J. Hlaváčová (Ed.): ITAT 2017 Proceedings, pp. 27–34
CEUR Workshop Proceedings Vol. 1885, ISSN 1613-0073, © 2017 M. Lipovský, T. Vinař, B. Brejová

Approximate Abundance Histograms and Their Use for Genome Size Estimation

Mário Lipovský, Tomáš Vinař, Broňa Brejová

Faculty of Mathematics, Physics, and Informatics,
Comenius University, Mlynská dolina, 842 48 Bratislava, Slovakia

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 paper, we study a recent 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 support our theoretical model by experimental data. Finally, we use another tool, CovEst, to compute genome size estimates from approximate 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 memory.

1 Introduction

A genome is a collection of DNA molecules storing genetic information in a cell. It can be represented as a set of long strings over the alphabet $\Sigma = \{A, C, G, T\}$ (each string corresponding to one chromosome). In a DNA sequencing experiment, many reads are produced from a genome. These reads are short substrings obtained from random locations of the genome, potentially with some sequencing errors. For a genome to be analyzed, the DNA sequence of individual chromosomes must be first assembled from the reads, but the process of genome assembly is computationally demanding and error-prone.

However, it is possible to estimate the genome size and some other characteristics without the need of genome assembly [2, 9, 5, 8]. These estimates are computed from a summary statistic of reads called the k-mer abundance histogram. A k-mer is a substring of length exactly k and the histogram summarizes the number of occurrences of individual k-mers in the input set of reads.

Most of the histogram computing methods count the occurrences of each k-mer in the input data using hash tables or suffix arrays [6, 4, 3]. Consequently, 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 [8], 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 input for genome size estimation [8], however the impact of the approximation errors on the accuracy of the resulting estimates was not evaluated. In this paper, we study the character of errors of the Kmerlight's 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 propose an unbiased modification of Kmerlight. 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 [2] to estimate the sizes of simulated genomes from approximate histograms produced by Kmerlight. We describe how different properties of the input influence the accuracy of the estimates, and we compare the estimates based on exact histograms to the estimates based on approximate histograms.

2 Errors in Kmerlight's Approximate Abundance Histograms

A k-mer is a substring of length exactly k. A read of length r thus contains $r - k + 1$ k-mers. If a k-mer occurs i times among all input reads, we say its abundance is i. In this paper, we use the value $k = 21$ as in previous works [9, 2].

Definition 1. *The k-mer abundance histogram is a sequence $f = f_1, f_2, \ldots f_m$, 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.*

Counting the abundance of each individual k-mer appearing in a large input file requires large memory. As we are only interested in the histogram, the problem of k-mer counting can be avoided, allowing us to estimate the histogram more efficiently. We can 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.

We will consider streaming algorithms, which in general process a sequence of items (in our context k-mers) in a single pass using only a limited amount of memory and time. 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 algorithms for estimating the number of distinct elements in a stream ($F_0 = \sum_{i=1}^{m} f_i$) with theoretical guarantees [1]. In 2014, Melsted and Halldórsson [5] implemented and extended Algorithm 2 from the aforementioned paper [1] and used it for k-mer counting. Their algorithm KmerStream can also estimate f_1, the number of k-mers with abundance one.

KmerStream was further improved by Sivadasan et al. in 2016 [8], and their software Kmerlight estimates the whole histogram $(f_1, \dots, f_m)$. As we focus on Kmerlight in our work, we will describe it in detail.

2.1 Kmerlight

Kmerlight maintains a sketch of processed k-mers. Its output are estimates $\hat{F}_0, \hat{f}_1, \dots, \hat{f}_m$ obtained from the final content of the sketch. The scheme uses parameters W, r, u and two hash functions $g : \Sigma^k \to \{0, \dots, 2^W - 1\}$ and $h : \Sigma^k \to \{0, \dots, r-1\} \times \{0, \dots, u-1\}$. In the analysis, we will assume that g and h hash each string from Σ^k uniformly and independently over their range of values.

The sketch consists of W levels. Each level w has a hash table with r counters $T_w[0], \dots, T_w[r-1]$. Each counter stores its value $T_w[c].v$ (the abundance of a particular k-mer) and an auxiliary information $T_w[c].p \in \{0, \dots, u-1\}$.

To process an input k-mer x we first select its level w as the number of trailing zeroes in the binary representation of $g(x)$. As a result, the probability of selecting level w is $1/2^w$. Next, using has function h, the k-mer is hashed into a pair (c, j) and processed as follows:

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

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

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 this level remains empty is approximately $p = (1 - \frac{1}{r})^{F_0/2^w}$. In this estimate and in all subsequent analyses, 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 the mean.

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 $t_0^{(w)}$. Using the assumption $t_0^{(w)} \approx r \cdot p$ we can derive the estimator of F_0:

$$\hat{F}_0 = 2^w \cdot \frac{\ln(t_0^{(w)}/r)}{\ln\left(1 - \frac{1}{r}\right)}. \tag{1}$$

To estimate the number of distinct k-mers F_0 using this formula, we choose level $w = w^*$, so that the number of empty counters $t_0^{(w)}$ at this level is closest to $r/2$. It has been shown that selecting this level provides a bounded error of $\hat{F}_0$ with a guaranteed probability [1].

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 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

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

If we denote the number of observed collision-free counters with value i as $t_i^{(w)}$, we can derive the estimator of f_i:

$$\hat{f}_i = t_i^{(w)} \cdot 2^w \cdot \left(1 - \frac{1}{r}\right)^{1 - F_0/2^w} \tag{3}$$

To estimate f_i, the algorithm selects level $w = w_i^*$ so that it maximizes $t_i^{(w)}$ – the number of observed collision-free counters with value i.

Undetected collisions. The value $t_i^{(w)}$ is based on the number of non-dirty counters with value i, but these include both true positives (collision-free counters) and false positives (counters with undetected collisions). The expected number of false positive at one level is at most r/u, where $0, \dots u-1$ is the range of auxiliary information [8]. Parameter u can be set to make false positives negligible; thus we will assume that all collisions are being detected.

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

Accuracy and complexity. Parameters r, u ant t provide a trade-off between the memory and accuracy. Assuming that $W = \Theta(\log F_0)$, the algorithm uses $O(t \cdot r \cdot \log(F_0))$ memory words, and 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 loose

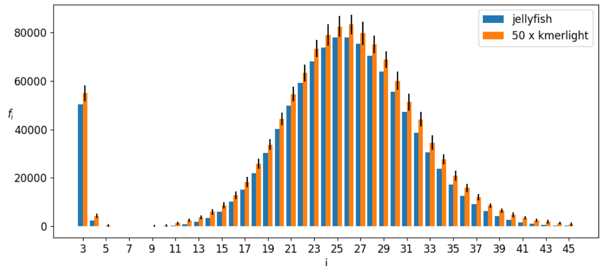


Figure 1: The exact values f_i produced by Jellyfish (blue) and approximate values $\hat{f}_i$ produced by Kmerlight averaged from 50 trials (orange). The error bars express the standard deviation of $\hat{f}_i$. Columns for $i = 1, 2$ are omitted, having values approximately $10^7, 10^6$ respectively.

constants, these asymptotic estimates cannot be used to set parameters for obtaining desirable error bounds. The accuracy of the algorithm was tested experimentally with arbitrary parameters $W = 64, t = 7, r \in \{2^{16}, 2^{18}\}, u = 2^{13}$.

2.2 Empirical Study of Approximation Errors

To study the character of approximation errors, we start with several observations on generated data. We have first generated a genome: a random sequence $g_1 \dots g_L$ consisting of L characters A, C, G, T, each with probability $1/4$. Next we have generated $c \cdot L/\ell$ reads of length $\ell = 100$, where c is a parameter describing the depth of coverage of the genome by reads. For each read, we have uniformly selected a random starting position $s \in \{1, \dots, L - \ell + 1\}$. The read then consists of characters $g_s g_{s+1} \dots g_{s+\ell-1}$. Finally we have simulated sequencing errors by modifying each base randomly with probability e. Our initial experiment has used a single data set with parameters $L = 10^6$, $c = 50$, and $e = 0.02$.

For this data set, Figure 1 shows the exact k-mer abundance histogram ($k = 21$) produced by Jellyfish [4] and the means and standard deviations of the estimates $\hat{f}_i$ from 50 Kmerlight runs on the same input. Kmerlight was run with parameters $W = 64$, $t = 7$, $r = 2^{15}$, $u = 2^{13}$.

Perhaps the most striking feature of this histogram is that Kmerlight systematically overestimates values of f_i. In Section 2.3, we clarify the source of this bias and present an unbiased estimator.

Kmerlight guarantees precise estimates of those values of f_i that are close to F_0. Unfortunately, in sequencing data, sequencing errors often create many unique k-mers, and thus we have f_1 close to F_0, while the remaining f_i are much smaller. In the extreme case of very low values of f_i, the probability that at least one k-mer with abundance i becomes stored in a collision-free counter at any level approaches to zero. If no k-mer survives the collisions ($t_i^{(w)} = 0$) then $\hat{f}_i = 0$. In our experiment, Kmerlight produced zero estimates for $f_i < 500$.

Figure 1 shows that the absolute error $(\hat{f}_i - f_i)$ and variance of $\hat{f}_i$ decrease with decreasing f_i. However, relative errors $(\hat{f}_i - f_i)/f_i$ and their variance increase with decreasing f_i (data not shown). Kmerlight guarantees bounded errors for high values of f_i, but a theoretical quantitative analysis of the error distribution was not presented in the previous work [8]. We provide a quantitative estimate of the variance in Section 2.4.

2.3 Kmerlight Approximation Bias and Unbiased Version

As we will explain in this section, the bias of Kmerlight estimates towards higher values is caused by its selection of level w_i^* used to calculate $\hat{f}_i$. Level w_i^* is chosen to maximize $t_i^{(w)}$ – the number of collision-free counters with value i at level w. 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.

Analytical w^+. To understand which levels w_i^* are selected by Kmerlight, we will analytically find the level w_i^+ that maximizes $E(t_i^{(w)})$ – the expected number of collision-free counters with value i.

As shown in equation (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 from inequalities $E(t_i^{(w_i^+ - 1)}) \le E(t_i^{(w_i^+)}) \le E(t_i^{(w_i^+ + 1)})$. By manipulating the inequalities we get

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

and finally, we can calculate w_i^+:

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

Symbol lg denotes the binary logarithm. Note these inequalities have at most two integer solutions.

The choice of level w_i^+ does not depend on values of i or f_i, but only on F_0 and r. Since $w_1^+ = w_2^+ = \dots = w_m^+$, from now on we will refer to this level simply by w^+. This level also maximizes the expected number of all non-empty collision-free counters $E(t^{(w)})$, where $t^{(w)} = \sum_{i=1}^m t_i^{(w)}$.

Explanation of bias. The number of collision-free counters at levels $w^+ - 1$ and $w^+ + 1$ is typically similar to the number of collision-free counters at level w^+. There are two times more k-mers hashed into level $w^+ - 1$ than into w^+, but also more collisions happen at $w^+ - 1$. These two effects partially cancel each other and maintain similar values of $E(t^{(w)})$ for $w = w^+ - 1, w^+, w^+ + 1$. As a result, any of these levels can hold the maximal $t_i^{(w)}$ and become chosen by Kmerlight as its w_i^*. This typically leads to values of $t_i^{(w_i^*)}$ higher than the expected value $E(t_i^{(w)})$ for fixed level $w = w_i^*$. To obtain $\hat{f}_i$, value $t_i^{(w_i^*)}$ is multiplied by

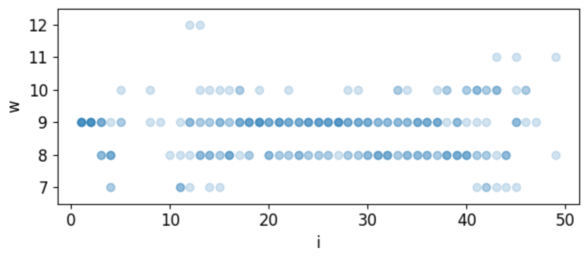


Figure 2: Each column shows up to seven levels w_i^* that were selected by one of $t = 7$ instances of Kmerlight in the experiment from Section 2.2. If a level was chosen by more instances, its circle is darker. Note that if no counters hold value i in the whole sketch, Kmerlight's instance does not choose any level.

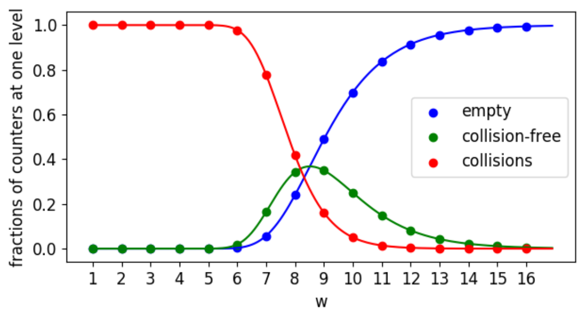


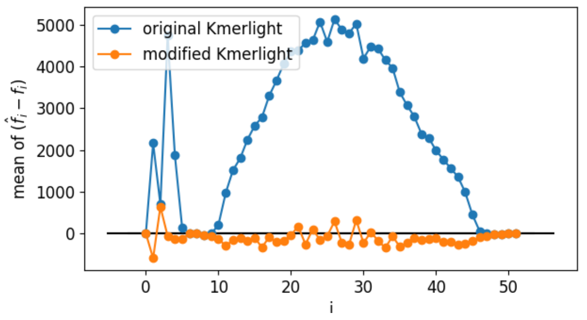
Figure 3: Values p_e, p_{cf}, p_c represent the theoretical fractions of empty, collision-free and collided counters at each level respectively. To make the values comparable to the results from Section 2.2, we set $F_0 = 1.2 \times 10^7, r = 2^{15}$.

$2^{w_i^*} \cdot \left(1-\frac{1}{r}\right)^{1-F_0/2^{w_i^*}}$. Thus biased $t_i^{(w_i^*)}$ also leads to biased estimator $\hat{f}_i$.

To illustrate this, Figure 2 shows values w_i^* for different f_i extracted from one Kmerlight run. The analytical level $w^+ = 9$ is selected most frequently, but 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 a high relative variance. The maximal $t_i^{(w)}$ can thus be reached at levels more distant from $w^+ = 9$.

In order to further demonstrate the similarity of $E(t^{(w)})$ at the levels close to w^+, Figure 3 displays the theoretical fractions of empty, collided, and non-empty collision-free counters at different levels of the sketch. Focusing on a single level w, let us denote as p_{cf} the probability that a single counter is non-empty and collision-free, p_e the probability that this counter is empty, and p_c the probability that it holds a collision. The number of distinct k-mers hashed into one counter (X) follows a binomial distribution, $X \sim \text{Bin}(F_0/2^w, 1/r)$, and we can use this distribution to compute $p_{cf} = P[X = 1]$, $p_e = P[X = 0]$, and $p_c = P[X > 1]$. Note that the expected number of non-empty collision-free counters at one level is $E(t^{(w)}) = r \cdot p_{cf}$.

The presented settings incidentally represent an unfavorable situation for Kmerlight, because $E(t^{(8)}) \approx E(t^{(9)})$. 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 $t_i^{(w_i^*)} > E(t^{(w^+)})$.



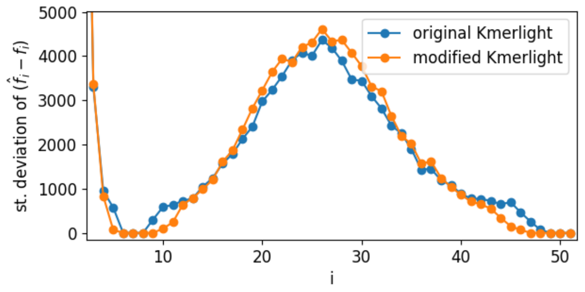


Figure 4: Mean (top) and standard deviation (bottom) of estimate errors $(\hat{f}_i - f_i)$ produced by the original (blue) and modified (orange) Kmerlight in 300 trials.

Removing bias from estimates of f_i. Our observation suggests a simple modification to remove the bias from estimates of f_i. In particular, we will use the level w^+ that maximizes the expected number of collision-free counters $E(t^{(w)})$, instead of the level w_i^* that maximizes the observed number of collision-free counters $t_i^{(w)}$.

The modified Kmerlight creates sketches and estimates F_0 in the same way as the original algorithm. Then using $\hat{F}_0$ and r, it calculates w^+, as discussed above. Finally, values of f_i are estimated from the observed counts of collision-free counters at level w^+ by equation (3).

Note that we do not prove analytically that this estimator is unbiased. Simplifications in our analysis may cause some small biases in the estimator, but we demonstrate experimentally that the estimator works well on our generated data. In Figure 4, we compare the original and modified Kmerlight in 300 trials of both versions on the same data as in Section 2.2. While the original Kmerlight overestimates f_i significantly, the modified Kmerlight achieves $E(\hat{f}_i) \approx f_i$, without much change in the variance.

2.4 Evaluation of Approximation Variance

In this section, we estimate the variance of $\hat{f}_i$. We will consider estimates obtained at a fixed level w (for example

$w = w^+$). Recall that $E(t_i^{(w)}) = f_i/2^w \cdot (1-1/r)^{F_0/2^w-1}$. 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, and we discard each k-mer with probability $1-p_s$. Note that this approach assumes that each k-mer remains collision free independently of others, which is not the case.

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

$$Var(\hat{f}_i) = Var\left(\frac{t_i}{p_s}\right) = \frac{1}{p_s^2} \cdot Var(t_i) = \frac{f_i \cdot (1-p_s)}{p_s} \quad (5)$$

Effect of medians. Kmerlight chooses the estimate $\hat{f}_i$ as a median of estimates of t independent sketches: $\hat{f}_i = \text{med}(\hat{f}_i^{(1)}, \ldots \hat{f}_i^{(t)})$. 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[1] normal:

$$\text{med}(x) \mathrel{\dot\sim} N\left(\bar{x}, \frac{1}{4nf(\bar{x})^2}\right). \quad (6)$$

As the binomial distribution of $t_i^{(w)}$ is a discrete distribution, 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 μ is $\frac{1}{\sqrt{2\pi\sigma^2}}$.

Using approximations (5), (6), variance of $\hat{f}_i$ selected as a median of t instances can be derived as follows:

$$Var(\hat{f}_i) \approx \frac{2\pi}{4t} Var(\hat{f}_i^{(l)}) = \frac{\pi}{2t} \frac{f_i \cdot (1-p_s)}{p_s}. \quad (7)$$

Experiments. We ran the modified Kmerlight in 300 trials on the data presented in Section 2.2. Figure 5 shows histograms of values of $\hat{f}_i$ for selected[2] values of i. We compare these histograms with two normal distributions. The best normal fit is a Gaussian with its mean and standard deviation obtained from the observed values of $\hat{f}_i$. The plotted theoretical prediction uses the exact value of f_i as its mean and the variance is calculated according to equation (7) using the exact values of f_i and F_0.

The quality of normal approximation largely depends on the true value of f_i. According to our approximation, the expected number of counters with value i at level w^+ is $E(t_i) = f_i \cdot p_s$. For this dataset, $w^+ = 9$, and thus the sampling probability p_s is approximately $1/2^9 \cdot \frac{1}{2} \approx 1/1000$ when we use the upper bound from equation (4).

For the lowest values of f_i (i.e. f_6, f_{10}), we have $E(t_i) < 1$. So typically no k-mers survive the collisions at level w^+, and thus $t_i = 0$ and $\hat{f}_i = 0$. Due to the use of medians, at least one k-mer must survive in at least four instances to produce $t_i \geq 1$. As a result, we obtain $\hat{f}_i = 0$ in all trials even for $f_{10} = 140$. Since the estimates $\hat{f}_i$ are always zero, our estimates of variances for these f_i are very imprecise.

If the value f_i is such that $E(t_i)$ is around 1 (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 5. The approximation with normal distribution is not precise for these values f_i, since the distribution of $\hat{f}_i$ is clearly discrete.

Finally, for higher f_i, where $E(t_i) \gg 1$, 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 row of Figure 5.

We have also applied Kolmogorov-Smirnov tests to test the normality of $\hat{f}_i$. These tests reject normality for $f_i < 20{,}000$ with our dataset of 300 trials at the significance level of 5%. Overall, we conclude that for sufficiently high values of f_i the distribution of $\hat{f}_i$ can be approximated by Gaussian with variance calculated by (7).

Finally we present the comparison of Kmerlight's variance and its theoretical prediction in Figure 6. In this experiment, our estimate of the standard deviation of Kmerlight's estimates has error less than 5% even for lower f_i with values around $1000 \approx 1/p_s$., where the normal approximation was still inadequate. The experiment further suggests that an accurate estimate of variance for the lowest values of $f_i < 100$ would be zero.

Figure 6 also reveals a difference in variances between the 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 to estimate $\hat{f}_i$. We did not include

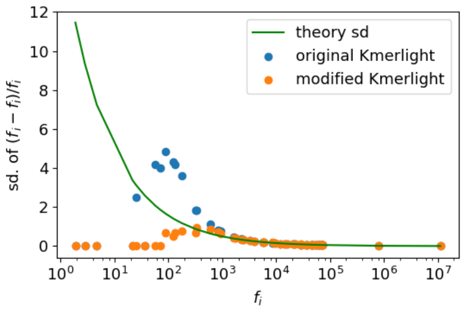


Figure 6: The blue and orange points show the standard deviation of relative error $(\hat{f}_i - f_i)/f_i$ of estimates $\hat{f}_i$ computed by the original and modified Kmerlight respectively in 300 runs on dataset from Section 2.2. The green line represents the approximation of standard deviation calculated using equation (7).

[1]Even for a sample of only 7 observations drawn from the normal distribution, the relative error of this approximation is only about 6% [7].

[2]To select values $i = 6, 10, 13, 16, 32, 23$, we have sorted the values f_i and selected each eighth value. We have also included f_1, f_2, since they differ from other f_i by orders of magnitude.

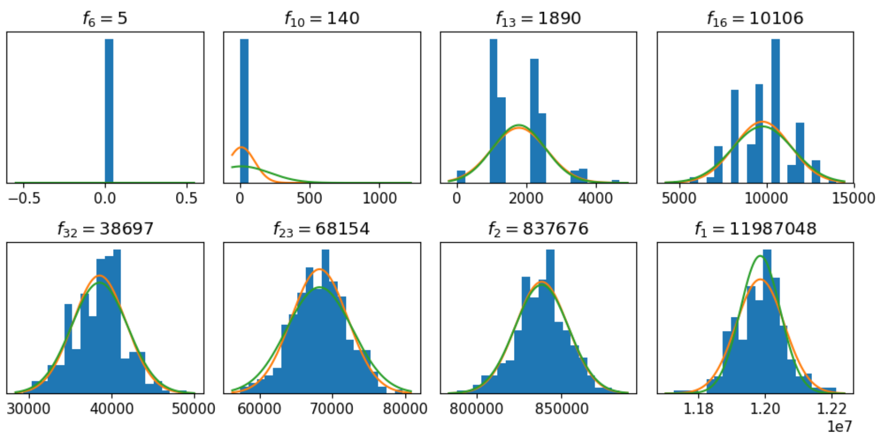


Figure 5: Empirical probability densities (blue histograms), the best normal fits (orange lines) and theoretical estimates (green) of distributions of $\hat{f}_i$ for multiple abundances. Histograms come from 300 runs of modified Kmerlight. Horizontal axes display the values of estimates $\hat{f}_i$.

this effect into the variance estimation, hence the estimates for these f_i are not precise for the original Kmerlight.

2.5 Choice of Kmerlight's Parameters

Sivadasan et al [8] provide theoretical bounds on Kmerlight's approximation errors, but they do not provide methods for setting parameters in practical scenarios. In this section, we show how to set the parameters in order to achieve relative error bounded by ε with probability $1-\delta$, under the assumption that $\hat{f}_i$ is distributed according to the normal approximation derived in the previous section.

We first derive a two-sided prediction interval of $\hat{f}_i$. Let $u\left(\frac{\alpha}{2}\right)$ be the critical value of the normal distribution for significance level $\frac{\alpha}{2}$. By standardization we get $(\hat{f}_i - f_i)/\sigma \dot{\sim} N(0,1)$, and so we have

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

If we set $\varepsilon = u\left(\frac{\alpha}{2}\right)\sigma/f_i$, we obtain the bounds for $\hat{f}_i$:

$$P\left[(1-\varepsilon)f_i \le \hat{f}_i \le (1+\varepsilon)f_i\right] \approx 1-\alpha.$$

To achieve bounded error with probability at least $1-\delta$ simultaneously for m different values of i, we set $\alpha = \frac{\delta}{m}$ following the Bonferroni correction.

Standard deviation σ can be calculated by the equation (7) using F_0, f_i, r, t. Note that σ depends logarithmically on F_0 and thus a rough estimate of F_0 is sufficient. We restrict 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 even more precise. Finally, by using $\varepsilon = u\left(\frac{\alpha}{2}\right)\sigma/f_i$, we can calculate error bound ε for a given probability δ and parameters r, t, λ. This allows us to efficiently explore various values of r, and t and their influence on approximation errors. For example, we can find parameters which minimize the approximation error for a fixed memory limit.

3 Genome Size Estimation

One motivation for computing k-mer abundance histograms from sequencing data is to obtain genome size estimates. Here we use CovEst software [2] developed in our group to compare the accuracy of genome size estimates produced from the exact and approximate histograms.

We use data generated as described in Section 2.2, but we vary parameters L (genome size), c (genome coverage by reads), e (sequencing error rate). For each set of genome parameters (c,e,L) we have generated 50 data sets. Then we computed the exact histogram using Jellyfish software [4] and two approximate histograms using the original and the modified version of Kmerlight. Finally we ran CovEst on these three histograms using the model without repeats, and we collected three estimates of coverage $\hat{c}_j, \hat{c}_{ok}, \hat{c}_{mk}$. In all experimental results, we focus on the estimates of coverage $\hat{c}$. By dividing the sum of lengths of all reads by $\hat{c}$, we can obtain estimates of genome size $\hat{L}$.

In the first experiment (top row of Figure 7) 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.

Mean errors are consistently lower for modified Kmerlight than the errors of the original Kmerlight. 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 all errors bounded by 0.4%, 1%, 4%, 15% for sequencing error rates of $0, 0.01, 0.05, 0.1$ respectively, and we consider these estimates sufficiently accurate.

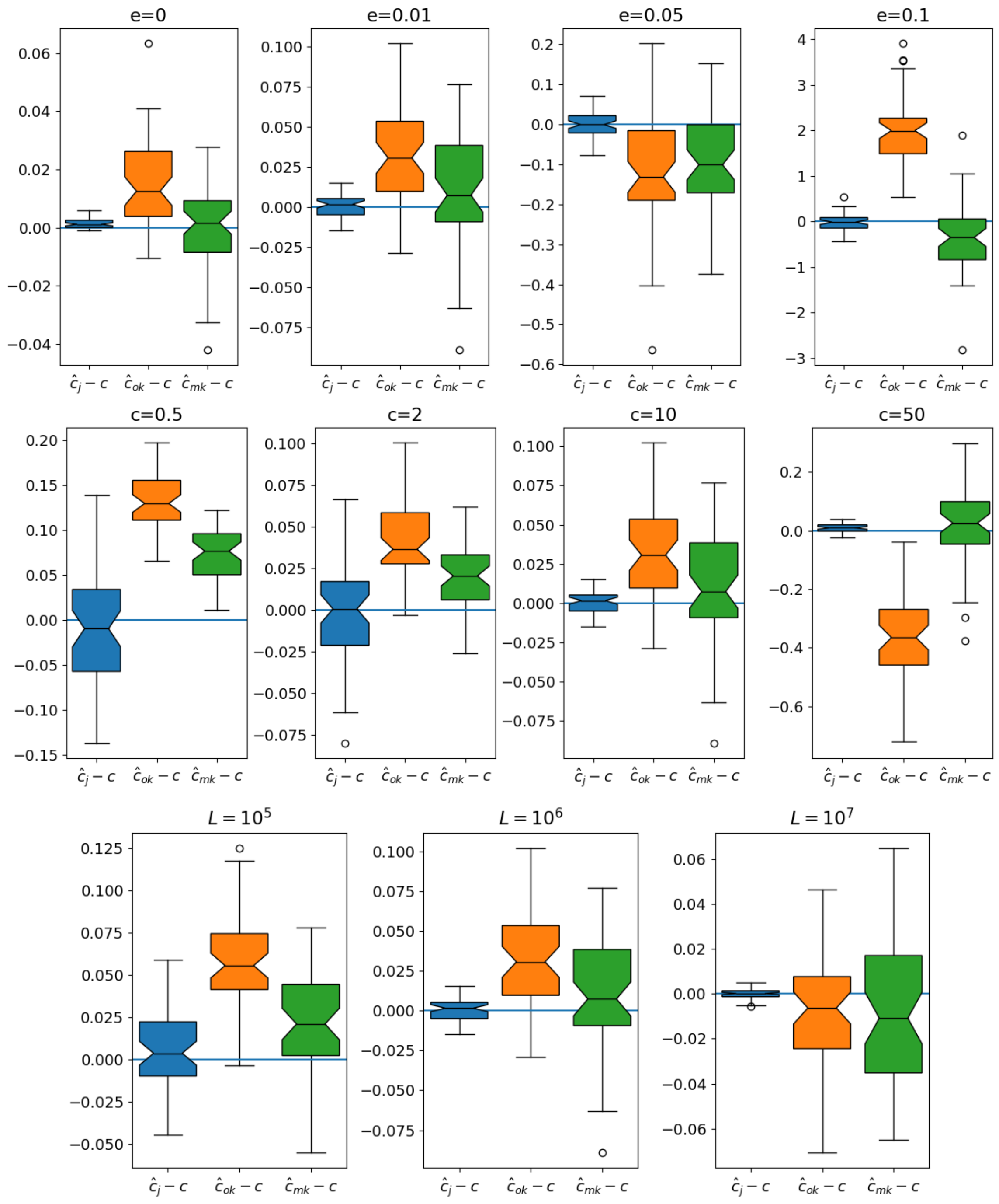


Figure 7: Distributions of coverage estimate errors $\hat{c} - c$ for different parameters. Each boxplot represents 50 estimates of the coverage on different generated genomes. Blue, orange and green boxes describe the estimate errors with Jellyfish, original Kmerlight and modified Kmerlight used to compute the k-mer abundance histogram. Note that vertical axes use different scales since the values span across multiple orders of magnitude in different datasets. As the default parameters, we use $L = 10^6$, $c = 10$ and $e = 0.01$. In each experiment, we altered the value of one parameter, keeping the other two fixed. Top row: different error rates, middle row: different coverage values, bottom row: different genome sizes.

In the second experiment (middle row of Figure 7), we study the accuracy for different genome coverage values. 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 outperforms the original Kmerlight on all four presented datasets (verified by t-tests).

Finally, with increasing genome size (bottom row of Figure 7), CovEst's estimates become more precise on the 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 the two smaller genomes than the original Kmerlight.

4 Conclusion

In this paper, we have studied the character of approximation errors in the k-mer approximation histograms produced by Kmerlight [8]. We have discovered that Kmerlight's estimates of $\hat{f}_i$ are biased towards higher values and provided a new estimate without this bias. We have demonstrated that for sufficiently large values of f_i, the distribution of estimates can be well approximated by a normal distribution. This approximation can be used to tune the parameters of the method.

We have also demonstrated that approximate histograms produced by Kmerlight are sufficiently accurate to produce reasonable estimates of genome size, and that for most settings our modified version of Kmerlight leads to more accurate genome size estimation than the original Kmerlight. An important avenue for further research is to compare the accuracy CovEst results on real sequencing data.

The use of approximate histograms allows significant reduction in memory; for example for genome size $L = 10^8$, coverage $c = 50$ and error rate $e = 0.1$ ($F_0 = 3.6 \cdot 10^9$), Kmerlight can produce an approximate histogram in only 83MB of memory, whereas Jellyfish needs 34Gb.

Note that our modified Kmerlight uses only one of 64 Kmerlight's levels to compute estimates for all f_i, so it naively seems that we could maintain only this level and reduce the memory consumption by a factor of 64. However, our version still uses the full 64 levels to compute $\hat{F}_0$, which is needed to find the desired level w^+. We could try to estimate $\hat{F}_0$ separately, particularly, if two passes over the data are allowed. For example, we could roughly guess F_0 from the size of sequencing data and use fewer levels. It might be also sufficient to estimate F_0 with smaller hash tables. We did not inquire deeper into this topic, but we believe that by such techniques the memory consumption could be further decreased by a significant factor.

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 goes beyond the field of bioinformatics.

Acknowledgments. This research was funded by VEGA grants 1/0684/16 (BB) and 1/0719/14 (TV), and a grant from the Slovak Research and Development Agency APVV-14-0253.

References

[1] Z. Bar-Yossef, T.S. Jayram, R. Kumar, D. Sivakumar, and L. Trevisan. Counting distinct elements in a data stream. In *International Workshop on Randomization and Approximation Techniques (RANDOM)*, pages 1–10. Springer, 2002.

[2] M. Hozza, T. Vinař, and B. Brejová. How Big is That Genome? Estimating Genome Size and Coverage from k-mer Abundance Spectra. In *String Processing and Information Retrieval (SPIRE)*, pages 199–209. Springer, 2015.

[3] S. Kurtz, A. Narechania, J. C. Stein, and D. Ware. A new method to compute k-mer frequencies and its application to annotate large repetitive plant genomes. *BMC Genomics*, 9(1):517, 2008.

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

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

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

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

[8] N. Sivadasan, R. Srinivasan, and K. Goyal. Kmerlight: fast and accurate k-mer abundance estimation. *CoRR*, abs/1609.05626, 2016.

[9] D. Williams, W. L. Trimble, M. Shilts, F. Meyer, and H. Ochman. Rapid quantification of sequence repeats to resolve the size, structure and contents of bacterial genomes. *BMC Genomics*, 14(1):537, 2013.