Stockholms universitet

Investigating Substring Based Methods for k-mer Abundance Prediction

Undersökning av delsträngsbaserade metoder för förutsägelse av k-mer förekomster

Florence Hugh

Handledare: Kristoffer Sahlin
Examinator: Lars Arvestad
Inlämningsdatum: 20 Maj 2024

Kandidatuppsats i datalogi
Bachelor Thesis in Computer Science

Abstract

With the exponential growth of sequencing data in bioinformatics, the need for efficient tools and methods to handle the data becomes increasingly important. Among these tools, k-mers play an essential role in data analysis, enabling computational tasks to be performed more effectively. While there are numerous methods available for counting k-mers, memory constraints remain a significant challenge.

In this thesis, we investigate a memory-frugal approximate method for predicting k-mer abundance by using shorter subsequences of k-mers, aiming to alleviate memory constraints. We specifically investigate the effectiveness of our developed function f_{min}, which calculates the minimum count from a set of substrings obtained from the k-mers. Our method demonstrates promising results, particularly in accurately predicting low-frequency k-mers. However, it faced some challenges when dealing with high-frequency k-mers and highly repetitive datasets. Nevertheless, our approach proves valuable for applications that prioritize the analysis of unique or rare k-mers.

Sammanfattning

Med en exponetiell tillväxt av sekvenseringsdata inom bioinformatik blir behovet av effektiva vertyg och metoder för att hantera datan allt viktigare. Bland dessa verktyg spelar k-mers en avgörande roll i dataanalysen, vilket möjliggör att beräkningar kan utföras mer effektivt. Trots att det finns många metoder tillgängliga för att räkna k-mers kvarstår minnesbegränsningar som en väsentlig utmaning.

I detta arbete undersöker vi en minnessnål approximativ metod för att förutsäga k-mer förekomster genom att använda kortare delsträngar av k-mers, med syfte att lindra minnesbegränsningar. Vi undersöker specifikt effektiviteten hos vår utvecklade funktion f_{min}, som beräknar det minsta antalet från en uppsättning delsträngar som erhålls från k-mern. Vår funktion visar lovande resultat, särskillt när det gäller att förutsäga lågfrekventa k-mers. Det stötte dock på vissa utmaningar när det gäller högfrekventa k-mers och mycket repetitiva dataset. Trots detta visar vårt tillvägagångssätt vara användbart för tillämpningar som prioriterar analysen av unika eller sällsynta k-mers.

Contents

1. Introduction

Advances in next-generation sequencing technology have made it possible to obtain a huge amount of sequencing data [15]. Consequently, there is a significant demand for the analysis and interpretation of the data, and bioinformatics is playing a crucial role in meeting this demand. Bioinformatics finds applications across various fields, including biological research and healthcare. These range from genome sequencing and protein structure prediction to drug discovery and personalized medicine [2]. In sequencing data analysis, many applications make use of substrings of length k extracted from biological sequences, commonly known as k-mers. These k-mers are used in various tasks such as genome assembly, multiple sequence alignment and duplicate detection. Additionally, they are valuable for analysing assembled genomes, enabling measurements of genome similarities and accuracy of metagenomic classification [18].

Due to the wide range of applications for k-mers, numerous methods for counting k-mers have been developed. Counting k-mers is the process of finding the frequency of each distinct k-mer present in a sequence. There exists approximate and exact methods and these can be classified based on the approach and data structure they use. This classification includes techniques that make use of specially tailored sorting methods [7], hash tables [11], and Bloom filters [12]. However, when dealing with larger datasets, memory constraints can become an issue. To address this challenge, many methods make use of a disk-based strategy, where k-mers are stored and processed directly on disk storage rather than in memory [10]. While disk-based (also known as external-memory based) strategies can overcome memory limitations, they can increase I/O overhead, resulting in longer processing times and reduced efficiency, particularly when accessing disk storage repeatedly. Given these challenges, this thesis aims to explore main memory approximate techniques to predict the abundance of k-mer sequences.

1.1. Research Topic and Questions

In general, counting the frequency of k-mers is straightforward for small values of k. With an alphabet of size 4, it is possible to store a vector of size 4^k without encountering memory constraints. Each memory slot in the vector will represent the count of a distinct k-mer. This thesis aims to investigate the possibility of using a vector containing counts of shorter k-mers to predict the abundance of longer k-mers, such as $k = 20$ and above, which are impractical or impossible to store in main memory. Specifically, we will explore the efficacy of using our implemented f_{min} function (defined in Section 4.1.2), which, intuitively, takes the minimum count over a set of substrings in the k-mer, as a predictor function and address the following research question:

- How accurately does the f_{min} function measure k-mer abundance on simulated and biological data?

2. Background

This chapter provides an overview of bioinformatics, focusing on the role of k-mers in sequence analysis and explores how k-mers are used for tasks like read mapping and genome assembly. Additionally, it discusses what k-mers are, how they appear in biological sequences and different strategies for k-mer counting.

2.1. Sequence Analysis

Next-generation sequencing (NGS) technology have the capacity to generate millions of nucleotide sequences from a genomic DNA, providing broad coverage of entire genomes or specific genomic regions [3]. These sequences, commonly known as *reads*, are obtained in unordered chunks and typically consist of relatively short genomic sequences. The length for short reads typically range from 25-450 bases [14], in contrast to a genome that can consist of millions or billions of nucleotides [16]. Reads serves as a basis for various genomic analysis, including read mapping and genome assembly.

In read mapping, the goal is to determine the most likely position for each read within the genome. This process requires the availability of a reference genome, in which reads can be mapped by comparison [1]. However, when the reference genome is very large, the mapping process can face computational challenges. This is because it has to search the whole genome database to map the millions of short reads. To simplify searching, one can make use of a seed-and-extend heuristic method [21]. The seed-and-extend heuristic involves using k-mers as seeds to identify regions where the short-read and a part of the reference genome matches approximately. These regions are known as k-mers, and can be used to narrow down the search range, enhancing the mapping process.

In genome assembly, reads serves as the basis for reconstructing the complete genome. This involves putting together overlapping reads into longer contiguous sequences, known as contigs, which represents segments of the genome. The goal is to accurately assembly contigs to form the whole genome. In many assembly algorithms, k-mers are used as assembly units instead of whole reads. A common approach is by constructing the de Bruijn graph [5], in which k-mers represents nodes and an edge between two k-mers exists if they overlap by $k - 1$ nucleotides. By constructing the de Bruijn graph, the assembly algorithm can find relationships between k-mers and identify paths through the graph that correspond to contiguous sequences in the genome.

In both read mapping and genome assembly k-mers play a crucial role, allowing researchers to extract genetic information encoded within DNA sequences obtained through NGS technology.

2.1.1. Applications of *k*-mers

In many bioinformatic applications, it is the frequency of each k-mer that is of interest. This frequency data can be used in various analyses. For instance, to get an overview of the distribution of k-mers within a genome, allows researchers to discover rare patterns and analyse the complexity of the genome [4]. Moreover, based on the frequency of k-mers, one can apply downsample methods on datasets by keeping only the most relevant k-mers to reduce the dataset size. Sketching-based algorithms make use of this technique to filter out reads by downsampling the occurrences of high-frequency k-mers within the reads [19], thereby speeding up the read mapping process.

Additionally, sketching is a common approach in metagenomic studies. Metagenomics involve the direct analysis of genetic material obtained from environmental samples, including microorganisms like bacteria, viruses and archaea [8]. Typically, a metagenomic dataset contains thousands of genomes, where the goal is to classify the origin of genetic sequences (reads) obtained from environmental samples. To achieve this, one can use set of k-mers to represent the reads and the genomes. By comparing the k-mer distribution of a read to the k-mers of the genomes in the dataset, one can identify the closest matching genome [6]. This approach allows for efficient and accurate classification of reads.

2.2. Understanding *k*-mers in Sequence Analysis

The term *k-mer* refers to a substring of length k within a biological sequence. Generating k-mers involves extracting consecutive sequences of length k from the original sequence. To generate all k-mers from a sequence, begin by extracting the first k characters, forming the initial k-mer. Then shift one character to the right to create the next k-mer. By repeating this process until reaching the end of the sequence, all k-mers are obtained. Given a sequence of length N, the total number of k-mers will be $N - k + 1$, see Figure 2.1.

```
1: ATTATTAGCG
2: ATTA
3:  TTAT
4:   TATT
5:    ATTA
6:     TTAG
7:      TAGC
8:       AGCG
```

Figure 2.1 The procedure of generating all 4-mers from the sequence `ATTATTAGCG`.

As mentioned in the previous section, it is the frequency of k-mers that matters. By counting the occurrence of each k-mers within a sequence, one obtains its frequency.

For instance, in Figure 2.1, the k-mer `ATTA` appears twice, giving it a frequency of two, while the other k-mers are unique, each with a frequency of one. The following section will explain the different strategies for quantifying the frequency of k-mers.

2.2.1. Overview of Memory and Disk-Based Strategies

To address the challenges associated with k-mer counting in large datasets from NGS, researchers have developed memory and disk-based strategies. Memory-based methods store data in Random Access Memory (RAM), allowing for fast access but are limited by memory capacity. In contrast, disk-based approaches store data on disk storage, enabling efficient processing of large datasets but at the expense of increased I/O overhead [10]. Understanding the trade-offs between memory and disk-based strategies is essential for optimizing bioinformatics analyses and effectively managing large-scale genomic datasets.

The k-mer counting tools that have been developed use various approaches and algorithms to quantify k-mer frequencies while effectively managing memory and disk resources. Here are some well known tools:

- KMC (K-mer Counter): KMC take on a disk-based approach. It uses a sorting algorithm, such as radix sort, to count k-mers. This involves extracting all k-mers from each read and storing them in disk files. Then the k-mers are collected from the files to be sorted in lexicography order and finally the frequency of each distinct k-mer are counted [7]. The disk-based approach allows KMC to handle datasets that exceed memory capacity, making it suitable for handling large datasets.

- BFCounter: BFCounter uses a Bloom filter, a probabilistic data structure, to store k-mers in memory. This method counts non-unique k-mers by filtering out unique ones with the Bloom filter. The Bloom filter is a memory frugal approach but comes with some loss in precision, as it has a false positive rate, meaning some unique k-mers can pass the filter and be counted [12]. Nevertheless, BFCounter can effectively manage memory resources and handle large datasets without exceeding memory limits.

- Jellyfish: Jellyfish uses a multithreaded, lock-free strategy to efficiently manipulate a hash-table, which allows parallel insertion of keys and updates to values [11]. This approach allow Jellyfish to effectively handle large datasets by distributing computational tasks across multiple threads, making it a fast and memory efficient tool.

Each of these k-mer counting tools offers unique advantages tailored to specific computational challenges. These tools collectively enhance the ability to handle large-scale genomic data, enabling faster and more efficient k-mer frequency analysis in various bioinformatics applications.

2.2.2. Biologically Repeated k-mers

Given a DNA sequence, which is composed of four nucleotides bases: adenine (`A`), citosine (`C`), guanine (`G`), and thymine (`T`). Let $\Sigma = \{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{T}\}$ denote the alphabet of a DNA sequence. A k-mer will be a sequence of length k over the alphabet Σ, giving a total number of possible k-mers to be 4^k.

To understand the likelihood of a k-mer occurring more than once in a randomly generated sequence, consider the case where $k = 20$. Assume that a randomly generated string of length N can be modelled by randomly generating $N - k + 1$ k-mers. While this is not accurate due to overlapping k-mers in the string, it serves as a reasonable approximation for our purposes. When $k = 20$, there exists $4^{20} \approx 1.1 \cdot 10^{12}$ unique k-mers. The probability for a k-mer of length 20 (20-mer) to appear at a given position in a randomly generated string is $(1/4)^{20}$, if the letters in Σ are evenly distributed. This probability is extremely low, indicating that the occurrence of a specific k-mer in a string of given length N is highly unlikely as k grows. Roughly, one might expect in a random sequence of length 10^9 (the order of the size of a human genome), any 20-mer that is present in the sequence will also occur only once.

This is because, in a sequence S of size $N = 10^9$, there are $M = 10^9 - 20 + 1$ positions for a 20-mer to start. To determine the probability for a specific 20-mer to appear at least once in S, we use the complement probability. The complement is that this 20-mer will never appear in a given position, which has probability $1 - (1/4)^{20}$. Assuming that each 20-mer are independent and identically distributed over Σ, the probability for a specific 20-mer to appear at least once can be calculated using the complementary probability for the 20-mer to never appear at any of the M positions:

$$1 - \left(1 - \left(\frac{1}{4}\right)^{20}\right)^{(10^9 - 20 + 1)} \approx 9.09 \cdot 10^{-4}.$$

This number is much less than 1, indicating that in a random sequence of length $N = 10^9$, the specific 20-mer is unlikely to appear even once. This confirms our assumption that each k-mer is likely to occur only once in a random sequence.

However, this assumption does not hold in biological sequences due to the presence of biological repeats. Biological repeats (also known as repetitive sequences), are sequences of DNA that appear multiple times within a genome. The repeats can be classified into two types: tandem and transposons repeats. Tandem repeats are sequences that are repeated one after another, often in a consecutive order, while transposons makes duplicates of sequences by copying them and adding it to another part of the genome [9]. These repeats results in k-mers occuring multiple times within a biological sequence.

3. Preliminaries

This chapter defines concepts to better understand our method, starting with the notation of k-mers and s-mers. Followed by the concepts of k-mer canonicity to address uncertainties in sequence orientation. Additionally, we introduce the mean absolute error metric for evaluating predictor function performance.

3.1. Sequence Representation

3.1.1. Definition: k-mer and s-mer

Given a DNA sequence S, we will use i to index (starting from 0) the position in S and denote a subsequence of length k starting at position i as $S[i : k]$. This will represent a k-mer covering the k consecutive positions $i, i + 1, ..., i + k - 1$ in S. Additionally, within a k-mer sequence, we will introduce shorter subsequences of length s. These will be called s-mers and can be denoted as $k[j : s]$, where j denotes the starting position in the k-mer.

3.1.2. k-mer Canonicity

DNA consists of two complementary strands, a forward and a reverse complement strand. In situations where the direction of a given DNA sequence is unknown, meaning it is uncertain whether the sequence should be read from left to right or from right to left, methods to handle the orientation must be considered. One approach to address this situation involves the concept of canonical k-mers. This approach assumes a k-mer and its reverse complement to be equivalent, regardless of the sequence's orientation. The canonical representation of both will be chosen as the lexicographically smaller one [20].

To define canonical k-mers formally, consider a k-mer at position i from a DNA sequence S as $S[i : k]$. The reverse complement of this k-mer, denoted as $rc(S[i : k])$, is obtained by first reversing the k-mer and then replacing the nucleotides with its complementary base, $\overline{\mathtt{A}} = \mathtt{T}$, $\overline{\mathtt{T}} = \mathtt{A}$, $\overline{\mathtt{C}} = \mathtt{G}$ and $\overline{\mathtt{G}} = \mathtt{C}$. For instance, if $S[i : k] = \mathtt{TTAG}$, the reverse becomes `GATT` and then the bases are replaced giving $rc(S[i : k]) = \mathtt{CTAA}$. The canonical k-mer, denoted k_c, satisfies the following:

$$k_c = \min\{S[i : k],\ rc(S[i : k])\},$$

where min represents the lexicography smaller of the two.

3.2. Evaluation Metrics

3.2.1. Mean Absolute Error

Mean Absolute Error (MAE) is a distance based metric used to measure the average absolute error between the predicted and actual values within a dataset. It can be defined as follows:

$$MAE = \frac{1}{n} \sum_{i=0}^{n} |y_i - \hat{y}_i|,$$

where n represents the total number of data points, y_i the actual value and $\hat{y}_i$ the corresponding predicted value. MAE provides a straightforward way to determine the accuracy of a predictive method, with lower MAE values indicating better performance. This is because smaller MAE values indicate fewer differences between predicted and actual values, suggesting that the method's predictions are closer to the real outcomes.

4. Method

This chapter outlines the development and application of the implemented method. The method is implemented in C++, focusing exclusively on processing DNA sequences and files written in FASTA format [13]. Moreover, the method is designed to handle canonical k-mers as it is standard in bioinformatic software. All source code can be found at `https://github.com/FlorenceHugh/BSc_thesis`.

4.1. Predicting k-mer Frequency using s-mer Frequency

Counting the occurrence of every k-mer becomes straightforward for small values of k. A simple approach involves using a vector of size 4^k, where each slot corresponds to a unique k-mer. When a k-mer is encountered, its respective slot in the vector is incremented to track its occurrence. Our implemented method stemmed from this fundamental concept of counting the frequency of k-mers efficiently.

Our method aims to predict the occurrence of every possible k-mer within a given sequence by making use of a vector containing counts of shorter k-mers, each of fixed length s. These shorter k-mers, referred to as s-mers, are substrings of the target k-mers and are essential in our analysis.

4.1.1. Constructing the s-mer Vector

The source file `s_mer_counter.cpp` is used to construct the s-mer vector. It starts by initializing the s-mer vector by taking an integer s as input to determine its size. Since our method works with DNA sequences, the size of the vector is set to be 4^s to accommodate all possible combination of the s-mers. As we are counting canonical k-mers, this vector will keep track of the frequency of every canonical s-mer occurring in the input sequence.

Each index of the vector represents a canonical s-mer. To map a sequence to its correct index, the algorithm first finds the canonical representation of the s-mer. Then it converts the nucleotides `A`, `C`, `G` and `T` to a binary representation of 00, 01, 10 and 11, respectively. This with the help of the source file `index_and_canonical.cpp`. For instance, the sequence `AAA` is represented binary as 000000, which has decimal representation of 0, so the sequence is added to index 0. Similarly, the sequence `ACA`, represented in binary as 000100 and converted to its decimal representation as 4, is added to index 4. The algorithm uses a 32-bit data type to store the binary representation, which limits the maximum size of the s-mer to 16 nucleotides.

To complete the vector with s-mer frequency, it iterates over the FASTA file containing the sequencing data, identifying each row containing sequences. For each sequence, it extracts every subsequence of length s and increments the count in the corresponding index of the vector. This process ensures that the s-mer vector is accurately filled with the counts of s-mers from the input sequences.

4.1.2. Predictor Functions

Let f denote the predictor functions $f : K \to \mathbb{Z}^+$, where K represents the set of all k-mers. Additionally, let c denote a counter function $c : M \to \mathbb{Z}^+$, where M represents a k- or s-mer. These functions are used to predict the abundance of k-mers based on the counts of all s-mers present within a k-mer. This section will explain the implementation of the predictor function, found in source file `predict_k_mer.cpp`, with a particular focus on the minimum function.

The Minimum Function: f_{min}

Let f_{min} denote the minimum function, defined as

$$f_{min}(k) = \min_{i \in [0,n]} \{c(k[i : s])\},$$

where i is the index position in k and n is the number of s-mers present within a k-mer. This predictor function was developed based on the fact that $f_{min}(k) \geq c(k)$. This inequality intuitively implies that the occurrence of any k-mer cannot exceed the count of its s-mers, since the presence of each s-mer contributes to the count of the k-mer. Therefore, the minimum count among all s-mers within the k-mer serves as an upper bound for the abundance of the k-mer itself.

The function works by iterating over the FASTA file containing the sequencing data to examine all k-mers. For each k-mer, it picks the canonical representation and the frequency of each s-mer present are compared and the smallest frequency value observed is selected. This minimum frequency value is then assigned as the predicted abundance for that specific k-mer.

4.2. Testing and Evaluating the Predictor Function

In the following section, we examine some distance metrics, mainly the implementation of the Mean Absolute Error (MAE) function, to evaluate the performance of our predictor function. For testing the method, the real abundance for the k-mers are needed, which is obtain by the Jellyfish tool [11]. This tool will provide us a file containing the real abundance of canonical k-mers saved in FASTA format, where the description line now represents the k-mer count and the sequence data lines a distinct canonical k-mer.

4.2.1. Testing on Simulated and Biological Data

To check the effectiveness of the predictor function, we tested it on simulated datasets with varying mutation rates and two biological datasets. This allows us to evaluate the function's performance under different conditions and levels of genetic variation. The two biological datasets was: the Human Y Chromosome (ChrY) and 20 genomes of Escherichia coli (E.coli). These real-world datasets offer valuable insights into how well the predictor function performs in practical genomic scenarios. By testing our method on both simulated and biological data, we can ensure its effectiveness and reliability across a range of genetic contexts.

Generate Simulated Data

The simulated data was generated using a Python script called `simulated_data.py`. This script starts by creating a random sequence of a specified length, uniformly distributed over the nucleotides `A`, `C`, `G` and `T`, which serves as the first reference sequence. Additional reference sequences are generated by introducing mutations into existing sequences. Each generated sequence is added to a pool from which the generator selects sequences to be mutated, creating new sequences in the process. This iterative process results in a tree structure, where each branch represents a distinct variant. As mutations builds up and sequences diverge from the original reference, the branching pattern of the tree reflects the evolutionary relationships between sequences.

The mutations involve deletions, insertions, and substitutions of nucleotides, with the mutation rate determining the number of nucleotides to be altered within a sequence. For instance, if the reference sequence consists of 2 000 nucleotides and the mutation rate is set to 1%, then the number of nucleotides to be mutated are $(2\,000 \cdot 0.01)/2 = 10$. The ten nucleotides to be mutated are picked uniformly across the reference sequence.

The script was used to generate five different dataset, each with a distinct mutation rate ranging from 1% to 5%. For each dataset we simulated 50 sequences, each with an approximate length of 2 000.

Human Y Chromosome and Escherichia coli Genomes

The E.coli dataset is similar to the simulated data in such a way that it contains mutated reference sequences. The mutation rate for this dataset is roughly 0.5-2%, but these mutations are not uniformly distributed as they are evolved biologically. Each row consists of approximately five million nucleotides and since we consider 20 genomes, this gives a dataset containing around 100 million nucleotides.

The ChrY dataset have a more complex repeated structure. It consists of around 60 million nucleotides in which approximately half of them are repetitive sequences [17].

4.2.2. Evaluation pipeline

The performance will be evaluated by analysing the MAE between the real and predicted k-mer abundances. Our source file `error_check.cpp` is used to compute the MAE for each distinct k-mer count individually for the simulated data. This to ensure our evaluation captures the variations across different frequency distributions. For the biological data, the MAE will be calculated for bins with different intervals of k-mers frequency. This is because k-mers occur less frequently, resulting in a sparser distribution for higher frequencies.

To obtain ground truth k-mer counts, our evaluation method iterates over the FASTA file obtained by Jellyfish, containing k-mers and their real abundance. As the lines in the file come in pairs, each containing a k-mer and its real abundance, the algorithm first stores the real abundance from the first line in a variable. Then, the k-mer from the second line is passed to the f_{min} function to predict its abundance. The absolute error between predicted and real abundances is then evaluated. This process continues until the end of the file is reached and ends with the algorithm calculating the MAE over the sum of k-mers for each distinct real abundance.

5. Results and Discussion

This chapter presents the findings of our analysis, which aims to investigate the performance and efficacy of the f_{min} predictor function in predicting k-mer abundances. The results will be interpreted, discussions of their implications will be made and possible explanations for the observed patterns.

5.1. Predictive Performance of the f_{min} Function

The MAE of the data is evaluated across different s values in order to analyse the extent to which the distance between the s-mer and the target k-mer impacts predicted abundance. Additionally, we examine the distribution of k-mers counts across the datasets. This for both actual and predicted abundance to gain a better understanding of the underlying patterns of the genomic sequences. Through these analyses, the aim is to gain insights into the performance of the predictor function.

The choice of s is crucial, as, at least for randomly evolving sequences, it needs to be large enough to cover the majority of possible s-mers within the data. For example, considering a random string of 60 million nucleotides, selecting s as 13 ensures coverage of all possible combinations. This is because $4^{13} \approx 67$ million, thereby covering all potential combinations. This consideration is essential because if the s value is too small, there will be few to none unique s-mers in the count vector. With only 4^s possible s-mers in a sequence, it becomes difficult for the predictor function to identify unique k-mers accurately. Refer to Appendix A.1 for an illustration of the consequences of choosing a small s value. Given the size of typical biological datasets, we will not consider values smaller than 10. There are also hardware constraints limiting us to $s \leq 15$. This is because the vector storing the s-mers will have a size of 4^{15}, and each slot holds a 32-bit (4-bytes) integer. This results in $4^{15} \cdot 4 \approx 4.3$ GB of RAM needed to keep it in memory, and the laptop being used is restricted to 8 GB of RAM.

5.1.1. Simulated Data Analysis

This section analyses the performance of the f_{min} function for predicting k-mers using simulated data. The results include the MAE across different values of s and examination of the k-mer distribution for both the actual and predicted abundance. These results are generated using the source codes `simulate_error.cpp` and `create_histogram.cpp` and are presented in Figure 5.1 and 5.2. The chosen values of s have been selected based on the sequences length, which contains approximately 100 000 nucleotides. This resulted in the chosen s values ranging from 10 to 15. Note that all figures are plotted with a log-scaled y-axis and the maximum count is set to 50 because we simulated 50 sequences and repeats within a random sequence is extremely unlikely.

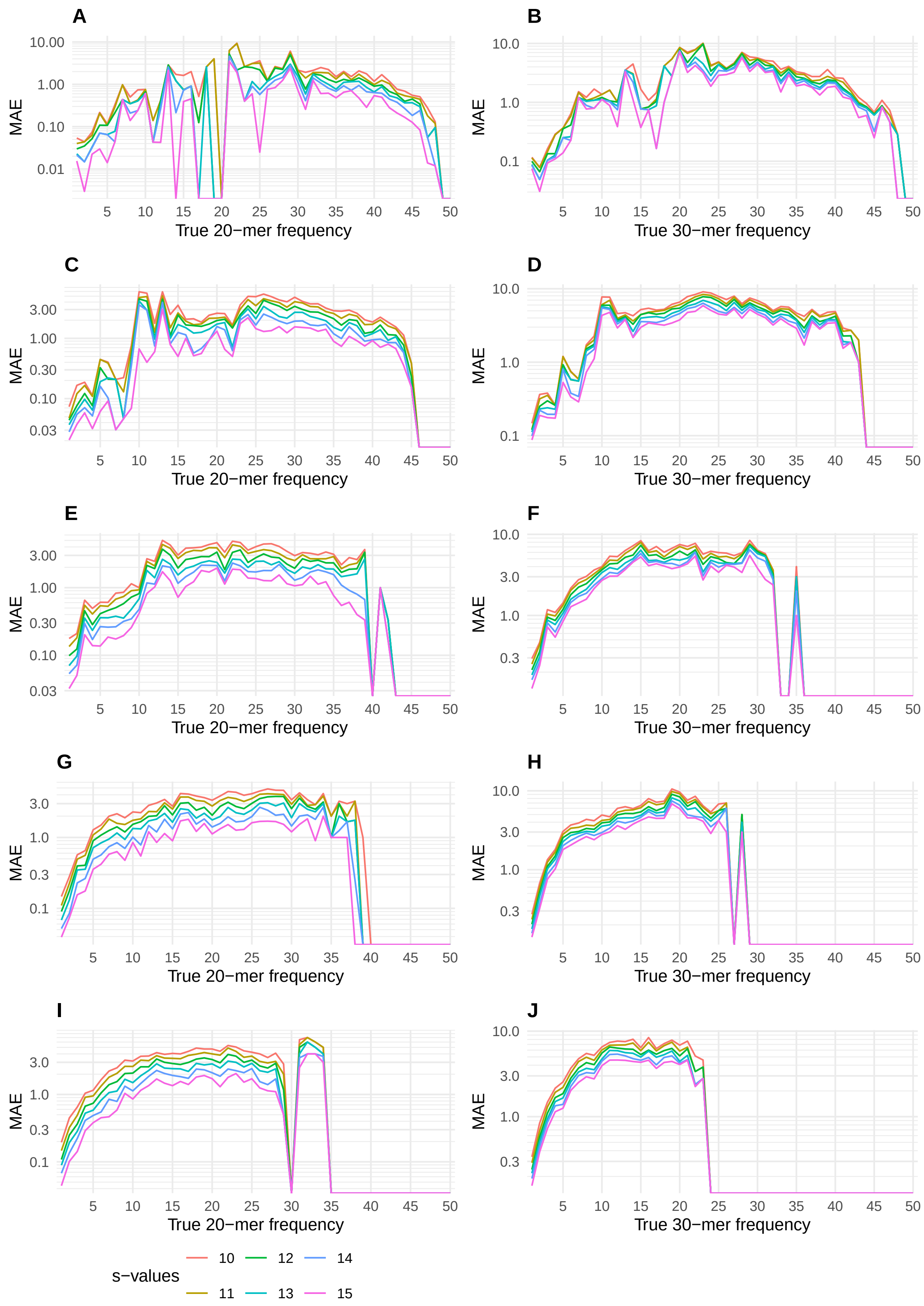

Figure 5.1 The MAE for $k = 20$ (first column) and 30 (second column). (**A**) and (**B**), (**C**) and (**D**), (**E**) and (**F**), (**G**) and (**H**) and (**I**) and (**J**) represents rates from 1% to 5% respectively.

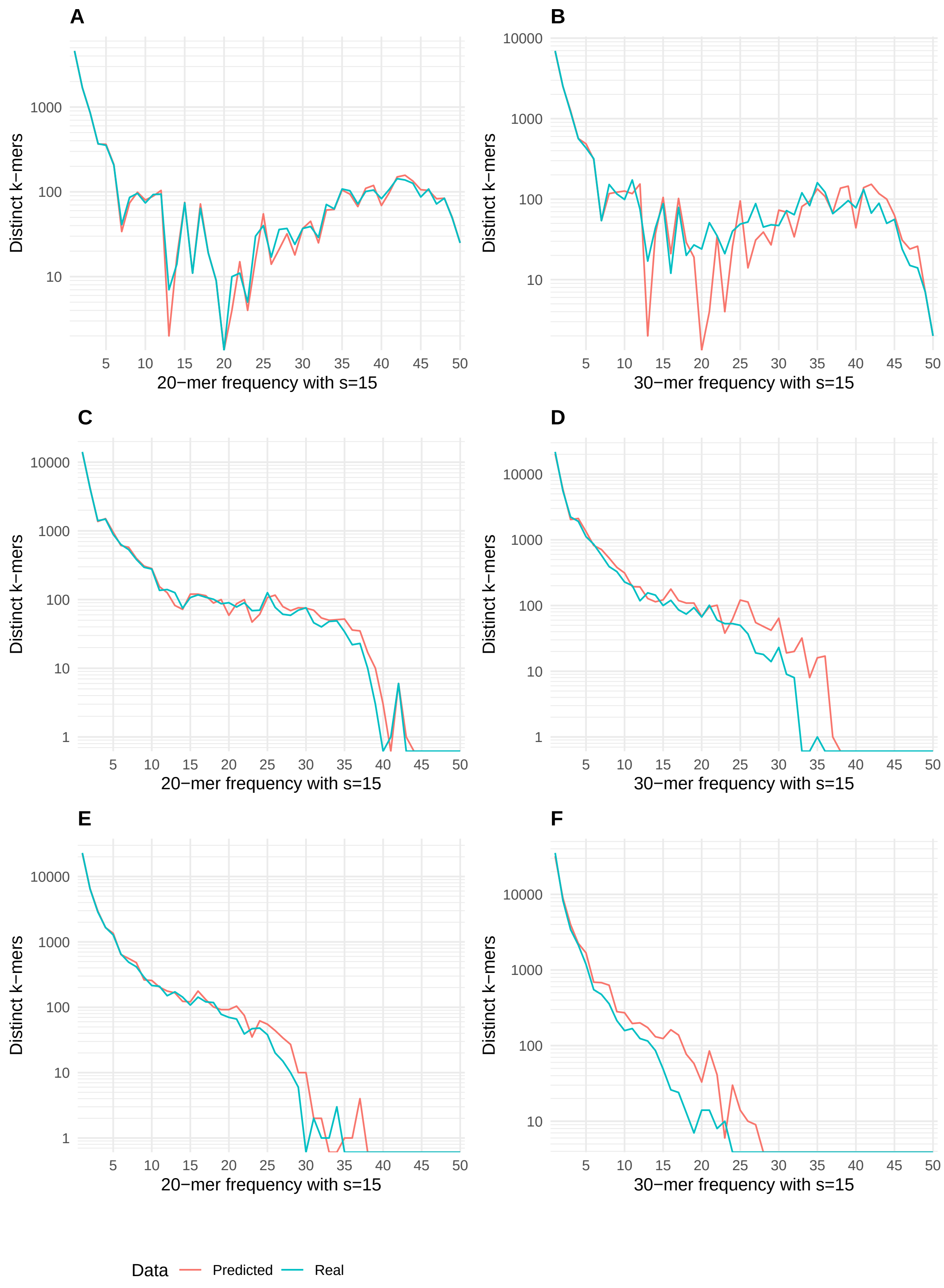

Figure 5.2 Shows the distribution for the k-mers with fixed size for s. (**A**) and (**B**), (**C**) and (**D**) and (**E**) and (**F**) represents rates 1%, 3% and 5% respectively.

Each figure in Figure 5.1 shows that the MAE decreases as the value of s becomes closer to the value of k. This observation lines up with our expectations, given that our method relies on shorter substrings of k-mers, greater accuracy in our predictions are achieved when s approaches the actual length of k. Furthermore, while larger values of s contribute to greater accuracy, larger values of k result in decreased accuracy. This suggests that there exists a trade-off between memory and accuracy in selecting optimal sizes for s and k to achieve optimal performance.

Another notable pattern is that k-mers appear less frequently as the mutation rate increases, as well as when the value of k increases. This happens because higher mutation rates creates more unique patterns within a sequence, resulting in a greater number of unique k-mers. Since there are 4^k possible k-mers, the likelihood for any specific k-mer appearing decreases exponentially as k increases, further increasing the number of distinct k-mers.

Additionally, every figure shows that accuracy stays high for k-mers occurring only once, but gradually decreases as they become more frequent. For almost every figure in Figure 5.1, the MAE stays below one for k-mers that appear no more than five times. Comparing this with Figure 5.2 showing the k-mer distribution, we notice that the majority of the k-mers are distributed below this point. This indicates that it is more difficult for the predictor function to predict k-mers of high-frequency, but gives better accuracy for low-frequency k-mers. Given that the majority of the k-mers appear less frequently, the predictor function shows good accuracy for the majority of the dataset. For sequence mapping applications, this outcome is desirable, as they want unique k-mers for confident mapping.

5.1.2. Biological Data Analysis

This section extends the evaluation to biological data. Specifically, it evaluates the performance of the f_{min} function on the following two datasets, the ChrY and 20 genomes of E.coli. Similar to the analyses of the simulated data, the MAE results will be included and the analyse of the k-mer distribution to understand the predictions. In both datasets, the choice for s will range from 13 to 15 to cover the majority of possible combinations of s-mers. As mentioned in Section 4.2.2, the k-mers are sparser distributed for higher frequencies. Therefore, they are gathered in intervals for k values over ten and presented in a barplot. All plots have a log-scaled y-axis.

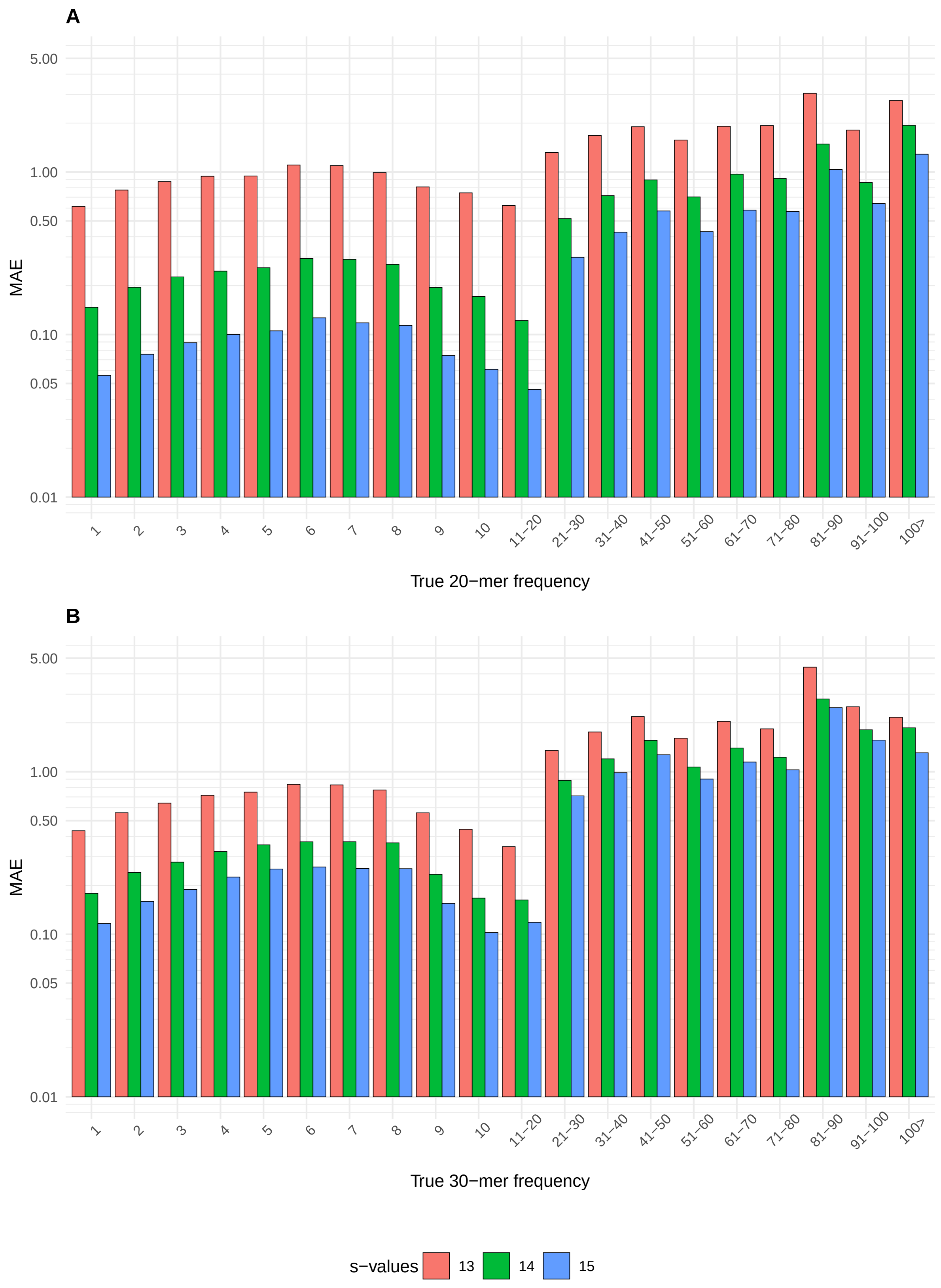

Figure 5.3 The MAE between the real and predicted abundance for different k-mer frequencies for the E.Coli dataset.

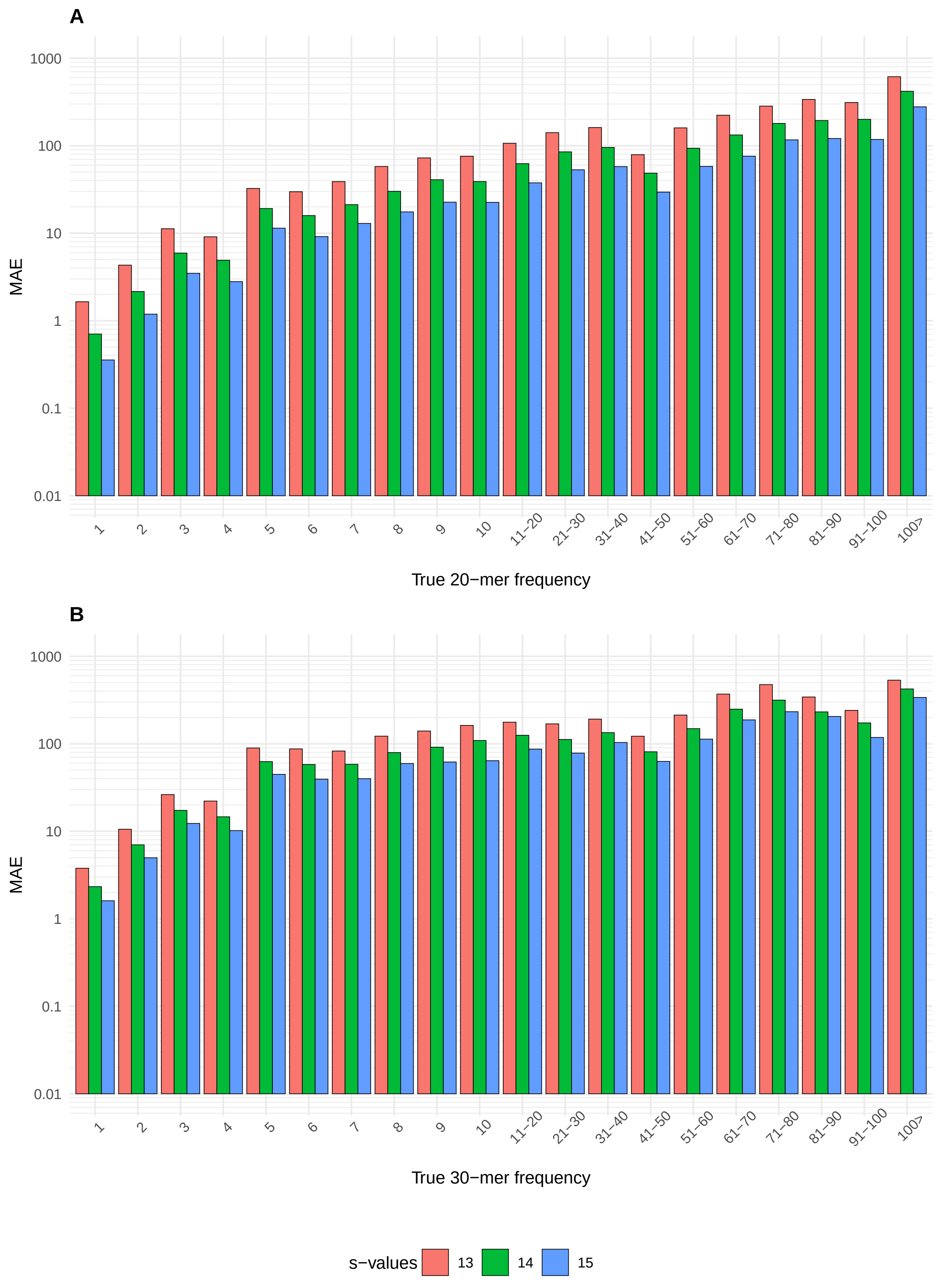

Figure 5.4 The MAE between the real and predicted abundance for different k-mer frequencies for the ChrY dataset.

Figure 5.3 and 5.4 shows similar results as the simulated data, where the MAE decreases as s becomes closer to the value of k. Looking at Figure 5.3, we see that even for k-mers that occur frequently, the MAE does not exceed five and it remains low for k-mers that do not occur frequently. The ChrY dataset, on the other hand, appears to have challenges with regards to accuracy. For unique k-mers, the average MAE exceeds one and rapidly increases as the frequency of k-mers increases. Compared to the E.coli dataset, the maximum MAE for the ChrY dataset is significantly higher, exceeding it by a hundred times. This is because the ChrY dataset has a lot of repetitive sequences, unlike the E.coli dataset. This result indicates that the f_{min} function is sensitive to repetitive sequences.

In Figure 5.5, we examine the behavior of the MAE as k increases, while keeping s fixed to 15. The error nearly always increases as k increases. This would suggest that the s-mers are becoming less representative of the targeted k-mer. Since the value for s is fixed, all the k-values in the plot are based on the same s-mer vector. When k increases, more unique k-mers are present in the data. However, because the counts in the s-mer vector remains the same, the newly generated k-mers previously had a non-unique count and since the s-mer vector can only store 4^s unique s-mers, the larger amount of unique k-mers finds it harder to be accurately represented within this smaller vector.

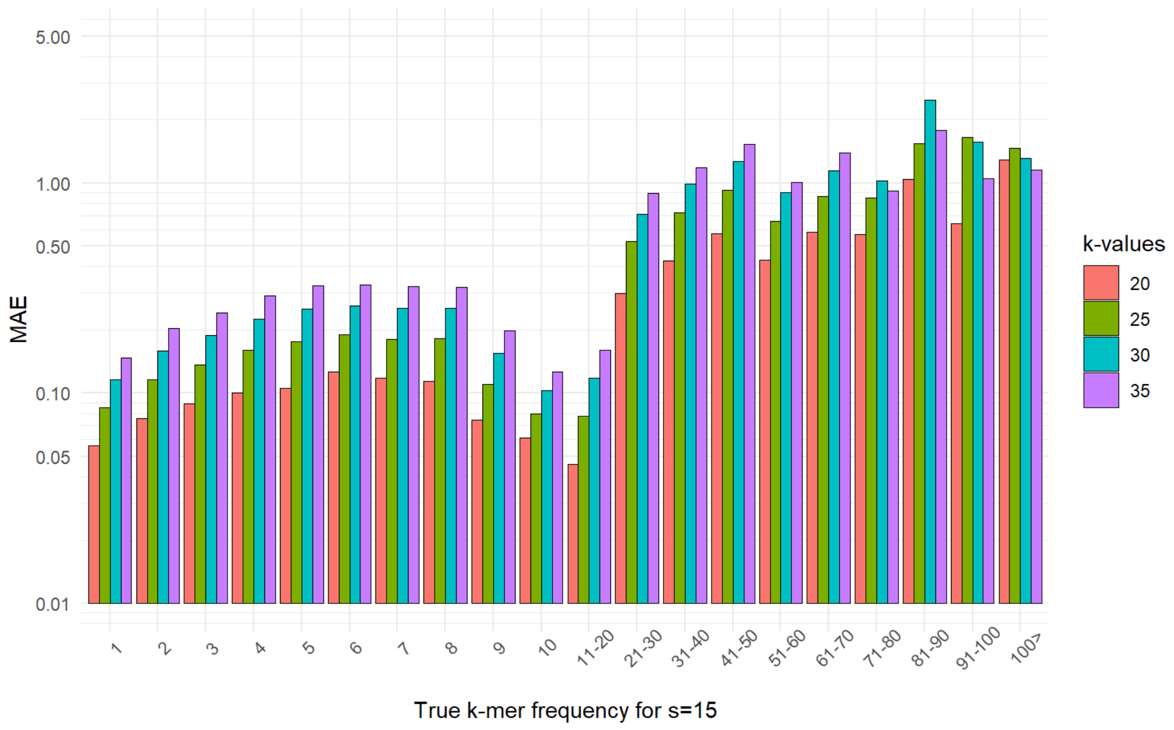

Figure 5.5 The MAE between the real and predicted abundance for different k-mer frequencies with fixed s for the E.Coli dataset.

6. Conclusion

In this thesis, we have explored a substring based method for predicting k-mer abundance, namely our implemented f_{min} function. The f_{min} predictor function shows promising accuracy, particularly for low-frequency k-mers and datasets with low repetition, which is of particular interest in many applications that aims to filter out abundant k-mers. Key observations reveal that as the length of s approaches the actual length of k, accuracy improves, indicating a trade-off between memory and accuracy. Notably, accuracy remains high for k-mers occurring only once but gradually decreases with increasing frequency. This suggests that the function is more capable of predicting rare k-mers, which is desirable for sketching-based methods that subsample low-frequency k-mers. Additionally, higher mutation rates lead to increased uniqueness of k-mers, influencing accuracy trends. However, challenges arise with repetitive datasets, where the function shows sensitivity, as seen in the ChrY dataset.

All this considered, the f_{min} function offers reliable predictions for k-mers with low abundance in low-repetitive datasets, making it a possible candidate for k-mer applications where unique k-mers are of interest. However, further optimization is required for handling repetitive sequences and high-frequency k-mers effectively.

6.1. Future Research

This thesis has evaluated a baseline approach using a minimum predictor function for k-mer abundance prediction. Moving forward, future research could focus on improving the predictor function's performance. For instance, the f_{min} function could be improved by considering all cyclic s-mers within a k-mer. This approach includes more s-mers, potentially leading to better minimum prediction. However, this is not guaranteed because the increased count in the vector (due to collisions) could raise the minimum value in each slot. To further investigate the effectiveness of substring based methods in k-mer prediction, future research could explore more advanced approaches that takes into account the sequence information within the k-mer. For instance, using machine learning methods like random forests or neural networks could help. These methods could understand complex patterns within a k-mer and potentially lead to better predictions.

Additionally, as this thesis only examined one distance metric, the MAE, exploring alternative ones or improving the existing one could further enhance the accuracy and reliability of k-mer abundance prediction. An alternative distance metric could be to use variance or standard deviation instead of mean to measure accuracy of our method. Moreover, we limited ourself to simulated data of size 100 000. While the datasets provided us valuable insight into the performance of the f_{min} function, increasing the size of the simulation could capture additional signals and uncover patterns or trends that were not apparent in smaller datasets.

Bibliography

[1] Mohammed Alser et al. “Technology dictates algorithms: recent developments in read alignment”. In: *Genome Biology* 22.1 (2021), p. 249.

[2] Ardeshir Bayat. “Science, medicine, and the future: Bioinformatics”. In: *BMJ: British Medical Journal* 324.7344 (2002), p. 1018.

[3] Sam Behjati and Patrick S Tarpey. “What is next generation sequencing?” In: *Archives of Disease in Childhood-Education and Practice* 98.6 (2013), pp. 236–238.

[4] Benny Chor et al. “Genomic DNA k-mer spectra: models and modalities”. In: *Genome Biology* 10 (2009), pp. 1–10.

[5] Phillip EC Compeau, Pavel A Pevzner, and Glenn Tesler. “How to apply de Bruijn graphs to genome assembly”. In: *Nature Biotechnology* 29.11 (2011), pp. 987–991.

[6] Arun Das and Michael C Schatz. “Sketching and sampling approaches for fast and accurate long read classification”. In: *BMC Bioinformatics* 23.1 (2022), p. 452.

[7] Sebastian Deorowicz, Agnieszka Debudaj-Grabysz, and Szymon Grabowski. “Disk-based k-mer counting on a PC”. In: *BMC Bioinformatics* 14.1 (2013), pp. 1–12.

[8] Victor Kunin et al. “A bioinformatician’s guide to metagenomics”. In: *Microbiology and Molecular Biology Reviews* 72.4 (2008), pp. 557–578.

[9] Xingyu Liao et al. “Repetitive DNA sequence detection and its role in the human genome”. In: *Communications Biology* 6.1 (2023), p. 954.

[10] Swati C Manekar and Shailesh R Sathe. “A benchmark study of k-mer counting methods for high-throughput sequencing”. In: *GigaScience* 7.12 (2018), giy125.

[11] Guillaume Marçais and Carl Kingsford. “A fast, lock-free approach for efficient parallel counting of occurrences of k-mers”. In: *Bioinformatics* 27.6 (2011), pp. 764–770.

[12] Pall Melsted and Jonathan K Pritchard. “Efficient counting of k-mers in DNA sequences using a Bloom filter”. In: *BMC Bioinformatics* 12 (2011), pp. 1–7.

[13] University of Michigan. *What is FASTA format?* Accessed on April 30, 2024. URL: `https://zhanggroup.org/FASTA/`.

[14] Kai-Oliver Mutz et al. “Transcriptome analysis using next-generation sequencing”. In: *Current Opinion in Biotechnology* 24.1 (2013), pp. 22–30.

[15] Rute Pereira, Jorge Oliveira, and Mário Sousa. "Bioinformatics and computational tools for next-generation sequencing analysis in clinical genetics". In: *Journal of Clinical Medicine* 9.1 (2020), p. 132.

[16] Mihai Pop, Steven L Salzberg, and Martin Shumway. "Genome sequence assembly: Algorithms and issues". In: *Computer* 35.7 (2002), pp. 47–54.

[17] Arang Rhie et al. "The complete sequence of a human Y chromosome". In: *Nature* 621.7978 (2023), pp. 344–354.

[18] Deyou Tang et al. "KCOSS: an ultra-fast k-mer counter for assembled genome analysis". In: *Bioinformatics* 38.4 (2022), pp. 933–940.

[19] Axel Wedemeyer et al. "An improved filtering algorithm for big read datasets and its application to single-cell assembly". In: *BMC Bioinformatics* 18 (2017), pp. 1–11.

[20] Roland Wittler. "General encoding of canonical k-mers". In: *Peer Community Journal* 3 (2023).

[21] Hongyi Xin et al. "Accelerating read mapping with FastHASH". In: *BMC Genomics*. Vol. 14. Springer. 2013, pp. 1–13.

Appendix A: Impact of Small *s*-Values on MAE For *k*-mer Prediction

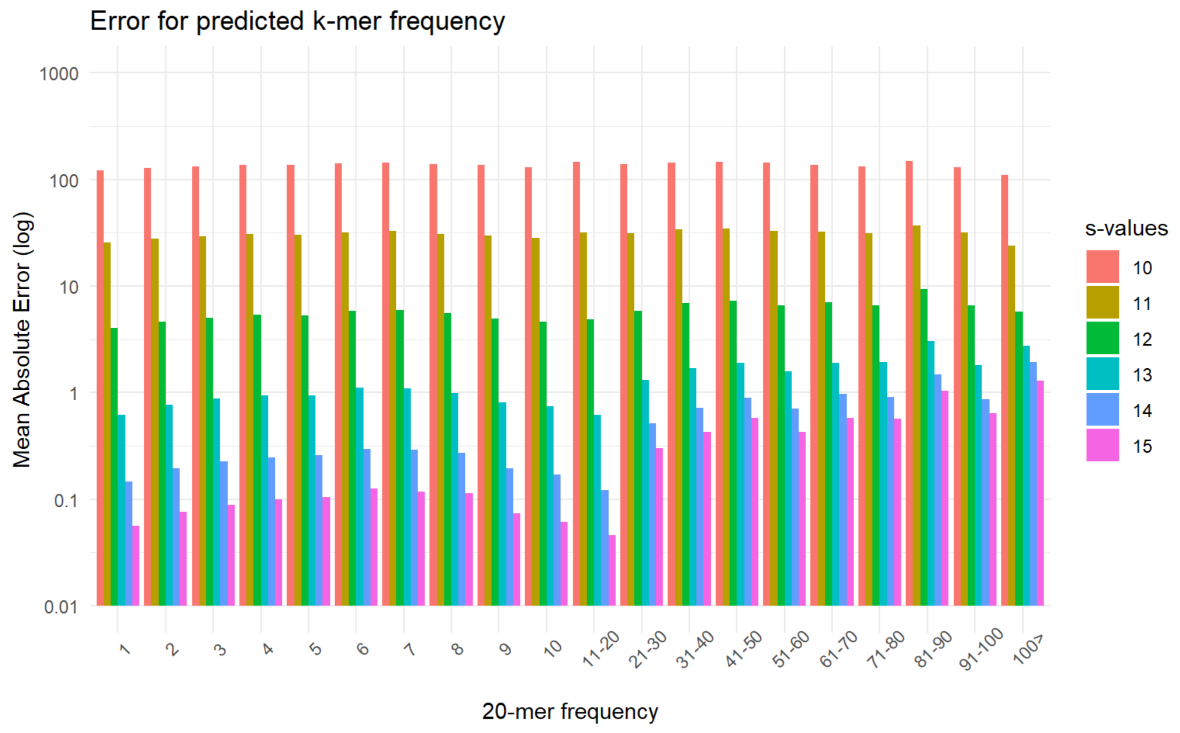

Figure A.1 MAE for E.Coli 20 genomes with too small s-values. The figure shows that for $s = 10$ the MAE is larger than 100, meaning that the predicted value is a hundred times larger than the real abundance. This is caused by the s-mer vector not having a large enough number of combinations to cover the desired k-mer.

Datalogi
www.math.su.se

Beräkningsmatematik
Matematiska institutionen
Stockholms universitet
106 91 Stockholm