CRANFIELD UNIVERSITY

TOMASZ JANUSZ KUROWSKI

DEVELOPING NOVEL BIOINFORMATICS TOOLS AND PIPELINES FOR WORKING WITH REFERENCE GENOMES AND LARGE SETS OF RESEQUENCED GENOMES

SCHOOL OF WATER, ENERGY AND ENVIRONMENT
Agrifood

PhD
Academic Year: 2015 - 2022

Supervisor: Dr Fady Mohareb

January 2022

CRANFIELD UNIVERSITY

SCHOOL OF WATER, ENERGY AND ENVIRONMENT
Agrifood

PhD

Academic Year 2015 - 2022

TOMASZ JANUSZ KUROWSKI

Developing novel bioinformatics tools and pipelines for working with reference genomes and large sets of resequenced genomes

Supervisor: Dr Fady Mohareb

January 2022

This thesis is submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy
(NB. This section can be removed if the award of the degree is based solely on examination of the thesis)

ABSTRACT

Both reference genomes assembled for individual species and large, publicly maintained sets of resequenced genomes are of immense value to researchers. The former represent important milestones for research involving the species of interest and serve as ostensibly static points of reference for other data, while the latter serve as catalogues of genetic variation, enabling researchers to place their own data in a wider context. However, maintaining sets of resequenced genomes and ensuring their integrity as they undergo updates to match any new releases of their reference genome poses certain computational challenges, as does manipulating and comparing those large sets of genomes in general.

This work reports on the detection and correction of significant errors which were introduced into resequenced tomato data in the course of updating them to a new version. It also introduces Tersect, a low-level utility optimized for manipulating and comparing large sets of resequenced genomic data, as well as Tersect Browser, a Web application which uses the high performance of Tersect, coupled with a higher-level indexing and precomputation scheme to allow for interactive comparison of large sets of resequenced genomes, giving biologists a tool capable of generating visualisations of genetic distance and phylogenetic relationships based on whole-genome sequence data from hundreds of genomes in seconds rather than hours.

Keywords:

Comparative genomics, Genotyping, SNP, SNV, Variant Call Format, Introgression, Tomato

ACKNOWLEDGEMENTS

I want to thank my supervisor Dr Fady Mohareb for the endless patience, support, and understanding he showed me throughout the project, and for providing me with ample opportunities to collaborate on many diverse projects, expanding my research horizons and helping me pick up a plethora of useful skills.

I also want to thank Prof. Andrew Thompson, under whose patient guidance I learned to apply my bioinformatician’s toolkit to the real-world problems of plant science.

Special thanks to Corentin Molitor, who shared an office with me for much of my project and was always within reach if I needed to trade ideas (or snacks!) with someone. Further, I would like to thank all the students whom I have had the opportunity to teach during the time I took to complete the thesis, for providing a useful distraction from research and writing – I never expected to enjoy teaching!

Last but not least, I would like to thank my loving family, who supported me and helped me stay motivated during the course of my work: my mother Renata, my sister Anna, my grandmother Irena, and especially my dear grandfather Henryk, who passed away last year.

TABLE OF CONTENTS

ABSTRACT i
ACKNOWLEDGEMENTS..... ii
LIST OF FIGURES v
LIST OF TABLES ix
LIST OF ABBREVIATIONS x
1 INTRODUCTION 1
  1.1 Background..... 1
  1.2 Aim and objectives..... 5
  1.3 Thesis outline..... 6
  1.4 References 7
2 CORRECTING AND VALIDATING VARIANT DATA LIFT-OVER IN RESEQUENCED TOMATO GENOMES 9
  2.1 Abstract..... 9
    2.1.1 Summary..... 9
    2.1.2 Availability 9
  2.2 Introduction 10
  2.3 Resequenced genome issues..... 12
  2.4 Reference genome issues 17
  2.5 Results..... 22
    2.5.1 SNV corrections 22
    2.5.2 InDel corrections 22
    2.5.3 Annotation 23
    2.5.4 SeqRemap lift-over pipeline 24
  2.6 Discussion 30
  2.7 References 32
3 TERSECT: A SET THEORETICAL UTILITY FOR EXPLORING SEQUECE VARIANT DATA 35
  3.1 Abstract..... 35
    3.1.1 Summary..... 35
    3.1.2 Availability 35
  3.2 Introduction 36
  3.3 Tersect..... 36
    3.3.1 Interface and command parser 36
    3.3.2 Indexing..... 37
  3.4 Benchmarking..... 42
  3.5 Results and discussion 43
  3.6 References 50
4 TERSECT BROWSER 52
  4.1 Abstract..... 52
    4.1.1 Summary..... 52

4.1.2 Availability 52
4.2 Introduction 53
4.3 Materials and methods 55
4.3.1 Benchmark and test data 55
4.3.2 Benchmark hardware 56
4.3 Implementation 57
4.4 Results and discussion 69
4.5 Future work 78
4.6 References 79
5 OVERALL DISCUSSION 84
5.1 Overview 84
5.2 Lift-over and validation 84
5.3 Tersect Browser and Tersect 85
5.4 References 88
APPENDICES 91
Appendix A Tersect User Manual 92
Appendix B Tersect Browser Supplementary Figures and Tables 106
Appendix C Tersect: a set theoretical utility for exploring sequence 111
Appendix D Mutagenesis of *Puccinia graminis f. sp. tritici* and Selection of Gain-of-Virulence Mutants 114

LIST OF FIGURES

Figure 2-1: Positions of "problem areas" where the reference alleles listed in the resequenced genome VCF files hosted by SGN do not match the SL2.50 sequence, marked in red along the length of chromosomes. 12

Figure 2-2: Filtered dot plots comparing the SL2.50 (x-axis) and SL2.40 (y-axis) sequences of chromosomes 1, 2, 3, and 4 alongside respective "problem areas". *The blue lines represent reverse complement sequences, indicating the presence of reversed scaffolds. The two reversed sequences at the start of chromosome 2 effectively form a single "problem area".* 14

Figure 2-3: Filtered dot plots comparing the SL2.50 (x-axis) and SL2.40 (y-axis) sequences of chromosomes 5, 6, 7, and 8 alongside respective "problem areas". *The blue lines represent reverse complement sequences, indicating the presence of reversed scaffolds. Note that chromosome 7 includes no reversed scaffolds and thus no problem areas. The group of four neighbouring reversed scaffolds in chromosome 8 form a largely contiguous "problem area".* 15

Figure 2-4: Filtered dot plots comparing the SL2.50 (x-axis) and SL2.40 (y-axis) sequences of chromosomes 9, 10, 11, and 12 alongside respective "problem areas". *The blue lines represent reverse complement sequences, indicating the presence of reversed scaffolds. Chromosome 10 includes re-arranged scaffolds but no reversed scaffolds and as a result it exhibits no "problem areas". The largest scaffold in chromosome 12 was reversed; this may be an error in the SL2.50 build (see* Reference genome issues*).* 16

Figure 2-5: Diagram of the Chromosome 9 scaffold order and orientation in the SL2.40 reference, according to the Shearer et al. paper (FISH), and in the SL2.50 reference. *Reversed red arrows indicate scaffolds which are reversed compared to SL2.40. The relative scaffold sizes are to scale except for scaffold 2 (SL2.40sc6916) which would have been too small to see clearly. Gap sizes are omitted.* 17

Figure 2-6: Diagram of the Chromosome 12 scaffold order and orientation in the SL2.40 reference, according to the Shearer et al. paper (FISH), and in the SL2.50 reference. *Reversed red arrows indicate scaffolds which are reversed compared to SL2.40. The relative scaffold sizes are to scale. Gap sizes are omitted.* 18

Figure 2-7: Example of an incorrect SNV in the SGN JBrowse genome browser. *The reference allele C matches the base on the complementary (bottom) DNA strand rather than the reference strand.* 22

Figure 2-8: Example of an incorrect InDel in the SGN JBrowse genome browser. *The reference allele matches the reverse of its preceding sequence (highlighted in cyan) on the complementary strand, showing that both the allele sequences and their positions need to be corrected.* 23

Figure 2-9: Diagram of the full SeqRemap lift-over pipeline as used with NUCmer whole-genome alignment. *The "custom section" of the pipeline, marked in red, is a (technically optional) mapping and variant calling step that has to be set up externally by the user, to match the pipeline used to generate the source VCFs (as closely as possible), hence why the tools (bwa, bcftools) are only named as examples. It is used to find variants in the "gaps" introduced by novel sequence data in the destination genome.* 26

Figure 3-1: Tersect index file construction diagram. *Parts A) and B) show the contents of example VCF input files (metadata and certain columns were omitted). The first input file contains data for two samples (GEN1 and GEN2) and the second file contains data for a single sample (GEN3). All alleles contained in a chromosome are stored in a single list as seen in part C). Membership of individual alleles in each of the samples is encoded in bit arrays as seen in part D), which shows a 32-bit word for the sake of simplicity (Tersect uses 64-bit words by default). The most significant bit is set for all three bit arrays, indicating that the specific word shown is a literal word (as opposed to a fill word – these terms are explained in section below). Note that the indices in the chromosome variant table and the sample bit arrays match – the lists are parallel.* 40

Figure 3-2: Diagrammatic example of WAH compression and variant retrieval by Tersect. *Three words (part A) encode allele contents for 315 successive alleles stored in the chromosome variant list shown in part B. The literal words encode the indices of variants present in a sample, while the fill word records the length of a run of empty words (each corresponding to seven absent alleles). The stored binary value is 0b0101011 (decimal 43). With seven alleles per word, this can be used to advance the index indicator of the variant list by 7 x 43 = 301 positions when the bit array is traversed. Note that, while for the sake of simplicity the example uses 8-bit words, Tersect uses 64-bit words by default.* 41

Figure 3-3: Benchmarking results for the identification of variants private to a single genome out of subsets of 444 tomato genomes. *See* Table 3-1 *for the numeric results.* 44

Figure 3-4: Benchmarking results for the intersections of subsets of 56 *Solanum pimpinellifolium* genomes. *This is a wild species of tomato closely related to the* S. lycopersicum*, the cultivated tomato and the most numerous wild tomato species in the source data sets. The shared variants identified through intersection represent alleles typical of S. pimpinellifolium as compared to the cultivated tomato reference genome. See* Table 3-2 *for the numeric results.* 45

Figure 3-5: Tersect index build time and peak memory usage. *It should be noted that inclusion time per genome varied significantly due to different variant content per genome, evident in the shape of the line. This is also evident in* Figure 3-6*, which traces a very similar path for the input data. The source genome list was shuffled to minimize this variation. The peak memory usage* *is defined as the maximum resident set size. See* Table 3-3 *for the numeric results.* 46

Figure 3-6: Size of input data and generated Tersect index files. *Note that the sizes of individual per genome data sets vary with the number of variants they contain. See* Table 3-3 *for the numeric results.* 47

Figure 4-1: Diagram of the distance matrix precomputation process. *The size of the tomato reference genome chromosome 4 is used alongside four partition sizes: 5 Mbp (the smallest and most significant size), 10 Mbp, 25 Mbp, and 50 Mbp. Note that for the largest size, the partition in fact covers the entire, shorter length of the chromosome (47.26 Mbp) instead. Tersect is only used to create the smallest partitions, while the rest are generated by adding up sub-partitions. This requires all the partition sizes (which can be set by the user) to be multiples of the smallest partition size. The "distance" metrics stored in the matrices are actually simple counts of the number of SNV differences rather than true genetic distance metrics; those are calculated downstream.* 61

Figure 4-2: Diagram of distance matrix request handling by the Tersect Browser back-end. *The requested interval is partitioned into a list of smaller intervals, for which the distance matrices can be either retrieved from among the precomputed partitions or generated* de novo *by Tersect, and then added and subtracted to yield the final distance matrix. Only two (at most) Tersect intervals are ever generated for a single request (one at each end of the requested interval) and they are always (at most) half the size of the smallest precomputed partition. In addition, Tersect will only calculate distance matrices for selected accessions, while the stored, precomputed matrices have to be filtered to select the appropriate rows and columns. Note that the distance metrics used throughout the process are simply the substitution (SNV) counts, due to their ease of addition of subtraction. The actual genetic distance metrics (Jukes-Cantor distance) are only calculated for the final matrix, which is then used downstream for phylogeny inference.* 62

Figure 4-3: Phylogeny inference times for the entire 444-genome tomato data set and chromosomal intervals of different sizes as a function of the smallest precomputed partition size. *One hundred random intervals were generated for each of the tested interval sizes (from 1 Mbp to 50 Mbp). The same sets of intervals were used to test each precomputed partition size, with fifty requests executed for each combination of interval size and smallest partition size. The median response times were then recorded for each partition size. See* Table B-3 *for the numeric results* 71

Figure 4-4: Phylogeny inference times for the entire 2548-genome tomato data set and chromosomal intervals of different sizes as a function of the smallest precomputed partition size. *Fifty random intervals (all on chromosome 1) were generated for each of the tested interval sizes (from 1 Mbp to 100 Mbp). The same sets of intervals were used to test each precomputed partition size, with fifty requests executed for each combination of interval size and smallest *partition size. The median response times were then recorded for each partition size. See* Table B-4 *for the numeric results.* 72

Figure 4-5: Time and storage space costs of distance matrix precomputation as functions of the smallest partition size. *The 444 resequenced tomato genome data were used for benchmarking. To create successive partitions larger than the smallest, their size was doubled until it was larger than the largest chromosome (98.5 Mbp), which is the default approach taken by the data set addition script. The time measurements were recorded on a desktop PC and would scale according to CPU speed, but the storage size should remain invariant for a given data set and partition size. Both cost metrics exhibit a component that is inversely proportional to the partition size, but for time this is obscured by the mostly constant cost of running Tersect once per genomic interval, especially for larger partition sizes. These costs have to be weighed against the intended request handling speed (see* Figure 4-3*) when selecting the partition sizes for a particular data set. See* Table B-1 *for the numeric results and* Figure B-1 *for equivalent data measured for the human data set* 73

Figure 4-6: Response time as a function of the number of genomes used in phylogeny inference requests for the tomato (A) and human (B) data sets. *The smallest precomputed partition size used was 1 Mbp for both data sets. Twenty requests were made for each tested number of genomes, using random subsets of the total genome set and random chromosomal intervals (with the length of 10 Mbp) for each request.* 75

Figure 4-7: Potential introgression from *S. pim* LYC2798 or a similar donor into the LA2706 (MoneyMaker) and LYC1365 (AllRound) cultivars. *The introgression, visible against a background of other tomato cultivars, spans from approximately 40.50 Mbp to 42.45 Mbp on chromosome 6. Both affected cultivars cluster closely with the potential donor within a sharply delineated interval. Note that the Introgression Browser article reports the same introgression as beginning at 36.75 Mbp, but that is due to its use of an older version of the tomato reference (SL2.40).* 76

Figure 4-8: TGRC gene annotation used to highlight different alleles of the tomato *uniform ripening* gene in Tersect Browser visualisations. *The two alleles of the gene are marked in red (-- allele) and green (+ allele). The top plot covers the first 10 Mbp of chromosome 10, and its phylogenetic tree structure shows no obvious relationship with the alleles. The bottom plot covers a smaller interval on the same short arm of chromosome 10. The 200 kbp interval (2,195,000 – 2,395,000) is centred on the* uniform ripening *gene locus, and it can be seen that the accessions now cluster more closely according to their alleles.* 77

LIST OF TABLES

Table 2-1: Areas in the SGN-hosted resequenced genomes which contain variants requiring correction. *The areas correspond to the scaffolds which were reversed between versions SL2.40 and SL2.50 of the tomato reference genome.* 13

Table 2-2: Comparison of the order and orientation of chromosome 9 scaffolds between the SL2.40 (based on linkage mapping) genome build, the Shearer et al. article (based on BAC-FISH), and the SL2.50 genome build (based on the Shearer et al. article). *The "reversed" scaffolds were reversed relative to their orientation in the SL2.40 build.* 18

Table 2-3: Comparison of the order and orientation of chromosome 12 scaffolds between the SL2.40 genome build (based on linkage mapping), the Shearer et al. article (based on BAC-FISH), and the SL2.50 genome build (based on the Shearer et al. article). *The "reversed" scaffolds were reversed relative to their orientation in the SL2.40 build.* 19

Table 2-4: Comparison of inter-scaffold gap sizes between the SL2.50 genome build and the Shearer et al. BAC-FISH results. *The "SL2.50" gap sizes are based on the build AGP files, while the "BAC-FISH" gap sizes are based on the corrected gap size estimates from the Shearer et al. publication. For each pair of scaffolds, "SCAFFOLD A" is the one closer to the short arm of the chromosome, and "SCAFFOLD B" is the one closer to the long arm of the chromosome. The values of "0" for BAC-FISH replace negative values post-correction. "N/A" entries appear for pairs of scaffolds which are neighbours only in the SL2.50 assembly or the Shearer et al. article, but not in both; this is due to the issues with scaffold ordering on chromosome 9 described earlier in this section. Particularly large discrepancies in gap sizes are marked in bold.* 20

Table 2-5: Size of intervals (in kbp) mapped between pseudomolecules (destination chromosomes 0 to 6) during lift over from version SL2.40 of the tomato genome to SL4.0 using SeqRemap. 29

Table 2-6: Size of intervals (in kbp) mapped between pseudomolecules (destination chromosomes 7 to 12) during lift over from version SL2.40 of the tomato genome to SL4.0 using SeqRemap. 29

Table 3-1: Private variant identification benchmark results. 47

Table 3-2: Intersection benchmark results. 48

Table 3-3: Tersect index build metrics. 49

LIST OF ABBREVIATIONS

AGP	A Golden Path
API	Application programming interface
AST	Abstract syntax tree
BAC-FISH	Bacterial artificial chromosome fluorescent *in situ* hybridisation
BGI	Beijing Genomics Institute
CRISPR	Clustered regularly interspaced short palindromic repeats
CSS	Cascading Style Sheets
CSV	Comma-separated values
DOM	Document Object Model
EMS	Ethyl methanesulfonate
FISH	Fluorescent *in situ* hybridisation
GATK	Genome Analysis Toolkit
GBS	Genotyping-by-sequencing
IGSR	The International Genome Sample Resource
JSON	JavaScript Object Notation
KASP	Kompetitive allele specific PCR
NGS	Next-generation sequencing
ODM	Object Document Mapping
PNG	Portable Network Graphics
QTL	Quantitative trait locus
RAM	Random-access memory
REST	Representational state transfer
SGN	Sol Genomics Network
SIMD	Single instruction, multiple data
SNV	Single-nucleotide variant
TGRC	C.M. Rick Tomato Genetics Resource Center
VCF	Variant Call Format
WAH	Word-aligned hybrid (compression)
WHATWG	Web Hypertext Application Technology Working Group
XOR	Exclusive disjunction

1 INTRODUCTION

1.1 Background

Genome sequencing and assembly projects are of central importance to modern biological research, with the development of a reference genome for an organism representing a particularly essential milestone for research related to that organism. A reference genome is crucial not only because of the genetic information it contains, but also because it serves as an ostensibly static point of reference for other information. Annotations, resequenced genomes, and other data sets can be described in terms of the reference genome, using its features and coordinates to characterise and locate their own contents. However, due to the size, complexity, ploidy, and repetitiveness of most eukaryotic genomes, their *de novo* assemblies, based on current sequencing technologies, do not generally reach the stage of absolute “completeness”, and they often retain known imperfections such as gaps and scaffolds of unknown chromosomal position or orientation (Muñoz et al., 2013). They may also contain hidden errors such as mis-assemblies or contamination. As new data, platforms, and methods become available, these problems can be addressed, resulting in incremental improvements to the reference sequence (Sedlazeck et al., 2018). Still, this also means that a reference genome is not yet, in practical terms, a truly static point of reference.

Improvements to a reference genome, while generally desirable, invalidate derivative data sets which depend on a previous version of the sequence. To correct this, the data sets need to be updated to match the new reference. Large sets of resequenced genomes, such as those publicly available for tomato (S. Aflitos et al., 2014; Lin et al., 2014) and human (1000 Genomes Project Consortium, 2015) can pose particular difficulties here. Repeating the whole process of alignment and variant calling for each of the genomes whenever a new version of the reference is released, as has been done with human genomes from the 1000 Genomes Project (Zheng-Bradley et al., 2017), can be considered the gold standard approach, but it is also very computationally expensive, with the overall cost increasing with each new resequenced genome. This incentivises the use of quicker, less expensive solutions, such as simply converting the coordinates between assemblies, usually based on pairwise alignment between them, a process often called “lift-over”. Tools like NUCmer (Marçais et al., 2018) and CrossMap (Zhao et al., 2014) are commonly used for lift-over, as are custom solutions developed for particular genome assembly projects. However, as the lift-over process is potentially error-prone, extra care should be taken to validate the results and make note of any artefacts which may arise. Work undertaken as part of this thesis, detailed in **Chapter 2**, led to the discovery of significant errors introduced into resequenced tomato genome data through lift-over, which highlighted certain important considerations in the processing and validation of such large data sets.

Besides simple validation and correctness, a further consideration when dealing with large sets of resequenced genomes is the performance and flexibility of tools used to process them. In particular, there exist multiple tools that allow for comparing variant content between genomes, such as BCFtools (Danecek et al., 2021), BEDOPS (Neph et al., 2012), and BEDTools (Quinlan & Hall, 2010). However, they are not optimized for the use of large numbers of genomes, such as those hosted for tomato by SGN, offering limited flexibility in queries and slow performance in such scenarios. Addressing this gap by investigating algorithms suitable for flexible, high-performance variant content comparison between large numbers of resequenced genomes, and ultimately implementing them in a lightweight software tool, was a central goal of this work, culminating in the development of Tersect, detailed in **Chapter 3**.

Nevertheless, performance improvement is ultimately not a goal in itself. The real goal is improving productivity, and human time is more valuable than computer time, especially among skilled researchers. The high performance of Tersect, as a specialised command-line utility, is only accessible in a direct manner to bioinformaticians. In order to share its benefits with the larger biological research community, the tool needs to be integrated into some larger system. This reasoning is in fact inherent in the design of Tersect as a small, self-contained utility, following the UNIX-style system design principles of modularity and composability (Pike & Kernighan, 1984). In the academic context this approach has the additional benefit of isolating interesting research problems, such as (for Tersect) the development of highly efficient methods for the comparison of variant sets in resequenced genomes. Such self-contained software packages can be published with a sharp focus on the knowledge gaps being addressed, without the distraction of a larger, monolithic application which would, by necessity, contain many scientifically uninteresting elements as well. The software can subsequently be integrated into the aforementioned larger system (or multiple such systems), which can then address different research problems, while providing functionalities and, most importantly, interfaces that non-bioinformaticians can interact with to improve their own productivity.

It follows that an important threshold to be aimed at when it comes to performance improvement is one of interactivity. The change from working with a sequence of non-interactive (passive) visualisations over a longer time period to engaging with an interactive system is not trivial. The change is qualitative, rather than merely quantitative, even though making it possible depends on the entirely quantitative measure of latency, itself a product of the system's performance (Godfrey et al., 2016). With latencies on the order of seconds rather than minutes or hours, users can actively explore and engage with the data, with each transformation of the data informing the next while fresh in the user's mind. Sub-second latencies are commonly suggested in research on human-computer interaction as the benchmark for a fluid user experience (Shneiderman, 1984), although in practice this can be difficult to achieve with large data sets.

The above means that, depending on the context and distance from the interactivity threshold, minor performance improvements to relatively quick operations could, in some circumstances, be considered more significant than major performance improvements to slow operations. For example, speeding up the generation of a visualisation from 100 hours to 2 hours might not have as much of an impact as speeding one up from 100 seconds to 2 seconds, even though in absolute terms the performance improvement is much larger in the first scenario, while in relative terms the improvement is the same in both (a factor of fifty). In the former scenario, visualisation remains a passive process, and iterative work with the data would require scheduling separate sessions (at least 2 hours apart – likely longer in practice) to interpret results. In the latter, 2 seconds of latency may be sufficient for interactive work, while having to wait 100 seconds for each step would be too slow for active engagement.

A situation of this sort became evident during research to which the author was a contributor, which aimed to map a tomato gene involved in genetic control of inflorescence branching, *bifurcate flower truss* (Silva Ferreira et al., 2018). A tool called Introgression Browser was used to demonstrate the origin of the mutant allele of interest as an introgression from a wild species, *Solanum galapagense*. This was achieved through visualising the genetic distance and phylogenetic relationships in a mapped interval of interest between a mutant line resequenced by Cranfield University and publicly available data on resequenced tomato genomes, which included closely related wild species (S. Aflitos et al., 2014; Lin et al., 2014).

This was, however, an iterative process, as such visualisations are sensitive to the set of genomes used, as well as to the exact position of the interval and its segmentation. With each visualisation requiring command-line setup and several hours of computation, the overall process of creating the final images took a long time and required coordination and consultation between multiple collaborators. Steps were undertaken to automate the Introgression Browser plot generation setup and improve the speed of visualisation, but despite some performance improvements the overall process remained thoroughly non-interactive. It was clear that a different algorithmic approach was required to offer the same functionality to biologists as an interactive tool.

Because Tersect is well-optimized for the operations required to calculate genetic distance (and, consequently, to infer phylogenies), namely the enumeration of discordant sites, this problem was an ideal use case for the tool. Combined with other algorithmic approaches commonly used for interactive visualisation in large data sets (Godfrey et al., 2016), primarily the precomputation and aggregation of indexed partial results, it was used to develop Tersect Browser, an interactive

Web application for visualising genetic distance and phylogenetic relationships between large numbers of resequenced genomes, as detailed in **Chapter 4**. Alongside providing its core functionality to the biologist community, Tersect Browser would also serve as a testbed for assessing the limitations of this approach, which could suggest future algorithmic improvements.

1.2 Aim and objectives

The overall aim of this thesis was to develop flexible, high-performance bioinformatics tools for low-level manipulation of large sets of resequenced genomic data, alongside developing methods of interactively applying those tools to solving higher-level biological problems related to genome structure and phylogenetic relationships.

Three more specific objectives were defined as follows:

1. Investigate issues associated with maintaining the integrity of resequenced genome data sets across different types of processing to ensure the continued validity of data.
2. Develop and evaluate low-level software optimized for high-performance manipulation and comparison of large numbers of resequenced eukaryotic genomes.
3. Develop and evaluate a tool for comparing large numbers of resequenced genomes and interactively visualising the phylogenetic relationships and genetic distances across their chromosomal structures.

Note that the objectives depend on each other in sequence, with the second and third objectives relying on data validity investigated as part of the first objective, and the tool developed to fulfil the third objective relying on the lower-level software developed to fulfil the second objective.

The “evaluation” component of the second and third components relates to measuring their performance as well as the identification of potential limitations, which may inform future work on further improving the algorithms which were used.

1.3 Thesis outline

The thesis uses a “paper format”, with chapters 2, 3, and 4 being organised as distinct, self-contained publications forming part of a larger body of research. An overview of the contents of each chapter is given below:

Chapter 1 discusses the general background of the work, the research gaps addressed by the individual blocks of work described in the following chapters, and how those blocks fit together, as well as the specific aims and objectives as well as the structure of the thesis.

Chapter 2 comprises a report, formatted as a publication, which was circulated to address issues with publicly maintained resequenced tomato genome data sets, as well as certain apparent deficiencies of one version of the tomato reference genome. Errors introduced by a failed lift-over process are diagnosed in detail and solutions are provided. This chapter relates to **Objective 1**.

Chapter 3 presents a peer-reviewed publication which introduces Tersect, a high-performance set theoretical utility for exploring sequence variant data. This chapter relates to **Objective 2**, and the publication is cited below:

> *Tomasz J Kurowski, Fady Mohareb, Tersect: a set theoretical utility for exploring sequence variant data, Bioinformatics, Volume 36, Issue 3, 1 February 2020, Pages 934–935*

Chapter 4 presents a publication manuscript intended for submission in the near future. It introduces Tersect Browser, a Web application which leverages specialized bioinformatics solutions (including Tersect) to provide an interactive visualisation system for the comparison of large numbers of resequenced genomes. This chapter relates to **Objective 3**.

Chapter 5 constitutes an overall discussion of the work, with particular focus on the real-world applications of the individual outputs, reasoning about the limitations of the implemented solutions based on a critical evaluation of their performance, and considerations of likely future work and improvements.

1.4 References

Aflitos, S., Schijlen, E., De Jong, H., De Ridder, D., Smit, S., Finkers, R., Wang, J., Zhang, G., Li, N., Mao, L., Bakker, F., Dirks, R., Breit, T., Gravendeel, B., Huits, H., Struss, D., Swanson-Wagner, R., Van Leeuwen, H., Van Ham, R. C. H. J., … Peters, S. (2014). Exploring genetic variation in the tomato (Solanum section Lycopersicon) clade by whole-genome sequencing. *Plant Journal*, *80*(1), 136–148. https://doi.org/10.1111/tpj.12616

Consortium, G. P., Auton, A., Abecasis, G. R., Altshuler, D. M., Durbin, R. M., Abecasis, G. R., Bentley, D. R., Chakravarti, A., Clark, A. G., Donnelly, P., Eichler, E. E., Flicek, P., Gabriel, S. B., Gibbs, R. A., Green, E. D., Hurles, M. E., Knoppers, B. M., Korbel, J. O., Lander, E. S., … National Eye Institute, N. I. H. (2015). A global reference for human genetic variation. *Nature*, *526*(7571), 68–74. https://doi.org/10.1038/nature15393

Danecek, P., Bonfield, J. K., Liddle, J., Marshall, J., Ohan, V., Pollard, M. O., Whitwham, A., Keane, T., McCarthy, S. A., Davies, R. M., & Li, H. (2021). Twelve years of SAMtools and BCFtools. *GigaScience*, *10*(2). https://doi.org/10.1093/gigascience/giab008

Godfrey, P., Gryz, J., & Lasek, P. (2016). Interactive Visualization of Large Data Sets. *{IEEE} Transactions on Knowledge and Data Engineering*, *28*(8), 2142–2157. https://doi.org/10.1109/tkde.2016.2557324

Lin, T., Zhu, G., Zhang, J., Xu, X., Yu, Q., Zheng, Z., Zhang, Z., Lun, Y., Li, S., Wang, X., Huang, Z., Li, J., Zhang, C., Wang, T., Zhang, Y., Wang, A., Zhang, Y., Lin, K., Li, C., … Huang, S. (2014). Genomic analyses provide insights into the history of tomato breeding. *Nature Genetics*, *46*(11), 1220–1226. https://doi.org/10.1038/ng.3117\rhttp://www.nature.com/ng/journal/v46/n11/abs/ng.3117.html#supplementary-information

Marçais, G., Delcher, A. L., Phillippy, A. M., Coston, R., Salzberg, S. L., & Zimin, A. (2018). {MUMmer}4: A fast and versatile genome alignment system. *{PLOS} Computational Biology*, *14*(1), e1005944. https://doi.org/10.1371/journal.pcbi.1005944

Muñoz, J. F., Gallo, J. E., Misas, E., McEwen, J. G., & Clay, O. K. (2013). The eukaryotic genome, its reads, and the unfinished assembly. *FEBS Letters*, *587*(14), 2090–2093. https://doi.org/10.1016/j.febslet.2013.05.048

Neph, S., Kuehn, M. S., Reynolds, A. P., Haugen, E., Thurman, R. E., Johnson, A. K., Rynes, E., Maurano, M. T., Vierstra, J., Thomas, S., Sandstrom, R., Humbert, R., & Stamatoyannopoulos, J. A. (2012). BEDOPS: High-performance genomic feature operations. *Bioinformatics*, *28*(14), 1919–1920. https://doi.org/10.1093/bioinformatics/bts277

Pike, R., & Kernighan, B. W. (1984). {TheUNIXSystem}: Program Design in {theUNIXEnvironment}. *{AT}{\&}T Bell Laboratories Technical Journal*, *63*(8), 1595–1605. https://doi.org/10.1002/j.1538-7305.1984.tb00055.x

Quinlan, A. R., & Hall, I. M. (2010). BEDTools: A flexible suite of utilities for comparing genomic features. *Bioinformatics*, *26*(6), 841–842. https://doi.org/10.1093/bioinformatics/btq033

Sedlazeck, F. J., Lee, H., Darby, C. A., & Schatz, M. C. (2018). Piercing the dark matter: bioinformatics of long-range sequencing and mapping. *Nature Reviews Genetics*, *19*(6), 329–346. https://doi.org/10.1038/s41576-018-0003-4

Shneiderman, B. (1984). Response Time and Display Rate in Human Performance with Computers. *ACM Comput. Surv.*, *16*(3), 265–285. https://doi.org/10.1145/2514.2517

Silva Ferreira, D., Kevei, Z., Kurowski, T., de Noronha Fonseca, M. E., Mohareb, F., Boiteux, L. S., & Thompson, A. J. (2018). BIFURCATE FLOWER TRUSS: a novel locus controlling inflorescence branching in tomato contains a defective MAP kinase gene. *Journal of Experimental Botany*, *69*(10), 2581–2593. https://doi.org/10.1093/jxb/ery076

Zhao, H., Sun, Z., Wang, J., Huang, H., Kocher, J. P., & Wang, L. (2014). CrossMap: A versatile tool for coordinate conversion between genome assemblies. *Bioinformatics*, *30*(7), 1006–1007. https://doi.org/10.1093/bioinformatics/btt730

Zheng-Bradley, X., Streeter, I., Fairley, S., Richardson, D., Clarke, L., & Flicek, P. (2017). Alignment of 1000 Genomes Project reads to reference assembly GRCh38. *GigaScience*, *6*(7), 1–8. https://doi.org/10.1093/gigascience/gix038

2 CORRECTING AND VALIDATING VARIANT DATA LIFT-OVER IN RESEQUENCED TOMATO GENOMES

2.1 Abstract

2.1.1 Summary

The tomato reference genome hosted and maintained by the Sol Genomics Network is an invaluable resource for plant scientists. Sets of resequenced genomes of various tomato lines and close wild relatives mapped to that reference are also valuable, providing insights about the history of tomato evolution and breeding, as well as representing a repository of data on the genomic diversity in tomatoes and a point of comparison for researchers working with their own resequenced genomes. The tomato reference genome has seen several improved releases over the years. After it was updated from version SL2.40 to SL2.50, resequenced genomes hosted by SGN were updated to match the new version. However, due to the nature of the lift-over process used, this introduced errors into this publicly available repository, which went undetected and uncorrected for nearly seventeen months. This publication reports on the detection, diagnosis, and ultimately the correction of those lift-over errors, and discusses their primary causes. It also introduces SeqRemap, a Python utility based on the validation and correction scripts developed during this process, which allows for fast, multi-threaded updates of coordinates in large numbers of resequenced genomes.

2.1.2 Availability

SeqRemap was released under the MIT license and is freely available at https://bitbucket.org/cranfieldbix/seqremap.

2.2 Introduction

The tomato reference genome is hosted and maintained by the Sol Genomics Network and has seen continuous updates since the initial build of version SL1.00 in December 2009, although the version described in the initial article announcing the release of the tomato reference genome was SL2.40 (Sato et al., 2012). Three more versions have been released since then (SL2.50, SL3.00, and SL4.00), with the latest having been assembled *de novo* from PacBio long reads and scaffolded using Hi-C contact maps, as well as validated using Bionano optical maps and 10x linked-read sequences (Hosmani et al., 2019).

SGN also hosts and maintains two large sets of resequenced tomato data, one containing 84 tomato accessions from the 150 Tomato Genome ReSequencing Project (S. Aflitos et al., 2014), and the other containing 360 tomato accessions by the Agricultural Genomic Institute at Shenzhen (Lin et al., 2014). They are available for download as VCF files and viewable through a JBrowse genome browser, which is a resource widely consulted by biologists who conduct research on tomatoes, e.g., as a reference for designing KASP markers or CRISPR targets.

Both data sets have originally been generated using SL2.40 as the reference, but they were later updated to version SL2.50, which was released in February 2014. According to VCF metadata, this update happened in April 2015, and both updated data sets were then shared with the research community. A custom tool (Bio-GenomeUpdate, https://github.com/solgenomics/Bio-GenomeUpdate) was used to update (lift) variant coordinates from SL2.40 to SL2.50.

The gold standard approach for updating variants to a different version of an assembly would be to simply repeat the entire alignment and variant calling pipeline, as has been done for human genomes from the 1000 Genomes Project, which were eventually remapped from their original GRCh37 reference to GRCh38 (Zheng-Bradley et al., 2017), although it should be noted that before that happened, dbSNP also simply lifted the variants to GRCh38 coordinates, which served as a temporary solution. Remapping is computationally expensive, especially for large numbers of genomes, but it avoids potential issues with the simple lift-over of coordinates, such as old variant calls no longer matching where the reference underwent significant structural or sequence changes, or regions which were not present in the previous version missing variants which would have been called at those locations.

However, the use of coordinate lift-over was more justified in the update of variants from SL2.40 to SL2.50 than it would be in most circumstances. The reference improvements were based on BAC-FISH and optical mapping results, which were aimed at establishing a more accurate ordering of scaffold sequences, as well as determining their orientation, and estimating the size of gaps between scaffolds (Shearer et al., 2014).

Therefore, the reference version update involved no change in any contiguous sequences and no sequence data were added or removed. The only changes were scaffolds changing position and orientation, and gap sizes being increased from an invariant, placeholder representation of one hundred unknown (N) bases placed between each pair of scaffolds to much larger estimates (the total pseudomolecule length increased from 781.67 Mbp to 823.94 Mbp, but all the added length consisted of unknown bases, representing gaps of a now known size). In principle, this meant that coordinate lift-over would be safe from the aforementioned issues and essentially equivalent to a complete remapping of the data, at a much lower computational cost.

Early work on a variant set comparison utility, eventually released as Tersect (Kurowski & Mohareb, 2019), identified discrepancies in the resequenced tomato genomes hosted by SGN. These discrepancies were investigated, leading to a discovery that the lift-over process was not executed correctly, and the variants listed were in fact incompatible with the SL2.50 reference genome in a number of locations. Additionally, certain differences between the SL2.50 reference and the published work it was based on were also identified.

This report documents the issues which were detected, the means of their diagnosis, the steps taken to correct them, and the tools developed for this purpose. Its initial version was compiled to notify SGN of the issue and to assist in correcting the problems with the resequenced tomato data.

2.3 Resequenced genome issues

When novel resequencing data sets generated at Cranfield University and based on the SL2.50 tomato reference were compared to resequenced genomes hosted by SGN it was discovered that in most chromosomes there exist large and very sharply delineated regions (see **Figure 2-1**) where the Cranfield results did not share a single variant with any of the hundreds of available genomes.

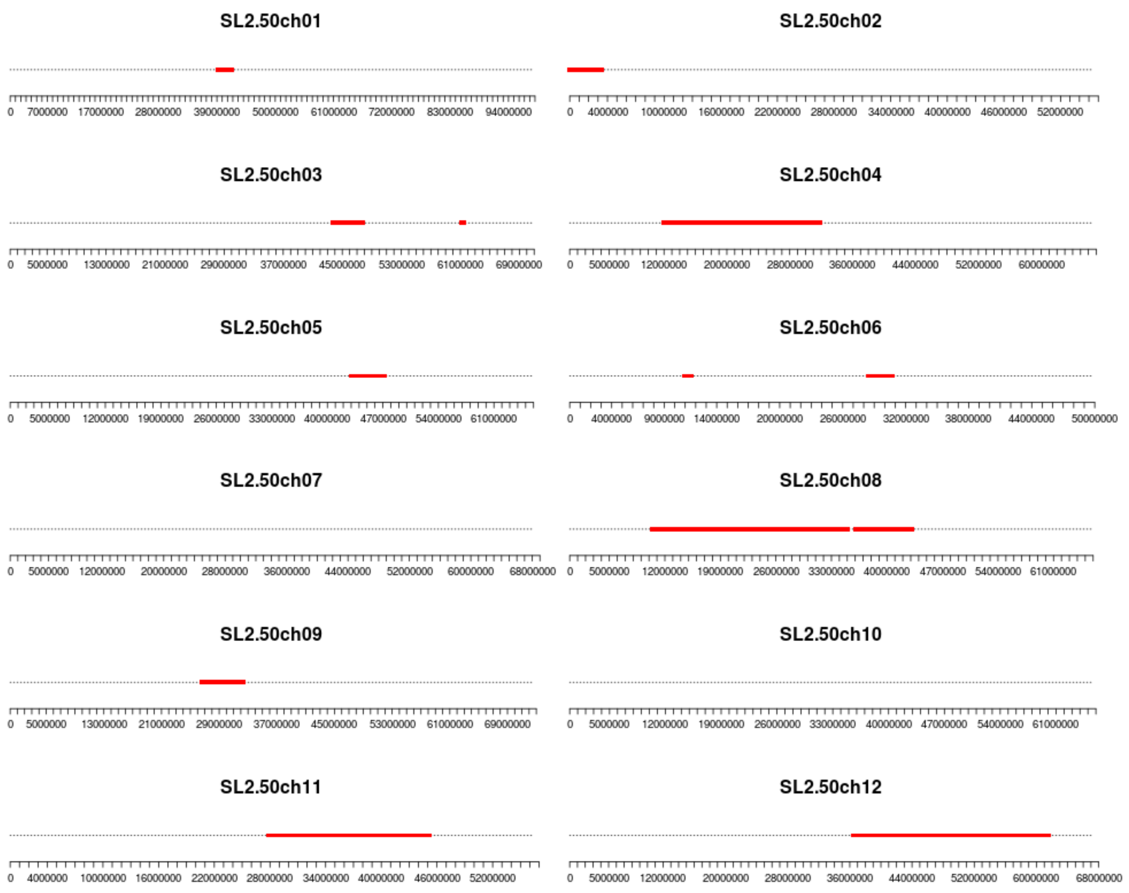


Figure 2-1: Positions of "problem areas" where the reference alleles listed in the resequenced genome VCF files hosted by SGN do not match the SL2.50 sequence, marked in red along the length of chromosomes.

Through a direct comparison – initially manual, then automated and confirmed via a command shell script – between the data fields contained in the VCF files hosted by SGN and the sequences in the reference FASTA files, it was found that the reference sequence (REF) fields of variants in the problematic regions do not match the SL2.50 reference genome, indicating that they are invalid. While the REF sequences matched a complementary sequence on the other strand, the VCF specification does not provide any way to indicate strandedness and all variants in VCF files are generally assumed to refer to the forward reference sequence strand.

It was then observed that the problematic regions correspond to scaffolds which were reversed when the genome was updated from version SL2.40 to SL2.50 (see **Figure 2-2**, **Figure 2-3**, and **Figure 2-4**). It became apparent that while VCF variant coordinates were updated based on the scaffold re-ordering and re-orientation, the REF and alternate allele (ALT) sequences of variants in re-oriented regions have not been modified to account for the fact that the sequence they were originally based on was now on the complementary strand.

The problem areas are listed in **Table 2-1** using SL2.50 coordinates. The SGN-hosted variants within those regions required correction.

Table 2-1: Areas in the SGN-hosted resequenced genomes which contain variants requiring correction. *The areas correspond to the scaffolds which were reversed between versions SL2.40 and SL2.50 of the tomato reference genome.*

CHROMOSOME	SCAFFOLD	START POSITION	END POSITION	SIZE
1	SL2.40sc03666	39,120,195	41,754,221	2.6 Mbp
2	SL2.40sc04732	1	1,697,214	1.7 Mbp
2	SL2.40sc04208	2,039,815	3,448,441	1.4 Mbp
3	SL2.40sc04822	43,731,686	47,848,114	4.1 Mbp
3	SL2.40sc06911	61,146,040	61,496,644	0.4 Mbp
4	SL2.40sc05339	11,890,154	13,863,001	2.0 Mbp
4	SL2.40sc03683	14,334,282	31,929,985	17.6 Mbp
5	SL2.40sc06155	42,974,962	47,280,334	4.3 Mbp
6	SL2.40sc06140	10,945,928	11,636,816	0.7 Mbp
6	SL2.40sc05188	28,400,004	30,805,544	2.4 Mbp
8	SL2.40sc03749	10,420,999	11,910,359	1.5 Mbp
8	SL2.40sc04236	12,010,360	26,652,460	14.6 Mbp
8	SL2.40sc03835	26,690,811	35,108,918	8.4 Mbp
8	SL2.40sc04701	36,082,319	43,105,760	7 Mbp
9	SL2.40sc04950	26,613,875	32,385,151	5.8 Mbp
11	SL2.40sc03752	27,858,571	45,260,389	17.4 Mbp
12	SL2.40sc04057	36,352,265	61,516,839	25.0 Mbp

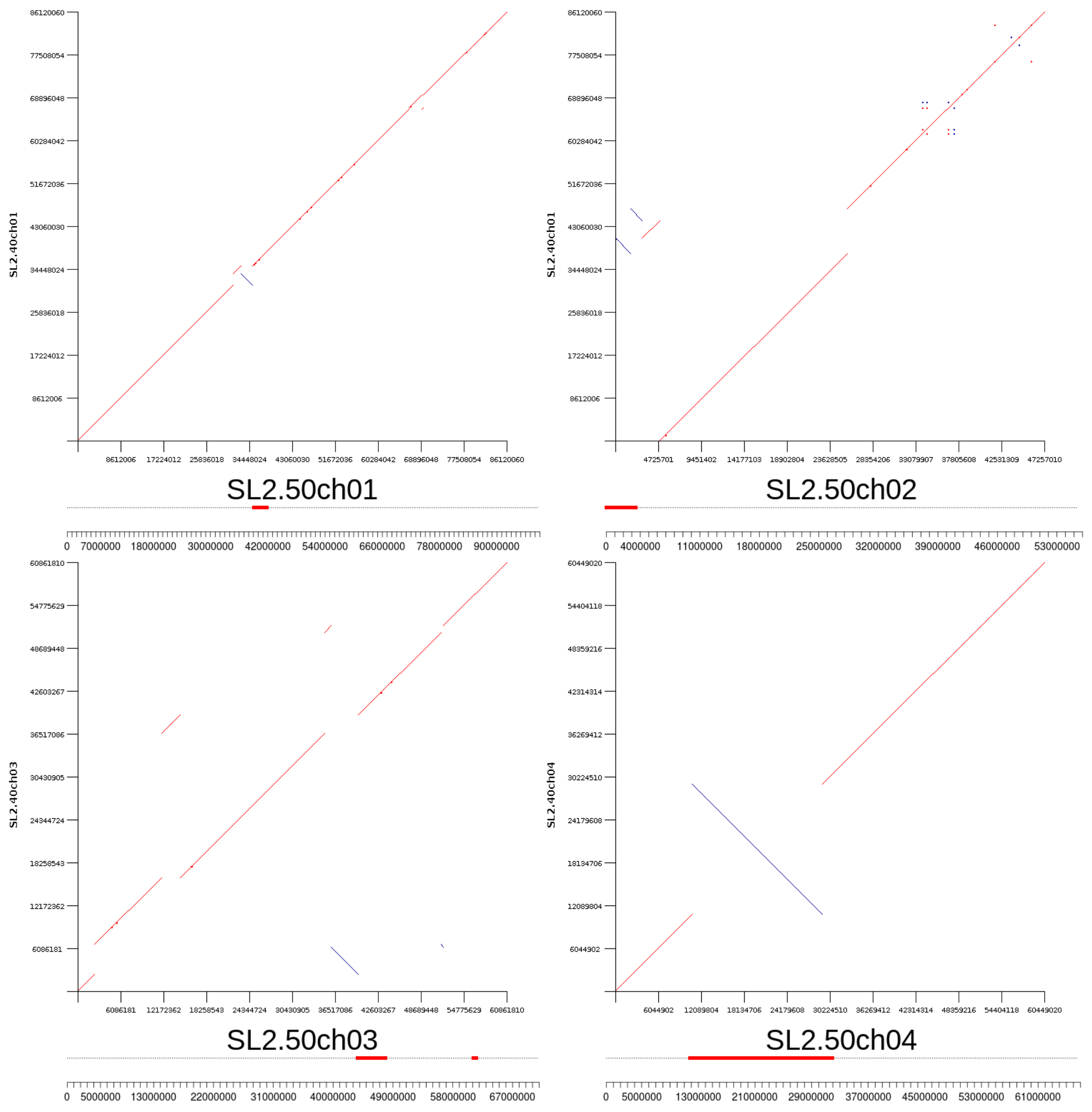


Figure 2-2: Filtered dot plots comparing the SL2.50 (x-axis) and SL2.40 (y-axis) sequences of chromosomes 1, 2, 3, and 4 alongside respective "problem areas". *The blue lines represent reverse complement sequences, indicating the presence of reversed scaffolds. The two reversed sequences at the start of chromosome 2 effectively form a single "problem area".*

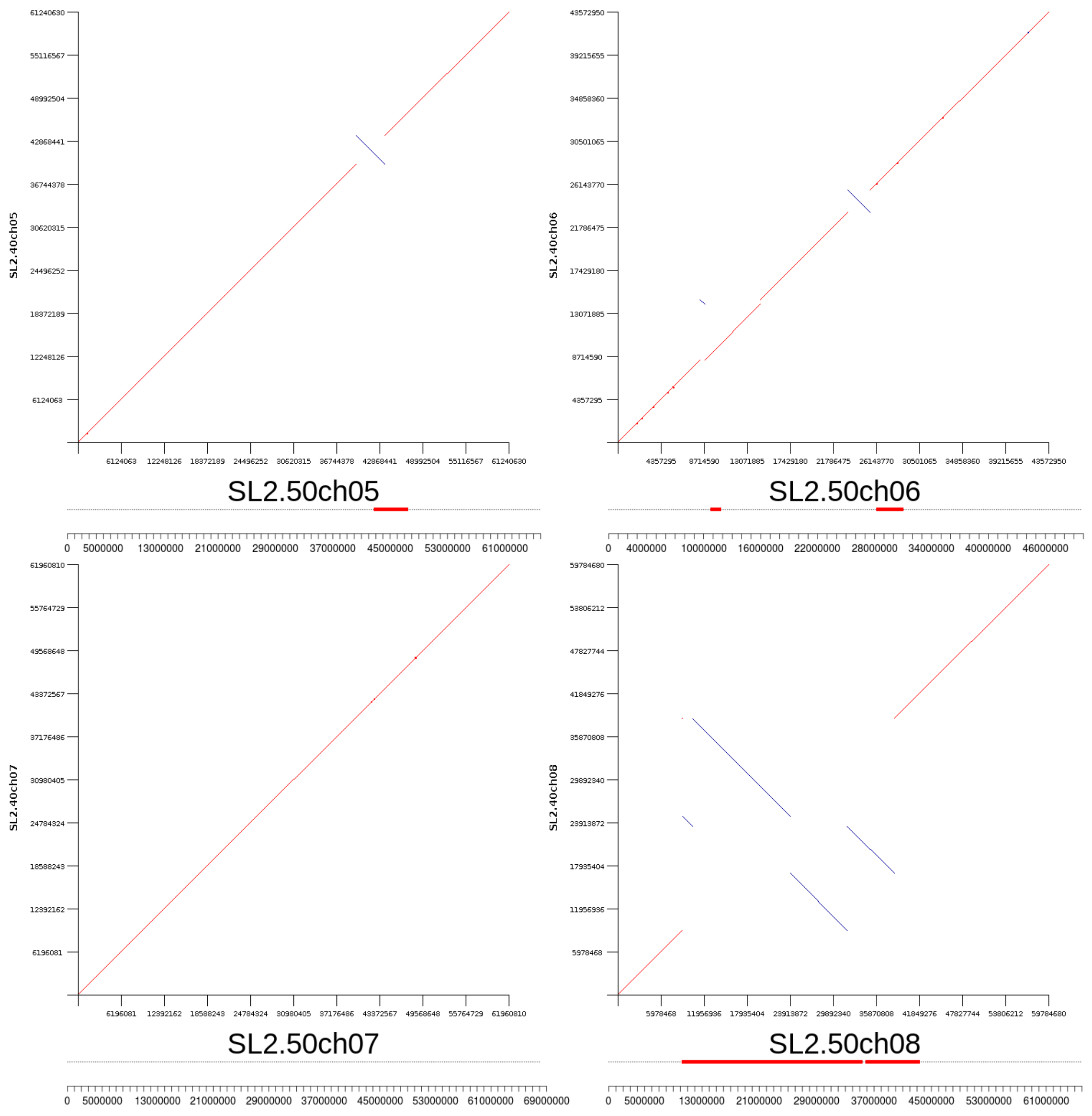


Figure 2-3: Filtered dot plots comparing the SL2.50 (x-axis) and SL2.40 (y-axis) sequences of chromosomes 5, 6, 7, and 8 alongside respective "problem areas". *The blue lines represent reverse complement sequences, indicating the presence of reversed scaffolds. Note that chromosome 7 includes no reversed scaffolds and thus no problem areas. The group of four neighbouring reversed scaffolds in chromosome 8 form a largely contiguous "problem area".*

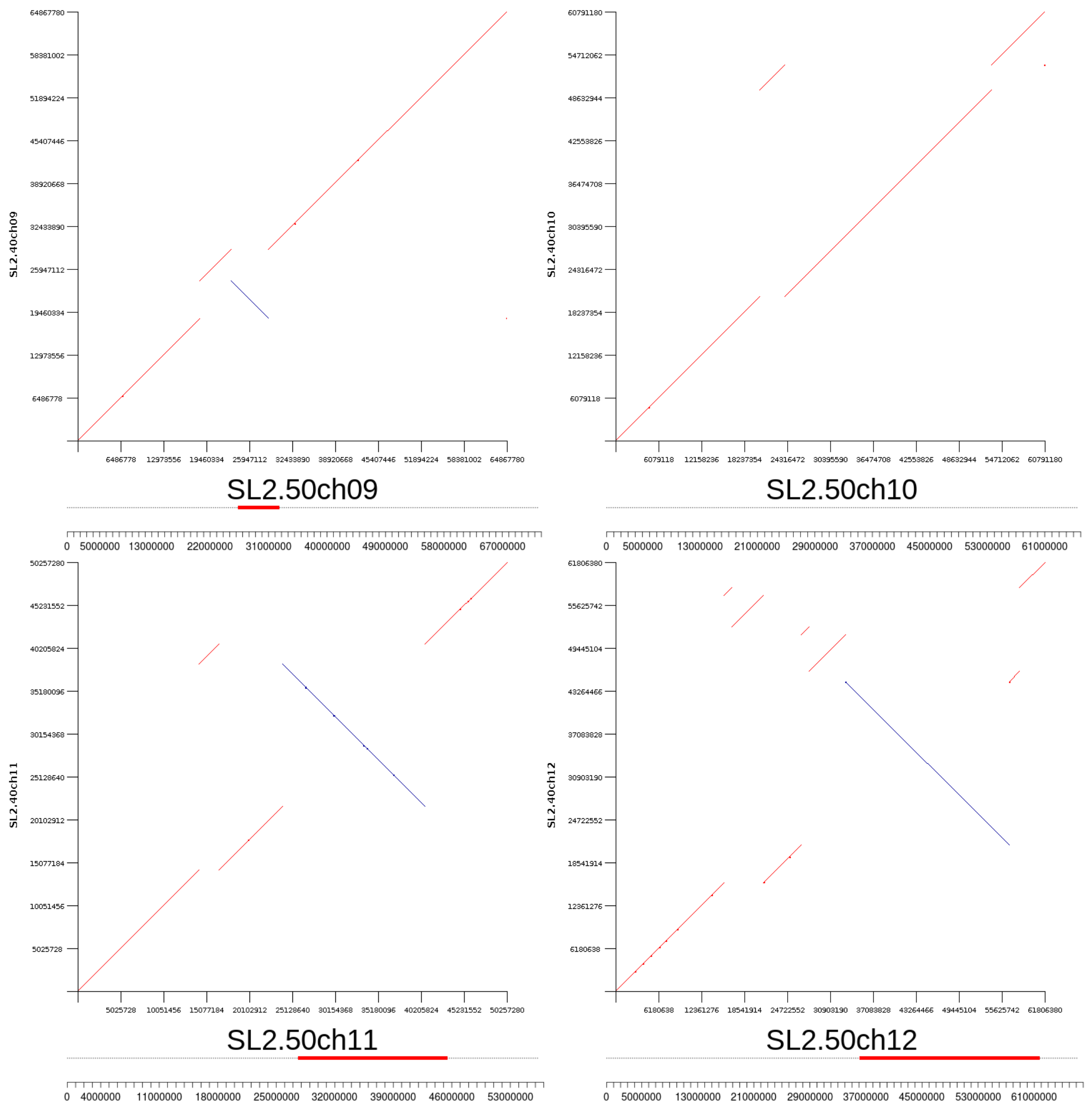


Figure 2-4: Filtered dot plots comparing the SL2.50 (x-axis) and SL2.40 (y-axis) sequences of chromosomes 9, 10, 11, and 12 alongside respective "problem areas". *The blue lines represent reverse complement sequences, indicating the presence of reversed scaffolds. Chromosome 10 includes re-arranged scaffolds but no reversed scaffolds and as a result it exhibits no "problem areas". The largest scaffold in chromosome 12 was reversed; this may be an error in the SL2.50 build (see* **Reference genome issues***).*

2.4 Reference genome issues

During the investigation of the origins of the VCF file issues described in this report, two discrepancies in the positions and order of scaffolds between the SL2.50 build of the tomato reference genome and the paper that the genome build was based on were found. In addition, in multiple locations the SGN document describing the changes (SGN, 2014) matches neither the Shearer et al. paper nor the SL2.50 build, which suggests that the differences are the result of errors in the genome update process rather than justified but undocumented changes. The discrepancies were found in genomes 9 and 12 and are described below.

As shown in **Figure 2-5** and detailed in **Table 2-2**, in chromosome 9 as present in the SL2.50 build of the reference genome, the smallest (93 kbp) scaffold SL2.40sc06916 is located at the very end, while the Shearer et al. BAC-FISH results indicated that its position should be between scaffolds SL2.40sc04777 and SL2.40sc05269.

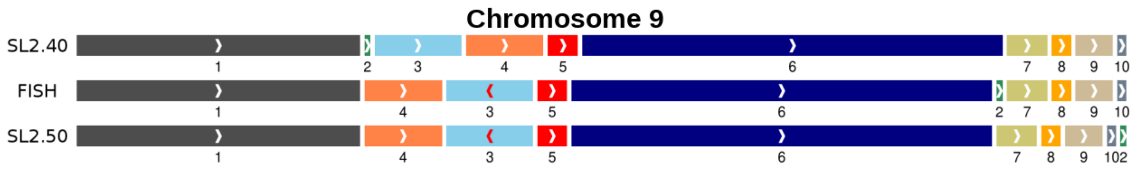


Figure 2-5: Diagram of the Chromosome 9 scaffold order and orientation in the SL2.40 reference, according to the Shearer et al. paper (FISH), and in the SL2.50 reference. *Reversed red arrows indicate scaffolds which are reversed compared to SL2.40. The relative scaffold sizes are to scale except for scaffold 2 (SL2.40sc6916) which would have been too small to see clearly. Gap sizes are omitted.*

The discrepancy in chromosome 12 seems much more significant. As shown in **Figure 2-6** and detailed in **Table 2-3**, it appears that while the scaffold order is consistent with the one suggested by Shearer et al., scaffold SL2.40sc04057 (the largest in the chromosome, at 25.2 Mbp) was reversed instead of scaffold SL2.40sc04039, resulting in two large sections of chromosome 12 being oriented incorrectly.

Table 2-2: Comparison of the order and orientation of chromosome 9 scaffolds between the SL2.40 (based on linkage mapping) genome build, the Shearer et al. article (based on BAC-FISH), and the SL2.50 genome build (based on the Shearer et al. article). *The "reversed" scaffolds were reversed relative to their orientation in the SL2.40 build.*

SCAFFOLD #	SCAFFOLD	SIZE	SCAFFOLD ORDER		
			SL2.40	BAC-FISH	SL2.50
1	SL2.40sc03771	19 Mbp	1	1	1
2	SL2.40sc06916	0.1 Mbp	2	4	4
3	SL2.40sc04950	5.8 Mbp	3	**3 reversed**	**3 reversed**
4	SL2.40sc04008	5.2 Mbp	4	5	5
5	SL2.40sc04785	2.0 Mbp	5	6	6
6	SL2.40sc04777	28.2 Mbp	6	2	7
7	SL2.40sc05269	2.7 Mbp	7	7	8
8	SL2.40sc03852	1.3 Mbp	8	8	9
9	SL2.40sc04828	2.5 Mbp	9	9	10
10	SL2.40sc06214	0.6 Mbp	10	10	2

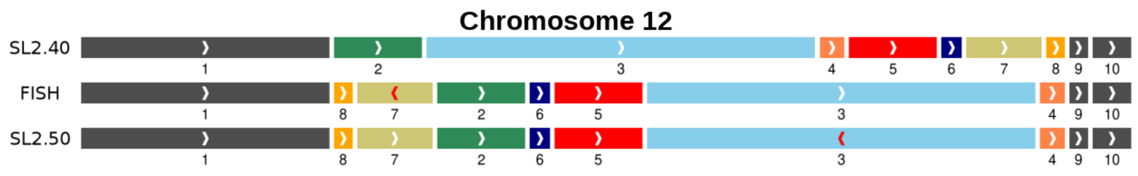


Figure 2-6: Diagram of the Chromosome 12 scaffold order and orientation in the SL2.40 reference, according to the Shearer et al. paper (FISH), and in the SL2.50 reference. *Reversed red arrows indicate scaffolds which are reversed compared to SL2.40. The relative scaffold sizes are to scale. Gap sizes are omitted.*

The order of the scaffolds suggests that the issue might have been caused by an improper order of operations when the chromosome was updated, as it was the third scaffold which was reversed – but it was the third scaffold *after* re-ordering, instead of the third scaffold *before* re-ordering.

The inter-scaffold gap sizes estimated by Shearer et al. were also found not to be fully consistent with the ones present in the SL2.50 build. This can be seen in **Table 2-4**. Most of the discrepancies appear to be due to rounding errors, as the values reported by Shearer et al. were rounded to the nearest 100 kbp, while the values included in the SL2.50 build appear to have been re-calculated to a higher level of precision. However, for six of the gaps the discrepancies are larger than 200 kbp, suggesting that rounding may not be their only cause. Additionally, the convention used by the SL2.50 release for gaps which are very small or, possibly, of an unknown size, appears to be applied inconsistently. Some of such gaps were given the size of 100 bp, which is a common convention suggested by the AGP format specification (NCBI, 2019), while others were given the size of 100 kbp.

Table 2-3: Comparison of the order and orientation of chromosome 12 scaffolds between the SL2.40 genome build (based on linkage mapping), the Shearer et al. article (based on BAC-FISH), and the SL2.50 genome build (based on the Shearer et al. article). *The "reversed" scaffolds were reversed relative to their orientation in the SL2.40 build.*

SCAFFOLD #	SCAFFOLD	SIZE	SCAFFOLD ORDER		
			SL2.40	BAC-FISH	SL2.50
1	SL2.40sc04607	16.1 Mbp	1	1	1
2	SL2.40sc04878	5.7 Mbp	2	8	8
3	SL2.40sc04057	25.2 Mbp	3	**7 reversed**	7
4	SL2.40sc04915	1.6 Mbp	4	2	2
5	SL2.40sc04757	5.7 Mbp	5	6	6
6	SL2.40sc04266	1.3 Mbp	6	5	5
7	SL2.40sc04039	4.9 Mbp	7	3	**3 reversed**
8	SL2.40sc06147	1.2 Mbp	8	4	4
9	SL2.40sc05611	1.2 Mbp	9	9	9
10	SL2.40sc05380	2.5 Mbp	10	10	10

It is difficult to say whether the gap size differences described in the previous paragraph are the result of errors or informed data curation, as the SL2.50 release makes no reference to them (SGN, 2014). However, they are not directly related to the lift-over errors addressed in this work, as they result only in shifts in position, without any sequence re-orientation.

Table 2-4: Comparison of inter-scaffold gap sizes between the SL2.50 genome build and the Shearer et al. BAC-FISH results. *The "SL2.50" gap sizes are based on the build AGP files, while the "BAC-FISH" gap sizes are based on the corrected gap size estimates from the Shearer et al. publication. For each pair of scaffolds, "SCAFFOLD A" is the one closer to the short arm of the chromosome, and "SCAFFOLD B" is the one closer to the long arm of the chromosome. The values of "0" for BAC-FISH replace negative values post-correction. "N/A" entries appear for pairs of scaffolds which are neighbours only in the SL2.50 assembly or the Shearer et al. article, but not in both; this is due to the issues with scaffold ordering on chromosome 9 described earlier in this section. Particularly large discrepancies in gap sizes are marked in bold.*

CHROMOSOME #	SCAFFOLD A	SCAFFOLD B	GAP [bp]	
			SL2.50	BAC-FISH
1	SL2.40sc04133	SL2.40sc04191	2280000	2400000
1	**SL2.40sc04191**	**SL2.40sc03666**	**2130000**	**1900000**
1	SL2.40sc03666	SL2.40sc03594	570000	600000
1	SL2.40sc03594	SL2.40sc05010	2120000	2200000
1	**SL2.40sc05010**	**SL2.40sc05941**	**510000**	**300000**
1	SL2.40sc05941	SL2.40sc06917	250000	300000
1	SL2.40sc06917	SL2.40sc06903	170000	200000
1	SL2.40sc06903	SL2.40sc04323	210000	200000
2	SL2.40sc04732	SL2.40sc04208	342600	400000
2	SL2.40sc04208	SL2.40sc05776	100000	0
2	SL2.40sc05776	SL2.40sc06593	100000	0
2	SL2.40sc06593	SL2.40sc04142	3046250	3200000
2	SL2.40sc04142	SL2.40sc03766	493900	500000
2	SL2.40sc03766	SL2.40sc03665	1340000	1400000
3	SL2.40sc04439	SL2.40sc04696	87200	100000
3	SL2.40sc04696	SL2.40sc05330	100000	0
3	SL2.40sc05330	SL2.40sc04126	316000	300000
3	**SL2.40sc04126**	**SL2.40sc04616**	**2580500**	**4700000**
3	SL2.40sc04616	SL2.40sc06725	163800	200000
3	SL2.40sc06725	SL2.40sc04704	741050	800000
3	**SL2.40sc04704**	**SL2.40sc03721**	**1094500**	**200000**
3	SL2.40sc03721	SL2.40sc04822	100	0
3	SL2.40sc04822	SL2.40sc03806	615000	600000
3	SL2.40sc03806	SL2.40sc03796	100000	0
3	SL2.40sc03796	SL2.40sc06911	70000	100000
3	SL2.40sc06911	SL2.40sc03701	80000	100000
4	SL2.40sc03604	SL2.40sc05339	296850	300000
4	SL2.40sc05339	SL2.40sc03683	471280	400000
4	SL2.40sc03683	SL2.40sc06101	84400	100000
4	SL2.40sc06101	SL2.40sc04680	29400	0
4	SL2.40sc04680	SL2.40sc04135	1525200	1600000
5	SL2.40sc03726	SL2.40sc06155	853750	900000

5	**SL2.40sc06155**	**SL2.40sc03902**	**100**	**800000**
6	SL2.40sc04474	SL2.40sc06140	2344700	2400000
6	SL2.40sc06140	SL2.40sc05383	616250	600000
6	SL2.40sc05383	SL2.40sc04279	349750	400000
6	SL2.40sc04279	SL2.40sc05188	100000	0
6	SL2.40sc05188	SL2.40sc05732	100000	0
6	SL2.40sc05732	SL2.40sc05054	100000	0
6	SL2.40sc05054	SL2.40sc03622	100000	0
7	SL2.40sc03731	SL2.40sc05397	385300	400000
7	SL2.40sc05397	SL2.40sc03685	2291400	2400000
7	SL2.40sc03685	SL2.40sc04626	100000	0
8	SL2.40sc04813	SL2.40sc03770	100000	0
8	SL2.40sc03770	SL2.40sc04167	518500	500000
8	SL2.40sc04167	SL2.40sc03749	259000	300000
8	SL2.40sc03749	SL2.40sc04236	100000	0
8	SL2.40sc04236	SL2.40sc03835	38350	50000
8	SL2.40sc03835	SL2.40sc04701	973400	1000000
8	SL2.40sc04701	SL2.40sc04948	802300	800000
8	SL2.40sc04948	SL2.40sc03923	43250	0
9	SL2.40sc03771	SL2.40sc04008	1466000	1500000
9	SL2.40sc04008	SL2.40sc04950	725500	800000
9	SL2.40sc04950	SL2.40sc04785	1053250	1100000
9	SL2.40sc04785	SL2.40sc04777	1250800	1300000
9	SL2.40sc04777	SL2.40sc05269	100000	N/A
9	SL2.40sc05269	SL2.40sc03852	100000	0
9	SL2.40sc03852	SL2.40sc04828	49950	0
9	SL2.40sc04828	SL2.40sc06214	75300	100000
9	SL2.40sc06214	SL2.40sc06916	100	N/A
9	SL2.40sc04777	SL2.40sc06916	N/A	0
9	SL2.40sc06916	SL2.40sc05269	N/A	0
10	SL2.40sc05925	SL2.40sc03798	100000	0
10	SL2.40sc03798	SL2.40sc04872	393600	400000
10	SL2.40sc04872	SL2.40sc05632	100000	0
10	SL2.40sc05632	SL2.40sc04199	100000	100000
10	SL2.40sc04199	SL2.40sc04534	100	0
11	SL2.40sc03748	SL2.40sc06763	353000	400000
11	SL2.40sc06763	SL2.40sc04054	534000	600000
11	SL2.40sc04054	SL2.40sc03752	1310000	1400000
11	SL2.40sc03752	SL2.40sc06137	70000	100000
11	SL2.40sc06137	SL2.40sc03876	650000	700000
12	SL2.40sc04607	SL2.40sc06147	175000	200000
12	SL2.40sc06147	SL2.40sc04039	100000	0
12	SL2.40sc04039	SL2.40sc04878	86000	100000
12	SL2.40sc04878	SL2.40sc04266	554200	600000
12	SL2.40sc04266	SL2.40sc04757	474550	500000
12	**SL2.40sc04757**	**SL2.40sc04057**	**100**	**800000**
12	SL2.40sc04057	SL2.40sc04915	100000	0
12	SL2.40sc04915	SL2.40sc05611	70000	100000

12	SL2.40sc05611	SL2.40sc05380	100000	0

2.5 Results

2.5.1 SNV corrections

The correction of SNVs was trivial, as it required only the change of REF and ALT alleles to complementary bases. No position adjustment was necessary since the position (POS) column refers directly to the affected base, and no actual sequence reversal was needed due to only individual bases being altered.

The VCF files for the BGI 360 genomes contain only SNVs with no other type of variants included. This made correcting them straightforward, as the only modification required was changing allele sequences to their complements. This can be seen in **Figure 2-7**, where the C→T variant required a correction to G→A.

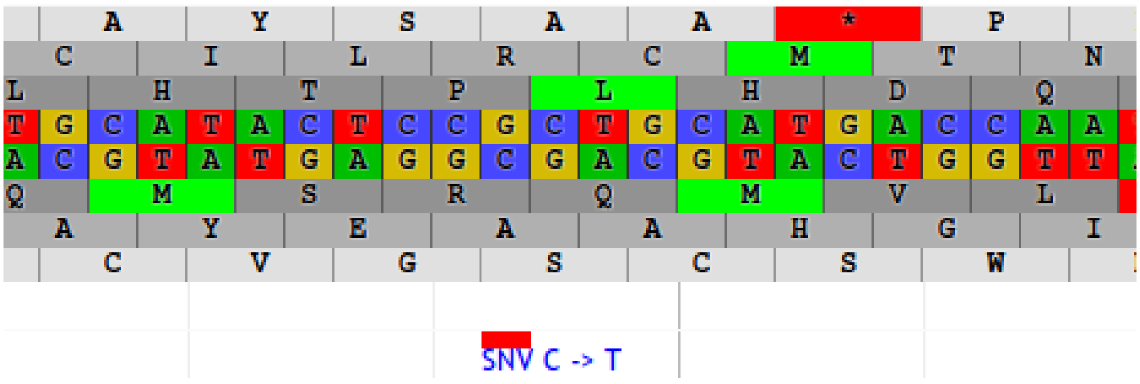


Figure 2-7: Example of an incorrect SNV in the SGN JBrowse genome browser. *The reference allele C matches the base on the complementary (bottom) DNA strand rather than the reference strand.*

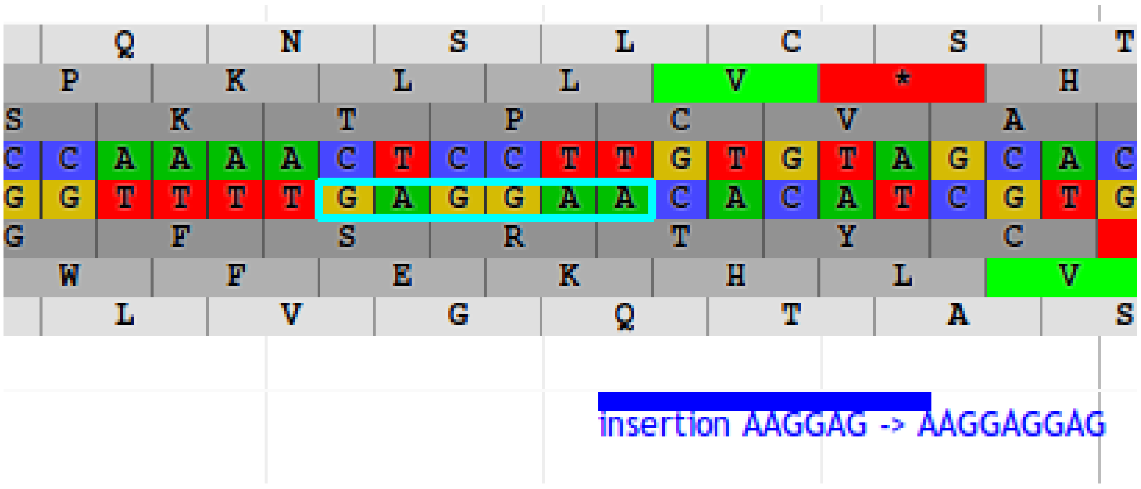
2.5.2 InDel corrections

Correcting insertion and deletion (InDel) variants is considerably more complicated than correcting SNVs. In fact, VCF files themselves do not contain all the data necessary to make such corrections. This is due to the fact that the VCF format specification requires the affected sequence given in the REF and ALT fields to be preceded by at least one base not affected by the InDel event.

In order to be corrected, InDel variants had their positions shifted towards the beginning of the chromosome by the number of bases in their reference alleles.

The first base of each REF and ALT allele was removed and the remaining sequences were replaced with their reverse complements. Finally, the allele sequences were prefixed with the preceding base retrieved from the SL2.50 reference genome.

Thus, the variant in **Figure 2-8** was corrected from AAGGAG→AAGGAGGAG to ACTCCT→ACTCCTCCT and had its position shifted by six bases towards the beginning of the chromosome.



Q N S L C S T
P K L L V * H
S K T P C V A
C C A A A A C T C C T T G T G T A G C A C
G G T T T T G A G G A A C A C A T C G T G
G F S R T Y C
W F E K H L V
L V G Q T A S
insertion AAGGAG -> AAGGAGGAG

Figure 2-8: Example of an incorrect InDel in the SGN JBrowse genome browser. *The reference allele matches the reverse of its preceding sequence (highlighted in cyan) on the complementary strand, showing that both the allele sequences and their positions need to be corrected.*

As mentioned in the previous section, the BGI 360 genome VCF files do not contain any InDel variants, so this section and the associated corrections were only necessary for the 84 accessions from the 150 Tomato Genome ReSequencing Project (S. Aflitos et al., 2014; Lin et al., 2014).

2.5.3 Annotation

The BGI 360 genomes hosted by SGN contain no annotation data, but those from the 150 Tomato Genome Resequencing Project have been annotated before their mapping to SL2.50 (S. Aflitos et al., 2014). As this annotation was based on the SL2.40 sequence and used the old 'EFF field' standard (since replaced by the

'ANN field' standard), it was removed. All of the VCF files were then re-annotated using SnpEff (Cingolani et al., 2012), based on the SL2.50 annotation database.

2.5.4 SeqRemap lift-over pipeline

A Python utility called SeqRemap was implemented based on the validation and correction scripts used to solve the issues described in this report. It was developed to allow for multi-threaded, concurrent lift-over of variant coordinates between two versions of a reference genome in large numbers of VCF files at once. Its lift-over functionality can be used both with simple, exact contig matches (as in the use case described in this report) and in more complicated scenarios requiring whole genome alignment, in which case it attempts to address a major shortcoming of lift-over by filling in the gaps created by novel reference sequences being introduced. This is achieved through running a specialised variant calling pipeline on affected intervals. SeqRemap was used to perform lift-over of the 150 Tomato Genome Resequencing Project to the latest (SL4.0) version of the tomato genome for internal use at Cranfield University.

SeqRemap is a Python-based lift-over tool capable of updating VCF variant files between different reference genomes (primarily different versions of the same reference genome), transforming variant positions and REF / ALT allele bases (if required) either on exact matches between contigs or on whole-genome alignments generated by NUCmer. The former approach is particularly appropriate for work-in-progress eukaryotic genomes whose anchoring and pseudomolecule structure, i.e., the position and orientation of contigs and scaffolds may be subject to change without significant changes to sequence contents. The latter approach is much slower (as it requires whole-genome alignment), but capable of handling significant differences in sequence structure and contents, such as those between independent *de novo* genome assemblies, including those of different, but related, species. A notable limitation in the latter approach, and in lift-over algorithms in general, is that novel sequences introduced in the destination genome, such as gaps closed in a new release of a reference, will be devoid of any variants in the updated VCF files, as they were absent in the source genome and could not have any reads mapped to them in the original variant calling pipeline.

SeqRemap seeks to address the aforementioned limitation by using a "gap patching" approach, in which a limited variant calling pipeline is executed on the novel intervals to "patch" the gaps in VCF files. The process is shown in the context of the full pipeline in Error! Reference source not found..

Whichever approach is chosen, SeqRemap is designed to allow for rapidly updating large numbers of VCF files in parallel through multiprocessing.

2.5.4.1 Required inputs

Besides the VCF files to be updated, SeqRemap requires the source and destination reference genome sequences as FASTA inputs. These are used to identify positional mappings between the two genomes, either through comparing sequence hashes (for exact matching) or through whole-genome alignment. As individual steps in the pipeline can be executed as separate scripts, it is technically possible to provide an externally generated NUCmer *delta* (or *coords*) file and skip the whole-genome alignment step of the pipeline. While this makes the source genome FASTA unnecessary, as it is not used in subsequent steps, the destination genome FASTA is always required.

If the optional "gap patching" functionality is to be used, the BAM read alignment files used in the original variant calling pipeline have to be provided. The original read FASTQ files can also be provided, which can speed up the pipeline, although this is optional as the reads can also be extracted from the BAM files.

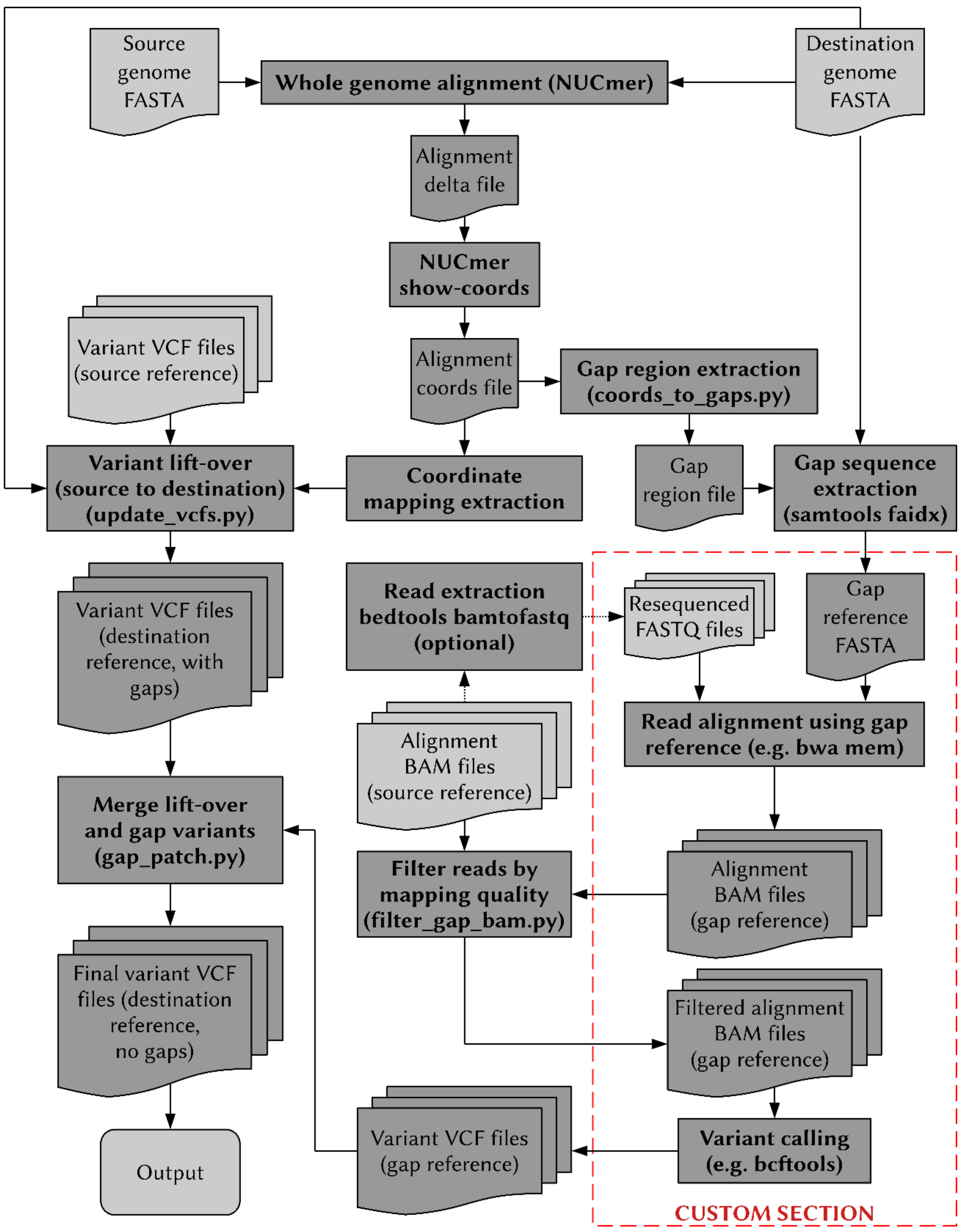

Source genome FASTA
Whole genome alignment (NUCmer)
Destination genome FASTA
Alignment delta file
NUCmer show-coords
Variant VCF files (source reference)
Alignment coords file
Gap region extraction (coords_to_gaps.py)
Variant lift-over (source to destination) (update_vcfs.py)
Coordinate mapping extraction
Gap region file
Gap sequence extraction (samtools faidx)
Variant VCF files (destination reference, with gaps)
Read extraction bedtools bamtofastq (optional)
Resequenced FASTQ files
Gap reference FASTA
Alignment BAM files (source reference)
Read alignment using gap reference (e.g. bwa mem)
Merge lift-over and gap variants (gap_patch.py)
Filter reads by mapping quality (filter_gap_bam.py)
Alignment BAM files (gap reference)
Final variant VCF files (destination reference, no gaps)
Filtered alignment BAM files (gap reference)
Variant VCF files (gap reference)
Variant calling (e.g. bcftools)
Output
CUSTOM SECTION

Figure 2-9: Diagram of the full SeqRemap lift-over pipeline as used with NUCmer whole-genome alignment. *The "custom section" of the pipeline, marked in red, is a (technically optional) mapping and variant calling step that has to be set up externally by the user, to match the pipeline used to generate the source VCFs (as closely as possible), hence why the tools (bwa, bcftools) are only named as examples. It is used to find variants in the "gaps" introduced by novel sequence data in the destination genome.*

2.5.4.2 Whole-genome alignment and gap patching

With the full SeqRemap pipeline (see Error! Reference source not found.), the source and destination genomes are aligned to each other using NUCmer, using a minimum match length of 1000 bases and default settings otherwise. A “coords” file containing a summary of alignment region coordinates and identity percentages is then extracted and used to generate a “coordinate mapping” structure used to translate coordinates between the two sequences, and a “gap region” text file containing Samtools-formatted (i.e., chrom:from-to, one region per line) interval coordinates corresponding to areas of the destination genome that no area of the source genome mapped to. Such regions should correspond primarily to novel sequence data, absent from the source genome, and would result in “gaps” in resequenced genomes in a basic lift-over pipeline. And identity threshold of 99.9% is used to select intervals used in mapping by default, but this can be freely adjusted.

SeqRemap extracts the sequences of gap regions from the destination genome (using samtools faidx), creating a “gap reference”. Reads (raw FASTQ files or reads extracted from the original BAM files used to generate the source VCFs) can then be mapped to this small “reference” sequence. This should be significantly faster than aligning to an entire genome. This alignment has to be executed separately by the user. The tools and settings used should ideally match the original pipeline used to generate the source VCF files; some of the settings are likely recorded in the metadata.

Variant calling on whole-genome reads mapped to a small “gap reference” would likely result in very large numbers of false positives, as it would contain many read alignments which would have preferentially aligned elsewhere had a proper, whole-genome reference sequence been used. The process would also be slow due to the volume of data. To avert these problems, SeqRemap uses BAM files from the original alignment to the source genome to filter the gap reference alignment. Only reads whose mapping quality in the gap reference mapping is higher than for any mapping in the original BAM are kept.

Filtered gap reference BAM files are then used for variant calling. As with the alignment, this is to be executed separately by the user, ideally using tools and settings matching the original pipeline. Finally, SeqRemap merges the "gap reference" variant calling results with the lift-over results, effectively "patching" the gaps with new variants.

It should be noted that this process is not a replacement for *de novo* resequencing using the destination genome as a reference; that remains the preferable, though also significantly slower, method of updating resequenced genome data sets, particularly variant discovery and single-variant resolution is required. However, this approach may be sufficient for lower-resolution approaches like bulk segregant analysis or phylogeny inference, where variant contents are used to derive some signal (e.g., one indicating selection or the presence of a potential introgressions) and the identity of individual variants may be less important.

2.5.4.3 Tomato SL2.40 / SL2.50 to SL4.0 lift-over

SeqRemap has been used internally at Cranfield to update coordinates in resequenced tomato genome data sets, primarily ones from the 150 Tomato Genome ReSequencing Project, from the SL2.50 (and SL2.40) version, which is the latest version of the data hosted by SGN, to the latest (SL4.0) version of the tomato genome.

The size of total intervals mapped between the pairs of chromosome pseudomolecules of the two versions are shown in **Table 2-5** and **Table 2-6**. It can be seen that the most of the change between the two genome versions involved assigning sequences from chromosome 0 to other pseudomolecules. This is expected, as the "chromosome 0" pseudomolecule in tomato assemblies represents sequence fragments which could not be mapped to any specific chromosome. These mappings thus represent gaps being filled and previously unplaced sequences having their location identified.

Table 2-5: Size of intervals (in kbp) mapped between pseudomolecules (destination chromosomes 0 to 6) during lift over from version SL2.40 of the tomato genome to SL4.0 using SeqRemap.

Source SL2.40	Destination SL4.0						
	Chr. 0	***Chr. 1***	***Chr. 2***	***Chr. 3***	***Chr. 4***	***Chr. 5***	***Chr. 6***
Chr. 0	1,119	635	2,840	607	605	399	1,726
Chr. 1	244	69,283	9	0	0	0	3
Chr. 2	161	18	44,210	34	5	10	24
Chr. 3	25	0	0	56,880	0	0	0
Chr. 4	114	0	13	0	54,111	0	0
Chr. 5	0	0	0	0	0	48,054	0
Chr. 6	120	0	0	5	0	0	37,539
Chr. 7	42	0	5	11	0	0	0
Chr. 8	1	5	0	7	0	19	0
Chr. 9	56	0	0	0	0	12	0
Chr. 10	81	171	0	0	0	0	0
Chr. 11	48	0	0	0	0	0	0
Chr. 12	53	0	0	10	0	0	0
Total lifted	2,062	70,111	47,077	57,554	54,721	48,494	39,292
Gaps left	7,581	20,753	6,396	7,744	9,739	16,775	7,967

Note that versions SL2.40 and SL2.50 do not differ in sequence content and are essentially equivalent for the purposes of this pipeline; SL2.40 was used preferentially, as it was the original version that the SGN-hosted genomes were mapped to, and does not contain the large gaps between scaffolds introduced by SL2.50, which inflate the apparent pseudomolecule size in comparison with the other version.

Table 2-6: Size of intervals (in kbp) mapped between pseudomolecules (destination chromosomes 7 to 12) during lift over from version SL2.40 of the tomato genome to SL4.0 using SeqRemap.

Source SL2.40	Destination SL4.0					
	Chr. 7	***Chr. 8***	***Chr. 9***	***Chr. 10***	***Chr. 11***	***Chr. 12***
Chr. 0	2,071	654	631	516	1,226	892
Chr. 1	3	0	0	2	0	0
Chr. 2	10	10	0	32	10	0
Chr. 3	0	0	0	4	0	0
Chr. 4	0	0	0	0	0	0
Chr. 5	0	0	0	5	0	0

Chr. 6	0	0	0	0	0	0
Chr. 7	58,145	0	0	0	0	0
Chr. 8	0	57,285	0	0	0	0
Chr. 9	0	0	58,684	1	0	3
Chr. 10	0	0	0	55,115	0	0
Chr. 11	0	0	0	0	47,151	0
Chr. 12	0	0	0	0	0	57,999
Total lifted	60,228	57,948	59,316	55,674	48,387	58,895
Gaps left	7,655	6,047	9,198	9,119	5,992	7,793

2.6 Discussion

An initial version of this report was shared with SGN alongside corrected VCF files and correction scripts in August 2016, and the flawed variant data sets hosted on the SGN site have been replaced with corrected versions by September 2016, seventeen months after the errors were introduced. They remain the latest publicly hosted version of the data as of January 2022, despite no longer matching the latest version of the reference.

The problems which were encountered highlight the need to validate VCF files for strict adherence to the format specification, including REF allele checks, which requires providing a reference FASTA file, as VCF files do not contain the required information. Only certain validators, such as the ValidateVariants tool from GATK (McKenna et al., 2010), provide this option.

SeqRemap has found usage in research projects conducted at Cranfield University. One such project was “Genomics-assisted selection of *Solanum chilense* introgression lines for enhancing drought resistance in tomatoes” (BBSRC project reference BB/L011611/1), where it was used for lifting over publicly available resequenced genomes originally mapped to SL2.50 for the purposes of comparison with new tomato data mapped directly to SL4.0, with the results to be used in an upcoming publication. Another project where SeqRemap found use was “AdRoot: Genetic control of adventitious rooting in horticultural crops” (BBSRC project reference BB/S007970/1), where it was used to allow for resequenced data sets to be quickly remapped between different versions of a work-in-progress raspberry genome assembly, as linkage maps and assembly versions based on those linkage maps got updated.

There exist other lift-over tools such as CrossMap (Zhao et al., 2014), UCSC liftover (Kuhn et al., 2013), and flo (Pracana et al., 2017). These depend on chain files (Kent et al., 2003), which contain equivalent data and can be generated based on the NUCmer alignment results used by SeqRemap (Marçais et al., 2018). An advantage of SeqRemap is that it looks for exact matches between contigs first, without necessarily requiring a whole-genome alignment between sequence versions. This is, of course, a much simpler scenario than ones which involve *de novo* assemblies rather than merely rearrangements and thus require whole-genome alignment. However, this scenario does occur, as in the tomato reference genome update between SL2.40 and SL2.50 discussed here, and it has posed problems, as evidenced by the resequenced data issues which required correction. Indeed, the apparent simplicity of the changes between SL2.40 and SL2.50, which clearly did not require whole-genome alignment (and thus the use of established lift-over tools), is likely what helped introduce the errors through oversights in the custom Bio-GenomeUpdate tool and, alongside a lack of output validation, also helped them remain undetected for over a year.

SeqRemap generates valid results without requiring a time-consuming whole-genome alignment in this deceptively simple scenario, while also allowing for the use of whole-genome alignment data.

Additionally, the “gap patching” functionality of SeqRemap represents a novel functionality unavailable in other lift-over tools, although it is experimental and would require additional work and testing to validate its usability. If successful, it would address a major limitation of lift-over tools, which is the omission of variants in novel regions (Zheng-Bradley et al., 2017).

The reference genome issues with scaffold mis-ordering and mis-orientation, as well as with discrepancies in gap size compared to published BAC-FISH results (Shearer et al., 2014) were not addressed prior to the tomato genome SL2.50 reference being replaced with version SL3.0 in February 2017. They are now largely irrelevant, at least for new work, as a result of this replacement.

2.7 References

Aflitos, S., Schijlen, E., De Jong, H., De Ridder, D., Smit, S., Finkers, R., Wang, J., Zhang, G., Li, N., Mao, L., Bakker, F., Dirks, R., Breit, T., Gravendeel, B., Huits, H., Struss, D., Swanson-Wagner, R., Van Leeuwen, H., Van Ham, R. C. H. J., … Peters, S. (2014). Exploring genetic variation in the tomato (Solanum section Lycopersicon) clade by whole-genome sequencing. *Plant Journal*, *80*(1), 136–148. https://doi.org/10.1111/tpj.12616

Cingolani, P., Platts, A., Wang, L. L., Coon, M., Nguyen, T., Wang, L., Land, S. J., Lu, X., & Ruden, D. M. (2012). A program for annotating and predicting the effects of single nucleotide polymorphisms, SnpEff: SNPs in the genome of Drosophila melanogaster strain w 1118; iso-2; iso-3. *Fly*, *6*(2), 80–92. https://doi.org/10.4161/fly.19695

Hosmani, P. S., Flores-Gonzalez, M., van de Geest, H., Maumus, F., Bakker, L. V, Schijlen, E., van Haarst, J., Cordewener, J., Sanchez-Perez, G., Peters, S., Fei, Z., Giovannoni, J. J., Mueller, L. A., & Saha, S. (2019). *An improved de novo assembly and annotation of the tomato reference genome using single-molecule sequencing, Hi-C proximity ligation and optical maps*. https://doi.org/10.1101/767764

Kent, W. J., Baertsch, R., Hinrichs, A., Miller, W., & Haussler, D. (2003). Evolution's cauldron: Duplication, deletion, and rearrangement in the mouse and human genomes. *Proceedings of the National Academy of Sciences of the United States of America*, *100*(20), 11484–11489. https://doi.org/10.1073/pnas.1932072100

Kuhn, R. M., Haussler, D., & James Kent, W. (2013). The UCSC genome browser and associated tools. *Briefings in Bioinformatics*, *14*(2), 144–161. https://doi.org/10.1093/bib/bbs038

Kurowski, T. J., & Mohareb, F. (2019). Tersect: a set theoretical utility for exploring sequence variant data. *Bioinformatics*, *36*(3), 934–935. https://doi.org/10.1093/bioinformatics/btz634

Lin, T., Zhu, G., Zhang, J., Xu, X., Yu, Q., Zheng, Z., Zhang, Z., Lun, Y., Li, S., Wang, X., Huang, Z., Li, J., Zhang, C., Wang, T., Zhang, Y., Wang, A., Zhang, Y., Lin, K., Li, C., … Huang, S. (2014). Genomic analyses provide insights into the history of tomato breeding. *Nature Genetics*, *46*(11), 1220–1226. https://doi.org/10.1038/ng.3117\rhttp://www.nature.com/ng/journal/v46/n11/abs/ng.3117.html#supplementary-information

Marçais, G., Delcher, A. L., Phillippy, A. M., Coston, R., Salzberg, S. L., & Zimin, A. (2018). {MUMmer}4: A fast and versatile genome alignment system. *{PLOS} Computational Biology*, *14*(1), e1005944. https://doi.org/10.1371/journal.pcbi.1005944

McKenna, A., Hanna, M., Banks, E., Sivachenko, A., Cibulskis, K., Kernytsky, A., Garimella, K., Altshuler, D., Gabriel, S., Daly, M., & DePristo, M. A. (2010). The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Research*, *20*(9), 1297–1303. https://doi.org/10.1101/gr.107524.110

NCBI. (2019). *AGP Specification v2.1*. Bethesda (MD): National Library of Medicine (US), National Center for Biotechnology Information. https://www.ncbi.nlm.nih.gov/assembly/agp/AGP_Specification/

Pracana, R., Priyam, A., Levantis, I., Nichols, R. A., & Wurm, Y. (2017). The fire ant social chromosome supergene variant Sb shows low diversity but high divergence from SB. *Molecular Ecology*, *26*(11), 2864–2879. https://doi.org/10.1111/mec.14054

Sato, S., Tabata, S., Hirakawa, H., Asamizu, E., Shirasawa, K., Isobe, S., Kaneko, T., Nakamura, Y., Shibata, D., Aoki, K., Egholm, M., Knight, J., Bogden, R., Li, C., Shuang, Y., Xu, X., Pan, S., Cheng, S., Liu, X., … Fabra, U. P. (2012). The tomato genome sequence provides insights into fleshy fruit evolution. *Nature*, *485*(7400), 635–641. https://doi.org/10.1038/nature11119

SGN. (2014). *“Updated tomato genome assembly based on FISH results” pdf document*.

ftp://ftp.sgn.cornell.edu/genomes/Solanum_lycopersicum/assembly/build_2.50/DotPlots.pdf

Shearer, L. A., Anderson, L. K., de Jong, H., Smit, S., Goicoechea, J. L., Roe, B. A., Hua, A., Giovannoni, J. J., & Stack, S. M. (2014a). *Fluorescence in situ hybridization and optical mapping to correct scaffold arrangement in the tomato genome*. *4*(8), 1395–1405. https://doi.org/10.1534/g3.114.011197

Shearer, L. A., Anderson, L. K., de Jong, H., Smit, S., Goicoechea, J. L., Roe, B. A., Hua, A., Giovannoni, J. J., & Stack, S. M. (2014b). Fluorescence in situ hybridization and optical mapping to correct scaffold arrangement in the tomato genome. *G3 (Bethesda, Md.)*, *4*(8), 1395–1405. https://doi.org/10.1534/g3.114.011197

Zhao, H., Sun, Z., Wang, J., Huang, H., Kocher, J. P., & Wang, L. (2014). CrossMap: A versatile tool for coordinate conversion between genome assemblies. *Bioinformatics*, *30*(7), 1006–1007. https://doi.org/10.1093/bioinformatics/btt730

Zheng-Bradley, X., Streeter, I., Fairley, S., Richardson, D., Clarke, L., & Flicek, P. (2017). Alignment of 1000 Genomes Project reads to reference assembly GRCh38. *GigaScience*, *6*(7), 1–8. https://doi.org/10.1093/gigascience/gix038

3 TERSECT: A SET THEORETICAL UTILITY FOR EXPLORING SEQUECE VARIANT DATA

3.1 Abstract

3.1.1 Summary

Comparing genomic features within a large panel of individuals across the same species is nowadays considered a core part of many bioinformatics analyses. Such analyses can usually be expressed as a series of complex set theoretical operations used to compare, intersect, or extract symmetric differences between individuals within a large set of genotypes. Several publicly available tools are capable of performing such tasks; however, due to the sheer size of variant sets being queried, such tasks can be computationally expensive, with runtimes ranging from a few minutes up to several hours, depending on the data set size. This makes existing tools unsuitable for interactive data querying or for use as part of genomic data visualization platforms, such as genome browsers. Tersect is a lightweight, high-performance command-line utility which interprets and applies flexible set theoretical expressions to sets of sequence variant data. It can be used both for interactive data exploration and as part of a larger pipeline thanks to its highly optimized variant data storage and indexing algorithms.

3.1.2 Availability

Tersect was implemented in C and released under the MIT license. Tersect is freely available at https://github.com/tomkurowski/tersect.

3.2 Introduction

Large-scale genome resequencing projects such as the 100 000 genome project (Turnbull et al., 2018), the 1000 Genomes Project (1000 Genomes Project Consortium, 2015) or the 150 Tomato Genome ReSequencing Project (S. Aflitos et al., 2014) provide researchers with large-scale references for genetic variation. These can be compared with novel data to help identify causal variants and QTLs, delimit haplotype blocks and introgressions, or infer phylogenetic relationships (Gao et al., 2019). All of those uses require means of filtering and comparing the variant contents between large phenotypic groups in order to identify concordant and discordant variants. These can be considered applications of set theoretical operations such as intersections or unions on sets of variants.

While multiple tools such as BEDOPS (Neph et al., 2012), BCFtools (Danecek et al., 2021) and BEDTools (Quinlan & Hall, 2010) offer the option to execute such operations, they are relatively inflexible in the complexity of possible queries and rely on parsing input files as they are executed, limiting their speed and responsiveness.

We hereby present Tersect, a tool which allows users to construct queries of any level of complexity by providing its own declarative query language and significantly speeds up their execution using specialized bitmap indices. Queries are interpreted, optimized, and executed in a single step, either on entire genomes or on selected genomic regions, making the process extremely fast and responsive, ideal for an exploratory approach to investigating genome contents.

3.3 Tersect

3.3.1 Interface and command parser

Tersect is a command-line tool and features a command parser which allows a user to enter set theoretical expressions operating on genomes (as sets) and variants (as set elements) and including set theoretical operations such as intersections, unions, and symmetric differences (see Appendix A). These can be arranged into queries of arbitrary complexity, including deeply nested expressions, using a simple syntax. Rather than merely executing the parsed operations in sequence, Tersect builds an abstract syntax tree (AST) which represents the entered expression. The tree can then be optimized to simplify and speed up operations. If the user requests data from multiple genomic regions using the same command, the same AST is re-used for each.

3.3.2 Indexing

Tersect imports VCF file data and uses bitmap indexing to encode binary information on the presence or absence of specific variants in each individual genome while building up a single unified database of alleles across all collected genomes.

The database is sorted and indexed by position and identity. When traversed in order, the stored list of variants is parallel to the per-genome bitmap indices, linking the two data structures. The index on variants and their positions allows for rapid identification of regions of interest by chromosome and position range, while the bitmap indices allow for highly efficient comparisons between genomes, leveraging bitwise operations to compare many sites at once. As any given genome contains only a relatively small subset of possible alleles, the bitmaps are sparse and easily compressed. Tersect uses a variant of the Word-Aligned Hybrid lossless compression method (Wu et al., 2006) which allows logical operations without an explicit decompression step.

The variant data and indices are stored in a special index file which only needs to be generated once per collection of genomes and can be shared and used independently of the source data. Tersect uses a memory-mapped I/O approach to access index file contents, allowing for random access to regions of interest and limiting the memory footprint of queries.

A disadvantage of bitmap indexing, shared by Tersect, is the relative inefficiency of updating and adding data. This indexing approach is generally best suited to read-only applications and is often used in data warehousing. However, the stored data (allele identities and presence in genomes) do not frequently change over time and are generally added in batches (at least one genome at a time), mitigating such disadvantages. The Tersect index builder uses a highly efficient, priority-queue-based merge method, allowing for an index file to be rapidly re-created.

3.3.2.1 Index format

Tersect relies on constructing index / database files to enable it to execute its high-performance queries. The files are in a custom binary format and use the “tsi” filename extension (standing for tersect index) by default.

The first fourteen bytes in the index files encode an ASCII representation of the TSI file format version used. This is a C-string (i.e., 13 characters followed by a null terminator) and “TersectDB 0.2” is currently the only valid value. This is to allow for the correct interpretation of the header that follows, and the rest of the data contained in the index file.

- The TSI header contains the following information:
- Database size in bytes (64-bit unsigned integer)
- Word size used by the database (16-bit unsigned integer)
- Offset of the chromosome list data structure (64-bit unsigned integer)
- Number of chromosomes (32-bit unsigned integer)
- Offset of the genome list data structure (64-bit unsigned integer)
- Number of genomes (32-bit unsigned integer)
- Offset of the free list data structure (64-bit unsigned integer)

While much of the data they contain is compressed, Tersect index files can still be quite large (several gigabytes and more on real data sets). Rather than fully parsing them, Tersect uses them as memory-mapped files. The data structures they contain refer to each other through offsets from the start of the index file. Tersect translates these offsets to pointers based on the mapping location, casting the data structures stored in the file into a representation used elsewhere in the application. The former ‘internal’ data structures are described in the *tersect_db_internal.h* header, while their ‘public’ interfaces used elsewhere can be seen in the *tersect_db.h* header.

Tersect manages memory within the index file using a simple free list and first-fit allocation, expanding the file in page-sized chunks if not enough space is available. Note that the index file is generally intended to be created in a single batch process (in which case there is little fragmentation and no wasted space) with very little later modification, such as renaming samples.

3.3.2.2 Index construction

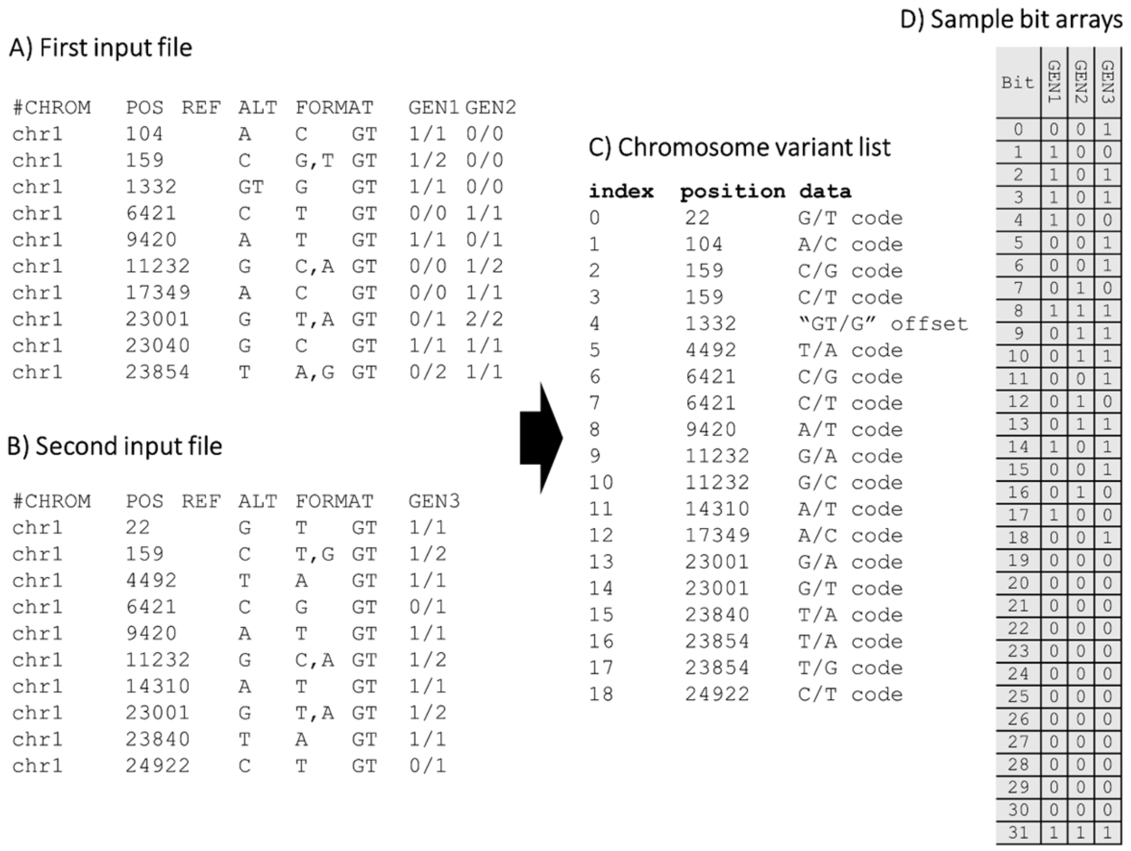
To construct an index file, Tersect uses a custom parser to merge the contents of input VCF files into per-chromosome lists of alleles. This is done using a priority queue algorithm that includes a normalization and local sorting step on each of the input files to ensure the variant alleles are stored in a normalized, unambiguous order (sorted first by position, then alphabetically by the allele base sequences). Single nucleotide variants are encoded using numeric codes for each reference/alternate base combination, while larger variants have their sequences stored as strings allocated in the index file, with the variant list recording the string location offset. An example of the process is shown in **Figure 3-1.**

3.3.2.3 Compression

Per-sample presence or absence of specific variants of a chromosome is encoded in bit arrays using a variant of the Word-Aligned Hybrid (WAH) compression algorithm. The primary data structure is stored as a simple array of 64-bit words corresponding to the entire length of the chromosome. Each of the words is either a “literal” word or a “fill” word; this distinction is indicated by the most significant bit of each word, which is set for literal words and unset for fill words. This leaves 63 bits for other data.

Literal words use their 63 bits to store the presence (set bit) or absence (unset bit) of up to 63 successive variants. Fill words store the length of a run of absent variants in multiples of 63, a type of run-length encoding (RLE). Thus, a fill word containing the (decimal) number 1 indicates a run of 63 absent variants, number 2 indicates a run of 126 absent variants, and so on.

Note that recording runs of empty words rather than empty bits, while wasteful of space, keeps the bits in literal words aligned so that no bit shifting is required to conduct binary operations on variants at the same positions. This trades space for execution speed.


A) First input file

```
#CHROM   POS  REF ALT  FORMAT   GEN1 GEN2
chr1     104      A    C    GT   1/1  0/0
chr1     159      C    G,T  GT   1/2  0/0
chr1     1332     GT   G    GT   1/1  0/0
chr1     6421     C    T    GT   0/0  1/1
chr1     9420     A    T    GT   1/1  0/1
chr1     11232    G    C,A  GT   0/0  1/2
chr1     17349    A    C    GT   0/0  1/1
chr1     23001    G    T,A  GT   0/1  2/2
chr1     23040    G    C    GT   1/1  1/1
chr1     23854    T    A,G  GT   0/2  1/1
```

B) Second input file

```
#CHROM   POS  REF ALT  FORMAT   GEN3
chr1     22       G    T    GT   1/1
chr1     159      C    T,G  GT   1/2
chr1     4492     T    A    GT   1/1
chr1     6421     C    G    GT   0/1
chr1     9420     A    T    GT   1/1
chr1     11232    G    C,A  GT   1/2
chr1     14310    A    T    GT   1/1
chr1     23001    G    T,A  GT   1/2
chr1     23840    T    A    GT   1/1
chr1     24922    C    T    GT   0/1
```

C) Chromosome variant list

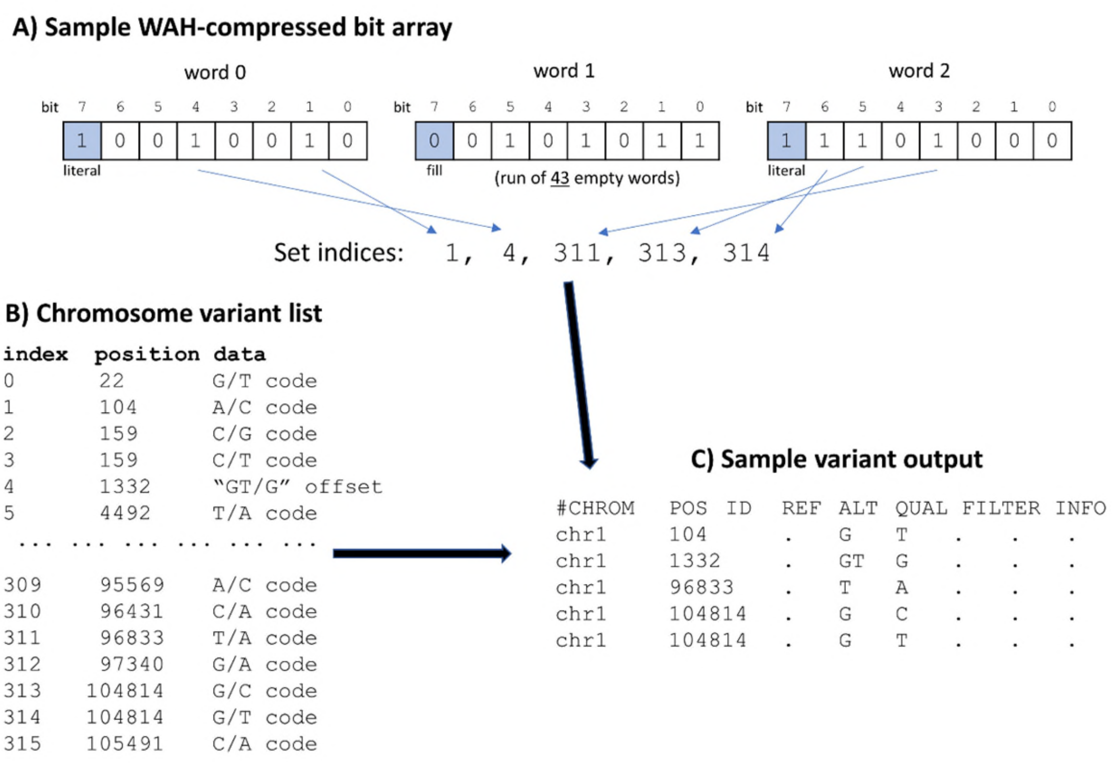
```
index  position  data
0      22        G/T code
1      104       A/C code
2      159       C/G code
3      159       C/T code
4      1332      "GT/G" offset
5      4492      T/A code
6      6421      C/G code
7      6421      C/T code
8      9420      A/T code
9      11232     G/A code
10     11232     G/C code
11     14310     A/T code
12     17349     A/C code
13     23001     G/A code
14     23001     G/T code
15     23840     T/A code
16     23854     T/A code
17     23854     T/G code
18     24922     C/T code
```

D) Sample bit arrays

Bit	GEN1	GEN2	GEN3
0	0	0	1
1	1	0	0
2	1	0	1
3	1	0	1
4	1	0	0
5	0	0	1
6	0	0	1
7	0	1	0
8	1	1	1
9	0	1	1
10	0	1	1
11	0	0	1
12	0	1	0
13	0	1	1
14	1	0	1
15	0	0	1
16	0	1	0
17	1	0	0
18	0	0	1
19	0	0	0
20	0	0	0
21	0	0	0
22	0	0	0
23	0	0	0
24	0	0	0
25	0	0	0
26	0	0	0
27	0	0	0
28	0	0	0
29	0	0	0
30	0	0	0
31	1	1	1

Figure 3-1: Tersect index file construction diagram. *Parts A) and B) show the contents of example VCF input files (metadata and certain columns were omitted). The first input file contains data for two samples (GEN1 and GEN2) and the second file contains data for a single sample (GEN3). All alleles contained in a chromosome are stored in a single list as seen in part C). Membership of individual alleles in each of the samples is encoded in bit arrays as seen in part D), which shows a 32-bit word for the sake of simplicity (Tersect uses 64-bit words by default). The most significant bit is set for all three bit arrays, indicating that the specific word shown is a literal word (as opposed to a fill word – these terms are explained in section below). Note that the indices in the chromosome variant table and the sample bit arrays match – the lists are parallel.*

In classical WAH compression, fill words can be used to indicate runs of either set or unset bits (and potentially other patterns), with the type of fill word being indicated by successive most significant bits. In the Tersect implementation this was simplified and limited to only runs of unset bits for several reasons. The arrays used for indicating variant contents are very sparse and runs of more than 63 set bits are rare, making the improvement in compression had they been included minor. At the same time, limiting fill word metadata to a single bit flag set to 0 means no further flag checks or manipulations are necessary and the value stored in the word can be used directly as an integer representing the run length. This simplifies the code and yields an improvement in execution speed. A diagrammatic representation of the compression can be seen in **Figure 3-2**.



```
index  position data
0      22       G/T code
1      104      A/C code
2      159      C/G code
3      159      C/T code
4      1332     "GT/G" offset
5      4492     T/A code
 ... ... ... ... ... ...
309    95569    A/C code
310    96431    C/A code
311    96833    T/A code
312    97340    G/A code
313   104814    G/C code
314   104814    G/T code
315   105491    C/A code
```

```
#CHROM  POS    ID  REF ALT QUAL FILTER INFO
chr1    104    .   G   T   .    .      .
chr1    1332   .   GT  G   .    .      .
chr1    96833  .   T   A   .    .      .
chr1    104814 .   G   C   .    .      .
chr1    104814 .   G   T   .    .      .
```

Figure 3-2: Diagrammatic example of WAH compression and variant retrieval by Tersect. *Three words (part A) encode allele contents for 315 successive alleles stored in the chromosome variant list shown in part B. The literal words encode the indices of variants present in a sample, while the fill word records the length of a run of empty words (each corresponding to seven absent alleles). The stored binary value is 0b0101011 (decimal 43). With seven alleles per word, this can be used to advance the index indicator of the variant list by 7 x 43 = 301 positions when the bit array is traversed. Note that, while for the sake of simplicity the example uses 8-bit words, Tersect uses 64-bit words by default.*

While Tersect uses 64-bit words by default, this is a value which can be changed at compile-time. Using a word size matching the processor word size (64-bit in most architectures common today) is generally the best choice from a performance standpoint, as it makes it possible to make better use of SIMD (single instruction, multiple data) extended instruction sets to speed up operations on bit arrays. However, smaller word sizes may yield superior compression due to higher data granularity: with 64-bit words one can only save a word of memory when at least 126 successive word-aligned variants are absent (and another word for each further 63 such variants), while with 32-bit words a word is saved starting with the 62nd absent variant (and another is saved for each further 31 such variants). Still, the proportion of metadata (the literal/fill word flag) also rises as the word size grows smaller: for 8-bit words, where metadata takes up 12.5% of the storage, the memory use actually increases.

Another consequence of changing the word size is that index files generated by Tersect compiled with a certain word size are not compatible with Tersect compiled with a different word size. This is why the default word size is set to 64-bits instead of varying based on architecture. Advanced users are free to fine-tune this at compile-time, but they will not be able to use the example data sets provided with tutorials.

3.4 Benchmarking

Tersect was benchmarked against three tools which offer similar functionalities: BCFtools (Danecek et al., 2021), BEDTools (Quinlan & Hall, 2010), and BEDOPS (Neph et al., 2012). It should be noted that, as they are designed to compare variant sets not only to each other but also to other types of data, the last two tools focus on positional overlap and intersection between features rather than variant identity. This means that overlapping but distinct variants, such as different alleles at multi-allelic sites or InDels which span across SNV sites, are considered to be intersecting. This can lead to subtly different results delivered by the tested tools; however, the benchmarks executed for the purposes of this work excluded InDel and multiallelic sites, as they were focused on performance comparisons. The test scenarios were set up to produce identical results in terms of variant content between each of the tools.

The data used for comparison were publicly available tomato genomes from two studies (S. Aflitos et al., 2014; Lin et al., 2014), for a total of 444 resequenced genomes of tomato cultivars and closely related species.

Two important shared functionalities were tested: the identification of private variants, that is, variants occurring only in a single specific genome out of a collection of genomes, and the intersection of a group of genomes to identify variants shared by each of them (also known as concordant variants).

For the former test, subsets of the 444 genomes collection were used. For the latter, subsets of 56 *S. pimpinellifolium* genomes which contain large regions of shared variation distinct from the *S. lycopersicum* reference were used.

Input data for each of the tools were converted into the most appropriate format (e.g., BED for BEDTools) and indexed (where appropriate) prior to the benchmarking. This also applies to Tersect, as the time taken to build an index, which needs to be done only once, was not included in the test runtimes.

3.5 Results and discussion

For the private variant identification benchmark (**Figure 3-3**), all four applications show a linear relationship between the input size (number of genomes) and execution time; for Tersect this relationship is partially obscured by the relatively slow disk read/write operations which comprise a significant proportion of the runtime, especially for small input sizes. This is also the reason why the advantage held by Tersect is the smallest for small numbers of genomes (three times faster than BEDOPS when identifying private variants in sets of 4 genomes), and grows for larger inputs (167 times faster than BCFtools when the full set of 444 genomes are used).

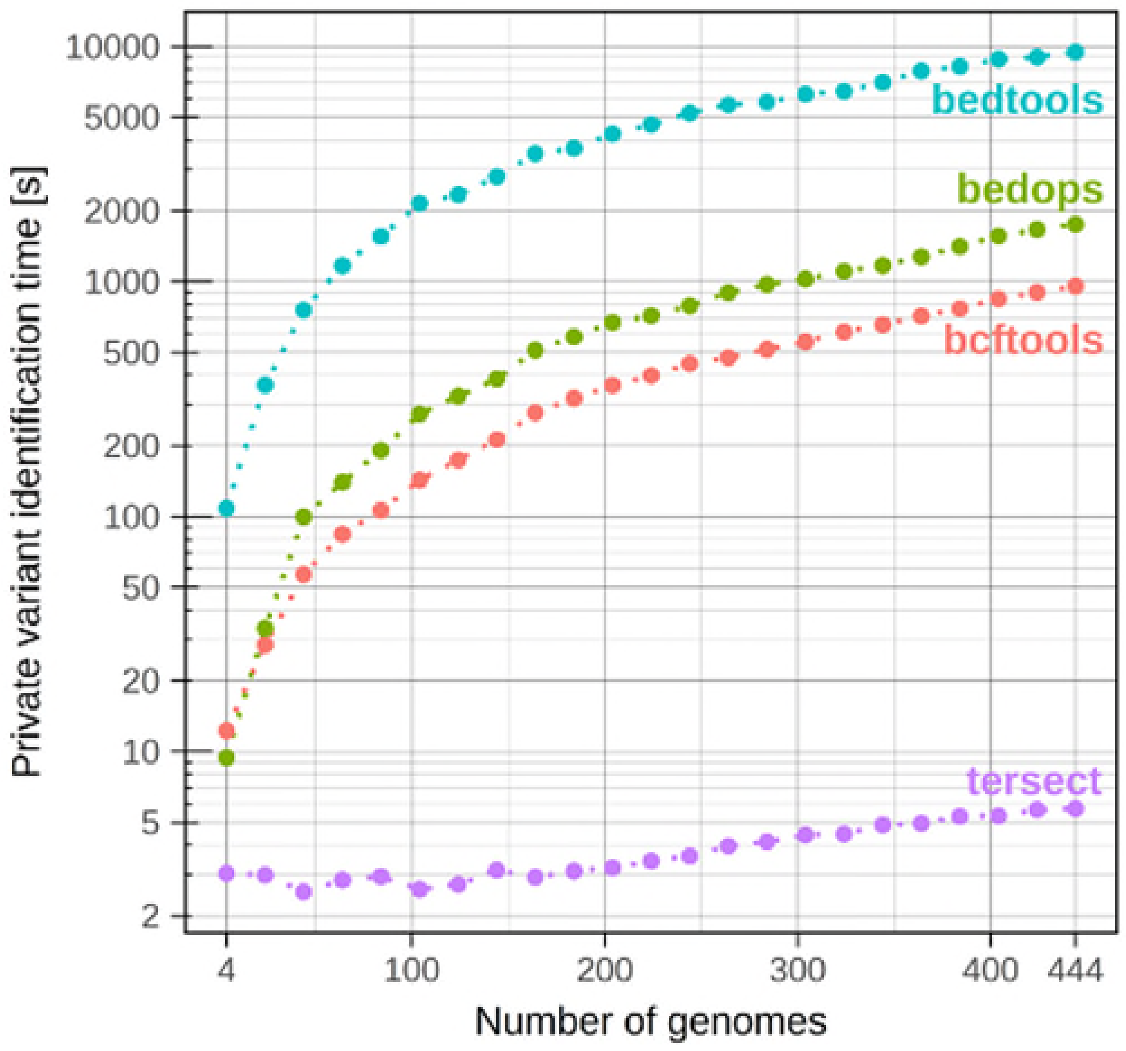


Figure 3-3: Benchmarking results for the identification of variants private to a single genome out of subsets of 444 tomato genomes. *See* **Table 3-1** *for the numeric results.*

For the intersection benchmark, seen in **Figure 3-4,** the results follow a very similar pattern, and all four applications again show a linear relationship between the input size (number of genomes) and execution time, though for Tersect the result is more distorted, and its execution time actually peaks at the smallest input size (two genomes). This is because, as is typical for intersection, the output variant set becomes smaller the more genomes are included. For only two genomes printing the result takes much longer than computing it. However, even for that worst-case scenario, Tersect is approximately three times faster than the next fastest tool (BCFtools). For the largest input size, consisting of 56 genomes, Tersect is approximately 120 times faster than BCFtools.

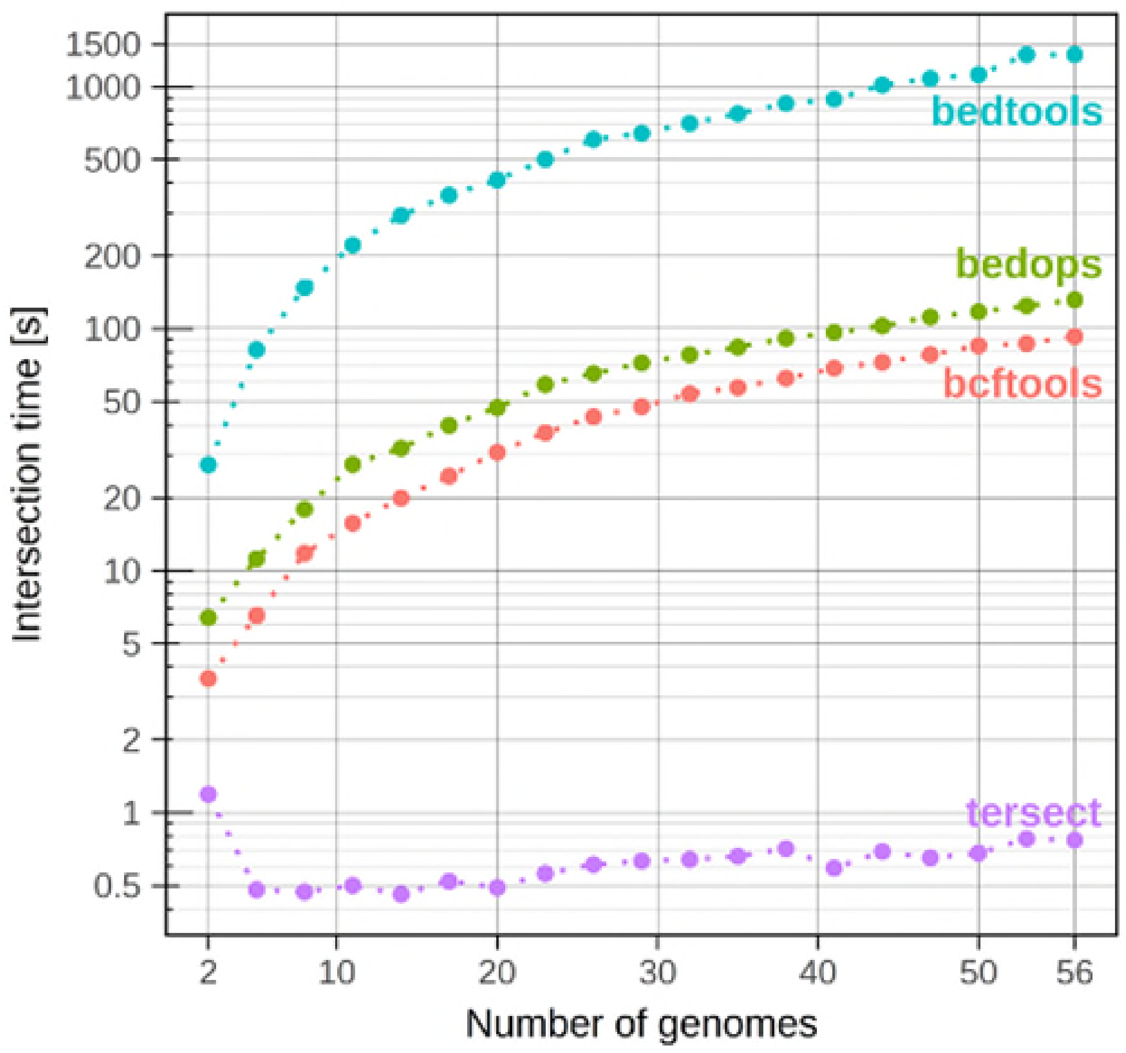


Figure 3-4: Benchmarking results for the intersections of subsets of 56 *Solanum pimpinellifolium* genomes. *This is a wild species of tomato closely related to the* S. lycopersicum, *the cultivated tomato and the most numerous wild tomato species in the source data sets. The shared variants identified through intersection represent alleles typical of S. pimpinellifolium as compared to the cultivated tomato reference genome. See* **Table 3-2** *for the numeric results.*

Index file generation for the largest genome collection (444 genomes) took 10 minutes (see **Figure 3-5**), which is fast enough to make Tersect the fastest tool even if indexing time were to be included in the benchmark, at least for the larger input sizes.

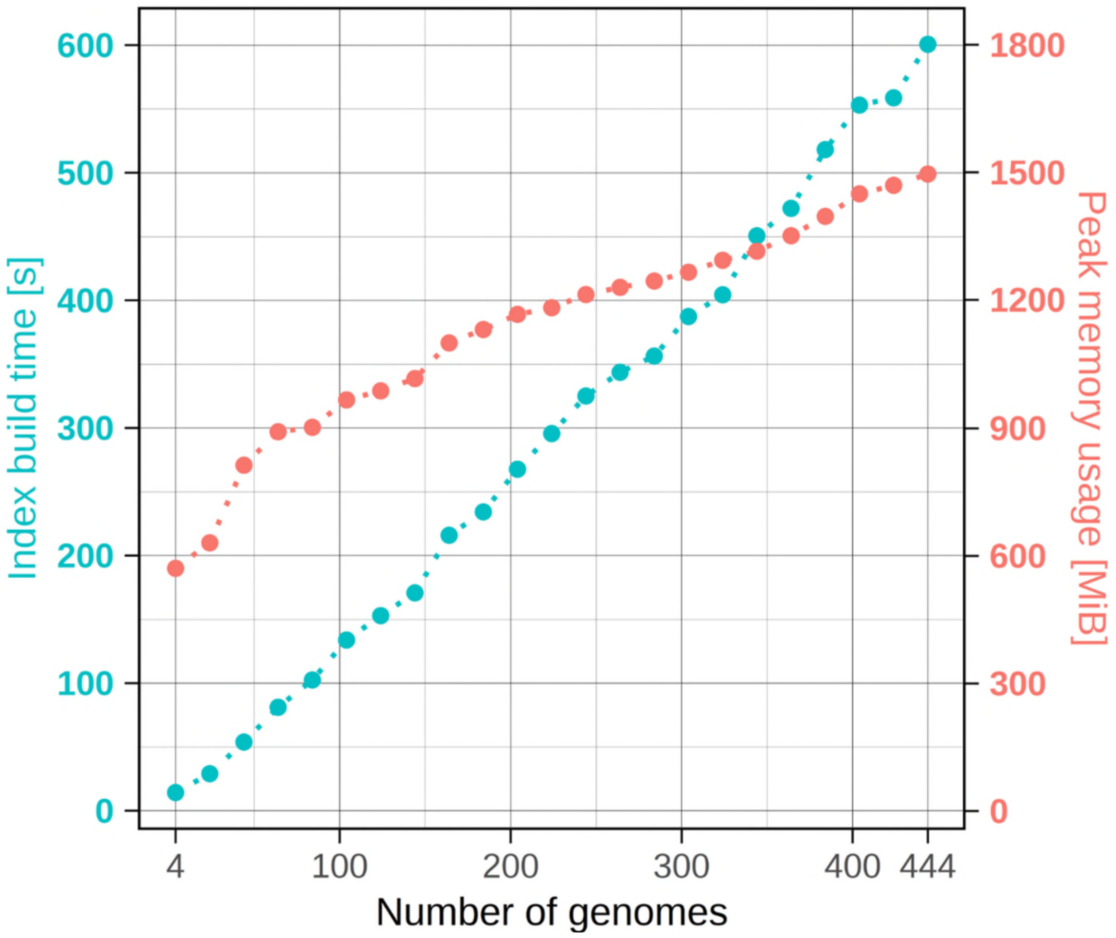


Figure 3-5: Tersect index build time and peak memory usage. *It should be noted that inclusion time per genome varied significantly due to different variant content per genome, evident in the shape of the line. This is also evident in* **Figure 3-6***, which traces a very similar path for the input data. The source genome list was shuffled to minimize this variation. The peak memory usage is defined as the maximum resident set size. See* **Table 3-3** *for the numeric results.*

As seen in **Figure 3-6**, while Tersect index files follow a linear relationship with the size of input data, they are considerably smaller than even compressed VCF files. This means that they can potentially serve as a more efficient storage medium for variant content data, although it should be noted that Tersect indices discard a lot of data normally stored in VCF files, and the input data cannot be recreated in full based on a Tersect index.

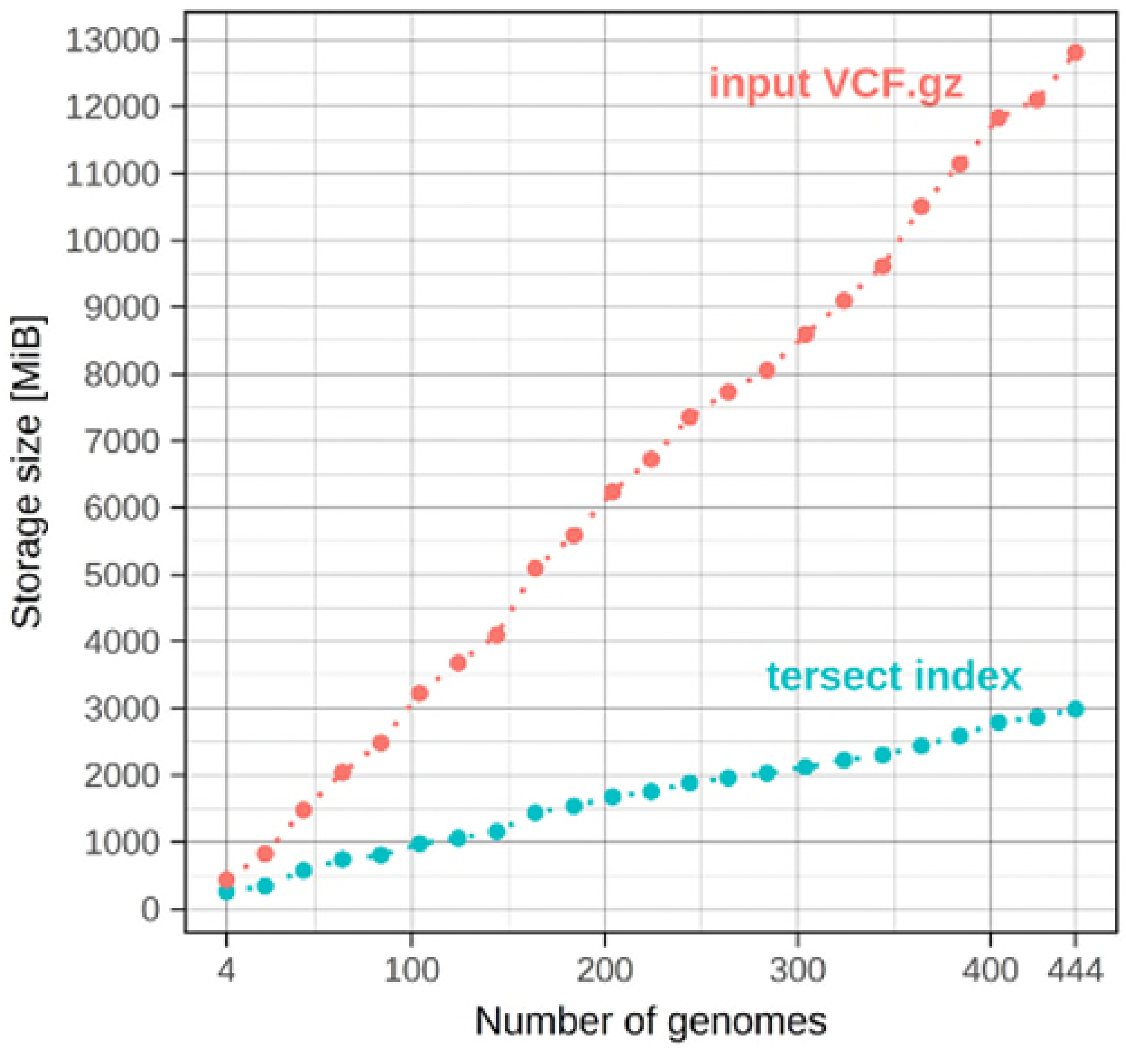


Figure 3-6: Size of input data and generated Tersect index files. *Note that the sizes of individual per genome data sets vary with the number of variants they contain. See* **Table 3-3** *for the numeric results.*

Table 3-1: Private variant identification benchmark results.

	Private variant identification time [seconds]			
Number of genomes	**Tersect**	**BCFtools**	**BEDTools**	**BEDOPS**
4	3.03	12.26	108.37	9.42
24	2.98	28.45	362.75	33.49
44	2.53	56.69	756.15	99.56
64	2.84	84.13	1167.93	139.80

84	2.94	106.18	1555.52	191.23
104	2.59	142.92	2152.15	273.26
124	2.72	173.96	2339.30	325.00
144	3.13	212.22	2786.44	384.52
164	2.92	276.66	3506.33	510.79
184	3.10	318.23	3694.72	581.05
204	3.20	361.42	4246.82	669.34
224	3.42	397.74	4652.93	715.00
244	3.59	447.70	5199.98	788.68
264	3.96	474.70	5654.76	896.68
284	4.13	515.08	5804.88	971.73
304	4.42	553.76	6273.72	1024.31
324	4.46	609.85	6460.30	1105.54
344	4.88	654.08	7038.13	1169.67
364	4.95	712.29	7871.98	1273.92
384	5.30	766.78	8232.68	1409.26
404	5.33	844.31	8833.28	1562.77
424	5.66	900.02	9001.77	1665.46
444	5.72	956.39	9463.02	1753.81

Table 3-2: Intersection benchmark results.

	Intersection time [seconds]			
Number of genomes	**Tersect**	**BCFtools**	**BEDTools**	**BEDOPS**
2	1.19	3.58	27.37	6.39
5	0.48	6.51	81.94	11.19
8	0.47	11.80	147.65	17.97
11	0.50	15.69	221.64	27.45
14	0.46	19.97	293.82	32.06
17	0.52	24.59	356.10	39.86
20	0.49	30.79	411.51	47.15
23	0.56	37.14	501.51	58.73
26	0.61	43.26	605.98	65.34
29	0.63	47.62	642.08	72.23
32	0.64	53.74	705.53	78.06
35	0.66	57.06	773.71	83.86
38	0.71	62.44	853.04	91.27
41	0.59	68.66	888.30	96.49
44	0.69	72.71	1016.69	102.64
47	0.65	78.17	1081.67	111.71
50	0.68	85.04	1122.27	117.50
53	0.78	86.84	1355.56	124.44
56	0.77	92.85	1358.65	131.78

Table 3-3: Tersect index build metrics.

Number of genomes	Input VCF.gz size [MiB]	Output Tersect index size [MiB]	Tersect index build time [s]
4	434	257	14.18
24	826	342	28.98
44	1479	574	53.76
64	2040	741	81.05
84	2483	805	102.43
104	3225	977	133.75
124	3677	1057	152.87
144	4091	1156	170.70
164	5093	1438	215.84
184	5585	1539	234.18
204	6238	1678	267.63
224	6722	1756	295.59
244	7356	1884	325.00
264	7730	1959	343.60
284	8054	2029	356.33
304	8590	2120	387.30
324	9097	2224	404.35
344	9610	2301	450.60
364	10505	2440	472.07
384	11140	2588	518.17
404	11831	2792	553.02
424	12099	2864	558.73
444	12809	2985	600.68

As seen through the benchmarking, Tersect generally performs from three to over a hundred times faster than BCFtools, which is generally the fastest of the other three applications.

The difference in performance is more pronounced for larger inputs and this trend is likely to continue for data sets larger than those examined in this article. This presents a promising outlook for the scalability and future usability of Tersect as more genomes are resequenced every year and the volume of available data continues to rapidly increase. The runtime of all four tools follows a roughly linear relationship with the size of the input. The superior speed of Tersect stems from the highly problem-specific optimization and indexing scheme rather than from improved algorithmic time complexity in the strict sense.

Tersect is the only one among the evaluated tools capable of executing complex queries on large real-world data sets in a matter of seconds, making this the tool of choice to be used interactively, rather than as part of a batch processing pipeline. In combination with the flexible query syntax, this high performance offers new possibilities for real-time, exploratory use of the ever-growing volume of genomic data being produced today.

3.6 References

Aflitos, S., Schijlen, E., De Jong, H., De Ridder, D., Smit, S., Finkers, R., Wang, J., Zhang, G., Li, N., Mao, L., Bakker, F., Dirks, R., Breit, T., Gravendeel, B., Huits, H., Struss, D., Swanson-Wagner, R., Van Leeuwen, H., Van Ham, R. C. H. J., … Peters, S. (2014). Exploring genetic variation in the tomato (Solanum section Lycopersicon) clade by whole-genome sequencing. *Plant Journal*, *80*(1), 136–148. https://doi.org/10.1111/tpj.12616

Consortium, G. P., Auton, A., Abecasis, G. R., Altshuler, D. M., Durbin, R. M., Abecasis, G. R., Bentley, D. R., Chakravarti, A., Clark, A. G., Donnelly, P., Eichler, E. E., Flicek, P., Gabriel, S. B., Gibbs, R. A., Green, E. D., Hurles, M. E., Knoppers, B. M., Korbel, J. O., Lander, E. S., … National Eye Institute, N. I. H. (2015). A global reference for human genetic variation. *Nature*, *526*(7571), 68–74. https://doi.org/10.1038/nature15393

Danecek, P., Bonfield, J. K., Liddle, J., Marshall, J., Ohan, V., Pollard, M. O., Whitwham, A., Keane, T., McCarthy, S. A., Davies, R. M., & Li, H. (2021). Twelve years of SAMtools and BCFtools. *GigaScience*, *10*(2). https://doi.org/10.1093/gigascience/giab008

Gao, L., Gonda, I., Sun, H., Ma, Q., Bao, K., Tieman, D. M., Burzynski-Chang, E. A., Fish, T. L., Stromberg, K. A., Sacks, G. L., Thannhauser, T. W., Foolad, M. R., Diez, M. J., Blanca, J., Canizares, J., Xu, Y., van der Knaap, E., Huang, S., Klee, H. J., … Fei, Z. (2019). The tomato pan-genome uncovers new genes and a rare allele regulating fruit flavor. *Nature Genetics*, *51*(6), 1044–1051. https://doi.org/10.1038/s41588-019-0410-2

Lin, T., Zhu, G., Zhang, J., Xu, X., Yu, Q., Zheng, Z., Zhang, Z., Lun, Y., Li, S., Wang, X., Huang, Z., Li, J., Zhang, C., Wang, T., Zhang, Y., Wang, A., Zhang, Y., Lin, K., Li, C., … Huang, S. (2014). Genomic analyses provide insights into the history of tomato breeding. *Nature Genetics*, *46*(11), 1220–1226. https://doi.org/10.1038/ng.3117\rhttp://www.nature.com/ng/journal/v46/n11/abs/ng.3117.html#supplementary-information

Neph, S., Kuehn, M. S., Reynolds, A. P., Haugen, E., Thurman, R. E., Johnson, A. K., Rynes, E., Maurano, M. T., Vierstra, J., Thomas, S., Sandstrom, R., Humbert, R., & Stamatoyannopoulos, J. A. (2012). BEDOPS: High-performance genomic feature operations. *Bioinformatics*, *28*(14), 1919–1920. https://doi.org/10.1093/bioinformatics/bts277

Quinlan, A. R., & Hall, I. M. (2010). BEDTools: A flexible suite of utilities for comparing genomic features. *Bioinformatics*, *26*(6), 841–842. https://doi.org/10.1093/bioinformatics/btq033

Turnbull, C., Scott, R. H., Thomas, E., Jones, L., Murugaesu, N., Pretty, F. B., Halai, D., Baple, E., Craig, C., Hamblin, A., Henderson, S., Patch, C., O'Neill, A., Devereau, A., Smith, K., Martin, A. R., Sosinsky, A., McDonagh, E. M., Sultana, R., … Caulfield, M. J. (2018). The 100{\hspace{0.167em}}000 Genomes Project: bringing whole genome sequencing to the {NHS}. *BMJ*, k1687. https://doi.org/10.1136/bmj.k1687

Wu, K., Otoo, E. J., & Shoshani, A. (2006). Optimizing bitmap indices with efficient compression. *ACM Transactions on Database Systems*, *31*(1), 1–38. https://doi.org/10.1145/1132863.1132864

4 TERSECT BROWSER

4.1 Abstract

4.1.1 Summary

Both the low cost of genome resequencing and the public availability of large sets of resequenced genomes, especially ones which include wild accessions, make it possible to delimit introgressions and identify their donor species through visualising the genetic distance and phylogenetic relationships based on whole-genome variant data. While software capable of generating such visualisations is available, it is not suitable for fully interactive exploration of the data due to the amount of time it takes to analyse such large data sets and the fact that any changes in the genome set, or the size and segmentation of an investigated interval, requires a recalculation of phylogenetic relationships. Tersect Browser is a Web application optimized for generating such visualisations in an interactive fashion, responsively recalculating, and displaying phylogenetic trees and genetic distance heat maps based on resequenced genome data, all in seconds rather than hours. This work presents the tool itself alongside examples of its usage, the algorithmic approaches (indexing schemes, partial result precomputation and aggregation) which make such performance possible, as well as the latency performance metrics on large-scale human and tomato data sets.

4.1.2 Availability

Tersect Browser was released under the MIT license and is freely available at https://bitbucket.org/tomkurowski/tersect-browser

4.2 Introduction

Introgressive hybridisation is an important factor in crop improvement, making it possible to transfer valuable traits from related species into cultivars (S. Aflitos et al., 2014), but the process of homoeologous recombination can also introduce other, often undesirable, genetic material via linkage drag (Qi et al., 2007). Latent genetic variation originating from wild species may persist in *S. lycopersicum* lines as poorly characterised, but potentially useful or otherwise interesting, cryptic introgressions (Labate & Robertson, 2012). More broadly, the characterisation of introgressions is an important factor in investigating the evolutionary history of many species, including humans (Evans et al., 2006; Nelson et al., 2021).

Large-scale resequencing of genomes enabled by the low cost of NGS makes large sets of variant data (either whole-genome or data with reduced complexity, e.g. GBS) an abundant resource, potentially useful in the characterisation of introgressions, and software tools such as SPrime (Zhou & Browning, 2021) or Introgression Browser (S. A. Aflitos et al., 2015), are now available to take advantage of them. Publicly available resequenced genome data sets, such as those maintained for tomato (S. Aflitos et al., 2014; Lin et al., 2014) which include wild species, can be used to identify donor species even without prior knowledge by comparing newly resequenced data with the larger repository (Silva Ferreira et al., 2018).

A software tool of particular value to such work is the aforementioned Introgression Browser, a Web application focused on the detection of introgressions and identification of the donor parents for introgressed segments through the visualisation of genetic distance and phylogenetic relationships between genomes using resequenced genome data. However, a significant limitation of Introgression Browser is that, despite providing a Web interface, the visualisations it generates are relatively static. While users can change the genome used as the reference, as well as edit row identifiers and heat map colour scales, the set of genomes as well as the interval and segmentation pattern used have to be provided when the database is generated and cannot be modified.

The entire analysis is precomputed, and user interaction is intended to allow the user to customise the visualisation for improved discrimination between samples, not to facilitate further analysis within the Web application itself.

In previous work, where Introgression Browser was successfully used for the identification of an introgression from *S. galapagense* into a domestic tomato line, the set of genomes included in the analysis, as well as the size of segments and borders of the examined interval had to be adjusted to help delimit the exact position of the introgression and improve contrast (Silva Ferreira et al., 2018). As reported, this had to be done iteratively, with a new database being created at each step. While database creation can be automated using scripts, every such step still requires multiple hours of database generation before any result can be shown in a browser. A further complication is that, particularly in collaborative projects, the most suitable person to administer a Web application and set up database generation scripts through a command-line interface (e.g., a bioinformatician) is not necessarily the most suitable person to interpret the results and decide on the next step (e.g., a geneticist), requiring repeatedly coordinating between multiple researchers on top of the computational cost.

Tools which allow for more interactive visualisation are better suited for this sort of iterative work, where the user views the data and decides on the next step (selecting a subset, applying some type of projection or aggregation) based on what can be seen as a result of the previous one. However, for large data sets, such as those in modern genomics, interactive visualisation is difficult to implement, requiring specialized algorithms, data indexing schemes, precomputation of partial results, or even the use of specialized hardware setups such as massively parallel systems (Godfrey et al., 2016). The exact cut-off threshold for “interactivity” is difficult to define, but typically an interactive system is expected to respond within seconds or, ideally, under a second (Shneiderman, 1984).

This work introduces Tersect Browser, a Web application which allows users to interactively generate and explore visualisations of genetic distance and phylogenetic relationships between large numbers of genomes. To make this possible, it takes advantage of the high-performance of its namesake Tersect, a lightweight utility optimized for comparing variant contents between indexed sets of resequenced genomes (Kurowski & Mohareb, 2019). The indexing scheme used by Tersect allows it to very efficiently find the number of discordant SNV sites in any given chromosomal interval, which is sufficient to calculate genetic distance between genomes according to the Jukes-Cantor model (Jukes & Cantor, 1969). Combined with a further, higher-level precomputation and indexing scheme, Tersect Browser can responsively produce *ad hoc* results analogous to those generated by tools like Introgression Browser in a matter of seconds rather than hours.

4.3 Materials and methods

4.3.1 Benchmark and test data

Two main sets of data were used to test and benchmark Tersect Browser. The first was a set of 444 resequenced tomato genomes, formed by combining two publicly available data sets (S. Aflitos et al., 2014; Lin et al., 2014) hosted and maintained by the SGN (Fernandez-Pozo et al., 2015). The tomato reference genome used for the data was SL2.50. While several newer versions of the reference are available (the latest is SL4.0), the publicly hosted data sets have not yet been updated to match them. Still, this version of the reference features large and well-defined sequence gaps, owing to the contribution of FISH and optical mapping data (Shearer et al., 2014), which made it particularly useful for the development and testing of gap-handling solutions in Tersect Browser plots.

The second set of data consisted of 2548 human genomes hosted and maintained by the IGSR (Fairley et al., 2019). The data originate from the 1000 Genomes Project (1000 Genomes Project Consortium, 2015) and have been recently updated to the GRCh38 version of the human reference genome (Zheng-Bradley et al., 2017), which was therefore the reference version used in benchmarking Tersect Browser as well. The actual variant data used were limited to chromosome 1 for ease of processing, but it should be noted that this should not significantly improve the relevant benchmarking metrics, other than in lowering the required storage space and data set preparation time. This is because the phylogenetic trees and heat maps generated by Tersect Browser correspond to chromosomal intervals rather than whole genomes, meaning that no more than one whole chromosome is ever being processed at once. In fact, because only the largest (248.96 Mbp) chromosomal sequence is being used, this setup provides the worst-case scenario with the highest computational demands among human chromosomes – ideal for benchmarking and stress-testing. Still, for the problems in question the most important metric for determining performance is not the sequence size but the number of samples, as phylogeny inference requires pairwise comparisons between taxa, and the number of such comparisons scales quadratically with the sample count. By this metric, the human set of data with its 2548 genomes is far more demanding than the 444-genome tomato data set, and provides a more difficult stress-test of Tersect Browser's performance.

Both sets of data were collected from their respective public repositories as compressed VCF files, which were then used to generate the Tersect index files provided as inputs to the back-end of the application.

Tomato accession data have also been extracted from the C.M. Rick Tomato Genetics Resource Center (TGRC) database hosted by UC Davis and were used to allow for the annotation of tomato genomes. This comprised a snapshot of database records for 5586 accessions and 1028 genes, extracted and stored as described in **Section 4.3.1.3** (TGRCmirror).

4.3.2 Benchmark hardware

The Tersect Browser server used for benchmarking was deployed on a typical desktop PC with a 2-core Intel Pentium G4600 processor and 24 gigabytes of RAM. The storage used for precomputed data was a software RAID 5 array composed of hard disk drives. This represents a low-end deployment and performance setup, available even to individual researchers intending to set up their own private Tersect Browser servers. Dedicated server infrastructure is likely to achieve better performance due to not having to share resources with a user desktop environment. As the application relies heavily on precomputed and temporary file storage, with a read-dominant workload, the use of solid-state storage is also likely to offer improved performance (Agrawal et al., 2008; Youngjae et al., 2011).

4.3 Implementation

While Tersect Browser is distributed as a monolithic repository for the sake of project simplicity, as well the ease of deployment, consistent versioning, and code sharing, it in fact consists of two separate applications, each of which contains its own project structure and can be deployed independently. The two applications represent the front-end and the back-end of the complete Web application respectively, and while the former is very much dependent on to the latter to function as intended, the back-end server application provides a well-defined REST API which could potentially prove useful in other contexts, without the need for the Web front-end. In particular, it provides a remote interface for accessing the functionalities of Tersect, which is ordinarily a command-line tool.

Additionally, a wholly separate server application called TGRCmirror was implemented alongside Tersect Browser. Through a REST interface, it provides tomato accession data based on the TGRC database, which can be used by Tersect Browser to annotate and enrich the plots it generates. As the data are specific to tomato, TGRCmirror is not an integral part of the general-purpose Tersect Browser system, but a working example of how its options can be extended through plugins implementing existing, generic interfaces.

Each of the applications was developed in TypeScript 3.5, a statically typed superset of JavaScript, with some use of Python in the deployment and precomputed data management scripts. A major benefit of TypeScript was that it allowed for the creation of the aforementioned well-defined interfaces, usable for effective communication between subsystems and potential future extensions.

4.3.1.1 Back-end

The back-end part of Tersect Browser is a NodeJS version 8.4 (*NodeJS*, n.d.) application which uses the Express framework version 4.16.3 (*Express*, n.d.) to serve its functionalities via a REST API (Richardson et al., 2013). Persistent data are stored both in the filesystem, which is used for storing Tersect index files and precomputed PHYLIP files (Felsenstein, 1989), and in a MongoDB database version 4.2 (*MongoDB*, n.d.), which is used for storing metadata, saved views and phylogenetic trees. The distinction is due to how the data are used – the stored files are used through command-line tools (Tersect version 0.12, RapidNJ version 2.3.2, Python utilities) launched by the server, while documents stored in the database are accessed via ODM provided by the Mongoose library version 5.3.1 and used directly by the back-end NodeJS application.

4.3.1.1.1 Data set preparation

Tersect Browser data sets can be added by an administrator, requiring a set of resequenced genomes (either a Tersect index file, or VCF files which will be used to create a Tersect index file) alongside configuration settings specifying which reference genome to use and other options. A FASTA file may also be provided to add a new reference genome to the application; this allows for correct display of chromosome sizes and sequence gap locations. Additional sample annotation data may also be provided, for example information on the origins of a sample. Such extra information can then be used to filter samples or colour-code them in the plots Tersect Browser generates.

4.3.1.1.2 Tersect indexing and distance matrix precomputation

The most important and novel functionality of Tersect Browser is the ability to interactively generate heat maps representing the respective genetic distances between large numbers of genomes, alongside trees representing their phylogenetic relationships. This means giving users the option to freely adjust the position and size of the investigated interval, as well as the bin size used in the distance calculations, and to select arbitrary groupings of available accessions (genomes) to be considered, with an accession of choice serving as the "reference" for heat map distances – all "on the fly", as close to real time as possible, to enable exploratory data analysis.

To make this possible, Tersect Browser depends on the high performance of Tersect and a certain advantage of the method in which the tool stores its index data. Because parallel lists of individual variants for each stored genome are encoded in (compressed) bit arrays indicating their presence or absence, the

Hamming distance between two such bit arrays represents a measure of genetic distance between two genomes which can be used to calculate the Jukes-Cantor distance, at least when the analysis is restricted to homozygous SNVs and rare multiallelic sites are excluded or corrected for. The Hamming distance between bit arrays is extremely fast to calculate, because it only requires two processor operations – the population count of the result of a bitwise XOR operation. This enables Tersect to rapidly generate pairwise genetic distance matrices between indexed genomes. While this functionality (as the *dist* command) has been present in Tersect since the initial release of the tool, it was not used as part of its core, documented features. Since then, options have been added to allow Tersect to output those distance matrices in both PHYLIP and JSON formats, directly usable by phylogenetics software and NodeJS applications, respectively.

While – thanks to the advantages described above – the distance matrix calculation by Tersect is fast, it may still not be fast enough to allow for responsive plot generation when larger intervals and higher numbers of genomes are used. The latter metric is especially problematic, because while the amount of required computation grows linearly with the interval size, the number of pairwise comparisons grows quadratically with the number of genomes being considered. This is not a problem for heat map generation, where only one reference genome is compared with each of the others. Said “reference” can either be the actual reference genome originally used for resequencing (in Tersect terms, this true reference can be represented by a completely zeroed bit array) or any one of the resequenced genomes contained in the index. However, the inference of phylogenies requires costly pairwise comparisons between all the genomes.

The amount of required work is reduced by partitioning the chromosomes into arbitrarily small intervals (called “partitions”), and precomputing pairwise distance matrices for those intervals. The matrices are stored in the file system as PHYLIP files, and further precomputed matrices, representing increasingly larger intervals up to whole-chromosome size, are then created and stored by adding up the smaller distance matrices (“sub-partitions”) which cover the same region. This matrix summation can be accomplished very rapidly thanks to the NumPy Python library, and it ensures that Tersect only needs to calculate a pairwise distance matrix once for any chromosomal region during the data set preparation stage. A diagram of this process is shown in **Figure 4-1**.

When a user interacts with Tersect Browser and specifies an interval, the back-end merely has to retrieve distance matrices for a number of precomputed partitions whose combined coverage of the chromosome approximates the interval. These can be added or subtracted (trivial heuristics are used to do this in as few operations as possible, favouring partitions which cover larger parts of the interval) to calculate a combined distance matrix for the interval. Note that – in general – the ends of intervals will not fall exactly on the boundaries of the smallest partitions. This means that Tersect will still have to generate two new distance matrices for small regions at each end of the interval to be included in the calculation. However, their combined size will never be larger than the smallest precomputed partition. A diagram of this process is shown in **Figure 4-2**.

Due to this approach, the size of the smallest partition, arbitrarily specified when a data set is added to Tersect Browser, limits the size of the largest (and computationally most expensive) Tersect operation that has to be executed when generating a pairwise genetic distance matrix for any given interval. This parameter can be tuned to make the complete request as fast as required. Any increase in speed is primarily at the cost of storage space, as smaller partitions will be stored in a larger number of precomputed PHYLIP files (Felsenstein, 1989). Precomputation time also increases, but this cost is proportionally less dramatic for most partition sizes, because the most expensive operation, which is the pairwise distance matrix calculation by Tersect, will only happen once per any chromosomal interval regardless of partition size. A Python utility (Partition Tuner) was developed to help users tune the partition size for a data set to match speed and storage requirements by estimating request execution time. This utility is included in the Tersect Browser repository and provides estimates of execution time by querying the back-end for random intervals of a given size. Such results were the basis for generating **Figure 4-3** and **Figure 4-4**, although at present Partition Tuner only outputs raw numeric results, which require significant manual curation to generate similar figures.

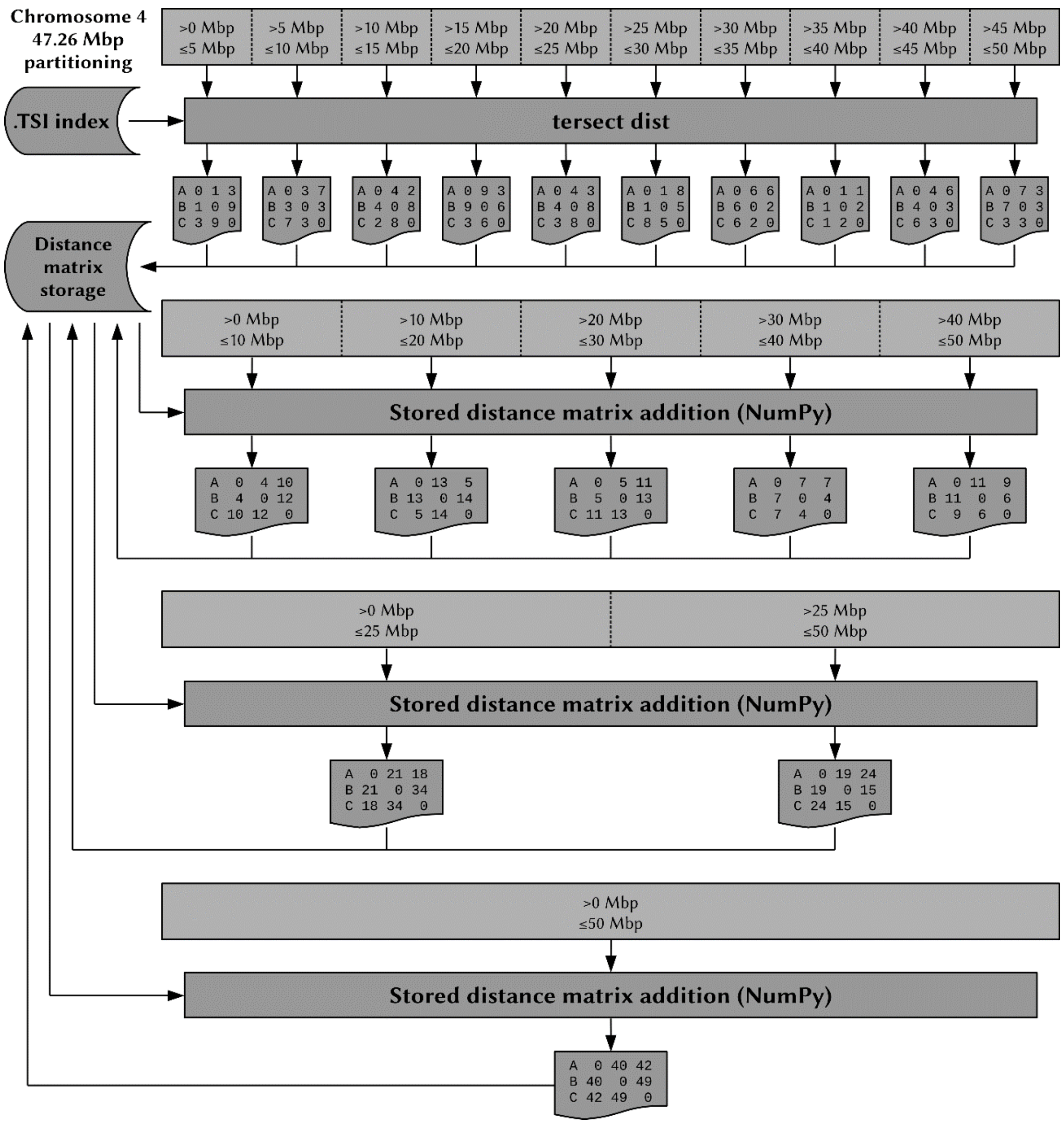
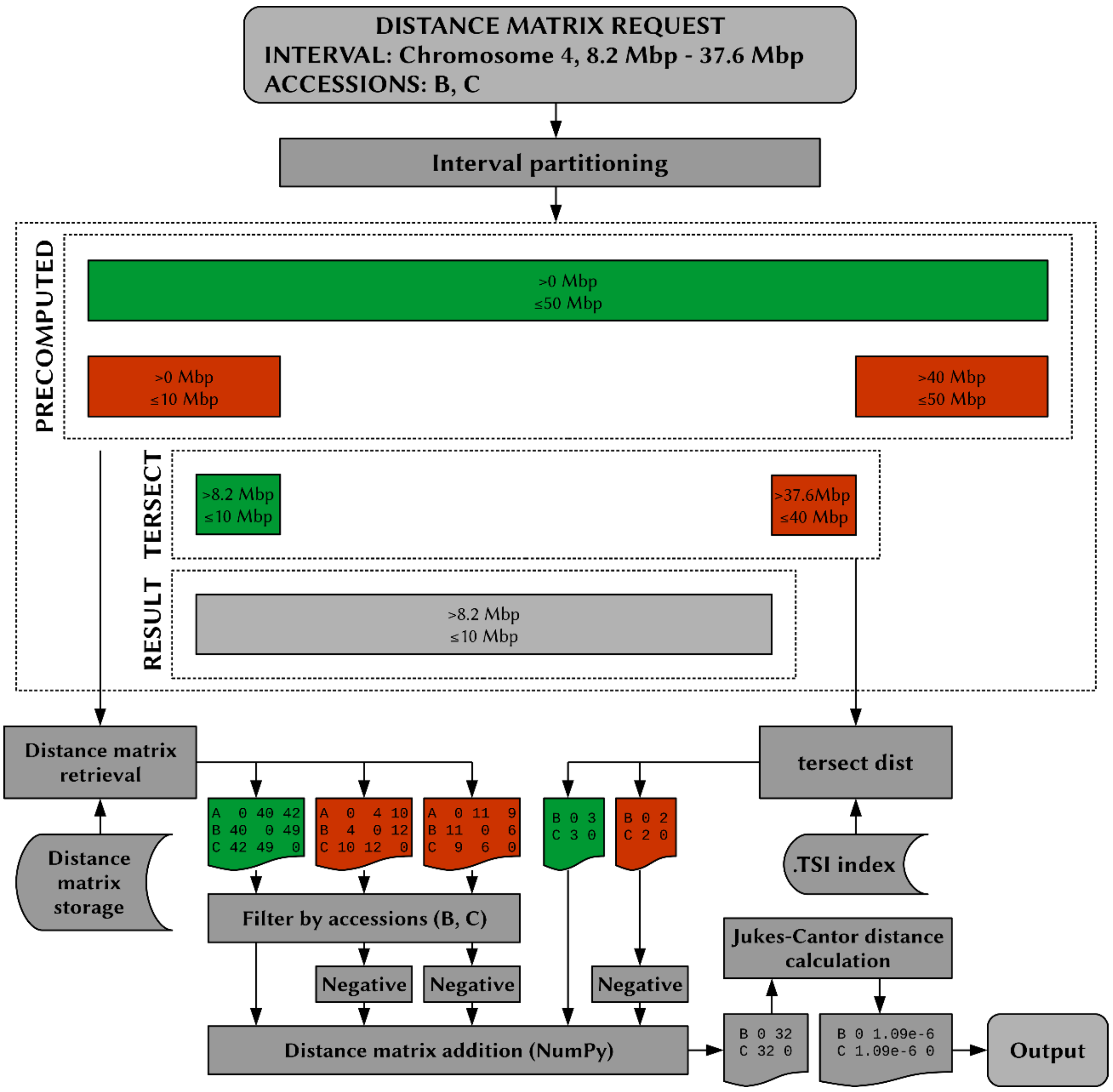


Figure 4-1: Diagram of the distance matrix precomputation process. *The size of the tomato reference genome chromosome 4 is used alongside four partition sizes: 5 Mbp (the smallest and most significant size), 10 Mbp, 25 Mbp, and 50 Mbp. Note that for the largest size, the partition in fact covers the entire, shorter length of the chromosome (47.26 Mbp) instead. Tersect is only used to create the smallest partitions, while the rest are generated by adding up sub-partitions. This requires all the partition sizes (which can be set by the user) to be multiples of the smallest partition size. The "distance" metrics stored in the matrices are actually simple counts of the number of SNV differences rather than true genetic distance metrics; those are calculated downstream.*



```
A  0 40 42
B 40  0 49
C 42 49  0
```

```
A  0  4 10
B  4  0 12
C 10 12  0
```

```
A  0 11  9
B 11  0  6
C  9  6  0
```

```
B 0 3
C 3 0
```

```
B 0 2
C 2 0
```

```
B 0 32
C 32 0
```

```
B 0 1.09e-6
C 1.09e-6 0
```

Figure 4-2: Diagram of distance matrix request handling by the Tersect Browser back-end. *The requested interval is partitioned into a list of smaller intervals, for which the distance matrices can be either retrieved from among the precomputed partitions or generated* de novo *by Tersect, and then added and subtracted to yield the final distance matrix. Only two (at most) Tersect intervals are ever generated for a single request (one at each end of the requested interval) and they are always (at most) half the size of the smallest precomputed partition. In addition, Tersect will only calculate distance matrices for selected accessions, while the stored, precomputed matrices have to be filtered to select the appropriate rows and columns. Note that the distance metrics used throughout the process are simply the substitution (SNV) counts, due to their ease of addition of*

subtraction. The actual genetic distance metrics (Jukes-Cantor distance) are only calculated for the final matrix, which is then used downstream for phylogeny inference.

It should be noted that heat maps cannot be precomputed in the same manner as the matrices used for phylogeny inference, since users can freely vary bin sizes and interval boundary positions, which correspondingly shifts the boundaries of bins for which distance metrics are calculated. Each of the numerous bins would require a separate recalculation analogous to the one conducted for the single interval for which a pairwise distance matrix has to be calculated. Heat map distance data are therefore always generated *de novo* by Tersect. The results are then returned directly as counts of differing SNV sites, without being processed into genetic distance metrics like the Jukes-Cantor distance. This is because they are outputted and sent as textual JSON files, meant for direct use by the front-end application. Using counts (which are integers) results in much smaller file sizes than if floating point distance metrics were to be used. Those are calculated on the front-end, with the additional benefit of off-loading that computational cost from the server to the user's computer.

4.3.1.1.3 Phylogeny inference

Tersect Browser uses RapidNJ version 2.3.2 (Simonsen et al., 2008) to generate canonical neighbour-joining phylogenetic trees based on the pairwise distance matrices for a given genomic interval and set of samples. RapidNJ was chosen from among publicly available implementations of the algorithm due to being optimized for high performance when using large numbers (up to tens of thousands) of taxa (Simonsen et al., 2011). It utilises PHYLIP format files, like the ones used elsewhere in the application for storing pairwise distance matrices, and returns textual outputs in the Newick format. A diagram of the process for generating the input distance matrices used by RapidNJ is shown in **Figure 4-2**. Note that these input PHYLIP files are not persistent – only the Newick-format output is reported and stored.

Generating a phylogenetic tree can be a relatively lengthy process, both in itself and, more importantly, through reliance on the upstream creation of an appropriate pairwise distance matrix. Instead of attempting to generate a new tree every time, Tersect Browser stores a database entry for each valid request, indexed by a data set identifier, chromosome name, interval position, and a list of included accessions (genomes). This is enough information to uniquely identify a tree, which – once generated – is added to the database entry as a Newick-formatted string of text. The status of an ongoing tree generation process is also continually updated in the database, reporting the percentage of completion (or potential errors) until the tree is ready to be retrieved. Any unique tree is therefore only ever generated once and can thereafter be immediately retrieved from the database when requested, even by users other than the original requestor.

4.3.1.2 Front-end

End users of Tersect Browser are intended to use it through a graphical user interface accessible through a Web browser. This front-end application was developed using the Angular framework version 8.2.1 (*Angular*, n.d.), which is used to provide the overall project structure and user interface, which is styled using the PrimeFlex component library version 1.0.0 (*PrimeFlex*, n.d.). The RxJS library version 6.4.0 (*RxJS*, n.d.) is used extensively to coordinate user inputs and asynchronous requests to the back-end made through Angular services. Basic diagrams of the interface can be seen in **Figure B-2** and **Figure B-3**.

4.3.1.2.1 Data requests

An important feature is that the requests needed to draw the heat map and phylogenetic tree are made independently. When both are needed, they can be concurrent, but a new phylogenetic tree is only requested when the interval or the list of selected genomes change, whereas the heat map has to be updated whenever the “reference” genome or bin size changes as well. The phylogenetic tree request is generally much slower than heat map requests, and is therefore the main determinant of how long a view takes to be displayed, except when the back-end is able to fetch a previously generated tree.

4.3.1.2.2 Heat map generation

The user can interact both with the overall heat map (e.g., by dragging it with the mouse, or zooming it in and out) as well as with its individual values (e.g., by hovering over a specific bin to ascertain its exact location). However, the number of bins can be very large – for the 444-tomato genome data set used in benchmarking, and the (default) bin size of 50 kbp, there are over 800,000 bins to keep track of on chromosome 1 of the tomato genome. The bins are therefore not represented as DOM elements (A. van Kesteren, A. Gregor, 2022) or even as rectangles in an image as displayed in the application, but as an array of 8-bit unsigned integer values between 0 and 255 (Uint8ClampedArray). When the heat map is drawn, values representing its currently visible area are extracted from the array and transferred to the `<canvas>` element rendered in the browser through the ImageData interface of the Canvas API (Fulton, 2013; WHATWG, 2022). This means that bins are drawn as individual pixels. The final, rectangular shape is achieved by stretching the canvas through the application of CSS styles for height and width in terms of percentages appropriate to the zoom level. This makes the drawing process very fast, allowing the browser to make full use of performance-enhancing features such as hardware acceleration, and significantly limits memory usage. As a result, Tersect Browser plots can be drawn, moved, and zoomed smoothly even on low-end machines.

As bins do not have their own DOM elements, the specific bins the user interacts with on mouse events (e.g., hovering, clicking, dragging) are determined programmatically, based on the position of the mouse relative to the canvas.

The heat maps are coloured monochromatically based on the distance of each bin to a single genome called the "reference", which can be any of the genomes present in the data set, not just the reference genome to which the resequenced data were aligned. The data are received from the back-end in the form of a JSON object with accession identifiers as keys and arrays containing substitution (SNV) counts between each specific accession and the reference as values. It should be noted that due to the drawing algorithm described above, the resolution of the genetic distance represented in each column of the heat map is limited to 256 distinct values, from 0 (identical to the reference within the bin) to 255 (furthest from the reference within the bin). The SNV counts received by the front-end are first used to calculate the Jukes-Cantor genetic distance for each bin. The distance metrics are then scaled to the 0 - 255 range on a per-bin basis.

A scale tracking chromosomal positions is drawn along the top of the heat map. It is zoomed and scrolled in sync with the heat map, although the scrolling is only horizontal, with the scale always remaining visible at the top of the plot. A user can drag their mouse along the scale to select a smaller interval for a new plot.

Gaps in the reference genome, identified based on a FASTA file uploaded with the data set, are drawn as red rectangles which obscure the relevant gap intervals along the length of a chromosome, provided they are larger than a single bin.

4.3.1.2.3 Phylogenetic tree generation

Phylogenetic trees generated on the back-end are received as Newick-formatted text strings. These are parsed into a graph and drawn on the left side of the interface in one of three ways: as accession labels ordered according to the tree structure (i.e., in the top-down order they would be drawn in had the tree been displayed), as a full, bifurcating tree with branch lengths proportional to genetic distance, or as a simplified bifurcating tree with branches of constant length (preserving the tree structure but not the distances shown in the full tree).

4.3.1.2.4 Accession selection and annotation

A tabular view of all accessions available in a data set can be seen by opening the *Accession selection* overlay in Tersect Browser (see **Figure B-3**). The accessions, listed as rows, can be selected and deselected for use in the visualisation through checkboxes. They can also be sorted and filtered based on columns, which – alongside the accession's label obtained from the Tersect index file – can contain arbitrary data added alongside the data set in the form of CSV files.

A user can also further annotate the accessions at run-time by adding columns through the use of plugins, which implement an "Accession Info Importer" interface and are available through buttons added below the main accession table. These are meant to contact external resources (e.g., REST APIs) and import additional data into the table based on matching some pre-existing identifier (e.g., the TGRC accession number). An example of this is the TGRC Gene Importer plugin, which allows a user to select a tomato gene listed in the

TGRC database. Doing so will add a new column to the accession table, listing the gene's allele for each of the stored tomato accessions (provided they have a TGRC accession number). This allows for easy selection or grouping of accessions containing specific alleles of TGRC genes.

Accessions can also be assigned to groups, which can be either defined in a JSON file added alongside the data set, or created by a user based on an arbitrary selection or column filter. It is possible to assign colours to groups. These colours are then used to mark the labels of group members in Tersect Browser plots.

Note that, while columns or groups introduced through CSV or JSON files when the data set was first added are available to any user who views said data set, any columns or groups added by a user are only visible to people who access views shared by that user (see the following section on view sharing). This avoids crowding the main view of the data set with annotations from different users, while allowing each to create their own, custom view, which they can share with others.

4.3.1.2.5 View sharing

Because Tersect Browser is meant to allow users to explore the available data sets, generate their own visualisations, and facilitate collaboration, the option to save, restore, and share particular states of the Web application (referred to as "views") is an especially important feature. The Tersect Browser interface contains a "Share" button, which generates a persistent link (containing a unique identifier) to the current view, which can be used to revisit or share a visualisation. All of the interface settings at the time of sharing are saved into the back-end database and restored upon visiting the link. As the back-end also stores previously generated phylogenetic trees, the restored view will generally be available immediately, without requiring a new phylogenetic tree to be generated.

It should be noted that the initial view of a data set, visible when a user accesses said data set from the home page of Tersect Browser, is in fact the same sort of stored view as the ones generated by pressing the "Share" button. This "default view" initially shows all the available accessions and uses the entirety of the first chromosome as the viewed interval, but it can be trivially replaced with any other view.

4.3.1.2.6 Plot export

A major feature of Tersect Browser is its ability to not only responsively generate shareable views of data for the sake of exploratory analysis, but also to export publication-quality, high-resolution plots directly from the Web application with just a press of a button. At present the only supported export format is PNG.

The process for exporting plots is similar to the one used to display them in the browser, but its individual elements (phylogenetic tree, heat map, scale) are first drawn in their entirety on separate, offscreen canvas before being combined at the appropriate scale in a final offscreen canvas, from which a data blob is extracted and downloaded as a file of the specified format.

As this process is wholly separate from the actual display of a Web browser, it could also be conducted on the server-side of Tersect Browser, but in the current implementation it runs on the client-side. This ensures consistency between what the user sees rendered on the Web page and the exported images and off-loads a relatively costly operation onto the client, but a potential risk is that user resources (particularly the available memory) may not be sufficient to render larger-resolution plots. This may require further optimization or a transfer of the export functionality to the back-end in future versions of Tersect Browser.

4.3.1.3 TGRCmirror

The regularly updated database maintained by TGRC contains information on individual tomato accessions, including their collection notes (e.g., country, site, habitat) and alternative identifiers other than TGRC's own accession numbers, which allows the information to be cross-referenced and combined with other data sets. It also stores information on genes and their alleles identified in the accessions, alongside notes on the resulting phenotypes. However, these resources are difficult to access programmatically, as no publicly accessible API is offered, with the data intended to be browsed through a Web-based form system.

A Web scraping pipeline was implemented and used to extract and parse TGRC data on 5586 accessions and 1028 genes (i.e., the entire relevant part of the database at the time of writing). The results were stored in a MongoDB database and a simple Express-based REST API was implemented to allow for programmatic queries to retrieve lists of genes and lists of accessions for which the allele of a particular gene has been characterised. The extraction pipeline and REST API are collectively called "TGRCmirror" and are intended to provide a stable, easily accessible snapshot of a subset of the TGRC database, without requiring continuous access to TGRC resources and constant parsing of its HTML-formatted contents.

Additionally, the TGRCmirror pipeline extracts accession information (identifiers, collection notes) into a CSV file, which can be added to Tersect Browser alongside a tomato data set. This allows the information to be seen and used to filter accessions in the Web application's accession selector control, provided matching accession identifiers are available.

As mentioned in section 4.3.1.2.4, the Tersect Browser front-end interface allows for the addition of annotation plugins, which let users add extra columns to the accession selector using external REST resources (Richardson et al., 2013). TGRCmirror is intended to serve as one example of such a resource, with a matching plugin on the front-end of the application used for importing TGRC gene data.

4.4 Results and discussion

The primary objective of Tersect Browser was to provide a Web application which would allow for interactive visualisation of phylogenetic relationships and potential introgressions in resequenced genomic data. The main metric for evaluating the level of success is therefore the time it takes to generate a new visualisation, which should be short enough to allow a user to explore the data in "real time", or as close to that as possible. Phylogeny inference – from generating a pairwise genetic distance matrix for a set of genomes and a specific chromosomal interval an interval to phylogenetic tree generation – is the slowest part of the process and its speed-limiting factor. It was therefore the primary subject of benchmarking, with the time taken to generate a phylogenetic tree being measured for different chromosomal intervals and precomputed partition settings as shown in **Figure 4-3** for the tomato genome data set and **Figure 4-4** for the human genome data set.

It is evident from the benchmarking that the median response time depends primarily on the size of the smallest precomputed partition, rather than on the size of the requested interval. This is because neither the number nor the complexity of required operations increases appreciably with the interval size: a similar number of precomputed matrices (which are themselves based on larger intervals, but this does not affect matrix dimensions) needs to be added up to create the final distance matrix, and only (up to) two Tersect operations, each of a size limited to half the smallest partition, are ever required. The smallest precomputed partition size drives performance because it limits the size of Tersect operations and allows for more granularity in partition sizes (which must be multiples of the smallest partition size).

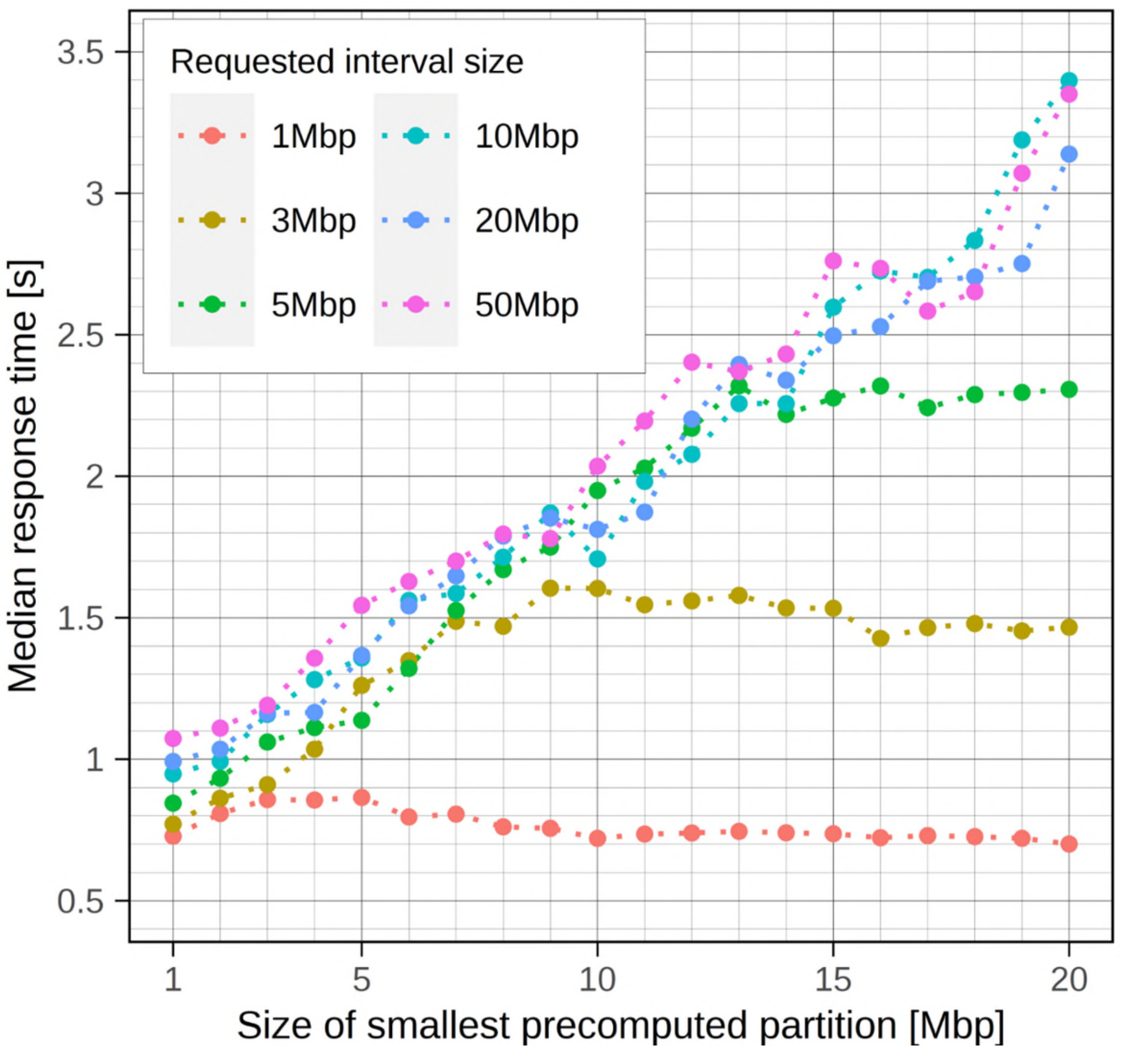


Figure 4-3: Phylogeny inference times for the entire 444-genome tomato data set and chromosomal intervals of different sizes as a function of the smallest precomputed partition size. *One hundred random intervals were generated for each of the tested interval sizes (from 1 Mbp to 50 Mbp). The same sets of intervals were used to test each precomputed partition size, with fifty requests executed for each combination of interval size and smallest partition size. The median response times were then recorded for each partition size. See* **Table B-3** *for the numeric results*

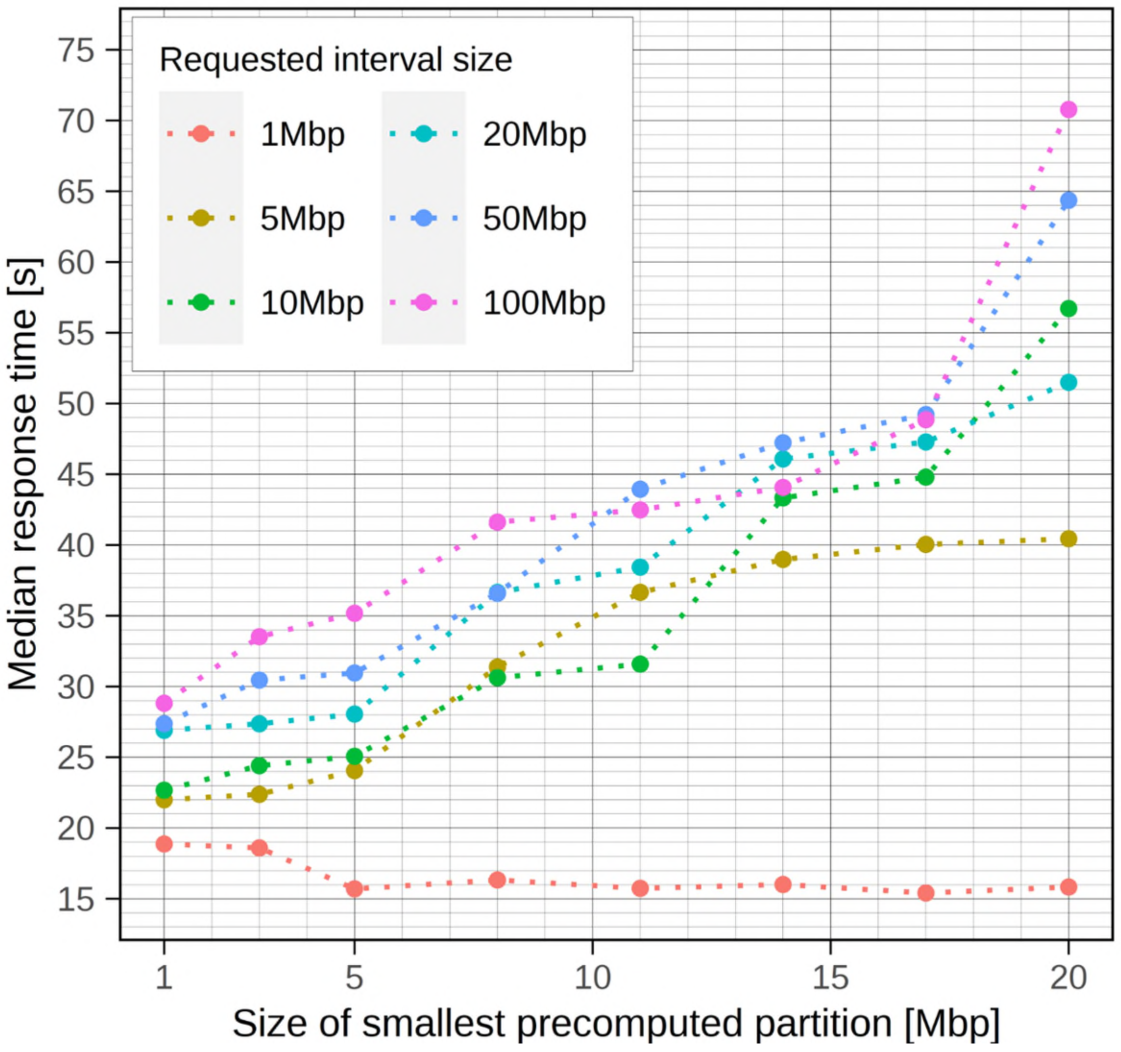


Figure 4-4: Phylogeny inference times for the entire 2548-genome tomato data set and chromosomal intervals of different sizes as a function of the smallest precomputed partition size. *Fifty random intervals (all on chromosome 1) were generated for each of the tested interval sizes (from 1 Mbp to 100 Mbp). The same sets of intervals were used to test each precomputed partition size, with fifty requests executed for each combination of interval size and smallest partition size. The median response times were then recorded for each partition size. See* **Table B-4** *for the numeric results.*

Additionally, it can be seen that for intervals much smaller (more than two times) than the smallest partition size, the response time becomes approximately constant. This is because with such large partitions, the precomputed matrices are no longer useful in reconstructing smaller intervals, and the response time approaches the execution time of a Tersect request on the whole interval instead.

Precomputed partition sizes are parameters set by the user, and can therefore be adjusted on a per-data set basis to provide a sufficiently fast response time at the cost of precomputation time and storage space (see **Figure 4-5** and **Figure B-1**).

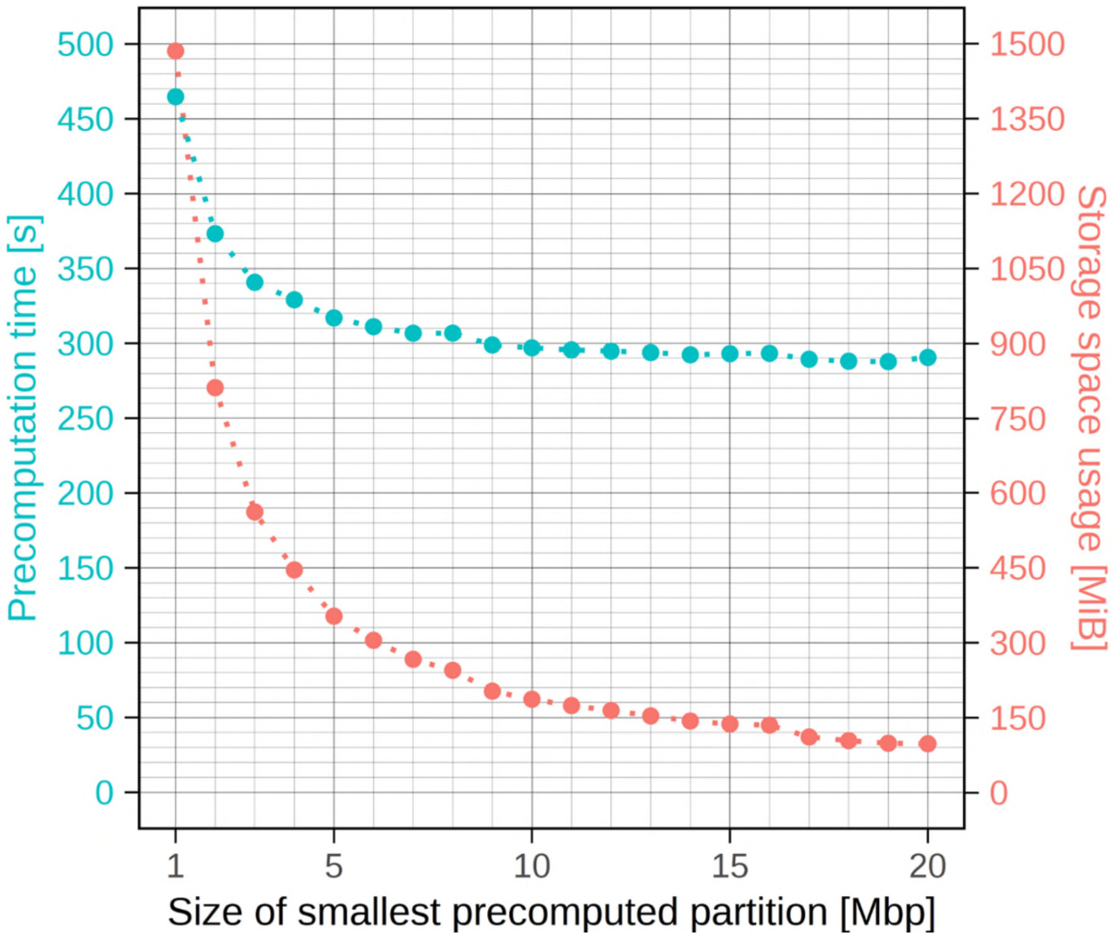


Figure 4-5: Time and storage space costs of distance matrix precomputation as functions of the smallest partition size. *The 444 resequenced tomato genome data were used for benchmarking. To create successive partitions larger than the smallest, their size was doubled until it was larger than the largest chromosome (98.5 Mbp), which is the default approach taken by the data set addition script. The time measurements were recorded on a desktop PC and would scale according to CPU speed, but the storage size should remain invariant for a given*

data set and partition size. Both cost metrics exhibit a component that is inversely proportional to the partition size, but for time this is obscured by the mostly constant cost of running Tersect once per genomic interval, especially for larger partition sizes. These costs have to be weighed against the intended request handling speed (see Figure 4-3*) when selecting the partition sizes for a particular data set. See* Table B-1 *for the numeric results and* Figure B-1 *for equivalent data measured for the human data set*

It can be seen that for the tomato data set, a median response time of around one second is achievable, while for the much larger human data set, median response times under 30 seconds can be reached. Both of these are arguably sufficient for interactive work, although the difference in performance highlights the role of sample size (i.e., the number of genomes) and suggests that for data sets significantly larger than the human one, the algorithms used by Tersect Browser may no longer be sufficient for interactive work.

The relationship between the number of genomes used and execution time is visible within individual data sets as well (see **Figure 4-6**) and is approximately linear in such a context, if not when comparing different data sets. The measurements made for all the genomes represent the worst-case (slowest) scenario, which is why they are suggested as guides to select appropriate precomputed partition sizes; this helps guarantee a certain lower bound for performance.

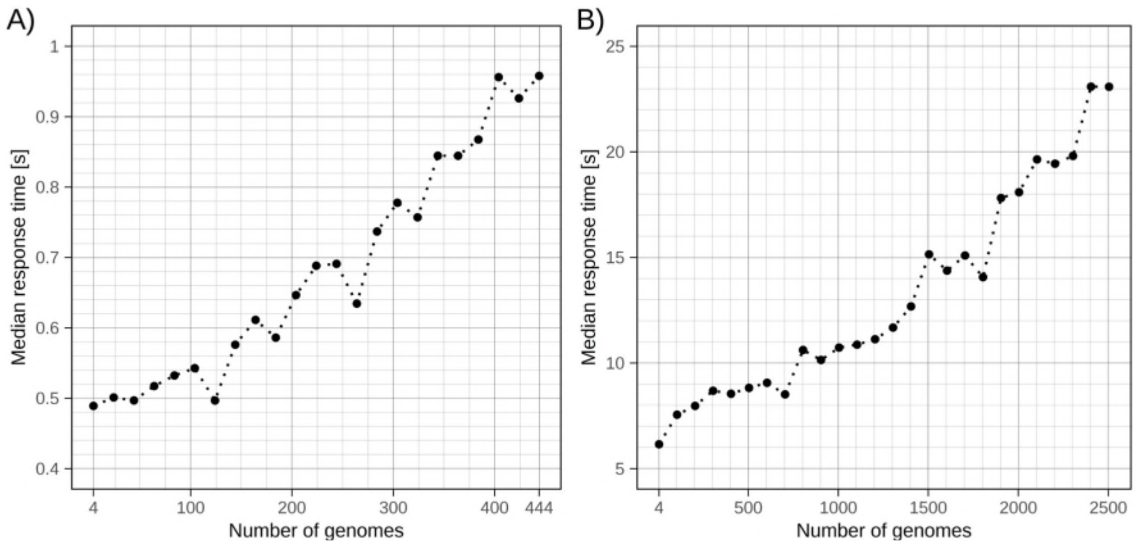


Figure 4-6: Response time as a function of the number of genomes used in phylogeny inference requests for the tomato (A) and human (B) data sets. *The smallest precomputed partition size used was 1 Mbp for both data sets. Twenty requests were made for each tested number of genomes, using random subsets of the total genome set and random chromosomal intervals (with the length of 10 Mbp) for each request.*

Still, it is clear that even for the same number of genomes, requests using the human data set are significantly slower than those using the tomato data set. Indeed, the minimum time taken by requests using the human data set (with 4 genomes) is longer than the time taken by requests using the entire tomato data set (with 444 genomes). This is because, even though only the specified genomes are being used, precomputed distance matrices still must be parsed in their entirety for the relevant rows and columns to be extracted. This constitutes a constant computational cost proportional to the total number of genomes. This cost could potentially be mitigated by the use of a more sophisticated storage or indexing scheme for the precomputed distance matrices.

Tersect Browser can be used to locate potential introgression sites and identify likely donors, as shown in **Figure 4-7**, which shows a possible introgression from *Solanum pimpinellifolium* LYC2798 (or a similar donor) into chromosome 6 in two cultivars, previously identified in a demonstration of the capabilities of Introgression Browser (S. A. Aflitos et al., 2015).

The TGRC gene annotation functionality (i.e., TGRCmirror and its matching front-end plugin) was validated by verifying that relevant known *loci* could be found and highlighted in Tersect Browser visualisations. The tomato *u* (uniform ripening) gene was used for this as seen in **Figure 4-8**. It was chosen because its TGRC allele information was available for a relatively large number of accessions present in our data set (34 in total, including 16 accessions annotated with a -- genotype and 18 with a + genotype), and because it has been mapped to a small region on chromosome 10 (Powell et al., 2012).

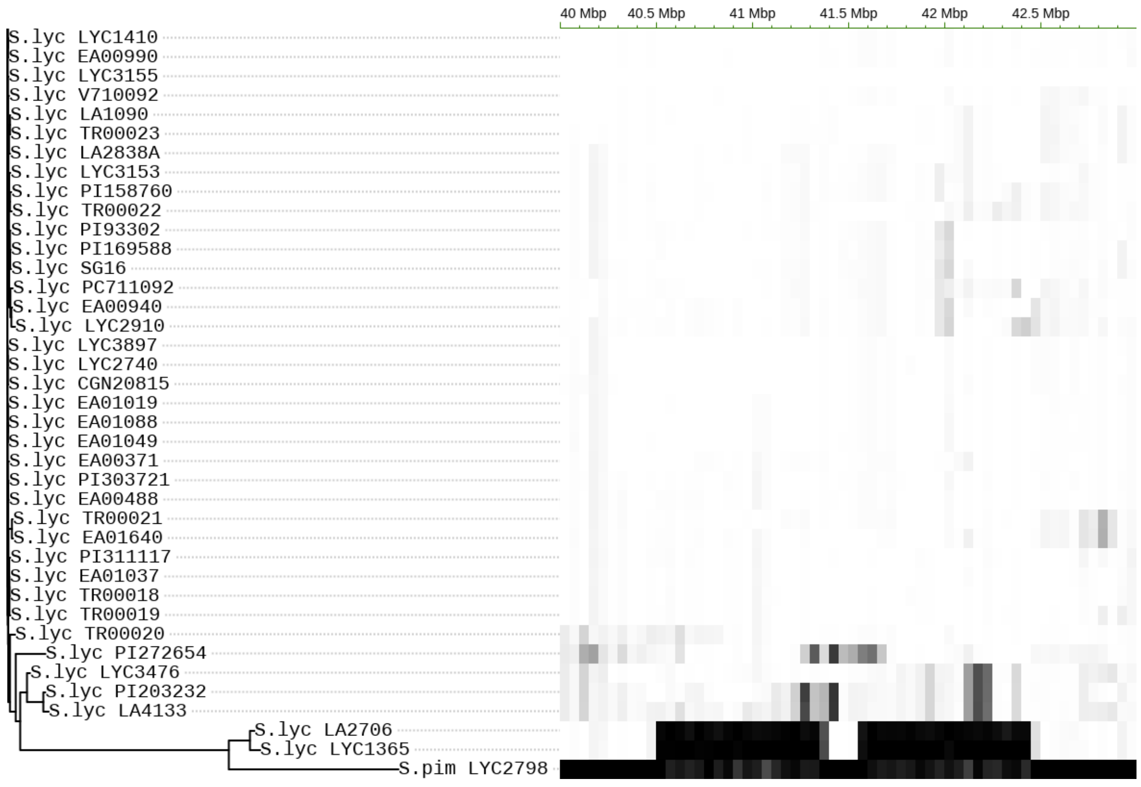


Figure 4-7: Potential introgression from *S. pim* LYC2798 or a similar donor into the LA2706 (MoneyMaker) and LYC1365 (AllRound) cultivars. *The introgression, visible against a background of other tomato cultivars, spans from approximately 40.50 Mbp to 42.45 Mbp on chromosome 6. Both affected cultivars cluster closely with the potential donor within a sharply delineated interval. Note that the Introgression Browser article reports the same introgression as beginning at 36.75 Mbp, but that is due to its use of an older version of the tomato reference (SL2.40).*

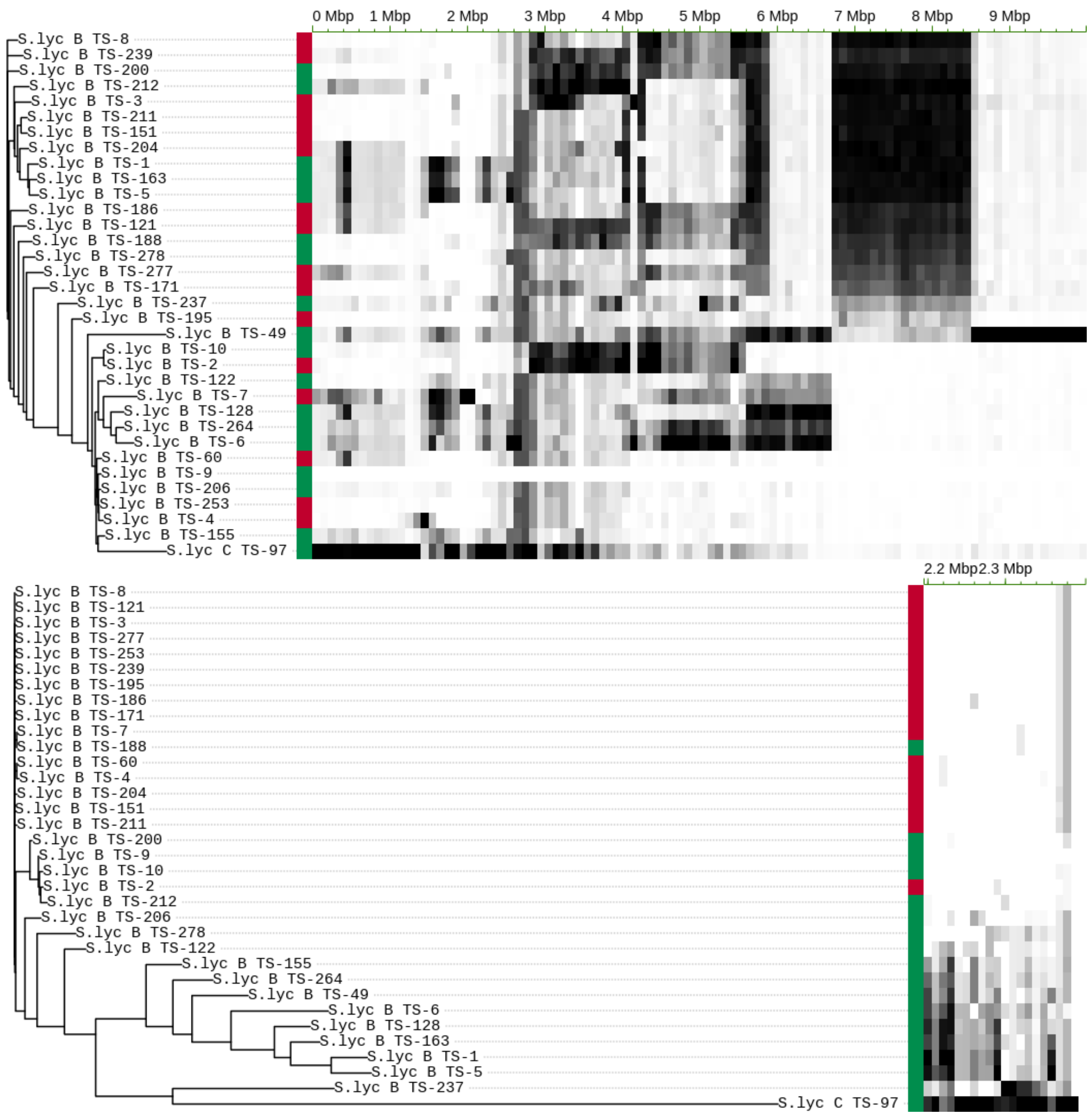


Figure 4-8: TGRC gene annotation used to highlight different alleles of the tomato *uniform ripening* gene in Tersect Browser visualisations. *The two alleles of the gene are marked in red (-- allele) and green (+ allele). The top plot covers the first 10 Mbp of chromosome 10, and its phylogenetic tree structure shows no obvious relationship with the alleles. The bottom plot covers a smaller interval on the same short arm of chromosome 10. The 200 kbp interval (2,195,000 – 2,395,000) is centred on the* uniform ripening *gene locus, and it can be seen that the accessions now cluster more closely according to their alleles.*

4.5 Future work

As discussed in the previous section, benchmarking using the 2548-genome human data set helped identify the cost of repeatedly parsing precomputed distance matrices, which are currently stored in textual PHYLIP files, as a major performance-limiting factor when phylogeny inference requests are executed on smaller subsets of genomes contained in larger indexes. Storing the same data in a more easily parsed binary format or in an indexed database would likely mitigate this issue and could be achieved with minimal changes to the pipeline, as the only actual step which requires textual inputs is the final phylogenetic tree construction with RapidNJ (Simonsen et al., 2008). This would significantly improve performance in the typical use case where a user works with smaller subsets of a larger data set, and is therefore a high-priority feature to be added in future versions of Tersect Browser.

An obvious avenue for performance improvement that has not been explored in the development of the application is parallel computing. This is despite the fact that many of the computational problems involved, such as matrix addition, are trivially parallelizable. The main reason for this omission is that, being a Web application, Tersect Browser is meant for parallel use by multiple users in its typical use case. As the server (or servers) will be fulfilling multiple independent requests from different users in parallel, computational resources such as multiple processors will be used even if the individual requests are fulfilled in a strictly sequential manner. Indeed, even for a single user, the requests for heat map data and phylogenetic trees are made independently, so that the process of plot generation can benefit from this type of primitive parallelism to a limited extent. However, it can be argued that for low-volume usage, for example in a small research group, it is unlikely for the number of concurrent users and requests to be high enough to make good use of the parallel computing resources available on modern servers. This has been the case with Tersect Browser as deployed at Cranfield University for internal use. In such a use case, individual requests could indeed benefit from parallel computing to further improve performance. An optional “parallel processing” mode is therefore a likely feature to be added in future versions of Tersect Browser.

At present the only way to add genomes to Tersect Browser is as part of a Tersect index file when a data set is being created. This means that a user cannot simply add a single new genome to an existing data set, but instead has to create a new data set, using a new index file. This is a limitation that Tersect Browser inherits from Tersect, and overcoming it is a feature under active development for that tool; it is likely to be resolved for both applications once complete.

Tersect Browser uses the Jukes-Cantor model to generate its genetic distance metrics. This means that the rate of nucleotide substitution is treated as equal for all pairs of nucleotides and all sites, with no correction for the higher rate of transitional substitutions. This is a consequence of the Tersect Hamming distance calculations not differentiating between transitions and transversions. However, this could be circumvented, for example by maintaining separate Tersect indices for transitions and transversions, which would allow future versions of Tersect Browser to support other, more sophisticated genetic distance metrics.

Finally, the usability of Tersect Browser is likely to be significantly improved through the addition of an integrated genome browser, allowing users to view individual variants present in the interval of interest directly through the application's main interface. The back-end functionality required for this is already present, as Tersect can efficiently report the variant contents of individual genomes in a stored data set.

4.6 References

A. van Kesteren, A. Gregor, M. (2022). *DOM Living Standard*. https://dom.spec.whatwg.org/

Aflitos, S. A., Sanchez-Perez, G., De Ridder, D., Fransz, P., Schranz, M. E., De Jong, H., & Peters, S. A. (2015). Introgression browser: High-throughput whole-genome SNP visualization. *Plant Journal*, *82*(1), 174–182. https://doi.org/10.1111/tpj.12800

Aflitos, S., Schijlen, E., De Jong, H., De Ridder, D., Smit, S., Finkers, R., Wang, J., Zhang, G., Li, N., Mao, L., Bakker, F., Dirks, R., Breit, T., Gravendeel, B., Huits, H., Struss, D., Swanson-Wagner, R., Van Leeuwen, H., Van Ham, R. C. H. J., … Peters, S. (2014). Exploring genetic variation in the tomato (Solanum section Lycopersicon) clade by whole-genome sequencing. *Plant Journal*, *80*(1), 136–148. https://doi.org/10.1111/tpj.12616

Agrawal, N., Prabhakaran, V., Wobber, T., Davis, J. D., Manasse, M., & Panigrahy, R. (2008). Design tradeoffs for SSD performance. *Proceedings of the 2008 USENIX Annual Technical Conference, USENIX 2008*, 57–70.

Angular. (n.d.). https://angular.io/

Consortium, G. P., Auton, A., Abecasis, G. R., Altshuler, D. M., Durbin, R. M., Abecasis, G. R., Bentley, D. R., Chakravarti, A., Clark, A. G., Donnelly, P., Eichler, E. E., Flicek, P., Gabriel, S. B., Gibbs, R. A., Green, E. D., Hurles, M. E., Knoppers, B. M., Korbel, J. O., Lander, E. S., … National Eye Institute, N. I. H. (2015). A global reference for human genetic variation. *Nature*, *526*(7571), 68–74. https://doi.org/10.1038/nature15393

Evans, P. D., Mekel-Bobrov, N., Vallender, E. J., Hudson, R. R., & Lahn, B. T. (2006). Evidence that the adaptive allele of the brain size gene microcephalin introgressed into Homo sapiens from an archaic Homo lineage. *Proceedings of the National Academy of Sciences*, *103*(48), 18178–18183. https://doi.org/10.1073/pnas.0606966103

Express. (n.d.). https://expressjs.com/

Fairley, S., Lowy-Gallego, E., Perry, E., & Flicek, P. (2019). The International Genome Sample Resource (IGSR) collection of open human genomic variation resources. *Nucleic Acids Research*, *48*(D1), D941–D947. https://doi.org/10.1093/nar/gkz836

Felsenstein, J. (1989). PHYLIP-Phylogeny inference package. *Cladistics*, *5*, 164–166.

Fernandez-Pozo, N., Menda, N., Edwards, J. D., Saha, S., Tecle, I. Y., Strickler, S. R., Bombarely, A., Fisher-York, T., Pujar, A., Foerster, H., Yan, A., & Mueller, L. A. (2015). The Sol Genomics Network (SGN)-from genotype to phenotype to breeding. *Nucleic Acids Research*, *43*(D1), D1036–D1041. https://doi.org/10.1093/nar/gku1195

Fulton, S. (2013). *{HTML5} Canvas* (2nd ed.). O’Reilly Media.

Godfrey, P., Gryz, J., & Lasek, P. (2016). Interactive Visualization of Large Data Sets. *{IEEE} Transactions on Knowledge and Data Engineering*, *28*(8), 2142–2157. https://doi.org/10.1109/tkde.2016.2557324

Jukes, T. H., & Cantor, C. R. (1969). Evolution of Protein Molecules. In *Mammalian Protein Metabolism* (pp. 21–132). Elsevier. https://doi.org/10.1016/b978-1-4832-3211-9.50009-7

Kurowski, T. J., & Mohareb, F. (2019). Tersect: a set theoretical utility for exploring sequence variant data. *Bioinformatics*, *36*(3), 934–935. https://doi.org/10.1093/bioinformatics/btz634

Labate, J. A., & Robertson, L. D. (2012). Evidence of cryptic introgression in tomato (Solanum lycopersicum L.) based on wild tomato species alleles. *BMC Plant Biology*, *12*(1), 133. https://doi.org/10.1186/1471-2229-12-133

Lin, T., Zhu, G., Zhang, J., Xu, X., Yu, Q., Zheng, Z., Zhang, Z., Lun, Y., Li, S., Wang, X., Huang, Z., Li, J., Zhang, C., Wang, T., Zhang, Y., Wang, A., Zhang, Y., Lin, K., Li, C., … Huang, S. (2014). Genomic analyses provide insights into the history of tomato breeding. *Nature Genetics*, *46*(11), 1220–1226. https://doi.org/10.1038/ng.3117\rhttp://www.nature.com/ng/journal/v46/n11/abs/ng.3117.html#supplementary-information

MongoDB. (n.d.). https://www.mongodb.com/

Nelson, T. C., Stathos, A. M., Vanderpool, D. D., Finseth, F. R., Yuan, Y., & Fishman, L. (2021). Ancient and recent introgression shape the evolutionary history of pollinator adaptation and speciation in a model monkeyflower radiation (Mimulus section Erythranthe). *{PLOS} Genetics*, *17*(2), e1009095. https://doi.org/10.1371/journal.pgen.1009095

NodeJS. (n.d.). https://nodejs.org/

Powell, A. L. T., Nguyen, C. V, Hill, T., Cheng, K. L., Figueroa-Balderas, R., Aktas, H., Ashrafi, H., Pons, C., Fernández-Muñoz, R., Vicente, A., Lopez-Baltazar, J., Barry, C. S., Liu, Y., Chetelat, R., Granell, A., Van Deynze, A., Giovannoni, J. J., & Bennett, A. B. (2012). Uniform ripening encodes a Golden 2-like transcription factor regulating tomato fruit chloroplast development. *Science (New York, N.Y.)*, *336*(6089), 1711–1715. https://doi.org/10.1126/science.1222218

PrimeFlex. (n.d.). https://www.primefaces.org/primeflex/

Qi, L., Friebe, B., Zhang, P., & Gill, B. S. (2007). Homoeologous recombination, chromosome engineering and crop improvement. *Chromosome Research*, *15*(1), 3–19. https://doi.org/10.1007/s10577-006-1108-8

Richardson, L., Amundsen, M., & Ruby, S. (2013). *Restful web {APIs}*. O'Reilly Media.

RxJS. (n.d.). https://rxjs.dev/

Shearer, L. A., Anderson, L. K., de Jong, H., Smit, S., Goicoechea, J. L., Roe, B. A., Hua, A., Giovannoni, J. J., & Stack, S. M. (2014). *Fluorescence in situ hybridization and optical mapping to correct scaffold arrangement in the tomato genome*. *4*(8), 1395–1405. https://doi.org/10.1534/g3.114.011197

Shneiderman, B. (1984). Response Time and Display Rate in Human Performance with Computers. *ACM Comput. Surv.*, *16*(3), 265–285. https://doi.org/10.1145/2514.2517

Silva Ferreira, D., Kevei, Z., Kurowski, T., de Noronha Fonseca, M. E., Mohareb, F., Boiteux, L. S., & Thompson, A. J. (2018). BIFURCATE FLOWER TRUSS: a novel locus controlling inflorescence branching in tomato contains a defective MAP kinase gene. *Journal of Experimental Botany*, *69*(10), 2581–2593. https://doi.org/10.1093/jxb/ery076

Simonsen, M., Mailund, T., & Pedersen, C. (2011). Inference of Large

Phylogenies Using Neighbour-Joining. *CCIS*, *127*. https://doi.org/10.1007/978-3-642-18472-7_26

Simonsen, M., Mailund, T., & Pedersen, C. N. S. (2008). Rapid Neighbour-Joining. In K. A. Crandall & J. Lagergren (Eds.), *Algorithms in Bioinformatics* (pp. 113–122). Springer Berlin Heidelberg.

WHATWG. (2022). *HTML Living Standard - 4.12.5 The canvas element*. https://html.spec.whatwg.org/multipage/canvas.html

Youngjae, K., Gupta, A., Urgaonkar, B., Berman, P., & Sivasubramaniam, A. (2011). *HybridStore: A Cost-Efficient, High-Performance Storage System Combining SSDs and HDDs*. 227–236. https://doi.org/10.1109/MASCOTS.2011.64

Zheng-Bradley, X., Streeter, I., Fairley, S., Richardson, D., Clarke, L., & Flicek, P. (2017). Alignment of 1000 Genomes Project reads to reference assembly GRCh38. *GigaScience*, *6*(7), 1–8. https://doi.org/10.1093/gigascience/gix038

Zhou, Y., & Browning, S. R. (2021). Protocol for detecting introgressed archaic variants with SPrime. *STAR Protocols*, *2*(2), 100550. https://doi.org/https://doi.org/10.1016/j.xpro.2021.100550

5 OVERALL DISCUSSION

5.1 Overview

All three objectives which were pursued in the thesis have produced outputs with real-world impact. Investigation of issues associated with resequenced genome validation led to the discovery and correction of major errors in two sets of publicly available tomato genome resources used by plant biologists. Among the software outputs, Tersect now offers the bioinformatics community the highest performance among similar tools when it comes to comparing and otherwise operating on large sets of variant data and is usable both as a standalone tool and as a building block for creating higher-level applications. Tersect Browser, which is one such application, is the first Web tool which can produce fully interactive visualisations of phylogenetic relationships and genetic distance comparisons between large numbers of genomes based on whole-genome SNV data, and is likely to prove useful to biologists in a wide array of analyses, including the characterisation of introgressions and pedigree analysis.

5.2 Lift-over and validation

The lift-over errors which were the focus of **Chapter 2**, and which remained unnoticed and uncorrected as part of a public resource used by the plant biologist community for seventeen months before being fixed, revealed a somewhat lax approach to VCF validation in many applications. After all, the problem areas covered nearly 15% of the total genome size, and some of the issues could (in principle) be seen with the naked eye in the genome browser hosted by SGN (see **Figure 2-7** and **Figure 2-8**), which would display the incorrect REF alleles alongside the mismatched reference sequence without any errors or warnings. In fact, some VCF format validators do not validate REF alleles at all, and would not find the errors in question – this notably includes VCFtools, which introduced the VCF format in the first place (Danecek et al., 2011). The issue highlights a need for stricter validation and the use of tools like GATK's ValidateVariants (McKenna et al., 2010).

It is notable that the two resequenced genome data sets maintained by SGN have not seen further official updates (at least at the time of writing), despite two new versions of the reference genome being released. At least in part this is likely to be due to the computational cost that re-doing the alignment and variant calling would carry. At the same time the primary alternative – coordinate lift-over – is more error prone, which would be compounded by the fact that both of the new reference genome versions have been generated *de novo*, based on new, long-read data, instead of being modifications of the previous version as in the SL2.40 to SL2.50 update (Hosmani et al., 2019). The SeqRemap lift-over pipeline described in **Section 2.5.4** was used to update some of the SGN data to SL4.0 for internal use at Cranfield University.

5.3 Tersect Browser and Tersect

The benchmarking of Tersect Browser demonstrated that it can deliver latencies that allow for interactive use even for relatively large data sets that include hundreds (for tomato) or even thousands (for human) of genomes. However, certain algorithmic limitations also became evident, as the tests revealed the total number of genomes to be a limiting performance factor for data sets. This is a major issue, as it would be ideal for an interactive system like Tersect Browser to allow for all comparable genomes to be stored (and viewed) together, in one data set, which could then be arbitrarily filtered by the users as needed. Yet if the total number of stored genomes lowers performance even when only a subset is visualised, it would instead be better to split the data up into smaller data sets in the first place. Still, this issue (or at least its current magnitude) was identified as the consequence of the approach used for storing precomputed results, and it may be possible to correct it as discussed in **Section 4.5**.

Perhaps a more important and harder to address issue, which is partially obscured by the limitation discussed above, is that the computational complexity of the algorithms used by Tersect Browser scales quadratically with the number of genomes. At the same time, the tuneable parameters (precomputed partition sizes) offer only linear increases at a significant cost of precomputation time and storage space. The 2548-genome human data set, with its ~30 second visualisation times when all the genomes are used, may in fact be near the limit of what Tersect Browser can handle in a broadly interactive manner. The use of larger data sets will likely require entirely different algorithmic approaches, for example the use of incremental or approximate solutions based on sampling the data, rather than exact solutions based on all the relevant data (Godfrey et al., 2016).

With regards to Tersect, it should first be noted that it was not developed as a subsystem of Tersect Browser. Instead, it is Tersect Browser which takes advantage of Tersect as the highest-performance, most appropriate tool available to deliver the back-end functionality the Web application requires. This distinction is important, as Tersect is intended to be a generic, stand-alone utility, usable in many different contexts, including other systems and pipelines.

No steps were therefore taken to allow for closer integration between Tersect and Tersect Browser on the side of the smaller application, with the sole exception being the addition of JSON as a format option for outputting genetic distance calculation results – an unusual priority in format support for a lightweight command-line utility, but very useful for downstream use by NodeJS applications, being their native object format.

Tersect has already seen independent use in other projects, including one which resulted in a peer-reviewed publication (Kangara et al., 2020), included as 5.4Appendix D. The work was concerned with developing a procedure for generating mutant populations of *Puccinia graminis* f. sp. *tritici* spores and screening them for gain-of-virulence mutants. *Puccinia graminis* is a fungal pathogen of cereal crops that causes stem rust, responsible for global grain losses representing around 1% of the annual wheat yield (Beddow et al., 2013), with much larger average losses of 30-40% being recorded on a regional scale in the past decade (Saunders et al., 2019). It is reported that, in particularly favourable environmental conditions, explosive outbreaks can cause losses as high as 50-70% over a region (Schumann & Leonard, 2000).

Wheat stem rust has seen a resurgence in recent years, linked with the emergence of fungal lineages that have overcome several wheat stem resistance genes, and have thus become virulent against certain current cultivars (Olivera Firpo et al., 2017; Pretorius et al., 2000). Stacking multiple resistance genes is an approach used to maximize the durability of plants to disease (Fukuoka et al., 2015; Zhang et al., 2009). Individually assessing the functionality of resistance genes in a stack requires the use of a matching pathogen effector probe per resistance gene, which in turn requires the identification of avirulence genes for each effector (Wulff & Moscou, 2014).

The method optimized by Kangara et al. seeks to aid in identifying candidate avirulence genes via ethyl methanesulfonate (EMS) mutagenesis followed by sequence comparison, in which independently derived gain-of-virulence mutants are expected to exhibit independent mutations in the same genes. Variant calling of mutagenized spore sequence data, mapped to a *Puccinia graminis* f. sp. *tritici* reference genome, was the basis of the sequence comparison. This presented an excellent use case for Tersect, which specializes in comparing variant contents between samples. In particular, it was used to exclude potential false positive variant calls by removing variants which co-occurred in three or more (out of seven) samples, as precisely identical mutations are unlikely to have been induced independently. The bioinformatics pipeline used in the work was used to measure the induced mutation density as a function of mutagen concentration and estimate the number of independent mutations required to identify avirulence effector genes in *Puccinia graminis* f. sp. *tritici*.

Both the pipeline mentioned above, and other, unpublished work have pointed to a certain limitation, arguably a key missing feature in Tersect, resulting from its narrow focus. The “sets” considered by Tersect and encoded in its indexing scheme are sets of variants, with each set corresponding to some specific genome or the result of set theoretical operations on multiple genomes (referred to as a “virtual genome”). This is central to its performance and allows the declarative query language used by Tersect to be very expressive when it comes to comparing sets between genomes. However, the tool is poorly equipped to answer queries about sets of genomes and their correspondence to specific variants, which is an inversion of the typical Tersect logic. An example would be attempting to find all genomes which contain a specific variant. The query language cannot, at present, express this sort of query directly, and the indexing scheme is poorly optimized for answering it.

A temporary workaround was developed for the purposes of the aforementioned publication – a Python wrapper script would convert such an exotic query into a much more complex one, which utilised the current syntax of Tersect and called the tool multiple times. However, a proper solution would be to add such a feature to Tersect directly, ideally with a change or expansion to the current indexing scheme to allow for good performance. Conceptually, if one treats a set of parallel bit arrays like those stored in Tersect index files as a matrix, a transposition of that matrix could serve as an index usable for this sort of "inverted" query. There exist methods for such transposition of bitmap indices (Nguyen et al., 2016a, 2016b), as bitmap indices in general are an established indexing method in scientific databases (Sinha et al., 2006). However, no implementation of a transposition method specific to the parallel, WAH-compressed bit arrays (Wu et al., 2006) used by Tersect currently exists, and creating one would not be trivial. It would therefore be necessary to precompute and store two complete sets of bit arrays, increasing (approximately doubling) the size of Tersect indices.

As with Tersect Browser, the above issue demonstrates the limitations of the system revealed in confrontation with real-life data, while pointing to an issue whose resolution would significantly improve the system.

5.4 References

Beddow, J. M., Hurley, T. M., Kriticos, D. J., & Pardey, P. G. (2013). Measuring the global occurrence and probabilistic consequences of wheat stem rust. *HarvestChoice Technical Note*, *c*, 23.

Danecek, P., Auton, A., Abecasis, G., Albers, C. A., Banks, E., DePristo, M. A., Handsaker, R. E., Lunter, G., Marth, G. T., Sherry, S. T., McVean, G., & Durbin, R. (2011). The variant call format and VCFtools. *Bioinformatics*, *27*(15), 2156–2158. https://doi.org/10.1093/bioinformatics/btr330

Fukuoka, S., Saka, N., Mizukami, Y., Koga, H., Yamanouchi, U., Yoshioka, Y., Hayashi, N., Ebana, K., Mizobuchi, R., & Yano, M. (2015). Gene pyramiding enhances durable blast disease resistance in rice. *Scientific Reports*, *5*. https://doi.org/10.1038/srep07773

Godfrey, P., Gryz, J., & Lasek, P. (2016). Interactive Visualization of Large Data Sets. *{IEEE} Transactions on Knowledge and Data Engineering*, *28*(8), 2142–2157. https://doi.org/10.1109/tkde.2016.2557324

Hosmani, P. S., Flores-Gonzalez, M., van de Geest, H., Maumus, F., Bakker, L. V, Schijlen, E., van Haarst, J., Cordewener, J., Sanchez-Perez, G., Peters, S., Fei, Z., Giovannoni, J. J., Mueller, L. A., & Saha, S. (2019). *An improved de novo assembly and annotation of the tomato reference genome using single-molecule sequencing, Hi-C proximity ligation and optical maps*. https://doi.org/10.1101/767764

Kangara, N., Kurowski, T. J., Radhakrishnan, G. V, Ghosh, S., Cook, N. M., Yu, G., Arora, S., Steffenson, B. J., Figueroa, M., Mohareb, F., Saunders, D. G. O., & Wulff, B. B. H. (2020). Mutagenesis of Puccinia graminis f. sp. tritici and Selection of Gain-of-Virulence Mutants. *Frontiers in Plant Science*, *11*, 570180. https://doi.org/10.3389/fpls.2020.570180

McKenna, A., Hanna, M., Banks, E., Sivachenko, A., Cibulskis, K., Kernytsky, A., Garimella, K., Altshuler, D., Gabriel, S., Daly, M., & DePristo, M. A. (2010). The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Research*, *20*(9), 1297–1303. https://doi.org/10.1101/gr.107524.110

Nguyen, X. T., Nguyen, H. T., & Pham, C. K. (2016a). A bit-level matrix transpose for bitmap-index-based data analytics. *2016 IEEE 6th International Conference on Communications and Electronics, IEEE ICCE 2016*, 217–220. https://doi.org/10.1109/CCE.2016.7562639

Nguyen, X. T., Nguyen, H. T., & Pham, C. K. (2016b). An FPGA approach for fast bitmap indexing. *IEICE Electronics Express*, *13*(4). https://doi.org/10.1587/elex.13.20160006

Olivera Firpo, P. D., Newcomb, M., Flath, K., Sommerfeldt-Impe, N., Szabo, L. J., Carter, M., Luster, D. G., & Jin, Y. (2017). Characterization of Puccinia graminis f. sp. tritici isolates derived from an unusual wheat stem rust outbreak in Germany in 2013. *Plant Pathology*, *66*(8), 1258–1266. https://doi.org/10.1111/ppa.12674

Pretorius, Z. A., Singh, R. P., Wagoire, W. W., & Payne, T. S. (2000). Detection of virulence to wheat stem rust resistance gene Sr31 in Puccinia graminis. f. sp. tritici in uganda. *Plant Disease*, *84*(2). https://doi.org/10.1094/PDIS.2000.84.2.203B

Saunders, D. G. O., Pretorius, Z. A., & Hovmøller, M. S. (2019). Tackling the re-emergence of wheat stem rust in Western Europe. *Communications Biology*, *2*(1). https://doi.org/10.1038/s42003-019-0294-9

Schumann, G. L., & Leonard, K. J. (2000). Stem rust of wheat (black rust). *The Plant Health Instructor*. https://doi.org/10.1094/phi-i-2000-0721-01

Sinha, R. R., Mitra, S., & Winslett, M. (2006). Bitmap indexes for large scientific data sets: A case study. *20th International Parallel and Distributed Processing Symposium, IPDPS 2006*, *2006*. https://doi.org/10.1109/IPDPS.2006.1639304

Wu, K., Otoo, E. J., & Shoshani, A. (2006). Optimizing bitmap indices with efficient compression. *ACM Transactions on Database Systems*, *31*(1), 1–38. https://doi.org/10.1145/1132863.1132864

Wulff, B. B. H., & Moscou, M. J. (2014). Strategies for transferring resistance into wheat: From wide crosses to GM cassettes. *Frontiers in Plant Science*, *5*(DEC). https://doi.org/10.3389/fpls.2014.00692

Zhang, N. W., Pelgrom, K., Niks, R. E., Visser, R. G. F., & Jeuken, M. J. W. (2009). Three combined quantitative trait loci from nonhost lactuca saligna are sufficient to provide complete resistance of lettuce against bremia lactucae. *Molecular Plant-Microbe Interactions*, *22*(9), 1160–1168. https://doi.org/10.1094/MPMI-22-9-1160

APPENDICES

Appendix A Tersect User Manual

Tersect is a command-line utility for conducting fast set theoretical operations and genetic distance estimation on biological sequence variant data. The tool generates index files based on provided variant data (VCF files) which can then be used to rapidly execute flexible set theoretical queries and output the resulting lists of variants in selected regions.

Tersect is intended to allow for highly responsive, exploratory interaction with variant data as well as for integration with larger tools and pipelines. It follows the Samtools/tabix convention for specifying genomic regions which allows for much faster operations and more manageable output sizes.

Tersect can also be used to provide estimates of genetic distance between sets of samples, using the number of differing sites as a proxy for distance measures.

Table of Contents

- Overview
- Table of Contents
- Installation
  - Pre-compiled releases
    - Linux
    - macOS
  - Building Tersect from source
    1. Cloning the repository
    2. Building
    3. Installing
- Example data
- Building a Tersect index
- Inspecting a Tersect index
- Set operations
  - Overview
  - Queries
  - Genomes
  - Binary operators
  - Genome list
  - Functional operators
  - Regions

Installation

Pre-compiled releases

Tersect packages and binaries are available for download below:

Linux

- 64-bit binaries:
  https://github.com/tomkurowski/tersect/releases/download/v0.12.0/tersect-0.12.0-Linux.tar.gz
- 64-bit .deb package (Debian, Ubuntu):
  https://github.com/tomkurowski/tersect/releases/download/v0.12.0/tersect-0.12.0-Linux.deb
- 64-bit .rpm package (Fedora, openSUSE):
  https://github.com/tomkurowski/tersect/releases/download/v0.12.0/tersect-0.12.0-Linux.rpm

macOS

- 64-bit binaries:
  https://github.com/tomkurowski/tersect/releases/download/v0.12.0/tersect-0.12.0-macOS.tar.gz

Building Tersect from source

Building Tersect from source requires CMake version 3.1+ as well as Flex (lexical analyzer) version 2.5+ and Bison (parser generator) version 2.6+.

1. Cloning the repository

```
git clone https://github.com/tomkurowski/tersect.git
```

2. Building

For an out-of-source build after cloning the repository use the following commands:

```
cd tersect
mkdir build
cd build
cmake ..
make
```

3. **Installing**

This step may require elevated permissions (e.g., prefacing the command with `sudo`). The default installation location for Tersect is `/usr/local/bin`.

```
make install
```

Example data

Two archives containing example Tersect index files (.tsi) are available for download below to allow you to try out the utility without needing to create an index file yourself.

The first index contains human genomic variant data for 2504 individuals from the 1000 Genomes Project. While Tersect is capable of handling the entire human genome, the index below is limited to chromosome 1 to make the example archive smaller and quicker to download.

The second index contains tomato genomic variant data for 360 tomato accessions from the AGIS Tomato 360 Resequencing Project and 84 accessions from the Wageningen UR 150 Tomato Genome ReSequencing Project for a combined data set of 444 accessions. Samples have been renamed according to a provided key (accession_names.tsv) to make them more informative and consistent between the two source data sets.

Note: the index files provided below are compressed using gzip and need to be uncompressed before use.

- 2504 human genomes, chromosome 1:
  https://github.com/tomkurowski/tersect/releases/download/v0.12.0/human_chr1.tsi.gz
- 444 tomato genomes and sample names:
  https://github.com/tomkurowski/tersect/releases/download/v0.12.0/tomato.tsi.gz
  https://github.com/tomkurowski/tersect/releases/download/v0.12.0/accession_names.tsv

Building a Tersect index

You can build your own Tersect index based on a set of VCF files using the tersect build command. You need to provide a name for your index file (a .tsi extension will be added if you omit it) as the first argument, followed by any number of input VCF files (which may be compressed using gzip) to be included in the index.

Please note that although from a technical point of view, Tersect would work even if your VCF files were called against different reference genomes or versions of the same reference, the biological context of your theoretical operations will not be accurate (depending on how different the reference genomes used). Therefore, we strongly recommend using VCF files called against the same reference version.

Example:

The command below builds a Tersect index file named *tomato.tsi* which includes variants from all *vcf.gz* files in the *data* directory. Depending on the input size this can take several minutes.

```
foo@bar:~$ tersect build tomato.tsi ./data/*.vcf.gz
```

Optionally, you can also provide a `--name-file` input file containing custom sample names to be used by Tersect. These names will replace the default sample IDs defined in the input VCF header lines. The `--name-file` should be a tab-delimited file containing two columns, the first with the sample IDs to be replaced and the second with the names to be used by Tersect. An example is shown below:

```
TS-1    S.lyc B TS-1
TS-10   S.lyc B TS-10
TS-100  S.lyc B TS-100
TS-101  S.lyc B TS-101
TS-102  S.lyc B TS-102
TS-103  S.lyc B TS-103
TS-104  S.lyc B TS-104
TS-108  S.lyc B TS-108
TS-11   S.lyc B TS-11
TS-110  S.lyc B TS-110
```

You can also modify sample names in an existing Tersect index file by using the `tersect rename` command.

It is worth noting that the descriptive fields of the VCF files are not stored within the Tersect database. The reason for that is once an operation is performed on two of more VCF files, these fields will be discarded anyway as they are genotype-specific. However, you should be able to retrieve it back by intersecting Tersect's output with any VCF files from this list.

Inspecting a Tersect index

The data contained in a Tersect index file can be inspected using several commands. The `tersect chroms` command prints information on the number of variants present in each of the reference chromosomes as well as the chromosome names and size.

Note: In the absence of a reference file, the *length* of each chromosome is represented by the position of the last variant, which will always be an underestimate.

Example:

The command below prints the per-chromosome variant content of the example Tersect index file named *tomato.tsi*:

```
foo@bar:~$ tersect chroms tomato.tsi
Chromosome  Length  Variants
SL2.50ch00  21805702    1343815
SL2.50ch01  98543411    9965680
SL2.50ch02  55340384    5189338
SL2.50ch03  70787603    6741448
SL2.50ch04  66470926    7257520
SL2.50ch05  65875078    6830857
SL2.50ch06  49751619    4870941
SL2.50ch07  68044764    6868152
SL2.50ch08  65866627    6504025
SL2.50ch09  72481975    7102356
SL2.50ch10  65527500    6712293
SL2.50ch11  56302478    5367032
SL2.50ch12  67145147    6719621
```

The tersect samples command prints the names of samples present in a Tersect index file. These can be either all samples or a subset based on a naming pattern (the `--match` parameter) and/or on the presence of specific variants (the `--contains` parameter).

Sample name patterns can include wildcard symbols (`*`) which match zero or more characters. For example, a pattern like `"S.lyc*"` will match all samples whose names begin with "`S.lyc`". A lone wildcard character matches all samples stored in the Tersect index file.

If you specify a list of variants via the `--contains` parameter, only samples which contain each of those variants will be printed. The variant format should look as follows: `chromosome:position:ref:alt` where `ref` and `alt` are reference and alternate alleles. Multiple variants can be included, separated by commas (e.g., `chr1:1245:A:G,chr8:5300:T:A`).

Examples:

The command below prints the names of samples matching the `"S.gal*"` wildcard pattern contained in the example Tersect index file *tomato.tsi*.

```
foo@bar:~$ tersect samples tomato.tsi -m "S.gal*"
Sample
S.gal W TS-208
S.gal LA1044
S.gal LA1401
S.gal LA0483
```

The command below prints the names of all samples containing both a T/G SNV at position 100642 on chromosome 3 and an A/G SNV at position 5001015 on chromosome 6 contained in the example Tersect index file *tomato.tsi*.

```
foo@bar:~$ tersect samples tomato.tsi -c
"SL2.50ch03:100642:T:G,SL2.50ch06:5001015:A:G"
Sample
S.lyc LA1479
S.pen LA0716
S.hab LYC4
S.hab LA0407
S.hab LA1777
S.hab LA1718
S.hab CGN15792
S.hab PI134418
S.hab CGN15791
S.chm LA2695
```

Set operations

Overview

Tersect can interpret and display the results of set theoretical commands using the `tersect view` command. This is the primary and most flexible functionality of the application and allows the user to construct arbitrarily complex queries. The expected format of a tersect view query is as follows:

```
 tersect view [options] index.tsi QUERY [REGION1...]
```

Queries

A query is a command interpreted and evaluated by Tersect which (if successful) prints either a list of variants (if the result is a single genome or virtual genome) or a list of genome sample names (if the result is a list of genomes). The simplest query consists of a genome name and prints out the variants belonging to that genome. More advanced queries can contain complex combinations of operations described in the sections below.

Note: The term *virtual genome* refers to a collection of variants not representing a specific genome - for example, the symmetric difference of two genomes (the collection of variants which appear in one but not both genomes). Tersect treats these *virtual genomes* the same way it treats "real" genomes so they can be used as operands in nested queries.

Genomes

Genomes can be referred to by their sample name, which is either taken from the header line of the source VCF file or set by the user either manually (see `tersect rename`) or through a tab-delimited name file (see `--name-file` in `tersect build` and `tersect rename`). A sample name can be of any length and can include any characters (including whitespace) except for single quotes (`'`). However, if a sample name includes whitespace, parentheses, or characters used as Tersect operators (`-^&|>,\`), it has to be surrounded by single quotes.

If the query is (or results in) a single genome or virtual genome, the variants contained by that one genome are printed out.

Example:

Print out all the variants belonging to the "S.hab LYC4" genome in the *tomato.tsi* Tersect index file:

```
foo@bar:~$ tersect view tomato.tsi "'S.hab LYC4'"
##fileformat=VCFv4.3
##tersectVersion=0.11.0
##tersectCommand='S.hab LYC4'
#CHROM  POS ID  REF ALT QUAL    FILTER  INFO
SL2.50ch00  391 .   C   T   .   .   .
SL2.50ch00  416 .   T   A   .   .   .
SL2.50ch00  734 .   T   G   .   .   .
SL2.50ch00  759 .   C   T   .   .   .
SL2.50ch00  771 .   A   G   .   .   .
SL2.50ch00  778 .   T   A   .   .   .
...
```

Note: The sample name had to be surrounded by single quotes because it contains a whitespace character.

Binary operators

Tersect supports four basic binary operators. Each operand has to be a **single** genome. All four operators have the same precedence and are left-associative. You can use parentheses to enforce precedence other than simple left-to-right.

Table A-1: Tersect binary operators.

Operator	Name	Usage	Result
`&`	intersection	`GENOME1 & GENOME2`	Virtual genome containing variants found in both `GENOME1` and `GENOME2`
`|`	union	`GENOME1 | GENOME2`	Virtual genome containing variants found in `GENOME1`, `GENOME2`, or both
`\`	difference	`GENOME1 \ GENOME2`	Virtual genome containing variants found in `GENOME1` but not in `GENOME2`
`^`	symmetric difference	`GENOME1 ^ GENOME2`	Virtual genome containing variants found in `GENOME1` or `GENOME2` but not in both

The result of a binary operation is treated as a single genome (though it does not have a sample name) called a *virtual genome*, which can be used in further operations.

Examples:

Print out the variants shared by `'S.hua LA1983'` and `'S.pim LYC2798'`:

```
foo@bar:~$ tersect view tomato.tsi "'S.hua LA1983' & 'S.pim LYC2798'"
##fileformat=VCFv4.3
##tersectVersion=0.11.0
##tersectCommand='S.hua LA1983' & 'S.pim LYC2798'
#CHROM  POS ID  REF ALT QUAL    FILTER  INFO
SL2.50ch00  3235    .   A   G   .   .   .
SL2.50ch00  3277    .   A   G   .   .   .
SL2.50ch00  3873    .   C   T   .   .   .
...
```

Print out the variants which appear in only one of `'S.gal LA1044'` or `'S.gal W TS-208'`:

```
foo@bar:~$ tersect view tomato.tsi "'S.gal LA1044' ^
'S.gal W TS-208'"
##fileformat=VCFv4.3
##tersectVersion=0.11.0
##tersectCommand='S.gal LA1044' ^ 'S.gal W TS-208'
#CHROM  POS ID  REF ALT QUAL    FILTER  INFO
SL2.50ch00  362 .   G   T   .   .   .
SL2.50ch00  867 .   G   T   .   .   .
SL2.50ch00  1198    .   G   A   .   .   .
...
```

Print out the variants which appear in `'S.chi CGN15532'` but not `'S.chi CGN15530'` or `'S.chi W TS-408'`:

```
foo@bar:~$ tersect view tomato.tsi "'S.chi CGN15532' \
'S.chi CGN15530' \ 'S.chi W TS-408'"
##fileformat=VCFv4.3
##tersectVersion=0.11.0
##tersectCommand='S.chi CGN15532' \ 'S.chi CGN15530' \
'S.chi W TS-408'
#CHROM  POS ID  REF ALT QUAL    FILTER  INFO
SL2.50ch00  1163    .   C   G   .   .   .
SL2.50ch00  1811    .   C   G   .   .   .
SL2.50ch00  1818    .   C   A   .   .   .
...
```

Note: A more convenient way to conduct the same operation on many genomes is by using functional operators (see **Table A-3**).

Genome list

Instead of individual genomes, Tersect can also operate on lists of genomes. These can be selected using wildcard patterns matching genome sample names, with the most general pattern of a lone wildcard operator (`*`) matching *all* the genomes in the Tersect index file. Individual genomes can also be appended to lists using commas (`,`) or removed from lists using minus signs (`-`).

Genome lists can also be filtered (using the > operator) by whether they contain a specified list of variants. The variant format should look as follows: `chromosome:position:ref:alt` where `ref` and `alt` are reference and alternate alleles. Multiple variants can be included, separated by commas (e.g., `chr1:1245:A:G,chr8:5300:T:A`).

Table A-2: Tersect genome list operators.

Operator	Name	Usage	Result
`*`	wildcard	`PATTERN`	Genome list containing all genomes whose sample names match the provided wildcard pattern
`,`	append	`GENOMELIST, GENOME`	Genome list containing all genomes in `GENOMELIST` and `GENOME`
`-`	remove	`GENOMELIST - GENOME`	Genome list containing all genomes in `GENOMELIST` except `GENOME`
`>`	superset	`GENOMELIST > VARIANTLIST`	Genome list containing all genomes in `GENOMELIST` which contain all variants in `VARIANTLIST`

Note: Tersect does not distinguish between a genome list which contains only one genome and a single genome. The former can be used in binary operations and the latter can be used in functional operations or in constructing genome lists.

If the query is (or results in) a genome list, the list of their genome sample names is printed out.

Examples:

Print out all the names of genomes which begin with "S.pim":

```
foo@bar:~$ tersect view tomato.tsi "S.pim*"
S.pim P TS-92
S.pim P TS-79
S.pim P TS-77
S.pim P TS-50
S.pim P TS-441
S.pim P TS-440
S.pim P TS-439
S.pim P TS-438
...
```

Print out all the names of genomes which contain an A/G single nucleotide variant at position 828587 in chromosome 7:

```
foo@bar:~$ tersect view tomato.tsi "* > SL2.50ch07:828587:A:G"
S.lyc C TS-97
S.lyc C TS-94
S.pim P TS-79
S.pim P TS-77
S.lyc B TS-68
S.lyc C TS-53
S.pim P TS-50
S.pim P TS-441
...
```

Print out all the names of genomes which contain a G/A SNV at position 1590608 in chromosome 5 and a T/C SNV at position 5230 in chromosome 12, except for `'S.gal LA1401'` and those whose names begin with "`S.pim`":

```
foo@bar:~$ tersect view tomato.tsi "* >
SL2.50ch05:1590608:G:A,SL2.50ch12:5230:T:C -
('S.pim*','S.gal LA1401')"
S.lyc C TS-431
S.lyc C TS-430
S.lyc LA1314
```

Functional operators

Functional operators are used to conduct operations on genome lists instead of individual genomes.

Table A-3: Tersect functional operators.

Operator	Name	Usage	Result

`union()`
`u()`
	arbitrary union	`union(GENOMELIST)`
		`u(GENOMELIST)`
			Virtual genome containing all variants contained in any of the genomes in `GENOMELIST`

`intersect()`
`i()`
	arbitrary intersection	`intersect(GENOMELIST)`
		`i(GENOMELIST)`
			Virtual genome containing all variants which appear in every genome in `GENOMELIST`

The result of a functional operation is treated as a single genome (though it does not have a sample name).

Examples:

Union of all genomes, containing every variant recorded in the *tomato.tsi* Tersect index file:

```
foo@bar:~$ tersect view tomato.tsi "u(*)"
##fileformat=VCFv4.3
##tersectVersion=0.11.0
##tersectCommand=u(*)
#CHROM  POS ID  REF ALT QUAL    FILTER  INFO
SL2.50ch00  280 .   A   C   .   .   .
SL2.50ch00  284 .   A   G   .   .   .
SL2.50ch00  316 .   C   T   .   .   .
SL2.50ch00  323 .   C   T   .   .   .
SL2.50ch00  332 .   A   T   .   .   .
SL2.50ch00  362 .   G   T   .   .   .
...
```

Intersection of all genomes which contain a T/A single nucleotide variant at position 12547 in chromosome 12, containing all variants that are shared by each of those genomes:

```
foo@bar:~$ tersect view tomato.tsi "i(* > SL2.50ch12:12547:T:A)"
##fileformat=VCFv4.3
##tersectVersion=0.11.0
##tersectCommand=i(* > SL2.50ch12:12547:T:A)
#CHROM  POS ID  REF ALT QUAL    FILTER  INFO
SL2.50ch00  16576   .   T   C   .   .   .
SL2.50ch00  26171   .   G   T   .   .   .
SL2.50ch00  29880   .   A   G   .   .   .
SL2.50ch00  37486   .   T   G   .   .   .
SL2.50ch00  40476   .   G   T   .   .   .
SL2.50ch00  436850  .   A   G   .   .   .
...
```

Print all the variants which appear only in genome S.hab CGN15792. This is achieved by finding the difference of that genome and the union of all genomes except S.hab CGN15792:

```
foo@bar:~$ tersect view tomato.tsi "'S.hab CGN15792' \ u(* -
'S.hab CGN15792')"
##fileformat=VCFv4.3
##tersectVersion=0.11.0
##tersectCommand='S.hab CGN15792' \ u(* - 'S.hab CGN15792')
#CHROM  POS ID  REF ALT QUAL    FILTER  INFO
SL2.50ch00  1163    .   C   T   .   .   .
SL2.50ch00  1596    .   G   A   .   .   .
SL2.50ch00  2048    .   G   A   .   .   .
SL2.50ch00  2933    .   G   A   .   .   .
SL2.50ch00  2987    .   A   T   .   .   .
...
```

Regions

By default, queries are executed, and results are returned for the entire genome. However, it is possible to selectively execute a query only on a specified region. The familiar tabix/samtools format `chromosome:beginPos-endPos` is used to specify those regions. The coordinates are one-based and inclusive.

Limiting queries to regions allows for much faster execution since far fewer positions need to be processed and printed, capturing only intervals of interest. This feature makes it possible to use Tersect's flexible queries as a high-performance part of a larger pipeline or the back-end of a highly responsive, interactive application.

Example:

Print a union, that is, all the variants appearing either in genome `'S.lyc SG16'`, `'S.lyc LA1421'`, or both, from the first 90 kbp of chromosome 2 in the *tomato.tsi* index file:

```
foo@bar:~$ tersect view tomato.tsi "'S.lyc SG16' | 'S.lyc LA1421'"
SL2.50ch02:1-90000
##fileformat=VCFv4.3
##tersectVersion=0.11.0
##tersectCommand='S.lyc SG16' | 'S.lyc LA1421'
##tersectRegion=SL2.50ch02:1-90000
#CHROM  POS     ID      REF     ALT     QUAL    FILTER  INFO
SL2.50ch02      204     .       A       G       .       .       .
SL2.50ch02      255     .       TCC     TCCC    .       .       .
SL2.50ch02      255     .       TCC     TCCCC   .       .       .
SL2.50ch02      2382    .       G       A       .       .       .
SL2.50ch02      13383   .       G       A       .       .       .
SL2.50ch02      21752   .       C       T       .       .       .
SL2.50ch02      24538   .       T       C       .       .       .
SL2.50ch02      29276   .       G       T       .       .       .
SL2.50ch02      71245   .       A       C       .       .       .
SL2.50ch02      73326   .       C       T       .       .       .
SL2.50ch02      86236   .       C       A       .       .       .
SL2.50ch02      86601   .       A       G       .       .       .
SL2.50ch02      86635   .       T       A       .       .       .
SL2.50ch02      86695   .       T       C       .       .       .
SL2.50ch02      86769   .       G       A       .       .       .
SL2.50ch02      87079   .       T       A       .       .       .
```

Appendix B Tersect Browser Supplementary Figures and Tables

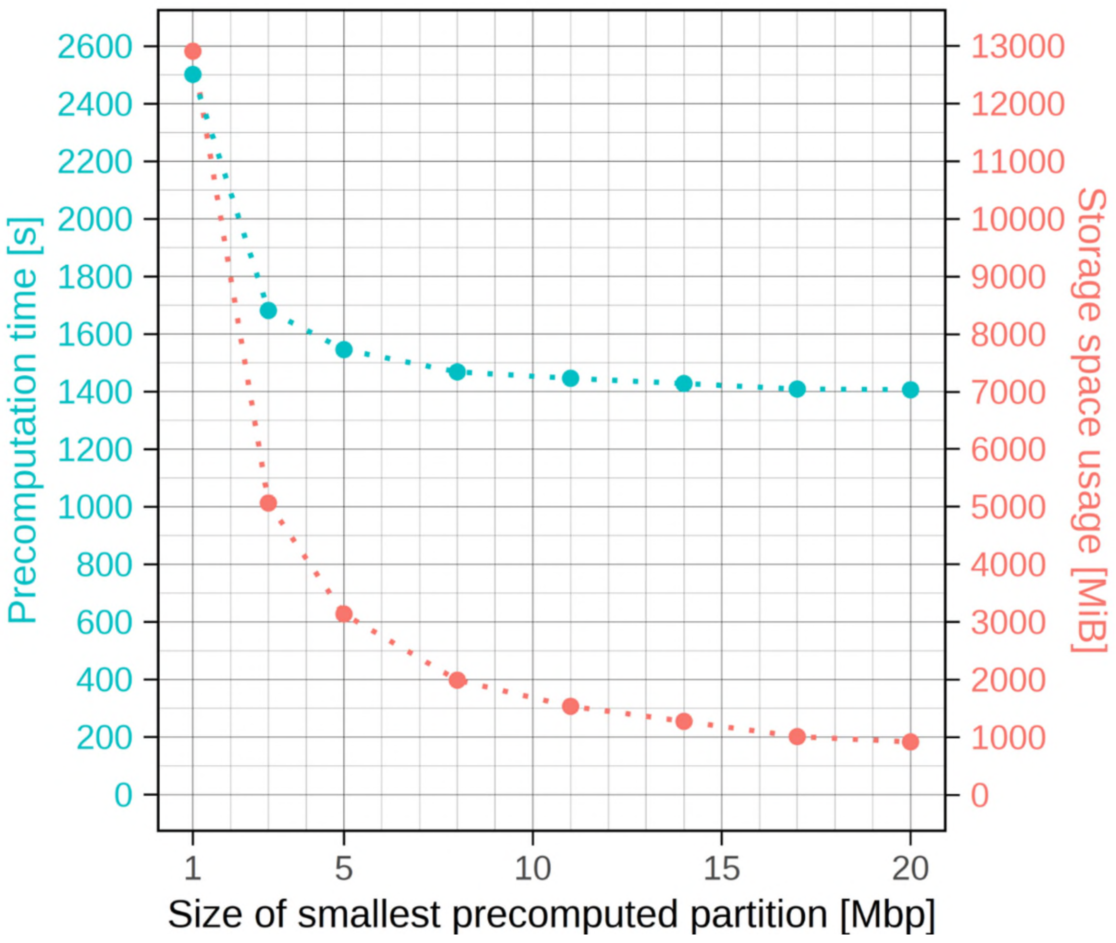


Figure B-1: Time and storage space costs of distance matrix precomputation as functions of the smallest partition size (human genome data set). *The 2548 resequenced human genome data were used for benchmarking. As with the tomato benchmarking (see* Figure 4-5*), successive precomputed partitions were generated by doubling the size of the smaller ones until a size larger than the human chromosome 1 (248.96 Mbp) was reached. The same general pattern is observed, with both metrics inversely proportional to partition size and precomputation time varying in a relatively narrow range due to close-to-constant computational cost of Tersect queries. See* **Table B-2** *for the numeric results.*

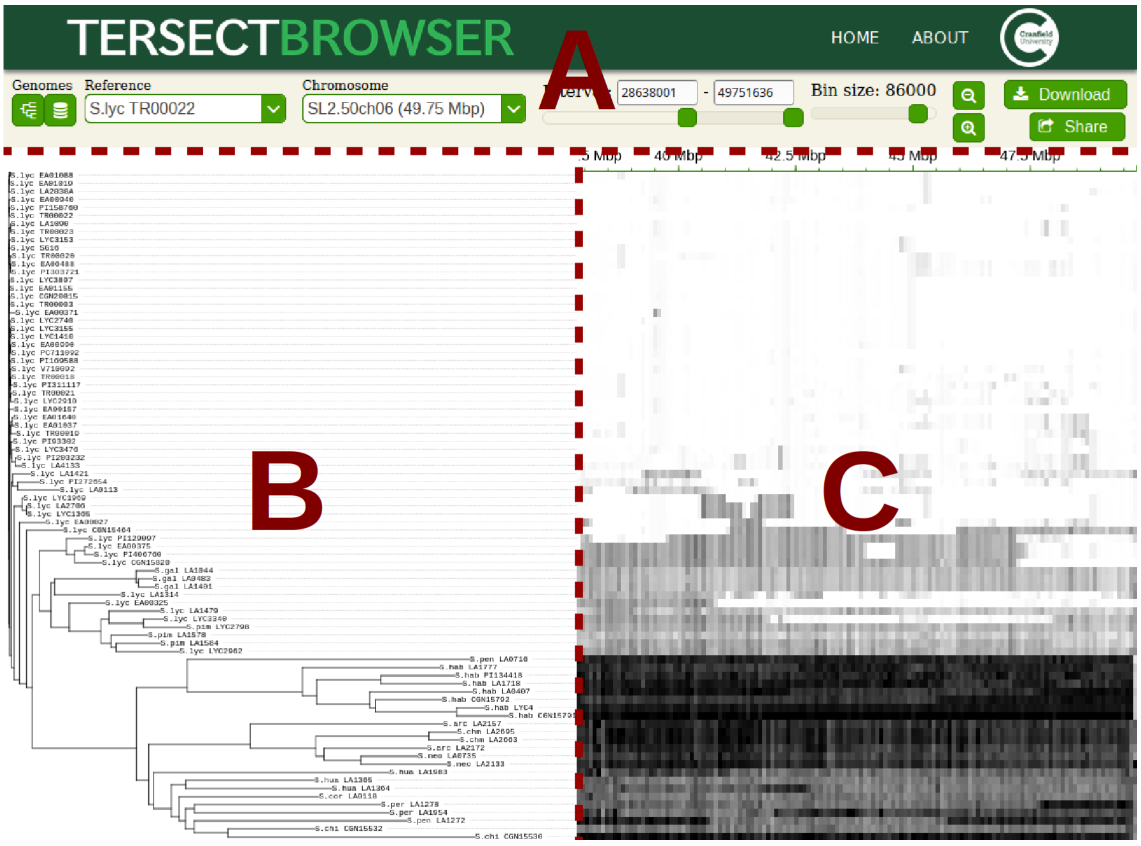
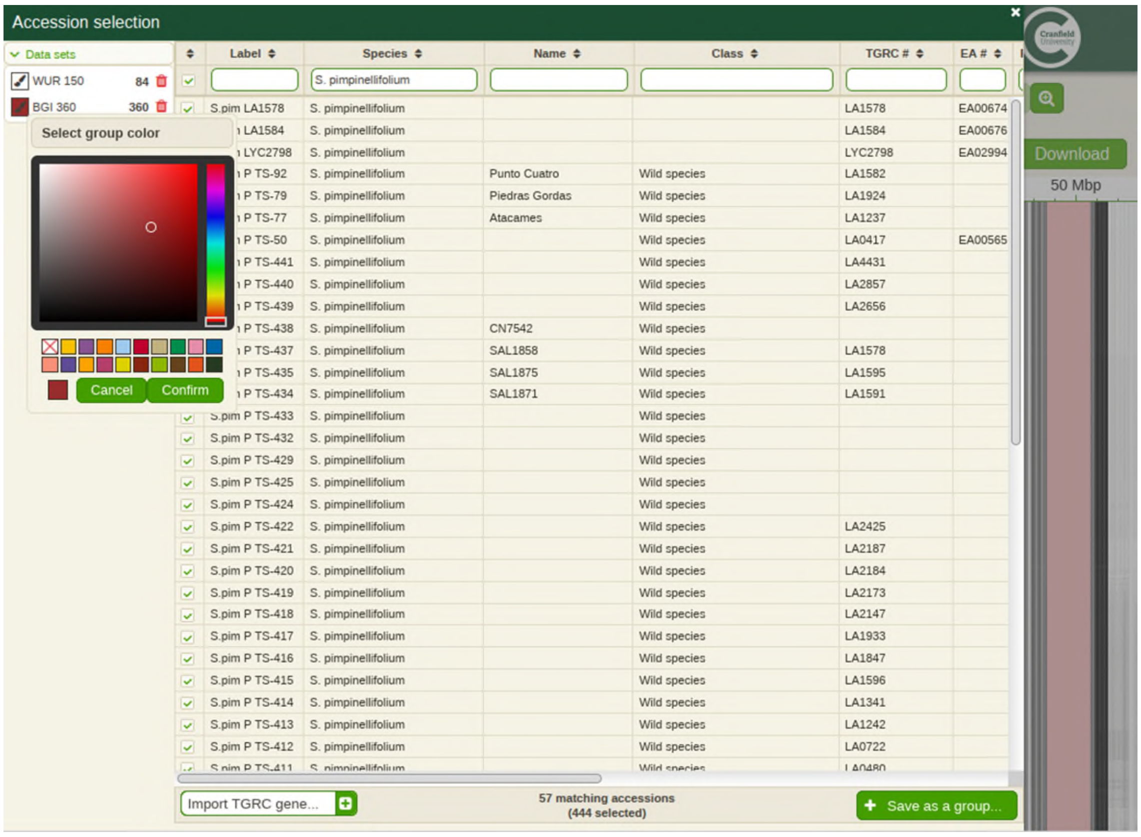


Figure B-2: Diagram of the primary Tersect Browser interface. *Part (A) of the interface contains controls which allow users to (left to right) select the phylogenetic tree display style, open the data set view, select a reference genome, select the chromosome, specify a chromosomal interval, select a bin size, zoom the plot in or out, download images, and share (export) views. The "Home" button at the top of the page returns the user to the index page, where they can select a different data set. Part (B) shows the phylogenetic tree for the current interval or simple labels, depending on the tree style setting. Individual genome labels can be clicked to select them as a reference or remove them from the current view. The tree (or labels) remains synchronized with the movements of the heatmap in the vertical axis and in terms of zoom level. Part (C) shows the heat map, representing the (binned) distances between each genome to the selected reference. The top of the heat map is bordered by a scale; mouse drags on the scale allow users to quickly select an interval. Individual bins can also be clicked to set interval borders, remove genomes from the view, or select them as the reference.*


Accession selection

Data sets

WUR 150 84

BGI 360 360

Select group color

Cancel Confirm

✓	Label	Species	Name	Class	TGRC #	EA #
✓		S. pimpinellifolium				
✓	S.pim LA1578	S. pimpinellifolium			LA1578	EA00674
	LA1584	S. pimpinellifolium			LA1584	EA00676
	LYC2798	S. pimpinellifolium			LYC2798	EA02994
	P TS-92	S. pimpinellifolium	Punto Cuatro	Wild species	LA1582	
	P TS-79	S. pimpinellifolium	Piedras Gordas	Wild species	LA1924	
	P TS-77	S. pimpinellifolium	Atacames	Wild species	LA1237	
	P TS-50	S. pimpinellifolium		Wild species	LA0417	EA00565
	P TS-441	S. pimpinellifolium		Wild species	LA4431	
	P TS-440	S. pimpinellifolium		Wild species	LA2857	
	P TS-439	S. pimpinellifolium		Wild species	LA2656	
	P TS-438	S. pimpinellifolium	CN7542	Wild species		
	P TS-437	S. pimpinellifolium	SAL1858	Wild species	LA1578	
	P TS-435	S. pimpinellifolium	SAL1875	Wild species	LA1595	
	P TS-434	S. pimpinellifolium	SAL1871	Wild species	LA1591	
✓	S.pim P TS-433	S. pimpinellifolium		Wild species		
✓	S.pim P TS-432	S. pimpinellifolium		Wild species		
✓	S.pim P TS-429	S. pimpinellifolium		Wild species		
✓	S.pim P TS-425	S. pimpinellifolium		Wild species		
✓	S.pim P TS-424	S. pimpinellifolium		Wild species		
✓	S.pim P TS-422	S. pimpinellifolium		Wild species	LA2425	
✓	S.pim P TS-421	S. pimpinellifolium		Wild species	LA2187	
✓	S.pim P TS-420	S. pimpinellifolium		Wild species	LA2184	
✓	S.pim P TS-419	S. pimpinellifolium		Wild species	LA2173	
✓	S.pim P TS-418	S. pimpinellifolium		Wild species	LA2147	
✓	S.pim P TS-417	S. pimpinellifolium		Wild species	LA1933	
✓	S.pim P TS-416	S. pimpinellifolium		Wild species	LA1847	
✓	S.pim P TS-415	S. pimpinellifolium		Wild species	LA1596	
✓	S.pim P TS-414	S. pimpinellifolium		Wild species	LA1341	
✓	S.pim P TS-413	S. pimpinellifolium		Wild species	LA1242	
✓	S.pim P TS-412	S. pimpinellifolium		Wild species	LA0722	
✓	S.pim P TS-411	S. pimpinellifolium		Wild species	LA0480	

Import TGRC gene...

57 matching accessions
(444 selected)

Save as a group...

Figure B-3: Diagram of the accession selection interface. *This overlay can be opened by pressing the data set button in the upper left corner of the main interface. The filterable data table contains a row for each genome in the data set, alongside columns provided during setup or added through plugins. The "Import TGRC gene…" control at the bottom is the entry point to one such (tomato-specific) plugin. Selections can be assigned to named groups, which can then be selected or unselected all at once, as well as marked with a specific group colour. The selection of genomes used for plot generation is updated when the accession selection interface closes.*

Table B-1: Distance matrix precomputation metrics for tomato (444 genomes) data.

Smallest partition size [Mbp]	Precomputation time [s]	Storage space usage [MiB]	Number of precomputed PHYLIP files
1	464.73	1486	1698
2	373.23	811	868
3	340.74	562	578
4	329.07	446	450
5	316.97	353	349

6	311.09	305	298
7	306.72	267	259
8	306.77	245	237
9	298.86	203	194
10	296.95	187	176
11	295.60	174	164
12	294.78	164	154
13	293.79	153	143
14	292.30	143	133
15	293.08	137	126
16	293.30	135	126
17	289.32	111	103
18	288.04	104	95
19	287.75	98	89
20	290.55	97	88

Table B-2: Distance matrix precomputation metrics for human (2548 genomes, restricted to chromosome 1) data.

Smallest partition size [Mbp]	Precomputation time [s]	Storage space usage [MiB]	Number of precomputed PHYLIP files
1	2502.01	12913	500
3	1681.96	5065	169
5	1545.87	3137	102
8	1468.47	1988	63
11	1446.28	1534	47
14	1428.10	1273	38
17	1409.05	1007	30
20	1406.40	916	27

Table B-3: Pairwise distance matrix generation times for tomato (444 genomes) data using different partition sizes and requested interval sizes.

	Median distance matrix generation time [s]					
Smallest partition size [Mbp]	1 Mbp intervals	3 Mbp intervals	5 Mbp intervals	10 Mbp intervals	20 Mbp intervals	50 Mbp intervals
1	0.728	0.771	0.845	0.948	0.992	1.074
2	0.808	0.862	0.933	0.993	1.036	1.110
3	0.858	0.910	1.061	1.159	1.163	1.191

4	0.856	1.036	1.112	1.282	1.166	1.358
5	0.865	1.261	1.138	1.358	1.367	1.544
6	0.796	1.349	1.321	1.561	1.542	1.628
7	0.806	1.488	1.526	1.586	1.648	1.700
8	0.761	1.470	1.670	1.714	1.789	1.796
9	0.756	1.605	1.750	1.871	1.853	1.780
10	0.720	1.604	1.950	1.708	1.812	2.036
11	0.736	1.546	2.029	1.982	1.874	2.195
12	0.740	1.560	2.170	2.078	2.202	2.403
13	0.745	1.579	2.320	2.257	2.395	2.370
14	0.740	1.535	2.219	2.257	2.340	2.432
15	0.737	1.534	2.277	2.598	2.497	2.761
16	0.723	1.427	2.319	2.726	2.529	2.735
17	0.730	1.465	2.243	2.703	2.690	2.584
18	0.727	1.480	2.289	2.834	2.705	2.652
19	0.721	1.454	2.296	3.189	2.752	3.071
20	0.700	1.467	2.307	3.398	3.139	3.351

Table B-4: Pairwise distance matrix generation times for human (2548 genomes) chromosome 1 data using different partition sizes and requested interval sizes.

	Median distance matrix generation time [s]					
Smallest partition size [Mbp]	**1 Mbp intervals**	**5 Mbp intervals**	**10 Mbp intervals**	**20 Mbp intervals**	**50 Mbp intervals**	**100 Mbp intervals**
1	18.87	22.00	22.67	26.91	27.37	28.82
3	18.59	22.38	24.41	27.36	30.45	33.52
5	15.70	24.06	25.07	28.05	30.96	35.17
8	16.32	31.38	30.62	36.65	36.59	41.62
11	15.74	36.65	31.58	38.44	43.94	42.48
14	16.01	38.98	43.33	46.08	47.23	44.07
17	15.41	40.04	44.80	47.28	49.22	48.85
20	15.83	40.44	56.71	51.50	64.36	70.78

Appendix C Tersect: a set theoretical utility for exploring sequence

Bioinformatics, 36(3), 2020, 934–935
doi: 10.1093/bioinformatics/btz634
Advance Access Publication Date: 8 August 2019
Applications Note

Genome analysis

Tersect: a set theoretical utility for exploring sequence variant data

Tomasz J. Kurowski and Fady Mohareb *

The Bioinformatics Group, School of Water, Energy and Environment, Cranfield University, Bedford MK43 0AL, UK

*To whom correspondence should be addressed.

Associate Editor: Bonnie Berger

Received on April 18, 2019; revised on July 5, 2019; editorial decision on August 4, 2019; accepted on August 6, 2019

Abstract

Summary: Comparing genomic features among a large panel of individuals across the same species is considered nowadays a core part of the bioinformatics analyses. This typically involves a series of complex theoretical expressions to compare, intersect, extract symmetric differences between individuals within a large set of genotypes. Several publically available tools are capable of performing such tasks; however, due to the sheer size of variants being queried, such tasks can be computationally expensive with a runtime ranging from few minutes up to several hours depending on the dataset size. This makes existing tools unsuitable for interactive data query or as part of genomic data visualization platforms such as genome browsers. Tersect is a lightweight, high-performance command-line utility which interprets and applies flexible set theoretical expressions to sets of sequence variant data. It can be used both for interactive data exploration and as part of a larger pipeline thanks to its highly optimized storage and indexing algorithms for variant data.

Availability and implementation: Tersect was implemented in C and released under the MIT license. Tersect is freely available at https://github.com/tomkurowski/tersect.

Contact: f.mohareb@cranfield.ac.uk

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Large-scale genome re-sequencing projects such as the 100 000 genome project (Turnbull *et al.*, 2018), the 1000 Genomes Project (Genomes Project Consortium *et al.*, 2015) or the 150 Tomato Genome Re-Sequencing Project (Tomato Genome Sequencing Consortium *et al.*, 2014) provide researchers with large-scale references for genetic variation. These can be compared with novel data to help identify causal variants and QTLs, delimit haplotype blocks and introgressions, or infer phylogenetic relationships (Gao *et al.*, 2019). All of those uses require means of filtering and comparing the variant contents between large phenotypic groups in order to identify concordant and discordant variants. These can be considered applications of set theoretical operations such as intersections or unions on sets of variants.

While multiple tools such as BEDOPS (Neph *et al.*, 2012), BCFtools (Danecek *et al.*, 2011) and BEDTools (Quinlan and Hall, 2010) offer the option to execute such operations, they are relatively inflexible in the complexity of possible queries and rely on parsing input files as they are executed, limiting their speed and responsiveness.

We hereby present Tersect, a tool which allows users to construct queries of any level of complexity by providing its own declarative query language and significantly speeds up their execution using specialized bitmap indices. Queries are interpreted, optimized and executed in a single step, either on entire genomes or on selected genomic regions, making the process extremely fast and responsive, ideal for an exploratory approach to investigating genome contents.

2 Tersect

Tersect imports VCF file data and uses bitmap indexing to encode binary information on the presence or absence of specific variants in each individual genome while building up a single unified database of alleles across all collected genomes.

The database is sorted and indexed by position and identity. When traversed in order, the stored list of variants is parallel to the per-genome bitmap indices, linking the two data structures. The index on variants and their positions allows for rapid identification of regions of interest by chromosome and position range, while the bitmap indices allow for highly efficient comparisons between genomes, leveraging bitwise operations to compare many sites at once. As any given genome contains only a relatively small subset of possible alleles, the bitmaps are sparse and easily compressed. Tersect uses a variant of the Word-Aligned Hybrid lossless compression method (Wu *et al.*, 2006) which allows logical operations without an explicit decompression step (See Supplementary Data S2 for detailed outline of Tersect storage architecture).

The variant data and indices are stored in a special index file which only needs to be generated once per collection of genomes and can be shared and used independently of the source data. Tersect uses a memory-mapped I/O approach to access index file contents, allowing for random access to regions of interest and limiting the memory footprint of queries.

A disadvantage of bitmap indexing, shared by Tersect, is the relative inefficiency of updating and adding data. This indexing approach is generally best suited to read-only applications and is often used in data warehousing. However, the stored data (allele identities and presence in genomes) do not frequently change over time and are generally added in batches (at least one genome at a time), mitigating such disadvantages. The Tersect index builder uses a highly efficient, priority-queue-based merge method, allowing for an index file to be rapidly re-created.

Tersect features a command parser which allows a user to enter set theoretical expressions operating on genomes (as sets) and variants (as set elements) and including set theoretical operations such as intersections, unions and symmetric differences (see Supplementary Data S1). These can be arranged into queries of arbitrary complexity, including deeply nested expressions, using a simple syntax. Rather than merely executing the parsed operations in sequence, Tersect builds an abstract syntax tree (AST) which represents the entered expression. The tree can then be optimized to simplify and speed up operations. If the user requests data from multiple genomic regions using the same command, the same AST is re-used for each.

3 Results and discussion

Tersect was benchmarked against three tools which offer similar functionalities: BCFtools, BEDTools and BEDOPS. It should be noted that, as they are designed to compare variant sets not only to each other but also to other types of data, the last two tools focus on *positional* overlap and intersection between features rather than variant identity. This means that overlapping but distinct variants, such as different alleles at multi-allelic sites or InDels which span across SNV sites, are considered to be intersecting. This leads to subtly different results delivered by the tested tools.

The data used for comparison were publicly available tomato genomes from two studies (Dinh *et al.*, 2018; Lin *et al.*, 2014; Tomato Genome Sequencing *et al.*, 2014), for a total of 444 re-sequenced genomes of tomato cultivars and closely related species.

Two important shared functionalities were tested: the identification of private variants, that is variants occurring only in a single specific genome out of a collection of genomes, and the intersection of a group of genomes to identify variants shared by each of them (also known as *concordant* variants). For the former test, subsets of the 444 genomes collection were used. For the latter, subsets of 56 *S.pimpinellifolium* genomes which contain large regions of shared variation distinct from the *S.lycopersicum* reference were used. Input data for each of the tools were converted into the most appropriate format (e.g. BED for BEDTools) and indexed (where appropriate) prior to the benchmarking. This also applies to Tersect, as the time taken to build an index, which needs to be done only once, was not included in the test runtimes. Index file generation for the largest genome collection (444 genomes) took 10 min (Supplementary Data S3— Fig. S1 and Table S3), which is fast enough to make Tersect the fastest tool even if indexing time were to be included in the benchmark.

It can be seen that Tersect performs from three to over a hundred times faster than BCFtools, which is the fastest of the other three applications (see Fig. 1 and Supplementary Fig. S2). The difference is more pronounced for larger inputs and this trend is likely to continue for datasets larger than those examined in this article. This presents a promising outlook for the scalability and future usability of Tersect as more genomes are re-sequenced every year and the volume of available data continues to rapidly increase. The runtime of all four tools follows a roughly linear relationship with the size of the input. The superior speed of Tersect stems from the highly

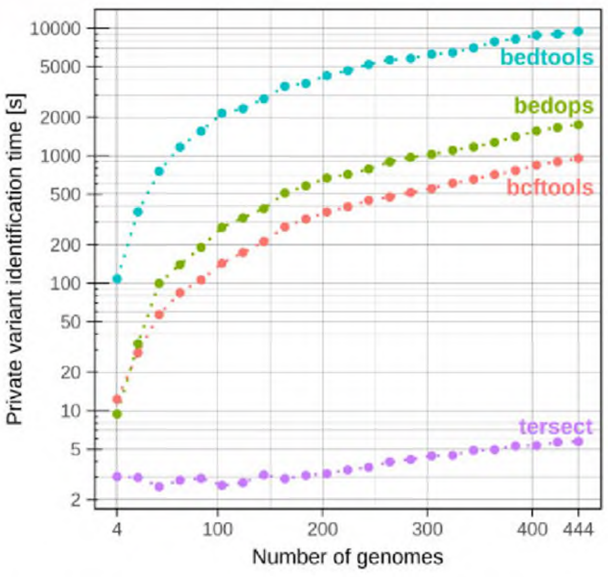


Fig. 1. Benchmarking results for the identification of variants private to a single genome out of subsets of 444 tomato genomes. All four applications show a linear relationship between the input size (number of genomes) and execution time; for Tersect this relationship is partially obscured by the relatively slow disk read/write operations which comprise a significant proportion of the runtime, especially for small input sizes. See Supplementary Table S1 for the numeric results

problem-specific optimization and indexing scheme rather than from improved algorithmic time complexity in the strict sense.

Tersect is the only among the evaluated tools capable of executing complex queries on large real-world datasets in a matter of seconds, making this the tool of choice to be used interactively, rather than as part of a batch processing pipeline. In combination with the flexible query syntax, this high performance offers new possibilities for real-time, exploratory use of the ever-growing volume of genomic data being produced today.

Funding

Tersect was developed as part of the BBSRC-funded project BB/L011611/1.

Conflict of Interest: none declared.

References

Danecek,P. *et al.* (2011) The variant call format and VCFtools. *Bioinformatics*, 27, 2156–2158.
Dinh,Q.D. *et al.* (2018) Exploring natural genetic variation in tomato sucrose synthases on the basis of increased kinetic properties. *PLoS One*, 13, e0206636.
Gao,L. *et al.* (2019) The tomato pan-genome uncovers new genes and a rare allele regulating fruit flavor. *Nat. Genet.*, **51**, 1044–1051.
Genomes Project Consortium *et al.* (2015) A global reference for human genetic variation. *Nature*, **526**, 68–74.
Lin,T. *et al.* (2014) Genomic analyses provide insights into the history of tomato breeding. *Nat. Genet.*, **46**, 1220–1226.
Neph,S. *et al.* (2012) BEDOPS: high-performance genomic feature operations. *Bioinformatics*, 28, 1919–1920.
Quinlan,A.R. and Hall,I.M. (2010) BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics*, **26**, 841–842.
Tomato Genome Sequencing Consortium *et al.* (2014) Exploring genetic variation in the tomato (Solanum section Lycopersicon) clade by whole-genome sequencing. *Plant J.*, **80**, 136–148.
Turnbull,C. *et al.* (2018) The 100 000 Genomes Project: bringing whole genome sequencing to the NHS. *BMJ*, **361**, k1687.
Wu,K. *et al.* (2006) Optimizing bitmap indices with efficient compression. *ACM Trans. Database Syst.*, **31**, 1–38.

Appendix D Mutagenesis of *Puccinia graminis f. sp. tritici* and Selection of Gain-of-Virulence Mutants

Mutagenesis of *Puccinia graminis* f. sp. *tritici* and Selection of Gain-of-Virulence Mutants

Ngonidzashe Kangara[1], *Tomasz J. Kurowski*[2], *Guru V. Radhakrishnan*[1], *Sreya Ghosh*[1], *Nicola M. Cook*[1], *Guotai Yu*[1], *Sanu Arora*[1], *Brian J. Steffenson*[3], *Melania Figueroa*[4], *Fady Mohareb*[2*], *Diane G. O. Saunders*[1*] and *Brande B. H. Wulff*[1*]

[1] Crop Genetics Department, John Innes Centre, Norwich, United Kingdom, [2] The Bioinformatics Group, Cranfield Soil and Agrifood Institute, Cranfield University, Bedford, United Kingdom, [3] Department of Plant Pathology, University of Minnesota, St. Paul, MN, United States, [4] Agriculture and Food, Commonwealth Scientific and Industrial Research Organisation, Canberra, NSW, Australia

Edited by:
Andreas Börner,
Leibniz Institute of Plant Genetics and Crop Plant Research (IPK), Germany

Reviewed by:
Christof Rampitsch,
Agriculture and Agri-Food Canada (AAFC), Canada
Walter Gassmann,
University of Missouri, United States

***Correspondence:**
Brande B. H. Wulff
brande.wulff@jic.ac.uk
Diane G. O. Saunders
diane.saunders@jic.ac.uk
Fady Mohareb
f.mohareb@cranfield.ac.uk

Specialty section:
This article was submitted to Plant Pathogen Interactions, a section of the journal Frontiers in Plant Science

Received: 06 June 2020
Accepted: 19 August 2020
Published: 16 September 2020

Citation:
Kangara N, Kurowski TJ, Radhakrishnan GV, Ghosh S, Cook NM, Yu G, Arora S, Steffenson BJ, Figueroa M, Mohareb F, Saunders DGO and Wulff BBH (2020) Mutagenesis of Puccinia graminis f. sp. tritici and Selection of Gain-of-Virulence Mutants. Front. Plant Sci. 11:570180. doi: 10.3389/fpls.2020.570180

Wheat stem rust caused by the fungus *Puccinia graminis* f. sp. *tritici* (*Pgt*), is regaining prominence due to the recent emergence of virulent isolates and epidemics in Africa, Europe and Central Asia. The development and deployment of wheat cultivars with multiple stem rust resistance (*Sr*) genes stacked together will provide durable resistance. However, certain disease resistance genes can suppress each other or fail in particular genetic backgrounds. Therefore, the function of each *Sr* gene must be confirmed after incorporation into an *Sr*-gene stack. This is difficult when using pathogen disease assays due to epistasis from recognition of multiple avirulence (Avr) effectors. Heterologous delivery of single *Avr* effectors can circumvent this limitation, but this strategy is currently limited by the paucity of cloned *Pgt Avrs*. To accelerate *Avr* gene cloning, we outline a procedure to develop a mutant population of *Pgt* spores and select for gain-of-virulence mutants. We used ethyl methanesulphonate (EMS) to mutagenize urediniospores and create a library of > 10,000 independent mutant isolates that were combined into 16 bulks of ~658 pustules each. We sequenced random mutants and determined the average mutation density to be 1 single nucleotide variant (SNV) per 258 kb. From this, we calculated that a minimum of three independently derived gain-of-virulence mutants is required to identify a given *Avr* gene. We inoculated the mutant library onto plants containing *Sr43*, *Sr44*, or *Sr45* and obtained 9, 4, and 14 mutants with virulence toward *Sr43*, *Sr44*, or *Sr45*, respectively. However, only mutants identified on *Sr43* and *Sr45* maintained their virulence when reinolculated onto the lines from which they were identified. We further characterized 8 mutants with virulence toward *Sr43.* These also maintained their virulence profile on the stem rust international differential set containing 20 *Sr* genes, indicating that they were most likely not accidental contaminants. In conclusion, our method allows selecting for virulent mutants toward targeted resistance (*R*) genes. The development of a mutant library from as little as 320 mg spores creates a resource that enables screening against several *R* genes without the need for multiple rounds of spore multiplication and mutagenesis.

Keywords: ***Puccinia graminis* f. sp. *tritici*, ethyl methanesulphonate mutagenesis, wheat, avirulence, effectors**

INTRODUCTION

Wheat stem rust is a destructive disease caused by the fungus *Puccinia graminis* f. sp. *tritici* (*Pgt*) that is resurging due to the evolution of virulent isolates that have overcome several stem rust resistance (*Sr*) genes (Pretorius et al., 2000; Olivera Firpo et al., 2017). Currently, 80% of the world's wheat cultivars are vulnerable to infection (Singh et al., 2008; Lewis et al., 2018) and worldwide yearly grain losses attributed to the disease are estimated at 6.2 million tonnes, equivalent to ~1% of the annual wheat yield (valued at USD 1.12 billion; (Beddow et al., 2013). However, this masks crop losses at local or regional levels, which can reach 40% or more (Schumann and Leonard, 2000; Saunders et al., 2019).

Many plant pests and pathogens, including *Pgt*, deliver effector molecules into host cells to facilitate successful parasitism. These effectors serve to suppress host defenses by inhibiting immune response-signaling following pathogen invasion. Effectors can also regulate host gene expression (Ahmed et al., 2018) and play a role in nutrient acquisition during rust infection (Sohn et al., 2000; Thara et al., 2003). Some effectors can be detected by the products of host *R* genes encoding either extracellular or intracellular immune receptors, leading to effector triggered immunity (Dodds and Rathjen, 2010; Lo Presti et al., 2015). Such effectors are termed avirulence (Avr) effectors and are often associated with a macroscopic hypersensitive cell death response.

The deployment of a single *R* gene in a disease hotspot leads to a large selection pressure which typically results in the rapid emergence of resistance-breaking strains of the pathogen (Johnson, 1961; Vale et al., 2001; Hovmøller and Justesen, 2007). However, the judicious stacking of multiple *R* genes would, from first principle, maximize the durability of resistance as there would be no selective advantage to a pathogen isolate which has overcome just a single *R* gene in the stack (Huang et al., 2006; Zhang et al., 2009; Fukuoka et al., 2015). To functionally test *R* genes in such stacks on a one-to-one basis first requires the identification of their corresponding Avr effectors. Certain *R* genes can interfere with each other (Hurni et al., 2014), do not work in certain backgrounds (Hiebert et al., 2020), or risk being silenced when delivered as transgenes (Anand et al., 2003; Li et al., 2005). Thus, it is essential to test the individual function of each *R* gene in a stack. As pathogens deliver multiple effectors, disease resistance cannot always be used to test the function of every gene in the stack (Vleeshouwers and Oliver, 2014; Wulff and Moscou, 2014; Chen et al., 2017; Salcedo et al., 2017). Therefore, Avr effectors can be used as probes to confirm the function of *R* genes in the absence of the pathogen (Upadhyaya et al., 2014; Vleeshouwers and Oliver, 2014; Bouton et al., 2018; Saur et al., 2019). This property makes cloned *Avr* genes useful tools in both fundamental and applied research. For example, effectors have been used extensively to study *R* gene structure/function relationships (Wulff et al., 2009; Maqbool et al., 2015; Ortiz et al., 2017; Seto et al., 2017) and to assist in the engineering of *R* genes with novel specificities (Harris et al., 2013; Segretin et al., 2014; Kim et al., 2016; De La Concepcion et al., 2019).

To date, several major dominant *Sr* genes have been cloned. Examples include *Sr13* (Zhang et al., 2017), *Sr21* (Chen et al., 2018), *Sr22* (Steuernagel et al., 2016), *Sr33* (Periyannan et al., 2013), *Sr35* (Saintenac et al., 2013), *Sr45* (Steuernagel et al., 2016), *Sr46* (Arora et al., 2019), *Sr50* (Mago et al., 2015) *Sr60* (Chen et al., 2019), and *SrTA1662* (Arora et al., 2019), with many more underway. These provide an excellent foundation for engineering multi-*Sr* gene stacks. In contrast, the cloning of their corresponding *Avr* genes has lagged behind. Most wheat rust (*Puccinia*) *Avr* gene cloning strategies have used genome sequencing followed by bioinformatics screens to shortlist candidate genes based on presence of a signal peptide, absence of a transmembrane domain, being cysteine-rich and size (< 300 amino acids), followed by subsequent transient heterologous expression assays to test function (Saunders et al., 2012; Upadhyaya et al., 2015; Anderson et al., 2016). Such strategies have been successful for other pathogens such as *Blumeria graminis* (Pedersen et al., 2012; Ahmed et al., 2015) *Phytophthora* sp. (Vleeshouwers et al., 2008), *Bremia lactucae* (Stassen et al., 2012), and *Magnaporthe oryzae* (Chen et al., 2013). However, for most rust fungi these approaches have been hampered by the lack of specific conserved sequence features for classifying effector genes and the absence of reproducible, genotype-independent and high-throughput transient assays for testing function. Moreover, conventional bi-parental genetics and positional cloning is impractical due to the difficulties in generating controlled sexual crosses and managing large numbers of segregating progeny (Johnson, 1954).

Mutagenesis followed by sequence-comparison of multiple independently-derived mutants presents an altogether different and promising approach, which has been successfully applied to identify genes from a wide array of organisms including worms, flies, and plants (Wang et al., 2010; Ashelford et al., 2011; Steuernagel et al., 2016; Addo-Quaye et al., 2017; Kawamura and Maruyama, 2019). *Pgt* work in the 1960s and 1970s showed that it is possible to mutagenize rust spores and select mutants that have lost the function of defined *Avr* genes (Teo and Baker, 1966; Luig, 1978). With the advent of genome sequencing, the first *Pgt* *Avr* effectors (*AvrSr35* and *AvrSr50*) were cloned by comparing induced or natural mutants to their wild type parents (Chen et al., 2017; Salcedo et al., 2017). The mutagenesis and screening methods described in these historical and recent studies, however, are lacking in detail and hence difficult to reproduce without significant investment in method optimization.

In the present study, we optimized procedures for generating mutant libraries of *Pgt* isolate UK-01, race TKTTF, using the chemical mutagen ethyl methanesulphonate (EMS) (**Figure 1**). We provide a detailed protocol on how to perform mutagenesis, identify the optimal concentration of EMS, select virulent mutants toward defined *Sr* genes, purify mutants, confirm their specificity and multiply purified spores for DNA extraction. We demonstrate this with screens against *Sr43*, *Sr44*, and *Sr45* in wheat. In order to accurately identify EMS-induced mutations, we generated a draft genome assembly of the wild type *Pgt* isolate UK-01 and conducted whole-genome resequencing of eight EMS-derived monopustule mutants to calculate mutation

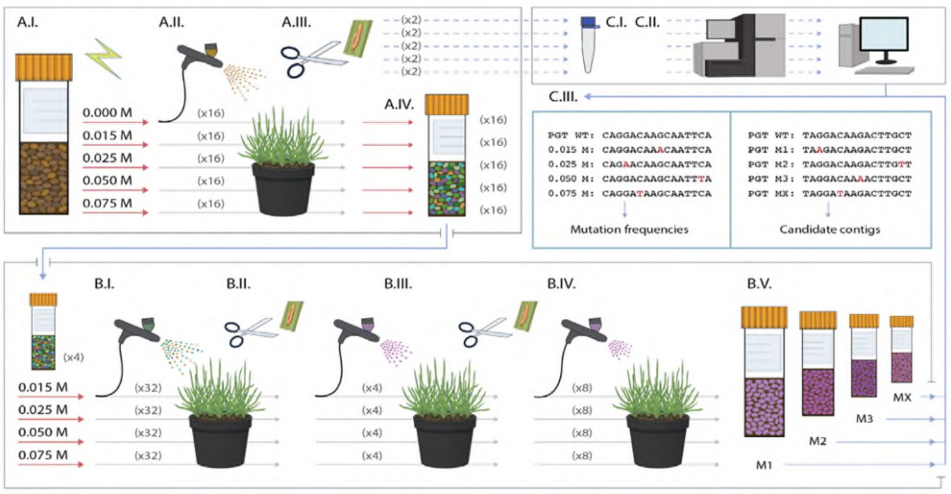


FIGURE 1 | Overview of the development, screening and analysis of *Pgt* ethyl methanesulphonate (EMS) mutants. Development of a *Pgt* mutant library starting with mutagenesis of the wild type **(A.I)**, inoculation of 16 pots each containing seven susceptible wheat plants per EMS treatment **(A.II)**, selection of two single pustule samples per treatment for spore increase, DNA extraction and sequencing **(A.III)**, followed by spore harvesting from each separate pot resulting in 16 mutant spore bulks per treatment **(A.IV)**. The 16 bulks were pooled to make four master bulks per EMS treatment **(B.I)** and each bulk was inoculated on eight pots containing seven plants of the *Sr* line of interest along with the *Pgt* wild type control. Virulent pustules were isolated **(B.II)**, reinoculated onto the *Sr* line they were identified from **(B.III)**, and pure, single pustules were isolated for bulking up for downstream experiments **(B.IV)** and sequencing **(B.V)**. Sample pustules from each EMS treatment were sequenced to determine single nucleotide variant (SNV) rates, while virulent mutants can be sequenced and analyzed to identify *Avr* candidate genes **(C.III)**.

density. Using these data, we make theoretical predictions on the number of independent virulence mutants that would require resequencing to identify causative mutations.

MATERIALS AND METHODS

Selection of *Aegilops tauschii* Lines Carrying Only *Sr45*, *Sr46*, or *SrTA1662*

The three genes, *Sr45*, *Sr46*, and *SrTA1662*, used to screen the mutant population were cloned from *Ae. tauschii* species (Steuernagel et al., 2016; Arora et al., 2019). To select the lines carrying only *Sr45*, *Sr46*, or *SrTA1662*, we inferred the presence of these genes by BLAST search (using a ≥ 99.8% identity and > 90% query coverage cut-off) in a panel of 151 *Ae. tauschii* accessions which had their NLR repertoires sequenced (Arora et al., 2019). In this way, we identified accessions carrying only one of the three genes – TOWWC0191 for *Sr45*, TOWWC0152 for *Sr46* and TOWWC0017 for *SrTA1662*. These accessions were then screened with *Pgt* UK-01 and all lines were found to be resistant, indicating that *Pgt* UK-01 is avirulent toward *Sr45*, *Sr46*, and *SrTA1662*. These accessions are available from the Germplasm Resource Unit at the John Innes Centre, UK.

Identifying Wheat-Alien *Sr* Introgression Lines Resistant to *Pgt* UK-01

Wild type *Pgt* UK-01 (race TKTTF) urediniospores were harvested 28 days after inoculation (see below) of cv. Vuka following single pustule isolation. The freshly harvested spores were placed in vials plugged with cotton wool and dried for four days using silica beads at room temperature and then stored at −80°C until use.

Seedlings of wheat or *Aegilops* lines containing the *Sr* genes *Sr22*, *Sr25*, *Sr33*, *Sr40*, *Sr43*, *Sr44*, *Sr45*, *Sr46*, *Sr51*, *Sr53*, *SrTA1662*, *Sr1644-1Sh*, *Sr2020*, recurrent parents, and the universal susceptible control wheat cvs Vuka and Chinese Spring (**Table 1**) were prepared by sowing two pots per line, each containing eight seeds. The seedlings were grown in a controlled environment room in a 23°C, 16 h light/15°C, 8 h dark cycle. Upon emergence of the coleoptile at approximately six days after germination the plants were treated with 0.2 g/L Maleic hydrazide solution (**Supplementary Table 1**) which was applied as a 50 ml pot drench. This stunted plant growth and enhanced leaf width.

For inoculations, urediniospores were taken from cold storage and "heat shocked" in a water bath at 45°C for 10 min. Each wheat or *Aegilops* line had three pots containing seven seedlings. The inoculum was applied to the leaves of the seedlings with an airbrush at a rate of 8 mg in 10 ml of 3 M^{TM} Novec 7000^{TM} Engineered Fluid (Novec) (**Supplementary Table 1**) for every eight pots (56 plants). Following inoculation, misting was conducted in the dark for 24 h at room temperature by placing the seedling pots in plastic bags and adding water to cover the bottom at a depth of about 5 cm. The open ends of the bags were sealed with cable ties. After 24 h, the pots were removed from the misting bags, re-bagged in breathable cellophane cross bottom bags fastened to the pots by elastic bands, and then transferred to the growth room with the previously described conditions. Phenotyping was conducted at 14 dpi using the Stakman *Pgt* phenotyping scale (Stakman et al., 1962).

Mutant Library Creation

We prepared sixteen pots of 12- to 14-day-old wheat seedlings of the universally susceptible cultivar Chinese Spring by sowing seven wheat seeds per 9 × 9 cm pot. The seedlings were grown in a controlled environment room as described above. For each EMS mutagenesis experiment, 200 mg of *Pgt* UK-01 urediniospores were "heat shocked" in a water bath (**Supplementary Table 1**) at 45°C for 10 min. Solutions of four EMS concentrations—0.015 M, 0.025 M, 0.05 M, 0.075 M plus a water control—were prepared in the fume hood with sterile H_2O containing 0.01% Tween 20 in 50 ml Greiner tubes (**Supplementary Table 1**). We added 40 mg of urediniospores to each tube and gently shook the suspensions by hand every 20 min over 1 h and 20 min at room temperature. The spore suspensions were collected separately by gravity filtration through a Whatman cellulose filter paper (**Supplementary Table 1**). The spores on each filter paper were washed with 500 ml of water with 0.01% Tween 20. After the water had drained, the urediniospores were washed off the filter paper using sterile water with 0.01% Tween 20 into 30 ml Nalgene™ Oakridge tubes (**Supplementary Table 1**). All EMS contaminated apparatus was immersed in EMS inactivation solution (0.1 M NaOH + 10% w/v $Na_2S_2O_3$) for at least 24 h in the fume hood.

We then proceeded to inoculate cv. Chinese Spring plants in a Class 2 biological safety cabinet using the mutagenized and control urediniospores suspended in water with 0.01% Tween 20. All 30 ml of the inoculum were used for each set of 16 pots containing a total of 112 plants (**Figure 1**). The procedures for misting and onward growth of the plants was as described above. At 12 dpi, three single pustules per treatment were randomly selected for bulking up and sequencing to determine mutation frequencies. After this, the cellophane bags (**Supplementary Table 1**) were bent sideways to allow the spores to collect in the bags, except for four randomly selected pots per treatment which were kept aside for conducting pustule counts at 14 dpi. At 35 dpi, the spores were collected from the cellophane bags. Spores were dried and stored as described above.

DNA Extraction and Whole-Genome Shotgun Sequencing

Sample pustules were harvested, diluted with Novec, and then inoculated onto four pots containing seedlings of cv. Vuka. The mono-pustule isolations were conducted twice. In the third cycle, 70–100 mg of urediniospores per sample were used for a CTAB DNA extraction. Urediniospores were mixed with 50 mg of glucose/sucrose mix and 50 mg sand and ground to a fine powder in a pestle and mortar. Prewarmed CTAB buffer (**Supplementary Table 1**) (50°C) was added to the ground mix followed by 10 µl proteinase K (20 mg/ml) and incubated at 50°C for 2 h with intermittent shaking. Another 10 µl proteinase K (20 mg/ml) was added followed by incubation at 50°C for 1 h. After this, 1 volume (V) chloroform:isoamyl alcohol (24:1) was added and the mix shaken vigorously and centrifuged at 5,000 g for 10 min. A total of 20 µl RNase (0.1 mg/ml) (**Supplementary Table 1**) was then added and incubated at room temperature for 1 h. A second chloroform: isoamyl extraction and centrifugation step was then conducted. DNA was precipitated from the aqueous phase using 1 V chilled (−20°C) isopropanol and left overnight in a −20°C freezer. DNA was pelleted by centrifugation at 16,000 g for 10 min. The DNA pellets were washed twice with 1 ml of 70% chilled ethanol and centrifuged at 16,000 g after each wash. The DNA pellets were dried and then suspended in 70 µl 1% TE buffer. DNA quantification and quality analysis was carried out using a Nanodrop™ spectrophotometer (ThermoFisher Scientific) and agarose gel electrophoresis and comparison to known concentrations of Lambda phage DNA. Illumina 350 bp insert library preparation and 150 bp paired-end whole genome shotgun sequencing was conducted at Novogene, Beijing or Genewiz.

MinION Library Preparation, Sequencing and Assembly

High molecular weight genomic DNA was obtained from *Pgt* UK-01 using a protocol developed by Nagar and Schwessinger (2018). All steps were followed with the following minor changes: Firstly, the mixture of the lysis buffer and ground spores was left to stand for 30 min before adding Proteinase K. Secondly, after recovering the aqueous phase from the first chloroform:isoamyl alcohol (24:1) step, 0.1 V of 3 M sodium acetate (pH 5.2) was added before DNA was precipitated using 1 V isopropanol, followed by pelleting at 16,000 g for 5 min, followed by washing the DNA pellet with 70% ethanol, drying and elution in 500 µl 10 mM Tris HCl. Thirdly, RNase digestion for 1 h followed by Proteinase K digestion was conducted according to the protocol before the second (final) chloroform: isoamyl alcohol (24:1) (**Supplementary Table 1**) step followed by DNA precipitation using 1 V isopropanol, pelleting, ethanol washing, DNA pellet drying as previously described and elution in 200 µl TE buffer. The DNA was then prepared for sequencing on the MinION sequencer (Oxford Nanopore Technologies, Oxford, UK). The DNA concentration was measured using a Qubit fluorometer (Thermo Fisher Scientific). A total of 305 ng of DNA was used as input for library preparation using the 1D Ligation Sequencing Kit (SQK-LSK109, Oxford Nanopore Technologies) carried out as per the manufacturer's instructions. The resulting library had a total mass of 67 ng and was sequenced on the MinION using a FLO-MIN106D flow cell (Oxford Nanopore Technologies) with 1,378 pores available for sequencing, following the manufacturer's instructions, for a total of 48 h.

Basecalling of the MinION reads was performed using Guppy v3.0.3 (https://community.nanoporetech.com) on CPU mode using the default parameters. Following basecalling, only reads longer than 1 kb were taken forward for genome assembly using Canu v1.8 (Koren et al., 2017) using the default parameters and an estimated genome size of 170 Mbp. Genome completeness was assessed using BUSCO v3. (Waterhouse et al., 2018) for the Basidiomycota fungal lineage on genome mode with *Ustilago maydis* as the reference species for gene prediction using Augustus v3.2.1 (Stanke and Morgenstern, 2005).

Polishing of the Nanopore Assembly

In order to polish the draft assembly, we generated PCR Illumina data by sequencing 250 bp paired-end Illumina reads with average insert size of 450 bp. We obtained three data sets containing a total of 19.7 Gbp which we aligned to the contigs of the Nanopore assembly using BWA-MEM 0.7.15 (Li and Durbin, 2009). The alignment results were sorted and duplicates marked using Picard 2.18 (Broad Institute, 2009) as the data were generated from PCR libraries. The alignment data were then used to improve the draft assembly using Pilon 1.23 (Walker et al., 2014) with the diploid setting and the default fix list: attempting to correct individual base errors, indel errors and local misassemblies, as well as fill gaps. This polishing workflow was repeated five times until the number of changes applied to the draft assembly plateaued. The resulting polished genome was re-assessed using BUSCO v3 (**Supplementary Table 2**) and used as the reference for downstream analysis.

Determining EMS Mutation Rates

Whole genome resequencing data from eight EMS-derived mutants consisting of 150 bp paired-end Illumina reads were aligned to the polished *Pgt* UK-01 genome assembly using BWA-MEM 0.7.15. As with other Illumina alignments, the results were sorted and duplicates marked using Picard 2.18. The GATK 3.8 (McKenna et al., 2010) IndelRealigner tool was then used to create the final BAM alignment files for each of the mutants. Variant calling was then carried out using two different tools, GATK 3.8 HaplotypeCaller and bcftools 1.6 within Samtools (Danecek et al., 2011), yielding two sets of results per sample. The results were filtered with a basic GATK hard filter, removing low-quality or low-depth calls using the filtering parameters Quality of Depth (QD) < 2 and Mapping Quality (MQ) < 40. The entire pipeline was carried out for each of the eight mutant samples separately.

Seven of the eight samples (B1, B2, C2, D1, D2, E1, E3) in both sets of variant calls were then passed through a filtering pipeline aimed at identifying EMS-induced SNVs. The eighth sample (A1) was the control not treated with EMS, therefore any variants called for it were assumed to represent heterozygous variants already present in the genome prior to applying the mutagen, and thus most likely shared with the other samples. Tersect 0.12 (Kurowski and Mohareb, 2020) was used to remove A1 variants from the other seven variant sets. Variants appearing in three or more of the samples were also removed, as identical mutations are unlikely to have been induced independently. Finally, the variants were filtered to remove all except SNVs known to be preferentially induced by the EMS mutagen, that is G to A and C to T transitions.

To eliminate likely false positives the two sets of variant calling results were then intersected, retaining only SNVs called by both GATK HaplotypeCaller and bcftools to create "high confidence" sets of SNVs for each mutant. These high confidence sets were then used to estimate the EMS mutation rate related to different concentrations of EMS used for each sample.

Calculating the Minimum Number of Independently Derived *Pgt* Mutants to Confidently Identify Candidate *Avrs*

For an unannotated genome to be used for comparative mutational genomics, every contig has to be treated as if it contains a potential gene candidate. If all the contigs in a given genome assembly are of length l, have the same GC content, and the canonical EMS mutations are distributed randomly across the assembly with a mutation density m, then, following the principles of binomial distribution, the probability (P) of a contig in such a genome assembly having mutations in n number of mutants by chance alone is:

$$P = (lm)^n$$

where, l = length of the gene (number of bases)

m = mutation density (number of SNVs per base of the genome)

n = number of mutants

If the number of contigs of length l in such a genome assembly is G, then the number of contigs that are likely to have mutations in all n mutants by chance alone, i.e., the number of false positive candidates, F, is:

$$F = G \times P$$

where, G = number of contigs in the assembly

P = probability of a contig in the assembly having mutations in all the surveyed mutants by chance alone.

Or, $F = G \times (lm)^n$

Given that the contigs of the UK-01 assembly have a uniform GC-content (**Supplementary Figure S1**), and assuming that the EMS SNPs are distributed evenly (Farrell et al., 2014; Shirasawa et al., 2016) the probability of a false positive candidate is therefore a function of the length of the contig.

Analyses of previously sequenced stem rust genomes have estimated that every 10 kb of the genome contains ~2 genes (Duplessis, 2011; Li et al., 2019). Thus, contigs of 10 kb or more can potentially complicate the search for candidate genes with mutations in all the mutants, as some of the mutants could have mutations in one gene within the contig while the other mutants could have mutations in the other gene. Therefore, to make our calculations, we chopped the contigs that were larger than 5 kb into smaller 5 kb contigs. The resultant chopped assembly contained contigs ranging from 1.001 - 5.999 kb. Since the equation for false positive number calculations only holds for contig lengths of roughly equal sizes, the UK-01 assembly contigs were divided according to their sizes into bins of 100 bp range, i.e., 1,000–1,100 bp, 1,100–1,200 bp, 1,200–1,300 bp and so on. The number of false positives within each bin was calculated using the formula for determining the value of F, where l = average length of contig in that bin, and G = number of contigs in that bin, and n = 1. In this way, the number of false positives was calculated for each bin and then these values were summed up to give the overall number of false positives from the whole assembly. If the number of false positives equalled a number greater than or equal to 1, then the value of *n* would be increased by 1 and the exercise repeated until the value of *F* was less than 1. The value of *n* for which F equals less than 1 is thus considered as the minimum number of mutants required to identify a candidate gene through comparison of re-sequenced mutant genomes. The calculation of the GC content of the contigs, division of the assembly into contig-length bins and iterative calculations of false positives were performed using custom code.

Screening for Gain-of-Virulence on *Sr44*, *Sr43*, and *Sr45*

We conducted the screen for gain-of-virulence pustules sequentially on each *Sr* line and per mutant batch and bulk. For each treatment, the sixteen vials of independent mutants were pooled into four bulks (a total of 16 bulks for all mutants across treatments) (**Figure 1**). Seedlings of introgression lines were prepared as eight pots per mutant bulk using the same Maleic hydrazide treatment as described above. For inoculations, 8 mg of spores for each *Pgt* bulk and the wild type control were weighed and then heat-shocked. Following this, the spores were suspended in Novec at a rate of 8 mg of spores in 10 ml Novec per eight pots (56 plants) using an airbrush. Misting, bagging and plant growth conditions were as described above. Gain-of-virulence pustules were recovered at 16 dpi, dried for four days using silica beads and stored at −80°C and then used to inoculate the lines on which they were identified. Pustules that maintained their phenotype were then isolated and bulked up on the lines they were identified on for further experiments.

Race Typing of *Sr43* Gain-of-Virulence Mutants

Virulent pustules were purified, multiplied and inoculation of the stem rust differential set was conducted as described above in the *Pgt* mutant screen for virulent candidates. Race phenotyping was carried out at 16 dpi.

RESULTS

Identification of *Sr* Genes Which Provide Effective Resistance to the *Pgt* Isolate UK-01

To select *Pgt* mutants with induced mutations in a defined *Avr* gene requires a wheat line in which the corresponding *Sr* gene has been genetically isolated in a background which is susceptible to the *Pgt* isolate chosen for mutagenesis. To identify such *Sr* gene stocks for the *Pgt* isolate UK-01 which was designated race type TKTTF according to the North American nomenclature (Lewis et al., 2018), we inoculated: (i) ten wheat lines with chromosome segments carrying defined *Sr* genes from wild wheats (wheat-alien introgression lines), (ii) the wheat recurrent parents used for *Sr* introgression, (iii) two *Ae. tauschii* accessions predicted to carry only *Sr46* or *SrTA1662* (Arora et al., 2019), one *Ae. sharonensis* accession carrying *Sr2020*, and (iv) the respective susceptible and resistant control wheat cultivars Vuka and Kavkaz/Federation4 (*Sr31*) (**Table 1**, **Supplementary Table 3**). The comparison of infection types between an introgression line and its recurrent parent provided a basis for selection of the *Sr* genes most effective against UK-01 and a reference for the expected infection type of gain-of-virulence *Pgt* mutants (i.e., an infection type close to or equal to that of the recurrent parent). The phenotypes were assessed using the 0 to 4 scoring scale (Stakman et al., 1962). Scores from 0 to 2+ were considered avirulent, while scores from 3 to 4 were considered virulent. In total, ten *Sr* introgression lines and three *Aegilops* accessions exhibited a resistance response of 2+ or less (**Figure 2**). The resistance conferred by *Sr44* fully suppressed pustule development resulting in a fleck (;) infection type (IT) whereas its recurrent parent cv. Angus had an IT of 4. The lines *Sr22* (IT 1+2), *Sr33* (IT 22+), *Sr40* (IT ;1−), *Sr43* (IT 1+), *Sr44* (IT ;), *Sr45* (IT 1−), *Sr53* (IT 1), *Sr2020*, and *SrTA1662* (IT 1−), were considered suitable for selecting virulent mutant pustules because of their low IT compared to lines not carrying the respective *Sr* gene (**Table 1**). In contrast, *Sr1644-1Sh* (IT 1−) and *Sr25* (IT ;1) were not shortlisted as their recurrent parents Zahir and Louise, respectively, both exhibited ITs of ;1− and 1− to

TABLE 1 | *Pgt* UK-01 infection types on *Sr* introgression lines and their recurrent parents, and *Aegilops* accessions predicted to carry single *Sr* genes.

Sr gene	*Sr* gene source	Infection type	Recurrent parent	Recurrent parent infection type	Selected for further analysis
Sr22	*Triticum monococcum*	1+2	Schomburgk	Not tested	Yes
Sr25	*Thinopyrum ponticum*	;1−	Louise	1−	No
Sr33	*Ae. tauschii*	22+	Chinese Spring	4	Yes
Sr43	*Th. ponticum*	1+2−	Chinese Spring	4	Yes
Sr45	*Ae. tauschii*	1−	Chinese Spring	4	Yes
Sr46	*Ae. tauschii*	1−	–	–	No
Sr51	*Ae. searsii*	1+2-	Chinese Spring	4	Yes
Sr53	*Ae. geniculata*	1	Chinese Spring	4	Yes
Sr40	*T. timopheevii* ssp. *armeniacum*	;1−	Westonia	4	Yes
Sr44	*Th. intermedium*	;	Angus	4	Yes
Sr1644-1Sh	*Ae. sharonensis*	1−	Zahir	;1−	No
SrTA1662	*Ae. tauschii*	1−	–	–	No
Sr2020	*Ae. sharonensis*	21	–	–	No
Sr31 (resistant control)	*Secale cereale*	1−	Kavkaz	–	No
Vuka (susceptible control)		32+	–	–	No

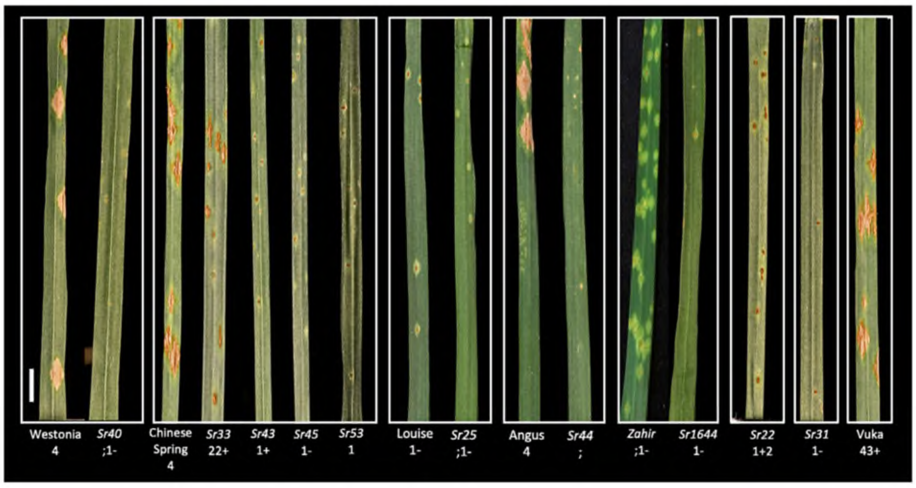


FIGURE 2 | Infection types (Stakman et al., 1962) of *Pgt* UK-01, race TKTTF against nine *Sr* genes. *Pgt* UK-01 infection types scored at 14 dpi on a panel of wheat recurrent parents and their wheat-alien introgression progeny carrying single defined *Sr* genes, plus resistant *Sr31* (cv. Kavkaz/Federation4) and susceptible (cv. Vuka) control lines. *Sr40* exhibited strong resistance as opposed to its recurrent parent Westonia, which was susceptible. *Sr33* exhibited moderate resistance, while *Sr43*, *Sr45*, and *Sr53* displayed strong resistance as opposed to their recurrent parent Chinese Spring, which was susceptible. Although *Sr25* showed strong resistance, its recurrent parent Louise also exhibited strong resistance. *Sr44* and its recurrent parent Angus displayed distinct resistant and susceptible ITs. Both *Sr1644-1Sh* and its recurrent parent Zahir were resistant. *Sr22* was resistant while its recurrent parent Schomburgk (not shown) was susceptible. *Pgt* UK-01 was avirulent on the resistant control *Sr31*. Introgression lines are grouped in the same box with their recurrent parents except for *Sr22* and *Sr31*. The scale bar represents 1 cm.

Pgt isolate UK-01 (**Figure 2**). In conclusion, we identified eleven *Sr* gene lines that exhibited sufficient resistance against the wild type *Pgt* isolate utilized to allow detection of gain-of-virulence mutant *Pgt* pustules.

Development of the Mutagenesis Procedure

We conducted two *Pgt* mutagenesis experiments by incubating urediniospores in 0 M, 0.015 M, 0.025 M, 0.05 M, and 0.075 M EMS and inoculating the mutagenized spores onto 12 to 14-day old seedlings of the susceptible host cv. Chinese Spring (**Figure 1A, I–IV**). To determine the effect of EMS mutagenesis on spore viability, we counted the number of pustules appearing on seven seedlings from each EMS treatment as well as the water control at 13 days post inoculation (dpi) (**Supplementary Tables 4** and **5**). Analysis of pustule count as a function of EMS concentration revealed a clear decline in pustule survival as the EMS concentration increased (**Figure 3**).

Five weeks after inoculation, the resultant spores were harvested separately for each pot of seven seedlings. In each of the two EMS experiments, 16 pots were harvested per EMS concentration making a total of 128 mutant harvests (2 EMS experiments × 4 EMS concentrations × 16 pots = 128 harvests). The 16 mutant collections from each EMS concentration were then combined into four bulks to give a final set of 32 bulks—16 from each experiment. Based on the pustule counts at 14 dpi, we estimate that we obtained a total count of 10,520 and 2,496 mutant pustules in mutagenesis experiments 1 and 2, respectively (**Supplementary Tables 4** and **5**). Following this, the mutant population consisting of ca. 13,016 pustules was screened against selected *Sr* genes to identify candidate *Avr* genes.

Sequencing and Assembly of the Wild Type UK-01 Reference Genome

We performed long-read sequencing of genomic DNA extracted from urediniospores of the wild type *Pgt* UK-01 isolate using the Oxford Nanopore MinION platform. We obtained 5.22 Gb of raw data, equivalent to an estimated 30-fold coverage of the ~170 Mbp dikaryotic *Pgt* genome (Li et al., 2019). The average read length was 2.58 kb, while the minimum and maximum read lengths were 0.08 and 51 kb, respectively (**Supplementary Table 2**). Only reads of a length greater than 1 kb were taken forward for assembly. We also generated 19.7 Gb of Illumina short read data from one 450 bp insert library with 250 bp paired-end reads. We first assembled the MinION reads with Canu and obtained an assembly size of 163.4 Mbp with 4,902 contigs and an N50 of 53.3 kbp (**Supplementary Table 2**). Assessment of genome completeness revealed that the Nanopore-only assembly contained over 93% of conserved fungal BUSCO genes (**Supplementary Table 2**). We then improved the assembly by incorporating the Illumina reads. Following polishing, the final assembly size was 164.3 Mbp. We therefore estimate that we assembled 96.6% of the ~170 Mbp dikaryotic *Pgt* genome.

EMS-Induced Mutation Frequency in the *Pgt* Genome

To determine the genome-wide mutation density, we collected eight random pustules from the first EMS population (one to two pustules per EMS treatment) for mono-pustule isolation (**Figure 1A, III**) and sequencing (**Figure 1C, I–III**),. We aligned the raw reads for each of the mutants to the final wild type genome assembly and called single nucleotide variants (SNVs). To reduce

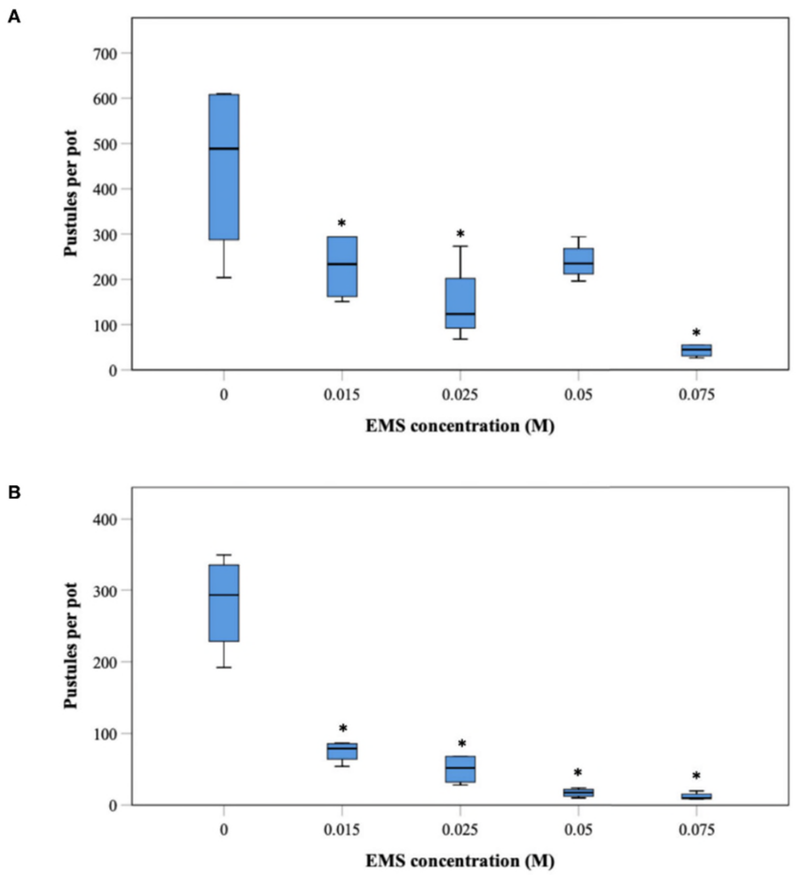


FIGURE 3 | Pustule survival as an effect of different concentrations of the chemical mutagen ethyl methanesulphonate. Spores of *Pgt* UK-01 were subjected to different concentrations of ethyl methanesulphonate (EMS) and inoculated onto leaves of 12-day-old wheat seedlings of the susceptible cultivar Chinese Spring. For each EMS treatment, 4 of 16 pots were picked at random. Each pot contained seven plants. Pustules were counted at 14 dpi. The results from two independent experiments are displayed in [**A**, (EMS experiment 1)] and [**B**, (EMS experiment 2)]. Asterisks indicate significant differences between treatments and null control according to Fishers LSD test (**A**, P=0.001; **B**, P=0.000).

the false discovery rate, we filtered the SNVs to (i) remove those SNVs which occurred in the nonmutated control, (ii) remove those SNVs that occurred in three or more mutants, and (iii) keep only the G:C to A:T transition mutations which typically account for around 98% of EMS-induced mutations (Lebkowski et al., 1986; Krasileva et al., 2017). The number of SNVs called by GATK (McKenna et al., 2010) per mutant genome ranged from 1,490 to 2,289, reflecting a G/C to A/T transition density per treatment of 1 per 103 kb for 0.015 M, 1 per 97 kb for 0.025 M, 1 per 87 kb for 0.05 M, and 1 per 80 kb for 0.075 M, which was similar in number to those called by SAMtools (Li et al., 2009) (**Table 2**). We then looked for overlap in the SNVs determined

TABLE 2 | Mutation frequency per ethyl methanesulphonate (EMS) treatment.

Sample	EMS level (M)	SNV count				Minimum no. of mutants required to identify causal mutations	
		GATK	Samtools mpileup	High confidence SNV GATK + Samtools shared	High + low confidence Nonredundant SNVs	Based on high confidence SNVs	Based on low confidence SNVs
B1	0.015	1,490	1,893	351	3,032	3	5
B2	0.015	1,678	2,358	524	3,512	3	5
C2	0.025	1,680	1,920	433	3,167	3	5
D1	0.05	1,861	2,307	701	3,467	3	5
D2	0.05	1,894	1,889	719	3,064	3	5
E1	0.075	2,289	2,599	1,033	3,855	4	5
E3	0.075	1,829	1,939	855	2,913	3	5

by GATK and Samtools to identify high confidence SNVs. This provided a high confidence SNV density of 1 per 404 kb for 0.015 M, 1 per 393 kb for 0.025 M, 1 per 239 kb for 0.05 M, and 1 per 182 kb for 0.075 M. The SNV densities were positively correlated with the EMS concentration ($r^2 = 0.9$, in the case of the high confidence SNV densities) (**Supplementary Table 6**). The density of total, nonreduntant SNVs per treatment (i.e., high and low confidence SNVs identified by either program) ranged from 1 SNV per 56 kb to 1 SNV per 44 kb.

We calculated the minimum number of independently derived gain-of-virulence mutants required for sequence comparison with each other in order to identify the causative mutations and clone an *Avr* gene with confidence. We considered (i) a 5 kb window for contigs larger than 5 kb given that there is on average 1 gene per 5 kb in *Pgt* (Duplessis, 2011; Li et al., 2019), (ii) an observed mean GC content of 42.9% (with a standard deviation of 3.8%) per contig/window in *Pgt* UK-01 (**Supplementary Figure S1**), and (iii) the density of the SNV overlap determined by GATK and Samtools (**Table 2**). The total number of false positives (i.e., 1–5 kb windows with mutations in all compared mutants by chance alone) was then calculated for analysis using from 1 up to 10 gain-of-virulence mutants according to the principle of a binomial distribution. As a result, we determined the minimum number of independently derived *Pgt* mutants required to reduce the probability of calling false positives to zero was three across all EMS treatments (**Table 2**). This number was five when identifying causal gene *Avr* mutations using low confidence SNVs (**Table 2**). The higher number based on the total SNVs across both GATK and Samtools (i.e., nonreduntant SNVs) sets the minimum target for identification of gain-of-virulence mutants during *Pgt* mutant screens.

EMS Mutagenesis of *Pgt* UK-01 Allows Recovery of Mutants With Stable Virulence to *Sr43* and *Sr45*, but Not *Sr44*

We screened the *Pgt* mutant population from Experiment 1 comprising 10,520 independent pustules on the wheat-*Th. intermedium Sr44* introgression line. We identified four weakly virulent mutant pustules which displayed an IT of 1+ at 21 dpi (**Figure 4A**, mutants M-1 and M-2; **Supplementary Table 7**). However, the IT of these mutants reverted to wild type (;) after multiplication and re-inoculation onto *Sr44*. We then proceeded to screen both mutant libraries, i.e., 13,016 *Pgt* mutants, on the wheat-*Th. ponticum Sr43* line. Here we obtained nine gain-of-virulence mutants at 16 dpi (**Figure 4B**). Each virulent mutant had an IT of at least 3 or 4 while the IT of the wild type *Pgt* isolate UK-01 on *Sr43* was 1+ (**Figure 4B**, **Supplementary Table 7**). The mutant ITs became distinguishable from that of the wild type at 12 dpi with the pustules reaching full size at 16 dpi. Finally, we screened the 13,016 mutant pustules against *Sr45* (either the wheat-*Ae. tauschii Sr45* stock, or the *Ae. tauschii* lines predicted to only contain *Sr45*) and identified fourteen virulent mutants with ITs ranging from 2 to 4 (**Figure 4C**, **Supplementary Table 7**). The *avrSr43* and *avrSr45* mutants maintained their virulence after multiplication and re-inoculation onto the *Sr43* and *Sr45* stocks, respectively. When inoculated onto the cv. Chinese Spring

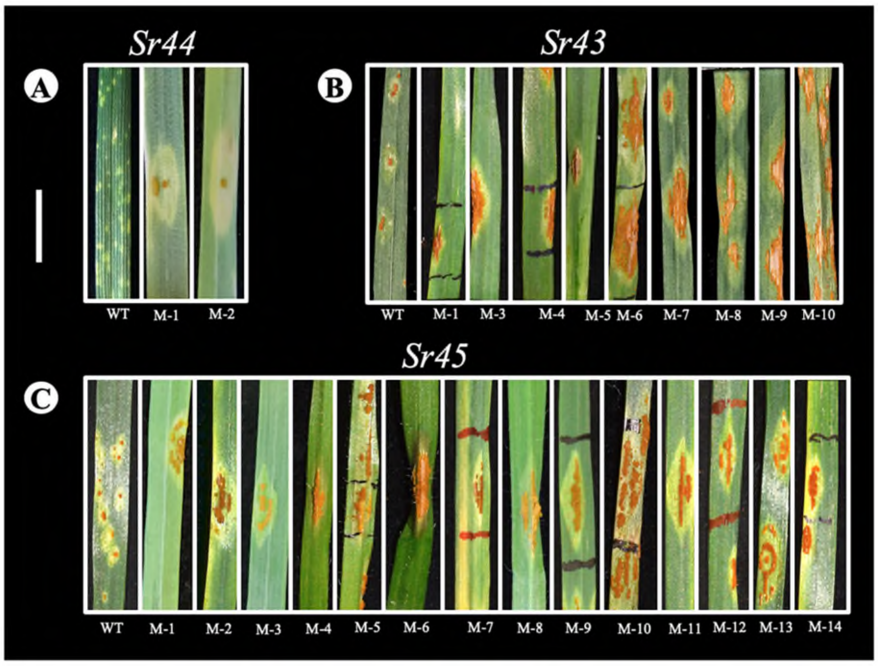


FIGURE 4 | *Pgt* UK-01 EMS mutants with virulence to *Sr44* **[A]**, *Sr43* **[B]** and *Sr45* **[C]**. Virulent *Pgt* UK-01 mutant pustules obtained on Sr44, Sr43, and Sr45 were visually assessed at 16 dpi. Each mutant within a series was obtained from different mutant bulks and they were only selected if the pustule size was significantly larger than that of the wild type Pgt UK-01 control. Pustule infection type on the controls indicate resistance, *Sr44* IT ; **[A]**, *Sr43* IT +1 **[B]** and *Sr45* 1- **[C]**. Scale bar represents 1 cm.

recurrent parent the virulent mutants did not show any change in pustule color or morphology compared to that of the wild type, *Pgt* isolate UK-01. However, some of the mutants grew more slowly and produced fewer spores than the wild type likely due to background mutations affecting fitness.

Pgt Mutants Virulent on *Sr43* Maintain the Same Virulence Profile as the Wild Type on the International Standard Differential Set

To test virulence specificity, we selected eight mutants with a gain-of-virulence on *Sr43* and inoculated them (alongside the *Pgt* UK-01 wild type control) onto the stem rust international differential set consisting of twenty lines with single defined *Sr* genes (Stakman et al., 1962). Infection types were observed between 14 and 16 dpi (**Supplementary Table 8**). Wild type *Pgt* UK-01 had low infection types on three (*Sr11*, *Sr24*, and *Sr31*) of the 20 lines, and the ITs of the nine mutants were indistinguishable from the wild type. Thus the mutants had the same race designation of TKTTF as defined by their virulence profile on the differential set. The mutants showed no additional virulence compared to the wild type on the 20 lines, apart from differing in *Sr43*.

DISCUSSION

The rapid advance of next-generation sequencing and computational technologies has enabled the cloning of several major dominant *Sr* genes and many more are in the process of being cloned (Keller et al., 2018; Periyannan, 2018). This has made the engineering of multi-*Sr* gene stacks for durable stem rust resistance a possibility. In contrast, the cloning of corresponding *Pgt Avr* genes has lagged behind as they do not have defined sequence signatures (unlike the *R* genes) making sequence analysis to identify *Avrs* difficult. To circumvent this and accelerate *Pgt Avr* gene cloning, we have outlined a procedure to generate mutant populations of *Pgt* spores and select for multiple gain-of-virulence mutants toward targeted *Sr* genes. Our work builds on historical studies from the '60s and '70s where chemical mutagens were used to obtain rust isolates virulent against *R* genes in oat and wheat (Teo and Baker, 1966; Luig, 1978). We innovated the development of bulks of mutant *Pgt* spores, creating a resource which enables screening against several *R* genes without the need for multiple rounds of spore bulking and mutagenesis.

Identification of Effective *Sr* Gene Lines to Clone Corresponding *Avr* Genes

We tested 13 *Sr* genes for their efficacy in controlling wild type *Pgt* UK-01 in order to identify *Sr* gene targets suitable for *Pgt* gain-of-virulence screens. The ITs of the *Pgt* UK-01 isolate on the 13 lines predicted to carry single *Sr* genes of interest that met the cut-off criteria ranged from necrotic fleck (;) to intermediate resistance (2+) (Stakman et al., 1962). However, two of these 13 *Sr* genes, *Sr1644-1Sh* and *Sr25*, were not shortlisted because their recurrent parents Zahir and Louise had an IT of 1+2− indicating the presence of background resistance. Additional resistance in the recurrent parents would significantly reduce the probability of obtaining gain-of-virulence *Pgt* EMS mutants because it would require the rare simultaneous loss of more than one *Avr*.

Some of the shortlisted introgression lines carry *Sr* genes which have been previously cloned, including *Sr22*, *Sr45* (Steuernagel et al., 2016), *Sr33* (Periyannan et al., 2013), and *Sr46* (Arora et al., 2019). These present an opportunity for faster functional validation of the genetic interaction of corresponding *Sr-Avr* pairs by transient coexpression in wheat protoplasts (Saur et al., 2019), by virus-induced overexpression in leaves of seedlings (Bouton et al., 2018) or in a heterologous system like *Nicotiana benthamiana* (Salcedo et al., 2017). Furthermore, with the exception of *Sr22*, these *Sr* genes have not yet been deployed and provide resistance to races that are predominant in areas where *Pgt* is prevalent. Examples include: *Sr33* (Periyannan et al., 2013), *Sr43* (Yu et al., 2017) and *Sr45* (Steuernagel et al., 2016). In summary, 11 *Sr* gene lines which displayed a stronger resistance than their respective recurrent parent background were identified as suitable for selecting gain-of-virulence mutants.

Creation of *Pgt* Mutant Libraries for Screening on Target *Sr* Introgression Lines

We present a robust and detailed step-by-step method for creating mutant *Pgt* libraries which can be returned to again and again as a resource for multiple mutant screens. We introduced a step where we inoculated mutagenized spores onto a susceptible cultivar, Chinese Spring, for propagating large amounts of mutant spores. Bagging of individual pots containing inoculated Chinese Spring plants prevented cross contamination and the spores from each pot were harvested separately. Creating a population of *Pgt* mutants allowed us to screen multiple *Sr* lines sequentially in a confined containment space, reduced the need for multiple rounds of mutagenesis and made extended use of the limited starting material of 400 mg of wild type urediniospores (including the control) in both experiments.

In previous *Pgt* EMS mutagenesis studies, freshly mutagenized spores were directly inoculated onto resistant plants carrying defined *Sr* genes (Teo and Baker, 1966; Luig, 1978; Salcedo et al., 2017). This requires a large amount of initial inoculum and generation of a new mutant library at the start of any new screen. In contrast, a mutant urediniospore library can be aliquoted and stored for long periods of time in a −80°C freezer or liquid nitrogen. We harvested mutant spores separately from each of the 16 pots per EMS treatment and pooled them into batches of four per EMS treatment (to create 16 bulks per experiment). This strategy allowed us to select independently derived gain-of-virulence mutants on either *Sr43* or *Sr45* as only one virulent pustule was selected per mutant bulk. This is an important consideration as multiple independent mutants are required to identify causative mutations through identification of a gene which is mutated in all or most individuals (Sánchez-Martín et al., 2016).

During mutant library creation, we observed a decline in pustule numbers with increasing EMS concentration (**Figure 3**) concomitant with a slight increase in high confidence SNV density from 1 SNV per 484 kb at the lowest EMS concentration to 1 SNV per 165 kb at the highest concentration (**Table 2**). This suggests a potential increase in deleterious mutations with increasing G/C to A/T mutation densities. The average mutation rate of 1 SNV per 258 kb for high confidence SNVs gave us the assurance that the mutagenesis had worked and screening for candidate loss of effector function mutants would be possible. In the previous *Pgt* mutagenesis screen which led to the cloning of the *AvrSr35* effector, the mutation density ranged from 1 SNV per 2,152 kb to 1 SNV per 10 kb with the average mutation rate being 1 SNV per 77 kb (Salcedo et al., 2017). Our mutation density for the nonredundant SNVs was slightly higher, but fell within the previously observed range between 1 SNV per 44 kb and 1 SNV per 54 kb. This 1.3-fold increase in the nonredundant SNV mutation density that we observed between the lowest and highest EMS concentration resulted in a 5 to 8-fold decrease in pustule number. However, there was no discernable difference in the frequency of obtaining gain-of-virulence mutants between the different EMS concentrations (expressed as number of mutants obtained per 10^6 SNVs screened, **Supplementary Tables 4** and **5**). Therefore, the lower EMS concentrations used here are ideal for use in *Pgt* forward genetic screens because these concentrations maximize spore survival against only a minor decrease in mutation density.

We proceeded to screen the mutants on *Sr43* and *Sr45*. From these screens, we identified multiple gain- of-virulence mutants on introgression lines carrying *Sr43* (11 mutants) and *Sr45* (14 mutants). This suggests that the *Pgt* UK-01 isolate is likely heterozygous for the corresponding effectors. In contrast, we did not obtain any *bona fide* mutants for *Sr44*, suggesting that *Pgt* UK-01 may contain two functional copies of *AvrSr44* or that the introgressed *Th. intermedium* segment carries more than one *Sr* gene effective toward *Pgt* UK-01. Based on the average mutation rates within the *Pgt* genome, we calculated that at least three independently derived mutants would be required to clone a candidate *Avr* gene when considering high confidence SNVs, while five mutants are the minimum number when considering the high and low confidence SNVs. This assumes a "gene-for-gene" *Avr-Sr* interaction (Flor, 1955) whereby the change in *Pgt* phenotype in all the mutant gain-of-virulence individuals identified per *Sr* line can be attributed to a mutation in the corresponding *Avr* (rather than a second site suppressor of an *Avr* gene function) (Melania et al., 2020). The phenotypes of the gain-of-virulence mutants had no distinct differences from wild type *Pgt* UK-01 on Chinese Spring, the recurrent parent for both the *Sr43* and *Sr45* introgression lines. Teo and Baker (1966) observed a small number of EMS mutants of *P. graminis* f. sp. *avenae* on resistant lines, which developed teliospores early, while (Luig, 1978) obtained some *Pgt* pustules that were yellow or orange. Such phenotypes were not reported in *Pgt* mutants by Salcedo et al. (2017), nor in the present study.

Race typing of the *Sr43* gain-of-virulence mutants on the stem rust international differential set containing lines carrying 20 single *Sr* genes confirmed that the mutants retained the same TKTTF race designation of the wild type. This indicated that the gain-of-virulence observed was likely due to loss of a single effector gene (*AvrSr43*) rather than contamination from other isolates – as *Pgt* UK-01 was the only isolate used in the area where our experiments were conducted and *Pgt* UK-01 is the only isolate present in the UK (Lewis et al., 2018) – or gross genetic aberrations taking out multiple *Avr* genes. In this study, the general conservation in virulence profiles between the *Pgt* wild type isolate and mutants also provides further support to these being genuine derived mutants rather than potential contaminants.

We observed (but did not quantify) that some virulent mutants did not sporulate well during multiplication of pure spores for additional experiments. This did not correlate with the EMS dosage used to generate the mutants. In the example of mutants identified on the line with *Sr43*, mutant M-7 was derived from a bulk developed using 0.075 M EMS and was more prolific in spore production than M-3, derived from 0.025 M EMS. Thus, M-3 required two or three rounds of multiplication to produce the equivalent amount of spores. This is likely due to second-site mutations in M-3 which reduced fitness.

In summary, we have developed a detailed protocol for obtaining *Pgt* gain-of-virulence mutants. We demonstrate this by obtaining gain-of-virulence mutants toward either *Sr43* or *Sr45*. This work, coupled with the ability to obtain and analyze whole genome sequence data of multiple virulent mutants (Salcedo et al., 2017) presents an opportunity to clone the *Avrs* corresponding to *Sr43* and *Sr45*. Beyond this work, this method can be applied to obtain mutants in other *Avr* genes of *Pgt* as well as for other fungal pathogens with haploid genomes that can be propagated using asexual spores. With time, we expect the near complete pan-genome complement of *Pgt Avrs* to be cloned. This will facilitate large-scale sequence-based monitoring of *Pgt* virulence in the field and predictive breeding, including the creation and deployment of efficient stacks to control this important pathogen.

DATA AVAILABILITY STATEMENT

The datasets presented in this study can be found in online repositories. The names of the repository/repositories and accession number(s) can be found below: https://www.ebi.ac.uk/ena, PRJEB38623.

AUTHOR CONTRIBUTIONS

NK and GY conducted the *Pgt* phenotyping on the *Sr* panel. NK reduced to practice the *Pgt* mutagenesis process originally developed by MF and BS, conducted the *Pgt* mutagenesis, mutant screens, data collection, analysis, DNA extractions and drafted the manuscript. NC prepared libraries and setup the sequencing of the *Pgt* DNA libraries on the MinION. GR conducted processing, analysis and assembly of the wildtype Nanopore sequencing data. TK and FM conducted polishing of the Nanopore assembly and the EMS mutation frequency analysis. SG calculated the minimum number of mutants required to clone a high confidence candidate *Avr*. SA analyzed *Ae. tauschii* accessions to identify lines carrying only *SrTA1662*, *Sr45* or *Sr46*. MF and BS developed the original mutagenesis procedure adapted in this study and reviewed the manuscript. BW conceived the study. BW and DS designed and supervised the study, and revised the manuscript with input from GR, SG, NC, SA, TK, MF, and BS. All authors contributed to the article and approved the submitted version.

FUNDING

We gratefully acknowledge funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 674964, the John Innes Centre Science For Africa Initiative (JIC-SFA) and the Biotechnology and Biological Sciences Research Council (BBSRC) cross-institute strategic programmes Designing Future Wheat (BB/P016855/1) and Plant Health (BB/P012574/1) through a Discovery Fellowship to GR (BB/S011005/1). SG was supported by a Monsanto's Beachell-Borlaug International Scholars' Program fellowship and 2Blades Foundation.

ACKNOWLEDGMENTS

We would like to thank: Jens Maintz for *Pgt* UK-01 purification and spore *Pgt* mutant harvesting, Yajuan Yue and Macarena Forner-Martínez for bulking up of seed for lines used in the experiments, Rebecca Doherty for harvesting of *Pgt* mutant spores as well as watering of inoculated plants in the CERs, JIC Horticultural Services for sowing the seedlings used in experiments and maintenance of plants during seed bulking in the glasshouses, the JIC Germplasm Resource Unit for the maintenance and disbursing of seed, the NBI Computing Infrastructure for Science (CiS) group for HPC maintenance, Cristobal Uauy and Chris Darby for help with funding acquisition, Tobin Florio (www.flozbox-science.com) for the artwork in **Figure 1**, Catherine Gardener for proof reading the manuscript, Simon Berry and Paul Fenwick from Limagrain for providing *Pgt* UK-01, Tom Fetch for supplying the *Sr* gene Core Differential Set, Evans Lagudah for supplying the *Sr22*, *Sr33* and *Sr45* lines, Ian Dundas for supplying the *Sr40* and *Sr44* lines, Steven Xu for supplying the *Sr43* line, Michael Pumphrey for supplying the *Sr51* and *Sr53* lines, Eitan Millet for supplying the *Sr1644-1Sh* line, the Wheat Genetic and Genomic Resource Center for the *Ae. tauschii Sr1662* line and Brian Steffenson for supplying the *Ae. sharonensis* accession 2020.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fpls.2020.570180/full#supplementary-material

REFERENCES

Addo-Quaye, C., Buescher, E., Best, N., Chaikam, V., Baxter, I., and Dilkes, B. P. (2017). Forward Genetics by Sequencing EMS Variation-Induced Inbred Lines. *G3: Genes Genomes Genet.* 7 (2), 413–425. doi: 10.1534/g3.116.029660

Ahmed, A. A., Pedersen, C., Schultz-Larsen, T., Mark, K., Jørgensen, H. JørgenL., and Thordal-Christensen, H. (2015). The Barley Powdery Mildew Candidate Secreted Effector Protein CSEP0105 Inhibits the Chaperone Activity of a Small Heat Shock Protein. *Plant Physiol.* 168 (1), 321–333. doi: 10.1104/pp.15.00278

Ahmed, Md B., dos Santos, K. C. Gonçalves, Sanchez, I. B., Petre, B., Lorrain, Cécile, Plourde, MélodieB., et al. (2018). A Rust Fungal Effector Binds Plant DNA and Modulates Transcription. *Sci. Rep.* 8 (1), 14718. doi: 10.1038/s41598-018-32825-0

Anand, A., Trick, H. N., Gill, B. S., and Muthukrishnan, S. (2003). Stable Transgene Expression and Random Gene Silencing in Wheat. *Plant Biotechnol. J.* 1 (4), 241–251. doi: 10.1046/j.1467-7652.2003.00023.x

Anderson, C., Khan, M. A., Catanzariti, A.-M., Jack, C. A., Nemri, A., Lawrence, G. J., et al. (2016). Genome Analysis and Avirulence Gene Cloning Using a High-Density RADseq Linkage Map of the Flax Rust Fungus, Melampsora Lini. *BMC Genomics* 17 (1), 667. doi: 10.1186/s12864-016-3011-9

Arora, S., Steuernagel, B., Gaurav, K., Chandramohan, S., Long, Y., Matny, O., et al. (2019). Resistance Gene Cloning from a Wild Crop Relative by Sequence Capture and Association Genetics. *Nat. Biotechnol.* 37 (2), 139–143. doi: 10.1038/s41587-018-0007-9

Ashelford, K., Eriksson, M. E., Allen, C. M., D'Amore, R., Johansson, M., Gould, P., et al. (2011). Full Genome Re-Sequencing Reveals a Novel Circadian Clock Mutation in Arabidopsis. *Genome Biol.* 12 (3), R28. doi: 10.1186/gb-2011-12-3-r28

Beddow, J. M., Hurley, T. M., Kriticos, D. J., and Pardey, P. G. (2013). *Measuring the Global Occurrence and Probabilistic Consequences of Wheat Stem Rust. HarvestChoice Technical Note* St. Paul, MN: HarvestChoice. Available at: https://www.instepp.umn.edu/products/measuring-global-occurrence-and-probabilistic-consequences-wheat-stem-rust. Accessed 2016/02/15.

Bouton, Clément, King, R. C., Chen, H., Azhakanandam, K., Bieri, Stéphane, Hammond-Kosack, K. E., et al. (2018). Foxtail Mosaic Virus: A Viral Vector for Protein Expression in Cereals. *Plant Physiol.* 177 (4), 1352–1367. doi: 10.1104/pp.17.01679

Broad Institute (2009). Picard Tools - By Broad Institute. Available at: http://broadinstitute.github.io/picard/

Chen, S., Songkumarn, P., Venu, R. C., Gowda, M., Bellizzi, M., Hu, J., et al. (2013). Identification and Characterization of in Planta-Expressed Secreted Effector Proteins from Magnaporthe Oryzae That Induce Cell Death in Rice. *Mol. Plant-Microbe Interactions : MPMI* 26 (2), 191–202. doi: 10.1094/MPMI-05-12-0117-R

Chen, J., Upadhyaya, N. M., Ortiz, D., Sperschneider, J., Li, F., Bouton, C., et al. (2017). Loss of AvrSr50 by Somatic Exchange in Stem Rust Leads to Virulence for Sr50 Resistance in Wheat. *Science* 358 (6370), 1607–1610. doi: 10.1126/science.aao4810

Chen, S., Zhang, W., Bolus, S., Rouse, M. N., and Dubcovsky, J. (2018). Identification and Characterization of Wheat Stem Rust Resistance Gene Sr21 Effective against the Ug99 Race Group at High Temperature. *PloS Genet.* 14 (4). doi: 10.1371/journal.pgen.1007287

Chen, S. S., Rouse, M. N., Zhang, W. J., Zhang, X. Q., Guo, Y., Briggs, J., et al. (2019). Wheat Gene Sr60 Encodes a Protein with Two Putative Kinase Domains That Confers Resistance to Stem Rust. *New Phytol.* 225 (2), 948–959. doi: 10.1111/nph.16169

Danecek, P., Auton, A., Abecasis, G., Albers, C. A., Banks, E., Depristo, M. A., et al. (2011). The Variant Call Format and VCFtools. *Bioinf. Appl. Note* 27 (15), 2156–2158. doi: 10.1093/bioinformatics/btr330

De La Concepcion, J. C., Franceschetti, M., Maclean, D., Terauchi, R., Kamoun, S., and Banfield, M. J. (2019). Protein Engineering Expands the Effector Recognition Profile of a Rice NLR Immune Receptor. *ELife* 8, e47713. doi: 10.7554/eLife.47713

Dodds, P. N., and Rathjen, J. P. (2010). Plant Immunity: Towards an Integrated View of Plant–Pathogen Interactions. *Nat. Rev. Genet.* 11 (8), 539–548. doi: 10.1038/nrg2812

Duplessis, S., Cuomo, C. A., Lin, Y., Aerts, A., Tisserant, E., Veneault-Fourrey, C., et al. (2011). Obligate Biotrophy Features Unraveled by the Genomic Analysis of Rust Fungi. *Proc. Natl. Acad. Sci.* 102 (22), 9166–9171. doi: 10.1073/pnas.1019315108

Farrell, A., Coleman, B.II, Benenati, B., Brown, K. M., Blader, I. J., Marth, G. T., et al. (2014). Whole Genome Profiling of Spontaneous and Chemically Induced Mutations in Toxoplasma Gondii. *BMC Genomics* 15 (1):354. doi: 10.1186/1471-2164-15-354

Flor, H. H. (1955). Host-Parasite Interactions in Flax Rust: Its Genetics and Other Implications. *Phytopathology* 45, 680–685.

Fukuoka, S., Saka, N., Mizukami, Y., Koga, H., Yamanouchi, U., Yoshioka, Y., et al. (2015). Gene Pyramiding Enhances Durable Blast Disease Resistance in Rice. *Sci. Rep.* 5:7773. doi: 10.1038/srep07773 (January).

Hiebert, C. W., Moscou, M. J.,, Hewitt, T., Steuernagel, B., Hernández-Pinzón, I., Green, P., et al. (2020). Stem rust resistance in wheat is suppressed by a subunit of the mediator complex. *Nat. Commun.* 11 (1), 1123. doi: 10.1038/s41467-020-14937-2

Harris, C. J., Slootweg, E. J., Goverse, A., and Baulcombe, D. C. (2013). Stepwise Artificial Evolution of a Plant Disease Resistance Gene. *Proc. Natl. Acad. Sci. United States America* 110 (52), 21189–21194. doi: 10.1073/pnas.1311134110

Hovmøller, M. S., and Justesen, A. F. (2007). Appearance of Atypical Puccinia Striiformis f. Sp. Tritici Phenotypes in North-Western Europe. *Aust. J. Agric. Res.* 58 (6), 518. doi: 10.1071/AR06146

Huang, Y. J., Li, Z. Q., Evans, N., Rouxel, T., Fitt, B. D. L., and Balesdent, M. H. (2006). Fitness Cost Associated with Loss of the AvrLm4 Avirulence Function in Leptosphaeria Maculans (Phoma Stem Canker of Oilseed Rape). In. *Eur. J. Plant Pathol.* 114, 77–89. doi: 10.1007/s10658-005-2643-4

Hurni, S., Brunner, S., Stirnweis, D., Herren, G., Peditto, D., McIntosh, R. A., et al. (2014). The Powdery Mildew Resistance Gene *Pm8* Derived from Rye Is Suppressed by Its Wheat Ortholog *Pm3*. *Plant J.* 79 (6), 904–913. doi: 10.1111/tpj.12593

Johnson, T. (1954). Selfing Studies With Physiologic Races Of Wheat Stem Rust, Puccinia Graminis Var Tritici. *Can. J. Bot.* 32 (4), 506–522. doi: 10.1139/b54-047

Johnson, T. (1961). Man-Guided Evolution in Plant Rusts: Through His Modification of the Host Plants of the Cereal Rusts, Man Is Also Modifying the Rusts. *Science* 133 (3450), 357–362. doi: 10.1126/science.133.3450.357

Kawamura, K., and Maruyama, I. N. (2019). Forward Genetic Screen for Caenorhabditis Elegans Mutants with a Shortened Locomotor Healthspan. *G3: Genes Genomes Genet.* 9 (8), 2415–2423. doi: 10.1534/g3.119.400241

Keller, B., Wicker, T., and Krattinger, S. G. (2018). Advances in Wheat and Pathogen Genomics: Implications for Disease Control. *Annu. Rev. Phytopathol.* 56 (1), 67–87. doi: 10.1146/annurev-phyto-080516-035419

Kim, S. H., Qi, D., Ashfield, T., Helm, M., and Innes, R. W. (2016). Using Decoys to Expand the Recognition Specificity of a Plant Disease Resistance Protein. *Science* 351 (6274), 684–687. doi: 10.1126/science.aad3436

Koren, S., Walenz, B. P., Berlin, K., Miller, J. R., Bergman, N. H., and Phillippy, A. M. (2017). Canu: Scalable and Accurate Long-Read Assembly via Adaptive κ-Mer Weighting and Repeat Separation. *Genome Res.* 27 (5), 722–736. doi: 10.1101/gr.215087.116

Krasileva, K. V., Vasquez-Gross, H. A., Howell, T., Bailey, P., Paraiso, F., Clissold, L., et al. (2017). Uncovering Hidden Variation in Polyploid Wheat. *Proc. Natl. Acad. Sci. United States America* 114 (6), E913–E921. doi: 10.1073/pnas.1619268114

Kurowski, T. J., and Mohareb, F. (2020). Tersect: A Set Theoretical Utility for Exploring Sequence Variant Data. *Bioinformatics* 36 (3), 934–935. doi: 10.1093/bioinformatics/btz634

Lebkowski, J. S., Miller, J. H., and Calos, M. P. (1986). Determination of DNA Sequence Changes Induced by Ethyl Methanesulfonate in Human Cells, Using a Shuttle Vector System. *Mol. Cell. Biol.* 6 (5), 1838–1842. doi: 10.1128/mcb.6.5.1838

Lewis, C. M., Persoons, A., Bebber, D. P., Kigathi, R. N., Maintz, J., Findlay, K., et al. (2018). Potential for Re-Emergence of Wheat Stem Rust in the United Kingdom. *Commun. Biol.* 1 (1), 13. doi: 10.1038/s42003-018-0013-y

Li, H., and Durbin, R. (2009). Fast and Accurate Short Read Alignment with Burrows-Wheeler Transform. *Bioinformatics* 25 (14), 1754–1760. doi: 10.1093/bioinformatics/btp324

Li, Z., Liu, Y., and Berger, P. H. (2005). Transgene Silencing in Wheat Transformed with the WSMV-CP Gene. *Biotechnology* 4 (1), 62–68. doi: 10.3923/biotech.2005.62.68

Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., et al. (2009). The Sequence Alignment/Map Format and SAMtools. *Bioinformatics* 25 (16), 2078–2079. doi: 10.1093/bioinformatics/btp352

Li, F., Upadhyaya, N. M., Sperschneider, J., Matny, O., Nguyen-Phuc, H., Mago, R., et al. (2019). Emergence of the Ug99 Lineage of the Wheat Stem Rust Pathogen through Somatic Hybridisation. *Nat. Commun.* 10 (1), 5068. doi: 10.1038/s41467-019-12927-7

Lo Presti, L., Lanver, D., Schweizer, G., Tanaka, S., Liang, L., Tollot, M., et al. (2015). Fungal Effectors and Plant Susceptibility. *Annu. Rev. Plant Biol.* 66, 513–545. doi: 10.1146/annurev-arplant-043014-114623

Luig, N. H. (1978). "Mutation Studies in Puccinia Graminis Tritici," in *Proc. 5th Int. Wheat Genetics Symposium*. Ed. S. Ramanujam (New Dehli: Indian Society of Genetics & Plant Breeding), 533–539.

Mago, R., Zhang, P., Vautrin, S., Šimková, H., Bansal, U., Luo, M. C., et al. (2015). The Wheat Sr50 Gene Reveals Rich Diversity at a Cereal Disease Resistance Locus. *Nat. Plants* 1 (12), 15186. doi: 10.1038/nplants.2015.186

Maqbool, A., Saitoh, H., Franceschetti, M., Stevenson, C. E. M., Uemura, A., Kanzaki, H., et al. (2015). Structural Basis of Pathogen Recognition by an Integrated HMA Domain in a Plant NLR Immune Receptor. *ELife* 4, e08709. doi: 10.7554/eLife.08709

Melania, M., Dodds, P. N., and Henningsen, E. C. (2020). Evolution of Virulence in Rust Fungi — Multiple Solutions to One Problem. *Curr. Opin. Plant Biol.* doi: 10.1016/j.pbi.2020.02.007

McKenna, A., Hanna, M., Banks, E., Sivachenko, A., Cibulskis, K., Kernytsky, A., et al. (2010). The Genome Analysis Toolkit: A MapReduce Framework for Analyzing next-Generation DNA Sequencing Data. *Genome Res.* 20 (9), 1297–1303. doi: 10.1101/gr.107524.110

Nagar, R., and Schwessinger, B. (2018). Multi-Step High Purity High Molecular Weight DNA Extraction Protocol from Challenging Fungal Tissues. *Protocols.Io* 1–10. doi: 10.17504/protocols.io.rzkd74w

Olivera Firpo, P. D., Newcomb, M., Flath, K., Sommerfeldt-Impe, N., Szabo, L. J., Carter, M., et al. (2017). Characterization of Puccinia Graminis f. Sp. Tritici Isolates Derived from an Unusual Wheat Stem Rust Outbreak in Germany in 2013. *Plant Pathol.* 66 (8), 1258–1266. doi: 10.1111/ppa.12674

Ortiz, D., de Guillen, K., Cesari, S., Chalvon, Véronique, Gracy, Jérome, Padilla, André, et al. (2017). Recognition of the Magnaporthe Oryzae Effector AVR-Pia by the Decoy Domain of the Rice NLR Immune Receptor RGA5. *Plant Cell* 29 (1), 156–168. doi: 10.1105/tpc.16.00435

Pedersen, C., van Themaat, E. V. L., McGuffin, L. J., Abbott, J. C., Burgis, T. A., Barton, G., et al. (2012). Structure and Evolution of Barley Powdery Mildew Effector Candidates. *BMC Genomics* 13 (1), 694. doi: 10.1186/1471-2164-13-694

Periyannan, S., Moore, J., Ayliffe, M., Bansal, U., Wang, X., Huang, Li, et al. (2013). The Gene Sr33, an Ortholog of Barley Mla Genes, Encodes Resistance to Wheat Stem Rust Race Ug99. *Sci. (New York N.Y.)* 341 (6147), 786–788. doi: 10.1126/science.1239028

Periyannan, S. (2018). Sustaining Global Agriculture through Rapid Detection and Deployment of Genetic Resistance to Deadly Crop Diseases. *New Phytol.* 219 (1), 45–51. doi: 10.1111/nph.14928

Pretorius, Z. A., Singh, R. P., Wagoire, W. W., and Payne, T. S. (2000). Detection of Virulence to Wheat Stem Rust Resistance Gene Sr31 in Puccinia Graminis. f. Sp. Tritici in Uganda. *Plant Dis.* 84 (2), 203. doi: 10.1094/PDIS.2000.84.2.203B

Saintenac, C., Zhang, W., Salcedo, A., Rouse, M. N., Trick, H. N., Akhunov, E., et al. (2013). Identification of Wheat Gene Sr35 That Confers Resistance to Ug99 Stem Rust Race Group. *Science* 341 (6147), 783–786. doi: 10.1126/science.1239022

Salcedo, A., Rutter, W., Wang, S., Akhunova, A., Bolus, S., Chao, S., et al. (2017). Variation in the AvrSr35 Gene Determines Sr35 Resistance against Wheat Stem Rust Race Ug99. *Science* 358 (6370), 1604–1606. doi: 10.1126/science.aao7294

Sánchez-Martín, J., Steuernagel, B., Ghosh, S., Herren, G., Hurni, S., Adamski, N., et al. (2016). Rapid Gene Isolation in Barley and Wheat by Mutant Chromosome Sequencing. *Genome Biol.* 17 (1), 221. doi: 10.1186/s13059-016-1082-1

Saunders, D. G.O., Win, J., Cano, L. M., Szabo, L. J., Kamoun, S., and Raffaele, S. (2012). Using Hierarchical Clustering of Secreted Protein Families to Classify and Rank Candidate Effectors of Rust Fungi. Edited by Jason E. Stajich. *PloS One* 7 (1), e29847. doi: 10.1371/journal.pone.0029847

Saunders, D. G. O., Pretorius, Z. A., and Hovmøller, M. S. (2019). Tackling the Re-Emergence of Wheat Stem Rust in Western Europe. *Commun. Biol.* 2 (1), 51. doi: 10.1038/s42003-019-0294-9

Saur, I. M. L., Bauer, S., Lu, X., and Schulze-Lefert, P. (2019). A Cell Death Assay in Barley and Wheat Protoplasts for Identification and Validation of Matching Pathogen AVR Effector and Plant NLR Immune Receptors. *Plant Methods* 15 (1), 118. doi: 10.1186/s13007-019-0502-0

Schumann, G. L., and Leonard, K. J. (2000). Stem Rust of Wheat (Black Rust). *Plant Health Instructor* 58, 1–12. doi: 10.1094/PHI-I-2000-0721-01 (April).

Segretin, E., Pais, M., Franceschetti, M., Chaparro-Garcia, A., Bos, J. I. B., Banfield, M. J., et al. (2014). Single Amino Acid Mutations in the Potato Immune Receptor R3a Expand Response to Phytophthora Effectors. *Mol. Plant-Microbe Interact. MPMI* 27 (7), 624-637. doi: 10.1094/MPMI-02-14-0040-R

Seto, D., Koulena, N., Lo, T., Menna, A., Guttman, D. S., and Desveaux, D. (2017). Expanded Type III Effector Recognition by the ZAR1 NLR Protein Using ZED1-Related Kinases. *Nat. Plants* 3 (4), 1–4. doi: 10.1038/nplants.2017.27

Shirasawa, K., Hirakawa, H., Nunome, T., Tabata, S., and Isobe, S. (2016). Genome-Wide Survey of Artificial Mutations Induced by Ethyl Methanesulfonate and Gamma Rays in Tomato. *Plant Biotechnol. J.* 14 (1), 51–60. doi: 10.1111/pbi.12348

Singh, R. P., Hodson, D. P., Huerta-Espino, J., Jin, Y., Njau, P., Wanyera, R., et al. (2008). Will Stem Rust Destroy the World's Wheat Crop? *Adv. Agron.* 98, 271–309. doi: 10.1016/S0065-2113(08)00205-8 Academic Press.

Sohn, Jürgen, Voegele, R. T., Mendgen, K., and Hahn, M. (2000). High Level Activation of Vitamin B1 Biosynthesis Genes in Haustoria of the Rust Fungus Uromyces Fabae. *Mol. Plant-Microbe Interact.* 13 (6), 629–636. doi: 10.1094/MPMI.2000.13.6.629

Stakman, E. C., Stewart, D. M., and Loegering, W. Q. (1962). *Identification of Physiologic Races of Puccinia Graminis Var. Tritici.* USDA ARS E-617 pp 5-53. Washington DC: U.S. Gov. Print. Office. Available at: https://naldc.nal.usda.gov/download/CAT10243018/PDF (Accessed: 24 March 2017).

Stanke, M., and Morgenstern, B. (2005). AUGUSTUS: A Web Server for Gene Prediction in Eukaryotes That Allows User-Defined Constraints. *Nucleic Acids Res.* 33 (SUPPL. 2), W465–W467. doi: 10.1093/nar/gki458

Stassen, J. H. M., Seidl, M. F., Vergeer, P. W. J., Nijman, IsaäcJ., Snel, B., Cuppen, E., et al. (2012). Effector Identification in the Lettuce Downy Mildew Bremia Lactucae by Massively Parallel Transcriptome Sequencing. *Mol. Plant Pathol.* 13 (7), 719–731. doi: 10.1111/j.1364-3703.2011.00780.x

Steuernagel, B., Periyannan, S. K., Hernández-Pinzón, I., Witek, K., Rouse, M. N., Yu, G., et al. (2016). Rapid Cloning of Disease-Resistance Genes in Plants Using Mutagenesis and Sequence Capture. *Nat. Biotechnol.* 34 (6), 652–655. doi: 10.1038/nbt.3543

Teo, C., and Baker, E. P. (1966). Mutants of Puccinia Graminis Avenae Induced by Ethyl Methane Sulphonate. *Nature* 209 (2), 632–633. doi: 10.1038/209632b0

Thara, V. K., Fellers, J. P., and Zhou, J. M. (2003). In Planta Induced Genes of Puccinia Triticina. *Mol. Plant Pathol.* 4 (1), 51–56. doi: 10.1046/j.1364-3703.2003.00142.x

Upadhyaya, N. M., Mago, R., Staskawicz, B. J., Ayliffe, M. A., Ellis, J. G., and Dodds, P. N. (2014). A Bacterial Type III Secretion Assay for Delivery of Fungal Effector Proteins into Wheat. *Mol. Plant-Microbe Interact.* 27 (3), 255–264. doi: 10.1094/MPMI-07-13-0187-FI

Upadhyaya, N. M., Garnica, D. P., Karaoglu, H., Sperschneider, J., Nemri, A., Xu, Bo, et al. (2015). Comparative Genomics of Australian Isolates of the Wheat Stem Rust Pathogen Puccinia Graminis f. Sp. Tritici Reveals Extensive Polymorphism in Candidate Effector Genes. *Front. Plant Sci.* 5, 759. doi: 10.3389/fpls.2014.00759 (January).

Vale, F., Xavier,, Ribeiro, D. O., Parlevliet, J. E., and Zambolim, L. (2001). Concepts in Plant Disease Resistance. *Fitopatologia Bras.* 26 (3), 577–589. doi: 10.1590/s0100-41582001000300001

Vleeshouwers, V. G. A. A., and Oliver, R. P. (2014). Effectors as Tools in Disease Resistance Breeding against Biotrophic, Hemibiotrophic, and Necrotrophic Plant Pathogens. *Mol. Plant Microbe Interact.* 27 (3), 196–206. doi: 10.1094/MPMI-10-13-0313-IA

Vleeshouwers, V. G. A. A., Rietman, H., Krenek, P., Champouret, N., Young, C., Oh, S. K., et al. (2008). Effector Genomics Accelerates Discovery and Functional Profiling of Potato Disease Resistance and Phytophthora Infestans Avirulence Genes. Edited by Hany A. El-Shemy. *PloS One* 3 (8), e2875. doi: 10.1371/journal.pone.0002875

Walker, B. J., Abeel, T., Shea, T., Priest, M., Abouelliel, A., Sakthikumar, S., et al. (2014). Pilon: An Integrated Tool for Comprehensive Microbial Variant Detection and Genome Assembly Improvement. Edited by Junwen Wang. *PloS One* 9 (11), e112963. doi: 10.1371/journal.pone.0112963

Wang, H., Chattopadhyay, A., Li, Z., Daines, B., Li, Y., Gao, C., et al. (2010). Rapid Identification of Heterozygous Mutations in Drosophila Melanogaster Using Genomic Capture Sequencing. *Genome Res.* 20 (7), 981–988. doi: 10.1101/gr.102921.109

Waterhouse, R. M., Seppey, M., Simao, F. A., Manni, Mosè, Ioannidis, P., Klioutchnikov, G., et al. (2018). BUSCO Applications from Quality Assessments to Gene Prediction and Phylogenomics. *Mol. Biol. Evol.* 35 (3), 543–548. doi: 10.1093/molbev/msx319

Wulff, B. B. H., and Moscou, M. J. (2014). Strategies for Transferring Resistance into Wheat: From Wide Crosses to GM Cassettes. *Front. Plant Sci.* 5 (692), 1–11. doi: 10.3389/fpls.2014.00692

Wulff, B. B. H., Hesse, A., Tomlinson-Buhot, L., Jones, D. A., De La Peña, M., and Jones, J. D. G. (2009). The Major Specificity-Determining Amino Acids of the Tomato Cf-9 Disease Resistance Protein Are at Hypervariable Solvent-Exposed Positions in the Central Leucine-Rich Repeats. *Mol. Plant-Microbe Interact.* 22 (10), 1203–1213. doi: 10.1094/MPMI-22-10-1203

Yu, G., Champouret, N., Steuernagel, B., Olivera, P. D., Simmons, J., Williams, C., et al. (2017). Discovery and Characterization of Two New Stem Rust Resistance Genes in Aegilops Sharonensis. *Theor. Appl. Genet.* 130 (6), 1207–1222. doi: 10.1007/s00122-017-2882-8

Zhang, N. W., Pelgrom, K., Niks, R. E., Visser, R. G. F., and Jeuken, M. J. W. (2009). Three Combined Quantitative Trait Loci from Nonhost Lactuca Saligna Are Sufficient to Provide Complete Resistance of Lettuce against Bremia Lactucae. *Mol. Plant-Microbe Interact.* 22 (9), 1160–1168. doi: 10.1094/MPMI-22-9-1160

Zhang, W., Chen, S., Abate, Z., Nirmala, J., Rouse, M. N., and Dubcovsky, J. (2017). Identification and Characterization of Sr13, a Tetraploid Wheat Gene That Confers Resistance to the Ug99 Stem Rust Race Group. *Proc. Natl. Acad. Sci. United States America* 114 (45), E9483–E9492. doi: 10.1073/pnas.1706277114

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.