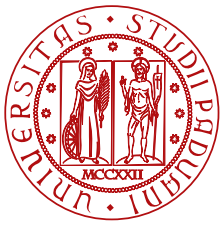


UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN COMPUTER ENGINEERING

Compression of Sequencing Data for Phylogeny Reconstruction

Relatore

Prof. Comin Matteo

Laureando

Nicetto Andrea

ANNO ACCADEMICO 2024-2025

Data di laurea 26/03/2025

Abstract

Next-generation sequencing technologies have revolutionized genomics by producing large amounts of data that challenge conventional methods of archiving and analysis. This thesis addresses two key issues: efficient compression of sequencing data and reconstruction of phylogenies using alignment-free approaches. A new compression strategy based on de Bruijn compacted graphs is introduced, which minimizes data redundancy by optimizing path coverages in k-mer sets. The proposed USTAR method exploits the inherent connectivity of these graphs to significantly reduce storage requirements while preserving the information in the data. Building on this foundation, the study employs some alignment-free phylogenetic reconstruction techniques, namely phyBWT2, Mash and SANS serif. These tools bypass the traditional, and computationally expensive, need for sequence alignment using innovative methods such as the extended Burrows-Wheeler transform and MinHash sketching, accelerating tree inference without compromising accuracy.

Extensive experiments on real and simulated datasets show that using USTAR for data compression, combined with alignment-free reconstruction, significantly improves computational efficiency by reducing storage space and execution time while preserving the integrity of the sequencing information and the quality of phylogenetic analyses.

Contents

List of Figures

Chapter 1

Introduction

The advent of next-generation sequencing technologies has radically transformed the field of genomics, generating larger amounts of data. This rapid growth of information creates numerous challenges in terms of both storage and computational processing. In this context, compression of sequencing data is not only essential to reduce the consumption of memory and storage resources, but is also a key step to facilitate subsequent analyses, such as phylogenetic reconstruction.

This thesis focuses on two fundamental aspects: on one hand, the evaluation of an efficient compression methodology based on compacted De Bruijn graphs; and on the other, the application of these techniques to phylogenetic reconstruction using alignment-free approaches. In particular, the USTAR method is proposed, which exploits the connectivity properties of graphs to optimize the representation of k-mer sets while concurrently reducing data redundancy. The efficient compression of k-mers allows for the management of considerably large datasets and accelerates comparison and phylogenetic inference.

The methods employed for phylogenetic reconstruction, including phyBWT2, Mash, and SANS serif, are examples of alignment-free software capable of processing compressed or uncompressed data directly without the costly assembly step. These innovative approaches, based respectively on the Extended Burrows-Wheeler Transform, the MinHash technique, and the analysis of splits derived from the entire genome, offer significant advantages in terms of speed and scalability while maintaining a high level of accuracy in estimating evolutionary relationships.

The thesis aims to: Examine the challenges associated with the management and compression of sequencing data, highlighting the limitations of traditional approaches.

Introduce and evaluate the USTAR method for compressing k-mer sets, illustrating its functionality and advantages.

Apply and compare alignment-free methods for phylogenetic reconstruction, analyzing the

computational performance and quality of the results obtained on both unassembled and compressed datasets.
Provide a quantitative evaluation based on standard metrics, such as the compression ratio and the Robinson-Foulds distance, to compare the different methodologies.

1.1 Project Structure

This thesis is divided into seven chapters.
The first chapter introduces the research context, highlighting the challenges in genomics data compression and outlining the objectives and contributions of the thesis.
The second chapter provides the theoretical background necessary to understand the study, covering next-generation sequencing technologies, k-mers, De Bruijn graphs, and the Burrows-Wheeler Transform.
Chapter three describes the methods employed, including the compression technique and the alignment-free phylogenetic reconstruction algorithms used in the analysis.
Chapter four details the experimental setup, including the datasets, simulation parameters, and the procedures followed to evaluate the proposed methods.
Chapter five presents the results obtained from the compression and phylogenetic analyses, providing performance metrics and comparative evaluations.
Chapter six discusses the findings, analyzing the strengths and limitations of the proposed framework and its implications for future research.
Finally, chapter seven summarizes the main conclusions, highlighting key contributions and suggesting potential directions for further studies.

Chapter 2

Theoretical Background

This chapter provides an overview of the fundamental concepts necessary to understand the methods employed in this study. It introduces the fundamental principles in sequencing technologies, graph-based representations of k-mers, and the Burrows-Wheeler Transform.

2.1 Next Generation Sequencing Technologies

Sequencing plays a key role in bioinformatics by generating DNA fragments that can be processed to study genetic variation, reconstruct genomes, or perform many other analyses.
Since the days of Sanger sequencing, next-generation sequencing technologies have significantly evolved to provide increased data output, efficiencies, and applications. These next generations of technologies can be categorized based on read length into two paradigms: short-read, or 'second-generation,' technologies, and long-read, or 'third-generation,' technologies.

2.1.1 Short-Read Sequencing Technologies

Short-read NGS, or second-generation sequencing, refers to the next advancement of sequencing technologies after traditional first-generation Sanger sequencing. The common feature of short-read technologies is massive sequencing of short (250–800 bp), clonally amplified DNA molecules sequenced in parallel [1].
The workflow of this procedure is divided into three parts: library preparation, sequencing and data analysis. First, the DNA is fragmented, then the ends of the fragments are repaired, adapters are ligated to the fragments and the library is amplified to ensure there is enough material for sequencing. After the library is ready, it is subjected to high-throughput sequencing strategies including the Illumina and Ion Torrent. Both are based on the sequencing-by-synthesis (SBS) method, in which nucleotides are added one at a time to growing DNA strands, and each nucleotide addition is followed by a detectable signal.

Among the available sequencing technologies, Illumina sequencing has emerged as the most popular tool for generating short sequence reads. The process starts with the adsorption of DNA fragments to a flow cell, which is then followed by bridge PCR to create millions of copy number matched, spatially separated molecules, each representing one DNA fragment. This is because during sequencing, chemically modified nucleotides are cycled in sequence, and the emitted light is detected by a high sensitivity optical system. The modified nucleotides are reversible, which means that no more than one nucleotide is added at a given cycle, thus reducing the chances of making errors during sequencing. The paired-end sequencing strategy, which is popular in Illumina systems, improves accuracy by sequencing the DNA fragments from both ends to cover more regions and solve problems in regions of the genome with complicated structure [1]. A simple example of the Illumina workflow is presented in Figure 2.1.
However, there are some disadvantages of the method; for example, in the case of repetitive genomes and highly complex structures, it is difficult to place short fragments back together. Nevertheless, it is precise and fast, and these features have made it the gold standard in many genomic analyses, including WGS, RNAseq, and metagenomics.

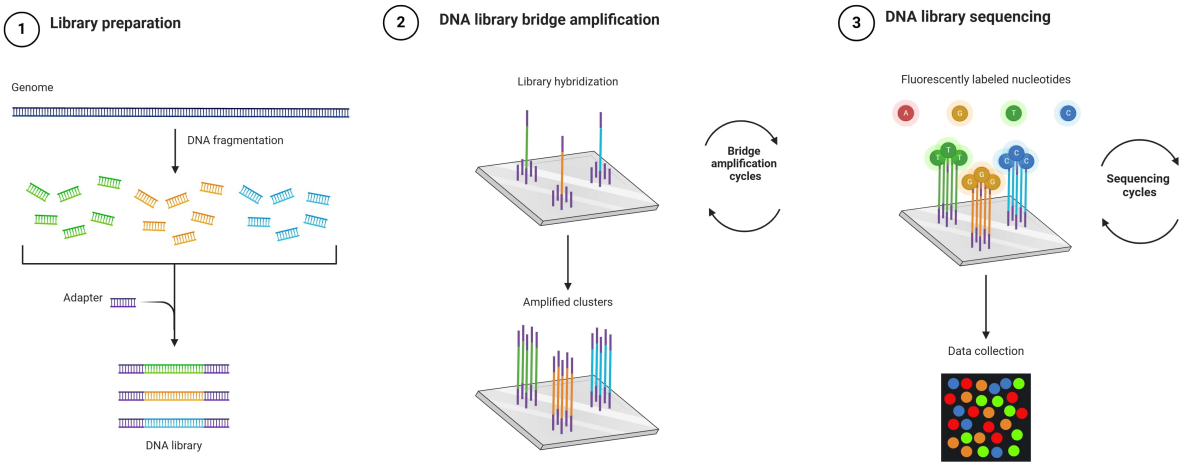


Figure 2.1: Example of Illumina technology workflow.

2.1.2 Long-Read Sequencing Technologies

Compared to short-read sequencing technologies, long-read technologies, also called third-generation sequencing technologies, can generate sequences greater than 10 kb directly from native DNA. While early iterations of these technologies were plagued with inaccuracies, more recent modifications and improvements have enabled much higher accuracy and offer exciting possibilities to sequence large DNA molecules, such as for the diagnosis of genetic diseases [1].

There are two main technologies for long-read sequencing. The first is Pacific Biosciences (PacBio) SMRT (Single Molecule, Real-Time) sequencing, which is a third-generation method for DNA sequencing that uses a circular single-stranded DNA, called the SMRTbell template. This template is created by joining hairpin adapters at both ends of the double-stranded DNA. Sequencing takes place on a zero-mode waveguide (ZMW) chip, where each ZMW houses a single DNA polymerase. When the polymerase extends the DNA, it incorporates fluorescently labeled nucleotides, producing light pulses that are captured and interpreted as sequences. These sequences, Continuous Long Reads (CLRs), can be subjected to further analysis to improve accuracy, creating Circular Consensus Sequences (CCS or HiFi reads). HiFi reads are highly accurate and provide longer sequences than traditional short-read methods.

The second technology reported in [1] is Oxford Nanopore Technology (ONT), which uses nanopores to sequence single-stranded nucleic acids, enabling very long reads of DNA or RNA. ONT can generate reads in excess of 1 Mb and, if computationally stitched, over 2 Mb. It works by detecting interruptions in ion current as nucleotides pass through protein pores. ONT technology can produce standard (10-100 kb) and ultra-long (>100 kb) reads, although ultra-long reads have lower throughput. The utility of this technology extends to DNA and RNA sequencing, including the detection of modifications without the biases typical of cDNA synthesis. Despite higher error rates than second-generation methods, improvements in base calling algorithms and software adaptations have made ONT sequencing increasingly viable in practical applications, such as real-time HLA typing with reduced run times. The technology remains promising, and future improvements in error rates could increase its clinical applicability, comparable to the advances seen in PacBio HiFi sequencing. An example of PacBio and ONT technologies can be observed in Figure 2.2 extracted from [2].

Despite these advantages, long read sequencing technologies still have limitations, including higher error rates per base than short read methods and higher sequencing costs. However, ongoing advances continue to refine these technologies, making them increasingly effective for a wide range of genomic studies.

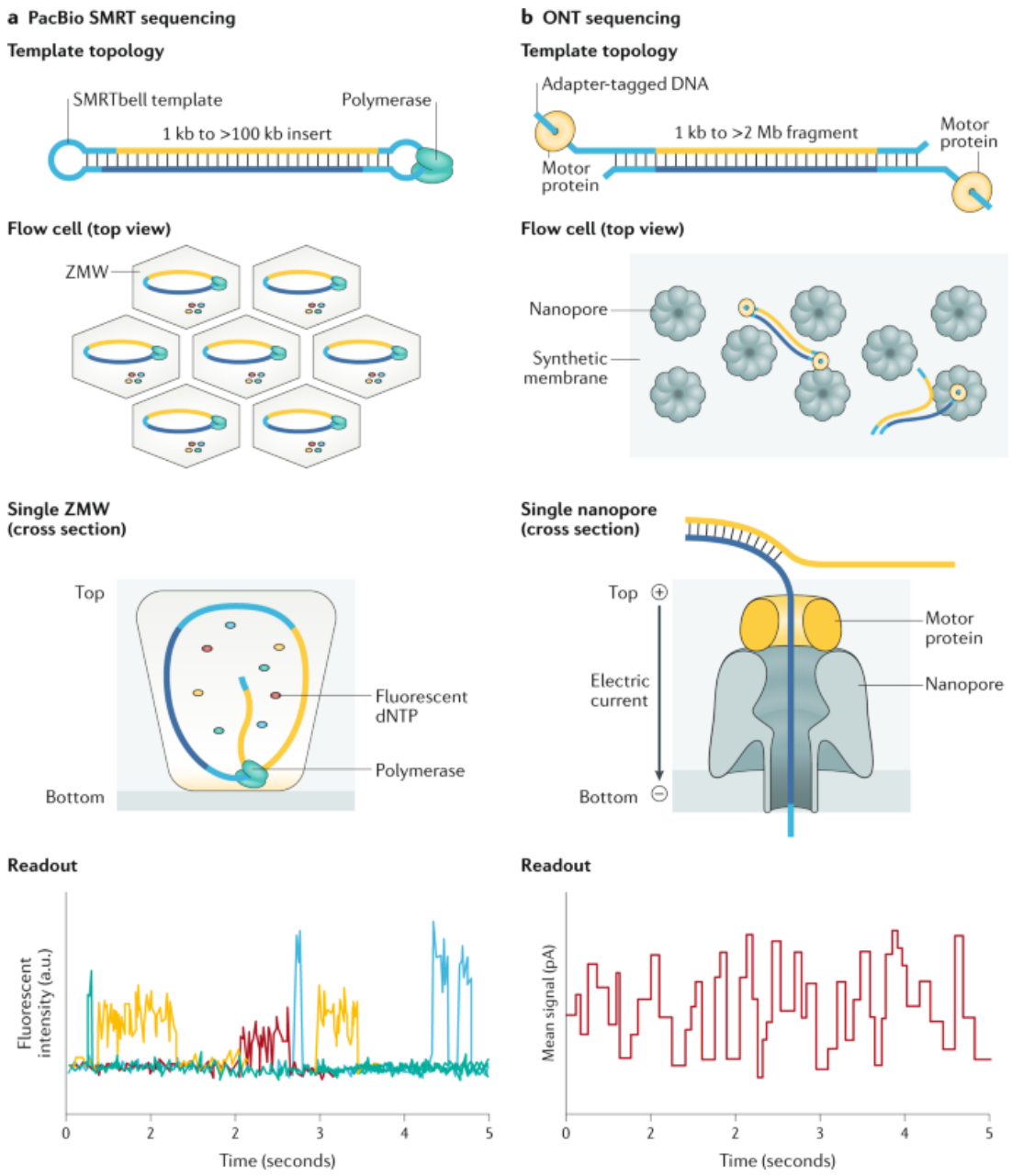


Figure 2.2: Example of PacBio SMRT and ONT technologies for long-read sequencing.

2.2 Definitions: k-mers and De Bruijn Graphs

2.2.1 k-mers

A k-mer is defined as a substring of length k extracted from a given sequence over an alphabet Σ. For example, given the nucleotide alphabet $\Sigma = \{A, C, G, T\}$ and $k = 3$, a k-mer is any sequence of three consecutive nucleotides (e.g., *ATG*, *TGC*). In general, there are $|\Sigma|^k$ possible k-mers [3].

2.2.2 De Bruijn Graphs

In the context of genome assembly, a de Bruijn graph is a directed graph constructed from a set K of k-mers. In this graph, each node represents a $(k-1)$-mer and a directed edge is drawn from node u to node v if there exists a k-mer in K whose prefix (the first $k-1$ symbols) is u and whose suffix (the last $k-1$ symbols) is v. Thus, every k-mer in K corresponds to an edge in the de Bruijn graph, linking its prefix and suffix.
Figure 2.3 illustrates the construction of a de Bruijn graph for genome assembly. The figure, taken from [3] contrasts the traditional Sanger approach, where reads are represented as nodes with overlaps as edges, with the modern short-read assembly method that builds the de Bruijn graph by representing k-mer prefixes and suffixes as nodes and connecting them with edges corresponding to k-mers.

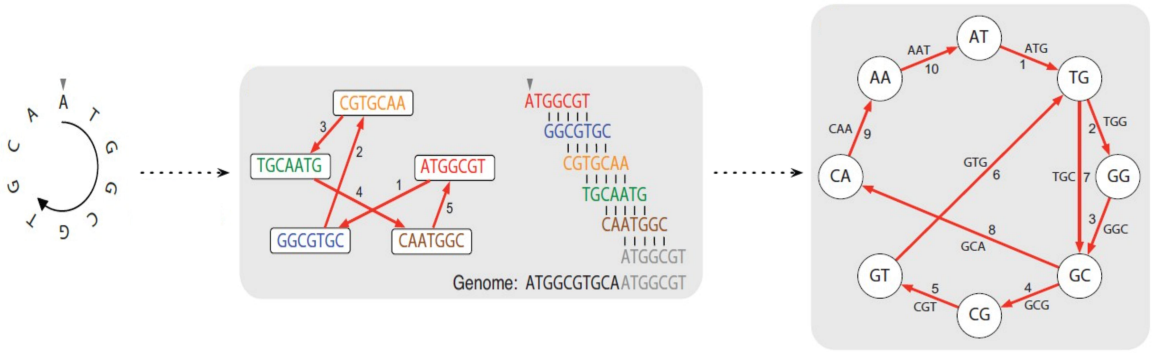


Figure 2.3: Example of De Bruijn graph for the string $S = ATGGCGT$, with k=3.

2.2.3 Compacted De Bruijn Graphs

A compacted de Bruijn graph is a simplified version of the standard de Bruijn graph. In a conventional de Bruijn graph, each node corresponds to a (k-1)-mer, which can lead to a large and redundant structure. Compacted de Bruijn graphs reduce this redundancy by joining linear, unbranched paths into single nodes. The label of a compacted node is the concatenation of the original k-mers along the path, with the overlapping regions included only once. This process significantly reduces the number of nodes and edges, thus decreasing memory usage and computational overhead. A minimal example of a de Bruijn graph and a compacted de Bruijn graph is provided in Figure 2.4, taken from [4].

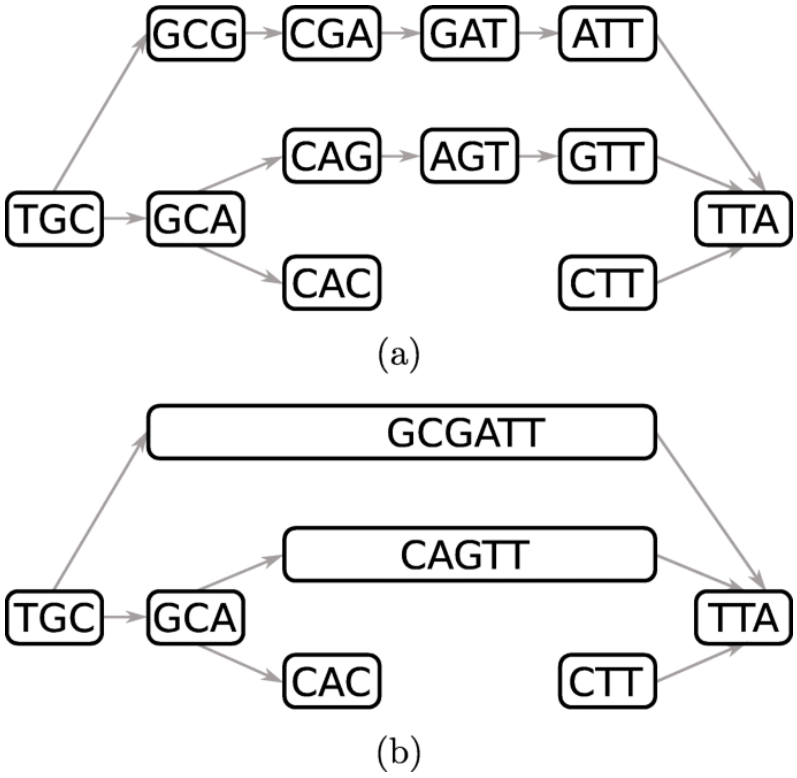


Figure 2.4: A de Bruijn graph in (a) and its compacted counterpart in (b) using 3-mers.

The compacted de Bruijn graphs provide an efficient representation of sequence data, which is essential for subsequent analyses, such as k-mer compression in USTAR, which will be presented in Chapter 3. For more information on compacted de Bruijn graphs, see [5].

2.3 The Burrows-Wheeler Transform

The Burrows-Wheeler Transform (BWT) is a reversible transformation applied to a block of text to rearrange its characters in a way that facilitates more effective compression by subsequent algorithms such as move-to-front coding and entropy coding. Given an input string T of length N, drawn from an ordered alphabet X, the transform begins by considering every cyclic rotation of T. These N rotations are then sorted lexicographically to form a conceptual $N \times N$ matrix M, where each row corresponds to one rotation and one of these rows is identical to the original string T. From the sorted matrix, the last column is extracted to form a new string L, so that each character $L[i]$ is the final character of the i-th sorted rotation. Simultaneously, the index I of the row containing the original string is recorded. The output of the transformation is thus the pair (L, I) [6]. A simple example of the transformation can be seen in Figure 2.5.

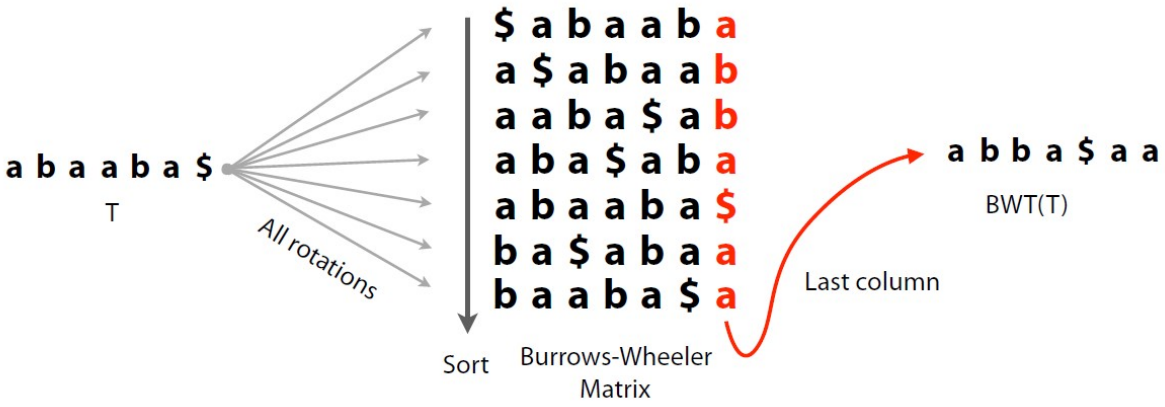


Figure 2.5: Example of Burrows-Wheeler Transform.

Although BWT does not compress data directly, it rearranges characters in a way that groups together symbols that occur in similar contexts, making the transformed string more suitable for subsequent compression and analysis. This has led to the development of variants of BWT, including the Extended Burrows-Wheeler Transform, which is an adaptation of the standard BWT designed to handle a set of multiple sequences rather than a single string.

2.3.1 The Extended Burrows-Wheeler Transform

The Extended Burrows-Wheeler Transform (EBWT) builds on this idea by extending the transformation to a collection of sequences rather than just one. In the EBWT, each sequence is treated as circular, and all its cyclic rotations are considered. These rotations are then sorted using an order that compares the infinite repetitions of the sequences, ensuring that rotations sharing long common prefixes are naturally grouped.

From the sorted list, two key pieces of information are extracted: one is a string composed of the last characters of each rotation, and the other is a characteristic vector that marks which rotations correspond to the original sequences. Together, these outputs form the EBWT of the set, and the transformation is fully reversible. More information can be found in [7].

A remarkable aspect of the EBWT is its clustering property. In the EBWT, sequences with common patterns are placed in nearby positions, so that regions sharing similar features naturally cluster together. This inherent clustering is the key concept underlying the phyBWT2 algorithm presented in Chapter 3.

Chapter 3

Methods

This chapter examines two key aspects of modern bioinformatics: efficient compression of genomic data and phylogenetic reconstruction without alignments. Advances in k-mer based compression reduce storage requirements and improve genomic analyses, while new phylogenetic methods rapidly infer evolutionary relationships without traditional alignments.

3.1 Compression in Bioinformatics

A fundamental operation in computational genomics is the reduction of input sequences into their constituent k-mers. This conversion transforms variable-length sequencing data into a fixed-length representation, which facilitates more efficient analyses on extensive datasets. However, storing all k-mers extracted from a set of sequences can be highly storage-intensive. To mitigate this issue, researchers have developed strategies that exploit the intrinsic redundancy present in sequencing data.

One adopted strategy is to represent the k-mer set as a plain text collection of strings. Formally, a plain text representation is defined as a collection of strings that includes every k-mer (and its reverse complement) derived from the input sequences, while excluding any extraneous substrings. This representation, allowing repeated k-mers within or across strings, is known as a *Spectrum-Preserving String Set* (SPSS) [8]. In contrast to alternative definitions (e.g., that by Rahman and Medvedev [9], which requires each k-mer to appear only once), the SPSS here is more flexible.

A SPSS for a given set of k-mers $K \subseteq \Sigma^k$, where $\Sigma = \{A,C,G,T\}$, is defined as a collection of strings S such that every string $s \in S$ satisfies $|s| \geq k$, and the k-mer spectrum of S is exactly equal to K.

$$\text{Spectrum}(S) = \{\, t \in \Sigma^k \mid \exists\, s \in S \text{ such that } t \text{ or } \text{rc}(t) \text{ is a substring of } s \,\} = K$$

Here, rc(t) denotes the reverse complement of t. An additional useful measure for evaluating the compactness of such a representation is the cumulative length (CL) of S, defined as

$$CL(S) = \sum_{s \in S} |s|$$

3.2 USTAR: Improved Compression of k-mer Sets with Counters Using de Bruijn Graphs

As reported in [10], USTAR is a computational tool designed for the efficient compression of k-mer sets along with their associated counts. It leverages the connectivity and density properties of the compacted de Bruijn graph to optimize path selection in the construction of the path cover, achieving superior compression efficiency compared to existing methodologies.

The core concept of USTAR involves the systematic investigation of the compacted de Bruijn graph $dBG(K)$, with K being a specified collection of k-mers. In a formal sense, the de Bruijn graph is expressed as $G = (V, A)$, where each node $v \in V$ signifies a k-mer, and the directed arcs A denote overlaps of length $k-1$ between successive k-mers.

The k-mer set compression problem can be formulated as a minimum vertex-disjoint path cover problem, which minimizes the number of vertex-disjoint paths that cover the graph G. The cumulative length (CL) of a set of strings S encoding K is given by:

$$CL(S) = |K| + (k-1) \cdot |S| \tag{3.1}$$

To minimize $CL(S)$, USTAR employs a two-stage heuristic for path construction. The first stage involves selecting an optimal seed node, while the second stage dictates the extension of paths to maximize compression efficiency.

In the seed selection phase, USTAR prioritizes nodes with the highest average k-mer count, as these nodes typically contribute the most to redundancy. This selection criterion can be expressed with the following equation:

$$\text{Seed node} = \arg\max_{v \in V} \frac{\sum_{u \in N(v)} C(u)}{|N(v)|} \tag{3.2}$$

where $C(u)$ denotes the k-mer count at node u, and $N(v)$ is the neighborhood of v.

Following seed selection, the path extension phase is governed by a strategy designed to prevent fragmentation. Instead of arbitrarily extending the path, USTAR selects the next node u by minimizing its degree, ensuring that highly connected nodes remain available for subsequent

paths:

$$\text{Next}(v) = \arg\min_{u \in N(v)} |N(u)| \tag{3.3}$$

This heuristic favors the formation of longer sequences, thus reducing storage requirements and improving compression performance. An example of this process is illustrated in Figure 3.1, where the connectivity of the compacted de Bruijn graph is exploited to optimize path selection. Empirical evaluations presented in [10] show that USTAR outperforms other state-of-the-art compression techniques for k-mer sets. The method's ability to navigate graph connectivity results in a more efficient representation from the perspective of k-mer set space, making it a valuable tool for large-scale genomic data analysis.

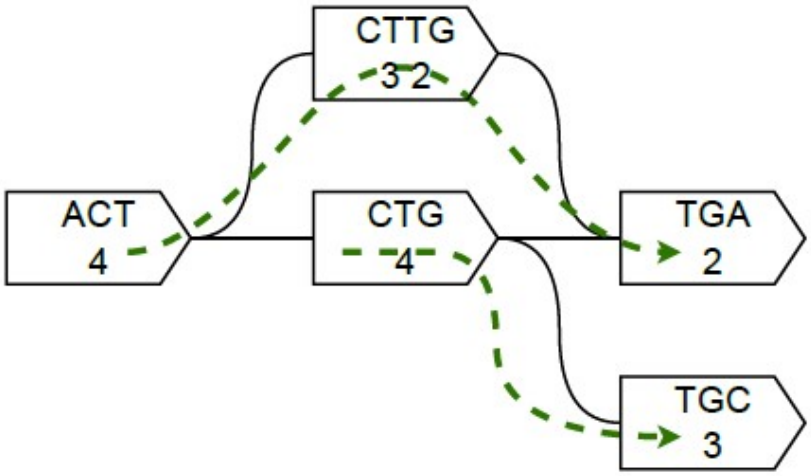


Figure 3.1: Example of USTAR's path selection strategy applied to a de Bruijn graph.

3.3 Phylogeny Reconstruction

The field of phylogenetic reconstruction has evolved considerably in recent decades. The work of Sokal and Sneath [11] and Cavalli-Sforza and Edwards [12] laid the conceptual foundation by advocating the use of parsimony as a criterion for inferring evolutionary relationships with the least amount of change. The emergence of molecular data in the 1970s, particularly following the development of Sanger sequencing, transformed these ideas, providing objective, sequence-based evidence for evolutionary history.

In the early 1980s, Felsenstein introduced maximum likelihood methods, which integrated statistical models of sequence evolution into phylogenetic inference [13]. These model-based approaches, which account for variable evolutionary rates across sites, marked a significant departure from purely parsimonious analyses and improved the robustness of tree estimation. Later, software such as MrBayes [14] advanced the field further by incorporating Bayesian methods, allowing researchers to tackle more complex data sets with greater statistical rigor.

More recently, the growth of genomic data has accentuated the development of alignment-free methods that bypass the computationally expensive step of sequence alignment. Some of these methods are presented in the following subsections.

3.3.1 phyBWT2: Phylogeny Reconstruction via eBWT Positional Clustering

phyBWT2 is an alignment-free, assembly-free, and reference-free tool for phylogenetic tree reconstruction that directly processes sequencing data, including raw reads, without requiring prior genome assembly or reference mapping. It represents an improvement over its predecessor, phyBWT, achieving better efficiency in tree construction while maintaining high accuracy. phyBWT2 exploits the combinatorial properties of the extended Burrows-Wheeler Transform (eBWT) and a positional clustering framework to detect relevant blocks of longest shared substrings of varying lengths. Unlike traditional k-mer-based methods that require a predefined k-mer length, phyBWT2 dynamically determines substring boundaries based on positional clustering in the eBWT.

The algorithm begins by constructing the eBWT of the input sequences. This transformation is accompanied by auxiliary data structures such as the Document Array (DA) and the Longest Common Prefix (LCP) array, which allow efficient retrieval of sequence similarity information, as illustrated in Figure 3.2.

i	cda	da	lcp	ebwt	Sorted suffixes
1	1	1	0	A	$
2	2	2	0	T	$
3	3	3	0	T	$
4	1	1	0	C	A$
5	1	1	1	T	ACCA$
6	3	3	2	T	ACGACT$
7	3	3	4	$	ACGAGTACGACT$
8	3	3	2	G	ACT$
9	3	3	1	G	AGTACGACT$
10	2	2	1	T	ATT$
11	1	1	0	C	CA$
12	1	1	1	A	CCA$
13	3	3	1	A	CGACT$
14	3	3	3	A	CGAGTACGACT$
15	1	1	2	G	CGTACCA$
16	2	2	4	G	CGTATT$
17	3	3	1	A	CT$
18	3	3	0	C	GACT$
19	3	3	2	C	GAGTACGACT$
20	1	1	1	G	GCGTACCA$
21	2	2	5	G	GCGTATT$
22	1	1	1	$	GGCGTACCA$
23	2	2	6	G	GGCGTATT$
24	2	2	2	G	GGGCGTATT$
25	2	2	3	$	GGGGCGTATT$
26	1	1	1	C	GTACCA$
27	3	3	4	A	GTACGACT$
28	2	2	3	C	GTATT$
29	2	2	0	T	T$
30	3	3	1	C	T$
31	1	1	1	G	TACCA$
32	3	3	3	G	TACGACT$
33	2	2	2	G	TATT$
34	2	2	1	A	TT$

Figure 3.2: Extended Burrows-Wheeler Transform (eBWT), LCP array, and the auxiliary data structures DA and CDA for the set $S = \{GGCGTACCA, ACGAGTACGACT, GGGGCGTATT\}$.

Using these structures, phyBWT2 identifies clusters of suffixes that share long common prefixes, thereby defining meaningful partitions of the taxa. Instead of applying a single partitioning step, the tool refines multiple partitions simultaneously, progressively shaping the final tree structure. Unlike phyBWT [15], which applies a more rigid partitioning strategy, phyBWT2 enhances tree reconstruction by refining subsets of taxa at each step. The resulting partition tree is iteratively refined until no further subdivisions can be made. The final output is an unrooted phylogenetic tree in Newick format, ready for further analysis.

The entire phylogenetic reconstruction process performed by phyBWT2 is described in the algorithm presented in [16], as shown in Figure 3.3. A crucial step in this process is the REFINEMENT function, which plays a key role in improving the accuracy of taxon partitions. This function takes an initial partitioning of the taxa and further refines it by analyzing shared substrings in the eBWT. Instead of relying on a predefined set of rules, REFINEMENT dynamically adjusts partitions based on positional clustering, ensuring that taxa sharing more informative substrings remain grouped together while reducing noise from spurious matches. This iterative refinement process is essential for achieving a final tree structure that accurately reflects evolutionary relationships.

Algorithm 1: Iterative refinement of the partition tree

input : ℓ, ebwt($\mathcal{S}$), lcp($\mathcal{S}$), cda($\mathcal{S}$)
output: A tree whose leaves are colored with $1 \dots \ell$, each color being a taxon of $\mathcal{S}$

Let $T \leftarrow$ Rooted star with a non-final root $\mathcal{S}$, and whose leaves are colored $1 \dots \ell$, each marked as final
Queue.push($\mathcal{S}$)
while Queue *is not empty* **do**
    $X \leftarrow$ Queue.pop()
    Let $C_1, \dots, C_h$ be the h children of X in T
    $L \leftarrow$ REFINEMENT(ebwt($\mathcal{S}$), lcp($\mathcal{S}$), cda($\mathcal{S}$), $\{C_1, \dots, C_h\}$)
    DRAW_AND_MARK($L, T, Queue$)
Function DRAW_AND_MARK *(T, L, X, Queue)*
    if L *is empty* **then**
        Mark X in T as final // cannot further refine X
    else
        foreach *set* L_i *of* L **do**
            Add L_i as a node in T if not already present
        Mark as final every new node with two children in T
        Add to the queue all new nodes not marked final
        Mark X as final if it has two children, otherwise add it to the Queue
Return T

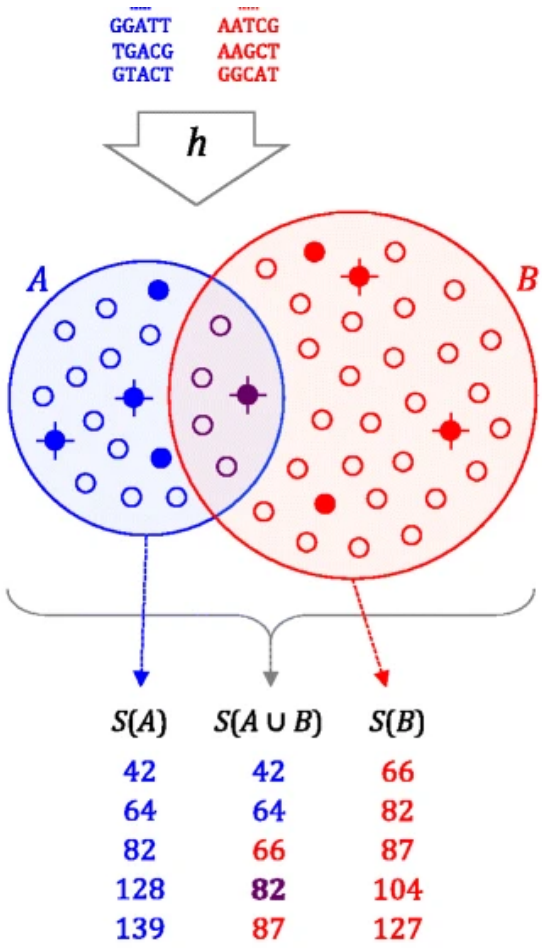
Figure 3.3: Algorithm describing the phylogenetic reconstruction process implemented in phyBWT2.

Empirical results reported in [16] demonstrate that phyBWT2 reconstructs phylogenies with comparable accuracy to traditional methods while significantly improving computational efficiency. The elimination of distance matrices and pairwise sequence comparisons makes it a scalable solution for large-scale phylogenetic analysis.

3.3.2 Mash: Fast Genome and Metagenome Distance Estimation

Mash is an efficient, alignment-free tool for estimating genomic and metagenomic distances using the MinHash technique. It provides a rapid method for clustering and searching large collections of sequences by reducing entire genomes to small representative sketches from which global mutation distances can be estimated. Unlike traditional alignment-based approaches, Mash enables rapid comparisons without the need to process entire sequences, making it particularly suitable for large-scale genomic datasets.

Instead of performing comparisons between full sequences, Mash divides the sequences into k-mers and selects a representative sketch, thus enabling rapid comparison of similarities. The degree of similarity is measured using the Jaccard index, which quantifies the percentage of k-mers shared between two organisms or DNA segments. An example of how the Jaccard index is calculated is shown in Figure 3.4.



$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \approx \frac{|S(A \cup B) \cap S(A) \cap S(B)|}{|S(A \cup B)|}$$

Figure 3.4: Overview of the MinHash bottom sketch strategy for estimating the Jaccard index.

Mash provides two basic functions for sequence comparisons: *sketch* and *dist*. The *sketch* function converts a sequence or collection of sequences into a MinHash sketch, while the *dist* function compares two sketches and returns an estimate of the Jaccard index (i.e., the fraction of shared k-mers), a P value, and the Mash distance, which estimates the rate of sequence mutation under a simple evolutionary model.

Experimental results demonstrate that Mash can efficiently process large datasets while maintaining high accuracy, making it a valuable tool in comparative genomics and metagenomics. Further details on the method and its performance can be found in [17].

3.3.3 SANS serif: Alignment-Free, Whole-Genome-Based Phylogeny Estimation

SANS serif is a novel software for alignment-free, whole-genome-based phylogeny estimation that follows a pangenomic approach to efficiently compute a set of splits in a phylogenetic tree or network [18]. In pangenomics and computational phylogenomics, conventional methods typically rely on aligning marker genes, a process that becomes computationally prohibitive for large datasets. In contrast, SANS serif leverages whole-genome information by identifying common sequence segments shared between subsets of genomes, which serve as markers of phylogenetic proximity.

The input consists of one sequence file per genome or assembly. From each file, k-mers are extracted and encoded into bit vectors using a 2-bit representation per nucleotide. These encoded k-mers are then stored in a hash table for efficient access.

For each k-mer, another bit vector is stored to encode its presence or absence; a one (or zero) at position i indicates its presence (resp. absence) in the i-th input file. Presence/absence patterns are combined into splits and stored in a second hash table together with two counts: the number of k-mers having that pattern and the number of k-mers having the complementary pattern.

Finally, both counts are combined to an overall weight per split (e.g., using the geometric mean). Splits can be filtered and visualized as a network. The whole process is illustrated in Figure 3.5.

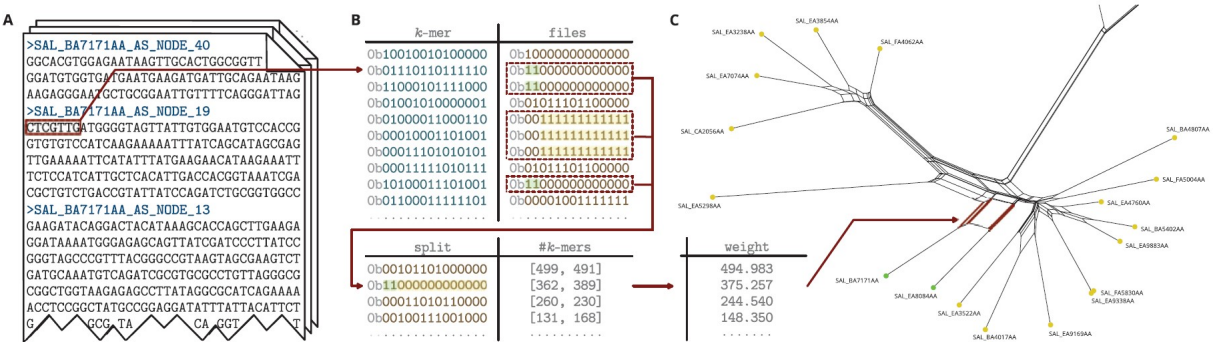


Figure 3.5: SANS serif methodology and evaluation.

By operating directly on k-mers extracted from the input sequences, SANS serif circumvents the need for an exhaustive, quadratic number of pairwise sequence comparisons. This pangenomic approach markedly improves both runtime and memory efficiency.

A key innovation of SANS serif is its efficient data processing strategy. Whereas the previous implementation of SANS [19] employed a trie-based data structure to store splits, resulting in substantial memory overhead for large numbers of genomes, the new version replaces this with two hash tables that store bit vectors. This redesign reduces both runtime and memory usage to approximately 20% of the original implementation, without compromising accuracy.

The output of SANS serif is a list of phylogenetic splits. This split-based representation is particularly advantageous for capturing ambiguous phylogenetic signals, such as those arising from horizontal gene transfers, which might be overlooked by conventional tree-based methods.

Chapter 4

Experimental Setup

This chapter provides a detailed description of the experimental setup adopted in this study. First, the datasets used in the experiments are introduced, followed by a comprehensive explanation of read generation using Mason2. Next, the chapter outlines the overall procedure and the configuration of each software, along with the definition and computation of the Robinson-Foulds metric for comparing phylogenetic trees. All programs and analyses are executed on the computing cluster provided by the Department of Information Engineering (DEI) at the University of Padova. The entire process is summarized in Figure 4.1, which offers a schematic overview of the pipeline.

4.1 Datasets

Three different datasets are used in this project.
The first dataset, referred to as the "*Shigella/E. coli*" dataset, consists of 27 complete genome sequences of E. coli/Shigella strains, obtained from [20].
The second dataset, referred to as the "*E. coli*" dataset, includes 29 assembled E. coli/Shigella strains, originally compiled from [21]. These two datasets contain some overlapping sequences, yet they remain distinct collections.
The third dataset is a *Simulated* set of 33 complete genomes generated using EvolSimulator [22], following an approach similar to [20]. This dataset is prepared with varying levels of horizontal gene transfer (HGT), measured by the average number of HGT events per iteration; however, for this study, analyses are conducted only on sequences with zero HGT events per iteration.
For each dataset, a reference phylogenetic tree is available, which serves as a benchmark for evaluating the trees produced by phyBWT2, Mash, and SANS serif. These reference trees, along with the datasets, are publicly accessible on the AFproject website under the Genome-based Phylogeny and Horizontal Gene Transfer sections. For more information, see [23].

4.2 Mason2: A Read Simulator for Illumina Sequencing and Read Simulation

Mason2 is a software for simulating Illumina, 454, and Sanger reads. It enables controlled simulations by incorporating position-specific error rates and base quality distributions, making it a valuable tool for benchmarking bioinformatics algorithms [24].

Mason2 is used to simulate Illumina reads from complete genomic sequences of various species. In this way, sequencing errors, depth of coverage, and read length are precisely defined. The simulated reads mimic real sequencing data while preserving the exact origin of each read within the reference genome.

For Illumina sequencing, Mason2 follows a model in which sequencing occurs in cycles, during which bases are incorporated one at a time. The sequencing process is subject to errors, including substitutions, insertions, and deletions, which occur with position-dependent probabilities. The probability of errors increases toward the end of the read due to signal decay and phase-shifting effects.

The error model in Mason2 is defined by a set of key properties. The read length, denoted as ℓ, remains constant throughout the simulation. Errors are modeled with position-dependent probabilities, where mismatches occur with probability $p_m(i)$, insertions with probability p_i, and deletions with probability p_d. In the case of a mismatch, the replacing base is chosen randomly from the three remaining nucleotides, excluding the replaced base. The probability of mismatches follows an empirical curve that explicitly defines how substitution rates vary along the length of the read.

These features allow Mason2 to generate realistic sequencing datasets, making it a widely used software for evaluating genome assembly, mapping, and variant calling pipelines. The ability to introduce controlled levels of sequencing noise enables rigorous testing of bioinformatics algorithms under different sequencing conditions.

In this project, Mason2 is used to simulate reads for each genome dataset. For each dataset, four separate sets of unassembled reads corresponding to different levels of coverage are generated: 10×, 30×, 50× and 100×. Each unassembled read produced has a fixed length of 150 base pairs. This simulation step reproduces realistic sequencing conditions and provides the raw data needed for downstream analysis.

4.3 Compression using USTAR

The unassembled reads produced with Mason2 are subsequently processed using USTAR to generate compressed datasets. USTAR is executed on each coverage dataset with varying k-mer

sizes, specifically with $k = 15$, $k = 21$, $k = 31$, and $k = 41$. These different parameter settings enable an evaluation of the effect of k-mer size on compression performance. The outcomes of this compression step, including measures such as the cumulative length of the compressed k-mer sets, are discussed in Chapter 5.

4.4 Phylogenetic Tree Reconstruction

This section describes the software used to generate phylogenetic trees from the simulated and compressed read datasets. The configuration and specific parameters for each program are detailed in the following subsections.

4.4.1 phyBWT2

phyBWT2 reconstructs phylogenetic trees in an alignment-, assembly-, and reference-free manner. The default parameters are used for all datasets: a k-mer size $k = 16$ and a threshold parameter $\tau = 0.6$. The software is available in two versions, one optimized for short reads (less than 250 bp) and one for long reads; the long-read version is employed for all analyses. Notably, although both versions ultimately yield the same tree topology, the long-read version was selected for both uncompressed and compressed datasets to ensure a fair comparison and to take advantage of its efficient preprocessing of input data. phyBWT2 accepts as input either the raw reads simulated by Mason2 or the compressed reads produced by USTAR, and it outputs a phylogenetic tree in Newick format, which is subsequently used for comparison with the corresponding reference tree.

4.4.2 Mash

Mash estimates genomic distances using MinHash sketches. For the analyses, Mash was performed with a k-mer dimension of $k = 21$ and a sketch dimension $s = 50000$ on the *E.coli* and *Shigella/E.coli* datasets. While for the dataset *Simulated* the parameter for the number of sketches was increased to $s = 500000$. The software first executes the `sketch` function to convert each input dataset into a compact representation. Then, the `dist` function is used to compute pairwise distances between all species in the dataset. These pairwise distances are subsequently used to construct a distance matrix, from which a phylogenetic tree is inferred. To achieve this, the Neighbor Joining algorithm is applied, as implemented in the Python library `dendropy`. Specifically, the method `dendropy.PhylogeneticDistanceMatrix.from_csv` is used to load the formatted distance matrix, and the `nj_tree()` method is then invoked to generate a tree in Newick format.

4.4.3 SANS serif

SANS serif performs alignment-free phylogeny estimation based on whole-genome splits. The k-mer size is set to $k = 21$. A notable characteristic of SANS serif is that its input requires a 'file of files', i.e. a file listing multiple read files, to allow processing of multiple sequences simultaneously. Prior to execution, this file is generated to include all the necessary read files. SANS serif produces a phylogenetic tree in Newick format, which is subsequently used for comparative analysis.

4.5 Phylogenetic Trees and Robinson-Foulds Metric

As reported in [25], a phylogenetic tree is a graphical representation of the evolutionary relationships among a set of taxa. In such a tree, the leaves, or terminal nodes, correspond to extant species or sequences, while the internal nodes denote hypothetical common ancestors. In a rooted tree, the root represents the most distant common ancestor in evolutionary time, and the branching pattern reflects the successive divergence of lineages. Unrooted trees, by contrast, illustrate relationships without a designated ancestral root and without an explicit temporal direction. Phylogenetic trees are fundamental in evolutionary biology and are constructed using various methods such as distance-based approaches (e.g., Neighbor Joining) and character-based approaches (e.g., Maximum Parsimony and Maximum Likelihood).

The Robinson-Foulds (RF) distance is a widely used metric for comparing the topologies of phylogenetic trees. Originally proposed by Robinson and Foulds [26], the RF distance quantifies the difference between two trees by counting the number of splits, i.e. bipartitions, that differ between them.

Given two phylogenetic trees T_1 and T_2 defined on the same set of species S, each tree induces a set of splits, denoted by $\Sigma(T_1)$ and $\Sigma(T_2)$. A split corresponds to the bipartition of S resulting from the removal of an edge from the tree. The RF distance between T_1 and T_2 is defined as:

$$d_{RF}(T_1, T_2) = |\Sigma(T_1) \setminus \Sigma(T_2)| + |\Sigma(T_2) \setminus \Sigma(T_1)|$$

In other words, the RF distance is the total number of splits that are present in one tree but absent in the other.

For fully resolved binary trees with n leaves, the maximum possible RF distance is $2(n-3)$. Given this, the normalized RF distance can be defined as:

$$d_{nRF}(T_1, T_2) = \frac{d_{RF}(T_1, T_2)}{2(n-3)}$$

which ranges from 0 to 1, representing identical trees or entirely distinct trees, respectively. This metric provides an objective means of assessing topological similarity between different phylogenetic reconstructions and is helpful in comparing trees produced by diverse methods. For each phylogenetic tree generated by phyBWT2, Mash, and SANS serif, the Robinson-Foulds (RF) distance is calculated to assess the topological similarity with the reference tree. This computation is performed using the Python library `dendropy` [27], specifically by employing the `dendropy.treecompare.symmetric_difference` method. Detailed results of these comparisons are provided in the next chapter.

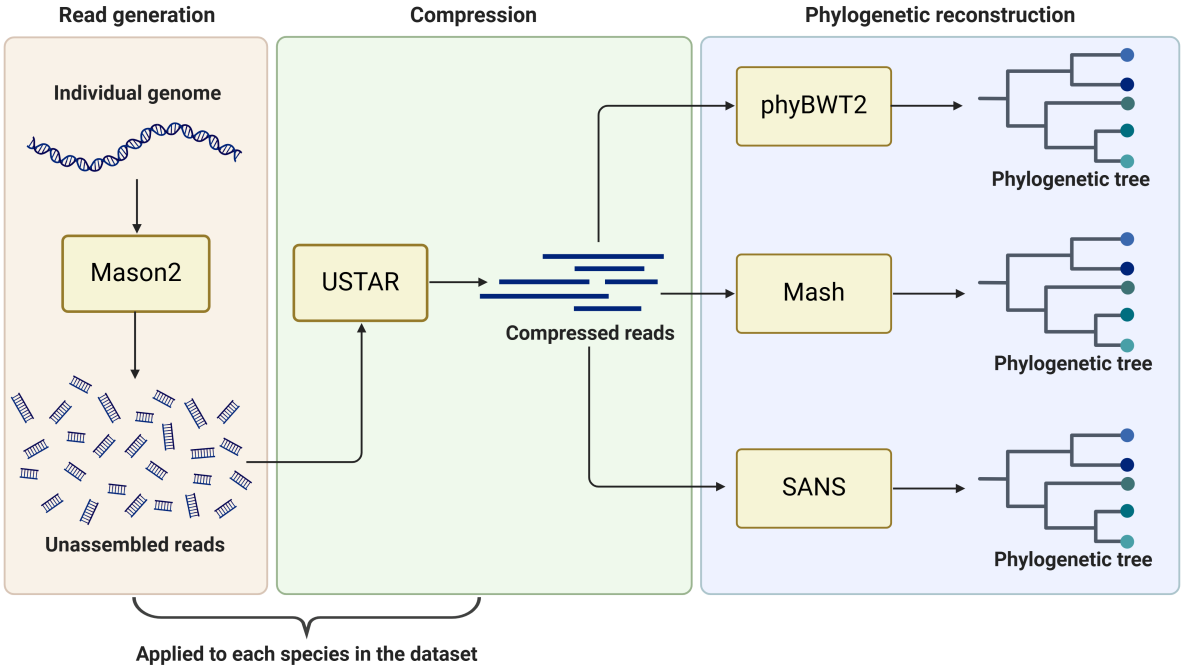


Figure 4.1: Simplified schema of the process, starting from the full genomes to the phylogenetic trees produced by the programs.

Chapter 5

Results

This chapter presents all the results obtained, starting with the compression of the datasets using USTAR with different values of k-mer, followed by performance evaluation in terms of execution time and memory usage. In addition, it includes the comparison between the obtained phylogenetic trees and the reference trees, reporting the RF distance values.

5.1 Dataset Compression

This section presents the dataset sizes before and after compression using USTAR with different k-mer lengths. Results are reported for the three datasets: *E. coli*, *Shigella*/*E. coli*, and *Simulated*. For each dataset, the tables provide the original size, the compressed size, and the corresponding compression ratio across different sequencing coverages.

5.1.1 E.coli Dataset

Tables showing the size of the *E.coli* dataset before and after compression by USTAR for different values of k and different sequencing coverages are presented here.
To better visualize the impact of compression, Figure 5.1 compares the original dataset size with the compressed versions, while Figure 5.2 illustrates the compression ratio trends across different coverages and k-mer lengths.

Coverage	Original Size	Compressed Size	Compression Ratio
10x	1.5 GB	301 MB	4.98
30x	4.5 GB	588 MB	7.65
50x	7.5 GB	861 MB	8.71
100x	15 GB	1.5 GB	10.00

Table 5.1: *E.coli* Dataset sizes before and after compression using USTAR with k=15 for different coverages, with compression ratios.

Coverage	Original Size	Compressed Size	Compression Ratio
10x	1.5 GB	318 MB	4.72
30x	4.5 GB	672 MB	6.70
50x	7.5 GB	1003 MB	7.50
100x	15 GB	1.8 GB	8.33

Table 5.2: *E.coli* Dataset sizes before and after compression using USTAR with k=21 for different coverages, with compression ratios.

Coverage	Original Size	Compressed Size	Compression Ratio
10x	1.5 GB	384 MB	3.91
30x	4.5 GB	866 MB	5.20
50x	7.5 GB	1.3 GB	5.77
100x	15 GB	2.4 GB	6.25

Table 5.3: *E.coli* Dataset sizes before and after compression using USTAR with k=31 for different coverages, with compression ratios.

Coverage	Original Size	Compressed Size	Compression Ratio
10x	1.5 GB	441 MB	3.40
30x	4.5 GB	1.1 GB	4.09
50x	7.5 GB	1.6 GB	4.68
100x	15 GB	2.9 GB	5.17

Table 5.4: *E.coli* Dataset sizes before and after compression using USTAR with k=41 for different coverages, with compression ratios.

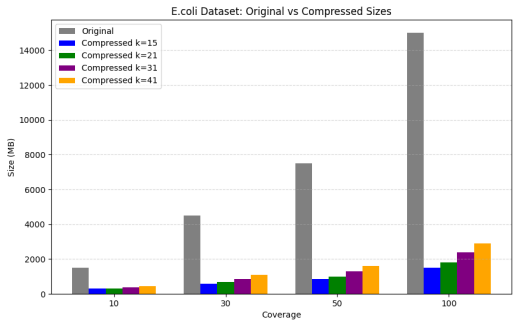


Figure 5.1: Comparison of the original and compressed dataset sizes for *E.coli* at different coverage values and k-mer sizes.

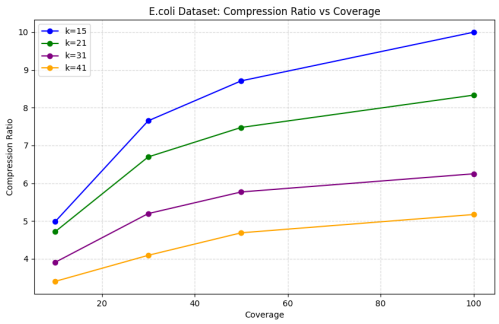


Figure 5.2: Compression ratio trends for *E.coli* dataset at different coverage values and k-mer sizes.

5.1.2 Shigella/E.coli Dataset

Tables showing the size of the *Shigella/E.coli* dataset before and after compression by USTAR for different values of k and different sequencing coverages are presented here.
To better visualize the impact of compression, Figure 5.3 compares the original dataset size with the compressed versions, while Figure 5.4 illustrates the compression ratio trends across different coverages and k-mer lengths.

Coverage	Original Size	Compressed Size	Compression Ratio
10x	1.4 GB	281 MB	4.98
30x	4.2 GB	549 MB	7.65
50x	7.0 GB	803 MB	8.72
100x	14.0 GB	1.4 GB	10.00

Table 5.5: *Shigella/E.coli* Dataset sizes before and after compression using USTAR with k=15 for different coverages, with compression ratios.

Coverage	Original Size	Compressed Size	Compression Ratio
10x	1.4 GB	297 MB	4.71
30x	4.2 GB	627 MB	6.70
50x	7.0 GB	935 MB	7.49
100x	14.0 GB	1.6 GB	8.75

Table 5.6: *Shigella/E.coli* Dataset sizes before and after compression using USTAR with k=21 for different coverages, with compression ratios.

Coverage	Original Size	Compressed Size	Compression Ratio
10x	1.4 GB	358 MB	3.91
30x	4.2 GB	808 MB	5.20
50x	7.0 GB	1.3 GB	5.38
100x	14.0 GB	2.2 GB	6.36

Table 5.7: *Shigella/E.coli* Dataset sizes before and after compression using USTAR with k=31 for different coverages, with compression ratios.

Coverage	Original Size	Compressed Size	Compression Ratio
10x	1.4 GB	411 MB	3.41
30x	4.2 GB	966 MB	4.35
50x	7.0 GB	1.5 GB	4.67
100x	14.0 GB	2.7 GB	5.19

Table 5.8: *Shigella/E.coli* Dataset sizes before and after compression using USTAR with k=41 for different coverages, with compression ratios.

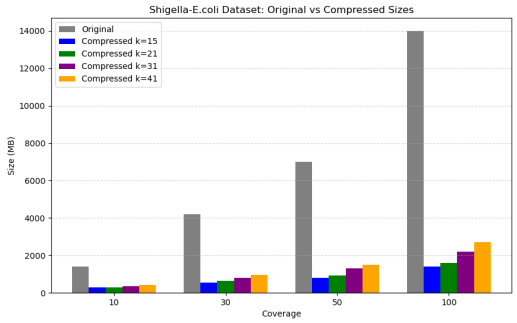


Figure 5.3: Comparison of the original and compressed dataset sizes for *Shigella/E.coli* at different coverage values and k-mer sizes.

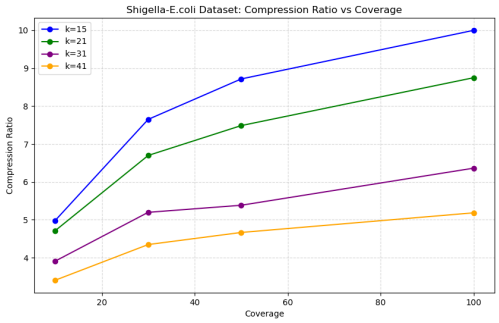


Figure 5.4: Compression ratio trends for *Shigella/E.coli* at different coverage values and k-mer sizes.

5.1.3 Simulated Dataset

Tables showing the size of the *Simulated* dataset before and after compression by USTAR for different values of k and different sequencing coverages are presented here.
To better visualize the impact of compression, Figure 5.5 compares the original dataset size with the compressed versions, while Figure 5.6 illustrates the compression ratio trends across different coverages and k-mer lengths.

Coverage	Original Size	Compressed Size	Compression Ratio
10x	782 MB	134 MB	5.84
30x	2.3 GB	277 MB	8.30
50x	3.9 GB	409 MB	9.54
100x	7.8 GB	702 MB	11.11

Table 5.9: *Simulated* Dataset sizes before and after compression using USTAR with k=15 for different coverages, with compression ratios.

Coverage	Original Size	Compressed Size	Compression Ratio
10x	782 MB	154 MB	5.08
30x	2.3 GB	337 MB	6.82
50x	3.9 GB	504 MB	7.74
100x	7.8 GB	871 MB	8.96

Table 5.10: *Simulated* Dataset sizes before and after compression using USTAR with k=21 for different coverages, with compression ratios.

Coverage	Original Size	Compressed Size	Compression Ratio
10x	782 MB	187 MB	4.18
30x	2.3 GB	436 MB	5.28
50x	3.9 GB	667 MB	5.85
100x	7.8 GB	1.2 GB	6.50

Table 5.11: *Simulated* Dataset sizes before and after compression using USTAR with k=31 for different coverages, with compression ratios.

Coverage	Original Size	Compressed Size	Compression Ratio
10x	782 MB	216 MB	3.62
30x	2.3 GB	523 MB	4.40
50x	3.9 GB	811 MB	4.81
100x	7.8 GB	1.5 GB	5.20

Table 5.12: *Simulated* Dataset sizes before and after compression using USTAR with k=41 for different coverages, with compression ratios.

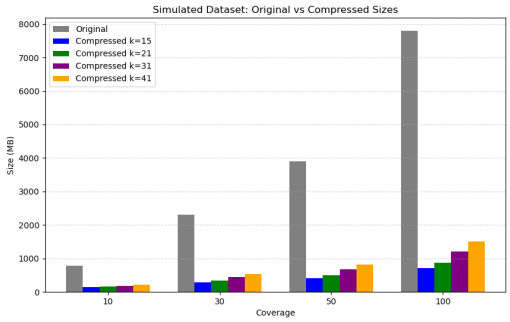


Figure 5.5: Comparison of the original and compressed dataset sizes for *Simulated* at different coverage values and k-mer sizes.

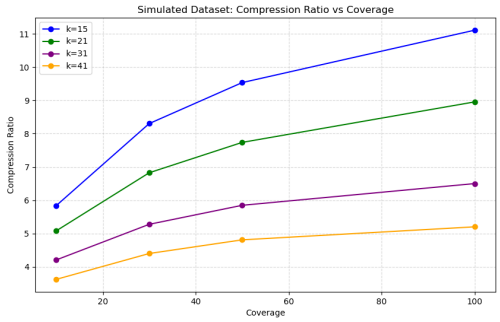


Figure 5.6: Compression ratio trends for *Simulated* at different coverage values and k-mer sizes.

5.2 Execution Times and Memory Usage

This section presents the performance metrics in terms of execution time and memory usage for each tool. Resource usage was recorded using the Unix command `time -v`, which reports detailed statistics; in the tables below, the execution times correspond to the wall-clock time, and the memory usage values represent the maximum resident size. These metrics provide an objective measure of the computational efficiency of each instrument when processing the three datasets with various sequencing coverages and different degrees of compression by USTAR.

5.2.1 E.coli Dataset

A comparative analysis of the execution time and memory usage of the phyBWT2, Mash and SANS methods applied to the *E.coli* dataset processed with USTAR for different k values is presented. Through tables and graphs, the results obtained with original and compressed data are compared, showing the advantages in terms of computational efficiency.

Execution Times

The following tables report the execution times for phyBWT2, Mash, and SANS on the *E.coli* dataset processed with USTAR using different k-mer values. For each k value, both the original and the compressed datasets are compared.

Dataset	phyBWT2	Ratio	Mash	Ratio	SANS	Ratio
Original Dataset 10x	00:30:26	4.58	00:01:22	4.56	00:02:30	2.73
Compressed Dataset 10x	00:06:39		00:00:18		00:00:55	
Original Dataset 30x	01:44:40	7.26	00:03:49	8.48	00:07:04	4.98
Compressed Dataset 30x	00:14:25		00:00:27		00:01:25	
Original Dataset 50x	03:17:21	8.76	00:06:21	10.58	00:12:10	6.08
Compressed Dataset 50x	00:22:31		00:00:36		00:02:00	
Original Dataset 100x	08:31:44	10.92	00:11:41	12.96	00:23:20	7.75
Compressed Dataset 100x	00:46:52		00:00:54		00:03:01	

Table 5.13: Execution times (hh:mm:ss) for phyBWT2, Mash, and SANS on the *E.coli* dataset using USTAR with $k = 15$, comparing original and compressed datasets.

Dataset	phyBWT2	Ratio	Mash	Ratio	SANS	Ratio
Original Dataset 10x	00:29:32	4.03	00:01:27	3.95	00:03:02	2.80
Compressed Dataset 10x	00:07:20		00:00:22		00:01:05	
Original Dataset 30x	01:48:15	5.85	00:04:13	6.49	00:09:43	3.51
Compressed Dataset 30x	00:18:29		00:00:39		00:02:46	
Original Dataset 50x	03:10:34	7.67	00:06:51	7.74	00:12:51	3.91
Compressed Dataset 50x	00:24:49		00:00:53		00:03:17	
Original Dataset 100x	07:58:28	9.10	00:14:30	9.56	00:51:37	4.53
Compressed Dataset 100x	00:52:35		00:01:31		00:11:24	

Table 5.14: Execution times (hh:mm:ss) for phyBWT2, Mash, and SANS on the *E.coli* dataset using USTAR with $k = 21$, comparing original and compressed datasets.

Dataset	phyBWT2	Ratio	Mash	Ratio	SANS	Ratio
Original Dataset 10x	00:29:50	3.37	00:01:28	3.26	00:02:42	2.49
Compressed Dataset 10x	00:08:52		00:00:27		00:01:05	
Original Dataset 30x	01:38:20	4.65	00:04:10	4.81	00:07:17	2.97
Compressed Dataset 30x	00:21:09		00:00:52		00:02:27	
Original Dataset 50x	03:24:07	5.02	00:07:03	5.64	00:13:28	3.28
Compressed Dataset 50x	00:40:42		00:01:15		00:04:06	
Original Dataset 100x	06:53:01	6.24	00:13:42	6.00	00:29:30	4.81
Compressed Dataset 100x	01:06:15		00:02:17		00:06:08	

Table 5.15: Execution times (hh:mm:ss) for phyBWT2, Mash, and SANS on the *E.coli* dataset using USTAR with $k = 31$, comparing original and compressed datasets.

Dataset	phyBWT2	Ratio	Mash	Ratio	SANS	Ratio
Original Dataset 10x	00:35:02	3.50	00:01:31	2.94	00:03:15	2.38
Compressed Dataset 10x	00:10:00		00:00:31		00:01:22	
Original Dataset 30x	01:45:51	3.81	00:03:49	3.95	00:09:47	2.58
Compressed Dataset 30x	00:27:47		00:00:58		00:03:47	
Original Dataset 50x	03:13:48	4.13	00:06:42	4.37	00:18:05	2.76
Compressed Dataset 50x	00:46:56		00:01:32		00:06:33	
Original Dataset 100x	06:28:59	5.10	00:12:46	4.94	00:34:32	3.38
Compressed Dataset 100x	01:16:13		00:02:35		00:10:14	

Table 5.16: Execution times (hh:mm:ss) for phyBWT2, Mash, and SANS on the *E.coli* dataset using USTAR with $k = 41$, comparing original and compressed datasets.

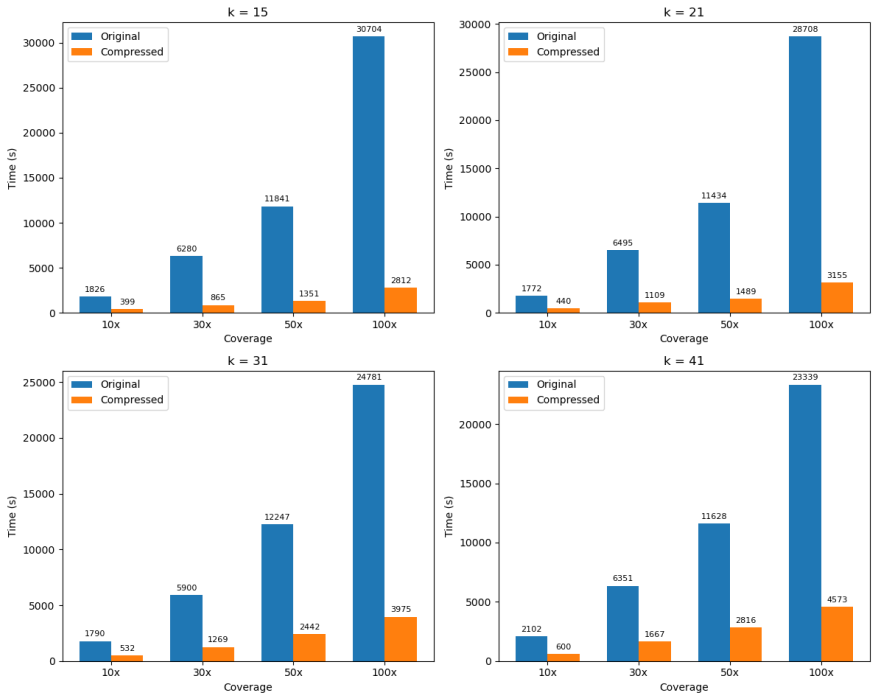


Figure 5.7: Execution times for phyBWT2 on the *E.coli* dataset, comparing original and compressed data across different coverage and k-mer values.

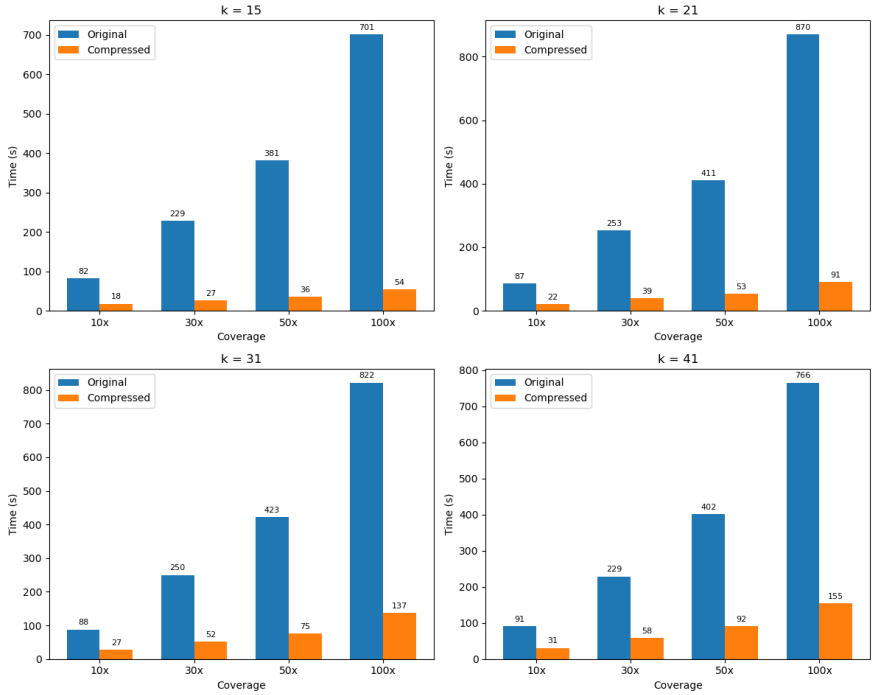


Figure 5.8: Execution times for Mash on the *E.coli* dataset, comparing original and compressed data across different coverage and k-mer values.

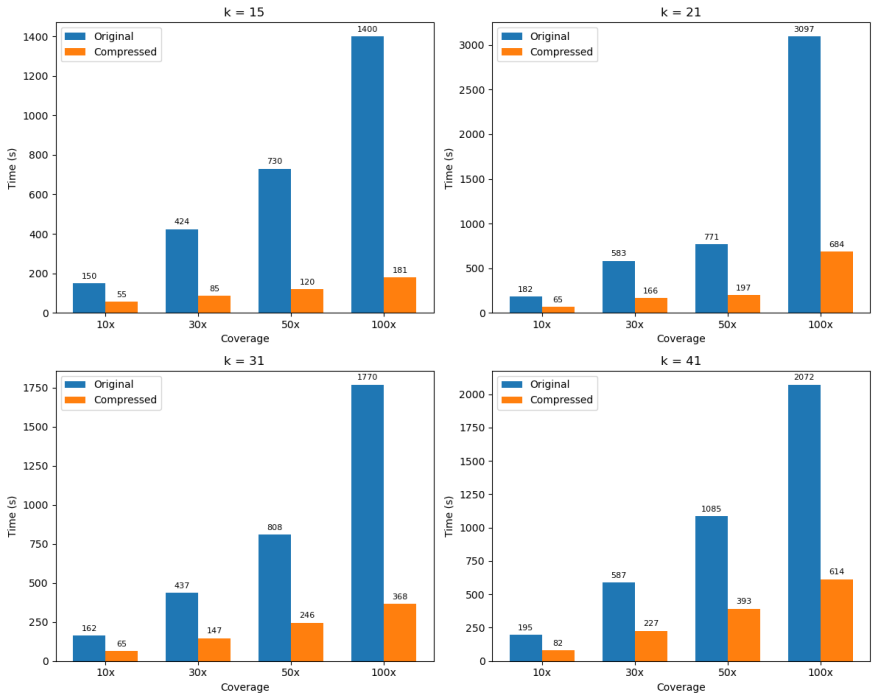


Figure 5.9: Execution times for SANS on the *E.coli* dataset, comparing original and compressed data across different coverage and k-mer values.

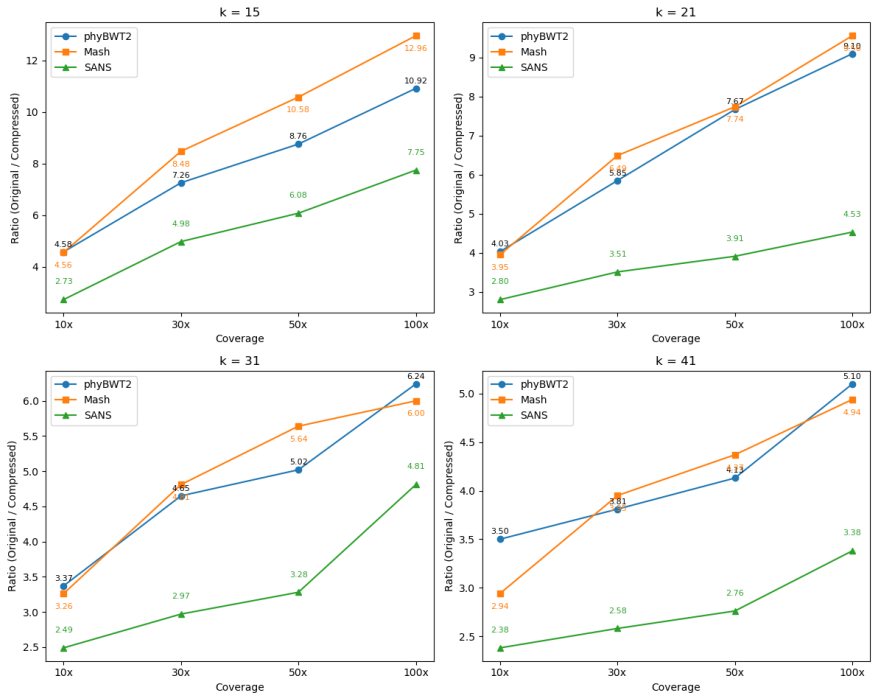


Figure 5.10: Execution time ratios for phyBWT2, Mash, and SANS on the *E.coli* dataset across different coverage and k-mer values.

Memory Usage

The following tables report the maximum memory usage for phyBWT2, Mash, and SANS on the *E.coli* dataset processed with USTAR using different k-mer values.

Dataset	phyBWT2	Mash	SANS
Original Dataset 10x	58.62	0.03	2.10
Compressed Dataset 10x	14.63	0.25	2.08
Original Dataset 30x	175.85	0.03	5.00
Compressed Dataset 30x	29.24	0.57	4.05
Original Dataset 50x	293.07	0.03	7.05
Compressed Dataset 50x	42.97	0.87	7.04
Original Dataset 100x	586.14	0.03	11.07
Compressed Dataset 100x	74.81	1.57	8.75

Table 5.17: Maximum memory usage (in GB) for phyBWT2, Mash, and SANS on the *E.coli* dataset using USTAR with $k = 15$, comparing original and compressed datasets.

Dataset	phyBWT2	Mash	SANS
Original Dataset 10x	58.62	0.03	2.11
Compressed Dataset 10x	14.27	0.18	2.30
Original Dataset 30x	175.85	0.03	5.05
Compressed Dataset 30x	32.40	0.46	5.24
Original Dataset 50x	293.07	0.03	7.05
Compressed Dataset 50x	48.38	0.71	7.21
Original Dataset 100x	586.14	0.03	11.08
Compressed Dataset 100x	84.27	1.23	11.19

Table 5.18: Maximum memory usage (in GB) for phyBWT2, Mash, and SANS on the *E.coli* dataset using USTAR with $k = 21$, comparing original and compressed datasets.

Dataset	phyBWT2	Mash	SANS
Original Dataset 10x	58.62	0.03	2.09
Compressed Dataset 10x	16.70	0.17	2.21
Original Dataset 30x	175.85	0.03	4.75
Compressed Dataset 30x	38.90	0.43	4.84
Original Dataset 50x	293.07	0.03	6.72
Compressed Dataset 50x	59.50	0.67	6.83
Original Dataset 100x	586.14	0.04	11.06
Compressed Dataset 100x	110.04	1.25	11.17

Table 5.19: Maximum memory usage (in GB) for phyBWT2, Mash, and SANS on the *E.coli* dataset using USTAR with $k = 31$, comparing original and compressed datasets.

Dataset	phyBWT2	Mash	SANS
Original Dataset 10x	58.62	0.03	2.10
Compressed Dataset 10x	20.62	0.18	2.18
Original Dataset 30x	175.85	0.03	4.99
Compressed Dataset 30x	47.91	0.45	5.03
Original Dataset 50x	293.07	0.03	7.07
Compressed Dataset 50x	74.99	0.70	7.08
Original Dataset 100x	586.14	0.03	11.08
Compressed Dataset 100x	134.90	1.26	11.14

Table 5.20: Maximum memory usage (in GB) for phyBWT2, Mash, and SANS on the *E.coli* dataset using USTAR with $k = 41$, comparing original and compressed datasets.

5.2.2 Shigella/E.coli Dataset

A comparative analysis of the execution time and memory usage of the phyBWT2, Mash and SANS methods applied to the *Shigella/E.coli* dataset processed with USTAR for different k values is presented. Through tables and graphs, the results obtained with original and compressed data are compared, showing the advantages in terms of computational efficiency.

Execution Times

The following tables report the execution times for phyBWT2, Mash, and SANS on the *Shigella/E.coli* dataset processed with USTAR using different k-mer values. For each k value, both the original and the compressed datasets are compared.

Dataset	phyBWT2	Ratio	Mash	Ratio	SANS	Ratio
Original Dataset 10x	00:28:33	4.47	00:01:17	4.53	00:03:12	3.10
Compressed Dataset 10x	00:06:23		00:00:17		00:01:02	
Original Dataset 30x	01:34:34	6.87	00:03:39	8.42	00:07:01	5.01
Compressed Dataset 30x	00:13:46		00:00:26		00:01:24	
Original Dataset 50x	02:45:49	8.51	00:06:18	10.22	00:11:55	6.75
Compressed Dataset 50x	00:19:29		00:00:37		00:01:46	
Original Dataset 100x	05:46:46	9.88	00:12:06	12.74	00:23:57	8.21
Compressed Dataset 100x	00:35:06		00:00:57		00:02:55	

Table 5.21: Execution times (hh:mm:ss) for phyBWT2, Mash, and SANS on the *Shigella/E.coli* dataset using USTAR with $k = 15$, comparing original and compressed datasets.

Dataset	phyBWT2	Ratio	Mash	Ratio	SANS	Ratio
Original Dataset 10x	00:28:01	4.00	00:01:19	3.95	00:03:43	2.79
Compressed Dataset 10x	00:07:00		00:00:20		00:01:20	
Original Dataset 30x	01:33:46	5.91	00:03:42	6.53	00:07:29	3.84
Compressed Dataset 30x	00:15:52		00:00:34		00:01:57	
Original Dataset 50x	02:48:43	6.39	00:06:00	7.83	00:11:24	3.91
Compressed Dataset 50x	00:26:24		00:00:46		00:02:55	
Original Dataset 100x	06:16:21	7.90	00:11:56	9.18	00:32:39	4.32
Compressed Dataset 100x	00:47:38		00:01:18		00:07:33	

Table 5.22: Execution times (hh:mm:ss) for phyBWT2, Mash, and SANS on the *Shigella/E.coli* dataset using USTAR with $k = 21$, comparing original and compressed datasets.

Dataset	phyBWT2	Ratio	Mash	Ratio	SANS	Ratio
Original Dataset 10x	00:28:44	3.43	00:01:17	3.21	00:02:33	2.48
Compressed Dataset 10x	00:08:22		00:00:24		00:01:02	
Original Dataset 30x	01:30:40	4.65	00:03:21	4.79	00:06:09	2.83
Compressed Dataset 30x	00:19:29		00:00:42		00:02:10	
Original Dataset 50x	02:44:08	5.70	00:05:50	5.30	00:11:21	3.15
Compressed Dataset 50x	00:28:48		00:01:06		00:03:36	
Original Dataset 100x	06:20:57	6.34	00:12:10	6.04	00:26:56	3.79
Compressed Dataset 100x	01:00:04		00:02:01		00:07:06	

Table 5.23: Execution times (hh:mm:ss) for phyBWT2, Mash, and SANS on the *Shigella/E.coli* dataset using USTAR with $k = 31$, comparing original and compressed datasets.

Dataset	phyBWT2	Ratio	Mash	Ratio	SANS	Ratio
Original Dataset 10x	00:25:24	3.00	00:01:05	2.83	00:02:07	2.15
Compressed Dataset 10x	00:08:28		00:00:23		00:00:59	
Original Dataset 30x	01:39:47	3.88	00:03:42	3.96	00:09:34	2.69
Compressed Dataset 30x	00:25:42		00:00:56		00:03:33	
Original Dataset 50x	07:15:53	4.41	00:06:09	4.34	00:22:37	3.17
Compressed Dataset 50x	01:38:37		00:01:25		00:07:08	
Original Dataset 100x	05:57:24	5.13	00:13:03	4.86	00:37:25	3.49
Compressed Dataset 100x	01:09:38		00:02:41		00:10:43	

Table 5.24: Execution times (hh:mm:ss) for phyBWT2, Mash, and SANS on the *Shigella/E.coli* dataset using USTAR with $k = 41$, comparing original and compressed datasets.

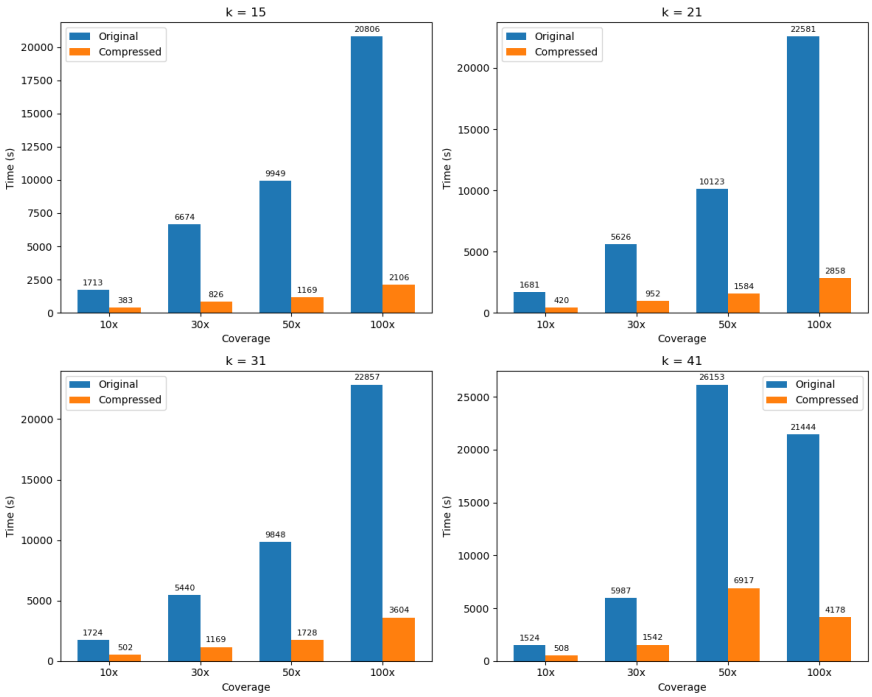


Figure 5.11: Execution times for phyBWT2 on the *Shigella/E.coli* dataset, comparing original and compressed data across different coverage and k-mer values.

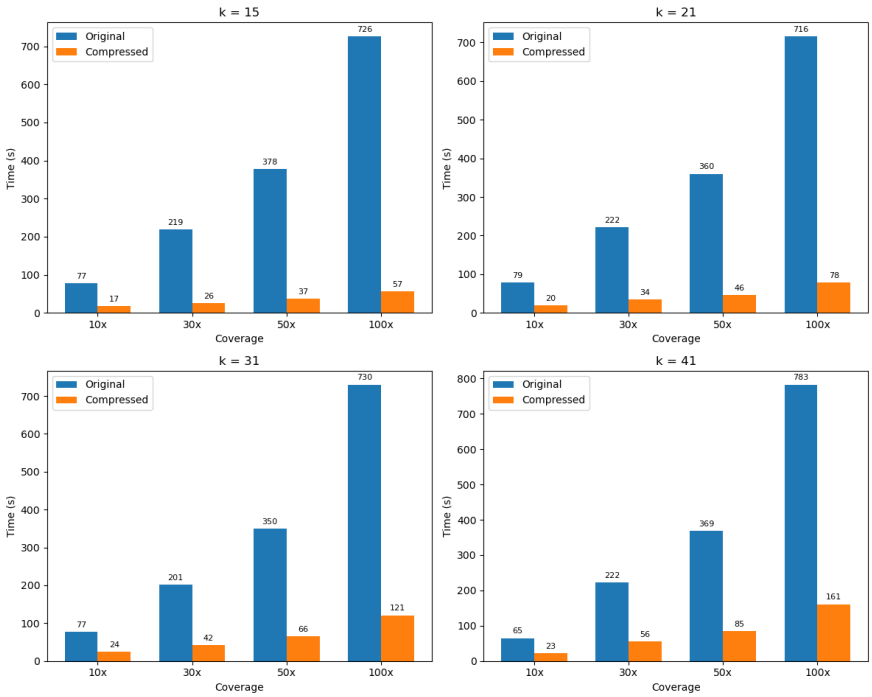


Figure 5.12: Execution times for Mash on the *Shigella/E.coli* dataset, comparing original and compressed data across different coverage and k-mer values.

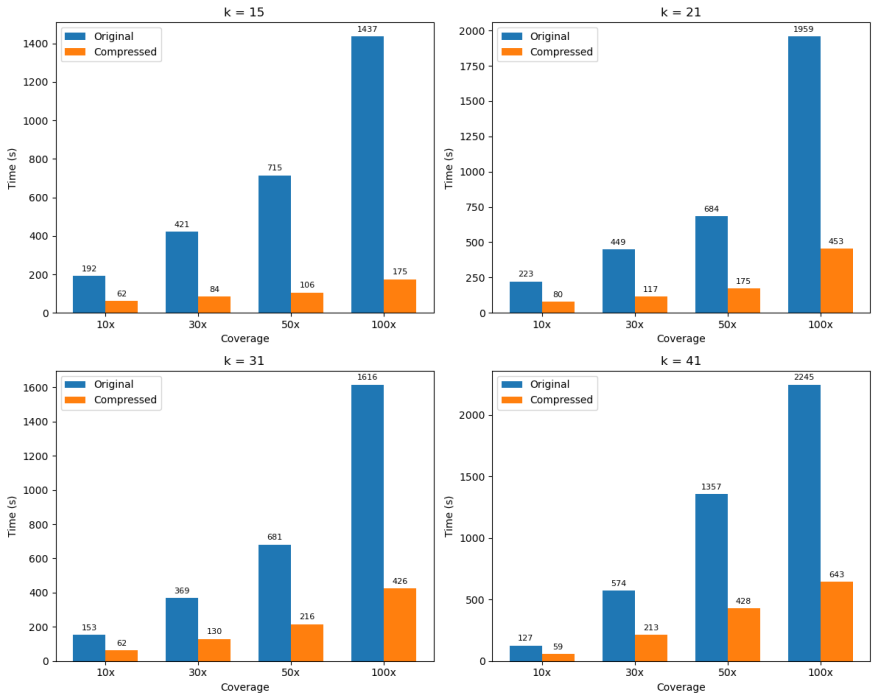


Figure 5.13: Execution times for SANS on the *Shigella/E.coli* dataset, comparing original and compressed data across different coverage and k-mer values.

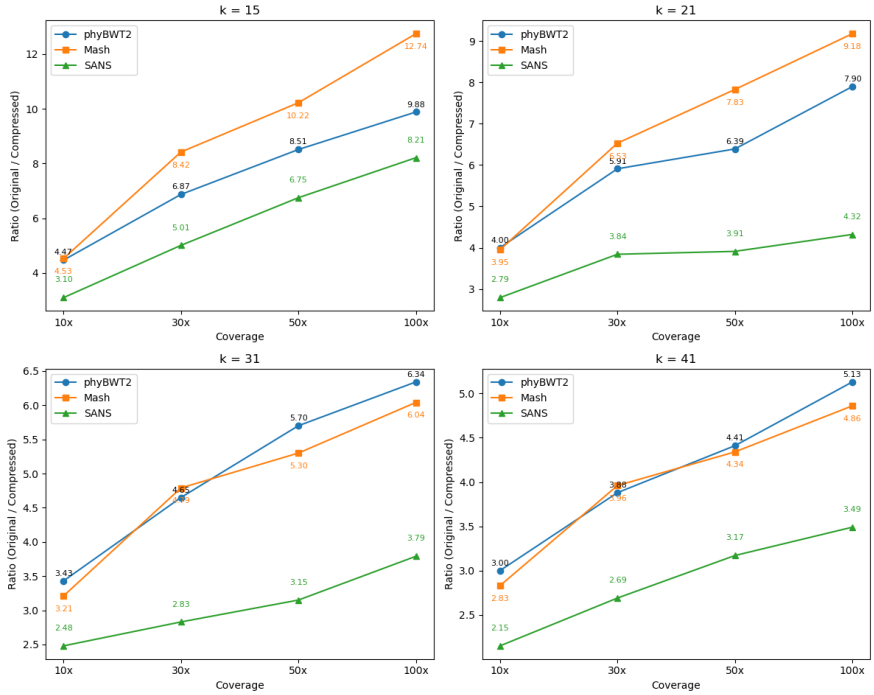


Figure 5.14: Execution time ratios for phyBWT2, Mash, and SANS on the *Shigella/E.coli* dataset across different coverage and k-mer values.

Memory Usage

The following tables report the maximum memory usage for phyBWT2, Mash, and SANS on the *Shigella/E.coli* dataset processed with USTAR using different k-mer values.

Dataset	phyBWT2	Mash	SANS
Original Dataset 10x	54.69	0.03	2.01
Compressed Dataset 10x	13.57	0.23	1.97
Original Dataset 30x	164.06	0.03	4.69
Compressed Dataset 30x	27.10	0.53	3.84
Original Dataset 50x	273.46	0.03	6.72
Compressed Dataset 50x	39.66	0.81	4.93
Original Dataset 100x	546.91	0.03	10.72
Compressed Dataset 100x	70.27	1.47	8.31

Table 5.25: Maximum memory usage (in GB) for phyBWT2, Mash, and SANS on the *Shigella/E.coli* dataset using USTAR with $k = 15$, comparing original and compressed datasets.

Dataset	phyBWT2	Mash	SANS
Original Dataset 10x	54.69	0.03	2.02
Compressed Dataset 10x	13.39	0.17	2.17
Original Dataset 30x	164.08	0.03	4.68
Compressed Dataset 30x	29.31	0.43	4.82
Original Dataset 50x	273.46	0.03	6.72
Compressed Dataset 50x	45.00	0.66	7.25
Original Dataset 100x	546.91	0.03	10.80
Compressed Dataset 100x	78.51	1.15	10.89

Table 5.26: Maximum memory usage (in GB) for phyBWT2, Mash, and SANS on the *Shigella/E.coli* dataset using USTAR with $k = 21$, comparing original and compressed datasets.

Dataset	phyBWT2	Mash	SANS
Original Dataset 10x	54.69	0.03	2.02
Compressed Dataset 10x	16.74	0.17	2.13
Original Dataset 30x	164.08	0.03	4.70
Compressed Dataset 30x	37.78	0.42	4.79
Original Dataset 50x	273.46	0.03	6.75
Compressed Dataset 50x	58.02	0.66	7.08
Original Dataset 100x	546.91	0.03	10.73
Compressed Dataset 100x	103.99	1.17	10.82

Table 5.27: Maximum memory usage (in GB) for phyBWT2, Mash, and SANS on the *Shigella/E.coli* dataset using USTAR with $k = 31$, comparing original and compressed datasets.

Dataset	phyBWT2	Mash	SANS
Original Dataset 10x	54.69	0.03	2.02
Compressed Dataset 10x	18.94	0.17	2.13
Original Dataset 30x	164.08	0.03	4.70
Compressed Dataset 30x	45.03	0.42	4.76
Original Dataset 50x	273.46	0.03	6.73
Compressed Dataset 50x	69.84	0.65	7.03
Original Dataset 100x	546.91	0.03	10.74
Compressed Dataset 100x	127.36	1.18	10.81

Table 5.28: Maximum memory usage (in GB) for phyBWT2, Mash, and SANS on the *Shigella/E.coli* dataset using USTAR with $k = 41$, comparing original and compressed datasets.

5.2.3 Simulated Dataset

A comparative analysis of the execution time and memory usage of the phyBWT2, Mash and SANS methods applied to the *E.coli* dataset processed with USTAR for different k values is presented. Through tables and graphs, the results obtained with original and compressed data are compared, showing the advantages in terms of computational efficiency.

Execution Times

The following tables report the execution times for phyBWT2, Mash, and SANS on the *Simulated* dataset processed with USTAR using different k-mer values. For each k value, both the original and the compressed datasets are compared.

Dataset	phyBWT2	Ratio	Mash	Ratio	SANS	Ratio
Original Dataset 10x	00:14:01	5.07	00:01:34	2.47	00:01:30	3.46
Compressed Dataset 10x	00:02:46		00:00:38		00:00:26	
Original Dataset 30x	00:38:50	7.00	00:03:22	4.12	00:04:35	6.25
Compressed Dataset 30x	00:05:33		00:00:49		00:00:44	
Original Dataset 50x	01:21:41	8.15	00:04:57	5.21	00:08:18	7.90
Compressed Dataset 50x	00:10:01		00:00:57		00:01:03	
Original Dataset 100x	03:07:14	11.85	00:08:45	7.00	00:21:38	8.60
Compressed Dataset 100x	00:15:48		00:01:15		00:02:31	

Table 5.29: Execution times (hh:mm:ss) for phyBWT2, Mash, and SANS on the *Simulated* dataset using USTAR with $k = 15$, comparing original and compressed datasets.

Dataset	phyBWT2	Ratio	Mash	Ratio	SANS	Ratio
Original Dataset 10x	00:50:43	4.49	00:01:19	2.14	00:02:27	3.13
Compressed Dataset 10x	00:11:18		00:00:37		00:00:47	
Original Dataset 30x	00:46:09	5.03	00:02:54	3.48	00:04:13	3.72
Compressed Dataset 30x	00:09:11		00:00:50		00:01:08	
Original Dataset 50x	01:22:39	6.04	00:04:18	4.30	00:08:11	4.31
Compressed Dataset 50x	00:13:42		00:01:00		00:01:54	
Original Dataset 100x	02:56:37	7.21	00:07:41	5.69	00:17:03	4.85
Compressed Dataset 100x	00:24:31		00:01:21		00:03:31	

Table 5.30: Execution times (hh:mm:ss) for phyBWT2, Mash, and SANS on the *Simulated* dataset using USTAR with $k = 21$, comparing original and compressed datasets.

Dataset	phyBWT2	Ratio	Mash	Ratio	SANS	Ratio
Original Dataset 10x	00:17:02	2.99	00:01:54	2.19	00:02:02	2.44
Compressed Dataset 10x	00:05:41		00:00:52		00:00:50	
Original Dataset 30x	00:47:46	4.23	00:03:53	3.07	00:04:19	2.88
Compressed Dataset 30x	00:11:17		00:01:16		00:01:30	
Original Dataset 50x	01:21:38	4.64	00:05:43	3.12	00:07:55	3.34
Compressed Dataset 50x	00:17:35		00:01:50		00:02:22	
Original Dataset 100x	03:05:20	5.74	00:09:44	4.36	00:18:42	3.84
Compressed Dataset 100x	00:32:18		00:02:14		00:04:52	

Table 5.31: Execution times (hh:mm:ss) for phyBWT2, Mash, and SANS on the *Simulated* dataset using USTAR with $k = 31$, comparing original and compressed datasets.

Dataset	phyBWT2	Ratio	Mash	Ratio	SANS	Ratio
Original Dataset 10x	00:16:51	2.36	00:01:50	2.04	00:02:37	2.09
Compressed Dataset 10x	00:07:09		00:00:54		00:01:15	
Original Dataset 30x	00:49:18	3.35	00:03:53	2.64	00:07:22	2.75
Compressed Dataset 30x	00:14:43		00:01:28		00:02:41	
Original Dataset 50x	01:19:23	4.19	00:05:36	3.03	00:10:28	2.79
Compressed Dataset 50x	00:18:58		00:01:51		00:03:45	
Original Dataset 100x	02:52:52	4.72	00:09:43	3.71	00:13:47	3.07
Compressed Dataset 100x	00:36:37		00:02:37		00:04:29	

Table 5.32: Execution times (hh:mm:ss) for phyBWT2, Mash, and SANS on the *Simulated* dataset using USTAR with $k = 41$, comparing original and compressed datasets.

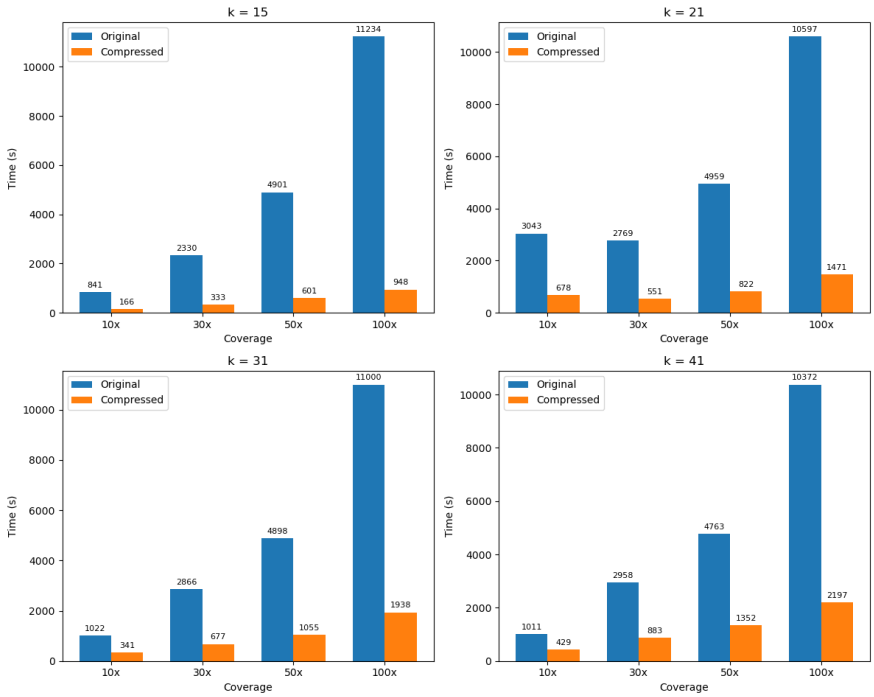


Figure 5.15: Execution times for phyBWT2 on the *Simulated* dataset, comparing original and compressed data across different coverage and k-mer values.

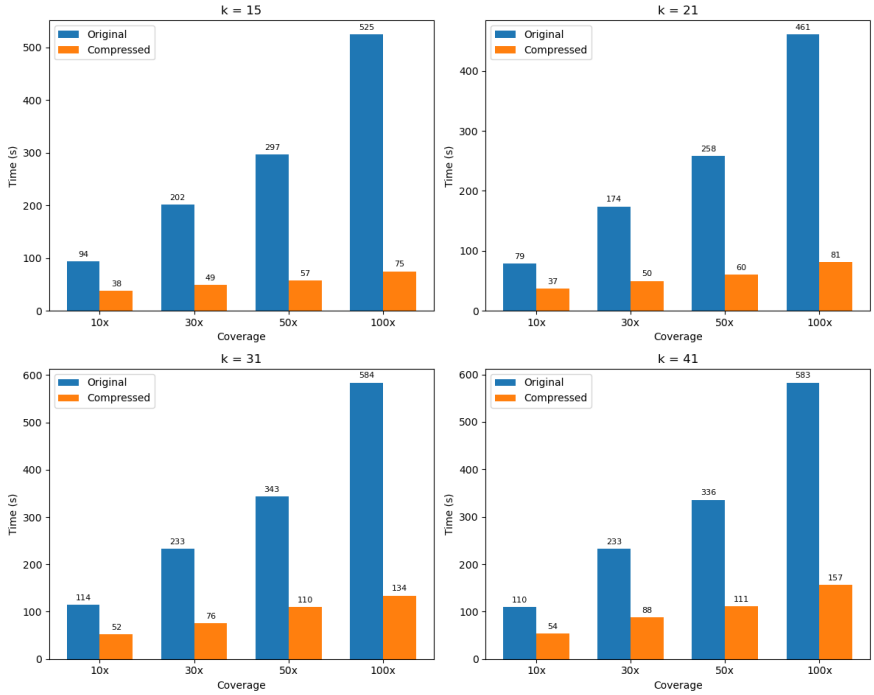


Figure 5.16: Execution times for Mash on the *Simulated* dataset, comparing original and compressed data across different coverage and k-mer values.

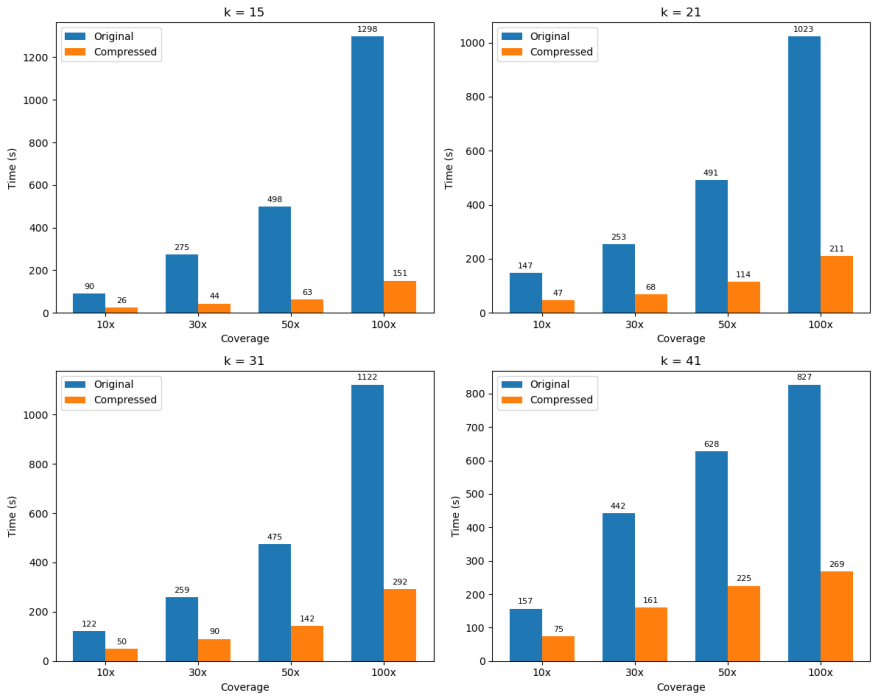


Figure 5.17: Execution times for SANS on the *Simulated* dataset, comparing original and compressed data across different coverage and k-mer values.

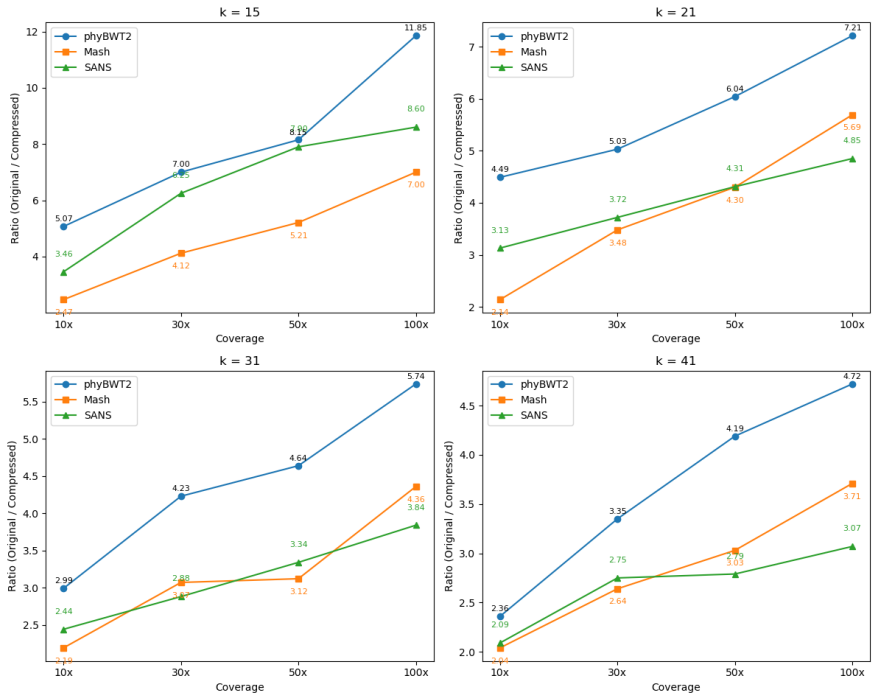


Figure 5.18: Execution time ratios for phyBWT2, Mash, and SANS on the *Simulated* dataset across different coverage and k-mer values.

Memory Usage

The following tables report the maximum memory usage for phyBWT2, Mash, and SANS on the *Simulated* dataset processed with USTAR using different k-mer values.

Dataset	phyBWT2	Mash	SANS
Original Dataset 10x	30.05	0.36	2.00
Compressed Dataset 10x	6.53	0.45	1.64
Original Dataset 30x	90.15	0.36	4.01
Compressed Dataset 30x	13.63	0.60	3.25
Original Dataset 50x	150.26	0.36	6.01
Compressed Dataset 50x	20.22	0.72	3.62
Original Dataset 100x	300.51	0.36	13.29
Compressed Dataset 100x	34.56	0.94	6.09

Table 5.33: Maximum memory usage (in GB) for phyBWT2, Mash, and SANS on the *Simulated* dataset using USTAR with $k = 15$, comparing original and compressed datasets.

Dataset	phyBWT2	Mash	SANS
Original Dataset 10x	30.05	0.36	2.00
Compressed Dataset 10x	7.37	0.44	2.09
Original Dataset 30x	90.15	0.36	4.00
Compressed Dataset 30x	16.13	0.58	4.07
Original Dataset 50x	150.26	0.36	6.01
Compressed Dataset 50x	24.09	0.72	6.16
Original Dataset 100x	300.51	0.36	13.31
Compressed Dataset 100x	41.86	0.62	13.45

Table 5.34: Maximum memory usage (in GB) for phyBWT2, Mash, and SANS on the *Simulated* dataset using USTAR with $k = 21$, comparing original and compressed datasets.

Dataset	phyBWT2	Mash	SANS
Original Dataset 10x	30.05	0.36	2.00
Compressed Dataset 10x	8.76	0.44	2.06
Original Dataset 30x	90.15	0.36	4.01
Compressed Dataset 30x	20.43	0.57	4.05
Original Dataset 50x	150.26	0.36	6.01
Compressed Dataset 50x	31.37	0.71	6.14
Original Dataset 100x	300.51	0.36	13.29
Compressed Dataset 100x	55.70	0.96	13.28

Table 5.35: Maximum memory usage (in GB) for phyBWT2, Mash, and SANS on the *Simulated* dataset using USTAR with $k = 31$, comparing original and compressed datasets.

Dataset	phyBWT2	Mash	SANS
Original Dataset 10x	30.05	0.36	2.00
Compressed Dataset 10x	10.12	0.44	2.05
Original Dataset 30x	90.15	0.36	4.01
Compressed Dataset 30x	24.47	0.57	4.04
Original Dataset 50x	150.26	0.36	6.01
Compressed Dataset 50x	37.66	0.70	6.09
Original Dataset 100x	300.51	0.36	13.29
Compressed Dataset 100x	68.63	0.98	13.24

Table 5.36: Maximum memory usage (in GB) for phyBWT2, Mash, and SANS on the *Simulated* dataset using USTAR with $k = 41$, comparing original and compressed datasets.

5.3 Phylogenetic Tree Comparison

This section presents a comparison of the phylogenetic trees produced by phyBWT2, Mash and SANS serif and the reference trees provided by the AFproject website. The comparison is based on the Robinson-Foulds distance metric, which quantifies the topological differences between the trees. Both raw and normalized RF distances are reported to understand how USTAR compression affects the accuracy of phylogenetic reconstruction. The following subsections present the RF distance results for each dataset, reporting results for both the original datasets and those compressed to different degrees of USTAR compression.

5.3.1 E.coli Dataset

Values of RF distance and normalized RF distance for the *E.coli* dataset, obtained with phyBWT2, Mash and SANS software, are presented. The tables, together with representative graphs, provide a visual overview of the results obtained by performing the analysis on the unassembled reads and the compressed reads using USTAR.

Dataset	phyBWT2	Mash	SANS
Original Dataset 10x	18	12	12
Compressed Dataset 10x	18	14	20
Original Dataset 30x	18	12	12
Compressed Dataset 30x	20	14	36
Original Dataset 50x	20	12	10
Compressed Dataset 50x	22	16	25
Original Dataset 100x	20	14	12
Compressed Dataset 100x	32	18	25
Full Genomes	20	12	14

Table 5.37: RF distance values for phyBWT2, Mash, and SANS on the *E.coli* dataset using USTAR with k=15. Results are reported for both the original and USTAR-compressed datasets.

Dataset	phyBWT2	Mash	SANS
Original Dataset 10x	0.3462	0.2308	0.2308
Compressed Dataset 10x	0.3462	0.2692	0.3846
Original Dataset 30x	0.3462	0.2308	0.2308
Compressed Dataset 30x	0.3846	0.2692	0.6923
Original Dataset 50x	0.3846	0.2308	0.1923
Compressed Dataset 50x	0.4231	0.3077	0.4808
Original Dataset 100x	0.3846	0.2692	0.2308
Compressed Dataset 100x	0.6154	0.3462	0.4808
Full Genomes	0.3846	0.2308	0.2692

Table 5.38: Normalized RF distance values for phyBWT2, Mash, and SANS on the *E.coli* dataset using USTAR with k=15. Comparison includes both the original and USTAR-compressed datasets.

Dataset	phyBWT2	Mash	SANS
Original Dataset 10x	18	12	12
Compressed Dataset 10x	16	12	12
Original Dataset 30x	18	12	12
Compressed Dataset 30x	20	12	12
Original Dataset 50x	20	12	10
Compressed Dataset 50x	18	12	10
Original Dataset 100x	20	14	12
Compressed Dataset 100x	18	14	12
Full Genomes	20	12	14

Table 5.39: RF distance values for phyBWT2, Mash, and SANS on the *E.coli* dataset using USTAR with k=21. Results are reported for both the original and USTAR-compressed datasets.

Dataset	phyBWT2	Mash	SANS
Original Dataset 10x	0.3462	0.2308	0.2308
Compressed Dataset 10x	0.3077	0.2308	0.2308
Original Dataset 30x	0.3462	0.2308	0.2308
Compressed Dataset 30x	0.3846	0.2308	0.2308
Original Dataset 50x	0.3846	0.2308	0.1923
Compressed Dataset 50x	0.3462	0.2308	0.1923
Original Dataset 100x	0.3846	0.2692	0.2308
Compressed Dataset 100x	0.3462	0.2692	0.2308
Full Genomes	0.3846	0.2308	0.2692

Table 5.40: Normalized RF distance values for phyBWT2, Mash, and SANS on the *E.coli* dataset using USTAR with k=21. Comparison includes both the original and USTAR-compressed datasets.

Dataset	phyBWT2	Mash	SANS
Original Dataset 10x	18	12	12
Compressed Dataset 10x	16	12	12
Original Dataset 30x	18	12	12
Compressed Dataset 30x	20	12	12
Original Dataset 50x	20	12	10
Compressed Dataset 50x	18	12	10
Original Dataset 100x	20	14	12
Compressed Dataset 100x	18	14	12
Full Genomes	20	12	14

Table 5.41: RF distance values for phyBWT2, Mash, and SANS on the *E.coli* dataset using USTAR with k=31. Results are reported for both the original and USTAR-compressed datasets.

Dataset	phyBWT2	Mash	SANS
Original Dataset 10x	0.3462	0.2308	0.2308
Compressed Dataset 10x	0.3077	0.2308	0.2308
Original Dataset 30x	0.3462	0.2308	0.2308
Compressed Dataset 30x	0.3846	0.2308	0.2308
Original Dataset 50x	0.3846	0.2308	0.1923
Compressed Dataset 50x	0.3462	0.2308	0.1923
Original Dataset 100x	0.3846	0.2692	0.2308
Compressed Dataset 100x	0.3462	0.2692	0.2308
Full Genomes	0.3846	0.2308	0.2692

Table 5.42: Normalized RF distance values for phyBWT2, Mash, and SANS on the *E.coli* dataset using USTAR with k=31. Comparison includes both the original and USTAR-compressed datasets.

Dataset	phyBWT2	Mash	SANS
Original Dataset 10x	18	12	12
Compressed Dataset 10x	16	12	12
Original Dataset 30x	18	12	12
Compressed Dataset 30x	20	12	12
Original Dataset 50x	20	12	10
Compressed Dataset 50x	18	12	10
Original Dataset 100x	20	14	12
Compressed Dataset 100x	18	14	12
Full Genomes	20	12	14

Table 5.43: RF distance values for phyBWT2, Mash, and SANS on the *E.coli* dataset using USTAR with k=41. Results are reported for both the original and USTAR-compressed datasets.

Dataset	phyBWT2	Mash	SANS
Original Dataset 10x	0.3462	0.2308	0.2308
Compressed Dataset 10x	0.3077	0.2308	0.2308
Original Dataset 30x	0.3462	0.2308	0.2308
Compressed Dataset 30x	0.3846	0.2308	0.2308
Original Dataset 50x	0.3846	0.2308	0.1923
Compressed Dataset 50x	0.3462	0.2308	0.1923
Original Dataset 100x	0.3846	0.2692	0.2308
Compressed Dataset 100x	0.3462	0.2692	0.2308
Full Genomes	0.3846	0.2308	0.2692

Table 5.44: Normalized RF distance values for phyBWT2, Mash, and SANS on the *E.coli* dataset using USTAR with k=41. Comparison includes both the original and USTAR-compressed datasets.

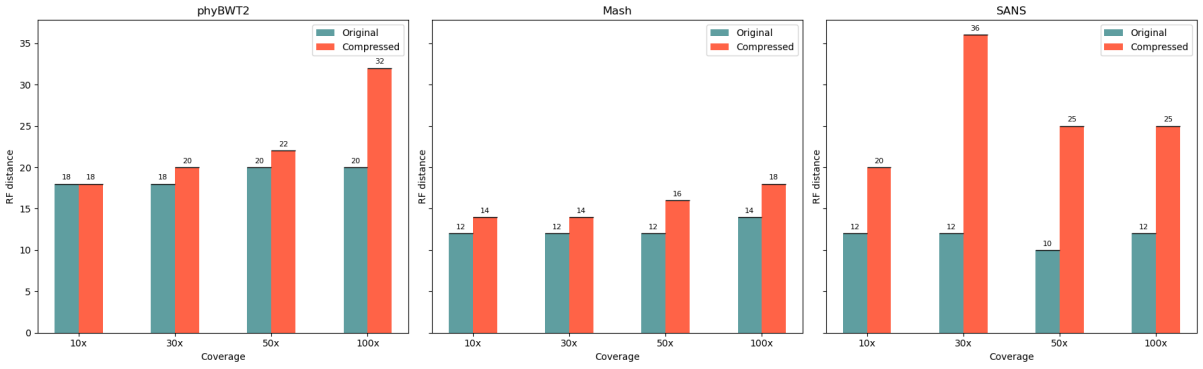


Figure 5.19: Graphical representation of RF distances for phyBWT2, Mash, and SANS on the *E. coli* dataset using USTAR k-mer size = 15.

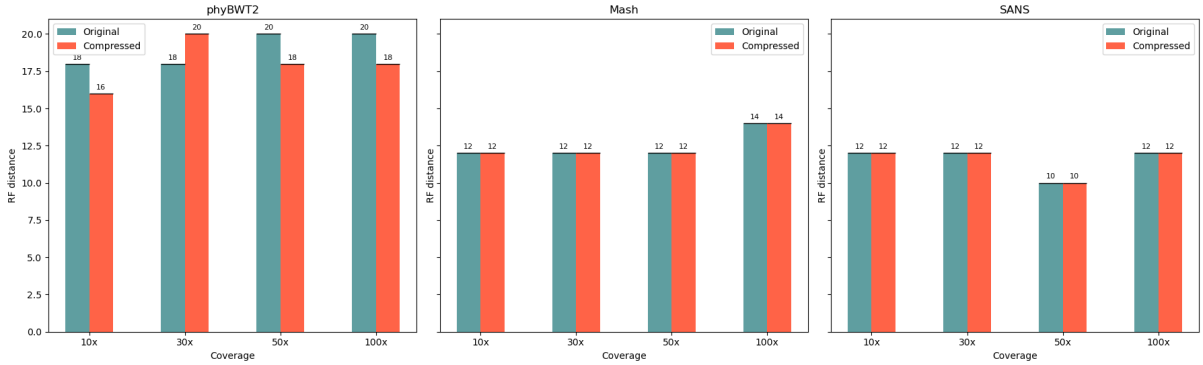


Figure 5.20: Graphical representation of RF distances for phyBWT2, Mash, and SANS on the *E. coli* dataset using USTAR k-mer size = 21.

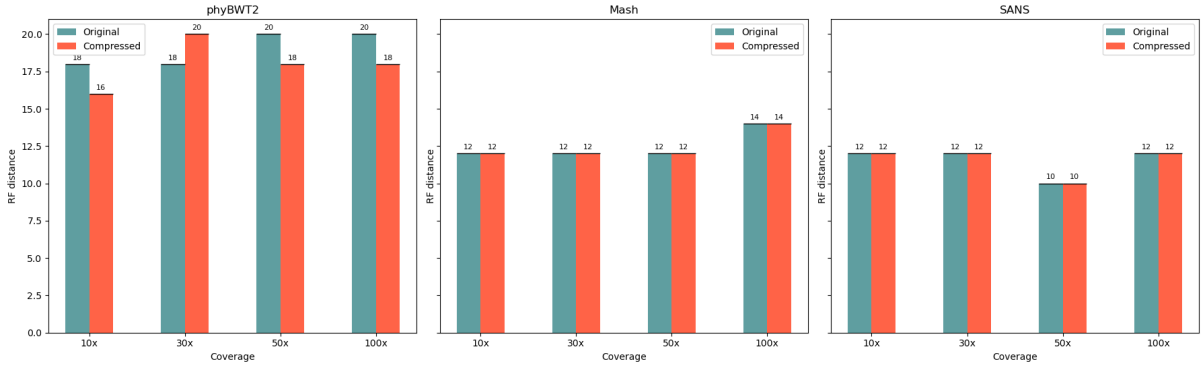


Figure 5.21: Graphical representation of RF distances for phyBWT2, Mash, and SANS on the *E. coli* dataset using USTAR k-mer size = 31.

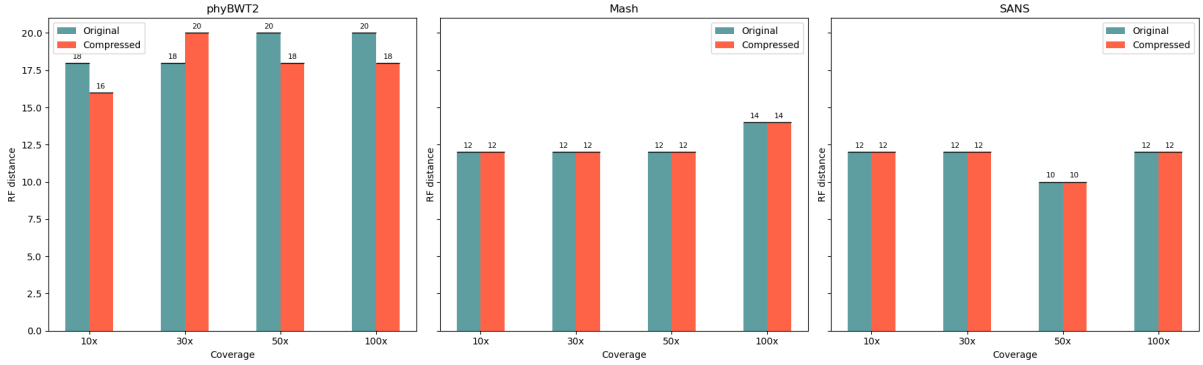


Figure 5.22: Graphical representation of RF distances for phyBWT2, Mash, and SANS on the *E. coli* dataset using USTAR k-mer size = 41.

5.3.2 Shigella/E.coli Dataset

Values of RF distance and normalized RF distance for the *Shigella/E.coli* dataset, obtained with phyBWT2, Mash and SANS software, are presented. The tables, together with representative graphs, provide a visual overview of the results obtained by performing the analysis on the unassembled reads and the compressed reads using USTAR.

Dataset	phyBWT2	Mash	SANS
Original Dataset 10x	16	8	10
Compressed Dataset 10x	12	8	14
Original Dataset 30x	14	12	12
Compressed Dataset 30x	18	10	24
Original Dataset 50x	14	12	16
Compressed Dataset 50x	22	16	20
Original Dataset 100x	16	10	20
Compressed Dataset 100x	18	12	24
Full Genomes	18	8	10

Table 5.45: RF distance values for phyBWT2, Mash, and SANS on the *Shigella/E.coli* dataset using USTAR with k=15. Results are reported for both the original and USTAR-compressed datasets.

Dataset	phyBWT2	Mash	SANS
Original Dataset 10x	0.3333	0.1667	0.2083
Compressed Dataset 10x	0.2500	0.1667	0.2917
Original Dataset 30x	0.2917	0.2500	0.2500
Compressed Dataset 30x	0.3750	0.2083	0.5000
Original Dataset 50x	0.2917	0.2500	0.3333
Compressed Dataset 50x	0.4583	0.3333	0.4167
Original Dataset 100x	0.3333	0.2083	0.4167
Compressed Dataset 100x	0.3750	0.2500	0.5000
Full Genomes	0.3750	0.1667	0.2083

Table 5.46: Normalized RF distance values for phyBWT2, Mash, and SANS on the *Shigella/E.coli* dataset using USTAR with k=15. Comparison includes both the original and USTAR-compressed datasets.

Dataset	phyBWT2	Mash	SANS
Original Dataset 10x	16	8	10
Compressed Dataset 10x	12	8	10
Original Dataset 30x	14	12	12
Compressed Dataset 30x	18	12	12
Original Dataset 50x	14	12	16
Compressed Dataset 50x	18	12	16
Original Dataset 100x	16	10	20
Compressed Dataset 100x	20	10	20
Full Genomes	18	8	10

Table 5.47: RF distance values for phyBWT2, Mash, and SANS on the *Shigella/E.coli* dataset using USTAR with k=21. Results are reported for both the original and USTAR-compressed datasets.

Dataset	phyBWT2	Mash	SANS
Original Dataset 10x	0.3333	0.1667	0.2083
Compressed Dataset 10x	0.2500	0.1667	0.2083
Original Dataset 30x	0.2917	0.2500	0.2500
Compressed Dataset 30x	0.3750	0.2500	0.2500
Original Dataset 50x	0.2917	0.2500	0.3333
Compressed Dataset 50x	0.3750	0.2500	0.3333
Original Dataset 100x	0.3333	0.2083	0.4167
Compressed Dataset 100x	0.4167	0.2083	0.4167
Full Genomes	0.3750	0.1667	0.2083

Table 5.48: Normalized RF distance values for phyBWT2, Mash, and SANS on the *Shigella/E.coli* dataset using USTAR with k=21. Comparison includes both the original and USTAR-compressed datasets.

Dataset	phyBWT2	Mash	SANS
Original Dataset 10x	16	8	10
Compressed Dataset 10x	12	8	10
Original Dataset 30x	14	12	12
Compressed Dataset 30x	18	12	12
Original Dataset 50x	14	12	16
Compressed Dataset 50x	14	12	16
Original Dataset 100x	16	10	20
Compressed Dataset 100x	14	10	20
Full Genomes	18	8	10

Table 5.49: RF distance values for phyBWT2, Mash, and SANS on the *Shigella/E.coli* dataset using USTAR with k=31. Results are reported for both the original and USTAR-compressed datasets.

Dataset	phyBWT2	Mash	SANS
Original Dataset 10x	0.3333	0.1667	0.2083
Compressed Dataset 10x	0.2500	0.1667	0.2083
Original Dataset 30x	0.2917	0.2500	0.2500
Compressed Dataset 30x	0.3750	0.2500	0.2500
Original Dataset 50x	0.2917	0.2500	0.3333
Compressed Dataset 50x	0.2917	0.2500	0.3333
Original Dataset 100x	0.3333	0.2083	0.4167
Compressed Dataset 100x	0.2917	0.2083	0.4167
Full Genomes	0.3750	0.1667	0.2083

Table 5.50: Normalized RF distance values for phyBWT2, Mash, and SANS on the *Shigella/E.coli* dataset using USTAR with k=31. Comparison includes both the original and USTAR-compressed datasets.

Dataset	phyBWT2	Mash	SANS
Original Dataset 10x	16	8	10
Compressed Dataset 10x	12	8	10
Original Dataset 30x	14	12	12
Compressed Dataset 30x	18	12	12
Original Dataset 50x	14	12	16
Compressed Dataset 50x	18	12	16
Original Dataset 100x	16	10	20
Compressed Dataset 100x	14	10	20
Full Genomes	18	8	10

Table 5.51: RF distance values for phyBWT2, Mash, and SANS on the *Shigella/E.coli* dataset using USTAR with k=41. Results are reported for both the original and USTAR-compressed datasets.

Dataset	phyBWT2	Mash	SANS
Original Dataset 10x	0.3333	0.1667	0.2083
Compressed Dataset 10x	0.2500	0.1667	0.2083
Original Dataset 30x	0.2917	0.2500	0.2500
Compressed Dataset 30x	0.3750	0.2500	0.2500
Original Dataset 50x	0.2917	0.2500	0.3333
Compressed Dataset 50x	0.3750	0.2500	0.3333
Original Dataset 100x	0.3333	0.2083	0.4167
Compressed Dataset 100x	0.2917	0.2083	0.4167
Full Genomes	0.3750	0.1667	0.2083

Table 5.52: Normalized RF distance values for phyBWT2, Mash, and SANS on the *Shigella/E.coli* dataset using USTAR with k=41. Comparison includes both the original and USTAR-compressed datasets.

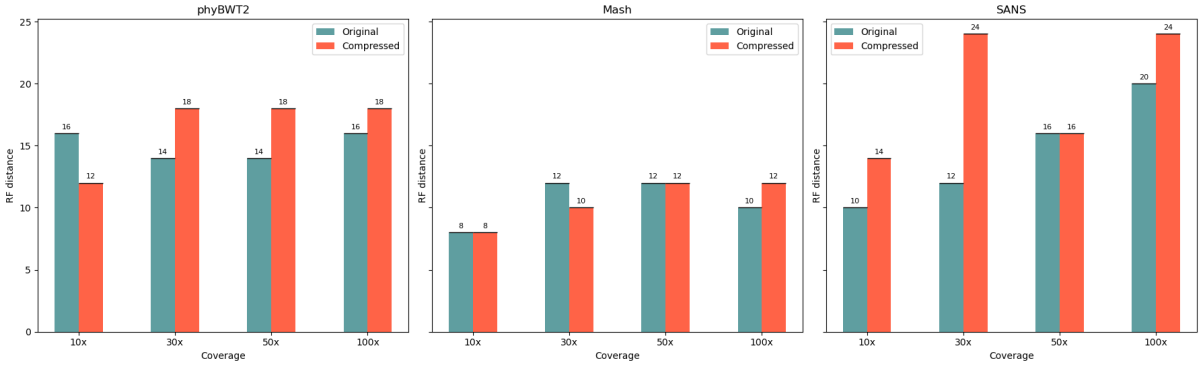


Figure 5.23: Graphical representation of RF distances for phyBWT2, Mash, and SANS on the *Shigella/E. coli* dataset using USTAR k-mer size = 15.

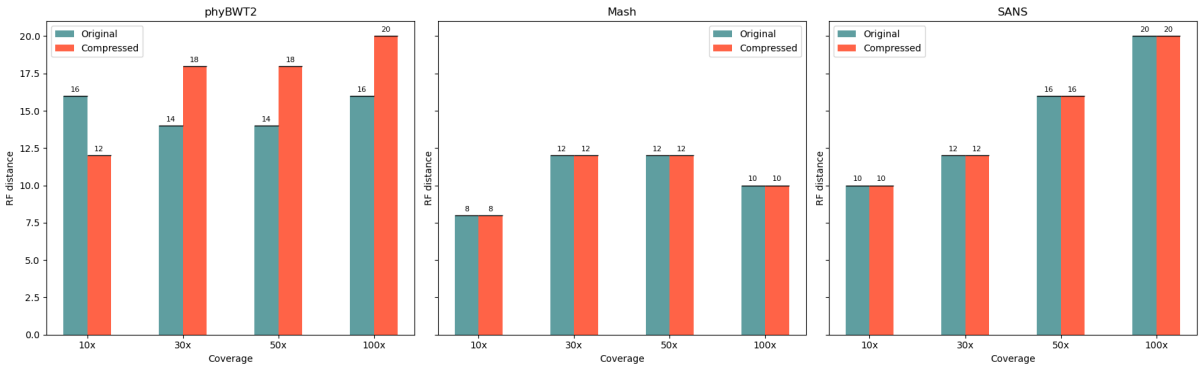


Figure 5.24: Graphical representation of RF distances for phyBWT2, Mash, and SANS on the *Shigella/E. coli* dataset using USTAR k-mer size = 21.

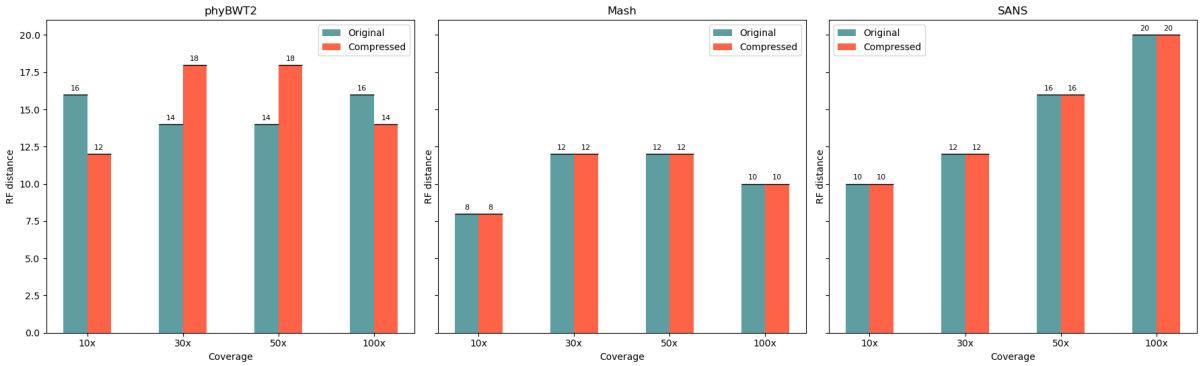


Figure 5.25: Graphical representation of RF distances for phyBWT2, Mash, and SANS on the *Shigella/E. coli* dataset using USTAR k-mer size = 31.

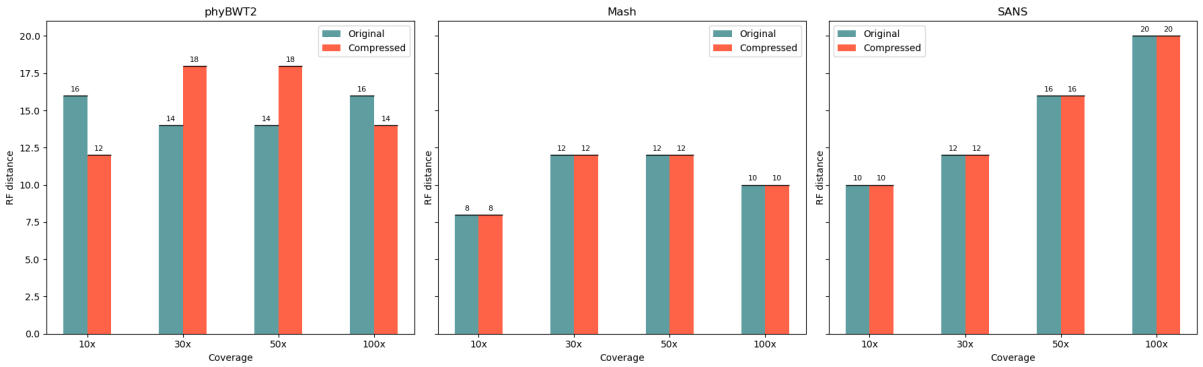


Figure 5.26: Graphical representation of RF distances for phyBWT2, Mash, and SANS on the *Shigella/E. coli* dataset using USTAR k-mer size = 41.

5.3.3 Simulated Dataset

Values of RF distance and normalized RF distance for the *Simulated* dataset, obtained with phyBWT2, Mash and SANS software, are presented. The tables, together with representative graphs, provide a visual overview of the results obtained by performing the analysis on the unassembled reads and the compressed reads using USTAR.

Dataset	phyBWT2	Mash	SANS
Original Dataset 10x	8	18	5
Compressed Dataset 10x	6	2	9
Original Dataset 30x	4	10	5
Compressed Dataset 30x	16	20	6
Original Dataset 50x	8	6	4
Compressed Dataset 50x	12	14	10
Original Dataset 100x	16	6	4
Compressed Dataset 100x	24	14	15
Full Genomes	10	24	4

Table 5.53: RF distance values for phyBWT2, Mash, and SANS on the *Simulated* dataset using USTAR with k=15. Results are reported for both the original and USTAR-compressed datasets.

Dataset	phyBWT2	Mash	SANS
Original Dataset 10x	0.1333	0.3000	0.0833
Compressed Dataset 10x	0.1000	0.0333	0.1500
Original Dataset 30x	0.0667	0.1667	0.0833
Compressed Dataset 30x	0.2667	0.3333	0.1000
Original Dataset 50x	0.1333	0.1000	0.0667
Compressed Dataset 50x	0.2000	0.2333	0.1667
Original Dataset 100x	0.2667	0.1000	0.0667
Compressed Dataset 100x	0.4000	0.2333	0.2500
Full Genomes	0.1667	0.4000	0.0667

Table 5.54: Normalized RF distance values for phyBWT2, Mash, and SANS on the *Simulated* dataset using USTAR with k=15. Comparison includes both the original and USTAR-compressed datasets.

Dataset	phyBWT2	Mash	SANS
Original Dataset 10x	8	18	5
Compressed Dataset 10x	4	18	5
Original Dataset 30x	4	10	5
Compressed Dataset 30x	4	10	5
Original Dataset 50x	8	6	4
Compressed Dataset 50x	14	6	4
Original Dataset 100x	16	6	4
Compressed Dataset 100x	12	6	4
Full Genomes	10	24	4

Table 5.55: RF distance values for phyBWT2, Mash, and SANS on the *Simulated* dataset using USTAR with k=21. Results are reported for both the original and USTAR-compressed datasets.

Dataset	phyBWT2	Mash	SANS
Original Dataset 10x	0.1333	0.3000	0.0833
Compressed Dataset 10x	0.0667	0.3000	0.0833
Original Dataset 30x	0.0667	0.1667	0.0833
Compressed Dataset 30x	0.0667	0.1667	0.0833
Original Dataset 50x	0.1333	0.1000	0.0667
Compressed Dataset 50x	0.2333	0.1000	0.0667
Original Dataset 100x	0.2667	0.1000	0.0667
Compressed Dataset 100x	0.2000	0.1000	0.0667
Full Genomes	0.1667	0.4000	0.0667

Table 5.56: Normalized RF distance values for phyBWT2, Mash, and SANS on the *Simulated* dataset using USTAR with k=21. Comparison includes both the original and USTAR-compressed datasets.

Dataset	phyBWT2	Mash	SANS
Original Dataset 10x	8	18	5
Compressed Dataset 10x	4	18	5
Original Dataset 30x	4	10	5
Compressed Dataset 30x	4	10	5
Original Dataset 50x	8	6	4
Compressed Dataset 50x	16	6	4
Original Dataset 100x	16	6	4
Compressed Dataset 100x	16	6	4
Full Genomes	10	24	4

Table 5.57: RF distance values for phyBWT2, Mash, and SANS on the *Simulated* dataset using USTAR with k=31. Results are reported for both the original and USTAR-compressed datasets.

Dataset	phyBWT2	Mash	SANS
Original Dataset 10x	0.1333	0.3000	0.0833
Compressed Dataset 10x	0.0667	0.3000	0.0833
Original Dataset 30x	0.0667	0.1667	0.0833
Compressed Dataset 30x	0.0667	0.1667	0.0833
Original Dataset 50x	0.1333	0.1000	0.0667
Compressed Dataset 50x	0.2667	0.1000	0.0667
Original Dataset 100x	0.2667	0.1000	0.0667
Compressed Dataset 100x	0.2667	0.1000	0.0667
Full Genomes	0.1667	0.4000	0.0667

Table 5.58: Normalized RF distance values for phyBWT2, Mash, and SANS on the *Simulated* dataset using USTAR with k=31. Comparison includes both the original and USTAR-compressed datasets.

Dataset	phyBWT2	Mash	SANS
Original Dataset 10x	8	18	5
Compressed Dataset 10x	4	18	5
Original Dataset 30x	4	10	5
Compressed Dataset 30x	4	10	5
Original Dataset 50x	8	6	4
Compressed Dataset 50x	6	6	4
Original Dataset 100x	16	6	4
Compressed Dataset 100x	16	6	4
Full Genomes	10	24	4

Table 5.59: RF distance values for phyBWT2, Mash, and SANS on the *Simulated* dataset using USTAR with k=41. Results are reported for both the original and USTAR-compressed datasets.

Dataset	phyBWT2	Mash	SANS
Original Dataset 10x	0.1333	0.3000	0.0833
Compressed Dataset 10x	0.0667	0.3000	0.0833
Original Dataset 30x	0.0667	0.1667	0.0833
Compressed Dataset 30x	0.0667	0.1667	0.0833
Original Dataset 50x	0.1333	0.1000	0.0667
Compressed Dataset 50x	0.1000	0.1000	0.0667
Original Dataset 100x	0.2667	0.1000	0.0667
Compressed Dataset 100x	0.2667	0.1000	0.0667
Full Genomes	0.1667	0.4000	0.0667

Table 5.60: Normalized RF distance values for phyBWT2, Mash, and SANS on the *Simulated* dataset using USTAR with k=41. Comparison includes both the original and USTAR-compressed datasets.

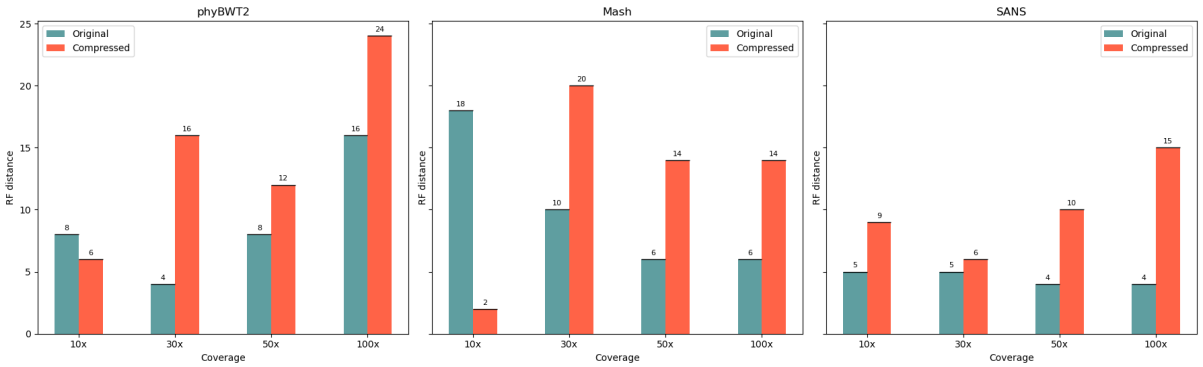


Figure 5.27: Graphical representation of RF distances for phyBWT2, Mash, and SANS on the *Simulated* dataset using USTAR k-mer size = 15.

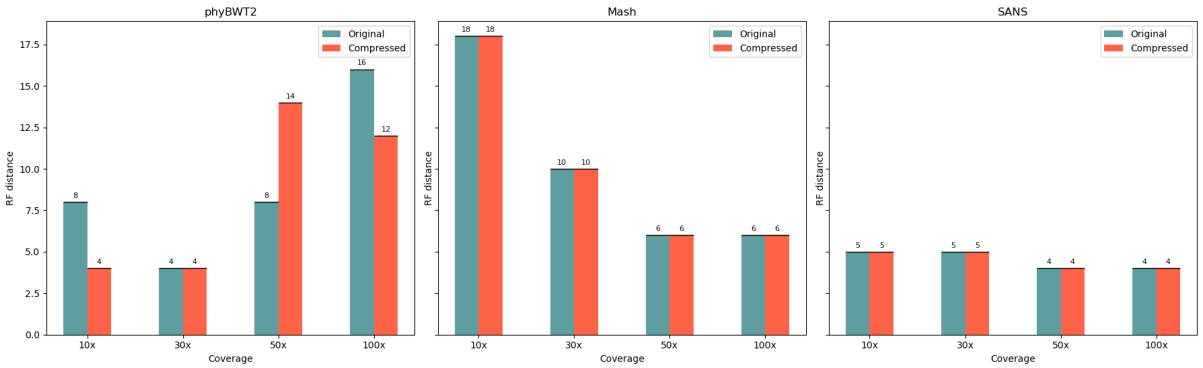


Figure 5.28: Graphical representation of RF distances for phyBWT2, Mash, and SANS on the *Simulated* dataset using USTAR k-mer size = 21.

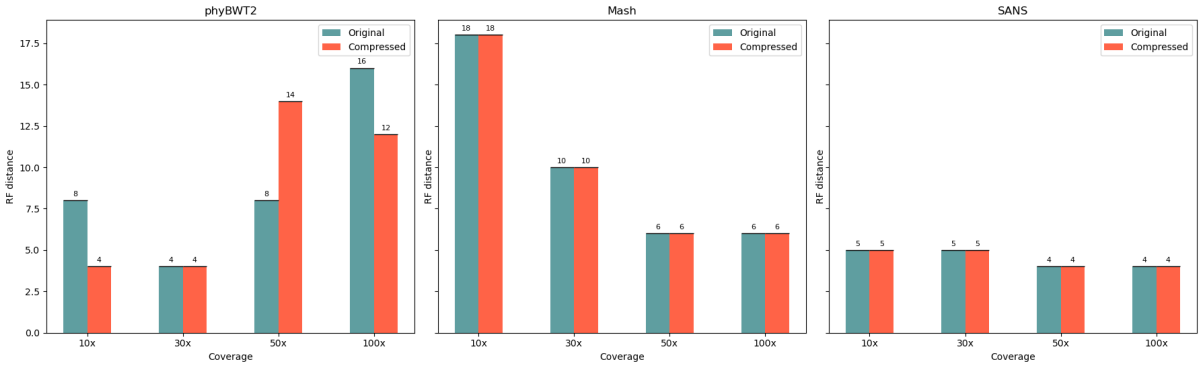


Figure 5.29: Graphical representation of RF distances for phyBWT2, Mash, and SANS on the *Simulated* dataset using USTAR k-mer size = 31.

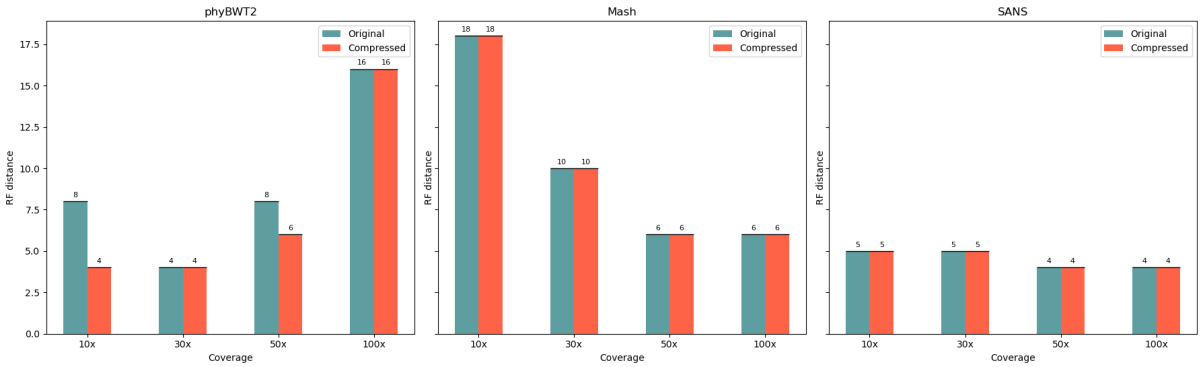


Figure 5.30: Graphical representation of RF distances for phyBWT2, Mash, and SANS on the *Simulated* dataset using USTAR k-mer size = 41.

Chapter 6

Discussion

This chapter presents an analysis and interpretation of the results from the previous chapter, focusing on the performance and accuracy of USTAR-based compression in preserving phylogenetic information. The computational efficiency of the various tools is examined, along with the impact of different k-mer sizes on the reconstructed phylogenies.
It is important to note that the results obtained are indicative of the performance of each program under the experimental conditions employed in this study. As mentioned in Chapter 4, all the experiments were conducted on the DEI cluster at the University of Padova, and the execution times can vary depending on the current load and resource availability on the cluster. Thus, while the reported values provide a clear picture of the benefits of USTAR compression, they should be interpreted with the understanding that actual performance may vary in different operational contexts.

6.1 Dataset compression

The compression results obtained with USTAR demonstrate a substantial reduction in dataset sizes across all experiments, with average size reductions of 6- to 7-fold compared to the original data. As shown in Tables 5.1-5.12, compression ratios increase with sequencing coverage, reflecting the greater redundancy that USTAR efficiently leverages.
For instance, in the *E. coli* dataset, the compression ratio rises from approximately 4.98 at 10× coverage to 10.00 at 100× coverage when using $k = 15$, with similar trends observed in the *Shigella/E. coli* and *Simulated* datasets.
The choice of k-mer size significantly impacts compression performance. Smaller k-mers capture more redundancy, leading to higher compression ratios. For example, in the *E. coli* dataset at 100× coverage, the ratio decreases from 10.00 with $k = 15$ to 8.33 with $k = 21$, 6.25 with $k = 31$, and 5.17 with $k = 41$. However, this comes at a cost: shorter k-mers increase the

risk of merging distinct DNA regions, potentially introducing errors. In contrast, larger k-mers preserve more sequence specificity, resulting in a more conservative compression that maintains precise genetic information. This trade-off is consistent across datasets, as seen in the *Shigella/E. coli* and *Simulated* cases, where higher k values correspond to lower compression ratios but better sequence preservation.

6.2 Computational performance

Experimental data in Tables 5.13-5.36 show that the reduced dataset size achieved with USTAR compression has a significant impact on both execution time and memory utilization. Due to the smaller amount of data processed, execution times are considerably reduced.
In the most favorable scenarios, the observed improvement ratios are notable: phyBWT2 achieves execution time reductions of up to ten times when comparing unassembled reads to their compressed counterparts, while Mash and SANS exhibit efficiency improvements ranging from approximately seven- to thirteen-fold, depending on the dataset and configuration. Notably, across all experiments, execution time is always reduced by at least a factor of two, confirming the consistent advantage of USTAR compression in computational efficiency.
Regarding memory usage, a significant contrast emerges between the different software. PhyBWT2 generally requires a large amount of memory when processing uncompressed datasets. However, its memory usage is greatly reduced when it operates on compressed datasets, thus reducing the resource load. In the cases of Mash and SANS serif, memory consumption remains essentially constant or increases only marginally with the use of compressed data. Since the absolute memory usage for these tools is only a few megabytes, the slight increase is not a problem.

6.3 Analysis of phylogenetic quality

The analysis of RF distance and tree comparisons across all datasets reveals some key insights. As can be seen in 5.19, 5.23 and 5.27, when USTAR compression is applied with a k-mer size of 15, phyBWT2, Mash, and SANS show a noticeable increase in RF distance compared to results obtained from unassembled reads. This degradation is likely caused by overly aggressive compression; with $k = 15$, USTAR tends to merge DNA segments that should ideally remain distinct. As a result, genomic detail is lost, introducing errors that affect the reconstruction process.
In contrast, for k-mer sizes of 21 and above, the results improve considerably. By examining 5.20–5.22, 5.24–5.26 and 5.28–5.30, it can be appreciated that for Mash and SANS the RF

distances between the trees obtained from unassembled reads and those from compressed data remain practically the same. PhyBWT2, while sometimes showing slight improvements or minor degradations, generally produces RF distances close to those observed with unassembled data.

Among the adopted tools, Mash generally provides the most stable and consistent results in terms of RF distance, although it should be noted that on the *Simulated* dataset, SANS performs very well at all levels of coverage, with RF distances never exceeding 5 when USTAR is applied with $k > 21$.

It is also worth mentioning that an increase in sequencing coverage does not uniformly result in lower RF distances. In some cases, as observed for Mash and SANS on the *Simulated* dataset, higher coverage may be associated with a decrease in RF distance; yet, in other instances, RF distances might increase with higher coverage. This variability likely reflects the need for careful tuning of software parameters, as settings optimal for one coverage level may not be ideal at another. However, this study did not aim to determine the optimal parameter settings for each software, as doing so would have required considerable time given the volume of data analyzed. More importantly, the objective of this work is not to fine-tune individual tools but rather to evaluate whether USTAR-based compression of sequencing reads provides computational advantages while preserving phylogenetic information.

Overall, these observations indicate that while USTAR compression with $k = 15$ tends to overly combine genomic segments, potentially compromising phylogenetic information. Nevertheless, utilizing k-mer sizes of 21 or higher preserves the tree topology. This preservation ensures that the compressed datasets result in RF distance values that are in good agreement with those derived from unassembled reads, thereby maintaining the integrity of the phylogenetic inference.

6.4 Additional considerations

It is also important to address the occasional occurrence of odd RF distances, particularly observed with SANS. In the computation of the Robinson–Foulds distance, one expects an even value because each split present in one tree and missing in the other contributes symmetrically to the total. However, odd RF distances can arise when the trees under comparison contain polytomies, i.e. nodes with more than two direct descendants. A fully bifurcating tree with n leaves has exactly $n - 3$ internal splits, whereas a tree with polytomies has fewer splits. As a result, when a fully resolved tree is compared to one with polytomies, the imbalance in split counts can lead to an odd RF value.

This phenomenon does not necessarily indicate greater phylogenetic divergence but rather a difference in node resolution. As illustrated in Figure 6.1, one of the cases where the RF distance

equals 5 is shown: the SANS tree obtained from compressed data exhibits a polytomy at the root, while the reference tree is fully resolved. Although the trees correctly position nearly all species, this unresolved node introduces an imbalance in the number of splits, resulting in an odd RF distance. Nonetheless, this structural difference does not substantially alter the overall inferred phylogenetic relationships.

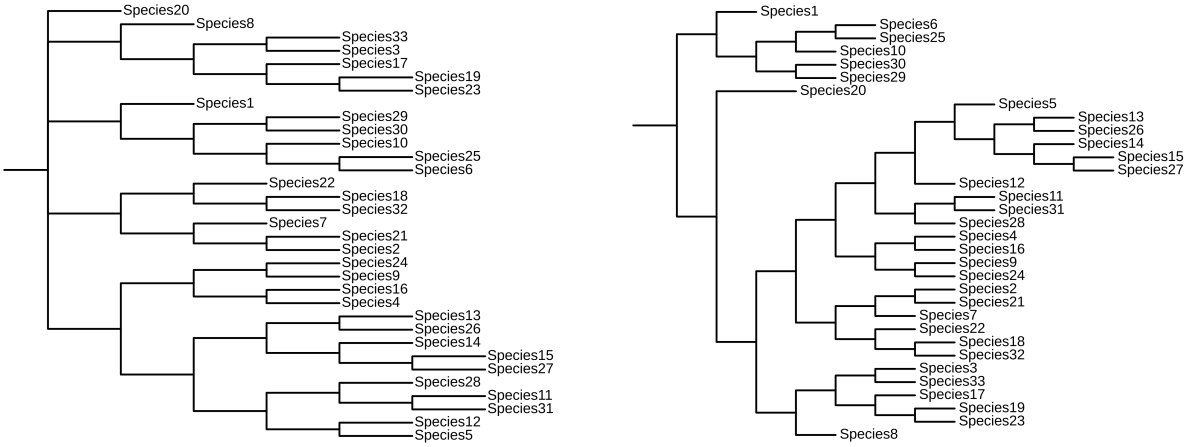


Figure 6.1: Phylogenetic tree obtained with SANS from the *Simulated* dataset at 30× coverage using USTAR with k=21 on the left, and the reference tree on the right, allowing for a direct comparison.

Chapter 7

Conclusion

This study investigated the effectiveness of USTAR for compressing k-mer sets derived from unassembled sequencing reads and evaluated its impact on both computational performance and phylogenetic accuracy.

The results highlight two primary advantages of USTAR. First, the considerable reduction in dataset size minimizes storage requirements, which is particularly useful for large-scale sequencing studies. Second, the computational complexity is significantly decreased due to the reduced number of sequences and the generation of longer, more informative compressed units. This leads to notably shorter execution times and lower memory usage during phylogenetic reconstruction.

These findings indicate that USTAR offers a valuable benefit by enabling a more efficient data processing pipeline while maintaining, and in some cases even improving, the quality of phylogenetic analyses. It is important to note that the choice of k-mer length is crucial: if the k-mer size is set too small, the compression may be overly aggressive and lead to a loss of phylogenetic resolution, while an appropriate k-mer length ensures that the compression preserves the essential phylogenetic information by producing fewer, longer sequences that encapsulate the same biological content.

Overall, the analyses performed suggest that USTAR is an effective approach to reduce both data size and computational load while preserving phylogenetic accuracy. This study contributes to the development of efficient compression strategies to support more scalable and computationally efficient bioinformatics workflows.

7.1 Future Work

Future work could explore the application of USTAR beyond the field covered in this study. In particular, it would be worth investigating whether the computational advantages observed in phylogenetic reconstruction extend to other applications, such as de novo assembly, gene expression analysis, or any bioinformatics domain that processes large volumes of data.
In addition, evaluating the robustness of USTAR under conditions of noisy data or sequencing errors could help determine whether the observed performance advantages persist under non-ideal operating scenarios.
Finally, integrating USTAR into larger workflows that include multidimensional analyses, such as simultaneous processing of genomic and transcriptomic data.

Bibliography

[1] T. Hu, N. Chitnis, D. Monos, and A. Dinh, "Next-generation sequencing technologies: An overview," *Human Immunology*, vol. 82, no. 11, pp. 801–811, 2021, Next Generation Sequencing and its Application to Medical Laboratory Immunology, ISSN: 0198-8859. DOI: https://doi.org/10.1016/j.humimm.2021.02.012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0198885921000628.

[2] G. A. Logsdon, M. R. Vollger, and E. E. Eichler, "Long-read human genome sequencing and its applications," *Nature Reviews Genetics*, vol. 21, no. 10, pp. 597–614, 2020, ISSN: 1471-0064. DOI: 10.1038/s41576-020-0236-x. [Online]. Available: https://doi.org/10.1038/s41576-020-0236-x.

[3] P. E. C. Compeau, P. A. Pevzner, and G. Tesler, "How to apply de bruijn graphs to genome assembly," *Nature Biotechnology*, vol. 29, no. 11, pp. 987–991, 2011, ISSN: 1546-1696. DOI: 10.1038/nbt.2023. [Online]. Available: https://doi.org/10.1038/nbt.2023.

[4] G. Holley and P. Melsted, "Bifrost: Highly parallel construction and indexing of colored and compacted de bruijn graphs," *Genome Biology*, vol. 21, no. 1, p. 249, 2020, ISSN: 1474-760X. DOI: 10.1186/s13059-020-02135-8. [Online]. Available: https://doi.org/10.1186/s13059-020-02135-8.

[5] R. Chikhi, A. Limasset, and P. Medvedev, "Compacting de bruijn graphs from sequencing data quickly and in low memory," en, *Bioinformatics*, vol. 32, no. 12, pp. i201–i208, Jun. 2016.

[6] M. Burrows, D. J. W. D. I. G. I. T. A. L, R. W. Taylor, D. J. Wheeler, and D. Wheeler, "A block-sorting lossless data compression algorithm," 1994. [Online]. Available: https://api.semanticscholar.org/CorpusID:2167441.

[7] S. Mantaci, A. Restivo, G. Rosone, and M. Sciortino, "An extension of the burrows–wheeler transform," *Theoretical Computer Science*, vol. 387, no. 3, pp. 298–312, 2007, The Burrows-Wheeler Transform, ISSN: 0304-3975. DOI: https://doi.org/

10.1016/j.tcs.2007.07.014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0304397507005282.

[8] S. Schmidt, S. Khan, J. N. Alanko, G. E. Pibiri, and A. I. Tomescu, "Matchtigs: Minimum plain text representation of k-mer sets," *Genome Biology*, vol. 24, no. 1, p. 136, Jun. 2023.

[9] A. Rahman and P. Medevedev, "Representation of k-mer sets using Spectrum-Preserving string sets," en, *J Comput Biol*, vol. 28, no. 4, pp. 381–394, Dec. 2020.

[10] E. Rossignolo and M. Comin, "Ustar: Improved compression of k-mer sets with counters using de bruijn graphs," in *Bioinformatics Research and Applications*, X. Guo, S. Mangul, M. Patterson, and A. Zelikovsky, Eds., Singapore: Springer Nature Singapore, 2023, pp. 202–213, ISBN: 978-981-99-7074-2.

[11] R. R. Sokal, "The principles and practice of numerical taxonomy," *Taxon*, vol. 12, no. 5, pp. 190–199, 1963, ISSN: 00400262. [Online]. Available: http://www.jstor.org/stable/1217562 (visited on 03/12/2025).

[12] L. L. Cavalli-Sforza and A. W. F. Edwards, "Phylogenetic analysis: Models and estimation procedures," *Evolution*, vol. 21, no. 3, pp. 550–570, 1967, ISSN: 00143820, 15585646. [Online]. Available: http://www.jstor.org/stable/2406616 (visited on 03/12/2025).

[13] J. Felsenstein, "Evolutionary trees from DNA sequences: A maximum likelihood approach," *Journal of Molecular Evolution*, vol. 17, no. 6, pp. 368–376, Nov. 1981.

[14] J. P. Huelsenbeck, F Ronquist, R Nielsen, and J. P. Bollback, "Bayesian inference of phylogeny and its impact on evolutionary biology," en, *Science*, vol. 294, no. 5550, pp. 2310–2314, Dec. 2001.

[15] V. Guerrini, A. Conte, R. Grossi, G. Liti, G. Rosone, and L. Tattini, "phyBWT: Alignment-Free Phylogeny via eBWT Positional Clustering," in *22nd International Workshop on Algorithms in Bioinformatics (WABI 2022)*, C. Boucher and S. Rahmann, Eds., ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 242, Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, 23:1–23:19, ISBN: 978-3-95977-243-3. DOI: 10.4230/LIPIcs.WABI.2022.23. [Online]. Available: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.WABI.2022.23.

[16] V. Guerrini, A. Conte, R. Grossi, G. Liti, G. Rosone, and L. Tattini, "Phybwt2: Phylogeny reconstruction via ebwt positional clustering," *Algorithms for Molecular Biology*, vol. 18, no. 1, p. 11, 2023, ISSN: 1748-7188. DOI: 10.1186/s13015-023-00232-4. [Online]. Available: https://doi.org/10.1186/s13015-023-00232-4.

[17] B. D. Ondov, T. J. Treangen, P. Melsted, *et al.*, "Mash: Fast genome and metagenome distance estimation using minhash," *Genome Biology*, vol. 17, no. 1, p. 132, 2016, ISSN: 1474-760X. DOI: 10.1186/s13059-016-0997-x. [Online]. Available: https://doi.org/10.1186/s13059-016-0997-x.

[18] A. Rempel and R. Wittler, "Sans serif: Alignment-free, whole-genome-based phylogenetic reconstruction," *Bioinformatics*, vol. 37, no. 24, pp. 4868–4870, Jun. 2021, ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btab444. eprint: https://academic.oup.com/bioinformatics/article-pdf/37/24/4868/50334826/btab444.pdf. [Online]. Available: https://doi.org/10.1093/bioinformatics/btab444.

[19] R. Wittler, "Alignment- and reference-free phylogenomics with colored de bruijn graphs," *Algorithms for Molecular Biology*, vol. 15, no. 1, p. 4, 2020, ISSN: 1748-7188. DOI: 10.1186/s13015-020-00164-3. [Online]. Available: https://doi.org/10.1186/s13015-020-00164-3.

[20] G. Bernard, C. X. Chan, and M. A. Ragan, "Alignment-free microbial phylogenomics under scenarios of sequence divergence, genome rearrangement and lateral genetic transfer," en, *Sci Rep*, vol. 6, p. 28 970, Jul. 2016.

[21] H. Yi and L. Jin, "Co-phylog: An assembly-free phylogenomic approach for closely related organisms," en, *Nucleic Acids Res*, vol. 41, no. 7, e75, Jan. 2013.

[22] R. G. Beiko and R. L. Charlebois, "A simulation test bed for hypotheses of genome evolution," en, *Bioinformatics*, vol. 23, no. 7, pp. 825–831, Jan. 2007.

[23] A. Zielezinski, H. Z. Girgis, G. Bernard, *et al.*, "Benchmarking of alignment-free sequence comparison methods," *Genome Biology*, vol. 20, no. 1, p. 144, Jul. 2019.

[24] M. Holtgrewe, "Mason - a read simulator for second generation sequencing data," *Technical Report FU Berlin*, 2010. [Online]. Available: http://publications.imp.fu-berlin.de/962/.

[25] J. Felsenstein, *Inferring Phylogenies*. Sunderland, MA: Sinauer Associates, 2004, ISBN: 978-0-87893-177-4.

[26] D. Robinson and L. Foulds, "Comparison of phylogenetic trees," *Mathematical Biosciences*, vol. 53, no. 1, pp. 131–147, 1981, ISSN: 0025-5564. DOI: https://doi.org/10.1016/0025-5564(81)90043-2. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0025556481900432.

[27] M. A. Moreno, M. T. Holder, and J. Sukumaran, *Dendropy 5: A mature python library for phylogenetic computing*, 2024. DOI: 10.21105/joss.06943. [Online]. Available: https://doi.org/10.21105/joss.06943.