Workshop on identifying drug resistance mutations in TB sequencing data
Tuesday, December 13th
Christopher Desjardins, Ph.D.
Broad Institute

In this workshop we'll examine some sequencing data from isolates of multidrug-resistant tuberculosis. We'll go from downloading the sequencing data all the way to identifying which drugs the strain is resistant to.

Downloading data from SRA

Sometimes you'll be analyzing data you sequenced yourself. But other times you will be augmenting your own data with already published sequence or even conducting a study entirely on already published sequences. In this case it is quite useful to use the SRA-toolkit (https://www.ncbi.nlm.nih.gov/books/NBK158900/) to download reads from NCBI's short read archive (SRA), where an enormous amount of sequencing data is deposited. The SRA-toolkit can download and convert SRA data to a variety of formats, but for our purposes we will only use the command that converts SRA data to FASTQ format. FASTQ format is going to be the format of sequencing data most commonly used by alignment and assembly programs.

We're going to start by downloading data from an isolate of drug-resistant *Mycobacterium tuberculosis*. The SRA identification number of the sequencing isolate is SRR1181122.

```
fastq-dump --split-3 SRR1181122
```

The option `--split-3` is telling the program to write two, potentially three, different FASTQ files. The way sequencing works is that you break of your DNA into many smaller fragments of DNA. Then you sequence both ends of each fragment, creating a pair of reads (called mates) for each fragment. Therefore you usually get two FASTQ files: one for the left read and one for the right read. Sometimes, you get a third file of reads without mates, depending on whether they are present in the dataset. It is important to exclude them from your two primary FASTQ files because they can cause downstream problems if included.

Let's take a closer look at the FASTQ format. Every sequence gets four lines in the file:

```
@SRR1181122.1
AGATTAGCATCACTGCTGGGTCCGTC
+SRR1181122.1
CCCFFFFFHHHHHJJJJJJJGHHFGI
```

The first and third lines are the read name, the second line is the actual sequence, and the fourth line shows the quality of, or confidence in, the base call at each

position. You can read more about the FASTQ format on Wikipedia (https://en.wikipedia.org/wiki/FASTQ_format).

Aligning reads to a reference with BWA

Once you reads have been converted to FASTQ format, the next step is to align those reads to a reference. We're going to use BWA (http://bio-bwa.sourceforge.net/), which is a widely used aligner for DNA sequence data.

The reference is usually in FASTA format, which is used to represent one or more DNA or amino acid sequences. In this format, each sequence starts with a header line that begins with ">" followed by the sequence name, while subsequent lines contain the sequence itself. For example:

```
>MT_H37RV_BRD_V5
TTGACCGATGACCCCGGTTCAGGCTTCACCACAGTGTGGAACGCGGTCGTCTCCGAACTT
AACGGCGACCCTAAGGTTGACGACGGACCCAGCAGTGATGCTAATCTCAGCGCTCCGCTG
```

Now that you have your reference, you still need to do one more thing before you begin your alignment: you need to index the reference. Think of indexing the reference like creating a table of contents or the index at the back of a book. This will allow the alignment program to search your genome much faster than if it had to read through the entire reference sequence each time it wanted to know something about it.

```
bwa index tb_reference/H37Rv.fa
```

Now that we have the referenced index, we can run the actual alignment. We give BWA the reference FASTA and the left and right FASTQ files, and we tell it to write the output to a specific file.

```
bwa mem tb_reference/H37Rv.fa SRR1181122_1.fastq
SRR1181122_2.fastq > SRR1181122.sam
```

BWA outputs alignments in a format called SAM. You can read more about it at the Samtools website (http://samtools.github.io/hts-specs/SAMv1.pdf). Next, view your SAM file with "more" or "head." The first part of the SAM file is called the header and is each line starts with @. The header gives information about the length and name of sequences in your reference, and what command produced the current SAM file. Below the header are columns that include all the information from your original FASTQ file plus information on how it aligns (or doesn't align) to your reference.

Unfortunately, while SAM files are human readable, they actually aren't very machine readable. So you'll want to convert your SAM files to BAM files, which is an equivalent format that will look like gibberish to you but is much more easily

understood by the computer. You'll also notice that the BAM file is more compact and requires less space to store on your computer, making it a preferred format for long-term storage of alignment data. For this conversion we're going to use a suite of tools for viewing and manipulating SAM and BAM files called Samtools (http://samtools.github.io/). Samtools has a number of subcommands and the one we'll use first is `view`.

```
samtools view -bh SRR1181122.sam > SRR1181122.bam
```

We've used two options in our command here: `-b` tells the program to output in BAM format and `-h` tells the program to include the header in the output. If we lose the header, this can cause problems in downstream analysis.

Finally, we'll index and sort our BAM file as many downstream applications will expect this and fail if we haven't done it.

```
samtools sort SRR1181122.bam > SRR1181122.sorted.bam

samtools index SRR1181122.sorted.bam
```

Calling SNPs with Pilon

Now that we've aligned our reads to our reference and prepared our BAM file, we're ready to call SNPs. For this we're going to use Pilon (https://github.com/broadinstitute/pilon/wiki), a dual function tool that does both assembly improvement and SNP calling. Two nice aspects of SNP calling in Pilon for our purposes are that it doesn't require a lot of user input and that it was originally designed for SNP calling in haploid genomes. Since Pilon is written in a computer language called java, we have to call the language first and then tell it where the program is. We also have to give it the genome FASTA, our aligned BAM file, tell it to call SNPs instead of improve an assembly, and where to write output, all like so:

```
java -Xmx4G -jar /usr/bin/pilon-1.20.jar --genome
tb_reference/H37Rv.fa --bam SRR1181122.sorted.bam --variant --
output SRR1181122
```

Pilon writes SNPs to a file format called VCF. You can read about VCF format at the Samtools website (https://samtools.github.io/hts-specs/VCFv4.2.pdf). The first part of the VCF file is called the header and is each line starts with **##**. The header gives a lot of information about how the VCF was generated and explains abbreviations that appear later in the VCF. The final header line of the VCF starts with **#** and gives names for the 10 columns in the body of the VCF:

1) `#CHROM`: the fasta sequence

2) POS: the position in the fasta sequence
3) ID
4) REF: the reference base at that position
5) ALT: the alternate base at that position
6) QUAL: quality of the base call at that position
7) FILTER: whether the position passes or fails quality controls
8) INFO: more detailed information
9) FORMAT: format of the following genotype columns
10) SAMPLE: sample genotype

Take a few minutes to examine the VCF. Can you use your UNIX skills to identify lines with variants?

Because there is information about every base, the VCF file produced by Pilon is quite large. However, we can simply the VCF if remove high quality reference bases and assume an then assume any base missing is a high quality reference base, which can be recovered from the original reference fasta. To do this we'll use a tool called reducevcf (http://genomeview.org/manual/Reducevcf).

```
java -jar /usr/bin/reducevcf.jar -i SRR1181122.vcf -o
SRR1181122.reduced.vcf –k
```

Note that the reduced VCF is only 1% of the size of the original VCF, saving a large amount of disk space. However, TB is a highly conserved genome and compression will scale inversely with divergence from the reference genome. Regardless, examine the VCF now and you'll see how much easier it is to examine sites with variants by eye.

Annotating VCFs with VCF Annotator

Once you've identified SNPs, the next step is to determine the functional impact of those SNPs. To do this we'll use VCF annotator (http://vcfannotator.sourceforge.net/), a script that reads in a genome sequence in FASTA format, an annotation in GFF format, and SNP calls in VCF format, and predicts functional impacts of SNPs.

```
VCF_annotator.pl --genome tb_reference/H37Rv.fa --gff3
tb_reference/H37Rv.gff3 --vcf SRR1181122.reduced.vcf --codon-
based > SRR1181122.reduced.annot.vcf
```

If you look at the new VCF file, you'll notice an 11th column has been added. This column gives extensive information about the functional effect of the mutation. A few key effect types are noted at the end of the column, including:

(NSY): nonsynonymous mutation

`(SYN)`: synonymous mutation
`(NON)`: nonsense mutation (gain of stop codon)
`(RTH)`: read-through mutation (loss of stop codon)
`INSERTION[length]`: insertion, with length in brackets
`DELETION[-length]`: deletion, with length in brackets

Take a few minutes to examine the VCF. Can you see different mutation types? Can you count the frequency of each mutation type?

Identifying drug resistance mutations

Unfortunately, there aren't good available tools for finding specific mutations in a VCF (at least not that I know of). However, you can use some basic command lines skills to pull them out.

First, we're going to look for a mutation in the *katG* gene (RVBD_1908c) called S315T (serine to threonine at codon 315), which encodes resistance to isoniazid, and is the most common drug resistance mutation in TB genomes. We can do this in a number of ways:

1) Look for any mutations in *katG*:

```
grep -i katg SRR1181122.reduced.annot.vcf
```

2) Look for any mutations in Rv1908c:

```
grep RVBD_1908c SRR1181122.reduced.annot.vcf
```

3) Look specifically for the S315T mutation in any gene:

```
grep Ser-315-Thr SRR1181122.reduced.annot.vcf
```

Can you think of pros and cons to each approach?

The isolate SRR1181122 came from a hospital in Belarus. The first-line treatment for TB in Belarus is a four-drug cocktail including isoniazid, rifampicin, pyrazinamide, and ethambutol. Below is a more comprehensive table of known drug resistance mutations for these drugs in TB. Can you figure out how resistant isolate SRR1181122 is to first-line treatment?

Drug	**Mutations**
Isoniazid	Any nonsynonymous mutation in *katG* at codon 315; any mutation in the *inhA* promoter
Rifampicin	Any nonsynonymous mutation in the

	rpoB codons 430-562
Pyrazinamide	Any mutation in the *pncA* promoter; any mutation in *pncA* that causes loss of function
Ethambutol	Any nonsynonymous mutation in the *embB* codons 306, 406, or 497

For second-line therapy, patients receive a different four-drug cocktail, including a fluoroquinalone such as ofloxacin, and injectable such as amikacin, ethionamide, and streptomycin. Below is a more comprehensive table of known drug resistance mutations for these drugs in TB. Can you figure out how resistant isolate SRR1181122 is to second-line treatment?

Drug	**Mutations**
Ofloxacin	Any nonsynonymous mutation in the *gyrA* codons 88, 90, 91, or 94
Amikacin	Any mutation at *rrs* positions 1400, 1401, or 1483; any mutation in the *eis* promoter
Ethionamide	Any mutation in *ethA* that causes loss of function; any mutation in the *inhA* promoter
Para-aminosalicylic acid	Any mutation in *thyA* that causes loss-of-function; any mutation in the *ribD* promoter*

*Much of the phenotypic resistance to para-aminosalicylic acid (PAS) has an unknown genetic basis

The NCBI short read archive contains sequence reads for over 5000 clinical strains of TB. Can you find a strain more drug resistant? Less drug resistant?

Bringing in more genomes for comparisons

Now that you understand the process, let's go the NCBI short read archive (https://www.ncbi.nlm.nih.gov/sra) and find another strain to compare ours to. All the strains from our study of TB from Belarus start with the name "XTB13," so let's find one of those. For time's sake, let's look for a strain without too much data. SRR1181122 had ~110 Mb of data, so look for a strain that has close to that amount or even a little less. Can you go from downloading the sequencing data to identifying which drugs the strain is resistant to? Is it more or less resistant than SRR1181122, and if so, should the patient be given a different treatment?