nature biotechnology

Sailfish enables alignment-free isoform quantification from RNA-seq reads using lightweight algorithms

Rob Patro[1], Stephen M Mount[2,3] & Carl Kingsford[1]

We introduce Sailfish, a computational method for quantifying the abundance of previously annotated RNA isoforms from RNA-seq data. Because Sailfish entirely avoids mapping reads, a time-consuming step in all current methods, it provides quantification estimates much faster than do existing approaches (typically 20 times faster) without loss of accuracy. By facilitating frequent reanalysis of data and reducing the need to optimize parameters, Sailfish exemplifies the potential of lightweight algorithms for efficiently processing sequencing reads.

RNA-seq has become the *de facto* standard technique to measure gene expression, and it is commonly the first step in an analysis of differential expression among multiple samples[1]. However, the throughput of technologies used to generate transcriptomic sequencing reads is accelerating faster than the speed of the computers used to analyze these data. The growing repository of data from archived experiments invites reanalysis in the light of new discoveries, but existing methods are too time consuming to allow frequent reanalysis. The divide between data-acquisition and data-analysis capabilities will only increase as RNA-seq is adopted for clinical use[2]. Finally, the sensitivity of existing methods to parameter choices can affect analysis time and accuracy and can make a priori selection of the appropriate parameters difficult. We must develop efficient, lightweight algorithms with few parameters that minimize unnecessary computation.

Existing approaches to abundance estimation first use read-mapping tools, such as Bowtie[3], to determine potential locations from which the RNA-seq reads originated. Mapping the reads can require substantial computational resources and often leads to complicated models that try to account for read bias and error during inference, further adding to the time spent on analysis. Finally, some reads, known as multireads[4,5], can map to multiple, sometimes many, different transcripts. This ambiguity complicates the estimation of transcript abundances. Given read alignments, some of the most accurate transcript quantification tools estimate relative abundance using expectation-maximization (EM) procedures[5–7], where reads are first assigned to transcripts, and these assignments are then used to estimate transcript abundances, and these steps are repeated until convergence. In practice, both steps can be time consuming. Even when exploiting the parallel nature of the problem, mapping the reads from a reasonably sized RNA-seq experiment can take hours.

Recent tools, such as eXpress[7], aim to reduce the computational burden of isoform quantification by substantially altering the EM algorithm. Even for such advanced approaches, performing read alignment and processing the large number of alignments that result from ambiguously mapped reads remain bottlenecks and fundamentally limit the scalability of these approaches. Read-mapping is a complex problem, and the results of existing approaches depend on a host of parameters that affect how errors, gaps and mismatches are tolerated. These parameters are not always easily interpretable, and they can affect both the resources required for alignment and the results of downstream analysis.

Sailfish, our software for isoform quantification from RNA-seq data, is based on the philosophy of lightweight algorithms, which make frugal use of data, respect constant factors and effectively use concurrent hardware by working with small units of data where possible. Sailfish avoids mapping reads entirely (**Fig. 1**), resulting in large savings in time and space, and substantially reducing parametric complexity. A key technical contribution behind our approach is the observation that transcript coverage can be accurately estimated using counts of k-mers occurring in reads instead of alignments of reads. This results in the ability to obtain accurate estimates more than an order of magnitude faster than existing approaches, often in minutes instead of hours. For example, for the data described in **Figure 2**, Sailfish is >25 times faster than the next fastest method while providing estimates of equal accuracy. This accuracy is possible, despite independent counting and assignment of k-mers, because of an EM procedure that introduces a statistical coupling between k-mers.

Although the use of k-mers for the purpose of transcript quantification has not been reported previously, recent work[8] has shown that using k-mers directly for other RNA-seq tasks can be as effective or more effective than traditional approaches. By working with k-mers, we can replace computationally intensive read mapping with the much faster and simpler process of k-mer counting. We also avoid any dependence on read-mapping parameters. Yet, our approach is still able to handle sequencing errors because only the k-mers that overlap the erroneous bases will be discarded or mis-assigned, whereas the rest of the read can be processed as if it were error-free. One can view the k-mer counting mechanism as a proportional assignment of a read to a set of potential loci, with the strength of the assignment varying with the number of k-mers in the read that match the locus. Sailfish has only a single explicit parameter, the k-mer length. Longer k-mers may result in less ambiguity, which makes resolving their origin easier, but

[1]Lane Center for Computational Biology, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA. [2]Department of Cell Biology and Molecular Genetics, University of Maryland, College Park, Maryland, USA. [3]Center for Bioinformatics and Computational Biology, University of Maryland, College Park, Maryland, USA. Correspondence should be addressed to C.K. (carlk@cs.cmu.edu).

Received 17 August 2013; accepted 4 March 2014; published online 20 April 2014; doi:10.1038/nbt.2862

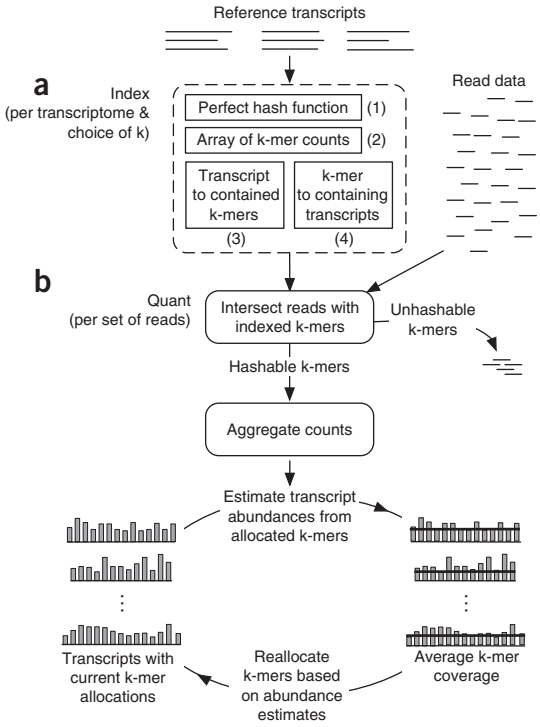
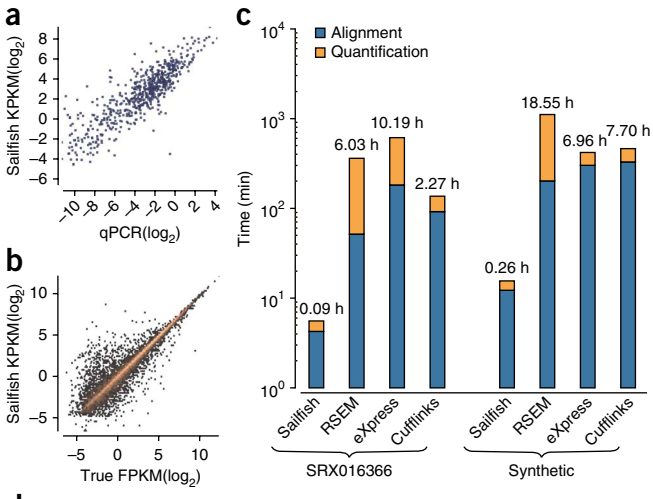


Figure 1 Overview of the Sailfish pipeline. (**a**,**b**) Sailfish consists of an indexing phase (**a**) that is invoked by the command 'sailfish index' and a quantification phase (**b**) invoked by the command 'sailfish quant'. The Sailfish index has four components: (1) a perfect hash function mapping each k-mer in the transcript set to a unique integer between 0 and *N*–1, where *N* is the number of unique k-mers in the set of transcripts; (2) an array recording the number of times each k-mer occurs in the reference set; (3) an index mapping each transcript to the multiset of k-mers that it contains; (4) an index mapping each k-mer to the set of transcripts in which it appears. The quantification phase consists of counting the indexed k-mers in the set of reads and then applying an EM procedure to determine the maximum-likelihood estimates of relative transcript abundance. K-mer count assignments are illustrated by vertical gray bars on lines representing known transcripts; the horizontal lines intersecting the gray bars represent the average of the current k-mer count assignments for each transcript.

may be more affected by errors in the reads (**Supplementary Fig. 1**). Further, many of our data structures can be represented as arrays of atomic integers (Online Methods). This allows our software to be concurrent and lock-free where possible, leading to an approach that scales well with the number of available CPUs (**Supplementary Fig. 2**). Additional benefits of the Sailfish approach are discussed in **Supplementary Note 1**.

Sailfish works in two phases: indexing and quantification (**Fig. 1**). A Sailfish index is built from a particular set of reference transcripts (a FASTA sequence file) and a specific choice of k-mer length, *k*. The index consists of data structures that make counting k-mers in a set of reads and resolving their potential origin in the set of transcripts efficient (Online Methods). The most important data structure in the index is the minimal perfect hash function[9] that maps each k-mer in the reference transcripts to an integer identifier such that no two k-mers share an identifier. Pairing the minimum perfect hash function with an atomically updateable array of k-mer counts allows k-mers to be counted even faster than with existing advanced lock-free hashes such as that used in Jellyfish[10]. The index also contains a pair of look-up tables that allow fast access to the k-mers appearing in a specific transcript as well as the transcripts in which a particular k-mer appears, both in amortized constant time. The index needs to be rebuilt only when the set of reference transcripts or the choice of *k* changes.

The quantification phase of Sailfish takes as input the above index and a set of RNA-seq reads and produces an estimate of the relative abundance of each transcript in the reference, measured in reads per kilobase per million mapped reads (RPKM), k-mers per kilobase per million mapped k-mers (KPKM) and transcripts per million (TPM) (see Online Methods for the definitions of these measures). First, Sailfish counts the number of times each indexed k-mer occurs in the set of reads. Owing to the index, this process is efficient and scalable (**Supplementary Fig. 2**). Sailfish then applies an EM procedure to determine maximum likelihood estimates



d

	Human brain tissue				Synthetic			
	Sailfish	RSEM	eXpress	Cufflinks	Sailfish	RSEM	eXpress	Cufflinks
Pearson	0.85	0.82	0.85	0.85	0.96	0.96	0.95	0.94
Spearman	0.84	0.80	0.85	0.85	0.76	0.77	0.77	0.75
RMSE	–	–	–	–	6.06	8.90	8.83	10.05
medPE	–	–	–	–	6.50	12.48	14.06	12.42

Figure 2 Speed and accuracy of Sailfish. (**a**) The correlation between qPCR estimates of gene abundance (*x* axis) and the estimates of Sailfish. The qPCR results are taken from the microarray quality control study (MAQC)[15]. The results shown here are for the human brain tissue, and the RNA-seq–based estimates were computed using the reads from SRA accession SRX016366 (81,250,481 35bp single-end reads). The set of transcripts used in this experiment were the curated RefSeq[20] transcripts (accession prefix NM) from hg18 (31,148 transcripts). (**b**) The correlation between the ground truth FPKM in a simulated data set (*x* axis) and the abundance estimates of Sailfish. The quantification in this experiment was performed on a set of 96,520 transcript sequences taken from Ensembl[21] GRCh37.73. (**c**) The total time taken by each method, Sailfish, RSEM, eXpress and Cufflinks, to estimate isoform abundance on each data set. The total time taken by a method is the height of the corresponding bar, and the total is further broken down into the time taken to perform read-alignment (for Sailfish, we instead measured the time taken to count the k-mers in the read set) and the time taken to quantify abundance given the aligned reads (or k-mer counts). All tools were run in multithreaded mode (where applicable) and were allowed to use up to 16 threads. (**d**) Accuracy of each of the methods on human brain tissue and a synthetic data set. Accuracy is measured by the Pearson (log-transformed) and Spearman correlation coefficients between estimated abundance values and MAQC qPCR data (for human brain tissue) or ground truth (for simulated data). Root-mean-square error (RMSE) and median percentage error (medPE) are calculated as described in the Online Methods.

for the relative abundance of each transcript. Conceptually, this procedure is similar to the EM algorithm used by RSEM[6], except that k-mers rather than fragments are probabilistically assigned to transcripts, and a two-step variant of EM is used to speed up convergence. The estimation procedure first assigns k-mers proportionally to their occurrence in the transcripts. These initial allocations are then used to estimate the expected coverage for each transcript (Online Methods, equation (3)). In turn, these expected coverage values alter the assignment probabilities of k-mers to transcripts (Online Methods, equation (1)). Using these basic EM steps as a building block, we apply a globally convergent EM acceleration method, SQUAREM[11], that substantially increases the convergence rate of the estimation procedure by modifying the parameter update step based on the current solution path and the estimated distance from the fixed point (Online Methods, algorithm 2).

We reduce the number of variables that need to be fit by the EM procedure by collapsing k-mers into equivalence classes, where two k-mers are equivalent if they occur in the same set of transcript sequences with the same rate (Online Methods). The idea of data reduction by equivalence classes has been explored in the context of reads[12,13]. Using k-mer equivalence classes has the added benefit that the classes can be computed based solely on the reference transcriptome without considering the read data. This reduction in variables substantially reduces the computational requirements of the EM procedure (**Supplementary Note 1**).

Once the EM procedure converges, the estimated abundances are corrected for systematic errors resulting from sequence composition bias and transcript length using a regression approach similar to a previous method[14], though using random forest regression instead of a generalized additive model. This correction is applied after initial estimates have been produced rather than at a read mapping or fragment assignment stage, requiring fewer variables to be fit during bias correction.

We compared Sailfish to RSEM[6], eXpress[7] and Cufflinks[5], using both real and synthetic data. Accuracy on real data was quantified by the agreement between RNA-seq–based expression estimates computed by each piece of software and qPCR measurements for the same sample (human brain tissue in **Fig. 2** and **Supplementary Fig. 3**, and universal human reference tissue in **Supplementary Fig. 4**). The qPCR measurements for these samples were performed as part of the Microarray Quality Control Study[15]; RNA-seq experiments were later performed on the same samples[16]. Abundance measurements for qPCR are given at the resolution of genes. To compare these measurements with the transcript-level estimates, we summed the estimates for all isoforms belonging to a gene. We compare predicted abundances using correlation coefficients and, on the synthetic data, root-mean-square error (RMSE) and median percentage error (medPE) (**Supplementary Note 2**). Taken together, these results show that the speed of Sailfish does not sacrifice accuracy (**Fig. 2**).

To show that Sailfish is accurate at the isoform level, we generated synthetic data using the Flux Simulator[17], which allows versatile modeling of various RNA-seq protocols (**Supplementary Note 3**). Unlike synthetic data used in previous work[6,7], the procedure used by the Flux Simulator is not based on the generative model underlying our estimation procedure. Sailfish remains accurate at the isoform level (**Fig. 2** and **Supplementary Figs. 5** and **6**). We also find that Sailfish is robust to differences in the underlying set of reference transcripts used for quantification (**Supplementary Fig. 7**), and that it remains accurate when applied to a collection of highly similar genes (**Supplementary Table 1**).

Sailfish applies the idea of lightweight algorithms to the problem of isoform quantification from RNA-seq reads and in doing so achieves a breakthrough in terms of speed. By eliminating read mapping from the expression estimation pipeline, we improve the speed of the process and also simplify it considerably, eliminating the burden of choosing all but a single parameter from the user. Sailfish can be used in a *de novo* setting for nonmodel organisms, along with other reference-free tools[18,19], where some or all of the transcripts are assembled directly from the RNA-seq reads. The memory usage of Sailfish is similar to that of other tools, using 2–6 Gb of RAM during isoform quantification for the experiments reported here. As the size and number of RNA-seq experiments grow, we expect Sailfish and its paradigm to remain efficient for isoform quantification because the memory footprint is bounded by the size and complexity of the target transcripts, not the number of reads, and the only computation that grows explicitly in the number of reads—k-mer counting—has been designed to effectively exploit many CPU cores.

Sailfish is free and open-source software and is available as **Supplementary Data** and at http://www.cs.cmu.edu/~ckingsf/software/sailfish.

Note: Any Supplementary Information and Source Data files are available in the online version of the paper.

ACKNOWLEDGMENTS

This work has been partially funded by the US National Science Foundation (CCF-1256087, CCF-1053918, and EF-0849899) and US National Institutes of Health (R21AI085376, R21HG006913 and R01HG007104). C.K. received support as an Alfred P. Sloan Research Fellow. We would like to thank A. Roberts for helping to diagnose and resolve an artifact in an earlier version of this manuscript pertaining to the synthetic data generated by the Flux Simulator.

AUTHOR CONTRIBUTIONS

R.P., S.M.M. and C.K. designed the method and algorithms, devised the experiments, and wrote the manuscript. R.P. implemented the Sailfish software.

COMPETING FINANCIAL INTERESTS

The authors declare no competing financial interests.

1. Soneson, C. & Delorenzi, M. *BMC Bioinformatics* **14**, 91 (2013).
2. Roychowdhury, S. *et al. Sci. Trans. Med.* 111ra121 (2011).
3. Langmead, B., Trapnell, C., Pop, M. & Salzberg, S.L. *Genome Biol.* **10**, R25 (2009).
4. Mortazavi, A., Williams, B.A., McCue, K., Schaeffer, L. & Wold, B. *Nat. Methods* **5**, 621–628 (2008).
5. Trapnell, C. *et al. Nat. Biotechnol.* **28**, 511–515 (2010).
6. Li, B. & Dewey, C. *BMC Bioinformatics* **12**, 323 (2011).
7. Roberts, A. & Pachter, L. *Nat. Methods* **10**, 71–73 (2012).
8. Philippe, N., Salson, M., Commes, T. & Rivals, E. *Genome Biol.* **14**, R30 (2013).
9. Botelho, F.C., Pagh, R. & Ziviani, N. *Proceedings of the 10th International Workshop on Algorithms and Data Structures*, Halifax, NS, Canada, August 15–17, 2007 (eds. Dehne, F., Sack, J.-R. & Zeh, N.)139–150 (Springer, 2007).
10. Marçais, G. & Kingsford, C. *Bioinformatics* **27**, 764–770 (2011).
11. Varadhan, R. & Roland, C. *Scand. J. Stat.* **35**, 335–353 (2008).
12. Nicolae, M., Mangul, S., Mandoiu, I. & Zelikovsky, A. *Algorithms Mol. Biol.* **6**, 9 (2011).
13. Salzman, J., Jiang, H. & Wong, W.H. *Stat. Sci.* **26**, 62–83 (2011).
14. Zheng, W., Chung, L.M. & Zhao, H. *BMC Bioinformatics* **12**, 290 (2011).
15. Shi, L. *et al. Nat. Biotechnol.* **24**, 1151–1161 (2006).
16. Bullard, J.H., Purdom, E., Hansen, K.D. & Dudoit, S. *BMC Bioinformatics* **11**, 94 (2010).
17. Griebel, T. *et al. Nucleic Acids Res.* **40**, 10073–10083 (2012).
18. Grabherr, M.G. *et al. Nat. Biotechnol.* **29**, 644–652 (2011).
19. Sacomoto, G.A. *et al. BMC Bioinformatics* **13** (suppl. 6), S5 (2012).
20. Pruitt, K.D., Tatusova, T., Brown, G.R. & Maglott, D.R. *Nucleic Acids Res.* **40**, D1, D130–D135 (2012).
21. Flicek, P. *et al. Nucleic Acids Res.* **41**, D1, D48–D55 (2013).

ONLINE METHODS

Indexing. The first step in the Sailfish pipeline is building an index from the set of reference transcripts T. Given a k-mer length k, we compute an index $I_k(T)$ containing four components. The first component is a minimum perfect hash function h on the set of k-mers **kmers**(T) contained in T. A minimum perfect hash function is a bijection between **kmers**(T) and the set of integers $\{0,1,\ldots, |\textbf{kmers}(T)| - 1\}$. Sailfish uses the BDZ minimum perfect hash function[9]. The second component of the index is an array C containing a count $C(s_i)$ for every $s_i \in$ **kmers**(T). Finally, the index contains a lookup table mapping each transcript to the multiset of k-mers that it contains, and a reverse lookup table mapping each k-mer to the set of transcripts in which it appears. The index is a product only of the reference transcripts and the choice of k, and thus needs only to be recomputed when either of these change.

Quantification. The second step in the Sailfish pipeline is the quantification of relative transcript abundance; this requires the Sailfish index $I_k(T)$ for the reference transcripts T as well as a set of RNA-seq reads $\mathfrak{R}$. First, we count the number of occurrences of each $s_i \in$ **kmers**(T) $\cap$ **kmers**($\mathfrak{R}$). Because we know exactly the set of k-mers that need to be counted and already have a perfect hash function h for this set, we can perform this counting in a particularly efficient manner, even faster than efficient hash-based approaches. For example, performing concurrent 20-mer lookups using 8 threads, Jellyfish[10] requires an average of 0.35 µs/key while the minimal perfect hash requires an average of 0.1 µs/key. We maintain an array $C_{\mathfrak{R}}$ of the appropriate size $|\textbf{kmers}(T)|$, where $C_{\mathfrak{R}}(h(s_i))$ contains the number of times we have thus far observed s_i in $\mathfrak{R}$.

In an unstranded protocol, sequencing reads, and hence the k-mers they contain, may originate from transcripts in either the forward or reverse direction. To account for both possibilities, we check both the forward and reverse-complement k-mers from each read and use a majority-rule heuristic to determine which of the k-mers to increment in the final array of counts $C_{\mathfrak{R}}$. If the number of k-mers appearing in h from the forward direction of the read is greater than the number of reverse-complement k-mers, then we only increment the counts for k-mers appearing in this read in the forward direction. Otherwise, only counts for k-mers appearing in the reverse-complement of this read are incremented in the array of counts. Ties are broken in favor of the forward-directed reads. In a stranded RNA-seq protocol, the reads have a known orientation. Reads provided to Sailfish can be specified as originating from unstranded, forward-strand or reverse-strand reads, and the appropriate k-mers are counted in each case. By taking advantage of atomic integers and the compare-and-swap (CAS) operation provided by modern processors, which allows many hardware threads to efficiently update the value of a memory location without the need for explicit locking, we can stream through and update the counts in $C_{\mathfrak{R}}$ in parallel while sustaining very little resource contention.

We then apply an expectation-maximization (EM) algorithm to obtain estimates of the relative abundance of each transcript. We define a k-mer equivalence class as the set of all k-mers that appear in the same set of transcripts with the same frequency. In other words, let $\times(s)$ be a vector such that entry t of $\times(s)$ gives how many times k-mer s appears in transcript $t \in T$. Then the equivalence class of a k-mer s_i is given by $[s_i] = \{s_j \in \textbf{kmers}(T) : \times(s_j) = \times(s_i)\}$. When performing the EM procedure, we will allocate counts to transcripts according to the set of equivalence classes rather than the full set of k-mers. We will let

$$\mathrm{L}(s_i) = \sum_{s_j \in [s_i]} C_{\mathfrak{R}}\left(h\left(s_j\right)\right) \tag{1}$$

denote the total count of k-mers in $\mathfrak{R}$ that originate from equivalence class $[s_i]$. We say that transcript t contains equivalence class $[s]$ if $[s]$ is a subset of the multiset of k-mers of t and denote this by $[s] \subseteq t$.

Estimating abundances via an EM algorithm. The EM algorithm (Algo. 1) alternates between estimating the fraction of counts of each observed k-mer that originates from each transcript (E-step) and estimating the relative abundances of all transcripts given this allocation (M-step). We initially allocate k-mers to transcripts proportional to their occurrences in the transcript (i.e., if a transcript is the only potential origin for a particular k-mer, then all observations of that k-mer are attributed to this transcript, whereas for a k-mer that appears once in each of n different transcripts and occurs m times in the set of reads, m/n observations are attributed to each potential transcript of origin).

The E-step of the EM algorithm computes the fraction of each k-mer equivalence class's total count that is allocated to each transcript. For equivalence class $[s_j]$ and transcript t_i, this value is computed by

$$\alpha(j,i) = \frac{\mu_i' \mathrm{L}\left(s_j\right)}{\sum_{t \supseteq [s_j]} \mu_t'} \tag{2}$$

where μ_i' is the currently estimated relative abundance of transcript i. These allocations are then used in the M-step of the algorithm to compute the relative abundance of each transcript. The relative abundance of transcript i is estimated by

$$\mu_i' = \frac{\mu_i}{\sum_{t_j \in T} \mu_j} \tag{3}$$

where μ_i is

$$\mu_i = \frac{\sum_{[s_j] \subseteq t_i} \alpha(j,i)}{l_i'} \tag{4}$$

The variable l_i' denotes the adjusted length of transcript i and is simply $l_i' = l_i - k + 1$, where l_i is the length of transcript i in nucleotides.

However, rather than perform the standard EM update steps, we perform updates according to the SQUAREM procedure[11] described in Algo. 2, where $\mu' = \mu'_0, \ldots, \mu'_{|T|}$ is a vector of relative abundance maximum-likelihood estimates, and EM(•) is a standard iteration of the expectation-maximization procedure as outlined in Algo. 1. For a detailed explanation of the SQUAREM procedure and its proof of convergence, see ref. 11. Intuitively, the SQUAREM procedure builds an approximation of the Jacobian of μ' from three successive steps along the EM solution path, and uses the magnitude of the differences between these solutions to determine a step size by which to update the estimates according to the update rule (line 7). The procedure is then capable of making relatively large updates to the μ' parameters, which substantially improves the speed of convergence. In Sailfish, the iterative SQUAREM procedure is repeated until a specified convergence criterion is met (by default, the procedure terminates when no transcript with a relative abundance greater than 10^{-7} has a relative change greater than half of a percent between consecutive iterations). Additionally, a user may specify either a fixed number of iterations to perform or a required minimum relative change, ε, in transcript abundance estimates between SQUAREM iterations; if no relative change in transcript abundance exceeds ε, then the procedure is considered to have converged and the estimation procedure terminates.

Bias correction. The bias correction procedure implemented in Sailfish is based on the model introduced by Zheng *et al.*[14]. Briefly, it performs a regression analysis on a set of potential bias factors where the response variables are the estimated transcript abundances (KPKMs). Sailfish automatically considers transcript length, GC content and dinucleotide frequencies as potential bias factors, as this specific set of features was suggested by Zheng *et al.*[14] For each transcript, the prediction of the regression model represents the contribution of the bias factors to this transcript's estimated abundance. Hence, these regression estimates (which may be positive or negative) are subtracted from the original estimates to obtain bias-corrected KPKMs. For further details

on this bias correction procedure, see ref. 14. The original method used a generalized additive model for regression; Sailfish implements the approach using random forest regression to leverage high-performance implementations of this technique. The key idea here is to do the bias correction after abundance estimation rather than earlier in the pipeline. The bias correction of Sailfish can be disabled with the `--no-bias-correction` command line option. Finally, we note that it is possible to include other potential features, like normalized coverage plots that can encode positional bias, into the bias correction phase. However, in the current version of Sailfish, we have not implemented or tested bias correction for these features.

Computing KPKM, RPKM and TPM. Sailfish outputs K-mers Per Kilobase per Million mapped k-mers (KPKM), Reads Per Kilobase per Million mapped reads (RPKM) and Transcripts Per Million (TPM) as quantities predicting the relative abundance of different isoforms. The RPKM estimate is the most commonly used and is ideally 10^9 times the rate at which reads are observed at a given position, but the TPM estimate has also become somewhat common[6]. For Sailfish, the KPKM measure is more natural than RPKM, as the k-mer is the basic unit of transcript coverage; however, the two measures are proportional. Given the relative transcript abundances estimated by the EM procedure described above, the TPM for transcript i is given by

$$\mathrm{TPM}_i = 10^6 \mu_i' \tag{5}$$

Let C_i be the number of k-mers mapped to transcript i. Then, the KPKM is given by

$$\mathrm{KPKM}_i = \frac{\frac{C_i}{l_i / 10^3}}{N / 10^6} = \frac{10^9 \frac{C_i}{l_i}}{N} \approx \frac{10^9 \mu_i}{N} \tag{6}$$

where N is the total count of mapped (i.e., hashable) k-mers and the final equality is approximate only because we replace l_i with l_i'. The KPKM is proportional to the RPKM, which can be estimated by replacing C_i with the estimated number of reads mapped to transcript i, and N with the estimated number of mapped reads; these can be calculated from the mapped k-mer counts and the average read length. The Fragments Per Kilobase per Million mapped fragments (FPKM) measure is also proportional to the K/RPKM, but is meant to denote that fragments rather than reads are mapped—for single end data, the reads and fragments coincide, for paired-end data, each concordant read pair is considered a single fragment.

Accuracy metrics. For all experiments, the Pearson correlation metric was computed between estimates of abundance that were log transformed. As in previous work[6], this was done to prevent the most abundant transcripts from dominating the correlations, but it necessitates discarding samples that have either a true or estimated abundance of 0. Spearman correlations were computed on estimates that were not log transformed. For the synthetic data, we additionally compute the Root Mean Squared Error (RMSE) and median Percentage Error. Given the vectors $\boldsymbol{x} = x_1, x_2, \ldots, x_n$ of ground-truth abundances and $\boldsymbol{y} = y_1, y_2, \ldots, y_n$ of estimated abundances, the RMSE is defined as

$$\mathrm{RMSE}(\mathbf{x}, \mathbf{y}) = \sqrt{\frac{\sum_{i=1}^{n} (x_i - y_i)^2}{n}} \tag{7}$$

The percentage error (PE) between a true abundance x_i and the corresponding estimate y_i is defined as

$$\mathrm{PE}(x_i, y_i) = 100 \times \frac{|x_i - y_i|}{x_i} \tag{8}$$

Unlike the definition presented in other work[12], we do not define division by 0 when computing the percentage error. We then define the medPE as the median of the percentage error for all transcripts that have a true abundance greater than 0 (i.e., the median of the values $\mathrm{PE}(x_i, y_i)$ for all $1 \le i \le n$ where $x_i \neq 0$).

When comparing against the qPCR-based abundances, we used the KPKM estimates produced by Sailfish and FPKM estimates produced by RSEM, Cufflinks and eXpress directly. The FluxSimulator records the number of reads that were actually generated by each transcript during the simulated sequencing experiment. These read counts and the transcript lengths were used to compute the ground truth FPKMs directly using equation (6). However, we found that the FPKM estimates computed by RSEM, eXpress and Cufflinks appeared, for a number of transcripts, to be systematically inflated with respect to the ground truth FPKMs (this inflation persisted even when estimates were compared directly to ground truth relative transcript fractions). This appears to be due to the effective-length normalization procedures employed by these methods. Indeed, when Cufflinks was run with `--no-effective-length-correction`, the RMSE and medPE decreased significantly, while the correlations with ground truth FPKMs increased marginally as well. Thus, for the synthetic tests, all Cufflinks results were computed with the `--no-effective-length-correction` flag enabled. Though RSEM and eXpress do not expose such a flag directly, we computed FPKM estimates without effective-length corrections for these methods by using the estimated read counts (output directly by both methods) and the true transcript lengths using equation (6). Using FPKMs computed in this manner reduced the RMSE from 2,083.04 to 8.83 and medPE from 19.03 to 14.06 for eXpress and reduced the RMSE from 18.77 to 8.90 and the medPE from 17.90 to 12.48 for RSEM. Sailfish computes the estimated k-mer counts and KPKMs in a consistent manner directly from the underlying transcript abundances, so there is no difference between the KPKMs provided by Sailfish and those that would be computed using the estimated k-mer counts directly.

Simulated data. The simulated *Homo sapiens* RNA-seq data were generated by the Flux Simulator[17] v1.2.1 with the parameters listed in **Supplementary Note 3**. This resulted in a dataset of 75 M 76 bp × 2 paired-end reads. The reads produced by the Flux Simulator are grouped by their origin (i.e., reads originating from the same genomic locus appear together in the read file). However, this synthetic artifact violates the assumption made by eXpress that reads are produced in a random order. Thus, we post-process the reads generated by the Flux Simulator and shuffle them (while keeping mates appropriately paired) into a random order. We also randomize the orientation of these reads to simulate an unstranded protocol. The shuffled, mate-pair reads are then split into separate files. To ensure a sufficient mapping rate under the parameters used to produce alignments for eXpress and Cufflinks (`-v 3` for Bowtie and `-N 3` for TopHat[22]), the synthetic reads were quality trimmed using SICKLE (https://github.com/najoshi/sickle) with the default parameters. Untrimmed reads are more appropriate for RSEM's default parameters, and so RSEM was provided with untrimmed reads. While mapping rates increased significantly for the trimmed reads, the overall accuracy of the methods depended little on whether trimmed or untrimmed reads were used. RSEM, eXpress and Cufflinks were all given paired-end alignments since they make special use of these data. TopHat[22] was provided with the option `--mate-inner-dist 200`, to adjust the expected mate-pair inner-distance to the simulated average, and Bowtie was given the option `-X 800` to ensure that paired-end reads originating from all valid fragments would be considered. The untrimmed synthetic read files were provided directly to Sailfish without any extra information since the same quantification procedure is used whether single or paired-end reads are provided.

Software comparisons. For eXpress, all reads were aligned with Bowtie[3] v0.12.9 using the parameters `-a` and `-v 3`, as suggested in the eXpress manuscript, which allows up to three mismatches per read and reports all

doi:10.1038/nbt.2862

alignments. All alignments for RSEM were computed with default parameters using the `rsem-calculate-expression` command, which used Bowtie v0.12.9 as the underlying aligner. To prepare alignments for Cufflinks, TopHat was run using Bowtie v0.12.9 (`--bowtie1`) and with options `-N3` and `--read-edit-dist 3` to allow up to three mismatches per read. For RSEM, eXpress and Cufflinks, the reported times are the sum of the times required for alignment and the times required for quantification. The time required for each method is further decomposed into the times for the alignment and quantification steps in **Figure 2**.

Choice of software options and effect on runtime. Most expression estimation software, including RSEM, eXpress and Cufflinks, provides a myriad of program options to the user, which allow for trade-offs between various desiderata. For example, the total time required by TopHat and Cufflinks is lower when Cufflinks is run without bias correction (e.g., 1.92 h as opposed to 2.27 h with bias correction on the SRX016366 data). However, without bias correction, Cufflinks yields slightly lower accuracy (Pearson $\sigma = 0.82$, Spearman $\rho = 0.81$) than the other methods, while still taking 16 times longer to run than Sailfish. Similarly, although aligned reads can be streamed directly into eXpress via Bowtie, we empirically observed lower overall runtimes when aligning reads and quantifying expressions separately (and in sequence), so these times were reported. Finally, when we provided RSEM with the alignments used by eXpress (as was done in previous work[7]), we observed substantially increased runtime on the qPCR datasets (46.5 h with the alignments generated from SRX016366 and 35.6 h with the alignments generated from SRX016367), since RSEM does not appear to efficiently handle the large number of multiple alignments that can be generated by short reads when allowing many mismatches; instead, we report timings for RSEM when the alignment is performed using its own default parameters. In general, we attempted to run each piece of software with options that would be common in a standard usage scenario. However, despite the inherent difficulty of comparing a set of tools parameterized on an array of potential options, the core thesis that Sailfish can provide accurate expression estimates much faster than any existing tool remains true, as the fastest performing alternatives, even when sacrificing accuracy for speed, were over an order of magnitude slower than Sailfish.

Sailfish version 0.6.3 was used for all experiments, and all analyses were done with a k-mer size of $k = 20$ and using 16 concurrent threads (`-p 16`). In all experiments involving the real but not simulated data, Sailfish was run with the `--polya` option, which discards k-mers consisting of k consecutive As or Ts, and bias correction was enabled. The KPKM values reported by Sailfish were used as transcript abundance estimates.

RSEM[6] version 1.2.3 was run with the `--no-bam-output` flag, and with –p 16. In the experiments involving the real data, `rsem-prepare-reference` was used to prepare a reference index that appended poly-A tails to all transcripts, and the quantification was performed with the `--estimate-rspd` flag. For the synthetic data, RSEM was given the `--paired-end` option to produce paired-end alignments from the reads. Apart from these options, RSEM was run with the default parameters for all experiments.

eXpress[7] version 1.3.1 was used for all experiments. It was run with default parameters on the MACQ data, and without bias correction (`--no-bias-correct`) on the synthetic data. For the synthetic test, Bowtie was given the appropriate `-1` and `-2` options to compute paired-end alignments.

Cufflinks[5] version 2.1.1 was used for experiments and was run with bias correction (`-b`) and multi-read recovery (`-u`) on the MACQ data, and with multi-read recovery (`-u`) and no effective-length correction (`--no-effective-length-correction`) on the synthetic data. For all tests, both Cufflinks and TopHat were run with the `-p 16` option to take advantage of up to 16 concurrent threads of execution.

All experiments were run on a computer with 8 AMD Opteron 6220 processors (four cores each) and 256 Gb of RAM. For all experiments, the wall time was measured using the built-in bash `time` command.

Implementation of Sailfish. Sailfish has two basic sub-commands, `index` and `quant`. The `index` command initially builds a hash of all k-mers in the set of reference transcripts using the Jellyfish[10] software. This hash is then used to build the minimum perfect hash, count array and look-up tables described above. The `index` command takes as input a k-mer size via the `-k` option and a set of reference transcripts in FASTA format via the `-t` parameter. It produces the Sailfish index described above, and it can optionally take advantage of multiple threads with the target number of threads being provided via a `-p` option.

The `quant` sub-command estimates the relative abundance of transcripts given a set of reads. The `quant` command takes as input a Sailfish `index` (computed via the `index` command described above and provided via the `-i` parameter). Additionally, it requires the set of reads, provided as a list of FASTA or FASTQ files. Finally, just as can the `index` command, the `quant` command can take advantage of multiple processors, the target number of which is provided via the – p option.

Sailfish is implemented in C++11 and takes advantage of several C++11 language and library features. In particular, Sailfish makes heavy use of built-in atomic data types. Parallelization across multiple threads in Sailfish is accomplished via a combination of the standard library's thread facilities and the Intel Threading Building Blocks (TBB) library[23]. Sailfish is available as an open-source program under the GPLv3 license, and has been developed and tested on Linux and Macintosh OS X.

Algorithm 1. An EM iteration. One iteration of the expectation-maximization procedure that updates the estimated k-mer allocations $\alpha(\cdot,\cdot)$ and computes new estimates of relative transcript abundance μ'' based on the current estimates of relative transcript abundance μ'.

function $\mathrm{EM}(\boldsymbol{\mu}')$
begin
  for $[s_j]$ **do**
    $w = \sum_{t \supseteq [s_j]} \mu'_t$
    for $t \supseteq [s_j]$ **do**
      $\alpha(j,i) = \mu'_i \mathrm{L}(s_j)/w$
  $y = 0$
  for $t \in \mathrm{T}$ **do**
    $C_t = \sum_{[s_j] \subseteq t_i} \alpha(j,i)$
    $\mu^*_t = C_t / l'_i$
    $y = y + \mu^*_t$
  for $t \in \mathrm{T}$ **do**
    $\mu''_t = \mu^*_t / y$
  return $\boldsymbol{\mu}'' = \langle \boldsymbol{\mu}''_0, \boldsymbol{\mu}''_1, \ldots, \boldsymbol{\mu}''_{|\mathrm{T}|} \rangle$

Algorithm 2. A SQUAREM iteration. One iteration updates the relative abundance estimates according to an accelerated EM procedure whose update direction and magnitude are dynamically computed[11].

$\boldsymbol{\mu}'_1 = \mathrm{EM}(\boldsymbol{\mu}'_0)$
$\boldsymbol{\mu}'_2 = \mathrm{EM}(\boldsymbol{\mu}'_1)$
$r = \boldsymbol{\mu}'_1 - \boldsymbol{\mu}'_0$
$v = (\boldsymbol{\mu}'_2 - \boldsymbol{\mu}'_1) - r$
$\gamma = -||r|| / ||v||$
modify γ by backtracking, if necessary, to ensure global convergence
$\boldsymbol{\mu}'_3 = \max(\mathbf{0}, \boldsymbol{\mu}'_0 - 2\gamma r + \gamma^2 v)$
$\boldsymbol{\mu}'_0 = \mathrm{EM}(\boldsymbol{\mu}'_3)$

22. Trapnell, C., Pachter, L. & Salzberg, S. *Bioinformatics* **25**, 1105–1111 (2009).
23. Pheatt, C. *J. Comput. Sci. Coll.* **23**, 298–298 (2008).