Fleximer: Accurate Quantification of RNA-Seq via Variable-Length k-mers

Chelsea J.-T. Ju
Department of Computer Science
University of California, Los Angeles
chelseaju@ucla.edu

Ruirui Li
Department of Computer Science
University of California, Los Angeles
rrli@cs.ucla.edu

Zhengliang Wu
Department of Computer Science
University of California, Los Angeles
lanzhou522@gmail.com

Jyun-Yu Jiang
Department of Computer Science
University of California, Los Angeles
jyunyu.jiang@gmail.com

Zhao Yang
Department of Computer Science
University of California, Los Angeles
yz931129@gmail.com

Wei Wang
Department of Computer Science
University of California, Los Angeles
weiwang@cs.ucla.edu

ABSTRACT

The advent of RNA-Seq has made it possible to quantify transcript expression on a large scale simultaneously. This technology generates small fragments of each transcript sequence, known as sequencing reads. As the first step of data analysis towards expression quantification, most of the existing methods align these reads to a reference genome or transcriptome to establish their origins. However, read alignment is computationally costly. Recently, a series of methods have been proposed to perform a lightweight quantification analysis in an alignment-free manner. These methods utilize the notion of *k-mers*, which are short consecutive sequences representing the signatures of each transcript, to estimate the relative abundance from RNA-Seq reads. Current *k-mer* based approaches make use of a set of fixed size *k-mers*; however, the true signatures of each transcript may not exist in a fixed size. In this paper, we demonstrate the importance of *k-mers* selection in transcript abundance estimation. We propose a novel method, Fleximer, to efficiently discover and select an optimal set of *k-mers* with flexible lengths. Using both simulated and real datasets, we show that, with fewer *k-mers*, Fleximer is able to cover the similar amount of reads as Sailfish and Kallisto. The selected *k-mers* own more distinguishing features, and thus substantially reduce the errors in transcript abundance estimation.

CCS CONCEPTS

• **Applied computing → Molecular sequence analysis**; **Computational transcriptomics**; **Transcriptomics**; *Computational genomics*; *Bioinformatics*;

KEYWORDS

RNA-Seq; Suffix tree; Aho-Corasick algorithm; EM algorithm; Transcript quantification

ACM-BCB '17, August 20-23, 2017, Boston, MA, USA
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-4722-8/17/08…$15.00
https://doi.org/10.1145/3107411.3107444

1 INTRODUCTION

Gene expression profiling serves the means to study transcriptome. The knowledge acquired from expression studies aims to create a global picture of cellular function. In the past two decades, microarray and high-throughput sequencing are the popular technologies, providing a rapid measurement to quantify a large number of genes simultaneously. The sequencing-based technology that captures the snapshot of existing RNAs is referred to as RNA-Seq.

The advent of RNA-Seq poses substantial computational challenges, specifically in handling the massive amount of read data [27]. The traditional data analysis framework contains two components: read alignment and expression quantification. Millions of short reads generated by the sequencer are first aligned back to a reference genome or transcriptome to re-establish their origins. Several aligners include, but are not limit to, Tophat2 [13], MapSplice [26], and SpliceMap [2]. However, this alignment step requires a significant amount of computational resources [8]. Even with parallel computing, analyzing a reasonable size of RNA-seq experiment can still take hours [19]. Recently, alignment-free or pseudoalignment approaches have been developed to alleviate the computational burden [4, 19, 23, 28]. Instead of aligning reads back to their references, these approaches design k-mers to infer transcript abundances from read data. They have all demonstrated a rapid analysis, improving the running time from hours to minutes. The k-mer approach harbors the features and advantages of both microarray and RNA-Seq. A set of k-mers behaves similarly to the probes in a microarray chip, which is used to identify the corresponding sequences in cDNA sample pool. Unlike the measurement in a microarray, the amount of cDNA identified by these k-mers is unbounded in RNA-Seq. In addition, since the k-mer design is performed *in-silico*, it is easier to re-construct a different set of k-mers for new analyses than re-designing a microarray chip.

Sailfish [19] is the first implementation to use k-mers for transcript quantification. It indexes all k-mers that appear at least once in the transcriptome and counts their occurrences in RNA-Seq data. Based on these counts, it estimates the relative transcript abundance through the expectation-maximization (EM) algorithm. Further ameliorating the throughput, RNA-Skim [28] divides the transcriptome into clusters based on sequence similarity. For each cluster, it selects a set of k-mers that do not appear in any other clusters, denoting as *sig-mers* (signatures of a cluster). Since each cluster contains a unique set of k-mers, the quantification step can be performed independently for smaller groups of transcripts. Zhang and Wang [28] have shown that RNA-Skim requires less computational time to achieve the same accuracy. Kallisto [4] builds targeted de-Bruijn graphs on all k-mers to facilitate the quantification, and claims faster and more accurate performance than Sailfish. All of these methods use a fixed k, which is a predefined parameter. However, selecting the appropriate k is not intuitive for a new experiment, and the best k characterizing a transcript sequence varies for different transcripts. For these reasons, we focus on the discussion of how to select a set of k-mers that can best describe transcript sequences with flexible lengths.

A k-mer cannot be too short because it can randomly match to a read that comes from a different transcript than the k-mer origins. Shorter k-mers have a higher chance to be repeated in the genome. Moreover, this event also arises when a read contains sequencing errors or individual variations. The accidental mismatch scenario is similar to the "cross-hybridization" phenomenon in microarray [14]. On the other hand, since the quantification step requires an exact match between a k-mer and a segment of a read, a long k-mer is less robust to reads containing sequencing errors or individual variations. An optimal set of k-mers should be robust to sequencing errors and individual variations, and should also be able to describe the uniqueness of each transcript. Sailfish originally set the default value of k to be 20, but changed it to 31 according to its website [1]. The initial choice is based on the evaluations over a small range of k (15-25) without further optimization. RNA-Skim studies the overall accuracy of transcript abundances for different ks before setting it to 60 as the default option. Neither of them has provided any guidance in selecting the right k for a new experiment or species.

To overcome the limitation of a fixed k, our main contribution focuses on developing an efficient approach to identify and select an optimal set of k-mers with flexible lengths. Our method takes a predefined genomic or transcriptomic partition as an input. The partition can be based on sequence similarity, biological functions, or any other user-defined scheme. In order to discover k-mers that are unique to each cluster with all possible lengths, we rely on the structure and properties of a suffix tree. We use the Aho-Corasick algorithm to efficiently match the reads to a set of representative k-mers of variable sizes. Based on the read counts, we use the EM algorithm to estimate the relative expression abundance for each transcript. We evaluate the performance of our method, Fleximer, using both synthetic datasets and real RNA-Seq data. Results show that Fleximer is able to minimize the prediction errors, providing a more accurate transcript abundance estimation compare to existing k-mer based methods. Fleximer is implemented in C++11 and available at https://github.com/chelseaju/Fleximer.git.

2 METHODS

2.1 Fleximer Overview

The lightweight algorithms for RNA-Seq quantification start with identifying a set of *k-mers* in a transcriptome, followed by either counting the occurrence of *k-mers* in RNA-Seq reads, or directly counting the reads that contain these *k-mers*. Since we know the transcript origins of these *k-mers*, transcript abundance can be inferred from read counts or *k-mers* counts. We first partition a transcriptome Θ into a set of non-overlapping clusters θ_1, θ_2, .., θ_n based on sequence similarity, and select a special type of *k-mers*, named *sig-mers*, for each cluster. These *sig-mers* represent the discriminative short sequences of a cluster of transcripts and do not appear in any other clusters. Adopting the terminology introduced by RNA-Skim, a set of *sig-mers s* in cluster θ_i is defined as

$$\Omega(\theta_i) = \{s \mid s \in (k\text{-mers in } \theta_i), \forall \theta_j \in \Theta \setminus \theta_i, s \notin (k\text{-mers in } \theta_j)\}$$

The size of *k-mers* can have a predominant effect toward the accuracy of transcript abundance estimation. We conducted an experiment to examine the effects of different ks. Experimental details are described in Section 4.1. We compared the estimated abundance to true abundance using 14 sets of *sig-mers* from RNA-Skim, and six sets of *k-mers* from Kallisto, with different values of k. We observed that the best prediction for each transcript falls in different ks. Additionally, the accuracy varied largely among different sets of *sig-mers* for some genes. Motivated by these observations, our method aims to efficiently select a set of *sig-mers* with flexible lengths, and to accurately quantify transcript abundance using this set of *sig-mers* in RNA-Seq data. We divide our objective into four components, with the first two focusing on *sig-mer* generation and the last two addressing transcript quantification. Figure 1 shows an overview of our approach. We follow the partition scheme in RNA-Skim [28] to generate a set of non-overlapping clusters of transcripts based on sequence similarity. Given this set of clusters, the transcript sequences are inspected using a suffix tree data structure to identify the *sig-mers* of each cluster. We select a subset of *sig-mers* that are robust to read errors, and can best describe the signatures of each transcript. We then use the Aho-Corasick algorithm to efficiently determine the occurrence of *sig-mers* in each read. The presence of these *sig-mers* provides the information of transcripts where each read is potentially originated from. The counts of these potential transcripts are then distributed by the EM algorithm for expression quantification. In this section, we describe each component of Fleximer in detail, including a brief review of the suffix tree.

2.2 Background on Suffix Tree

Let t' be a string over the alphabet $\Sigma = \{A, C, G, T\}$, and "\$" be a unique character, such that $\$ \notin \Sigma$. This unique character is appended to the end of the string t' to guarantee that every suffix ends at a leaf in the tree. Therefore, we use the following notations: t denotes the entire string with a terminator ($t' + \$$), and $|t|$ denotes the size of t. The string t can be degenerated into $|t|$ suffixes, and stored into a compressed tree structure, known as the suffix tree. This suffix tree contains exactly $|t|$ leaves, which are labeled by the starting position of the corresponding suffix. Each internal node has at least two children. The label of an internal node is omitted for memory efficiency. Each edge is labeled with a non-empty substring of t; edges connecting from the same parent contains different starting characters. The edge labels can be concatenated through a path in the tree, forming a longer substring of t. Thus, any individual suffix of t can be recreated by traversing along a path from the root to a leaf and concatenating the labels of the edges along the way.

Multiple sequences can be stored in the same tree, forming a generalized suffix tree (GST) [3]. A GST can be constructed by appending different unique symbols at the end of each sequence to ensure no suffix is a substring of another. Alternatively, we can

[1] http://www.cs.cmu.edu/~ckingsf/software/sailfish/faq.html

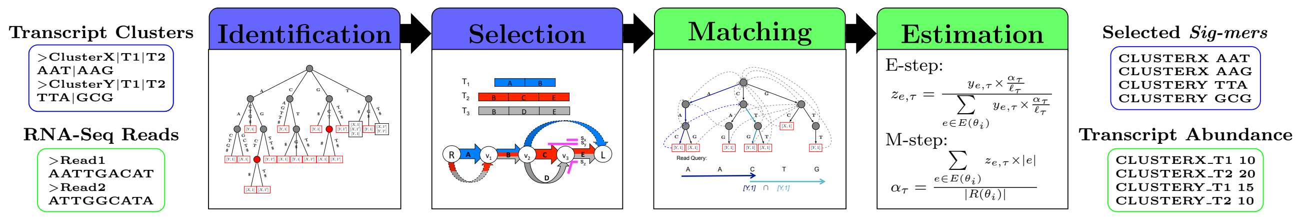


Figure 1: Fleximer Overview. Data input contains a predefined partition of a transcriptome, and RNA-Seq reads. The partition is stored in a specialized fasta format, where each entry represents one cluster. The header starts with a cluster name, follow by transcript names. The sequence field contains all transcript sequences in a cluster, separated by a special character "|". In *sig-mer* identification, we construct a suffix tree based on forward and reverse complementary sequences of all transcripts. We apply a post-order traversal on the suffix tree to identify all *sig-mers* with different lengths. We filter these *sig-mers* to keep only the optimal ones and use them to estimate the transcript expression in RNA-Seq data. In the matching step, we use the Aho-Corasick algorithm to determine the occurrence of *sig-mers* in each read. We then apply the EM algorithm to resolve the count of reads that are shared by multiple transcripts. The blue boxes denote the preparation stage for *sig-mer* generation, which only needs to be executed once. The green boxes denote the analysis stage, which applies to each RNA-Seq experiment.

concatenate these sequences into one long string, separated by a single terminator, and build a regular suffix tree for the resulting string. Regardless of using a single or multiple terminators, each leaf node stores sufficient information to retrieve the starting positions and sequence of origin for each suffix.

The construction of a suffix tree takes linear time and space respect to the total sequence size [6, 16, 24]. The most memory efficient implementation to handle genomic sequences is the Sadakane's compressed suffix tree [21, 25], which relies on several abstract data structures to reduce the memory footprint. It also preserves all the operations for a normal suffix tree. One of the abstract data structures is balanced parentheses [17], which represents the tree in pre-order. This structure can be further modified to retrieve node information in post-order. For these reasons, we implement our method using Sadakane's compressed suffix tree, which is available from Succinct Data Structure Library (SDSL) [2].

2.3 *Sig-mers* Identification in Fleximer

Given a set of n transcripts in a cluster θ_i, our first task is to identify the *sig-mers* that can characterize the uniqueness of a cluster. Suppose the average length of transcripts is m, the complexity of retrieving and examining all *k-mers* with length less than k is bounded by $O(nkm)$ (Supplementary A). In the human genome, there are over 20,000 protein coding genes, corresponding to more than 198,000 transcripts. The average length of a transcript is approximately 2,000 bp. Therefore, enumerating all possible substrings to check for their uniqueness is computationally intractable. The suffix tree is a powerful data structure for string searching algorithm, and it has been widely applied to a diverse range of biological sequence analyses [10, 11, 15]. Leveraging the suffix tree structure and properties, we apply a post-order traversal to identify all unique substrings.

We build a suffix tree based on all prefixes and suffixes of transcript sequences. We use both forward and reverse complementary sequences to construct the tree. There are two practical aspects to include the reverse complementary sequences. First, reads from the sequencing technology can be generated from the complementary strand. Second, suffixes of a reverse complementary sequence serve as the same mean as the prefixes of its forward sequence. If a substring is a *sig-mer*, then its reverse complementary sequence is also a *sig-mer*. In a suffix tree, if a substring from the root to any node appears only once in our transcript sequences, this path contains at least one *sig-mer*. We refer this node as a candidate. An internal node typically represents a substring that appears more than once in our transcript sequences. The path to any internal node contains a *sig-mer* only if all of the starting positions of this substring are located in the same cluster. These starting positions can be retrieved recursively from its descendants. Specifically, we use a bottom-up approach to examine each node through a post-order traversal.

The post-order traversal starts with the left subtree, followed by the right subtree, and the parent. We store the substring information of the nodes that have already been visited. This information does not include the actual sequence, but is sufficient to retrieve the cluster and sequence IDs, and the starting positions of a substring. At each internal node, we determine if it is a candidate by examining all substrings from its children. If all substrings come from the same cluster, then it is a candidate. Otherwise, the search process terminates for this branch, marking this node and its ancestors disqualified. Since the cluster information of a substring propagates from the leaves, all the ancestors of a disqualified node can be pruned. We use this anti-monotonic constraint to avoid further computation of its ancestors. Figure 2 uses a toy example to illustrate this idea to discover all *sig-mers*.

A substring from the root to a candidate node represents the longest *sig-mer* at this candidate. On the other hand, a substring from the root to the first character on the edge between the candidate node and its parent represents the shortest *sig-mer* for this candidate. In order to obtain *sig-mers* with all possible sizes, we need to append all prefixes of the edge label between the candidate and its parent to the path from the root to its parent. An example is illustrated in Figure 2.

2.4 *Sig-mers* Selection in Fleximer

The compressed suffix tree allows us to efficiently discover all *sig-mers* with different sizes. However, not all of them are necessary for the quantification stage. We select a set of representative *sig-mers* that possess three properties: 1) Be robust to sequencing errors and

[2]https://github.com/simongog/sdsl-lite/

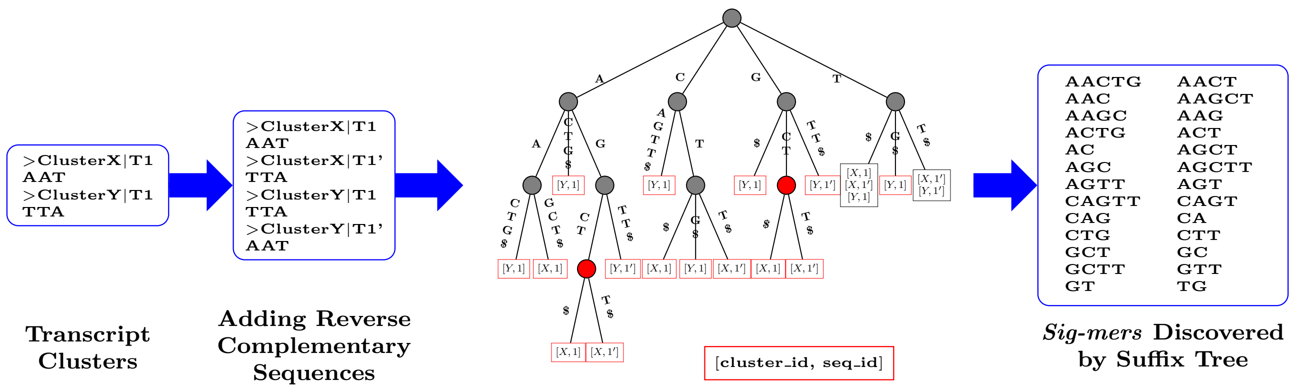
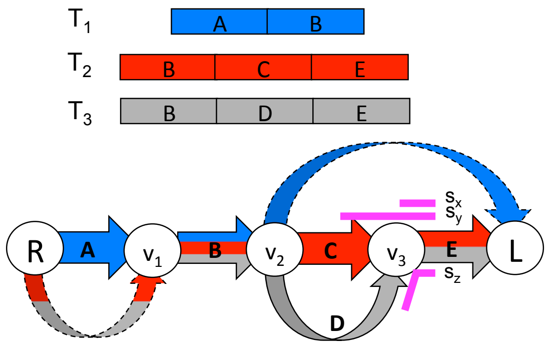
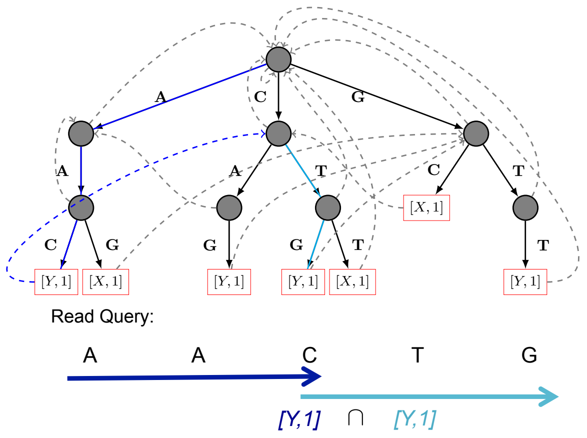


Figure 2: *Sig-mers* Identification. We use both forward and reverse complementary sequences of each transcript to construct a generalized suffix tree. Each leaf contains cluster and sequence IDs of a suffix. Each edge is labeled by a substring. The cluster and sequence IDs of an internal node are retrieved from its descendants through a post-order traversal. Candidates are highlighted in red. For example, substring *AGCT$* and *AGCTT$* appear uniquely in sequence 1 and 1' (reverse complementary of sequence 1) of cluster X, respectively. Thus, their nodes are candidates. Their parent is also a candidate since *AGCT* only appears in cluster X but not in cluster Y. However, their grandparent is not a candidate as *AG* appears in both cluster X and Y. The traversal process terminates here for this subtree. In order to generate all *sig-mers* with different sizes, we consider all prefixes of an edge label between a candidate and its parent. Therefore, *AGC* is also a *sig-mer* in addition to *AGCT*.



(a) *Sig-mer* Selection



(b) Read Matching

Figure 3: a) A cluster contains three transcripts, T_1, T_2, and T_3. The upper figure shows the composition of these transcripts. Each box represents either a unique or shared sequence segment. For example, segment B is shared by all transcripts, where segment A is unique to transcript T_1. If they are isoforms of the same gene, segment B can be interpreted as a common exon. Since a partition may include transcripts from more than one gene, we use the term 'segment' instead of the exon. A compact splicing graph can be constructed for all member transcripts in a cluster, as demonstrated in the lower figure. Each edge represents a segment, and each vertex indicates a transition point. Two virtual edges are included as dotted arrows from R to v_1 and from v_2 to L. In this graph, sequences of these three transcripts are depicted by three paths (blue, red, and gray) from R to L. *Sig-mers* can be viewed as a substring of an edge, denoted by the pink lines. *Sig-mers* span through a vertex are more powerful to determine the origin of a read, and thus are prioritized for selection. b) We use an automaton to facilitate the matching step between reads and a set of representative *sig-mers*. Each node consists a failure link to guide the search. Given a query "AACTG", we follow the path AAC (dark blue) and find the substring belongs to sequence 1 in cluster Y. Following the failure link, we can quickly find CTG (light blue), which also belongs to sequence 1 in cluster Y.

individual variations. 2) Characterize the uniqueness of each transcript. 3) Provide sufficient coverage across all transcript sequences. We will first discuss the optimal range of k for robustness.

In an ideal scenario where there are no sequencing errors and individual variations, *sig-mers* will match all reads that come from the same cluster. In reality, reads contain sequencing errors and individual variations. These reads are less likely to be recognized by long *sig-mers* since the matching process requires an exact match. On the other hand, these reads have a higher risk to be "matched" by short *sig-mers* that belong to other clusters. Therefore, a *sig-mer* cannot be too long or too short, with the optimal range depending on the sequencing error rate and the fraction of individual variations. In addition, the upper bound of k is constrained by the read length of an RNA-Seq experiment. Based on these considerations, we set the lower bound to 25bp, and the upper bound to 80% of a read length. *Sig-mers* are filtered based on their sizes.

After filtering, we use the concept of a splicing graph [9, 22] to guide the selection. Splicing graph is first introduced to predict and represent the choices of alternative splicing for a gene model [18]. It is a directed cyclic graph, in which vertices represent the splicing sites and edges are the exons or introns between two splicing sites. Two virtual vertices are added to the graph, root (R) and leaf (L), along with virtual edges, so that each transcript can be represented by a path that goes from R to L. Vertices with their indegree and outdegree equal to 1 are uninformative for delimiting alternative splicing event, and are often collapsed to obtain a more compact representation. Analogously, we can use a compact splicing graph to represent all member transcripts in a cluster. A cluster may contain transcripts from more than one gene, depending on the partitioning scheme. To incorporate this scenario, we broaden the definition of the edges and vertices. Edges represent different sequence segments that appear at least once in a cluster, and vertices represent transition points where the incoming edge covers a different set of transcripts from the outgoing edge. Figure 3a illustrates the compact splicing graph of a cluster with three member transcripts. A *sig-mers* can be viewed as a substring of an edge. As we observe from the graph, a *sig-mer* possesses higher discriminative power if it spans through a vertex. Considering three *sig-mers* s_x, s_y, and s_z in Figure 3a, s_x resides completely on segment E and is shared by two transcripts. Given a read that contains s_x, the origin of this read can be either one of these transcripts. On the other hand, s_y and s_z span through vertex v_3, and are unique to transcript T_2 and T_3, respectively. Since s_y and s_z are superstrings of s_x, reads containing s_y and s_z also contain s_x. However, s_y and s_z provide more specific information regarding the origin of these reads. Therefore, the selection process prioritizes *sig-mers* that span through a vertex.

The original proposal of splicing graph is constructed through ESTs (expressed sequence tags) or RNA-Seq reads assembly. Since we already know the starting positions of all *sig-mers*, we do not need to assemble these *sig-mers*. Instead, we order the *sig-mers* based on their starting positions. As we traverse through all starting positions in a sequential order, we pave all paths of a splicing graph. To provide a sufficient coverage for each transcript, we select additional *sig-mers* along the path, spaced by a small gap (e.g., 5bp) between two *sig-mers*. Since RNA-Seq reads may originate from transcripts in either forward or reverse direction, complementary sequences of the selected *sig-mers* are added to the final list.

2.5 RNA-Seq Reads Matching

The quantification stage starts with determining the potential transcript origins of each RNA-Seq read. We refer this set of potential transcripts as the "transcript profile" of a read. In the traditional framework, the transcript profile of a read is determined by aligning the read sequence to the reference transcriptome. In our setting, each *sig-mer* is associated with a transcript profile indicating its origins. We can construct the transcript profile for each read by taking the intersection of all transcript profiles associated with the *sig-mers* that appear in a read.

Discovering the occurrence of our representative *sig-mers* in RNA-Seq reads is equivalent to the keyword searching problem in computer science. A trivial approach is to index the representative *sig-mers* (keywords) with a hash table. Since we allow variable *sig-mer* sizes, the challenge lies on scanning through each read with different window sizes. Even with an efficient hashing function, such as rolling hash [12], the number of comparisons increases with the variety of window sizes. Assume that the read length is ℓ, and total length of *sig-mers* is M. The sizes of these *sig-mers* range from k_1 to k_d. The time complexity for constructing the hash is $O(M)$, and for searching *sig-mers* in one read is $O(\ell \times k_1) + \ldots + O(\ell \times k_d)$. The total time complexity would be $O(M + \ell \times \frac{k_1+k_d}{2} \times d)$.

A linear search solution is the Aho-Corasick algorithm [1], which constructs a finite state automaton for all *sig-mers*. This automaton is a keyword tree with additional links between internal nodes. These extra links allow fast transition between *sig-mer* matches without the need for backtracking. The complexity of building the automaton is $O(M)$ and for searching is $O(\ell + z)$ where z refers to the total number of occurrences of *sig-mers* in a read. Figure 3b illustrates the search using the Aho-Corasick algorithm.

2.6 Transcript Abundance Estimation

Since the transcriptome is partitioned into a set of non-overlapping cluster, each cluster can be quantified independently. After the *sig-mer* matching step, each read is associated with a transcript profile indicating its potential origins. Based on these profiles, we can group reads into their corresponding clusters. For each cluster θ_i, we use $\mathrm{R}(\theta_i)$ to represent the set of reads assigned to cluster θ_i, and $\mathrm{T}(\theta_i)$ to represent the set of transcripts in θ_i. Each transcript $\tau \in \mathrm{T}(\theta_i)$ is associated with a probability α_τ indicating its proportion of reads in cluster θ_i, and $\sum \alpha_\tau = 1$. If we know the exact origin of each read, we can form an indicator matrix Z, where $z_{r,\tau} = 1$ indicating that read $r \in \mathrm{R}(\theta_i)$ comes from transcript τ, and 0 otherwise. Since each read only comes from one transcript, α_τ can be estimated by $\sum z_{r,\tau} \ / \ |\mathrm{R}(\theta_i)|$, where $|\mathrm{R}(\theta_i)|$ is the total number of reads in θ_i. However, Z is not fully observed, and what we observed is the transcript profiles for each read, forming another indicator matrix Y. Similarly, $y_{r,\tau} = 1$ if τ is in the transcript profile of read r. Z is the hidden variable, and can be recovered from the observation Y. Therefore, we use the following likelihood function to estimate α:

$$\mathcal{L}(\alpha|Y) = \prod_{r\in\mathrm{R}(\theta_i)} \sum_{\tau\in\mathrm{T}(\theta_i)} y_{r,\tau} \frac{\alpha_\tau}{\ell_\tau} = \prod_{e\in\mathrm{E}(\theta_i)} \left(\sum_{\tau\in\mathrm{T}(\theta_i)} \frac{\alpha_\tau}{\ell_\tau} \right)^{|e|}$$

In this log-likelihood function, ℓ_τ is the effective length for transcript τ. We can improve the computation speed and memory by grouping reads with the same transcript profile into equivalence classes. The concept of the equivalence class is widely used in transcript abundance estimation [4, 19, 28]. We use $\mathrm{E}(\theta_i)$ to represent the set of equivalence classes in θ_i, and equivalence class $e \in \mathrm{E}(\theta_i)$ contains $|e|$ reads. We use the EM algorithm to compute the maximum likelihood estimates of α from our observed data Y. The algorithm alternates between allocating the fraction of counts of each equivalence class (E-step), and estimating the relative abundance given this allocation (M-step). More specifically, the E-step reconstruct the hidden variable Z as $z_{e,\tau} = \frac{y_{e,\tau} \times \frac{\alpha_\tau}{\ell_\tau}}{\sum_{e \in E(\theta_i)} y_{e,\tau} \times \frac{\alpha_\tau}{\ell_\tau}}$, and the M-step updates the probability through Z, $\alpha_\tau = \frac{\sum_{e \in E(\theta_i)} z_{e,\tau} \times |e|}{|R(\theta_i)|}$.The algorithm converges when the change of α is less than 10^{-7}. At convergence, read count of τ can be recovered through α_τ and $|R(\theta_i)|$. We report transcript abundance in three commonly used metrics: raw read count, RPKM (Reads Per Kilobase of transcript per Million mapped reads), and TPM (Transcripts Per Kilobase Million). Both RPKM and TPM are normalized measurements that account for sequencing depth and transcript length.

3 DATA

3.1 Reference Transcriptome

We used the gene annotation from Ensembl release 81 of Human Genome GRCh38 [5], which contained more than 22,000 protein-coding genes. Among these, there are 19,817 protein-coding genes located on the canonical chromosomes (chr1-21, X, Y, Mt). Our experiments focus only on the transcripts encoded by these genes, which correspond to 143,609 protein-coding transcripts.

Sequences of these transcripts were decomposed into k-mers and indexed by Sailfish and Kallisto. For RNA-Skim and Fleximer, these transcripts were first partitioned into non-overlapping clusters. Transcripts with high sequence similarity were placed in the same cluster. This partitioning scheme was adopted from RNA-Skim, resulting in 16,149 clusters. Details of this process were described in [28]. We used the specialized fasta format suggested in RNA-Skim to store the sequences. The header of each entry contains a cluster name and transcript names, separated by a special character "|". Similarly, the body of an entry contains transcript sequences in a cluster, separated by "|". Parameter settings of each software are described in Supplementary B.

3.2 Simulated Dataset

We used *polyester* [7] to generate 15 sets of single-end RNA-Seq reads for selected transcripts to evaluate the accuracy of different methods. We randomly selected 2-10 % of the protein-coding transcripts to be expressed in each sample. Each sample contained 3-97 millions of reads, with a read length of 75bp. Three different read depths were used: 10X, 20X, and 30X. We selected an empirical sequencing error model from *polyester* to simulate the protocol of Illumina Sequencing Kit v4. Each read was tagged with the name of the transcript where it originated. We computed the expected number of reads for each transcript by counting the transcript names in the raw read file. The expected RPKM and TPM were calculated from the expected read count.

3.3 Real Dataset

We included a public dataset provided by the Illumina Human BodyMap 2.0 Project (GSE30611). The samples contained individual and a mixture of 16 human tissues RNA, and were sequenced using Illumina HiSeq 2000 System. Among all these samples, we used six of the single-end data with 75bp in our evaluation: female brain, breast, and colon; male adrenal, lung, and liver. The Expression Atlas [20] provides gene expression information of highly-curated and quality-checked RNA-seq experiments, including all 16 human tissues of the Human BodyMap Project. We used the gene expression values reported in http://www.ebi.ac.uk/gxa/experiments/E-MTAB-513/ as our ground standard.

4 EXPERIMENTS AND RESULTS

4.1 Motivating Example

Current lightweight quantification methods use a fixed k; however, a set of fixed size *k-mers* may not be sufficient to capture the signature of each transcript. We evaluated the effect of different ks in abundance estimation for both Kallisto and RNA-Skim. Kallisto allows users to set k to be any odd integer up to 31; RNA-Skim does not have any restriction on the choice of k. For Kallisto, we indexed the transcriptome using 6 different ks. For RNA-Skim, we selected 14 sets of *sig-mers* with different ks. We used these *k-mers* to estimate the TPM of a simulated dataset containing 11,484 expressed transcripts. We computed the absolute differences between true and estimated TPM. A smaller value indicates a better estimation.

Among these sets of *k-mers*, we marked the k that gave the best prediction (smallest difference) for each transcript. Figure 4a shows the distribution of the best k. We also plotted three transcripts that displayed interesting estimation patterns over different ks in Figures 4b-d. The estimations fluctuate largely in RNA-Skim for these transcripts, indicating that this method is very sensitive to the choice of k. The variation is less severe in Kallisto; however, setting k to 21 produces the best prediction for ENST00000626009, but gives the worst prediction for both ENST00000265881 and ENST00000357370. These results indicate that the optimal k varies for different transcripts, and the selection of k has a substantial influence on the prediction accuracy.

4.2 *k-mers* Generation

Fleximer selects 6,299,554 *sig-mers* with sizes ranging from 25 to 60. Note that since the sequencing reads of our datasets were 75bp, we set the upper bound of k to 60. The distribution of the *sig-mers* size is illustrated in Figure 5a, where the frequencies are plotted in log-10 scale. The majority of the *sig-mers* reside on our lower bound; however, the rest of the *sig-mers* are equally distributed among this range. On the other hand, using the default parameters, RNA-Skim generates a set of 4,221,162 *sig-mers* with a fixed size of 60. Using the same fixed size of 31, Sailfish produces a set of 95,601,889 *k-mers* and Kallisto produces 94,945,357 *k-mers*. Figure 5b compares the number of *k-mers* generated in different methods. In terms of computational resources, Fleximer needs more time to identify a set of optimal *sig-mers* than other methods since the search space is much bigger. However, it is worth mentioning that this step only needs to be executed once. Sadakane's compressed suffix tree allows us to perform this step in less than 10G, which requires less memory than Sailfish and RNA-Skim.

4.3 Simulation Study

We evaluated the prediction accuracy using 15 simulated datasets. The accuracy metrics include Pearson correlation, rank correlation, and root-mean-square error (RMSE) (Supplementary C).

Figure 6 illustrates three metrics separately for different read depths. Each bar shows the mean and standard deviation over 5

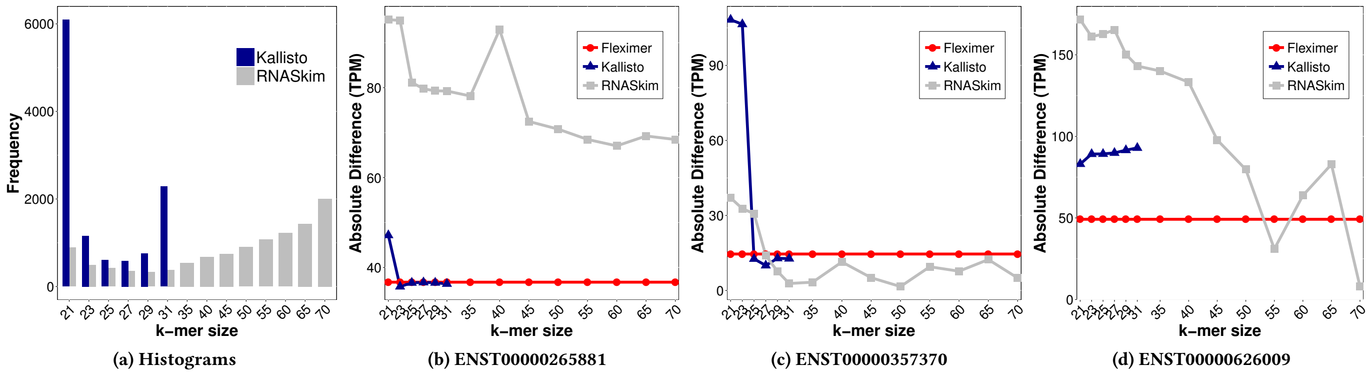


Figure 4: Evaluation of the TPM estimation over different sets of *k-mers*. a) The histogram shows the number of transcripts with the smallest error at different *k*s. b-d) Errors (absolute differences; smaller is better) at different *k*s for individual transcripts. Fleximer uses a wide range of *k*s, so the error is plotted as a single line.

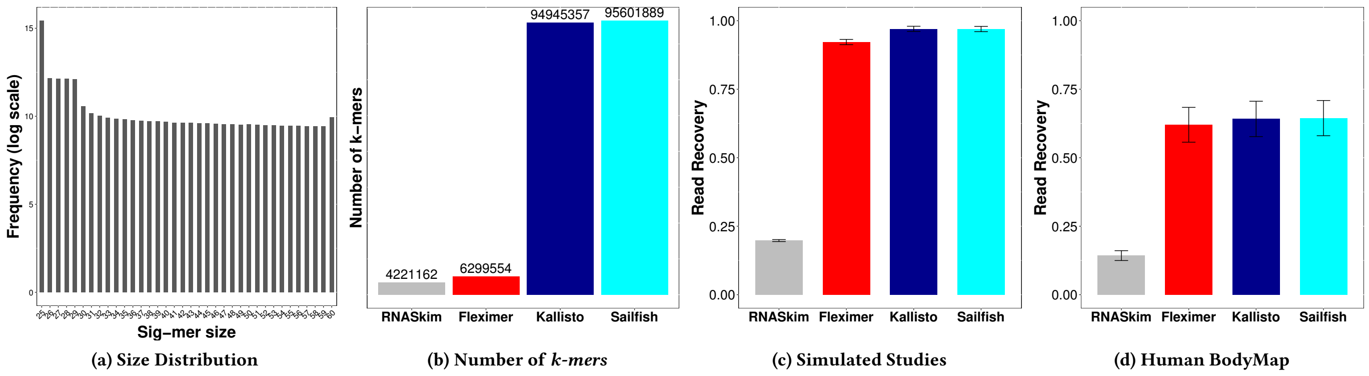


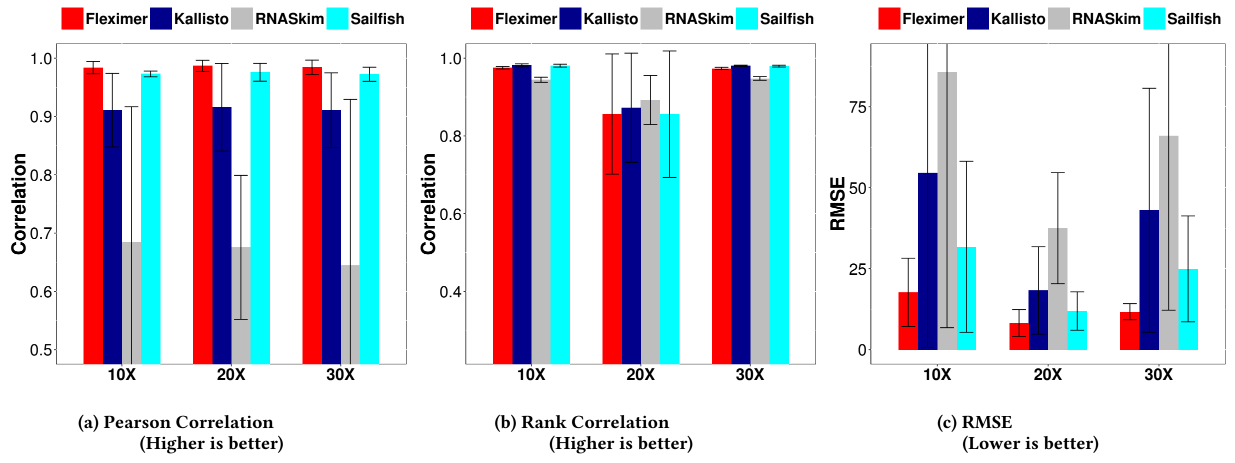
Figure 5: a) Distribution of *Sig-mers* sizes (25 to 60) in Fleximer b) Number of *k-mers* used in different methods for transcript abundance quantification. c-d) The average number of reads recovered by different sets of *k-mers* for simulated and real data, respectively. Error bars are included using one standard deviation.

datasets. Results indicate that Fleximer and Sailfish perform similarly with consistent Pearson and rank correlations. The Pearson correlations are slightly higher for Fleximer at different read depths. Higher correlation values are better as the predictions are in the same trend as the ground truth. On the other hand, lower residual errors are better as the predictions are closer to the ground truth. Fleximer presents the smallest RMSE among four methods. We also evaluated the running time as shown in Figure 7. Fleximer requires more time than others due to searching *sig-mers* of different sizes; however, it scales linearly with the number of reads. All experiments finish in less than 25 minutes (1500s). We further examined the utilization of our representative *sig-mers*. We calculated the fraction of reads recovered by each method. Figure 5c shows that Fleximer is able to cover more than 90% of the reads with fewer *k-mers* than Kallisto and Sailfish. The *sig-mers* Fleximer used are much more effective than the *sig-mer* set in RNA-Skim, which only recovers less than 30% of the reads.

4.4 Human BodyMap 2.0 Project

Simulated datasets could not capture all of the biases presented in a real RNA-Seq experiment; therefore, we evaluated the performance of different methods using six datasets from the Human BodyMap project. Figure 5d shows that on average, Fleximer is able to identify a similar amount of reads in these datasets as Sailfish and Kallisto ($> 60\%$), but with less amount of *k-mers*. This result is consistent as demonstrated in the simulated studies.

Since we did not know the true abundance of each transcript, we used the estimated abundance reported in Expression Atlas as our ground standard. Expression Atlas provided expression values in RPKM on gene-level (instead of transcript-level). Two post-processing techniques are applied to normalize these values. First, we converted expression values reported in Expression Atlas to TPM. Second, we aggregated the expressions of all transcript isoforms estimated in each method to represent the gene expression. After normalization, we removed the non-protein coding genes from Expression Atlas and conducted the analyses on gene-level. Figure 8 shows an overall comparison of the number of expressed genes identified by each method, and Table 1 summarizes the comparison of expressed genes between each method and the ground standard. Expressed genes are defined as those with TPM greater than zero. Across six datasets, all methods identify a large number of shared expressed genes as demonstrated by the large overlap in six Venn diagrams. Each method identifies a few amount of unique genes that are not picked up by others. Among the four methods, RNA-Skim misses out the largest number of expressed genes reported in Expression Atlas (false negative). However, Sailfish and Kallisto pick up around 100 more genes than Fleximer that are not reported in Expression Atlas (false positive). We also experimented



(a) Pearson Correlation (Higher is better)

(b) Rank Correlation (Higher is better)

(c) RMSE (Lower is better)

Figure 6: Accuracy Evaluations for Simulated Studies on Transcript-Level

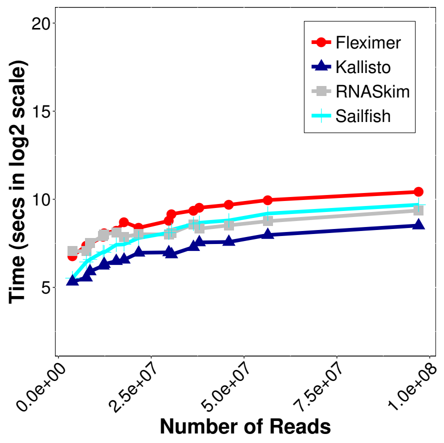


Figure 7: Running Time for Simulated Studies. Time is represented as seconds in the log-2 scale, where 10 represents to 1024 seconds (approximately 17 minutes).

with different thresholds in defining the expressed genes (i.e. 0.1, 0.5, and 1), and observed similar results as setting the TPM threshold to 0.

We further evaluated the prediction power of four different methods against the values reported in Expression Atlas using all genes. Figure 9 summarizes the average values of each accuracy metric over six datasets. Fleximer demonstrates the best accuracy, with the highest Pearson and rank correlations and the lowest RMSE.

5 CONCLUSION AND DISCUSSIONS

RNA-Seq has been the dominating technology in quantifying transcript abundance. The advent of this technology also comes with numerous computational challenges, especially in analyzing a large amount of data. Sailfish, RNA-Skim, and Kallisto have recently developed to aim for a rapid quantification using a pre-selected set of *k-mers*. These *k-mers* are of the same length, and present a limitation in characterizing different transcripts. We propose a novel method that allows selecting *k-mers* of flexible sizes.

Our approach leverages the properties and structure of a suffix tree to efficiently discover these *sig-mers*, and selects a subset of representative *sig-mers* for quantification. In order to determine the occurrences of these variable-length *sig-mers* in reads, Fleximer utilizes the Aho-Corasick algorithm to scan through reads in linear time. The final transcript abundance is estimated by the EM algorithm using the read counts. The results show that Fleximer is able to improve the accuracy of abundance estimations with fewer *sig-mers*. This observation is due to the fact that these variable-length *sig-mers* are capable of recognizing the signatures of each read in the RNA-Seq datasets. In addition, we only need a small set of *sig-mers* since they are unique enough to characterize different transcripts. In the future work, we would like to further improve the running time of our quantification step with parallel computing. We are also interested in exploring the possibilities of applying the idea of flexible *k-mers* to other sequencing applications.

ACKNOWLEDGMENTS

We thank Dr. Zhaojun Zhang and Dr. Farhad Hormozdiari for their comments and discussions, and Ms. Kim Vo for her proofreading. We would also like to thank the anonymous reviewers for their valuable comments and helpful suggestions. This work is supported by NIH R01GM115833, NSF DBI-1565137 and NSF IIS-1313606.

A RETRIEVING ALL *K-MERS*

Suppose the average length of a transcript is m, the complexity of retrieving and examining all *k-mers* with length less than and equal to k in one transcript is

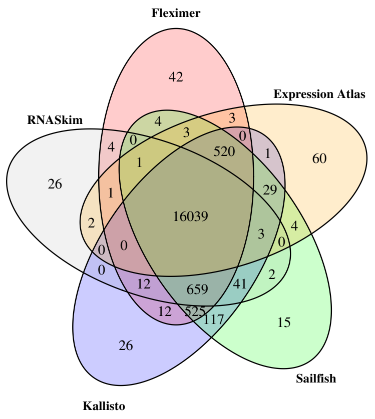
$$\begin{aligned}\text{retrieve k-mers} &= m + (m-1) + (m-2) + \cdots + (m-k+1)\\ &= km - (1 + 2 + \cdots + (k-1))\\ &= km - (k \times (k-1)/2)\\ &= O(km)\end{aligned}$$

Given n transcripts, retrieving all *k-mers* is bounded by $O(nkm)$.

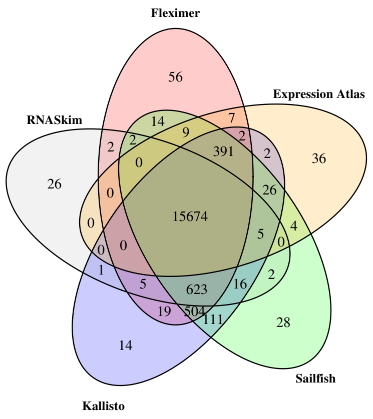
B SOFTWARE COMPARISON

B.1 Sailfish

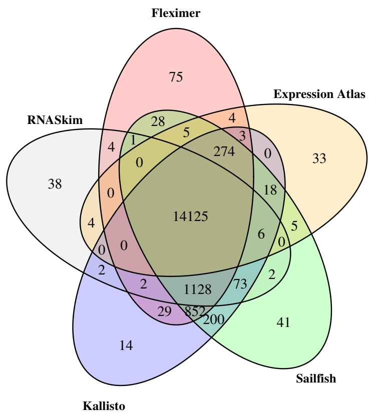
The binary version of Sailfish-0.9.2 was downloaded. We followed the default setting of *k-mer* size for the indexing step (`--kmerSize = 31`). To conduct a fair comparison of all software, we used a



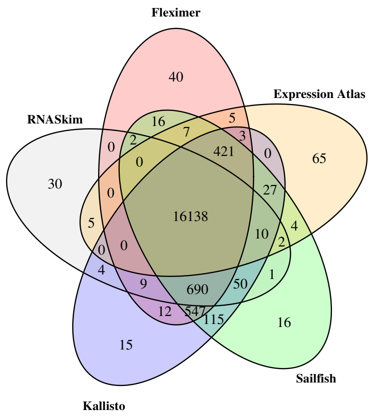
(a) Male Adrenal



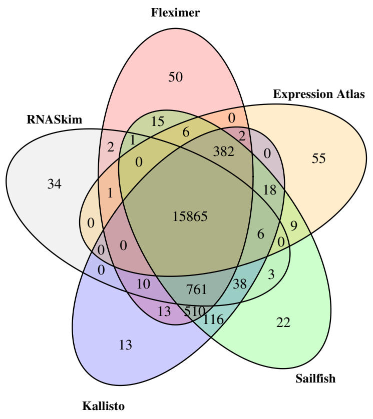
(b) Male Lung



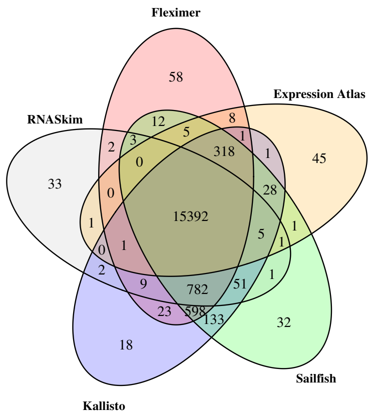
(c) Male Liver



(d) Female Brain



(e) Female Breast



(f) Female Colon

Figure 8: Evaluation of Human BodyMap. Venn diagrams showing the number of common and unique expressed genes identified by different methods.

Table 1: Comparison with Expression Atlas. True positives (TP) refer to expressed genes that are identified by each method and reported in Expression Atlas. False negatives (FN) refer to expressed genes that are reported in Expression Atlas but missed by each method. False positives (FP) refer to genes that are identified by each method but are not reported in Expression Atlas.

Method	Fleximer			RNA-Skim			Sailfish			Kallisto		
Evaluation	TP	FN	FP	TP	FN	FP	TP	FN	FP	TP	FN	FP
Male Adrenal	16567	99	1258	16046	620	744	16599	67	1363	16592	74	1392
Male Lung	16083	73	1225	15679	477	677	16109	47	1300	16100	56	1293
Male Liver	14411	66	2119	14135	342	1250	14433	44	2325	14426	51	2300
Female Brain	16574	113	1316	16155	532	786	16609	78	1437	16599	88	1442
Female Breast	16256	88	1362	15872	472	849	16286	58	1466	16273	71	1461
Female Colon	15725	82	1487	15400	407	883	15750	57	1612	15746	61	1616

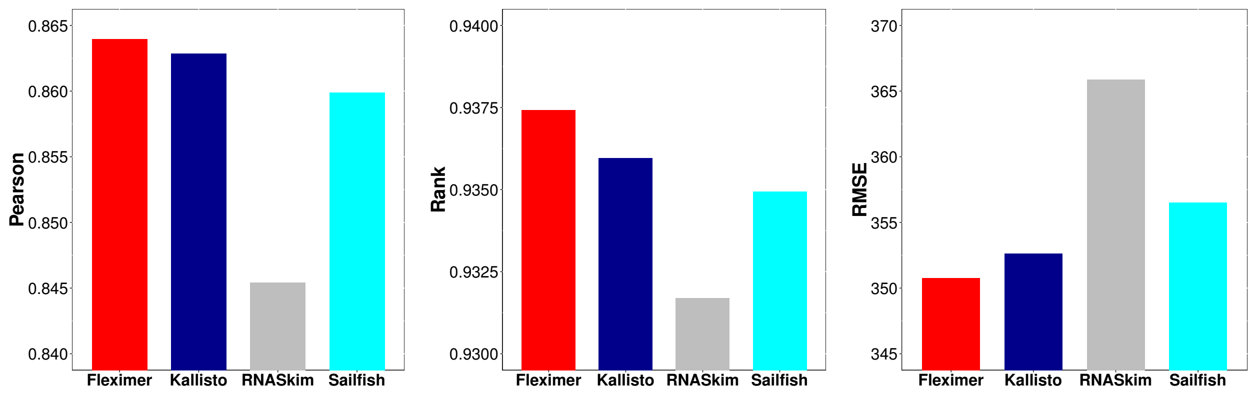
single thread for both of the indexing and quantification steps (`--threads = 1`). All of our datasets contained single-end reads with a read length of 75bp; therefore, we used these parameters in the quantification stage: `-l -U --fldMean 75 and --fldSD 1`.

B.2 Kallisto

The source code of Kallisto 0.43 was compiled with GCC/4.8.3. The suggested *kmer* size was the same as Sailfish. We set `--kmer-size=31` for indexing. For quantification, we used a single thread, `--threads=1`. To adjust for effective length, we used `--single -l 75 -s 1`.

B.3 RNA-Skim

The source code was compiled with GCC/4.7.2. We ran `rs_cluster` to generate the partition, with `-rs_length=60`. We ran `rs_index` and `rs_select` to generate a set of *sig-mers*. In the motivating example, `rs_length` was set to different values to evaluate the



(a) Pearson Correlation (Higher is better)

(b) Rank Correlation (Higher is better)

(c) RMSE (Lower is better)

Figure 9: Accuracy Evaluations for Real Datasets on Gene-Level

effects of different ks. In both simulated and real studies, `rs_length` was set to 60 as suggested by RNA-Skim. We ran `rs_count` and `rs_estimate` using the default settings. We set `-num_threads=1` to enforce the software running on one thread.

C ACCURACY METRICS

We used Pearson, rank correlations, and Root Mean Squared Error (RMSE) to evaluate the accuracy of four different methods. Both of the Pearson and rank correlations were computed between the estimated abundance (TPM) and the true TPM. Given a list of n transcripts with estimated TPM of $x_1, x_2, \ldots, x_n$ and their true TPMs $y_1, y_2, \ldots, y_n$, the RMSE was calculated as $\sqrt{\frac{\sum_{i=1}^{n}(x_i - y_i)^2}{n}}$.

REFERENCES

[1] Alfred V. Aho and Margaret J. Corasick. 1975. Efficient string matching: an aid to bibliographic search. *Commun. ACM* 18, 6 (jun 1975), 333–340. https://doi.org/10.1145/360825.360855

[2] Kin Fai Au, Hui Jiang, Lan Lin, Yi Xing, and Wing Hung Wong. 2010. Detection of splice junctions from paired-end RNA-seq data by SpliceMap. *Nucleic acids research* 38, 14 (aug 2010), 4570–8. https://doi.org/10.1093/nar/gkq211

[3] Paul Bieganski, John Riedl, John V. Carlis, and Ernest F. Retzel. 1994. Generalized suffix trees for biological sequence data: applications and implementation. In *Proceedings of the Twenty-Seventh Hawaii International Conference on System Sciences HICSS-94*, Vol. 5. IEEE Comput. Soc. Press, 35–44. https://doi.org/10.1109/HICSS.1994.323593

[4] Nicolas L Bray, Harold Pimentel, Páll Melsted, and Lior Pachter. 2016. Near-optimal probabilistic RNA-seq quantification. *Nature biotechnology* 34, 5 (may 2016), 525–527. https://doi.org/10.1038/nbt.3519 arXiv:1505.02710

[5] Fiona Cunningham, M Ridwan Amode, Daniel Barrell, Kathryn Beal, Konstantinos Billis, Simon Brent, Denise Carvalho-Silva, Peter Clapham, Guy Coates, Stephen Fitzgerald, et al. 2015. Ensembl 2015. *Nucleic Acids Research* 43, D1 (jan 2015), D662–D669. https://doi.org/10.1093/nar/gku1010

[6] Martin Farach. 1997. Optimal Suffix Tree Construction with Large Alphabets. *38th IEEE Symp. on Foundations of Computer Science* (1997), 137–143.

[7] Alyssa C Frazee, Andrew E Jaffe, Ben Langmead, and Jeffrey T Leek. 2015. Polyester: simulating RNA-seq datasets with differential transcript expression. *Bioinformatics* 31, 17 (jun 2015), 2778–84. https://doi.org/10.1101/006015

[8] Manuel Garber, Manfred G Grabherr, Mitchell Guttman, and Cole Trapnell. 2011. Computational methods for transcriptome annotation and quantification using RNA-seq. *Nature methods* 8, 6 (jun 2011), 469–77. https://doi.org/10.1038/nmeth.1613

[9] Steffen Heber, Max Alekseyev, Sing-Hoi Sze, Haixu Tang, and Pavel A Pevzner. 2002. Splicing graphs and EST assembly problem. *Bioinformatics* 18 Suppl 1, 1 (2002), S181–S188. https://doi.org/10.1093/bioinformatics/18.suppl_1.S181

[10] Michael Höhl, Stefan Kurtz, and Enno Ohlebusch. 2002. Efficient multiple genome alignment. *Bioinformatics* 18, Suppl 1 (jul 2002), S312–S320. https://doi.org/10.1093/bioinformatics/18.suppl_1.S312

[11] Lars Kaderali and Alexander Schliep. 2002. Selecting signature oligonucleotides to identify organisms using DNA arrays. *Bioinformatics* 18, 10 (oct 2002), 1340–1349. https://doi.org/10.1093/bioinformatics/18.10.1340

[12] Richard M. Karp and Michael O. Rabin. 1987. Efficient randomized pattern-matching algorithms. *IBM Journal of Research and Development* 31, 2 (mar 1987), 249–260. https://doi.org/10.1147/rd.312.0249

[13] Daehwan Kim, Geo Pertea, Cole Trapnell, Harold Pimentel, Ryan Kelley, and Steven L Salzberg. 2013. TopHat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions. *Genome biology* 14, 4 (apr 2013), R36. https://doi.org/10.1186/gb-2013-14-4-r36

[14] Hinanit Koltai and Carmiya Weingarten-Baror. 2008. Specificity of DNA microarray hybridization: characterization, effectors and approaches for data correction. *Nucleic acids research* 36, 7 (apr 2008), 2395–405. https://doi.org/10.1093/nar/gkn087

[15] Stefan Kurtz and Chris Schleiermacher. 1999. REPuter: fast computation of maximal repeats in complete genomes. *Bioinformatics* 15, 5 (may 1999), 426–427. https://doi.org/10.1093/bioinformatics/15.5.426

[16] Edward M. McCreight. 1976. A Space-Economical Suffix Tree Construction Algorithm. *J. ACM* 23, 12 (1976), 262–272. http://libeccio.di.unisa.it/TdP/suffix.pdf

[17] J. Ian Munro and Venkatesh Raman. 2001. Succinct Representation of Balanced Parentheses and Static Trees. *SIAM J. Comput.* 31, 3 (jan 2001), 762–776. https://doi.org/10.1137/S0097539799364092

[18] Herve Pagès, Daniel Bindreither, Marc Carlson, and Martin Morgan. 2013. SplicingGraphs: Create, manipulate, visualize splicing graphs, and assign RNA-seq reads to them. *R package version 1.14.0* (2013).

[19] Rob Patro, Stephen M Mount, and Carl Kingsford. 2014. Sailfish enables alignment-free isoform quantification from RNA-seq reads using lightweight algorithms. *Nature biotechnology* 32, 5 (may 2014), 462–4. https://doi.org/10.1038/nbt.2862

[20] Robert Petryszak, Tony Burdett, Benedetto Fiorelli, Nuno A Fonseca, Mar Gonzalez-Porta, Emma Hastings, Wolfgang Huber, Simon Jupp, Maria Keays, Nataliya Kryvych, et al. 2016. Expression Atlas update - An integrated database of gene and protein expression in humans, animals and plants. *Nucleic Acids Research* 44, D1 (jan 2016), D746–D752. https://doi.org/10.1093/nar/gkv1045

[21] Kunihiko Sadakane. 2007. Compressed Suffix Trees with Full Functionality. *Theory of Computing Systems* 41, 4 (2007), 589–607. http://web.stanford.edu/

[22] Michael Sammeth. 2009. Complete Alternative Splicing Events Are Bubbles in Splicing Graphs. *Journal of computational biology* 16, 8 (2009), 1117–1140. https://doi.org/10.1089/cmb.2009.0108

[23] Avi Srivastava, Hirak Sarkar, Nitish Gupta, and Rob Patro. 2016. RapMap: a rapid, sensitive and accurate tool for mapping RNA-seq reads to transcriptomes. *Bioinformatics* 32, 12 (oct 2016), i192–i200. https://doi.org/10.1101/029652

[24] Esko Ukkonen. 1995. On-line construction of suffix trees. *Algorithmica* 14, 3 (sep 1995), 249–260. https://doi.org/10.1007/BF01206331

[25] Niko Välimäki, Wolfgang Gerlach, Kashyap Dixit, and Veli Mäkinen. 2007. Compressed suffix tree —a basis for genome-scale sequence analysis. *Bioinformatics* 23, 5 (mar 2007), 629–30. https://doi.org/10.1093/bioinformatics/btl681

[26] Kai Wang, Darshan Singh, Zheng Zeng, Stephen J Coleman, Yan Huang, Gleb L Savich, Xiaping He, Piotr Mieczkowski, Sara a Grimm, Charles M Perou, James N MacLeod, Derek Y Chiang, Jan F Prins, and Jinze Liu. 2010. MapSplice: accurate mapping of RNA-seq reads for splice junction discovery. *Nucleic acids research* 38, 18 (oct 2010), e178. https://doi.org/10.1093/nar/gkq622

[27] Brian T Wilhelm and Josette-Renée Landry. 2009. RNA-Seq-quantitative measurement of expression through massively parallel RNA-sequencing. *Methods* 48, 3 (jul 2009), 249–57. https://doi.org/10.1016/j.ymeth.2009.03.016

[28] Zhaojun Zhang and Wei Wang. 2014. RNA-Skim: a rapid method for RNA-Seq quantification at transcript level. *Bioinformatics* 30, 12 (jun 2014), i283–i292. https://doi.org/10.1093/bioinformatics/btu288