Methods

Benchmarking VERSION

We compared the detected antibiotics, memory usage, and time usage for freezeTB version 2024-05-05 https://github.com/jeremybuttler/freezeTB and TBProfiler version 6.2.0 and database version 82777ea (Phelan et al. 2019). For TBProfiler we used Nanopore settings (`--platform nanopore`) and bcftools for variant calling (`--caller bcftools`), to output and csv file (`--csv`) with the detected AMRs and spoligotype (`--spoligotype`) for the input reads (`--read1 reads.fastq`). We extracted the spoligotype and antibiotics that were not listed as "Uncertain significance", "Not assoc w R", or "Mutation from literature" using an awk script https://github.com/jeremybutter/freezeTB/benchmark/TBProfiler-process.awk. Next we built an consensus with bcftools consensus version 1.20 (Danecek et al. 2021) using the vcf file made by TBProfiler and the NC000962.3 reference included with the TBProfiler database (tbdb.fasta). We automated these steps using an bash script https://github.com/jeremybutter/freezeTB/benchmark/TBOneBench.sh.

To detect AMRs with freezeTB https://github.com/jeremybuttler/freezeTB, we used an sam file (`-sam reads.sam`) with reads mapped to the NC000962.3 reference genome and the 2nd edition of the WHO's tuberculosis drug resistant mutation catalog (Organization 2023). We mapped the reads to NC000962.3 using minimap2 version 2.28-r1209 (Li 2018; Li 2021) with Nanopore settings (`-x map-ont`) to make an sam file (`-a`). After running freezeTB, our automated bash script (TBOneBench.sh) extracted the detected antibiotics and spoligotypes.

We compared how close freezeTB consensuses were to the consensus made by TBProfiler with minimap2 and filtsam (from freezeTB). The consensus from freezeTB was mapped to the consensus from TBProfiler using minimap2 to output an sam file (`-a`) with an eqx cigar entry (`--eqx`). We then used filtsam to convert the sam file (`-sam mapping.sam`) into an tsv with the number non-anonymous and anonymous (`-p-n`) matches, snps, indels, and masked bases (`-out-stats`). Also, for cases were an reference genome was provided, we used the same steps to get the accuracy of consensuses from freezeTB and TBPofiler.

We also compared the time and memory usage of both TBProfiler and freezeTB. For time and memory comparisons we used an desktop running Ubuntu, with an AMD Ryzen 9 5950X 16 core cpu, and DDR4 ram. We measured the time using gnutime (`/usr/bin/time`) to `find the elapsed time (%e), the resident memory usage (actual memory used) (%M), and the processor usage` (%P) (/usr/bin/time -f "%eGraphs comparing time, memory usage, and output for freezeTB and TBProfile were made in R version 3.6.3 (R Core Team 2020) using the ggplot2 version 3.2.1 (Wickham 2016), data.table version 1.12.8 (Dowle and Srinivasan 2019), and viridisLite 0.3.0 (Garnier 2018) packages.

Supplemental

Supplemental methods

How freezeTB works

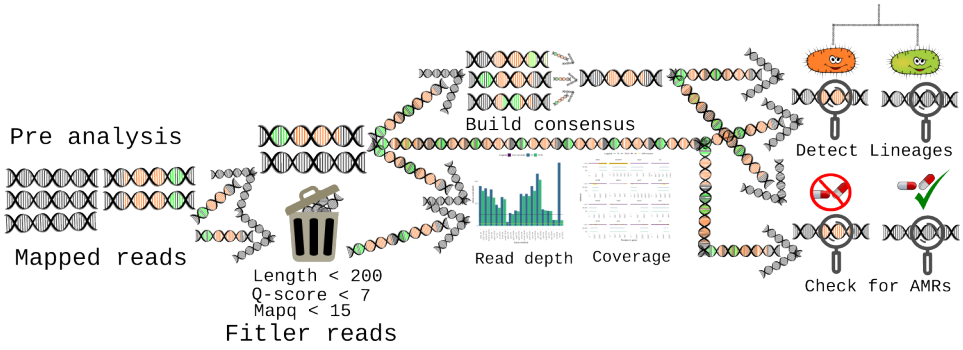

Figure Sup. 1: An flow digram showing an overview of freezeTB.

FreezeTb is an program that can detect AMRs, MIRU-VNTR lineages, spoligotypes, and build an consensus from an sam file of Nanopore sequence reads mapped to the NC000962.3 reference genomes. The output of freezeTB is an tsv file with the detected AMRs (for reads and consensus), an tsv file with the consensus MIRU-VNTR lineages, and an tsv file with the consensus spoligotype. FreezeTb also outputs an tsv with the mean read depth per gene and if requested an graph of read depth and coverage that compares the filtered reads to the unfiltered reads.

Read depth and coverage reports To find the mean read depth and coverage for each gene we build an histogram of trimmed (soft masked bases removed) unfiltered and filtered read depths of each position supported (Figure Sup. 1). To make the filtered histogram we removed any reads with an mapping quality under 15, an median Q-score under 7, an mean Q-score under 7, and that mapped to less then 200 reference bases. We then find the mean read depth for all mapped genes or gene fragments that have at least 20x read depth. Next, we make the mean read depth and coverage graphs using an R (R Core Team 2020) script that uses ggplot2 (Wickham 2016), virdisLite (Garnier 2018), data.table (Dowle and Srinivasan 2019), and the WHO's 2023 tuberculosis catalog (Organization 2023).

Primer masking Before building the consensus, we mask any primers in the reads using the primer mapping coordinates on the references (NC000962.3) genome.

Consensus building For consensus building Figure Sup. 1, we use an majority consensus step that is somewhat similar to Ivar (Grubaugh et al. 2019). We first remove any SNPs, matches, or insertions from the masked reads that have an quality score under seven. After adding all reads to the consensus, we then collapse the consensus by keeping the most supported SNP, match, deletion or insertion.

Next, we mask SNPs and matches or remove unsupported indels in the consensus that have low percentage of mapped reads. For SNPs and matches, we mask any position that has less then 50% support. While for indels, we require at least 70% support.

To find the percentage of support we divide the number of reads supporting the SNP, match, or indel by an total. The total for SNPs and matches is the number of reads supporting the position without anonymous bases. For deletions, our total is the number of mapped reads. While for insertions, we use the number of mapped reads from the neighbor with the highest number of reads.

Before outputting the consensus, we create fragments by removing any position that has less than 20x read depth. We then remove any fragment that is under 200 bases long. Finally, we output the fragments as an sam file.

AMR detection For AMRs detection (Figure Sup. 1) we use an tsv version of the WHOs 2023 tuberculosis catalog that has had all grade thee, four, and five AMRs removed. We find potential AMR variants for each sequence with the sequences reference (NC000962.3) mapping coordinates. For AMR variants with amino acid sequence, we convert the sequences AMR region into amino acids. We then check if the sequences nucleotide or amino acid sequence matches the AMR variant. Next, we detect and remove false positive AMRs by discarding variants were the sequences AMR region length differs from the AMR variants length.

On default settings freezeTb does no additional checks for frame shifts and loss of function AMRs. However, though not recommended, freezeTB does include an option to allow an simple check for frame shifts and loss of function AMRs. For loss of function AMRs, we translate the target gene in the sequence. We then look for early stop codons, lost stop codons (when possible), and lost start codons (when possible). Also, we count the number of indels and check if they are divisible by three (not an LoF).

For frame shifts we look to see if their are any indels in the AMR region. We also count the number of indels and check if they are divisible by three.

MIRU-VNTR lineages To detect MIRU-VNTR lineages (Figure Sup. 1) we identify primers sites on the reference (NC000962.3) using the sequences mapping coordinates. We find the lineage for the site by comparing the sequences repeat region an modified MIRU-VNTR table from MIRUReader (Tang and Ong

2020). To account for for indel errors we allow an 15 base pair difference between the sequences length and the actual lineages length.

Spoligotyping To detect spoligotypes we find the direct repeat region (from NC000962.3) in the consensus using its mapping coordinates. We then build the barcode by mapping each internal spacer sequence to the direct repeat region in the sequence using and Waterman Smith alignment. Next, we search the lineage database (csv) from TBProfiler for matching lineages.

References

Danecek, Petr, James K Bonfield, Jennifer Liddle, John Marshall, Valeriu Ohan, Martin O Pollard, Andrew Whitwham, et al. 2021. "Twelve Years of SAMtools and BCFtools." *GigaScience* 10 (2): giab008. https://doi.org/10.1093/gigascience/giab008.

Dowle, Matt, and Arun Srinivasan. 2019. *Data.table: Extension of 'Data.frame'*. https://CRAN.R-project.org/package=data.table.

Garnier, Simon. 2018. *viridisLite: Default Color Maps from 'Matplotlib' (Lite Version)*. https://CRAN.R-project.org/package=viridisLite.

Grubaugh, Nathan D, Karthik Gangavarapu, Joshua Quick, Nathaniel L Matteson, Jaqueline Goes De Jesus, Bradley J Main, Amanda L Tan, et al. 2019. "An Amplicon-Based Sequencing Framework for Accurately Measuring Intrahost Virus Diversity Using PrimalSeq and iVar." *Genome Biology* 20 (1): 8. https://doi.org/10.1186/s13059-018-1618-7.

Li, Heng. 2018. "Minimap2: Pairwise Alignment for Nucleotide Sequences." *Bioinformatics (Oxford, England)* 34 (18): 3094–3100. https://doi.org/10.1093/bioinformatics/bty191.

———. 2021. "New Strategies to Improve Minimap2 Alignment Accuracy." *Bioinformatics (Oxford, England)* 37 (23): 4572–74. https://doi.org/10.1093/bioinformatics/btab705.

Organization, Geneva: World Health. 2023. "Catalogue of Mutations in Mycobacterium Tuberculosis Complex and Their Association with Drug Resistance." 2.

Phelan, Jody E, Denise M O'Sullivan, Diana Machado, Jorge Ramos, Yaa E A Oppong, Susana Campino, Justin O'Grady, et al. 2019. "Integrating Informatics Tools and Portable Sequencing Technology for Rapid Detection of Resistance to Anti-Tuberculous Drugs." *Genome Medicine* 11 (1): 41. https://doi.org/10.1186/s13073-019-0650-x.

R Core Team. 2020. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Tang, Cheng Yee, and Rick Twee-Hee Ong. 2020. "MIRUReader: MIRU-VNTR Typing Directly from Long Sequencing Reads." *Bioinformatics (Oxford, England)* 36 (5): 1625–26. https://doi.org/10.1093/bioinformatics/btz771.

Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org.