Towards comparative pan-genomics

Siavash Sheikhizadeh Anari

Towards comparative pan-genomics

Siavash Sheikhizadeh Anari

Thesis committee

Promotors
Prof. Dr D. de Ridder
Professor of Bioinformatics
Wageningen University & Research

Prof. Dr M.E. Schranz
Professor of Biosystematics
Wageningen University & Research

Co-promotor
Dr S. Smit
Assistant Professor, Bioinformatics Group
Wageningen University & Research

Other members
Prof. Dr M. Groenen, Wageningen University & Research
Prof. Dr S. Rahmann, University of Duisburg-Essen and
Technical University of Dortmund, Germany
Dr E.M. Zdobnov, University of Geneva Medical School, Switzeland
Dr G. Bonnema, Wageningen University & Research

This research was conducted under the auspices of the Graduate School of Experimental Plan Sciences (EPS)

Towards comparative pan-genomics

Siavash Sheikhizadeh Anari

Thesis
submitted in fulfilment of the requirements for the degree of doctor
at Wageningen University
by the authority of the Rector Magnificus,
Prof. Dr A.P.J. Mol
in the presence of the
Thesis Committee appointed by the Academic Board
to be defended in public
on Thursday 12 November 2020
at 11 a.m. in the Aula.

Siavash Sheikhizadeh Anari
Towards comparative pan-genomics
109 pages

PhD thesis, Wageningen University, Wageningen, the Netherlands (2020)
With references, with summary in English

ISBN: 978-94-6395-568-3
DOI: https://doi.org/10.18174/532136

Contents

Chapter 1

Introduction

1.1 Basics of molecular biology

When Friedrich Miescher isolated an unknown material from the nuclei of white blood cells in 1869, he had no idea that these molecules encoded the information needed to perpetuate life of all organisms. It took until the mid-1950s for researchers to elucidate the hereditary role of these so-called deoxyribonucleic acid, or DNA, molecules and thus revolutionize our understanding of biology and life on earth. DNA has a twisted ladder-like structure of nucleotides, folded and packed in genetic units called chromosomes. The complete set of chromosomes of a species is called the genome. Chromosomes appear in different numbers, sizes and content in different species. Where prokaryotes have a single circular chromosome as genome, the genome of eukaryotes are usually made up of sets of highly similar homologous chromosomes, half of which are inherited from each parent. The number of homologous chromosomes, called the ploidy, also varies between species. For example, the human genome comprises of 23 pairs (diploid) of chromosomes but the wheat genome has 6 homologous copies (hexaploid) of each of its 7 chromosomes.

Genomic differences do not occur by chance but as a result of billion years of evolution, through which genomic content of species has been shuffled, mixed and mutated. By investigating these differences scientists are able to trace back the evolutionary history of different species to unravel their ancestral relationship. Genomic differences can also explain important characteristics of species, for example resistance of a crop to a pest or vulnerability of a person to a type of cancer. Casual genomic differences, leading to different characteristics, often appear in functional parts of the genome, called *genes*, that encode for building blocks of living cells. Genes are transcribed into transcripts, messenger *RNA* (ribonucleic acid), which are then used by the translation machinery of the cell to produce the macromolecule of *proteins*. There are numerous types of proteins within the cell, forming its physical structure or regulating various biological functions. The entire set of proteins of species is called the *proteome*. The human proteome, for example, consists of ~20,000 different proteins.

1.2 Genomics and the revolution of sequencing

To investigate genomes, first we must uncover the nucleotide sequence of chromosomes, through whole-genome sequencing (WGS) or in specific regions/genes using targeted amplicon sequencing (TAS) [1]. DNA material is sequenced in short/long overlapping fragments called *reads* [2]. Sequencing reads are pieces of the original genome which can be *assembled* into longer contiguous sequences (*contigs*), which may be ordered and oriented into *scaffolds* to give a more complete picture of the genome. Finally, assembled genomes are *annotated* to determine structure and function of genomic features, such as protein-coding genes, *tRNAs* [3], *miRNAs* [4] and *motifs* [5].

Materialized in 1970s, *DNA sequencing* is now in the midst of a revolution which currently allows us to read the entire 3.5 billion base pairs of the human genome in a single day. The number of nucleotide records in the NCBI RefSeq database [6] has been annually increasing by 50%, on average, since its foundation in 2003. The latest release (March 2020) harbors 1.86×10^{12} nucleotides from 99,842 taxa (Figure 1.1). It is expected that the number of assembled genomes will keep expanding in the coming decades. At the same time, the quality and contiguity of genome assemblies will also improve, due to advances in sequencing technologies. Second-generation sequencing reads are short (a few hundred base pairs) and introduce ambiguity in assembling repetitive regions of the genome and also in separating haplotypes. New technologies such as *long-read sequencing* (Oxford Nanopore Technology) and *optical mapping* (Bionano Genomics) are very promising towards chromosome-scale assembly of large and complex genomes [7,8]. Combined with high-quality Illumina short reads, long reads can also facilitate haplotype-resolved assembly [9].

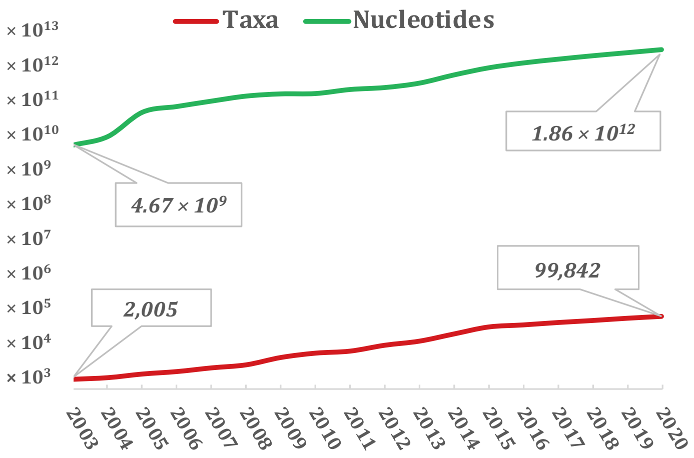


Figure 1.1. The NCBI RefSeq database has been growing exponentially. The green line indicates the number of nucleotides and the red line shows the number of different taxa over a period of 18 years in this database till March 2, 2020 (RefSeq release 99).

1.3 Comparative genomics

Individual genomes are not very useful without any knowledge about their structure and function. The collective study of genomes enables us to use the existing knowledge on some genomes to infer function and characteristics of other related genomes. Unraveling these relationships has led to novel discoveries in many application domains, such as microbiology to design medicines, plant breeding to improve the yield of crops and cancer research to decide on effective treatments. Physical characteristics or *traits* of individuals/species define their general *phenotypes*. Differences in phenotypes, besides environmental factors, stem from variation in the genomic makeup or *genotypes* of species. *Comparative genomics* is a branch of research that investigates the evolution of genotypes and links to phenotypes by characterizing similarity and divergence of the genomes of species. Genomic differences can vary from simple mutations, such as *single nucleotide polymorphisms* (*SNPs*) and short *insertions/deletions* (*indels*), to large *segmental duplications*, or drastic changes of karyotype and ploidy of the genomes [10]. Considering a single gene of a species, the genotype of an individual with regard to that gene is determined by the copies of that gene (two copies in diploids) in the homologous chromosomes, called *alleles*. If the sequence of two alleles are identical, they are *homozygous*, otherwise they are *heterozygous* at that locus.

Occurring randomly, most sequence mutations are neutral, however mutations in functional features of the genome often impact biological functions of the cell. Evolution preserves beneficial mutations (*positive selection*) and purifies populations from deleterious mutations (*negative selection*). The evolutionary relationship of species is usually visualized as a *phylogenetic tree*, where the leaves are species and internal nodes represent the *common ancestor*s. Genomic features which are conserved between two species are ancestrally related (*homologs*). Homologous features may be inherited from a single feature (*orthologs*) or from distinct duplicated features (*paralogs*) in their last common ancestor. Orthologous features often preserve their order in a chromosome (*synteny*).

Interspecies comparative genomics tries to detect the functional parts of genomes of different species by *pair-wise alignment* of genomes [11]. The principle behind it is that genomic features that encode/regulate proteins responsible for similar biological functions are conserved between two species and, conversely, those that encode/regulate their differences are diverged [12]. However, this is very challenging as computation becomes harder with increasing genome sizes and the number of pair-wise comparisons grows quadratically with the number of species. As a result, many research groups provide precomputed browsable alignments and synteny maps of species of interest [13-16]. Different species can also be compared based on their functional features through *orthology inference*. There are numerous tools and public databases which infer orthology based on sequence similarity, phylogeny, synteny or a combination of those [17].

Intraspecies comparative genomics tries to find genomic variation between individuals of the same species. In such studies, simple variation is detected between individuals by aligning sequencing reads to a reference genome (*read mapping)* and collecting variable loci such as SNPs and short indels from the pileup of reads (*variant calling*). Large regions in which no reads map show large *deletions* in the sequenced genome, split reads help to determine the boundaries of large structural events such as *insertions*, *inversions* and *translocations*, and drastic changes in the average number of reads spanning a locus (*coverage*) compared to flanking regions can indicate *duplications*.

It becomes clear that reference genomes play a central role in intraspecies comparative genomics. However, a single reference genome cannot represent an entire species. In the first place, it ignores the intraspecies variability. Second, a reference genome is a haploid simplification of the genome of species that are often diploid or even polyploid and, as a result, mapping reads against a single reference is always biased towards the reference allele in highly heterozygous sites. This can mislead genotyping as the alternative alleles may not be discovered due to failure of read mapping. Third, reads originating from genomic regions that are absent in the reference will be ignored while they may contain important rare variants, for example somatic mutations that are causal for cancer. To overcome these limitations and make full use of the wealth of genomes available for many species, a transition from reference-based to pan-genomic comparative genomics has emerged in recent years [18].

1.4 Pan-genomes

In line with the number of sequenced individuals, research in the field of *pan-genomics* has been growing rapidly in the recent decade. In the PubMed database [19], the term "*pangenome*" or "*pan-genome*" receives hits (in title/abstract) since the year 2000, reaching to over 1,000 hits by the end of 2019 (Figure 1.2). A collection of genomes, haplotypes, genes, or any genomic feature from a phylogenetic clade which are analyzed jointly can be called a pan-genome [18]. As a reference, pan-genomes are supposed to capture a wider genomic landscape of species compared to a single reference genome.

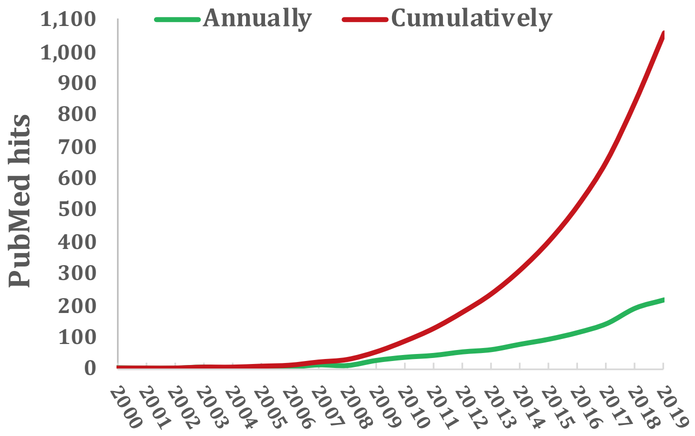


Figure 1.2. The number of publications with terms "pangenome" or "pan-genome" in the title or abstract has been growing in PubMed database, since 2000. Red curve indicates cumulative number of hits, so 1000 publications by the end of 2019.

In literature, various definitions of pan-genomes are in use, in terms of genomic content, taxonomy level, functionality and data structures. Initially, pan-genomes were defined at the gene level in microbial species to study their gene repertoire and categorize genes as core, dispensable or strain-specific [20]. Subsequent developments constructed pan-genomes from complete genomes [21,22]. There are also examples of pan-genomes at the transcriptome level (pan-transcriptome) utilized to investigate presence/absence variation between transcriptomes [23]. Pan-genomes have been also defined at different taxonomic levels. For example, viral quasi-species [24,25] are pan-genomes at the strain level, the *Vibrio cholerae* pan-genome [26] is defined at the species level, and the Bacillus pan-genome [27] at the level of a genus. Pan-genomes have been built even from different meta-genomes, for example, the Prochlorococcus pan-genome is a pan-metagenome used to link clusters of genes to their environmental distribution in a marine metagenomics context [28].

The Computational Pan-Genomics Consortium, a group of researchers from different application domains, from virology to microbiology, human and cancer genomics, have presented the most comprehensive definition of a pan-genomic platform, in the literature of the field [18] (Figure 1.3). In this view, it should be possible to construct pan-genomes from whole-genome or targeted assemblies. Any representation requires efficient and consistent retrieval and storage of data. To be used as a reference and detect small and large-scale genomic variation, pan-genomes should facilitate read mapping and variant calling. Variants should be expressed based on a consistent and stable pan-genomic coordinate system which is efficiently projectable to individual genomes. Additional data stores might be considered for discovered variants and haplotypes. To incorporate new genomes, the pan-genome data structure and, in turn, the coordinates should be efficiently updated. Structural features of genomes need be annotated in the pan-genome. Simplified visualizations at different layers of aggregation are needed to unfold pan-genome substructures. Finally, more realistic genome simulators can be developed considering the common haplotypes and their frequency information. Such a platform can be even extended to be able to answer other relevant biological questions. Currently, a lot of research is being dedicated to exploring data structures and algorithms that can make such computational infrastructure a reality.

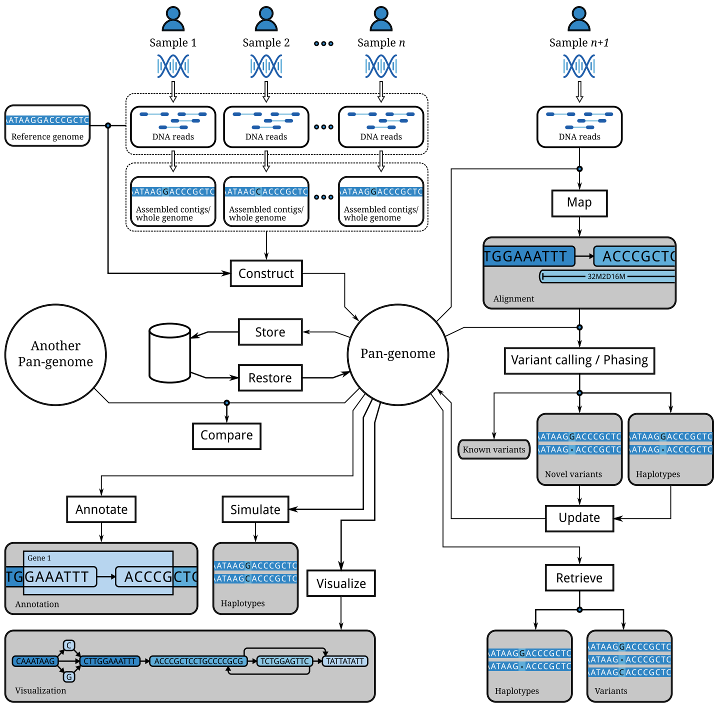


Figure 1.3. Functionalities suggested by the Computational Pan-Genomics Consortium to be supported in a pan-genomic platform. This figure is re-used from [18]. Pan-genome is **construct**ed from a given set of samples/genomes and is permanently **stored** for future use. Reads from newly sequenced individuals are **map**ped against the pan-genome to call novel **variants** and **haplotypes** which will be incrementally added to the pan-genome and can be **retrieve**d on demand. Population haplotype frequencies cab be used to **simulate** new populations. **Comparing** two pan-genomes can narrow down the candidate genomic source of phenotypes in different populations. Pan-genomes need to be **annotated** and **visualized** to be used in real applications.

1.5 Data structures

In a pan-genomic platform, data organization directly affects the feasibility and efficiency of the functionalities that manipulate the data. Given the wide range of potential uses, it is not expected that a single data structure can be found which satisfies all the needs, equally well. We classify existing pan-genomic data structures into two general representations: variation-aware and multi-genome data structures. Variation-aware data structures enrich a single reference with known variants, while multi-genome data structures combine multiple assembled genomes.

1.5.1 Variation-aware structures

In a population of the same species with rare large-scale genomic rearrangements, a reference genome plus a large number of simple variants/haplotypes can sufficiently represent that population. The genomic makeup of any new individual can then be expressed as a combination of alleles at the variable loci of a single reference. Such variation-aware data structures have been extensively used in human pan-genomics [29], as a wealth of variants have been already discovered in human genome. For example, dbSNP Build 152 contains more than 650 million short variants (<=50bp). Variation-aware data structures can be categorized in *reference-based* and *graph-based* structures.

Reference-based variation-aware methods keep the reference genome separated from the data structure of the variants. For example, RCSI [30] builds two indices for pattern search in the genomes, one for the reference and the other for deviation of genomes from the reference. Similarly, BWBBLE [31] extends the reference genome to a linear multi-genome, by appending IUPAC-coded variants in other genomes to the reference. This linear structure is then BWT-indexed for read mapping against all the constituent genomes. In the same vein, MuGI [32] constructs a variation database and a *k*-mer index for read mapping. The journaled string tree [33] is a reference-based data structure which provides an efficient simultaneous sequential search over a set of highly similar genomes.

Graph-based variation-aware methods combine the reference genome and known variants or/and haplotypes in a graph structure, where paths represent the possible recombinants in the population. For example, GraphTyper [34] constructs a directed acyclic graph from a reference genome and a set of known variants. Unaligned/clipped reads coming from complex regions of the genome are mapped to this augmented graph using a *k*-mer index, haplotypes are called, variants are genotyped with respect to the reference, and the novel variants are incorporated in the graph. Similarly, the population reference graph [29] is a directed acyclic graph constructed from assembled haplotypes of MHC region and SNPs from the 1000 Genomes Project and classical HLA alleles from IMGT [35], to improve the accuracy of genome inference in this complex region. Likewise, the variation graph (vg) toolkit [36] constructs a bi-directed variation-aware graph to improve read mapping in highly polymorphic regions of human genome.

1.5.2 Multi-genome structures

Variation-aware data structures have been effectively utilized to improve variant calling in highly polymorphic regions of the human genome, but they do not facilitate detection of structural variation as found in dynamic genomes such as those of fungi and plants. They assume a strong collinearity between genomes and require a large number of known variants, beforehand. When a wealth of assemblies is available, genomes are highly structurally dynamic, or the variability of the genomes is less well-known, use of a multi-genome data structure is the desirable approach. Multi-genome data structures can be categorized in *alignment-based* and *graph-based* structures.

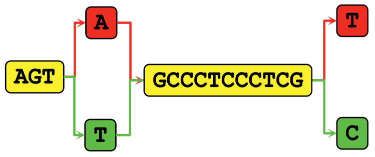
Alignment-based data structures mostly use pre-calculated multiple sequence alignments (MSA) to represent sequence similarities in the collinear segments of the genomes. An advantage is that the columns of an MSA define a coordinate system over sequences, which can be efficiently projected to original coordinates in each sequence [37]. Also, MSAs can be indexed for pattern search; for example, GCSA [38] uses a (Borrows-Wheeler transform) BWT-index over a finite automaton representation of an MSA to allow pattern search inside recombinants, and PanVC [39] constructs a hybrid index [40] to map reads against the MSA. Such tools have been applied to short polymorphic regions of the human genome. However, building an MSA of a large number of structurally variable genomes is a large challenge [41-43].

Graph-based alignment structures, such as POA [44], A-Bruijn [45] and Cactus [46] graphs have tried to address representation of recombinants, structural variation and duplications, at the same time. ProgressiveMauve [47] takes a similar approach, although it does not make a graph. It partitions genomes into locally collinear blocks (LCB) and builds MSAs of each block in a multiple whole-genome alignment (MWGA). To extend an existing MWGA with a new genome, seq-seq-pan [48] generates pairwise alignments between the linear consensus of existing LCBs and the new genome and splits or merges the blocks to update the alignment.

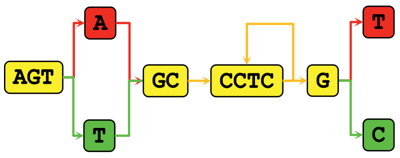
Graph-based multi-genome data structures are able to efficiently represent multiple genomes by collapsing identical regions. In such graphs, nodes are labeled with pieces of nucleotide sequences annotated with their coordinate in each genome, enabling traversing the path of each genome in the graph. Nodes can be of constant or variable length and are connected by directed edges, or by bi-directed edges to represent the strand of the sequences. In a pan-genome graph, cycles represent repetitive sequences in one genome or between multiple genomes. Cyclic pan-genome graphs can be replaced by their acyclic version to preserve all copies of repeated sequences in the structure, of course at the cost of redundant nodes. For example, in Figure 1.4, it is clear that pattern CCTC occurs twice in tandem in both sequences, but in the cyclic version it is not clear how many times this pattern is repeated in each sequence. This issue can also be addressed by annotating nodes or edges with genomic coordinates. This also links coordinates of similar regions between genomes, facilitating homology detection algorithms.

In the absence of a linear coordinate system, graph coordinates can be expressed by pairs of node identifier and offset. Such coordinates demand some book-keeping when node identifiers change, for example due to adding new sequences and splitting some nodes. Alternatively, one of the genomes can be considered to be the reference and corresponding loci in other genomes can be defined with respect to the reference coordinates. This has been implemented in some human pan-genomes [29,34], but it is not applicable in the presence of genomic rearrangements.

The *De Bruijn graph* (DBG) or colored DBG [49] has traditionally been used in genome assemblers [50,51], but became popular as a multi-genome data structure as well [52,53]. It efficiently compresses multiple genomes by storing each *k*-mer of constituent genomes only once. Figure 1.5A illustrates a DBG representing three sequences, where the first two sequences share some *k*-mers but the third sequences shares no *k*-mer with either. Compressing non-branching paths significantly reduces the size of this graph (Figure 1.5B). Considering the reverse-complement *k*-mers in a stranded version of the graph reveals that the third sequence also shares some *k*-mers with the first two sequences, but in reverse direction. Genomic inversions between genomes can be detected in such a stranded graph (Figure 1.5C).

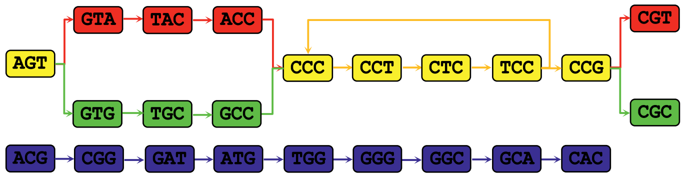


A. Acyclic

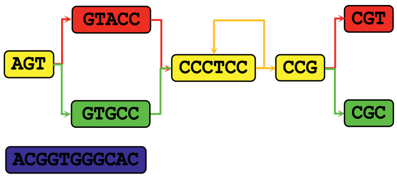


B. Cyclic

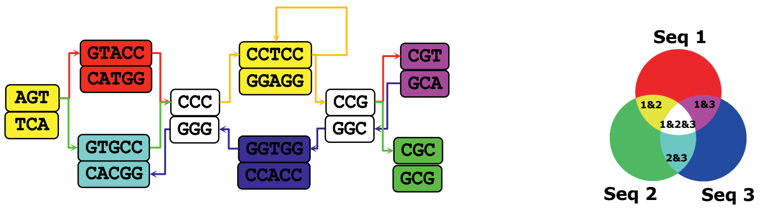
Figure 1.4. A. The acyclic and **B.** cyclic pan-genome graphs for two sequences AGTAGCCCTCCCTCGT (red) and AGTTGCCCTCCCTCGC (green). Yellow nodes are shared between these two sequences. The tandem repeat of CCTC in the longest node in the acyclic graph is represented by a loop in the cyclic graph, however the number of copies needs to be annotated for the repeated part.



A. *De Bruijn* graph



B. Compressed *De Bruijn* graph



C. Stranded *De Bruijn* graph

Figure 1.5. Three varieties of the *De Bruijn* graph (k = 3) for sequences AGTACCCTCCCTCCGT (red), AGTGCCCTCCCTCCGC (green), and ACGGTGGGCAC (blue). Nodes shared between two or all three the sequences are colored as determined in the legend. The number of nodes is much lower in the compressed DBG compared to the original DBG. Reverse-complement of the blue sequence shares several parts with two other sequences which can be captured in a stranded DBG.

1.6 Research objectives and outline of thesis

As outlined above, pan-genomes have received a lot of attention in recent years. However, existing pan-genomic tools are mostly specialized to the human genome and do not scale to large collections of structurally dynamic genomes, such as plant genomes. In this project, we investigate computational methods to compress large sets of related genomes into a pan-genome with basic functionalities for construction, update and exploration of such pan-genomes. As a data representation, we develop a generalized *De Bruijn* graph which scales to thousands of prokaryotic or hundreds eukaryotic genomes. The implemented pan-genomic toolset, PanTools, provides useful functionality to construct and update the pan-genome, add new genomes and annotations, retrieve genomic features/regions, add proteome space, detect homology groups in this space, and finally map short sequencing reads against a pan-genome.

In Chapter 2, we present our pan-genome representation, as well as the construction algorithm and our annotation approach. We introduce a generalized *De Bruijn* graph as the pan-genome data structure which is compressed, bi-directed, localized and indexed for efficiency and applicability. We demonstrate applicability of our pan-genome toolbox, PanTools, to large number of bacterial, fungal and plant genomes. In Chapter 3, we present a method to incorporate proteomes in the pan-genome and detect homology groups in the proteome space, *de novo* and efficiently. We show sensitivity and specificity of the implemented *k*-mer-based approach, demonstrate its scalability to large bacterial, fungal, plant and metazoan proteomes, and show its applicability to proteomes of species at various evolutionary distances. In Chapter 4, we present a *k*-mer-based approach to correct for substitution errors in short-read data. We show that this method increases the horizontal and vertical coverage of read mapping, which are important to improve variant calling and genome assembly, respectively. Finally, Chapter 5 introduces a pan-genome read mapping approach capable of aligning millions of short reads to hundreds of eukaryotic or thousands of prokaryotic genomes, simultaneously. We show that mapping against multiple genomes reduces the number of unmapped reads. We also show how the implemented competitive mapping approach can be effectively used for abundance estimation and binning of metagenomics reads.

References

1. Bybee SM, Bracken-Grissom H, Haynes BD, Hermansen RA, Byers RL, Clement MJ, *et al.* Targeted amplicon sequencing (TAS): a Scalable next-gen approach to multilocus, multitaxa phylogenetics. *Genome Biol Evol*. 2011;3:1312–23.
2. Goodwin S, McPherson JD, McCombie WR. Coming of age: ten years of next-generation sequencing technologies. *Nat Rev Genet*. 2016;17(6):333–51.
3. Lowe TM, Chan PP. tRNAscan-SE On-line: integrating search and context for analysis of transfer RNA genes. *Nucleic Acids Res*. 2016;44(W1):W54–7.
4. Agarwal V, Bell GW, Nam J-W, Bartel DP. Predicting effective microRNA target sites in mammalian mRNAs. *Elife*. 2015;4:e05005.
5. Bailey TL, Boden M, Buske FA, Frith M, Grant CE, Clementi L, *et al.* MEME SUITE: tools for motif discovery and searching. *Nucleic Acids Res*. 2009;37:W202–8.
6. O'Leary NA, Wright MW, Brister JR, Ciufo S, Haddad D, McVeigh R, *et al.* Reference sequence (RefSeq) database at NCBI: current status, taxonomic expansion, and functional annotation. *Nucleic Acids Res*. 2016;44(D1):D733–45.
7. Deschamps S, Zhang Y, Llaca V, Ye L, Sanyal A, King M, *et al.* A chromosome-scale assembly of the sorghum genome using nanopore sequencing and optical mapping. *Nat Commun*. 2018;9(1):4844.
8. Jain M, Koren S, Miga KH, Quick J, Rand AC, Sasani TA, *et al.* Nanopore sequencing and assembly of a human genome with ultra-long reads. *Nat Biotechnol*. 2018;36(4):338–45.
9. Koren S, Rhie A, Walenz BP, Dilthey AT, Bickhart DM, Kingan SB, *et al. De novo* assembly of haplotype-resolved genomes with trio binning. *Nat Biotechnol*. 2018;36(12):1174–82.
10. Parfrey LW, Lahr DJG, Katz LA. The Dynamic nature of eukaryotic genomes. *Mol Biol Evol*. 2008;25(4):787–94.
11. Earl D, Nguyen N, Hickey G, Harris RS, Fitzgerald S, Beal K, *et al.* Alignathon: a competitive assessment of whole-genome alignment methods. *Genome Res*. 2014;24(12):2077–89.
12. de Crécy-Lagard V, Hanson A. Comparative genomics. In: Brenner's Encyclopedia of Genetics. *Elsevier*; 2013. p. 102–5.
13. Florea L. EnteriX 2003: visualization tools for genome alignments of Enterobacteriaceae. *Nucleic Acids Res*. 2003;31(13):3527–32.
14. Frazer KA, Pachter L, Poliakov A, Rubin EM, Dubchak I. VISTA: computational tools for comparative genomics. *Nucleic Acids Res*. 2004;32(Web Server):W273–9.
15. Haussler M, Zweig AS, Tyner C, Speir ML, Rosenbloom KR, Raney BJ, *et al.* The UCSC genome browser database: 2019 update. *Nucleic Acids Res*. 2019;47(D1):D853–8.
16. Cunningham F, Achuthan P, Akanni W, Allen J, Amode MR, Armean IM, *et al.* Ensembl 2019. *Nucleic Acids Res*. 2019;47(D1):D745–51.
17. Tekaia F. Inferring orthologs: open questions and perspectives. *Genomics Insights*. 2016;9:17–28.
18. Marschall T, Marz M, Abeel T, Dijkstra L, Dutilh BE, Ghaffaari A, *et al*. Computational pan-genomics: Status, promises and challenges. *Brief Bioinform*. 2018;19(1):118–35.
19. PubMed - NCBI. https://www.ncbi.nlm.nih.gov/pubmed. 2020; Apr 21.
20. Tettelin H, Masignani V, Cieslewicz MJ, Donati C, Medini D, Ward NL, *et al.* Genome analysis of multiple pathogenic isolates of *Streptococcus agalactiae*: Implications for the microbial "pan-genome". *Proc Natl Acad Sci*. 2005;102(39):13950–5.
21. Marcus S, Lee H, Schatz MC. SplitMEM: A graphical algorithm for pan-genome analysis with suffix skips. *Bioinformatics*. 2014;30(24):3476–83.
22. Baier U, Beller T, Ohlebusch E. Graphical pan-genome analysis with compressed suffix trees and the Burrows–Wheeler transform. *Bioinformatics*. 2016;32(4):497–504.
23. Jin M, Liu H, He C, Fu J, Xiao Y, Wang Y, *et al.* Maize pan-transcriptome provides novel insights into genome complexity and quantitative trait variation. *Sci Rep*. 2016;6(1):18936.
24. Baaijens J, Van der Roest B, Köster J, Stougie L, Schönhuth A. Full-length *de novo* viral quasispecies assembly through variation graph construction. *bioRxiv*. 2018;287177.
25. Huang S, Zhang S, Jiao N, Chen F. Comparative genomic and phylogenomic analyses reveal a conserved core genome shared by Estuarine and Oceanic Cyanopodoviruses. *PLoS One*. 2015;10(11):e0142962.
26. Dutilh BE, Thompson CC, Vicente ACP, Marin MA, Lee C, Silva GGZ, *et al.* Comparative genomics of 274 *Vibrio cholerae* genomes reveals mobile functions structuring three niche dimensions. *BMC Genomics*. 2014;15(1):654.
27. Kim Y, Koh I, Young Lim M, Chung W-H, Rho M. Pan-genome analysis of Bacillus for microbiome profiling. *Sci Rep*. 2017;7(1):10984.
28. Delmont TO, Eren AM. Linking pangenomes and metagenomes: the Prochlorococcus metapangenome. *PeerJ*. 2018;6:e4320.

29. Dilthey A, Cox C, Iqbal Z, Nelson MR, McVean G. Improved genome inference in the MHC using a population reference graph. *Nat Genet*. 2015;47(6):682–8.
30. Wandelt S, Starlinger J, Bux M, Leser U. RCSI. *Proc VLDB Endowment*. 2014;6(13):1534–45.
31. Huang L, Popic V, Batzoglou S. Short read alignment with populations of genomes. *Bioinformatics*. 2013;29(13):i361–70.
32. Danek A, Deorowicz S, Grabowski S. Indexes of large genome collections on a PC. *PLoS One*. 2014;9(10):e109384.
33. Rahn R, Weese D, Reinert K. Journaled string tree-a scalable data structure for analyzing thousands of similar genomes on your laptop. *Bioinformatics*. 2014;30(24):3499–505.
34. Eggertsson HP, Jonsson H, Kristmundsdottir S, Hjartarson E, Kehr B, Masson G, *et al.* Graphtyper enables population-scale genotyping using pangenome graphs. *Nat Genet*. 2017;49(11):1654–60.
35. Lefranc M-P, Giudicelli V, Ginestoux C, Jabado-Michaloud J, Folch G, Bellahcene F, *et al.* IMGT(R), the international ImMunoGeneTics information system(R). *Nucleic Acids Res*. 2009;37:D1006–12.
36. Garrison E, Sirén J, Novak AM, Hickey G, Eizenga JM, Dawson ET, *et al.* Variation graph toolkit improves read mapping by representing genetic variation in the reference. *Nat Biotechnol*. 2018;36(9):875–9.
37. Navarro G, Mäkinen V. Compressed full-text indexes. *ACM Comput Surv*. 2007;39(1):2.
38. Sirén J, Välimäki N, Mäkinen V. Indexing finite language representation of population genotypes. *LNCS*. 2011. 6833:270–81.
39. Valenzuela D, Norri T, Välimäki N, Pitkänen E, Mäkinen V. Towards pan-genome read alignment to improve variation calling. *BMC Genomics*. 2018;19(S2):87.
40. Valenzuela D, Mäkinen V. CHIC: a short read aligner for pan-genomic references. *bioRxiv*. 2017.
41. Nguyen ND, Mirarab S, Kumar K, Warnow T. Ultra-large alignments using phylogeny-aware profiles. *Genome Biol*. 2015;16(1):124.
42. Price MN, Dehal PS, Arkin AP. FastTree 2 – approximately maximum-likelihood trees for large alignments. *PLoS One*. 2010;5(3):e9490.
43. Darling ACE. Mauve: multiple alignment of conserved genomic sequence with rearrangements. *Genome Res*. 2004;14(7):1394–403.
44. Lee C, Grasso C, Sharlow MF. Multiple sequence alignment using partial order graphs. *Bioinformatics*. 2002;18(3):452–64.
45. Raphael B, Zhi D, Tang H, Pevzner P. A novel method for multiple alignment of sequences with repeated and shuffled elements. *Genome Res*. 2004;14(11):2336–46.
46. Paten B, Earl D, Nguyen N, Diekhans M, Zerbino D, Haussler D. Cactus: algorithms for genome multiple sequence alignment. *Genome Res*. 2011;21(9):1512–28.
47. Darling AE, Mau B, Perna NT. progressiveMauve: multiple genome alignment with gene gain, loss and rearrangement. *PLoS One*. 2010;5(6):e11147.
48. Jandrasits C, Dabrowski PW, Fuchs S, Renard BY. Seq-seq-pan: building a computational pan-genome data structure on whole genome alignment. *BMC Genomics*. 2018;19(1):47.
49. Iqbal Z, Caccamo M, Turner I, Flicek P, McVean G. *De novo* assembly and genotyping of variants using colored De Bruijn graphs. *Nat Genet*. 2012;44(2):226–32.
50. Pevzner PA, Tang H, Waterman MS. An Eulerian path approach to DNA fragment assembly. *Proc Natl Acad Sci*. 2002;98(17):9748–53.
51. Zerbino DR, Birney E. Velvet: Algorithms for *de novo* short read assembly using De Bruijn graphs. *Genome Res*. 2008;18(5):821–9.
52. Minkin I, Pham S, Medvedev P. TwoPaCo: an efficient algorithm to build the compacted De Bruijn graph from many complete genomes. *Bioinformatics*. 2016;33(24):4024–32.
53. Beller T, Ohlebusch E. A representation of a compressed De Bruijn graph for pan-genome analysis that enables search. *Algorithms Mol Biol*. 2016;11(1):20.

Chapter 2

PanTools: representation, storage and exploration of pan-genomic data

This chapter is published as:
Sheikhizadeh S, Schranz ME, Akdel M, de Ridder D, Smit S. PanTools: representation, storage and exploration of pan-genomic data. *Bioinformatics*. 2016;32(17):487–93.
DOI: 10.1093/bioinformatics/btw455

Abstract

Next-generation sequencing technology is generating a wealth of highly similar genome sequences for many species, paving the way for a transition from single-genome to pan-genome analyses. Accordingly, genomics research is going to switch from reference-centric to pan-genomic approaches. We define the pan-genome as a comprehensive representation of multiple annotated genomes, facilitating analyses on the similarity and divergence of the constituent genomes at the nucleotide, gene and genome structure level. Current pan-genomic approaches do not thoroughly address scalability, functionality and usability. We introduce a generalized De Bruijn graph as a pan-genome representation, as well as an online algorithm to construct it. This representation is stored in a Neo4j graph database, which makes our approach scalable to large eukaryotic genomes. Besides the construction algorithm, our software package, called PanTools, currently provides functionality for annotating pan-genomes, adding sequences, grouping genes, retrieving gene sequences or genomic regions, reconstructing genomes and comparing and querying pan-genomes. We demonstrate the performance of the tool using datasets of 62 *E. coli* genomes, 93 yeast genomes and 19 *Arabidopsis thaliana* genomes. The Java implementation of PanTools is publicly available at http://www.bif.wur.nl.

2.1 Introduction

Since the assembly of the first bacterial genome in 1995 [1], the concept of a reference genome has been the cornerstone for gene discovery, functional analysis and comparative genomics. Reference genomes are typically linear sequence representations that facilitate genome browsing and sequence-based analyses. In recent years, large genome projects, such as the 150 tomato genomes project [2] and the 3000 rice genomes project [3], have led to a deluge of data. Thus, many species and phylogenetic groups are no longer represented by a single reference genome but by numerous related genomes. In this situation, analyzing hundreds of genomes by individual comparison to a single reference genome becomes inefficient and misses genomic content not present in the reference, while ignoring the availability of the other near-complete genomes. Likewise, pair-wise comparison of hundreds of linear genomes is also far from practical. Hence, to capitalize on the genomic diversity in large collections of genomes, we need to transition from a reference-centric approach to a pan-genome approach.

Originally, the term pan-genome has been used to describe the totality of genes found in a species or phylogenetic clade in order to classify specific genes as either core or dispensable [4]. More recent conceptions of the pan-genome are defined at the sequence level, compressing multiple genomes into a (compressed) De Bruijn graph (DBG) using additional data structures such as a suffix tree [5], FM-index [6], Burrows–Wheeler transform [7] or Bloom filter trie [8]. Accordingly, here we define the pan-genome as a comprehensive representation of multiple annotated genomes, facilitating analyses on the content and organization of the constituent genomes.

Aiming to replace linear genome representations, a computational pan-genome solution should contain annotations, be mutable (to incorporate novel genomes), allow long-term storage and be usable for comparative genomics. The storage property is especially important when working on large eukaryotic genomes, where in-memory solutions are no longer sufficient. Being able to update a pan-genome is essential as the rate at which genomes are produced will only increase in the future. Existing pan-genome approaches fulfill only part of these requirements.

To overcome these limitations and work towards a computational pan-genome approach that allows to incorporate these desirable features, we developed an algorithm to condense multiple annotated genome sequences into a single representation. As in other approaches, the core of our pan-genome is a compressed De Bruijn graph. What differentiates our method is that we construct the pan-genome in a Neo4j graph database [9], which scales to arbitrary graph sizes and thus allows for the analysis of large collections of complex eukaryotic genomes. Our pan-genome graph is created using an online algorithm that, like current methods, has a runtime linear in the total sequence length. Besides construction, we provide useful functionality for annotating pan-genomes, grouping genes, retrieving sequences and comparing pan-genomes. The pan-genome is stored on disk and new genomes or annotated features can be added. We have implemented a stand-alone command-line Java application, called PanTools, for the representation, storage and exploration of pan-genomic data. The software is publicly available on http://www.bif.wur.nl.

This article details our algorithm for pan-genome construction and discusses its functionality, performance and applications. Primarily we explain the data structure, the construction algorithm, and the implemented storage method. In addition, we address running time, memory usage and scaling behavior of PanTools. We end by illustrating possible applications that can be developed using our pan-genome representation as a foundation.

2.2 Methods

2.2.1 Overview

In De Bruijn graphs (DBGs), nodes correspond to unique *k*-mers (words) in the input sequences, and edges connect nodes whose words overlap by *k*–1 nucleotides. Storing only one copy of each word, DBGs efficiently compress the input sequences. However, the precision in detecting sequence similarities depends on the choice of *k*. DBGs have been effectively employed for sequence applications such as *de novo* assembly [10,11], *de novo* repeat classification [12], genotyping [13], synteny block finding [14] and, more recently, for pan-genome representation [5-8].

As in some other existing approaches, the core of our pan-genome is a compressed DBG, which is generalized through a number of key properties to make it efficient and applicable to real data. It is:

1. *Compressed*, to preserve space and allow efficient traversal. Compressing non-branching paths also improves the interpretability of the topology which is important for mining structures in the graph, for example, using the graph database query language, Cypher [15].

2. *Bi-directed*, to allow reverse complement sequences to be stored in the same nodes of the graph. This property slightly reduces the size of the graph, but most importantly makes it applicable to double-stranded data, for instance to detect inversions between genomes.

3. *Localized*, to store the genomic positions at which each forward or reverse sequence of a node appears.

4. *Indexed*, by all canonical *k*-mers of the dataset, to provide quick access to the node where a given *k*-mer occurs. A *k*-mer is called canonical if it is smaller than its reverse complement, lexicographically.

5. *General*, to allow storing ambiguous genomic regions in the pan-genome, which are widespread in real datasets.

The most straightforward way of constructing a compressed DBG is to build the original uncompressed DBG and compress non-branching paths. However, branch compression in a graph with billions of nodes requires an amount of memory only found in high-end machines. Therefore, efficient construction methods like those proposed in [6] and [7] create a compressed DBG directly from input sequences. Below, we describe an alternative direct method for construction of the compressed DBG which is optimized for disk-based storage in a graph database. It is an online algorithm, i.e. the graph is updated as soon as the next *k*-mer of the input is scanned. This property enables us to cumulatively add new genomes to the pan-genome over time.

2.2.2 Data structures

The pan-genome is represented as a bi-directed graph, with two-sided nodes [forward (F) and reverse (R)] corresponding to the nucleotide sequence and its reverse complement. As a result, there are four types of edges (FF, FR, RF and RR) depending on the sides that are connected. Figure 2.1 illustrates a pan-genome graph (*k* = 3) of two genomes, each with one sequence, white and black, with shared parts colored gray. In this picture, circles, called sequence nodes, locate the nodes of the pan-genome where a sequence starts and ends, rectangles represent normal nodes and the rounded rectangle is an instance of a so-called degenerate node representing the ambiguous region in the first sequence. Ambiguous regions are sub-sequences in which all consecutive *k*-mers contain one or more ambiguous bases. In this example, the first sequence contains an R (a purine, i.e. an A or G), which has been stored in degenerate node 2. Degenerate nodes save the connectivity of the paths that input sequences take through the graph and facilitate the reconstruction of the constituent genomes or genomic regions. For simplicity, from here on we use the term 'node' as shorthand for normal nodes. In the current implementation, we also use some other types of nodes, which will be introduced in the Section 3. In Figure 2.1, all edges are of type FF except for the loop over node 4 and the edge between nodes 10 and 1, which both are of type FR (determined by the orientation of the arrows on the edge).

Table 2.1 lists the coordinates stored in each node. A coordinate determines the genomes where the sequence of the node occurs, the position at which it occurs, and whether the forward or reverse sequence occurs. As each genome may contain several sequences (chromosomes, contigs etc.) coordinates are represented by three numbers: genome, sequence and position. For example, in node 1, the forward sequence, AAA, occurs in genome 1, sequence 1 at positions 0 and 1, and its reverse sequence, TTT, occurs in genome 2, sequence 1 at position 8.

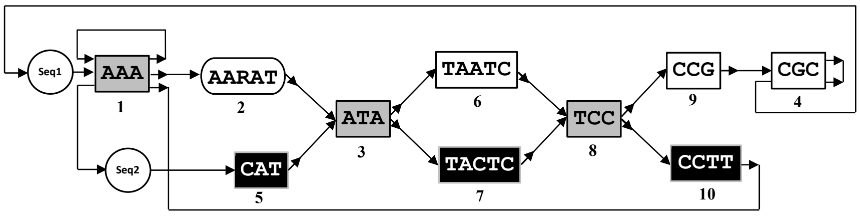


Figure 2.1. A pan-genome graph (*k* = 3) for two sequences, AAAARATAATCCGCG (in white) and CATACTCCTTT (in black). Shared nodes are gray. Rectangle: normal node, rounded rectangle: degenerate node, circle: sequence node. White sequence starts with a forward-forward loop over AAA followed by a degenerate node. Nodes 3, 6,7 and 8 form a bubble structure representing a A-C SNP between two sequences. Node 4 is an example of a forward-reverse loop. Sequence black ends with TTT captured by a cycle back to the reverse strand of Node 1.

Table 2.1. Node properties for the graph in Figure 2.1. Coordinates are given as (orientation:genome:sequence:position(s)). Orientation: F = forward, R = reverse. Positions are 0-based, for example, Node 7 occurs in the first sequence of the second genome at position 2 which is actually the third position.

Node	Sequence	Coordinates	Node	Sequence	Coordinates
1	AAA	F:1:1:0,1; R:2:1:8	6	TAATC	F:1:1:6
2	AARAT	F:1:1:2	7	TACTC	F:2:1:2
3	ATA	F:1:1:5; F:2:1:1	8	TCC	F:1:1:9; F:2:1:5
4	CGC	F:1:1:11; R:1:1:12	9	CCG	F:1:1:10
5	CAT	F:2:1:0	10	CCTT	F:2:1:6

DBGs have proven useful to visualize short variations between sequences, as they create various known substructures in the pan-genome graph, a fact well-known from (co)assembly [10,13,16]. For example in Figure 2.1, the shared sequence ATA(C/A)TCC, with one SNP in the middle, forms a bubble in the center of the graph (induced by nodes 3, 6, 7 and 8). Moreover, the bi-directed pan-genome allows for detecting inversions. For instance, the subsequence TTT at the end of the second sequence can be detected as an inverted translocation of AAA at the start of the first sequence, by comparing the orientations of the coordinates stored in node 1.

Locating *k*-mers within the pan-genome is an essential operation for construction of the pan-genome and facilitates applications such as sequence retrieval, read mapping and sequence alignment. For these reasons, the pan-genome graph is accompanied by an ordered *k*-mer index which quickly locates each canonical *k*-mer by giving the node number, format (canonical or non-canonical) and relative position of its occurrence in the node (note that each *k*-mer occurs just once in the graph). The format of the *k*-mer at its first visit is stored and used by the construction algorithm. Table 2.1 shows the *k*-mer index of the example graph in Figure 2.1. Positions are expressed left to right, starting at 0.

Table 2.2. The *k*-mer index for the graph in Figure 2.1. Pointers are given as (node:format:position). For format, C = canonical, N = non-canonical. Since in a DBG every *k*-mer occurs only once, they can be pinpointed by unique pointers, for example during construction of the graph *k*-mer AAT has been visited in Node 6, position 1 in its canonical form (AAT).

k-mer	Pointer	*k*-mer	Pointer	*k*-mer	Pointer
AAA	1:C:0	ATA	3:C:0	CGC	4:C:0
AAG	10:N:1	ATC	6:C:2	CTC	7:C:2
AAT	6:C:1	GGA	8:N:0	GTA	7:N:0
ACT	7:C:1	ATG	5:N:0	TAA	6:C:0
AGG	10:N:2	CCG	9:C:0		

2.2.3 Online construction of the pan-genome graph

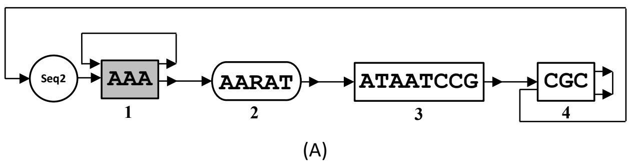
We employ KMC2 [17] to build a *k*-mer index. Our construction algorithm (Algorithm 1) is then based on four elementary operations: **Create**, **Extend**, **Follow** and **Split**. In the main algorithm, sequences are scanned in turn, and each subsequent *k*-mer is looked up in the index. If a *k*-mer is not visited yet in either orientation, a new node of length *k* is added to the graph by the function **Create**, which is then **Extend**ed until we encounter a previously encountered *k*-mer. We locate a previously visited *k*-mer in the graph and, depending on the orientation in which it appears in a node, one of the **Follow-forward** or **Follow-reverse** operations will be performed. Both of these operations take advantage of the **Split** function to divide a node into two, taking care of the bookkeeping.

Figure 2.2A illustrates how these four simple operations work when the first five 3-mers of the black sequence (CATACTCCTTT) are added to the pan-genome of the white sequence (Figure 2.2A). The first 3-mer, CAT, creates node 5 as it is not available in the white sequence. ATA appears in node 3, so it is followed to reach the next 3-mer of the node, TAA, which differs from the next 3-mer in the black sequence, TAC. Node 3 is therefore split, node 7 is initialized with TAC and is extended with ACT and CTC as these two 3-mers have not been visited before. Continuing this process, the next 3-mer, TCC, appearing in the middle of the node 6, results in another split to enter this node.

Algorithm 2.1. Pseudo-code of the construction algorithm. The algorithm simply scans the genomes one after the other and keeps track of the *k*-mers to decide to create, extend or follow a node. As a result, the structure of the graph is updated immediately after reading the next *k*-mer.

```
Data: one or more Genomes each containing one or more Sequences, k-mer length k
Result: a pan-genome Graph and k-mer Index

Initialize empty Graph, build Index of all canonical k-mers;
for g = 1 .. number of Genomes do
    for s = 1 .. number of Sequences in g do
        position = 0;
        while position < length(Genomes[g].Sequences[s]) − k + 1 do
            kmer = Genomes[g].Sequences[s][position . . . position + k − 1];
            if kmer is visited for the first time then
                n = Create(kmer);
                Extend(n);
            else
                n = node where kmer occurs;
                if kmer visited in this orientation then
                    Follow-forward(n, kmer);
                else
                    Follow-reverse(n, kmer);
                end
            end
        end
    end
end
```



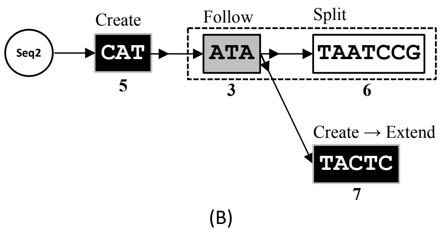


Figure 2.2. Four basic operations of the construction algorithm (Create, Extend, Follow, Split). **A.** Pan-genome of sequence AAAARATAATCCGCG. **B.** Adding the first five 3-mers of the sequence CATACTCCTTT to the pan-genome of AAAARATAATCCGCG, results in creating node 5, splitting node 3 and creating and extending node 7.

2.2.4 Choice of k

k-mers are the building blocks of every DBG, determining to a large extent its properties. Chikhi and Medvedev [18] describe that in DBG-based assemblers, the best choice of k is the one that provides the largest number of distinct, non-erroneous genomic k-mers to the assembler. When using a (compressed) DBG to represent a pan-genome graph, although it is not easy to define an optimal value for k, it is possible to suggest a lower bound to avoid tangling the graph. Short k-mers increase the chance of single nodes representing unrelated subsequences within and between different genomes, making the graph tangled and hard to interpret. On the other hand, selecting a large value for k will decrease connectivity and lead to distinct node sets representing each genome or at least will obscure small-scale variation, e.g. SNPs less than k bases apart, which is not desirable for a pan-genome representation. There is thus a trade-off between the specificity of k-mers and the resolution of the pan-genome variation in the graph structure. We therefore choose the smallest possible value of k which, with high probability, does not lead to the collapse of unrelated sequences into single nodes. Call two k-mers identical if their canonical form is the same, then the probability of a k-mer being identical to another k-mer is $2\alpha^k$, where α is the probability of a single identical symbol. For nucleotide sequences, we can set $\alpha = 0.25$ or take actual frequencies of occurrence into account. Given a set of n random k-mers, we are interested in the number of non-unique k-mers in this set as a function of k. For odd values of k, the probability that an arbitrary k-mer is different from all others in this random set is $(1 - 2\alpha^k)^{n-1}$ and the probability that a k-mer occurs at least twice (i.e. is non-unique) is$1 - (1 - 2\alpha^k)^{n-1}$. The ex- pected number of non-unique k-mers is then $n[1 - (1 - 2\alpha^k)^{n-1}]$. By setting this less than 1 and using the limit definition of the exponential function, we can derive a lower bound for k:

$$k > log_\alpha \frac{-ln\left(1-\frac{1}{n}\right)}{2(n-1)} \tag{1}$$

Figure 2.3 shows the estimated number of non-unique k-mers for two different values of α. It clearly shows that for a set of 100M random k-mers, this number drops abruptly and exponentially when we increase k. For $\alpha = 0.25$, up to $k = 11$, almost all k-mers are non-unique; for larger k, some k-mers start to be unique; and at $k = 27$ all k-mers are unique. In contrast to random data, real genomic sequence data contains meaningfully identical k-mers, which makes the righthand side of inequality (1) a (less tight) lower bound on k in practice. Using this estimate and a $\alpha = 0.3$, the lower bound of k for the pan-genome of 1000 human genomes would be 49. The maximum value of k is limited to 256 by KMC2, which is far larger than what is useful for pan-genome construction.

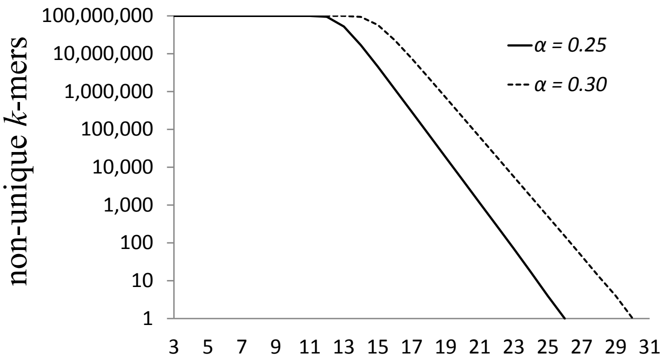


Figure 2.3. The number of non-unique k-mers in a set of 100M random k-mers, as a function of k. The number of unique random k-mers grows exponentially as size of k increase.

2.2.5 Implementation

A major goal of our pan-genome project is to allow storage and exploration of variable pan-genomes for large collections of crop genomes, for example maize, rice and tomato [2,3,19]. To achieve this goal, the pan-genome graph and accompanying data structures are not maintained in memory, but in memory-mapped databases. The advantage is that the operating system takes care of reading and writing required chunks of files (pages) and the application just interacts with memory, which results in very fast I/O operations. Furthermore, memory mapped databases can be shared between different processes, which paves the way for developing multi-threaded pan-genomic applications in future. It should be noted that interacting with large files using disproportionately small amounts of memory increases the number of page faults, drastically reducing performance. Also, like any other disk-based program, the performance of PanTools depends on disk speed. Thus, to achieve the best performance we suggest to have a dedicated machine, preferably with a RAM drive or a solid-state drive (SSD).

We use three memory mapped databases: the Neo4j graph database, the index database and the genome database. The graph database contains information about nodes, relationships (edges) and their properties. The index database is a set of files representing the k-mer index, and the genome database, which is only used during the construction of the pan-genome, stores the compressed input sequences. The only data structure which is kept in memory is the small database of prefixes produced by KMC2. As a result, the memory requirement of the construction algorithm (and any other application which needs the index database) stays independent of the size of data. This enables us to create and explore graphs with millions of nodes using a constant, limited amount of memory, say 8GB.

2.3 Results

In this section, we demonstrate the functionality and performance of our pan-genome approach, and discuss its application in comparative genomics. To this end, we constructed pan-genomes of two HIV-1 strains (AF069671.1 and AF413987.1), 62 Escherichia coli genomes [5], 93 yeast genomes [20] and 19 Arabidopsis thaliana genomes [21]. Our experiments were conducted on a Linux server (Ubuntu 14.04) with an Intel® Xeon® X5660@2.8GHz, with 24 logical cores, 64GB RAM and a 32GB RAM disk.

2.3.1 Functionality

The current implementation of PanTools provides the following functionality:
1. *Build*, given a number of FASTA files, constructs the pan-genome.
2. *Add*, adds one or more new genomes to a given pan-genome.
3. *Annotate*, given one or more GFF files, adds gene nodes to the graph corresponding to the annotated genes.
4. *Group*, adds group nodes to the graph linking genes by some criterion, for instance orthology or name.
5. *Retrieve*, extracts the sequence of specified genes or genomic regions from the pan-genome graph.
6. *Reconstruct*, reconstructs some/all of the constituent genomes from a given pan-genome.
7. *Compare*, compares the topology of two existing pan-genomes.
8. *Query*, gives a command prompt to run Cypher queries and receive the results.

To demonstrate the Build, Annotate and Group functionality, we constructed the pan-genome of two HIV-1 strains, annotated the genomes and grouped the homologous genes. Figure 2.4 visualizes the graph, with different types of nodes indicated by different colors explained in the caption. The pan-genome node points to two genome nodes (1 and 2) each containing a single sequence (1_1 and 2_1) (Figure 2.4A). Two instances of *vpu* genes of equal length (246 nt) have been grouped together by a *vpu* group node (Figure 2.4B). One of these genes begins and ends at a node of length 437, the other at another node of length 443. The exact position where each gene starts and ends in these nodes is stored as a property in edges labeled begin and end. It is also clear in Figure 2.4C that the two homologous *pol* genes have different lengths, 3012 and 3006, and begin and end in distinct nodes. The pair of nodes where these genes begin and end belong to a bubble, which indicates that there is some variation (indel or SNP) between them. The *pol* genes are rather variable, as they are represented by the entire chain of bubbles at the top-right of the graph. SNPs can be distinguished based on the fact that the length of both branching nodes equals 2*k*-1 (here 63), which means nodes differ in a single nucleotide in the middle of their sequence (Figure 2.4D). Each edge has a three-letter label, starting with the two nucleotides which appear at the borders of the *k*-1 overlap of the nodes and ending with a number in the range 0–3 which codes for the four types of edges. The incoming and outgoing edges of the top node have been labeled CA0 and AG0, respectively, while those of the node at the bottom are labeled CG0 and GG0, indicating an A-G SNP.

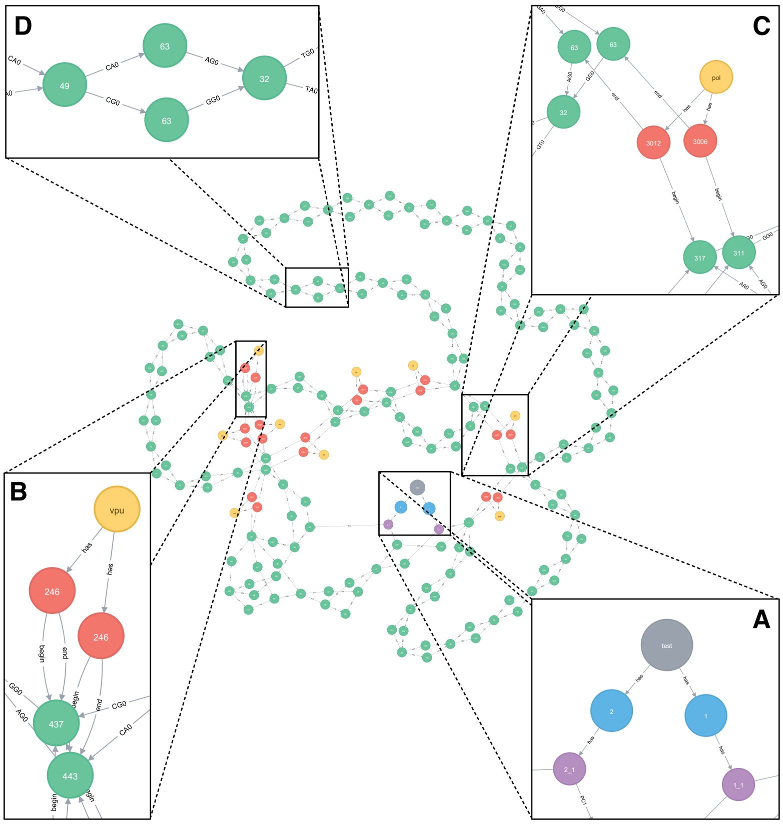


Figure 2.4. Visualization of the annotated pan-genome (*k* = 32) of two HIV strains, AF069671.1 and AF413987.1, in the Neo4j browser. **A.** The single pan-genome node (gray) points to genome nodes (blue), each pointing to their constituent sequence nodes (purple). **B.C.** Examples of group nodes (yellow) linking genes (red), which point to DBG nodes (green) where they start and end in the pan-genome. **D.** An example of a bubble structure created by an A-G SNP. Plenty of simple and complex bubbles are formed representing the variation of these two genomes.

Our method allows to incrementally adding new genomes to an existing pan-genome. To show this, we constructed a pan-genome of three yeast genomes and iteratively added sets of 10 new yeast genomes. Then, we compared the nine intermediate pan-genomes with those constructed directly from 13, ..., 93 genomes from scratch (called Y13 to Y93). Using the Compare functionality, we observed that pan-genomes containing the same genomes but constructed in different ways (directly or iteratively) were identical, having the same number of *k*-mers, nodes, edges and bases as well as the same properties stored in corresponding nodes. We also verified that changing the order of the genomes in the input dataset does not affect the construction, resulting in isomorphic pan-genomes.

2.3.2 Performance

To verify the resource requirements of the construction algorithm when it scales to larger datasets, we constructed nine *A. thaliana* pan-genomes containing 3, 5, . . ., 19 genomes. Table 2.3 reports properties of the resulting pan-genomes, as well as the resources consumed during the construction. In all experiments, we set k=31.

PanTools created the pan-genome of 19 *A. thaliana* genomes in less than an hour using just above 6 GB of memory. To verify that our method scales to arbitrary genome sizes, we also successfully constructed a pan-genome graph of seven human genomes in a preliminary experiment.

The numbers of k-mers, nodes and edges in the pan-genome grow in a sub-linear fashion with respect to the size of the datasets. The ratio of the number of k-mers to the number of nodes in the Arabidopsis pan-genomes ranges between 22 and 49, which indicates the significant effect of compressing the non-branching paths. However, this ratio declines as more genomes are added to the pan-genome.

Pan-genomes store far fewer base pairs than linear genomes because redundancy is eliminated. Table 2.3 also shows that the number of base pairs stored in the pan-genome representation of 19 *A. thaliana* genomes (A19) is just under one fourth of those stored in the linear genomes. This difference becomes more significant as more genomes are added to the pan-genome.

Table 2.3. Scalability of the program to large pan-genomes of up to 19 Arabidopsis genomes (A3–A19). As expected, the number of k-mers and resource requirements levels off as the number of genomes increase.

	k-mers (M)	Nodes (M)	Degenerate (M)	Edges (M)	Bases (M)	Time (second)	Memory (MB)
A3	143.1	2.9	0.05	4.1	236.8	977	3,578
A5	158.1	4.3	0.09	5.9	294.5	1,346	4,258
A7	168.3	5.3	0.22	7.5	344.8	1,644	4,692
A9	177.4	6.2	0.25	8.7	382.3	1,943	4,865
A11	184.9	6.9	0.29	9.8	414.3	2,284	4,974
A13	189.6	7.5	0.47	10.8	447.2	2,443	5,261
A15	197.4	8.2	0.52	11.9	481	2,828	5,695
A17	203.9	8.9	0.55	12.9	508.9	3,130	5,967
A19	209.5	9.4	0.58	13.7	532.9	3,456	6,254

To compare our construction method with the existing in-memory methods we run the FM-index based algorithm (FMI) presented by Beller and Ohlebusch [6] and the BWT-based algorithm presented by Baier *et al*. [7], on datasets of 62 *E. coli* (E62), 93 yeast (Y93) and 19 *A. thaliana* (A19) genomes; results are presented in Table 2.4. We also tried to run SplitMEM [5], but it ran out of memory even on the smallest dataset in this experiment. The DBGs produced by the FMI- and BWT-based tools are exactly identical and these tools show the same performance. However, their graph contains more nodes than the graph produced by PanTools. The reason is that they do not build a bi-directed DBG and reverse complement sequences are stored in different nodes. The smaller number of edges in our pan-genome is explained by the fact that we store each edge only once, whereas the two other tools store each individual transition. To conclude, besides the various useful features that our disk-based approach provides, the amounts of time and memory it needs are comparable to the best in-memory methods, in particular for larger datasets.

Table 2.4. Comparing PanTools with the two existing tools shows that PanTools requires comparable computational resources as in-memory tools.

	Tool	Nodes (M)	Edges (M)	Time (Second)	Memory (MB)
	FMI-based	1.16	21.5	364	879
E62	BWA-based	1.16	21.5	370	879
	PanTools	1.07	1.43	1,261	2,344
	FMI-based	1.52	79.3	1,432	3,093
Y93	BWA-based	1.52	79.3	1,454	3,093
	PanTools	1.47	2.01	4,827	1,621
	FMI-based	9.9	123.1	2,878	6,044
A19	BWA-based	9.9	123.1	2,750	6,044
	PanTools	9.4	13.7	3,456	6,254

Efficient extraction of genes, genomic regions and genomes from a pan-genome is a key functionality for downstream pan-genome applications. A genomic feature is extracted by traversing the path it takes in the pan-genome, which is determined using the genomic positions of each node, stored as node properties. To examine annotation and retrieval efficiency, we annotated and then retrieved all genes, extracted thousand 1000 nt-long random genomic regions and reconstructed the whole set of constituent genomes in five different yeast pan-genomes; Table 2.5 gives the run-times in milliseconds. PanTools retrieves a gene in around one millisecond. As expected, the average annotation and retrieval times increase slightly as the pan-genome grows.

Table 2.5. Average time (in milliseconds) for annotating one gene, retrieving one gene, retrieving 1 kbp and reconstructing one genome increases slightly with the number of genomes.

	Gene annotation	Gene retrieval	1 kbp retrieval	Genome reconstruction
Y13	2.2	0.3	2.3	3,615
Y33	3.5	0.5	3.4	7,453
Y53	4.8	0.8	4.6	11,639
Y73	6.2	1.2	6.6	15,236
Y93	7.6	1.6	7.9	19,544

2.3.3 Pan-genome applications

Expecting an increasing rate of genome production, pan-genomes should ideally take over the role of linear reference genomes in comparative genomics. This implies that in the future, we will analyze novel genomic data with respect to all genomes in the pan-genome at once, move from pairwise genome comparisons to multiple genome comparisons at once, and browse pan-genomes rather than reference genomes. Another important application is variation detection in pan-genomes, including single-nucleotide polymorphisms, structural variation, copy-number variation, synteny, transposon-insertion polymorphisms, etc. A prerequisite for such high-level applications is a solid data structure and construction algorithm, as presented in this article, possibly enhanced with application-specific indices.

A basic application, underlying genome browsing and several analyses, is fast retrieval of genes and genomic regions of interest and assess the variations. To demonstrate this feature, we retrieved all instances of the well-known *FRIGIDA* gene from the pan-genome of 19 *A. thaliana* genomes (A19). This gene encodes a major determinant of natural variation in flowering time and its allelic diversity among these 19 accessions is of interest to plant biologists [21]. There is significant allelic variation in this gene, as shown in Figure 2.5. The variants differ at the start of the gene, but are identical in the last 106 nucleotides. We can distinguish types of variants by assessing the 'bubbles' in the graph: the 11 bubbles correspond to 6 SNPs, 1 deletion and 4 mismatches less than *k* nucleotides apart. The exact presentation of genomic variations like these needs to be worked out in more detail, but the example illustrates that we can quickly assess genomic variation in multiple genomes at once, rather than compose the full picture from many pairwise genome comparisons

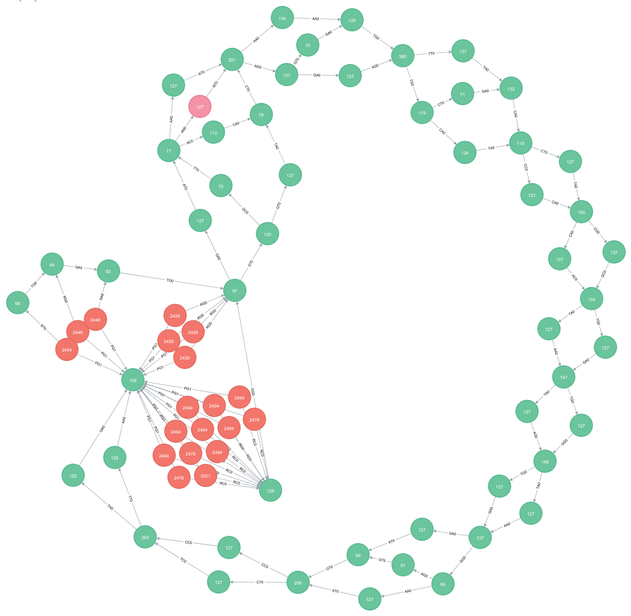


Figure 2.5. Subgraph induced by 19 FRIGIDA genes contains a lot of bubbles representing variability of this gene among different *A. thaliana* accession. Red and green nodes represent genes and nodes, respectively. The pink node is a degenerate node caused by an M symbol in the fifth genome (Edi-0).

Many pan-genome analyses will require searches in the highly connected graph database. We will use the query language of Neo4j, Cypher, to search pan-genomes for specific structures and properties. The efficiency of the queries depends on how constrained the query is. Table 2.6 presents the result of some queries from a pan-genome of three yeast genomes. The first three queries ask for the number of bubbles created by two branching paths with at most 2 (simple bubble), 3 or 4 edges, respectively. As expected, the run-time of such queries increases as the length of the branching path increases. The fourth query returns all sequences that belong to simple SNPs (one node in each branch). The fifth query gives the sequence of two branches of simple bubbles where one branch is a degenerate node. The sixth query returns the occurrence arrays of nodes shared by chromosome 1 of all the genomes, and the seventh one gives the number of nodes that are specific to chromosome 1 of genome 1. These examples show that Cypher supports many different queries, which can be done in reasonable time. Users can run these queries on their own pan-genome using the query command of PanTools.

Table 2.6. Different types of substructures were mined in the pan-genome of three yeast genomes (*k* = 31) using Cypher query language of the Neo4j graph database. Run-time depends on the complexity of the query and the graph.

	Cypher query	Hits	Time (s)
1	match(n:node)–>()–>(m:node)<–()<–(n) return count(*)	120,778	6
2	match(n:node)–>(a)-[*0.2]–>(m:node)<–[*0.2]-(b)<–(n) where a<>b return count(*)	138,346	23
3	match(n:node)–>(a)-[*0.3]–>(m:node)<–[*0.3]-(b)<–(n) where a<>b return count(*)	367,300	89
4	match(n:node)–>(a:node)–>(m:node)<–(b:node)<–(n) where a.length = b.length and a.length = 61 return a.sequence, b.sequence	87,138	12
5	match(n:node)–>(a:node)–>(m:node)<–(b:degenerate)<–(n) return a.sequence, b.sequence	16	0.34
6	match(n:node) where has(n.F1_1) and has(n.F2_1) and has(n.F3_1) return n.F1_1, n.F2_1, n.F3_1	1,273	0.72
7	match(n:node) where has(n.F1_1) and not has(n.F2_1) and not has(n.F3_1) return count(n)	602	0.65

2.4 Conclusion

Thanks to large sequencing efforts, many species or phylogenetic clades are no longer represented by a single reference genome, but by a multitude of genomes. Besides the sequence similarities between related genomes, there may be significant variation in genomic content and organization, which was often the reason to sequence and study them. To deal with this new data challenge, there is an increasing need for new ways of storing and constructing unified representations of large collections of genomes. In addition, we need accompanying algorithms to answer key questions, such as on core and dispensable genes, recurring genetic variants and structural variation, which are cumbersome to address using linear representations of large numbers of genomes. In this article, we have presented PanTools, an implementation of a pan-genome representation based on the Neo4j graph database, focusing on the application to large sets of complex eukaryotic genomes. The program allows for the construction of pan-genome databases of many genomes, and contains extensions such as adding sequences, genes and orthology annotations, using relatively modest computational resources.

PanTools offers a good starting point for developing various pan-genomic applications, such as multi-genome read mapping, pan-genome exploration (visualization, browsing), structure-based variation detection and comparative genomics. To efficiently implement algorithms supporting such analyses it is likely that additional layers of annotation, summaries (e.g. synteny blocks) or different indices will be needed. The current base implementation in Neo4j is adaptable and extensible and offers an excellent foundation for such extensions. In summary, we have presented a first implementation of a pan-genome representation and construction algorithm, which can form the basis of a collection of tools to allow pan-genomes to take over the role of linear reference genomes in genomics.

References

1. Fleischmann RD, Adams MD, White O, Clayton RA, Kirkness EF, Kerlavage AR, *et al*. Whole-genome random sequencing and assembly of *Haemophilus influenzae* Rd. *Science*. 1995;269:496–512.
2. Aflitos S, Schijlen E, De Jong H, De Ridder D, Smit S, Finkers R, *et al*. Exploring genetic variation in the tomato (*Solanum* section *Lycopersicon*) clade by whole-genome sequencing. *Plant J*. 2014;80(1):136–48.
3. Li J-Y, Wang J, Zeigler RS. The 3,000 rice genomes project: new opportunities and challenges for future rice research. *Gigascience*. 2014;3(1):8.
4. Tettelin H, Masignani V, Cieslewicz MJ, Donati C, Medini D, Ward NL, *et al*. Genome analysis of multiple pathogenic isolates of Streptococcus agalactiae: Implications for the microbial "pan-genome". *Proc Natl Acad Sci*. 2005;102(39):13950–5.
5. Marcus S, Lee H, Schatz MC. SplitMEM: A graphical algorithm for pan-genome analysis with suffix skips. *Bioinformatics*. 2014;30(24):3476–83.
6. Beller T, Ohlebusch E. A representation of a compressed De Bruijn graph for pan-genome analysis that enables search. *Algorithms Mol Biol*. 2016;11(1):20.
7. Baier U, Beller T, Ohlebusch E. Graphical pan-genome analysis with compressed suffix trees and the Burrows–Wheeler transform. *Bioinformatics*. 2016;32(4):497–504.
8. Holley G, Wittler R, Stoye J. Bloom filter trie: an alignment-free and reference-free data structure for pan-genome storage. *Algorithms Mol Biol*. 2016;11:3.
9. Have CT, Jensen LJ. Are graph databases ready for bioinformatics? *Bioinformatics*. 2013;29(24):3107–8.
10. Pevzner PA, Tang H, Waterman MS. An Eulerian path approach to DNA fragment assembly. *Proc Natl Acad Sci*. 2002;98(17):9748–53.
11. Zerbino DR, Birney E. Velvet: Algorithms for *de novo* short read assembly using De Bruijn graphs. *Genome Res*. 2008;18(5):821–9.
12. Pevzner PA, Tang H, Tesler G. *De novo* repeat classification and fragment assembly. *Genome Res*. 2004;14(9):1786–96.
13. Iqbal Z, Caccamo M, Turner I, Flicek P, McVean G. *De novo* assembly and genotyping of variants using colored De Bruijn graphs. *Nat Genet*. 2012;44(2):226–32.
14. Minkin I, Patel A, Kolmogorov M, Vyahhi N, Pham S. Sibelia: A scalable and comprehensive synteny block generation tool for closely related microbial genomes. *LNCS*. 2013. p. 215–29.
15. Van Bruggen R. Learning Neo4j. Birmingham: Packt Publishing Ltd; 2014.
16. Nijkamp JF, Pop M, Reinders MJT, De Ridder D. Exploring variation-aware contig graphs for (comparative) metagenomics using MARYGOLD. *Bioinformatics*. 2013;29(22):2826–34.
17. Deorowicz S, Kokot M, Grabowski S, Debudaj-Grabysz A. KMC 2: Fast and resource-frugal k-mer counting. *Bioinformatics*. 2014;31(10):1569–76.
18. Chikhi R, Medvedev P. Informed and automated k-mer size selection for genome assembly. *Bioinformatics*. 2014;30(1):31–7.
19. Chia JM, Song C, Bradbury PJ, Costich D, de Leon N, Doebley J, *et al*. Maize HapMap2 identifies extant variation from a genome in flux. *Nat Genet*. 2012;44(7):803–7.
20. Strope PK, Skelly DA, Kozmin SG, Mahadevan G, Stone EA, Magwene PM, *et al*. The 100-genomes strains, an *S. cerevisiae* resource that illuminates its natural phenotypic and genotypic variation and emergence as an opportunistic pathogen. *Genome Res*. 2015;125(5):762–74.
21. Gan X, Stegle O, Behr J, Steffen JG, Drewe P, Hildebrand KL, *et al*. Multiple reference genomes and transcriptomes for *Arabidopsis thaliana*. *Nature*. 2011;477(7365):419–23.

Chapter 3

Efficient inference of homologs in large eukaryotic pan-proteomes

This chapter is published as:
Sheikhizadeh S, Schranz ME, de Ridder D, Smit S. Efficient inference of homologs in large eukaryotic pan-proteomes. *BMC Bioinformatics*. 2018;19(1):340.
DOI: 10.1186/s12859-018-2362-4.

Abstract

Identification of homologous genes is fundamental to comparative genomics, functional genomics and phylogenomics. Extensive public homology databases are of great value for investigating homology but need to be continually updated to incorporate new sequences. As new sequences are rapidly being generated, there is a need for efficient standalone tools to detect homologs in novel data. To address this, we present a fast method for detecting homology groups across a large number of individuals and/or species. We adopted a k-mer based approach which considerably reduces the number of pairwise protein alignments without sacrificing sensitivity. We demonstrate accuracy, scalability, efficiency and applicability of the presented method for detecting homology in large proteomes of bacteria, fungi, plants and Metazoa. We clearly observed the trade-off between recall and precision in our homology inference. Favoring recall or precision strongly depends on the application. The clustering behavior of our program can be optimized for particular applications by altering a few key parameters. The program is available for public use at https://github.com/sheikhizadeh/pantools as an extension to our pan-genomic analysis tool, PanTools.

3.1 Introduction

Detection of homologous genes (genes that share evolutionary ancestry) is fundamental to comparative genomics, functional genomics and phylogenomics. Homologs inherited from a single gene in the last common ancestor of two species are called orthologs, while those inherited from distinct duplicated genes are called paralogs [1]. Orthologs are usually under selection pressure, which conserves their sequence, structure and function; while paralogs can diverge rapidly and lose their previous functions or achieve completely or partially new functions [2].

With increasingly evolutionary distance and/or increased data-set sizes, there will be greater sets of gene and genome changes, that can complicate orthology inference [3]. Whole-genome and segmental duplications increase genomic content, local and structural mutations lead to gene losses and gains, and horizontal gene transfers mix genomic content between species. As a result, orthology detection is increasingly difficult in higher organisms and across large evolutionary distances. In the presence of gene duplications, orthology is not always a one-to-one relationship but rather can be a one-to-many or even many-to-many relationship [4]. As a consequence, an orthology group may contain not only orthologous pairs, but also pairs of homologs duplicated after the speciation of the two species, so-called in-paralogs. In the rest of this text we therefore use the term homology group instead of orthology group to be more precise.

To date, several databases of homology groups have been established, which need to be continually updated to incorporate new genomes [5-8]. As genomic projects are generating novel data at an unprecedented scale, the analysis of new data means that researchers have to automate the process of inferring homology in their large gene sets. Consequently, in parallel to the static databases there has been a development of standalone tools for automatic detection of homologs [9-11]. Accurate homology detection tools rely on all-pairs comparison of proteins. However, calculating all-pair similarity scores quickly becomes a major computational burden as the number of proteomes increases. As the number of eukaryotic proteomes keeps expanding in the coming years, there is a need for even more efficient homology detection methods.

Here, we present an efficient graph-based approach towards homology detection. This method extends the functionality of our pan-genomic data analysis tool, PanTools [12], which integrates genomes, annotations and proteomes in a single graph database to facilitate comparative studies at the levels of structure, variation and function [13]. The motivation of this study was to detect homology groups *de novo* and efficiently, in large datasets of hundreds of eukaryotic genomes. The presented method scales to large proteome sets while maintaining its accuracy and can be tuned for different application scenarios.

3.2 Methods

We represent a pan-genome by a hierarchy of genome, annotation and proteome layers stored in a Neo4j graph database to connect different types of data (Figure 3.1). The genome layer consists of pan-genome, genome, sequence and nucleotide nodes which contain some essential information about these entities. Nucleotide nodes form the generalized De Bruijn graph [12] which enables the compression and reconstruction of the constituent genomes. The annotation layer, currently, consists of the genomic features like genes, mRNAs, etc. Finally, the proteome layer of the pan-genome is formed by proteins and homology nodes which group the homologous proteins. Before homology detection, first the protein nodes should be stored in a pan-genome graph. Instructions for constructing a pan-genome can be found in the supplementary information. Having the proteins available in the proteome layer of the pan-genome, we take the steps described in Algorithm 1 to cluster them in homology groups.

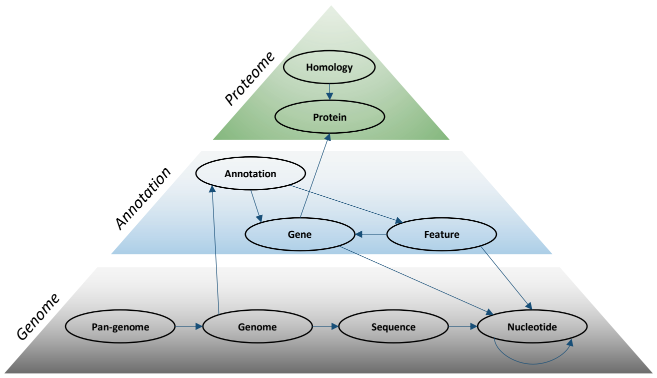


Figure 3.1. PanTools integrates heterogeneous data in a hierarchical pan-genome. The Neo4j graph data model allows to store many properties in the nodes and edges of the graph.

Algorithm 3.1. Homology detection algorithm consist of 5 main stages: hexamerization of sequences, intersection discovery, sequence alignment, similarity component discovery and clustering.

Input: the pan-genome containing the protein nodes
Output: the homology groups
Parameters: ***I***: intersection rate, ***T***: similarity threshold, ***C***: contrast, ***M***: MCL inflation

```
1.   // Hexamerize proteins:
2.   for each protein p,
3.       for each hexamer h of p,
4.           append identifier of p to the list of proteins[h].
     In parallel do A, B and C:
     // What A produces, B consumes. What B produces, C consumes.
     // A. Detect intersections:
5.   for each protein p,
6.       for each hexamer h of p,
7.           append proteins[h] to the list of candidate intersecting proteins ofp, CIP[p].
8.       for each candidate protein c in CIP[p],
9.           if c occurs more than I × min(length of p - 5, length of c - 5) timesin CIP[p]
10.              add intersecting pair (p, c) to the intersection queue IQ
     // B. Calculate similarity scores:
11.  for each intersecting pair (p, c) in IQ,
12.      if the normalized similarity score NSS(p, c) is greater than T,
13.          add (p, c, NSS(p, c)) to the similarity queue SQ
```

```
    // C. Add similarity links to the pan-genome :
14. for each (p, c, s) in SQ,
15.     connect protein p to c by an edge with similarity score s
    In parallel do D, E and F:
    // What D produces, E consumes. What E produces, F consumes.
    // D. Build similarity components:
16. for each protein p,
17.     build the similarity component s_comp through a breadth-first searchstarting from p,
18.     put s_comp in the component queue CQ.
    // E. Split similarity components:
19. for each s_comp in CQ,
20.     for each pair of proteins (p1, p2) in s_comp belonging to species s1 and s2, resp.
21.         calculate the distance dist(s1, s2).
22.         similarity_matrix[p1, p2] = ( similarity[p1, p2] – T + dist(s1, s2)) ^ C
23.     pass the similarity _matrix to MCL algorithm with inflation
24.     add the resulting homology groups h_group to the homology queue HQ,
    // F. Add homology group annotation to the pan-genome:
25. for each homology group h_group in HQ,
26.     create a homology node h_node in the pan-genome
27.     connect h_node to the proteins in h_group
```

First, we extract the hexamers of all proteins and, for each hexamer, keep track of the proteins containing that hexamer (lines 1-4). Then, we find all pairs of intersecting proteins (lines 5-10) and calculate their similarity score by aligning them. Two proteins intersect (Figure 3.2A-B) if the number of hexamers they share is greater than the product of the intersection parameter (I) and the total number of hexamers of the shorter protein. We connect the intersecting proteins with a similarity score greater than the similarity threshold T (lines 11-15) to form the similarity graph (Figure 3.2C). For reasons of efficiency, we have implemented this as three parallel routines A-C, in which B consumes the output of A and C the output of B. A and C employ one working thread and B multiple threads to maximize performance. Next, all the connectivity components of the resulting similarity graph are found using a simple breadth-first search (lines 16-18). This search allows to detect not only the directly connected proteins but also those connected through a path in the graph, the potential distant homologs. Every similarity component is then passed to the MCL (Markov clustering) algorithm [14] to be possibly broken into several homology groups (lines 19-24) (Figure 3.2D). MCL has been frequently employed in homology inference methods [11,15,16]. Finally, the members of each homology group are connected to a single homology node in the graph (lines 25-27).

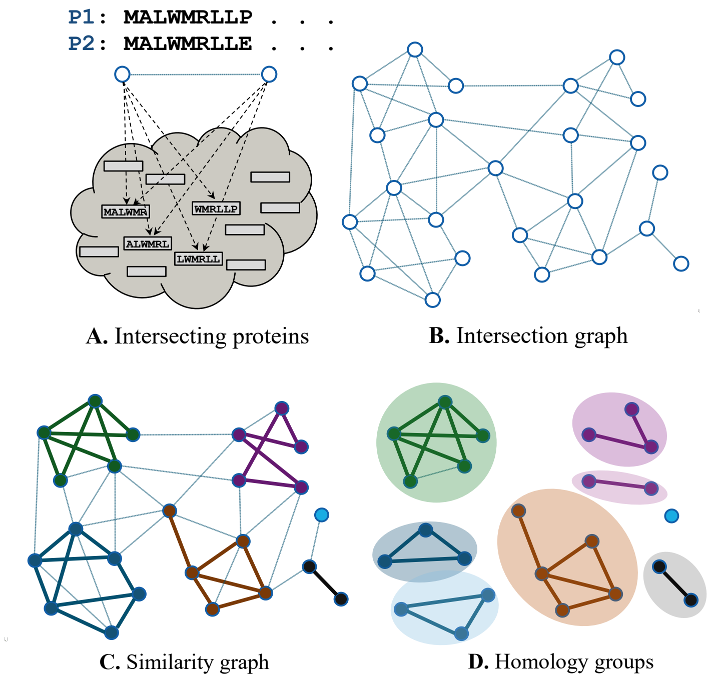


Figure 3.2. A) Two intersecting proteins, P1 and P2 share some hexamers. **B)** The intersection graph is built from interesting pairs of proteins. **C)** The similarity graph consists of similarity components. Each bold edge represents a similarity score greater than the threshold (*T*). **D)** Homology groups are detected in each similarity component by MCL.

3.2.1 Normalizing the raw similarity scores

We compare intersecting pairs of proteins by a Smith-Waterman local alignment algorithm with an affined gap penalty (opening = -10, extension = -1) using the BLOSUM62 (Blocks Substitution Matrix 62) scoring matrix. After calculating the raw similarity scores, we normalize them to be independent of the protein lengths. To this end we divide each raw score by the score achieved by aligning the shorter protein to itself and multiply the result by 100; this way, the normalized similarity scores will always be less than or equal to 100. For the sake of simplicity, we use the term similarity score to refer to the normalized similarity score between pairs of proteins.

3.2.2 Rescaling the similarity scores

The pairwise similarity scores of highly similar homologs, which usually lie in the same similarity component, are very close to each other. This makes it very hard for MCL to detect the underlying substructures in such similarity components. To resolve this problem, we rescale the similarity scores in three different ways (Algorithm 1, line 22). First, we subtract the value *T* from these scores to emphasize small differences for the MCL process.

Furthermore, we would like the clustering to be relatively insensitive to evolutionary sequence divergence. That is, within a similarity component pairs of homologs from two distant species are ideally scored nearly as high as pairs from two closely related species.

To achieve this, in each similarity component we calculate the average distance between each pair of species as 100 minus the average inter-species similarity score and add it to all the similarity scores between those species within the similarity component.

Finally, to increase the contrast between the final similarity scores, before the similarity component is passed to the MCL algorithm, we raise the scores to the power of C, the contrast parameter. This operation is similar to one round of expansion as explained in [14] and was experimentally observed to increase the specificity of the resulting clusters.

3.2.3 Choice of *k*

Short peptide k-mers may occur in many proteins. This raises the number of intersecting proteins which will be aligned, increasing the resource consumption of the program significantly. On the other hand, long k-mers are more specific and decrease the sensitivity of the program in detecting the intersecting pair of proteins, thereby reducing the recall. As a result, we calculate the smallest k value which keeps the probability of random occurrences of a k-mer below a desirable probability p. For peptide sequences $\alpha = 20$, and considering $L = 30{,}000$ the length of the largest known protein [15] and setting $p = 0.001$, the smallest suitable k will be 6 (see supplementary information). Therefore, we chose to use hexamers for detecting the intersections.

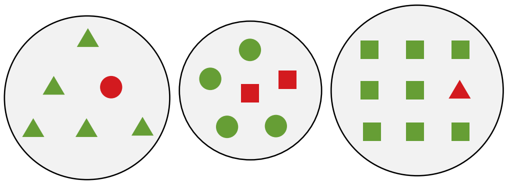
To reduce the memory needs of the program and increase the specificity of the intersections, we ignore extremely abundant hexamers (For example "QQQQQQ" in the yeast datasets), which their frequency exceeds $p \times n + c \times m$, where $p = 0.001$, n is the total number of proteins, $c = 50$ is an *a priori* estimate of the maximum number of occurrences of a hexamer in the proteome of a species, and m is the total number of species (proteomes). Likewise, hexamers with frequency 1 are considered rare and thereby ignored. This filtration notably improves the efficiency and the precision of the method.

3.2.4 Measures of accuracy for evaluation

To evaluate the accuracy of the method, we used the recall, precision and F-score measures as defined previously [16,17] (Figure 3.3). Given a set of real and detected homology groups, for each true homology group, THG, we find the detected homology group, DHG, which has the largest overlap with the THG. Then we consider true positives (tp) as the number of proteins in both THG and DHG, false negatives (fn) as the number of proteins in THG but not in DHG, and false positives (fp) as the number of proteins avilable in DHG but not in THG. Then TP, FP and FN are defined as the summation of the tp's, fp's and fn's over all true homology groups, respectively. Finally, the recall, precision and F-score measures are calculated as follows:

recall = TP / (TP + FN)
precision = TP / (TP + FP)
F-score = 2 × (Recall × Precision) / (Recall + Precision)

Recall represents the ability of the method to put the true homologs together in one group, precision shows its ability to separate the non-homologs, and the F-score is the harmonic mean of these two measures combining them in one. There is always a trade-off between recall and precision, since detecting more TPs often leads to some FPs.



	△	○	□	
tp	5	4	8	**TP = 17**
fn	1	1	2	**FN = 4**
fp	1	2	1	**FP = 4**

Figure 3.3. Proteins of three distinct homology groups are represented as triangles, circles and squares. Green shapes are true positives (tp) which have been assigned to the true group; red shapes are false positives (fp) for the group they have been incorrectly assigned to, and false negatives (fn) for their true group. Having these three factors, recall, precision and F-score of the homology detection can be calculated.

In the following experiments, we need to know the real groups in various datasets to serve as a ground truth for evaluation. For the *S.cerevisiae* datasets and the single *E.coli* dataset, the real groups are defined based on the locus tags of the proteins extracted from the GenBank files (Supplementary spreadsheet datasets.xlsx). For *A.thaliana* datasets the real groups are defined based on the gene identifiers which end with .1, for example AT3G54340.1, which correspond to the first annotated isoform of the genes. For the single Metazoa dataset, we used the identifiers of the 70 protein families of OrthoBench as the real group identifiers.

3.3 Results and discussion

Here, we present results demonstrating the accuracy, scalability, efficiency and applicability of PanTools for detecting homology in large proteomes of bacteria, fungi, plants and Metazoa (Supplementary Table 3.S1). We compare PanTools to the BLAST-based OrthoFinder [16] orthology detector and to DIAMOND-based PanX [18], a pipeline dedicated to microbial data (Supplementary Table 3.S2-5). First, we evaluated the methods on OrthoBench [17], a public benchmark of curated protein families from 12 metazoans. Unfortunately, we were not able to run PanX on this data (M12), as this benchmark only provides the protein sequences but not the gene sequences PanX requires. Next, we tested scalability on 5 datasets of increasing size compiled from 93 *Saccharomyces cerevisiae* strains [19] and 5 datasets compiled from 19 *Arabidopsis thaliana* accessions [20]. Additionally, we compared the performance of PanTools and PanX on a large dataset of 600 *Escherichia coli*; we did not run OrthoFinder, as we estimated it would need ~5000 hours on this dataset. Finally, we studied the effect of evolutionary distance on homology detection using 12 Brassicaceae species proteomes. Experiments were executed on an Ubuntu 14.04 server, Intel® Xeon® X5660@2.8GHz, with 64GB RAM using 16 processing cores and 32GB of RAM disk.

3.3.1 PanTools is adaptable to handle varying degrees of input divergence

PanTools has four main parameters that affect the homology clustering: intersection rate, similarity threshold, contrast and MCL inflation. To examine the general effect of these parameters on the accuracy of the method on proteomes of diverged species, we used the set of 1695 proteins from the OrthoBench. Figures 3.4 and 3.5 present contour plots illustrating the effect of these four parameters on the recall and precision of PanTools, respectively; lighter colors represent higher values.

The first parameter, intersection rate (I) (in the range of 0.01-0.1), determines the minimum number of hexamers that two proteins need to have in common to be considered intersecting proteins in order to be selected for exact alignment. This number is calculated as the product of the intersection rate parameter and the total number of hexamers of the shorter protein. In general, by choosing lower intersection values the number of pairwise alignments and, in turn, the resource consumption of the program increases significantly. The lower the intersection value, the higher the recall and the lower the precision.

The second parameter affecting the clustering is the similarity threshold (T) (in the range of 25-99). Two proteins are considered similar if the normalized similarity score of their local alignment exceeds this threshold. Lower thresholds increase the number of detected similarities, boosting the sensitivity of the homology detection. So, the lower the threshold, the higher the recall, but the lower the precision.

The connectivity components of the similarity graph (similarity components) are the candidate homology groups which are then passed to the MCL clustering algorithm to be possibly split into more specific homology groups. To increase the granularity of the clustering and split the similarity components into a larger number of groups, we choose greater MCL inflations (M). Finally, we raise the scores to the power of the contrast parameter (C) to increase the contrast between the final similarity scores. Like for I and T, the lower the inflation and/or contrast, the higher the recall and the lower the precision.

The resulting F-scores (Supplementary Figure 3.S1) suggest that higher values of the four parameters are not desirable for grouping the proteome of these distant species. In support of this, we observed that increasing the parameter values improves the F-score of the method when analyzing the proteomes of closely related species. Based on these observations, we experimentally optimized 8 groups of default parameter settings (*d1*-*d8*), ranging from strict to relaxed by linearly decreasing the 4 mentioned parameters (Supplementary Table 3.S6). This allows the user to fine-tune the settings for different types of datasets and/or downstream applications. We recommend users to either use Table 3.S6 to choose appropriate settings based on the divergence of the proteomes or try multiple settings and pick one based on the desired resolution from one-to-one orthologs to multi-gene families. In our experiments, we used the most strict setting (*d1*) for the closely related strains of *E.coli* and *S.cerevisiae*, the next strict setting (*d2*) for *A.thaliana* datasets, and the most relaxed setting (*d8*) for the OrthoBench data.

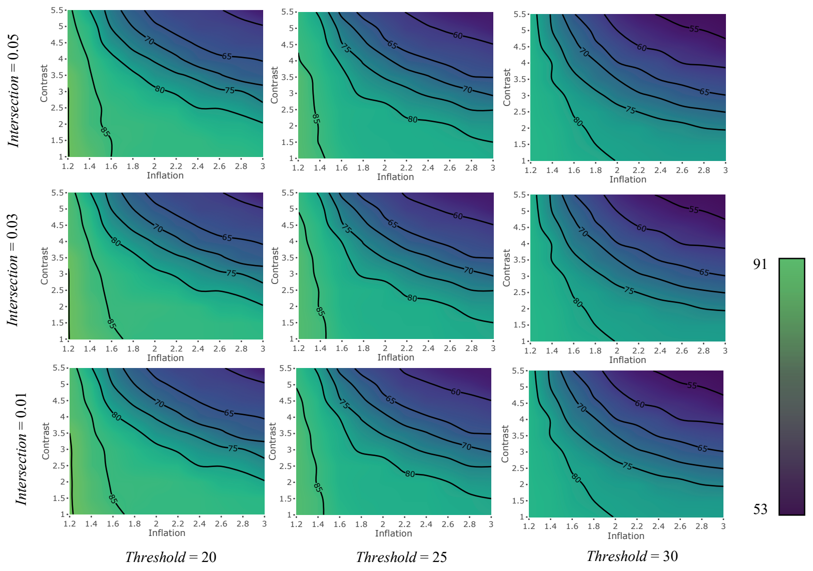


Figure 3.4. The effect of intersection rate, similarity threshold, contrast and inflation rate, on the recall of PanTools. Each contour plot belongs to a pair of intersection and threshold values, with the x and y axis representing inflation and contrast parameters.

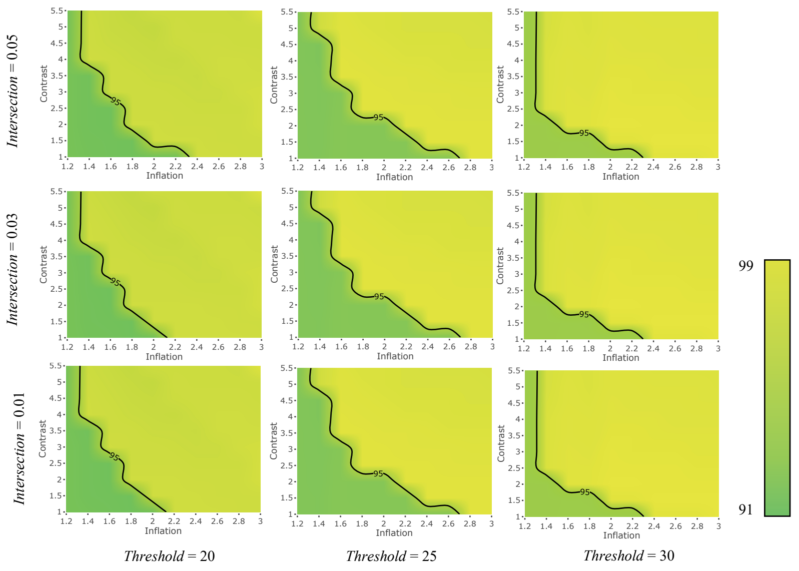


Figure 3.5. The effect of intersection rate, similarity threshold, contrast and inflation rate, on the precision of PanTools. Each contour plot belongs to a pair of intersection and threshold values, with the x and y axis representing inflation and contrast parameters.

3.3.2 PanTools is efficient and accurate on OrthoBench data

OrthoBench is a resource of 70 curated eukaryotic protein families from 12 metazoans which was established to assess the performance of TreeFam [21], eggNOG [22], OrthoDB [23], OrthoMCL [24], and OMA [25]. We call this benchmark M12 in the rest of this paper. The homology relationships between these protein families are difficult to detect due to differences in their rate of evolution, domain architecture, low-complexity regions/repeats, lineage-specific losses/duplications, and alignment quality [17].

We compared the performance of PanTools to that of OrthoFinder, which previously showed the highest accuracy on this benchmark data. We first created a mapping from the 1695 OrthoBench proteins to the 404,657 proteins of the 12 metazoans available in Ensembl release 60. We then ran PanTools and OrthoFinder independently on these 12 complete proteomes and calculated the recall, precision and F-score using the same procedure as proposed for OrthoFinder. In this experiment, PanTools achieved the same recall as OrthoFinder but at a remarkably higher precision, resulting in a 3% higher overall F-score of 85.5%. Additionally, there were significant differences in run-times. Running on 16 cores, PanTools terminated after 2 hours and OrthoFinder after 77.6 hours.

3.3.3 PanTools scales to large eukaryotic datasets and maintains accuracy

To examine the scalability of our method to large eukaryotic datasets, we first ran it on 5 datasets of *Saccharomyces cerevisiae* (Y3, Y13, …, Y93) and on 5 datasets of *Arabidopsis thaliana* accessions (A3, A7, …, A19) with an increasing number of proteomes. We compared the run-time and accuracy (F-score) of PanTools to those of OrthoFinder and PanX (Figure 3.6).

On the largest yeast dataset (Y93), PanTools was 112 times faster than OrthoFinder (0.9 hours vs. 4 days) and 7.6 times faster than PanX, with a slightly higher F-score. Similarly, on the largest Arabidopsis dataset (19 accessions), PanTools was 42 times faster (1 hour vs. 2.7 days) than OrthoFinder and 5.2 times faster than PanX while maintaining its higher F-score. Overall, OrthoFinder starts with a low accuracy but seems to level out at a higher value as the number of proteomes grows, albeit at the cost of drastic increase in run-time. Although PanX was almost as accurate as OrthoFinder on the *S.cerevisiae* data, its accuracy fell below that of OrthoFinder on the *A. thaliana* data, likely because plants have more diverse proteomes than the bacteria PanX was designed for.

3.3.4 PanTools is applicable to large microbial datasets

To compare the performance of our approach to PanX, a recently published tool dedicated to the microbial data, we applied both tools to the proteomes of 600 *E.coli* strains downloaded from GenBank (Supplementary spreadsheet datasets.xlsx). Both PanX and PanTools processed this large dataset in ~15 hours, resulting in F-scores of 71.6 and 72.9, respectively. In this experiment, we ran PanX in divide-and-conquer mode to speed it up.

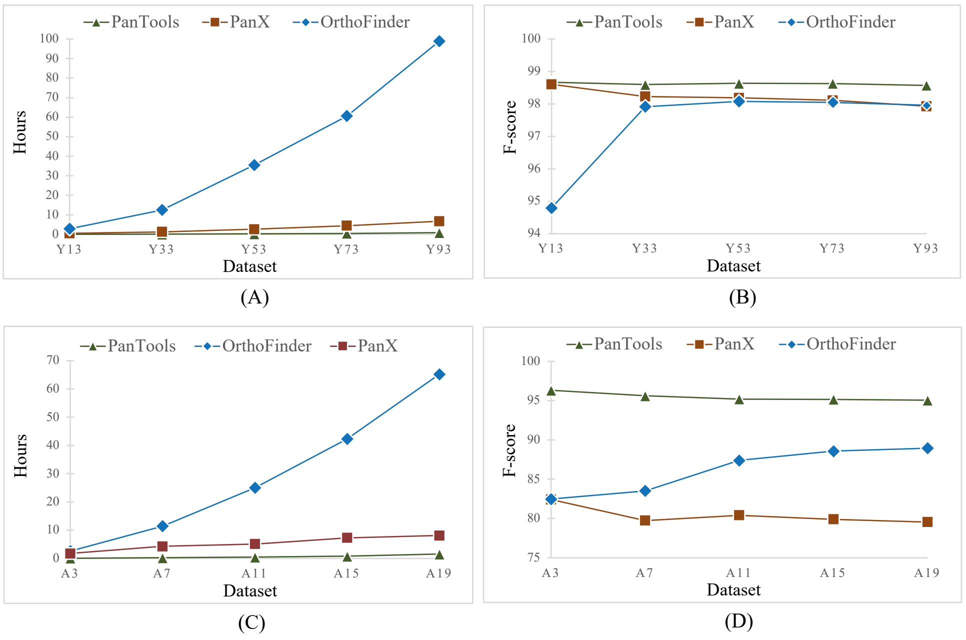


Figure 3.6. Run-time and accuracy of PanTools compared to those of PanX and OrthoFinder. Run-time and F-score of the three tools were calculated on the pan-proteomes of 5 S. cerevisiae 5 *A. thaliana* accessions.

3.3.5 PanTools significantly reduces the number of pairwise comparisons

The efficiency of PanTools is due to its *k*-mer-based approach, which significantly reduces the number of fruitless protein alignments. Table 3.1 shows that the numbers of pairwise comparisons in different experiments are thousands-fold less than what is needed in a naïve all-pairs approach.

Table 3.1. The number of PanTools comparisons is extremely less than what is needed in a naïve all-pairs approach, since we efficiently filter out un-related pairs of proteins.

Dataset	Naïve (millions)	PanTools (thousands)	Fold decrease
Y13	2,472	507	4,874
Y33	15,979	3,415	4,679
Y53	41,181	8,888	4,633
Y73	78,121	16,937	4,613
Y93	126,519	27,494	4,602
A3	4,284	508	8,435
A7	23,225	2,889	8,038
A11	57,382	7,229	7,938
A15	105,111	12,904	8,146
A19	169,570	21,022	8,066
M12	81,873	20,094	4,074
E600	4,993,364	926,638	5,389

To scale to hundreds of eukaryotic or thousands of prokaryotic proteomes using reasonable amount of resources, there were two main limitations to be resolved: first, the local sequence alignment of proteins, which we tried to mitigate by distributing the intersecting pairs among multiple threads to be aligned in parallel; second, the size of the data structure used for detecting the intersecting proteins, which grows linearly with the size of the input data. To reduce the memory needs, currently we ignore extremely abundant and rare hexamers, which are less informative. By using space-efficient data structures, for example MinHash sketches [26], we may be able to further decrease the memory consumption of the program.

3.3.6 PanTools reproduced the majority of groups detected by other tools

In all experiments, PanTools was able to perfectly reproduce the majority of the groups detected by OrthoFinder and PanX. Table 3.2 shows the percentage of the groups generated by OrthoFinder and PanX which have an identical counterpart in the PanTools groups. Generally, the overlap decreases as the size of data grows, because the probability of having exactly identical groups drops, although the corresponding groups have highly similar compositions.

Table 3.2. There is a large overlap between PanTools groups and those of OrthoFinder and PanX. The overlap between PanTools and OrthoFinder drops slightly with the size of the pan-proteome. The overlaps with PanX groups are generally larger especially on *A.thaliana* pan-proteomes.

Dataset	Reproduced OrthoFinder groups	Reproduced PanX groups
Y13	94.9 %	96.3 %
Y33	94.9 %	95.5 %
Y53	93.9 %	94.8 %
Y73	93.6 %	94.5 %
Y93	93.3 %	93.8 %
A3	72.1 %	80.1 %
A7	64.9 %	71.5 %
A11	64.9 %	69.1 %
A15	64.8 %	72.5 %
A19	64.6 %	79.8 %
M12	76.3 %	-
E600	-	59.7 %

3.3.7 Parameters can affect the performance of different application scenarios

To investigate the effect of the 8 suggested parameter sets (from strict to relaxed) on homology clustering, we used a large proteome of 12 phylogenetically diverse Brassicaceae species, including the model plant *Arabidopsis thaliana*, plus *Vitis vinifera* as an outgroup. We specifically considered four genes with different copy numbers in *A.thaliana*, including three MADS-box genes – the floral homeotic protein APETALA 3 (AP3), the floral homeotic protein AGAMOUS (AG) and the flowering locus C (FLC) – and one housekeeping gene: the ubiquitin extension protein 1 (UBQ1), and looked into the composition of their homology groups detected by PanTools using the 8 parameter settings from strictest (*d*1) to the most relaxed (*d*8). Each column of Table 3.3 represents a homology group and each entry reflects the count of homologs of the genes AP3, AG, FLC and UBQ1 from different species in that group.

With all settings, we detected a single *AP3* homolog in arabidopsis, which indicates that this MADS-box gene is significantly differentiated from other MADS-box genes. We also found unique orthologues for most of the other species. We detected a single ortholog of AG until *d*5, after which we also identify its ancient paralogs Shatterproof 1 and 2 (SHP1/2). The duplication that gave rise to the split between AG and SHP1/2 is quite old (the gamma triplication shared by most eudicot species). At *d*6 we also detect STK which comes from an even earlier duplication (perhaps at the origin of angiosperms) [27]. At *d*7 and *d*8 we identify many of the various MADS-box genes across different lineages. FLC is alone until *d*4, where the transposition duplicate MAF1 (but not yet members of the MAF2-5 clade) is added. Then MAF2-5 members derived from the At-alpha WGD (whole genome duplication) from FLC come up, followed by inclusion of the tandem expansion of these genes. At subsequent settings, we start picking up other MADS-box genes.

UBQ1 is a house-keeping gene that was duplicated by the ancient whole genome duplication (WGD) At-alpha shared across the Brassicaceae (PGDD database) [28]. Our method recovered both the ortholog and its in-paralog (UBQ2) even using the strictest setting (*d*1), meaning that these genes are very similar despite having diverged around 40 mya. Thus, the function of the two genes is likely highly conserved. From *d*5 on, PanTools identifies other, more distantly related homologs and ultimately (*d*8) all members of the larger family (UBQ1-UBQ14) plus a few related genes.

Table 3.4 shows the distribution of the normalized similarity scores in each of the detected homology groups. It is clear that more relaxed settings allow including more diverse pairs of homologs, which are less similar in the final clusters.

Table 3.3. Entries represent the number of homologs of the 4 selected genes of *A.thaliana* (AP3, AG, FLC, UBQ1) in the 12 other Brassicaceae species using different settings (d1-d8). The abrupt increase at d5 shows that overly relaxed parameters (d5 to d8) are not suitable for homology detection at the level of the same species.

	AP3 (AT3G54340)								**AG** (AT4G18960)								**FLC** (AT5G10140)								**UBQ1** (AT3G52590)							
	d1	***d2***	***d3***	***d4***	***d5***	***d6***	***d7***	***d8***	***d1***	***d2***	***d3***	***d4***	***d5***	***d6***	***d7***	***d8***	***d1***	***d2***	***d3***	***d4***	***d5***	***d6***	***d7***	***d8***	***d1***	***d2***	***d3***	***d4***	***d5***	***d6***	***d7***	***d8***
A. thaliana	1	1	1	1	1	1	1	1	1	1	1	1	3	4	8	27	1	1	1	2	4	6	8	36	2	2	2	2	15	16	17	17
A. lyrata	1	1	1	1	1	1	1	3	1	1	1	1	4	4	7	24	0	1	1	1	4	5	5	32	0	0	0	0	19	20	21	22
C. rubella	1	1	1	1	1	1	1	2	1	1	1	1	3	4	8	24	0	1	1	1	2	3	4	33	2	2	2	2	15	15	18	19
M. maritima	1	1	1	1	1	1	1	3	1	1	1	1	4	4	7	24	0	1	1	1	4	4	6	34	2	2	2	2	12	12	16	18
D. sophiades	1	1	1	1	1	1	1	3	1	1	1	1	4	4	9	26	0	1	1	1	7	9	12	42	2	2	2	2	11	12	12	12
S. irio	0	1	1	1	1	1	3	6	0	0	2	3	8	8	10	26	0	0	0	2	4	4	11	36	3	3	3	3	13	14	17	20
M. perfoliatum	1	1	1	1	1	1	1	2	1	1	1	1	4	4	8	28	0	1	1	1	3	5	8	35	3	3	3	3	13	13	18	21
T. salsuginea	1	1	1	1	1	1	1	3	1	1	1	1	4	4	8	27	0	1	1	1	5	5	7	33	2	2	2	2	12	13	13	13
T. halophila	0	0	2	2	2	2	2	3	0	0	0	4	6	9	13	41	0	0	0	0	0	4	4	48	4	4	4	4	15	15	15	15
A. alpina	1	1	1	1	1	1	2	3	0	1	2	2	5	7	9	21	0	0	1	3	6	7	11	30	5	6	6	7	17	17	21	23
E. syriacum	1	1	1	1	1	1	1	2	0	0	0	0	2	3	7	21	0	0	1	1	3	5	9	31	3	3	3	3	18	20	21	22
A. arabicum	0	1	1	1	2	3	4	5	0	1	2	2	8	11	13	24	0	1	2	3	4	5	7	25	3	3	3	3	12	12	12	13
V. vinifera	0	0	0	0	1	1	1	4	0	0	1	1	4	4	8	26	0	0	0	0	0	0	3	34	2	2	3	3	7	9	9	10

Table 3.4. The large difference between minimum and maximum of normalized similarity scores in the detected homology groups of the 4 genes confirms that settings relaxer than d4 should not be used at the species level.

Gene name (ID)		*d1*	*d2*	*d3*	*d4*	*d5*	*d6*	*d7*	*d8*
AP3	Min	95.1	88.1	79.3	79.3	57.5	57.5	57.5	25.0
(AT3G54340)	Avg	97.0	95.5	91.7	91.7	87.2	87.2	87.2	38.7
	Max	98.8	98.8	98.8	98.8	98.8	98.8	98.8	100.0
AG	Min	95.1	85.2	75.7	75.7	56.2	45.0	35.0	25.0
(AT4G18960)	Avg	96.7	93.6	90.1	89.3	82.6	62.7	52.2	38.7
	Max	99.5	99.5	99.5	98.0	99.5	100.0	100.0	100.0
FLC	Min	-	85.1	79.9	65.0	55.0	45.0	35.0	25.0
(AT5G10140)	Avg	-	87.3	85.2	75.1	68.3	62.7	52.2	38.7
	Max	-	94.4	94.4	100.0	100.0	100.0	100.0	100.0
UBQ1	Min	98.2	96.9	81.7	81.7	57.2	45.0	35.0	25.1
(AT3G52590)	Avg	99.6	99.3	99.6	99.6	97.2	78.3	72.0	68.5
	Max	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0

3.4 Conclusion

We presented an efficient method for detecting homology groups across a large number of individuals and/or species. To make homology detection efficient we adopted a *k*-mer-based approach, which substantially reduces the number of pairwise comparisons. Specifically, we first count the number of peptide hexamers two proteins share, and only if this number is high enough, we perform a local alignment of the so-called intersecting proteins to calculate their exact similarity score.

We clearly observed a trade-off between recall and precision of the homology inference. Favoring recall or precision strongly depends on the application [29]. In a phylogenetic study one may specifically be interested in identifying precise one-to-one orthologs, while others may want to capture a complete protein family to achieve insights into gene-duplication events across species. The four defined parameters (and the 8 default settings) give users the flexibility to control the program's behavior. It is important to note that different types of genes may be under different selection pressures and constraints and have different evolutionary dynamics. Thus, the optimal parameter setting will depend both on the specific gene and on the desired application, as demonstrated by the four genes across the Brassicaceae.

As we store the homology groups in the pan-genome, it is possible to query the pan-genome graph database for statistics on, for example, the size of the homology groups, the copy number of the genes and the conservation rate of the proteins in different groups. In the future, we will extend PanTools with additional functionality to exploit this pan-genome database for comparative genomics on large collections of complex genomes.

3.5 Supplementary materials

Here we present instructions for running the homology detection functionality of PanTools. First, the PanTools package should be cloned to the home directory from GitHub:

```
cd
git clone https://github.com/sheikhizadeh/pantools
```

You should already have installed Java (JDK 1.8 or higher) and the latest version of MCL clustering program on your machine and have added the path to their executables to the PATH shell environment variable.

3.5.1 Constructing the proteome layer of the pan-genome

To construct the proteome layer of a pan-genome from a set of protein sequences, collect the paths to all the protein files in FASTA format in a text file (for example, `proteins.txt`) which contains these lines:

```
$HOME/proteins/p1.faa
$HOME/proteins/p2.faa
$HOME/proteins/p3.faa
```

Then run:

```
java -jar pantools/dist/pantools.jar build_panproteome -dp ./DB -pf
./proteins.txt
```

To start homology detection using 4 threads and the most relaxed setting (8) type:

```
java -jar pantools/dist/pantools.jar group -dp ./DB -tn 4 -rn 8
```

Alternatively, you can start from genomes and GFF files and build a complete pan-genome. First, you need to construct the genome layer of a pan-genome by collecting the path to all the genome files in FASTA format in a text file (for example, `genomes.txt`) which contains these lines:

```
$HOME/genomes/g1.fna
$HOME/genomes/g2.fna
$HOME/genomes/g2.fna
```

Then:

```
java -jar pantools/dist/pantools.jar build_pangenome -dp ./DB -gf
./genomes.txt
```

The genomes' annotation should be added to build the annotation layer of the pan-genome by giving the path to the `GFF` file for each genome as indicated by the number at the start of each line (for example, in `annotations.txt`) which contains these lines:

```
1 $HOME/annotations/a1.gff
2 $HOME/annotations/a2.gff
3 $HOME/annotations/a3.gff
```

Then run:

```
java -jar pantools/dist/pantools.jar add_annotations -dp ./DB -af
./annotations.txt
```

Now, the proteins are annotated in the pan-genome and homology detection can be started by:

```
java -jar pantools/dist/pantools.jar group -dp ./DB -tn 4 -rn 8
```

3.5.2 Choice of K

Here, we calculate the smallest k value which keeps the probability of random occurrences of a k-mer below a desirable probability p. Given sequences S of length L from an alphabet of size α, the probability of a given k-mer from this alphabet being present at a given position of S by chance is:

$$x = \frac{1}{\alpha^k} \tag{1}$$

then, the probability of the given k-mer being absent at a given position of S is:

$$1 - x \tag{2}$$

and the probability that the k-mer occurs nowhere in S will be:

$$(1 - x)^n \tag{3}$$

, where n is the number of k-mers in sequence S: $L - k + 1$
so, the probability that it occurs somewhere in S would be:

$$1 - (1 - x)^n \tag{4}$$

Using the Taylor (or Maclaurin) series when $|x| \ll 1$, which holds here as $\alpha^k \gg 1$:

$$\ln(1 - x) = -x - \frac{x^2}{2} - \frac{x^3}{3} - \frac{x^4}{4} - \cdots \approx -x$$
$$n \ln(1 - x) \approx -nx$$
$$(1 - x)^n \approx \exp\,(-nx)$$

So, the probability (4) is well approximated by:

$$1 - exp(-xn) \tag{5}$$

setting this probability less than the desired value p we will have:

$$1 - exp(-xn) \; < p$$
$$-xn > \ln\,(1 - p)$$
$$x < \frac{\ln\,(1 - p)}{-n}$$

substituting x gives:

$$\frac{1}{\alpha^k} < \frac{\ln\,(1 - p)}{-n}$$

$$k \;\geq\; \left\lceil \log_\alpha \frac{-n}{\ln(1-p)} \right\rceil \tag{6}$$

For peptide sequences α = 20, and considering $n \approx L \;=\; 30{,}000$ the length of the largest known protein, and setting p = 0.001, the smallest suitable k is 6.

3.5.3 Experiments

We demonstrate the accuracy, scalability, efficiency and applicability of PanTools on 12 datasets of bacteria, fungi, plants and Metazoa. Y13-Y93 are 5 datasets of increasing size compiled from 93 *Saccharomyces cerevisiae* strains and A3-A19 are 5 datasets compiled from 19 *Arabidopsis thaliana* accessions. M12 is the OrthoBench data from 12 metazoans and E600 a large dataset of 600 *Escherichia coli* strains. For most of these data sets we selected a subset of the total proteins for evaluation of the methods (Table 3.S1), because we were not able to establish a ground truth for all. Columns “Intersecting” and “Similar” give the number of intersecting resp. similar pairs of proteins in the constructed pan-genome.

Table 3.S1. Statistics on the pan-proteomes constructed from the 12 different datasets.

Dataset	*k*-mers	All	Selected	Intersecting	Similar
Y13	2,634,965	70,312	61,414	507,164	423,162
Y33	2,722,436	178,769	156,135	3,414,746	2,872,902
Y53	2,794,291	286,987	250,649	8,887,967	7,486,596
Y73	2,947,823	395,274	345,333	16,936,559	14,275,904
Y93	3,027,750	503,028	439,573	27,494,049	23,189,200
A3	8,324,590	92,564	92,564	507,874	139,384
A7	8,861,259	215,523	215,523	2,889,361	888,305
A11	9,167,464	338,769	338,769	7,228,520	2,273,496
A15	9,425,939	458,499	458,499	12,903,576	4,197,580
A19	9,621,923	582,357	582,357	21,021,793	6,853,404
M12	29,193,967	404,657	404,657	20,094,137	12,942,693
E600	7,580,644	3,160,178	688,160	926,638,469	734,423,473

Table 3.S2 shows the number of groups detected by each of the three tools, the number of real groups and the running time of tools in hours. For the Y datasets and the E600 dataset, the real groups are determined by the valid locus tags of the proteins extracted from the GenBank files. For A datasets, the real groups are the gene identifiers which end with .1, corresponding to the first annotated isoform of the genes. For M12 we used the identifier of the known 70 protein families in the OrthoBench as the real group identifiers.

Table 3.S2. Results of running PanTools, OrthoFinder and PanX on the 12 datasets. Run-times are presented in hours.

Dataset	PanTools groups	OrthoFinder groups	PanX groups	Real groups	PanTools run-time	OrthoFinder run-time	PanX run-time
Y13	4,990	4,746	4,830	4,894	0.02	2.86	0.52
Y33	5,078	4,853	4,849	4,906	0.11	12.50	1.28
Y53	5,151	4,886	4,855	4,909	0.28	35.53	2.65
Y73	5,199	4,889	4,857	4,910	0.51	60.60	4.47
Y93	5,257	4,886	4,867	4,911	0.88	98.95	6.72
A3	32,236	29,231	30,065	30,971	0.05	2.58	1.77
A7	35,083	31,368	31,356	31,037	0.25	11.47	4.28
A11	36,842	33,367	32,048	31,046	0.43	25.02	5.12
A15	38,292	34,192	32,637	31,047	0.79	42.30	7.28
A19	39,579	34,896	33,095	31,049	1.55	65.20	8.13
M12	171	147	-	70	1.96	77.60	-
E600	10,196	-	7,843	10,152	15.65	-	15.48

Table 3.S3. Accuracy of PanTools on the 12 datasets.

Pan-genome	TP	FP	FN	Recall	Precision	F_score
Y13	61,101	1,339	313	99.5	97.9	98.7
Y33	155,361	3,659	774	99.5	97.7	98.6
Y53	249,402	5,679	1,247	99.5	97.8	98.6
Y73	343,636	7,852	1,697	99.5	97.8	98.6
Y93	437,305	10,435	2,268	99.5	97.7	98.6
A3	90,033	4,353	2,531	97.3	95.4	96.3
A7	207,675	11,168	7,848	96.4	94.9	95.6
A11	325,625	19,646	13,144	96.1	94.3	95.2
A15	440,596	26,837	17,903	96.1	94.3	95.2
A19	558,677	34,412	23,680	95.9	94.2	95.1
M12	1,328	135	315	80.8	90.8	85.5
E600	665,167	472,215	22,993	96.7	58.5	72.9

Table 3.S4. Accuracy of OrthoFinder on the 11 datasets.

Pan-genome	TP	FP	FN	Recall	Precision	F_score
Y13	61,356	6,678	58	99.9	90.2	94.8
Y33	155,945	6,457	190	99.9	96.0	97.9
Y53	250,363	9,532	286	99.9	96.3	98.1
Y73	344,989	13,859	344	99.9	96.1	98.0
Y93	439,192	17,943	381	99.9	96.1	98.0
A3	86,363	34,435	2,246	97.5	71.5	82.5
A7	205,912	74,595	6,657	96.9	73.4	83.5
A11	325,099	82,904	10,849	96.8	79.7	87.4
A15	444,098	98,963	15,102	96.7	81.8	88.6
A19	560,002	119,627	19,534	96.6	82.4	88.9
M12	1,328	243	315	80.8	84.5	82.6

Table 3.S5. Accuracy of PanX on the 11 datasets.

Pan-genome	TP	FP	FN	Recall	Precision	F_score
Y13	61,302	1,717	19	100.0	97.3	98.6
Y33	155,800	5,566	58	100.0	96.6	98.2
Y53	250,144	9,176	68	100.0	96.5	98.2
Y73	344,660	13,193	79	100.0	96.3	98.1
Y93	438,691	18,460	108	100.0	96.0	97.9
A3	87,047	35,756	1,387	98.4	70.9	82.4
A7	208,010	101,599	4,072	98.1	67.2	79.7
A11	328,418	153,371	6,688	98.0	68.2	80.4
A15	448,690	216,306	9,317	98.0	67.5	79.9
A19	568,800	280,513	11,987	97.9	67.0	79.5
E600	667,680	513,833	15,785	97.7	56.5	71.6

Table 3.S6. Pre-cooked default parameters: the values of the four parameters *I*, *T*, *M* and *C* in each of the default pre-cooked set of parameters.

Parameter	***d1***	***d2***	***d3***	***d4***	***d5***	***d6***	***d7***	***d8***
Similarity threshold	95	85	75	65	55	45	35	25
Intersection rate (*I*)	0.09	0.08	0.07	0.06	0.05	0.04	0.03	0.02
MCL inflation (*M*)	9.6	8.4	7.2	6.0	4.8	3.6	2.4	1.2
Contrast (*C*)	8	7	6	5	4	3	2	1

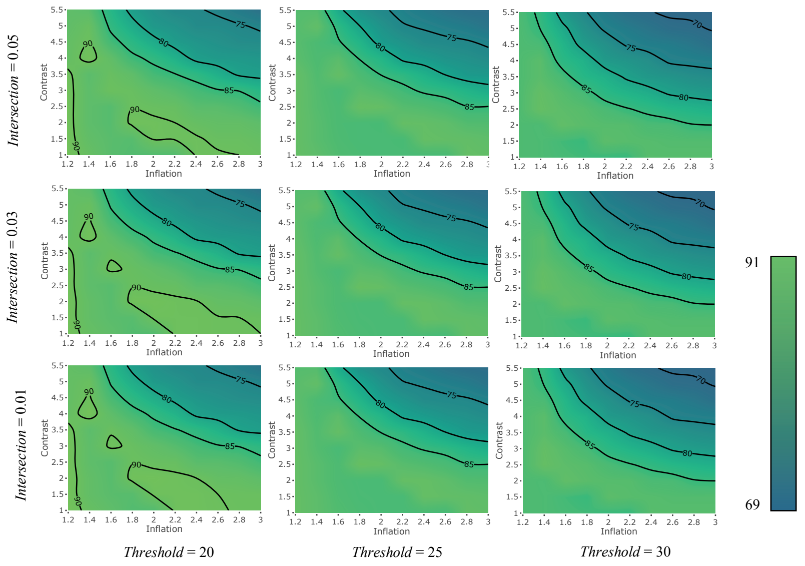


Figure 3.S1. The effect of intersection rate, similarity threshold, contrast and inflation rate, on the **F-score** of PanTools. Each contour plot belongs to a pair of intersection and threshold values, with the x and y axis representing inflation and contrast parameters.

References

1. Koonin E V. Orthologs, paralogs, and evolutionary genomics. *Annu Rev Genet*. 2005;39:309–38.
2. Zhu J, Vinothkumar KR, Hirst J. Structure of mammalian respiratory complex I. *Nature*. 2016;536(7616):354–8.
3. Tekaia F. Inferring orthologs: open questions and perspectives. *Genomics Insights*. 2016;9:17–28.
4. Tatusov RL. A genomic perspective on protein families. *Science*. 1997;278(5338):631–7.
5. Powell S, Forslund K, Szklarczyk D, Trachana K, Roth A, Huerta-Cepas J, *et al*. EggNOG v4.0: Nested orthology inference across 3686 organisms. *Nucleic Acids Res*. 2014;42(D1).
6. Zdobnov EM, Tegenfeldt F, Kuznetsov D, Waterhouse RM, Simão FA, Ioannidis P, *et al*. OrthoDB v9.1: cataloging evolutionary and functional annotations for animal, fungal, plant, archaeal, bacterial and viral orthologs. *Nucleic Acids Res*. 2017;45(D1):D744–9.
7. Huerta-Cepas J, Capella-Gutiérrez S, Pryszcz LP, Marcet-Houben M, Gabaldón T. PhylomeDB v4: zooming into the plurality of evolutionary histories of a genome. *Nucleic Acids Res*. 2014;42(D1): D897-D902.
8. Li H. TreeFam: a curated database of phylogenetic trees of animal gene families. *Nucleic Acids Res*. 2006;34(90001):D572–80.
9. Remm M, Storm CEV, Sonnhammer ELL. Automatic clustering of orthologs and in-paralogs from pairwise species comparisons. *J Mol Biol*. 2001;314(5):1041–52.
10. Roth AC, Gonnet GH, Dessimoz C. Algorithm of OMA for large-scale orthology inference. *BMC Bioinformatics*. 2008;9(1):518.
11. Li L, Stoeckert CJ, Roos DS. OrthoMCL: Identification of ortholog groups for eukaryotic genomes. *Genome Res*. 2003;13(9):2178–89.
12. Sheikhizadeh S, Schranz ME, Akdel M, de Ridder D, Smit S. PanTools: representation, storage and exploration of pan-genomic data. *Bioinformatics*. 2016;32(17):i487–93.
13. Marschall T, Marz M, Abeel T, Dijkstra L, Dutilh BE, Ghaffaari A, *et al*. Computational pan-genomics: Status, promises and challenges. *Brief Bioinform*. 2018;19(1):118–35.
14. Enright AJ, Dongen S V, Ouzounis CA, Van Dongen S, Ouzounis CA. An efficient algorithm for large-scale detection of protein families. Nucleic Acids Res. 2002;30(7):1575–84.
15. Opitz CA, Kulke M, Leake MC, Neagoe C, Hinssen H, Hajjar RJ, *et al*. Damped elastic recoil of the titin spring in myofibrils of human myocardium. *Proc Natl Acad Sci*. 2003;100(22):12688–93.
16. Emms DM, Kelly S. OrthoFinder: solving fundamental biases in whole genome comparisons dramatically improves orthogroup inference accuracy. *Genome Biol*. 2015;16(1):157.
17. Trachana K, Larsson T a, Powell S, Chen W-H, Doerks T, Muller J, *et al*. Orthology prediction methods: A quality assessment using curated protein families. *Bioessays*. 2011;33(10):769–80.
18. Ding W, Baumdicker F, Neher RA. panX: pan-genome analysis and exploration. *Nucleic Acids Res*. 2018;46(1):e5–e5.
19. Strope PK, Skelly DA, Kozmin SG, Mahadevan G, Stone EA, Magwene PM, *et al*. The 100-genomes strains, an *S. cerevisiae* resource that illuminates its natural phenotypic and genotypic variation and emergence as an opportunistic pathogen. *Genome Res*. 2015;125(5):762–74.
20. Gan X, Stegle O, Behr J, Steffen JG, Drewe P, Hildebrand KL, *et al*. Multiple reference genomes and transcriptomes for Arabidopsis thaliana. *Nature*. 2011;477(7365):419–23.
21. Ruan J, Li H, Chen Z, Coghlan A, Coin LJM, Guo Y, *et al*. TreeFam: 2008 Update. *Nucleic Acids Res*. 2007;36:D735–40.
22. Muller J, Szklarczyk D, Julien P, Letunic I, Roth A, Kuhn M, *et al*. eggNOG v2.0: extending the evolutionary genealogy of genes with enhanced non-supervised orthologous groups, species and functional annotations. *Nucleic Acids Res*. 2010;38:D190–5.
23. Waterhouse RM, Zdobnov EM, Tegenfeldt F, Li J, Kriventseva E V. OrthoDB: the hierarchical catalog of eukaryotic orthologs in 2011. *Nucleic Acids Res*. 2011;39:D283–8.
24. Chen F. OrthoMCL-DB: querying a comprehensive multi-species collection of ortholog groups. *Nucleic Acids Res*. 2006;34(90001):D363–8.
25. Altenhoff AM, Schneider A, Gonnet GH, Dessimoz C. OMA 2011: orthology inference among 1000 complete genomes. *Nucleic Acids Res*. 2011;39(Database):D289–94.
26. Ondov BD, Treangen TJ, Melsted P, Mallonee AB, Bergman NH, Koren S, *et al*. Mash: fast genome and metagenome distance estimation using MinHash. *Genome Biol*. 2016;17(1):132.
27. Cheng S, van den Bergh E, Zeng P, Zhong X, Xu J, Liu X, *et al*. The *Tarenaya hassleriana* genome provides insight into reproductive trait and genome evolution of crucifers. *Plant Cell*. 2013;25(8):2813–30.
28. Lee T-H, Tang H, Wang X, Paterson AH. PGDD: a database of gene and genome duplication in plants. *Nucleic Acids Res*. 2012;41(D1):D1152–8.

29. Altenhoff AM, Boeckmann B, Capella-Gutierrez S, Dalquen DA, DeLuca T, Forslund K, *et al*. Standardized benchmarking in the quest for orthologs. *Nat Methods*. 2016;13(5):425–30.

Chapter 4

ACE: accurate correction of errors using K-mer tries

This chapter is published as:
Sheikhizadeh S, de Ridder D. ACE: accurate correction of errors using *K*-mer tries. *Bioinformatics*. 2015;31(19): 3216–18.
DOI: 10.1093/bioinformatics/btv332

Abstract

The quality of high-throughput next-generation sequencing data significantly influences the performance and memory consumption of assembly and mapping algorithms. The most ubiquitous platform, Illumina, mainly suffers from substitution errors. We have developed a tool, ACE, based on *K*-mer tries to correct such errors. On real MiSeq and HiSeq Illumina archives, ACE yields higher gains in terms of coverage depth, outperforming state-of-the-art competitors in the majority of cases. ACE is licensed under the GPL license and can be freely obtained at https://github.com/sheikhizadeh/ACE/. The program is implemented in C++ and runs on most Unix-derived operating systems.

4.1 Introduction

Genome sequencing involves reading thousands or millions of genome fragments and reconstructing the original genome, either by assembling these reads in *de novo* assembly projects, or aligning them to a known reference genome in re-sequencing studies. Over the last decade, next-generation sequencing (NGS) technology dramatically increased the ease with which material can be sequenced, yielding millions of short reads in a short time. The lower quality of the data (compared to Sanger sequencing) however significantly influences performance and memory consumption of assemblers and alignment algorithms; as a result, there has been a growing interest in correcting errors in short-read archives. Sequencing errors can result in substitutions, insertions, deletions and unconfirmed nucleotides represented by 'N' symbols. The most ubiquitous platform, Illumina, mostly suffers from substitution errors while for others, like 454 and Ion Torrent, insertions and deletions are most abundant.

As an error at a specific genomic position occurs infrequently and randomly, an erroneous base can be detected and corrected taking advantage of the high frequency of the reads that cover that position. This is the idea behind all count-based error correction methods which count K-mers using various data structures.

For example, SHREC [1] constructs a generalized suffix trie while HiTEC [2] uses a suffix array. Built upon SHREC, Hybrid-SHREC [3] captures InDel (insertion, deletion) errors as well as substitutions. SGA [4] performs error-correction using the FM-index derived from the compressed Burrows-Wheeler transform. BLESS [5] employs a bloom-filter and RACER [6] organizes 2-bit-encoded K-mers as 64-bit integers and stores them in a hash table. Fiona, based on partial suffix array, is also able to deal with InDel errors [7]. Alternatively, K-spectrum based error correction methods, like Quake [8] and Musket [9] collect all K-mers appearing in the set of reads, and align those with a small Hamming distance from each other to achieve the correct consensus. Finally, MSA-based methods, like Coral [10], apply multiple sequence alignment between reads that share K-mers to detect errors. A recent survey provides a comprehensive review of error-correction methods, and establishes a common set of benchmark data and evaluation criteria [11].

Here we present ACE, a new K-mer count-based algorithm. We employ the K-mer trie, a data structure more time/space-efficient than the suffix trees employed in SHREC. K-mer tries have been effective in solving some bioinformatics problems [12,13].

4.2 Methods

ACE is the C++ implementation of our algorithm, equipped with Open-MP directives to scale with the number of available processors. It organizes the K-mers of short reads (and their reverse-complement) in a K-mer trie.

A K-mer trie of a sequence s (or set of sequences) is a trie of depth K which contains all K-mers of the sequence. Each edge has a label from the alphabet $\sum$; the concatenation of edge labels along the path from root to a node is called the spelled string of that node. Each leaf corresponds to one or more K-mers of s, and each node can contain the number of times that its spelled string appears in the sequence. A K-mer trie gives constant-time access to all patterns of length at most *K*, useful for counting and storing *K*-mers, detecting zygosity, determining ploidy and genome assembly.

To cope with large datasets, ACE constructs *K*-mer sub-tries one by one (Figure 4.1) by applying a prefix-based classification on *K*-mers and shortening them to *k*-mers where $k = K - p$ and *p* is the prefix length, determined based on the amount of available memory and the size of the input data. As the *K*-mer trie is very dense in the top levels, ACE further reduces memory consumption (and the size of search space) by building a root array instead of constructing the top triangle of the trie. Moreover, to efficiently organize billions of *K*-mers in the trie ACE applies another prefix-based division to allow parallel construction of the branches of each subtrie. In Figure 4.1, a branch has been divided into four sub-branches to be constructed and scanned for errors by four independent parallel threads. Branches are divided into 16, 64 or more sub-branches if more cores are available. More details on the algorithm, including a pseudo-code description, can be found in the Supplementary Material.

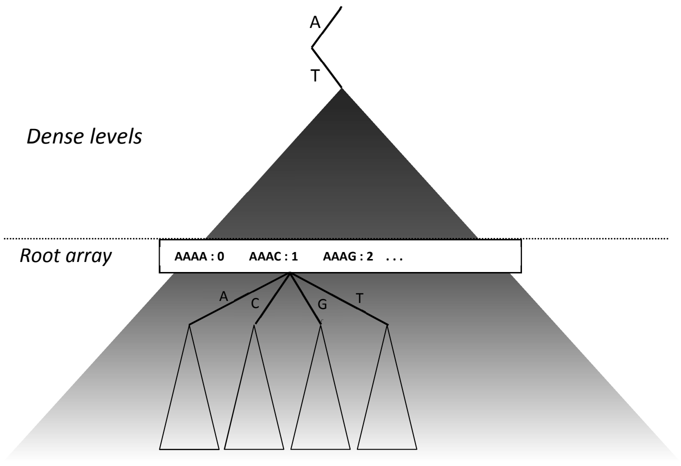


Figure 4.1. To reduce the memory consumption on large datasets, the full trie is broken into sub-tries which are processed serially (AT sub-trie is shown). To speed-up the algorithm each sub-trie is divided into a few branches to be processed in parallel by multiple threads.

4.3 Results

We experimentally compared the performance of ACE in increasing the coverage depth/breadth of reads/*K*-mers to those of seven state-of-the-art tools, using the benchmark data and following the same evaluation procedure as presented in a recent survey [11]. To be consistent with the result of Molnar *et al.*, we chose the same value of $K = 20$ for evaluations. The specifications of the MiSeq (M1–M9) and the HiSeq datasets (H1-H13) are presented in Supplementary Table 4.S1. All experiments were conducted on a Linux server (SUSE 3.8.6-2) with an Intel® Xeon® ES-2667 CPU, exploiting 16 logical cores running at 2.9 GHz and 256 GB RAM.

Table 4.1 compares the gain of ACE to that of its best competitor among seven state-of-the-art read cleaners: BLESS, Coral, HiTEC, Musket, RACER, SGA and SHREC; more detailed results can be found in Supplementary Tables 4.S3–11. In these evaluations, Depth of coverage indicates the average number of times each base is covered by reads/K-mers and Breadth of coverage indicates the proportion of the genome covered by reads/K-mers [11]. The first criterion is useful for quantitative applications and overlap-layout-consensus assembly, while the second is more applicable for De Bruijn graph-based genome assembly.

Table 4.1. The gain of ACE in increasing the depth/breadth of reads/K-mers, compared to that of the best tool in [11]. Note: Highlights indicate the level of improvement.

	Depth gain						**Breadth gain**					
	Read			**K-mer**			**Read**			**K-mer**		
		Best	**ACE**		**Best**	**ACE**		**Best**	**ACE**		**Best**	**ACE**
M1	*SGA*	26.30	56.10	*RACER*	37.42	58.90	*BLESS*	6.75	9.13	*SGA*	0.00	-202.26
M2	*RACER*	58.06	57.05	*RACER*	49.48	51.73	*RACER*	6.31	6.23	*Cora*	0.01	-11.48
M3	*RACER*	13.09	14.06	*RACER*	19.05	23.01	*RACER*	1.94	2.08	*Cora*	-0.01	-3.92
M4	*RACER*	74.96	83.91	*RACER*	58.72	73.90	*RACER*	7.73	8.60	*Cora*	0.00	-0.94
M5	*RACER*	0.88	0.89	*RACER*	4.00	4.39	*RACER*	0.098	0.096	*SGA*	-0.95	-13.53
M6	*RACER*	16.46	18.03	*RACER*	21.09	26.01	*RACER*	3.47	3.75	*Cora*	0.00	-5.35
M7	*RACER*	86.11	90.26	*RACER*	65.59	79.48	*RACER*	28.41	29.80	*Cora*	0.00	-2.56
M8	*HiTEC*	37.78	73.18	*HiTEC*	47.65	69.47	*BLESS*	67.96	79.83	*HiTE*	5.00	-5.00
M9	*RACER*	0.39	0.45	*HiTEC*	6.71	7.90	*BLESS*	0.97	0.49	*Cora*	0.00	-45.94
H1	*BLESS*	61.72	63.55	*Musket*	51.28	67.67	*BLESS*	5.66	5.43	*Cora*	-1.74	-7.49
H2	*BLESS*	44.30	44.93	*HiTEC*	33.03	34.86	*BLESS*	4.54	4.56	*SGA*	-1.73	-1.96
H3	*RACER*	34.13	34.69	*RACER*	19.26	22.15	*BLESS*	2.28	2.25	*SGA*	-0.88	-1.20
H4	*RACER*	92.27	93.71	*HiTEC*	85.83	88.77	*BLESS*	49.49	49.67	*Cora*	-0.36	-1.72
H5	*BLESS*	13.70	13.80	*HiTEC*	9.27	9.48	*BLESS*	4.63	4.58	*SGA*	-9.83	-12.55
H6	*BLESS*	86.16	92.35	*HiTEC*	82.83	90.23	*BLESS*	82.87	85.46	*Cora*	0.00	-8.12
H7	*SGA*	52.89	54.26	*RACER*	47.35	50.35	*BLESS*	4.69	4.70	*SGA*	-1.99	-24.40
H8	*RACER*	26.77	27.83	*RACER*	11.92	13.74	*BLESS*	4.06	4.10	*SGA*	-2.84	-280.70
H9	*BLESS*	18.14	20.35	*RACER*	16.19	19.70	*BLESS*	4.50	4.62	*SGA*	-5.70	-27.11
H10	*BLESS*	26.13	27.53	*Musket*	19.64	21.67	*BLESS*	7.98	8.03	*SGA*	-6.49	-31.77
H11	*SGA*	61.66	57.52	*RACER*	38.11	46.19	*SGA*	12.92	12.03	*SGA*	-4.89	-11.59
H12	*SGA*	65.46	62.61	*RACER*	42.42	48.73	*SGA*	16.15	14.95	*SGA*	-5.63	-13.81
H13	*SGA*	27.86	27.70	*RACER*	35.49	40.35	*SGA*	3.18	3.01	*SGA*	-4.32	-11.29

ACE outperforms most other tools in terms of coverage depth gain, improving on the best competitor on 18 resp. 22 out of 22 datasets for reads resp. K-mers. In particular for MiSeq data, which contains more errors, the improvements can be significant. For coverage breadth, the picture is less clear: ACE outperforms the best alternative tool on 13 datasets on read coverage breadth gain, whereas K-mer coverage breadth gain was generally worse. However, all tools actually yield low read coverage breadth on most MiSeq data (as low as 0.25%) and decrease K-mer coverage breadth compared to the raw data (see Supplementary Tables 4.S5–6, 4S9–10).

Table 4.2 compares the time and memory consumption of ACE to those of the three competitors which were able to successfully correct all datasets. While for most datasets memory consumption is reasonable, ACE has higher computational cost than most other tools, trading speed for accuracy.

Table 4.2. Time and memory requirements of the tools that were able to successfully correct all datasets.

	Time (s/Mb)				Space (MB/Mb)			
	Musket	**RACER**	**SGA**	**ACE**	**Musket**	**RACER**	**SGA**	**ACE**
M1	0.05	0.55	2.07	10.22	3.64	17.80	10.02	30.92
M2	0.14	0.18	0.85	3.14	4.31	10.00	5.40	5.30
M3	0.07	0.27	1.12	3.76	3.47	7.20	4.52	6.21
M4	0.07	0.35	1.16	3.92	3.18	8.15	3.95	7.01
M5	0.07	0.25	0.56	3.48	2.95	6.24	4.02	5.52
M6	0.07	0.34	1.25	3.90	2.35	6.78	3.10	5.70
M7	0.06	0.23	1.05	3.29	1.53	3.18	2.03	2.73
M8	0.06	0.43	1.82	8.29	0.26	2.24	0.90	12.86
M9	0.06	0.33	1.83	5.64	0.24	1.21	0.75	4.94
Average	0.07	0.32	1.30	5.07	2.44	6.98	3.86	9.02
H1	0.27	0.17	0.87	4.28	4.49	9.40	5.99	5.16
H2	0.23	0.22	0.82	5.78	4.28	9.68	6.10	4.23
H3	0.20	0.16	0.75	5.21	2.93	7.59	4.28	5.12
H4	0.28	0.31	1.20	5.88	1.58	4.03	2.37	5.67
H5	0.20	0.24	0.92	4.77	1.17	3.01	1.73	2.45
H6	0.48	0.25	1.08	3.62	0.66	2.04	1.09	4.48
H7	0.29	0.30	1.11	4.97	0.70	2.83	0.85	6.70
H8	0.27	0.31	1.09	4.56	0.52	2.36	0.56	4.83
H9	0.42	0.32	1.14	4.29	0.61	2.70	0.55	4.75
H10	0.32	0.31	1.22	4.17	0.62	2.26	0.46	4.64
H11	0.24	0.74	1.55	3.93	0.20	1.66	0.23	1.39
H12	0.28	0.47	1.69	3.92	0.19	2.28	0.23	1.36
H13	0.26	0.47	1.62	3.92	0.16	1.37	0.22	1.32
Average	0.29	0.33	1.16	4.56	1.39	3.94	1.90	4.01

4.4 Conclusion

We developed ACE, a command-line tool to accurately correct substitution errors in Illumina short-read archives. ACE generally out- performs the best among seven state-of-the-art read cleaners in terms of coverage depth, at higher computational cost. This makes it a useful tool for small to medium-sized datasets or applications where accuracy requirements warrant the investment in computational resources.

In future work, we aim to lower the runtime of ACE by updating the *K*-mer trie instead of rebuilding it for each round of execution. This future version should also be able to handle InDel errors to extend its application to all sequencing platforms.

4.5 Supplementary methods

4.5.1 *K*-mer trie

Supplementary Figure S4.1 shows the 3-mer trie for the sequence AATCAAT from $\Sigma = \{A, C, G, T\}$. In this trie, for example, AAT occurs 2 times along the sequence. This data

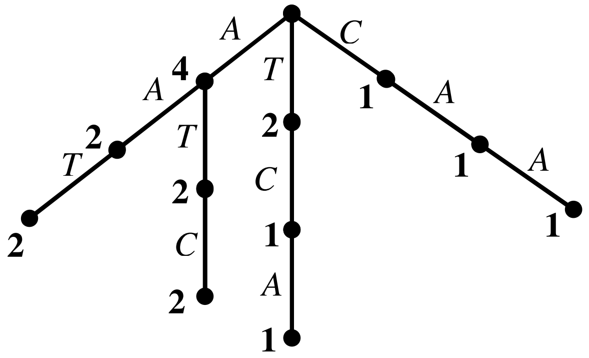


Figure S4.1. The 3-mer trie for the sequence AATCAAT.

structure could similarly be constructed over a set of sequences; in this case we could call it generalized *K*-mer trie; however, for simplicity we excluded the term generalized in the paper.

4.5.2 Algorithms

The pseudo-code of ACE is presented in Supplementary Table 4.S1. The first step is to load the entire read bank into the memory (Line 1). We use 2-bit coding for nucleotides to reduce the occupied memory by 75%. ACE consists of two main consecutive steps, Construct_Subtrie( ) and Detect_and_Correct( ), which iterate in a loop for all prefixes. For each prefix of length p, ACE constructs its corresponding sub-tries (Line 4). The Construct-Subtrie() method applies another prefix-based division to provide the possibility of parallel construction of the branches of a subtrie. It invokes n parallel tasks, each responsible for constructing a sub-branch by inserting *K*-mers into the corresponding sub-branch.

Detect_and_Correct() detects substitutions in parallel by traversing the deepest level of the sub-tries. θ is the frequency below which *K*-mers are classified as low-frequent. It is the frequency at which the spectrum curve reaches its minimum. In each low-frequent *K*-mer, ACE mutates each nucleotide to the three alternatives to see whether it turns into a high-frequent *K*-mer; we call this process the 1-change search. In Supplementary Figure S4.2, consider the low-frequent leaf GCA. To find the correct *K*-mer, the algorithm examines (at most) nine leaves corresponding to GCN, GNA and NCA patterns, where each 'N' is replaced with three other bases different from those appearing in GCA. The 1-change search stops once ACE finds a high-frequent *K*-mer; otherwise, a 2-change search will be launched in the hope of finding an alternative *K*-mer with two mismatches.

Table 4.S1. Pseudo-code of ACE.

```
Main Algorithm
Load the entire (compressed) read bank R into memory
For seven rounds
    Initialize K, p and θ and set k = K-2;
    For each prefix of length p
        ST = Construct_Subtrie( k, prefix )    // The subtrie of depth k  associated with prefix
        Detect_ and_Correct( ST , θ )
        If the available memory falls below a minimum
            Free the whole trie and collect garbage
Store the corrected read bank into a new bank
```

```
Construct_Subtrie(prefix)
Determine n as the number of parallel tasks with regard to the number of available cores
In parallel, invoke n tasks $t_i$, $i = 0,1, 2, … ,4^p$
    For each K-mer in the read bank R beginning with i as prefix (or ending in reverse complement of i)
            insert K-mer into i-branch of the subtrie
Return subtrie
```

```
Detect_and_Correct(subtrie ST , threshold θ )
In parallel, for each i-branch, $i =0,1, 2, … ,4^p$
            For each leaf of this i-branch
                If frequency of leaf is not greater than θ
                    Perform a 1_change search
                    If a high-frequent K-mer is found
                        Correct the erroneous base
                    Else
                        Perform a 2_change search
                        If a high-frequent K-mer is found
                            Correct the two erroneous bases
```

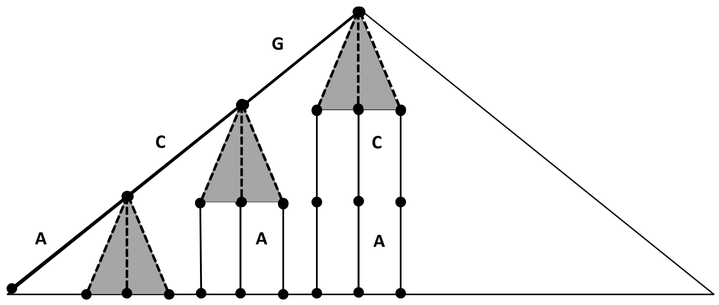


Figure S4.2. The 1-change search tries to find a highly frequent leaf among candidate leaves of the trie which is in Hamming distance 1 to the the low-frequent (erroneous) K-mer.

4.5.3 Choice of parameters

Error correction based on *K*-mer counting relies heavily on high-frequent *K*-mers to correct similar low-frequent ones. Normally, high-frequent i-mers occur due to high coverage, sequencing biases and genomic repeats. They can also occur when *K* is disproportionately small with respect to the length of the genome or the size of the read bank. These kinds of repeats increase the number of wrong corrections and decrease performance. On the other hand, long *K*-mers are more likely to be exposed to more than one error, which hampers their correction.

Consequently, determining an appropriate value for K is vital to the performance of the algorithm. In our experiments, we found the most efficient value for the length of K-mers to be $K = 10 + log_4^G$, where G is the approximate genome length.

The prefix length p is determined by the algorithm according to the amount of available memory and the size of the input data, to guarantee fitting of one sub-trie in the available memory. Θ is initialized by 2 + Coverage / 10. Coverage is computed as $N \times L / G$, where N, L and G are the number of reads, length of reads and genome length, respectively.

4.5.4 Memory management

ACE is equipped with built-in memory management to successfully construct huge tries in limited memory. First, as the full K-mer trie may not fit in the available memory, ACE constructs its sub-tries, one by one, by applying a prefix-based classification on all K-mers. In our experience, consecutive construction of sub-tries also improves the correction performance of the algorithm, since corrections made in one sub-trie reduce the error rate of the read bank and this, in turn, increases the accuracy of detections in the next sub-trie. To save time, ACE reuses the structure of one sub-trie for building the next one reducing the number of dynamic memory allocations. However, as this structure could grow rapidly on large datasets it should be freed before saturating the entire memory. ACE performs this task whenever the amount of free memory falls below a specified threshold.

4.5.5 Supplementary Tables

Table 4.S2 shows the specifications of the 9 MiSeq (M1-M9) and the 13 HiSeq (H1-H13) datasets presented in the survey of Molnar, *et al*. Table 4.S2 gives an outline of the ACE algorithm in pseudo-code. Supplementary Tables 4.S3-9 contain benchmarking results for ACE, compared to results reported for seven state-of-the-art read cleaners recently in a recent survey by Molnar and Ilie (11). Missing results are due to problems in running some tools on certain datasets, because of their inability to deal with varying read length or the algorithms simply taking too long.

Table 4.*S2*. The specifications of the MiSeq (M1-M9) and HiSeq (H1-H13) datasets.

	Organism	Accession number	Read Length	Number of Reads	Total bp	Coverage	Reference Genome	Genome Length
M1	*E.coli*	SRR519926	251	801,192	201,099,192	43	NC_000913.	4,639,675
M2	*M.tuberculosis*	SRR1200797	50-250	1,482,716	348,224,181	79	NC_000962.	4,411,532
M3	*S.enterica*	SRR1203044	35-250	1,784,756	433,166,399	89	NC_011083.	4,888,768
M4	*S.enterica*	SRR1206093	35-251	1,977,970	472,256,906	97	NC_011083.	4,888,768
M5	*L.monocytogenes*	SRR1198952	35-251	2,177,790	507,711,040	171	NC_017546.	2,973,801
M6	*P.syringae*	SRR1119292	35-251	2,576,622	639,853,726	105	NC_007005.	6,093,698
M7	*B.dentium*	SRR1151311	35-251	3,926,618	984,280,778	373	NC_013714.	2,636,367
M8	*E.coli*	SRR522163	251	11,181,452	2,806,544,452	605	NC_000913.	4,639,675
M9	*O.tsutsugamushi*	SRR1202083	301	10,315,434	3,104,945,634	1,460	NC_009488.	2,127,051
H1	*M.tuberculosis*	ERR400373	151	2,092,946	316,034,846	72	NC_000962.	4,411,532
H2	*S.enterica*	ERR230402	100	3,257,972	325,797,200	67	NC_011083.	4,888,768
H3	*S.cerevisiae*	ERR422544	100	4,776,774	477,677,400	40	R64-1-1	12,071,326
H4	*L.pneumophila*	SRR801797	100	8,850,220	885,022,000	260	NC_002942.	3,397,754
H5	*E.coli*	SRR1191655	101	11,726,414	1,184,367,814	255	NC_000913.	4,639,675
H6	*E.coli*	SRR490124	100	21,553,358	2,155,335,800	465	NC_000913.	4,639,675
H7	*C.elegans*	SRX218989	100	31,642,176	3,164,217,600	32	WS222	100,286,070
H8	*C.elegans*	SRR543736	101	57,721,732	5,829,894,932	58	WS222	100,286,070
H9	*D.melanogaster*	SRR823377	100	63,014,762	6,301,476,200	52	Release 5	120,381,546
H10	*D.melanogaster*	SRR988075	101	75,938,276	7,669,765,876	64	Release 5	120,381,546
H11	*Human*	ERX069715	100-102	1,357,751,670	137,132,918,670	43	Build 38	3,209,286,105
H12	*Human*	ERX069504	100-102	1,637,816,924	165,419,509,324	52	Build 38	3,209,286,105
H13	*Human*	ERX069505	101	1,708,169,546	172,525,124,146	54	Build 38	3,209,286,105

Table 4S3. Details of read depth coverage analyses for MiSeq datasets (TP: true positive, FP: false positive, FN: false negative, TN: true negative).

		BLESS	Coral	HiTEC	Musket	RACER	SGA	ACE	SHREC
M1	*TP*	43,668,729	119,225	52,111,616	0	51,728,339	52,111,867	111,171,665	28,948,332
	FP	1,506	0	2,042,136	0	0	0	0	2,442,230
	FN	154,488,743	198,038,247	146,045,856	198,157,472	146,429,133	146,045,605	86,985,807	169,209,140
	TN	2,940,214	2,941,720	899,584	2,941,720	2,941,720	2,941,720	2,941,720	499,490
M2	*TP*		37,441,167		174,089	70,946,701	67,318,527	69,725,941	65,325,070
	FP		10,076		202,786	26,192	19,438	44,908	59,446
	FN		84,701,060		121,968,138	51,195,526	54,823,700	52,416,286	56,817,157
	TN		226,071,878		225,879,168	226,055,762	226,062,516	226,037,046	226,022,508
M3	*TP*		34,078,092		241,956	43,577,951	38,479,415	46,783,362	45,770,835
	FP		32,223		36,655	23,750	23,253	23,999	86,877,007
	FN		298,565,704		332,401,840	289,065,845	294,164,381	285,860,434	286,872,961
	TN		100,490,380		100,485,948	100,498,853	100,499,350	100,498,604	13,645,596
M4	*TP*		137,281,061		1,659,328	167,793,410	148,087,585	187,822,776	81,037,228
	FP		12,628		97,625	4,619	3,113	4,692	169,597,265
	FN		86,559,186		222,180,919	56,046,837	75,752,662	36,017,471	142,803,019
	TN		248,404,031		248,319,034	248,412,040	248,413,546	248,411,967	78,819,394
M5	*TP*		2,179,733		340,451	4,523,655	3,275,149	4,580,433	5,551,003
	FP		97,221		32,888	157,730	84,281	165,402	11,843,552
	FN		493,590,159		495,429,441	491,246,237	492,494,743	491,189,459	490,218,889
	TN		11,843,927		11,908,260	11,783,418	11,856,867	11,775,746	97,596
M6	*TP*		73,904,029		48,804	93,366,114	82,815,864	102,268,102	55,399,341
	FP		29,817		30,085	24,529	23,278	26,269	62,816,188
	FN		493,011,302		566,866,527	473,549,217	484,099,467	464,647,229	511,515,990
	TN		72,908,578		72,908,310	72,913,866	72,915,117	72,912,126	10,122,207
M7	*TP*		234,434,318		0	259,738,331	204,346,520	272,260,950	186,436,927
	FP		13,037		0	4,258	0	251	287,229,870
	FN		67,210,155		301,644,473	41,906,142	97,297,953	29,383,523	115,207,546
	TN		682,623,268		682,636,305	682,632,047	682,636,305	682,636,054	395,406,435
M8	*TP*	917,620,609	142,066	1,073,307,124	0	551,372,704	620,227,777	1,998,156,282	601,186,666
	FP	251	0	41,640,398	0	251	0	0	56,399,449
	FN	1,812,741,07	2,730,219,61	1,657,054,561	2,730,361,685	2,178,988,981	2,110,133,908	732,205,403	2,129,175,019
	TN	76,182,516	76,182,767	34,542,369	76,182,767	76,182,516	76,182,767	76,182,767	19,783,318
M9	*TP*	11,606,259	7,224	20,560,407	0	12,166,721	5,476,996	13,908,909	18,752,902
	FP	6,923	0	8,584,520	0	15,351	4,816	6,923	8,601,677
	FN	3,084,376,49	3,095,975,53	3,075,422,350	3,095,982,757	3,083,816,036	3,090,505,761	3,082,073,848	3,077,229,855
	TN	8,955,954	8,962,877	378,357	8,962,877	8,947,526	8,958,061	8,955,954	361,200

Table 4S4. Details of read depth coverage analyses for HiSeq datasets (TP: true positive, FP: false positive, FN: false negative, TN: true negative).

		BLESS	Coral	HiTEC	Musket	RACER	SGA	ACE	SHREC
H1	*TP*	38,798,393	32,544,879	56,611,108	36,658,572	37,052,380	32,100,184	39,889,368	55,729,872
	FP	82,748	20,687	19,971,411	86,674	9,664	7,248	24,462	23,563,248
	FN	23,933,500	30,187,014	6,120,785	26,073,321	25,679,513	30,631,709	22,842,525	7,002,021
	TN	253,220,205	253,282,266	233,331,542	253,216,279	253,293,289	253,295,705	253,278,491	229,739,705
H2	*TP*	26,092,400	24,447,500	52,661,000	25,192,500	25,157,200	23,636,900	26,460,300	52,589,200
	FP	14,100	218,200	27,131,500	17,800	13,300	12,700	13,100	27,488,700
	FN	32,769,300	34,414,200	6,200,700	33,669,200	33,704,500	35,224,800	32,401,400	6,272,500
	TN	266,921,400	266,717,300	239,804,000	266,917,700	266,922,200	266,922,800	266,922,400	239,446,800
H3	*TP*	29,305,600	25,529,600	49,157,700	28,454,300	29,339,700	28,141,200	29,824,600	48,883,400
	FP	22,600	1,109,200	20,430,600	15,100	8,700	7,200	10,800	21,757,000
	FN	56,640,800	60,416,800	36,788,700	57,492,100	56,606,700	57,805,200	56,121,800	37,063,000
	TN	391,708,400	390,621,800	371,300,400	391,715,900	391,722,300	391,723,800	391,720,200	369,974,000
H4	*TP*	269,792,900	163,889,400	286,237,400	254,853,500	271,647,000	255,971,600	275,909,400	282,598,900
	FP	1,000	4,600	15,327,900	9,100	1,000	100	3,200	20,864,600
	FN	24,626,000	130,529,500	8,181,500	39,565,400	22,771,900	38,447,300	18,509,500	11,820,000
	TN	590,602,100	590,598,500	575,275,200	590,594,000	590,602,100	590,603,000	590,599,900	569,738,500
H5	*TP*	83,739,302	76,462,050	339,677,847	81,013,817	82,654,461	76,475,180	85,475,189	
	FP	181,901	346,733	255,045,907	178,568	182,709	166,650	187,052	
	FN	534,120,320	541,397,572	278,181,775	536,845,805	535,205,161	541,384,442	532,384,433	
	TN	566,326,291	566,161,459	311,462,285	566,329,624	566,325,483	566,341,542	566,321,140	
H6	*TP*	918,033,300	46,606,600	981,535,800	842,711,500	882,718,300	592,452,800	984,066,200	881,970,900
	FP	2,400	0	81,621,300	51,700	7,700	200	2,200	179,178,100
	FN	147,509,600	1,018,936,300	84,007,100	222,831,400	182,824,600	473,090,100	81,476,700	183,572,000
	TN	1,089,790,500	1,089,792,900	1,008,171,600	1,089,741,200	1,089,785,200	1,089,792,700	1,089,790,700	910,614,800
H7	*TP*	523,410,700	473,843,600		424,362,600	529,779,400	533,602,900	552,505,000	
	FP	2,527,500	4,986,700		5,232,200	1,028,700	226,100	5,242,500	
	FN	485,147,600	534,714,700		584,195,700	478,778,900	474,955,400	456,053,300	
	TN	2,153,131,800	2,150,672,600		2,150,427,100	2,154,630,600	2,155,433,200	2,150,416,800	
H8	*TP*	454,178,214	439,078,411		443,450,701	477,389,933	458,849,666	510,661,656	814,419,459
	FP	10,121,513	11,454,208		25,147,485	4,503,691	113,221	19,143,641	940,215,060
	FN	1,312,147,257	1,327,247,060		1,322,874,770	1,288,935,538	1,307,475,805	1,255,663,815	951,906,012
	TN	4,053,447,948	4,052,115,253		4,038,421,976	4,059,065,770	4,063,456,240	4,044,425,820	3,123,354,401
H9	*TP*	654,362,400	397,459,800		609,199,000	617,239,100	580,178,600	709,691,700	1,834,496,800
	FP	39,685,900	26,173,100		40,098,000	10,218,000	3,000,100	20,083,900	1,332,909,300
	FN	2,734,234,500	2,991,137,100		2,779,397,900	2,771,357,800	2,808,418,300	2,678,905,200	1,554,100,100
	TN	2,873,193,400	2,886,706,200		2,872,781,300	2,902,661,300	2,909,879,200	2,892,795,400	1,579,970,000
H10	*TP*	1,224,222,111	1,183,698,891		1,221,214,028	1,199,764,860	1,180,369,022	1,285,325,798	2,514,864,347
	FP	23,094,155	75,877,967		24,294,439	8,466,931	2,325,323	19,902,252	1,391,574,566
	FN	3,372,129,521	3,412,652,741		3,375,137,604	3,396,586,772	3,415,982,610	3,311,025,834	2,081,487,285
	TN	3,050,320,089	2,997,536,277		3,049,119,805	3,064,947,313	3,071,088,921	3,053,511,992	1,681,839,678
H11	*TP*				37,150,982,594	40,075,406,388	42,985,454,686	40,113,218,550	
	FP				15,439,080	8,158,656	9,213,474	23,386,884	
	FN				32,548,080,466	29,623,656,672	26,713,608,374	29,585,844,510	
	TN				67,418,416,530	67,425,696,954	67,424,642,136	67,410,468,726	
H12	*TP*				46,900,867,926	51,634,839,736	55,142,513,014	52,768,565,090	
	FP				15,343,576	10,163,820	10,120,212	32,801,606	
	FN				37,321,732,986	32,587,761,176	29,080,087,898	31,454,035,822	
	TN				81,181,564,836	81,186,744,592	81,186,788,200	81,164,106,806	
H13	*TP*				7,752,423,468	9,660,964,514	10,145,209,317	10,090,726,647	
	FP				19,235,248	4,802,954	3,700,943	9,457,965	
	FN				28,645,673,833	26,737,132,787	26,252,887,984	26,307,370,654	
	TN				136,107,791,597	136,122,223,891	136,123,325,902	136,117,568,880	

Table 4.S5. Details of read breadth coverage analyses for MiSeq datasets (TP: true positive, FP: false positive, FN: false negative, TN: true negative).

		BLESS	**Coral**	**HiTEC**	**Musket**	**RACER**	**SGA**	**ACE**	**SHREC**
M1	*TP*	309,722	451	193,884	0	200,121	201,659	418,923	103,654
	FP	1	0	7	0	0	0	0	1
	FN	4,279,872	4,589,143	4,395,710	4,589,594	4,389,473	4,387,935	4,170,671	4,485,940
	TN	11,671	11,672	11,665	11,672	11,672	11,672	11,672	11,671
M2	*TP*		120,938		385	226,218	214,706	223,470	208,161
	FP		40		1,119	110	78	240	278
	FN		3,459,821		3,580,374	3,354,541	3,366,053	3,357,289	3,372,598
	TN		800,234		799,155	800,164	800,196	800,034	799,996
M3	*TP*		68,946		316	89,002	77,786	95,461	78,077
	FP		166		90	92	90	93	97
	FN		4,509,779		4,578,409	4,489,723	4,500,939	4,483,264	4,500,648
	TN		280,963		281,039	281,037	281,039	281,036	281,032
M4	*TP*		262,268		1,233	322,454	284,073	358,940	278,406
	FP		56		254	19	13	20	26
	FN		3,909,569		4,170,604	3,849,383	3,887,764	3,812,897	3,893,431
	TN		687,754		687,556	687,791	687,797	687,790	687,784
M5	*TP*		2,469		332	3,534	2,918	3,524	3,275
	FP		392		178	658	354	703	663
	FN		2,927,702		2,929,839	2,926,637	2,927,253	2,926,647	2,926,896
	TN		25,188		25,402	24,922	25,226	24,877	24,917
M6	*TP*		161,402		71	203,331	184,119	219,281	150,584
	FP		130		81	94	91	101	95
	FN		5,690,622		5,851,953	5,648,693	5,667,905	5,632,743	5,701,440
	TN		219,680		219,729	219,716	219,719	219,709	219,715
M7	*TP*		277,933		0	309,814	246,419	324,955	291,445
	FP		36		0	17	0	1	29
	FN		812,524		1,090,457	780,643	844,038	765,502	799,012
	TN		1,523,359		1,523,395	1,523,378	1,523,395	1,523,394	1,523,366
M8	*TP*	2,941,766	471	2,357,552	0	1,456,582	1,614,136	3,455,630	1,463,894
	FP	1	0	1	0	0	0	0	2
	FN	1,386,908	4,328,203	1,971,122	4,328,674	2,872,092	2,714,538	873,044	2,864,780
	TN	272,591	272,592	272,591	272,592	272,592	272,592	272,592	272,590
M9	*TP*	16,708	13	8,178	0	7,952	4,905	8,573	7,362
	FP	20	0	105	0	32	16	23	23
	FN	1,712,607	1,729,302	1,721,137	1,729,315	1,721,363	1,724,410	1,720,742	1,721,953
	TN	17,608	17,628	17,523	17,628	17,596	17,612	17,605	17,605

Table 4S6. Details of read breadth coverage analyses for HiSeq datasets (TP: true positive, FP: false positive, FN: false negative, TN: true negative).

		BLESS	**Coral**	**HiTEC**	**Musket**	**RACER**	**SGA**	**ACE**	**SHREC**
H1	*TP*	172,808	135,129	152,688	152,602	153,705	132,858	165,667	134,076
	FP	412	285	407	551	64	48	158	830
	FN	2,875,029	2,912,708	2,895,149	2,895,235	2,894,132	2,914,979	2,882,170	2,913,761
	TN	1,328,069	1,328,196	1,328,074	1,327,930	1,328,417	1,328,433	1,328,323	1,327,651
H2	*TP*	133,118	123,460	122,626	127,407	126,619	118,842	133,424	120,531
	FP	134	2,076	134	175	135	127	132	193
	FN	2,793,086	2,802,744	2,803,578	2,798,797	2,799,585	2,807,362	2,792,780	2,805,673
	TN	1,924,522	1,922,580	1,924,522	1,924,481	1,924,521	1,924,529	1,924,524	1,924,463
H3	*TP*	196,029	179,016	185,749	183,448	192,376	183,348	193,801	175,092
	FP	167	17,345	262	161	102	87	117	208
	FN	8,407,698	8,424,711	8,417,978	8,420,279	8,411,351	8,420,379	8,409,926	8,428,635
	TN	3,096,065	3,078,887	3,095,970	3,096,071	3,096,130	3,096,145	3,096,115	3,096,024
H4	*TP*	328,204	205,969	298,164	312,129	325,945	312,368	329,461	289,456
	FP	12	31	26	73	17	4	39	97
	FN	334,983	457,218	365,023	351,058	337,242	350,819	333,726	373,731
	TN	2,716,828	2,716,809	2,716,814	2,716,767	2,716,823	2,716,836	2,716,801	2,716,743
H5	*TP*	106,699	95,417	100,388	100,828	102,148	94,175	105,717	
	FP	1,791	2,582	1,821	1,774	1,813	1,656	1,859	
	FN	2,160,433	2,171,715	2,166,744	2,166,304	2,164,984	2,172,957	2,161,415	
	TN	2,319,618	2,318,827	2,319,588	2,319,635	2,319,596	2,319,753	2,319,550	
H6	*TP*	773,111	12,358	753,819	724,219	740,486	553,995	797,319	640,984
	FP	12	0	293	339	18	2	15	345
	FN	159,818	920,571	179,110	208,710	192,443	378,934	135,610	291,945
	TN	3,655,470	3,655,482	3,655,189	3,655,143	3,655,464	3,655,480	3,655,467	3,655,137
H7	*TP*	3,795,965	3,376,412		2,977,786	3,730,912	3,733,768	3,835,798	
	FP	19,234	75,288		51,192	10,769	2,414	51,445	
	FN	76,806,449	77,226,002		77,624,628	76,871,502	76,868,646	76,766,616	
	TN	17,767,359	17,711,305		17,735,401	17,775,824	17,784,179	17,735,148	
H8	*TP*	2,815,166	2,634,766		2,558,045	2,786,766	2,678,506	2,951,320	1,875,372
	FP	71,750	181,793		239,556	46,334	1,361	183,835	3,876,572
	FN	64,751,070	64,931,470		65,008,191	64,779,470	64,887,730	64,614,916	65,690,864
	TN	30,768,303	30,658,260		30,600,497	30,793,719	30,838,692	30,656,218	26,963,481
H9	*TP*	4,570,399	2,579,150		3,854,813	3,967,219	3,725,736	4,518,676	3,436,012
	FP	354,948	251,780		384,511	101,129	30,056	195,294	385,819
	FN	89,010,842	91,002,091		89,726,428	89,614,022	89,855,505	89,062,565	90,145,229
	TN	22,000,880	22,104,048		21,971,317	22,254,699	22,325,772	22,160,534	21,970,009
H10	*TP*	7,612,885	7,322,763		7,289,921	7,343,606	7,231,255	7,644,107	6,930,745
	FP	204,628	715,976		233,197	82,864	23,145	191,182	289,134
	FN	85,180,201	85,470,323		85,503,165	85,449,480	85,561,831	85,148,979	85,862,341
	TN	22,948,023	22,436,675		22,919,454	23,069,787	23,129,506	22,961,469	22,863,517
H11	*TP*				256,571,812	276,655,312	296,656,390	276,406,914	
	FP				154,411	79,662	92,403	230,686	
	FN				2,038,938,486	2,018,854,986	1,998,853,908	2,019,103,384	
	TN				566,366,373	566,441,122	566,428,381	566,290,098	
H12	*TP*				302,538,890	331,954,805	354,940,902	328,902,412	
	FP				152,460	98,782	98,799	326,574	
	FN				1,894,657,043	1,865,241,128	1,842,255,031	1,868,293,521	
	TN				664,682,689	664,736,367	664,736,350	664,508,575	
H13	*TP*				45,127,805	56,164,775	59,212,467	56,010,699	
	FP				186,960	50,923	39,752	80,998	
	FN				1,813,725,556	1,802,688,586	1,799,640,894	1,802,842,662	
	TN				1,002,990,761	1,003,126,798	1,003,137,969	1,003,096,723	

Table 4.S7. Details of *K*-mer depth coverage analyses for MiSeq datasets (TP: true positive, FP: false positive, FN: false negative, TN: true negative).

		BLESS	**Coral**	**HiTEC**	**Musket**	**RACER**	**SGA**	**ACE**	**SHREC**
M1	*TP*	21,893,438	34,274	44,305,488	0	34,389,661	8,730,200	54,149,062	36,832,223
	FP	2,394,154	215	10,963,521	0	15,350	1	38,556	14,960,390
	FN	69,978,421	91,837,585	47,566,371	91,871,859	57,482,198	83,141,659	37,722,797	55,039,636
	TN	91,610,531	94,004,470	83,041,164	94,004,685	93,989,335	94,004,684	93,966,129	79,044,295
M2	*TP*		4,856,648		60,059	10,790,468	8,966,676	11,417,744	9,456,749
	FP		2,808		93,922	30,760	3,731	170,176	150,325
	FN		16,887,536		21,684,125	10,953,716	12,777,508	10,326,440	12,287,435
	TN		298,305,585		298,214,471	298,277,633	298,304,662	298,138,217	298,158,068
M3	*TP*		12,828,116		84,335	20,780,510	13,276,357	25,091,651	80,893,557
	FP		32,626		9,436	89,902	27,630	96,836	71,193,733
	FN		95,812,138		108,555,919	87,859,744	95,363,897	83,548,603	27,746,697
	TN		290,583,153		290,606,345	290,525,879	290,588,151	290,518,945	219,422,048
M4	*TP*		22,744,602		317,160	38,874,909	22,062,396	48,969,731	56,686,283
	FP		5,173		9,929	11,534	749	57,469	41,688,332
	FN		43,438,982		65,866,424	27,308,675	44,121,188	17,213,853	9,497,301
	TN		368,486,719		368,481,963	368,480,358	368,491,143	368,434,423	326,803,560
M5	*TP*		7,891,814		362,587	11,280,358	7,850,095	12,357,772	109,894,061
	FP		152,339		28,024	287,169	129,662	291,995	114,632,174
	FN		267,209,287		274,738,514	263,820,743	267,251,006	262,743,329	165,207,040
	TN		191,079,590		191,203,905	190,944,760	191,102,267	190,939,934	76,599,755
M6	*TP*		21,970,326		26,998	40,702,447	21,722,408	50,154,244	133,655,363
	FP		64,750		10,786	182,965	53,775	173,238	102,101,690
	FN		170,189,860		192,133,188	151,457,739	170,437,778	142,005,942	58,504,823
	TN		398,672,972		398,726,936	398,554,757	398,683,947	398,564,484	296,636,032
M7	*TP*		24,827,764		0	33,636,426	19,623,389	40,772,124	49,398,205
	FP		13,269		0	11,284	975	27,619	18,913,350
	FN		26,439,301		51,267,065	17,630,639	31,643,676	10,494,941	1,868,860
	TN		858,394,702		858,407,971	858,396,687	858,406,996	858,380,352	839,494,621
M8	*TP*	340,610,162	9,076,621	585,869,161	0	302,048,660	116,207,551	683,416,830	502,807,883
	FP	18,758,031	104,255	117,448,985	0	507,788	1,176	468,541	153,389,743
	FN	642,417,174	973,950,715	397,158,175	983,027,336	680,978,676	866,819,785	299,610,506	480,219,453
	TN	1,592,311,49	1,610,965,27	1,493,620,543	1,611,069,528	1,610,561,740	1,611,068,352	1,610,600,987	1,457,679,785
M9	*TP*	82,152,087	10,705,146	749,779,497	0	106,159,433	26,886,761	133,143,795	739,501,173
	FP	11,028,358	494,406	638,265,470	0	3,365,501	157,737	1,934,286	644,101,171
	FN	1,578,901,51	1,650,348,45	911,274,106	1,661,053,603	1,554,894,170	1,634,166,842	1,527,909,808	921,552,430
	TN	1,236,870,42	1,247,404,37	609,633,315	1,247,898,785	1,244,533,284	1,247,741,048	1,245,964,499	603,797,614

Table 4.S8. Details of *K*-mer depth coverage analyses for HiSeq datasets (TP: true positive, FP: false positive, FN: false negative, TN: true negative).

		BLESS	**Coral**	**HiTEC**	**Musket**	**RACER**	**SGA**	**ACE**	**SHREC**
H1	*TP*	9,297,926	5,847,153	15,800,449	8,821,881	8,797,946	5,863,914	11,803,953	15,603,537
	FP	532,563	6,129	7,059,095	32,587	37,403	2,819	206,355	8,431,569
	FN	7,841,484	11,292,257	1,338,961	8,317,529	8,341,464	11,275,496	5,335,457	1,535,873
	TN	258,596,899	259,123,333	252,070,367	259,096,875	259,092,059	259,126,643	258,923,107	250,697,893
H2	*TP*	5,853,038	5,006,149	17,606,358	5,562,270	5,777,671	4,870,210	6,466,719	17,598,076
	FP	51,075	25,593	11,495,391	7,397	9,289	5,779	17,637	11,595,626
	FN	12,646,883	13,493,772	893,563	12,937,651	12,722,250	13,629,711	12,033,202	901,845
	TN	245,344,736	245,370,218	233,900,420	245,388,414	245,386,522	245,390,032	245,378,174	233,800,185
H3	*TP*	6,715,796	5,247,704	14,055,870	6,646,972	6,936,005	5,705,931	8,121,552	13,916,387
	FP	332,996	82,154	7,545,806	18,666	53,259	2,954	205,520	7,916,896
	FN	29,023,737	30,491,829	21,683,663	29,092,561	28,803,528	30,033,602	27,617,981	21,823,146
	TN	350,846,165	351,097,007	343,633,355	351,160,495	351,125,902	351,176,207	350,973,641	343,262,265
H4	*TP*	69,781,071	38,997,403	81,651,891	67,090,715	70,508,576	62,676,853	73,897,785	81,292,911
	FP	637,860	3,069	10,324,820	7,841	22,219	518	127,374	12,392,167
	FN	13,321,403	44,105,071	1,450,583	16,011,759	12,593,898	20,425,621	9,204,689	1,809,563
	TN	633,127,486	633,762,277	623,440,526	633,757,505	633,743,127	633,764,828	633,637,972	621,373,179
H5	*TP*	24,408,883	19,637,794	202,987,312	23,116,555	24,041,508	19,640,577	26,088,041	
	FP	350,952	123,677	177,607,518	95,514	114,545	82,165	142,374	
	FN	249,341,042	254,112,131	70,762,613	250,633,370	249,708,417	254,109,348	247,661,884	
	TN	687,465,071	687,692,346	510,208,505	687,720,509	687,701,478	687,733,858	687,673,649	
H6	*TP*	192,667,583	7,083,845	232,202,234	174,081,685	189,580,164	100,690,680	218,182,920	222,018,320
	FP	565,059	23,580	32,102,623	36,908	58,752	3,578	196,475	58,427,446
	FN	48,918,078	234,501,816	9,383,427	67,503,976	52,005,497	140,894,981	23,402,741	19,567,341
	TN	1,503,671,278	1,504,212,757	1,472,133,714	1,504,199,429	1,504,177,585	1,504,232,759	1,504,039,862	1,445,808,891
H7	*TP*	103,291,755	85,423,973		91,520,472	111,079,256	98,206,533	121,727,928	
	FP	5,741,744	410,492		1,589,044	1,671,795	83,418	5,390,550	
	FN	127,788,034	145,655,816		139,559,317	120,000,533	132,873,256	109,351,861	
	TN	2,326,194,723	2,331,525,975		2,330,347,423	2,330,264,672	2,331,853,049	2,326,545,917	
H8	*TP*	107,017,510	85,281,799		111,060,650	120,723,993	89,174,601	143,720,203	388,443,620
	FP	8,224,557	567,682		4,284,718	1,991,390	34,971	6,825,599	368,431,665
	FN	888,990,088	910,725,799		884,946,948	875,283,605	906,832,997	852,287,395	607,563,978
	TN	3,728,949,869	3,736,606,744		3,732,889,708	3,735,183,036	3,737,139,455	3,730,348,827	3,368,742,761
H9	*TP*	202,317,831	99,164,403		194,063,684	196,857,022	150,344,610	241,025,350	929,764,788
	FP	35,375,270	8,821,549		15,861,059	9,514,193	1,102,157	13,141,006	764,976,340
	FN	954,636,724	1,057,790,152		962,890,871	960,097,533	1,006,609,945	915,929,205	227,189,767
	TN	3,911,865,897	3,938,419,618		3,931,380,108	3,937,726,974	3,946,139,010	3,934,100,161	3,182,264,827
H10	*TP*	303,785,349	279,374,787		311,092,393	307,269,099	270,168,655	343,538,697	1,223,613,081
	FP	23,911,995	24,705,560		9,773,448	7,396,974	1,000,241	11,039,941	938,030,041
	FN	1,230,288,285	1,254,698,847		1,222,981,241	1,226,804,535	1,263,904,979	1,190,534,937	310,460,553
	TN	4,668,953,003	4,668,159,438		4,683,091,550	4,685,468,024	4,691,864,757	4,681,825,057	3,754,834,957
H11	*TP*				2,384,660,385	2,521,496,564	2,047,453,498	3,091,834,935	
	FP				22,844,814	71,143,582	3,819,127	122,117,486	
	FN				4,044,853,283	3,908,017,104	4,382,060,170	3,337,678,733	
	TN				104,883,278,458	104,834,979,690	104,902,304,145	104,784,005,786	
H12	*TP*				2,965,425,250	3,390,896,152	2,796,375,723	3,943,946,512	
	FP				33,008,853	93,872,959	3,724,003	155,846,813	
	FN				4,807,599,482	4,382,128,580	4,976,652,182	3,829,078,220	
	TN				126,494,954,183	126,434,090,077	126,524,235,860	126,372,116,223	
H13	*TP*				2,193,391,842	2,542,284,172	1,951,397,719	2,942,291,002	
	FP				30,449,373	95,993,401	3,649,395	161,168,357	
	FN				4,699,768,996	4,350,876,666	4,941,766,326	3,950,869,836	
	TN				133,146,292,561	133,080,748,533	133,173,089,332	133,015,573,577	

Table 4.S9. Details of *K*-mer breadth coverage analyses for MiSeq datasets (TP: true positive, FP: false positive, FN: false negative, TN: true negative).

		BLESS	Coral	HiTEC	Musket	RACER	SGA	ACE	SHREC
M1	*TP*	0	0	0	0	5	0	3	4
	FP	250	0	1,604	0	22	0	272	151
	FN	133	133	133	133	128	133	130	129
	TN	4,541,767	4,542,017	4,540,413	4,542,017	4,541,995	4,542,017	4,541,745	4,541,866
M2	*TP*		5		48	43	13	244	170
	FP		955		44	4,038	2,424	7,192	4,052
	FN		60,527		60,484	60,489	60,519	60,288	60,362
	TN		4,262,574		4,263,485	4,259,491	4,261,105	4,256,337	4,259,477
M3	*TP*		51		30	259	9	201	1,009
	FP		21,214		96	30,752	23,649	32,008	30,310
	FN		811,216		811,237	811,008	811,258	811,066	810,258
	TN		3,986,045		4,007,163	3,976,507	3,983,610	3,975,251	3,976,949
M4	*TP*		2		0	8	0	2	32
	FP		425		0	731	438	1,132	1,131
	FN		120,125		120,127	120,119	120,127	120,125	120,095
	TN		4,697,974		4,698,399	4,697,668	4,697,961	4,697,267	4,697,268
M5	*TP*		84		73	363	235	364	1,644
	FP		109,746		13,976	182,441	96,309	198,528	173,136
	FN		1,464,470		1,464,481	1,464,191	1,464,319	1,464,190	1,462,910
	TN		1,364,326		1,460,096	1,291,631	1,377,763	1,275,544	1,300,936
M6	*TP*		182		99	917	21	778	3,764
	FP		39,520		40	66,792	42,902	69,642	65,059
	FN		1,285,818		1,285,901	1,285,083	1,285,979	1,285,222	1,282,236
	TN		4,654,927		4,694,407	4,627,655	4,651,545	4,624,805	4,629,388
M7	*TP*		0		0	1	0	1	1
	FP		1		0	13	3	6	14
	FN		195		195	194	195	194	194
	TN		2,604,046		2,604,047	2,604,034	2,604,044	2,604,041	2,604,033
M8	*TP*	0	0	9	0	0	0	3	4
	FP	8	0	3	0	6	1	9	7
	FN	120	120	111	120	120	120	117	116
	TN	4,542,022	4,542,030	4,542,027	4,542,030	4,542,024	4,542,029	4,542,021	4,542,023
M9	*TP*	1,046	2,926	8,255	0	3,433	381	5,221	9,386
	FP	126,501	8,311	136,797	0	136,470	48,656	150,337	126,496
	FN	314,851	312,971	307,642	315,897	312,464	315,516	310,676	306,511
	TN	818,703	936,893	808,407	945,204	808,734	896,548	794,867	818,708

Table 4.S10. Details of *K*-mer breadth coverage analyses for HiSeq datasets (TP: true positive, FP: false positive, FN: false negative, TN: true negative).

		BLESS	Coral	HiTEC	Musket	RACER	SGA	ACE	SHREC
H1	*TP*	61	0	295	11	36	3	266	225
	FP	2,003	1,100	4,222	2,546	1,932	1,322	4,992	2,853
	FN	63,068	63,129	62,834	63,118	63,093	63,126	62,863	62,904
	TN	4,258,929	4,259,832	4,256,710	4,258,386	4,259,000	4,259,610	4,255,940	4,258,079
H2	*TP*	0	0	21	5	34	15	33	28
	FP	5,598	5,627	5,588	5,449	5,584	5,105	5,791	5,570
	FN	293,703	293,703	293,682	293,698	293,669	293,688	293,670	293,675
	TN	4,519,225	4,519,196	4,519,235	4,519,374	4,519,239	4,519,718	4,519,032	4,519,253
H3	*TP*	10	55	176	78	46	0	67	187
	FP	2,138	10,170	2,674	1,934	2,268	1,756	2,450	2,215
	FN	198,661	198,616	198,495	198,593	198,625	198,671	198,604	198,484
	TN	11,216,232	11,208,200	11,215,696	11,216,436	11,216,102	11,216,614	11,215,920	11,216,155
H4	*TP*	17	0	2	11	2	8	1	7
	FP	143	30	621	176	225	95	2,615	399
	FN	35,452	35,469	35,467	35,458	35,467	35,461	35,468	35,462
	TN	3,326,335	3,326,448	3,325,857	3,326,302	3,326,253	3,326,383	3,323,863	3,326,079
H5	*TP*	77	25	360	167	173	73	100	
	FP	83,918	72,191	85,300	82,308	84,682	71,129	90,839	
	FN	722,838	722,890	722,555	722,748	722,742	722,842	722,815	
	TN	3,735,317	3,747,044	3,733,935	3,736,927	3,734,553	3,748,106	3,728,396	
H6	*TP*	0	0	13	1	0	0	0	3
	FP	42	0	107	204	55	4	155	112
	FN	1,910	1,910	1,897	1,909	1,910	1,910	1,910	1,907
	TN	4,540,198	4,540,240	4,540,133	4,540,036	4,540,185	4,540,236	4,540,085	4,540,128
H7	*TP*	3,832	1,222		18,197	11,401	3,262	24,312	
	FP	170,571	89,004		458,444	262,825	68,264	822,654	
	FN	3,267,733	3,270,343		3,253,368	3,260,164	3,268,303	3,247,253	
	TN	86,888,901	86,970,468		86,601,028	86,796,647	86,991,208	86,236,818	
H8	*TP*	931	909		10,499	4,241	113	10,491	16,995
	FP	190,578	55,384		907,452	432,166	14,235	1,405,121	1,435,642
	FN	495,905	495,927		486,337	492,595	496,723	486,345	479,841
	TN	89,643,623	89,778,817		88,926,749	89,402,035	89,819,966	88,429,080	88,398,559
H9	*TP*	5,029	3,296		30,175	12,532	1,609	18,015	54,541
	FP	2,330,548	1,271,660		2,632,483	1,329,931	439,431	2,100,960	2,610,962
	FN	7,678,630	7,680,363		7,653,484	7,671,127	7,682,050	7,665,644	7,629,118
	TN	102,855,170	103,914,058		102,553,235	103,855,787	104,746,287	103,084,758	102,574,756
H10	*TP*	2,055	10,548		16,401	8,070	912	8,612	32,813
	FP	1,277,631	2,038,716		1,570,921	1,003,571	337,009	1,654,216	1,759,713
	FN	5,178,012	5,169,519		5,163,666	5,171,997	5,179,155	5,171,455	5,147,254
	TN	106,411,679	105,650,594		106,118,389	106,685,739	107,352,301	106,035,094	105,929,597
H11	*TP*				114,557	117,413	14,610	142,200	
	FP				1,681,767	1,626,134	1,231,081	3,027,118	
	FN				24,778,064	24,775,208	24,878,011	24,750,421	
	TN				2,165,531,157	2,165,586,790	2,165,981,843	2,164,185,806	
H12	*TP*				124,522	131,834	20,816	209,891	
	FP				1,747,295	1,755,678	1,299,126	3,345,399	
	FN				22,575,726	22,568,414	22,679,432	22,490,357	
	TN				2,167,658,002	2,167,649,619	2,168,106,171	2,166,059,898	
H13	*TP*				180,126	194,414	22,569	213,213	
	FP				1,777,300	1,983,440	1,403,458	3,823,523	
	FN				31,783,772	31,769,484	31,941,329	31,750,685	
	TN				2,158,364,347	2,158,158,207	2,158,738,189	2,156,318,124	

Table 4.S11. Details of *K*-mer breadth coverage analyses for MiSeq datasets. Coverage depth/breadth of reads/K-mers of ACE, compared to that of the best tool in (11). Highlights indicate the level of improvement.

	Coverage Depth						**Coverage breadth**					
	Read			**Kmer**			**Read**			**Kmer**		
	Best		**ACE**	**Best**		**ACE**	**Best**		**ACE**	**Best**		**ACE**
M1	SGA	27.38	56.74	RACER	69.0	79.68	BLESS	6.98	9.36	SGA	100.00	99.99
M2	RACER	85.29	84.93	RACER	96.5	96.72	RACER	23.43	23.36	Coral	98.58	98.44
M3	RACER	33.26	34.00	RACER	77.9	79.05	RACER	7.61	7.75	Coral	82.72	82.50
M4	RACER	88.13	92.37	RACER	93.7	96.03	RACER	20.79	21.54	Coral	97.50	97.48
M5	RACER	3.21	3.22	RACER	43.3	43.59	RACER	0.963	0.961	SGA	46.89	43.42
M6	RACER	25.99	27.38	RACER	74.3	75.94	RACER	6.97	7.23	Coral	77.84	77.35
M7	RACER	95.74	97.01	RACER	98.0	98.84	RACER	70.13	70.71	Coral	99.99	99.99
M8	HiTEC	39.47	73.91	HiTEC	80.1	88.43	BLESS	69.85	81.03	HiTEC	100.00	100.0
M9	RACER	0.68	0.74	HiTEC	46.7	47.41	BLESS	1.96	1.50	Coral	74.52	63.44
H1	BLESS	92.40	92.76	Musket	96.9	97.99	BLESS	34.29	34.14	Coral	98.51	98.43
H2	BLESS	89.94	90.05	HiTEC	95.3	95.43	BLESS	42.417	42.424	SGA	93.80	93.79
H3	RACER	88.15	88.25	RACER	92.5	92.81	BLESS	28.14	28.12	SGA	98.24	98.23
H4	RACER	97.43	97.91	HiTEC	98.3	98.70	BLESS	90.09	90.13	Coral	98.94	98.87
H5	BLESS	54.98	55.03	HiTEC	74.1	74.23	BLESS	52.88	52.85	SGA	82.52	82.09
H6	BLESS	93.16	96.22	HiTEC	97.6	98.65	BLESS	96.51	97.04	Coral	99.96	99.95
H7	SGA	84.98	85.42	RACER	95.2	95.52	BLESS	21.916	21.924	SGA	96.31	95.49
H8	RACER	77.81	78.13	RACER	81.4	81.85	BLESS	34.13	34.15	SGA	99.43	97.91
H9	BLESS	55.98	57.17	RACER	81.0	81.80	BLESS	22.92	23.01	SGA	92.80	91.35
H1	BLESS	55.73	56.57	Musket	80.2	80.70	BLESS	26.36	26.40	SGA	95.11	93.95
H1	RACER	80.51	78.41	RACER	96.4	96.89	SGA	30.16	29.44	SGA	98.81	98.73
H1	RACER	82.41	80.97	RACER	96.6	97.03	SGA	35.63	34.71	SGA	98.91	98.82
H1	RACER	84.78	84.75	RACER	96.8	97.06	SGA	37.12	37.01	SGA	98.48	98.38
	Averag	65.34	68.73		84.6	86.11		34.60	35.22		92.27	91.30

References

1. Schröder J, Schröder H, Puglisi SJ, Sinha R, Schmidt B. SHREC: a short-read error correction method. *Bioinformatics*. 2009;25(17):2157–63.
2. Ilie L, Fazayeli F, Ilie S. HiTEC: accurate error correction in high-throughput sequencing data. *Bioinformatics*. 2011;27(3):295–302.
3. Salmela L. Correction of sequencing errors in a mixed set of reads. *Bioinformatics*. 2010;26(10):1284–90.
4. Simpson JT, Durbin R. Efficient de novo assembly of large genomes using compressed data structures. *Genome Res*. 2012;22(3):549–56.
5. Heo Y, Wu X-L, Chen D, Ma J, Hwu W-M. BLESS: bloom filter-based error correction solution for high-throughput sequencing reads. *Bioinformatics*. 2014;30(10):1354–62.
6. Ilie L, Molnar M. RACER: Rapid and accurate correction of errors in reads. *Bioinformatics*. 2013;29(19):2490–3.
7. Schulz MH, Weese D, Holtgrewe M, Dimitrova V, Niu S, Reinert K, et al. Fiona: a parallel and automatic strategy for read error correction. *Bioinformatics*. 2014;30(17):i356–63.
8. Kelley DR, Schatz MC, Salzberg SL. Quake: quality-aware detection and correction of sequencing errors. *Genome Biol*. 2010;11(11):R116.
9. Liu Y, Schröder J, Schmidt B. Musket: a multistage k-mer spectrum-based error corrector for Illumina sequence data. *Bioinformatics*. 2013;29(3):308–15.
10. Salmela L, Schroder J. Correcting errors in short reads by multiple alignments. *Bioinformatics*. 2011;27(11):1455–61.
11. Molnar M, Ilie L. Correcting Illumina data. *Brief Bioinform*. 2015;16(4):588–99.
12. Brudno M. LAGAN and Multi-LAGAN: efficient tools for large-scale multiple alignment of genomic DNA. *Genome Res*. 2003;13(4):721–31.
13. Sheikhizadeh S, Hosseini S. SMOTER, a structured motif finder based on an exhaustive tree-based algorithm. *Curr Bioinform*. 2014;9(1):34–43.

Chapter 5

Pan-genomic read mapping

This chapter is published as:
Sheikhizadeh S, de Ridder D, Schranz ME, Smit S. Pan-genomic read mapping. *bioRxiv*. 2019.
DOI: 10.1101/813634

Abstract

In modern genomics, mapping reads to a single reference genome is common practice. However, a reference genome does not necessarily accurately represent a population or species and as a result a substantial percentage of reads often cannot be mapped. A number of graph-based variation-aware mapping methods have recently been proposed to remedy this. Here, we present an alternative multi-reference approach, which aligns reads to large collections of genomes simultaneously. Our pan-genomic approach is implemented as extension to our pan-genomics suite PanTools*. Through direct comparisons to state-of-the-art tools, we show that it is as accurate and more efficient on large numbers of genomes. We successfully applied PanTools to map genomic and metagenomic reads to large collections of viral, archaeal, bacterial, fungal and plant genomes. Pan-genomic read mapping resolves the reference bias in mapping approaches, by including regions that are entirely missing in the reference (but present in another accession or strain) or very different from the reference. This enables a more extensive analysis of the genetic makeup of non-reference species or strains/accessions.

* https://git.wur.nl/bioinformatics/pantools

5.1 Introduction

Mapping short reads against a reference genome is the starting point of almost all quantitative and comparative genomics pipelines [1]. However, it suffers from a systematic bias towards the reference alleles (often referred to as reference bias): reads which are highly polymorphic or totally absent in the reference are discarded. This means variants that could be of great value, for example in disease diagnostics in humans or resistance genes in plants, are potentially overlooked. Mapping reads against multiple genomes representing a genus, a species or a population, i.e. a pan-genome, partially addresses this problem, allowing the detection of variants that would not be detected using a single reference. A variety of graph-based pan-genomic approaches have therefore emerged recently, which can generally be categorized as either variation-aware or multi-reference. Variation-aware mapping is useful for applications in which variation between individuals is limited and measured extensively, such as in human genetics [2]. Multi-reference approaches in contrast are more suited for applications in which more divergent individuals are studied, such as in comparative genomics [3].

BWBBLE [4] is a variation-aware method that maps reads against a BWT-indexed linear multi-genome reference, built from one reference and a set of variant (VCF) files. Graphtyper [5] iteratively enriches a variation-aware graph with known or discovered variants for read mapping and population-scale genotyping. Similarly, the variation graph toolkit [6] constructs a bi-directed variation-aware graph as a reference to improve the accuracy of mapping, specifically in highly polymorphic regions. GenomeMapper [7] is the first multi-reference approach that represents a reference genome and its differences to a set of other genomes in a hash-based graph structure against which reads can be aligned. GCSA [8] converts a multiple sequence alignment (MSA) of genomes into a finite automaton which is BWT-indexed to allow pattern search. PanVC [9] uses the MSA as a pan-genome reference and map reads against the matrix, where the heaviest path serves as an *ad hoc* reference to improve the accuracy of downstream variant callers.

These approaches, mostly targeting the human genome and/or focusing on specific variable regions, demonstrate that using broader reference representations can improve read mapping. However, they do not suffice to study collections of individual genomes of highly dynamic species such as fungi and plants. In such collections, genome co-linearity is often not preserved; moreover, scalability becomes an issue as the number of genomes grows. To tackle these issues, we propose a multi-reference read mapping approach, as an extension to PanTools. Briefly, PanTools is a suite of tools for large-scale comparative analysis building on a pan-genome representation stored in a graph database [10]. Our mapping method can align millions of short reads to hundreds of eukaryotic or thousands of prokaryotic genomes simultaneously, producing one SAM/BAM file per genome. It also provides a competitive mapping mode, which is useful for abundance estimation and binning in metagenomics samples. We demonstrate that PanTools is as accurate as the state-of the-art read mappers and per-genome mapping time decreases with increasing numbers of genomes.

5.2 Results

We have extended our pan-genome tool suite, PanTools, with a method to efficiently map genomic reads against multiple genomes in a graph-based representation (the algorithm is described under Methods). Conceptually, this eliminates the strong reference bias, which stems from mapping to a single genome. Reads that do not map on one genome may map on another genome, yielding a more complete picture of the genomic makeup of a sample. In application, PanTools offers two advantages that allow mapping efficiency to improve as the number of genomes grows. First, a single joint *k*-mer index is available for all genomes, resulting in fast identification of candidate hits that can then be targeted for full alignment. Second, redundant sequence alignments are avoided by recording previous alignments; if two genomes are similar, fewer alignments have to be made.

PanTools features two modes of read mapping. In 'normal' mode, genomic reads are independently mapped against all genomes in the pan-genome, identifying the most likely mapping location of each read in each genome. In contrast, in 'competitive' mode reads are mapped to the most likely location in the entire pan-genome, such that a read with the highest mapping score on genome A will not be mapped to genome B with a lower score. Competitive mapping is useful in various applications involving mixed samples, such as metagenomics samples, pathogen/host samples, or nuclear/organellar samples.

Here we present the performance of PanTools as a multi-genome read mapper on various sets of simulated and real data from bacteria, fungi and plants. We describe the accuracy and runtime of our approach, compared to a number of other read mappers. In addition, we present two use cases, demonstrating scalability to large genomes and application in metagenomics.

5.2.1 PanTools is as effective as current read mappers

To learn about accuracy and speed of pan-genomic read mapping compared to state-of-the-art single-reference mappers, we simulated read data from two Illumina platforms (HiSeq 2500 and MiSeq v3) and mapped these against the reference genomes of *E. coli* and *S. cerevisiae*. There is often a trade-off between runtime and accuracy of read-mappers, i.e. more accurate results can be attained by using more sensitive settings at the cost of a higher runtime.

While the speed advantage of PanTools becomes apparent in a multi-genome context, this experiment demonstrates that, even on a single genome, PanTools achieves comparable speed and accuracy as widely used methods. Figure 5.1 shows time-accuracy plots of five read mappers (running with default settings on a single processing core): PanTools; two

BWT-based methods, BWA-MEM [11] and Bowtie2 [12]; and two hash-based mappers, Stampy [13] and NextGenMap [14]. There is much more variation in running time than in accuracy, mostly caused by Stampy and Bowtie2 deviating from BWA-MEM, NextGenMap and PanTools, which are highly comparable in terms of speed and accuracy (Additional file 1: Experiment 1).

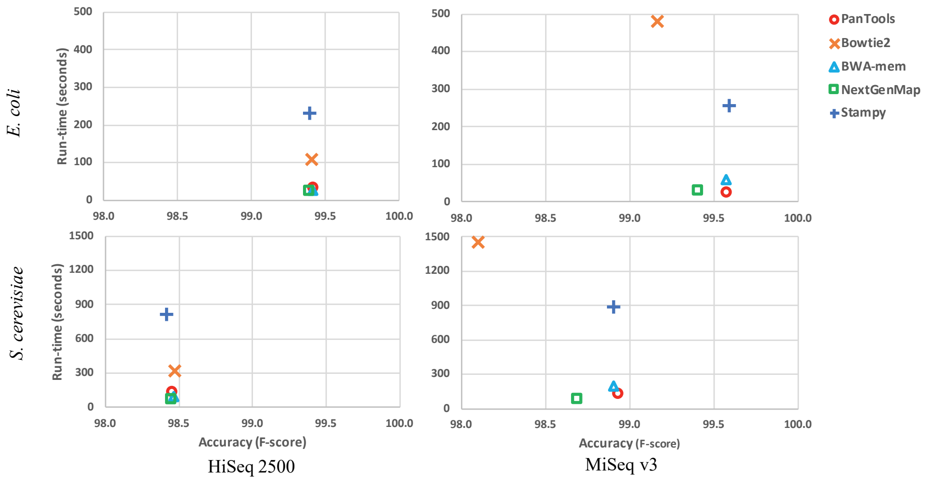


Figure 5.1. Runtime versus accuracy plots of five read mappers on four simulated Illumina datasets shows that PanTools is as accurate as the other tools and much faster than Stampy and Bowtie2 especially on MiSeq data. Accuracy is presented in terms of the F-score (see Methods).

5.2.2 The pan-genomic approach becomes more efficient as the number of genomes grows

We compared the scalability of the best performing tools (PanTools, BWA-MEM and NextGenMap) to large sets of genomes. To learn about the effect of evolutionary distance between genomes, we mapped simulated *S. cerevisiae* reads against four pan-genomes of ten fungi chosen at the levels of strain (ST), species (SP), genus (GN) and family (FM) (see Additional file 1: Experiment 2). Figure 5.2A shows the average runtime per genome of mapping reads against 1-10 genomes. For PanTools, as a multi-genome read mapper, this time is calculated as the total runtime divided by the number of genomes in each experiment. For the singe-genome read mappers, BWA-MEM and NextGenMap, it reflects the average of runtimes up until each point. All tools were running with 8 threads.

In PanTools the runtime per genome decreased when the number of genomes in the pan-genome increased; the more related the genomes were, the higher the speedup. The runtime of BWA-MEM was very stable, around 30 seconds per genome, whereas that of NextGenMap radically increased as more divergent genomes were included in the set. All the tools had very similar mapping percentages at the strain and species levels, yet NexGenMap had the highest mapping percentage on diverged genomes at the genus and family levels (Figure 5.2B), correlated with its high runtime. The mapping percentage of PanTools can likewise be increased (at the cost of a higher runtime) through parameter settings. However, we chose less sensitive default settings, because in many applications read mapping is limited to the species level. PanTools parameters and settings are described in detail in the Methods.

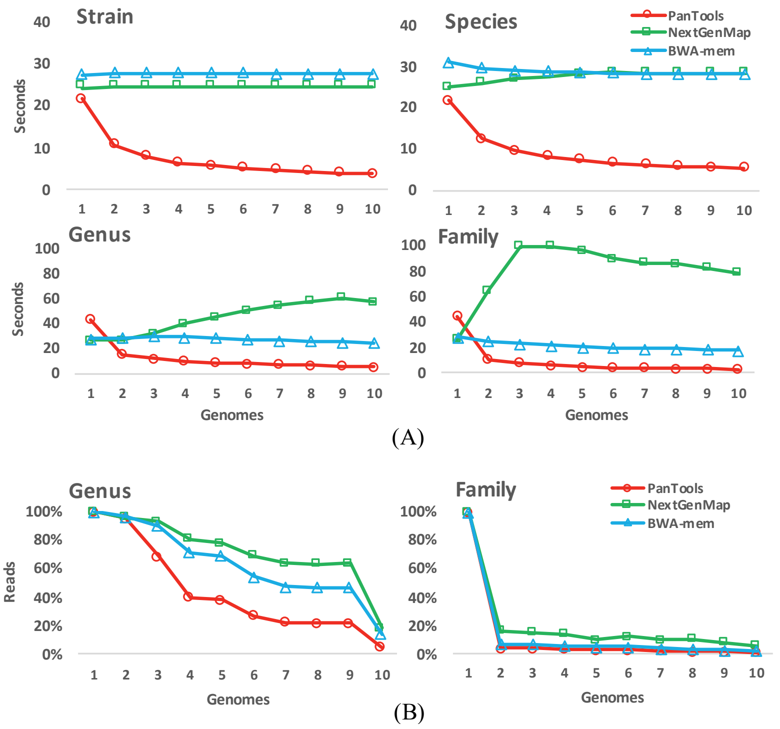


Figure 5.2. Mapping simulated reads from *S. cerevisiae* strain S288C on four pan-genomes of ten fungi. **(A)** Runtime of three methods at the level of strain S288C, species S. cerevisiae, genus Saccharomyces, and family Saccharomycetaceae. Genomes are sorted in decreasing order by the average number of reads that the tools managed to map, roughly reflecting their similarity to the reference genome of S. cerevisiae. **(B)** The mapping percentage depends on the similarity between the sequenced strain and the reference genome and on the default sensitivity of the methods.

PanTools' speed is largely due to avoiding redundant alignments by maintaining a list of previous alignments. When the constituent genomes are closely related, the chance of finding an alignment in this list is high. Table 5.1 shows that this approach saves computations in this experiment, in particular when genomes are highly similar.

Table 5.1. PanTools avoids redundant sequence alignments.

Taxonomy level	Candidate hits	Alignments performed	Alignments avoided
Strain S288C (ST)	12,767,990	1,030,981	91.9%
Species *S. cerevisiae* (SP)	9,041,689	4,773,482	47.2%
Genus *Saccharomyces* (GN)	5,611,299	5,015,910	10.6%
Family Saccharomycetaceae (FM)	2,437,554	2,175,803	10,7%

5.2.3 Use case 1: Pan-genomic read mapping in plants

To illustrate the utility of mapping to a panel of genomes rather than to a single reference, we applied PanTools to a case typically encountered in plant genomics: mapping reads of various (often relatively distant) accessions to a reference genome. We started with the model plant, *Arabidopsis thaliana*, and mapped reads of accession DJA-1 (Illumina library ERR2721960) to the reference genome Col-0 (both nuclear and organellar sequences). In this experiment, 6.25% of reads could not be mapped to the reference, potentially preventing the discovery of important variants. We next mapped these unmapped reads to the pan-genome of 19 accessions of *A. thaliana* [15] and managed to map 4-6% of these on other accessions. Clustering these reads based on overlap reveals moderately covered genomic regions (coverage ≥ 10), which are either absent from or highly different in the reference Col-0. We detected from 476 to 915 of such regions with sizes ranging from 151 to 6,124 base pairs in the other accessions (Table 5.2).

Table 5.2. Reads unmapped to the *A. thaliana* reference Col-0 re-aligned to a pan-genome of 19 accessions. Around 5% of reads mapped to the other accessions could be assembled into regions which are possibly absent or highly polymorphic in the reference of *A. thaliana*.

Genome	Accession	Number of reads	Mapping percentage	Number of regions	Length of regions (min-avg-max)
1	Col-0	-	-	-	-
2	Bur-0	95,362	5.72	803	(151-439-2917)
3	Can-0	100,220	6.01	914	(151-398-2917)
4	Ct-1	72,973	4.38	478	(151-461-2917)
5	Edi-0	94,400	5.66	815	(151-429-6124)
6	Hi-0	68,272	4.10	485	(151-409-2918)
7	Kn-0	92,001	5.52	778	(151-404-2918)
8	Ler-0	91,870	5.51	780	(151-407-5000)
9	Mt-0	73,848	4.43	564	(151-386-2574)
10	No-0	85,449	5.13	710	(151-437-3306)
11	Oy-0	70,119	4.21	476	(151-447-3103)
12	Po-0	72,800	4.37	590	(151-389-2085)
13	Rsch-4	84,523	5.07	706	(151-434-2917)
14	Sf-2	97,827	5.87	915	(151-397-2897)
15	Tsu-0	90,339	5.42	780	(151-424-2903)
16	Wil-2	93,771	5.63	798	(151-406-2903)
17	Ws-0	90,897	5.45	745	(151-437-2917)
18	Wu-0	81,783	4.91	652	(151-447-2917)
19	Zu-0	90,171	5.41	759	(151-426-2917)

As an example, we found a region of 248 base pairs absent in the reference Col-0 (first row, position Chr1:24,201,231) as well as in accession Wil-2 (position Chr1:23,713,822), but present in all other accessions. Figure 5.3 shows the multiple sequence alignment of this region in all accessions and reads which failed to be mapped against this region in Col-0 and Wil-2 but mapped to the other accessions. Similarly, Figure 5.4 shows such an alignment for a region of length 283bp in the sequenced DJA-1 individual, which is highly variable in all accessions. None of the 36 reads covering this region were mapped to Col-0, however between 24-33 reads were mapped to the other 18 accessions. There were 24 SNPs and 7 short indels in the alignment of the assembled reads and the corresponding region in Col-0, which explains the read mapping problems. This experiment shows that even in a rather complete reference genome like for Arabidopsis, potentially important allelic variants are missed using a biased reference-based approach.

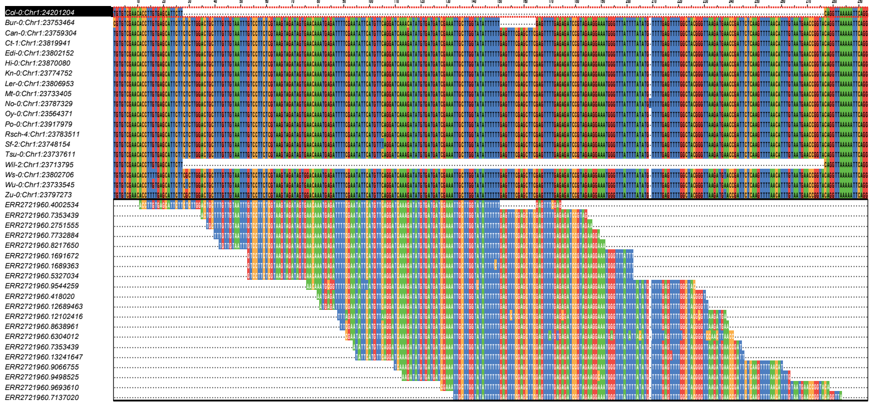


Figure 5.3. A deletion of 248 bp in chromosome 1 of the reference Col-0 and accession Wil-2 detected by mapping reads to the pan-genome.

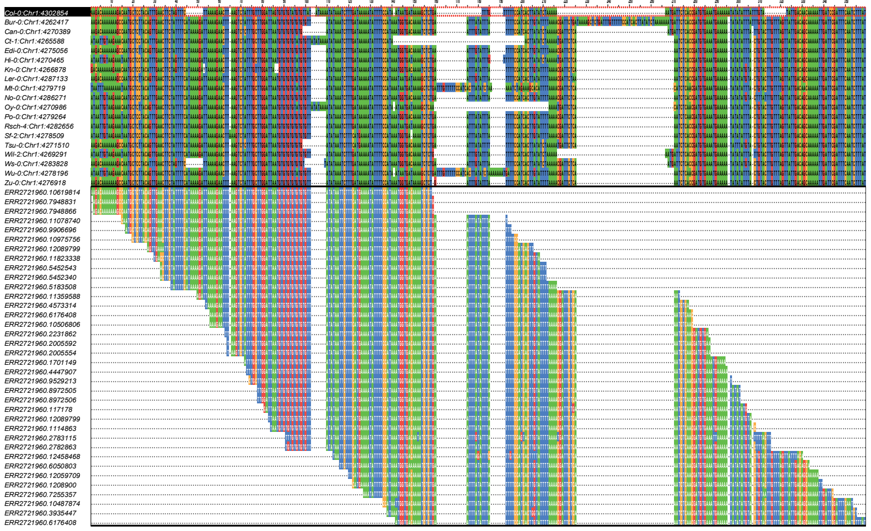


Figure 5.4. A highly variable region starting at Chr1:4,302,854 in the reference Col-0 detected through pan-genomic read mapping. The large number of SNPs and indels in Col-0 prohibits alignment of reads that can be mostly mapped to the other accessions.

Next, we investigated whether PanTools' read mapping would scale to larger, more complex genomes. To this end, we mapped three large paired-end sequencing libraries from the 150 Tomato Genome Resequencing Project [3] to the reference genome of tomato *Solanum lycopersicum* (Heinz 1706) and the three additional species *Solanum pennellii* (LA716), *Solanum pimpinellifolium* (LA480), and *Solanum habrochaites* (LYC4). PanTools achieved a high mapping percentage and additionally captured a large number of regions absent in the reference, or present but highly variable (Table 5.3).

Table 5.3. Mapping reads of three libraries unmapped to the Heinz tomato reference genome to a pan-genome of four tomato accessions detected large number of regions which are absent or highly variable in the reference. The number and length of detected regions are given in the last two columns.

Sequencing library	Species	Mapping percentage	Number of regions	Length of regions (min-avg-max)
	Heinz	99.63	-	-
S. lycopersicum	LA716	98.43	1,754	(101-461-7515)
LA2706	LA480	96.55	2,286	(101-417-6069)
	LYC4	95.42	1,273	(101-370-10586)
	Heinz	99.04	-	-
S. pimpinellifolium	LA716	98.24	11,073	(101-533-7427)
LA1584	LA480	96.51	10,452	(101-462-6081)
	LYC4	95.44	8,072	(101-446-6096)
	Heinz	96.09	-	-
S. pennellii	LA716	95.55	6,550	(101-503-7912)
LA716	LA480	99.20	47,042	(101-615-10505)
	LYC4	96.48	26,688	(101-538-7690)

5.2.4 Use case 2: Abundance estimation and binning of metagenomics data

In metagenomics studies the goal is often to identify the constituent organisms at a specific taxonomic level (e.g. by binning) and estimate their abundances. Numerous pipelines are available, usually based on targeted sequencing of the 16S ribosomal gene or on whole metagenome shotgun (WMGS) sequencing [16]. In the latter case, mapping to a set of reference genomes is an extremely computationally intensive step. PanTools provides a competitive mapping mode to support such analyses. To demonstrate its use, we competitively mapped a metagenomics stool sample (SRS011061) from HMRARG2 [17] on the pan-genome of the reference genome database of the Human Microbiome Project (HMP) [18] (see Methods). A list of all strains and their estimated abundances is available in Additional file 1: Experiment 5. We found a strong correlation between our estimated abundances and those found in the HMSCP report [19] (Supplementary Figure 5.S1). Two bacterial strains, *Parabacteroides merdae* (ATCC 43184) and *Bacteroides cellulosilyticus* (DSM 14838), were the most abundant strains in this sample.

We also evaluated the accuracy of abundance estimates of PanTools on three benchmark data sets provided by the CAMI (Critical Assessment of Metagenome Interpretation) initiative [20], of low, medium and high complexity and compared it to those of two tools specifically developed for this problem, Kallisto [21] and DiTASiC [22]. We ran PanTools in two *random-best* competitive modes; in the first run, we uniformly distributed shared reads between genomes, where in the second we considered the coverage of uniquely mapped reads in the first run to calculate the probabilities by which shared reads are assigned to the genomes. The idea behind this approach was that unique reads come from the strain-specific regions of the genomes and their abundance reflects the relative abundance of the genome in the sample. This approach significantly improved the accuracy of PanTools on the medium complexity CAMI data set, where there was a large imbalance between the abundance of some extremely similar strains.

Table 5.4 shows the accuracy of the abundance estimates of Kallisto, DiTASiC and PanTools (see Supplementary Figure 5.2) in terms of the root mean squared error and correlation coefficient between estimates and the ground truth. In this experiment, Kallisto performed best in terms of speed and accuracy. The runtime of DiTASiC grows quadratically with the number of genomes, as it needs to calculate the pairwise similarity between the genomes. On the high-complexity dataset of 1074 genomes we killed the process after 10 days. In contrast, PanTools was able to handle all the three datasets in reasonable time with accuracy comparable to Kallisto, while additionally simultaneously binning reads in individual SAM files which could in principle directly be passed on to an assembler to build the contigs.

Table 5.4. PanTools is as accurate as Kallisto and DiTASiC in abundance estimation of metagenomic samples with different levels of complexity. Root mean square error (RMSE), correlation coefficient between estimates and ground truth and runtime of three methods have been presented on the three CAMI benchmark datasets.

Dataset	Tool	RMSE	Correlation coefficient	Runtime (seconds)
Low	Kallisto	121,072	**0.999**	361
	DiTASiC	205,980	0.998	2,820
	PanTools	367,277	0.994	2,263
	PanTools (coverage-based)	148,138	**0.999**	4,619
Medium	Kallisto	141,309	0.996	542
	DiTASiC	121,824	**0.997**	52,200
	PanTools	707,460	0.881	2,218
	PanTools (coverage-based)	375,896	0.969	4,479
High	Kallisto	35,960	**0.980**	1,321
	DiTASiC	-	-	>10days
	PanTools	42,404	0.972	5,585
	PanTools (coverage-based)	44,866	0.969	11,112

5.3 Discussion

Multi-genome read mapping is necessary to overcome the “reference bias” that comes from only considering reads that map to a single reference. Unmapped reads are typically not considered for downstream analyses, while these could point to interesting variants. As we have demonstrated, unmapped reads in a sample can originate from genomic regions absent in or highly different from the reference. Ideally, the known variation between different genomes is exploited to improve read mapping across these regions. Existing variation-aware read mappers, such as Graphtyper [5], enrich a reference genome with known variants to improve read mapping across highly variable regions and capture polymorphisms, which are finally called with respect to the reference. This approach works well for specific genomic regions, e.g. HLA genes in human, where many variants are already known [23].

Still, reads from regions not present in such enriched references will remain unmapped. In studies on species with highly dynamic genomes, e.g. crops, where gene content varies and co-linearity is typically not preserved, a multi-genome read mapping approach is therefore preferable. PanTools is not variation-aware in the sense that information from genome A is used to map a read to genome B, but it efficiently maps reads to all genomes in a (potentially large) set. A set of reads may be mapped to a region in genome B, while they do not map on the homologous region in genome A because of SNPs and indels (as shown in Figure 5.4). By detecting homologous regions between genomes, it is, in principle, possible to project reads from one genome to the corresponding region in another genome, resembling the results of variation-aware methods.

PanTools can simultaneously generate alignment files (SAM/BAM) for multiple genomes. These can be fed to any variant caller to detect variants with respect to all the constituent genomes. Variants not captured in one reference thus may be found with respect to one or more of the other genomes. PanTools scales well to thousands of complete bacterial or fungal genomes and to collections of large genomes, such as those of plants. However, interacting with extremely large databases (e.g. tens of plant genomes) is time-consuming, as the database cannot be fully buffered in memory. A high repeat content of genomes also increases the runtime, as it causes certain nodes to have many genomic locations. In our experiments with the pan-genome of four tomato accessions, we overcame this limitation by ignoring low-complexity nodes when collecting candidate hits.

Our current method is designed to map genomic short reads, single or paired-end. Soft clipping has been implemented, but split alignments are not reported. We intend to develop this further in the future, as it is required for the detection of (some forms of) structural variation. Along the same lines, we will investigate spliced mapping of transcriptome data, considering multiple partial hits per read. Mapping long reads, e.g. PacBio or Oxford Nanopore, would be another useful extension, but this requires additional work, for example to implement an additional k-mer index with smaller k to handle higher rates of error and an alternative (banded) alignment approach which would scale to longer sequences.

5.4 Conclusions

The number of sequenced species is increasing rapidly and chromosome-scale, haplotype-resolved genomes are now within reach for many of these. This necessitates a transition from linear, single-reference to pan-genome approaches in genomics. Graphs can represent such pan-genomes, large numbers of related sequences, in a compact fashion. PanTools offers a practical pan-genome sequence representation, indexed and stored in a graph database, annotated with structural and functional information.

In this work we have extended PanTools with read-mapping functionality. The method generates accurate alignments to all (or a subset) of the constituent genomes at once. Simultaneous mapping of reads allows avoiding redundant computations and can optionally distribute reads over genomes in a competitive manner, required in applications such as metagenomics. PanTools thus offers a solid basis, which can and will be further extended to integrate and mine different types of -omics data, paving the way towards comparative pan-genomics.

5.5 Methods

Before we present the read-mapping algorithm, first we briefly explain the pan-genomic data structure to which the reads will be mapped. Then, we discuss our approach to competitive read mapping and finally introduce the data and methods used in the experiments.

5.5.1 Structure of the pan-genome

PanTools condenses multiple genomes in a generalized De Bruijn graph (gDBG), stored with structural annotations and proteomes in a graph database [24]. There are two additional databases, an index database and a genome database, which facilitate efficient graph indexing and sequence retrieval respectively. A memory-mapped implementation of graph, index and genome databases minimizes the required I/O operations even in random access scenarios.

The gDBG captures the similarity and divergence of genomes at the resolution of *k*-mers. It is a compressed, bi-directed DBG, i.e. there is no non-branching path in the graph and every node represents a piece of double-stranded DNA of minimum length *k*, which occurs only once in the graph. Each sequence (contig, scaffold or chromosome) can be traversed as a continuous path in the graph in either forward or reverse direction. The positions of the node in the constituent sequences are stored on the edges of the graph. Figure 5.5 illustrates a node of this graph, a piece of DNA occurring in sequence 1 at position 8, sequence 2 at position 12, both in forward direction (TAC); and in sequence 3 at position 4, in reverse direction (GTA). During the gDBG construction we build a *k*-mer index that maps canonical *k*-mers to a unique graph coordinate: a triple of the identifier of the node, the zero-based offset of the *k*-mer in the node and the direction of the *k*-mer. For example, the 2-mer AC is mapped to coordinate (56, 1, F) if it occurs in node 56 at offset 1 in forward direction.

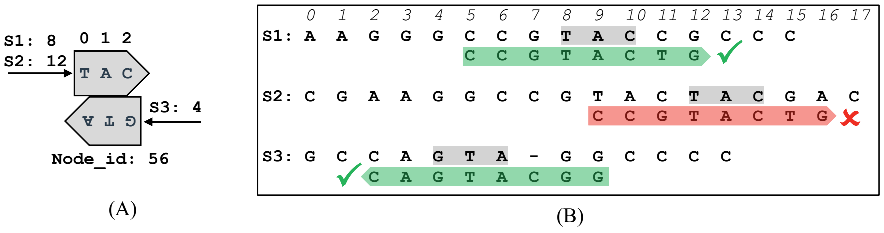


Figure 5.5. Pan-genomic read mapping. **A.** Structure of a node of generalized DBG; this piece of DNA occurs in three sequences S1, S2 and S3, respectfully, at positions 8, 12 and 4. **B.** After retrieving the candidate hits in each sequence and performing exact alignments, the read is mapped to S1 and S3 but not to S2.

5.5.2 Read mapping

Given a genomic read, α (by default 15) equidistant k-mers are sampled from it and looked up in the k-mer index to retrieve the graph coordinates where the reads should map. The k-mer size used for read mapping is the same as the k-mer size used in graph construction, since the existing k-mer index is exploited. The number of k-mer samples α should be chosen higher when the error/mutation rate is high. The collected graph coordinates can then be translated into the start position of potential hits (the *candidate hits*) in the constituent sequences. For example, again consider 2-mer AC from read CCGTACTG. The position of AC in sequence 1 is the position of node 56 in this sequence (8) plus the forward offset of AC (1) in the node, i.e. 8 + 1 = 9. The offset of AC in the read is 4, so the position of the candidate hit on sequence 1 is 9 – 4 = 5. Candidate hits could be supported by different number of k-mer samples. Candidate hits are therefore sorted by the number of supporting k-mers (in decreasing order) and local alignments are only calculated for the first ω (by default 15) hits in this ordered list. Local alignment is performed using a Smith-Waterman algorithm with a banded matrix to reduce the number of calculations by limiting the number of gaps (by default 5) allowed in the alignment. As PanTools was developed for short reads, where a limited number of insertions/deletions is expected, this seems reasonable. If the alignment identity, defined as the number of identical positions divided by the length of alignment, is higher than a threshold π (by default 0.5), the hit will be reported as a *proper hit* (Figure 5.1B).

Algorithm 1 shows the pseudocode of our read mapping approach. Ideally, all k-mers, sampled from a read, point to the same position in a sequence. However, in the presence of sequencing errors, polymorphisms and genomic duplications some k-mers may not be found or may be found at multiple locations. Hence, these locations are clustered, collected in $Pos[S]$, based on their proximity and the cluster size is considered as a score for that candidate hit. As many candidate hits may be false positives with low scores, only the ω most high-scoring hits are considered (Line 7). If all of these hits are supported only by one k-mer, potentially a low-complexity one, we just consider the first hit for the alignment. For each sequence, all proper hits, whose alignment identity is greater than a minimum threshold, are collected (Lines 8-9). Users have the option of reporting all the highest-scored hits (all-best), a random one if there are multiple best choices (random-best), or all the collected hits (all).

Algorithm 5.1. Pseudocode of read-mapping algorithm of PanTools.

Input: the pan-genome, reads in FASTQ format
Output: alignments in SAM or BAM format

```
1   for each read $R$
2       for $\alpha$ equidistant $k$-mers $K$ of $R$
3           find node $N$ containing $K$
4           for each sequence $S$ passing through node $N$
5               $Pos[S]$ collects candidate hits of $R$ in $S$
6       for each sequence $S$
7           for the first $\omega$ members in sorted list of candidate hits $H$ in $Pos[S]$
8               if identity of $H$ aligned to $R$ is greater than a threshold $\pi$
9                   put $H$ in $Hits[R,S]$
10          Report $Hits[R,S]$ in random-best, all-best or all mode
```

5.5.3 Competitive mapping

PanTools is able to map reads in competitive mode, which is required for some applications, e.g. metagenomics and contamination screening. In this mode, proper hits to all the constituent genomes are collected and only those with the highest alignment identities (best hits) are considered. If there is a single best hit it is reported, otherwise PanTools offers three options for reporting the multiple best hits. First, *none-best* does not report any ambiguous hit; second, *random-best* selects one of the best hits randomly, either uniformly or based on some probabilities given to each genome; third, *all-best* which reports all the best hits. The random-best option works best for abundance estimation in metagenomics. When read mapping is followed by reference-guided assembly of the generated SAM files, it is preferable to use the all-best option to increase the horizontal coverage of the genomes.

5.5.4 Data and experimental setup

All experiments were executed on an Ubuntu 14.04 server, Intel® Xeon® X5660@2.8GHz, with 6GB RAM and 16 processing cores. To generate synthetic reads, a 1% mutation rate was applied to the reference genomes of two model species *E. coli* (str. K-12 substr. MG1655) and *S. cerevisiae* (S288C, assembly R64) and 10x HiSeq 2500 (2×100) and MiSeq v3 (2×250) reads were simulated from the mutated genomes using the ART Illumina simulator [26]. To show the accuracy and efficiency of PanTools, it was compared to four single-reference methods: Stampy, Bowtie2 and BWA-MEM and NextGenMap. The known genomic origin of the simulated reads allowed to compare the accuracy of the methods by counting the number of properly mapped (TP), wrongly mapped (FP) and unmapped (FN) reads, calculating the sensitivity = TP/(TP+FN) and specificity = TP/(TP+FP) of the tools which were then combined to an F-score as the ultimate measure of accuracy: F-score = 2×sensitivity×specificity/(sensitivity+specificity).

5.5.5 PanTools dependencies and parameters

For read mapping, PanTools depends only on KMC [25] for construction of the graph. The parameters affecting its mapping behavior are listed in Table 5.5.

Table 5.5. PanTools read mapping algorithm comes with several parameters which can be adjusted to trade-off between accuracy and speed.

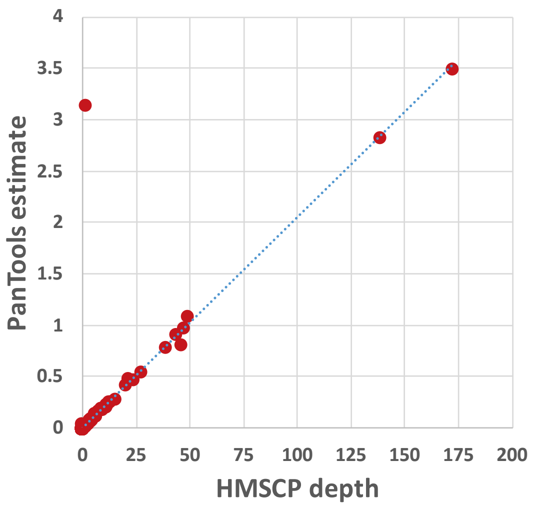
Parameter	Range	Default
Number of parallel working threads	[1 .. cores]	1
Minimum acceptable identity of the alignment (π)	[0 .. 1)	0.5
Number of *k*-mers sampled from the read (α)	$(0 .. r - k + 1]$	15
Minimum acceptable length of alignment after soft-clipping	[10 .. 100]	13
Maximum acceptable length of alignment	[50 .. 5000]	1000
Maximum acceptable length of fragment	[50 .. 5000]	2000
Maximum number of candidate hits to examine (ω)	[1 .. 100]	15
Length of band in banded alignment	[1..100]	5
Stringency of soft-clipping	[0..3]	1
Alignment mode (Negatives for competitive, see manual)	[-3..3]	2

To demonstrate the scalability of PanTools compared to the other tools, four sets of fungal genomes (pan-genomes) were considered at different taxonomic levels. The first pan-genome consisted of ten copies of the reference genome (R64) of *Saccharomyces cerevisiae* S288c*.* The second one contained ten different strains of *Saccharomyces cerevisiae* (including the reference R64). The third pan-genome included genomes from ten different species in the *Saccharomyces* genus. Finally, the fourth pan-genome contained genomes from ten different fungal genera in the family of Saccharomycetaceae (see Additional file 1: Experiment 2). For this experiment, the simulated MiSeq library of *S. cerevisiae* was used.

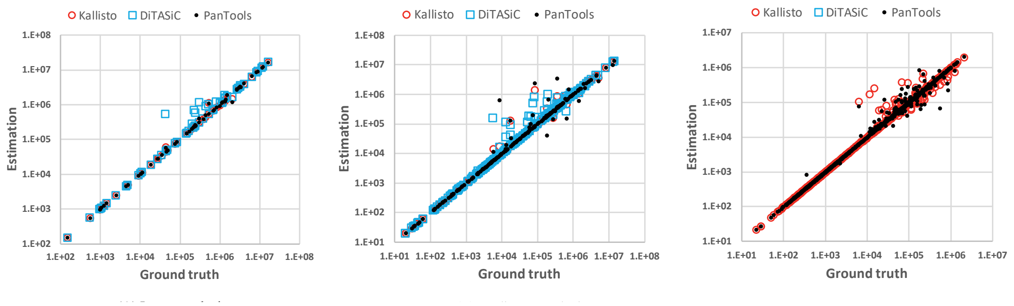
We demonstrated two real use cases on plant pan-genomes. First, a recent llumina HiSeq 2500 paired-end sequencing archive (ERR2721960) of ~13.3 million paired-end reads from DJA-1 accession was mapped to the reference and 18 additional high-quality assemblies of *Arabidopsis thaliana*. Second, three large paired-end sequencing libraries from the 150 Tomato Genome Resequencing Project [3] were mapped to the reference genome of *Solanum lycopersicum* (Heinz 1706) [27] and three additional accessions, *Solanum pennellii* (LA716) [28], *Solanum pimpinellifolium* (LA0480) [29], and *Solanum habrochaites* (LYC4) [3].

To test PanTools' competitive mode of read mapping, a large library of 89.6 million paired-end reads from a stool sample was mapped (competitive random-best mode with uniform distribution) on a large pan-genome of the reference genome database of the

Human Microbiome Project. This database comprised of 130 archaeal strains over 97 species, 326 lower eukaryotes over 326 species, 3683 viral strains over 1420 species, and 1733 bacterial strains over 1253 species. The construction of this pan-genome took 17 CPU hours, resulting in a database of size 104 GB, and read mapping was performed in 4.3 CPU hours. Additionally, we constructed a pan-genome of the reference genomes from the “Critical Assessment of Metagenome Interpretation” (CAMI) benchmark, and compared our abundance estimates to those achieved by Kallisto quantification and DiTASiC on the three provided metagenomics datasets of low, medium and high complexity. Kallisto is based on a fast pseudo-alignment followed by an expectation–maximization (EM) approach to resolve the read abundance ambiguities. DiTASiC takes the raw pseudo-alignments of Kallisto, calculates the pairwise similarity of genomes and fits a generalized linear model (GLM) to resolve the read assignment ambiguities.



Supplementary figure 5.S1. There is a strong correlation between abundances estimated by Human Microbiome project (HMSCP report) and PanTools.



Supplementary figure 5.S2. There is a strong correlation between abundances estimated by Kallisto, DiTASiC and PanTools versus the ground truth provided in the three CAMI benchmark data sets.

References

1. Reinert K, Langmead B, Weese D, Evers DJ. Alignment of next-generation sequencing reads. Annu Rev Genomics *Hum Genet*. 2015;16(1):133–51.
2. Gibbs RA, Boerwinkle E, Doddapaneni H, Han Y, Korchina V, Kovar C, *et al*. A global reference for human genetic variation. *Nature*. 2015;526(7571):68–74.
3. Aflitos S, Schijlen E, De Jong H, De Ridder D, Smit S, Finkers R, *et al*. Exploring genetic variation in the tomato (*Solanum* section *Lycopersicon*) clade by whole-genome sequencing. *Plant J*. 2014;80(1):136–48.
4. Huang L, Popic V, Batzoglou S. Short read alignment with populations of genomes. *Bioinformatics*. 2013;29(13):i361–70.
5. Eggertsson HP, Jonsson H, Kristmundsdottir S, Hjartarson E, Kehr B, Masson G, *et al*. Graphtyper enables population-scale genotyping using pan-genome graphs. *Nat Genet*. 2017;49(11):1654–60.
6. Garrison E, Sirén J, Novak AM, Hickey G, Eizenga JM, Dawson ET, *et al*. Variation graph toolkit improves read mapping by representing genetic variation in the reference. *Nat Biotechnol*. 2018;36(9):875–9.
7. Schneeberger K, Hagmann J, Ossowski S, Warthmann N, Gesing S, Kohlbacher O, *et al*. Simultaneous alignment of short reads against multiple genomes. *Genome Biol*. 2009;10(9):R98.
8. Sirén J, Välimäki N, Mäkinen V. Indexing finite language representation of population genotypes. LNCS. 2011. p. 270–81.
9. Valenzuela D, Norri T, Välimäki N, Pitkänen E, Mäkinen V. Towards pan-genome read alignment to improve variation calling. *BMC Genomics*. 2018 May 9;19(S2):87.
10. Sheikhizadeh S, Schranz ME, Akdel M, de Ridder D, Smit S. PanTools: representation, storage and exploration of pan-genomic data. *Bioinformatics*. 2016;32(17):i487–93.
11. Li H, Durbin R. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*. 2009;25(14):1754–60.
12. Langmead B, Salzberg SL. Fast gapped-read alignment with Bowtie 2. Nat Methods. 2012;9(4):357–9.
13. Lunter G, Goodson M. Stampy: A statistical algorithm for sensitive and fast mapping of Illumina sequence reads. *Genome Res*. 2011;21(6):936–9.
14. Sedlazeck FJ, Rescheneder P, Von Haeseler A. NextGenMap: fast and accurate read mapping in highly polymorphic genomes. *Bioinformatics*. 2013;29(21):2790–1.
15. Gan X, Stegle O, Behr J, Steffen JG, Drewe P, Hildebrand KL, *et al*. Multiple reference genomes and transcriptomes for Arabidopsis thaliana. *Nature*. 2011;477(7365):419–23.
16. Breitwieser FP, Lu J, Salzberg SL. A review of methods and databases for metagenomic classification and assembly. *Brief Bioinform*. 2019;20(4):1125–36.
17. NIH Human Microbiome Project - HMRARG2. https://www.hmpdacc.org/hmrarg2. 2019;Feb 8.
18. Lloyd-Price J, Mahurkar A, Rahnavard G, Crabtree J, Orvis J, Hall AB, *et al*. Strains, functions and dynamics in the expanded Human Microbiome Project. *Nature*. 2017;550(7674):61–6.
19. NIH Human Microbiome Project - HMSCP. https://www.hmpdacc.org/hmrarg2. 2019;Feb 8.
20. Sczyrba A, Hofmann P, Belmann P, Koslicki D, Janssen S, Dröge J, *et al*. Critical assessment of metagenome interpretation - a benchmark of metagenomics software. *Nat Methods*. 2017;14(11):1063–71.
21. Bray NL, Pimentel H, Melsted P, Pachter L. Near-optimal probabilistic RNA-seq quantification. *Nat Biotechnol*. 2016;34(5):525–7.
22. Fischer M, Strauch B, Renard BY. Abundance estimation and differential testing on strain level in metagenomics data. *Bioinformatics*. 2017;33(14):i124–32.
23. Dilthey AT, Gourraud P-A, Mentzer AJ, Cereb N, Iqbal Z, McVean G. High-accuracy HLA type inference from whole-genome sequencing data using population reference graphs. Franke A, editor. *PLOS Comput Biol*. 2016;12(10):e1005151.
24. Storm CE V, Sonnhammer ELL. Automated ortholog inference from phylogenetic trees and calculation of orthology reliability. *Bioinformatics*. 2002;18(1):92–9.
25. Kokot M, Długosz M, Deorowicz S. KMC 3: counting and manipulating k-mer statistics. *Bioinformatics*. 2017;33(17):2759–61.
26. Huang W, Li L, Myers JR, Marth GT. ART: A next-generation sequencing read simulator. *Bioinformatics*. 2012;28(4):593–4.
27. The tomato genome sequence provides insights into fleshy fruit evolution. *Nature*. 2012;485(7400):635–41.
28. Bolger A, Scossa F, Bolger ME, Lanz C, Maumus F, Tohge T, *et al*. The genome of the stress-tolerant wild tomato species *Solanum pennellii*. *Nat Genet*. 2014;46(9):1034–8.

29. Razali R, Bougouffa S, Morton MJL, Lightfoot DJ, Alam I, Essack M, *et al*. The genome sequence of the wild tomato *Solanum pimpinellifolium* provides insights into salinity tolerance. *Front Plant Sci*. 2018;9:1402.

Chapter 6

Discussion

In this thesis we introduced PanTools, our approach to pan-genomics, and presented some key functionalities to make it useful in practical comparative analysis. Here, we put forward ideas on the future challenges and opportunities in using a pan-genome as a reference.

6.1 The emerging pan-genome paradigm shift

Since the publication of the first assembled genome in 1995 [1], reference genomes have been effectively used for re-sequencing and genotyping of new individuals of the same species. Advances in NGS technologies have popularized reference-based re-sequencing for comparative and functional genomics. However, taking a single genome or a consensus genome as a reference is insufficient in many domains and for many applications, such as human genomics and agriculture [2,3]. To capture the wider genomic landscape of species, references need to integrate the genomic content of multiple individuals into a pan-genome. Pan-genomes include variants and genomic regions missed in the reference genomes, and also can represent the structural variability in populations of a species. We are entering the pan-genomic era, replacing linear references with pan-genomes [4].

Switching to pan-genome references is a large paradigm shift, which demands substantial adjustments and redevelopments of existing reference-based methods and applications. At the core of pan-genomic redevelopment lies a redefinition of an efficient coordinate system on which pan-genomic applications can be reliably based. Reference-based variant calling approaches pile up reads on a reference genome and call variants with respect to that reference as a simple linear coordinate system. With pan-genomics, variants should be called with respect to the pan-genome structure. For example, in a graph-based pan-genome representation coordinates are defined as pairs for the node identifier and an offset pointing to a position in that node. Pan-genome coordinates need to be efficiently translated into genomic coordinates in order to map variants back to individual genomes. The reverse conversion (from genomic coordinates to pan-genome coordinates) is also required to investigate where a specific locus in one genome occurs in other genomes.

The structural annotation of genomes is another core functionality that needs to be redefined for pan-genomic platforms. Traditionally, assembled genomes are independently annotated through time-consuming *ab initio* and/or evidence-based gene model predictions. These annotation pipelines introduce serious bottlenecks to high-throughput genome projects if large numbers of related genomes are sequenced and assembled in a short period of time. At the same time, they ignore the fact that related individuals share the majority of their genes and will have highly similar gene models. By taking into account RNA-Seq evidence and the phylogenetic relationships of samples, joint gene-structure models have reduced annotation errors and incompatible predictions at close to medium evolutionary distances [5]. Using similar joint gene models, collective annotation of many genomes can be achieved through mapping transcriptome reads from different individuals to a pan-genome. As we have shown in Chapter 5, such a read mapping approach scales sub-linearly with the number of genomes and thus reduces the computational cost of genome annotation.

6.2 Future advances for pan-genome representation

Graph-based data structures have been frequently used to represent pan-genomes (e.g. Chapter 1). However, the variability of genomes at the sequence and structural level can increase the complexity of these graphs [6]. For example, a compressed DBG is not suitable for representation of highly variable genomes, as every single mismatch between two genomes adds three nodes to the graph (i.e. a simple bubble). As a result, the compressed DBG quickly approaches the uncompressed version of the graph as variability and the number of genomes increase. Alignment-based structures, such as the Enredo graph [7] or the data structure introduced in PanCake [8], are more efficient choices for highly variable genomes as they tolerate simple variants in nodes. However, calculating whole-genome alignments of large number of genomes using such alternative structures is not trivial. Similarly, structurally variable genomes such as those of plants, with many duplications, inversions and translocations, induce (nested) cycles in the graph. Acyclic graph representations can avoid such complex sub-structures at the cost of introducing redundant nodes, however finding the optimal assignment of duplicated or translocated segments is a non-trivial task [9]. Thus, it will be very hard to develop a one-size-fits-all pan-genome data structure.

Pan-genome representations try to condense a large number of linear genomes by storing similar nucleotide sequences only once. In a (compressed) DBG, only sequences with 100% identity are condensed into one node. Far higher rates of compression can be achieved by representations which allow for lower identity thresholds between aligned sub-sequences. However, setting such a similarity (or identity) threshold is not straightforward, since it is strongly dependent on the species analyzed, variability of the genomes and conservation of the aligned regions. In highly repetitive genomes, such as those of plants that have undergone rounds of segmental and whole genome duplications, this choice is even more critical as a low similarity threshold can lead to collapsing ancient and recent duplications making the representation highly noisy. At large evolutionary distances, nucleotide sequences are more diverged and choosing a high similarity threshold disconnects the sequences, making the representation less informative. This suggests that measures of similarity have a major impact on downstream analyses and should be adopted carefully, considering variability and evolutionary distance of species.

There are many visualization tools to explore bacterial pan-genomes with useful features for orthologous clustering, pan-gene profiling, and functional classification of genes [10-13]. However, visualization of eukaryotic pan-genomes has hardly been explored, due to the challenges imposed by size and complexity of such genomes. Novel approaches are required to visualize and represent whole-genome differences among large set of genomes from chromosome-level to nucleotide-level. To provide genomic scaling or zooming to different levels of representation, various filtering and hierarchical aggregations might be needed. To date, such approaches for scaling have not yet been investigated or proposed. Another challenge is developing interactive visualizations of pan-genomes with, fast response times. This would require ultra-fast data retrieval and rendering. In a pan-genome viewer, users should be able to switch between genomes as the reference, select subset of genomes and filter for common or unique variants.

6.3 Solving the scalability problem

Scalability has been, and will remain, a challenge in pan-genomics. The very first pan-genome (published in 2005) was mainly a bag of genes found in 6 bacterial strains [14]; a decade later, pan-genomes managed to condense 62 complete *E. coli* genomes [15], and one year later 7 complete human genomes were successfully represented as a pan-genome [16]. In the last decade, scaling to both larger genomes and larger numbers of genomes has been the main concern of pan-genomics. However, advances have been limited to the completeness of pan-genomes, but unfortunately not to their functionality and applicability. In our vision, a scalable pan-genomic solution should be able to address issues of completeness, efficiency and applicability, at the same time. To date, there is no scalable solution capable of addressing all these aspects in a single platform [17].

It is not straightforward to call a pan-genome complete, but ideally, a complete pan-genome represents the entire genomic diversity of a cohort of species and/or samples of interest. Bacterial pan-genomes are, traditionally, called closed when the number of new genes introduced by new genomes approaches zero [18]. This definition has also been used for some crop plant pan-genomes such as maize [19], wheat [20] and rice [21]. However, a closed pan-genome is not essentially complete, as new individuals can introduce novel variants (alleles in eukaryotes) contributing to the diversity of a population. In a practical setup, a pan-genome can be considered complete when it includes as much existing genomic content as possible and is generated in the course of a research project.

6.4 Opportunities and future directions

Future pan-genome approaches will be highly influenced by advances in sequencing and assembly technologies. Long-read sequencing technologies are very promising to overcome the limitations of current short read technologies, facilitating the resolution of large structural variants (SVs), repetitive regions, and haplotypes. PacBio technology is able to achieve read lengths over 10kbp-long stretches of DNA with uniform coverage [22], recently also with lower error rates [23]. Oxford Nanopore Technologies (ONT) devices generate reads even one or two orders of magnitude longer [24], but still with high base-calling error rates. Combined with high quality Illumina short reads and data from scaffolding technologies such as Bionano Genomics optical maps and Hi-C proximity ligation, long-read sequencing is very promising to deliver high-quality haplotype-separated chromosome-level genome assemblies [25,26].

Structural variation (SV) drives many important traits such as genetic diseases in humans [27] and grain size in rice [28]. SVs are very hard to detect using NGS short reads as they are usually large and enriched in repeat regions [29]. SV detection has recently been significantly boosted using long reads [30]. As accuracy and continuity of genome assemblies will be increasingly improved, traditional alignment-based SV detection will be replaced by pan-genomic SV detection [31]. In a pan-genomic approach, maximal collinear blocks among large sets of genomes can be captured, first, then SVs are detected by mining specific substructures that signify structural differences between genomes.

Simple variants are also detected differently in pan-genomes. First, variants are called inside collinear blocks during the construction of a pan-genome, then, reads of newly sequenced individuals are mapped against those blocks to detect variants. As the vast majority of variants observed in a species and/or population are supposedly available in the pan-genome, the recurring variants can be quickly detected and genotyped at low-coverage and thus significantly reducing the cost of re-sequencing [32]. An alternative approach is to assemble and align new genomes to the pan-genome, which depending on the pan-genome representation demands novel indexing and alignment approaches such as partial order alignment [33].

Besides the genomic content, pan-genomes need to be able to integrate other heterogeneous biological data to expand the general applicability of pan-genomes. For example, for plant breeding new entities such as traits and quantitative trait loci (QTL) should be defined and linked to the genomes to be able to identify the casual variation of agronomical traits of a crop by comparing mapped regions between genotypes with contrasting phenotypes [34]. By integrating genotype and phenotype data for a large number of individuals, pan-genomes would also facilitate genome-wide association studies (GWAS). Graph databases are very powerful for representation and mining of various types of biological data. In this thesis, we integrated genomes, structural features and proteomes of large number of species in a Neo4j graph database [35]. Graph databases have demonstrated to significantly outperform relational databases on querying complex biological networks with protein-protein interaction, drug-target, and gene-disease relationships [36].

6.5 Concluding remarks

At the time this PhD project started in May 2015, state-of-the-art pan-genome methods were able to represent only tens of whole bacterial genomes without any possibility to be utilized in real practice. In this thesis, we laid the foundation for practical pan-genomics specifically for large and complex genomes, opening up the way for crop pan-genomics. PanTools is among the first pan-genomic platforms able to offer some useful key functionalities for comparative studies. The design and engineering introduced in this thesis contributes ample novelty to the field which can be reused and further developed in future pan-genomic platforms. There are extensive unexplored areas in the field which will open up new applications and interesting bioinformatics challenges. At the time of this writing, people from various fields of biological research have come to the consensus that pan-genomes will make the future of comparative genomics. Considering the growing number of pan-genomic tools and the amount of effort on new developments, applications and improvements, there is a bright future ahead for the field of pan-genomics.

References

1. Fleischmann RD, Adams MD, White O, Clayton RA, Kirkness EF, Kerlavage AR, *et al*. Whole-genome random sequencing and assembly of *Haemophilus influenzae Rd*. *Science*. 1995;269:496–512.
2. Yang X, Lee W-P, Ye K, Lee C. One reference genome is not enough. *Genome Biol*. 2019;20(1):104.
3. Tao Y, Jordan DR, Mace ES. Crop genomics goes beyond a single reference genome. *Trends Plant Sci*. 2019;24(12):1072–4.
4. Marschall T, Marz M, Abeel T, Dijkstra L, Dutilh BE, Ghaffaari A, *et al*. Computational pan-genomics: Status, promises and challenges. *Brief Bioinform*. 2018;19(1):118–35.
5. König S, Romoth LW, Gerischer L, Stanke M. Simultaneous gene finding in multiple genomes. *Bioinformatics*. 2016;32(22):3388–95.
6. Iqbal Z, Caccamo M, Turner I, Flicek P, McVean G. *De novo* assembly and genotyping of variants using colored De Bruijn graphs. *Nat Genet*. 2012;44(2):226–32.
7. Paten B, Herrero J, Beal K, Fitzgerald S, Birney E. Enredo and Pecan: genome-wide mammalian consistency-based multiple alignment with paralogs. *Genome Res*. 2008;18(11):1814–28.
8. Ernst C, Rahmann S. PanCake: a data structure for pangenomes. *Proc Ger Conf Bioinforma*. 2013;34:35-45.
9. Kahn CL, Hristov BH, Raphael BJ. Parsimony and likelihood reconstruction of human segmental duplications. Bioinformatics. 2010;26(18):i446–52.
10. Chen X, Zhang Y, Zhang Z, Zhao Y, Sun C, Yang M, *et al*. PGAweb: a web server for bacterial pan-genome analysis. *Front Microbiol*. 2018;9(1910).
11. Ding W, Baumdicker F, Neher RA. panX: pan-genome analysis and exploration. *Nucleic Acids Res*. 2018;46(1):e5–e5.
12. Peng Y, Tang S, Wang D, Zhong H, Jia H, Cai X, *et al*. MetaPGN: a pipeline for construction and graphical visualization of annotated pangenome networks. *Gigascience*. 2018;7(11):giy121.
13. Clarke TH, Brinkac LM, Inman JM, Sutton G, Fouts DE. PanACEA: a bioinformatics tool for the exploration and visualization of bacterial pan-chromosomes. *BMC Bioinformatics*. 2018;19(1):246.
14. Tettelin H, Masignani V, Cieslewicz MJ, Donati C, Medini D, Ward NL, *et al*. Genome analysis of multiple pathogenic isolates of Streptococcus agalactiae: Implications for the microbial "pan-genome". *Proc Natl Acad Sci*. 2005;102(39):13950–5.
15. Marcus S, Lee H, Schatz MC. SplitMEM: A graphical algorithm for pan-genome analysis with suffix skips. *Bioinformatics*. 2014;30(24):3476–83.
16. Baier U, Beller T, Ohlebusch E. Graphical pan-genome analysis with compressed suffix trees and the Burrows–Wheeler transform. *Bioinformatics*. 2016;32(4):497–504.
17. Sherman RM, Salzberg SL. Pan-genomics in the human genome era. *Nat Rev Genet*. 2020;21(4):243–54.
18. Tettelin H, Riley D, Cattuto C, Medini D. Comparative genomics: the bacterial pan-genome. *Curr Opin Microbiol*. 2008;11(5):472–7.
19. Jin M, Liu H, He C, Fu J, Xiao Y, Wang Y, *et al*. Maize pan-transcriptome provides novel insights into genome complexity and quantitative trait variation. *Sci Rep*. 2016;6(1):18936.
20. Montenegro JD, Golicz AA, Bayer PE, Hurgobin B, Lee H, Chan C-KK, *et al*. The pangenome of hexaploid bread wheat. *Plant J*. 2017;90(5):1007–13.
21. Wang W, Mauleon R, Hu Z, Chebotarov D, Tai S, Wu Z, *et al*. Genomic variation in 3,010 diverse accessions of Asian cultivated rice. *Nature*. 2018;557(7703):43–9.
22. Schadt EE, Turner S, Kasarskis A. A window into third-generation sequencing. *Hum Mol Genet*. 2010;19(R2):R227–40.
23. Vollger MR, Logsdon GA, Audano PA, Sulovari A, Porubsky D, Peluso P, *et al*. Improved assembly and variant detection of a haploid human genome using single-molecule, high-fidelity long reads. *Ann Hum Genet*. 2020;84(2):125–40.
24. Jain M, Olsen HE, Paten B, Akeson M. The Oxford Nanopore MinION: delivery of nanopore sequencing to the genomics community. *Genome Biol*. 2016;17(1):239.
25. Deschamps S, Zhang Y, Llaca V, Ye L, Sanyal A, King M, *et al*. A chromosome-scale assembly of the sorghum genome using nanopore sequencing and optical mapping. *Nat Commun*. 2018;9(1):4844.
26. Jiao W-B, Accinelli GG, Hartwig B, Kiefer C, Baker D, Severing E, *et al*. Improving and correcting the contiguity of long-read genome assemblies of three plant species using optical mapping and chromosome conformation capture data. *Genome Res*. 2017;27(5):778–86.
27. Brandler WM, Antaki D, Gujral M, Noor A, Rosanio G, Chapman TR, *et al*. Frequency and complexity of *de novo* structural mutation in autism. *Am J Hum Genet*. 2016;98(4):667–79.
28. Wang Y, Xiong G, Hu J, Jiang L, Yu H, Xu J, *et al*. Copy number variation at the GL7 locus contributes to grain size diversity in rice. *Nat Genet*. 2015;47(8):944–8.

29. Alkan C, Coe BP, Eichler EE. Genome structural variation discovery and genotyping. *Nat Rev Genet*. 2011;12(5):363–76.
30. Huddleston J, Chaisson MJP, Steinberg KM, Warren W, Hoekzema K, Gordon D, *et al*. Discovery and genotyping of structural variation from long-read haploid genome sequence data. *Genome Res*. 2017;27(5):677–85.
31. Jandrasits C, Dabrowski PW, Fuchs S, Renard BY. Seq-seq-pan: building a computational pan-genome data structure on whole genome alignment. *BMC Genomics*. 2018;19(1):47.
32. Zan Y, Payen T, Lillie M, Honaker CF, Siegel PB, Carlborg Ö. Genotyping by low-coverage whole-genome sequencing in intercross pedigrees from outbred founders: a cost-efficient approach. *Genet Sel Evol*. 2019;51(1):44.
33. Lee C, Grasso C, Sharlow MF. Multiple sequence alignment using partial order graphs. *Bioinformatics*. 2002;18(3):452–64.
34. Tao Y, Zhao X, Mace E, Henry R, Jordan D. Exploring and exploiting pan-genomics for crop improvement. *Mol Plant*. 2019;12(2):156–69.
35. Van Bruggen R. Learning Neo4j. *Packt Publishing*; 2014.
36. Yoon B-H, Kim S-K, Kim S-Y. Use of graph database for the integration of heterogeneous biological data. *Genomics Inform*. 2017;15(1):19.

Summary

Comparative genomics investigates the genomic makeup of species to unravel their unique variations and evolutionary relationships. High-throughput sequencing technologies have enabled reading the DNA content of a wide variety of species at an unprecedented rate. With the ongoing advances in these technologies, many species are or will soon be represented by a large number of genomes. Such genomes can be highly similar, but their differences in sequence and structure are of interest in many applications as they usually underlie specific traits. Having a wealth of genomes for a species, the current practice of basing comparative studies on a single reference genome is neither efficient nor effective. Traditional reference-based approaches make use of only a single reference genome, ignoring the potentially novel genomic content found in other individuals. As a result, over the last decade there has been a growing interest in developing pan-genome structures capable of capturing a wide genomic landscape of species. In this thesis, we develop a pan-genomic platform based on a novel representation of genomes with some functionalities for sequence retrieval, structural annotation, homology detection and read mapping.

Chapter 1 briefly introduces molecular biology and the revolution in genome sequencing. Then we introduce evolution and some basic concepts in genomics and comparative genomics which are necessary for the readers to be able to follow the chapters of this thesis. We emphasize the shortcomings of traditional reference-based approaches in comparative genomics and introduce pan-genomics as a solution which recently has received much attention. We introduce the essentials of a pan-genomic platform from the perspective of the *Computational Pan-genomics Consortium*, and classify existing pan-genomic data structures into two general categories of variation-aware and multi-genome data structures. Finally, we discuss the de Bruijn graph including the stranded version we introduce in chapter 2.

Chapter 2 highlights the necessity of a transition from reference-centric to pan-genomic approaches. As a comprehensive representation of large number of genomes, we introduce a generalized *de Bruijn* graph. We present a novel algorithm to construct such a DBG and take advantage of the Neo4j graph database for consistent and scalable storage of the graph. We develop a toolset, called PanTools, which provides some useful functionalities e.g. for annotation, graph update and sequence retrieval. We demonstrate the performance of PanTools on large datasets of bacterial, fungal and plant genomes. We illustrate how sequence variation creates specific sub-structures in the pan-genome including an example of the variability of a famous gene, called FRIGIDA, among 19 *A. thaliana* accessions.

Chapter 3 emphasizes the need for highly efficient tools to detect homology in the ever-increasing genomic data. We present an efficient method for detecting homology across a large number of individuals at various evolutionary distances. The presented *k*-mer based approach considerably reduces the number of alignments between pairs of peptide sequences without sacrificing sensitivity. We demonstrate accuracy, scalability,

efficiency and applicability of the presented method in large proteomes of bacteria, fungi, plants and Metazoa. The detected homology groups are stored in the pan-genome graph database, and can be queried, for example, for their size, copy number and conservation rate.

Chapter 4 focuses on correcting errors in next-generation sequencing reads which can improve the performance of assembly and increase the accuracy and sensitivity of quantitative analyses such as differential expression analyses and variant calling. We develop a tool, called ACE, based on a *k*-mer trie data structure to correct for substitution errors in short read data. We show that ACE yields higher gains in terms of coverage depth, outperforming state-of-the-art competitors in the majority of cases, on both MiSeq and HiSeq Illumina data.

Chapter 5 presents a multi-genome read mapping approach which utilizes the index and pan-genome structure, introduced in Chapter 2, to map short reads to large number of genomes, simultaneously. One advantage is the efficiency as the joint index enables anchoring the reads to all the genomes at once avoiding repetitive alignments when the genomes are highly similar. Another advantage is that we can resolve the reference bias by including regions that are entirely missing in the reference but present in some other accessions. Moreover, such a multi-genome read mapper can be utilized in binning and abundance estimation of meta-genomic samples. In this chapter, we successfully apply this approach to map genomic and metagenomic reads to large collections of viral, archaeal, bacterial, fungal and plant genomes.

Chapter 6 puts forward some ideas on the future challenges and opportunities in the field of pan-genomics. We discuss the emerging shift from reference-centric to pan-genomic approaches and the necessity of substantial adjustments and redevelopments of traditional methods and applications such as genome annotation, structural variation detection and real-time pan-genome visualization. We conclude that the design and engineering introduced in this thesis contributes to the field and the growing number of similar efforts indicates a bright future ahead for comparative pan-genomics.

Acknowledgements

I would like to start this piece of writing with my upmost gratitude to my beloved wife for all the support, patience, and love she has devoted to our married life. Samin, you were the one who motivated me to take this overseas adventure, and stood by me till it is about to be finished now. Since we arrived to Wageningen in that gloomy afternoon of October 1st 2014, you have been giving me warmth, confidence and love in every single moment; you made everything easy. Over these years, you have been always ambitious and strong: you gave up your prestigious position as a faculty member to take this adventure with me, you overcame the challenge of achieving a Bioinformatics MSc here at WUR and endured the labor of bringing our sweetest present ever, Raybod, to this world; I am just proud of you.

My deepest sense of appreciation also goes to my beloved parents. Dear *baba* and *maman*, I left you for my bachelor studies when I was 18, and since then the story of my studies and separation never came to an end. That is of course a pity, but you embraced it all the time to let me grow and experience. I cannot put into the words my passion towards you, but can remark it that I have been always proud of you and feel very thankful for having you in my life. Thank you for making my childhood like a sweet dream. I also would like to express my appreciation to my dearest family members, my sisters *Sahar* and *Sorour* and my brothers *Siamak* and *Soroush*, and of course my beloved nieces and nephews. Over these years, I always missed you and all the beautiful moments we used to spend together. Thank you for looking after our parents and undertaking part of the responsibilities that I should have taken care of if I was there.

I also would like to extend my gratitude to my parents in law. Having three of four children apart pursuing their wishes has not been a favorable life style for you, but you have been so loving to accept that in favor of our preferences. Thank you for all your support and patience. Also, I would like to thank my sister in law, *Hadis* and her husband *Amir* for taking many family responsibilities in Iran. We were also so lucky to have *Mohsen*, *Pauline* and *Sara*, my brother and sisters in law here in Wageningen, for quite a long period. Knowing you are here and we are not alone in this land gave us the confidence to stay and study. Thank you for all your support and help.

I am also very grateful of our Iranian friends for their company giving us the warm feeling of being home, in the Netherlands. Thank you, *Mohsen, Leila, Nikdad, Arman, Nafiseh, Narges, Mehdi, Mohammad, Hamed, Behzad, Naser, Mahrooz, Farshid*, and other friends whose name does not appear here. I cherish our time together and hope to meet you again. My special thanks to *Ehsan* for being such a great friend and colleague. I never forget the wonderful time we had together in Wageningen. Thank you for all your support, advice and fantastic memories. I also would like to thank my best friends in Iran, *Keyvan*, *Ali*, *Pouya* and *Bijan*.

I also would like to extend my gratitude to all my colleagues in Bioinformatics group. Thank you *Harm* for teaching me *Advanced Bioinformatics*, and many other lessons during our meetings and retreats. You are very kind, knowledgeable and modest. Thank you

Marnix, *Aalt-jan* and *Justin* for your efforts and contribution to make such a flourishing group. I learned a lot from you and felt very proud of working next to you. I also never forget the wonderful time I had working with the other PhDs, PostDocs and master students, sitting in the corridor or the open space at Radix west. Thank you, *Judith, Ben, Mohammad, Rens, Sander, Sevgin, Miguel, Vittorio, Satria, Carlos, Christian, Eef, Ronald, Raul, Hernando, Victoria, Barbara, Roven, Margi*, and my dear paranymphs *Mehmet* and *Janani*. I appreciate all sweet memories we share from retreats, courses and conferences over these years. I am sure that we have made friendships for life; I hope we can catch up every once in a while.

Also, my sincere thanks to the secretaries *Maria* and *Marie-José* for all their endless help and also all people from applied Bioinformatics group for being present to share their experiences and expertise. During my PhD studies, I also met great scientists in workshops and conferences which I would like to recognize their impact in my learning process: many thanks to *Tobias Marschall, Thomas Abeel, Paul Kersey, Veli Mäkinen, Eric Rivals, Rayan Chikhi, Paul Medvedev, Sven Rahmann, Erik Garrison, Alexander Schönhuth* and *Jasmijn Baaijens*. I also would like to thank *David Judge* and *Freek Bakker* for their wonderful style of teaching during the courses that I had the chance to participate in. I also would like to express my appreciation for my current KeyGene's colleagues, in particular, managers of our group and department, *Antoine Janssen* and *Roeland van Ham*, firstly for hiring me in their team after my PhD period and, secondly for giving me sufficient free time to wrap up my thesis after being employed.

Last but not least, I would like to appreciate all kind supports from my supervisors and promotors. *Sandra*, this thesis would not have been reached to such a rewarding end without your continuous supervision and guidance. Believe or not, you were my best teacher ever with the greatest sense of responsibility. Thank you very much for all your efforts you dedicated to my training and research. *Eric*, I received precious support from you in experimental design of my research. The certainty of your guidance gave me the drive to run all the experiments you suggested. Without your guidance, PanTools would not have been demonstrated as an applicable pan-genomic platform. I also would like to thank to my external supervisor, *Jaap Heringa* for his precious input to my project. I would like to finish the acknowledgements with my uppermost appreciation to you *Dick*. You believed me as a MSc graduate from Iran, hired me and provided me the chance to grow here. When first arrived in the Netherlands, I got a great reception from you which was beyond my expectation. I cannot show my gratitude within these lines, but I will always recall that on October 1st 2014, you drove to the Schiphol airport with your own car to pick me and Samin up, took us to an apartment you had already rented, and invited us to join you , *Barbara* and your beautiful children for dinner; that just meant a lot to us. You are a great professor full of understanding. Thank you very much.

List of publications

Sheikhizadeh S, de Ridder D, Schranz ME, Smit S. Pan-genomic read mapping. *bioRxiv*, **2019**. DOI: 10.1101/813634.

Sheikhizadeh S, de Ridder D, Schranz ME, Smit S. Efficient inference of homologs in large eukaryotic pan-proteomes. *BMC Bioinformatics*. **2018**;19(1):340.

The Computational Pan-Genomics Consortium. Computational pan-genomics: status, promises and challenges. *Briefings in Bioinformatics*, **2018**; 19(1):118-35,

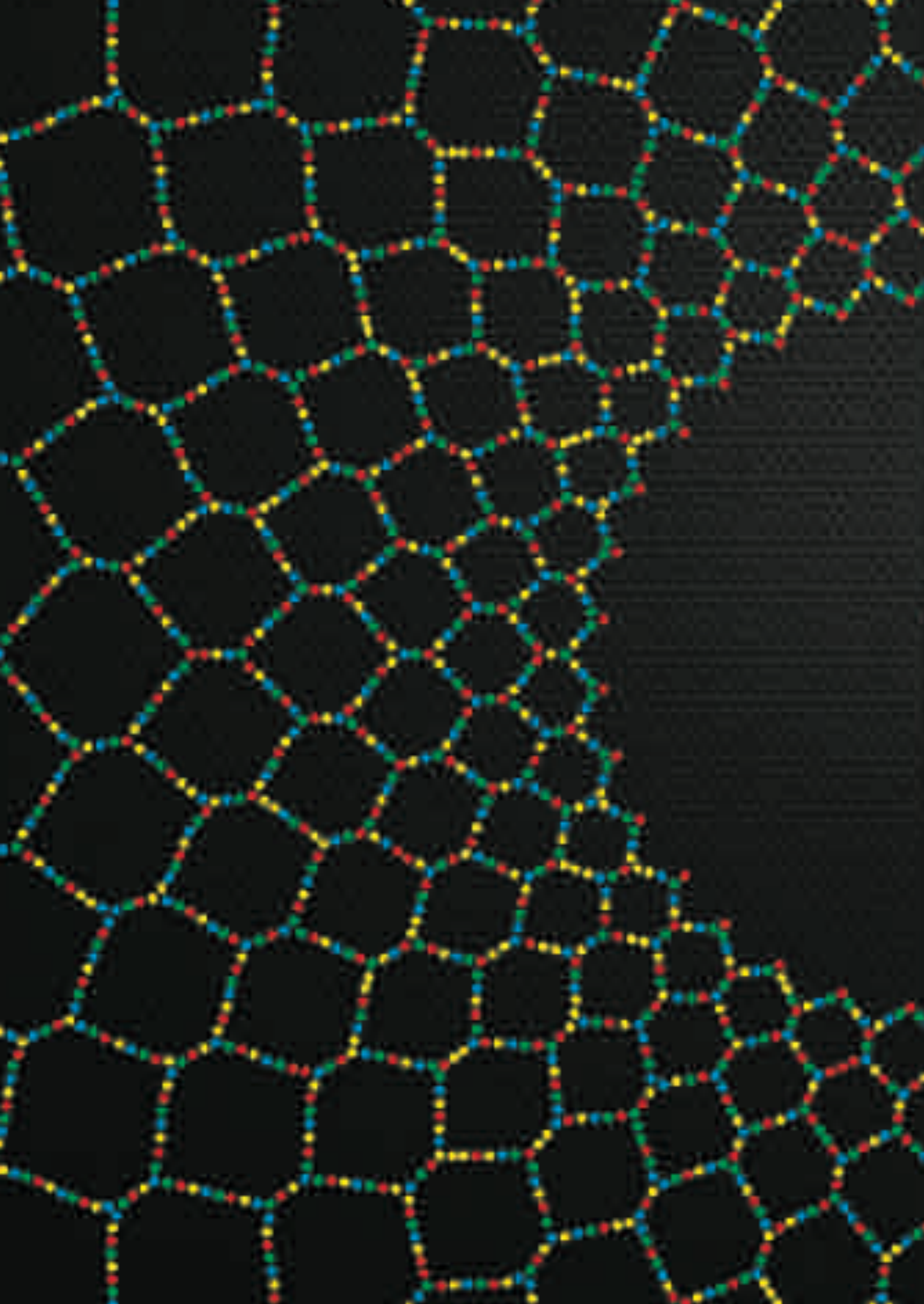
Sheikhizadeh S, Schranz ME, Akdel M, de Ridder D, Smit S. PanTools: representation, storage and exploration of pan-genomic data. *Bioinformatics*. **2016**;32(17):487–93.

Sheikhizadeh S, de Ridder D. ACE: accurate correction of errors using K-mer tries. *Bioinformatics*. **2015**;31(19): 3216–18.

Sheikhizadeh S, Hosseini S. SMOTER, a structured motif finder based on an exhaustive tree-based algorithm. *Current Bioinformatics*. **2014**;9(1):34–43.

This PhD project *Pan-genomics for crops*, was funded by the Graduate School Experimental Plant Sciences (EPS), Wageningen University & Research, in 2015.

Thesis layout: Siavash Sheikhizadeh Anari
Cover design: Samad Bahrami Kashkooli, kashkoolisamad@gmail.com
Printed by: ProefschriftMaken, www.proefschriftmaken.nl



Propositions

1. To support practical use, computational pan-genomics should focus more on including relevant annotations and less on developing novel efficient data structures.
   (this thesis)

2. *De Bruijn* graphs are not appropriate representations of highly diverged sequences as their resolution is limited by the *k*-mer size.
   (this thesis)

3. *Debugging* is not essentially the process of removing bugs, but replacing them with preferably less fatal ones (*rebugging*).

4. To prevent researchers from wasting effort on resolving inconsistencies and dependencies, FAIRness principles need to be enforced by public scientific software portals.

5. Social distancing leads to an appreciation of social networking even by those who did not appreciate this earlier.

6. *Developing* countries that do not cherish their human resources should be classified as *de-developing* countries by UN.

Prepositions belonging to the thesis, entitled

Towards comparative pan-genomics

Siavash Sheikhizadeh Anari

Wageningen, 14 July 2020