Shaleigh Smith
Next Generation Sequencing Informatics
Assigned Coursework #5

The nanopore fastq file analyzed in this assignment consisted of long reads obtained from VZV-infected ARPE-19 cells. The nanopore data was aligned first to the human genome, hg38, then to the human transcriptome, Homo_sapiens.GRCh38.all.cds.ncrna.fa, with minimap2. The parameters for both were as follows: map long reads with cigar (-ax), enable splice alignment enabled (splice), find splice sites on the transcript (-uf), kmer size of 14 (-k14), and no secondary alignments (--secondary=no). Secondary alignments were not including in output because this study required one gene alignment per read rather than multiple gene alignments per read. Without specifying the parameter, the feature counts output is drastically increased per gene. After the dataset was aligned to the genome and transcriptome each sam file output was converted to bam and sorted using samtools.

The genome gene counts were generated from the sorted bam file using featureCounts from subread. The feature counts command specified unpaired strand-specific read counting (-s 1) for long reads (-L) that was grouped by gene id (-g gene_id). The transcriptome gene counts were generated manually in R from the transcriptome sam file. First, the sam file was truncated to the first 4 rows and filtered for only forward mapping reads that matched a transcript (flag 0, no *). Then a counts column was added to the truncated sam file data frame. This was aggregated and summed per gene, resulting in a transcriptome gene count data frame. The transcript counts were converted to gene counts using an ensembl id conversion table.

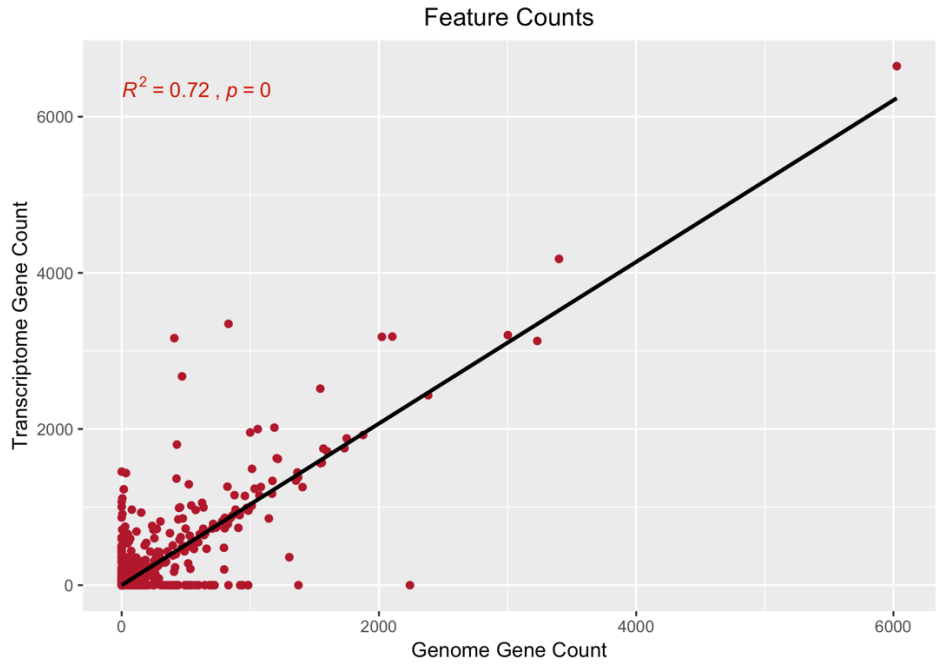
The genome and transcriptome gene count data frames were merged in R then plotted using ggplot2. A regression line was added to the plots and the Pearson's coefficient of determination was calculated and visualized by stat_cor. The data was plotted two ways: the first included genes that had zero counts for both the transcriptome and genome (figure 1a) and the second did not include genes that had zero counts for both (figure 1b). Regardless of removing zeros, the $R^2$ values were very similar: 0.07223 for the data frame including zeros and 0.07164 for the data frame without zeros. The p value was 0 for both plots.

The proportion of reads mapping within 50nt of the 5' end of transcripts was calculated manually in R using the truncated transcriptome sam file including a non-aggregated count column. First, an empty column (within_prime) was added to the data frame: if the position of the read was within 50 basepairs of the beginning of the transcript a 1 was put in this column, otherwise it was filled with a 0. This was done per read. This new data frame was then aggregated and summed per

transcript, resulting in a counts column that gave the total number of reads that mapped to any position in the transcript and a within_prime column that gave the total number of reads that mapped within the 5’ end of the transcript. These columns were divided, per transcript, then the results were averaged to give the final proportion of 0.506328.

*See below for figures and code.*

**A**




**B**

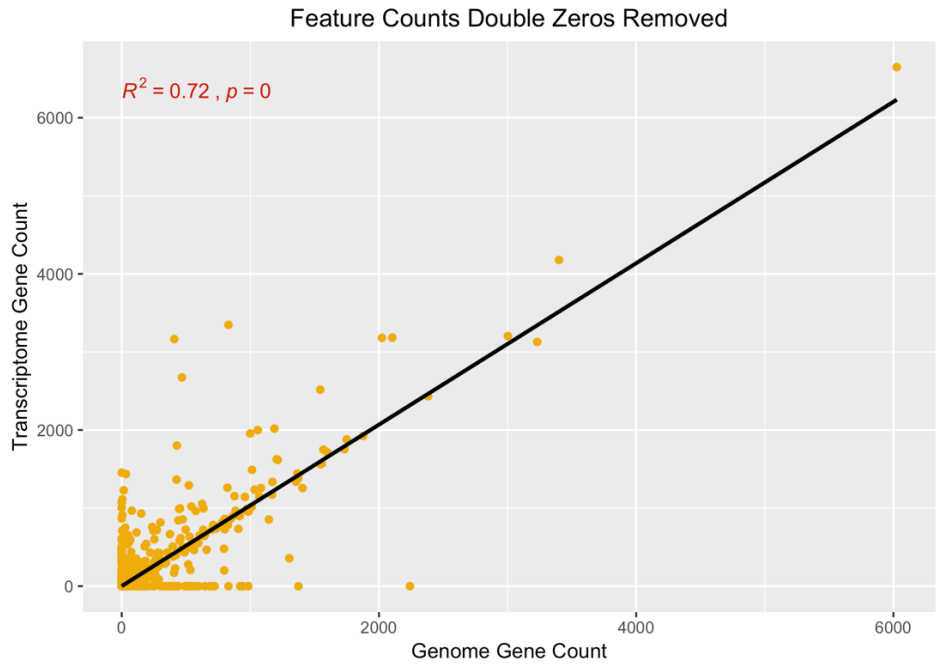



**Figure 1.** Gene counts (genome) vs. gene counts (transcriptome). The black line is a linear model regression line; the Pearson's Coefficient of Determination ($R^2$) and the p value (p) are written in red. **a**. Scatter plot of gene counts for the genome and transcriptome (red). Counts where transcriptome and genome are *both* equal to zero are included. **b**. Scatter plot of gene counts for the genome and transcriptome (yellow). Counts that are zero for *both* genome and transcriptome have been removed.

**Code:**

### Script for mapping nanopore fastq to genome with minimap2, sorting, converting to bam, and creating feature counts output

```
#!/bin/bash
#SBATCH --job-name=minimap2_genome # Job name
#SBATCH --mail-type=END,FAIL # Mail events (NONE, BEGIN, END, FAIL, ALL)
#SBATCH --mail-user=shaleigh.smith@nyulangone.org # Where to send mail
#SBATCH --ntasks=4 # Run on mulitple CPU
#SBATCH --mem=32gb # Job memory request
#SBATCH --time=2:00:00 # Time limit hrs:min:sec
#SBATCH --output=/gpfs/scratch/sas1531/ngs5_coursework/mimimap2_genome_%j.log # Standard output and error log
#SBATCH -p cpu_short

### This script is to align nanopore data against the human genome with minimap2 and generate gene counts

module load minimap2/2.15
module load samtools/1.9
module load subread/1.6.3

# Align with minimap2
# -ax: long reads with cigar
# -k14: kmer size
# --cs=long: output the cigar string (not necessary but could be interesting)
# splice: enable splice alignment
# -uf: find splice sites on transcript strand
# --secondary: Whether to output secondary alignments
minimap2 -ax splice -uf -k14 --secondary=no
/gpfs/scratch/sas1531/hg38/Homo_sapiens/UCSC/hg38/Sequence/WholeGenomeFasta/genome.fa
/gpfs/scratch/sas1531/ngs5_coursework/nanopore.fastq >
/gpfs/scratch/sas1531/ngs5_coursework/nanopore_genome.sam

# Get stats, convert sam to bam, sort
# -S: input sam
# -b: output bam
# --secondary: Whether to output secondary alignments [yes]
samtools view -S -b /gpfs/scratch/sas1531/ngs5_coursework/nanopore_genome.sam >
/gpfs/scratch/sas1531/ngs5_coursework/nanopore_genome.bam
samtools sort /gpfs/scratch/sas1531/ngs5_coursework/nanopore_genome.bam -o
/gpfs/scratch/sas1531/ngs5_coursework/nanopore_genome.sorted.bam

# Generate gene counts (this is NOT paired)
# -a: annotation file
# -s: strand-specific read counting - 1
# -L: used when counting long reads
# -M: multi-mapping reads/fragments will be counted
# -g: specifies the attribute to group
# -O: reads (or fragments if -p is specified) will be al- lowed to be assigned to more than one matched meta-feature
featureCounts -s 1 -L -a
/gpfs/scratch/sas1531/hg38/Homo_sapiens/UCSC/hg38/Annotation/Genes.gencode/genes.gtf -g gene_id -o
/gpfs/scratch/sas1531/ngs5_coursework/feature_counts_genome
/gpfs/scratch/sas1531/ngs5_coursework/nanopore_genome.sorted.bam
```

### Script for mapping nanopore fastq to transcriptome with minimap2, sorting, converting to bam, and creating truncated sam file to be manipulated in R

```
#!/bin/bash
#SBATCH --job-name=minimap2_transcriptome # Job name
#SBATCH --mail-type=END,FAIL # Mail events (NONE, BEGIN, END, FAIL, ALL)
#SBATCH --mail-user=shaleigh.smith@nyulangone.org # Where to send mail
#SBATCH --ntasks=4 # Run on mulitple CPU
#SBATCH --mem=32gb # Job memory request
#SBATCH --time=2:00:00 # Time limit hrs:min:sec
#SBATCH --output=/gpfs/scratch/sas1531/ngs5_coursework/minimap2_transcriptome _%j.log # Standard output and error log
#SBATCH -p cpu_short

### This script is to align nanopore data against the human transcriptome with minimap2 and generate transcript counts

module load minimap2/2.15
module load samtools/1.9
module load subread/1.6.3

# align nanpore data against human transcriptome with minimap2
# /gpfs/scratch/sas1531/ngs5_coursework/Homo_sapiens.GRCh38.all.cds.ncrna.fa

# Align with minimap2
# -ax: long reads with cigar
# -k14: kmer size
# splice: enable splice alignment
# -uf: find splice sites on transcript strand
# --secondary: Whether to output secondary alignments
minimap2 -ax splice -uf -k14 --secondary=no
/gpfs/scratch/sas1531/ngs5_coursework/Homo_sapiens.GRCh38.all.cds.ncrna.fa
/gpfs/scratch/sas1531/ngs5_coursework/nanopore.fastq >
/gpfs/scratch/sas1531/ngs5_coursework/nanopore_transcriptome_secondary.sam

# Get stats, convert sam to bam, sort
# -S: input sam
# -b: output bam
samtools view -S -b /gpfs/scratch/sas1531/ngs5_coursework/nanopore_transcriptome_secondary.sam >
/gpfs/scratch/sas1531/ngs5_coursework/nanopore_transcriptome_secondary.bam
samtools sort /gpfs/scratch/sas1531/ngs5_coursework/nanopore_transcriptome_secondary.bam -o
/gpfs/scratch/sas1531/ngs5_coursework/nanopore_transcriptome_secondary.sorted.bam

### Select first 4 columns of sam, this will be imported into R
samtools view nanopore_transcriptome_secondary.sam | cut -f 1,2,3,4 > truncated_transcript_secondary.sam
```

### R script for creating feature counts output for transcriptome, comparing and plotting gene counts vs. transcript counts, calculating Pearson's coefficient of determination, and determining the proportion of near full length reads

```
# Load Libraries
library(tidyverse)
library(plyr)
library(dplyr)
library(ggplot2)
library(ggpubr)
library(data.table)

### Read in files ###
# Read truncated sam (secondary=no)
# This only contains reads, sam flags, transcript ids, and start position of the read in respect to the transcript
transcript_flag <- read.table("truncated_transcript_secondary.sam", skip = 0, header = FALSE, sep = "\t", fill = TRUE)
as.tibble(transcript_flag)

# Read transcript id gene to transcript id conversion file
gene_conversion <- read.table("GeneID-TranscriptID.txt", skip = 0, header = TRUE, sep = "\t", fill = TRUE)
as.tibble(gene_conversion)

# Load genome counts table
# This is the direct result of feature counts for the genome data against the hg38 gtf file
genome <- read.table("feature_counts_genome", skip = 0, header = TRUE, sep = "\t", fill = TRUE)
as.tibble(genome)

### Clean and quantify feature counts for transcriptome ###
# Rename columns and change gene_id columns to character for join
colnames(transcript_flag)[1:4] <- c("read", "flag", "transcript_id", "position")
colnames(gene_conversion)[1:2] <- c("gene_id", "transcript_id")
gene_conversion$transcript_id <- as.character(gene_conversion$transcript_id)
gene_conversion$gene_id <- as.character(gene_conversion$gene_id)

# Add column for feature count
# Select for forward reads that mapped to a transcript
transcript_flag$feature_count_transcript <- 1
transcript_flag <- filter(transcript_flag, (flag == 0 & transcript_id != '*'))

# Remove extra columns and remove decimal places from transcript id
transcript <- dplyr::select(transcript_flag, -read, -flag, -position)
transcript$transcript_id <- substring(transcript$transcript_id, 1, 15)

# Merge with gene_ids, replace NA with 0, aggregate by gene id (sum)
transcriptome <- join(gene_conversion, transcript, by = "transcript_id", type = "inner")
transcriptome <- dplyr::select(transcriptome, -transcript_id)
transcriptome[is.na(transcriptome)] <- 0
transcriptome <- aggregate(. ~ gene_id, data = transcriptome, sum)

### Clean and join transcriptome and genome counts for plotting ###
# Clean up genome counts
colnames(genome)[c(1, 7)] <- c("gene_id", "feature_count_gene")
genome <- dplyr::select(genome, gene_id, feature_count_gene)
```

```
genome$gene_id <- substring(genome$gene_id, 1, 15)

# Join genome and transcriptome for plotting, replace NA with 0
ome <-  join(genome, transcriptome, by = "gene_id", type = "full") #left
ome[is.na(ome)] <- 0

# If both counts have zero, remove the row
ome_zero <- ome[apply(ome[,-1], 1, function(x) !all(x==0)),]

### Make scatter plots ###
# Plot with double zeros
plot_1 <- ggplot(data = ome, mapping = aes(x = feature_count_gene, y = feature_count_transcript)) +
  geom_point(color = "#B2182B") +
  xlab("Genome Gene Count") + ylab("Transcriptome Gene Count") +
  ggtitle("Feature Counts") + theme(plot.title = element_text(hjust = 0.5)) +
  geom_smooth(method = lm, se = FALSE, col = "black")
plot_1 <- plot_1 + stat_cor(aes(label = paste(..rr.label.., ..p.label.., sep = "~`,`~")),label.x = 3, col = "red3")
tiff("count_with_double_zeros.tiff", units="in", width=7, height=5, res=300)
plot_1
dev.off()

# Calculate R^2
# Adding the 0 term tells the lm() to fit the line through the origin
lm_1 <- lm(feature_count_gene ~ feature_count_transcript, data = ome)
lm_10 <- lm(feature_count_gene ~ 0 + feature_count_transcript, data = ome)
summary(lm_1)
summary(lm_10)

# Plot without double zeros
plot_2 <- ggplot(data = ome_zero, mapping = aes(x = feature_count_gene, y = feature_count_transcript)) +
  geom_point(color = "darkgoldenrod2") +
  xlab("Genome Gene Count") + ylab("Transcriptome Gene Count") +
  ggtitle("Feature Counts Double Zeros Removed") + theme(plot.title = element_text(hjust = 0.5)) +
  geom_smooth(method = lm, se = FALSE, col = "black")
plot_2 <- plot_2 + stat_cor(aes(label = paste(..rr.label.., ..p.label.., sep = "~`,`~")),label.x = 3, col = "red3")
tiff("count_without_double_zeros.tiff", units="in", width=7, height=5, res=300)
plot_2
dev.off()

# Calculate R^2
# Adding the 0 term tells the lm() to fit the line through the origin
lm_2 <- lm(feature_count_gene ~ feature_count_transcript, data = ome_zero)
lm_20 <- lm(feature_count_gene ~ 0 + feature_count_transcript, data = ome_zero)
summary(lm_2)
summary(lm_20)

### Quantify transcripts within 5' end ###
# Add column for 5' range:
# If the read is wihtin 50 basepairs of the 5' end of the transcript, put a one in the column
# If the read isn't withing 50 basepairs, put a 0
# Remove decimals in transcript ids
transcript_flag$within_prime <- NA
transcript_flag$within_prime[transcript_flag$position > 0 & transcript_flag$position <= 50] <- 1
transcript_flag$within_prime[transcript_flag$position > 50] <- 0
transcript_flag$transcript_id <- substring(transcript_flag$transcript_id, 1, 15)

# Aggregate transcripts, sum
# This gives the total count of reads within the 5' end per transcript
```

```
transcript_5 <- dplyr::select(transcript_flag, -read, -flag, -position)
transcript_5 <- aggregate(. ~ transcript_id, data = transcript_5, sum)

# Create ratio column
# The ratio column is the number of reads withing 5' end divided by total number of reads in that transcript
transcript_5$ratio <- (transcript_5$within_prime / transcript_5$feature_count_transcript)

# Find the mean of the ratio column
ratio_mean <- mean(transcript_5$ratio)

# The proportion is 0.506328
ratio_mean
```