Supplementary Methods: Effects of transcriptional noise on estimates of gene and transcript expression in RNA sequencing experiments

Ales Varabyou [1,3*], Steven L. Salzberg [1,2,3,4], and Mihaela Pertea [1,2,3*]

1 Center for Computational Biology, Johns Hopkins University, Baltimore, MD

2 Department of Biomedical Engineering, Johns Hopkins University, Baltimore, MD

3 Department of Computer Science, Johns Hopkins University, Baltimore, MD

4 Department of Biostatistics, Johns Hopkins University, Baltimore, MD

*Corresponding author. Email: ales.varabyou@jhu.edu, mpertea@jhu.edu

Methods

Reference data

To avoid unnecessary complexity, we restricted our analysis to the primary chromosomes of the GRCh38.12 human assembly (Graves-Lindsay et al. 2017), excluding all alternative scaffolds and patches. As reference annotation we used the version of CHESS 2.2 human gene catalog tailored for transcriptome assembly, which does not include pseudogenes, V|J|D|C segments, snoRNAs, snRNAs, telomerase RNAs, guide RNAs, or other small RNAs.

Command: build_indices.py --idx_dir <output_directory> --reference <reference_genome.fasta> --annotation <reference_annotation.gff> --hisat <path_to_hisat2_executable> --salmon <path_to_salmon_executable> --kallisto <path_to_kallisto_executable> --gffread <path_to_gffread_executable>

Identification and typing of noisy transcription

To identify other types of transcription observed across the GTEx dataset (Lonsdale et al. 2013), we relied on the set of ~30 million transcripts assembled as part of building the CHESS gene database (Pertea et al. 2018). These 30 million transcripts were the unfiltered results of assembling the GTEx data.

Filtering and typing

We first used gffcompare (Pertea and Pertea 2020) to identify overlaps between all ~30 million assembled transcripts and known features in the CHESS 2.2 annotation. We then used gffread (Pertea and Pertea 2020) to extract the loci of all intron-compatible transcripts that overlapped any of the annotated genes. Next, to reduce the number of confounding factors in our analysis, we removed any noisy isoforms that overlapped annotated loci on the opposite strand, contained annotated loci within their introns, or were in close proximity to known genes but did not overlap their exons. Together, our filtering criteria reduced the initial set of transcripts from ~30 million down to ~20 million assembled isoforms.

Transcripts that did not represent annotated isoforms in CHESS were further classified into:

1. Intronic noise: any assembled feature entirely contained within an intronic region of a gene, including both coding and noncoding genes.
2. Splicing noise: transcripts that shared a splice site with any of the annotated transcripts in CHESS. These could either be un-annotated but nonetheless functional transcripts, or transcripts that occur due to errors produced by the splicing machinery, incompletely processed pre-mRNA, or other biological phenomena not accounted for by the annotation.
3. Intergenic noise: the set of all transcripts that do not overlap any of the annotated transcripts.

Multiple transcripts in the assembled dataset did not have a strand assigned to them. For consistency in the downstream analysis, we randomly assigned a strand to all intergenic transcripts, making sure that all transcripts in the same locus share the same strand. For all other transcripts, we assigned the strand of the annotated gene containing or overlapping the transcript.

Command: setup.py --base_dir_data <directory_with_input_data> --base_dir_out <output_directory> --gtex_stats <gtex_stats> --filters <filters.sh> --seed 101

Extracting parameters of the simulation from GTEx

Expression estimates and frequency of occurrence of each transcript and locus in the GTEx tissues and samples were computed based on the tracking information which gffcompare builds between merged transcripts and constituent assemblies. These summaries were computed for each type of real and noise transcripts and loci.

Command: gtex_stats –t <path_to_tissues_list> -a ALL.merged.filtered.tracking -r real.gtf -g ALL.merged.filtered.gtf -s splicing.gtf -i intronic.gtf -p RNApol.gtf -o <output_basename>

Simulation

Selecting the loci to be expressed in a simulated tissue

We assumed a Gaussian distribution of the number of real and intergenic loci per each tissue, where we determined the mean and the standard deviation based on the observations extracted from the GTEx analysis described in section 2.2. To generate the number of loci expressed in a tissue, we sampled from these two distributions independently. Furthermore, we assumed that if a locus is

chosen to be expressed in a tissue, all transcripts at that locus can also be expressed in the given tissue.

Command: step1_sim_tissues.py --out_dir <output_directory> --base_dir_data <input_directory_from_setup.py> --base_dir_out <output_directory_from_setup.py> --num_tissues 3 --seed 101

Selecting expression levels for the transcripts in a simulated tissue

During our analysis of the GTEx dataset, we noticed that the relationship between expression of real and noisy transcripts in a given locus is non-linear and strongly correlated, and annotated transcripts tend to dominate the expression of a locus (Fig. 1D). We also observed that the expression values of a transcript within a tissue are more correlated than the expression values of a transcript between different tissues. To preserve these and any other possible relationships existing in the data, we sampled the number of transcripts per each locus as well as corresponding expression values as they appeared in GTEx. To randomize our sampling algorithm without changing any potential relationships we relied on the following procedure:

1. For each locus selected for a simulated tissue, we identified all loci with identical numbers of real, splicing noise and intronic noise isoforms.
2. We then randomly selected one of the matching loci and obtained a list of expression values as computed in section 2.2. Each such list contained expression values of each transcript from the selected locus observed in a single tissue, with absent transcripts marked by a zero. Values within the list were grouped by sample and each group was sorted by unique transcript identifier, thus guaranteeing identical ordering between groups. Each group in the list represented a possible set of expressions for a single simulated sample.
3. Keeping the groups unchanged, we shuffled a list of transcript identifiers for the selected locus to further randomize associations between isoforms and expression values, without changing any relationships in expression between values. This step produced a matrix in which rows denote individual transcripts of a locus, columns denote possible samples and values contain

expression values of transcripts in a sample, with zeros indicating that a transcript is not expressed in a sample.

The current approach is limited by using observed values only. A possible addition to the method would be to replace observed values with a multivariate distribution of expression values of each type, which can be estimated from the observed values. This would allow creating more random samples. However, further work needs to be done to determine the types of distributions suitable for modeling expression values.

Command: step2_sim_attach_exp.py --out_dir <output_directory_from_sim_tissues.py> --base_dir_data <input_directory_from_setup.py> --base_dir_out <output_directory_from_setup.py> --num_tissues 3 --seed 101

Selecting transcripts for the simulated samples

Given a set of loci to be expressed in a simulated tissue and a set of possible expression values for each real and noisy isoform at each locus, we randomly drew individual samples. Not all tissue loci are typically expressed in a single sample of that tissue. To simulate this process, we took a random subset of the tissue loci based on a random sampling from the distribution of the number of loci per sample as observed in GTEx.

Most of the tissues in GTEx contain many more samples than are simulated in our analysis and not all loci that are detected in a tissue are expressed in all samples of that tissue, which may be caused by multiple intrinsic and extrinsic factors contributing to the stochasticity of gene expression in living organisms (Swain et al. 2002). As such, simulating a smaller number of samples per tissue often results, as evident from our simulation, in a lower total number of loci within simulated tissues than observed in GTEx.

Next, for each locus we selected a random column from the simulated matrix of possible expression values as described in step 3 from section 3.2. The number of transcripts expressed for a given locus in a sample is then dictated by the number of non-zero values in the sampled column.

At the end of this step the simulation produces 4 groups of files for each type of transcription being simulated. Each group includes:

1. an annotation in a GTF format, listing transcript and exon features only.
2. a list of expression values in TPM format.

Command: step3_sim_samples.py --out_dir <output_directory_from_sim_tissues.py> --base_dir_data <input_directory_from_setup.py> --base_dir_out <output_directory_from_setup.py> --num_tissues 3 –num_samples 10 --seed 101

Normalization of transcript abundances in the simulated samples

In the next step we randomly sample the number of reads to be simulated for a given sample from the observations in GTEx. To compute the target number of reads to be simulated for each transcript in a sample given the TPM of each molecule and the total number of reads, we reversed the TPM calculation as follows:

$$C_i = \left(\frac{E_i \times L_i}{\sum_{j=0}^{N} (E_j \times L_j)} \times N \times l \right) \div L_i$$

where C_i is the coverage of transcript i, E_i is its expression measured in TPM, L_i is length, and N is the number of reads in the sample and l is the read length (Li and Dewey 2011).

Command: step4_sim_normalize.py --out_dir <output_directory_from_sim_tissues.py> --base_dir_data <input_directory_from_setup.py> --base_dir_out <output_directory_from_setup.py> --num_tissues 3 –num_samples 10 --seed 101 --readlen 101

Visualization of summarized statistics for the GTEx and simulated datasets

All scripts for re-producing visualizations from our analysis of GTEx and simulated datasets are provided in a public Python **notebook** at**:** evaluation_stage1_results.ipynb

Simulating reads

We used the simulated transcripts selected for each sample and their corresponding coverages to simulate reads for each experiment. Reads were simulated using the Polyester package (Frazee et al. 2015), and the input FASTA file was prepared using gffread.

In our preliminary analysis we found that Polyester is unable to accurately model the sequencing protocol when simulating paired-end reads. Specifically, for transcripts shorter than the fragment length, Polyester left gaps in coverage unless the fragment length was reduced, or the standard deviation was set high enough to cause frequent overlaps between pairs. Setting fragment length and standard deviation according to the minimum transcript length instead of realistic observations from sequencing data would inaccurately represent sequencing protocols. Therefore, to avoid introducing artificial coverage gaps we chose to simulate single-end reads instead.

We also found that when simulating single-end reads Polyester was often unable to extend read coverage to the end of the last exon in the transcript. We assume that internally Polyester always simulates paired-end reads and only reports the first mate in the single-end mode. We found that setting the fragment length to be the same as the read length and standard deviation to zero fixed the issue.

Each sample was simulated with 101-bp reads, as was done in the majority of the GTEx samples. Error rate was set at 0.4 percent.

Next, we shuffled reads within each simulated experiment using a custom script. A non-random order could present issues for many alignment and quantification methods, which perform a quick scan through a subset of the reads for calibration.

Lastly, we combined reads generated from real transcripts, splicing noise, intronic noise and intergenic noise transcripts together for each sample.

Notebook: evaluation_stage2_genReads.ipynb

Ground truth alignment

While not used in our final analysis, we found it useful to have a ground-truth alignment of the simulated reads, since any alignment algorithm inevitably introduces errors due to sequencing errors, splice-site detection, low-complexity or homologous regions. We designed a separate method to convert standard Polyester and RSEM output into SAM formatted alignments.

Notebook: evaluation_stage3_process.ipynb

Alignment of simulated reads and assembly-based quantification

Reads were aligned with HISAT2 version 2.2 (Kim et al. 2019) against the indices constructed for this analysis. We used a copy of HISAT2 from the rna_sensitive branch which introduces the "--rna-sensitive" flag. This flag guarantees that for all multimapping reads at least one mapping will be reported, unlike the default behavior of HISAT2, which skips reporting mappings for reads which align to more than "-k" locations on the genome.

Alignments were sorted with samtools (Li et al. 2009) and assembled with StringTie2 (Kovaka et al. 2019) providing chess2.2_assembly.gff as guides.

Notebook: evaluation_stage4_align_assemble.ipynb

Abundance quantification via pseudo-mapping

Abundance of transcripts in each sample was estimated using Salmon version 0.14.0 (Patro et al. 2017) and Kallisto version 0.46.1 (Bray et al. 2016). Salmon was run with "--validateMappings" flag in compliance with recommended parameters. For Kallisto, we set the read length to 101 using "-l" option and the standard deviation was set to 0.0001 with the "-s" option.

Notebook: evaluation_stage5_quantify.ipynb

Processing of transcript-level results

We counted the number of simulated reads for each transcript in a sample using the metadata provided by Polyester. Read counts were used to compute the TPM of all real and noisy transcripts in each dataset. It must be noted that, since simulated transcripts came from independently assembled

data, the positions of the 3' and 5' ends may have been different from positions reported in the annotation provided to all methods. In our analysis expression was measured based on the true structure of the simulated molecules, to provide a more realistic simulated dataset.

Additionally, TPM values were computed separately for each of the combinations of noisy and real transcripts, since fluctuations in the number of molecules of different effective lengths changes the TPM values of all transcripts. However, in our analysis we refer to simulated TPM values only in the context of abundance of false negatives reported by each method, while keeping other comparisons strictly based on the values reported by each method.

Lastly, for StringTie2 we computed the raw number of reads assigned to each transcript based on the reported coverage with respect to the effective length of the assembled molecule.

Notebook: evaluation_stage6_get_results.ipynb

Processing of gene-level results

To avoid differences in normalization and to measure raw effects of transcriptional noise, we computed gene-level abundances using only raw read counts from real transcripts for each gene in each sample. True read counts for each transcript in each sample were computed from the simulated data and labeled by whether the transcript was simulated as part of a real or noisy transcription. To obtain expected read counts per gene, transcript-level read counts were summed considering real transcripts only. For each gene, we computed the number of reads which came from noisy transcripts.

Notebook: evaluation_stage7_get_gene_expression.ipynb

Visualization of results

All scripts for re-producing transcript-level visualizations from our analysis are provided in a Python **notebook** at: evaluation_stage8_results_tx.ipynb

All scripts for re-producing gene-level visualizations from our analysis are provided in a Python **notebook** at: evaluation_stage9_results_gene.ipynb

References

Bray NL, Pimentel H, Melsted P, Pachter L. 2016. Near-optimal probabilistic RNA-seq quantification. *Nature Biotechnology* **34**: 525–527.

Frazee AC, Jaffe AE, Langmead B, Leek JT. 2015. Polyester: simulating RNA-seq datasets with differential transcript expression. *Bioinformatics* **31**: 2778–2784.

Graves-Lindsay T, Albracht D, Fulton RS, Kremitzki M, Magrini V, Meltz K, al et. 2017. Evaluation of GRCh38 and de novo haploid genome assemblies demonstrates the enduring quality of the reference assembly. *Genome Research* **27**: 849–864.

Kim D, Paggi JM, Park C, Bennett C, Salzberg SL. 2019. Graph-based genome alignment and genotyping with HISAT2 and HISAT-genotype. *Nature Biotechnology* **37**: 907–915.

Kovaka S, Zimin A v., Pertea GM, Razaghi R, Salzberg SL, Pertea M. 2019. Transcriptome assembly from long-read RNA-seq alignments with StringTie2. *Genome Biology* **20**.

Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R. 2009. The sequence alignment/map format and SAMtools. *Bioinformatics* **25**: 2078–2079.

Lonsdale J, Thomas J, Salvatore M, Phillips R, Lo E, Shad S, Hasz R, Walters G, Garcia F, Young N, et al. 2013. The Genotype-Tissue Expression (GTEx) project. *Nature Genetics* **45**: 580–585.

Patro R, Duggal G, Love MI, Irizarry RA, Kingsford C. 2017. Salmon provides fast and bias-aware quantification of transcript expression. *Nature Methods* **14**: 417–419.

Pertea G, Pertea M. 2020. GFF Utilities: GffRead and GffCompare. *F1000Research* **9**.

Pertea M, Shumate A, Pertea G, Varabyou A, Breitwieser FP, Chang YC, Madugundu AK, Pandey A, Salzberg SL. 2018. CHESS: A new human gene catalog curated from thousands of large-scale RNA sequencing experiments reveals extensive transcriptional noise. *Genome Biology* **19**.

Swain PS, Elowitz MB, Siggia ED. 2002. Intrinsic and extrinsic contributions to stochasticity in gene expression. *Proceedings of the National Academy of Sciences* **99**: 12795–12800.