Erica Chio, Assigned coursework #7

Due date: April 16th (two weeks), 11pm EST
Submission: PDF document by email to daniel.depledge@nyulangone.org

- You are being provided with a recent nanopore direct RNA sequencing dataset (RNA-002) that was sequenced on a MinIONflowcell (9.4.1D)
- The aim of this assignment is to re-basecall the data, align to the human transcriptome, and determine the relative abundance of different transcripts.
- This will test your ability to use Guppy and MiniMap2 while also testing your data parsing skills.

1. Use the latest version of Guppy (BigPurple) to re-basecall data using sensible flags and output fastq

– note : look at tuesday lecture

- Explain your choice of flags

`/gpfs/data/courses/bminga3004/Practicum10/Assignment/Ad5-12h`

```
#!/bin/bash
#SBATCH --job-name=guppy # Job name
#SBATCH --mail-type=END,FAIL # Mail events (NONE, BEGIN, END, FAIL, ALL)
#SBATCH --mail-user=Erica.Chio@nyulangone.org # Where to send mail
#SBATCH --nodes=1 # Nodes
#SBATCH --gres=gpu:2 # Calling gpu nodes
#SBATCH --ntasks=8 # Run on a single CPU
#SBATCH --mem=32gb # Job memory request
#SBATCH --time=12:00:00 # Time limit hrs:min:sec
#SBATCH --output=guppy_%j.log # Standard output and error log
#SBATCH -p gpu8_short # Specifies location to submit job

module purge
module load guppy/3.4.5

# --compress_fastq Compress the fastq output
# --flowcell FLO-MIN106 (MinIONflowcell)
# --kit SQK-RNA002 (RNA-002)
# -r recursively go into directories
# --trim_strategy rna (because working with RNA dataset)
```

```
# --reverse_sequence true (reverse so its 5` to 3`)
# --u_substitution true (covert u to t for better alignment)
# -i where the raw read files are located
# -s where the basecalled files will be saved
# --device (similar to -x, assigning graphics card)

guppy_basecaller --compress_fastq --flowcell FLO-MIN106 --kit SQK-RNA002 -r --
trim_strategy rna --reverse_sequence true --u_substitution true -i /gpfs/data/
courses/bminga3004/Practicum10/Assignment/Ad5-12h/fast5_pass -s /gpfs/scratch/
ebc308/AIS/coursework7data/guppy_notcompressed --device "cuda:0 cuda:1"
```

2. Align to the human transcriptome using MiniMap2

- Explain your choice of flags

`minimap2 -a ref.fa query.fq > alignment.sam`

```
#!/bin/bash
#SBATCH --job-name=minimap2 # Job name
#SBATCH --mail-type=END,FAIL # Mail events (NONE, BEGIN, END, FAIL, ALL)
#SBATCH --mail-user=Erica.Chio@nyulangone.org # Where to send mail
#SBATCH --ntasks=8 # Run on a single CPU
#SBATCH --mem=16gb # Job memory request
#SBATCH --time=24:00:00 # Time limit hrs:min:sec
#SBATCH --output=minimap2_%j.log # Standard output and error log
#SBATCH -p cpu_short # Specifies location to submit job

module purge

module load minimap2/2.15

# took noisy Nanopore Direct RNA-seq flags
# -u CHAR How to find canonical splicing sites GT-AG - f: transcript strand;
# -k INT(14) Minimizer k-mer length

# -L Write CIGAR with >65535 operators at the CG tag. Older tools are unable to
convert alignments with >65535 CIGAR ops to BAM. This option makes minimap2 SAM
```

```
compatible with older tools. Newer tools recognizes this tag and reconstruct
the real CIGAR in memory.

minimap2 -ax map-ont -uf -k14 -L /gpfs/scratch/ebc308/AIS/coursework4data/
Homo_sapiens.GRCh38.cdna.all.fa /gpfs/scratch/ebc308/AIS/coursework7data/
basecalled.fastq.gz > aligned_ax_map_ont.sam
```

3. Filter reads to retain only those whose mapping extends from the 3′ end to 50 nt of the transcript start. Remove all supplemental alignments.

- Yes this is tricky. My recommended approach is to manipulate SAM/BAM or BED12 files to achieve this. Consider filtering on alignment flags and mapping position and don't forget that only reads mapping to the transcript (i.e. not the reverse strand) should be retained

SAM ➜ BAM ➜ BED

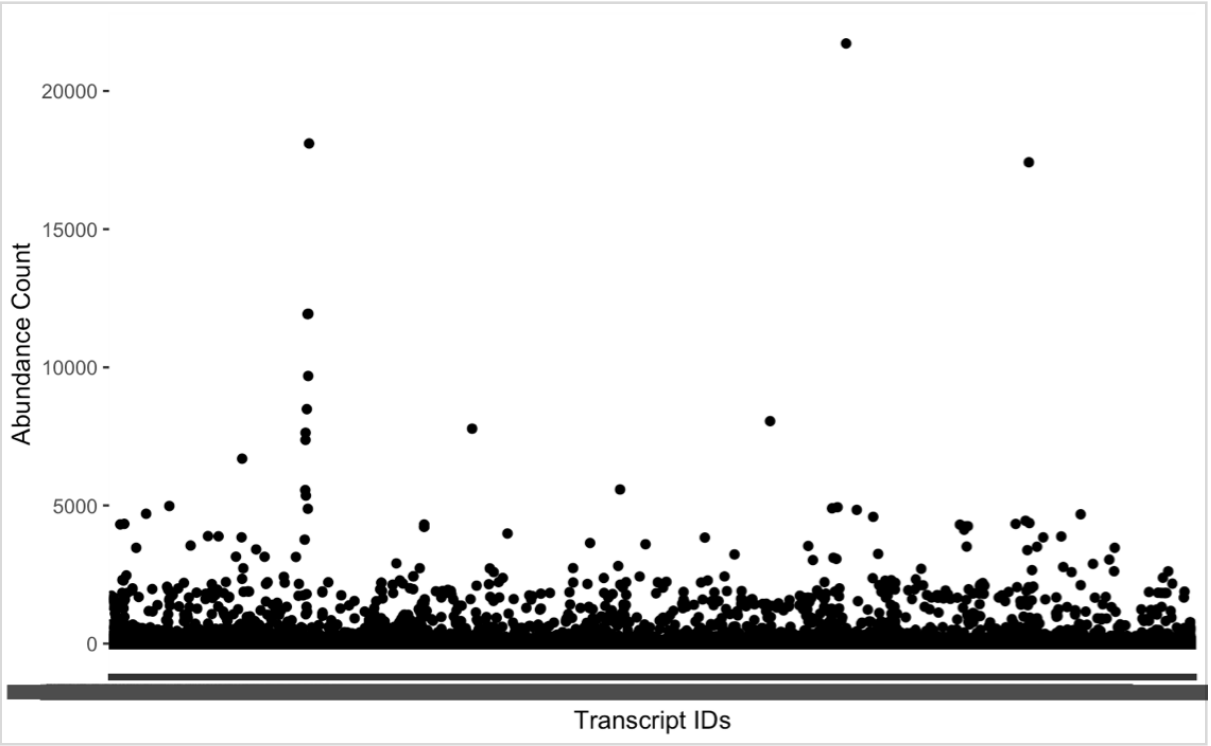
```
module purge

module load samtools/1.9-new
module load bedtools/2.27.1

samtools view -Sb /gpfs/scratch/ebc308/AIS/coursework7data/minimap2/
aligned_ax_map_ont.sam > aligned.bam

# -f only include
# -F filter out
samtools view -F 2324 aligned.bam

# -bed12 to make it a bed12 file
bamToBed -bed12 -i aligned.bam > aligned.bed
```

4. Generate a plot showing the number of transcripts identified and their abundance – do you see multiple transcript isoforms from the same gene?

- This is a data presentation exercise – get creative in your visualizations

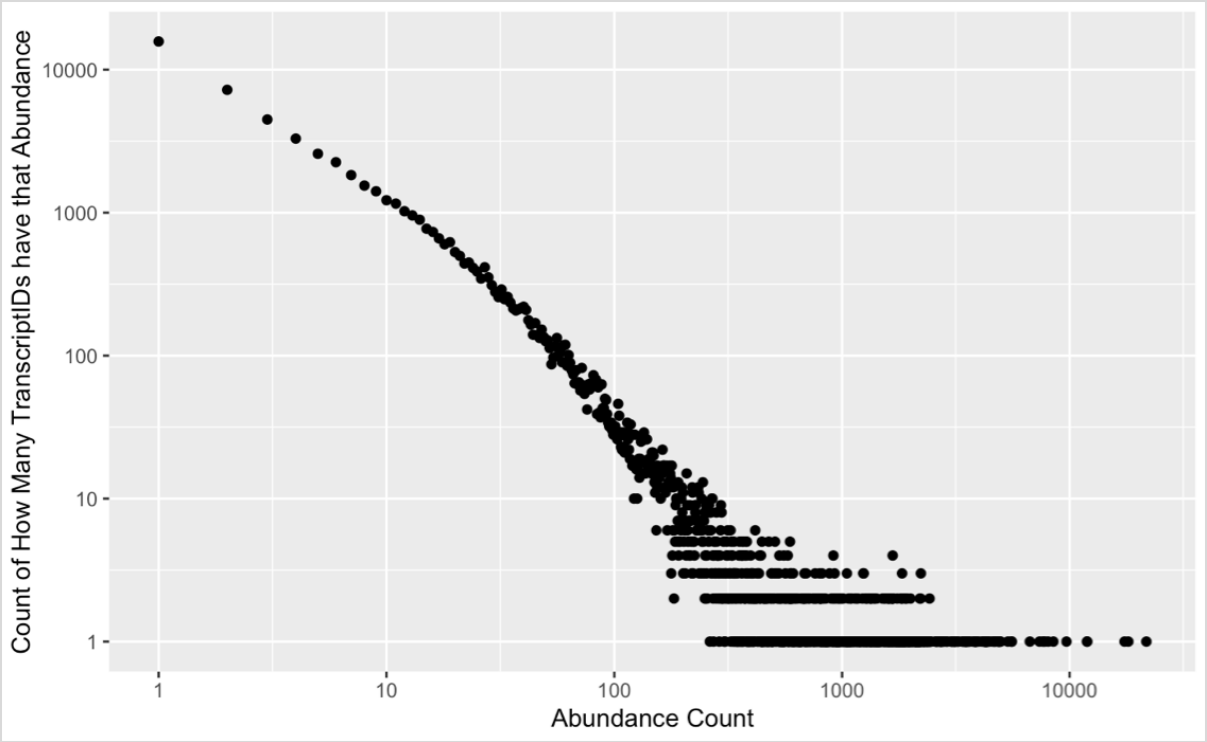
Abundance

```
library(dplyr)
library(ggplot2)
library(ggpubr)
library(scales)
library(tidyr)
library(biomaRt)

aligned <- read.table("aligned.bed", header = FALSE)
colnames(aligned) <- c("transcript", "start", "end", "name", "score", "strand",
"thickStart", "thickEnd", "itemRgb", "blockCount", "blockSizes", "blockStarts")
# head(aligned)

result <- aligned %>%
  filter(start <= 50)
# head(result)

nlevels(unique(result$transcript)) #64078 unique transcript IDs

# get trascript abundance by tallying how many times a transcript ID is mapped
transcript_abundance <- aligned %>% group_by(transcript) %>% tally()
transcript_abundance

# plotting transcript id transcript abundance
ggplot(transcript_abundance, aes(x=(transcript), y=n)) + geom_point()
```



This plot was too dense, unable to gather any information. So I thought of another way to try and visualize abundance counts. I tallied the abundance counts so that I had a new table - abundance counts & how many transcript IDs had that abundance

```
transcript_abundance_frequency <- transcript_abundance %>% group_by(n) %>%
tally()
head(transcript_abundance_frequency)

ggplot(freqFreq, aes(x=(n), y=nn)) + geom_point() + scale_x_log10() +
scale_y_log10() + labs(x = "Abundance Count", y = "Count of How Many
TranscriptIDs have that Abundance")
```

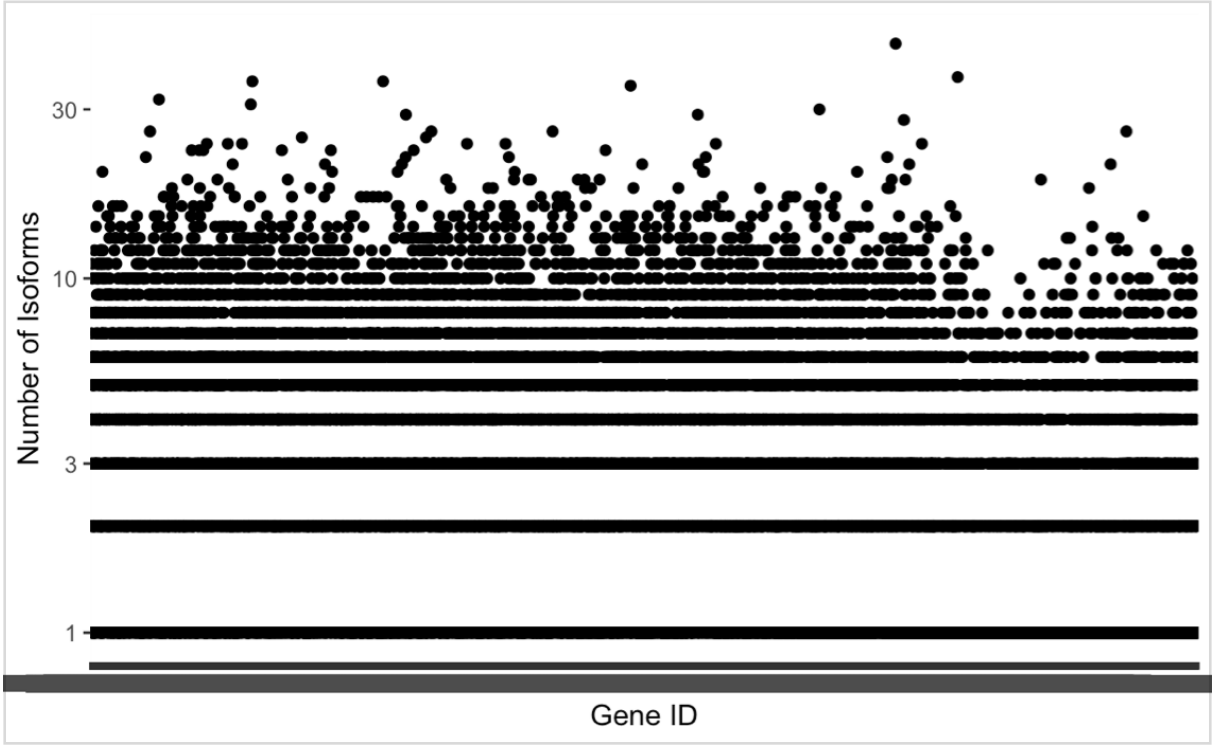


From this scatter plot, you can see that ~15k transcript IDs were only identified once, and that there is an inverse proportion between abundance count and amount of transcripts that had that abundance.

Gene Isoforms

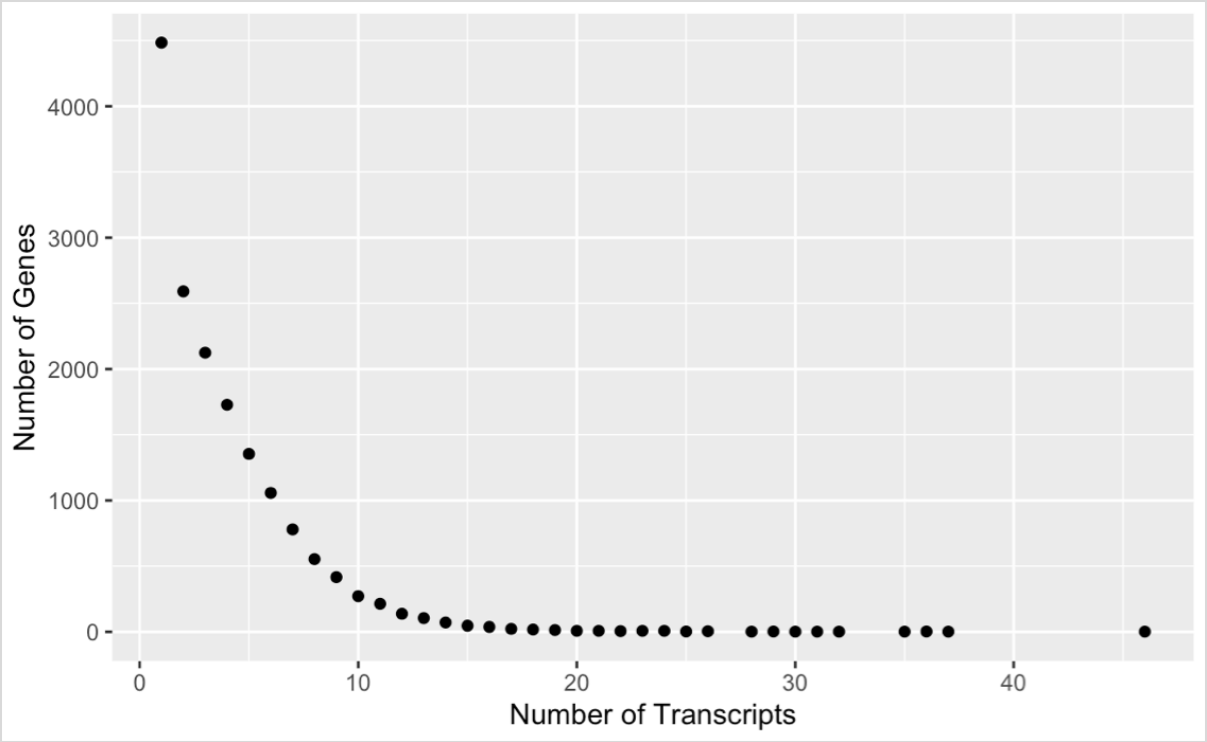
```
# define biomart object
mart <- biomaRt::useMart(biomart = "ensembl", dataset =
"hsapiens_gene_ensembl")
# ensembl_gene_id
t2g <- biomaRt::getBM(attributes = c("ensembl_transcript_id",
"transcript_version", "ensembl_gene_id", "external_gene_name"), mart = mart)

t2g$target_id <- paste(t2g$ensembl_transcript_id, t2g$transcript_version,
sep=".") # append version number to the transcript ID

t2g[,c("ensembl_transcript_id","transcript_version")] <- list(NULL) # delete
the ensembl
# head(t2g)

genes_transcripts <- merge(freq ,t2g, by.x="transcript", by.y="target_id")
colnames(genes_transcripts) <- c("transcript", "abundance", "gene_id",
"gene_name")
```

```
gene_isoforms <- genes_transcripts %>% group_by(gene_id) %>% tally()
# head(gene_isoforms)

ggplot(gene_isoforms, aes(x=(gene_id), y=n)) + geom_point() + scale_y_log10() +
labs(x = "Gene ID", y = "Number of Isoforms")
```



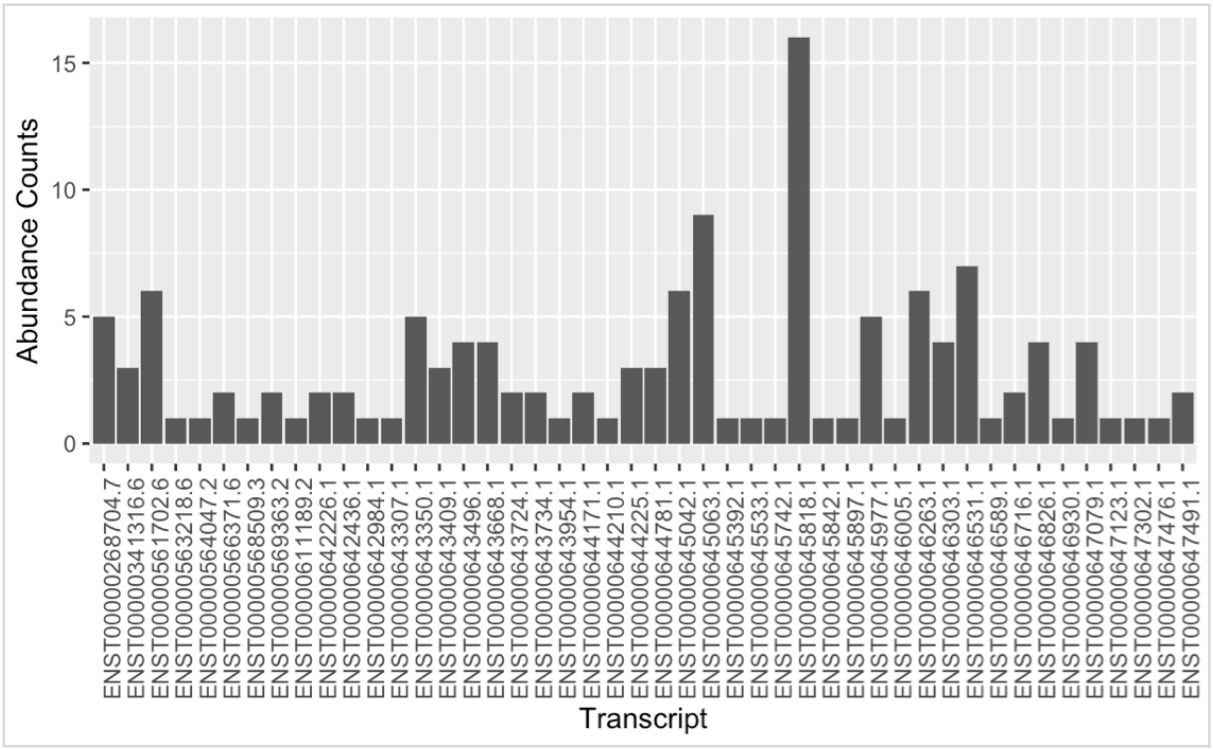
Similar to when graphing abundance, this plot was too dense. To try and visualize it differently, I plotted the frequency of the frequency.

```
isoform_freq <- gene_isoforms %>% group_by(n) %>% tally()
# head(isoform_freq)

ggplot(isoform_freq, aes(x=(n), y=nn)) + geom_point() + labs(x = "Number of
Transcripts", y = "Number of Genes")
```



This graph shows that fewer genes have more transcript variants.

```
# getting gene with most isoforms
colMax <- function(data) sapply(data, max, na.rm = TRUE)
colMax(gene_isoforms) #n = 46 isoforms

highest_num_isoform <- gene_isoforms %>% filter(n == 46)
highest_num_isoform # gene_id = ENSG00000197912

ENSG00000197912 <- genes_transcripts %>% filter(gene_id == "ENSG00000197912")
ggplot(ENSG00000197912, aes(x=(transcript), y=abundance)) +
geom_bar(stat="identity") +  labs(x = "Transcript", y = "Abundance Counts") +
theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



#appliedsequencinginformatics