Alveolar Macrophages after Murine Lung Transplant

180166459

2/8/2021

## R Markdown

# **BMS353**

# **Bioinformatics**

**Alveolar Macrophages after Murine Lung Transplant - (mouse)**

This is an R Markdown document. Markdown is a simple formatting syntax.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

**INTRODUCTION**

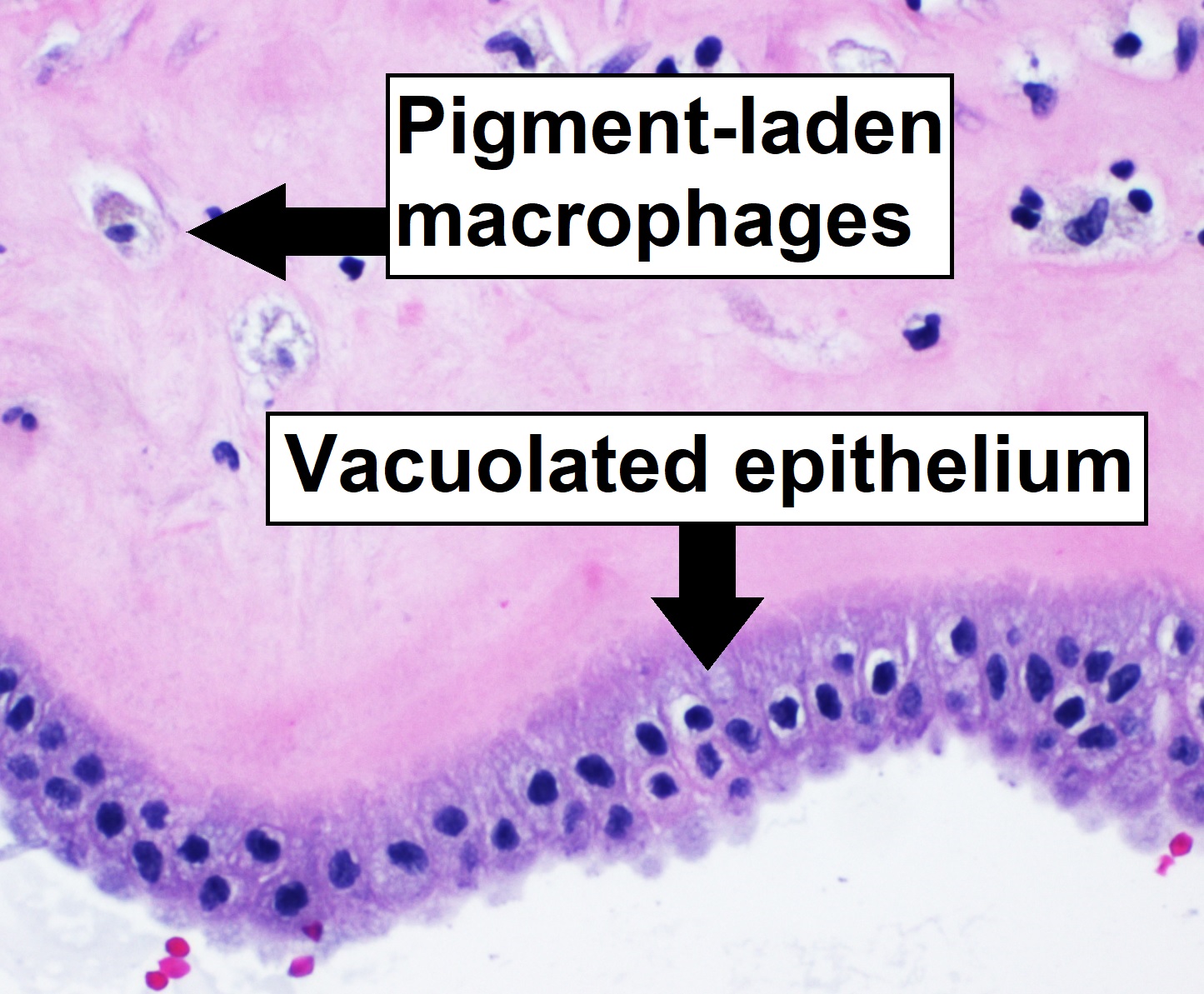


Figure 1 - Image that shows stained macrophage that is found in alveolar sac. Image is just informative but not necessarly relevant to this project

In this report we will analyze data for a research study that covers the following topic: Alveolar Macrophages after Murine Lung Transplant. For this we have the following resources provided:

*METHODS*

|  |  |
| --- | --- |
| Sample number | Condition |
| 4 | naive |
| 4 | 2h post-perfusion |
| 4 | 24 h post-perfusion |

[GEO](https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE116583)

[Google drive](https://drive.google.com/drive/folders/1AqYi0Ps5t5xo6XYWXjOtO6eDOMpT1ADE) - Google drive folder with the following content:

* meta\_data
* salmon-quant.sf files
* tx2gene.csv

The project name BMS353-Alveolar-Macrophages—RNA-seq-dataset with all files and the code are available on github in a repository:

[Github](https://github.com/IoanaAndra/BMS353-Alveolar-Macrophages---RNA-seq-dataset)

In order to use the necessary functions and analyze the data, we use some packages available in R:

* readr
* ggplot2
* dplyr

Every chunk of code can be name like this:

Installing all packages necessary for this project in R.

#SECTION FOR PACKAGES  
#installing all necessary packages  
# IF using same R version (4.0.3) and have all packages installed already, all lines of this section can remain commented   
   
# install.packages("readr")  
# install.packages("ggplot2") #\*

\*All code is visible in Rmd file.

CSV and TSV file formats can be stored in variables and display first row of them using head()

With head, we display first rows of data tx2gene.csv and a quant.sf

tx2gene.csv is the transcripts and genes relationship table file

library(readr)  
  
dataPath <- "tx2gene.csv" #asigning value which represent path of file  
  
file.exists(dataPath) #file.exists() return TRUE id file can be found or FALSE if it is not found

## [1] TRUE

tx2GeneFile <- read.csv(file = 'tx2gene.csv') #file with delimiter ","  
quantFile <- read\_tsv(file = 'salmon\_quant/SRR7457551/quant.sf') #file with delimiter "\t"  
head(tx2GeneFile)

## TXNAME GENEID  
## 1 ENSMUST00000193812 ENSMUSG00000102693  
## 2 ENSMUST00000082908 ENSMUSG00000064842  
## 3 ENSMUST00000192857 ENSMUSG00000102851  
## 4 ENSMUST00000161581 ENSMUSG00000089699  
## 5 ENSMUST00000192183 ENSMUSG00000103147  
## 6 ENSMUST00000193244 ENSMUSG00000102348

head(quantFile)

## # A tibble: 6 x 5  
## Name Length EffectiveLength TPM NumReads  
## <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 ENSMUST00000193812 1070 756. 0 0  
## 2 ENSMUST00000082908 110 4 0 0  
## 3 ENSMUST00000162897 4153 3719. 0 0  
## 4 ENSMUST00000159265 2989 2604. 0.0174 2  
## 5 ENSMUST00000070533 3634 3376. 0 0  
## 6 ENSMUST00000192857 480 230 0 0

#view(tx2GeneFile)

Name represents in quant. sf the TXNAME found in tx2gene.csv. But tx2gene.csv contains all 12 samples (in order …51-…62)

**METHODS**

**ANALYSIS**

Sample txt. presents the full experimental design that was followed. 12 columns for all 12 samples that are found in a salmon output format file that can be further read and accessed using read\_tsv

library(tximport)  
library(readr)  
library(dplyr)  
# Read the sample information into R  
sampleinfo <- read.delim("meta\_data/sampleInfo.txt") #reads in delimited text file  
View(sampleinfo) # view sampleinfo.txt in a new tab  
#glimpse(sampleinfo)  
sampleinfo

## geo\_accession name condition time run  
## 1 GSM3243460 N01\_AM\_Naive naive 0hr SRR7457557  
## 2 GSM3243461 N02\_AM\_Naive naive 0hr SRR7457558  
## 3 GSM3243462 N03\_AM\_Naive naive 0hr SRR7457559  
## 4 GSM3243463 N04\_AM\_Naive naive 0hr SRR7457560  
## 5 GSM3243464 R01\_AM\_Allo2h post\_reperfusion 2hr SRR7457553  
## 6 GSM3243465 R02\_AM\_Allo2h post\_reperfusion 2hr SRR7457554  
## 7 GSM3243466 R03\_AM\_Allo2h post\_reperfusion 2hr SRR7457555  
## 8 GSM3243467 R04\_AM\_Allo2h post\_reperfusion 2hr SRR7457556  
## 9 GSM3243468 R05\_AM\_Allo24h post\_reperfusion 24hr SRR7457551  
## 10 GSM3243469 R06\_AM\_Allo24h post\_reperfusion 24hr SRR7457552  
## 11 GSM3243470 R07\_AM\_Allo24h post\_reperfusion 24hr SRR7457561  
## 12 GSM3243471 R08\_AM\_Allo24h post\_reperfusion 24hr SRR7457562

rownames(sampleinfo) <- sampleinfo$run   
  
dirs <- list.files("salmon\_quant/")  
quant\_files <- list.files("salmon\_quant/",pattern="quant.sf",recursive = TRUE,full.names = TRUE)  
#names(quant\_files) <- dirs  
quant\_files

## [1] "salmon\_quant//SRR7457551/quant.sf" "salmon\_quant//SRR7457552/quant.sf"  
## [3] "salmon\_quant//SRR7457553/quant.sf" "salmon\_quant//SRR7457554/quant.sf"  
## [5] "salmon\_quant//SRR7457555/quant.sf" "salmon\_quant//SRR7457556/quant.sf"  
## [7] "salmon\_quant//SRR7457557/quant.sf" "salmon\_quant//SRR7457558/quant.sf"  
## [9] "salmon\_quant//SRR7457559/quant.sf" "salmon\_quant//SRR7457560/quant.sf"  
## [11] "salmon\_quant//SRR7457561/quant.sf" "salmon\_quant//SRR7457562/quant.sf"

# tx2gene links transcript IDs to gene IDs for summarization  
tx2geneVariable <- read.csv("tx2gene.csv")  
  
txi <- tximport(files=quant\_files, type="salmon", tx2gene=tx2geneVariable)  
  
View(txi) #view txi in a new tab  
  
#inspect salmon output (quant.sf files)  
quants <- read\_tsv(quant\_files[1])  
head(quants)

## # A tibble: 6 x 5  
## Name Length EffectiveLength TPM NumReads  
## <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 ENSMUST00000193812 1070 756. 0 0  
## 2 ENSMUST00000082908 110 4 0 0  
## 3 ENSMUST00000162897 4153 3719. 0 0  
## 4 ENSMUST00000159265 2989 2604. 0.0174 2  
## 5 ENSMUST00000070533 3634 3376. 0 0  
## 6 ENSMUST00000192857 480 230 0 0

# spec(quants)  
   
 filter(quants, quants$TPM ==0) #filter data where TPM = 0

## # A tibble: 60,226 x 5  
## Name Length EffectiveLength TPM NumReads  
## <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 ENSMUST00000193812 1070 756. 0 0  
## 2 ENSMUST00000082908 110 4 0 0  
## 3 ENSMUST00000162897 4153 3719. 0 0  
## 4 ENSMUST00000070533 3634 3376. 0 0  
## 5 ENSMUST00000192857 480 230 0 0  
## 6 ENSMUST00000195335 2819 250 0 0  
## 7 ENSMUST00000192336 2233 250 0 0  
## 8 ENSMUST00000194099 2309 250 0 0  
## 9 ENSMUST00000161581 250 20 0 0  
## 10 ENSMUST00000192973 2057 250 0 0  
## # ... with 60,216 more rows

#print.data.frame(quants) #print quants, maximum output to print in console 200 lines  
   
tail(quants, n = 3)

## # A tibble: 3 x 5  
## Name Length EffectiveLength TPM NumReads  
## <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 ENSMUST00000179077 887 762. 17.0 574.   
## 2 ENSMUST00000189352 548 295. 1.45 18.9  
## 3 ENSMUST00000178569 1083 250 0 0

max(quants$TPM)

## [1] 63095.94

After asigning to quants variable the files

rpk <- quants$NumReads / quants$EffectiveLength  
scale\_factor <- sum(rpk) / 1e6  
tpm <- rpk / scale\_factor  
  
#define transcript mapping  
  
gtf\_file <- "Mus\_musculus.GRCm38.91.chr.gtf.gz"  
file.exists(gtf\_file)

## [1] TRUE

download.file("ftp://ftp.ensembl.org/pub/release-91/gtf/mus\_musculus/Mus\_musculus.GRCm38.91.chr.gtf.gz",destfile = gtf\_file) #gtf based on organism of interest  
  
# create a database of transcripts  
# Could take a few minutes to run the makeTxDbFromGFF command  
library(GenomicFeatures)  
txdb <- makeTxDbFromGFF(gtf\_file)  
  
  
#specify number of keys and columns  
  
keytypes(txdb)

## [1] "CDSID" "CDSNAME" "EXONID" "EXONNAME" "GENEID" "TXID" "TXNAME"

columns(txdb)

## [1] "CDSCHROM" "CDSEND" "CDSID" "CDSNAME" "CDSPHASE"   
## [6] "CDSSTART" "CDSSTRAND" "EXONCHROM" "EXONEND" "EXONID"   
## [11] "EXONNAME" "EXONRANK" "EXONSTART" "EXONSTRAND" "GENEID"   
## [16] "TXCHROM" "TXEND" "TXID" "TXNAME" "TXSTART"   
## [21] "TXSTRAND" "TXTYPE"

#get names for all transcripts - using keys function  
#compose query - using select function - this will return data frame  
  
k <- keys(txdb, keytype="TXNAME")  
tx\_map <- select(txdb, keys = k, columns="GENEID", keytype = "TXNAME")  
  
#visualise first rows of the transcript map  
head(tx\_map)

## TXNAME GENEID  
## 1 ENSMUST00000193812 ENSMUSG00000102693  
## 2 ENSMUST00000082908 ENSMUSG00000064842  
## 3 ENSMUST00000192857 ENSMUSG00000102851  
## 4 ENSMUST00000161581 ENSMUSG00000089699  
## 5 ENSMUST00000192183 ENSMUSG00000103147  
## 6 ENSMUST00000193244 ENSMUSG00000102348

#use tximport package  
  
library(tximport)  
tx2gene <- tx\_map  
# write.csv(tx2gene,file="tx2gene.csv",row.names = FALSE,quote=FALSE)  
txi <- tximport(quant\_files,type="salmon",tx2gene = tx2gene)  
  
table(tx\_map$TXNAME %in% quants$Name)  
  
tx2gene <- tx\_map  
txi <- tximport(quant\_files,type="salmon",tx2gene = tx2gene,ignoreTxVersion = TRUE)  
  
names(txi)  
  
head(txi$counts)   
  
all(rownames(sampleinfo) == colnames(txi$counts))  
  
library(tidyr)  
library(dplyr)  
  
quants <- separate(quants, Name, c("TXNAME","Number"),remove = FALSE)  
head(quants)  
  
  
quants <- left\_join(quants, tx\_map, by="TXNAME")  
head(quants)  
  
tx2gene <- dplyr:::select(quants, Name, GENEID)  
head(tx2gene)  
  
any(is.na(tx2gene$GENEID))  
  
tx2gene <- filter(tx2gene, !is.na(GENEID))  
  
library(tximport)  
txi <- tximport(quant\_files,type="salmon",tx2gene = tx2gene)

CHECK OUTPUT HERE Summarising

library(readr)  
## library(dplyr)  
##   
## head(quants)  
## summarise(quants, min(quants$Length), max(quants$Length), min(quants$EffectiveLength),max(quants$EffectiveLength)) #summarises in a table values that we select from object  
##   
## write.table(colnames(quants), col.names=FALSE)

Next, testing quality assesment for the raw reads of gene transcripts that were imported with tximport.

## DataFrame with 12 rows and 5 columns  
## geo\_accession name condition time  
## <character> <character> <factor> <character>  
## SRR7457557 GSM3243460 N01\_AM\_Naive naive 0hr  
## SRR7457558 GSM3243461 N02\_AM\_Naive naive 0hr  
## SRR7457559 GSM3243462 N03\_AM\_Naive naive 0hr  
## SRR7457560 GSM3243463 N04\_AM\_Naive naive 0hr  
## SRR7457553 GSM3243464 R01\_AM\_Allo2h post\_reperfusion 2hr  
## ... ... ... ... ...  
## SRR7457556 GSM3243467 R04\_AM\_Allo2h post\_reperfusion 2hr  
## SRR7457551 GSM3243468 R05\_AM\_Allo24h post\_reperfusion 24hr  
## SRR7457552 GSM3243469 R06\_AM\_Allo24h post\_reperfusion 24hr  
## SRR7457561 GSM3243470 R07\_AM\_Allo24h post\_reperfusion 24hr  
## SRR7457562 GSM3243471 R08\_AM\_Allo24h post\_reperfusion 24hr  
## run  
## <character>  
## SRR7457557 SRR7457557  
## SRR7457558 SRR7457558  
## SRR7457559 SRR7457559  
## SRR7457560 SRR7457560  
## SRR7457553 SRR7457553  
## ... ...  
## SRR7457556 SRR7457556  
## SRR7457551 SRR7457551  
## SRR7457552 SRR7457552  
## SRR7457561 SRR7457561  
## SRR7457562 SRR7457562

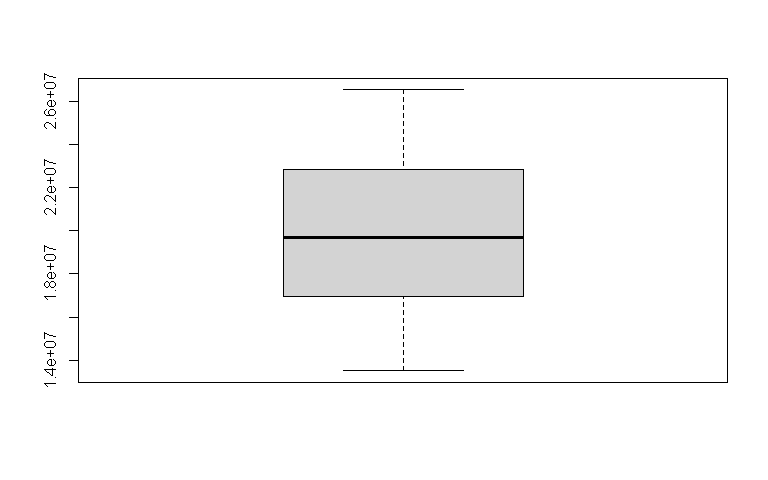
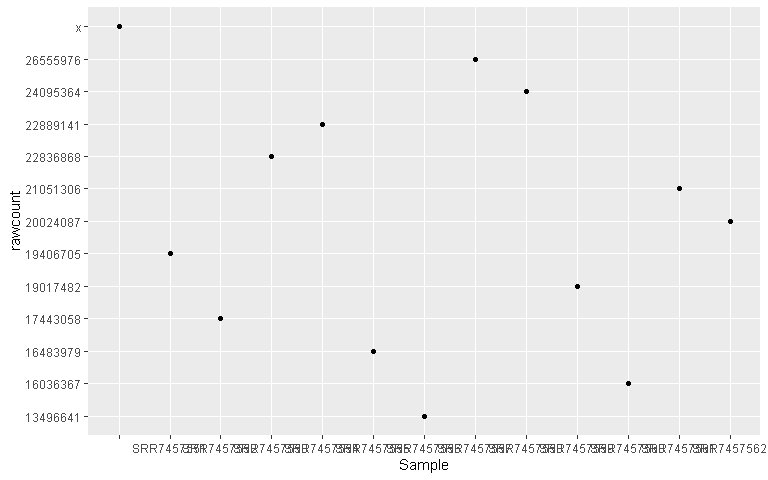
we can count the number of reads for each sample and print in the console. This can be written in a csv file.

## [1] "SRR7457557"  
## [1] 26555976  
##   
## [1] "SRR7457558"  
## [1] 24095364  
##   
## [1] "SRR7457559"  
## [1] 19017482  
##   
## [1] "SRR7457560"  
## [1] 16036367  
##   
## [1] "SRR7457553"  
## [1] 22836868  
##   
## [1] "SRR7457554"  
## [1] 22889141  
##   
## [1] "SRR7457555"  
## [1] 16483979  
##   
## [1] "SRR7457556"  
## [1] 13496641  
##   
## [1] "SRR7457551"  
## [1] 19406705  
##   
## [1] "SRR7457552"  
## [1] 17443058  
##   
## [1] "SRR7457561"  
## [1] 21051306  
##   
## [1] "SRR7457562"  
## [1] 20024087

colSums(assay(dds))

## SRR7457557 SRR7457558 SRR7457559 SRR7457560 SRR7457553 SRR7457554 SRR7457555   
## 26555976 24095364 19017482 16036367 22836868 22889141 16483979   
## SRR7457556 SRR7457551 SRR7457552 SRR7457561 SRR7457562   
## 13496641 19406705 17443058 21051306 20024087

This can be plotted into bar chart



## SRR7457557 SRR7457558 SRR7457559 SRR7457560 SRR7457553  
## ENSMUSG00000000001 TRUE TRUE TRUE TRUE TRUE  
## ENSMUSG00000000003 FALSE FALSE FALSE FALSE FALSE  
## ENSMUSG00000000028 TRUE TRUE TRUE TRUE TRUE  
## ENSMUSG00000000031 FALSE FALSE FALSE FALSE FALSE  
## ENSMUSG00000000037 TRUE TRUE FALSE TRUE TRUE  
## ENSMUSG00000000049 FALSE FALSE FALSE FALSE FALSE  
## SRR7457554 SRR7457555 SRR7457556 SRR7457551 SRR7457552  
## ENSMUSG00000000001 TRUE TRUE TRUE TRUE TRUE  
## ENSMUSG00000000003 FALSE FALSE FALSE FALSE FALSE  
## ENSMUSG00000000028 TRUE TRUE TRUE TRUE TRUE  
## ENSMUSG00000000031 FALSE FALSE FALSE FALSE FALSE  
## ENSMUSG00000000037 FALSE TRUE FALSE FALSE FALSE  
## ENSMUSG00000000049 FALSE FALSE FALSE FALSE FALSE  
## SRR7457561 SRR7457562  
## ENSMUSG00000000001 TRUE TRUE  
## ENSMUSG00000000003 FALSE FALSE  
## ENSMUSG00000000028 TRUE TRUE  
## ENSMUSG00000000031 FALSE FALSE  
## ENSMUSG00000000037 TRUE TRUE  
## ENSMUSG00000000049 FALSE FALSE

library(dplyr)  
library(ggplot2)  
sampleinfo %>% mutate(MillionsOfReads = colSums(assay(dds))/1e6) %>%   
 ggplot(aes(x = sampleinfo$run, y = MillionsOfReads)) + geom\_col(fill="darkgoldenrod2") + ggtitle ("Number of milliions of reads for each sample")

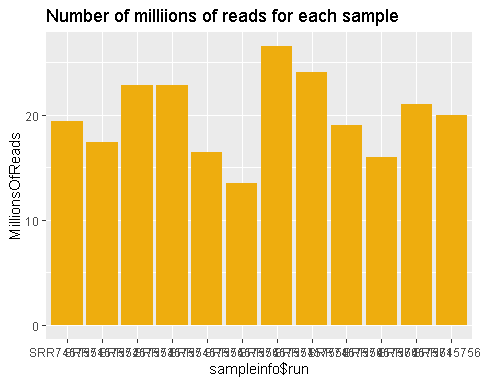
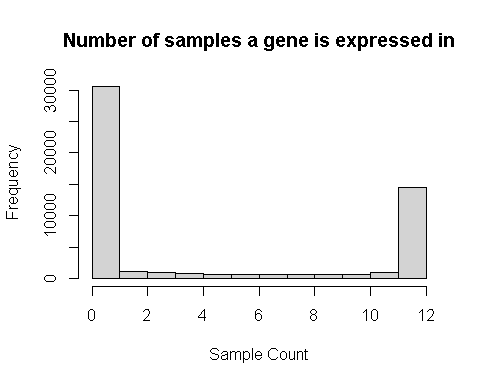
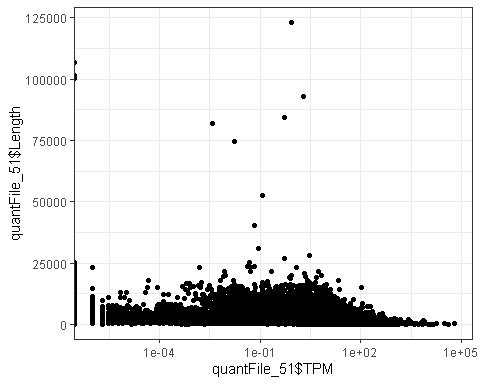


Figure - Bar charts that show number of millions of reads for each of the 12 samples

print('Gene from row 3 expressed in this number of samples:')  
sum(is\_expressed[3,])  
  
print('Gene from row 400 expressed in this number of samples:')  
sum(is\_expressed[400,])

hist(rowSums(is\_expressed),main="Number of samples a gene is expressed in",xlab="Sample Count")





**CONCLUSIONS**

*References*