Examining bacterial variation with genome graphs and Nanopore sequencing

Michael Benjamin Hall

Supervisor: Dr. Zamin Iqbal

European Bioinformatics Institute
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

Jesus College

November 2021

I would like to dedicate this thesis to

Declaration

declaration

Michael Benjamin Hall

November 2021

Acknowledgements

And I would like to acknowledge ...

Abstract

A bacterial species' genetic content can be remarkably fluid. The collection of genes found within a given species is called the pan-genome and is generally much larger than the gene repertoire of a single cell. A consequence of this pan-genome is that bacterial genomes are highly adaptable and thus variable.

The dominant paradigm for analysing genetic variation relies on a central idea: all genomes in a species can be described as minor differences from a single reference genome, which serves as a coordinate system. As an introduction to this thesis, we outline why this approach is inadequate for bacteria and describe a new approach using genome graphs.

In the first chapter, we present algorithms for *de novo* variant discovery within such genome graphs and evaluate their performance with empirical data. The remaining chapters address a question relating to a critical bacterial pathogen: can Nanopore sequencing of *Mycobacterium tuberculosis* provide high-quality public health information? We collect data from Madagascar, South Africa, and England to help answer this question. First, we assess outbreaks identified using single-reference and genome graph methods. Second, we evaluate AMR predictions and introduce a framework for using genome graphs to improve current methods. Lastly, we train an *M. tuberculosis*-specific Nanopore basecalling model with considerable accuracy improvement.

Together, this thesis provides general methods for uncovering bacterial variation and applies them to an important global public health question.

Table of contents

List of figures xix

List of tables xxxv

Abbreviations xli

1 Background 1

1.1 Bacterial genomes . . . 1

1.1.1 Drivers of bacterial diversity . . . 1

1.1.2 The pan-genome . . . 2

1.1.3 How are pan-genomes analysed? . . . 4

1.1.4 Variant calling of bacterial genomes . . . 6

1.2 Genome graphs . . . 8

1.2.1 Existing methods . . . 9

1.3 Pandora: bacterial pan-genomics with reference graphs . . . 12

1.3.1 Population reference graph construction . . . 12

1.3.2 Index, quasi-map, and sequence inference . . . 13

1.3.3 Variation inference . . . 15

1.4 Nanopore sequencing . . . 16

1.4.1 Basecalling . . . 17

1.4.2 Variant calling . . . 19

1.4.3 Benefits and limitations . . . 20

1.5 Tuberculosis and its causative agent . . . 22

1.5.1 Epidemiological and phylogenetic diagnostics . . . 23

1.5.2 Antimicrobial resistance prediction . . . 25

1.6 Executive summary of this thesis 28

2 Variant discovery in genome graphs **31**

2.0 Publication and collaboration acknowledgements 31

2.1 Introduction 32

2.2 Methods 33

2.2.1 Finding candidate regions 33

2.2.2 Enumerating paths through candidate regions 34

2.2.3 Pruning the path-space in a candidate region 35

2.2.4 Updating a PanRG with candidate paths 37

2.3 Initial assessment via simulations 37

2.3.1 Methods 37

2.3.2 Evaluation 38

2.3.3 Results 39

2.3.4 Summary 42

2.4 Validation with empirical data 43

2.4.1 Dataset 44

2.4.2 Variant calling 45

2.4.3 Evaluation 45

2.4.4 Effect of different Nanopore basecalling models 46

2.4.5 Performance of Pandora against single-reference tools 48

2.4.6 Summary 49

2.5 Discussion 50

2.6 Limitations and future work 53

2.6.1 Path enumeration 53

2.6.2 Cycle detection 53

2.6.3 Nanopore homopolymer deletions 54

2.6.4 Inserting novel alleles into the PanRG 54

2.6.5 Insertions and deletions 55

2.7 Availability of data and materials 55

3 Nanopore sequencing for *M. tuberculosis* transmission clustering **57**

3.0 Publication and collaboration acknowledgements . . . 57
3.1 Introduction . . . 57
3.2 Dataset . . . 59
3.2.1 Illumina sequencing . . . 59
3.2.2 Nanopore sequencing . . . 60
3.2.3 PacBio sequencing . . . 60
3.2.4 Data preprocessing . . . 61
3.3 Genome assemblies for validating variant calls . . . 61
3.4 Quality control . . . 62
3.5 Construction of *M. tuberculosis* reference graphs . . . 63
3.5.1 Computational performance . . . 64
3.6 Variant calling and filtering assessment . . . 65
3.6.1 Validating variant calls . . . 66
3.6.2 Illumina variant calling . . . 66
3.6.3 Nanopore variant calling: BCFtools . . . 66
3.6.4 Nanopore variant calling: Pandora . . . 67
3.6.5 Computational performance . . . 71
3.6.6 Summary . . . 72
3.7 Pairwise SNP distance comparison . . . 74
3.8 Nanopore transmission clustering . . . 77
3.8.1 Transmission cluster similarity . . . 77
3.8.2 Evaluation of transmission clusters . . . 81
3.8.3 Summary . . . 88
3.9 Mixed Illumina and Nanopore transmission clusters . . . 89
3.9.1 Summary . . . 92
3.10 Discussion . . . 92
3.11 Conclusion . . . 98
3.12 Future work . . . 98
3.12.1 Dataset with known epidemiological information . . . 98
3.12.2 Computational performance of variant calling . . . 98
3.12.3 Improving PanRG construction . . . 99

3.13 Availability of data and materials 100

4 Predicting *M. tuberculosis* drug resistance **101**

4.0 Publication and collaboration acknowledgements 101

4.1 Introduction 101

4.2 Dataset 103

4.2.1 Drug susceptibility testing 103

4.3 Drug resistance prediction with genome graphs 103

4.3.1 Constructing a panel reference graph 105

4.3.2 Predicting resistance phenotype 107

4.3.3 Computational performance 109

4.4 Concordance of Nanopore-based resistance predictions with Illumina 109

4.4.1 Summary 112

4.5 Concordance of genotype-based predictions with culture-based phenotypes 115

4.5.1 Summary 117

4.6 Effect of Nanopore read depth 119

4.7 Detecting off-catalogue mutations 119

4.7.1 Summary 125

4.8 Discussion 125

4.9 Conclusion 129

4.10 Future work 130

4.10.1 Mutation catalogue improvement 130

4.10.2 Low read depth samples 131

4.10.3 Validation on larger datasets 131

4.10.4 Indel calls 131

4.10.5 Lineage classification 132

4.10.6 Other species 132

4.11 Availability of data and materials 133

5 Improving Nanopore sequencing accuracy for *M. tuberculosis* **135**

5.0 Publication and collaboration acknowledgements 135

5.1 Introduction . . . 135
5.2 Dataset . . . 136
5.3 Training an *M. tuberculosis*-specific Nanopore basecalling model . . . 137
5.4 Evaluating a custom Nanopore basecalling model . . . 139
5.4.1 Read-level performance . . . 140
5.4.2 Consensus-level performance . . . 142
5.4.3 Error types . . . 147
5.5 Discussion . . . 149
5.6 Conclusion . . . 152
5.7 Future work . . . 153
5.7.1 *de novo* assembly assessment . . . 153
5.7.2 *M. tuberculosis* methylation sites . . . 153
5.7.3 Impact on previous work . . . 153
5.7.4 Increased diversity of data . . . 154
5.7.5 New Conditional random field models . . . 154
5.8 Availability of data and materials . . . 154

References **157**

Appendix A Supporting work for Chapter 2 **173**
A.1 Length of missing loci in simulations . . . 173
A.2 Constructing a pan-genome SNP truth set . . . 173

Appendix B Supporting work for Chapter 3 **177**
B.1 DNA sequencing methods . . . 177
B.1.1 Isolate selection and DNA extraction . . . 177
B.1.2 PacBio sequencing . . . 178
B.2 Benchmark of long read genome assembly methods for *M. tuberculosis* 179
B.2.1 Assembly Likelihood Evaluation score . . . 180
B.2.2 Disagreement rate . . . 181
B.2.3 Number of contigs . . . 182
B.2.4 Length of assembly . . . 183
B.2.5 Contamination detection . . . 184

B.2.6 Summary 184
B.3 Masking of the *M. tuberculosis* reference graph 185
B.4 Precision and recall of all `pandora` variant calls 186
B.5 Selecting Nanopore SNP thresholds to define transmission clusters . . 186
B.5.1 BCFtools 188
B.5.2 `pandora` single-sample 188
B.5.3 `pandora` multi-sample 188
B.6 Full mixed technology simulation results 188

Appendix C Supporting work for Chapter 4 **195**
C.1 Drug susceptibility testing 195
C.1.1 Madagascar 195
C.1.2 South Africa 195
C.1.3 Full data availability 195
C.2 Constructing a panel reference graph 198
C.2.1 Example panel and associated VCF produced by `drprg` . . . 198
C.2.2 Panel-based PRG density and haplotype problems 198
C.3 Example `drprg` prediction report 204
C.4 Adjustment of default `mykrobe` Nanopore settings 204

Todo

I would like to dedicate this thesis to iii
declaration v
And I would like to acknowledge vii
waiting for email from Anzaan and Tash confirming this 101
need to make tubby public prior to submission 154
I have emailed Tash and Anzaan for this 195

List of figures

1.1 An illustration of the three main mechanisms of horizontal inheritance in bacteria. **a**) Transduction is facilitated by phages that encapsulate host DNA in one cell and eject that DNA into another cell. **b**) Conjugation, in its prevalent form, is the transfer of a plasmid copy from a donor to a receiver cell via a pilus "bridge". In a rarer form, a plasmid that has been incorporated into the donor cell chromosome is transferred. **c**) Transformation (competence) is the uptake of exogenous DNA and subsequent incorporation into the chromosome. 3

1.2 The size and gene frequency distribution of the bacterial pan-genome. **a**) Venn diagram representation of the pan-genome and its core and accessory components. **b**) The asymmetric U-shaped gene frequency distribution for 10 genomes within 6 bacterial species. Genes are generally rare (left) or common (right). **c**) the size (y-axis; number of genes) of the core (red) and accessory (blue; Pan) genome of *E. coli* as more genomes are sampled (x-axis). 5

1.3 An illustration of the single-reference problem. Each vertical column depicts a genome, with the coloured blocks representing loci (genes). The numbers next to each locus indicate a variant with respect to other loci of the same colour. The percentage figures at the bottom indicate what percentage of variants in the other genomes could be found by a perfect variant caller if that genome was used as a reference. Not all genomes contain the same loci; hence no genome can capture all of the variants. 9

1.4 Conceptual representation of a genome graph. **a**) Variants cause a "bubble", or divergent path. Note, the second bubble represents an insertion/deletion. **b**) Variation can be arbitrarily nested. In this example, there is a SNP within an insertion. **c**) Haplotype information can be encoded by "colouring" variants and disallowing paths that mix colours. 10

1.5 Construction of a locus reference graph (PRG) from a multiple sequence alignment (MSA; left) with the recursive cluster and collapse algorithm implemented in `make_prg`. Vertical slices in the MSA are collapsed when there is a minimum match length of 4. Sections not collapsed are recursively clustered and collapsed (if possible), until no further clustering is possible or a maximum nesting level is attained. In this example, a nesting level of 2 is reached. 13

1.6 The impact of reference choice on variant representation. In both **(a)** and **(b)** the left panel (i) shows the PRG. **a)** the blue line indicates the reference sequence. ii and iii show how the choice of this sequence affects the representation of variant sites when genotyping. **b**) the black line indicates the reference sequence, while the blue and orange lines are two different samples. ii and iii show that small differences between the samples are represented as small variants by selecting the sequence that is maximally close to the two samples (iii). 16

1.7 Nanopore sequencing. **a**) A single strand of DNA or RNA (black) is passed through a nanopore (green) by an enzyme (red). A sensor records the electrical current flowing through the nanopore. **b**) The raw signal (electrical current; y-axis) changes as nucleotides move through the nanopore. Each nucleotide can be inferred by a characteristic alteration in the signal, indicated by the black arrows. 17

1.8 Timeline of ONT nanopore chemistry and the accuracy of Nanopore sequencing data. The numbers attached to each point in the accuracy plot relate to published work reporting the accuracy measurement. A list of those publications can be found in the original article this figure was taken from [72]. HMM=hidden Markov model; RNN=recurrent neural network; 2D=a form of Nanopore sequencing where both strands are sequenced. 19

2.1 An illustration of graph pruning. The graph represents a de Bruijn graph of a candidate region. Nodes represent k-mers and are arbitrarily labelled, except a_L and a_R (red), which are the start and end anchor k-mers, respectively. When enumerating paths, we aim to find all paths between a_L and a_R with a length no greater than a specified maximum. Numbers next to nodes indicate the length of the shortest path from that node to the end k-mer a_R. In this example, we set a maximum path length of 4. As such, light blue nodes and dashed edges indicate sections of the graph that would be pruned (not explored). 36

2.2 Recall (x-axis) and precision (y-axis) of *de novo* variants discovered by `pandora` on a simulated dataset. Subplots style the points by the parameter indicated in the subtitle. Each point indicates a single run of `pandora` with a unique combination of all parameters. 41

2.3 The proportion of simulated SNPs that are not detectable by *de novo* variant discovery. The red box represents SNPs that occur in loci designated as absent by `pandora`. The blue box depicts the SNPs that occur within $2k-1$ positions of the start or end of a locus. Each point indicates a single run of `pandora` with a unique combination of parameters. 42

2.4 Precision-recall curve for increasing genotype confidence score thresholds. The curves are coloured by read depth (coverage). Each marker/point is a different genotype confidence threshold, starting with 0 as the right-most value and increasing as the line moves towards the (top) left. Note, the y-axis has been cut to provide more clarity of precision. 43

2.5 Effect of Nanopore methylation-aware basecalling on `pandora` *de novo* variant discovery. The lines show the average allelic recall (AvgAR; y-axis) and error rate ($1-$precision; x-axis) of `pandora` with increasing genotype confidence score thresholds with (solid line) and without (dashed line) *de novo* variant discovery. The red line shows data basecalled with the default `guppy` model and blue being basecalled with a methylation-aware model. 47

2.6 Precision-recall curve for variant calls from Illumina (left; **(a)**) and Nanopore (right; **(b)**) data for different variant calling tools (colours). Each curve represents increasing genotype confidence thresholds, starting with no filtering in the top-right of each curve and increasing towards the bottom-left. `pandora` has a single line with (blue) and without (red) *de novo* variant discovery, which represents the error rate and recall across all 20 samples. The other variant callers are traditional single-reference tools; therefore, each line represents the use of 1/24 reference genomes from across five major *E. coli* phylogroups. Inset windows are used to give more granularity for error rate in the best performing tools. 49

3.1 Precision (left) and recall (right) of SNPs for COMPASS (red) and a selection of `bcftools` filters. `#nofilter` (blue) is `bcftools` with no filtering of variants. `QUAL` (purple) is `bcftools` SNPs with a quality score of 60 or more. `+RPB+VDB+SGB` (grey) indicates `bcftools` variants with the INFO field values ≥ 0.05, ≥ 0.002, and ≤ -0.5, respectively, plus QUAL. `+FRS` (yellow) shows `bcftools` SNPs with all previous filters, plus only SNPs where the fraction of reads supporting the variant is at least 90%. Note: the precision plot y-axis was cut causing some `#nofilter` points to be hidden. 68

3.2 Precision (bottom) and recall (top) of SNPs for COMPASS (purple) and `pandora` with sparse (red) and dense (blue) PanRGs. The `pandora` boxes start with no filters on the left, with each box moving to the right adding a filter to the previous box. The COMPASS box is a reference to the precision and recall of Illumina variant calls. Linear PRG density refers to the fact that COMPASS uses a single, linear reference genome as opposed to `pandora`, which uses a genome graph. The black points refer to single data points for the seven samples used. MIN_COV=minimum depth of coverage;MIN_SB=minimum strand bias;MIN_GT_CONF=minimum genotype confidence score;MIN_FRS=minimum fraction of read support. 70

3.3 The CPU time (in seconds; y-axis; top) and maximum memory usage (in megabytes; y-axis; bottom) for each Nanopore variant-calling job. Sparse (red) and dense (blue) refer to `pandora` steps with the respective density PanRG. `bcftools` (purple) only has one step (`pileup nanopore`). The violins represent the distribution of CPU time over all samples. Note, the y-axis for both plots is log-scaled. 73

3.4 Precision (left) and recall (right) of filtered SNPs for COMPASS (red), `bcftools` (blue), and `pandora` (purple). Each black point represents one of seven evaluation samples. 74

3.5 Pairwise SNP distance relationship between Illumina (COMPASS; x-axis) and Nanopore (`bcftools` (red), and `pandora` single-sample (blue) and multi-sample (purple) mode; y-axis) data. Each point represents the SNP distance between two samples for the two sequencing modalities. The black, dashed line shows the identity line (i.e. $y = x$) and the coloured lines shows the line of best fit based on the robust linear model fit to the data. The zoomed inset shows all pairs where the COMPASS distance is ≤ 100. 76

3.6 Illustrative examples of transmission clustering. **a)** represents truth clusters, while **b)** is clustering from some "test" method we would like to compare to **a**. The nodes represent samples with the numbers on the edges connecting them indicating the distance between those two samples. The red nodes indicate samples with a clustering disparity between the two clusterings. Note, we do not show singletons (disconnected nodes) - e.g., *J* is missing from **(b)**. 80

3.7 Agreement of Illumina and Nanopore transmission clustering at an Illumina SNP threshold of **0**. The expected (Illumina) clusters are shown in the top-left panel. The other panels show the Nanopore-based clustering from `bcftools` (top-right), `pandora map` (bottom-left), and `pandora compare` (bottom-right), with the title indicating the SNP threshold used for clustering. Nodes are coloured and numbered according to their Illumina cluster membership. Samples not clustered (singletons) in Illumina are represented as white boxes with red stripes and are named "S". Clusters are horizontally aligned and connected with black lines. Where a sample that Illumina clustered is *not* clustered by Nanopore, the sample retains its original colour and number but is represented as an unconnected node on the top row of the plot. Each Nanopore panel has a legend showing the SACR, SACP, and XCR value with respect to the Illumina clustering. SACR=sample-averaged cluster recall; SACP=sample-averaged cluster precision; XCR=excess clustering rate. 83

3.8 Agreement of Illumina and Nanopore transmission clustering at an Illumina SNP threshold of **2**. The expected (Illumina) clusters are shown in the top-left panel. The other panels show the Nanopore-based clustering from `bcftools` (top-right), `pandora map` (bottom-left), and `pandora compare` (bottom-right), with the title indicating the SNP threshold used for clustering. Nodes are coloured and numbered according to their Illumina cluster membership. Samples not clustered (singletons) in Illumina are represented as white boxes with red stripes and are named "S". Clusters are horizontally aligned and connected with black lines. Where a sample that Illumina clustered is *not* clustered by Nanopore, the sample retains its original colour and number but is represented as an unconnected node on the top row of the plot. Each Nanopore panel has a legend showing the SACR, SACP, and XCR value with respect to the Illumina clustering. SACR=sample-averaged cluster recall; SACP=sample-averaged cluster precision; XCR=excess clustering rate. 84

3.9 Agreement of Illumina and Nanopore transmission clustering at an Illumina SNP threshold of **5**. The expected (Illumina) clusters are shown in the top-left panel. The other panels show the Nanopore-based clustering from `bcftools` (top-right), `pandora map` (bottom-left), and `pandora compare` (bottom-right), with the title indicating the SNP threshold used for clustering. Nodes are coloured and numbered according to their Illumina cluster membership. Samples not clustered (singletons) in Illumina are represented as white boxes with red stripes and are named "S". Clusters are horizontally aligned and connected with black lines. Where a sample that Illumina clustered is *not* clustered by Nanopore, the sample retains its original colour and number but is represented as an unconnected node on the top row of the plot. Each Nanopore panel has a legend showing the SACR, SACP, and XCR value with respect to the Illumina clustering. SACR=sample-averaged cluster recall; SACP=sample-averaged cluster precision; XCR=excess clustering rate. 85

3.10 Agreement of Illumina and Nanopore transmission clustering at an Illumina SNP threshold of **12**. The expected (Illumina) clusters are shown in the top-left panel. The other panels show the Nanopore-based clustering from `bcftools` (top-right), `pandora map` (bottom-left), and `pandora compare` (bottom-right), with the title indicating the SNP threshold used for clustering. Nodes are coloured and numbered according to their Illumina cluster membership. Samples not clustered (singletons) in Illumina are represented as white boxes with red stripes and are named "S". Clusters are horizontally aligned and connected with black lines. Where a sample that Illumina clustered is *not* clustered by Nanopore, the sample retains its original colour and number but is represented as an unconnected node on the top row of the plot. Each Nanopore panel has a legend showing the SACR, SACP, and XCR value with respect to the Illumina clustering. SACR=sample-averaged cluster recall; SACP=sample-averaged cluster precision; XCR=excess clustering rate. 86

3.11 Mixed modality "self-distance". This plot shows the SNP distance (x-axis) between each sample's COMPASS (Illumina) and `bcftools` (Nanopore) VCF calls. 90

3.12 The relationship of the distance between all pairs of samples based on Illumina (COMPASS) VCF calls (X-axis) and mixed COMPASS-`bcftools` calls (Y-axis). The black, dashed line indicates the relationship we would expect if the distance between a pair of samples were the same for both approaches. The blue line indicates the line of best fit based on fitting a robust linear regression model to the data. The inset gives a closer look at the relationship for all sample pairs where the COMPASS distance is less than or equal to 100 SNPs. The legend indicates the linear equations for the lines. Note: to prevent model skew, we do not include self-distance pairs. 91

3.13 Simulating various ratios (x-axis) of Nanopore/Illumina sample mixtures. The different thresholds (subplots) indicate the cutoff for defining samples as part of a cluster. The y-axis depicts the Sample-Averaged Cluster Precision and Recall (SACP/SACR) and Excess Clustering Rate (XCR) distributions over all simulation runs (XCR is shown as (1-XCR) for better axis-scaling). For each ratio/threshold combination we run 1000 simulations where the Nanopore and Illumina data is randomly split into the relevant ratio and clusters are defined based on the relevant threshold. The titles for each subplot indicate the SNP threshold used when comparing Illumina (Ill.), Nanopore (NP), or mixed-technology sample pairs. 93

4.1 Culture-based drug susceptibility data available for samples. Each row is a drug, and the columns represent a set of samples with phenotype information for those drugs with a filled cell. The top panel shows the number of samples in the set for that combination of drugs. The bar plot in the left panel shows the number of samples with phenotype information for that drug. 104

4.2 The time (seconds; top) and memory (megabytes; bottom) usage of `mykrobe` and `drprg` for generating drug resistance predictions. Each tool is additionally split into sequencing technology, with Illumina in red and Nanopore in blue. Data points indicate individual results for a single sample. 110

4.3 Number of resistant (left) and susceptible (right) WGS-based drug resistance phenotypes correctly predicted by `mykrobe` Nanopore (purple) and `drprg` (blue) with Illumina (striped) and Nanopore (non-striped) data. For this analysis, the Illumina-based `mykrobe` predictions are considered truth and the other predictions are assessed accordingly. The red bars indicate missed (FN) or incorrect (FP) predictions. The x-axis shows the drugs for which `mykrobe` makes predictions. E - ethambutol; H - isoniazid; Z - pyrazinamide; R - rifampicin; S - streptomycin; Km - kanamycin; Am - amikacin; Ofx - ofloxacin; Cm - capreomycin; Mfx - moxifloxacin. 113

4.4 Number of resistant (left) and susceptible (right) culture-based drug susceptibility testing (DST) phenotypes correctly identified by `mykrobe` (purple) and `drprg` (blue) with Illumina (non-striped) and Nanopore (striped) data. The red bars indicate missed (FN) or incorrect (FP) predictions. The x-axis shows the drugs with available phenotype data. E - ethambutol; H - isoniazid; R - rifampicin; S - streptomycin; Km - kanamycin; Am - amikacin; Ofx - ofloxacin; Cm - capreomycin. . . . 116

4.5 Effect of Nanopore read depth on `drprg` (top) and `mykrobe` (bottom) drug susceptibility phenotype prediction. Each point indicates the proportion (left y-axis) of classifications of that type at the read depth (x-axis). Read depth is binned such that 40 is all samples with a read depth ≥ 40 and less than 50. The blue bars indicate the number of samples (right y-axis) contained in each bin. FP=false positive (red); TN=true negative (grey); FN=false negative (purple); TP=true positive (blue). 120

4.6 Classifications of novel SNP calls made from Illumina (left) and Nanopore (right) data by `drprg` in resistance-associated genes (y-axis). Counts (x-axis) are across all samples. TP (blue) - true positive; FN (red) - false negative; FP (purple) - false positive. 121

4.7 Number of novel variants found in resistance-associated genes for each sample. The x-axis indicates the number of classifications of the relevant kind for a sample, and the y-axis is the number of samples with that classification count. The classifications are: TP (blue) - true positive; FN (red) - false negative; FP (purple) - false positive. . . . 123

4.8 Number of resistant (left) and susceptible (right) culture-based drug susceptibility testing (DST) phenotypes correctly identified by `mykrobe` (purple) and `drprg` (blue) with Illumina (non-striped) and Nanopore (striped) data. The red bars indicate missed (FN) or incorrect (FP) predictions. The yellow bars represent `drprg` "unknown" calls, which is when there is an off-panel variant found in a gene associated with the respective drug. If there is no resistance-associated variant found for that drug, then an unknown call is made. The x-axis shows the drugs with available phenotype data. E - ethambutol; H - isoniazid; R - rifampicin; S - streptomycin; Km - kanamycin; Am - amikacin; Ofx - ofloxacin; Cm - capreomycin. 124

5.1 Runtimes (x-axis) of the different stages (rules; y-axis) of preparing data for basecall model training (red) and training the model itself (blue). Each individual run is represented with a black point, and rules that have multiple runs have black 95% confidence interval bars. s=seconds; m=minutes; h=hours; d=days; w=weeks. 138

5.2 Read BLAST identity (x-axis) for the *M. tuberculosis*-specific basecalling model `tubby` (red) compared with the default `guppy` model (blue). The subtitle of each plot indicates the data being assessed. The validation data (**a**) refers to *M. tuberculosis* reads that were not used in (`tubby`) model training, while test data (**b**) is reads from a BCG sample that was not involved in training. Version (y-axis) indicates the `guppy` version used for the basecalling before and after training. BLAST identity is the number of matching bases in a read alignment divided by the length of the alignment. The dashed lines represent the 25, 50, and 75^{th} percentiles. 141

5.3 Chromosome identity (x-axis) for the *M. tuberculosis*-specific basecalling model `tubby` (red) compared with the default `guppy` model (blue) on the validation data - *M. tuberculosis* reads not used for training `tubby`. Version (y-axis) indicates the `guppy` version used for the basecalling before and after training. For each chromosome's longest alignment to its truth assembly, chromosome identity is the number of matching bases divided by the alignment length. The coloured points indicate the individual chromosome identity for each contig in each sample. 143

5.4 Consensus BLAST identity (Y-axis), where consensus refers to 10kbp "chunks" of the genome assembly produced by the basecalled reads for each model (colours), mapped to the truth genome. The subtitle of each plot indicates the data being assessed. The validation data (**a**) refers to *M. tuberculosis* reads that were not used in (`tubby`) model training, while test data (**b**) is reads from a BCG sample that was not involved in training. Version (y-axis) indicates the `guppy` version used for the basecalling before and after training. BLAST identity is the number of matching bases (in a chunk alignment) divided by the alignment length. The median value for each boxplot is annotated on the middle line. 145

5.5 Error types in the `rebaler` assemblies produced from `tubby` and default `guppy` basecalling models. The consensus error rate is the number of errors attributable to a given type, divided by the length of the truth assembly. The errors are per assembly, so the confidence intervals represent variation in error types between samples/assemblies, with the points showing each assembly's value. `dcm` (red) refers to errors in Dcm-methylation motifs. `homo` refers to homopolymer insertions (purple) or deletions (blue). `other` refers to non-homopolymer insertions (yellow) or deletions (grey). `sub` (green) is substitutions (single- or multi-base). The subtitle of each plot indicates the data being assessed. The validation data (**a**) refers to *M. tuberculosis* reads that were not used in (`tubby`) model training, while test data (**b**) is reads from a BCG sample that was not involved in training. Note, the y-axes of **(a)** and **(b)** are different. 148

5.6 Impact of error rate on expected *k*-mer depth. The y-axis shows the expected proportion of *k*-mers with no sequencing errors for a given basecalling model (colours) with error rate, *e*. 150

A.1 The lengths (x-axis; base pairs) of missing loci in the *de novo* simulations in Section 2.3 174

B.1 The normalised assembly likelihood evaluation (ALE) score (Y-axis) for each sample (X-axis), coloured by assembly tools. ALE score is a metric describing the likelihood of an assembly. The normalisation is done by subtracting the assembly's score from the maximum (best) score for that sample, giving a relative probability of correctness. Each box represents different technologies and polishing status for each assembler. 181

B.2 The disagreement rate (y-axis) for each assembler (x-axis), coloured by the sequencing technology. Disagreement rate is the percentage of sites in the assembly where Illumina reads do not have at least 90% quorum. Each box/point represents different samples and polished status for the relevant assembler/technology combination. 182

B.3 The number of contigs (Y-axis) produced from each assembly (X-axis), coloured by sequencing technology. Each box/point represents different samples and polished status for the relevant assembler-technology combination. 183

B.4 Size/length (Y-axis; in base-pairs (bp)) of each assembly (X-axis), coloured by each sequencing technology. The horizontal dashed line represents the size of the *M. tuberculosis* reference genome (4,411,532bp). Each box/point represents different samples and polished status for the relevant assembler-technology combination. Note: the Y-axis has been limited to allow for greater resolution of similarity to the H37Rv size 184

B.5 **a)** Proportion of loci lost (y-axis) when removing those with a certain fraction (x-axis) of their positions within the genome mask. **b)** Proportion of total genome positions lost (y-axis) when removing loci with a certain fraction (x-axis) of their positions within the genome mask. . . 186

B.6 Precision (bottom) and recall (top) of SNPs for COMPASS (purple) and all variants (maximum indel length of 20bp) for `pandora` with sparse (red) and dense (blue) PRGs. The `pandora` boxes start with no filters on the left, with each box moving to the right adding a filter to the previous box. The COMPASS box is a reference to the precision and recall of Illumina variant calls. Linear PRG density refers to the fact that COMPASS uses a single, linear reference genome as opposed to `pandora`, which uses a genome graph. The black points refer to single data points for the seven samples used. MIN_COV=minimum depth of coverage; MIN_SB=minimum strand bias; MIN_GT_CONF=minimum genotype confidence score; MIN_FRS=minimum fraction of read support. 187

B.7 Illumina and Nanopore (`bcftools`) transmission cluster similarity for various SNP distance threshold. Each subplot compares the Nanopore clustering for the threshold on the x-axis to the Illumina clustering based on the distance (threshold) in the subplot title. SACR (red), SACP (blue), and $1-$XCR are represented by the lines with the band around the line indicating the 95% confidence interval. The red, vertical, dashed lines indicate the model-based prediction of what the Nanopore SNP distance threshold should be based on the Illumina distance for that subplot. 189

B.8 Illumina and Nanopore (`pandora` single-sample) transmission cluster similarity for various SNP distance threshold. Each subplot compares the Nanopore clustering for the threshold on the x-axis to the Illumina clustering based on the distance (threshold) in the subplot title. SACR (red), SACP (blue), and $1-$XCR are represented by the lines with the band around the line indicating the 95% confidence interval. The red, vertical, dashed lines indicate the model-based prediction of what the Nanopore SNP distance threshold should be based on the Illumina distance for that subplot. 190

B.9 Illumina and Nanopore (`pandora` multi-sample) transmission cluster similarity for various SNP distance threshold. Each subplot compares the Nanopore clustering for the threshold on the x-axis to the Illumina clustering based on the distance (threshold) in the subplot title. SACR (red), SACP (blue), and 1−XCR are represented by the lines with the band around the line indicating the 95% confidence interval. The red, vertical, dashed lines indicate the model-based prediction of what the Nanopore SNP distance threshold should be based on the Illumina distance for that subplot. 191

C.1 Culture-based and line probe assay (LPA) drug susceptibility data available for samples. Each row is a drug, and the columns represent a set of samples that have phenotype information for those drugs with a filled cell. The top panel shows the number of samples in the set for that combination of drugs. The bar plot in the left panel shows the number of samples with phenotype information for that drug. 197

C.2 An example panel (top) required by `drprg`. The columns indicate the gene, mutation, residue the variant describes, and drug(s) the entry impacts. The panel is turned into a VCF (bottom) with proteins converted to DNA. Associated information from the panel and annotation are encoded in the `INFO` field for each entry. Note: some unnecessary information for this example is removed from the VCF entry shown here to reduce the size. 199

C.3 An example of how the `make_prg` minimum match length (m) parameter effects PRG structure. The PRGs for three sequences (top) with three positions of difference (lower-case letters) are shown. m controls when sequence is collapsed. When $m = 3$ (middle), all runs of `A` and `T` are collapsed because at least three continuous bases agree between the three sequences. However, when $m = 4$ (bottom), the `A`s are no longer collapsed, leaving a single, longer branched path for each sequence, rather than the three smaller, single-base paths when $m = 3$. $*$ is a placeholder to indicate an insertion/deletion. 200

C.4 Contrasting examples of the same *pncA* variant site (S65F) from a panel-based PRG (top) and a population-based PRG (bottom). `VARID` indicates the panel variants this site overlaps, and `PREDICT` is the resistance prediction for the relevant `VARID`. `MEAN_FWD_COVG` and `MEAN_REV_COVG` specify the mean forward and reverse *k*-mer coverage on minimizer *k*-mers that overlap this site. Note: due to a large number of alternate alleles (126) in the panel-based record and overlapping `VARID`s, some data has been elided for illustrative purposes. 201

C.5 Contrasting examples of the same *gyrA* variant site from a panel-based PRG (top) and a population-based PRG (bottom). `VARID` indicates the panel variants this site overlaps and `PREDICT` is the resistance prediction for the relevant `VARID`. `MEAN_FWD_COVG` and `MEAN_REV_COVG` specify the mean forward and reverse *k*-mer coverage on minimizer *k*-mers that overlap this site. 203

C.6 An example drug susceptibility JSON report file produced by `drprg`. The report maps a drug to its prediction and any relevant evidence supporting that prediction. Note, evidence for susceptibility (`"S"`) predictions is not provided as susceptibility is assumed where no `R/U` prediction is made. S=susceptible; U=unknown (i.e., novel variant); R=resistant. 205

C.7 Number of resistant (left) and susceptible (right) culture-based drug susceptibility phenotypes correctly identified by `mykrobe` with default (blue) or adjusted (purple) settings. Nanopore data is indicated by diagonal stripes, with Illumina having no stripes. The red bars indicate missed (FN) or incorrect (FP) predictions. The x-axis shows the drugs with available phenotype data. E - ethambutol; H - isoniazid; R - rifampicin; S - streptomycin; Km - kanamycin; Am - amikacin; Ofx - ofloxacin; Cm - capreomycin. 206

C.8 Number of resistant (left) and susceptible (right) Illumina WGS-based drug susceptibility phenotypes correctly identified using Nanopore data. Nanopore predictions for `mykrobe` with default (blue) and adjusted (purple) settings are compared to those from Illumina with the same settings. The red bars indicate missed (FN) or incorrect (FP) predictions. The x-axis shows the drugs with available phenotype data. E - ethambutol; H - isoniazid; Z - pyrazinamide; R - rifampicin; S - streptomycin; Km - kanamycin; Am - amikacin; Ofx - ofloxacin; Cm - capreomycin; Mfx - moxifloxacin. 209

List of tables

2.1 The median precision and recall for all parameter combinations, grouping by the maximum PRG nesting level or the *de novo* *k*-mer size used for variant discovery in `pandora`. The values in parentheses indicate the parameter value that leads to the specified precision or recall. . . . 41

2.2 Effect of Nanopore methylation-aware basecalling on `pandora` *de novo* variant discovery (unfiltered) error rate (1−precision) and average allelic recall (AvgAR). 48

3.1 Computational time and memory (RAM) usage for the main steps of building a *M. tuberculosis* reference graph. Sparse and Dense refer to two different densities with respect to the number of variants used. All steps were run on a single compute node with 32 CPU cores. MSA=multiple sequence alignment;PRG=population reference graph. 65

3.2 CPU and wall clock time, and memory (RAM) usage for the main steps of running `pandora`'s multi-sample routine `compare`. Sparse and Dense refer to two different densities with respect to the number of variants used. All steps were run on a single compute node with 32 CPU cores. MSA=multiple sequence alignment; PRG=population reference graph. 72

3.3 Cluster recall and precision results for each sample in Figure 3.6. . . . 81

3.4 Summary of `bcftools` clustering metrics for four (Illumina) SNP distance thresholds. The threshold(s) in parentheses are the Nanopore equivalent threshold used. The fractions in parentheses for XCR indicate the underlying numbers. SACR=sample-averaged cluster recall; SACP=sample-averaged cluster precision; XCR=excess clustering rate. 82

3.5 Summary of `pandora` single-sample clustering metrics for four (Illumina) SNP distance thresholds. The threshold(s) in parentheses are the Nanopore equivalent threshold used. The fractions in parentheses for XCR indicate the underlying numbers. SACR=sample-averaged cluster recall; SACP=sample-averaged cluster precision; XCR=excess clustering rate. 87

3.6 Summary of `pandora` multi-sample clustering metrics for four (Illumina) SNP distance thresholds. The threshold(s) in parentheses are the Nanopore equivalent threshold used. The fractions in parentheses for XCR indicate the underlying numbers. SACR=sample-averaged cluster recall; SACP=sample-averaged cluster precision; XCR=excess clustering rate. 88

4.1 Culture-based drug susceptibility data available for samples. The counts are the number of samples with phenotype information available for that drug. 105

4.2 Summary statistics for the CPU time and peak memory usage of drug resistance prediction with `mykrobe` and `drprg` for Illumina and Nanopore data. std=standard deviation 110

4.3 Comparison of `drprg` and `mykrobe` Nanopore drug resistance predictions with `mykrobe` Illumina predictions. For this comparison, we assume the `mykrobe` resistance prediction from Illumina data is correct and evaluate the other predictions accordingly. Bold text is used to highlight differences of note. FN=false negative; R=number of resistant samples; FP=false positive; S=number of susceptible samples; FNR=false negative rate; FPR=false positive rate; PPV=positive predictive value; NPV=negative predictive value; CI=Wilson score confidence interval 114

4.4 Comparison of WGS-based drug resistance prediction concordance with culture-based drug susceptibility testing (DST) phenotypes. For this comparison, we assume the DST phenotype is correct and evaluate the WGS predictions accordingly. `drprg` and `mykrobe` are the two tools that provide WGS predictions. Bold text is used to highlight differences of note between `drprg` and `mykrobe`. FN=false negative; R=number of resistant samples; FP=false positive; S=number of susceptible samples; FNR=false negative rate; FPR=false positive rate; PPV=positive predictive value; NPV=negative predictive value; CI=Wilson score confidence interval 118

4.5 Classification of novel SNP calls made by `drprg` in resistance-associated genes. Counts are across all samples. TP - true positive; FN - false negative; FP - false positive. 122

5.1 Read BLAST identity summary statistics for the *M. tuberculosis*-specific basecalling model `tubby` compared with the default `guppy` model on the validation data - *M. tuberculosis* reads not used for training `tubby`. Version indicates the `guppy` version used for the basecalling before and after training. BLAST identity is the number of matching bases in a read alignment divided by the length of the alignment. Count refers to the number of reads evaluated. Bold text highlights the best-performing model for the relevant metric. std=standard deviation. 140

5.2 Read BLAST identity summary statistics for the *M. tuberculosis*-specific basecalling model `tubby` compared with the default `guppy` model on the test data - a BCG sample not involved in training `tubby`. Version indicates the `guppy` version used for the basecalling before and after training. BLAST identity is the number of matching bases in a read alignment divided by the length of the alignment. Count refers to the number of reads evaluated. Bold text highlights the best-performing model for the relevant metric. std=standard deviation. 142

5.3 Chromosome identity summary statistics for the *M. tuberculosis*-specific basecalling model `tubby` compared with the default `guppy` model on the validation data - *M. tuberculosis* reads not used for training `tubby`. Version indicates the `guppy` version used for the basecalling before and after training. For each chromosome's longest alignment to its truth assembly, chromosome identity is the number of matching bases divided by the alignment length. Bold text highlights the best-performing model for the relevant metric. std=standard deviation. 144

5.4 Consensus BLAST identity summary statistics for the validation data - *M. tuberculosis* reads not used for training `tubby`. Where consensus refers to 10kbp "chunks" of the genome assembly produced by the basecalled reads, for each model, mapped to the truth genome. Version indicates the `guppy` version used for the basecalling before and after training. BLAST identity is the number of matching bases (in a chunk alignment) divided by the length of the alignment. Count refers to the number of consensus chunks assessed. Bold text highlights the best-performing model for the relevant metric. std=standard deviation. 144

5.5 Chromosome identity for the *M. tuberculosis*-specific basecalling model `tubby` compared with the default `guppy` model on the BCG test sample. Version indicates the `guppy` version used for the basecalling before and after training. For each chromosome's longest alignment to its truth assembly, chromosome identity is the number of matching bases divided by the alignment length. Bold text highlights the best-performing model's chromosome identity. 146

5.6 Consensus BLAST identity summary statistics for the BCG test sample. Where consensus refers to 10kbp "chunks" of the genome assembly produced by the basecalled reads, for each model, mapped to the truth genome. Version indicates the `guppy` version used for the basecalling before and after training. BLAST identity is the number of matching bases (in a chunk alignment) divided by the length of the alignment. Count refers to the number of consensus chunks assessed. Bold text highlights the best-performing model for the relevant metric. std=standard deviation. 146

B.1 Summary statistics from the simulations of various sequencing technology ratios and the assessment of transmission clusters produced at different SNP thresholds in Section 3.9. SACR=sample-averaged cluster recall; SACP=sample-average cluster precision; XCR=excess clustering rate; std=standard deviation. 194

C.1 Culture-based and line probe assay (LPA) drug susceptibility data available for samples. The counts are the number of samples with phenotype information available for that drug. 196

C.2 Comparison of WGS drug resistance predictions with culture-based phenotype. For this comparison, we assume the culture-based phenotype is correct and evaluate `mykrobe` with default and adjusted settings for Illumina and Nanopore resistance predictions accordingly. Pyrazinamide and Moxifloxacin are excluded as phenotype information is only available for one sample. Bold text is used to highlight differences of note. FN=false negative; R=number of resistant samples; FP=false positive; S=number of susceptible samples; FNR=false negative rate; FPR=false positive rate; PPV=positive predictive value; NPV=negative predictive value; CI=Wilson score confidence interval . 207

C.3 Comparison of Nanopore drug resistance predictions concordance with Illumina predictions. For this comparison, we assume the `mykrobe` resistance prediction from Illumina data is correct and evaluate the Nanopore prediction accordingly. `mykrobe`-default and `mykrobe` indicates `mykrobe` with default or adjusted settings, respectively. Bold text is used to highlight differences of note. FN=false negative; R=number of resistant samples; FP=false positive; S=number of susceptible samples; FNR=false negative rate; FPR=false positive rate; PPV=positive predictive value; NPV=negative predictive value; CI=Wilson score confidence interval 208

Abbreviations

Acronyms / Abbreviations

AMR Antimicrobial resistance

AvgAR Average allelic recall

bp Base pairs - used as a way of measuring the number of nucleotides

CRyPTIC Comprehensive Resistance Prediction for Tuberculosis: an International Consortium

dBG de Bruijn graph

DST Drug susceptibility testing

FN False negative

FP False positive

FRS Fraction of read support

H37Rv The *M. tuberculosis* strain with the best characterised reference genome. Interchangeably use to indicate the genome assembly.

IGR Intergenic region

indel Insertion/deletion

LPA Line probe assay

MDR Multi-drug resistant

ML Maximum likelihood

MSA Multiple sequence alignment

Abbreviations

ONT Oxford Nanopore Technologies

PanRG Pan-genome Reference Graph. A collection of PRGs

PHE Public Health England

PRG Population Reference Graph

QC Quality control

SNP Single Nucleotide Polymorphism

STEC Shiga toxin-producing *E. coli*

TB Tuberculosis

TN True negative

TP True positive

VCF Variant call format

WGS Whole-genome sequencing

WHO World Health Organization

Chapter 1

Background

This thesis examines graph genome methods and their application to bacterial genomes. In particular, we focus our attention on *Mycobacterium tuberculosis* and Nanopore sequencing for these applications.

We begin with the concept of a bacterial pan-genome and discuss the limitations of current approaches for its interrogation. Next, we introduce graph genomes as a more intuitive model for investigating the bacterial pan-genome. We then provide more detail on a particular genome graph method, Pandora, a focal point in the ensuing chapters. A survey of nanopore-based sequencing is then provided, motivating its use as the primary source of genomic data in this thesis. Finally, we introduce tuberculosis (TB) and its causative pathogen, *M. tuberculosis*. We outline the global effort to end TB and discuss the diagnostic challenges we address in this thesis.

1.1 Bacterial genomes

1.1.1 Drivers of bacterial diversity

Bacteria have incredibly flexible genomes. The mechanisms that generate genetic diversity can be grouped by how they are passed on between cells. *Vertical* inheritance is the passing of genetic material from parent to offspring during replication, while *horizontal* inheritance describes the acquisition of genetic information between unrelated cells.

Vertically inherited variation generally consists of genetic point mutations, insertions and deletions (indels), and structural rearrangements. Such mutations can arise due to a multitude of reasons: homologous recombination, DNA deamination,

replication-transcript conflict, and replication errors, to name a few [1]. However, the dominant form of genomic variability in bacteria is horizontal inheritance [2].

Horizontal inheritance operates via the three main mechanisms *transduction*, *conjugation*, and *transformation*. These are illustrated in Figure 1.1.

Transduction is mediated by bacteriophages (phages) - viruses which infect bacteria and are the most abundant organism on Earth [3]. During phage propagation, parts of the host (bacteria) DNA can become encapsulated in the virus. If a phage goes on to infect another cell and eject this encapsulated DNA (transduction), it can recombine into the chromosome or begin replicating as a plasmid [4]. When the transduced DNA is a gene that imparts a new (beneficial or deleterious) function, it can impact the recipient's evolution.

Conjugation is the cell-to-cell transfer of DNA. A donor cell contacts another via a pilus, and a copy of the DNA - usually a plasmid - is transmitted [5]. A rarer form of conjugation can occur when a plasmid has become incorporated into the chromosome of the donor, and this portion of the chromosome is transferred to the recipient cell [6].

Transformation occurs when a cell internalises exogenous DNA, which in turn becomes incorporated into the chromosome by homologous recombination [7]. There is some debate about the exact purpose of transformation, with the consensus being it increases genetic diversity; however, a nutritional role is also a possibility [7].

These different means of inheritance compound to create varying diversity levels within bacterial species and give rise to the *pan-genome*.

1.1.2 The pan-genome

A pan-genome is the full complement of genetic loci found within a given species. Traditionally, loci refer to genes, although we note that loci need not be genes for the work we will describe in this thesis.

The pan-genome can be broken into two subsets: the *core* and *accessory* genome. Loci that occur in the majority of species members are considered core, whilst everything else is deemed accessory (see Figure 1.2a). The accessory genome can be further divided into intermediate and rare loci.

The proportional size of the core genome varies dramatically between species. For instance, if we assume a gene is core when present in $\geq 95\%$ of a sampled species, the *Escherichia coli* pan-genome is composed of 10% core genes. Conversely, 89% of the *Mycobacterium tuberculosis* pan-genome is core genes (data was obtained from

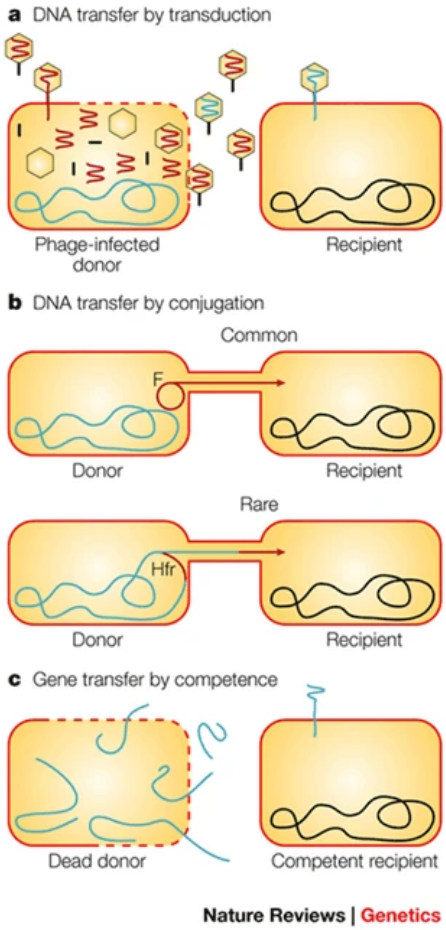


Fig. 1.1: An illustration of the three main mechanisms of horizontal inheritance in bacteria. **a**) Transduction is facilitated by phages that encapsulate host DNA in one cell and eject that DNA into another cell. **b**) Conjugation, in its prevalent form, is the transfer of a plasmid copy from a donor to a receiver cell via a pilus "bridge". In a rarer form, a plasmid that has been incorporated into the donor cell chromosome is transferred. **c**) Transformation (competence) is the uptake of exogenous DNA and subsequent incorporation into the chromosome.

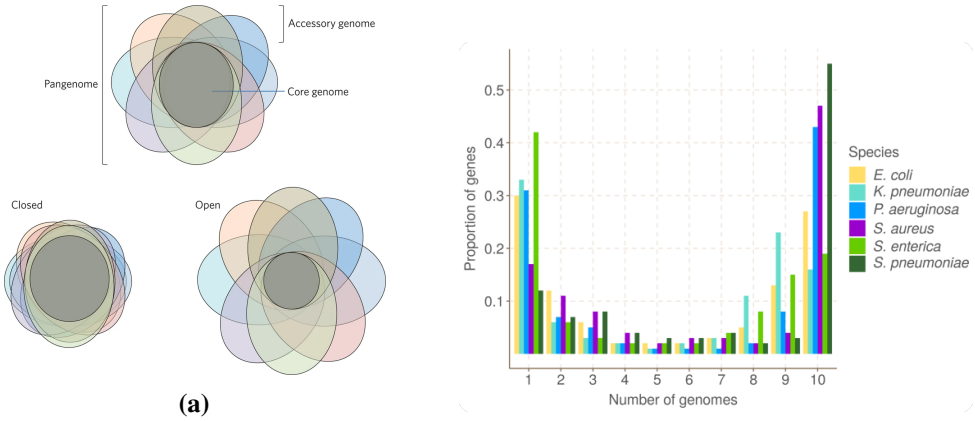
the panX database [8]). Species with a large (ever-expanding) pan-genome, such as *E. coli*, have what is called an "open" pan-genome, while those with more conserved gene content, such as *M. tuberculosis*, are deemed "closed".

Another interesting property of the bacterial genome is the distinctive "U-shaped" gene frequency distribution [9–11], shown in Figure 1.2b. This frequency distribution is a consequence of the fact that, in general, genes are either rare or common within a pan-genome [9, 12]. Moreover, the size of the bacterial pan-genome is estimated to be infinite [11], as hinted at by Figure 1.2c.

These definitions of the pan-genome components (core and accessory) are somewhat simplistic. Recent work by Horesh *et al.* has highlighted that these traditional definitions are biased by lineage sampling [14]. For example, we have a collection of 100 genomes, with 50 being from the same lineage (L_1). Let us say gene *abc* occurs in all 50 members of L_1, but none of the other 50 genomes. Under the traditional pan-genomic definitions, we would call *abc* an intermediate accessory gene. However, if we gathered a further 1,000 genomes, none of which are lineage L_1, *abc* would now be considered rare. In the new pan-genome model proposed by Horesh *et al.*, loci are given a classification that is population-structure aware. The *abc* gene from our example would be classified as "lineage-specific core", acknowledging the fact that the sampled lineages in a collection provide essential contextual information. Other categories include multi-lineage and collection core and the same categories for intermediate, rare, and a new varied frequency class. The collection core is analogous to the traditional core, with everything else being the accessory genome - albeit with a much finer level of detail.

1.1.3 How are pan-genomes analysed?

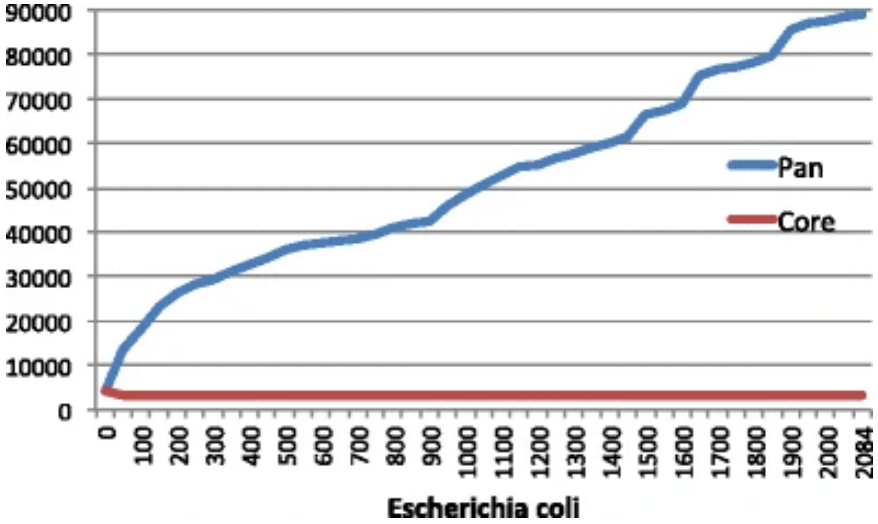
Most genomic analyses of bacterial collections follow a similar approach: extract core gene alignments and typically ignore the accessory, except where investigations focus on a small subset of genes related to phenotypes [15, 16]. Current pan-genomic analyses use a similar principle of core alignment (e.g., Roary [17], Panaroo [18]) to identify core and accessory but lack detail when describing the accessory, which is usually presented as a presence-absence matrix [12, 19–21]. However, as we have seen, the pan-genome size varies significantly, so depending on the species, such approaches could be "ignoring" large portions of the total genomic repertoire. Despite this, we have learned an enormous amount about bacteria and their pan-genomes with these methods.



(a) (b)

Source: Reprinted by permission from Springer Nature Customer Service Centre GmbH: Springer Nature Nature Microbiology [2], Copyright © 2017, Macmillan Publishers Limited.

Source: [10] under the terms of the Creative Commons CC BY license



(c)

Source: [13] under the terms of the Creative Commons CC BY license

Fig. 1.2: The size and gene frequency distribution of the bacterial pan-genome. **a**) Venn diagram representation of the pan-genome and its core and accessory components. **b**) The asymmetric U-shaped gene frequency distribution for 10 genomes within 6 bacterial species. Genes are generally rare (left) or common (right). **c**) the size (y-axis; number of genes) of the core (red) and accessory (blue; Pan) genome of *E. coli* as more genomes are sampled (x-axis).

1.1.4 Variant calling of bacterial genomes

A typical (pan-)genomic analysis requirement, and a focus of this thesis, is variant calling. However, depending on the application, this can be done in many different ways. For example, when characterising an outbreak, common approaches are to use a reference genome of the same, or very close, strain to the outbreak [22], or assemble each sample and select the closest reference to it based on some typing strategy [23]. Alternatively, a reference-free approach can alleviate some of the reference bias induced when selecting a genome to call variants against and provide better resolution of an outbreak [24].

Given the importance of bacterial variant calling to this thesis, we will briefly outline various approaches to calling variants in bacterial genomes and highlight their strengths and limitations.

Alignment-based methods

Alignment-based variant calling requires a reference genome. In this mode of variant calling, raw sequencing reads are aligned to a given reference genome to generate a Sequence Alignment/Map (SAM) file. Common software programs used to perform these alignments for bacterial variant calling include BWA-MEM [25], Bowtie2 [26], and Novoalign (http://www.novocraft.com/products/novoalign) for short (Illumina) read technology. (Nanopore-based variant calling will be detailed in Section 1.4.2, for now we focus on Illumina-based sequencing reads).

Where variant calling programs distinguish themselves is in how they handle the alignment information. This includes, but is not limited to, the number of base calls disagreeing with the reference, the quality of the read alignment, the alignment locations of a read pair, or the quality score of the mapping [27]. Popular methods for calling variants generally employ either Bayesian, likelihood, or machine learning algorithms to infer candidate variants given this alignment data. While many of these models were designed for human variant calling, a selection have shown themselves to be perfectly applicable to bacteria. The most frequently used Bayesian method for bacterial variant calling is Freebayes [28]; however, it is generally used via a wrapper, Snippy (https://github.com/tseemann/snippy), which handles the alignment (BWA-MEM), variant calling (Freebayes), and additionally applies filters to the resulting VCF file. Of the likelihood-based callers, Samtools/BCFtools [29, 30] and GATK [31] tend to be most often employed.

Alignment-free methods

In general, methods that do not align reads to a reference genome use *k*-mer-based algorithms for variant inference. FastGT [32] and LAVA [33] are two such programs that require a database of known variants and use *k*-mer counts in a sample to determine the presence of any of these variants. The major limitation with these tools is their inability to call variants not present in the provided database. Kestrel [34] is a *k*-mer-based variant caller that can *de novo* discover variants and does this by detecting unique *k*-mers in a sample, with respect to a given reference genome. However, Kestrel is not strictly alignment-free, as it does use local alignment to place candidate variants in relation to the reference genome. Additionally, it was shown to have much lower sensitivity than alignment-based methods.

Another popular alignment-free single nucleotide polymorphism (SNP) caller is kSNP, which finds SNPs *between* samples by detecting *k*-mers where the central base varies [35]. It is regularly used in outbreak settings where differences between samples are crucial [36–38]. However, kSNP cannot detect SNPs within *k* positions (bases) of each other or detect indels, and cannot deal with sequencing errors - requiring extensive pre-filtering.

A benchmark of alignment-free methods for various sequence analysis applications can be found in [39].

Assembly-based methods

There are two forms of assembly-based variant calling. In the first, an assembled genome for a sample (or samples) is compared to a reference via whole-genome alignment. Software such as MUMmer [40] or `minimap2`/`paftools` [41] facilitate this assembly-to-assembly alignment and then identify positions where the two disagree. An assembly-based method that is prevalent in bacterial genomics is Parsnp [42], which aligns the *core* genome of assemblies and then calls SNPs (only) *between* those genomes. A major limitation with these types of assembly-based approaches is that there is no sense of the quality of calls. As an assembly naturally has a depth of 1x at all positions, there is no information about variant support - all variants are considered equal in this scheme.

The second form of assembly-based variant calling performs assembly and genotyping. Cortex *de novo* assembles a sample from sequencing reads and genotypes variants at "bubble" sites in its de Bruijn graph [43]. It produces variant calls with

respect to a provided reference genome and has been used extensively in bacterial genomics [44–48].

A recent comprehensive benchmark of alignment-based variant calling found that the choice of reference genome, rather than the choice of tools, has the most critical impact on accuracy [49]. The best general-purpose pipeline was found to be Snippy; however, they note that species-specific filtering of the final VCF file can cause the performance of many tools to converge.

Reference genome bias is the most significant limitation of bacterial variant calling approaches [27, 49, 50]. The bacterial pan-genome highlights this impediment in a stark way. As an illustration of this, Figure 1.3 shows a cartoon depiction of the "single-reference problem". In this figure, we see that no two genomes contain the same complete set of loci - precisely what we expect from nearly any pan-genome [2]. Thus, using any of these genomes as a reference to call variants against will inevitably mean we cannot describe variants in loci not found in both the reference and query genomes. This type of bias is termed *hard* reference bias. Another, more subtle, form is *soft* reference bias, which results from difficulties aligning to the reference due to divergence in shared loci between the reference and query - especially around structural variants [27, 51, 52]. However, these biases tend to impact clonal species, such as *M. tuberculosis*, much less than those with open pan-genomes [49].

Given the biases resulting from the use of a single reference genome, an alternative solution is needed. One solution that is rapidly maturing is the use of a *genome graph* to replace a single reference.

1.2 Genome graphs

Genome graphs are a way of representing variation within a population, be it a bacterial species, a human gene, or a viral quasi-species [53]. Figure 1.4 illustrates a generic representation of a genome graph, where redundant information (consensus) is collapsed into a linear sequence and variation is represented as divergent paths leading in and out of these consensus segments. Thus, a walk (path) through such a graph represents a mosaic of the population variation.

A rich array of algorithms and methods have been proposed for representing and operating on genome graphs across all kingdoms of life [53–55]. In this section, we will highlight some mature genome graph frameworks, along with their limitations. In

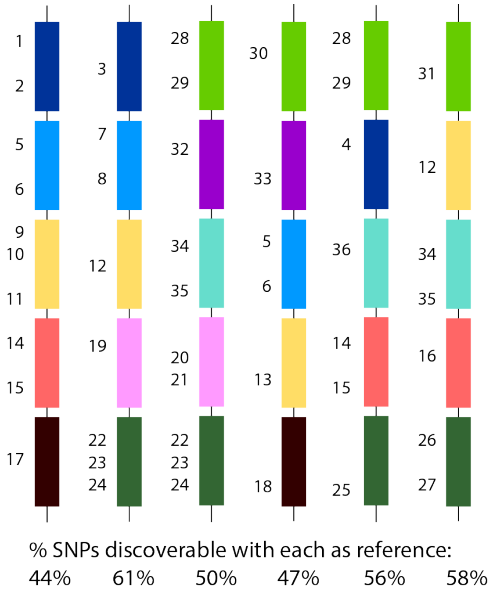


Fig. 1.3: An illustration of the single-reference problem. Each vertical column depicts a genome, with the coloured blocks representing loci (genes). The numbers next to each locus indicate a variant with respect to other loci of the same colour. The percentage figures at the bottom indicate what percentage of variants in the other genomes could be found by a perfect variant caller if that genome was used as a reference. Not all genomes contain the same loci; hence no genome can capture all of the variants.

the context of this thesis, limitations will focus on the applicability of a method to the bacterial pan-genome. Finally, we follow these existing methods by introducing a new genome graph approach relevant to this thesis.

1.2.1 Existing methods

GraphTyper

GraphTyper [56, 57] represents a genome graph as a directed acyclic graph (DAG) built from a reference sequence plus known variants (similar to Figure 1.4, but with directionality). Sequencing reads are mapped to the reference genome with BWA-MEM. The reference sequence is then broken into 50kbp regions, and reads are realigned to the graph in the respective region they map to. A path for the read is detected using a seed-and-extend approach, and variants are genotyped based on the read-support from this alignment. Impressively, GraphTyper can genotype SNPs, indels, and complex and structural variants.

In the context of bacterial genomes, there are several limitations with GraphTyper. First, a single reference genome is used as the backbone of the graph. This is a feasible

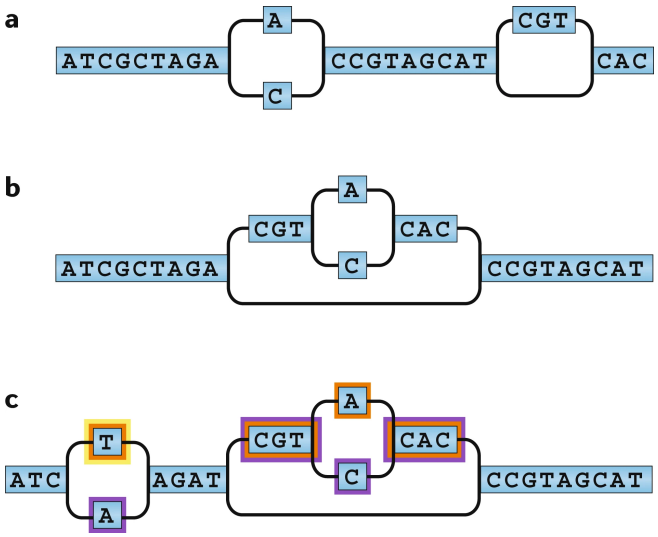


Fig. 1.4: Conceptual representation of a genome graph. **a**) Variants cause a "bubble", or divergent path. Note, the second bubble represents an insertion/deletion. **b**) Variation can be arbitrarily nested. In this example, there is a SNP within an insertion. **c**) Haplotype information can be encoded by "colouring" variants and disallowing paths that mix colours.

solution for humans, but not for bacteria, where, as we have seen, it is likely that two genomes do not have an identical gene repertoire. Second, the initial alignment of reads to the reference genome suffers the same soft and hard reference bias discussed in Section 1.1.4. The realignment of reads reduces the soft reference bias compared to linear genome methods; however, the hard bias remains. Third, no long-read sequencing technology support is available and is unlikely to be possible with the current seed-and-extend approach used for alignment [41].

Variation graph toolkit

The Variation Graph Toolkit (VG) [58] is a suite of tools for construction, alignment, and genotyping of genome graphs. The representation used by VG is an *un*directed, potentially cyclic, graph. Variation graphs can be constructed from a single reference and associated VCF file or multiple genome assemblies. Read alignment to the graph uses a seed-and-extend approach. Variant calls are made via a basic read pileup on the graph and then augmenting the original graph with novel candidates, followed by genotyping [59].

As with GraphTyper, the use of seed-and-extend alignment makes the support of long reads unlikely. (We note there have been discussions within the VG GitHub repository for four years about supporting long reads; however, no support has been

announced). Another limitation of VG is that in order to produce variant calls, the user must provide a reference genome, again, either inheriting reference bias or requiring a verbose description of simple variants (see Section 1.3.3 for an elaboration of this point).

A significant limitation of VG is its computational resource requirements. As VG attempts to be a "general" method - i.e., it is undirected and allows cycles to support events such as inversions and repeats naturally - it is CPU and disk intensive [60–62] to the point of requiring over one terabyte of temporary disk space to construct a bacterial (*M. tuberculosis*) graph [61], or not being useable [62].

Minigraph

Minigraph [62] represents a genome graph as a bidirectional graph, which allows cycles. The construction process starts with a single genome and iteratively adds structural variants (SVs; regions of divergence $\geq$ 100bp and $\leq$ 100kbp). In each round, a genome is aligned to the existing graph (a linear sequence in the beginning), augmenting it with sequence from poorly mapped regions (SVs). Minigraph aligns sequencing reads or assemblies to this graph using a modified version of `minimap2`'s minimizer *k*-mer-based seed-and-chain approach [41]. As such, Minigraph should inherently support long reads.

As Minigraph only incorporates SVs of 100bp or longer, it does not variant-call in the typical sense. It instead produces a BED-like file that calls SVs from the alignment. This is the main limitation of Minigraph: it cannot call variants smaller than 100bp, which are especially important in bacterial genomes. However, the authors acknowledge this limitation and state that the reason for this exclusion is that smaller variants can be easily identified with standard approaches.

Gramtools

Gramtools [61, 63] represents a genome graph as a DAG that can be constructed from either a single reference and associated VCF file or a multiple sequence alignment (MSA) (it uses the same model outlined below in Section 1.3.1). Alignment of sequencing reads is facilitated by the variation-aware Burrows-Wheeler Transform (vBWT; [63]), which is an extension of the original linear BWT to graphs. It genotypes variants under a haploid or diploid likelihood-based model and produces variant calls in the standard VCF. Alternatively, Gramtools can write variant calls to a new JSON-like VCF (jVCF) file, which stores the standard VCF information, with the addition

of graph-relevant details about the nesting of sites [61]. As with the other genome graph methods, though, Gramtools only supports short Illumina sequencing data and is unlikely to support long reads with a higher error rate than Illumina.

All existing genome graph methods require an enforced ordering - i.e., loci are not considered independently. Despite the fluidity of bacterial genomes, there is surprisingly conserved gene ordering [64, 65]. However, the enforced order of these genome graphs cannot account for variations in gene repertoire - i.e., the bacterial pan-genome.

VG has been applied to bacteria for strain-typing and abundance estimates in *E. coli*; however, individual graphs need to be concatenated together for each gene, thus enforcing an order [60]. None of the methods, to our knowledge, natively allows the independence of loci. This behaviour can be approximated but requires custom pipelines, as in [60]. Additionally, no existing genome graph method supports long-read sequencing technologies such as Nanopore.

These limitations are a driving motive for the development of the genome graph method Pandora.

1.3 Pandora: bacterial pan-genomics with reference graphs

As we have seen, the bacterial pan-genome can be amazingly diverse at both the nucleotide and gene (locus) levels. Thus, using a single reference to describe such variation is inadequate; the pan-genome seems a perfect application for genome graphs. However, existing methods fail to allow for structural differences at the locus level (Section 1.2.1) and therefore are unable to describe nucleotide-level variation in the accessory genome. Another shared limitation of existing genome graph tools is the lack of support for long-read sequencing technologies (we outline the significance of this in Section 1.4).

Pandora is a genome graph method that addresses these limitations. Rachel Colquhoun developed `pandora` during her PhD thesis [66]; we provide a brief overview of its methodology here as we extend and apply it throughout this thesis.

1.3.1 Population reference graph construction

The genome graph representation used by `pandora` is a DAG. However, unlike Gramtools, which uses the same representation [61], `pandora` is agnostic to locus ordering.

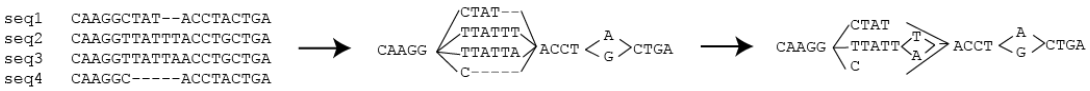


Fig. 1.5: Construction of a locus reference graph (PRG) from a multiple sequence alignment (MSA; left) with the recursive cluster and collapse algorithm implemented in `make_prg`. Vertical slices in the MSA are collapsed when there is a minimum match length of 4. Sections not collapsed are recursively clustered and collapsed (if possible), until no further clustering is possible or a maximum nesting level is attained. In this example, a nesting level of 2 is reached.

Instead, `pandora` interprets a genome graph (interchangeably referred to as a *reference graph*) at two levels: locus and pan-genome. We call a locus-level reference graph a *population reference graph* (PRG) as it represents the variation within a given population for a locus. A PRG is not restricted in its scope for a locus; it can be a gene, intergenic region, operon, or any other grouping desired. A pan-genome-level reference graph is termed a *pan-genome reference graph* (PanRG) and is a collection of PRGs. Again, a PanRG is not limited in its scope; it could describe a pan-genome, a meta-genome, or a collection of antimicrobial resistance-associated genes.

Construction of a PRG is accomplished with a recursive cluster and collapse algorithm, implemented in the software program `make_prg` (https://github.com/iqbal-lab-org/make_prg; [10, 66]). Two parameters are key to this process: the minimum match length, m, and the maximum nesting level, n. Starting with an MSA of locus sequences, when $\geq m$ positions agree, they are collapsed into a single sequence. The remaining non-collapsed sections of the MSA are recursively clustered, with (sub)sequences in the cluster being collapsed or clustered again. This recursive clustering and collapsing continues until all clusters contain a single sequence or recursion has occurred more than n times. This process is illustrated in Figure 1.5, which uses $m = 4$ and $n \geq 2$.

1.3.2 Index, quasi-map, and sequence inference

(w,k)-minimizers

A core concept within `pandora` is (w,k)-minimizers [67] - interchangeably referred to as minimizer (or minimizing) k-mers. A (w,k)-minimizer is a representative k-mer from a collection of w consecutive k-mers in a string (sequence). `pandora` uses the same strategy as minimap [68] - the k-mer with the minimum invertible integer hash function value is selected as the minimizer. The purpose of minimizer k-mers is to reduce the number of k-mers required to represent a sequence but ensure that if

two strings share a significant exact match, they will share at least one minimizer. Additionally, `pandora` requires $w \leq k$, ensuring all PRG bases are covered by a minimizer (except, at most, the first and last $w-1$ bases).

Indexing

Each PRG is represented within `pandora` as a minimizer k-mer graph. This graph is constructed by walking all paths in the PRG and selecting minimizer k-mers as outlined above. As $w \leq k$, walking each path ensures a minimizer covers every site within the graph. The index of a PanRG is a map from a minimizer k-mer to the position(s) and PRG(s) it occurs in.

Quasi-mapping

Quasi-mapping - as opposed to mapping - is a form of approximate alignment. The goal of quasi-mapping is to identify which locus (or loci) a read originates from, and *roughly* where within that locus each section of the read maps. To perform this quasi-mapping, `pandora` looks up all (w,k)-minimizers of a read in the index. Then, for every read minimizer that occurs in the index, a *hit* (read and PRG positions) is recorded. A single read minimizer can have multiple hits if the minimizing k-mer occurs in multiple locations in the PanRG. As such, once all hits are identified, they are filtered to remove spurious ones. This filtering is done by keeping only those hits that cluster together on a read and only occur in a single PRG. Thus, all PRGs associated with a cluster of hits are deemed present in the sample, while the remaining loci are considered absent.

Sequence inference

A major reason for `pandora`'s reduced reference bias is that it does not demand a single reference genome. Instead, it takes a PanRG and infers the closest sequence in that PanRG to the sample under consideration. As we saw in Section 1.1.4, the choice of reference is often the biggest limitation when calling variants in bacterial genomes.

For each PRG deemed present after quasi-mapping, `pandora` has (filtered) coverage information for the minimizers in the k-mer graph. A dynamic programming algorithm is used to find the path through the k-mer graph that maximises the log-likelihood score. This inferred sequence (path) is also referred to as the *maximum likelihood path*.

1.3.3 Variation inference

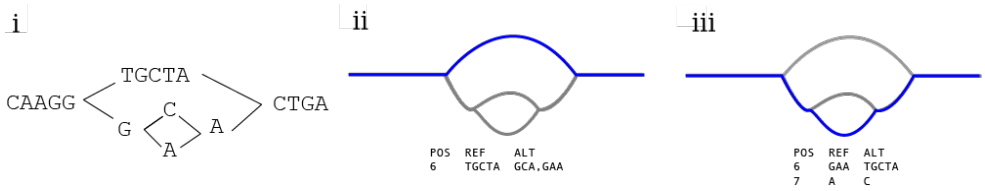
Single-sample

The single-sample inference and genotyping mode of `pandora` is coordinated by the `pandora map` routine. It quasi-maps sequencing reads to the PanRG and infers a sequence for each PRG. In addition, if requested, `pandora map` will also genotype the sample against the maximum likelihood path for each PRG (or a user-provided sequence if it exists). Thus, genotyping occurs for all variation sites within the PRG and is returned as a VCF file. An example of this single-sample genotyping and VCF is shown in Figure 1.6a.

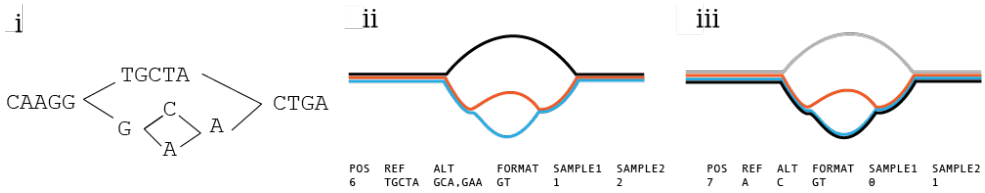
Multi-sample

Variation inference can also be performed for a collection of samples with the `pandora compare` protocol. Reference bias problems have traditionally plagued this type of analysis. If the collection of samples are of the same strain, then analysis against the same reference is not problematic, but once even a single sample originates from a different strain, the pan-genome exerts itself. As we outlined in Section 1.1.3, analysing divergent samples generally works by looking at variation in the core genome while resorting to locus presence-absence in the accessory genome. Multi-sample inference with `pandora compare` offers the best of both worlds; variation is inferred for both the core and accessory genome. Where a locus is absent from a sample, all sites for that locus are represented with a null genotype. In this approach, if a locus is present in only 2/20 samples in the collection, variants for that locus are only inferred for those two samples.

To allow this multi-sample variation inference, `pandora` infers the maximum likelihood path for each sample (Section 1.3.2). Then, using the same dynamic programming algorithm, `pandora` infers a maximum likelihood path for *the collection* of samples; instead of k-mer coverage, the number of maximum likelihood paths covering each minimizer is used for inferring the most likely path. In the end, the inferred sequence is selected to be maximally close to all samples in the collection. Therefore, all samples are genotyped against the same reference at all variant sites, making direct sample comparisons possible. This approach also ensures that small differences between samples are described as such - as shown in Figure 1.6b.



(a) Single-sample variant representation



(b) Multi-sample variant representation

Source: [66]

Source: [10] under the terms of the Creative Commons CC BY license

Fig. 1.6: The impact of reference choice on variant representation. In both **(a)** and **(b)** the left panel (i) shows the PRG. **a)** the blue line indicates the reference sequence. ii and iii show how the choice of this sequence affects the representation of variant sites when genotyping. **b**) the black line indicates the reference sequence, while the blue and orange lines are two different samples. ii and iii show that small differences between the samples are represented as small variants by selecting the sequence that is maximally close to the two samples (iii).

The `pandora` method addresses the main limitations of existing genome graph approaches (Section 1.2.1) in the context of bacterial pan-genomes. In particular, `pandora` supports both short (Illumina) and long (Nanopore) sequencing reads and removes hard reference bias by letting go of locus ordering and genotyping loci regardless of their genomic context.

Despite these advantages, the method has a notable limitation: an inability to detect novel variants. As variation inference is achieved by genotyping all sites in a PRG, it follows that if a variant does not exist in a PRG, it cannot be detected by `pandora`. Chapter 2 of this thesis will address this limitation.

1.4 Nanopore sequencing

The sequencing of DNA and RNA with a nanopore was conceived of by multiple parties in the 1980s [69]. However, it was not commercially available until the release of the Oxford Nanopore Technologies (ONT) MinION™ device in 2014 [69, 70]. (We use *Nanopore* sequencing to indicate sequencing with an ONT device throughout this thesis). The MinION device is smaller than a smartphone and plugs into a computer's USB port.

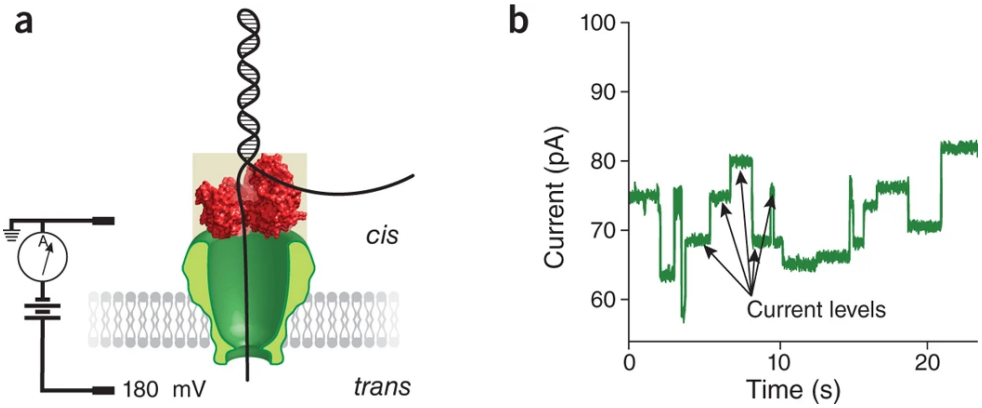


Fig. 1.7: Nanopore sequencing. **a**) A single strand of DNA or RNA (black) is passed through a nanopore (green) by an enzyme (red). A sensor records the electrical current flowing through the nanopore. **b**) The raw signal (electrical current; y-axis) changes as nucleotides move through the nanopore. Each nucleotide can be inferred by a characteristic alteration in the signal, indicated by the black arrows.

Nanopore sequencing works by passing a strand of DNA or RNA through a nanopore. The nanopore is embedded in an electro-resistant membrane, with a flow cell containing an array of such pores. A sensor attached to each nanopore records the electrical current passing through it; as the DNA or RNA strand moves through the nanopore, this current signal changes. There are approximately five nucleotides present in the nanopore at a time. As such, the identity of each nucleotide (base) can be inferred by a characteristic signal alteration (see Figure 1.7). Because the sensor takes measurements at a faster speed than the DNA moves through the nanopore, multiple recordings are obtained for each base transition - see the distinctive "steps" in Figure 1.7b. The process of inferring a DNA or RNA sequence from this raw signal is called *basecalling*.

1.4.1 Basecalling

The raw signal and metadata for each molecule read by a nanopore are deposited into a hierarchical data format file (HDF5; referred to as a *fast5* file; [71]). Furthermore, these fast5 files are produced in real-time. Thus, the user has immediate access to the sequencing data as soon as it is produced, a unique advantage over existing sequencing technologies.

Basecalling algorithms have seen substantial development since the release of ONT's first device. The progression of these algorithms, along with nanopore structure and chemistry, has led to a steady increase in the accuracy of genomic sequences

inferred from Nanopore sequencing (as shown in Figure 1.8; [72]). For example, from an initial read-level accuracy of approximately 60% [73], recent studies have reported accuracy of 93.2% [74].

Traditionally, there were two key components to basecalling. The first was the segmentation of the raw signal into "events" (the plateaus in Figure 1.7). These events represent a 5-mer within the nanopore; therefore, consecutive events describe a sequence of nucleotides entering and leaving the nanopore. The second component of traditional basecalling is applying an algorithm to these events to infer the genomic sequence.

Basecalling algorithms require an *a priori* model trained on molecules for which the sequence is known. Metrichor was the original software provided by ONT for basecalling; it used a hidden Markov model (HMM) to turn events into a sequence. In 2017, Boža *et al.* developed the first basecalling method to use a recurrent neural network (RNN) - instead of an HMM - to turn events into sequence with increased accuracy [75]. RNN basecalling was provided soon after this with ONT's new Albacore algorithm. The following major algorithmic development was the removal of the segmentation step (the most error-prone stage of basecalling). These programs, Chiron and BasecRAWller, used a Connectionist temporal classification (CTC) decoder to label unsegmented raw signal and subsequent basecalling with an RNN and convolutional neural network [76, 77]. Again, soon after this, ONT released a new basecalling program, Guppy, which incorporated these new ideas. Since its release, Guppy has been the gold-standard basecalling algorithm, and all algorithmic development has focused on labelling the raw signal data.

The pre-trained models distributed with guppy aim to be general. That is, they can be used on Nanopore data from any organism. The species these models were trained with is not common knowledge; however, it is fair to assume a variety were used given guppy's consistent performance across kingdoms and species. In 2019, Wick *et al.* showed that training a taxon-specific (Enterobacteriaceae) basecalling model can provide increased accuracy when used to basecall a sample from that taxon [78]. Their analysis also revealed that - at least in Enterobacteriaceae - Dcm-methylation sites were the primary source of guppy basecalling errors and that the taxon-specific model removes nearly all of the errors of this type. In addition, as has also been described elsewhere [79], homopolymer deletions were found to be the next most common source of systematic errors and were also reduced with the use of a taxon-specific model.

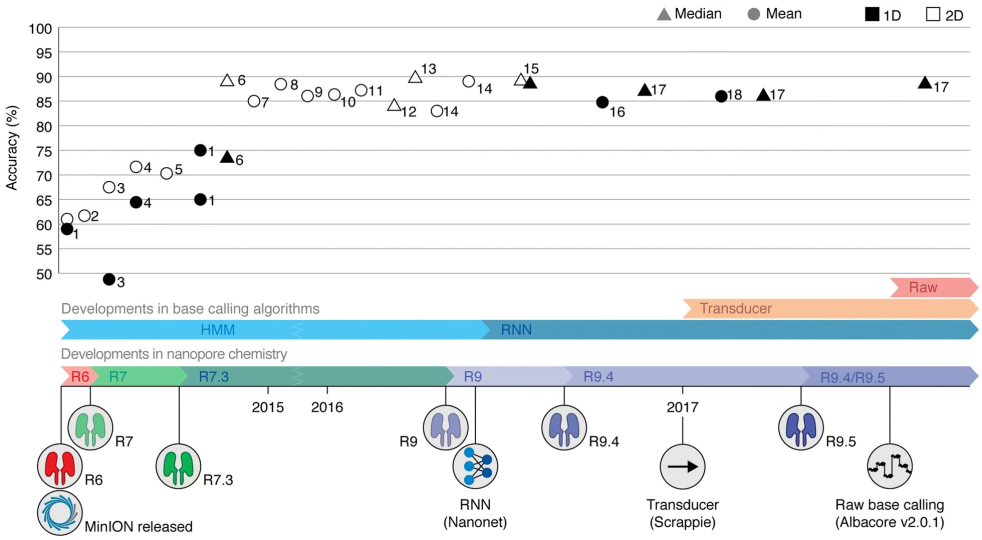


Fig. 1.8: Timeline of ONT nanopore chemistry and the accuracy of Nanopore sequencing data. The numbers attached to each point in the accuracy plot relate to published work reporting the accuracy measurement. A list of those publications can be found in the original article this figure was taken from [72]. HMM=hidden Markov model; RNN=recurrent neural network; 2D=a form of Nanopore sequencing where both strands are sequenced.

The current state of Nanopore basecalling suggests that improvements in accuracy will come from three areas: chemistry, basecalling algorithm, and training data improvements. Therefore, in Chapter 5 we will look to improve Nanopore accuracy by focusing on the latter of those areas and training a *species*-specific basecalling model.

1.4.2 Variant calling

Variant-calling from Nanopore sequencing data is slowly maturing but is yet to reach the standards set by other sequencing technologies. Much of the problems relate to the lower read accuracy of Nanopore data compared to Illumina and PacBio. A higher error rate means it is harder for genotyping models to distinguish genuine variants from technology-derived errors.

The first method specifically designed to call variation from Nanopore data was Nanopolish [80, 81] - developed in 2015. It maps the raw signal for each read to a reference genome and uses an HMM to determine if the reads show support for a variant. However, Nanopolish requires access to the fast5 files in order to call variants. Operating on fast5 files is incredibly IO intensive, and this contributes to observed Nanopolish runtimes in the order of days, and peak memory in the 100's of gigabytes [82]. More recent variant callers designed to support Nanopore

data such as DeepVariant [83], Clairvoyante [82], and Clair [84] all use multi-layer neural networks and require significant effort on behalf of the user to operate [85]. An ONT-developed variant caller (and sequence correction) tool, Medaka (https://github.com/nanoporetech/medaka), also uses neural networks but is much simpler to use. In addition to these newer, complex methods, some researchers have found that more traditional short-read approaches such as GATK [31] can be tuned to work well with Nanopore data [86–88].

Despite the wealth of Nanopore basecalling [78] and assembly [89] benchmarking studies, there are relatively few independent variant calling assessments. A recent analysis by Sanderson *et al.* evaluated Clair, Nanopolish, and Medaka for variant calling in *Neisseria gonorrhoeae* [85]. However, this study used Illumina as the "truth" and only assessed SNP calls. They found Clair was the best-performing method and could detect 94-98% of Illumina SNP calls; however, it required training a random forest classifier to achieve this result. In addition, to reduce false-positive SNPs from 49-289/genome to 4-19/genome, SNP detection dropped to 76-94% - the upper end of this range is excellent, while the lower is not. Therefore, it remains to be seen what the true precision and recall of Nanopore variant calling is - for both SNPs and indels.

1.4.3 Benefits and limitations

As a result of its vastly different approach to sequencing, Nanopore offers many unique benefits over the gold-standard Illumina. However, there are limitations, and Nanopore will unlikely ever be a solution for all problems. We will briefly summarise some of these pros and cons with respect to Illumina in the context of pathogen genomics and its application to real-world settings.

Cost

Depending on the context of the application, Nanopore sequencing is both more and less expensive than Illumina sequencing. At a device and peripherals level, a few layers of cost need to be understood. The upfront price of an ONT MinION device is €900, while an Illumina MiSeq costs €100,000 - 111 times more than the MinION [90]. The cost per lane/flow cell and library prep is €600 and €90-130 for the MinION, while for the MiSeq, these two items are €2,000 and €50-100, respectively. Considering these prices, the cost of generating one billion bases (1gbp) of sequencing data is €12 on a MinION device and €175 for a MiSeq. However, the caveat here is that, currently, the 1gbp of Illumina data will be of superior quality. Other costs that are harder to

quantify are the need for more extended library preparation for Illumina sequencing, which causes additional labour expenses [90].

A recent study from the New York State Department of Health also found that, for *M. tuberculosis* clinical diagnostics, Nanopore sequencing cost ~US$57 per sample, while Illumina came in at US$130 per sample [91].

Portability

The ONT MinION weighs just 87g and has a volume of 80cm^3, while the Illumina MiSeq respective weight and volume are 57.2kg and 202,709cm^3. In addition, the MinION can be powered through the USB port of a computer, while the MiSeq requires significantly more power resources [90]. These two aspects combine to make the MinION extraordinarily portable and the MiSeq completely stationary.

The benefit of the MinION's portability has famously been exhibited during the Zika and Ebola outbreaks, where Nanopore sequencing was used on-the-ground to aid public health investigations [92–94].

Diagnostics

Another benefit of Nanopore sequencing is that the data is made available in real-time. A standard MiSeq runtime is approximately 55 hours [90], and therefore it takes at least 55 hours to have access to the sequencing information. In contrast, MinION diagnostics can operate as rapidly as the sequencing. Indeed, Nanopore sequencing has been applied to same-day *M. tuberculosis* diagnostics, with phylogenetic placement and full drug susceptibility profiles in 12.5 hours [95]. Other real-time diagnostic applications have included detection of surgical device infections [96], Ebola [93, 94] and Zika [92] virus outbreak surveillance, and Enterobacteriaceae strain and drug resistance identification [97]. Even conventional genomics for bacterial diagnostics stands to gain from the use of Nanopore sequencing. For example, Greig *et al.* from Public Health England found that the better resolution of the Shiga toxin-producing *E. coli* accessory genome from Nanopore sequencing lead to improved resolution during outbreak investigation [86].

Another emerging benefit of Nanopore sequencing is the artificial enrichment/ depletion of samples [98, 99]. An API (Read Until) facilitates this enrichment within the MinION device, allowing for ejecting (rejecting) molecules from the nanopore. The sequencing data is mapped to a reference database, and rules can be provided that reject a molecule if it originates from a specified reference or once a certain read depth

has been reached. One application by Kovaka *et al.* was to deplete bacterial DNA, thus enriching yeast genetic material [99]; however, this could easily be employed in reverse, depleting human DNA and enriching bacterial or viral material in a patient-derived sample.

While there are many benefits to Nanopore sequencing, the accuracy of data provided is still well behind Illumina. However, as we have seen, increased algorithm and method development is helping to reduce the difference. Therefore, a primary focus of this thesis is the improvement of Nanopore-derived sequencing information. In particular, after adding the ability for `pandora` to discover novel variants, we will turn our attention to genome graph and Nanopore sequencing applications for the pathogen *Mycobacterium tuberculosis*.

1.5 Tuberculosis and its causative agent

Tuberculosis (TB) is an ancient airborne disease that predominantly affects the lungs [100]; evidence suggests it has been with humans since leaving Africa [101, 102]. In 2019, an estimated 1.4 million people died of TB globally, and over 10 million fell ill to the disease[103]. The causative agent of TB is principally the bacterium *Mycobacterium tuberculosis*, which has no known reservoir other than humans [102]. However, other members of the *Mycobacterium tuberculosis* complex (MTBC), such as *M. africanum* and *M. bovis*, can also cause TB [100, 104].

TB is a preventable and curable disease; 85% of patients with active disease can be successfully treated with a 6-month drug treatment that has the added benefit of preventing transmission [103]. Furthermore, estimates show that 60 million TB-caused deaths have been avoided since 2000. Despite this, TB remains in the top 10 causes of death worldwide [103].

In 2015, the World Health Organization (WHO) and United Nations developed the End TB Strategy, which aims - among other goals - for a 95% reduction in TB deaths by 2035 [105]. One of the central pillars of the End TB Strategy is "Intensified research and innovation" through the "discovery, development and rapid uptake of new tools, interventions and strategies." Another pillar, which will be a focus of this thesis, is improved patient-centred diagnostics for the "early diagnosis of tuberculosis including universal drug-susceptibility testing, and systematic screening of contacts and high-risk groups."

We highlight these specific focuses of the End TB strategy as they motivate much of the work in this thesis.

1.5.1 Epidemiological and phylogenetic diagnostics

Public health applications for whole-genome sequencing (WGS) of *M. tuberculosis* generally focus on three diagnostic use-cases: species/lineage identification, prediction of drug resistance, and clustering of samples for epidemiological purposes [106, 107].

Species/lineage identification

The MTBC is composed of seven species of mycobacteria with varying growth, host, and pathology characteristics [108]. In addition, non-tuberculous mycobacteria (NTM), such as *M. abscessus* and *M. avium*, can cause infections that present similarly to those of the MTBC [109]. However, NTMs can have very different drug resistance profiles to MTBC members [109, 110], highlighting the importance of correct species and lineage identification.

Routine species and lineage identification for suspected TB cases involve the use of the GenoType CM, and AS line probe assays (LPA; Hain Lifescience) [111, 112]. These LPAs work by reverse hybridisation of PCR products from the sample to species-specific probes from the 23S rDNA region [111].

WGS identifies species and lineage by detecting SNPs known to be unique to each species, lineage, and sublineage [108, 113–116], and has been proven as accurate enough for adoption by Public Health England [112]. In contrast to the LPAs, the resolution provided by WGS means that the various lineages and species can be delineated in a single test and can be easily adapted to new markers of lineages.

Transmission cluster detection

The inference of possible transmission events is an important component of preventing ongoing TB infections. Given the low mutation rate in *M. tuberculosis* - 0.5 SNPs/genome/year [117] - clustering tends to operate on the assumption that samples with few SNP differences are likely part of a transmission event.

Until recently, mycobacterial interspersed repetitive-unit–variable-number tandem-repeat (MIRU-VNTR) genotyping was the primary method for TB outbreak investigation [117]. MIRU-VNTR is a PCR test that measures the size of tandem repeats (VNTRs) from 24 loci in the *M. tuberculosis* genome. The size of these VNTRs,

as the name suggests, is variable among strains, and so this can be used to *exclude* transmission. However, without epidemiological data in support, it is unable to provide the fine-grained information needed to infer likely transmission [117, 118].

Illumina WGS has been extensively validated as a preferred means of identifying *M. tuberculosis* genetic relatedness - at least in high-income settings [117–121]. It provides greater resolution and lower costs than MIRU-VNTR and is now routinely used in some public health agencies, such as Public Health England, the New York State Department of Health, and the National Institute for Public Health and the Environment in the Netherlands [91, 118, 122].

Clustering samples based on WGS data is done by first calling SNPs with respect to the *M. tuberculosis* reference genome. The number of SNP differences between two samples defines their genetic distance (relatedness). Second, the pairwise distances for all of the samples under investigation are used to cluster cases if they have a distance less than or equal to a predefined threshold [117, 119]. For example, if the threshold is set to 5 and two samples A and B have a distance of 4, they are considered part of the same cluster. If a third sample C has a distance of 6 from A, but 2 from B, C is added to the cluster. Further epidemiological information can also be incorporated to improve connections [119, 123].

Two problems that afflict WGS-based clustering are bioinformatic and threshold disparity. The method for obtaining SNPs for a sample is far from standardised. The consequence of this lack of consistency was masterfully demonstrated by Walter *et al.* when they asked five different genomic epidemiological research groups to produce variant calls from the same outbreak data [124]. They found these variant call sets did not produce consistent transmission inferences and found that filtering of variants had a noticeably negative effect. Furthermore, an important study from Stimson *et al.* recently highlighted the issues with SNP threshold-based WGS clustering [123]. These limitations come down to the variety of SNP thresholds used and the contexts in which they are calibrated [123]. They provide an alternative approach that uses SNP difference, the timing of cases, molecular clock rates, and transmission processes to produce clusters based on the probability of two cases being separated by a given threshold of transmission events.

In addition to the limitations just described, Nanopore WGS for transmission inference has yet to see a thorough investigation. Smith *et al.* recently assessed Nanopore sequencing against Illumina but provided very little detail about the clustering and only used a small portion of their data for assessment [91]. This shortcoming motivates

the work in Chapter 3 where we examine Nanopore's ability to produce transmission clusters concordant with Illumina.

1.5.2 Antimicrobial resistance prediction

Antimicrobial resistance (AMR) is a global concern for TB care and prevention. The End TB Strategy seeks universal access to drug susceptibility testing (DST). The first-line drugs used in TB treatment are isoniazid, rifampicin, pyrazinamide, and ethambutol, while second-line antimicrobials include fluoroquinolones and aminoglycosides. In addition, the new and repurposed drugs bedaquiline, delamanid, pretomanid, clofazimine and linezolid are providing reduced treatment times and improved patient outcomes [125]. *M. tuberculosis* resistant to rifampicin (RR-TB) and isoniazid is deemed multi-drug resistant (MDR-TB). As these two drugs form a critical part of front-line treatment, detection of RR/MDR-TB is crucial in reducing the global TB burden [103]. In 2019, it was estimated that 3.3% of new and 18% of treated TB cases - 465,000 cases in total - were RR-TB, with 78% also being MDR-TB. [103].

Traditionally, the TB standard of care required phenotypic testing of the infecting organism against the four first-line drugs to prescribe appropriate treatment. However, *M. tuberculosis* is a slow-growing organism; therefore, phenotypic testing - which is still the "gold-standard" - takes many weeks to complete and can delay correct treatment by up to 80 days [126]. Thankfully, the Xpert® MTB/RIF assay (Cepheid) has helped provide rapid RR-TB testing (and TB detection), and in 2019, 61% of confirmed TB cases were tested for rifampicin resistance [103]. The Xpert® MTB/RIF assay is a PCR test that simultaneously detects MTBC and known rifampicin resistance-causing mutations in the *rpoB* gene in as little as two hours [127]. In addition, the newly developed and tested Xpert® MTB/XDR (Cepheid), which tests for resistance to isoniazid, fluoroquinolones, ethionamide, and aminoglycosides [128], stands to provide much greater access to diagnostics [129].

These Xpert® assays are a welcome addition to the DST of TB. However, due to their (necessary) use of a fixed set of resistance-causing genotypes, drug *susceptibility* cannot be inferred [130]. This inflexibility was highlighted in 2015, when 30% (38/125) of phenotypically RR-TB in Swaziland returned negative resistance results on the Xpert® MTB/RIF assay. The missed resistance was due to the presence of *rpoB* mutation I491F [130], which is not a mutation the Xpert® MTB/RIF recognises. As a result, the Xpert® MTB/RIF could not be reliably used in Swaziland or any other country with this RR-TB population.

WGS offers a more flexible solution that is much faster than gold-standard phenotyping methods - and now cheaper [95, 126, 131]. In addition, the accuracy of WGS-based predictions is now comparable to phenotyping [44, 45, 126, 131, 132], and can even be used as a replacement for first-line DST [133].

Predicting drug resistance from WGS data typically works by detecting known resistance-causing mutations from a curated catalogue - for *M. tuberculosis* these are almost always SNPs or indels [134]. The two most commonly used open-source software programs for WGS AMR prediction are TBProfiler [135, 136] and Mykrobe [44, 45]; although, in-house custom scripts are common [91, 133]. TBProfiler uses an alignment-based variant calling pipeline (Section 1.1.4) to identify the presence of mutations in their catalogue. Mykrobe, on the other hand, takes an assembly-based approach (Section 1.1.4) and maps sequencing data to de Bruijn graphs built from *in silico* probes of catalogue mutations.

Previous assessments of WGS-based AMR prediction for TB have focused on Illumina sequencing. However, as we saw in Section 1.4, Nanopore provides greater speed to results, reduced costs, and portability of sequencing. Indeed, proof-of-concept work by Votintseva *et al.* found it took 44 and 16 hours to gain WGS AMR predictions from Illumina's MiSeq and MiniSeq instruments, respectively [95]. In contrast, Nanopore-based predictions were available in 12.5 hours; the technology has improved significantly since then, so this interval is likely to have reduced.

Both TBProfiler and Mykrobe support Nanopore; however, both used a small sample size (Mykrobe n=5 and TBProfiler n=3) for validation. Recent work from Smith *et al.* confirmed the feasibility of Nanopore WGS for TB AMR prediction on a much larger, but homogeneous, cohort (n=431) with an in-house script [91].

To date, the most influential factor in the accuracy of a method's AMR predictions has been the catalogue [45]. These catalogues are constructed by aggregating mutations from large cohort studies where the impact of individual mutations is linked to a phenotype [45, 133, 134, 136, 137]. These catalogues are set to expand significantly after the work of the *Comprehensive Resistance Prediction for Tuberculosis: an International Consortium* (CRyPTIC), who sequenced 10,228 isolates with phenotypic information for thirteen drugs [138–140]. In addition, the WHO recently issued a mutation catalogue with metrics for association with resistant and susceptible isolates, along with a confidence grading for each [141]. However, the genetic basis of resistance for some drugs, such as the new and repurposed ones, remains only partially understood [125].

While WGS catalogue-based AMR predictions provide more flexibility than molecular assays, current approaches do not detect off-catalogue mutations. As in the Swaziland Xpert® MTB/RIF example [130], if a novel mutation arises in a population, current WGS methods will not identify the resistance. The CRyPTIC consortium recently introduced a new approach whereby if an unknown mutation is identified in a gene known to be involved in resistance, they refuse to make a prediction [133]. On their 10,290 samples, this approach achieved a specificity and sensitivity for first-line drugs acceptable for clinical usage. This method is now in use at Public Health England for all *M. tuberculosis* samples in England, where samples with unknown mutations are sent for phenotyping. Additionally, Hunt *et al.* quantified the cost of the pure-genotyping approach of `mykrobe`, showing that 2.4-4.6% of resistant samples were missed [45].

The lack of sufficient Nanopore WGS validation for TB AMR prediction, along with the inability of current methods to detect off-catalogue mutations, motivate the work in Chapter 4. In this chapter, we will leverage the new *de novo* variant discovery from Chapter 2 to develop an AMR prediction method that uses `pandora` and can flag off-catalogue mutations.

Using genome graphs for drug resistance prediction

As a brief aside, we introduce the benefits of `pandora` over `mykrobe` for the task of AMR prediction; the focus of Chapter 4. `mykrobe` uses population genome graphs built from a catalogue of known resistance-causing mutations for the genotyping of samples. The underlying method `mykrobe` uses for this is Cortex [43] - an assembly-based variant caller that uses de Bruijn graphs (dBGs; see Section 1.1.4). Cortex is somewhat of a precursor to `pandora`. However, `pandora` offers several advantages over Cortex. The first being the representation of the genome graph itself. Cortex uses k-mers in coloured dBGs, while `pandora` uses minimizing k-mers in a *directed* acyclic graph (see Section 1.3.1). In the context of Nanopore data, this distinction is important. Building dBGs from Nanopore data creates very complex graphs due to the number of erroneous k-mers that result from the high error rate. In addition, as the Nanopore error rate is higher than Illumina, a smaller k-mer size is required, another factor that increases the complexity of the dBG.

A second difference in the graph representations of Cortex and `pandora` is how k-mer "hits" are incorporated. In a dBG, anywhere that a k-mer matches, the depth is incremented by one. However, in `pandora` such hits are dependent on the context

of the read (see Section 1.3.2). If a k-mer matches two locations in the graph, but one location has many hits close by from the same read while the other does not, the spurious hit is discarded. This filtering of k-mer hits allows `pandora` to use a lower k-mer size ($k = 15$) than Cortex (`mykrobe` uses $k = 21$). The flow-on effect of using a smaller k-mer size is `pandora` does not require as much read depth as there is a much higher chance of matches to smaller k-mers, especially when the error rate is high. For example, assuming a Nanopore error rate of 0.08, we would expect the probability of a $15-mer$ and $21-mer$ having no errors to be 0.30 and 0.19, respectively.

1.6 Executive summary of this thesis

We begin this thesis in Chapter 2 by describing algorithms to facilitate the *de novo* discovery of variants in a (`pandora`) genome graph. We first calibrate this method, and associated parameters, on a simulated dataset. Next, we use an empirical dataset of 20 diverse *E. coli* genomes with Illumina and Nanopore data to evaluate the precision and recall of `pandora`, with and without *de novo* variant discovery. We additionally show that `pandora`'s representation of a pan-genome as a genome graph provides superior recall to single-reference methods and a substantially lower Nanopore error rate than existing methods.

For the remainder of the thesis, we turn our attention to *M. tuberculosis* and investigate how genome graphs and Nanopore sequencing can improve public health applications for this pathogen. In Chapter 3, we provide a detailed analysis of Nanopore-based transmission cluster inference. Using a diverse dataset of 150 *M. tuberculosis* clinical isolates, we show that BCFtools SNP calls from Nanopore data produce transmission clusters that are highly concordant with those inferred from Illumina and do not miss any samples from their expected cluster. Additionally, we explore the efficacy of `pandora` for constructing transmission clusters and find that while no samples are missed from their expected clusters, more work is needed to prevent the merging of separate clusters.

In Chapter 4, we outline a method for drug resistance prediction with `pandora` reference graphs (`drprg`). We compare this method and `mykrobe` against available first- and second-line drug phenotypes for the 150 samples from Chapter 3. As a result, we simultaneously show that Nanopore WGS AMR predictions are concordant with Illumina and `drprg` predictions are consistent with `mykrobe`. We also find the major sources of error for both `mykrobe` and `drprg` with Nanopore data and investigate. In

addition, we measure `drprg`'s ability to detect off-catalogue mutations and find that it leads to less missed resistance predictions.

Finally, in Chapter 5, we train an *M. tuberculosis*-specific Nanopore basecalling model and illustrate its increased read- and consensus-level accuracy over the default basecalling model. Furthermore, we show that our species-specific model reduces homopolymer deletion errors - an error type we encounter multiple times in this thesis.

Chapter 2

Variant discovery in genome graphs

2.0 Publication and collaboration acknowledgements

The software program that this chapter extends, `pandora`, was first conceived and implemented in Rachel Colquhoun's DPhil thesis [66].

A paper describing `pandora` and the work in this chapter is available at [10]. This paper was a collaborative project that spans five years of work by Rachel (first author), myself (second author), Leandro Ishi (third author), and others.

My aim here is to clarify what work is solely mine and what was completed in collaboration. Where possible, I have also attempted to indicate within certain sections the work not completed by myself.

The method for performing *de novo* variant discovery in Section 2.2 is my own, with input from collaborators. However, I implemented it within the `pandora` codebase. Section 2.2.3 describes a process for pruning the path-space in a de Bruijn graph, this work was the joint idea of myself and Leandro Ishi, and Leandro added the implementation for the distance map calculation.

All of the work in Section 2.3 is my own.

Section 2.4 describes a subset of the results in [10]. The evaluation framework used in the paper (and this section) underwent many iterations over two years. Leandro Ishi and I conceived and implemented the original method of calculating precision and recall in a coordinate-agnostic manner with the mapping of variant probes. This framework was later incorporated and adapted by Martin Hunt in the tool Varifier (https://github.com/iqbal-lab-org/varifier). Leandro Ishi implemented the final evaluation method but it is heavily based on the original work we performed together.

In addition, several components of the pipeline to run the analysis in Section 2.4 and [10] were initially implemented by myself but were later refactored or changed by Leandro.

The idea for the Nanopore basecalling model comparison in Section 2.4.4 was mine, as was the initial implementation - but not the final. Section 2.4.5 was ultimately performed by Leandro Ishi; however, I had much input and contributed pipeline code in the beginning.

The construction of the pan-genome truth set of variants discussed in Section 2.4.3 and Section A.2 is the work of Leandro Ishi. It is included in this chapter to aid the reader's understanding of how recall is calculated.

2.1 Introduction

Standard approaches to variant analysis are effectively a first-order approximation. In such an approximation, samples are considered identical to a selected reference; one aligns sequencing reads to it, identifying apparent variations via the read pileup, and then the reference is modified to get an estimate of the sample's genome. We term such a procedure a "linear" or "single-reference" approach.

As mentioned in Section 1.3, `pandora` is a method developed by a previous PhD student in our lab, Rachel Colquhoun [66]. `pandora` works on the principle of approximating a genome as a hierarchical mosaic. At a high level, `pandora` represents a pan-genome as a mosaic of loci, while at the locus level, it is a mosaic of previously observed genomes. Loci in this context can represent any genomic unit desired; a gene, intergenic region, or a mobile genetic element - the method is agnostic. Sequences from many genomes in a population are collapsed into a graph structure (Section 1.3.1) to form a locus population reference graph (PRG). All of the pan-genome's PRGs are collected into the high-level pan-genome reference graph (PanRG).

When given a set of Illumina or Nanopore sequencing reads, `pandora` identifies which PanRG loci are present and infers a consensus sequence for each. This consensus is the maximum likelihood path through the respective PRG and is used as the first-order approximation for the sample.

While `pandora` - before the work in this chapter - enables the comparison of genomes to a level of detail provided by no other tool, there is still a significant shortcoming: it cannot discover novel mutations. As such, before the work presented in this chapter, `pandora` was effectively a genotyping tool. If a sample contains a

variant not present in a PRG, the best `pandora` can do is select the path closest to that variant. Therefore, we begin this chapter by describing a method for performing *de novo* variant discovery in a genome graph and implement it in `pandora` - turning `pandora` into a two-stage approximation method.

We use a simulated genome and associated Nanopore dataset to show that without this discovery capability, `pandora` cannot detect variants absent from a PRG. In addition, we explore the impact of various parameters on the *de novo* discovery method and find read depth to have a vital influence.

Finally, we use an empirical dataset consisting of 20 diverse *E. coli* samples to show that with our new variant discovery approach, `pandora` has higher recall than all single-reference tools tested for both Illumina and Nanopore data. In addition, we identify methylation sites as a major source of our errors on Nanopore data and provide a solution for removing many of these errors. In the process of performing this analysis, we outline a coordinate-free method for evaluating variant caller precision and recall, facilitating the comparison of linear- and graph-based methods.

2.2 Methods

We define a method that extends `pandora`, with a subcommand `discover`, to allow for the *de novo* discovery of variants not present in a PRG. We implemented it within the `pandora` codebase in the C++ programming language.

`pandora`, as implemented by Rachel Colquhoun (before this chapter), approximates a novel genome as a mosaic of prior genomes - the nearest path through the PanRG. In this chapter, we add two further steps: first, locating regions of the mosaic which were not supported by reads ("candidate regions"), and second, performing a particular type of local assembly in those regions.

The first step of *de novo* variant discovery in genome graphs is finding the candidate regions that show evidence of dissimilarity from the sample's reads.

2.2.1 Finding candidate regions

The input required for finding candidate regions are a locus PRG (node), n, within the `pandora` PanRG; the maximum likelihood path of n, as both sequence mlp_n and (minimizer) k-mers, $kmlp_n$; and a padding size, w, for the number of positions surrounding the candidate region to retrieve.

We define a candidate region, r, as an interval within mlp_n where read depth (coverage) is less than a given threshold, for less than m consecutive positions. m acts to restrict the size of variants we can detect. If set too large, the following steps become much slower due to the combinatorial expansion of possible paths. We note that coverage is actually stored on $kmlp_n$, but is stored for the whole k-mer. We convert the coverage on $kmlp_n$ into per-position coverage on mlp_n and use that to identify low-coverage segments as described.

2.2.2 Enumerating paths through candidate regions

For all candidate regions, r, we construct a de Bruijn graph G_r from the subsequences of the reads that overlap r, using the GATB library [142].

We define A_L and A_R as sets of k-mers to the left and right of r in the maximum likelihood path, mlp_n. They are anchors to allow insertion of new sequences found by *de novo* discovery into the PRG. Each set has a maximum size of k.

We abandon *de novo* discovery if no pairwise combination of A_L and A_R exists in the de Bruijn graph G_r.

We use sets of k-mers for A_L and A_R, rather than a single anchor k-mer, to provide redundancy in the case where sequencing errors cause some anchors to not be in G_r. We define the start anchor k-mer, a_L, as the first (left-most) $a_L \in A_L \wedge a_L \in G_r$. Likewise, we define the end anchor k-mer, a_R, as the left-most $a_R \in A_R \wedge a_R \in G_r$.

Now that we have two anchor k-mers, a_L and a_R, our goal is to find all valid paths between these anchors in G_r.

To identify valid paths, we perform depth-first search (DFS) on G_r, beginning from a_L, to obtain a spanning tree, T_r. p_r is defined as a path from the root node a_L of T_r and ending at node a_R, which fulfils the following two conditions: i) p_r is shorter than the maximum allowed path length; ii) no more than k nodes along p_r have coverage $< (0.1 n_r e_r)$, where e_r is the expected k-mer coverage for r and n_r is the number of iterations of path enumeration for r.

V_r is the set of all p_r satisfying these conditions. If $|V_r|$ is greater than a predefined threshold, η, n_r is incremented by 1 and V_r is repopulated. If $0.1 n_r = 1.0$ then *de novo* discovery is abandoned for r.

The second condition listed above, which relies on n_r and e_r, has the effect of progressively increasing the amount of coverage we demand on a candidate path (p_r). In the first iteration, $n_r = 1$ - i.e., we require the path to have 10% of the expected read

depth (coverage). If this yields too many paths ($|V_r| > \eta$), we restart and require all paths to have 20% of the expected coverage. Finally, if we reach a stage where we require 100% of the expected coverage but still have too many paths, we quit *de novo* discovery for the candidate region.

2.2.3 Pruning the path-space in a candidate region

As `pandora` operates on both accurate and error-prone sequencing reads, the number of valid paths in G_r can be immense. In testing, we found that the path enumeration process can result in runtimes beyond seven days in some scenarios. The increased runtime is due to cycles occurring in G_r and exploring paths that will never reach our required end anchor (a_R).

In order to reduce the path-space within G_r, we prune paths based on multiple criteria. Critically, this pruning happens at each step of the graph walk (path enumeration; Section 2.2.2).

In addition to T_r, obtained by performing DFS on G_r, we produce a distance map D_r that results from running reversed breadth-first search (BFS) on G_r, beginning from the *end* anchor (a_R). We say reversed BFS as we explore the *predecessors* of each node, rather than the successors. D_r is a binary search tree where each node in the tree represents a k-mer in G_r that is reachable from a_R via reversed BFS. Each node additionally has an integer that describes the shortest path from that node to a_R.

We use D_r to prune the path-space as follows. As we walk the candidate path (p_r) in Section 2.2.2, for each node (k-mer; v) in the de Bruijn graph G_r, starting at a_L, we lookup v in D_r to see if a_R can be reached in a minimum of i nodes, where i is defined as the maximum allowed path length minus the number of nodes walked to reach v. If one of these conditions is not met, we abandon p_r.

The advantage of this pruning strategy is that we never explore paths that will not reach our endpoint. Additionally, we will discard any path once we have made too many loops around a graph cycle.

To illustrate the benefit of this pruning algorithm, we present an example in Figure 2.1. This graph represents G_r (T_r more specifically, as only nodes reachable from a_L are present), with the nodes representing k-mers. The anchor k-mers a_L and a_R are coloured red, and the numbers associated with each node represent the distance map D_r - indicating the length of the shortest path to a_R from that node. The lighter blue nodes and dashed edges indicate nodes that would cause us to abandon a path walk. For example, if we have stipulated a maximum allowed path length of 4, starting at

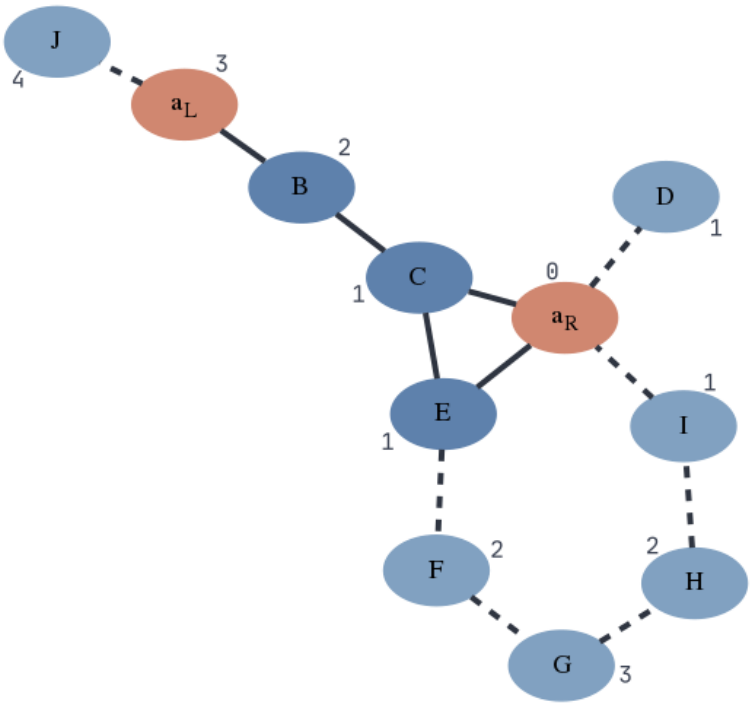


Fig. 2.1: An illustration of graph pruning. The graph represents a de Bruijn graph of a candidate region. Nodes represent k-mers and are arbitrarily labelled, except a_L and a_R (red), which are the start and end anchor k-mers, respectively. When enumerating paths, we aim to find all paths between a_L and a_R with a length no greater than a specified maximum. Numbers next to nodes indicate the length of the shortest path from that node to the end k-mer a_R. In this example, we set a maximum path length of 4. As such, light blue nodes and dashed edges indicate sections of the graph that would be pruned (not explored).

a_L, whenever we reach a light blue node, the number of steps taken to reach that node, plus the length of the shortest path from that node to a_R, will always be greater than 4. Take the path $a_L \rightarrow B \rightarrow C \rightarrow E \rightarrow F$; we have taken four steps to reach this node, yet the shortest path to a_R is 2, meaning, in the best-case scenario, p_r would have length 6. In this way, we prevent needlessly walking the section of the graph $F \rightarrow G \rightarrow H \rightarrow I$. While this is a small example, in real graphs, the saving is potentially huge.

In the end, for each candidate region (r), we are left with a collection of paths (V_r) between two k-mers (a_L and a_R). We create the final candidate paths by replacing the sequence between a_L and a_R in the maximum likelihood path (mlp_n) with each path (p_r) in V_r. These are written to file - with one file per candidate region. Padding the candidate paths in this way ensures they are inserted into the PRG in the correct location (see Section 2.2.4).

2.2.4 Updating a PanRG with candidate paths

As new paths may alter the structure of a PRG, we cannot insert them directly and must rebuild each PRG for which a candidate path is discovered.

The first step of rebuilding each locus PRG is to add the new candidate paths to the original multiple sequence alignment (MSA). We ensure the novel paths align with the correct section of the locus because we padded each candidate with the maximum likelihood path. Next, we combine all candidate paths for a locus into a single, unaligned FASTA file and add them to the existing locus MSA with the `--add` protocol in MAFFT [143].

Finally, we run `make_prg` on the subsequent alignments, and the resulting updated PRGs are combined into a single PanRG and indexed with `pandora`.

This updated PanRG can then be used as input to `pandora`, and subsequent genotyping will include the novel variants.

2.3 Initial assessment via simulations

Having described an extension of the `pandora` program that allows for *de novo* variant discovery, we now perform an initial evaluation and explore the impact of various parameters using a simulated dataset.

2.3.1 Methods

The first step in evaluating the effect of adding *de novo* variant calling to `pandora` is with a simulated dataset. We aim to show that the addition of *de novo* discovery allows `pandora` to improve its capacity for variant detection (recall) with minimal impact on the quality of the calls (precision).

To construct our simulated dataset, we randomly select 100 gene MSAs from a pool of 29,702 obtained for *E. coli* from the panX database [8]. Next, a PRG is constructed for each MSA with `make_prg` (Section 1.3.1). We used a range of `make_prg` maximum nesting levels - 1, 3, 5, and 10 - to investigate whether PRG nesting has an impact on our ability to discover novel variants. The PRGs are combined into a single PanRG for each nesting level. A random path through each PRG is selected using `pandora`, and these sequences are concatenated together to form a single "genome" sequence.

We subsequently add SNPs to the simulated genome at different per-gene rates using `snp-mutator` [144]. For this work, we introduce 100, 400, and 1,000 SNPs to

the simulated genome, which equate to approximately 1, 4, and 10 SNPs per gene, respectively. `snp-mutator` produces a VCF of the SNPs that were introduced, along with the mutated genome sequence.

Next, we simulated 30,000 Nanopore reads from the mutated genomes using `nanosim-h` [145, 146]. As the most recent model offered by `nanosim-h` was from the old R9 Nanopore flow cell, we trained and used a model from a freely-available *E. coli* R9.4 dataset (http://lab.loman.net/2017/03/09/ultrareads-for-nanopore/). Each read set was randomly subsampled to a read depth (coverage) of 15, 30, 60, and 100 with `rasusa` [147] so we can investigate the impact of coverage on our ability to discover novel variants.

`pandora`'s `discover` routine is then run, using the original panX-derived PanRG and the reads simulated from the mutated genome. Using this approach, we know that the reads originate from a sequence in our PanRG, but with some SNP differences and Nanopore errors. It is possible that some of the random SNPs introduced by `snp-mutator` already exist in the PanRG, but this is likely a minimal number. We use three different *k*-mer sizes for the *de novo* discovery: 11, 13, and 15.

After running the `discover` routine, we are left with a collection of candidate paths produced by the *de novo* component. We then add these candidate paths back into the PanRG as per Section 2.2.4. The updated PanRG is then used as input - along with the simulated reads - to `pandora map` to produce a genotyped VCF that hopefully contains all of the simulated SNPs.

In parallel to this, we also run `pandora map` on the original PanRG and simulated reads - i.e., without variant discovery. The genotyped VCF produced by this run shows how `pandora` performed before the addition of *de novo* variant discovery in this chapter. Theoretically, we only expect this VCF to contain simulated SNPs that were already in the PanRG.

At the end of this workflow, we have a genotyped VCF with and without *de novo* variant discovery for each combination of maximum nesting, *de novo* *k*-mer size, SNP rate, and read depth (coverage).

2.3.2 Evaluation

Comparing the truth VCF to the one produced by `pandora` requires care. The variants in the truth VCF are with respect to a linear reference genome; as we only simulated SNPs, these are single-position records. However, the `pandora` variants are with

respect to a graph and, depending on the density of variation in the graph, may not appear as single-position records (see Figure C.3 for an illustration of this).

We avoid the error-prone conversion of linear coordinates into graph coordinates, or vice versa, by using a coordinate-free evaluation. This approach maps variant *probes* to each other and compares the probe sequences.

We define a probe-set P as a collection of probes, p, where p represents an entry, e, in a VCF file, V. For each $e \in V$, p is constructed by the concatenation of l_w, e_c, and r_w (in that order), where e_c is the called allele of e, and l_w and r_w are the sequences, of maximum length w, in the VCF reference to the left and right, respectively, of e_c. For `pandora`, the VCF reference is the maximum likelihood sequence, and for the truth VCF it is the simulated genome (without the simulated SNPs).

A truth probe-set, P_t, was constructed from the VCF of variants added to the simulated genome and a query probe-set, P_q, from the variants called by `pandora`. We then mapped all probes from P_t to P_q using `bwa mem` [25]. We classify each probe in P_t as a true positive (TP) if the e_c part of the probe exactly matches the sequence it aligns to in P_q, or a false negative (FN) otherwise. Any probe in P_q that does not have a TP truth probe mapped to it is classified as a false positive (FP). We perform this assessment for the "no *de novo*" and "with *de novo*" VCF files from `pandora`.

Precision is defined as the number of TPs divided by the number of TPs and FPs $precision = \frac{TP}{TP+FP}$; it represents the fraction of variant calls made that are correct. Likewise, recall is calculated as $recall = \frac{TP}{TP+FN}$ and describes the proportion of expected variants correctly discovered.

2.3.3 Results

We first look at Figure 2.2, which shows how precision and recall of the `pandora` *de novo* variants changes depending on the combination of parameters chosen. Those parameters were the read depth (coverage) of the simulated reads (Figure 2.2a), the number of SNPs introduced into the simulated genome (Figure 2.2b), the k-mer size used for variant discovery (Figure 2.2c), and the maximum nesting level allowed in the PRGs (Figure 2.2d). In total, there are 144 different combinations of parameters, and thus data points.

The parameter that appears to have the most visible impact on the precision and recall is the coverage (Figure 2.2a). It is somewhat unsurprising that as coverage increases, so do both precision and recall. However, there does not seem to be any noticeable difference for coverage ≥ 60.

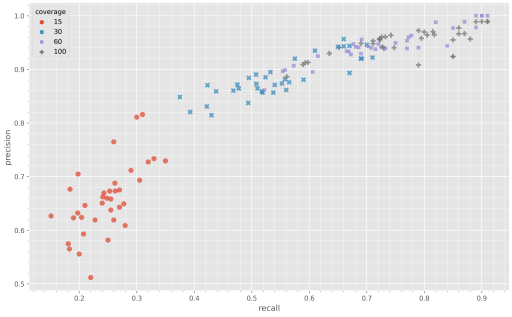
In the best case, the highest recall and precision values are 0.91 and 1.0, respectively. The data point is the same in both instances, with a coverage of 60, k-mer size of 13, number of SNPs 100, and a maximum nesting level of 10. Upon further investigation of the nine missed variants (FNs) for this data point, six were within $2k-1$ positions of the start or end of the locus, one was a null call (indicating genotyping uncertainty), one was falsely called as a homopolymer deletion, and the remaining missed call never had *de novo* discovery triggered for that region of the locus. Therefore, only 3/9 FNs for this example (the last three) were discoverable with our *de novo* method.

The last point requires some elaboration, as it may not be clear why only three FNs in the best-performing data point were expected to be detected by *de novo*. As a reminder, the role of the *de novo* component is to collect candidate alleles that are potentially in the sample but missing from the graph; if that set includes the truth, it has succeeded. Whether or not that true allele is genotyped as being present and recorded in the VCF is the job of the sequence inference and genotyping components of `pandora`. In the case of the null genotype call, the correct variant was discovered, and it had higher coverage than the reference allele (26 vs 11); therefore, it is a failure of the genotyping. The homopolymer deletion is a failure of the genotyping; while *de novo* (incorrectly) discovered the candidate indel, it also discovered the correct allele, but the genotyping chose the homopolymer deletion. Furthermore, the variant which *de novo* discovery was never initiated for most likely has enough coverage on the reference allele that a candidate region was not detected - by default, we only flag a candidate region if coverage drops below 3.

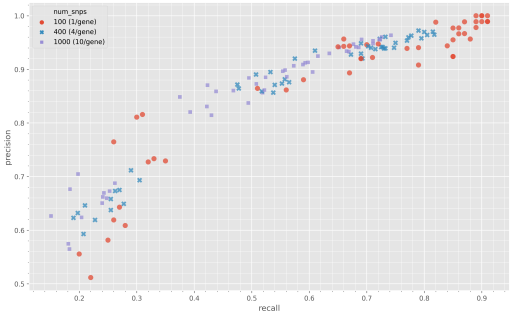
In the case of the six missed calls near the ends of loci, these are not detectable by our current *de novo* method as they occur within $2k-1$ positions of the boundary of a locus. The reason this makes them undetectable is related to our need for start and end anchor k-mers in order to find candidate paths (Section 2.2.2). The start and end k-mers are a collection of k k-mers, meaning $2k-1$ positions are required surrounding a candidate region in order to be able to initiate *de novo* discovery. We will return to this limitation later (Section 2.6.1).

While not an issue for the best-performing example we have just been examining, missing loci were another common source of FNs. If `pandora` decides a locus is not present after quasi-mapping (Section 1.3), then it is impossible for *de novo* to discover any variants in it. We note that the vast majority of missing loci have a length of less than 250 base pairs (Section A.1).

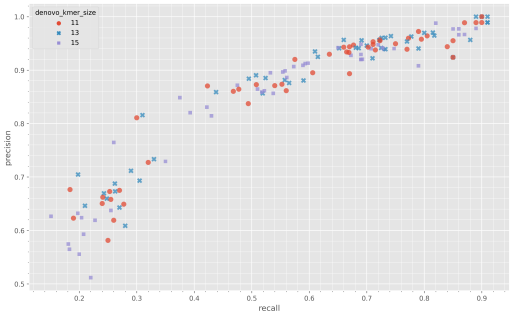
(a) Simulated coverage (read depth)



(b) Number of SNPs simulated



(c) *de novo* discovery *k*-mer size



(d) PRG maximum nesting level

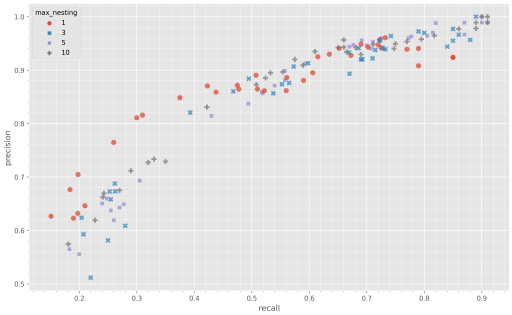


Fig. 2.2: Recall (x-axis) and precision (y-axis) of *de novo* variants discovered by `pandora` on a simulated dataset. Subplots style the points by the parameter indicated in the subtitle. Each point indicates a single run of `pandora` with a unique combination of all parameters.

	Max. nesting	*de novo* *k*-mer size
Precision	0.934 (10)	0.937 (13)
Recall	0.674 (5)	0.671 (13)

Table 2.1: The median precision and recall for all parameter combinations, grouping by the maximum PRG nesting level or the *de novo* *k*-mer size used for variant discovery in `pandora`. The values in parentheses indicate the parameter value that leads to the specified precision or recall.

When looking across all 144 combinations of parameters, we found that, on average, 7.8% of the true variants are near the ends of loci and 2.8% are in absent loci (see Figure 2.3).

The parameters that we can directly control with respect to *de novo* discovery within `pandora` are the PRG maximum nesting level and the *de novo* *k*-mer size. Figure 2.2 shows no clear optimum for either of these options. However, when taking the median precision and recall values across all data points (Table 2.1), a maximum nesting level of 5 and *de novo* *k*-mer size of 13 seem the best choice.

For the final analysis of the simulation data, we look at how the precision and recall change with an increasing genotype confidence threshold. We select the data point with the optimal maximum nesting level (5) and *de novo* *k*-mer size (13), along

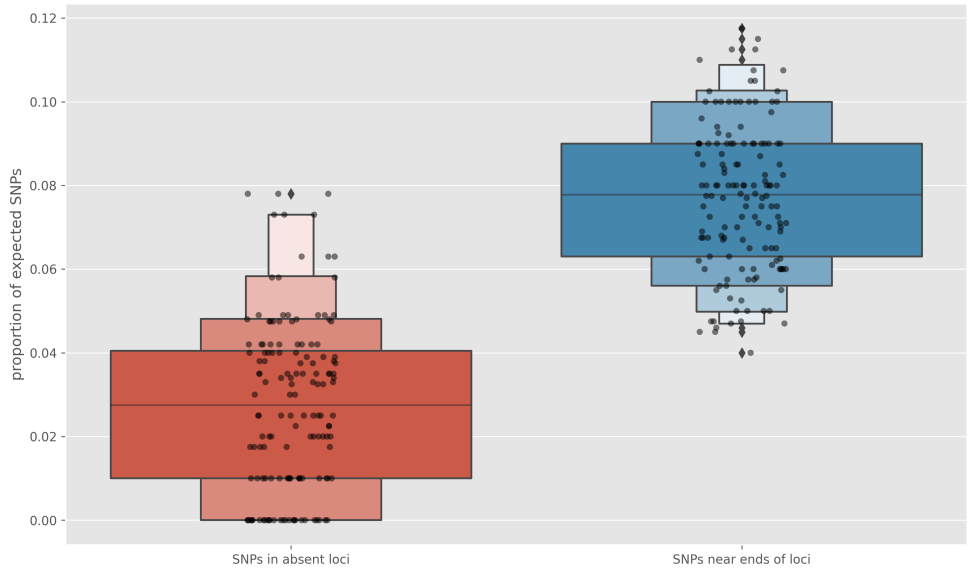


Fig. 2.3: The proportion of simulated SNPs that are not detectable by *de novo* variant discovery. The red box represents SNPs that occur in loci designated as absent by `pandora`. The blue box depicts the SNPs that occur within $2k-1$ positions of the start or end of a locus. Each point indicates a single run of `pandora` with a unique combination of parameters.

with 4 SNPs per gene, as this is within the range expected for an *E. coli* genome. Next, starting at 0 and increasing by 10 until 700, we filter out any variant with a genotype confidence score below the current threshold. The purpose of this analysis is to illustrate what the cost on recall is for requiring more confident variant calls at different read depths.

Figure 2.4 shows the same relationship we saw earlier: coverage has a significant impact on precision and recall. Most importantly, though, it shows that the inclusion of *de novo* discovery is vital for finding novel variants. Precision and recall for `pandora` *without* variant discovery are not shown in Figure 2.4, as the best recall achievable for this set of parameters was only 1.0% (indicating that 4 SNPs were incidentally in the PanRG). This is compared to a maximum of 81.5% when using *de novo* discovery. Focusing on the 100x coverage data point with *de novo* discovery, the best recall (81.5%) leads to a precision of 97.0%, but the cost of increasing precision to 99% is decreasing recall to 25%.

2.3.4 Summary

In summary, we have shown that the addition of the *de novo* variant discovery method outlined in Section 2.2 gives `pandora` the ability to find many variants not present in its PanRG. Using simulated data, we find a k-mer size of 13 gives slightly better recall

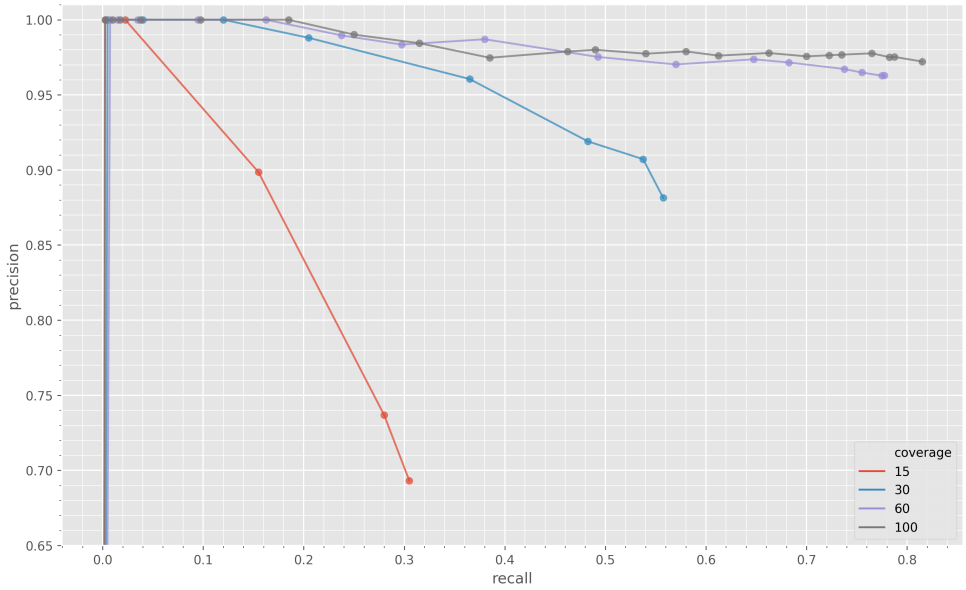


Fig. 2.4: Precision-recall curve for increasing genotype confidence score thresholds. The curves are coloured by read depth (coverage). Each marker/point is a different genotype confidence threshold, starting with 0 as the right-most value and increasing as the line moves towards the (top) left. Note, the y-axis has been cut to provide more clarity of precision.

than 11 or 15, but that the sample read depth has the most considerable impact on our ability to discover novel variants.

We have also shown that, on average, approximately 10.5% of (simulated) SNPs are not detectable based on their membership in either loci `pandora` does not detect, or lying within $2k - 1$ positions of the start or end of a locus.

2.4 Validation with empirical data

Having shown, for simulated data, that the *de novo* variant discovery method indeed alleviates a major limitation of `pandora`, we turn our attention to evaluating its performance on empirical data. However, rather than the single-sample (`pandora map`) approach we used for genotyping in Section 2.3, we evaluate `pandora`'s multi-sample comparison protocol - `pandora compare`.

The inference of variation within a collection of samples is one of the unique aspects of `pandora`. As detailed in Section 1.3.3, the `compare` routine of `pandora` genotypes multiple samples against a PanRG simultaneously, with the aim of representing variation between the samples in the most succinct manner possible (see Figure 1.6b). It does this by selecting a single maximum likelihood path that best approximates the samples under investigation - akin to an "average" path of the samples.

Evaluating the variant calls from such a graph-based method is by no means trivial. In this section, we aim to compare the variant calls made by `pandora` against those made by single-reference (linear) methods for both Illumina and Nanopore data.

In keeping with the focus of this chapter, we will also assess the utility of *de novo* variant discovery within `pandora`. While we have shown its benefit on simulated data, the PanRG used did not contain many of the simulated SNPs. However, we expect many of the SNPs in real data also to be present in a PanRG built from a pan-genome of diverse samples.

Note: all work in this section (Section 2.4) is described in full in [10]. See Section 2.0 for a detailed description of what work was completed by myself.

2.4.1 Dataset

Samples

The empirical data we use for this evaluation is a diverse set of 20 *E. coli* samples from four different phylogroups. Each sample has both Nanopore and Illumina sequencing data and high-quality assemblies available. The sequencing reads for each technology were subsampled to read depth 100x.

References

The PanRG we use as the reference for `pandora` was constructed from a combination of *E. coli* genes and intergenic regions. 23,054 gene MSAs from 350 RefSeq genomes were obtained from the panX database [8], while 14,374 intergenic region MSAs from 228 ST131 genomes were collected from [148]. PRGs were constructed for each locus using `make_prg` and then all were combined into a single PanRG file and indexed with `pandora`.

As single-reference variant callers cannot use a PanRG, we selected 24 reference genomes from five major phylogroups - one phylogroup is not contained within the sample set. These references were selected to be spread across phylogroups, and where the phylogroup was present in our samples, we chose the nearest RefSeq genome according to Mash, and a phylogenetic tree [10]. By calling variants for each sample with respect to each reference genome, we can directly view the impact of reference selection on the results of standard variant callers.

2.4.2 Variant calling

Graph-based: Pandora

To produce a multi-sample VCF file of variants for `pandora` we follow a somewhat similar approach to Section 2.3.1, with some important differences. Rather than adding all *de novo* variants for a single sample to the original PanRG we instead add the novel variants for *all* samples to the original PanRG. In the end, we have an updated PanRG which contains all novel variants for all samples under comparison. We then perform multi-sample genotyping with this updated PanRG using `pandora compare`. This entire process is completed separately for Illumina and Nanopore data.

Linear-based

We compare the Illumina variant calls from `pandora` against those from Samtools [30] and Snippy (https://github.com/tseemann/snippy) (which is a wrapper around Freebayes [28]). The Nanopore variant callers we evaluate against are Medaka (https://github.com/nanoporetech/medaka) and Nanopolish [149]. Each variant caller is run on all 20 samples with all 24 reference genomes.

In total, we produce 480 VCFs for each linear variant caller and 20 (one per sample) for `pandora`.

2.4.3 Evaluation

A direct comparison of the VCF files produced by `pandora` and the single-reference tools is not possible due to coordinate incompatibilities. As such, we use a probe-based method, akin to that in Section 2.3.2 for assessing the variant calls.

The first step in this evaluation is the generation of a pan-genome SNP truth set. Section A.2 details the construction of this truth set, resulting in 618,305 SNPs we expect to find amongst the 20 samples.

There are two measures of recall in a pan-genome (see Section A.2), but of interest to this section is the average allelic recall (AvgAR). Briefly, AvgAR is the average recall of all pan-genome variants. For example, we have three genomes with two pan-genome variants P_1 and P_2 between them. P_1 has the alleles A, A, and T across the three genomes, and P_2 has alleles C, T, and T. If we find the P_1 alleles A (for only one sample) and T, we have a P_1 recall of 0.66 (2/3), and if we only find the P_2 allele

C, we have a P_2 recall of 0.33 (1/3). Therefore, in this example, we have an AvgAR of $\frac{0.66+0.33}{2} = 0.5$.

To calculate AvgAR (recall) for tool-reference pairs, we perform the following: i) apply the variant calls for each sample to the reference sequence the calls were made with respect to (giving 20 mutated sequences); ii) we map all truth set probes to these mutated reference sequences with `bwa mem`; iii) we classify a mapping as TP if the alleles within the aligned sequences match. Then, for each pan-genome variant, we count the proportion of its alleles with a TP mapping and calculate AvgAR accordingly. In the end, we have an AvgAR value for each variant caller and reference sequence combination - i.e., 24 AvgAR values for each caller.

We determine precision in a somewhat similar manner. First, we create probes for each variant in a given VCF, with 150bp of flanking sequence taken from the VCF reference sequence. Second, we map each probe to the sample's assembly sequence. Next, we filter out poor quality mappings or mappings to low-quality regions of the assembly. Finally, each mapping's precision is classified as a continuous score - rather than a binary true or false positive - by dividing the number of matching bases (ignoring the flanking sequences) by the alignment length. Thus, for example, if the called allele is ATG and maps to a sequence ATTG, the precision score is 0.75. Ultimately, we calculate precision as the sum of precision scores, divided by the number of evaluated calls.

2.4.4 Effect of different Nanopore basecalling models

Previous work from Wick *et al.* has shown that for Enterobacteriaceae, the majority of Nanopore sequencing errors are related to Dcm methylation sites [78]. In version 3.2.1 of `guppy` (the ONT-provided basecalling software) a new *methylation-aware* model was made available. However, this new model is considerably slower to basecall reads than the default model. As *E. coli* is a member of this family, we set out to test a subset of 4 samples from our dataset to see whether a methylation-aware model indeed has a noticeable impact on the precision and recall from `pandora` - with and without *de novo* variant discovery.

We basecalled the raw data for four samples from our dataset with both the default and methylation-aware models from `guppy` version 3.4.5. Precision and recall (AvgAR) are calculated as per Section 2.4.3 and presented in Figure 2.5.

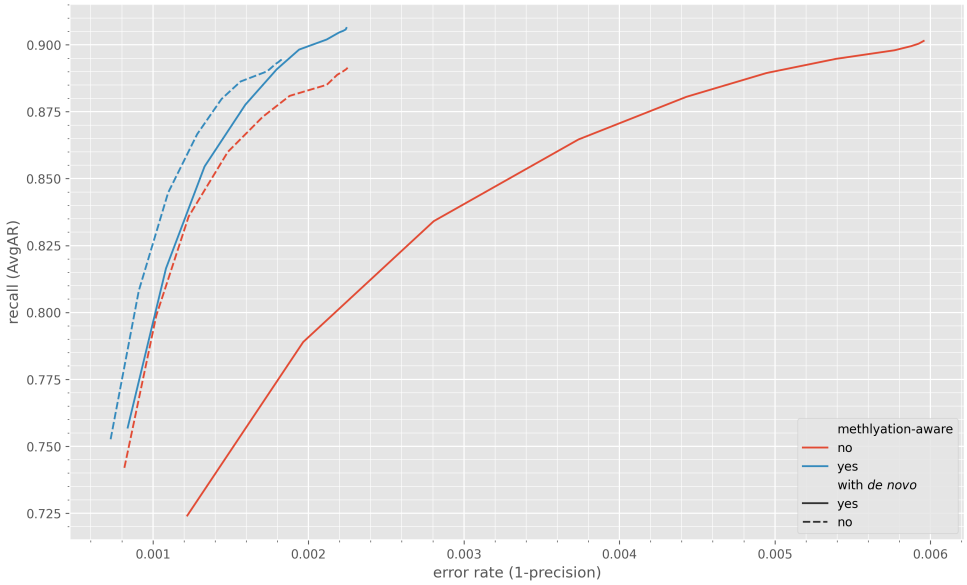


Fig. 2.5: Effect of Nanopore methylation-aware basecalling on `pandora` *de novo* variant discovery. The lines show the average allelic recall (AvgAR; y-axis) and error rate (1−precision; x-axis) of `pandora` with increasing genotype confidence score thresholds with (solid line) and without (dashed line) *de novo* variant discovery. The red line shows data basecalled with the default guppy model and blue being basecalled with a methylation-aware model.

Source: Adapted from [10] under the terms of the Creative Commons CC BY license. The colour scheme has been altered, along with the wording of some labels. However, none of these changes in any way alters the underlying data or interpretation of that data.

The precision-recall curves represent increasing genotype confidence filtering; the top-right of each curve is no filtering, and as the genotype confidence requirement is gradually increased, we reduce the error rate (increase precision) at the loss of recall.

Two crucial observations from Figure 2.5 are that the use of a methylation-aware model increases both precision *and* recall, and the use of *de novo* discovery increases recall at the cost of precision.

Table 2.2 shows the precision and recall values for the unfiltered results (i.e., the top-right of each curve in Figure 2.5). From this, we see that without *de novo* variant discovery, using a methylation-aware model would allow us to recover 894.7 variants in 1000 - 3.3 more than with the default model; likewise, we would expect to make 0.5 errors per 1000 variants. Using *de novo* variant discovery, the methylation-aware model allows us to discover 906.3 variants in 1000 - 4.8 more than the default model and 11.6 more than methylation-aware without *de novo* discovery. In terms of errors, using a methylation-aware model leads to 3.7 less errors per 1000 variants with novel variant discovery enabled; however, it makes 0.41 more errors per 1000 variants than without novel variant discovery.

Methylation-aware	with *de novo*	Recall (AvgAR)	Error rate (1−precision)
no	yes	0.9015	0.0060
	no	0.8914	0.0023
yes	yes	**0.9063**	0.0022
	no	0.8947	**0.0018**

Table 2.2: Effect of Nanopore methylation-aware basecalling on `pandora` *de novo* variant discovery (unfiltered) error rate (1−precision) and average allelic recall (AvgAR).

Given the dramatic improvement in precision and recall from using the methylation-aware basecalling model, we re-basecalled all data for subsequent analyses with this model.

2.4.5 Performance of Pandora against single-reference tools

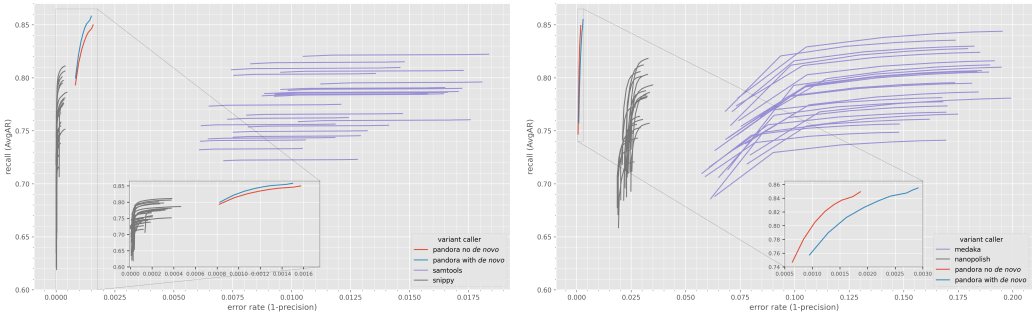
We now compare the precision and recall of `pandora` on Illumina and Nanopore data against two single-reference variant callers for each technology. For the Nanopore analysis, we use the reads basecalled with the methylation-aware model (Section 2.4.4). In addition, we run `pandora` with and without *de novo* variant discovery to see the impact of the work in this chapter.

We use AvgAR as the measure of recall (see Section 2.4.3), error rate as the measure of precision (1−precision), and apply increasing genotype confidence thresholds to illustrate the precision-recall trade-off.

The results of this analysis are shown in Figure 2.6. For both sequencing technologies, `pandora` with *de novo* variant discovery has the highest (unfiltered) recall of 85.82% on Illumina data and 85.51% on Nanopore. In terms of error rate, `snippy` had the lowest Illumina unfiltered value of 0.01% (with reference CP010170.1), while `pandora` without *de novo* variant discovery had the lowest Nanopore unfiltered rate of 0.19%.

Of particular interest to the work in this chapter is the observation that, on Illumina data, *de novo* variant discovery leads to greater recall and precision (see inset of Figure 2.6a); however, on Nanopore data, *de novo* discovery provides greater recall, but at the cost of lower precision (see inset of Figure 2.6b). This suggests that systematic errors in Nanopore create incorrect novel alleles, which are in turn deemed correct by genotyping.

The most striking result from this work, though, is the error rate of `pandora` on Nanopore data, which is 12.9 times lower than `nanopolish` and 79 times lower than



(a) Illumina **(b)** Nanopore

Fig. 2.6: Precision-recall curve for variant calls from Illumina (left; **(a)**) and Nanopore (right; **(b)**) data for different variant calling tools (colours). Each curve represents increasing genotype confidence thresholds, starting with no filtering in the top-right of each curve and increasing towards the bottom-left. `pandora` has a single line with (blue) and without (red) *de novo* variant discovery, which represents the error rate and recall across all 20 samples. The other variant callers are traditional single-reference tools; therefore, each line represents the use of 1/24 reference genomes from across five major *E. coli* phylogroups. Inset windows are used to give more granularity for error rate in the best performing tools.

Source: Adapted from [10] under the terms of the Creative Commons CC BY license. The colour scheme has been altered, along with the wording of some labels, and the inclusion of inset windows. However, none of these changes in any way alters the underlying data or interpretation of that data.

`medaka`. In real terms, this equates to 22 and 146 fewer errors per 1000 variants, respectively.

Equally impressive is the improvement in recall over `snippy` using `pandora` with variant discovery on Illumina data, leading to 47/1000 more variants being discovered. However, this does come at the expense of a higher error rate than `snippy`.

2.4.6 Summary

In this section, using 20 diverse *E. coli* samples, we have shown that the *de novo* variant discovery method outlined in Section 2.2 allows `pandora` to discover more variants (increases recall). In addition, it improves the error rate on Illumina data.

We also demonstrated that a methylation-aware Nanopore basecalling model decreases the `pandora` error rate - quite significantly for *de novo* - indicating that many of the novel variants "discovered" by our method are, in fact, systematic technology errors.

Finally, we show that `pandora` provides higher recall than single-reference-based variant callers for both Illumina and Nanopore data and leads to a Nanopore error rate that is an order of magnitude lower than other tools. However, `snippy` provides lower error rates than `pandora` on Illumina data.

2.5 Discussion

In this chapter, we described a method for discovering *de novo* variation in a genome graph from both Illumina and Nanopore data. We implemented it in the `discover` subcommand of the reference graph program `pandora` and evaluated its utility on both simulated and empirical data. Additionally, we demonstrate an approach for comparing variant calls made from single-reference or graph-based tools independent of genome coordinates.

Before the work in this chapter, `pandora` was only able to genotype with respect to variation present in a pan-genome reference graph. To allow `pandora` to discover novel variants, we use a localised form of genome assembly in segments of the graph with low k-mer coverage. We first identify candidate regions of the genome where read depth (observed as k-mer coverage) drops below a predefined threshold. Next, we slice out the segments of the reads that map to these candidate regions and construct a de Bruijn graph from them. Finally, we enumerate paths in this de Bruijn graph that pass coverage and insertion size filters and output them as novel candidate alleles. These alleles can then be added into the original PanRG to allow for genotyping against these new alleles.

This *de novo* discovery process, detailed in Section 2.2, is somewhat analogous to the method used by GATK's HaplotypeCaller [31]. With two noticeable differences. First, GATK determines candidate regions (described as *ActiveRegions*) based on an "activity score" that is calculated from alignment quality and genotype likelihood at a locus. In contrast, we use a naive approach that looks for k-mers in the maximum likelihood path with coverage less than a hard threshold. While our approach may seem simple, it relies on information gained by performing quasi-mapping to the graph and inference of a maximum likelihood path. Second, the pruning methods employed by GATK involve removing edges with low support, and when the terminal k-mer of a path does not match the reference sequence, they attempt to use local alignment to merge it. By comparison, we require a path contain the terminal k-mer, thus avoiding local alignment. Additionally, we prune the de Bruijn graph by using depth- and breadth-first search to construct a distance map so that we know whether a given k-mer in the de Bruijn graph can reach the terminal k-mer within a predefined number of graph walks. In this way, we never get stuck in cycles or produce paths longer than a user-defined insertion size.

While we did not compare `pandora` runtime to GATK, given the results in [31] and the algorithmic details of their method, we suspect our method leads to much faster runtimes due to the pruning of paths and lack of local alignment.

In Section 2.3 we selected random paths from a PanRG and randomly introduced SNPs to each at varying rates. We then simulated Nanopore reads from these mutated random genomes and used them to test different parameters relating to our *de novo* discovery method. Read depth (coverage) had the most obvious impact on the ability to detect novel variants. Given the simulated Nanopore read error rate of approximately 11%, the decrease in recall with coverage is expected as our method relies on k-mer coverage along the candidate paths. Without sufficient k-mer coverage to support a path, we cannot produce any candidate alleles. While a k-mer size of 13 for *de novo* assembly was found to perform slightly better than 11 and 15, this was only marginal and may warrant further investigation. Indeed, for the work in [10] we used a *de novo* k-mer size of 15 in order to speed up variant discovery in some problematic loci.

While we compared the precision and recall of `pandora` with and without variant discovery in the simulations, this was not a realistic scenario. However, these simulations were not intended to be entirely realistic (that was the purpose of Section 2.4). They were designed to illustrate the limitations of `pandora` without the capacity to find novel variation and allow us to explore the impact of various parameters on variant discovery.

Given the contrived nature of the simulation analysis, in Section 2.4 we used an empirical dataset of 20 *E. coli* samples from across four major phylogroups to examine the benefit of *de novo* variant discovery. In performing this analysis, we devised a novel method for comparing variant-calling precision and recall across tools in a coordinate-agnostic manner. The main benefit of such an approach is the mitigation of hard reference bias, which occurs when the reference genome used for variant calling (or mapping) does not contain a locus (see Figure 1.3). As the pan-genome of *E. coli* is open, hard reference bias is a problem when comparing cohorts of diverse samples. Indeed, as shown in Figure 7 of [10], `pandora`'s use of a locus-specific reference that is dynamically selected based on the cohort under study leads to tens of thousands more SNPs being discovered in rare loci (loci in 2-5 genomes) compared to single-reference variant callers.

An important finding from our empirical analysis is the impact of Enterobacteriaceae methylation on `pandora`'s Nanopore error rate (Figure 2.5). In particular, when using *de novo* discovery, the Nanopore error rate is 3-fold lower when a methylation-aware basecalling model is utilised. This observation is also confirmed by Wick *et al.*

in their work benchmarking Nanopore basecalling methods for Enterobacteriaceae [78]. They saw that the overwhelming error source was in Dcm methylation sites and that these errors could be virtually erased by training a taxon-specific basecalling model. It follows that given the dramatic decrease in *de novo* errors as a result of using the methylation-aware model, these systematic biases are being captured and incorporated back into the PanRG by variant discovery. Therefore, any Nanopore work involving genomes subject to Dcm methylation would be wise to employ methylation-aware basecalling. Other tools such as `nanopolish` have incorporated error modelling to remove these systematic biases [149]. However, as Nanopore sequencing evolves at a rapid pace, the constant maintenance required to model these errors can become very laborious. As such, we feel that taxon-specific basecalling models are a more robust solution to these systematic errors (we explore this further in Chapter 5).

Despite the impact of Nanopore systematic errors on *de novo* variant discovery, we did find that for both Illumina and Nanopore, *de novo* discovery increased the recall of `pandora` (Figure 2.5 and Figure 2.6). In addition, we also found that *de novo* discovery provided lower error rates on Illumina data (Figure 2.6a). As the Illumina and Nanopore data for all samples are matched - i.e., from the same DNA extraction - the difference in error rate between *de novo* and no-*de novo* must be technology-driven. However, we do note that the `pandora` error rate for Nanopore was an order of magnitude lower than both `medaka` and `nanopolish`.

Without the addition of *de novo* variant discovery, `pandora`'s recall is higher than all other variant callers tested - although only marginally higher than `medaka`. This shows the power of genome graphs to provide access to sections of the genome previously unavailable to single-reference methods. Now that we have provided the functionality for `pandora` to perform novel variant detection, the door is open to access even more of the pan-genome.

One application that stands to benefit from pan-genome graphs is prospective surveillance of a bacterial population. Current approaches to such surveillance perform multiple levels of genome alignment in order to find the most appropriate reference for a given sample [150]. In contrast, the PanRG used by `pandora` is a stable reference capable of handling an evolving cohort.

2.6 Limitations and future work

2.6.1 Path enumeration

A fundamental limitation of the *de novo* variant discovery method outlined in this chapter is the need for anchor k-mers to initiate local assembly and perform path enumeration. While this is not concerning in most locus areas, it becomes problematic near the start and ends of loci (within $2k-1$ positions of the ends). Indeed, we see in Figure 2.3 that on simulated data, variants that occur near the ends of loci account for a recall loss of 8%.

One solution for removing this anchor k-mer limitation would be to use a unique k-mer approach analogous to Kevlar [151]. Briefly, Kevlar aims to detect *de novo* variants (in human trios/families) by looking for unique k-mers in a child, with respect to the parents. Reads containing these "interesting" k-mers are then assembled with an overlap graph, and the resulting contig is aligned to and genotyped against the reference.

Our idea for how the Kevlar method could be adapted to `pandora` is to identify candidate regions and extract the read pileups for these regions as we currently do (Section 2.2.1). Then, for each candidate region, perform the following: i) let M be the set of all minimizer k-mers along the maximum likelihood path in the candidate region; ii) construct a k-mer count table from the read pileup of the candidate region, only counting minimizer k-mers not in M; iii) filter out any k-mer with a count less than some threshold determined by the number of reads in the pileup, leaving a set of unique minimizer k-mers N. We would then construct a de Bruijn graph from only those reads containing k-mers in N. The enumeration of paths through this graph would then involve beginning from any one of the k-mers in N - removing the need for anchors - and only keeping those paths that contain a k-mer in N. One element of this strategy that requires careful thought is how to incorporate paths back into the graph without the anchor k-mers we currently use.

2.6.2 Cycle detection

The primary computational bottleneck we have encountered during *de novo* variant discovery development is infinite cycles in the de Bruijn graph. As de Bruijn graphs are directed, but not necessarily acyclic, a cycle can occur when a k-mer (node) can reach itself by following a path from any of its successor (out) nodes. When multiple such cycles occur within a graph, computational performance suffers. Such cycles

can be detected using depth-first search; however, we do not want to disallow them as cycles are not invalid biologically.

To limit becoming stuck in cycles, we employ pruning heuristics (Section 2.2.3) that prevent paths from reaching a maximum length or the exploration of paths that will never reach the end anchor *k*-mer. However, if many cycles occur in close succession, even these heuristics do not prevent bottlenecks. Moreover, in some low complexity PRGs, we have seen this process take on the order of days for a single candidate region. As such, future work to improve the computational performance would be well-served to investigate additional pruning strategies.

2.6.3 Nanopore homopolymer deletions

Wick *et al.* found Dcm methylation sites to be the primary source of errors in Enterobacteriaceae, and when these were eliminated with their taxon-specific model, homopolymer deletions became the chief error source [78].

While we did not report the number of homopolymer deletions we found, we did discover one as the source of 1/9 false negatives in our best-performing parameter set in the simulations (Section 2.3.3). As mentioned, some methods attempt to remove these errors as part of their model explicitly. One such possibility would be to refuse to perform *de novo* discovery on candidate regions containing homopolymer sites. However, we feel such an approach is too coarse and explore a potential solution in Chapter 5 that entails training a species-specific Nanopore basecalling model.

2.6.4 Inserting novel alleles into the PanRG

In Section 2.2.4 we described a somewhat convoluted process for inserting candidate paths found by *de novo* discovery into a PanRG. This insertion approach is another of the main computational bottlenecks we encountered due to the need to perform a multiple sequence alignment.

Leandro Ishi has been working on a prototype version of `make_prg` (https://github.com/leoisl/make_prg) that facilitates much faster insertion of novel variants into a PanRG. When building a PRG, it additionally produces a customised data structure that acts as a way of remembering how sequences were clustered and collapsed. In addition, a prototype version (0.9.0) of `pandora` changes the way *de novo* discovery produces candidate paths. Rather than outputting them as sequences flanked by the maximum likelihood path sequence, it describes them in a custom format similar to

VCF, with the addition of information about the PRG site the novel variant occurs in. Leandro has then implemented a new routine within `make_prg` - `update` - which takes these custom data structures and attempts to add the novel variants directly into the PRG, only requiring the relevant sites to be recalculated.

2.6.5 Insertions and deletions

A limitation we regret is not including indels in the simulations (Section 2.3). Evaluating indels can become quite complex, and we suspect fine-tuning our method to produce high-quality indel calls will require a lot of time and care. On the other hand, simulations are the ideal environment in which to begin this work, as gathering a truth set of indels for empirical data is notoriously tricky. Thus, future work will begin by including indels in these simulations and then moving to well-characterised indel models for empirical data such as [152].

2.7 Availability of data and materials

The `pandora` software is available at https://github.com/rmcolq/pandora under an MIT license. The *de novo* method described in this chapter is implement in the command `pandora discover`.

The analysis pipeline for the simulations in Section 2.3 is available at https://github.com/mbhall88/pandora_simulations. The data used for the PanRG in these simulations was obtained from https://pangenome.org/Escherichia_coli. The Nanopore reads used in the simulations was downloaded from Nic Loman's blog post http://lab.loman.net/2017/03/09/ultrareads-for-nanopore/.

All data and materials for the empirical data analysis in Section 2.4 is described in [10] and the corresponding GitHub repository https://github.com/iqbal-lab-org/paper_pandora2020_analyses.

Chapter 3

Nanopore sequencing for *M. tuberculosis* transmission clustering

3.0 Publication and collaboration acknowledgements

A manuscript comprising the work in this chapter and Chapter 4 is currently in preparation. The DNA extractions and sequencing of the data in this chapter were performed by: Marie Sylvianne Rabodoarivelo and Simon Grandjean Lapierre for the Madagascar samples; Anzaan Dippenaar, Anastasia Koch, and Helen Cox for the South African samples; Sara Goodwin at the Next Generation Genomics Core within Cold Spring Harbor Laboratory performed the PacBio sequencing for the Madagascar samples; and Sophie George, Grace Smith and Esther Robinson for the English samples. Fan Yang-Turner of the Nuffield Department of Medicine, University of Oxford, performed the variant-calling of all Illumina samples. All other work in this chapter is my own.

3.1 Introduction

Mycobacterium tuberculosis is the causative agent of the infectious disease tuberculosis (TB). It accounts for more deaths than any other pathogen each year [103] — as such, the epidemiology of *M. tuberculosis* transmission is of the utmost importance. Whole-genome sequencing (WGS) has established itself as a vital tool for identifying possible transmission clusters and is being used by some leading public health agencies to aid contact tracing [153, 154]. Illumina is considered the gold standard for this type of WGS work. However, Illumina is not readily available in many high-burden TB

settings and requires considerable time and resources to start and maintain. Nanopore has shown itself to be adept in these types of settings, having been used to notable effect during recent Zika [92] and Ebola [93, 94] outbreaks. Even in environments where resource availability is not an obstacle, Nanopore's rapid turnaround time has been used for monitoring COVID-19 and informing infection control measures [155]. The time and resources required to set up Nanopore sequencing are far lower than Illumina, but despite this, there has been little work done to assess its suitability for *M. tuberculosis* WGS-based transmission clustering. The lack of work in this space likely stems from the long-held belief that due to its higher sequencing error rate, Nanopore is not capable of such fine-grained analyses. However, Nanopore has seen considerable improvements in its accuracy in recent years [78], and studies using variant calls from the technology are becoming increasingly common [85, 156]. In particular, Public Health England (PHE) has investigated the use of Nanopore for the analysis of Shiga toxin-producing *E. coli* and found it to be well-suited to the application [86].

In this chapter, we evaluate whether Nanopore sequencing can provide *M. tuberculosis* transmission clusters consistent with Illumina. To facilitate this investigation, we collect a new dataset of 150 samples - from Madagascar, South Africa, and England - sequenced on both Illumina and Nanopore platforms. We first assess Nanopore variant calls and outline a filtering strategy to provide Illumina-level precision. Next, we use these variant calls to cluster samples based on SNP (single nucleotide polymorphism) distance thresholds and find Nanopore does not miss any samples from their expected cluster. Finally, we confirm that reliable clustering of samples from a mixture of Illumina and Nanopore modalities is achievable.

M. tuberculosis has a "closed" pan-genome; all species members share most (but not all) gene content. In Chapter 2 we sought to improve variant calling of bacterial pan-genomes with genome graphs. The work in that chapter concentrated on *E. coli*, which has an "open" pan-genome. In the interest of understanding how such genome graph methods can aid in closed pan-genomes, we additionally assess transmission clusters produced from `pandora` variant calls. We construct two *M. tuberculosis* reference graphs from different densities of population variation towards this end. While the clustering from `pandora` does not perform to the standards of the single-reference caller BCFtools, we gain many insights for the improvement of `pandora` and the construction of genome graphs.

3.2 Dataset

The data used for the work in this chapter, and Chapter 4, are patient-derived *M. tuberculosis* isolates from culture. We gathered samples from Madagascar (118), South Africa (83), and England's National Mycobacteria Reference Service in Birmingham (46), giving us a total of 247 samples. Each sample was sequenced on both Nanopore and Illumina platforms. We aimed to perform all sequencing for a sample from a single DNA extraction. Performing all sequencing on the same DNA extract ensures that any variation identified between technologies for the same sample would be due to differences in the sequencing platform and not *in vitro* evolution. As these samples are not reference isolates, and we want to be able to compare both Illumina and Nanopore to a "truth", we also sequenced 35 of the Malagasy isolates with PacBio.

3.2.1 Illumina sequencing

An extended description of isolate selection, DNA extraction, and sequencing methods is provided in Section B.1.

Madagascar

Illumina sequencing was carried out on the HiSeq 2500 platform at the Wellcome Trust Centre for Human Genetics, Oxford, and paired-end libraries were prepared according to the manufacturer's instruction.

England

Illumina sequencing was performed on a MiSeq instrument at Public Health England (Birmingham) by Grace Smith, Esther Robinson and their team. Sample preparation and sequencing methodology were as described previously [131].

South Africa

Paired-end genomic libraries were prepared using the Illumina Nextera XT library or NEBNext Ultra TM II FS DNA Library Preparation Kits (Illumina Inc, San Diego, CA, USA) according to the manufacturers' instructions. Pooled samples were sequenced on an Illumina HiSeq2500 or NextSeq500 instrument.

3.2.2 Nanopore sequencing

Madagascar

Nanopore library preparation was carried out using the Oxford Nanopore Technology (ONT) Ligation Sequencing Kit 1D (SQK-LSK108) and the Native Barcoding Kit 1D (EXP-NBD103) according to the ONT standard protocols. One microgram of DNA was used as input for each library. Multiplexed sequencing was performed by pooling 6-8 barcoded DNA samples. Prepared libraries were loaded onto an R9.4 flow cell and sequenced on a Minion device with ONT MinKNOW software.

England

Nanopore sequencing was performed at the John Radcliffe Hospital, Oxford, by Sophie George.

South Africa

Remnant stored DNA used for Illumina WGS from each isolate was retrieved from storage and used for Nanopore library preparation. Per isolate, one microgram of undigested DNA was prepared for Nanopore sequencing using the ligation sequencing kit (SQK-LSK109). In addition, the native barcoding expansion kit (EXP-NBD104) was used for multiplexing. The protocols for sequencing genomic DNA by ligation and native barcoding were carried out according to the manufacturers' instructions. Multiplexed sequencing libraries consisted of 6-12 barcoded DNA samples, and all libraries were sequenced using SpotON R9.4.1 flow cells on a MinION device.

3.2.3 PacBio sequencing

Thirty-five of the Malagasy samples were sequenced and processed at the Next Generation Genomics Core within Cold Spring Harbor Laboratory with PacBio Sequel 1M V2 SMRT cells. The circular consensus was generated via the SMRTlink graphical user interface version 6.0.0.47841. Section B.1.2 outlines the full details of the sequencing protocol.

3.2.4 Data preprocessing

All Nanopore data for this project was basecalled and de-multiplexed using the Nanopore proprietary software tool guppy (version 3.4.5). We used default parameters for basecalling, and the only non-default parameter used for de-multiplexing was to trim barcodes from the resulting sequences.

3.3 Genome assemblies for validating variant calls

A central component of the work in this chapter is validating the quality of variant calls - without being biased by assuming short reads are the "truth". In Section 3.6 we compare the precision and recall of Nanopore and Illumina variant calls. A necessary component of such analysis is a reference genome for each sample. Thirty-five of the Malagasy samples in the dataset were sent for PacBio Circular Consensus Sequencing (CCS) in addition to the matched sequencing on both the Nanopore and Illumina platforms. PacBio CCS produces so-called high-fidelity sequencing reads with a base-level accuracy greater than 99.8% [157]. These reads have such a high accuracy because each one is the consensus from multiple passes of the DNA enzyme around a circular copy of the original double-stranded read. As the CCS reads are both long and accurate, they are regularly used to produce high-quality *de novo* assemblies and complete existing reference genomes [158, 159]. For the samples with available PacBio data, we construct high-quality assemblies to use as reference genomes.

We chose samples with greater than 30x coverage across all three sequencing technologies to produce high-quality assemblies. In total, this left us with 9 Malagasy samples. There have been many new genome assembly methods produced since the last known assessment of *M. tuberculosis* long-read assemblies [87]. As such, we benchmarked five assemblers and chose the best for each sample. The reason for this comparison is that different assembly algorithms can produce quite varied results depending on sequencing technology used, species, or computational resource availability [89, 160]. The assembly tools used were Canu (v2.0;[161, 162]), Flye (v2.8;[163]), Unicycler (v0.4.8;[164]), HASLR (v0.8a1;[165]), and Spades(v3.14.0;[166]). Section B.2 presents the complete benchmark.

In summary, we use the unpolished PacBio-only assemblies produced by `flye` as reference genomes for eight samples. Although we assembled nine samples, we exclude one from further analysis due to significant contamination.

3.4 Quality control

Before any variant calling, all samples were subjected to a quality control (QC) pipeline to ensure all data used was of the highest quality.

The first step in this QC was decontamination of both Illumina and Nanopore sequencing reads. We use the decontamination database from `clockwork` (https://github.com/iqbal-lab-org/clockwork), which contains a wide range of organisms, including viral, human, *M. tuberculosis*, Nontuberculous Mycobacteria (NTM), and nasopharyngeal-associated bacterial genomes. Next, reads are mapped to the database using `bwa mem` (Illumina;v0.7.17) [25] and `minimap2` (Nanopore;v2.17) [41]. A read is retained if it has any mapping to a non-contamination genome (*M. tuberculosis*) in the database and is output to a final decontaminated fastq file. All other reads are considered contamination.

All decontaminated fastq files were randomly subsampled to a depth of 60x (Illumina) and 150x (Nanopore) using `rasusa` [147]. The reason for subsampling is to limit large read sets that can drastically slow down later steps in the analysis process and do not provide any benefit [160]. Any sample with a depth less than the maximum threshold remains unchanged by this subsampling.

The last step in the QC pipeline is to assign lineages for each sample. A panel of lineage-defining SNPs [167–169] is used in conjunction with a sample's Illumina VCF from Section 3.6.2 for the lineage assignment. At each lineage-defining position in the sample's VCF, we determine if the called allele is the same as the panel allele. If it is, we add the full lineage that allele defines (e.g. 4.1.1) to a list of called lineages. We abandon lineage assignment for a sample if more than one heterozygous call is made at lineage-defining positions. After classifying all of a sample's lineage-defining positions, we then produce a lineage assignment based on the list of called lineages. We use the most recent common ancestor of all the called lineages as the lineage assignment. For example, if the called lineages are $[4, 4.2.3, 4.2.5]$, the lineage assignment would be 4.2. Finally, a mixed-lineage assignment is made if more than one called lineage is from a different major lineage group. For example, $[4, 4.2.3, 4.2.5, 3.2]$ would still be called lineage 4.2; however, $[4, 4.2.3, 4.2.5, 3.2, 3.1]$ would be deemed mixed.

The purpose of QC is to ensure that all samples used in later analyses are of the highest quality. By highest quality, we mean all samples have perfectly matched Illumina and Nanopore data, sufficient read depth of coverage on both sequencing technologies (Illumina ≥ 20 and Nanopore ≥ 30), no contamination, and no evidence

of a mixed *M. tuberculosis* population. Fifty-eight samples failed to pass our QC measures, and 39 had non-matched Illumina and Nanopore data. One of the samples that failed QC is part of the eight PacBio samples we generated assemblies for in Section 3.3 - we exclude this sample from any further analysis. In total, we use the 150 samples that have passed QC in the remainder of this chapter and Chapter 4.

3.5 Construction of *M. tuberculosis* reference graphs

A parallel line of investigation in this chapter is to assess the benefit of `pandora` for *M. tuberculosis* transmission cluster inference. In particular, we focus on its use of prior knowledge about variation in a population and ability to genotype collections of samples against a reference chosen to be maximally close to all samples (Section 1.3.3). `pandora` requires a pan-genome reference graph (PanRG) in order to operate; for the work in this chapter, we chose to construct a reference PanRG based on the *M. tuberculosis* reference genome, H37Rv (accession NC_000962.3). We add variants sampled from 15,211 global *M. tuberculosis* isolates gathered by the CRyPTIC consortium [138]. We sampled at two different rates to evaluate how varying complexity of PanRGs affect variant-calling precision and recall.

To ensure the reference PanRG is not biased towards a particular lineage, we first split the global CRyPTIC VCF into separate lineage VCFs. We assign lineages for each of the global samples using the same approach as in Section 3.4. In addition, we include variant calls from 14 high-quality *M. tuberculosis* assemblies, representing lineages 1-7 [61, 170]. The two PanRG complexities we construct were termed "sparse" and "dense". From each lineage VCF, we took a random subsample of 50 and 200 samples and combined them into single sparse and dense VCFs, respectively. Note, we use the same fixed random seed for the subsampling to ensure the sparse PanRG is a subset of the dense PanRG. Finally, we filtered the resulting VCFs to remove any positions with no alternate allele calls or that failed the filtering applied by the CRyPTIC pipeline `clockwork`. One exception is that we did not remove positions that overlap a genome mask of repetitive regions in H37Rv [120].

The reference PanRG that `pandora` uses is actually a collection of local PRGs (loci). These loci are effectively partitions of the original genome; one can partition based on any criteria. We chose to split the H37Rv genome based on the genomic features outlined in the accompanying General Feature Format (GFF) file from the NCBI database. We also retain the segments *between* the features - so-called intergenic regions (IGRs). We combine genomic features with overlapping coordinates (e.g.,

transcribed on opposite strands or different reading frames) into a single locus and join any locus (feature or IGR) shorter than 500bp with its 3' neighbour. By building the reference PanRG in this manner, we ensure representation of every position in the H37Rv genome amongst all loci. We then remove any locus with 30% or more of its positions overlapping the H37Rv genome mask mentioned above [120]. Refer to Section B.3 for a detailed description of how we chose this masking strategy.

We form the sparse and dense PanRGs by applying the variants from the respective VCF to the template (reference) sequence of each locus; for each position in the VCF, we infer the locus it corresponds to. We then take all (called) alternate alleles and create a sequence for each; that is, the template sequence, with the reference allele replaced by the alternate allele. Note, we disregard any indels longer than 20bp or that span a locus boundary. Finally, all of these sequences are pooled into a single fasta file for each locus.

The multi-sequence fasta file for each locus is then subjected to multiple sequence alignment (MSA) using MAFFT (v7.471; [171]). We use the accurate global alignment setting, G-INS-i [172], with default parameters, using the `ginsi` script provided with MAFFT. The resulting MSA is then converted to a `pandora`-compatible PRG using the `make_prg` program (v0.1.1; Section 1.3.1; [10]) with a maximum nesting level of 5 and minimum match length of 7. All of the local PRGs are then combined into a single PanRG file and indexed with `pandora` using a k-mer size of 15 and window size of 14. In the end, we have two PanRG files - sparse and dense.

3.5.1 Computational performance

An important consideration for the usability of any genomic method is the computational cost in terms of time and memory resources. The construction process just outlined need only be run once, and then it can be used as a reference for subsequent `pandora` usage. However, it is necessary to understand the time and resources required in order to identify bottlenecks. Additionally, if the resource usage is high enough, it may also limit who can build a reference graph. We outline the time and memory requirements for each step of the graph construction in Table 3.1. All times are on a single compute node with 32 CPU cores. We only report the MSA, `make_prg`, and `pandora` index steps as these are necessary steps; those preceding use little time and memory and can be done in several different ways.

The most notable computational usage value from Table 3.1 is the fact that the sparse and dense MSAs require 209 and 301GB of memory, respectively. The exact

	Sparse			Dense		
Step	CPU time (sec)	Real time (H:m)	Max. RAM (GB)	CPU time (sec)	Real time (H:m)	Max. RAM (GB)
MSA	138576	1:16	**209**	445284	3:56	**301**
Make PRG	3746	0:04	0.9	4269	0:05	0.9
Index	142	0:01	1.5	361	0:01	1.7

Table 3.1: Computational time and memory (RAM) usage for the main steps of building a *M. tuberculosis* reference graph. Sparse and Dense refer to two different densities with respect to the number of variants used. All steps were run on a single compute node with 32 CPU cores. MSA=multiple sequence alignment;PRG=population reference graph.

cause of this high memory usage is not precisely known because we run the MSA of all loci in parallel and do not receive memory usage for each CPU core. However, MSA memory usage is known to scale exponentially with the number of sequences [173], and the sparse and dense loci with the highest number of sequences had 180 and 335 sequences, respectively. Therefore, we could reduce memory usage by placing a cap on the number of sequences in a locus. In addition, there are memory-saving options within MAFFT that could also be used, along with reducing the number of CPU cores, at the cost of increasing the runtime.

3.6 Variant calling and filtering assessment

One approach to determining genetic distance is to count the number of SNPs that differentiate two samples. These distances enable the identification of possible transmission clusters based on some predefined number of expected SNPs. Filtering of variant calls is integral to creating trusted variant calls on which to base such distances. However, there are many such filters used for Illumina genomic data, and they can produce inconsistent results [124]. As our focus is on whether Nanopore can be used for public health applications, we use the PHE Illumina pipeline - COMPASS [174] - as a comparison.

Before attempting to define SNP thresholds for Nanopore data, we explore the impact of a range of filtering parameters for both `bcftools` and `pandora`. The aim of this filter calibration is ultimately to determine if SNP-calling precision for Nanopore is comparable with Illumina, and if not, how close can we get it.

We evaluate the resulting, filtered SNP calls against the COMPASS Illumina SNP calls for the seven samples with high-quality PacBio assemblies (see Section 3.3) to ensure no bias for Illumina or Nanopore. For others interested in investigating variant filters for Nanopore data, we also hope this calibration acts as a good starting point for deeper analysis.

3.6.1 Validating variant calls

We evaluate the precision and recall of SNPs using the method outlined in Section 2.4.3 - `varifier` - with a flank length of 100bp [175]. The samples we evaluated are those seven with PacBio assemblies that passed QC. As a truth genome for each, we use the unpolished `flye` PacBio assembly, along with a mask for low-quality regions. These low-quality regions were identified by aligning the sample's Illumina reads to the assembly with `bwa mem` and flagging any position with less than 10 reads mapping to it or less than 90% agreement (see Section B.2.2).

3.6.2 Illumina variant calling

Fan Yang-Turner performed the Illumina variant calls (see Section 3.0) with the COMPASS pipeline used by PHE. Briefly, reads are mapped to H37Rv, and `samtools mpileup` is used to identify SNPs [30]. SNPs are filtered based on the following criteria: i) must have at least five high-quality supporting reads, ii) must have at least one read in each direction, iii) 75% of reads must be high-quality, iv) the diploid genotype must be homozygous, v) fraction of reads supporting the major allele must be at least 90%. In addition, any SNPs falling within masked sites - as defined by aligning the H37Rv to itself and identifying repetitive regions [120] - are excluded. This mask is the same as in Section 3.5.

3.6.3 Nanopore variant calling: BCFtools

As there is no standard variant caller used for *M. tuberculosis* Nanopore data, we chose to use BCFtools (v1.11; [29]), as it has a long history of use in bioinformatics and is one of the main variant callers used for Illumina data. It is readily available to all users and is much more user-friendly than other available tools, some of which have only be trained on Human data [84], or require the raw Nanopore data [80]. Another main reason for its use is that it is the updated form of the `samtools` pipeline used by COMPASS and thus provides a somewhat "fair" comparison.

Nanopore reads were aligned to H37Rv using `minimap2`, with options to produce SAM output containing no secondary alignments. The resulting SAM file is provided as input to the `bcftools` subcommand `mpileup` with the option to skip indels. The resulting pileup is then used to call SNPs with `bcftools call` using the multiallelic caller with a haploid model and an option to skip indels.

There are many fields in the resulting VCF relating to the information about the reads that support each position. After a thorough examination of how filtering based on each field impacts precision and recall, we settled on five filters for `bcftools`. First, we filter out positions with a quality (QUAL field) score less than 60. The quality is a log-scaled probability for the assertion made by the alternate allele. Second, a read position bias (RPB) of at least 0.05 is required. RPB indicates a bias for support from the ends of reads, as they are usually low quality. Third, we filter out positions with a segregation-based metric (SGB) less than -0.5. SGB is a measure of how read depths across alleles match expected depths. Fourth, variant distance bias (VDB) less than 0.002 is filtered out. VDB measures whether a variant's position is randomly distributed within the reads that support it or biased (e.g. near the start). Fifth, the fraction of reads supporting the called allele (FRS) must be 90% or more.

Figure 3.1 shows how the addition of each of these filters impacts precision (proportion of calls that are correct) and recall (proportion of variants found) for `bcftools` compared with COMPASS (Illumina). The trade-off between precision and recall is dependent on the question one is trying to answer. For transmission clustering, we place greater importance on precision as we seek to ensure the SNPs used are of the highest quality. The consequence of this is we miss some variants - compared to COMPASS.

The filtering we use for the remainder of the transmission inference work is to apply all five filters mentioned above. These filters, represented by the yellow box in Figure 3.1, lead to median precision and recall of 99.94% and 84.26%, respectively for the seven validation samples with PacBio assemblies. This is compared to the COMPASS median precision and recall values of 100% and 92.58%, respectively.

In summary, we produce Nanopore variant calls with equivalent precision to Illumina but with lower recall.

3.6.4 Nanopore variant calling: Pandora

When assessing the best filters for increasing the precision of variant calls from `pandora`, we are also interested in determining whether PanRG density has a noticeable impact on performance. Therefore, we use the sparse and dense PanRGs from Section 3.5 and look at the precision and recall these produce for the same filters.

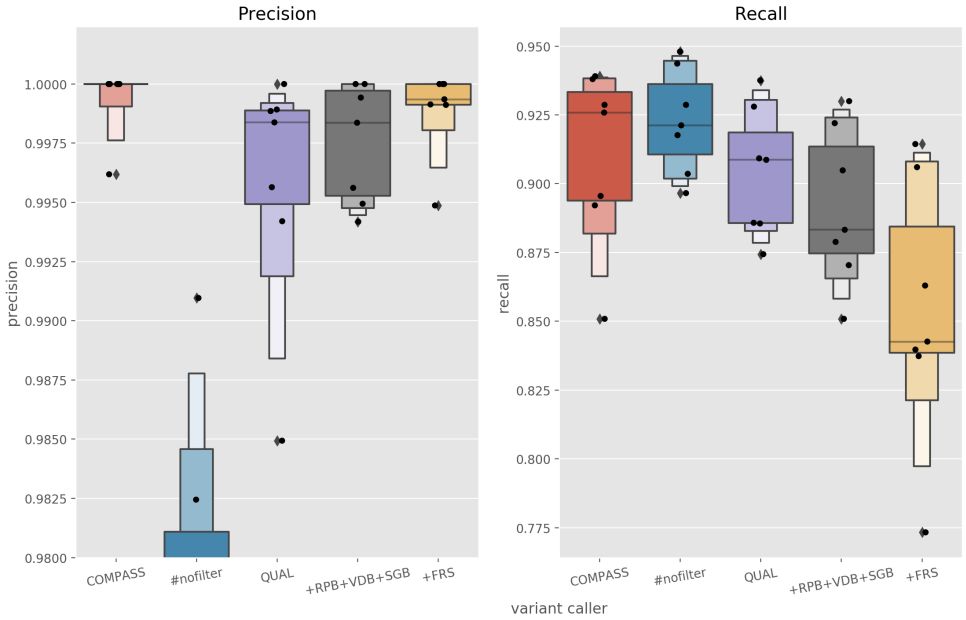


Fig. 3.1: Precision (left) and recall (right) of SNPs for COMPASS (red) and a selection of `bcftools` filters. `#nofilter` (blue) is `bcftools` with no filtering of variants. `QUAL` (purple) is `bcftools` SNPs with a quality score of 60 or more. `+RPB+VDB+SGB` (grey) indicates `bcftools` variants with the INFO field values ≥ 0.05, ≥ 0.002, and ≤ -0.5, respectively, plus QUAL. `+FRS` (yellow) shows `bcftools` SNPs with all previous filters, plus only SNPs where the fraction of reads supporting the variant is at least 90%. Note: the precision plot y-axis was cut causing some `#nofilter` points to be hidden.

Single-sample

For each sample, we discover *de novo* variants using the method outlined in Section 2.2.4 using the `discover` command of `pandora` (version 0.8.0). We use default parameters, except limiting the number of novel variants from a candidate region to 10. Novel variants are added to the original MSAs from Section 3.5 with the `-add` routine in MAFFT [143]. Next, `make_prg` is run on the subsequent alignments, and the resulting updated PanRG is indexed with `pandora`. Finally, the `map` routine of `pandora` genotypes a sample's reads and produces a VCF. To be able to compare the `pandora` VCF to the truth assemblies, we instruct `pandora` to output coordinates with respect to the H37Rv reference sequence for each locus. Running `pandora` in this way leads to some alleles being quite long and redundant, so we use `bcftools norm` to trim unused alleles and reduce variants down to their most succinct representation.

We apply four filters to the `pandora` variant calls. First, there must be at least 3 reads supporting the called allele. Second, we keep positions with a strand bias of at least 1%, which is the lowest depth on the forward or reverse strand divided by the total depth. This is a somewhat different definition of strand bias to that used in human genetics [176]; our definition is testing whether there are significantly more reads on one strand. For example, in this definition, if the forward and reverse strand have read depths of 1 and 9, respectively, the strand bias is 10%. Therefore, this example position would not be filtered out. Third, a genotype confidence score no less than 5. This score is the difference between the log-likelihoods of the called allele and the next most likely allele. Fourth, the fraction of reads supporting the called allele (FRS) must be at least 90% - calculated the same way as in Section 3.6.3.

The results of incrementally applying these filters, along with no filters and COMPASS, are shown in Figure 3.2. Of the two PanRGs used, `pandora`'s best median precision (100%) is with the sparse PanRG and all filters applied. With all filters, the sparse PanRG leads to a median recall of 71.99%. When compared to the COMPASS median precision and recall values of 100% and 92.58%, respectively, `pandora` produces Nanopore SNP calls with equivalent precision to Illumina, but with 20.59% less recall (SNPs *and* indels are assessed in Section B.4). Part of the recall disparity between `pandora` and COMPASS is explained by the masking of loci in the reference graph (see Section 3.5 and Section B.3). Despite this large difference in recall, we chose to use all of the filters outlined above because, as mentioned earlier, we value Nanopore precision more than recall for this transmission cluster work - our genomic epidemiologist collaborators in Oxford/PHE, with whom we are doing this, prefer

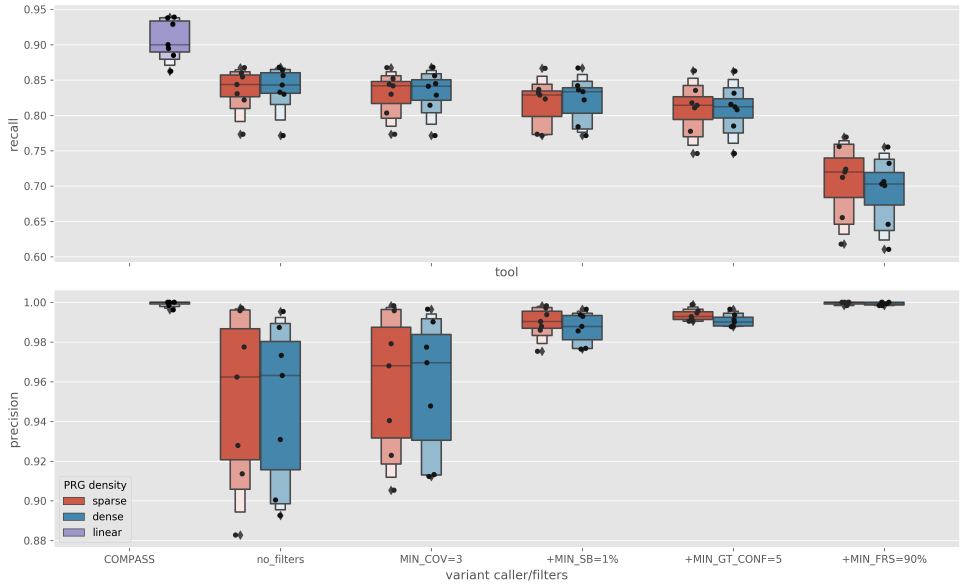


Fig. 3.2: Precision (bottom) and recall (top) of SNPs for COMPASS (purple) and `pandora` with sparse (red) and dense (blue) PanRGs. The `pandora` boxes start with no filters on the left, with each box moving to the right adding a filter to the previous box. The COMPASS box is a reference to the precision and recall of Illumina variant calls. Linear PRG density refers to the fact that COMPASS uses a single, linear reference genome as opposed to `pandora`, which uses a genome graph. The black points refer to single data points for the seven samples used. MIN_COV=minimum depth of coverage;MIN_SB=minimum strand bias;MIN_GT_CONF=minimum genotype confidence score;MIN_FRS=minimum fraction of read support.

precision. (We note that others argue for allowing more SNPs and are happy to deal with the higher SNP error rates [124].)

In nearly every filtering combination, the sparse PanRG lead to higher recall *and* precision, albeit marginally. As a result, the remaining work featuring `pandora` in this chapter will use the sparse PanRG given the increased computational cost of using the dense PanRG (Section 3.5.1 and Section 3.6.5), without any benefit for precision and recall.

Multi-sample

`pandora`'s `map` routine infers a consensus sequence for a single sample and outputs variant calls with respect to that consensus. However, `pandora` also has a multi-sample counterpart - the `compare` command. The `pandora compare` routine infers a single reference sequence that is maximally close to *all* samples. It outputs a locus presence/absence matrix, along with a VCF of genotypes for all samples with respect to the inferred reference sequence (see Section 1.3.3 for a description of the `pandora`

`compare` method). As `pandora compare` was designed for analysing collections of (potentially divergent) samples, we assess its ability to describe *M. tuberculosis* transmission clusters.

The process for calling variants using `pandora compare` is first to aggregate the novel variants discovered for each sample in Section 3.6.4. Then, instead of creating an updated PanRG for each sample, we use the aggregated novel variants to update the MSAs and PRGs as in Section 3.6.4. In the end, we have a PanRG that has novel variants from all samples contained within it, rather than the PRGs used by `pandora map`, which only have the novel variants for a single sample. Next, we run `pandora compare` using these updated sparse and dense PanRGs and filter the resulting VCF as per Section 3.6.4.

As a result of its design, it is not possible to provide `pandora compare` with a reference to base VCF coordinates on (as in Section 3.6.4). Consequently, we cannot assess the precision and recall for the seven samples as above. However, we have evaluated the accuracy of `pandora compare` in Section 2.4; therefore, in this chapter, we assess its usefulness for calculating genetic relatedness.

3.6.5 Computational performance

Single sample methods

In addition to the quality of the variant calls, the computational cost of producing them is also important. The CPU time and maximum memory usage for performing the Nanopore variant calling is shown in Figure 3.3. `pandora`'s performance is broken down into the individual stages, while `bcftools` is represented by a single job (`pileup_nanopore`). The median maximum memory for `bcftools` was 8.2GB, although the maximum was as high as 58.5GB. This is compared to the highest `pandora` step - updating the MSAs with novel variants - with a median maximum memory usage of 9.7GB and 13.3GB for the sparse and dense PanRGs respectively. The highest memory usage for `pandora` was 18.6GB during the updating of MSAs, nearly 40GB lower than the peak of `bcftools`. However, we note that the peak memory usage for `pandora`'s sparse and dense PanRG construction (Section 3.5.1) was 209 and 301GB, respectively, although this is a one-off cost and does not need to be run for each sample.

The median CPU time for `bcftools` was 35129 seconds, or 9.75 hours, with the longest run coming in at 138364 seconds (38.4 hours). To be able to compare

	Sparse			Dense		
Step	CPU time (sec)	Real time (H:m)	Max. RAM (GB)	CPU time (sec)	Real time (H:m)	Max. RAM (GB)
Update MSA	114677	1:01	38	130221	1:15	37
Make PRG	4700	0:05	1.2	5403	0:06	1.1
Index	538	0:01	2.1	1224	0:02	2.4
Compare	90486	4:25	5.4	162294	6:04	6.1

Table 3.2: CPU and wall clock time, and memory (RAM) usage for the main steps of running `pandora`'s multi-sample routine `compare`. Sparse and Dense refer to two different densities with respect to the number of variants used. All steps were run on a single compute node with 32 CPU cores. MSA=multiple sequence alignment; PRG=population reference graph.

`bcftools` with `pandora` as a whole, we can sum the median CPU time over each step, which gives 21704 and 53194 seconds, or 6.0 and 14.7 hours, for the sparse and dense PanRGs respectively. As with the memory usage, the longest runtime component of the `pandora` pipeline was updating the MSAs with *de novo* variants. Note, `bcftools`' `mpileup` command is not parallelised, while `pandora`'s steps are, so the wall clock time for the two is dependent on the number of CPU cores available.

Multi sample method

The time and memory of `pandora compare` is not directly comparable to `bcftools` and `pandora map` as it runs on all samples at the same time. Additionally, the novel variant discovery phase of `pandora map` for all samples technically contributes to the overall runtime of `pandora compare`. As the performance of this discovery step has already been reported, we outline the remainder of the `pandora compare` steps in Table 3.2. In total, the remainder of the sparse and dense PanRG steps took 58.4 and 83.1 CPU hours, respectively. The actual wall clock time for these steps was 5.5 and 7.3 hours using 32 CPUs. Maximum memory usage occurred while updating the MSAs and peaked at 38 and 37GB for the sparse and dense PanRGs, respectively.

3.6.6 Summary

In summary, Figure 3.4 shows that our selection of filters for Nanopore variant callers provides precision on-par with Illumina. However, this precision comes at the cost of a loss in recall. Additionally, both `bcftools` and `pandora` have considerable computational costs compared to what is typical for Illumina data.

The remainder of this chapter explores how the SNP calls from Nanopore can be used to calculate distances between samples and define putative transmission clusters from these distances. To recapitulate, we are using SNP calls from `bcftools` (per-sample), `pandora map` (per-sample; sparse PanRG), and `pandora compare` (multi-

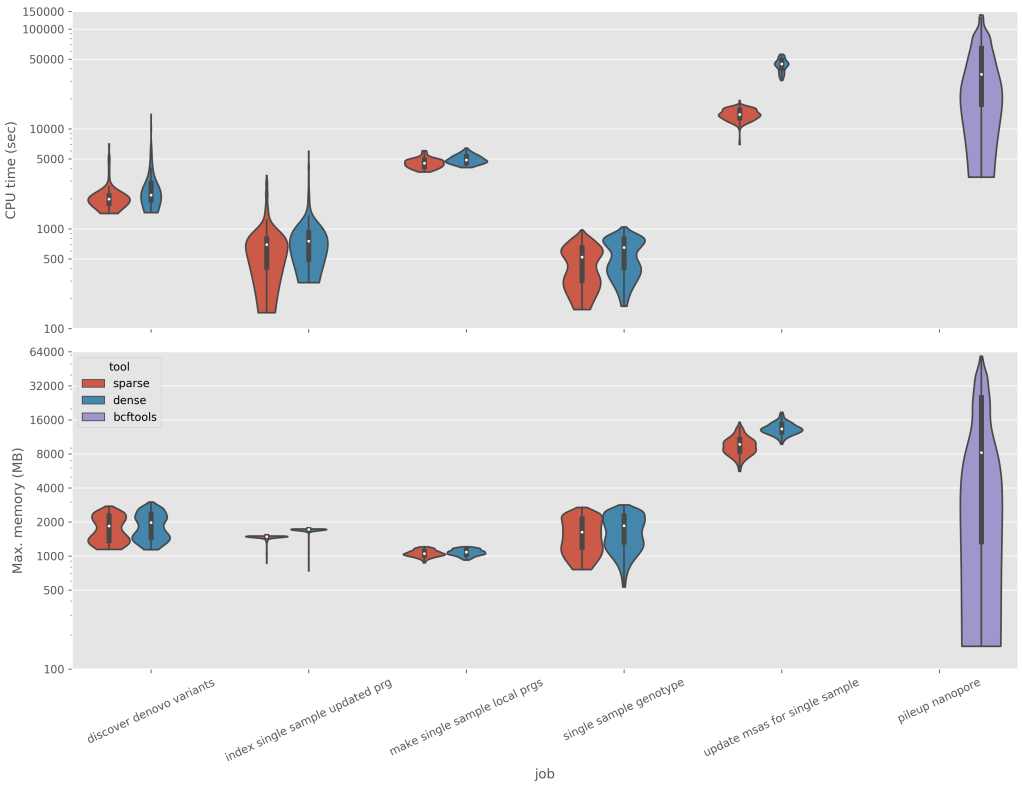


Fig. 3.3: The CPU time (in seconds; y-axis; top) and maximum memory usage (in megabytes; y-axis; bottom) for each Nanopore variant-calling job. Sparse (red) and dense (blue) refer to `pandora` steps with the respective density PanRG. `bcftools` (purple) only has one step (`pileup nanopore`). The violins represent the distribution of CPU time over all samples. Note, the y-axis for both plots is log-scaled.

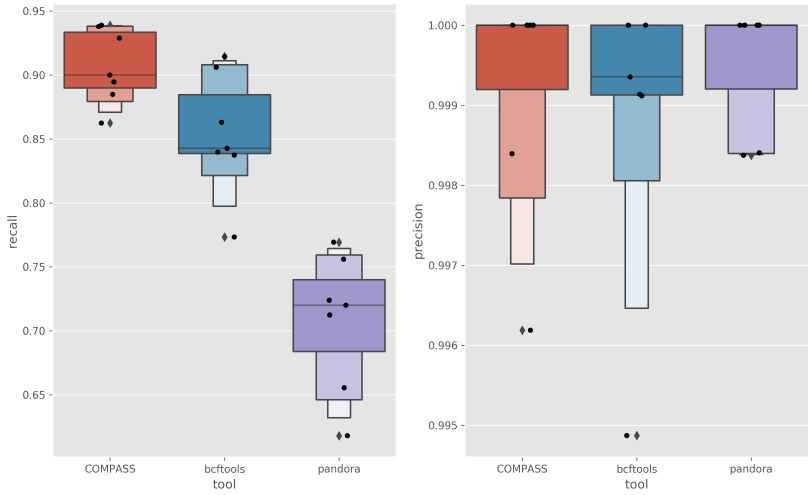


Fig. 3.4: Precision (left) and recall (right) of filtered SNPs for COMPASS (red), `bcftools` (blue), and `pandora` (purple). Each black point represents one of seven evaluation samples.

sample; sparse PanRG). We are especially interested in how similar the pairwise distances are between samples and sequencing modality and whether the same distance thresholds used for Illumina can also be used for Nanopore.

3.7 Pairwise SNP distance comparison

When attempting to infer transmission clusters, one approach defines a SNP distance threshold and says that any genomes within this distance of each other are clustered (possible transmissions) [117]. It follows that the SNPs used must be trusted. Having shown we can achieve SNP precision on-par with Illumina using Nanopore data (see Section 3.6), we investigate the pairwise SNP distance between samples produced by both Illumina and Nanopore sequencing technologies. The intention here is to determine whether the thresholds typically used for Illumina data can also be used for Nanopore, or whether adjustments are required.

To determine the distance between samples, we first generate sample consensus sequences. We do this for each variant-caller: COMPASS (Illumina), `bcftools` (Nanopore), and `pandora map` (Nanopore) (not `pandora compare` - see below). A consensus sequence is obtained by applying the calls from a given VCF (from Section 3.6) to the *M. tuberculosis* reference genome. We nullify (mark as `N`) any positions where: i) the position failed filtering, ii) the reference genome position does not appear

in the VCF file (except for `pandora map`), iii) the called genotype is null, or iv) the position is within the reference genome mask.

Next, all sample consensus sequences for a variant-caller are joined into a single FASTA file and a pairwise distance matrix is calculated using `snp-dists` (version 0.7.0) [177]. In the case of `pandora compare` (multi-sample mode), we cannot follow this approach for generating a consensus sequence and distance matrix due to the inability to translate the coordinates from a graph to a linear reference. However, as `pandora compare` selects a cohort-specific reference, it effectively allows one to go directly to a distance matrix. Therefore, we generate a genotype array instead of a consensus sequence by extracting the called genotype for each sample at each site (VCF entry). Where a site has failed a filter, we use a genotype value of -2. To calculate the distance between two samples, we compare their genotype arrays; if either sample's genotype is < 0 (i.e., null or filtered) or the genotypes are the same, we record a distance of 0, otherwise 1. The sum of these comparisons for each genotype is the distance between the two samples.

The pairwise SNP distance relationship is presented in Figure 3.5. For a given pair of samples, we plot their SNP distance, based on the COMPASS (Illumina) variant calls (x-axis), against the SNP distance for the same pair, based on the Nanopore variant calls (y-axis). All pairwise comparisons between a sample and itself are absent from the visualisation, and only a single value was used for each pair (i.e., we keep sample1 vs. sample2 and discard sample2 vs. sample1 as they are the same). RANSAC Robust Linear Regression [178], as implemented in the Python library `scikit-learn` [179], was used for determining a linear equation and line-of-best fit for the relationship between pairwise Illumina and Nanopore SNP distance.

If the same thresholds used for Illumina can also be used for Nanopore, we would expect the distances to be the same and the bulk of the points in the plot to fall on the dashed, diagonal identity line in Figure 3.5. What we see instead is a linear relationship that falls *under* this identity line - for all Nanopore variant callers. Given the filtered Nanopore SNP calls made by `bcftools` and `pandora` have lower recall than Illumina (Section 3.6.6), this is expected, as they miss some SNPs found by Illumina.

We highlight one important observation in the zoomed inset of Figure 3.5. As SNP thresholds used for *M. tuberculosis* are generally well below 100 [123], it makes more sense to base SNP distance relationships on those samples that are "close". And indeed, when we zoom in on pairs of samples within 100 (Illumina) SNPs of each other, we see an association that is closer to the identity line. Fitting a linear model to this close subset of pairwise distances yields a relationship defined by the

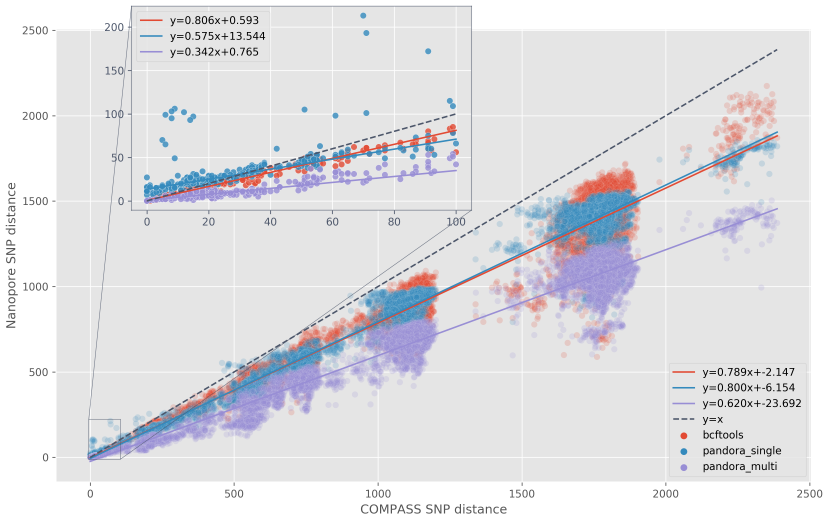


Fig. 3.5: Pairwise SNP distance relationship between Illumina (COMPASS; x-axis) and Nanopore (`bcftools` (red), and `pandora` single-sample (blue) and multi-sample (purple) mode; y-axis) data. Each point represents the SNP distance between two samples for the two sequencing modalities. The black, dashed line shows the identity line (i.e. $y = x$) and the coloured lines shows the line of best fit based on the robust linear model fit to the data. The zoomed inset shows all pairs where the COMPASS distance is ≤ 100.

equation $y = 0.806x + 0.593$ for `bcftools`, $y = 0.575x + 13.544$ for `pandora map`, and $y = 0.342x + 0.765$ for `pandora compare`. Replacing x with an Illumina SNP threshold gives the (predicted) equivalent Nanopore SNP threshold based on these relationships. For example, at an Illumina SNP distance of 12, the linear equation would predict a corresponding `bcftools` Nanopore SNP distance of 10.

In the middle-left of the inset in Figure 3.5 a small cluster of `pandora map` (blue) points can be seen. These have an approximate pairwise Nanopore distance of 100, but $\sim$10 for Illumina. Upon further investigation, the cause of the large discrepancy in the distance was due to `pandora map` failing to identify (and filter) some heterozygous calls. Two samples, in particular, occur as one member in all of the major outlying pairs. 94% of the false-positive differences leading to the large Nanopore distances occur at positions that are filtered due to evidence of heterozygosity in COMPASS. That is, in the Illumina consensus sequence, these positions are ignored due to filtering and do not count as a difference. However, `pandora` did not have sufficient read depth on both alleles to trigger the FRS filter (Section 3.6.4) - leading to a passing variant call that differs from the sample it is being compared with.

The relationship between Illumina and Nanopore distances is indeed linear for all three variant-calling methodologies. While the relationship is not identical, we will attempt to use a linear model fit to the relationship to infer what Nanopore SNP distance threshold is likely to align with a given Illumina threshold for defining putative transmission clusters.

3.8 Nanopore transmission clustering

While the relationship between Illumina and Nanopore pairwise SNP distance is enlightening, ultimately, the fundamental question is: do Nanopore SNPs lead to transmission clusters consistent with those obtained with Illumina SNPs? To answer this question, we compare Illumina- and Nanopore-based clusters for four Illumina SNP thresholds.

Selecting a SNP threshold to infer transmission clusters from has seen a variety of values recommended [123]. As we seek to show concordance of Nanopore data with PHE's Illumina-based strategy, we opt to investigate Illumina threshold values 0, 2, 5, and 12. PHE define two cases as clustered if they have a SNP distance ≤ 12 as "*12 SNPs represents the maximum SNP difference between 2 isolates for which epidemiological links have previously been identified [117] and is a conservative measure for reporting isolate relatedness*" [153]. Five was likewise selected as Walker *et al.* [117] found it to indicate membership in a recent transmission chain. Finally, threshold values 0 and 2 were chosen to provide insight into the level of granularity possible and are of clinical interest in some settings (personal correspondence with Tim Peto). For each of these four thresholds, we investigate what corresponding Nanopore SNP distance threshold yields the most similar clustering.

3.8.1 Transmission cluster similarity

We use the distance matrices from Section 3.7 to infer transmission clusters. To cluster samples, for a given SNP threshold t, we use pairs of samples with a distance $\leq t$ to define a graph, $G = (V, E)$, where samples (nodes, V) are connected by weighted edges (E), with the weight of an edge indicating the distance between the two samples it connects. We define clusters as the set of connected components $\{C_1, C_2 ... C_N\} \in G$, where N is the number of clusters. That is, a cluster (connected component), C_i, is a subgraph of G where a path exists between any two samples in C_i, but no path exists

to any samples in the rest of G. With this definition, all clusters have a minimum of two members.

To assess how closely Nanopore SNP-based clustering approximates Illumina SNP-based clustering, we adapt a similarity measure on sets; the Tversky Index [180]. We define the Illumina clustering as G and the Nanopore clustering as H. We are interested in being able to quantify the recall and precision of the Nanopore clustering with respect to Illumina. In this sense, recall describes the proportion of clustered samples in G clustered with the expected (correct) samples in H. Likewise, precision in this context tells us when extra samples are added to existing clusters by H or when clusters in G are joined in H.

In order to be able to define precision and recall when comparing two clustering graphs G and H, we define the Tversky Index

$$TI(n,G,H) = \frac{\left|C_{n,G} \cap C_{n,H}\right|}{\left|C_{n,G} \cap C_{n,H}\right| + \alpha|C_{n,G} - C_{n,H}| + \beta|C_{n,H} - C_{n,G}|} \tag{3.1}$$

where $C_{n,G}$ is the cluster in G that sample n is a member of. When $\alpha = 1$ and $\beta = 0$ in Equation 3.1, we get a metric analogous to recall - as described above. Therefore, we define recall, R, for a single sample n as

$$R(n,G,H) = \frac{\left|C_{n,G} \cap C_{n,H}\right|}{\left|C_{n,G} \cap C_{n,H}\right| + |C_{n,G} - C_{n,H}|} = \frac{\left|C_{n,G} \cap C_{n,H}\right|}{|C_{n,G}|} \tag{3.2}$$

When $\alpha = 0$ and $\beta = 1$ in Equation 3.1, we get a metric analogous to precision. As such, we define precision P, for a single sample n as

$$P(n,G,H) = \frac{\left|C_{n,G} \cap C_{n,H}\right|}{\left|C_{n,G} \cap C_{n,H}\right| + |C_{n,H} - C_{n,G}|} = \frac{\left|C_{n,G} \cap C_{n,H}\right|}{|C_{n,H}|} \tag{3.3}$$

With these definitions for a single sample, we can assess the recall and precision of the Nanopore clustering, H, with respect to the Illumina clustering, G, by averaging each metric over all samples in G. This gives us the Sample-Averaged Cluster Recall (SACR)

$$SACR = \frac{\sum_n^{V_G} R(n,G,H)}{|V_G|} \tag{3.4}$$

where V_G is the set of samples (nodes) in G (Illumina graph). Likewise, we define the Sample-Averaged Cluster Precision (SACP) as

$$SACP = \frac{\sum_{n}^{V_G} P(n, G, H)}{|V_G|} \tag{3.5}$$

SACR states, on average, what proportion of the samples clustered together in G are also clustered together in H (Nanopore) - it is a measure of how many true positives Nanopore retains. Inversely, SACP states, on average, what proportion of the samples clustered together in H are also clustered together in G - it is a measure of how many extra samples Nanopore adds to clusters.

However, SACR and SACP do not inherently account for when H has clusters containing only samples deemed non-clustered (singleton) in G. In order to quantify any extra clustering by H, we establish the Excess Clustering Rate (XCR) as the proportion of singletons (disconnected nodes) in G that are connected in H. We define XCR as

$$XCR = \frac{|S_G - S_H|}{|S_G|} \tag{3.6}$$

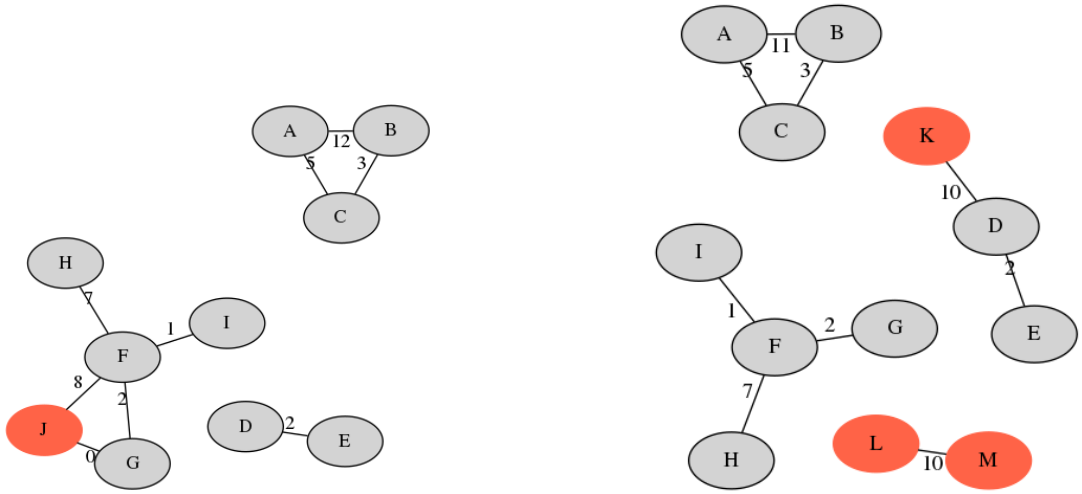
where S_G and S_H are the sets of singletons in the respective graphs.

We assess the cluster similarities using the Python programming language with the `networkx` library [181]. For a given threshold, we create the Illumina clustering (graph), G, and the Nanopore clustering, H - as described above - and use these to calculate the SACR, SACP, and XCR using Equation 3.4, Equation 3.5, and Equation 3.6 respectively.

An illustrated example of cluster similarity metrics

Section 3.8.1 outlines three metrics - SACR, SACP and XCR - for evaluating the similarity between two different strategies for transmission clustering. In order to provide the reader with greater intuition for the purpose of each metric, we present an illustrated example in Figure 3.6.

We take Figure 3.6a to be the truth clusters and Figure 3.6b to be test clusters. These are akin to Illumina and Nanopore clusters, respectively, in Section 3.8.1. The individual recall and precision values (defined in Equation 3.2 and Equation 3.3) for each sample in Figure 3.6a are shown in Table 3.3. SACR and SACP (defined in Equation 3.4 and Equation 3.5) are *sample-averaged*, so their values for this example are 0.82 and 0.83 respectively.



(a) Truth clusters **(b)** Test clusters

Fig. 3.6: Illustrative examples of transmission clustering. **a)** represents truth clusters, while **b)** is clustering from some "test" method we would like to compare to **a**. The nodes represent samples with the numbers on the edges connecting them indicating the distance between those two samples. The red nodes indicate samples with a clustering disparity between the two clusterings. Note, we do not show singletons (disconnected nodes) - e.g., *J* is missing from **(b)**.

To highlight the objective of SACR, we use the truth and test clusters containing the sample F. Samples F, G, H and I are shared between both, but J is missing from the test cluster. To calculate the individual recall for F, we take the intersection size of the truth and test clusters it exists in and divide it by the size of the truth cluster - $\frac{4}{5} = 0.8$. We do the same for the precision of sample D, except we divide by the size of the test cluster - giving $\frac{2}{3} = 0.66$.

The relevance of the XCR metric is best exemplified by the test cluster containing samples L and M. As we calculate SACR and SACP for all samples in the *truth* clusters, these two samples would be ignored. However, they are samples that - according to the truth - should not be part of any cluster (singletons). Therefore, SACR and SACP cannot capture these extra clusterings if they do not contain clustered truth samples. XCR covers this limitation and is the proportion of singletons in the truth that are clustered in the test (see Equation 3.6). As Figure 3.6 does not show singletons, let us pretend there are 20 singletons in the truth (including samples L and M). This would give an XCR of $2/20 = 0.1$.

Summary

To summarise, for each sample in an Illumina-defined cluster, SACR is the proportion of samples in its Illumina cluster also in its Nanopore cluster - averaged over all

sample	recall	precision
A	1.0	1.0
B	1.0	1.0
C	1.0	1.0
D	1.0	0.66
E	1.0	0.66
F	0.8	1.0
G	0.8	1.0
H	0.8	1.0
I	0.8	1.0
J	0.0	0.0

Table 3.3: Cluster recall and precision results for each sample in Figure 3.6.

samples. SACP is the proportion of samples in its Nanopore cluster also in its Illumina cluster - averaged over all samples. SACR indicates whether samples have been missed from Nanopore clustering (false negatives), and SACP reveals if additional samples are being added to Nanopore clusters (false positives). One shortcoming of SACR and SACP is that they do not account for when the Nanopore clustering contains clusters where no member of the cluster is part of an Illumina cluster. To that end, XCR is the proportion of Illumina non-clustered (singleton) samples added to a cluster by Nanopore. For example, an XCR value of 0.1 would indicate that 10% of non-clustered samples were part of a cluster in the Nanopore clustering. We provide an illustrated, worked example of these metrics in Section 3.8.1.

Of the metrics outlined above, our primary focus is SACR, as samples missed from clusters are of particular concern for public health agencies.

3.8.2 Evaluation of transmission clusters

The clustering produced for the four Illumina clusters of 0, 2, 5, and 12 are shown in Figure 3.7, Figure 3.8, Figure 3.9, and Figure 3.10, respectively. We discuss the results for each Nanopore variant caller below.

BCFtools

For the four Illumina SNP distance thresholds of interest - 0, 2, 5, and 12 - the corresponding `bcftools` thresholds we use are 0, 2, 5, and 11. We chose to forego the model-based predicted thresholds and instead use the hand-picked ones based on a threshold parameter-sweep outlined in Section B.5.

Threshold	SACR	SACP	XCR
0	1.0	1.0	0.015 (2/137)
2	1.0	0.966	0.008 (1/128)
5	1.0	0.949	0.057 (7/122)
12 (11)	1.0	0.845	0.031 (3/97)

Table 3.4: Summary of `bcftools` clustering metrics for four (Illumina) SNP distance thresholds. The threshold(s) in parentheses are the Nanopore equivalent threshold used. The fractions in parentheses for XCR indicate the underlying numbers. SACR=sample-averaged cluster recall; SACP=sample-averaged cluster precision; XCR=excess clustering rate.

The `bcftools` clustering results are summarised in Table 3.4 for all four SNP thresholds analysed. Of note, `bcftools` achieves a SACR of 1.0 at all thresholds - meaning Nanopore does not miss any samples from their correct clustering.

For the SNP threshold of 0 (Figure 3.7; top-right), `bcftools` perfectly recreated the Illumina clusters, with the addition of a cluster of two samples that were singletons (not clustered) in Illumina. At the SNP threshold of 2 (Figure 3.8; top-right), `bcftools` clustering only differed from Illumina by the addition of one singleton to a cluster of three (cluster 1). SNP threshold 5 (Figure 3.9; top-right) had the highest XCR (0.057) due to two new singleton clusters of size 2 and 3 and the addition of 2 singletons to a cluster of 5 (cluster 1). The lowest SACP was at threshold 12 (Figure 3.10; top-right) due to the joining of clusters 1 and 2, and clusters 7 and 8, and with three singletons being added to existing clusters.

Pandora single-sample

For `pandora` single-sample (`pandora map`), we also chose to use the hand-picked SNP distance thresholds from analysis in Section B.5. These are 16, 18, 18, and 27 for the Illumina thresholds of interest 0, 2, 5, and 12, respectively. The clustering results for each of these thresholds are summarised in Table 3.5.

At no threshold was `pandora map` clustering able to achieve perfect SACR, SACP or XCR. In particular, all thresholds had an SACP value less than 0.69 and an XCR greater than 0.11. These results outline the fact that many singletons were erroneously clustered, and many clusters merged. In large part, this is expected due to the much wider spread of distances along the y-axis, in the inset of Figure 3.5, when comparing `pandora map` to `bcftools` or `pandora compare`. Although the SACR values are not as low as the SACP, we place a higher value on them.

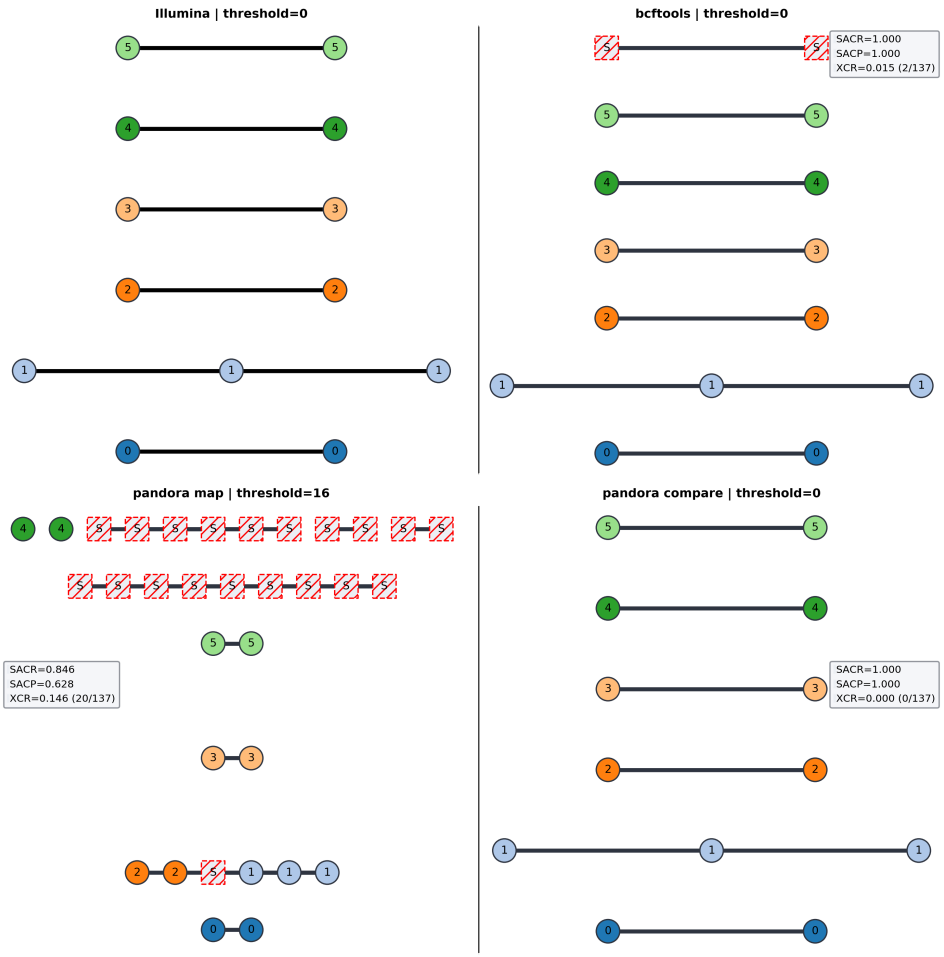


Fig. 3.7: Agreement of Illumina and Nanopore transmission clustering at an Illumina SNP threshold of **0**. The expected (Illumina) clusters are shown in the top-left panel. The other panels show the Nanopore-based clustering from `bcftools` (top-right), `pandora map` (bottom-left), and `pandora compare` (bottom-right), with the title indicating the SNP threshold used for clustering. Nodes are coloured and numbered according to their Illumina cluster membership. Samples not clustered (singletons) in Illumina are represented as white boxes with red stripes and are named "S". Clusters are horizontally aligned and connected with black lines. Where a sample that Illumina clustered is *not* clustered by Nanopore, the sample retains its original colour and number but is represented as an unconnected node on the top row of the plot. Each Nanopore panel has a legend showing the SACR, SACP, and XCR value with respect to the Illumina clustering. SACR=sample-averaged cluster recall; SACP=sample-averaged cluster precision; XCR=excess clustering rate.

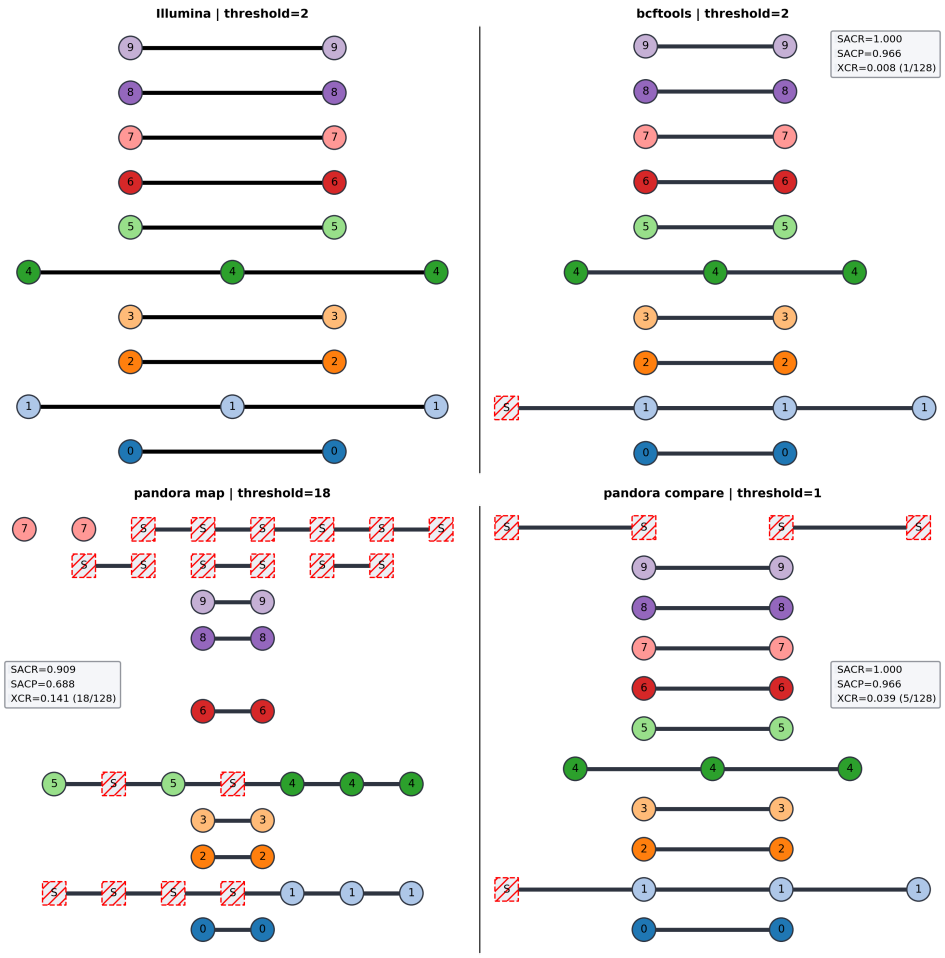


Fig. 3.8: Agreement of Illumina and Nanopore transmission clustering at an Illumina SNP threshold of **2**. The expected (Illumina) clusters are shown in the top-left panel. The other panels show the Nanopore-based clustering from `bcftools` (top-right), `pandora map` (bottom-left), and `pandora compare` (bottom-right), with the title indicating the SNP threshold used for clustering. Nodes are coloured and numbered according to their Illumina cluster membership. Samples not clustered (singletons) in Illumina are represented as white boxes with red stripes and are named "S". Clusters are horizontally aligned and connected with black lines. Where a sample that Illumina clustered is *not* clustered by Nanopore, the sample retains its original colour and number but is represented as an unconnected node on the top row of the plot. Each Nanopore panel has a legend showing the SACR, SACP, and XCR value with respect to the Illumina clustering. SACR=sample-averaged cluster recall; SACP=sample-averaged cluster precision; XCR=excess clustering rate.

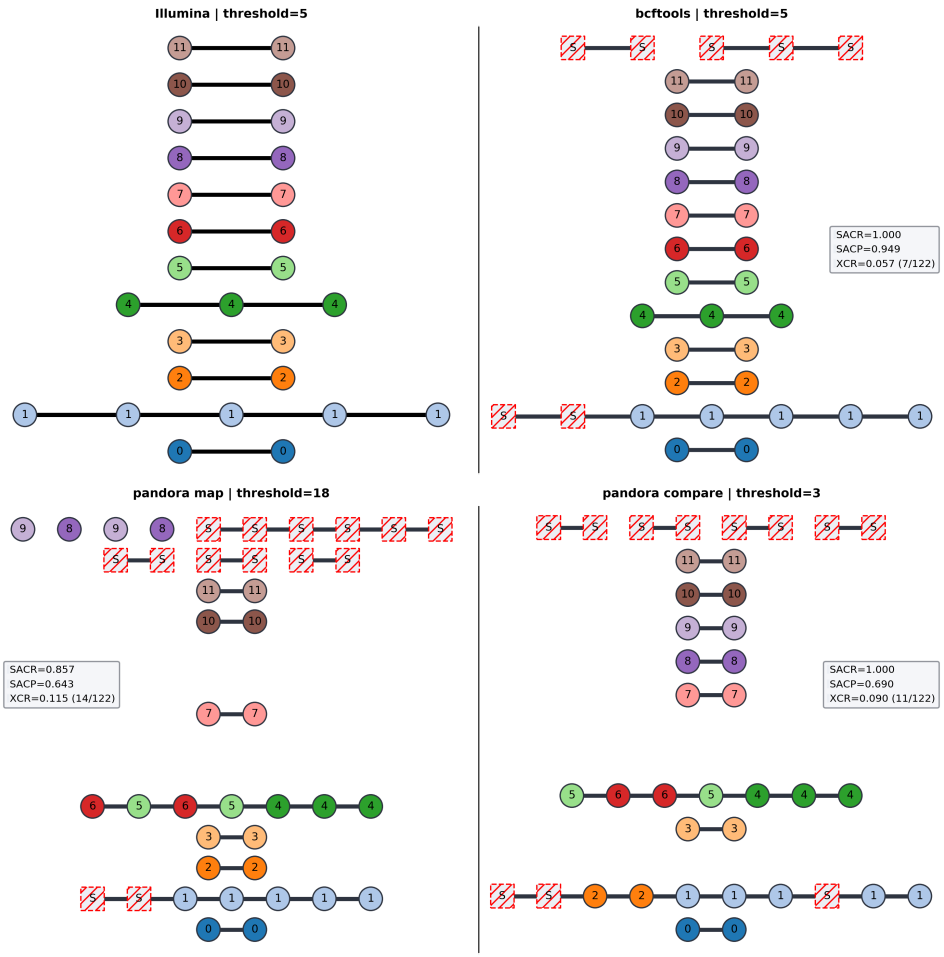


Fig. 3.9: Agreement of Illumina and Nanopore transmission clustering at an Illumina SNP threshold of **5**. The expected (Illumina) clusters are shown in the top-left panel. The other panels show the Nanopore-based clustering from `bcftools` (top-right), `pandora map` (bottom-left), and `pandora compare` (bottom-right), with the title indicating the SNP threshold used for clustering. Nodes are coloured and numbered according to their Illumina cluster membership. Samples not clustered (singletons) in Illumina are represented as white boxes with red stripes and are named "S". Clusters are horizontally aligned and connected with black lines. Where a sample that Illumina clustered is *not* clustered by Nanopore, the sample retains its original colour and number but is represented as an unconnected node on the top row of the plot. Each Nanopore panel has a legend showing the SACR, SACP, and XCR value with respect to the Illumina clustering. SACR=sample-averaged cluster recall; SACP=sample-averaged cluster precision; XCR=excess clustering rate.

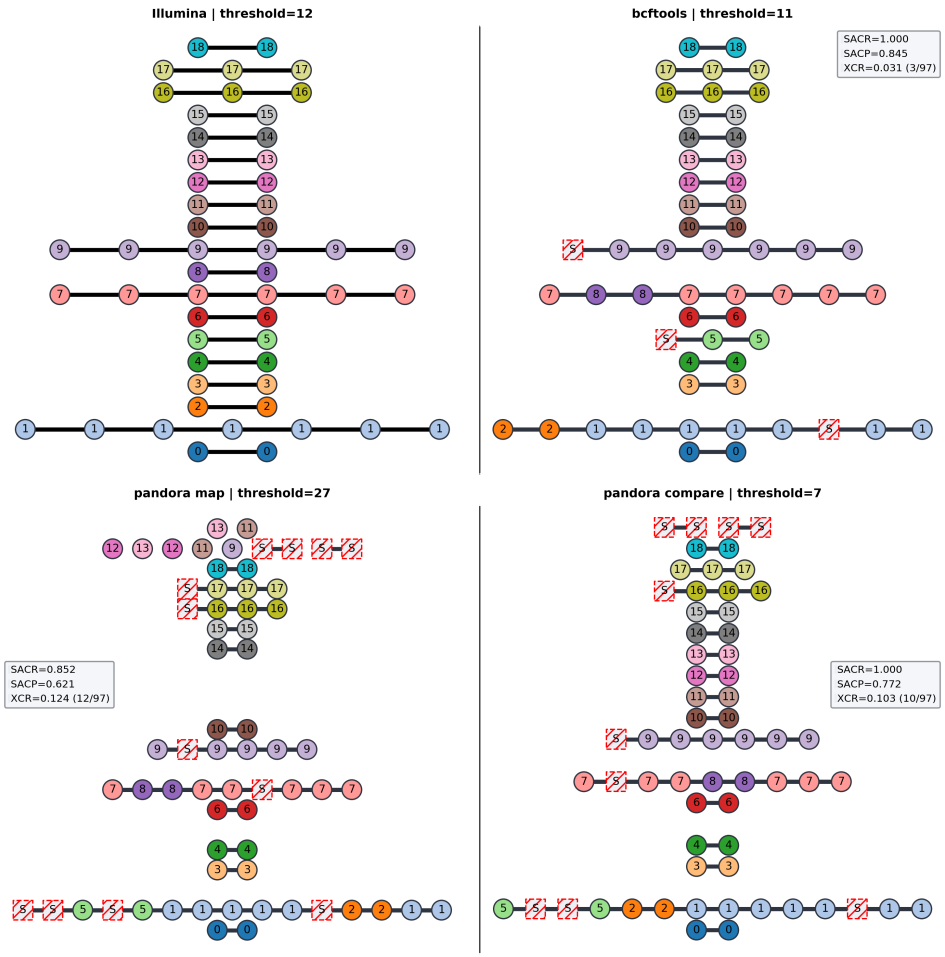


Fig. 3.10: Agreement of Illumina and Nanopore transmission clustering at an Illumina SNP threshold of **12**. The expected (Illumina) clusters are shown in the top-left panel. The other panels show the Nanopore-based clustering from `bcftools` (top-right), `pandora map` (bottom-left), and `pandora compare` (bottom-right), with the title indicating the SNP threshold used for clustering. Nodes are coloured and numbered according to their Illumina cluster membership. Samples not clustered (singletons) in Illumina are represented as white boxes with red stripes and are named "S". Clusters are horizontally aligned and connected with black lines. Where a sample that Illumina clustered is *not* clustered by Nanopore, the sample retains its original colour and number but is represented as an unconnected node on the top row of the plot. Each Nanopore panel has a legend showing the SACR, SACP, and XCR value with respect to the Illumina clustering. SACR=sample-averaged cluster recall; SACP=sample-averaged cluster precision; XCR=excess clustering rate.

Threshold	SACR	SACP	XCR
0 (16)	0.846	0.628	0.146 (20/137)
2 (18)	0.909	0.688	0.141 (18/128)
5 (18)	0.857	0.643	0.115 (11/122)
12 (27)	0.852	0.621	0.124 (12/97)

Table 3.5: Summary of `pandora` single-sample clustering metrics for four (Illumina) SNP distance thresholds. The threshold(s) in parentheses are the Nanopore equivalent threshold used. The fractions in parentheses for XCR indicate the underlying numbers. SACR=sample-averaged cluster recall; SACP=sample-averaged cluster precision; XCR=excess clustering rate.

For the (Illumina) SNP threshold of 0 (Figure 3.7; bottom-left), `pandora map` failed to recreate cluster 4. In addition, clusters 1 and 2 were merged with a singleton added, and four new clusters of singletons (one with 9 members) we created. At the SNP threshold of 2 (Figure 3.8; bottom-left), `pandora map` clustering failed to recreate cluster 7, joined clusters 4 and 5, and clustered 18 singletons. SNP threshold 5 (Figure 3.9; bottom-left) failed to recreate clusters 8 and 9; merged clusters 4, 5, and 6; and clustered an additional 14 singletons. Finally, at threshold 12 (Figure 3.10; bottom-left), `pandora map` failed to recreate clusters 11, 12, and 13; missed one sample from cluster 9; merged clusters 1, 2, and 5 and also joined clusters 7 and 8; and clustered 12 singletons.

Pandora multi-sample

The SNP thresholds we use for `pandora compare` (multi-sample) clustering are 0, 1, 3, and 7. The results of this clustering are summarised in Table 3.6. One important result is that unlike the single-sample approach of `pandora`, the multi-sample mode leads to perfect SACR across all thresholds. Additionally, clustering at the threshold of 0 (Figure 3.7; bottom-right) perfectly mirrors Illumina. At a threshold of 2 (Figure 3.8; bottom-right), there was one singleton added to an otherwise perfect cluster (cluster 1) and two additional singleton clusters of size 2 (doubletons).

For threshold 5 (Figure 3.9; bottom-right), `pandora compare` merged clusters 4, 5, and 6, as well as clusters 1 and 2. Three singletons were added to the merged cluster 1/2 and four new doubletons were created. Threshold 12 (Figure 3.10; bottom-right) likewise saw cluster mergers (1/2/5 and 7/8), two new doubletons, and six singletons added to existing clusters; however, SACR remained perfect.

Threshold	SACR	SACP	XCR
0	1.0	1.0	0.0 (0/137)
2 (1)	1.0	0.966	0.039 (5/128)
5 (3)	1.0	0.690	0.090 (11/122)
12 (7)	1.0	0.772	0.103 (10/97)

Table 3.6: Summary of `pandora` multi-sample clustering metrics for four (Illumina) SNP distance thresholds. The threshold(s) in parentheses are the Nanopore equivalent threshold used. The fractions in parentheses for XCR indicate the underlying numbers. SACR=sample-averaged cluster recall; SACP=sample-averaged cluster precision; XCR=excess clustering rate.

3.8.3 Summary

The results presented in this section show that when using `bcftools` for variant-calling, Nanopore is capable of producing transmission clusters with a high degree of similarity to Illumina. Most importantly, no samples deemed part of a cluster by Illumina were missed by `bcftools` (i.e., $SACR = 1.0$). However, as the SNP threshold increases, `bcftools` erroneously adds more samples to clusters - or joins existing clusters - with an SACP of 0.845 at a SNP threshold of 12. That is, on average, 84.5% of the members in a sample's Nanopore cluster are also in its Illumina cluster.

We have also shown that the Illumina SNP thresholds of 0, 2, and 5 are also valid for Nanopore variant calls from `bcftools` and the threshold of 12 needs only decrease to 11.

We additionally investigated whether the genome graph method of `pandora` can produce accurate transmission clusters. While the single-sample approach did not yield outstanding results, the multi-sample method shows promise. For all SNP thresholds assessed, `pandora compare` did not miss any samples from clustering. The SACP values for thresholds 0 and 2 were as good as `bcftools`, but at thresholds 5 and 12 `pandora compare` did not perform as well. For example, at threshold 12, `bcftools` erroneously added two doubleton clusters and three singletons to larger clusters, while `pandora compare` added three doubletons and six singletons.

In conclusion, we recommend clustering Nanopore data based on `bcftools` SNP calls for concordant clusters with Illumina.

3.9 Mixed Illumina and Nanopore transmission clusters

Having established that Nanopore data can recreate Illumina-defined transmission clusters with high recall and acceptable precision, we turn to the question of whether this holds when mixing Illumina and Nanopore data.

Inferring transmission clusters from a mixture of sequencing modalities would allow greater integration across datasets from various sources and prevent laboratories from being locked into one sequencing technology. As the uptake of Nanopore sequencing increases, it seems inevitable that there will be cases where comparisons between these sequencing modalities are necessary. To address this question, we simulate varying degrees of Nanopore/Illumina mixtures and look at how this impacts clustering. To this end, we investigate what the impact (if any) of combining Illumina and Nanopore datasets is on SACR, SACP and XCR (see Section 3.8.1 for definitions). For the Nanopore data, we use the `bcftools` distance matrices as they were shown to be the most concordant with Illumina (Section 3.8.3).

First, we get a sense of how comparable the distances are likely to be by looking at the "self-distance" for each sample - the distance between a sample's Illumina and Nanopore data. As the sequencing data originate from the same source, we know the self-distance for any sample *should* be 0. However, we also know there are major technical differences between Illumina and Nanopore; therefore, small variability in self-distance is likely. We plot the self-distances in Figure 3.11 and see that 64% (96/150) of the samples have a distance of 0 between their Illumina (COMPASS) and Nanopore (`bcftools`) data, with 84% (126/150) less than 2 SNPs apart. All samples have a self-distance of less than 9, except one sample (`mada_1-33`), which has a self-distance of 53. We investigated the possibility of a sample mix-up being the cause of this discrepancy but could not find any such convincing evidence.

Next, we look at the pairwise SNP distance relationship, akin to that in Section 3.7. Figure 3.12 shows that the mixed SNP distances have a similar relationship to the single-technology correlation in Figure 3.5. The difference, however, is that in Figure 3.12, the y-axis represents the distance between one sample's Illumina data and the other's Nanopore. There are twice as many data points in this plot as the distance between two samples is not necessarily reciprocal for mixed modality distances (as we saw with the self-distances). That is, for two samples a and b, $distance(a_I, b_N) \neq distance(a_N, b_I)$, where I and N refer to Illumina and Nanopore data respectively.

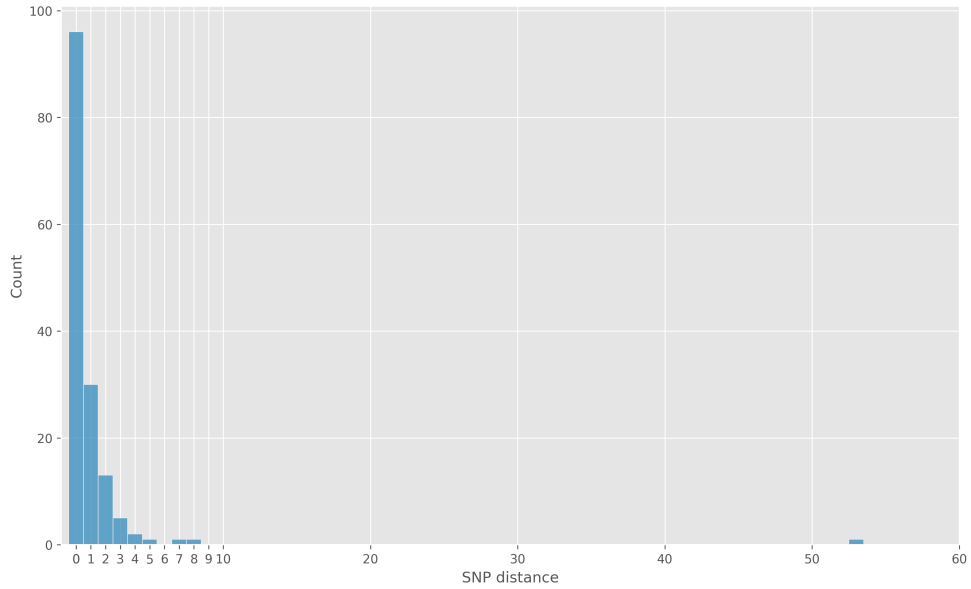


Fig. 3.11: Mixed modality "self-distance". This plot shows the SNP distance (x-axis) between each sample's COMPASS (Illumina) and `bcftools` (Nanopore) VCF calls.

In the zoomed inset window of Figure 3.12, there is a cluster of outlying points with a higher mixed distance than Illumina distance. All of these points relate to combinations of 6 particular samples. We investigated these samples for evidence of a sample swap or low data quality, but nothing was found to support such a claim. In reality, it just seems the Nanopore data for some of the samples are quite different to the Illumina data of the other samples.

We now examine transmission clusters for mixtures of Nanopore and Illumina data using the same SNP thresholds from Section 3.8. The SNP threshold we use when comparing different modalities is the Illumina SNP threshold. The mixture ratios we investigate are 0.01, 0.05, 0.1, 0.25, 0.5, 0.75, and 0.9. That is, for a ratio of 0.25, we *randomly* allocate 25% of the samples to Nanopore and the remainder to Illumina. For each SNP threshold and ratio, we calculate the XCR, SACR and SACP that the clustering produces. We repeat this process 1000 times for each threshold and ratio to simulate different mixtures of sample/technology pairs. The intention for simulating so many different mixed pairs is to provide insight into how robust clustering is to different ratios of sequencing datasets.

The results of these simulations are shown in Figure 3.13 (full summary statistics in Table B.1). We found that for all SNP thresholds and ratios, the median SACR was 1.0. In other words, regardless of the Nanopore/Illumina mixture ratio, for all thresholds we used, no sample is missed from its expected clustering - on average. The

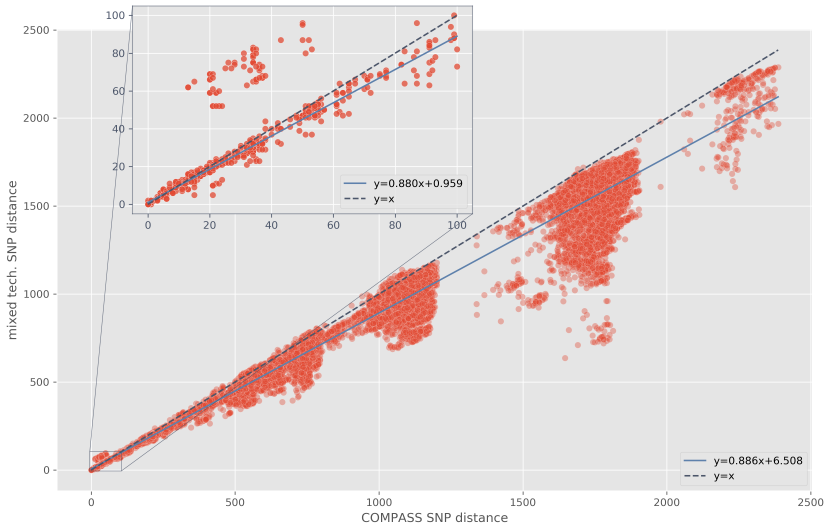


Fig. 3.12: The relationship of the distance between all pairs of samples based on Illumina (COMPASS) VCF calls (X-axis) and mixed COMPASS-`bcftools` calls (Y-axis). The black, dashed line indicates the relationship we would expect if the distance between a pair of samples were the same for both approaches. The blue line indicates the line of best fit based on fitting a robust linear regression model to the data. The inset gives a closer look at the relationship for all sample pairs where the COMPASS distance is less than or equal to 100 SNPs. The legend indicates the linear equations for the lines. Note: to prevent model skew, we do not include self-distance pairs.

SACP values decrease somewhat as the Nanopore ratio increases. However, the lowest median SACP value was 0.845 (threshold 12, ratio 0.9), which is also the SACP value obtained for the Nanopore-only clustering in Section 3.8.2 with the same threshold. The XCR values tend to increase slightly with the addition of more Nanopore samples. In the most extreme case, 0.057 was the highest XCR value in any simulation (SNP threshold 5). Incidentally, this is the same as the XCR obtained for the Nanopore-only clustering of the same SNP threshold, which equates to 7 of the 122 non-clustered samples being clustered. However, regardless of the XCR, no samples that should have been clustered were missed (on average).

3.9.1 Summary

In this section, we have shown that putative transmission clusters constructed using mixtures of Illumina and Nanopore data are consistent with those produced by Illumina data alone. As such, datasets from different sequencing technologies can be combined for transmission clustering analysis using the methods in this chapter.

3.10 Discussion

Recent work from Smith *et al.* is the first effort to assess Nanopore for the clustering of *M. tuberculosis* samples based on genetic distance [91]. While their work had more samples than ours (431), the SNP distance comparison details were very brief and only presented for a subset of 14 samples. They present the results as a distance matrix and leave it as an exercise for the reader to compare the Illumina and Nanopore matrices. There is no quantification of the clustering similarities or investigation into whether Illumina and Nanopore data can be mixed for this application. In contrast, the work presented in this chapter provides a detailed analysis of all of these topics - and more.

In addition to the conventional single-reference variant-calling approach, we also assessed the performance of the genome graph method presented in Chapter 2, for *M. tuberculosis*. We built two *M. tuberculosis* population reference graphs with different variant densities. Intuition would say that the more variants in the PanRG, the better the ability to find and call variants. However, we found the opposite. The sparse PanRG produced marginally higher precision and recall, on average, compared to its dense counterpart. As the computational resources required to construct and operate the sparse PanRG are a lot less than the dense, we chose to use it for the subsequent analysis. The lack of improvement by adding more variants is consistent with previous

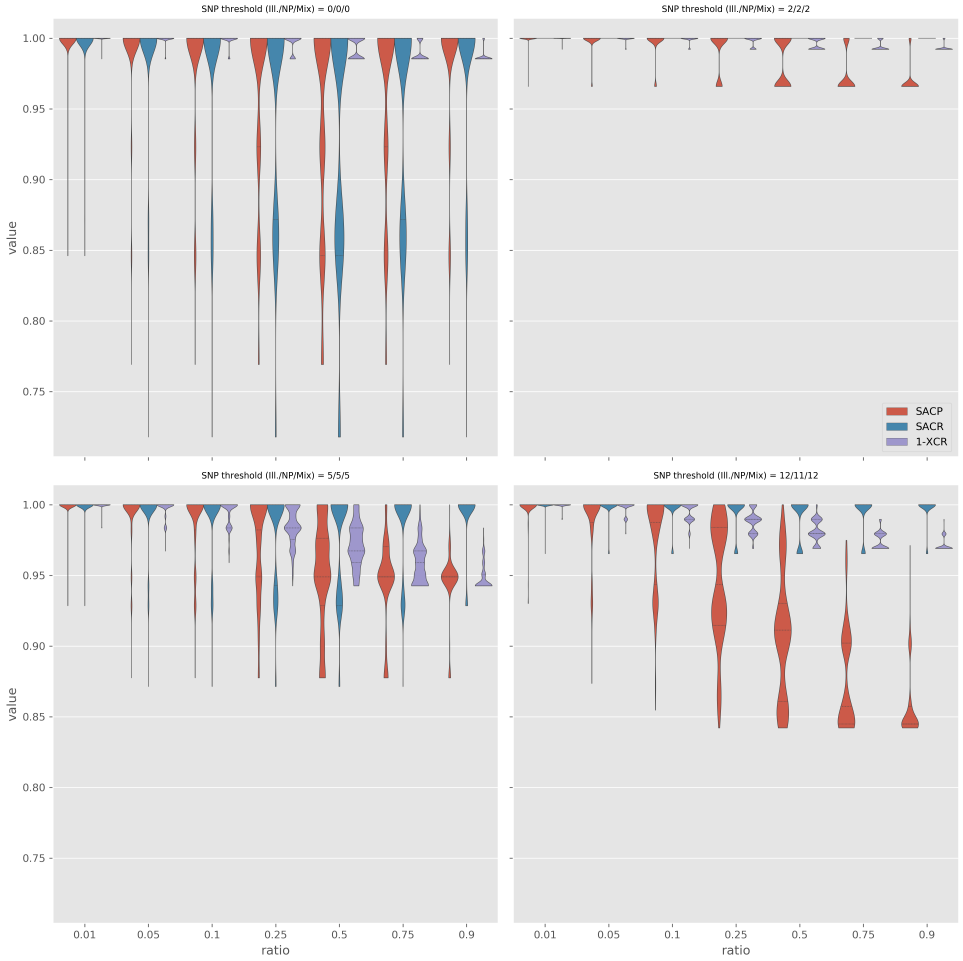


Fig. 3.13: Simulating various ratios (x-axis) of Nanopore/Illumina sample mixtures. The different thresholds (subplots) indicate the cutoff for defining samples as part of a cluster. The y-axis depicts the Sample-Averaged Cluster Precision and Recall (SACP/SACR) and Excess Clustering Rate (XCR) distributions over all simulation runs (XCR is shown as (1-XCR) for better axis-scaling). For each ratio/threshold combination we run 1000 simulations where the Nanopore and Illumina data is randomly split into the relevant ratio and clusters are defined based on the relevant threshold. The titles for each subplot indicate the SNP threshold used when comparing Illumina (Ill.), Nanopore (NP), or mixed-technology sample pairs.

work from Pritt *et al.*, who found a ceiling in gains by adding more variants [182]. They note that eventually, the extra variants cause complexity "blow-ups" that manifest as increased computational resource requirements and reference ambiguity, all of which lead to a decay in overall performance. This reduction in performance is the same thing we see. Many of the errors made by the dense PanRG relate to shared *k*-mers between alternate paths through sites in the graph. These shared *k*-mers, in turn, confuse the genotyping by adding coverage to multiple alleles. We discuss this further in Section 3.12.3 and investigate this complexity problem further in Chapter 4.

The initial step in this chapter was the first investigation of the precision and recall of Nanopore variant calls for *M. tuberculosis*. Previous work from Bainomugisa *et al.* has only looked at one sample and assessed variants in the *pe/ppe* genes [87]. While there are several Nanopore variant callers recently published, we chose to use `bcftools` due to its similarity to the Illumina strategy we are comparing to and for its ease of use. Many of the Nanopore variant callers are neural network-based and require considerable bioinformatics knowledge to operate and, in some cases, require training of variant models. As our goal in this chapter is to investigate the use of Nanopore by public health laboratories (and for clinical purposes), we try to use methods that can be easily duplicated by others who may not have extensive bioinformatics training. It is difficult to directly compare our precision and recall values to other Nanopore variant-calling studies as we value precision higher than recall for the work in this chapter. Much of the Nanopore variant-calling benchmarks focus on balancing precision and recall. The precision from both Nanopore variant-calling strategies we analysed were consistent with Illumina and much higher than previous Nanopore benchmarks [82, 84]. However, we acknowledge the unfair comparison to other works given the different focus. Recall for both `bcftools` and `pandora` were lower than Illumina - by quite a lot for `pandora`. Compared to other Nanopore variant-calling work, the recall values we can obtain are a few percentage points below the best [84, 85]. Given that we also report results for various variant filtering levels, we hope these can be used by others who may place a higher value on recall.

One unfortunate limitation of the variant-calling validation was the number of PacBio assemblies we could use. We sent 35 samples for PacBio sequencing, but we only received sufficient data for the assembly of 9 samples - and two of those failed QC due to technical difficulties in the sequencing lab. These results would have been even more robust with 35 validation samples; however, seven is comparable with the numbers used in other Nanopore variant calling evaluation work [82, 84, 85].

We have outlined three new metrics for comparing the similarity of two transmission clustering approaches, the sample-averaged cluster recall (SACR) and precision (SACP), and the excess clustering rate (XCR). SACR and SACP are derived from the set-similarity measure, the Tversky Index [180]. XCR has not been described elsewhere to the best of our knowledge. Cluster similarity is a rich field of research, yet, there are not many examples of this quantitative approach to comparing transmission cluster methods. Of the studies that *do* compare clusters between methods, none provide the level of information provided by SACR, SACP, and XCR collectively. Meehan *et al.* use a clustering rate metric, which is the number of samples clustered, minus the number of clusters and then divided by the number of samples [183]. Roetzer *et al.* focused on manually comparing a single large cluster but did not compare all clusters [184]. Perhaps the closest to our approach is that by Stimson *et al.* [123], who use an information theory metric called variation of information (VI) [185]. VI works well and is not too dissimilar to our approach. It measures how much information is lost and gained in between two clustering approaches.

Our main reason to forego these previous methods in favour of our three has to do with the granularity of information. The studies mentioned all use a single metric to classify the performance of the clustering. However, using SACR, SACP and XCR, we see how changes in the methods for producing clusters impact whether samples are missed from clusters (SACR), wrongfully added to existing clusters (SACP), or if previously unclustered samples form new clusters (XCR). Such granularity allows users to tweak their clustering approach to meet their situation. For example, while we place a higher value on SACR, others may find the reduction of cluster merging is of more importance and can focus on improving SACP instead. A single metric does not allow for this kind of targeted evaluation.

The first important finding of this chapter is that Nanopore data can produce transmission clusters comparable to Illumina. Indeed, `bcftools` and `pandora compare` do not miss any samples from clusters - the most important consideration for transmission chain investigation [117]. This result agrees with the only other *M. tuberculosis* study of this kind [91]. Additionally, Nanopore's suitability for transmission investigation has been confirmed for other pathogens such as Human metapneumovirus [186], Shiga toxin-producing *E. coli* (STEC) [86], and *Neisseria gonorrhoeae* [85].

It is essential to highlight that the focus of this work is not as a variant-calling benchmark for WGS technologies. We acknowledge that COMPASS may not be the best Illumina-based variant calling strategy. Indeed, there are many bioinformatic pipelines available for the analysis of *M. tuberculosis* Illumina data, all with different

results from one another [124]. Instead, we take an approach used by PHE and ask whether Nanopore can provide information of the same quality. In effect, our study is a "non-inferiority" one; we are attempting to show that Nanopore is not *worse* than Illumina; as such, we can treat Illumina as "truth" in this respect. For the application of clustering *M. tuberculosis* genomes based on genetic distance, we find Nanopore does provide comparable information when using `bcftools` to call variants. In addition, we found that using the multi-sample comparison mode of `pandora` we also succeed in clustering all samples that should be clustered, albeit at the cost of adding more false-positive connections.

While the precision of variant calls for `pandora` was as high as Illumina, the clustering produced by the single-sample mode (`map`) was much worse than the other approaches. In general, the distances between samples based on `pandora map` variant calls were much higher than the Illumina data implied they should be. One point that contributes to this difference is the subtle difference in how we generate the `pandora map` consensus sequences. The main difference compared to `bcftools` and Illumina is that when a position in the H37Rv reference genome is missing from the `pandora map` VCF, we assume it is the reference position, rather than nullifying it as we do with COMPASS and `bcftools`. We initially took the nullify approach for missing positions but found this lead to a substantial under-calling of the distances. The bulk of the extra pairwise differences (false positives) called by `pandora map` were positions missing from one of the samples and present in the other. 96% of those false-positives positions were filtered out in the COMPASS and `bcftools` VCFs due to evidence of heterozygosity. Ultimately, this issue stems from the fact that COMPASS and `bcftools` make calls at all positions of the genome (with read depth), while `pandora` only makes calls at sites with alternate alleles. A new approach for calculating the distance between `pandora` single-sample VCFs certainly warrants further investigation.

The difference in clustering obtained by the two `pandora` approaches highlights their intended use cases. The multi-sample approach, `pandora compare`, was designed for allowing the comparison of collections of samples. It integrates information from *all* samples by selecting a consensus sequence that best approximates them and then calls variation against that consensus. This approach allows for easily identifying differences between samples as the VCF produced by `pandora compare` has genotype information for all samples at all sites. While `pandora compare` did not miss any samples from their correct clusters, it did incorrectly join some clusters and create new clusters from samples Illumina deemed singletons. This incorrect joining of samples

and clusters is not entirely unexpected. Incorrectly joining samples indicates that the distances between samples are lower than expected for `pandora compare` (this is supported by Figure 3.5). Given the `pandora` variant calls showed significantly lower recall than COMPASS and `bcftools` (see Figure 3.4), a smaller distance between samples is expected. Two obvious ways of improving recall are masking less of the genome (see Section B.3) or using less stringent variant filters (Section 3.6.4).

In addition to acknowledging that this variant-calling approach may not be the absolute best approach, we also acknowledge that SNP distance clustering has shortcomings. Again, our intention is not to claim to be the best clustering method but to mimic the process currently used by PHE - which is the SNP threshold approach used here. Stimson *et al.* recently published a notable study showing that combining a SNP threshold approach with epidemiological data can lead to superior transmission chain reconstruction compared to SNP threshold alone [123]. With the establishment of Nanopore's ability to provide accurate SNP threshold-based clusters, it seems certain that the inclusion of epidemiological data using the same approach as Stimson *et al.* can only improve inference for this application.

With the knowledge that Nanopore can detect likely clusters of transmission for *M. tuberculosis* we ask a logical next question: can transmission clusters be accurately constructed from a mixture of Nanopore and Illumina data? As Nanopore sequencing increasingly pervasive, it seems inevitable that groups using different sequencing modalities will want to compare data. We find that they can be mixed and produce clusters consistent with Illumina-only data.

This analysis is the first known case (to the author's knowledge) of testing this mixing of data for *M. tuberculosis*. The mixture of Nanopore and Illumina consensus sequences have been investigated for hepatitis C [187], and STEC [86], with the authors also finding the modalities can be mixed without degradation of results. Others have also compared phylogenetic trees constructed from a combination of the two modalities [86, 188, 189] with similar findings. Perhaps the unique insight from our work is that we assess the effect of different mixture ratios on clustering.

Another interesting insight from this study of technology mixtures is self-distance (Figure 3.11). In their work on *N. gonorrhoeae*, Sanderson *et al.* found a median self-distance of 5, with a range of 1-10 and interquartile range (IQR) of 3-6 ($n = 8$) [85]. While Greig *et al.* saw self-distances of 5 and 6 ($n = 4$) in STEC [86]. In contrast, we found a (`bcftools`) median self-distance of 0 with an IQR of 0-1 ($n = 150$). Our range was 0-53, and with the outlier of 53 removed, the range becomes 0-8. Both of these studies used similar variant filtering strategies to ours, except with different

variant callers - highlighting the need for continued standardisation of Nanopore variant calling, or even recommendations for specific species.

3.11 Conclusion

In conclusion, the work in this chapter has shown that Nanopore data can produce transmission clusters consistent with those from Illumina. Additionally, it is also possible to mix data of the two modalities and produce concordant clusters. Finally, we provide the first evaluation of Nanopore variant-calling for *M. tuberculosis*, and three new metrics for assessing transmission cluster similarity.

These results are consistent with another *M. tuberculosis* Nanopore-based transmission cluster study and similar work on other bacterial and viral pathogens. As a result, we believe Nanopore sequencing has reached sufficient quality to be considered for public health investigation of transmission clusters.

3.12 Future work

3.12.1 Dataset with known epidemiological information

Perhaps the most important follow up of the work in this chapter is to gather a dataset with epidemiologically linked cases and known transmission clusters. While these datasets do exist for Illumina data, there are none yet with matched Illumina and Nanopore sequencing. Matched sequencing data is necessary to know that differences in DNA are solely driven by sequencing technology. A dataset with solid evidence for transmission clusters would remove the main limitation to this chapter and be an even stronger statement for the use of Nanopore sequencing in public health laboratories.

3.12.2 Computational performance of variant calling

In Section 3.6.5 we assessed the time and memory usage for variant calling with `bcftools` and `pandora`. `bcftools` in particular had, in the worst case, the highest memory and CPU of the callers. Nearly all of this time and memory is spent realigning reads in the pileup in order to calculate the base alignment quality (BAQ) score [190]. When we disabled this BAQ setting for one sample, the CPU time dropped from 3 hours to 30 minutes (6-fold decrease) and peak memory reduced from 58GB to

70MB (829-fold decrease). However, this did come at the cost of a slight reduction in precision and recall. As we write this chapter, the newest release of BCFtools (version 1.13) has addressed this problem by only doing the BAQ realignment in areas overlapping problematic indel sites. Their testing shows this drastically reduces the peak memory and overall runtime and *increased* recall (the realignment can sometimes be detrimental). As such, an obvious task for future development would be to rerun this analysis with the latest `bcftools` version and assess the expected changes in computational resource usage and recall.

Much of the memory and CPU time in the `pandora` pipelines lies in updating the multiple sequence alignments used to build the PanRG after novel variants have been added. Recent work by Leandro Ishi in our research group has produced a prototype of the `make_prg` program that significantly reduces the time and memory required to update the PanRG (as discussed in Section 2.6.4). It remains to be seen whether these updates will also improve `pandora`'s precision and recall, but it will undoubtedly improve the computational requirements.

3.12.3 Improving PanRG construction

The current process for building the *M. tuberculosis* PanRG is, for each locus, to apply a single VCF alternate allele to the reference sequence for that locus and collect all of these mutated sequences into a multi-sequence FASTA file. One limitation of this approach is that variants do not always occur in isolation like this. Where this becomes important is when turning an MSA into a PRG with `make_prg`. An important parameter in this process is the minimum match length, m. When two variants are within m positions of each other, creating two separate sequences for them (as we do) creates alternate paths in the PRG, with neither path containing the correct allele combination. This situation is best understood with an example. We set m to 3 and have two variants at positions 2 and 4 - in a hypothetical genome sequence `AAAC`. The first variant is a SNP changing an `A` to a `T` and the second a `C` to a `G`. The two mutated sequences we produce for these two variants are `ATAC` and `AAAG`. Because these two sequences do not have a minimum match length of 3 or more, they become two alternate paths in the PRG. However, these two variants come from the same sample, so in reality, the true sequence is `ATAG`. When using the PRG containing the two alternate alleles, if we have a sample that contains both of the variants (i.e., `ATAG`), it does not match either of the two alleles in our PRG, even though they come from a sample with `ATAG` at this site. Ultimately, we rely on the *de novo* variant discovery

from Chapter 2 to fix this. Unfortunately, this does not always work and, as we will see in Chapter 4, even if *de novo* discovery adds the correct allele combination, `ATAG`, we now have three alleles that could share minimizing *k*-mers. Having shared minimizers over multiple alleles at the same position can lead to read coverage across all of those alleles - ultimately skewing genotyping.

One way to minimise these excess alleles would be to construct the PanRG by producing a single sequence at each locus from *all* variants for a sample - rather than a sequence for each variant we use its actual haplotype. The reason we did not construct the PanRG in this fashion in Section 3.5 was that for each locus, we would have had to perform an MSA on *n* sequences - where *n* is the number of samples. Instead, we chose to apply single variants as the number of variants in a locus was, in most cases, *much* smaller than *n* and thus, the MSA ran quicker and used much less memory.

In addition to improvements in variant inclusion, some changes can be made in the masking of loci. Our current method of removing loci from the PanRG when they have 30% or more overlap with a genome mask (Section B.3) leads to approximately 6% of loci being removed - 10% of the genome. As the genome mask used covers 7.4% of the genome, we remove more than is necessary, which impacts our recall. A recent study by Marin *et al.* has shown this genome mask to be excessive, and they present a new mask that covers only 4% of the H37Rv reference genome [191]. So a first step for improving the recall of `pandora` would be to rebuild the PanRG using this new mask.

3.13 Availability of data and materials

The pipelines and scripts used in this chapter are available at https://github.com/mbhall88/head_to_head_pipeline. A special mention must go to the workflow management program `snakemake` [192], which was used to coordinate all analyses. All figures were generated using the Python libraries `matplotlib` [193], `seaborn` [194], and `bokeh` [195].

Chapter 4

Predicting *M. tuberculosis* drug resistance

4.0 Publication and collaboration acknowledgements

A manuscript comprising the work in this chapter and Chapter 3 is currently in preparation. In addition to the acknowledgements in Section 3.0 the work not completed by myself in this chapter was the drug susceptibility testing (DST) in the laboratory.

The DST for the samples from Madagascar was conducted by Marie Sylvianne Rabodoarivelo and Simon Grandjean Lapierre. In addition, the DST for the South African data was performed by waiting for email from Anzaan and Tash confirming this

4.1 Introduction

Resistance to antibiotics is a quantitative phenotype that has been intensively studied for decades. Typically this is measured by growing replicates of a defined inoculum of the bacterium on a controlled medium; each replicate having a different concentration of antibiotic added. The minimum inhibitory concentration (MIC) is the lowest concentration needed to prevent bacterial growth. The genetic causes of drug resistance have proven comparatively easy to at least partially decode, compared with studies of human diseases, and it is possible to predict resistance to numerous anti-tuberculosis drugs by simply testing for the presence of a list of SNPs, commonly referred to as a catalogue (see Section 1.5.2).

Antimicrobial resistance (AMR) predictions from Illumina whole-genome sequencing (WGS) data are now routine for *M. tuberculosis* infections in an increasing number of public health agencies. In particular, a WGS-based susceptible prediction for first-line drugs - isoniazid, rifampicin, ethambutol, and pyrazinamide - is now considered accurate enough for clinical use [133]. However, being a relatively new sequencing modality, Nanopore has yet to see such large-scale validation.

As discussed in Chapter 3, routine Illumina WGS is not readily available or feasible in many high-burden tuberculosis (TB) settings. However, Nanopore sequencing is proving to be a useful alternative is such contexts [92, 94, 196, 197].

Only two *M. tuberculosis* AMR prediction tools - TB-Profiler [136] and Mykrobe [45] - have reported results for, and support, Nanopore data; albeit with small sample sizes ($n = 3$ and 5 respectively). A recent study from the New York State Department of Health is the largest validation of Nanopore AMR predictions to date, assessing 431 samples. However, they use in-house (closed-source) custom scripts and mutation catalogues, which is unsuitable for general use.

The software package Mykrobe provides AMR predictions and lineage classification for *M. tuberculosis* - amongst other species. It has been validated for Illumina WGS on a comprehensive global cohort and uses a mutation catalogue (panel) that aggregates curated resistance- and susceptibility-associated variants from multiple sources [45]. However, as mentioned above, `mykrobe` has only been validated on five Nanopore samples.

A primary aim of this chapter is to test Nanopore-based AMR predictions from `mykrobe` with a larger sample size. We address this in two steps. First, by assessing whether genotype-based AMR predictions from Nanopore data are concordant with those from Illumina. And second, by checking the concordance of both Illumina and Nanopore predictions with gold-standard culture-based drug susceptibility testing (DST) phenotypes, where available. In both cases, we use a fixed mutational catalogue to ensure differences in predictions are purely sequencing modality-driven.

One acknowledged limitation of `mykrobe` and all other *M. tuberculosis* AMR prediction tools is that they do not detect off-catalogue (novel) mutations. That is, they cannot detect mutations in resistance-associated genes that do not appear in the catalogue used by the tool. However, a recent study from the CRyPTIC Consortium showed that classifying such novel mutations as "unknown" can improve the pan-susceptibility prediction for first-line drugs [133].

To this end, we develop and evaluate a genome graph-based *M. tuberculosis* AMR prediction software tool, `drprg`, which can produce these unknown predictions when

off-catalogue mutations are detected. We use the methods and understanding developed in Chapter 2 and Chapter 3 to build a *M. tuberculosis* PanRG for AMR-associated genes and genotype mutations with `pandora`. In particular, we use the *de novo* variant discovery functionality of `pandora` developed in Chapter 2 to identify off-catalogue mutations and evaluate how these impact AMR predictions. We show that the novel variants discovered by `drprg` are precise and reduce the number of missed resistance calls.

4.2 Dataset

The *M. tuberculosis* samples used in this chapter are those described in Section 3.2. We use the 150 samples that passed quality control (Section 3.4).

4.2.1 Drug susceptibility testing

See Section C.1 for the DST phenotyping methods.

In total, 128 samples have phenotypic information for at least one drug, with 80 having phenotypes for eight drugs. Table 4.1 shows the number of samples with culture-based DST for each drug, and Figure 4.1 depicts the combinations of drugs available for samples. For instance, in Figure 4.1, the third column reveals that 29 samples have phenotype information for ofloxacin and amikacin. At the same time, row three shows that 51 samples have available DST for kanamycin.

Although line probe assay (LPA) results are also available for many samples, we base our phenotype concordance analysis on the culture-based DST data. However, we do use the LPA information to inform reasons for possible errors with resistance prediction. Table C.1 and Figure C.1 in Section C.1.3 show the full available DST data from both culture-based and LPA methods.

4.3 Drug resistance prediction with genome graphs

In this section, we describe a method to use genome graphs for predicting drug resistance in *M. tuberculosis*; building on previous work in this thesis constructing *M. tuberculosis* PanRGs (Section 3.5) and calling variants (Section 3.6.4 and Chapter 2). The tool we developed, `drprg` (Drug Resistance Prediction with Reference Graphs),

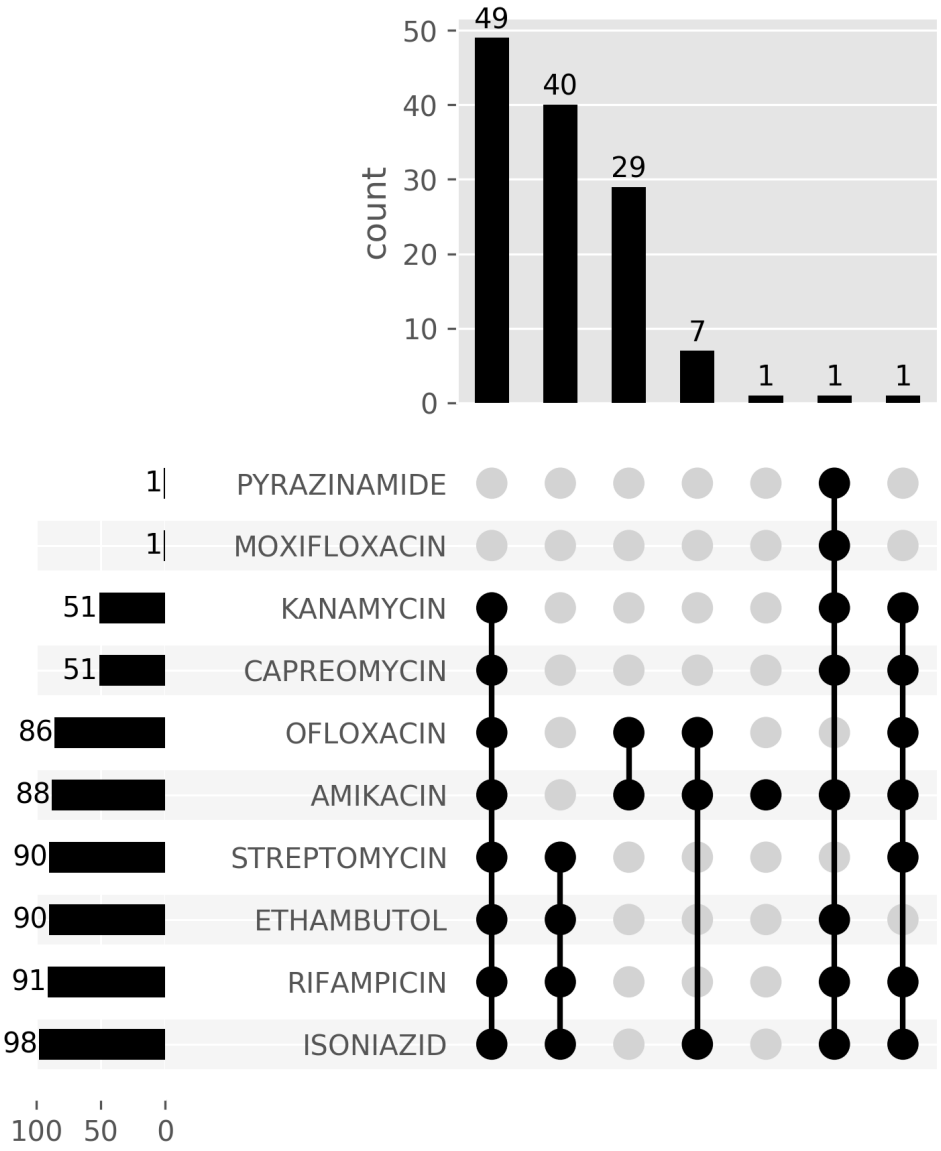


Fig. 4.1: Culture-based drug susceptibility data available for samples. Each row is a drug, and the columns represent a set of samples with phenotype information for those drugs with a filled cell. The top panel shows the number of samples in the set for that combination of drugs. The bar plot in the left panel shows the number of samples with phenotype information for that drug.

Drug	Count
Amikacin	88
Capreomycin	51
Ethambutol	90
Isoniazid	98
Kanamycin	51
Moxifloxacin	1
Ofloxacin	86
Pyrazinamide	1
Rifampicin	91
Streptomycin	90

Table 4.1: Culture-based drug susceptibility data available for samples. The counts are the number of samples with phenotype information available for that drug.

is written in the Rust programming language and can be found at https://github.com/mbhall88/drprg.

While tools already exist to predict *M. tuberculosis* AMR, the unique component of `drprg` is the ability to call novel variants. As the CRyPTIC Consortium recently showed, refusing to predict drug phenotypes in cases where "unknown" mutations are present in the associated target gene can lead to increased detection of pan-susceptibility for first-line drugs [133]. However, no tool usable by others was made available in that work, and reproducing the results requires running multiple separate tools and scripts. `drprg` offers a single program for producing this information.

`drprg` has two distinct phases. In the first phase, we build an index and (optionally) a PanRG from a panel (catalogue) of mutations known to cause resistance or susceptibility. Second, we genotype Illumina or Nanopore reads against the PanRG, and phenotype predictions are made for the drugs in the provided panel.

We now outline these two steps in detail.

4.3.1 Constructing a panel reference graph

The first stage of predicting drug resistance from `drprg` is coordinated by the `build` subcommand. As input, `build` requires a panel of mutations and a reference genome and annotation. We use the reference and annotation for the *M. tuberculosis* strain H37Rv (accession NC_000962.3) and the default panel used by `mykrobe` (v0.10.0) [45].

The panel of variants can be either resistance- or susceptibility-associated. Each entry in the panel file describes the gene the variant occurs in, the mutation it causes, and any drug it impacts. The mutation can represent a nucleic or amino acid change and is of the form reference, position, alternate. For example, a DNA mutation, `A5T`, indicates the reference base, `A`, at position 5 in the gene, is changed to a `T`. An `X` indicates any nucleic or amino acid other than the one listed as the reference.

After loading the panel, we generate a reference sequence for each listed gene using the provided reference genome and annotation. If the optional padding argument is given, we add the provided number of bases to the start and end of each gene sequence; we use 100bp of padding for the work in this chapter.

The next step is to convert the panel into a VCF representation. For protein mutations, we convert the amino acids into their respective DNA forms (all possible codons). We confirm that the reference codon matches the reference sequence at the given position in the gene. Protein coordinates (positions) are carefully converted to nucleic acid-space, taking into account the transcription strand specified in the annotation. DNA mutations are likewise checked against the reference. A VCF entry is then generated for each panel entry using the DNA representation. For protein mutations and DNA mutations with an alternate allele of `X`, all possible changes (except stop codons) are listed in the entry. Figure C.2 shows an example panel and associated VCF.

Another optional parameter of `drprg build` is for the user to provide a prebuilt PanRG. If no prebuilt PanRG is provided, `drprg` will construct one from the panel. (The previous steps just outlined remain the same regardless of whether a prebuilt PanRG is provided.) `drprg` builds a panel-based PanRG in a similar fashion to the method described in Section 3.5. That is, for each VCF entry and alternate allele, we replace the gene reference position(s) with the alternate and combine all such mutated sequences into a single FASTA file. We perform a multiple sequence alignment (MSA) on each file, followed by converting the MSA into a local PRG using `make_prg` (Section 1.3.1). Finally, the resulting local PRGs, which represent each gene in the panel, are combined into a single PanRG and indexed with `pandora`.

For the work in this chapter, we chose to use a prebuilt, population-based PanRG, rather than the panel PanRG constructed by `drprg`. We chose a prebuilt PanRG because, in the early stages of testing and developing `drprg`, we found the panel-based PanRG density and lack of haplotype information were causing a lot of missed resistance (false negatives). Section C.2.2 provides a thorough investigation into these problems with the panel-based PanRG.

The population-based PanRG we use is built from the same variants as the sparse PanRG in Section 3.5 - with an important difference. In Section 3.12.3 we discussed a potential improvement for the PanRG construction process whereby whole haplotypes are applied to a reference sequence. When we constructed the original sparse PanRG in Section 3.5, we applied each variant, *in isolation*, to the reference sequence. So, if a sample has 3 variants, we built the PRG from the reference sequence, plus 3 mutated versions of that reference (see Section 3.12.3 for a detailed example). Instead, for the PanRG we use in this chapter, we apply *all* variants for a sample in the sparse VCF to the (gene) reference sequence - producing a single (gene) haplotype sequence for that sample. We do this for all genes, and use the same MSA, `make_prg`, `pandora` index approach mentioned above (and in Section 3.5). One major difference being the use of a `make_prg` prototype (v0.2 - mentioned in Section 3.12.2) with a minimum match length of 5. This prototype, developed by Leandro Ishi, retains information about the clustering of sites for each local PRG, allowing for quicker updating of PRGs with novel variants (as discussed in Section 4.3.2).

4.3.2 Predicting resistance phenotype

The second and final stage of predicting drug resistance with `drprg` is the `predict` routine. It takes an index produced by `drprg build` (Section 4.3.1) and a file containing sequencing reads (Illumina or Nanopore). One optional parameter of interest for the work in this chapter is the ability to discover novel variants.

If requested, the first stage of `drprg predict` is novel variant discovery in the sequencing reads with `pandora discover` (version 0.9.0). This version of `pandora` differs to the one used in Chapter 3 in that it outputs all novel variants into a single file compatible with the `make_prg` prototype used in Section 4.3.1. Next, `make_prg` updates the `drprg` PanRG with these new variants and the resulting updated PanRG is indexed with `pandora`.

One downside to using a population-based PanRG as we do, is that unlike the panel-based PanRG, it does not contain all panel variants. As such, for the work in this chapter, we request novel variant discovery in `drprg predict`.

The second step of `predict` is genotyping of the sample's sequencing data against the PanRG (updated or original depending on if variant discovery is requested) with `pandora map`. An important part of the genotyping step is that we force `pandora` to output variant coordinates with respect to the gene reference sequences in the index

from `build`. That way, the resulting genotyped VCF coordinates can be compared to the panel VCF constructed in Section 4.3.1.

After producing the `pandora` genotyped VCF, we filter out variants that do not meet the provided filtering criteria. `drprg predict` allows filtering based on: minimum and maximum read depth, strand bias, minimum genotype confidence, minimum fraction of read support (FRS), and maximum indel size. (See Section 3.6.4 for more information on these fields). For the results presented in this chapter, we use a minimum read depth of 3, a minimum FRS of 70%, a minimum genotype confidence score of 5, a maximum indel size of 20, and a minimum strand bias of 1% (i.e., we require at least 1% of total read depth on both strands). We ignore a variant in all subsequent analyses if it fails any of these filters.

Next, we make resistance predictions from the variants that pass filtering. For each call in the genotyped VCF, we fetch all entries in the panel VCF that have overlapping coordinates. If there are no panel variants that overlap the call and novel variant discovery is requested, we classify all drugs associated with the gene of the present call as "unknown" ('U'). If the call is a null genotype (`.`), we predict all overlapping panel variants as "failed" ('F').

If the variant call was not deemed unknown or failed in the previous step, we check for a match between the call and a panel allele. To determine whether a match exists, we iterate through each panel allele (including the reference) and extract an overlapping sequence between it and the called sequence. If the panel and called allele start at the same position, this is as simple as trimming both sequences to the shortest length of the two. Otherwise, we extract the subsequence denoted by the intersection between the two allele's intervals if they start at different positions.

For example, if the called allele, `AT`, starts at position 4 and the panel allele, `TG`, starts at position 5, their (half-open) intervals are $[4,6)$ and $[5,7)$ respectively. Thus, the intersection of these two alleles would be $[5,6)$, which yields a matching subsequence of `T` for both. When the two alleles do not have the same length - e.g., indels - we also track the matching sequences' length relative to the reference allele. That way, when there is more than one panel allele that matches the called allele, we return the allele with the match whose length is closest to the called allele.

We add two annotations to each of the genotyped VCF entries that overlap variants in the panel: `VARID` and `PREDICT`. `VARID` is a list of panel variants the position overlaps, and `PREDICT` is a prediction for each of those panel variants. If there was a non-reference match between a panel variant and the called allele, and the panel variant is resistance-causing, a resistant prediction is recorded. If there was no match,

or the match is with the reference allele, the prediction is susceptible; unless variant discovery was requested, in which case the prediction is unknown.

After adding the prediction annotations to each entry in the genotyped VCF, we produce a final prediction report as a JSON file. A prediction is provided for every drug present in the panel, along with the supporting evidence (variant(s)). We generate these predictions by going back through the genotyped VCF and using the `PREDICT` annotation added in the previous step. When different predictions are present for the same drug, the precedence, in order, is resistant, unknown, failed, and susceptible. Figure C.6 shows an example report.

All `drprg` results in this chapter were generated using the commit `cb4f9b8`.

4.3.3 Computational performance

We compare the peak memory usage and runtime for `drprg predict` with `mykrobe` - another genome graph-based AMR predictor. Figure 4.2 shows the computational performance of both tools, split by sequencing technology. On average, `drprg` is faster than `mykrobe` for both Illumina and Nanopore and uses less memory. As summarised in Table 4.2, `drprg` has a median CPU time of 111s (Illumina) and 178s (Nanopore) compared to `mykrobe`'s 197s (Illumina) and 249s (Nanopore). For memory usage, `drprg` had a median peak of 224MB (Illumina) and 346MB (Nanopore) compared with `mykrobe`'s 1193MB (Illumina) and 1195MB (Nanopore). However, these time and memory statistics are more than sufficient for both tools to be comfortably run on a standard laptop.

Building the `drprg` population-based PanRG and index took 175s of CPU time and had a peak memory usage of 368MB, which occurred during the `make_prg` step.

4.4 Concordance of Nanopore-based resistance predictions with Illumina

The first step in assessing Nanopore-based AMR predictions is to compare its level of agreement with genotype-based predictions from Illumina data, which have been extensively validated elsewhere (Section 1.5.2).

Given the comprehensive validation of `mykrobe`'s prediction power on Illumina data [44, 45], if Nanopore results are concordant, this will go a long way to endorsing

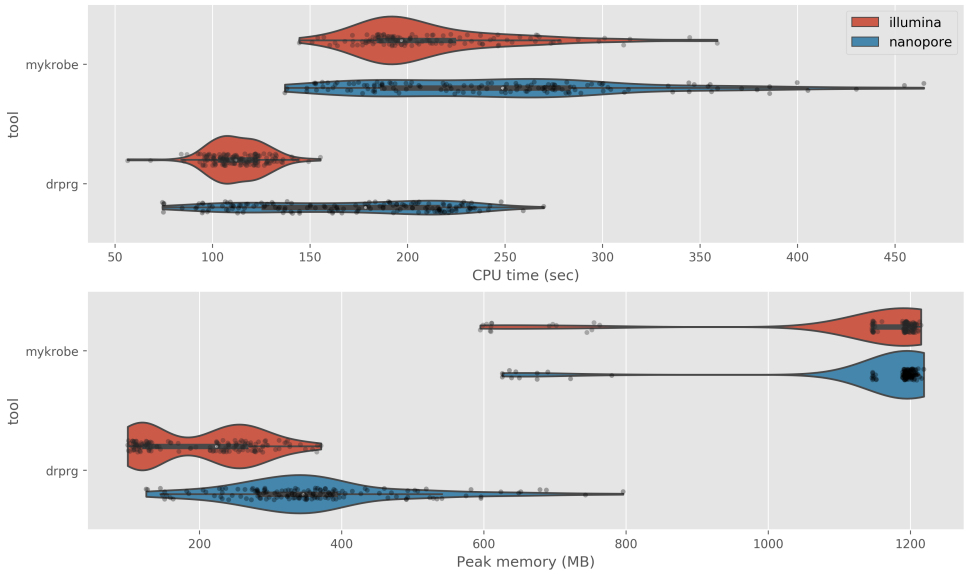


Fig. 4.2: The time (seconds; top) and memory (megabytes; bottom) usage of `mykrobe` and `drprg` for generating drug resistance predictions. Each tool is additionally split into sequencing technology, with Illumina in red and Nanopore in blue. Data points indicate individual results for a single sample.

Tool		`drprg`		`mykrobe`	
Technology		Illumina	Nanopore	Illumina	Nanopore
time (s)	mean	113	170	210	248
	std	14	49	39	70
	min	57	74	145	137
	25%	104	126	186	187
	50%	**112**	**178**	197	249
	75%	122	215	223	282
	max	155	270	359	465
memory (MB)	mean	203	357	1126	1151
	std	79	121	173	146
	min	99	125	595	626
	25%	123	285	1148	1192
	50%	**224**	**346**	1193	1195
	75%	265	390	1201	1201
	max	371	796	1215	1219

Table 4.2: Summary statistics for the CPU time and peak memory usage of drug resistance prediction with `mykrobe` and `drprg` for Illumina and Nanopore data. std=standard deviation

its use for AMR prediction. In addition, if the Illumina and Nanopore predictions from `drprg` are consistent with `mykrobe`, this is an excellent first validation of the method.

We compare the WGS-based drug resistance predictions from `mykrobe` (version 0.10.0; [45]) and `drprg` for both Illumina and Nanopore data, using the same catalogue of resistance-causing mutations. For the predictions from `mykrobe`, we altered some default settings as a result of the analysis in Section C.4. The Illumina expected error rate was decreased from 0.05 (default) to 0.001, and the preset Nanopore settings were disabled in favour of an expected error rate of 0.08 and disabling minor resistance calls (haploid model). The `drprg` settings are detailed in Section 4.3.2.

For this genotype concordance analysis, we assume the Illumina predictions from `mykrobe` are the true phenotype and compare the `mykrobe` Nanopore and `drprg` (both modalities) predictions accordingly. For each drug/sample combination, we classify the WGS predictions as true-positive (TP) or -negative (TN) when the prediction is resistant or susceptible, respectively, and matches the `mykrobe`-Illumina prediction. When the predictions do not match `mykrobe`-Illumina, we classify them as false-positive (FP) or -negative (FN) for resistance or susceptible predictions, respectively. For this concordance analysis, we ignore failed, and unknown resistance calls from `drprg` and consider them as susceptible (we assess the utility of these features in Section 4.7). Additionally, for `mykrobe` Illumina predictions, we interpret minor resistance calls ('r') as resistance ('R').

Across all drugs, there were 8 FN (missed resistance) calls by `mykrobe` Nanopore - out of a total of 387 expected resistant calls (see Table 4.3 for a full breakdown). Seven of those FNs are situations where the Illumina data has called minor resistance - a heterozygous call at a resistance-causing variant. As we use a haploid model for Nanopore (see Section C.4), this is unavoidable until a model that allows complex ploidy can be employed. We did attempt to use a diploid model for Nanopore, but found a dramatic increase in the number of FNs and FPs due to indel calls.

For the `drprg` Nanopore predictions, there were 32 FNs across all drugs. Ten of these were cases where `mykrobe` called minor resistance. Similar to `mykrobe` for Nanopore, `drprg` employs a haploid model for both Illumina and Nanopore; therefore, minor allele calls are not possible. Another 14 FNs were called resistance by `drprg` Nanopore, but the variant was filtered due to low FRS (less than 70% reads support the called allele) - meaning we call susceptible. All but one of these FRS cases was a promoter mutation C-15X in *fabG1*. A further two FNs were also filtered out due to strand bias. Lastly, five FNs were associated with missed indel calls in *pncA*; indels are a known systematic problem with Nanopore data [79].

`drprg` with Illumina data missed 17 resistance calls compared to `mykrobe` with Illumina. Ten of these were cases of `mykrobe` calling minor resistance, as mentioned above. Of the remaining 7 FNs, four were related to indel calls in *pncA*, and three have no evidence in the `drprg` VCF.

Of the 17 FPs made by `mykrobe` Nanopore, nine show strong support for the resistance-causing variant in the VCFs from Section 3.6 for both technologies. In each of these cases, the `mykrobe` results from Illumina also called the variant but filtered it out due to low coverage. A further eight are false-positive isoniazid indel calls made by Nanopore in a GC-rich region of *katG*.

The `drprg` predictions had 23 and 22 FPs for Nanopore and Illumina data respectively. The one difference being the Nanopore predictions have one more isoniazid FP from a *katG* indel call. 11/23 FPs are associated with indel calls in *pncA* causing a false resistance call for pyrazinamide. Indel calls were also the cause of two FPs for isoniazid (*katG*) and one for rifampicin (*rpoB*). Similar to the `mykrobe` Nanopore calls, 10 of the `drprg` FPs showed strong support for the resistance-causing variant in VCFs (Section 3.6) for both technologies, with the `mykrobe` results from Illumina having also called the variant, but filtered out due to low coverage. Additionally, in 11 FPs, the culture-based phenotype supports the `drprg` prediction.

4.4.1 Summary

In summary, `mykrobe` Nanopore AMR predictions are consistent with those from Illumina data. There were some discrepancies; however, all but one of the missed resistance calls was due to minor allele calls, which Nanopore cannot replicate yet. Additionally, many of the false resistance calls were either Nanopore making incorrect indel calls, or Illumina missing variants with strong support from the analysis in Section 3.6.

For most drugs, `drprg` predictions are concordant with `mykrobe`'s Illumina calls. More than half of the `drprg` missed resistance calls were due to it not being able to detect minor alleles. `drprg` Nanopore predictions also missed some resistant calls because of a low fraction of read support for the mutation in question. A major reason for false resistance calls from `drprg`, for both technologies, was incorrect indel calls in *pncA*. However, nearly half of the `drprg` false positives (SNPs) are potentially missed resistance on `mykrobe`'s part due to it having filtered out the calls due to low support.

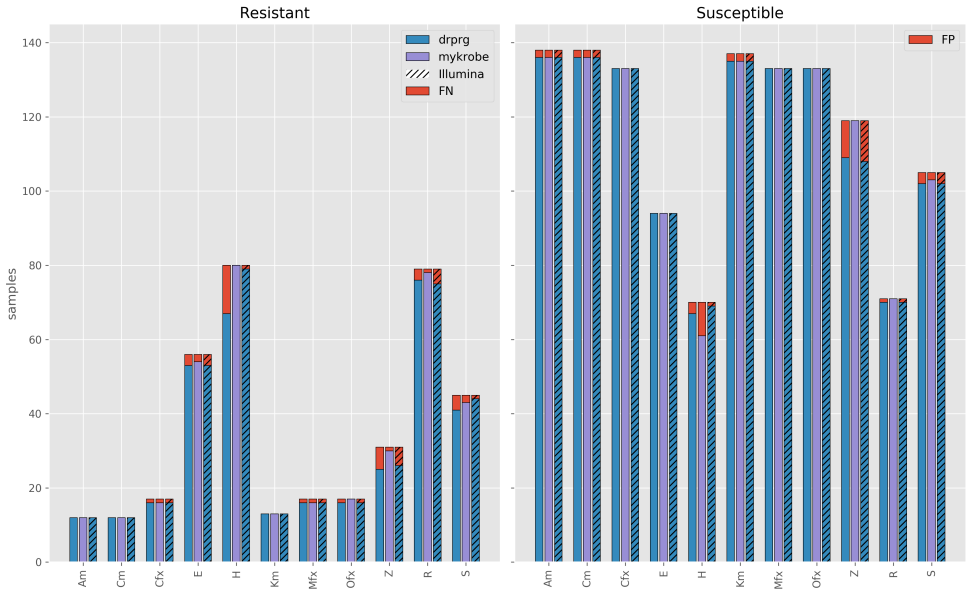


Fig. 4.3: Number of resistant (left) and susceptible (right) WGS-based drug resistance phenotypes correctly predicted by `mykrobe` Nanopore (purple) and `drprg` (blue) with Illumina (striped) and Nanopore (non-striped) data. For this analysis, the Illumina-based `mykrobe` predictions are considered truth and the other predictions are assessed accordingly. The red bars indicate missed (FN) or incorrect (FP) predictions. The x-axis shows the drugs for which `mykrobe` makes predictions. E - ethambutol; H - isoniazid; Z - pyrazinamide; R - rifampicin; S - streptomycin; Km - kanamycin; Am - amikacin; Ofx - ofloxacin; Cm - capreomycin; Mfx - moxifloxacin.

Drug	Tool	Technology	FN(R)	FP(S)	FNR(95% CI)	FPR(95% CI)	PPV(95% CI)	NPV(95% CI)
Amikacin	drprg	Illumina	0(12)	2(138)	0.0% (0.0-24.2%)	1.4% (0.4-5.1%)	85.7% (60.1-96.0%)	100.0% (97.3-100.0%)
	drprg	Nanopore	0(12)	2(138)	0.0% (0.0-24.2%)	1.4% (0.4-5.1%)	85.7% (60.1-96.0%)	100.0% (97.3-100.0%)
	mykrobe	Nanopore	0(12)	2(138)	0.0% (0.0-24.2%)	1.4% (0.4-5.1%)	85.7% (60.1-96.0%)	100.0% (97.3-100.0%)
Capreomycin	drprg	Illumina	0(12)	2(138)	0.0% (0.0-24.2%)	1.4% (0.4-5.1%)	85.7% (60.1-96.0%)	100.0% (97.3-100.0%)
	drprg	Nanopore	0(12)	2(138)	0.0% (0.0-24.2%)	1.4% (0.4-5.1%)	85.7% (60.1-96.0%)	100.0% (97.3-100.0%)
	mykrobe	Nanopore	0(12)	2(138)	0.0% (0.0-24.2%)	1.4% (0.4-5.1%)	85.7% (60.1-96.0%)	100.0% (97.3-100.0%)
Ciprofloxacin	drprg	Illumina	1(17)	0(133)	5.9% (1.0-27.0%)	0.0% (0.0-2.8%)	100.0% (80.6-100.0%)	99.3% (95.9-99.9%)
	drprg	Nanopore	1(17)	0(133)	5.9% (1.0-27.0%)	0.0% (0.0-2.8%)	100.0% (80.6-100.0%)	99.3% (95.9-99.9%)
	mykrobe	Nanopore	1(17)	0(133)	5.9% (1.0-27.0%)	0.0% (0.0-2.8%)	100.0% (80.6-100.0%)	99.3% (95.9-99.9%)
Ethambutol	drprg	Illumina	3(56)	0(94)	5.4% (1.8-14.6%)	0.0% (0.0-3.9%)	100.0% (93.2-100.0%)	96.9% (91.3-98.9%)
	drprg	Nanopore	3(56)	0(94)	5.4% (1.8-14.6%)	0.0% (0.0-3.9%)	100.0% (93.2-100.0%)	96.9% (91.3-98.9%)
	mykrobe	Nanopore	2(56)	0(94)	3.6% (1.0-12.1%)	0.0% (0.0-3.9%)	100.0% (93.4-100.0%)	97.9% (92.7-99.4%)
Isoniazid	drprg	Illumina	1(80)	1(70)	1.2% (0.2-6.7%)	1.4% (0.3-7.7%)	98.8% (93.3-99.8%)	98.6% (92.3-99.7%)
	drprg	Nanopore	13(80)	3(70)	**16.2% (9.7-25.8%)**	4.3% (1.5-11.9%)	95.7% (88.1-98.5%)	83.8% (74.2-90.3%)
	mykrobe	Nanopore	0(80)	9(70)	0.0% (0.0-4.6%)	**12.9% (6.9-22.7%)**	89.9% (81.9-94.6%)	100.0% (94.1-100.0%)
Kanamycin	drprg	Illumina	0(13)	2(137)	0.0% (0.0-22.8%)	1.5% (0.4-5.2%)	86.7% (62.1-96.3%)	100.0% (97.2-100.0%)
	drprg	Nanopore	0(13)	2(137)	0.0% (0.0-22.8%)	1.5% (0.4-5.2%)	86.7% (62.1-96.3%)	100.0% (97.2-100.0%)
	mykrobe	Nanopore	0(13)	2(137)	0.0% (0.0-22.8%)	1.5% (0.4-5.2%)	86.7% (62.1-96.3%)	100.0% (97.2-100.0%)
Moxifloxacin	drprg	Illumina	1(17)	0(133)	5.9% (1.0-27.0%)	0.0% (0.0-2.8%)	100.0% (80.6-100.0%)	99.3% (95.9-99.9%)
	drprg	Nanopore	1(17)	0(133)	5.9% (1.0-27.0%)	0.0% (0.0-2.8%)	100.0% (80.6-100.0%)	99.3% (95.9-99.9%)
	mykrobe	Nanopore	1(17)	0(133)	5.9% (1.0-27.0%)	0.0% (0.0-2.8%)	100.0% (80.6-100.0%)	99.3% (95.9-99.9%)
Ofloxacin	drprg	Illumina	1(17)	0(133)	5.9% (1.0-27.0%)	0.0% (0.0-2.8%)	100.0% (80.6-100.0%)	99.3% (95.9-99.9%)
	drprg	Nanopore	1(17)	0(133)	5.9% (1.0-27.0%)	0.0% (0.0-2.8%)	100.0% (80.6-100.0%)	99.3% (95.9-99.9%)
	mykrobe	Nanopore	0(17)	0(133)	0.0% (0.0-18.4%)	0.0% (0.0-2.8%)	100.0% (81.6-100.0%)	100.0% (97.2-100.0%)
Pyrazinamide	drprg	Illumina	5(31)	11(119)	**16.1% (7.1-32.6%)**	**9.2% (5.2-15.8%)**	70.3% (54.2-82.5%)	95.6% (90.1-98.1%)
	drprg	Nanopore	6(31)	10(119)	**19.4% (9.2-36.3%)**	**8.4% (4.6-14.8%)**	71.4% (54.9-83.7%)	94.8% (89.1-97.6%)
	mykrobe	Nanopore	1(31)	0(119)	3.2% (0.6-16.2%)	0.0% (0.0-3.1%)	100.0% (88.6-100.0%)	99.2% (95.4-99.9%)
Rifampicin	drprg	Illumina	4(79)	1(71)	5.1% (2.0-12.3%)	1.4% (0.2-7.6%)	98.7% (92.9-99.8%)	94.6% (86.9-97.9%)
	drprg	Nanopore	3(79)	1(71)	3.8% (1.3-10.6%)	1.4% (0.2-7.6%)	98.7% (93.0-99.8%)	95.9% (88.6-98.6%)
	mykrobe	Nanopore	1(79)	0(71)	1.3% (0.2-6.8%)	0.0% (0.0-5.1%)	100.0% (95.3-100.0%)	98.6% (92.5-99.8%)
Streptomycin	drprg	Illumina	1(45)	3(105)	2.2% (0.4-11.6%)	2.9% (1.0-8.1%)	93.6% (82.8-97.8%)	99.0% (94.7-99.8%)
	drprg	Nanopore	4(45)	3(105)	**8.9% (3.5-20.7%)**	2.9% (1.0-8.1%)	93.2% (81.8-97.7%)	96.2% (90.7-98.5%)
	mykrobe	Nanopore	2(45)	2(105)	4.4% (1.2-14.8%)	1.9% (0.5-6.7%)	95.6% (85.2-98.8%)	98.1% (93.3-99.5%)

Table 4.3: Comparison of `drprg` and `mykrobe` Nanopore drug resistance predictions with `mykrobe` Illumina predictions. For this comparison, we assume the `mykrobe` resistance prediction from Illumina data is correct and evaluate the other predictions accordingly. Bold text is used to highlight differences of note. FN=false negative; R=number of resistant samples; FP=false positive; S=number of susceptible samples; FNR=false negative rate; FPR=false positive rate; PPV=positive predictive value; NPV=negative predictive value; CI=Wilson score confidence interval

4.5 Concordance of genotype-based predictions with culture-based phenotypes

In this section, we assess the concordance of Nanopore and Illumina WGS genotype-based AMR predictions with gold-standard culture-based DST phenotypes.

We saw in Section 4.4 that Nanopore-derived resistance predictions are in line with those from Illumina when comparing to `mykrobe` Illumina-based predictions. However, as this concordance was not perfect, we explore the agreement of WGS predictions with culture-based phenotype - where this information is available. As we use the same catalogue for all WGS predictions, this analysis aims to assess both the prediction methods and sequencing modalities and determine whether either or both of these factors drive the genotype discrepancies.

The culture-based DST profiles gathered in Section 4.2.1 are considered the true phenotype against which we will appraise the WGS methods. We exclude moxifloxacin and pyrazinamide from this analysis as DST is only available for one sample. Table 4.1 outlines the total number of samples with phenotype information for each drug.

We follow the same classification approach as in Section 4.4, but use the culture-based phenotype as the truth and compare the `mykrobe` and `drprg` predictions accordingly - for both sequencing technologies.

Figure 4.4 shows the concordance of WGS predictions with culture-based phenotypes. The left panel illustrates the number of correct and missed resistance calls made by `mykrobe` and `drprg` for both Illumina and Nanopore data. In the right panel, we show the number of correct and incorrect susceptibility predictions. Table 4.4 provides the number of FNs and FPs, along with the false-negative rate (FNR), false-positive rate (FPR), positive predictive value (PPV; precision), and negative predictive value (NPV) for each technology/drug combination.

These results provide a positive outcome for both our aims. First, Nanopore predictions (striped bars in Figure 4.4) are highly congruent with Illumina. That is, they have equivalent numbers for each classification category. One exception to this is `mykrobe`-Nanopore having noticeably more FPs for isoniazid compared with its Illumina equivalent. All of the extra Nanopore FP calls were indels, in *katG*, with no Illumina support - any frameshift indel in *katG* is expected to cause resistance to isoniazid [134]. Second, the genome graph-based approach `drprg`, developed in this chapter, provides predictions consistent with `mykrobe` - and in some cases, slightly better.

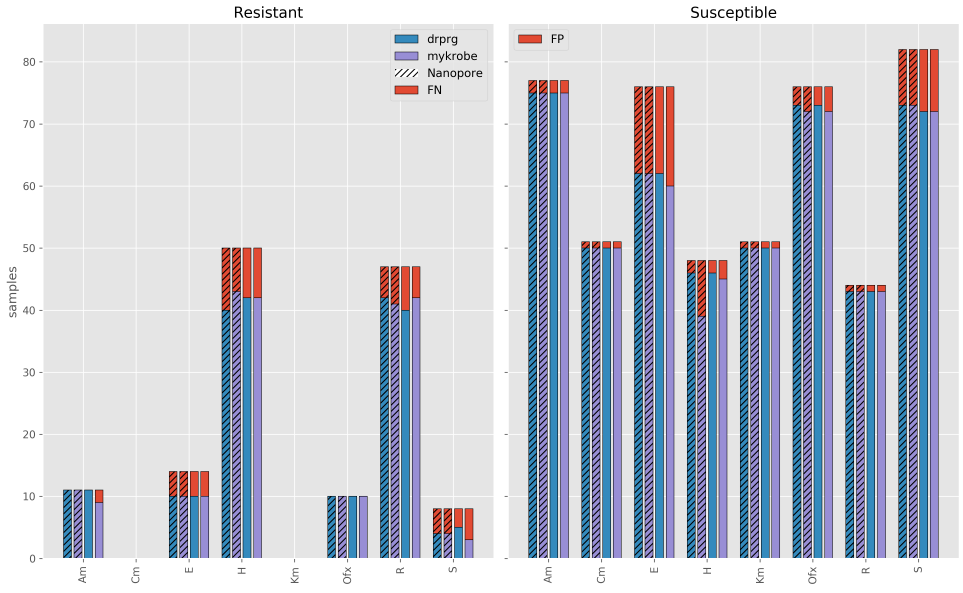


Fig. 4.4: Number of resistant (left) and susceptible (right) culture-based drug susceptibility testing (DST) phenotypes correctly identified by `mykrobe` (purple) and `drprg` (blue) with Illumina (non-striped) and Nanopore (striped) data. The red bars indicate missed (FN) or incorrect (FP) predictions. The x-axis shows the drugs with available phenotype data. E - ethambutol; H - isoniazid; R - rifampicin; S - streptomycin; Km - kanamycin; Am - amikacin; Ofx - ofloxacin; Cm - capreomycin.

The most striking and important result from this analysis is the isoniazid FP discrepancy between `mykrobe` and `drprg` Nanopore predictions, with 9 and 2 FPs, respectively. As isoniazid is one of the most crucial first-line anti-TB drugs, accurate predictions are critical (Section 1.5.2). Six of the `mykrobe`-Nanopore isoniazid FPs were caused by erroneous indel calls, which show no support on Illumina, and `drprg`-Nanopore either had no support or filtered out the call due to poor support. An interesting (but not surprising) observation is that all of these indels occur at homopolymer sites.

Ethambutol and streptomycin had many more FPs and FNs compared to the other drugs for both prediction tools. On further investigation, all ethambutol FPs contain strong evidence for a non-synonymous mutation at codon 306 in *embB*, which is strongly linked to drug resistance [198–200]. Of the 14 samples with FP ethambutol calls (16 for `mykrobe`-Illumina), six additionally had LPA phenotypes. In all cases, the LPA disagreed with the culture-based phenotype (i.e., LPA was resistant and culture was susceptible). In addition, half of the FNs also have LPA phenotypes available, and both of these cases differed from the culture-based phenotype. When taken together, this information would suggest that the culture-based phenotype for ethambutol *may* be wrong in most of these erroneous cases. However, as highlighted

in Section 1.5.2, we stress that while LPA is predictive of resistance, it cannot be used as a means for inferring susceptibility. Therefore, some phenotype-LPA discrepancies may be mutations missing from the LPA catalogue or phenotype minimum inhibitory concentrations (MICs) being close to the cut-off for resistance.

Perhaps the most concerning result from this analysis is the high FNR on isoniazid. However, this is not limited to Nanopore predictions. It is difficult to pinpoint the reasons for missed resistance as there is no way of knowing which mutations we expect to find. There were six samples for which all tool-modality combinations missed resistance. Interestingly, four of these six also had FN calls for rifampicin (accounting for nearly all of the rifampicin FNs). Additionally, 3/10 `drprg`-Nanopore FNs were missed due to (FRS) filtering of the *fabG1* promoter mutation C-15X, a mutation that has been known to cause phenotype discrepancies before [133]. Given the consistent missed resistance across sequencing modalities, the most probable cause of the FNs is an incomplete catalogue - although, we cannot rule out samples swaps. Not all resistance-causing mutations are known, and thus, there will always be limitations to such a panel.

The results from our genome graph-based tool `drprg` are very positive. There were only two drug/technology situations where `drprg` had more errors than `mykrobe`; isoniazid/Nanopore, with `drprg` having 10 FNs and `mykrobe` 7; rifampicin/Illumina, with `drprg` having 7 FNs and `mykrobe` 5. For all other drug/technology combinations, `drprg` has equivalent, or fewer, errors compared to `mykrobe`.

4.5.1 Summary

WGS-based drug resistance predictions from Nanopore are as reliable as those from Illumina when compared to culture-based DST phenotypes. Additionally, we have shown that genome graphs provide reliable predictions for both Illumina and Nanopore data using our tool `drprg`.

While there is a seemingly high number of FNs and FPs for some drugs, many of these errors result from a (`mykrobe`) Nanopore-related indel issues or the absence of known resistance-causing mutations in the panel. Importantly, missed resistance predictions were not technology-driven and presumably occurred because the catalogue lacks the relevant resistance-causing mutations.

Drug	Technology	Tool	FN(R)	FP(S)	FNR(95% CI)	FPR(95% CI)	PPV(95% CI)	NPV(95% CI)
Amikacin	Illumina	`drprg`	0(11)	2(77)	0.0% (0.0-25.9%)	2.6% (0.7-9.0%)	84.6% (57.8-95.7%)	100.0% (95.1-100.0%)
		`mykrobe`	2(11)	2(77)	**18.2% (5.1-47.7%)**	2.6% (0.7-9.0%)	81.8% (52.3-94.9%)	97.4% (91.0-99.3%)
	Nanopore	`drprg`	0(11)	2(77)	0.0% (0.0-25.9%)	2.6% (0.7-9.0%)	84.6% (57.8-95.7%)	100.0% (95.1-100.0%)
		`mykrobe`	0(11)	2(77)	0.0% (0.0-25.9%)	2.6% (0.7-9.0%)	84.6% (57.8-95.7%)	100.0% (95.1-100.0%)
Capreomycin	Illumina	`drprg`	0(0)	1(51)	-	2.0% (0.3-10.3%)	0.0% (0.0-79.3%)	100.0% (92.9-100.0%)
		`mykrobe`	0(0)	1(51)	-	2.0% (0.3-10.3%)	0.0% (0.0-79.3%)	100.0% (92.9-100.0%)
	Nanopore	`drprg`	0(0)	1(51)	-	2.0% (0.3-10.3%)	0.0% (0.0-79.3%)	100.0% (92.9-100.0%)
		`mykrobe`	0(0)	1(51)	-	2.0% (0.3-10.3%)	0.0% (0.0-79.3%)	100.0% (92.9-100.0%)
Ethambutol	Illumina	`drprg`	4(14)	14(76)	28.6% (11.7-54.6%)	**18.4% (11.3-28.6%)**	41.7% (24.5-61.2%)	93.9% (85.4-97.6%)
		`mykrobe`	4(14)	16(76)	28.6% (11.7-54.6%)	21.1% (13.4-31.5%)	38.5% (22.4-57.5%)	93.8% (85.0-97.5%)
	Nanopore	`drprg`	4(14)	14(76)	28.6% (11.7-54.6%)	18.4% (11.3-28.6%)	41.7% (24.5-61.2%)	93.9% (85.4-97.6%)
		`mykrobe`	4(14)	14(76)	28.6% (11.7-54.6%)	18.4% (11.3-28.6%)	41.7% (24.5-61.2%)	93.9% (85.4-97.6%)
Isoniazid	Illumina	`drprg`	8(50)	2(48)	16.0% (8.3-28.5%)	**4.2% (1.2-14.0%)**	95.5% (84.9-98.7%)	85.2% (73.4-92.3%)
		`mykrobe`	8(50)	3(48)	16.0% (8.3-28.5%)	6.2% (2.1-16.8%)	93.3% (82.1-97.7%)	84.9% (72.9-92.1%)
	Nanopore	`drprg`	10(50)	2(48)	**20.0% (11.2-33.0%)**	**4.2% (1.2-14.0%)**	**95.2% (84.2-98.7%)**	82.1% (70.2-90.0%)
		`mykrobe`	7(50)	9(48)	14.0% (7.0-26.2%)	18.8% (10.2-31.9%)	82.7% (70.3-90.6%)	84.8% (71.8-92.4%)
Kanamycin	Illumina	`drprg`	0(0)	1(51)	-	2.0% (0.3-10.3%)	0.0% (0.0-79.3%)	100.0% (92.9-100.0%)
		`mykrobe`	0(0)	1(51)	-	2.0% (0.3-10.3%)	0.0% (0.0-79.3%)	100.0% (92.9-100.0%)
	Nanopore	`drprg`	0(0)	1(51)	-	2.0% (0.3-10.3%)	0.0% (0.0-79.3%)	100.0% (92.9-100.0%)
		`mykrobe`	0(0)	1(51)	-	2.0% (0.3-10.3%)	0.0% (0.0-79.3%)	100.0% (92.9-100.0%)
Ofloxacin	Illumina	`drprg`	0(10)	3(76)	0.0% (-0.0-27.8%)	3.9% (1.4-11.0%)	76.9% (49.7-91.8%)	100.0% (95.0-100.0%)
		`mykrobe`	0(10)	4(76)	0.0% (-0.0-27.8%)	5.3% (2.1-12.8%)	71.4% (45.4-88.3%)	100.0% (94.9-100.0%)
	Nanopore	`drprg`	0(10)	3(76)	0.0% (-0.0-27.8%)	3.9% (1.4-11.0%)	76.9% (49.7-91.8%)	100.0% (95.0-100.0%)
		`mykrobe`	0(10)	4(76)	0.0% (-0.0-27.8%)	5.3% (2.1-12.8%)	71.4% (45.4-88.3%)	100.0% (94.9-100.0%)
Rifampicin	Illumina	`drprg`	7(47)	1(44)	14.9% (7.4-27.7%)	2.3% (0.4-11.8%)	97.6% (87.4-99.6%)	86.0% (73.8-93.0%)
		`mykrobe`	5(47)	1(44)	10.6% (4.6-22.6%)	2.3% (0.4-11.8%)	97.7% (87.9-99.6%)	89.6% (77.8-95.5%)
	Nanopore	`drprg`	5(47)	1(44)	10.6% (4.6-22.6%)	2.3% (0.4-11.8%)	97.7% (87.9-99.6%)	89.6% (77.8-95.5%)
		`mykrobe`	6(47)	1(44)	12.8% (6.0-25.2%)	2.3% (0.4-11.8%)	97.6% (87.7-99.6%)	87.8% (75.8-94.3%)
Streptomycin	Illumina	`drprg`	3(8)	10(82)	37.5% (13.7-69.4%)	12.2% (6.8-21.0%)	33.3% (15.2-58.3%)	96.0% (88.9-98.6%)
		`mykrobe`	5(8)	10(82)	**62.5% (30.6-86.3%)**	12.2% (6.8-21.0%)	23.1% (8.2-50.3%)	93.5% (85.7-97.2%)
	Nanopore	`drprg`	4(8)	9(82)	50.0% (21.5-78.5%)	11.0% (5.9-19.6%)	30.8% (12.7-57.6%)	94.8% (87.4-98.0%)
		`mykrobe`	4(8)	9(82)	50.0% (21.5-78.5%)	11.0% (5.9-19.6%)	30.8% (12.7-57.6%)	94.8% (87.4-98.0%)

Table 4.4: Comparison of WGS-based drug resistance prediction concordance with culture-based drug susceptibility testing (DST) phenotypes. For this comparison, we assume the DST phenotype is correct and evaluate the WGS predictions accordingly. `drprg` and `mykrobe` are the two tools that provide WGS predictions. Bold text is used to highlight differences of note between `drprg` and `mykrobe`. FN=false negative; R=number of resistant samples; FP=false positive; S=number of susceptible samples; FNR=false negative rate; FPR=false positive rate; PPV=positive predictive value; NPV=negative predictive value; CI=Wilson score confidence interval

4.6 Effect of Nanopore read depth

An important consideration when performing Nanopore sequencing for AMR prediction is how much data is needed. The quantity of data required has implications for how long the Nanopore sequencing device needs to be run or how many samples can be multiplexed in a single run in order to yield sufficient data for reliable predictions. A previous study by Votintseva *et al.* found that deep coverage is required of Nanopore to predict drug resistance accurately[95]. As Nanopore sequencing advances quickly, and our dataset contains a broad Nanopore depth-of-coverage range (29-150x), we explore whether this requirement of high coverage still holds.

We binned samples into groups based on their read depth. The groups were segregated into lots of 10x depth. That is, if a sample has a read depth of 56, it is assigned to the 50x bin, which covers samples with read depth less than 60, but ≥ 50. We calculate the proportions of each drug resistance classification (FN, TP, TN, FP) present for each bin. For this analysis, we use the culture-based phenotype classifications from Section 4.5. If low read depth leads to poor resistance prediction, we would expect the proportions of FPs and FNs in the low-depth bins to be greater than in the high-depth bins.

As Figure 4.5 illustrates, we see no relationship between Nanopore read depth and erroneous predictions. Further work at depths lower than 30x is warranted, but this is convincing evidence that samples with "only" 30x Nanopore read depth still yield reliable resistance predictions.

4.7 Detecting off-catalogue mutations

One of the main advantages `drprg` has over `mykrobe` is the ability to discover off-catalogue (novel) variants. The reason `drprg` can call mutations outside of the panel is that it uses `pandora` as the underlying method for producing predictions from sequencing reads.

The CRyPTIC Consortium have shown that refusing to make predictions in the case where a non-panel variant is discovered in a resistance-associated gene can improve pan-susceptibility prediction by reducing missed resistance (FN) calls [133]. However, that work focused only on first-line drugs and did not detail the variants discovered.

In this section, we aim to assess the novel variant detection capacity of `drprg` on both Illumina and Nanopore data. We have a dataset with well-characterised SNPs

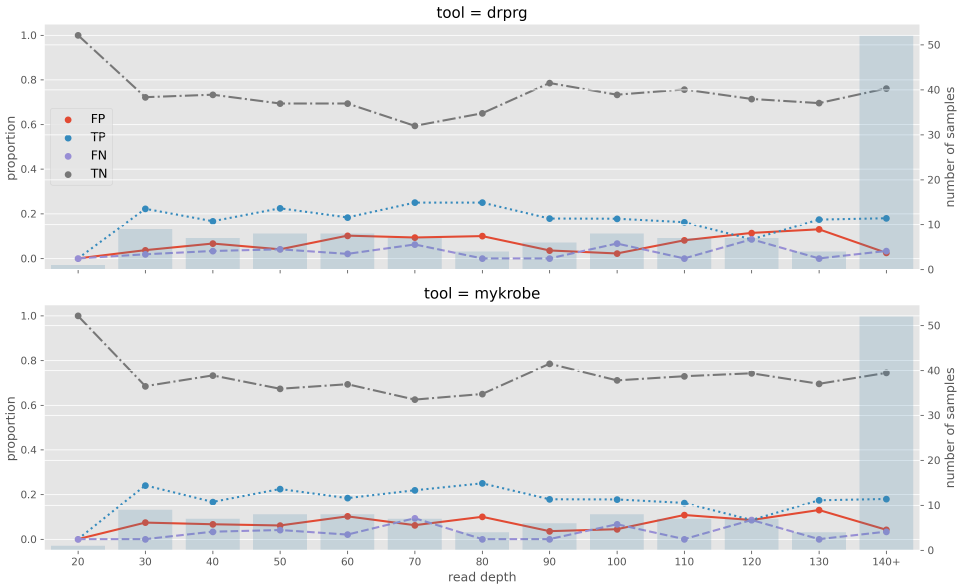


Fig. 4.5: Effect of Nanopore read depth on `drprg` (top) and `mykrobe` (bottom) drug susceptibility phenotype prediction. Each point indicates the proportion (left y-axis) of classifications of that type at the read depth (x-axis). Read depth is binned such that 40 is all samples with a read depth ≥ 40 and less than 50. The blue bars indicate the number of samples (right y-axis) contained in each bin. FP=false positive (red); TN=true negative (grey); FN=false negative (purple); TP=true positive (blue).

from Chapter 2, and these high-quality variant calls give us a way of determining how many SNPs `drprg` finds and misses in resistance-associated genes.

We remove any position in the COMPASS SNPs VCF from Section 3.6.2 that falls into one of three categories: i) does not occur inside any mutation catalogue gene; ii) does not call an alternate allele; or iii) exists in the panel. After this filtering, we have a VCF containing SNPs in resistance-associated genes that do not occur in the panel - referred to as the truth VCF.

We classify the novel SNPs called by `drprg` against the truth VCF using `hap.py` - a VCF comparison program developed by Illumina and the Global Alliance for Genomics and Health Benchmarking Team [201]. Figure 4.6 and Table 4.5 show the number of TP, FP, and FN novel SNP calls made by `drprg` for each drug across all 150 samples.

From Figure 4.6 we clearly see *rpoB* contains the most off-panel SNPs. Of particular note is the large percentage of *gyrA* FNs, which leads to a recall of 0.35 for both Illumina and Nanopore in this gene. Upon further investigation, all of these missed calls were filtered out by `drprg` due to low FRS. Additionally, we see the same scenario when looking at the *ahpC* FNs - which are much higher in Nanopore data.

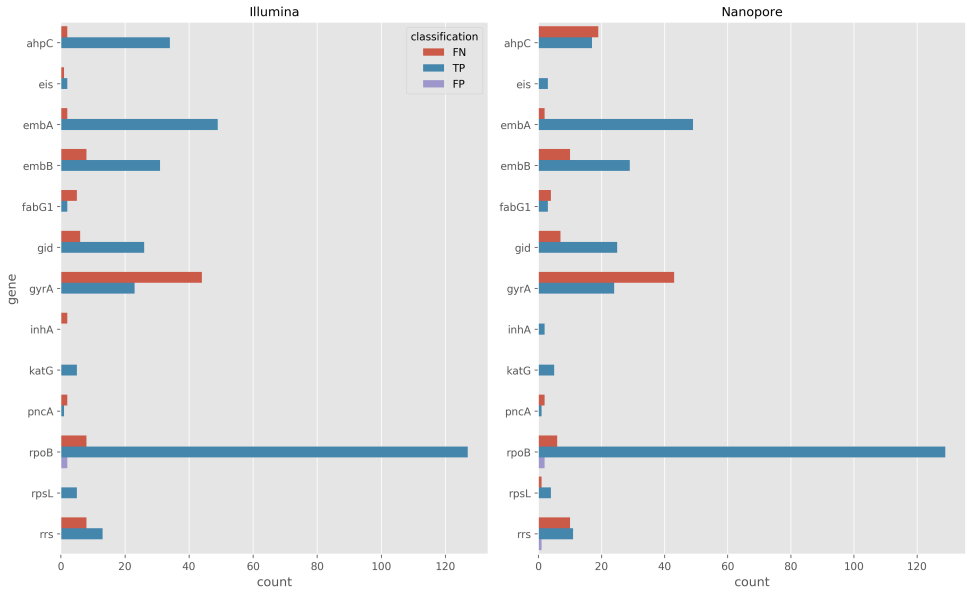


Fig. 4.6: Classifications of novel SNP calls made from Illumina (left) and Nanopore (right) data by `drprg` in resistance-associated genes (y-axis). Counts (x-axis) are across all samples. TP (blue) - true positive; FN (red) - false negative; FP (purple) - false positive.

Despite the poor recall in *gyrA* and *ahpC*, the precision of novel `drprg` calls is encouraging. In total, there were two FPs detected in *rpoB* (for both technologies), and one Nanopore FP in *rrs*. This result suggests we can be very confident in the novel SNP calls made by `drprg`.

Figure 4.7 illustrates the number of novel calls of different classification on a per-sample basis. It shows that on average, a samples will have 2 novel TP calls and 0 FPs and FNs.

Having seen that the off-panel SNPs called by `drprg` are precise, we look at how incorporating this information impacts the AMR predictions. Figure 4.8 is the same as Figure 4.4, except with unknown (off-panel) calls included. If we find a novel variant, an unknown prediction is given for the drug(s) associated with that gene. The exception to this is when a known resistance-causing mutation is also found for the respective drug(s), in which case a resistant prediction is made.

Of note in Figure 4.8 is the reduced number of missed resistance calls made when we include unknown variants - when compared to Figure 4.4. That is, a call of unknown is our way of indicating to the user that we did not find a *known* resistance-causing mutation; however, we did find a mutation that is not known to be either resistance- or susceptibility-associated.

Gene	Technology	FN	FP	TP	Recall	Precision
ahpC	Illumina	2	0	34	0.944	1.000
	Nanopore	19	0	17	0.472	1.000
eis	Illumina	1	0	2	0.667	1.000
	Nanopore	0	0	3	1.000	1.000
embA	Illumina	2	0	49	0.961	1.000
	Nanopore	2	0	49	0.961	1.000
embB	Illumina	8	0	31	0.795	1.000
	Nanopore	10	0	29	0.744	1.000
fabG1	Illumina	5	0	2	0.286	1.000
	Nanopore	4	0	3	0.429	1.000
gid	Illumina	6	0	26	0.812	1.000
	Nanopore	7	0	25	0.781	1.000
gyrA	Illumina	44	0	23	0.343	1.000
	Nanopore	43	0	24	0.358	1.000
inhA	Illumina	2	0	0	0.000	-
	Nanopore	0	0	2	1.000	1.000
katG	Illumina	0	0	5	1.000	1.000
	Nanopore	0	0	5	1.000	1.000
pncA	Illumina	2	0	1	0.333	1.000
	Nanopore	2	0	1	0.333	1.000
rpoB	Illumina	8	2	127	0.941	0.984
	Nanopore	6	2	129	0.956	0.985
rpsL	Illumina	0	0	5	1.000	1.000
	Nanopore	1	0	4	0.800	1.000
rrs	Illumina	8	0	13	0.619	1.000
	Nanopore	10	1	11	0.524	0.917

Table 4.5: Classification of novel SNP calls made by `drprg` in resistance-associated genes. Counts are across all samples. TP - true positive; FN - false negative; FP - false positive.

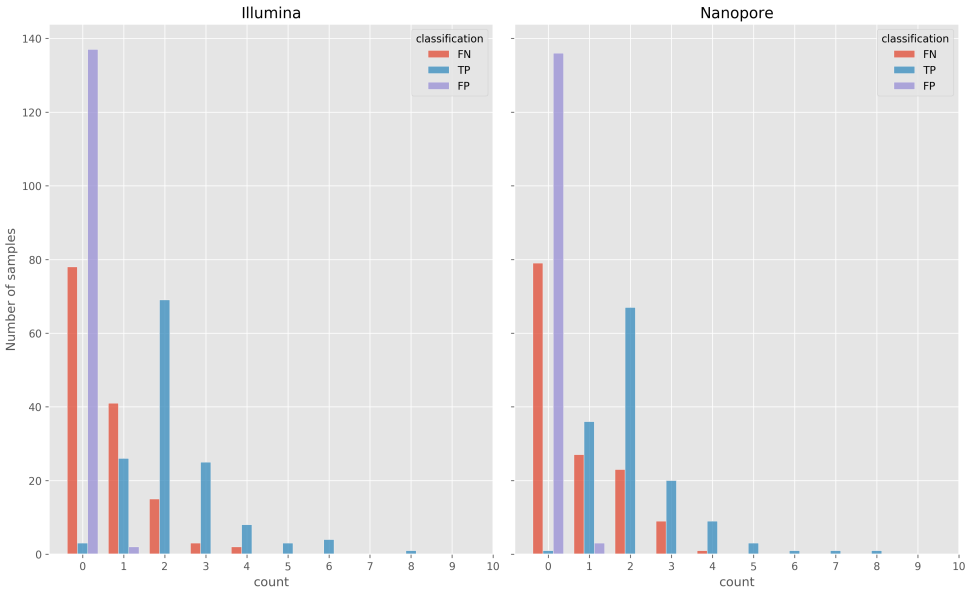


Fig. 4.7: Number of novel variants found in resistance-associated genes for each sample. The x-axis indicates the number of classifications of the relevant kind for a sample, and the y-axis is the number of samples with that classification count. The classifications are: TP (blue) - true positive; FN (red) - false negative; FP (purple) - false positive.

While we see a reduction in the number of missed resistance calls (FN) when calling unknown variants, the number of definitive susceptibility (TN) calls is also reduced. However, these are not "missed" susceptibility calls and would just require phenotyping more samples (a situation preferred by many clinicians in response to [45]). In particular, ethambutol, rifampicin, and streptomycin see a large number of unknown calls. A large number of unknown calls for rifampicin is not unexpected as *rpoB* (associated with rifampicin resistance) had by far the most novel variants in the above analysis (Figure 4.6). Nearly every one of these unknown rifampicin calls relates to synonymous mutations D103D (C309T) and A1075A (T3225C) that are known not to be associated with drug resistance [202]. Likewise, synonymous mutations were also the cause of the bulk of unknown calls for other drugs. However, there were some indel calls found. As our truth VCF only contains SNPs, we cannot assess the validity of these indel calls.

A simple solution for the excessive unknown calls in susceptible samples would include synonymous mutations in the panel as susceptibility-associated.

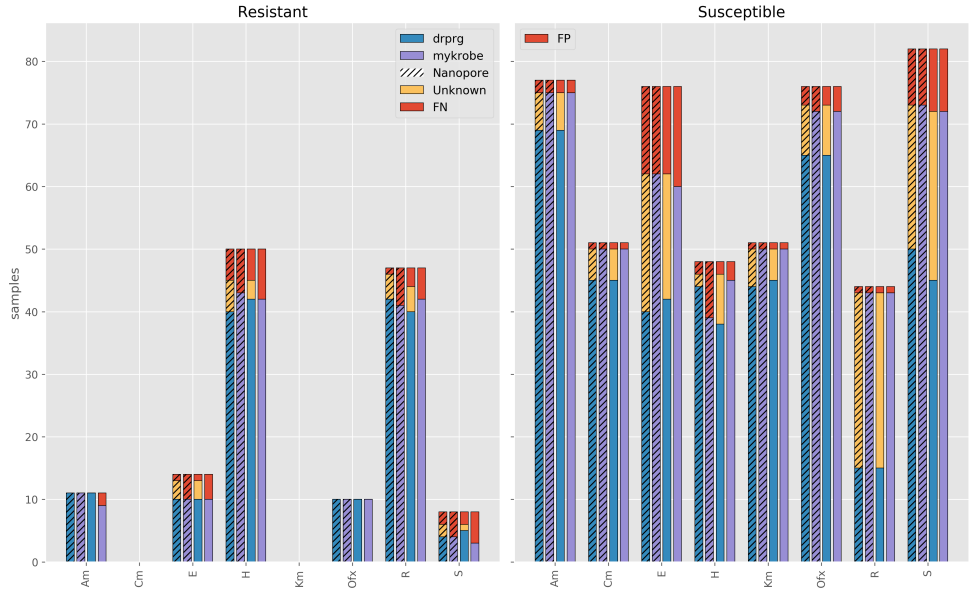


Fig. 4.8: Number of resistant (left) and susceptible (right) culture-based drug susceptibility testing (DST) phenotypes correctly identified by `mykrobe` (purple) and `drprg` (blue) with Illumina (non-striped) and Nanopore (striped) data. The red bars indicate missed (FN) or incorrect (FP) predictions. The yellow bars represent `drprg` "unknown" calls, which is when there is an off-panel variant found in a gene associated with the respective drug. If there is no resistance-associated variant found for that drug, then an unknown call is made. The x-axis shows the drugs with available phenotype data. E - ethambutol; H - isoniazid; R - rifampicin; S - streptomycin; Km - kanamycin; Am - amikacin; Ofx - ofloxacin; Cm - capreomycin.

4.7.1 Summary

Producing unknown drug resistance predictions based on novel variant calls is a point of difference for `drprg`. In this section, we have shown that novel variants detected by `drprg` are precise, but that missed calls require some more work.

We have also shown that we can reduce missed resistance calls by refusing to make predictions for drugs with variants of unknown consequences. However, this currently comes at the cost of fewer susceptibility calls, which means sending more samples for phenotyping, but not missing susceptibility. Although, an easy fix for many of these susceptible samples sent for phenotyping - incorporating synonymous mutations - has been identified.

4.8 Discussion

Illumina WGS has become a standard tool for *M. tuberculosis* drug susceptibility testing. Its use is underpinned by a vast number of studies validating the sensitivity and specificity of such predictions, all with similar results [44, 45, 132, 133, 135, 136, 203]. In this chapter, we have provided evidence that Nanopore WGS-based DST can provide predictions consistent with Illumina. We did this by showing that when comparing Nanopore-predicted phenotypes to those from Illumina-predicted and culture-based phenotypes, there is very little difference between the predictions from the two technologies. However, for isoniazid, we do note a significantly higher Nanopore false positive rate (18.8%) than Illumina (6.2%) when using `mykrobe` - but we offer a solution to this below.

In addition to validating the use of Nanopore for DST, we also developed and evaluated a software tool `drprg` for the same task. Many programs exist to predict drug resistance from WGS data; however, all have the same underlying limitation: an inability to detect off-panel (novel) mutations. With `drprg` we have made a first step in removing this limitation from *M. tuberculosis* AMR prediction.

`drprg` is not the first tool to use the concept of genome graphs for AMR prediction. `mykrobe`, the other tool used in this chapter, uses population genome graphs for genotyping of samples. The underlying method `mykrobe` uses is Cortex [43] - a program that uses coloured de Bruijn graphs for genotyping samples via *de novo* assembly. Cortex is somewhat of a precursor to `pandora` - the genome graph method underpinning `drprg`. However, `pandora` offers a number of advantages over Cortex (see Section 1.5.2 for a full description of these). Ultimately, the differences mean,

theoretically, `pandora` (`drprg`) requires less read depth to produce the same information as Cortex (`mykrobe`) and is much better suited to use with noisy, error-prone Nanopore data.

When comparing the computational performance of `mykrobe` and `drprg`, we found `drprg` to be faster and more memory efficient for both technologies (Section 4.3.3).

Given the extensive validation of Illumina WGS for DST, with many more samples than we have in this chapter, our intention is not to assert the clinical use of WGS. Instead, we seek to gauge Nanopore's suitability for WGS-based DST.

As `mykrobe` has been extensively validated on large datasets with Illumina [44, 45], using it as a truth to compare Nanopore to provides evidence for Nanopore's use for AMR prediction. We performed this analysis in Section 4.4 and found `mykrobe`'s Nanopore predictions to be highly concordant with those from Illumina. There were only eight missed resistance calls, seven of which were minor resistance calls on Illumina. We attempted to allow minor resistance calls for Nanopore, but this led to every sample being predicted as resistant to isoniazid and pyrazinamide due to false heterozygous indel calls in *katG* and *pncA*. Of the false-positive calls made by `mykrobe` on Nanopore data, half were due to indel calls in *katG*, while the other half are quite likely (SNP) errors on behalf of Illumina due to filtering.

When comparing the `drprg` predictions to `mykrobe` Illumina profiles, we see somewhat similar results. As with `mykrobe`-Nanopore errors, almost all `drprg`-Nanopore errors were due to indel calls in *katG* and *pncA*, while the rest were most likely `mykrobe`-Illumina errors, as mentioned above.

We next assessed Illumina and Nanopore predictions against "gold-standard" culture-based phenotypic testing in Section 4.5. This analysis allows an unbiased assessment of the two sequencing modalities, intending to detect any technology-driven differences. As all our samples were sequenced from the same DNA extraction, we know the resistance profile for the underlying data is the same for both technologies.

Nanopore has been assessed for use in providing AMR predictions previously [44, 45, 136], but with single-digit sample sizes. An exception to this is the work by the New York State Department of Health, which used a sample size of 431 [91]. Our results from Section 4.5 are in agreement with all of these previous studies; Nanopore AMR predictions are highly concordant with Illumina when compared to culture-based phenotypes - using our altered parameters from Section C.4. This agreement is found for both `mykrobe` and `drprg`. As mentioned, the one exception to this was `mykrobe`'s

Nanopore isoniazid predictions, which had a higher false-positive rate compared to the other technology/tool combinations. All of these extra FPs were due to *katG* indel calls in the Nanopore data, which as mentioned, is a systematic problem currently affecting Nanopore [79].

While there is a high level of similarity between predictions from both tools and sequencing modalities, disagreement with the culture-based phenotype is high for some drugs. For example, isoniazid FNR of 14-20% is higher than previous work [45, 133]. In almost all of these (FN) errors, we see no strong evidence for mutations in our catalogue. These discrepancies raise the ongoing need for a curated panel of known mutations and their impacts on susceptibility.

The ethambutol false negative rate (FNR; 28.6%) was much higher than previous reports, which generally fall in the 5-10% range [45, 91, 133]. However, Smith *et al.* did see an FNR of 20%. In addition, the positive predictive value (PPV; also known as precision) was 41.7%, again, much lower than previous studies. However, all FP predictions were made due to a non-synonymous mutation at codon 306 in *embB*. We saw consistent support for the presence of these FP mutations on all technologies and tools, and multiple previous reports have strongly linked this mutation to ethambutol resistance [198–200]. Additionally, phenotyping of isolates with this particular mutation is known to be unreliable [132, 204, 205], raising the possibility of a spurious culture-based phenotype in these cases. Indeed, sequencing of the resistance-associated genes is now recommended over culture-based phenotyping for ethambutol [206].

The main limitation for this section of the chapter is the number of resistant samples. Although there was 50 isoniazid- and 47 rifampicin-resistant isolates, the other drugs had no more than 15 samples representing. The low number of resistant samples is reflected in the breadth of confidence intervals for all drugs. However, combining this dataset with Smith *et al.* [91] would provide an increase in the matched Illumina/Nanopore data with susceptibility profiles for all drugs.

Taking the analyses in Section 4.4 and Section 4.5 together, it is clear that one of the main limitations of `drprg` is faulty indel calls. Indeed, on many occasions in this thesis, we have stated this is a known systematic issue of Nanopore [79]. However, we believe this is a problem worth careful consideration as it has a big impact on isoniazid and pyrazinamide resistance - both first-line *M. tuberculosis* drugs. We discuss future directions for improving on this limitation of `drprg` in Section 4.10.4.

Having established Nanopore-based AMR predictions for *M. tuberculosis* are comparable to Illumina, we next looked at whether the Nanopore performance is dependent

on the read depth available. In 2017, Votintseva *et al.* concluded that the accuracy of Nanopore for providing real-time AMR predictions for *M. tuberculosis* was dependent on "deep coverage" [95]. While four years may not seem like a long time in science, in the world of Nanopore sequencing, it is an age. Technological advances in Nanopore sequencing happen so quickly that regular benchmarks are warranted (but tedious). Given that our dataset has samples with read depth ranging from 29-150x, in Section 4.6 we assessed whether there was a higher proportion of misclassifications at lower depths. As Figure 4.5 very clearly shows, there is no evidence in our dataset that read depth has any influence on AMR predictions. To our knowledge, this is the first time such an analysis correlating read depth and classifications has been presented. The consequences of this finding are quite important. Being able to obtain reliable predictions from low-coverage data saves time and money by not having to re-sequence low-yield Nanopore runs, something which is common when multiplexing samples on the one flowcell.

Although there are many *M. tuberculosis* AMR prediction tools available, `drprg` is the first to recognise novel variants. When an off-panel variant is found in a gene associated with drug resistance, the prediction for the drug(s) it impacts is designated "unknown". While this is not the first study to investigate the impact of unknown variants on predictions [45, 133], it is the first to provide the functionality to do so in a single program.

We used the Illumina variant calls from COMPASS in Section 3.6.2 to curate a list of off-panel SNPs we expect to find for each sample. As would be expected from the `pandora` validation in Section 3.6.4, `drprg` produces very precise novel SNP calls, but has some room for improvement for SNP discovery (Figure 4.6). Many of the novel SNPs were synonymous mutations, highlighting a gap in the panel construction step of `drprg` (Section 4.3.1) - where we do not incorporate such mutations unless they are explicitly provided. In addition, the higher number of FN novel SNPs in *ahpC* and *gyrA* were caused by `drprg` filtering them out for low fraction of read support. Given that these SNPs did not show signs of heterozygosity in the COMPASS VCFs, this warrants further investigation into why were are seeing considerable read depth on multiple alleles at these sites.

When unknown predictions for a drug are interpreted as a refusal to make a prediction - as in [133] and [45] - we see a noticeable decrease in the number of missed resistance calls for ethambutol, isoniazid, rifampicin, and streptomycin. That is, rather than erroneously calling those samples susceptible, we raise the presence of an unknown mutation, indicating further testing is warranted. On the other hand, the

same approach classifies some susceptible samples as unknown. In particular, the same drugs mentioned above would have a sizeable proportion of susceptible isolates called unknown. However, as mentioned in Section 4.7, this would just require phenotyping more samples (a situation preferred by many clinicians in response to [45]). Although, as mentioned, many of these unknown predictions are synonymous mutations, so these unnecessary unknowns are likely easy to fix.

A limitation to our novel variant analysis was the absence of novel indel call validation. Indels are an especially important determinant of resistance for pyrazinamide and isoniazid, so future work should focus on generating a high-quality truth set for the evaluation of `drprg` indels.

4.9 Conclusion

In conclusion, the work in this chapter provides validation that WGS-based drug resistance predictions from Nanopore data are mostly consistent with those from Illumina. However, further work is required for isoniazid, where we see a higher FPR than Illumina when using `mykrobe`.

Given the extensive validation of Illumina for *M. tuberculosis* AMR prediction in clinical settings, the congruence we present suggests that Nanopore can also be used in the same setting. However, further validation on a more extensive dataset is warranted, especially for isoniazid.

We also describe and evaluate a new method for AMR prediction using genome graphs - `drprg`. We showed that predictions from `drprg` are mostly consistent with those from `mykrobe` whilst being faster and more memory frugal. However, further work is required to improve `drprg`'s isoniazid FNR on Nanopore data, and both FNR and FPR for pyrazinamide for both sequencing modalities.

Importantly, `drprg` is the first *M. tuberculosis* AMR prediction tool with the option to return an unknown prediction in cases where novel variants are found in resistance-associated genes. Furthermore, the novel variants discovered by `drprg` are precise and lead to a reduced number of missed resistance calls.

4.10 Future work

4.10.1 Mutation catalogue improvement

The panel (catalogue) describing the consequence of each mutation is a vital component of any AMR prediction program and can be the primary point-of-difference between tools [45]. For the work in this chapter, we used the default `mykrobe` panel for `drprg`. However, there are two changes to the `drprg` panel that we would like to incorporate in future work: synonymous mutations and including variants from a new World Health Organization (WHO) catalogue.

Synonymous mutations

As we saw in Section 4.7, many of the off-catalogue mutations signalled as unknown predictions were synonymous mutations - i.e., they lead to no amino acid change. To our knowledge, there has been only one synonymous mutation linked to *M. tuberculosis* drug resistance [207]. As such, adding the ability for `drprg` to inherently allow such mutations, without raising them as unknown, would lead to dramatically less unknown predictions - especially in *rpoB* (rifampicin).

WHO panel

The WHO has recently released a curated catalogue of *M. tuberculosis* mutations and their associated impact on drug susceptibility [141]. Incorporating such a panel into the one used in this chapter would likely improve the ability to detect resistance. Indeed, the inclusion of known susceptibility-associated mutations into the `drprg` helped reduce the novel classifications.

As an example, in Section 4.5, we saw a streptomycin FN classification, where `drprg` had called an unknown mutation. This unknown mutation represents the amino acid change K88M in *rpsL*. However, this mutation is listed as resistance-associated in the WHO catalogue and has been linked to streptomycin resistance [208].

The apparent first improvement would be to incorporate all WHO variants associated with resistance that are not already in our panel. Further work could look to include the mutations listed as "uncertain significance" in a meaningful way.

4.10.2 Low read depth samples

In Section 4.6 we showed that read depth has no noticeable impact on AMR predictions. However, the lowest depth sample in this chapter has 29x read depth. This depth threshold was a conscious decision made in Section 3.4, where we excluded any sample with less than approximately 30x read depth from further analysis. Thus, there are 33 samples in our full dataset that have Nanopore read depth between 5 and 30x. For those with culture-based phenotype information, we could repeat the analysis in Section 4.6. We would then have an even more fine-grained idea of where the Nanopore read depth limits are for reliable AMR predictions. Additionally, we could subsample those isolates with phenotype data and $\geq$ 30x depth to select low depths.

4.10.3 Validation on larger datasets

Smith *et al.* recently published a dataset of 431 *M. tuberculosis* isolates with matched Nanopore and Illumina sequencing [91]. As they are from the New York State Department of Health, these samples also have gold-standard DST information for eight drugs we provide predictions. Importantly, they have phenotype information for pyrazinamide, one drug we did not have sufficient data for in this chapter. Combining this dataset with ours would give us a very good Nanopore validation set and provide improved confidence in the false-positive and -negative rates.

In addition to improving the validation numbers for Nanopore, we could also provide vastly improved validation for `drprg` Illumina predictions by running on previous large cohort studies such as [45, 133, 136]. Given that we know Nanopore and Illumina predictions from `drprg` are consistent, validating Illumina on such large and diverse datasets would be extremely informative overall.

4.10.4 Indel calls

Indels are important variants for *M. tuberculosis* drug resistance. In particular, any frameshift in *katG* or *pncA* leads to isoniazid or pyrazinamide resistance, respectively [134]. Perhaps the limitation we rue the most from Chapter 3, and this chapter, is the lack of in-depth indel analysis. These mutation types have appeared as error causes for several tools and drugs in this chapter. In particular, as we saw in Section 4.4, `mykrobe`-Nanopore has a high FPR for isoniazid, due to erroneous indel calls. As isoniazid is an important first-line drug, understanding how these FPs can be avoided is vital. Thus, indel-calling must be assessed in future work.

To properly evaluate and improve indel calls, we need a trustworthy set of expected variants. In Section 4.7 we only investigated novel SNP calls made by `drprg`, as we already had a reliable truth set from Chapter 3. Our proposed method for gathering such a truth panel of indels would be to run Clockwork (https://github.com/iqbal-lab-org/clockwork; [175]) on all samples. Clockwork calls variants with `samtools` and Cortex and integrates them for a high-quality, filtered set of SNPs and indels.

Nanopore's ability to produce reliable indel calls is still poor; recent benchmarks show F1-scores around 50% [82]. The overwhelming cause of indel errors from Nanopore is around sites with homopolymer deletions. In Chapter 5 we will attempt to mitigate these errors by training a *M. tuberculosis*-specific Nanopore basecalling model.

4.10.5 Lineage classification

One noticeable feature lacking from `drprg`, when compared to other tools such as `mykrobe` and TB-Profiler [136], is the ability to identify the lineage of a sample. This functionality is easily added to `drprg` by building a PanRG containing only small regions around lineage-defining variants [116, 167–169]. Such a PanRG would only need the wild-type allele and an alternate allele that defines a given lineage. Identifying the lineage would be as simple as matching the genotypes across these sites to the lineage from a lookup table.

4.10.6 Other species

While the initial development of `drprg` has focused solely on *M. tuberculosis*, there is no reason it should be exclusive to that species. All that is needed to adapt it for another species is a panel of variants, a reference genome and annotation. Or, as was done in Section 4.3.1, a prebuilt PanRG from species representatives. `mykrobe` also provides inbuilt panels for *S. aureus* and *Shigella sonnei*, so beginning with these species and validating on the same data would be the logical first step.

The main challenge when adapting to other species is how to allow gene presence/absence detection, which is a mechanism of resistance detected by `mykrobe` for *S. aureus* [44]. As `pandora` outputs a consensus sequence for each *present* locus, all that would be required is investigating incorrect gene presence/absence and determining if default parameters need to be altered.

Another potential challenge is that we implicitly assume the reference allele is susceptible. For example, some bacteria, such as Gram-negatives, can have intrinsic AMR [209]. Therefore, designing a way for this information to be provided to `drprg` is necessary for adapting `drprg` to some species.

4.11 Availability of data and materials

`drprg` is open-source and freely available under an MIT license at https://github.com/mbhall88/drprg. The pipelines and scripts used in this chapter are available at https://github.com/mbhall88/head_to_head_pipeline/tree/master/analysis/resistance_prediction. All analyses were run using the workflow management program `snakemake` [192]. All figures were generated using the Python libraries `matplotlib` [193] and `seaborn` [194] - Figure 4.1 was generated using https://github.com/jnothman/UpSetPlot.

Chapter 5

Improving Nanopore sequencing accuracy for *M. tuberculosis*

5.0 Publication and collaboration acknowledgements

In addition to the sequencing in Section 3.0, the work *not* completed by myself in this chapter was the sequencing of the BCG sample.

The Nanopore sequencing of the BCG sample was performed by Sophie George at the Nuffield Department of Clinical Medicine, John Radcliffe Hospital, Oxford University.

5.1 Introduction

The accuracy of Nanopore sequencing data has been steadily increasing since its inception. For example, read accuracy has progressed from 65% in June 2014 [149] to 89.17% in December 2018 [78] - an extraordinary 37% increase in the space of 4.5 years.

While there have been many successful attempts by the research community to develop new methods for basecalling Nanopore data [76, 77, 210], most cannot keep pace with the speed of new Nanopore developments. As such, `guppy`, the basecalling software provided by Oxford Nanopore Technologies (ONT), is the most commonly used method.

Taxon-specific Nanopore basecalling models have been shown to provide even further increased accuracy. For instance, Wick *et al.* trained a *Klebsiella pneumoniae*-

specific basecalling model [78], showing their custom model improves the accuracy of Nanopore for that species. Importantly, they also found it reduced systematic errors around homopolymer deletions and methylation sites. However, it remains to be seen if this approach achieves similar results for other species.

In both Chapter 3 and Chapter 4 of this thesis, we use Nanopore data from *M. tuberculosis* and were somewhat limited by systematic errors (see Section 4.10.4). Therefore, in this chapter, we set out to train an *M. tuberculosis*-specific Nanopore basecalling model using the unique dataset we collected in Section 3.2. The eight samples with sequencing data from Illumina, PacBio, and Nanopore provide the perfect training input for such a model as they have high-quality assemblies (Section B.2) free from Nanopore biases, but with available Nanopore reads.

We show that our *M. tuberculosis* Nanopore model yields increased read- and consensus-level accuracy and reduced homopolymer deletions. Additionally, we demonstrate the generalisability of this model to another member of the *M. tuberculosis* complex (MTBC; see Section 1.5) - an *M. bovis* BCG strain.

By making this *M. tuberculosis*-specific model available to all, we hope the application of Nanopore sequencing to *M. tuberculosis* - a major theme of this thesis - will increasingly provide novel insights.

5.2 Dataset

Perhaps the most critical aspect of training a basecalling model is providing a "truth" for the data. In this context, truth data refers to high-quality genome assemblies for the training samples. The ONT-maintained basecaller, guppy, uses neural networks to convert the raw electrical signal into a DNA sequence. In order to train the network to make this inference, it is necessary to label the raw signal with its corresponding truth sequence. Unfortunately, such datasets are difficult to find for certain species. However, the dataset we have collected for the work in Chapter 3 and Chapter 4 is ideally suited. It contains samples with Illumina, PacBio, and Nanopore sequencing data from the same DNA extraction, ensuring any discrepancies between the Nanopore data and the truth are technology differences, and not *in vitro* evolution.

Hence, for the training and validation of our *M. tuberculosis*-specific model, we use the eight samples we generated high-quality PacBio assemblies for in Section 3.3 (see Section B.2 for full methods). We use the PacBio assemblies produced by `flye` and polished with Illumina by Pilon - not correcting for SNPs as the PacBio CCS SNP

error rate is much lower than Illumina's [157]. By using the Illumina-polished PacBio assemblies for each sample, we ensure no Nanopore biases are present in the truth genomes from which the custom model is trained.

In addition to the eight training and validation samples, we evaluate on a Nanopore sequencing run of the *Mycobacterium bovis* strain used in the Bacille Calmette-Guérin (BCG) vaccine. This strain (AF2122/97) is an attenuated *M. bovis* bacillus [211] with a well-characterised reference genome (accession LT708304.1) [212]. As the genome similarity between *M. bovis* and *M. tuberculosis* is 99.95% [213], this BCG strain acts as a great test for the model's performance on an independent sample from a different but very closely related species.

The Nanopore sequencing of this BCG sample was performed by Sophie George at the Nuffield Department of Clinical Medicine, John Radcliffe Hospital, Oxford University. The sample preparation was performed according to [214], except, rather than being spiked in and sequenced direct from human sputum, this sample was direct from Mycobacteria Growth Indicator Tube (MGIT) culture on a single sample flow cell.

5.3 Training an *M. tuberculosis*-specific Nanopore basecalling model

In order to train a basecalling model for use with `guppy`, there are numerous preparation steps required. For many of these, we use the open-source software, Taiyaki, developed by ONT to train their RNA and DNA models for `guppy` (https://github.com/nanoporetech/taiyaki).

We trained an *M. tuberculosis*-specific basecalling model - named `tubby` - for use with three different versions of `guppy`: 3.4.5, 3.6.0, and 4.4.0. The preparation, training, and evaluation steps are the same for each version, with the only difference being the version of `guppy` used to basecall the initial data and the pre-trained models from which to begin training.

Preparation

In the first stage of preparing the data for training, we basecalled and demultiplexed the data with the relevant version of `guppy` using the default high-accuracy model configurations (HAC). Next, we aligned the basecalled reads for each sample to their

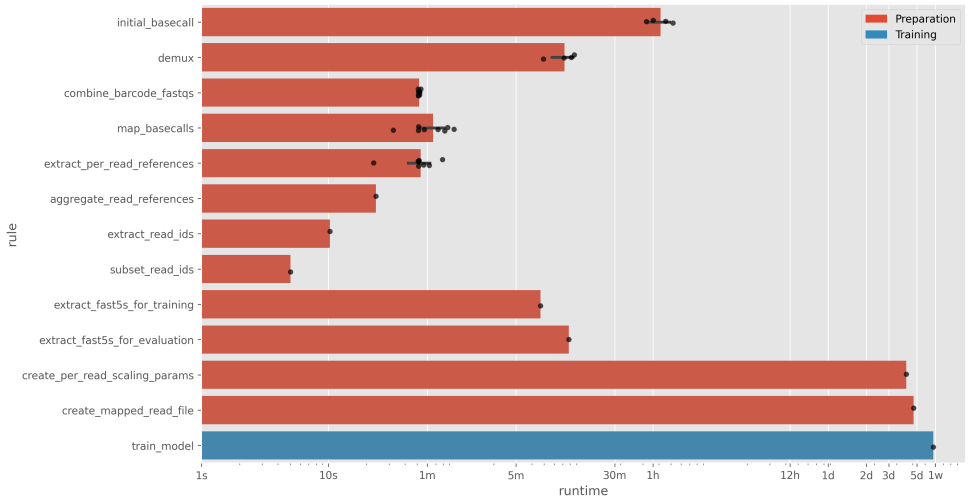


Fig. 5.1: Runtimes (x-axis) of the different stages (rules; y-axis) of preparing data for basecall model training (red) and training the model itself (blue). Each individual run is represented with a black point, and rules that have multiple runs have black 95% confidence interval bars. s=seconds; m=minutes; h=hours; d=days; w=weeks.

truth assembly (Section 5.2) with `minimap2` [41], discarding unmapped and secondary alignments.

Next, using the alignment from the previous step, we replaced the sequence for each read with the reference (truth assembly) sequence it aligns to - in the same orientation - using `taiyaki`. Finally, we discarded any read with less than 50% of its sequence aligned. The resulting "read-reference" FASTA file acts as a mapping between a read identifier and its true sequence - and is later used to match raw signal to DNA sequence for each read. At the end of this step, 1,309,759 read-references were available for subsequent training and validation.

The recommended number of reads for `taiyaki` model-training is in the range of 10,000 to 100,000. As our read-reference file had many more sequences than is required, we randomly subsampled the FASTA file into two chunks with 20% (261,950) for training and the remainder (1,047,807) to be used for validation of the final model. Using the list of read identifiers in these two subsets, we extract each read's raw (fast5) data into separate training and validation batches.

Next, we used `taiyaki` to trim 200 raw signal events from the start of each training read and 100 from the end. This trimming serves to remove adapter and barcode signals. `taiyaki` then aligns the raw signal for a read to its sequence in the read-reference file. This mapping is a vital preparation step that creates a signal-to-sequence file indicating what nucleotides are associated with a given collection of the raw signals and vice versa.

Figure 5.1 shows the runtime of each step in the preparation phase of training (in red). The two longest stages, trimming the raw signal (`create_per_read_scaling_params`) and mapping the raw signal to read-references (`create_mapped_read_file`), took 4.1 and 4.7 days respectively. In total, the full preparation pipeline ran in 8.9 days.

Training

The signal-to-sequencing mapping file produced from the preparation pipeline is the input for model training. In addition, we provided an initial model file from which training begins; the mLstm flipflop model distributed with `taiyaki`.

We trained the model using `taiyaki`'s `train_flipflop.py` script, which took 162 hours (6.75 days) to complete (blue bar in Figure 5.1) on 2 GPUs and had a peak memory usage of 57GB.

The final output from training the model is a checkpoint file, which we then convert to a `guppy`-compatible JSON configuration file using `taiyaki`.

5.4 Evaluating a custom Nanopore basecalling model

The model-training process generates a JSON file that can be used as a model configuration to basecall Nanopore reads using `guppy`. Therefore, the first step in evaluating whether our *M. tuberculosis*-specific model, `tubby`, provides improved accuracy compared to `guppy`'s default model is to basecall the validation reads set aside before training. These validation reads provide an unbiased dataset to evaluate as they were not involved in the training process.

Although the validation data was absent from training, they are from the same source (sample). Therefore, we additionally assess both models on an independent BCG dataset (Section 5.2). This BCG sample was not sequenced in the same experiment as the training and validation samples. Additionally, it is a different species (*M. bovis*) but still part of the MTBC. As such, it serves as a test of each model's generalisation capabilities.

We evaluate the read- and consensus-level accuracy of reads produced by `guppy` and `tubby` and assess the types of errors made by each.

Version	Model	Count	Mean	std	Min	25%	50%	75%	Max	Mode
3.4.5	guppy	1047829	0.9067	0.0480	0.4186	0.8838	0.9202	0.9416	1.0000	0.9444
	tubby	1047508	0.9295	**0.0402**	0.4619	0.9129	0.9415	0.9574	1.0000	0.9545
3.6.0	guppy	1110664	0.9268	0.0474	0.4186	0.9063	0.9413	0.9604	1.0000	0.9615
	tubby	1110098	**0.9423**	0.0413	0.4297	**0.9269**	**0.9554**	**0.9704**	1.0000	**0.9688**
4.4.0	guppy	1144426	0.9247	0.0496	**0.4751**	0.9038	0.9403	0.9600	1.0000	0.9628
	tubby	1143410	0.9381	0.0435	0.4723	0.9220	0.9520	0.9678	1.0000	**0.9688**

Table 5.1: Read BLAST identity summary statistics for the *M. tuberculosis*-specific basecalling model `tubby` compared with the default `guppy` model on the validation data - *M. tuberculosis* reads not used for training `tubby`. Version indicates the `guppy` version used for the basecalling before and after training. BLAST identity is the number of matching bases in a read alignment divided by the length of the alignment. Count refers to the number of reads evaluated. Bold text highlights the best-performing model for the relevant metric. std=standard deviation.

5.4.1 Read-level performance

The first evaluation metric, read BLAST identity, determines the read-level accuracy produced by the basecalling model. First, we align the basecalled reads to their respective truth assembly with `minimap2`, discarding secondary alignments (but keeping unmapped reads). Then, from the resulting pairwise alignment (PAF) file, we calculate the BLAST identity as, for each mapping, the number of matching bases divided by the length of the alignment.

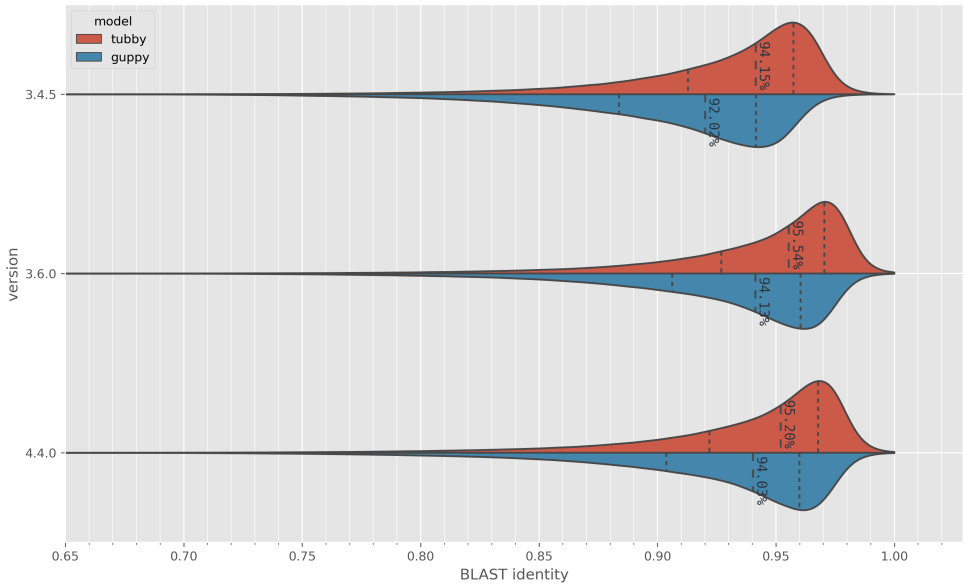
Validation data

Figure 5.2a shows the distribution of read BLAST identity values for each `guppy` version and associated `tubby` model. For all versions, `tubby` has the highest average read BLAST identity values. Interestingly, the best performing version for both models was 3.6.0, with a median BLAST identity of 95.54% and 94.13% for `tubby` and `guppy` respectively. Table 5.1 describes the summary statistics of the read identity distributions.

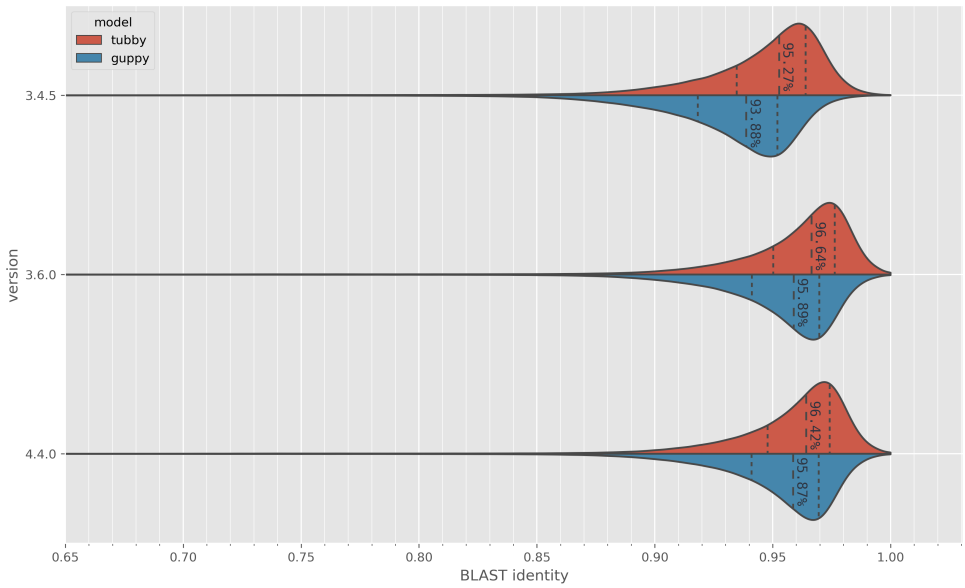
While version 3.6.0 has the highest median values for both models, version 4.4.0 has the highest minimum value for both models and the equal highest mode. We note this result as others in the Nanopore community prefer modal accuracy to median accuracy.

Test data

Figure 5.2b shows the read BLAST identities for the test BCG sample. As with the validation data, `tubby` version 3.6.0 has the highest median read identity of 96.64%



(a) Validation data (*M. tuberculosis* reads not used for training)



(b) Test data (BCG reads)

Fig. 5.2: Read BLAST identity (x-axis) for the *M. tuberculosis*-specific basecalling model `tubby` (red) compared with the default guppy model (blue). The subtitle of each plot indicates the data being assessed. The validation data (**a**) refers to *M. tuberculosis* reads that were not used in (`tubby`) model training, while test data (**b**) is reads from a BCG sample that was not involved in training. Version (y-axis) indicates the guppy version used for the basecalling before and after training. BLAST identity is the number of matching bases in a read alignment divided by the length of the alignment. The dashed lines represent the 25, 50, and 75^{th} percentiles.

Version	Model	Count	Mean	std	Min	25%	50%	75%	Max	Mode
3.4.5	guppy	787170	0.9309	0.0330	0.4738	0.9183	0.9388	0.9520	1.0000	0.9444
	tubby	790230	0.9453	0.0308	0.4764	0.9348	0.9527	0.9640	1.0000	0.9545
3.6.0	guppy	791527	0.9509	0.0321	0.3841	0.9412	0.9589	0.9698	1.0000	0.9688
	tubby	791356	**0.9587**	0.0307	**0.4819**	**0.9502**	**0.9664**	**0.9763**	1.0000	**0.9767**
4.4.0	guppy	790291	0.9507	0.0318	0.4328	0.9411	0.9587	0.9695	1.0000	0.9688
	tubby	791217	0.9566	**0.0305**	0.3915	0.9479	0.9642	0.9742	1.0000	0.9688

Table 5.2: Read BLAST identity summary statistics for the *M. tuberculosis*-specific basecalling model `tubby` compared with the default `guppy` model on the test data - a BCG sample not involved in training `tubby`. Version indicates the `guppy` version used for the basecalling before and after training. BLAST identity is the number of matching bases in a read alignment divided by the length of the alignment. Count refers to the number of reads evaluated. Bold text highlights the best-performing model for the relevant metric. std=standard deviation.

- 1.10% greater than the validation data. From Table 5.2, we see that `tubby` version 3.6.0 has the highest percentiles, mode (97.67%), and mean (95.87%) of all the models and versions.

Again, for each version, `tubby` outperforms `guppy` on all summary statistics (except the mode for version 4.4.0, which is the same).

5.4.2 Consensus-level performance

To assess the consensus-level accuracy of each model, we require assemblies from the basecalled reads. However, to compare these model-specific consensus sequences (assemblies) to the truth assembly, a reference-guided method is needed to ensure the overall structure of the truth and consensus sequences is the same. We use Rebaler (version 0.2.0; https://github.com/rrwick/Rebaler), a tool developed for exactly this use-case [78]. Briefly, `rebaler` aligns the reads to a reference and replaces the reference sequence with the sequence from the best alignments, producing an unpolished assembly. After this, it polishes the assembly with multiple rounds of `racon` to produce a consensus sequence.

We evaluate consensus accuracy in two ways. In the first, we use the same approach as in [89] to quantify the similarity of the consensus and truth assemblies for each sample. For that approach, each contig (chromosome) in the consensus assembly is aligned to the truth assembly with `minimap2`. Then, we calculate the chromosome identity for each contig's longest alignment in the same manner as the read BLAST identity (Section 5.4.1). In the second, matching [78], we cut the consensus assembly into 10kbp chunks - imitating consensus "reads" - and align these to the truth assembly

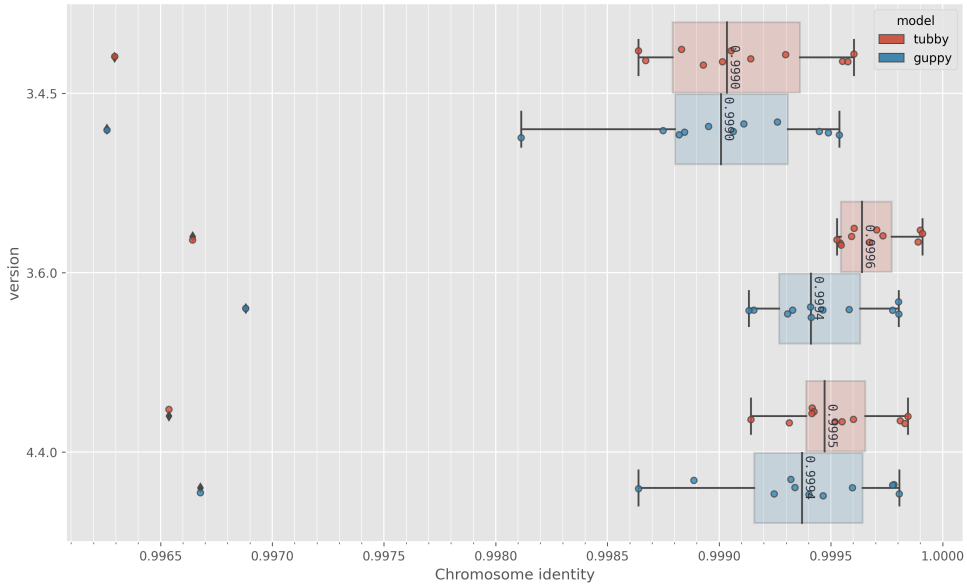


Fig. 5.3: Chromosome identity (x-axis) for the *M. tuberculosis*-specific basecalling model `tubby` (red) compared with the default guppy model (blue) on the validation data - *M. tuberculosis* reads not used for training `tubby`. Version (y-axis) indicates the guppy version used for the basecalling before and after training. For each chromosome's longest alignment to its truth assembly, chromosome identity is the number of matching bases divided by the alignment length. The coloured points indicate the individual chromosome identity for each contig in each sample.

to calculate the BLAST identity. See Section 5.5 for an extended discussion of the advantages and disadvantages of these two approaches to consensus accuracy.

Validation data

Figure 5.3 shows the chromosome identity for each version and model. We again see that `tubby` version 3.6.0 leads to the highest identity values, with a median chromosome identity value of 99.96%. Compared to the best median guppy identity of 99.94% (version 3.6.0), `tubby` provides an improvement of 0.02%, which equates to approximately 880 less erroneous positions in the *M. tuberculosis* assembly. Full summary statistics are also given in Table 5.3.

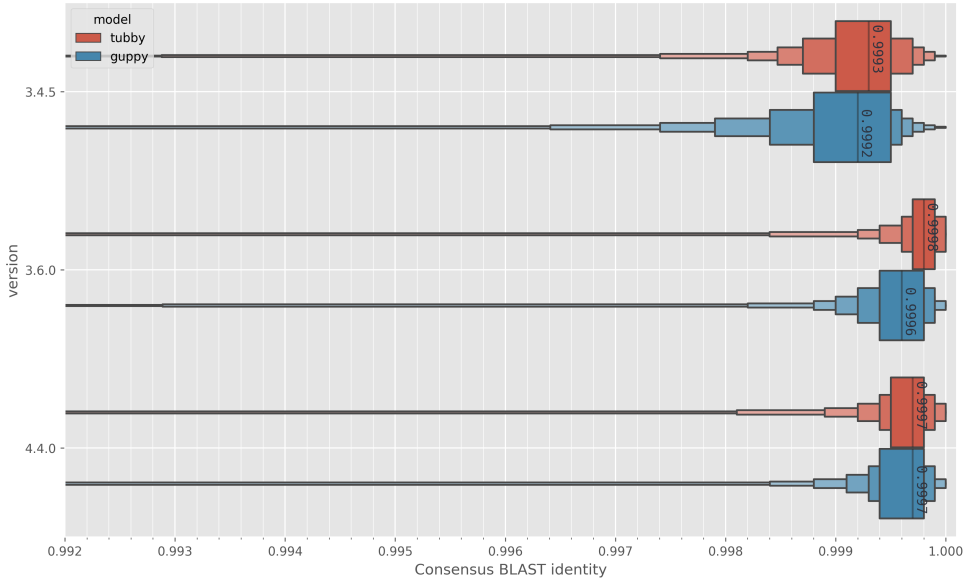
Next, for consensus BLAST identity, Figure 5.4a shows `tubby` version 3.6.0 also has the highest median value (99.98%) compared with the best guppy version (99.97%; v4.4.0). The consensus accuracy improvement of 0.01% equates to approximately 440 less erroneous positions in the *M. tuberculosis* assembly. Summary statistics are shown in Table 5.4.

Version	Model	Contigs	Mean	std	Min	25%	50%	75%	Max
3.4.5	guppy	12	0.9988	0.0009	0.9963	0.9988	0.9990	0.9993	0.9995
	tubby	12	0.9989	0.0009	0.9963	0.9988	0.9990	0.9994	0.9996
3.6.0	guppy	12	0.9993	**0.0008**	**0.9969**	0.9993	0.9994	0.9996	0.9998
	tubby	12	**0.9994**	0.0009	0.9966	**0.9995**	**0.9996**	**0.9998**	**0.9999**
4.4.0	guppy	12	0.9992	0.0009	0.9967	0.9992	0.9994	0.9996	0.9998
	tubby	12	0.9993	0.0009	0.9965	0.9994	0.9995	0.9997	0.9998

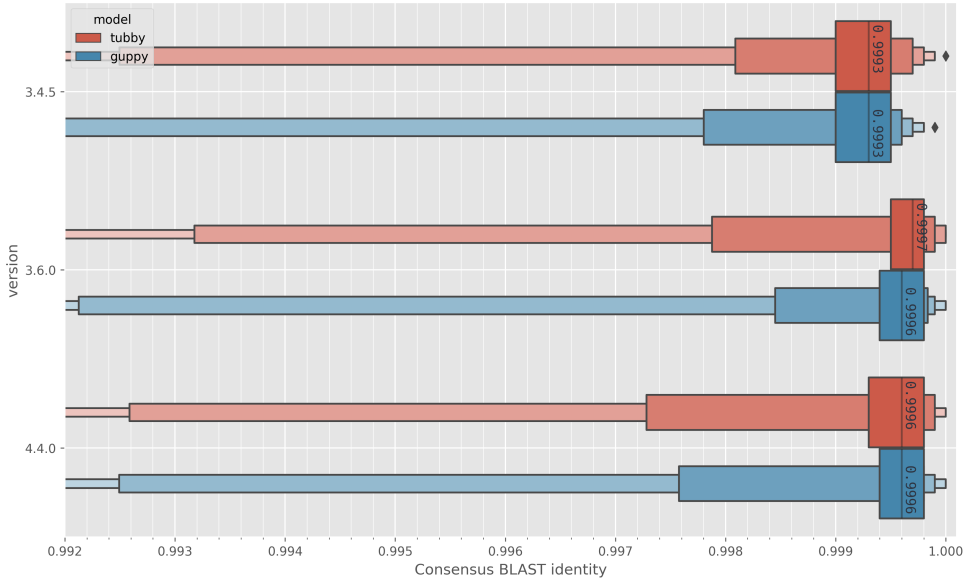
Table 5.3: Chromosome identity summary statistics for the *M. tuberculosis*-specific basecalling model `tubby` compared with the default `guppy` model on the validation data - *M. tuberculosis* reads not used for training `tubby`. Version indicates the `guppy` version used for the basecalling before and after training. For each chromosome's longest alignment to its truth assembly, chromosome identity is the number of matching bases divided by the alignment length. Bold text highlights the best-performing model for the relevant metric. std=standard deviation.

Version	Model	Contigs	Mean	std	Min	25%	50%	75%	Max
3.4.5	guppy	3531	0.9987	**0.0044**	**0.8635**	0.9988	0.9992	0.9995	1.0000
	tubby	3532	0.9990	0.0041	0.8630	0.9990	0.9993	0.9995	1.0000
3.6.0	guppy	3532	0.9993	0.0041	**0.8635**	0.9994	0.9996	0.9998	1.0000
	tubby	3534	**0.9995**	0.0043	0.8634	**0.9997**	**0.9998**	**0.9999**	1.0000
4.4.0	guppy	3531	0.9992	0.0058	0.7961	0.9994	0.9997	0.9998	1.0000
	tubby	3533	0.9993	0.0049	0.8633	0.9995	0.9997	0.9998	1.0000

Table 5.4: Consensus BLAST identity summary statistics for the validation data - *M. tuberculosis* reads not used for training `tubby`. Where consensus refers to 10kbp "chunks" of the genome assembly produced by the basecalled reads, for each model, mapped to the truth genome. Version indicates the `guppy` version used for the basecalling before and after training. BLAST identity is the number of matching bases (in a chunk alignment) divided by the length of the alignment. Count refers to the number of consensus chunks assessed. Bold text highlights the best-performing model for the relevant metric. std=standard deviation.



(a) Validation data (*M. tuberculosis* reads not used for training)



(b) Test data (BCG reads)

Fig. 5.4: Consensus BLAST identity (Y-axis), where consensus refers to 10kbp "chunks" of the genome assembly produced by the basecalled reads for each model (colours), mapped to the truth genome. The subtitle of each plot indicates the data being assessed. The validation data (**a**) refers to *M. tuberculosis* reads that were not used in (`tubby`) model training, while test data (**b**) is reads from a BCG sample that was not involved in training. Version (y-axis) indicates the guppy version used for the basecalling before and after training. BLAST identity is the number of matching bases (in a chunk alignment) divided by the alignment length. The median value for each boxplot is annotated on the middle line.

Version	3.4.5		3.6.0		4.4.0	
Model	guppy	tubby	guppy	tubby	guppy	tubby
Identity	0.9886	0.9878	0.9881	**0.9911**	0.9883	0.9884

Table 5.5: Chromosome identity for the *M. tuberculosis*-specific basecalling model tubby compared with the default guppy model on the BCG test sample. Version indicates the guppy version used for the basecalling before and after training. For each chromosome's longest alignment to its truth assembly, chromosome identity is the number of matching bases divided by the alignment length. Bold text highlights the best-performing model's chromosome identity.

Version	Model	Contigs	Mean	std	Min	25%	50%	75%	Max
3.4.5	guppy	431	0.9963	0.0159	0.8247	0.9990	0.9993	0.9995	0.9999
	tubby	431	0.9974	**0.0096**	**0.8575**	0.9990	0.9993	0.9995	1.0000
3.6.0	guppy	431	0.9965	0.0162	0.8415	0.9994	0.9996	**0.9998**	1.0000
	tubby	432	0.9966	0.0171	0.8030	**0.9995**	**0.9997**	**0.9998**	1.0000
4.4.0	guppy	431	**0.9975**	0.0118	0.8445	0.9994	0.9996	**0.9998**	1.0000
	tubby	432	0.9966	0.0183	0.7393	0.9993	0.9996	**0.9998**	1.0000

Table 5.6: Consensus BLAST identity summary statistics for the BCG test sample. Where consensus refers to 10kbp "chunks" of the genome assembly produced by the basecalled reads, for each model, mapped to the truth genome. Version indicates the guppy version used for the basecalling before and after training. BLAST identity is the number of matching bases (in a chunk alignment) divided by the length of the alignment. Count refers to the number of consensus chunks assessed. Bold text highlights the best-performing model for the relevant metric. std=standard deviation.

Test data

For the BCG test sample, the rebaler assembly for each basecalling model and version has a single contig. The chromosome identity values are listed in Table 5.5, where we see tubby version 3.6.0 produces the highest identity of 99.11%. This is a lot lower than the median value of 99.96% obtained on the validation data; it is even less than the minimum chromosome identity for any validation data consensus assembly (99.63%; Table 5.3). Although, it is significantly higher than the best guppy value of 98.86%, with the difference (0.25%) equating to approximately 11,000 less erroneous positions in the tubby assembly.

Figure 5.4b shows tubby version 3.6.0 has a higher median consensus BLAST identity (99.97%) compared with the best guppy version (99.96%; v3.6.0 and v4.4.0). The consensus accuracy improvement of 0.01% equates to approximately 440 less erroneous positions in the *M. tuberculosis* assembly. However, as shown in Table 5.6, while tubby version 3.6.0 had the highest percentile values, guppy v3.6.0 had the highest mean (99.75%).

5.4.3 Error types

Lastly, we classify the types of errors that occur in the consensus assemblies. Each `rebaler` assembly was aligned to the truth assembly using `nucmer` (from the MUMmer4 software suite [40]). `nucmer` identifies all positions of difference - errors in this case - between the two sequences. We classify errors as *Dcm* if the reported difference occurs in a known 5-methylcytosine methyltransferase motif (CCAGG or CCTGG; note, these motifs are consistent across eukaryotes and prokaryotes [215]). We label an error as *homo_del* or *homo_ins* if there are three or more consecutive occurrences of the same nucleotide surrounding a deletion or insertion, respectively. Finally, any deletions, insertions, or substitutions that do not fit into one of the categories as mentioned above are classified as *other_del*, *other_ins*, or *sub*, respectively.

We report the error rate for each error type as the number of errors attributable to that error type divided by the length of the truth assembly.

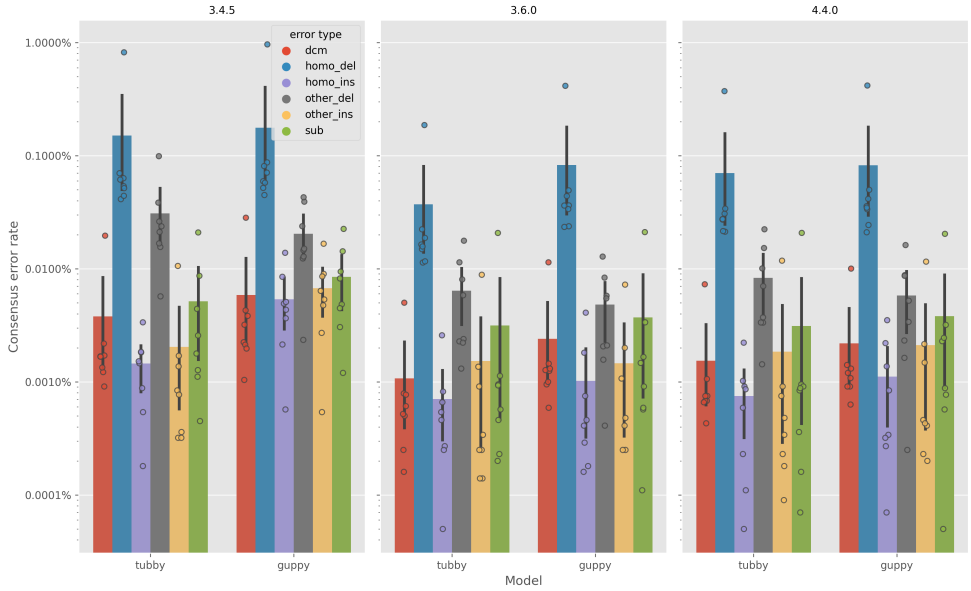
Validation data

Figure 5.5a shows that the bulk of the errors for both models are deletions, with the predominant form being homopolymers. However, we do see that, except for non-homopolymer (other) deletions, `tubby`'s errors are lower than `guppy`'s. Of particular note is homopolymer deletions in version 3.6.0, where `tubby` (0.012%) has nearly half the number of errors as `guppy` (0.021%). Both models have a very low level of Dcm-methylation errors, which is a good control as *M. tuberculosis* does not have any known 5-methylcytosine methyltransferases - although we do note this is still an active debate [216].

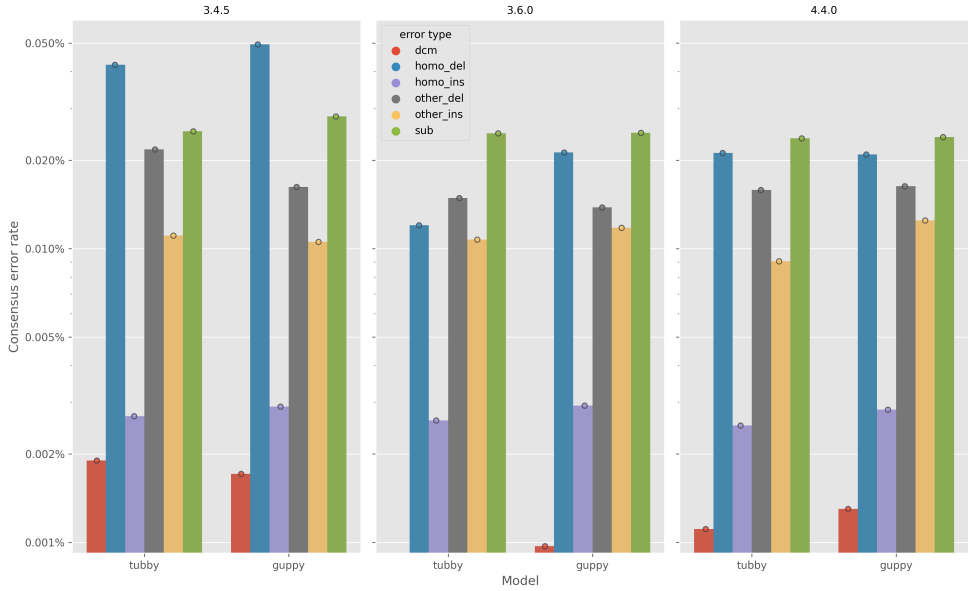
When assessing each version and model, we find that `tubby` version 3.6.0 has the overall lowest error rates.

Test data

Figure 5.5b shows the error types for the BCG test sample. Interestingly, for versions 3.6.0 and 4.4.0, substitutions are the most common form of error in the test data, in contrast to the validation data, where homopolymer deletions are the dominant form. The homopolymer deletion rate for the test data is the same as the validation, but the substitution rate is much higher. However, we note that the test sample is a single sample, whereas the evaluation data is from eight samples.



(a) Validation data (*M. tuberculosis* reads not used for training)



(b) Test data (BCG reads)

Fig. 5.5: Error types in the `rebaler` assemblies produced from `tubby` and default guppy basecalling models. The consensus error rate is the number of errors attributable to a given type, divided by the length of the truth assembly. The errors are per assembly, so the confidence intervals represent variation in error types between samples/assemblies, with the points showing each assembly's value. `dcm` (red) refers to errors in Dcm-methylation motifs. `homo` refers to homopolymer insertions (purple) or deletions (blue). `other` refers to non-homopolymer insertions (yellow) or deletions (grey). `sub` (green) is substitutions (single- or multi-base). The subtitle of each plot indicates the data being assessed. The validation data (**a**) refers to *M. tuberculosis* reads that were not used in (`tubby`) model training, while test data (**b**) is reads from a BCG sample that was not involved in training. Note, the y-axes of **(a)** and **(b)** are different.

`tubby` version 3.6.0 has the lowest homopolymer deletion error rates, and the lowest overall error rate (0.0656%) compared to `guppy` (0.0755%). We could not discover an explanation for the higher substitution rate in the test data.

5.5 Discussion

The accuracy of any sequencing data influences almost all meaningful bioinformatics applications. We have explored a number of these in Chapter 2, Chapter 3 and Chapter 4, with most of the focus being on Nanopore data. In this chapter, we trained an *M. tuberculosis*-specific Nanopore basecalling model, `tubby`, which improves the accuracy of ONT's basecalling software guppy at both the read and consensus level.

Wick *et al.* were the first to show that training a Nanopore basecalling model for a specific taxon leads to increased accuracy [78]. Indeed, their work has stood as a valuable resource for the comparison of basecalling methods and the temporal improvement of Nanopore sequencing error rates. Although their taxon of interest was *K. pneumoniae*, we find the same accuracy improvements over the default methods as a result of training specifically for *M. tuberculosis*.

The most striking difference between the results obtained by Wick *et al.* and those in this chapter are the advancement in basecalling accuracy in the space of two years. The latest `guppy` version tested by Wick *et al.* was version 2.2.3 - released in January 2019 - which yielded a median read identity of 87.15% for the default model and 90.87% for their taxon-specific model. In contrast, for `guppy` version 3.6.0 - released in April 2020 - we find read-level accuracy of 94.13% for the default model and 95.54% for our custom `tubby` model on the validation data. On our test data, we see even higher median identities of 95.89% and 96.64%, respectively.

These improvements in read identity will likely affect many applications. For example, k-mer-based methods, such as genome assembly [161], read alignment [41], species identification [217], and even variant calling ([43] and Chapter 2), are heavily affected by error rates. To illustrate, consider how the error rate for `guppy` v2.2.3 (0.1285; 1− read identity), and `guppy` (0.0587) and `tubby` (0.0446) versions 3.6.0 effects the number of erroneous k-mers. The equation $p = (1-e)^k$, describes the probability p of a k-mer having *no* errors, given an error rate e. With a commonly used Nanopore k-mer size of 15, `guppy` v2.2.3 has $p = 0.1271$, while `guppy` and `tubby` version 3.6.0 have $p = 0.4036$ and $p = 0.5044$, respectively. If a genome contains a given k-mer once, and is sequenced with no errors to a depth of 100, we would expect

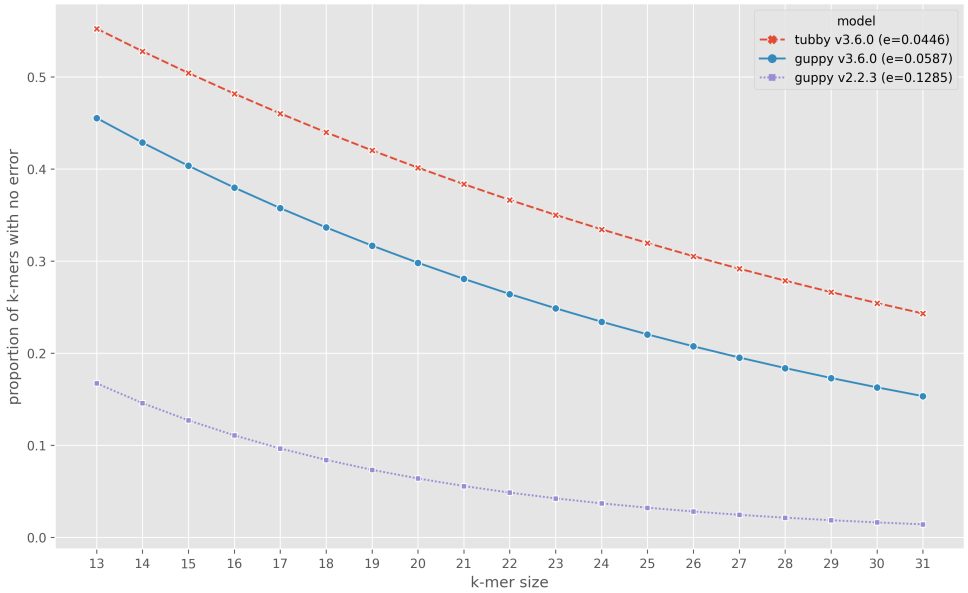


Fig. 5.6: Impact of error rate on expected *k*-mer depth. The y-axis shows the expected proportion of *k*-mers with no sequencing errors for a given basecalling model (colours) with error rate, *e*.

100 copies. However, with the Nanopore basecalling error rates just listed, guppy v2.2.3 would produce only 13 perfect copies, while guppy and tubby v3.6.0 would generate 40 and 50 perfect copies respectively.

The differences between these basecalling models are quite stark when the impact on "lost" sequencing depth is considered this way. Focusing on the version 3.6.0 models assessed in this chapter, tubby would provide 25% (1.25x) more 15-mers than guppy, and 285% (3.85x) more than guppy v2.2.3 reported by Wick *et al.*. This disparity in expected *k*-mer depth grows as the *k*-mer size increases - as shown in Figure 5.6.

An interesting observation is that for both guppy and tubby, the test data yielded *higher* read identity values than the validation data - e.g., 1.10% higher for tubby v3.6.0. As no test data were used in the training process, this suggests our model generalises well. However, we caution that our test data contains only one sample, and we do not claim generalisability outside of the MTBC. Future work would benefit from increasing the number of test samples to consolidate this result.

We chose to assess consensus accuracy in two ways to balance the strengths and weaknesses of each. The chromosome identity, as used by Wick *et al.* in 2020 [89], which provides a single value for each contig, helps gain a sense of the reliability of the assembly produced from the Nanopore reads. However, one weakness of this approach is that if there are a small number of large erroneous regions, the overall

metric is impacted. The downside here is that one may lose sight of how well the rest of the genome is assembled. A perfect example of this is the *pe/ppe* genes in *M. tuberculosis*, which can have a GC-content as high as 80%, and generally fail to assemble accurately [218]. To counteract this, we also calculate the consensus BLAST identity, as per Wick *et al.* in their 2019 bascaller comparison work mentioned above [78]. This method breaks the assembly into 10kbp pieces and aligns those to the truth genome, calculating BLAST identity as we did for the read identity. The benefit to this approach is that we get a better sense of the average accuracy of an assembly, and any erroneous regions, such as *pe/ppe* genes, do not have a significant impact on the overall result. For applications where repetitive and GC-rich regions are masked, this metric is a fairer guide than the overall chromosome identity.

In terms of the chromosome identity, `tubby` v3.6.0 had the highest median value for the validation data (99.96%) and the highest value for the single test data sample (99.11%). In the case of the validation data, this was 0.02% greater than the best `guppy` value, equating to approximately 880 fewer erroneous positions. The test data difference was a massive 0.25% (11,000 positions) difference between `tubby` and `guppy`. It is worth noting that these consensus assemblies are reference-guided, so comparisons to work such as [89] are not valid.

When assessing the consensus BLAST identity, we again find that `tubby` version 3.6.0 has the highest median value for both the validation and test data. As we used the same method as Wick *et al.* in their basecaller comparison work, we can compare our results to theirs. Doing this, we see the same trend as the read-level identity; consensus accuracy has increased dramatically in the last two years. Additionally, our taxon-specific basecalling model leads to even further improvements, as did theirs.

Comparing the latest version used by Wick *et al.*, `guppy`'s default model has improved from a v2.2.3 (Jan. 2019) median of 99.47% to our 99.97% in v4.4.0 (Dec. 2020) - 22,000 less erroneous positions in an *M. tuberculosis* genome. However, we did not see the dramatic gain in a taxon-specific model they did. For `guppy` v2.2.3, they saw the consensus identity increase to 99.93% (+0.46%), whereas we only found a 0.01% increase to 99.98%. Nevertheless, any improvement in consensus identity is valuable, especially when the percentage is nearing 100. Even an increase of 0.01% leads to 440 fewer erroneous positions in an *M. tuberculosis* genome.

The improved results from our species-specific basecalling model are not surprising given the previous work by Wick *et al.* [78]. However, it does raise the question of what data is used to train the default `guppy` models. Information about the species used is not readily available to users, which is disappointing given the impact on the

results. As we have seen, the most recent guppy version (4.4.0) did not yield the highest accuracy, leading us to wonder whether there was a shift in the species and abundances used for training in the more recent version.

While we appreciate that training a general model with all knowns species is not possible, providing a list of those used would alert users to whether their species of interest might require a taxon-specific model.

Ultimately the Nanopore community may be better served by sharing species-specific basecalling models. An important caveat here, though, is that regular benchmarks, like those maintained by Wick *et al.* [78, 89], would be necessary whenever a new guppy version is released.

Indels have always been a known systematic issue in Nanopore data [79], with the main difficulty being homopolymer deletions and methylation sites [78, 219]. Indeed, in Section 5.4.3, we found the most common error types were homopolymer deletions (and substitutions in the test data). Notably, `tubby` v3.6.0 reduced the number of homopolymer deletions by nearly half. This reduction is the same as Wick *et al.* when comparing their taxon-specific model to the default.

As Wick *et al.* used *K. pneumoniae*, which has active 5-methylcytosine methyltransferases, their major source of errors with the default guppy model was Dcm methylation sites. However, training the taxon-specific model removed nearly all of those errors. *M. tuberculosis* does not have the same methyltransferases, and so Dcm methylation sites were not an issue.

We did not see any reduction in substitutions or other types of indels using `tubby`, indicating these errors may originate from a source other than the basecaller, such as the pore itself.

The two main limitations for the work in this chapter were that we only have one test sample and that we did not assess how the basecaller improvements impact applications outside of assembly. We discuss these further in Section 5.7.

5.6 Conclusion

In conclusion, the work in this chapter shows that an *M. tuberculosis*-specific Nanopore basecalling model yields higher read- and consensus-level accuracy than default models. Additionally, the *M. tuberculosis* model reduces the main source of Nanopore errors - homopolymer deletions.

We trained and compared models for three guppy versions. Surprisingly, the most recent version did not yield the best results for *M. tuberculosis* and highlights the need for regular benchmarks of Nanopore basecalling performance.

5.7 Future work

5.7.1 *de novo* assembly assessment

The methods we used for assessing the consensus-level accuracy of `tubby` created reference-guided assemblies using `rebaler`. While this helps keep the overall structure of the assemblies the same and thus allow for focusing on the base-level accuracy, it fails to show what differences basecalling models cause on the overall assembly structure.

One solution would be to additionally assess the consensus accuracy of *de novo* genome assemblies generated from the Nanopore reads basecalled with each model. Such an analysis is similar to Wick *et al.* (2020) and could also assess other metrics like contiguity of assemblies [89].

5.7.2 *M. tuberculosis* methylation sites

The major cause of guppy basecalling errors in Enterobacteriaceae are Dcm methylation motifs [78]. While *M. tuberculosis* does not show evidence for these sites ([216] and Section 5.4.3), it would be useful to see what impact *M. tuberculosis*-specific methylation sites have on the error rate. As such, future versions of this model benchmark could include such *M. tuberculosis*-specific methylation motifs [170, 220].

5.7.3 Impact on previous work

While we have shown in this chapter that an *M. tuberculosis*-specific basecalling model improves accuracy and reduces homopolymer deletions, it remains to be seen what impact such advances have on "real-world" applications.

Chapter 3 and Chapter 4 of this thesis use Nanopore data for *M. tuberculosis* transmission clustering and drug resistance prediction. One test of the utility of `tubby` would be to rerun the analyses for those chapters with Nanopore data basecalled with `tubby`. That way, any change in results can be directly attributed to the taxon-specific model.

5.7.4 Increased diversity of data

One of the main challenges with training a basecalling model is data. The data used to train, validate, and test must have a known, reliable truth genome and, ideally, be free of laboratory biases. While the number of reads we use for training and validation in this chapter was more than sufficient, they were all sequenced at a similar time by the same technician in the same laboratory. Therefore, it would be prudent to include some data in the training process sequenced at a separate time and ideally by different people in a different place. Additionally, including data from other species in the MTBC would likely prove beneficial as non-tuberculous mycobacteria (NTM) are becoming an ever-increasing cause for concern globally [109].

5.7.5 New Conditional random field models

As mentioned already, Nanopore sequencing technology is evolving at a rapid pace. Whenever a study using Nanopore is published, it is undoubtedly out-of-date with the latest tools being used - due to no fault of the authors. At the time of writing, a new major version of guppy (version 5) has been released. It uses a new model format based on conditional random fields (CRFs), which employ a probabilistic model to segment and label data [221] - a vital component of the basecalling process. These new models can be trained using a new (easier) process in the ONT software Bonito (https://github.com/nanoporetech/bonito).

The new CRF-based models (dubbed "super-accurate" models by ONT) are reported to provide a modal read accuracy increase of 0.5% over the guppy versions used in this chapter (https://community.nanoporetech.com/posts/guppy-v5-0-7-release-note). Therefore, updating `tubby`'s training process for these new CRF-based models may yield an even further improvement to the accuracy reported in this chapter. (However, we note this section itself will likely be out-of-date by the time it is read).

5.8 Availability of data and materials

need to make tubby public prior to submission

`tubby` is open-source and freely available under an MIT license at https://github.com/mbhall88/tubby. The models are available under Releases in compressed JSON format. The pipelines and scripts used in this chapter are available at the same location. All analyses were run using the workflow management program `snakemake` [192].

All figures were generated using the Python libraries `matplotlib` [193] and `seaborn` [194].

References

[1] Ruiting Lan et al. “Intraspecies variation in bacterial genomes: the need for a species genome concept”. In: *Trends in Microbiology* 8.9 (2000), pp. 396–401. ISSN: 0966-842X. DOI: 10.1016/s0966-842x(00)01791-1.

[2] James O. McInerney, Alan McNally, and Mary J. O’Connell. “Why prokaryotes have pangenomes”. In: *Nature Microbiology* 2.4 (2017), p. 17040. DOI: 10.1038/nmicrobiol.2017.40.

[3] Stephen McGrath, D van Sinderen, et al. *Bacteriophage: genetics and molecular biology*. Caister Academic Press, 2007.

[4] Yin Ning Chiang, José R. Penadés, and John Chen. “Genetic transduction by phages and chromosomal islands: The new and noncanonical”. In: *PLOS Pathogens* 15.8 (2019), e1007878. ISSN: 1553-7366. DOI: 10.1371/journal.ppat.1007878.

[5] Shannon M. Soucy, Jinling Huang, and Johann Peter Gogarten. “Horizontal gene transfer: building the web of life”. In: *Nature Reviews Genetics* 16.8 (2015), pp. 472–482. ISSN: 1471-0056. DOI: 10.1038/nrg3962.

[6] Rosemary J. Redfield. “Do bacteria have sex?” In: *Nature Reviews Genetics* 2.8 (2001), pp. 634–639. ISSN: 1471-0056. DOI: 10.1038/35084593.

[7] Calum Johnston et al. “Bacterial transformation: distribution, shared mechanisms and divergent control”. In: *Nature Reviews Microbiology* 12.3 (2014), pp. 181–196. ISSN: 1740-1526. DOI: 10.1038/nrmicro3199.

[8] Wei Ding, Franz Baumdicker, and Richard A Neher. “panX: pan-genome analysis and exploration”. In: *Nucleic Acids Research* 46.1 (2018), e5. DOI: 10.1093/nar/gkx977. eprint: /oup/backfile/content_public/journal/nar/46/1/10.1093_nar_gkx977/1/gkx977.pdf. URL: http://dx.doi.org/10.1093/nar/gkx977.

[9] Alexander E. Lobkovsky, Yuri I. Wolf, and Eugene V. Koonin. “Gene Frequency Distributions Reject a Neutral Model of Genome Evolution”. In: *Genome Biology and Evolution* 5.1 (2013), pp. 233–242. ISSN: 1759-6653. DOI: 10.1093/gbe/evt002.

[10] Rachel M. Colquhoun et al. “Pandora: nucleotide-resolution bacterial pan-genomics with reference graphs”. In: *Genome Biology* 22.1 (Sept. 2021), p. 267. ISSN: 1474-760X. DOI: 10.1186/s13059-021-02473-1. URL: https://doi.org/10.1186/s13059-021-02473-1.

[11] Pascal Lapierre and J. Peter Gogarten. “Estimating the size of the bacterial pan-genome”. In: *Trends in Genetics* 25.3 (2009), pp. 107–110. ISSN: 0168-9525. DOI: 10.1016/j.tig.2008.12.004.

[12] Hervé Tettelin and Duccio Medini, eds. *The Pangenome*. Springer International Publishing, 2020. DOI: 10.1007/978-3-030-38281-0. (Visited on 09/14/2021).

[13] Miriam Land et al. “Insights from 20 years of bacterial genome sequencing”. In: *Functional & Integrative Genomics* 15.2 (2015), pp. 141–161. ISSN: 1438-793X. DOI: 10.1007/s10142-015-0433-4.

References

[14] Gal Horesh et al. "Different evolutionary trends form the twilight zone of the bacterial pan-genome". In: *Microbial Genomics* 7.9, 000670 (2021). ISSN: 2057-5858. DOI: https://doi.org/10.1099/mgen.0.000670. URL: https://www.microbiologyresearch.org/content/journal/mgen/10.1099/mgen.0.000670.

[15] Manish Boolchandani, Alaric W. D'Souza, and Gautam Dantas. "Sequencing-based methods and resources to study antimicrobial resistance". In: *Nature Reviews Genetics* 20.6 (2019), pp. 356–370. ISSN: 1471-0056. DOI: 10.1038/s41576-019-0108-4.

[16] Alejandro Vasquez-Rifo et al. "The Pseudomonas aeruginosa accessory genome elements influence virulence towards Caenorhabditis elegans". In: *Genome Biology* 20.1 (2019), p. 270. DOI: 10.1186/s13059-019-1890-1.

[17] Andrew J. Page et al. "Roary: rapid large-scale prokaryote pan genome analysis". In: *Bioinformatics* 31.22 (2015), pp. 3691–3693. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btv421.

[18] Gerry Tonkin-Hill et al. "Producing polished prokaryotic pangenomes with the Panaroo pipeline". In: *Genome Biology* 21.1 (2020), p. 180. DOI: 10.1186/s13059-020-02090-4.

[19] Brian J. Arnold et al. "Weak Epistasis May Drive Adaptation in Recombining Bacteria". In: *Genetics* 208.3 (2018), genetics.300662.2017. ISSN: 0016-6731. DOI: 10.1534/genetics.117.300662.

[20] Taj Azarian et al. "The impact of serotype-specific vaccination on phylodynamic parameters of Streptococcus pneumoniae and the pneumococcal pan-genome". In: *PLOS Pathogens* 14.4 (2018), e1006966. ISSN: 1553-7366. DOI: 10.1371/journal.ppat.1006966.

[21] Alan McNally et al. "Combined Analysis of Variation in Core, Accessory and Regulatory Genome Regions Provides a Super-Resolution View into the Evolution of Bacterial Populations". In: *PLOS Genetics* 12.9 (2016), e1006280. ISSN: 1553-7390. DOI: 10.1371/journal.pgen.1006280.

[22] Angela J. Taylor et al. "Characterization of Foodborne Outbreaks of Salmonella enterica Serovar Enteritidis with Whole-Genome Sequencing Single Nucleotide Polymorphism-Based Analysis for Surveillance and Outbreak Detection". In: *Journal of Clinical Microbiology* 53.10 (2015), pp. 3334–3340. ISSN: 0095-1137. DOI: 10.1128/jcm.01280-15.

[23] Kelly L. Wyres et al. "Genomic surveillance of antimicrobial resistant bacterial colonisation and infection in intensive care patients". In: *BMC Infectious Diseases* 21.1 (2021), p. 683. DOI: 10.1186/s12879-021-06386-z.

[24] A. J. H. Cremers et al. "Surveillance-embedded genomic outbreak resolution of methicillin-susceptible Staphylococcus aureus in a neonatal intensive care unit". In: *Scientific Reports* 10.1 (2020), p. 2619. DOI: 10.1038/s41598-020-59015-1.

[25] Heng Li. "Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM". In: *arXiv* (2013). eprint: 1303.3997.

[26] Ben Langmead and Steven L Salzberg. "Fast gapped-read alignment with Bowtie 2". In: *Nature Methods* 9.4 (2012), pp. 357–359. ISSN: 1548-7091. DOI: 10.1038/nmeth.1923.

[27] Nathan D. Olson et al. "Best practices for evaluating single nucleotide variant calling methods for microbial genomics". In: *Frontiers in Genetics* 6 (2015), p. 235. DOI: 10.3389/fgene.2015.00235.

[28] Erik Garrison and Gabor Marth. "Haplotype-based variant detection from short-read sequencing". In: *arXiv* (2012). eprint: 1207.3907.

[29] Petr Danecek et al. “Twelve years of SAMtools and BCFtools”. In: *GigaScience* 10.2 (2021), giab008. ISSN: 2047-217X. DOI: 10.1093/gigascience/giab008.

[30] Heng Li et al. “The Sequence Alignment/Map format and SAMtools”. In: *Bioinformatics* 25.16 (2009), pp. 2078–2079. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btp352.

[31] Ryan Poplin et al. “Scaling accurate genetic variant discovery to tens of thousands of samples”. In: *bioRxiv* (2018), p. 201178. DOI: 10.1101/201178.

[32] Fanny-Dhelia Pajuste et al. “FastGT: an alignment-free method for calling common SNVs directly from raw sequencing reads”. In: *Scientific Reports* 7.1 (2017), p. 2537. DOI: 10.1038/s41598-017-02487-5.

[33] Ariya Shajii et al. “Fast genotyping of known SNPs through approximate k -mer matching”. In: *Bioinformatics* 32.17 (2016), pp. i538–i544. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btw460.

[34] Peter A Audano, Shashidhar Ravishankar, and Fredrik O Vannberg. “Mapping-free variant calling using haplotype reconstruction from k-mer frequencies”. In: *Bioinformatics* 34.10 (2017), pp. 1659–1665. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btx753.

[35] Shea N Gardner, Tom Slezak, and Barry G. Hall. “kSNP3.0: SNP detection and phylogenetic analysis of genomes without genome alignment or reference genome”. In: *Bioinformatics* 31.17 (2015), pp. 2877–2878. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btv271.

[36] Jose A Bazan et al. “Large Cluster of Neisseria meningitidis Urethritis in Columbus, Ohio, 2015”. In: *Clinical Infectious Diseases* 65.1 (May 2017), pp. 92–99. ISSN: 1058-4838. DOI: 10.1093/cid/cix215. eprint: https://academic.oup.com/cid/article-pdf/65/1/92/24266277/cix215.pdf. URL: https://doi.org/10.1093/cid/cix215.

[37] Brian H. Raphael et al. “Genomic Resolution of Outbreak-Associated Legionella pneumophila Serogroup 1 Isolates from New York State”. In: *Applied and Environmental Microbiology* 82.12 (2016), pp. 3582–3590. DOI: 10.1128/AEM.00362-16. eprint: https://journals.asm.org/doi/pdf/10.1128/AEM.00362-16. URL: https://journals.asm.org/doi/abs/10.1128/AEM.00362-16.

[38] Sopio Chochua et al. “Invasive Serotype 35B Pneumococci Including an Expanding Serotype Switch Lineage, United States, 2015–2016”. In: *Emerging Infectious Diseases* 23.6 (2017), pp. 922–930. ISSN: 1080-6040. DOI: 10.3201/eid2306.170071.

[39] Andrzej Zielezinski et al. “Benchmarking of alignment-free sequence comparison methods”. In: *Genome Biology* 20.1 (2019), p. 144. DOI: 10.1186/s13059-019-1755-7.

[40] Guillaume Marçais et al. “MUMmer4: A fast and versatile genome alignment system”. In: *PLOS Computational Biology* 14.1 (2018), e1005944. ISSN: 1553-734X. DOI: 10.1371/journal.pcbi.1005944.

[41] Heng Li. “Minimap2: pairwise alignment for nucleotide sequences”. In: *Bioinformatics* 34.18 (2018), pp. 3094–3100. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bty191. eprint: 1708.01492.

[42] Todd J Treangen et al. “The Harvest suite for rapid core-genome alignment and visualization of thousands of intraspecific microbial genomes”. In: *Genome Biology* 15.11 (2014), p. 524. DOI: 10.1186/s13059-014-0524-x.

[43] Zamin Iqbal et al. “De novo assembly and genotyping of variants using colored de Bruijn graphs”. In: *Nature Genetics* 44.2 (2012), pp. 226–232. ISSN: 1061-4036. DOI: 10.1038/ng.1028.

References

[44] Phelim Bradley et al. "Rapid antibiotic-resistance predictions from genome sequence data for Staphylococcus aureus and Mycobacterium tuberculosis". In: *Nature Communications* 6.1 (2015), p. 10063. DOI: 10.1038/ncomms10063.

[45] Martin Hunt et al. "Antibiotic resistance prediction for Mycobacterium tuberculosis from genome sequence data with Mykrobe". In: *Wellcome Open Research* 4 (2019), p. 191. ISSN: 2398-502X. DOI: 10.12688/wellcomeopenres.15603.1.

[46] Matthew J. Stasiewicz et al. "Whole-Genome Sequencing Allows for Improved Identification of Persistent Listeria monocytogenes in Food-Associated Environments". In: *Applied and Environmental Microbiology* 81.17 (2015), pp. 6024–6037. DOI: 10.1128/AEM.01049-15. eprint: https://journals.asm.org/doi/pdf/10.1128/AEM.01049-15. URL: https://journals.asm.org/doi/abs/10.1128/AEM.01049-15.

[47] Bernadette C Young et al. "Severe infections emerge from commensal bacteria by adaptive evolution". In: *eLife* 6 (Dec. 2017). Ed. by Matthew TG Holden, e30637. ISSN: 2050-084X. DOI: 10.7554/eLife.30637. URL: https://doi.org/10.7554/eLife.30637.

[48] John A Lees et al. "Large scale genomic analysis shows no evidence for pathogen adaptation between the blood and cerebrospinal fluid niches during bacterial meningitis". In: *Microbial Genomics* 3.1, mgen000103 (2017). ISSN: 2057-5858. DOI: https://doi.org/10.1099/mgen.0.000103. URL: https://www.microbiologyresearch.org/content/journal/mgen/10.1099/mgen.0.000103.

[49] Stephen J Bush et al. "Genomic diversity affects the accuracy of bacterial single-nucleotide polymorphism–calling pipelines". In: *GigaScience* 9.2 (2020), giaa007–. ISSN: 2047-217X. DOI: 10.1093/gigascience/giaa007.

[50] Frederic Bertels et al. "Automated Reconstruction of Whole-Genome Phylogenies from Short-Sequence Reads". In: *Molecular Biology and Evolution* 31.5 (2014), pp. 1077–1088. ISSN: 0737-4038. DOI: 10.1093/molbev/msu088.

[51] Adam Price and Cynthia Gibas. "The quantitative impact of read mapping to non-native reference genomes in comparative RNA-Seq studies". In: *PLOS ONE* 12.7 (2017), e0180904. DOI: 10.1371/journal.pone.0180904.

[52] Arthur W. Pightling, Nicholas Petronella, and Franco Pagotto. "Choice of Reference Sequence and Assembler for Alignment of Listeria monocytogenes Short-Read Sequence Data Greatly Influences Rates of Error in SNP Analyses". In: *PLoS ONE* 9.8 (2014), e104579. DOI: 10.1371/journal.pone.0104579.

[53] "Computational pan-genomics: status, promises and challenges". In: *Briefings in Bioinformatics* (2016), bbw089. ISSN: 1477-4054. DOI: 10.1093/bib/bbw089.

[54] Rachel M. Sherman and Steven L. Salzberg. "Pan-genomics in the human genome era". In: *Nature Reviews Genetics* 21.4 (2020), pp. 243–254. ISSN: 1471-0056. DOI: 10.1038/s41576-020-0210-7.

[55] Jordan M. Eizenga et al. "Pangenome Graphs". In: *Annual Review of Genomics and Human Genetics* 21.1 (2020), pp. 139–162. ISSN: 1527-8204. DOI: 10.1146/annurev-genom-120219-080406.

[56] Hannes P Eggertsson et al. "Graphtyper enables population-scale genotyping using pangenome graphs". In: *Nature Genetics* 49.11 (2017), pp. 1654–1660. ISSN: 1061-4036. DOI: 10.1038/ng.3964.

[57] Hannes P. Eggertsson et al. "GraphTyper2 enables population-scale genotyping of structural variation using pangenome graphs". In: *Nature Communications* 10.1 (2019), p. 5402. DOI: 10.1038/s41467-019-13341-9.

[58] Erik Garrison et al. “Variation graph toolkit improves read mapping by representing genetic variation in the reference”. In: *Nature Biotechnology* 36.9 (2018), pp. 875–879. ISSN: 1087-0156. DOI: 10.1038/nbt.4227.

[59] Adam M. Novak et al. “Genome Graphs”. In: *bioRxiv* (2017). This preprint describes the vg variant calling component (as referenced in the vg nature biotech paper), p. 101378. DOI: 10.1101/101378.

[60] Kévin Da Silva et al. “StrainFLAIR: Strain-level profiling of metagenomic samples using variation graphs”. In: *bioRxiv* (2021). DOI: 10.1101/2021.02.12.430979.

[61] Brice Letcher, Martin Hunt, and Zamin Iqbal. “Gramtools enables multiscale variation analysis with genome graphs”. In: *Genome Biology* 22.1 (2021), p. 259. DOI: 10.1186/s13059-021-02474-0.

[62] Heng Li, Xiaowen Feng, and Chong Chu. “The design and construction of reference pangenome graphs with minigraph”. In: *Genome Biology* 21.1 (2020), p. 265. DOI: 10.1186/s13059-020-02168-z.

[63] Sorina Maciuca et al. “Algorithms in Bioinformatics, 16th International Workshop, WABI 2016, Aarhus, Denmark, August 22-24, 2016. Proceedings”. In: *Lecture Notes in Computer Science* (2016), pp. 222–233. ISSN: 0302-9743. DOI: 10.1007/978-3-319-43681-4_18.

[64] Javier Tamames. “Evolution of gene order conservation in prokaryotes”. In: *Genome Biology* 2.6 (2001), research0020.1. ISSN: 1465-6906. DOI: 10.1186/gb-2001-2-6-research0020.

[65] Eduardo P.C. Rocha. “The Organization of the Bacterial Genome”. In: *Annual Review of Genetics* 42.1 (2008), pp. 211–233. ISSN: 0066-4197. DOI: 10.1146/annurev.genet.42.110807.091653.

[66] Rachel Colquhoun. “Pan-genomic analysis of clonal bacterial samples using nanopore reads and genome graphs”. DPhil thesis. University of Oxford, 2019.

[67] Michael Roberts et al. “Reducing storage requirements for biological sequence comparison”. In: *Bioinformatics* 20.18 (2004), pp. 3363–3369. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bth408.

[68] Heng Li. “Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences”. In: *Bioinformatics* 32.14 (2016), pp. 2103–2110. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btw152. eprint: 1512.01801.

[69] David Deamer, Mark Akeson, and Daniel Branton. “Three decades of nanopore sequencing”. In: *Nature Biotechnology* 34.5 (2016), pp. 518–524. ISSN: 1087-0156. DOI: 10.1038/nbt.3423.

[70] Joshua Quick, Aaron R Quinlan, and Nicholas J Loman. “A reference bacterial genome dataset generated on the MinION™ portable single-molecule nanopore sequencer”. In: *GigaScience* 3.1 (2014), p. 22. DOI: 10.1186/2047-217x-3-22.

[71] The HDF Group. *Hierarchical Data Format, version 5*. https://www.hdfgroup.org/HDF5/. 2021.

[72] Franka J. Rang, Wigard P. Kloosterman, and Jeroen de Ridder. “From squiggle to basepair: computational approaches for improving nanopore sequencing read accuracy”. In: *Genome Biology* 19.1 (2018), p. 90. DOI: 10.1186/s13059-018-1462-9.

[73] Sara Goodwin et al. “Oxford Nanopore sequencing, hybrid error correction, and de novo assembly of a eukaryotic genome”. In: *Genome Research* 25.11 (2015), pp. 1750–1756. ISSN: 1088-9051. DOI: 10.1101/gr.191395.115.

[74] Jordi Silvestre-Ryan and Ian Holmes. "Pair consensus decoding improves accuracy of neural network basecallers for nanopore sequencing". In: *Genome Biology* 22.1 (2021), p. 38. DOI: 10.1186/s13059-020-02255-1.

[75] Vladimír Boža, Broňa Brejová, and Tomáš Vinař. "DeepNano: Deep recurrent neural networks for base calling in MinION nanopore reads". In: *PLOS ONE* 12.6 (2017), e0178751. DOI: 10.1371/journal.pone.0178751. eprint: 1603.09195.

[76] Haotian Teng et al. "Chiron: Translating nanopore raw signal directly into nucleotide sequence using deep learning". In: *GigaScience* 7.5 (2018), giy037–. ISSN: 2047-217X. DOI: 10.1093/gigascience/giy037.

[77] Marcus Stoiber and James Brown. "BasecRAWller: Streaming Nanopore Basecalling Directly from Raw Signal". In: *bioRxiv* (2017), p. 133058. DOI: 10.1101/133058.

[78] Ryan R. Wick, Louise M. Judd, and Kathryn E. Holt. "Performance of neural network basecalling tools for Oxford Nanopore sequencing". In: *Genome Biology* 20.1 (2019), p. 129. DOI: 10.1186/s13059-019-1727-y.

[79] Mick Watson and Amanda Warr. "Errors in long-read assemblies can critically affect protein prediction". In: *Nature Biotechnology* 37.2 (2019), pp. 124–126. ISSN: 1087-0156. DOI: 10.1038/s41587-018-0004-z.

[80] Nicholas J Loman, Joshua Quick, and Jared T Simpson. "A complete bacterial genome assembled de novo using only nanopore sequencing data". In: *Nature Methods* 12.8 (2015), pp. 733–735. ISSN: 1548-7091. DOI: 10.1038/nmeth.3444.

[81] Jared T Simpson et al. "Detecting DNA cytosine methylation using nanopore sequencing". In: *Nature Methods* 14.4 (2017), pp. 407–410. ISSN: 1548-7091. DOI: 10.1038/nmeth.4184.

[82] Ruibang Luo et al. "A multi-task convolutional deep neural network for variant calling in single molecule sequencing". In: *Nature Communications* 10.1 (2019), p. 998. DOI: 10.1038/s41467-019-09025-z.

[83] Ryan Poplin et al. "A universal SNP and small-indel variant caller using deep neural networks". In: *Nature Biotechnology* 36.10 (2018), pp. 983–987. ISSN: 1087-0156. DOI: 10.1038/nbt.4235.

[84] Ruibang Luo et al. "Exploring the limit of using a deep neural network on pileup data for germline variant calling". In: *Nature Machine Intelligence* 2.4 (2020), pp. 220–227. DOI: 10.1038/s42256-020-0167-4.

[85] Nicholas D. Sanderson et al. "High precision Neisseria gonorrhoeae variant and antimicrobial resistance calling from metagenomic Nanopore sequencing". In: *Genome Research* (2020). ISSN: 1088-9051. DOI: 10.1101/gr.262865.120.

[86] David R Greig et al. "Analysis of a small outbreak of Shiga toxin-producing Escherichia coli O157:H7 using long-read sequencing". In: *Microbial Genomics* (2021). DOI: 10.1099/mgen.0.000545.

[87] Arnold Bainomugisa et al. "A complete high-quality MinION nanopore assembly of an extensively drug-resistant Mycobacterium tuberculosis Beijing lineage strain identifies novel variation in repetitive PE/PPE gene regions". In: *Microbial Genomics* 4.7 (2018). ISSN: 2057-5858. DOI: 10.1099/mgen.0.000188.

[88] David R Greig et al. "Comparison of single-nucleotide variants identified by Illumina and Oxford Nanopore technologies in the context of a potential outbreak of Shiga toxin–producing Escherichia coli". In: *GigaScience* 8.8 (2019). DOI: 10.1093/gigascience/giz104.

[89] Ryan R. Wick and Kathryn E. Holt. "Benchmarking of long-read assemblers for prokaryote whole genome sequencing". In: *F1000Research* 8 (2020), p. 2138. DOI: 10.12688/f1000research.21782.4.

[90] Leho Tedersoo et al. "Perspectives and Benefits of High-Throughput Long-Read Sequencing in Microbial Ecology". In: *Applied and Environmental Microbiology* 87.17 (2021), e00626–21. ISSN: 0099-2240. DOI: 10.1128/aem.00626-21.

[91] Carol Smith et al. "Assessing Nanopore sequencing for clinical diagnostics: A comparison of NGS methods for Mycobacterium tuberculosis". In: *Journal of Clinical Microbiology* (2020). ISSN: 0095-1137. DOI: 10.1128/jcm.00583-20.

[92] Nuno Rodrigues Faria et al. "Mobile real-time surveillance of Zika virus in Brazil". In: *Genome Medicine* 8.1 (2016), p. 97. DOI: 10.1186/s13073-016-0356-2.

[93] Thomas Hoenen et al. "Nanopore Sequencing as a Rapidly Deployable Ebola Outbreak Tool". In: *Emerging Infectious Diseases* 22.2 (2016), pp. 331–334. ISSN: 1080-6040. DOI: 10.3201/eid2202.151796.

[94] Joshua Quick et al. "Real-time, portable genome sequencing for Ebola surveillance". In: *Nature* 530.7589 (2016), pp. 228–232. ISSN: 0028-0836. DOI: 10.1038/nature16996.

[95] Antonina A. Votintseva et al. "Same-Day Diagnostic and Surveillance Data for Tuberculosis via Whole-Genome Sequencing of Direct Respiratory Samples". In: *Journal of Clinical Microbiology* 55.5 (2017), pp. 1285–1298. ISSN: 0095-1137. DOI: 10.1128/jcm.02483-16.

[96] Nicholas D Sanderson et al. "Real-time analysis of nanopore-based metagenomic sequencing from infected orthopaedic devices". In: *BMC Genomics* 19.1 (2018), p. 714. DOI: 10.1186/s12864-018-5094-y.

[97] Minh Duc Cao et al. "Streaming algorithms for identification of pathogens and antibiotic resistance potential from real-time MinIONTM sequencing". In: *GigaScience* 5.1 (2016), p. 32. ISSN: 2047-217X. DOI: 10.1186/s13742-016-0137-2.

[98] Alexander Payne et al. "Readfish enables targeted nanopore sequencing of gigabase-sized genomes". In: *Nature Biotechnology* 39.4 (2021), pp. 442–450. ISSN: 1087-0156. DOI: 10.1038/s41587-020-00746-x.

[99] Sam Kovaka et al. "Targeted nanopore sequencing by real-time mapping of raw electrical signal with UNCALLED". In: *Nature Biotechnology* 39.4 (2021), pp. 431–441. ISSN: 1087-0156. DOI: 10.1038/s41587-020-0731-9.

[100] Madhukar Pai et al. "Tuberculosis". In: *Nature Reviews Disease Primers* 2.1 (2016), p. 16076. DOI: 10.1038/nrdp.2016.76.

[101] Thierry Wirth et al. "Origin, Spread and Demography of the Mycobacterium tuberculosis Complex". In: *PLoS Pathogens* 4.9 (2008), e1000160. ISSN: 1553-7366. DOI: 10.1371/journal.ppat.1000160.

[102] Iñaki Comas et al. "Out-of-Africa migration and Neolithic coexpansion of Mycobacterium tuberculosis with modern humans". In: *Nature Genetics* 45.10 (2013), pp. 1176–1182. ISSN: 1061-4036. DOI: 10.1038/ng.2744.

[103] *Global tuberculosis report 2020*. Tech. rep. Geneva: World Health Organization, 2020.

[104] L.M. O'Reilly and C.J. Daborn. "The epidemiology of Mycobacterium bovis infections in animals and man: A review". In: *Tubercle and Lung Disease* 76 (1995), pp. 1–46. ISSN: 0962-8479. DOI: https://doi.org/10.1016/0962-8479(95)90591-X. URL: https://www.sciencedirect.com/science/article/pii/096284799590591X.

[105] *The end TB strategy*. Tech. rep. WHO/HTM/TB/2015.19. Geneva: World Health Organization, 2015.

[106] Alice Kizny Gordon et al. “Clinical and public health utility of Mycobacterium tuberculosis whole genome sequencing”. In: *International Journal of Infectious Diseases* (2021). ISSN: 1201-9712. DOI: 10.1016/j.ijid.2021.02.114.

[107] Conor J. Meehan et al. “Whole genome sequencing of Mycobacterium tuberculosis: current standards and open issues”. In: *Nature Reviews Microbiology* 17.9 (2019), pp. 533–545. ISSN: 1740-1526. DOI: 10.1038/s41579-019-0214-5.

[108] Susanne Homolka et al. “High Resolution Discrimination of Clinical Mycobacterium tuberculosis Complex Strains Based on Single Nucleotide Polymorphisms”. In: *PLoS ONE* 7.7 (2012), e39855. DOI: 10.1371/journal.pone.0039855.

[109] Matt D. Johansen, Jean-Louis Herrmann, and Laurent Kremer. “Non-tuberculous mycobacteria and the rise of Mycobacterium abscessus”. In: *Nature Reviews Microbiology* 18.7 (2020), pp. 392–407. ISSN: 1740-1526. DOI: 10.1038/s41579-020-0331-1.

[110] R Andres Floto et al. “US Cystic Fibrosis Foundation and European Cystic Fibrosis Society consensus recommendations for the management of non-tuberculous mycobacteria in individuals with cystic fibrosis: executive summary”. In: *Thorax* 71.1 (2016), p. 88. ISSN: 0040-6376. DOI: 10.1136/thoraxjnl-2015-207983.

[111] J. Mäkinen et al. “Evaluation of a novel strip test, GenoType Mycobacterium CM/AS, for species identification of mycobacterial cultures”. In: *Clinical Microbiology and Infection* 12.5 (2006), pp. 481–483. ISSN: 1469-0691. DOI: 10.1111/j.1469-0691.2006.01380.x.

[112] T Phuong Quan et al. “Evaluation of Whole-Genome Sequencing for Mycobacterial Species Identification and Drug Susceptibility Testing in a Clinical Setting: a Large-Scale Prospective Assessment of Performance against Line Probe Assays and Phenotyping”. In: *Journal of Clinical Microbiology* 56.2 (2018), e01480–17. ISSN: 0095-1137. DOI: 10.1128/jcm.01480-17.

[113] Francesc Coll et al. “A robust SNP barcode for typing Mycobacterium tuberculosis complex strains”. In: *Nature Communications* 5.1 (2014), p. 4812. DOI: 10.1038/ncomms5812.

[114] David Stucki et al. “Standard Genotyping Overestimates Transmission of Mycobacterium tuberculosis among Immigrants in a Low-Incidence Country”. In: *Journal of Clinical Microbiology* 54.7 (2016), pp. 1862–1870. ISSN: 0095-1137. DOI: 10.1128/jcm.00126-16.

[115] Samuel Lipworth et al. “Ahead of Print - SNP-IT Tool for Identifying Subspecies and Associated Lineages of Mycobacterium tuberculosis Complex - Volume 25, Number 3—March 2019 - Emerging Infectious Diseases journal - CDC”. In: *Emerging Infectious Diseases* 25.3 (2019), pp. 482–488. ISSN: 1080-6040. DOI: 10.3201/eid2503.180894.

[116] Luca Freschi et al. “Population structure, biogeography and transmissibility of Mycobacterium tuberculosis”. In: *bioRxiv* (2020), p. 2020.09.29.293274. DOI: 10.1101/2020.09.29.293274.

[117] Timothy M Walker et al. “Whole-genome sequencing to delineate Mycobacterium tuberculosis outbreaks: a retrospective observational study”. In: *The Lancet Infectious Diseases* 13.2 (2013), pp. 137–146. ISSN: 1473-3099. DOI: 10.1016/s1473-3099(12)70277-3.

[118] David H. Wyllie et al. “A Quantitative Evaluation of MIRU-VNTR Typing Against Whole-Genome Sequencing for Identifying Mycobacterium tuberculosis Transmission: A Prospective Observational Cohort Study”. In: *EBioMedicine* 34 (2018), pp. 122–130. ISSN: 2352-3964. DOI: 10.1016/j.ebiom.2018.07.019.

[119] Jennifer L. Gardy et al. “Whole-Genome Sequencing and Social-Network Analysis of a Tuberculosis Outbreak”. In: *The New England Journal of Medicine* 364.8 (2011), pp. 730–739. ISSN: 0028-4793. DOI: 10.1056/nejmoa1003176.

[120] Timothy M Walker et al. “Assessment of Mycobacterium tuberculosis transmission in Oxfordshire, UK, 2007–12, with whole pathogen genome sequences: an observational study”. In: *The Lancet Respiratory Medicine* 2.4 (2014), pp. 285–292. ISSN: 2213-2600. DOI: 10.1016/s2213-2600(14)70027-x.

[121] Hollie-Ann Hatherell et al. “Interpreting whole genome sequencing for investigating tuberculosis transmission: a systematic review”. In: *BMC Medicine* 14.1 (2016), p. 21. DOI: 10.1186/s12916-016-0566-x.

[122] Rana Jajou et al. “Epidemiological links between tuberculosis cases identified twice as efficiently by whole genome sequencing than conventional molecular typing: A population-based study”. In: *PLOS ONE* 13.4 (2018). DOI: 10.1371/journal.pone.0195413.

[123] James Stimson et al. “Beyond the SNP Threshold: Identifying Outbreak Clusters Using Inferred Transmissions”. In: *Molecular Biology and Evolution* 36.3 (2019), pp. 587–603. ISSN: 0737-4038. DOI: 10.1093/molbev/msy242.

[124] Katharine S Walter et al. “Genomic variant-identification methods may alter Mycobacterium tuberculosis transmission inferences”. In: *Microbial Genomics* 6.8 (2020). DOI: 10.1099/mgen.0.000418.

[125] Suha Kadura et al. “Systematic review of mutations associated with resistance to the new and repurposed Mycobacterium tuberculosis drugs bedaquiline, clofazimine, linezolid, delamanid and pretomanid”. In: *Journal of Antimicrobial Chemotherapy* 75.8 (2020), pp. 2031–2043. ISSN: 0305-7453. DOI: 10.1093/jac/dkaa136.

[126] Louise J Pankhurst et al. “Rapid, comprehensive, and affordable mycobacterial diagnosis with whole-genome sequencing: a prospective study”. In: *The Lancet Respiratory Medicine* 4.1 (2016), pp. 49–58. ISSN: 2213-2600. DOI: 10.1016/s2213-2600(15)00466-x.

[127] Catharina C Boehme et al. “Feasibility, diagnostic accuracy, and effectiveness of decentralised use of the Xpert MTB/RIF test for diagnosis of tuberculosis and multidrug resistance: a multicentre implementation study”. In: *The Lancet* 377.9776 (2011), pp. 1495–1505. ISSN: 0140-6736. DOI: 10.1016/s0140-6736(11)60438-8.

[128] Yuan Cao et al. “Xpert MTB/XDR: a 10-Color Reflex Assay Suitable for Point-of-Care Settings To Detect Isoniazid, Fluoroquinolone, and Second-Line-Injectable-Drug Resistance Directly from Mycobacterium tuberculosis-Positive Sputum”. In: *Journal of Clinical Microbiology* 59.3 (2021). ISSN: 0095-1137. DOI: 10.1128/jcm.02314-20.

[129] Arnold Bainomugisa et al. “New Xpert MTB/XDR: added value and future in the field”. In: *European Respiratory Journal* 56.5 (2020), p. 2003616. ISSN: 0903-1936. DOI: 10.1183/13993003.03616-2020.

[130] Elisabeth Sanchez-Padilla et al. “Detection of Drug-Resistant Tuberculosis by Xpert MTB/RIF in Swaziland”. In: *The New England Journal of Medicine* 372.12 (2015), pp. 1181–1182. ISSN: 0028-4793. DOI: 10.1056/nejmc1413930.

[131] Antonina A. Votintseva et al. “Mycobacterial DNA Extraction for Whole-Genome Sequencing from Early Positive Liquid (MGIT) Cultures”. In: *Journal of Clinical Microbiology* 53.4 (2015), pp. 1137–1143. ISSN: 0095-1137. DOI: 10.1128/jcm.03073-14.

[132] Timothy M Walker et al. "Whole-genome sequencing for prediction of Mycobacterium tuberculosis drug susceptibility and resistance: a retrospective cohort study". In: *The Lancet Infectious Diseases* 15.10 (2015), pp. 1193–1202. ISSN: 1473-3099. DOI: 10.1016/s1473-3099(15)00062-6.

[133] Project, The CRyPTIC Consortium and the 100000 Genomes. "Prediction of Susceptibility to First-Line Tuberculosis Drugs by DNA Sequencing". In: *New England Journal of Medicine* 379.15 (2018), pp. 1403–1415. ISSN: 0028-4793. DOI: 10.1056/nejmoa1800474.

[134] Paolo Miotto et al. "A standardised method for interpreting the association between mutations and phenotypic drug resistance in Mycobacterium tuberculosis". In: *European Respiratory Journal* 50.6 (2017), p. 1701354. ISSN: 0903-1936. DOI: 10.1183/13993003.01354-2017.

[135] Francesc Coll et al. "Rapid determination of anti-tuberculosis drug resistance from whole-genome sequences". In: *Genome Medicine* 7.1 (2015), p. 51. DOI: 10.1186/s13073-015-0164-0.

[136] Jody E. Phelan et al. "Integrating informatics tools and portable sequencing technology for rapid detection of resistance to anti-tuberculous drugs". In: *Genome Medicine* 11.1 (2019), p. 41. DOI: 10.1186/s13073-019-0650-x.

[137] Maha R. Farhat et al. "GWAS for quantitative resistance phenotypes in Mycobacterium tuberculosis reveals resistance genes and regulatory regions". In: *Nature Communications* 10.1 (2019), p. 2128. DOI: 10.1038/s41467-019-10110-6.

[138] Alice Brankin et al. "A data compendium of Mycobacterium tuberculosis antibiotic resistance". In: *bioRxiv* (2021), p. 2021.09.14.460274. DOI: 10.1101/2021.09.14.460274.

[139] The CRyPTIC Consortium, Sarah G Earle, and Daniel J Wilson. "Genome-wide association studies of global Mycobacterium tuberculosis resistance to thirteen antimicrobials in 10,228 genomes". In: *bioRxiv* (2021), p. 2021.09.14.460272. DOI: 10.1101/2021.09.14.460272.

[140] The CRyPTIC Consortium and Joshua J Carter. "Quantitative measurement of antibiotic resistance in Mycobacterium tuberculosis reveals genetic determinants of resistance and susceptibility in a target gene approach". In: *bioRxiv* (2021), p. 2021.09.14.460353. DOI: 10.1101/2021.09.14.460353.

[141] *Catalogue of mutations in Mycobacterium tuberculosis complex and their association with drug resistance*. Tech. rep. Geneva: World Health Organization, 2021.

[142] Erwan Drezen et al. "GATB: Genome Assembly & Analysis Tool Box". In: *Bioinformatics* 30.20 (2014), pp. 2959–2961. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btu406.

[143] Kazutaka Katoh and Martin C. Frith. "Adding unaligned sequences into an existing alignment using MAFFT and LAST". In: *Bioinformatics* 28.23 (2012), pp. 3144–3146. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bts578.

[144] Steve Davis et al. "CFSAN SNP Pipeline: an automated method for constructing SNP matrices from next-generation sequence data". In: *PeerJ Computer Science* 1 (2015), e20. ISSN: 2376-5992. DOI: 10.7717/peerj-cs.20. URL: https://doi.org/10.7717/peerj-cs.20.

[145] Chen Yang et al. "NanoSim: nanopore sequence read simulator based on statistical characterization". In: *GigaScience* 6.4 (2017).

[146] Karel Břinda et al. *karel-brinda/NanoSim-H: NanoSim-H 1.1.0.4*. 2018. DOI: 10.5281/zenodo.1341250. URL: https://doi.org/10.5281/zenodo.1341250.

[147] Michael B. Hall. *Rasusa: Randomly subsample sequencing reads to a specified coverage*. Nov. 2019. DOI: 10.5281/zenodo.3731394.

[148] Harry A Thorpe et al. "Piggy: A Rapid, Large-Scale Pan-Genome Analysis Tool for Intergenic Regions in Bacteria." In: *GigaScience* 7.4 (2018), giy015–. ISSN: 2047-217X. DOI: 10.1093/gigascience/giy015.

[149] Nicholas J Loman and Mick Watson. "Successful test launch for nanopore sequencing". In: *Nature Methods* 12.4 (2015), pp. 303–304. ISSN: 1548-7091. DOI: 10.1038/nmeth.3327.

[150] Claire L Gorrie et al. "Key parameters for genomics-based real-time detection and tracking of multidrug-resistant bacteria: a systematic analysis". In: *The Lancet Microbe* (2021). ISSN: 2666-5247. DOI: 10.1016/s2666-5247(21)00149-x.

[151] Daniel S. Standage, C. Titus Brown, and Fereydoun Hormozdiari. "Kevlar: A Mapping-Free Framework for Accurate Discovery of De Novo Variants". In: *iScience* 18 (2019), pp. 28–36. ISSN: 2589-0042. DOI: 10.1016/j.isci.2019.07.032.

[152] Stephen J Bush. "Generalizable characteristics of false-positive bacterial variant calls". In: *Microbial Genomics* 7.8 (2021). DOI: 10.1099/mgen.0.000615.

[153] Michael Reynolds et al. *Tuberculosis in England: 2020 report*. GW-1672. London, UK: Public Health England, Nov. 2020.

[154] Ellen Brooks-Pollock et al. "A model of tuberculosis clustering in low incidence countries reveals more transmission in the United Kingdom than the Netherlands between 2010 and 2015". In: *PLOS Computational Biology* 16.3 (2020), e1007687. ISSN: 1553-734X. DOI: 10.1371/journal.pcbi.1007687.

[155] Luke W Meredith et al. "Rapid implementation of SARS-CoV-2 sequencing to investigate cases of health-care associated COVID-19: a prospective genomic surveillance study". In: *The Lancet Infectious Diseases* 20.11 (2020), pp. 1263–1272. ISSN: 1473-3099. DOI: 10.1016/s1473-3099(20)30562-4.

[156] Christopher M. Watson et al. "Long-read nanopore sequencing resolves a TMEM231 gene conversion event causing Meckel–Gruber syndrome". In: *Human Mutation* 41.2 (2020), pp. 525–531. ISSN: 1059-7794. DOI: 10.1002/humu.23940.

[157] Aaron M. Wenger et al. "Accurate circular consensus long-read sequencing improves variant detection and assembly of a human genome". In: *Nature Biotechnology* 37.10 (2019), pp. 1155–1162. ISSN: 1087-0156. DOI: 10.1038/s41587-019-0217-9.

[158] Shilpa Garg et al. "Chromosome-scale, haplotype-resolved assembly of human genomes". In: *Nature Biotechnology* 39.3 (2021), pp. 309–312. ISSN: 1087-0156. DOI: 10.1038/s41587-020-0711-0.

[159] Rick E. Masonbrink et al. "A chromosomal assembly of the soybean cyst nematode genome". In: *Molecular Ecology Resources* (2021). ISSN: 1755-098X. DOI: 10.1111/1755-0998.13432.

[160] Nicola De Maio et al. "Comparison of long-read sequencing technologies in the hybrid assembly of complex bacterial genomes". In: *Microbial Genomics* 5.9 (2019). DOI: 10.1099/mgen.0.000294.

[161] Sergey Koren et al. "Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation". In: *Genome Research* 27.5 (2017), pp. 722–736. ISSN: 1088-9051. DOI: 10.1101/gr.215087.116.

[162] Sergey Nurk et al. "HiCanu: accurate assembly of segmental duplications, satellites, and allelic variants from high-fidelity long reads". In: *Genome Research* 30.9 (2020), gr.263566.120. ISSN: 1088-9051. DOI: 10.1101/gr.263566.120.

[163] Mikhail Kolmogorov et al. “Assembly of long, error-prone reads using repeat graphs”. In: *Nature Biotechnology* 37.5 (2019), pp. 540–546. ISSN: 1087-0156. DOI: 10.1038/s41587-019-0072-8.

[164] Ryan R. Wick et al. “Unicycler: Resolving bacterial genome assemblies from short and long sequencing reads”. In: *PLOS Computational Biology* 13.6 (2017), e1005595. ISSN: 1553-734X. DOI: 10.1371/journal.pcbi.1005595.

[165] Ehsan Haghshenas et al. “HASLR: Fast Hybrid Assembly of Long Reads”. In: *iScience* 23.8 (2020), p. 101389. ISSN: 2589-0042. DOI: 10.1016/j.isci.2020.101389.

[166] Dmitry Antipov et al. “hybrid SPA des : an algorithm for hybrid assembly of short and long reads”. In: *Bioinformatics* 32.7 (2016), pp. 1009–1015. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btv688.

[167] Egor Shitikov et al. “Evolutionary pathway analysis and unified classification of East Asian lineage of Mycobacterium tuberculosis”. In: *Scientific Reports* 7.1 (2017), p. 9227. DOI: 10.1038/s41598-017-10018-5.

[168] Liliana K. Rutaihwa et al. “Multiple Introductions of Mycobacterium tuberculosis Lineage 2–Beijing Into Africa Over Centuries”. In: *Frontiers in Ecology and Evolution* 7 (2019), p. 112. DOI: 10.3389/fevo.2019.00112.

[169] David Stucki et al. “Mycobacterium tuberculosis lineage 4 comprises globally distributed and geographically restricted sublineages”. In: *Nature Genetics* 48.12 (2016), pp. 1535–1543. ISSN: 1061-4036. DOI: 10.1038/ng.3704.

[170] Álvaro Chiner-Oms et al. “Genome-wide mutational biases fuel transcriptional diversity in the Mycobacterium tuberculosis complex”. In: *Nature Communications* 10.1 (2019), p. 3994. DOI: 10.1038/s41467-019-11948-6.

[171] Tsukasa Nakamura et al. “Parallelization of MAFFT for large-scale multiple sequence alignments”. In: *Bioinformatics* 34.14 (2018), pp. 2490–2492. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bty121.

[172] Kazutaka Katoh and Daron M. Standley. “A simple method to control over-alignment in the MAFFT multiple sequence alignment program”. In: *Bioinformatics* 32.13 (2016), pp. 1933–1942. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btw108.

[173] LUSHENG WANG and TAO JIANG. “On the Complexity of Multiple Sequence Alignment”. In: *Journal of Computational Biology* 1.4 (1994), pp. 337–348. ISSN: 1066-5277. DOI: 10.1089/cmb.1994.1.337.

[174] Rana Jajou et al. “Towards standardisation: comparison of five whole genome sequencing (WGS) analysis pipelines for detection of epidemiologically linked tuberculosis cases”. In: *Eurosurveillance* 24.50 (2019). URL: https://www.eurosurveillance.org/content/10.2807/1560-7917.ES.2019.24.50.1900130.

[175] Martin Hunt et al. “Minos: variant adjudication and joint genotyping of cohorts of bacterial genomes”. In: *bioRxiv* (2021), p. 2021.09.15.460475. DOI: 10.1101/2021.09.15.460475.

[176] Yan Guo et al. “The effect of strand bias in Illumina short-read sequencing data”. In: *BMC genomics* 13.1 (2012), pp. 1–11.

[177] Torsten Seemann. *snp-dists: Pairwise SNP distance matrix from a FASTA sequence alignment*. Version 0.7.0. Sept. 2018. DOI: 10.5281/zenodo.1411986. URL: https://doi.org/10.5281/zenodo.1411986.

[178] Martin A. Fischler and Robert C. Bolles. “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography”. In: *Commun. ACM* 24.6 (June 1981), pp. 381–395. ISSN: 0001-0782. DOI: 10.1145/358669.358692. URL: https://doi.org/10.1145/358669.358692.

[179] Fabian Pedregosa et al. “Scikit-Learn: Machine Learning in Python”. In: *J. Mach. Learn. Res.* 12.null (Nov. 2011), pp. 2825–2830. ISSN: 1532-4435.

[180] Amos Tversky. “Features of similarity”. In: *Psychological Review* 84.4 (1977), pp. 327–352. ISSN: 0033-295X. DOI: 10.1037/0033-295x.84.4.327.

[181] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. “Exploring Network Structure, Dynamics, and Function using NetworkX”. In: *Proceedings of the 7th Python in Science Conference*. Ed. by Gaël Varoquaux, Travis Vaught, and Jarrod Millman. Pasadena, CA USA, 2008, pp. 11–15.

[182] Jacob Pritt, Nae-Chyun Chen, and Ben Langmead. “FORGe: prioritizing variants for graph genomes”. In: *Genome Biology* 19.1 (2018), p. 220. DOI: 10.1186/s13059-018-1595-x.

[183] Conor J. Meehan et al. “The relationship between transmission time and clustering methods in Mycobacterium tuberculosis epidemiology”. In: *EBioMedicine* 37 (2018), pp. 410–416. ISSN: 2352-3964. DOI: 10.1016/j.ebiom.2018.10.013.

[184] Andreas Roetzer et al. “Whole Genome Sequencing versus Traditional Genotyping for Investigation of a Mycobacterium tuberculosis Outbreak: A Longitudinal Molecular Epidemiological Study”. In: *PLoS Medicine* 10.2 (2013), e1001387. DOI: 10.1371/journal.pmed.1001387.

[185] Marina Meilă. “Comparing clusterings—an information based distance”. In: *Journal of Multivariate Analysis* 98.5 (2007), pp. 873–895. ISSN: 0047-259X. DOI: 10.1016/j.jmva.2006.11.013.

[186] Yifei Xu et al. “Nanopore metagenomic sequencing to investigate nosocomial transmission of human metapneumovirus from a unique genetic group among haematology patients in the United Kingdom”. In: *Journal of Infection* 80.5 (2020), pp. 571–577. ISSN: 0163-4453. DOI: 10.1016/j.jinf.2020.02.003.

[187] Nasir Riaz et al. “Adaptation of Oxford Nanopore technology for hepatitis C whole genome sequencing and identification of within-host viral variants”. In: *BMC Genomics* 22.1 (2021), p. 148. DOI: 10.1186/s12864-021-07460-1.

[188] Jun Li et al. “Rapid genomic characterization of SARS-CoV-2 viruses from clinical specimens using nanopore sequencing”. In: *Scientific Reports* 10.1 (2020), p. 17492. DOI: 10.1038/s41598-020-74656-y.

[189] Anna L. McNaughton et al. “Illumina and Nanopore methods for whole genome sequencing of hepatitis B virus (HBV)”. In: *Scientific Reports* 9.1 (2019), p. 7081. DOI: 10.1038/s41598-019-43524-9.

[190] Heng Li. “Improving SNP discovery by base alignment quality”. In: *Bioinformatics* 27.8 (2011), pp. 1157–1158. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btr076.

[191] Maximillian Marin et al. “Genomic sequence characteristics and the empiric accuracy of short-read sequencing”. In: *bioRxiv* (2021), p. 2021.04.08.438862. DOI: 10.1101/2021.04.08.438862.

[192] Felix Mölder et al. “Sustainable data analysis with Snakemake”. In: *F1000Research* 10 (2021), p. 33. DOI: 10.12688/f1000research.29032.2.

[193] John D. Hunter. “Matplotlib: A 2D Graphics Environment”. In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. ISSN: 1521-9615. DOI: 10.1109/mcse.2007.55.

[194] Michael Waskom. “seaborn: statistical data visualization”. In: *Journal of Open Source Software* 6.60 (2021), p. 3021. DOI: 10.21105/joss.03021.

[195] Bokeh Development Team. *Bokeh: Python library for interactive visualization*. 2014. URL: http://www.bokeh.pydata.org.

[196] Seth C Inzaule et al. "Genomic-informed pathogen surveillance in Africa: opportunities and challenges". In: *The Lancet Infectious Diseases* (2021). ISSN: 1473-3099. DOI: 10.1016/s1473-3099(20)30939-7.

[197] *The use of next-generation sequencing technologies for the detection of mutations associated with drug resistance in Mycobacterium tuberculosis complex: technical guide*. Tech. rep. WHO/CDS/TB/2018.19. Geneva: World Health Organization, 2018.

[198] Nontuthuko E. Maningi et al. "Comparison of line probe assay to BACTEC MGIT 960 system for susceptibility testing of first and second-line anti-tuberculosis drugs in a referral laboratory in South Africa". In: *BMC Infectious Diseases* 17.1 (2017), p. 795. DOI: 10.1186/s12879-017-2898-3.

[199] Shashikant Srivastava et al. "emb nucleotide polymorphisms and the role of embB306 mutations in Mycobacterium tuberculosis resistance to ethambutol". In: *International Journal of Medical Microbiology* 299.4 (2009), pp. 269–280. ISSN: 1438-4221. DOI: 10.1016/j.ijmm.2008.07.001.

[200] Florence Brossier et al. "Molecular Analysis of the embCAB Locus and embR Gene Involved in Ethambutol Resistance in Clinical Isolates of Mycobacterium tuberculosis in France". In: *Antimicrobial Agents and Chemotherapy* 59.8 (2015), pp. 4800–4808. ISSN: 0066-4804. DOI: 10.1128/aac.00150-15.

[201] Peter Krusche et al. "Best practices for benchmarking germline small-variant calls in human genomes". In: *Nature Biotechnology* 37.5 (2019), pp. 555–560. ISSN: 1087-0156. DOI: 10.1038/s41587-019-0054-x.

[202] Tomasz Jagielski et al. "Characterization of Mutations Conferring Resistance to Rifampin in Mycobacterium tuberculosis Clinical Strains". In: *Antimicrobial Agents and Chemotherapy* 62.10 (2018), e01093–18. ISSN: 0066-4804. DOI: 10.1128/aac.01093-18.

[203] Thomas Andreas Kohl et al. "MTBseq: a comprehensive pipeline for whole genome sequence analysis of Mycobacterium tuberculosis complex isolates". In: *PeerJ* 6 (2018), e5895. ISSN: 2167-8359. DOI: 10.7717/peerj.5895.

[204] Z Zhang et al. "Ethambutol Resistance as Determined by Broth Dilution Method Correlates Better than Sequencing Results with embB Mutations in Multidrug-Resistant Mycobacterium tuberculosis Isolates". In: *Journal of Clinical Microbiology* 52.2 (2014), pp. 638–641. ISSN: 0095-1137. DOI: 10.1128/jcm.02713-13.

[205] Frederick A. Sirgel et al. "embB306 Mutations as Molecular Indicators to Predict Ethambutol Susceptibility in Mycobacterium tuberculosis". In: *Chemotherapy* 58.5 (2013), pp. 358–363. ISSN: 0009-3157. DOI: 10.1159/000343474.

[206] World Health Organization et al. *Technical report on critical concentrations for drug susceptibility testing of medicines used in the treatment of drug-resistant tuberculosis*. Tech. rep. World Health Organization, 2018.

[207] Hiroki Ando et al. "A silent mutation in mabA confers isoniazid resistance on Mycobacterium tuberculosis". In: *Molecular Microbiology* 91.3 (2014), pp. 538–547. ISSN: 1365-2958. DOI: 10.1111/mmi.12476.

[208] Nat Smittipat et al. "Mutations in rrs, rpsL and gidB in streptomycin-resistant Mycobacterium tuberculosis isolates from Thailand". In: *Journal of Global Antimicrobial Resistance* 4 (2016), pp. 5–10. ISSN: 2213-7165. DOI: 10.1016/j.jgar.2015.11.009.

[209] Henrietta Venter et al. "Intrinsic, adaptive and acquired antimicrobial resistance in Gram-negative bacteria". In: *Essays in Biochemistry* 61.1 (Mar. 2017), pp. 49–59. ISSN: 0071-1365. DOI: 10.1042/EBC20160063. eprint: https://portlandpress.com/essaysbiochem/article-pdf/61/1/49/482596/ebc-2016-0063c.pdf. URL: https://doi.org/10.1042/EBC20160063.

[210] Vladimír Boža et al. “DeepNano-blitz: a fast base caller for MinION nanopore sequencers”. In: *Bioinformatics* 36.14 (2020), pp. 4191–4192. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btaa297.

[211] Simona Luca and Traian Mihaescu. “History of BCG Vaccine.” In: *Mædica* 8.1 (2013), pp. 53–8. ISSN: 1841-9038.

[212] Kerri M. Malone et al. “Updated Reference Genome Sequence and Annotation of Mycobacterium bovis AF2122/97”. In: *Genome Announcements* 5.14 (2017), e00157–17. DOI: 10.1128/genomea.00157-17.

[213] Carly Kanipe and Mitchell V. Palmer. “Mycobacterium bovis and you: A comprehensive look at the bacteria, its similarities to Mycobacterium tuberculosis, and its relationship with human disease”. In: *Tuberculosis* 125 (2020), p. 102006. ISSN: 1472-9792. DOI: 10.1016/j.tube.2020.102006.

[214] Sophie George et al. “DNA Thermo-Protection Facilitates Whole-Genome Sequencing of Mycobacteria Direct from Clinical Samples”. In: *Journal of Clinical Microbiology* 58.10 (2020). ISSN: 0095-1137. DOI: 10.1128/jcm.00670-20.

[215] Tyson A Clark et al. “Enhanced 5-methylcytosine detection in single-molecule, real-time sequencing via Tet1 oxidation”. In: *BMC Biology* 11.1 (2013), pp. 4–4. DOI: 10.1186/1741-7007-11-4.

[216] Lawal Danjuma et al. “Genomic plasticity between human and mycobacterial DNA: A review”. In: *Tuberculosis* 107 (2017), pp. 38–47. ISSN: 1472-9792. DOI: 10.1016/j.tube.2017.03.006.

[217] F. P. Breitwieser, D. N. Baker, and S. L. Salzberg. “KrakenUniq: confident and fast metagenomics classification using unique k-mer counts”. In: *Genome Biology* 19.1 (2018), p. 198. DOI: 10.1186/s13059-018-1568-0.

[218] Jody E. Phelan et al. “Recombination in pe/ppe genes contributes to genetic variation in Mycobacterium tuberculosis lineages”. In: *BMC Genomics* 17.1 (2016), p. 151. DOI: 10.1186/s12864-016-2467-y.

[219] Miten Jain et al. “Nanopore sequencing and assembly of a human genome with ultra-long reads”. In: *Nature Biotechnology* 36.4 (2018), pp. 338–345. ISSN: 1087-0156. DOI: 10.1038/nbt.4060.

[220] Samuel J Modlin et al. “Drivers and sites of diversity in the DNA adenine methylomes of 93 Mycobacterium tuberculosis complex clinical isolates”. In: *eLife* 9 (2020). DOI: 10.7554/elife.58542.

[221] John Lafferty, Andrew McCallum, and Fernando CN Pereira. “Conditional random fields: Probabilistic models for segmenting and labeling sequence data”. In: *In International Conference on Machine Learning*. 2001.

[222] J D van Embden et al. “Strain identification of Mycobacterium tuberculosis by DNA fingerprinting: recommendations for a standardized methodology”. In: *Journal of Clinical Microbiology* 31.2 (1993), pp. 406–409. DOI: 10.1128/jcm.31.2.406-409.1993. eprint: https://journals.asm.org/doi/pdf/10.1128/jcm.31.2.406-409.1993. URL: https://journals.asm.org/doi/abs/10.1128/jcm.31.2.406-409.1993.

[223] Robin Warren et al. “Safe Mycobacterium tuberculosis DNA extraction method that does not compromise integrity”. In: *Journal of clinical microbiology* 44.1 (Jan. 2006), pp. 254–256. ISSN: 0095-1137. DOI: 10.1128/jcm.44.1.254-256.2006. URL: https://europepmc.org/articles/PMC1351970.

[224] Anthony M. Bolger, Marc Lohse, and Bjoern Usadel. “Trimmomatic: a flexible trimmer for Illumina sequence data”. In: *Bioinformatics* 30.15 (2014), pp. 2114–2120. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btu170.

[225] Daehwan Kim et al. “Centrifuge: rapid and sensitive classification of metagenomic sequences”. In: *Genome Research* 26.12 (2016), pp. 1721–1729. ISSN: 1088-9051. DOI: 10.1101/gr.210641.116.

[226] Robert Vaser et al. “Fast and accurate de novo genome assembly from long uncorrected reads”. In: *Genome Research* 27.5 (2017), pp. 737–746. ISSN: 1088-9051. DOI: 10.1101/gr.214270.116.

[227] Bruce J. Walker et al. “Pilon: An Integrated Tool for Comprehensive Microbial Variant Detection and Genome Assembly Improvement”. In: *PLoS ONE* 9.11 (2014), e112963. DOI: 10.1371/journal.pone.0112963.

[228] Torsten Seemann. “Prokka: rapid prokaryotic genome annotation”. In: *Bioinformatics* 30.14 (2014), pp. 2068–2069. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btu153.

[229] Scott C. Clark et al. “ALE: a generic assembly likelihood evaluation framework for assessing the accuracy of genome and metagenome assemblies”. In: *Bioinformatics* 29.4 (2013), pp. 435–443. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bts723.

[230] Alexey Gurevich et al. “QUAST: quality assessment tool for genome assemblies”. In: *Bioinformatics* 29.8 (2013), pp. 1072–1075. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btt086.

[231] Midori Kato-Maeda et al. “Comparing Genomes within the Species Mycobacterium tuberculosis”. In: *Genome Research* 11.4 (2001), pp. 547–554. ISSN: 1088-9051. DOI: 10.1101/gr166401.

[232] Srinand Sreevatsan et al. “Restricted structural gene polymorphism in the Mycobacterium tuberculosis complex indicates evolutionarily recent global dissemination”. In: *Proceedings of the National Academy of Sciences* 94.18 (1997), pp. 9869–9874. ISSN: 0027-8424. DOI: 10.1073/pnas.94.18.9869.

[233] Federico Giannoni et al. “Evaluation of a New Line Probe Assay for Rapid Identification of gyrA Mutations in Mycobacterium tuberculosis”. In: *Antimicrobial Agents and Chemotherapy* 49.7 (2005), pp. 2928–2933. ISSN: 0066-4804. DOI: 10.1128/aac.49.7.2928-2933.2005.

[234] Ritu Singhal et al. “Sequence Analysis of Fluoroquinolone Resistance-Associated Genes gyrA and gyrB in Clinical Mycobacterium tuberculosis Isolates from Patients Suspected of Having Multidrug-Resistant Tuberculosis in New Delhi, India”. In: *Journal of Clinical Microbiology* 54.9 (2016), pp. 2298–2305. ISSN: 0095-1137. DOI: 10.1128/jcm.00670-16.

[235] Leigh J Manley, Duanduan Ma, and Stuart S Levine. “Monitoring Error Rates In Illumina Sequencing”. In: *Journal of Biomolecular Techniques : JBT* 27.4 (2016), pp. 125–128. ISSN: 1524-0215. DOI: 10.7171/jbt.16-2704-002.

Appendix A

Supporting work for Chapter 2

A.1 Length of missing loci in simulations

Figure A.1 shows the distribution of lengths for the missing loci in Section 2.3. The most commonly missed locus was `GC00000093_18`, which has a length of 146bp.

A.2 Constructing a pan-genome SNP truth set

The work in this section was performed by Leandro Ishi and is described here to aid the reader's understanding of the evaluation in Section 2.4. It is also described in detail in [10].

The purpose of a truth set of pan-genome SNPs is to describe the variation that exists *between* a collection of samples, rather than just between a single sample and a reference genome. Thus, for example, in the traditional model where a single reference genome is used to call variants for a collection of samples, if a gene that occurs in some of the samples does not occur in the reference genome used, there would be no recall penalty (see Figure 1.3).

We first identify all pairwise differences between the 20 samples in this study (Section 2.4.1) with `dnadiff` [40] and `minimap2` [41]. We join these two sets of SNP differences and create a probe of each SNP with 50bp of flanking sequence from one of the genomes in the pairwise comparison. This probe is then mapped to the other genome, and if it maps to multiple locations or the allele does not perfectly match the sequence it aligns to, we remove it, ensuring our truth set is unambiguous.

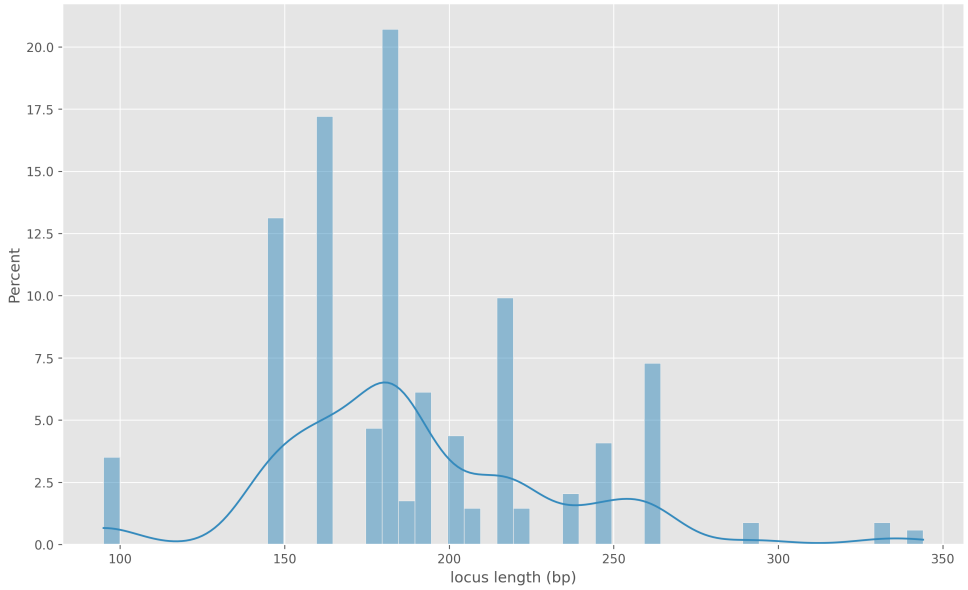


Fig. A.1: The lengths (x-axis; base pairs) of missing loci in the *de novo* simulations in Section 2.3

These filtered pairwise variants (PwV) are then grouped into pan-genome variants (PgV). For instance, a variant between two genomes could be the same as a variant that exists between two different genomes, making it the same variant in the context of a pan-genome.

Each allele is uniquely defined as $A = (G_A, C_A, P_A)$, where G_A, C_A and P_A are the genome, chromosome, and position that the allele occurs in, respectively. Thus, a pan-genome variant is formally described as an allele, A, between two genomes - $PwV = \{A_1, A_2\}$. Lastly, we define a pan-genome variant $PgV = \{PwV_1, PwV_2, ..., PwV_n\}$. That is, if two pairwise variants PwV_1 and PwV_2 share an allele A_i, they are in the same pan-genome variant, PgV. Given the removal of multi-mapping probes, we also know that a PwV can only occur in one PgV.

The truth set of pan-genome variants, P, is constructed using an undirected graph $G = (V, E)$, where V is a set of nodes (pairwise variants (PwV)), and E is a set of edges connecting two nodes if they share an allele. Therefore, a pan-genome variant, PgV, is a connected component (subgraph) $C \in G$, such that there exists a path between any two nodes (PwV) in C.

Finally, the pan-genome variants (connected components, C; PgV) are filtered such that a PgV has no more than one allele (A) originating from the same genome, G_A, and can only be biallelic. Thus, in the end, each pan-genome variant describes the same variant in different genomes, letting go of a coordinate system; we find 618,305 such variants between the 20 samples.

There are two measures of recall within such a pan-genome variant set: average allelic recall (AvgAR) and pan-variant recall (PVR). AvgAR is an average of the recall for all *PgV*s. That is, it describes what proportion of *all* alleles in a *PgV* are correctly discovered, averaged over all *PgV*s. In contrast, PVR is the proportion of *PgV*s where at least one allele was found. For example, we have two *PgV*s, V_1, which contains two alleles present in 5 genomes each, and V_2, which has two alleles present in one genome each. If 8/10 alleles in V_1 and 0/2 alleles in V_2 are discovered, the AvgAR would be $\frac{(8/10)+(0/2)}{2} = 0.4$ and PVR would be $\frac{1+0}{2} = 0.5$.

Appendix B

Supporting work for Chapter 3

B.1 DNA sequencing methods

B.1.1 Isolate selection and DNA extraction

Madagascar

The Institut Pasteur de Madagascar hosts the Madagascar Programme National de Lutte Contre la Tuberculose (National TB Program), which provides reference DST for TB patients suspected of facing infection relapse or treatment failure. Culture-confirmed multi-drug resistant (MDR) isolates were included. All MDR isolates were matched with culture-confirmed drug-susceptible isolates referred during the same period and from the same region of the country. All available isolates were included for patients undergoing TB treatment and for which a follow-up culture was positive.

DNA from *M. tuberculosis* culture was extracted using the cetyltrimethylammonium bromide (CTAB) method previously described by Van Embden with minor modification [222]. Briefly, samples were inactivated by heating at 80°C for 30 minutes. Next, 10mg/ml of lysozyme was added to the suspension and incubated for at least one hour at 37°C. After adding proteinase K 10mg/ml and sodium dodecyl sulphate (SDS) 10%, the suspension was incubated for 10 minutes at 65°C. A mixture of CTAB and 5M NaCl preheated to 65°C was then added. The suspension was mixed until a milky mixture was obtained and incubated for 10 minutes at 65°C. Next, chloroform isoamyl alcohol (24:1) mixture was added, followed by centrifugation for 5 minutes at 10,000rpm and 4°C. The upper phase was recovered, and DNA was precipitated by adding isopropanol to the solution. The mixture was frozen for at least one hour then centrifuged at 10000rpm and 4°C, for 5 minutes. The supernatant was

discarded, and the pellet was washed with 70% ethanol and centrifuged at 10,000rpm and 4°C, for 5 minutes. The DNA pellet was dried with speed-vac for 2 minutes and resuspended in 1X TE. DNA was quantified using the Qubit dsDNA HS Assay Kit (Thermo Fisher Scientific, USA).

South Africa

Clinical *M. tuberculosis* isolates routinely collected in the Western Cape Province of South Africa, processed by the National Health Laboratory Service (NHLS) and diagnosed as rifampicin-resistant tuberculosis, are biobanked at the South African Medical Research Council Centre for Tuberculosis Research (SAMRC-CTR) housed at the Division of Molecular Biology and Human Genetics at Stellenbosch University, South Africa. A convenience data set of 82 clinical Mtb isolates for which Illumina WGS data was available were selected for Nanopore sequencing.

Clinical *M. tuberculosis* isolates were cultured on supplemented 7H10 solid media under BSL3 conditions. Phenol-chloroform DNA extraction was performed on heat-inactivated cultured isolates as previously described [223].

B.1.2 PacBio sequencing

Thirty-five of the Malagasy samples were sequenced and processed at the Next Generation Genomics Core within Cold Spring Harbor Laboratory. Samples were quantified with a Qubit dsDNA HS Assay Kit and QC'd through a Pulsed Field Gel Electrophoresis system. Then, samples were sheared at 10kb using a Megaruptor device and size-selected to 8-10 kb with a Blue pippin instrument - followed by 0.45X ampure bead purification. The PacBio library protocol SMRTbell Express Template Prep Kit 2.0 was used for each sample. Briefly, the first step was the removal of single-stranded overhangs followed by DNA Damage Repair, End-Repair/A-tailing, Ligation of overhang barcoded adaptors and sample pooling. A total of 3 pools were produced: LID50532 (16 samples), LID50533 (10 samples), and LID50534 (9 samples). After pooling, 0.5X ampure bead clean up was performed. A Sequel I instrument was used to sequence the 3 library pools. Libraries were annealed for an hour and bounded for an hour using sequel binding kit 3.0. Bound SMRTbell complexes were then purified with ampure beads. The run was set up as 10kb length for 10 hours movie time. The Sequel 1M V2 SMRT cells were used for each library.

The circular consensus was called via the SMRTlink graphical user interface version 6.0.0.47841.

B.2 Benchmark of long read genome assembly methods for *M. tuberculosis*

We chose samples with greater than 30x coverage across all three sequencing technologies to produce high-quality assemblies. In total, this left 9 Malagasy samples. There have been many new genome assembly methods produced since the last known assessment of *M. tuberculosis* long-read assemblies [87]. As such, we compare five assemblers and select the one that produces the most consistently good results. The reason for this comparison is that different assembly algorithms can produce quite varied results depending on sequencing technology used, species, or computational resource availability [89, 160].

As outlined in Section 3.3, the assembly tools we assess are Canu, Flye, Unicycler, HASLR, and Spades. `HASLR` and `unicycler` are hybrid assemblers that take Illumina reads along with one long-read file. `spades` is also a hybrid assembler but takes an arbitrary number of different sequencing technologies. `canu` and `flye` are both long-read-only assemblers.

The first step in the assembly pipeline is trimming adapter sequences in the Illumina reads using Trimmomatic (v0.39; [224]). Two assemblies were produced for each sample - one for each long-read technology. The exception to this was `spades`, for which there is just one assembly for each sample, as it accepts all reads simultaneously.

`canu`, in some cases, produces assembly bubbles, which are regions where it believes there is a heterozygous locus due to differences in haplotypes. While some samples could be multi-clonal, we chose to remove bubbles from the `canu` assemblies, effectively choosing the dominant haplotype for downstream analysis.

All contigs in the resulting assemblies were species-classified using Centrifuge (v1.0.4beta; [225]). We remove any contigs whose classification places them outside of the Mycobacterium Tuberculosis Complex. Polishing these decontaminated assemblies is done in two steps; first using long-reads and Racon (v1.4.13; [226]) with default settings, followed by short reads with Pilon (v1.23; [227]). Default Pilon settings were used for Nanopore assemblies, but for PacBio we do not correct for SNPs. PacBio CCS reads are already a consensus from multiple reads, so allowing Illumina reads to fix at a per-base level lead to decreased per-base accuracy (results not shown).

We annotate both the polished and unpolished genome assemblies using Prokka (v1.14.5; [228]). We assess the relative correctness of all assembly variations for a given sample using Assembly Likelihood Estimator (ALE; [229]). Assembly statistics

were generated for each sample using Quast [230] with the *M. tuberculosis* reference genome, H37Rv (accession: NC_000962.3), as a reference. We do not expect our assemblies to be the same as H37Rv, but it can provide insights into the structural completeness and genome size. Lastly, we assess per-base accuracy using a custom script. As input for the script, we provide a BAM file of the Illumina reads mapped to the assembly and the pileup generated by the Samtools subroutine `mpileup` [30]. We require a quorum of 90% and a minimum read depth of 10x. Quorum is the percentage of reads that must agree with the assembly at each position. Any position in the assembly that does not meet both of these conditions is considered a disagreement. The output from the script is a collection of statistics and a BED file containing all disagreement positions.

Assessing the quality of *de novo* assemblies is non-trivial and requires aggregation of various metrics. Whilst a reference genome exists for *M. tuberculosis* (H37Rv), there are enough differences between the *M. tuberculosis* lineages that using this reference would not be appropriate for our purposes. Instead, we will look at each assessment metric individually and then decide on the best assembly method from this information.

B.2.1 Assembly Likelihood Evaluation score

The assembly likelihood evaluation (ALE) score is a reference-free metric that describes the likelihood of an assembly given its *k*-mer distribution and the likelihood of the reads being generated from that assembly. It combines information such as read quality, agreement in the alignment of reads to assembly, mate-pair orientation, paired-end read length, and depth of sequencing. Notably, the ALE score can be used to compare assemblies of the same genome - precisely the use case we have. Whilst ALE scores are not insightful on their own, the difference *between* assembly scores gives the relative probability of correctness. So for each sample, we are interested in the assembly that has the *highest* ALE score.

Across the 9 samples, `flye` has the highest ALE score in five cases, `spades` was the most probable assembly in two, with `unicycler` and `canu` having the maximum score for one sample each (Figure B.1). Note, this is considering both polished and unpolished assemblies together for each tool. When considering which long-read sequencing technology produces the greatest ALE score, PacBio had the top score in 7/9 samples. In 6/9 samples, the highest ALE score was from a polished assembly.

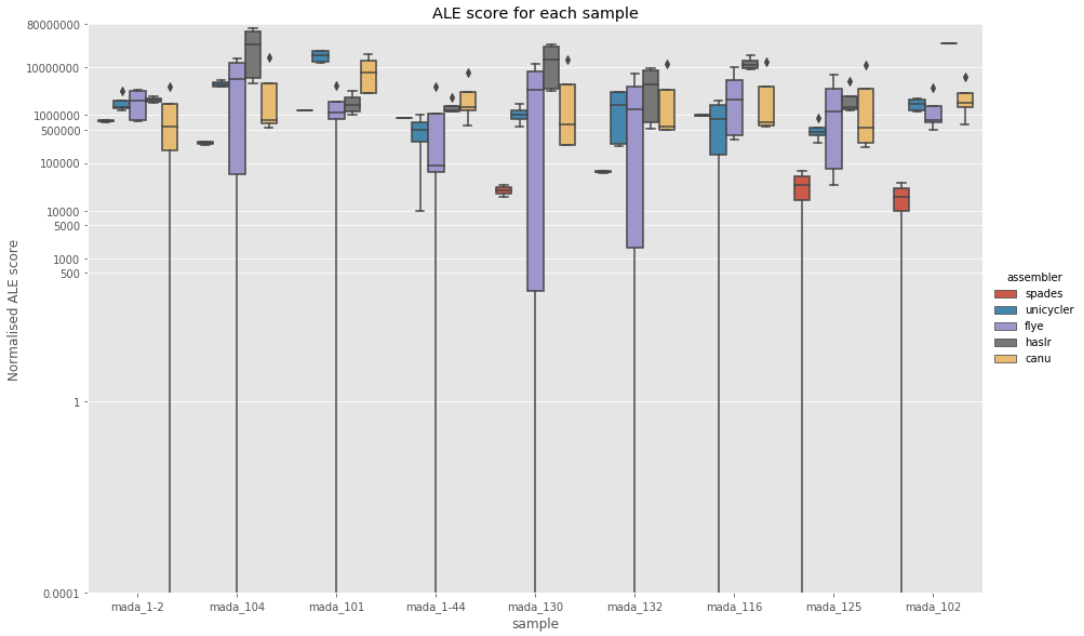


Fig. B.1: The normalised assembly likelihood evaluation (ALE) score (Y-axis) for each sample (X-axis), coloured by assembly tools. ALE score is a metric describing the likelihood of an assembly. The normalisation is done by subtracting the assembly's score from the maximum (best) score for that sample, giving a relative probability of correctness. Each box represents different technologies and polishing status for each assembler.

B.2.2 Disagreement rate

The disagreement rate is an approximation of the per-base accuracy of the assembly. We map Illumina reads to the assembly with `bwa mem` and calculate what proportion of positions do 90% of the reads agree with the assembly nucleotide.

In 7 of 9 samples, a `HASLR` assembly had the lowest disagreement rate, followed by `unicycler` having the minimum in 2. Polished genomes produced the lowest disagreement rate in 8/9 samples, and Nanopore-based assemblies had the best accuracy in 6/9 samples. While it is not so surprising that assemblies polished with Illumina reads have a lower disagreement rate, it is unexpected that Nanopore would produce more accurate assemblies (Figure B.2). One caveat to keep in mind here is that there is an element of overfitting to this metric; we assess using Illumina, and so naturally, assemblies polished with Illumina produce better results. That is not to say this statistic is void, but that it should be used cautiously and not in isolation.

As part of calculating the disagreement rate, we also produce a BED file listing all positions with less than 90% agreement. This BED file can be used as a genome mask when using the assembly for later analysis.

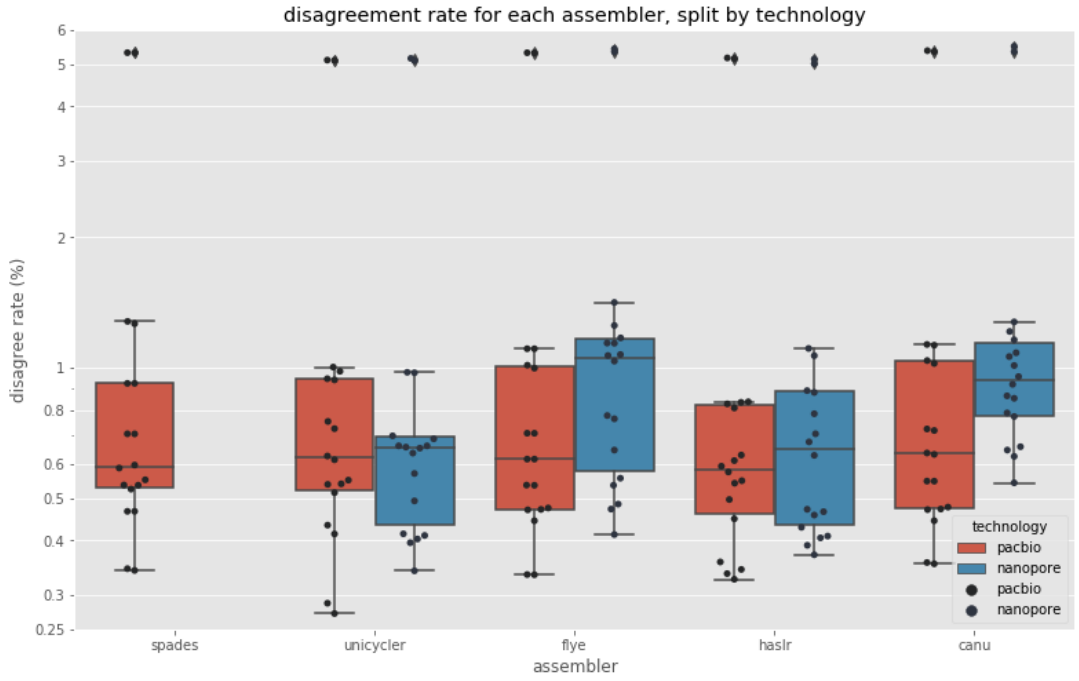


Fig. B.2: The disagreement rate (y-axis) for each assembler (x-axis), coloured by the sequencing technology. Disagreement rate is the percentage of sites in the assembly where Illumina reads do not have at least 90% quorum. Each box/point represents different samples and polished status for the relevant assembler/technology combination.

B.2.3 Number of contigs

As *M. tuberculosis* has only a single circular chromosome, for an assembly to be structurally complete, there should only be a single contig in the final assembly. However, it is not always appropriate for an assembly method to produce a single contig as data quality, depth of sequencing, read length, and repetitive content of the genome can hamper this goal. Conversely, receiving a single contig as output is no guarantee of its quality for similar reasons to the multi-contig scenario. For this benchmark, considered in conjunction with the other metrics, the number of contigs can be a helpful datum for selecting our favoured assembly. If an assembly has a single contig and scores well on other metrics, we would be more inclined to choose it over another assembly with similar metrics but many more contigs.

Across all combinations of assembly conditions, `spades` (8/18), `flye` (18/32) and `canu` (16/32) produced far more single-contig assemblies than the hybrid methods (Figure B.3). `unicycler` produced no single-contig genomes, whilst `HASLR` yielded only 2/32. When considering sequencing technology, PacBio (26/72) had many more single-contig assemblies than Nanopore (10/72). Note, `spades` assemblies use all three technologies and are not considered in the technology single-contig totals.

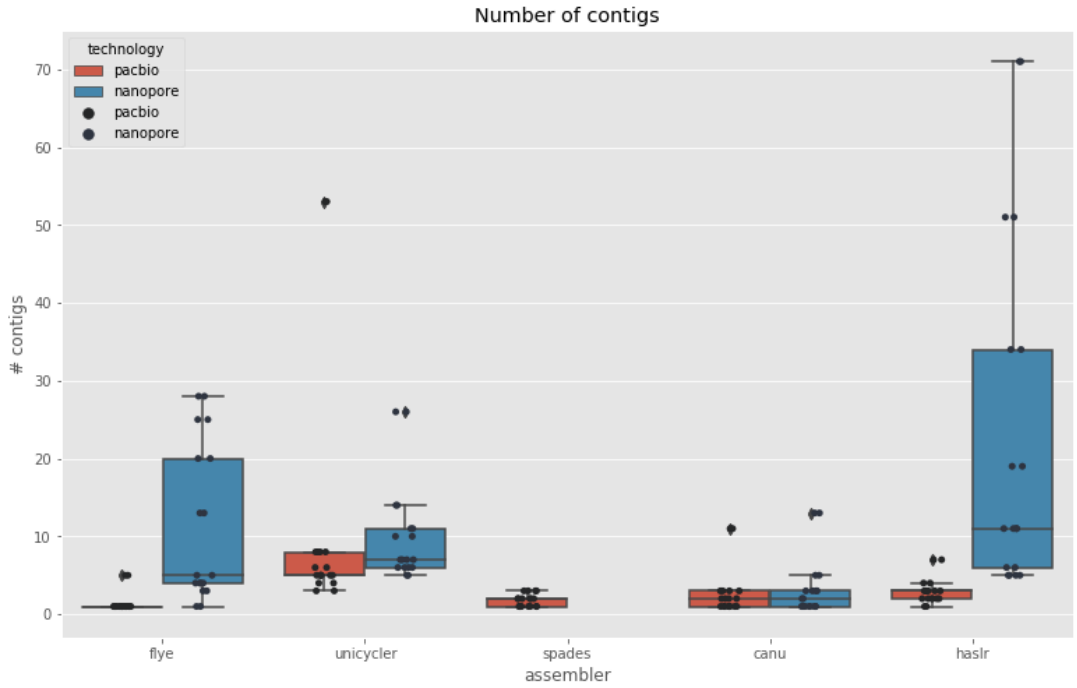


Fig. B.3: The number of contigs (Y-axis) produced from each assembly (X-axis), coloured by sequencing technology. Each box/point represents different samples and polished status for the relevant assembler-technology combination.

B.2.4 Length of assembly

As mentioned earlier, comparing assemblies to the *M. tuberculosis* reference genome (H37Rv) is not appropriate due to lineage differences. However, assembly length/size can be used as an aid for selection. The size of any lineage's genome is not expected to differ from the reference by more than ~30 kilobases [231]. Considering the genome size in addition to the disagreement rate is particularly informative. It would be easy for an assembly method to produce very accurate per-base contigs by refusing to produce sequences for "harder" parts of the genome. While such an assembly would score well on the disagreement rate, it would not do so well when considering how close it is to the expected genome size. The length of the assembly also clearly shows when a method is outputting *too much* sequence.

When comparing each assembly to H37Rv, we found a fairly even spread across assemblers for the closest size to H37Rv. For 3/9 samples, `canu` has the smallest size difference, followed by `spades` (2/9), `unicycler` (2/9), `HASLR` (1/9) and `flye` (1/9). An honourable mention should be made of `flye` and `spades` as they had much lower variation in size compared to the other approaches. In terms of sequencing

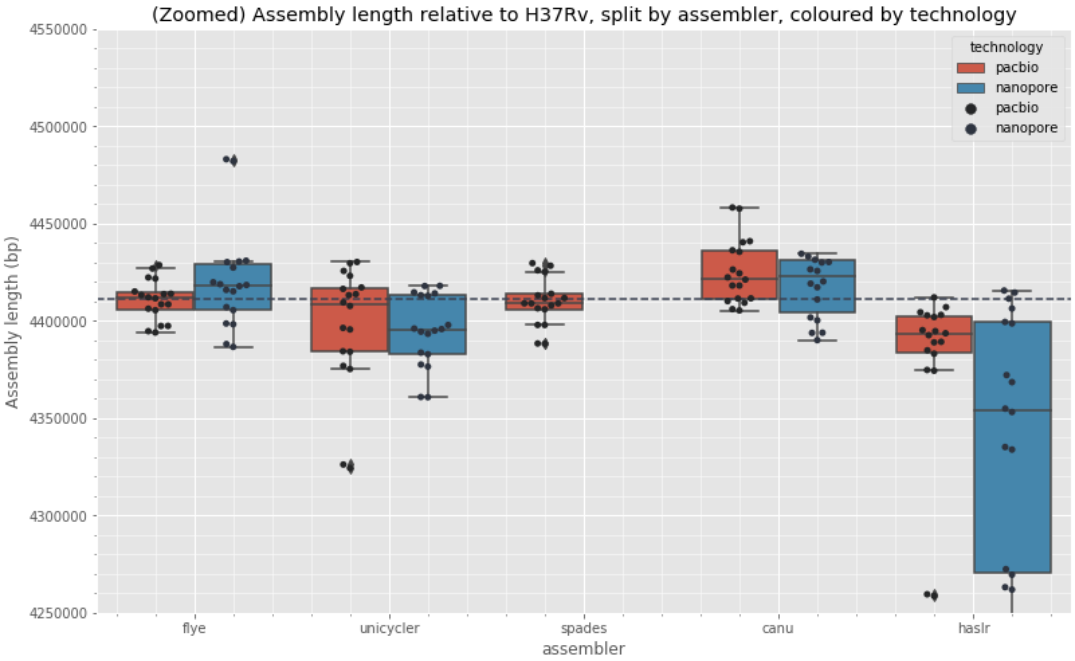


Fig. B.4: Size/length (Y-axis; in base-pairs (bp)) of each assembly (X-axis), coloured by each sequencing technology. The horizontal dashed line represents the size of the *M. tuberculosis* reference genome (4,411,532bp). Each box/point represents different samples and polished status for the relevant assembler-technology combination. Note: the Y-axis has been limited to allow for greater resolution of similarity to the H37Rv size

technology, in 6/9 samples, PacBio produced the genome size closest to H37Rv. Polished assemblies had the closer size in 5/9 samples.

B.2.5 Contamination detection

The decontamination step in the assembly pipeline revealed that one sample, `mada_1-2`, contained contigs from three different species: *Mycobacterium intracellulare*, *Dermacoccus nishinomiyaensis*, and *M. tuberculosis*. These contigs were all at sufficient coverage to not be considered background noise. For the assembly assessment analysis, only the contigs from *M. tuberculosis* were used. However, this sample is an outlier in almost all the assessment metrics in the previous sections. Given this profuse contamination, `mada_1-2` will not be used in any analysis where these assemblies are used for truth validation purposes.

B.2.6 Summary

In conclusion, considering all assessment metrics, `flye` and `spades` assemblies were consistently the better performing methods across all of the criteria outlined in this

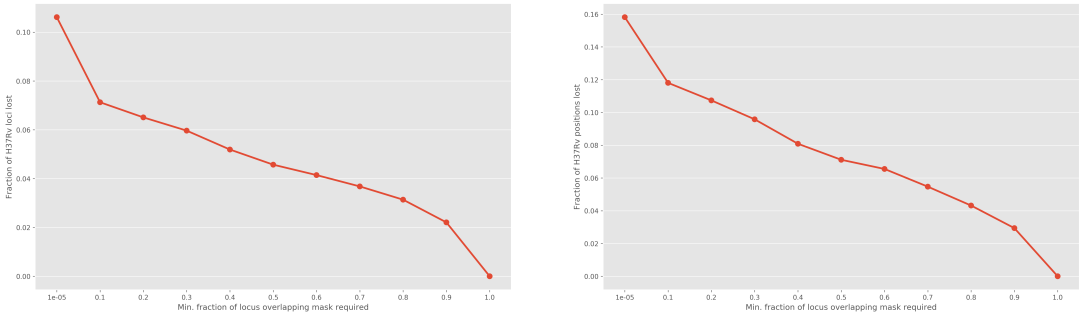
section. As most of the validation analyses that these assemblies will be used for involve comparing Illumina and Nanopore data to a "neutral truth", the unpolished PacBio CCS assemblies from `flye` were selected for use. The differences between the polished and unpolished CCS assemblies were negligible. Therefore, they did not outweigh the benefit of having a single-technology PacBio assembly that can be used as an unbiased reference point for comparing the other two technologies.

B.3 Masking of the *M. tuberculosis* reference graph

We investigated three different strategies for masking the positions that go into the *M. tuberculosis* reference graph in Section 3.5. The first was not to use a mask and apply all variants that passed all other filters in the CRyPTIC VCF. Compared to the final method we chose, this lead to the sparse PRG MSA step having a peak memory usage of 357GB (1.7 times more than Table 3.1) and a wall clock time of 13.5 hours (compared to 1.25). The dense PRG MSA likewise saw higher memory usage (370GB) and wall clock time (44.6 hours). In addition, when updating these PRGs to include novel variants (Section 3.6.4), the MSA stage took on the order of days to complete.

The second masking strategy was to remove the VCF positions that occur within the H37Rv genome mask mentioned in Section 3.5. This approach would ensure there would be sequence covering the whole H37Rv genome within the reference graph. This approach yields construction times similar to those in Table 3.1. However, this caused the novel variant discovery stage of `pandora` to hit the seven-day compute node run limit for some samples. The cause of this huge increase can be explained by the fact that the positions within the mask are, by definition, repetitive (low complexity). As such, when `pandora` initiates *de novo* variant discovery in these sections of the genome, the path enumeration step outlined in Section 2.2.2 gets caught in cycles within the de Bruijn graph - a limitation outlined in Section 2.6.2.

The third strategy is the one we ended up using - removing loci with $\geq 30\%$ of their positions in the mask. The 30% figure was settled on as it provided a good balance between losing sections of the H37Rv genome (see Figure B.5) and providing acceptable computational performance for all steps in the reference graph construction and `pandora` variant-calling pipeline.



(a) **(b)**

Fig. B.5: a) Proportion of loci lost (y-axis) when removing those with a certain fraction (x-axis) of their positions within the genome mask. **b)** Proportion of total genome positions lost (y-axis) when removing loci with a certain fraction (x-axis) of their positions within the genome mask.

B.4 Precision and recall of all `pandora` variant calls

Although we only use SNPs for identifying transmission clusters (Section 3.6.4), we also assessed `pandora` variant calls for all variants - SNPs and indels up to a length of 20bp. The precision and recall using the same filters as in Section 3.6.4 are shown in Figure B.6. We did this for the sake of future work that might be interested in using `pandora` indel calls, such as predicting drug resistance. Again, the sparse PRG gave better precision and recall than the dense one. The use of all variants leads to a median recall improvement to 75.48% (all filters) - up 3.49% from SNPs only. On the other hand, precision sees a drop to 95.90% (median) when assessing all variants - compared to 100% for SNPs only. Given the large drop in precision, it is clear that `pandora` indel-calling needs further improvement. However, indel-calling (deletions especially) is a known limitation of Nanopore [78, 219] and others have seen precision drops when adding indels to the evaluation [84].

B.5 Selecting Nanopore SNP thresholds to define transmission clusters

In Section 3.7, we fit linear models that describe the relationship between the Illumina and Nanopore SNP distance between pairs of samples. Using the equations for each model, we can infer a Nanopore threshold (y) corresponding to a given Illumina threshold (x). Next, we investigate how well these model-based thresholds perform by using the cluster similarity metrics defined in Section 3.8.1. To assess the performance,

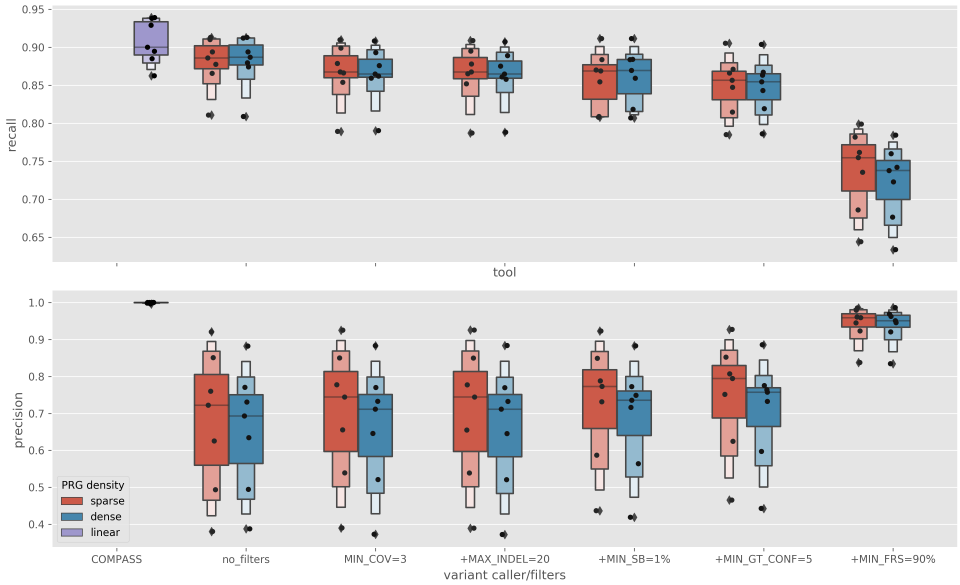


Fig. B.6: Precision (bottom) and recall (top) of SNPs for COMPASS (purple) and all variants (maximum indel length of 20bp) for `pandora` with sparse (red) and dense (blue) PRGs. The `pandora` boxes start with no filters on the left, with each box moving to the right adding a filter to the previous box. The COMPASS box is a reference to the precision and recall of Illumina variant calls. Linear PRG density refers to the fact that COMPASS uses a single, linear reference genome as opposed to `pandora`, which uses a genome graph. The black points refer to single data points for the seven samples used. MIN_COV=minimum depth of coverage; MIN_SB=minimum strand bias; MIN_GT_CONF=minimum genotype confidence score; MIN_FRS=minimum fraction of read support.

we look at the SACR, SACP, and XCR for all threshold values surrounding the model-based threshold and see whether the model-based threshold performs best.

B.5.1 BCFtools

For the Illumina SNP thresholds 0, 2, 5, and 12, the corresponding `bcftools` model-inferred thresholds are 1, 3, 5, and 10 (red vertical dashed lines in Figure B.7). Based on the threshold sweep in Figure B.7, the best SNP distance thresholds to use for `bcftools` are deemed 0, 2, 5, and 11 as these provide the best balance between the three similarity metrics.

B.5.2 `pandora` single-sample

For the Illumina SNP thresholds 0, 2, 5, and 12, the corresponding `pandora` single-sample model-inferred thresholds are 15, 17, 19, and 24 (red vertical dashed lines in Figure B.8). Based on the threshold sweep in Figure B.8, the best SNP distance thresholds to use for `pandora` single-sample are deemed 16, 18, 18, and 27 as these provide the best balance between the three similarity metrics.

B.5.3 `pandora` multi-sample

For the Illumina SNP thresholds 0, 2, 5, and 12, the corresponding `pandora` multi-sample model-inferred thresholds are 0, 0, 1, and 4 (red vertical dashed lines in Figure B.9). Based on the threshold sweep in Figure B.9, the best SNP distance thresholds to use for `pandora` multi-sample are deemed 0, 1, 3, and 7 as these provide the best balance between the three similarity metrics.

In summary, the model-based thresholds chosen for the Nanopore variant callers are not the best-performing. Therefore, for all further transmission cluster work, the hand-picked thresholds are used instead.

B.6 Full mixed technology simulation results

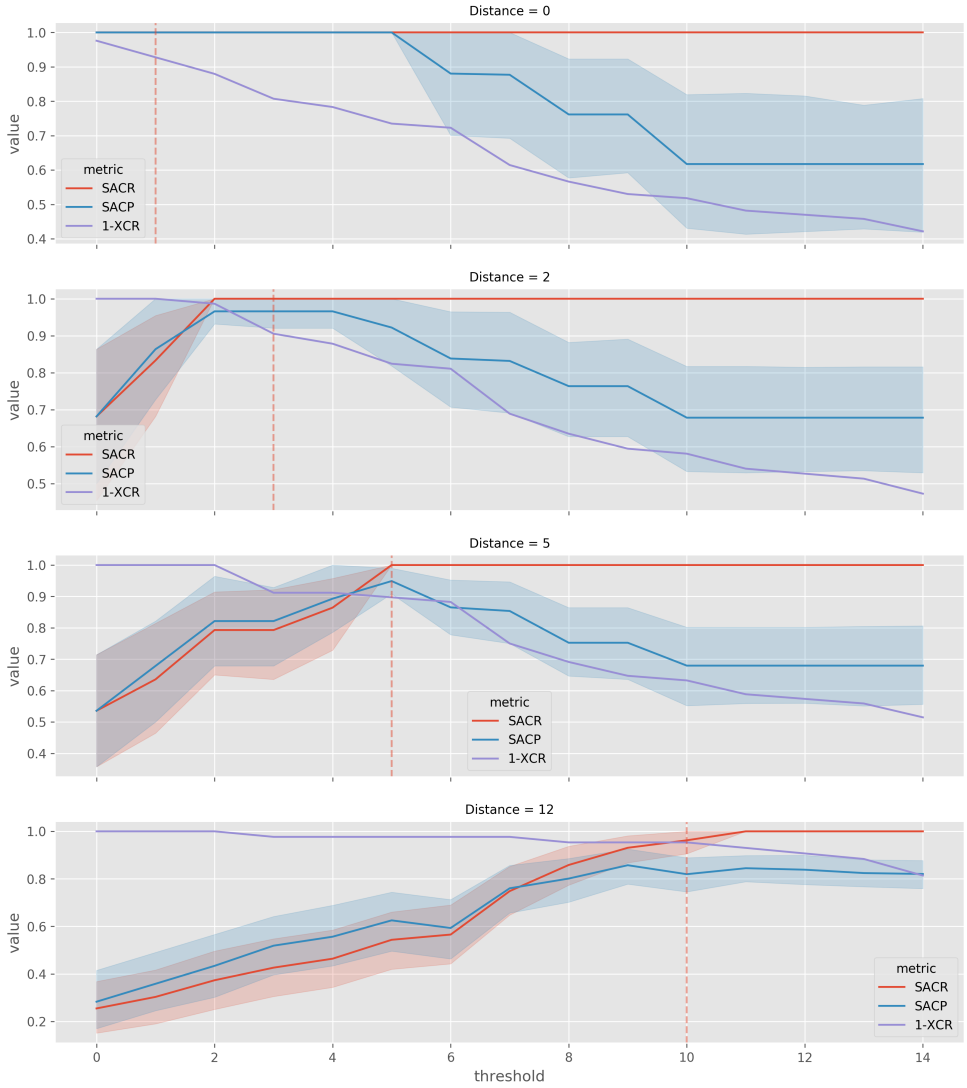


Fig. B.7: Illumina and Nanopore (`bcftools`) transmission cluster similarity for various SNP distance threshold. Each subplot compares the Nanopore clustering for the threshold on the x-axis to the Illumina clustering based on the distance (threshold) in the subplot title. SACR (red), SACP (blue), and 1−XCR are represented by the lines with the band around the line indicating the 95% confidence interval. The red, vertical, dashed lines indicate the model-based prediction of what the Nanopore SNP distance threshold should be based on the Illumina distance for that subplot.

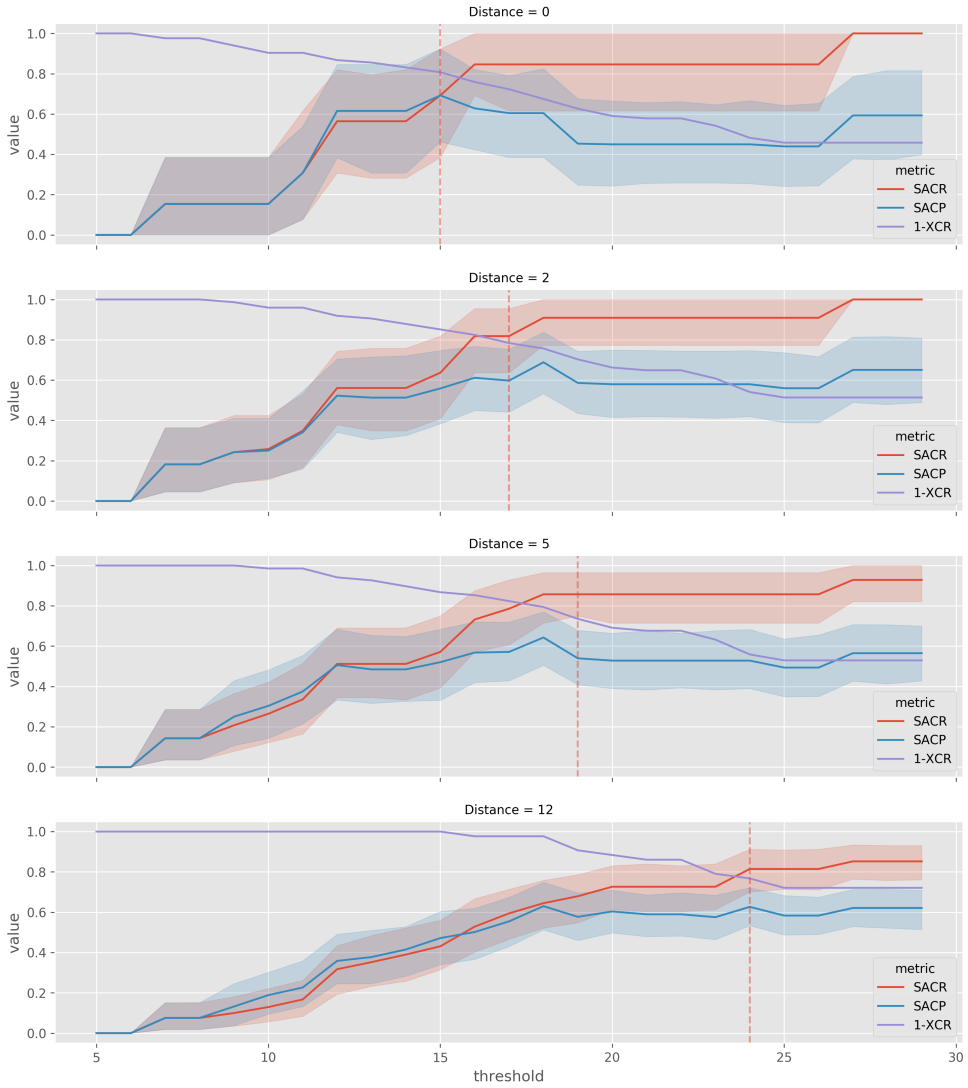


Fig. B.8: Illumina and Nanopore (`pandora` single-sample) transmission cluster similarity for various SNP distance threshold. Each subplot compares the Nanopore clustering for the threshold on the x-axis to the Illumina clustering based on the distance (threshold) in the subplot title. SACR (red), SACP (blue), and $1-\text{XCR}$ are represented by the lines with the band around the line indicating the 95% confidence interval. The red, vertical, dashed lines indicate the model-based prediction of what the Nanopore SNP distance threshold should be based on the Illumina distance for that subplot.

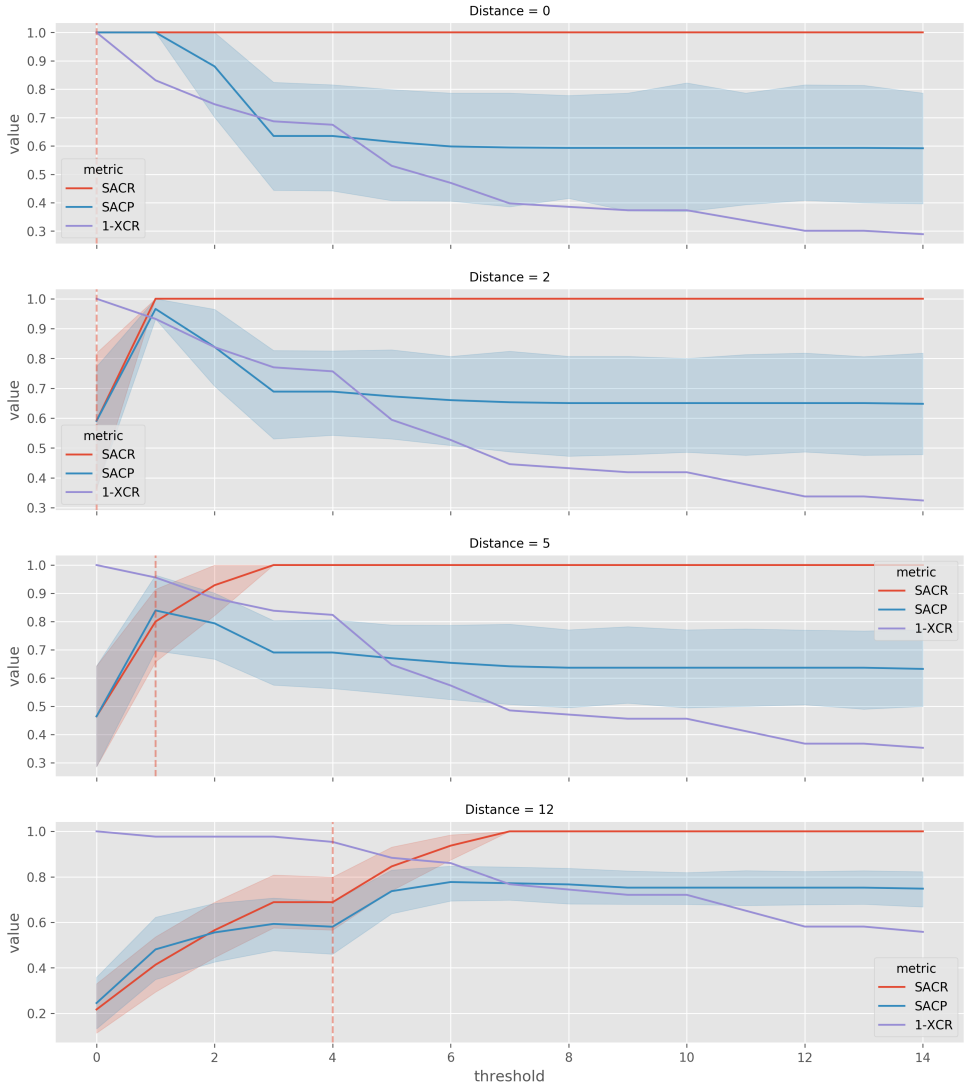


Fig. B.9: Illumina and Nanopore (`pandora` multi-sample) transmission cluster similarity for various SNP distance threshold. Each subplot compares the Nanopore clustering for the threshold on the x-axis to the Illumina clustering based on the distance (threshold) in the subplot title. SACR (red), SACP (blue), and 1−XCR are represented by the lines with the band around the line indicating the 95% confidence interval. The red, vertical, dashed lines indicate the model-based prediction of what the Nanopore SNP distance threshold should be based on the Illumina distance for that subplot.

Ratio	Threshold	Metric	Mean	std	Min.	25%	50%	75%	Max.
0.01	0	1-XCR	1.000	0.001	0.985	1.000	1.000	1.000	1.000
		SACP	0.998	0.017	0.846	1.000	1.000	1.000	1.000
		SACR	0.997	0.020	0.846	1.000	1.000	1.000	1.000
	2	1-XCR	1.000	0.000	0.992	1.000	1.000	1.000	1.000
		SACP	1.000	0.002	0.966	1.000	1.000	1.000	1.000
		SACR	1.000	0.000	1.000	1.000	1.000	1.000	1.000
	5	1-XCR	1.000	0.002	0.984	1.000	1.000	1.000	1.000
		SACP	0.999	0.007	0.929	1.000	1.000	1.000	1.000
		SACR	0.999	0.008	0.929	1.000	1.000	1.000	1.000
	12	1-XCR	1.000	0.002	0.990	1.000	1.000	1.000	1.000
		SACP	0.998	0.010	0.930	1.000	1.000	1.000	1.000
		SACR	1.000	0.002	0.965	1.000	1.000	1.000	1.000
0.05	0	1-XCR	0.999	0.003	0.985	1.000	1.000	1.000	1.000
		SACP	0.990	0.035	0.769	1.000	1.000	1.000	1.000
		SACR	0.988	0.041	0.718	1.000	1.000	1.000	1.000
	2	1-XCR	1.000	0.002	0.992	1.000	1.000	1.000	1.000
		SACP	0.998	0.008	0.966	1.000	1.000	1.000	1.000
		SACR	1.000	0.000	1.000	1.000	1.000	1.000	1.000
	5	1-XCR	0.997	0.006	0.967	1.000	1.000	1.000	1.000
		SACP	0.993	0.019	0.878	1.000	1.000	1.000	1.000
		SACR	0.993	0.020	0.871	1.000	1.000	1.000	1.000
	12	1-XCR	0.998	0.005	0.979	1.000	1.000	1.000	1.000
		SACP	0.989	0.024	0.874	0.987	1.000	1.000	1.000
		SACR	0.998	0.007	0.965	1.000	1.000	1.000	1.000
0.10	0	1-XCR	0.999	0.004	0.985	1.000	1.000	1.000	1.000
		SACP	0.980	0.049	0.769	1.000	1.000	1.000	1.000
		SACR	0.976	0.056	0.718	1.000	1.000	1.000	1.000
	2	1-XCR	0.999	0.002	0.992	1.000	1.000	1.000	1.000
		SACP	0.997	0.010	0.966	1.000	1.000	1.000	1.000
		SACR	1.000	0.000	1.000	1.000	1.000	1.000	1.000
	5	1-XCR	0.994	0.009	0.959	0.984	1.000	1.000	1.000

Table B.1 continued from previous page

Ratio	Threshold	Metric	Mean	std	Min.	25%	50%	75%	Max.
		SACP	0.987	0.025	0.878	0.982	1.000	1.000	1.000
		SACR	0.989	0.025	0.871	1.000	1.000	1.000	1.000
		1-XCR	0.995	0.006	0.969	0.990	1.000	1.000	1.000
	12	SACP	0.975	0.033	0.855	0.943	0.987	1.000	1.000
		SACR	0.997	0.010	0.965	1.000	1.000	1.000	1.000
		1-XCR	0.996	0.006	0.985	1.000	1.000	1.000	1.000
	0	SACP	0.958	0.067	0.769	0.923	1.000	1.000	1.000
		SACR	0.948	0.078	0.718	0.872	1.000	1.000	1.000
		1-XCR	0.998	0.003	0.992	0.992	1.000	1.000	1.000
	2	SACP	0.991	0.015	0.966	0.966	1.000	1.000	1.000
0.25		SACR	1.000	0.000	1.000	1.000	1.000	1.000	1.000
		1-XCR	0.985	0.013	0.943	0.975	0.984	1.000	1.000
	5	SACP	0.972	0.035	0.878	0.949	0.982	1.000	1.000
		SACR	0.979	0.035	0.871	0.943	1.000	1.000	1.000
		1-XCR	0.988	0.008	0.969	0.979	0.990	0.990	1.000
	12	SACP	0.946	0.042	0.842	0.915	0.943	0.984	1.000
		SACR	0.994	0.013	0.965	1.000	1.000	1.000	1.000
		1-XCR	0.993	0.007	0.985	0.985	1.000	1.000	1.000
	0	SACP	0.941	0.076	0.769	0.846	1.000	1.000	1.000
		SACR	0.929	0.089	0.718	0.846	1.000	1.000	1.000
		1-XCR	0.996	0.004	0.992	0.992	1.000	1.000	1.000
	2	SACP	0.983	0.017	0.966	0.966	1.000	1.000	1.000
0.50		SACR	1.000	0.000	1.000	1.000	1.000	1.000	1.000
		1-XCR	0.971	0.015	0.943	0.959	0.967	0.984	1.000
	5	SACP	0.952	0.037	0.878	0.929	0.949	0.976	1.000
		SACR	0.974	0.037	0.871	0.929	1.000	1.000	1.000
		1-XCR	0.981	0.008	0.969	0.979	0.979	0.990	1.000
	12	SACP	0.906	0.043	0.842	0.861	0.911	0.930	1.000
		SACR	0.991	0.015	0.965	0.965	1.000	1.000	1.000
		1-XCR	0.989	0.006	0.985	0.985	0.985	1.000	1.000
	0	SACP	0.955	0.069	0.769	0.923	1.000	1.000	1.000

0.75

Table B.1 continued from previous page

Ratio	Threshold	Metric	Mean	std	Min.	25%	50%	75%	Max.
		SACR	0.945	0.080	0.718	0.872	1.000	1.000	1.000
		1-XCR	0.994	0.003	0.992	0.992	0.992	0.992	1.000
	2	SACP	0.974	0.015	0.966	0.966	0.966	0.966	1.000
		SACR	1.000	0.000	1.000	1.000	1.000	1.000	1.000
		1-XCR	0.958	0.014	0.943	0.943	0.959	0.967	1.000
	5	SACP	0.945	0.032	0.878	0.949	0.949	0.970	1.000
		SACR	0.984	0.030	0.871	1.000	1.000	1.000	1.000
		1-XCR	0.974	0.006	0.969	0.969	0.969	0.979	0.990
	12	SACP	0.875	0.035	0.842	0.845	0.857	0.902	0.975
		SACR	0.994	0.013	0.965	1.000	1.000	1.000	1.000
		1-XCR	0.987	0.004	0.985	0.985	0.985	0.985	1.000
	0	SACP	0.980	0.048	0.769	1.000	1.000	1.000	1.000
		SACR	0.976	0.055	0.718	1.000	1.000	1.000	1.000
		1-XCR	0.993	0.002	0.992	0.992	0.992	0.992	1.000
	2	SACP	0.970	0.011	0.966	0.966	0.966	0.966	1.000
0.90		SACR	1.000	0.000	1.000	1.000	1.000	1.000	1.000
		1-XCR	0.948	0.009	0.943	0.943	0.943	0.951	0.984
	5	SACP	0.944	0.023	0.878	0.949	0.949	0.949	1.000
		SACR	0.992	0.022	0.929	1.000	1.000	1.000	1.000
		1-XCR	0.971	0.005	0.969	0.969	0.969	0.969	0.990
	12	SACP	0.855	0.022	0.842	0.845	0.845	0.857	0.971
		SACR	0.997	0.010	0.965	1.000	1.000	1.000	1.000

Table B.1: Summary statistics from the simulations of various sequencing technology ratios and the assessment of transmission clusters produced at different SNP thresholds in Section 3.9. SACR=sample-averaged cluster recall; SACP=sample-average cluster precision; XCR=excess clustering rate; std=standard deviation.

Appendix C

Supporting work for Chapter 4

C.1 Drug susceptibility testing

C.1.1 Madagascar

Culture on Löwenstein-Jensen (LJ) is still the gold-standard method for *M. tuberculosis* identification and the detection of resistance. The indirect proportion method on LJ medium was performed to test the susceptibility of positive cultures against anti-*M. tuberculosis* drugs. $4\,\mu g\,ml^{-1}$, $0.2\,\mu g\,ml^{-1}$, $40\,\mu g\,ml^{-1}$, $2\,\mu g\,ml^{-1}$, $30\,\mu g\,ml^{-1}$, $30\,\mu g\,ml^{-1}$, and $40\,\mu g\,ml^{-1}$ were the critical concentrations used for Streptomycin, Isoniazid, Rifampicin, Ethambutol, Kanamycin, Amikacin and Capreomycin, respectively. The growth on a drug-free medium was compared with the growth on a medium containing an anti-*M. tuberculosis* agent. An isolate was identified as resistant if at least 1% of growth is present at the critical concentration of the drug in the culture medium.

C.1.2 South Africa

I have emailed Tash and Anzaan for this

C.1.3 Full data availability

Available phenotype information for culture-based *and* line probe assay (LPA) drug susceptibility testing (DST) are shown in Table C.1 and Figure C.1.

Drug	Count
Amikacin	88
Amikacin-LPA	30
Capreomycin	51
Capreomycin-LPA	27
Ciprofloxacin-LPA	5
Ethambutol	90
Ethambutol-LPA	21
Isoniazid	98
Isoniazid-LPA	124
Kanamycin	51
Kanamycin-LPA	27
Moxifloxacin	1
Moxifloxacin-LPA	5
Ofloxacin	86
Ofloxacin-LPA	30
Pyrazinamide	1
Rifampicin	91
Rifampicin-LPA	124
Streptomycin	90

Table C.1: Culture-based and line probe assay (LPA) drug susceptibility data available for samples. The counts are the number of samples with phenotype information available for that drug.

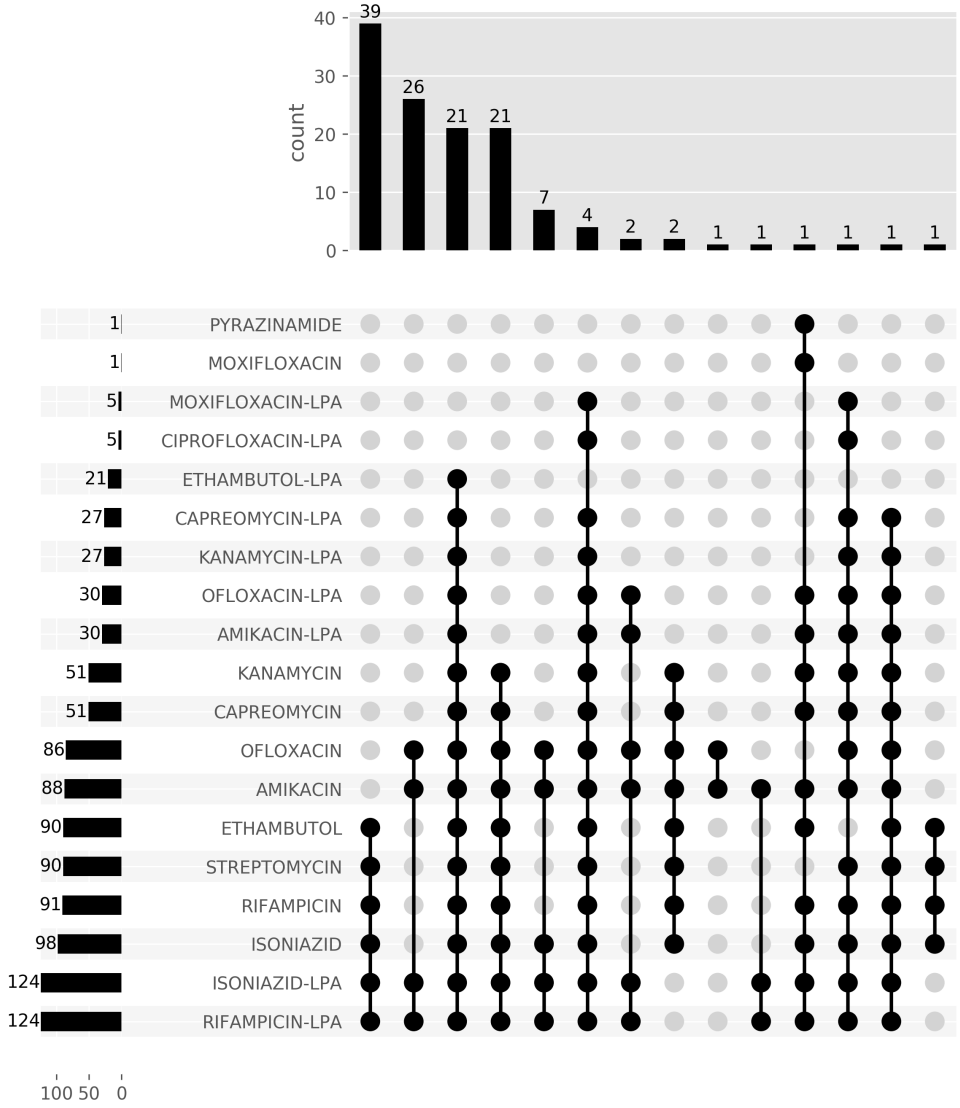


Fig. C.1: Culture-based and line probe assay (LPA) drug susceptibility data available for samples. Each row is a drug, and the columns represent a set of samples that have phenotype information for those drugs with a filled cell. The top panel shows the number of samples in the set for that combination of drugs. The bar plot in the left panel shows the number of samples with phenotype information for that drug.

C.2 Constructing a panel reference graph

C.2.1 Example panel and associated VCF produced by `drprg`

C.2.2 Panel-based PRG density and haplotype problems

In the initial development stage of `drprg`, we tried using a PRG built from a panel of known resistance-causing mutations (Section 4.3.1). However, when we began assessing the performance of `drprg`, we discovered two common issues with this panel-based PRG. First, sites in certain genes were far too dense and led to many alleles having shared minimizers, thus causing shared read depth (example below). Second, the lack of haplotype information - particularly close panel variants - leads to the bulk of the missed resistance (false negative) calls. We will use two real examples to illustrate these issues.

Complex PRG sites

The first issue of the panel-based PRG mentioned above is sites in some genes being too dense (a large number of alternate alleles). This density occurs at gene locations where there are many panel variants next to each other or variants where a change to *any* amino acid (denoted by `X`) leads to drug resistance.

The reason for adjacent variants causing increased density is a parameter in `make_prg` called the minimum match length (m; also discussed in Section 3.12.3). m controls the number of base pairs that must agree between all sequences at the same position for those positions to be collapsed. Otherwise, if there is a disagreement, the alleles are split into alternate paths.

Figure C.3 shows an example of how m can impact the structure of a PRG. In this example, there are three sequences with three disagreeing sites (positions 5, 9 and 13; lower-case letters; top panel). When $m = 3$, there are three "neat" single-base sites, and all `A`s and `T`s are collapsed because at least three continuous bases agree between the three sequences (middle panel). However, when m is increased to 4 (bottom panel), the runs of three `A`s in between the disagreeing positions are no longer collapsed. In this example, it may appear that $m = 3$ has *more* density, as there are more sites in total, but $m = 4$ has more alternate alleles; the more there are, the greater the likelihood of them sharing minimizer k-mers.

Panel

```
rrs     C492X   DNA     NONE
inhA    S94A    PROT    Isoniazid
```

VCF

```
##INFO=<ID=RES,Number=1,Type=String,Description="Residue the variant describes (i.e. Nucleic/Amino)">
##INFO=<ID=DRUGS,Number=.,Type=String,Description="Drugs this variant causes resistance to">
##INFO=<ID=PAD,Number=1,Type=Integer,Description="Number of bases added to start and end of gene">
##INFO=<ID=ST,Number=1,Type=String,Description="Strand the gene is on">
#CHROM  POS  ID          REF  ALT     ...  INFO
rrs     592  rrs_C492X   C    A,G,T   ...  PAD=100;RES=DNA;DRUGS=NONE;ST=+
inhA    380  inhA_S94A   TCG  GCT,GCC,GCA,GCG ...  PAD=100;RES=PROT;DRUGS=Isoniazid;ST=+
```

Fig. C.2: An example panel (top) required by `drprg`. The columns indicate the gene, mutation, residue the variant describes, and drug(s) the entry impacts. The panel is turned into a VCF (bottom) with proteins converted to DNA. Associated information from the panel and annotation are encoded in the `INFO` field for each entry. Note: some unnecessary information for this example is removed from the VCF entry shown here to reduce the size.

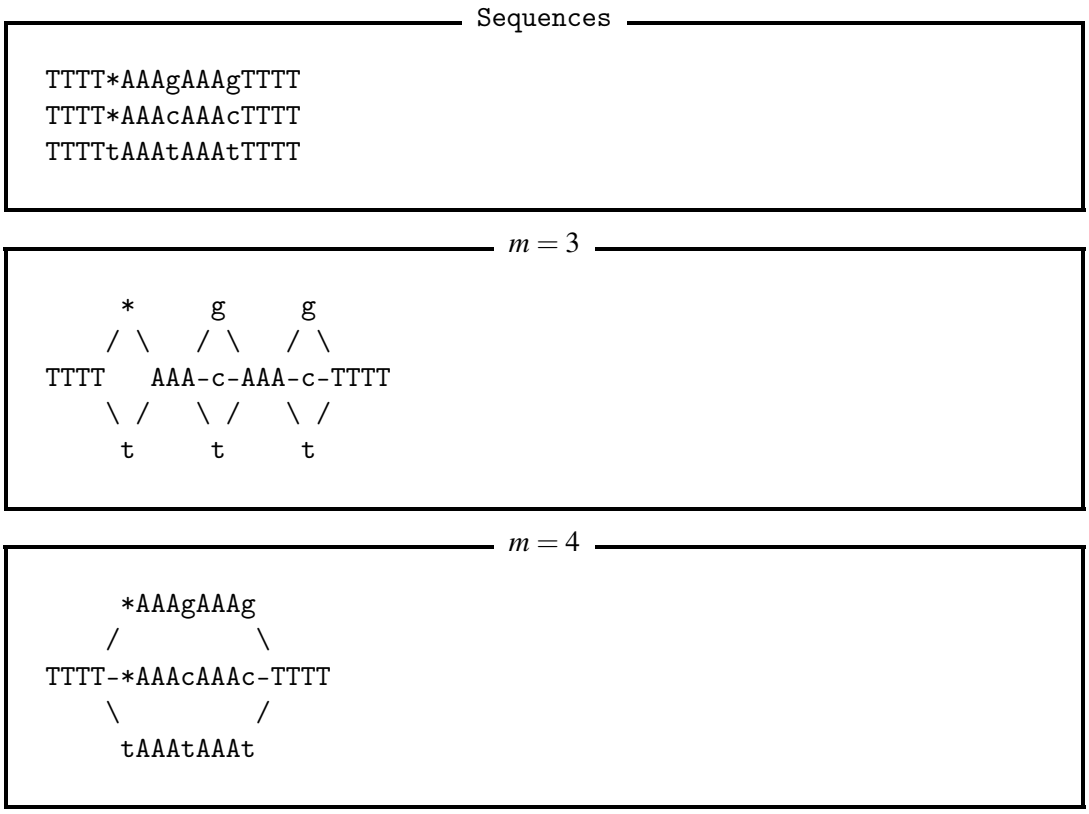


Fig. C.3: An example of how the `make_prg` minimum match length (m) parameter effects PRG structure. The PRGs for three sequences (top) with three positions of difference (lower-case letters) are shown. m controls when sequence is collapsed. When $m = 3$ (middle), all runs of `A` and `T` are collapsed because at least three continuous bases agree between the three sequences. However, when $m = 4$ (bottom), the `A`s are no longer collapsed, leaving a single, longer branched path for each sequence, rather than the three smaller, single-base paths when $m = 3$. `*` is a placeholder to indicate an insertion/deletion.

Let us use an example of a minimizer k-mer of `TTTAAA` starting at position 1. When $m = 4$, the top two paths (alleles) share this minimizer. As a result, if a sequencing read has that k-mer, both alleles have their read coverage incremented by one. However, the sequence cannot have come from two alleles.

The issue of shared minimizer k-mers in dense regions of the panel-based PRG led to many `drprg` prediction errors (`drprg` uses `pandora` to facilitate this - Section 4.3.2). One such example from real data is shown in Figure C.4, which focuses on the *pncA* mutation S65F. In this example, the panel-based PRG (top panel) has 126 alternate alleles due to many variants within less than m positions of each other. In the alleles shown, there is a lot of shared sequence, and as a result, when looking at the coverage information (`MEAN_FWD_COVG` and `MEAN_REV_COVG`), we can see that *all* alleles have what looks to be confident read depth. However, biologically, we know that it is *extremely* unlikely there are 126 different strains in this sample; the null genotype

pncA mutation S65F VCF entry for panel-based PRG

```
#CHROM  POS     ID      REF     ALT     QUAL    FILTER  INFO    FORMAT
↪  sample
pncA    284     5bf4ae25        CCGGACTATTCCTCGTCGTGGCCACCGCATTGC
↪  AGAGACTATTCCTCGTCGTGGCCACCGCATTGC,AGCGACTATTCCTCGTCGTGGCCACCGCATTGC,AG
↪  GGACTATTCCTCGTCGTGGCCACCGCATTGC,AGTGACTATTCCTCGTCGTGGCCACCGCATTGC,CAAG
↪  ACTATTCCTCGTCGTGGCCACCGCATTGC,CAGGACTATTCCTCGTCGTGGCCACCGCATTGC,CCAGAC
↪  TATTCCTCGTCGTGGCCACCGCATTGC,CCCGACTATTCCTCGTCGTGGCCACCGCATTGC,CCGCACTA
↪  TTCCTCGTCGTGGCCACCGCATTGC,CCGCATTATTCCTCGTCGTGGCCACCGCATTGC,CCGGAATATT
↪  CCTCGTCGTGGCCACCGCATTGC,CCGGACTAATCCTCGTCGTGGCCACCGCATTGC,CCGGACTAGTCC
↪  TCGTCGTGGCCACCGCATTGC,CCGGACTATTCCCCATCGTGGCCACCGCATTGC,CCGGACTATTCCCC
↪  CTCGTGGCCACCGCATTGC,CCGGACTATTCCCCGTCGTGGCCACCGCATTGC,<110 hidden>   .
↪       frs       VARID=pncA_S65F,<rest hidden>;PREDICT=S,<rest hidden>
↪     GT:MEAN_FWD_COVG:MEAN_REV_COVG:GT_CONF
↪  .:31,14,14,12,14,12,13,20,20,17,17,22,20,16,20,20,<110
↪  hidden>:29,14,16,13,15,13,13,20,21,17,17,21,20,16,20,20,<110
↪  hidden>:14.4118
```

pncA mutation S65F VCF entry for population-based PRG

```
#CHROM  POS     ID      REF     ALT     QUAL    FILTER  INFO    FORMAT
↪  sample
pncA    292     f2ff5236        TTCC    TATCT,TTTC      .       PASS
↪  VARID=pncA_S65F,<rest hidden>;PREDICT=R,<rest hidden>
↪  GT:MEAN_FWD_COVG:MEAN_REV_COVG:GT_CONF     2:6,0,19:9,0,26:179.789
```

Fig. C.4: Contrasting examples of the same *pncA* variant site (S65F) from a panel-based PRG (top) and a population-based PRG (bottom). `VARID` indicates the panel variants this site overlaps, and `PREDICT` is the resistance prediction for the relevant `VARID`. `MEAN_FWD_COVG` and `MEAN_REV_COVG` specify the mean forward and reverse *k*-mer coverage on minimizer *k*-mers that overlap this site. Note: due to a large number of alternate alleles (126) in the panel-based record and overlapping `VARID`s, some data has been elided for illustrative purposes.

call and low genotype confidence also corroborate this. Because no genotype can be confidently called, the prediction for this sample is susceptible (`PREDICT` tag in the VCF entry).

In contrast, when using the population-based PRG (bottom panel), we see a very different variant record. First, there are now only two alternate alleles indicating that in the population, there is not as much variation at this site as the panel-based PRG would suggest. Second, due to this reduced density, the coverage information is much "cleaner", and the genotyping much more confident. Subsequently, we now (correctly) classify this sample as having the S65F mutation in *pncA*, leading to a resistance prediction.

The scenario in Figure C.4 was repeatedly encountered across the samples in this study. It was particularly common in genes with a lot of mutations in close proximity, such as *pncA*, *katG*, and *rpoB*. Switching to the use of a population-based PRG helped reduce the majority of density-related errors that were being made by `drprg`.

Lack of haplotype information

The second panel-based PRG problem mentioned was the lack of haplotype information. This issue is of particular relevance to close panel variants and was the cause of many missed resistance calls when using the panel-based PRG. Additionally, we raised it in Section 3.12.3 as an avenue for improvement when building PRGs.

When constructing a panel-based PRG, as outlined in Section 4.3.1, the use of haplotype information is not possible. The variants in the panel do not have accompanying data indicating what other variants they do or do not occur in tandem with. As a result, when two variants occur within m base pairs of each other, the co-occurrence of the two in the same allele is not possible. For example, in Figure C.3, when $m = 3$, it is possible to take a path through the PRG which would yield a sequence containing the three variants `t`, `g`, and `c`. However, when $m = 4$, such a path is not possible as all possible combinations of variants are not allowed. This failure to construct all possible combinations of haplotypes is a feature of `make_prg` and avoids combinatorial "explosions" that would occur - e.g., in the example of $m = 4$, there are 18 possible recombinants of the variants that would need to be listed.

A common site where this lack of haplotype information was causing a lot of missed resistance is shown in Figure C.5. This site occurs in the *gyrA* gene and is known to cause resistance to fluoroquinolones. Of relevance to the haplotype scenario we are discussing are the mutations at codons 94 and 95. An amino acid change from aspartic acid (D) at codon 94 is known to cause resistance, while the mutation of serine (S) to threonine (T) at position 95 is a common polymorphism not related to resistance [232–234]. Indeed, the S95T variant is so common, it is used in a line probe assay for fluoroquinolones [233]. S95T can co-occur with D94 mutations, but due to this haplotype issue, there are no alternate alleles with a combination of the resistance-causing mutation and the natural polymorphism (the last ALT in Figure C.5 top represents S95T). The sample in Figure C.5 has both the D94N and S95T mutations. This combination does not occur in the panel-based PRG - leading to no coverage on any alleles (top panel). However, the allele combination *is* found in the population-based PRG (bottom panel), allowing correct genotyping and thus (correctly) calling resistance to fluoroquinolones.

One conceivable way to avoid this haplotype problem would be to reduce the value of m. As we see in Figure C.3, if m is low enough, the variants can be separated by matching sequence, thus allowing recombinant paths through the PRG; however, there are two drawbacks to this approach. First, if two variants occur directly next to each

gyrA mutation S95T VCF entry for panel-based PRG

```
#CHROM  POS     ID      REF     ALT     QUAL    FILTER  INFO    FORMAT
↪  sample
gyrA    380     164fc346        GACAG   AACAG,AATAG,CACAG,CATAG,GCAAG,GC
↪  CAG,GCGAG,GCTAG,GGAAG,GGCAG,GGGAG,GGTAG,TACAG,TATAG,TGCAG,TGTAG,GACAC
↪  .       ld;lgc  VARID=gyrA_D94N,gyrA_D94H,gyrA_D94A,gyrA_D94G,gyrA_D94
↪  Y,gyrA_D94C,gyrA_S95T;PREDICT=S,S,S,S,S,S,S
↪  GT:MEAN_FWD_COVG:MEAN_REV_COVG:GT_CONF     .:0,0,0,0,0,0,0,0,0,0,0,0,0
↪  ,0,0,0,0,0:0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0:0
```

gyrA mutation S95T VCF entry for population-based PRG

```
#CHROM  POS     ID      REF     ALT     QUAL    FILTER  INFO    FORMAT
↪  sample
gyrA    380     01a6abfd        GACAG
↪  AACAC,CACAC,GACAC,GCCAC,GGCAC,TACAC,TACAG       .       PASS
↪  VARID=gyrA_D94N,gyrA_D94H,gyrA_D94A,gyrA_D94G,gyrA_D94Y,gyrA_D94C,gyrA
↪  _S95T;PREDICT=R,S,S,S,S,S,S
↪  GT:MEAN_FWD_COVG:MEAN_REV_COVG:GT_CONF
↪  1:0,38,0,0,13,0,0,0:0,24,0,0,25,0,0,0:117.539
```

Fig. C.5: Contrasting examples of the same *gyrA* variant site from a panel-based PRG (top) and a population-based PRG (bottom). `VARID` indicates the panel variants this site overlaps and `PREDICT` is the resistance prediction for the relevant `VARID`. `MEAN_FWD_COVG` and `MEAN_REV_COVG` specify the mean forward and reverse k-mer coverage on minimizer k-mers that overlap this site.

other, even $m = 1$ will be unable to separate them by matching sequence. Second, there is an inverse relationship between m and indexing runtime and size; as m decreases, indexing the PRG takes longer as there are more paths to walk, and its size increases for the same reason (i.e., more minimizer k-mers).

In these situations, we would expect *de novo* variant discovery to be able to recover the correct allele combination. However, we found it struggled to do so due to the choice of maximum likelihood (ML) path in the beginning. As mentioned in Section 1.3.2, after `pandora` assigns k-mer coverage (hits) to the local PRGs, it uses dynamic programming to select the ML path through each. In the panel-based PRG example in Figure C.5, because there is no coverage on any allele, the ML path is effectively a random choice at this site. As such, if the random choice is quite different to the allele present in the sample, the *de novo* variant discovery step is unable to find anchor k-mers (Section 2.2.2) and thus cannot perform local assembly on the region.

Ultimately, the easiest way to solve the lack of haplotype information was to build the PRG from a cross-section of variation seen in a population. As we ensure there is enough diversity in the sampled population (Section 3.5), the commonly occurring allele combinations are present in the final PRG.

C.3 Example `drprg` prediction report

Figure C.6 is an example of the JSON file output from `drprg predict` (see Section 4.3.2). This JSON file shows the resistance prediction for each drug and the supporting evidence. It has had some drugs removed for brevity.

C.4 Adjustment of default `mykrobe` Nanopore settings

During the process of gathering the resistance prediction results in Section 4.5 and Section 4.4, we investigated whether the `mykrobe` default parameters for Nanopore are optimal. Since they were calibrated on only five samples [45]. In addition, we look at the impact of changing the default Illumina expected error rate (0.05) to the reported 0.001 [235].

The Nanopore preset in `mykrobe` sets the expected error rate to 0.15, the ploidy model to haploid, and ignores calls below the 90% confidence threshold (as judged by simulating confidence scores from those present in the data) [45]. Instead, we disable this preset and change the expected error rate to 0.08, leave the ploidy as haploid (avoiding minor resistance calls), and turn off the confidence threshold simulations.

The results of these parameter changes are shown in Figure C.7 and summarised in Table C.2 for concordance with phenotype (see Section 4.5). In addition, as culture-based phenotypes are not available for all drugs and samples, we compare the concordance of Nanopore predictions with Illumina in Figure C.8 and Table C.3.

When comparing the WGS concordance with culture-based phenotypes, these results reveal that the default Nanopore settings for `mykrobe` lead to a much higher number of missed resistance (FN) classifications for all drugs when compared to the adjusted settings. However, the default settings do lead to fewer false-positive predictions for all drugs. Additionally, the adjusted Illumina error rate led to one FN being recovered (classified TP) for ethambutol and unchanged classifications for everything else.

Nanopore concordance with Illumina predictions shows default settings lead to more FN calls than with adjusted settings, especially for the first-line drugs ethambutol, isoniazid, and rifampicin. The adjusted settings for `mykrobe` led to quite similar predictions between Nanopore and Illumina. Additionally, they lead to much less Nanopore missed resistance calls but more false positives. However, missed resistance

```
{
  "sample": "R21770",
  "susceptibility": {
    "Amikacin": {
      "evidence": [],
      "predict": "S"
    },
    "Ethambutol": {
      "evidence": [
        {
          "gene": "embB",
          "residue": "DNA",
          "variant": "G2995A",
          "vcfid": "4a883e71"
        }
      ],
      "predict": "U"
    },
    "Isoniazid": {
      "evidence": [],
      "predict": "S"
    },
    "Ofloxacin": {
      "evidence": [],
      "predict": "S"
    },
    "Pyrazinamide": {
      "evidence": [],
      "predict": "S"
    },
    "Rifampicin": {
      "evidence": [
        {
          "gene": "rpoB",
          "residue": "PROT",
          "variant": "L430X",
          "vcfid": "52981ff5"
        }
      ],
      "predict": "R"
    },
    "Streptomycin": {
      "evidence": [],
      "predict": "S"
    }
  }
}
```


Fig. C.6: An example drug susceptibility JSON report file produced by `drprg`. The report maps a drug to its prediction and any relevant evidence supporting that prediction. Note, evidence for susceptibility (`"S"`) predictions is not provided as susceptibility is assumed where no `R/U` prediction is made. S=susceptible; U=unknown (i.e., novel variant); R=resistant.

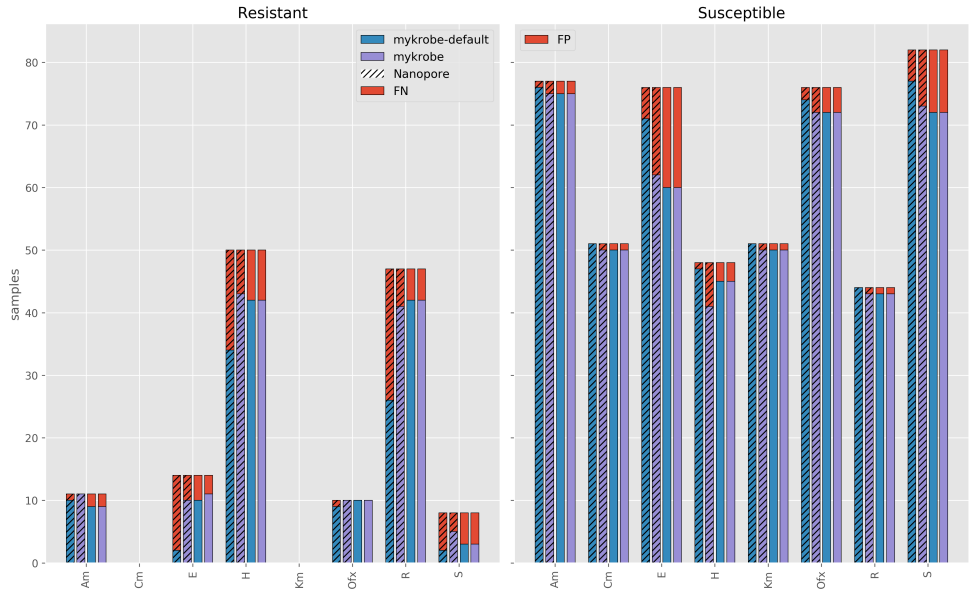


Fig. C.7: Number of resistant (left) and susceptible (right) culture-based drug susceptibility phenotypes correctly identified by `mykrobe` with default (blue) or adjusted (purple) settings. Nanopore data is indicated by diagonal stripes, with Illumina having no stripes. The red bars indicate missed (FN) or incorrect (FP) predictions. The x-axis shows the drugs with available phenotype data. E - ethambutol; H - isoniazid; R - rifampicin; S - streptomycin; Km - kanamycin; Am - amikacin; Ofx - ofloxacin; Cm - capreomycin.

is generally deemed a "worse" error, and as such, for the work in Chapter 4 we chose to use these adjusted parameters for `mykrobe`.

Drug	Technology	mykrobe	FN(R)	FP(S)	FNR(95% CI)	FPR(95% CI)	PPV(95% CI)	NPV(95% CI)
Amikacin	Nanopore	mykrobe-default	1(11)	1(77)	9.1% (1.6-37.7%)	1.3% (0.2-7.0%)	90.9% (62.3-98.4%)	98.7% (93.0-99.8%)
		mykrobe	0(11)	2(77)	0.0% (0.0-25.9%)	2.6% (0.7-9.0%)	84.6% (57.8-95.7%)	100.0% (95.1-100.0%)
	Illumina	mykrobe-default	2(11)	2(77)	18.2% (5.1-47.7%)	2.6% (0.7-9.0%)	81.8% (52.3-94.9%)	97.4% (91.0-99.3%)
		mykrobe	2(11)	2(77)	18.2% (5.1-47.7%)	2.6% (0.7-9.0%)	81.8% (52.3-94.9%)	97.4% (91.0-99.3%)
Capreomycin	Nanopore	mykrobe-default	0(0)	0(51)	-	0.0% (0.0-7.0%)	-	100.0% (93.0-100.0%)
		mykrobe	0(0)	1(51)	-	2.0% (0.3-10.3%)	0.0% (0.0-79.3%)	100.0% (92.9-100.0%)
	Illumina	mykrobe-default	0(0)	1(51)	-	2.0% (0.3-10.3%)	0.0% (0.0-79.3%)	100.0% (92.9-100.0%)
		mykrobe	0(0)	1(51)	-	2.0% (0.3-10.3%)	0.0% (0.0-79.3%)	100.0% (92.9-100.0%)
Ethambutol	Nanopore	mykrobe-default	12(14)	5(76)	85.7% (60.1-96.0%)	**6.6% (2.8-14.5%)**	28.6% (8.2-64.1%)	85.5% (76.4-91.5%)
		mykrobe	4(14)	14(76)	**28.6% (11.7-54.6%)**	18.4% (11.3-28.6%)	41.7% (24.5-61.2%)	93.9% (85.4-97.6%)
	Illumina	mykrobe-default	4(14)	16(76)	28.6% (11.7-54.6%)	21.1% (13.4-31.5%)	38.5% (22.4-57.5%)	93.8% (85.0-97.5%)
		mykrobe	3(14)	16(76)	21.4% (7.6-47.6%)	21.1% (13.4-31.5%)	40.7% (24.5-59.3%)	95.2% (86.9-98.4%)
Isoniazid	Nanopore	mykrobe-default	16(50)	1(48)	32.0% (20.8-45.8%)	**2.1% (0.4-10.9%)**	97.1% (85.5-99.5%)	74.6% (62.7-83.7%)
		mykrobe	7(50)	7(48)	**14.0% (7.0-26.2%)**	14.6% (7.2-27.2%)	86.0% (73.8-93.0%)	85.4% (72.8-92.8%)
	Illumina	mykrobe-default	8(50)	3(48)	16.0% (8.3-28.5%)	6.2% (2.1-16.8%)	93.3% (82.1-97.7%)	84.9% (72.9-92.1%)
		mykrobe	8(50)	3(48)	16.0% (8.3-28.5%)	6.2% (2.1-16.8%)	93.3% (82.1-97.7%)	84.9% (72.9-92.1%)
Kanamycin	Nanopore	mykrobe-default	0(0)	0(51)	-	0.0% (0.0-7.0%)	-	100.0% (93.0-100.0%)
		mykrobe	0(0)	1(51)	-	2.0% (0.3-10.3%)	0.0% (0.0-79.3%)	100.0% (92.9-100.0%)
	Illumina	mykrobe-default	0(0)	1(51)	-	2.0% (0.3-10.3%)	0.0% (0.0-79.3%)	100.0% (92.9-100.0%)
		mykrobe	0(0)	1(51)	-	2.0% (0.3-10.3%)	0.0% (0.0-79.3%)	100.0% (92.9-100.0%)
Ofloxacin	Nanopore	mykrobe-default	1(10)	2(76)	10.0% (1.8-40.4%)	2.6% (0.7-9.1%)	81.8% (52.3-94.9%)	98.7% (92.8-99.8%)
		mykrobe	0(10)	4(76)	0.0% (-0.0-27.8%)	5.3% (2.1-12.8%)	71.4% (45.4-88.3%)	100.0% (94.9-100.0%)
	Illumina	mykrobe-default	0(10)	4(76)	0.0% (-0.0-27.8%)	5.3% (2.1-12.8%)	71.4% (45.4-88.3%)	100.0% (94.9-100.0%)
		mykrobe	0(10)	4(76)	0.0% (-0.0-27.8%)	5.3% (2.1-12.8%)	71.4% (45.4-88.3%)	100.0% (94.9-100.0%)
Rifampicin	Nanopore	mykrobe-default	21(47)	0(44)	44.7% (31.4-58.8%)	0.0% (0.0-8.0%)	100.0% (87.1-100.0%)	67.7% (55.6-77.8%)
		mykrobe	6(47)	1(44)	**12.8% (6.0-25.2%)**	2.3% (0.4-11.8%)	97.6% (87.7-99.6%)	87.8% (75.8-94.3%)
	Illumina	mykrobe-default	5(47)	1(44)	10.6% (4.6-22.6%)	2.3% (0.4-11.8%)	97.7% (87.9-99.6%)	89.6% (77.8-95.5%)
		mykrobe	5(47)	1(44)	10.6% (4.6-22.6%)	2.3% (0.4-11.8%)	97.7% (87.9-99.6%)	89.6% (77.8-95.5%)
Streptomycin	Nanopore	mykrobe-default	6(8)	5(82)	75.0% (40.9-92.9%)	**6.1% (2.6-13.5%)**	28.6% (8.2-64.1%)	92.8% (85.1-96.6%)
		mykrobe	3(8)	9(82)	**37.5% (13.7-69.4%)**	11.0% (5.9-19.6%)	35.7% (16.3-61.2%)	96.1% (89.0-98.6%)
	Illumina	mykrobe-default	5(8)	10(82)	62.5% (30.6-86.3%)	12.2% (6.8-21.0%)	23.1% (8.2-50.3%)	93.5% (85.7-97.2%)
		mykrobe	5(8)	10(82)	62.5% (30.6-86.3%)	12.2% (6.8-21.0%)	23.1% (8.2-50.3%)	93.5% (85.7-97.2%)

Table C.2: Comparison of WGS drug resistance predictions with culture-based phenotype. For this comparison, we assume the culture-based phenotype is correct and evaluate mykrobe with default and adjusted settings for Illumina and Nanopore resistance predictions accordingly. Pyrazinamide and Moxifloxacin are excluded as phenotype information is only available for one sample. Bold text is used to highlight differences of note. FN=false negative; R=number of resistant samples; FP=false positive; S=number of susceptible samples; FNR=false negative rate; FPR=false positive rate; PPV=positive predictive value; NPV=negative predictive value; CI=Wilson score confidence interval

Drug	Tool	FN(R)	FP(S)	FNR(95% CI)	FPR(95% CI)	PPV(95% CI)	NPV(95% CI)
Amikacin	mykrobe-default	2(12)	2(138)	16.7% (4.7-44.8%)	1.4% (0.4-5.1%)	83.3% (55.2-95.3%)	98.6% (94.9-99.6%)
	mykrobe	0(12)	2(138)	**0.0% (0.0-24.2%)**	1.4% (0.4-5.1%)	85.7% (60.1-96.0%)	100.0% (97.3-100.0%)
Capreomycin	mykrobe-default	2(12)	2(138)	16.7% (4.7-44.8%)	1.4% (0.4-5.1%)	83.3% (55.2-95.3%)	98.6% (94.9-99.6%)
	mykrobe	0(12)	2(138)	**0.0% (0.0-24.2%)**	1.4% (0.4-5.1%)	85.7% (60.1-96.0%)	100.0% (97.3-100.0%)
Ciprofloxacin	mykrobe-default	4(17)	0(133)	23.5% (9.6-47.3%)	0.0% (0.0-2.8%)	100.0% (77.2-100.0%)	97.1% (92.7-98.9%)
	mykrobe	1(17)	0(133)	**5.9% (1.0-27.0%)**	0.0% (0.0-2.8%)	100.0% (80.6-100.0%)	99.3% (95.9-99.9%)
Ethambutol	mykrobe-default	40(56)	0(94)	71.4% (58.5-81.6%)	0.0% (0.0-3.9%)	100.0% (80.6-100.0%)	70.1% (61.9-77.2%)
	mykrobe	3(57)	0(93)	**5.3% (1.8-14.4%)**	0.0% (0.0-4.0%)	100.0% (93.4-100.0%)	96.9% (91.2-98.9%)
Isoniazid	mykrobe-default	15(80)	1(70)	18.8% (11.7-28.7%)	**1.4% (0.3-7.7%)**	98.5% (91.9-99.7%)	82.1% (72.6-88.9%)
	mykrobe	0(80)	7(70)	**0.0% (0.0-4.6%)**	10.0% (4.9-19.2%)	92.0% (84.3-96.0%)	100.0% (94.3-100.0%)
Kanamycin	mykrobe-default	2(13)	2(137)	15.4% (4.3-42.2%)	1.5% (0.4-5.2%)	84.6% (57.8-95.7%)	98.5% (94.8-99.6%)
	mykrobe	0(13)	2(137)	**0.0% (0.0-22.8%)**	1.5% (0.4-5.2%)	86.7% (62.1-96.3%)	100.0% (97.2-100.0%)
Moxifloxacin	mykrobe-default	4(17)	0(133)	23.5% (9.6-47.3%)	0.0% (0.0-2.8%)	100.0% (77.2-100.0%)	97.1% (92.7-98.9%)
	mykrobe	1(17)	0(133)	**5.9% (1.0-27.0%)**	0.0% (0.0-2.8%)	100.0% (80.6-100.0%)	99.3% (95.9-99.9%)
Ofloxacin	mykrobe-default	4(17)	0(133)	23.5% (9.6-47.3%)	0.0% (0.0-2.8%)	100.0% (77.2-100.0%)	97.1% (92.7-98.9%)
	mykrobe	0(17)	0(133)	**0.0% (0.0-18.4%)**	0.0% (0.0-2.8%)	100.0% (81.6-100.0%)	100.0% (97.2-100.0%)
Pyrazinamide	mykrobe-default	13(31)	0(119)	41.9% (26.4-59.2%)	0.0% (0.0-3.1%)	100.0% (82.4-100.0%)	90.2% (83.9-94.2%)
	mykrobe	1(31)	0(119)	**3.2% (0.6-16.2%)**	0.0% (0.0-3.1%)	100.0% (88.6-100.0%)	99.2% (95.4-99.9%)
Rifampicin	mykrobe-default	30(79)	0(71)	38.0% (28.1-49.0%)	0.0% (0.0-5.1%)	100.0% (92.7-100.0%)	70.3% (60.8-78.3%)
	mykrobe	1(79)	0(71)	**1.3% (0.2-6.8%)**	0.0% (0.0-5.1%)	100.0% (95.3-100.0%)	98.6% (92.5-99.8%)
Streptomycin	mykrobe-default	20(45)	2(105)	44.4% (30.9-58.8%)	1.9% (0.5-6.7%)	92.6% (76.6-97.9%)	83.7% (76.2-89.2%)
	mykrobe	2(46)	2(104)	**4.3% (1.2-14.5%)**	1.9% (0.5-6.7%)	95.7% (85.5-98.8%)	98.1% (93.3-99.5%)

Table C.3: Comparison of Nanopore drug resistance predictions concordance with Illumina predictions. For this comparison, we assume the mykrobe resistance prediction from Illumina data is correct and evaluate the Nanopore prediction accordingly. mykrobe-default and mykrobe indicates mykrobe with default or adjusted settings, respectively. Bold text is used to highlight differences of note. FN=false negative; R=number of resistant samples; FP=false positive; S=number of susceptible samples; FNR=false negative rate; FPR=false positive rate; PPV=positive predictive value; NPV=negative predictive value; CI=Wilson score confidence interval

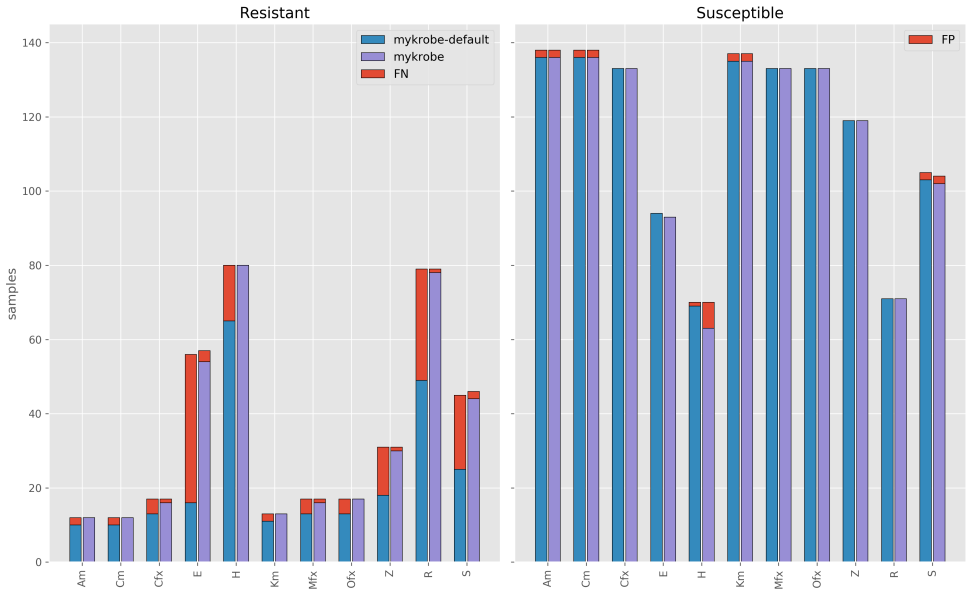


Fig. C.8: Number of resistant (left) and susceptible (right) Illumina WGS-based drug susceptibility phenotypes correctly identified using Nanopore data. Nanopore predictions for `mykrobe` with default (blue) and adjusted (purple) settings are compared to those from Illumina with the same settings. The red bars indicate missed (FN) or incorrect (FP) predictions. The x-axis shows the drugs with available phenotype data. E - ethambutol; H - isoniazid; Z - pyrazinamide; R - rifampicin; S - streptomycin; Km - kanamycin; Am - amikacin; Ofx - ofloxacin; Cm - capreomycin; Mfx - moxifloxacin.