Models and Algorithms for Metagenomics Analysis and Plasmid Classification

Anuradha Wickramarachchi

A thesis submitted for the degree of Doctor of Philosophy

School of Computing
The Australian National University

February, 2022

Declaration

This thesis presents research undertaken between August 2018 and February 2022 at the School of Computing, The Australian National University, Canberra, Australia.

The work presented in this thesis is that of the candidate alone under the supervision of Dr Yu Lin, except where indicated. None work in this thesis has been submitted in whole or in part for any other degree at this or any other university. This thesis is based on several manuscripts, peer reviewed journals/publications and online archives outlines below.

- Chapter 3 discusses the work published as the paper titled "MetaBCC-LR: Metagenomics Binning by Coverage and Composition for Long Leads" (Wickramarachchi, Mallawaarachchi, Rajan et al., 2020a).
- Chapter 4 discusses the work published as the paper titled "RBinner: Binning Long Reads in Metagenomics Datasets" (Wickramarachchi and Lin, 2021a). An extended version is published in BMC Algorithms for Molecular Biology titled "Binning long reads in metagenomics datasets using composition and coverage information" (Wickramarachchi and Lin, 2022a).
- Chapter 5 discusses the work published as the paper titled "Metagenomics Binning of Long Reads Using Read-Overlap Graphs" (Wickramarachchi and Lin, 2022b).
- Chapter 6 discusses the work presented at GLBIO 2021 (`http://iscb.org/cms_addon/conferences/glbio2021/tracks/General`) published as a preprint titled "PlasLR Enables Adaptation of Plasmid Prediction for Error-Prone Long Reads" (Wickramarachchi, Mallawaarachchi, Pu et al., 2021b).
- Chapter 7 discusses the work published as the paper titled "GraphPlas: Refined Classification of Plasmid Sequences using Assembly Graphs" (Wickramarachchi and Lin, 2021).

Anuradha Wickramarachchi
30 July 2022

Acknowledgements

I acknowledge and celebrate the First Australians on whose traditional lands we meet, and pay my respect to the elders past, present and emerging.

I sincerely thank Dr Yu Lin from the Australian National University (ANU) for his tremendous support, encouragement and understanding throughout the period of my PhD. His role as the panel chair and primary supervisor has been invaluable and phenomenal. Thank you, Dr Hardip Patel and Dr Zhenchang Xing for being my panel members, evaluating student milestones and providing valuable feedback. Furthermore, I also thank ANU administration who facilitated a great study environment. Moreover, I pay my gratitude towards external collaborators Dr Vaibhav Rajan from NUS, Singapore and Dr Lianrong Pu from Tel Aviv University, Israel for being knowledgeable collaborators and advisors in conducting impactful research.

I would like to acknowledge the motivation provided by Prof Indika Perera and Dr Dulani Meedeniya from University of Moratuwa (UoM), Sri Lanka which encouraged me to pursue my studies at ANU, Australia. Thank you all the lovely members of the Department of Computer Science and Engineering at UoM for all the support you offered.

My PhD has been possible because of the completely free education I had in Sri Lanka funded by people of Sri Lanka. Hence, I pay my respect to all the taxpayers in Sri Lanka, who made the free education a reality. I also thank ANU, Australia for fully funding my PhD including the tuition fee and living expenses, which made my studies possible.

Most importantly, I pay my lifelong gratitude towards my loving parents Sanath and Pradeepa Wickramarachchi, and my brother Dilshan Wickramarachchi for the tremendous motivation and support. I thank my wife, Vijini Mallawaarachchi for her motivation, love and courage during the past four years which helped me succeed.

Thank you!

Anuradha

Abstract

Metagenomics studies have provided key insights into the composition and structure of microbial communities found in different environments. Among the techniques used to analyze metagenomics data, binning is considered a crucial step to characterize the different species of microorganisms present. Metagenomics binning can be extended further towards determination of plasmids and chromosomes to study environmental adaptations. The field of metagenomics binning is mostly done on contigs from genome assemblies. Metagenomics studies are mostly performed with short read sequencing. Direct binning of short reads suffers from insufficient species-specific signal, thus they are usually assembled into longer contigs before binning. Therefore, the emergence of long-read sequencing technologies gives us the opportunity to study the binning of long reads directly, where such studies have been carried out in limited numbers.

Firstly, this thesis presents the challenges in binning long reads compared to contigs assembled from short reads. One key challenge in binning long reads is the absence of coverage information, which is typically obtained from assembly. Moreover, the scale of long reads compared to contigs demands more computationally efficient methods for binning. Therefore, we develop MetaBCC-LR to address these challenges and perform metagenomics binning of long reads. We introduce the concept of k-mer coverage histogram to estimate the coverage of long reads without alignments and use a sampling strategy to handle the immense number of long reads. Since MetaBCC-LR is limited by the use of coverage and composition information in a stepwise manner, we further develop LRBinner to combine the coverage and composition information. This enables LRBinner to effectively combine coverage and composition features and use them simultaneously for binning. LRBinner also implemented a novel clustering algorithm that performs better on binning long-read datasets from species with varying abundances. Moreover, we propose OBLR to improve the coverage estimation of long reads via a read-overlap graph instead of k-mers. The read-overlap graph also enables OBLR to perform probabilistic sampling to better recover low-abundant species. Secondly, we investigate opportunities to improve plasmid detection which is considered as a binary plasmid-chromosome classification problem. We introduce PlasLR that enables adaptation of plasmid prediction tools designed for contigs to classify long and error-prone reads. We also develop GraphPlas that uses the assembly graph to improve plasmid classification results for assembled contigs. In summary, this thesis presents the progressive development of models and algorithms for metagenomics binning and plasmid classification.

Keywords: metagenomics, binning, clustering, algorithms, classification

Contents

Figures

Tables

Abbreviations

ADAM	A Method for Stochastic Optimization
AMBER	Assessment of Metagenome BinnERs
ANU	Australian National University
BLAST	Basic Local Alignment Search
BWT	Burrows-Wheeler transform
GraphPlas	Graph based Plasmid recovery
HMP	Human Microbiome Project
LAST	Local Alignment Search
LCA	Least Common Ancestor
LRBinner	Long Read Binner
MetaBCC-LR	Metagenomics Binning using Coverage and Composition for Long Reads
NCBI	National Center for Biotechnology Information
NGS	Next Generation Sequencing
NIH	National Institutes of Health
NUS	National University of Singapore
OBLR	Overlap Based Long Read binning
ONT	Oxford Nanopore Technologies
PCA	Principal Component Analysis
PlasLR	Plasmid recovery using Long Reads
t-SNE	t-distributed Stochastic Neighbor Embedding
TGS	Third Generation Sequencing
UMAP	Uniform Manifold Approximation and Projection

To my parents, brother, wife and teachers!

Chapter 1

Introduction

In this chapter, I present a high level introduction to the background and motivation of my research. Contents of this chapter are specialized to introduce the thesis' research field, metagenomics analysis and plasmid recovery. Firstly, I discuss different sequencing technologies and their implications. Secondly, I discuss the area of research metagenomics binning and approaches. Finally, I discuss the area of plasmid recovery and its importance.

1.1 Background

1.1.1 Genome sequencing

A huge leap in science was made when the DNA was discovered for the first time in the late twentieth century. Followed by this, genome sequencing arose giving us the capability to read the DNA molecule chain which would later evolve to characterize functions of living tissues from different origins. Genome sequencing has evolved passing several generations, in each, improving the existing state of the art. These advancements have enabled the exploration of the environment and applications of genomics facilitating the study of genes, microbiomes, and supporting disease diagnosis. Recent emerging studies include gene editing, targeted medicine, energy/biofuels and sustainable agriculture.

Shotgun sequencing

Shotgun sequencing was an early technique of sequencing where chunks of DNA material were extracted, hence the term "Shotgun". Traditionally, the exploration of the microbial content of a sample was performed using culture based methods. Culturing a given microbial sample imposes amplification biases due to various stressors and competing factors between different microorganisms present in the samples. This often means the resulting culture hardly depict the actual composition of microorganisms that originally existed. Shotgun sequencing facilitates the discovery of what microorganisms are there and what metabolic processes they carry out in the community (Segata et al., 2013). Culturing was often a necessity in order to yield sufficient amount of genetic material for scientists to work with. This was a laborious task that was expensive and ineffective. Later high throughput sequencing was invented to address these limitations.

High throughput sequencing

High throughput sequencing facilitates the scientists to sequence a large amount of genetic material in a single run from a given sample. Moreover, labor-intensive and time-consuming tasks such as lab culturing was rendered obsolete. This mode of sequencing could produce numerous sequencing reads by performing the genome sequencing in massively parallel setups (Poinar et al., 2006). Illumina is the most popular high throughput sequencing technology and produces genomic reads of length 100-300 base pairs. Since lab culturing was totally eliminated, scientists are able to explore the true nature of a given environment. Hence, the field of metagenomics was originated.

1.1.2 Metagenomics

What is metagenomics analysis?

Metagenomics is the study of microbial genetic material obtained directly from the environment (Chen and Pachter, 2005). Metagenomics analysis could be performed on various environments including human gut, animal gut, soil, atmosphere, biofilms, skin, etc. Metagenomics has expanded significantly resulting in massive projects such as the Human Microbiome Project (HMP) which explored the human microbiome (The Human Microbiome Project Consortium, 2012). Such broad-spread studies have made metagenomics popular while providing novel insights into different environments.

What is metagenomics binning?

Metagenomics analyses intend to answer the basic questions; "Who is there?", "What are they doing?" and "What is new?" (Nealson, 1997). "Who is there?" questions the composition of a given microbial sample and tries to quantify the sample in terms of number of taxonomic groups present and the relative abundance of each taxa (genetic material). "What are they doing?" questions the functional aspects of the microbial content such as host interactions, environmental adaptations, symbiotic relationships, etc. Finally, the question"What is new?" tries to decipher the characteristics unique to a given sample in that particular host environment.

The first and foremost question; "Who is there?" is answered computationally using the process *Metagenomics binning*. Metagenomics binning quantifies the number of taxonomic groups in a given microbial sample. Metagenomics binning facilitates the separation of genetic material into different groups where separate analysis can be carried out deeper into the genes, etc. This is an important analysis as it facilitates the study of microbial interaction, study of relative abundance, time-series analysis of a sample, etc.

Motivation

Metagenomics study is important in various fields of study including fighting and understanding implications of climate change (Long et al., 2016), designing of biofuels (Pabbathi et al., 2021), study food products and understanding their impact of human gut (De Filippis et al., 2017), healthcare in drug design and personalized medicine (Virgin and Todd, 2011).

1.1.3 Plasmid Recovery

What are plasmids?

Plasmids are extra-chromosomal genetic elements which allow their hosts to rapidly adapt and survive under changing environmental conditions (Smalla et al., 2015; Thomas and Nielsen, 2005). These small and widespread genetic elements can be commonly found in bacterial cells (Rodriguez-Beltran et al., 2018). Plasmids are responsible for a significant amount of genetic variation within populations which ensures persistence towards changes in the environment and various selective pressures. Plasmids also play an important role in horizontal gene transfer among unrelated bacterial species to spread genes related to virulence, resistance to different classes of antibiotics and heavy metal resistance (Carattoli, 2013; Li, Li et al., 2015; Smalla et al., 2015). Hence, it is important to identify and study plasmids present in the environmental samples (Li, Li et al., 2015). High-throughput sequencing such as next-generation sequencing (NGS) and third-generation sequencing (TGS) technologies have allowed us to sequence and analyze bacterial genomes including their plasmids (Forster et al., 2019; Margos et al., 2017).

What is plasmid recovery?

Plasmid recovery addresses the problem of detecting the available plasmid sequences in a given dataset. This is a challenging task as composition of plasmids and the chromosomes of a given species follows a very close pattern. Furthermore, plasmids are significantly shorter compared to other genomic sequences. Hence, plasmids contribute to a minute quantity of relative genetic material. Hence, the recovery of plasmids is a challenging task in the study of different microorganisms. Moreover, genome assembly can be negatively affected by the plasmid sequences making it difficult to obtain complete genomes. Hence, plasmid recovery has become an area of attention among the assembly researchers.

Motivation

Plasmid recovery is an important area of study in metagenomics. Despite the low relative genetic content in plasmids, they support environmental adaptations of microorganisms, especially in bacteria, where antibiotic resistance plays a key role in modern society (Levy et al., 1976). Hence, accurate determination of functional aspects of plasmids is AN important area of research. Plasmids are also of interest, among researchers in design of drugs and gene therapy as a method of DNA delivery in different treatments (Gill et al., 2009). Furthermore, plasmids are of interest among gene editing researchers due to the functional importance of plasmids among microorganisms.

1.2 Thesis outline

This section outlines the chapters in the rest of the thesis.

Chapter 1

This chapter provides an overview into the subject areas discussed in this thesis; metagenomics binning and plasmid recovery.

Chapter 2

This chapter presents the relevant literature in the fields of metagenomics binning and plasmid recovery.

Chapter 3

This chapter presents the development of MetaBCC-LR, which was the first standalone tool to utilize both coverage and composition to support binning of long reads.

Chapter 4

This chapter presents the development of LRBinner, which brings in deep learning techniques and a novel clustering algorithm to support binning of long reads by addressing challenges and limitations of MetaBCC-LR.

Chapter 5

This chapter introduces the concept of read-overlap graph with graph representation learning techniques to support binning of long reads.

Chapter 6

This chapter presents the development of PlasLR which facilitates plasmid recovery at reads level, thus providing capabilities for improved genome assemblies.

Chapter 7

This chapter presents the development of GraphPlas which facilitates improvements in plasmid recovery by incorporating assembly graph and coverage information to existing reference based plasmid recovery tools.

Chapter 8

This chapter outlines the conclusions and future work of the research presented in this thesis.

Chapter 2

Literature Review

This chapter discusses the relevant literature in genome sequencing, metagenomics binning and plasmid recovery. More specifically, this chapter elaborates on the evolution of genome sequencing technologies followed by the advancements in metagenomics binning and plasmid recovery. Genome sequencing and genome assembly has played a key role in facilitating the study of microbiome of environments skipping the traditional culturing techniques, resulting in the field of metagenomics. Starting from Sanger sequencing, the technologies have gone through two major milestones providing short reads followed by the era of long reads. Both short read sequencing and long read sequencing are equally popular and have supported developments in metagenomics analysis. Please refer to paper titled "Sequencing technologies and genome sequencing" (Pareek et al., 2011) for an in-depth literature review on genome sequencing technologies. Please refer to paper titled "Comparison of long-read sequencing technologies" (De Maio et al., 2019) for comparison of different assemblers and their performance in long reads assembly. Furthermore, please refer to papers from CAMI challenge (Meyer, Lesker et al., 2021; Yue et al., 2020) for an in-depth literature in metagenomics binners in general and their performance.

2.1 Genome Sequencing and Assembly

Genome sequencing is the process where genetic material is captured in machine-readable characters, *i.e.*, ASCII coded DNA characters. DNA is captured in a highly fragmented form of strings called reads which need to go through a process called genome assembly. The genome assembly process attempts to generate a complete and continuous sequence that represents the origin of the fragmented DNA strings.

2.1.1 First Generation Sequencing

Genome sequencing started in 1977 with the introduction of Sanger sequencing by Frederick Sanger and colleagues. This approach involved a lot of manual labor and was very expensive. However, the technique is still in prevalence where verification of sequencing results is needed for precision and/or to obtain more accurate genome assemblies (DiGuistini et al., 2009). Further advancements in sequencing resulted in the short read sequencing which produces genomic reads of lengths in range 100 - 300 base pairs, resulting in the Next Generation Sequencing era.

2.1.2 Next Generation Sequencing

Next Generation Sequencing technologies include mainly 454 pyrosequencing (Hyman, 1988) and Illumina (Solexa) sequencing (Canard and Sarfati, 1994a; Canard and Sarfati, 1994b) where Illumina has been the most popular sequencing method in metagenomics studies. These technologies produce read lengths that vary from 100 to 300 base pairs and has a very low error rate. More specifically, Illumina technology has been used in the Human Microbiome Project (HMP) by United States National Institute of Health (NIH) (The Human Microbiome Project Consortium, 2012; Turnbaugh et al., 2007). Illumina reads are highly accurate with error rates from 0.1%-1% (Stoler and Nekrutenko, 2021). They were also capable in producing millions of short reads on a single run and was relatively cheaper. This technology has been extensively applied in the study of microbial datasets (Kang, Froula et al., 2015; Mallawaarachchi and Lin, 2022; Mallawaarachchi, Wickramarachchi et al., 2020a; Mallawaarachchi, Wickramarachchi et al., 2020b; Nissen et al., 2021; Wu, Tang et al., 2014) and in plasmid related studies (Antipov, Hartwick et al., 2016; Krawczyk et al., 2018; Lanza et al., 2014; Pellow, Mizrahi et al., 2020).

2.1.3 Third Generation Sequencing

Third Generation Sequencing (TGS) which is also known as long read sequencing is the successor of short read sequencing (Bleidorn, 2016). Pacific BioSciences and Oxford Nanopore are the most prominent figures in the manufacturing of long read sequencing machines. These sequencers are capable of producing reads from 1000 base pairs to several hundred kilo base pairs in length, at a greater error rate compared to short read sequencing (Gupta, 2008). Thanks to the longer read length, these technologies have been massively helpful in the resolution of complex structures and variants in genome assemblies (Tham et al., 2020). Furthermore, long read sequencing has been used in metagenomics studies due to its ability to provide more complete assemblies and being able to accurately classify in to taxonomic groups (Albrecht et al., 2020; Ekim et al., 2021; Feng et al., 2021; Huson, Albrecht et al., 2018; Menzel et al., 2016; Wood, Lu et al., 2019). In general, the error rates can vary from 1%-20% depending on the sequencing length and the specific technology used (*i.e.*, CCS, HiFi, R10, etc).

2.1.4 Genome Assembly

Genome assembly is the following process to genome sequencing. The genome assembly collapse sequencing reads to form longer sequences that has a higher base accuracy (*i.e.*, low error rate compared to input reads). These longer and accurate sequences are called contigs. Contigs have the coverage attribute, which can be identified as the average number of read bases that support the contig. This is obtained computationally by $\frac{\textit{total overlap length}}{\textit{contig length}}$. Abundance is a similar metric, defined as the $\frac{\textit{total contig length of a species}}{\textit{total contig length of the dataset}}$. Understandable, abundance is more relevant in metagenomics, where studies are carried out in samples of many species. Hence, abundance is a relative measure of genetic content. This is affected by genome length, but coverage has the ability to discriminate species despite their genome lengths. The process of genome assembly uses data structures such as overlap-graphs (Koren et al., 2017; Simpson and Durbin, 2012) (read overlaps are created followed by collapsing paths in the overlap graph) and de Bruijn graphs (Bankevich et al., 2012; Kolmogorov, Yuan et al., 2019) (k-mers are used to build a graph and non-branching paths are resolved to form contigs.).

2.2 Metagenomics Binning

Approaches for metagenomics binning has changed over time with the advancements in sequencing. In early metagenomics studies, short reads were assembled using a genome assembler prior to the binning process (*e.g.*, Spades (Bankevich et al., 2012) and SGA (Simpson and Durbin, 2012)). However, metagenomics specific assembly started to arise with the popularity of metagenomics studies. Such assemblers include metaSPAdes (Nurk et al., 2017) and MEGAHIT (Li, Liu et al., 2015) for short read metagenomics assembly and metaFlye (Kolmogorov, Rayko et al., 2019), hifiasm-meta (Feng et al., 2021) and rust-mdbg (Ekim et al., 2021) for long read metagenomic assembly. Such assemblers are capable of producing more complete assemblies by considering uneven coverage that arise due to varying coverages of species in metagenomics samples.

Assembly of TGS reads from metagenomic datasets produces longer contigs that are accurate to perform multiple downstream analyses. Followed by assembly of the metagenomics datasets clustering operations are performed to discover distinct species in the sample. In some cases, reference based approaches (*i.e.*, classification based on taxonomy) are used to discover the species present. This is because, shorter contigs can have greater ambiguity due to having shared regions between species (affecting reference based approaches) and poor genomic signals to extract features (reference free methods require genomic signatures such as oligonucleotide frequencies to perform binning). The overall process of binning contigs is illustrated in Figure 2.1.

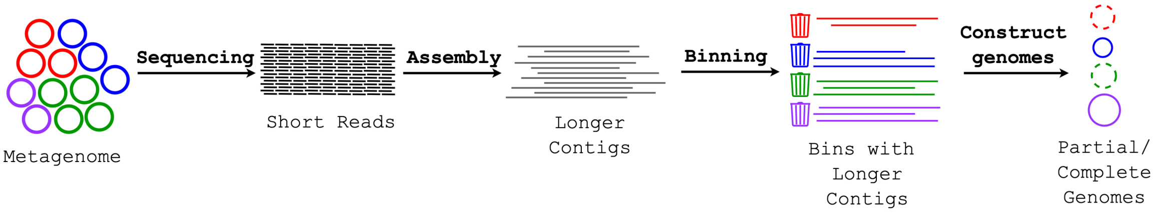


Figure 2.1: The process of binning contigs using the short reads as input followed by genome assembly. The resulting contigs are used to perform the actual binning operation.

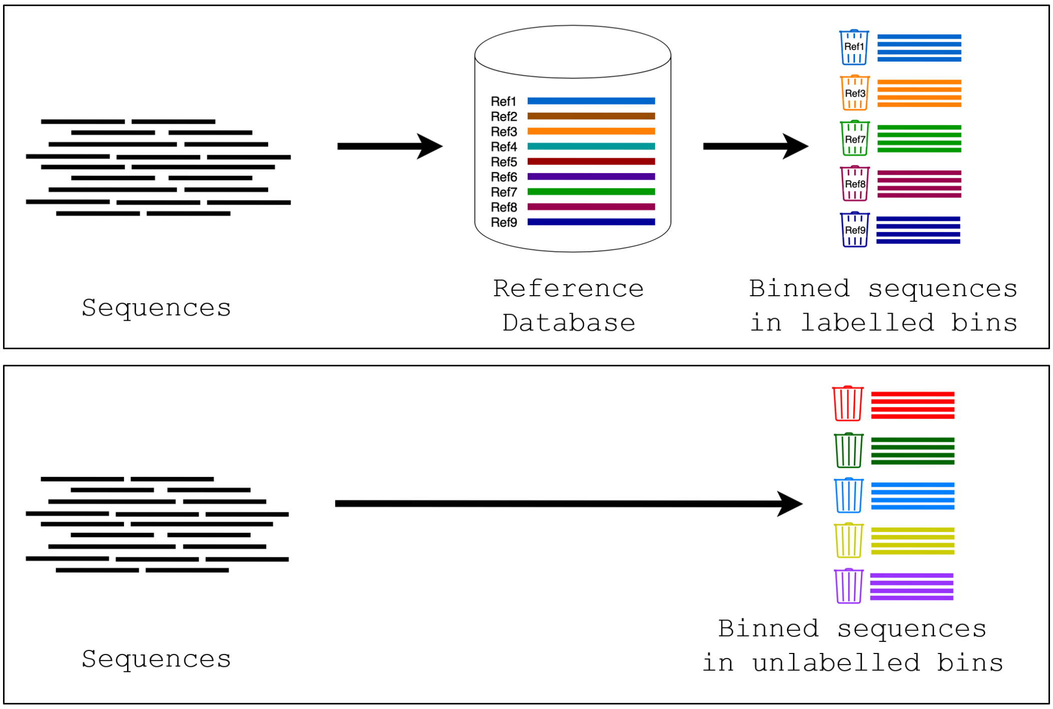


Figure 2.2: Illustration of reference based binning (top) and reference free binning (bottom).

2.2.1 Existing Approaches for Metagenomics Binning

There are two main approaches for metagenomics binning of sequences (contigs or reads), (1) reference based approach and (2) reference free approach as illustrated in Figure 2.2.

Reference Based Binning

In reference based binning, the sequences are classified using a database or an index that contains the reference sequence information. Table 2.1 presents details of most popular reference based binning tools from both contig binning and read binning paradigms.

Table 2.1: Comparison of reference based binning tools.

Tool	Application	Database	Method
MEGAN (Huson, Auch et al., 2007)	Contigs, Short reads	Nucleotide Database	Search using BLAST/BLASTX followed by LCA filtering
Kraken (Wood and Salzberg, 2014)	Contigs, Short reads, Long reads	k-mer index	Exact alignment of k-mers
CLARK (Ounit et al., 2015)	Short reads	k-mer index for target sequences	Clasasify using target specific k-mers
Centrifuge (Kim et al., 2016)	Contigs, Short reads	Index of non shared regions in references	BWT and FM index based search
Kaiju (Menzel et al., 2016)	Contigs, Short reads	Protein-coding gene database	Protein-level matches with BWT
MEGAN-LR (Huson, Albrecht et al., 2018)	Long reads	NCBI-nr protein reference database	Perform LAST search followed by LCA filtering
Kraken2 (Wood, Lu et al., 2019)	Contigs, Short reads, Long reads	k-mer index	Minimizer matches for k-mers

Almost all the tools are designed to handle short reads and their assembled contigs except for MEGAN-LR (Huson, Albrecht et al., 2018) which is the long read version of MEGAN (Huson, Auch et al., 2007). These tools are capable of performing in a competitive manner in the classification of metagenomic reads and contigs. MEGAN, MEGAN-LR and Kaiju (Menzel et al., 2016) perform reference search in indexes in the protein space while Kraken (Wood and Salzberg, 2014), Kraken2 (Wood, Lu et al., 2019), Centrifuge (Kim et al., 2016) and CLARK (Ounit et al., 2015) perform reference based binning using the nucleotide references.

Reference based tools are still challenged by the error rate in long reads, but can be expected to improve as the sequencing technologies tend to produce more accurate long reads with time. The key limitation in the reference based approach is the need to have a complete database of references for effective application in analysis. For example, Kraken2 (Wood, Lu et al., 2019), a very popular reference based binning tool, has multiple databases hosted by different researchers which makes the choice of database a significant challenge.

Reference free metagenomics binning approaches are often the first choice in metagenomics analysis due to the capability of *de-novo* metagenomics binning without having to know the underlying composition. Such tools often consume far less computational resources, especially in terms of peak memory and CPU time (Kim et al., 2016).

Reference Free Binning

Reference free binning aims to perform metagenomics binning without the knowledge of pre-existing reference genomes. Features such as coverage, composition and single-copy marker genes are often considered. Coverage is defined as the average times a given genomic region is represented by the sequencing reads. Typically, coverage can help metagenomics binning using the relative population of species in a sample. Composition is often computed using genomic signatures such as oligonucleotide frequencies. This is achieved via representing a given genomic sequence as a vector representing the normalized counts of all the possible short substrings of a given length (*i.e.*, *k*-mers or sub-strings of length *k*), usually of size 3, 4 or 5 (3-mers, 4-mers or 5-mers). Single-copy marker genes are the genes that appear just once and at least once in all the bacteria, archaea or fungi signifying important common functionality in the species. Note that, marker genes are obtained from a tool such as Prodigal (Hyatt et al., 2010) where the input sequences are scanned for genes that are known to appear *only once*, in all bacterial, archaeal or fungal genomes (Wu, Simmons et al., 2015). Thus, the number of occurrences of a given marker gene in an assembly indicates the number of species which the marker gene belongs, (*i.e.*, bacterial, archaeal or fungal). Table 2.2 tabulate information relevant to popular reference free binning tools.

Table 2.2: Comparison of reference free binning tools.

Tool	Application	Features	Method
MetaWATT (Strous et al., 2012)	Contigs	Coverage, Composition	Splitting contigs into bins starting from the longest
Maxbin 2.0 (Wu, Simmons et al., 2015)	Contigs	Coverage, Composition, Marker genes	Detect bins using marker genes, cluster using coverage and composition
MetaBAT (Kang, Froula et al., 2015)	Contigs	Coverage, Composition	Bin using pairwise distances and precomputed probabilistic models
VizBin (Laczny, Sternal et al., 2015)	Contigs	Coverage, Composition	Dimension reduction using BH-SNE and human augmented binning
MetaProb (Girotto et al., 2016)	Short reads, Long reads	Compositions	Read grouping and clustering using k-mer frequency probabilities
BusyBeeWeb (Laczny, Kiefer et al., 2017)	Contigs, Long reads	Composition	Dimension reduction using BH-tSNE (Web App)
SolidBin (Wang, Wang, Lu et al., 2019)	Contigs	Coverage, Composition, Marker genes	Normalized cut on marker gene-constraints and feature similarity
VAMB (Nissen et al., 2021)	Contigs	Coverage, Composition	Dimensionality reduction by auto-encoder, clustering using an iterative method
SemiBin (Pan et al., 2021)	Contigs	Coverage, Composition, Marker genes	Dimensionality reduction by auto-encoder, clustering the affinity graph and marker gene constraints
RepBin (Xue et al., 2021)	Contigs	Marker genes, Assembly Graph	Graph representation learning based on marker gene constraints.
MetaCoAG (Mallawaarachchi and Lin, 2022)	Contigs	Coverage, Composition, Marker genes, Assembly graph	Graph partitioning using features

Coverage and composition are among the most common features used for reference free binning (MetaWATT (Strous et al., 2012), VAMB (Nissen et al., 2021), VizBin (Laczny, Sternal et al., 2015), BusyBeeWeb (Laczny, Kiefer et al., 2017), MetaProb (Girotto et al., 2016), AbundanceBin (Wu and Ye, 2011)). However, marker genes have also been used in some tools to build constraints between contigs and to detect the total number of bins (MaxBin 2.0 (Wu, Simmons et al., 2015), SolidBin (Wang, Wang, Lu et al., 2019) and SemiBin (Pan et al., 2021)). Assembly graph was considered in RepBin (Xue et al., 2021) and MetaCoAG (Mallawaarachchi and Lin, 2022) on top of all other features available to perform binning.

All above tools require contigs as the input data source except for BusyBeeWeb and MetaProb. This is because, no coverage information is available for long reads, thus fits the criterion of composition based binning. Moreover, short reads binning is not available in the reference free form except in MetaProb and AbundanceBin because consistent genomic signatures require longer sequences, 1000 base pairs or more (Wu, Simmons et al., 2015). Note that MetaProb groups short reads to obtain more accurate genomic signature. Out of the tools above, only MetaProb and BusyBeeWeb were capable of handling long reads. MetaProb can only handle small long-read datasets and do not scale reasonably well for millions of long reads (Wickramarachchi, Mallawaarachchi, Rajan et al., 2020a). Furthermore, due to the web implementation of BusyBeeWeb and the use of t-SNE (Van der Maaten and Hinton, 2008) dimension reduction, it has limited input size and do not scale for millions of long reads (Wickramarachchi, Mallawaarachchi, Rajan et al., 2020a).

2.2.2 Metagenomics Binning of Long Reads

Metagenomics binning of long reads facilitates the analysis of long reads as soon as they become available from the sequencers. Furthermore, binning prior to assembly or other evaluation has the potential to improve downstream analysis by providing insightful information into datasets such as number of species, separation of reads into bins and individual bin assembly for a more isolated genome analysis.

Although approaches has been studied to perform metagenomics binning of long reads, such efforts are either supported through reference based methods (METGAN-LR, Kraken and Kraken2) or provided as an additional feature apart from contig binning (MetaProb, BusyBeeWeb). Furthermore, the existing *de-novo* long read binners do not consider coverage information due to the unavailability of coverage information for any kind of raw reads. Hence all the *de-novo* long read binning methods rely totally on composition information. This makes it difficult to distinguish reads originating from similar but distinct species, whereas, using coverage could help separate such reads due to varying underlying coverage of their species.

This thesis explores models and algorithms to design efficient and effective metagenomics binning tools targeting long reads. Refer to chapters 3, 4 and 5.

2.3 Plasmid Recovery

Plasmids are important genetic material that govern environmental adaptations of microorganisms. There have been several studies aiming to develop methods and algorithms that perform plasmid recovery using both the existing knowledge of reference genomes (Zhou and Xu, 2010a) and de-novo methods (Rozov et al., 2017).

2.3.1 Existing Approaches for Plasmid Recovery

Reference Based Approaches

Reference based methods search the input sequences against a database containing plasmid information to discover plasmids. Hence, all the reference based binning approaches in Table 2.1 can be used for this task, given that the database used consists of plasmid-chromosome level taxonomic information (Gomi et al., 2021). However, plasmid recovery often takes a unique approach where features of known plasmids and chromosomes are used in conjunction with machine learning techniques to model the situation as a binary classification problem.

Table 2.3 summarizes popular plasmid recovery tools. cBar (Zhou and Xu, 2010b), mlplasmids (Arredondo-Alonso et al., 2018), PlasFlow (Krawczyk et al., 2018) and PlasClass (Pellow, Mizrahi et al., 2020) use oligonucleotide frequencies from reference plasmids and chromosomes to train machine learning models to classify unseen sequences. PlaScope (Royer et al., 2018), PlasmidFinder (Carattoli et al., 2014) and Platon (Schwengers et al., 2020) uses databases and BLAST search to perform plasmid discovery. A key similarity in all these approaches is that they can only use composition information. This is because, reference databases can only offer composition information. However, plasmids typically occur in multiples of their chromosomes called copy numbers (Friehs, 2004). This gives a significant variation of coverage between chromosomes and associated plasmids. However, such information cannot be used in machine learning models or BLAST databases as this information is dataset specific.

Table 2.3: Reference based plasmid recovery tools.

Tool	Application	Coverage	Composition	Method
cBar (Zhou and Xu, 2010b)	Contigs	✗	✓	Decision tree, Bayes network, SVM, SMO, nearest neighbor
PlasmidFinder (Carattoli et al., 2014)	Contigs	✗	✓	Similarity comparison with a database of replicon sequences
mlplasmids (Arredondo-Alonso et al., 2018)	Contigs	✗	✓	Logistic regression, Bayesian classifier, decision trees, RF, SVM
PlasFlow (Krawczyk et al., 2018)	Contigs	✗	✓	Neural networks
PlaScope (Royer et al., 2018)	Contigs	✗	✓	Classification on top of Centrifuge (Kim et al., 2016)
PlasClass (Pellow, Mizrahi et al., 2020)	Contigs	✗	✓	Logistic regression
Platon (Schwengers et al., 2020)	Contigs	✗	✓	BLAST search for plasmid-borne genes

Assembly Based Approaches

Assembly based approaches try to separate plasmids and chromosomes by observing plasmid specific structures in genomes assembly graphs. Table 2.4 summarizes the popular assembly based plasmid recovery tools. All the tools either perform on top of assembly graph or perform plasmid resolution during the assembly process. However, in this approach only circular contigs can be extracted given sufficient connectivity exists. Moreover, when plasmids are non-circular, other means of classification is necessary to extract plasmids (Casali and Preston, 2003). Interestingly, the only features used in these approaches are the coverage, calculated in assembly and the overlap information. Genomics signatures such as oligonucleotide frequencies are not considered.

Table 2.4: Assembly based plasmid recovery tools.

Tool	Method
Recycler (Rozov et al., 2016)	Extracting circular contigs using assembly graph
metaplasmidSPAdes (Antipov, Hartwick et al. (2016), Antipov, Raiko et al. (2019))	SPAdes assembler process with circular contig resolution
SCAPP (Pellow, Zorea et al., 2021)	Uniform coverage cycle peeling from assembly graph

2.3.2 Limitations in Plasmid Recovery

Plasmid recovery is often performed using information that is either available on reference genomes (*i.e.*, composition information such as oligonucleotide frequencies or genes) or that is solely available through the data (*i.e.*, assembly graph and coverage). There has been little effort in discovery of methods that exploit both forms of information to improve the classifications. For example, reference based methods can be used to classify plasmids and chromosomes, followed by refinement through dataset specific information such as coverage. Furthermore, assembly graph can be used to determine the class of contigs (plasmid or chromosome) in cases where the reference based predictions are poor. Moreover, such combination of methods can further expand the scope of existing methods by removing the limitations while incorporating more useful information otherwise overlooked. In chapters 6 and 7 we explore such avenues of improvements by incorporating dataset specific information into reference based plasmid recovery methods.

Chapter 3

Metagenomics Binning of Long Reads

The work presented in this chapter was published as

A. Wickramarachchi, V. Mallawaarachchi, V. Rajan et al. (2020a). 'MetaBCC-LR: Metagenomics Binning by Coverage and Composition for Long Leads'. *Bioinformatics*, 36(Supplement_1), pp. i3–i11

This piece of work was presented at the annual International Conference on Intelligent Systems for Molecular Biology (ISMB) 2020.
`https://www.iscb.org/cms_addon/conferences/ismb2020/tracks/microbiomecosi`

The software is freely available at `https://github.com/anuradhawick/MetaBCC-LR`.

3.1 Overview and Motivation

Metagenomics binning is benefitted by the use of both coverage and composition over complete reliance on composition information. In this chapter we identify the challenges exist in binning long reads right away and address them accordingly to develop MetaBCC-LR, a standalone tool for metagenomics binning of long reads using coverage and composition.

3.1.1 Binning contigs vs binning long reads

There are several challenges in binning long reads compared to contigs. Binning long reads requires us to handle unprocessed sequencing reads in contrast to contigs binning. Contigs are obtained from the assembly process, where the reads are collapsed based on overlaps to form more accurate consensus sequences. Hence, more information is available for contigs compared to sequencing reads. Table 3.1 summarizes the major differences between binning contigs and long reads.

Table 3.1: Differences between contigs binning and long read binning

Contigs binning	Long read binning
Availability of accurate coverage information	No coverage information is available by default
Relatively low in error rate	Higher error rate
Manageable number of contigs from assembly	Millions of sequencing reads
Many existing tools and analyses are feasible	Limited number of operations are possible

3.1.2 Challenges in Binning Long Reads

Absence of coverage information for long reads

One key piece of information used for binning contigs is the coverage information. Coverage information is available from the assembly process, or can be obtained by aligning sequencing reads to the contigs produced. As expected, the long sequencing reads possess no coverage information unlike contigs. A naïve approach to estimate coverage is the all-vs-all alignment of sequencing reads. However, such alignment is computationally expensive and results in massive alignment files for processing (Balvert et al., 2021).

Noise in long reads compared to contigs

A major distinction between contigs and the raw reads is the error rate. While sequencing reads possess error, the majority of errors are corrected in the process of genome assembly (Antipov, Hartwick et al., 2016; Bankevich et al., 2012; Kolmogorov, Bickhart, Behsaz, Gurevich, Rayko, Shin, Kuhn, Yuan, Polevikov, Smith et al., 2020; Nurk et al., 2017). Hence, more insightful explorations such as single copy marker genes (Wu, Simmons et al., 2015; Wu, Tang et al., 2014) or unique k-mers (Wood, Lu et al., 2019) can be studied to support binning.

Scale of long reads compared to contigs

Due to the nature of genome assembly, a contig may represent thousands of sequencing reads. Because of this, contigs are typically in the range of 1,000-100,000 base pairs in length. However, reads are in the range of millions. This demands, the long read binning to be scalable enough to handle massive scales of data.

Limited availability of analytical tools

The accuracy and length of contigs are far greater compared to sequencing reads of any form (long or short). Hence, complex analyses such as k-mer based binning are feasible (Wood, Lu et al., 2019). Furthermore, the assembly process ensures that duplicate sequences that may mislead marker gene analyses are minimized. For example, contigs can be studied for their marker genes to perform analyses (Hyatt et al., 2010; Rho et al., 2010; Zhu, Lomsadze et al., 2010). However, the premise does not hold true for long reads. Usually, we are required to assemble the reads before any analysis could be performed accurately, especially due to higher error rates.

In this chapter, we design and implement MetaBCC-LR, a scalable reference-free binning tool to bin metagenomic long reads. MetaBCC-LR uses discriminatory features, that capture differential species coverage and composition information to perform binning. In MetaBCC-LR we investigate the main three challenges that we have identified; (1) absence of coverage information for long reads, (2) higher error rate in long reads compared to contigs and (3) massive number of sequences that we need to handle.

3.2 Methods

The MetaBCC-LR pipeline takes the raw long reads as the input and outputs the bin annotations for each input read. The following sections describe how we address the identified challenges to implement the complete binning pipeline. Firstly, we propose a k-mer based coverage estimation technique. Secondly, we evaluate the impact of error, on long reads' composition information. Lastly, we implement a sampling strategy to manage the shear number of reads in the input datasets.

3.2.1 k-mer based estimation of coverage of a read's underlying species

Species found in metagenomic samples generally have varying abundances which would result in different sequencing coverages for the corresponding genomes. This is an important discriminatory factor in cases where species exist with similar composition but varies in terms of their coverage. Hence, a sophisticated technique to estimate coverage for a given long read is required as all-vs-all pairwise alignment approaches could be resource intensive.

In order to address this challenge we propose the "k-mer coverage histogram". We estimate the coverage of each read by breaking down the reads into long substrings called k-mers. k-mer stands for a substring of length k. We achieve this by (1) computing k-mer counts for the entire dataset followed by (2) obtaining k-mer count histograms for reads, using each read's k-mers and the counts from (1).

k-mer counts for an entire dataset

k-mers are of two kinds, "genomic" and "non-genomic". A *k*-mer is considered as *genomic* if it appears in at least one genome in the metagenomic dataset, otherwise it is considered as *non-genomic*. Given a metagenomic dataset, the *coverage* of a *k*-mer is defined as the number of times this *k*-mer appears in the entire dataset. Naturally, it is expected that the number of genomic *k*-mers is greater than the number of non-genomic ones.

Typically, *k*-mer counting can be performed using a tool such as DSK efficiently (Rizk et al., 2013). However, such computations carry the burden of heavy disk I/O where *k*-mer counts output file needs to be parsed and re-indexed in-memory for the next step. Hence, we implement a simple and effective *k*-mer counter similar to previous studies (Kolmogorov, Yuan et al., 2019; Lin, Yuan et al., 2016), In MetaBCC-LR we use *k*=15. We borrow the idea of "compare and swap" atomic instruction from Jellyfish (Marcais and Kingsford, 2012) counter. Initially, DNA alphabet "A, C, G and T" are encoded in to 2 bit digits where the length 15 can be encoded in 30 bits without overflow in a 64 bit machine. Here, the DNA alphabet is individually encoded as A=00, C=01, T=10 and G=11. Instead of using an actual hash table, we initialize an array of size 4^{15} unsigned 32-bit integers. This ensures that our bit encoded 15-mers are accessed with $O(1)$ complexity. Moreover, the entire hash table is guaranteed to consume no more than 4 GB of memory. The choice *k* for coverage estimation is further elaborated in Appendix A.

Building k-mer coverage histogram for a single read

The *k*-mer coverage histogram represents the coverage of a given read. In order to compute the coverage histogram for a single read, we obtain all 15-mers for the selected read. Now the counts for each of these 15-mers are obtained from the hash table we generated for the entire dataset. Naturally, this histogram represents the prominence of 15-mers from a given read in the whole dataset. Based on the varying length of reads, the 15-mers counts can deviate even within the same species. Hence, we normalize the histograms using the total number of 15-mers in the read. In order to formalize the representation, we denote this feature vector by $V_{coverage}$. Furthermore, in our experiments, we set the histogram properties as bin_width=10 and bin_count=32.

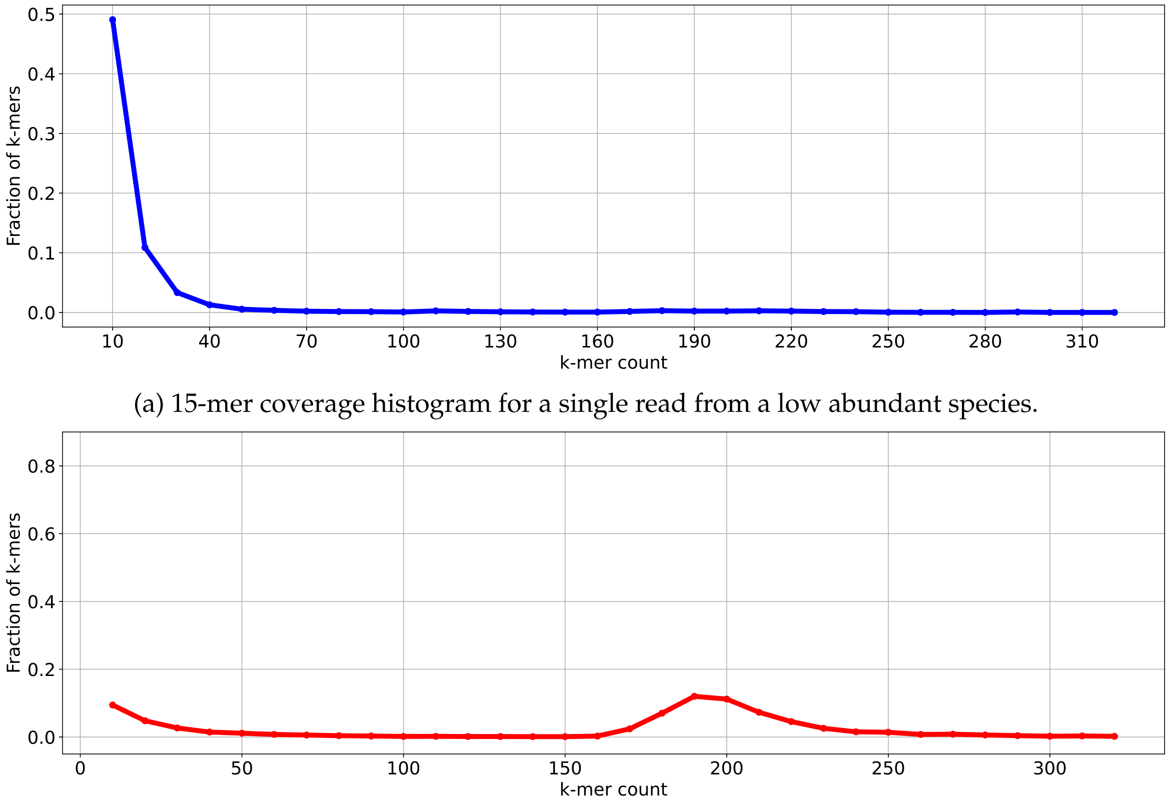
Although the set of genomic 15-mers in a metagenomic dataset is unknown, the coverage of a genomic 15-mers correlates to the coverage of the unknown genome which it belongs to. It has been shown that a high-coverage peak in the 15-mers coverage histogram mainly consists of genomic 15-mers (for sufficiently long *k*, *e.g.*, *k*=13, 15, 17, 19 refer to Appendix A) while a low-coverage peak in the same histogram typically corresponds to non-genomic *k*-mers due to sequencing errors (Kolmogorov, Yuan et al., 2019; Lin, Yuan et al., 2016).

This can be understood using the Figures 3.1 and 3.2 that illustrates various *k*-mer coverage histograms for simulated long reads samples (*C. neoformans* (10×), *S. cerevisiae* (15×), *P. aeruginosa* (550×), *E. faecalis* (450×) and *S. aureus* (600×)). Please note that this dataset is referred to as **2Y3B**. For example, the *k*-mer coverage histograms of long reads in Figure 3.2 (a) consists of long reads sampled from genomes of species with low-coverage (15×) whereas Figure 3.2 (b) contains high-coverage (450× – 600×). In high abundant species, the corresponding genomic 15-mers are in the second peak. However, genomic 15-mers from low-coverage genomes get mixed with non-genomic 15-mers due to their low-coverages.

Conclusively, we can say that 15-mer coverage histograms captures discriminatory coverage information that is effectively used in clustering in our next step.

3.2.2 Perform Dimension Reduction and Clustering based on *k*-mer Coverage Histograms

Before clustering of histogram features, dimension reduction is performed to aid visualization. Such visualization has been previously used to aid manual resolution of metagenomic data in order to determine the number of bins present (Laczny, Pinel et al., 2014). We use the popular Barnes-Hut *t*-distributed Stochastic Neighbor Embedding (BH-tSNE) (Van Der Maaten, 2014). BH-tSNE is an efficient dimension reduction technique and has been used in previous studies to bin contigs (Kouchaki et al., 2019). Each histogram vector is reduced to two dimensions whose original neighborhood



(a) 15-mer coverage histogram for a single read from a low abundant species.

(b) 15-mer coverage histogram for a single read from a high abundant species.

Figure 3.1: Visualization of 15-mer coverage histogram of an individual (a) from a low coverage species and a (b) high coverage species.

information in the low dimensions are preserved enabling effective clustering of data (Kouchaki et al., 2019).

MetaBCC-LR employs the popular density-based clustering algorithm, DBSCAN (Ester et al., 1996), to cluster the two-dimensional data points. We utilize the variation of nearest neighbor distance to estimate the ϵ parameter in DBSCAN (Rahmah and Sitanggang, 2016). The advantage of using DBSCAN is that it enables clustering based on local density and thereby discards outlier points that arise due to sequencing errors. Moreover, it automatically detects the number of clusters.

Figure 3.3(b) demonstrates the plot of 15-mer coverage histogram and its variances within a given species. As indicated in colors, we can clearly see the segregation of band peaks to represent varying levels of coverage of the underlying species. Note that they correspond well to genomes with low, medium and high coverages. The two-dimensional projections of this sampled set of reads (10,000 reads) datasets is demonstrated in Figure 3.4 (a).

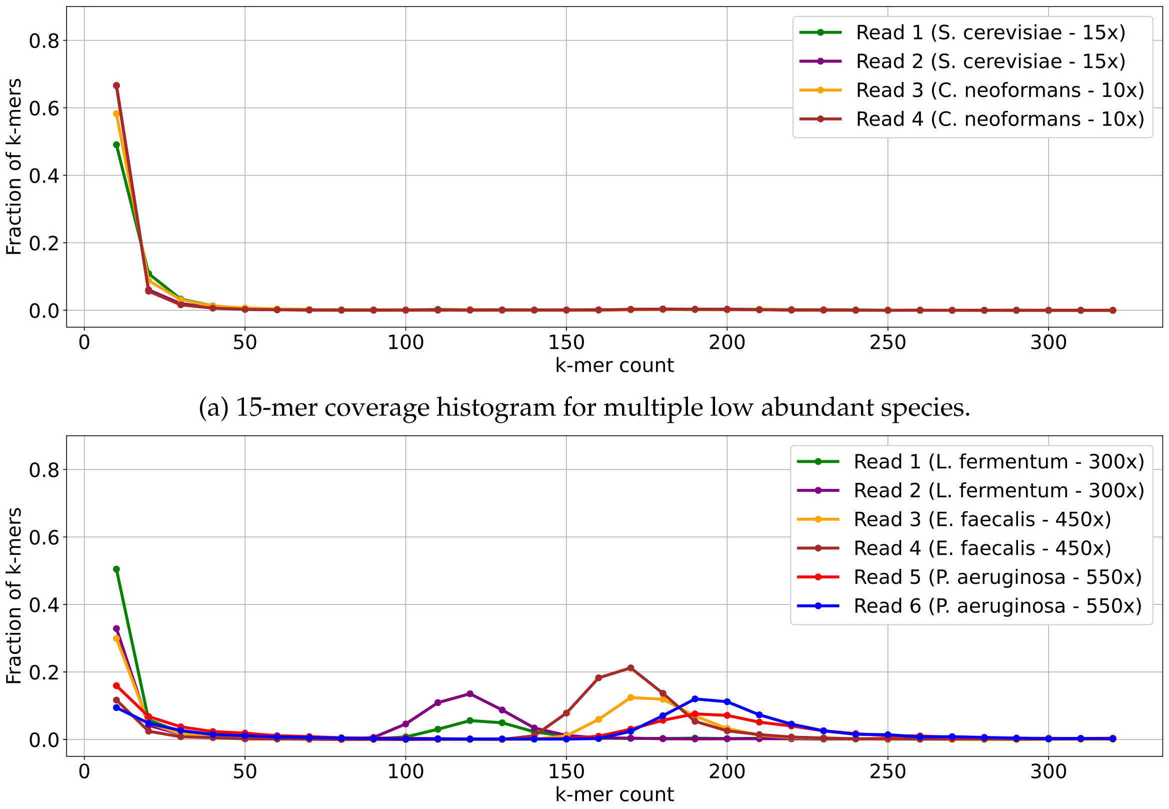
Once the clustering is performed using DBSCAN we observe three major clusters as demonstrated in Figure 3.4 (b). Note that, all the clusters observed in the two-dimensional plots are not preserved in the clustering result due to the noise. In such scenarios, we rely on the separation by coverage to cluster the data points even further.

3.2.3 Obtain Trinucleotide Composition Profiles for Reads

In this section, we investigate the impact of noise in long reads in composition computations and design an appropriate composition feature to support binning of long reads.

Composition for accurate short reads

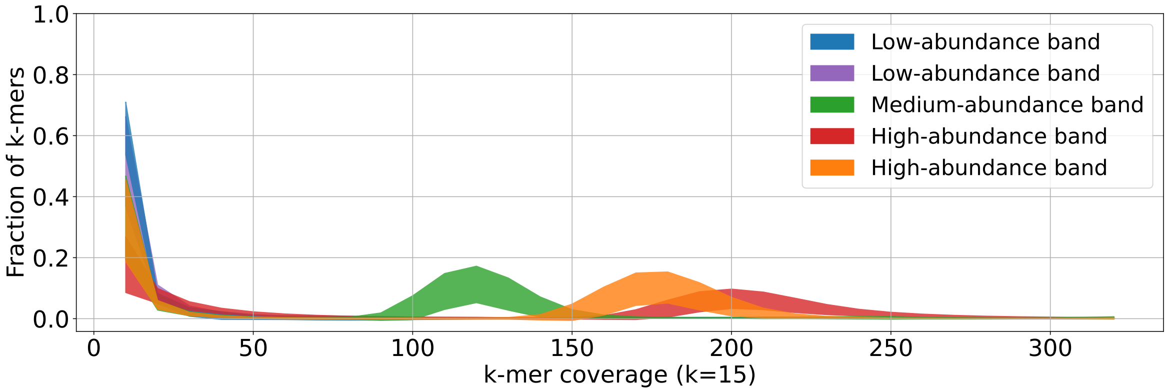
Genomic signatures of microbes display patterns which are unique to each species (Abe et al., 2003; Deschavanne et al., 1999). Hence, these genomic signatures have been used in metagenomics binning.



(a) 15-mer coverage histogram for multiple low abundant species.

(b) 15-mer coverage histogram for multiple high abundant species.

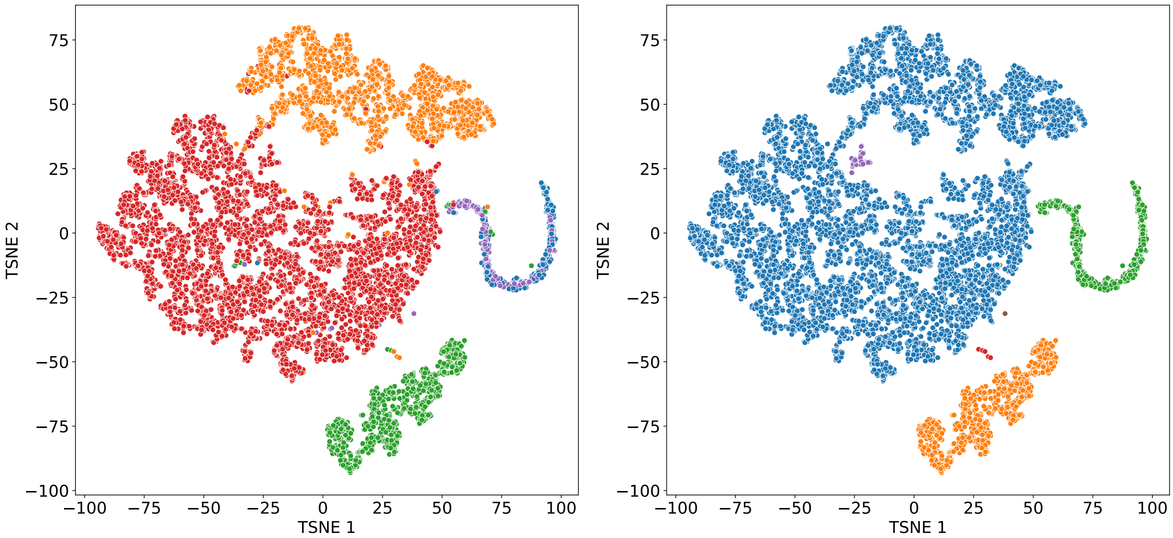
Figure 3.2: Visualization of 15-mer coverage histograms under varying coverages of underlying species. (a) Reads from multiple low coverage species. (a) Reads from multiple high coverage species.



15-mer coverage histograms of reads from species of different abundances

Figure 3.3: Bands of 15-mer coverage histograms for the simulated dataset **2Y3B**.

One of the widely used genomic signatures is the oligonucleotide (*k*-mer for relatively small *k* *e.g.*. *k*=3,4,5) composition where studies have shown that oligonucleotide frequencies are conserved within a species and vary between species (Wu, Simmons et al., 2015). Oligonucleotide frequencies are defined as the normalized frequency of *k*-mers for a given size. These frequencies are treated as a feature vector in metagenomics binning. Similar to the contigs (assembled from short reads), long reads have sufficient length to inherit oligonucleotide composition information of the underlying genomes despite their high error rates. Note that this is not true for short but accurate NGS reads. Please refer to Figure 3.5 for the visualization of oligonucleotide composition for short reads. It clearly evident that the deviation from the reference genome is significant. Hence, the confidence of such vectors from short reads cannot be used for useful application in metagenomics binning. This is



(a) Two dimensional projection of *k*-mer coverage histograms using BH-tSNE.

(b) Clustering of the two-dimensional projection of *k*-mer coverage histograms.

Figure 3.4: (a) Dimension reduction of *k*-mer coverage histograms. The different colors correspond to the underlying species. (b) Clustering using DBSCAN. Illustrated colors corresponds to the different clusters obtained from DBSCAN.

evidently the reason for most traditional metagenomics binners to rely on assembled contigs whilst discarding shorted contigs (Wu, Simmons et al., 2015). In conclusion, despite the favorable error rates in short reads, they cannot be directly used in metagenomics binning.

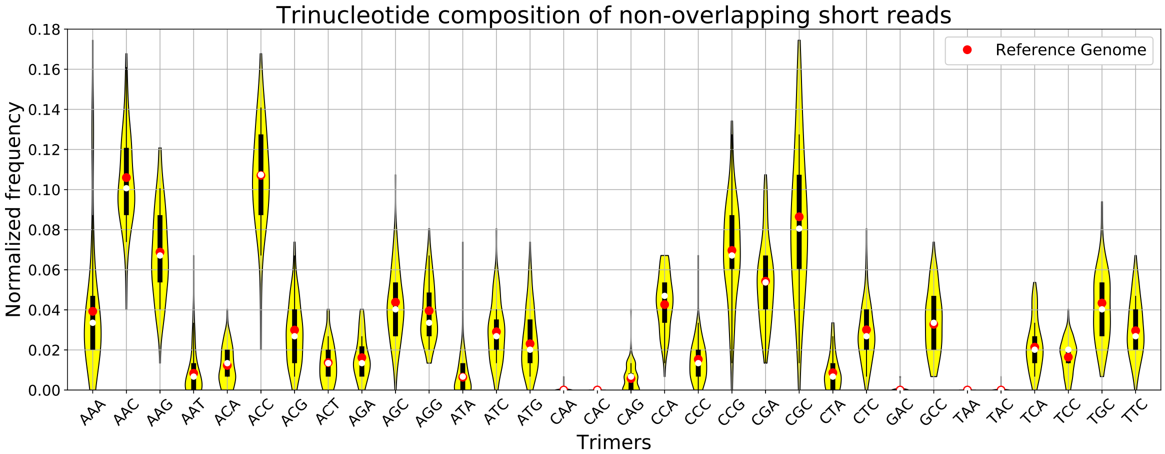


Figure 3.5: Trinucleotide composition of 100 non-overlapping short reads (150bp) for species, *P. aeruginosa*.

Composition for noisy long reads

Figure 3.6 shows the violin plots of trinucleotide composition features for 100 non-overlapping long reads (10kbp-15kbp) simulated from the reference genome of *P. aeruginosa*. From Figure 3.6 we can see that the trinucleotide frequencies of short reads show significant deviations from that of their reference genomes due to their short lengths. On the other hand, Figure 3.6 shows that the trinucleotide frequencies of long reads follow a close pattern to that of the reference genome despite their high error rates. Long reads follow similar trinucleotide composition patterns as that of their reference genomes and show less deviation than short reads.

For each long read, MetaBCC-LR builds a profile vector $V_{composition}$ consisting of *trinucleotide* (*3-mer* or *trimer*) occurrences. Since there are 32 unique trinucleotides (combining reverse complements as well), the resulting vectors will have 32 dimensions. The vector coordinates are then normalized by the total

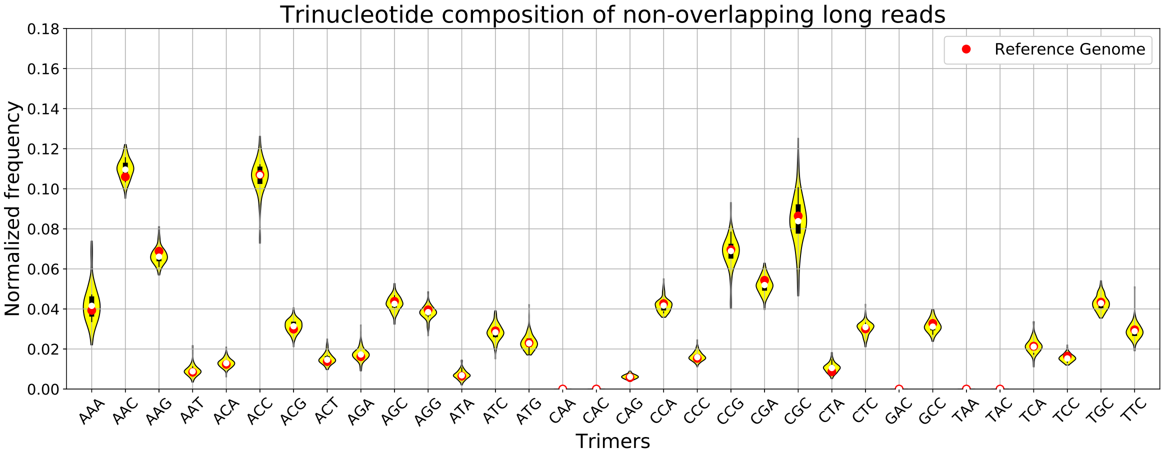
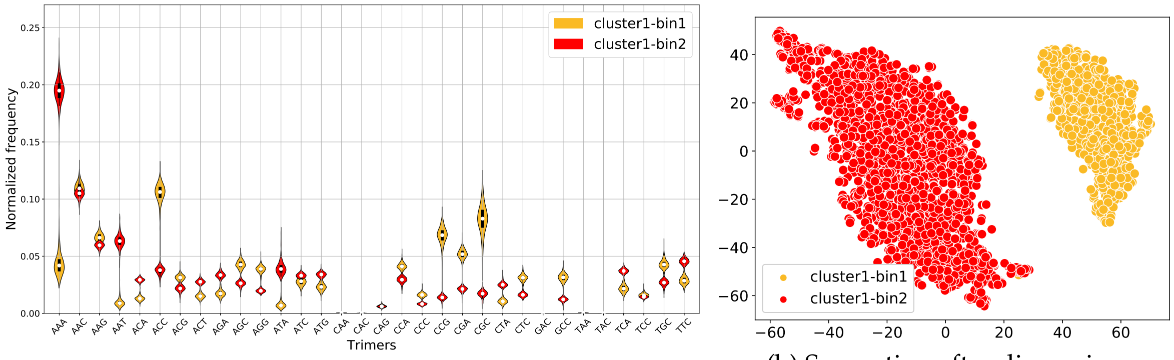


Figure 3.6: Trinucleotide composition 100 non-overlapping long reads (10kbp-15kbp) simulated from the reference genome of *P. aeruginosa*.

number of trinucleotides in the long read to avoid any read-length bias. Such discriminatory patterns are utilized to further subdivide the clusters obtained from Step 2. Hence, despite the higher error rate in long reads, such reads can still produce useful composition profiles for binning purposes.



(a) Trinucleotide composition of reads in high abundance cluster

(b) Separation after dimension reduction and clustering

Figure 3.7: (a) The trinucleotide composition of reads in the high-abundance cluster of Zymo-1Y3B dataset and (b) the binning result after performing dimension reduction. Please note that colors in (a) are for illustration purposes which MetaBCC-LR is not aware of. Colors (clustering result) are inferred in (b) from the mixed signals in (a).

The toy example demonstrated in Figure 3.7 (a) illustrates the mean and standard deviation of composition vectors of two species from the **2Y3B** dataset for simplicity. As illustrated in the following figure, Figure 3.7 (b), two clear clusters can thus be inferred via dimension reduction and clustering using DBSCAN.

3.2.4 Perform Dimension Reduction and Binning based on Trinucleotide Composition

Similar to Step 2, MetaBCC-LR first uses BH-tSNE (Van Der Maaten, 2014) to map the trinucleotide composition profiles to two-dimensional vectors. Clusters of long reads derived from Step 2 are further divided into bins according to their trinucleotide composition profiles using DBSCAN. For example, Figure 3.7 (b) shows the two-dimensional plot of reads in the high-abundance cluster of Zymo-1Y3B dataset after dimension reduction using BH-tSNE and the two clusters inferred by DBSCAN that correspond to two genomes with distinct trinucleotide composition in Figure 3.7 (b) (a). At this stage, MetaBCC-LR has developed features to represent coverage and composition.

3.2.5 Binning large metagenomics long reads datasets using data sampling and statistical models

Recall that our third challenge in metagenomics binning is the existence of numerous data points for clustering. In MetaBCC-LR, we employ a sampling strategy where we sample 10% of reads to identify the bins (*i.e.*, cluster detection). This is mainly adopted since, non-linear dimension reduction techniques such as BH-tSNE is computationally intensive at large scales (time complexity $O(N^2)$ for exact algorithm (Van Der Maaten, 2014; Van der Maaten and Hinton, 2008)). MetaBCC-LR first computes 15-mer coverage histograms and trinucleotide composition profiles for the entire dataset. Next a subset of reads is sampled and clustered (1) first using coverage histograms and then (2) using composition profiles. Now we have discovered B bins for the sampled reads.

For the i^{th} bin B_i, MetaBCC-LR builds a statistical model by calculating the mean $\mu_{coverage(i)}$ and standard deviation $\sigma_{coverage(i)}$ for the vectors $V_{coverage}$ (*i.e.*, k-mer coverage histograms) and the mean $\mu_{composition(i)}$ and standard deviation $\sigma_{composition(i)}$ for the vector $V_{composition}$ (*i.e.*, trinucleotide composition profile). For each read vector $\bar{v}$, MetaBCC-LR computes the multivariate probability that it belongs to a bin with mean vector $\bar{\mu}$ and standard deviation vector $\bar{\sigma}$ using the following Gaussian distribution:

$$PDF(\bar{v}, \bar{\mu}, \bar{\sigma}) = \prod_{j}^{|\bar{v}|} \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}} \tag{3.1}$$

Here, $|\bar{v}|$ stands for the size of each vector that we compare (*i.e.*, 32). More specifically, a long read r, with a k-mer coverage histogram $V_{coverage(r)}$ and trinucleotide frequency vector $V_{composition(r)}$, is assigned into a bin in a maximum likelihood framework demonstrated in equation 3.2. Finally, after assigning all the reads to the identified bins, MetaBCC-LR will output a bin identifier for each read.

$$B_i = \text{argmax}_i \left\{ PDF(V_{coverage(r)}, \mu_{coverage(i)}, \sigma_{coverage(i)}) \times PDF(V_{composition(r)}, \mu_{composition(i)}, \sigma_{composition(i)}) \right\} \tag{3.2}$$

The complete MetaBCC-LR pipeline is shown in Figure 3.8. Firstly, MetaBCC-LR samples a subset of reads and computes coverage histograms for long reads followed by clustering using these features (steps 1 and 2). Secondly, composition profiles are computed, and the already clustered reads are re-clustered (steps 3 and 4). Finally, the unsampled reads are assigned to the identified clusters; known as bins (step 5).

3.3 Experimental Setup

We evaluated our approach using several simulated and publicly available real datasets. In order to compare our approach we also used several baseline tools that support long reads data as direct inputs. It must be noted that, *de-novo* long read binning is a novel concept, although there are several reference based attempts have been made. In our evaluation, we narrow down our comparison to reference free methods to show a fair comparison.

3.3.1 Datasets

We used the following long-read datasets for our evaluation.

1. Oxford Nanopore GridION mock community dataset from the ZymoBIOMICS EVEN Microbial Community Standards (Nicholls et al., 2019) (denoted by **ZymoEVEN**). The original **ZymoEVEN** dataset containing ONT reads and five additional PacBio datasets (denoted by **Zymo-1Y2B**, **Zymo-1Y3B**, **Zymo-2Y2B**, **Zymo-2Y3B** and **Zymo-2Y4B**) are simulated using the species found in the **ZymoEVEN** dataset.

   Similar to experiments conducted by researchers at Loman Lab (LomanLab, 2019), we only considered the reads that matched to a specific reference genome in the mock community

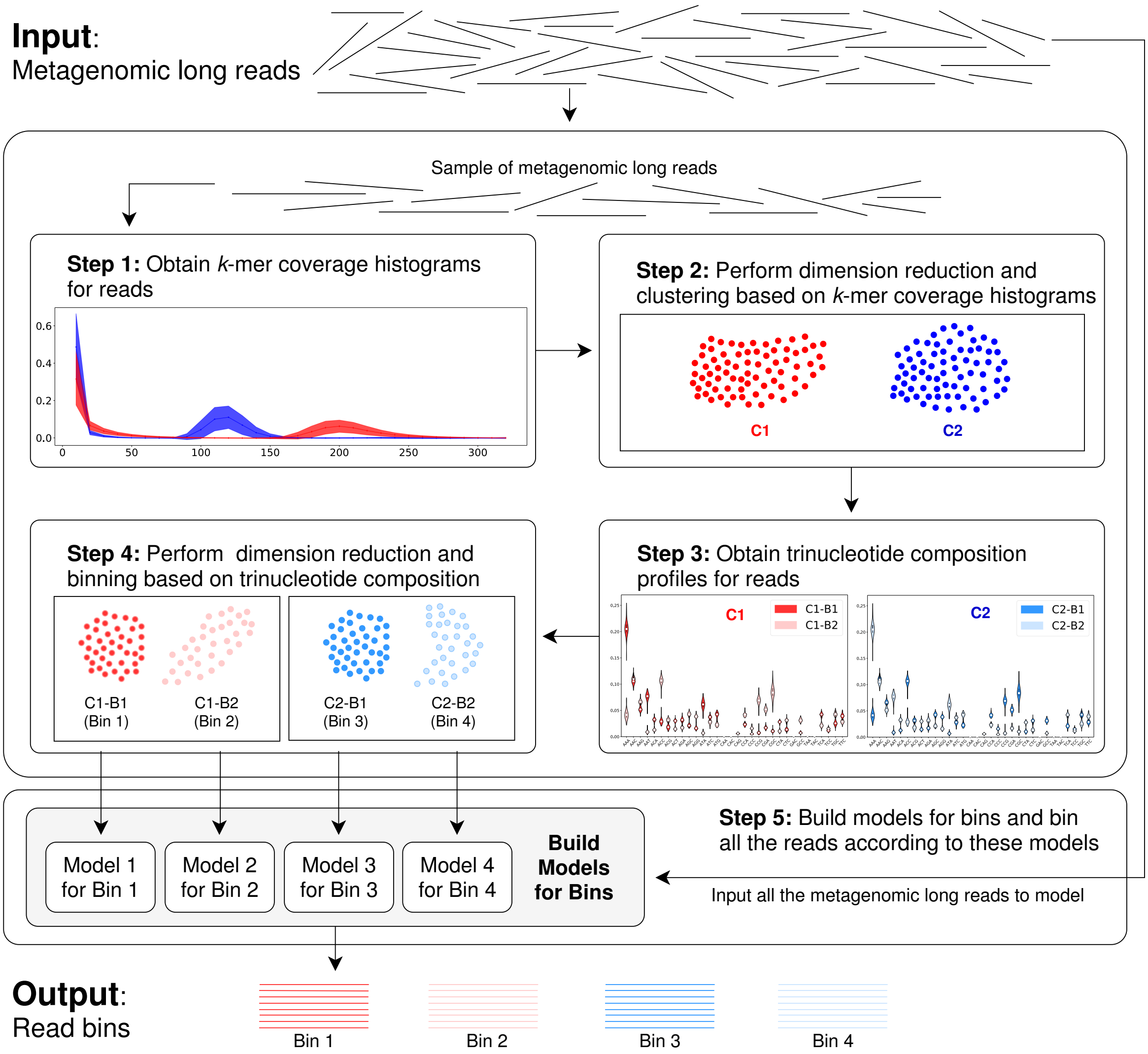

Input:
Metagenomic long reads
Sample of metagenomic long reads
Step 1: Obtain k-mer coverage histograms for reads
0.6
0.4
0.2
0.0
0
50
100
150
200
250
300
Step 2: Perform dimension reduction and clustering based on k-mer coverage histograms
C1
C2
Step 3: Obtain trinucleotide composition profiles for reads
C1
C1-B1
C1-B2
C2
C2-B1
C2-B2
Step 4: Perform dimension reduction and binning based on trinucleotide composition
C1-B1 (Bin 1)
C1-B2 (Bin 2)
C2-B1 (Bin 3)
C2-B2 (Bin 4)
Model 1 for Bin 1
Model 2 for Bin 2
Model 3 for Bin 3
Model 4 for Bin 4
Build Models for Bins
Step 5: Build models for bins and bin all the reads according to these models
Input all the metagenomic long reads to model
Output:
Read bins
Bin 1
Bin 2
Bin 3
Bin 4

Figure 3.8: The Workflow of MetaBCC-LR

standard. Note that, Loman Lab (LomanLab, 2019) performed taxonomic binning of reads and performed assembly possibly filtering away the noisy ONT reads and/or eukaryotic/host sequences.

2. Artificial Skin Microbiome datasets (NCBI BioProject number PRJNA533970) consisting of four different mixes of five common skin bacteria with various noise levels (denoted by **ASM-0**, **ASM-5**, **ASM-10** and **ASM-15**). This dataset was released by Leiden University Medical Center on 21-Apr-2019.

3. The Preborn infant gut metagenome (NCBI Accession No. SRA052203) (Sharon et al., 2013). A PacBio dataset is simulated using the five most abundant species with their corresponding coverages (denoted by **Sharon**).

4. Simulated metagenome with 100 species (Wu, Tang et al., 2014). A PacBio dataset is simulated using the 100 species found (denoted as **100-genomes**).

Note that simulated datasets were generated by the long-read simulator SimLoRD using default parameters for PacBio reads (Stöcker et al., 2016). Further details about each simulated dataset can be found in Appendix Table B.1 and B.2. The composition of the real datasets are available in Appendix Table B.6.

3.3.2 Tools Compared

We compared MetaBCC-LR with two recent reference-free binning tools which support long reads, MetaProb (Girotto et al., 2016) and BusyBee Web (Laczny, Kiefer et al., 2017). BusyBee Web is a web based tool where users have restricted access to how much data can be uploaded. The website has a limit of 200MB for the upload. Hence, we input 10,000 sampled reads from each of our datasets in the comparison. MetaProb is limited by the input size due to its memory limitations in the implementation. Therefore, we provide the same sampled 10,000 reads to MetaProb as well.

3.3.3 Evaluation Criteria

Since the reference genomes for ZymoEVEN and ASM datasets are available, we mapped the reads to these reference genomes using Minimap2.1 (Li, 2018a). The reads which had over 90% of the bases mapped to a reference genome were considered for evaluation. For the simulated datasets from known reference genomes, we used their ground-truth label to evaluate the binning result. We determined the precision, recall, F1 score and Adjusted Rand Index (ARI) for the binning results of each dataset.

The binning result is represented as an $M \times N$ matrix where M refers to the number of bins and N refers to the number of species. In this matrix, the element R_{ij} denotes the number of reads in bin i belonging to species j. Let T be the total number of reads binned. The precision, recall, F1 score and Adjusted Rand Index (ARI) are calculated as follows (Girotto et al., 2016; Wang, Leung et al., 2012; Wang, Wang, Lu and Sun, 2017).

$$Precision(\%) = \frac{\sum_{i=1}^{M} max_j\{R_{ij}\}}{\sum_{i=1}^{M}\sum_{j=1}^{N}\{R_{ij}\}} \times 100 \tag{3.3}$$

$$Recall(\%) = \frac{\sum_{j=1}^{N} max_i\{R_{ij}\}}{\sum_{i=1}^{M}\sum_{j=1}^{N}\{R_{ij}\} + Number\ of\ unclassified\ reads} \times 100 \tag{3.4}$$

$$F1\ score(\%) = 2 \times \frac{Precision \times Recall}{Precision + Recall} \times 100 \tag{3.5}$$

$$ARI(\%) = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N}\binom{R_{ij}}{2} - t_3}{\frac{1}{2}(t_1 + t_2) - t_3} \times 100 \tag{3.6}$$

$$where\ t_1 = \sum_{i=1}^{M}\binom{\sum_{j=1}^{N} R_{ij}}{2},\ t_2 = \sum_{j=1}^{N}\binom{\sum_{i=1}^{M} R_{ij}}{2},\ and\ t_3 = \frac{t_1 t_2}{\binom{T}{2}}$$

3.4 Results and discussion

3.4.1 Binning Results

We recorded the number of bins identified by each tool for all the datasets and the values can be found in Table 3.2.

Number of bins estimated

It was observed that MetaProb and BusyBee Web tend to produce more bins than the actual number of species present. In comparison, MetaBCC-LR was able to identify a number closer to the actual number of species present in all the datasets.

Accuracy of binning

Figure 3.9 compares the precision of the binning results of all the datasets obtained from MetaBCC-LR with MetaProb and BusyBee Web. It is clearly evident that MetaBCC-LR has resolved bins with a

Table 3.2: Comparison of the actual number of species present and the number of bins identified by the tools for different datasets.

Dataset	Actual no. of species present	No. of bins identified by **MetaProb**	No. of bins identified by **BusyBee Web**	No. of bins identified by **MetaBCC-LR**
Zymo-1Y2B	3	6	23	**3**
Zymo-1Y3B	4	7	16	**4**
Zymo-2Y2B	4	8	21	**4**
Zymo-2Y3B	5	8	18	**5**
Zymo-2Y4B	6	9	18	**6**
ZymoEVEN	10	13	26	**8**
Sharon	5	8	123	**4**
ASM-0	5	11	10	**5**
ASM-5	5	13	8	**4**
ASM-10	5	14	14	**4**
ASM-15	5	22	10	**5**
100-genomes	100	22	**74**	70

greater precision compared to the baselines. BusyBee Web maintains reasonable precision values for **Zymo** datasets except for the real dataset. This is mainly because BusyBee Web relies on 5-mers, which are too long to handle the noise in ONT long reads. For **Zymo** simulated datasets MetaProb has a competitive precision while for **ZymoEVEN** dataset the tool performs rather poorly. This is mainly because, MetaProb is a short reads specialized binner where read overlaps are considered to identify bins. However, due to the higher error rate in the real dataset (**ZymoEVEN**), MetaProb has failed to properly capture such overlaps to perform accurate binning. Note that MetaProb and BusyBee Web perform poorly on the **ASM** datasets. This is because BusyBee Web utilizes a pre-trained model based on a selected set of reference genomes, whereas MetaProb requires overlapping k-mers to form read groups which are used for clustering reads into bins. Moreover, **ASM** datasets are much smaller with possibly very few overlaps between reads to perform clustering.

Although BusyBee Web has comparable recalls as presented in Figure 3.10, this is computed using the set of sampled reads provided for the tool. Hence, the recall values do not indicate a straight comparison with MetaBCC-LR. Due to lower coverage in **ASM** datasets, MetaProb has poor recall producing fragmented bins with poor recall.

Figure 3.11 illustrates the F1-scores of the binning results. It is clearly evident that performance of MetaBCC-LR is superior in handling the novel problem of binning metagenomics long reads.

Table 3.3 compares the mean and standard deviation of each evaluation metric averaged over all the datasets for each tool. MetaBCC-LR outperforms the other tools in all the metrics. The increase in F1 score and ARI is ∼13% and ∼30% respectively over the best baseline method, MetaProb. MetaBCC-LR also has the most consistent performance with the lowest standard deviation values (less than 10%) in all metrics.

3.4.2 Metagenome Assembly Results

One major advantage in metagenomics binning of reads is the ability to influence the downstream assembly process. Typically, assembly is conducted using a genome assembly software where the

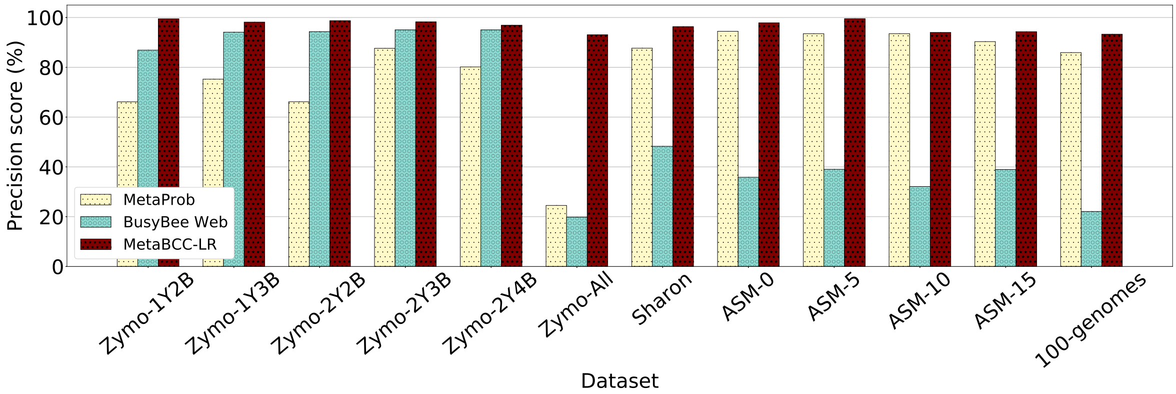


Figure 3.9: Binning precision of datasets.

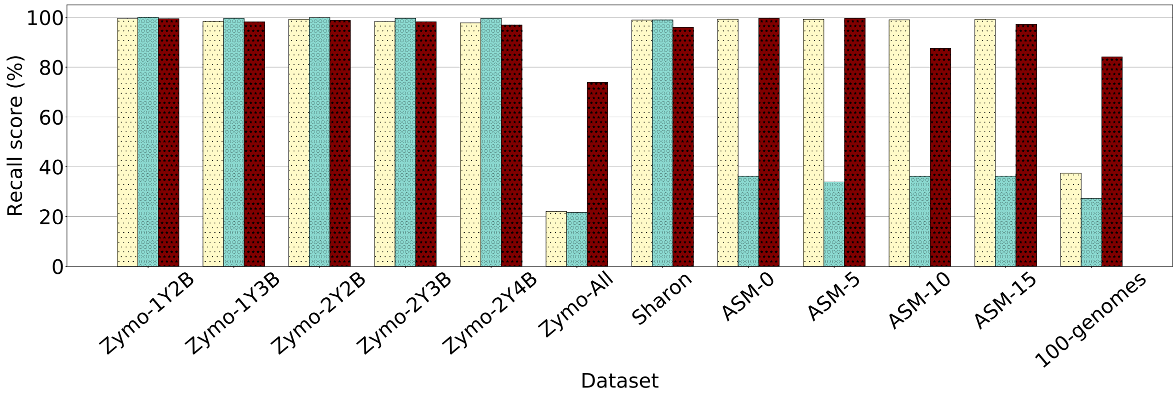


Figure 3.10: Binning recall of datasets.

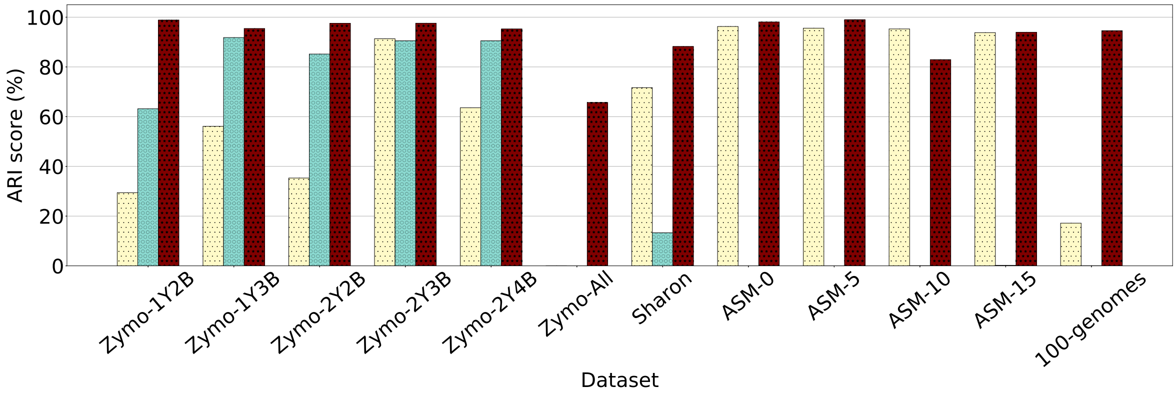


Figure 3.11: Binning F1-score of datasets.

contigs are produced based on overlapping reads. Unless the assembler is metagenomic specialized the process of assembly can be very challenging. This is because, typical assemblers are based on uniform coverage and has tendency to discard low coverage regions. This results in missing assemblies or highly fragmented assemblies of low abundant species in a sample. However, the scientific importance of such low abundant species is not negligible, hence, complete recovery of all the available genomes is the utmost priority.

Assemblers used

To demonstrate the effect of MetaBCC-LR on metagenomics assembly, we assembled all the Zymo datasets (simulated and real) individually using two popular long-read assemblers metaFlye (Kolmogorov, Rayko et al., 2019) (available in Flye v2.4.2) and Canu v1.8 (Koren et al., 2017) (denoted as **complete**

Table 3.3: Comparison of mean and standard deviation (STD) of each evaluation metric averaged over all the datasets for each tool.

Tool	Precision (%)		Recall (%)		F1 score (%)		ARI (%)	
	Mean	STD	Mean	STD	Mean	STD	Mean	STD
MetaProb	78.77	19.77	87.35	27.11	81.42	22.19	62.13	34.42
BusyBee Web	58.46	31.52	65.75	35.58	61.27	32.64	36.24	43.15
MetaBCC-LR	**96.66**	**2.40**	**94.12**	**8.08**	**95.27**	**5.35**	**92.28**	**9.61**

assembly) and also assembled the partitioned reads of the individual bins from MetaBCC-LR using metaFlye and Canu (denoted as **partitioned assembly**). We selected the Zymo dataset (Nicholls et al., 2019) because it has been used to evaluate the metagenomic assembler metaFlye (Kolmogorov, Rayko et al., 2019). We evaluated all the assemblies using MetaQUAST (Mikheenko et al., 2015a). Please note that binning results from other tools were not assembled as MetaProb and BusyBee Web were not able to bin the entire dataset at once.

Table 3.4: Comparison of the assembled genome fraction of the different assemblies for different datasets.

Dataset	metaFlye assembly		Canu assembly	
	Complete	Partitioned	Complete	Partitioned
Zymo-1Y2B	93.12%	**99.60%**	78.74%	**98.69%**
Zymo-1Y3B	93.97%	**99.65%**	78.20%	**98.79%**
Zymo-2Y2B	93.90%	**97.78%**	57.28%	**97.18%**
Zymo-2Y3B	**97.35%**	93.44%	63.53%	**95.66%**
Zymo-2Y4B	94.55%	**97.59%**	71.35%	**86.69%**
ZymoEVEN	86.47%	**88.68%**	68.79%	**85.42%**

Quality of assemblies

Genome fraction is a key indicator of assembly quality. It is defined as the ratio of total number of aligned bases in the reference to the genome size in base pairs. In other words, genome fraction speaks out for the content of the genome recovered through the process of assembly.

Table 3.4 shows the comparison between genome fraction of complete and partitioned assemblies. Applying MetaBCC-LR to bin long reads prior to assembly (*i.e.*, partitioned assemblies) improves the genome fraction over the complete assemblies. One possible reason for such improvement is that partitioned assemblies allow assemblers to estimate more appropriate parameters for reads in each bin rather than applying the same parameters to the entire dataset. This may help to recover more genomic sequences, especially for low-abundant species.

Resource usage for assemblies

Tables 3.5 and 3.6 shows the comparison of resource usage (assembly time and peak memory) for complete assembly and partitioned assembly. Assembly time of partitioned assemblies includes the CPU time elapsed for binning using MetaBCC-LR (refer to Table 3.8) and metagenome assembly. As expected, partitioned assemblies have consumed lesser time and memory than the complete

Table 3.5: Comparison of resource usage between complete assembly and partitioned assembly for different datasets using metaFlye assembler. Assembly time for partitioned assemblies includes the CPU time elapsed for binning using MetaBCC-LR and metagenome assembly.

Dataset	Performance metric	metaFlye assembly	
		Complete assembly	Partitioned assembly
Zymo-1Y2B	Assembly Time (h)	12.15	**9.25**
	Memory usage (GB)	35.34	**24.21**
Zymo-1Y3B	Assembly Time (h)	15.40	**11.96**
	Memory usage (GB)	36.53	**22.16**
Zymo-2Y2B	Assembly Time (h)	13.41	**10.43**
	Memory usage (GB)	35.41	**23.42**
Zymo-2Y3B	Assembly Time (h)	16.51	**13.49**
	Memory usage (GB)	35.81	**23.81**
Zymo-2Y4B	Assembly Time (h)	20.74	**15.63**
	Memory usage (GB)	54.84	**21.65**
ZymoEVEN	Assembly Time (h)	45.95	**36.45**
	Memory usage (GB)	129.36	**16.19**

assemblies. The reduction in resource usage is more significant in Canu assemblies than in metaFlye assemblies because Canu is a generic assembler whereas metaFlye is a dedicated metagenomics assembler.

Table 3.7 shows the average improvements of the partitioned assemblies after using MetaBCC-LR, compared to the complete assemblies. We observed improvements in genome fraction with significant reduction in time and memory usage. These results show that MetaBCC-LR can be used to improve metagenomic assembly by binning long reads before assembly.

3.4.3 Running Times of MetaBCC-LR

Table 3.8 shows the times taken by MetaBCC-LR to bin the datasets Zymo-1Y2B, Zymo-1Y3B, Zymo-2Y2B, Zymo-2Y3B, Zymo-2Y4B and ZymoEVEN. Please note that the running times for MetaProb and BusyBee Web were not recorded. They were not able to bin the entire dataset at once and running times for BusyBee Web could not be measured since it is a web application. These factors make it challenging to conduct a fair running time comparison.

Our approach, MetaBCC-LR, is a two-phased binning approach to bin noisy long reads without the use of reference databases. The first phase uses k-mer coverage histograms and the second phase uses trinucleotide composition to separate reads. The two phases are executed sequentially. The two phases of MetaBCC-LR and the final step of building a statistical model for each bin, use a sample of the input dataset. Finally, a bin is assigned to all the reads in the input data.

We conducted experiments to see how alternative approaches and read assignment would affect the final binning results. Moreover, we conducted experiments to see how sampling improves the scalability of MetaBCC-LR without compromising the binning quality. Furthermore, we conducted experiments to show the importance of MetaBCC-LR in solving real-world metagenomics binning problems.

Table 3.6: Comparison of resource usage between complete assembly and partitioned assembly for different datasets using Canu assembler. Assembly time for partitioned assemblies includes the CPU time elapsed for binning using MetaBCC-LR and metagenome assembly.

Dataset	Performance metric	Canu assembly	
		Complete assembly	Partitioned assembly
Zymo-1Y2B	Assembly Time (h)	74.61	**58.01**
	Memory usage (GB)	13.18	**8.01**
Zymo-1Y3B	Assembly Time (h)	86.51	**78.33**
	Memory usage (GB)	18.43	**7.85**
Zymo-2Y2B	Assembly Time (h)	75.20	**61.12**
	Memory usage (GB)	19.45	**11.46**
Zymo-2Y3B	Assembly Time (h)	87.22	**82.93**
	Memory usage (GB)	22.15	**7.94**
Zymo-2Y4B	Assembly Time (h)	102.13	**101.16**
	Memory usage (GB)	25.78	**12.44**
ZymoEVEN	Assembly Time (h)	437.99	**258.00**
	Memory usage (GB)	108.50	**21.76**

Table 3.7: Average improvement in Genome Fraction (GF) due to MetaBCC-LR.

Assembly	Complete	Partitioned: with MetaBCC-LR		
	GF	GF	Memory Saved	Time Saved
metaFlye	93.23%	96.12%	87.48%	24.64%
Canu	69.65%	93.74%	79.94%	41.09%

Table 3.8: Time taken by MetaBCC-LR to bin different datasets.

Dataset	Size (GB)	Running time (CPU hours)
Zymo-1Y2B	4.2	0.91
Zymo-1Y3B	5.45	1.19
Zymo-2Y2B	4.35	0.95
Zymo-2Y3B	5.65	1.24
Zymo-2Y4B	7.15	1.60
ZymoEVEN	14.24	3.12

3.4.4 Alternative Approaches for Binning Long Reads

We conducted experiments on two alternative approaches; (1) separate long reads first by their trinucleotide composition and then by their coverage (Composition first) and (2) combining the *k*-mer coverage histograms and trinucleotide composition profiles and apply dimension reduction to bin reads (*k*-mer Coverage + Composition). Table 3.9 shows the comparison of results obtained for the Zymo-2Y4B dataset using each of these methods with MetaBCC-LR.

Table 3.9: Comparison of alternative approaches with MetaBCC-LR using the Zymo-2Y4B dataset.

Method	No. of bins identified	Precision	Recall	F1 score	ARI
Composition first	5	92.71%	90.14%	91.41%	79.78%
k-mer Coverage + Composition	2	**99.59%**	47.80%	64.60%	0.43%
Coverage first (MetaBCC-LR)	**6**	98.46%	**98.46%**	**98.64%**	**97.21%**

We observed that even though we obtained very high precision values with the combined approach (*k*-mer Coverage + Composition), it resulted in poor recall and ARI values. Overall, MetaBCC-LR has outperformed other alternative methods and produced better scores for the evaluation metrics.

3.4.5 Read Assignment

We conducted experiments to compare the performance of doing dimension reduction and binning on the entire dataset with our method where we assign reads based on the models obtained from doing dimension reduction and binning on the sample of reads from a dataset. Our method took 1 minute and 19 seconds of wall time (on a Linux system with Ubuntu 18.04.1 LTS, 16G memory and Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz with 4 CPU cores and 8 threads) to bin and assign the reads of the Zymo-2Y4B dataset (with a sample of 8,620 reads) whereas when we performed dimension reduction and binning on the entire dataset at once, the process did not finish even after 3 hours of wall time had elapsed.

Table 3.10: Comparison of the precision and recall of sampled reads and after assigning all the reads.

Dataset	Sampled reads	Precision of sampled reads	Recall of sampled reads	Total reads	Final precision after	Final recall
Zymo-1Y2B	5,028	99.42%	99.42%	502,890	99.47%	99.47%
Zymo-1Y3B	6,576	98.97%	98.97%	657,610	99.27%	99.27%
Zymo-2Y2B	5,259	99.13%	99.13%	525,922	99.51%	99.51%
Zymo-2Y3B	6,806	98.81%	98.81%	680,642	99.24%	99.24%
Zymo-2Y4B	8,620	98.46%	98.46%	862,021	98.46%	98.46%
ZymoEVEN	34,910	97.64%	71.92%	3,491,078	93.09%	73.84%

We observed that the precision values of clustering of sampled reads and read assignment afterwards remain similar. Initial uniform sampling was done in order to obtain a smaller representation of the dataset considering the computational complexity of BH-tSNE. Read assignment is labelling the unsampled reads to the bins identified from the sampled set of reads. Table 3.10 shows the precision and recall of the initial sampled binning and final binning after read assignment for the Zymo-1Y2B, Zymo-1Y3B, Zymo-2Y2B, Zymo-2Y3B, Zymo-2Y4B and ZymoEVEN datasets. Even though the precision has reduced slightly after read assignment, we obtained a significant performance gain by using this method over doing dimension reduction for the entire dataset.

3.4.6 Effect of Initial Sample Size

We conducted experiments to check the effect of varying initial sample sizes on the final binning results. We selected sample sizes 0.5%, 1% and 1.5% of reads from each of the complete simulated Zymo datasets to determine the number of bins and build their corresponding statistical profiles. The average results obtained for the simulated Zymo datasets for 0.5% and 1% are shown in Table 3.11 as we observed the most significant gain in performance when the sampling size was increased from 0.5% to 1%.

Table 3.11: Comparison of average evaluation metrics for varying sample sizes of the simulated Zymo datasets.

Sample size	Precision	Recall	F1 score	ARI
0.5%	98.49%	98.49%	98.49%	97.03%
1%	**99.19%**	**99.19%**	**99.19%**	**98.14%**

We observed that the final binning precision and recall after performing read assignment remains very similar for each sample size. Finally, 1% was chosen in order to retain reads of very low abundant species during the clustering process.

3.4.7 MetaBCC-LR to Separate Coral Genome from Its Symbiont Genomes

Many adult reef-building corals are known to be supplied through a mutually-beneficial relationship with its microbial symbionts that live inside the coral cells. However, this symbiotic relationship makes it very challenging to separate adult coral from its microbial symbionts as they live inside the coral cells ((Ying et al., 2018) and personal communications). In order to evaluate the utility of MetaBCC-LR in coral studies, we simulated a PacBio dataset using the Coral *P. lutea* and its microbial symbiont *Cladocopium C15* found from the Coral and its microbial symbiont communities from the Orpheus Island, Australia (Robbins et al., 2019) (denoted by **Coral+Symbio**). Details about the dataset can be found in Appendix Table B.6. We used MetaBCC-LR to separate the reads of coral from its microbial symbiont, to show the importance of our tool in solving real-world metagenomics binning problems.

Figure 3.12 denotes the two-dimensional plot of the read clusters obtained from the sample of the Coral+Symbio dataset, (a) after separating by coverage and (b) after separating by composition. According to Figure 3.12(a) it can be seen that separating using coverage (*k*-mer coverage histograms) has resulted in a single cluster since *P. lutea* and *Cladocopium C15* are equally abundant in the Coral+Symbio dataset. However, after separating using composition (trinucleotide composition), we can see that there are two clearly separated clusters corresponding to *P. lutea* and *Cladocopium C15* (Figure 3.12(b)). Moreover, MetaBCC-LR resulted in precision, recall, F1 score and ARI of 98.21%, 98.21%, 98.21% and 92.97% respectively for the final binning of the entire Coral+Symbio dataset. Hence, it can be seen that MetaBCC-LR can be used to solve real-world metagenomic binning problems.

3.5 Discussion

3.5.1 Summary

With the increased popularity of long-read sequencing technologies, the field of metagenomics has shown a rising interest in using long reads for analyses. In metagenomics binning studies, most of the available reference-free binning tools are designed for and tested on short reads and cannot handle large datasets of long reads due to the higher error rate and shear volume compared to assembled contigs. In this chapter, we designed and evaluated MetaBCC-LR, a scalable reference-free binning tool to cluster large long-read datasets. MetaBCC-LR uses *k*-mer coverage histograms and oligonucleotide composition of the reads in a carefully designed pipeline of steps to estimate the number of bins and classify the input reads to the bins. Our extensive experimental results, on several datasets with

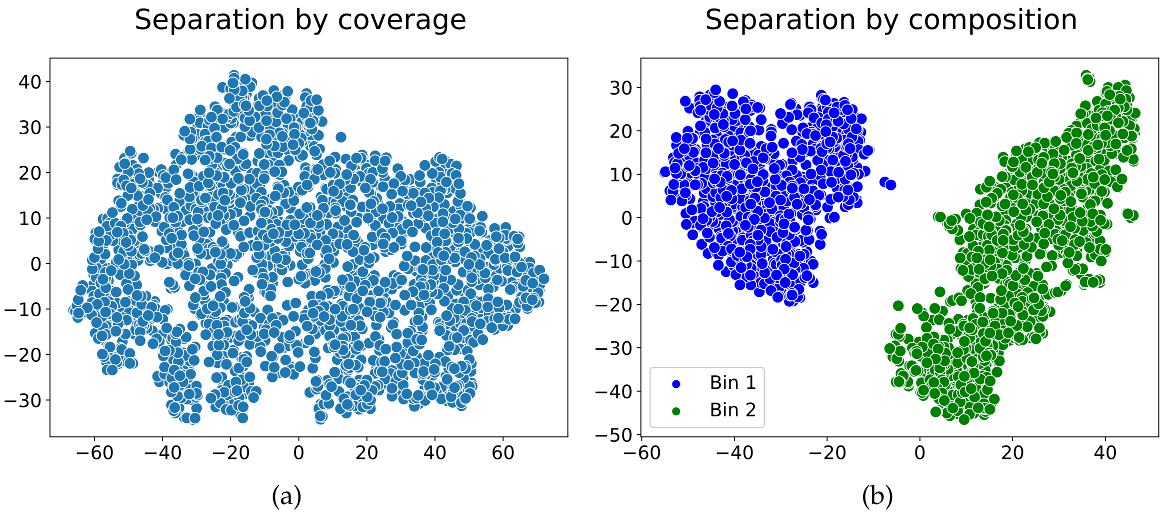


Figure 3.12: Read clusters obtained from simulated PacBio reads from the Coral *P. lutea* and its microbial symbiont *Cladocopium C15* found from the Coral and its microbial symbiont communities from the Orpheus Island, Australia (Robbins et al., 2019) (denoted by Coral+Symbio). MetaBCC-LR achieved the final binning precision, recall, F1 score and ARI as 98.21%, 98.21%, 98.21% and 92.97%, respectively.

varying coverage and error rates, show that MetaBCC-LR outperforms state-of-the-art reference-free binning tools by a substantial margin.

Typically, assemblers erroneously assume uniform coverages and low abundant species tend to be ignored – that can be ameliorated by binning reads before assembly. We indeed observe that binning long reads using MetaBCC-LR prior to assembly improves assembled genome fractions. Further, this is achieved along with considerable reduction in time and memory usage. Binning is a crucial step in many metagenomics studies and the efficiency and accuracy of MetaBCC-LR can potentially lead to improved characterization of microbial communities to provide valuable biological insights.

3.5.2 Room for improvements

Step wise clustering approach

So far, we could only utilize coverage and composition in a step wise manner. This means, the errors in cluster due to noise in features would be carried forward to the second clustering step. Hence, the chance for errors getting amplified is rather high. Hence, room for improvement exists in terms of feature aggregation for coverage and composition information. This could in a way eliminate the amplification of error as discussed.

Sampling strategy

Since MetaBCC-LR uses a sampling strategy, the chance of low abundant species being discarded is higher. This is because, sampling data uniformly and treating all the data points equally can result in disappearance of clusters having very small number of data points, *i.e.*, low abundant species. Furthermore, this effect can have a negative impact on the composition based clustering which follows the coverage based clustering step. Therefore, there exists room for improvement in terms of eliminating sampling step, where more robust dimension reduction and clustering techniques will be required.

Chapter 4

Efficient Feature Combination for Metagenomics Binning

The work presented in this chapter was published as

A. Wickramarachchi and Y. Lin (2021a). 'LRBinner: Binning Long Reads in Metagenomics Datasets'. In: *21st International Workshop on Algorithms in Bioinformatics (WABI 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik

A. Wickramarachchi and Y. Lin (2022a). 'Binning long reads in metagenomics datasets using composition and coverage information'. *Algorithms for Molecular Biology*, 17(1), pp. 1–15

This piece of work was presented at The Workshop on Algorithms in Bioinformatics (WABI) 2021 (`https://acm-bcb.org/WABI/2021/`).

The software is freely available at `https://github.com/anuradhawick/LRBinner`.

4.1 Overview and Motivation

In this chapter, we present LRBinner to bin long reads without using any reference databases. In LRBinner we address the limitations that exist in MetaBCC-LR to create an improved long read binning tool. Furthermore, we evaluate LRBinner on more recent and challenging datasets compared to the datasets used for the evaluation of MetaBCC-LR.

In LRBinner we combine the coverage and composition features using the deep learning techniques of variational auto-encoders. This addresses the first limitation of MetaBCC-LR where we had to cluster in a step-wise manner. Indirectly, this approach eliminates the need to sub-sample large datasets. More specifically, LRBinner uses a variational auto-encoder to obtain lower dimensional representations by incorporating both composition and coverage information of the complete dataset. Given that the approach relies on deep learning, the larger scale directly contributes towards better performance of the tool.

In order to perform clustering of the entire dataset at once, LRBinner further uses a distance-histogram-based clustering algorithm. We design the algorithm such that clusters of varying sizes can be extracted efficiently. LRBinner finally assigns unclustered reads to identified clusters using their statistical profiles similar to that in MetaBCC-LR. The experimental results of LRBinner compared against other baselines show that LRBinner achieves better binning results on both simulated and real datasets. Moreover, we show that binning long reads by LRBinner prior to assembly helps to improve genome fraction of assemblies while reducing the memory consumption for metagenomics assembly.

4.2 Methods

LRBinner pipeline primarily addresses the limitations of MetaBCC-LR; (1) combining coverage and composition to eliminate the step wise clustering pipeline, (2) eliminating the need to sample reads which negatively affected low abundant species and (3) perform clustering of the entire dataset at once by implementing a scalable clustering algorithm. LRBinner consists of three main steps; (1) learning lower dimensional latent representations of composition and coverage, (2) clustering the latent representations and (3) obtaining complete clusters.

The complete workflow for LRBinner is illustrated in Figure 4.1. In the following sections, we will explain these three steps in detail.

4.2.1 Feature combination and dimensionality reduction using variational auto-encoders

LRBinner uses two typical binning features of metagenomic sequences, composition and coverage. The composition and coverage of each long read is represented as trimer frequency vectors and k-mer coverage histograms (Wickramarachchi, Mallawaarachchi, Rajan et al., 2020a), respectively.

Computing Composition Vectors

Similar to MetaBCC-LR we use oligonucleotide frequency vectors as composition profiles for binning long reads in LRBinner. Trinucleotide and tetranucleotide frequencies have been used in the past to discriminate assembled contigs of different origins (Alneberg et al., 2014; Kang, Li et al., 2019; Laczny, Kiefer et al., 2017; Pellow, Mizrahi et al., 2020; Wu, Simmons et al., 2015) and Trinucleotide frequencies have been used in metagenomics binning of error-prone long reads (Wickramarachchi, Mallawaarachchi, Rajan et al., 2020a) which shows that trinucleotide frequency vectors provide stable binning despite the noise level that exist in TGS reads. LRBinner utilizes trinucleotide composition vectors. In contrast with MetaBCC-LR, we provide users with the flexibility to use different oligonucleotides for composition profiles in our implementation. This has been mainly motivated by the advent of PacBio HiFi reads which yield sequencing reads with higher base accuracies. We refer to this frequency vector as $V_{composition}$.

Computing Coverage Vectors

From MetaBCC-LR (Section 3.2.1) we adopt the same 15-mer coverage histograms to represent coverage of reads in LRBinner, formally denoted as $V_{coverage}$. We perform our experiments with the same bin_width for the histogram and obtain a vector of bin_count dimensions where we set bin_width=10 and $bins$=32.

4.2.2 Obtaining Latent Representations

For each long read, its coverage ($V_{coverage}$) and composition ($V_{composition}$) vectors are concatenated to form a single vector V of 64 dimensions. We borrow the idea of variational auto-encoders from the recent literature; VAMB (Nissen et al., 2021) to combining these two features. Please note that, VAMB cannot be directly applied for long reads because (1) it is designed for contigs, where the clustering algorithm is not designed to handle large and noisy long read clusters, and (2) requires coverage information by aligning contigs with sequencing reads. However, the ideology of using a variational auto-encoder can be re-used in our problem domain following other applications in machine learning (Pu et al., 2016).

Therefore, we use a variational auto-encoder to obtain lower dimensional latent representations. The key motivation for using a variational auto-encoder is to combine coverage and composition features effectively. In MetaBCC-LR we showed that a simple concatenation of coverage and composition vectors made BH-tSNE less effective. This is mainly because BH-tSNE does not attempt to learn how to effectively combine composition and coverage features, but rather sticks with the spatial distances on concatenated features. However, the variational auto-encoder is able to learn lower dimensional representations by combining both composition and coverage features through a deep neural network.

Implementation of the variational auto encoder

Our implementation of the variational auto-encoder consists of two hidden-layers in the encoder and decoder. Each layer uses batch normalization and dropout with p=0.1 during the training phase. For each input vector V, the auto-encoder learns a latent representation $V_{latent(i)}$, where

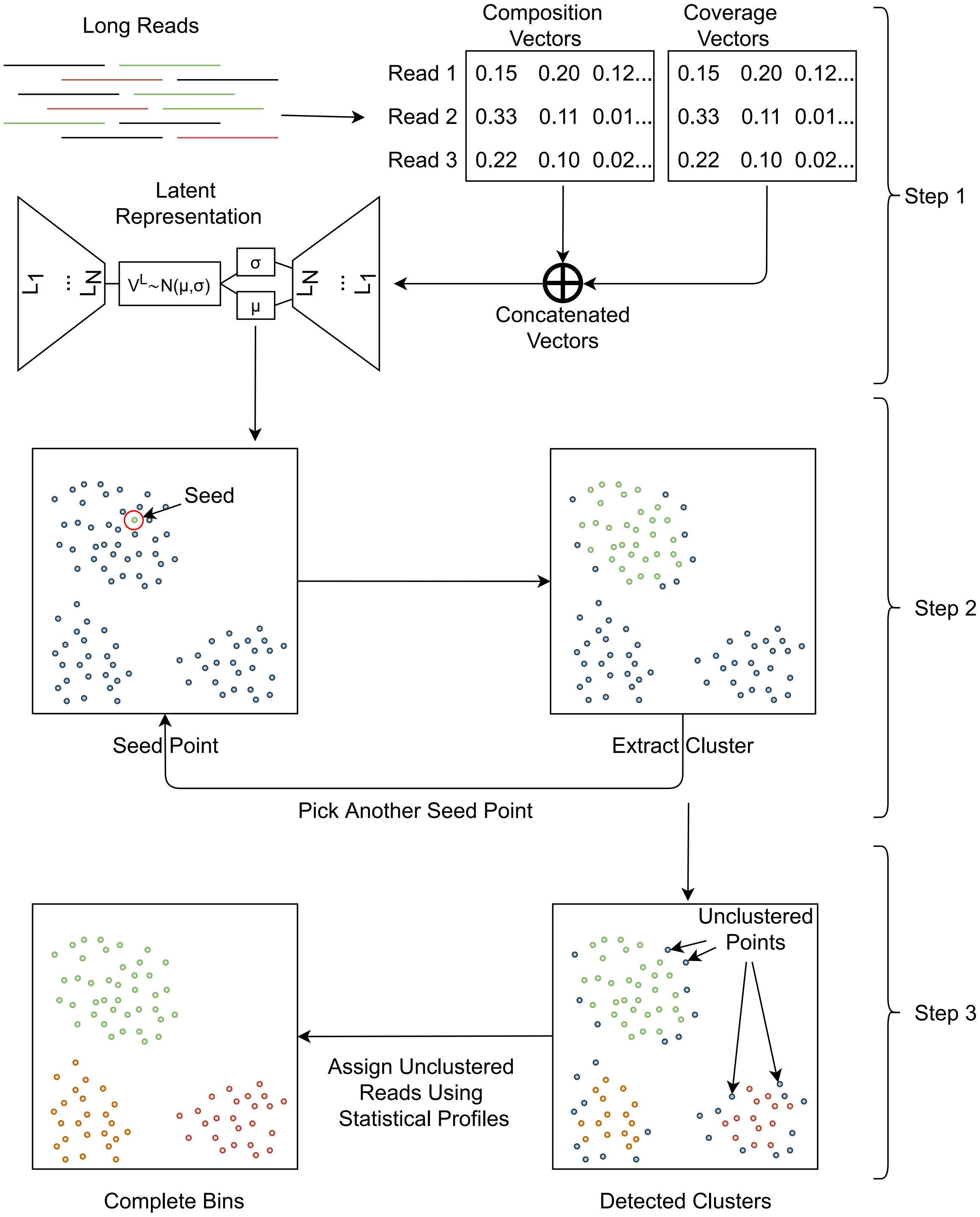


Figure 4.1: Overall workflow of LRBinner. (Step 1) The feature vectors of composition and coverage information are computed from long reads. The feature vectors are fed into a variational auto-encoder to obtain low-dimensional latent representations. (Step 2) Sample a seed point (read) in the latent space and derive a confident cluster (bin) that contains this seed point. Step 2 is iterated until there is no seed point. (Step 3) The unclustered points are assigned to the clusters using a statistical model. Note that the 2-dimensional representation of points is only for the illustration purpose.

$V_{latent(i)} \sim \mathcal{N}(\mu_i, \sigma_i)$. The latent representation consists of 8 dimensions. Each layer in the encoder and decoder contains 128 neurons. Similar to previous studies (Nissen et al., 2021), we use *LeakyRELU* (leaky rectified linear unit function) for μ and *softplus* function for σ layers. Note that μ and σ represent neural network layers intended to learn the lower dimensional means and standard deviations of each read's distribution. We use the weighted sum of reconstruction error E (equation 4.1) and

Kullback–Leibler divergence (Kullback and Leibler, 1951; Nissen et al., 2021) D_{KL} (equation 4.2) as the loss function. E_{cov} and E_{com} represent reconstruction errors of coverage and composition respectively. Equation 4.3 demonstrates the complete loss function used.

$$E = \sum(V_{in} - V_{out})^2 \tag{4.1}$$

$$D_{KL}(latent|prior) = -\sum \frac{1}{2}(1 + ln(\sigma) - \mu^2 - \sigma) \tag{4.2}$$

$$Total\ Loss = w_{cov}E_{cov} + w_{com}E_{com} + w_{kld}D_{KL} \tag{4.3}$$

Here we set w_{cov}=0.1, w_{com}=1 and w_{kld}=1/500 as determined empirically using simulated data. The decoder output was obtained through LeakyRELU activation in order to reconstruct the scaled positive inputs. We train the auto-encoder with read batches of size 10,240 for 200 epochs. Finally, we obtain the predicted latent means of the input data from the encoder for clustering. Each point in the latent mean corresponds to the relevant read in the original input.

4.2.3 Clustering using distance histograms

Distance metric for clustering

In this step, we perform clustering of the latent means learned by the variational auto-encoder. The complete clustering algorithm of LRBinner is illustrated in Figure 4.3. Similar to previous studies (Nissen et al., 2021), we use the cosine distance as the distance measure for clustering. Note that cosine distance, $d(a, b)$ between point a and b in latent space V_{latent} is defined according to the equation 4.4. The key importance in using cosine distance as the metric is that $0 <= d(a, b) <= 1$ is guaranteed. This simplified the distance search space in the distance histograms in the next step.

$$d(a, b) = \frac{V_{latent(a)} \cdot V_{latent(b)}}{||V_{latent(a)}||||V_{latent(b)}||} \tag{4.4}$$

Distance histogram as the point density metric

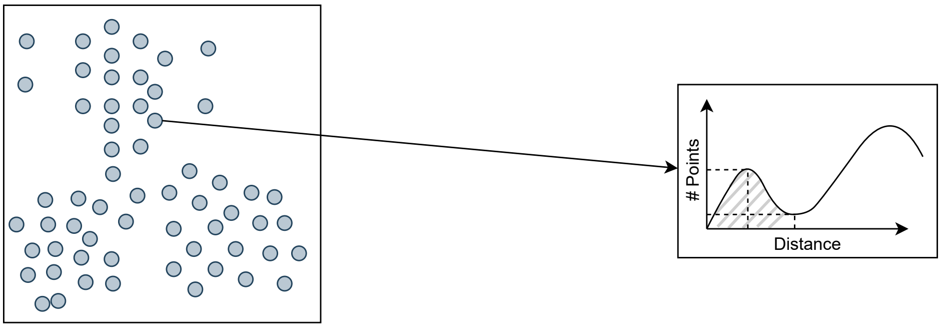


Figure 4.2: Distance histogram for a point

For a point a, we generate a distance histogram H_a by computing the pairwise distances between a and all other points. Note that, for this histogram we term *bin_width* as Δ, where Δ=0.005 for clarity. The resulting histogram is illustrated in the Figure 4.2. In the histogram shown in the right-hand side demonstrates the density around the given point. Note that the origin is hard coded at the 0 density to avoid numerical overflows.

Clustering algorithm

We define *peak* as the index of the first maxima of the distance histogram H_a. Similarly, the *valley* is defined as the index of the first minima after the *peak* in the distance histogram H_a. Refer to the top

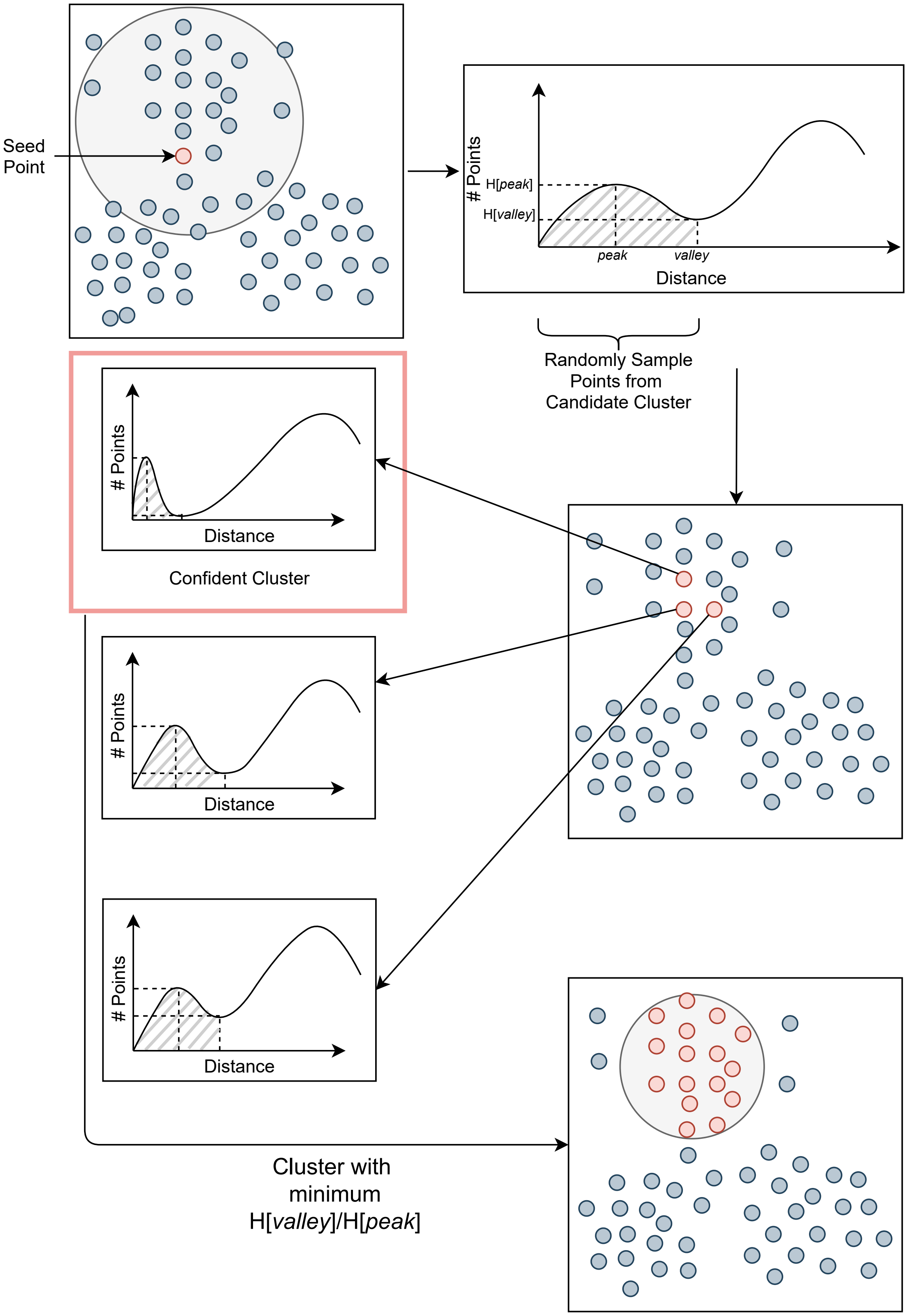


Figure 4.3: Illustration of the clustering algorithm. First select a seed point, generate its distance histogram and derive a *candidate cluster*. Sample from the *candidate cluster points* and choose a point with the minimum valley-to-peak ratio. Extract points before the *valley* to form a *confident cluster*. Note that the 2D representation of points is only for the illustration purposes.

right figure in Figure 4.3 for an example of the *peak* and *valley* in a distance histogram. The shaded region in this diagram is the area of interest here onwards. This is because, this region corresponds to the likely cluster that point a belongs.

Intuitively, a point with smaller valley-to-peak ratio $H[valley]/H[peak]$ is more likely to be the medoid of a cluster (Nissen et al., 2021), where $H[valley]$ and $H[peak]$ are the number of points corresponding to the *valley* and *peak* in the distance histogram H, respectively. In the recent contig binning tool VAMB (Nissen et al., 2021), it randomly samples points, searches within a distance of 0.05 (up to 25 neighboring points) and moves to another point if $H[valley]/H[peak]$ can be further reduced. This step is iterated until a local minimal point of $H[valley]/H[peak]$ is inferred as a proper cluster medoid and then the corresponding cluster is extracted by removing points within a distance $\Delta \times valley$ of the distance histogram. Typically, clusters of contigs or contig bins are of several megabases at maximum leaving with less than few dozens of contigs per bin. Hence, the iterative search for a medoid for the cluster shows promising results at acceptable speeds.

However, clusters of long reads are orders of magnitude larger than clusters of contigs ($1,000-100,000$ or more), thus mere local search of a cluster medoid could be inefficient and time consuming. Furthermore, while most contig clusters consist of hundreds of points per species(Lin and Liao, 2016), the long-read clusters vary in size drastically (from hundreds of points to millions of points), which demand for a more flexible search strategy rather than sampling points within a fixed radius and up to a fixed number of neighbors. Hence, we design the following strategy to dynamically extract clusters of varying sizes. Our algorithm consists of two steps; (1) from a seed point to a candidate cluster and (2) from a candidate cluster to a confident cluster.

(1) From a Seed Point to a Candidate Cluster

A point s is called a *seed point* if its valley-to-peak ratio $H_s[valley]/H_s[peak] < 0.5$ in its distance histogram H_s. Initially, LRBinner randomly picks a seed point s from the entire dataset and obtains its distance histogram H_s. Note that a distance histogram demonstrates a *candidate cluster*. This *candidate cluster* consists of the points within the distance $\Delta \times valley$ in H_s, referred to as *candidate cluster points*. Compared to the seed point, some candidate cluster points may have lower valley-to-peak ratio that result in more confident clusters. However, the number of candidate cluster points may vary significantly depending on the size of the ground-truth clusters. In the next section, we will show how to use sampling strategies to find a confident cluster from a candidate cluster.

(2) From a Candidate Cluster to a Confident Cluster

Given a *candidate cluster*, we sample 10% of candidate cluster points (up to 1,000 points) to compare their corresponding distance histograms. For each point p in *candidate cluster points*, we compute the valley-to-peak ratio $H_p[valley]/H_p[peak]$ in its corresponding distance histogram H_p. We chose a point x from the sample with the minimum $H[valley]/H[peak]$ value and extract a *confident cluster* which consists of points within a distance $\Delta \times valley$ of the distance histogram H_x. In contrast with the iterative medoid search in VAMB (Nissen et al., 2021), this approach takes advantage of the rough estimation of the *candidate cluster* from a seed point and thus allows us to dynamically and efficiently discover clusters with varying sizes. This process is iterated until no further *candidate clusters* or *seed* points are observed. Please refer to Section 4.5 for detailed information. The resulting clusters are depicted as detected clusters in Figure 4.1. Note that few reads still remain unclustered due to the noise present in composition and coverage vectors of error-prone long reads and we will show how to assign them to existing bins in the next section.

Iterative Cluster Discovery

Following the establishment of the clustering algorithm where a cluster center is estimated followed by the cluster extraction, we repeat the two steps until it can no longer be repeated. This process is further illustrated in the Figure 4.4. The number of clusters at the end of the algorithm indicate the number of final bins detected by the LRBinner clustering algorithm. Note that in each iteration, there are few reads left that are further away from the confident cluster region. Such points are illustrated in grey color in the diagram. In the next step, these reads are binned to the identified bins.

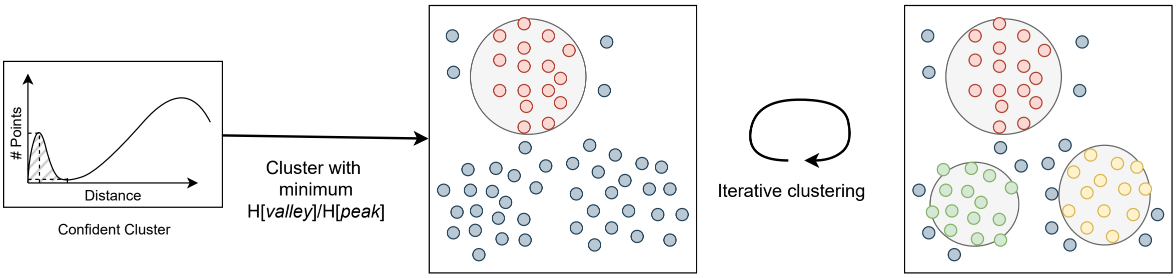


Figure 4.4: Illustration of the iterative clustering algorithm. Steps (1) and (2) are repeated until no more clusters are discovered by the algorithm

4.2.4 Obtaining Final Bins

Once all the clusters have been yielded, the points that are sparsely located are left aside. However, such points could have the potential to improve the downstream assembly processes. Hence, we assign such points to the detected clusters using a statistical model similar to MetaBCC-LR (Wickramarachchi, Mallawaarachchi, Rajan et al., 2020a).

$$PDF(\bar{v},\bar{\mu},\bar{\sigma}) = \prod_{j}^{|\bar{v}|} \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(x_j-\mu_j)^2}{2\sigma_j^2}} \tag{4.5}$$

Finally the unclustered reads are assigned to the bin B_i using a maximum likelihood computed using equation 4.5. The assignment of reads is performed such that equation 4.6 is maximized.

$$B_i = \text{argmax}_i \left\{ PDF(V_{coverage(r)}, \mu_{coverage(i)}, \sigma_{coverage(i)}) \times PDF(V_{composition(r)}, \mu_{composition(i)}, \sigma_{composition(i)}) \right\} \tag{4.6}$$

4.3 Experimental Setup

4.3.1 Datasets

We evaluated LRBinner using several simulated and real datasets containing long reads. Detailed information about the simulated datasets and constituent species are tabulated in the Appendix Tables B.2-B.4. Information about real datasets are available in Appendix Tables B.7.

Simulated Datasets

We simulated four datasets using SimLoRD (Stöcker et al., 2016) to evaluate the performance of our method. The datasets consist of 8, 20, 50 and 100 species. These datasets are named as **Sim-8**, **Sim-20**, **Sim-50** and **Sim-100** respectively. We set the average read length to be 5,000bp with default error model of SimLoRD (insertion probability=0.11, deletion probability=0.04 and substitution probability=0.01).

Real Datasets

In order to evaluate LRBinner, we used several real datasets with known ground-truth references. To determine the origins of the reads in these datasets, the reads were mapped to the respective reference species using Minimap2(Li, 2018a). The information about the datasets are as follows.

- Reads from ATCC SRR9202034 Mock Microbial Community with PacBio CCS reads from NCBI BioProject number *PRJNA546278* (**SRR9202034**). For the evaluation we used the top 10 species which have more than 1% abundance.

- PacBio-HiFi reads obtained from NCBI BioProject number *PRJNA680590*. There are 3 read samples (NCBI BioSample number *SAMN16885726*) and each sample consists of 21 strains for 17 species as follows;
  - **SRX9569057**: Standard input library
  - **SRX9569058**: Low input library
  - **SRX9569059**: Ultra low input library (PCR amplified sample)

4.3.2 Tools for Benchmarking

There is a limited number of tools that support binning of long reads. Remind that most contig-binning tools cannot be directly applied to bin long reads (even for highly accurate PacBio HiFi reads) because there is no coverage information available for each long read. Hence, in our evaluation we use BusyBeeWeb (Laczny, Kiefer et al., 2017) and MetaBCC-LR (Wickramarachchi, Mallawaarachchi, Rajan et al., 2020a) which support error-prone long-reads as the baseline. However, BusyBeeWeb only supports up to 200MB of FASTA formatted data. Hence, in our evaluation we have to provide BusyBeeWeb with a sub-sampled set of reads and evaluated the binning precision and recall on this sub-sampled set.

4.3.3 Evaluation Criteria

In our evaluation we report precision, recall and F1-score (refer to Section 3.3.3) of binning. In order to evaluate the quality of binning, we used AMBER (Meyer, Hofmann et al., 2018a) to obtain the completeness (defined as $\frac{true\ positives_b}{true\ positives_b + false\ negatives_b}$ for each bin b) and contamination (defined as $1 - \frac{true\ positives_b}{true\ positives_b + false\ positives_b}$ for each bin b). Please note that we only compare AMBER results of MetaBCC-LR and LRBinner as BusyBeeWeb does not bin the entire datasets due to limited input size. Furthermore, we assemble the reads before and after binning using LRBinner. Metagenomics assemblies were performed using wtdbg2 (Ruan and Li, 2020a) and metaFlye (Kolmogorov, Bickhart, Behsaz, Gurevich, Rayko, Shin, Kuhn, Yuan, Polevikov, Smith and Pevzner, 2020). We compare genome fractions, CPU-time and memory usage in assembly evaluation. We used MetaQUAST (Mikheenko et al., 2015a) to obtain the genome fraction (average percentage of bases aligned per reference genome) for the qualitative evaluation of assembled contigs.

4.4 Results and Discussion

We first compare precision, recall, F1 score and the estimated number of bins for binning performance. We further present the completeness and contamination results of bins produced by different binners. Furthermore, we evaluate assembly results using genome fraction and recorded the resource utilization for the chosen assembly tools.

4.4.1 Binning Results

We benchmarked the binning performance for BusyBeeWeb, MetaBCC-LR and LRBinner. Table 4.1 demonstrates the binning results in terms of precision, recall, F1-score and the number of inferred bins. While BusyBeeWeb, MetaBCC-LR and LRBinner perform comparably on simulated datasets, LRBinner achieves the best estimation on the number of bins with respect to the ground truth for most of the datasets. As BusyBeeWeb has a limitation of input data size (200Mb), its binning accuracy deteriorates on the large real datasets due to its limited access to the complete dataset. Although BusyBeeWeb demonstrates good performance in simulated datasets the performance degrades rapidly in real datasets. This is especially true when Sim-50 and Sim-100 where the coverage variation is low compared to real datasets with large variation in abundances. This results in uniform clusters of reads when sampled for BusyBeeWeb input which is not the case for real datasets. Note that LRBinner improves binning results for all the real datasets as indicated by the higher F1 scores.

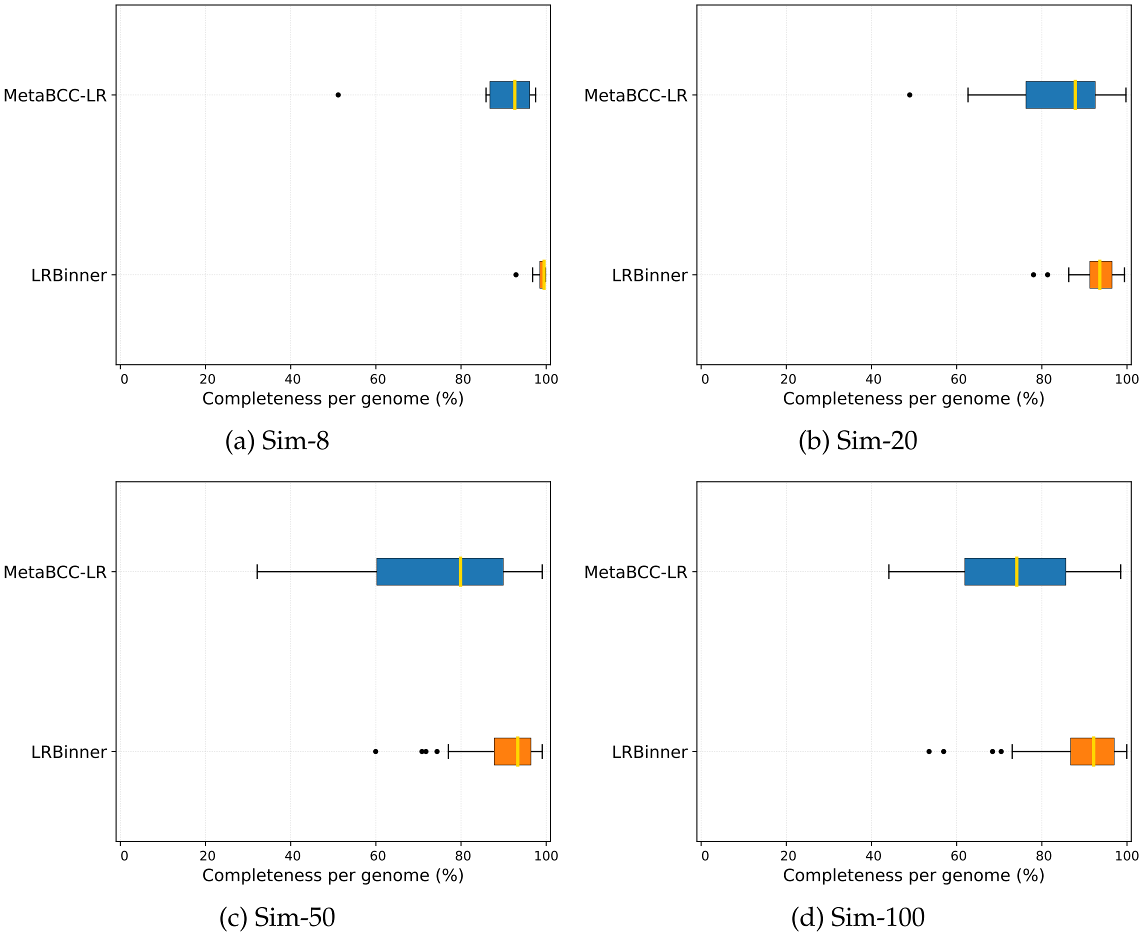
Figures 4.5 and 4.6 illustrates the completeness of bins produced by MetaBCC-LR and LRBinner, for simulated and real datasets respectively. Note that BusyBeeWeb is not included in this comparison

Table 4.1: Comparison of binning results of BusyBeeWeb, MetaBCC-LR and LRBinner.

Dataset	Actual No. of Bins	Evaluation Criteria	BusyBeeWeb †	MetaBCC-LR	LRBinner
Sim-8	8	Precision	90.41%	90.78%	**99.14%**
		Recall	**99.80%**	96.18%	99.14%
		F1 score	94.87%	93.40 %	**99.14%**
		Bins Detected	50	13	**8**
Sim-20	20	Precision	**95.88%**	82.97%	90.53%
		Recall	**97.99%**	81.95%	88.23%
		F1 score	**96.92%**	82.46%	89.36%
		Bins Detected	30	29	**18**
Sim-50	50	Precision	**94.15%**	82.23%	91.92%
		Recall	**94.34%**	70.56%	77.03%
		F1 score	**94.24%**	75.95%	83.82%
		Bins Detected	**59**	32	31
Sim-100	100	Precision	**95.48%**	90.50%	82.60%
		Recall	87.85%	84.54%	**92.78%**
		F1 score	**91.50%**	88.54%	87.39%
		Bins Detected	75	**89**	63
SRR9202034$^{\gamma}$	10	Precision	68.30%	93.69%	**95.30%**
		Recall	81.96%	95.50%	**95.99%**
		F1 score	74.51%	94.59%	**95.64%**
		Bins Detected	87	14	**10**
SRX9569057	17	Precision	48.63%	**80.94%**	80.47%
		Recall	72.68%	85.82%	**90.68%**
		F1 score	58.27%	83.31%	**85.27%**
		Bins Detected	111	23	**16**
SRX9569058	17	Precision	23.01%	70.18%	**73.72%**
		Recall	32.64%	86.63%	**91.03%**
		F1 score	26.99%	77.54%	**81.46%**
		Bins Detected	117	37	**22**
SRX9569059	17	Precision	65.70%	66.69%	**79.70%**
		Recall	**95.36%**	73.76%	91.25%
		F1 score	77.80%	70.05%	**85.08%**
		Bins Detected	124	**16**	20

† BusyBeeWeb is only evaluated on the sampled reads due to the input size limitation of the tool.
γ Only the top 10 abundant species are considered (More than 1% of abundance).

as it cannot handle the entire dataset in most cases which makes it incompatible for the AMBER evaluation. LRBinner has been able to produce bins with better average completeness over MetaBCC-LR. Figure 4.7 and 4.8 also illustrates the contamination levels of bins produced by MetaBCC-LR and



(a) Sim-8

(b) Sim-20

(c) Sim-50

(d) Sim-100

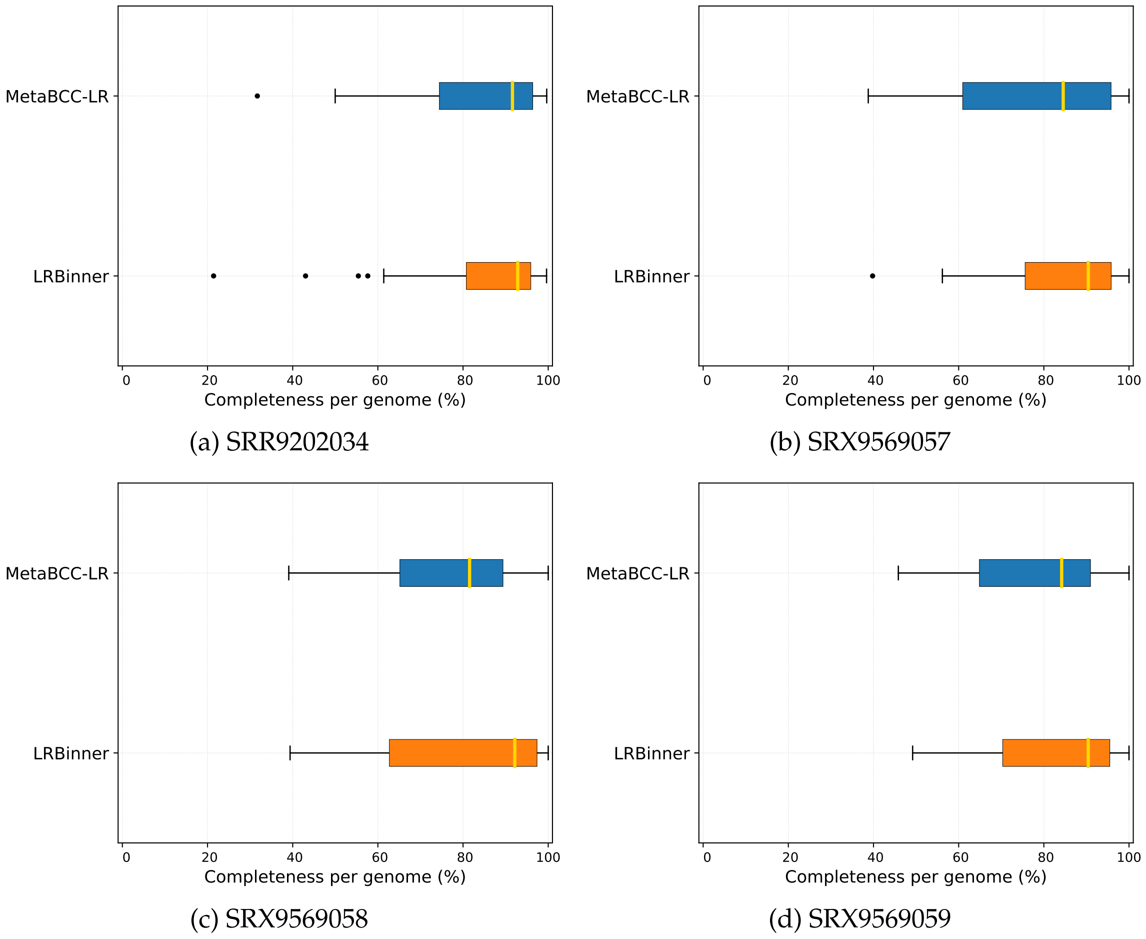
Figure 4.5: Comparison of bin completeness between MetaBCC-LR and LRBinner for the simulated datasets.

LRBinner, for simulated and real datasets respectively. From the plots it is evident that LRBinner produces bins with lower contamination in all datasets except for **SRX9569059**. Note that the dataset **SRX9569059** has been generated from a PCR amplified sample leading to a significant deviation from the original sample abundances in contrast with **SRX9569057** and **SRX9569058** datasets. For example, in **SRX9569059**, the abundance of *Faecalibacterium prausnitzii* drops from $\sim$ 16% to $\sim$ 8% whereas the abundance of *Fusobacterium nucleatum* surges from $\sim$ 4% to $\sim$ 7%, which may result in contamination of long reads in binning results.

4.4.2 Assembly Results

We assembled the reads binned by LRBinner to evaluate the potential assembly quality changes. For the assembly, we chose the two state-of-the-art long-read assemblers wtdbg2 (Ruan and Li, 2020a), version 2.5 and metaFlye (Kolmogorov, Bickhart, Behsaz, Gurevich, Rayko, Shin, Kuhn, Yuan, Polevikov, Smith and Pevzner, 2020), available in Flye v2.4.2. Table 4.2, demonstrates that binning long reads prior to assembly by LRBinner improves the genome fraction for all wtdbg2 assemblies (up to 40%) and maintains comparable genome fractions for metaFlye assemblies. This is not surprising as metaFlye is a metagenomics specialized assembler in contrast with wtdbg2. For example, in the datasets SRX9569057, SRX9569058 and SRX9569059, binning via LRBinner enabled wtdbg2 to recover low-abundance species which were ignored in the assembly of the entire raw dataset, *e.g.*, *Methanobrevibacter smithii* (from 0 to 96%), *Saccharomyces cerevisiae* (from 0 to 75%) and *Candida albican* (from 0 to 70%). This is because LRBinner allows wtdbg2 to estimate more appropriate parameters in each bin rather than applying the same parameters across the entire dataset.

Another advantage of binning prior to assembly is the reduction of the computing resources for assembly. As demonstrated in Table 4.2, the peak-memory usage has been drastically reduced in both



(a) SRR9202034

(b) SRX9569057

(c) SRX9569058

(d) SRX9569059

Figure 4.6: Comparison of bin completeness between MetaBCC-LR and LRBinner for the real datasets.

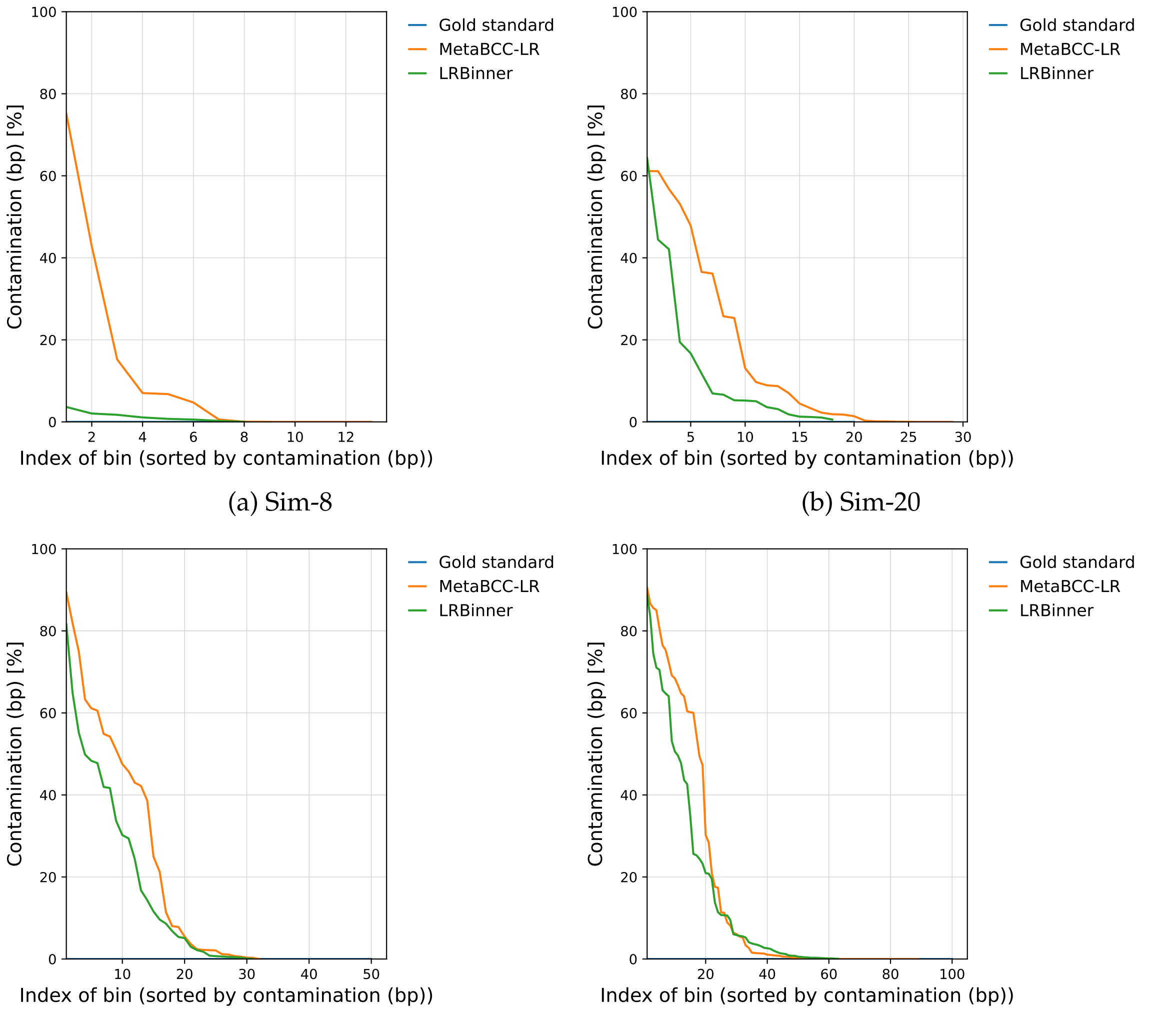
wtdbg2 (up to 10×) and metaFlye (up to 4×) assemblies. Note that the CPU time is comparable as binning long reads may not lead to significant reduction of *k*-mer indexing cost and the construction and simplification of assembly graphs.

4.4.3 Effect of chosen oligonucleotide composition on binning

We conducted experiments to evaluate the effect of varying oligonucleotide frequency vector size in composition vectors to see the impact on the binning performance. We tested tri-, tetra- and penta-nucleotide frequencies where the composition vectors took size 32, 136 and 512 dimensions. The results are tabulated in the Table 4.3. It is reasonably evident that increased vector size produced false bin numbers compared to smaller ones. This is because, increase in vector size makes the composition vectors more amenable to errors due to erroneous *k*-mers. Moreover, increased size of the composition vectors can lead biases in the auto-encoder training phase and require rigorous parameter tuning to compensate such biases.

4.4.4 Effect of changes in choice of *k*-mer on resource utilization

Table 4.3 also tabulates the change in performance requirements for different *k* values chosen for the composition vectors. Intuitively, we see an increasing trend in the resource usage. In datasets **Sim-100** and **SRX9569058** the memory consumption is closer to 40 GB which renders LRBinner non-functional in desktop class computers. Although not severely affected, the CPU hours consumed follows a similar trend.



(a) Sim-8

(b) Sim-20

(c) Sim-50

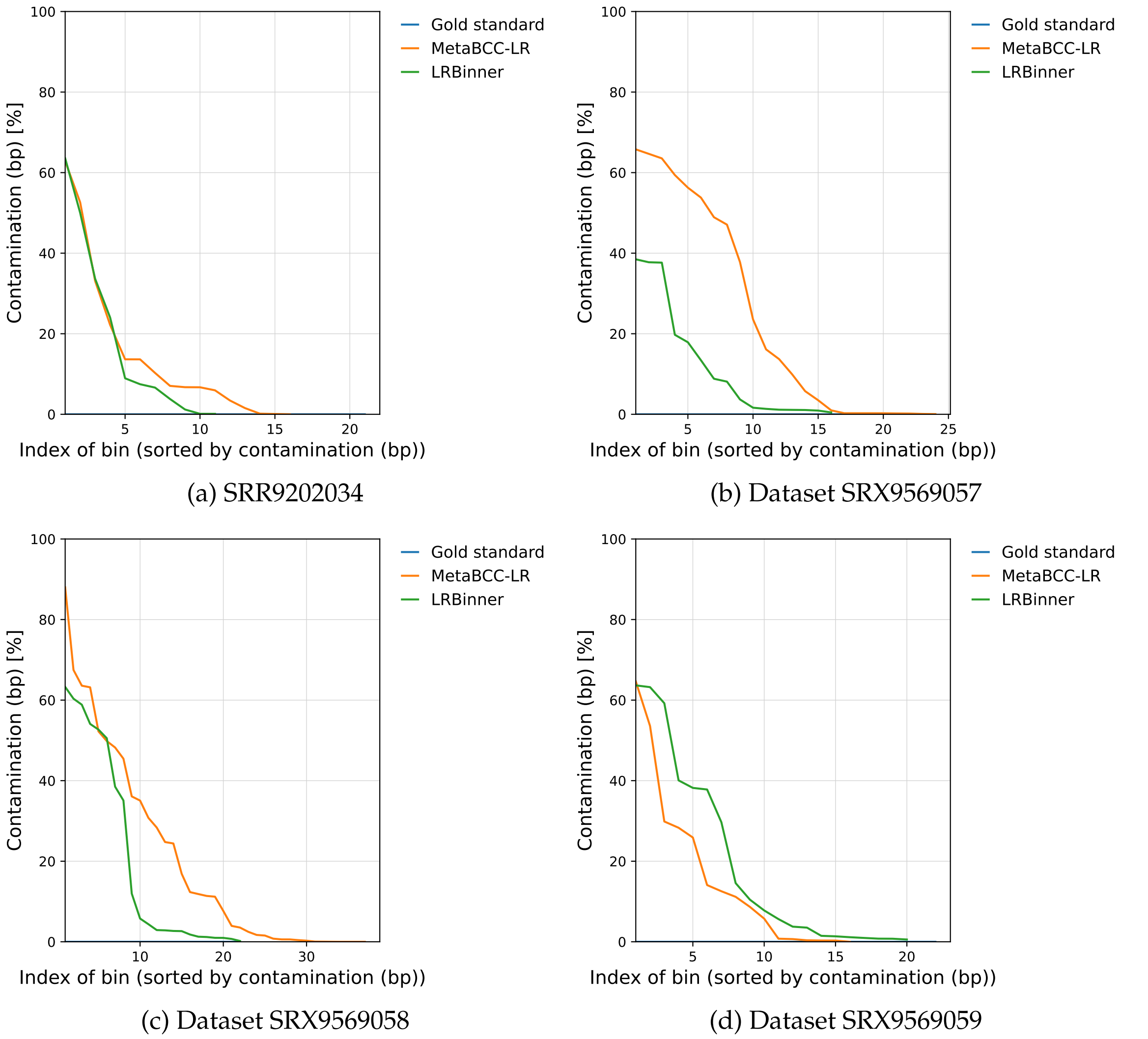
(d) Sim-100

Figure 4.7: Comparison of bin contamination between MetaBCC-LR and LRBinner for the simulated datasets.

4.5 Implementation

In order to restrict the iterative search for clusters, we use early termination parameters in our algorithm. We stop drawing seed points when the remaining number of reads reaches below *min_cluster_size* (=5000 by default) or the number of iterations has passed *max_iterations* (=1000). We evaluated the performance of LRBinner with varying size of the composition vectors. The related resource utilization and binning accuracy are tabulated in Table 4.3. The GPU utilization was below 4GB during all the experiments due to fixed batch size of 1024 reads. Coverage vectors were fixed at *bins*=32 and *bin_width*=10.

LRBinner was implemented using C++ and Python version 3.7. The deep learning component is implemented using PyTorch (Paszke et al., 2019) and Numpy (Harris et al., 2020). We conducted our assemblies on NCI Australia with 2 x 28-core Intel Xeon Platinum 8274 (Cascade Lake) 3.2 GHz CPUs 192 GB RAM and binning on Ubuntu 20.04.3 LTS system running on AMD Ryzen 9 5950X with 16-core processor with 32 threads and 128 GB of RAM with NVIDIA RTX 3090 GPU with 24 GB VRAM. We used 56 threads for assembly and 32 threads for binning with GPU acceleration.



(a) SRR9202034

(b) Dataset SRX9569057

(c) Dataset SRX9569058

(d) Dataset SRX9569059

Figure 4.8: Comparison of bin contamination between MetaBCC-LR and LRBinner for the real datasets.

4.6 Discussion

4.6.1 Summary

We presented LRBinner, a long read binner capable of binning error-prone long reads using both coverage and composition information. Our work extends the use of variational auto-encoders to combine raw features and learn a better latent representation for long-read binning. Furthermore, we presented a novel clustering strategy that can perform clustering on large datasets with varying cluster sizes. Performance of LRBinner was evaluated against existing long-read binners using simulated and real datasets. Our experimental results show that LRBinner outperforms state-of-the-art long-read binning tools and also improves resource usage of downstream assembly.

4.6.2 Room for improvements

LRBinner possess several limitations. (1) Although, the use of entire dataset at once can provide better clustering of long reads, this imposes a major challenge. Since metagenomic datasets are imbalanced in nature, *i.e.*, species have varying genome sizes and abundances. This means there could be large clusters that can present peaks in distance histograms that overthrow low abundant clusters. This is because, in the distance histograms, small clusters could be seen as noise or their peaks could be hidden. Hence, an approach that would somehow balance the clusters in metagenomic datasets could improve the binning performance. (2) Estimation of coverage using *k*-mer coverage histograms can be

Table 4.2: Comparison of assembled genome fractions, CPU time consumed for assembly and peak memory usage of assembly before and after binning the reads.

Dataset	Assembly Tool	Genome Fraction		CPU Hours		Peak Memory (GB)	
		Raw	Binned	Raw	Binned	Raw	Binned
Sim-8	wtdbg2	98.80%	**98.90%**	**0.26**	0.84	9.28	**0.96**
	metaFlye	**99.90%**	99.85%	16.13	**11.64**	44.12	**10.65**
Sim-20	wtdbg2	97.84%	**99.19%**	**0.16**	2.28	10.60	**0.92**
	metaFlye	**99.80%**	99.75%	**19.44**	20.28	44.70	**11.23**
Sim-50	wtdbg2	97.83%	**98.06%**	6.03	**5.98**	15.7	**2.68**
	metaFlye	**99.35%**	98.43%	23.03	**20.01**	64.21	**14.58**
Sim-100	wtdbg2	91.70%	**93.67%**	9.2	**9.1**	36.16	**10.84**
	metaFlye	97.68%	**98.01%**	69.79	**59.89**	116.11	**27.48**
SRR9202034 †	wtdbg2	67.45%	**82.50%**	**0.31**	1.05	23.43	**19.61**
	metaFlye	91.40%	**91.74%**	**155.96**	158.59	62.28	**45.38**
SRX9569057	wtdbg2	40.40%	**73.02%**	**0.26**	1.56	21.72	**3.88**
	metaFlye	**77.73%**	73.68%	122.00	**116.20**	57.91	**26.31**
SRX9569058	wtdbg2	37.51%	**80.65%**	**0.30**	1.98	30.79	**3.86**
	metaFlye	79.16%	**79.63%**	**211.61**	212.58	87.62	**41.37**
SRX9569059	wtdbg2	41.00%	**80.38%**	**0.26**	1.82	25.63	**3.80**
	metaFlye	**79.69%**	77.46%	152.64	**129.41**	62.62	**30.56**

† Genome fraction computed using the top 10 species with at least 1% abundance.

inaccurate due to the presence of shared *k*-mers and erroneous *k*-mers. Moreover, the binning could benefit from a more accurate estimation of coverage beyond a pure *k*-mer driven approach. Because of these limitations LRBinner cannot fully recovery the low abundant species.

Table 4.3: Performance of LRBinner with varying k-sizes for composition vectors.

Dataset	k-size	No. of Bins	Precision	Recall	F1 score	CPU Hours	Peak Memory (GB)
Sim-8	3	8	99.14%	99.14%	99.14%	0.72	4.52
	4	10	97.22%	96.25%	96.73%	0.79	5.51
	5	10	97.18%	98.47%	97.82%	0.79	9.29
Sim-20	3	18	90.53%	88.23%	89.36%	1.06	4.83
	4	20	91.58%	91.58%	91.58%	1.11	6.34
	5	21	93.11%	95.96%	94.51%	1.14	12.16
Sim-50	3	31	82.60%	92.78%	87.39%	1.70	5.46
	4	42	92.78%	87.00%	89.79%	1.77	7.96
	5	45	93.72%	88.63%	91.11%	1.82	19.06
Sim-100	3	63	82.60%	92.78%	87.39%	4.12	7.95
	4	57	92.87%	82.13%	87.17%	4.39	14.59
	5	55	93.22%	81.05%	86.71%	4.37	42.27
SRR9202034	3	10	95.30%	95.99%	95.64%	3.31	7.09
	4	23	91.71%	96.66%	94.12%	3.38	12.35
	5	33	94.84%	97.72%	96.26%	3.49	33.70
SRX9569057	3	16	80.47%	90.68%	85.27%	2.98	6.59
	4	30	82.07%	90.92%	86.27%	3.03	11.00
	5	38	80.18%	95.07%	86.99%	3.14	28.56
SRX9569058	3	22	73.72%	91.03%	81.46%	1.65	7.66
	4	30	81.88%	90.80%	86.11%	2.34	13.81
	5	42	80.44%	95.80%	87.45%	4.41	39.28
SRX9569059	3	20	79.70%	91.25%	85.08%	1.62	7.25
	4	29	83.18%	89.42%	86.19%	2.46	12.76
	5	56	77.94%	96.02%	86.04%	3.90	35.32

Chapter 5

Read-Overlap Graph for Binning Long Reads

The work presented in this chapter was published as

A. Wickramarachchi and Y. Lin (2022b). 'Metagenomics Binning of Long Reads Using Read-Overlap Graphs'. In: *RECOMB International Workshop on Comparative Genomics*. Springer, pp. 260–278

This piece of work was presented at the The 19th Annual Satellite Conference of RECOMB on Comparative Genomics (RECOMB-CG) conference 2022.
`https://recombcg2022.usask.ca/pages/program/`

The software is freely available at `https://github.com/anuradhawick/OBLR`.

5.1 Overview and Motivation

In this chapter, we propose a novel binning approach named **OBLR** to bin long reads using the read-overlap graph. This approach intends to address the two main limitations of LRBinner (1) imbalanced clusters and (2) inaccurate estimation of coverage. In order to address these limitations we introduce the idea of read-overlap graph. In OBLR, we compute the read-overlap graph using an efficient mapping tool such as **kbm2**. We use the read-overlap graph obtained to estimate the coverage of reads based on the degree in the graph. Furthermore, using the degree as a probabilistic measure, we down sample the datasets to obtain a version that has more balanced clusters.

Using a simulated sample dataset we show that the degree of the read-overlap graph has a correlation with the underlying coverage of the reads' species. Moreover, we show that when the dataset is down-sampled using the degree of the nodes as a probabilistic metric, we can obtain a balanced version of the dataset. Following the sampling, we detect the number of clusters dynamically by observing the clustering of several sample sizes scored using the Silhouette score. We use UMAP for dimension reduction and HDBSCAN as the clustering algorithm to detect the clusters. At the end, we use GraphSAGE to label the entire read-overlap graph using the identified clusters in the dataset.

We conduct several experiments using simulated and real datasets to evaluate the performance of OBLR. We show that OBLR not only performs well in binning but also enables to improve the genome fractions of assembled metagenomic datasets following the binning process. The implementation of OBLR enables the efficient computation of the read-overlap graph using a read blocking strategy and the rest of the pipeline is implemented using GPU accelerated implementations of algorithms.

5.2 Methods

Our pipeline consists of 5 steps performing the tasks, (1) building the read-overlap graph, (2) obtaining read features, (3) performing probabilistic sampling, (4) detecting clusters for sampled reads and (5) binning remaining reads by inductive learning. Figure 5.1 illustrates the overall pipeline of OBLR. The following sections explain each step in detail.

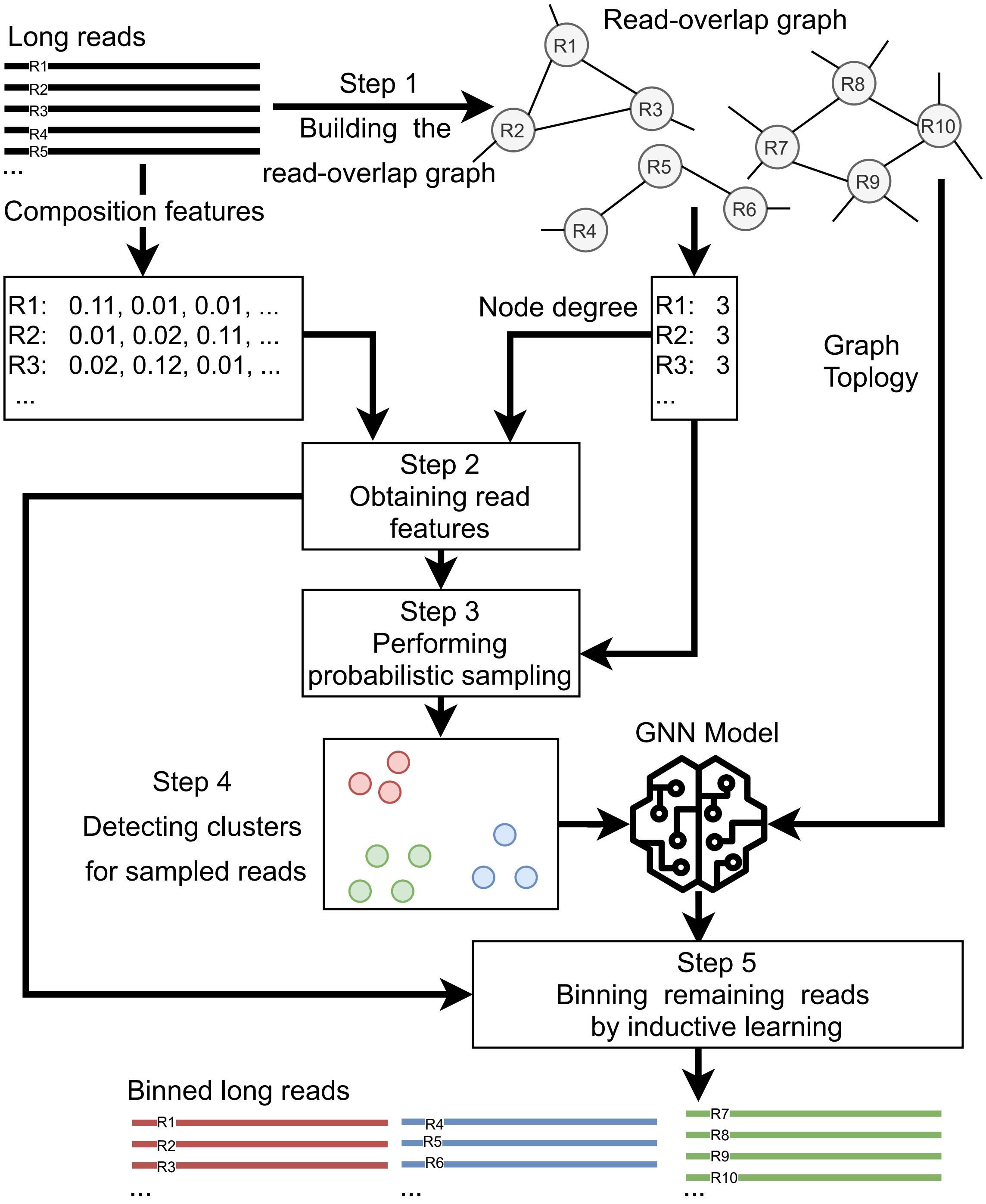


Figure 5.1: An overview of the workflow of the proposed pipeline OBLR.

5.2.1 Constructing Read-Overlap Graph

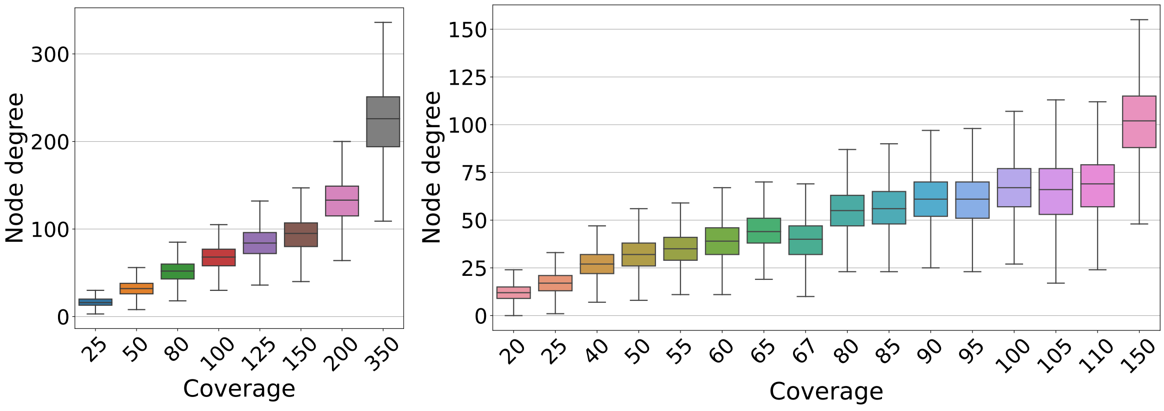
The read-overlap graph is introduced to demonstrate the overlapping information between raw reads. Two raw reads are overlapping (connected by an edge in the read-overlap graph) iff their overlapping length is at least $L_{overlap}$ and the overhang length is at most $L_{overhang}$ (computed according to (Li, 2016)). In our pipeline, we use k-mer bin map (kbm2) program to compute the approximate overlaps between reads and set $L_{overlap} = 2560$ and $L_{overhang} = 512$ in the default setting. Note that kbm2 is a sub-routine of the recent assembler wtdbg2 and is extremely fast to detect overlapping reads using k-mer bins without performing pairwise alignment (Ruan and Li, 2020b). In the read-overlap graph, each node R_i represents a read while each edge (R_i, R_j) indicates that R_i and R_j are overlapping. We also define $D(R_i)$ as the degree of R_i in this read-overlap graph.

5.2.2 Obtaining Read Features

In our pipeline, we intend to derive read features that incorporate both composition and coverage information of long reads.

The composition information of long reads can be computed as their oligonucleotide frequencies which are shown to be conserved within a given species while being reasonably distinct between species (Strous et al., 2012; Wu, Simmons et al., 2015). More specifically, we compute a tetra-nucleotide frequency vector for each long read R_i, *i.e.*, $X(R_i) \in \mathbb{R}^{136}$ as there are 136 distinct tetra-mers when combining reverse complements.

The coverage information of long reads usually refers to the coverage of underlying genomes from which the long reads are drawn. This is also important in metagenomics binning as long reads from the same species tend to have similar coverages (Nissen et al., 2021; Wickramarachchi, Mallawaarachchi, Rajan et al., 2020a). While such coverage information is usually available for contigs assembled from short reads (as a byproduct of assembly), long reads do not come with their coverage information. However, a read from a high-coverage genome is likely to have more overlaps compared to that from a low-coverage genome. Therefore, it is a natural choice to use the node degree in the read-overlap graph to estimate the coverage of the corresponding read. This choice is supported by Figure 5.2 which shows a clear correlation between the node degree in the read-overlap graph and the coverage information of the corresponding read.



(a) Sim-8

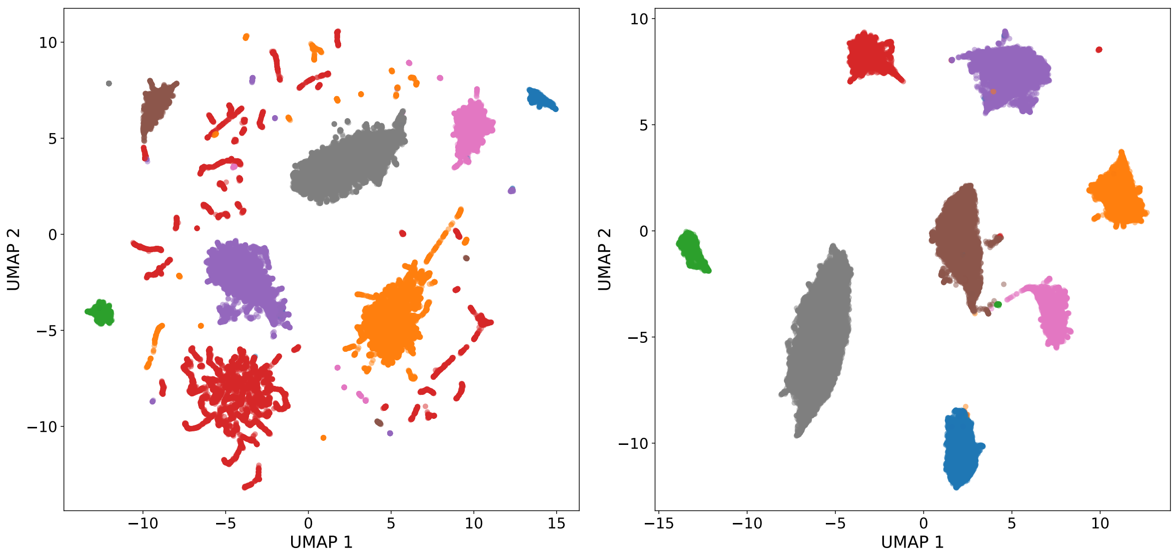
(b) Sim-20

Figure 5.2: The correlation between the node degree in read-overlap graph and the coverage information of the corresponding read for **Sim-8** and **Sim-20** datasets.

In summary, we combine both tetra-nucleotide frequency vector $X(R_i)$ (for composition) and the node-degree information $D(R_i)$ (for coverage) to derive the read feature vector as $XD(R_i) = X(R_i) \times max(1, lg(D(R_i)))$ for each long read R_i. Note that the $max(\ )$ and $lg(\ )$ are introduced to dampen rigorous fluctuations in coverage, especially for low-coverage genomes. Henceforth, $XD(R_i)$ refers to the read features using degree and composition for read R_i.

5.2.3 Performing Probabilistic Sampling

While it is now possible to bin all long reads using their read feature vectors, such a clustering will suffer from the well-known class-imbalance problem (Japkowicz and Stephen, 2002) because metagenomic samples consist of species with varying coverages resulting in a significant variation of the cluster sizes. It has been shown that probabilistic sampling of data can effectively address this problem by generating datasets of similar cluster sizes (Liu et al., 2009). The remaining challenge is how to perform probabilistic sampling on long-read datasets and derive more uniform clusters of long reads across species of varying abundances, *i.e.* low probabilities to sample long reads from high-coverage genomes and vice versa. We recall the degree information of nodes and use the equation 5.1 to compute the relative probability of sampling R_i.



(a) Uniform sampling

(b) Probabilistic sampling

Figure 5.3: Comparison of uniform sampling and probabilistic sampling of long reads in **Sim-8** dataset. Different colors corresponds to reads that belong to a unique species.

$$P(R_i) = \begin{cases} \frac{1}{D(R_i)} & \text{if} \quad D(R_i) \neq 0 \\ 0 & \text{if} \quad D(R_i) = 0 \end{cases} \tag{5.1}$$

Note that $D(R_i) = 0$ when R_i is an isolated node in the read-overlap graph and this is very helpful for OBLR to distinguish reads of low-abundance species from noisy or chimeric reads. Figure 5.3 clearly demonstrates that the probabilistic sampling can derive more balanced clusters of long reads compared to uniform sampling, and thus results in better cluster separations.

5.2.4 Detecting Clusters for Sampled Reads

UMAP (McInnes, Healy and Melville, 2020) is used to project the sampled reads into lower dimensions followed by clustering using HDBSCAN (McInnes, Healy and Astels, 2017). To accommodate long-read datasets with different relative species abundances, the sample size is set to be $25,000$, $50,000$ and $100,000$ and the Silhouette score (Rousseeuw, 1987) is used to determine the best clustering result in an unsupervised manner. Afterwards, sampled reads are binned into clusters followed by binning the remaining reads into the identified clusters.

5.2.5 Binning Remaining Reads by Inductive Learning

OBLR employs GraphSAGE (Hamilton et al., 2017) to bin the remaining reads into the identified clusters in the previous step. GraphSAGE is a Graph Neural Network (GNN) architecture and has been designed to perform inductive learning using large-scale graphs (Hamilton et al., 2017). GraphSAGE can be represented as a layer in a GNN that aggregates the neighborhood features to represent the features of a node itself. Formally, the l-th layer can be formulated according to equations 5.2 and 5.3 (Xu et al., 2018).

$$a_i^{(l)} = \text{Mean}(h_j^{(l-1)} : j \in \mathcal{N}(R_i)) \tag{5.2}$$

$$h_v^{(l)} = \text{Concatenation}(h_v^{(l-1)}, a_v^{(l)}) \tag{5.3}$$

where $h_i^{(l)}$ is the feature vector of node R_i at layer l. Note that $h_v^{(0)}=XD(R_i)$ and $\mathcal{N}(R_i)$ represent neighbors of node R_i. While GraphSAGE supports arbitrary aggregate functions, we choose the $Mean(\ )$ as the aggregation operator to be tolerant towards noise and false connections in the read-overlap graph due to repeats. Furthermore, we use $Concatenation(\ )$ as the layer-wise feature combination strategy to retain features from both the node itself and its neighborhood.

We use two GraphSAGE layers followed by a fully-connected layer with K outputs, where K is the number of bins estimated in Step (4). Two GraphSAGE layers use LeakyRELU activation while the final layer uses *log softmax* activation resulting in the output probabilities for K bins. We train the GNN using 200 epochs using sampled reads binned in Step (4) and use negative log-likelihood (cross-entropy) as the loss function. During the training phase, we use a neighbor sampler that samples up to 20 neighbors in GraphSAGE layers. We use Adam optimizer for gradient descent. The trained GNN on sampled reads provides assignment probabilities for remaining reads to the bins derived in Step (4). The remaining reads are thus assigned to the bins with the highest probabilities.

5.3 Experimental Setup

We evaluate our pipeline using four simulated and two real datasets. Detailed dataset information for simulations are available in Appendix Tables B.3, B.4 and B.2. Information about real datasets are available in Appendix Table B.7.

5.3.1 Simulated Datasets

We simulate four PacBio datasets using SimLoRD (Stöcker et al., 2016) containing 8, 20, 50 and 100 (Wickramarachchi, Mallawaarachchi, Rajan et al., 2020a; Wu, Tang et al., 2014) species with average read length 5,000base pairsand the default PacBio error profiles in SimLoRD (insertion=0.11, deletion=0.04 and substitution=0.01). These datasets are named as **Sim-8**, **Sim-20**, **Sim-50** and **Sim-100** respectively.

5.3.2 Real Datasets

Two real datasets with known reference genomes are also used to evaluate read-level binning performance. Long reads from these datasets were aligned references using Minimap2 (Li, 2018b) to obtain the ground truth.

- **ZymoEVEN**: Oxford Nanopore reads sequenced from GridION device from NCBI Accession Number *ERR3152364*(Nicholls et al., 2019). The dataset consists of 10 species with average read length 4,119.
- **SRR9202034**: PacBio CCS reads of the ATCC MSA-1003 Mock Microbial Community from NCBI BioProject number *PRJNA546278* Accession Number *SRR9202034*. The dataset contains 15 species with more than 0.1% relative abundance and average read length 8,263.
- **SRX9569057**: PacBio-HiFi reads of the NCBI BioSample *SAMN16885726* Accession Number *SRX9569057*. This dataset contains 13 species (18 strains) with more than 0.1% relative abundance and average read length 9,093.

Table 5.1 summarizes the information about the datasets including the number of species, dataset sizes and the number of nodes (reads) and edges of the read-overlap graphs.

5.3.3 Baselines and Evaluation Criteria

We benchmark our approach against two recent long-read binners, MetaBCC-LR (Wickramarachchi, Mallawaarachchi, Rajan et al., 2020a) and LRBiner (Wickramarachchi and Lin, 2021a). The binning results are evaluated using precision, recall and F1-score (refer to Section 3.3.3). We also used AMBER (Meyer, Hofmann et al., 2018b) to compute completeness and purity for derived bins. Long-read metagenomics assemblies before and after binning are performed using metaFlye (Kolmogorov, Bickhart, Behsaz, Gurevich, Rayko, Shin, Kuhn, Yuan, Polevikov, Smith et al., 2020) version 2.9-b1774. The genome fractions (computed by MetaQUAST (Mikheenko et al., 2016)), CPU-time and memory usage are used to evaluate assembly performance before and after binning.

Table 5.1: Summary of the datasets

Dataset	No. of species	Dataset size (GB)	No. of nodes	No. of edges
Sim-8	8	3.5	432,333	47,984,545
Sim-20	20	5.3	666,735	42,642,457
Sim-50	50	9.5	1,119,439	86,245,400
Sim-100	100	24.6	2,991,815	1,198,753,181
ZymoEVEN	10	8.2	1,688,672	611,447,694
SRR9202034	15	19.5	2,358,257	2,105,962,083
SRX9569057	13	18.0	1,978,852	1,421,138,836

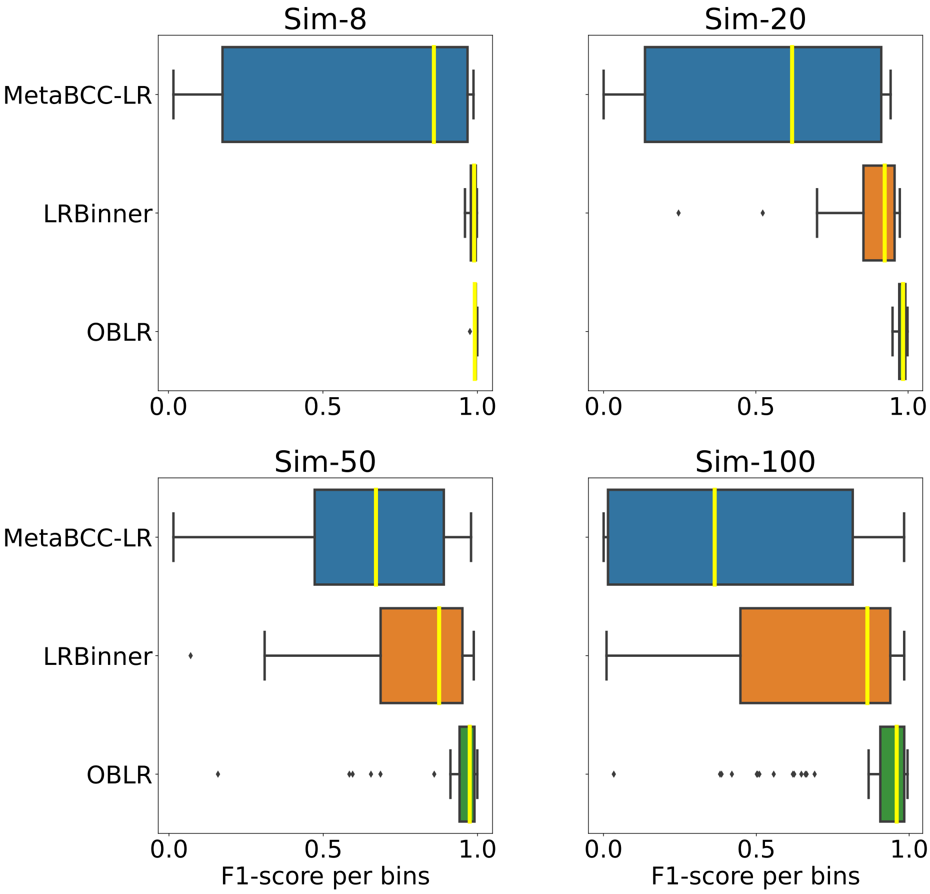


Figure 5.4: Per bin F1-score comparison between MetaBCC-LR, LRBinner and OBLR computed by AMBER (Meyer, Hofmann et al., 2018b) for simulated datasets.

5.4 Results and discussion

We evaluate binning results at the read-level using precision, recall, F1 score and the number of bins produced as well as per-bin F1-scores of read bins using AMBER (Meyer, Hofmann et al., 2018b). Moreover, we conducted a quantitative evaluation of assemblies using MetaQuast (Mikheenko et al., 2016) before and after binning, especially on recovering low-abundance species.

5.4.1 Binning Results

We benchmark OBLR against MetaBCC-LR (Wickramarachchi, Mallawaarachchi, Rajan et al., 2020a) and LRBiner (Wickramarachchi and Lin, 2021a) which are two recent long-read binning tools as presented in Table 5.2. We observed that OBLR results in the highest F1-scores across all the datasets with the overall best performance. OBLR also produces more accurate estimates of the number of bins in most datasets. These observations are further supported by the AMBER (Meyer, Hofmann et al., 2018b) evaluation in Figures 5.4 and 5.5 where OBLR produces best per bin F1-scores ($2 \times \frac{purity \times completeness}{purity + completeness}$) among three long-read binners. Please refer to Appendix D for a detailed discussion on AMBER evaluation and the reason for it to penalize the results from tools producing inaccurate number of bins.

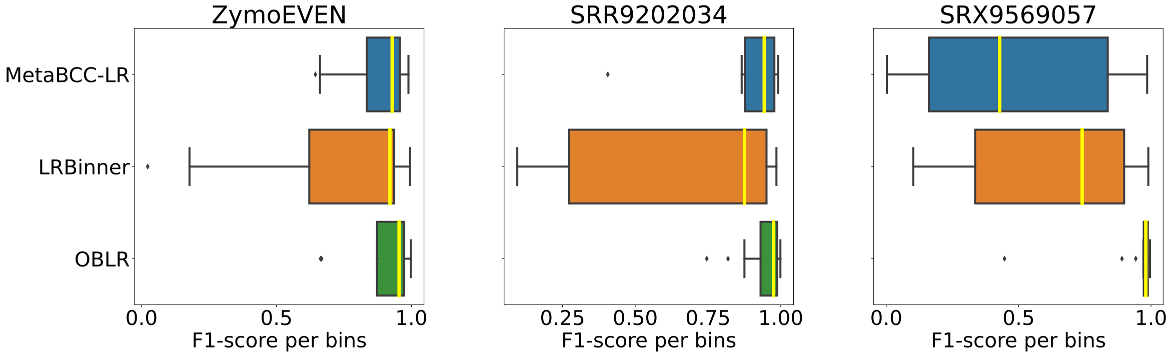


Figure 5.5: Per bin F1-score comparison between MetaBCC-LR, LRBinner and OBLR computed by AMBER (Meyer, Hofmann et al., 2018b) for real datasets.

5.4.2 Assembly Results

We perform assembly using the long-read metagenomics assembler metaFlye (Kolmogorov, Bickhart, Behsaz, Gurevich, Rayko, Shin, Kuhn, Yuan, Polevikov, Smith et al., 2020). Table 5.3 demonstrates the genome fraction and resource usage for assembling raw reads (termed raw) and assembling reads binned by OBLR (termed Binned), respectively. Binning long reads before assembly in general improves (85.11% to 89.53% on **SRX9569057**) or maintains (86.51% 86.67% in **ZymoEVEN** and 90.30% to 90.39% in **SRR9202034**) the genome fraction while significantly saving on the resources. The saving on peak memory varies from 40% to 80%.

5.5 Implementation

In our implementation, we use several approaches to improve performance of OBLR. In step 1, we filter the stdout from kbm2 to simplify the graph while recording total node degrees. We chunk the reads in to blocks of 250, 000 reads, and each block of reads is mapped with entire dataset resulting in several mapping files. Finally, the mapping files are merged into a single file containing edges between reads and degree. For step 2 we use seq2vec (Wickramarachchi, 2021) to compute the tetra-nucleotide frequency vectors for long reads. In steps 3 and 4 we use Rapids.AI (RAPIDS Development Team, 2018) GPU libraries (Nolet et al., 2020) (on NVIDIA RTX 3090 with 24 GB VRAM) for UMAP and HDBSCAN. Finally, we use PyTorch Geometric(Fey and Lenssen, 2019) in step 5 for the GraphSAGE. Performance wise, for **Sim-50** dataset, kbm2 consumes 2 CPU hours while the rest of the OBLR steps consume 0.34 CPU hours. We conducted our experiments on an Ubuntu 20.04.3 LTS system running on AMD Ryzen 9 5950X with 16-core Processor with 32 threads and 128 GB of RAM. Detailed system resource usages are presented in Table 5.4.

Table 5.2: Comparison of binning results of MetaBCC-LR, LRBinner and OBLR.

Dataset	No. of Bins	Evaluation Criteria	MetaBCC-LR	LRBinner	OBLR
Sim-8	8	Precision	90.78%	99.14%	**99.33%**
		Recall	96.18%	99.14%	**99.33%**
		F1 score	93.40 %	99.14%	**99.33%**
		Bins Detected	13	**8**	**8**
Sim-20	20	Precision	82.97%	90.53%	**97.88%**
		Recall	81.95%	88.23%	**97.88%**
		F1 score	82.46%	89.36%	**97.88%**
		Bins Detected	29	18	**20**
Sim-50	50	Precision	82.23%	91.92%	**92.94%**
		Recall	70.56%	77.03%	**97.81%**
		F1 score	75.95%	83.82%	**95.32%**
		Bins Detected	32	31	**45**
Sim-100	100	Precision	**90.50%**	82.60%	87.61%
		Recall	84.54%	92.78%	**95.00%**
		F1 score	88.54%	87.39%	**91.16%**
		Bins Detected	**89**	63	74
ZymoEVEN	10	Precision	**93.09%**	72.41%	75.44%
		Recall	73.84%	92.97%	**95.33%**
		F1 score	82.36%	81.41%	**84.23%**
		Bins Detected	8	**9**	8
SRR9202034	15†	Precision	91.30%	93.16%	**98.48%**
		Recall	69.59%	91.94%	**98.52%**
		F1 score	78.98%	92.55%	**98.50%**
		Bins Detected	11	10	**15**
SRX9569057	13†	Precision	80.94%	80.47%	**95.03%**
		Recall	85.82%	90.68%	**97.70%**
		F1 score	83.31%	85.27%	**96.35%**
		Bins Detected	23	16	**14**

† Species with at least 0.1% abundance.

Table 5.3: Comparison of genome fraction, memory usage and CPU time consumed for assemblies conducted using metaFlye (Kolmogorov, Bickhart, Behsaz, Gurevich, Rayko, Shin, Kuhn, Yuan, Polevikov, Smith et al., 2020) before and after binning.

Dataset	Genome Fraction Raw	Genone Fraction OBLR	Peak Memory (GB)		CPU Time (Hours)	
			Raw	Binned	Raw	Binned
Sim-8	99.90%	99.90%	44.12	9.14	7.98	7.76
Sim-20	99.80%	99.85%	71.70	8.52	14.75	12.84
Sim-50	99.25%	99.32%	58.12	14.36	22.95	20.09
Sim-100	97.70%	97.77%	51.95	25.91	76.62	43.78
ZymoEVEN	86.51%	86.67%	31.67	14.82	15.17	13.22
SRR9202034†	90.30%	90.39%	52.80	28.48	173.20	140.60
SRX9569057†	94.13%	94.27%	49.43	25.84	112.32	98.82

† Genome fraction computed from species with at least 0.1% abundance..

5.6 Discussion

5.6.1 Summary

We presented a novel long-read binner, OBLR, which utilizes the read-overlap graph to bin long reads in metagenomic samples. Recent advances such as the k-mer bins mapping (kbm2) (Ruan and Li, 2020b) enable extremely fast detection of overlapping reads and construction of the read-overlap graph before assembly. OBLR thus makes use of the read-overlap graph to improve the state-of-the-art long-read binning approaches. The read-overlap graph not only helps to estimate the coverage of a single long read, but also allow us to sample the long-reads more uniformly across species of varying abundances. The connectivity information in the read-overlap graph further incorporates the overlapping information between reads into the binning process as overlapped reads are more likely to be in the same species. As a result, OBLR demonstrated promising results in producing more accurate bins for long-read datasets and has the potential to improve on metagenomic assemblies in terms of computing resources and genome fraction, especially for low-abundance species. In the future, we plan to investigate how OBLR can be adapted to take the advantage of the high-accuracy long reads including PacBio HiFi (Wenger et al., 2019) and Nanopore Q20+ (*New nanopore sequencing chemistry in developers' hands; set to deliver Q20 (99%) "raw read" accuracy* 2021) and how to incorporate the binning process into long-read metagenomic assemblies (Feng et al., 2021; Kolmogorov, Bickhart, Behsaz, Gurevich, Rayko, Shin, Kuhn, Yuan, Polevikov, Smith et al., 2020).

5.6.2 Room for improvements

OBLR performed binning of long reads, assigning each read to a single bin. However, in metagenomic datasets, there can be regions shared by multiple species. Although such regions can be relatively smaller for microbial genomes, absence of such regions from bins can result in fragmentation in assembly. An area of improvement would be to assign reads to top N bins having probability beyond the threshold t. Implementation of such a multi-label classifier requires modifications to the step where the reads are classified to identified bins using the GraphSAGE framework. Moreover, activation function should be adjusted supporting multi label classification capabilities which is not currently supported by *log softmax* function. Additionally, post assembly refinements can also be considered. This is because, the assembled bins can either be merged or further separated using information such as marker genes. This could further improve and complete the binning pipeline by producing more accurate bins with high quality genomes. More solid future improvements are discussed in Chapter 8.

Table 5.4: Resource usage of each step in the OBLR pipeline.

Dataset	OBLR step	Peak Memory (GB)	CPU Time (H)	Peak GPU Memory (GB)
Sim-8	Building read-overlap graph	7.15	0.93	-
	Detecting optimum clustering	5.39	0.06	18.70
	Binning all reads	6.38	0.01	
Sim-20	Building read-overlap graph	7.26	1.27	-
	Detecting optimum clustering	5.94	0.05	11.56
	Binning all reads	7.15	0.06	
Sim-50	Building read-overlap graph	7.36	2.68	-
	Detecting optimum clustering	7.08	0.06	18.75
	Binning all reads	8.80	0.16	
Sim-100	Building read-overlap graph	19.23	26.62	-
	Detecting optimum clustering	14.43	0.13	18.73
	Binning all reads	19.03	0.65	
ZymoEVEN	Building read-overlap graph	4.78	4.8	-
	Detecting optimum clustering	22.4	0.22	11.57
	Binning all reads	28.98	0.11	
SRR9202034	Building read-overlap graph	36.54	118.73	-
	Detecting optimum clustering	15.09	0.15	18.89
	Binning all reads	18.48	0.08	
SRX9569057	Building read-overlap graph	4.97	82.81	-
	Detecting optimum clustering	30.20	0.29	18.96
	Binning all reads	38.75	0.15	

Chapter 6

Plasmid Recovery from Long Reads Datasets

The work presented in this chapter was published as

A. Wickramarachchi, V. Mallawaarachchi, L. Pu et al. (2021b). 'PlasLR Enables Adaptation of Plasmid Prediction for Error-Prone Long Reads'. *bioRxiv*

This piece of work was presented at the 14th Great Lakes Bioinformatics (GLBIO) conference 2021. `http://iscb.org/cms_addon/conferences/glbio2021/tracks/General`

6.1 Overview and Motivation

So far we have been developing methods and algorithms to support metagenomics binning of long reads. However, plasmids, a variant of microbial genetic materials is an interesting element that has many biological implications. Recovery of plasmids is the separation of microbial genetic material into the two classes *Plasmid* and *Chromosome*. In contrast with metagenomics binning, plasmids recovery is usually studied as a binary classification problem, mainly targeting assembled contigs.

In this chapter, we study the potential of existing plasmids classification tools to be used for the recovery of plasmid contigs from long reads. We develop the tool PlasLR, a tool that adapts existing contig-classification tools for plasmids to directly classify long and error-prone long reads. PlasLR makes use of both the composition information and k-mer coverage information of long reads. PlasLR leverages the k-mer coverage histograms introduced in MetaBCC-LR (Wickramarachchi, Mallawaarachchi, Rajan et al., 2020a) of Chapter 3 as the coverage metric and tetranucleotide frequency vectors as the composition component.

We evaluate the utility of PlasLR via experimentation which shows that PlasLR achieves high accuracy of plasmid detection on top of state-of-the-art plasmid classification tools. Furthermore, we show that using PlasLR before long-read assembly facilitates enhancements to the assembly quality while recovering more complete plasmids and chromosomes.

6.2 Methods

The contig-classification tools for plasmids have been successfully applied to classify contigs assembled from NGS reads. However, they are not robust on direct classification of all long reads (by treating long reads as contigs) because (1) long reads are more error-prone than contigs assembled from short reads leading to a higher deviation in oligonucleotide frequency vectors and (2) study of plasmid specific genes cannot be conducted on long reads due to the large number of reads available and possible errors on the reads. However, these tools are capable of producing a subset of confident classification at higher confident thresholds, *i.e.* chromosomes closer to probability 0 and plasmids closer to probability 1 in the binary classification setting. Hence, there exists a significant compromise between precision and recall depending on the chosen threshold. This limitation provides PlasLR with the opportunity to employ high confident classifications as initial results. Introduction of dataset specific features alongside facilitates PlasLR to apply semi-supervised machine learning techniques to improve classifications.

PlasLR takes long reads (either PacBio or ONT) as the input. The minimum read length accepted by PlasLR is 1000bp. If we directly apply an existing contig-classification tool to classify all the long reads, unlike contigs classification, the reads get classified into incorrect class (plasmid or chromosome). The intuition is to select and retain the most confident labels (plasmid or chromosome) from a subset of long reads and propagate these labels to other long reads originating from the same chromosome or plasmid.

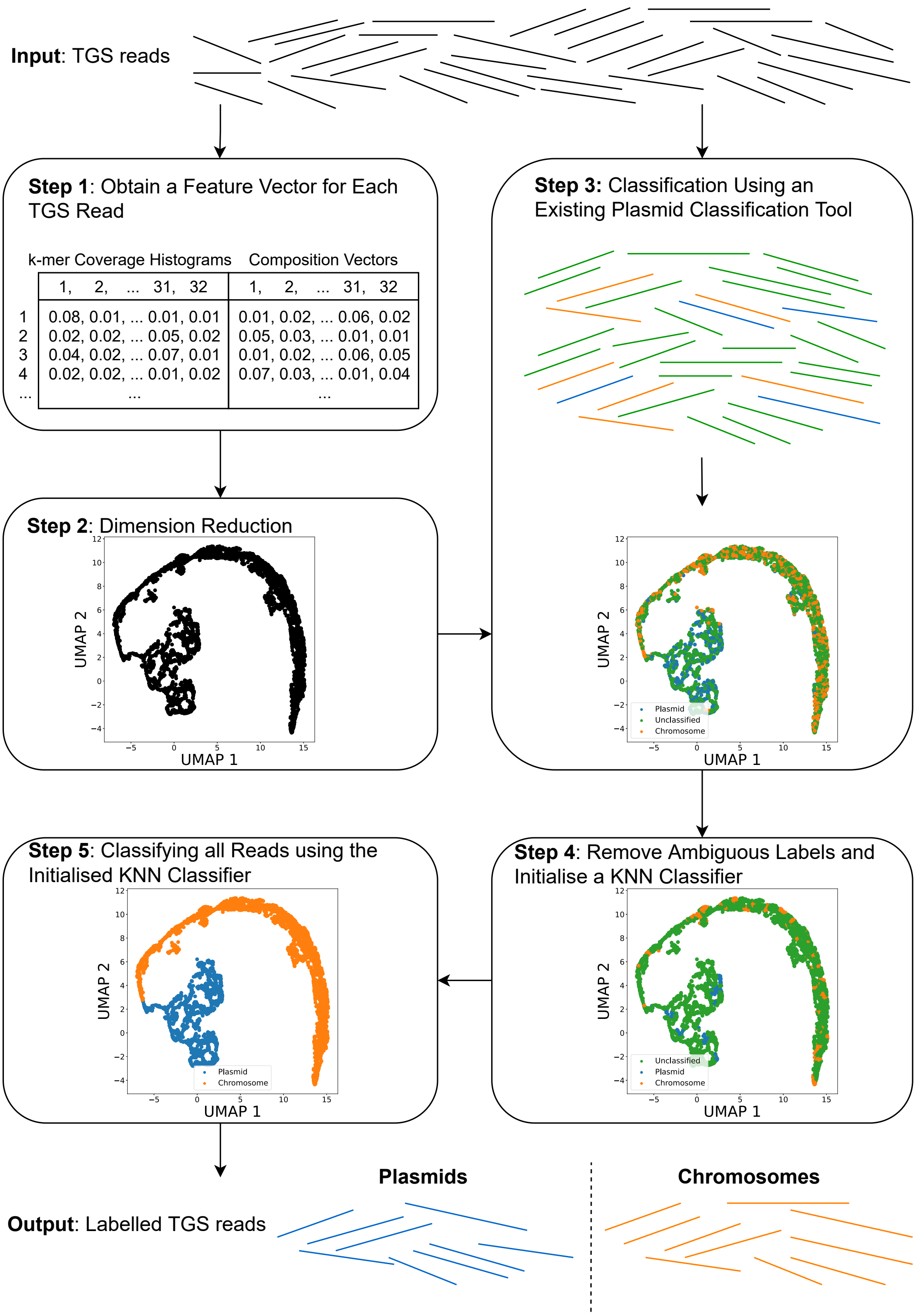
Previous studies have shown that plasmids and chromosomes have different oligonucleotide compositions which have been used in plasmid classification (Davis and Olsen, 2009; Wong et al., 2002; Zhou, Olman et al., 2008; Zhou and Xu, 2010b). The oligonucleotide composition is thus a potential feature for long reads because long reads from the same chromosome or plasmid tend to have similar oligonucleotide composition. Plasmids may also have different coverages compared to chromosomes and such coverage variations have been used in plasmid reconstruction (Antipov, Hartwick et al., 2016). Although it is challenging to estimate the coverage of a chromosome or plasmid that a long read belongs to, the *k*-mer coverage histogram of a long read has been shown to approximate such coverage and has been used in clustering long reads (Wickramarachchi, Mallawaarachchi, Rajan et al., 2020a). Therefore, PlasLR extracts both oligonucleotide composition and *k*-mer coverage histogram as features of a long read and propagates reliable labels from a subset of long reads to others based on these features.

The complete workflow is presented in Figure 6.1. In Step 1, the *k*-mer coverage histograms and the trinucleotide frequency vectors are computed respectively and concatenated together for each read to form a single feature vector. In Step 2 the concatenated vectors are subjected to dimension reduction using UAMP (the default choice) (McInnes, Healy and Melville, 2020). In Step 3, reads are classified using an existing tool and only the high-confident labels (plasmid or chromosome) are retained. In Step 4, nearest neighbors in the UMAP projections are used to detect and remove ambiguous labels, and a K-Nearest Neighbor (KNN) classifier (Mitchell, 1997) is initialized on the remaining labelled reads. Finally, in Step 5, all the unlabelled reads are labelled using the KNN classifier. Details of each step are explained in the following sections.

6.2.1 Obtain a Feature Vector for Each Read

In real-world genomic samples, the species (chromosomes and plasmids) are of varying abundances but contigs from the same chromosome or plasmid are likely to have similar coverages. In other words, the reads should overlap and stack on each other to reflect the actual coverages. Intuitively, this can be achieved using an all-vs-all alignment approach. However, such a naïve approach would be computationally expensive as the number of reads can be very large. Therefore, the *k*-mer based approach is used MetaBCC-LR (Wickramarachchi, Mallawaarachchi, Rajan et al., 2020a) is reused for coverage estimation in PlasLR.

Microbial genomes carry signatures of oligonucleotide frequencies that are unique to each species (Abe et al., 2003). These signatures are preserved within a species across its genome and varies between different species (Wu, Simmons et al., 2015). Moreover, previous studies have shown that chromosomes and plasmids have different oligonucleotide compositions (Davis and Olsen, 2009; Wong et al., 2002; Zhou, Olman et al., 2008). Given the longer lengths of reads, despite their error prone nature, the reads carry closely similar oligonucleotide frequencies to that of their underlying chromosomes/plasmids. While PlasFlow and PlasClass use the composition information from 3-mers to 7-mers in contigs, PlasLR only uses 3-mers (trinucleotides or trimers) to compute frequency vectors for the long reads because higher error rates in long reads affect more *k*-mers as *k* increases. 3-mer frequency vectors have been successfully utilized in metagenomics binning of long reads (Wickramarachchi and Lin, 2021a; Wickramarachchi, Mallawaarachchi, Rajan et al., 2020a). Now, we concatenate the coverage histograms and trinucleotide frequency vectors to derive a vector with 64 dimensions for each long read as shown in Figure 6.1 Step 1. Note that, in contrast with LRBinner and MetaBCC-LR, PlasLR uses concatenated features as they are only required for approximate neighborhood analysis.


Input: TGS reads
Step 1: Obtain a Feature Vector for Each TGS Read
Step 2: Dimension Reduction
UMAP 2
UMAP 1
Step 3: Classification Using an Existing Plasmid Classification Tool
Plasmid
Unclassified
Chromosome
Step 4: Remove Ambiguous Labels and Initialise a KNN Classifier
Step 5: Classifying all Reads using the Initialised KNN Classifier
Output: Labelled TGS reads
Plasmids
Chromosomes

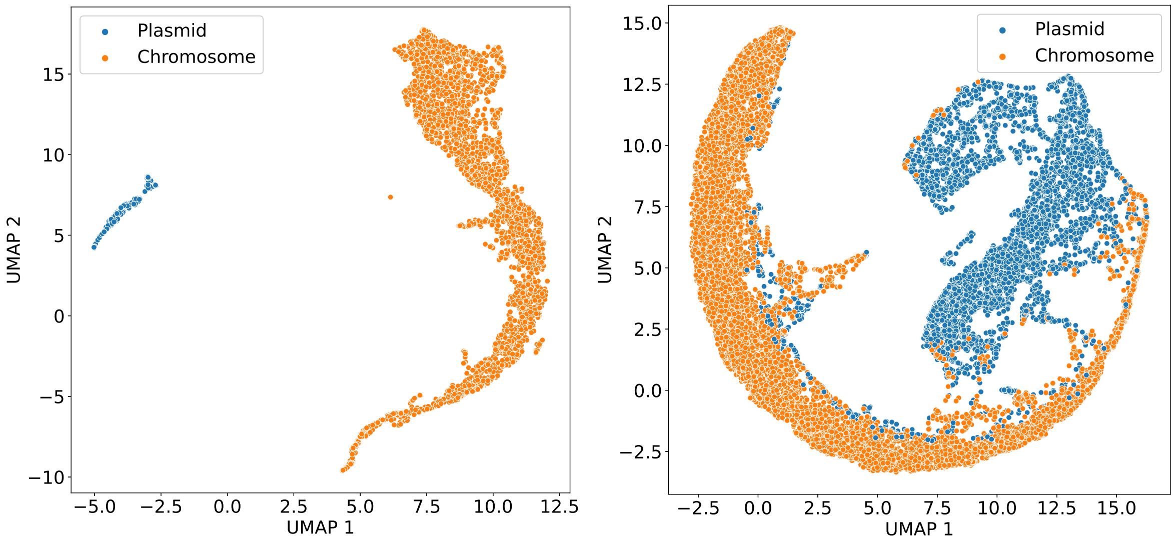
	k-mer Coverage Histograms	Composition Vectors
	1, 2, ... 31, 32	1, 2, ... 31, 32
1	0.08, 0.01, ... 0.01, 0.01	0.01, 0.02, ... 0.06, 0.02
2	0.02, 0.02, ... 0.05, 0.02	0.05, 0.03, ... 0.01, 0.01
3	0.04, 0.02, ... 0.07, 0.01	0.01, 0.02, ... 0.06, 0.05
4	0.02, 0.02, ... 0.01, 0.02	0.07, 0.03, ... 0.01, 0.04
...	...	...

Figure 6.1: Long reads are provided as inputs for PlasLR. In Step 1, the *k*-mer coverage histograms and the trinucleotide frequency vectors are computed respectively and concatenated together for each read. In Step 2 the concatenated vectors are subjected to dimension reduction using UMAP. In Step 3, reads are classified using an existing tool and only the high-confident labels (plasmid or chromosome) are retained. In Step 4, nearest neighbors in the UMAP plot are used to detect and remove some ambiguous labels, and a KNN classifier is initialized on the remaining labelled reads. Finally, in Step 5, all the unlabelled reads are labelled using the KNN classifier.

6.2.2 Dimension Reduction

UMAP (Uniform Manifold Approximation and Projection) (McInnes, Healy and Melville, 2020) is used to project the concatenated vectors of Step 1 into the two-dimensional space. UMAP is a technique that is used to reduce the dimensionality of the data while preserving most of the variation among data points in the dataset. UMAP tries to embed data in lower dimensions such that the fuzzy topological structure is conserved (McInnes, Healy and Melville, 2020). Hence, PlasLR uses two UMAP components to visualize (refer to Step 2 of Figure 6.1) and classify long reads into the two classes.

Figure 6.2 denotes the UMAP plots of 64-dimension vectors of simulated PacBio reads originating from the *Aquifex aeolicus* chromosome (NCBI accession number NC_000918, 20x coverage) and the *Lactococcus lactis* plasmid (NCBI accession number NC_000906, 200x coverage) and **Sim-2C5P** dataset. We include the ground-truth information as colored points in Figure 6.2 to demonstrate a locality of the chromosomal and plasmid sequences in the UMAP plot. Note that PlasLR provides users with the ability to select dimensionality reduction algorithm (PCA, UMAP, OpenTSNE (Poličar et al., 2019)).



(a)

(b)

Figure 6.2: The UMAP decomposition plot of concatenated coverage histograms and 3-mer frequency vectors of a set of simulated PacBio reads originating from one *Aquifex aeolicus* chromosome (20X coverage) and one *Lactococcus lactis* plasmid (200X coverage) is shown on left. In the right, a **Sim-2C5P** dataset which is more complicated is demonstrated.

6.2.3 Classification Using an Existing Plasmid Classification Tool

As discussed above, PlasLR makes use of plasmid classification results from existing contig classification tools. We have selected two of the latest plasmid detection tools, PlasFlow (Krawczyk et al., 2018) and PlasClass (Pellow, Mizrahi et al., 2020) to obtain the initial classifications. PlasFlow and PlasClass are shown to outperform earlier tools in recent benchmarks (Krawczyk et al., 2018; Pellow, Mizrahi et al., 2020) and provide confidence levels for the output labels predicted. This facilitates PlasLR to perform downstream tasks by filtering classifications based on these confidence levels. Note that, for a fair comparison, we have not considered reference based tools such as Platon (Schwengers et al., 2020) or Kraken2 (Wood, Lu et al., 2019).

Note that the initial classification results of either PlasFlow or PlasClass consist of the three classes; (1) plasmid, (2) chromosome and (3) unclassified. Most (>50%) reads will be in the unclassified class as only high-confident assignments of plasmids and chromosomes are selected. PlasFlow and PlasClass output formats are as follows.

PlasFlow Output

The output of PlasFlow (Krawczyk et al., 2018) consists of a tabular file containing all the predictions of contigs consisting of several columns including *contig_id*, *contig_name*, *contig_length*, *id* and *label*. Additional columns showing probabilities of assignment to 26 genus classes were also included.

PlasClass Output

In the output of PlasClass (Pellow, Mizrahi et al., 2020), each line represents the probability of the sequence being a plasmid.

The Step 3 of Figure 6.1 demonstrates the labels of plasmids and chromosomes incorporated into the two dimensional UMAP plot, and will be used in subsequent refinement and propagation.

6.2.4 Remove Ambiguous Labels and Initialize a KNN Classifier

Using the set of labelled reads from Step 3, nearest neighbors (50 neighbors by default) are computed for each read using the lower dimensional representation. The label of a read (plasmid or chromosome) is defined to be *ambiguous* if more than 20% of its labelled nearest neighbors have a different label. PlasLR removes those ambiguous labels and set them to be unclassified. Then, a K-Nearest Neighbor (KNN) classifier (Mitchell, 1997) is initialized on the remaining labelled points (refer to Step 4 of Figure 6.1). In this step, the KNN indexes the labelled points thus enabling predictions on unseen points using the K (K=50) closest labelled data points.

6.2.5 Classifying all Reads using the Initialized KNN Classifier

The initialized KNN classifier is used to predict the probability that a read is a plasmid. We label the reads exceeding 0.5 probability threshold as plasmids and vice versa. Step 5 of Figure 6.1 demonstrates the visualization of assigning unclassified reads to the two classes using the KNN classifier. PlasLR finally classifies all the input reads into two classes (plasmid and chromosome)

6.3 Experimental Setup

6.3.1 Datasets

We used the following datasets in our experiments.

1. We simulated several datasets with genomic abundances obtained from a log normal distribution following the model in (Pellow, Mizrahi et al., 2020). The plasmid copy numbers were taken from a geometric distribution where the probability of success is $min(1, log(L)/10)$ and L is the length of the plasmid. This was performed to amplify the abundance of short plasmids (Pellow, Mizrahi et al., 2020). The following three datasets are simulated using SimLoRD (Stöcker et al., 2016). The error rate was set to 10% and the minimum read length was set to 1000bp. The detailed information about species (chromosomes and plasmids) is available in Appendix Table B.5.
    - **Sim-2C5P**: Contains 1 species with a total of 2 chromosomes and 5 plasmids.
    - **Sim-4C11P**: Contains 2 species with a total of 4 chromosomes and 11 plasmids.
    - **Sim-10C16P**: Contains 5 species with a total of 10 chromosomes and 16 plasmids.
2. Zymo GridION dataset (*Zymo-GridION-EVEN-BB-SN*) from the ZymoBIOMICS Microbial Community Standards (Nicholls et al., 2019) was used, which consists of real Oxford Nanopore (ONT) reads. In order to obtain datasets with a considerable portion of plasmid content, we subsample reads from each of the three species *Escherichia coli*, *Staphylococcus aureus* and *Salmonella enterica*. We refer to the three datasets as **Zymo-EC**, **Zymo-SA** and **Zymo-SE** respectively. We randomly subsampled up to 10,000 reads from each plasmid and chromosome class.

Table 6.1: Information about the datasets in the experiments.

Dataset	Read type	Total number of reads	Average read length (bp)	Number of plasmid reads	Number of chromosomal reads
Sim-2C5P	PacBio	20250	8246	6720	13530
Sim-4C11P	PacBio	119957	8200	35110	84847
Sim-10C16P	PacBio	199983	8200	32223	168760
Zymo-EC	ONT	207442	6836	6437	10000
Zymo-SA	ONT	375654	4058	10000	10000
Zymo-SE	ONT	199974	6785	2850	10000

6.3.2 Tools Used for Initial Classification

We choose PlasFlow and PlasClass as initial classifiers for plasmids and chromosomes since they have outperformed other tools in recent evaluations (Krawczyk et al., 2018; Pellow, Mizrahi et al., 2020). PlasFlow outputs probabilities for 26 different classes under the phylum of each chromosome and plasmid. Furthermore, the results contain a single field labelling the classification in the form $Class.phylum$.

PlasClass outputs single-valued probabilities to indicate how probable a sequence belongs to a plasmid. PlasLR chooses the thresholds such that 5% of reads are plasmids if such an amount of reads exists above 70% confidence. Similarly up to 20% reads are chosen to be plasmids below 50% confidence.

6.3.3 Evaluation Criteria

Different classification tools (PlasFlow, PlasClass and PlasLR) are evaluated based on the precision, recall and F1 score of plasmid and chromosome classifications separately. For this purpose, we define the following terms.

- TP_p: the number of actual plasmid sequences that were classified as plasmid (true positives for plasmids)
- TP_c: the number of actual chromosomal sequences that were classified as chromosomal (true positives for chromosomes)
- FP_p: the number of non-plasmid sequences that were classified as plasmid (false positives for plasmids)
- FP_c: the number of non-chromosomal sequences that were classified as chromosomal (false positives for chromosomes)
- FN_p: the number of plasmid sequences that were not classified as plasmid (false negatives for plasmids)
- FN_c: the number of chromosomal sequences that were not classified as chromosomal (false negatives for chromosomes)

The micro-averaged precision, recall and F1 score are calculated as follows. Note that FN_p and FN_c include the number of unclassified chromosomal and plasmid sequences, respectively. We use the following equations to evaluate the classification of chromosomes.

For the evaluation of plasmids the following equations were used;

$$Plasmid\ Precision\ (\%) = \frac{TP_p}{TP_p + FP_p} \tag{6.1}$$

$$Plasmid\ Recall\ (\%) = \frac{TP_p}{TP_p + FN_p} \tag{6.2}$$

For both the classes we compute the F1-score using the following equation;

$$F1\ score\ (\%) = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{6.3}$$

For the reads in the real dataset (with unknown ground truth), the reads are mapped to the known reference genomes using Minimap 2 (Li, 2018a) and the chromosome or plasmid that results in the best alignment is considered as the origin of the read.

6.4 Results and Discussion

6.4.1 Classification Results on Long Reads

We executed PlasFlow (Krawczyk et al., 2018) and PlasClass (Pellow, Mizrahi et al., 2020) and plotted precision-vs-recall curves to examine the performance of each tool as the decision boundary changes. The resulting trade-offs between precision and recall are demonstrated in Figure 6.3 for the Zymo-SA dataset (details about this dataset can be found in Section 3.1). As expected, the higher the recall PlasFlow and PlasClass achieve (*i.e.*, classify more reads), the lower the precision their classification results have. Hence, it is evident that mere parameter setting of a decision threshold is insufficient to improve the overall classification of results (Note that both PlasFlow and PlasClass provide a probability to assign each input sequence to either plasmids or chromosomes). When we select higher thresholds on such probabilities, both PlasFlow and PlasClass are able to generate relatively accurate classification results for a small subset of input sequences. In Figure 6.3, we demonstrate the PlasLR precision-vs-recall curves which show a significant shift of precision and recall from the initial results.

We present the results of PlasLR on classification of long reads into two classes (plasmid and chromosome). Table 6.2 and 6.3 denotes the comparison of results of PlasFlow (Krawczyk et al., 2018), PlasClass (Pellow, Mizrahi et al., 2020) and PlasLR (on top of PlasFlow and PlasClass) respectively. Note that the probabilities given by PlasClass, PlasFlow and PlasLR are converted into respective classes by considering the decision boundary of 0.5.

According to the F1-scores highlighted in Table 6.2 and 6.3, we can see that PlasLR has improved the classification results based on the initial results obtained from PlasFlow and PlasClass. Similar to Figure 6.3, PlasLR has shown significant improvements over the recall while maintaining a high precision (similar to or even better than the precision of the high-confident subset of reads initially classified by PlasFlow or PlasClass). The improvement on precision in PlasLR is gained by removing ambiguous labels while the improvement on recall of PlasLR is achieved by applying the initialized KNN classifier to unclassified reads on the UMAP plot. Although the precision drops in a few cases, PlasLR improves on the recall, F1 score and percentage of plasmid reads recovered on the datasets. Note that in dataset **Sim-10C5P** PlasLR has compromised a bit of plasmid F1-score in improving that of chromosomes. The initial results of dataset are relatively poor due to the mere complexity of the composition. However, the improvements on the Zymo datasets are significant due to the relative simplicity of the datasets.

6.4.2 Improved Assemblies on Long Reads

After classification of reads into the two classes, we assembled the classified chromosomal and plasmid reads separately using metaFlye (Kolmogorov, Rayko et al., 2019) (available in Flye v2.4.2). metaFlye

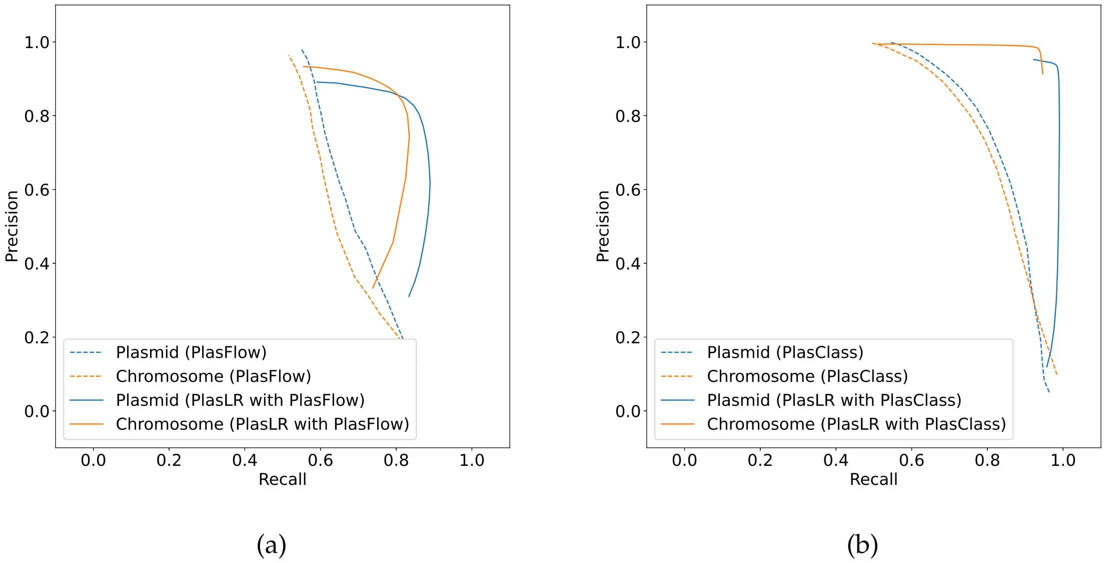


Figure 6.3: The trade-off between precision and recall for (a) PlasFlow results and for (b) PlasClass results on the Zymo-SA dataset. The precision-recall curve is generated by varying the probability threshold in PlasFlow and PlasClass, respectively. The classification results of PlasLR on top of PlasFlow and PlasClass achieve high precision and recall. Precision and recall are computed separately for plasmid and chromosomal sequences using $TP/(TP + FP)$ and $TP/(TP + FN)$ respectively. Unclassified reads are considered under FN for recall.

is a popular metagenomic long-read assembler which has shown best performance in assembling plasmids in a recent benchmark (Wick and Holt, 2019). Moreover, we also assemble the complete set of reads (without classification) using metaFlye as the baseline for comparison.

Table 6.4 demonstrates the assembly results for metaFlye, for all possible assembly approaches along with PlasLR results on top of both PlasFlow and PlasClass. The results show that PlasLR classification can improve the genome fraction (computed by MetaQUAST (Mikheenko et al., 2015b)) and the number of plasmids recovered by metaFlye. The chromosomal assemblies are significantly improved over the non-PlasLR approaches. Note that assembly performance gain is significant wherever there is a higher number of plasmids. Furthermore, poor chromosomal assembly in non-PlasLR approaches indicates the classification of chromosomal reads into plasmid class. This may result in plasmid assemblies whose contig set contain partial chromosomal assemblies. PlasLR mitigate this situation by pushing chromosomal and plasmid reads into appropriate bins facilitating proper assembly. This is evident in dataset **Sim-10C5P** where PlasLR with PlasFlow demonstrates the most number of sequences assembled (5 chromosomes and 13 plasmids). Note that, although genome fractions are comparable in some PlasLR results (especially in **Zymo-EC**, **Zymo-SA** and **Zymo-SE** datasets), each assembled class may contain multiple contigs that corresponds to the opposite class which might result in misleading downstream analysis. PlasLR supports the mitigation of such false positive plasmid/chromosome assemblies by accurate classification of long reads as indicated in Table 6.3 (refer to **Zymo-EC**, **Zymo-SA** and **Zymo-SE** datasets).

These improvements were achieved by classifying reads before assembly, where metaFlye can assemble plasmids and chromosomes independently. The independent assembly of chromosomes and plasmids allows metaFlye to estimate more appropriate assembly parameters for plasmids and chromosomes, respectively.

6.5 Implementation

PlasLR uses the same optimization techniques introduced in MetaBCC-LR for the computation of coverager and composition features. All steps of PlasLR are performed using multi threading (8

Table 6.2: Comparison of classification performance of PlasLR and PlasFlow

Dataset	Evaluation Criteria	PlasFlow		PlasLR with PlasFlow result	
		Chromosome	Plasmid	Chromosome	Plasmid
Sim-2C5P	Precision	87.75%	73.50%	94.25%	75.28%
	Recall	26.84%	47.31%	85.40%	89.51%
	F1-Score	41.10%	57.56%	**89.61%**	**81.78%**
Sim-4C11P	Precision	93.94%	65.81%	94.62%	63.16%
	Recall	27.27%	54.71%	78.45%	89.22%
	F1-Score	42.28%	59.75%	**85.78%**	**73.97%**
Sim-10C16P	Precision	96.64%	24.45%	94.42%	33.73%
	Recall	23.28%	30.22%	72.01%	77.01%
	F1-Score	37.52%	27.03%	**81.71%**	**46.92%**
Zymo-EC	Precision	95.15%	80.79%	93.68%	81.22%
	Recall	32.95%	33.65%	86.46%	90.94%
	F1-Score	48.95%	47.51%	**89.93%**	**85.80%**
Zymo-SA	Precision	72.59%	79.82%	73.04%	88.50%
	Recall	31.23%	24.29%	90.25%	69.25%
	F1-Score	43.67%	37.25%	**80.74%**	**77.70%**
Zymo-SE	Precision	98.67%	70.80%	98.03%	69.71%
	Recall	26.79%	42.46%	88.39%	93.75%
	F1-Score	42.14%	53.08%	**92.96%**	**79.96%**

threads by default). Our PlasLR implementation uses the python scikit-learn library implementation of the KNN classifier (known as *KNeighborsClassifier*) and PCA (Pedregosa et al., 2011). Furthermore, UMAP-learn (McInnes, Healy and Melville, 2020) (used as the default dimensionality reducer) and OpenTSNE (Poličar et al., 2019) were obtained from the repositories of respective authors of work. Users are provided with the parameter `--dimension-reduction` which accepts values *pca*, *tsne* or *umap* should they wish to use a different dimensionality reduction technique.

6.6 Discussion

6.6.1 Summary

In this chapter, we designed and evaluated PlasLR, a tool that adapts contig-classification tools for plasmids to long reads. While existing contig-classification tools are able to accurately classify a subset of long reads (by treating them as contigs), the use of PlasLR refines existing labels and extends them to all reads. Moreover, we showed that the classification of long reads (into chromosomal and plasmid classes) before assembly improves the recovery of plasmids from both simulated and real metagenomic datasets.

The classification improvements are prevailing due to the combination of coverage and composition information, along with a set of initial classifications in a semi-supervised manner. Coverage

Table 6.3: Comparison of classification performance of PlasLR and PlasClass

Dataset	Evaluation Criteria	PlasClass		PlasLR with PlasClass result	
		Chromosome	Plasmid	Chromosome	Plasmid
Sim-2C5P	Precision	82.40%	56.76%	90.93%	61.30%
	Recall	24.75%	45.34%	73.25%	85.30%
	F1-Score	38.06%	50.41%	**81.14%**	**71.33%**
Sim-4C11P	Precision	94.66%	50.91%	94.17%	43.51%
	Recall	26.80%	61.58%	50.27%	92.48%
	F1-Score	41.77%	55.74%	**65.55%**	**59.18%**
Sim-10C16P	Precision	96.62%	31.49%	97.25%	24.77%
	Recall	23.66%	53.73%	47.91%	92.68%
	F1-Score	38.02%	**39.71%**	**64.20%**	39.09%
Zymo-EC	Precision	98.29%	89.26%	92.35%	91.30%
	Recall	34.59%	11.50%	94.61%	87.82%
	F1-Score	51.17%	20.37%	**93.47%**	**89.52%**
Zymo-SA	Precision	89.74%	94.96%	76.43%	98.96%
	Recall	39.38%	14.28%	99.18%	71.77%
	F1-Score	54.74%	24.82%	**86.33%**	**83.20%**
Zymo-SE	Precision	99.80%	94.78%	95.71%	91.72%
	Recall	29.21%	22.95%	97.59%	85.90%
	F1-Score	45.19%	36.95%	**96.64%**	**88.72%**

information facilitates the discrimination of plasmids and chromosomes based on the copy number. Composition information predominantly enables the discrimination of sequences based on their origin species. The lower dimensional projection provides a spatial representation where the sequences with similar coverage and composition are closer, enabling the expansion of initial classification labels.

6.6.2 Room for improvements

Inevitably, the final results are influenced by the quality of the initial classifications. Therefore, the selection of high confident initial classification is vital for PlasLR to perform accurately. Similar to contig-based classifiers, PlasLR also faces challenges when classifying long reads from similar regions of plasmids and chromosomes. In such situations, the coverage information estimated by PlasLR may be misleading and thus result in possible misclassifications.

As a matter of fact, PlasLR, like other tools evaluated, performs binary classification (i.e., plasmids v.s. chromosomes). A fine-grained classification is needed to provide distinctions between different plasmids and chromosomes and has the potential to be incorporated in the assembly process. In such scenarios, PlasLR could benefit from reference database-based approaches such as Kraken2 (Wood, Lu et al., 2019), where predictions can be made at species level followed by plasmid detection.

Table 6.4: Comparison of metaFlye assemblies using MetaQUAST (Mikheenko et al., 2015b). The number of plasmids recovered is the number of plasmids which have a genome fraction greater than 85%. Genome fractions are computed separately on the assemblies of plasmid and chromosome classes. Where genome fraction is defined as the total number of aligned bases as a percentage of genome size.

Dataset	Classification	metaFlye assembly			
		Genome fraction		Sequences recovered	
		Chromosome	Plasmid	Chromosome	Plasmid
Sim-2C5P	Raw reads	82.19%	96.21%	0	4
	PlasFlow	88.30%	97.81%	2	5
	PlasClass	65.64%	99.46%	0	5
	PlasLR with PlasFlow result	**95.99%**	**99.63%**	**2**	**5**
	PlasLR with PlasClass result	94.13%	99.40%	**2**	**5**
Sim-4C11P	Raw reads	**99.10%**	88.34%	**4**	8
	PlasFlow	73.83%	85.91%	1	9
	PlasClass	79.26%	85.56%	2	8
	PlasLR with PlasFlow result	93.20%	91.00%	3	**10**
	PlasLR with PlasClass result	96.19%	**92.08%**	**4**	9
Sim-10C16P	Raw reads	47.47%	39.63%	4	3
	PlasFlow	57.30%	**96.00%**	3	**13**
	PlasClass	64.04%	95.70%	3	**13**
	PlasLR with PlasFlow result	68.35%	95.88%	**5**	**13**
	PlasLR with PlasClass result	**86.41%**	75.50%	6	10
Zymo-EC	Raw reads	89.95%	100.00%	**1**	**1**
	PlasFlow	94.57%	100.00%	**1**	**1**
	PlasClass	98.82%	100.00%	**1**	**1**
	PlasLR with PlasFlow result	94.46%	100.00%	**1**	**1**
	PlasLR with PlasClass result	**99.00%**	**100.00%**	**1**	**1**
Zymo-SA	Raw reads	95.66%	66.63%	**1**	2
	PlasFlow	95.49%	99.96%	**1**	**3**
	PlasClass	97.51%	99.99%	**1**	**3**
	PlasLR with PlasFlow result	**98.52%**	**100.00%**	**1**	**3**
	PlasLR with PlasClass result	98.39%	99.93%	**1**	**3**
Zymo-SE	Raw reads	96.60%	**100.00%**	**1**	**1**
	PlasFlow	94.93%	**100.00%**	**1**	**1**
	PlasClass	99.58%	**100.00%**	**1**	**1**
	PlasLR with PlasFlow result	97.23%	**100.00%**	**1**	**1**
	PlasLR with PlasClass result	**99.70%**	**100.00%**	**1**	**1**

Chapter 7

Improving Plasmid Classification using Assembly Graphs

The work presented in this chapter was published as

A. Wickramarachchi and Y. Lin (2021). 'GraphPlas: Refined Classification of Plasmid Sequences using Assembly Graphs'. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*

This piece of work was presented at the 19th Asia Pacific Bioinformatics Conference (APBC) 2021. `http://www.binfo.ncku.edu.tw/APBC2021/accepted_papers.html`

The software is freely available at `https://github.com/anuradhawick/GraphPlas`.

7.1 Overview and Motivation

In this chapter, we propose an assembly graph assisted approach of improving plasmid recovery by harnessing information such as composition, coverage and connectivity in the assembly graph. So far we have been working on improving plasmid classification using the combination of dataset specific features on top of existing plasmid classification tools. Here we develop GraphPlas which extends plasmid classification of contigs by introducing dataset specific information such as composition, coverage and assembly graph to improve the classification of existing methods. The proposed methodology significantly improves the precision and recall of the plasmid classification on top of popular tools in the field. To the best of our knowledge, this is the first time that composition, coverage and graph topology information of assembled contigs have been utilized together to address the problem of plasmid sequence classification.

7.2 Methods

The complete workflow of GraphPlas is demonstrated in Figure 7.1. GraphPlas takes assembled contigs and the relevant assembly graph as the input. Then the contigs are classified using an existing plasmid detection program that predicts the plasmid probability of contigs. Then the contigs are labelled as *chromosomes*, *plasmids* or *unclassified* depending on the probability values. Note that the *unclassified* class contains contigs that are either shorter than 1,000 base pairs or ones with probabilities that are in-between the probability boundaries of plasmids and chromosomes.

For the initial prediction, we chose the approaches presented by PlasClass (Pellow, Mizrahi et al., 2020) and PlasFlow (Krawczyk et al., 2018). This is because PlasClass and PlasFlow outputs probability values on which a confidence level can be applied. Also, the classifications at higher confidence regions are reliable. We varied the probability threshold for PlasClass in order to investigate if the classification can be improved just by parameter tuning. However, from Figure 7.2 and Figure 7.3, it is clearly evident that means beyond parameter tuning are required for better results. Furthermore, it is evident that at high confidence thresholds, the precision of classification is reasonably high for GraphPlas to improve the results.

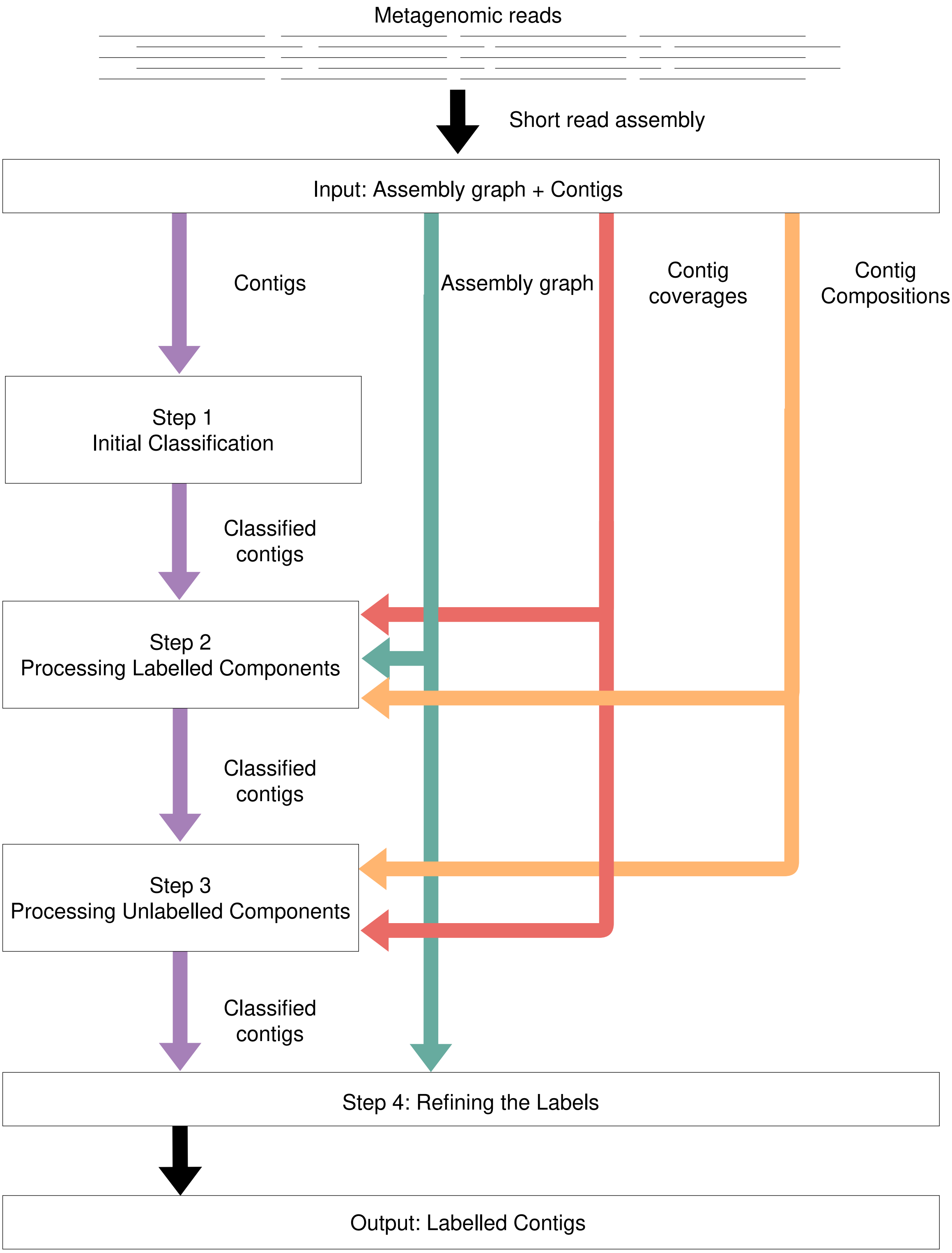
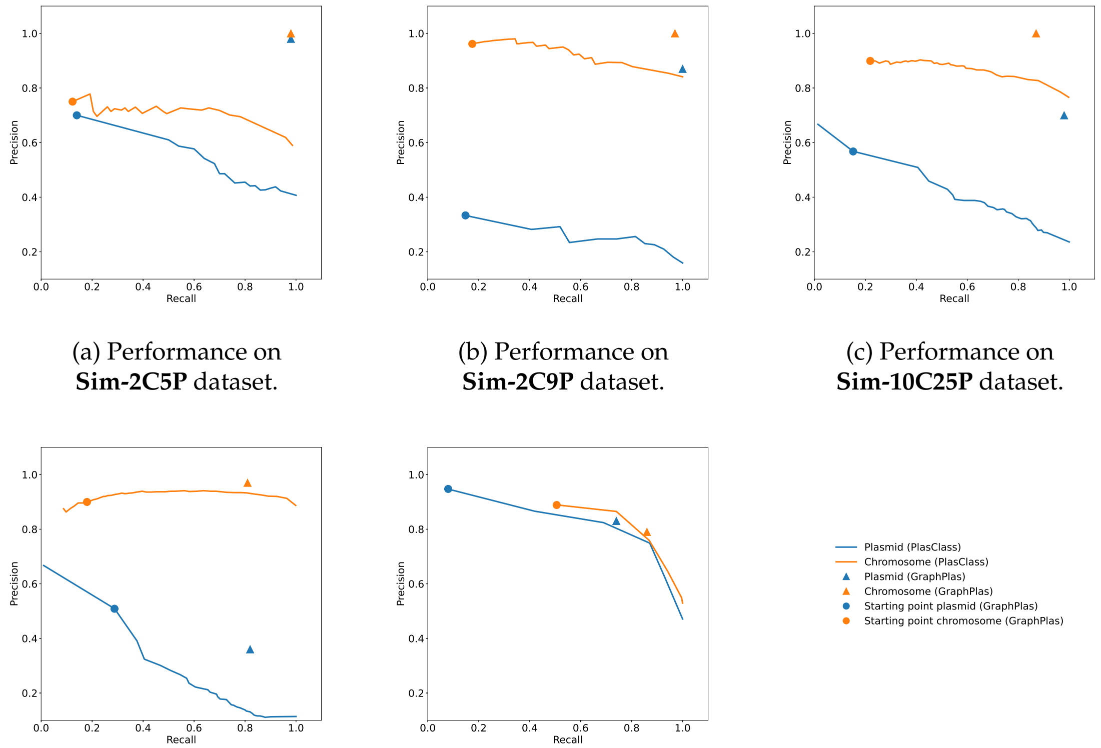


Figure 7.1: The workflow of GraphPlas. The inputs for the workflow are the contigs and the assembly graph output from the assembler. The contigs are initially classified and the most confident set of contigs are used as seeds. Contig labels are propagated to other contigs using graph topology, composition and coverage information. Finally, the contig labels are refined again and IDs are output with the assigned label.

7.2.1 Computation of contig similarity metrics

For the steps 2 and 3 in Figure 7.1, distances between contigs are required to label the assembly graph in a semi-supervised manner. These distances are derived from summing up negative log values of



(a) Performance on **Sim-2C5P** dataset.

(b) Performance on **Sim-2C9P** dataset.

(c) Performance on **Sim-10C25P** dataset.

(d) Performance on **Sim-14C38P** dataset.

(e) Performance on **Wastewater-Plas** dataset.

Legend.

Figure 7.2: Precision-recall curves for PlasClass and GraphPlas. Plots obtained by varying the probability threshold for classification from 0 to 1. For GraphPlas we have a single point for each class as we pick the best starting threshold for each tool.

similarities obtained from three different methods. The computation of each similarity measure is explained in the following subsections.

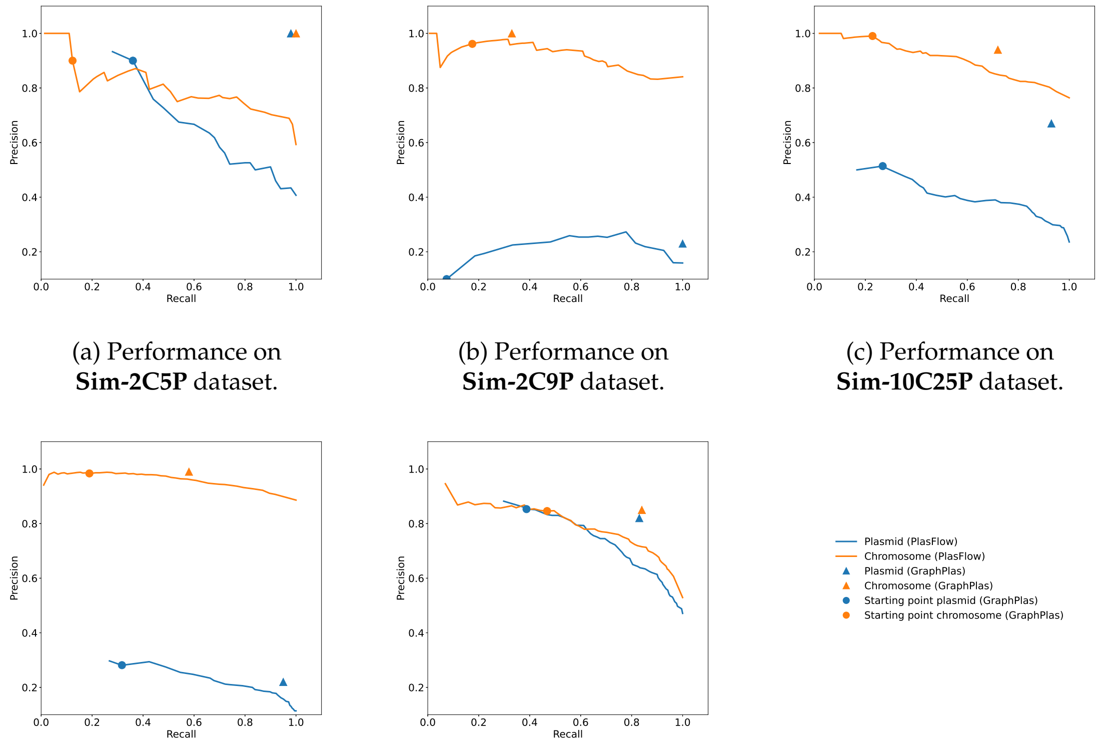
Computing similarity of contigs using the topology of assembly graph

Consider the assembly graph G where V is the set of vertices that represents the contigs in the graph. Let L be the set of labelled vertices and U be the set of unlabelled vertices. Topological similarity $S_t(V_i, V_j)$, where $V_i \in U$ and $V_j \in L$, is computed using the random walk probabilities. In GraphPlas we use a variant of label propagation algorithm proposed in (Zhu and Ghahramani, 2002). The detailed algorithm is as follows.

First we obtain the degree matrix D and adjacency matrix A using the assembly graph. Each contig is considered as a vertex whereas connections in the assembly graph are denoted as edges in the adjacency matrix. A is ordered such that A_{ll}, A_{lu}, A_{ul} and A_{uu} respectively represent the labelled to labelled, labelled to unlabelled, unlabelled to labelled and unlabelled to unlabelled edge connections between vertices as demonstrated in equation 7.1.

$$A = \begin{pmatrix} A_{ll} & A_{lu} \\ A_{ul} & A_{uu} \end{pmatrix} \tag{7.1}$$

The same order is followed in the degree matrix D, where the degrees associated with labelled vertices are arranged on top in the same order of A. We adopt the methodology of (Zhu and Ghahramani, 2002) for the estimation of similarity of contigs using the graph topology. The transition probabilities are computed using equation 7.2.



(a) Performance on **Sim-2C5P** dataset.

(b) Performance on **Sim-2C9P** dataset.

(c) Performance on **Sim-10C25P** dataset.

(d) Performance on **Sim-14C38P** dataset.

(g) Performance on **Wastewater-Plas** dataset.

Legend.

Figure 7.3: Precision-recall curves for PlasFlow and GraphPlas. Plots obtained by varying the probability threshold for classification from 0 to 1. For GraphPlas we have a single point for each class as we pick the best starting threshold for each tool.

$$P_t = D^{-1} \times A \tag{7.2}$$

$$P_l = (P_t)^n \times Y_l \tag{7.3}$$

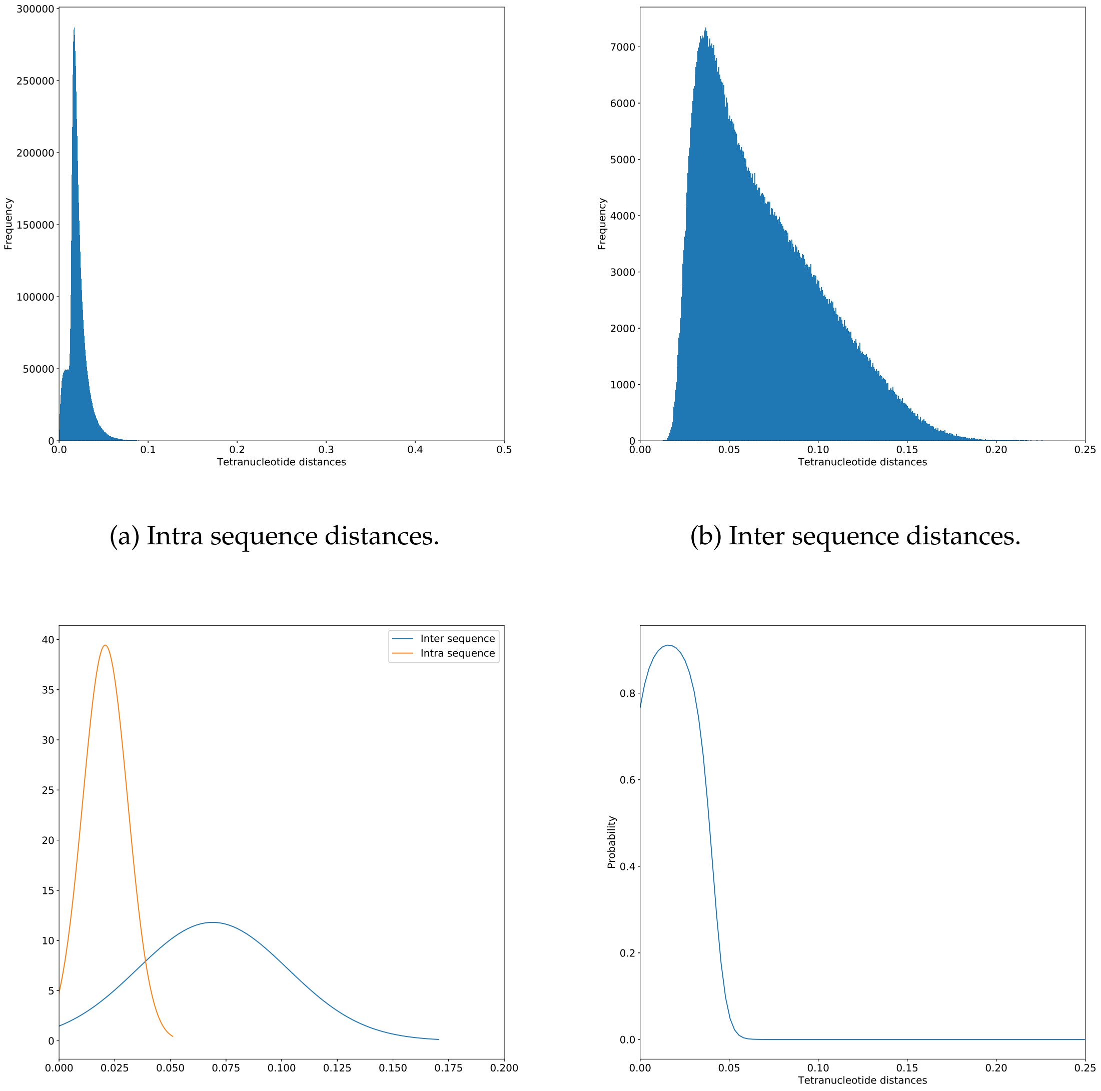
In order to obtain the final probability of random walk from unlabelled vertices to labelled vertices, we use equation 7.3 where n is the number of iterations until the convergence. Y_l is the initial label probabilities of vertices. In contrast with (Zhu and Ghahramani, 2002), we assume separate labels for each labelled vertex despite them being from the two classes, *plasmid* and *chromosome*. This enables us to compute probability of random walk from each unlabelled vertex to each of the labelled vertices. In GraphPlas, we terminate the operation at either 1000 iterations or when the difference is smaller than ϵ(=0.00001). Resulting matrix P_l consists of required random walk probabilities from a given node to another.-

Computing similarity of contigs using composition

The composition similarity S_k is computed using equation 7.4 (Wu, Simmons et al., 2015).

$$S_k = \frac{\mathcal{N}(D_e(V_i, V_j)|\mu_{intra}, \sigma^2_{intra})}{\mathcal{N}(D_e(V_i, V_j)|\mu_{intra}, \sigma^2_{intra}) + \mathcal{N}(D_e(V_i, V_j)|\mu_{inter}, \sigma^2_{inter})} \tag{7.4}$$

Here $D_e(V_i, V_j)$ represents Euclidean distance between 4-mer (tetramer) vectors of vertices V_i and V_j respectively. For this formula we use the approach presented by (Wu, Simmons et al., 2015). The mean and standard deviation are computed for each of the tetramer frequency vector distances within



(a) Intra sequence distances.

(b) Inter sequence distances.

(c) Distribution of distances.

(d) Probability vs distance.

Figure 7.4: Distance histograms for intra and inter sequence distances for normalized tetranucleotide distances. (a) and (b) demonstrate the histograms for Euclidean distances of tetramer vectors for sequences within a given species and between different species respectively. (c) demonstrates the resulting normal distributions plotted using means and standard deviations obtained from (a) and (b). (d) demonstrates the probability vs distance curve computed using equation 7.4.

and between different microbial species. μ_{intra} and μ_{inter} represents mean distances and, σ_{intra} and σ_{inter} represents standard deviations of distances within and between species respectively. In order to estimate μ_{intra}, σ_{intra}, μ_{inter} and σ_{inter}, we consider the set of all the reference chromosomes and plasmid assemblies from NCBI RefSeq database. First we seed 50 subsequences of length 10,000 base pairs from each of the reference sequences and compute their normalized tetranucleotide frequency vectors. The resulting histograms are presented in Figure 7.4. From these histograms we compute that $\mu_{intra} = 0.02$, $\sigma_{intra} = 0.010/2$, $\mu_{inter} = 0.069$ and $\sigma_{inter} = 0.034$. However, for probability computations in equation 7.4, we use $\mu_{intra} = 0$ to ensure that nearly identical sequences will have high similarity similar to (Wu, Simmons et al., 2015).

Computing similarity of contigs using coverage

The similarity of two coverages is computed as $S_c = Poisson(Cov(V_i)|Cov(V_j))$, where $Cov(V_i)$ and $Cov(V_i)$ demonstrate the coverage of contigs corresponding to vertices V_i and V_j respectively similar to work done by (Wu, Simmons et al., 2015).

7.2.2 Initial Classification

GraphPlas is capable of obtaining the seed classifications from either PlasClass or PlasFlow. The composition profiles tend to deviate significantly as the length becomes shorter. Hence, we chose 1,000 base pairs as the cutoff for the initial classification, similar to the default settings of many other binning tools in metagenomics (Wang, Wang, Lu et al., 2019; Wickramarachchi, Mallawaarachchi, Rajan et al., 2020a; Wu, Simmons et al., 2015). Therefore, we consider contigs that are longer than 1,000 base pairs for the initial results of PlasClass and PlasFlow.

We chose the most confident set of classifications from the classification result provided by the selected tool. First we order the contigs based on the predicted probability in the decreasing order. We label 50% of the contigs below 0.5 probability threshold as *chromosomes* in both tools. We label the top 10% above 0.5 probability threshold as *plasmids* for PlasClass. We chose 20% for PlasFlow as it tends to predict plasmids with a slightly lower tendency. The contigs that are neither classified as plasmids nor chromosomes are labelled as *unclassified*.

We introduce *labelled components* and *unlabelled components* to support the next step. A *labelled component* is defined as a component in the assembly graph with at least one contig with a label other than *unclassified*. Conversely, an *unlabelled component* is a component who's all contigs are labelled as *unclassified*. Refer to Figure 7.5(a) for the classification result of dataset **Sim-2C9P** using PlasClass. The initial classification result that is chosen by GraphPlas is demonstrated in Figure 7.5(b), and unlabelled components are circled in red color.

7.2.3 Processing labelled components

In this step, we consider components of the assembly graph that have at least one labelled contig from first step. Using the assembly graph and its labelled vertices from first step, we label the rest of the unclassified contigs. We first consider the contigs that are either 1,000 base pairs or longer for labelling.

We propagate the labels from labelled contigs to the other contigs based on the distance computed under all three topological, composition and coverage similarities. Equation 7.5 is used to compute the combined distance $D(V_i, V_j)$ between an unlabelled and a labelled vertex.

$$D(V_i, V_j) = -log(S_t \times S_k \times S_c) \tag{7.5}$$

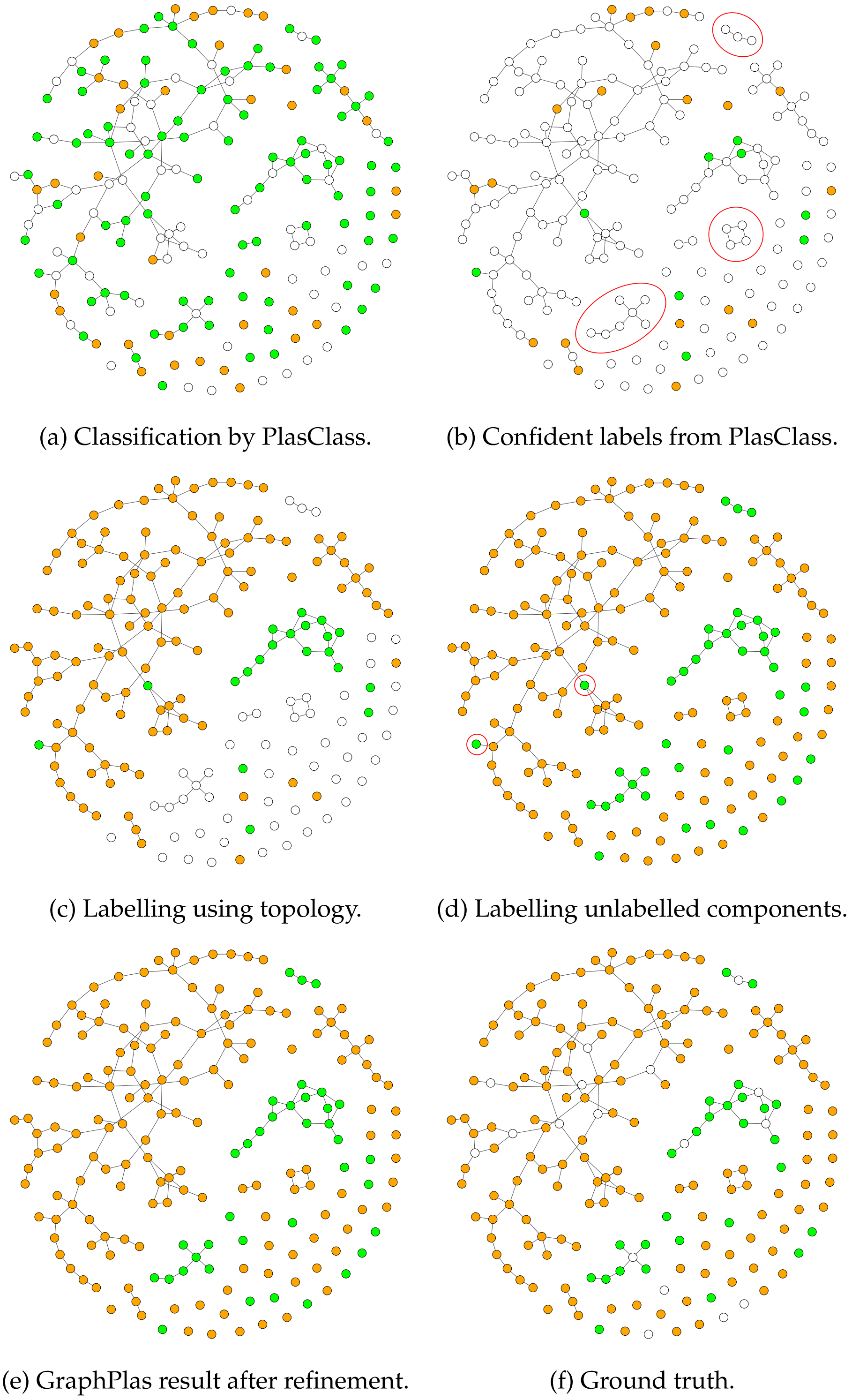
Next we use equation 7.6 to compute the distance $D_{short}(V_i, V_j)$ for the contigs that are shorter than 1,000 base pairs. Composition of shorter contigs is not considered because composition information can be unreliable for shorter contigs (Wickramarachchi, Mallawaarachchi, Rajan et al., 2020a). Hence, the distance computation is limited only to topological and coverage similarities.

$$D_{short}(V_i, V_j) = -log(S_t \times S_c) \tag{7.6}$$

We use the above equations as the distance metric for a KNN (K-Nearest Neighbors) classifier with up to 5 nearest neighbors. The contigs are then labelled using the majority vote. This classification is done in a step wise fashion for long and short contigs (shorter than 1,000 base pairs) in order. Refer to Figure 7.5(c) for the classification result of dataset **Sim-2C9P** after processing labelled components.

7.2.4 Processing unlabelled components

In this step we label the contigs from assembly graph components that do not contain any labelled contigs. Therefore, we rely on the labelled set of contigs in the entire assembly graph in this step. In order to label such isolated components in the graph, we use a KNN classifier using only the



(a) Classification by PlasClass.

(b) Confident labels from PlasClass.

(c) Labelling using topology.

(d) Labelling unlabelled components.

(e) GraphPlas result after refinement.

(f) Ground truth.

Figure 7.5: Assembly graph with contig labels for the dataset **Sim-2C9P** at different stages of GraphPlas. Chromosomes and plasmids are represented in orange and green colors respectively. Unlabelled components are circled in (b). Contigs that need to be refined are circled in (d). The contigs without a unique mapping to a single class are indicated in white.

composition and coverage information. We use equations 7.7 and 7.8 to compute the distances $D_{isolated}(V_i, V_j)$ and $D_{isolated\_short}(V_i, V_j)$ respectively. $D_{isolated}(V_i, V_j)$ is computed for vertices that are 1,000base pairs or longer and $D_{isolated\_short}(V_i, V_j)$ is computed for contigs that are shorter than 1,000 base pairs.

$$D_{isolated}(V_i, V_j) = -log(S_k \times S_c) \tag{7.7}$$

$$D_{isolated\_short}(V_i, V_j) = -log(S_c) \tag{7.8}$$

Similar to previous steps, we use equations 7.7 and 7.8 as the distance metric for a KNN classifier to classify the vertices in the isolated components. The KNN classifier is initiated on contigs longer than 1,000 base pairs with only two neighboring vertices. This is because the longer contigs in the assembly graph tend to have better coverage and composition representations. Furthermore, contigs at the repeat points have multiple edges and elevated coverage values. Hence, such contigs are avoided by limiting the number of neighbors to a maximum of 2. Majority voting of up to 5 neighbors is used for the labelling process. Refer to Figure 7.5(d) for the classification result of dataset **Sim-2C9P** after processing labelled components. Note that there are still unsupported labels from the initial classification, circled in red. There are no neighboring contigs that support the labels of such contigs.

7.2.5 Refining the labels

We define contigs that are connected to other contigs from a different label without any support of its own label as *ambiguously* labelled contigs. In this step, we utilize the assembly graph to correct the labels of such ambiguously labelled contigs. Majority voting is used in order to correct such labels. We start the label correction from non-leaf vertices (*i.e.* vertices with more than 1 neighbor) since they are more informative in terms of neighbors. Finally, the labels of the leaf vertices (*i.e.* vertices with only one neighbor) are corrected to match the neighboring vertex. Figure 7.5(e) demonstrates the assembly graph once GraphPlas refines the final result from previous steps.

7.3 Experimental Setup

7.3.1 Datasets

We evaluated GraphPlas using four simulated datasets and one real dataset with varying complexities and plasmid copy numbers. The information on the datasets considered are as follows. Please refer to Appendix Table B.8 for the detailed information of the simulation.

Table 7.1: Information on the datasets used for the experiments.

Dataset	Read length (base pairs)	Number of reads	Number of contigs	Edges in the graph
Sim-2C5P	300	239425	128	471
Sim-2C9P	300	368632	187	921
Sim-10C25P	300	1359961	636	2913
Sim-14C38P	300	3371230	1881	3977
Wastewater-Plas†	125	8757400	32510	3215

†We only considered contigs with length 1,000 base pairs or longer. A unique ground truth was discovered only for 436 contigs. However, the complete graph was utilized in the program.

1. We simulated four datasets using InSilicoSeq simulator (Gourlé et al., 2018) with MiSeq configuration that produces reads of length 300 base pairs. Similar to the work done by Pellow *et al.*, (Pellow, Mizrahi et al., 2020) we used the equation $5 \times min(1, log(L)/10)$, where L is the length

of the plasmid reference, to compute probability of success for a geometric distribution to obtain the plasmid copy numbers. These copy numbers were used to calculate the simulation coverage of each plasmid. This was performed to amplify the plasmid copy numbers of shorter plasmids.

- **Sim-2C5P**: Contains 1 species with a total of 2 chromosomes and 5 plasmids.
- **Sim-2C9P**: Contains 1 species with a total of 2 chromosomes and 9 plasmids.
- **Sim-10C25P**: Contains 5 species with a total of 10 chromosomes and 25 plasmids.
- **Sim-14C38P**: Contains 7 species with a total of 14 chromosomes and 38 plasmids.

2. We used the wastewater plasmidome sample **ERR1538272** (referred as **Wastewater-Plas**) (Shi et al., 2018) in order to evaluate the performance of GraphPlas on real datasets. The dataset was assembled using metaSPAdes (Nurk et al., 2017). The dataset consists of Illumina HiSeq 2500 paired end reads with a read length of 125 base pairs.

Table 7.1 indicates the number of reads, contigs and the read length of each of the datasets assembled.

7.3.2 Evaluation Criteria

We evaluated GraphPlas using macro-averaged precision, recall and F1-score using the following standard equations where;

- TP_p: the number of actual plasmid sequences that were classified as plasmids (true positives for plasmids)
- TP_c: the number of actual chromosomal sequences that were classified as chromosomes (true positives for chromosomes)
- FP_p: the number of non-plasmid sequences that were classified as plasmid (false positives for plasmids)
- FP_c: the number of non-chromosomal sequences that were classified as chromosomes (false positives for chromosomes)
- FN_p: the number of plasmid sequences that were not classified as plasmids (false negatives for plasmids)
- FN_c: the number of chromosomal sequences that were not classified as chromosomes (false negatives for chromosomes)

$$Precision(\%) = \frac{1}{2} \times \frac{TP_p}{TP_p + FP_p} + \frac{1}{2} \times \frac{TP_c}{TP_c + FP_c} \tag{7.9}$$

$$Recall(\%) = \frac{1}{2} \times \frac{TP_p}{TP_p + FN_p} + \frac{1}{2} \times \frac{TP_c}{TP_c + FN_c} \tag{7.10}$$

$$F1\ score(\%) = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{7.11}$$

The metrics for each class is averaged in order to obtain values with fair representation on each imbalanced class. Similar to previous evaluations in plasmid studies (Pellow, Mizrahi et al., 2020), the fraction of plasmid contigs recovered, $TP_p/(TP_p + FN_p)$, is also considered in our comparison. For the set of simulated datasets, the ground truth label was assigned by mapping the assembled contigs to the respective set of reference genomes. The mapping was performed using Minimap 2.1 (Li, 2018a). Only the contigs with a unique mapping to either plasmids or chromosomes were considered in the evaluation. Furthermore, the assembled contigs from the real dataset, with unknown ground truth were aligned to the NCBI assemblies. The contigs that had a unique mapping to either plasmids or chromosomes were considered in the evaluation. Moreover, we only considered the contigs that are either 1,000 base pairs or longer with an alignment beyond 50% of the query length.

7.4 Results and Discussion

Table 7.2: Comparison of macro-averaged classification results of PlasClass (Pellow, Mizrahi et al., 2020) and GraphPlas with PlasClass as the initial classifier.

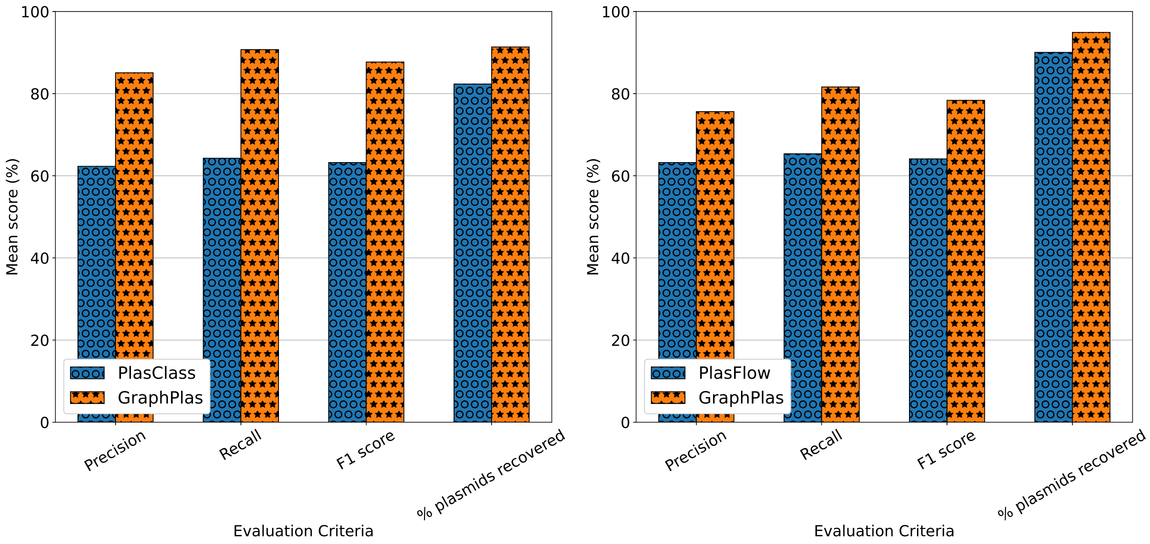
Datase	Tool used	Precision (%)	Recall (%)	F1 score (%)	Percentage of plasmids recovered (%)
Sim-2C5P	PlasClass	58.70	56.01	57.33	86.00
	GraphPlas	**99.02**	**99.32**	**99.17**	**100.00**
Sim-2C9P	PlasClass	58.92	64.48	61.58	92.59
	GraphPlas	**93.55**	**98.60**	**96.01**	**100.00**
Sim-10C25P	PlasClass	60.82	64.06	62.40	84.06
	GraphPlas	**85.20**	**93.51**	**89.17**	**100.00**
Sim-14C38P	PlasClass	53.88	58.81	56.23	80.00
	GraphPlas	**66.66**	**81.82**	**73.47**	**82.44**
Wastewater-Plas	PlasClass	79.18	77.97	78.57	69.04
	GraphPlas	**80.98**	**80.35**	**80.67**	**74.39**

Table 7.3: Comparison of macro averaged classification results of PlasFlow (Krawczyk et al., 2018) and GraphPlas with PlasFlow as the initial classifier.

Dataset	Tool used	Precision (%)	Recall (%)	F1 score (%)	Percentage of plasmids recovered (%)
Sim-2C5P	PlasFlow	65.90	59.33	62.44	94.00
	GraphPlas	**100.00**	**100.00**	**100.00**	**100.00**
Sim-2C9P	PlasFlow	59.05	64.83	61.80	92.59
	GraphPlas	**60.98**	**66.43**	**63.59**	**100.00**
Sim-10C25P	PlasFlow	62.53	65.82	64.13	89.13
	GraphPlas	**73.11**	**81.97**	**77.28**	**96.38**
Sim-14C38P	PlasFlow	57.14	65.88	61.02	94.15
	GraphPlas	**60.66**	**76.37**	**67.61**	**95.12**
Wastewater-Plas	PlasFlow	71.47	70.95	71.21	80.40
	GraphPlas	**83.36**	**83.40**	**83.38**	**83.07**

7.4.1 Recovery of Plasmids from Metagenomics Assemblies

In this section, we present the results of GraphPlas on several assembled datasets including a real dataset. All the datasets were assembled using metaSPAdes (Nurk et al., 2017) assembler. Table 7.2 and Table 7.3 show the comparison results with PlasClass and PlasFlow respectively. In Figure 7.6 the performance values are summarized for GraphPlas, PlasClass and PlasFlow. Furthermore, we



(a) GraphPlas with PlasClass

(b) GraphPlas with PlasFlow

Figure 7.6: Mean binning results of GraphPlas with original PlasFlow (Krawczyk et al., 2018) and PlasClass (Pellow, Mizrahi et al., 2020) results for all the datasets.

Table 7.4: Contig classification by GraphPlas with PlasFlow as Initial Classifier.

Classification	Dataset	Total Number of Contigs	Plasmids Classified as Plasmids	Chromosomes Classified as Chromosomes	Plasmids Classified as Chromosomes	Chromosomes Classified as Plasmids
GraphPlas with PlasClass	Sim-2C5P	123	50	72	0	1
	Sim-2C9P	170	27	139	0	4
	Sim-10C25P	585	138	389	0	4
	Sim-14C38P	1800	169	1295	36	300
	Wastewater-Plas	953	334	435	115	69
GraphPlas with PlasFlow	Sim-2C5P	123	50	73	0	0
	Sim-2C9P	170	27	47	0	96
	Sim-10C25P	585	133	302	5	145
	Sim-14C38P	1800	195	919	10	676
	Wastewater-Plas	953	373	422	76	82

show the results graphically in Figure 7.2 and Figure 7.3, where we indicate the starting points for GraphPlas, which are essentially high confidence points on the PlasClass result. Starting from those points GraphPlas pushes towards increasing direction on both recall and precision. In more challenging scenarios with relatively lower precision values, GraphPlas improves the recall while maintaining the same precision.

According to Table 7.2 and Table 7.3, it is evident that the utilization of assembly graph with composition and coverage information improves the results of plasmid detection over conventional machine learning approaches. Furthermore, the significant compromise on recall to achieve higher precision is also mitigated in GraphPlas leading to better F1-scores.

The demonstrated improvements in GraphPlas prevail due to two main reasons. Firstly, we utilize the most confident classifications provided by the initialization tools. Hence, at the starting point, the bootstrapping labels are more accurate. Secondly we employ the initial set of labels to train other contigs based on the composition, coverage and topology of the assembly graph. Note that it is highly likely for contigs of the same species to demonstrate a link in the assembly graph since there exists a

Table 7.5: Running time and memory consumed by GraphPlas (using PlasClass as the initial classifier).

Classification	Dataset	Running time (s)	Peak Memory Usage (MB)
GraphPlas with PlasClass	Sim-2C5P	4	112
	Sim-2C9P	6	114
	Sim-10C25P	21	140
	Sim-14C38P	79	158
	Wastewater-Plas	77	196
GraphPlas with PlasFlow	Sim-2C5P	3	112
	Sim-2C9P	4	114
	Sim-10C25P	21	140
	Sim-14C38P	79	158
	Wastewater-Plas	80	200

path in the de Bruijn graph that completes the reference genome (Compeau et al., 2011). Therefore, GraphPlas is always capable of classifying contigs more confidently and reliably. Moreover, the shorter lengths of contigs do not affect the results significantly because of the connected contigs that are long and confidently classifiable. This is further evident through Figure 7.5 where the labelling of contigs in each step are demonstrated. Finally, the actual number of contigs classified into each class are tabulated in Table 7.4. Note that the improvements over the real dataset is not significant since we have limited the ground truth computations only for contigs longer than 1,000 base pairs. Hence, the classification of contigs shorter than the threshold are not considered.

7.5 Implementation

GraphPlas consists of two main components as the initial classifier and the graph classifier. We have integrated the PlasClass classifier as the initial classifier. The entire program is implemented using python 3.6.7 and tested on an Intel Core i7-7700 CPU @ 3.60GHz×8 machine with 16 GB of RAM. The host operating system was Ubuntu 18.04.3 LTS. Multithreading capabilities are used for computing the tetramer frequency vectors and in the KNN classifier. The summary of resource utilization is tabulated under Table 7.5. GraphPlas only considers the longer contigs for computation of plasmid probabilities. Hence, the memory utilization for all the experiments were below 300 MB. Furthermore, the GraphPlas algorithm completes within 4 minutes for all the datasets considered.

7.6 Discussion

7.6.1 Summary

GraphPlas proposes the ideology of incorporating the assembly graph in plasmid classification. We designed and evaluated GraphPlas which combines conventional machine learning tools with topological information from the assembly graph for the detection of plasmids. We also highlighted the importance of assembly graph and its potential to support in bootstrapping a dataset-specific model to address the problem of plasmid detection.

7.6.2 Room for improvements

The inclusion of assembly graph information to improve performance of the plasmid classification has room for further improvements. The erroneous classifications in the initial seed contigs could mislead the label propagation, degrading the overall performance. Furthermore, the coverage of contigs could be inaccurate, leading to misclassifications and hinder the label refinement. Hence, use of novel coverage computation techniques apart from what the assemblers report could be considered as future improvements (*e.g.*, CoverM from `https://github.com/wwood/CoverM`). In the future, we intend to investigate the viability of using third-generation sequencing (TGS) assemblies for the recovery of plasmid sequences. Moreover, graph representation learning approaches can potentially lead to improvements in the classification of plasmids due to their applicability in much larger graphs and graphs with noise.

Chapter 8

Conclusion and Future Work

The thesis presented tools that progressively improves metagenomics binning and novel techniques for plasmids recovery. First, we presented MetaBCC-LR which clustered long reads using coverage and composition in a stepwise manner. MetaBCC-LR introduces the *k*-mer coverage histogram as a feature for coverage based binning of long reads. We improved MetaBCC-LR's approach by introducing LRBinner which uses an auto-encoder to combine features meaningfully. This eliminated the necessity of stepwise clustering of reads which has the tendency of false bin fragmentation in certain scenarios. Moreover, we implemented an algorithm to extract clusters efficiently using the density of bins. However, the issue of imbalanced classes and approximate coverage estimation remained to be a limitation as we were using the same input features for binning. This means, smaller bins and noisy clusters were indistinguishable, resulting in difficulties of clustering. Moreover, the coverage estimate was rather an approximation which was limited to *k*-mer coverage. We addressed these issues using OBLR which introduced the idea of read-overlap graph. Read-overlap graph was able to produce more information including read coverage, read connectivity and a sampling measure to obtain a balanced dataset for clustering. Hence, we were able to perform binning more effectively and accurately in OBLR compared to its predecessors.

Another area of study in this thesis was the recovery of plasmids. This is a similar problem to metagenomics binning which is studied as a binary classification problem. We introduced the idea of combining reference based methods with dataset features to improve plasmid recovery in PlasLR and GraphPlas. Both PlasLR and GraphPlas were of semi-supervised nature where existing classifiers were used alongside dataset specific information to arrive at better results. Most importantly, we were able to expand the agility of existing tools, that were trained machine learning classifiers, to use dataset specific information that were not available during the design of those classifiers. This technique can potentially improve not only plasmid classifications, but also can be applied in metagenomics binning. Such semi-supervised approaches are discussed in the following sections under future work.

Improvements to models and algorithms I have implemented can be investigated to explore the possibility pushing the limits of metagenomics analysis even further. Mainly, the scope of the thesis was limited to reference-free metagenomics binning. Furthermore, plasmid discovery was discussed based on data-driven methods rather than direct reference based methods such as k-mer index (*e.g.*, Kraken2 (Wood, Lu et al., 2019)) and gene based (*e.g*, Platon (Schwengers et al., 2020)) methods. Moreover, presented methodologies are standalone tools, implemented as preprocessors for metagenomics assembly. This is mainly because, the literature explores metagenomics binning as a separate problem since contigs binning is the preferred approach in NGS based metagenomics analyses.

8.1 Metagenomics Binning and Assembly of Long Reads

As a part of future microbiome exploration, I plan to expand on several aspects of metagenomics analysis.

8.1.1 Simultaneous Binning and Assembly

The binning tools developed and presented in the thesis focussed on performing metagenomic binning as a pre-processor for metagenomic assembly. Hence, the process of assembly is unaware of the binning process and vice-versa. This can, not often though, lead to poor separation of reads in shared regions. Moreover, metagenomic assemblers are sensitive towards variations in coverage and can perform assembly of low coverage regions. It is possible to have fragmented assemblies if regions of the same genome have significantly different coverages. Therefore, it is important to couple binning and assembly to avoid duplicate and fragmented assemblies that may arise due to false binning of certain regions in the genomes. Moreover, having an assembly pipeline that is aware of the binning process can improve the speed of assemblies by performing parallel assembly.

8.1.2 Reads from Similar Regions Across Different Species

Species in metagenomics datasets can have common genomic regions, resulting in highly similar reads across such species from those regions. However, the developed binning approaches bin a read only to a particular cluster and never consider the possibility of a read belonging to multiple bins. This is not a commonly studied phenomenon in metagenomics binning. However, limited studies have been conducted related to contigs binning (Mallawaarachchi, Wickramarachchi et al., 2020b) by assigning contigs to multiple bins. Therefore, it is evident that shared regions can exist in metagenomic datasets and assemblies can indeed be improved by duplicating such reads across bins. Hence, improvements in binning algorithms to handle such shared regions, ideally outside the initial clustering phase can improve the binning performance. Moreover, such an approach can be thought to reduce the fragmentation that may otherwise arise. Figure 8.1 illustrates a generic binning pipeline for binning shared reads extending the latest binning tool of the thesis, OBLR. Note that the key difference is the multi-label GNN model which facilitates the assignment of a given read to multiple bins.

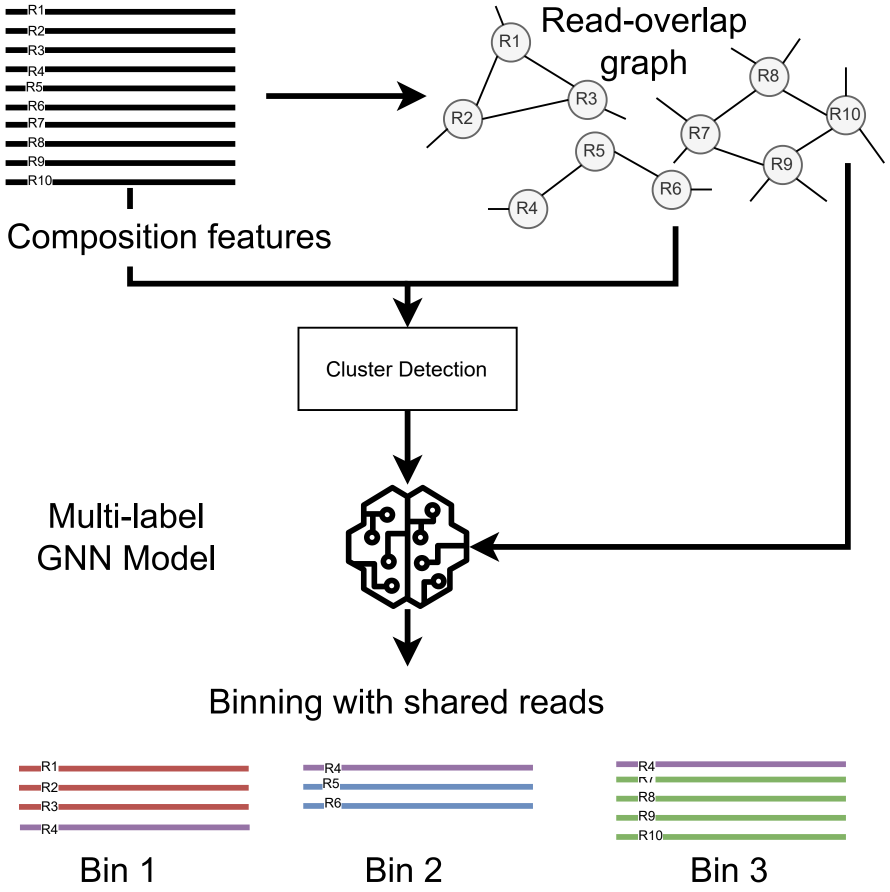


Figure 8.1: Binning shared reads by assigning shared reads into all the potential bins.

8.1.3 Semi-supervised Binning and Plasmid Recovery

The scope of the thesis was limited to reference free approaches and data driven plasmid recovery techniques (*i.e.*, oligonucleotide frequency based analysis). They are mostly effective in scenarios where the environment is either novel or the references used are incomplete. However, there is potential to use reference based methods to improve the accuracy binning. This is because, reference based methods are capable of providing viable annotations at a higher taxonomic level if the particular species cannot be annotated with greater confidence (Huson, Albrecht et al., 2018; Wood, Lu et al., 2019).

Reference based methods can still provide reasonable binning results by observing the unique regions in genomes of different species. Labels of such unique regions can be combined with read overlap graph along with the resulting cluster sizes (with some threshold size) to improve binning. An interesting study would be to perform binning at different taxonomic levels based on the classifications produced by reference based methods such as Kraken2. In this case, Figure 8.1 will have an aforementioned semi-supervised pipeline tool in place of *Cluster Detection* to complete the binning pipeline. Please refer to Figure 8.2 for an example pipeline for semi-supervised binning. Note that taxonomy aware label propagation will only propagate species labels into other reads having a suitable genus.

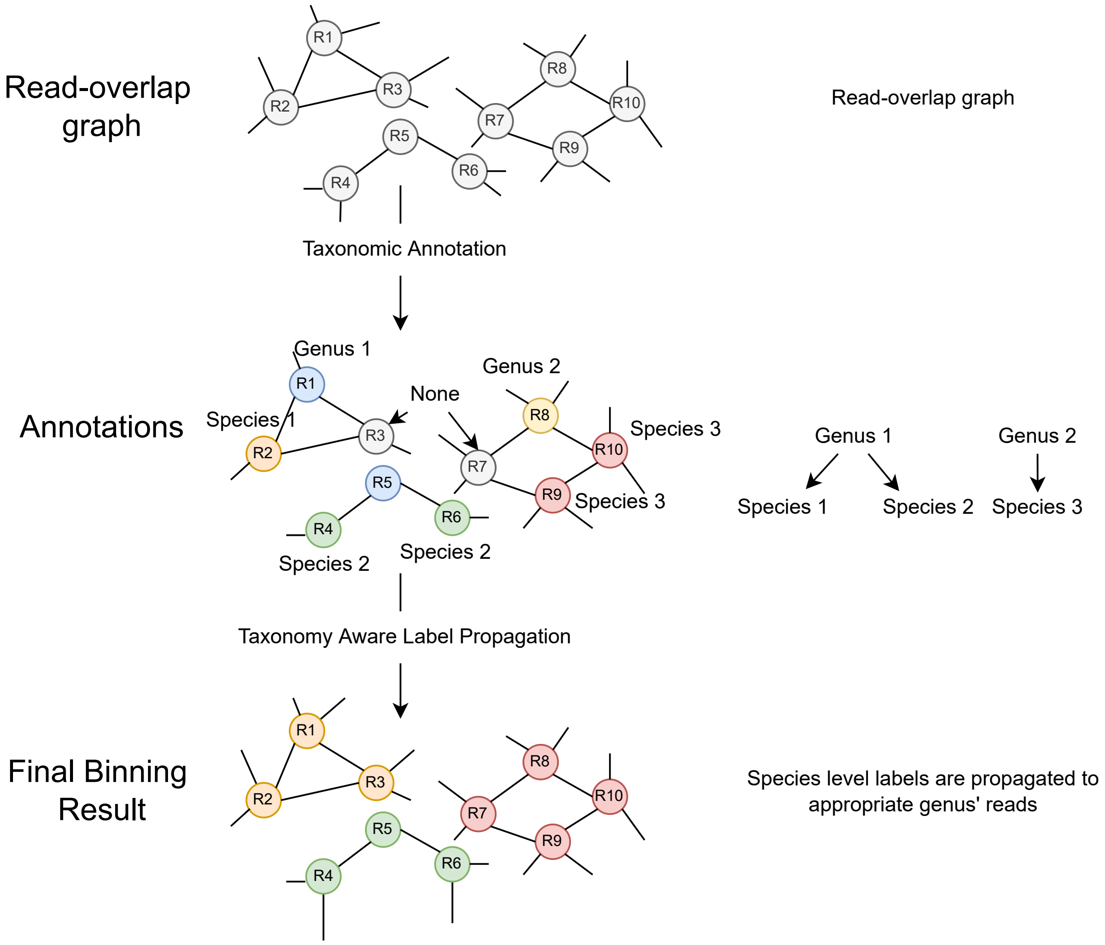


Figure 8.2: Binning reads using semi-supervised annotations.

The above approach can be extended to plasmid classifications using an appropriate database, *e.g.*, Kraken2 plasmid database (Wood, Lu et al., 2019) because, currently plasmids are studied as a single entity in a given dataset. However, plasmids often associate a bacterial origin which can be isolated in the binning process. This would not only improve the resolution of plasmid study but also enable better application of reference based studied. For example, bins can be studied under different levels of taxa followed by a plasmid study. Moreover, we can further annotate plasmids based on the species level classification because not all bacteria carry plasmids. This could help us analyze and isolate the exact plasmid present, and its role in the environment.

Bibliography

Abe, T., Kanaya, S., Kinouchi, M., Ichiba, Y., Kozuki, T. and Ikemura, T. (2003). 'Informatics for Unveiling Hidden Genome Signatures'. *Genome Research*, 13(4), pp. 693–702 (cited on pp. 14, 55).

Albrecht, B., Bağcı, C. and Huson, D. H. (2020). 'MAIRA-real-time taxonomic and functional analysis of long reads on a laptop'. *BMC bioinformatics*, 21(13), pp. 1–12 (cited on p. 5).

Alneberg, J., Bjarnason, B. S., Bruijn, I. de et al. (2014). 'Binning metagenomic contigs by coverage and composition'. *Nature Methods*, 11, p. 1144 (cited on p. 30).

Antipov, D., Hartwick, N., Shen, M., Raiko, M., Lapidus, A. and Pevzner, P. A. (2016). 'plasmidSPAdes: assembling plasmids from whole genome sequencing data'. *Bioinformatics*, 32(22), pp. 3380–3387. DOI: 10.1093/bioinformatics/btw493 (cited on pp. 5, 10, 12, 55).

Antipov, D., Raiko, M., Lapidus, A. and Pevzner, P. A. (2019). 'Plasmid detection and assembly in genomic and metagenomic data sets'. *Genome research*, 29(6), pp. 961–968 (cited on p. 10).

Arredondo-Alonso, S., Rogers, M. R. C., Braat, J. C., Verschuuren, T. D., Top, J., Corander, J., Willems, R. J. L. and Schürch, A. C. (2018). 'mlplasmids: a user-friendly tool to predict plasmid- and chromosome-derived sequences for single species'. *Microbial Genomics*, 4(11)e000224. DOI: https://doi.org/10.1099/mgen.0.000224 (cited on pp. 9, 10).

Balvert, M., Luo, X., Hauptfeld, E., Schönhuth, A. and Dutilh, B. E. (2021). 'OGRE: Overlap Graph-based metagenomic Read clustEring'. *Bioinformatics*, 37(7), pp. 905–912 (cited on p. 12).

Bankevich, A., Nurk, S., Antipov, D., Gurevich, A. A., Dvorkin, M., Kulikov, A. S., Lesin, V. M., Nikolenko, S. I., Pham, S., Prjibelski, A. D. et al. (2012). 'SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing'. *Journal of computational biology*, 19(5), pp. 455–477 (cited on pp. 5, 12).

Bleidorn, C. (2016). 'Third generation sequencing: technology and its potential impact on evolutionary biodiversity research'. *Systematics and biodiversity*, 14(1), pp. 1–8 (cited on p. 5).

Canard, B. and Sarfati, R. S. (1994a). 'DNA polymerase fluorescent substrates with reversible 3'-tags'. *Gene*, 148(1), pp. 1–6 (cited on p. 4).

Canard, B. and Sarfati, S. (1994b). 'Novel derivatives usable for the sequencing of nucleic acids'. *CA Patent*, 2, p. 13 (cited on p. 4).

Carattoli, A. (2013). 'Plasmids and the spread of resistance'. *International Journal of Medical Microbiology*, 303(6). Special Issue Antibiotic Resistance, pp. 298–304. DOI: https://doi.org/10.1016/j.ijmm.2013.02.001 (cited on p. 2).

Carattoli, A., Zankari, E., Garcia-Fernandez, A., Voldby Larsen, M., Lund, O., Villa, L., Møller Aarestrup, F. and Hasman, H. (2014). 'In Silico Detection and Typing of Plasmids using PlasmidFinder and Plasmid Multilocus Sequence Typing'. *Antimicrobial Agents and Chemotherapy*, 58(7), pp. 3895–3903. DOI: 10.1128/AAC.02412-14 (cited on pp. 9, 10).

Casali, N. and Preston, A. (2003). *E. coli plasmid vectors: methods and applications*. Vol. 235. Springer Science & Business Media (cited on p. 10).

Chen, K. and Pachter, L. (2005). 'Bioinformatics for Whole-Genome Shotgun Sequencing of Microbial Communities'. *PLOS Computational Biology*, 1(2) (cited on p. 2).

Compeau, P. E. C., Pevzner, P. A. and Tesler, G. (2011). 'How to apply de Bruijn graphs to genome assembly'. *Nature Biotechnology*, 29(11), pp. 987–991. DOI: 10.1038/nbt.2023 (cited on p. 76).

Davis, J. J. and Olsen, G. J. (2009). 'Modal Codon Usage: Assessing the Typical Codon Usage of a Genome'. *Molecular Biology and Evolution*, 27(4), pp. 800–810. DOI: 10.1093/molbev/msp281 (cited on p. 55).

De Filippis, F., Parente, E. and Ercolini, D. (2017). 'Metagenomics insights into food fermentations'. *Microbial biotechnology*, 10(1), pp. 91–102 (cited on p. 2).

De Maio, N., Shaw, L. P., Hubbard, A., George, S., Sanderson, N. D., Swann, J., Wick, R., AbuOun, M., Stubberfield, E., Hoosdally, S. J. et al. (2019). 'Comparison of long-read sequencing technologies in the hybrid assembly of complex bacterial genomes'. *Microbial genomics*, 5(9) (cited on p. 4).

Deschavanne, P. J., Giron, A., Vilain, J., Fagot, G. and Fertil, B. (1999). 'Genomic signature: characterization and classification of species assessed by chaos game representation of sequences.' *Molecular Biology and Evolution*, 16(10), pp. 1391–1399 (cited on p. 14).

DiGuistini, S., Liao, N. Y., Platt, D., Robertson, G., Seidel, M., Chan, S. K., Docking, T. R., Birol, I., Holt, R. A., Hirst, M. et al. (2009). 'De novo genome sequence assembly of a filamentous fungus using Sanger, 454 and Illumina sequence data'. *Genome biology*, 10(9), pp. 1–12 (cited on p. 4).

Ekim, B., Berger, B. and Chikhi, R. (2021). 'Minimizer-space de Bruijn graphs: Whole-genome assembly of long reads in minutes on a personal computer'. *Cell Systems* (cited on p. 5).

Ester, M., Kriegel, H.-P., Sander, J. and Xu, X. (1996). 'A Density-based Algorithm for Discovering Clusters a Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise'. In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD'96. Portland, Oregon: AAAI Press, pp. 226–231 (cited on p. 14).

Feng, X., Cheng, H., Portik, D. and Li, H. (2021). 'Metagenome assembly of high-fidelity long reads with hifiasm-meta'. *arXiv preprint arXiv:2110.08457* (cited on pp. 5, 52).

Fey, M. and Lenssen, J. E. (2019). 'Fast Graph Representation Learning with PyTorch Geometric'. In: *ICLR Workshop on Representation Learning on Graphs and Manifolds* (cited on p. 50).

Forster, S. C. et al. (2019). 'A human gut bacterial genome and culture collection for improved metagenomic analyses'. *Nature Biotechnology*, 37(2), pp. 186–192. DOI: 10.1038/s41587-018-0009-7 (cited on p. 2).

Friehs, K. (2004). 'Plasmid copy number and plasmid stability'. *New trends and developments in biochemical engineering*, pp. 47–82 (cited on p. 9).

Gill, D. R., Pringle, I. A. and Hyde, S. C. (2009). 'Progress and prospects: the design and production of plasmid vectors'. *Gene therapy*, 16(2), pp. 165–171 (cited on p. 3).

Girotto, S., Pizzi, C. and Comin, M. (2016). 'MetaProb: accurate metagenomic reads binning based on probabilistic sequence signatures'. *Bioinformatics*, 32(17), pp. i567–i575 (cited on pp. 8, 20).

Gomi, R., Wyres, K. L. and Holt, K. E. (2021). 'Detection of plasmid contigs in draft genome assemblies using customized Kraken databases'. *Microbial genomics*, 7(4) (cited on p. 9).

Gourlé, H., Karlsson-Lindsjö, O., Hayer, J. and Bongcam-Rudloff, E. (2018). 'Simulating Illumina metagenomic data with InSilicoSeq'. *Bioinformatics*, 35(3), pp. 521–522. DOI: 10.1093/bioinformatics/bty630 (cited on pp. 72, 100).

Gupta, P. K. (2008). 'Single-molecule DNA sequencing technologies for future genomics research'. *Trends in biotechnology*, 26(11), pp. 602–611 (cited on p. 5).

Hamilton, W. L., Ying, R. and Leskovec, J. (2017). 'Inductive representation learning on large graphs'. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 1025–1035 (cited on p. 47).

Harris, C. R. et al. (2020). 'Array programming with NumPy'. *Nature*, 585, pp. 357–362. DOI: 10.1038/s41586-020-2649-2 (cited on p. 40).

Huson, D. H., Albrecht, B., Bağcı, C., Bessarab, I., Gorska, A., Jolic, D. and Williams, R. B. (2018). 'MEGAN-LR: new algorithms allow accurate binning and easy interactive exploration of metagenomic long reads and contigs'. *Biology direct*, 13(1), pp. 1–17 (cited on pp. 5, 7, 80).

Huson, D. H., Auch, A. F., Qi, J. and Schuster, S. C. (2007). 'MEGAN analysis of metagenomic data'. *Genome research*, 17(3), pp. 377–386 (cited on p. 7).

Hyatt, D., Chen, G.-L., LoCascio, P. F., Land, M. L., Larimer, F. W. and Hauser, L. J. (2010). 'Prodigal: prokaryotic gene recognition and translation initiation site identification'. *BMC bioinformatics*, 11(1), pp. 1–11 (cited on pp. 8, 12).

Hyman, E. D. (1988). 'A new method of sequencing DNA'. *Analytical biochemistry*, 174(2), pp. 423–436 (cited on p. 4).

Japkowicz, N. and Stephen, S. (2002). 'The class imbalance problem: A systematic study'. *Intelligent data analysis*, 6(5), pp. 429–449 (cited on p. 46).

Kang, D. D., Froula, J., Egan, R. and Wang, Z. (2015). 'MetaBAT, an efficient tool for accurately reconstructing single genomes from complex microbial communities'. *PeerJ*, 3, e1165 (cited on pp. 4, 8).

Kang, D. D., Li, F., Kirton, E., Thomas, A. et al. (2019). 'MetaBAT 2: an adaptive binning algorithm for robust and efficient genome reconstruction from metagenome assemblies'. *PeerJ*, 7, e7359 (cited on p. 30).

Kim, D., Song, L., Breitwieser, F. P. and Salzberg, S. L. (2016). 'Centrifuge: rapid and sensitive classification of metagenomic sequences'. *Genome Research*, 26(12), pp. 1721–1729 (cited on pp. 7, 10).

Kolmogorov, M., Bickhart, D. M., Behsaz, B., Gurevich, A., Rayko, M., Shin, S. B., Kuhn, K., Yuan, J., Polevikov, E., Smith, T. P. et al. (2020). 'metaFlye: scalable long-read metagenome assembly using repeat graphs'. *Nature Methods*, 17(11), pp. 1103–1110 (cited on pp. 12, 48, 50, 52).

Kolmogorov, M., Bickhart, D. M., Behsaz, B., Gurevich, A., Rayko, M., Shin, S. B., Kuhn, K., Yuan, J., Polevikov, E., Smith, T. P. L. and Pevzner, P. A. (2020). 'metaFlye: scalable long-read metagenome assembly using repeat graphs'. *Nature Methods*, 17(11), pp. 1103–1110. DOI: 10.1038/s41592-020-00971-x (cited on pp. 36, 38).

Kolmogorov, M., Rayko, M., Yuan, J. et al. (2019). 'metaFlye: scalable long-read metagenome assembly using repeat graphs'. *bioRxiv* (cited on pp. 5, 22, 23, 60, 98).

Kolmogorov, M., Yuan, J., Lin, Y. and Pevzner, P. A. (2019). 'Assembly of long, error-prone reads using repeat graphs'. *Nature Biotechnology*, 37(5), pp. 540–546 (cited on pp. 5, 13).

Koren, S., Walenz, B. P., Berlin, K. et al. (2017). 'Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation'. *Genome Research*, 27(5), pp. 722–736 (cited on pp. 5, 22).

Kouchaki, S., Tapinos, A. and Robertson, D. L. (2019). 'A signal processing method for alignment-free metagenomic binning: multi-resolution genomic binary patterns'. *Scientific Reports*, 9(1), p. 2159 (cited on pp. 13, 14).

Krawczyk, P. S., Lipinski, L. and Dziembowski, A. (2018). 'PlasFlow: predicting plasmid sequences in metagenomic data using genome signatures'. *Nucleic Acids Research*, 46(6), e35–e35. DOI: 10.1093/nar/gkx1321 (cited on pp. 5, 9, 10, 57–60, 65, 74, 75, 90).

Kullback, S. and Leibler, R. A. (1951). 'On Information and Sufficiency'. *The Annals of Mathematical Statistics*, 22(1), pp. 79–86. DOI: 10.1214/aoms/1177729694 (cited on p. 32).

Laczny, C. C., Sternal, T., Plugaru, V., Gawron, P., Atashpendar, A., Margossian, H. H., Coronado, S., Van der Maaten, L., Vlassis, N. and Wilmes, P. (2015). 'VizBin-an application for reference-independent visualization and human-augmented binning of metagenomic data'. *Microbiome*, 3(1), pp. 1–7 (cited on p. 8).

Laczny, C. C., Kiefer, C., Galata, V. et al. (2017). 'BusyBee Web: metagenomic data analysis by bootstrapped supervised binning and annotation'. *Nucleic Acids Research*, 45(W1), W171–W179 (cited on pp. 8, 20, 30, 36).

Laczny, C. C., Pinel, N., Vlassis, N. and Wilmes, P. (2014). 'Alignment-free Visualization of Metagenomic Data by Nonlinear Dimension Reduction'. *Scientific Reports*, 4. Article, p. 4516 (cited on p. 13).

Lanza, V. F., Toro, M. de, Garcillán-Barcia, M. P., Mora, A., Blanco, J., Coque, T. M. and Cruz, F. de la (2014). 'Plasmid Flux in Escherichia coli ST131 Sublineages, Analyzed by Plasmid Constellation Network (PLACNET), a New Method for Plasmid Reconstruction from Whole Genome Sequences'. *PLOS Genetics*, 10(12), pp. 1–21. DOI: 10.1371/journal.pgen.1004766 (cited on p. 5).

Levy, S. B., Fitzgerald, G. B. and Macone, A. B. (1976). 'Spread of antibiotic-resistant plasmids from chicken to chicken and from chicken to man'. *Nature*, 260(5546), pp. 40–42 (cited on p. 3).

Li, D., Liu, C.-M., Luo, R., Sadakane, K. and Lam, T.-W. (2015). 'MEGAHIT: an ultra-fast single-node solution for large and complex metagenomics assembly via succinct de Bruijn graph'. *Bioinformatics*, 31(10), pp. 1674–1676 (cited on p. 5).

Li, A.-D., Li, L.-G. and Zhang, T. (2015). 'Exploring antibiotic resistance genes and metal resistance genes in plasmid metagenomes from wastewater treatment plants'. *Frontiers in Microbiology*, 6, p. 1025. DOI: 10.3389/fmicb.2015.01025 (cited on p. 2).

Li, H. (2016). 'Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences'. *Bioinformatics*, 32(14), pp. 2103–2110 (cited on p. 45).

Li, H. (2018a). 'Minimap2: pairwise alignment for nucleotide sequences'. *Bioinformatics*, 34(18), pp. 3094–3100 (cited on pp. 20, 35, 60, 73).

Li, H. (2018b). 'Minimap2: pairwise alignment for nucleotide sequences'. *Bioinformatics*, 34(18), pp. 3094–3100 (cited on p. 48).

Lin, H.-H. and Liao, Y.-C. (2016). 'Accurate binning of metagenomic contigs via automated clustering sequences using information of genomic signatures and marker genes'. *Scientific reports*, 6. 27067514[pmid], pp. 24175–24175. DOI: 10.1038/srep24175 (cited on p. 34).

Lin, Y., Yuan, J., Kolmogorov, M. et al. (2016). 'Assembly of long error-prone reads using de Bruijn graphs'. *Proceedings of the National Academy of Sciences*, 113(52), E8396–E8405 (cited on p. 13).

Liu, X.-Y., Wu, J. and Zhou, Z.-H. (2009). 'Exploratory Undersampling for Class-Imbalance Learning'. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2), pp. 539–550. DOI: 10.1109/TSMCB.2008.2007853 (cited on p. 46).

LomanLab (2019). *Loman Lab Mock Community Experiments.* ↗ (cited on pp. 18, 19).

Long, P. E., Williams, K. H., Hubbard, S. S. and Banfield, J. F. (2016). 'Microbial metagenomics reveals climate-relevant subsurface biogeochemical processes'. *Trends in microbiology*, 24(8), pp. 600–610 (cited on p. 2).

Mallawaarachchi, V. and Lin, Y. (2022). 'Metacoag: Binning metagenomic contigs via composition, coverage and assembly graphs', pp. 70–85 (cited on pp. 4, 8).

Mallawaarachchi, V., Wickramarachchi, A. and Lin, Y. (2020a). 'GraphBin: refined binning of metagenomic contigs using assembly graphs'. *Bioinformatics*, 36(11), pp. 3307–3313 (cited on p. 4).

Mallawaarachchi, V. G., Wickramarachchi, A. S. and Lin, Y. (2020b). 'GraphBin2: Refined and Overlapped Binning of Metagenomic Contigs Using Assembly Graphs'. In: *20th International Workshop on Algorithms in Bioinformatics (WABI 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik (cited on pp. 4, 79).

Marcais, G. and Kingsford, C. (2012). 'Jellyfish: A fast k-mer counter'. *Tutorialis e Manuais*, 1, pp. 1–8 (cited on p. 13).

Margos, G., Hepner, S., Mang, C., Marosevic, D., Reynolds, S. E., Krebs, S., Sing, A., Derdakova, M., Reiter, M. A. and Fingerle, V. (2017). 'Lost in plasmids: next generation sequencing and the complex genome of the tick-borne pathogen Borrelia burgdorferi'. *BMC Genomics*, 18(1), p. 422. DOI: 10.1186/s12864-017-3804-5 (cited on p. 2).

McInnes, L., Healy, J. and Astels, S. (2017). 'hdbscan: Hierarchical density based clustering'. *Journal of Open Source Software*, 2(11), p. 205 (cited on p. 47).

McInnes, L., Healy, J. and Melville, J. (2020). *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction* (cited on pp. 47, 55, 57, 62).

Menzel, P., Ng, K. L. and Krogh, A. (2016). 'Fast and sensitive taxonomic classification for metagenomics with Kaiju'. *Nature Communications*, 7. Article, p. 11257 (cited on pp. 5, 7).

Meyer, F., Hofmann, P., Belmann, P., Garrido-Oter, R., Fritz, A., Sczyrba, A. and McHardy, A. C. (2018a). 'AMBER: Assessment of Metagenome BinnERs'. *GigaScience*, 7(6). giy069. DOI: 10.1093/gigascience/giy069 (cited on pp. 36, 104, 109).

Meyer, F., Hofmann, P., Belmann, P., Garrido-Oter, R., Fritz, A., Sczyrba, A. and McHardy, A. C. (2018b). 'AMBER: assessment of metagenome binners'. *Gigascience*, 7(6), giy069 (cited on pp. 48–50).

Meyer, F., Lesker, T.-R., Koslicki, D., Fritz, A., Gurevich, A., Darling, A. E., Sczyrba, A., Bremges, A. and McHardy, A. C. (2021). 'Tutorial: assessing metagenomics software with the CAMI benchmarking toolkit'. *Nature protocols*, 16(4), pp. 1785–1801 (cited on p. 4).

Mikheenko, A., Saveliev, V. and Gurevich, A. (2015a). 'MetaQUAST: evaluation of metagenome assemblies'. *Bioinformatics*, 32(7), pp. 1088–1090 (cited on pp. 23, 36).

Mikheenko, A., Saveliev, V. and Gurevich, A. (2015b). 'MetaQUAST: evaluation of metagenome assemblies'. *Bioinformatics*, 32(7), pp. 1088–1090. DOI: `10.1093/bioinformatics/btv697` (cited on pp. 61, 64).

Mikheenko, A., Saveliev, V. and Gurevich, A. (2016). 'MetaQUAST: evaluation of metagenome assemblies'. *Bioinformatics*, 32(7), pp. 1088–1090 (cited on pp. 48, 49).

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill ScienceEngineeringMath (cited on pp. 55, 58).

Nealson, K. H. (1997). 'Sediment bacteria: who's there, what are they doing, and what's new?' *Annual Review of Earth and Planetary Sciences*, 25(1), pp. 403–434 (cited on p. 2).

New nanopore sequencing chemistry in developers' hands; set to deliver Q20 (99%) "raw read" accuracy (2021). ↗ (cited on p. 52).

Nicholls, S. M., Quick, J. C., Tang, S. and Loman, N. J. (2019). 'Ultra-deep, long-read nanopore sequencing of mock microbial community standards'. *GigaScience*, 8(5). giz043 (cited on pp. 18, 23, 48, 58).

Nissen, J. N., Johansen, J., Allesøe, R. L., Sønderby, C. K., Armenteros, J. J. A., Grønbech, C. H., Jensen, L. J., Nielsen, H. B., Petersen, T. N., Winther, O. et al. (2021). 'Improved metagenome binning and assembly using deep variational autoencoders'. *Nature biotechnology*, 39(5), pp. 555–560 (cited on pp. 5, 8, 30–32, 34, 46).

Nolet, C. J., Lafargue, V., Raff, E., Nanditale, T., Oates, T., Zedlewski, J. and Patterson, J. (2020). *Bringing UMAP Closer to the Speed of Light with GPU Acceleration* (cited on p. 50).

Nurk, S., Meleshko, D., Korobeynikov, A. and Pevzner, P. A. (2017). 'metaSPAdes: a new versatile metagenomic assembler'. *Genome research*, 27(5), pp. 824–834 (cited on pp. 5, 12, 73, 74).

Ounit, R., Wanamaker, S., Close, T. J. and Lonardi, S. (2015). 'CLARK: fast and accurate classification of metagenomic and genomic sequences using discriminative k-mers'. *BMC Genomics*, 16(1), p. 236 (cited on p. 7).

Pabbathi, N. P. P., Velidandi, A., Tavarna, T., Gupta, S., Raj, R. S., Gandam, P. K. and Baadhe, R. R. (2021). 'Role of metagenomics in prospecting novel endoglucanases, accentuating functional metagenomics approach in second-generation biofuel production: a review'. *Biomass Conversion and Biorefinery*, pp. 1–28 (cited on p. 2).

Pan, S., Zhu, C., Zhao, X.-M. and Coelho, L. P. (2021). 'SemiBin: incorporating information from reference genomes with semi-supervised deep learning leads to better metagenomic assembled genomes (MAGs)'. *BioRxiv* (cited on p. 8).

Pareek, C. S., Smoczynski, R. and Tretyn, A. (2011). 'Sequencing technologies and genome sequencing'. *Journal of applied genetics*, 52(4), pp. 413–435 (cited on p. 4).

Paszke, A. et al. (2019). 'PyTorch: An Imperative Style, High-Performance Deep Learning Library'. In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox and R. Garnett. Curran Associates, Inc., pp. 8024–8035. ↗ (cited on p. 40).

Pedregosa, F. et al. (2011). 'Scikit-learn: Machine Learning in Python'. *Journal of Machine Learning Research*, 12, pp. 2825–2830 (cited on p. 62).

Pellow, D., Mizrahi, I. and Shamir, R. (2020). 'PlasClass improves plasmid sequence classification'. *PLOS Computational Biology*, 16(4), pp. 1–9. DOI: `10.1371/journal.pcbi.1007781` (cited on pp. 5, 9, 10, 30, 57–60, 65, 72–75, 90).

Pellow, D., Zorea, A., Probst, M., Furman, O., Segal, A., Mizrahi, I. and Shamir, R. (2021). 'SCAPP: an algorithm for improved plasmid assembly in metagenomes'. *Microbiome*, 9(1), pp. 1–12 (cited on p. 10).

Poinar, H. N. et al. (2006). 'Metagenomics to Paleogenomics: Large-Scale Sequencing of Mammoth DNA'. *Science*, 311(5759), pp. 392–394. DOI: 10.1126/science.1123360 (cited on p. 1).

Poličar, P. G., Stražar, M. and Zupan, B. (2019). 'openTSNE: a modular Python library for t-SNE dimensionality reduction and embedding'. *bioRxiv*. DOI: 10.1101/731877 (cited on pp. 57, 62).

Pu, Y., Gan, Z., Henao, R., Yuan, X., Li, C., Stevens, A. and Carin, L. (2016). 'Variational autoencoder for deep learning of images, labels and captions'. *Advances in neural information processing systems*, 29, pp. 2352–2360 (cited on p. 30).

Rahmah, N. and Sitanggang, I. S. (2016). 'Determination of Optimal Epsilon (Eps) Value on DBSCAN Algorithm to Clustering Data on Peatland Hotspots in Sumatra'. *IOP Conference Series: Earth and Environmental Science*, 31, p. 012012 (cited on p. 14).

RAPIDS Development Team (2018). *RAPIDS: Collection of Libraries for End to End GPU Data Science.* (cited on p. 50).

Rho, M., Tang, H. and Ye, Y. (2010). 'FragGeneScan: predicting genes in short and error-prone reads'. *Nucleic acids research*, 38(20), e191–e191 (cited on p. 12).

Rizk, G., Lavenier, D. and Chikhi, R. (2013). 'DSK: k-mer counting with very low memory usage'. *Bioinformatics*, 29(5), pp. 652–653 (cited on p. 13).

Robbins, S. J., Singleton, C. M., Chan, C. X. et al. (2019). 'A genomic view of the reef-building coral Porites lutea and its microbial symbionts'. *Nature Microbiology* (cited on pp. 27, 28).

Rodriguez-Beltran, J., Hernandez-Beltran, J. C. R., DelaFuente, J., Escudero, J. A., Fuentes-Hernandez, A., MacLean, R. C., Peña-Miller, R. and San Millan, A. (2018). 'Multicopy plasmids allow bacteria to escape from fitness trade-offs during evolutionary innovation'. *Nature Ecology & Evolution*, 2(5), pp. 873–881. DOI: 10.1038/s41559-018-0529-z (cited on p. 2).

Rousseeuw, P. J. (1987). 'Silhouettes: A graphical aid to the interpretation and validation of cluster analysis'. *Journal of Computational and Applied Mathematics*, 20, pp. 53–65. DOI: https://doi.org/10.1016/0377-0427(87)90125-7 (cited on p. 47).

Royer, G., Decousser, J., Branger, C., Dubois, M., Médigue, C., Denamur, E. and Vallenet, D. (2018). 'PlaScope: a targeted approach to assess the plasmidome from genome assemblies at the species level'. *Microbial genomics*, 4(9) (cited on pp. 9, 10).

Rozov, R., Brown Kav, A., Bogumil, D., Shterzer, N., Halperin, E., Mizrahi, I. and Shamir, R. (2016). 'Recycler: an algorithm for detecting plasmids from de novo assembly graphs'. *Bioinformatics*, 33(4), pp. 475–482. DOI: 10.1093/bioinformatics/btw651 (cited on p. 10).

Rozov, R., Brown Kav, A., Bogumil, D., Shterzer, N., Halperin, E., Mizrahi, I. and Shamir, R. (2017). 'Recycler: an algorithm for detecting plasmids from de novo assembly graphs'. *Bioinformatics*, 33(4), pp. 475–482 (cited on p. 9).

Ruan, J. and Li, H. (2020a). 'Fast and accurate long-read assembly with wtdbg2'. *Nature Methods*, 17(2), pp. 155–158. DOI: 10.1038/s41592-019-0669-3 (cited on pp. 36, 38).

Ruan, J. and Li, H. (2020b). 'Fast and accurate long-read assembly with wtdbg2'. *Nature methods*, 17(2), pp. 155–158 (cited on pp. 45, 52).

Schwengers, O., Barth, P., Falgenhauer, L., Hain, T., Chakraborty, T. and Goesmann, A. (2020). 'Platon: Identification and characterization of bacterial plasmid contigs in short-read draft assemblies exploiting protein sequence-based replicon distribution scores'. *Microbial genomics*, 6(10) (cited on pp. 9, 10, 57, 78).

Segata, N., Boernigen, D., Tickle, T. L., Morgan, X. C., Garrett, W. S. and Huttenhower, C. (2013). 'Computational meta'omics for microbial community studies'. *Molecular Systems Biology*, 9(1), p. 666. DOI: https://doi.org/10.1038/msb.2013.22 (cited on p. 1).

Sharon, I., Morowitz, M. J., Thomas, B. C. et al. (2013). 'Time series community genomics analysis reveals rapid shifts in bacterial species, strains, and phage during infant gut colonization'. *Genome Research*, 23(1), pp. 111–120 (cited on p. 19).

Shi, Y., Zhang, H., Tian, Z., Yang, M. and Zhang, Y. (2018). 'Characteristics of ARG-carrying plasmidome in the cultivable microbial community from wastewater treatment system under high oxytetracycline concentration'. *Applied Microbiology and Biotechnology*, 102(4), pp. 1847–1858. DOI: 10.1007/s00253-018-8738-6 (cited on p. 73).

Simpson, J. T. and Durbin, R. (2012). 'Efficient de novo assembly of large genomes using compressed data structures'. *Genome research*, 22(3), pp. 549–556 (cited on p. 5).

Smalla, K., Jechalke, S. and Top, E. M. (2015). 'Plasmid Detection, Characterization, and Ecology'. *Microbiology Spectrum*, 3(1). ↗ (cited on p. 2).

Stöcker, B. K., Köster, J. and Rahmann, S. (2016). 'SimLoRD: simulation of long read data'. *Bioinformatics*, 32(17), pp. 2704–2706 (cited on pp. 19, 35, 48, 58, 91).

Stoler, N. and Nekrutenko, A. (2021). 'Sequencing error profiles of Illumina sequencing instruments'. *NAR genomics and bioinformatics*, 3(1), lqab019 (cited on p. 4).

Strous, M., Kraft, B., Bisdorf, R. and Tegetmeyer, H. (2012). 'The Binning of Metagenomic Contigs for Microbial Physiology of Mixed Cultures'. *Frontiers in Microbiology*, 3, p. 410 (cited on pp. 8, 46).

Tham, C. Y., Tirado-Magallanes, R., Goh, Y., Fullwood, M. J., Koh, B. T., Wang, W., Ng, C. H., Chng, W. J., Thiery, A., Tenen, D. G. et al. (2020). 'NanoVar: accurate characterization of patients' genomic structural variants using low-depth nanopore sequencing'. *Genome biology*, 21(1), pp. 1–15 (cited on p. 5).

The Human Microbiome Project Consortium (2012). 'Structure, function and diversity of the healthy human microbiome'. *Nature*, 486, pp. 207–214 (cited on pp. 2, 4).

Thomas, C. M. and Nielsen, K. M. (2005). 'Mechanisms of, and Barriers to, Horizontal Gene Transfer between Bacteria'. *Nature Reviews Microbiology*, 3(9), pp. 711–721. DOI: 10.1038/nrmicro1234 (cited on p. 2).

Turnbaugh, P. J., Ley, R. E., Hamady, M., Fraser-Liggett, C. M., Knight, R. and Gordon, J. I. (2007). 'The human microbiome project'. *Nature*, 449(7164), pp. 804–810 (cited on p. 4).

Van Der Maaten, L. (2014). 'Accelerating t-SNE Using Tree-based Algorithms'. *J. Mach. Learn. Res.*, 15(1), pp. 3221–3245 (cited on pp. 13, 17, 18).

Van der Maaten, L. and Hinton, G. (2008). 'Visualizing data using t-SNE.' *Journal of machine learning research*, 9(11) (cited on pp. 9, 18).

Virgin, H. W. and Todd, J. A. (2011). 'Metagenomics and personalized medicine'. *Cell*, 147(1), pp. 44–56 (cited on p. 2).

Wang, Y., Leung, H. C., Yiu, S. and Chin, F. Y. (2012). 'MetaCluster 5.0: a two-round binning approach for metagenomic data for low-abundance species in a noisy sample'. *Bioinformatics*, 28(18), pp. i356–i362 (cited on p. 20).

Wang, Y., Wang, K., Lu, Y. Y. and Sun, F. (2017). 'Improving contig binning of metagenomic data using d2S oligonucleotide frequency dissimilarity'. *BMC Bioinformatics*, 18(1), p. 425 (cited on p. 20).

Wang, Z., Wang, Z., Lu, Y. Y. et al. (2019). 'SolidBin: improving metagenome binning with semi-supervised normalized cut'. *Bioinformatics*. btz253 (cited on pp. 8, 70).

Wenger, A. M., Peluso, P., Rowell, W. J., Chang, P.-C., Hall, R. J., Concepcion, G. T., Ebler, J., Fungtammasan, A., Kolesnikov, A., Olson, N. D. et al. (2019). 'Accurate circular consensus long-read sequencing improves variant detection and assembly of a human genome'. *Nature biotechnology*, 37(10), pp. 1155–1162 (cited on p. 52).

Wick, R. and Holt, K. (2019). 'Benchmarking of long-read assemblers for prokaryote whole genome sequencing [version 1; peer review: 4 approved]'. *F1000Research*, 8(2138). DOI: 10.12688/f1000research.21782.1 (cited on p. 61).

Wickramarachchi, A. (2021). *anuradhawick/seq2vec: release v1.0*. Version v1.0. DOI: 10.5281/zenodo.5515743 (cited on p. 50).

Wickramarachchi, A. and Lin, Y. (2021). 'GraphPlas: Refined Classification of Plasmid Sequences using Assembly Graphs'. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (cited on pp. i, 65).

Wickramarachchi, A. and Lin, Y. (2022a). 'Binning long reads in metagenomics datasets using composition and coverage information'. *Algorithms for Molecular Biology*, 17(1), pp. 1–15 (cited on pp. i, 29).

Wickramarachchi, A. and Lin, Y. (2022b). 'Metagenomics Binning of Long Reads Using Read-Overlap Graphs'. In: *RECOMB International Workshop on Comparative Genomics*. Springer, pp. 260–278 (cited on pp. i, 44).

Wickramarachchi, A. and Lin, Y. (2021a). 'LRBinner: Binning Long Reads in Metagenomics Datasets'. In: *21st International Workshop on Algorithms in Bioinformatics (WABI 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik (cited on pp. i, 29, 48, 50, 55).

Wickramarachchi, A., Mallawaarachchi, V., Pu, L. and Lin, Y. (2021b). 'PlasLR Enables Adaptation of Plasmid Prediction for Error-Prone Long Reads'. *bioRxiv* (cited on pp. i, 54).

Wickramarachchi, A., Mallawaarachchi, V., Rajan, V. and Lin, Y. (2020a). 'MetaBCC-LR: Metagenomics Binning by Coverage and Composition for Long Leads'. *Bioinformatics*, 36(Supplement_1), pp. i3–i11 (cited on pp. i, 9, 11, 30, 35, 36, 46, 48, 50, 54, 55, 70).

Wong, K., Finan, T. M. and Golding, B. G. (2002). 'Dinucleotide compositional analysis of Sinorhizobium meliloti using the genome signature: distinguishing chromosomes and plasmids'. *Functional & Integrative Genomics*, 2(6), pp. 274–281. DOI: 10.1007/s10142-002-0068-0 (cited on p. 55).

Wood, D. E., Lu, J. and Langmead, B. (2019). 'Improved metagenomic analysis with Kraken 2'. *Genome biology*, 20(1), pp. 1–13 (cited on pp. 5, 7, 12, 57, 63, 78, 80, 103).

Wood, D. E. and Salzberg, S. L. (2014). 'Kraken: ultrafast metagenomic sequence classification using exact alignments'. *Genome biology*, 15(3), pp. 1–12 (cited on p. 7).

Wu, Y.-W., Simmons, B. A. and Singer, S. W. (2015). 'MaxBin 2.0: an automated binning algorithm to recover genomes from multiple metagenomic datasets'. *Bioinformatics*, 32(4), pp. 605–607 (cited on pp. 8, 12, 15, 16, 30, 46, 55, 68–70).

Wu, Y.-W., Tang, Y.-H., Tringe, S. G. et al. (2014). 'MaxBin: an automated binning method to recover individual genomes from metagenomes using an expectation-maximization algorithm'. *Microbiome*, 2(1), p. 26 (cited on pp. 5, 12, 19, 48, 92).

Wu, Y.-W. and Ye, Y. (2011). 'A Novel Abundance-Based Algorithm for Binning Metagenomic Sequences Using l-tuples'. *Journal of Computational Biology*, 18(3), pp. 523–534 (cited on p. 8).

Xu, K., Hu, W., Leskovec, J. and Jegelka, S. (2018). 'How powerful are graph neural networks?' *arXiv preprint arXiv:1810.00826* (cited on p. 47).

Xue, H., Mallawaarachchi, V., Zhang, Y., Rajan, V. and Lin, Y. (2021). 'RepBin: Constraint-based Graph Representation Learning for Metagenomic Binning'. *arXiv preprint arXiv:2112.11696* (cited on p. 8).

Ying, H., Cooke, I., Sprungala, S., Wang, W., Hayward, D. C., Tang, Y., Huttley, G., Ball, E. E., Forêt, S. and Miller, D. J. (2018). 'Comparative genomics reveals the distinct evolutionary trajectories of the robust and complex coral lineages'. *Genome biology*, 19(1), p. 175 (cited on p. 27).

Yue, Y., Huang, H., Qi, Z., Dou, H.-M., Liu, X.-Y., Han, T.-F., Chen, Y., Song, X.-J., Zhang, Y.-H. and Tu, J. (2020). 'Evaluating metagenomics tools for genome binning with real metagenomic datasets and CAMI datasets'. *BMC bioinformatics*, 21(1), pp. 1–15 (cited on p. 4).

Zhou, F., Olman, V. and Xu, Y. (2008). 'Barcodes for genomes and applications'. *BMC Bioinformatics*, 9(1), p. 546. DOI: 10.1186/1471-2105-9-546 (cited on p. 55).

Zhou, F. and Xu, Y. (2010a). 'cBar: a computer program to distinguish plasmid-derived from chromosome-derived sequence fragments in metagenomics data'. *Bioinformatics*, 26(16), pp. 2051–2052 (cited on p. 9).

Zhou, F. and Xu, Y. (2010b). 'cBar: a computer program to distinguish plasmid-derived from chromosome-derived sequence fragments in metagenomics data'. *Bioinformatics*, 26(16), pp. 2051–2052. DOI: 10.1093/bioinformatics/btq299 (cited on pp. 9, 10, 55).

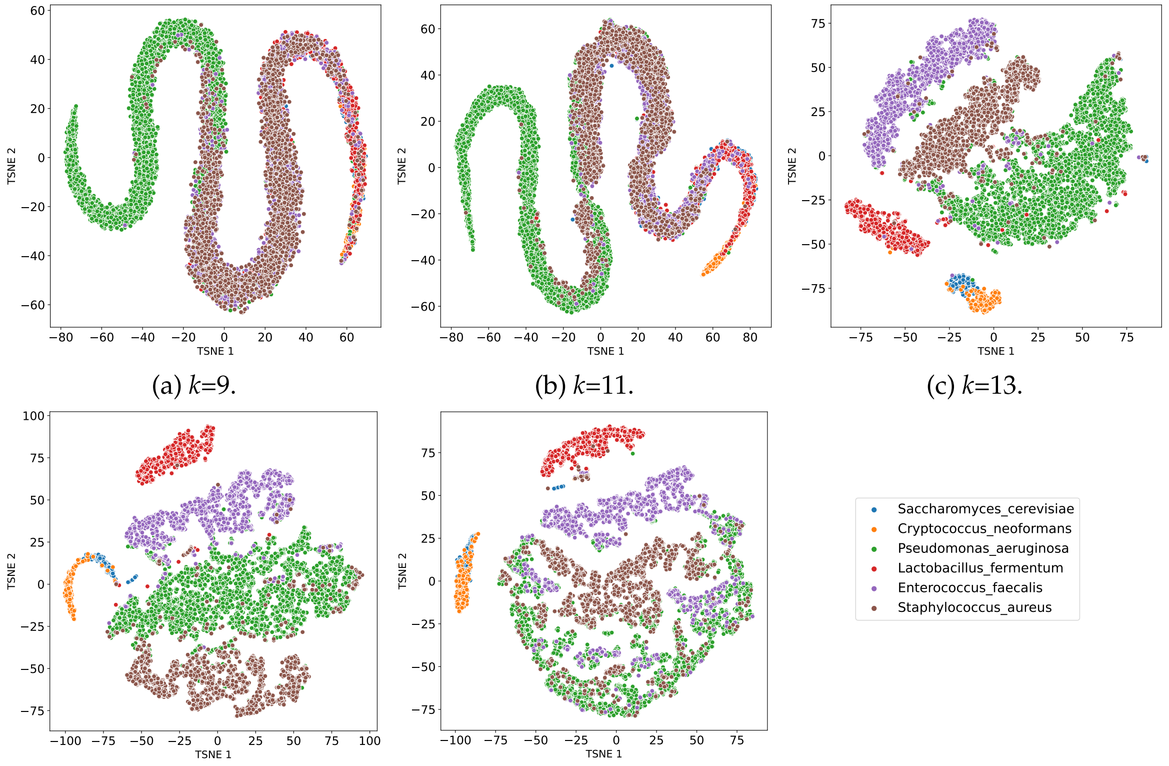
Zhu, W., Lomsadze, A. and Borodovsky, M. (2010). 'Ab initio gene identification in metagenomic sequences'. *Nucleic acids research*, 38(12), e132–e132 (cited on p. 12).

Zhu, X. and Ghahramani, Z. (2002). *Learning from Labeled and Unlabeled Data with Label Propagation*. Tech. rep. School of Computer Science, Carnegie Mellon University (cited on pp. 67, 68).

Appendix A

k-mer Coverage Histograms and *k* size

This section discusses the impact of varying size of *k* on *k*-mer coverage histograms. Use of smaller *k* (less than 7 base pairs) results in *k*-mers that are too common across genomes hence does not support our aim of computing coverage. This is because, we rely on rare genomic *k*-mers that does not occur across different species to estimate coverage. For example, PlasFlow (Krawczyk et al., 2018) and PlasClass (Pellow, Mizrahi et al., 2020) utilize 7-mers in the detection of plasmids, which implies that 7-mers are expected to share common patterns within plasmids and chromosomes while discriminating between plasmids and chromosome.



(a) *k*=9. (b) *k*=11. (c) *k*=13.

(d) *k*=15. (e) *k*=17.

Figure A.1: TSNE plots for varying *k*-mer coverage histogram under varying *k*-mer lengths.

Therefore, we start our analysis at 9-mers and go up to 17-mers to evaluate the impact of *k*-mer length on coverage histograms. Figure A.1 illustrates the TSNE embeddings for a sample of 10,000 *k*-mer coverage histograms of the dataset **2Y4B** from Chapter 3. It is clearly evident that *k*=9 and *k*=11 perform poorly. This is because, these shorter *k*-mers are shared across multiple species in the sample distorting the *k*-mer coverage. However, *k*=13 and *k*=15 demonstrates good coverage based separation of species. However, *k*=17 has poor performance due to *k*-mers being overly unique, thus, failing to capture genomic *k*-mers of species across different reads. This is mainly because, longer the *k*-mer, the chance of that being erroneous is higher. Hence, *k*-mer size 13 and 15 could be considered as optimum values for coverage estimation.

Appendix B

Datasets Information

B.1 Simulated Long Reads Datasets

This section tabulates information of the simulated datasets presented in this thesis. All the simulations have been conducted using SimLoRD (Stöcker et al., 2016).

Table B.1: Information about the simulated PacBio datasets

Dataset	Read type	Species present	Genome size (Mb)	Coverage	Abundance	Dataset size (GB)	Average read length (kb)
Zymo-1Y2B	PacBio	*S. cerevisiae*	13.163	15x	4.4%	4.2	8.298
		P. aeruginosa	6.792	550x	82.9%		
		L. fermentum	1.905	300x	12.7%		
Zymo-1Y3B	PacBio	*S. cerevisiae*	13.163	15x	3.4%	5.45	8.297
		P. aeruginosa	6.792	550x	64.6%		
		L. fermentum	1.905	300x	9.9%		
		E. faecalis	2.845	450x	22.1%		
Zymo-2Y2B	PacBio	*S. cerevisiae*	13.163	15x	4.2%	4.35	8.299
		C. neoformans	19.325	10x	4.1%		
		P. aeruginosa	6.792	550x	79.5%		
		L. fermentum	1.905	300x	12.2%		
Zymo-2Y3B	PacBio	*S. cerevisiae*	13.163	15x	3.3%	5.65	8.298
		C. neoformans	19.325	10x	3.2%		
		P. aeruginosa	6.792	550x	62.5%		
		L. fermentum	1.905	300x	9.6%		
		E. faecalis	2.845	450x	21.4%		
Zymo-2Y4B	PacBio	*S. cerevisiae*	13.163	15x	2.6%	7.15	8.294
		C. neoformans	19.325	10x	2.5%		
		P. aeruginosa	6.792	550x	49.0%		
		L. fermentum	1.905	300x	7.5%		
		E. faecalis	2.845	450x	16.8%		
		S. aureus	2.730	600x	21.5%		
Sharon	PacBio	*E. faecalis*	3.069	2370x	72.6%	9.8	8.281
		S. aureus	2.913	677x	19.7%		
		P. rhinitidis	2.562	148x	3.8%		
		C. avidum	2.562	136x	3.5%		
		S. epidermidis	2.536	17x	0.4%		

Table B.2: Information about the 100-genomes dataset. Relative abundance ratios were used according to the simMC+ dataset (Wu, Tang et al., 2014)

NCBI Genbank ID	Species present	Relative abundance ratios
256653503	*Acetobacter pasteurianus*	14.5%
330827700	*Aeromonas veronii*	14.5%
398314590	*Amycolatopsis mediterranei*	11.6%
308175814	*Arthrobacter arilaitensis*	7.0%
158421624	*Azorhizobium caulinodans*	4.7%
217957581	*Bacillus cereus*	4.3%
296500838	*Bacillus thuringiensis*	1.2%
42521650	*Bdellovibrio bacteriovorus*	0.6%
119025018	*Bifidobacterium adolescentis*	0.6%
295793053	*Bifidobacterium animalis*	0.6%
343385146	*Brachyspira intermedia*	0.5%
15791399	*Campylobacter jejuni*	0.5%
71082709	*Candidatus Pelagibacter ubique*	0.5%
194246403	*Candidatus Phytoplasma mali*	0.5%
256370581	*Candidatus Sulcia muelleri*	0.5%
297749010	*Chlamydia trachomatis*	0.5%
334694771	*Chlamydophila psittaci*	0.5%
325507407	*Clostridium acetobutylicum*	0.5%
331268188	*Clostridium botulinum*	0.5%
28209834	*Clostridium tetani*	0.5%
125972525	*Clostridium thermocellum*	0.5%
376247367	*Corynebacterium diphtheriae*	0.5%
385806437	*Corynebacterium pseudotuberculosis*	0.5%
334695745	*Corynebacterium ulcerans*	0.5%
284928601	*Cyanobacterium UCYN*	0.5%
307149945	*Cyanothece sp*	0.5%
46562128	*Desulfovibrio vulgaris*	0.5%
58616727	*Ehrlichia ruminantium*	0.5%
378937014	*Enterococcus faecium*	0.5%
336065242	*Erysipelothrix rhusiopathiae*	0.5%
209917191	*Escherichia coli*	0.5%
385805051	*Fervidicoccus fontis*	0.5%
302325342	*Fibrobacter succinogenes*	0.5%

Continued to next page

NCBI Genbank ID	Species present	Relative abundance ratios
347534971	*Flavobacterium branchiophilum*	0.5%
118496615	*Francisella novicida*	0.5%
156501369	*Francisella tularensis*	0.5%
19703352	*Fusobacterium nucleatum*	0.5%
333392846	*Gardnerella vaginalis*	0.5%
322433659	*Granulicella tundricola*	0.5%
148826757	*Haemophilus influenzae*	0.5%
301154649	*Haemophilus parainfluenzae*	0.5%
170717206	*Haemophilus somnus*	0.5%
12057215	*Halobacterium sp*	0.5%
261854630	*Halothiobacillus neapolitanus*	0.5%
261838873	*Helicobacter pylori*	0.5%
338736863	*Hyphomicrobium sp*	0.5%
385808586	*Ignavibacterium album*	0.5%
375256816	*Klebsiella oxytoca*	0.5%
332290650	*Krokinobacter sp*	0.5%
116332681	*Lactobacillus brevis*	0.5%
327384027	*Lactobacillus casei*	0.5%
104773257	*Lactobacillus delbrueckii*	0.5%
94986445	*Lawsonia intracellularis*	0.5%
296105497	*Legionella pneumophila*	0.5%
330833867	*Metallosphaera cuprina*	0.5%
124484829	*Methanocorpusculum labreanum*	0.5%
19918815	*Methanosarcina acetivorans*	0.5%
73667559	*Methanosarcina barkeri*	0.5%
239916571	*Micrococcus luteus*	0.5%
356592064	*Mycobacterium bovis*	0.5%
108796981	*Mycobacterium sp*	0.5%
330723203	*Mycoplasma hyorhinis*	0.5%
308388224	*Neisseria meningitidis*	0.5%
300112745	*Nitrosococcus watsonii*	0.5%
325980881	*Nitrosomonas sp*	0.5%
54021964	*Nocardia farcinica*	0.5%
325278757	*Odoribacter splanchnicus*	0.5%
	Continued to next page	

NCBI Genbank ID	Species present	Relative abundance ratios
386720569	*Paenibacillus mucilaginosus*	0.5%
261403876	*Paenibacillus sp*	0.5%
54307237	*Photobacterium profundum*	0.5%
126695337	*Prochlorococcus marinus*	0.5%
347537839	*Pseudogulbenkiania sp*	0.5%
313496345	*Pseudomonas putida*	0.5%
116249766	*Rhizobium leguminosarum*	0.5%
111017022	*Rhodococcus jostii*	0.5%
380760311	*Rickettsia prowazekii*	0.5%
378722019	*Rickettsia rickettsii*	0.5%
374318767	*Rickettsia slovaca*	0.5%
99079841	*Ruegeria sp*	0.5%
194447306	*Salmonella enterica*	0.5%
269118642	*Sebaldella termitidis*	0.5%
114045513	*Shewanella sp*	0.5%
30061571	*Shigella flexneri*	0.5%
85057978	*Sodalis glossinidius*	0.5%
311222926	*Staphylococcus aureus*	0.5%
182682970	*Streptococcus pneumoniae*	0.5%
28894912	*Streptococcus pyogenes*	0.5%
354984442	*Streptococcus suis*	0.5%
116626972	*Streptococcus thermophilus*	0.5%
290954631	*Streptomyces scabiei*	0.5%
51891138	*Symbiobacterium thermophilum*	0.5%
320114857	*Thermoanaerobacter brockii*	0.5%
307723218	*Thermoanaerobacter sp*	0.5%
242397997	*Thermococcus sibiricus*	0.5%
239819985	*Variovorax paradoxus*	0.5%
323436265	*Weeksella virosa*	0.5%
225629872	*Wolbachia sp*	0.5%
154243958	*Xanthobacter autotrophicus*	0.5%
162418099	*Yersinia pestis*	0.5%

Table B.3: Information of simulated datasets.

Dataset	Number of Reads	Total Size	Species	Coverage
Sim-8	432,333	3.5Gb	Acetobacter pasteurianus	25
			Bacillus cereus	50
			Chlamydophila psittaci	80
			Escherichia coli	125
			Haemophilus parainfluenzae	350
			Lactobacillus casei	200
			Thermococcus sibiricus	150
			Streptomyces scabiei	100
Sim-20	666,735	5.3Gb	Amycolatopsis mediterranei	25
			Arthrobacter arilaitensis	65
			Brachyspira intermedia	20
			Corynebacterium ulcerans	40
			Erysipelothrix rhusiopathiae	55
			Enterococcus faecium	50
			Mycobacterium bovis	80
			Photobacterium profundum	85
			Streptococcus pyogenes	100
			Xanthobacter autotrophicus	150
			Rhizobium leguminosarum	100
			Francisella novicida	150
			Candidatus Pelagibacter ubique	67
			Halobacterium sp	65
			Lactobacillus delbrueckii	60
			Paenibacillus mucilaginosus	90
			Rickettsia prowazekii	100
			Thermoanaerobacter brockii	110
			Yersinia pestis	105
			Nitrosococcus watsonii	95

Table B.4: Information of simulated dataset containing 50 species.

Dataset	Number of Reads	Total Size	Species	Coverage
Sim-50	1,119,439	9.5GB	Azorhizobium caulinodans	25
			Bacillus cereus	35
			Bdellovibrio bacteriovorus	21
			Bifidobacterium adolescentis	44
			Bifidobacterium animalis	31
			Campylobacter jejuni	11
			Clostridium tetani	36
			Clostridium thermocellum	31
			Corynebacterium diphtheriae	42
			Corynebacterium ulcerans	33
			Ehrlichia ruminantium	26
			Enterococcus faecium	24
			Erysipelothrix rhusiopathiae	44
			Escherichia coli	20
			Fervidicoccus fontis	49
			Francisella novicida	42
			Francisella tularensis	49
			Fusobacterium nucleatum	39
			Haemophilus influenzae	12
			Haemophilus parainfluenzae	11
			Haemophilus somnus	44
			Helicobacter pylori	47
			Hyphomicrobium sp	44
			Lawsonia intracellularis	46
			Metallosphaera cuprina	33
			Methanosarcina barkeri	44
			Micrococcus luteus	46
			Mycobacterium bovis	42
			Mycoplasma gallisepticum	29
			Neisseria meningitidis	38
			Nitrosococcus watsonii	42
			Paenibacillus mucilaginosus	14
			Paenibacillus sp	31
			Photobacterium profundum	45
			Pseudogulbenkiania sp	25
			Pseudomonas putida	10
			Rhizobium leguminosarum	20
			Rickettsia prowazekii	38
			Rickettsia rickettsii	100
			Ruegeria sp	200
			Shewanella sp	90
			Sodalis glossinidius	120
			Staphylococcus aureus	220
			Streptococcus pyogenes	110
			Streptococcus suis	100
			Streptomyces scabiei	110
			Symbiobacterium thermophilum	250
			Thermoanaerobacter sp	220
			Thermococcus sibiricus	210
			Variovorax paradoxus	100

Table B.5: Information about the long reads simulated datasets containing plasmids.

Dataset	Read type	Tax ID	Plasmid/Chromosome	Accession Number	Length (bp)	Plasmid/Chromosome copy number	Number of reads
Sim-2C5P	PacBio	311402	plasmid	NC_011981	631775	2	3410
		311402	Plasmid	NC_011982	258824	1	690
		311402	Plasmid	NC_011984	211620	3	1710
		311402	Plasmid	NC_011986	78730	1	210
		311402	Plasmid	NC_011991	130435	2	700
		311402	Chromosome	NC_011988	1283187	1	3460
		311402	Chromosome	NC_011989	3726375	1	10070
Sim-4C11P	PacBio	243230	plasmid	NC_000958	177466	1	1923
		243230	Plasmid	NC_000959	45704	1	495
		243230	Plasmid	NZ_CP015083	203183	1	2201
		243230	Plasmid	NZ_CP015084	61707	3	2005
		243230	Chromosome	NZ_CP015081	2646742	1	28680
		243230	Chromosome	NZ_CP015082	433133	1	4693
		358	Plasmid	NC_002147	206479	2	3876
		358	Plasmid	NC_002377	194140	1	1822
		358	Plasmid	NC_006277	44420	2	834
		358	Plasmid	NC_010929	244978	2	4600
		358	Plasmid	NC_019555	176574	3	4974
		358	Plasmid	NZ_CP011248	544752	1	5114
		358	Plasmid	NZ_CP011249	194264	4	7296
		358	Chromosome	NZ_CP014259	2497934	1	23456
		358	Chromosome	NZ_CP014260	2983661	1	28018
Sim-10C16P	PacBio	243230	Plasmid	NC_000958	177466	1	3759
		243230	Plasmid	NC_000959	45704	1	968
		243230	Plasmid	NZ_CP015083	203183	2	8608
		243230	Plasmid	NZ_CP015084	61707	5	6536
		243230	Chromosome	NZ_CP015081	2646742	1	56071
		243230	Chromosome	NZ_CP015082	433133	1	9175
		1492	Plasmid	NC_012760	8060	1	79
		1492	Plasmid	NZ_CP013354	8060	2	159
		1492	Plasmid	NZ_CP014706	8061	4	318
		1492	Chromosome	NZ_CP014704	3794139	1	37535
		1492	Chromosome	NZ_CP014705	795002	1	7865
		529	Plasmid	NC_010917	4227	3	11
		529	Plasmid	NZ_CP008817	155838	2	294
		529	Plasmid	NZ_CP008818	106739	1	100
		529	Chromosome	NZ_CP008819	1930134	1	1824
		529	Chromosome	NZ_CP008820	2708454	1	2560
		176299	Plasmid	NC_003064	542868	3	5192
		176299	Plasmid	NC_003065	214233	2	1366
		176299	Plasmid	NC_004972	8288	2	52
		176299	Chromosome	NC_003062	2841580	1	9059
		176299	Chromosome	NC_003063	2075577	1	6617
		311403	Plasmid	NC_011987	388169	1	2219
		311403	Plasmid	NC_011990	184668	1	1055
		311403	Plasmid	NC_011994	44420	2	507
		311403	Chromosome	NC_011983	2650913	1	15156
		311403	Chromosome	NC_011985	4005130	1	22898

B.2 Publicly Available Datasets

This section tabulates information of the real datasets presented in this thesis. All the presented datasets are publicly available, and they can be obtained from NCBI database at NIH freely using respective accession numbers.

Table B.6: Information about the publicly available datasets. †The coverage values for the **Zymo-All** dataset were obtained from (Kolmogorov, Rayko et al., 2019). *The coverage values for the **ASM** datasets were obtained from the NCBI SRA taxonomy analysis.

Dataset	Read type	Species present	Genome size (Mb)	Coverage	Abundance	Dataset size (GB)	Average read length (kb)
ZymoEVEN†	ONT	*P. aeruginosa*	6.792	155x	9.7%	14.24	4.079
		E. coli	4.875	220x	9.9%		
		S. enterica	4.760	227x	10.0%		
		L. fermentum	1.905	528x	9.3%		
		E. faecalis	2.845	464x	12.2%		
		S. aureus	2.730	445x	11.2%		
		L. monocytogenes	2.992	525x	14.5%		
		B. subtilis	4.045	516x	19.3%		
		S. cerevisiae	13.163	17x	2.1%		
		C. neoformans	19.325	10x	1.8%		
ASM-0*	PacBio	*P. aeruginosa*	6.631	5.8x	36.0%	0.10	10.601
		A. pittii	3.917	5.9x	21.6%		
		S. epidermidis	2.535	6.1x	14.5%		
		C. acnes	2.524	6.1x	14.4%		
		S. mitis	2.177	6.6x	13.5%		
ASM-5*	PacBio	*P. aeruginosa*	6.631	5.4x	36.1%	0.10	10.313
		A. pittii	3.917	5.5x	21.7%		
		S. epidermidis	2.535	5.6x	14.3%		
		C. acnes	2.524	5.7x	14.5%		
		S. mitis	2.177	6.1x	13.4%		
ASM-10*	PacBio	*P. aeruginosa*	6.631	5.1x	36.0%	0.10	10.322
		A. pittii	3.917	5.2x	21.7%		
		S. epidermidis	2.535	5.3x	14.3%		
		C. acnes	2.524	5.4x	14.5%		
		S. mitis	2.177	5.8x	13.4%		
ASM-15*	PacBio	*P. aeruginosa*	6.631	4.8x	35.9%	0.10	10.330
		A. pittii	3.917	4.9x	21.7%		
		S. epidermidis	2.535	5.0x	14.3%		
		C. acnes	2.524	5.1x	14.5%		
		S. mitis	2.177	5.5x	13.5%		
Coral+Symbio	PacBio	*P. lutea*	561.222	20x	47.2%	27.65	8.865
		Cladocopium C15	628.606	20x	52.8%		

Table B.7: Information of publicly available real datasets SRR9202034 , SRX9569057, SRX9569058 and SRX9569059.

Dataset	Number of Reads	Total Size	Species	Abundance
SRR9202034	2,358,257	19Gb	Acinetobacter baumannii	0.18%
			Bacillus pacificus	1.80%
			Bacteroides vulgatus	0.02%
			Bifidobacterium adolescentis	0.02%
			Clostridium beijerinckii	1.80%
			Cutibacterium acnes	0.18%
			Deinococcus radiodurans	0.02%
			Enterococcus faecalis	0.02%
			Escherichia coli	18.0%
			Helicobacter pylori	0.18%
			Lactobacillus gasseri	0.18%
			Neisseria meningitidis	0.18%
			Porphyromonas gingivalis	18.0%
			Pseudomonas aeruginosa	1.80%
			Rhodobacter sphaeroides	18.0%
			Schaalia odontolytica	0.02%
			Staphylococcus aureus	1.80%
			Staphylococcus epidermidis	18.0%
			Streptococcus agalactiae	1.80%
			Streptococcus mutans	18.0%
SRX9569057 SRX9569058 SRX9569059	1,978,852 2,770,833 2,480,208	17Gb 25Gb 20Gb	Faecalibacterium prausnitzii	14.82%
			Veillonella rogosae	20.01%
			Roseburia hominis	12.47%
			Bacteroides fragilis	8.36%
			Prevotella corporis	6.28%
			Bifidobacterium adolescentis	8.86%
			Fusobacterium nucleatum	7.56%
			Lactobacillus fermentum	9.71%
			Clostridioides difficile	1.10%
			Akkermansia muciniphila	1.62%
			Methanobrevibacter smithii	0.17%
			Salmonella enterica	0.0065%
			Enterococcus faecalis	0.0011%
			Clostridium perfringens	0.00009%
			Escherichia coli (JM109)	1.83%
			Escherichia coli (B-3008)	1.82%
			Escherichia coli (B-2207)	1.65%
			Escherichia coli (B-766)	1.66%
			Escherichia coli (B-1109)	1.77%
			Candida albicans	0.16%
			Saccharomyces cerevisiae	0.16%

B.3 Simulated Short Read datasets

This section tabulates details about the simulated datasets for short read datasets including the species, chromosomes, plasmids, genome lengths, coverage values, abundance and copy numbers. This data has been simulated using InSilicoSeq (Gourlé et al., 2018).

Table B.8: Information on the short read simulated datasets.

Dataset	Accession number	Type	Length (bp)	Simulation Coverage (×)	Abundance (%)	Copy number
Sim-2C5P	NZ_CP015287.1	chromosome	3188516	10	68.57	1
	NZ_CP015288.1	chromosome	942922	10	20.28	1
	NZ_CP015289.1	plasmid	152958	350	3.29	35
	NZ_CP015290.1	plasmid	124313	100	2.67	10
	NZ_CP015291.1	plasmid	114172	100	2.46	10
	NZ_CP015292.1	plasmid	105299	200	2.26	20
	NZ_CP015293.1	plasmid	21883	200	0.47	20
Sim-2C9P	NZ_CP005083.1	chromosome	4249857	10	68.23	1
	NZ_CP005084.1	chromosome	989120	10	15.88	1
	NZ_CP005085.1	plasmid	520614	150	8.36	15
	NZ_CP005086.1	plasmid	195308	300	3.14	30
	NZ_CP005087.1	plasmid	87635	100	1.41	10
	NZ_CP005088.1	plasmid	75938	150	1.22	15
	NZ_CP005090.1	plasmid	34300	100	0.55	10
	NZ_CP005091.1	plasmid	9585	150	0.15	15
	NZ_CP005092.1	plasmid	7223	150	0.12	15
	NZ_CP005093.1	plasmid	5391	250	0.09	25
	NZ_CP005089.1	plasmid	53908	100	0.87	10

Dataset	Accession number	Type	Length (bp)	Simulation Coverage (×)	Abundance (%)	Copy number
Sim-10C25P	NZ_CP015287.1	chromosome	3188516	10	12.51	1
	NZ_CP015288.1	chromosome	942922	10	3.70	1
	NZ_CP015289.1	plasmid	152958	100	0.60	10
	NZ_CP015290.1	plasmid	124313	100	0.49	10
	NZ_CP015291.1	plasmid	114172	100	0.45	10
	NZ_CP015292.1	plasmid	105299	150	0.41	15
	NZ_CP015293.1	plasmid	21883	200	0.09	20
	NZ_CP022745.1	chromosome	3058963	10	12.00	1
	NZ_CP022746.1	chromosome	711561	10	2.79	1
	NZ_CP022747.1	plasmid	270473	100	1.06	10
	NZ_CP022748.1	plasmid	247521	250	0.97	25
	NZ_CP022749.1	plasmid	171560	100	0.67	10
	NZ_CP022750.1	plasmid	79937	300	0.31	30
	NZ_CP022751.1	plasmid	62622	100	0.25	10
	NZ_CP034183.1	chromosome	2784072	10	10.92	1
	NZ_CP034184.1	chromosome	515055	10	2.02	1
	NZ_CP034185.1	plasmid	427730	100	1.68	10
	NZ_CP034186.1	plasmid	406291	100	1.59	10
	NZ_CP034187.1	plasmid	164279	100	0.64	10
	NZ_CP034188.1	plasmid	63735	150	0.25	15
	NZ_CP034189.1	plasmid	36543	200	0.14	20
	NZ_CP014595.1	chromosome	2902170	10	11.38	1
	NZ_CP014596.1	chromosome	1472180	10	5.77	1
	NZ_CP014597.1	plasmid	315806	150	1.24	15
	NZ_CP014598.1	plasmid	274486	100	1.08	10
	NZ_CP014599.1	plasmid	280527	100	1.10	10
	NZ_CP014600.1	plasmid	222528	150	0.87	15
	NZ_CP014601.1	plasmid	54364	100	0.21	10
	NC_011989.1	chromosome	3726375	10	14.62	1
	NC_011988.1	chromosome	1283187	10	5.03	1
	NC_011986.1	plasmid	78730	100	0.31	10
	NC_011991.1	plasmid	130435	150	0.51	15
	NC_011984.1	plasmid	211620	200	0.83	20
	NC_011981.1	plasmid	631775	100	2.48	10
	NC_011982.1	plasmid	258824	100	1.02	10

Dataset	Accession number	Type	Length (bp)	Simulation Coverage (×)	Abundance (%)	Copy number
Sim-14C38P	NZ_CP015287.1	chromosome	3188516	10	8.31	1
	NZ_CP015288.1	chromosome	942922	10	2.46	1
	NZ_CP015289.1	plasmid	152958	150	0.40	15
	NZ_CP015290.1	plasmid	124313	150	0.32	15
	NZ_CP015291.1	plasmid	114172	100	0.30	10
	NZ_CP015292.1	plasmid	105299	150	0.27	15
	NZ_CP015293.1	plasmid	21883	150	0.06	15
	NZ_CP029352.1	chromosome	1351902	20	3.52	1
	NZ_CP029353.1	chromosome	2538983	20	6.62	1
	NZ_CP029354.1	chromosome	605262	20	1.58	1
	NZ_CP029355.1	chromosome	77812	20	0.20	1
	NZ_CP029356.1	plasmid	882608	200	2.30	10
	NZ_CP029357.1	plasmid	672413	200	1.75	10
	NZ_CP029358.1	plasmid	405625	400	1.06	20
	NZ_CP029359.1	plasmid	186821	300	0.49	15
	NZ_CP029360.1	plasmid	54887	300	0.14	15
	NZ_CP014505.1	chromosome	3260450	50	8.50	1
	NZ_CP014508.1	plasmid	286935	750	0.75	15
	NZ_CP014506.1	chromosome	1696029	50	4.42	1
	NZ_CP014509.1	plasmid	62715	500	0.16	10
	NZ_CP014507.1	chromosome	1495500	50	3.90	1
	NZ_CP014510.1	plasmid	52585	500	0.14	10
	NZ_CP014511.1	plasmid	20973	500	0.05	10
	NZ_CP014512.1	plasmid	6086	750	0.02	15
	NC_017486.1	chromosome	2399458	20	6.26	1
	NC_017483.1	plasmid	44098	200	0.11	10
	NC_017487.1	plasmid	35934	300	0.09	15
	NC_017484.1	plasmid	31442	300	0.08	15
	NC_017485.1	plasmid	5543	400	0.01	20
	NC_017488.1	plasmid	2262	300	0.01	15
	NZ_CP005083.1	chromosome	4249857	5	11.08	1
	NZ_CP005084.1	chromosome	989120	5	2.58	1
	NZ_CP005085.1	plasmid	520614	50	1.36	10
	NZ_CP005086.1	plasmid	195308	75	0.51	15
	NZ_CP005087.1	plasmid	87635	100	0.23	20
	NZ_CP005088.1	plasmid	75938	50	0.20	10
	NZ_CP005090.1	plasmid	34300	150	0.09	30
	NZ_CP005091.1	plasmid	9585	75	0.02	15
	NZ_CP005092.1	plasmid	7223	75	0.02	15
	NZ_CP005093.1	plasmid	5391	50	0.01	10
	NZ_CP005089.1	plasmid	53908	100	0.14	20
	NZ_CP028797.1	chromosome	5360164	3	13.97	1
	NZ_CP028794.1	plasmid	54847	60	0.14	20
	NZ_CP028795.1	plasmid	5596	60	0.01	20
	NZ_CP028796.1	plasmid	112467	30	0.29	10
	NZ_CP024528.1	chromosome	5297108	30	13.81	1
	NZ_CP024529.1	plasmid	221606	600	0.58	20
	NZ_CP024530.1	plasmid	147932	900	0.39	30
	NZ_CP024531.1	plasmid	89345	450	0.23	15
	NZ_CP024532.1	plasmid	4660	900	0.01	30
	NZ_CP024533.1	plasmid	4510	600	0.01	20
	NZ_CP024534.1	plasmid	3825	450	0.01	15

Appendix C

Evaluation Against Reference Based Binning

In this section we perform reference based binning of datasets from Chapter 5 using the popular reference based binning tool Kraken2 (Wood, Lu et al., 2019). Kraken2 perform taxonomic annotation of sequences at different taxonomic levels using minimizers and a minimizer index built using known genomes. In our evaluation we use the kraken2 database **MiniKraken2_v1_8GB** from `https://ccb.jhu.edu/software/kraken2/downloads.shtml`. Note that this database is built from the refseq bacteria, archaea, and viral libraries making it a strong candidate to perform metagenomics binning.

We evaluated kraken2 performance with two setups; (1) the default mode and (2) high confidence mode. Kraken2 defines confidence as the C/Q, where C is the number of k-mers mapped to the assigned label, and Q is the number of k-mers in the sequence that are also observed in the database. By default, kraken2 reports all labels that have positive confidence values. In our high confidence mode, we slightly harden the confidence by specifying 0.1 for `--confidence` parameters of kraken2.

C.1 Binning performance

Table C.1: Performance of Kraken2 reference based binning tool with default parameters

Dataset	No. of species	Bins Detected	Precision	Recall	F1-Score
Sim-8	8	525	99.99%	89.18%	94.28%
Sim-20	20	428	100.00%	84.83%	91.79%
Sim-50	50	1243	99.79%	91.16%	95.28%
Sim-100	100	1550	99.21%	92.80%	95.90%
ZymoEVEN	10	1477	98.53%	81.37%	89.13%
SRR9202034	20	602	99.96%	90.66%	95.08%
SRX9569057	21	1134	94.91%	70.09%	80.63%

It is evident from Table C.1 that under default operating confidence, kraken2 over-estimates the number of bins, thus reducing the recall of bins. This is mainly due to erroneous k-mers that mislead kraken2 to assign false labels to sequences. This renders kraken2 as a non-viable binning tool to perform binning of long reads for assembly. We try to mitigate the over estimation of number of bins by restricting the classification confidence to be above 0.1. Note that, we cannot further increase this as genomic k-mers are in scarcity within long reads. As expected, Table C.2 demonstrates that higher confidence mode yield a lower number of bins, however, still over estimating the species count significantly.

Table C.2: Performance of Kraken2 reference based binning tool with higher confidence

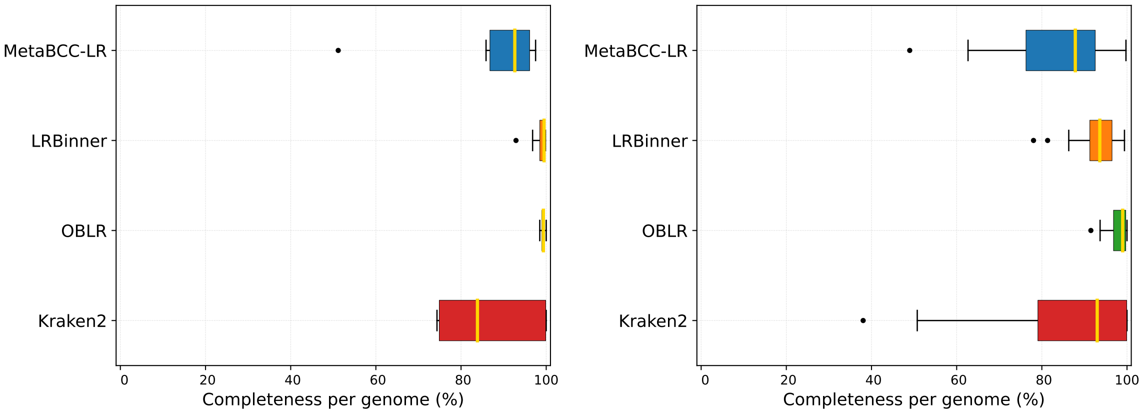
Dataset	No. of species	Bins Detected	Precision	Recall	F1-Score
Sim-8	8	47	39.55%	83.00%	53.57%
Sim-20	20	97	27.93%	83.26%	41.83%
Sim-50	50	181	25.82%	83.20%	39.41%
Sim-100	100	315	42.00%	77.72%	54.53%
ZymoEVEN	10	37	21.66%	99.82%	35.59%
SRR9202034	15	74	99.79%	92.64%	96.08%
SRX9569057	13	80	67.04%	88.18%	76.17%

C.2 AMBER evaluation of binning results

Evaluation of binning using AMBER (Meyer, Hofmann et al., 2018a) shows that Kraken2 produces bins that are lower in completeness compared to our long read binning tools. This is because, the erroneous *k*-mers in long reads confuse Kraken2 to predict different species. This reduces the number of reads in a given bin affecting the overall completeness of the bins as demonstrated in Figures C.1 and C.2. Similarly, the contamination is affected by confusion between different species and the unclassified bins which is created by Kraken2 that contains the reads that did not classify under the criterion it operates (Please refer to Figures C.3 and C.4). The same adverse effects persists in the mode (2) as demonstrated in Figures C.5-C.8.

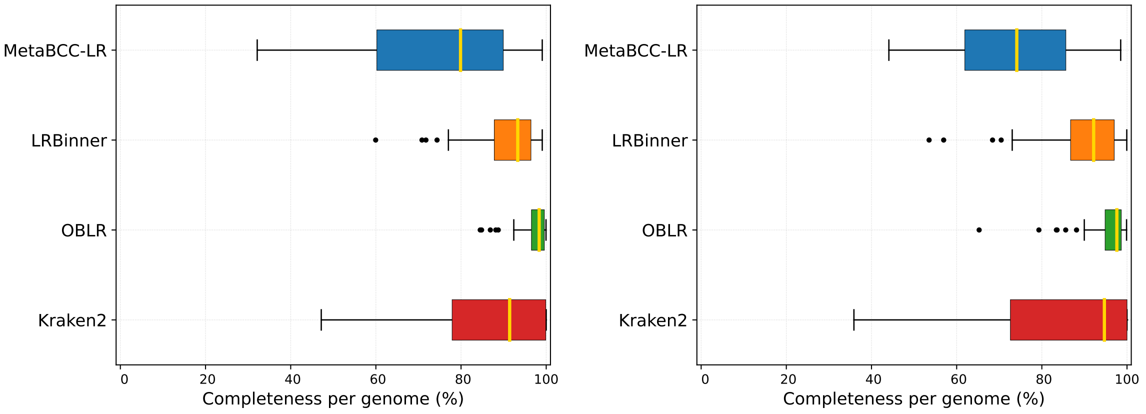
C.3 Discussion

It is evident that reference based binning tools alone may face challenging inputs when faced with erroneous long reads. Reference free binning tools have the tendency of under-estimating the number of bins when presented with complex datasets. This emphasizes the requirement of more sophisticated techniques of combining reference based and reference free binning tools as discussed in Chapter 8.1.3.



(a) Sim-8

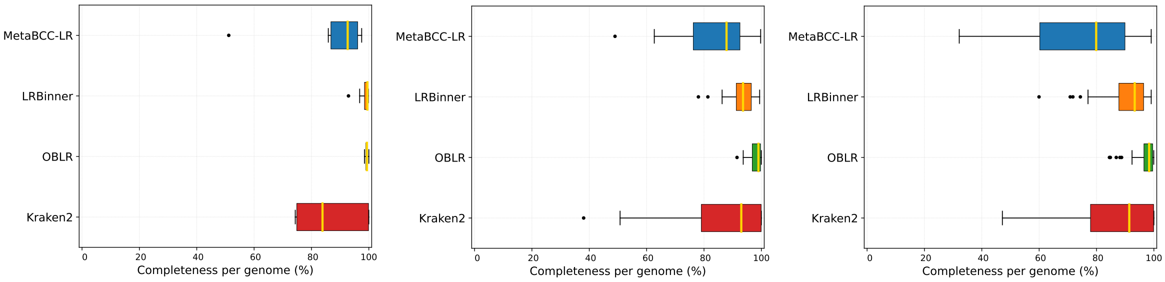
(b) sim-20



(c) Sim-50

(d) Sim-100

Figure C.1: Comparison of bin completeness between simulated datasets with default kraken2 parameters.

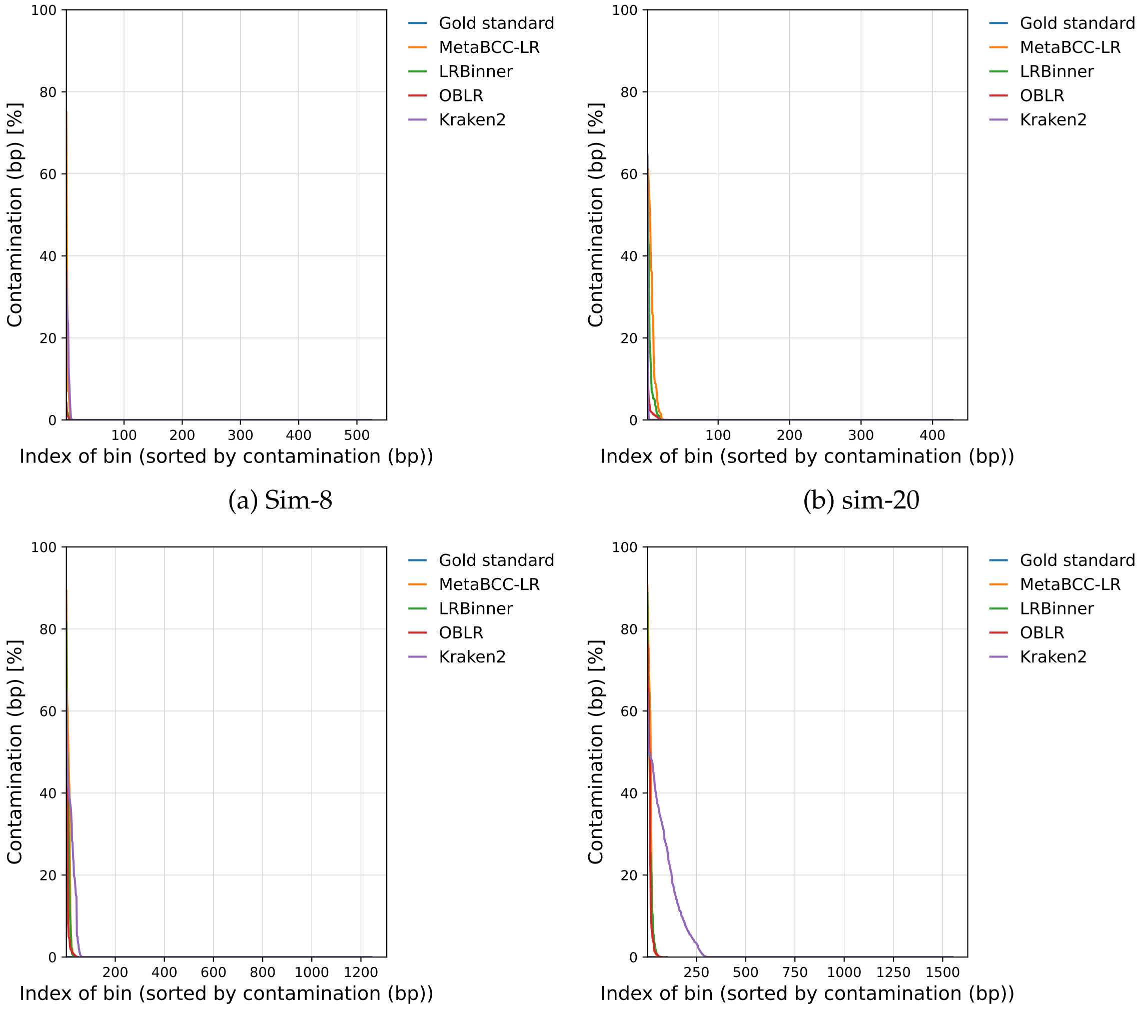


(a) ZymoEVEN

(b) SRR9202034

(c) SRX9569057

Figure C.2: Comparison of bin completeness between real datasets with default kraken2 parameters.



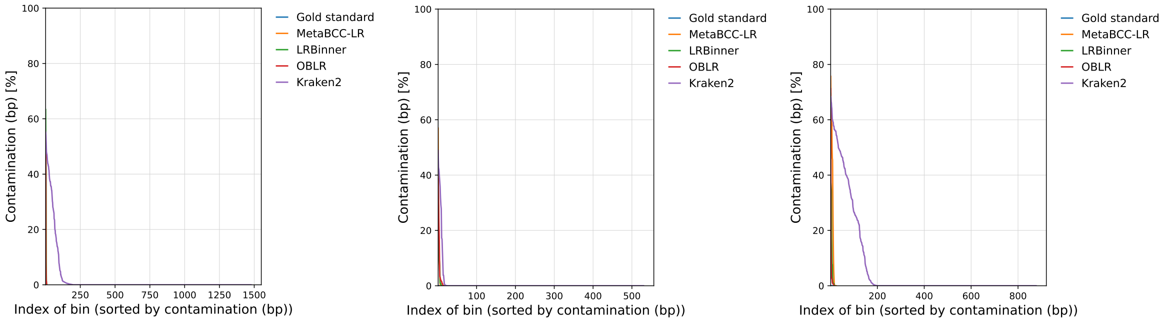
(a) Sim-8

(b) sim-20

(c) Sim-50

(d) Sim-100

Figure C.3: Comparison of bin completeness between simulated datasets with default kraken2 parameters.

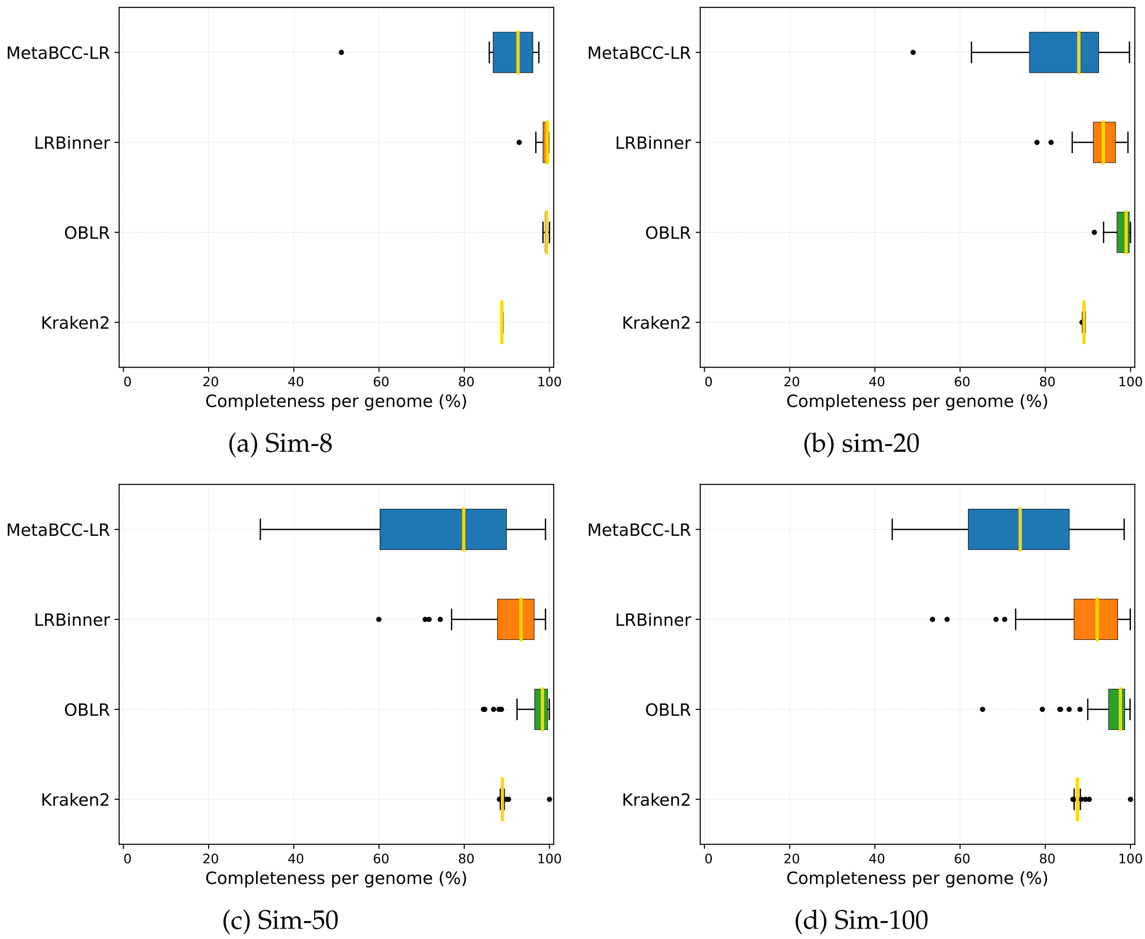


(a) ZymoEVEN

(b) SRR9202034

(c) SRX9569057

Figure C.4: Comparison of bin contamination between real datasets with default kraken2 parameters.



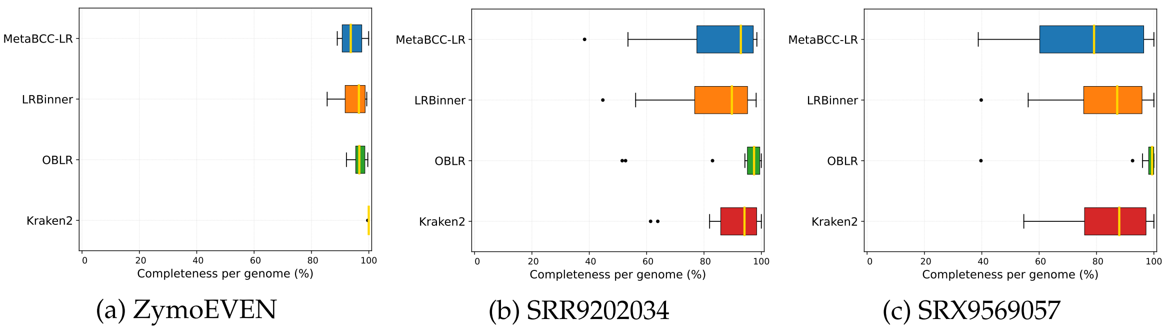
(a) Sim-8

(b) sim-20

(c) Sim-50

(d) Sim-100

Figure C.5: Comparison of bin completeness between simulated datasets with kraken2 higher confidence.

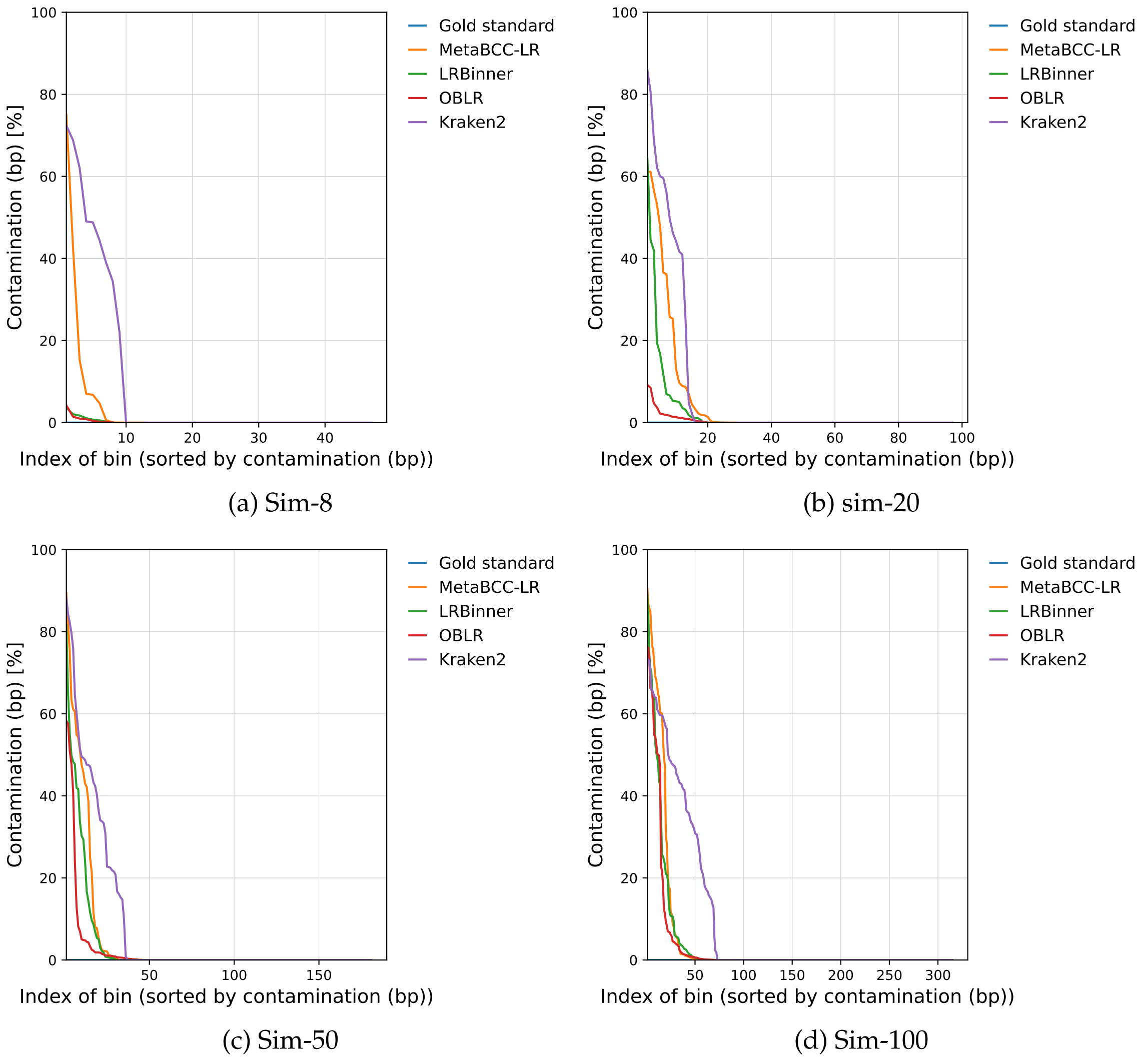


(a) ZymoEVEN

(b) SRR9202034

(c) SRX9569057

Figure C.6: Comparison of bin completeness between real datasets with kraken2 higher confidence.



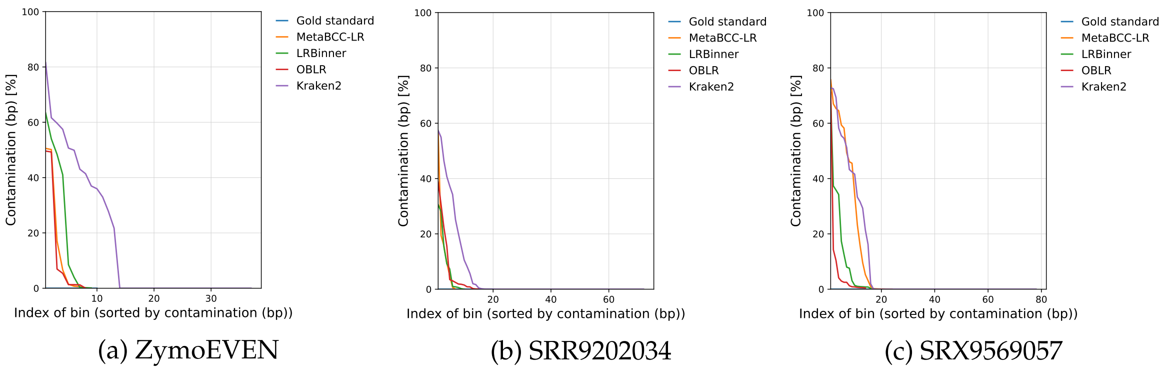
(a) Sim-8

(b) sim-20

(c) Sim-50

(d) Sim-100

Figure C.7: Comparison of bin completeness between simulated datasets with kraken2 higher confidence.



(a) ZymoEVEN

(b) SRR9202034

(c) SRX9569057

Figure C.8: Comparison of bin contamination between real datasets with kraken2 higher confidence.

Appendix D

Discussion on Binning Evaluation Metrics

The binning evaluations are presented using Precision, Recall and F1 score as indicated in Section 3.3.3. Furthermore, stricter evaluations are presented using AMBER (Meyer, Hofmann et al., 2018a) for LRBinner and OBLR. This section explains the evaluation metrics in detail and discusses as to why AMBER evaluations are poor in some cases where the number of bins predicted is further away from the actual number of species in the dataset. Note that the bin assignment matrix a can be presented as $M \times N$, illustrated in Table D.1. Note that $N = 5$ and $M = 7$.

Table D.1: Binning matrix

	Bin 1	Bin 2	Bin 3	Bin 4	Bin 5	Bin 5	Bin 7
Species 1	$a_{11} = 99$	$a_{12} = 0$	$a_{13} = 0$	$a_{14} = 0$	$a_{15} = 0$	$a_{16} = 1$	$a_{17} = 0$
Species 2	$a_{21} = 0$	$a_{22} = 100$	$a_{23} = 0$	$a_{24} = 0$	$a_{25} = 0$	$a_{26} = 0$	$a_{27} = 0$
Species 3	$a_{31} = 0$	$a_{32} = 20$	$a_{33} = 0$	$a_{34} = 0$	$a_{35} = 0$	$a_{36} = 0$	$a_{37} = 0$
Species 4	$a_{41} = 0$	$a_{42} = 0$	$a_{43} = 1000$	$a_{44} = 50$	$a_{45} = 0$	$a_{46} = 0$	$a_{47} = 0$
Species 5	$a_{51} = 0$	$a_{52} = 0$	$a_{53} = 0$	$a_{54} = 0$	$a_{55} = 200$	$a_{56} = 0$	$a_{57} = 300$

Recall the equations Precision and Recall.

$$Precision(\%) = \frac{\sum_{i=1}^{M} max_j\{R_{ij}\}}{\sum_{i=1}^{M}\sum_{j=1}^{N}\{R_{ij}\}} \times 100 \tag{D.1}$$

$$Recall(\%) = \frac{\sum_{j=1}^{N} max_i\{R_{ij}\}}{\sum_{i=1}^{M}\sum_{j=1}^{N}\{R_{ij}\} + Number\ of\ unclassified\ reads} \times 100 \tag{D.2}$$

Recall is computed for each species, by taking the largest assignment to a bin. Precision is computed per bin taking the largest assignment of the bin to a given species. In contrast, AMBER uses purity and completeness to compute the per-bin F1 score using the following equations, for each bin b.

$$Purity = \frac{true\ positives_b}{true\ positives_b + false\ positives_b} \tag{D.3}$$

$$Completeness = \frac{true\ positives_b}{true\ positives_b + false\ negatives_b} \tag{D.4}$$

The true positives are computed using the majority species in a given bin. Because of this, if a bin appears as a result of a false bin split (1% reads), the completeness of the smaller bin will be very low (approximately 1%) according to AMBER evaluation. In comparison, the recall of the species using equation D.2 will report 99% since 99% of the reads are in a single bin despite having the false bin split. Similarly, the false split of the bin will report a greater precision as long as the bin has no other species mixed according to equation D.1. Such precision and recalls are appropriate in the read binning context, given the fact that a proper assembly can be reached with 99% reads due to the larger number. This is not the case in AMBER which is designed to mostly evaluate contig bins where wrong splits can significantly misinterpret the biological environment. Consider the following running example.

Example 1

Suppose Species 1 has $a_{11} = 99$ and $a_{16} = 1$ with rest of the row having no reads and Bin 1 and 6 has no reads from another species. Purity in this case will be 100% for both bins 1 and 6 while completeness will be 99% and 1% respectively. F1-score will be 99.5% and 1.98% with average being very low at 50.7%. Recall will be 99% for Species 1 with 100% precision on both bins 1 and 6 since there are no impurities in each bin, thus, F1-score is 99.5% for each bin.

Example 2

Suppose Bin 2 has $a_{22} = 100$ and $a_{32} = 20$, with two species 2 and 3, with no other contaminants and species 2 and 3 are fully contained in the bin. Now, the purity of the bin is 83.33% and completeness is 83.33%, hence, F1-score is 83.33%. Recall for species 2 and 3 will be 100% since it is not split into multiple bins. However, the precision for Bin 2 will be 83.33%, hence a F1 score of 90.91%.

This means, AMBER penalizes whenever a species is split into many bins while not significantly penalizing bin mergers between large bins and smaller bins. This is because, dominant species in the bin will determine the purity and completeness.

In conclusion, the disparity of results in AMBER evaluation is largely due to the computation of per-bin F1 score using per-bin purity and completeness. However, such comparison is essential to show the importance of accurate binning.