ABSTRACT

Title of Dissertation:	REFERENCE-GUIDED ASSEMBLY OF METAGENOMES
	Victoria Paz Cepeda-Espinoza, Doctor of Philosophy, 2020
Dissertation Directed By:	Professor Mihai Pop, Department of Computer Science

Microorganisms play an important role in all of the Earth's ecosystems, and are critical for the health of humans [1], plants, and animals. Most microbes are not easily cultured [2]; yet, Metagenomics, the analysis of organismal DNA sequences obtained directly from an environmental sample, enables the study of these microorganisms. Metagenomic assembly is a computational process aimed at reconstructing genes and genomes from metagenomic mixtures. The two main paradigms for this method are *de novo* assembly (i.e., reconstructing genomes directly from the read data), and *reference-guided* assembly (i.e., reconstructing genomes using closely related organisms). Because the latter paradigm has a high computational cost—due to the mapping of tens of millions of reads to thousands of full genome sequences—Metagenomic studies have primarily relied on the former paradigm.

However, the increased availability of high-throughput sequencing technologies has generated thousands of bacterial genomes, making reference-guided assembly a valuable resource regardless of its computational cost. Thus, this study describes a novel metagenome assembly approach, called MetaCompass, that combines reference-guided assembly and *de novo* assembly, and it is organized in the following stages: (i) selecting reference genomes from a database using a metagenomic taxonomy classification software that combines gene and genome comparison methods, achieving species and strain level resolution; (ii) performing reference-guided assembly in a new manner, which uses the minimum set cover principle to remove redundancy in a metagenome read mapping while performing consensus calling; and (iii) performing *de novo* assembly using the reads that have not been mapped to any reference genomes.

We show that MetaCompass improves the most common metrics used to evaluate assembly quality—contiguity, consistency, and reference-bases metrics—for both synthetic and real datasets such as the ones gathered in the Human Microbiome Project (HMP) [3], and it also facilitates the assembly of low abundance microorganisms retrieved with the reference-guided approach. Lastly, we used our HMP assembly results to characterize the relative advantages and limitations of *de novo* and reference-guided assembly approaches, thereby providing guidance on analytical strategies for characterizing the human-associated microbiota.

REFERENCE-GUIDED ASSEMBLY OF METAGENOMES

by

Victoria Paz Cepeda Espinoza

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2020

Preface

The algorithms, software, and results in this dissertation have either been published in peer-reviewed journals and conferences or are currently under preparation for submission and/or available as a preprint. At the time of this writing, parts of Chapters 1, 2 already been published and are reformatted here. Chapter 3 is under preparation for publication. Chapter 4 and 6 are available in a preprint and are under preparation for submission to a peer-reviewed journal.

- **Chapter 1: Introduction.**

  Ghurye, J. S., **Cepeda-Espinoza, V.**, & Pop, M. (2016). Focus: Microbiome: Metagenomic Assembly: Overview, Challenges and Applications. *The Yale journal of biology and medicine*, *89*(3), 353. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5045144

  My contributions to these works include (1) surveying the literature for recent and seminal results, and (2) writing the manuscript.

- **Chapter 2: Related work.**

  Olson, N. D., Treangen, T. J., Hill, C. M., **Cepeda-Espinoza, V.**, Ghurye, J., Koren, S., & Pop, M. (2017). Metagenomic assembly through the lens of validation: recent advances in assessing and improving the quality of genomes assembled from metagenomes. *Briefings in Bioinformatics*, bbx098. https://doi.org/10.1093/bib/bbx098

My contributions to these works include (1) surveying the literature for recent and seminal results, and (2) writing the manuscript.

- **Chapter 3: Selecting references genome for metagenomic reference-guided assembly.**

  Cepeda-Espinoza & Pop, M. "Reference selection of Metagenomes". Under preparation.

  My contributions to this work include (1) design and implementation of the method, (2) doing software evaluation, and (3) writing the manuscript.

- **Chapters 4 and 5:**
  - **Chapter 4: Reference-guided metagenomic assembly.**
  - **Chapter 5: Hybrid reference-guided and de novo assembly of metagenomes.**

  Cepeda-Espinoza, V., Liu, B., Almeida, M., Hill, C. M., Koren, S., Treangen, T. J., & Pop, M. "MetaCompass: Reference-guided Assembly of Metagenomes". *bioRxiv* (2017): 212506. https://www.biorxiv.org/content/early/2017/11/01/212506

  My contributions to this work include (1) design and implementation of the algorithm, (2) doing software evaluation, and (3) writing the manuscript.

Here is the list of other publications in which I have been involved, as a contributing author.

- Meisel, J. S., Nasko, D. J., Brubach, B., **Cepeda-Espinoza, V.**, Chopyk, J., Corrada-Bravo, H., ... & Shah, N. (2018). Current progress and future opportunities in applications of bioinformatics for biodefense and pathogen detection: report from the Winter Mid-Atlantic Microbiome Meet-up, College Park, MD, January 10, 2018.

My contributions to this work include (1) Participating in the Winter Mid-Atlantic Microbiome Meet-up as a speaker and discussion sessions, and (2) reviewing the manuscript before publication.

- Mihai Pop, Jacquelyn Meisel, **Victoria Cepeda Espinoza**, Kiran Javkar and Dylan Taylor. " Introducing genome assembly to the general public through interactive word games" (2020). Education COSI: Computational Biology Education. ISMB 2020.

  My contributions to this work include (1) Designing word games to simulate genome assembly and participating in the Maryland Day.

Dedication

To my family and friends for their unconditional love and support.

Acknowledgements

First, I wish to express my sincere appreciation to my supervisor, Professor Mihai Pop, who gave the opportunity to join his lab to work on challenging problems. He convincingly guided and encouraged me to be professional and do the right thing even when the road got tough. Without his persistent help, this dissertation would not have been possible.

I would also like to thank all my other committee members —Héctor Corrada-Bravo, Rob Patro, Abhinav Bhatele and Stephanie Yarwood—for being interested in my research and for giving me insightful feedback to complete my dissertation.

I wish to express my deepest gratitude to all my colleagues and collaborators, which are all current and former members of the Pop Lab: Brian Brubach, Brook Stacy, Christopher Hill, Dan Nasko, Domenick Braccia, Dylan Taylor, Jay Ghurye, Hari Muralidharan, Irina Astrovskaya, Jacquelyn Meisel, Jeremy Selengut, Joseph Paulson, Kiran Javkar, Lee Mendelowitz, Marcus Fedarko, Mathieu Almeida, Nate Olson, Nidhi Shah, Saul Sarria, Senthil Muthiah, Seth Commichaux, Sergey Koren, Todd Treangen and Tu Luan. Your academic and moral support throughout my entire PhD. Process was a blessing.

Finally, I would like to thank every person who made my graduate student life happier, including students, faculty, and staff from CBCB, UMIACS, the CS department and UMD.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

Bp	Base pair
DNA	Deoxyribonucleic acid
Gbp	1 Giga base pair equal to 10^9 base pairs
Kbp	1 Kilo base pairs equal to 1000 base pairs
Mpb	1 Mega base pairs equal to 10^6 base pairs
ORF	Open reading frame
RNA	Ribonucleic acid

Chapter 1: Introduction

1.1 DNA basics

DNA is a long chain of molecules that holds all the genetic information needed for the organisms to function and reproduce. The DNA alphabet is composed of four letters: A (Adenine), T (Thymidine), G (Guanine), and C (Cytosine), known as nucleotides. A DNA sequence can be represented as a string consisting of series of these four letters, and it is composed of two strands of nucleotides that "match" each other following a constraint: i.e., A always matches T and C always matches G. Because nucleotides pair up, each letter in the DNA sequence is called base pair (bp). Here we will only refer to the sequence of one strand.

An organism's genome sequence refers to an organism's complete set of DNA (e.g., the human genome has 3.2 billion bp). A genome consists of a set of genes—which are contiguous intervals of DNA that contain information needed to code for proteins or RNAs—that are paramount for answering a variety of biological questions, such as inheritance, ancestry, health and disease of an organism.

Table 1.1. Overview of sequencing technologies.

	Technology	**Read Length**	**Accuracy**	**Time per run**	**Bases per run**
Third-generation sequencing technologies	Single Molecular Real Time Sequencing (Pacific Biosciences)	10 kbp to 15 kbp	87% (Low)	30 min. to 4 hrs.	5-10 Gbp
	Oxford Nanopore MinION Sequencing	5 kbp to 10 kbp	70% to 90% (Low)	1 to 2 days	500 Mbp
Second-generation sequencing technologies	Ion Semiconductor (Ion Torrent sequencing)	Up to 400 bp	98% (Medium)	2 hrs.	10 Gbp
	Sequencing by synthesis (Illumina)	50-300bp	99.9% (High)	1 to 11 days	300Gbp
	Sequencing by Ligation (SOLiD sequencing)	75 bp	99.9% (High)	1 to 2 weeks	3 Gbp
	Pyrosequencing (454)	700 bp	99.9% (High)	24 hrs.	400 Mbp
First-generation sequencing technologies	Chain termination (Sanger sequencing)	400 to 900 bp	99.9% (High)	N/A	50-100 Kbp

1.2 DNA sequencing

The process of determining the complete base pair sequence order of a string of DNA sequencing is called *DNA sequencing* or simply *sequencing*. The technique called *shotgun sequencing* randomly “breaks up” DNA into a collection of small fragments, and each fragment is individually sequenced into a *read*. In many cases, reads are *pair* *ended* or *mate-paired*, which means that pairs of reads are sequenced from the same DNA fragment. The distance between the reads in each pair, and their relative orientation are approximately known.

The various sequencing technologies developed over the past 40 years can be broadly classified into three generations based on key technological innovations (Table 1.1). First-generation sequencing, usually referred to as Sanger sequencing, relied on DNA cloning, and therefore had limited throughput. The second-generation sequencing, often called short-read sequencing due to the much shorter length of the sequences compared to the Sanger technology, massively increased throughput by parallelizing many reactions on chips. Third-generation sequencing, also called single molecule or long-read sequencing, allows sequencing of single DNA molecules in contrast with prior technologies that required each molecule to be amplified [4]. Single molecule technologies can read substantially longer sequences than prior technologies.

Sequencing technologies randomly oversample the genome and produce many overlapping *reads* such that the total amount of reads is greater than the amount needed to "cover" each base pair of the genome. The theoretical or expected coverage is the average number of times that each nucleotide is expected to be sequenced given a certain number of reads of a given length and the assumption that reads are randomly distributed across an idealized genome[5]. Empirically, the term "depth of coverage" (usually referred to as *coverage*) is defined as the average number of times a base of a genome is sequenced or "covered" by a read. The term "breadth of coverage", on the other hand, is defined as the percentage of bases of a genome that are sequenced, or the percentage of the genome "covered" by reads (Figure 1.1).

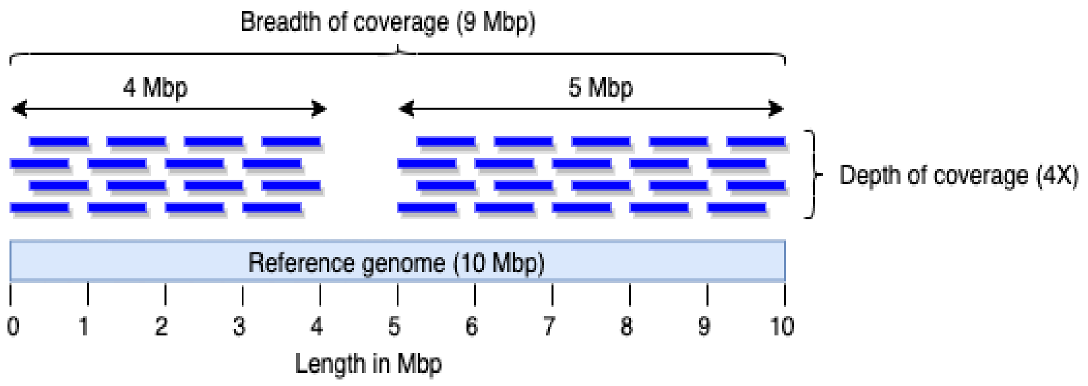


Figure 1.1. Depth and Breadth of coverage.
In this example, the depth of coverage of the reference genome (10 Mbp) is 4X (there is average of 4 reads per base pair). The breadth of coverage is 90% of the reference genomes, or, in other words, 1 Mbp of the genome is not covered by any read.

1.3 Genome assembly

Genome assembly is the reconstruction of a genome from short overlapping reads, and it is a complex computational tasks due to DNA segments repeated within a same organism, also known as "intragenomic repeats" (Figure 1.2 A) [6]. Repeats present a challenge, because different genomic regions that share repeats can be indistinguishable if the repeats are longer than the reads. Therefore, if a genome region has a repeat, the repeat will introduce several possible sequences or paths in the assembly graph. It has been shown that assembly complexity is directly tied to the ratio between the sequencing read length and the length of repeats [7].

Algorithms and computational tools called "genome assemblers" are able to reconstruct near-complete genome sequences from reads and they fall into two paradigms: the *de novo* assembly strategy where reads are used to reconstruct the genome without prior knowledge of the source of DNA, and a *reference-guided* assembly (also known as *comparative* assembly), in which reads are aligned to a reference genome. Although the first genome assembler was developed almost three decades ago and now there are numerous computational tools to tackle genome assembly, genome assembly remains a challenging computational problem. In most cases, genome assemblers cannot fully reconstruct an organism's genome and the output consists of a set of continuous fragments called *contigs*.

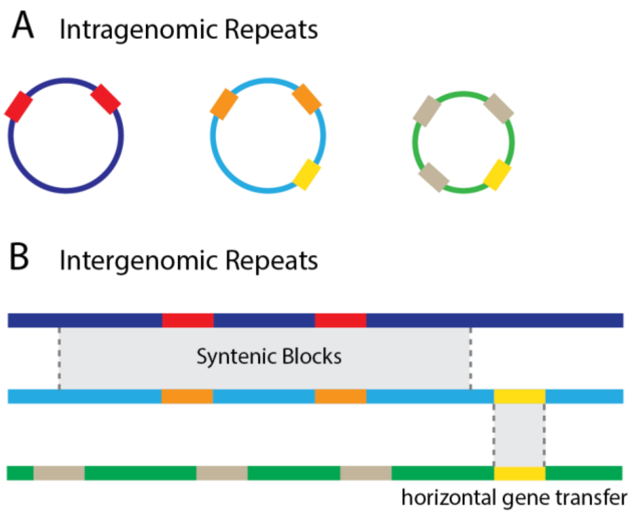
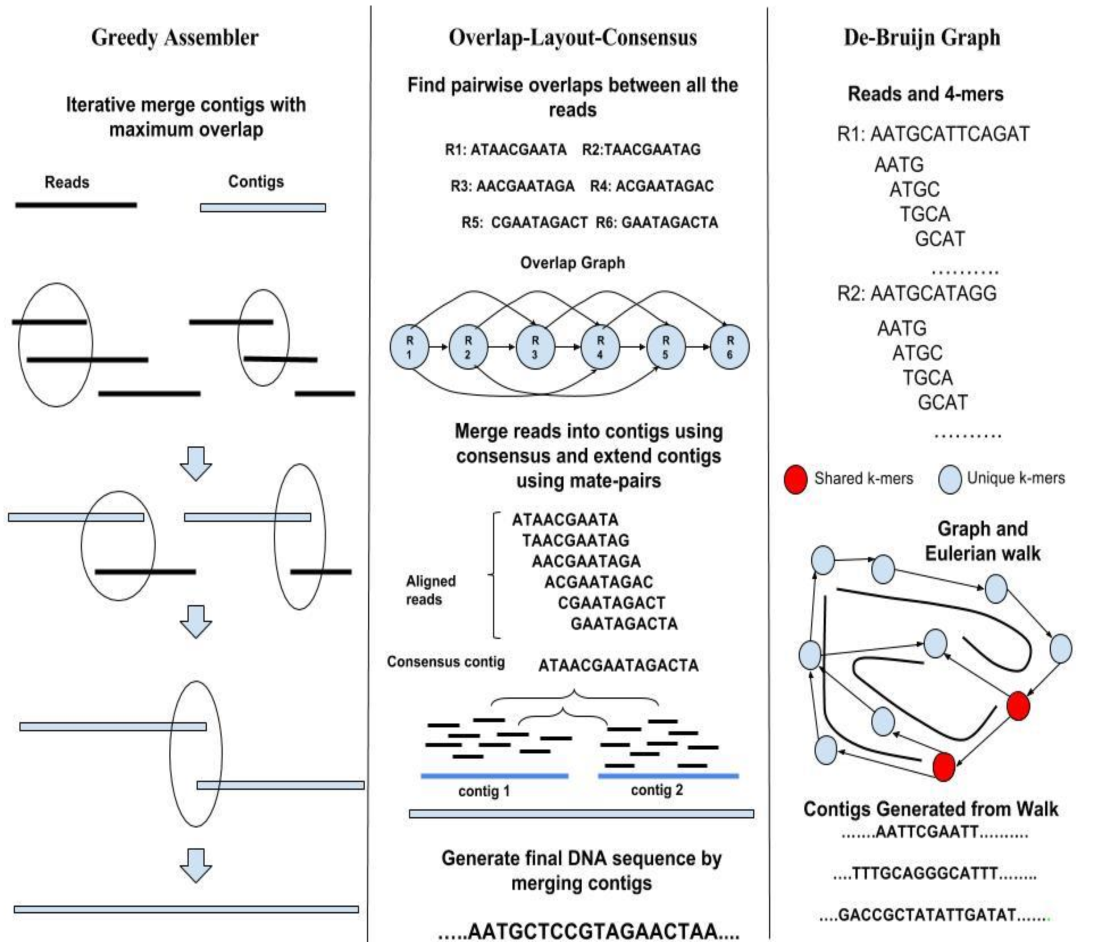


Figure 1.2. The challenge of repeats in metagenomes.
Three genomes are used to depict intragenomic (A) and intergenomic (B) repeats. The dark blue and light blue genomes represent two closely related strains and the green genome an unrelated strain. Within the genomes the red, orange, and tan blocks represent inparalogs. The yellow blocks represent a horizontal gene transfer event between the light blue and green genomes. In traditional assembly, any reads longer than the inparalog blocks (red, orange) would be sufficient to fully resolve the genome. In metagenomic assembly, reads longer than the full syntenic block (gray) would be necessary.

1.3.1 De novo assembly

Currently, most state of the art *de novo* genome assemblers use a graph-based approach, where the problem can be formulated as a Hamiltonian or Eulerian path problem [8], depending on how the reads and overlaps are defined. Ideally, the genome assembly problem has one solution, but the graph formulations can have many solutions and finding the correct solution (genome assembly) is NP-hard [9]. Due to the computational intractability, three heuristic based methods have been developed throughout time to perform *de novo* assembly: Greedy assembly, Overlap-layout consensus assembly, and de Bruijn graph assembly—currently the most widely used technique (Figure 1.3).



Greedy Assembler
Iterative merge contigs with maximum overlap
Reads
Contigs
Overlap-Layout-Consensus
Find pairwise overlaps between all the reads
R1: ATAACGAATA R2:TAACGAATAG
R3: AACGAATAGA R4: ACGAATAGAC
R5: CGAATAGACT R6: GAATAGACTA
Overlap Graph
Merge reads into contigs using consensus and extend contigs using mate-pairs
Aligned reads
ATAACGAATA
TAACGAATAG
AACGAATAGA
ACGAATAGAC
CGAATAGACT
GAATAGACTA
Consensus contig
ATAACGAATAGACTA
contig 1
contig 2
Generate final DNA sequence by merging contigs
.....AATGCTCCGTAGAACTAA....
De-Bruijn Graph
Reads and 4-mers
R1: AATGCATTCAGAT
AATG
ATGC
TGCA
GCAT
R2: AATGCATAGG
AATG
ATGC
TGCA
GCAT
Shared k-mers
Unique k-mers
Graph and Eulerian walk
Contigs Generated from Walk
.......AATTCGAATT..........
....TTTGCAGGGCATTT........
....GACCGCTATATTGATAT.......

Figure 1.3. Overview of different de novo assembly paradigms.
Schematic representation of the three main paradigms for genome assembly – Greedy, Overlap-Layout-Consensus, and de Bruijn. In Greedy assembler, reads with maximum overlaps are iteratively merged into contigs. In Overlap-Layout-Consensus approach, a graph is constructed by finding overlaps between all pairs of reads. This graph is further simplified and contigs are constructed by finding branch-less paths in the graph and taking the consensus sequence of the overlapping reads implied by the corresponding paths. Contigs are further organized and extended using mate pair information. In *de Bruijn* graph assemblers, reads are cut into short overlapping segments (k-mers) organized in a *de Bruijn* graph structure based on their co-occurrence across reads. The graph is simplified to remove artifacts due to sequencing errors, and branch-less paths are reported as contigs.

1.3.1.1 Greedy assembly

This approach first identifies overlaps between pairs of reads and merges reads with the best overlaps. The overlap and merging steps continue in an iterative way until all reads and overlaps are merged. The main advantages of the greedy assembly method is the simplicity of its algorithm, which makes it easy to implement, and its effectivity when the genome contains only short or no repeats. Its disadvantage, on the other hand, is that the choices made during the merging steps are locally optimal and do not consider global relationships between reads. As a result, this approach can produce incorrect assemblies within repetitive sequences.

1.3.1.2 Overlap-layout consensus (OLC)

This method was developed in 1995[9] and was used to assemble the first bacterial genome, *Haemophilus influenzae* [10][11], and the first human genome [12][13].

The OLC approach has three steps. In the first *overlap* step, it computes all pairwise overlaps with a dynamic programming-based alignment algorithm. The complexity of this computational step is quadratic in terms of the number of reads. Then an overlap graph is generated from both the reads and pairwise overlaps, using reads as vertices and overlaps as edges. In the second *layout* step, the overlap graph is simplified to identify a path (or “layout”) of the reads along the genome that corresponds to its sequence. And, finally, in the *consensus* step the layout is used to construct a multiple alignment of the reads to infer the sequence of the genome.

This approach is effective at high error rates, but its efficiency is reduced with high depth of coverage due to the complexity of the overlap computation step. Therefore, this approach is more suitable for relatively long reads such as those generated by first and third-generation technologies, and it is also particularly beneficial in assembling reads with high error rates, such as those generated by third generation technologies.

1.3.1.3 De Bruijn Graph (DBG)

The DBG approach became popular with the appearance of second-generation sequencing technologies, which increased the throughput to hundreds of millions of reads, as opposed to the Greedy and OLC assembly approaches that were designed for first-generation sequencing, and which did not scale well.

The reads are used to construct a DBG as follows: each read is decomposed into overlapping segments of equal length *k*, called *k-mers*. The k-mers become the nodes of the graph, and the edges connect nodes with k-1 matching bases. In this approach reads are not explicitly aligned to each other, rather their overlaps can be inferred from the fact that they share k-mers.

The DBG is a multigraph due to repeats. Repeats create additional edges in the graph, increasing the number of possible traversals. Given a collection of all k-mers in a genome sequence, the assembly problem reduces to finding an Eulerian path—a path through the graph that visits each edge once.

The de Bruijn formulation above assumes perfect data and makes the assumption that for a given read length k, we are given all length-k substrings of a genome as well as the number of times they occur. In practice, not all substrings are obtained, and several factors impact the performance of de Bruijn graph assemblers: (i) sequencing errors; (ii) repeats; and (iii) the depth of sequencing coverage. The interplay between these factors drives the choice of optimal k-mer size for a specific application as well as the ultimate performance of an assembler.

Unlike the Overlap-Layout-Consensus approach, the DBG paradigm is affected by read errors. Sequencing errors create incorrect k-mers thereby increasing the complexity of the graph and making it more difficult to identify an unambiguous reconstruction of a sequence. These errors must be eliminated prior to identifying an Eulerian path in the graph. Every error impacts at most k different k-mers, thus the impact of sequencing errors increases with the size of k. As a result, assemblers often include a correction step or assume pre-corrected data as input. Initial de Bruijn assemblers used spectral correction [14], which attempts to make a minimum number of changes in a sequence to make it consistent with correct or "solid" k-mers [15,16].

Repeats create ambiguity in the reconstruction of the genome and therefore a larger possible space of solutions must be explored [7]. Without further information, an assembler can randomly choose one of the branches, possibly leading to assembly errors, or decide to break the assembly, leading to fragmented results. Large values of k reduce the complexity of the graph and impact of repeats, but using such values requires longer sequences (longer than the k-mer size) as well as a higher depth of coverage, leading to an increased impact of sequencing errors (each error impacts k different k-mers). Conversely, shorter k-mers mitigate the impact of sequencing errors but lead to a higher impact of repeats on assembly effectiveness. Assuming uniform error and random sequencing, it is possible to compute the expected surviving coverage for a given k-mer size and input coverage [17]. These trade-offs represent a key component of the algorithmic choices made by the assembly software and also guide the empirical choices made by users of assembly tools.

Finally, the depth of coverage impacts the connectivity of the DBG graph. A path stretching from one read to another across an entire genome can only be found if adjacent reads share k-mers. At low depths of coverage, the adjacent reads are only expected to overlap by a small extent, and as a result the assembly is only possible for small values of k.

1.3.2 Reference-guided assembly

The reference-guided assembly approach consists of two steps: first, all the reads are aligned against the reference genome; then a consensus sequence is generated by calling a base at each position where reads have mapped along the reference genome. This approach is more effective than *de novo* assembly in resolving repeats and is thus able to get better results than *de novo* approaches especially at low depths of coverage. Long repeats are still a challenge as they lead to an ambiguous alignment of reads against the genome, though the use of mate-pair information can partly mitigate this issue and can help to identify the correct placement of reads. At the same time, the effectiveness of the comparative assembly approach depends on the availability of a closely related reference sequence. Differences between the genome being assembled and the reference can lead to either errors in reconstruction or to a fragmented assembly. The AMOScmp [18,19] comparative assembler attempts to identify such polymorphisms and rearrangements between genomes, and breaks the assembly at these locations in order to avoid mis-assemblies.

Several tools were developed to help augment or improve *de novo* assemblies with the help of reference genomes. OSLay [20], Projector 2 [21], ABACAS [22] and r2cat [23] simply use a reference sequences to identify the correct order and orientation of contigs from a *de novo* assembly. An extension of this approach was proposed by Husemann et al. [24] that leverages information from multiple related genomes, weighted by their evolutionary distance from the sequence being assembled. Scaffold_builder [25] also provides functionality to join together contigs that were left unassembled by the *de novo* approach, thereby helping improve the assembly through the use of a reference sequence. Finally, E-RGA [26] performs *de novo* and reference guided assembly independently first and then merges two assemblies later using a novel data structure called merge graph to avoid mis-assemblies and ambiguous overlaps.

1.4 Metagenomics

Metagenomics studies microbial samples directly taken from the environment. This technology allows research in human microbiome, soil, air, bodies of water, surfaces, and virtually any place where there is a community of interest. The advantage of metagenomic sequencing over single genome sequencing is the possibility of studying all the archaea, bacteria, plasmids, and viruses present at a given time in a sample. This same feature is also a challenge, for the microbial composition of the sample is unknown.

In the following chapter we further outline several approaches developed to address both the composition and the assembly of a metagenomic sample.

Chapter 2: Related work

2.1 Discovering microbes present in a metagenomic sample

Finding and quantifying the composition of a microbial community is a fundamental part of metagenomics. This process is both biologically and computationally challenging.

In this chapter, we outline key biological concepts to understand how microorganisms can be computationally classified and quantified. Then, we describe current computational approaches used to classify and determine the composition of a metagenomic sample.

2.1.1 Taxonomic classification of metagenomes

Taxonomy classifies living organisms into eight ranks: domain, kingdom, phylum, class, order, family, genus, and species. Each taxonomic rank is called *taxon* (plural *taxa*). Bacteria and archaea—the organism of interest in a metagenomic sample—belong to the prokaryotic domain in the tree of life and can be further classified into a taxonomy rank below species called *strain*—which are genetic variants (or subtypes) within a species.

Taxonomic profiling is the computational process of inferring which taxonomic ranks are present in a microbial community (*taxonomy classification*) and estimating their relative abundances [27]. In metagenomics, marker genes are genes that are conserved across species, and thus are suitable to discriminate between taxonomic ranks. The most widely used approaches for metagenomics taxonomy profiling use two type of marker genes universally present in prokaryotes: single genetic markers called 16S rRNA genes, and protein coding single-copy orthologous genes.

In bacteria and archaea, the genes coding for the 16S ribosomal RNA (16S rRNA), part of the 30S small ribosomal unit, are referred to as 16S rRNA genes. These genes sequences consist of nine conserved regions separated by nine hypervariable regions (V1-V9). Highly conserved regions can be used as PCR primer bonding sites to amplify and sequence one or more hypervariable regions of the 16S rRNA gene, which, in turn, are used to identify the phylogeny of microorganism [28]. Such characteristics have made 16S rRNA sequencing one of the most widely used approaches to characterize the taxonomic diversity of a metagenomic sample.

After sequencing, 16S rRNA analysis pipelines [29–31] start by clustering reads based on sequence similarity into Operational Taxonomic Units (OTUs). Then, a representative sequence from each OTU is compared against curated 16S rRNA reference databases [32–35] to assign taxonomic labels. The taxonomic resolution of 16S pipelines is usually limited to genus level.

The 16S approach has several known shortcomings: (i) different organisms contain different copy numbers of 16S rRNA genes, introducing abundance estimation biases [36]; (ii) the amplification process introduces biases [37,38]; (iii) targeting different sub-regions of the 16S rRNA gene can influence the taxonomic assignment [39]; and (iv) problems differentiating species in an accurate and consistent manner [40].

One of the possible explanations of the 16S rRNA gene problems to delineate taxa is its extremely slow rate of evolution. Thus, organisms from closely related but different taxa (e.g. different species from the same genera) might not have evolved fast enough to diverge in their 16S rRNA gene sequences [41,42]. Single-copy protein-coding orthologous genes, usually called *single-copy marker genes*, evolve faster than 16S rRNA and have been shown to have more power at resolving the relationships of closely related species [43]. Moreover, single-copy marker genes overcome many of the shortcomings of 16S rRNA genes. First, single-copy marker genes can be retrieved by sequencing the whole metagenomic dataset instead of targeting one gene, avoiding amplification biases. Second, they are not biased by copy number variation, allowing a more accurate abundance estimation of metagenomes. Lastly, single-copy marker genes can provide microbial species boundaries at higher resolution than 16S rRNA genes [44,45].

Several studies (Ciccarelli et al. [46] , Sorek et al. [44]) identified 40 universal single copy marker gene families that are present in all bacteria and archaea and can be used to reconstruct a phylogenetic tree [43,44]. These marker genes families are available in the Clusters of Orthologous Groups of proteins (COGs) public database [47,48]. Single copy marker genes can be extracted from microbial genomes using Hidden Markov Models (HMMs) trained on protein alignments [45]. Single-copy protein-coding orthologous genes are currently the most used marker genes in metagenomic studies and are part of some of the methods described below.

2.1.2 Current methods for taxonomic classification of metagenomes

The taxonomic classification of metagenomes is computationally challenging for two reasons. First, high-throughput sequences technologies generate millions of reads that need to be analyzed. Second, there are hundreds of thousands microbial genomes available in public databases and the number is constantly increasing. Thus, the number of comparisons that need to be performed to analyze a metagenomic sequencing dataset is considerably large.

In this section, we highlight published methods for metagenomic taxonomy classification. Most methods gather genomic, genetic, and taxonomy information from the public NCBI Reference Sequence Database (RefSeq) [49–52]. Several methods performed well in a recent review [53].

2.1.2.1 Whole genome alignment-based methods

The most intuitive approach for predicting the composition of a metagenomic sample is comparing each read to a database of reference genomes. This task can be accomplished by traditional methods based on local sequence alignment [54,55], which are highly accurate. Yet, while effective, aligning each read individually to a database of whole genomes can become prohibitively slow. Here we described the most popular alignment-based methods.

BLAST (basic local alignment search tool) [56,57] is the most widely used software suite for sequence alignment (at nucleotide and protein level) and has been shown to align reads with greater accuracy than other sequence alignment methods [58]. BLAST was designed to find local similarities between one or more "query" sequences and one or more “subject” sequences within a database. The intuition behind BLAST is that if the query and subject sequences are highly similar, they will contain exactly matching k-mers or “seeds”. Before running a BLAST search, a database is created from subject sequences by decomposing them into seeds (k-mers of length 7 to 11) and then storing them in a hash table. During the search, BLAST performs local sequence alignment using a seed-and-extend algorithm. In the seed step, the query is decomposed into seeds and then looked up into the hash table to locate seed matches between the query and subject. In the extend step, matching seeds are joined and extend using the Smith-Waterman alignment algorithm [59]. In a metagenomic sequencing experiment, a massive set of reads correspond to the query and the hundreds of thousands microbial reference genomes correspond to the subject. Although BLAST was not designed for metagenomics and it is computationally intensive, it has been incorporated in several metagenomic classification tools described below as a pre-filter for read classification due to his accuracy.

MegaBLAST [60], which is part of the BLAST software suite, was designed to compare highly similar sequences. MegaBLAST uses a greedy algorithm to perform gapped alignments between nucleotide sequences, and longer seeds (length 28) to reduce the number of alignments and accelerate the search. This is the only tool from the BLAST+ software suit that can compare metagenomic reads to a reference genome database in a feasible amount of time. MegaBLAST can also serve as a pre-filter for read classification.

DIAMOND (double index alignment of next-generation sequencing data) [61] is a protein-based method similar to BLASTx—BLAST module that translates the query into its six reading frames and compares it to a protein database— to align read queries to a protein database. Similar to MegaBLAST, it uses a longer seed (length from 15 to 24) to speed up the search of its BLASTx-like approach, and its main novelty is the use of double-indexing to determine the list of all seeds and their locations in both the query and subject. Double indexing improves cache locality, thus reducing memory usage. Although not designed for metagenomics classification, a DIAMOND search is usually followed by MEGAN [55], which post-processes sequence alignments and assigns each read to taxa using the lowest common ancestor (LCA) algorithm.

Kaiju [62] is a protein-based classifier that finds maximum inexact matches on the protein-level using the Burrows–Wheeler transform (BWT) [63] and FM-index [64]. The use of the FM-index to store the reference genomes reduces memory requirements compared to both BLAST and DIAMOND. Kaiju first translates each read into six-reading frames and then searches for MEMs (Maximal exact matches) in the FM-index. Then taxonomic assignment is done by assigning reads to the longest MEM, or to the LCA taxon if a read matches multiple taxa. Kaiju also has a greedy search mode which allows some mismatches by searching backwards in the BWT.

2.1.2.2 Marker gene alignment-based methods

As described in the previous section, marker genes are ideal for taxonomic profiling of metagenomic samples. While traditional sequence alignment methods can become prohibitively slow, by using marker genes instead is much faster due to the reduced size of the subject database [62,65–67]. Thus, traditional sequence alignment methods, such as BLAST, can quickly align reads to marker genes while maintaining high accuracy. The following methods are based on marker gene sequence alignment.

MetaPhyler [65] is a taxonomic classifier that relies on 30 marker genes as a taxonomic reference. First, MetaPhyler aligns a metagenomic sample against a marker gene database using BLAST. It then classifies each read individually based on its best blast-hit alignments to the database, and it uses different thresholds (automatically learned from the reference database) for each combination of taxa, reference gene, and sequence length. MetaPhyler achieves genus and species level taxonomic resolution.

MetaPhlAn (Metagenomic Phylogenetic Analysis) is a taxonomic classifier that relies on a clade-specific marker gene database. Reads are aligned to the marker gene database using Bowtie2 [68]. The total number of reads in each clade is normalized by marker gene length to then provide a relative abundance of each taxon. MetaPhlAn achieves genus and species level taxonomic resolution.

2.1.2.3 K-mer-based methods

K-mer based methods provide a fast identification of a metagenomic sample by relying on exact-match database queries instead of alignment. To achieve this, these methods pre-compute all k-mers contained in a database of complete microbial genomes [69–71].

Although k-mer approaches were created to achieve maximal speed, their main drawback is its substantial memory requirement. More recently, methods using a subset of k-mers to reduce the dimension of the problem have been developed [72–74], drastically reducing memory requirements. Regardless of the efficiency of k-mer-based approaches, a main shortcoming is their lower accuracy compared to alignment-based sequencing methods.

Kraken [71,75], the first k-mer based taxonomic classifier, uses a hash-based index to store a genome's k-mers along with its taxonomic label. If a k-mer is shared across multiple taxa, the k-mer is stored along with the LCA of those taxa. During the classification, Kraken decomposes each read into its constituent k-mers and then maps each k-mer to the database with an inexpensive table lookup. Because Kraken assigns reads to the LCA of taxa, many reads don't get specific labels assigned. To tackle this problem, BRACKEN (Bayesian Reestimation of Abundance after Classification with Kraken) [76] was designed to re-estimate taxonomic abundance from Kraken results, and it estimates abundance by redistributing read assignments in the taxonomic tree using Bayesian probabilities. Bracken achieves genus and species level taxonomic resolution.

CLARK [70], similar to Kraken, builds a database of genome's k-mers. However, CLARK reduces the size of the k-mer by storing only species or genus-level specific k-mers and removing nonunique k-mers and rare k-mers, which also reduces noise during the classification. CLARK-S [77] improves CLARK's sensitivity by replacing fixed-length k-mers with target-specific or discriminative spaced k-mers.

Mash Screen [78] is an extension of Mash [73], a tool that uses MinHash dimensionality reduction techniques to quickly calculate the approximated distance between two genomes via Jaccard index. Mash Screen introduces the concept of "screen", in which a genome database is tested for their containment within a set of metagenomic reads. For each reference genome, Mash Screen computes a containment score that measures the similarity of the reference genome to a metagenomic dataset. Similar to BLAST and DIAMOND, Mash Screen was not designed for taxonomy classification but can serve as a pre-filtering step.

2.1.3 Metrics to evaluate taxonomy classification of metagenomes

The metrics selected to benchmark metagenomic classifiers greatly influence their relative rankings and performance. Different metrics have been applied for evaluating the binary classification task of predicting taxa presence or absence. The most commonly metrics for presence or absence in metagenomic classification used across benchmarking studies [79–81] are *precision*, *recall*, *F1 score*, and the *Jaccard index.*

Precision, also known as positive predictive value, refers to the proportion of true positive classifications, out of the total number of classifications attempted:

$$\boldsymbol{Precision} = \frac{TP}{TP + FP}$$

In the context of metagenome classification, precision can be calculated by taxon as the proportion of correct classification in the sample divided by the number of total classifications identified by the method.

Recall, also known as sensitivity or true positive rate, is defined as the proportion of true positive classifications out of the total true positives plus false negatives being tested:

$$\boldsymbol{Recall} = \frac{TP}{TP + FN}$$

In the context of metagenome classification, recall can be calculated by taxon as the proportion of correct classifications divided by the number of distinct elements in the sample.

There is a fundamental trade-off between precision and recall. Depending on the downstream analysis being performed after taxonomy classification, achieving either a higher precision or a higher recall can be preferred. Precision can represent a measure of exactness or quality, while recall a measure of completeness or quantity. In reference-based methods, such as reference-guided metagenome assembly and pangenome-based analysis, it is desired to retrieve all-known taxa present in the classification (higher recall) without sacrificing precision.

The F1 Score measures the balance between precision and recall. The F1 score is the harmonic mean of recall and precision, weighting them equally in a single metric:

$$\boldsymbol{F1\ Score} = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

Finally, the Jaccard index refers to the number of true positives (intersect between predicted and real communities) divided by the true positives plus the false positives and negatives:

$$\boldsymbol{Jaccard\ index} = \frac{TP}{TP + FP + FN}$$

To provide a realistic estimate of precision, recall, F1 score, and Jaccard index for benchmarking, all taxonomy classifiers should be tested using the same abundance threshold.

2.2 Metagenomic assembly

The goal of metagenomic assembly is to reconstruct the genomes of all organisms in a given microbial community. Metagenomic datasets are commonly sequenced using short second-generation reads. Thus, similarly to the *de novo* genome assembly problem, the short-metagenome assembly problem can be formulated as a DBG problem. In a metagenomic context, the task of a DBG metagenome assembler is not just to reconstruct one path through a graph, but a multitude of paths that come together and split apart at different places.

As described in Chapter 1, the main factors affecting the performance of single genome DBG assemblers are sequencing errors, repeats—the presence of repetitive DNA segments within an organism's genome (intragenomic repeats)—, and the depth of sequencing coverage. Furthermore, the problem of reconstructing a metagenome is complicated by: (i) the presence of strain variants; (ii) the combination of both intragenomic and intergenomic repeats (DNA segments shared between distinct organisms, Figure 1B); and (iii) the uneven sequencing coverage within a metagenome.

Strain variants create a challenge similar to sequencing errors and in highly polymorphic samples the assembly result will likely be fragmented [82]. Furthermore, distinguishing true biological differences from sequencing errors becomes nearly impossible in a metagenomic setting.

Intragenomic repeats are generally small (usually smaller than ~10,000 bp in bacteria [83,84]) but intergenomic repeats can be nearly the entire chromosomes for closely related strains. Multiple bacteria from the same species in a community (strain variants) may differ in just one gene, in which case almost the entire genomes are inter-genomic repeats. The decision of whether such differences can be ignored when reconstructing the corresponding genome, or whether it is proper to reconstruct individual-specific genomes, is not only computationally difficult but also ill-defined from a biological point of view.

Due to uneven sequencing coverage within a metagenome, coverage heuristics employed for single genome assembly can no longer be used to detect repeats [85]. Organisms sequenced at high depth of coverage (often exceeding 1000-fold) lead to high computational costs. In a DBG, the higher depth of coverage amplifies the effect of errors on the assembly graph and may even confuse error correction algorithms (simply by chance multiple random errors can confirm each other). Organisms sequenced at low depth of coverage (less than 10-fold) can be assembled using shorter k-mers, but this strategy can lead to a higher impact of repeats on the assembly.

Due to these complications, despite initial attempts, algorithms developed for a single genome assembly cannot be applied directly to metagenomics data. Instead, several approaches, mostly under the DBG paradigm, have been developed that explicitly consider the specific characteristics of metagenomic datasets.

2.2.1 Current methods for metagenomic assembly

Various *de novo* assemblers [3,4,8,10] have been developed and applied to the assembly of metagenomes from massive amounts of short reads. In this section, we highlight three published algorithms developed specifically for *de novo* metagenomic assembly that perform well in recent reviews [86,87].

IDBA-UD [88] is part of the IDBA (Iterative *De Bruijn* Graph *De Novo* Assembler) [89] suite of assemblers. A key algorithmic component of IDBA assemblers is the use of multiple k-mer sizes to address the trade-offs of different choices of k. To improve the DBG, IDBA-UD iterates through a range of k-mer values in a stepwise fashion. Sequencing errors are corrected at each iteration, reducing its impact in the assembly. The assembly graph becomes more resolved with increasing k-mer size in each iteration step, resulting in a more contiguous assembly.

MEGAHIT [90] relies on the same multiple k-mer strategy as the [90] IDBA assemblers [89]. MEGAHIT is currently the most efficient *de novo* assembler largely due to its use of efficient data-structures for storing the *de Bruijn* graph[87]. Memory requirements are reduced by using a new data structure, called succinctly *de Bruijn* graph [91]. Memory is further reduced by eliminating k-mers below a defined frequency threshold from the graph. This step also minimizes the negative impact of sequencing errors on the assembly. To retain k-mers from low abundance organisms, distinguishing them from errors, MEGAHIT reconsiders discarded k-mers in low-coverage regions of the assembly graph.

MetaSPAdes [92] is a metagenomic-specific version of the SPAdes assembler [93], and it was originally designed to address two major issues of single cell sequencing data: the uneven read coverage and chimeric sequences—issues that are also relevant to metagenomic assembly. The main innovation in these family of assemblers is the use of paired-end information during the assembly process rather than afterwards [94]. This information is incorporated in the DBG by using pairs of k-mers separated by an estimated distance. Similar to IDBA-UD and MEGAHIT, SPAdes follows an iterative multiple k-mer approach, and, moreover, it uses the complete read information together with the preassembled contigs at every step. In addition, metaSPAdes was extended to handle strain variation; micro-variations between highly similar "strain-contigs" are combined to form high quality consensus sequences, aiming at the best possible representation of each species instead of every strain variant. MetaSPAdes is slower than IDBA-UD and MEGAHIT and it is not scalable to large datasets[87].

Despite advances in metagenomic assembly algorithms [88,95–98], the assembly problem remains computationally challenging. As mentioned in Chapter 1, reference-guided assembly is more effective than the *de novo* assembly when sufficiently closely related sequences are available, yet, it has not been applied to metagenomics. In this regard, a metagenomic reference-guided approach will be discussed in Chapter 4.

2.2.2 Metagenome assembly validation

The validation of genome assemblies has been an active area of interest since the development of the first genome assemblers in the late 1970s [99]. The most commonly used metrics to evaluate the quality of metagenomics assembly can be classified into (i) contiguity-based, (ii) completeness-based, (iii) reference-based, and (iv) consistency-based. Contiguity, completeness and consistency-based metrics rely on features of the assembled data, seeking to identify internal inconsistencies indicative of potential assembly errors. Reference-based metrics need to know the “ground truth” genomes used for generating the metagenomic reads.

2.2.2.1 Contiguity-based metrics

Contiguity-based metrics are the most intuitive. These metrics evaluate how fragmented the final assembly is. The most common metric used to compare assemblies, the *number of contigs* (total number of assembled contigs reported by each assembler), attempts to assess how far the assembly is from the ideal goal of one contig per chromosome. Since most assemblies consist of many small contigs, usually due to sequencing errors or other artifacts, this metric can be misleading. More robust measures are the *Contig Number at 1Mbp* (the number of contigs required to exceed 1Mbp) and the *Assembly size at 1MBp* (the size of the largest contig C such that the sum of all contigs larger than C exceeds 1Mbp). The choice of 1Mbp is driven by bacterial and archaeal genome sizes: while the average length of such organisms is usually longer than 3Mbp, many of them encompass multiple replicons of shorter length (~0.3-1.3Mbp) [100]. Since the real average length of replicons in a metagenome is arbitrary, using 2-5Mbps is also reasonable.

The contiguity metrics already described do not take any correctness information into account and can be misled by accepting errors—a single long contig can be constructed by concatenating all the reads in an incorrect order.

2.2.2.2 Completeness-based metrics

The most intuitive completeness-based metric is the *total assembly* size, which is the total number of bases in the assembly. The gene information contained in a metagenome assembly can also be used not only to evaluate completeness but also to measure how useful an assembly may be to downstream analyses. As genes are used to address biological questions, a greater number or density of genes results in more information available for testing biological hypotheses. Single-copy marker genes, here referred simply as marker genes, can be assumed to exist in all newly assembled bacterial and archaeal sequence. Thus, an assembly where some of these genes are missing can be assumed to be incomplete. Additionally, complete genes and marker genes metrics can be used as a measure of correctness, as assembly errors would disrupt ORFs (open reading frames). The completeness-based metrics used to represent a metagenome's gene content are *complete genes* and *complete marker genes*, which are the median number of fully reconstructed complete genes and the marker genes, respectively.

2.2.2.3 Reference-based metrics

To calculate reference-based validation metrics, the assembly is compared to a database containing previously assembled genes or genomes [101,102]. The most common reference-based metrics are: (i) *Genome Recovery* (%), which is the median percentage of each truth genome that is recovered; (ii) *Total Aligned Length*, which is the sum of the length of contigs aligned to the truth genomes; (iii) *Total Unaligned Length*, which is the sum of the length of unaligned contigs; and (iv) *NGAx*, which is the length of the contig that covers at least half the reference genome.

Reference-based metrics are particularly effective in benchmarking experiments that try to reconstruct communities with known composition. However, these metrics can have limited effectiveness in real datasets. For example, metagenomic segments originating from a genome for which no reference sequence is available cannot be verified through a reference-based approach. It is also difficult to determine whether differences between an assembled contig and the reference genome are true differences o errors.

2.2.2.4 Consistency-based metrics

It is often important to determine where exactly errors were introduced in the assembly, either to correct these mistakes, or to ensure that the errors do not influence the results of downstream analyses. The major types of assembly errors are: repeat collapse, insertions, deletions, and inversions (Figure 2.1).

Consistency-based metrics evaluate assembly errors by aligning sequencing reads to the assembly and finding regions where the mappings are inconsistent.

Common consistency-based metrics include *depth of coverage*, *consensus*, *split read mapping* and *insert size consistency*.

The *depth of coverage* metric is a statistical comparison of global vs local coverage, as signature of compressed or expanded repeats. Increases in coverage show collapsed repeats, while drops in coverage or coverage gaps can show breakpoints due to insertions, deletions, and inversions. *Consensus* refers to the concordance of the consensus to the read pileup. *Split-read mapping* measures single reads with partial alignments to separate locations of a genome. Lastly, *insert size consistency* evaluates the concordance of the insert size (distance between read pairs); increase in insert size shows expanded repeats and decrease size shows collapsed repeats.

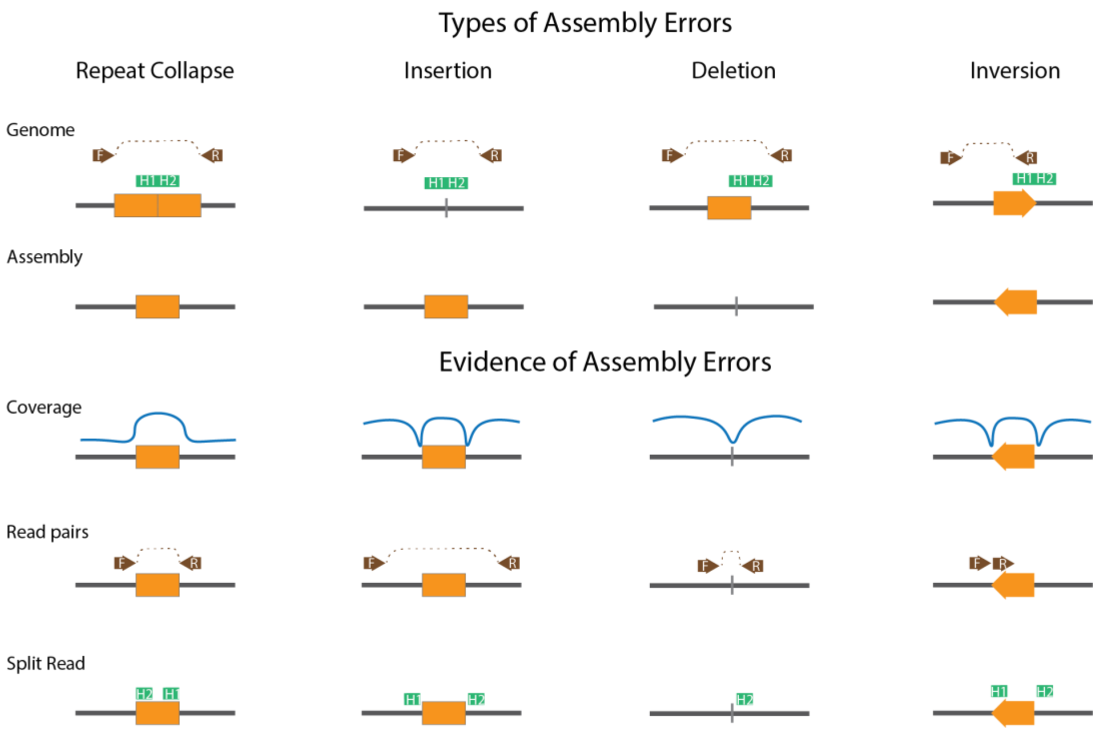


Figure 2.1. Metagenome assembly error signatures.
There are four primary types of assembly errors, repeat collapse, insertions, deletions, and inversions. These assembly errors can be identified by mapping reads to the assembly and evaluating the coverage (blue curve), distance between read pairs (green reads), and split read mapping data (green reads). Increase in coverage indicates repeat collapse whereas drops in

coverage indicate breakpoints for insertions, deletions, and inversions. Shorter than expected distance between read pairs indicates potential repeat collapse or deletion, whereas increase in distance between read pairs indicates a potential insertion. Inconsistency in read pair direction can indicate an inversion. Finally, split-read mapping data, obtained by independently aligning the first and last third of a read can be used in a similar manner to read pair information to identify assembly errors [103].

2.2.2.5 Current Methods for Metagenome assembly validation

The software packages CheckM [104] and BUSCO [105] are only based on completeness-based metrics. CheckM relies on single-copy marker genes that are specific to a genome-based lineage within a reference tree, while also supplying information to correct the assemblies. BUSCO (Benchmarking Universal Single-Copy Orthologs) evaluates assemblies by measuring single copy ortholog marker genes and it estimates contamination from the recovered genes.

The most used tool for metagenome assembly validation is the tool called MetaQUAST [106], which, unlike the previous ones that only use completeness metrics, incorporates contiguity, consistency, and reference-based metrics. MetaQUAST is a modification of QUAST [107], an isolate genome assembly validation tool that computes alignments of assembled contigs to a single reference genome. Similar to QUAST, MetaQUAST identifies mis-assemblies relative to a set of reference genomes. Additionally, metaQUAST applies a structural variant finding algorithm to distinguish between structural variants and true assembly errors.

2.3 Conclusion

In this chapter, we first introduced the concept of taxonomy classification—finding the organisms present in a metagenomic sample—and then described several methods—whole genome alignment, marker gene alignment, and k-mer-based—to accomplish this task. Secondly, we reviewed contemporary advances and challenges in *de novo* metagenomic assembly and outlined the main challenges faced by *de novo* metagenomic assemblers. Lastly, we described current methods and strategies for metagenome assembly validation and error characterization based on contiguity, completeness, consistency and references. In the following two chapters, we present our method for both taxonomy classification and metagenome assembly.

Chapter 3: Selecting references genome for metagenomic reference-guided assembly

3.1 Introduction

As mentioned in Chapter 1, reference-guided assembly is an effective approach when sufficiently closely related sequences are available, yet, it has not been applied to metagenomics. Differences between the genomes being assembled and the references can greatly affect the final assembly by either leading to errors in reconstruction or to a fragmented assembly. Therefore, selecting closely related reference genomes is a crucial step before reference-guiding the assembly of a microbial community.

The most popular metagenomic classification methods, Kraken and MetaPhlAn, are designed to achieve, at most, species level taxonomic resolution. Kraken was the first k-mer-based approach for metagenomic classification, and it is best-suited to rapidly match metagenomic sequences to large databases of complete genomes. The drawback of using k-mer approaches is that they are not as accurate as older sequence alignment-based methods. MetaPhlAn, on the other hand, maps metagenomic reads to a database of clade-specific marker genes to perform taxonomy classification. Although marker genes are a well-known resource to select biologically relevant genomes and can led to good precision and recall at species level, such genes only account for a small part of the complete microbial genome, excluding additional genomic information that can be relevant for strain resolution.

To tackle the above-mentioned problems, we designed MetaCompassRS (MetaCompass Reference Selection), a metagenome classification approach that achieves strain-level resolution by combining a marker gene sequence alignment approach with a whole genome k-mer matching approach. MetaCompassRS can be used both as a standalone software and as part of the MetaCompass pipeline described in Chapter 5.

3.2 Methods

MetaCompassRS (Figure 3.1) follows a two-stage strategy: a marker gene alignment stage and a complete genome k-mer matching stage.

3.2.1 Marker gene alignment

Each genome is assumed to have a defined marker gene set, thus, if a genome is present in a metagenomic sample, it should have a sufficient portion of its marker gene set covered by reads. We define this concept as “marker gene set containment” to estimate how well a set of reference genomes is contained in metagenomic sample. Although aligning millions of reads to a marker gene database is relatively fast, we further speed up this process by pre-filtering the reads. We use kmer-mask [108] to extract k-mers from both metagenomic reads and a marker genes database, and then filter out reads without exact k-mer matches. Next, we use Blastn to align the complete sequence of the pre-filtered reads against the complete marker gene sequences. Lastly, we estimate a marker gene set containment score for each reference genome and only keep references above a certain percentage threshold.

3.2.2 Complete genome k-mer matching

In this stage, we use the complete sequence of the pre-selected reference genomes from the first stage and the complete set of reads to re-estimate read containment. We use Mash Screen to identify which reference genomes are sufficiently contained within the reads. We then select the strains with higher containment score, which are more likely to be present in the metagenomic sample.

By combining these two stages, we take advantage of the high accuracy of marker gene alignment methods and the efficiency of k-mer based approaches to compare complete genomes.

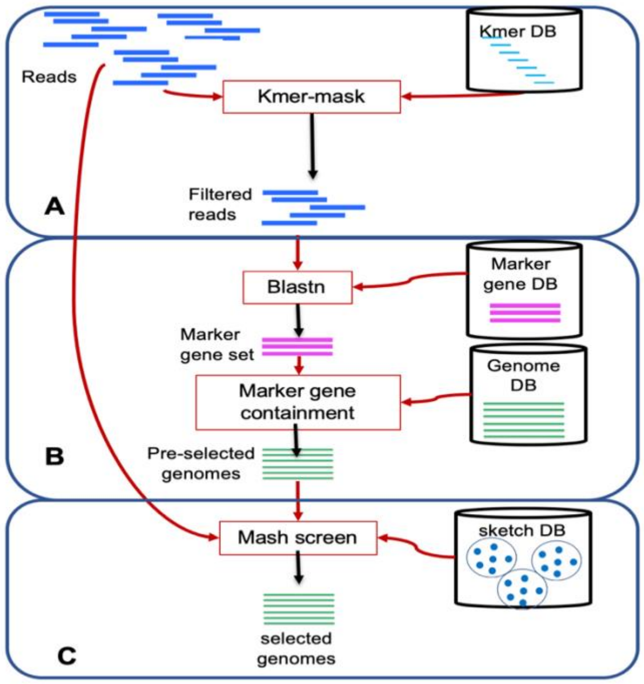


Figure 3.1. Overview of MetaCompassRS.
A) Reads are prefiltered using k-mer-mask. B) Pre-filtered reads are aligned to marker genes using blastn. Then Marker gene set containment is estimated from Blast results and complete genomes C) Preselected genomes are screened for containment using Mash screen.

3.2.3 Database construction

To create our reference database, we retrieve high-quality genome assemblies from the NCBI Refseq database, including complete genome assemblies and chromosome level assemblies (which include chromosomes, scaffolds and contigs). We also retrieve taxa, genes, and protein sequences associated to each genome.

After retrieving all the necessary information from RefSeq, we used several tools to gather marker gene information from each genome. First, use the tool FetchMG [45,109] to predict the 40 universal single-copy marker genes present in each genome. Due to the high redundancy of organisms currently available at RefSeq, many genomes share almost identical marker gene sequences. To further speed up the marker gene alignment stage (described below), we cluster almost identical marker genes with CDHIT [110]. Lastly, we use kmer-mask, part of the [111] k-mer counter package Meryl, to create a marker gene k-mer database.

Finally, we process each genome retrieved from RefSeq to gather the k-mer information used in the second stage of MetaCompassRS. We pre-compute k-mer sketches for each genome using *Mash sketch* and estimate pairwise average nucleotide identity (ANI) between genomes using *Mash dist*. The former information is used by Mash screen to estimate containment. The latter is used to filter out almost identical genomes from the final metagenomic classification.

3.2.4 Implementation details

3.2.4.1 Marker gene alignment stage

Pre-filtering reads before alignment: We pre-filter the reads to speed up the Blastn alignment. Given a k-mer database and a set of reads, kmer-mask computes the fraction of the reads which are covered by k-mers in the database. We use a seed size of 28.

Aligning reads to marker genes: We run blastn with a word size or seed of 28, to retrieve only highly similar alignments. Since closely related genomes can share the same marker genes, each read can be aligned to multiple marker genes.

Estimating marker gene set containment: After aligning reads to marker genes, we process the alignment results to estimate the "marker gene set containment" per each genome. Intuitively, this metric estimates how well a genome is contained in the reads, using a set of marker genes as representation of such genome. Given a genome "G" with "n" marker genes, we define the marker gene set containment score as the total number of marker genes bases covered by the reads divided by the sum of marker gene lengths:

$$Marker\ gene\ set\ containment\ for\ genome\ G = \sum_{i}^{n} \frac{bases\ coverd\ by\ reads}{marker\ gene\ length}$$

The number of bases covering a marker gene is calculated from the blast alignment results by extracting all the alignment intervals for that marker gene, then merging overlapping intervals, and finally adding the merged interval lengths.

We exclude from further consideration all the genomes with an estimated marker gene set containment below a certain threshold (one third of the complete marker gene set, by default).

3.2.4.2 Whole genome comparison stage.

To re-estimate read containment, we use Mash Screen with a minimum identity threshold of 0.95 percent. This step removes many reference genomes that had a sufficient marker gene containment score but were poorly contained in the reads when considering the complete genome. In case of finding multiple strains that satisfied all the criteria mentioned above, we only select the top ten strains per species.

3.3 Results

3.3.1 Evaluation of performance on Salmonella enterica simulated genome

To test the efficacy of our approach in finding the correct organism among multiple strains, we simulated reads from a strain of one of the most abundant species in our database (Table 3.1), *Salmonella enterica*—well-known food pathogen that causes gastroenteritis in humans. We chose the strain *Salmonella enterica serovar Typhimurium LT2* (available at RefSeq with accession NC_003197.2), which has close homologue genes with eight genomes from the Enterobacteriaceae family: *Salmonella enterica* serovar Typhi CT18, *Salmonella enterica* serovar Paratyphi A, *Salmonella enterica* serovar Paratyphi B, *Salmonella enterica* arizonae, *Salmonella bongori*, *E. coli* K12, *E. coli* O157:H7, and *Klebsiella pneumoniae* [112].

Table 3.1. Most abundant species in MetaCompassRS database.

Species ID	**Number of strains**	**Description**
562	3217	*Escherichia coli*
573	2022	*Klebsiella pneumoniae*
28901	1779	*Salmonella enterica*
47466	1634	*Borrelia miyamotoi*
1280	987	*Staphylococcus aureus*

We compared the performance of our approach with the most widely used marker gene and k-mer based taxonomic classifiers, MetaPhlAn, Kraken2, respectively. We also used Kraken2 with its abundance estimation companion method Bracken. We evaluated the genus and strain level classification results in terms of precision, recall and F1 score.

At the genus and species level, MetaCompassRS and MetaPhlAn made a perfect prediction (Table 3.2). In Contrast, Kraken2 and Kraken2+Bracken reported several low abundance false positives. At the strain level, MetaCompass outperformed Kraken2 and Metaphlan2 by reporting fewer false positives. Among all strains reported by the two versions of Kraken2, *Salmonella enterica serovar Typhimurium LT2* was not assigned any read. Conversely, MetaCompassRS reported the correct strain with a marker gene containment score of 99. 8% and a genome containment of 99.99%. Compared to the true *Salmonella enterica* strain, the nine false positive strains reported by MetaCompassRS (Table 3.3) shared more than 99% of identity and covered 97-99% of the true genome. This experiment highlights the effectiveness of our approach in selecting highly closed reference genomes.

Table 3.2. Taxonomy classifier predictions at different taxonomy levels for *Salmonella enterica serovar Typhimurium LT2* simulated dataset.
Note that MetaPhlAn does not provide strain level resolution.

	Kraken2	**Kraken2+Bracken**	**MetaPhlAn2**	**MetaCompass**
Genus	14	35	1	1
Species	9	59	1	1
Strains	161	240	NA	10

Table 3.3. Comparison between *Salmonella enterica serovar Typhimurium LT2* (NC_003197.2) and MetaCompassRS false positive (FP) strains.
Query cover describe how much the query genome, NC_003197.2, is covered by each FP and percentage of identity describes how similar the query is to each FP.

Accession ID	**Description**	**Query cover**	**Percentage of Identity**
NC_021151.1	*Salmonella enterica subsp. enterica serovar Typhimurium str. U288*	99%	99.99%
NZ_CP007523.1	*Salmonella enterica subsp. enterica serovar Typhimurium str. CDC 2011K-0870*	98%	99.99%
NZ_CP014051.2	*Salmonella enterica strain LT2*	100%	100%
NZ_CP014971.2	*Salmonella enterica subsp. enterica serovar Typhimurium str. USDA-ARS-USMARC-1898*	98%	99.97%
NZ_CP021909.1	*Salmonella enterica subsp. enterica strain ST1120*	99%	100%
NZ_CP028199.1	*Salmonella enterica subsp. enterica serovar Typhimurium strain CFSAN018746*	100%	100%
NZ_CP025736.1	*Salmonella enterica strain FORC_079*	99%	99.98%
NZ_CP041005.1	*Salmonella enterica strain FDAARGOS_768*	100%	100%
NZ_CP032494.1	*Salmonella enterica subsp. enterica serovar Typhimurium strain SO21*	97%	99.99%

3.3.2 Evaluation of performance on synthetic metagenomic dataset

We evaluated our method on synthetic microbial community published by Shakya et al. [113]. The synthetic sample was downloaded from the NCBI Short Read Archive (SRA) database, (SRR606249) and contains 54 bacteria and 10 archaea. Among these organisms, 55 had complete genome sequences in the NCBI RefSeq database (the database used by default by MetaCompassRS), and 9 are available only as a high-quality draft assembly. The sample contains 61 species.

At the species level, MetaCompassRS outperformed Kraken2 and MetaPhlan2 in recall, correctly predicting 59 out of the 61 species. Although MetaPhlan2 achieved the highest precision, thus, predicting fewer false positives, it only predicted 54 out of the 61 species. Both MetaCompassRS and MetaPhlan2 had almost identical F1 score (Figure 3.2A).

At the strain level, MetaCompassRS outperformed Kraken2 in all metrics, correctly predicting 59 out of the 64 strains. The 5 strains that MetaCompassRS did not predict are draft assemblies without complete reference genomes available to date. (Figure 3.2B).

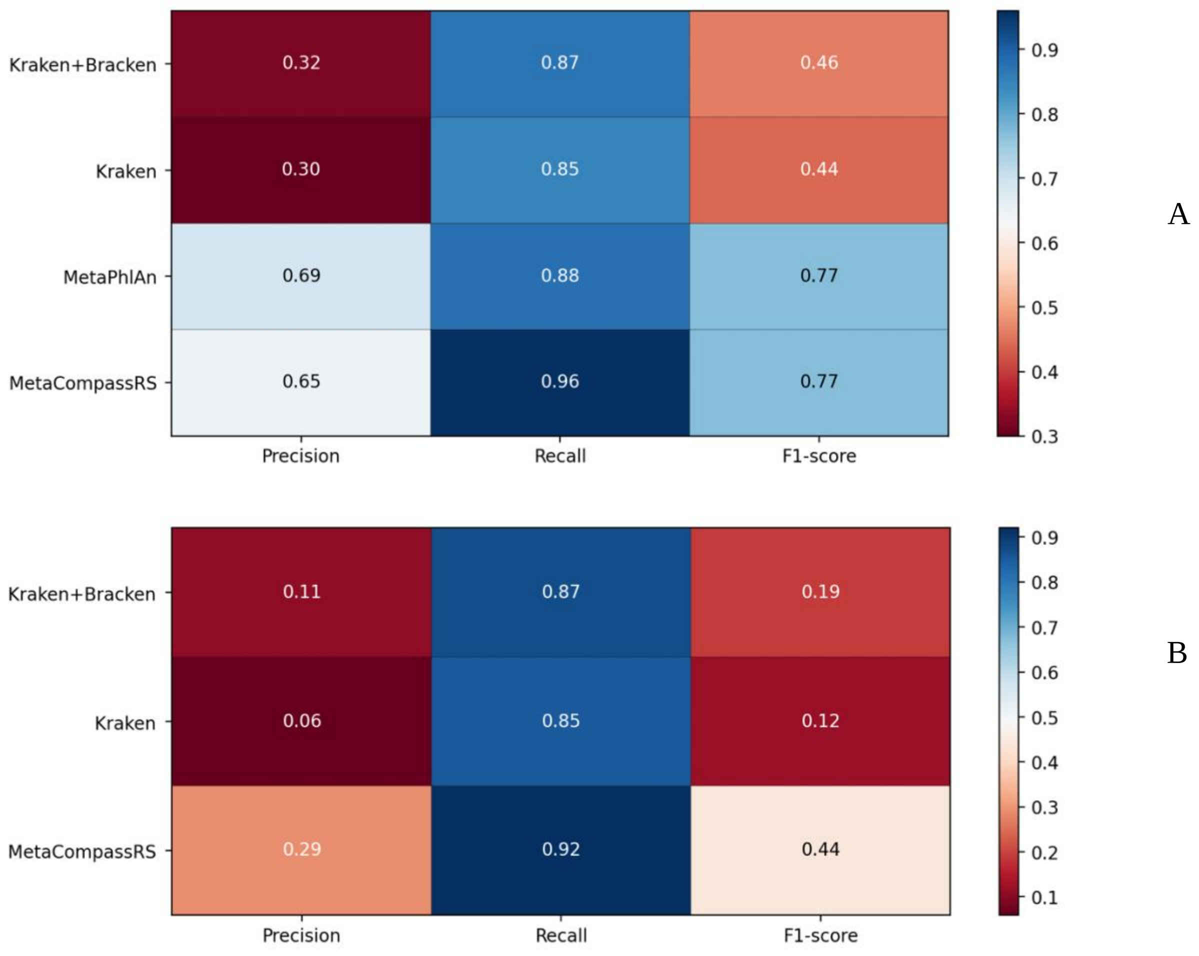
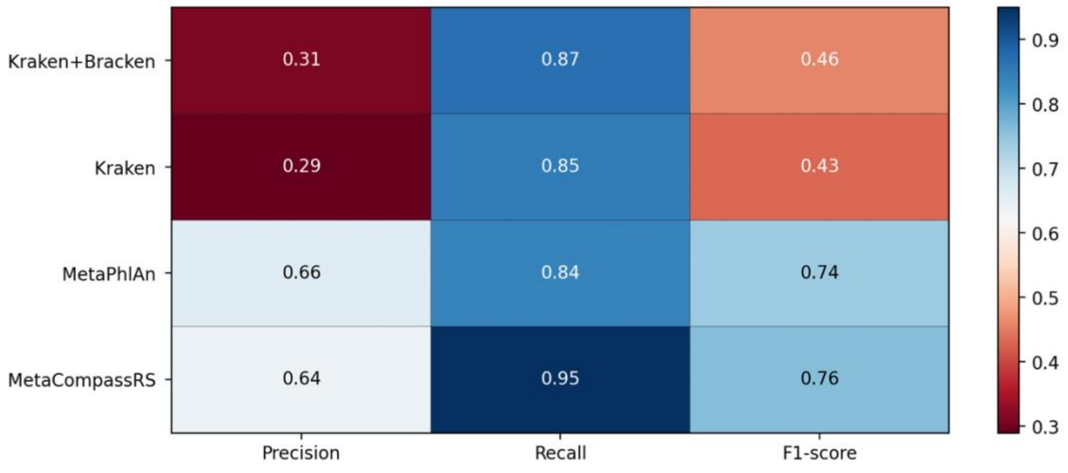


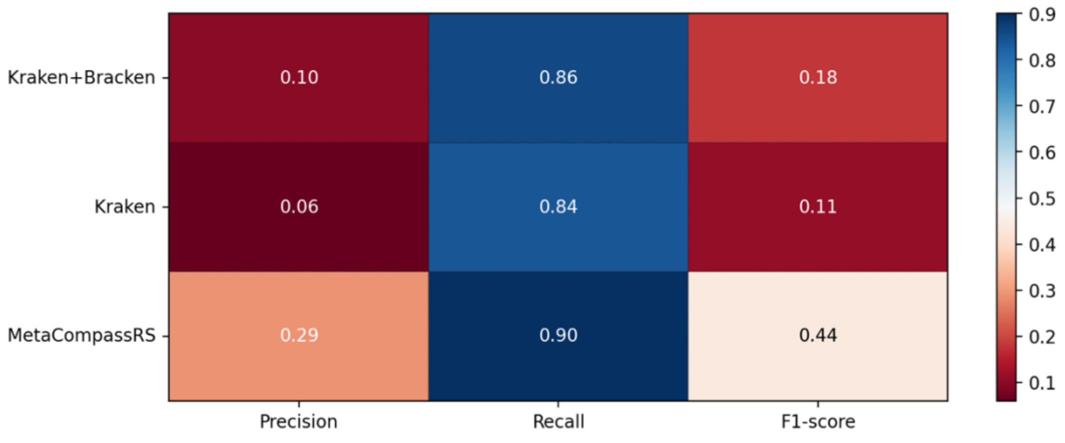
Figure 3.2. Species and strain level classification results on Shakya et al. dataset. (A) Heatmap showing Precision, Recall, and F1 score at species level. (B) Heatmap showing Precision, Recall, and F1 score at strain level.

To evaluate the ability of MetaCompassRS to classify low-coverage genomes, we downsampled the synthetic dataset to 10% of its original size. The results (Figure

3.3) highlight that MetaCompassRS is also highly effective at low coverage and outperformed the other tools in several metrics.



A



B

Figure 3.3. Species and strain level classification results on down-sampled Shakya et al. dataset.
(A) Heatmap showing Precision, Recall, and F1 score at species level. (B) Heatmap showing Precision, Recall, and F1 score at strain level.

We further evaluated the classification averaging across all datasets on the species and strain level (Figure 3.4). Only MetaCompass achieved more than 90% recall for both taxa. Similarly, only MetaPhlAn achieved more than 50% of precision.

In terms running time, Kraken2 was the fastest across all datasets. MetaCompassRS and MetaPhlAn, which are both based on read alignment, took a very similar amount of time (Table 3.4).

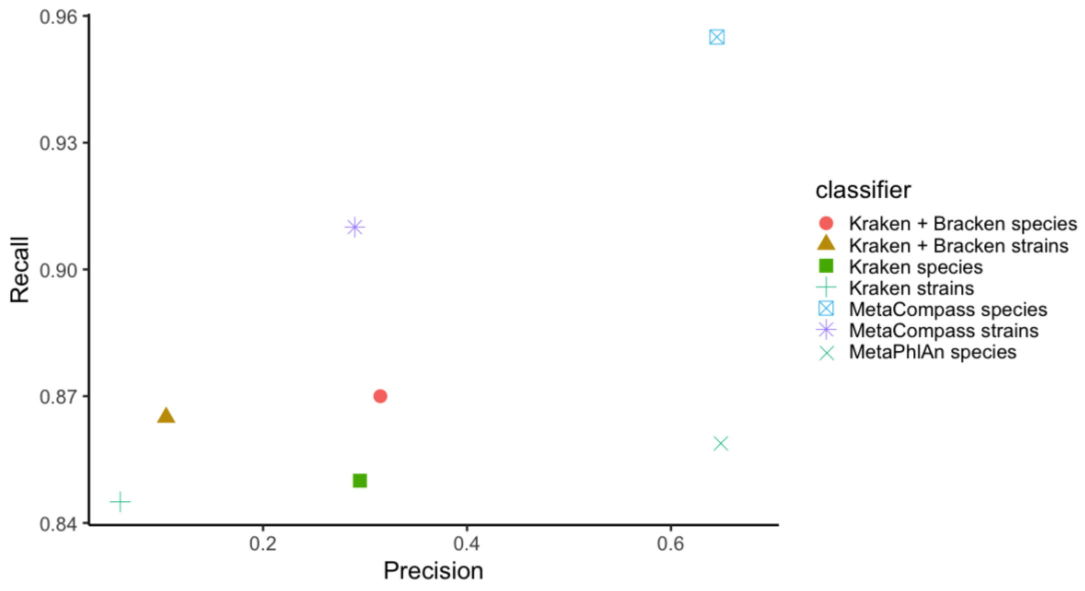


Figure 3.4. Scatter plot of Precision (x-axis) versus Recall (y-axis) across all datasets and taxa.

3.3.3 Evaluation of computational performance on simulated and synthetic metagenomic datasets

We evaluated the running time performance of MetaCompassRS on a Linux 12-core server node with 16 GB of memory using the *Salmonella enterica* simulated dataset, the Shakya et al. synthetic dataset, and the downsampled Shakya et al. synthetic dataset. The wall clock running time on this synthetic dataset for MetaCompassRS was slightly higher than Kraken2 and considerably lower than MetaPhlAn2 (Table 3.4). MetaPhlAn2 had the lowest memory usage among all taxonomic classifiers, followed by Kraken2. Note that since Kraken2 loads the database into memory, its memory usage is determined by the size of the Kraken2 database. We used the reduced size Kraken2 database (MiniKraken2 database, 8GB) because the default Kraken2 database (29 GB) required more memory than the limit used in our experiments. MetaCompassRS was able to process a 100 million read dataset using the complete MetaCompassRS database in less than 16 minutes without prohibitive memory requirements (13.06GB), highlighting the scalability of this method to large datasets.

Table 3.4. Running time for taxonomy classifiers on simulated and synthetic datasets. We evaluated the running time performance of MetaCompassRS and three taxonomic classifiers for the simulated *Salmonella enterica* sample, the full Shakya et al. sample (100 million paired-end reads), and a 10% of the original Shakya et al. sample (5 million paired-end reads). We used the default MetaCompassRS and MetaPhlSn2 databases and the reduced size Kraken2 database (MiniKraken2 database). All dataset were run using 16 GB of memory and 12 CPUs.

Classifier	***Salmonella enterica***		**Shakya et al.**		**Downsampled Shakya et al.**	
	Time (mm:ss)	**Memory (Gb)**	**Time (mm:ss)**	**Memory (Gb)**	**Time (mm:ss)**	**Memory (Gb)**
MetaCompassRS	10:21	7.42	15:32	13.06	5:38	12.40
MetaPhlAn	62:14	**3.27**	177:44	**3.05**	151:00	**2.66**
Kraken2	**1:03**	8.67	**1:33**	8.22	**1:12**	8.21
Kraken2+Bracken	1:04	8.67	1:35	8.22	1:13	8.21

3.3.4 Evaluation of performance on CAMI medium dataset

To provide a better idea of how MetaCompassRS would perform in a worst-case scenario (the closest genomes contained in the metagenomic sample are not present in the database), we used a medium complexity dataset generated by the benchmarking study CAMI (Critical Assessment of Metagenome Interpretation) [81]. From the two medium complexity datasets generated by CAMI, we used the medium complexity dataset consisting of 132 newly sequenced genomes (not present in public databases) with an insert size of 270bp. We ran MetaCompassRS on the selected medium complexity dataset and compared our species-level performance with the publicly-released results from CAMI. We only included methods that achieved species level taxonomy resolution (FOCUS [114], TIPP [115], MetaPhlAn2 [66] MetaPhyler [116], mOTU [45], Quikr [117], Taxy-pro2 [118], and CommonKmers [119]).

Notably, the least precise profiling methods (TIPP, MetaPhyler, and Quikr) had the highest recall, introducing a high false positive rate in their prediction (Figure 3.5 A). MetaPhlAn2 and MetaCompassRS achieved not only the highest precision (fewer false positives) but also the highest F1 score (balance between precision and recall) (Figure 3.5 B). The similar results obtained by MetaPhlAn2 and MetaCompassRS are expected as both tools use marker genes, which are known for being precise at higher taxonomic ranks up to species level. We did not evaluate strain level predictions as CAMI didn't not include such results in their study.

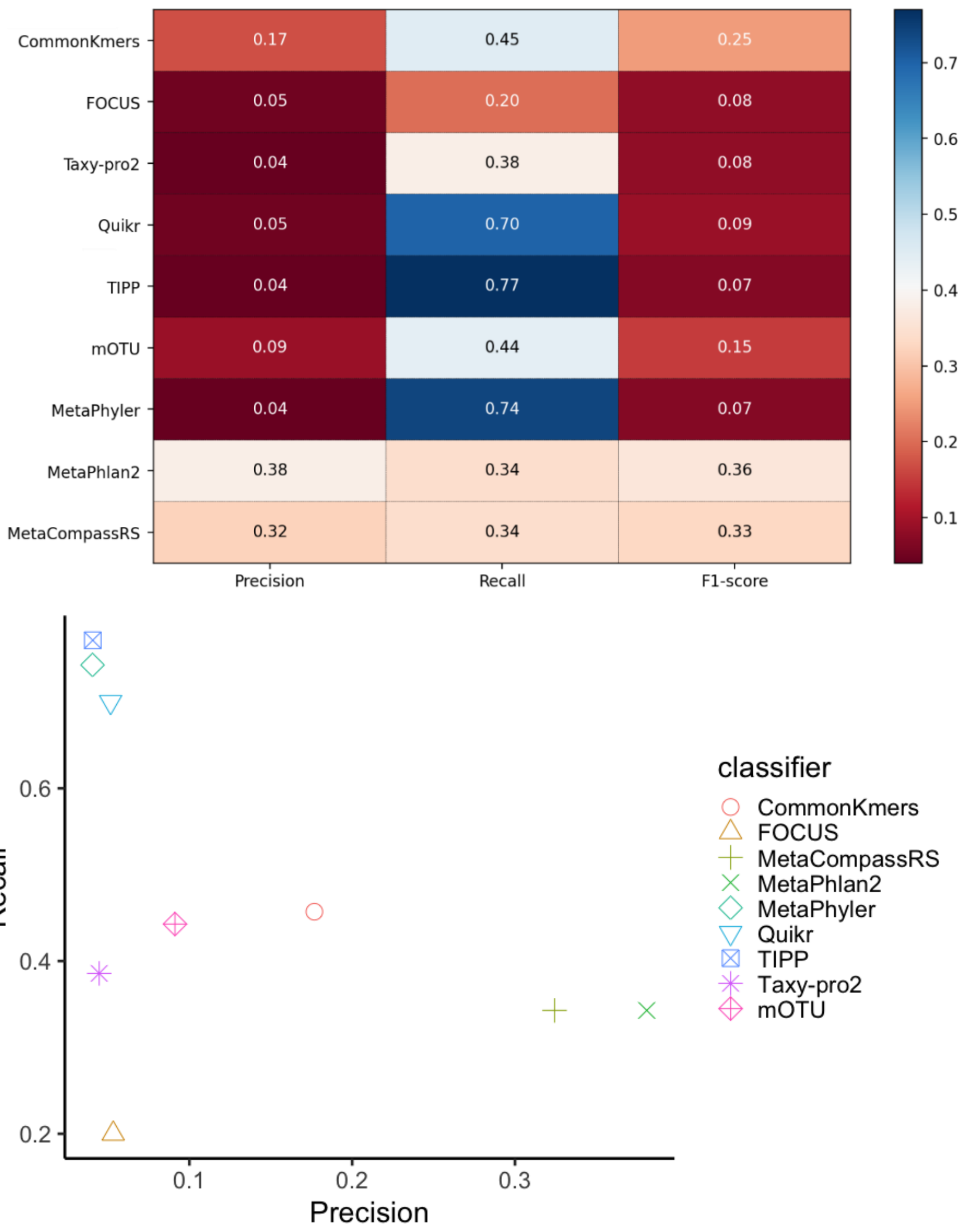


Figure 3.5. Species and strain level classification results on a medium complexity CAMI dataset.
(A) Heatmap showing Precision, Recall, and F1 score at species level. (B) Scatter plot of Precision (x-axis) versus Recall (y-axis) across all taxa.

3.4 Conclusion and discussion

We presented MetaCompassRS, a taxonomic classification method that outperforms previous methods in both species and strain level recall, while maintaining a strong balance between precision and recall. Achieving a high recall at the strain level is ideal if the end goal of the classification is to capture all relevant genomes from a database. MetaCompassRS achieves such results by combining alignment-based and k-mer based approach with a highly comprehensive reference database.

MetaCompassRS maintains a competitive running time and memory usage due to its marker gene clustering and k-mer pre-filtering steps. Clustering almost identical marker genes reduces the marker gene database size, and pre-filtering the reads dramatically reduces the query size. Furthermore, the use of Mash Screen adds genomic information beyond marker genes while keeping a low running time and memory usage.

Any of the methods presented, including ours, was capable to achieve full strain resolution. In fact, full strain resolution might not be possible by only analyzing short reads due to their short genomic context. Assembling the reads after the metagenomic classification can provide a more complete picture of the microbial community and further improve strain resolution [120–123]. In the following chapters, we describe how to use a set of reference genomes to perform reference-guided metagenomic assembly (Chapter 4 and 5).

Chapter 4: Reference-guided metagenomic assembly

4.1 Introduction

In the previous chapter, we described a method to infer the microbial genomes present in a metagenomic sample and select closely related genomes. In this chapter, we describe a metagenomic reference-guided assembly approach that uses a set of microbial genomes to reconstruct a metagenome.

Several *de novo* assembly methods have been applied to metagenomic data sets, but very little progress has been made on reference-guided assembly for metagenomics. Reference-guided assembly approaches are commonly used to assist the assembly of short reads when a closely related reference genome is available [19,23]. This process overcomes, in part, the challenge posed by repeats as the entire read provides information about its location in the genome.

Currently, thousands of bacterial genomes have been sequenced and finished [50,124]. These genomes are a great resource for performing comparative assembly of metagenomic sequences. However, to date, they have not been used for assembly, primarily due to the tremendous computational cost of aligning metagenomic reads to the entire reference collection of bacterial genomes. In this chapter, we describe our approach for reference-guided assembly of metagenomes.

4.2 Related work

The reference-guided metagenomic assembly process has two steps: (i) reads are aligned to a set of closely related reference genomes (read mapping step); and (ii) contigs are built from the relative locations of the reads in the reference genomes (consensus calling step). In this section we describe the read mapping step, consensus calling step and also present methods suitable for polishing metagenome assemblies.

4.2.1 Read mapping

A fundamental part of reference-guided assembly and many other bioinformatics analyses is the mapping of millions of short reads to reference genomes. A variety of algorithms and tools have been developed for read alignment[125,126]. Currently, the most widely used methods for read mapping can be divided into hash table based algorithms and Burrows-Wheeler Transform (BWT) [63] based algorithms.

Hash table methods can index either the genomes or the reads. Some methods for indexing genomes include GSNAP [127], Novoalign [128], mrFAST [129] , mrsFAST [130], and FANGS [131]. Methods for indexing reads include MAQ [132] and RMAP [133].

The most popular read mapping tools rely on the Burrows-Wheeler transform to reduce memory requirements [68,134–136]. Some BWT based read mapping tools are Bowtie [134], Bowtie2 [68], BWA [135], and SOAP2 [136]. Among them, the most widely used tool is Bowtie2, and improved version of Bowtie. Bowtie and Bowtie2 index the reference genome using a FM-index a [64] to maintain a small memory footprint. Bowtie was designed to find ungapped alignments, reporting end-to-end read alignments. Bowtie2 was extended to support local alignment— i.e. doesn’t require end-to-end read alignments.

For a single reference sequence, the read mapping problem has mostly been solved by indexing the reference into a data structure that supports efficient pattern search queries [125]. The read mappers described above [68,130,132,134–136] provide different trade-offs between speed and quality of the mapping [126,137].

Read mappers for single genomes are not suited for classification of metagenomic sequences, because they usually use a semi-global alignment model and assume near-identity of read sequences and reference genomes. Some metagenomic-specific mappers have been developed by adding filtration and normalization techniques to previously described single-genome mapping approaches [138,139]. Despite these efforts, metagenomic read mapping remain an open area of research.

The standard output format of read mappers is the *Sequence Alignment/Map* (SAM) format [140]. A SAM file has the information for each individual read mappings, including the read and reference genome identifiers, leftmost mapping position, mapping quality, and the CIGAR (Concise Idiosyncratic Gapped Alignment Report) string. A CIGAR string is a compress representation of an alignment that shows how the reads align to a reference genome. The CIGAR string has key information that can be used for consensus calling, such as matches, mismatches, gaps, deletion and insertion positions.

4.2.2 Consensus calling

The process of getting a consensus from the bases aligned to a genome is called consensus calling. In metagenomics, we need to find consensus sequences—equivalent to a *de novo* contigs—for each individual reference genome.

The most common approach for single genome consensus calling is using the *mpileup* and *BCFtools* utilities from the SAMtools package[141]. First, Mpileup summarize the base call information at each position in the reference genome into a "pileup of reads". The pileups of reads are generated by calculating the likelihoods of a base at each genomic position based on depth of coverage. Then, *BCFtools call* performs *variant calling* on each pileup of reads. The variant calling process involves identifying difference between the reads and the reference genome—such as single base changes, such as SNPs and indels, or larger scale structural variants. Finally, *BCFtools consensus* generates consensus calls from pileup of reads using the variant information.

There are multiple variant calling methods described in literature that could be used instead of BCFtools call, however, a broader discussion of such methods is beyond the scope of this dissertation.

4.2.3 Assembly polishing

Both de novo and reference-guided assemblies may have considerable base errors. Compared to de novo assembly, reference guided assembly has less space for misassembles. However, even small errors can degrade the performance of the reference-guided assembly process.

Assembly polishing tools can be used to correct base errors in draft assemblies. State of the art assembly polishing tools are GATK [142], Pilon [143], Racon [144], POLCA [145], and ntEdit [146].

Pilon and GATK are the most well-established polishers and can fix single bases changes, small and large indels, local misassemblies and can also fill gaps in the assembly. RACON and POLCA, part of the MaSuRca assembler [147], are more recent tools aimed to correct assemblies from long reads. Pilon, GATK and RACON work by mapping all reads against the assembly and then re-doing the consensus calling. This read mapping step, although accurate, makes the running time prohibitive to samples with high depth of coverage. POLCA is a little bit faster than Pilon and GATK by calling variants first and then only correcting the variants found in the assembly, thus avoiding remapping the reads.

A more recent tool, ntEDIT [148] is a bloom filter k-mer based approach that reduces time significatively compared to the previous described tools. First, ntEdit runs the tool ntHits [149], which removes erroneous k-mers and build a canonical representation of "coverage-thresholded k-mers" using a bloom filter. Then, ntEdit process contigs by interrogating the bloom filter for presence/absence. If a k-mer presence is confirmed, consecutive k-mers are skipped to avoid repetitive computation. If a k-mer or a part of a k-mer is absent from the reads, that part of the assembly is reported as a misassembly and the contig is polished. Warren et al. reported that ntEdit its faster and makes fewer mistakes than Pilon, its closest competitor.

4.3 Methods

4.3.1 Read mapping

The reference-guided metagenomic assembly approach involves mapping metagenomics reads to a set of genomes and then using their relative placement within each genome to guide the assembly of each reference. To achieve this task, we use Bowtie2 (parameters: --sam-nohead --sam-nosq --end-to-end --quiet --all -p 12). The output is filtered to keep alignments with the lowest edit distance for each read, allowing a read to be aligned in multiple locations (similar to the best-strata option of Bowtie 1).

4.3.2 Selecting a minimal reference set for consensus calling

In metagenomics, the relative placement of the reads within a mixture of genomes is more complex than in a single genome. This process is complicated by the fact that individual reads may map to multiple reference genomes, some of which are highly similar. Adequately dealing with this ambiguity is critical for effective assembly. If all read mappings are kept, allowing a read to be associated with multiple reference genomes, the resulting assembly will be redundant, reconstructing multiple copies of homologous genomic regions (Figure 4.1a). If for each read a random placement is selected from among the multiple equivalent matches, none of the related genomes may recruit enough reads to allow assembly, thereby leading to a fragmented reconstruction (Figure 4.1b). Assigning reads to genomes according to their estimated representation in the sample (determined, e.g., based on the depth of coverage), may bias the reconstruction towards the more divergent reference genomes, which may lead to an overall poorer reconstruction of the genomic regions shared across related genomes (Figure 4.1c). Here we propose a parsimony-driven approach: finding the minimal set of reference genomes that explains all read alignments (Figure 4.1d).

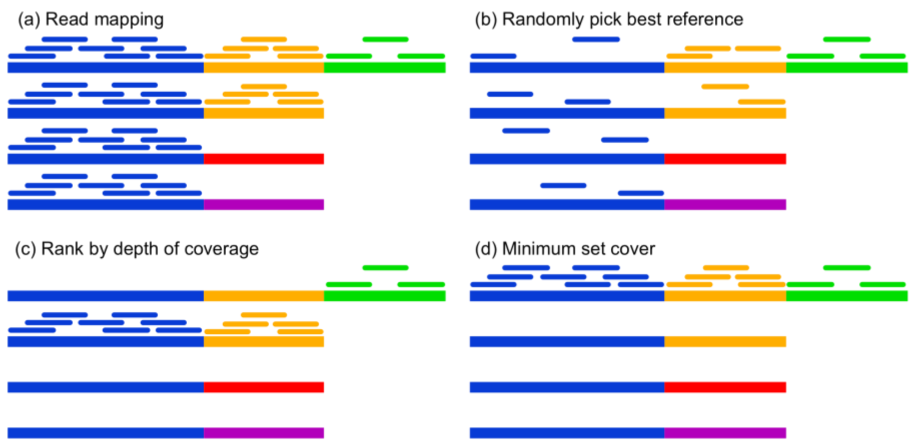

(a) Read mapping
(b) Randomly pick best reference
(c) Rank by depth of coverage
(d) Minimum set cover

Figure 4.1. Aligning read to reference genomes.
Shorter bars represent shotgun reads; longer bars represent reference genomes (4 genomes in this figure). Regions with the same color in the reference genomes represent homologous sequences. (a) All read mapping records. A read may be mapped to several reference genomes equally well, e.g., 5 yellow reads are mapped to both of the first two genomes. (b) For each read, if it is mapped to more than one reference genome, we randomly pick one. (c) A read is assigned to a reference with highest depth of coverage. (d) We pick the minimum number of reference genomes, to which all reads can be mapped.

4.3.2.1 Minimum set cover problem.

This parsimony-driven approach can be outlined as the set cover problem, an NP-hard optimization problem [151]. An instance (X, F) of the set-covering problem consists of a finite set X and a family F of subsets of X, such that every element of X belongs to at least one subset S in F:

$$X = \bigcup_{S \subseteq F} S$$

The problem is to find a minimum-size subset C ⊆F whose members cover all of X:

$$X = \bigcup_{S \subseteq C} S$$

We use a greedy approximation algorithm (see Algorithm 1), which iteratively picks the set of genomes using the greatest number of remaining unused reads. The algorithm works as follows. The set U has, at each stage, the set of remaining uncovered elements (uncovered reads). The set C has the cover being constructed (reference genomes that are picked). In the greedy decision-making step (line 4) a subset S of genomes is chosen that covers as many uncovered reads as possible with ties broken randomly. After S is selected, its elements are removed from U, and S is placed into C. When the algorithm ends, the set C has a subfamily of F that covers X with the greatest number of reads. It can be shown that this greedy algorithm is the best-possible polynomial time approximation algorithm for the set cover problem, under plausible complexity assumptions [150].

Algorithm 1: Greedy approximation for minimum set covering problem.
Input: a finite set X; a family F of subsets of X.
Output: a minimum-size subset $C \subseteq$F whose members cover all of X.

```
1: $U \leftarrow X$
2: $C \leftarrow \emptyset$
3: while $U \neq \emptyset$ do
4:     select an $S \in F$ that maximizes $|S \cap U|$
5:     $U \leftarrow U - S$
6:     $C \leftarrow C \cup \{S\}$
7: return C
```

4.3.3 Building contigs (consensus calling)

In order to apply the minimum set cover concept to metagenome assembly, we developed a consensus caller called *Buildcontig*. Buildcontig starts assembling the genome with the highest breadth of coverage first. Buildcontig evaluates the bases from the reads that are mapped to each position in the reference genomes and reports the genome with the highest depth of coverage as the consensus. Buildcontig can introduce indels up to a threshold. To introduce an indel, its depth of coverage should be higher than half of that of its neighbor nucleotides (Figure 4.2). Nucleotides from a reference sequence that don't match any base from the reads are discarded from the consensus sequence. This guarantee that the consensus sequence is not overly biased against the reference.

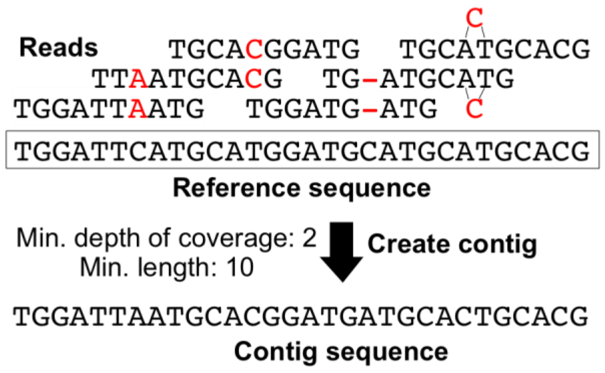


Figure 4.2. Creating contigs from reads that are mapped to reference genome using the majority rule.
Nucleotides that differ from the reference sequences are highlighted in red.

Buildcontig received two inputs: a SAM file with the read alignments, and a file with reference genomes. The minimum depth of coverage and minimum length for creating contigs can be specified through the program command-line options.

Finally, to remove reference-bias, we employ ntEdit (v1.18) to modify the consensus sequence to better represent the input data rather than the reference genome. In this step, contigs can be broken if the metagenomic sequence diverges from the reference sequence.

4.4 Results

We evaluated the performance of our reference-guided approach by using reads from a synthetic microbial community, which consists of a set of metagenomic reads from ground truth genomes [113]. After aligning the synthetic reads to the reference genomes, we generated consensus sequences (or assemblies) with Buildcontig under two settings. We first assembled the synthetic metagenomic skipping the minimum set cover algorithm and using all read mappings to guide the assembly (see Buildcontig_all results, Table 4.1). The aim of this experiment is to show that the performance of Buildcontig can be undermined by multi-mapped reads. Secondly, we ran Buildcontig including the minimum set cover algorithm (see Buildcontig results, Table 4.1). For both experiments, we set the minimum depth of coverage at 1-fold. We also assembled the reads using Samtools. For all experiments, we performed error correction with ntEdit (see "+ntEdit" results, Table 4.1).

When analyzing assembly statistics without reference genomes (Table 4.1), we observed that Buildcontig_all performed better than Buildcontig and Samtools in terms of contiguity (maximum contig size and size to 1 Mbp, Table 4.1) and completeness (total assembly size(bp) and # genes, Table 4.1). However, the higher contiguity and completeness of Buildcontig_all were hampered by the highest duplication and error rates (Table 4.2). Buildcontig produced the lowest duplication ratio (1.0), which indicates that Buildcontig was the only tool without a redundant assembly. Buildcontig_all had the highest duplication ratio (2.1), which indicates that Buildcontig_all generated a highly redundant assembly. In terms of assembly errors, Buildcontig produced the fewest misassemblies, mismatches, and indels.

For all assemblers, we observed a decrease in the total number of contigs shorter than 500bp after running ntEdit (see #contigs(<=0bp) and #contigs(<=500bp), Table 4.1), indicating that short erroneous contigs were effectively removed from the assemblies.

Table 4.1. Evaluation of performance on synthetic dataset without using reference genomes.
Tool indicates the consensus calling method: Buildcontig, Buildcontig_all and Samtools. Buildcontig indicates the default settings the minimum set coverage setting was used, and Buildcontig_all indicates that all read mapping were used (no minimum set coverage setting).“+ntEdit” indicates that ntEdit was run over the . # ctgs is the total number of assembled contigs reported by each assembler, Total assembly size is the total assembled length per assembler, Max ctg is the maximum contig length (broken at errors) for all assembled contigs, Size to 1 Mbp is the size of the largest contig C such that the sum of all contigs larger than C exceeds 1Mbp, #Genes is the number of fully reconstructed genes.

Tool	# Contigs (>=0bp)	# Contigs (>=500bp)	Total assembly size (bp)	Max contig size (bp)	Size to 1Mbp (Kbp)	# Genes
Buildcontig	54,207	13,727	187,980,023	7,057,101	7,057.10	179,428
Buildcontig+ntEdit	**52,954**	**13,727**	187,980,311	7,057,103	7,057.10	179,376
Buildcontig_all	821,792	139,413	383,249,716	**7,145,578**	**7,145.58**	**281,753**
Buildcontig_all+ntEdit	806,658	139,412	**383,252,412**	7,145,577	**7,145.58**	280,262
Samtools	815,862	63,570	377,636,570	7,057,100	7,057.10	242,306
Samtools+ntEdit	793,884	63,572	375,724,066	7,057,099	7,057.10	241,692

Table 4.2. Evaluation of performance on synthetic dataset using reference genomes.
Total aligned Length is the sum of the length of contigs aligned to the reference genomes, Total unaligned Length is the sum of the length of unaligned contigs, Genome fraction(%) is the total number of aligned bases in the references divided by genome size, and Duplication ratio(%) is the total number of aligned bases in the assembly divided by the total number of aligned bases in the reference. The last five statistics are reference-based errors reported by MetaQUAST.

Tool	Total aligned length	Fully unaligned length	Genome fraction (%)	Duplication ratio (%)	# Mismatches (/100 kbp)	# Indels (/100 kbp)	# Mis-assemblies (>1 Mbp)	# Local Misassm (<1 Mbp)	# Total Misassm
Buildcontig	185,773,672	1,364,126	89.366	**1.00**	137.74	**3.53**	**242**	**155**	**357**
Buildcontig+ntEdit	185,774,130	**1,363,100**	89.367	**1.00**	**136.91**	3.55	**242**	156	358
Buildcontig_all	377,259,399	3,834,531	**92.211**	2.10	395.55	15.36	1581	780	2361
Buildcontig_all+ntEdit	**377,267,056**	3,833,765	92.210	2.10	391.16	13.29	1586	781	2367
Samtools	250,165,735	3,197,557	89.548	1.45	138.33	5.21	512	**301**	813
Samtools+ntEdit	250,161,490	3,200,782	89.547	1.45	141.25	4.90	512	**301**	813

4.5 Conclusion and future directions

In this chapter, we first described concepts relevant to the reference-guided metagenomic assembly problem. We introduced the concept of read mapping in the context of both single genome and metagenomes, highlighting the most widely used indexing data structures for read mapping—Hash Tables and the Burrows-Wheeler Transform (BWT). Next, we briefly described the consensus calling process for single genomes. Lastly, we explained how assembly polishing can boost the correctness of the final assembly.

Secondly, we presented our reference-guided metagenomic assembly strategy. Our strategy starts by aligning a set of metagenomic reads to reference genomes using Bowtie2. Then, our consensus caller Buildcontig applies the minimum set cover algorithm to select a minimal reference set. After calling the consensus, we use the error correction tool ntEdit to polish the assembly and remove reference-bias. We showed that our reference-guided metagenome assembly strategy outperforms previous methods in terms of reference-free and reference-based assembly statistics. Finally, we showed that our assembly strategy generates non-redundant assemblies (low duplication ratio) while maintaining a high genome recovery.

Our reference guided assembly method could be further improved by adopting different read mapping and consensus calling strategies. As previously mentioned, Bowtie2 was not designed for metagenomics. As the number of available bacterial genomes increases, mapping reads with Bowie2 will get increasingly difficult. A more suitable mapping strategy for metagenomics would be a graph-based approach. In particular, the use of *de Bruijn* graphs for pan-genome analysis is well-suited for the tasks of compressing genomes and mapping reads, as described in previous work [152–155]. The most recent tools designed to align reads to *de Bruijn* graphs are Puffaligner, part of Pufferfish [156], and an extension of the pangenomic suit PanTools [157].

Our current assembly algorithm uses the minimum set cover algorithm, a winner-take-all strategy to minimize redundancy. When multiple closely related species co-exist, one will be well-assembled and the other species assemblies will be shattered into small contigs. We want to explore strategies for re-distributing multi-mapped reads across all aligned locations, resulting in a “resolve strains” mode. One strategy could be probabilistic assignment to pick the best strains given a species. This problem is similar to estimating differential abundance of transcript isoforms in RNA-sequencing data. Several methods for estimating differential abundance analysis employ the EM algorithm, which has also been successfully applied to metagenomics datasets [74].

Chapter 5: Hybrid reference-guided and *de novo* assembly of metagenomes

5.1 Introduction

In this chapter, we present MetaCompass, a metagenomic assembly approach that combines reference-guided and *de novo* assembly. MetaCompass selects reference genomes using MetaCompassRS (Chapter 3), and then follows the reference-guided assembly method described in Chapters 4 to reconstruct a metagenomic sample. Finally, to reconstruct genomes missing from our database, MetaCompass incorporates a *de novo* assembly step.

5.2 Method

MetaCompass is divided into five steps (Figure 5.1): (i) selecting reference genomes, (ii) reference guided assembly, (iii) removing reference bias, (iv) de novo assembly, and (v) combining reference-guided and de novo assembly.

First, we use the taxonomic classifier method MetaCompassRS to find the reference genomes most closely related to the input metagenomic sample. In the reference-guided assembly process reads are mapped to the selected genomes using Bowtie2, and then the consensus calling is performed with Buildcontig. After consensus calling, we rely on ntEdit to correct the contigs and avoid biasing the reconstruction towards the reference sequences. Finally, the reads that were not included in the reference-guided process outlined above are *de novo* assembled using

MEGAHIT [90] (v1.0.6). We chose MEGAHIT because it is the fastest and lowest-memory metagenomic assembler available, and it was shown to perform excellent in recovering the genomes of closely related strains [87]. Finally, we combined reference-guided contigs and assembly contigs. This hybrid approach allows the final assembly to capture microbes with closest reference genomes available and microbes that are missing from our reference database (such as novel variants).

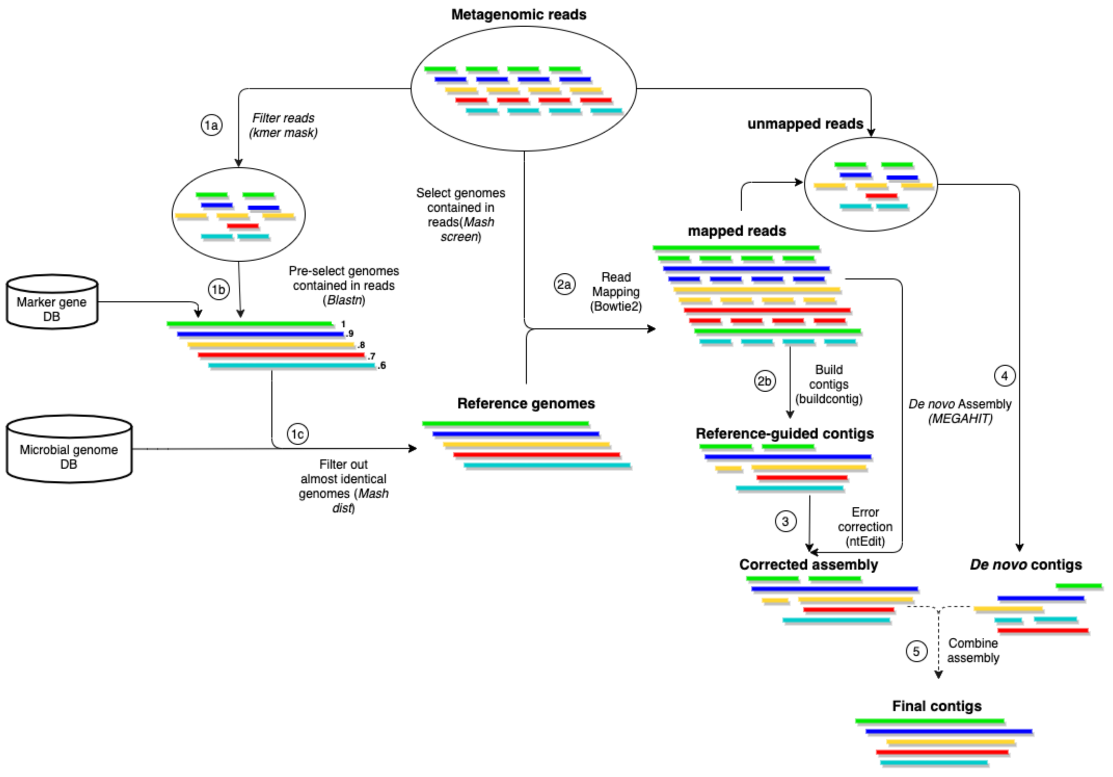


Figure 5.1. Overview of the MetaCompass pipeline.
Short colored lines represent reads and long lines genomes. Each color represents a different genome from a metagenomic sample. 1a-1c are part of the taxonomy classifier MetaCompassRS. 2a and 2b are part of the reference-guided assembly step.

5.2.1 Datasets used to evaluate metagenomic assemblies

5.2.1.1 Synthetic dataset

As described in Chapter 3, the synthetic microbial community published by Shakya et al. [113] contains 64 genomes. The set of known genomes for the synthetic dataset is available in the Supplementary Table 2 from Shakya *et al* [113]. The synthetic sample was downloaded from the NCBI Short Read Archive (SRA) database, (SRR606249) and has 54 bacterial and 10 archaeal strains from, representing a total of 61 species. Among these organisms, 55 had complete genome sequences in the NCBI RefSeq database (the database used by default by MetaCompass), and 9 were available only as a high-quality draft assembly.

5.2.1.2 HMP2 dataset

The Human Microbiome Project (HMP) is a collection of organisms living in association with the human body. The HMP has more than two thousand samples from different body sites sequenced and assembled. A list of all available HMP samples was obtained by from the HMP Data Analysis and Coordination Center (DACC) (www.hmpdacc.org). Some samples were excluded from the downloaded set because they were corrupt or extracted to a duplicate SRS identifier. Additional samples had no references recruited and were excluded from further analysis. A total of 2,294 samples had both an HMP2 assembly and a MetaCompass assembly and were used for the analysis.

5.2.2 Parameters used for metagenome assembly and metagenome assembly validation

5.2.2.1 Metagenomic assembly parameters

We compared MetaCompass with the *de novo* assemblers IDBA-UD (July 2016) [88], MEGAHIT (v1.0.6) [90], and MetaSPAdes (v3.9.0) [9]. IDBA-UD requires a single fasta file that was generated using the IDBA 'fq2fa --merge --filter' command. MEGAHIT was run using the options '--presets meta-sensitive --min-count 3 --min-contig-len 300 -t 12'. MetaSPAdes was run using the options '--meta -t 12', then all contigs shorter than 300nt and with less than 3X coverage were removed. IDBA-UD was run using the options '--min_count 3 --min_contig 100 --mink 20 --maxk 100 --num_threads 12'. MetaCompass was run using the options -m [1,2,3] -g 100 -t 16' on the synthetic dataset and '-m 3 -g 100 -t 16' on the HMP2 samples.

5.2.2.2 Metagenomic assembly validation parameters

We used MetaQUAST, a reference-based metagenomic assembly validation method that finds misassemblies and structural variants in an assembly relative to reference genomes. The command used to run MetaQUAST on the Shakya et al. synthetic dataset was: 'metaquast.py -R ./shakya_references --fragmented --gene-finding'.

5.3 Results

Although real metagenomic reads are the most proper test of performance, it is not possible to assess accuracy from such data because true species in metagenomic datasets are unknown. We first evaluated the performance of MetaCompass using synthetic datasets. Since the true genome sequences are known, these data are ideal as they allow us to fully quantify the quality of the genomic reconstruction.

Additionally, we generated improved assemblies of almost the entire dataset generated by the Human Microbiome Project (2,294 distinct samples in total), and use the results to characterize the relative advantages and limitations of *de novo* and reference-guided assembly approaches, thereby providing guidance on analytical strategies for characterizing the human-associated microbiota.

5.3.1 Evaluation of performance on synthetic metagenomic dataset

We valuated MetaCompass by assembling a synthetic microbial community [113]. We assembled this synthetic metagenomic with MetaCompass under two settings. We first assembled the synthetic metagenome skipping the reference selection step and using the exact genomes present in the sample as a reference to guide the assembly. The aim of this experiment is to show that the performance of MetaCompass can be excellent if the reference collection has genomes highly similar to those in the metagenomic sample being assembled. Secondly, we ran the complete MetaCompass pipeline including both the reference selection step and reference-guided assembly. We set the minimum depth of coverage in MetaCompass at 1-fold and 2-fold for both experiments.

The assembly results of our fist experiment (Table 5.1, see MetaCompass 1X and 2X) can be considered an approximate upper bound on the performance of any assembly tool, as in this case almost all of the genomes recruited (90%) were exactly those from which the metagenomic reads were obtained. We compared the performance of the two rans of MetaCompass with that of three widely used *de novo* assemblers: IDBA-UD, MEGAHIT, and metaSPAdes. Compared with these assemblers, MetaCompass achieved higher genome recovery (Table 5.1, Figure 5.2) and produced significantly larger and more accurate contigs (Table 5.1). When we decreased the MetaCompass minimum coverage threshold from 2-fold to 1-fold, we observed gains in maximum contig size and total aligned length, while retaining a similar error profile.

Table 5.1. Evaluation of performance on synthetic dataset.
MetaCompass (X) indicates the minimum coverage setting (1X or 2X), and MetaCompass.nr indicates all 64 reference genomes comprising the Shakya et al. dataset were removed from the database. # ctgs is the total number of assembled contigs reported by each assembler, Max ctg is the maximum contig length for all assembled contigs, Gen. Rec. (%) is the median percentage of each of the synthetic genomes that is recovered, Complete Marker Genes (median) is the median number of fully reconstructed marker genes, Total aligned length is the sum of the length of contigs aligned to the reference genomes, Total unaligned length is the sum of the length of unaligned contigs. Mismatches, Indels, and Misassemblies (Misassm) are error statistics generated with MetaQUAST.

Assembler	**#contigs**	**Max Ctg**	**Gen. Rec. (%)**	**Complete marker genes (median)**	**Total aligned length**	**Total unaligned length**	**Mismatches (/100 kbp)**	**Indels (/100 kbp)**	**Misassm (>1 Mbp)**	**Misassm (<1 Mbp)**	**Total Misassm (<1 Mbp)**
MetaCompass (1X)	**18,766**	**7,057,109**	**100**	**40**	**198,113,036**	**6,340,278**	**61.9**	1.9	**0.8**	1.1	**1.9**
MetaCompass (2X)	23,648	5,841,107	100	**40**	195,836,655	6,198,040	63.1	**1.8**	0.9	1.1	2.0
MetaCompass.nr (2X)	42,852	1,151,857	98	**40**	195,225,556	6,338,183	89.9	3.6	3.3	1.6	4.9
IDBA-UD	22,355	991,792	98	39	186,777,879	6,186,424	98.6	3.5	5.3	**1.0**	6.3
MEGAHIT	35,351	1,151,857	99	**40**	195,334,581	6,263,018	66.5	2.8	1.5	**1.0**	2.5
metaSPAdes	21,424	1,438,235	99	**40**	192,795,050	6,208,276	97.1	3.7	1.3	**1.0**	2.3

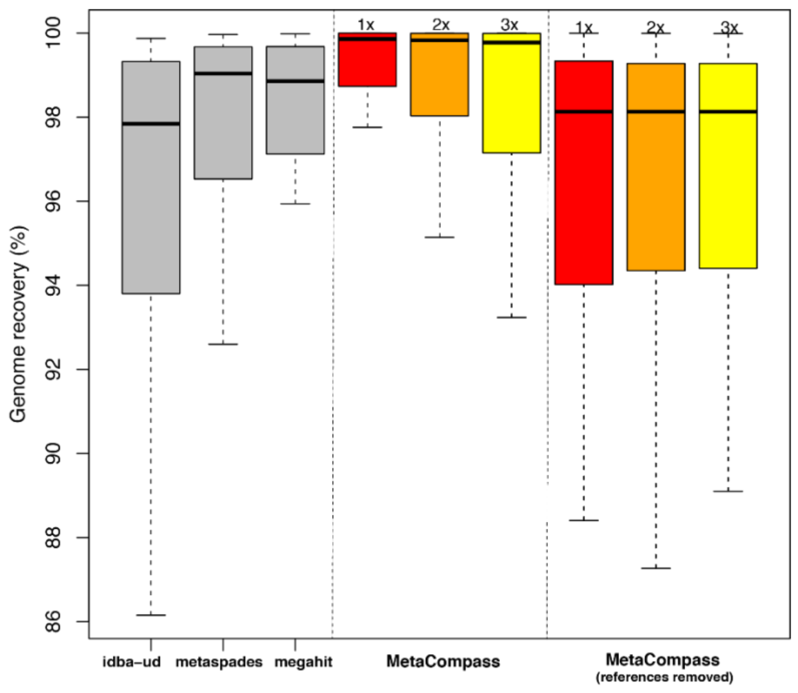


Figure 5.2. Genome recovery percentages in synthetic metagenome (MetaCompass versus *de novo* assembly).
Box plots represent distribution of genome recovery percentages (for the 64 genomes present in the synthetic metagenome). x-axis indicates the assembly method, either IDBA-UD, metaSPAdes, MEGAHIT, or MetaCompass. MetaCompass was run both with the reference genomes present in the database (recruited as described in the methods) and without the truth reference genomes in the database (they were individually removed). y-axis indicates the genome recovery percentage, 0% indicates the genome was unassembled, whereas 100% indicates the genome was fully assembled.

5.3.1.1 References removed from database

To provide a better idea of how MetaCompass would perform in a worst-case scenario, we removed from the database the genomes represented in the synthetic community (Appendix A), thereby forcing MetaCompass to recruit near-neighbor reference genomes, when available. (see 'MetaCompass.nr' row, Table 5.1). In this case, we found that MetaCompass still performed almost as well as *de novo* assemblers while making far fewer errors than if it simply mimicked the reference genome. Median genome recovery for MetaCompass is just 1% less than that of *de novo* assemblers. The accuracy of the reconstruction, as measured by mismatch and indel rates, is lower than that of IDBA-UD and metaSPAdes (Table 5.1, MetaCompass.nr (2x)), while moderately higher than MEGAHIT.

The number of misassemblies and local misassemblies per 1 Mbp of assembled sequence (as reported by MetaQUAST [158]) increased from 2.0 to 4.9 when reducing the coverage threshold to 1. To put this increase into context, we measured the total number of possible errors by evaluating the "accuracy" of the near-neighbor reference genomes recruited by MetaCompass with respect to the correct reference sequence (Figure 4 see hashed blue bar). This allows us to capture the real differences between the recruited reference genomes and the actual genome represented in the synthetic dataset [113], providing an upper bound on the number of errors MetaCompass could make if it simply recapitulated the sequence of the selected reference genomes. As seen in Figure 5.3, the MetaCompass assembly is much closer to the correct genome than the reference sequence.

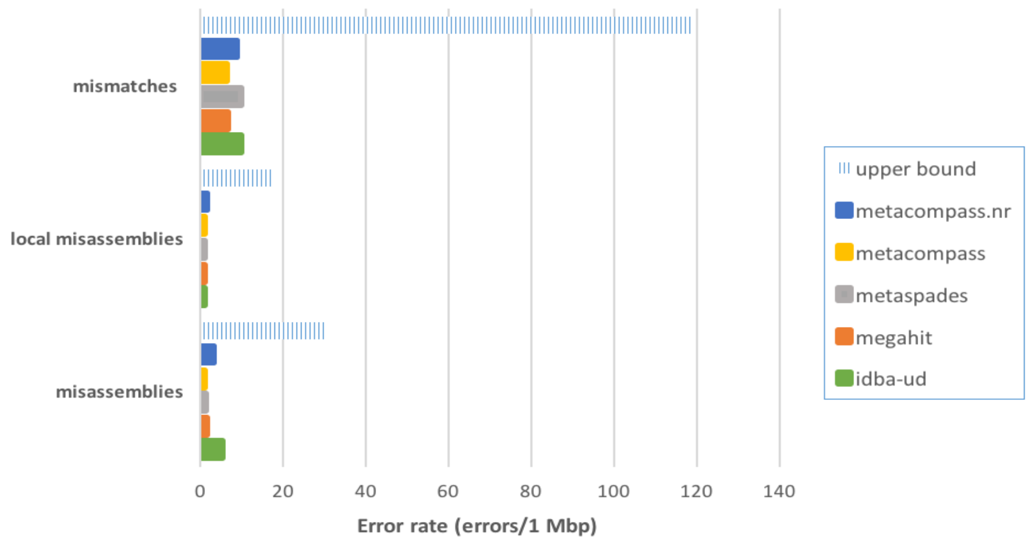


Figure 5.3. Error profile on synthetic dataset.
The hashed blue bar represents the difference between the second-best reference genome (recruited by MetaCompass) and the true genome represented in the sample. This bar can be viewed as an upper bound on the errors metacompass.nr could make if it simply reconstructed the reference genome. Mismatches are the number of bases in a contig that differ from the reference genome. Misassemblies include large-scale (left flanking region aligns >1 kbp away from right flanking region) relocations, interspecies relocations, translocations, and inversions. Local misassemblies include small-scale (left flanking region aligns <=1 kbp away from right flanking region) translocations and inversions. All errors are normalized to represent rates per 1 Mbp.

5.3.1.2 Evaluation of performance on down sampled synthetic metagenomic dataset

To evaluate the ability of MetaCompass to assemble low-coverage genomes, we downsampled the synthetic dataset to just 5 million paired-end reads, or 10% of the original data set. After downsampling, the average coverage was reduced to approximately 3-fold. The results (Table 5.2, Figure 5.4) highlight that MetaCompass can recover a median of 90% of each of the 64 genomes in the sample. While metaSPAdes comes in second place and is able to recover 80% (median recovery), it does so at the cost of four times higher misassembly rate (Table 5.2). The two remaining methods, MEGAHIT and IDBA-UD leave a quarter to a half of the genomes unassembled and also produce higher misassembly rates assemblers: IDBA-UD, MEGAHIT, and metaSPAdes. Compared with these assemblers, MetaCompass achieved higher genome recovery (Table 5.2, Figure 5.2) and produced significantly larger and more accurate contigs (Table 5.2). When we decreased the MetaCompass minimum coverage threshold from 2-fold to 1-fold, we observed gains in maximum contig size and total aligned length, while retaining a similar error profile.

Table 5.2. Evaluation of performance on down-sampled synthetic dataset.
The synthetic dataset was down-sampled to only contain 10% of the total reads.
# ctgs is the total number of assembled contigs reported by each assembler, Max ctg is the maximum contig length for all assembled contigs, Median Genome Recovery (%) is the median percentage of each of the synthetic genomes that is recovered, Complete Marker Genes (median) is the median number of fully reconstructed marker genes, Total aligned Length is the sum of the length of contigs aligned to the reference genomes, Total unaligned Length is the sum of the length of unaligned contigs. Mismatches, Indels, and Misassemblies (Misassm) are error statistics generated with MetaQUAST.

Assembler	**#contigs**	**Max Ctg**	**Median Genome Recovery (%)**	**Complete Marker Genes (median)**	**Total aligned length**	**Total unaligned length**	**Mismatches (/100kbp)**	**Indels (/100kbp)**	**Misassm (>1 kbp)**	**Misassm (<1 kbp)**	**Total Misassm (<1 kbp)**
MetaCompass	71457	**962,929**	**90%**	**22**	**134,008,055**	3,009,931	**117.6**	**1.9**	**112**	**33**	**145**
IDBA-UD	43973	120159	45%	6	75,970,693	**1,564,008**	175.0	5.3	3447	93	3540
MEGAHIT	62842	209,706	76%	15	105,665,678	2,774,432	128.0	4.1	772	122	894
metaSPAdes	67138	287,554	80%	16	111,636,826	3,154,199	133.0	4.3	470	115	585

5.3.1.3 Computational performance

When dealing with large-scale data sets, the total required memory and running time are important factors in determining the applicability of a computational tool. We first evaluated the running time performance of MetaCompass on a Linux 12-core server node with 80 GB of memory using the Shakya et al. synthetic dataset. The wall clock running time on this synthetic dataset for MetaCompass is comparable to the evaluated *de novo* assemblers and sometimes lower (Table 5.4). MetaCompass and Megahit were the only approaches that required less than 16GB of RAM on a 100 million read dataset, highlighting the scalability of this methods to large datasets.

Table 5.3. Running time for assemblers on Shakya et al. sample.
We evaluated the running time performance of MetaCompass and three *de novo* assemblers for the full Shakya et al. sample (100 million paired-end reads) and a 10% of the original data set (5 million paired-end reads). The full dataset was run using 80 GB of memory and 12 CPUs and the down-sampled dataset using 36GB of memory and 4 CPUs.

	Shakya et al.		**Downsampled Shakya et al.**	
Assembler	**Time (hh:mm)**	**Memory (Gb)**	**Time (mm:ss)**	**Memory (Gb)**
MetaCompass	3:53	19.82	3:35	10.34
IDBA-UD	3:53	16.78	2:42	7.39
MEGAHIT	2:26	8.61	2:03	2.35
metaSPAdes	6:02	28.07	8:25	19.63

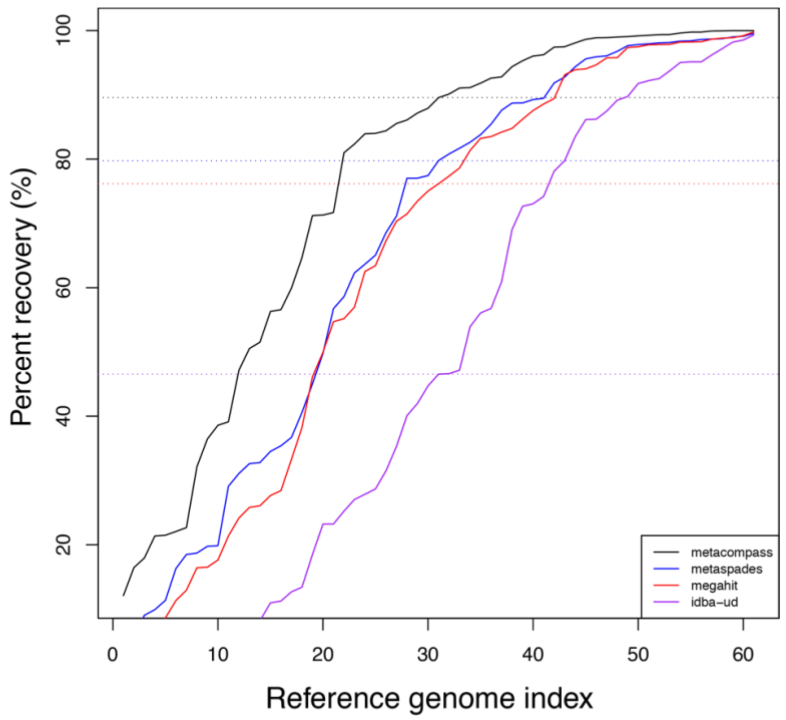


Figure 5.4. MetaCompass performance on low coverage dataset.
Results obtained by down-sampling the Shakya et al. synthetic genome to just 10% of the original set of reads. The 64 genomes present in the sample are ordered per assembler by percent recovery, from lowest to highest. The y-axis indicates how much of the n-th reference was covered by correctly assembled contigs (can range from 0% to 100%). The colored dashed lines indicate the median percent recovery for each assembler.

5.3.2 Evaluation of performance on Human Microbiome Project (HMP2)

5.3.2.1 Reassembly of the data generated by the Human Microbiome Project (HMP2)

To further explore the benefits and limits of comparative approaches for metagenomic assembly, we re-analyzed with MetaCompass 2,294 metagenomic samples from the HMP Project. These samples cover 15 body sites from four broad regions of the human body: oral, skin, stool, and vaginal. We compared the assemblies produced by MetaCompass with the assemblies reported by the HMP project [159]. Across all samples, on average, MetaCompass outperforms the HMP2 *de novo* approach, leading to an overall better assembly of the original data (Table 5.4, Figure 10).

The relative performance of the MetaCompass and HMP2 assemblies varied across body sites due to the specific characteristics of the microbial communities being reconstructed. While MetaCompass generates more assembled sequence and complete marker genes across all body sites, the maximum contig size and size at 1 Mbp metrics vary per body site. In oral and stool samples (Figure 5.5), MetaCompass outperforms *de novo* assembly for all metrics. In skin and vaginal samples (Figure 5.5), the *de novo* (HMP2) assemblies have better contiguity statistics but MetaCompass assembles more complete marker genes. To gain further insight into these results we calculated the average nucleotide identity of the *de novo* assembled contigs to the recruited reference genomes for each body site. In all body sites, except for oral, the assembled contigs had 99% average nucleotide identity to the reference genomes. In the oral samples, the most distant reference genomes had only 97% identity to the assembled contigs.

To further explore the drop-in contiguity in skin and vaginal samples, we focused on just the contigs that mapped to bacterial genomes contained in the reference database, allowing for a direct comparison between MetaCompass and *de novo* contigs. The results in Table 5.4 show that for this set of contigs, MetaCompass outperforms the *de novo* approach for the vaginal samples. However, the *de novo* HMP2 assembly of the skin sample is still better in terms of complete genes recovered, but equivalent to MetaCompass with respect to complete marker genes recovered (a measure of assembly completeness).

Table 5.4. Re-assembly of 2,294 samples generated in the Human Microbiome Project. The results are aggregated by body site. # indicates the total reads per sample, Avg cvg per sample (X) is the mean estimate read coverage calculated based on the *de novo* assembly of each sample and body site, Shannon Entropy (median) is the Shannon diversity value per body site as reported in Li *et al.* [160]. The rows labeled MC contain results obtained with MetaCompass. The rows labeled HMP2 show the statistics for contigs from the production HMP2 assembly. Total Size (Mbp) is the total assembly size for each method, Max ctg size (kbp) is the size of the largest contig, Median Size 1Mbp (kbp) represents the median size of the largest contig C such that the sum of all contigs larger than C exceeds 1Mbp. Median Complete Genes represents the median number of complete genes per sample. Median Marker Genes indicates the median number of complete marker genes per sample.

HMP2 body site	Num of samples	Avg cvg per sample	Shanon Entropy (median)	Asm	Total size (Mbp)	Max ctg size (kbp)	Median size 1 Mbp (kbp)	Median complete genes	Median marker genes
Oral	1259	20.0	2.4	HMP2	106,693	546.4	70.8	54,1	762
		±8.1		**MC**	**135,586**	**892.3**	**95.8**	**63,144**	**915**
Skin	291	17.4	1.5	HMP2	2,944	890.7	**36.5**	4,654	78
		±4.7		**MC**	**3,782**	**2,159.3**	15.1	**5,01**	**79**
Stool	524	18.4	**2.6**	HMP2	56,573	592.8	109.1	84,193	847
		±4.9		**MC**	**66,838**	**3,301.0**	**230.9**	**94,297**	**1,043**
Vagina	220	7.8	0.2	HMP2	1,179	465.8	**28.7**	2,539	45
		±4.5		**MC**	**1,458**	**558.0**	16.1	**2,934**	**60**
All	2294	18.2	1.9	HMP2	184,518	890.7	79.0	48,836	633
		± 5.6		**MC**	**232,161**	**3,301.0**	**114.6**	**57,639**	**764**

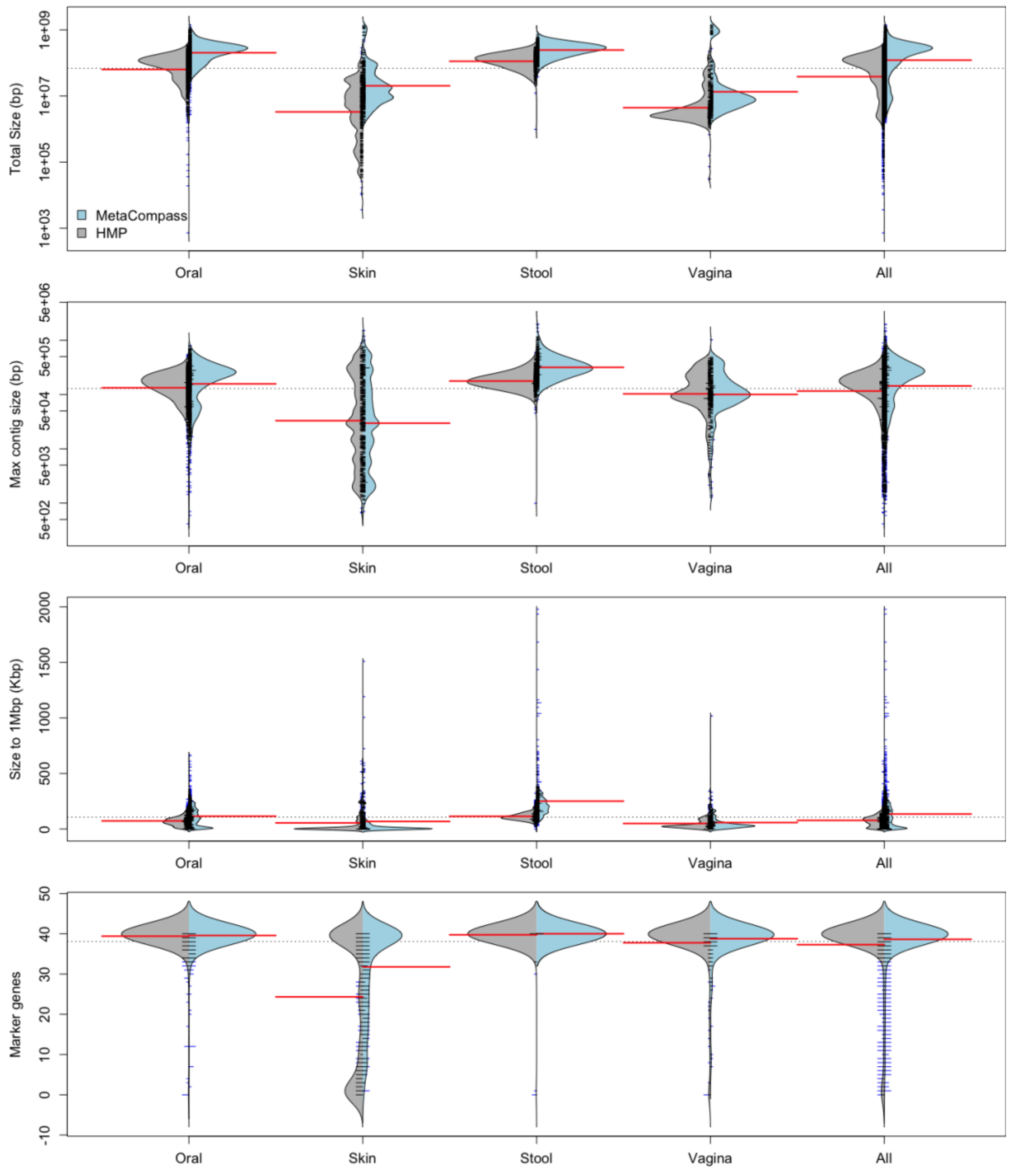


Figure 5.5. Comparative assembly of 2,294 metagenomic samples from the HMP2 Project.
The bean plots represent the distribution of assembly contiguity and completeness statistics across all samples within the data. The x axis organizes the data by assembly and body site. The y-axis indicates the statistic used to evaluate the assembly contiguity or completeness. The top panel shows total assembly size, the second panel shows maximum contig size, the third panel shows the size of the contig at 1 Mbp, and the bottom panel shows the complete marker genes assembled per sample.

5.3.2.2 Comparing reference-guided to de novo assembly on low-coverage HMP2 samples

To assess the ability of MetaCompass to assemble low-abundance organisms, we focused on all skin HMP2 samples. The skin samples had the second lowest average number of reads while still containing reasonable diversity and richness, as reported in Table 5.4. We removed the contigs assembled via *de novo* assembly from the MetaCompass output, collected the reference genomes that were used, mapped the HMP2 contigs to these reference genomes, and then evaluated the number of complete genes and complete marker genes. Compared to the HMP2 assembly, reference-guided assembly of these low coverage samples is able to reconstruct approximately 10% more marker genes (4,423 versus 3,915) than the *de novo* approach, roughly equating to 10 additional complete bacterial genomes.

We next searched for microbes that were present in the skin samples at relatively low coverage and explored the differences between the reconstructions generated by the HMP2 project and MetaCompass. Specifically, we identified the low coverage assembly of a *Propionibacterium acnes* genome reconstructed by both MetaCompass and the HMP in sample SRS057083**.** The HMP2 assembly covers less than 40% of the closest reference genome (NC_016516.1, *Propionibacterium acnes* TypeIA2 P.acn33), while the MetaCompass assembly covers more than 90% of the same genome.

5.3.2.3 Comparing reference-guided to de novo assembly on high coverage HMP2 samples

To assess the ability of MetaCompass to assemble high-abundance organisms, we focused on all stool HMP2 samples. The stool assemblies had the longest maximum contig and median size to 1Mbp, as reported in Table 5.4. We searched for microbes with the best assembly among all stool samples (NZ_CP012801, *Bacteroides cellulosilyticus* WH2, HMP2 sample SRS143342), and explored the differences between the reconstructions generated by *de novo* assemblers and MetaCompass.

We next collected the reference genomes that were used by MetaCompass and mapped both *de novo* and reference-guided assemblies to these reference genomes. The *Bacteroides cellulosilyticus* WH2 genome was recovered by all assemblies with more than 70% of genome recovery. As show in Figure 5.6, all tools reconstructed a fragmented assembly towards the beginning of the genome, were more sequencing errors are usually found. Overall, after the initial fragmented contigs, MetaCompass assembled ten long contigs with length ranging from 0.5 to 2.28MBp.

The longest MetaCompass contig covers 0.32% of the *Bacteroides cellulosilyticus* WH2 genome (Figure 5.7) and aligned almost perfectly to the reference genome (2 mismatches). In contrast, MetaSPAdes, Megahit and IDBA-UD reconstructed an extremely fragmented assembly with many misassembled contigs. To further investigate how the reads were distributed across both the reference genome and contigs, we mapped both reads and contigs to the genome with Bowtie2 and Minimap2 [161], respectively (Figure 5.8). Although the read mapping visualization shows a relatively even depth of coverage, *de novo* assemblers were unable to reconstruct a contiguous assembly. Conversely, MetaCompass reconstructed the full segment of the genome. MEGAHIT was the second-best assembler, almost reconstructing the full segment.

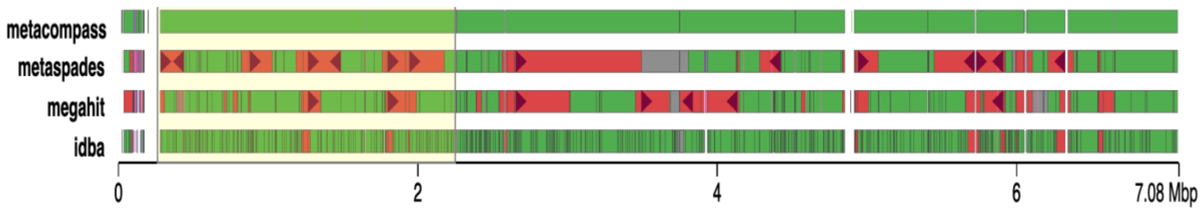


Figure 5.6. Icarus view of metagenomic assembly of the stool sample SRS143342 from the HMP2 Project.
The contigs largest than 1000bp from MetaCompass, MetaSPAdes, Megahit, and IDBA were aligned to the *Bacteroides cellulosilyticus* WH2 genome (NZ_CP012801, 7084828 bp). Colors indicate how well the contigs aligned to the reference. Green represent correct contigs, red misassembled contigs, purple ambiguously mapped contigs, and gray unaligned contigs.

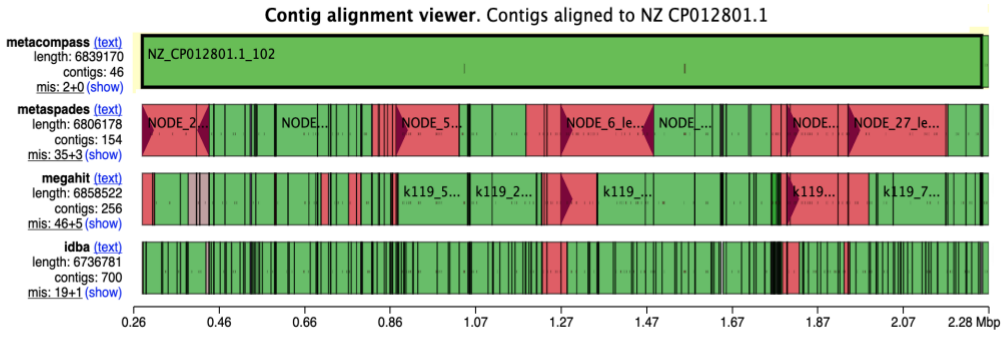


Figure 5.7. Longest contig from *Bacteroides cellulosilyticus* strain WH2 chromosome genome assembly (accession:NZ_CP012801.1, length: 7084828 bp).
The length of contig NCP012801.1_102 is almost 2.28 Mbp, covering 0.32% of the complete genome.

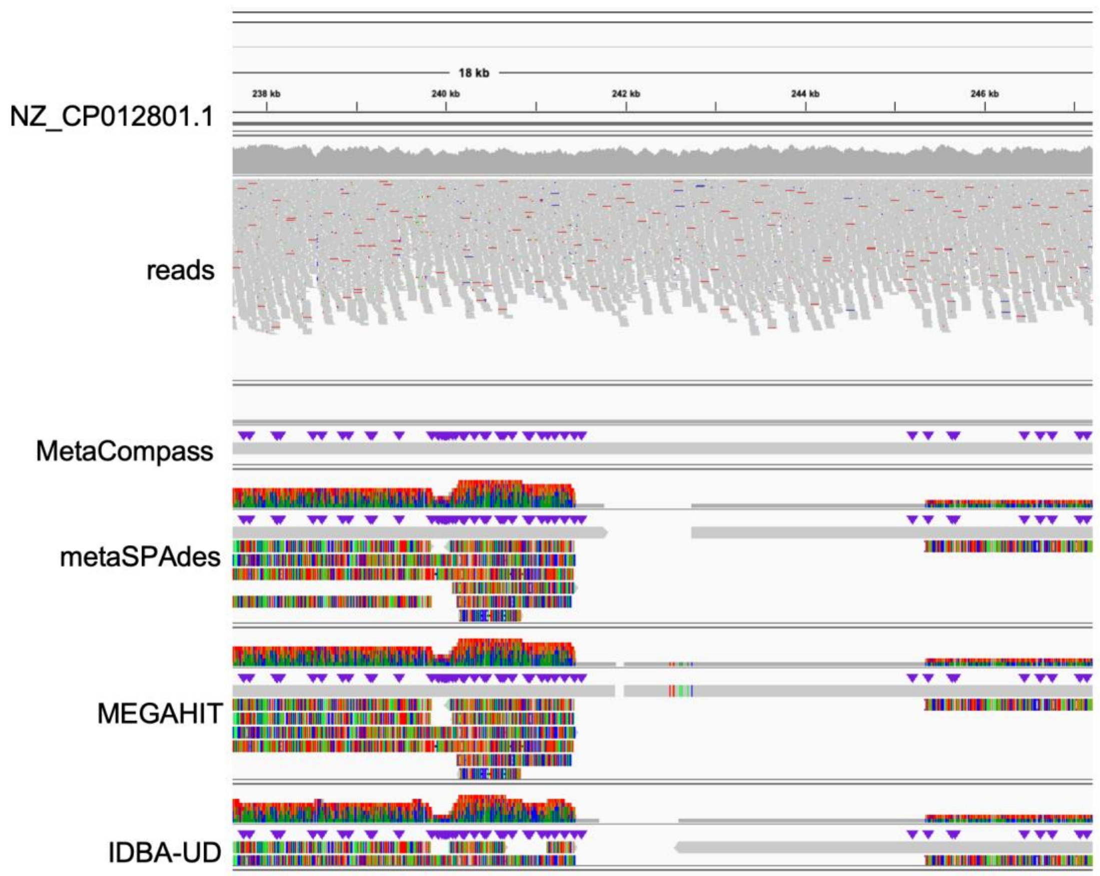


Figure 5.8. IGV visualization of read and contig mapped against a segment of the *Bacteroides cellulosilyticus* WH2 genome (accession:NZ_CP012801.1, length: 7084828 bp).
MetaCompass reconstructed the full segment of the genome. MEGAHIT almost reconstructed the full segment. MetaSPAdes and IDBA-UD had the biggest assembly gap.

5.4 Conclusion and discussion

We have described MetaCompass, a comparative metagenome assembly method that relies on an indexing strategy to construct sample-specific reference collections. We show that comparative and *de novo* assemblies provide complementary strengths, and that combining both approaches effectively improves the overall assembly, providing a consistent increase in the quality of the assembly. Even when distant reference genomes are recruited, we remain competitive with *de novo* genome assembly methods. We accomplish this via two critical steps. First, we avoid reference bias by constructing the consensus sequence from the reads within the sample, using the reference genome as just a guide, and we break the assembly where the reads indicate a structural disagreement with the reference. Second, we use unmapped reads in a *de novo* assembly process to reconstruct the sections of the metagenomic sample that are not similar to known reference genomes. We have shown MetaCompass to be particularly effective in the assembly of low coverage or rare microbes, a setting in which *de novo* assembly approaches simply cannot be used with good results. Improved assembly of low-abundance, rare microbes from existing datasets has the potential to provide additional resolution in complex microbial communities or clinical samples where the host DNA comprises a large fraction of the data. Finally, we have shown that in high-abundance genomes, MetaCompass is more effective that *de novo* in generating complete and contiguous assemblies.

The benefit of comparative assembly is highly dependent on the reference genomes available in the database provided to MetaCompass. While MetaCompass can effectively use reference genomes that are distantly related to the genomes being assembled, the quality of the reconstruction is lower than can be achieved with closely related reference sequences. Many bacteria found in the human microbiota are difficult to culture (e.g., the many anaerobes inhabiting the human intestinal tract) and are, therefore, under-represented in public databases. Despite this fact, MetaCompass was able to improve, often significantly, upon the assembly of the data generated by the Human Microbiome Project 2. However, the contiguity of MetaCompass on skin samples was not improved upon the assemblies generated by HMP2. This could be due to the structural genome dynamics of bacterial defense systems commonly found in the skin microbe [162–164]. Future work will focus on elucidating the effect of each of these factors via assembly graph-based approaches. In addition, as the number of genomes in public databases is increasing, comparative approaches such as ours will be increasingly valuable for reconstructing near-complete genome sequences from metagenomic data.

Chapter 6: Conclusion

Metagenomic assembly, the process of reconstructing large genomic segments from metagenomic reads, is a formidable computational challenge. Even for single organisms, the assembly of genome sequences from next-generation sequencing (NGS) reads is a complex task, primarily due to ambiguities in the reconstruction that are caused by genomic repeats. In addition, metagenomic assemblers must be tolerant of non-uniform representation of genomes in a sample as well as of the genomic variants between the sequences of closely related organisms. Despite advances in metagenomic assembly algorithms over the past years, the computational difficulty of the assembly process remains high and the quality of the resulting assemblies requires improvement. The reference-guided assembly paradigm has been shown to outperform the *de novo* assembly paradigm under certain settings, yet, the former has not been extensively explored.

In this dissertation, we designed methods to address the reference-guided metagenomic assembly problem. This problem consists of two subproblems: selecting closely related genomes to guide the assembly and reconstructing each genome individually. To address the first subproblem, we developed MetaCompassRS, a taxonomy classification approach that is able to retrieve the closest reference genomes available in a database that are contained in a metagenomic sample. We showed that MetaCompassRS achieves higher recall than state of the art taxonomy classification tools, while maintaining a competitive running time.

The second subproblem is further subdivided into read mapping and consensus calling. We used Bowtie2—the most widely used short read mapper—for the former task and developed an approached inspired on the minimum set cover problem for the latter task. We implemented the minimum set cover algorithm in our tool Buildcontig and showed its efficiency and effectivity in reducing the redundancy of metagenome assemblies.

Finally, we developed MetaCompass, a metagenomic assembly pipeline that encompass MetaCompassRS, Buildcontig and *de novo* assembly to reconstruct a metagenomic sample. When combined with de novo assembly approaches, we showed that reference-guided assembly is able to generate more complete assemblies than the ones obtained by the *de novo* assembly alone. We also showed that MetaCompass performs better than the state of the art methods in real world datasets—such as the ones gather by the HMP.

We believe that reference-guided metagenomic assembly approaches, and with MetaCompass being one of the first ones reported in the literature, will increasingly replace the more computationally expensive and error-prone *de novo* assembly approaches as the collection of available reference genome sequences increases. Furthermore, reference-guided assembly provides new opportunities for the development of both clinical and computational applications. Clinical applications are a particularly relevant application domain for reference-guided approaches because the vast majority of publicly available genome sequences comprises human pathogens. Computational methods capable of handling a large amount of metagenomic sequencing data are an active area of research. One of the most promising strategies to handle metagenomics reference collections is using pangenome graphs, which we plan to further explore in the future.

Appendices

Appendix A. References removed from database used by MetaCompass.

Species name
Acidobacterium capsulatum
Aciduliprofundum boonei
Akkermansia muciniphila
Archaeoglobus fulgidus
Bacteroides thetaiotaomicron
Bacteroides vulgatus
Bordetella bronchiseptica
Burkholderia xenovorans LB400
Caldicellulosiruptor bescii
Caldicellulosiruptor saccharolyticus
Chlorobium limicola
Chlorobium phaeobacteroides
Chlorobium phaeovibrioides
Chlorobium tepidum
Chloroflexus aurantiacus J-10-fl
Clostridium thermocellum
Deinococcus radiodurans R1
Desulfovibrio piger
Desulfovibrio vulgaris DP4
Dictyoglomus turgidum
Enterococcus faecalis
Fusobacterium nucleatum nucleatum
Gemmatimonas aurantiaca
Geobacter sulfurreducens PCA
Haloferax volcanii
Herpetosiphon aurantiacus
Hydrogenobaculum sp. Y04AAS1
Ignicoccus hospitalis
Leptothrix cholodnii
Methanocaldococcus jannaschii
Methanococcus maripaludis C5
Methanococcus maripaludis S2
Methanopyrus kandleri

Methanosarcina acetivorans C2A

Nanoarchaeum equitans

Nitrosomonas europaea

Nostoc sp. PCC 7120

Pelodictyon phaeoclathratiforme

Persephonella marina EX-H1

Porphyromonas gingivalis

Pyrobaculum aerophilum IM2

Pyrobaculum arsenaticum

Pyrobaculum calidifontis

Pyrococcus furiosus

Pyrococcus horikoshii

Rhodopirellula baltica

Ruegeria pomeroyi

Salinispora arenicola

Salinispora tropica

Shewanella baltica OS185

Shewanella baltica OS223

Sulfitobacter sp. EE-36

Sulfitobacter sp. NAS-14.1

Sulfolobus tokodaii

Sulfurihydrogenibium sp. YO3AOP1

Sulfurihydrogenibium yellowstonense SS-5

Thermoanaerobacter pseudethanolicus

Thermotoga neapolitana DSM 4359

Thermotoga petrophila RKU-1

Thermotoga sp. RQ2

Thermus thermophilus HB8

Treponema denticola

Wolinella succinogenes

Zymomonas mobilis

Bibliography

1. Hooper L V. Commensal Host-Bacterial Relationships in the Gut. Science (80-.). [Internet]. 2001 [cited 2017 Mar 23];292:1115–8. Available from: http://www.sciencemag.org/cgi/doi/10.1126/science.1058709
2. Tringe SG, Rubin EM. Metagenomics: DNA sequencing of environmental samples. Nat. Rev. Genet. [Internet]. Nature Publishing Group; 2005 [cited 2017 Mar 23];6:805–14. Available from: http://www.nature.com/doifinder/10.1038/nrg1709
3. Methé BA, Nelson KE, Pop M, Creasy HH, Giglio MG, Huttenhower C, et al. A framework for human microbiome research. Nature [Internet]. Nature Research; 2012 [cited 2017 Mar 23];486:215–21. Available from: http://www.nature.com/doifinder/10.1038/nature11209
4. Heather JM, Chain B. The sequence of sequencers: The history of sequencing DNA. Genomics. Academic Press Inc.; 2016. p. 1–8.
5. Sims D, Sudbery I, Ilott NE, Heger A, Ponting CP. Sequencing depth and coverage: key considerations in genomic analyses. Nat. Rev. Genet. [Internet]. Nature Research; 2014 [cited 2017 Aug 5];15:121–32. Available from: http://www.nature.com/doifinder/10.1038/nrg3642
6. Kingsford C, Schatz MC, Pop M. Assembly complexity of prokaryotic genomes using short reads. BMC Bioinformatics [Internet]. 2010 [cited 2017 Apr 28];11:21. Available from: http://www.biomedcentral.com/1471-2105/11/21
7. Nagarajan N, Pop M. Parametric Complexity of Sequence Assembly: Theory and Applications to Next Generation Sequencing. J. Comput. Biol. [Internet]. 2009 [cited 2017 Apr 28];16:897–908. Available from: http://www.ncbi.nlm.nih.gov/pubmed/19580519
8. Compeau PEC, Pevzner PA, Tesler G. How to apply de Bruijn graphs to genome assembly. Nat. Biotechnol. [Internet]. 2011 [cited 2017 Mar 23];29:987–91. Available from: http://www.nature.com/nbt/journal/v29/n11/pdf/nbt.2023.pdf
9. Kececioglu JD, Myers EW. Combinatorial algorithms for DNA sequence assembly. Algorithmica [Internet]. Springer-Verlag; 1995 [cited 2017 Jun 28];13:7–51. Available from: http://link.springer.com/10.1007/BF01188580
10. 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 (80-.). [Internet]. American Association for the Advancement of Science; 1995 [cited 2020 Jul 7];269:496–512. Available from: https://science.sciencemag.org/content/269/5223/496
11. Ann Liebert M, Sutton GG, White O, Adams MD, Kerlavage AR. TIGR Assembler: A New Tool for Assembling Large Shotgun Sequencing Projects [Internet]. GENOME Sci. Technol. 1995. Available from: www.liebertpub.com
12. Lander ES, Linton LM, Birren B, Nusbaum C, Zody MC, Baldwin J, et al. Initial sequencing and analysis of the human genome. Nature [Internet]. Nature; 2001 [cited 2020 Jul 7];409:860–921. Available from: https://pubmed.ncbi.nlm.nih.gov/11237011/

13. Craig Venter J, Adams MD, Myers EW, Li PW, Mural RJ, Sutton GG, et al. The sequence of the human genome. Science (80-.). [Internet]. Science; 2001 [cited 2020 Jul 7];291:1304–51. Available from: https://pubmed.ncbi.nlm.nih.gov/11181995/
14. Pevzner PA, Tang H, Waterman MS. An Eulerian path approach to DNA fragment assembly. Proc. Natl. Acad. Sci. U. S. A. [Internet]. 2001 [cited 2017 Apr 28];98:9748–53. Available from: http://www.pnas.org/cgi/doi/10.1073/pnas.171285098
15. Nagarajan N, Pop M. Sequence assembly demystified. Nat. Rev. Genet. [Internet]. 2013 [cited 2017 Jun 15];14:157–67. Available from: http://www.nature.com/doifinder/10.1038/nrg3367
16. Miller JR, Koren S, Sutton G. Assembly algorithms for next-generation sequencing data. Genomics [Internet]. 2010 [cited 2017 Jun 15];95:315–27. Available from: http://linkinghub.elsevier.com/retrieve/pii/S0888754310000492
17. Salmela L, Walve R, Rivals E, Ukkonen E. Accurate self-correction of errors in long reads using de Bruijn graphs. [cited 2020 Aug 4]; Available from: http://www.cs.helsinki.fi/u/lmsalmel/LoRMA/.
18. Schatz MC, Phillippy AM, Sommer DD, Delcher AL, Puiu D, Narzisi G, et al. Hawkeye and AMOS: visualizing and assessing the quality of genome assemblies. [cited 2020 Jun 30]; Available from: http://amos.sourceforge.net.
19. Pop M, Phillippy A, Delcher AL, Salzberg SL. Comparative genome assembly. Brief. Bioinform. [Internet]. Oxford University Press; 2004 [cited 2017 Mar 23];5:237–48. Available from: https://academic.oup.com/bib/article-lookup/doi/10.1093/bib/5.3.237
20. Richter DC, Schuster SC, Huson DH. OSLay: optimal syntenic layout of unfinished assemblies. 2007 [cited 2020 Jun 30];23:1573–9. Available from: http://www-ab.informatik.unituebingen.de/software/oslay
21. Van Hijum SAFT, Zomer AL, Kuipers OP, Kok J. Projector 2: contig mapping for efficient gap-closure of prokaryotic genome sequence assemblies. [cited 2020 Jun 30]; Available from: http://genome.nhgri.nih.gov/blastall/
22. Assefa S, Keane TM, Otto TD, Newbold C, Berriman M. ABACAS: algorithm-based automatic contiguation of assembled sequences. Bioinforma. Appl. NOTE [Internet]. 2009 [cited 2020 Jun 30];25:1968–9. Available from: http://abacas.sourceforge.
23. Husemann P, Stoye J. r2cat: synteny plots and comparative assembly. Bioinformatics [Internet]. 2010 [cited 2015 Feb 12];26:570–1. Available from: http://bioinformatics.oxfordjournals.org/content/26/4/570.full
24. Husemann P, Stoye J. Phylogenetic comparative assembly. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics) [Internet]. Springer, Berlin, Heidelberg; 2009 [cited 2020 Jun 30]. p. 145–56. Available from: https://link.springer.com/chapter/10.1007/978-3-642-04241-6_13
25. Silva GG, Dutilh BE, David Matthews T, Elkins K, Schmieder R, Dinsdale EA, et al. Combining de novo and reference-guided assembly with scaffold_builder [Internet]. 2013. Available from: http://www.scfbm.org/content/8/1/23
26. Vezzi F, Cattonaro F, Policriti A. e-RGA: enhanced Reference Guided Assembly of Complex Genomes. EMBnet.journal [Internet]. 2011 [cited 2020 Jun 30];17:46–54. Available from:

http://journal.embnet.org/index.php/embnetjournal/article/view/208/484
27. Scholz M, Tett A, Segata N. Computational Tools for Taxonomic Microbiome Profiling of Shotgun Metagenomes. Metagenomics Microbiol. Elsevier Inc.; 2015. p. 67–80.
28. Weisburg WG, Barns SM, Pelletier DA, Lane DJ. 16S ribosomal DNA amplification for phylogenetic study. J. Bacteriol. [Internet]. American Society for Microbiology (ASM); 1991 [cited 2020 Jul 7];173:697–703. Available from: /pmc/articles/PMC207061/?report=abstract
29. Schloss PD, Westcott SL, Ryabin T, Hall JR, Hartmann M, Hollister EB, et al. Introducing mothur: Open-source, platform-independent, community-supported software for describing and comparing microbial communities. Appl. Environ. Microbiol. 2009;75:7537–41.
30. Callahan BJ, McMurdie PJ, Rosen MJ, Han AW, Johnson AJA, Holmes SP. DADA2: High-resolution sample inference from Illumina amplicon data. Nat. Methods [Internet]. Nature Publishing Group; 2016 [cited 2020 Jul 12];13:581–3. Available from: https://github.com/benjjneb/dada2
31. Caporaso JG, Kuczynski J, Stombaugh J, Bittinger K, Bushman FD, Costello EK, et al. QIIME allows analysis of high-throughput community sequencing data [Internet]. Nat. Methods. Nature Publishing Group; 2010 [cited 2020 Jul 12]. p. 335–6. Available from: http://qiime.sourceforge.
32. Quast C, Pruesse E, Yilmaz P, Gerken J, Schweer T, Yarza P, et al. The SILVA ribosomal RNA gene database project: Improved data processing and web-based tools. Nucleic Acids Res. 2013;41.
33. DeSantis TZ, Hugenholtz P, Larsen N, Rojas M, Brodie EL, Keller K, et al. Greengenes, a chimera-checked 16S rRNA gene database and workbench compatible with ARB. Appl. Environ. Microbiol. 2006;72:5069–72.
34. Cole JR, Wang Q, Chai B, Tiedje JM. The Ribosomal Database Project: Sequences and Software for High-Throughput rRNA Analysis. Handb. Mol. Microb. Ecol. I [Internet]. Hoboken, NJ, USA: John Wiley & Sons, Inc.; 2011 [cited 2020 Jun 30]. p. 313–24. Available from: http://doi.wiley.com/10.1002/9781118010518.ch36
35. Nilsson RH, Larsson K-H, Taylor AFS, Bengtsson-Palme J, Jeppesen TS, Schigel D, et al. The UNITE database for molecular identification of fungi: handling dark taxa and parallel taxonomic classifications. Nucleic Acids Res. [Internet]. 2018 [cited 2020 Jun 30];47:259–64. Available from: https://www.postgresql.org/
36. Case RJ, Boucher Y, Dahllöf I, Holmström C, Doolittle WF, Kjelleberg S. Use of 16S rRNA and rpoB genes as molecular markers for microbial ecology studies. Appl. Environ. Microbiol. [Internet]. American Society for Microbiology (ASM); 2007 [cited 2020 Jul 12];73:278–88. Available from: /pmc/articles/PMC1797146/?report=abstract
37. Engelbrektson A, Kunin V, Wrighton KC, Zvenigorodsky N, Chen F, Ochman H, et al. Experimental factors affecting PCR-based estimates of microbial species richness and evenness. ISME J. [Internet]. Nature Publishing Group; 2010 [cited 2020 Jul 12];4:642–7. Available from: www.nature.com/ismej
38. Kennedy K, Hall MW, Lynch MDJ, Moreno-Hagelsieb G, Neufeld JD. Evaluating bias of Illumina-based bacterial 16S rRNA gene profiles. Appl. Environ. Microbiol. [Internet]. American Society for Microbiology; 2014 [cited 2020 Jul

14];80:5717–22. Available from: http://dx.doi.org/10.1128
39. Comeau AM, Douglas GM, Langille MGI. Microbiome Helper: a Custom and Streamlined Workflow for Microbiome Research. mSystems [Internet]. American Society for Microbiology; 2017 [cited 2020 Jul 12];2. Available from: /pmc/articles/PMC5209531/?report=abstract
40. Gevers D, Cohan FM, Lawrence JG, Spratt BG, Coenye T, Feil EJ, et al. Re-evaluating prokaryotic species [Internet]. Nat. Rev. Microbiol. Nature Publishing Group; 2005 [cited 2020 Jul 12]. p. 733–9. Available from: www.nature.com/reviews/micro
41. Ochman H, Wilson AC. Evolution in bacteria: Evidence for a universal substitution rate in cellular genomes [Internet]. J. Mol. Evol. Springer-Verlag; 1987 [cited 2020 Jul 13]. p. 377. Available from: https://link.springer.com/article/10.1007/BF02101157
42. Yamamoto S, Harayama S. Phylogenetic relationships of Pseudomonas putida strains deduced from the nucleotide sequences of gyrB, rpoD and 16S rRNA genes. Int. J. Syst. Bacteriol. [Internet]. Microbiology Society; 1998 [cited 2020 Jul 13];48:813–9. Available from: https://www.microbiologyresearch.org/content/journal/ijsem/10.1099/00207713-48-3-813
43. Palys T, Berger E, Mitrica I, Nakamura LK, Cohan FM. Protein-coding genes as molecular markers for ecologically distinct populations: The case of two Bacillus species. Int. J. Syst. Evol. Microbiol. [Internet]. Society for General Microbiology; 2000 [cited 2020 Jul 13];50:1021–8. Available from: https://pubmed.ncbi.nlm.nih.gov/10843041/
44. Mende DR, Sunagawa S, Zeller G, Bork P. Accurate and universal delineation of prokaryotic species. Nat. Methods [Internet]. Nature Publishing Group; 2013 [cited 2020 Jul 13];10:881–4. Available from: http://www.bork.embl.de/software/speci/
45. Sunagawa S, Mende DR, Zeller G, Izquierdo-Carrasco F, Berger SA, Kultima JR, et al. Metagenomic species profiling using universal phylogenetic marker genes. Nat. Methods [Internet]. Nature Publishing Group, a division of Macmillan Publishers Limited. All Rights Reserved.; 2013 [cited 2014 Jul 18];10:1196–9. Available from: http://dx.doi.org/10.1038/nmeth.2693
46. Ciccarelli FD, Doerks T, von Mering C, Creevey CJ, Snel B, Bork P. Toward automatic reconstruction of a highly resolved tree of life. Science [Internet]. 2006 [cited 2014 Jul 10];311:1283–7. Available from: http://www.ncbi.nlm.nih.gov/pubmed/16513982
47. Tatusov RL, Galperin MY, Natale DA, Koonin E V. The COG database: a tool for genome-scale analysis of protein functions and evolution [Internet]. Nucleic Acids Res. 2000. Available from: http://www.ncbi.nlm.nih.gov/COG
48. Galperin MY, Makarova KS, Wolf YI, Koonin E V. Expanded microbial genome coverage and improved protein family annotation in the COG database. Nucleic Acids Res. [Internet]. 2014 [cited 2020 Jul 14];43:261–9. Available from: http://www.ncbi.nlm.nih.gov/COG/
49. Pruitt K, Brown G, Tatusova T, Maglott D. The Reference Sequence (RefSeq) Database [Internet]. National Center for Biotechnology Information (US); 2012 [cited 2015 Sep 17]. Available from: http://www.ncbi.nlm.nih.gov/books/NBK21091/

50. 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. [Internet]. Oxford University Press; 2016 [cited 2017 Mar 27];44:D733-45. Available from: http://www.ncbi.nlm.nih.gov/pubmed/26553804
51. RefSeq: NCBI Reference Sequence Database [Internet]. [cited 2020 Jul 2]. Available from: https://www.ncbi.nlm.nih.gov/refseq/
52. Pruitt KD, Tatusova T, Maglott DR. NCBI reference sequences (RefSeq): a curated non-redundant sequence database of genomes, transcripts and proteins. [cited 2020 Jul 2]; Available from: http://www.ncbi.nlm.nih.gov/RefSeq/
53. Ye SH, Siddle KJ, Park DJ, Sabeti PC. Benchmarking Metagenomics Tools for Taxonomic Classification [Internet]. Cell. Cell Press; 2019 [cited 2020 Jul 2]. p. 779–94. Available from: https://doi.org/10.1016/j.cell.2019.07.010
54. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ, Yu Y, et al. Basic local alignment search tool. J. Mol. Biol. [Internet]. BioMed Central; 1990 [cited 2017 Mar 23];215:403–10. Available from: http://linkinghub.elsevier.com/retrieve/pii/S0022283605803602
55. Huson DH, Beier S, Flade I, Górska A, El-Hadidi M, Mitra S, et al. MEGAN Community Edition - Interactive Exploration and Analysis of Large-Scale Microbiome Sequencing Data. Poisot T, editor. PLOS Comput. Biol. [Internet]. Public Library of Science; 2016 [cited 2016 Jul 19];12:e1004957. Available from: http://dx.plos.org/10.1371/journal.pcbi.1004957
56. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. Basic local alignment search tool. J. Mol. Biol. [Internet]. J Mol Biol; 1990 [cited 2020 Jul 1];215:403–10. Available from: https://pubmed.ncbi.nlm.nih.gov/2231712/
57. Altschul SF, Madden TL, Schäffer AA, Zhang J, Zhang Z, Miller W, et al. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs [Internet]. Nucleic Acids Res. Oxford University Press; 1997. Available from: https://academic.oup.com/nar/article-abstract/25/17/3389/1061651
58. McCall C, Xagoraraki I. Comparative study of sequence aligners for detecting antibiotic resistance in bacterial metagenomes. Lett. Appl. Microbiol. [Internet]. Blackwell Publishing Ltd; 2018 [cited 2020 Jul 2];66:162–8. Available from: http://doi.wiley.com/10.1111/lam.12842
59. Smith TF, Waterman MS. Identification of common molecular subsequences. J. Mol. Biol. [Internet]. 1981 [cited 2014 Dec 11];147:195–7. Available from: http://www.ncbi.nlm.nih.gov/pubmed/7265238
60. Zhang Z, Schwartz S, Wagner L, Miller W. A Greedy Algorithm for Aligning DNA Sequences [Internet]. J. Comput. Biol. Mary Ann Liebert, Inc. Pp; 2000. Available from: www.liebertpub.com
61. Buchfink B, Xie C, Huson DH. Fast and sensitive protein alignment using DIAMOND [Internet]. Nat. Methods. Nature Publishing Group; 2014 [cited 2020 Jul 16]. p. 59–60. Available from: http://ab.inf.uni-tuebingen.de/software/diamond
62. Menzel P, Ng KL, Krogh A, Marth G, Lipman D. Fast and sensitive taxonomic classification for metagenomics with Kaiju. Nat. Commun. [Internet]. Nature Publishing Group; 2016 [cited 2017 Mar 23];7:11257. Available from: http://www.nature.com/doifinder/10.1038/ncomms11257

63. Burrows M, Burrows M, Wheeler DJ. A block-sorting lossless data compression algorithm. 1994 [cited 2017 Sep 25];16. Available from: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.3.8069
64. Ferragina P, Manzini G. Opportunistic data structures with applications. Annu. Symp. Found. Comput. Sci. - Proc. IEEE; 2000. p. 390–8.
65. Liu B, Gibbons T, Ghodsi M, Treangen T, Pop M. Accurate and fast estimation of taxonomic profiles from metagenomic shotgun sequences. BMC Genomics [Internet]. 2011 [cited 2017 Mar 23];12:S4. Available from: http://bmcgenomics.biomedcentral.com/articles/10.1186/1471-2164-12-S2-S4
66. Segata N, Waldron L, Ballarini A, Narasimhan V, Jousson O, Huttenhower C. Metagenomic microbial community profiling using unique clade-specific marker genes. Nat. Methods [Internet]. Nature Publishing Group, a division of Macmillan Publishers Limited. All Rights Reserved.; 2012 [cited 2015 Mar 6];9:811–4. Available from: http://dx.doi.org/10.1038/nmeth.2066
67. Buchfink B, Xie C, Huson DH. Fast and sensitive protein alignment using DIAMOND [Internet]. Nat. Methods. Nature Publishing Group; 2014 [cited 2020 Jun 30]. p. 59–60. Available from: https://www.nature.com/articles/nmeth.3176
68. Langmead B, Salzberg SL. Fast gapped-read alignment with Bowtie 2. Nat. Methods [Internet]. Nature Publishing Group, a division of Macmillan Publishers Limited. All Rights Reserved.; 2012 [cited 2014 Jul 10];9:357–9. Available from: http://dx.doi.org/10.1038/nmeth.1923
69. Ames SK, Hysom DA, Gardner SN, Lloyd GS, Gokhale MB, Allen JE. Scalable metagenomic taxonomy classification using a reference genome database. Bioinformatics [Internet]. Ottawa, Canada; 2013 [cited 2017 Jul 5];29:2253–60. Available from: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btt389
70. Ounit R, Wanamaker S, Close TJ, Lonardi S. CLARK: fast and accurate classification of metagenomic and genomic sequences using discriminative k-mers. BMC Genomics [Internet]. 2015 [cited 2017 Jul 5];16:236. Available from: http://www.biomedcentral.com/1471-2164/16/236
71. Wood DE, Salzberg SL. Kraken: ultrafast metagenomic sequence classification using exact alignments. Genome Biol. [Internet]. 2014 [cited 2017 Mar 23];15:R46. Available from: http://genomebiology.biomedcentral.com/articles/10.1186/gb-2014-15-3-r46
72. Ulyantsev VI, Kazakov S V., Dubinkina VB, Tyakht A V., Alexeev DG. MetaFast: fast reference-free graph-based comparison of shotgun metagenomic data. Bioinformatics [Internet]. Springer, Berlin/Heidelberg; 2016 [cited 2017 Jul 5];32:2760–7. Available from: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btw312
73. 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. [Internet]. 2016 [cited 2017 Jun 30];17:132. Available from: http://genomebiology.biomedcentral.com/articles/10.1186/s13059-016-0997-x
74. Kim D, Song L, Breitwieser FP, Salzberg SL. Centrifuge: rapid and sensitive classification of metagenomic sequences. Genome Res. [Internet]. Cold Spring Harbor Laboratory Press; 2016 [cited 2017 Jul 9];26:1721–9. Available from:

http://www.ncbi.nlm.nih.gov/pubmed/27852649
75. Wood DE, Lu J, Langmead B. Improved metagenomic analysis with Kraken 2. Genome Biol. [Internet]. BioMed Central Ltd.; 2019 [cited 2020 Jul 18];20:257. Available from: https://genomebiology.biomedcentral.com/articles/10.1186/s13059-019-1891-0
76. Lu J, Breitwieser FP, Thielen P, Salzberg SL. Bracken: estimating species abundance in metagenomics data. PeerJ Comput. Sci. [Internet]. PeerJ Inc.; 2017 [cited 2017 Jul 9];3:e104. Available from: https://peerj.com/articles/cs-104
77. Ounit R, Lonardi S. Higher classification sensitivity of short metagenomic reads with CLARK-S. [cited 2020 Jul 2]; Available from: http://clark.cs.ucr.edu/
78. Ondov BD, Starrett GJ, Sappington A, Kostic A, Koren S, Buck CB, et al. Mash Screen: High-throughput sequence containment estimation for genome discovery. Genome Biol. [Internet]. BioMed Central Ltd.; 2019 [cited 2020 Jul 2];20:232. Available from: https://genomebiology.biomedcentral.com/articles/10.1186/s13059-019-1841-x
79. McIntyre ABR, Ounit R, Afshinnekoo E, Prill RJ, Hénaff E, Alexander N, et al. Comprehensive benchmarking and ensemble approaches for metagenomic classifiers. Genome Biol. [Internet]. BioMed Central Ltd.; 2017 [cited 2020 Jul 3];18:182. Available from: https://genomebiology.biomedcentral.com/articles/10.1186/s13059-017-1299-7
80. Meyer F, Bremges A, Belmann P, Janssen S, McHardy AC, Koslicki D. Assessing taxonomic metagenome profilers with OPAL. Genome Biol. [Internet]. BioMed Central Ltd.; 2019 [cited 2020 Jul 3];20:51. Available from: https://genomebiology.biomedcentral.com/articles/10.1186/s13059-019-1646-y
81. 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 [Internet]. Nature Publishing Group; 2017 [cited 2020 Jul 3];14:1063–71. Available from: https://www.nature.com/articles/nmeth.4458
82. Morowitz MJ, Denef VJ, Costello EK, Thomas BC, Poroyko V, Relman DA, et al. Strain-resolved community genomic analysis of gut microbial colonization in a premature infant. Proc. Natl. Acad. Sci. [Internet]. 2011 [cited 2017 May 1];108:1128–33. Available from: http://www.ncbi.nlm.nih.gov/pubmed/21191099
83. Treangen TJ, Abraham A-L, Touchon M, Rocha EPC. Genesis, effects and fates of repeats in prokaryotic genomes. FEMS Microbiol. Rev. [Internet]. 2009 [cited 2017 May 1];33:539–71. Available from: http://www.ncbi.nlm.nih.gov/pubmed/19396957
84. Koren S, Harhay GP, Smith TP, Bono JL, Harhay DM, Mcvey SD, et al. Reducing assembly complexity of microbial genomes with single-molecule sequencing. Genome Biol. [Internet]. 2013 [cited 2017 Apr 26];14:R101. Available from: http://www.ncbi.nlm.nih.gov/pubmed/24034426
85. Namiki T, Hachiya T, Tanaka H, Sakakibara Y. MetaVelvet: an extension of Velvet assembler to de novo metagenome assembly from short sequence reads. Nucleic Acids Res. [Internet]. 2012 [cited 2014 Jul 10];40:e155. Available from: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3488206&tool=pmcentrez&rendertype=abstract
86. Greenwald WW, Klitgord N, Seguritan V, Yooseph S, Venter JC, Garner C, et al.

Utilization of defined microbial communities enables effective evaluation of meta-genomic assemblies. BMC Genomics [Internet]. 2017 [cited 2017 May 1];18:296. Available from: http://bmcgenomics.biomedcentral.com/articles/10.1186/s12864-017-3679-5
87. Awad S, Irber L, Brown CT. Evaluating Metagenome Assembly on a Simple Defined Community with Many Strain Variants. bioRxiv [Internet]. 2017 [cited 2017 Jun 29]; Available from: http://www.biorxiv.org/content/early/2017/06/25/155358
88. Peng Y, Leung HCM, Yiu SM, Chin FYL. IDBA-UD: a de novo assembler for single-cell and metagenomic sequencing data with highly uneven depth. Bioinformatics [Internet]. Oxford University Press; 2012 [cited 2017 Mar 23];28:1420–8. Available from: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/bts174
89. Peng Y, Leung HCM, Yiu SM, Chin FYL. IDBA – A Practical Iterative de Bruijn Graph De Novo Assembler. Springer, Berlin, Heidelberg; 2010 [cited 2017 Apr 28]. p. 426–40. Available from: http://link.springer.com/10.1007/978-3-642-12683-3_28
90. Li D, Liu C-M, Luo R, Sadakane K, Lam T-W. MEGAHIT: an ultra-fast single-node solution for large and complex metagenomics assembly via succinct de Bruijn graph. Bioinformatics [Internet]. Oxford University Press; 2015 [cited 2017 Mar 23];31:1674–6. Available from: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btv033
91. Bowe A, Onodera T, Sadakane K, Shibuya T. Succinct de Bruijn Graphs. Springer, Berlin, Heidelberg; 2012 [cited 2017 Apr 27]. p. 225–35. Available from: http://link.springer.com/10.1007/978-3-642-33122-0_18
92. Nurk S, Meleshko D, Korobeynikov A, Pevzner PA. metaSPAdes: a new versatile metagenomic assembler. Genome Res. [Internet]. Cold Spring Harbor Laboratory Press; 2017 [cited 2017 Mar 29];gr.213959.116. Available from: http://www.ncbi.nlm.nih.gov/pubmed/28298430
93. Bankevich A, Nurk S, Antipov D, Gurevich AA, Dvorkin M, Kulikov AS, et al. SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. J. Comput. Biol. [Internet]. Mary Ann Liebert, Inc.; 2012 [cited 2017 Apr 27];19:455–77. Available from: http://www.ncbi.nlm.nih.gov/pubmed/22506599
94. Prjibelski AD, Vasilinetc I, Bankevich A, Gurevich A, Krivosheeva T, Nurk S, et al. ExSPAnder: a universal repeat resolver for DNA fragment assembly. Bioinformatics [Internet]. 2014 [cited 2017 May 1];30:i293–301. Available from: http://www.ncbi.nlm.nih.gov/pubmed/24931996
95. Laserson J, Jojic V, Koller D. Genovo: *De Novo* Assembly for Metagenomes. J. Comput. Biol. [Internet]. Mary Ann Liebert, Inc. 140 Huguenot Street, 3rd Floor New Rochelle, NY 10801 USA ; 2011 [cited 2017 Mar 23];18:429–43. Available from: http://www.liebertonline.com/doi/abs/10.1089/cmb.2010.0244
96. Zerbino DR, Birney E. Velvet: algorithms for de novo short read assembly using de Bruijn graphs. Genome Res. [Internet]. Cold Spring Harbor Laboratory Press; 2008 [cited 2017 Mar 23];18:821–9. Available from: http://www.ncbi.nlm.nih.gov/pubmed/18349386
97. Simpson JT, Wong K, Jackman SD, Schein JE, Jones SJM, Birol I. ABySS: a parallel assembler for short read sequence data. Genome Res. [Internet]. 2009 [cited 2014 Jul 11];19:1117–23. Available from:

http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2694472&tool=pmcentrez&rendertype=abstract
98. Li R, Zhu H, Ruan J, Qian W, Fang X, Shi Z, et al. De novo assembly of human genomes with massively parallel short read sequencing. Genome Res. [Internet]. 2010 [cited 2014 Jul 22];20:265–72. Available from: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2813482&tool=pmcentrez&rendertype=abstract
99. Staden R. A strategy of DNA sequencing employing computer programs. Nucleic Acids Res. [Internet]. Oxford University Press; 1979 [cited 2017 May 1];6:2601–10. Available from: http://www.ncbi.nlm.nih.gov/pubmed/461197
100. diCenzo GC, Finan TM. The Divided Bacterial Genome: Structure, Function, and Evolution. Microbiol. Mol. Biol. Rev. [Internet]. American Society for Microbiology; 2017 [cited 2020 Jul 16];81. Available from: https://doi.org/10.1128/
101. Mikheenko A, Saveliev V, Gurevich A. MetaQUAST: evaluation of metagenome assemblies. Bioinformatics [Internet]. 2016 [cited 2017 Apr 26];32:1088–90. Available from: http://www.ncbi.nlm.nih.gov/pubmed/26614127
102. Parks DH, Imelfort M, Skennerton CT, Hugenholtz P, Tyson GW. CheckM: assessing the quality of microbial genomes recovered from isolates, single cells, and metagenomes. Genome Res. [Internet]. 2015 [cited 2015 May 20];gr.186072.114-. Available from: http://genome.cshlp.org/content/early/2015/05/14/gr.186072.114.abstract
103. Walker BJ, Abeel T, Shea T, Priest M, Abouelliel A, Sakthikumar S, et al. Pilon: an integrated tool for comprehensive microbial variant detection and genome assembly improvement. PLoS One [Internet]. Broad Institute of MIT and Harvard, Cambridge, Massachusetts, United States of America. Broad Institute of MIT and Harvard, Cambridge, Massachusetts, United States of America; VIB Department of Plant Systems Biology, Ghent University, Ghent, Belgium. Broa; 2014;9:e112963. Available from: http://dx.doi.org/10.1371/journal.pone.0112963
104. Parks DH, Imelfort M, Skennerton CT, Hugenholtz P, Tyson GW. CheckM: assessing the quality of microbial genomes recovered from isolates, single cells, and metagenomes. Genome Res. [Internet]. Cold Spring Harbor Laboratory Press; 2015 [cited 2017 Apr 26];25:1043–55. Available from: http://www.ncbi.nlm.nih.gov/pubmed/25977477
105. Simã FA, Waterhouse RM, Ioannidis P, Kriventseva E V, Zdobnov EM. BUSCO: assessing genome assembly and annotation completeness with single-copy orthologs. [cited 2020 Jul 17]; Available from: https://academic.oup.com/bioinformatics/article-abstract/31/19/3210/211866
106. Mikheenko A, Saveliev V, Gurevich A. MetaQUAST: evaluation of metagenome assemblies. Bioinformatics [Internet]. 2016 [cited 2017 Mar 27];32:1088–90. Available from: http://www.ncbi.nlm.nih.gov/pubmed/26614127
107. Gurevich A, Saveliev V, Vyahhi N, Tesler G. QUAST: quality assessment tool for genome assemblies. Bioinformatics [Internet]. 2013 [cited 2017 May 1];29:1072–5. Available from: http://www.ncbi.nlm.nih.gov/pubmed/23422339
108. kmer-mask [Internet]. [cited 2020 Jul 2]. Available from: http://kmer.sourceforge.net/wiki/index.php?Main_Page)
109. Kultima JR, Sunagawa S, Li J, Chen W, Chen H, Mende DR, et al. MOCAT: a

metagenomics assembly and gene prediction toolkit. Gilbert JA, editor. PLoS One [Internet]. 2012 [cited 2017 Mar 28];7:e47656. Available from: http://dx.plos.org/10.1371/journal.pone.0047656
110. Li Ã W, Godzik A. BIOINFORMATICS APPLICATIONS NOTE Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. 2006 [cited 2020 Jul 3];22:1658–9. Available from: http://cd-hit.org
111. Getting Started with Meryl - kmer [Internet]. [cited 2020 Jul 2]. Available from: http://kmer.sourceforge.net/wiki/index.php/Getting_Started_with_Meryl
112. McClelland M, Sanderson KE, Spieth J, Clifton SW, Latreille P, Courtney L, et al. Complete genome sequence of Salmonella enterica serovar Typhimurium LT2. Nature [Internet]. ASM Press; 2001 [cited 2020 Jul 3];413:852–6. Available from: http://www.sanger.ac.uk/Software/
113. Shakya M, Quince C, Campbell JH, Yang ZK, Schadt CW, Podar M. Comparative metagenomic and rRNA microbial diversity characterization using archaeal and bacterial synthetic communities. Environ. Microbiol. [Internet]. 2013 [cited 2017 Mar 23];15:1882–99. Available from: http://doi.wiley.com/10.1111/1462-2920.12086
114. Silva GGZ, Cuevas DA, Dutilh BE, Edwards RA. FOCUS: An alignment-free model to identify organisms in metagenomes using non-negative least squares. PeerJ [Internet]. PeerJ Inc.; 2014 [cited 2020 Aug 15];2014. Available from: /pmc/articles/PMC4060023/?report=abstract
115. Nguyen NP, Mirarab S, Liu B, Pop M, Warnow T. TIPP: Taxonomic identification and phylogenetic profiling. Bioinformatics [Internet]. Oxford University Press; 2014 [cited 2020 Aug 14];30:3548–55. Available from: https://pubmed.ncbi.nlm.nih.gov/25359891/
116. Liu B, Gibbons T, Ghodsi M, Treangen T, Pop M. Accurate and fast estimation of taxonomic profiles from metagenomic shotgun sequences. BMC Genomics [Internet]. 2011 [cited 2015 Feb 25];12 Suppl 2:S4. Available from: http://www.biomedcentral.com/1471-2164/12/S2/S4
117. Koslicki D, Foucart S, Rosen G. Quikr: A method for rapid reconstruction of bacterial communities via compressive sensing. Bioinformatics [Internet]. Bioinformatics; 2013 [cited 2020 Aug 14];29:2096–102. Available from: https://pubmed.ncbi.nlm.nih.gov/23786768/
118. Klingenberg H, Aßhauer KP, Lingner T, Meinicke P. Protein signature-based estimation of metagenomic abundances including all domains of life and viruses. Bioinformatics [Internet]. Bioinformatics; 2013 [cited 2020 Aug 15];29:973–80. Available from: https://pubmed.ncbi.nlm.nih.gov/23418187/
119. Koslicki D, Falush D. MetaPalette: a k-mer Painting Approach for Metagenomic Taxonomic Profiling and Quantification of Novel Strain Variation. mSystems [Internet]. American Society for Microbiology; 2016 [cited 2020 Aug 15];1. Available from: /pmc/articles/PMC5069763/?report=abstract
120. Podell S, Ugalde JA, Narasingarao P, Banfield JF, Heidelberg KB, Allen EE. Assembly-driven community genomics of a hypersaline microbial ecosystem. Mormile MR, editor. PLoS One [Internet]. 2013 [cited 2017 Apr 28];8:e61692. Available from: http://dx.plos.org/10.1371/journal.pone.0061692
121. Narasingarao P, Podell S, Ugalde JA, Brochier-Armanet C, Emerson JB, Brocks

JJ, et al. De novo metagenomic assembly reveals abundant novel major lineage of Archaea in hypersaline microbial communities. ISME J. [Internet]. 2012 [cited 2017 Apr 28];6:81–93. Available from: http://www.nature.com/doifinder/10.1038/ismej.2011.78
122. Ji P, Zhang Y, Wang J, Zhao F. MetaSort untangles metagenome assembly by reducing microbial community complexity. Nat. Commun. [Internet]. 2017 [cited 2017 Apr 28];8:14306. Available from: http://www.nature.com/doifinder/10.1038/ncomms14306
123. Sangwan N, Xia F, Gilbert JA. Recovering complete and draft population genomes from metagenome datasets. Microbiome [Internet]. 2016 [cited 2017 Apr 28];4:8. Available from: http://www.microbiomejournal.com/content/4/1/8
124. Mukherjee S, Seshadri R, Varghese NJ, Eloe-Fadrosh EA, Meier-Kolthoff JP, Göker M, et al. 1,003 reference genomes of bacterial and archaeal isolates expand coverage of the tree of life. Nat. Biotechnol. [Internet]. Nature Research; 2017 [cited 2017 Jul 20];35:676–83. Available from: http://www.nature.com/doifinder/10.1038/nbt.3886
125. Li H, Homer N. A survey of sequence alignment algorithms for next-generation sequencing. [cited 2020 Jul 1]; Available from: http://novocraft.com
126. Hatem A, Bozdağ D, Toland AE, Çatalyürek Ü V. Benchmarking short sequence mapping tools. BMC Bioinformatics [Internet]. 2013 [cited 2017 Jul 5];14:184. Available from: http://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-14-184
127. Wu Thomas D, Nacu Serban. Fast and SNP-tolerant Detection of Complex Variants and Splicing in Short Reads - PubMed. Bioinformatics [Internet]. 2010 [cited 2020 Jul 1];26:873–81. Available from: https://pubmed.ncbi.nlm.nih.gov/20147302/
128. Novocraft [Internet]. [cited 2020 Jul 1]. Available from: http://www.novocraft.com/
129. Alkan C, Kidd JM, Marques-Bonet T, Aksay G, Antonacci F, Hormozdiari F, et al. Personalized copy number and segmental duplication maps using next-generation sequencing. Nat. Genet. [Internet]. Nat Genet; 2009 [cited 2020 Jul 1];41:1061–7. Available from: https://pubmed.ncbi.nlm.nih.gov/19718026/
130. Hach F, Hormozdiari F, Alkan C, Hormozdiari F, Birol I, Eichler EE, et al. mrsFAST: a cache-oblivious algorithm for short-read mapping. Nat. Methods [Internet]. Nature Research; 2010 [cited 2017 Jun 30];7:576–7. Available from: http://www.nature.com/doifinder/10.1038/nmeth0810-576
131. Misra S, Narayanan R, Lin S, Choudhary A. FANGS: High speed sequence mapping for next generation sequencers. Proc. ACM Symp. Appl. Comput. [Internet]. 2010 [cited 2020 Jul 1]. p. 1539–46. Available from: https://www.scholars.northwestern.edu/en/publications/fangs-high-speed-sequence-mapping-for-next-generation-sequencers
132. Li H, Ruan J, Durbin R. Mapping short DNA sequencing reads and calling variants using mapping quality scores. Genome Res. [Internet]. Cold Spring Harbor Laboratory Press; 2008 [cited 2017 Jul 6];18:1851–8. Available from: http://www.ncbi.nlm.nih.gov/pubmed/18714091
133. Smith AD, Xuan Z, Zhang MQ. Using quality scores and longer reads improves

accuracy of Solexa read mapping. BMC Bioinformatics [Internet]. BMC Bioinformatics; 2008 [cited 2020 Jul 1];9. Available from: https://pubmed.ncbi.nlm.nih.gov/18307793/
134. Langmead B, Trapnell C, Pop M, Salzberg SL. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biol. [Internet]. 2009 [cited 2014 Jul 9];10:R25. Available from: http://genomebiology.com/2009/10/3/R25
135. Li H, Durbin R. Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics [Internet]. 2009 [cited 2014 Jul 9];25:1754–60. Available from: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2705234&tool=pmcentrez&rendertype=abstract
136. Li R, Yu C, Li Y, Lam T-W, Yiu S-M, Kristiansen K, et al. SOAP2: an improved ultrafast tool for short read alignment. Bioinformatics [Internet]. Digital Equipment Corporation, CA; 2009 [cited 2017 Jul 6];25:1966–7. Available from: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btp336
137. Ruffalo M, LaFramboise T, Koyuturk M. Comparative analysis of algorithms for next-generation sequencing read alignment. Bioinformatics [Internet]. Springer, Berlin/Heidelberg; 2011 [cited 2017 Jul 6];27:2790–6. Available from: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btr477
138. Chouvarine P, Wiehlmann L, Moran Losada P, DeLuca DS, Tümmler B, Speed T. Filtration and Normalization of Sequencing Read Data in Whole-Metagenome Shotgun Samples. Dalby AR, editor. PLoS One [Internet]. Public Library of Science; 2016 [cited 2017 Jul 6];11:e0165015. Available from: http://dx.plos.org/10.1371/journal.pone.0165015
139. Petersen TN, Lukjancenko O, Thomsen MCF, Maddalena Sperotto M, Lund O, Møller Aarestrup F, et al. MGmapper: Reference based mapping and taxonomy annotation of metagenomics sequence reads. An L, editor. PLoS One [Internet]. Public Library of Science; 2017 [cited 2017 Jul 6];12:e0176469. Available from: http://dx.plos.org/10.1371/journal.pone.0176469
140. Sequence Alignment/Map Format Specification [Internet]. 2020. Available from: https://github.com/samtools/hts-specs.
141. Li H, Barrett J. A statistical framework for SNP calling, mutation discovery, association mapping and population genetical parameter estimation from sequencing data. 2011 [cited 2020 Jul 3];27:2987–93. Available from: http://samtools.sourceforge.net
142. McKenna A, Hanna M, Banks E, Sivachenko A, Cibulskis K, Kernytsky A, et al. The genome analysis toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data. Genome Res. [Internet]. Cold Spring Harbor Laboratory Press; 2010 [cited 2020 Jul 3];20:1297–303. Available from: http://www.genome.org/cgi/doi/10.1101/gr.107524.110.
143. Walker BJ, Abeel T, Shea T, Priest M, Abouelliel A, Sakthikumar S, et al. Pilon: An Integrated Tool for Comprehensive Microbial Variant Detection and Genome Assembly Improvement. Wang J, editor. PLoS One [Internet]. Public Library of

Science; 2014 [cited 2017 Mar 23];9:e112963. Available from: http://dx.plos.org/10.1371/journal.pone.0112963
144. Vaser R, Sović I, Nagarajan N, Šikić M. Fast and accurate de novo genome assembly from long uncorrected reads. Genome Res. [Internet]. Cold Spring Harbor Laboratory Press; 2017 [cited 2020 Jul 3];27:737–46. Available from: /pmc/articles/PMC5411768/?report=abstract
145. Zimin A V, Salzberg SL. The genome polishing tool POLCA makes fast and accurate corrections in genome assemblies. [cited 2020 Jul 3]; Available from: https://doi.org/10.1101/2019.12.17.864991
146. Warren RL, Coombe L, Mohamadi H, Zhang J, Jaquish B, Isabel N, et al. ntEdit: scalable genome sequence polishing. [cited 2020 Jul 3]; Available from: https://academic.oup.com/bioinformatics/article-abstract/35/21/4430/5490204
147. Zimin A V, Març Ais G, Puiu D, Roberts M, Salzberg SL, Yorke JA. Genome analysis The MaSuRCA genome assembler. 2013 [cited 2020 Jul 3];29:2669–77. Available from: https://academic.oup.com/bioinformatics/article-abstract/29/21/2669/195975
148. Solomon B, Kingsford C. Large-Scale Search of Transcriptomic Read Sets with Sequence Bloom Trees. doi.org [Internet]. Cold Spring Harbor Laboratory; 2015 [cited 2017 Sep 25];017087. Available from: https://www.biorxiv.org/content/early/2015/03/26/017087
149. bcgsc/ntHits: Identifying repeats in high-throughput sequencing data [Internet]. [cited 2020 Jul 4]. Available from: https://github.com/bcgsc/nthits
150. Feige U. A threshold of ln n for approximating set cover. J. ACM [Internet]. ACM; 1998 [cited 2015 May 7];45:634–52. Available from: http://dl.acm.org/citation.cfm?id=285055.285059
151. Liu B. Title of dissertation: COMPUTATIONAL METAGENOMICS: NETWORK, CLASSIFICATION AND ASSEMBLY [Internet]. 2012. Available from: http://drum.lib.umd.edu/handle/1903/13278
152. Beller T, Ohlebusch E. A representation of a compressed de Bruijn graph for pan-genome analysis that enables search. Algorithms Mol. Biol. [Internet]. 2016 [cited 2017 Jul 9];11:20. Available from: http://almob.biomedcentral.com/articles/10.1186/s13015-016-0083-7
153. Baier U, Beller T, Ohlebusch E. Graphical pan-genome analysis with compressed suffix trees and the Burrows–Wheeler transform. Bioinformatics [Internet]. Oldenbusch Verlag, Bremen, Germany,; 2016 [cited 2017 Jul 9];32:497–504. Available from: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btv603
154. Marcus S, Lee H, Schatz MC. SplitMEM: a graphical algorithm for pan-genome analysis with suffix skips. Bioinformatics [Internet]. 2014 [cited 2017 Jul 9];30:3476–83. Available from: http://www.ncbi.nlm.nih.gov/pubmed/25398610
155. Holley G, Wittler R, Stoye J. Bloom Filter Trie: an alignment-free and reference-free data structure for pan-genome storage. Algorithms Mol. Biol. [Internet]. 2016 [cited 2017 Jul 9];11:3. Available from: http://almob.biomedcentral.com/articles/10.1186/s13015-016-0066-8
156. Almodaresi F, Sarkar H, Srivastava A, Patro R. A space and time-efficient index for the compacted colored de Bruijn graph. [cited 2020 Jul 5]; Available from:

https://academic.oup.com/bioinformatics/article-abstract/34/13/i169/5045749
157. Anari SS, Ridder D de, Schranz ME, Smit S. Pangenomic read mapping. bioRxiv [Internet]. Cold Spring Harbor Laboratory; 2019 [cited 2020 Jul 5];813634. Available from: https://doi.org/10.1101/813634
158. Gurevich A, Saveliev V, Vyahhi N, Tesler G. QUAST: quality assessment tool for genome assemblies. Bioinformatics [Internet]. 2013 [cited 2017 Mar 28];29:1072–5. Available from: http://www.ncbi.nlm.nih.gov/pubmed/23422339
159. HMP2 assembly details [Internet]. Available from: http://gembox.cbcb.umd.edu/metacompass
160. Li K, Bihan M, Yooseph S, Methé BA, Ludwig W. Analyses of the Microbial Diversity across the Human Microbiome. Xu P, editor. PLoS One [Internet]. Public Library of Science; 2012 [cited 2017 Mar 23];7:e32118. Available from: http://dx.plos.org/10.1371/journal.pone.0032118
161. Li H. Minimap2: pairwise alignment for nucleotide sequences. [cited 2020 Jul 5]; Available from: https://github.com/ruanjue/smartdenovo;
162. Puigbò P, Makarova KS, Kristensen DM, Wolf YI, Koonin E V. Reconstruction of the evolution of microbial defense systems. BMC Evol. Biol. [Internet]. 2017 [cited 2017 May 6];17:94. Available from: http://www.ncbi.nlm.nih.gov/pubmed/28376755
163. Belkaid Y, Segre JA. Dialogue between skin microbiota and immunity. Science (80-.). [Internet]. 2014 [cited 2017 May 6];346:954–9. Available from: http://www.ncbi.nlm.nih.gov/pubmed/25414304
164. Ambur OH, Davidsen T, Frye SA, Balasingham S V, Lagesen K, Rognes T, et al. Genome dynamics in major bacterial pathogens. FEMS Microbiol. Rev. [Internet]. 2009 [cited 2017 May 6];33:453–70. Available from: http://www.ncbi.nlm.nih.gov/pubmed/19396949