Variance All the Way Down: Quantifying the Uncertainty Introduced at each Stage of an End-to-End RNA-Seq Analysis

Hunter Schuler and Art Tay

Abstract

In the realm of RNA-Seq research, rigorous data preprocessing is a critical foundation for meaningful analysis. Despite its importance, this preprocessing involves numerous stages, each introducing potential sources of variance. While previous studies have examined the overall variance between entire RNA-Seq pipelines, (Arora et al. 2020) (Tong et al. 2020), (Vieth et al. 2019), the impact of individual stages remains less understood. We propose a comprehensive investigation into the variance introduced at each stage of RNA-Seq preprocessing. Our goal is to quantify these variances, study their distributions, and understand their statistical implications on downstream modeling. This will include exploring the multitude of decisions researchers face — from quality control to normalization — and evaluating how these choices propagate uncertainty through the analysis. Of particular interest is whether variance amplifies due to interactions between decisions made at different stages. By modeling these interactions, we aim to identify cases where suboptimal combinations of preprocessing choices exacerbate variability, potentially distorting biological interpretations. This work aims to provide researchers with actionable insights to mitigate preprocessing-induced variance, ultimately enhancing the reliability and reproducibility of RNA-Seq studies.

Preliminary Results

Code Availability & Reproducibility

All code will be open sourced and available on GitHub. The repository will contain a `docker-compose.ylm` file that should allow the exact development environment to be recreated. An `R` package is insufficient due to a heavy use of command line tools, some of which are platform dependent. All code targets `linux` and builds off of the official `Bioconductor` docker image.

Preliminary Methodology Section

Table 1: Basic RNA-Seq Differential Analysis End-to-End Pipeline

Pipeline Steps	Software	Options	Choices
1. Pull SRA data from the NIH.	prefetch	NA	NA
2. Compute quality scores.	fasterq-dump	–skip-technical –threads X	Boolean Integer
3. Filter low quality reads.	fastp	–qualified_quality_phred X –length_required X	Integer Integer
4. Trim excess bases.	fastp	–trim_poly_g –trim_ploy_x	Boolean Boolean

5. Align and count genes.	Various	Default	Salmon, Kallisto
6. Count normalization.	edgeR	calcNormFactors(method='X')	TMM, RLE, upperquartile
7. Differential expression analysis.	edgeR	Default	NA

Statistical Model

Assume there are n samples of G gene counts. Let B_{gi} denote the count for gene g in sample i reported to the NIH database, and let C_{giX} denote the count obtained from pipeline with choices X. Similar let D_g and E_{gX} denote the p-values obtained from `edgeR`. Now,

$$Y_{1X} = \frac{1}{nG} \sum_{i=1}^{n} \sum_{g=1}^{G} (C_{giX} - B_{gi})^2 \tag{1}$$

and

$$Y_{2X} = \frac{1}{G} \sum_{g=1}^{G} (E_{gX} - D_g)^2 \tag{2}$$

Our primary analysis will focus on the two following regression models:

$$Y_{1X} = \beta_0 + \sum_{i=1}^{p} \beta_i X_i + \sum_{1 \leq i < j \leq p} \beta_{ij}(X_i \times X_j) + \epsilon \tag{3}$$

and

$$Y_{2X} = \beta_0 + \sum_{i=1}^{p} \beta_i X_i + \sum_{1 \leq i < j \leq p} \beta_{ij}(X_i \times X_j) + \epsilon \tag{4}$$

where p is the number of pipeline choices from Table 1. The first model studies the effect of each pipeline choice, include all pairwise interactions, on the average square deviation from the official NIH count matrix. The second model does the same, but for the p-values from a differential expression analysis.

Simulated Regression Power Analysis

```r
set.seed(33025)

num_predictors <- 50
num_sig_predictors <- 5
alpha <- 0.05
effect_size_mean <- 0.1
effect_size_sd <- 0.05

iter <- 100
sample_sizes <- seq(from = 60, to = 200, by = 10)

simulate_reg <- function(sample_size, num_predictors, num_sig_predictors,
                         effect_size_mean, effect_size_sd){
    X <- matrix(rnorm(sample_size * num_predictors),
                nrow = sample_size, ncol = num_predictors)

    colnames(X) <- paste0("X", 1:num_predictors)

    true_coef <- rnorm(num_sig_predictors, effect_size_mean, effect_size_sd)
    true_coef <- c(true_coef, rep(0, num_predictors - num_sig_predictors))
    Y <- X %*% true_coef + rnorm(sample_size)
    data <- data.frame(Y, X)

    lm_fit <- lm(Y ~ X, data = data)
    p_values <- summary(lm_fit)$coefficient[, 4][-1]

    return(p_values)
}

power_reg <- function(sample_sizes, num_predictors, num_sig_predictors,
                      effect_size_mean, effect_size_sd, iter){
    df <- data.frame()
    for(sample_size in sample_sizes){
        type_2_errors <- 0
        type_1_errors <- 0
        for(i in 1:iter){
            p_values <- simulate_reg(
                sample_size, num_predictors, num_sig_predictors,
                effect_size_mean, effect_size_sd
            )
            type_2_errors <- type_2_errors +
                sum(p_values[num_sig_predictors] > alpha)

            type_1_errors <- type_1_errors +
                sum(p_values[(num_sig_predictors + 1):num_predictors] < alpha)
        }
        power <- 1 - (type_2_errors / (iter * num_sig_predictors))
        type_1_error_rate <- (
            type_1_errors / (iter * (num_predictors - num_sig_predictors))
        )
        df <- rbind(df, c(sample_size, power, type_1_error_rate))
    }
    colnames(df) <- c("n", "Power", "Type I Error Rate")
    return(df)
}

power_df <- power_reg(sample_sizes, num_predictors, num_sig_predictors,
          effect_size_mean, effect_size_sd, iter)
```

```r
power_df |> ggplot(aes(x = n, y = Power)) +
            geom_line() +
            theme_bw()
```

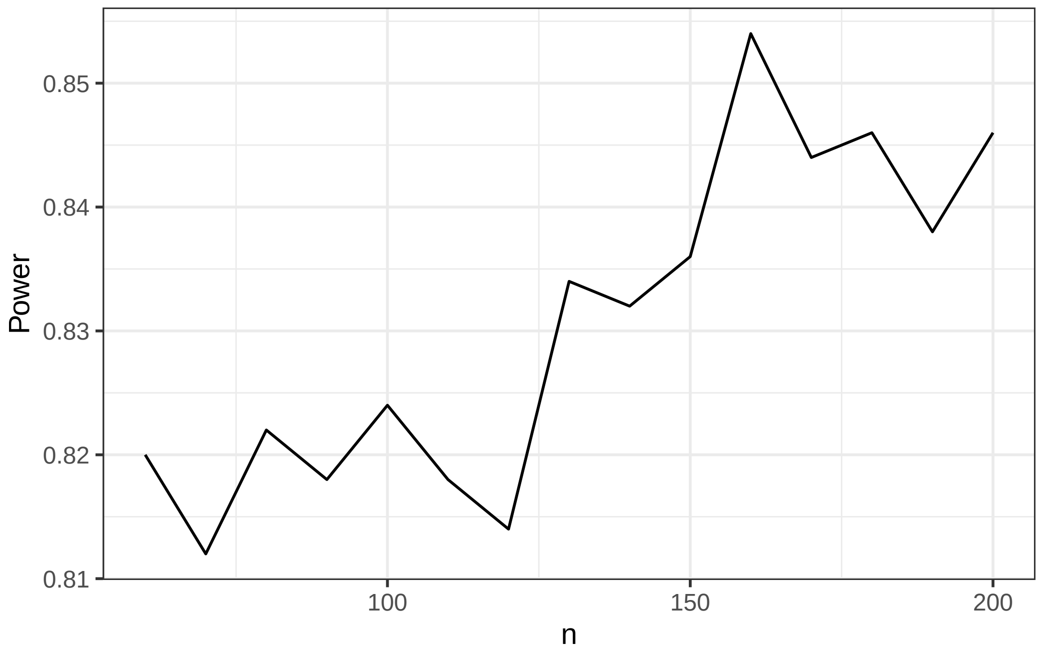

Figure 1: Multiple Linear Regression t-test Simulation Based Power Curve

Quality Score Variance Due to Fasterq-dump Options

```
#!/bin/bash

ACCESSION_LIST="SRR_Acc_List.txt"

mkdir -p SRA
mkdir -p dump_1 dump_2 dump_3

while read -r SRR_ID; do

    echo "Processing SRR ID: $SRR_ID"

    prefetch $SRR_ID --output-directory SRA

    # Option 1: Default
    fasterq-dump SRA/$SRR_ID/$SRR_ID.sra --outdir dump_1 --split-files --progress

    # Option 2: Skip Technical
    fasterq-dump SRA/$SRR_ID/$SRR_ID.sra --outdir dump_2 --split-files --progress --skip-technical

    # Option 3: 10 Threads
    fasterq-dump SRA/$SRR_ID/$SRR_ID.sra --outdir dump_3 --split-files --progress --threads 10

done < "$ACCESSION_LIST"

echo "Processing complete!"
```

```r
if (!require("BiocManager", quietly = TRUE))
    install.packages("BiocManager")

BiocManager::install("ShortRead")
BiocManager::install("Rsubread")
```

```r
library(ShortRead)

sample_1_fq_1 <- readFastq("./data/dump_1/SRR31476642.fastq")
sample_1_fq_2 <- readFastq("./data/dump_2/SRR31476642.fastq")
sample_1_fq_3 <- readFastq("./data/dump_3/SRR31476642.fastq")
```

```r
sample_1_fq_1_qual <- as(quality(sample_1_fq_1), "matrix")
sample_1_fq_2_qual <- as(quality(sample_1_fq_2), "matrix")
sample_1_fq_3_qual <- as(quality(sample_1_fq_3), "matrix")

sample_1_fq_13_qual_diff <- sample_1_fq_1_qual - sample_1_fq_3_qual
sample_1_fq_12_qual_diff <- sample_1_fq_1_qual - sample_1_fq_2_qual
```

```r
mean(sample_1_fq_13_qual_diff)
mean(sample_1_fq_12_qual_diff)
```

None of the `Fasterq-dump` options we tested resulted in differing quality scores.

Sampling Count Matrices under Different Pipeline Choices

Below is a script to sample 1 count matrix from an random pipeline that uses salmon as its aligner.

```bash
#!/bin/bash

THREADS=8
SALMON_INDEX="salmon_index"
TX2GENE="tx2gene.csv"
SRR_LIST="../SRR_Acc_List.txt"  # File containing SRR IDs, one per line
FASTQ_DIR="../fastq_data"
OUTPUT_DIR="salmon_count_matrices"
R_SCRIPT="generate_salmon_count_matrix.R"

mkdir -p "$OUTPUT_DIR" "$FASTQ_DIR"

TEMP_DIR=$(mktemp -d)

PARAMS_STR=""
# Randomly select parameters.
QUAL_PHRED=$(awk -v min=20 -v max=30 'BEGIN{srand(); print int(min+rand()*(max-min+1))}')
LEN_REQ=$(awk -v min=30 -v max=50 'BEGIN{srand(); print int(min+rand()*(max-min+1))}')

TRIM_G=$((RANDOM % 2))
TRIM_X=$((RANDOM % 2))

PARAMS_STR="Q${QUAL_PHRED}_L${LEN_REQ}_G${TRIM_G}_X${TRIM_X}"
```

```
echo "Processed files with QUAL_PHRED=$QUAL_PHRED, LEN_REQ=$LEN_REQ, TRIM_G=$TRIM_G, TRIM_X=$TRIM_X"

conda install -c bioconda fastp salmon

for FILE in "$FASTQ_DIR"/*.fastq; do

    echo "Now processing $FILE..."

    BASENAME=$(basename "$FILE" .fastq)

    # Filter and trim based on sampled parameters.
    fastp \
        --in1 "$FILE" \
        --qualified_quality_phred "$QUAL_PHRED" \
        --length_required "$LEN_REQ" \
        $( [ "$TRIM_G" -eq 1 ] && echo "--trim_poly_g" ) \
        $( [ "$TRIM_X" -eq 1 ] && echo "--trim_poly_x" ) \
        --out1 "$TEMP_DIR/${BASENAME}_trimmed.fastq" \
        --json "$TEMP_DIR/${BASENAME}_fastp.json" \
        --html "$TEMP_DIR/${BASENAME}_fastp.html"

    echo "hello"

    # Run Salmon quantification
    salmon quant -i "$SALMON_INDEX" \
        -l A \
        -r "$TEMP_DIR/${BASENAME}_trimmed.fastq" \
        -o "$TEMP_DIR/${BASENAME}_salmon" \
        --gcBias --seqBias --validateMappings

done

# Define output file with selected parameters
FINAL_COUNT_MATRIX="$OUTPUT_DIR/gene_count_matrix_${PARAMS_STR}.csv"

# Run R script to generate gene count matrix
Rscript "$R_SCRIPT" "$TEMP_DIR" "$FINAL_COUNT_MATRIX"

# Remove temporary files
rm -rf "$TEMP_DIR"

echo "Pipeline complete. Final count matrix stored in $FINAL_COUNT_MATRIX"
```

References

Arora, S., Pattwell, S. S., Holland, E. C., and Bolouri, H. (2020), "Variability in estimated gene expression among commonly used RNA-seq pipelines," *Scientific reports*, Nature Publishing Group UK London, 10, 2734.

Tong, L., Wu, P.-Y., Phan, J. H., Hassazadeh, H. R., Tong, W., and Wang, M. D. (2020), "Impact of RNA-seq data analysis algorithms on gene expression estimation and downstream prediction," *Scientific reports*, Nature Publishing Group UK London, 10, 17925.

Vieth, B., Parekh, S., Ziegenhain, C., Enard, W., and Hellmann, I. (2019), "A systematic evaluation of single cell RNA-seq analysis pipelines," *Nature communications*, Nature Publishing Group UK London, 10, 4667.