Comenius University, Bratislava
Faculty of Mathematics, Physics and Informatics

# Approximate Abundance Histograms and Their Use for Genome Size Estimation

Bachelor Thesis

2017
Mário Lipovský

Comenius University, Bratislava
Faculty of Mathematics, Physics and Informatics

# Approximate Abundance Histograms and Their Use for Genome Size Estimation

Bachelor Thesis

| | |
|---|---|
| Study programme: | Informatics |
| Study field: | 2508 Informatics |
| Department: | Department of Computer Science |
| Supervisor: | Mgr. Bronislava Brejová, PhD. |

Bratislava, 2017
Mário Lipovský


Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

# THESIS ASSIGNMENT

| | |
|---|---|
| **Name and Surname:** | Mário Lipovský |
| **Study programme:** | Computer Science (Single degree study, bachelor I. deg., full time form) |
| **Field of Study:** | Computer Science, Informatics |
| **Type of Thesis:** | Bachelor´s thesis |
| **Language of Thesis:** | English |
| **Secondary language:** | Slovak |

**Title:** Approximate Abundance Histograms and Their Use for Genome Size Estimation

**Aim:** Technologies for DNA sequencing produce many short sequencing reads coming from a much longer genome. One way of succinctly summarizing this data is to compute a histogram of the number of occurrences of substrings of a fixed length in an input set of sequencing reads. Such histograms can be then used for example to estimate the size of the target genome. Some methods estimate histograms only approximately to reduce the memory required for the computation. The goal of the thesis is to select a method for histogram estimation and to explore the nature of its errors and how these errors influence the accuracy of the genome size estimate.

| | |
|---|---|
| **Supervisor:** | doc. Mgr. Bronislava Brejová, PhD. |
| **Department:** | FMFI.KI - Department of Computer Science |
| **Head of department:** | prof. RNDr. Martin Škoviera, PhD. |
| **Assigned:** | 26.10.2016 |
| **Approved:** | 26.10.2016 |

doc. RNDr. Daniel Olejár, PhD.
Guarantor of Study Programme

.....
Student

.....
Supervisor


Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

# ZADANIE ZÁVEREČNEJ PRÁCE

**Meno a priezvisko študenta:** Mário Lipovský
**Študijný program:** informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
**Študijný odbor:** informatika
**Typ záverečnej práce:** bakalárska
**Jazyk záverečnej práce:** anglický
**Sekundárny jazyk:** slovenský

**Názov:** Approximate Abundance Histograms and Their Use for Genome Size Estimation
*Približné histogramy počtu výskytov a ich použitie na odhad veľkosti genómu*

**Cieľ:** Prístroje na sekvenovanie DNA vedia čítať len pomerne krátke úseky genómu. Tieto dáta môžeme zosumarizovať vo forme histogramu počtu výskytov jednotlivých podslov pevnej dĺžky v danej sade sekvenačných čítaní. Takéto histogramy sa dajú využiť napríklad na odhad dĺžky pôvodného genómu. Niektoré metódy počítajú histogram iba približne, s cieľom ušetriť množstvo pamäte potrebnej na výpočet. Cieľom práce je zvoliť si metódu na odhad histogramu a preskúmať, aké typy chýb robí a ako tieto chyby ovplyvnia presnosť výsledného odhadu veľkosti genómu.

**Vedúci:** doc. Mgr. Bronislava Brejová, PhD.
**Katedra:** FMFI.KI - Katedra informatiky
**Vedúci katedry:** prof. RNDr. Martin Škoviera, PhD.

**Dátum zadania:** 26.10.2016

**Dátum schválenia:** 26.10.2016 doc. RNDr. Daniel Olejár, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

# Acknowledgement

I would like to thank my supervisor doc. Mgr. Bronislava Brejová, PhD. for her advice, optimism and openness, which made my work on this thesis more enjoyable.

# Abstrakt

Sekvenovaním DNA zvyčajne vzniká množstvo krátkych reťazcov. Sekvenovacie dáta môžeme zosumarizovať vo forme histogramu počtov výskytov jednotlivých podslov pevnej dĺžky. Takéto histogramy sa dajú použiť napríklad na odhadovanie dĺžky genómu. V našej práci skúmame algoritmus Kmerlight, ktorý počíta spomínaný histogram približne. Zistili sme, že Kmerlight počíta vychýlené odhady histogramov, no podarilo sa nám navrhnúť novú verziu algoritmu Kmerlight, ktorej odhady sú už nevychýlené. V práci ďalej teoreticky modelujeme pravdepodobnostné rozdelenie chýb odhadov histogramu a pomocou experimentov sme overili správnosť nášho modelu. Na záver sme použili program CovEst na výpočet odhadov dĺžok genómov z približných histogramov a preskúmali sme, ako chyby v histogramoch ovplyvňujú presnosť týchto odhadov. Napriek tomu, že CovEst bol navrhnutý na spracúvanie presných histogramov, naše výsledky ukazujú, že CovEst môže byť použitý aj na približných histogramoch, ktorých výpočet si vyžaduje menšie množstvo pamäte.

**Kľúčové slová:** histogram počtov $k$-tic, Kmerlight, odhad dĺžky genómu, CovEst

# Abstract

DNA sequencing data is typically a large collection of short strings called reads. We can summarize such data by computing a histogram of the number of occurrences of substrings of a fixed length. Such histograms can be used for example to estimate the size of a genome. In this thesis we study an existing tool, Kmerlight, which computes approximate histograms. We discover an approximation bias, and we propose a new, unbiased version of Kmerlight. We also model the distribution of approximation errors, and we support our theoretical model by experimental data. Furthermore, we use another tool, CovEst, to compute genome size estimates with use of approximate histograms, and we evaluate the precision of such estimates compared to results obtained from exact histograms. Our results show that although CovEst was designed to work with exact histograms, it can be used with their approximate versions, which can be produced in a much smaller amount of memory.

**Keywords:** $k$-mer abundance histogram, Kmerlight, genome size estimation, CovEst

# Contents

# List of Figures

# List of Tables

# Introduction

In a DNA sequencing experiment, many reads are produced from a genome. These reads are short strings obtained from random locations of the genome, potentially with some sequencing errors. For a genome to be analyzed, the genomic sequence must be first assembled from the reads, but the process of genome assembly is computationally demanding.

In previous years, several methods were devised to estimate the size of a genome and some other genome characteristics without the need of genome assembly [6, 2, 9, 12]. To compute these estimates, a summary statistic of reads, $k$-mer abundance histogram is used. In order to produce such a histogram, the sequencing reads are first processed into $k$-mers (substrings of a length $k$). Then the histogram of the number of occurrences of $k$-mers can be computed.

Most of the histogram computing methods are based on hash tables or suffix arrays [8, 7, 11, 3] and their memory usage increases at least linearly with the number of processed distinct $k$-mers. To reduce the memory requirements, $k$-mer abundance histograms can be computed approximately. One of the newest such algorithms, Kmerlight [12], combines the techniques of sampling and hashing to maintain a sketch of $k$-mers and from the contents of the sketch computes an estimate of the histogram.

The approximate histograms were already used as an inputs for genome size estimation tools [12], however the impact of the approximation errors on estimate precision was not evaluated.

The goal of our thesis is to study the character of errors of the approximate histograms and their influence on the subsequent genome size estimation.

We start with an empirical study of the approximation errors of Kmerlight algorithm, and we discover that Kmerlight produces systematically biased estimates of some histograms. We explain the source of the bias and mathematically support our claims, and then we also propose a modification of Kmerlight which eliminates this bias.

Next we model the distribution of Kmerlight's errors with the normal distribution, and we propose a formula that describes Kmerlight's variance. We then experimentally test our theoretical model and explore its limitations.

Finally, we use CovEst software [6] to estimate the sizes of simulated genomes from approximate histograms produced by Kmerlight. We describe how different parameters of the genome influence the accuracy of the estimates and we compare the estimates based on exact histograms to the estimates based on approximate histograms.

In the first chapter we explain the essential biological vocabulary and we outline the principles of histogram computing and of genome size estimation. In the second chapter we study the approximation errors made by Kmerlight and in the last chapter we investigate the accuracy of genome size estimates based on approximate histograms.

# Chapter 1

# Background and Problem Statement

In this chapter we firstly explain the biological context and the reason why is the genome size estimate necessary. Then we define a $k$-mer abundance histogram and present algorithms which compute this histogram exactly and approximately. Finally, we briefly describe a method for genome size estimation and we will be able to present the goals of our thesis.

## 1.1 Biological Motivation

### 1.1.1 DNA Sequencing

DNA molecules store genetic information in cells in the form of long linear or circular chains of nucleotides. Four types of nucleotides can be present in DNA – adenine, cytosine, guanine and thymine. Therefore, we can represent a DNA molecule as a sequence of characters `A, C, G, T`.

The whole genetic information of an organism, stored in several DNA molecules, is called a genome. Genome sizes range from thousands of nucleotides in viruses up to hundreds of billions nucleotides in some plants. The length of a human genome is known to be around 3.2 billion nucleotides.

In a sequencing process we try to obtain a nucleotide sequence of DNA from a given biological sample. We will focus on the next generation sequencing technologies that can read only short segments of a nucleotide chain, producing reads of length 100–1000 nucleotides (depending on the specific technology used).

In a sequencing experiment, the DNA is at first randomly fragmented to shorter pieces. Then the ends of these pieces are sequenced, producing two reads from one fragment. Therefore, each read originates from a specific but unknown location in the genome. Since every read is a segment of DNA, it can also be represented as a string of `A, C, G, T` characters.

There are multiple factors that affect the quality of reads, but we will mostly consider only one type of errors: a single nucleotide substitutions caused by sequencer itself. Occasionally a nucleotide of DNA is read incorrectly and so it is considered to be one of three other nucleotides. We use the term error rate to quantify the fraction of incorrectly read nucleotides. For example, if the sum of lengths of all reads is $10^9$ and the error rate is 0.02, we can expect $2 \times 10^7$ nucleotides in the reads to be incorrect.

### 1.1.2 Genome Assembly Problems

When the reads are retrieved, it is then the task for bioinformatics to assemble the whole genome from many short fragments. The genome assembly relies on the overlaps of reads such as in Figure 1.1: if a suffix of one read is identical to a prefix of another read, they presumably originate from overlapping locations in the genome, and thus can be joined to form a longer sequence.

```
genomic sequence:    CCGACGTCCACCTGTGATCGGATG
          read A:         GTCCACCT
          read B:             ACCTGTGA
```

Figure 1.1: Two reads of length 8 overlapping one another in 4 nucleotides.

Without a large enough number of reads, the reconstruction of genome would not be possible. Not only that the reads should cover the whole length of the genome, but there should also be a sufficient number of read overlaps. In other words, each location in the genome should be a part of multiple reads. The average number of reads covering each nucleotide is called coverage.

Since the source locations of reads are randomly distributed over the genome, even at high coverage some areas of the genome can remain unsequenced, thus preventing complete assembly.

In order to estimate the number of reads necessary for a certain coverage, we must first estimate the genome length. We can base the estimate on the known genomes of related species. A different approach would be to sequence the genome in a preliminary experiment, estimate the coverage and the genome length and then continue the sequencing process until a desired coverage is achieved.

But the genome assembly problems do not end with sufficiently high coverage. For example, during the evolution, long segments of DNA can be copied to different locations, creating multiple identical sequences in different parts of genome.

In section 1.4 we will outline a method that can estimate the genome size and also other characteristics such as coverage and extent and multiplicity of duplicated regions without the need of genome assembly, but this method uses a summary statistic called $k$-mer abundance histogram.

## 1.2 K-mer Abundance Histogram

To obtain information from reads, the reads are at first processed into $k$-mers. A $k$-mer is a substring of length exactly $k$. Using a read of length $r$ we produce $r-k+1$ $k$-mers. The same $k$-mer may be produced by multiple reads. If a $k$-mer occurs $i$ times among all $k$-mers, we say its abundance is $i$.

**Definition.** *The $k$-mer abundance histogram is a sequence $f = f_1, f_2, f_3, \ldots f_m$, where $f_i$ is the number of $k$-mers that occur in the input set exactly $i$ times and $m$ is the maximum observed abundance.*

Apart from estimating the genome size, several other methods in bioinformatics are also based on $k$-mers. For example, during genome assembly, the overlaps of $k$-mers are usually considered instead of overlaps of reads. As the $k$-mer abundance histogram can be calculated quickly, it can be also used for selection of an optimal $k$ in such methods [1].

Also note that the $k$-mer abundance histogram and many algorithms used for its computation can be generalized to a histogram of any input items. Thus the applicability of this topic outreaches the field of bioinformatics.

The value of $k$ must be set high enough to reduce the chance of two unrelated genome locations from producing the same $k$-mers[1]. However, with higher values of $k$, each sequencing error affects more $k$-mers, thus increasing the fraction of erroneous $k$-mers which produce problems in the downstream analysis. In our thesis we always use the value $k = 21$ as suggested in [2, 6].

## 1.3 Computing K-mer Abundance Histogram

A simple approach to compute $k$-mer abundance histogram would be firstly to compute $k$-mer abundances – the exact number of occurrences of each $k$-mer – and then count different $k$-mers with $j$ occurrences for each $j$. Since the role of $k$-mer abundances is important in bioinformatics, there exist many tools that compute them. In the next subsection (1.3.1) we briefly summarize these algorithms.

However, counting $k$-mer abundances is a computationally demanding task for large inputs. As we are only interested in the histogram, the problem of $k$-mer counting can be avoided, allowing us to estimate the histogram very efficiently without an intermediate step.

[1] Under the assumption that each of `A, C, G, T` nucleotides occurs at each position with probability 1/4, we can expect approximately $L^2 \cdot 4^{-k}$ *collisions* in a genome of size $L$.

### 1.3.1 Exact K-mer Abundance Counting

In a naive hashing algorithm, a hash function $h$ would uniformly distribute strings of length $k$ to a hash table $T$. We would store the number of occurrences of a $k$-mer $s$ in the counter $T[h(s)]$. In a single scan through all the reads we would then increment the appropriate counters. This solution works well for millions of $k$-mers, but as the number of distinct $k$-mers increases, we must use larger hash tables in order to prevent collisions. This solution becomes much slower when the hash table $T$ becomes larger than RAM available.

A few techniques were used to decrease the time and memory consumption. These improvements allowed the hashing approach to be used in practice:

- Based on an observation that most of the $k$-mers with only one occurrence come from erroneous reads, BFCounter [8] uses a Bloom filter to exclude rare $k$-mers from the hash table, thus saving memory.
- Jellyfish [7] software uses a thread-safe hash table utilizing the advantage of parallel computing.
- To decrease the size of the hash table, Disk Streaming of K-mers (DSK) [11] algorithm scans the input data in more iterations, processing only a subset of $k$-mers in each iteration. A second hash function is used to determine the subset (and the iteration) for each $k$-mer (a similar principle is used to randomly sample $k$-mers in section 1.3.2).

A different, but still memory-demanding, approach based on suffix arrays was used in Tallymer software [3]. A suffix array is a data structure holding all suffixes of a string sorted in a lexicographical order. Suffixes with identical prefixes of length at least $k$ represent different occurrences of a $k$-mer. Since they are stored in a sorted order, abundances of $k$-mers can be computed by grouping adjacent suffixes.

### 1.3.2 Approximating the Histogram

As we drop the requirement to compute the exact histogram, it is no longer necessary to store the abundance of each $k$-mer. This provides a way to reduce the amount of required memory from dozens of gigabytes to hundreds of megabytes, allowing these computations to be performed on a personal computer rather than on a cluster.

To use all available data, we still analyse every $k$-mer of every read, so the time complexity of the following algorithms will still be at least linear in the number of $k$-mers.

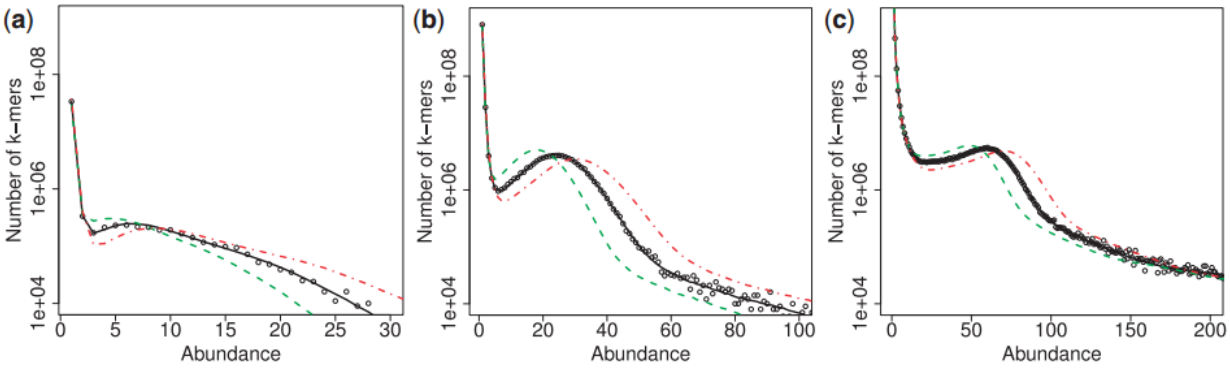


Figure 1.2: The accuracy of the sampling method. Reprinted from the original paper [1]. The panels reflect three datasets. Each plot shows the exact histogram curves for $k = 51$ (solid black curve), $k = 41$ (dash-dot red curve) and $k = 61$ (dashed green curve). The approximate (sampled) histogram is shown using black dots. Note that $f_i$ is shown on a log-scale, exaggerating the differences at lower $f_i$ values.

### Simple sampling from $k$-mers

The simplest optimization, which was used in a tool KmerGenie [1], is to sample from $k$-mers. With the parameter $s$, we can choose a hash function $\rho_s : \{A, C, G, T\}^k \to \{0, 1, \ldots, s-1\}$ that uniformly distributes the $k$-mers to $s$ buckets. Afterwards, we can compute the histogram by a naive hashing approach or by any other algorithm presented in the previous section 1.3.1 using only the $k$-mers that hashed to 0. Of all the distinct $k$-mers, only a randomly selected fraction of $1/s$ is considered.

The authors used value $s = 1000$ in their experiments. As it can be seen from the experimental data (Fig. 1.2), the approximation closely follows the exact histogram $f$ at abundances with higher values of $f_i$, where enough of unique $k$-mers of abundance $i$ were sampled. Fewer $k$-mers reach higher abundances $i$, and thus even fewer of them are sampled, which leads to decreased relative precision of approximation of lower values $f_i$. The authors did not include any analytical bounds of errors, however.

### Multilevel sampling

The inspiration for the next approach comes from a class of streaming algorithms. Streaming algorithms are used to process a sequence of items (in our context $k$-mers) usually in only one pass with a limited memory and time per item. A common problem solved by streaming algorithms is counting distinct elements in a stream [14].

These algorithms maintain an approximate summary or a sketch of the previously viewed $k$-mers and with each new $k$-mer the sketch is updated. When all the $k$-mers are processed, the sketch can be analyzed to provide the estimate of the $k$-mer abundance histogram.

In 2002 Bar-Yossef et al. presented three streaming algorithms that were able to estimate the number of distinct elements in a stream ($F_0 = \sum_{i=1}^{m} f_i$) with theoretical guarantees [4].

In 2014 Melsted and Halldórsson [9] implemented and extended Algorithm 2 from the aforementioned paper [4] and used it for $k$-mer counting. Their algorithm KmerStream was also able to estimate the number of $k$-mers with abundance one, $f_1$. According to the authors, this algorithm is at least three times faster than KmerGenie, and it can use ten times less memory.

KmerStream was further improved by Sivadasan et al. in 2016 [12], and their software Kmerlight is able to estimate the whole histogram $(f_1, f_2, \ldots, f_m)$. The authors also included theoretical bounds of approximation errors.

As we focus on Kmerlight in our work, we will describe Kmerlight in detail in the next chapter, in section 2.1.

## 1.4 Genome Size Estimation

As we mentioned in section 1.1.2, it would be helpful to know the length of a genome before its assembly. Furthermore, we might also be interested in other genome characteristics, such as the extent and multiplicity of its duplicated regions.

In previous years there were several researches concerned with the estimation of these characteristics without the need of computationally demanding genome assembly [6, 2, 9, 12], using only the $k$-mer abundance histogram.

Let us summarize the parameters of the genome in Table 3.3. Then we will describe how different genome characteristics influence the shape of the histogram, and how these parameters can be estimated from the histogram. We use the assumption that the reads are distributed uniformly across the genome.

| | | |
|---|---|---|
| $L$ | genome size | the number of nucleotides in the genome |
| $l$ | read length | the number of nucleotides in one read |
| $c$ | coverage | an average number of reads covering each genome position |
| $e$ | error rate | probability that a single nucleotide was read incorrectly |

Table 1.1: A set of simple genome parameters as described in the section 1.1.1.

When we consider a genome without duplicated sequences and sequencing without errors, we would expect the histogram to reach its maximum close to the abundance $c$. Under the assumption that the reads are uniformly distributed over the genome, most of the locations in genome are expected to be covered by approximately $c$ reads, thus contributing to values close to $f_c$.

Note that the average coverage is a fraction of the sum of lengths of all reads and the genome size ($c = n \cdot l/L$, if $n$ denotes the number of all reads). Since the sum of lengths of reads is known, with the knowledge of the coverage, we can estimate the genome size, and conversely, with the knowledge of the genome size we can estimate the coverage.

If the average coverage of the whole genome is $c$, then a duplicated sequence is expected to be a part of twice as many reads as a unique sequence, and so to have coverage $2c$. A sequence copied three times in the genome is expected to have coverage $3c$. The number of copies present can be thus deduced from the positions of peaks in the histogram, expected at abundances $2c, 3c, \ldots nc$ and the extents of copied regions can be deduced from the relative heights of these peaks $f_{2c}, f_{3c}, \ldots f_{nc}$.

The sequencing error rates also change the shape of the histogram. With error rate of 1% and $k = 21$, about 19% of $k$-mer occurrences are erroneous and thus unique with high probability, creating a notable peak in the histogram at value one. From the height of $f_1$, the error rate can be estimated.

Furthermore, the erroneous $k$-mers also decrease the average effective coverage, as some $k$-mers covering each location are wasted because of errors. As a result, the whole histogram is shifted towards lower abundances.

The aforementioned characteristics lay out the basis of $k$-mer abundance histogram analysis. Approaches introduced in the previous work [6, 2, 9, 12] use probabilistic generative models that can generate the expected shape of the histogram based on several parameters. These parameters are chosen and optimized to produce a histogram most alike to the observed one. We will provide a more detailed description of one such model, CovEst [6, 5], in section 3.1.

## 1.5 Problem Statement

The main goal of our thesis is to determine whether the approximate histograms produced by Kmerlight can be used as inputs for CovEst to estimate the genome size. We mainly focus on preserving the accuracy of CovEst's estimates, and we focus less on the processing time and memory consumption.

In order to achieve our goal, in chapter 2 we study the qualitative and quantitative character of inaccuracies introduced by Kmerlight with the aim to model or predict the distribution of the estimates of $f_i$. We discover that Kmerlight can be modified to produce more accurate histograms and we present a model of Kmerlight's errors distribution.

In chapter 3 we we study the performance of CovEst on approximate histograms. After providing a more detailed description of CovEst software we experimentally eval-

uate the effects of various coverages, error rates and genome sizes on precision of genome size estimates. We find that CovEst is robust with respect to Kmerlight's errors and that CovEst's estimates based on approximate histograms maintain a sufficient precision.

# Chapter 2

# Analysis and Improvement of Kmerlight

In this chapter we first present a detailed description of an existing algorithm Kmerlight [12] which computes an approximated $k$-mer abundance histogram.

Afterwards we investigate the character of inaccuracies of the estimated histogram, and we present a novel estimate of error variance. As we discover a systematic estimation bias, we alter Kmerlight to produce unbiased histogram estimates.

## 2.1 Kmerlight

The input expected by Kmerlight is a collection of $ACGT$ sequences (reads). Kmerlight transforms reads into $k$-mers, as it was described in 1.2 and then processes a stream of $k$-mers. Kmerlight maintains a sketch of previously processed $k$-mers and updates the sketch with each $k$-mer. The estimates of $F_0$ ($F_0 = \sum_{i=1}^{m} f_i$) and $f_i$ are computed from the content of the sketch in the end. The output consists of values $\hat{F}_0, \hat{f}_1, \hat{f}_2, \ldots, \hat{f}_m$.

Kmerlight's sketch consists of $W = 64$ levels. There is a hash table $T_w$ at each level $w$ with $r$ counters $T_w[0], T_w[1], \ldots, T_w[r-1]$. Each counter $c$ stores its value $T_w[c].v$ (the number of elements stored in the counter) and an auxiliary information $T_w[c].p$ from within a range $0, \ldots, u-1$.

### Update

- For a distinct $k$-mer, its level is selected so that the probability of selecting level $w$ is $1/2^w$. In particular, the $k$-mer is hashed into an integer of $W$ bits and the number of trailing zeroes determines the level $w$. Thus all occurrences of the same $k$-mer will be placed to the same level $w$.
- Next, using a different hash function $h : \{A, C, G, T\}^k \to \{0, \ldots, r-1\} \times \{0, \ldots, u-1\}$, the $k$-mer is hashed into a pair $(c, j)$.

- If the counter $c$ at level $w$ is empty, its value is increased to 1 and $j$ is stored as an auxiliary information $T_w[c].p$.
- If the counter $T_w[c]$ is not empty, but the auxiliary information in $T_w[c].p$ is equal to $j$, the counter value $T_w[c].v$ is increased.
- Finally, if $T_w[c].p \neq j$, the counter is marked as dirty with value -1. Dirty counters are not modified by future updates.

Note that all occurrences of the same $k$-mer will be stored in the same counter, and the value of the counter should correspond to the abundance of this $k$-mer. Since two or more different $k$-mers may hash into the same counter at the same level $T_w[c]$, collisions may occur. The auxiliary information helps to detect some of these collisions.

In the analysis we will always assume that both hashing functions perform uniform hashing, though only pairwise independent hashing is used in the implementation of the algorithm.

**Estimator of $F_0$** Since on average $F_0/2^w$ distinct $k$-mers are hashed into level $w$, the probability that a counter at level $w$ remains empty is approximately $p = (1 - \frac{1}{r})^{F_0/2^w}$. In this estimate and in all subsequent analysis we assume that the number of distinct $k$-mers at level $w$ is exactly $F_0/2^w$, although in fact it is a binomial random variable with this number as mean[1].

The expected number of empty counters at level $w$ is thus $r \cdot p$. Let us denote the number of observed empty counters at level $w$ as $t_0$. Using the assumption $t_0 \approx r \cdot p$ we can easily derive the estimator of $F_0$:

$$\hat{F_0} = 2^w \cdot \frac{\ln(t_0/r)}{\ln\left(1 - \frac{1}{r}\right)} \tag{2.1}$$

To estimate the number of distinct $k$-mers, we choose one level of the sketch $w^*$, so that the number of empty counters at this level ($t_0$) is closest to $r/2$. This estimator of $F_0$ was first presented in the article by Bar-Yossef et al. in 2002 [4] and it has been shown that selecting this level provides a bounded error of $\hat{F_0}$ with guaranteed probability.

**Estimator of $f_i$** The expected number of distinct $k$-mers with abundance $i$ hashed to level $w$ is $f_i/2^w$. When a $k$-mer is hashed into level $w$, the probability that it is stored in a collision-free counter is $(1-\frac{1}{r})^{F_0/2^w-1}$, which is the probability that no other

[1] An exact value of $p$ is $(1 - \frac{1}{r \cdot 2^w})^{F_0}$, which considers that any of $F_0$ $k$-mers may cause a collision at level $w$. The approximate value of $p$ is almost equal to the exact value for $r, F_0$ used, however, and the calculations with the approximate $p$ are much simpler.

$k$-mer from level $w$ will get hashed into the same counter. Thus we can estimate the number of collision-free counters with value $i$ as

$$t_i \approx f_i/2^w \cdot \left(1 - \frac{1}{r}\right)^{F_0/2^w - 1} \tag{2.2}$$

If we denote the number of observed collision-free counters with value $i$ as $t_i$, we can derive the estimator of $f_i$:

$$\hat{f}_i = t_i \cdot 2^w \cdot \left(1 - \frac{1}{r}\right)^{1 - F_0/2^w} \tag{2.3}$$

Again, one level $w^*$ is selected to estimate $f_i$, so that it maximizes $t_i$ – the number of observed collision-free counters with value $i$. This decision was not discussed by the authors, but it seems to be a reasonable choice to achieve the highest accuracy, since higher levels would contain fewer counters with value $i$ and lower levels would contain fewer collision-free counters.

**Undetected collisions** We use an assumption that if a counter holds a value $i$, it must originate from a $k$-mer with abundance $i$, but hashing collisions can occur. With the collision detection mechanism in place, most of the counters with collisions are discarded, but some collisions can remain undetected. The value $t_i$ is based on the number of non dirty counters, but these include both true positives (collision-free counters) and false positives (counters with an undetected collision).

The authors have shown that the expected number of false positive at one level is at most $r/u$ and that parameter $u$ can be set to make false positives negligible[2]. As with $2k$ bits per counter we would be able to retain the whole $k$-mer stored in this counter and thus detect all collisions perfectly, the parameter $u$ only provides an effective trade-off between memory usage and accuracy.

In our analysis we will ignore the effect of false positives and we will use an assumption that all collisions are being detected.

**Median amplification** To further decrease the variance of estimates and to make use of multiprocessing, $t$ independent instances of Kmerlight's sketch are run concurrently. Estimate $\hat{F}_0$ is then selected as median of $\hat{F}_0^{(1)}, \ldots, \hat{F}_0^{(t)}$, and estimates of $f_i$ are also selected as medians from $t$ instances.

**Accuracy and complexity** The parameters $r$, $u$ ant $t$ can be altered to achieve a viable memory-accuracy trade-off. The algorithm uses $O(t \cdot r \cdot \log(F_0))$ memory words

[2]We note that the expected number of false positives does not depend on the number of distinct $k$-mers $F_0$. With increasing $F_0$ more collisions take place, but also more collisions are detected and these two effects cancel each other out. The proof can be found in Lemma 4 of Appendix D of [12].

by $t$ instances of sketches with $W = \log F_0$ levels with $r$ counters each. An update of $t$ sketches (processing of one $k$-mer) requires $O(t)$ time.

The authors have shown that the algorithm computes estimates $\hat{F}_0$ and $\hat{f}_i$ for sufficiently large $f_i$ ($f_i \geq F_0/\lambda$) with a bounded relative error $(1-\varepsilon)F_0 \leq \hat{F}_0 \leq (1+\varepsilon)F_0$, $(1-\varepsilon)f_i \leq \hat{f}_i \leq (1+\varepsilon)f_i$ with probability at least $1-\delta$, when the parameters are set as follows: $t = O(\log(\lambda/\delta))$, $r = O(\frac{\lambda}{\varepsilon^2})$ and $u = O(\frac{\lambda F_0}{\varepsilon^2})$.

Due to the loose constants in the asymptotic estimate, these values of $t, r, u$ cannot be directly used in practice to guarantee the error bounds. The accuracy of this algorithm was tested experimentally with arbitrary parameters $t = 7, r = 2^{16}, 2^{18}, u = 2^{13}$.

## 2.2 Empirical Study of Approximation Errors

To study the character of approximation errors, we first made several experimental observations on generated data.

### 2.2.1 Data Generation

Since we will later use the approximated histogram to estimate a genome size, we will mainly study Kmerlight's performance on data that are to some degree biologically plausible. We will base our data generation on a sequencing process as it was described in section 1.1.1.

As the Kmerlight's input data consist of genome reads, we first generate a genome $g_1 g_2 g_3 \ldots g_L$: a random sequence of length $L$ consisting of characters $A, C, G, T$ each with probability $1/4$ at every position.

Afterwards we generate reads, each of length $l = 100$. Instead of creating an explicit number of reads, we choose a parameter $c$ (coverage) and generate $c \cdot L/l$ independent reads.

To generate a single read we uniformly select a random starting position $s$ in genome from a range $1, \ldots, L - l + 1$ and then the read consists of characters $g_s g_{s+1} \ldots g_{s+l-1}$. Finally, to simulate sequencing errors, we change each read character with probability $e/3$ ($e$ denotes error rate) into a one of the three other characters.

### 2.2.2 Error Characteristics

In this section, we compare the exact $k$-mer abundance histogram produced by Jellyfish software [7] with approximated histograms computed by Kmerlight (with parameters $t = 7, r = 2^{15}, u = 2^{13}$ using 60MB of RAM). We ran Kmerlight in 50 trials on the same input data, and we investigate means and standard deviations of the estimates

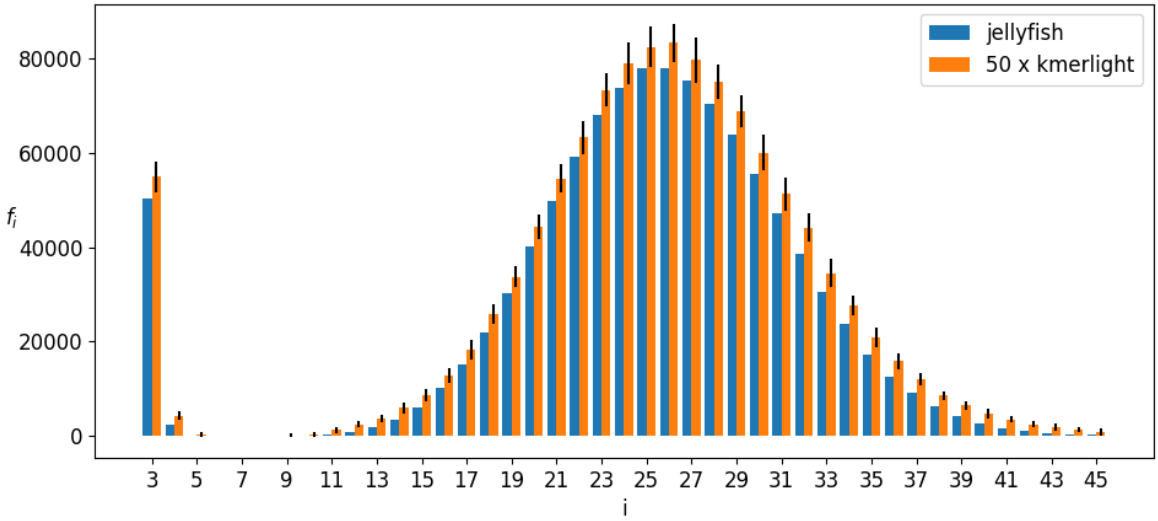


Figure 2.1: The exact values $f_i$ produced by Jellyfish software (blue) and approximated values $\hat{f}_i$ produced by Kmerlight averaged from 50 trials (orange). The errorbars express the standard deviation of each estimate. Note that columns for $i = 1, 2$ were trimmed, having values approximately $10^7, 10^6$ respectively.

$\hat{f}_i$. We demonstrate three error characteristics on a genome generated with parameters $L = 10^6, c = 50, e = 0.02$. In all our experiments we use value $k = 21$.

**Overestimation** In Figure 2.1 it is clearly visible that Kmerlight systematically overestimates values of $f_i$. Mean errors for abundances around 25 reach absolute values of 4000, which is 5% relative error. Lower accuracy is expected, since these values $f_i$ are considerably smaller than the number of all $k$-mers $F_0$, and Kmerlight guarantees high accuracy only for high values of $f_i$. But the clear bias of the estimate towards higher values ($E(\hat{f}_i) > f_i$) is unexpected and unexplained yet.

In the following sections we clarify the source of this bias (2.3), and we present means to make the estimator $\hat{f}_i$ unbiased (2.4).

**Higher relative errors and higher relative variance with lower $f_i$** Another characteristic of the errors is a trend of increasing relative variance and relative error of the estimates for lower values $f_i$[3]. In Figure 2.2 we visualize mean and standard deviation of relative errors $(\hat{f}_i - f_i)/f_i$ of $f_i$ estimates. Kmerlight guarantees bounded errors only for the most frequent $k$-mers, those with high $f_i/F_0$ ratio, but a theoretical quantitative analysis of the error distribution was not presented in the previous work [12, 9].

[3]Note that the absolute mean error $(\hat{f}_i - f_i)$ and absolute variance decrease with decreasing $f_i$, as it can be seen in Figure 2.1. However, if the errors are considered proportionally to $f_i$, as $(\hat{f}_i - f_i)/f_i$, these relative errors and their variance increase with decreasing $f_i$.

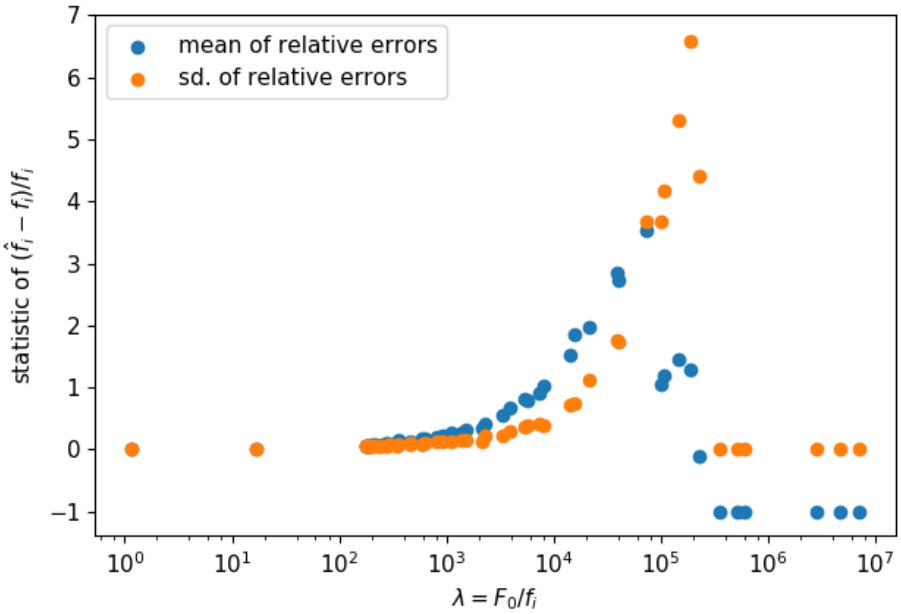


Figure 2.2: The mean and standard deviation of relative errors $(\hat{f}_i - f_i)/f_i$ for columns sorted by decreasing $f_i$ (increasing $F_0/f_i$). The drop in both statistics at values $\lambda = 10^5$ is caused by Kmerlight's insensitivity to infrequent $k$-mers.

We provide a quantitative estimate of the variance in section 2.5 and then we use this knowledge to set Kmerlight's parameters to achieve a desired error in 2.6.

**Insensitivity to lowest $f_i$** Since the hash tables of Kmerlight's sketch are smaller than the number of all $k$-mers, many $k$-mers collide. When $f_i$, the number of distinct $k$-mers with abundance $i$, reaches a specific boundary (in Figure 2.3 it seems to be $5 \times 10^2$), the probability that at least one $k$-mer with abundance $i$ becomes stored in a collision-free counter at any level approaches to zero.

As a result, the estimates $\hat{f}_i$ for the lowest $f_i$ may be based on none or very few counters with value $i$. If no $k$-mer survives the collisions ($t_i = 0$) then $\hat{f}_i = 0$. If very few $k$-mers survive the collisions, the estimator multiplies the number collision-free counters by factor $2^{w^*}$, and the estimate may overestimate $f_i$ drastically if a higher level $w^*$ was selected.

This effect is expected, but we point it out as it may limit the usage of the approximated histogram to some applications.

### 2.2.3 Explanation of the Histogram Shape

Kmerlight guarantees precise estimates of those $f_i$ that have values close to $F_0$. Unfortunately, in the sequencing data, sequencing errors often create many unique $k$-mers, and thus we have $f_1$ close to $F_0$ and the remaining $f_i$ (for $i > 1$) are much smaller

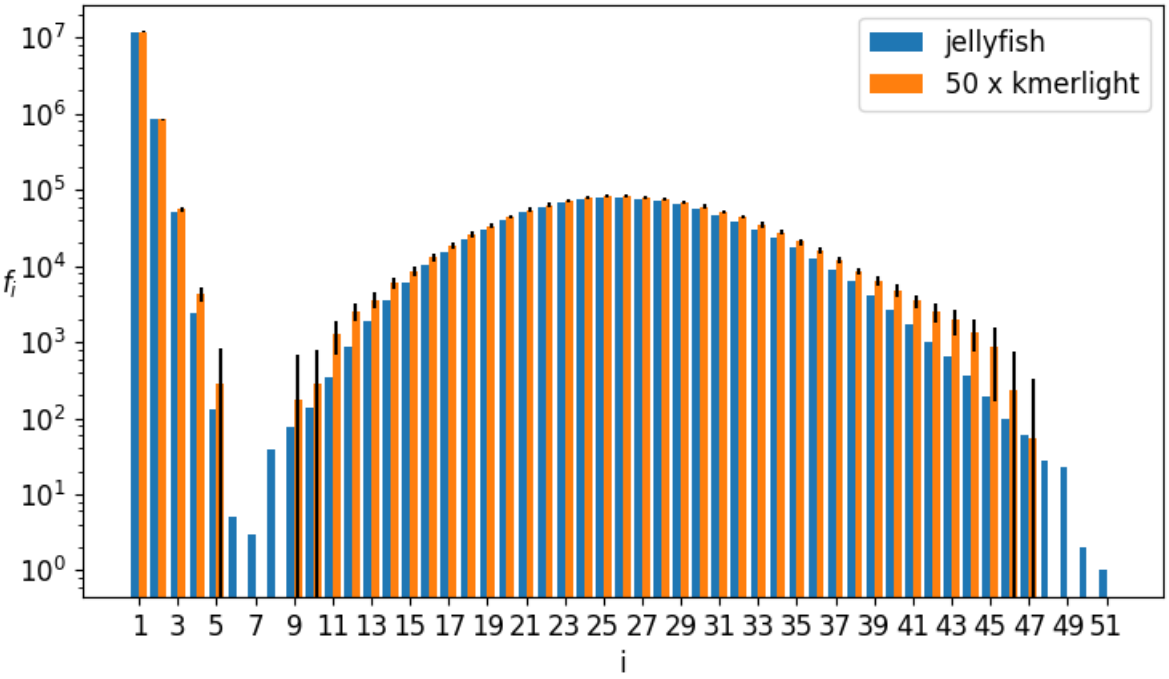


Figure 2.3: The exact (blue) and average approximated (orange) histograms, identical to those in Figure 2.1, but displayed in a logarithmic scale with values $i = 1, 2$ included. Note the lack of orange columns where blue columns reach lower values – due to hashing collisions, Kmerlight cannot estimate $f_i$ lower than $5 \times 10^2$.

than $F_0$.

For example, with sequencing error rate of 2%, 35% occurrences of 21-mers are erroneous and thus unique with high probability. As a result, 35% of all input $k$-mers are concentrated in $f_1$, and 65% of $k$-mers are distributed to other $f_i$.

Furthermore, to increase $f_1$ by one, only one $k$-mer occurrence is needed, but to increase $f_{10}$ by one, ten error-free occurrences of the same $k$-mer must be present in the input stream. As a result, 65% of error-free $k$-mer occurrences constitute a much smaller fraction of all distinct $k$-mers[4].

These effects explain, why the value of $f_1$ is approximately a hundred times larger than the values of $f_i$ for $i > 10$ in the presented experimental data in figures 2.1 and 2.3.

## 2.3 The Source of Approximation Bias

To calculate an estimate of $f_i$, the level $w_i^*$ is chosen by Kmerlight to maximize $t_i^{(w)}$ – the number of collision-free counters with value $i$ at level $w$.

[4]If we take a hundred $k$-mer occurrences, 35 of them are expected to be erroneous, and so they produce 35 distinct unique $k$-mers. If the remaining 65 $k$-mers contribute to abundance 10, we are left with approximately 41 distinct $k$-mers from which 35 are unique.

$$t_i^{(w)} = |\{T_w[c].v = i \mid c \in \{0, \ldots, r-1\}\}| \qquad w_i^* = \arg\max_w t_i^{(w)}$$

Then the number of counters at this level is converted into the estimate of $f_i$ by multiplying it by $2^{w_i^*} \cdot \left(1 - \frac{1}{r}\right)^{1-F_0/2^{w_i^*}}$.

The reason for selecting level $w_i^*$ by this criterion is not clearly stated in Kmerlight's original paper [12]. We believe that the authors hoped to minimize the variance of the estimator $\hat{f}_i$ by including as many $k$-mers in the estimate as possible.

### 2.3.1 Analytical $w^+$

In order to get insight into which levels $w_i^*$ are selected by Kmerlight, we will analytically find the levels $w_i^+$ that maximize $E(t_i^{(w)})$ – the expected number of collision-free counters with value $i$.

As it was derived in (2.2), $E(t_i^{(w)}) = f_i/2^w \cdot \left(1 - \frac{1}{r}\right)^{F_0/2^w - 1}$. The value of $w_i^+$ at which $E(t_i^{(w)})$ is maximized can be obtained by differentiation ($\frac{\mathrm{d}}{\mathrm{d}w} E(t_i^{(w)}) = 0$) or by using the discrete inequalities $E(t_i^{(w_i^+-1)}) \leq E(t_i^{(w_i^+)}) \leq E(t_i^{(w_i^++1)})$. By manipulating the inequalities we get

$$\frac{1}{4} \leq \left(1 - \frac{1}{r}\right)^{F_0/2^{w_i^+}} \leq \frac{1}{2} \tag{2.4}$$

and finally we can calculate $w_i^+$:

$$\lg F_0 + \lg\lg \frac{r}{r-1} - 1 \leq w_i^+ \leq \lg F_0 + \lg\lg \frac{r}{r-1}$$

Symbol lg denotes the binary logarithm.

As $w_i^+$ denotes a Kmerlight's level, it should always be a positive integer. In rare cases there can be two solutions $w_i^+ = \lfloor \lg F_0 + \lg\lg \frac{r}{r-1} \rfloor, w_i^+ = \lceil \lg F_0 + \lg\lg \frac{r}{r-1} \rceil$. For some values of $F_0$ the optimal value of $w_i^+$ may be less than one, but in these cases we choose $w_i^+ = 1$. In practice, $w_i^+$ can be simply computed by a for-loop through $w = 1, \ldots, W$, choosing $w$ that maximizes $E(t_i^{(w)})$.

Note that the choice of the optimal level $w_i^+$ does not depend on values of $i$ or $f_i$, but only on values of $F_0$ and $r$. Since $w_1^+ = w_2^+ = \cdots = w_m^+$, from now on we will refer to this level simply by $w^+$. As the level $w^+$ maximizes $E(t_i^{(w)})$ for any non-zero value $i$, this level also maximizes the expected number of all non-empty collision-free counters $E(t^{(w)}) = \sum_{i=1}^m E(t_i^{(w)}) = F_0/2^w \cdot \left(1 - \frac{1}{r}\right)^{F_0/2^w - 1}$.

### 2.3.2 Explanation of Bias

Since the ratio $f_i/F_0$ is typically very low for higher values of $i$, and this ratio determines the value of $t_i^{(w)}$ (the number of collision-free counters holding the value $i$), each $f_i$ is

represented by only a relatively small number of collision-free counters at each level of Kmerlight's sketch. Therefore $t_i^{(w)}$ have high relative variances.

By examining the levels $w^+ - 1$ and $w^+ + 1$ we learn that the number of collision-free counters at these levels is similar to the number of collision-free counters at level $w^+$. From (2.2) we can see that there are two times more $k$-mers hashed into the level $w^+ - 1$ than $k$-mers hashed into $w^+$, but also more collisions happen at $w^+ - 1$. These two effects partially cancel each other out and maintain similar values of $E(t^{(w)})$ for $w = w^+ - 1, w^+, w^+ + 1$.

As we substitute bounds from (2.4) into (2.2), we learn that

$$\frac{F_0}{2^{w^+}} \cdot \frac{1}{4} \cdot \left(1 - \frac{1}{r}\right)^{-1} \leq E(t^{(w^+)}) \leq \frac{F_0}{2^{w^+}} \cdot \frac{1}{2} \cdot \left(1 - \frac{1}{r}\right)^{-1}$$

and if we do the same for the level $w^+ - 1$ we get:

$$2 \cdot \frac{F_0}{2^{w^+}} \left(\frac{1}{4}\right)^2 \left(1 - \frac{1}{r}\right)^{-1} \leq E(t^{(w^+-1)}) \leq 2 \cdot \frac{F_0}{2^{w^+}} \left(\frac{1}{2}\right)^2 \left(1 - \frac{1}{r}\right)^{-1}$$

Finally, for the level $w^+ + 1$ we get:

$$\frac{1}{2} \cdot \frac{F_0}{2^{w^+}} \left(\frac{1}{4}\right)^{1/2} \left(1 - \frac{1}{r}\right)^{-1} \leq E(t^{(w^++1)}) \leq \frac{1}{2} \cdot \frac{F_0}{2^{w^+}} \left(\frac{1}{2}\right)^{1/2} \left(1 - \frac{1}{r}\right)^{-1}$$

If we substitute $c = F_0/2^{w^+} \cdot (1 - 1/r)^{-1}$, we can summarize the similarity of levels as follows:

$$E(t^{(w^+)}) \in \left\langle \frac{1}{4}c, \frac{1}{2}c \right\rangle \quad E(t^{(w^+-1)}) \in \left\langle \frac{1}{8}c, \frac{1}{2}c \right\rangle \quad E(t^{(w^++1)}) \in \left\langle \frac{1}{4}c, \frac{1}{2\sqrt{2}}c \right\rangle \quad (2.5)$$

As a result of low values and high variances of $t_i^{(w)}$ and similar values of $E(t^{(w^+-1)})$, $E(t^{(w^+)})$ and $E(t^{(w^++1)})$, any of these levels could hold the maximal $t_i^{(w)}$ and become chosen by Kmerlight as its $w_i^*$.

Kmerlight always chooses the level which maximizes $t_i^{(w)}$ and this freedom of choice leads to values of $t_i^{(w_i^*)}$ consistently higher than values of $E(t_i^{(w_i^*)})$ which consequently biases the estimator $\hat{f}_i$ towards higher values.

We extracted the values $w_i^*$ for different $f_i$ from one Kmerlight run and we present them in Figure 2.4. For each $i$ there are up to seven values of $w_i^*$, each selected by one instance of seven Kmerlight's sketches[5]. The most of selected levels follow the analytical $w^+ = 9$; however, Kmerlight often chooses level 8 or 10 to maximize $t_i^{(w)}$. Note that for $3 \leq i \leq 15$ and $i \geq 40$, the values of $f_i$ are low, and thus $t_i^{(w)}$ have high relative variance (as it was presented in section 2.2.2). The maximal $t_i^{(w)}$ can thus also be reached at levels more distant from the analytical $w^+ = 9$.

[5]If there are no counters holding the value $i$ in the whole sketch, Kmerlight's instance does not choose any level and estimates $f_i$ as zero.

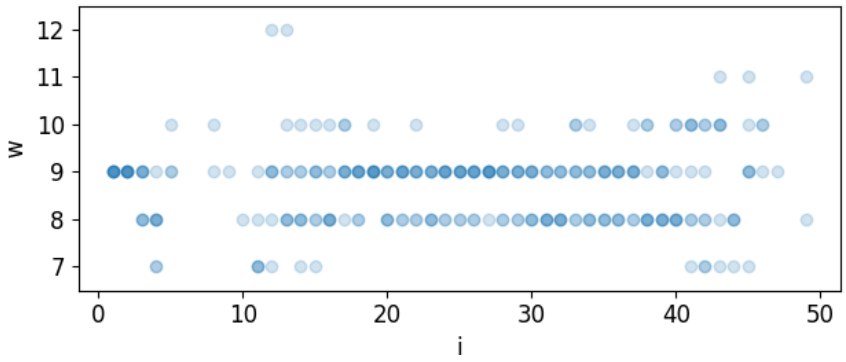


Figure 2.4: Each column shows at most seven levels $w_i^*$ that were selected by one of seven instances of Kmerlight. If a level was chosen by more instances for a specific $i$, its circle is darker. Kmerlight was run with the same parameters and on the same set of reads as in experiment in section 2.2.2.

### 2.3.3 The Content of Counters

To provide an insight into the relations presented in (2.5), let us now briefly investigate, how the numbers of empty, collision-free and colliding counters vary across levels.

As we focus on a single level $w$, let us denote as $p_{cf}$ the probability that a single counter is non-empty and collision-free. In other words, $p_{cf}$ is the probability that exactly one of $F_0/2^w$ $k$-mers was hashed into that counter.

The number of distinct $k$-mers hashed into one counter ($X$) follows a binomial distribution, $X \sim \mathrm{Bin}(F_0/2^w, 1/r)$, so $p_{cf}$ can be expressed as follows:

$$p_{cf} = P[X = 1] = \frac{F_0}{2^w} \cdot \frac{1}{r} \left(1 - \frac{1}{r}\right)^{F_0/2^w - 1} \tag{2.6}$$

Similarly, we can derive the probability $p_e$ that a counter is empty and the probability $p_c$ that a counter holds a collision:

$$p_e = P[X = 0] = \left(1 - \frac{1}{r}\right)^{F_0/2^w} \qquad p_c = P[X > 1] = 1 - p_e - p_{cf}$$

Note that the expected number of non-empty collision-free counters at one level is $E(t^{(w)}) = r \cdot p_{cf}$.

In order to further demonstrate the similarity of $E(t^{(w)})$ at the levels close to $w^+$ we display the values of $p_{cf} = E(t^{(w)})/r, p_e, p_c$ for different levels in Figure 2.5. These probabilities represent the fractions of non-empty collision-free, empty and collided counters respectively.

The presented settings show one of the worst possible situations. Since $E(t^{(8)}) \approx E(t^{(9)})$, Kmerlight very frequently chooses the level $w_i^*$ from $\{8, 9\}$ (as it can be seen at Figure 2.4), but it always prefers the one with the highest value of $t^{(w)}$. Thus $E(t_i^{(w_i^*)}) > E(t^{(8)}) \approx E(t^{(9)})$, which results in overestimation.

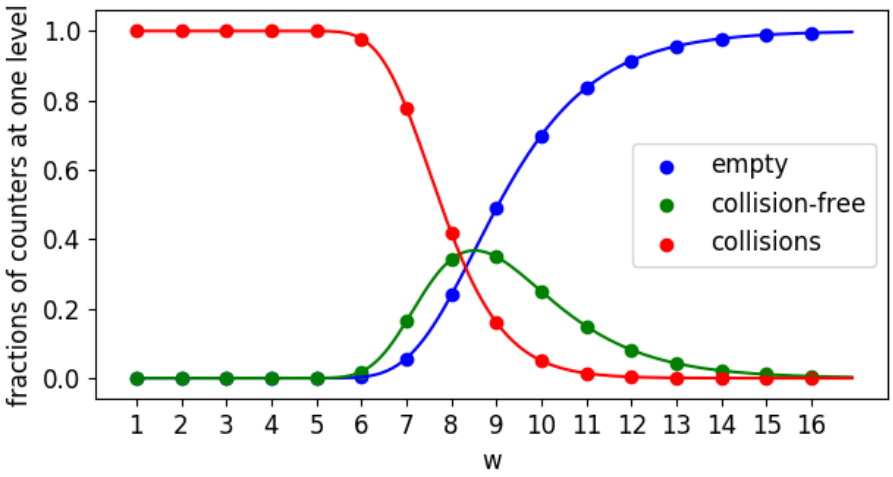


Figure 2.5: The values of $p_e, p_{cf}, p_c$ represent the theoretical fractions of empty, collision-free and collided counters in each level respectively. To make these theoretical values comparable to the experimental results from section 2.2.2, we set $F_0 = 1.2 \times 10^7, r = 2^{15}$.

If there was a greater difference between $E(t^{(w^+)})$, $E(t^{(w^+-1)})$ and $E(t^{(w^++1)})$, Kmerlight would choose the level $w^+$ much more often than the other levels and so it would have a smaller chance of choosing $t_i^{(w_i^*)} > E(t^{(w^+)})$.

## 2.4 Unbiased $f_i$ Estimation

Based on our observations in the previous section we will alter Kmerlight to produce unbiased estimates of $f_i$ (under the assumption that there are no undetected collisions).

To achieve an unbiased estimate of $f_i$ we will use the level that maximizes the expected number number of collision-free counters, instead of the level that maximizes the number of collision-free counters. We will base our estimates of $f_i$ on the level $w^+$, instead of $w_i^*$. Our modified Kmerlight thus loses the ability to choose the maximal $t^{(w)}$, and the estimates will be now based on values of $t^{(w^+)}$ which are expected to vary evenly around their mean $E(t^{(w^+)})$, reaching both higher and lower values than $E(t^{(w^+)})$.

Note that the value of $\hat{f}_i$ will be calculated from the same level for every abundance $i$. Level $w^+$ will be calculated using only the estimated number of all distinct $k$-mers ($F_0$) and the number of counters at each level ($r$).

Modified Kmerlight first processes all the $k$-mers in the same way as the original Kmerlight. Then $F_0$ is estimated from all Kmerlight's instances[6] and the value $w^+$

[6] The value of $F_0$ is estimated in the original way as presented in (2.1). Note that the estimate depends on the value of $t_0^{(w)}$, $E(t_0^{(w)}) = r \cdot p_e$. Number of empty counters reach high values ($t_0^{(w^*)} \approx r/2$)

is calculated to maximize the expected number of collision-free counters as it was described in section 2.3.1. Finally, values of $f_i$ are estimated from the observed counts of collision-free counters at level $w^+$ by (2.3).

In figures 2.6 and 2.7, we present a comparison of the original Kmerlight and the modified Kmerlight. We ran both versions of Kmerlight on the same data as described in 2.2.2 in 300 trials. Our new algorithm removes bias and maintains the same accuracy (variance) of the estimates.

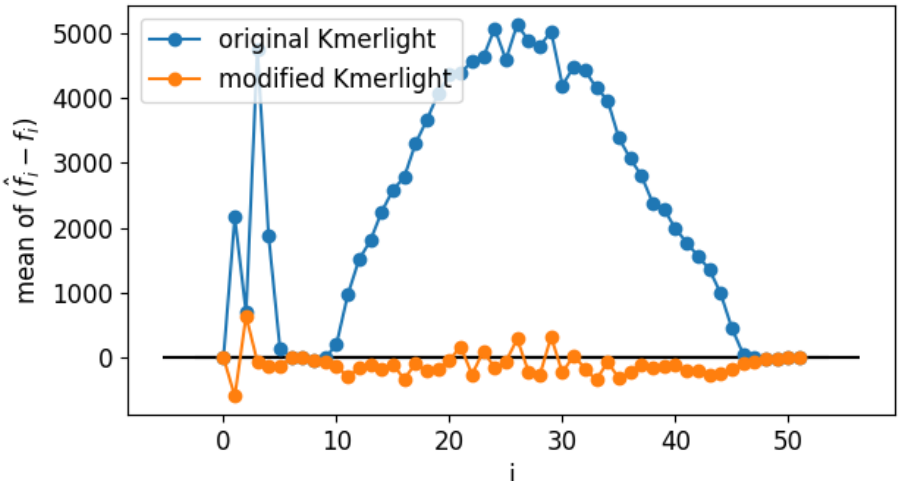


Figure 2.6: Mean errors of estimates $(\hat{f}_i - f_i)$ produced by original (blue) and modified (orange) Kmerlight in 300 trials. While the original Kmerlight overestimates $f_i$ significantly in an average run, modified Kmerlight achieves $E(\hat{f}_i) \approx f_i$.

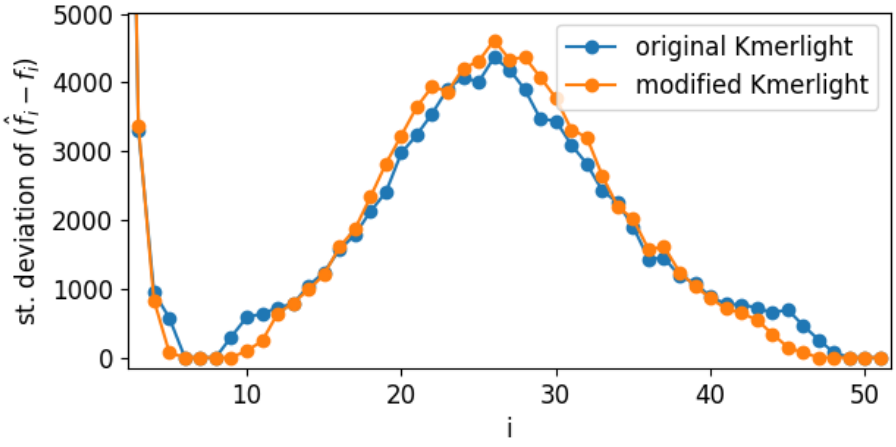


Figure 2.7: Standard deviation of estimates $(\hat{f}_i - f_i)$ produced by original (blue) and modified (orange) Kmerlight in 300 trials. Our modification of Kmerlight does not change the variance of estimates.

thus $t_0$ has low relative variance and so the effects that biased $\hat{f}_i$ do not occur here or they cause a smaller scale differences. We do no challenge the unbiasedness of the estimator $\hat{F}_0$ in our work.

**Decreased memory consumption** Note that we used only one of 64 Kmerlight's levels to compute estimates for all $f_i$. We cannot decrease the overall memory consumption by a factor of 64, however. To compute $\hat{F}_0$, more levels are needed, but still, we can compute the estimate of $F_0$ separately from the estimates of $f_i$.

If we roughly guess $F_0$ from the size of sequencing data, we might be able to estimate $F_0$ with a lower number of levels. It might be also sufficient to estimate $F_0$ with smaller arrays of counters at each level (smaller parameter $r$). Also a completely different method might be used for $F_0$ computation.

We did not inquire deeper into the topic of decreasing the memory consumption, but we believe that at least a tenfold improvement might be achieved in this direction.

## 2.5 Evaluation of Approximation Variance

In this section we will derive a rough quantitative estimate of variance of $\hat{f}_i$ and then support its credibility by experimental results.

### 2.5.1 Derivation of $Var(\hat{f}_i)$

Using the observation that values $w_i^*$ chosen by Kmerlight often correspond to the analytical $w^+$ and that the value of analytical $w^+$ is a constant for all $i = 1, 2, \ldots, m$, we will approximate Kmerlight's variance by a variance of $t_i^{(w^+)}$. This variance approximation will be even more accurate for our modified Kmerlight. In this section we will denote $w^+$ simply as $w$, since we will not use any specific property of $w^+$.

The number of collision-free $k$-mers is equal to the number of collision-free counters as each collision-free counter stores a single $k$-mer. Thus we will use the symbol $t_i^{(w)}$ to denote both quantities and from (2.2) it holds that $E(t_i^{(w)}) = f_i/2^w \cdot (1 - 1/r)^{F_0/2^w - 1}$.

**Sampling View** In order to approximate the variance of estimator $\hat{f}_i$, we will consider $t_i^{(w)}$ to follow a binomial distribution $Bin(f_i, p_s)$, where $p_s = 1/2^w \cdot (1-1/r)^{F_0/2^w-1}$. This simplification corresponds to a simple sampling process in which we sample each of $f_i$ $k$-mers with probability $p_s$ (sampling probability), and we discard each $k$-mer with probability $1 - p_s$. Note that this approach ignores the fact that the events of $k$-mers remaining collision-free are not independent.

Since a random variable following $Bin(n, p)$ has variance of $np(1-p)$, $Var(t_i^{(w)}) = f_i \cdot p_s \cdot (1 - p_s)$. The estimate of $f_i$ is obtained as $t_i/p_s$, so

$$Var(\hat{f}_i) = Var\left(\frac{t_i}{p_s}\right) = \frac{1}{p_s^2} \cdot Var(t_i) = \frac{f_i \cdot (1 - p_s)}{p_s} \tag{2.7}$$

**Effect of Medians** Kmerlight chooses the estimate $\hat{f}_i$ as a median of estimates of $t$ independent sketches: $\hat{f}_i = \text{med}(\hat{f}_i^{(1)}, \dots \hat{f}_i^{(t)})$.

To incorporate the effect of medians, we will use the claim that for a continuous random variable with a density function $f(x)$ and mean $\bar{x}$ its sample median from a sample of size $n$ is asymptotically[7] normal [13]:

$$\text{med}(x) \dot{\sim} N\left(\bar{x}, \frac{1}{4nf(\bar{x})^2}\right) \tag{2.8}$$

As the binomial distribution of $t_i^{(w)}$ is a discrete distribution and does not have a density function, we will approximate $Bin(f_i, p_s)$ with $N(\mu = f_i p_s, \sigma^2 = f_i p_s(1-p_s))$. The density function of normal distribution in its mean $\mu$ is $\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(\mu-\mu)^2}{2\sigma^2}}$ which is $\frac{1}{\sqrt{2\pi\sigma^2}}$.

Using the two previous approximations (2.7), (2.8), variance of $\hat{f}_i$ selected as a median of $t$ instances can be derived as follows:

$$Var(\hat{f}_i) \approx \left(4t\frac{1}{2\pi Var(\hat{f}_i^{(l)})}\right)^{-1} = \frac{2\pi}{4t} Var(\hat{f}_i^{(l)}) = \frac{\pi}{2t}\frac{f_i \cdot (1-p_s)}{p_s} \tag{2.9}$$

### 2.5.2 Comparison With Experiments

In order to validate our estimate of variance, we compare our theoretical model of the distribution of $\hat{f}_i$ with the experimentally obtained distribution.

We ran our modified Kmerlight with $t = 7$ instances in 300 trials on the data presented in section 2.2.2. In Figure 2.8 we present histograms of values of $\hat{f}_i$ for multiple values of $i$ selected to represent a range of $f_i$ values[8]. In Figure 2.9 we present empirical cumulative distribution functions of the same data.

We compare these histograms with two normal distributions. The best normal fit is a Gaussian with its mean and standard deviation obtained directly from the observed values of $\hat{f}_i$. The plotted theoretical prediction uses the exact value of $f_i$ as its mean $\mu$ and the variance $\sigma^2$ is calculated according to (2.9) with the use of exact values of $f_i$ and $F_0$.

Note first that for this dataset, $w^+ = 9$ and thus the sampling probability $p_s$ (introduced in the previous section 2.5) is approximately $1/2^9 \cdot \frac{1}{2} \approx 1/1000$ when we use the upper bound from (2.4).

[7] Rider [10] has shown that even for a sample of only 7 observations, the relative error of this approximation is only circa 6% for a normal random variable.

[8] The values $i = 6, 10, 13, 16, 32, 23, 2, 1$ were selected as follows. We sorted the values $f_i$ in increasing order, and then we selected each eighth value. Since the values of $f_1, f_2$ differ from other $f_i$ by orders of magnitude, we decided to also include $i = 1, 2$.

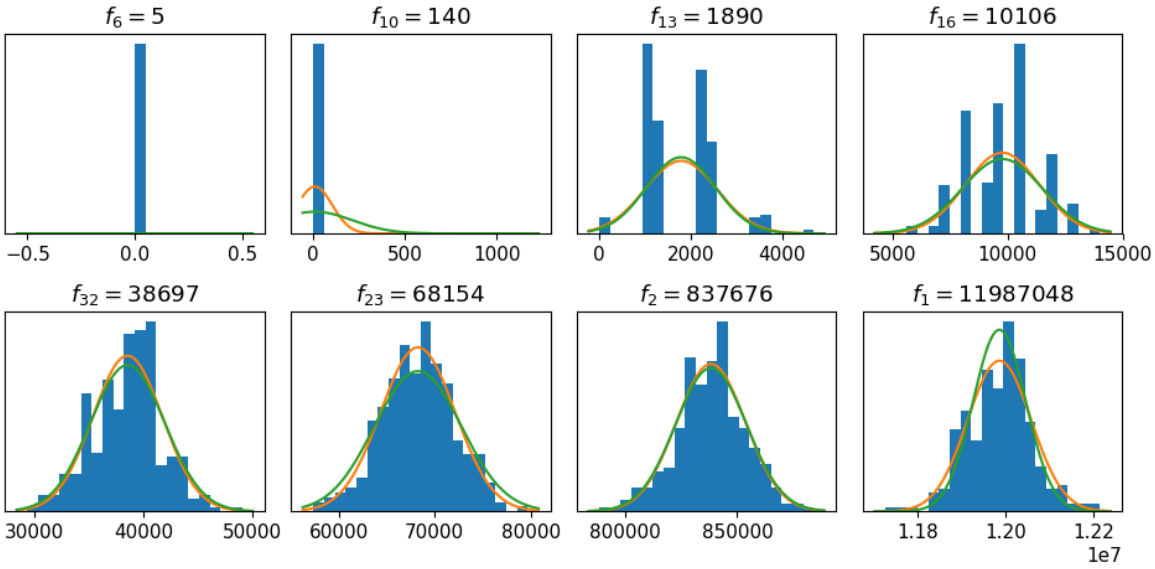


Figure 2.8: Empirical probability densities (blue histograms), the best normal fits (orange lines) and theoretical estimates (green) of distributions of $\hat{f}_i$ for multiple abundances. Histograms come from 300 runs of modified Kmerlight. Horizontal axes display the values of estimates $\hat{f}_i$.

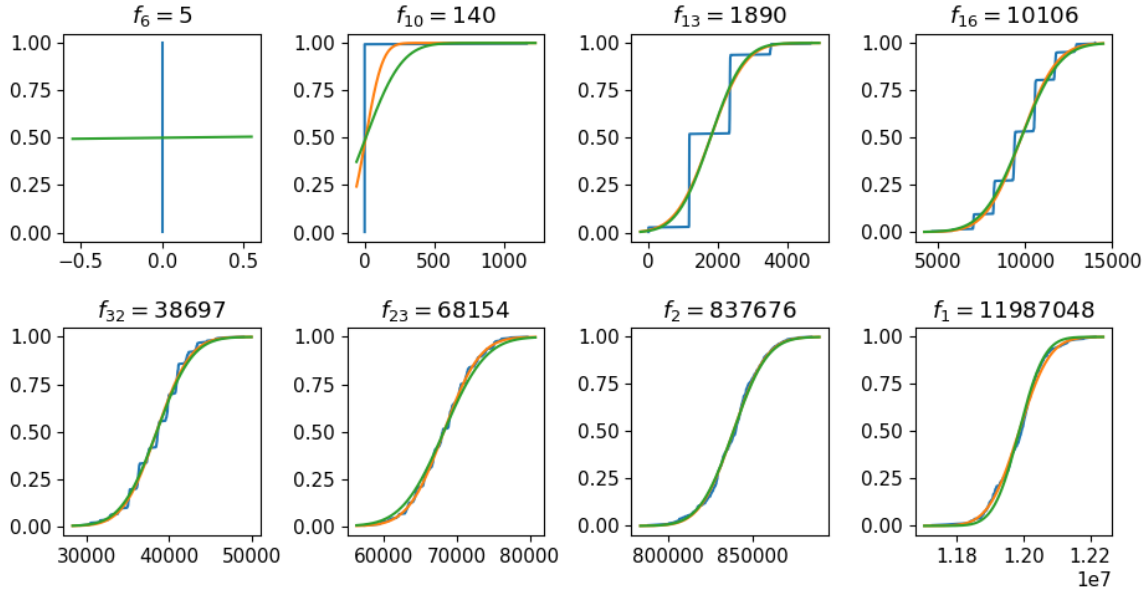


Figure 2.9: Empirical cumulative distribution functions (blue), the best normal fits (orange), theoretical estimates (green).

**The lowest $f_i$ values, $E(t_i) = f_i \cdot p_s < 1$** For the lowest values of $f_i$ (i.e. $f_6, f_{10}$), no $k$-mers survive the collisions at level $w^+$ and thus $t_i = 0$ and $\hat{f}_i = 0$. As $f_{10} = 140$ is only approximately ten times lower than $p_s$, we might expect at least one $k$-mer to survive collisions in some of 300 trials. But since the final estimate is selected as a median of seven instances, at least one $k$-mer must survive in at least four instances in

order to produce $t_i = 1$. Thus $\hat{f}_i = 0$.

Since the estimates $\hat{f}_i$ are always zero, our estimates of variances for these $f_i$ are very imprecise.

**Low $f_i$ values, $E(t_i) = f_i \cdot p_s \approx 1$** For low values of $f_i$ (i.e. $f_{13}, f_{16}$), a very small number of $k$-mers hash into collision-free counters at level $w_i^*$ or $w^+$. Therefore the estimator $\hat{f}_i$ can reach only a limited set of discrete values ($\hat{f}_i = 0$ if no $k$-mer survives collisions, $\hat{f}_i = 1/p_s \approx 1000$ if one $k$-mer is in collision-free counter, $\hat{f}_i = 2/p_s \approx 2000$ if two $k$-mers survive, ...), as it can be seen in Figure 2.8.

The approximation with normal distribution is not precise for these values $f_i$, since the distribution of $\hat{f}_i$ is clearly discrete, but Figure 2.9 shows that our approximation can be used to estimate the empirical distribution functions (and thus approximate $p$-values).

**High $f_i$ values, $E(t_i) = f_i \cdot p_s \gg 1$** For higher $f_i$, the estimator $\hat{f}_i$ takes on various values, and the approximation with normal distribution seems reasonable, as it can be seen from the bottom rows of figures 2.8 and 2.9.

We have also applied Kolmogorov-Smirnov tests to test the normality of $\hat{f}_i$. Discrete values of $\hat{f}_i$ create steps in the cumulative distribution functions, and therefore these tests reject normality hypotheses for columns with few $k$-mers. KS tests reject normality for $f_i < 20\,000$ with our dataset of 300 trials and the significance level of 5%.

From previous observations we conclude that the distribution of $\hat{f}_i$ can be approximated by Gaussian with variance calculated by (2.9), although we are also aware of the limitations of this approximation.

**Notes on experimental design** Our theory does not model the mean value of Kmerlight's estimates. We assume that the estimates $\hat{f}_i$ are unbiased, and therefore we simply use $\mu = f_i$ and we focus on modified Kmerlight in the presented experiment. If we used one of the observed estimates $\hat{f}_i$ as the parameter $\mu$, the fluctuation of $\hat{f}_i$ would shift the plotted theoretical curves to the side and thus it would make the comparison less clear.

In practice, estimates of $f_i$ and $F_0$ will be used to calculate the variance. We also might have used estimates $\hat{f}_i, \hat{F}_0$ from a single trial to calculate the variance estimate, but by using the average $\hat{f}_i$ ($f_i$) we investigate the average estimate of variance.

We also used the original Kmerlight to produce the values $\hat{f}_i$ in another experiment. Since we do not model the bias of Kmerlight's estimates, to make the histogram comparable to our theory, we had to use the sample mean of $\hat{f}_i$ as the parameter $\mu$.

With this adjustment we were able to observe the same phenomena as those presented in figures 2.8, 2.9 and described above.

Finally we present the comparison of Kmerlight's variance and its theoretical prediction in Figure 2.10. Although our theoretical approximation is based on the analytical value of level $w^+$, it can also be used to predict the variance of the original Kmerlight algorithm.

From the experiment we know that we can estimate standard deviation of Kmerlight's estimates with error less than 5% even for lower $f_i \approx 1000 \approx 1/p_s$. The experiment further suggests that an accurate estimate of variance for the lowest values of $f_i < 100$ would be zero.

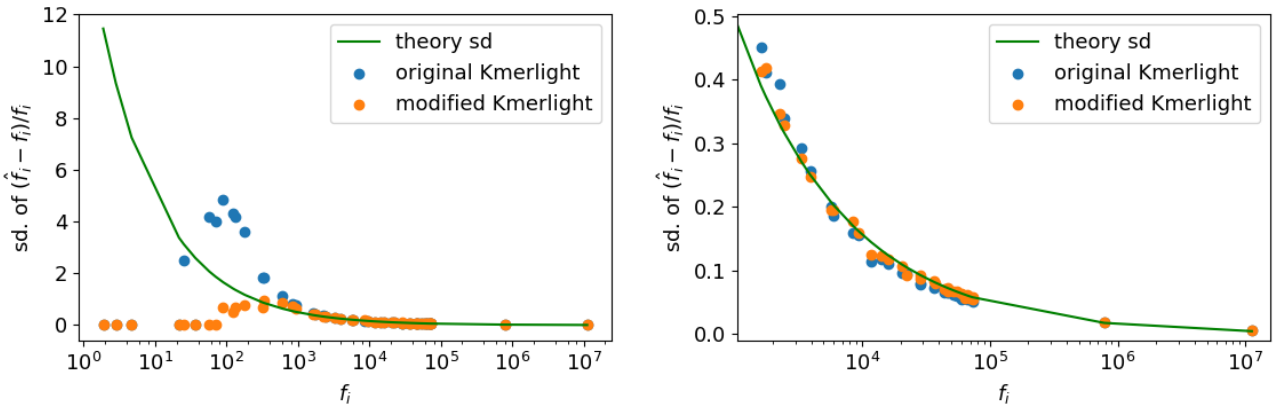


Figure 2.10: The blue and orange points show standard deviations of relative errors $(\hat{f}_i - f_i)/f_i$ of estimates $\hat{f}_i$ computed by original and modified Kmerlight respectively in 300 runs on dataset from 2.2.2. The green line represents the approximation of standard deviation calculated using (2.9). If $\sigma_i^2$ is the variance of $\hat{f}_i$, the standard deviation of relative errors can be calculated as $\sqrt{\sigma_i^2/f_i^2}$. The figure on the right is only a magnified view of the the same plot for $f_i > 10^3$.

There is one more interesting thing that can be noticed in Figure 2.10: the difference of variances between original and modified Kmerlight's estimates for $f_i \in (100, 1000)$ . The original Kmerlight searches for a level $w$ that maximizes $t_i^{(w)}$ so even if only one $k$-mer with abundance $i$ survives the collisions at any level of the sketch, Kmerlight will use it for estimation of $\hat{f}_i$. We did not include this effect into the variance estimation, hence the estimates for these $f_i$ are not precise for the original Kmerlight.

## 2.6 Choice of Kmerlight's Parameters

As we have noted in section 2.1, the authors of the original paper about Kmerlight [12] have provided us theoretical bounds of Kmerlight's errors in the following form:

Kmerlight computes estimates $\hat{f}_i$ (and $\hat{F}_0$) for sufficiently large $f_i$ ($f_i \geq F_0/\lambda$) with an error $(1-\varepsilon)f_i \leq \hat{f}_i \leq (1+\varepsilon)f_i$ with probability at least $1-\delta$.

It is unclear, however, how to set the parameters $r$ (the number of counters at one level) and $t$ (the number of sketch instances from which a median will be selected).

In this section we use the approximate distribution of Kmerlight's errors (derived in the previous section) to set the parameters in order to achieve desired error bounded by $\varepsilon$ with probability $1-\delta$ (under the assumption that $\hat{f}_i$ is distributed according to our approximation; Kmerlight's authors do not make such assumptions).

**Prediction/confidence interval** To obtain the error bounds in the aforementioned form, we derive a two-sided prediction interval of $\hat{f}_i$.

By standardization we get $(\hat{f}_i - f_i)/\sigma \dot{\sim} N(0,1)$ and so it holds that

$$P\left[-u\left(\frac{\alpha}{2}\right) \leq \frac{\hat{f}_i - f_i}{\sigma} \leq u\left(\frac{\alpha}{2}\right)\right] \approx 1-\alpha$$

where $u\left(\frac{\alpha}{2}\right)$ is the critical value of the normal distribution, for significance level $\frac{\alpha}{2}$. This formula can be simply manipulated to yield the bounds for $\hat{f}_i$:

$$P\left[\left(1-u\left(\frac{\alpha}{2}\right)\sigma/f_i\right)f_i \leq \hat{f}_i \leq \left(1-u\left(\frac{\alpha}{2}\right)\sigma/f_i\right)f_i\right] \approx 1-\alpha$$

From this form we can see that error bound $\varepsilon$ can be computed as $\varepsilon = u\left(\frac{\alpha}{2}\right)\sigma/f_i$.

**Simultaneous intervals** Since we would like to compute the estimates for multiple ($m$) values of $f_i$, to achieve the bounded error with probability at least $1-\delta$, we set $\alpha = \frac{\delta}{m}$ following Bonferroni correction.

Let us finish this section by describing the method that can be used to choose the suitable parameters $r, t$ for Kmerlight to produce estimates with relative errors bounded by $\varepsilon$ with probability at least $1-\delta$.

Standard deviation $\sigma$ can be calculated by the equation (2.9) using $F_0, f_i, r, t$. Note that $\sigma$ depends logarithmically on $F_0$ and thus a very rough estimate of $F_0$ should be sufficient for parameter setting. We can demand the bounded precision only for sufficiently large $f_i$ by setting $f_i = F_0/\lambda$ ($\lambda = 1000$ for example). The estimates of larger $f_i$ will be more precise. Finally, using the relation $\varepsilon = u\left(\frac{\alpha}{2}\right)\sigma/f_i$ we can now directly calculate error bound $\varepsilon$ for a given probability $\delta$ and the parameters $r, t, \lambda$.

This result allows us to explore various values of the parameters $r, t$ and their influence on approximation errors without the need to actually run any computationally demanding program. For example, with the use of procedure described above, we can find the combination of parameters $r, t$, which minimizes the approximation errors for a fixed memory limit.

# Chapter 3

# Use of Histograms for Genome Size Estimation

In the beginning of this chapter we provide a more detailed description of a probabilistic model that can be used to estimate genome characteristics using only $k$-mer abundance histogram. This model was presented in [6, 5] and was implemented in a software CovEst.

Next we present our work. We study the accuracy of CovEst estimates based on approximate histograms, and we improve the accuracy by using our new modified version of Kmerlight.

## 3.1 CovEst

CovEst uses a $k$-mer abundance histogram $f = f_1, f_2, \ldots, f_m$ as its input, and finds the parameters $\theta = (c, e)$[1] that maximize the likelihood of the observed histogram $L(\theta) = P(f \mid \theta)$.

Using parameters $\theta$, CovEst, in fact, first generates the expected histogram shape $p_1, p_2, \ldots p_m$, where $p_i$ denotes the probability that a distinct $k$-mer occurs with abundance $i$. Then the likelihood of this shape is evaluated and thus different sets of parameters $\theta, \theta'$ can be compared.

**Generative model** While the average coverage of a single nucleotide in the genome is $c$, the average coverage of a $k$-mer is lower, since to produce a $k$-mer, a read must cover the whole $k$-mer. CovEst thus internally uses a value $c_k = (r - k + 1)/r \cdot c$ to describe the average coverage of a $k$-mer. This is the only assumption made which involves the reads.

[1] Coverage and error rate, as described in the introductory section 1.4. In a more advanced version, $\theta$ also includes parameters related to the extent of repeated regions.

In the following text, we will consider the $k$-mers to be uniformly distributed over the genome. This assumption enabled the authors to model the abundance of a $k$-mer with a Poisson distribution with mean $c_k$. Since $k$-mers with abundance zero are never observed, it is important to use a zero-truncated Poisson distribution, which leads to the following formula for $p_i$:

$$p_i = Poi_0(i; c_k) = \frac{c_k^i e^{-c_k}}{i!(1 - e^{-c_k})}$$

The authors assume that at sequencing, the probability that a nucleotide will be read correctly is $1 - \varepsilon$, and the probability that it will be read as a specific different nucleotide is $\varepsilon/3$. Therefore, if the Hamming distance of two $k$-mers is $s$, the probability that one was obtained by reading another one is $(\varepsilon/3)^s(1-\varepsilon)^{k-s}$. The distribution of the $k$-mers produced by $s$ errors can be therefore modelled with Poisson distribution with mean $\lambda_s = (\varepsilon/3)^s(1-\varepsilon)^{k-s} \cdot c_k$ and to compute $p_i$ we would use a mixture of these distributions:

$$p_i = \sum_{s=0}^{k} \alpha_s \cdot Poi_0(i; \lambda_s)$$

where the coefficients $\alpha_s$ sum to one ($\sum_{s=0}^{k} \alpha_s = 1$) and they correspond to fractions of $k$-mers produced by $s$ errors. From one $k$-mer $\binom{k}{s}3^s$ different $k$-mers can be obtained as a product of $s$ errors and the probability, that at least one of these $k$-mers will be observed is $(1 - e^{-\lambda_s})$. Therefore the coefficients $\alpha_s$ will be set to have values corresponding to $\alpha_s \propto \binom{k}{s}3^s(1 - e^{-\lambda_s})$.

The authors also modelled the repeats by adding together the contributions made by $k$-mers that were copied once, twice... which can be again expressed as a mixture of distributions. We will not describe the repeat modelling, since we will not use this extension in our thesis, but the details can be found in the original work [6, 5].

**Likelihood computation** To evaluate the validity of the parameters $\theta$, the theoretical shape of the histogram is compared to the observed histogram by likelihood computation $L(\theta) = P(f \mid \theta) = P(f_1, f_2, \ldots f_m \mid p_1, p_2, \ldots, p_m)$. In order to compare two histograms, the authors assume that vector $(f_1, f_2, \ldots, f_m)$ is sampled from a multinomial distribution $Mult(F_0, p_1, p_2, \ldots p_m)$ and thus the likelihood of the observed histogram is $\frac{F_0!}{f_1! \cdot f_2! \cdot \ldots f_m!} p_1^{f_1} p_2^{f_2} \ldots p_m^{f_m}$. Since we are interested only in maximizing the likelihood, log-likelihood is used and the constant terms are omitted, leaving the following expression to be maximized:

$$l(\theta) = \sum_{i=1}^{m} f_i \log p_i \tag{3.1}$$

Note that this method maximizes the likelihood for the multinomial distribution, and thus assumes that the height of $i$-th column of histogram follows distribution $Bin(F_0, p_i)$.

**Parameters optimization** As a starting point for parameter optimization, CovEst uses a very simple model to produce an initial guess of $\theta$. This model assumes that every erroneous $k$-mer has abundance one and thus the histogram shape can be modelled as a mixture of two distributions of erroneous and error-free $k$-mers. The simplicity of this model allows us to compute the optimal $c, e$ only by the Newton method.

Then a standard optimization algorithm L-BFGS-B is used to find the parameters $\theta$ that maximize the likelihood of the observed histogram, using the full model and the likelihood computation explained above.

**Experimental evaluation** The authors evaluated the precision of CovEst on generated genomes [6] and concluded that CovEst can estimate the coverage (and thus the genome size) within 1% of the true value for the common parameters. Moreover, CovEst produces reasonable estimates even for the datasets with coverage less than one or with error rate of 10%. Limited evaluation on a real sequencing reads was also performed.

## 3.2 CovEst's Performance on Kmerlight's Histograms

In this section we compare the accuracy of CovEst estimates based on exact histograms with estimates based on approximate histograms produced by the original and modified Kmerlight. We find that the approximate histograms can be used for genome size estimation and that our modified Kmerlight significantly increases the accuracy of CovEst estimates compared to the estimates based on the original Kmerlight.

In all experimental results we focus on the estimates of coverage $\hat{c}$. From these estimates, estimates of genome size can be produced simply by multiplying $\hat{c}$ by the amount of input data, thus the relative accuracy of the coverage estimate is the same as the relative accuracy of the genome size estimate.

For each set of genome parameters $(c, e, L)$ we performed 50 trials. In every trial we generated a genome and a set of reads with the procedure described in 2.2.1. Then we computed the exact histogram using Jellyfish software [7] and two approximate histograms using the original and the modified version of Kmerlight. Finally we ran CovEst on these three histograms and we collected the estimates of coverage $\hat{c}_j, \hat{c}_{ok}, \hat{c}_{mk}$.

As the default parameters we used genome size $L = 10^6$, coverage $c = 10$ and error rate $e = 0.01$, and in each of three experiments we altered the value of one parameter, while the two other parameters remained fixed.

In figures 3.1, 3.2, 3.3 we present boxplots of absolute errors of coverage estimates $(\hat{c} - c)$. The blue, orange and green boxes describe the distributions of errors from the histograms computed by Jellyfish, original Kmerlight and modified Kmerlight respectively. Vertical axes display values of absolute estimate errors. Note that these axes

use different scales since the estimate errors span across multiple orders of magnitude in different datasets.

Each box corresponds to 50 coverage estimates produced by CovEst. Kolmogorov-Smirnov tests do not reject normality with $p$-values higher than 0.4 for most boxes, and so we assume that CovEst estimates follow the normal distribution. The bottom borderline, the middle line and the top borderline of a box denote $25\%, 50\%$ and $75\%$ sample quantiles (or the first quartile, median and the third quartile) respectively. In normal distribution the mean is very close to median and approximately 70 per cent of data are located within one-$\sigma$-range of the mean. Therefore we can consider the middle line as a good approximation of mean and an inter-quartile-region size as a rough approximation of standard deviation.

In the end of this section we also present tables with mean errors and standard deviations of the coverage estimates, so the absolute values of these statistics can be compared.

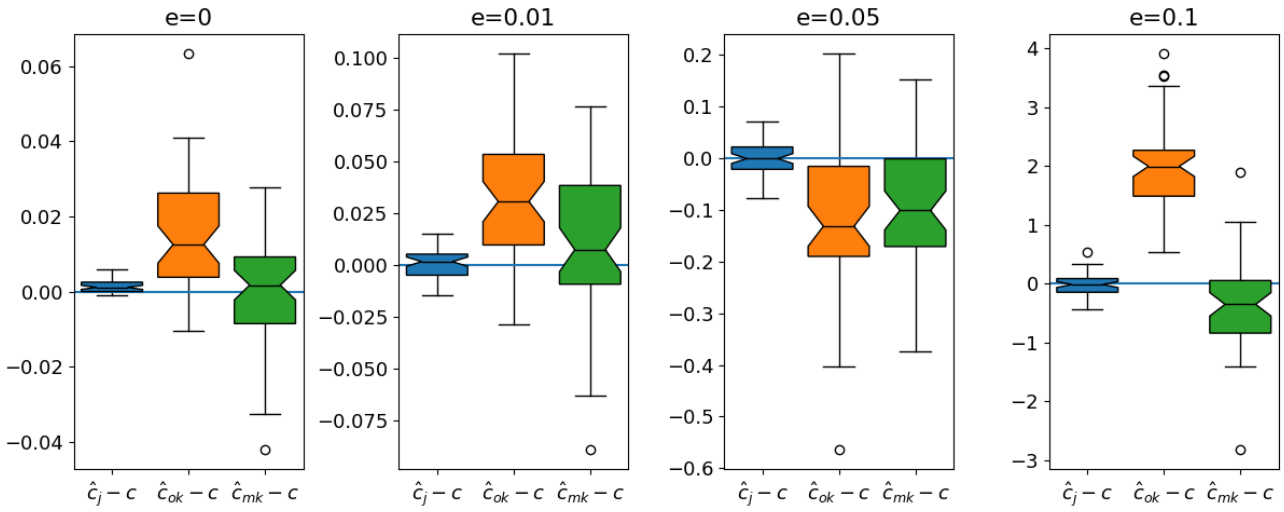


Figure 3.1: Distributions of coverage estimate errors for multiple sequencing error rates (each subfigure corresponds to one error rate). Each boxplot represents 50 estimates of the coverage on different generated genomes. Blue, orange and green boxes describe the estimate errors with Jellyfish, original Kmerlight and modified Kmerlight used to compute the $k$-mer abundance histogram.

**Effects of error rate on precision** In the first experiment (Figure 3.1) we investigate the effect of increasing sequencing error rates on the accuracy of coverage estimates.

On exact histograms CovEst produces unbiased estimates of coverage with their variance increasing with error rate. Estimates based on approximate histograms are clearly less accurate but still achieve a relatively good precision. Standard deviations of

$\hat{c}_{ok}$ and $\hat{c}_{mk}$ are similar, and they are approximately five times larger than the standard devations of $\hat{c}_j$.

Mean errors are consistently lower for modified Kmerlight than the errors of the original Kmerlight. Note that to prove the statistical significance of the difference between $\hat{c}_{ok}$ and $\hat{c}_{mk}$ it is not sufficient to compare the confidence intervals presented in boxplots (as boxplot notches), since $\hat{c}_{ok}$ and $\hat{c}_{mk}$ are clearly correlated. Pair Student's $t$-tests reject hypotheses $mean(\hat{c}_{ok} - \hat{c}_{mk}) = 0$ for three of four presented datasets, with exception of the dataset with $e = 0.05$ where the $p$-value is 0.3. Thus we conclude that the estimates based on modified Kmerlight's histograms are significantly better than the estimates based on original Kmerlight's histograms.

With modified Kmerlight, CovEst produces estimates with relative mean errors approximately <0.1%, <0.1%, 1%, 4% for sequencing error rates of $0, 0.01, 0.05, 0.1$ respectively, and we consider these estimates sufficiently accurate.

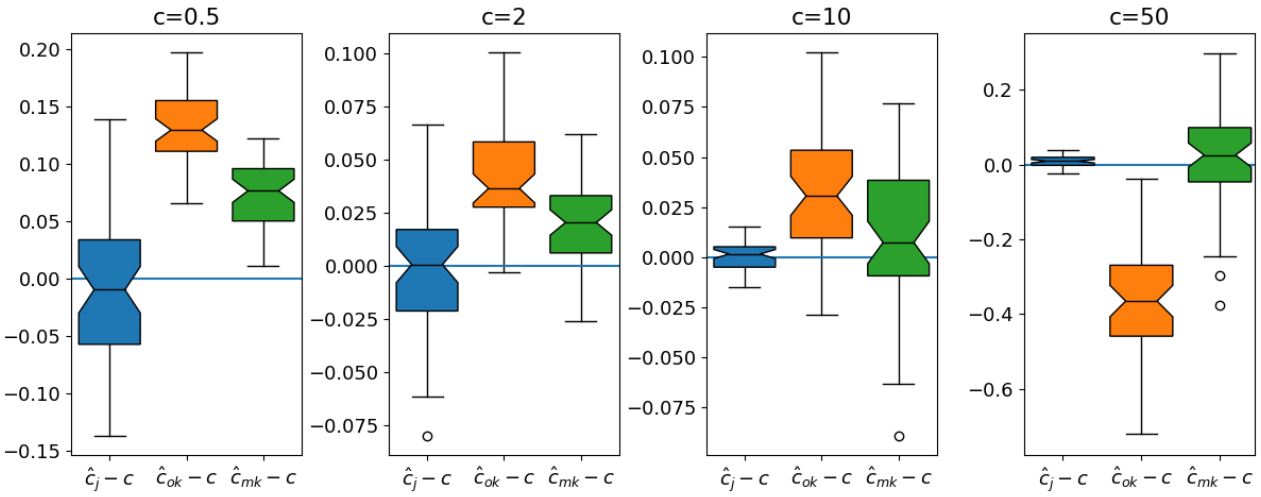


Figure 3.2: Distributions of coverage estimate errors for multiple coverages.

**Effects of coverage on precision** In the second experiment (Figure 3.2) we study the estimate accuracy for different coverages.

All CovEst estimates $\hat{c}_j$ in our experiment range in $30\%, 6\%, 0.3\%, < 0.1\%$ intervals around the true coverage for coverages of 0.5, 2, 10 and 50. The variance of $\hat{c}_j$ is comparable to variance of $\hat{c}_{ok}, \hat{c}_{mk}$ for two lower coverages but with increasing coverage, estimates based on approximate histograms become less accurate. However, since the maximal errors reach values of only 0.3, which is less than 1% of the true coverage $c = 50$, we also consider these estimates as useful.

Modified Kmerlight significantly (verified by t-tests) outperforms the original Kmerlight on all four presented datasets.

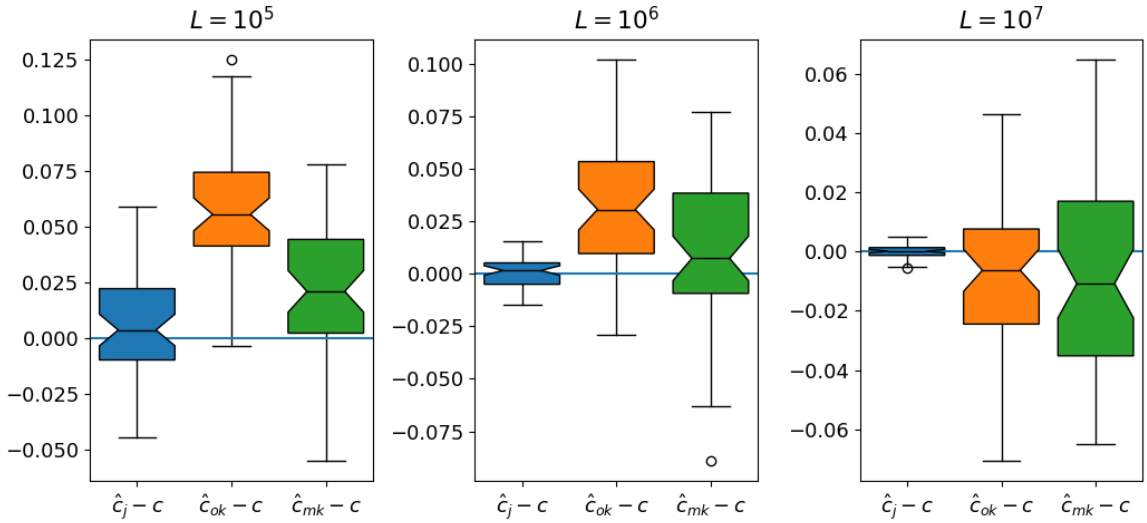


Figure 3.3: Distributions of coverage estimate errors for multiple genome sizes.

**Effects of genome size on precision** With increasing genome size, CovEst's estimates become more precise on exact histograms, and estimates on approximated histograms maintain a roughly constant precision (for the examined genome sizes).

As the coverage estimate errors are bounded by 1% for all datasets, we also consider the approximate histograms sufficiently accurate for the genome size estimation.

Modified Kmerlight reaches smaller mean error on first two datasets than the original Kmerlight, while the difference of estimates for the largest genome is insignificant.

With approximate histograms used, CovEst still maintains errors bounded by 1%[2]. Therefore we conclude that Kmerlight's histograms can be used for sufficiently precise genome size estimation.

With increasing error rate, coverage and genome size, the number of unique $k$-mers $F_0$ increases. Note that $F_0$ is the main factor influencing Kmerlight's absolute precision, since based on $F_0$, the level $w^+$ (and expectedly also levels $w_i^*$) is selected, which directly influences the sampling probability $p_s$. With increasing $L$ the ratios of $f_i/F_0$ do not change very much, and thus the relative accuracy of Kmerlight's $\hat{f}_i$ estimates do not change as significantly as with increasing error rate and coverage. Therefore, also the precision of the coverage estimates remains approximately constant with varying genome size for Kmerlight.

Note that we used Kmerlight with parameter $r = 2^{15} \approx 30\,000$ (arrays of only 30 000 counters) to process histograms with approximately $10^7$ unique $k$-mers. There are, of course, seven instances of sketch and 64 levels of these counters in each sketch[3], but as we mentioned in discussion in the end of section 2.4, the factor of levels can be reduced. While all exact methods need to use $O(F_0)$ memory (to store all $k$-mers or at

[2]For all but extreme cases. For $e > 0.05, c = 2$ are the errors are bounded by 10%.

[3]As a result, approximately $1.5 \times 10^7$ memory words are used, which is approximately 60MB.

least to prevent hashing collisions), with 10-100 times smaller memory requirements, Kmerlight is able to produce histogram estimates from which a genome size can be estimated with 1% precision.

Furthermore, our observations suggest that these Kmerlight's settings may be sufficient to produce precise estimates even for larger genomes.

**Tables of the experimental results**

| | Mean error | | | Standard deviation | | |
|---|---|---|---|---|---|---|
| Dataset | $\hat{c}_j - c$ | $\hat{c}_{ok} - c$ | $\hat{c}_{mk} - c$ | $\hat{c}_j - c$ | $\hat{c}_{ok} - c$ | $\hat{c}_{mk} - c$ |
| $e = 0$ | 0.001 | 0.015 | 0.000 | 0.002 | 0.015 | 0.014 |
| $e = 0.01$ | 0.001 | 0.031 | 0.011 | 0.007 | 0.030 | 0.034 |
| $e = 0.05$ | 0.002 | -0.118 | -0.093 | 0.036 | 0.140 | 0.124 |
| $e = 0.1$ | -0.002 | 1.934 | -0.380 | 0.189 | 0.842 | 0.747 |

Table 3.1: Mean errors and standard deviations of CovEst estimates based on three types of histograms (in columns for Jellyfish, original Kmerlight, modified Kmerlight) for various error rates (in rows).

| | Mean error | | | Standard deviation | | |
|---|---|---|---|---|---|---|
| Dataset | $\hat{c}_j - c$ | $\hat{c}_{ok} - c$ | $\hat{c}_{mk} - c$ | $\hat{c}_j - c$ | $\hat{c}_{ok} - c$ | $\hat{c}_{mk} - c$ |
| $c = 0.5$ | -0.010 | 0.133 | 0.072 | 0.065 | 0.030 | 0.029 |
| $c = 2$ | -0.001 | 0.042 | 0.019 | 0.031 | 0.022 | 0.020 |
| $c = 10$ | 0.001 | 0.031 | 0.011 | 0.007 | 0.030 | 0.034 |
| $c = 50$ | 0.008 | -0.360 | 0.024 | 0.014 | 0.141 | 0.147 |

Table 3.2: Mean errors and standard deviations of CovEst estimates on three types of histograms for various coverages.

| | Mean error | | | Standard deviation | | |
|---|---|---|---|---|---|---|
| Dataset | $\hat{c}_j - c$ | $\hat{c}_{ok} - c$ | $\hat{c}_{mk} - c$ | $\hat{c}_j - c$ | $\hat{c}_{ok} - c$ | $\hat{c}_{mk} - c$ |
| $L = 10^5$ | 0.006 | 0.059 | 0.022 | 0.024 | 0.028 | 0.032 |
| $L = 10^6$ | 0.001 | 0.031 | 0.011 | 0.007 | 0.030 | 0.034 |
| $L = 10^7$ | 0.000 | -0.008 | -0.008 | 0.002 | 0.025 | 0.032 |

Table 3.3: Mean errors and standard deviations of CovEst estimates on three types of histograms for various genome sizes.

# Conclusion

The objective of our thesis was to study the character of errors of the approximate histograms, and to determine whether these histograms can be used for sufficiently precise genome size estimates.

We focused on software Kmerlight [12], since it is, to our knowledge, the fastest and most memory efficient algorithm for $k$-mer abundance histogram approximation.

From the experimental observations, we discovered that Kmerlight's estimates of $\hat{f}_i$ are biased towards higher values for histogram columns with low $f_i/F_0$ ratio. We identified the source of bias: Kmerlight chooses levels $w_i^*$ that maximize $t_i^{(w)}$, the number of collision-free counters with value $i$; and we proposed and tested a modification which successfully eliminated the bias under the assumption that all hashing collisions can be detected. Our modified version of Kmerlight uses only one level $w^+$ which maximizes $E(t_i^{(w)})$. To select $w^+$ we only need an estimate of $F_0$ and therefore, we believe that a more memory-efficient method of histogram approximation can be devised by further improvement of Kmerlight.

As we could compute histograms with zero mean estimation error $E(\hat{f}_i - f_i) = 0$, we focused on the variance of approximation errors. We were able to precisely model the distribution of Kmerlight's errors with normal distribution for sufficiently high values of $f_i$ and we produced a quantitative estimate of errors' variance for all reasonable values of $f_i$ (those $f_i$ for which $E(t_i^{(w^+)}) > 1$).

Using the estimate of errors, we proposed a method for selecting suitable parameters for Kmerlight, and generally, we believe that the ability to estimate the extent of Kmerlight's errors without the actual histogram computation, makes a more practical tool. Since Kmerlight can be generalized to approximate an abundance histogram of any items, not only of $k$-mers, the applicability of our work is thus not limited to the bioinformatics context.

With the first part of our objective successfully fulfilled, we turned our attention towards the genome size estimation and CovEst software. To investigate the precision of genome size estimates based on approximated histograms, we performed experiments on simulated genomes.

We examined the influence of genome parameters on the accuracy of coverage esti-

mates. With increasing error rate and coverage, the number of unique $k$-mers increases and thus the histogram approximations produced by Kmerlight are less precise. Therefore also the estimates of the coverage are less precise. The genome size does not seem to significantly influence the relative precision of coverage estimates based on Kmerlight's histograms. Since the ratios of $f_i/F_0$ change only slightly with various genome sizes, relative errors of Kmerlight also do not change much.

Our experimental evaluation showed that even with approximated histograms, an estimation error of less than 1% can be achieved for most usual datasets (sets of reads with $c > 2$ and $e < 0.05$), and also that estimates based on modified Kmerlight's histograms achieve lower mean errors than the estimates of the original Kmerlight.

We note that we did not test CovEst estimates on real genomes, but only on simulated genomes. A future work may clarify how other biological phenomena influence CovEst's estimates.

# Bibliography


[1] Rayan Chikhi and Paul Medvedev. Informed and automated k-mer size selection for genome assembly. *Bioinformatics*, 30(1):31, 2013.

[2] David Williams et al. Rapid quantification of sequence repeats to resolve the size, structure and contents of bacterial genomes. *BMC Genomics*, 14(1):537, 2013.

[3] Stefan Kurtz et al. A new method to compute k-mer frequencies and its application to annotate large repetitive plant genomes. *BMC Genomics*, 9(1):517, 2008.

[4] Ziv Bar-Yossef et al. Counting distinct elements in a data stream. In *Proceedings of the 6th International Workshop on Randomization and Approximation Techniques*, RANDOM '02, pages 1–10, London, UK, UK, 2002. Springer-Verlag.

[5] Michal Hozza. *Algorithms and Models for Analysis of DNA Sequencing Data*. PhD thesis, Univerzita Komenského v Bratislave, 2016. Unfinished unpublished thesis, retrieved from `https://github.com/mhozza/dissertation`.

[6] Michal Hozza, Tomáš Vinař, and Broňa Brejová. How Big is That Genome? Estimating Genome Size and Coverage from k-mer Abundance Spectra. In *String Processing and Information Retrieval (SPIRE)*, volume 9309 of *Lecture Notes in Computer Science*, pages 199–209, London, UK, September 2015. Springer.

[7] Guillaume Marçais and Carl Kingsford. A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics*, 27(6):764, 2011.

[8] Páll Melsted and Jonathan Pritchard. Efficient counting of k-mers in DNA sequences using a bloom filter. *BMC Bioinformatics*, 12(1):333, 2011.

[9] Páll Melsted and Bjarni V. Halldórsson. Kmerstream: streaming algorithms for k-mer abundance estimation. *Bioinformatics*, 30(24):3541–3547, 2014.

[10] Paul R. Rider. Variance of the median of small samples from several special populations. *Journal of the American Statistical Association*, 55(289):148–150, 1960.

[11] Guillaume Rizk, Dominique Lavenier, and Rayan Chikhi. DSK: k-mer counting with very low memory usage. *Bioinformatics*, 29(5):652–653, January 2013.

[12] Naveen Sivadasan, Rajgopal Srinivasan, and Kshama Goyal. Kmerlight: fast and accurate k-mer abundance estimation. *CoRR*, abs/1609.05626, 2016.

[13] Wikipedia. Median. In Wikipedia, the free encyclopedia, 2017. Retrieved May 12, 2017, from `https://en.wikipedia.org/wiki/Median#Sampling_distribution`.

[14] Wikipedia. Streaming algorithm. In Wikipedia, the free encyclopedia, 2017. Retrieved April 18, 2017, from `https://en.wikipedia.org/wiki/Streaming_algorithm`.

# Appendix A – Software and Scripts

The source codes of used software and scripts can be found at GitHub:

- Scripts that were used to generate the experimental data and figures: `https://github.com/maaario/bachelor-thesis-scripts`
- Our modified version of Kmerlight software: `https://github.com/maaario/kmerlight` . The most important modification, the new method of selecting the level $w^+$, is implemented in function `computeFJ_modified` in the file `src/CountSketchInstance.h`.

  The original version of Kmerlight can be downloaded from `https://github.com/nsivad/kmerlight`.
- CovEst software [6] can be downloaded from `https://github.com/mhozza/covest`. Apart from using CovEst to compute coverage estimates, we also used scripts from the folder `tools/simulator` to generate simulated genomes and reads.