Bacteria Variant Calling:

Author: **Noah A. Legall**, [noahaus@uga.edu](mailto:noahaus@uga.edu), **Liliana C. M. Salvador** PhD

# The commands and rationale behind the Salvador lab in house script to perform variant calling on bacteria isolates.

Link to script: <https://github.com/noahaus/bacteria_SNP>

## Current functionality

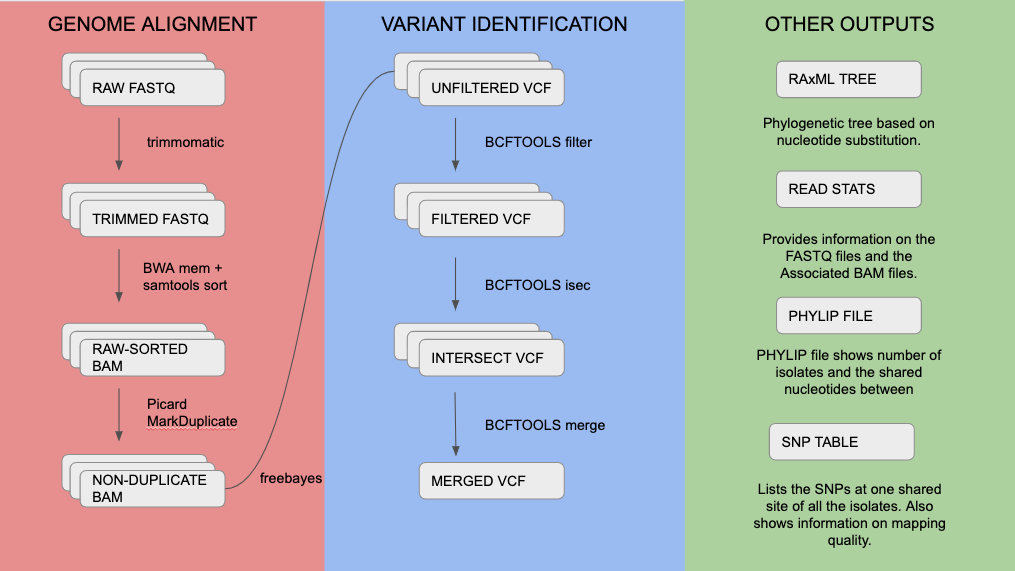
* Pair read variant calling
* RAxML tree generation
* Alignment + Read Statistic csv
* Consensus SNP table

## In Development

* Basic figures.
  + Figures from stats file

## Future Additions

* Bacterial Recombination
* Genetic Distance Heatmap



## Changes

|  |  |
| --- | --- |
| 2019 | |
| July 12th | Added Mandatory FASTQ trimming |
| July 5th | SNP called by individual sample, more confident SNP calling. Individual files then merged.  Freebayes parallel used. |
| June 27th | SNP table functionality |
| June 17th | Stat file created |

## Before You Use This Pipeline

Before we start analysis on bacterial evolutionary data, we should make sure some basic caveats are meet before running the pipeline. For starters, we need pair read FASTQ sequence data. For calling SNPs, pair read sequence data is better for making more accurate base calls, so for best results the pipeline only takes pair reads. Next, make sure your FASTQ files have the prefix .R1.fastq.gz and .R2.fastq.gz. It doesn’t matter if the file is gziped or not, but at least the file should end in .R1.fastq or .R2.fastq. Lastly, your isolate samples should be related to each other in some way or fashion (it wouldn’t be appropriate to compare Flu isolates with Tuberculosis samples).

## FASTQ Trimming

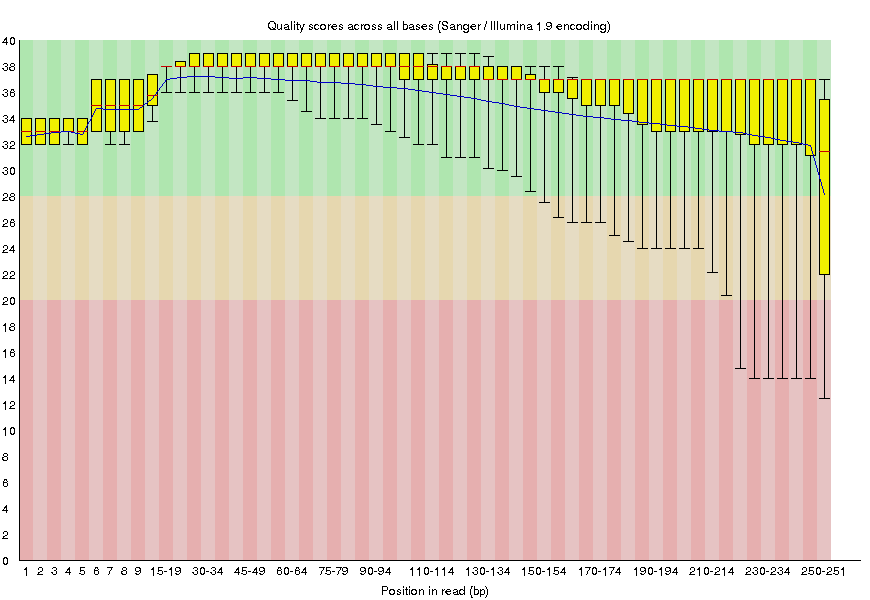
Base quality, or confidence in the base read we are witnessing will tend to drop off due to the biochemistry of sequencing. During each cycle of the process the sequencer washes chemicals that include variants for all four nucleotide over the flow cell (which has different clusters with identical DNA fragments for each cluster). The nucleotides have a blocker (terminator cap) so that only 1 base gets added to each molecule of DNA at a time. After the detection of the coupled fluorescence signal the blocker can be removed and the cycle can start again. This way, the DNA fragments in each cluster get sequenced synchronously by expressing specific fluorescence signals.

The main reason for the decreasing sequence quality is the so-called phasing. Phasing means that the blocker of a nucleotide is not correctly removed after signal detection. In the next cycle no new nucleotide can bind on this DNA fragment and the old nucleotide is detected one more time whereby the fluorescence signal of this old nucleotide (probably) differs from the synchronous signal of the other nucleotides

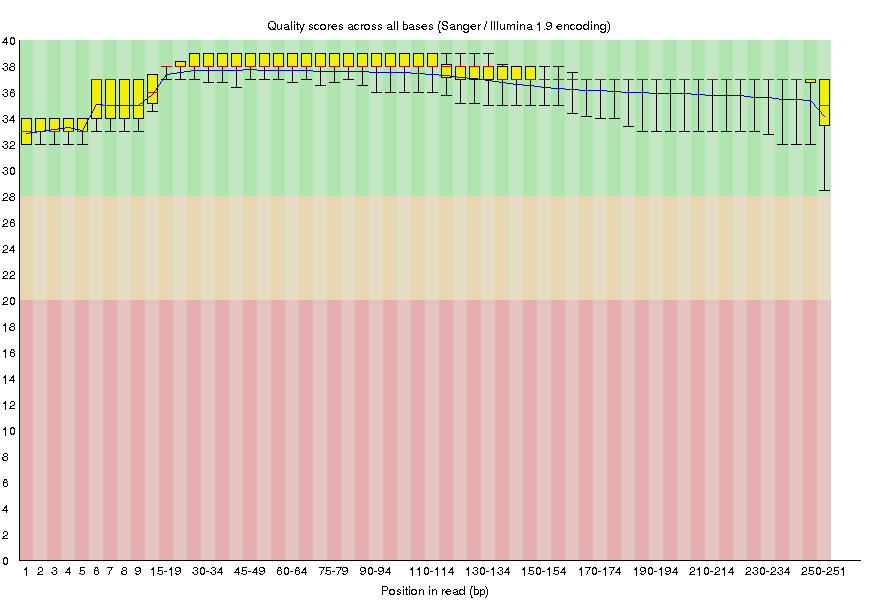
Even though some FASTQ files are already judiciously filtered, not all data that is gathered through public repositories might not be of great quality. Because of this, the pipeline makes the decision to just filter poor quality reads.

To visually show how read quality is improved through trimming, let’s observe these two figures created from FASTQC that show the effect of read trimming:

**Original – no trimming**



**Trim poor-quality ends**



## Align reads and sort output

We start by assuming that you possess reads and also a reliable reference genome to align. We should align to the genome because otherwise the reads we have are just individual blocks of sequence; how should they relate to each other? BAM files are used to delineate this relationship between reads and where the read maps to on the reference genome. After successful mapping, we are not guaranteed that your reads are in the proper order. For downstream analysis, it is best to sort the mapped reads within the BAM file.

bwa mem -M -t 4 reference\_genome.fasta sample.R1.fastq sample.R2.fastq | / samtools sort -@4 -o sample.sorted.bam > output.quiet.log

|  |  |
| --- | --- |
| bwa mem | this is the one of best-known algorithm for aligning short pair reads to a reference genome |
| -M | Mark shorter split hits as secondary (for **Picard** compatibility). |
| -t | number of processors to use when aligning to the reference. |

|  |  |
| --- | --- |
| samtools sort | Sort alignments by leftmost coordinates |
| -@ | number of processors to use when sorting the SAM file output. the more processors, the faster the program becomes. |
| -o | BAM output file name. |
| -> output.quiet.log | I do this to not make the screen so verbose. Totally optional. |

## Remove Duplicates

Duplicate reads are inevitable in the PCR process of creating the sequences we study (<http://www.cureffi.org/2012/12/11/how-pcr-duplicates-arise-in-next-generation-sequencing/>

) and if they stay within our BAM file, they could skew later downstream analysis results by possibly inflating the importance of certain ‘called SNPs’ especially if these variants are sequencing errors that were replicated. It is best to just remove them all together.

java -jar $EBROOTPICARD/picard.jar MarkDuplicates INPUT= sample.sorted.bam / OUTPUT=sample.nodup.bam METRICS\_FILE= sample.nodup.metric.txt / REMOVE\_SEQUENCING\_DUPLICATES=true REMOVE\_DUPLICATES=true /

|  |  |
| --- | --- |
| picard.java | A set of command line tools (in Java) for manipulating high-throughput sequencing (HTS) data and formats such as SAM/BAM/CRAM and VCF. |
| MarkDuplicates | Picard command line tool will go through a bam file and find the duplicates |
| REMOVE\_SEQUENCING\_DUPLICATES | If true remove 'optical' duplicates and other duplicates that appear to have arisen from the sequencing process |
| REMOVE\_DUPLICATES | if true, will remove all duplicates |

## Call SNPs, hard filter

The most time consuming of these steps is the process of finding what is a SNP and what is not. First, we look at every shared position from the isolate sample and then determine if the depth of nucleotides found in that position from the BAM file constitutes proof of a variant. Then we have a candidate for a consensus SNP! We record these potential SNPs in a file format known as VCF, which contains information on genome position and also type of variant displayed. However, the quality of SNP calls varies, so by placing rigid criteria for the type of SNPs we wish to study through caveats and rules (hard filtering), we end up with a VCF file that has SNPs that the researcher feels comfortable with.

After creating the hard filtered VCFs for each isolate, we can intersect all these VCF files to only keep the SNPs that are common in all isolates. After every file has been intersected, we merge them into one VCF file separated by isolates, making it easy to create files that can be used to create phylogenetic trees.

freebayes-parallel chrom\_ranges.txt 8 -E -1 -e 1 -u --strict-vcf -f /path/to/reference /path/to/isolate/bamfile > isolate.individual.raw.snp.

|  |  |
| --- | --- |
| freebayes | [Bayesian](http://en.wikipedia.org/wiki/Bayesian_inference) genetic variant detector designed to find small polymorphisms, specifically SNPs (single-nucleotide polymorphisms), indels (insertions and deletions), MNPs (multi-nucleotide polymorphisms), and complex events (composite insertion and substitution events) smaller than the length of a short-read sequencing alignment. |
| -parallel chrom\_ranges.txt | A provided file that breaks the reference genome into discrete pieces to parallelize the computation. |
| -E | Allow complex alleles with contiguous embedded matches of up to this length. |
| -e | Exclude reads with more than N separate gaps. |
| -u | Ignore complex events |
| --strict-vcf | output should be a vcf file |
| -f | Use FILE as the reference sequence for analysis. An index file (FILE.fai) will be created if none exists. |

bcftools filter -s LowQual -i '%QUAL>150 && TYPE=\"snp\" && INFO/DP > 10' isolate.individual.raw.vcf output.filter.vcf

|  |  |
| --- | --- |
| bcftools filter | Apply fixed-threshold filters to VCF file |
| -s | if the variant site doesn’t pass all caveats, label it as such. |
| -e | exclusion statement. vSNP didn’t include variants with QUAL scores lower than 20, and I choose to exclude indels. |
| -> | analysis ready VCF. |

python vcf2phylip.py -i output.filter.vcf

|  |  |
| --- | --- |
| vcf2phylip.py | Convert SNPs in VCF format to PHYLIP, NEXUS, binary NEXUS, or FASTA alignments for phylogenetic analysis <https://github.com/edgardomortiz/vcf2phylip> |

bcftools isec -p /dir -n=[# of samples] [samples you want to intersect]

|  |  |
| --- | --- |
| bcftools isec | intersect the available vcf files and return only SNPs that are common amongst all files |
| -p | output all the created intersection files to this directory |
| -n | only print out VCF files where records are shared amongst the designated number. If you put the number of samples, we will output only files that are the intersection |

bcftools merge [samples to merge] > output.merge.vcf

|  |  |
| --- | --- |
| bcftools merge | merge the available vcf files into one vcf |

## Output: RAxML (preliminary tree)

In the second phase of this analysis, we will be interested in phylogenetic properties associated with a multi host disease using a powerful program known as BEAST which implements Bayesian methods to infer tree features based on varying models. Until then, in order to see if our results from the pipeline make sense, we should just create a simple tree that shows the basic evolutionary relationship amongst the sequences we see. RAxML is a popular choice for making these trees because of using maximum likelihood methods to infer relationships; based on a nucleotide substitution model that the researcher deems is biologically appropriate, the program finds the MOST LIKELY tree topology GIVEN the sequence data presented.

mpirun raxmlHPC-MPI-AVX -s 53\_isolates.filter.min4.phy -n 53\_isolates -m GTRGAMMA -N /. 100 -p 1000

|  |  |
| --- | --- |
| mpirun raxmlHPC-MPI-AVX | Sapelo2 way of calling RAxML software. Produces maximum likelihood trees based on the SNP data. |
| -s | name of the file we will perform maximum likelihood on. |
| -n | what name should the output trees contain |
| -m | the nucleotide substitution model used to predict sequence relatedness. GTRGAMMA was used as the option in a paper comparing the effectiveness of tree reconstruction software. On E. Coli data, RAxML with GTRGAMMA did the best at predicting relatedness (of course, given the data is correct) <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5930550/pdf/wellcomeopenres-3-15939.pdf> |
| -N | how many iterations would you like the algorithm to run in order to find the best tree? |
| -p | a random seed. Maintains the same parameters for a run of raxml. Useful for recovering results. |

## Output: SNP Table

After SNP calling is complete, creates csv file that shows the shared SNPs at each site in the isolate genomes. Columns include the Reference name, position in genome, the QUAL score of the SNP site, and nucleotide at that position. Rows contain the isolate names and the collection of all SNPs at found in the isolate genome. An example format is displayed:

|  |  |  |
| --- | --- | --- |
|  | reference 20 | reference 546 |
|  | 98 | 150 |
| Isolate 1 | A | T |
| Isolate 2 | C | G |
| Isolate 3 | A | T |
| Isolate 4 | A | A |

## Output: Alignment Stats

This file just contains basic stats on the isolate sequence reads (Quality scores, Read Length, File size) and the corresponding isolate BAM file (Mapped Reads, Coverage, Unmapped reads). Example format is as follows:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| sample\_name | R1\_size(full) | R2\_size(full) | Q\_ave\_R1 | Q\_ave\_R2 | R1\_ave\_read\_length | R2\_ave\_read\_length | total\_mapped\_reads | ave\_coverage | unmapped\_reads |
| 00-17MIDNRdeerMontm\_S23\_L001 | 22092 | 22159 | 35.2438 | 32.9664 | 226.845 | 227.271 | 2404271 | 122.417 | 7734 |
| 00-21MIDNRdeerOts\_S14\_L001 | 21892 | 21983 | 35.4091 | 31.8204 | 238.18 | 238.572 | 1992234 | 105.899 | 7170 |
| 00-27MIDNRdeerAlp\_S27\_L001 | 21606 | 21697 | 35.0635 | 31.6564 | 234.922 | 235.421 | 1724278 | 90.562 | 6758 |
| 00-32MIDNRdeerOts\_S19\_L001 | 21591 | 21626 | 34.8233 | 31.982 | 229.27 | 229.629 | 1806447 | 92.86 | 8898 |
| 00-33MIDNRdeerPI\_S7\_L001 | 21595 | 21606 | 34.7459 | 31.2365 | 236.42 | 236.717 | 1603441 | 84.7634 | 7667 |

## Output of Alignment Pipeline (Developing)