Salmon provides fast and bias-aware quantification of transcript expression

Rob Patro[1], Geet Duggal[2], Michael I Love[3,4], Rafael A Irizarry[3,4] & Carl Kingsford[5]

We introduce Salmon, a lightweight method for quantifying transcript abundance from RNA–seq reads. Salmon combines a new dual-phase parallel inference algorithm and feature-rich bias models with an ultra-fast read mapping procedure. It is the first transcriptome-wide quantifier to correct for fragment GC-content bias, which, as we demonstrate here, substantially improves the accuracy of abundance estimates and the sensitivity of subsequent differential expression analysis.

Estimating transcript abundance is a fundamental task in genomics. These estimates are used for the classification of diseases and their subtypes[1], understanding expression changes during development[2], and tracking the progression of cancer[3]. Accurate and efficient quantification of transcript abundance from RNA–seq data is an especially pressing problem owing to the wide range of technical biases that affect the RNA–seq fragmentation, amplification, and sequencing process[4,5], the exponentially increasing number of experiments, and the adoption of expression data for medical diagnosis[6]. Traditional quantification algorithms that use alignments of sequencing reads to the genome or transcriptome require substantial computational resources[7] and do not scale well with the rate at which data are produced[8]. Sailfish[9] achieved an order-of-magnitude improvement in speed by replacing traditional read alignment with the allocation of exact *k*-mers to transcripts. kallisto[10] achieves similar speed improvements and further reduces the gap in accuracy with traditional alignment-based methods by replacing the *k*-mer-counting approach with a procedure called pseudoalignment, which is capable of rapidly determining the set of transcripts that are compatible with a given sequenced fragment.

However, existing methods for transcriptome-wide abundance estimation—both alignment-based and alignment-free—lack sample-specific bias models rich enough to capture important effects like fragment GC-content bias. When correction is not applied, these biases can lead to undesired effects, for example, a loss of false discovery rate (FDR) control in differential expression studies[5].

Our novel quantification procedure, Salmon, employs a new dual-phase statistical inference procedure and sample-specific bias models that account for sequence-specific, fragment GC-content, and positional biases. It achieves the same order-of-magnitude benefits in speed as kallisto and Sailfish but with greater accuracy. Salmon consists of three components: a lightweight mapping model, an online phase that estimates initial expression levels and model parameters, and an offline phase that refines expression estimates (**Supplementary Fig. 1**). This two-phase inference procedure allows Salmon to build a probabilistic model of the sequencing experiment that incorporates information—like terms contributing to the conditional probability of drawing a fragment of a given transcript—not considered by Sailfish[9] and kallisto[10]. Salmon is capable of either mapping sequencing reads itself by using a fast and lightweight procedure called quasi-mapping or accepting precomputed read alignments in the form of a SAM or BAM file. In this paper, we mainly employ Salmon's lightweight mapping mode, although we also test using precomputed alignments. Furthermore, Salmon provides the ability to estimate abundance uncertainty due to random sampling and the ambiguity introduced by reads mapping to multiple transcripts (Online Methods).

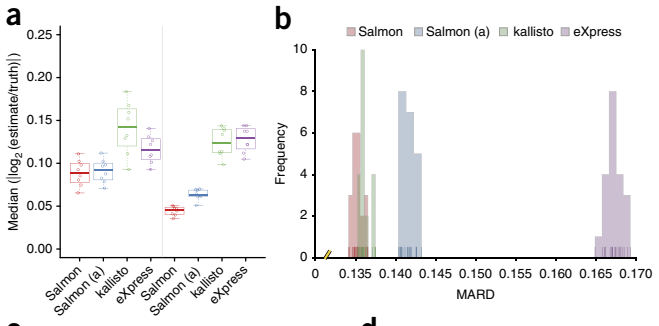
Unlike pseudoalignment, Salmon's lightweight mapping procedure tracks, by default, the position and orientation of all mapped fragments. This information is used in conjunction with the abundances from online inference to compute per-fragment conditional probabilities, rather than just transcript compatibility. These probabilities are used to estimate auxiliary models and bias terms and to update abundance estimates, and they are subsequently aggregated into weights for the rich equivalence classes used during offline inference (Online Methods **Supplementary Algorithm 1**).

Using experimental data from the GEUVADIS[11] and SEQC[12] studies and synthetic data from the Polyester[13] and RSEM-sim[14] simulators, we benchmarked Salmon against kallisto[10] and eXpress[15] + Bowtie2 (ref. 16); both of these methods also implement their own bias models. We also tested Salmon using traditional alignments (from Bowtie2) as input (denoted as "Salmon (a)"). We show that Salmon typically outperforms both kallisto and eXpress in terms of accuracy (**Fig. 1** and **Supplementary Figs. 2** and **3**). We note that all these tools address transcript quantification and do not identify or assemble novel transcripts (**Supplementary Note 1**).

Salmon's dual-phase inference algorithm and sample-specific bias models yielded improved inter-replicate concordance (**Supplementary Fig. 4**) when compared with both kallisto and eXpress. For example, when used for differential expression (DE) testing, the quantification estimates produced by Salmon exhibited markedly higher sensitivity than those from existing methods at

[1]Department of Computer Science, Stony Brook University, Stony Brook, New York, USA. [2]DNAnexus, Mountain View, California, USA. [3]Department of Biostatistics and Computational Biology, Dana-Farber Cancer Institute, Cambridge, Massachusetts, USA. [4]Department of Biostatistics, Harvard T.H. Chan School of Public Health, Cambridge, Massachusetts, USA. [5]Computational Biology Department, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA. Correspondence should be addressed to R.P. (rob.patro@cs.stonybrook.edu) or C.K. (carlk@cs.cmu.edu).

RECEIVED 29 AUGUST 2016; ACCEPTED 22 JANUARY 2017; PUBLISHED ONLINE 6 MARCH 2017; DOI:10.1038/NMETH.4197



c

Sensitivity at given FDR				
FDR	Salmon	Salmon (a)	kallisto	eXpress
0.01	0.326	0.233	0.072	0.128
0.05	0.409	0.379	0.248	0.162
0.1	0.454	0.442	0.296	0.211

d

Type	Salmon	Salmon (a)
All transcripts	1,183	1,197
Two-isoform genes	228	244
Type	**kallisto**	**eXpress**
All transcripts	2,620	2,472
Two-isoform genes	545	531

Figure 1 | Performance of Salmon. (**a**) The median of absolute log-transformed fold changes (lfc) between the estimated and true transcript abundances under all 16 replicates of the Polyester simulated data. An lfc closer to 0 indicates more similar true and estimated abundances. The two halves of the plot show the distributions of each method's lfc values for samples simulated with different GC-bias curves that were learned from experimental data using alpine[5] (Online Methods). In the box plot, the middle line of each box is the median, the box has width of the interquartile range (IQR), and the whiskers extend to the farthest observation still less than 1.5 times the IQR; all points are plotted for clarity. (**b**) The distribution of mean absolute relative differences (MARDs; Online Methods) of Salmon, Salmon using traditional alignments ("Salmon (a)"), kallisto, and eXpress under 20 simulated replicates generated by RSEM-sim. (**c**) The sensitivity of Salmon, Salmon using traditional alignments ("Salmon (a)"), kallisto, and eXpress in finding truly differentially expressed (DE) transcripts at typical FDR values for Polyester simulated data. (**d**) For 30 GEUVADIS samples, the number of transcripts called as DE at an expected FDR of 1% when the contrast between groups (i.e., the center at which they were sequenced) is a technical confound and not related to biology.

the same false discovery rates (FDRs) (**Fig. 1c**), achieving 53% to 450% higher sensitivity. Likewise, Salmon produced fewer than half as many DE calls as the other methods in comparisons that were expected to contain few or no true differences in transcript expression (**Fig. 1d**). Permuting samples or testing for DE within sequencing center among these samples resulted in an average of <<1 transcript called as DE for all methods. Salmon's benefits persisted at the gene level as well, where the use of Salmon's estimates for gene-level DE analysis led to a decrease by a factor of ~2.6 in the number of genes that were called as DE (**Supplementary Table 1**). **Supplementary Figure 5** shows specific examples from the GEUVADIS experiments where dominant isoform switching was observed ($P < 1 \times 10^{-6}$) between samples under the quantification estimates produced by kallisto or eXpress but this isoform switching was eliminated under Salmon's GC-bias-aware abundance estimates. **Supplementary Table 2** provides the rate of isoform switching for each of the different methods, underscoring that Salmon reduces this rate across various thresholds and for various categories of genes. In idealized simulations, like those generated by RSEM-sim, where realistic biases are not simulated, the accuracy of the different methods tended to be more similar (**Fig. 1b** and **Supplementary Fig. 6**), although we found that Salmon's distribution of mean absolute relative differences (MARDs) was significantly smaller (Mann–Whitney *U* test, $P = 0.00017$) than those of kallisto, whereas both methods outperformed eXpress (Mann–Whitney *U* test, $P = 3.39781 \times 10^{-8}$). These idealized simulations, where the fragments were generated without bias and in perfect accordance with the generative model adopted by the quantifiers, serve as a useful measure of the internal consistency of the algorithms[10,14]. However, we expect results on the SEQC[12], GEUVADIS[11], and Polyester[13] data sets (simulated with bias) to be more representative of typical real-world performance. On the RSEM-sim data, we evaluated all methods without bias correction. On all other data, we enabled bias correction for all methods. Additionally, on the Polyester simulated data, we enabled --noBiasLengthThreshold (to allow correction of even very short transcripts) for Salmon, as we were interested in assessing the maximum sensitivity of the model and the simulator produces fragments that are well behaved with respect to very short transcripts (see the Online Methods for details).

Salmon's rich model accounts for the effects of sample-specific parameters and biases that are typical of RNA–seq data, including positional biases in coverage, sequence-specific biases at the 5′ and 3′ ends of sequenced fragments, fragment-level GC bias, strand-specific protocols, and the fragment length distribution. These parameters are automatically learned in the online phase of the algorithm, which also estimates the conditional probability of a fragment being generated from each transcript to which it maps, as assessed by a general fragment–transcript agreement model (Online Methods, "Fragment–transcript agreement model"). This provides considerable information beyond simple fragment–transcript compatibility. Salmon incorporates these parameters by learning auxiliary models that describe the relevant distributions and by maintaining 'rich' equivalence classes of fragments (Online Methods, "Equivalence classes").

Salmon encompasses both alignment and quantification in a single tool. When run in quasi-mapping mode, Salmon takes as input an index of the transcriptome and a set of raw sequencing reads (i.e., unaligned reads in FASTA/Q format) and performs quantification directly, without generating any intermediate alignment files. This saves considerable time, as quasi-mapping is much faster than traditional alignment, and substantial memory and hard disk space because Salmon avoids writing and reading large alignment files.

Salmon is designed to take advantage of multiple CPU cores, and the mapping and inference procedures scale well with the number of reads in an experiment. Salmon can quantify abundance either via a built-in, ultra-fast read mapping procedure (quasi-mapping)[17] or using precomputed alignments provided in SAM or BAM format. Salmon can quantify an experiment of approximately 600 million reads (75 bp, paired end) in ≈23 min (1,384 s) using 30 threads—this roughly matches the speed of the recently introduced kallisto, which takes ≈20 min (1,198 s) to complete the same task using the same number of threads on the same machine (a 24-core machine with hyperthreading and 256 GB of RAM; each core was Intel Xeon CPU E5-4607 v2 2.60 GHz).

Salmon's approach is unique in how it combines models of experimental data and bias with an efficient dual-phase inference procedure. Salmon's ability to compute high-quality estimates of transcript abundances at the scale of thousands of samples while also accounting for the prevalent technical biases affecting transcript quantification[18] will enable individual expression experiments to be interpreted in the context of many rapidly growing sequence expression databases. This will allow for a more

comprehensive comparison of the similarity of experiments across large populations of individuals and across different environmental conditions and cell types. Salmon is open source and freely licensed (GPLv3). It is written in C++11 and is available at https://github.com/COMBINE-lab/Salmon.

METHODS

Methods, including statements of data availability and any associated accession codes and references, are available in the online version of the paper.

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

ACKNOWLEDGMENTS

We wish to thank those who have been using and providing feedback on Salmon since early in its (open) development cycle. The software has been greatly improved in many ways based on their feedback. This research is funded in part by the Gordon and Betty Moore Foundation's Data-Driven Discovery Initiative through Grant GBMF4554 to C.K. It is partially funded by the US National Science Foundation (CCF-1256087, CCF-1319998, BBSRC-NSF/BIO-1564917) and the US National Institutes of Health (R21HG006913, R01HG007104). C.K. received support as an Alfred P. Sloan Research Fellow. This work was partially completed while G.D. was a postdoctoral fellow in the Computational Biology Department at Carnegie Mellon University. M.I.L. was supported by NIH grant 5T32CA009337-35. R.A.I. was supported by NIH R01 grant HG005220.

AUTHOR CONTRIBUTIONS

R.P. and C.K. designed the method, which was implemented by R.P. R.P., G.D., M.I.L., R.I., and C.K. designed the experiments, and R.P., G.D., and M.I.L. conducted the experiments. R.P., G.D., M.I.L., R.A.I., and C.K. wrote the manuscript.

COMPETING FINANCIAL INTERESTS

The authors declare no competing financial interests.

1. Hoadley, K.A. *et al. Cell* **158**, 929–944 (2014).
2. Li, J.J., Huang, H., Bickel, P.J. & Brenner, S.E. *Genome Res.* **24**, 1086–1101 (2014).
3. Weinstein, J.N. *et al. Nat. Genet.* **45**, 1113–1120 (2013).
4. Roberts, A., Trapnell, C., Donaghey, J., Rinn, J.L. & Pachter, L. *Genome Biol.* **12**, R22 (2011).
5. Love, M.I., Hogenesch, J.B. & Irizarry, R.A. *Nat. Biotechnol.* **34**, 1287–1291 (2016).
6. Morán, I. *et al. Cell Metab.* **16**, 435–448 (2012).
7. Teng, M. *et al. Genome Biol.* **17**, 74 (2016).
8. Kodama, Y., Shumway, M. & Leinonen, R. *Nucleic Acids Res.* **40**, D54–D56 (2012).
9. Patro, R., Mount, S.M. & Kingsford, C. *Nat. Biotechnol.* **32**, 462–464 (2014).
10. Bray, N.L., Pimentel, H., Melsted, P. & Pachter, L. *Nat. Biotechnol.* **34**, 525–527 (2016).
11. Lappalainen, T. *et al. Nature* **501**, 506–511 (2013).
12. SEQC/MAQ-III Consortium. *Nat. Biotechnol.* **32**, 903–914 (2014).
13. Frazee, A.C., Jaffe, A.E., Langmead, B. & Leek, J.T. *Bioinformatics* **31**, 2778–2784 (2015).
14. Li, B., Ruotti, V., Stewart, R.M., Thomson, J.A. & Dewey, C.N. *Bioinformatics* **26**, 493–500 (2010).
15. Roberts, A. & Pachter, L. *Nat. Methods* **10**, 71–73 (2013).
16. Langmead, B. & Salzberg, S.L. *Nat. Methods* **9**, 357–359 (2012).
17. Srivastava, A., Sarkar, H., Gupta, N. & Patro, R. *Bioinformatics* **32**, i192–i200 (2016).
18. t'Hoen, P.A. *et al. Nat. Biotechnol.* **31**, 1015–1022 (2013).

ONLINE METHODS

Objectives and models for abundance estimation. Our main goal is to quantify, given a known transcriptome $\mathcal{T}$ and a set of sequenced fragments $\mathcal{F}$, the relative abundance of each transcript in our input sample. This problem is challenging both statistically and computationally. The main statistical challenges arise from the need to resolve a complex and often very high-dimensional mixture model (i.e., estimating the relative abundances of the transcripts given the collection of ambiguously mapping sequenced fragments). The main computational challenges derive from the need to process data sets that commonly consist of tens of millions of fragments in conditions where each fragment might reasonably map to many different transcripts. We lay out below how we tackle these challenges, beginning with a description of our assumed generative model of the sequencing experiment, upon which we will perform inference to estimate transcript abundances.

Our approach in Salmon consists of three components: a lightweight mapping model; an online phase that estimates initial expression levels, auxiliary parameters, and foreground bias models and constructs equivalence classes over the input fragments; and an offline phase that refines these expression estimates. The online and offline phases together optimize the estimates of the transcript abundances.

The online phase uses a variant of stochastic, collapsed variational Bayesian inference[19]. The offline phase applies either a standard EM algorithm or a variational Bayesian EM algorithm[20] over a reduced representation of the data represented by the rich equivalence classes until a data-dependent convergence criterion is satisfied. An overview of our method is given in **Supplementary Figure 1**, and we describe each component in more detail below.

Here we use a vertical bar "|" to indicate that the fixed quantities that follow it are parameters that are used to calculate the probability. For the Bayesian objective, the notation implies conditioning on these random variables.

Generative process. Assume that, for a particular sequencing experiment, the underlying true transcriptome is given as $\mathcal{T} = \{(t_1,\ldots,t_M),(c_1,\ldots,c_M)\}$, where each t_i is the nucleotide sequence of some transcript (an isoform of some gene) and each c_i is the corresponding number of copies of t_i in the sample. Further, we denote the length of transcript t_i as ℓ_i and the effective length of transcript t_i as $\widetilde{\ell}_i$, as defined in equation (1).

We adopt a generative model of the sequencing experiment that dictates that, in the absence of experimental bias, library fragments are sampled proportional to $c_i \cdot \widetilde{\ell}_i$. That is, the probability of drawing a sequencing fragment from some position on a particular transcript t_i is proportional to the total fraction of all nucleotides in the sample that originate from a copy of t_i. This quantity is called the nucleotide fraction[14].

$$\eta_i = \frac{c_i \cdot \widetilde{\ell}_i}{\sum_{j=1}^{M} c_j \cdot \widetilde{\ell}_j}$$

The true nucleotide fractions $\boldsymbol{\eta}$, although not directly observable, would provide us with a way to measure the true relative abundance of each transcript in our sample. Specifically, if we normalize η_i by the effective transcript length $\widetilde{\ell}_i$, we obtain a quantity

$$\tau_i = \frac{\frac{\eta_i}{\widetilde{\ell}_i}}{\sum_{j=1}^{M} \frac{\eta_j}{\widetilde{\ell}_j}}$$

called the transcript fraction[14]. These $\boldsymbol{\tau}$ values can be used to immediately compute common measures of relative transcript abundance, such as transcripts per million (TPM). The TPM measure for a particular transcript is the number of copies of this transcript that we would expect to exist in a collection of 1 million transcripts, assuming that this collection had exactly the same distribution of abundances as our sample. The TPM for transcript t_i is given by $\mathrm{TPM}_i = \tau_i \cdot 10^6$. Of course, in a real sequencing experiment, there are numerous biases and sampling effects that may alter the above assumptions, and accounting for them is essential for accurate inference. Below we describe how Salmon accounts for 5′- and 3′-sequence-specific biases (which are not considered separately by kallisto) and fragment GC bias, which is modeled by neither kallisto nor eXpress.

Effective length. A transcript's effective length depends on the empirical fragment length distribution of the underlying sample and the length of the transcript. It accounts for the fact that the range of fragment sizes that can be sampled is limited near the ends of a transcript. Here 'fragments' refers to the (potentially size-selected) cDNA fragments of the underlying library, from the ends of which sequencing reads are generated. In paired-end data, the mapping positions of the reads can be used to infer the empirical distribution of fragment lengths in the underlying library, while the expected mean and s.d. of this distribution must be provided for single-end libraries. We compute effective transcript lengths using the approach of kallisto[10], which defines the effective length of a transcript t_i as

$$\widetilde{\ell}_i = \ell_i - \mu_d^{\ell_i} \tag{1}$$

where $\mu_d^{\ell_i}$ is the mean of the truncated empirical fragment length distribution. Specifically, let d be the empirical fragment length distribution and $\Pr\{X = x\}$ be the probability of drawing a fragment of length x under d, then

$$\mu_d^{\ell_i} = \sum_{j=1}^{\ell_i} j \cdot \Pr\{X = j\} / \sum_{k=1}^{\ell_i} \Pr\{X = k\}.$$

Given a collection of observations (raw sequenced fragments or alignments thereof) and a model similar to the one described above, there are numerous approaches to inferring the relative abundance of the transcripts in the target transcriptome $\mathcal{T}$. Here we describe two basic inference schemes, both available in Salmon, which are commonly used to perform inference in such a model. All of the results reported in the manuscript were computed using the maximum-likelihood objective (i.e., the EM algorithm) in the offline phase, which is the default in Salmon.

Maximum-likelihood objective. The first scheme takes a maximum-likelihood approach to solve for the quantities of interest. Specifically, if we assume that all fragments are generated

doi:10.1038/nmeth.4197

independently and we are given a vector of known nucleotide fractions $\boldsymbol{\eta}$, a binary matrix of transcript–fragment assignment $\boldsymbol{Z}$ where $z_{ij} = 1$ if fragment j is derived from transcript i, and the set of transcripts $\mathcal{T}$, we can write the probability of observing a set of sequenced fragments $\mathcal{F}$ as follows:

$$\Pr\left\{\mathcal{F}\,|\,\boldsymbol{\eta},\boldsymbol{Z},\mathcal{T}\right\} = \prod_{j=1}^{N}\Pr\left\{f_j\,|\,\boldsymbol{\eta},\boldsymbol{Z},\mathcal{T}\right\} = \prod_{j=1}^{N}\sum_{i=1}^{M}\Pr\{t_i\,|\,\boldsymbol{\eta}\}\cdot\Pr\{f_j|t_i,z_{ij}=1\} \quad (2)$$

where $|\mathcal{F}| = N$ is the number of sequenced fragments, $\Pr\{t_i|\boldsymbol{\eta}\}$ is the probability of selecting transcript t_i to generate some fragment given the nucleotide fraction $\boldsymbol{\eta}$, and $\Pr\{t_i|\boldsymbol{\eta}\} = \eta_i$.

We have that $\Pr\{f_j|t_i,z_{ij} = 1\}$ is the probability of generating fragment j given that it came from transcript i. We will use $\Pr\{f_j|t_i\}$ as shorthand for $\Pr\{f_j|t_i,z_{ij} = 1\}$ since $\Pr\{f_j|t_i,z_{ij} = 0\}$ is defined to be uniformly 0. The determination of $\Pr\{f_j|t_i\}$ is defined in further detail in the "Fragment–transcript agreement model" section of the Online Methods. The likelihood associated with this objective can be optimized using the EM algorithm as in ref. 14.

Bayesian objective. One can also take a Bayesian approach to transcript abundance inference as done in refs. 21,22. In this approach, rather than directly seeking maximum-likelihood estimates of the parameters of interest, we want to infer the posterior distribution of $\boldsymbol{\eta}$. In the notation of ref. 21, we wish to infer $\Pr\{\boldsymbol{\eta}|\mathcal{F},\mathcal{T}\}$, the posterior distribution of nucleotide fractions when given the transcriptome $\mathcal{T}$ and the observed fragments $\mathcal{F}$. This distribution can be written as

$$\Pr\{\boldsymbol{\eta}|\mathcal{F},\mathcal{T}\} \propto \sum_{Z\in\mathcal{Z}}\Pr\left\{\mathcal{F}|\mathcal{T},\boldsymbol{Z}\right\}\cdot\Pr\{\boldsymbol{Z}|\boldsymbol{\eta}\}\cdot\Pr\{\boldsymbol{\eta}\} \quad (3)$$

where

$$\Pr\{\boldsymbol{Z}|\boldsymbol{\eta}\} = \prod_{i=1}^{M}\prod_{j=1}^{N}\eta_i^{z_{ji}} \quad (4)$$

and

$$\Pr\left\{\mathcal{F}|\mathcal{T},\boldsymbol{Z}\right\} = \prod_{i=1}^{M}\prod_{j=1}^{N}\Pr\left\{f_j\,|\,t_i\right\}^{z_{ji}} \quad (5)$$

Unfortunately, direct inference on the distribution $\Pr\{\boldsymbol{\eta}|\mathcal{F},\mathcal{T}\}$ is intractable because its evaluation requires summation over the latent variable configuration space that is exponentially large (i.e., all $\boldsymbol{Z} \in \mathcal{Z}$). Since the posterior distribution cannot be directly estimated, we must rely on some form of approximate inference. One particularly attractive approach is to apply variational Bayesian (VB) inference, in which some tractable approximation to the posterior distribution is assumed.

Subsequently, one seeks the parameters for the approximate posterior under which it best matches the true posterior. Essentially, this turns the inference problem into an optimization problem—finding the optimal set of parameters—which can be efficiently solved by several different algorithms. Variational inference seeks to find the parameters for the approximate posterior that minimizes the Kullback–Leibler (KL) divergence between the approximate and true posterior distributions. Although the true posterior may be intractable, this minimization can be achieved by maximizing a lower bound on the marginal likelihood of the posterior distribution[21], written in terms of the approximate posterior. When run with the VB objective, Salmon optimizes the collapsed variational Bayesian objective[21] in its online phase and the full variational Bayesian objective[22] in the variational Bayesian mode of its offline phase (see the "Offline phase" section in the Online Methods).

Fragment–transcript agreement model. Fragment–transcript assignment scores are defined as proportional to (i) the chance of observing a fragment length given a particular transcript, (ii) the chance that a fragment starts at a particular position on the transcript, (iii) the concordance of the fragment aligning with a user-defined (or automatically inferred) sequencing library format (for example, a paired-end, stranded protocol), and (iv) the chance that the fragment came from the transcript based on a score obtained from the alignment procedure (if alignments are being used). We model this agreement as a conditional probability $\Pr\{f_j|t_i\}$ for generating f_j given t_i. This probability, in turn, depends on auxiliary models whose parameters do not explicitly depend upon the current estimates of transcript abundance. Thus, once the parameters of these models have been learned and are fixed, these terms do not change even when the estimate for $\Pr\{t_i|\boldsymbol{\eta}\} = \eta_i$ needs to be updated. Salmon uses the following auxiliary terms

$$\Pr\{f_j|t_i\} = \Pr\{\ell\,|\,t_i\}\cdot\Pr\{p|\,t_i,\ell\}\cdot\Pr\{o|\,t_i\}\cdot\Pr\{a\,|\,f_j,t_i,p,o,\ell\} \quad (6)$$

where $\Pr\{\ell|t_i\}$ is the probability of drawing a fragment of the inferred length ℓ given t_i, and it is evaluated based on an observed empirical fragment length distribution. $\Pr\{p|t_i, \ell\}$ is the probability of the fragment starting at position p on t_i and is a function of the transcript's length. $\Pr\{o|t_i\}$ is the probability of obtaining a fragment aligning (or mapping) with the given orientation to t_i. This is determined by the concordance of the fragment with the user-specified library format. It is 1 if the alignment agrees with the library format and a user-defined prior value otherwise. Finally, $\Pr\{a|f_j, t_i, p, o, \ell\}$ is the probability of generating alignment a of fragment f_j given that it is drawn from t_i with orientation o starting at position p and is of length ℓ; this term is set to 1 when using quasi-mapping and is given by equation (7) for traditional alignments. The parameters for all auxiliary models are learned during the streaming phase of the inference algorithm from the first N' observations (5,000,000 by default). These auxiliary terms can then be applied to all subsequent observations.

Sequence-specific bias. It has been previously observed that the sequence surrounding the 5′ and 3′ ends of RNA–seq fragments influences the likelihood that these fragments are selected for sequencing[4]. If not accounted for, these biases can have a substantial effect on abundance estimates and can confound downstream analyses. To learn and correct for such biases, Salmon adopts a modification of the model introduced by Roberts *et al.*[4]. A (foreground) variable-length Markov model (VLMM) is trained on sequence windows surrounding the 5′ ($b_{s+}^{5'}$) and 3′ ($b_{s+}^{3'}$) read start positions. Then, a different (background) VLMM is trained on sequence windows drawn uniformly across known transcripts. Each is weighted by that transcript's abundance; the 5′ and 3′ background models are denoted as $b_{s-}^{5'}$ and $b_{s-}^{3'}$, respectively.

Fragment GC bias. In addition to the sequence surrounding the 5′ and 3′ ends of a fragment, it has also been observed that the GC content of the entire fragment can play a substantial role in the likelihood that it will be selected for sequencing[5]. These biases are largely different from sequence-specific biases, and thus accounting for both the context surrounding the fragments and the GC content of the fragments themselves is important when one wishes to learn and correct for some of the most prevalent types of bias *in silico*. To account for fragment GC bias, Salmon learns a foreground and background model of this fragment GC bias (and defines the bias as the ratio of the score of a fragment under each). Our fragment-GC-bias model consists of the observed distribution of sequenced fragments for every possible GC-content value (in practice, we discretize GC content and maintain a distribution over 25 bins, for fragments with GC content ranging from 0 to 1 in increments of 0.04). The background model is trained on all possible fragments (drawn uniformly and according to the empirical fragment length distribution) across known transcripts, with each fragment weighted by that transcript's abundance. The foreground and background fragment-GC-bias models are denoted as $b_{gc}+$ and $b_{gc}-$ respectively. Additionally, we note that sequence-specific and fragment GC biases do seem to display a conditional dependence. To account for this, Salmon learns (by default) three different bias models, each conditioned on the average GC content of the 5′ and 3′ sequence context of the fragment. A separate model is trained and applied for fragments with average 5′ and 3′ context GC contents from 0–0.33, 0.33–0.66, and 0.66–1. The performance of the model appears robust to the number of conditional GC models used (**Supplementary Fig. 7**).

Incorporating the bias models. These bias models are used to re-estimate the effective length of each transcript, such that a transcript's effective length now also takes into account the likelihood of sampling each possible fragment that transcript can produce—an approach to account for bias first introduced by Roberts *et al.*[4]. Before learning the bias-corrected effective lengths, the offline optimization algorithm is run for a small number of rounds (10 by default) to produce estimated abundances that are used when learning the background distributions for the various bias models. For a transcript t_i, the effective length becomes

$$\widetilde{\ell}'_i = \sum_{j=1}^{\ell_i} \sum_{k=1}^{f_i(j,L)} \frac{b_{gc^+}(t_i, j, j+k)}{b_{gc^-}(t_i, j, j+k)} \cdot \frac{b_{s^+}^{5'}(t_i, j)}{b_{s^-}^{5'}(t_i, j)} \cdot \frac{b_{s^+}^{3'}(t_i, j+k)}{b_{s^-}^{3'}(t_i, j+k)} \cdot \Pr\{X = j\}$$

where $\Pr\{X = j\}$ is the probability, under the empirical fragment length distribution, of observing a fragment of length j, L is the maximum observed fragment length, $f_i(j,L) = \min(\ell_i - j + 1, L)$, $b_{s^+}^{5'}(t_i, j)$ is the score given to the jth position for transcript t_i under the foreground 5′-sequence-specific bias model ($b_{s^-}^{5'}(t_i, j)$, $b_{s^+}^{3'}(t_i, j)$, and $b_{s^-}^{3'}(t_i, j)$ are defined similarly), and $b_{gc^+}(t_i, j, j+k)$ is the score given by the foreground fragment-GC-content model for the sequence of transcript t_i from positions j to $j + k$ (and similarly for $b_{gc^-}(t_i, j, j+k)$).

Once these bias-corrected lengths have been computed, they are used in all subsequent rounds of the offline inference phase (i.e., until the estimates of $\boldsymbol{\alpha}$, as defined in the "Algorithms" section of the Online Methods, converge). Typically, the extra computational cost required to apply bias correction is rather small, and the learning and application of these bias weights is parallelized in Salmon (for example, when processing sample ERR188021 of the GEUVADIS data set using 16 threads, the full fidelity bias modeling added 2 min (119 s) to the non-bias-corrected quantification time). Moreover, since the bias model works by evaluating the bias along bases of the reference transcriptome, it scales in the number of active (i.e., expressed) transcripts rather than the number of reads, so the extra cost of bias modeling is almost entirely independent of the number of fragments in an experiment. However, both the memory and time requirements of bias correction can be adjusted by the user to trade off time and space with model fidelity. To make the computation of GC fractions efficient for arbitrary fragments from the transcriptome, Salmon computes and stores the cumulative GC count for each transcript. To reduce memory consumption, this cumulative count can be sampled using the -- gcSizeSamp option to Salmon. This will increase the time required to compute the GC fraction for each fragment by a constant factor. Similarly, when attempting to determine the effective length of a transcript, Salmon will evaluate the contribution of all fragments longer than the shortest 0.5% and shorter than the longest 0.5% of the full empirical fragment length distribution that could be derived from this transcript. The program option --biasSpeedSamp will instead sample fragment lengths at a user-defined factor, speeding up the computation of bias-corrected effective lengths by this factor but coarsening the model in the process. However, sampling the fragment length distribution tends to produce substantial speed improvements with only a very moderate effect on model fidelity. For example, setting biasSpeedSamp to 5 reduces the additional bias correction time on sample ERR188021 mentioned above from 119 to 20 s, yet it leads to only a marginal increase in the number of false positive calls on the GEUVADIS data (from 1,183 to 1,190 total transcripts and from 228 to 231 transcripts from two-isoform genes; it decreased the number of false positive calls at the gene level slightly from 455 to 451). All results reported in this manuscript where bias correction was included were run without either of these sampling options (i.e., using the full-fidelity model).

Alignment model. When Salmon is given read alignments as input, it can learn and apply a model of read alignments to help assess the probability that a fragment originated from a particular locus. Specifically, Salmon's alignment model is a spatially varying first-order Markov model over the set of CIGAR symbols and nucleotides. To account for the fact that substitution and indel rates can vary spatially over the length of a read, we partition each read into a fixed number of bins and learn a separate model for each of these bins. This allows us to learn effects that vary spatially without making the model itself too large (as if, for example, we had attempted to learn a separate model for each position in the read). We choose six bins by default since this allows multiple different models for the beginning, middle, and trailing segments of a read, which tend to display distinct error profiles. However, we note that even a single, spatially homogeneous error profile appears to work reasonably well[22] and that, given sufficient training data, it is even possible to learn a separate bin for each position of a read[15]. Given the CIGAR string $s = s_0, \ldots, s_{|s|}$ for an alignment a, we compute the probability of a as

$$\Pr\{a | f_j, t_i, p, o, \ell\} = \Pr\{s_o\} \prod_{k=1}^{|s|} \Pr_{(\mathcal{M}_k)}\{s_{k-1} \to s_k | f_j, t_i, p, o, \ell\} \quad (7)$$

doi:10.1038/nmeth.4197

where $\Pr\{s_0\}$ is the start probability and $\Pr_{(M_k)}\{\cdot\}$ is the transition probability under the model at the kth position of the read (i.e., in the bin corresponding to position k). To compute these probabilities, Salmon parses the CIGAR string s, moves appropriately along both fragment f_j and reference transcript t_i, and computes the probability of transitioning to the next observed state in the alignment (a tuple consisting of the CIGAR operation and the nucleotides in the fragment and reference) given the current state of the model. The parameters of this Markov model are learned from sampled alignments in the online phase of the algorithm (**Supplementary Algorithm 1**). When quasi-mapping is used instead of user-provided alignments, the probability of the alignment is not taken into account (i.e., $\Pr\{a|f_j, t_i, p, o, \ell\}$ is set to 1 for each mapping).

Algorithms. We describe the online and offline inference algorithms of Salmon, which together optimize the estimates of $\boldsymbol{\alpha}$, a vector of the estimated number of reads originating from each transcript. Given $\boldsymbol{\alpha}$, $\boldsymbol{\eta}$ can be directly computed. An overview of the Salmon execution timeline, which describes when different estimates are made and quantities of interest are computed during the execution of the algorithm, is given in **Supplementary Figure 1**.

Online phase. The online phase of Salmon attempts to solve the variational Bayesian inference problem described in the "Objectives and models" section for abundance estimation and optimizes a collapsed variational objective function[21] using a variant of stochastic collapsed variational Bayesian inference[19]. The inference procedure is a streaming algorithm, similar to that of ref. 15, but it updates estimated read counts $\boldsymbol{\alpha}$ after every small group B^k (called a mini-batch) of observations, and the processing of mini-batches is done asynchronously and in parallel. The pseudo-code for the algorithm is given in **Supplementary Algorithm 1**.

The observation weight v^k for mini-batch B^k, in line 15 of **Supplementary Algorithm 1**, is an increasing sequence in k and is set, as in ref. 15, to adhere to the Robbins–Monroe conditions. Here the $\boldsymbol{\alpha}$ values represent the (weighted) estimated counts of fragments originating from each transcript. Using this method, the expected value of $\boldsymbol{\eta}$ can be computed directly using equation (16). We employ a weak Dirichlet conjugate prior with $\alpha_i^0 = 0.001 \cdot \widetilde{\ell_i}$ for all $t_i \in \mathcal{T}$. As outlined in ref. 19, the SCVB0 inference algorithm is essentially equivalent to variants of the online-EM[23] algorithm with a modified prior. The procedure in **Supplementary Algorithm 1** is run independently by as many worker threads as the user has specified. The threads share a single work queue upon which a parsing thread places mini-batches of alignment groups. An alignment group is simply the collection of all alignments (i.e., all multi-mapping locations) for a read. The mini-batch itself consists of a collection of some small, fixed number of alignment groups (1,000 by default). Each worker thread processes one alignment group at a time, using the current weights of each transcript and the current auxiliary parameters to estimate the probability that a read came from each potential transcript of origin. The processing of mini-batches occurs in parallel, such that very little synchronization is required, only an atomic compare-and-swap loop to update the global transcript weights at the end of processing of each mini-batch—hence, the moniker laissez-faire in the label of **Supplementary Algorithm 1**. This lack of synchronization means that, when estimating x_y, we cannot be certain that the most up-to-date values of $\boldsymbol{\alpha}$ are being used. However, owing to the stochastic and additive nature of the updates, this has little to no detrimental effect[24]. The inference procedure itself is generic over the type of alignments being processed; they may be either regular alignments (for example, coming from a BAM file) or quasi-mappings computed from the raw reads (for example, coming from FASTA/Q files). After the entire mini-batch has been processed, the global weights for each transcript are updated. These updates are sparse: that is, only transcripts that appeared in some alignment in mini-batch B^k will have their global weight updated after B^k has been processed. This ensures, as in ref. 15, that updates to the $\boldsymbol{\alpha}$ parameters can be performed efficiently.

Equivalence classes. During its online phase, in addition to performing streaming inference of transcript abundances, Salmon also constructs a highly reduced representation of the sequencing experiment. Specifically, Salmon constructs 'rich' equivalence classes over all the sequenced fragments. Collapsing fragments into equivalence classes is a well-established idea in the transcript quantification literature, and numerous different notions of equivalence classes have been previously introduced and shown to greatly reduce the time required to perform iterative optimization such as that described in the "Offline phase" section of the Online Methods. For example, Salzman *et al.*[25] suggested factorizing the likelihood function to speed up inference by collapsing fragments that align to the same exons or exon junctions (as determined by a provided annotation) into equivalence classes. Similarly, Nicolae *et al.*[26] used equivalence classes over fragments to reduce memory usage and speed up inference—they define as equivalent any pair of fragments that align to the same set of transcripts and whose compatibility weights (i.e., conditional probabilities) with respect to those transcripts are proportional. Turro *et al.*[27] introduced a notion of equivalence classes that considers as equivalent any pair of fragments (sequenced reads and read pairs) that multimap to the same set of target transcripts. The model of Turro *et al.* does not have the same restriction as that of Nicolae *et al.* on the proportional conditional probabilities of the equivalent fragments. Patro *et al.*[9] define equivalence classes over k-mers, treating as equivalent any k-mers that appear in the same set of transcripts at the same frequency, and use this factorization of the likelihood function to speed up optimization. Bray *et al.*[10] define equivalence classes over fragments and define as equivalent any fragments that pseudoalign to the same set of transcripts.

To compute equivalence classes, we define an equivalence relation $\sim$ over fragments. Let $A(\mathcal{T}, f_x)$ denote the set of quasi-mappings (or alignments) of f_x to the transcriptome $\mathcal{T}$, and let $M(f_x) = \{t_i \mid (t_i, p_i, o_i) \in A(\mathcal{T}, f_x)\}$ be the set of transcripts to which f_x maps according to $A(\mathcal{T}, f_x)$. We say $f_x \sim f_y$ if and only if $M(f_x) = M(f_y)$. Fragments that are equivalent are grouped together for inference. Salmon builds up a set of fragment-level equivalence classes by maintaining an efficient concurrent cuckoo hash map[28]. To construct this map, we associate each fragment f_x with $\boldsymbol{t}^x = M(f_x)$, which we will call the label of the fragment. Then, we query the hash map for $\boldsymbol{t}^x$. If this key is not in the map, we create a new

equivalence class with this label and set its count to 1. Otherwise, we increment the count of the equivalence class that we find in the map with this label. The efficient, concurrent nature of the data structure means that many threads can simultaneously query and write to the map while encountering very little contention. Each key in the hash map is associated with a value that we call a rich equivalence class. For each equivalence class $\mathcal{C}^j$, we retain a count $d^j = |\mathcal{C}^j|$, which is the total number of fragments contained within this class. We also maintain, for each class, a weight vector w^j. The entries of this vector are in one-to-one correspondence with transcripts i in the label of this equivalence class.

$$w_i^j = \frac{\sum_{f \in \mathcal{C}^j} \Pr\{f|t_i\}}{\sum_{t_k \in \boldsymbol{t}^j} \sum_{f \in \mathcal{C}^j} \Pr\{f|t_k\}} \tag{8}$$

That is, w_i^j is the average conditional probability of observing a fragment from $\mathcal{C}^j$ given t_i over all fragments in this equivalence class. Although the likelihood function over equivalence classes that considers these weights (equation (10)) is no longer exactly equivalent to the likelihood defined over all fragments (equation (9)), these weights nonetheless allow us to take into consideration the conditional probabilities specified in the full model without having to continuously reconsider each of the fragments in $\mathcal{F}$. There is a spectrum of possible representations of rich equivalence classes. This spectrum spans from the notion adopted here, which collapses all conditional probabilities into a single aggregate scalar, to an approach that clusters together fragments based not only on the transcripts to which they match but also on the vector of normalized conditional probabilities for each of these transcripts. The former approach represents a more coarse-grained approximate factorization of the likelihood function, while the latter represents a more fine-grained approximation. We believe that studying how these different notions of equivalence classes affect the factorization of the likelihood function, and hence its optimization, is an interesting direction for future work.

Offline phase. In its offline phase, which follows the online phase, Salmon uses the rich equivalence classes learned during the online phase to refine the inference. Given the set $\mathcal{C}$ of rich equivalence classes of fragments, we can use an EM algorithm to optimize the likelihood of the parameters given the data. The abundances $\boldsymbol{\eta}$ can be computed directly from $\boldsymbol{\alpha}$, and we compute maximum-likelihood estimates of these parameters that represent the estimated counts (i.e., number of fragments) deriving from each transcript, where

$$\mathcal{L}\{\boldsymbol{\alpha} \,|\, \mathcal{F}, \boldsymbol{Z}, \mathcal{T}\} = \prod_{j=1}^{N} \sum_{i=1}^{M} \widehat{\eta_i} \Pr\left\{f_j \,|\, t_i\right\} \tag{9}$$

and $\widehat{\eta_i} = \frac{\alpha_i}{\sum_j \alpha_j}$. If we write this same likelihood in terms of the equivalence classes $\boldsymbol{\mathcal{C}}$, we have the following.

$$\mathcal{L}\{\boldsymbol{\alpha} \,|\, \mathcal{F}, \boldsymbol{Z}, \mathcal{T}\} \approx \prod_{\mathcal{C}^j \in \boldsymbol{\mathcal{C}}} \left(\sum_{t_i \in \boldsymbol{t}^j} \widehat{\eta_i} \, w_i^j \right)^{d^j} \tag{10}$$

This likelihood, and hence that represented in equation (9), can then be optimized by applying the following update equation iteratively.

$$\alpha_i^{u+1} = \sum_{\mathcal{C}^j \in \boldsymbol{\mathcal{C}}} d^j \left(\frac{\alpha_i^u w_i^j}{\sum_{t_k \in \boldsymbol{t}^j} \alpha_k^u w_k^j} \right) \tag{11}$$

We apply this update equation until the maximum relative difference in the $\boldsymbol{\alpha}$ parameters satisfies

$$\Delta(\alpha^u, \alpha^{u+1}) = \max \frac{|\alpha_i^u - \alpha_i^{u+1}|}{\alpha_i^{u+1}} < 1 \times 10^{-2} \tag{12}$$

for all $\alpha_i^{u+1} > 1 \times 10^{-8}$. Let $\boldsymbol{\alpha}'$ be the estimates after having achieved convergence. We can then estimate η_i by $\widehat{\eta_i}$.

$$\widehat{\eta_i} = \frac{\alpha_i'}{\sum_j \alpha_j'} \tag{13}$$

Variational Bayes optimization. Instead of the standard EM updates of equation (11), we can, optionally, perform variational Bayesian optimization by applying VBEM updates as in ref. 22 but adapted to be with respect to the equivalence classes

$$\alpha_i^{u+1} = \sum_{\mathcal{C}^j \in \boldsymbol{\mathcal{C}}} d^j \left(\frac{e^{\gamma_i^u} w_i^j}{\sum_{t_k \in \boldsymbol{t}^j} e^{\gamma_k^u} w_k^j} \right) \tag{14}$$

where

$$\gamma_i^u = \Psi(\alpha_i^0 + \alpha_i^u) - \Psi\left(\sum_{k=1}^{M} \alpha_k^0 + \alpha_k^u \right) \tag{15}$$

Here $\Psi(\cdot)$ is the digamma function, and, upon convergence of the parameters, we can obtain an estimate of the expected value of the posterior nucleotide fractions as

$$\mathbb{E}\{\eta_i\} = \frac{\alpha_i^0 + \alpha_i'}{\sum_j \alpha_j^0 + \alpha_j'} = \frac{\alpha_i^0 + \alpha_i'}{\hat{\alpha}^0 + N} \tag{16}$$

where $\hat{\alpha}^0 = \sum_{i=1}^{M} \alpha_i^0$. Variational Bayesian optimization in the offline phase of Salmon is selected by passing the --useVBOpt flag to the Salmon quant command.

Sampling from the posterior. After convergence of the parameter estimates has been achieved in the offline phase, it is possible to draw samples from the posterior distribution of transcript abundances using Gibbs sampling. Salmon's Gibbs sampler samples, in turn, from the transcript abundances given the fragment assignments and then reassigns the fragments within each equivalence class given these sampled abundances. To perform this Gibbs sampling, we adopt the model of Turro *et al.*[27] (details in **Supplementary Note 2**). Posterior samples can be drawn by passing the --numGibbsSamples option to Salmon with the argument determining the number of samples to save.

Additionally, inspired by kallisto[10], Salmon also provides the ability to draw bootstrap samples, which is an alternative way to

doi:10.1038/nmeth.4197

assign confidence to the estimates returned by the main inference algorithm. Bootstrap samples can be drawn by passing the --numBootstraps option to Salmon with the argument determining the number of bootstraps to perform. The bootstrap sampling process works by sampling (with replacement) counts for each equivalence class and then rerunning the offline inference procedure (either the EM or VBEM algorithm) for each bootstrap sample.

Validation. *Metrics for accuracy.* Throughout this paper, we use several different metrics to summarize the agreement of the estimated TPM for each transcript with the TPM computed from simulated counts. While most of these metrics are commonly used and self-explanatory, we here describe the computation of the mean absolute relative difference (MARD), which is less common than some of the other metrics.

The MARD is computed using the absolute relative difference ARD_i for each transcript i

$$\mathrm{ARD}_i = \begin{cases} 0 \text{ if } x_i = y_i = 0 \\ \dfrac{|x_i - y_i|}{x_i + y_i} \text{ otherwise} \end{cases} \quad (17)$$

where x_i is the true value of the TPM and y_i is the estimated TPM. The relative difference is bounded above by 1 and takes on a value of 0 whenever the prediction perfectly matches the truth. To compute the mean absolute relative difference, we simply take $\mathrm{MARD} = \frac{1}{M}\sum_{i=1}^{M} \mathrm{ARD}_i$. We note that Salmon and kallisto, by default, truncate very tiny expression values to 0. For example, any transcript estimated to produce $<1 \times 10^{-8}$ reads is assigned an estimated read count of 0 (which, likewise, affects the TPM estimates). However, eXpress does not perform such a truncation, and very small, nonzero values may have a negative effect on the MARD metric. To mitigate such effects, we first truncate to 0 all TPMs less than 0.01 before computing the MARDs.

Ground truth simulated data. To assess accuracy in a situation where the true expression levels are known, we generate synthetic data sets using both Polyester[13] and RSEM-sim[14].

RSEM-sim simulations. To generate data with RSEM-sim, we follow the procedure used in ref. 10. RSEM was run on sample NA12716_7 from the GEUVADIS RNA–seq data to learn model parameters and estimate true expression, and the learned model was then used to generate 20 different simulated data sets, each consisting of 30 million 75-bp paired-end reads.

Polyester simulations. In addition to the ability to generate reads, Polyester allows the simulation of experiments with differential transcript expression and biological variability. Thus, we can assess not only the accuracy of the resulting estimates but also how these estimates would perform in a typical downstream analysis task like differential expression testing.

The Polyester simulation of an RNA–seq experiment with empirically derived fragment GC bias was created as follows: the transcript abundance quantifications from RSEM run on NA12716_7 from the GEUVADIS RNA–seq data[11] were summed to the gene level using version 75 of the Ensembl gene annotation for GRCh37. Subsequently, whole-transcriptome simulation was carried out using Polyester. Abundance (TPMs) was allocated to isoforms within a gene randomly using the following rule: for genes with two isoforms, TPMs were either (i) split according to a flat Dirichlet distribution ($\boldsymbol{\alpha}$ = (1,1)) or (ii) attributed to a single isoform. The choice of (i) or (ii) was decided by a Bernoulli trial with probability 0.5. For genes with three or more isoforms, TPMs were either (i) split among three randomly chosen isoforms according to a flat Dirichlet distribution ($\boldsymbol{\alpha}$ = (1,1,1)) or (ii) attributed to a single isoform. Again, (i) or (ii) was decided by a Bernoulli trial with probability 0.5. The choice of distributing expression among three isoforms was motivated by exploratory data analysis of estimated transcript abundance revealing that nearly all of the expression for most genes was concentrated in the three most abundant isoforms for genes with four or more isoforms.

Expected counts for each transcript were then generated according to the transcript-level TPMs, multiplied by transcript lengths. Forty million 100-bp paired-end reads were simulated using the Polyester software for each of 16 samples, and 10% of transcripts were chosen to be differentially expressed across an 8 vs. 8 sample split. The fold change was chosen to be either 1/2 or 2 with a probability of 0.5. Fragments were downsampled with Bernoulli trials according to an empirically derived fragment GC content dependence estimated with alpine[5] on RNA–seq samples from the GEUVADIS project. The first eight GEUVADIS samples exhibited weak GC content dependence while the last eight samples exhibited more severe fragment-level GC bias. Paired-end fragments were then shuffled before being supplied to transcript abundance quantifiers. Estimated expression was compared to true expression calculated on transcript counts (before these counts were downsampled according to the empirically derived fragment GC bias curve), divided by effective transcript length, and scaled to TPM. Global differences across condition for all methods were removed using a scaling factor per condition. Differences across condition for the different methods' quantifications were tested using a *t*-test of $\log_2$ (TPM + 1).

Software versions and options. All tests were performed with eXpress v1.5.1, kallisto v0.43.0, Salmon v0.8.0, and Bowtie2 v2.2.4. Reads were aligned with Bowtie2 using the parameters --no-discordant -k 200 and -p to set the number of threads. On the RSEM-sim data, all methods were run without bias correction. On all other data sets, methods were run with bias correction unless otherwise noted. Additionally, on the Polyester simulated data, Salmon was run with the option --noBiasLengthThreshold, which allows bias correction, even for very short transcripts. This option was used because we were most interested in assessing the maximum sensitivity of the model.

GEUVADIS data. The analyses presented in **Figure 1d**, **Supplementary Table 1**, and **Supplementary Figure 5** were carried out on a subset of 30 samples from publicly available GEUVADIS[11] data. The accessions used and information about the center at which the libraries were prepared and sequenced are recorded in **Supplementary Table 3**. All methods were run with bias correction enabled, using a transcriptome built with both the RefSeq gene annotation file and the genome FASTA contained within the hg19 Illumina iGenome to allow for comparison with the results in ref. 5.

For each transcript, a *t*-test was performed, comparing $\log_2$ (TPM+1) from 15 samples from one sequencing center against

15 samples from another sequencing center. *P* values were then adjusted using the method of Benjamini–Hochberg, over the transcripts with mean TPM > 0.1. The number of positives for given false discovery rates was then reported for each method by taking the number of transcripts with an adjusted *P* value that is less than a given threshold.

Because the samples are from the same human population, it is expected that there would be few to no true differences in transcript abundance produced by this comparison. This assumption was confirmed by permuting the samples and performing *t*-tests as well as making *t*-test comparisons of random subsets within sequencing centers, which consistently produced <<1 DE transcript on average for all methods. Such an analysis—comparing samples across sequencing center—was specifically chosen to highlight transcripts with false quantification differences arising from technical artifacts.

SEQC data. The consistency analysis presented in **Supplementary Figure 4** was carried out on a subset of the publicly available SEQC[12] data. Specifically, the accessions used, along with corresponding information about the center at which they were sequenced, are recorded in **Supplementary Table 4**. For each sample, 'same center' comparisons were made between all unique pairs of replicates labeled as coming from the same sequencing center, while 'different center' comparisons were made between all unique pairs of replicates labeled as coming from different centers ("Center" column of **Supplementary Table 4**).

Statistics. Comparisons of RSEM-sim simulated data mean absolute relative differences (MARDs) and Spearman correlations had sample sizes $n_1 = 20$, $n_2 = 20$. A Mann–Whitney *U* test (two-sided) was performed. Switching of the dominant isoform on GEUVADIS data had sample sizes $n_1 = 15$, $n_2 = 15$. A *t*-test (two-sided) was performed. Polyester simulated data differential expression analysis had sample sizes $n_1 = 8$, $n_2 = 8$. A *t*-test (two-sided) was performed. FDR sets were defined using Benjamini–Hochberg multiple-test correction.

Salmon development, support, and validation. Salmon is developed openly on GitHub (https://github.com/COMBINE-lab/Salmon), which is the primary venue for users to make feature requests and to file bug reports. However, support is also provided via a Google Users Group (https://groups.google.com/forum/#!forum/Sailfish-users) and a gitter channel (https://gitter.im/COMBINE-lab/Salmon). This provides multiple venues for users to have their questions answered quickly and efficiently. Further, Salmon is available through both homebrew-science[29] and bioconda to ease installation and upgrading of the package.

Testing during the Salmon development and release process is highly automated. In addition to any major feature branches, the Salmon repository retains both master and develop branches. The master branch corresponds to the most recent tagged release of the software, while the develop branch is where new feature development takes place before it has been sufficiently well tested to be included in a tagged release. A publicly facing, continuous integration service (Travis-CI) automatically builds each commit, and it runs a small set of functionality tests to ensure that no breaking changes have been committed. However, these tests do not assess or track quantification accuracy. For this task, a parallel, self-hosted, continuous integration system has been created. On each commit to the Salmon repository, a Drone (Drone-CI) server pulls the latest commit and builds it in a clean CentOS5 environment using Docker[30]. In addition to the functionality tests, Salmon is run on simulated samples (generated using Polyester[13]). These tests (automated using Next Flow[31]) build the Salmon index, quantify all simulated samples, and store the resulting accuracy metrics in a JSON formatted file. The results are copied back from the Docker container to the host and are placed in a uniquely named directory that corresponds with the SHA1 hash of the commit that produced them. This allows us to track accuracy over various Salmon commits and to identify the commit corresponding to any performance regressions. We note that this setup overlaps considerably with the setup suggested for "continuous analysis" by Beaulieu-Jones and Greene[32]. Going forward, we anticipate expanding the test suite to include even more data and performance metrics.

Data availability. Accession information for experimental data used in this manuscript has been provided in the text and in **Supplementary Tables 3** and **4**. Data simulation has been carried out in accordance with the procedures detailed in the "Ground truth simulated data" section of the Online Methods.

The source code for Salmon is freely available and licensed under the GNU General Public License (GPLv3). The latest version of Salmon can be obtained from https://github.com/COMBINE-lab/salmon.

19. Foulds, J., Boyles, L., DuBois, C., Smyth, P. & Welling, M. in *Proc. 19th ACM SIGKDD Int. Conf. Knowledge Discov. & Data Mining* 446–454 (ACM, 2013).
20. Bishop, C.M. *et al. Pattern Recognition and Machine Learning* (Springer, 2006).
21. Hensman, J., Papastamoulis, P., Glaus, P., Honkela, A. & Rattray, M. *Bioinformatics* **31**, 3881–3889 (2015).
22. Nariai, N. *et al. BMC Genomics* **15** (Suppl. 10), S5 (2014).
23. Cappé, O. in *Mixtures: Estimation and Applications* (eds. Mengersen, K.L., Robert, C.P. & Titterington, D.M.) Ch. 2 (John Wiley & Sons, 2011).
24. Hsieh, C.-J., Yu, H.-F. & Dhillon, I.S. *ICML* **15**, 2370–2379 (2015).
25. Salzman, J., Jiang, H. & Wong, W.H. *Stat. Sci.* **26**, 1 (2011).
26. Nicolae, M., Mangul, S., Măndoiu, I.I. & Zelikovsky, A. *Algorithms Mol. Biol.* **6**, 9 (2011).
27. Turro, E. *et al. Genome Biol.* **12**, R13 (2011).
28. Li, X., David, G., Andersen, M.K. & Freedman, M.J. in *Proc. Ninth Eur. Conf. Computer Syst.* 27 (ACM, 2014).
29. Jackman, S. & Birol, I. *F1000Research* **5**, 1795 (2016).
30. Merkel, D. *Linux J.* **2014** (2014).
31. Di Tommaso, P., Chatzou, M., Baraja, P.P. & Notredame, C. *figshare* https://dx.doi.org/10.6084/m9.figshare.1254958.v2 (2014).
32. Brett, K.B.-J. & Greene, C.S. Preprint at https://doi.org/10.1101/056473 (2016).

doi:10.1038/nmeth.4197