Harvard Stat 115/215 HW 1

Jenna Landy

Finding the Stat 115 directory

Once you log in to Cannon, type `cd /n/stat115` to go to the Stat 115 directory. This folder contains all of the data needed to complete this assignment.

Problem 1: STAR alignment

We will give you a simple example to test high throughput sequencing alignment for RNA-seq data. Normally for paired-end sequencing data, each sample will have two separate FASTQ files, with line-by-line correspondence to the two reads from the same fragment. Read mapping could take a long time, so we have created just two FASTQ files of one RNA-seq sample with only 30,000 fragments (2 * 30,000 reads) for you to run STAR instead of the full data. The files are located at /2021/HW1/raw_data1. The mapping will generate one single output file.

Use STAR (Dobin et al, Bioinformatics 2012) to map the reads to the reference genome, available on Cannon at /2021/HW1/index/star_hg38_index. Use the paired-end alignment mode and generate the output in SAM format. Please include full STAR report. How many reads are mappable and how many are uniquely mappable?

```
Sequencing data:
/2021/HW1/raw_data1

module: STAR/2.6.0c-fasrc01
index: /2021/HW1/index/star_hg38_index
```

```
module load STAR/2.6.0c-fasrc01

STAR --genomeDir /n/stat115/2021/HW1/index/star_hg38_index --readFilesIn /n/stat115/2021/HW1/raw_data1/

cat Log.final.out
```

I used STAR to map the reads in the `raw_data` folder to the referenge genome. The STAR manual says that if using Illumina paired-end reads, the read1 and read2 files have to be supplied. There are two fastq files, `subA_l.fastq` and `subA_r.fastq1`, in `/n/stat115/2021/HW1/raw_data1/*`, so this paired-end alignment mode will be used as directed! The STAR manual says that SAM is the default output format, so I didn't add any extra options for this requirement. STAR results in five output files: `Aligned.out.sam`, `Log.final.out`, `Log.out`, `Log.process.out`, and `SJ.out.tab`. The information to answer the question is in `Log.final.out`. Here, I found that **28194 reads are mappable** (i.e. all of them), of which **19292 reads are uniquely mappable**. The full breakdown is as follows:

- Number of reads mapped: 28194, sum of:
  - Uniquely mapped reads number: 19292
  - Number of reads mapped to multiple loci: 8902
- Number of reads unmapped: 0

Full STAR Report:

```
                                 Started job on |   Feb 03 13:24:10
                               Started mapping on |   Feb 03 13:29:03
                                      Finished on |   Feb 03 13:29:07
       Mapping speed, Million of reads per hour |   26.99
(base) [bst282u2114@boslogin04 hw1p1]$
                             Number of input reads |   29988
                         Average input read length |   96
                                       UNIQUE READS:
                        Uniquely mapped reads number |   19292
                             Uniquely mapped reads % |   64.33%
                               Average mapped length |   95.62
                            Number of splices: Total |   2705
                 Number of splices: Annotated (sjdb) |   2675
                            Number of splices: GT/AG |   2678
                            Number of splices: GC/AG |   19
                            Number of splices: AT/AC |   2.50
                    Number of splices: Non-canonical |   6
                           Mismatch rate per base, % |   0.15%
                              Deletion rate per base |   0.01%
                             Deletion average length |   1.50
                             Insertion rate per base |   0.01%
                            Insertion average length |   1.28
                                 MULTI-MAPPING READS:   0.26%
        Number of reads mapped to multiple loci |   8902
             % of reads mapped to multiple loci |   29.69%
        Number of reads mapped to too many loci |   79
             % of reads mapped to too many loci |   0.26%
                                   UNMAPPED READS:
       % of reads unmapped: too many mismatches |   0.00%
                 % of reads unmapped: too short |   5.58%
                     % of reads unmapped: other |   0.14%
                                   CHIMERIC READS:
                         Number of chimeric reads |   0
                              % of chimeric reads |   0.00%
```

Problem 2: RNA-seq quality control

You are asked by a collaborator to analyze four RNA-seq libraries. She suspects that the libraries are generally of high-quality but is concerned that a sample may have been switched with her benchmates during processing. To save time, we have provided four bam files generated by STAR in /2021/HW1/raw_data2.

Please use RSeQC (Liguo Wang et al, Bioinformatics 2012) geneBody_coverage.py and tin.py modules to determine whether any of the samples exhibit unusual quality control metrics. To expedite the process, you can use housekeeping genes as reference, which provided in /2021/HW1/raw_data2. Overall, identify the best and worst libraries. Your answer should include screen shots and tables as necessary as if you were delivering a report to the collaborator.

data:
/2021/HW1/raw_data2/bamFile

```
# Set up RSeQC
module load Anaconda/5.0.1-fasrc02
conda create -n my_env python=2.7
source activate my_env
conda install -c bioconda rseqc
```

```
module load Anaconda/5.0.1-fasrc02
source activate my_env

# TIN
tin.py -i /n/stat115/2021/HW1/raw_data2/bamFile -r /n/stat115/2021/HW1/raw_data2/hg38.HouseKeepingGenes

tin.py -i /n/stat115/2021/HW1/raw_data2/bamFile/res_YAligned.sortedByCoord.out.bam,/n/stat115/2021/HW1/

# Gene Body Coverage
module load R

geneBody_coverage.py -i /n/stat115/2021/HW1/raw_data2/bamFile -r /n/stat115/2021/HW1/raw_data2/hg38.Hous
```

I used RSeQC to run `tin.py`, which computes the Transcript Integrity Number (TIN) for each transcript, and `geneBody_coverage.py`, which calculates gene body coverage and makes two plots. In running `tin.py`, I used the default minimum coverage of 10 reads. For `geneBody_coverage.py`, I used the default minimum mRNA length of 100 base pairs, as well as the default output format of pdf.

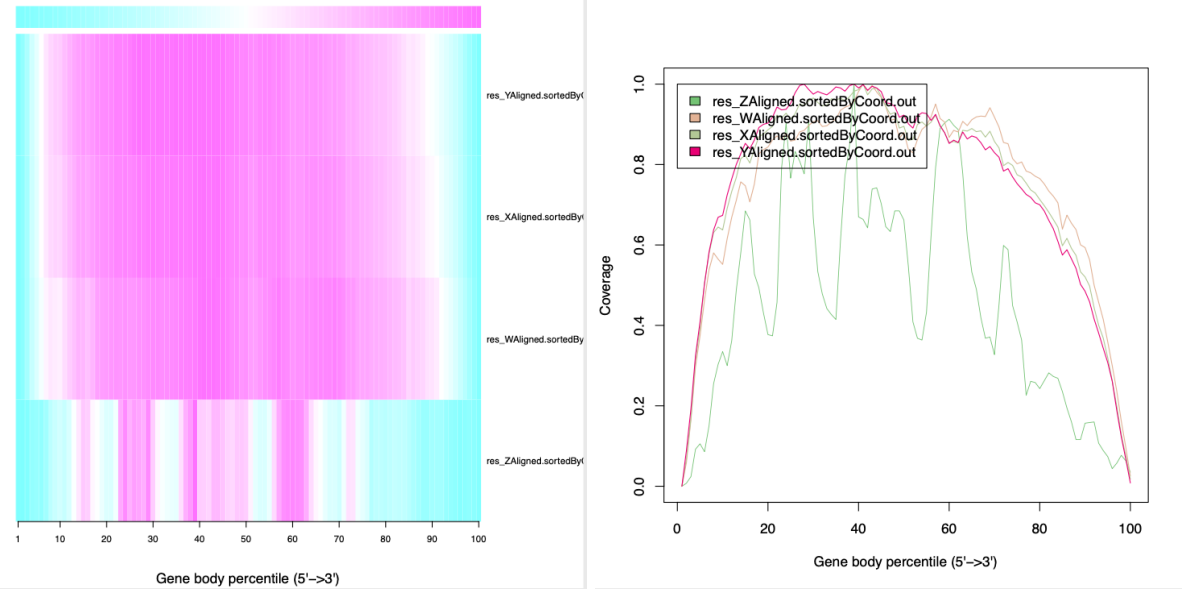
The summary statistics resulting from `tin.py` are displayed in the table below, as well as the two gene body coverage plots from `geneBody_coverage.py`.

Based on the summary statistics of transcript integrity numbers of the four different gene libraries, library Z has the worst integrity by far. Library Y has the best integrity, closely followed by library X. Library W has integrity a bit below X and Y, but no where near as bad as Z. I suspect that library Z may have been the one swapped with her benchmate's sample.

The heatmap displays the number of reads covering each nucleotide position in each gene library. The ordering, top to bottom, shows us the least skewed (best) to most skewed (worse) as determined by Pearson's skewness coefficient. This corroborates the idea that library Z is the worst, and library Y is the best. X is second best and W is closely behind. The line plot displays the same data as the heatmap. The skewness of library Z (green) is very clear. Further, library Z has much less read coverage than the other libraries.

Table 1: Summary Statistics

BAM.File	TIN..mean.	TIN..median.	TIN..stdev.
res_WAligned.sortedByCoord.out.bam	45.74958	45.48953	21.33583
res_XAligned.sortedByCoord.out.bam	53.78790	55.81712	20.62771
res_YAligned.sortedByCoord.out.bam	56.20032	58.76576	19.80363
res_ZAligned.sortedByCoord.out.bam	24.55179	21.92499	14.20632



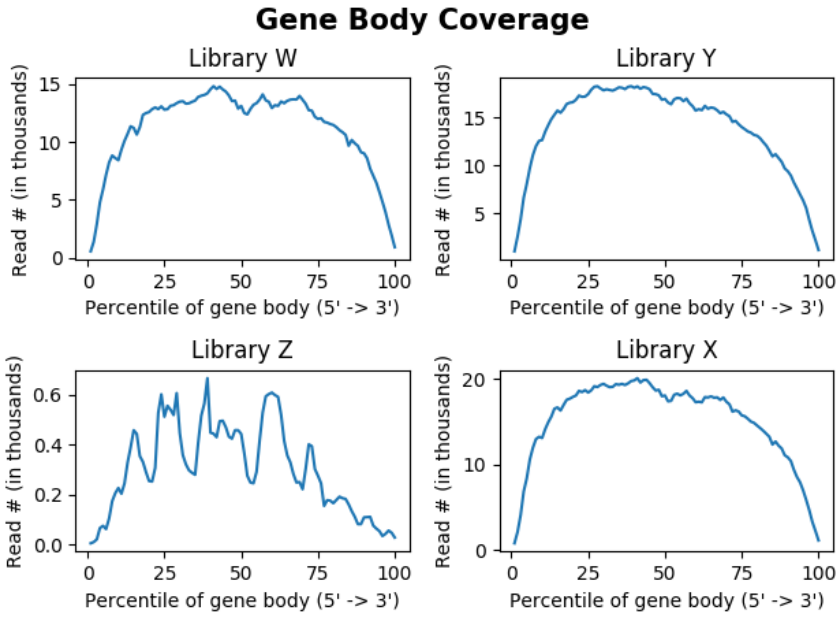
Problem 3: Python programming

One output file of RSeQC is called geneBodyCoverage.txt which contains normalized reads mapped to each % of gene / transcript body. Suppose that we want to visualize all 4 samples together to quickly perform quality control. Write a python program to extract the values and name from each file. The same script should then plot the gene body coverage for all the samples (2 rows x 2 cols) in one figure. We provide an example with 3 x 2 samples in one figure. Please identify the worst sample by your result.

This plot is the same as the curves plot in problem 2, but with the y axis as unscaled counts, and separated into four plots.

```
import pandas as pd
import matplotlib.pyplot as plt

with open('Out.geneBodyCoverage.txt') as f:
    lines = f.readlines()

percentiles = {}
for i in range(1,len(lines)):
    # skip i = 0, this is the header
    line_parts = lines[i].replace('\n','').split('\t')
    file = line_parts[0]
    percs = [float(x) for x in line_parts[1:]]
    percentiles[file] = percs

df = pd.DataFrame(percentiles)
x = range(1, 101)

fig, axs = plt.subplots(2, 2)
plt.suptitle('Gene Body Coverage', size = 15, weight = 'bold')
for i in range(len(df.columns)):
    col = df.columns[i]
    axs[int(i>1), i%2].plot(x, df[col]/1000)
    axs[int(i>1), i%2].set_xlabel("Percentile of gene body (5' -> 3')")
    axs[int(i>1), i%2].set_ylabel("Read # (in thousands)")
```

```
    axs[int(i>1), i%2].set_title(
        'Library ' + col.split('_')[1][0]
    )

plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```



Problem 4: RNA-seq quantification

Transcript quantification plays an important role in the analysis of RNA-seq data. A large number of tools have been developed to quantify expression at the transcript level. RSEM (Bo Li et al, BMC Bioinformatics 2011) is a software package for estimating gene and isoform expression levels from single-end or paired-end RNA-Seq data, it can perform the alignment step with three different aligners: bowtie, bowtie2, or STAR. Salmon (Rob Patro et al, Nature Methods 2017) is an ultra-fast alignment-free method which also can correct for GC-bias.

Please run STAR+RSEM and Salmon on one good quality sample from problem 2 and 3 to get FPKM and TPM. Identify the transcript and gene with the highest expression in this library from the Salmon output.

```
data: /2021/HW1/raw_data2/
module: STAR/2.6.0c-fasrc01, rsem/1.2.29-fasrc03, salmon/0.12.0-fasrc01
index: /2021/HW1/index/salmon_hg38_index,/2021/HW1/index/rsem_hg38_index
```

I used **library Y**, which was decided to be the best library in parts 2 and 3. First, I ran STAR with the `--quantMode TranscriptomeSAM` option, and set the output to a BAM file sorted by coordinates. This creates the file Aligned.toTranscriptome.out.bam, which us used as an input to RSEM. I then ran RSEM, and used the options recommended in lab with 4 threads. `Aligned.toTranscriptome.out.bam` is used as input, `/n/stat115/2021/HW1/index/rsem_hg38_index` is used as the reference file, and results are stored in `"RSEMOut"`. The `RSEMOut.isoforms.results` output file from RSEM includes length, effective length, expected count, TPM, FPKM, and IsoPct for each gene. Next, I ran salmon on the fastq file for library Y. I specified 4 threads and to automatically detect whether the input is paired end or single end. The `quant.sf` output file from salmon includes length, effective length, TPM, and the number of reads for each gene.

To view the top expressed gene in this library from the Salmon output, I sorted the `quant.sf` file by the TPM column (column 4, so `-k4` option) in descending order (`-r` option), and then displayed the top row with `head -n 1`. This reported the ENSEMBL Transcript ID `ENST00000361851`. I then used BiopMart to get the gene symbol from the transcript id. The gene is **MT-ATP8, mitochondrially encoded ATP synthase membrane subunit 8**.

```
module load STAR/2.6.0c-fasrc01

STAR --genomeDir /n/stat115/2021/HW1/index/star_hg38_index --readFilesIn /n/stat115/2021/HW1/raw_data2/


module load rsem/1.2.29-fasrc03

rsem-calculate-expression --no-bam-output --time --bam -p 8 Aligned.toTranscriptome.out.bam /n/stat115/


module load salmon/0.12.0-fasrc01
salmon quant -i /n/stat115/2021/HW1/index/salmon_hg38_index -l A -r /n/stat115/2021/HW1/raw_data2/fastq


cat SalmonOut/quant.sf | sort -k4 -n -r |  head -n 1
```

Problem 5: Speed Comparison

Report the relative speed of STAR+RSEM and Salmon for the analyses of the sample. Comment on your results based on the lecture material.

Running STAR and RSEM took a total of 515 seconds, while Salmon was much faster at 108 seconds. This makes sense with the lecture material, where we learned that while RSEM is considered the best quantitative approach, it's slow, and Salmon is much faster because it uses pseudo-alignment as opposed to full alignment.

STAR Time: 5 minutes 04 seconds (= 304 seconds)

- Started job on | Feb 05 12:26:13
- Started mapping on | Feb 05 12:30:44
- Finished on | Feb 05 12:31:17

RSEM Time: 3 minutes 41 seconds (= 211 seconds)

- Aligning reads: 0 s.
- Estimating expression levels: 211 s.
- Calculating credibility intervals: 0 s.

Salmon Time: 1 minute 48 seconds (= 108 seconds)

- start_time: Fri Feb 5 12:51:20 2021
- end_time: Fri Feb 5 12:53:08 2021

Problem 6:

Plot the relationship between effective length, normalized read counts, TPM, and FPKM for this sample from the RSEM and Salmon output. Comment on the relative utility of each metric when analyzing gene expression data.

First, I calculated FPKM (fragments per kilobase million) for the salmon results, which is the fragment count first scaled by the replicate scaling factor, and then scaled by gene length. I calculated the replicate scaling factor, which is the sum of fragment counts divided by 1 million. Then, each read number is first divided by this scaling factor, then by the length of the gene.

I combined the data from the two results, and then plotted a heatmap of the correlation matrix between effective length, normalized read counts, TPM, and FPKM from the RSEM and Salmon output.

The two other plots show the association between the TPM values produced by Salmon and RSEM, and the association between the FPKM produced by Salmon and RSEM.

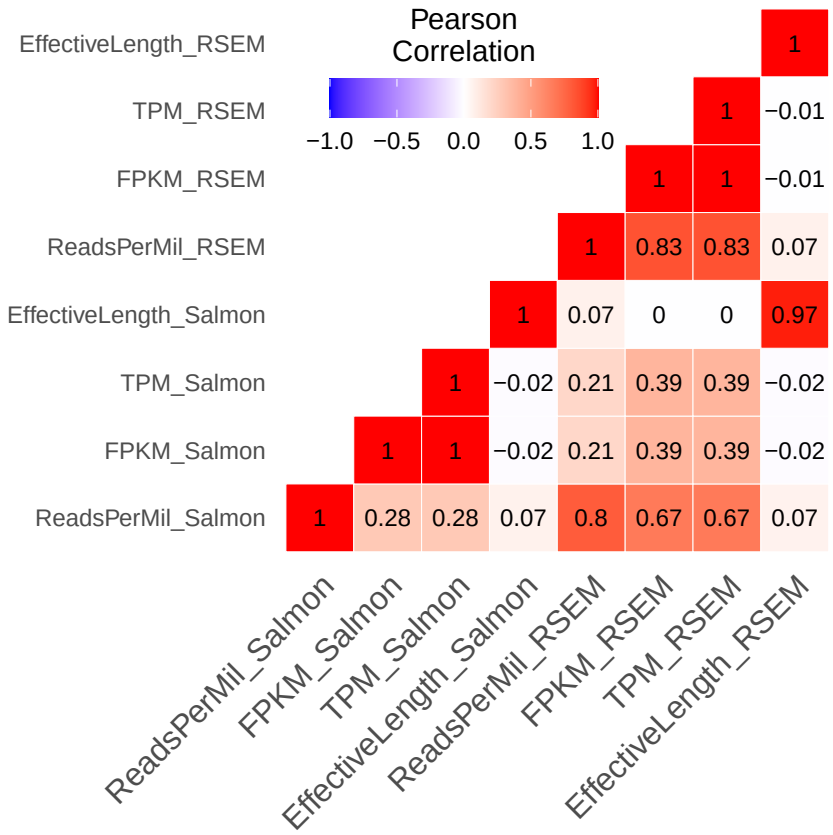
Interpretations are provided below each respective plot.

```
library(dplyr)
library(ggplot2)
```

```
Salmon <- read.table('quant.sf', header = 1) %>%
  mutate(
    ReadsPerMil_Salmon = NumReads/(sum(NumReads)/1000000),
    FPKM_Salmon = ReadsPerMil_Salmon/EffectiveLength,
    TPM_Salmon = TPM,
    EffectiveLength_Salmon = EffectiveLength
  ) %>%
  select(
    -Length, -NumReads,
    -EffectiveLength, -TPM
  )

RSEM <- read.table('RSEMOut.isoforms.results', header = 1) %>%
  mutate(
    ReadsPerMil_RSEM = expected_count/(sum(expected_count)/1000),
    FPKM_RSEM = FPKM,
    TPM_RSEM = TPM,
    EffectiveLength_RSEM = effective_length
  ) %>%
  select (
    -gene_id, -length, -effective_length,
    -expected_count, -IsoPct, -TPM, -FPKM
  )

dat <- Salmon %>% merge(RSEM, by.x = 'Name', by.y = 'transcript_id')
```

```
library(reshape2)

get_upper_tri <- function(cormat){
    cormat[lower.tri(cormat)]<- NA
    return(cormat)
}

cormat <- dat %>%
  select( -Name ) %>%
  cor() %>%
  get_upper_tri() %>%
  melt(na.rm = TRUE)

# Create a ggheatmap
ggheatmap <- ggplot(cormat, aes(Var2, Var1, fill = value))+
 geom_tile(color = "white")+
 scale_fill_gradient2(low = "blue", high = "red", mid = "white",
   midpoint = 0, limit = c(-1,1), space = "Lab",
    name="Pearson\nCorrelation") +
```

```
  theme_minimal()+ # minimal theme
 theme(axis.text.x = element_text(angle = 45, vjust = 1,
    size = 12, hjust = 1))+
 coord_fixed()+
geom_text(aes(Var2, Var1, label = round(value,2)), color = "black", size = 3) +
theme(
  axis.title.x = element_blank(),
  axis.title.y = element_blank(),
  panel.grid.major = element_blank(),
  panel.border = element_blank(),
  panel.background = element_blank(),
  axis.ticks = element_blank(),
  legend.justification = c(1, 0),
  legend.position = c(0.6, 0.7),
  legend.direction = "horizontal")+
  guides(fill = guide_colorbar(barwidth = 7, barheight = 1,
                title.position = "top", title.hjust = 0.5))

# Print the heatmap
print(ggheatmap)
```



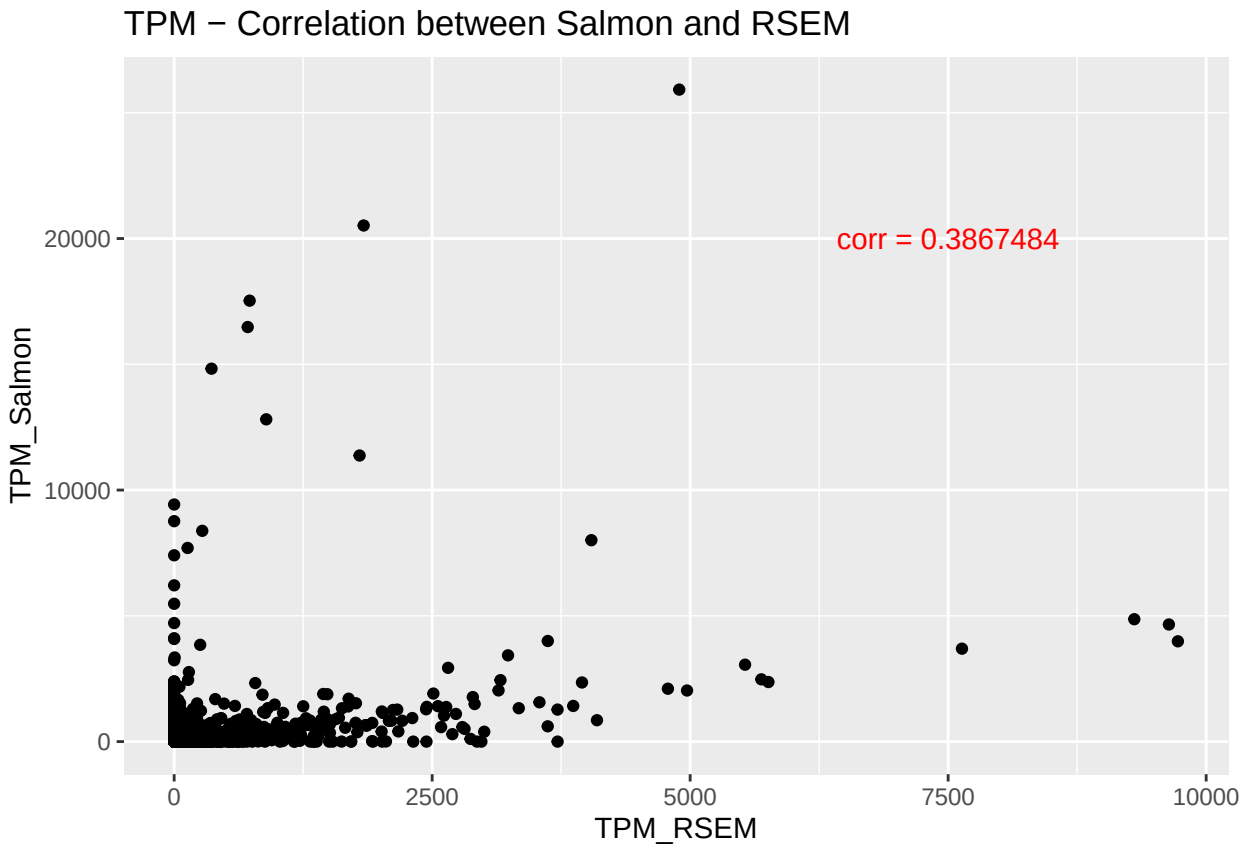
The heatmap shows that the effective length from the RSEM results is highly correlated with the effective length from the salmon results. TPM from Salmon is highly correlated with TPM from Salmon, and the same for RSEM results. Reads per million in RSEM results are also highly associated with FPKM and TPM from the RSEM results. Further, reads per million in the salmon results is moderately associated with reads per million, FPKM, and TPM of the RSEM results.

There is little to no correlation between effect length in the salmon results and any other metric besides

RSEM effective length. This makes sense because TPM and FPKM adjust for effective length.

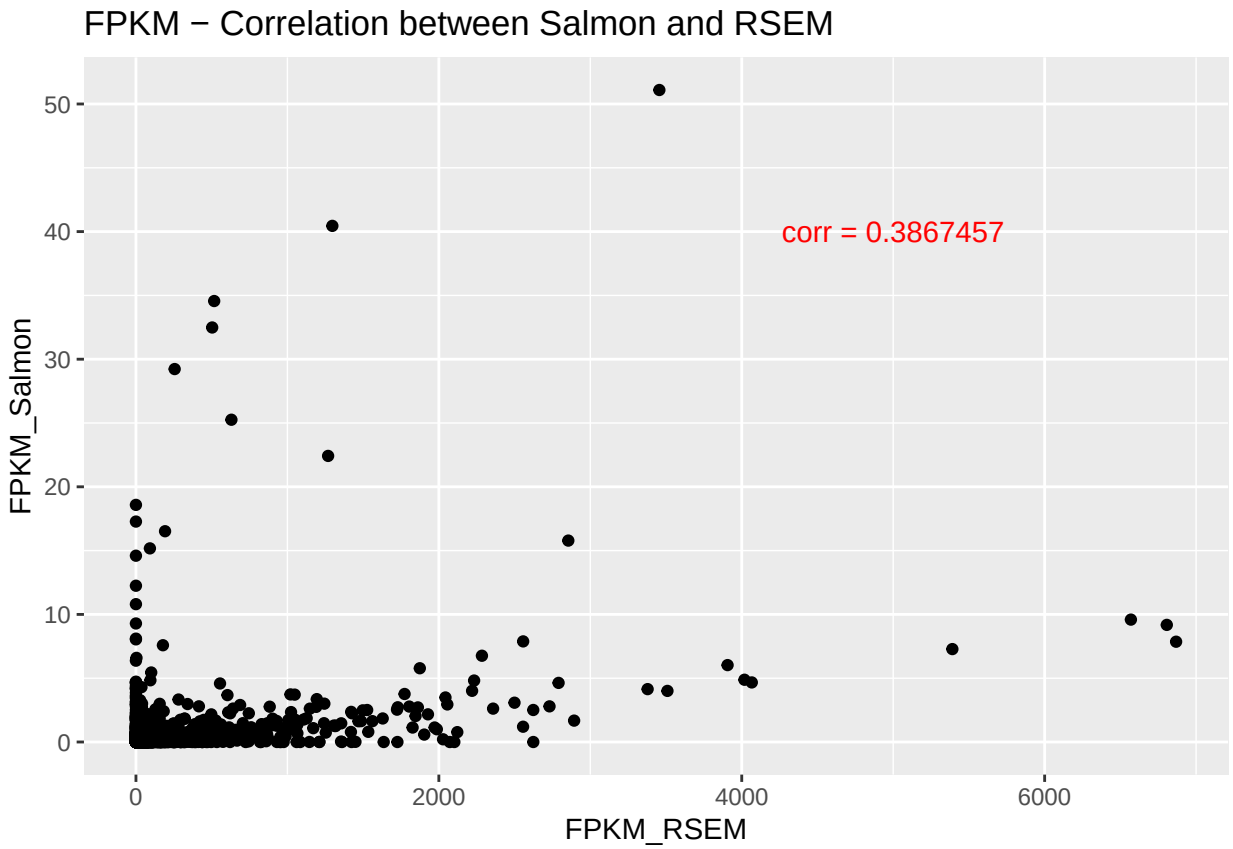
All correlations are positive.

```
dat %>%
  ggplot(aes(x = TPM_RSEM, y = TPM_Salmon)) +
  geom_point() +
  annotate(geom = 'text', x = 7500, y = 20000,
           label = paste("corr = 0.3867484"), color = 'red') +
  ggtitle("TPM - Correlation between Salmon and RSEM")
```



This plot shows that there is a weak positive correlation between TPM from the RSEM results and TPM from the Salmon results.

```
dat %>%
  ggplot(aes(x = FPKM_RSEM, y = FPKM_Salmon)) +
  geom_point() +
  annotate(geom = 'text', x = 5000, y = 40,
           label = paste("corr = 0.3867457"), color = 'red') +
  ggtitle("FPKM - Correlation between Salmon and RSEM")
```



This plot shows that there is a weak positive correlation between FPKM from the RSEM results and FPKM from the Salmon results.