

Minos: variant adjudication and joint genotyping of cohorts of bacterial genomes

– supplementary information

Hunt, M.^{1,2}, Letcher, B.¹, Malone, K.M.¹, Nguyen, G.¹, Hall, M.B.¹, Colquhoun, R.M.³, Lima, L.¹, Schatz, M.C.⁴, Ramakrishnan, S.⁴, CRyPTIC consortium*, Iqbal, Z.¹

¹EMBL-EBI, Cambridge, UK

²Nuffield Department of Medicine, University of Oxford, Oxford, UK

³Institute of Evolutionary Biology, Ashworth Laboratories, University of Edinburgh, UK

⁴Department of Computer Science, Johns Hopkins University, Baltimore, MD, USA

*Please see main manuscript section “CRyPTIC consortium” for details.

Contents

1	Simulated data	2
2	Conflicting Joint Genotyping Calls	2
3	Example complex variant	3
4	Software versions and command lines	4
5	Figures	8

1 Simulated data

Simulations were performed as a sanity check on performance of all benchmarked tools, and were not intended (and are not claimed) to be good representatives of real-world data. They do however provide a way to potentially reveal limitations of tools. Simulated data sets were generated from the *M. tuberculosis* genome H37Rv using simulator (<https://github.com/iqbal-lab-org/simulator>). A mutated genome was generated for each type of variation, with variants distributed uniformly across the genome: a SNP every 200bp, an insertion of length 1 every kilobase, a deletion of length 1 every kilobase etc. Insertions and deletions were simulated up to a length of 10kbp. Paired Illumina reads were simulated at 50X depth from each mutated reference sequence using ART.

BayesTyper, GraphTyper, and Minos achieved near-perfect precision and recall on all data sets, showing that all tools are effective on clean data (Additional files 4,5: Tables S3,S4). All three tools had perfect precision when calling isolated SNPs, insertions and deletions. They also had perfect recall of SNPs, except for Minos with a recall of 99.68% using its default call filters. However, including all Minos calls results in perfect recall (and still perfect precision). This trend continued in the remainder of the simulation results. We found that the Minos default filters are negatively affected by the very clean simulated reads with idealised read depth distributions. As shown later, the filters are however effective on real data. The recall of SNPs and the full range of indels tested is shown in Supplementary figure 1. Again, all tools have near 100% recall, except filtered Minos calls which loses up to 0.9% recall on small (< 50bp) variants.

In addition to isolated SNPs, insertions, and deletions, four data sets were generated with a complex variants every kilobase, with each complex variant consisting of SNPs and/or indels of varying length in a window of size 10 to 50bp. The results were very similar across all tools (Additional file 5: Table S4).

2 Conflicting Joint Genotyping Calls

As noted in the main text, Minos resolves overlapping input variants, so that no two sites in its output contain reference positions in common. Furthermore, this means Minos cannot output two separate VCF records with incompatible genotype calls, in contrast to BayesTyper and GraphTyper. A typical example is shown next.

On the Walker 2013 outbreak data set, BayesTyper and GraphTyper reported the following two lines in their output VCF files for each sample (first five columns shown only for brevity – these are enough to describe the variants that are being called):

```
NC_000962.3      397272      .          TCGGCGGCC      T
NC_000962.3      397275      .          G              C,*
```

BayesTyper and GraphTyper made inconsistent genotype calls of these variants. For example in sample ERR046906, BayesTyper and GraphTyper genotyped the deletion at 397272 as not present (genotype 0 or 0/0), saying in particular that this sample has a G at position 397275. However, both tools also genotyped the second variant line at 397275 as having a C (genotype 1 or 1/1). These genotype calls cannot both be correct.

Minos reported the SNP and deletion alleles with the following single line in its VCF output file:

```
NC_000962.3      397272      .          TCGGCGGCC      T,TCGCCGCC
```

and called the second alternative allele TCGCCGCC as correct – this is a SNP from G to C at position 397275. Manual inspection of the reads pileup confirmed that this is the correct call.

3 Example complex variant

It is common for insertions and deletions to result in variants with more than one correct representation in VCF format. Here we show one such example, found in real *M. tuberculosis* data, which is successfully evaluated by Varifier, but not by hap.py. Here a sample (top sequence) has an insertion compared to the *M. tuberculosis* H37Rv reference genome (bottom sequence), plus a G to C SNP at position 4359165.

Alignment 1:

```

CCCCGGATGGGTGTGATTGGGGTTCCCGGGGTGATCGGGGTTCCCGCGTGATCGGGGTTCCCGGGGTGATCGGGGTTCCCGCGTGATCGGGGTTCCCGGTGTGCCCGGTGTGCCCGGGATGGC
|||||
CCCCGGATGGGTGTGATTGGGG-----TTCCCGGGGTGATCGGGGTTCCCGGGGTGATCGGGGTTCCCGGTGTGCCCGGTGTGCCCGGGATGGC
|                                     |                                     |
4359139                             4359140                             4359165

```

The corresponding VCF records are:

```

NC_000962.3    4359139    .    G    GTTCCCGGGGTGATCGGGGTTCCCGCGTGATCGGGG
NC_000962.3    4359165    .    G    C

```

This region is a tandem repeat, containing copies of the sequences GGGGTTCCCGGGGTGATC and GGGGTTCCCGCGGTGATC (which differ by one G/C SNP at position 12). The insertion and SNP can be placed in multiple positions (or, could be represented by two insertions only, which we have actually seen in real calls). Three more of the many possible alignments are shown below, with their matching VCF records.

Alignment 2:

```

CCCCGGATGGGTGTGATTGGGGTTCCCGGGGTGATCGGGGTTCCCGCGTGATCGGGGTTCCCGGGGTGATCGGGGTTCCCGCGTGATCGGGGTTCCCGGTGTGCCCGGTGTGCCCGGGATGGC
|||||
CCCCGGATGGGTGTGATTGGGGTTCCCGGGGTGATCGGGGTTCCCGGGGTGATCGGGGTTCCCGG-----TGTGCCCGGTGTGCCCGGGATGGC
|                                     |
4359165                             4359182
NC_000962.3    4359165    .    G    C
NC_000962.3    4359182    .    G    GGGTGATCGGGGTTCCCGCGTGATCGGGGTTCCCGG

```

Alignment 3:

```

CCCCGGATGGGTGTGATTGGGGTTCCCGGGGTGATCGGGGTTCCCGCGTGATCGGGGTTCCCGGGGTGATCGGGGTTCCCGCGTGATCGGGGTTCCCGGTGTGCCCGGTGTGCCCGGGATGGC
|||||
CCCCGGATGGGTGTGATTGGGGTTCCCGGGGTGATCGGGGTTCCCG-----GG-GTGATCGGGGTTCCCGGTGTGCCCGGTGTGCCCGGGATGGC
|                                     |
4359163                             4359165
NC_000962.3    4359163    .    G    GGCGTGATCGGGGTTCCCGGGGTGATCGGGGTTCCC
NC_000962.3    4359165    .    G    GC

```

Alignment 4:

```

CCCCGGATGGGTGTGATTGGGGTTCCCGGGGTGATCGGGGTTCCCGCGTGATCGGGGTTCCCGGGGTGATCGGGGTTCCCGCGTGATCGGGGTTCCCGGTGTGCCCGGTGTGCCCGGGATGGC
|||||
CCCCGGATGGGTGTGATTGGGGTTCCCGGGGTGATCGGGGTT-----CCCGGGGTGATCGGGGTTCCC-----GGTGTGCCCGGTGTGCCCGGGATGGC
|                                     |
4359159                             4359180
NC_000962.3    4359159    .    T    TCCCGCGTGATCGGGGTT
NC_000962.3    4359180    .    C    CGCGTGATCGGGGTTCCC

```

In particular, this example highlights how confusion can arise from alternative representations of the same variants. Looking in isolation at these two calls from alignments 1 and 3, one might incorrectly conclude that they are incompatible:

```

NC_000962.3    4359165    .    G    C
NC_000962.3    4359165    .    G    GC

```

4 Software versions and command lines

All software was run using Singularity containers, which were made from the definition file `singularity.def` in the github repository <https://github.com/iqbal-lab-org/minos-paper-benchmarking>. The main results, which used Cortex and SAMtools calls as input to BayesTyper, GraphTyper, and Minos, used a build of git commit `dbbc62a9388fd30869406cc63c10e70c08d45a0b`. The supplementary results that also include Snippy variant calls was run using a build of git commit `346b3fa901b26edb9f9fc946fa998ed15c456ebf`. The containers are available at <https://doi.org/10.6084/m9.figshare.16613383> and <https://doi.org/10.6084/m9.figshare.19130396>.

Simulated data

The simulated genomes were made using Simulator (<https://github.com/iqbal-lab-org/simulator>) git commit `0218c8a5b37fd72eb4e5b2df4cba9f6118f96788`. The command line was:

```
simulator mutate_fasta --seed 42 --snps 200 \  
  --dels 1000:1,1000:2,1000:3,1000:4,1000:5,1000:10,1000:20,1000:50,10000:100,\  
  20000:500,20000:1000,20000:2000,20000:5000,20000:10000 \  
  --ins 1000:1,1000:2,1000:3,1000:4,1000:5,1000:10,1000:20,1000:50,10000:100,\  
  20000:500,20000:1000,20000:2000,20000:5000,20000:10000 \  
  --complex 1000:10:3:0:0:0,1000:10:3:1:1:2,1000:50:10:1:1:3,1000:50:20:3:3:4 \  
  tb_ref.h37rv.fa out
```

ART version 20160605 (<https://www.niehs.nih.gov/research/resources/assets/docs/artbinmountrainier20160605linux64tgz.tgz>) was used to simulate Illumina reads from each of the mutated genomes output by simulator, using the command:

```
art_illumina --in mutated.fa --out reads.out \  
  --noALN --seqSys HS25 --len 150 --fcov 50 \  
  --mflen 500 --sdev 25 --rndSeed 42
```

Variant calling pipeline

The pipeline described next is implemented in the `minos-paper-benchmarking` container as a single script that runs all stages on one sample. First, all the relevant genome index files must be made (for example BWA index). This needs to be run once on each mapping reference FASTA file:

```
clockwork reference_prepare --outdir ref_dir mapping_reference.fasta
```

to make a directory (in this example, called `ref_dir`), which is used by the next command. The whole variant calling and evaluation pipeline was then run on one sample with the command

```
minospb run_one_sample \  
  --truth_mask_bed truth_mask.bed \  
  --ref_mask_bed ref_mask.bed \  
  sample_name out_directory ref_dir \  
  reads_1.fastq.gz reads_2.fastq.gz
```

The stages run by that script are described next. First, reads were trimmed using Trimmomatic version 0.36, with the options

```
ILLUMINACLIP:/Trimmomatic-0.36/adapters/TruSeq3-PE-2.fa:2:30:10
LEADING:10 TRAILING:10 SLIDINGWINDOW:4:15 MINLEN:50 -phred33
```

where /Trimmomatic/ is the root directory of the Trimmomatic download.

Reads were mapped with BWA MEM version 0.7.17 with option -M, then The output SAM file was sorted using SAMtools version 1.10.2 `samtools sort`, and then duplicates removed with `samtools rmdup` (defaults used for both commands), to make a file called `rmdup.bam`.

Variant calls were made with SAMtools and BCFtools version 1.10.2 with the command:

```
samtools mpileup -ugf rmdup.bam | bcftools call -vm -O v -o out.vcf
```

Cortex git commit 3a235272e4e0121be64527f01e73f9e066d378d3 (and dependencies VCFtools version 0.1.15 and Stampy version 1.0.32) was used with the Cortex Perl script to call variants:

```
run_calls.pl --fastaq_index in.index --auto_cleaning yes --first_kmer 31 \
--bc yes --pd no --outdir cortex.out --outvcf cortex --ploidy 2 \
--stampy_hash ref.stampy --stampy_bin /path/to/stampy.py \
--list_ref_fasta /path/to/refs_file --refbindir /path/to/ref_dir \
--genome_size N --qthresh 5 --mem_height 22 --mem_width 100 \
--vcftools_dir /path/to/vcftools --do_union yes --ref CoordinatesAndInCalling \
--workflow independent --logfile out.log
```

BayesTyper

BayesTyper version 1.5 was used, with the dependency KMC version 3.1.1. KMC was run with the command

```
kmc -t1 -m7 -k55 -ci1 -fbam rmdup.bam 01.kmc .
```

followed by

```
bayesTyperTools makeBloom -k 01.kmc
```

Then for each of the SAMtools and Cortex VCF files:

```
bcftools norm -f /path/to/ref.fa <samtools|cortex>.vcf \
-o 02.typertools.<samtools|cortex>.norm.vcf
```

Then the commands:

```
bayesTyperTools combine -v samtools:samtools.norm.vcf,cortex:cortex.norm.vcf \
-o 02.typertools.combine -z
```

```
bayesTyper cluster -r 42 -v 02.typertools.combine.vcf.gz -s \
03.typertools.samples.tsv -g /path/to/ref.fa
```

```
bayesTyper genotype --noise-genotyping -r 42 \
-v bayestyper_unit_1/variant_clusters.bin \
-y 04.typertools.ploidy.tsv -c bayestyper_cluster_data \
-s 03.typertools.samples.tsv -g /path/to/ref.fa -o bayestyper_unit_1/bayestyper
```

GraphTyper

GraphTyper version 2.5.1 was used, together with custom Python code to glue the various stages together and other required processing. These commands were used on the input VCF files:

```
bcftools concat -a -O v <VCF files> | bcftools sort -Oz -o concat.vcf.gz
```

```
tabix -p vcf concat.vcf.gz
```

to make a sorted bgzipped indexed VCF file.

Then to genotype using the default mode:

```
graphtyper genotype --output 01.genotype -vverbose --sam rmdup.bam \  
  --region_file 00.regions.txt --threads 1 /path/to/ref.fa \  
  --vcf concat.vcf.gz
```

```
bcftools concat -a -O v <list of all VCF files made by previous command>
```

Alternatively, to genotype using SV mode:

```
graphtyper genotype_sv --output 01.genotype -vverbose --sam rmdup.bam \  
  --region_file 00.regions.txt --threads 1 /path/to/ref.fa concat.sv.vcf.gz
```

```
bcftools concat -a -O v <list of all VCF files made by previous command>
```

Minos

Minos `git commit c22f4b00c61be7854336dad5b31b44c0d11e8e81` was used, and dependencies `gramtools` (`git commit 8af53f6c8c0d72ef95223e89ab82119b717044f2`), `vcflib` (`git commit da9929c00938b1d1b042e8ce42514211ef34875d`), and `vt` (`git commit a549707d8e33006ed89c46ca7fbda007123418ab`). The command line was:

```
minos adjudicate --reads rmdup.bam outdir /path/to/ref.fa samtools.vcf cortex.vcf
```

Genome masks

An existing genome mask was used for the *M. tuberculosis* H37Rv reference genome, which is routinely used by Public Health England variant calling pipeline COMPASS.

For all other genomes, masks for each mapping and truth genome were generated using agreement between the reads and the genomes as follows. The whole process is wrapped in a single script, run with the command:

```
minospb make_mask genome.fasta reads.1.fastq.gz reads.2.fastq.gz outdir
```

Illumina reads expected to perfectly match the genome (ie reads that made the genome assembly sequence) were trimmed, mapped, and PCR duplicates removed using exactly the same method as described earlier for variant calling. The resulting BAM file is parsed using `pysam`. A position is included in the mask if the depth of reads matching the reference is less than 5, or if the percent of reads that match the reference is less than 90%. The output is a mask in BED file format, which was then used as input to Varifier (see next section).

Varifier

varifier git commit ad3d6db3b851eb46357b0308e7496580e191700f was used, with the command line

```
varifier vcf_eval --ref_mask ref_mask.bed \  
  --truth_mask truth_mask.bed --filter_pass PASS \  
  truth_ref.fasta mapping_ref.fasta to_evaluate.vcf outdir
```

To evaluate all calls made by a tool in order to see the effect of the FILTER column in the VCF, the option `--filter_pass` was omitted.

Joint Genotyping

The per-sample VCF files were generated using the variant calling method described above, making calls from SAMtools and Cortex. For each sample, these were input into Minos to make a single VCF for each sample. This was run using the script described above `minospb run_one_sample`.

These per-sample VCF files were used as input to the Minos joint genotyping pipeline, which is implemented using Nextflow. The Nextflow script and configuration file are in the `nextflow/` directory of the Minos repository, called `regenotype.nf` and `regenotype.config`. On the Walker 2013 data set, the pipeline was run with the command:

```
nextflow run \  
  -w /path/to/working/dir/  
  -with-singularity /path/to/minospb_image \  
  -c regenotype.config \  
  -profile medium \  
  regenotype.nf  
  --ref_fasta /path/to/tb_ref.h37rv.fa \  
  --make_distance_matrix \  
  --manifest manifest.tsv \  
  --mask_bed_file tb.h37rv.mask.R000000039_repreions.bed \  
  --outdir output_directory
```

where `manifest.tsv` is the required manifest file, containing for each sample its name, input VCF file, and BAM file of reads. The reads must be in a mapped, sorted, indexed BAM file for the pipeline. This enables it to split the reference into chunks and map reads to each chunk, which saves run time and memory. On the large CRyPTIC and Mykrobe data sets, the option `-profile large` was used, which allocates more memory and CPUs to tasks where necessary, and the option `--make_distance_matrix` was omitted.

5 Figures

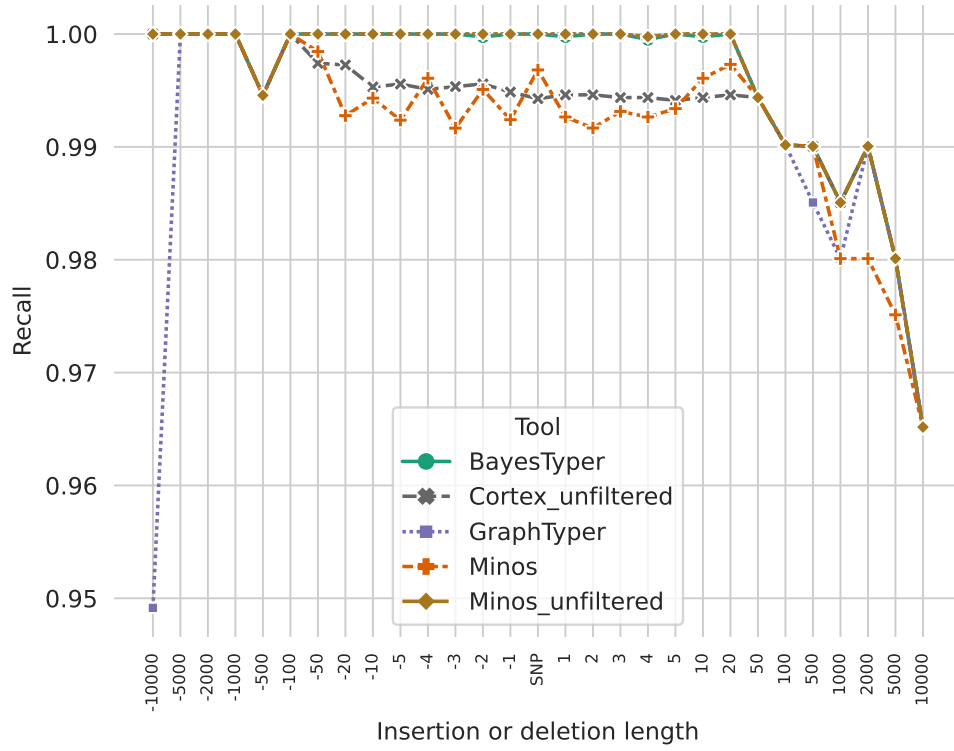


Fig S1: Recall of isolated deletions, SNPs, and insertions on the simulated data set. Negative length values show deletions, and positive values show insertions. SNPs are shown in the centre. Note that the x -axis is non-linear.

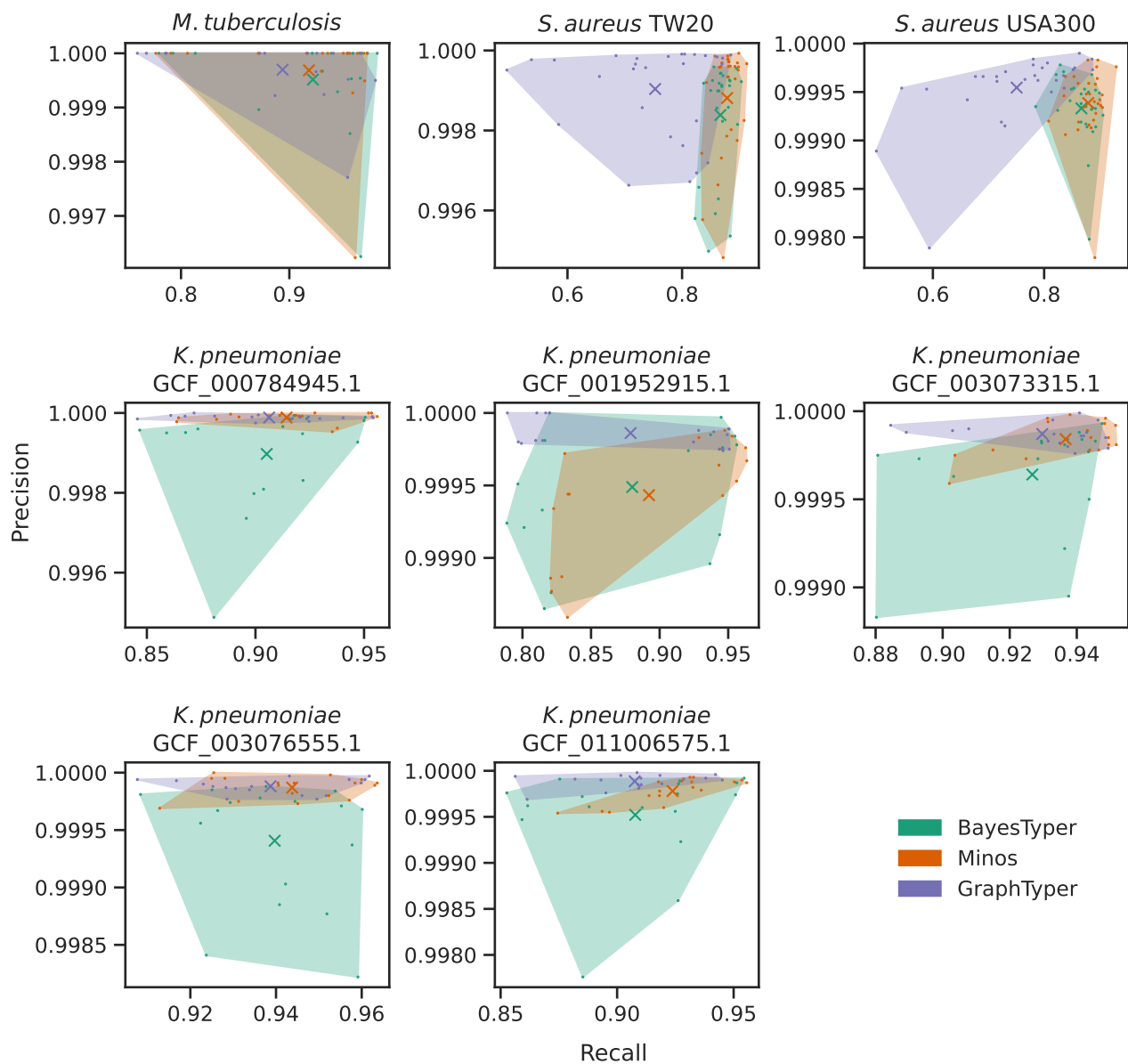


Fig S2: Comparison of BayesTyper, GraphTyper and Minos on the bacteria data set. Precision and recall of each sample is shown as a dot, and the mean precision and recall for each tool is marked with a cross. The convex hull of the data points for each caller is shaded with an associated colour. Note the axis scales are different between the plots.

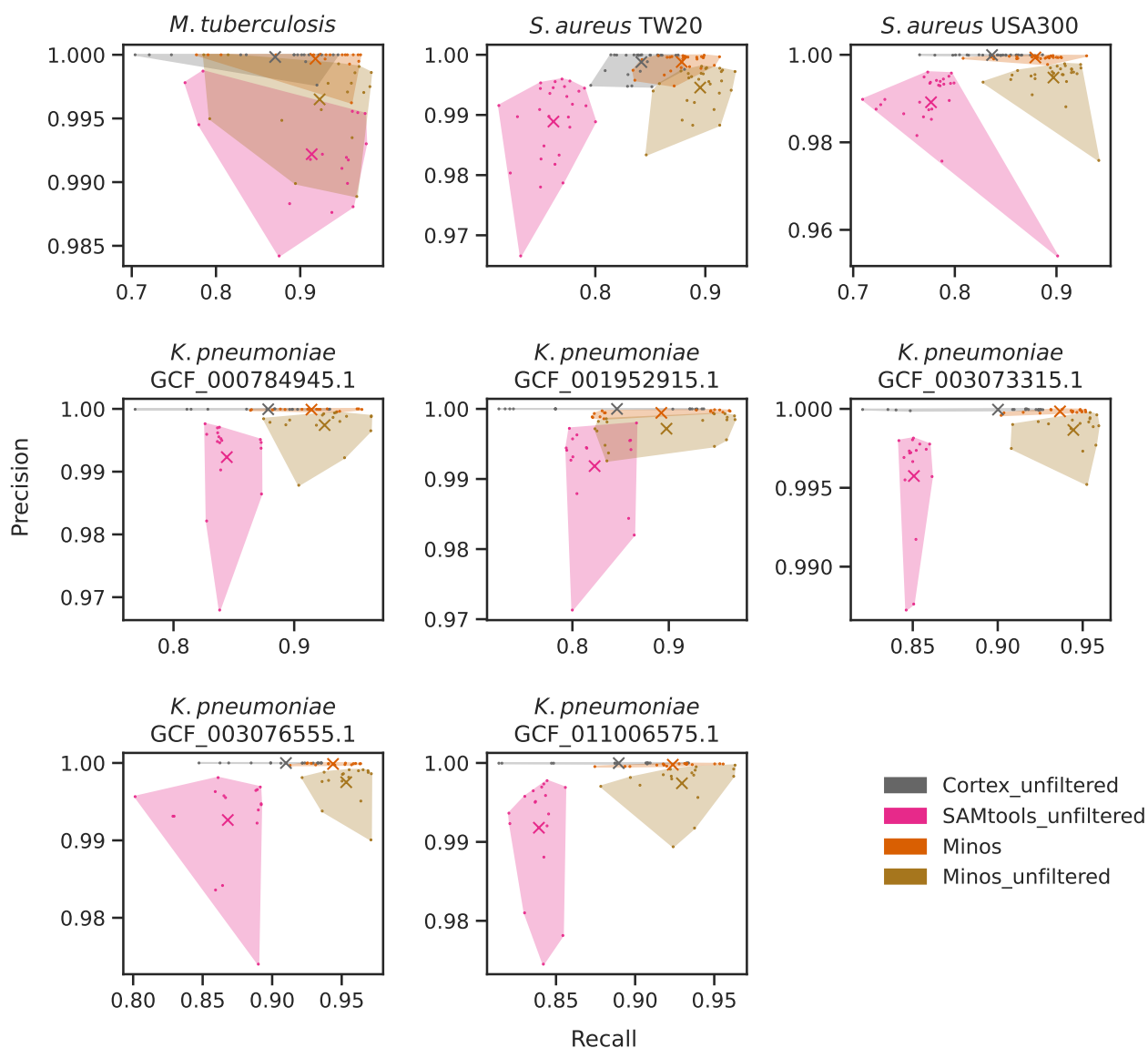


Fig S3: Comparison of filtered and unfiltered Minos results, together with unfiltered Cortex and unfiltered SAMtools on the bacteria data set. Precision and recall of each sample is shown as a dot, and the mean precision and recall for each tool is marked with a cross. The convex hull of the data points for each caller is shaded with an associated colour. Note the axis scales are different between the plots.

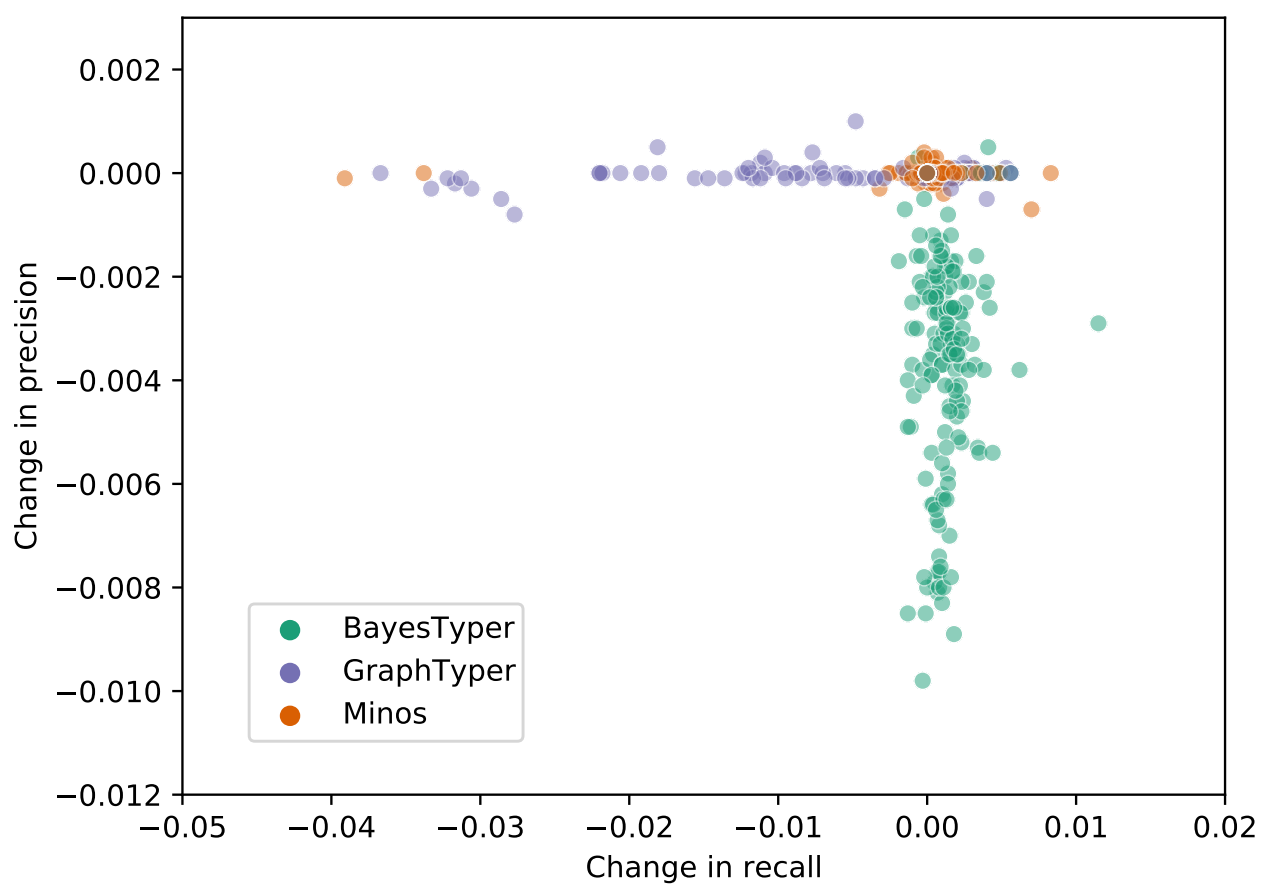


Fig S4: Effect of including variants from Snippy, in addition to Cortex and SAMtools, as input to BayesTyper, GraphTyper, and Minos. The change in precision and recall is shown across all simulated data and all bacteria data. Each point represents one run of one tool on one sample.

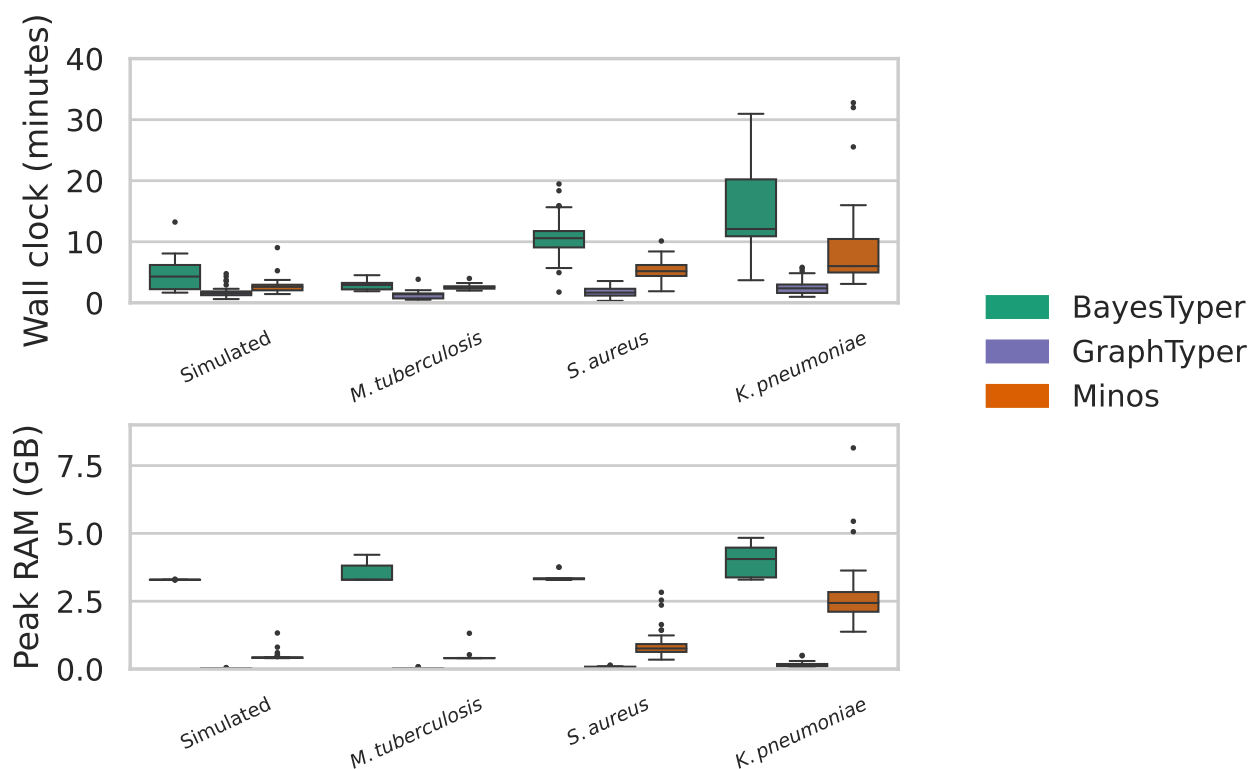


Fig S5: Run time and peak RAM usage on the simulated and bacteria data sets. Values are taken from the output of the Unix command `time -v`.

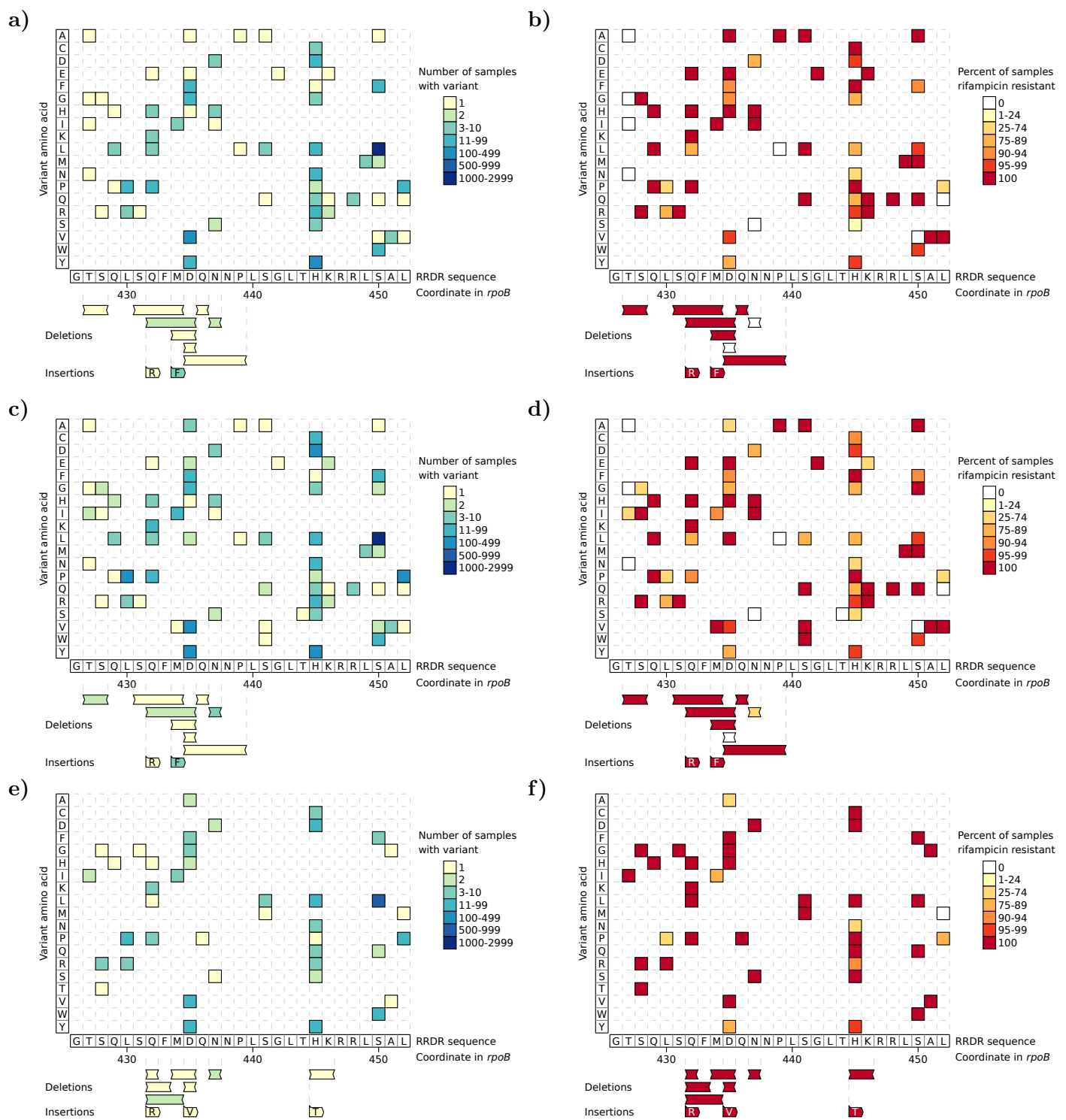


Fig S6: All amino acid variants identified in the RRDR of the *rpoB* gene by joint genotyping the CRyPTIC and Mykrobe *M. tuberculosis* data sets. a) and b) are the same as Figure 4 in the main manuscript, showing samples from the CRyPTIC data set that have high quality phenotypes only. c) and d) again show the CRyPTIC set, but include all samples regardless of phenotype quality. e) and f) show the Mykrobe data set. In a), c), e) each variant is coloured by the number of samples possessing that variant. Plots b), d), f) colour the variants by the percent of samples with that variant that are rifampicin resistant. Each plot shows the RRDR region from left to right. Single amino acid variants are shown in the upper grid, with the y axis corresponding to the variant amino acid. The lower area shows deletions and insertions, with the inserted sequence given in the coloured boxes. For example, in a) and b) the leftmost deletion of amino acids TS at position 427-428 is found in one resistant sample. The leftmost insertion adds R after the S at position 431.

Reference: CGAT
Variants: GA2A, A3C

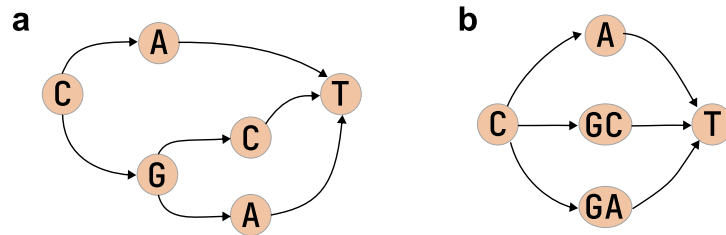


Fig S7: a) Nested and b) non-nested graphs. The nested graph in a) is not allowed, and is transformed by Minos into b) for graph construction and then read mapping. Note that the two graphs contain the same sequences.

Variants: G3GG, G4C, C6A, A8T, TG9G

Position	1	2	3	4	5	6	7	8	9	10	11
Ref	A	G	G	G	C	C	A	A	T	G	C
Alt			GG	C		A		T	G	-	

Fig S8: Duplicated paths. The listed variants when applied to the reference sequence AGGGCCAATGC result in the possible reference and alternative alleles shown. The path marked in orange shows an allele combination that results in exactly the same sequence as the reference sequence. Minos prevents this situation from arising, so that reads cannot have the option of mapping to two different paths that are in fact the same sequence.

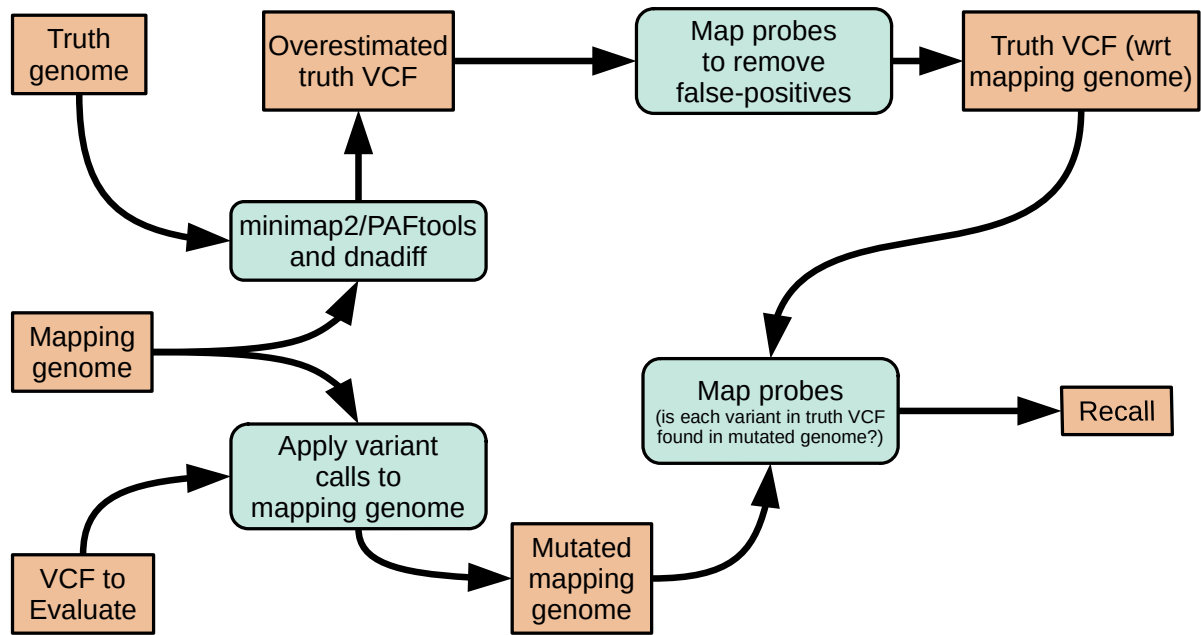


Fig S9: Recall pipeline implemented by Varifier. See the Methods section in the main text for a description.