14 collect_metrics . . . 69
8.15 combine_metrics . . . 70
8.16 purge . . . 70

**9 Contributing** **71**
9.1 Types of Contributions . . . 71
9.2 Get Started! . . . 72
9.3 Pull Request Guidelines . . . 73
9.4 Tips . . . 73

**10 Credits** **75**
10.1 CFSAN BioInformatics Team . . . 75
10.2 External Contributors . . . 75

**11 History** **77**
11.1 2.2.1 (2020-09-23) - docs . . . 77
11.2 2.2.0 (2020-05-19) - docs . . . 77
11.3 2.1.1 (2019-09-09) - docs . . . 77
11.4 2.1.0 (2019-07-01) - docs . . . 77
11.5 2.0.2 (2018-09-20) - docs . . . 78
11.6 2.0.1 (2018-09-20) - docs . . . 78
11.7 2.0.0 (2018-07-20) - docs . . . 78
11.8 1.0.1 (2017-09-28) - docs . . . 79
11.9 1.0.0 (2017-09-01) - docs . . . 79
11.10 0.8.2 (2017-06-09) - docs . . . 80
11.11 0.8.1 (2017-06-02) - docs . . . 80
11.12 0.8.0 (2017-05-09) - docs . . . 80
11.13 0.7.0 (2016-11-30) - docs . . . 81
11.14 0.6.1 (2016-05-23) . . . 81
11.15 0.6.0 (2016-04-11) . . . 82
11.16 0.5.2 (2016-03-07) . . . 82
11.17 0.5.1 (2016-02-19) . . . 82
11.18 0.5.0 (2016-01-19) . . . 83
11.19 0.4.1 (2015-10-30) . . . 83
11.20 0.4.0 (2015-10-22) . . . 84
11.21 0.3.4 (2015-06-25) . . . 84
11.22 0.3.3 (2015-04-14) . . . 85
11.23 0.3.2 (2015-01-14) . . . 86
11.24 0.3.1 (2014-10-27) . . . 86
11.25 0.3.0 (2014-10-22) . . . 86
11.26 0.2.1 (2014-09-24) . . . 87
11.27 0.2.0 (2014-09-17) . . . 87
11.28 0.1.1 (2014-07-28) . . . 88
11.29 0.1.0 (2014-07-03) . . . 89

12 Indices and tables 91

Contents:

CHAPTER 1

# CFSAN SNP Pipeline

The CFSAN SNP Pipeline is a Python-based system for the production of SNP matrices from sequence data used in the phylogenetic analysis of pathogenic organisms sequenced from samples of interest to food safety.

The SNP Pipeline was developed by the United States Food and Drug Administration, Center for Food Safety and Applied Nutrition.

- Free software: See license below.
- Documentation: http://snp-pipeline.readthedocs.io/en/latest/readme.html
- Source Code: https://github.com/CFSAN-Biostatistics/snp-pipeline
- PyPI Distribution: https://pypi.python.org/pypi/snp-pipeline

## 1.1 Introduction

The CFSAN SNP Pipeline uses reference-based alignments to create a matrix of SNPs for a given set of samples. The process generally starts off by finding a reference that is appropriate for the samples of interest, and collecting the sample sequence data into an appropriate directory structure. The SNP pipeline can then be used to perform the alignment of the samples to the reference. Once the sample sequences are aligned, a list of SNP positions is generated. The list of SNP positions is then used in combination with alignments of the samples to the reference sequence to call SNPs. The SNP calls are organized into a matrix containing (only) the SNP calls for all of the sequences.

This software was developed with the objective of creating high quality SNP matrices for sequences from closely-related pathogens, e.g., different samples of Salmonella enteriditis from an outbreak investigation. The focus on closely related sequences means that this code is not suited for the analysis of relatively distantly related organisms, where there is not a single reference sequence appropriate for all the organisms for which an analysis is desired.

The CFSAN SNP Pipeline is written in Python with some embedded bash snippets. The code is designed to be straightforward to install and run from the command line. A configuration file supports customizing the behavior of the pipeline. In situations where additional customization is desired, the code is not highly complex and should be easy to modify as necessary.

Examples of using the code are provided. These examples serve as both unit tests, and as examples that can be modified to work on other data sets of interest.

## 1.2 Citing SNP Pipeline

Please cite the publication below:

> Davis S, Pettengill JB, Luo Y, Payne J, Shpuntoff A, Rand H, Strain E. (2015) CFSAN SNP Pipeline: an automated method for constructing SNP matrices from next-generation sequence data. PeerJ Computer Science 1:e20 https://doi.org/10.7717/peerj-cs.20

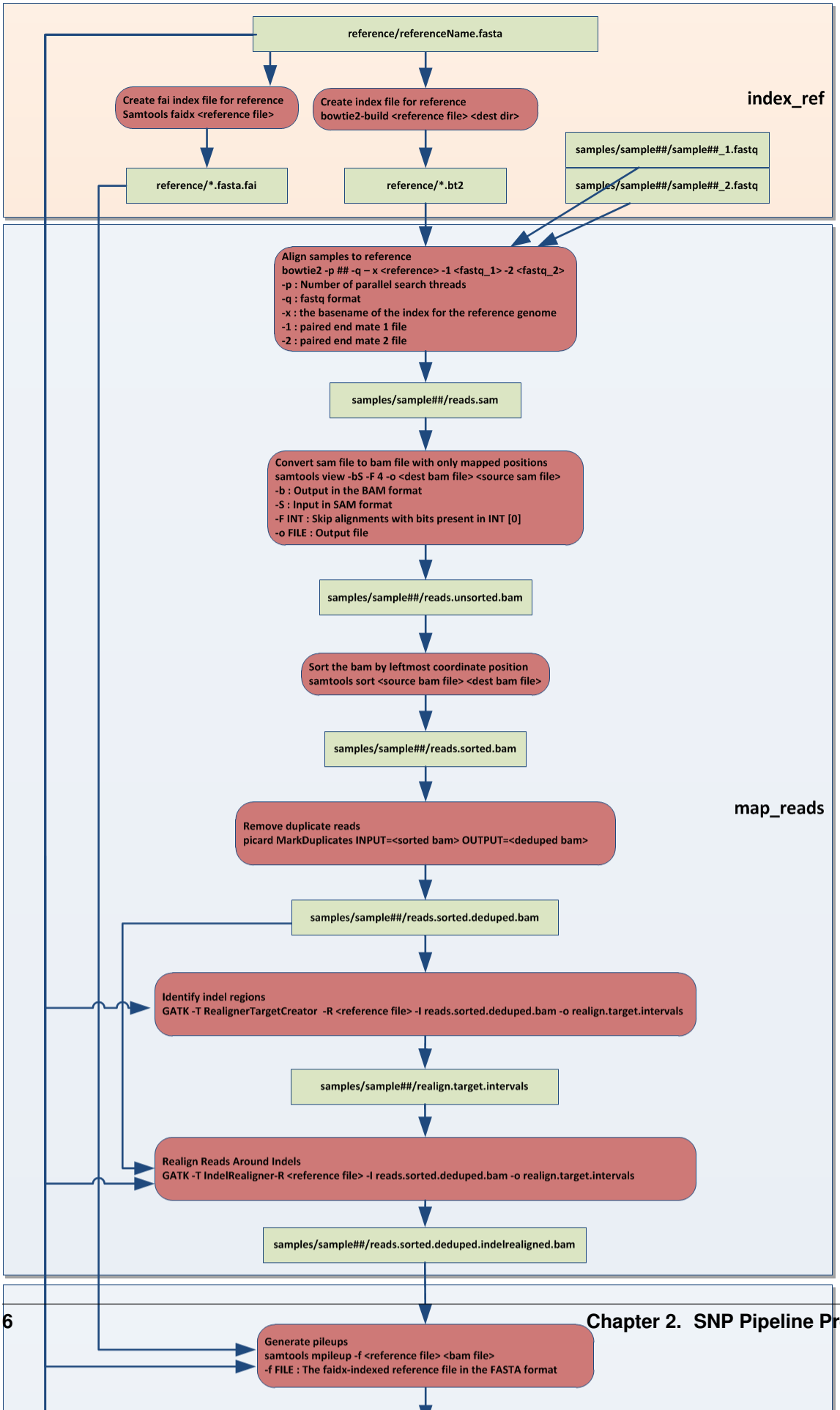
## 1.3 License

See the LICENSE.txt file included in the SNP Pipeline distribution.

CHAPTER 2

# SNP Pipeline Processes

The drawing below depicts the processes, files, and data flows within the SNP Pipeline.


reference/referenceName.fasta
Create fai index file for reference
Samtools faidx <reference file>
Create index file for reference
bowtie2-build <reference file> <dest dir>
index_ref
samples/sample##/sample##_1.fastq
samples/sample##/sample##_2.fastq
reference/*.fasta.fai
reference/*.bt2
Align samples to reference
bowtie2 -p ## -q – x <reference> -1 <fastq_1> -2 <fastq_2>
-p : Number of parallel search threads
-q : fastq format
-x : the basename of the index for the reference genome
-1 : paired end mate 1 file
-2 : paired end mate 2 file
samples/sample##/reads.sam
Convert sam file to bam file with only mapped positions
samtools view -bS -F 4 -o <dest bam file> <source sam file>
-b : Output in the BAM format
-S : Input in SAM format
-F INT : Skip alignments with bits present in INT [0]
-o FILE : Output file
samples/sample##/reads.unsorted.bam
Sort the bam by leftmost coordinate position
samtools sort <source bam file> <dest bam file>
samples/sample##/reads.sorted.bam
Remove duplicate reads
picard MarkDuplicates INPUT=<sorted bam> OUTPUT=<deduped bam>
map_reads
samples/sample##/reads.sorted.deduped.bam
Identify indel regions
GATK -T RealignerTargetCreator -R <reference file> -I reads.sorted.deduped.bam -o realign.target.intervals
samples/sample##/realign.target.intervals
Realign Reads Around Indels
GATK -T IndelRealigner-R <reference file> -I reads.sorted.deduped.bam -o realign.target.intervals
samples/sample##/reads.sorted.deduped.indelrealigned.bam
Generate pileups
samtools mpileup -f <reference file> <bam file>
-f FILE : The faidx-indexed reference file in the FASTA format

CHAPTER 3

# Installation

The SNP Pipeline software package consists of python scripts with dependencies on executable programs launched by the scripts.

## 3.1 Step 1 - Operating System Requirements

The SNP Pipeline runs in a Linux environment. It has been tested on the following platforms:

- Red Hat
- CentOS
- Ubuntu

## 3.2 Step 2 - Executable Software Dependencies

You should have the following software installed before using the SNP Pipeline.

| Software | Tested Version | Description |
|---|---|---|
| Bowtie2 | 2.3.4.1 | A tool for aligning reads to long reference sequences |
| SMALT | 0.7.6 | A tool for aligning reads to long reference sequences |
| SAMtools | 1.8 | Utilities for manipulating alignments in the SAM format |
| Picard | 2.18.4 | A set of tools for manipulating sequence data |
| GATK | 3.8-1-0 | Variant discovery and genotyping tools |
| VarScan | 2.3.9 | A tool to detect variants in NGS data |
| tabix | 1.8 | A generic indexer for tab-delimited genome position files |
| bgzip | 1.8 | Part of the tabix package, bgzip is a block compression utility |
| BcfTools | 1.8 | Utilities for variant calling and manipulating VCFs and BCFs |

Note: the versions above are tested and known to work together. Other versions may also work.

Note: you will need either Bowtie2 or SMALT. You do not have to install both. However, the included result files were generated with Bowtie2. Your results may differ when using SMALT.

Note: Picard is required when removing deplicate reads and when realigning reads around indels. Both of these functions are enabled by default, but can be disabled in the configuration file.

Note: GATK is required when realigning reads around indels, which is enabled by default, but can be disabled in the configuration file.

## 3.3 Step 3 - Environment Variables

Define the CLASSPATH environment variable to specify the location of the Picard, VarScan, and GATK jar files. Add the following lines (or something similar) to your .bashrc file:

```
export CLASSPATH=~/software/varscan.v2.3.9/VarScan.jar:$CLASSPATH
export CLASSPATH=~/software/picard/picard.jar:$CLASSPATH
export CLASSPATH=~/software/GenomeAnalysisTK-3.8-1-0-gf15c1c3ef/GenomeAnalysisTK.jar:
↪$CLASSPATH
```

## 3.4 Step 4 - Python

The SNP pipeline is compatible with python version 2.7, 3.4, 3.5, 3.6 and 3.7. The pipeline has not been tested on other python versions. You can either build from source or install a precompiled version with your Linux package manager.

## 3.5 Step 5 - Pip

This can be a troublesome installation step – proceed with caution. The pip tool is used to install python packages including the snp-pipeline and other packages used by the snp-pipeline. Some newer versions of Python include pip. Check to see if pip is already installed:

```
$ pip -V
```

If pip is not already installed, proceed as follows:

```
Download get-pip.py from https://pip.pypa.io/en/latest/installing.html#install-pip
$ python get-pip.py --user
```

Note: avoid using sudo when installing pip. Some users have experienced problems installing and loading packages when pip is installed using sudo.

## 3.6 Step 6 - Install the SNP Pipeline Python Package

There is more than one way to install the SNP Pipeline depending on whether you intend to work with the source code or just run it.

### 3.6.1 Installation Method 1 for Most Users

This is the recommended installation method for new users.

If you want to run the software without viewing or changing the source code, follow the instructions below.

At the command line:

```
$ pip install --user snp-pipeline
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv snp-pipeline
$ pip install snp-pipeline
```

### 3.6.2 Installation Method 2 for Software Developers

If you intend to work with the source code in the role of a software developer, you should clone the GitHub repository as described in the *Contributing* section of this documentation.

## 3.7 Upgrading SNP Pipeline

If you previously installed with pip, you can upgrade to the newest version from the command line:

```
$ pip install --user --upgrade snp-pipeline
```

## 3.8 Uninstalling SNP Pipeline

If you installed with pip, you can uninstall from the command line:

```
$ pip uninstall snp-pipeline
```

## 3.9 Tips

There is a dependency on the python psutil package. Pip will attempt to install the psutil package automatically when installing snp-pipeline. If it fails with an error message about missing Python.h, you will need to manually install the python-dev package. In Ubuntu, use this command:

```
$ sudo apt-get install python-dev
```

You may need to upgrade your Java Runtime Environment (JRE) to run Picard.

# CHAPTER 4

# Usage

The SNP Pipeline is run from the Unix command line. The pipeline consists of multiple commands.

| Command | Description |
|---|---|
| run_snp_pipeline | This do-it-all script runs the other scripts listed below, comprising all the pipeline steps. This is the same as running cfsan_snp_pipeline run. |
| cfsan_snp_pipeline | This is the main command line tool with subcommands listed below |
| run | This all-in-one command runs all the necessary pipeline subcommands listed below, in the right order, comprising all the pipeline steps |
| data | Copies supplied example data to a work directory |
| index_ref | Indexes the reference genome |
| map_reads | Aligns sample reads to the reference genome |
| call_sites | Finds high-confidence SNPs in each sample |
| filter_regions | Remove SNPs in abnormal regions |
| merge_sites | Combines the SNP positions across all samples into a single unified SNP list file |
| call_consensus | Calls the consensus SNPs for each sample |
| merge_vcfs | Creates a multi-sample VCF file with the SNPs found in all samples |
| snp_matrix | Creates a matrix of SNPs across all samples |
| distance | Computes the SNP distances between all pairs of samples |
| snp_reference | Writes the reference sequence bases at SNP locations to a fasta file |
| collect_metrics | Collects useful coverage and variant statistics about each sample |
| combine_metrics | Creates a table of coverage and variant statistics for all samples |
| purge | Removes the intermediate output files in the “samples” directory to save space upon successful completion of a SNP Pipeline run |

## 4.1 Inputs

Before using the SNP Pipeline, make sure your input data is organized and named the way the pipeline expects. Follow these guidelines:

- No spaces in file names and directory names.
- A fasta genome reference file must exist in a separate directory.
- The samples must be organized with a separate directory for each sample. Each sample directory should contain the fastq files for that sample. The name of the directory should match the name of the sample. When using paired-end fastq files, the forward and reverse files must be in the same directory.
- The script needs to know how to find all the samples. You have two choices:
  1. You can organize all the sample directories under a common parent directory.
  2. You can have sample directories anywhere you like, but you will need to create a file listing the path to all the sample directories.
- The sample fastq files must be named with one of the following file patterns: (*.fastq, *.fq, *.fastq.gz, *.fq.gz). It's okay if different samples are named differently, but the two mate files of paired-end samples must be named with the same extension.
- If there is an outgroup among samples, a file containing the sample ids of the outgroup samples must be created in advance, and the relative or absolute path of this file should be specified in the parameter "FilterRegions_ExtraParams" in the configuration file "snppipeline.conf" (see the description of the parameter "FilterRegions_ExtraParams").

## 4.2 Outputs

By default, the SNP Pipeline generates the following output files. If you need more control over the output, you can run the pipeline one step at a time (not recommended). See *Step-by-Step Workflows*.

- `snplist.txt` : contains a list of the high-confidence SNP positions identified by the phase 1 SNP caller (VarScan) in at least one of the samples. These are the only positions where the consensus caller subsequently looks for SNPs in all samples. The consensus caller often finds SNPs at the same positions in other samples, and those additional SNPs are not listed in the snplist.txt file. While the snplist.txt file has an accurate list of SNP positions, it does not contain the final list of samples having SNPs at those positions. If you need the final set of SNPs per sample, you should not use the snplist.txt file. Instead, refer to the snpma.fasta file or the snpma.vcf file. The corresponding `snplist_preserved.txt` file is produced when snp filtering removes the abnormal snps.
- `consensus.fasta` : for each sample, the consensus base calls at the high-confidence positions where SNPs were detected in any of the samples. The corresponding `consensus_preserved.fasta` file is produced when snp filtering removes the abnormal snps.
- `consensus.vcf` : for each sample, the VCF file of SNPs called, as well as failed SNPs at the high-confidence positions where SNPs were detected in any of the samples. The corresponding `consensus_preserved.vcf` file is produced when snp filtering removes the abnormal snps.
- `snpma.fasta` : the SNP matrix containing the consensus base for each of the samples at the high-confidence positions where SNPs were identified in any of the samples. The matrix contains one row per sample and one column per SNP position. Non-SNP positions are not included in the matrix. The matrix is formatted as a fasta file, with each sequence (all of identical length) corresponding to the SNPs in the correspondingly named sequence. The corresponding `snpma_preserved.fasta` file is produced when snp filtering removes the abnormal snps.

- `snp_distance_pairwise.tsv` : contains the pairwise SNP distance between all pairs of samples. The file is tab-separated, with a header row and three columns identifing the two sequences and their distance. The corresponding `snp_distance_pairwise_preserved.tsv` file is produced when snp filtering removes the abnormal snps.
- `snp_distance_matrix.tsv` : contains a matrix of the SNP distances between all pairs of samples. The file is tab-separated, with a header row and rows and columns for all samples. The corresponding `snp_distance_matrix_preserved.tsv` file is produced when snp filtering removes the abnormal snps.
- `snpma.vcf` : contains the merged multi-sample VCF file identifying the positions and snps for all samples. The corresponding `snpma_preserved.vcf` file is produced when snp filtering removes the abnormal snps.
- `referenceSNP.fasta` : a fasta file containing the reference sequence bases at all the SNP locations. The corresponding `referenceSNP_preserved.fasta` file is produced when snp filtering removes the abnormal snps.
- `metrics` : for each sample, contains the size of the sample, number of reads, alignment rate, pileup depth, and number of SNPs found.
- `metrics.tsv` : a tab-separated table of metrics for all samples containing the size of the samples, number of reads, alignment rate, pileup depth, and number of SNPs found.
- `error.log` : a summary of errors detected during SNP Pipeline execution

## 4.3 All-In-One SNP Pipeline Script

Most users should run the SNP Pipeline by executing the `cfsan_snp_pipeline run` command. This command is easy to use and works equally well on your desktop workstation or on a High Performance Computing cluster. You can find examples of using the command in the sections below.

If you need more flexibility, you can run the individual pipeline scripts one step at a time (not recommended). See *Step-by-Step Workflows*.

## 4.4 Logging

The SNP Pipeline generates log files for each processing step of the pipeline. The logs for each pipeline run are stored in a time-stamped directory under the output directory. If the pipeline is re-run on the same samples, the old log files are kept and a new log directory is created for the new run. For example, the output directory might look like this after two runs:

```
drwx------ 2 me group 4096 Oct 17 16:37 logs-20141017.154428/
drwx------ 2 me group 4096 Oct 17 16:38 logs-20141017.163848/
drwx------ 2 me group 4096 Oct 17 16:37 reference/
-rw------- 1 me group  194 Oct 17 16:38 referenceSNP.fasta
-rw------- 1 me group  182 Oct 17 16:38 referenceSNP_preserved.fasta
-rw------- 1 me group  104 Oct 17 16:38 sampleDirectories.txt
drwx------ 6 me group 4096 Oct 17 16:37 samples/
-rw------- 1 me group 7216 Oct 17 16:38 snplist.txt
-rw------- 1 me group 6824 Oct 17 16:38 snplist_preserved.txt
-rw------- 1 me group  708 Oct 17 16:38 snpma.fasta
-rw------- 1 me group  682 Oct 17 16:38 snpma_preserved.fasta
```

A log file is created for each step of the pipeline for each sample. For performance reasons, the samples are sorted by size and processed largest first. This sorting is reflected in the naming of the log files. The log files are named with a suffix indicating the sample number:

```
-rw------- 1 me group 12045 Oct 17 16:37 indexRef.log
-rw------- 1 me group  1330 Oct 17 16:37 mapReads.log-1
-rw------- 1 me group  1330 Oct 17 16:37 mapReads.log-2
-rw------- 1 me group  1330 Oct 17 16:37 mapReads.log-3
-rw------- 1 me group  1686 Oct 17 16:37 callSites.log-1
-rw------- 1 me group  1686 Oct 17 16:37 callSites.log-2
-rw------- 1 me group  1686 Oct 17 16:37 callSites.log-3
-rw------- 1 me group  1035 Oct 17 16:37 filterRegions.log
-rw------- 1 me group  1275 Oct 17 16:37 mergeSites.log
-rw------- 1 me group  1375 Oct 17 16:37 mergeSites_preserved.log
-rw------- 1 me group  1509 Oct 17 16:37 callConsensus.log-1
-rw------- 1 me group  1509 Oct 17 16:37 callConsensus.log-2
-rw------- 1 me group  1509 Oct 17 16:37 callConsensus.log-3
-rw------- 1 me group  1509 Oct 17 16:37 callConsensus_preserved.log-1
-rw------- 1 me group  1509 Oct 17 16:37 callConsensus_preserved.log-2
-rw------- 1 me group  1509 Oct 17 16:37 callConsensus_preserved.log-3
-rw------- 1 me group  1047 Oct 17 16:37 snpMatrix.log
-rw------- 1 me group  1147 Oct 17 16:37 snpMatrix_preserved.log
-rw------- 1 me group   806 Oct 17 16:37 snpReference.log
-rw------- 1 me group   806 Oct 17 16:37 snpReference_preserved.log
-rw------- 1 me group  1895 Oct 17 16:37 mergeVcfs.log
-rw------- 1 me group  2039 Oct 17 16:37 mergeVcfs_preserved.log
-rw------- 1 me group   887 Oct 17 16:37 distance.log
-rw------- 1 me group   977 Oct 17 16:37 distance_preserved.log
-rw------- 1 me group  2169 Oct 17 16:37 collectMetrics.log-1
-rw------- 1 me group  2169 Oct 17 16:37 collectMetrics.log-2
-rw------- 1 me group  2169 Oct 17 16:37 collectMetrics.log-3
-rw------- 1 me group   983 Oct 17 16:37 combineMetrics.log
```

To determine which samples correspond to which log files, you can either grep the log files for the sample name or inspect the sorted sampleDirectories.txt file to determine the sequential position of the sample. The file names are consistent regardless of whether the pipeline is run on a workstation or HPC cluster.

In addition to the processing log files, the log directory also contains a copy of the configuration file used for each run – capturing the parameters used during the run.

## 4.5 Mirrored Inputs

The SNP Pipeline has the optional capability to create a mirrored copy of the input fasta and fastq files. You might use this feature to avoid polluting the reference directory and sample directories with the intermediate files generated by the snp pipeline. The mirroring function can either create normal copies of the files, or it can create links to the original files – saving both time and disk space. With linked files, you can easily run multiple experiments on the same data or different overlapping sets of samples without having duplicate copies of the original sample files. See the *run* command reference for the mirroring syntax.

The mirroring function creates a "reference" subdirectory and a "samples" subdirectory under the main output directory. One directory per sample is created under the "samples" directory. The generated intermediate files are placed into the mirrored directories, not in the original locations of the inputs. The SNP Pipeline attempts to preserve the time stamps of the original files in the mirrored directories.

Keep in mind the following limitations when mirroring the inputs.

- Some file systems do not support soft (symbolic) links. If you attempt to create a soft link on a file system without the capability, the operation will fail with an error message.
- Hard links cannot be used to link files across two different file systems. The original file and the link must both reside on the same file system.

- Normal file copies should always work, but the copy operation can be lengthy and the duplicate files will consume extra storage space.

## 4.6 High Performance Computing

The SNP Pipeline can be executed on a High Performance Computing cluster. The Grid Engine, SLURM, and Torque job queue managers are supported.

### 4.6.1 Grid Engine

To run the SNP Pipeline on grid engine you must use a configuration file to specify the name of your parallel environment.

Grab the default configuration file:

```
cfsan_snp_pipeline data configurationFile
```

Edit the snppipeline.conf file and make the following change:

```
GridEngine_PEname="myPE" # substitute the name of your PE
```

You may also need to change the `GridEngine_StripJobArraySuffix` configuration parameter if you see qsub illegal dependency errors.

Then run the pipeline with the -c and -Q command line options:

```
cfsan_snp_pipeline run -c snppipeline.conf -Q grid -s mySamplesDir myReference.fasta
```

You can pass extra options to the Grid Engine qsub command by configuring the `GridEngine_QsubExtraParams` parameter in the configuration file. Among other things, you can control which queue the snp-pipeline will use when executing on an HPC with multiple queues.

### 4.6.2 SLURM

To run the SNP Pipeline on SLURM:

```
cfsan_snp_pipeline run -Q slurm -s mySamplesDir myReference.fasta
```

You can pass extra options to the SLURM sbatch command by configuring the `Slurm_SbatchExtraParams` parameter in the configuration file.

### 4.6.3 Torque

To run the SNP Pipeline on torque:

```
cfsan_snp_pipeline run -Q torque -s mySamplesDir myReference.fasta
```

You may need to change the `Torque_StripJobArraySuffix` configuration parameter if you see qsub illegal dependency errors.

You can pass extra options to the Torque qsub command by configuring the `Torque_QsubExtraParams` parameter in the configuration file.

See also: *Performance*.

## 4.7 Tool Selection

The SNP Pipeline lets you choose either the Bowtie2 aligner or the Smalt aligner. Your choice of aligner, as well as the command line options for the aligner are specified in the SNP Pipeline configuration file.

Grab the default configuration file:

```
cfsan_snp_pipeline data configurationFile
```

To run the SNP Pipeline with Bowtie2, edit `snppipeline.conf` with these settings:

```
SnpPipeline_Aligner="bowtie2"
Bowtie2Build_ExtraParams="" # substitute the command line options you want here
Bowtie2Align_ExtraParams="" # substitute the command line options you want here
```

To run the SNP Pipeline with Smalt, edit `snppipeline.conf` with these settings:

```
SnpPipeline_Aligner="smalt"
SmaltIndex_ExtraParams="" # substitute the command line options you want here
SmaltAlign_ExtraParams="" # substitute the command line options you want here
```

Then run the pipeline with the -c command line option:

```
cfsan_snp_pipeline run -c snppipeline.conf -s mySamplesDir myReference.fasta
```

See also *Configuration*.

## 4.8 All-In-One SNP Pipeline Workflows

The sections below give detailed examples of workflows you can execute with the all-in-one `run` command.

*All-In-One Workflow - Lambda Virus*
*All-In-One Workflow - Salmonella Agona*
*All-In-One Workflow - Listeria monocytogenes*

### 4.8.1 All-In-One Workflow - Lambda Virus

The SNP Pipeline software distribution includes a small Lambda Virus data set that can be quickly processed to verify the basic functionality of the software.

Step 1 - Gather data:

```
# The SNP Pipeline distribution includes sample data organized as shown below:
snppipeline/data/lambdaVirusInputs/reference/lambda_virus.fasta
snppipeline/data/lambdaVirusInputs/samples/sample1/sample1_1.fastq
snppipeline/data/lambdaVirusInputs/samples/sample1/sample1_2.fastq
snppipeline/data/lambdaVirusInputs/samples/sample2/sample2_1.fastq
snppipeline/data/lambdaVirusInputs/samples/sample2/sample2_2.fastq
snppipeline/data/lambdaVirusInputs/samples/sample3/sample3_1.fastq
snppipeline/data/lambdaVirusInputs/samples/sample3/sample3_2.fastq
snppipeline/data/lambdaVirusInputs/samples/sample4/sample4_1.fastq
```

(continues on next page)

(continued from previous page)

```
snppipeline/data/lambdaVirusInputs/samples/sample4/sample4_2.fastq

# Copy the supplied test data to a work area:
cd test
cfsan_snp_pipeline data lambdaVirusInputs testLambdaVirus
cd testLambdaVirus
```

Step 2 - Run the SNP Pipeline:

```
# Run the pipeline, specifing the locations of samples and the reference
#
# Specify the following options:
#   -s : samples parent directory
cfsan_snp_pipeline run -s samples reference/lambda_virus.fasta
```

Step 3 - View and verify the results:

Upon successful completion of the pipeline, the snplist.txt file should have 166 entries, and the snplist_preserved.txt should have 137 entries. The SNP Matrix can be found in snpma.fasta and snpma_preserved.fasta. The corresponding reference bases are in the referenceSNP.fasta and referenceSNP_preserved.fasta:

```
# Verify the result files were created
ls -l snplist.txt
ls -l snpma.fasta
ls -l snpma.vcf
ls -l referenceSNP.fasta
ls -l snp_distance_matrix.tsv
ls -l snplist_preserved.txt
ls -l snpma_preserved.fasta
ls -l snpma_preserved.vcf
ls -l referenceSNP_preserved.fasta
ls -l snp_distance_matrix_preserved.tsv

# Verify correct results
cfsan_snp_pipeline data lambdaVirusExpectedResults expectedResults
diff -q -s snplist.txt             expectedResults/snplist.txt
diff -q -s snpma.fasta             expectedResults/snpma.fasta
diff -q -s referenceSNP.fasta      expectedResults/referenceSNP.fasta
diff -q -s snp_distance_matrix.tsv expectedResults/snp_distance_matrix.tsv
diff -q -s snplist_preserved.txt             expectedResults/snplist_preserved.txt
diff -q -s snpma_preserved.fasta             expectedResults/snpma_preserved.fasta
diff -q -s referenceSNP_preserved.fasta      expectedResults/referenceSNP_preserved.
↪fasta
diff -q -s snp_distance_matrix_preserved.tsv expectedResults/snp_distance_matrix_
↪preserved.tsv

# View the per-sample metrics
xdg-open metrics.tsv
```

### 4.8.2 All-In-One Workflow - Salmonella Agona

The Salmonella Agona data set contains a small number of realistic sequences that can be processed in a reasonable amount of time. Due to the large size of real data, the sequences must be downloaded from the NCBI SRA. Follow the instructions below to download and process the data set.

Step 1 - Gather data:

```
# The SNP Pipeline distribution does not include the sample data, but does
#   include information about the sample data, as well as the reference
#   sequence.  The files are organized as shown below:
snppipeline/data/agonaInputs/sha256sumCheck
snppipeline/data/agonaInputs/reference/NC_011149.fasta
snppipeline/data/agonaInputs/sampleList

# Copy the supplied test data to a work area:
mkdir testAgona
cd testAgona
cfsan_snp_pipeline data agonaInputs cleanInputs
cd cleanInputs

# Create sample directories and download sample data from SRA at NCBI. Note that
#   we use the fastq-dump command from the NCBI SRA-toolkit to fetch sample
#   sequences. There are other ways to get the data, but the SRA-toolkit is
#   easy to install, and does a good job of downloading large files.
mkdir samples
< sampleList xargs -I % sh -c 'mkdir samples/%; fastq-dump --gzip --origfmt --split-
↪files --outdir samples/% %;'

# Check the data
#   The original data was used to generate a hash as follows:
#     sha256sum sampleList reference/*.fasta samples/*/*.fastq.gz > sha256sumCheck
#   The command below checks the downloaded data (and the reference sequence) against␣
↪the
#     hashes that are saved in the sha256sumCheck file using sha256sum command, which␣
↪is
#     generally available on unix systems.
sha256sum -c sha256sumCheck
cd ..
```

Step 2 - Run the SNP Pipeline:

```
# Run the pipeline
# Specify the following options:
#   -m : mirror the input samples and reference files
#   -o : output directory
#   -s : samples parent directory
cfsan_snp_pipeline run -m soft -o outputDirectory -s cleanInputs/samples cleanInputs/
↪reference/NC_011149.fasta
```

Step 3 - View and verify the results:

Upon successful completion of the pipeline, the snplist.txt file should have 2620 entries, and the snplist_preserved.txt should have 233 entries. The SNP Matrix can be found in snpma.fasta. The corresponding reference bases are in the files referenceSNP.fasta and referenceSNP_preserved.fasta:

```
# Verify the result files were created
ls -l outputDirectory/snplist.txt
ls -l outputDirectory/snpma.fasta
ls -l outputDirectory/snpma.vcf
ls -l outputDirectory/referenceSNP.fasta
ls -l outputDirectory/snp_distance_matrix.tsv
ls -l outputDirectory/snplist_preserved.txt
ls -l outputDirectory/snpma_preserved.fasta
ls -l outputDirectory/snpma_preserved.vcf
```

(continues on next page)

(continued from previous page)

```
ls -l outputDirectory/referenceSNP_preserved.fasta
ls -l outputDirectory/snp_distance_matrix_preserved.tsv

# Verify correct results
cfsan_snp_pipeline data agonaExpectedResults expectedResults
diff -q -s outputDirectory/snplist.txt             expectedResults/snplist.txt
diff -q -s outputDirectory/snpma.fasta             expectedResults/snpma.fasta
diff -q -s outputDirectory/referenceSNP.fasta      expectedResults/referenceSNP.fasta
diff -q -s outputDirectory/snp_distance_matrix.tsv expectedResults/snp_distance_
↪matrix.tsv
diff -q -s outputDirectory/snplist_preserved.txt             expectedResults/snplist_
↪preserved.txt
diff -q -s outputDirectory/snpma_preserved.fasta             expectedResults/snpma_
↪preserved.fasta
diff -q -s outputDirectory/referenceSNP_preserved.fasta      expectedResults/
↪referenceSNP_preserved.fasta
diff -q -s outputDirectory/snp_distance_matrix_preserved.tsv expectedResults/snp_
↪distance_matrix_preserved.tsv

# View the per-sample metrics
xdg-open outputDirectory/metrics.tsv
```

### 4.8.3 All-In-One Workflow - Listeria monocytogenes

This Listeria monocytogene data set is based on an oubreak investigation related to contamination in stone fruit. It only contains environmental/produce isolates, though the full investigation contained data obtained from clinical samples as well. Due to the large size of the data, the sequences must be downloaded from the NCBI SRA. The instructions below show how to create the data set and process it.

This workflow illustrates how to run the SNP Pipeline on a High Performance Computing cluster (HPC) running the Torque job queue manager. If you do not have a cluster available, you can still work through this example – just remove the `-Q torque` command line option in step 2.

Step 1 - Create dataset:

```
# The SNP Pipeline distribution does not include the sample data, but does
#   include information about the sample data, as well as the reference
#   sequence.  The files are organized as shown below:
snppipeline/data/listeriaInputs/sha256sumCheck
snppipeline/data/listeriaInputs/reference/CFSAN023463.HGAP.draft.fasta
snppipeline/data/listeriaInputs/sampleList

# Copy the supplied test data to a work area:
mkdir testDir
cd testDir
cfsan_snp_pipeline data listeriaInputs cleanInputs
cd cleanInputs

# Create sample directories and download sample data from SRA at NCBI. Note that
#   we use the fastq-dump command from the NCBI SRA-toolkit to fetch sample
#   sequences. There are other ways to get the data, but the SRA-toolkit is
#   easy to install, and does a good job of downloading large files.
mkdir samples
< sampleList xargs -I % sh -c ' mkdir samples/%; fastq-dump --gzip --split-files --
↪outdir samples/% %;'
```

(continues on next page)

(continued from previous page)

```
# Check the data
#   The original data was used to generate a hash as follows:
#     sha256sum sampleList reference/*.fasta samples/*/*.fastq.gz > sha256sumCheck
#   The command below checks the downloaded data (and the reference sequence) against␣
↪the
#     hashes that are saved in the sha256sumCheck file using sha256sum command, which␣
↪is
#     generally available on unix systems.
sha256sum -c sha256sumCheck
cd ..
```

Step 2 - Run the SNP Pipeline:

There are a couple of parameters you may need to adjust for this analysis or other analysis work that your do. These parameters are the number of CPU cores that are used, and the amount of memory that is used by the java virtual machine. Both can be set in a configuration file you can pass to `cfsan_snp_pipeline run` with the `-c` option. See *Performance*.

Launch the pipeline:

```
# Run the pipeline.
# Specify the following options:
#   -m : mirror the input samples and reference files
#   -Q : HPC job queue manager
#   -o : output directory
#   -s : samples parent directory
cfsan_snp_pipeline run -m soft -Q torque -o outputDirectory -s cleanInputs/samples␣
↪cleanInputs/reference/CFSAN023463.HGAP.draft.fasta
```

Step 3 - View and verify the results:

Upon successful completion of the pipeline, the snplist.txt file should have 10,102 entries, and the snplist_preserved.txt file should have 1,040 entries. The SNP Matrix can be found in snpma.fasta and snpma_preserved.fasta. The corresponding reference bases are in the referenceSNP.fasta and referenceSNP_preserved.fasta:

```
# Verify the result files were created
ls -l outputDirectory/snplist.txt
ls -l outputDirectory/snpma.fasta
ls -l outputDirectory/snpma.vcf
ls -l outputDirectory/referenceSNP.fasta
ls -l outputDirectory/snp_distance_matrix.tsv
ls -l outputDirectory/snplist_preserved.txt
ls -l outputDirectory/snpma_preserved.fasta
ls -l outputDirectory/snpma_preserved.vcf
ls -l outputDirectory/referenceSNP_preserved.fasta
ls -l outputDirectory/snp_distance_matrix_preserved.tsv

# Verify correct results
cfsan_snp_pipeline data listeriaExpectedResults expectedResults
diff -q -s outputDirectory/snplist.txt             expectedResults/snplist.txt
diff -q -s outputDirectory/snpma.fasta             expectedResults/snpma.fasta
diff -q -s outputDirectory/referenceSNP.fasta      expectedResults/referenceSNP.fasta
diff -q -s outputDirectory/snp_distance_matrix.tsv expectedResults/snp_distance_
↪matrix.tsv
diff -q -s outputDirectory/snplist_preserved.txt            expectedResults/snplist_
↪preserved.txt
```

(continues on next page)

(continued from previous page)

```
diff -q -s outputDirectory/snpma_preserved.fasta             expectedResults/snpma_
↪preserved.fasta
diff -q -s outputDirectory/referenceSNP_preserved.fasta      expectedResults/
↪referenceSNP_preserved.fasta
diff -q -s outputDirectory/snp_distance_matrix_preserved.tsv expectedResults/snp_
↪distance_matrix_preserved.tsv

# View the per-sample metrics
xdg-open outputDirectory/metrics.tsv
```

## 4.9 Step-by-Step Workflows

The `cfsan_snp_pipeline run` command described above provides a simple and powerful interface for running all the pipeline steps from a single command. If you need more control over the inputs, outputs, or processing steps, you can run the pipeline one step at a time, however this is not recommended.

The sections below give detailed examples of workflows you can run with the component tools of the pipeline.

*Step-by-Step Workflow - Lambda Virus*
*Step-by-Step Workflow - Salmonella Agona*
*Step-by-Step Workflow - General Case*

### 4.9.1 Step-by-Step Workflow - Lambda Virus

The SNP Pipeline software distribution includes a small Lambda Virus data set that can be quickly processed to verify the basic functionality of the software.

Step 1 - Gather data:

```
# The SNP Pipeline distribution includes sample data organized as shown below:
snppipeline/data/lambdaVirusInputs/reference/lambda_virus.fasta
snppipeline/data/lambdaVirusInputs/samples/sample1/sample1_1.fastq
snppipeline/data/lambdaVirusInputs/samples/sample1/sample1_2.fastq
snppipeline/data/lambdaVirusInputs/samples/sample2/sample2_1.fastq
snppipeline/data/lambdaVirusInputs/samples/sample2/sample2_2.fastq
snppipeline/data/lambdaVirusInputs/samples/sample3/sample3_1.fastq
snppipeline/data/lambdaVirusInputs/samples/sample3/sample3_2.fastq
snppipeline/data/lambdaVirusInputs/samples/sample4/sample4_1.fastq
snppipeline/data/lambdaVirusInputs/samples/sample4/sample4_2.fastq

# Copy the supplied test data to a work area:
cd test
cfsan_snp_pipeline data lambdaVirusInputs testLambdaVirus
cd testLambdaVirus
```

Step 2 - Prep work:

```
# Create files of sample directories and fastQ files:
ls -d samples/* > sampleDirectories.txt
rm sampleFullPathNames.txt 2>/dev/null
```

(continues on next page)

(continued from previous page)

```
cat sampleDirectories.txt | while read dir; do echo $dir/*.fastq* >>␣
↪sampleFullPathNames.txt; done
# Determine the number of CPU cores in your computer
numCores=$(grep -c ^processor /proc/cpuinfo 2>/dev/null || sysctl -n hw.ncpu)
```

Step 3 - Prep the reference:

```
cfsan_snp_pipeline index_ref reference/lambda_virus.fasta
```

Step 4 - Align the samples to the reference:

```
# Align each sample, one at a time, using all CPU cores
export Bowtie2Align_ExtraParams="--reorder -X 1000"
export SamtoolsSamFilter_ExtraParams="-F 4 -q 30"
cat sampleFullPathNames.txt | xargs -n 2 -L 1 cfsan_snp_pipeline map_reads --threads
↪$numCores reference/lambda_virus.fasta
```

Step 5 - Find the sites having high-confidence SNPs:

```
# Process the samples in parallel using all CPU cores
export SamtoolsMpileup_ExtraParams="-q 0 -Q 13 -A"
export VarscanMpileup2snp_ExtraParams="--min-var-freq 0.90"
cat sampleDirectories.txt | xargs -n 1 -P $numCores cfsan_snp_pipeline call_sites␣
↪reference/lambda_virus.fasta
```

Step 6 - Identify regions with abnormal SNP density and remove SNPs in these regions:

```
cfsan_snp_pipeline filter_regions --window_size 1000 125 15 --max_snp 3 2 1 -n var.
↪flt.vcf sampleDirectories.txt reference/lambda_virus.fasta
```

Step 7 - Combine the SNP positions across all samples into the SNP list file:

```
cfsan_snp_pipeline merge_sites -n var.flt.vcf -o snplist.txt sampleDirectories.txt␣
↪sampleDirectories.txt.OrigVCF.filtered
cfsan_snp_pipeline merge_sites -n var.flt_preserved.vcf -o snplist_preserved.txt␣
↪sampleDirectories.txt sampleDirectories.txt.PresVCF.filtered
```

Step 8 - Call the consensus base at SNP positions for each sample:

```
# Process the samples in parallel using all CPU cores
cat sampleDirectories.txt | xargs -n 1 -P $numCores -I XX cfsan_snp_pipeline call_
↪consensus -l snplist.txt --minConsDpth 3 --vcfFileName consensus.vcf --vcfRefName␣
↪lambda_virus.fasta -o XX/consensus.fasta XX/reads.all.pileup
cat sampleDirectories.txt | xargs -n 1 -P $numCores -I XX cfsan_snp_pipeline call_
↪consensus -l snplist_preserved.txt --minConsDpth 3 --vcfFileName consensus_
↪preserved.vcf --vcfRefName lambda_virus.fasta -o XX/consensus_preserved.fasta -e XX/
↪var.flt_removed.vcf XX/reads.all.pileup
```

Step 9 - Create the SNP matrix:

```
cfsan_snp_pipeline snp_matrix -c consensus.fasta -o snpma.fasta sampleDirectories.txt.
↪OrigVCF.filtered
cfsan_snp_pipeline snp_matrix -c consensus_preserved.fasta -o snpma_preserved.fasta␣
↪sampleDirectories.txt.PresVCF.filtered
```

Step 10 - Create the reference base sequence:

```
cfsan_snp_pipeline snp_reference -l snplist.txt -o referenceSNP.fasta reference/
↪lambda_virus.fasta
cfsan_snp_pipeline snp_reference -l snplist_preserved.txt -o referenceSNP_preserved.
↪fasta reference/lambda_virus.fasta
```

Step 11 - Collect metrics for each sample:

```
cat sampleDirectories.txt | xargs -n 1 -P $numCores -I XX cfsan_snp_pipeline collect_
↪metrics -o XX/metrics XX reference/lambda_virus.fasta
```

Step 12 - Tabulate the metrics for all samples:

```
cfsan_snp_pipeline combine_metrics -n metrics -o metrics.tsv sampleDirectories.txt
```

Step 13 - Merge the VCF files for all samples into a multi-sample VCF file:

```
cfsan_snp_pipeline merge_vcfs -n consensus.vcf -o snpma.vcf sampleDirectories.txt.
↪OrigVCF.filtered
cfsan_snp_pipeline merge_vcfs -n consensus_preserved.vcf -o snpma_preserved.vcf␣
↪sampleDirectories.txt.PresVCF.filtered
```

Step 14 - Compute the SNP distances between samples:

```
cfsan_snp_pipeline distance -p snp_distance_pairwise.tsv -m snp_distance_matrix.tsv␣
↪snpma.fasta
cfsan_snp_pipeline distance -p snp_distance_pairwise_preserved.tsv -m snp_distance_
↪matrix_preserved.tsv snpma_preserved.fasta
```

Step 15 - View and verify the results:

Upon successful completion of the pipeline, the snplist.txt file should have 166 entries. The SNP Matrix can be found in snpma.fasta. The corresponding reference bases are in the referenceSNP.fasta file:

```
# Verify the result files were created
ls -l snplist.txt
ls -l snpma.fasta
ls -l snpma.vcf
ls -l referenceSNP.fasta
ls -l snp_distance_matrix.tsv
ls -l snplist_preserved.txt
ls -l snpma_preserved.fasta
ls -l snpma_preserved.vcf
ls -l referenceSNP_preserved.fasta
ls -l snp_distance_matrix_preserved.tsv

# Verify correct results
cfsan_snp_pipeline data lambdaVirusExpectedResults expectedResults
diff -q -s snplist.txt             expectedResults/snplist.txt
diff -q -s snpma.fasta             expectedResults/snpma.fasta
diff -q -s referenceSNP.fasta      expectedResults/referenceSNP.fasta
diff -q -s snp_distance_matrix.tsv expectedResults/snp_distance_matrix.tsv
diff -q -s snplist_preserved.txt             expectedResults/snplist_preserved.txt
diff -q -s snpma_preserved.fasta             expectedResults/snpma_preserved.fasta
diff -q -s referenceSNP_preserved.fasta      expectedResults/referenceSNP_preserved.
↪fasta
diff -q -s snp_distance_matrix_preserved.tsv expectedResults/snp_distance_matrix_
↪preserved.tsv
```

(continues on next page)

(continued from previous page)

```
# View the per-sample metrics
xdg-open metrics.tsv
```

### 4.9.2 Step-by-Step Workflow - Salmonella Agona

The Salmonella Agona data set contains realistic sequences that can be processed in a reasonable amount of time. Due to the large size of real data, the sequences must be downloaded from the NCBI SRA. Follow the instructions below to download and process the data set.

Step 1 - Gather data:

```
# The SNP Pipeline distribution does not include the sample data, but does
#   include information about the sample data, as well as the reference
#   sequence.  The files are organized as shown below:
snppipeline/data/agonaInputs/sha256sumCheck
snppipeline/data/agonaInputs/reference/NC_011149.fasta
snppipeline/data/agonaInputs/sampleList

# Copy the supplied test data to a work area:
mkdir testAgona
cd testAgona
cfsan_snp_pipeline data agonaInputs .

# Create sample directories and download sample data from SRA at NCBI. Note that
#   we use the fastq-dump command from the NCBI SRA-toolkit to fetch sample
#   sequences. There are other ways to get the data, but the SRA-toolkit is
#   easy to install, and does a good job of downloading large files.
mkdir samples
< sampleList xargs -I % sh -c 'mkdir samples/%; fastq-dump --gzip --origfmt --split-
↪files --outdir samples/% %;'

# Check the data
#   The original data was used to generate a hash as follows:
#     sha256sum sampleList reference/*.fasta samples/*/*.fastq.gz > sha256sumCheck
#   The command below checks the downloaded data (and the reference sequence) against␣
↪the
#     hashes that are saved in the sha256sumCheck file using sha256sum command, which␣
↪is
#     generally available on unix systems.
sha256sum -c sha256sumCheck
```

Step 2 - Prep work:

```
# Create files of sample directories and fastQ files:
ls -d samples/* > sampleDirectories.txt
rm sampleFullPathNames.txt 2>/dev/null
cat sampleDirectories.txt | while read dir; do echo $dir/*.fastq* >>␣
↪sampleFullPathNames.txt; done
# Determine the number of CPU cores in your computer
numCores=$(grep -c ^processor /proc/cpuinfo 2>/dev/null || sysctl -n hw.ncpu)
```

Step 3 - Prep the reference:

```
cfsan_snp_pipeline index_ref reference/NC_011149.fasta
```

Step 4 - Align the samples to the reference:

```
# Align each sample, one at a time, using all CPU cores
export Bowtie2Align_ExtraParams="--reorder -X 1000"
export SamtoolsSamFilter_ExtraParams="-F 4 -q 30"
cat sampleFullPathNames.txt | xargs -n 2 -L 1 cfsan_snp_pipeline map_reads --threads
↪$numCores reference/NC_011149.fasta
```

Step 5 - Find the sites having high-confidence SNPs:

```
# Process the samples in parallel using all CPU cores
export SamtoolsMpileup_ExtraParams="-q 0 -Q 13 -A"
export VarscanMpileup2snp_ExtraParams="--min-var-freq 0.90"
cat sampleDirectories.txt | xargs -n 1 -P $numCores cfsan_snp_pipeline call_sites␣
↪reference/NC_011149.fasta
```

Step 6 - Identify regions with abnormal SNP density and remove SNPs in these regions:

```
cfsan_snp_pipeline filter_regions --window_size 1000 125 15 --max_snp 3 2 1 -n var.
↪flt.vcf sampleDirectories.txt reference/NC_011149.fasta
```

Step 7 - Combine the SNP positions across all samples into the SNP list file:

```
cfsan_snp_pipeline merge_sites -n var.flt.vcf -o snplist.txt sampleDirectories.txt␣
↪sampleDirectories.txt.OrigVCF.filtered
cfsan_snp_pipeline merge_sites -n var.flt_preserved.vcf -o snplist_preserved.txt␣
↪sampleDirectories.txt sampleDirectories.txt.PresVCF.filtered
```

Step 8 - Call the consensus base at SNP positions for each sample:

```
# Process the samples in parallel using all CPU cores
cat sampleDirectories.txt | xargs -n 1 -P $numCores -I XX cfsan_snp_pipeline call_
↪consensus -l snplist.txt --minConsDpth 3 --vcfFileName consensus.vcf --vcfRefName␣
↪NC_011149.fasta -o XX/consensus.fasta XX/reads.all.pileup
cat sampleDirectories.txt | xargs -n 1 -P $numCores -I XX cfsan_snp_pipeline call_
↪consensus -l snplist_preserved.txt --minConsDpth 3 --vcfFileName consensus_
↪preserved.vcf --vcfRefName NC_011149.fasta -o XX/consensus_preserved.fasta -e XX/
↪var.flt_removed.vcf XX/reads.all.pileup
```

Step 9 - Create the SNP matrix:

```
cfsan_snp_pipeline snp_matrix -c consensus.fasta -o snpma.fasta sampleDirectories.txt.
↪OrigVCF.filtered
cfsan_snp_pipeline snp_matrix -c consensus_preserved.fasta -o snpma_preserved.fasta␣
↪sampleDirectories.txt.PresVCF.filtered
```

Step 10 - Create the reference base sequence:

```
cfsan_snp_pipeline snp_reference -l snplist.txt -o referenceSNP.fasta reference/NC_
↪011149.fasta
cfsan_snp_pipeline snp_reference -l snplist_preserved.txt -o referenceSNP_preserved.
↪fasta reference/NC_011149.fasta
```

Step 11 - Collect metrics for each sample:

```
cat sampleDirectories.txt | xargs -n 1 -P $numCores -I XX cfsan_snp_pipeline collect_
↪metrics -o XX/metrics XX reference/NC_011149.fasta
```

Step 12 - Tabulate the metrics for all samples:

```
cfsan_snp_pipeline combine_metrics -n metrics -o metrics.tsv sampleDirectories.txt
```

Step 13 - Merge the VCF files for all samples into a multi-sample VCF file:

```
cfsan_snp_pipeline merge_vcfs -n consensus.vcf -o snpma.vcf sampleDirectories.txt.
↪OrigVCF.filtered
cfsan_snp_pipeline merge_vcfs -n consensus_preserved.vcf -o snpma_preserved.vcf␣
↪sampleDirectories.txt.PresVCF.filtered
```

Step 14 - Compute the SNP distances between samples:

```
cfsan_snp_pipeline distance -p snp_distance_pairwise.tsv -m snp_distance_matrix.tsv␣
↪snpma.fasta
cfsan_snp_pipeline distance -p snp_distance_pairwise_preserved.tsv -m snp_distance_
↪matrix_preserved.tsv snpma_preserved.fasta
```

Step 15 - View and verify the results:

Upon successful completion of the pipeline, the snplist.txt file should have 2620 entries. The SNP Matrix can be found in snpma.fasta. The corresponding reference bases are in the referenceSNP.fasta file:

```
# Verify the result files were created
ls -l snplist.txt
ls -l snpma.fasta
ls -l snpma.vcf
ls -l referenceSNP.fasta
ls -l snp_distance_matrix.tsv
ls -l snplist_preserved.txt
ls -l snpma_preserved.fasta
ls -l snpma_preserved.vcf
ls -l referenceSNP_preserved.fasta
ls -l snp_distance_matrix_preserved.tsv

# Verify correct results
cfsan_snp_pipeline data agonaExpectedResults expectedResults
diff -q -s snplist.txt             expectedResults/snplist.txt
diff -q -s snpma.fasta             expectedResults/snpma.fasta
diff -q -s referenceSNP.fasta      expectedResults/referenceSNP.fasta
diff -q -s snp_distance_matrix.tsv expectedResults/snp_distance_matrix.tsv
diff -q -s snplist_preserved.txt             expectedResults/snplist_preserved.txt
diff -q -s snpma_preserved.fasta             expectedResults/snpma_preserved.fasta
diff -q -s referenceSNP_preserved.fasta      expectedResults/referenceSNP_preserved.
↪fasta
diff -q -s snp_distance_matrix_preserved.tsv expectedResults/snp_distance_matrix_
↪preserved.tsv

# View the per-sample metrics
xdg-open metrics.tsv
```

### 4.9.3 Step-by-Step Workflow - General Case

Note: the step-by-step workflows are not recommended. Most users should run the pipeline with the all-in-one `cfsan_snp_pipeline run` command.

Step 1 - Gather data:

You will need the following data:

- Reference genome
- Fastq input files for multiple samples

Organize the data into separate directories for each sample as well as the reference. One possible directory layout is shown below. Note the mix of paired and unpaired samples:

```
./myProject/reference/my_reference.fasta
./myProject/samples/sample1/sampleA.fastq
./myProject/samples/sample2/sampleB.fastq
./myProject/samples/sample3/sampleC_1.fastq
./myProject/samples/sample3/sampleC_2.fastq
./myProject/samples/sample4/sampleD_1.fastq
./myProject/samples/sample4/sampleD_2.fastq
```

Step 2 - Prep work:

```
# Optional step: Copy your input data to a safe place:
cp -r myProject myProjectClean
# The SNP pipeline will generate additional files into the reference and sample␣
↪directories
cd myProject

# Create file of sample directories:
ls -d samples/* > sampleDirectories.txt

# get the *.fastq or *.fq files in each sample directory, possibly compresessed, on␣
↪one line per sample, ready to feed to bowtie
TMPFILE1=$(mktemp tmp.fastqs.XXXXXXXX)
cat sampleDirectories.txt | while read dir; do echo $dir/*.fastq* >> $TMPFILE1; echo
↪$dir/*.fq* >> $TMPFILE1; done
grep -v '*.fq*' $TMPFILE1 | grep -v '*.fastq*' > sampleFullPathNames.txt
rm $TMPFILE1

# Determine the number of CPU cores in your computer
numCores=$(grep -c ^processor /proc/cpuinfo 2>/dev/null || sysctl -n hw.ncpu)
```

Step 3 - Prep the reference:

```
cfsan_snp_pipeline index_ref reference/my_reference.fasta
```

Step 4 - Align the samples to the reference:

```
# Align each sample, one at a time, using all CPU cores
export Bowtie2Align_ExtraParams="--reorder -X 1000"
export SamtoolsSamFilter_ExtraParams="-F 4 -q 30"
cat sampleFullPathNames.txt | xargs -n 2 -L 1 cfsan_snp_pipeline map_reads --threads
↪$numCores reference/my_reference.fasta
```

Step 5 - Find the sites having high-confidence SNPs:

```
# Process the samples in parallel using all CPU cores
export SamtoolsMpileup_ExtraParams="-q 0 -Q 13 -A"
export VarscanMpileup2snp_ExtraParams="--min-var-freq 0.90"
cat sampleDirectories.txt | xargs -n 1 -P $numCores cfsan_snp_pipeline call_sites␣
↪reference/my_reference.fasta
```

Step 6 - Identify regions with abnormal SNP density and remove SNPs in these regions:

```
cfsan_snp_pipeline filter_regions --window_size 1000 125 15 --max_snp 3 2 1 -n var.
↪flt.vcf sampleDirectories.txt reference/my_reference.fasta
```

Step 7 - Combine the SNP positions across all samples into the SNP list file:

```
cfsan_snp_pipeline merge_sites -n var.flt.vcf -o snplist.txt sampleDirectories.txt␣
↪sampleDirectories.txt.OrigVCF.filtered
cfsan_snp_pipeline merge_sites -n var.flt_preserved.vcf -o snplist_preserved.txt␣
↪sampleDirectories.txt sampleDirectories.txt.PresVCF.filtered
```

Step 8 - Call the consensus base at SNP positions for each sample:

```
# Process the samples in parallel using all CPU cores
cat sampleDirectories.txt | xargs -n 1 -P $numCores -I XX cfsan_snp_pipeline call_
↪consensus -l snplist.txt --minConsDpth 3 --vcfFileName consensus.vcf -o XX/
↪consensus.fasta XX/reads.all.pileup
cat sampleDirectories.txt | xargs -n 1 -P $numCores -I XX cfsan_snp_pipeline call_
↪consensus -l snplist_preserved.txt --minConsDpth 3 --vcfFileName consensus_
↪preserved.vcf -o XX/consensus_preserved.fasta -e XX/var.flt_removed.vcf XX/reads.
↪all.pileup
```

Step 9 - Create the SNP matrix:

```
cfsan_snp_pipeline snp_matrix -c consensus.fasta -o snpma.fasta sampleDirectories.txt.
↪OrigVCF.filtered
cfsan_snp_pipeline snp_matrix -c consensus_preserved.fasta -o snpma_preserved.fasta␣
↪sampleDirectories.txt.PresVCF.filtered
```

Step 10 - Create the reference base sequence:

```
# Note the .fasta file extension
cfsan_snp_pipeline snp_reference -l snplist.txt -o referenceSNP.fasta reference/my_
↪reference.fasta
cfsan_snp_pipeline snp_reference -l snplist_preserved.txt -o referenceSNP_preserved.
↪fasta reference/my_reference.fasta
```

Step 11 - Collect metrics for each sample:

```
cat sampleDirectories.txt | xargs -n 1 -P $numCores -I XX cfsan_snp_pipeline collect_
↪metrics -o XX/metrics XX reference/my_reference.fasta
```

Step 12 - Tabulate the metrics for all samples:

```
cfsan_snp_pipeline combine_metrics -n metrics -o metrics.tsv sampleDirectories.txt
```

Step 13 - Merge the VCF files for all samples into a multi-sample VCF file:

```
cfsan_snp_pipeline merge_vcfs -n consensus.vcf -o snpma.vcf sampleDirectories.txt.
↪OrigVCF.filtered
cfsan_snp_pipeline merge_vcfs -n consensus_preserved.vcf -o snpma_preserved.vcf␣
↪sampleDirectories.txt.PresVCF.filtered
```

Step 14 - Compute the SNP distances between samples:

```
cfsan_snp_pipeline distance -p snp_distance_pairwise.tsv -m snp_distance_matrix.tsv␣
↪snpma.fasta
cfsan_snp_pipeline distance -p snp_distance_pairwise_preserved.tsv -m snp_distance_
↪matrix_preserved.tsv snpma.fasta
```

Step 15 - View the results:

Upon successful completion of the pipeline, the snplist.txt identifies the SNP positions in all samples. The SNP Matrix can be found in snpma.fasta. The corresponding reference bases are in the referenceSNP.fasta file:

```
ls -l snplist.txt
ls -l snpma.fasta
ls -l snpma.vcf
ls -l referenceSNP.fasta
ls -l snp_distance_matrix.tsv
ls -l snplist_preserved.txt
ls -l snpma_preserved.fasta
ls -l snpma_preserved.vcf
ls -l referenceSNP_preserved.fasta
ls -l snp_distance_matrix_preserved.tsv

# View the per-sample metrics
xdg-open metrics.tsv
```

## 4.10 Duplicate Read Removal

Prior to creating the pileup and calling snps, the pipeline detects and removes duplicate reads from the sample BAM files. When duplicates are found, the highest quality read among the duplicates is retained. Removing duplicate reads slightly reduces the depth of coverage in pileups and will sometimes impact the number of called snps. The number of called snps could either increase or decrease depending on whether reference-supporting or variant-supporting reads are removed. Removing duplicate reads impacts the subsequent application of virtually all snp filters: depth, variant allele frequency, strand bias, strand depth, and high density snp filtering.

Duplicate reads are removed with the `Picard` software tool which must be installed for this functionality.

You can disable this step and keep the duplicate reads by configuring the `RemoveDuplicateReads` parameter in the configuration file.

You can customize the picard MarkDuplicates behavior to some extent by configuring the `PicardMarkDuplicates_ExtraParams` parameter in the configuration file.

Duplicate read removal works best when the read names in the fastq files are in the original Illumina format. When downloading fastq files from NCBI with `fastq-dump`, you should use the `--origfmt` command line option. See *Why are there no optical duplicates?*

If you see `No space left on device` errors, you should set either the `TMPDIR` or `TMP_DIR` environment variable to a directory with plenty of space for temp files:

```
export TMPDIR=/scratch/tmp
```

More information about the Picard MarkDuplicates tool can be found here:

- https://broadinstitute.github.io/picard/command-line-overview.html#MarkDuplicates
- http://gatkforums.broadinstitute.org/gatk/discussion/6747/how-to-mark-duplicates-with-markduplicates-or-markduplicateswithm
- http://broadinstitute.github.io/picard/faq.html

See also *Configuration*.

## 4.11 Local Realignment

When reads are mapped to the reference, each read is mapped independently. The reads may be misaligned around insertions or deletions. The local realignment process attempts to minimize the total number of mismatched bases in all the reads around the indels.

The reads are realigned with the `GATK` software (prior to version 4.0) which must be installed for this functionality. It is a two step process. First, the pipeline identifies regions where indels are likely. Then, the reads are realigned in the identified regions.

Local realignment can be a time-consuming process. You can disable this step by configuring the `EnableLocalRealignment` parameter in the configuration file.

More information about the GATK indel realigner can be found here:

- https://software.broadinstitute.org/gatk/documentation/tooldocs/3.8-0/org_broadinstitute_gatk_tools_walkers_indels_IndelRealigner.php
- https://software.broadinstitute.org/gatk/documentation/article.php?id=38

See also *Configuration*.

## 4.12 SNP Filtering

The SNP Pipeline removes abnormal SNPs from the ends of contigs and from regions where many SNPs are found in close proximity. The pipeline runs both ways, with SNP filtering, and without SNP filtering, generating pairs of output files. You can compare the output files to determine which positions were filtered. The filtered output files are named with the `_preserved` suffix, for example:

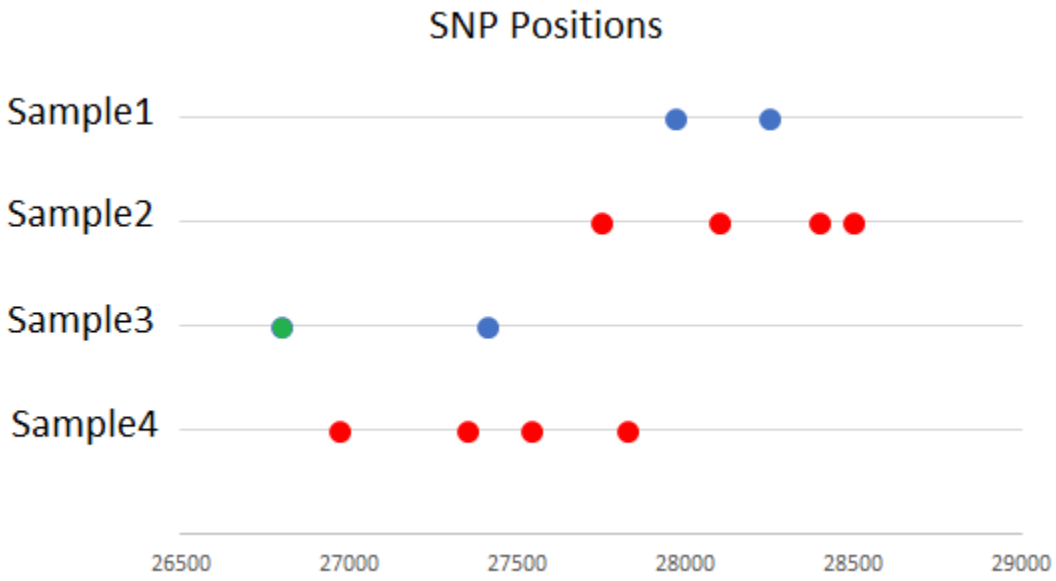
- snplist.txt : contains the unfiltered SNP positions with abnormal SNPs included
- snplist_preserved.txt : contains the filtered SNP positions without abnormal SNPs
- snpma.fasta : contains the unfiltered SNP matrix with abnormal SNPs included
- snpma_preserved.fasta : contains the filtered SNP matrix without abnormal SNPs

Other output files are named similarly.

The SNP filtering is performed by the `filter_regions` command. It runs after the phase 1 SNP detection and impacts all subsequent processing steps.

Dense regions found in any sample are filtered from all the other samples by default. In this mode, if you add or remove a sample from your analysis it may affect the final SNPs detected in all other samples. If you choose to do so, the dense regions can be filtered from each sample individually with the `--mode each` command line option. See *FilterRegions_ExtraParams*.

The drawing below depicts the behavior of the SNP Filtering.




Sample 2 has enough snps in close proximity to form a dense region shown in red. These snps are filtered regardless of the `--mode` option.

Sample 4 has enough snps in close proximity to form a dense region shown in red. These snps are filtered regardless of the `--mode` option.

Sample 1 has two snps which by themselves do not form a dense region, but when the `--mode all` option is used, the snps are engulfed by the dense region in sample 2 and removed.

Sample 3 has two snps which by themselves do not form a dense region. When the `--mode all` option is used, the blue snp is engulfed by the dense region in sample 4 and removed. The green snp is not filtered regardless of the `--mode` option because it is not within a dense region of any other sample. Merely being nearby a dense region in another sample does not cause snp filtering.

The sensitivity of the SNP filtering can be controlled with parameters in the configuration file by setting values in `FilterRegions_ExtraParams`. You can control the length of end-of-contig trimming, dense region window size, and maximum snps allowed within the window. It's possible to configure the SNP filter to find dense regions in multiple window sizes, each with a different maximum allowed number of SNPs. For example, you can allow no more than 3 SNPs per 1000 bases and 2 SNPs per 100 bases. See *Configuration*.

### 4.12.1 SNP Filtering With Outgroups

If there is an outgroup among the samples, you should configure the pipeline to exclude the outgroup samples from snp filtering. To exclude the outgroup samples:

First, make a file containing the sample ids of the outgroup samples, one sample id per line. The sample id is the name of the last subdirectory in the path to the sample:

```
SRR1556289
SRR1556294
```

Grab the default configuration file:

```
cfsan_snp_pipeline data configurationFile
```

Edit `snppipeline.conf`, and change the `FilterRegions_ExtraParams` parameter:

```
Add the --out_group option with the path to the file containing the outgroup sample␣
↪ids.
```

Then run the snp pipeline with the -c command line options:

```
cfsan_snp_pipeline run -c snppipeline.conf  -s mySamplesDir myReference.fasta
```

See also *Configuration*.

## 4.13 Excessive SNPs

Samples having many SNPs relative to the reference can slow the performance of the SNP Pipeline and greatly increase the size of the SNP matrix. The SNP Pipeline has the capability to exclude samples from processing when those samples have too many SNPs. This function excludes entire samples, not just regions within a sample. The samples with excessive SNPs exceeding a user-specified limit are excluded from the snp list, snp matrix, and snpma.vcf files.

There is also an indicator in the metrics file to identify the samples that have too many SNPs. A column in the metrics.tsv file, `Excluded_Sample`, indicates when a sample has been excluded from the snp matrix. This column is normally blank. See *Metrics*.

To exclude samples with excessive SNPs:

Grab the default configuration file:

```
cfsan_snp_pipeline data configurationFile
```

Edit `snppipeline.conf`, and change this setting:

```
MaxSnps=1000  # substitute your threshold value here, or -1 to disable this function
```

Then run the pipeline with the -c command line option:

```
cfsan_snp_pipeline run -c snppipeline.conf -s mySamplesDir myReference.fasta
```

See also *Configuration*.

## 4.14 Metrics

After creating the SNP matrix, the pipeline collects and tabulates metrics for all of the samples. The metrics are first collected in one file per sample in the sample directories. A subsequent step combines the metrics for all the samples together into a single tab-separated file with one row per sample and one column per metric. The tabulated metrics file is named metrics.tsv by default.

The metrics are:

| Metric | Description |
|---|---|
| Sample | The name of the directory containing the sample fastq files. |
| Fastq Files | Comma separated list of fastq file names in the sample directory. |
| Fastq File Size | The sum of the sizes of the fastq files. This will be the compressed size if the files are compressed. |
| Machine | The sequencing instrument ID extracted from the compressed fastq.gz file header. If the fastq files are not compressed, the machine ID is not captured. |
| Flowcell | The flowcell used during the sequencing run, extracted from the compressed fastq.gz file header. If the fastq files are not compressed, the flowcell is not captured. |
| Number of Reads | The number of reads in the SAM file. When using paired fastq files, this number will be twice the number of reads reported by bowtie. |
| Duplicate Reads | The number of reads marked as duplicates. These reads are not included in the pileup and are not used to call snps. When a set of duplicate reads is found, only the highest-quality read in the set is retained. |
| Percent of Reads Mapped | The percentage of reference-aligned reads in the SAM file. |
| Percent Proper Pair | The percentage of all reads in the SAM file that are aligned to the reference in the proper orientation and within the expected paired-end distance. The Percent Proper Pair metric is less than 100% when there are discordant alignments, unpaired alignments, or reads that are not mapped at all. This metric reflects the state of the alignment immediately after the mapping, however, subsequent steps of the pipeline remove the reads with low mapping quality, which has the effect of increasing the percentage of proper pairs prior to calling snps. |
| Average Insert Size | The average insert size of mapped paired reads in the SAM file as reported by SAMtools stats. Discordant alignments will increase the average insert size because by definition, discordant alignments are not within the expected paired-end distance. |
| Average Pileup Depth | The average depth of coverage in the sample pileup file. This is calculated as the sum of the depth of the pileup across all pileup positions divided by the number of positions in the reference. |
| Phase1 SNPs | The number of phase 1 SNPs found for this sample. The count is computed as the number of SNP records in the VCF file generated by the phase 1 snp caller (VarScan). |
| Phase1 Preserved SNPs | The number of phase 1 SNPs found by VarScan and preserved by SNP Filtering. The count is computed as the number of SNP records in the preserved VCF file generated by the `filter_regions` command. |
| Phase2 SNPs | The number of phase 2 SNPs found for this sample. The count is computed as the number of SNP records in the VCF file generated by the consensus caller. |
| Phase2 Preserved SNPs | The number of phase 2 SNPs found for this sample and preserved by SNP Filtering. The count is computed as the number of SNP records in the preserved VCF file generated by the consensus caller. |
| Missing SNP Matrix Positions | The number of positions in the SNP matrix for which a consensus base could not be called for this sample. The inability to call a consensus base is caused by either a pileup file with no coverage at a SNP position, or by insufficient agreement among the pileup bases at the SNP position. The minimum fraction of reads that must agree at a position to make a consensus call is controlled by the `minConsFreq` parameter. |
| Missing Preserved SNP Matrix | The number of positions in the preserved SNP matrix for which a consensus base could not be called for this sample. The inability to call a consensus base is caused by either a pileup file with no coverage at a SNP position, or by insufficient agreement among the pileup bases at the SNP position. The minimum fraction of reads that must agree at a position to make a consensus call is controlled by the `minConsFreq` parameter. |

## 4.15 Error Handling

The SNP Pipeline detects errors during execution and prevents execution of subsequent steps when earlier steps fail. A summary of errors is written to the `error.log` file. Detailed error messages are found in the log files for each process. See *Logging*.

By default, the SNP Pipeline is configured to stop when execution errors occur. However, it is possible some errors may affect only individual samples and other samples can still be processed. If you want the pipeline to continue processing after an error affecting only a single sample has occurred, you can try disabling the `StopOnSampleError` configuration parameter (not recommended). See *Configuration*. When `StopOnSampleError` is `false` the pipeline will attempt to continue subsequent processing steps when an error does not affect all samples. Errors are logged in the `error.log` file regardless of how the `StopOnSampleError` parameter is configured. You should review the `error.log` after running the pipeline to see a summary of any errors detected during execution.

Note: currently, when using the Torque job queue manager, the pipeline will always stop on errors regardless of the `StopOnSampleError` parameter setting.

When errors stop the execution of the pipeline on a High Performance Computing cluster, other non-failing jobs in progress will continue until complete. However, subsequent job steps will not execute and instead will remain in the queue. On Grid Engine, the `qstat` command will show output like the following:

```
3038927 0.55167 mapReads   app_sdavis   Eqw   07/15/2017 16:50:03
3038928 0.00000 callSites  app_sdavis   hqw   07/15/2017 16:50:04
3038929 0.00000 filterRegi app_sdavis   hqw   07/15/2017 16:50:04
3038930 0.00000 mergeSites app_sdavis   hqw   07/15/2017 16:50:04
3038931 0.00000 callConsen app_sdavis   hqw   07/15/2017 16:50:04
3038932 0.00000 snpMatrix  app_sdavis   hqw   07/15/2017 16:50:04
3038933 0.00000 snpReferen app_sdavis   hqw   07/15/2017 16:50:04
3038934 0.00000 mergeVcfs  app_sdavis   hqw   07/15/2017 16:50:05
3038935 0.00000 distance   app_sdavis   hqw   07/15/2017 16:50:05
3038936 0.00000 collectMet app_sdavis   hqw   07/15/2017 16:50:05
3038937 0.00000 combineMet app_sdavis   hqw   07/15/2017 16:50:05
```

To clear the jobs from the queue on Grid Engine:

```
seq 3038927 3038937 | xargs qdel
```

CHAPTER 5

# Correct and Reproducible Results

It is our goal to make the SNP Pipeline results both fully reproducible and as correct as current scientific understanding allows. As part of this effort, we document here problems we have found that have affected correctness. We also detail how we have built this software so the results are as reproducible as possible. In addition, we are building, collecting, and collating data sets that we use to assess both correctness and reproducibility of our software. As a project that is made possible only by very recent developments in science and technology, our efforts to ensure correctness and reproducibility are an ongoing effort. The publications we have produced in an effort to ensure scientific correctness are listed as references at the bottom of this document. This document will continue to evolve as we improve our process and as scientific advances occur.

## 5.1 Reproducible Results

We have made the SNP Pipeline results fully reproducible – not just the final SNP matrix, but each intermediate file as well. Reproducible results help us test and debug the pipeline and also facilitate collaborative efforts between researchers.

### 5.1.1 Public Availability

The SNP Pipeline source code is available on GitHub so anyone can download our source code. The GitHub repository contains some data sets that can be used to reproduce selected results. We also provide information on how to obtain and verify other data sets that we have used. (These data sets are large, so we do not provide them directly.)

### 5.1.2 Version Control

We use git internally for code development to ensure that we have control over our source code and can identify which version of code was used to produce any particular result. We tag/version commits of the code that we consider production releases and use them for the majority of our internal analyses. We also release each of these tagged versions to GitHub and to the Python Package Index for easy installation.

### 5.1.3 Parameters

The SNP pipeline behavior depends on the setting of a number of parameters that determine the behavior of various software packages that the pipeline uses. These parameters affect both the correctness and reproducibility of results. We have set all the parameters so that the results are reproducible. This entails setting seeds for all random number dependent processes, as well as specific choices for other parameters that can affect such behavior as the order of the results. We discuss these aspects of ensuring reproducibility in more detail in other portions of this document.

The pipeline depends on some fairly complex software packages, and these packages have large numbers of parameters. As released, the pipeline does not specify values for every possible parameter, but only those we have found it useful to modify in our work. A configuration file used by the pipeline provides a record of the parameters used and also makes it possible to customize the behavior of the pipeline by adding or modifying parameters as needed. Those wishing to further modify the software behavior will have to adjust the code to meet their needs.

We recommend retaining the parameter values used for any important results, ideally in a script or configuration file that is under version control. The pipeline generates log files documenting each run by capturing software versions and parameters used for each run.

### 5.1.4 Concurrency

The SNP Pipeline takes advantage of multiple CPU cores to run portions of the processing in parallel. However, concurrency can lead to non-deterministic behavior and different results when the pipeline is run repeatedly. The pipeline addresses known concurrency issues with bowtie and samtools.

**Bowtie**

The SNP Pipeline uses multiple CPU cores during the bowtie alignment. Unless told otherwise, when bowtie runs multiple concurrent threads, it generates output records in the SAM file in non-deterministic order. The consequence of this is the SAM files and Pileup files can differ between runs. This may appear as two adjacent read-bases swapped in the pileup files.

To work around this problem, the pipeline uses the `--reorder` bowtie command line option. The reorder option causes bowtie to generate output records in the same order as the reads in the input file. This is discussed in the bowtie documentation here: http://bowtie-bio.sourceforge.net/bowtie2/manual.shtml#performance-options

**SAMtools**

The SNP Pipeline runs multiple samtools processes concurrently to generate pileups for each sample. When the samtools pileup process runs, it checks for the existence of the reference faidx file, `*.fai`. If the faidx file does not exist, samtools creates it automatically. However, multiple samtools mpileup processes can interfere with each other by attempting to create the file at the same time. This interference causes incorrect pipeline results.

To work around this problem, the pipeline explicitly creates the faidx file by running `samtools faidx` on the reference before running the mpileup processes. This prevents errors later when multiple samtools mpileup processes run concurrently.

### 5.1.5 Software Versions

Different versions of the software packages this pipeline uses can generate different results. This is important to be aware of if you end up comparing the results between runs. We share our observations from the versions of SAMtools and Bowtie that we have used below.

**Bowtie**

Different versions of Bowtie can generate different SAM files, which subsequently causes different pileups and different variant detection. For example, with our included data sets, Bowtie 2.1.0 and 2.2.2 produce functionally identical

SAM files when run on the Lambda Virus data set. However, the generated SAM files (and downstream results) are different when run on the Salmonella Agona data set.

**SAMtools**

SAMtools mpileup version 0.1.18 and version 0.1.19 differ in their default behavior. Version 0.1.19 can filter out bases with low quality, and by default, it excludes bases with quality score below 13 (95% accuracy). Version 0.1.18 does not have this capability, and thus different versions of SAMtools mpileup when run with the default parameters can produce different pileup files which can impact the snp list and snp matrix.

On one of our data sets with 116 samples, we observed these results:

- 36030 snps found when pileups generated with SAMtools 0.1.18
- 38154 snps found when pileups generated with SAMtools 0.1.19

## 5.2 Test Data Sets

We have created/curated a number of data sets for use in testing both the reproducibility and correctness of the pipeline. In the following sections we briefly describe these data sets.

### 5.2.1 Lambda Virus

This data set was built using the bowtie2 example, and intended to be a small test case and example that will run quickly and verify the basic functionality of the code.

### 5.2.2 Salmonella Agona

This data set was designed to contain realistic sequences, but not very many of them, so that it could be run in a reasonable amount of time. The data must be downloaded from the NCBI due to its large size. We provide a file of hashes that can easily be used to verify that the data downloaded matches the data originally used to produce our results. (Use sha256sum at the unix command line.)

### 5.2.3 Listeria monocytogenes

This is designed to be a realistic-sized data set based on an outbreak of L. m. in stone fruit. The data must be downloaded from the NCBI due to its large size. We provide a file of hashes that can easily be used to verify that the data downloaded matches the data originally used to produce our results. (Use sha256sum at the unix command line.)

### 5.2.4 Synthetic data sets

*Coming soon in a future release*

We are currently creating synthetic data sets based on simulating various evolutionary scenarios. The simulations are designed to be similar to what we would expect in the types of organisms we study (food-borne pathogens), with error structure appropriate for the platforms we use to do sequencing.

CHAPTER 6

# FAQ / Troubleshooting Guide

## 6.1 Installation

**Q: How can I avoid polluting my global python installation when installing the SNP pipeline?**

A: You can either use a python virtual environment or install into your user area. To use a python virtual environment, see the *Get Started!* section for developers who want to contribute. To install into your user area instead of installing into your global site packages, do this:

```
$ pip install --user snp-pipeline
```

**Q: The SNP Pipeline cannot find VarScan. How should I install it?**

A: Download the VarScan jar file from SourceForge. Put the jar file anywhere. You need read-access to the jar file, but not execute-access. Set the CLASSPATH environment variable to specify the path to the VarScan jar file. The CLASSPATH should include the filename, not just the directory. Define it like this in your .bashrc file:

```
export CLASSPATH=~/software/varscan.v2.3.9/VarScan.v2.3.9.jar:$CLASSPATH
```

**Q: How can I uninstall the SNP pipeline?**

A: If you installed with pip, you can uninstall from the command line:

```
$ pip uninstall snp-pipeline
```

**Q: How can I rollback to an older version of the SNP pipeline?**

A: You can revert to an older version with these commands:

```
$ pip uninstall snp-pipeline
$ pip install --user snp-pipeline==0.3.2  # substitute the version you want here
```

**Q: Is there a way to install a specific release of the SNP pipeline from the github repository?**

A: Yes, you can install a release from github with this command:

```
$ pip install --user https://github.com/CFSAN-Biostatistics/snp-pipeline/archive/v0.3.
↪2  # substitute the version you want here
```

## 6.2 Running the Pipeline

**Q: Nothing works.**

A: Make sure you have the proper dependencies on your path. Modify your path if necessary to include bowtie, samtools, and bcftools. See the question above about installing VarScan. See the *Installation* section of this documentation.

**Q: How can I verify the pipeline is installed and working properly?**

A: The SNP Pipeline includes sets of test data with result files. You can run the pipeline against the test data to verify correct results. Follow the lambda virus workflow steps here: *All-In-One Workflow - Lambda Virus*.

Upon successful completion of the pipeline, the snplist.txt file should have 166 entries. The SNP Matrix can be found in snpma.fasta. It should have the following contents:

```
>sample1
AGCACCGGGACCCACGGCGCACGCAAAGATCCGAATTGCAGGGCGTACCTGGACCCCGGT
GACGGGGGATCGGGGACTCTTGGTCGAGGAACTAAAACGAACATCCACGTTTTCATGGCG
ACTGCTTGCCAGGTGTCAGCACATTCCCTATATCGGTGGACACGTA
>sample2
GGCGCTAGGAGGCAAGCCTTGGTCGTGGTTAATAGTTACAAGGCGTGCGCGTACTGCCGT
CTCCTACTATCTCTGCCGCCTCTCCGCGATCCGGACCGCAACACCAACTCTCTGGTGGCA
TCCTCTGAATCGTCGTGAGCATCTCAATTATATATTCGTCCGCGCG
>sample3
GGCGCTAGGAGGTACGCTTCGCGTGTGGATCAGCGCTACGGTGCCTATGCGTGACCCGCG
GAACTGGGTTCGCGTAAGGCAGTTTCAGGTACGGCAACGTAGATCAAAGT-TAGAAACCA
TACTCGTAATCCGCCTGACGCTACTCATTATGTATGTGGACGCCTG
>sample4
GAGGTTAACTGGCTCACCTCGCGCGTGTAACAGAGTAATAGGTTGAACGCCTACCCTGGT
GACCTGGGACGGCGGACGCCTGTTCGAAGTAAGGAAACGATCCTAAGCGTCTTGATGGGA
TCCTATTAATCGGCGCGTGCATATTCATCGGACATGTCGAGGGGTG
```

Note: the expected pipeline results are also included in the distribution. To fetch the expected result files:

```
cfsan_snp_pipeline data lambdaVirusExpectedResults myDirectoryForExpectedResults
```

After verifying correct results on the lambda data set, you can follow the workflow steps for the agona data set, *All-In-One Workflow - Salmonella Agona* or the listeria data set, *All-In-One Workflow - Listeria monocytogenes*. To fetch the expected result files:

```
cfsan_snp_pipeline data agonaExpectedResults myDirectoryForExpectedResults
cfsan_snp_pipeline data listeriaExpectedResults myDirectoryForExpectedResults
```

**Q: My results for the included test data do not match the expected results. What is the cause?**

A: Different versions of the executable tools can generate different results. The test data was generated with these versions:

- bowtie2 2.3.4.1
- samtools 1.8
- varscan 2.3.9

**Q: How can I run the SNP Pipeline with a mix of paired and unpaired samples?**

A: This is handled automatically if you use the `run` command. If you are running the `map_reads` command, run the script once per sample with either 1 fastq file or 2 fastq files. For example:

```
cfsan_snp_pipeline map_reads  reference/NC_011149  samples/CFSAN000448/G0H235M04.RL10.
↪fastq
cfsan_snp_pipeline map_reads  reference/NC_011149  samples/CFSAN000449/G00JH2D03.RL11.
↪fastq
cfsan_snp_pipeline map_reads  reference/NC_011149  samples/CFSAN000450/HB4DJL101.RL1.
↪fastq
cfsan_snp_pipeline map_reads  reference/NC_011149  samples/ERR178930/ERR178930_1.
↪fastq  samples/ERR178930/ERR178930_2.fastq
cfsan_snp_pipeline map_reads  reference/NC_011149  samples/ERR178931/ERR178931_1.
↪fastq  samples/ERR178931/ERR178931_2.fastq
```

**Q: How can I re-run some of the SNP Pipeline processing steps when I see a message that the results are already freshly built?**

A: The SNP Pipeline detects freshly built result files and does not rebuild them. Result files are not rebuilt when the file timestamp is newer than all of the input files. To force a rebuild, specify the `-f` option on the command line of any of the tools. To re-run only some of the steps, you can either delete the output files for that step or touch the input files for that step. All subsequent processing steps will also be re-run since their results will be out-of-date.

**Q: How does the SNP Pipeline know which processing steps should be re-run after changing the configuration file?**

A: It doesn't. If you change the configuration file, you may want to re-run some parts of the pipeline. The SNP Pipeline does not detect which parameters have changed since the last run. You must manually intervene to cause the pipeline to re-run the impacted processing steps. See the question above for guidance.

**Q: What do the dashes ("-") in the snp matrix indicate?**

A: Gaps, "-", are either missing bases (indels) or cases where there is insufficient information to make a consensus call (coverage depth too low, or consensus base frequency too low).

**Q: Why are some snps missing from the snp matrix even when the snps were called by VarScan?**

A: Older versions of VarScan failed to generate the header section of some VCF files. This in turn, caused the SNP Pipeline to ignore the first snp in the VCF file. Upgrade to a newer version VarScan.

**Q: Why are there no optical duplicate reads and why am I seeing the warning message "Default READ_NAME_REGEX '<optimized capture of last three ':' separated fields as numeric values>' did not match read name"?**

A: First, this is not a serious problem – optical duplicate reads occur much less frequently than PCR amplification duplicates. This message appears in the log file when Picard MarkDuplicates cannot identify the tile number, x-position, and y-position in the read names in the BAM file. Without those data elements, optical duplicates cannot be identified. You will see this warning only once, but it's usually a problem for every read in the file. When downloading fastq files from NCBI with `fastq-dump`, you can specify the `--origfmt` option to format the read names in the original Illumina format when possible. It is not always possible because NCBI does not always store the original read names in the SRA database.

## 6.3 Performance

**Q: How can I limit the CPU resources consumed by the pipeline?**

A: By default, the pipeline will use all available CPU resources. You can limit the number of CPU cores the pipeline will use with the *MaxCpuCores* parameter in the configuration file. This works on your workstation and also on a high performance computing cluster. See also the questions below.

**Q: How can I control the number of CPU cores used by the bowtie2, smalt, samtools, and GATK?**

A: You can set the number of CPU cores per process with the *CpuCoresPerProcessOnHPC* and *CpuCoresPerProcessOnWorkstation* parameters. This is the recommended way to control CPU cores per process for all multi-threaded processes in the pipeline:

```
CpuCoresPerProcessOnHPC = 16
CpuCoresPerProcessOnWorkstation = 8
```

If you want to allocate a different number of CPU cores for different processes, you can customize each command. For example, you can set the number of bowtie threads with the `-p` option and the number of samtools threads with the `-@` option. Set the options either in the configuration file if you are using the `run` command, or in the environment variables if you are running the map_reads command directly.

```
Bowtie2Align_ExtraParams = "-p 20"
SamtoolsSort_ExtraParams = "-@ 15"
```

You cannot use more threads than the number of allowed CPU cores. For example, if you set bowtie to use 10 threads and you set MaxCpuCores to 8, bowtie will only get 8 threads, not 10.

**Q: How can I control the number of concurrent processes?**

A: You can set configuration parameters to indirectly control the number of processes. First, set the *MaxCpuCores* parameter. Then set the *CpuCoresPerProcessOnHPC* and *CpuCoresPerProcessOnWorkstation* parameters. The number of concurrently executing processes will be the total number of CPUs divided by the number of CPU cores per process.

Multiple processes are run concurrently, each with multiple threads. The number of concurrently executing processes depends on the *MaxCpuCores* parameter setting. For example if you set bowtie to use 10 CPU cores per process and set MaxCpuCores to 20, you will have 2 concurrent bowtie2 processes each with 10 CPU threads.

**Q: How can I control the amount of memory that is used by the Picard, GATK, and VarScan java virtual machines?**

A: The amount of memory used by the java VM can be set by using the `-Xmx` java VM option. Set the option either in the configuration file if you are using the `run` command, or in the VarscanJvm_ExtraParams environment variable if you are running the call_sites command directly. For example, to set maximum java heap size to 3000 MB:

```
PicardJvm_ExtraParams="-Xmx3000m"
GatkJvm_ExtraParams="-Xmx3000m"
VarscanJvm_ExtraParams="-Xmx3000m"
```

## 6.4 Developer Questions

**Q: What causes "ImportError: No module named sphinx_rtd_theme" when building the documentation?**

A: The documentation uses the *Read The Docs* theme. Install it like this:

```
$ pip install --user sphinx_rtd_theme
```

**Q: I installed sphinx_rtd_theme, but I still get error "ImportError: No module named sphinx_rtd_theme".**

A: Try running sphinx like this:

```
$ python /usr/bin/sphinx-build -b html  .  ./_build
```

**Q: I changed one of the shell scripts, but the changes are ignored.**

A: Reinstall the distribution. Do this:

```
$ python setup.py develop
```

CHAPTER 7

# Configuration

You can customize the behavior of the SNP Pipeline by configuring parameters. Each step in the pipeline has a corresponding parameter allowing you to set one or more options for each tool in the pipeline.

Parameters can be configured either in a configuration file if you are using the `run` command, or in environment variables.

The pipeline comes with a default configuration file. When you use the `run` command without specifying a configuration file, it automatically uses the default supplied configuration file.

To get a copy of the default configuration file, run the following command. This will create a file called `snppipeline.conf`:

```
cfsan_snp_pipeline data configurationFile
```

To customize the pipeline behavior, edit the configuration file and pass the file to the `run` command:

```
cfsan_snp_pipeline run -c snppipeline.conf ...
```

When the `run` command executes, it copies the configuration file to the log directory for the run, capturing the configuration used for each run.

If you decide not to use the `run` command, you can still customize the behavior of the pipeline tools, but you will need to set (and export) environment variables with the same names as the parameters in the configuration file.

The available configuration parameters are described below.

## 7.1 StopOnSampleError

Controls whether the pipeline exits upon detecting errors affecting only a single sample. The pipeline will always stop upon detecting global errors affecting all samples.

**Default**:

> When this parameter is not set to a value, the pipeline will stop upon detecting single sample errors. If you want the pipeline to continue, you must explicitly set this parameter false.

**Example**:

```
StopOnSampleError=false
```

## 7.2 MaxCpuCores

Controls the number of CPU cores used by the SNP Pipeline. This parameter controls CPU resources on both your workstation and on an HPC cluster. By limiting the number of CPU cores used, you are also limiting the number of concurrently executing processes.

**Default**:

When this parameter is not set to a value, the pipeline will launch multiple concurrent processes using all available CPU cores.

**Example**:

```
MaxCpuCores=2
```

## 7.3 CpuCoresPerProcessOnHPC

Controls how many CPU cores are concurrently used per process when running multi-threaded processes on Grid or Torque. This parameter affects bowtie2, smalt, samtools, and GATK. You can set this parameter in one place instead of setting special options in multiple places for bowtie, smalt, samtools, and GATK. The pipeline will automatically use this setting whenever processes use multiple CPU cores. You should set this parameter to the typical number of CPUs available on the compute nodes in your HPC cluster. The *MaxCpuCores* parameter takes precedence over CpuCoresPerProcessOnHPC. If you set CpuCoresPerProcessOnHPC higher than MaxCpuCores, you will only get MaxCpuCores threads per process.

**Default**:

Do not leave this parameter unset.

**Example**:

```
CpuCoresPerProcessOnHPC=16
```

## 7.4 CpuCoresPerProcessOnWorkstation

Controls how many CPU cores are concurrently used per process when running multi-threaded processes on your workstation. This parameter affects bowtie2, smalt, samtools, and GATK. You can set this parameter in one place instead of setting special options in multiple places for bowtie, smalt, samtools, and GATK. The pipeline will automatically use this setting whenever processes use multiple CPU cores. You should set this parameter to some even divisor of the number of CPUs on your workstation. When this parameter is less than the number of CPU cores in your workstation, multiple processes will be launched, each using the number of cores specified here. The *MaxCpuCores* parameter takes precedence over CpuCoresPerProcessOnWorkstation. If you set CpuCoresPerProcessOnWorkstation higher than MaxCpuCores, you will only get MaxCpuCores threads per process.

**Default**:

When this parameter is not set to a value, all CPU cores are consumed by one process, instead of splitting the CPU cores between multiple processes.

**Example**:

```
CpuCoresPerProcessOnWorkstation=8
```

## 7.5 MaxSnps

Controls the maximum number of snps allowed for each sample. Any sample with excessive snps exceeding this limit will be excluded from the snp list, snp matrix, and snpma.vcf file. When set to -1, this parameter is disabled.

**Default**:

Do not leave this parameter unset. To disable the excessive snp filtering and include all samples regardless of the number of snps, set the parameter to -1

**Example**:

```
MaxSnps=1000
```

## 7.6 SnpPipeline_Aligner

Controls which reference-based aligner is used to map reads to the reference genome. The choices are `bowtie2` or `smalt`.

**Default**:

When this parameter is not set to a value, the pipeline will use the bowtie2 aligner.

**Example**:

```
SnpPipeline_Aligner="smalt"
```

## 7.7 Bowtie2Build_ExtraParams

Specifies options passed to the bowtie2 indexer. Any of the bowtie2-build options can be specified.

**Default**: none

**Example**:

```
Bowtie2Build_ExtraParams="--offrate 3"
```

## 7.8 SmaltIndex_ExtraParams

Specifies options passed to the smalt indexer. Any of the smalt index options can be specified.

**Default**: none

**Example**:

```
SmaltIndex_ExtraParams="-k 20 -s 1"
```

## 7.9 SamtoolsFaidx_ExtraParams

Specifies options passed to the SAMtools faidx indexer. Any of the SAMtools faidx options can be specified.

**Default**: none

**Example**:

```
SamtoolsFaidx_ExtraParams=""
```

## 7.10 CreateSequenceDictionary_ExtraParams

Specifies options passed to the Picard CreateSequenceDictionary indexer. Any of the CreateSequenceDictionary options can be specified.

**Default**: none

**Example**:

```
CreateSequenceDictionary_ExtraParams=""
```

## 7.11 Bowtie2Align_ExtraParams

Specifies options passed to the bowtie2 aligner. Any of the bowtie2 aligner options can be specified.

**Default**:

If you do not specify the -p option, the CFSAN SNP Pipeline will automatically set the number of threads using the values *CpuCoresPerProcessOnHPC* and *CpuCoresPerProcessOnWorkstation*. To disable bowtie2 multithreading, specify "-p 1".

If Bowtie2Align_ExtraParams is not set to any value, the `--reorder` option is enabled by default. Any value, even a single space, will suppress this default option.

**Parameter Notes**:

**`-p threads`** Bowtie2 uses the specified number of parallel search threads. Setting this option is not recommended. The recommended way to control threads per process is by setting *CpuCoresPerProcessOnHPC* and *CpuCoresPerProcessOnWorkstation*.

**`--reorder`** Generate output records in the same order as the reads in the input file.

**`-X distance`** Maximum inter-mate fragment length for valid concordant paired-end alignments.

**Example**:

```
Bowtie2Align_ExtraParams="--reorder -p 16 -X 1000"
```

## 7.12 SmaltAlign_ExtraParams

Specifies options passed to the smalt mapper. Any of the smalt map options can be specified.

**Default**:

If you do not specify the -n option, the CFSAN SNP Pipeline will automatically set the number of threads using the values *CpuCoresPerProcessOnHPC* and *CpuCoresPerProcessOnWorkstation*. To disable smalt multithreading, specify "-n 1".

If SmaltAlign_ExtraParams is not set to any value, the `-O` option is enabled by default. Any value, even a single space, will suppress this default option.

**Parameter Notes**:

**`-n threads`** Number of parallel alignment threads. Setting this option is not recommended. The recommended way to control threads per process is by setting *CpuCoresPerProcessOnHPC* and *CpuCoresPerProcessOnWorkstation*.

**`-O`** Generate output records in the same order as the reads in the input file.

**`-i size`** Maximum insert size for paired-end reads.

**`-r seed`** Random number seed, if seed < 0 reads with multiple best mappings are reported as 'not mapped'.

**`-y fraction`** Filters output alignments by a threshold in the number of exactly matching nucleotides

**Example**:

```
SmaltAlign_ExtraParams="-O -i 1000 -r 1"
```

## 7.13 SamtoolsSamFilter_ExtraParams

Specifies options passed to the SAMtools view tool when filtering the SAM file. Any of the SAMtools view options can be specified.

**Default**:

If SamtoolsSamFilter_ExtraParams is not set, the "-F 4" option is enabled by default. Any value, even a single space, will suppress the -F option.

If you do not specify the -@ option, the CFSAN SNP Pipeline will set the number of threads using the values *CpuCoresPerProcessOnHPC* and *CpuCoresPerProcessOnWorkstation*.

**Parameter Notes**:

**`-F 4`** Exclude unmapped reads.

**`-q threshold`** Exclude reads with map quality below threshold.

**-@ threads** Number of parallel threads. Setting this option is not recommended. The recommended way to control threads per process is by setting *CpuCoresPerProcessOnHPC* and *CpuCoresPerProcessOnWorkstation*.

**Example**:

```
SamtoolsSamFilter_ExtraParams="-F 4 -q 30"
```

## 7.14 SamtoolsSort_ExtraParams

Specifies options passed to the SAMtools sort tool when sorting the BAM file. Any of the SAMtools sort options can be specified.

**Default**:

If you do not specify the -@ option, the CFSAN SNP Pipeline will set the number of threads using the values *CpuCoresPerProcessOnHPC* and *CpuCoresPerProcessOnWorkstation*.

**Parameter Notes**:

**-@ threads** Number of parallel threads. Setting this option is not recommended. The recommended way to control threads per process is by setting *CpuCoresPerProcessOnHPC* and *CpuCoresPerProcessOnWorkstation*.

**Example**:

```
SamtoolsSort_ExtraParams=""
```

## 7.15 SamtoolsIndex_ExtraParams

Specifies options passed to the SAMtools index tool when indexing the BAM file. Any of the SAMtools index options can be specified.

**Default**:

If you do not specify the -@ option, the CFSAN SNP Pipeline will set the number of threads using the values *CpuCoresPerProcessOnHPC* and *CpuCoresPerProcessOnWorkstation*.

**Parameter Notes**:

**-@ threads** SPECIAL NOTE: ordinarily, we recommended specifying the threads with the the *CpuCoresPerProcessOnHPC* and *CpuCoresPerProcessOnWorkstation* parameters. However, when we tested samtools index on a Lustre file system, we found it runs slower with multiple threads. For this reason, we recommend customizing this parameter for your environment.

**Example**:

```
SamtoolsIndex_ExtraParams="-@ 2"
```

## 7.16 RemoveDuplicateReads

Controls whether the pipeline removes duplicate reads prior to creating the pileup and calling snps.

**Default**:

When this parameter is not set to a value, the pipeline removes duplicate reads.

**Example**:

```
RemoveDuplicateReads=false
```

## 7.17 PicardJvm_ExtraParams

Specifies options passed to the Picard Java Virtual Machine. Any of the JVM options can be specified.

**Default**: None

**Parameter Notes**:

`-Xmx300m` : use 300 MB memory (modify as needed)

**Example**:

```
PicardJvm_ExtraParams="-Xmx300m"
```

## 7.18 PicardMarkDuplicates_ExtraParams

Specifies options passed to the Picard MarkDuplicates tool when removing duplicate reads.

**Default**: None

**Example**:

```
PicardMarkDuplicates_ExtraParams="DUPLICATE_SCORING_STRATEGY=TOTAL_MAPPED_REFERENCE_
↪LENGTH"
```

## 7.19 EnableLocalRealignment

Controls whether the pipeline realigns reads around indels prior to creating the pileup and calling snps.

**Default**:

When this parameter is not set to a value, the pipeline realigns reads around indels.

**Example**:

```
EnableLocalRealignment=false
```

## 7.20 GatkJvm_ExtraParams

Specifies options passed to the GATK Java Virtual Machine. Any of the JVM options can be specified.

**Default**: None

**Parameter Notes**:

`-Xmx3500m` : use 3500 MB memory (modify as needed)

**Example**:

```
GatkJvm_ExtraParams="-Xmx3500m"
```

## 7.21 RealignerTargetCreator_ExtraParams

Specifies options passed to the GATK RealignerTargetCreator tool.

**Default**:

If you do not specify the -nt option, the CFSAN SNP Pipeline will set the number of threads
using the values *CpuCoresPerProcessOnHPC* and *CpuCoresPerProcessOnWorkstation*.
To disable RealignerTargetCreator multithreading, specify "-nt 1".

**Parameter Notes**:

**`-nt threads`** Number of parallel threads. Setting this option is not recommended. The recommended way to control threads per process is by setting *CpuCoresPerProcessOnHPC* and *CpuCoresPerProcessOnWorkstation*.

**Example**:

```
RealignerTargetCreator_ExtraParams="--minReadsAtLocus 7"
```

## 7.22 IndelRealigner_ExtraParams

Specifies options passed to the GATK IndelRealigner tool.

**Default**: None

**Example**:

```
IndelRealigner_ExtraParams="--consensusDeterminationModel USE_READS"
```

## 7.23 SamtoolsMpileup_ExtraParams

Specifies options passed to the SAMtools mpileup tool. Any of the SAMtools mpileup options can be specified.

**Default**: None

**Parameter Notes**:

`-q` : minimum mapping quality for an alignment to be used
`-Q` : minimum base quality for a base to be considered
`-x` : disable read-pair overlap detection
`-A` : include alignments that are not proper-pairs

**Example**:

```
SamtoolsMpileup_ExtraParams="-q 0 -Q 13 -A"
```

## 7.24 VarscanMpileup2snp_ExtraParams

Specifies options passed to the Varscan mpileup2snp tool. Any of the Varscan mpileup2snp options can be specified.

**Default**: None

**Parameter Notes**:

`--min-avg-qual` : minimum base quality at a position to count a read
`--min-var-freq` : minimum variant allele frequency threshold
`--min-reads2` : minimum supporting reads at a position to call variants

**Example**:

```
VarscanMpileup2snp_ExtraParams="--min-avg-qual 15 --min-var-freq 0.90 --min-reads2 5"
```

## 7.25 VarscanJvm_ExtraParams

Specifies options passed to the Varscan Java Virtual Machine. Any of the JVM options can be specified.

**Default**: None

**Parameter Notes**:

`-Xmx300m` : use 300 MB memory (modify as needed)

**Example**:

```
VarscanJvm_ExtraParams="-Xmx300m"
```

## 7.26 FilterRegions_ExtraParams

Specifies options passed to the filter_regions command.

**Default**: None

**Parameter Notes**:

**--edge_length** The length of the edge regions in a contig, in which all SNPs will be removed.

**--window_size** The length of the window in which the number of SNPs should be no more than max_num_snp.

**--max_snp** The maximum number of SNPs allowed in a window.

**--out_group** Relative or absolute path to the file indicating outgroup samples, one sample ID per line.

**--mode all** Dense regions found in any sample are filtered from all of the samples.

**--mode each** Dense regions found in any sample are filtered from each sample independently.

You can filter snps more than once by specifying multiple window sizes and max snps. For example "–max_snp 3 2 –window_size 1000 100" will filter more than 3 snps in 1000 bases and also more than 2 snps in 100 bases.

**Example**:

```
FilterRegions_ExtraParams="--edge_length 500 --window_size 1000 125 15 --max_snp 3 2␣
↪1 --out_group /path/to/outgroupSamples.txt"
```

## 7.27 MergeSites_ExtraParams

Specifies options passed to the merge_sites command.

**Default**: None

**Example**:

```
MergeSites_ExtraParams="--verbose 1"
```

## 7.28 CallConsensus_ExtraParams

Specifies options passed to the call_consensus command.

**Default**: None

**Parameter Notes**:

**--minBaseQual** Mimimum base quality score to count a read. All other snp filters take effect after the low-quality reads are discarded.

**--minConsFreq** Consensus frequency. Mimimum fraction of high-quality reads supporting the consensus to make a call.

**--minConsDpth** Consensus depth. Minimum number of high-quality reads supporting the consensus to make a call. This impacts both variant calls and reference calls.

**--minConsStrdDpth** Consensus strand depth. Minimum number of high-quality reads supporting the consensus which must be present on both the forward and reverse strands to make a call

**--minConsStrdBias** Strand bias. Minimum fraction of the high-quality consensus-supporting reads which must be present on both the forward and reverse strands to make a call. The numerator of this fraction is the number of high-quality consensus-supporting reads on one strand. The denominator is the total number of high-quality consensus-supporting reads on both strands combined.

**--vcfFileName** VCF Output file name. If specified, a VCF file with this file name will be created in the same directory as the consensus fasta file for this sample.

**--vcfAllPos** Flag to cause VCF file generation at all positions, not just the snp positions. This has no effect on the consensus fasta file, it only affects the VCF file. This capability is intended primarily as a diagnostic tool and enabling this flag will greatly increase execution time.

**--vcfPreserveRefCase** Flag to cause the VCF file generator to emit each reference base in uppercase/lowercase as it appears in the reference sequence file. If not specified, the reference bases are emitted in uppercase.

**Example**:

```
CallConsensus_ExtraParams="--minBaseQual 15 --minConsDpth 3 --vcfFileName consensus.
↪vcf"
```

## 7.29 SnpMatrix_ExtraParams

Specifies options passed to the snp_matrix command.

**Default**: None

**Example**:

```
SnpMatrix_ExtraParams="--verbose 1"
```

## 7.30 SnpReference_ExtraParams

Specifies options passed to the snp_reference command.

**Default**: None

**Example**:

```
SnpReference_ExtraParams="--verbose 1"
```

## 7.31 MergeVcfs_ExtraParams

Specifies options passed to the merge_vcfs command.

**Default**: none

**Example**:

```
MergeVcfs_ExtraParams="-n sample.vcf"
```

## 7.32 BcftoolsMerge_ExtraParams

Specifies options passed to the bcftools merge tool.

**Default**:

When this parameter is not set to a value, the pipeline uses the settings: `--merge all --info-rules NS:sum`. Any value, even a single space, will suppress the default settings.

**Parameter Notes**:

**`--merge`**

**Controls the creation of multiallelic records.**

- none = no new multiallelics, output multiple records instead
- snps = allow multiallelic SNP records
- indels = allow multiallelic indel records
- both = both SNP and indel records can be multiallelic
- all = SNP records can be merged with indel records
- id = merge by ID

**`--filter-logic`**

**Controls the content of the filter data element.**

- x = set the output record filter to PASS if any of the inputs pass
- + = set the output record filter to PASS when all of the inputs pass

**`--info-rules`** Rules for merging INFO fields (scalars or vectors) or - to disable the default rules. METHOD is one of sum, avg, min, max, join. Default is DP:sum,DP4:sum if these fields exist in the input files. Fields with no specified rule will take the value from the first input file.

**Example**:

```
BcftoolsMerge_ExtraParams="--merge all --info-rules NS:sum"
```

## 7.33 CollectSampleMetrics_ExtraParams

Specifies options passed to the collect_metrics command.

**Default**: none

**Example**:

```
CollectSampleMetrics_ExtraParams="-v consensus.vcf"
```

## 7.34 CombineSampleMetrics_ExtraParams

Specifies options passed to the combine_metrics command.

**Default**: none

**Parameter Notes**:

`-s` : Emit column headings with spaces instead of underscores

**Example**:

```
CombineSampleMetrics_ExtraParams="-s"
```

## 7.35 Torque_StripJobArraySuffix

Controls stripping the suffix from the job id when specifying Torque job array dependencies. It may be necessary to change this parameter if the pipeline fails with an illegal qsub dependency error.

**Example**:

```
Torque_StripJobArraySuffix=false
```

## 7.36 GridEngine_StripJobArraySuffix

Controls stripping the suffix from the job id when specifying Grid Engine job array dependencies. It may be necessary to change this parameter if the pipeline fails with an illegal qsub dependency error.

**Example**:

```
GridEngine_StripJobArraySuffix=true
```

## 7.37 GridEngine_PEname

Specifies the name of the Grid Engine parallel environment. This is only needed when running the SNP Pipeline on a High Performance Computing cluster with the Grid Engine job manager. Contact your HPC system administrator to determine the name of your parallel environment. Note: the name of this parameter was PEname in releases prior to 0.4.0.

**Example**:

```
GridEngine_PEname="mpi"
```

## 7.38 Slurm_SbatchExtraParams

Specifies extra options passed to sbatch when running the SNP Pipeline on the SLURM job scheduler.

**Default**: None

**Example**:

```
Slurm_SbatchExtraParams="--time=12:00:00"
```

## 7.39 GridEngine_QsubExtraParams

Specifies extra options passed to qsub when running the SNP Pipeline on the Grid Engine job scheduler.

**Default**: None

**Example**:

```
GridEngine_QsubExtraParams="-q bigmem.q -l h_rt=12:00:00"
```

## 7.40 Torque_QsubExtraParams

Specifies extra options passed to qsub when running the SNP Pipeline on the Torque job scheduler.

**Default**: None

**Example**:

```
Torque_QsubExtraParams="-l pmem=16gb -l walltime=12:00:00"
```

CHAPTER 8

# Command Reference

## 8.1 cfsan_snp_pipeline

```
usage: cfsan_snp_pipeline [-h] [--version] subcommand        ...

The CFSAN SNP Pipeline is a collection of tools using reference-based
alignments to call SNPs for a set of samples.

positional arguments:
  subcommand
    run               This do-it-all script runs all the pipeline steps
    data              Copy included data to a specified directory
    index_ref         Index the reference
    map_reads         Align reads to the reference
    call_sites        Find the sites with high-confidence SNPs in a sample
    filter_regions    Remove abnormally dense SNPs from all samples
    merge_sites       Prepare the list of sites having SNPs
    call_consensus    Call the consensus base at high-confidence sites
    merge_vcfs        Merge the per-sample VCF files
    snp_matrix        Create a matrix of SNPs
    distance          Calculate the SNP distances between samples
    snp_reference     Write reference bases at SNP locations to a fasta file
    collect_metrics   Collect quality and SNP metrics for a sample
    combine_metrics   Merge the per-sample metrics
    purge             Purge the intermediate output files

optional arguments:
  -h, --help          show this help message and exit
  --version           show program's version number and exit
```

## 8.2 run

```
usage: cfsan_snp_pipeline run [-h] [-f] [-m MODE] [-c FILE]
                              [-Q grid|slurm|torque] [-o DIR]
                              (-s DIR | -S FILE) [--purge] [-v 0..5]
                              [--version]
                              referenceFile

Run the SNP Pipeline on a specified data set.

positional arguments:
  referenceFile         Relative or absolute path to the reference fasta file

optional arguments:
  -h, --help            show this help message and exit
  -f, --force           Force processing even when result files already exist
                        and are newer than inputs (default: False)
  -m MODE, --mirror MODE
                        Create a mirror copy of the reference directory and
                        all the sample directories.  Use this option to avoid
                        polluting the reference directory and sample
                        directories with the intermediate files generated by
                        the snp pipeline.  A "reference" subdirectory and a
                        "samples" subdirectory are created under the output
                        directory (see the -o option).  One directory per
                        sample is created under the "samples" directory.
                        Three suboptions allow a choice of how the reference
                        and samples are mirrored.
                          -m soft : creates soft links to the fasta and fastq
                                    files instead of copying
                          -m hard : creates hard links to the fasta and fastq
                                    files instead of copying
                          -m copy : copies the fasta and fastq files
                        (default: None)
  -c FILE, --conf FILE  Relative or absolute path to a configuration file for
                        overriding defaults and defining extra parameters for
                        the tools and scripts within the pipeline.
                        Note: A default parameter configuration file named
                              "snppipeline.conf" is used whenever the pipeline
                              is run without the -c option.  The configuration
                              file used for each run is copied into the log
                              directory, capturing the parameters used during
                              the run. (default: None)
  -Q grid|slurm|torque, --queue_mgr grid|slurm|torque
                        Job queue manager for remote parallel job execution in
                        an HPC environment. Currently "grid", "slurm", and
                        "torque" are supported. If not specified, the pipeline
                        will execute locally. (default: None)
  -o DIR, --out_dir DIR
                        Output directory for the result files. Additional
                        subdirectories are automatically created under the
                        output directory for logs files and the mirrored
                        samples and reference files (see the -m option). The
                        output directory will be created if it does not
                        already exist. If not specified, the output files are
                        written to the current working directory. If you re-
                        run the pipeline on previously processed samples, and
```

(continues on next page)

(continued from previous page)

```
                        specify a different output directory, the pipeline
                        will not rebuild everything unless you either force a
                        rebuild (see the -f option) or you request mirrored
                        inputs (see the -m option). (default: .)
  -s DIR, --samples_dir DIR
                        Relative or absolute path to the parent directory of
                        all the sample directories.  The -s option should be
                        used when all the sample directories are in
                        subdirectories immediately below a parent directory.
                        Note: You must specify either the -s or -S option, but
                              not both.
                        Note: The specified directory should contain only a
                              collection of sample directories, nothing else.
                        Note: Unless you request mirrored inputs, see the
                              -m option, additional files will be written to
                              each of the sample directories during the
                              execution of the SNP Pipeline (default: None)
  -S FILE, --samples_file FILE
                        Relative or absolute path to a file listing all of the
                        sample directories.  The -S option should be used when
                        the samples are not under a common parent directory.
                        Note: If you are not mirroring the samples (see the
                              -m option), you can improve parallel processing
                              performance by sorting the the list of
                              directories descending by size, largest first.
                              The -m option automatically generates a sorted
                              directory list.
                        Note: You must specify either the -s or -S option, but
                              not both.
                        Note: Unless you request mirrored inputs, see the
                              -m option, additional files will be written to
                              each of the sample directories during the
                              execution of the SNP Pipeline (default: None)
  --purge               Purge the intermediate output files (the entire
                        "samples" directory) when the pipeline completes
                        successfully. (default: False)
  -v 0..5, --verbose 0..5
                        Verbose message level (0=no info, 5=lots) (default: 1)
  --version             show program's version number and exit
```

## 8.3 data

```
usage: cfsan_snp_pipeline data [-h] [--version] whichData [destDirectory]

Copy data included with the CFSAN SNP Pipeline to a specified directory.

positional arguments:
  whichData          Which of the supplied data sets to copy.  The choices are:
                         lambdaVirusInputs          : Input reference and fastq files
                         lambdaVirusExpectedResults : Expected results files
                         agonaInputs                : Input reference file
                         agonaExpectedResults       : Expected results files
                         listeriaInputs             : Input reference file
                         listeriaExpectedResults    : Expected results files
```

(continues on next page)

(continued from previous page)

```
                        configurationFile          : File of parameters to customize␣
↪the
                                                     SNP pipeline

                    Note: the lambda virus data set is complete with input data and␣
↪expected
                    results.  The agona and listeria data sets have the reference␣
↪genome and
                    the expected results, but not the input fastq files, because the␣
↪files are
                    too large to include with the package.

  destDirectory     Destination directory into which the SNP pipeline data files␣
↪will be copied.
                    The data files are copied into the destination directory if the␣
↪directory
                    already exists.  Otherwise the destination directory is created␣
↪and the
                    data files are copied there.  (default: current directory)

optional arguments:
  -h, --help     show this help message and exit
  --version      show program's version number and exit

Example:
# create a new directory "testLambdaVirus" and copy the Lambda virus input data there
$ cfsan_snp_pipeline data lambdaVirusInputs testLambdaVirus
```

## 8.4 index_ref

```
usage: cfsan_snp_pipeline index_ref [-h] [-f] [-v 0..5] [--version]
                                    referenceFile

Index the reference genome for subsequent read mapping, and create the faidx
index file for subsequent pileups. The output is written to the reference
directory.

positional arguments:
  referenceFile         Relative or absolute path to the reference fasta file

optional arguments:
  -h, --help            show this help message and exit
  -f, --force           Force processing even when result files already exist
                        and are newer than inputs (default: False)
  -v 0..5, --verbose 0..5
                        Verbose message level (0=no info, 5=lots) (default: 1)
  --version             show program's version number and exit
```

## 8.5 map_reads

```
usage: cfsan_snp_pipeline map_reads [-h] [-f] [-v 0..5] [--threads INT]
                                    [--version]
                                    referenceFile sampleFastqFile1
                                    [sampleFastqFile2]

Align the sequence reads for a specified sample to a specified reference
genome. The reads are sorted, duplicates marked, and realigned around indels.
The output is written to the file "reads.sorted.deduped.indelrealigned.bam" in
the sample directory.

positional arguments:
  referenceFile         Relative or absolute path to the reference fasta file
  sampleFastqFile1      Relative or absolute path to the fastq file
  sampleFastqFile2      Optional relative or absolute path to the mate fastq
                        file, if paired (default: None)

optional arguments:
  -h, --help            show this help message and exit
  -f, --force           Force processing even when result files already exist
                        and are newer than inputs (default: False)
  -v 0..5, --verbose 0..5
                        Verbose message level (0=no info, 5=lots) (default: 1)
  --threads INT         Number of CPU cores to use (default: 8)
  --version             show program's version number and exit
```

## 8.6 call_sites

```
usage: cfsan_snp_pipeline call_sites [-h] [-f] [-v 0..5] [--version]
                                     referenceFile sampleDir

Find the sites with high-confidence SNPs in a sample.

positional arguments:
  referenceFile         Relative or absolute path to the reference fasta file
  sampleDir             Relative or absolute directory of the sample

optional arguments:
  -h, --help            show this help message and exit
  -f, --force           Force processing even when result files already exist
                        and are newer than inputs (default: False)
  -v 0..5, --verbose 0..5
                        Verbose message level (0=no info, 5=lots) (default: 1)
  --version             show program's version number and exit
```

## 8.7 filter_regions

```
usage: cfsan_snp_pipeline filter_regions [-h] [-f] [-n NAME] [-l EDGE_LENGTH]
                                         [-w [WINDOW_SIZE [WINDOW_SIZE ...]]]
                                         [-m [MAX_NUM_SNPs [MAX_NUM_SNPs ...]]]
```

(continues on next page)

(continued from previous page)

```
                                        [-g OUT_GROUP] [-M {all,each}]
                                        [-v 0..5] [--version]
                                        sampleDirsFile refFastaFile

Remove abnormally dense SNPs from the input VCF file, save the reserved SNPs
into a new VCF file, and save the removed SNPs into another VCF file.

positional arguments:
  sampleDirsFile        Relative or absolute path to file containing a list of
                        directories -- one per sample
  refFastaFile          Relative or absolute path to the reference fasta file

optional arguments:
  -h, --help            show this help message and exit
  -f, --force           Force processing even when result files already exist
                        and are newer than inputs (default: False)
  -n NAME, --vcfname NAME
                        File name of the input VCF files which must exist in
                        each of the sample directories (default: var.flt.vcf)
  -l EDGE_LENGTH, --edge_length EDGE_LENGTH
                        The length of the edge regions in a contig, in which
                        all SNPs will be removed. (default: 500)
  -w [WINDOW_SIZE [WINDOW_SIZE ...]], --window_size [WINDOW_SIZE [WINDOW_SIZE ...]]
                        The length of the window in which the number of SNPs
                        should be no more than max_num_snp. (default: [1000])
  -m [MAX_NUM_SNPs [MAX_NUM_SNPs ...]], --max_snp [MAX_NUM_SNPs [MAX_NUM_SNPs ...]]
                        The maximum number of SNPs allowed in a window.
                        (default: [3])
  -g OUT_GROUP, --out_group OUT_GROUP
                        Relative or absolute path to the file indicating
                        outgroup samples, one sample ID per line. (default:
                        None)
  -M {all,each}, --mode {all,each}
                        Control whether dense snp regions found in any sample
                        are filtered from all of the samples, or each sample
                        independently. (default: all)
  -v 0..5, --verbose 0..5
                        Verbose message level (0=no info, 5=lots) (default: 1)
  --version             show program's version number and exit

You can filter snps more than once by specifying multiple window sizes and max
snps. For example "-m 3 2 -w 1000 100" will filter more than 3 snps in 1000
bases and also more than 2 snps in 100 bases.
```

## 8.8 merge_sites

```
usage: cfsan_snp_pipeline merge_sites [-h] [-f] [-n NAME] [--maxsnps INT]
                                      [-o FILE] [-v 0..5] [--version]
                                      sampleDirsFile filteredSampleDirsFile

Combine the SNP positions across all samples into a single unified SNP list
file identifying the positions and sample names where SNPs were called.

positional arguments:
```

(continues on next page)

(continued from previous page)

```
  sampleDirsFile        Relative or absolute path to file containing a list of
                        directories -- one per sample
  filteredSampleDirsFile
                        Relative or absolute path to the output file that will
                        be created containing the filtered list of sample
                        directories -- one per sample. The samples in this
                        file are those without an excessive number of snps.
                        See the --maxsnps parameter.

optional arguments:
  -h, --help            show this help message and exit
  -f, --force           Force processing even when result file already exists
                        and is newer than inputs (default: False)
  -n NAME, --vcfname NAME
                        File name of the VCF files which must exist in each of
                        the sample directories (default: var.flt.vcf)
  --maxsnps INT         Exclude samples having more than this maximum allowed
                        number of SNPs. Set to -1 to disable this function.
                        (default: -1)
  -o FILE, --output FILE
                        Output file. Relative or absolute path to the SNP list
                        file (default: snplist.txt)
  -v 0..5, --verbose 0..5
                        Verbose message level (0=no info, 5=lots) (default: 1)
  --version             show program's version number and exit
```

## 8.9 call_consensus

```
usage: cfsan_snp_pipeline call_consensus [-h] [-f] [-l FILE] [-e FILE]
                                         [-o FILE] [-q INT] [-c FREQ] [-D INT]
                                         [-d INT] [-b FREQ]
                                         [--vcfFileName NAME]
                                         [--vcfRefName NAME] [--vcfAllPos]
                                         [--vcfPreserveRefCase]
                                         [--vcfFailedSnpGt {.,0,1}] [-v 0..5]
                                         [--version]
                                         allPileupFile

Call the consensus base for a sample at the specified positions where high-
confidence SNPs were previously called in any of the samples. Generates a
single-sequence fasta file with one base per specified position.

positional arguments:
  allPileupFile         Relative or absolute path to the genome-wide pileup
                        file for this sample.

optional arguments:
  -h, --help            show this help message and exit
  -f, --force           Force processing even when result file already exists
                        and is newer than inputs. (default: False)
  -l FILE, --snpListFile FILE
                        Relative or absolute path to the SNP list file across
                        all samples. (default: snplist.txt)
  -e FILE, --excludeFile FILE
```

(continues on next page)

(continued from previous page)

```
                      VCF file of positions to exclude. (default: None)
  -o FILE, --output FILE
                      Output file. Relative or absolute path to the
                      consensus fasta file for this sample. (default:
                      consensus.fasta)
  -q INT, --minBaseQual INT
                      Mimimum base quality score to count a read. All other
                      snp filters take effect after the low-quality reads
                      are discarded. (default: 0)
  -c FREQ, --minConsFreq FREQ
                      Consensus frequency. Mimimum fraction of high-quality
                      reads supporting the consensus to make a call.
                      (default: 0.6)
  -D INT, --minConsDpth INT
                      Consensus depth. Minimum number of high-quality reads
                      supporting the consensus to make a call. (default: 1)
  -d INT, --minConsStrdDpth INT
                      Consensus strand depth. Minimum number of high-quality
                      reads supporting the consensus which must be present
                      on both the forward and reverse strands to make a
                      call. (default: 0)
  -b FREQ, --minConsStrdBias FREQ
                      Strand bias. Minimum fraction of the high-quality
                      consensus-supporting reads which must be present on
                      both the forward and reverse strands to make a call.
                      The numerator of this fraction is the number of high-
                      quality consensus-supporting reads on one strand. The
                      denominator is the total number of high-quality
                      consensus-supporting reads on both strands combined.
                      (default: 0)
  --vcfFileName NAME  VCF Output file name. If specified, a VCF file with
                      this file name will be created in the same directory
                      as the consensus fasta file for this sample. (default:
                      None)
  --vcfRefName NAME   Name of the reference file. This is only used in the
                      generated VCF file header. (default: Unknown
                      reference)
  --vcfAllPos         Flag to cause VCF file generation at all positions,
                      not just the snp positions. This has no effect on the
                      consensus fasta file, it only affects the VCF file.
                      This capability is intended primarily as a diagnostic
                      tool and enabling this flag will greatly increase
                      execution time. (default: False)
  --vcfPreserveRefCase  Flag to cause the VCF file generator to emit each
                      reference base in uppercase/lowercase as it appears in
                      the reference sequence file. If not specified, the
                      reference base is emitted in uppercase. (default:
                      False)
  --vcfFailedSnpGt {.,0,1}
                      Controls the VCF file GT data element when a snp fails
                      filters. Possible values: .) The GT element will be a
                      dot, indicating unable to make a call. 0) The GT
                      element will be 0, indicating the reference base. 1)
                      The GT element will be the ALT index of the most
                      commonly occuring base, usually 1. (default: .)
  -v 0..5, --verbose 0..5
                      Verbose message level (0=no info, 5=lots) (default: 1)
```

(continues on next page)

(continued from previous page)

```
  --version             show program's version number and exit
```

## 8.10 merge_vcfs

```
usage: cfsan_snp_pipeline merge_vcfs [-h] [-f] [-n NAME] [-o FILE] [-v 0..5]
                                     [--version]
                                     sampleDirsFile

Merge the consensus vcf files from all samples into a single multi-vcf file
for all samples.

positional arguments:
  sampleDirsFile        Relative or absolute path to file containing a list of
                        directories -- one per sample

optional arguments:
  -h, --help            show this help message and exit
  -f, --force           Force processing even when result files already exist
                        and are newer than inputs (default: False)
  -n NAME, --vcfname NAME
                        File name of the vcf files which must exist in each of
                        the sample directories (default: consensus.vcf)
  -o FILE, --output FILE
                        Output file. Relative or absolute path to the merged
                        multi-vcf file (default: snpma.vcf)
  -v 0..5, --verbose 0..5
                        Verbose message level (0=no info, 5=lots) (default: 1)
  --version             show program's version number and exit
```

## 8.11 snp_matrix

```
usage: cfsan_snp_pipeline snp_matrix [-h] [-f] [-c NAME] [-o FILE] [-v 0..5]
                                     [--version]
                                     sampleDirsFile

Create the SNP matrix containing the consensus base for each of the samples at
the positions where high-confidence SNPs were found in any of the samples. The
matrix contains one row per sample and one column per SNP position. Non-SNP
positions are not included in the matrix. The matrix is formatted as a fasta
file, with each sequence (all of identical length) corresponding to the SNPs
in the correspondingly named sequence.

positional arguments:
  sampleDirsFile        Relative or absolute path to file containing a list of
                        directories -- one per sample

optional arguments:
  -h, --help            show this help message and exit
  -f, --force           Force processing even when result file already exists
                        and is newer than inputs (default: False)
  -c NAME, --consFileName NAME
```

(continues on next page)

(continued from previous page)

```
                        File name of the previously created consensus SNP call
                        file which must exist in each of the sample
                        directories (default: consensus.fasta)
  -o FILE, --output FILE
                        Output file. Relative or absolute path to the SNP
                        matrix file (default: snpma.fasta)
  -v 0..5, --verbose 0..5
                        Verbose message level (0=no info, 5=lots) (default: 1)
  --version             show program's version number and exit
```

## 8.12 distance

```
usage: cfsan_snp_pipeline distance [-h] [-f] [-p FILE] [-m FILE] [-v 0..5]
                                   [--version]
                                   snpMatrixFile

Calculate pairwise SNP distances from the multi-fasta SNP matrix. Generates a
file of pairwise distances and a file containing a matrix of distances.

positional arguments:
  snpMatrixFile         Relative or absolute path to the input multi-fasta SNP
                        matrix file.

optional arguments:
  -h, --help            show this help message and exit
  -f, --force           Force processing even when result file already exists
                        and is newer than inputs (default: False)
  -p FILE, --pairs FILE
                        Relative or absolute path to the pairwise distance
                        output file. (default: None)
  -m FILE, --matrix FILE
                        Relative or absolute path to the distance matrix
                        output file. (default: None)
  -v 0..5, --verbose 0..5
                        Verbose message level (0=no info, 5=lots) (default: 1)
  --version             show program's version number and exit
```

## 8.13 snp_reference

```
usage: cfsan_snp_pipeline snp_reference [-h] [-f] [-l FILE] [-o FILE]
                                        [-v 0..5] [--version]
                                        referenceFile

Write reference sequence bases at SNP locations to a fasta file.

positional arguments:
  referenceFile         Relative or absolute path to the reference bases file
                        in fasta format

optional arguments:
  -h, --help            show this help message and exit
```

(continues on next page)

(continued from previous page)

```
  -f, --force           Force processing even when result file already exists
                        and is newer than inputs (default: False)
  -l FILE, --snpListFile FILE
                        Relative or absolute path to the SNP list file
                        (default: snplist.txt)
  -o FILE, --output FILE
                        Output file. Relative or absolute path to the SNP
                        reference sequence file (default: referenceSNP.fasta)
  -v 0..5, --verbose 0..5
                        Verbose message level (0=no info, 5=lots) (default: 1)
  --version             show program's version number and exit
```

## 8.14 collect_metrics

```
usage: cfsan_snp_pipeline collect_metrics [-h] [-f] [-o FILE] [-m INT]
                                          [-c NAME] [-C NAME] [-v NAME]
                                          [-V NAME] [--verbose 0..5]
                                          [--version]
                                          sampleDir referenceFile

Collect alignment, coverage, and variant metrics for a single specified
sample.

positional arguments:
  sampleDir             Relative or absolute directory of the sample
  referenceFile         Relative or absolute path to the reference fasta file

optional arguments:
  -h, --help            show this help message and exit
  -f, --force           Force processing even when result files already exist
                        and are newer than inputs (default: False)
  -o FILE, --output FILE
                        Output file. Relative or absolute path to the metrics
                        file (default: metrics)
  -m INT, --maxsnps INT
                        Maximum allowed number of SNPs per sample (default:
                        -1)
  -c NAME               File name of the consensus fasta file which must exist
                        in the sample directory (default: consensus.fasta)
  -C NAME               File name of the consensus preserved fasta file which
                        must exist in the sample directory (default:
                        consensus_preserved.fasta)
  -v NAME               File name of the consensus vcf file which must exist
                        in the sample directory (default: consensus.vcf)
  -V NAME               File name of the consensus preserved vcf file which
                        must exist in the sample directory (default:
                        consensus_preserved.vcf)
  --verbose 0..5        Verbose message level (0=no info, 5=lots) (default: 1)
  --version             show program's version number and exit
```

## 8.15 combine_metrics

```
usage: cfsan_snp_pipeline combine_metrics [-h] [-f] [-n NAME] [-o FILE] [-s]
                                          [-v 0..5] [--version]
                                          sampleDirsFile

Combine the metrics from all samples into a single table of metrics for all
samples. The output is a tab-separated-values file with a row for each sample
and a column for each metric. Before running this command, the metrics for
each sample must be created with the collect_metrics command.

positional arguments:
  sampleDirsFile        Relative or absolute path to file containing a list of
                        directories -- one per sample

optional arguments:
  -h, --help            show this help message and exit
  -f, --force           Force processing even when result files already exist
                        and are newer than inputs (default: False)
  -n NAME, --metrics NAME
                        File name of the metrics files which must exist in
                        each of the sample directories. (default: metrics)
  -o FILE, --output FILE
                        Output file. Relative or absolute path to the combined
                        metrics file. (default: metrics.tsv)
  -s, --spaces          Emit column headings with spaces instead of
                        underscores (default: False)
  -v 0..5, --verbose 0..5
                        Verbose message level (0=no info, 5=lots) (default: 1)
  --version             show program's version number and exit
```

## 8.16 purge

```
usage: cfsan_snp_pipeline purge [-h] [-v 0..5] [--version] work_dir

Purge the intermediate output files in the "samples" directory upon successful
completion of a SNP Pipeline run if no errors are encountered.

positional arguments:
  work_dir              Path to the working directory containing the "samples"
                        directory to be recursively deleted

optional arguments:
  -h, --help            show this help message and exit
  -v 0..5, --verbose 0..5
                        Verbose message level (0=no info, 5=lots) (default: 1)
  --version             show program's version number and exit
```

CHAPTER 9

# Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

## 9.1 Types of Contributions

### 9.1.1 Report Bugs

Report bugs at https://github.com/CFSAN-Biostatistics/snp-pipeline/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

### 9.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

### 9.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

### 9.1.4 Write Documentation

SNP Pipeline could always use more documentation, whether as part of the official SNP Pipeline docs, in docstrings, or even on the web in blog posts, articles, and such.

### 9.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/CFSAN-Biostatistics/snp-pipeline/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.

## 9.2 Get Started!

Ready to contribute? Here's how to set up *snp-pipeline* for local development.

1. Fork the *snp-pipeline* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/snp-pipeline.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv snppipeline
$ cd snppipeline/
$ python setup.py develop
$ pip install sphinx_rtd_theme    # the documentation uses the ReadTheDocs theme
```

4. Run the unit tests on the supplied data set to verify your installation is working:

```
$ python setup.py test
```

5. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

6. When you're done making changes, check that your changes pass the tests, including testing other Python versions:

```
$ python setup.py test
$ tox
```

   To get tox, just pip install it into your virtualenv.

7. Run the regression tests:

```
$ test/regression_tests.sh
```

   To get shunit2, install from https://code.google.com/p/shunit2/

8. Update the documentation and review the changes locally with sphinx:

```
$ cd docs
$ sphinx-build -b html . ./_build
$ xdg-open _build/index.html
```

9. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

10. Submit a pull request through the GitHub website.

## 9.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, 3.4, 3.5, 3.6, and 3.7.

## 9.4 Tips

To run a subset of tests:

```
$ python -m unittest test.test_snppipeline
$ python -m unittest test.test_utils
```

CHAPTER 10

# Credits

## 10.1 CFSAN BioInformatics Team

- Errol Strain
- Yan Luo
- James Pettengill
- Hugh A. Rand
- Steve Davis
- Justin Payne
- Al Shpuntoff
- Joseph D. Baugher
- Yu Wang

## 10.2 External Contributors

None yet. Why not be the first?

CHAPTER 11

# History

## 11.1 2.2.1 (2020-09-23) - docs

- Fix the slurm job id in the log files.
- Fix the bug preventing the use of samtools 1.10 and higher.

## 11.2 2.2.0 (2020-05-19) - docs

- Add support for the SLURM job scheduler.

## 11.3 2.1.1 (2019-09-09) - docs

- Fix incompatibility involving temp file deletion when running the pipeline in Ubuntu VirtualBox on Windows.

## 11.4 2.1.0 (2019-07-01) - docs

- Added dependency on jobrunner and qarrayrun Python packages. These modules were formerly part of the CFSAN SNP Pipeline and have been released as separate packages.
- Dropped support for Python 2.6 and 3.3 as some of the external packages used by the pipeline have stopped supporting these.
- Added support for Python 3.6 and 3.7.
- VCF file generation: identify multiple alternate alleles per sample. The AD, ADF, and ADR depths are comma-separated and ordered in the same order as the alternate alleles.
- VCF file generation: identify the alternate allele and alternate allele depth regardless of whether the alternate allele is the consensus call.

- VCF file generation: fixed a software defect causing unreproducible VCF file results when a sample has multiple alleles with identical depth. Older versions of the SNP Pipeline sometimes identified different alternate alleles and different genotype on different runs with the same datasets. In some cases, older versions of the SNP Pipeline incorrectly called the reference allele when the snp call failed one or more filters. This change only affects VCF files. The snp matrix file was unaffected by the software defect.
- Added the capability to control whether dense regions are filtered from each sample independently or removed from all samples. The default behavior removes dense snp regions from all samples, matching the behavior of previous releases. See *SNP Filtering* for a detailed explanation.
- Added the `--purge` command line option to save space by removing the intermediate output files (the entire samples directory) upon successful completion of a run. See the *run* command reference.

## 11.5 2.0.2 (2018-09-20) - docs

- Fixed inconsistent indentation problem when running with Python 3.

## 11.6 2.0.1 (2018-09-20) - docs

- Fixed the Picard dependency check and improved the error messages when Picard or GATK dependency errors are reported.

## 11.7 2.0.0 (2018-07-20) - docs

**Changes Impacting Backwards Compatibility:**

- Moved the bam file creation functionality from the call_sites command to the map_reads command. The map_reads command takes fastq files as input (as before) and produces a finished bam file. The call_sites command now only creates a pileup and finds high-confidence variant sites.
- Added local realignment around indels. This is an optional step enabled by default. When enabled, local realignment creates a dependency on Picard and GATK. See *Local Realignment*.
- The pipeline uses multiple CPU threads when running SAMtools.
- SAMtools version 1.4 or higher is required. Older versions are not supported anymore.
- When running in an HPC environment, the pipeline is configured to use 20 CPU threads by default when mapping reads to the reference. You can change this by configuring the *CpuCoresPerProcessOnHPC* parameter.
- When running on a workstation, the pipeline is configured to use 8 CPU threads by default when mapping reads to the reference. You can change this by configuring the *CpuCoresPerProcessOnWorkstation* parameter.
- Changed the algorithm used to compute the Average Insert Size metric. The new algorithm uses SAMtools stats. In most cases the average insert size will be larger than before.
- Added a new metric called `Percent Proper Pair` which measures the percentage of all reads that are aligned to the reference in the proper orientation and within the expected paired-end distance. See *Metrics*.
- Fixed a bug causing non-compliance with the VCF version 4.1 specification. Prior to this fix, the VCF files sometimes wrongly contained − characters in the ALT field to represent deletions. VCF files are now generated according to the VCF version 4.2 specification, and deletions, if present, are represented with $\star$ characters.

**Other Changes:**

- Increased the configurable map quality threshold to exclude poorly mapped reads from analysis. See *SamtoolsSamFilter_ExtraParams*.
- Enhanced the SNP density filter to find dense regions of SNPs in multiple window sizes, each with a different number of allowed snps. See *FilterRegions_ExtraParams*.
- Changed the SAMtools mpileup options to include read alignments that are not properly paired. This change increases the number of detected snps. It also increases the effectiveness of the density filter by causing the removal of snps in dense regions that would not otherwise have been detected. See *SamtoolsMpileup_ExtraParams*.
- Increased the minimum required variant-supporting depth to call variants in phase 1 with VarScan. See *VarscanMpileup2snp_ExtraParams*.
- Increased the minimum required supporting depth to make a call in phase 2 with the consensus caller. See *CallConsensus_ExtraParams*.
- Added a `--threads` option to the map_reads script. This should only be used when building custom *Step-by-Step Workflows*.
- Updated the included datasets.
- Documented the tested versions of other software used by the pipeline. See *Installation*.
- Fixed compatibility with Python 3 when running with Grid Engine.
- Fixed merge_vcf failure when merging many VCF files. Increased the number of open file descriptors when needed.

## 11.8 1.0.1 (2017-09-28) - docs

- Updated usage instructions and expected result files for the Agona and Listeria datasets.

## 11.9 1.0.0 (2017-09-01) - docs

**Changes Impacting Backwards Compatibility:**

- Some configuration parameter names are changed. If you have been using a customized configuration file, you should begin using a new configuration file.
- Simplified the configuration of multi-threading. Replaced the configuration parameters MaxConcurrentCollectSampleMetrics, MaxConcurrentCallConsensus, and MaxConcurrentPrepSamples with a single new configuration parameter `MaxCpuCores`. See also *Performance*.
- The configuration file is not an executable bash script anymore. However, you can still substitute environment variables with the $VAR_NAME notation.
- Log file names are changed to harmonize with cfsan_snp_pipeline sub-command names.
- Grid and Torque job names are changed to match cfsan_snp_pipeline sub-command names.
- Deprecated all the old step-by-step scripts. These will be removed in a future release:
  - copy_snppipeline_data.py
  - prepReference.sh
  - alignSampleToReference.sh
  - prepSamples.sh

  - snp_filter.py
  - create_snp_list.py
  - call_consensus.py
  - mergeVcf.sh
  - create_snp_matrix.py
  - calculate_snp_distances.py
  - create_snp_reference_seq.py
  - collectSampleMetrics.sh
  - combineSampleMetrics.sh
- You may safely continue using `run_snp_pipeline.sh`. It is not deprecated and will not be removed in future releases.

**Other Changes:**

- Sweeping changes under the hood replacing the main run_snp_pipeline shell script with equivalent python code.
- Added a new helper utility, `qarrayrun` to simplify creating and running array jobs on Grid Engine or Torque.

## 11.10 0.8.2 (2017-06-09) - docs

- Fix samtools sort compatibility with samtools 0.1.19.

## 11.11 0.8.1 (2017-06-02) - docs

- Fix collect metrics failure when the fastq sequence id line is missing the machine or flowcell.

## 11.12 0.8.0 (2017-05-09) - docs

**Changes Impacting Backwards Compatibility:**

- Changed the collectSampleMetrics script to only accept input files in the sample directory, not in arbitrary locations.
- Changed the combineSampleMetrics script to write to metrics.tsv by default, not stdout.
- Leading zeros are stripped from Miseq flowcell identifiers in the metrics files.
- Added a dependency on Picard. You need to install Picard and change your CLASSPATH. See *Installation*.
- Removed the unused create_snp_pileup.py script.

**Bug Fixes:**

- Fixed the machine and flow cell reporting in the metrics file when the fastq read names are not in the original Illumina format.
- Fixed the calculation of average pileup depth in the metrics file. The formula previously included whitespace characters when calculating the length of the reference. The correct average depth is slightly deeper than previously calculated.

**Other Changes:**

- Sweeping changes under the hood replacing most shell scripts with equivalent python code. Repackaged the SNP Pipeline as a single executable with multiple sub-commands. The old scripts still exist for backwards compatibility and are rewritten as one-liners calling the new replacement commands. The main executable program is called *cfsan_snp_pipeline*.
- Added the capability to remove duplicate reads from BAM files prior to creating the pileup and calling snps. See *Duplicate Read Removal*. This change introduces a dependency on `Picard` and will require changing your CLASSPATH. See *Installation*. You can disable this step and keep the duplicate reads by configuring `RemoveDuplicateReads=false` in the configuration file.
- Added a new metric to count the number of duplicate reads in each sample.
- Capture read-group metadata in the SAM/BAM files during the read mapping step.
- Added a new configuration parameter, `BcftoolsMerge_ExtraParams` to allow customizing the snpma.vcf files created when merging the consensus VCF files. See *Configuration*.
- Removed the hard-coded wall-clock run-time limits for Torque and Sun Grid Engine jobs. Added default limits (12 hours) to the configuration file. You can change the runtime limits for all SNP Pipeline job steps with the `Torque_QsubExtraParams` or `GridEngine_QsubExtraParams` configuration parameters.
- Log the SNP Pipeline version in the header of all the log files.
- Changed the composition of the included Salmonella Agona data set to remove the excessively large sample ERR178930 and include a more diverse set of isolates from different geographic locations, different environmental sources, and different types of sequencing instruments.

## 11.13 0.7.0 (2016-11-30) - docs

- Added a new script to the pipeline: `snp_filter.py` removes snps from the ends of contigs and from regions where the snp density is abnormally high. This is an important change to the pipeline with additional processing and new output files. See *SNP Filtering*.
- NOTE: You cannot re-use an old configuration file when running SNP Pipeline version 0.7.0. You must create a new configuration file. See *Configuration*.
- Fixed compatibility with bcftools 1.2 and higher.
- Updated the result files in the included data sets with the results obtained using bcftools v1.3.1 and bowtie2 v2.2.9. Note: upgrading from bowtie 2.2.2 to 2.2.9 did not change the snp matrix on any of the included datasets.

## 11.14 0.6.1 (2016-05-23)

- Fixed compatibility with SAMtools 1.3.
- Changed the expected results data sets to match the results obtained using SAMtools version 1.3.1. Starting with SAMtools version 1.0, the samtools mpileup command implemented a feature to avoid double counting the read depth when the two ends of a paired-end read overlap. If you use this feature of SAMtools, the pileup depth will be noticably reduced. You can still count the overlapping read sections twice by using SAMtools v0.1.19 or by using a configuration file specifying the `-x` option in `SamtoolsMpileup_ExtraParams`.
- Removed the obsolete `reads.snp.pileup` files from the included results data sets.

## 11.15 0.6.0 (2016-04-11)

**Bug fixes:**

- Fixed compatibility with the newly released PyVCF 0.6.8 package.

**Other Changes:**

- A new configuration parameter, `MaxSnps`, controls the maximum number of snps allowed for each sample. Samples with excessive snps exceeding this limit are excluded from the snp list and snp matrix. See *Excessive SNPs*.
- A new column in the metrics.tsv file, `Excluded_Sample`, indicates when a sample has been excluded from the snp matrix. This column is normally blank.
- Added a new script to the pipeline: `calculate_snp_distances.py` computes the SNP distances between all pairs of samples. The SNP distances are written to the output files `snp_distance_pairwise.tsv` and `snp_distance_matrix.tsv`.
- Changed Sun Grid Engine execution to use array-slot dependency where possible, resulting in less idle time waiting for job steps to complete.

## 11.16 0.5.2 (2016-03-07)

**Bug fixes:**

- An empty snplist.txt file should not cause errors when creating the referenceSNP.fasta.
- An empty snplist.txt file should not preclude re-running subsequent steps of the pipeline.
- When configured to ignore single-sample errors, a missing var.flt.vcf file should not preclude rebuilding the snplist.txt file during a pipeline re-run.
- The metrics file did not properly capture the total number of snps per sample. See below for the details.

**Other Changes:**

- Capture separate metrics counting phase 1 snps (varscan) and phase 2 snps (consensus). Previously, the metrics only included phase 1 snps. This changes the contents of both the `metrics` and `metrics.tsv` files. The metrics file now contains a new tag `phase1Snps`. The old tag `snps` now correctly counts the total number of snps. The metrics.tsv file now has separate column headers for phase 1 snps and phase 2 snps. Any code that parses those files may need modifications to work properly with v0.5.2.
- Added the `Average Insert Size` metric.
- The metrics.tsv column headings now contain underscores instead of spaces for better interoperability with some downstream analysis tools. Column headings with spaces can be generated by specifing the combineSampleMetrics.sh `-s` option in the configuration file.
- Remove the dependence on the snp matrix when collecting sample metrics.
- Improve the speed of metrics calculation when rerunning the pipeline. Reuse the previously computed metrics when recalculation would be slow.

## 11.17 0.5.1 (2016-02-19)

**Bug fixes:**

- Do not shutdown the pipeline when the generated snplist is empty when there are no snps.
- Do not attempt to merge VCF files when there are fewer than two VCF files to merge.

**Other Changes:**

- Added the `vcfFailedSnpGt` option to the call_consensus.py script to control how the VCF file GT data element is emitted when the snp is failed because of depth, allele frequency, or some other filter. If not specified, the GT element will contain a dot. Prior to this release, the behavior was to emit the ALT allele index. The old behavior can be retained by setting `--vcfFailedSnpGt 1`
- Changed the setup to require PyVCF version 0.6.7 or higher. It will automatically upgrade if necessary.
- Added error checking after running SamTools and VarScan to detect missing, empty, or erroneous output files.

## 11.18 0.5.0 (2016-01-19)

**Bug fixes:**

- Changed VCF file generator to not emit multiple alleles when the reference base is lowercase.

**Other Changes:**

- Trap errors, shutdown the pipeline, and prevent execution of subsequent steps when earlier processing steps fail. A summary of errors is written to the `error.log` file. See *Error Handling*.
- Check for the necessary software tools (bowtie, samtools, etc.) on the path at the start of each pipeline run.
- Check for missing or empty input files at the start of each processing step.
- Added two new parameters, `GridEngine_QsubExtraParams` and `Torque_QsubExtraParams`, to the configuration file to pass options to qsub when running the SNP Pipeline on an HPC computing cluster. Among other things, you can control which queue the snp-pipeline will use when executing on an HPC with multiple queues. See *Configuration*.
- Removed the "job." prefix to shorten job names when running on an HPC.
- Changed the vcf file generator to emit reference bases in uppercase. Added the `vcfPreserveRefCase` flag to the call_consensus.py script to cause the vcf file generator to emit each reference base in uppercase/lowercase as it appears in the original reference sequence file. If not specified, the reference bases are emitted in uppercase. Prior to this release, the behavior was to always preserve the original case.
- Added support for Python 3.3, 3.4, 3.5.
- Implemented a regression test suite for the bash shell scripts, using the shUnit2 package.

## 11.19 0.4.1 (2015-10-30)

**Bug fixes:**

- Fixed a Python 2.6 incompatibility with the new consensus caller.

**Other Changes:**

- Added Tox support for automatically testing installation and execution with multiple Python versions.

## 11.20 0.4.0 (2015-10-22)

**Bug fixes:**

- When run on Grid Engine with the default settings, bowtie2 was consuming all available CPU cores per node while scheduled with Grid to use only 8 cores. On a lightly loaded cluster, this bug made the pipeline run faster, but when the cluster was full or nearly full, it would cause contention for available CPU resources and cause jobs to run more slowly. Changed to use only 8 CPU cores by default.
- The consensus snp caller miscounted the number of reference bases when the pileup record contained the ^ symbol marking the start of a read segment followed by a dot or comma. In this situation, the dot or comma should not be counted as reference bases.

**Other Changes:**

- Added support for the Smalt aligner. You can choose either bowtie2 or smalt in the configuration file. A new parameter in the configuration file, `SnpPipeline_Aligner`, selects the aligner to use. Two additional configuration parameters, `SmaltIndex_ExtraParams` and `SmaltAlign_ExtraParams` can be configured with any Smalt command line options. See *Tool Selection*. The default aligner is still bowtie2.
- Split the create_snp_matrix.py script into two pieces. The new script, call_consensus.py, is a redesigned consensus caller which is run in parallel to call snps for multiple samples concurrently. The create_snp_matrix.py script simply merges the consensus calls for all samples into a multi-fasta file.
- The new consensus caller has the following adjustable parameters. See the *call_consensus* command reference.
  - `minBaseQual` : Mimimum base quality score to count a read.
  - `minConsFreq` : Minimum consensus frequency.
  - `minConsStrdDpth` : Minimum consensus-supporting strand depth.
  - `minConsStrdBias`: Strand bias.
- Added the capability to generate VCF files. By default, a file named consensus.vcf is generated by the consensus caller for each sample, and the merged multi-sample VCF file is called snpma.vcf. This capability introduces a new dependency on bgzip, tabix, and bcftools. You can disable VCF file generation by removing the `--vcfFileName` option in the configuration file. Also, be aware the contents of the VCF files may change in future versions of the SNP Pipeline.
- Added configuration parameters `Torque_StripJobArraySuffix` and `GridEngine_StripJobArraySuffix` to improve compatibility with some HPC environments where array job id suffix stripping is incompatible with qsub.
- Renamed the configuration parameter `PEname` to `GridEngine_PEname`.

## 11.21 0.3.4 (2015-06-25)

**Bug fixes:**

- The referenceSNP.fasta file was missing newlines between sequences when the reference fasta file contained multiple sequences. In addition, each sequence was written as a single long string of characters. Changed to emit a valid fasta file. Updated the expected result files for the datasets included with the distribution accordingly.
- Changed the run_snp_pipeline.sh script to allow blank lines in the file of sample directories when called with the -S option.
- Changed the run_snp_pipeline.sh script to allow trailing slashes in the file of sample directories when called with the -S option.

- Do not print system environment information when the user only requests command line help.
- Fixed the broken pypi downloads per month badge on the readme page.

**Other Changes:**

- Changed the default configuration file to specify the `-X 1000` option to the bowtie2 aligner. This parameter is the maximum inter-mate distance (as measured from the furthest extremes of the mates) for valid concordant paired-end alignments. Previously this value was not explicitly set and defaulted to 500. As a result of this change, the generated SAM files may have a different number of mapped reads, the pileup files may have different depth, and the number of snps called may change.
- We now recommend using VarScan version 2.3.9 or later. We discoved VarScan v2.3.6 was occasionally omitting the header section of the generated VCF files. This in turn, caused the SNP Pipeline to miss the first snp in the VCF file. This is not a SNP Pipeline code change, only a documentation and procedural change.
- Updated the result files in the included data sets with the results obtained using VarScan v2.3.9 and the Bowtie -X 1000 option.
- Log the Java classpath to help determine which version of VarScan is executed.
- Changed the python unit tests to execute the non-python processes in a temporary directory instead of assuming the processes were already run in the test directory.

## 11.22 0.3.3 (2015-04-14)

**Bug fixes:**

- Improve HPC qsub submission speed throttling to avoid errors with the HPC job scheduler when submitting large and small jobs. Dynamically adjust the delays between HPC array job submission so small datasets have small delays and large datasets have large delays between qsub submissions.
- Process the sample directories in order by size, largest first, considering only the size of fastq files and ignoring all other files. Previously non-fastq files were affecting the processing order.
- Fixed divide-by-zero error in create_snp_matrix when no snps are detected.
- Don't skip the last sample when run_snp_pipeline is started with the -S option and the file of sample directories is not terminated with a newline.
- Gracefully exit run_snp_pipeline with error messages when run with -S option and any of the sample directories in the sample directory file is missing, empty, or does not contain fastq files.
- Gracefully exit run_snp_pipeline with an error message when run with -s option and the samples directory is empty or contains no subdirectories with fastq files.
- Fixed the sun grid engine "undefined" task id reported in non-array job log files.

**Other Changes:**

- Sample Metrics. The pipeline generates a table of sample metrics capturing various alignment, coverage, and snp statistics per sample. See *Metrics*.
- Explicitly expose the `minConsFreq` parameter in the supplied default configuration file to make it easier to adjust.
- Updated the FAQ with instructions to install to an older version.

## 11.23 0.3.2 (2015-01-14)

**Bug fixes:**

- Fixed (again) a Python 2.6 incompatibility with formatting syntax when printing the available RAM. This affected the shell scripts (prepReference.sh, alignSampleToReference.sh, prepSamples.sh).
- Improved installation in a Python 2.6 environment. Added several Python packages to the automatic setup script.

**Other Changes:**

- Added support for the Grid Engine job queue manager. See *High Performance Computing*.
- Added a configurable parameter, `minConsFreq`, to the create_snp_matrix.py script. This parameter specifies the mimimum fraction of reads that must agree at a position to make a consensus call. Prior to version 0.3.2, the snp pipeline required that a majority (more than half) of the reads must agree to make a snp call. In version 0.3.2, the default behavior requires at least 60% of reads must agree to make a consensus call.
- Changed the included snp matrix files for the agona and listeria data sets to match the new results obtained by setting minConsFreq=0.6. The lambda virus results were not impacted by this change.
- Revised the Installation instructions with more detailed step-by-step procedures.
- Added a Dockerfile for automated docker builds. This feature is still experimental.

## 11.24 0.3.1 (2014-10-27)

**Bug fixes:**

- Fixed a Python 2.6 incompatibility with formatting syntax when printing the available RAM. Also added the Python version to the log files.

## 11.25 0.3.0 (2014-10-22)

**Bug fixes:**

- Fixed some Mac OSX incompatibilities.
- Fixed a bug in copy_snppipeline_data.py that caused copy failure when the destination directory did not exist.
- Fixed alignSampleToReference.sh to properly handle unpaired gzipped fastq files.

**Installation Changes:**

- There is a new dependency on the python psutil package. When you install the SNP Pipeline, pip will attempt to install the psutil package automatically. If it fails, you may need to manually install the python-dev package. In Ubuntu, `sudo apt-get install python-dev`

**Other Changes:**

*Note a possible loss of backward compatibilty for existing workflows using alignSampleToReference.sh and prepSamples.sh*

- All-in-one script: Added a new script, run_snp_pipeline.sh, to run the entire pipeline either on a workstation or on a High Performance Computing cluster with the Torque job queue manager. See *All-In-One SNP Pipeline Script*.
- Logging: The run_snp_pipeline.sh script adds consistent logging functionality for workstation and HPC runs. The logs for each pipeline run are stored in a time-stamped directory under the output directory. See *Logging*.

- Timestamp checking: Changed the python scripts (create_snp_list.py, create_snp_pileup.py, create_snp_matrix.py, create_snp_reference.py) to skip processing steps when result files already exist and are newer than the input files. If you modify an upstream file, any dependent downstream files will be rebuilt. You can force processing regardless of file timestamps with the `-f` option. Similar functionality for the shell scripts was previously implemented in release 0.2.0.
- Mirrored input files: The run_snp_pipeline.sh script has the capability to make a mirrored copy of the input reference and samples to avoid polluting a clean repository. You have the choice to create copies, soft links, or hard links. See *Mirrored Inputs*.
- Configuration file: Added the capability to customize the behavior of the SNP Pipeline by specifying parameters either in a configuration file, or in environment variables. You can create a configuration file with default values pre-set by executing `copy_snppipeline_data.py configurationFile` from the command line. Pass the configuration file to the run_snp_pipeline.sh script with the `-c` option. Alternatively, environment variables matching the names of the parameters in the configuration file can be manually set (be sure to export the variables). When the run_snp_pipeline.sh script is run, it copies the configuration file for the run into the log directory for the run. See *Configuration*.
- Removed the `-p INT` command line option, to specify the number of cpu cores, from the alignSampleToReference.sh script. You can now control the number of cpu cores used by bowtie2 with the `-p INT` option either in the configuration file when running run_snp_pipeline.sh, or in the `Bowtie2Align_ExtraParams` environment variable when running alignSampleToReference.sh directly. If not specified, it defaults to 8 cpu cores on a HPC cluster, or all cpu cores on a workstation.
- Removed the `--min-var-freq 0.90` varscan mpileup2snp option from the prepSamples.sh script. This parameter is now specified in the `VarscanMpileup2snp_ExtraParams` environment variable or in the configuration file.
- Listeria monocytogenes data set: Added a Listeria monocytogenes data set. Updated the usage instructions, illustrating how to download the Listeria samples from NCBI and how to run the SNP Pipeline on the Listeria data set. The distribution includes the expected result files for the Listeria data set. Note that due to the large file sizes, the Listeria expected results data set does not contain all the intermediate output files.
- Added a command reference page to the documentation. See *Command Reference*.

## 11.26 0.2.1 (2014-09-24)

**Bug fixes:**

- Version 0.2.0 was missing the Agona data files in the Python distribution. The GitHub repo was fine. The missing files only impacted PyPi. Add the Agona data files to the Python distribution file list.

## 11.27 0.2.0 (2014-09-17)

**Changes Impacting Results:**

- Previously, the pipeline executed SAMtools mpileup twice – the first pileup across the whole genome, and the second pileup restricted to those positions where snps were identified by varscan in *any* of the samples. This release removes the second SAMtools pileup, and generates the snp pileup file by simply extracting a subset of the pileup records from the genome-wide pileup at the positions where variants were found in *any* sample. The consequence of this change is faster run times, but also an improvement to the results – there will be fewer missing values in the snp matrix.
- Changed the the supplied lambda virus expected results data set to match the results obtained with the pipeline enhancements in this release and now using SAMtools version 0.1.19. SAMtools mpileup version

0.1.19 excludes read bases with low quality. As a reminder, the expected results files are fetched with the copy_snppipeline_data.py script.

- Removed the "<unknown description>" from the snp matrix fasta file.

**Other Changes:**

*Note the loss of backward compatibilty for existing workflows using prepReference.sh, alignSampleToReference.sh, prepSamples.sh, create_snp_matrix.py*

- Split the create_snp_matrix script into 4 smaller scripts to simplify the code and improve performance when processing many samples in parallel. Refer to the *Usage* section for the revised step-by-step usage instructions. The rewritten python scripts emit their version number, arguments, run timestamps, and other diagnostic information to stdout.
- Changed the default name of the reads.pileup file to reads.snp.pileup. You can override this on the command line of the create_snp_pileup.py script.
- Added the referenceSNP.fasta file to the supplied lambda virus expected results data set.
- Updated the usage instructions, illustrating how to download the Agona samples from NCBI and how to run the SNP Pipeline on the Agona data set.
- Updated the supplied expected result files for the Agona data set. Note that due to the large file sizes, the Agona expected results data set does not contain all the intermediate output files.
- Improved the online help (usage) for all scripts.
- The copy_snppipeline_data.py script handles existing destination directories more sensibly now. The example data is copied into the destination directory if the directory already exists. Otherwise the destination directory is created and the example data files are copied there.
- Changed the alignSampleToReference.sh script to specify the number of CPU cores with the -p flag, rather than a positional argument. By default, all CPU cores are utilized during the alignment.
- Changed the shell scripts (prepReference.sh, alignSampleToReference.sh, prepSamples.sh) to expect the full file name of the reference including the fasta extension, if any.
- Changed the shell scripts (prepReference.sh, alignSampleToReference.sh, prepSamples.sh) to skip processing steps when result files already exist and are newer than the input files. If you modify an upstream file, any dependent downstream files will be rebuilt. You can force processing regardless of file timestamps with the `-f` option.
- Changed the name of the sorted bam file to reads.sorted.bam.
- Changed the general-case usage instructions to handle a variety of fastq file extensions (*.fastq* and *.fq*).

## 11.28 0.1.1 (2014-07-28)

**Bug fixes:**

- The snp list, snp matrix, and referenceSNP files were incorrectly sorted by position alphabetically, not numerically.
- The SNP Pipeline produced slightly different pileups each time we ran the pipeline. Often we noticed two adjacent read-bases swapped in the pileup files. This was caused by utilizing multiple CPU cores during the bowtie alignment. The output records in the SAM file were written in non-deterministic order when bowtie ran with multiple concurrent threads. Fixed by adding the `--reorder` option to the bowtie alignment command line.

- The snp list was written to the wrong file path when the main working directory was not specified with a trailing slash.

**Other Changes:**

*Note the loss of backward compatibilty for existing workflows using prepSamples.sh*

- Moved the bowtie alignment to a new script, alignSampleToReference.sh, for better control of CPU core utilization when running in HPC environment.
- Changed the prepSamples.sh calling convention to take the sample directory, not the sample files.
- prepSamples.sh uses the CLASSPATH environment variable to locate VarScan.jar.
- Changed prepReference.sh to run `samtools faidx` on the reference. This prevents errors later when multiple samtools mpileup processes run concurrently. When the faidx file does not already exist, multiple samtools mpileup processes could interfere with each other by attempting to create it at the same time.
- Added the intermediate lambda virus result files (*.sam, *.pileup, *.vcf) to the distribution to help test the installation and functionality.
- Changed the usage instructions to make use of all CPU cores.
- Log the executed commands (bowtie, samtools, varscan) with all options to stdout.

## 11.29 0.1.0 (2014-07-03)

- Basic functionality implemented.
- Lambda virus tests created and pass.
- S. Agona tests created – UNDER DEVELOPMENT
- Installs properly from PyPI.
- Documentation available at ReadTheDocs.

CHAPTER 12

# Indices and tables

- genindex
- modindex
- search