# Fast and accurate quantification and differential analysis of transcriptomes

by

Harold Joseph Pimentel

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer science

and the Designated Emphasis

in

Computational and genomic biology

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Lior Pachter, Chair
Assistant Professor Nir Yosef
Associate Professor Haiyan Huang

Summer 2016

**Fast and accurate quantification and differential analysis of transcriptomes**

# Abstract

Fast and accurate quantification and differential analysis of transcriptomes

by

Harold Joseph Pimentel

Doctor of Philosophy in Computer science

University of California, Berkeley

Professor Lior Pachter, Chair

As access to DNA sequencing has become ubiquitous to scientists, the use of sequencers has expanded from determining the genomes of individuals to performing molecular probing assays. These assays have turned DNA sequencers into molecule counting machines and can be used to measure biological activities such as gene expression (RNA-Seq [60]), DNA accessibility (ATAC-Seq [12]) and many others [91].

Each new assay poses new analytical challenges, and the main focus presented here is in analyzing RNA-Seq data. One of the main challenges in RNA-Seq is that sequenced fragments are often ambiguous, meaning they are compatible with multiple splice forms or genomic locations. In order to estimate gene abundances effectively, these ambiguous fragments should be used in a comprehensive model in order to not bias results [87]. Analysis has come a long way from ignoring ambiguous mappings, to maximum likelihood models [87, 50, 49], and even streaming models [72]. Advancements in these models have greatly improved the accuracy of estimating gene and transcript abundances.

In parallel, methods for determining true expression differences between experimental conditions, termed differential expression, have been developed [4, 56, 75, 45]. Historically, these methods have mostly ignored advancements in gene expression estimation but have made much progress in between-sample variance estimation when sample sizes are small – a common practice in this field.

Herein, we present advancements to both abundance estimation and differential expression analysis. We show dramatic improvements to the speed of abundance estimation while maintaining accuracy. Furthermore, we bridge these two fields by developing a differential expression model incorporating the uncertainty introduced by abundance estimation. We show that this model outperforms existing techniques at both the transcript and gene level.

Additionally, we show that these methods can be used to address other biological questions such as the discovery of novel retained introns and estimation of their abundances. An extension to the differential expression model is proposed to identify differences in retained intron levels while incorporating abundance estimation uncertainty.

To Suzette and my parents.

# Contents

# List of Figures

# List of Tables

# Acknowledgments

Firstly, I would like to thank my advisor, Lior Pachter. You have been an inspirational mentor and collaborator. I am incredibly grateful for all of the countless discussions about science, math, ethics, and nothing at all. I can only hope to be as great of a mentor as you someday. I would also like to thank my qual and disseration committees: Haiyan Huang for guiding me through my MA and teaching me statistical rigor, Nir Yosef for though provoking methodology and application questions, and Doris Bachtrog for challenging my biological knowledge. Spiros Courellis for taking me aside as a clueless undergrad and encouraging me to go to grad school when I had no idea what a PhD really was. Amybeth Cohen for pushing me (and sometimes pulling me) through the MARC program.

Thanks to my "senior" labmates that helped guide and encourage me when I was lost: Adam Roberts, Atif Rahman, and Meromit Singer. And thanks to the rest of the Pachter lab for endless helpful discussions and collaborations: Lorian Schaeffer, Isaac Joseph, Shannon McCurdy, Brielin Brown, Shannon Hately, Akshay Tambe, and Bo Li. A special thanks to Nicolas Bray who always encouraged me to think about yet another edge case and always helped me work it out. Thanks to Páll Melsted for pushing me as an engineer and computer scienctist and leading by example. Thanks to all of the staff who helped along the way: Saheli Datta for helping me get my NSF fellowship back, Brian McClendon, Xuan Quach, La Shana Polaris, and Audrey Sillers for tirelessly helping me navigate the beuracracy. Thanks to John Conboy for patiently teaching me blood biology and what it really means to "look at the data."

Finally, I would like to thank my family. My parents for the endless sacrifices and support. They taught me my work ethic and continue to inspire me with their ongoing dedication. My incredible wife for being my first stop for almost everything. For comforting me when I was down, for sharing laughter and joy, and ultimately, for sharing life with me. None of this would not have been possible without her endless academic and moral support.

# Chapter 1

# Introduction

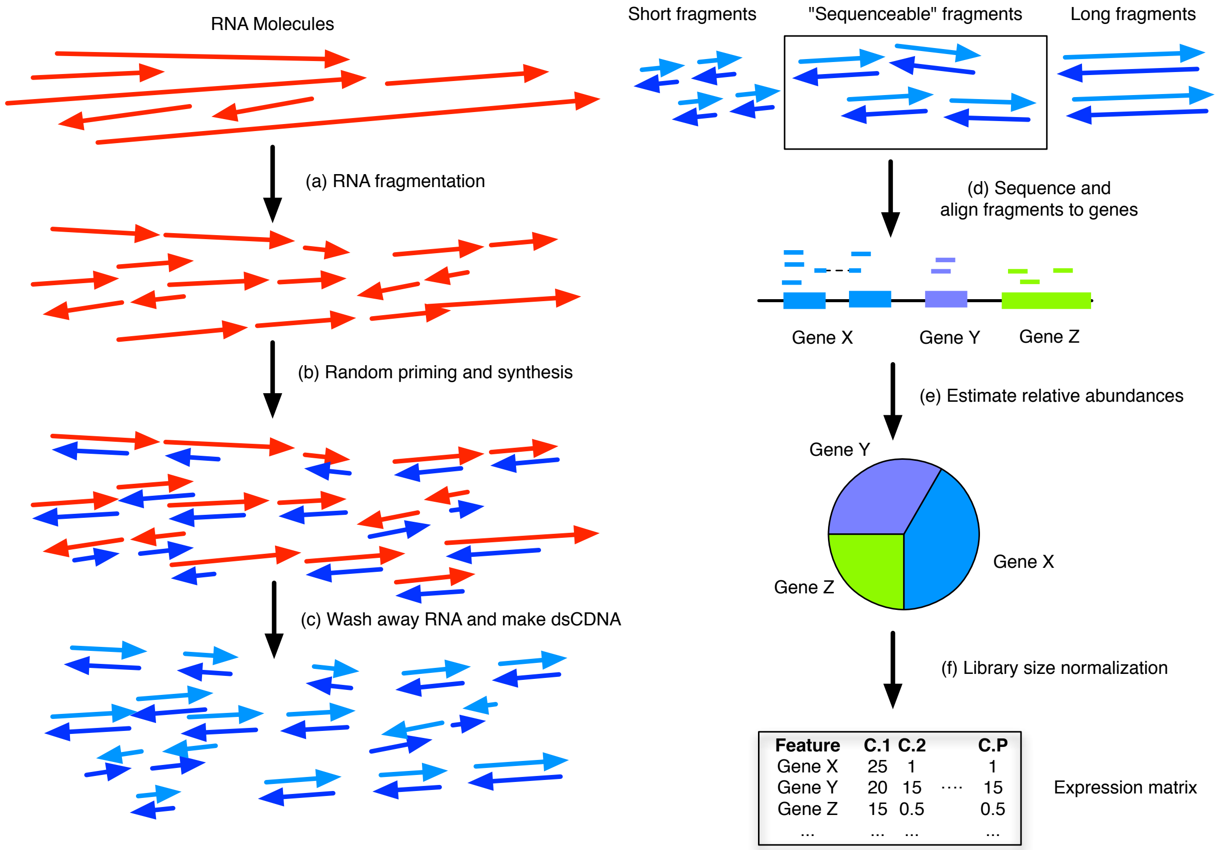
Incredible innovations have been made over the past ten years in DNA sequencing. The cost of DNA sequencing has dropped drastically, throughput has been greatly increased with improved accuracy, sample preparations have become much simpler, and above all, sequencing has become easily attainable even for small laboratories. Much of this innovation is a result of the Illumina sequencing platform which uses a massively parallel sequencing-by-synthesis method. This platform has the capability of sequencing billions of short reads commonly ranging from 75 base pairs to 300 base pairs. Here, we call standard short read DNA sequencing on the Illumina platform DNA-Seq to avoid confusion with other DNA sequencing platforms. While DNA-Seq is a great feat in its own right, perhaps even more incredible is the inevitable adaptation of the sample preparation to perform other assays such as detection of chromatin binding regions, methylation sites, and RNA abundance [91]. At a high level, if the molecule of interest can be translated into DNA, it can be measured using DNA sequencing by turning the sequencer into a molecular counter.

There have been a plethora of protocols released in the last decade built atop DNA sequencing [94, 57, 54, 34, 30, 37], but perhaps the most popular is RNA-Seq [60], the process of sequencing RNA fragments. There are many forms of RNA-Seq, but the most common is mRNA-Seq which sequences mature messenger RNA. The process can be broken down into a few major steps: isolate RNA molecules, fragment RNA molecules, perform reverse transcription to generate cDNA, amplify, and sequence (see Figure 1.1 for an overview) [88]. The output is typically a set of short reads that represent cDNA fragments derived from the mRNA. Reads are then output in large text files containing the sequences of the ends of cDNA fragments. There are other sequencing platforms that I will not discuss, most of which are tailored to perform long-range sequencing, but none of which have had nearly as many applications as the Illumia platform.

Post-sequencing, analysis of the resulting data is required which in itself poses many issues. Since the reads are not labeled, sophisticated algorithms have to be used in all steps of the analysis to recover the signal from the cDNA fragments. Thus, this data leads to a number of technical issues:

1. How is a read's true source transcript/gene identified?
2. How can the transcript abundances be estimated?
3. What types of questions can be answered with this data after the transcript abundances have been estimated?

These questions are the main focus of this thesis. Questions (1) and (2) are very related and I will call them "alignment and quantification". Question (3) is much more broad but I will address it with a few specific analyses that can be performed, in particular differential expression analysis and intron retention detection.



| Feature | C.1 | C.2 | | C.P |
|---|---|---|---|---|
| Gene X | 25 | 1 | | 1 |
| Gene Y | 20 | 15 | ···· | 15 |
| Gene Z | 15 | 0.5 | | 0.5 |
| ... | ... | ... | | ... |

Figure 1.1: Typical workflow of RNA-Seq adapted from [73]. (a) RNA molecules are isolated and fragmented. (b) Primers are used to initiate reverse transcription and create single-stranded cDNA. (c) RNA is removed and double-stranded cDNA is created. Typically, PCR amplification is performed at this step. It is also typical to perform a size selection step to remove fragments that are either too short or too long to sequence well. This step often selects fragments around 200 base pairs long. (d) Sequencing is performed and short reads are reported in plaintext files. These reads are aligned to a reference genome or transcriptome. (e) The reads are summarized by estimating the relative abundance of each gene or transcript. (f) After the above steps have been performed on several samples, the samples can be aggregated and compared to each other using between-sample normalization techniques [56, 74]. This is necessary because they will likely be sequenced to different depths and will likely have different relative distributions. After normalization, postprocessing and other analyses can proceed, such as differential expression.

## 1.1 Alignment and quantification

After reads have been generated, their origin should be determined in order to estimate the abundance of each transcript. The procedure to determine possible locations of origin is alignment. RNA-Seq alignment is very similar to DNA-Seq alignment [43] with some added complexities. One complexity is that the reads actually come from RNA transcripts resulting in reads that span exon junctions. There are two common ways to deal with this: (1) align directly to a reference transcriptome including only exonic sequences, or (2) align across splice junctions at the genomic level. Method 1 simplifies the matter as DNA-Seq aligners can be used [50]. However, a drawback of this is that new splice junctions cannot be discovered without considerable effort. Usually, this is not an issue if one is studying a model organism that is well annotated. Method 2 typically involves trying to map directly to the genome, then mapping the remaining reads across splice junctions. There have been many approaches, but it is quite typical to create indexes of known splice junctions, then to try to map to these indexes [86, 41]. Regardless, both of these methods are typically the most time consuming part of the workflow, often taking several hours for just one sample [11].

After alignment, many reads typically map ambiguously at both the transcript level due to alternative splicing and genome level due to duplications. Because of this fact, simple counting procedures can often give biased results of transcript or gene expression [85] leading to invalid inferences about the underlying biology. This is typically dealt with by probabilistic models that attempt to intelligently assign fragments based on a model of the sequencing procedure. This is quite a difficult problem and much attention has been given to it [72, 63, 87, 50, 49, 38]. In these models, after potential sites have been identified for each read, the true origin of the read is typically treated as an unobserved random variable and often solved for using the Expectation-Maximization algorithm [63].

In 2014, researchers realized that one could get away from read alignments by “shredding” reads into k-mers and aligning those quickly by using a hash table [66]. While this has speed advantages compared to typical alignment strategies, the spatial integrity of the reads is lost when ignoring the fact that k-mers are correlated. As a result, considerable accuracy is lost [11]. Our approach, kallisto (Chapter 2), improves accuracy by using information from all of the k-mers in a read at once rather than shredding them and also includes major speed improvements based on the way we perform the “pseudoalignment” procedure (discussed in Section 2.2).

## 1.2 Differential expression analysis

After the abundance of each transcript has been quantified in a set of RNA-Seq samples, a common question is: are there any features (e.g. transcripts or genes) expressed differently in different experimental conditions? Experimental conditions can be genetically different individuals, different experimental perturbations, cancer versus normal cells, etc. Classical statistics has tackled these types of questions for many years, however, with RNA-Seq data there are additional complications making these methods inappropriate off-the-shelf.

Perhaps the most common difficulty in finding differentially expressed features is the small sample size in most experiments. While it is true that sequencing is relatively inexpensive, the time and effort in producing samples for sequencing can be considerable. It is fairly common that an experiment contain as few as 3 control samples and as few as 3 treatment samples. Small samples typical in RNA-Seq studies violate the assumptions of classical statistics which typically assume data sizes approaching infinity.

The result of violating the large sample assumptions is that estimators of the biological variance which are required for differential expression testing are very unstable leading to potentially unreproducible results and many false-positives [77]. To address this issue there have been many methods for estimating the variance in a "pooled" manner, commonly referred to as "shrinkage" in the statistical literature [4, 56, 75, 45]. These methods typically differ in how they estimate the variance, often by making assumptions of a mean-variance relationship.

Most of this effort has been at the gene level with a number of assumptions that go into this modeling. The most common is that the variance has a specific form where the variance of estimating the abundance (called inferential variance) is constant with respect to the mean abundance [4]. In Chapter 3 we show that this assumption can be dropped leading to an increase in performance. When testing for differential expression at the transcript level, this assumption is even more highly violated and while some methods do exist [48, 85, 28], these methods have strong assumptions about the inferential variance which are certainly violated. Our method drops these strong parametric assumptions and learns this component of the variance by using the bootstrap as implemented in kallisto. Additionally, our method is generalized to testing arbitrary sets of transcripts and the transcript-level model is a special case of the gene-level model.

## 1.3 Intron retention

Another application of RNA-Seq is the abundance estimation and discovery of novel retained introns. Retained introns have been studied much less than differential expression, not only from a methods point of view, but also from a functional point of view [9, 10]. Even so, there have been numerous implications of inappropriate intron retention induced by mutations. For example, intron retention events have recently been implicated in breast cancer [25] and MDS [58]. Briefly, a retained intron typically occurs when an intron normally spliced out during transcription is retained in the final mRNA product. Assuming the library is poly-A selected, RNA-Seq can be used to detect these events. While some quantification tools support quantification of retained introns that are annotated [38, 1], no tools exist to perform discovery of these events. We present our work on discovering novel retained introns as well as finding differentially expressed introns in Chapter 4.

# Chapter 2

# Fast quantification with kallisto

We present kallisto, an RNA-seq quantification program that is two orders of magnitude faster than previous approaches and achieves similar accuracy. Kallisto pseudoaligns reads to a reference, producing a list of transcripts that are compatible with each read while avoiding alignment of individual bases. We use kallisto to analyze 30 million unaligned paired-end RNA-seq reads in $<$ 10 min on a standard laptop computer. This removes a major computational bottleneck in RNA-seq analysis.

This work was previously published as "Near-optimal probabilistic RNA-seq quantification" [11] and is being reproduced with permission of the co-authors.

## 2.1 Introduction

The first two steps in typical transcript-level RNA-seq processing workflows are alignment to a transcriptome or a reference genome and estimation of transcript abundances. These steps can be time consuming. For example, aligning 20 samples, each with 30 million RNA-seq reads, using the widely used program TopHat2 [41] takes 28 core hours on 20 cores, while quantification with the companion program Cufflinks [87] takes another 14 h. Such running times are likely to become prohibitive as sequence data from increasing numbers of samples are generated. Although the quantification of aligned reads can be sped up with streaming algorithms [72] or by naive counting of reads [5], these approaches have resulted in a decrease in quantification accuracy. To circumvent the alignment step, a recent study proposed quantifying samples by extraction of k-mers from reads followed by exact matching of the k-mers using a hash table [66]. However, shredding reads into k-mers discards valuable information present in complete reads since each k-mer can align to more transcripts than the read itself. This results in a substantial loss of accuracy (Fig. 2.1).

Although the direct use of k-mers is inadequate for accurate quantification, the hash-based approach provides a basis for speeding up RNA-seq processing. Here we investigate whether information from k-mers within reads can be combined to maintain the accuracy of alignment-based quantification. We examine the central difficulty and key requirement for

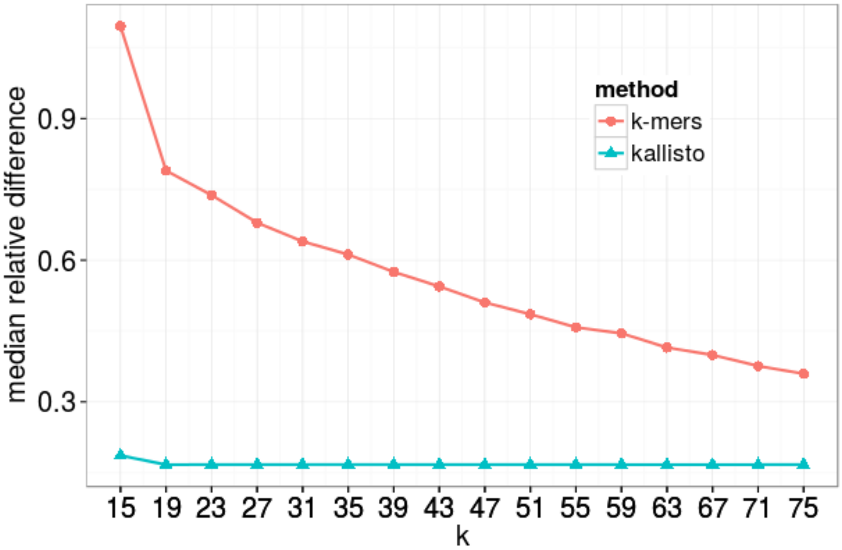


Figure 2.1: Median relative difference for abundance estimates using varying values of k on a dataset of 30 million 75bp paired-end reads that were simulated without errors. The "k-mers method" uses the k-compatibility of each k-mer independently and runs the EM algorithm on k-mers, whereas kallisto uses the intersection of k-compatibility classes across both ends of a read. Even for k = 75, the full read length in the simulation, independent use of k-mers results in a significant drop in accuracy due to the loss of paired-end information.

accurate quantification, which is the assignment of reads that cannot be uniquely aligned [60]. Typically, these multi-mapping reads are accounted for using a statistical model of RNA-seq [60] that probabilistically assigns such reads while inferring maximum likelihood estimates of transcript abundances. However, it has been shown that accurate quantification does not require information on where inside transcripts the reads may have originated from, but rather which transcripts could have generated them [61]. On the basis of this information, we develop a method based on pseudoalignment of reads and fragments, which focuses only on identifying the transcripts from which the reads could have originated and does not try to pinpoint exactly how the sequences of the reads and transcripts align.

## 2.2 Pseudoalignment

A pseudoalignment of a read to a set of transcripts, $T$, is a subset, $S \subseteq T$, without specific coordinates mapping each base in the read to specific positions in each of the transcripts in $S$. Accurate pseudoalignments of reads to a transcriptome can be obtained using fast hashing of k-mers together with the transcriptome de Bruijn graph (T-DBG). de Bruijn graphs have been crucial for DNA and RNA assembly [20], where they are usually constructed from reads. Kallisto uses a T-DBG, which is a de Bruijn graph constructed from k-mers present in the transcriptome (Fig. 2.2a), and a path covering of the graph, a set of paths whose union covers all edges of the graph, where the paths correspond to transcripts (Fig. 2.2b). This path covering of a T-DBG induces multi-sets on the vertices, called k-compatibility classes. A compatibility class can be associated to an error-free read by representing it as a path in the graph and defining the k-compatibility class of a path in the graph as the intersection of the k-compatibility classes of its constituent k-mers (Fig. 2.2c). An equivalence class for a read is a multi-set of transcripts associated with the read; ideally it represents the transcripts a read could have originated from and provides a sufficient statistic for quantification. A key point is that the k-compatibility class of an error-free read coincides with the minimal equivalence class consisting of transcripts containing the read for large k (Subsection 2.2).

Previously, the equivalence classes of reads have been determined via the time-consuming alignment of the reads to the transcriptome. However, since a hash of k-mers provides a fast way to determine their k-compatibility classes, the equivalence class of (error-free) reads can be efficiently determined by selecting suitably large k and then intersecting the reads' constituent k-compatibility classes. The difficulty of implementing such an approach for RNA-seq lies in the fact that reads have errors. However, with very high probability, an error in a k-mer will result in it not appearing in the transcriptome, and such k-mers are simply ignored. The issue of errors is also ameliorated by a technique that we implemented to improve the efficiency of pseudoalignment that removes redundant k-mers from the computation on the basis of information contained in the T-DBG (Subsection 2.2). Because fewer k-mers are inspected, there is less opportunity for erroneous k-mers to produce misleading results. With pseudoalignments efficiently computable, we explored the use of the expectation-maximization (EM) algorithm applied to equivalence classes for quantification [66] (Subsection 2.3). Although the likelihood function is simpler than some other models used for RNA-seq [87, 72, 49] it still includes a model for bias, and its use has the advantage that the EM algorithm can be applied for many rounds very rapidly.

## Index construction

The construction of the index starts with the formation of a colored de Bruijn graph [35] from the transcriptome, where the colors correspond to transcripts. In the colored transcriptome de Bruijn graph, each node corresponds to a k-mer and every k-mer receives a color for each transcript it occurs in. Contigs are defined to be linear stretches of the de Bruijn graph that have identical colorings. This ensures that all k-mers in a contig are associated with the same equivalence class (the converse is not true: two different contigs can be associated with the same equivalence class). Once the graph and contigs have been constructed, kallisto stores a hash table mapping each k-mer to the contig it is contained in, along with the position within the contig. This structure is called the kallisto index.

For error-free reads, there can be a difference between the equivalence class of a read and the intersection of its k-compatibility classes. But for a read of length $l$ this can only happen if there are two transcripts that have the same $l-k+1$ k-mers occurring in different order.

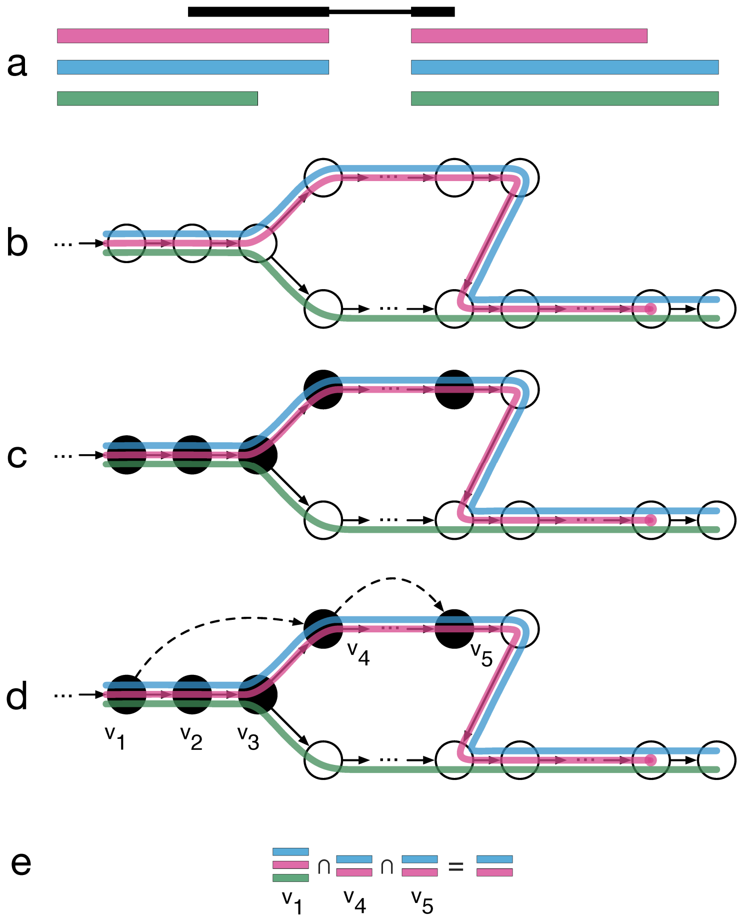


Figure 2.2: The input consists of a reference transcriptome and reads from an RNA-seq experiment. (a) An example of a read (in black) and three overlapping transcripts with exonic regions as shown. (b) An index is constructed by creating the transcriptome de Bruijn Graph (T-DBG) where nodes ($v_1, v_2, v_3, \dots$ ) are k-mers, each transcript corresponds to a colored path as shown and the path cover of the transcriptome induces a k-compatibility class for each k-mer. (c) Conceptually, the k-mers of a read are hashed (black nodes) to find the k-compatibility class of a read. (d) Skipping (black dashed lines) uses the information stored in the T-DBG to skip k-mers that are redundant because they have the same k-compatibility class. (e) The k-compatibility class of the read is determined by taking the intersection of the k-compatibility classes of its constituent k-mers.

This is unlikely to happen for large k because it would imply that the T-DBG has a directed cycle shorter than $l - k + 1$. This fact also provides a criterion that can be tested.

### Finding pseudoalignments

Reads are pseudoaligned by looking up the k-compatibility class for each k-mer in the read in the kallisto index and then intersecting the identified k-compatibility classes. In the case of paired-end reads, the k-compatibility class lookup is done for both ends of the fragment and all the resulting classes are intersected. Since the T-DBG identifies each k-mer with its reverse complement, the k-mer hashing in kallisto is strand-agnostic; however, the implementation could also be adapted to require specific strandedness of reads from strand-specific protocols. To further speed up the processing, kallisto uses the structural information stored in the index: because all k-mers in a contig of the T-DBG have the same k-compatibility class, it would be redundant to include more than one k-mer from a contig in the intersection of k-compatibility classes. This observation is leveraged in kallisto by finding the distances to the junctions at the end of its contig each time a k-mer is looked up using the hash. If the read does arise from a transcript in the T-DBG, the k-mers up to those distances can be skipped without affecting the result of the intersection, resulting in fewer hash lookups. To help ensure that the read is consistent with the T-DBG, kallisto checks the last k-mer that is skipped to ensure its k-compatibility class is equal as expected. In rare case when there is a mismatch, kallisto defaults to examining each k-mer of the read. For the majority of reads, kallisto ends up performing a hash lookup for only two k-mers (Fig. 2.3). While pseudoalignment does not require or make use of the locations of k-mers in transcripts, it is possible to extract such data from the T-DBG, and a “pseudobam output” option of kallisto takes advantage of this to produce an alignment file containing positions of reads within transcripts. With pseudobam it is possible to examine the location of reads within transcripts and genes of interest for quality control and analysis purposes.

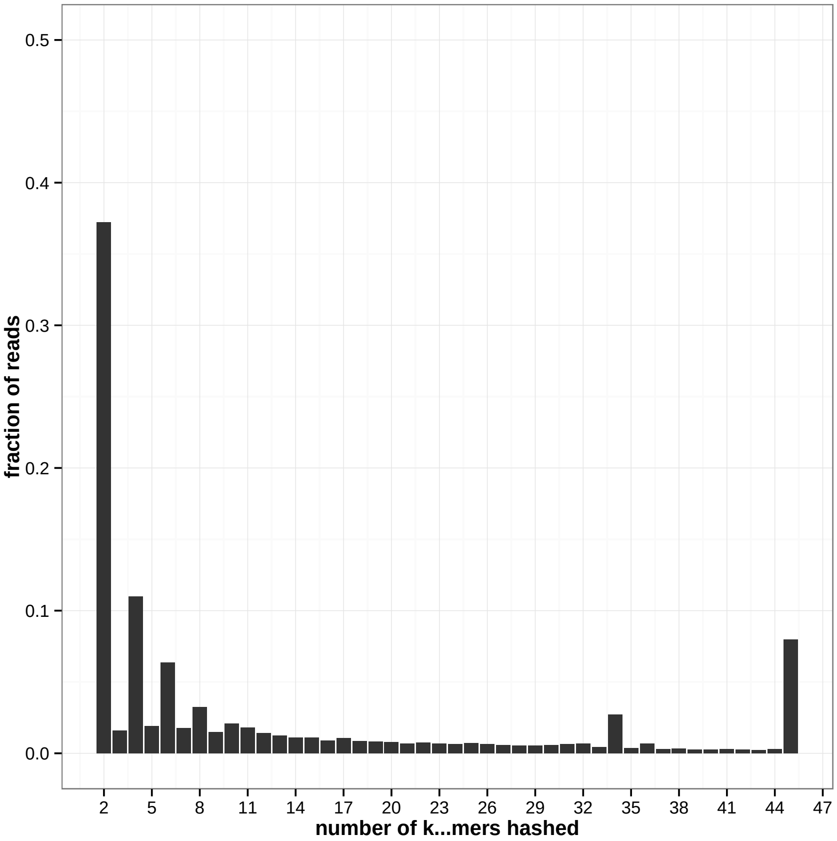


Figure 2.3: The distribution of the number of k-mers hashed per read for k = 31. Note that for the majority of reads (61.35%) only two k-mers are hashed. This happens when the entire read pseudoaligns to a single contig of the T-DBG and we can skip to the end of the read. Since we also check the last k-mer we can skip over, the most common cases are checking 2, 4, 6, and 8 k-mers. Only 1.6% of reads required hashing every k-mer of the read.

## 2.3 Quantification

In order to rapidly quantify transcript abundances from pseudoalignments, kallisto makes use of the following form of the likelihood function for RNA-seq:

$$L(\alpha) \propto \prod_{f \in F} \sum_{t \in T} y_{f,t} \frac{\alpha_t}{l_t} \tag{2.1}$$

$$= \prod_{e \in E} \left( \sum_{t \in e} \frac{\alpha_t}{l_t} \right)^{c_e} \tag{2.2}$$

In equation 2.1, $F$ is the set of fragments, $T$ is the set of transcripts, $l_t$ is the (effective) length [72] of transcript $t$ and $y_{f,t}$ is a compatibility matrix defined as 1 if fragment $f$ is compatible with $t$ and 0 otherwise. The parameters are the $\alpha_t$, the probabilities of selecting fragments from transcripts. The likelihood can be rewritten as a product over equivalence classes, in which similar summation terms have been factored together. In the factorization the numbers $c_e$ are the number of counts observed from equivalence class $e$. When equation 2.1 is written in terms of the equivalence classes, the equivalence class counts are sufficient statistics and thus, in the computations, are based on a much smaller set of data (usually hundreds of thousands of equivalence classes instead of tens of millions of reads). The likelihood function is iteratively optimized with the EM algorithm, with iterations terminating when, for every transcript $t$, $\alpha_t N > 0.01$ ($N$ is the total number of fragments) changes less than 1% from iteration to iteration.

The transcript abundances are output by kallisto in transcripts per million [49] (TPM) units.

## Inferential variability

The bootstrap is highly efficient in kallisto both because the EM algorithm is very fast and because the sufficient statistics of the model are the equivalence class counts. This latter fact means that bootstrap samples can be very rapidly generated once pseudoalignment of the fragments is completed. With the $N$ original fragments having been categorized by equivalence class, generating a new bootstrap sample consists of sampling $N$ counts from a multinomial distribution over the equivalence classes, with the probability of each class being proportional to its count in the original data. The transcript abundances for these new samples are then recomputed using the EM algorithm.

In kallisto the number of bootstraps to be performed is an option passed to the program, and because a large amount of data can be produced, the output is compressed in HDF5. The HDF5 files can be read into another program for processing (for example, R) or can be converted to plain text using kallisto.

## Bias

There are many sources of bias in RNA-seq, but previous work has identified sequence-specific bias [73] as particularly problematic. Sequence-specific bias arises as a result of nonrandom priming of fragments, where the nucleotide sequences at the 3' and 5' ends affect the probability of sampling. The kallisto correction is similar to that of [73]; however, it uses 6-mers of the transcript sequence overlapping the 5' fragment, starting 2 bp upstream of the fragment. First kallisto measures the empirical frequency of 6-mers as estimated from the first 1 million pseudoalignable reads. To apply the bias correction, it uses an initial estimate for the abundance, using 50 rounds of the EM algorithm. The bias of 6-mers is used to adjust the effective length of each transcript by adding the bias of each 6-mer on both strands. To account for edge effects, kallisto only add the 6-mers from the start up to the length of the transcript minus the average fragment length. This process is repeated once more with an updated expression estimate after 550 rounds of the EM algorithm.

## 2.4 Evaluation

To validate and benchmark kallisto, we tested it on a set of 20 RNA-seq simulations generated with the program RSEM (RNA-Seq by Expectation Maximization)[49], as well as on RNA-seq data from the Sequencing Quality Control Consortium (SEQC) [22] for which quantitative PCR (qPCR) can be used as an independent validation of quantification. The transcript abundances and error profiles for the simulated data were based on the quantification of sample NA12716_7 from the Genetic European Variation in Health and Disease (GEUVADIS) data set [44]. To accord with GEUVADIS samples, the simulations consisted of 30 million reads. We examine the quality of the kallisto pseudoalignments as compared to pseudoalignments extracted from Bowtie2 alignments. The two methods agreed exactly on the set of reported transcripts for 70.7% of the reads, but when they differed on the (pseudo)alignment of a read, Bowtie2 reported 8.02 transcripts on average compared to 4.96 for kallisto. Despite being much more specific than Bowtie2, kallisto had almost 100% sensitivity. The transcript of origin was contained in the set of reported transcripts for 99.89% of the reads, only 0.1% less than with Bowtie2 (99.99%). On the real data used as the basis for the simulations (NA12716_7), the programs displayed similar characteristics. The two methods agreed exactly for 66.22% of reads where both (pseudo)aligned, and for differing reads Bowtie2 aligned to 8.94 transcripts on average, versus 4.86 for kallisto. As expected, the number of (pseudo)aligned reads was lower for the real data, with 86.5% of the reads aligned by Bowtie2 versus 90.8% pseudoaligned by kallisto.

The accuracy of kallisto is similar to those of existing RNA-seq quantification tools (Fig. 2.4a and Fig. 2.5) and enables a substantial improvement over Cufflinks [87] and Sailfish [66]. The inferior performance of Cufflinks can be attributed to its limited application of the EM algorithm in cases where reads multi-map across genomic locations [73]. Unlike Sailfish [66], which shreds reads into k-mers for fast hashing, resulting in a loss of information, kallisto's pseudoalignments explicitly preserve the information provided by k-mers across reads (Fig. 2.1).

All programs have reduced performance on paralogs owing to the similarity among genes within a family, but kallisto remains highly competitive, again almost matching RSEM's performance (Figs. 2.6, 2.7). To test kallisto's suitability for allele-specific expression quantification, we simulate reads from a transcriptome with two distinct haplotypes. The Spearman's correlation for kallisto was 0.833 vs. 0.848 for RSEM, 0.830 for eXpress and 0.706 for Sailfish, showing that kallisto is suitable for allele-specific expression. Notably, the simulation was based on RSEM, for generating both the parameters and then the data using them.

We also tested kallisto on SEQC data that has independently been quantified with qPCR. Kallisto performed similarly to other programs (Tables 2.1, 2.2). Learning sequence specific bias (Subsection 2.3 and Table 2.3) provides a slight improvement in agreement with qPCR, similar to improvements with bias learning in Cufflinks and eXpress.

Kallisto outperformed all other methods in speed, thanks to optimizations made possible by the pseudoalignment framework (Fig. 2.2d,e, and Section 2.2). Each simulation was

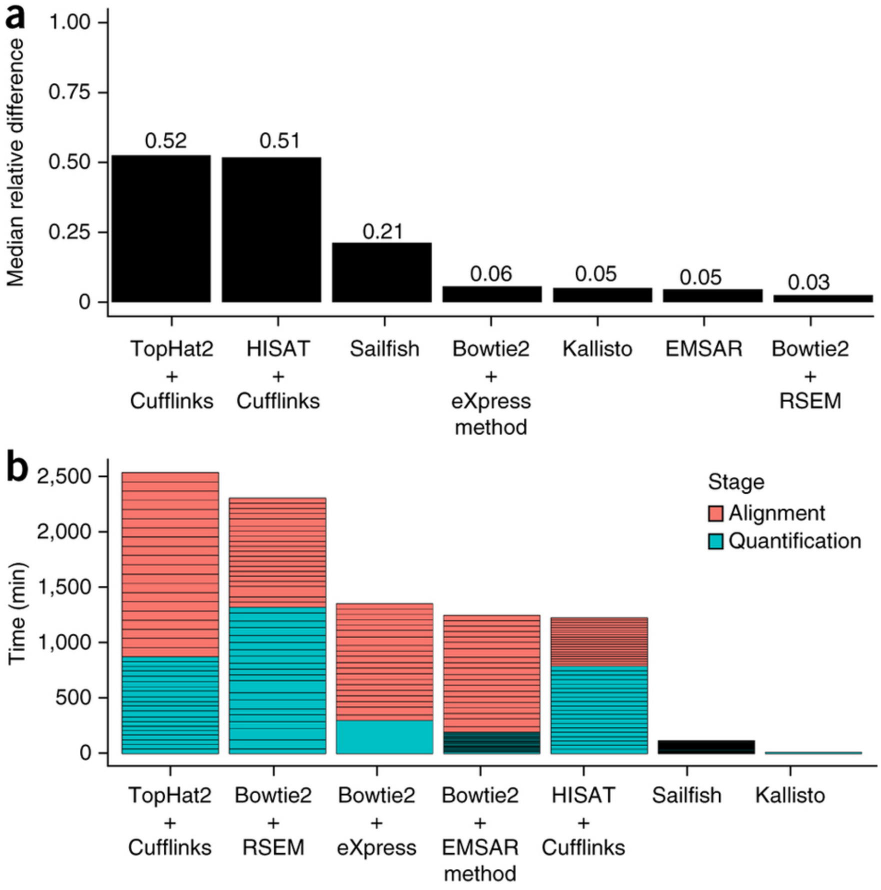


Figure 2.4: (a) Accuracy of kallisto, Cufflinks, Sailfish, EMSAR, eXpress and RSEM on 20 RSEM simulations of 30 million 75-bp paired-end reads based on the abundances and error profile of GEUVADIS sample NA12716_7 (selected for its depth of sequencing). For each simulation, we report the accuracy as the median relative difference in the estimated read count of each transcript. Estimated counts were used rather than transcripts per million (TPM) because the latter is based on both the assignment of ambiguous reads and the estimation of effective lengths of transcripts, so a program might be penalized for having a differing notion of effective length despite accurately assigning reads. The values reported are means across the 20 simulations (the variance was too small to be visible in this plot). Relative difference is defined as the absolute difference between the estimated abundance and the ground truth divided by the average of the two. (b) Total running time in minutes for processing the 20 simulated data sets of 30 million paired-end reads described in a. All processing was done using 20 cores, with programs being run with 20 threads when possible (Bowtie2, TopHat2, RSEM, Cufflinks) and 20 parallel processes otherwise (eXpress, kallisto). Each box represents one dataset. Since eXpress and kallisto process all datasets in parallel, the only quantification time shown is the maximum of all the quantifications.

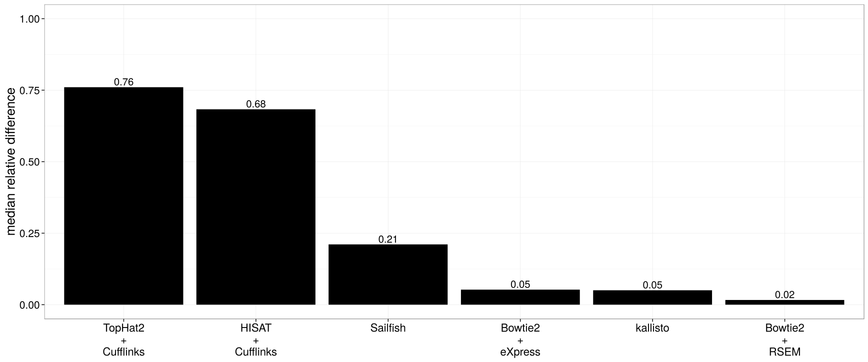


Figure 2.5: Accuracy of kallisto, Cufflinks, Sailfish, eXpress and RSEM on 20 RSEM simulations of 30 million 75bp paired-end reads based on the TPM estimates and error profile of Geuvadis sample NA12716 (selected for its depth of sequencing). For each simulation we report the accuracy as the median relative difference in the estimated TPM value of each transcript. The values reported are means across the 20 simulations (the variance was too small for this plot). Relative difference is defined as the absolute difference between the estimated TPM values and the ground truth divided by the average of the two.

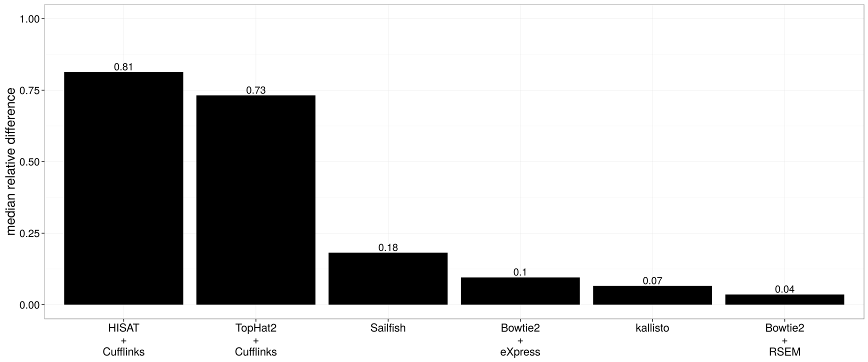


Figure 2.6: Performance of different quantification programs on the set of paralogs in the human genome supplied by the Duplicated Genes Database (http://dgd.genouest.org). This set includes 8,636 transcripts in 3,163 genes.

processed on average in less than 7.5 min on a single core. The total runtime for kallisto on the simulated data was 11.47 min (Fig. 2.4b). A simple word count of a simulated data set took 75 s, providing a lower bound for optimal quantification time and demonstrating that kallisto's speed is near optimal. The software is also memory efficient, requiring a maximum of 3.2 Gb of RAM per sample. This allows kallisto to process 30 million read simulations in less than 10 min on a small laptop with a 1.3-GHz processor, demonstrating that with kallisto, RNA-seq analysis of even large data sets is tractable on non-specialized hardware.

The speed of kallisto also enables uncertainty of abundance estimates to be quantified

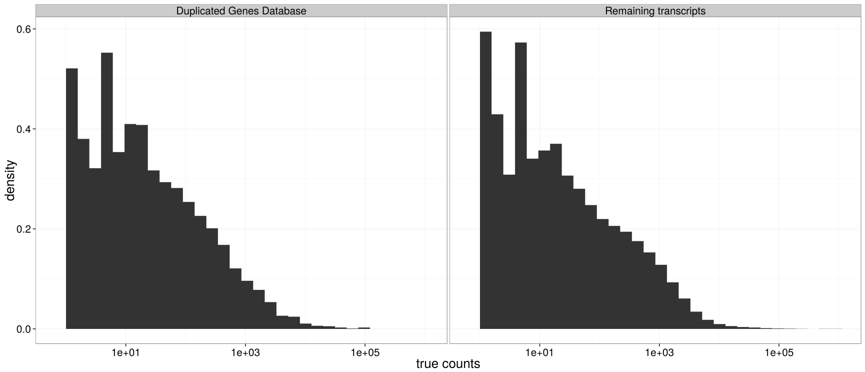


Figure 2.7: Count distribution of one simulation. The left panel contains the transcripts used in Figure 2.6. The right panel contains the remaining transcripts. The x-axis is on the log scale. Both distributions appear very similar, suggesting that the drop in performance in Figure 2.6 is from sequence similarity and not oddities in the distribution such as very low counts.

| Method | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Cufflinks | 0.6644 | 0.6617 | 0.6648 | 0.6651 |
| Cufflinks (bias) | 0.6716 | 0.6675 | 0.6696 | 0.6697 |
| EMSAR | 0.6677 | 0.6633 | 0.6688 | 0.6683 |
| eXpress | 0.668 | 0.6656 | 0.6679 | 0.6693 |
| kallisto | 0.6673 | 0.6625 | 0.665 | 0.6664 |
| kallisto (bias) | 0.6694 | 0.6649 | 0.6664 | 0.6686 |
| RSEM | 0.6658 | 0.6593 | 0.6664 | 0.6675 |
| Sailfish | 0.658 | 0.6553 | 0.6599 | 0.6606 |

Table 2.1: Performance of quantification at the transcript level as measured by SEQC qPCR.

| Method | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Cufflinks | 0.7378 | 0.74 | 0.7424 | 0.7416 |
| Cufflinks (bias) | 0.7503 | 0.7504 | 0.7508 | 0.7503 |
| EMSAR | 0.742 | 0.7439 | 0.747 | 0.7467 |
| eXpress | 0.7526 | 0.7528 | 0.7533 | 0.7532 |
| kallisto | 0.74 | 0.7417 | 0.7446 | 0.7446 |
| kallisto (bias) | 0.7465 | 0.7473 | 0.75 | 0.75 |
| RSEM | 0.74 | 0.7421 | 0.745 | 0.7447 |
| Sailfish | 0.7444 | 0.7465 | 0.7494 | 0.7494 |

Table 2.2: Performance of quantification at the gene level as measured by SEQC qPCR.

via the bootstrap technique of repeating analyses after resampling with replacement from

| Method | Error | | Error + Bias | |
|---|---|---|---|---|
| | MRD | Spearman | MRD | Spearman |
| kallisto (bias) | 0.0788 | 0.9849 | 0.103 | 0.9609 |
| kallisto | 0.0794 | 0.9852 | 0.1226 | 0.9509 |
| eXpress | 0.061 | 0.9859 | 0.1126 | 0.9722 |

Table 2.3: Performance of kallisto with and without bias.

the data. After the equivalence classes of the original reads have been computed, kallisto samples multinomially from the equivalence classes according to their counts and runs the EM algorithm on those newly sampled equivalence class counts. The running time for each bootstrap sample depends on the number of equivalence classes, which is much smaller than, and roughly independent of, the number of reads. While run times are transcriptome-specific, each sample typically takes on the order of 10 s, and kallisto can multithread the bootstrapping. Since the data associated with each bootstrap consists solely of a set of equivalence class counts and transcript abundances, the memory usage is trivial. We explore the accuracy with which the bootstrap can estimate the uncertainty inherent in a dataset by examining repeated 30 million read subsamples of a deep 216-million-read human RNA-seq dataset from the SEQC-MAQCIII [22] consortium (Fig. 2.8). We perform 40 bootstraps (see Fig. 2.9 for an analysis of convergence) on only a single sample of 30 million reads, yet the variance in estimates correlated highly (R = 0.933) with the variance of abundance estimates obtained from the other subsamples. While it is expected that the variance on abundance estimates should increase approximately linearly with abundance [59], our results show that there is high variability in uncertainty of estimates as a result of the complex structure of similarity among transcripts, especially multiple isoforms of genes. A naive attribution of Poisson variance to the shot noise in read count estimates from transcripts, as is commonly done in gene-level RNA-seq analyses, is revealed to be a poor proxy for the true variance (Figs. 2.10, 2.11). Thus, the bootstrap should prove to be valuable in downstream applications of RNA-seq, as kallisto now allows the uncertainty in estimates to be factored in to downstream statistical computations.

The simplicity of kallisto means that the software has few parameters. Only the k-mer length and the mean of the fragment length distribution are required for quantification. The latter is estimated during run-time when paired-end reads are provided. The k-mer length must be large enough that random sequences of length k do not match to the transcriptome and short enough to ensure robustness to errors. Subject to those constraints, the performance of kallisto is robust to the k-mer length chosen (Figs. 2.12 and 2.13). Although we have focused on the performance of kallisto on RNA-seq, the method should be generally applicable to quantification of sequence census datasets [92].

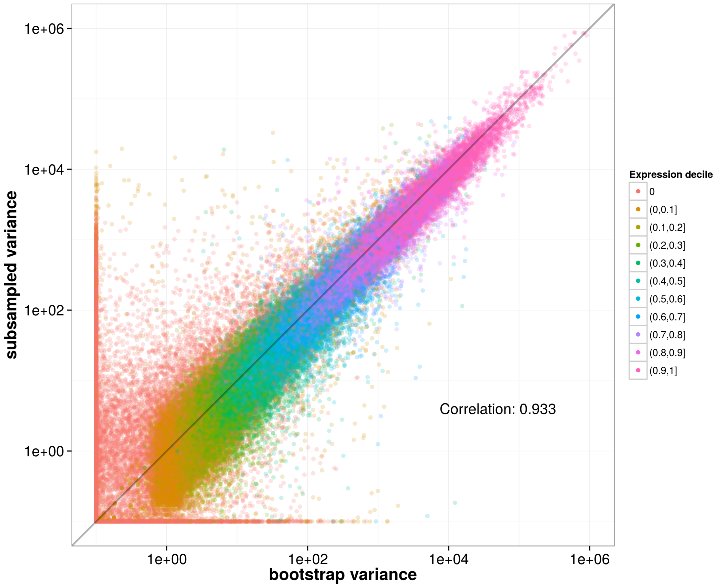


Figure 2.8: The data comes from a single library with 216M, 101bp paired-end reads sequenced. Each point corresponds to a transcript and is colored by the decile of its expression level in the single bootstrapped subsample. The Y-axis represents variance of abundance estimates across 40 subsamples, with 30M reads in each subsample. The X-axis represents variance as computed from 40 bootstraps of a single subsampled dataset of 30M reads. The red lines emanating from the lower left corner consist of transcripts that have an estimated abundance of zero in the single bootstrapped experiment, but show expression in some of the subsamples (12968 transcripts), and vice versa (720 transcripts).

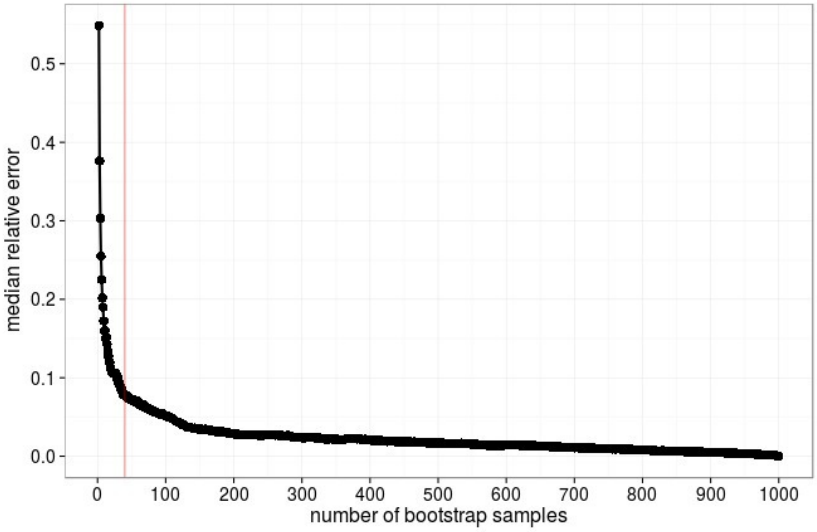


Figure 2.9: Median relative error (with respect to 1000 bootstraps) of inferred transcript variances as a function of number of bootstrap samples performed. The relative error with 40 bootstraps (red line) is 7.8%.

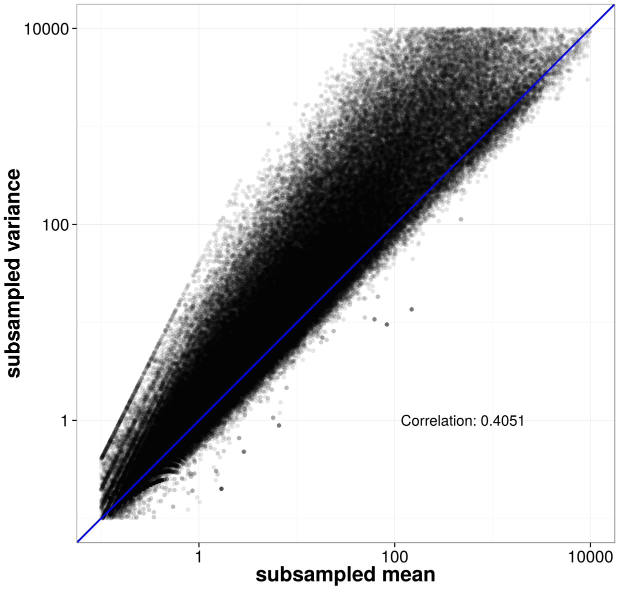


Figure 2.10: Relationship between the mean and variance of estimated counts for each transcript (x and y axes are on log scale) based on 40 subsamples of 30M reads from a dataset of 216M PE reads. The x-axis is the mean of each count estimate calculated across the subsamples. The y-axis is the variance of the count estimates calculated across subsamples.

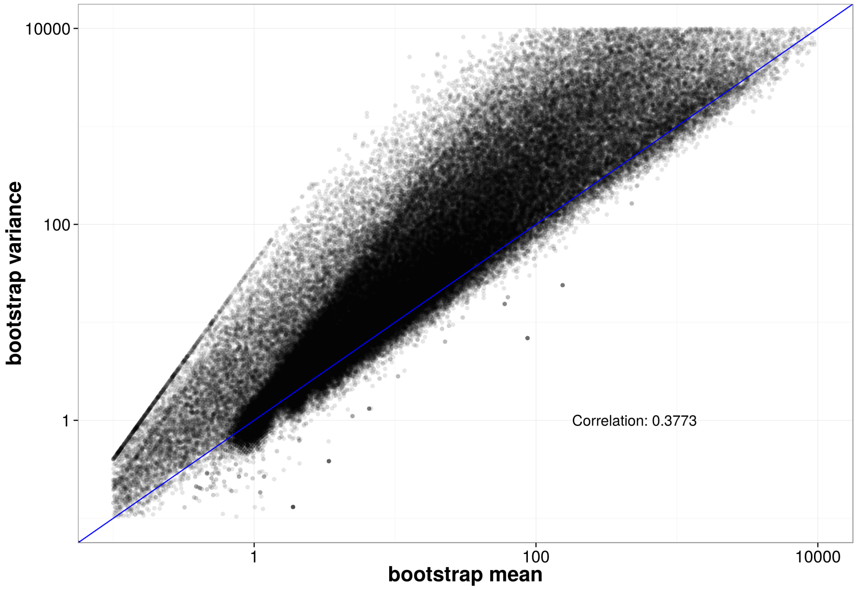


Figure 2.11: Relationship between the mean and variance of estimated counts for each transcript (x and y axes are on log scale) based on 40 bootstraps of a single subsample of 30M reads from the same 216M PE read dataset. The x-axis is the mean of the count estimates calculated across the 40 bootstraps. The y-axis is the variance of the count estimates calculated across the 40 bootstraps.

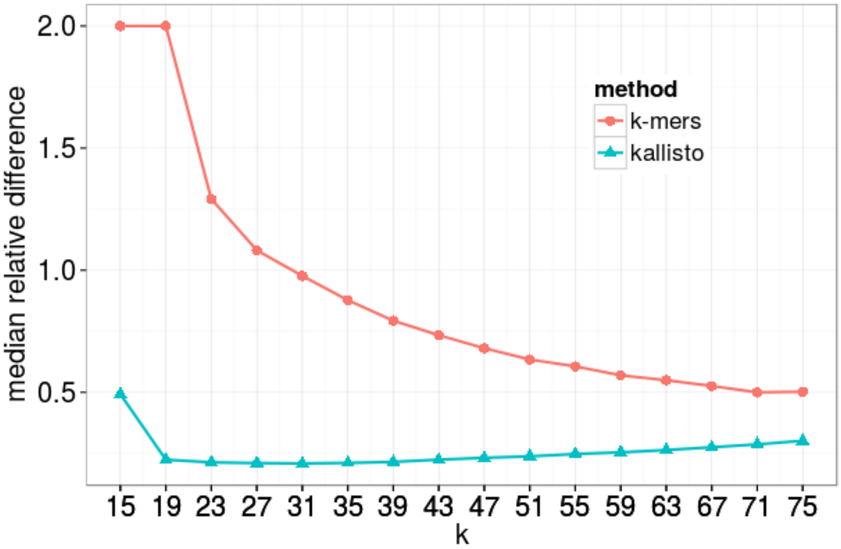


Figure 2.12: Median relative difference from 30M 75bp PE reads simulated with error for different values of k. The "k-mers method" uses the k-compatibility of each k-mer independently and runs the EM algorithm on k-mers, whereas kallisto uses the intersection of k-compatibility classes across both ends of reads. When there are errors in the reads, kallisto requires smaller k-mer lengths for robustness in pseudoalignment.

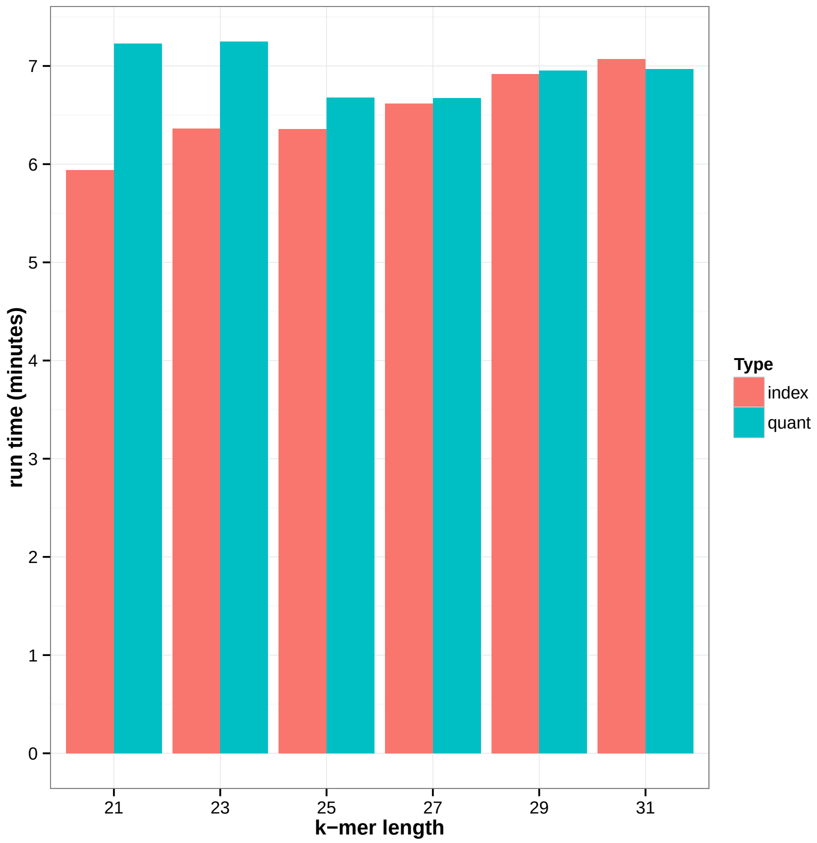


Figure 2.13: Run time for index building and quantification as a function of k-mer length for one of the simulated samples.

## 2.5 Software, simulations and analysis

The kallisto program is available for download from `http://pachterlab.github.io/kallisto/`. The parameters and procedures for Cufflinks, Sailfish, EMSAR [46], eXpress, and RSEM used for the results and figures in the paper are available via a Snakefile [42] at `https://github.com/pachterlab/kallisto_paper_analysis`. Source code for reproducing results and figures of the paper is available here as well.

# Chapter 3

# Differential expression analysis

We describe a novel method for the differential analysis of RNA-Seq data that utilizes bootstrapping in conjunction with response error linear modeling to decouple biological variance from inferential variance. The method is implemented in an interactive shiny app called sleuth that utilizes kallisto quantifications and bootstraps for fast and accurate analysis of RNA-Seq experiments.

This work is currently available by preprint as "Differential analysis of RNA-Seq incorporating quantification uncertainty" [70] and is being reproduced with the permission of the co-authors.

## 3.1 Introduction

RNA-Seq technology has largely replaced microarray measurement as a tool for identifying gene expression differences in comparative and clinical analyses of RNA samples [15]. In analogy with microarrays, differential analysis of RNA-Seq experiments requires careful assessment of variability of gene expression from few replicate samples, so as to be able to identify biologically relevant expression differences between conditions [29, 4]. However there are also key differences between the technologies. While microarrays measure cDNA hybridization intensities at pre-defined probes, RNA-Seq provides a de novo sampling of the transcriptome, a feature that makes it much more powerful for detecting transcription of individual isoforms of genes, but that also complicates differential analysis.

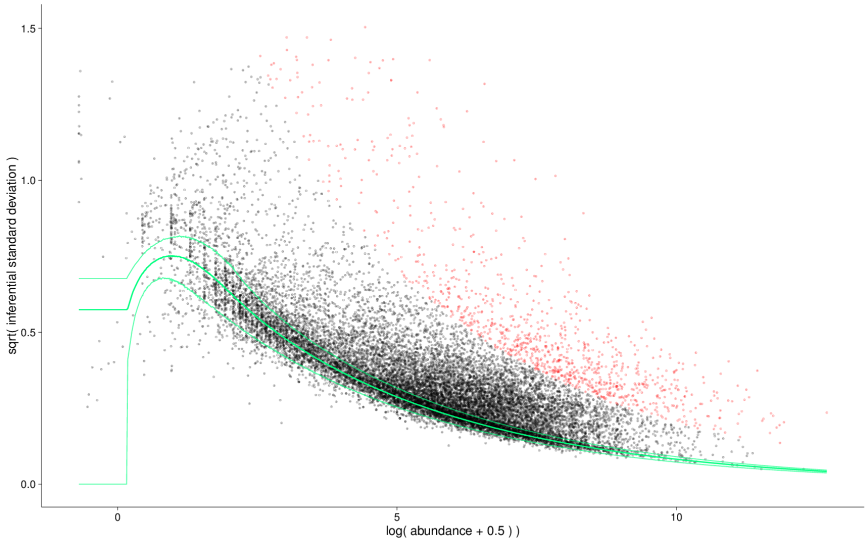
Many methods have been developed for differential analysis of RNA-Seq data [21]. Some of these seek to translate ideas developed for microarray analysis to the RNA-Seq setting [45] whereas others are based on models tailored to RNA-Seq [36, 50, 87, 63, 28]. One of the key differences between RNA-Seq and microarray technology is that the data of the former consists of counts of reads rather than intensities measured at probes, and there has therefore been considerable effort devoted to exploring appropriate distributions for the modeling of count-based data in the RNA-Seq context [29, 4, 45, 76, 51, 23, 52]. However the question of how to best utilize RNA-Seq data for differential analysis continues to be debated, with disagreements persisting on some of the most basic questions such as how to measure the abundance of genes [21], whether there is sufficient power to test for differences in abundance of individual isoforms [48, 82] and how to best utilize biological replicates [79].

Part of the reason for the continuing uncertainty regarding how best to analyze RNA-Seq data is the lack of agreed upon standards for testing and benchmarking methods. In most cases accuracy claims are based on simulations of counts of reads from distributions assumed in the models, rather than simulations of raw reads [4, 45, 28, 51, 52, 75, 56]. Such count-based simulations typically discount the effects of ambiguously mapping reads and fail to capture both the possibilities for, and challenges of, isoform-specific differential analysis. Even when simulation studies are based on reads, they are sometimes restricted to a small portion of the transcriptome [83, 27] thereby biasing results due to the dependence of some methods on transcriptome-wide data for obtaining variance estimates from replicates. Studies utilizing biological data frequently make use of questionable choices for "ground truth", e.g. utilizing results from microarrays, or from a differential analysis with a method that is imperfect [83]. The resulting benchmarks are therefore difficult to interpret.

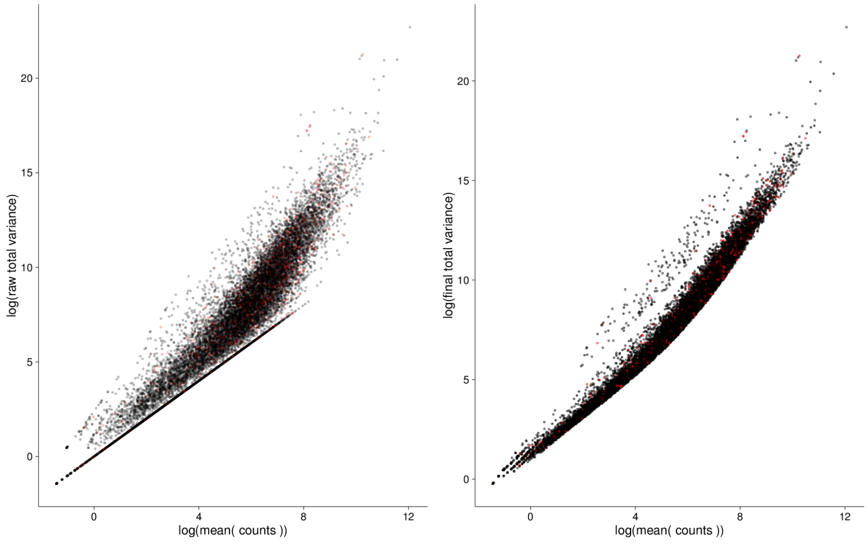
In this work we describe a novel approach to differential analysis of RNA-Seq data, a comprehensive framework for benchmarking our method and others that is unprecedented in its scale and scope, and interactive visualization software for exploring the results of our method and the data they are based on. The latter is crucial for providing transparency in assessing our results, and has the benefit of offering users a convenient tool for exploratory data analysis. Throughout the paper we use the name sleuth to refer both to our statistical method, as well as the app that allows for working with and exploring results.

The motivation for the conceptual approach underlying sleuth is illustrated in Figure 3.1. A key element of sleuth is borrowed from previous work[4, 45], namely shrinkage to stabilize variance estimates from few samples. But sleuth is able to leverage recent advances in quantification[11] to obtain error estimates for quantifications that can in turn be used to decouple biological variance from inferential variance before shrinkage. A few other methods have attempted to compute and utilize error estimates on quantifications, but some major computational and statistical hurdles have been difficult to overcome. For example, the BitSeq method[28] obtains quantification error estimates via Markov Chain Monte Carlo sampling, a process that requires significant, and in some cases prohibitive, computational resources. An update to the initial paper introduces quantification using variational Bayes, however it is not recommended for use in differential analysis[31]. The Cuffdiff 2 method[85] also performs sampling to assess variance arising from quantification but has some model limitations: Cuffdiff 2 fails to accurately assess the increase in variance due to reads mapping ambiguously across the genome. EBSeq[48] also models inferential variance, but does so in discrete classes that only act as a proxy for high inferential variance.

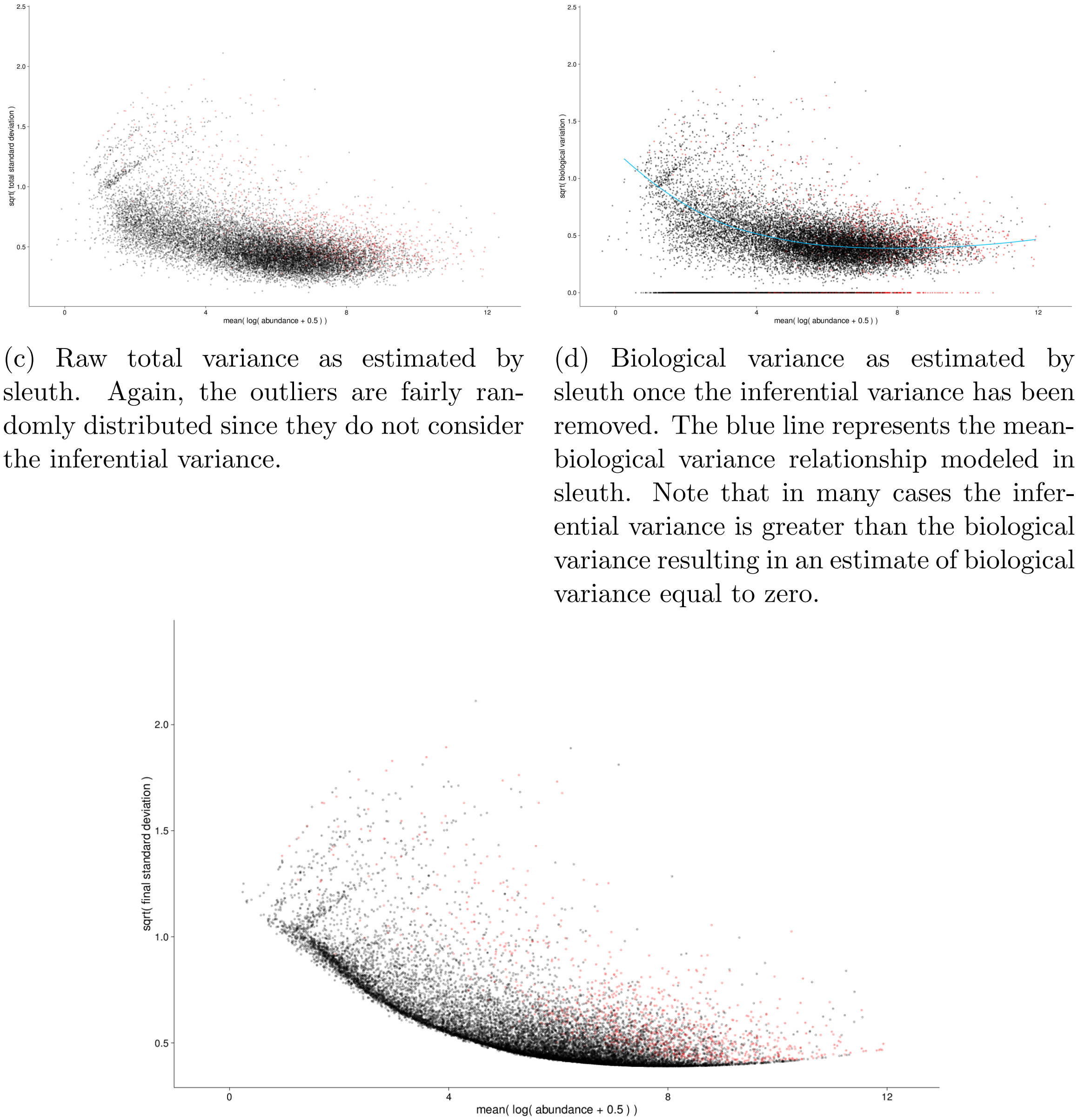
Thus, sleuth is able to improve on traditional "count-based" methods by utilizing improved estimates of transcript and gene abundances in a flexible and powerful statistical framework. The sleuth concept is illustrated by example in Figure 3.1 via the tracing of genes whose abundances are difficult to estimate. In the example shown, which is based on a gene-level analysis of data from Bottomly et al. [8], genes with high inferential variance



(a) Inferential variance on sample SRR099228. The x-axis is the gene abundance, and the y-axis is the bootstrap estimate of the inferential variance. The green lines represent the 5% confidence bound, mean, and 95% confidence bound expected under the Poisson model.



(b) Mean expression versus total variance as estimated by DESeq2. The left panel contains the raw estimates of the variance. The right panel contains the smoothed estimate of the variance. Note that the outliers are fairly randomly distributed across variance and expression patterns.



(c) Raw total variance as estimated by sleuth. Again, the outliers are fairly randomly distributed since they do not consider the inferential variance.

(d) Biological variance as estimated by sleuth once the inferential variance has been removed. The blue line represents the mean-biological variance relationship modeled in sleuth. Note that in many cases the inferential variance is greater than the biological variance resulting in an estimate of biological variance equal to zero.

(e) Final total variance as modeled by sleuth. Note that almost all of the outliers have higher abundance than the non-outliers due to high inferential variance.

Figure 3.1: The effect of modeling inferential variance in sleuth at the gene level. Outliers are colored red and traced across all plots. A point is an outlier if the variance is greater than 100 times the interquartile range plus the upper quartile.

(red dots in Figure 3.1a) are lumped together with genes with high biological variance by DESeq2 (Figure 3.1b). This makes it difficult to correctly assign a high total variance to those genes prior to differential analysis (Figure 3.1c). Unlike DESeq2, by decoupling biological and inferential variance (Figure 3.1c, d), sleuth assigns a high total variance to most of the genes with high inferential variance (Figure 3.1e).

The key innovation in sleuth, namely the explicit modeling of biological and inferential variance, is performed with a response error model (see Methods). The model is simple, transparent and its parameters are easily interpretable. Inference of parameters is straightforward (see Methods). Thus, when coupled with kallisto [11], which is fast in both the quantification and variance estimation, sleuth provides a statistically rigorous, flexible and efficient solution for RNA-Seq analysis.

## 3.2 Results

### Modeling inferential variability

The fundamental issue in differential analysis of RNA-Seq experiments is quantifying the variance in experiments so that true differences in expression can be identified as such. There are multiple sources of variance that contribute to the total variance observed between samples in an RNA-Seq experiment which we group into two classes: (1) "biological variance" which is a term used to describe variance in transcript abundance of biological and experimental origin and (2) "inferential variance" which is a term we use to describe both variance in the number of reads sequenced from a transcript due to the random nature of sequencing as well as the variance that emerges as a result of the statistical nature of transcript abundance estimation with ambiguously mapping reads. In the absence of ambiguously mapping reads there is no increase in inferential variance as the origin of each read can be inferred exactly. However when reads map to many transcripts the read counts must be deconvoluted to obtain the abundance estimates [87] leading to uncertainty in abundance estimates which translates into variance in quantification across samples.

Response error modeling [13] allows for the separate modeling of biological variance and inferential variance. The use of the bootstrap during quantification allows us to estimate the inferential variance directly for each sample [11], whereas biological replicates allow us to estimate the total variance, albeit via shrinkage due to the limited number of replicates in most experiments (see Methods). The biological variance can then be estimated (see Methods). In Figure 3.1 we illustrate this procedure by first showing the inferential variance versus the mean at the gene level (Figure 3.1a). Notice that for many genes the inferential variance is much higher than the expected Poisson variance (which is assumed by most methods). We track the outliers in red throughout all of Figure 3.1. Figure 3.1b shows the total variance when estimated using DESeq2 (raw estimate on left, final shrinkage estimate on right). The inferential variance outliers are randomly distributed throughout the mean variance relationship since there is no additional information about the inferential variance.

Figure 3.1c shows the raw estimate of the total variance in sleuth without any knowledge of the inferential variance. Here, the red points are randomly distributed, while Figure 3.1d shows the biological variance after the inferential variance has been removed. Many of the red points are distributed near biological variance zero. These points have more inferential variance than biological variance. The points with nonzero biological variance are a mix of points where the total variance is very high or inferential variance is very low. Sleuth performs shrinkage on these estimates (blue line). Figure 3.1d displays the final estimates of the total variance, which are the smoothed biological estimates plus the inferential variability estimates (Figure 3.1a). Due to the shrinkage procedure, the red points have the highest total variance and are penalized heavily as a result unlike the final variance in Figure 3.1b and 3.1c.

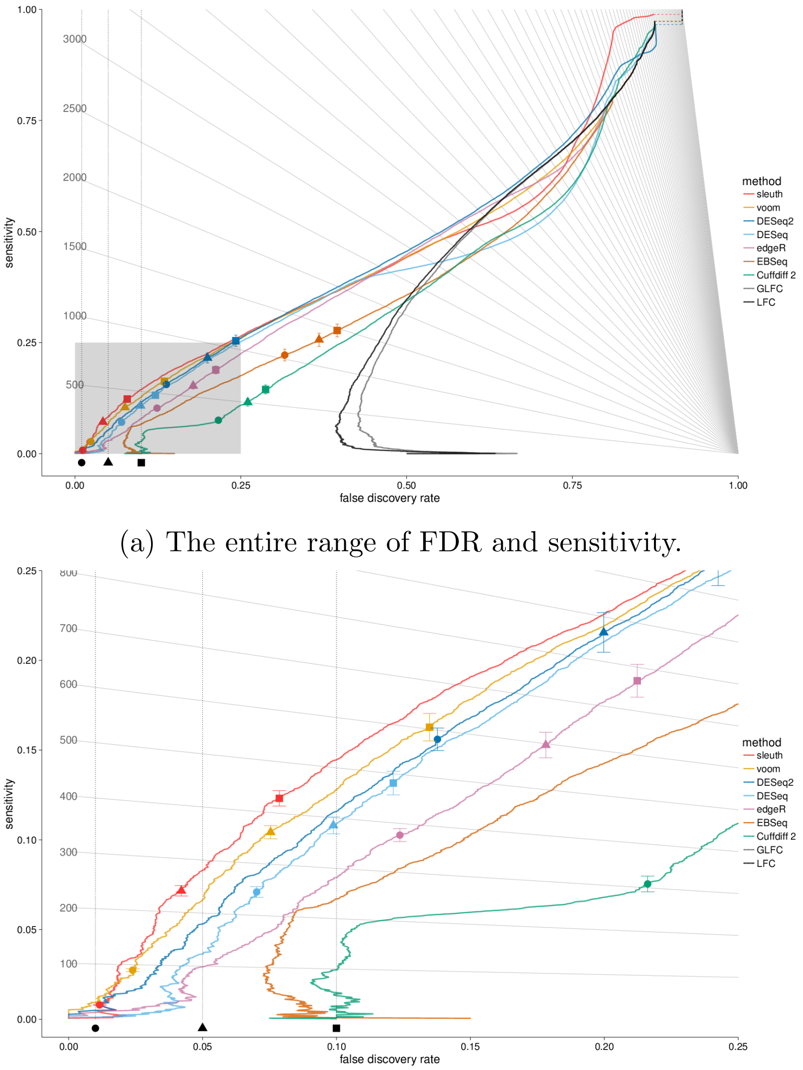
## Improved accuracy in differential gene analysis

In order to test the performance of sleuth in differential gene analysis we examined both simulated and real data and compared it to numerous other widely used methods. Our simulation was derived from an experiment with two conditions and three replicates in each condition (see Methods). We simulated biological variance (dispersion) according to the negative binomial model for counts used by DESeq2 [56] (see Methods). To be able to accurately assess performance, each simulation was performed with 20 replicates.

Figure 3.2 shows the result of our method in identifying differentially expressed genes, alongside results from Cuffdiff 2[85], DESeq [4], DESeq2 [56], EBSeq [48], edgeR [76, 75], voom [45], and log fold change [83]. The sensitivity of sleuth is higher than all other methods in the false discovery rate (FDR) range of usual interest and beyond, up to FDR 0.3. The figure also shows that as expected, DESeq2 has more power than DESeq at all the relevant FDRs, and that the naïve approach of ranking genes by log-fold change produces poor results. Even when benchmarked in simulation conditions favorable to traditional “count-based” methods, sleuth outperforms other programs (Figures 3.7, 3.9). We also examined the effect of different filtering strategies on performance by comparing sleuth with other programs on a common filtered set of genes, showing that sleuth maintains its advantage independent of filtering (Section 3.13).

## Estimation of false discovery rate

Since the control of the false discovery rate is fundamental for identifying differentially expressed genes in experiments with few replicates, we examined carefully the accuracy of methods in self-reporting their false discovery rates [77]. This was easy to do with our simulated data where the truth was known. In Figures 3.2 and 3.4 the circles, squares, and diamonds represent the average estimated FDR output by each program across the 20 replicates performed for each simulation scenario. Other than sleuth and voom, other methods significantly underestimated the FDR with several methods reporting an estimated FDR of 0.01 when the true FDR was greater than 0.1. While sleuth overestimates the FDR, this



(a) The entire range of FDR and sensitivity.

(b) Zoomed in version of the gray section from (a).

Figure 3.2: Sensitivity versus FDR in the "effect from experiment" simulation at the gene level. The x- and y-axes show the true false discovery rate and sensitivity, respectively, and for each program one has a curve showing how those values change as one moves down its ranking of all genes passing its filter. The circle, triangle, and square on each curve show where in its ranking that program estimates an FDR of 0.01, 0.05, and 0.10 respectively. Ideally, each symbol would lie directly above the corresponding symbol on the x-axis indicating a true FDR of 0.01, 0.05, or 0.10. Also displayed are isolines for a constant number of genes being called differentially expressed. Where an isoline intersects with the curve for a given program shows its performance when looking at that many genes from the top of its ranking. The FDR lines were averaged over 20 replications of the simulation.

error is conservative, i.e. fewer genes are reported, yet they are highly enriched for being differentially expressed.

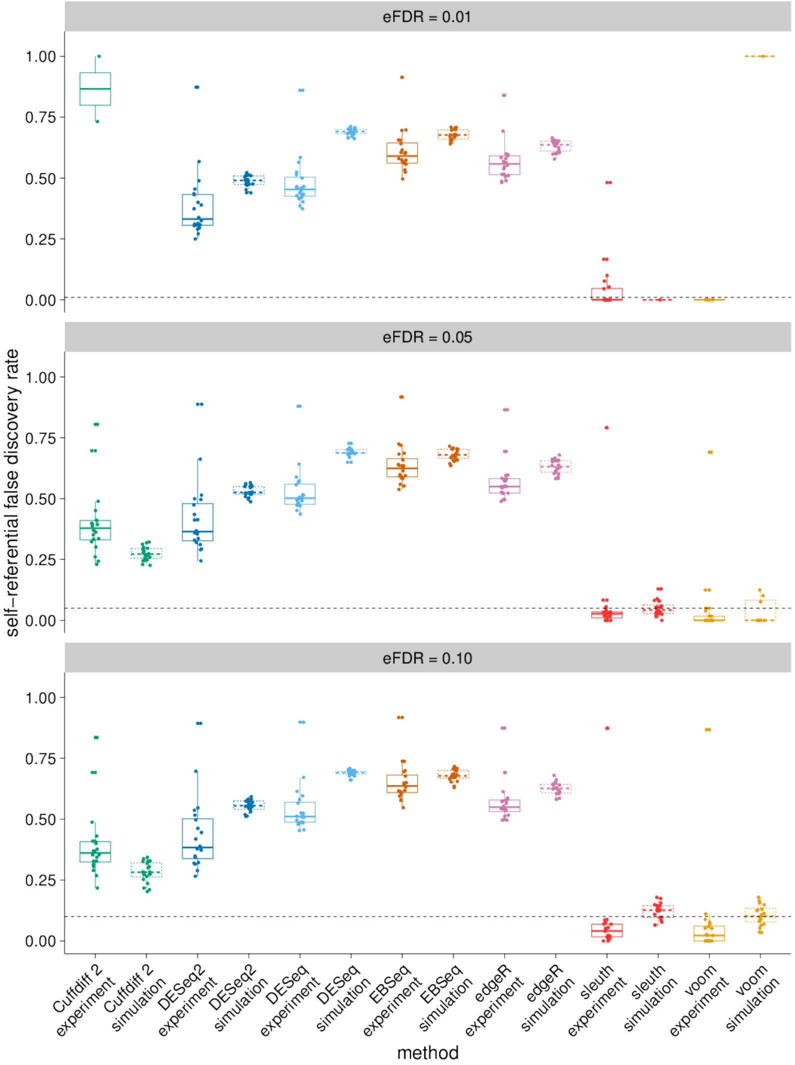
To test whether our results translated to FDR estimation accuracy on real data, we repeated an experiment from the DESeq2 paper [56]. Using the Bottomly data set [8], which contains multiple replicates from two mice strains (10 and 11 respectively), we created a training set by randomly selecting 3 versus 3 samples and using differential expression results from the remaining 7 versus 8 as the "truth". Each method was compared to itself to see how well it could recapitulate its results with a smaller set of data and how it controlled the FDR as assessed by comparing to the results of the high replicate analysis. We iterated this procedure 20 times. Figure 3.3a shows that as in the simulation, sleuth and voom are the only methods able to estimate their FDR to within a reasonable approximation.

To see whether the consistency experiment provided results concordant with those of our simulations, we performed the consistency experiment with simulated data. The results are shown in Figure 3.3b, which illustrates both that our simulated data recapitulates the results on real data, and that the self-referential FDRs are good proxies for true FDRs, thus validating the reliability of the DESeq2 consistency experiment.

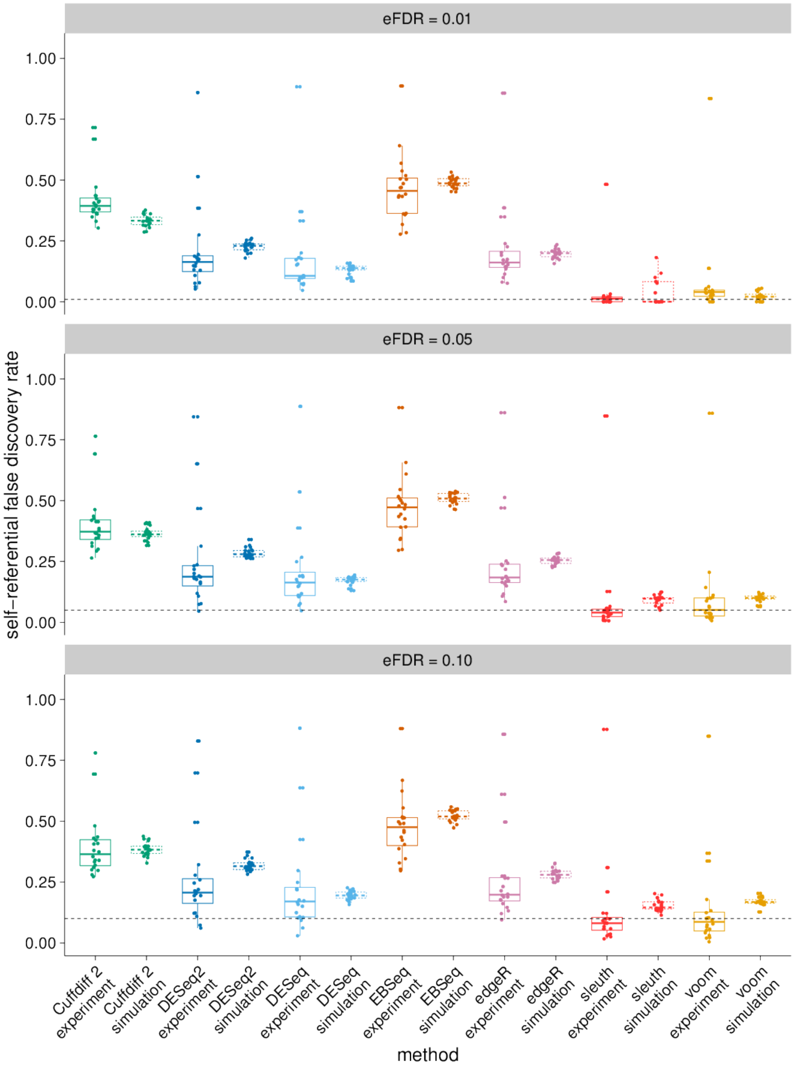
## sleuth allows for isoform-level differential analysis

While RNA-Seq has become the standard technology for gene-level differential analysis, there has been some debate about its suitability and power for isoform-level differential analysis. In previous work, we and others have provided examples of how isoform-level differential analysis can highlight interesting splicing and differential promoter usage between conditions [85, 6] but there has been debate about the significance and reliability of such results [82, 83].

In order to examine this question, we repeated the gene-level analysis at the transcript level (see Figure 3.4). We confirm previous findings that because increased testing is required for isoform-level analysis, there is a decrease in sensitivity in comparison to gene level analysis. However we also find that sleuth can still control the false discovery rate at the isoform level while calling many isoforms differentially expressed. Interestingly, while there is less power to discover differentially expressed isoforms, our simulations show that at a given FDR the number of differentially expressed features is fairly similar to that of genes (isolines in Figures 3.2, 3.4). Moreover, when simulating from a scenario in which isoform abundances change independently between conditions (Figures 3.6), we find highly significant improvements in sleuth with respect to other methods. The same is also true for the correlated effect simulation. In addition, we tested BitSeq [28] (Figure 3.16) on a single sample as its run-time was prohibitive on the entire simulation set. We found BitSeq performed well overall although sleuth outperformed it when the true FDR was less than 0.12.

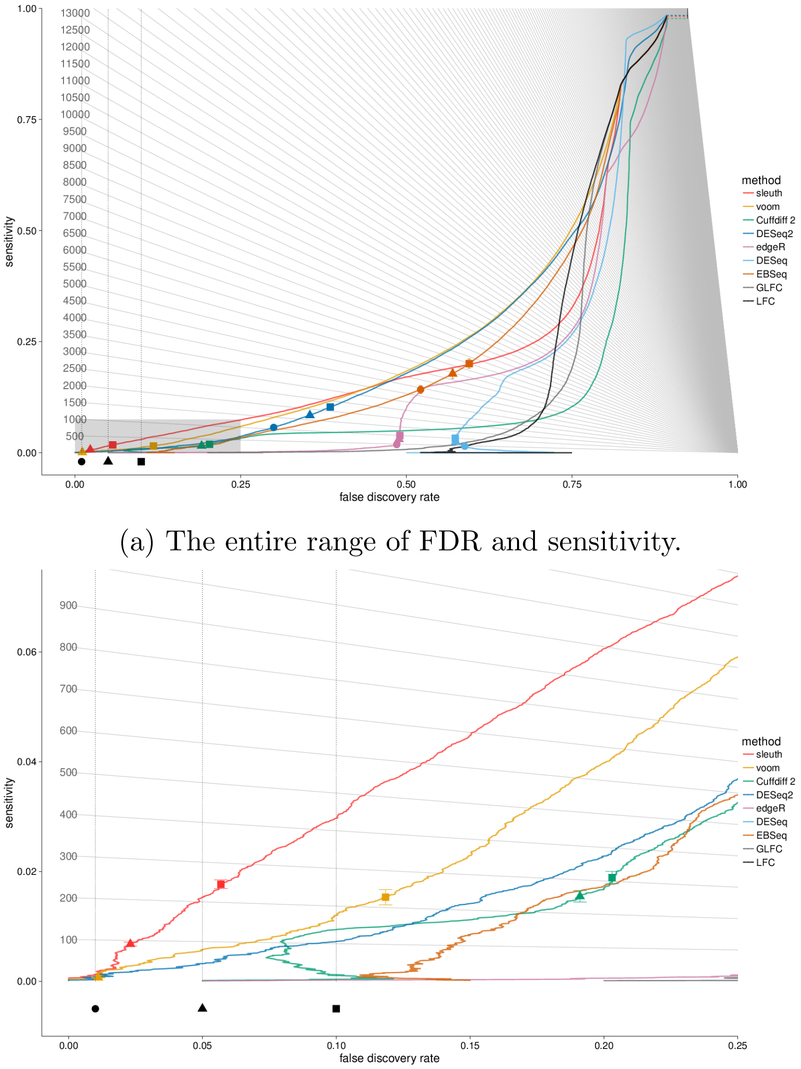


(a) Self-referential FDR at the isoform level.



(b) Self-referential FDR at the gene level.

Figure 3.3: Self-referential FDR for the Bottomly data set and our simulation at the (a) isoform level, and the (b) gene level. The suffix "experiment" refers to the Bottomly data set whereas "simulation" refers to our simulated experiments to mimic the Bottomly resampling experiment. The panels from top to bottom display the true FDR for each program when it estimates the FDR as 0.01, 0.05, and 0.10, respectively. The dashed horizontal line represents the target FDR. Only sleuth and voom seem to control the self-referential FDR reasonably well at both the isoform and gene level.



(a) The entire range of FDR and sensitivity.

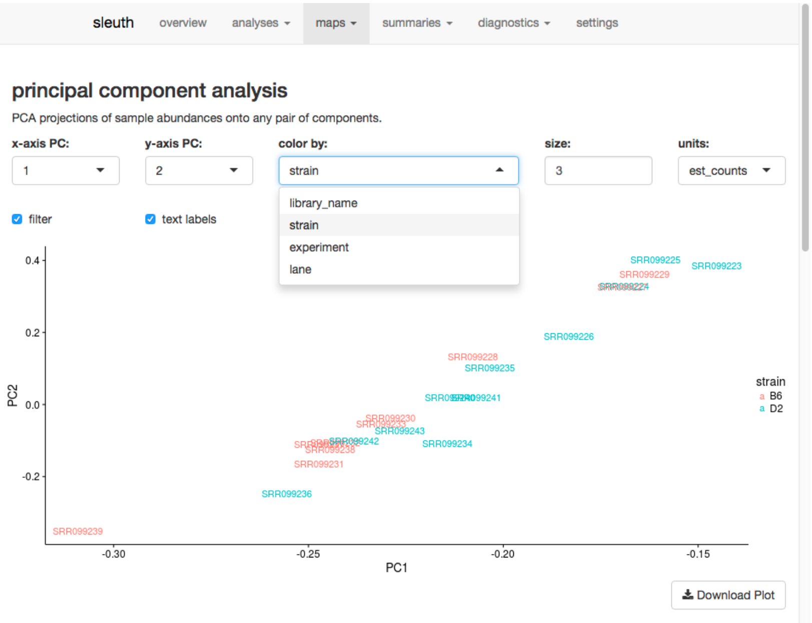
(b) Zoomed in version of the gray section from (a).

Figure 3.4: Sensitivity versus FDR in the “effect from experiment” simulation as in Figure 3.2 at the transcript rather level.

## Interactive exploratory data analysis using sleuth

The interpretation and analysis of RNA-Seq data is complicated by a number of factors: the large quantity of reads sequenced in typical experiments (many millions) and the large number of transcripts / genes (typically tens of thousands) make it difficult to interactively examine the data. However exploratory data analysis is important both for understanding how to analyze the data and in the formation of hypotheses about the results. To address this issue, and to make it possible to evaluate and assess the results of sleuth, we have developed a Shiny[16]-based interactive app for examining sleuth results.

Figure 3.5 shows some screenshots from a sleuth analysis of the Bottomly data[8]. Figure 3.5a shows the principal component analysis of the data set colored by the different conditions. One can see that the first two principal components do not segregate the data by experimental condition (mouse strain). Figure 3.5b shows how one can use the drop-down menu to change the coloring, revealing that the first two principal components seem to explain some of the variation due to the batch. In addition, there are many other features assisting in exploring the data, such as the ability to view, sort and search the table of differential expression results. For example, sorting by the inferential variability and then by largest p-values, we find transcript ENSMUST00000113388, which is not reported as differentially expressed by sleuth, but is reported as differentially expressed by both voom and DESeq2. This is likely due to the high inferential variability which is not being properly assessed and adjusted for by those programs. The transcript name can be pasted into the "transcript view" window and the distribution of inferential variability can be explored with boxplots describing the variability within each sample (Figure 3.5c).



principal component analysis

PCA projections of sample abundances onto any pair of components.

x-axis PC: 1

y-axis PC: 2

color by: strain

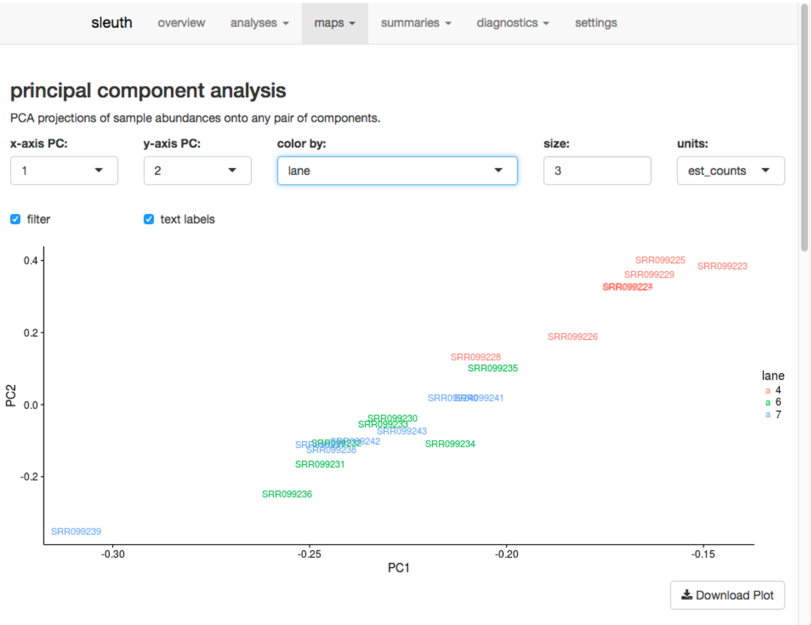
size: 3

units: est_counts

☑ filter

☑ text labels

(a) PCA plot colored by strain shows that the strain does not explain much of the variance in the first two principal components.



principal component analysis

PCA projections of sample abundances onto any pair of components.

x-axis PC: 1

y-axis PC: 2

color by: lane

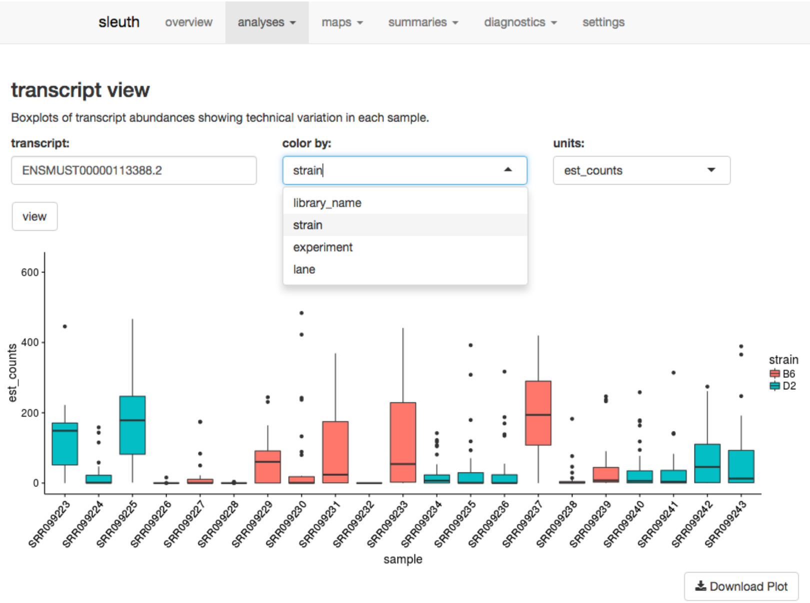
size: 3

units: est_counts

☑ filter

☑ text labels

(b) The coloring can be changed immediately by drop-down as shown here which indicates that there are possible lane effects.



transcript view

Boxplots of transcript abundances showing technical variation in each sample.

transcript: ENSMUST00000113388.2

color by: strain

units: est_counts

(c) Sample specific bootstraps for a particular transcript ENSMUST00000113388, which does not show differential expression by sleuth, but shows differential expression by limma and DESeq2. A possible explanation for this is that the inferential variance is quite high.

Figure 3.5: Interactive sleuth live Shiny interface on complete Bottomly data set.

## 3.3 Simulations

A null distribution for transcript abundances was learned from the largest homogeneous population in the GEUVADIS data set: 59 samples of Finnish females [44]. We estimated transcript-level abundances with kallisto, then estimated parameters for negative binomial distributions (using the Cox-Reid dispersion estimator) to model count distributions using DESeq2.

After the null distribution was constructed, expression features (isoforms or genes depending on the type of simulation) were chosen to be differentially expressed. Transcripts with less than 5 estimated counts on average across the GEUVADIS samples were marked as too rare to be simulated as differentially expressed. A gene was assumed to pass the filter if at least one of its constituent transcripts passed the filter. In each simulation, 20% of the features that passed the filter were chosen to be differentially expressed at random. If the simulation had unequal size factors, random size factors were chosen from the set 1/3, 1, 3 such that the geometric mean equaled 1 similar to the simulation procedure in DESeq2. However, unlike the DESeq2 simulation procedure our size factors were chosen at random. Counts were generated from the negative binomial distribution after which reads were simulated using the RSEM simulator [50]. This resulted in about 30 million 75 base-pair paired-end reads per sample for a total of 13.8 billion reads overall (Tables 3.2 - 3.5). Three types of simulations were performed:

## Independent effect simulation

Isoforms were chosen to be differentially expressed at random. This simulation was similar to the default in polyester [26], however our simulation was performed across the entire transcriptome rather than just a few chromosomes. The simulations were generated with equal size factors. Effect sizes were chosen from a truncated normal distribution such that the minimum absolute fold change for differential transcripts or genes was 1.5.

## Correlated effect simulation

Genes (instead of isoforms) were randomly chosen to be differentially expressed. A direction (sign) for each effect size was chosen at random, then all the effects were simulated from a truncated normal with minimum absolute fold change 1.5. The simulation used random unequal size factors generated as described above.

## Effect from experiment

To mimic the types of changes seen in real experiments, fold changes were learned from Trapnell et al. [85] from the set of transcripts that either DESeq2 or sleuth found to be differentially expressed at FDR 0.05. Genes were chosen at random to be differentially expressed. The null mean counts were used to determine the rank of each transcript relative to its parent gene. These ranks were matched between the Trapnell data set and the null distribution learned from the GEUVADIS data set.

## Self-consistency experiment

In order to validate whether methods would produce similar results with less data, we performed an experiment similar to Love et al. [56]. For each iteration we randomly selected 3 samples from condition C57BL/6J and 3 samples from condition DBA/2J and ran each tool. The remaining samples were used as the “truth” by calling differentially expressed genes or transcripts using them. For each FDR level (0.01, 0.05, 0.10), we compared the results from the smaller data set to the larger data set for each tool. The FDR was then computed and plotted in Figure 3.4.

## Software notes

The following R programs were used to compile the results: sleuth 0.28.1, BitSeq 1.16.0, DESeq 0.24.0, DESeq2 1.12.0, EBSeq 1.12.0, edgeR 3.14.0, limma-voom 3.28.2. When testing programs at the isoform-level, kallisto 0.42.4 was used to obtain quantifications. Cuffdiff 2.21 was used with alignments from HISAT2 2.0.1 [40]. Subread (featureCounts) 1.5.0 [53] was used with alignments from HISAT2 to get raw gene counts. BitSeq was provided alignments from Bowtie 1.1.2 [43]. All analyses in the paper are fully reproducible through the Snakemake system [42].

## 3.4 The model

We use the term *experiment* to denote the measurement of transcript abundances from a series of $n$ samples which are related by an $n \times p$ design matrix $x$. Each row vector $x_i$ ($i = 1, \ldots, n$) of the matrix $x$ records the fixed design characteristics of sample $i$ with respect to the $p$ covariates.

For each transcript $t$ and sample $i$, we model the logarithm of transcript abundance (measured in counts) with a latent random variable $Y_{ti}$. A vector $\beta_t$ of length $p$ associates fixed effects to each transcript, and "biological noise" $\epsilon_{ti}$ perturbs the response $x_i^T \beta_t$ so that

$$Y_{ti} \mid x_i = x_i^T \beta_t + \epsilon_{ti}. \tag{3.1}$$

While many RNA-Seq models posit that $Y_{ti}$ is observed, the ambiguity of read (pseudo)-alignments means that instead what is measured is

$$D_{ti} \mid y_{ti} = y_{ti} + \xi_{ti}, \tag{3.2}$$

where $\xi_{ti}$ is "inferential noise".

Assuming that $\epsilon_{ti}$ and $\xi_{ti}$ are random variables satisfying $\epsilon_{ti} \sim \mathcal{N}(0, \sigma_t^2), \xi_{ti} \sim \mathcal{N}(0, \tau_t^2)$, $\mathrm{cov}(\epsilon_{ti}, \epsilon_{tj}) = \mathrm{cov}(\xi_{ti}, \xi_{tj}) = 0 \ \forall i \neq j, \mathrm{cov}(\epsilon_{ti}, \xi_{tj}) = 0 \ \forall i, j$ and that $\forall t \neq u$, $\epsilon_t, \xi_t$ are independent of $\epsilon_u$ and $\xi_u$ respectively, we have that $Y_t = (Y_{t1}, Y_{t2}, ..., Y_{tn})$ and $D_t = (D_{t1}, D_{t2}, ..., D_{tn})$ are both normally distributed as

$$Y_t \sim \mathcal{N}(x\beta_t, \sigma_t^2 I_n), \tag{3.3}$$

$$D_t \sim \mathcal{N}(x\beta_t, (\sigma_t^2 + \tau_t^2) I_n). \tag{3.4}$$

This model is known as the *response error measurement model* with no error on the covariates [13]. For completeness, we describe some of its properties below and explain how they apply to parameter estimation in the context of the sleuth workflow.

## 3.5 Overview of sleuth workflow

The input to sleuth consists of estimated counts for transcripts in the samples constituting the experiment as well as estimates of variance for those counts obtained from bootstraps. Both the estimated counts and the variance are output by kallisto. The (estimated) counts for transcript $t$ in sample $i$ is referred to as $c_{ti}$ and the variance of $D_{ti}$ given $y_{ti}$ estimated from the bootstraps of kallisto. The sleuth workflow begins with a filtering of low abundance transcripts, followed by the application of two normalizations and then parameter estimation for the model described above. This enables the regularization of the biological variance contributing to transcript abundance variance across samples, and finally to an overall total variance estimate for each transcript. The workflow can be applied to either transcripts, or groups of transcripts such as genes, and the two modes are described below.

## 3.6 Filtering prior to parameter estimation

Prior to estimating parameters of the model we filter low abundance transcripts. This helps in fitting the model. We ignore transcripts where there are less than 5 estimated counts in more than 47% of the samples, i.e. when $|\{i : c_{ti} < 5\}| \geq 0.47 \cdot n$.

## 3.7 Normalization and transformation

Following the filtering there are two different normalizations that we apply to the estimated counts $c_{ti}$: between sample normalization and within sample normalization. First, we perform between sample normalization to estimate sample specific size factors $s_i$ on the estimated counts following the DESeq procedure [4] applied to transcripts:

$$\hat{s}_i = \text{median}_t \frac{c_{ti}}{\left(\prod_{j=1}^{n} c_{tj}\right)^{\frac{1}{n}}}.$$

Following between sample normalization, we log transform the data so that transcripts have similar variance across samples.

For each transcript, the abundance is estimated as the (normalized) log estimated count

$$\begin{aligned} d_{ti} &= \log\left(\frac{1}{\hat{s}_i}\tilde{l}_{ti}\frac{c_{ti}}{\tilde{l}_{ti}} + 0.5\right) \\ &= \log\left(\frac{1}{\hat{s}_i}c_{ti} + 0.5\right), \end{aligned}$$

where $\tilde{l}_{ti}$ is the effective length of transcript $t$ in sample $i$. Note that the expression $\frac{c_{ti}}{\tilde{l}_{ti}}$is proportional to the abundance of transcript $t$ in sample $i$, and that the multiplication by the effective length serves to rescale the abundance estimate to a count estimate. The offset of 0.5 is used to ensure that the argument to the logarithm is positive.

## 3.8 Estimation of $\beta_t$

Conveniently, the standard ordinary least squares estimators for the fixed effects are unbiased under this model. The standard estimator is

$$\hat{\beta}_t = (x^T x)^{-1} x^T d_t \tag{3.5}$$

where $d_t = (d_{t1}, \ldots, d_{tn})$. The expected value of $\hat{\beta}_t$ is

$$\begin{aligned}
\mathbb{E}[\hat{\beta}_t] &= \mathbb{E}[\mathbb{E}[(x^T x)^{-1} x^T d_t \mid y_t]] \\
&= \mathbb{E}[(x^T x)^{-1} x^T \mathbb{E}[y_t + \xi_t | y_t]] \\
&= \mathbb{E}[(x^T x)^{-1} x^T (y_t + 0)] \\
&= \mathbb{E}[(x^T x)^{-1} x^T (x\beta_t + \epsilon_t)] \\
&= (x^T x)^{-1} x^T \mathbb{E}[(x\beta_t + \epsilon_t)] \\
&= (x^T x)^{-1} x^T (x\beta_t + 0) \\
&= \beta_t.
\end{aligned}$$

Thus $\hat{\beta}_t$ is an unbiased estimator of $\beta_t$.

## 3.9 Estimation of the variance of $D_{ti}$

The variance of $D_{ti}$ decomposes according to the law of total variance:

$$\begin{aligned}
\mathbb{V}[D_{ti}] &= \mathbb{E}[\mathbb{V}[D_{ti} \mid y_{ti}]] + \mathbb{V}[\mathbb{E}[D_{ti} \mid y_{ti}]] \\
&= \mathbb{E}[\mathbb{V}[y_{ti} + \xi_{ti} \mid y_{ti}]] + \mathbb{V}[y_{ti}] \\
&= \mathbb{E}[\tau_t^2] + \sigma_t^2 \\
&= \tau_t^2 + \sigma_t^2.
\end{aligned}$$

The inferential variance $\tau_t^2$ is estimated as the mean of the sample variance estimates $\hat{\tau}_{ti}^2$ which are obtained from kallisto with the bootstrap:

$$\hat{\tau}_t^2 = \frac{1}{n} \sum_i \hat{\tau}_{ti}^2. \tag{3.6}$$

Using the second moment as an estimator for the total variance, namely

$$\hat{\mathbb{V}}[D_{ti}] = \frac{1}{n-p} \sum_{i=1}^{n} (d_{ti} - x_i^T \hat{\beta}_t)^2$$

and solving for the (raw) biological variance, we obtain

$$\hat{\sigma}_t^2 = \max\left( \left( \frac{1}{n-p} \sum_{i=1}^{n} (d_{ti} - x_i^T \hat{\beta}_t)^2 \right) - \hat{\tau}_t^2, 0 \right), \tag{3.7}$$

where the max operation is necessary to ensure that the (raw) biological variance is nonnegative.

When $n - p$ is small, the (raw) biological variance estimate $\hat{\sigma}_t^2$ is unstable. Since this is the situation in almost all RNA-Seq studies, we regularize the biological variance estimate by shrinkage. We split the abundance values into 100 "windows" (ranges) such that each window, $w$, contains 1% of the mean abundance under the intercept only model. Note that each transcript $t$ has an abundance contained in a single window denoted by $w(t)$ and a distribution of estimated biological variance is associated to that window, namely of $\hat{\sigma}_t^2$. We denote by $IQR(w(t))$ the interquartile range of the distribution associated to a window $w(t)$ and identify a training set of transcripts $R$ for shrinkage using these interquantile ranges:

$$R = \{t : \hat{\sigma}_t^2 \in IQR(w(t)).\} \tag{3.8}$$

We perform LOESS on the set $R$ and perform shrinkage on the square root of the standard deviation similar to voom [45] as this results in more stable estimates. Our shrunken estimate of $\sigma_t^2$ is then a function of the mean $\bar{d}_t = \frac{1}{n}\sum_{i=1}^n d_{ti}$ (the parameter estimate under the intercept only model):

$$\tilde{\sigma}_t^2 = f(\bar{d}_t) = \left[(\text{loess}_{r\in R}(\bar{d}_r, \hat{\sigma}_r^{\frac{1}{2}}))(\bar{d}_t)\right]^4. \tag{3.9}$$

Our final estimate of the total variance of transcript $t$ in sample $i$ is therefore (the sample independent expression)

$$\hat{\mathbb{V}}[D_{ti}] = \max(\tilde{\sigma}_t^2, \hat{\sigma}_t^2) + \hat{\tau}_t^2. \tag{3.10}$$

## 3.10 Gene level estimates

The sleuth model for transcript abundance, and the associated parameter estimation described above can be generalized to groups of transcripts such as genes. To do so, we first note that a set of genes can be viewed as a partition of the set of transcripts, so that each gene $g$ is just a set of transcripts. To model gene abundances, we replace transcript abundance with gene abundance in the model as follows:

Starting with the same design matrix $x$ as in the transcript case, for each gene $g$ and sample $i$, we model the logarithm of transcript abundance (measured in counts) with a latent random variable $Y_{gi}$. A vector $\beta_g$ of length $p$ associates fixed effects to each gene, and "biological noise" $\epsilon_{gi}$ perturbs the response $x_i^T\beta_g$ so that

$$Y_{gi} \mid x_i = x_i^T\beta_g + \epsilon_{gi}. \tag{3.11}$$

As before, we use a response error measurement model based on underlying normality assumption which leads to

$$Y_g \sim \mathcal{N}(x\beta_g, \sigma_g^2 I_n), \tag{3.12}$$

$$D_g \sim \mathcal{N}(x\beta_g, (\sigma_g^2 + \tau_g^2)I_n). \tag{3.13}$$

The workflow at the gene level is identical to that of transcript level analysis, with a few minor differences:

1. At the gene level, if at least one isoform in that gene passes the filter, the entire gene passes the filter.

2. The normalization at the gene level is analogous to that at the transcript level except for two differences: the abundance of genes is first calculated by summing up the abundances of the constituent isoforms and the effective length of a single transcript is replaced by an effective length for the gene (consisting of the median of the effective lengths of the constituent transcripts). For a gene $G$ the normalized estimate for abundance in "effective counts" is therefore

$$d_{gi} = \log\left(\frac{1}{\hat{s}_i}(\text{median}_{t \in G}\, \tilde{l}_{ti}) \sum_{t \in G} \frac{c_{ti}}{\tilde{l}_{ti}} + 0.5\right).$$

3. The shrinkage procedure is applied at the gene level, leading to a total variance estimate of
$$\hat{\mathbb{V}}[D_{gi}] = \max(\tilde{\sigma}_g^2, \hat{\sigma}_g^2) + \hat{\tau}_g^2,$$
where the estiamtes of $\hat{\sigma}_g^2$ and $\hat{\tau}_g^2$ are analogous to their transcript counterparts.

Note that $d_{gi}$ reduces to $d_{ti}$ when $g$ consists of just a single isoform, so that the workflow can be viewed as a direct generalization of the transcript level case. Moreover, the procedure outlined above can be applied to sets of transcripts obtained from any partition of the transcriptome.

## 3.11 Description of filters of benchmarked programs

In this section we describe the filtering procedures of the programs benchmarked (in a few cases methods did not specify filtering procedures so we selected one for them). Table 3.1 shows the filters used at the transcript and gene level:

| **method** | **isoform mode filter** | **gene mode filter** |
|---|---|---|
| Cuffdiff 2 | Cuffdiff 2 | Cuffdiff 2 |
| DESeq | DESeq | DESeq |
| DESeq2 | DESeq2 | DESeq2 |
| edgeR | edgeR | edgeR |
| EBSeq | sleuth | edgeR |
| GLFC | sleuth | edgeR |
| LFC | sleuth | edgeR |
| sleuth | sleuth | sleuth |
| voom | sleuth | edgeR |

Table 3.1: The filters used with each program.

### Cuffdiff 2

The default filter for the program is described as: "The minimum number of alignments in a locus for needed to conduct significance testing on changes in that locus observed between samples. If no testing is performed, changes in the locus are deemed not significant, and the locus' observed changes don't contribute to correction for multiple testing. The default is 10 fragment alignments." [84]

### DESeq

The DESeq vignette [3] describes a filter discarding the lowest 40% of expressed features, where expression is defined as the total number of counts across all experiments. In some cases more than 40% of the features were lowly expressed so we implemented a slightly modified version that first applied the DESeq2 filter.

```
DESeq_filter <- function(mat, ...) {
  # a modified version of the DESeq filter to first remove things that are 0
  # before doing the quantile filter
  nonzero <- DESeq2_filter(mat)
  rs <- rowSums(mat[nonzero, ])
  theta <- 0.4
  use <- (rs > quantile(rs, probs=theta))
  ret <- nonzero
  ret[nonzero] <- use
  ret
}
```

## DESeq2

The filter is described in the DESeq2 vignette [55]. It removes features whose total counts across all experiments is less than 2:

```
DESeq2_filter <- function(mat, ...) {
  rowSums(mat) > 1
}
```

## edgeR

The filter is described in the edgeR vignette [18]. It removes features where less than 2 experiments contain less than or equal to 1 count per million:

```
edgeR_filter <- function(mat, ...) {
  rowSums(cpm(mat) > 1) >= 2
}
```

## EBSeq

Based on the EBSeq vignette [47] we decided to use the sleuth filter at the isoform level and edgeR filter at the gene level.

### voom

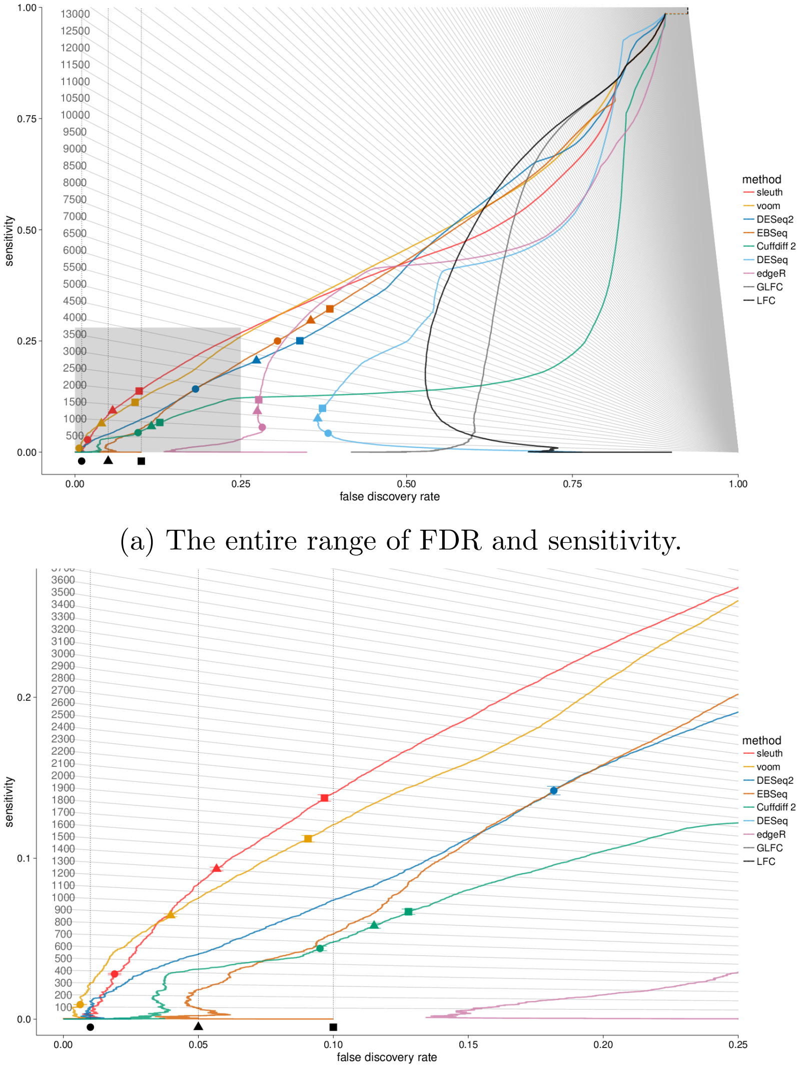
The voom vignette [81] states "The limma-voom method assumes that rows with zero or very low counts have been removed", so we decided to use the sleuth filter at the isoform level and edgeR at the gene level.

## 3.12 Performance on independent and correlated effect simulations

In this section we show how each of the method benchmarked performs on the independent effect and correlated effect simulations with the filtering procedures described in Section 8.

## Independent effect simulation

### Isoform level

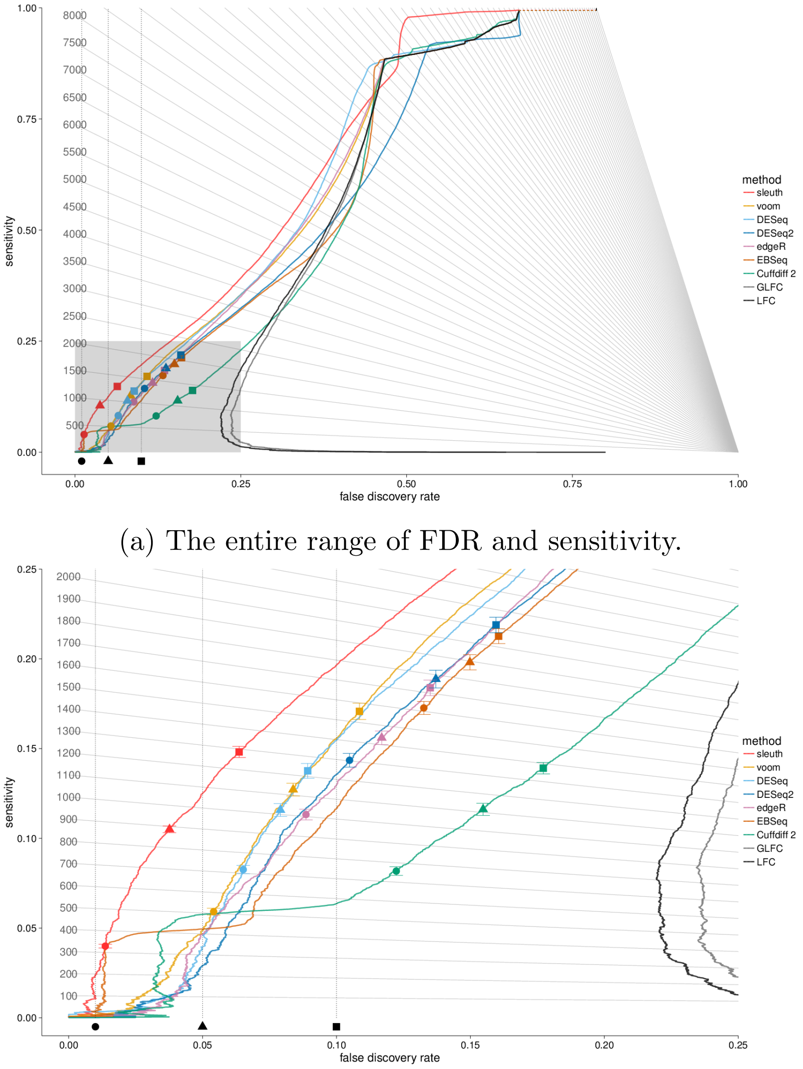


(a) The entire range of FDR and sensitivity.

(b) Zoomed in version of the gray section from (a).

Figure 3.6: Sensitivity versus FDR in the "independent effect" simulation at the isoform level.

## Gene level



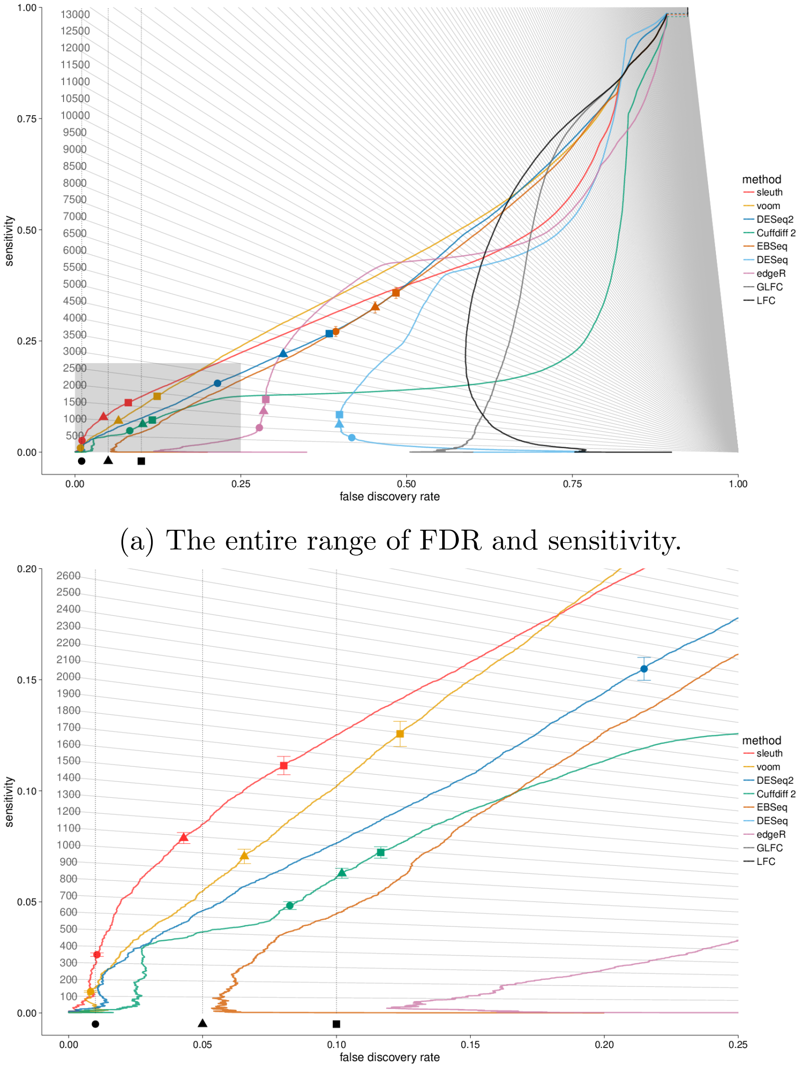
(a) The entire range of FDR and sensitivity.

(b) Zoomed in version of the gray section from (a).

Figure 3.7: Sensitivity versus FDR in the “independent effect” simulation at the gene level.

## Correlated effect simulation

### Isoform level

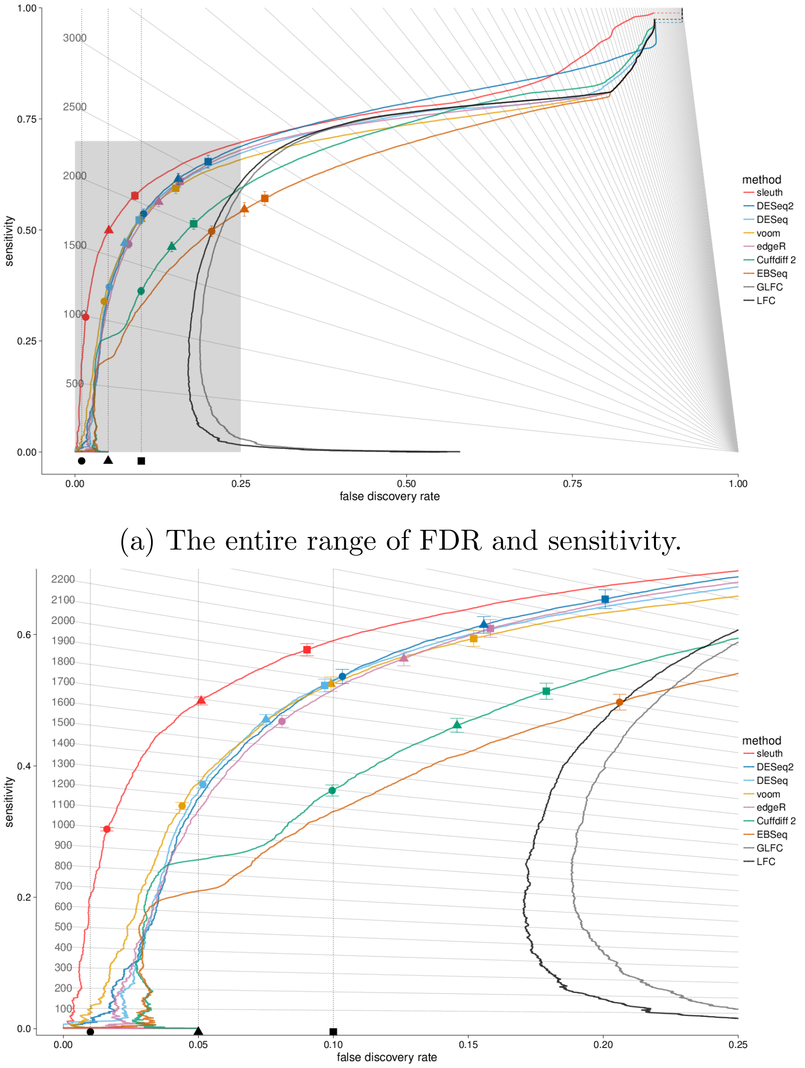


(a) The entire range of FDR and sensitivity.

(b) Zoomed in version of the gray section from (a).

Figure 3.8: Sensitivity versus FDR in the “correlated effect” simulation at the isoform level.

## Gene level



(a) The entire range of FDR and sensitivity.

(b) Zoomed in version of the gray section from (a).

Figure 3.9: Sensitivity versus FDR in the "correlated effect" simulation at the gene level.

## 3.13 Performance with common filtering

This section shows the comparison of sleuth to other methods when each pair (sleuth and another method) are tested with a common filter based on intersecting the filtering criteria of both programs. In each case, the two methods were trained using only the data passing both filters. This results in higher power for both methods as there are less tests and fewer low count, high variance targets disrupting the shrinkage estimation.

## Independent effect simulation

### Isoform level

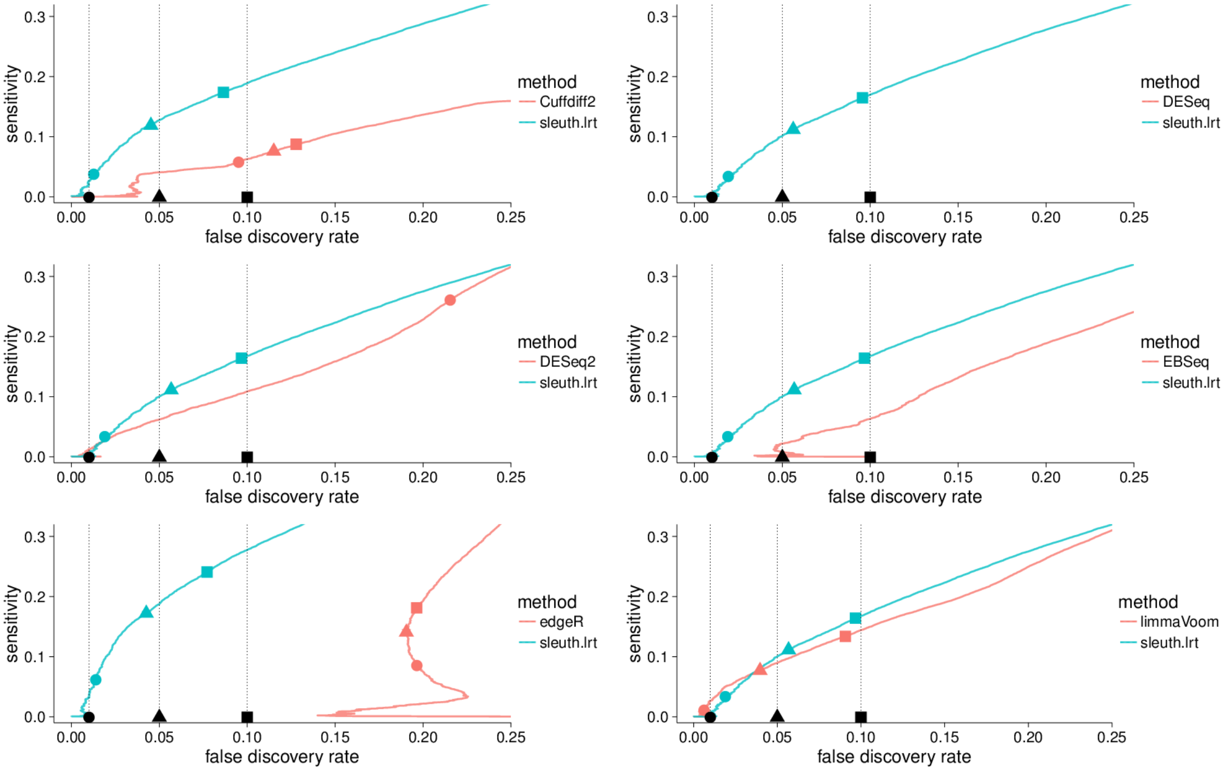


Figure 3.10: Pairwise comparisons in the independent effect simulation at isoform level with common filtering (DESeq did not register a datapoint in the FDR-sensivity range).

### Gene level

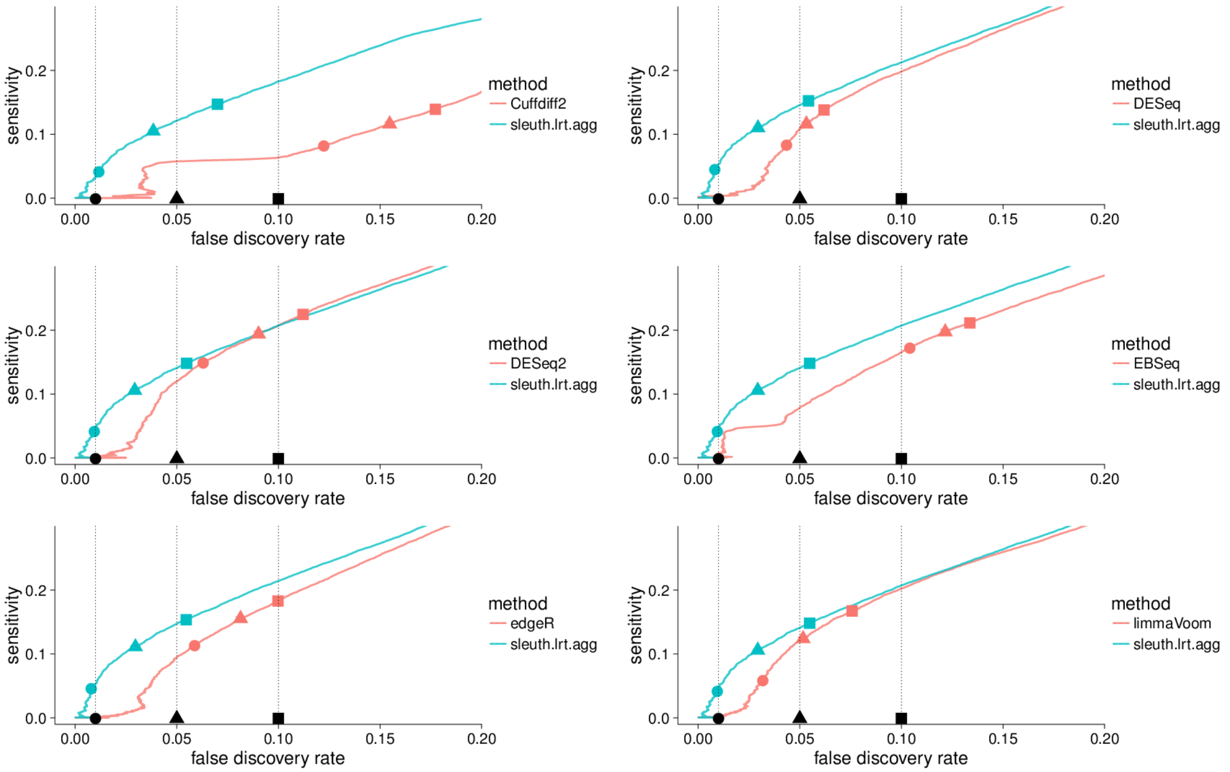


Figure 3.11: Pairwise comparisons in the independent effect simulation at gene level with common filtering.

## Correlated effect simulation

### Isoform level

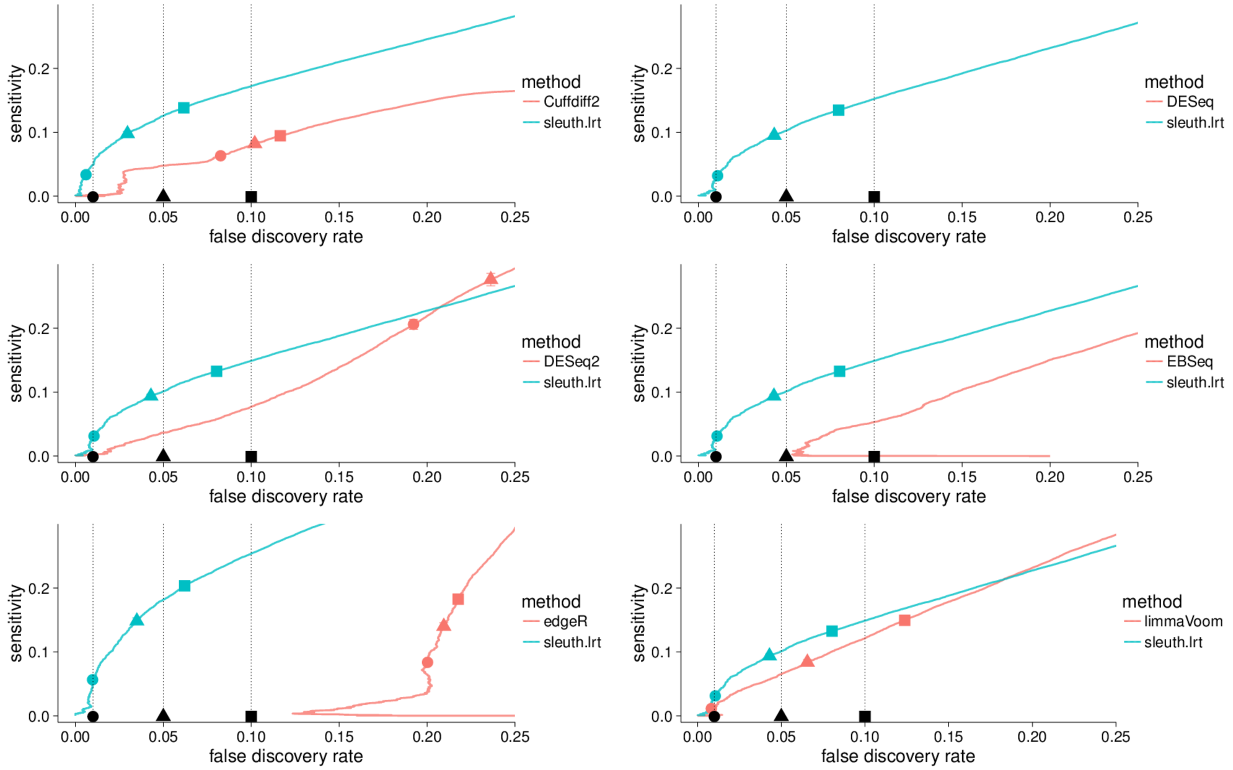


Figure 3.12: Pairwise comparisons in the correlated effect simulation at isoform level with common filtering (DESeq did not register a datapoint in the FDR-sensivity range).

### Gene level

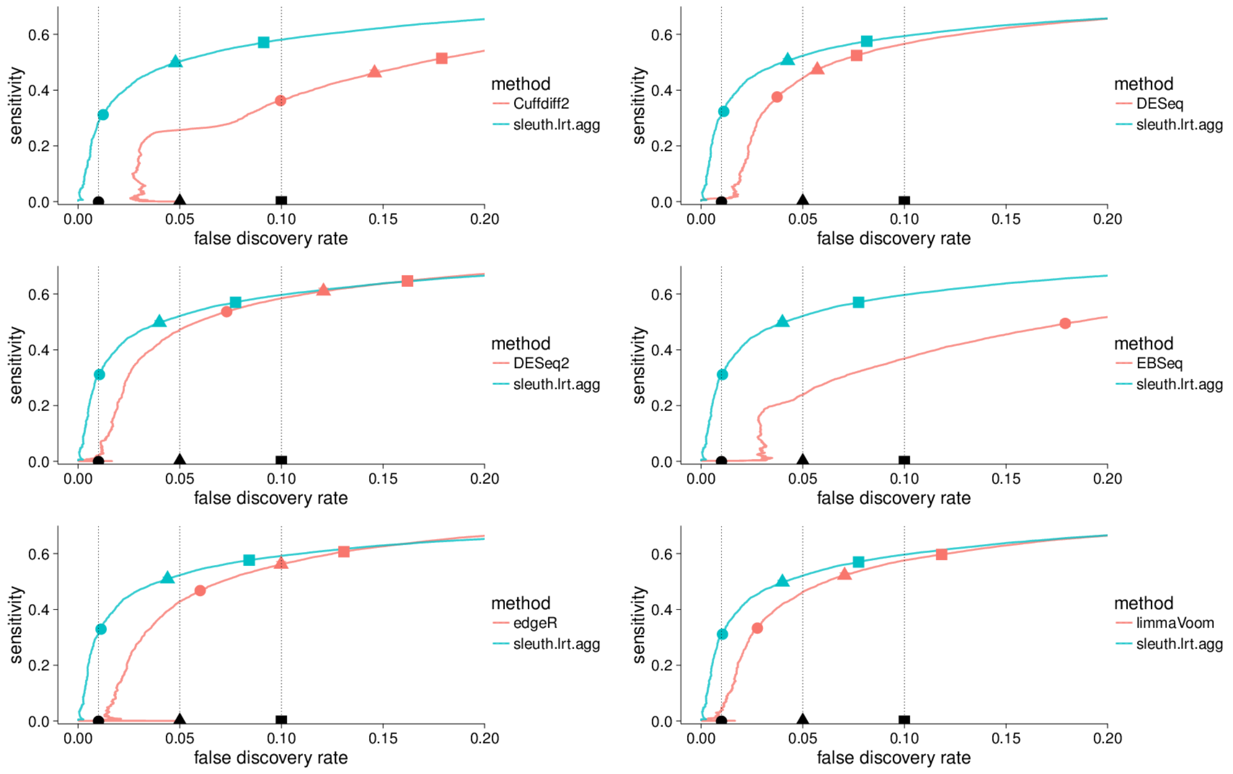


Figure 3.13: Pairwise comparisons in the correlated effect simulation at gene level with common filtering.

## Effect from experiment simulation

### Isoform level

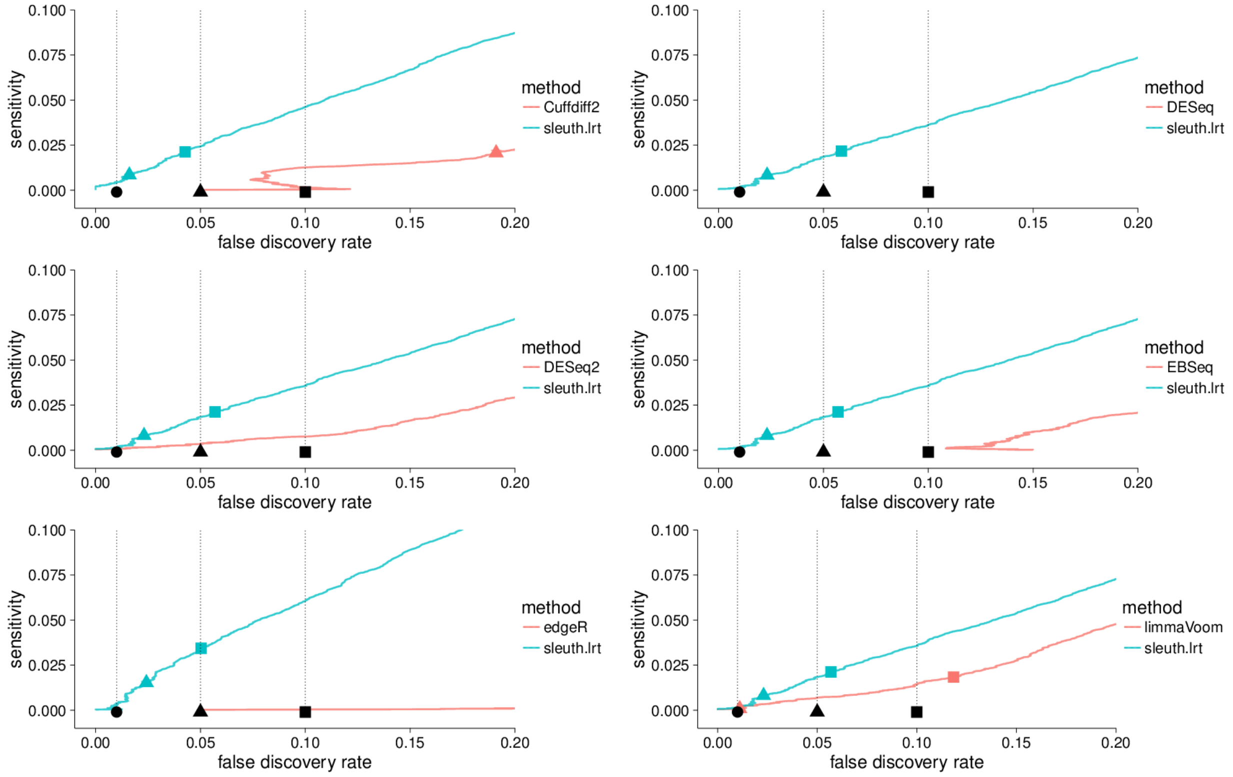


Figure 3.14: Pairwise comparisons in the effect from experiment simulation at isoform level with common filtering (DESeq did not register a datapoint in the FDR-sensivity range).

### Gene level

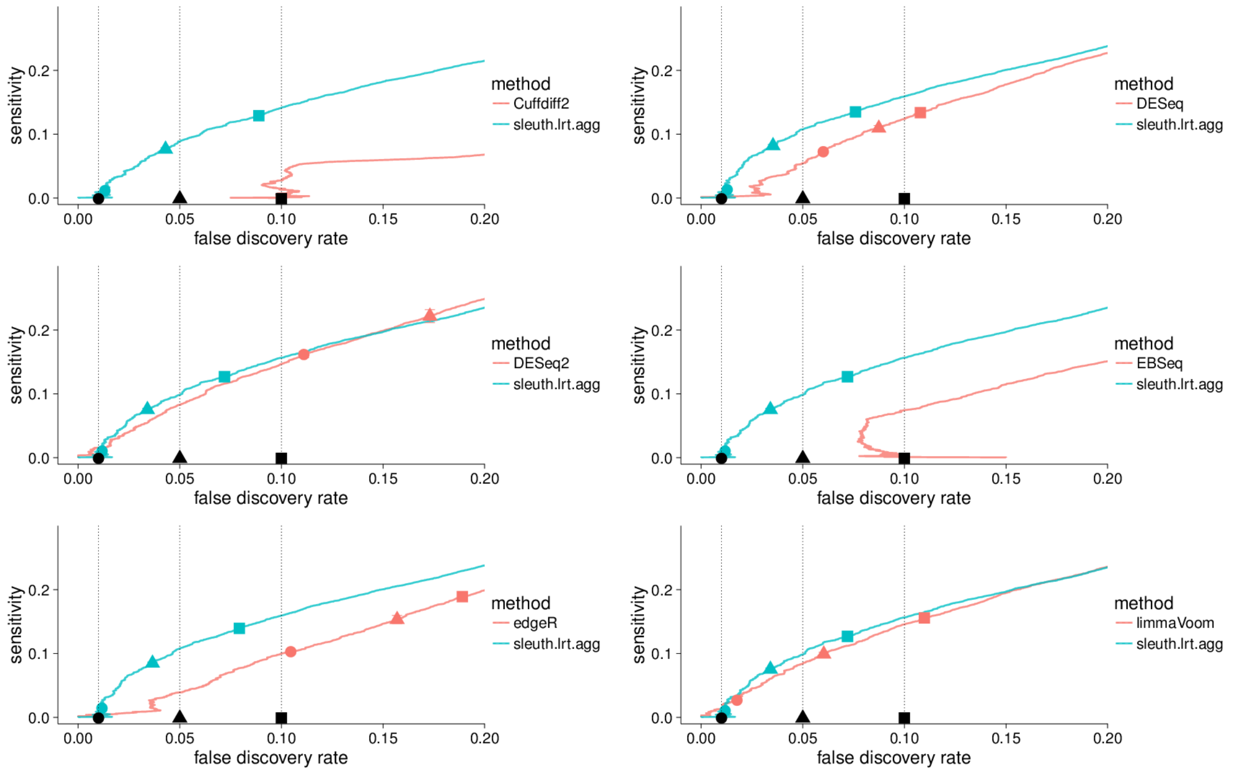


Figure 3.15: Pairwise comparisons in the effect from experiment simulation at gene level with common filtering.

## 3.14 Number of reads simulated

Below are tables listing the total number of reads per each sample for each different type of simulation.

| | A1 | A2 | A3 | B1 | B2 | B3 |
|---|---|---|---|---|---|---|
| 1 | 25804803 | 25470509 | 26167030 | 29341697 | 29519936 | 29488175 |
| 2 | 25885277 | 25249979 | 25389240 | 32370503 | 30343889 | 30738282 |
| 3 | 26120446 | 25827345 | 25899623 | 30713524 | 30243242 | 30751697 |
| 4 | 25358596 | 25751655 | 25122351 | 33060914 | 32410330 | 32989822 |
| 5 | 25256842 | 25311042 | 25593932 | 31890641 | 30658059 | 30919290 |
| 6 | 25889040 | 25051039 | 26038569 | 29281347 | 28643895 | 28669578 |
| 7 | 25999345 | 25722280 | 25292680 | 30984262 | 30684305 | 30958786 |
| 8 | 25601571 | 25532946 | 25772470 | 30649733 | 30345697 | 30902790 |
| 9 | 25753144 | 25161751 | 26544777 | 31186168 | 29804316 | 30045082 |
| 10 | 26108266 | 25569269 | 25855553 | 32841294 | 32708783 | 33222170 |
| 11 | 25742811 | 25627297 | 25596448 | 29374021 | 28834454 | 28869359 |
| 12 | 25364068 | 25137212 | 25946183 | 29745829 | 29789553 | 30543497 |
| 13 | 25634414 | 25444127 | 25682130 | 27982129 | 29518922 | 30585275 |
| 14 | 25090260 | 25645622 | 25095267 | 30182778 | 29152706 | 29049277 |
| 15 | 25835121 | 25671692 | 25224938 | 30470838 | 30650333 | 30239396 |
| 16 | 25852931 | 26191755 | 25509063 | 31266671 | 29757417 | 29669651 |
| 17 | 25615156 | 25575068 | 25406585 | 29418611 | 30004286 | 29928450 |
| 18 | 25524970 | 25356389 | 26081851 | 31500275 | 30436352 | 30703447 |
| 19 | 26546122 | 25225294 | 25908252 | 30499047 | 30497149 | 31153162 |
| 20 | 25949790 | 25832498 | 26399110 | 30414105 | 29711824 | 29007750 |

Table 3.2: Total number of reads for each sample in the independent effect simulation.

| | A1 | A2 | A3 | B1 | B2 | B3 |
|---|---|---|---|---|---|---|
| 1 | 75312822 | 25593373 | 25235809 | 9839136 | 9929876 | 90460751 |
| 2 | 76754874 | 77164818 | 8448302 | 10128818 | 91488796 | 10260035 |
| 3 | 25605293 | 8695919 | 25491639 | 88543891 | 91209909 | 10263281 |
| 4 | 78615038 | 8506970 | 8385197 | 9938514 | 92448519 | 89436730 |
| 5 | 77364271 | 74865995 | 8528707 | 10076750 | 10042850 | 88754945 |
| 6 | 25860713 | 76275460 | 8440966 | 9610553 | 89957747 | 29271608 |
| 7 | 8597415 | 25281982 | 75251435 | 10213462 | 30323374 | 95540321 |
| 8 | 25522744 | 76270625 | 25088664 | 10137245 | 10421655 | 91773477 |
| 9 | 8666259 | 76943845 | 8559758 | 9987635 | 88007539 | 89331914 |
| 10 | 8566416 | 25410620 | 8490243 | 30555362 | 90843296 | 91010457 |
| 11 | 8455041 | 75784558 | 25372293 | 30267698 | 87570393 | 9787472 |
| 12 | 8492502 | 78575682 | 77077410 | 29654868 | 29818839 | 9886687 |
| 13 | 25457086 | 26158929 | 8471727 | 29118444 | 29471345 | 90230274 |
| 14 | 26002373 | 26260610 | 25956110 | 97485646 | 31266203 | 10420089 |
| 15 | 25487509 | 8493425 | 25243386 | 87415490 | 88713112 | 9713101 |
| 16 | 77486094 | 8448668 | 8663999 | 29099399 | 86839687 | 28969442 |
| 17 | 8407592 | 25888803 | 76954183 | 10420214 | 30992646 | 94468557 |
| 18 | 25873615 | 76371611 | 8359367 | 31089875 | 10122162 | 88988075 |
| 19 | 25678980 | 8501752 | 76183257 | 10537835 | 91483251 | 32587127 |
| 20 | 8454584 | 25950638 | 78035981 | 10428599 | 30550328 | 91394732 |

Table 3.3: Total number of reads for each sample in the correlated effect simulation.

| | A1 | A2 | A3 | B1 | B2 | B3 |
|---|---|---|---|---|---|---|
| 1 | 75168780 | 25485734 | 8384121 | 25747475 | 8534405 | 77027331 |
| 2 | 8501164 | 76807800 | 77633767 | 8740185 | 8684330 | 79993067 |
| 3 | 25331289 | 76540098 | 78511757 | 8447971 | 8427418 | 25987110 |
| 4 | 8513095 | 77563156 | 76171563 | 8746602 | 8828539 | 78111712 |
| 5 | 25944623 | 25748634 | 78161328 | 8493836 | 77658353 | 8625305 |
| 6 | 74728858 | 25432069 | 78836864 | 8805673 | 8985756 | 26342445 |
| 7 | 25774678 | 76049917 | 25580674 | 8843625 | 9010990 | 81891037 |
| 8 | 8443669 | 25980365 | 76146962 | 26349727 | 78607969 | 8721963 |
| 9 | 8526924 | 25514760 | 25393205 | 8893176 | 81080795 | 79541529 |
| 10 | 76685128 | 25597960 | 8447489 | 25772831 | 8863348 | 79552345 |
| 11 | 26124390 | 76923452 | 8679035 | 26027252 | 26363462 | 26557951 |
| 12 | 77306728 | 8528461 | 8413117 | 8841991 | 80696745 | 78431231 |
| 13 | 8467790 | 8686289 | 75067527 | 78678591 | 8653662 | 78027739 |
| 14 | 25726894 | 76768529 | 8477740 | 79007076 | 27257417 | 8990530 |
| 15 | 8317198 | 26005481 | 76363404 | 8420821 | 76489073 | 26140406 |
| 16 | 77110090 | 76973967 | 8803533 | 8982341 | 8838866 | 78798474 |
| 17 | 8595280 | 8660676 | 78462128 | 9011452 | 79474418 | 78080390 |
| 18 | 78598555 | 8504087 | 76359175 | 8780854 | 80048666 | 8739335 |
| 19 | 76495399 | 25188780 | 8438441 | 25570493 | 25599853 | 25619140 |
| 20 | 74677076 | 8417349 | 25938160 | 8933135 | 26341988 | 79293035 |

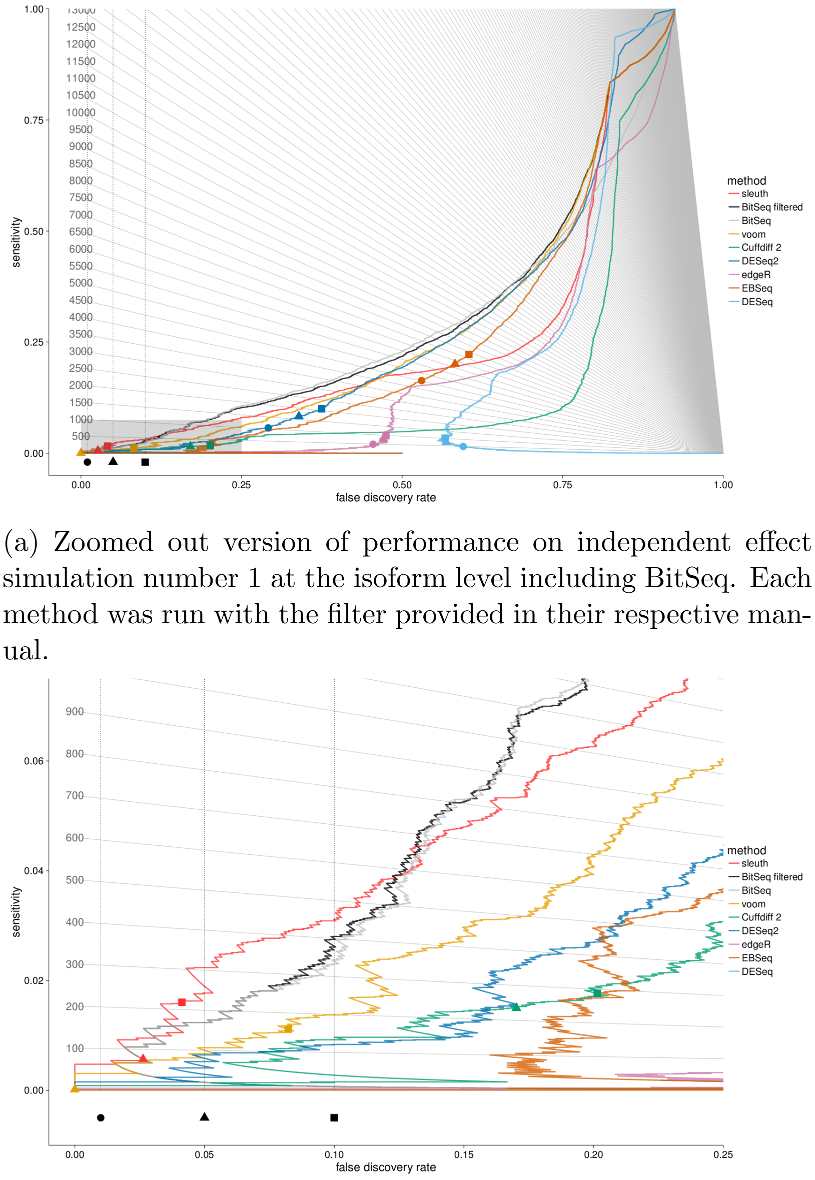
Table 3.4: Total number of reads for each sample in the effect from experiment simulation.

| | | | |
|---|---|---|---|
| A1 | 76567345 | B1 | 77353279 |
| A2 | 25501413 | B2 | 8851265 |
| A3 | 25201440 | B3 | 26670650 |
| A4 | 8629446 | B4 | 77175275 |
| A5 | 77679146 | B5 | 8502371 |
| A6 | 76799156 | B6 | 26087240 |
| A7 | 8496819 | B7 | 25635027 |
| A8 | 77535590 | B8 | 8644561 |
| A9 | 8575806 | B9 | 26012768 |
| A10 | 25820678 | B10 | 79047338 |
| | | B11 | 8562183 |

Table 3.5: Total number of reads for each sample in the effect from experiment simulation used in the self-referential FDR experiment.

## 3.15 Effect from experiment - BitSeq

We only ran BitSeq on one sample due to the long run time. BitSeq does not allow a external filtering method, but we attempted to increase power by intersecting the results with the sleuth filter (BitSeq filtered).



(a) Zoomed out version of performance on independent effect simulation number 1 at the isoform level including BitSeq. Each method was run with the filter provided in their respective manual.

(b) Zoomed in version of the gray section from (a).

Figure 3.16: Sensitivity versus FDR in the "effect from experiment" simulation 1 at the isoform level including BitSeq.

## 3.16 Discussion

RNA-Seq experiments produce data that is complex in structure and rich in information. This data presents an unprecedented opportunity for studying transcriptional mechanisms and, but its analysis is fraught with challenges. In the case of differential expression, the large volumes of data and the large sizes of transcriptomes have made it difficult to explore results "by hand" in order to gain intuition and insight into the experiments. "Count-based" methods for differential analysis have been popular partly because they are simple in their approach and present researchers with numbers to examine that are easy to relate to. However the simplicity of count-based methods comes at a cost: by ignoring the complexity of ambiguously mapped reads they introduce biases that can have detrimental effects on results [85].

The sleuth model provides a solution to a perplexing difficulty in RNA-Seq analysis: it offers a simple yet powerful framework for "counting" even when reads cannot be assigned unambiguously to transcripts, and therefore allows for robust and accurate RNA-Seq analyses. Our results show that by virtue of appropriately accounting for uncertainty in quantifications, sleuth is more accurate than previous approaches at both the gene and isoform levels. Crucially, the estimated FDRs reported by sleuth reflect the true FDRs, and therefore make the predictions of sleuth reliable and useful in practice.

The sleuth workflow has been deliberately designed to be simple so that it is interpretable and fast. The model was chosen in part because of its tractability and the Shiny framework for visualization was chosen for its portability. The modularity of the algorithm also makes it easy to explore improvements and extensions, such as analysis of more general transcript groups (e.g. as defined by shared exons, or 5'/3' UTRs) and different shrinkage and normalization schemes to improve performance. As a result, when coupled with kallisto, which has dramatically reduced running times for quantification based on the idea of pseudoalignment, sleuth is a quick, accurate, and versatile tool for the analysis of RNA-Seq data.

# Chapter 4

# Intron retention

Many organisms exhibit intron retention events that can be measured with RNA-Seq [14, 10], and recent publications suggest that these events are important constituents of transcriptome regulation. While some existing tools can detect intron retention events [38, 6, 7], none that we are aware of incorporate biological replicates to reduce the reporting of false-positives or discover novel intron retention events. Other tools have been mentioned in the literature, but do not have freely available software [39, 93, 10, 9]. There is therefore a need for a robust intron retention detection method that is based on rigorous quantification of intron retention followed by assessment of significance using biological replicates.

We present **k**eep **m**e **a**round (`kma`), a set of tools for detecting intron retention in RNA-Seq experiments that utilizes biological replicates to improve accuracy. `kma` currently uses the transcript quantification method eXpress [72] due to the fact that it was developed before kallisto, but is compatible with with any RNA-Seq quantification pipeline.

The majority of this chapter is derived from work previously presented in [69] and [67] and is being reproduced with permission of the co-authors.

## 4.1 Estimation

`kma` begins by performing a pre-processing step consisting of several python scripts that find "measurable" intronic regions called inclusion regions (regions in which none of the overlapping isoforms contain an exon), together with the corresponding isoforms which could retain the intron called overlap isoforms. `kma` then outputs a table of intron-transcript relationships. This table includes the (1) intron coordinates, (2) intron quantification coordinates (3) transcripts which could potentially retain the intron, and (4) the gene name. The intron quantification coordinates differ from the exact intron coordinates by including a small region of the neighboring exons which is several bases shorter than the read length (Figure 4.1). This exonic overlap ensures that the reads spanning the intron-exon junctions are included into the intron expression. These reads are often valuable information; if they are unique, they give strong evidence for the expression of the intron. `kma` also outputs a BED

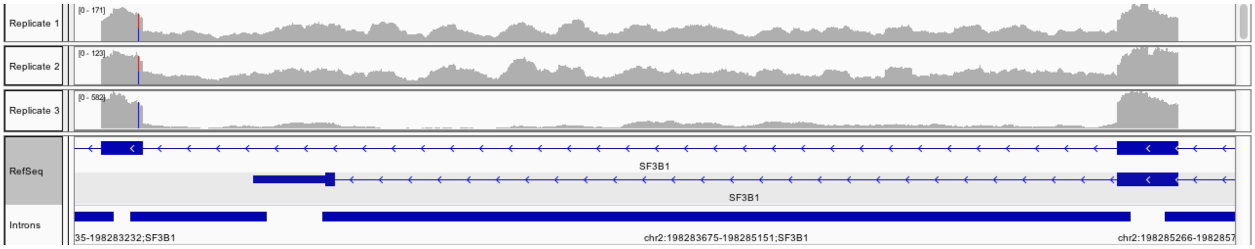


Figure 4.1: Example of intron retention in orthochromatic erythroblasts from [69] in gene SF3B1. The "Intron" track shows the regions of included pseudo-transcripts and that each pseudo-transcript overlaps neighboring exons by 25 bases. The coverage track shows that the first few introns are covered by very few reads, but intron 4 (chr2:198283675-198185151) shows intron retention in all replicates.

track containing intronic quantification coordinates, as well as a FASTA file containing the intronic sequences to quantify against. This pre-processing step only has to be performed once assuming the transcriptome annotation does not change and read size is at least a few bases longer than exon overlap.

`kma` is designed to leverage existing transcript quantification methods. This allows for the computation of relative abundance of introns as well as transcripts while allowing multi-mapping reads to be processed using well understood models already developed in existing tools [63, 50]. After the pre-processing step, the intronic sequences are added to the transcriptome and the chosen quantification method is run using the augmented transcriptome. Any method can be used, provided it outputs expression in a unit that is additive, e.g. transcripts per million (TPM).

Once introns and transcripts are quantified from all samples in the experiment, the data can be post-processed and further analyzed in an `R` package [71] that is part of `kma`. We currently provide functions to read data from eXpress, but it is quite simple to add a new function that reads in other formats; all that is required is the target identifier and corresponding expression estimate. Once data is read in, retention is computed by taking the intron expression (numerator) and summing the expression of the overlapping transcripts plus the intron expression (denominator) (Figure 4.2). This calculation leads to a natural measurement of intron retention, the proportion of the transcript expression containing the intron, also known as the proportion spliced in [38].

While we store a special object of class IntronRetention, the majority of the operations depend only on the data stored in database-like data frames with each row being an intron observation from one sample. A common row contains categorical fields intron, sample, condition which serve as a key, along with measurements retention, numerator, denominator, unique_reads and various columns for filters. This allows for fast aggregation and manipulation via packages such as dplyr [90]. Summaries of retention across subgroups such as specific introns, conditions, or samples can be quickly computed by simple queries. We provide common summaries as functions, but the raw data frame is always available for further analysis.

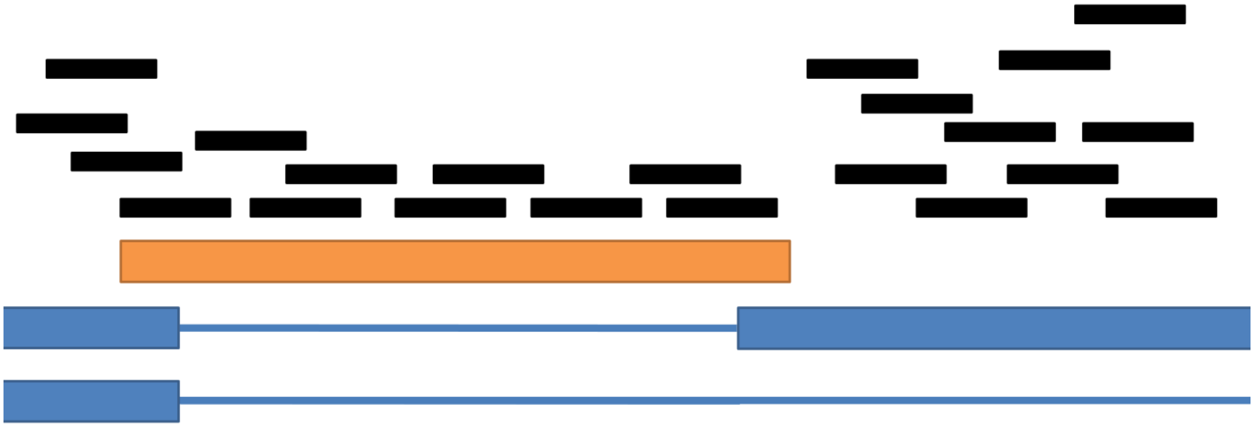


Figure 4.2: Example of a region around an intron with KeepMeAround. The blue rectangles are exons from different transcripts of the gene SRSF6. The orange transcript is a target that is added to the transcriptome and quantified against. The black rectangles are single-end fragments, typical from RNA-Seq data. The estimate of the percent spliced in, $\Psi =$ TPM(orange) / (TPM(orange) + TPM(blue)).

In addition to easy manipulation, this data format is suitable for exploratory analysis in plotting tools such as ggplot2 [89].

## 4.2 Filtering unlikely regions

Occasionally quantification targets contain repeat elements that are not actually expressed at that locus. A clear indicator is that the expression increases immensely for only a short number of bases. A way to identify such a region is to compute the probability of observing long regions of zeros given the abundance estimated of that region. While the algorithm below is used to remove targets with very uneven coverage, the algorithm can also be used to identify potential issues in a transcriptome annotation.

Consider a locus (e.g. gene) with $m$ expression targets (isoforms) $t_1, t_2, \ldots, t_m$. We define

- $\lambda(t_i)$ be the expected number of reads per base (i.e. the rate determined from quantification using a method such as eXpress [72]).
- $X_p(t_i) \sim \text{Poi}\left(\lambda(t_i)\right)$ indicating the number of reads starting at position $p$ on expression target $t_i$.

Thus, we assume uniform coverage according to the Poisson distribution for each expression target. We also assume that the number of reads starting at each position of each expression targets are idependent and identically distributed random variables.

We can identify targets which have coverage patterns that are not consistent with the above model (therefore implying incorrect annotation) by finding long regions with no coverage that exceed in length what is expected based on covered regions. Let $\alpha$ be the level of significance we wish to test at. We assume that $\alpha$ is appropriately scaled by the Bonferonni method. For example, if we choose to test at level 0.01, we set $\alpha = 0.01/(\text{total number of tests})$.

For every $t_i$ that has a zero region of length $z$, we can bound $\lambda(t_i)$. Since

$$\begin{aligned} & \Pr(X_p(t_i) = 0, \ldots, X_{p+z}(t_i) = 0) = (\Pr(X_t = 0))^z \\ \Rightarrow\ & \exp(-z\lambda(t_i)) \leq \alpha \\ \Rightarrow\ & \lambda(t_i) \geq -\log(\alpha)\frac{1}{z}. \end{aligned}$$

Thus, with probability $1 - \alpha$, we expect

$$\lambda(t_i) \leq -\log(\alpha)\frac{1}{z}.$$

A target is labeled as problematic if the previous bound does not hold. To filter out such targets we chose our (uncorrected for multiple testing) $\alpha = 0.01$. If any of problematic targets were part of an overlap set, we removed exons corresponding to that overlap set.

## 4.3 Testing for inclusion

Our method provides a resampling hypothesis testing procedure to determine whether the mean is greater than what one would expect due to reshuffling of the given data in those samples. The null distribution is generated from the filtered list by randomly selecting a retention value from each sample per condition $B$ times. For each set of samples, the mean is computed. After the null distribution is generated, the $p$-value is computed by finding the proportion of null values that the observed mean is greater than. This allows for a lower false-positive rate when detecting IR events. This procedure also helps shield against samples that have contamination of non-mature mRNA.

## 4.4 Application: terminal erythropoiesis

Erythroid differentiation represents an excellent model system for exploring stage-specific post-transcriptional remodeling of gene expression during terminal differentiation. Fluorescence-activated cell sorting (FACS) makes possible isolation of discrete, highly purified populations of cells as they differentiate, enucleate to form reticulocytes and ultimately mature into red cells. Early progenitors known as burst-forming unit-erythroid (BFU-E) and colony-forming unit-erythroid (CFU-E) can be highly purified by this approach, as can proerythroblasts (proE) and several stages of terminally differentiating erythroblasts termed basophilic erythroblasts (basoE), polychromatophilic erythroblasts (polyE) and orthochromatophilic erythroblasts (orthoE). We and others have analyzed RNA-seq libraries prepared from these purified populations of human erythroid cells to gain new insights into the evolving erythroid transcriptome at the level of gene-level expression, alternative splicing, non-coding RNA expression, etc. [2, 68, 65].

Proliferating mammalian erythroblasts exhibit a robust, dynamic alternative splicing program [68, 19, 80] enriched in genes involved in cell cycle, organelle organization, chromatin function and RNA processing [68]. A prominent feature of the erythroblast splicing program is a number of alternative splicing 'switches' that increase PSI (percent spliced in) values predominantly in late erythroblasts at the polyE and orthoE stages, temporally correlated with major cellular remodeling as cells conclude their proliferation phase and prepare for enucleation. Splicing switches can alter protein function in physiologically important ways, e.g. upregulation of exon 16 splicing in protein 4.1R transcripts leads to synthesis of protein isoforms that bind spectrin and actin with high affinity, mechanically strengthening the red cell membrane prior to release into the circulation [17, 24, 32]. In most cases, however, understanding the physiological functions of alternative protein isoforms generated via the erythroblast splicing program remains a challenge for future studies.

Previous studies of the erythroid transcriptome entirely over-looked the IR component of the splicing program. Our new study shows that erythroblasts elaborate an extensive and diverse intron retention program encompassing numerous essential erythroid genes including those encoding splicing factors and proteins involved in iron homeostasis. Differentiation stage-specific changes in IR efficiency largely paralleled switches in splicing of cassette exons described earlier [68], reinforcing and expanding the concept that careful regulation of RNA processing plays a major role in terminal erythroid differentiation as cells mature along the path from proE to orthoE.

## RNA-Seq data and processing

RNA-Seq data obtained from five highly purified human erythroblast populations—proerythroblasts (proE), early basophilic erythroblasts (e-basoE), late basophilic erythroblasts (l-basoE), polychromatophilic erythroblasts (polyE) and orthochromatophilic erythroblasts (orthoE) [33] is available at GSE53635. The data include three biological replicates of each population. For other tissues, we imported wiggle plots, showing RNA-seq coverage along the genome, that were generated from Illumina BodyMap 2.0 data available at `http://www.ensembl.org/info/genome/genebuild/rnaseq_annotation.html`.

RNA-seq reads were mapped using Bowtie v2.1.0 to an augmented transcriptome output by KeepMeAround (`kma` ) as described in Section 4.1. Transcripts and introns were then quantified using eXpress v1.5.1 (19). We identified an unambiguous set of 186,838 quantifiable introns in the RefSeq transcriptome, but only 10,152 unique introns passed filters in every condition. To reduce false positives, we removed introns with fewer than three uniquely mapped reads, denominator values of less than 1 TPM (excluding the intron expression) and introns with zero coverage regions longer than 20% of the intron length (Section 4.2). KMA's hypothesis testing feature was then used on the filtered set of introns to test whether retention levels were higher than expected given the background retention levels in each experimental condition. This test also incorporates biological replicates to further reduce the chance of false positives.

Cluster analysis was performed using k-means clustering on a set of introns that passed the above filters in every sample and every condition.

## Interesting intron retention events

Preliminary inspection of mapping data in the wiggle plot format, displaying RNA-seq read density along the genome, revealed that most introns were efficiently spliced in all erythroblast populations. For example, the $\alpha$ and $\beta$ globin genes exhibited major peaks in read density over the exons and deep troughs in intronic regions due to highly efficient joining of exons and removal of introns during pre-mRNA splicing (Figure 4.4, upper). Many housekeeping genes such as those encoding glycolytic enzymes also exhibited negligible IR (Figure 4.3). In contrast, a number of important erythroid transcripts exhibited substantial IR (Figure 4.4, lower). A very prominent IR event was found in the mitoferrin-1 gene (SLC25A37), which encodes a mitochondrial iron import protein that is critical for iron homeostasis and abundant heme biosynthesis in late erythroblasts. SLC25A37 intron 2, $\sim$2 kb in length, was highly retained in orthoE, while introns 1 and 3 were retained at much lower levels. Another major IR event occurs in the SPTA1 gene, encoding the structural protein $\alpha$-spectrin best known for its essential role in promoting assembly of a mechanically stable erythroid membrane skeleton. Intron 20 (1.8 kb) exhibited substantial retention. We also observed moderate IR in EPOR (encoding the erythropoietin receptor), and in spliceosome-associated RNA binding proteins including UAP56 (encoded by DDX39B) and SAP155 (encoded by SF3B1). The latter is an important RNA splicing factor that is frequently mutated in the RARS (refractory anemia with ringed sideroblasts) subtype of myelodysplasia syndrome (MDS). As reported previously [62], IR also occurs in the CLK1 gene, encoding a tyrosine kinase that phosphorylates splicing factors of the SR protein family.

## Validation

Validation of several intron retention events were performed by the Conboy lab. These validations were performed on CD34+ purified cells purified from cord blood and differentiated into erythroblasts over the course of 16 days as described [33].

### RT-PCR analysis of IR transcripts

RNA was purified from cultured erythroblasts as described previously using RNeasy columns according to the manufacturer's instructions (Qiagen), but with the addition of a DNase step to eliminate potential contamination by genomic DNA. RNA from nuclear and cytoplasmic

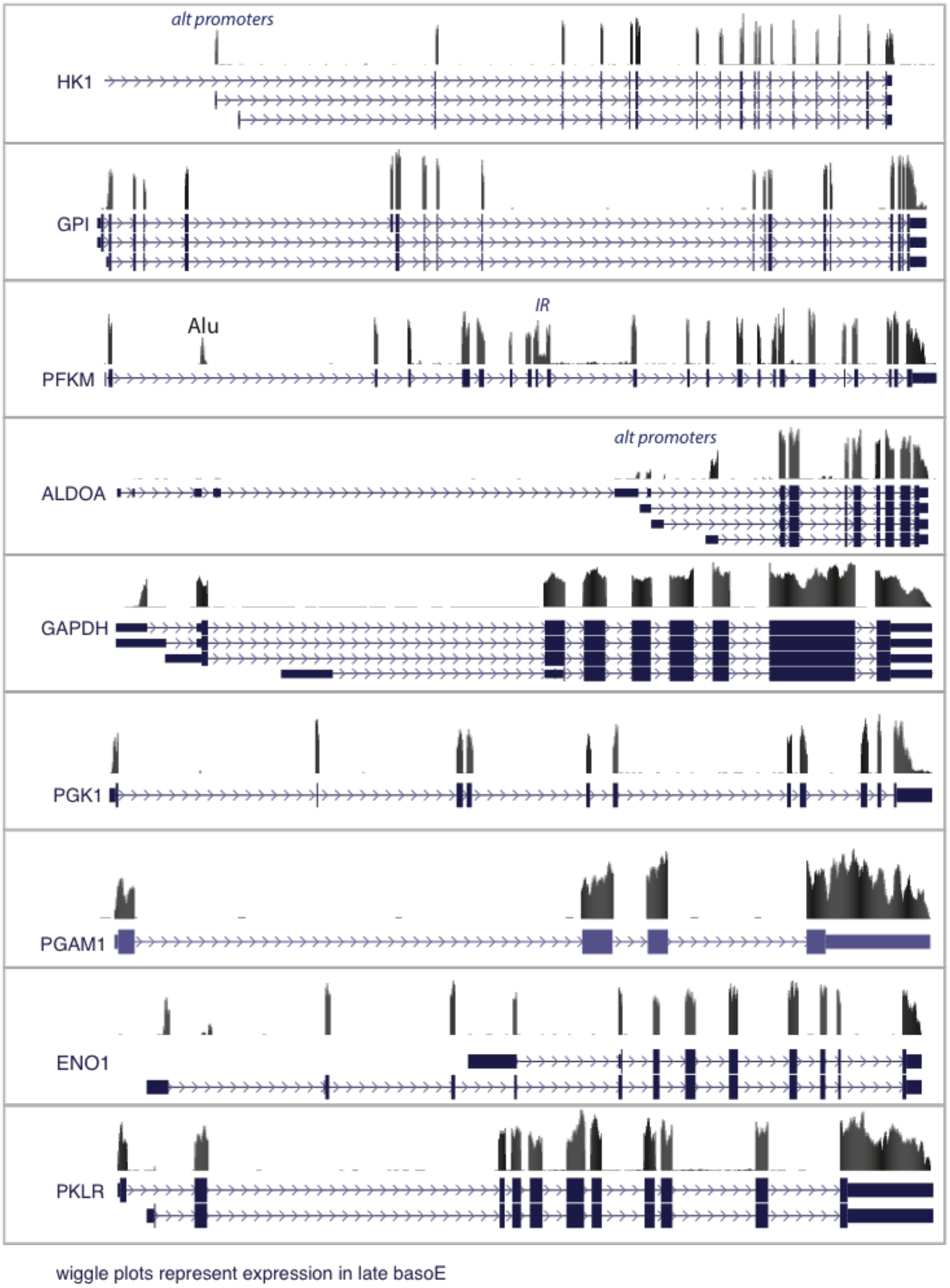


Figure 4.3: Wiggle plots of glycolysis pathway transcripts (housekeeping genes) reveal that IR is uncommon in these genes.

fractions was prepared using Trizol (Life Technologies). To provide additional assurance that intron-containing polymerase chain reaction (PCR) products were not derived from contaminating genomic DNA, we designed PCR assays to span at least one constitutively spliced intron as well as the candidate retained intron. PCR reaction conditions were adjusted to allow for amplification of IR products ≥3 kb in length (denaturation at 95°C for 20", annealing at 60°C for 10", extension at 70°C for 1'15"; 35 cycles) using KOD polymerase in the presence of betaine to enhance amplification. PCR products were analyzed on either 2% agarose gels (for products >1.5 kb) or 4.5% acrylamide gels. All PCR products discussed in the manuscript were confirmed by DNA sequencing.

### Nuclear isolation

Nuclei were prepared from ∼20 million erythroblasts according to published methods [64], with minor modifications. In brief, the erythroblast plasma membrane was lysed using 0.05% NP40, and nuclei were separated from the reddish hemoglobin-rich cytoplasmic fraction by centrifugation through a sucrose cushion at ∼2000 rpm. The whitish nuclear pellet was rinsed with ice-cold phosphate buffered saline containing 1 mM ethylenediaminetetraacetic acid and was resuspended gently to generate a turbid suspension in which nuclei were microscopically verified. Purity of the nuclear fractions was further confirmed by immunoblotting with antibodies to U1–70K protein (a kind gift from D. Black, UCLA).

## Clustering and genome wide analysis

Applying these tools to RNA-seq data from the five erythroblast populations revealed wide variations in percent intron retention (IR), length of retained introns and number of introns retained per transcript. Cluster analysis was performed using $k$-means clustering on a set of introns that passed the above filters in every sample and every condition. Hundreds of introns were retained at IR $>$ 0.10 in at least one erythroblast population. Some of these represented single IR events in otherwise efficiently-spliced transcripts; however, there were also many transcripts that retained multiple introns. The distribution of IR values across the erythroblast populations showed that overall IR increases as erythroblasts differentiate, with highest IR in cells at the orthoE stage (Supplementary Figure S2). These data demonstrate that a robust IR program affects the expression of many important erythroid genes.

We reasoned that dynamic regulation of IR events might be an important gene regulatory mechanism during terminal erythropoiesis, similar to stage-specific exon splicing switches executed in late erythroblasts [68]. Cluster analysis of IR values for each intron at all five maturational stages revealed nine groups of introns (Figure 2). Clusters C1 and C2, comprising ∼470 introns, represent developmentally dynamic events that substantially increase IR in the last two differentiation stages. In contrast, clusters C3–C9 constitute a graded series of developmentally stable intron groups with differentiation-independent IR values, i.e. relatively little change from proE to orthoE. IR is relatively high in C3 but much lower in C9.

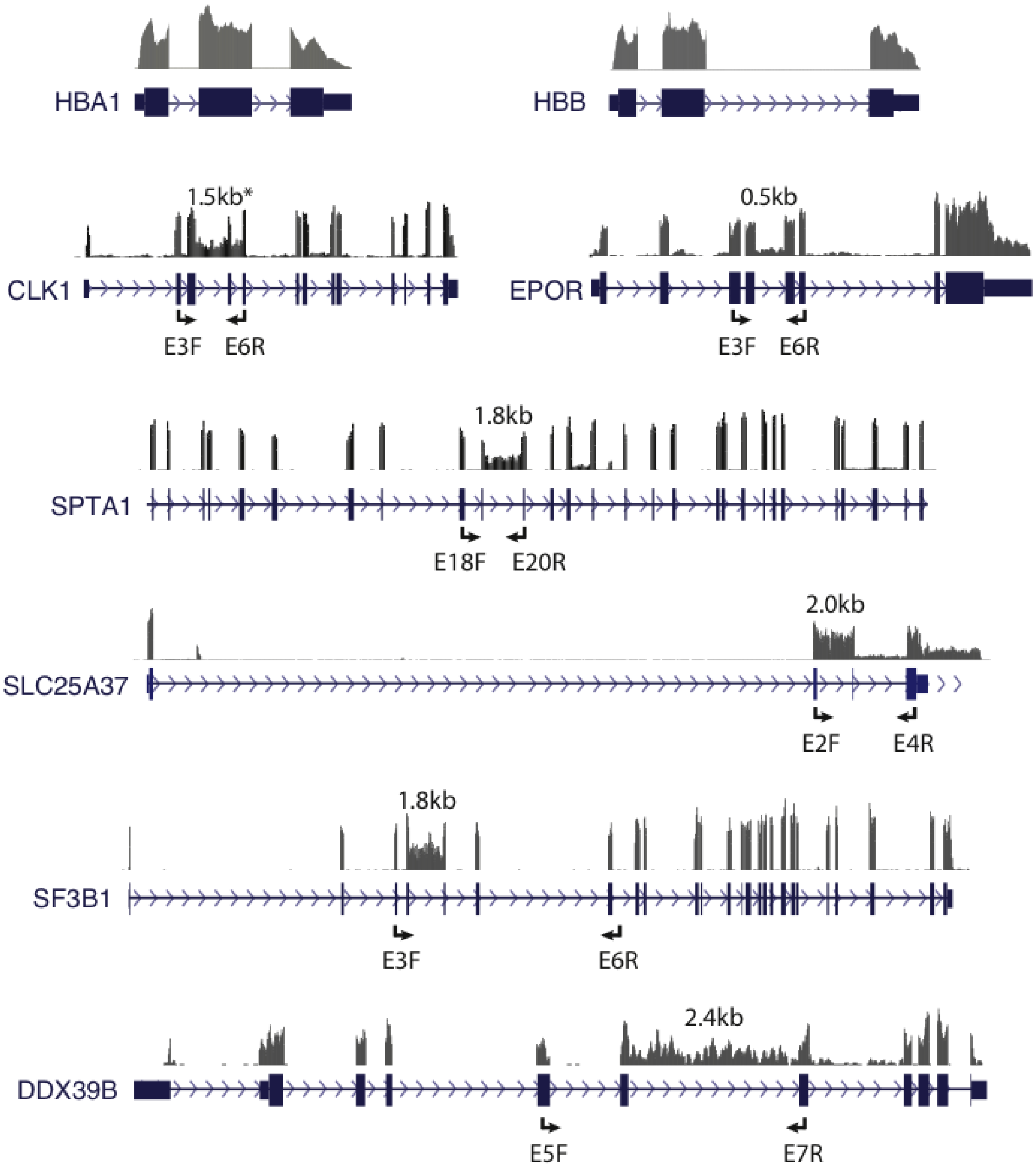


Figure 4.4: Intron retention in important erythroid genes from RNA-Seq data. Wiggle plots showing RNA-seq reads mapped to genes with no IR (top panel, HBA1 and HBB) and genes with significant retention of one or more introns (CLK1, SPTA1, SLC25A37, SF3B1, and DDX39B). Portions of the SPTA1 gene were removed due to size constraints. Size of retained introns is indicated in kilobases and primer locations for PCR validations are shown.

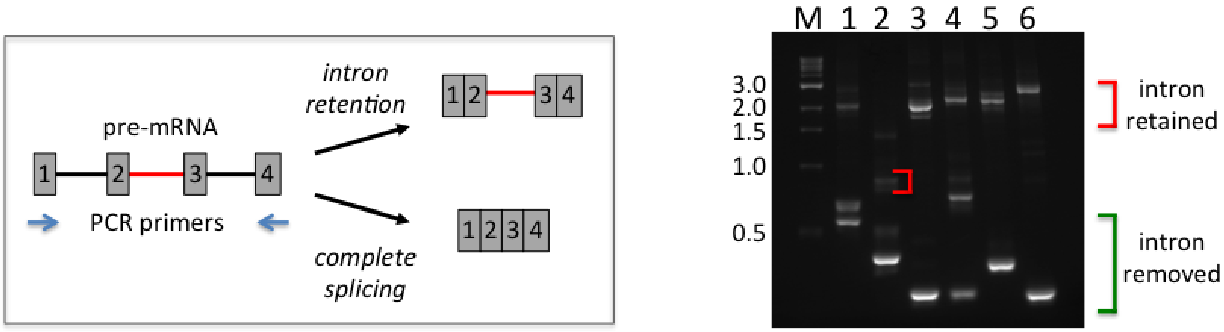


Figure 4.5: RT-PCR confirmation of IR. The general PCR scheme is pictured at the left, while PCR products are shown at the right. Lanes M, size markers; 1, CLK1; 2, EPOR; 3, SPTA1; 4, SLC25A37; 5, SF3B1; 6, DDX39B.

Analysis of intragenic IR patterns revealed that C3, and to a lesser extent in C4, differed qualitatively from the other clusters in that many of the highly retained introns mapped to the first or last intron of a transcript (Supplementary Figure S3). Some of these events might therefore represent alternative initiation or termination of transcription, rather than intron retention per se. However, a few high-level IR events did localize to internal introns (e.g. in SLC25A37).

## 4.5 Differential expression analysis

The model from Chapter 3 can be extended to also test for differential expression. What we describe below is a simple extension to the sleuth model in conjunction with kallisto to implement differential intron retention while incorporating inferential variability.

Firstly, estimate the intron retention percent spliced in, $\Psi$, using the method from KeepMeAround. Using the bootstraps from kallisto, compute bootstrap estimates of $\Psi$. We call this estimate $\hat{\tau}^2_{ti}$, denoting the inferential variability from the $t$th intron from the $i$th sample. Transform everything using the logit transform:

$$\operatorname{logit}(\Psi) = \log\left(\frac{\Psi}{1-\Psi}\right). \tag{4.1}$$

We assume that the unobserved truth follows:

$$\operatorname{logit}(p) \sim \mathrm{N}(\operatorname{logit}(\Psi), \sigma^2) \tag{4.2}$$

where $\sigma^2$ denotes the between sample (biological) variance. Then, we observe a noisy sample, $q$, of this process:

$$q \mid p = \operatorname{logit}(p) + \xi \tag{4.3}$$

$$\xi \sim \mathrm{N}(0, \tau^2). \tag{4.4}$$

This model then appears to be very similar to Equation 3.3. Much like the transcript or gene level model, unconditionally we have an unbiased estimate of the expectation, whereas we unconditionally have two additive components of variance. The estimation procedure can proceed as described in Chapter 3 without any additional modifications.

# Chapter 5

# Conclusion

In the preceding chapters we provided an overview of RNA-Seq and associated analysis problems (Chapter 1), a method for transcript-level abundance estimation called kallisto (Chapter 2), a method for performing differential expression at the transcript and gene level while incorporating inferential variance called sleuth (Chapter 3), and a method for discovering novel retained introns called KeepMeAround (Chapter 4). These methods are general and can be used to answer many biological questions in conjunction with RNA-Seq. The resulting software has also been engineered to empower users to quickly explore their data in a flexible manner, much faster and more easily than existing tools.

In addition, the assumptions of kallisto and sleuth are general and can be applied to other types of sequencing data or other types of RNA-Seq analyses. In fact, we are currently using kallisto with shotgun sequencing data in the metagenomics context [78].

Other areas of extension using RNA-Seq data with kallisto and sleuth include allele specific expression (ASE) and quantitative trait loci (QTL) analyses. In Section 2.4, we showed that ASE estimation is possible with kallisto, but one could go even further and directly estimate haplotypes within kallisto while estimating abundances.

While sleuth has shown great promise by outperforming the current state of the art, there are several obvious areas for extension. Firstly, the assumption of equal variance could be relaxed to support unequal variances between samples. This assumption could be important in QTL analyses where the variance might be different for different genetic mutations. Secondly, other estimators of the variance, perhaps using empirical Bayes, might be fruitful when sample sizes are slightly larger. Thirdly, filtering appears to play a crucial role before model fitting and analysis but is not widely studied.

We showed that by incorporating inferential uncertainty into a differential expression model, sensitivity can be improved. This process is simple and while we are certainly not the first to describe it, we are the first to perform shrinkage in this manner. This shrinkage procedure might also be useful in other areas where inferential variance or other "technical" variance can be quantified well but other variance components should be regularized due to small sample sizes. Additionally, while sleuth assumes the log abundance is normally distributed, it is possible to extend this model to a more general framework, such as the exponential family. This could enable further adaptation of similar procedures in different contexts.

# Bibliography

[1] G. P. Alamancos, A. Pagès, J. L. Trincado, N. Bellora, and E. Eyras. Leveraging transcript quantification for fast computation of alternative splicing profiles. *RNA*, 21(9):1521–1531, 2015.

[2] X. An, V. P. Schulz, J. Li, K. Wu, J. Liu, F. Xue, J. Hu, N. Mohandas, and P. G. Gallagher. Global transcriptome analyses of human and murine terminal erythroid differentiation. *Blood*, 123(22):3466–3477, 2014.

[3] S. Anders and W. Huber. Analysing RNA-Seq data with the DESeq package. `https://www.bioconductor.org/packages/3.3/bioc/vignettes/DESeq/inst/doc/DESeq.pdf`.

[4] S. Anders and W. Huber. Differential expression analysis for sequence count data. *Genome biol*, 11(10):R106, 2010.

[5] S. Anders, P. T. Pyl, and W. Huber. Htseq—a python framework to work with high-throughput sequencing data. *Bioinformatics*, 31(2):166–169, 2015.

[6] S. Anders, A. Reyes, and W. Huber. Detecting differential usage of exons from RNA-seq data. *Genome research*, 22(10):2008–2017, 2012.

[7] Y. Bai, S. Ji, and Y. Wang. IRcall and IRclassifier: two methods for flexible detection of intron retention events from RNA-seq data. *BMC Genomics*, 16(Suppl 2):S9, Jan. 2015.

[8] D. Bottomly, N. A. Walter, J. E. Hunter, P. Darakjian, S. Kawane, K. J. Buck, R. P. Searles, M. Mooney, S. K. McWeeney, and R. Hitzemann. Evaluating gene expression in C57BL/6J and DBA/2J mouse striatum using RNA-Seq and microarrays. *PloS one*, 6(3):e17820, 2011.

[9] P. L. Boutz, A. Bhutkar, and P. A. Sharp. Detained introns are a novel, widespread class of post-transcriptionally spliced introns. *Genes & development*, 29(1):63–80, 2015.

[10] U. Braunschweig, N. L. Barbosa-Morais, Q. Pan, E. N. Nachman, B. Alipanahi, T. Gonatopoulos-Pournatzis, B. Frey, M. Irimia, and B. J. Blencowe. Widespread in-

tron retention in mammals functionally tunes transcriptomes. *Genome Research*, page gr.177790.114, Sept. 2014.

[11] N. L. Bray, H. Pimentel, P. Melsted, and L. Pachter. Near-optimal probabilistic RNA-seq quantification. *Nature biotechnology*, 34(5):525–527, 2016.

[12] J. D. Buenrostro, P. G. Giresi, L. C. Zaba, H. Y. Chang, and W. J. Greenleaf. Transposition of native chromatin for fast and sensitive epigenomic profiling of open chromatin, DNA-binding proteins and nucleosome position. *Nature methods*, 10(12):1213–1218, 2013.

[13] J. P. Buonaccorsi. *Measurement error: models, methods, and applications*. CRC Press, 2010.

[14] D. J. Burgess. Alternative splicing: Retaining introns to sculpt gene expression. *Nature Reviews Genetics*, 15(11):707–707, Nov. 2014.

[15] S. A. Byron, K. R. Van Keuren-Jensen, D. M. Engelthaler, J. D. Carpten, and D. W. Craig. Translating RNA sequencing into clinical diagnostics: opportunities and challenges. *Nature Reviews Genetics*, 2016.

[16] W. Chang, J. Cheng, J. Allaire, Y. Xie, and J. McPherson. Shiny: web application framework for R. *R package version 0.11*, 1, 2015.

[17] J. A. Chasis, L. Coulombel, J. Conboy, S. McGee, K. Andrews, Y. W. Kan, and N. Mohandas. Differentiation-associated switches in protein 4.1 expression. Synthesis of multiple structural isoforms during normal human erythropoiesis. *Journal of Clinical Investigation*, 91(1):329, 1993.

[18] Y. Chen, D. McCarthy, A. Lun, X. Zhou, M. Robinson, and G. K. Smyth. edgeR Package Introduction. `https://www.bioconductor.org/packages/3.2/bioc/vignettes/edgeR/inst/doc/edgeRUsersGuide.pdf`.

[19] A. W. Cheng, J. Shi, P. Wong, K. L. Luo, P. Trepman, E. T. Wang, H. Choi, C. B. Burge, and H. F. Lodish. Muscleblind-like 1 (Mbnl1) regulates pre-mRNA alternative splicing during terminal erythropoiesis. *Blood*, 124(4):598–610, 2014.

[20] P. E. Compeau, P. A. Pevzner, and G. Tesler. How to apply de Bruijn graphs to genome assembly. *Nature biotechnology*, 29(11):987–991, 2011.

[21] A. Conesa, P. Madrigal, S. Tarazona, D. Gomez-Cabrero, A. Cervera, A. McPherson, M. Wojciech Szcześniak, D. Gaffney, L. L. Elo, X. Zhang, et al. A Survey of Best Practices for RNA-seq Data Analysis. 2016.

[22] S.-I. Consortium et al. A comprehensive assessment of RNA-seq accuracy, reproducibility and information content by the Sequencing Quality Control Consortium. *Nature biotechnology*, 32(9):903–914, 2014.

[23] Y. Di, D. W. Schafer, J. S. Cumbie, and J. H. Chang. The NBP negative binomial model for assessing differential gene expression from RNA-Seq. *Statistical Applications in Genetics and Molecular Biology*, 10(1), 2011.

[24] D. Discher, M. Parra, J. G. Conboy, and N. Mohandas. Mechanochemistry of the alternatively spliced spectrin-actin binding domain in membrane skeletal protein 4.1. *Journal of Biological Chemistry*, 268(10):7186–7195, 1993.

[25] J. Eswaran, A. Horvath, S. Godbole, S. D. Reddy, P. Mudvari, K. Ohshiro, D. Cyanam, S. Nair, S. A. Fuqua, K. Polyak, et al. RNA sequencing of cancer reveals novel splicing alterations. *Scientific reports*, 3, 2013.

[26] A. C. Frazee, A. E. Jaffe, B. Langmead, and J. T. Leek. Polyester: simulating RNA-seq datasets with differential transcript expression. *Bioinformatics*, 31(17):2778–2784, 2015.

[27] P.-L. Germain, A. Vitriolo, A. Adamo, P. Laise, V. Das, and G. Testa. RNAontheBENCH: computational and empirical resources for benchmarking RNAseq quantification and differential expression methods. *Nucleic acids research*, page gkw448, 2016.

[28] P. Glaus, A. Honkela, and M. Rattray. Identifying differentially expressed transcripts from RNA-seq data with biological variation. *Bioinformatics*, 28(13):1721–1728, 2012.

[29] T. J. Hardcastle and K. A. Kelly. baySeq: empirical Bayesian methods for identifying differential expression in sequence count data. *BMC bioinformatics*, 11(1):422, 2010.

[30] A. Helwak, G. Kudla, T. Dudnakova, and D. Tollervey. Mapping the human miRNA interactome by CLASH reveals frequent noncanonical binding. *Cell*, 153(3):654–665, 2013.

[31] J. Hensman, P. Papastamoulis, P. Glaus, A. Honkela, and M. Rattray. Fast and accurate approximate inference of transcript expression from RNA-seq data. *Bioinformatics*, 31(24):3881–3889, 2015.

[32] W. Horne, S. Huang, P. Becker, T. Tang, and E. J. Benz. Tissue-specific alternative splicing of protein 4.1 inserts an exon necessary for formation of the ternary complex with erythrocyte spectrin and F-actin. *Blood*, 82(8):2558–2563, 1993.

[33] J. Hu, J. Liu, F. Xue, G. Halverson, M. Reid, A. Guo, L. Chen, A. Raza, N. Galili, J. Jaffray, J. Lane, J. A. Chasis, N. Taylor, N. Mohandas, and X. An. Isolation and functional characterization of human erythroblasts at distinct stages: implications for understanding of normal and disordered erythropoiesis in vivo. *Blood*, 121(16):3246–3253, 2013.

[34] N. T. Ingolia, S. Ghaemmaghami, J. R. S. Newman, and J. S. Weissman. Genome-Wide Analysis in Vivo of Translation with Nucleotide Resolution Using Ribosome Profiling. *Science*, 324(5924):218–223, 2009.

[35] Z. Iqbal, M. Caccamo, I. Turner, P. Flicek, and G. McVean. De novo assembly and genotyping of variants using colored de Bruijn graphs. *Nature genetics*, 44(2):226–232, 2012.

[36] H. Jiang and W. H. Wong. Statistical inferences for isoform expression in RNA-Seq. *Bioinformatics*, 25(8):1026–1032, 2009.

[37] D. S. Johnson, A. Mortazavi, R. M. Myers, and B. Wold. Genome-Wide Mapping of in Vivo Protein-DNA Interactions. *Science*, 316(5830):1497–1502, 2007.

[38] Y. Katz, E. T. Wang, E. M. Airoldi, and C. B. Burge. Analysis and design of RNA sequencing experiments for identifying isoform regulation. *Nature methods*, 7(12):1009–1015, 2010.

[39] Y. L. Khodor, J. Rodriguez, K. C. Abruzzi, C.-H. A. Tang, M. T. Marr, and M. Rosbash. Nascent-seq indicates widespread cotranscriptional pre-mRNA splicing in Drosophila. *Genes & Development*, 25(23):2502–2512, 2011.

[40] D. Kim, B. Langmead, and S. L. Salzberg. HISAT: a fast spliced aligner with low memory requirements. *Nature methods*, 12(4):357–360, 2015.

[41] D. Kim, G. Pertea, C. Trapnell, H. Pimentel, R. Kelley, S. L. Salzberg, et al. TopHat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions. *Genome Biol*, 14(4):R36, 2013.

[42] J. Köster and S. Rahmann. Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics*, 28(19):2520–2522, 2012.

[43] B. Langmead, C. Trapnell, M. Pop, S. L. Salzberg, et al. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol*, 10(3):R25, 2009.

[44] T. Lappalainen, M. Sammeth, M. R. Friedländer, P. AC‘t Hoen, J. Monlong, M. A. Rivas, M. Gonzàlez-Porta, N. Kurbatova, T. Griebel, P. G. Ferreira, et al. Transcriptome and genome sequencing uncovers functional variation in humans. *Nature*, 501(7468):506–511, 2013.

[45] C. W. Law, Y. Chen, W. Shi, and G. K. Smyth. Voom: precision weights unlock linear model analysis tools for RNA-seq read counts. *Genome Biol*, 15(2):R29, 2014.

[46] S. Lee, C. H. Seo, B. H. Alver, S. Lee, and P. J. Park. Emsar: estimation of transcript abundance from rna-seq data by mappability-based segmentation and reclustering. *BMC Bioinformatics*, 16(1):1–16, 2015.

[47] N. Leng, J. Dawson, and C. Kendziorski. EBSeq: An R package for differential expression analysis using RNA-seq data. `https://www.bioconductor.org/packages/3.2/bioc/vignettes/EBSeq/inst/doc/EBSeq_Vignette.pdf`.

[48] N. Leng, J. A. Dawson, J. A. Thomson, V. Ruotti, A. I. Rissman, B. M. Smits, J. D. Haag, M. N. Gould, R. M. Stewart, and C. Kendziorski. EBSeq: an empirical Bayes hierarchical model for inference in RNA-seq experiments. *Bioinformatics*, page btt087, 2013.

[49] B. Li and C. Dewey. RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome. *BMC bioinformatics*, 12(1):323, 2011.

[50] B. Li, V. Ruotti, R. M. Stewart, J. A. Thomson, and C. N. Dewey. RNA-Seq gene expression estimation with read mapping uncertainty. *Bioinformatics*, 26(4):493–500, 2010.

[51] J. Li and R. Tibshirani. Finding consistent patterns: a nonparametric approach for identifying differential expression in RNA-Seq data. *Statistical methods in medical research*, 22(5):519–536, 2013.

[52] J. Li, D. M. Witten, I. M. Johnstone, and R. Tibshirani. Normalization, testing, and false discovery rate estimation for RNA-sequencing data. *Biostatistics*, page kxr031, 2011.

[53] Y. Liao, G. K. Smyth, and W. Shi. featureCounts: an efficient general purpose program for assigning sequence reads to genomic features. *Bioinformatics*, page btt656, 2013.

[54] E. Lieberman-Aiden, N. L. van Berkum, L. Williams, M. Imakaev, T. Ragoczy, A. Telling, I. Amit, B. R. Lajoie, P. J. Sabo, M. O. Dorschner, R. Sandstrom, B. Bernstein, M. A. Bender, M. Groudine, A. Gnirke, J. Stamatoyannopoulos, L. A. Mirny, E. S. Lander, and J. Dekker. Comprehensive Mapping of Long-Range Interactions Reveals Folding Principles of the Human Genome. *Science*, 326(5950):289–293, 2009.

[55] M. I. Love, S. Anders, and W. Huber. Differential analysis of count data - the DESeq2 package. `https://www.bioconductor.org/packages/3.2/bioc/vignettes/DESeq2/inst/doc/DESeq2.pdf`.

[56] M. I. Love, W. Huber, and S. Anders. Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome biology*, 15(12):1–21, 2014.

[57] J. B. Lucks, S. A. Mortimer, C. Trapnell, S. Luo, S. Aviran, G. P. Schroth, L. Pachter, J. A. Doudna, and A. P. Arkin. Multiplexed RNA structure characterization with selective 2'-hydroxyl acylation analyzed by primer extension sequencing (SHAPE-Seq). *Proceedings of the National Academy of Sciences*, 108(27):11063–11068, 2011.

[58] V. Madan, D. Kanojia, J. Li, R. Okamoto, A. Sato-Otsubo, A. Kohlmann, M. Sanada, V. Grossmann, J. Sundaresan, Y. Shiraishi, et al. Aberrant splicing of U12-type introns is the hallmark of ZRSR2 mutant myelodysplastic syndrome. *Nature communications*, 6, 2015.

[59] J. C. Marioni, C. E. Mason, S. M. Mane, M. Stephens, and Y. Gilad. RNA-seq: An assessment of technical reproducibility and comparison with gene expression arrays. *Genome Research*, 18(9):1509–1517, 2008.

[60] A. Mortazavi, B. A. Williams, K. McCue, L. Schaeffer, and B. Wold. Mapping and quantifying mammalian transcriptomes by RNA-Seq. *Nature methods*, 5(7):621–628, 2008.

[61] M. Nicolae, S. Mangul, I. I. Măndoiu, and A. Zelikovsky. Estimation of alternative splicing isoform frequencies from RNA-Seq data. *Algorithms for Molecular Biology*, 6(1):1, 2011.

[62] K. Ninomiya, N. Kataoka, and M. Hagiwara. Stress-responsive maturation of Clk1/4 pre-mRNAs promotes phosphorylation of SR splicing factor. *The Journal of Cell Biology*, 195(1):27–40, 2011.

[63] L. Pachter. Models for transcript quantification from RNA-Seq. *arXiv*, 1104.3889, 2011.

[64] A. Pandya-Jones and D. L. Black. Co-transcriptional splicing of constitutive and alternative exons. *RNA*, 15(10):1896–1908, 2009.

[65] V. R. Paralkar, T. Mishra, J. Luan, Y. Yao, A. V. Kossenkov, S. M. Anderson, M. Dunagin, M. Pimkin, M. Gore, D. Sun, N. Konuthula, A. Raj, X. An, N. Mohandas, D. M. Bodine, R. C. Hardison, and M. J. Weiss. Lineage and species-specific long noncoding RNAs during erythro-megakaryocytic development. *Blood*, 123(12):1927–1937, 2014.

[66] R. Patro, S. M. Mount, and C. Kingsford. Sailfish enables alignment-free isoform quantification from RNA-seq reads using lightweight algorithms. *Nat Biotech*, 32(5):462–464, 05 2014.

[67] H. Pimentel, J. G. Conboy, and L. Pachter. Keep Me Around: Intron Retention Detection and Analysis. *ArXiv e-prints*, Oct. 2015.

[68] H. Pimentel, M. Parra, S. Gee, D. Ghanem, X. An, J. Li, N. Mohandas, L. Pachter, and J. G. Conboy. A dynamic alternative splicing program regulates gene expression during terminal erythropoiesis. *Nucleic Acids Research*, 42(6):4031–4042, 2014.

[69] H. Pimentel, M. Parra, S. L. Gee, N. Mohandas, L. Pachter, and J. G. Conboy. A dynamic intron retention program enriched in RNA processing genes regulates gene expression during terminal erythropoiesis. *Nucleic Acids Research*, 44(2):838–851, 2016.

[70] H. J. Pimentel, N. Bray, S. Puente, P. Melsted, and L. Pachter. Differential analysis of RNA-Seq incorporating quantification uncertainty. *bioRxiv*, 2016.

[71] R Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2014.

[72] A. Roberts and L. Pachter. Streaming fragment assignment for real-time analysis of sequencing experiments. *Nature methods*, 10(1):71–73, 2013.

[73] A. Roberts, C. Trapnell, J. Donaghey, J. L. Rinn, L. Pachter, et al. Improving RNA-Seq expression estimates by correcting for fragment bias. *Genome Biol*, 12(3):R22, 2011.

[74] M. D. Robinson, A. Oshlack, et al. A scaling normalization method for differential expression analysis of RNA-seq data. *Genome Biol*, 11(3):R25, 2010.

[75] M. D. Robinson and G. K. Smyth. Moderated statistical tests for assessing differences in tag abundance. *Bioinformatics*, 23(21):2881–2887, 2007.

[76] M. D. Robinson and G. K. Smyth. Small-sample estimation of negative binomial dispersion, with applications to SAGE data. *Biostatistics*, 9(2):321–332, 2008.

[77] D. M. Rocke, L. Ruan, J. J. Gossett, B. Durbin-Johnson, and S. Aviran. Controlling False Positive Rates in Methods for Differential Gene Expression Analysis using RNA-Seq Data. *bioRxiv*, page 018739, 2015.

[78] L. Schaeffer, H. Pimentel, N. Bray, P. Melsted, and L. Pachter. Pseudoalignment for metagenomic read assignment. *arXiv preprint arXiv:1510.07371*, 2015.

[79] F. Seyednasrollah, A. Laiho, and L. L. Elo. Comparison of software packages for detecting differential expression in RNA-seq studies. *Briefings in bioinformatics*, 16(1):59–70, 2015.

[80] L. Shi, Y.-H. Lin, M. Sierant, F. Zhu, S. Cui, Y. Guan, M. A. Sartor, O. Tanabe, K.-C. Lim, and J. D. Engel. Developmental transcriptome analysis of human erythropoiesis. *Human Molecular Genetics*, 23(17):4528–4542, 2014.

[81] G. K. Smyth, M. Ritchie, N. Thorne, J. Wettenhall, W. Shi, and Y. Hu. limma: Linear Models for Microarray and RNA-Seq Data User's Guide. `https://www.bioconductor.org/packages/3.2/bioc/vignettes/limma/inst/doc/usersguide.pdf`.

[82] C. Soneson, M. I. Love, and M. D. Robinson. Differential analyses for RNA-seq: transcript-level estimates improve gene-level inferences. *F1000Research*, 4, 2015.

[83] M. Teng, M. I. Love, C. A. Davis, S. Djebali, A. Dobin, B. R. Graveley, S. Li, C. E. Mason, S. Olson, D. Pervouchine, et al. A benchmark for RNA-seq quantification pipelines. *Genome biology*, 17(1):1, 2016.

[84] C. Trapnell. Cuffdiff 2 Manual. `http://cole-trapnell-lab.github.io/cufflinks/cuffdiff/index.html#differential-expression-tests`.

[85] C. Trapnell, D. G. Hendrickson, M. Sauvageau, L. Goff, J. L. Rinn, and L. Pachter. Differential analysis of gene regulation at transcript resolution with RNA-seq. *Nature biotechnology*, 31(1):46–53, 2013.

[86] C. Trapnell, L. Pachter, and S. L. Salzberg. TopHat: discovering splice junctions with RNA-Seq. *Bioinformatics*, 25(9):1105–1111, 2009.

[87] C. Trapnell, B. A. Williams, G. Pertea, A. Mortazavi, G. Kwan, M. J. van Baren, S. L. Salzberg, B. J. Wold, and L. Pachter. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nature biotechnology*, 28(5):511–515, 2010.

[88] Z. Wang, M. Gerstein, and M. Snyder. RNA-Seq: a revolutionary tool for transcriptomics. *Nature Reviews Genetics*, 10(1):57–63, 2009.

[89] H. Wickham. *ggplot2: elegant graphics for data analysis*. Springer New York, 2009.

[90] H. Wickham and R. Francois. *dplyr: A Grammar of Data Manipulation*, 2014. R package version 0.3.0.2.

[91] B. Wold and R. M. Myers. Sequence census methods for functional genomics. *Nature methods*, 5(1):19–21, 2008.

[92] B. Wold and R. M. Myers. Sequence census methods for functional genomics. *Nature methods*, 5(1):19–21, 2008.

[93] J. J.-L. Wong, W. Ritchie, O. A. Ebner, M. Selbach, J. A. Wong, Y. Huang, D. Gao, N. Pinello, M. Gonzalez, K. Baidya, A. Thoeng, T.-L. Khoo, C. G. Bailey, J. Holst, and J. E. Rasko. Orchestrated intron retention regulates normal granulocyte differentiation. *Cell*, 154(3):583 – 595, 2013.

[94] Q. Zheng, P. Ryvkin, F. Li, I. Dragomir, O. Valladares, J. Yang, K. Cao, L.-S. Wang, and B. D. Gregory. Genome-Wide Double-Stranded RNA Sequencing Reveals the Functional Significance of Base-Paired RNAs in Arabidopsis. *PLoS Genet*, 6(9):1–14, 09 2010.