

TB Variable Site Alignment Pipeline

James Abbott (j.abbottimperial.ac.uk)

July 11, 2017

Contents

1	Introduction	1
1.1	Prerequisites	2
1.1.1	Software packages	2
1.1.2	Installing the pipeline	2
1.1.3	Perl modules	2
1.1.4	Configuring the software	3
2	Running the Pipeline	3
2.1	Preparing to run the pipeline	3
2.1.1	Input Sequences	3
2.1.2	Reference genome	4
2.2	Hyper-variable regions	4
2.3	Running per-sample analysis	4
2.3.1	Cluster submission	4
2.3.2	Analysing a single sample	5
2.3.3	Identifying failed tasks	5
2.3.4	Identifying outliers	5
2.3.5	Generating a variable site alignment	5
3	Pipeline Workflow	6
3.1	QC	6
3.2	Read trimming	6
3.3	Read alignment	6
3.4	Marking duplicates	6
3.5	Local realignment round indels	6
3.6	SNV calling	7
3.7	Identifying outliers	7
3.8	Generating variable-site alignment	8
4	Additional scripts	8
4.0.1	download_ena_project	8

1 Introduction

This document describes a pipeline for generating variable-site alignments (VSA) of populations of *Mycobacterium tuberculosis* isolates from paired-end Illumina WGS data. This has been produced on behalf of Caroline Colijn, Dept of Mathematics, Imperial College London. The pipeline carries out data QC and read-trimming/adaptor removal, followed by alignment of the sequence reads to a reference TB strain using BWA. Duplicate reads are marked, and local realignment of reads around potential indel sites is then carried out. SNVs are then identified using FreeBayes. An alignment consisting of only the variable sites in the population is then created excluding hyper-variable regions.

The pipeline has been designed to run on a cluster (cx1) under a batch queuing system (PBS Pro). Moving the software to run under a different environment will require modification of PBS directives etc. accordingly.

Package	Tested Version	Availability
bwa	0.7.15	http://bio-bwa.sourceforge.net
cutadapt	1.10	https://github.com/marcelm/cutadapt
fastqc	0.11.2	https://www.bioinformatics.babraham.ac.uk/projects/fastqc
freebayes	1.1.0	https://github.com/ekg/freebayes
gatk	3.6	https://software.broadinstitute.org/gatk
picard	2.6.0	https://broadinstitute.github.io/picard
sambamba	0.6.5	http://lomoreiter.github.io/sambamba
samtools	1.3.1	http://www.htslib.org
trim.galore	0.4.1	https://www.bioinformatics.babraham.ac.uk/projects/trim_galore/
vt	0.5.77	http://genome.sph.umich.edu/wiki/Vt

Table 1: Prerequisite software packages used by pipeline.

Module	Core?	Notes
Archive::Zip	No	Only required for download scripts
BioPerl	No	
Cwd	Yes	
Digest::MD5::File	No	Only required for download scripts
File::Basename	Yes	
File::Find::Rule	No	
File::Path	Yes	
FindBin	Yes	
Getopt::Long	Yes	
IPC::Cmd	Yes	
IPC::Open3	Yes	
LWP::UserAgent	No	Only required for download scripts
Pod::Usage	Yes	
XML::XPath	No	Only required for download scripts
XML::XPath::Parser	No	Only required for download scripts

Table 2: Required Perl modules.

Note that this pipeline has been produced specifically for TB isolates, considering the low recombination rates present in this organism, consequently is not necessarily generally applicable to other species.

1.1 Prerequisites

1.1.1 Software packages

A number of software packages are necessary to run this pipeline. These should be installed so that their binaries are available on the default path in the environment the software will be executed under. If run on the cx1 cluster, the appropriate environment modules will be automatically loaded at runtime so no action is required to add the binaries to the path. The required packages are listed in table 1. Other versions of these packages may well work, however the results of using an untested version can not be predicted.

1.1.2 Installing the pipeline

The pipeline is available in the github repository located at <https://github.com/ImperialCollegeLondon/TB-VSA-Pipeline>. The latest release can be obtained as a gzip compressed tar file from <https://github.com/ImperialCollegeLondon/TB-VSA-Pipeline/releases>. Once downloaded, the contents of this can be extracted using the command:

```
tar xzvf downloaded_file.tar.gz
```

Alternately, the latest updates to the software can be obtained directly from the git repository by running:

```
git clone https://github.com/ImperialCollegeLondon/TB-VSA-Pipeline
```

This will result in the creation of a new directory named 'TB-VSA-Pipeline'.

1.1.3 Perl modules

Certain perl modules are used by the pipeline, as indicated in table 2. Those which are not generally installed along with perl will be downloaded and installed into the software installation directory when the `setup.sh` script is executed. Should you wish to use other versions of these modules already installed on the system, it will be necessary to update the 'use lib' statements in the scripts to include the path to the libraries you wish to use.

1.1.4 Configuring the software

Once the software is downloaded, it is necessary to run the `setup.sh` script contained in the software installation directory. This script modifies the script templates in the `src` directory to set:

1. the path to the preferred version of perl.
2. the number of parallel threads to run. This defaults to the number of available cores on the system where installation is taking place.
3. the name of a PBSPro queue to submit jobs to (optional). The software can be run manually without using PBSPro, but will require each sample analysis to be run independently.
4. module load commands to make the necessary software prerequisites available on the default system path (only if installed on the ICL HPC cx1 cluster)

Default values can be selected by pressing `enter` without inputting any data. An example of running the `setup.sh` is shown below:

```
[jamesa@wssb-james TB-VSA_Pipeline]$ ./setup.sh
Enter path to perl: [/usr/bin/perl]
If this software is to be run on a PBSpro cluster requiring job submission to a particular queue,
enter the desired queue name (enter to skip): med-bio
Enter number of parallel threads to use: [4] 20

Installing Archive::Zip...
Installing Digest::MD5::File...
Installing LWP::UserAgent...
Installing XML::XPath...
Installing XML::XPath::Parser...

Updating scripts...
```

Once the script has completed, appropriately configured versions of the scripts will be available in the `bin` directory of the software installation directory.

To make the scripts available directly on the path of your system (so the scripts can be executed by typing their name without having to include the full path to the script), it is necessary to modify the `.bashrc` file located in your home directory with the inclusion of the following line:

```
export PATH=/path/to/installation/directory:${PATH}
```

where `/path/to/installation/directory` is replaced with the path to the location where the software is installed. It will be necessary to logout and login again for this change to take effect.

2 Running the Pipeline

The pipeline has been designed to be as easy as possible to run, and consists of two major stages: Carrying out per-sample analysis and variant calling, followed by the creation of the variable site alignment from the outputs of the per-sample analysis. A separate instance of the per-sample analysis is executed on multiple cluster nodes in parallel to minimise runtime, while the generation of the variable site alignment is carried out by a single process on one node.

2.1 Preparing to run the pipeline

The following data are required to run the pipeline:

1. A pair of gzip compressed fastq files for each isolate.
2. A fasta-formatted genome sequence of the reference isolate
3. A BED file describing hyper-variable regions of the reference genome to exclude from the VSA.

2.1.1 Input Sequences

A new directory should be created for the input sequences for each pipeline run. This should contain a separate directory for each sample, named with the sample name, and contain the fastq files for the sample, named `[sample]_1.fastq.gz` and `[sample]_2.fastq.gz`. No other files/directories should be placed in the input directory since a list of samples to be analysed will be determined by the software from a listing of the contents of this directory.

AL123456.3	3736984	3738438
AL123456.3	1541994	1542980
AL123456.3	2430159	2431199
AL123456.3	2784657	2785697
AL123456.3	3120566	3121552

Figure 1: Example of BED format file describing regions to be excluded from analysis

2.1.2 Reference genome

The sequence of the isolate to be used as a reference should be available as a fasta format file. An appropriately formatted file is available in the data directory of the pipeline installation for the H37Rv isolate.

2.2 Hyper-variable regions

A list of genomic regions to exclude from the analysis (i.e. highly variable regions which do not aid phylogenetic analysis) can optionally be provided as a BED format file. The BED format is a tab-delimited file format using one line per feature, with 3-12 columns used to describe genomic locations. The pipeline only requires the first three columns, which describe the reference identifier, the start co-ordinate of the region and the end co-ordinate. Co-ordinates are 0-based, meaning the first base of the reference sequence is numbered base 0. An example of part of valid BED file is shown in figure 1.

An example BED file for the H37Rv isolate is provided in the data directory of the pipeline installation.

2.3 Running per-sample analysis

2.3.1 Cluster submission

The pipeline is designed to be run on a cluster running the PBSPro job scheduling software, with specific configurations defined for the Imperial College HPC cx1 cluster. Modification to use different cluster configurations/middleware will require the lines beginning #PBS in the `tb_pipeline_run` and `build_vsa_run` scripts to be updated according to the desired cluster configuration/queueing system. Additionally the `qsub` commands in `submit_tb_run` and `submit_build_vsa` will need to be updated to suit the syntax of the required queue submission command.

A single script (`submit_tb_run`) needs to be run to carry out job submission of the per-sample analysis for an entire dataset. Basic usage information can be obtained by running the script without any arguments:

```
[jamesa@wssb-james colijn]$ submit_tb_run
Usage: submit_tb_run -i /path/to/input/dir -o /path/to/output/dir -r /path/to/reference
```

The script requires three arguments:

- **-i:** Path to input directory. The fully-qualified path to the directory described above containing a subdirectory of compressed fastq files for each sample.
- **-o:** Path to output directory. The fully-qualified path to a directory for storing intermediate run files and outputs in per-sample subdirectories. This directory will be created automatically if it doesn't exist. **N.B. The directory name will be used as a project name when reporting outputs**
- **-r:** Path to reference. The fully qualified path to the fasta-formatted genome sequence of the reference isolate.

The script will determine the number of samples being analysed by obtaining a directory listing of the specified input directory, then submit an array job to the cluster for the relevant number of tasks, with one task per sample. The progress of these jobs can be monitored using the `qstat -f` command i.e.

```
-bash-4.2$ qstat -f
Job Id: 4830274[.].cx1b
  Job_Name = analyse_tb_sample
  Job_Owner = jamesa@login-2-internal.cx1.hpc.ic.ac.uk
[truncated for clarity]
  array_state_count = Queued:8 Running:0 Exiting:0 Expired:0
  array_indices_submitted = 1-8
  array_indices_remaining = 1-8
  project = _pbs_project_default
```

This example `qstat -f` output shows an array of 8 tasks which have been submitted, all of which are queued pending execution. Each task will execute a single instance of the `tb_pipeline_run` script, which is a queue execution wrapper around `analyse_tb_sample`.

2.3.2 Analysing a single sample

Individual sample analysis can be running the `analyse_tb_sample` command directly e.g for running outside a cluster environment. It's usage is as follows:

```
analyse_tb_sample --input input_directory --output output_directory \
  --reference /path/to/reference.fasta [--sample sample_name]
```

- **--input:** Path to input directory. The fully-qualified path to the directory described above containing a subdirectory of compressed fastq files for each sample.
- **--output:** Path to output directory. The fully-qualified path to a directory for storing intermediate run files and outputs in per-sample subdirectories. This directory will be created automatically if it doesn't exist.
- **--reference:** Path to reference. The fully qualified path to the fasta-formatted genome sequence of the reference isolate.
- **--sample:** Sample name to analyse (must match directory name containing sample's reads in the input directory)

Note that the script will refuse to run if an output directory already exists for the defined sample, consequently it will be necessary to remove existing output directories or specific a different output directory should it be necessary to rerun the analysis of any samples.

2.3.3 Identifying failed tasks

There is a possibility that individual sample analysis jobs may fail, for example if they exceed the job limits defined by the submission script i.e. memory requirement, or runtime. Following the completion of all the scheduled tasks, it is possible to identify any which have failed using the `find_failed_tasks` script.

```
find_failed_tasks --input input_directory --output output_directory [--clean]
```

- **--input:** Path to input directory specified during job submission
- **--output:** Path to output directory specified during job submission
- **--clean** (optional): Remove output directories of failed jobs

When run, this script will report any samples which have failed the analysis run. The standard output/standard error of any failed tasks should be inspected to identify the cause of the failure.

The script can also be run with the `clean` flag which will remove the output directory of any failed samples. Once the cause of the job failures has been addressed, `submit_tb_run` should be rerun as before. Any samples which have existing output directories will be skipped, while those which are missing from previously failed jobs will be rerun.

This process should be repeated until all samples have successfully completed analysis.

2.3.4 Identifying outliers

Outliers can be automatically identified using the `identify_outliers` script. Any samples identified as being overly divergent from the majority of the data will have their sample ids written to an 'outliers.txt' file in the analysis directory. This will be read by subsequent pipeline stages and any samples listed in this file excluded from further analysis.

The `identify_outliers` script requires two arguments:

```
identify_outliers -i /path/to/analysis/dir -r /path/to/reference/seq
```

- **-i:** Path to analysis directory (output directory specified during previous job submission)
- **-r:** Path to fasta-format reference sequence

2.3.5 Generating a variable site alignment

Once all the tasks have successfully completed, the `build_vsa` script can be used to generate a variable site alignment from the samples. This can be run interactively if resources allow, or submitted to a cluster using the `submit_build_vsa` script. Note that when run on the Imperial HPC systems, the job should be submitted to cx1's queues.

```
submit_build_vsa -d /path/to/analysis/dir -r /path/to/reference \
-e /path/to/exclude.bed -o /path/to/outliers.txt
```

The following arguments should be passed to the `submit_build_vsa` script:

- **-d**: Path to analysis directory (output directory specified during job submission)
- **-r**: Path to fasta formatted reference sequence
- **-e**: Path to bed format file describing genomic regions to exclude from VSA
- **-o**: Path to text file listing outliers, one sample id per line

The resulting variable site alignment will be written to the output directory as a fasta-formatted alignment, along with an index file indicating the genomic locus represented by each position of the alignment.

3 Pipeline Workflow

The pipeline consists of a series of operations. Many of the tools and algorithms used can be readily replaced with alternatives if desired, or runtime parameters altered by editing the command lines defined in the `analyse_tb_sample` script.

3.1 QC

An assessment of sequence quality is initially carried out using FastQC. This carries out a separate quality assessment on each of read-pairs taking account of a considerable number of metrics. A rating of PASS, WARN or FAIL is assigned for each metric. These ratings are somewhat arbitrary distinctions but for the most part hold true. It is also important to understand the basis of each metric and what may cause abnormal results. For example kmer content and overrepresented sequences may well result from adapter contaminations, which will be handled automatically by the pipeline.

Text format summaries are generated for each fastq file and can be found in the per-sample output directory name `samplename_fastqc.summary.txt`. Should further investigation of the results be required, a `fastqc` subdirectory is present in each sample output directory which contains HTML format outputs for each fastq file which can be viewed with a web browser. Documentation on interpreting the various quality statistics is available from <https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%20Analysis%20Modules/>.

3.2 Read trimming

Reads are next trimmed using `trim_galore` to remove residual adapter sequences and bases of low quality. Bases with quality scores below 20 are removed from the 3' end of the reads, as well as sequences matching standard Illumina adapters. Any reads which are shorter than 70bp following quality trimming are discarded.

3.3 Read alignment

Reads are aligned to the reference genome using the BWA MEM algorithm, using default parameters, and the outputs converted to bam format using `sambamba`. Read group information is then added to the bam files to ensure compatibility with downstream tools such as GATK which require bam files to have read groups assigned. This process results in a sorted, indexed bam file which includes a RG header line describing the read group, and an associated RG tag on each read.

3.4 Marking duplicates

Duplicate reads should occur at a very low frequency at the depths of sequencing typically employed, and add no meaningful information. PCR over-amplification can result in over-representation of particular reads which can influence variant calling accuracy. These duplicate reads therefore need to be identified and marked in the bam files to ensure they are not included in downstream analysis. Duplicates are identified using Picard's `MarkDuplicates` tool, and the resulting bam file indexed.

3.5 Local realignment round indels

Each read-pair is aligned independently, resulting in the identification of the optimal alignment for each individual read pair. When reads span indel loci, the best alignment of each read-pair may differ between read-pairs resulting in some 'left-justified' alignments, and some which are 'right-justified' (see figure 2). As well as not correctly representing the indel locus, these alignments also introduce a number of false-positive SNV loci in the region surrounding the indel. The process of local indel realignment identifies potential indel loci and carries out a realignment of the reads

ref:	ttataaaac----	aattaagt	ttataaaac----	aattaagt
sample:	ttataaaacAAATaattaagt		ttataaaacAAATaattaagt	
	-----		-----	
read1:	ttataaaac	aaAtaa	ttataaaacAAATaa	
read2:	ttataaaac	aaAtaaTt	ttataaaacAAATaatt	
read3:	ttataaaacAAATaattaagt		ttataaaacAAATaattaagt	
read4:	CaaaT	aattaagt	cAAATaattaagt	
read5:	aaT	aattaagt	AAATaattaagt	
read6:	T	aattaagt	Taattaagt	

Figure 2: Example of misalignment surround indel locus, both before (left) and after (right) local indel realignment. Variant bases are indicated in upper case

spanning this region, only carries out the alignment in the context of all the reads to produce a best alignment of all the reads together. This not only results in the indel being correctly represented in the alignments, but also removes the spurious SNV calls.

Some variant callers carry out this process themselves, but to allow the variant caller in the pipeline to be readily substituted a standalone realignment task is carried out to allow the use of variant callers which do not take indel loci into account in this manner. This is carried out using the GATK IndelRealigner tool.

3.6 SNV calling

SNVs are identified using FreeBayes, a bayesian variant caller which allows ploidy to be taken into account when identifying variants. The current iteration of the pipeline has been developed purely with SNVs in mind, so FreeBayes is run so as not to attempt identification of MNPs, INDELs or complex variants (which are combinations of other variant types). Standard quality filters are used, requiring a minimum mapping quality of 30 and minimum base quality of 20. The minimum alternate fraction is also defined as 0.8, which will only reports on variants identified in $\geq 80\%$ of the reads at a locus. This helps reduce false-positives which is appropriate for sequence from a clonal isolate but will result in missed variant calls in clinical samples containing co-infecting isolates.

The parameters used for FreeBayes variant calling are as follows:

- **-p 1**: Ploidy = 1 (haploid)
- **-i** : no INDELs
- **-X** : no MNPs
- **-u** : no complex
- **-j** : use mapping quality
- **-O** : standard quality filters - min mapping quality 30; min base quality 20; min supporting allele qsum 0; genotype variant threshold 0
- **-F** : min alternate fraction 0.8

3.7 Identifying outliers

Some samples may not have issues during sequencing, or simply be overly divergent from the reference sequence to be sensible included in a variable site alignment. These can be automatically identified using the `find_outliers` script. This consolidates per-sample statistics from output files in each samples output directory into a single file of samples statistics for the whole project, then calls the 'outlier_plots.R' script to identify outliers and produce plots illustrating how these compare to the distribution of the full set of samples.

Samples are identified as outliers if

- have less than 20x genomic coverage
- have a mapping rate below 90
- have an unusually high number of SNVs identified (≥ 3 the interquartile range)

The distribution of read-count/SNVs and sample/mapping-rate are plotted in PDF files (named `projectname_snv_dist.pdf` and `projectname_mapping_rate.pdf`), with identified outliers highlighted and labelled with the sample ID. Outliers are also written to a `projectname_outliers.txt` file in the analysis directory, which is read during generation of the variable-site alignment and samples listed in it excluded from the analysis. Sample IDs can be added to or removed from this file manually as required to add or remove outliers from the dataset.

3.8 Generating variable-site alignment

The variable-site alignment is essentially a fasta-format multiple sequence alignment which excludes non-variant bases. An array (@variants) is used to track which loci are variant, and is initialised with one element for each base of the reference, each element being set to '0'. The VCF output files from the variant calling are then sequentially processed, and variant loci in each sample parsed and stored in a hash (sample_vars), keyed on the sample name and containing an array reference to a list of variant bases identified for the locus. For each variant identified at a locus, the value of the variants array representing the locus is incremented.

Once all VCF files have been parsed, the variants array is processed incrementally. When a variant locus is identified, each of the samples in the sample_vars hash is examined and if a variant locus has been identified in that sample, then the variant base is included in the alignment for that sample, otherwise the corresponding reference allele is used.

Following processing of all variant loci, the alignment is written, including the loci from the reference sequence as the first line of the alignment. An index is also created mapping the base number of the alignment to the corresponding location on the reference genome.

N.B. At present only a single variant base is included for each sample in the output alignment. This needs to be updated to produce the relevant IUPAC ambiguity code in the event that multiple variant bases are identified i.e in a non-clonal sample.

4 Additional scripts

The bin directory of the software installation includes a number of additional scripts which may be helpful in using the pipeline.

4.0.1 download_ena_project

The download_ena_project script is used to download all the sequence reads for a given ENA project, which typically consists of hundreds of files which otherwise require downloading independently. It is necessary to provide the script with the ID of the ENA project (also referred to as an ENA Study), and the path to the directory where the data should be downloaded. The downloaded data will be arranged with the sequence read files for each sample in a separate subdirectory, named with the sample identifier, as required by the sample analysis process. An example of downloading a project is show below:

```
download_ena_project --acc ERP004677 --output reads/PRJEB5280
```