Assignment 3

Daniel Lee

Question 1. Short read alignment.

To answer this question, I used the following steps:

1. Download GSM1650456 SRA file.
2. Convert it to FASTQ file.
3. Used TopHat2 to align the read using a pre-built index from `ftp://ftp-mouse.sanger.ac.uk/ref/GRCm38_68.fa`.
4. Use `featureCounts` from package `Rsubread` to summarize alignment, yielding gene-level read counts. The annotation file was obtained from `ftp://ftp.ensembl.org/pub/release-88/gtf/mus_musculus`. I downloaded the file `Mus_musculus.GRCm38.88.gtf`.
5. The code and output is shown below.

```
library(Rsubread)

summary.alignment <- featureCounts(files="accepted_hits.bam",
                                   annot.ext="Mus_musculus.GRCm38.88.gtf",
                                   isGTFAnnotationFile=TRUE,
                                   GTF.featureType="exon",
                                   GTF.attrType="gene_id")
```

```
##
##          ==========     _____ _    _ ____  _____  ______          _____
##          =====         / ____| |  | |  _ \|  __ \|  ____|   /\   |  __ \
##            =====      | (___ | |  | | |_) | |__) | |__     /  \  | |  | |
##              ====      \___ \| |  | |  _ <|  _  /|  __|   / /\ \ | |  | |
##                ====    ____) | |__| | |_) | | \ \| |____ / ____ \| |__| |
##          ==========   |_____/ \____/|____/|_|  \_\______/_/    \_\_____/
##         Rsubread 1.26.0
##
## //========================== featureCounts setting ===========================\\
## ||                                                                            ||
## ||             Input files : 1 BAM file                                       ||
## ||                           S accepted_hits.bam                              ||
## ||                                                                            ||
## ||      Dir for temp files : .                                                ||
## ||                 Threads : 1                                                ||
## ||                   Level : meta-feature level                               ||
## ||               Paired-end : no                                              ||
## ||          Strand specific : no                                              ||
## ||      Multimapping reads : not counted                                      ||
## || Multi-overlapping reads : not counted                                      ||
## ||   Min overlapping bases : 1                                                ||
## ||                                                                            ||
## \\====================== http://subread.sourceforge.net/ ======================//
##
## //================================= Running ==================================\\
## ||                                                                            ||
## || Load annotation file Mus_musculus.GRCm38.88.gtf ...                        ||
## ||    Features : 831                                                          ||
```

```
## ||    Meta-features : 82                                                    ||
## ||    Chromosomes/contigs : 1                                               ||
## ||                                                                          ||
## || Process BAM file accepted_hits.bam...                                    ||
## ||    Single-end reads are included.                                        ||
## ||    Assign reads to features...                                           ||
## ||    Total reads : 2552                                                    ||
## ||    Successfully assigned reads : 364 (14.3%)                             ||
## ||    Running time : 0.00 minutes                                           ||
## ||                                                                          ||
## ||                         Read assignment finished.                        ||
## ||                                                                          ||
## \\====================== http://subread.sourceforge.net/ ======================//
```

```
summary.alignment$counts
```

```
##                    accepted_hits.bam
## ENSMUSG00000102693                 0
## ENSMUSG00000064842                 0
## ENSMUSG00000051951                 0
## ENSMUSG00000102851                 0
## ENSMUSG00000103377                 0
## ENSMUSG00000104017                 0
## ENSMUSG00000103025                 0
## ENSMUSG00000089699                 0
## ENSMUSG00000103201                 0
## ENSMUSG00000103147                 0
## ENSMUSG00000103161                 0
## ENSMUSG00000102331                 0
## ENSMUSG00000102348                 0
## ENSMUSG00000102592                 0
## ENSMUSG00000088333                 0
## ENSMUSG00000102343                 0
## ENSMUSG00000025900                 0
## ENSMUSG00000102948                 0
## ENSMUSG00000104123                 0
## ENSMUSG00000025902                 0
## ENSMUSG00000104238                 0
## ENSMUSG00000102269                 0
## ENSMUSG00000096126                 0
## ENSMUSG00000103003                 0
## ENSMUSG00000104328                 0
## ENSMUSG00000102735                 0
## ENSMUSG00000098104                40
## ENSMUSG00000102175                 0
## ENSMUSG00000088000                 0
## ENSMUSG00000103265                 0
## ENSMUSG00000103922                 0
## ENSMUSG00000033845                31
## ENSMUSG00000102275                19
## ENSMUSG00000025903                 5
## ENSMUSG00000104217                 0
## ENSMUSG00000033813                 2
## ENSMUSG00000062588                 0
## ENSMUSG00000103280                 0
```

```
## ENSMUSG00000002459               2
## ENSMUSG00000091305               0
## ENSMUSG00000102653               0
## ENSMUSG00000085623               0
## ENSMUSG00000091665               0
## ENSMUSG00000033793               0
## ENSMUSG00000104352               0
## ENSMUSG00000104046               0
## ENSMUSG00000102907               0
## ENSMUSG00000025905               0
## ENSMUSG00000103936               0
## ENSMUSG00000093015               0
## ENSMUSG00000103519               0
## ENSMUSG00000033774               0
## ENSMUSG00000103090               0
## ENSMUSG00000025907               0
## ENSMUSG00000090031               0
## ENSMUSG00000087247               0
## ENSMUSG00000103355               0
## ENSMUSG00000102706               0
## ENSMUSG00000103845               0
## ENSMUSG00000033740               0
## ENSMUSG00000103329               0
## ENSMUSG00000104385               0
## ENSMUSG00000102135             265
## ENSMUSG00000103282               0
## ENSMUSG00000102534               0
## ENSMUSG00000102213               0
## ENSMUSG00000103629               0
## ENSMUSG00000051285               0
## ENSMUSG00000098201               0
## ENSMUSG00000103509               0
## ENSMUSG00000048538               0
## ENSMUSG00000103709               0
## ENSMUSG00000077244               0
## ENSMUSG00000102768               0
## ENSMUSG00000097797               0
## ENSMUSG00000103498               0
## ENSMUSG00000103067               0
## ENSMUSG00000102320               0
## ENSMUSG00000104226               0
## ENSMUSG00000103903               0
## ENSMUSG00000103557               0
## ENSMUSG00000025909               0
```

Based on the summary of counts, I see that there were 265 mapped reads for gene `ENSMUSG00000102135`.

Question 2. Exploratory data analysis

a. Examine the variables in the table `meta`. Specifically, how are they related to each other? To the genel-level counts data? Is there evidence of technical confounding?

The variables are related to each other because each variable describes the cell sample used. The variables provide information such as describing the experiment number the cells were part of, who prepared them, how many unique genes were found, how many ERCC spike-in control RNA sequences were read, and the RNA ChIP serial number. The variables in the table `meta` are related to the gene-level counts data because the gene-level counts data contain gene-level counts from the cells that are listed in the metadata. Full list of the variables in `meta` is provided below.

There is evidence of technical confounding especially when I examine the total counts for different experiment numbers and ChIP serial numbers.

The main biological effect in this experimental design was the distinction between IFE/infundibulum vs upper and lower HF. Sca1-microbeads were used to select between these two main types of cells. Sca1+ cells are the IFE/infundibulum cells, while Sca1- cells are the HF cells. This information is included in the metadata column named `cell_fraction`.

```
library(stringr)
library(plyr)
library(dplyr)
require(RColorBrewer)
require(ggplot2)
require(reshape)
require(matrixStats)
library(plyr)
source("multiplot.R")
```

```
set1 <- brewer.pal(9, "Set1")
set2 <- c("#ced64a","#6042bb","#70d454","#c14bcb","#83d7a3",
          "#c7407f","#5b8842","#d04338","#76c4d1","#c67a3b",
          "#6b84cd","#c4ad5d","#492b5b","#c8bbae","#642f2b",
          "#c58dc8","#404d2d","#c1777c","#546d7d")
set3 <- c("#3790ff","#f2d34b","#34056c","#b3e460","#58007b",
          "#5ebd49","#6a32a3","#6ab336","#3b4fc2","#99a400",
          "#4175ed","#fdc341","#835ed5","#37b34b","#910b86",
          "#2f9a2b","#c16be0","#357e00","#ea77e6","#009e45",
          "#9e38a8","#91e983","#5f0067","#8de991","#b7329e",
          "#01ba6d","#db3d9a","#01d798","#bb005a","#009c52",
          "#ff73cd","#016a1a","#d38bff","#989700","#025cbe",
          "#c39f03","#7d93ff","#bce170","#3b0b5b","#80e9ae",
          "#710065","#ead473","#003a87","#c37900","#0184dc",
          "#d77119","#009fee","#c24418","#01bfff","#bd331d",
          "#00ccf8","#b90033","#007837","#fa4c7c","#004509",
          "#ffa5fc","#2c5e00","#a99eff","#717c00","#93a3ff",
          "#517000","#dea8ff","#505400","#f3b7ff","#8b6a00",
          "#005aaa","#ffaf59","#002869","#ffc675","#2f1856",
          "#ffbe7e","#0080c9","#a45800","#345a9b","#d13b34",
          "#497030","#8a0061","#784f00","#ff98da","#8f4e00",
          "#473e7d","#ff8866","#632d65","#ff6f66","#620041",
          "#ff9183","#5e0036","#ff939a","#570010","#ff96bf",
          "#660700","#ff71a0","#862200","#b3689b","#850013",
```

```
          "#9d0050","#92522d","#df315b","#822c36","#7b0026")
```

```
setwd("C:/Users/Daniel/Desktop/Spring 2017/PH 240F/Assignment 3")

#Read in the tables

meta <- read.table("https://www.stat.berkeley.edu/~sandrine/Teaching/PH240F.S17/Assignments/HW3/meta.tx
                   sep = "\t",
                   header = TRUE,
                   stringsAsFactors = FALSE,
                   row.names = 1)

geneLevelCounts <- read.table("GSE67602_Joost_et_al_expression.txt",
                          stringsAsFactors = FALSE,
                          header = TRUE,
                          row.names = 1)

colnames(geneLevelCounts) <- gsub("_", ".", colnames(geneLevelCounts))

#subset data such that metadata only contains the samples used in the data of gene level counts
meta <- meta[ , which(colnames(meta)%in% colnames(geneLevelCounts))]

#reorder data frame according to column names
meta <- meta[, order(colnames(meta))]

geneLevelCounts <- geneLevelCounts[, order(colnames(geneLevelCounts))]
dim(geneLevelCounts)
```

```
## [1] 26024  1422
```

```
dim(meta)
```

```
## [1]   35 1422
```

```
#Check to see if all names match
sum(colnames(meta) == colnames(geneLevelCounts))
```

```
## [1] 1422
```

```
samplecells <- colnames(meta)

colnames.meta <- colnames(meta)

#transpose the metadata table
meta.transpose <- as.data.frame(t(meta))
colnames(meta.transpose) <- rownames(meta)
```

The table `meta` includes the following variables:

Chip serial number, Chip type, Date of Run, Species, Tissue/cell type/source, Principal Investigator, Scientist, Operator, Protocol, Comments, experiment_number, mouse_number, strain, gender, breeder, date_of_birth, age, weight, time_of_death, date, attached_chip_number, tissue, cell_fraction, hair_cycle_stage, comments, area(um^2), diameter(um), red_flag, green_flag, blue_flag, volume(um^3), sum_spikes, sum_repeats, sum_transcripts, sum_genes

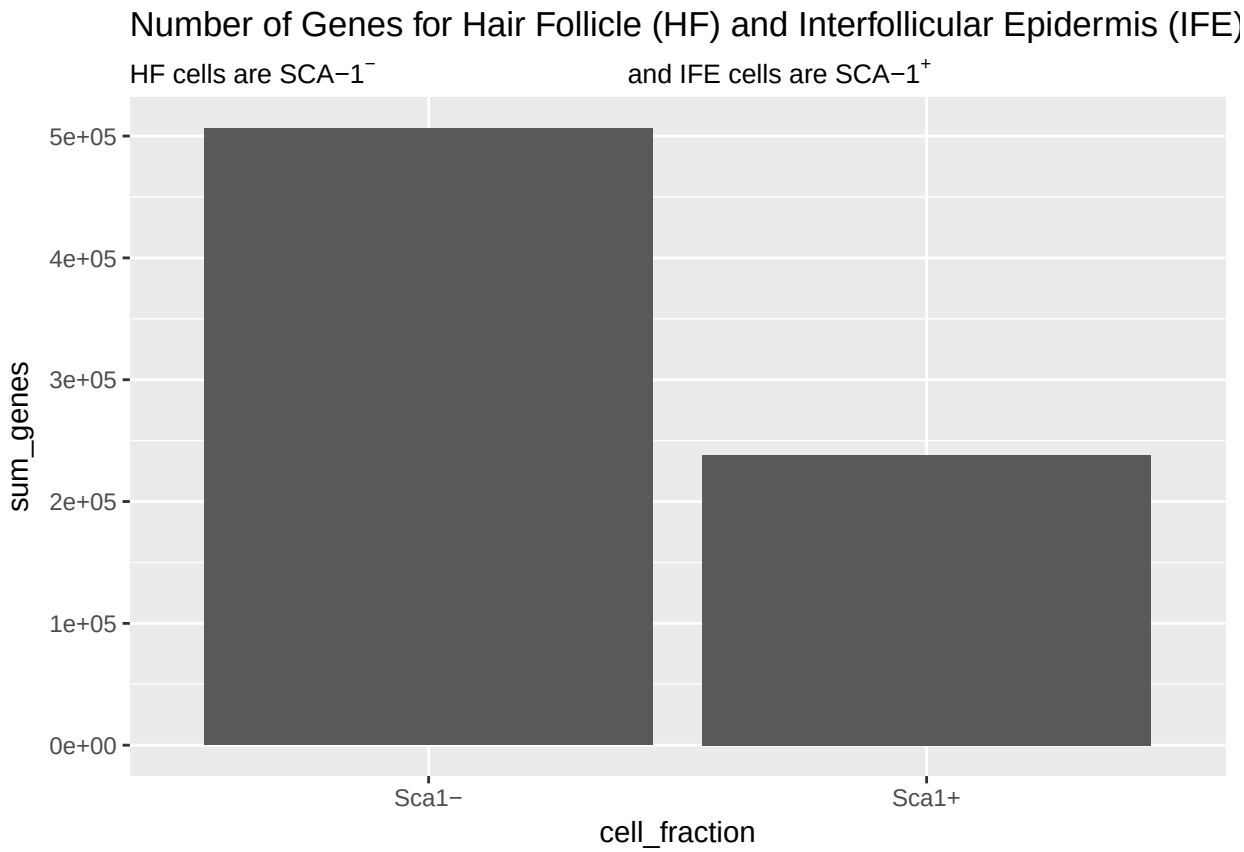
```
num <- colwise(as.numeric)
meta.transpose[,c(26, 27, 31:35)] <- num(meta.transpose[c(26, 27, 31:35)])
```

```r
meta.transpose <- data.frame(meta.transpose, cell.samples = colnames.meta)
```

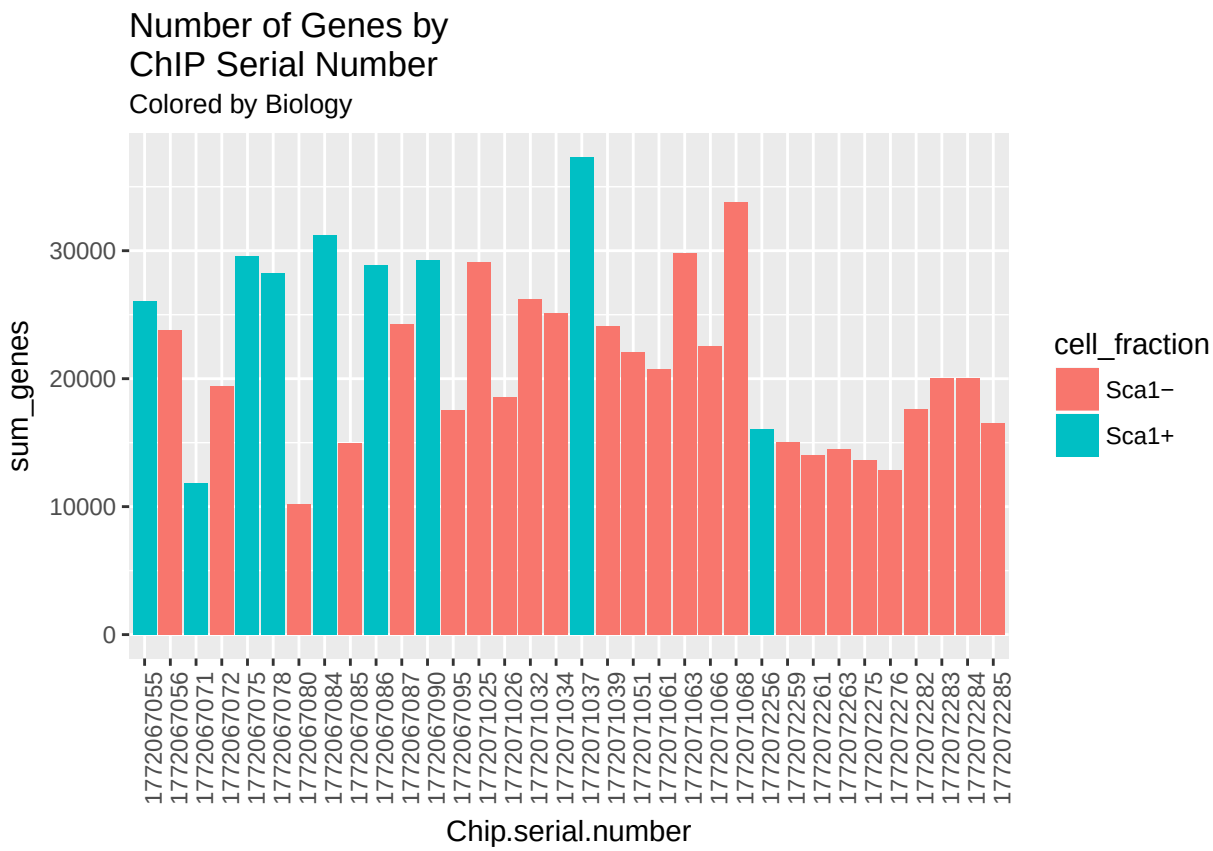
```r
#number of transcripts by biology
ggplot(meta.transpose, aes(x = cell_fraction, y = sum_genes)) +
  geom_bar(stat = "identity") +
  labs(title = "Number of Genes for Hair Follicle (HF) and Interfollicular Epidermis (IFE) Cells",
       subtitle = expression(paste("HF cells are SCA-1"^"-","
                                   and IFE cells are SCA-1"^"+")))
```

```
## Warning in grid.Call(L_stringMetric, as.graphicsAnnot(x$label)): font
## metrics unknown for character 0xa

## Warning in grid.Call(L_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## font metrics unknown for character 0xa

## Warning in grid.Call(L_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## font metrics unknown for character 0xa

## Warning in grid.Call(L_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## font metrics unknown for character 0xa

## Warning in grid.Call(L_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## font metrics unknown for character 0xa

## Warning in grid.Call(L_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## font metrics unknown for character 0xa

## Warning in grid.Call(L_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## font metrics unknown for character 0xa

## Warning in grid.Call(L_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## font metrics unknown for character 0xa

## Warning in grid.Call(L_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## font metrics unknown for character 0xa

## Warning in grid.Call(L_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## font metrics unknown for character 0xa

## Warning in grid.Call(L_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## font metrics unknown for character 0xa

## Warning in grid.Call(L_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## font metrics unknown for character 0xa

## Warning in grid.Call(L_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## font metrics unknown for character 0xa

## Warning in grid.Call(L_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## font metrics unknown for character 0xa

## Warning in grid.Call(L_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## font metrics unknown for character 0xa

## Warning in grid.Call.graphics(L_text, as.graphicsAnnot(x$label), x$x, x
```

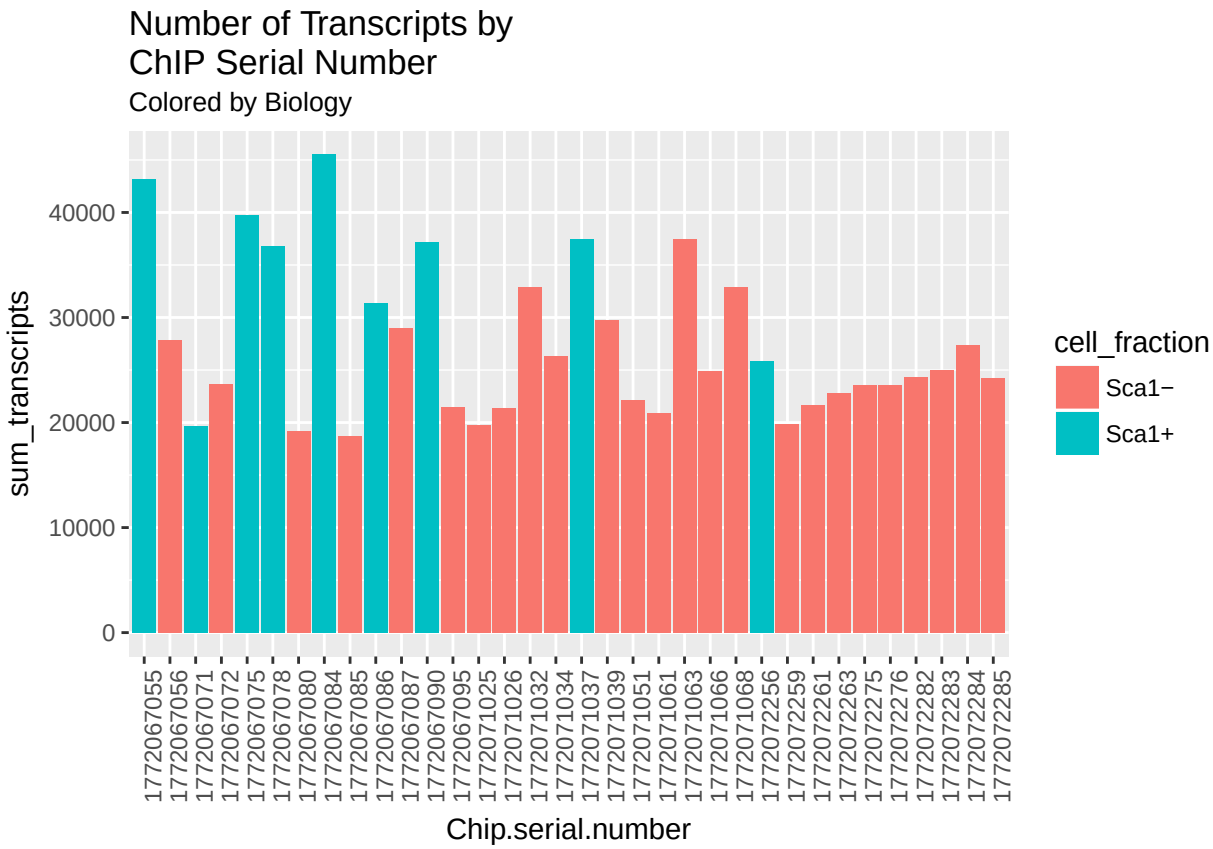
```
## $y, : font metrics unknown for character 0xa

## Warning in grid.Call.graphics(L_text, as.graphicsAnnot(x$label), x$x, x
## $y, : font metrics unknown for character 0xa
```




```
#number of transripts by serial number, colored by biology
p1 <- ggplot(meta.transpose, aes(x = Chip.serial.number, y = sum_genes, fill = cell_fraction)) +
  geom_bar(stat = "identity") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  labs(title = "Number of Genes by \nChIP Serial Number",
       subtitle = "Colored by Biology")
p1
```



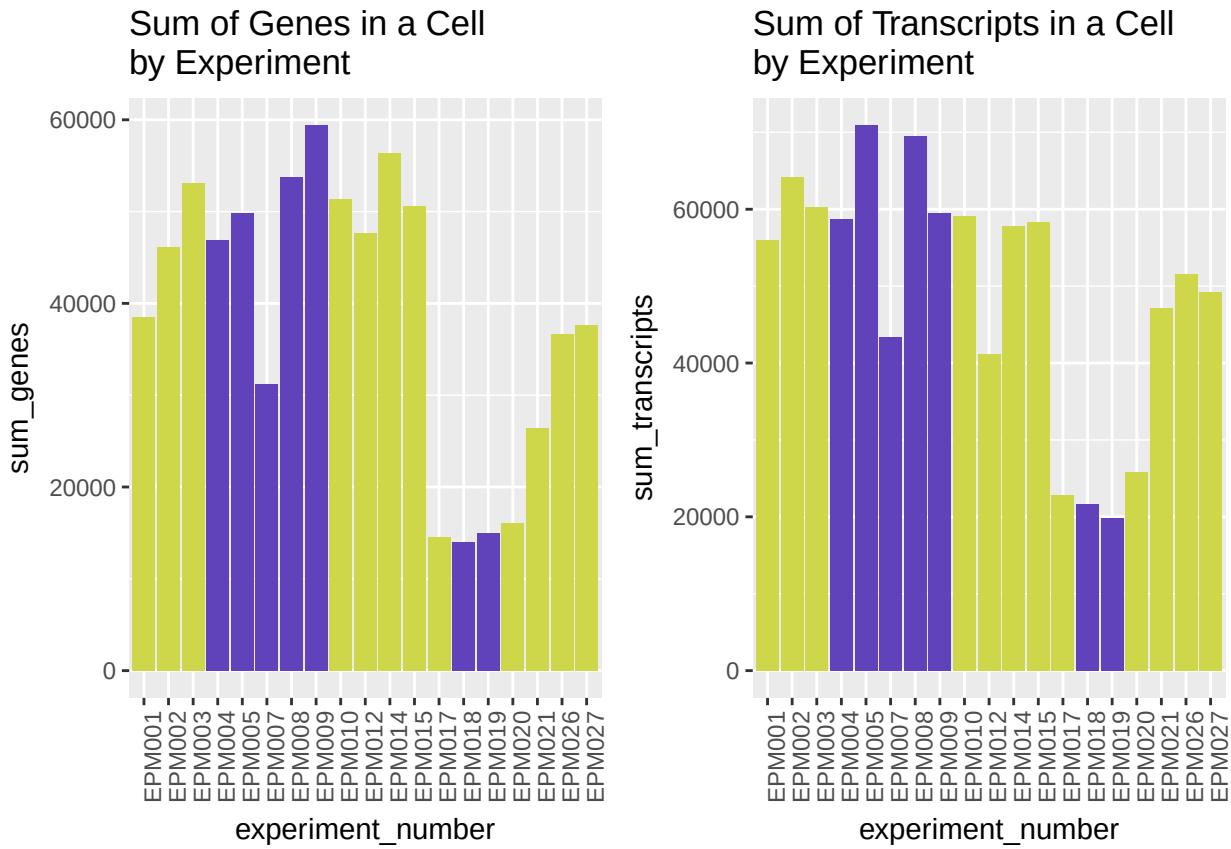

```
p2 <- ggplot(meta.transpose, aes(x = Chip.serial.number, y = sum_transcripts, fill = cell_fraction)) +
  geom_bar(stat = "identity") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  labs(title = "Number of Transcripts by \nChIP Serial Number",
       subtitle = "Colored by Biology")
p2
```



Here, we see confounding as certain ChIPs had fewer reads than other ChIPs. The coloring by cell fraction is inconclusive.

```
p1 <- ggplot(meta.transpose, aes(x = experiment_number, y = sum_genes, fill = breeder)) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = set2) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1),
        legend.position="none") +
  labs(title = "Sum of Genes in a Cell \nby Experiment")

p2 <- ggplot(meta.transpose, aes(x = experiment_number, y = sum_transcripts, fill = breeder)) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = set2) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1),
        legend.position="none") +
  labs(title = "Sum of Transcripts in a Cell \nby Experiment")

multiplot(p1, p2, cols = 2)
```

```
## Loading required package: grid
```

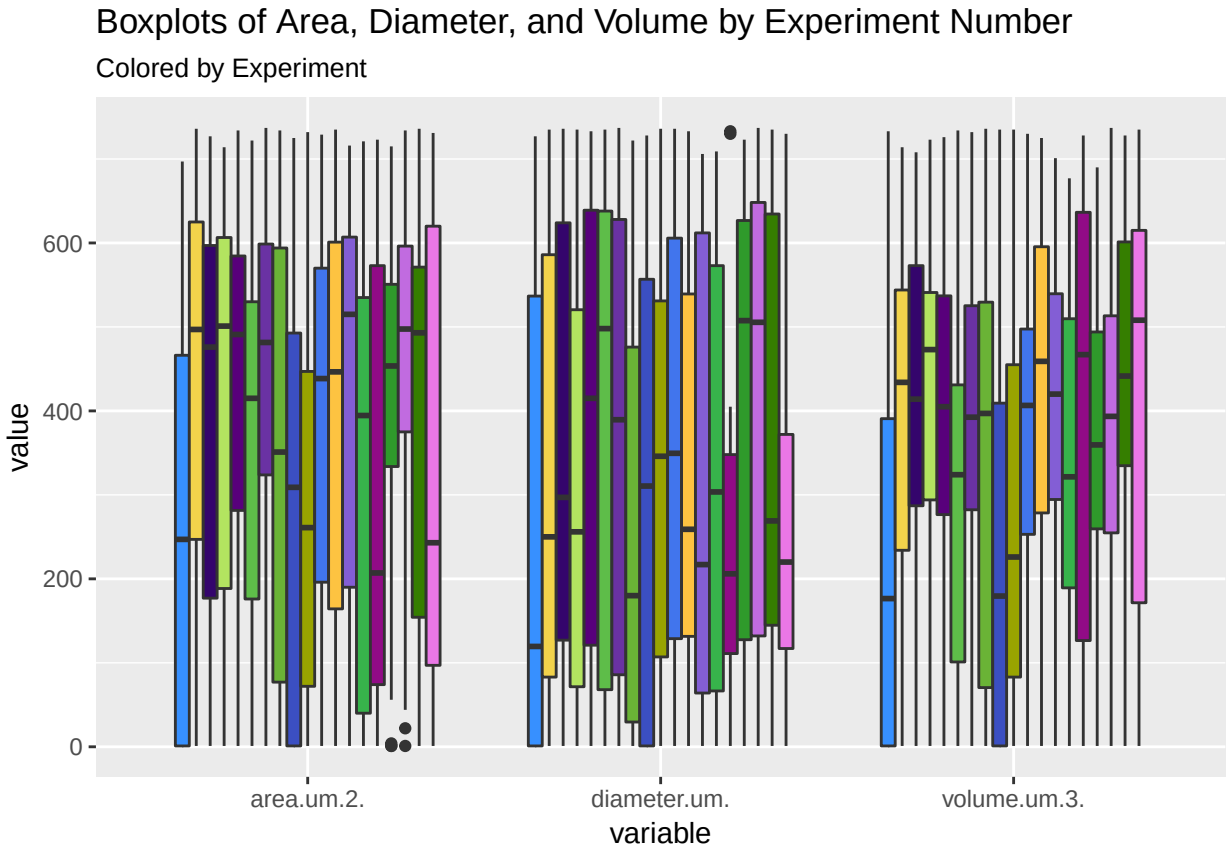


Here is another evidence of technical confounding because different experiments have different number of total genes and transcripts read. The coloring by breeder is inconclusive.

```
sub.meta <- data.frame(experiment_number = meta.transpose$experiment_number,
           sum_genes = meta.transpose$sum_genes,
           area.um.2. = meta.transpose$area.um.2.,
           diameter.um. = meta.transpose$diameter.um.,
           volume.um.3. = meta.transpose$volume.um.3.)

melt.sub.meta <- melt(sub.meta, id.vars = colnames(sub.meta)[1:2])

ggplot(data = melt.sub.meta, mapping = aes(x = value, y = sum_genes, col = experiment_number)) +
  geom_point() +
  facet_wrap( ~ variable) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1),
        legend.position="none") +
  labs(title = "Scatterplot of Total Gene Reads vs Area,
       Diameter, and Volume\nColord by Experiment Number")
```

The scatterplots above show that there seems to be random distribution of sizes of cells for the experiments.

```
ggplot(data = melt.sub.meta, mapping = aes(x = variable, y = value, fill = experiment_number)) +
  geom_boxplot() +
  scale_fill_manual(values = set3) +
  labs(title = "Boxplots of Area, Diameter, and Volume by Experiment Number",
       subtitle = "Colored by Experiment") +
  theme(legend.position="none")
```

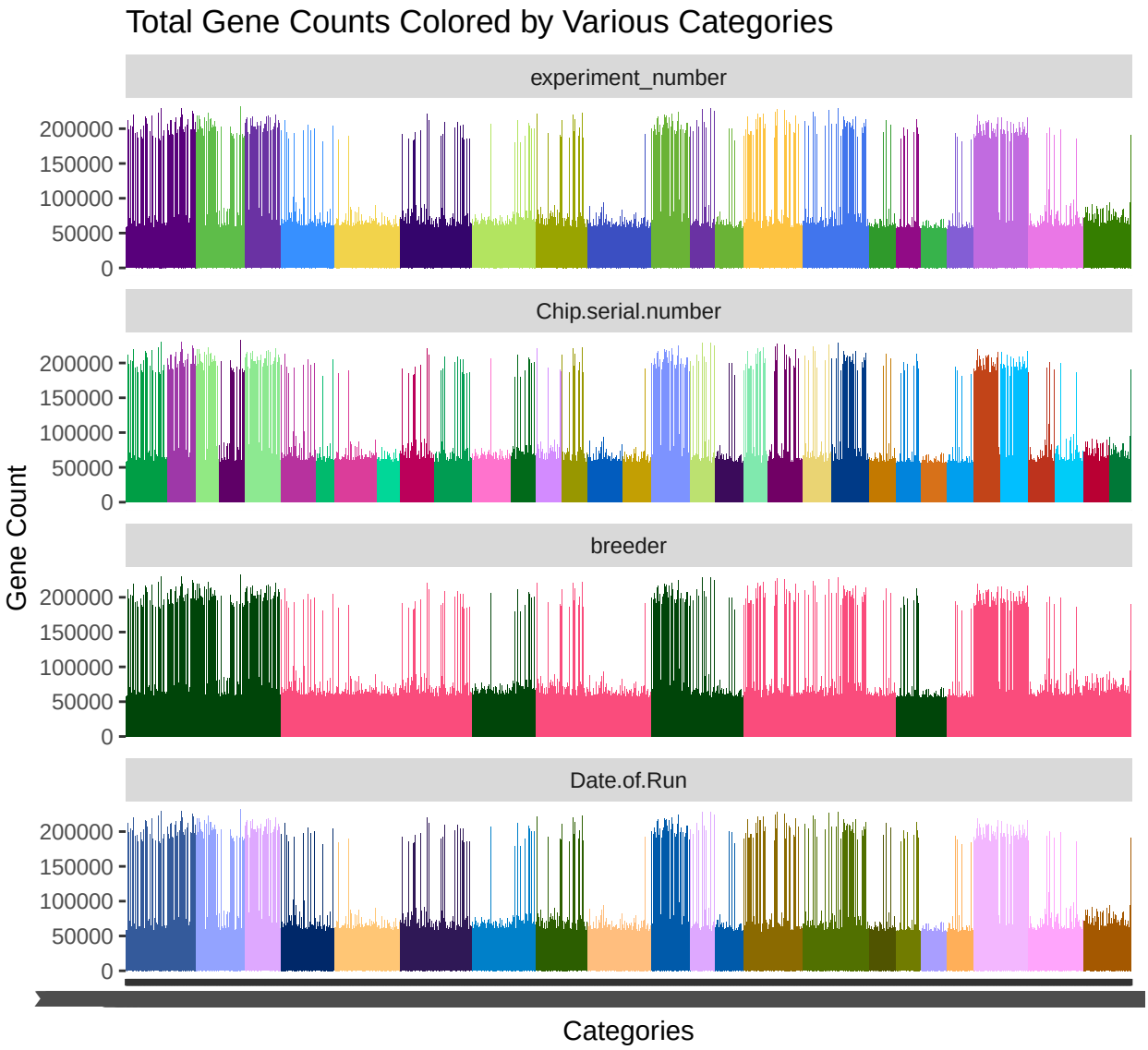


The boxplots above show that there seems to be some varying distributions of cell dimensions across experiments. However, the distributions seem to be randomly assigned.

```
# transpose the dataset
geneLevelCountsTranspose <- t(geneLevelCounts)
geneLevelCountsTranspose <- as.data.frame(geneLevelCountsTranspose)
colnames(geneLevelCountsTranspose) <- colnames.transpose
geneLevelCountsTranspose <- geneLevelCountsTranspose[-1,]

# make columns into numeric
num <- colwise(as.numeric)
geneLevelCountsTranspose[1:ncol(geneLevelCountsTranspose)] <-
  num(geneLevelCountsTranspose[1:ncol(geneLevelCountsTranspose)])
```

```
total.count <- rowSums(geneLevelCountsTranspose)

mod.meta <- data.frame(experiment_number = meta.transpose$experiment_number,
                       Chip.serial.number = meta.transpose$Chip.serial.number,
                       breeder = meta.transpose$breeder,
                       Date.of.Run = meta.transpose$Date.of.Run,
                       total = total.count,
                       cell.samples = meta.transpose$cell.samples)

melt.mod.meta <- melt(mod.meta, id.vars = c("cell.samples", "total"))

ggplot(melt.mod.meta, mapping = aes(x = cell.samples,
                                    y = total,
```

```
                                    fill = value)) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = set3) +
  facet_wrap( ~variable, ncol = 1) +
  labs(title = "Total Gene Counts Colored by Various Categories",
       x = "Categories",
       y = "Gene Count") +
  theme(legend.position="none")
```



The figures show that there's confounding due to technical effects because certain experiments, ChIP serial number, breeder, and date of runs show different number of total gene counts.

2.b. Examine the distributions of read counts originating from ERCC spike-in control sequences. Do these features show the desired properties of negative controls? How might you use them for normalization purposes?

The analysis below reveals that the ERCC spike-in negative controls do not have the desired properties of negative controls. In theory, I might use these negative controls for normalization purposes because these negative control genes should not be affected by technical effects and biological effects. The gene counts are expected to be constant for each gene. Using these as references, I can normalize the rest of the data to eliminate the technical effects.

```
spike.count <- read.table("spike_count.txt", header = FALSE, sep = "\t")
dim(spike.count)
```

```
## [1] 92  2
```

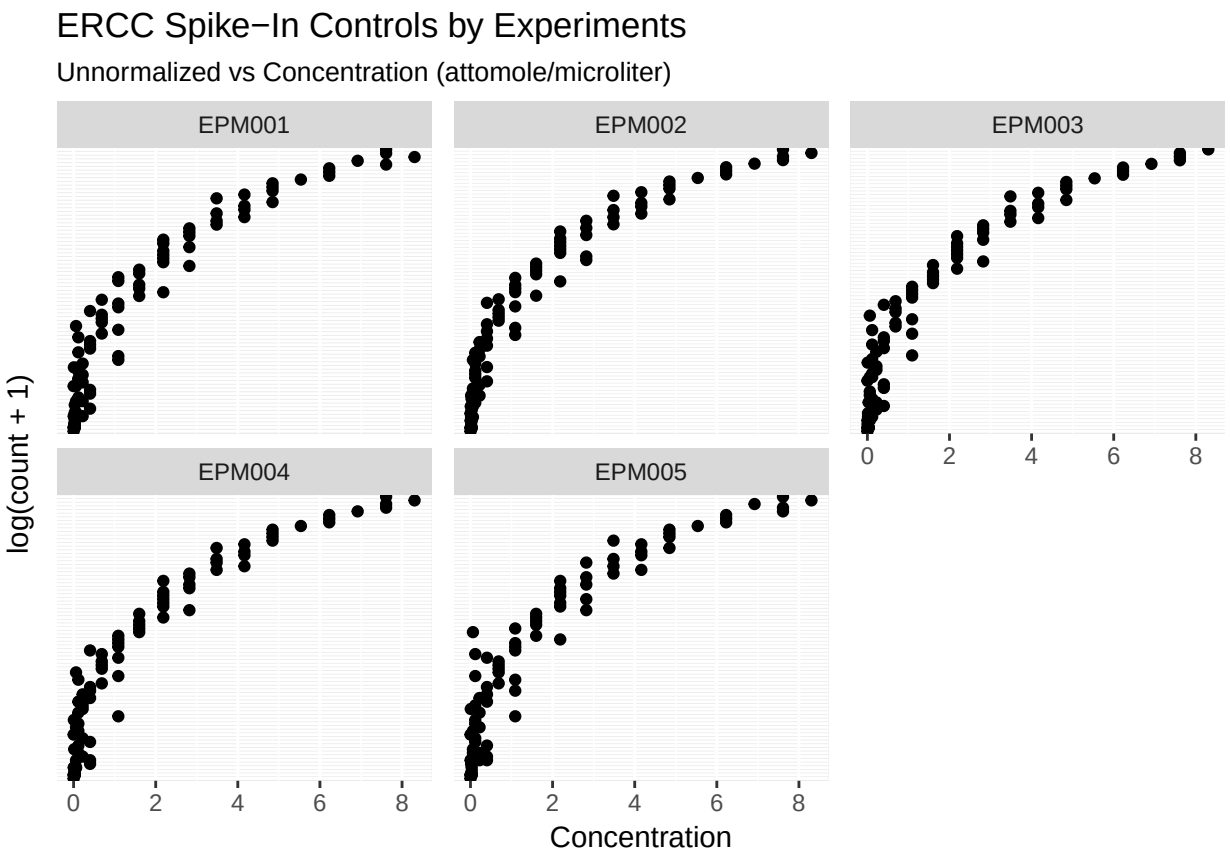
```
spike.count$V2 <- log(spike.count$V2 + 1)

ercc <- geneLevelCountsTranspose[,1:92]
log.ercc <- log(ercc + 1)
ercc.scatter <- data.frame(log.ercc,
           Experiments = meta.transpose$experiment_number)

ercc.log.sum.experiments <- ercc.scatter %>%
  group_by(Experiments) %>%
  summarise_each(funs(sum))
dim(ercc.log.sum.experiments)
```

```
## [1] 19 93
```

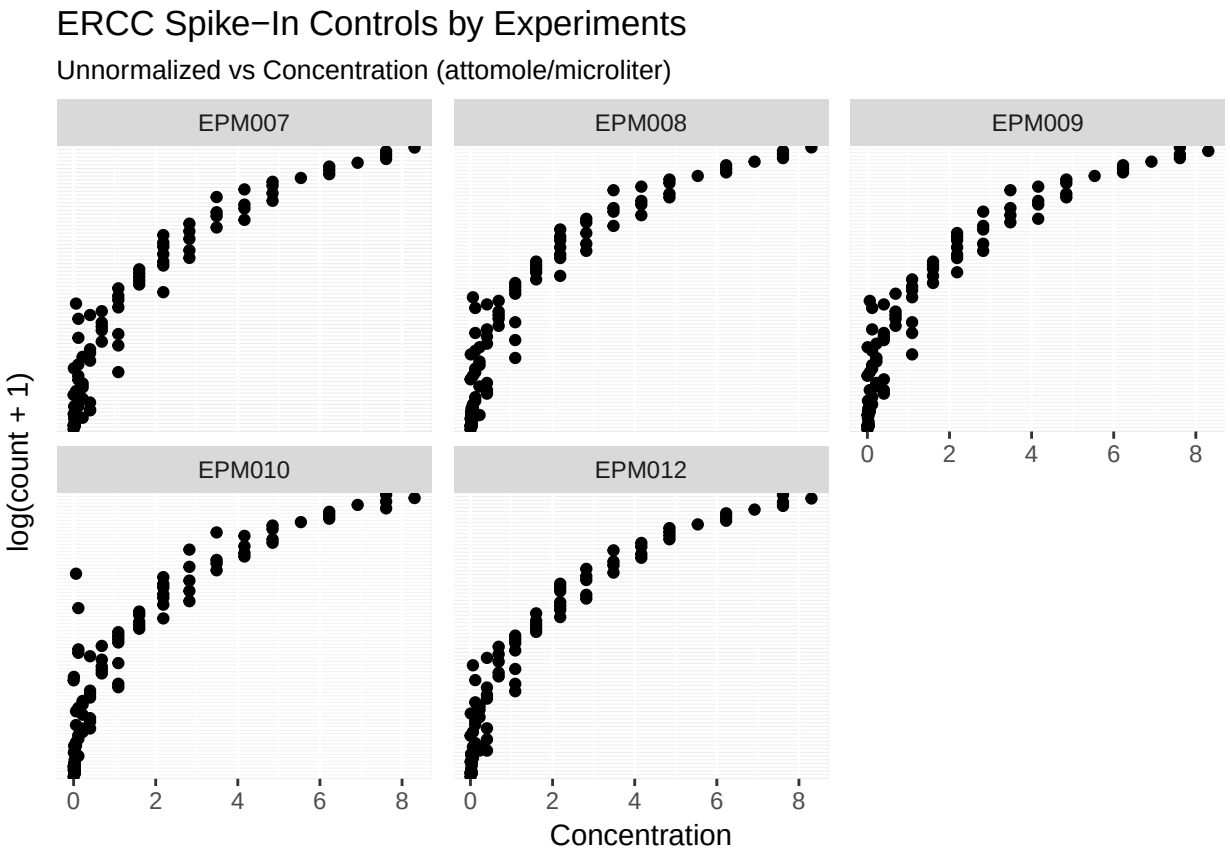
```
ercc.df <- data.frame(t(ercc.log.sum.experiments)[-1,])
dim(ercc.df)
```

```
## [1] 92 19
```

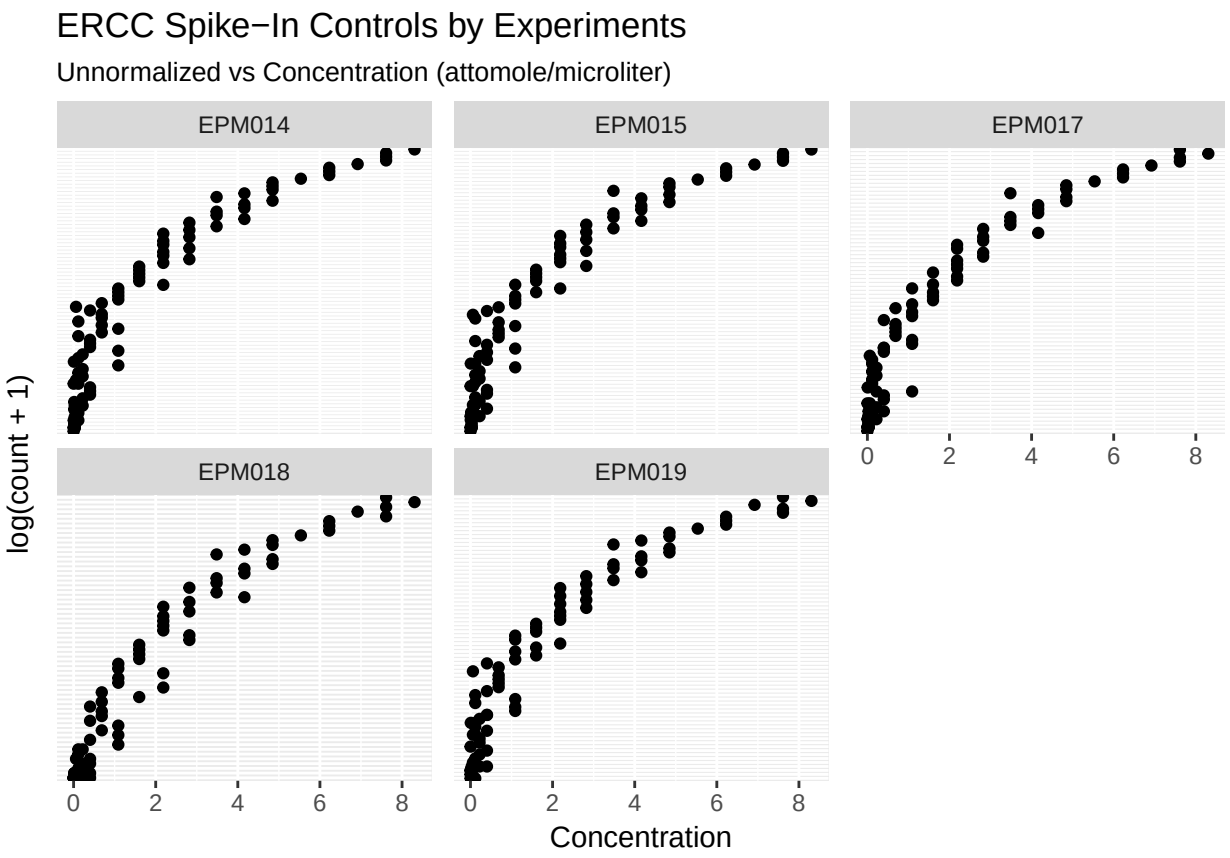
```
colnames(ercc.df) <- c(t(ercc.log.sum.experiments)[1,])

ercc.df.2 <- data.frame(ercc.df,
                      Gene = rownames(ercc.df),
                      Concentration = spike.count[order(spike.count$V1),]$V2)

ercc.df.group1 <- data.frame(Gene = ercc.df.2$Gene,
                             Concentration = ercc.df.2$Concentration,
                             ercc.df.2[,1:5])

ercc.scatter.df.group1 <- melt(ercc.df.group1, id.vars = c("Gene", "Concentration"))

ggplot(data = ercc.scatter.df.group1, mapping = aes(x = Concentration, y = value)) +
  geom_point() +
  facet_wrap(~ variable, scales = "free_y") +
  labs(y = "log(count + 1)") +
  theme(axis.ticks.y = element_blank(),
        axis.text.y=element_blank()) +
  labs(title = "ERCC Spike-In Controls by Experiments",
       subtitle = "Unnormalized vs Concentration (attomole/microliter)")
```




```
ercc.df.group2 <- data.frame(Gene = ercc.df.2$Gene,
                             Concentration = ercc.df.2$Concentration,
                             ercc.df.2[,6:10])

ercc.scatter.df.group2 <- melt(ercc.df.group2, id.vars = c("Gene", "Concentration"))

ggplot(data = ercc.scatter.df.group2, mapping = aes(x = Concentration, y = value)) +
  geom_point() +
  facet_wrap(~ variable, scales = "free_y") +
  labs(y = "log(count + 1)") +
  theme(axis.ticks.y = element_blank(),
        axis.text.y=element_blank()) +
  labs(title = "ERCC Spike-In Controls by Experiments",
       subtitle = "Unnormalized vs Concentration (attomole/microliter)")
```




```
ercc.df.group3 <- data.frame(Gene = ercc.df.2$Gene,
                             Concentration = ercc.df.2$Concentration,
                             ercc.df.2[,11:15])

ercc.scatter.df.group3 <- melt(ercc.df.group3, id.vars = c("Gene", "Concentration"))

ggplot(data = ercc.scatter.df.group3, mapping = aes(x = Concentration, y = value)) +
  geom_point() +
  facet_wrap(~ variable, scales = "free_y") +
  labs(y = "log(count + 1)") +
  theme(axis.ticks.y = element_blank(),
        axis.text.y=element_blank()) +
  labs(title = "ERCC Spike-In Controls by Experiments",
       subtitle = "Unnormalized vs Concentration (attomole/microliter)")
```



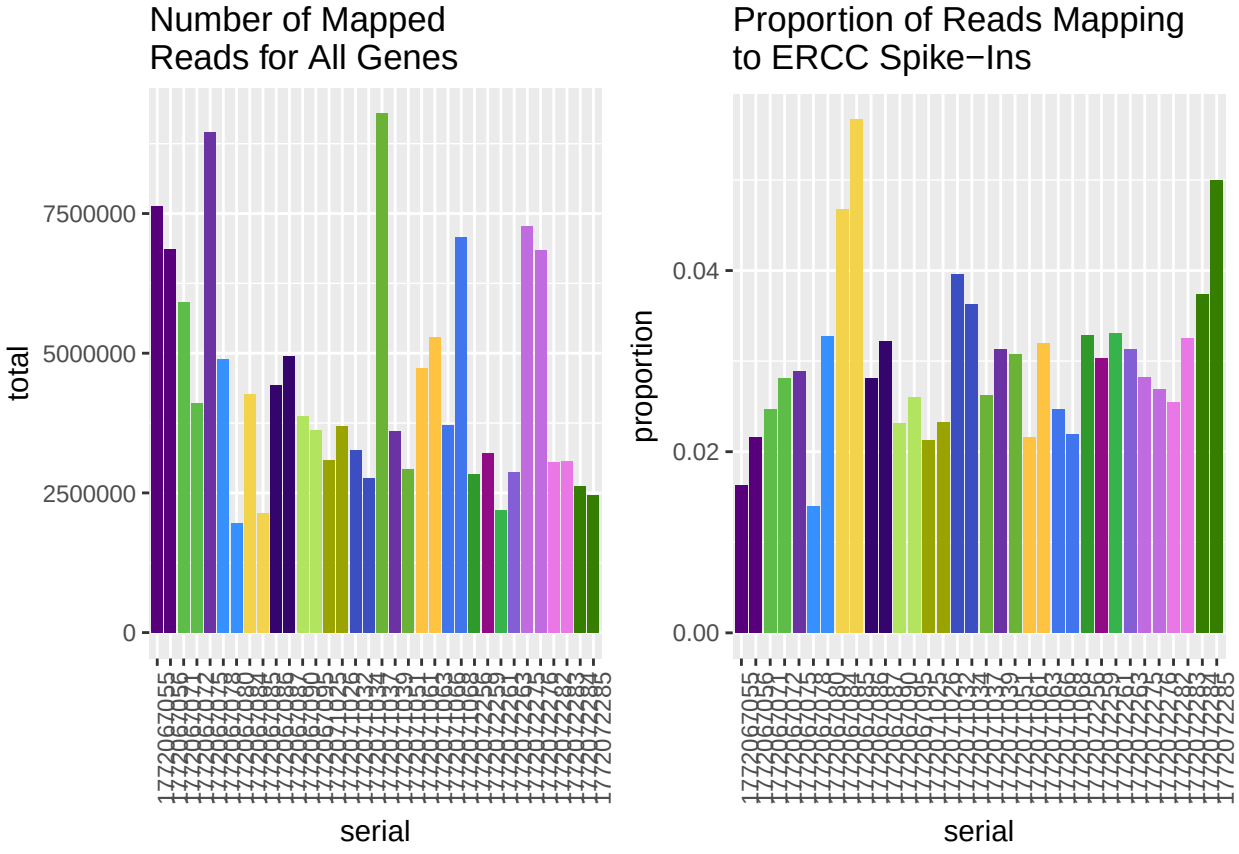
The expected concentration levels of the ERCC spike-in controls were provided by the authors Joost, et. al. They do not correspond to the stock concentrations found on Thermo Scientific because the concentrations were changed by the investigators. The expected amount of molecules from the spike-in species ideally expected were calculated considering the original concentraion of the spike-ins provided by ERCC, their molecular weight, the volumne used, the dilution factor, and the volume of the C1 reaction environment. Detail calculations are not available. Based on these plots, I see that the higher the concentration of the sequence, the better the read counts. I expect a linear relationship between read counts and nominal concentrations on the log scale. I see this more for the higher concentrations. But I also see a curvature pattern. From these plots, I can conclude that the controls are behaving more or less as they should.

```
#total gene counts
dim(geneLevelCountsTranspose)
```

```
## [1]  1422 26024
```

```
total.count.df <- data.frame(experiment_number = meta.transpose$experiment_number,
                             serial = meta.transpose$Chip.serial.number,
                             total = total.count)

total.count.serial <- aggregate(data = total.count.df, total ~., FUN = sum)

total.count.serial <- total.count.serial[order(total.count.serial$experiment_number),]

p1 <- ggplot(data = total.count.serial,
             mapping = aes(x = serial, y = total, fill = experiment_number)) +
  geom_bar(stat = "identity") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1),
```

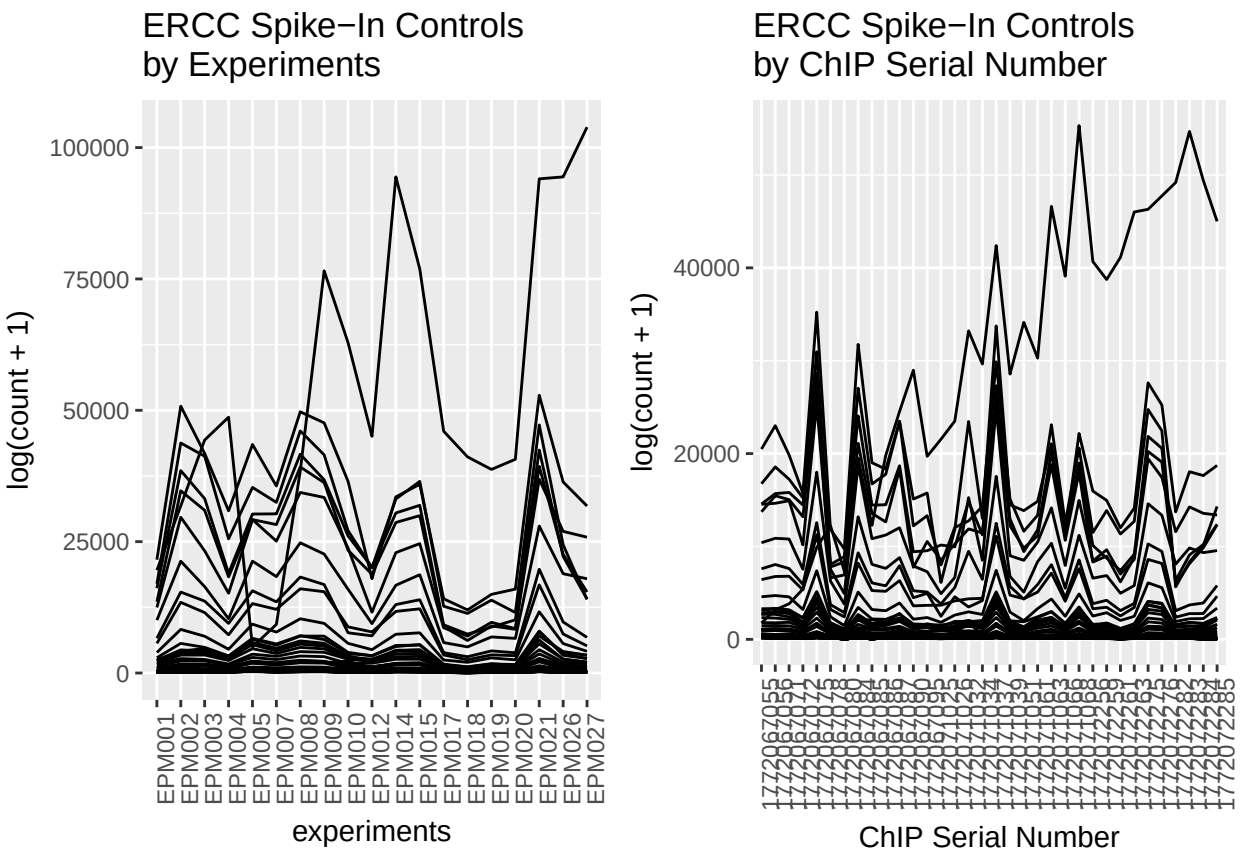
```
        legend.position="none") +
  scale_fill_manual(values = set3) +
  labs(title = "Number of Mapped \nReads for All Genes")

#proportion of ERCC spike-ins
ercc.proportion <- rowSums(ercc) / total.count
ercc.proportion.df <- data.frame(experiment_number = meta.transpose$experiment_number,
                                 serial = meta.transpose$Chip.serial.number,
                                 proportion = ercc.proportion)

ercc.proportion.serial <- aggregate(data = ercc.proportion.df, proportion ~., FUN = mean)

ercc.proportion.serial <- ercc.proportion.serial[order(ercc.proportion.serial$experiment_number),]

p2 <- ggplot(data = ercc.proportion.serial,
             mapping = aes(x = serial, y = proportion, fill = experiment_number)) +
  geom_bar(stat = "identity") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1),
        legend.position="none") +
  scale_fill_manual(values = set3) +
  labs(title = "Proportion of Reads Mapping \nto ERCC Spike-Ins")

multiplot(p1, p2, cols = 2)
```



We expect about 5% of the reads to map to ERCC controls. However, based on the barplot, it seems that on average, the percentage mapped to ERCC controls are lower. Further, there seems to be quite a bit of

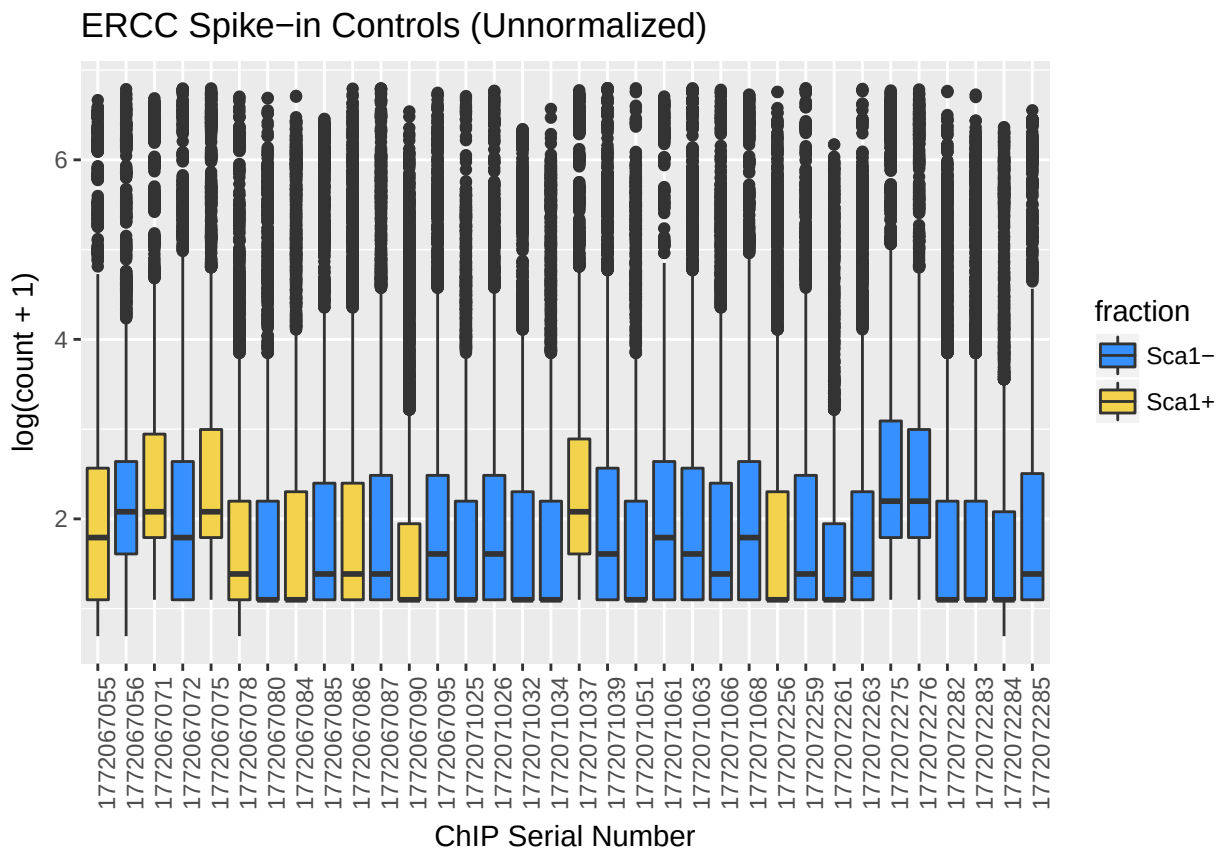
technical effect based on the different ChIP serial numbers. There doesn't seem to be a strong relationship between the total mapped reads for all genes and the proportion of mapped reads for the ERCC spike-in controls.

```
ercc.by.experiments <- data.frame(ercc,
                                  experiments = meta.transpose$experiment_number,
                                  id = 1:nrow(ercc))

ercc.sum.experiments <- ercc.by.experiments %>%
  group_by(experiments) %>%
  summarise_each(funs(sum))

plot_data <- melt(data.frame(ercc.sum.experiments), id.var= "experiments")

#it's not behaving well
p1 <- ggplot(data=plot_data, aes(x=experiments, y=value, group=variable)) +
   geom_line() +
   theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  labs(title = "ERCC Spike-In Controls \nby Experiments",
       y = "log(count + 1)")

#look at by chip serial number
ercc.by.serial <- data.frame(ercc,
                                  serial = meta.transpose$Chip.serial.number,
                                  id = 1:nrow(ercc))
ercc.sum.serial <- ercc.by.serial %>%
  group_by(serial) %>%
  summarise_each(funs(sum))

plot.data.serial <- melt(data.frame(ercc.sum.serial), id.var= "serial")

p2 <- ggplot(data=plot.data.serial, aes(x=serial, y=value, group=variable)) +
   geom_line() +
   theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  labs(title = "ERCC Spike-In Controls \nby ChIP Serial Number",
       x = "ChIP Serial Number",
       y = "log(count + 1)")

multiplot(p1, p2, cols = 2)
```



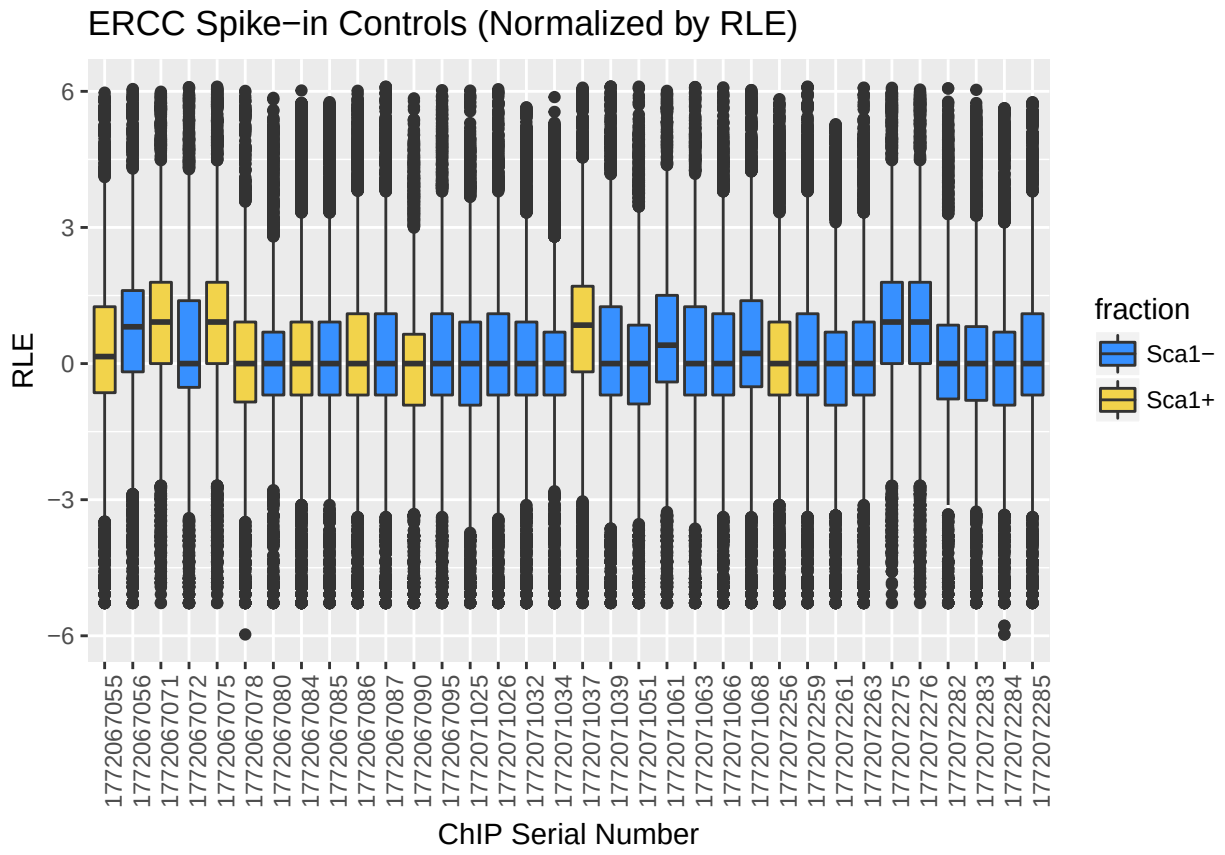
The line plots above show that the ERCC spike-in controls are not working as they ought to work. There are 92 lines, one for each control gene. If the controls are working well, I would see horizontal lines across experiments. That is, I would see similar counts of genes across experiments and ChIP serial numbers since I expect the control genes to be expressed at the same levels despite the different circumstances. Instead, I see quite a bit of variation. There seems to be not too much variation in the lower concentrations but quite a bit of noise in the higher concentrations.

```
#ercc multiple boxplot

log.ercc.by.serial <- data.frame(log.ercc,
                                 serial = meta.transpose$Chip.serial.number,
                                 fraction = meta.transpose$cell_fraction)

melt.log.ercc.by.serial <- melt(log.ercc.by.serial, id.vars = c("serial", "fraction"))

p1 <- ggplot(melt.log.ercc.by.serial) +
  geom_boxplot(aes(x = serial, y = value, fill = fraction)) +
  scale_fill_manual(values = set3) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  labs(title = "ERCC Spike-in Controls (Unnormalized)",
       x = "ChIP Serial Number",
       y = "log(count + 1)")

p1
```




```
#rle

rle.ercc <- log(ercc / colMedians(as.matrix(ercc)))

rle.ercc.by.serial <- data.frame(rle.ercc,
                                 serial = meta.transpose$Chip.serial.number,
                                 fraction = meta.transpose$cell_fraction)
```

```
melt.rle.ercc.by.serial <- melt(rle.ercc.by.serial, id.vars = c("serial", "fraction"))

p2 <- ggplot(melt.rle.ercc.by.serial) +
  geom_boxplot(aes(x = serial, y = value, fill = fraction)) +
  scale_fill_manual(values = set3) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  labs(title = "ERCC Spike-in Controls (Normalized by RLE)",
       x = "ChIP Serial Number",
       y = "RLE")

p2
```




```
# all genes multiple boxplot

log.count <- log(geneLevelCountsTranspose + 1)

log.all.by.serial <- data.frame(log.count,
                                serial = meta.transpose$Chip.serial.number,
                                fraction = meta.transpose$cell_fraction)

melt.log.all.by.serial <- melt(log.all.by.serial, id.vars = c("serial", "fraction"))

all.genes.log.boxplot <- ggplot(data = melt.all.log.by.serial,
                                mapping = aes(x = as.factor(serial),
                                              y = value, fill = fraction)) +
  geom_boxplot() +
```

```
  labs(title = "All Genes (Unnormalized)",
      x = "ChIP Serial Number",
      y = "log(count + 1)") +
  scale_fill_manual(values = set3) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))

# all genes rle

rle.all <- log(geneLevelCountsTranspose / colMedians(as.matrix(geneLevelCountsTranspose)))

rle.all.by.serial <- data.frame(rle.all,
                                serial = meta.transpose$Chip.serial.number,
                                fraction = meta.transpose$cell_fraction)

melt.all.rle.by.serial <- melt(rle.all.by.serial, id.vars = c("serial", "fraction"))

all.genes.rle.boxplot <- ggplot(melt.rle.all.by.serial,
                                mapping = aes(x = as.factor(serial),
                                              y = value,
                                              fill = fraction)) +
   geom_boxplot() +
   labs(title = "All Genes (Normalized by RLE)",
        x = "ChIP Serial Number",
        y = "RLE") +
   scale_fill_manual(values = set3) +
   theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

```
all.genes.log.boxplot

all.genes.rle.boxplot
```

These boxplots show that the ERCC spike-in controls are not behaving as they should. The unnormalized boxplots are expected to have the same mean and spread. The normalized plots are expected to be lined up around zero and have similar spreads as well. Further, the boxplots of the unnormalized log counts of all genes reveal that the distributions of the gene counts are right skewed. That is, there are many outliers at the upper quantiles of the distribution. The RLE boxplots further confirm the large number of outliers present in the data.

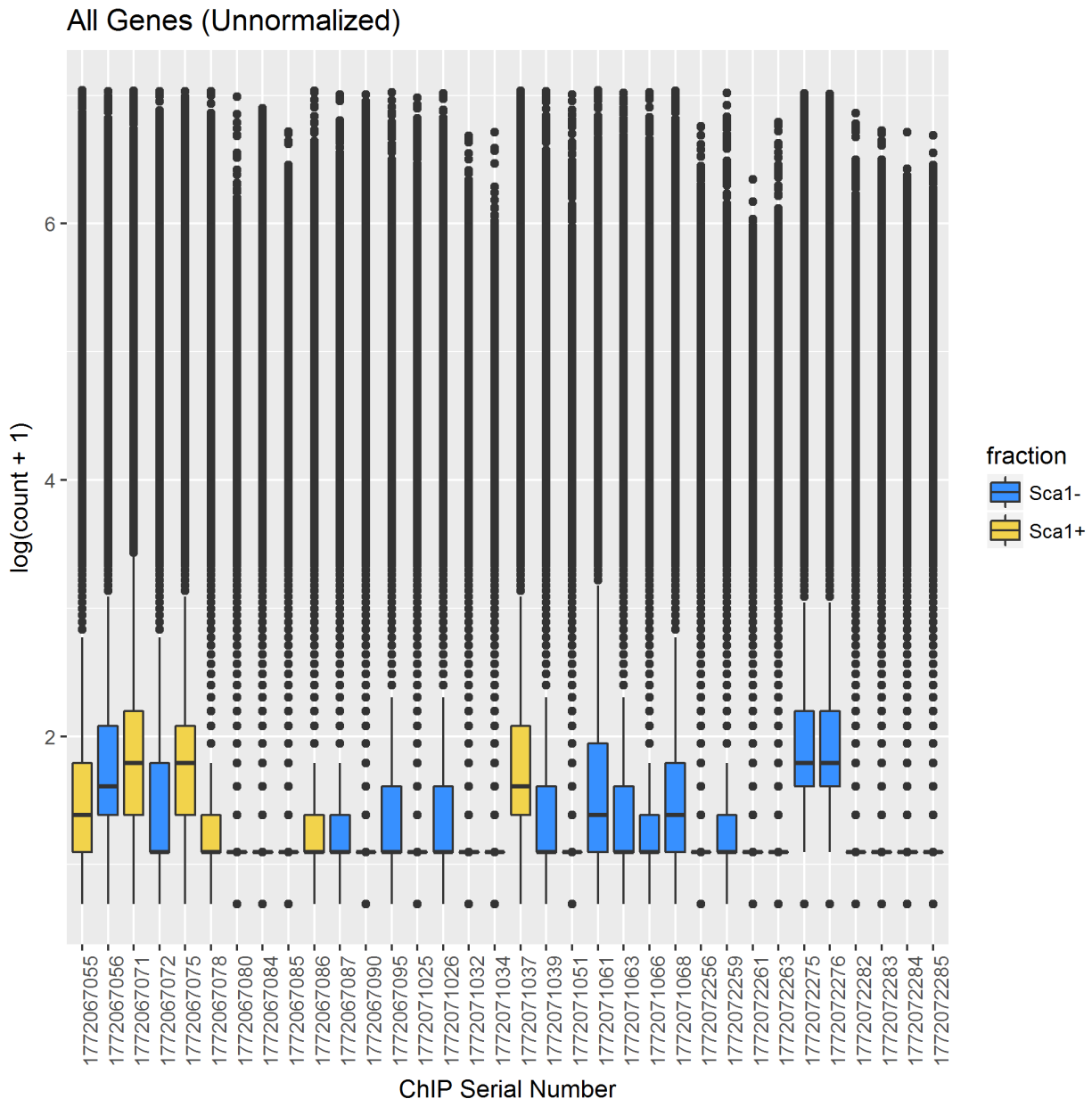


Figure 1: All Genes (Unnormalized)

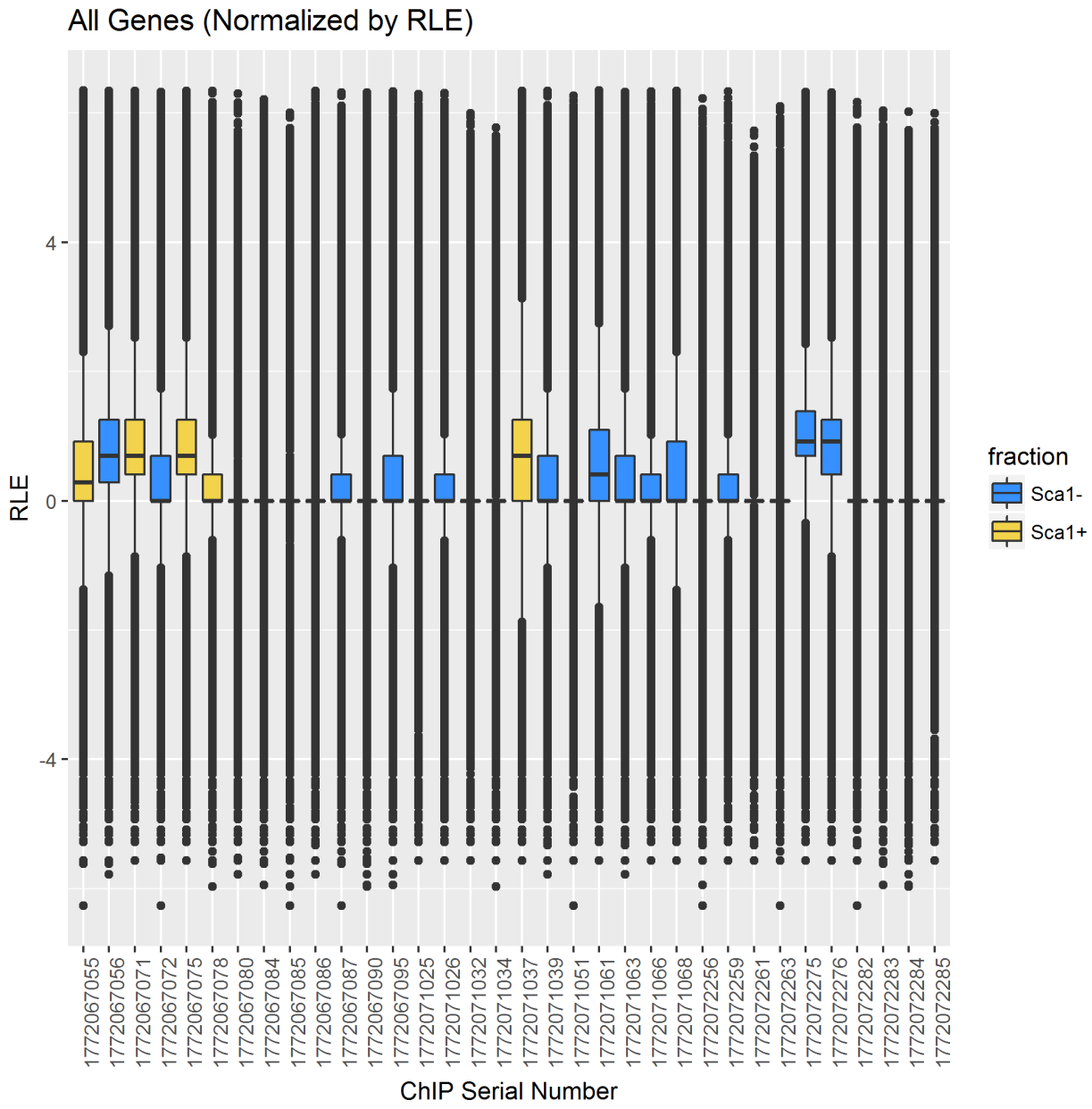


Figure 2: All Genes (Normalized by RLE)

2.c. Assess the extent of zero inflation for the Joost et al. (2016) dataset.

First, I want to see how many genes have zero count in any cell sample.

```
sum(geneLevelCountsTranspose == 0)
```

```
## [1] 0
```

There are zero genes that have zero counts in any cell.

Now, I check for any read counts that are less than zero in the count dataset.

```
sum(geneLevelCountsTranspose < 20)
```

```
## [1] 36627368
```

There are 36627368 counts that are less than 20.

I will define a a count to be zero if it's less than 20 and then assess the extent of zero inflation for this dataset.

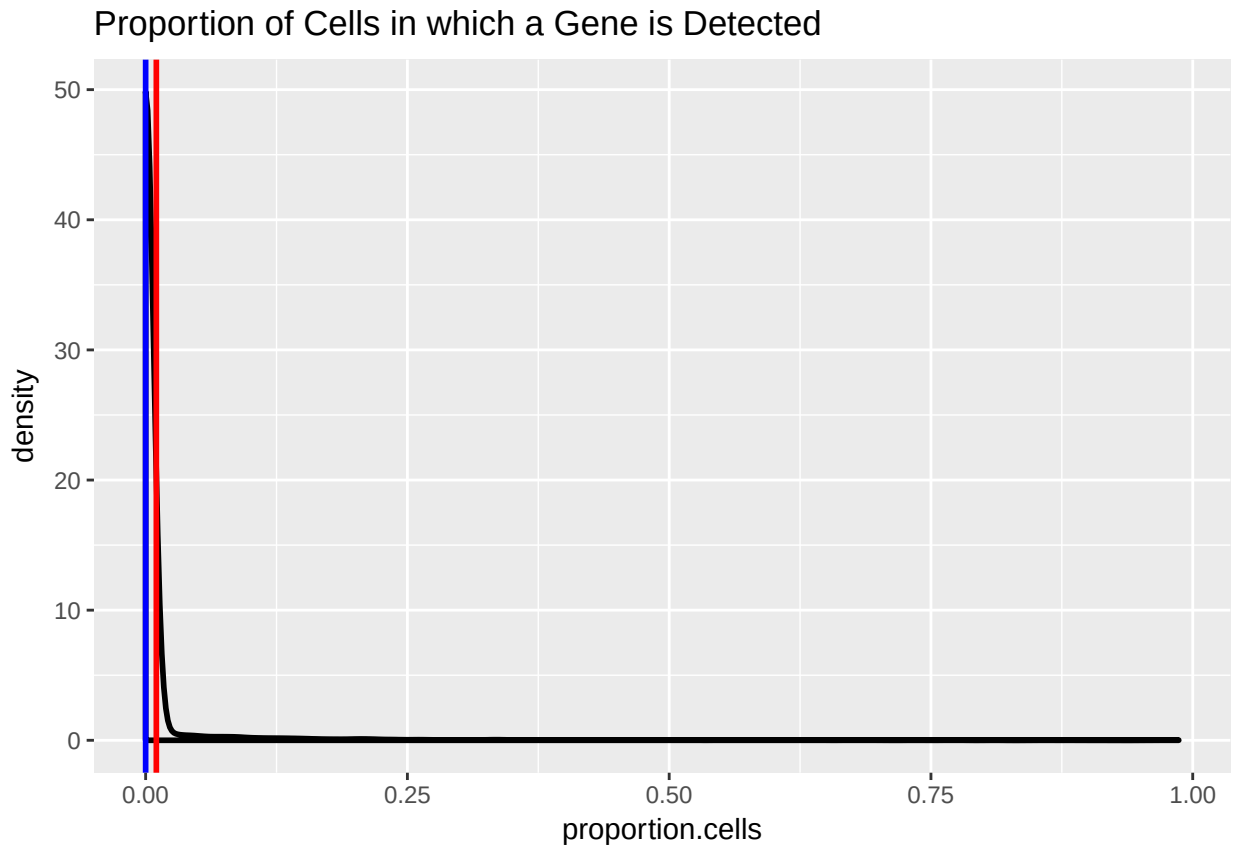
```
#zero-count gene filtering (at least 20 reads in at least 40 cells)
sum(colSums(geneLevelCountsTranspose >= 20) >= 40)
```

```
## [1] 1316
```

```
#proportion of genes with zero count in each cell sample (less than 20 reads)

proportion.zero <- rowSums(geneLevelCountsTranspose < 20) /
  ncol(geneLevelCountsTranspose)

ggplot(meta.transpose, aes(x = cell.samples,
                           y = proportion.zero,
                           fill = experiment_number)) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = set2) +
  theme(axis.text.x=element_blank(),
        axis.ticks.x=element_blank()) +
  labs(title = "Proportion of Genes with Zero Counts \nBefore Gene Filtering",
       subtitle = "Colored by Experiment Number") +
  theme(legend.position="none")
```

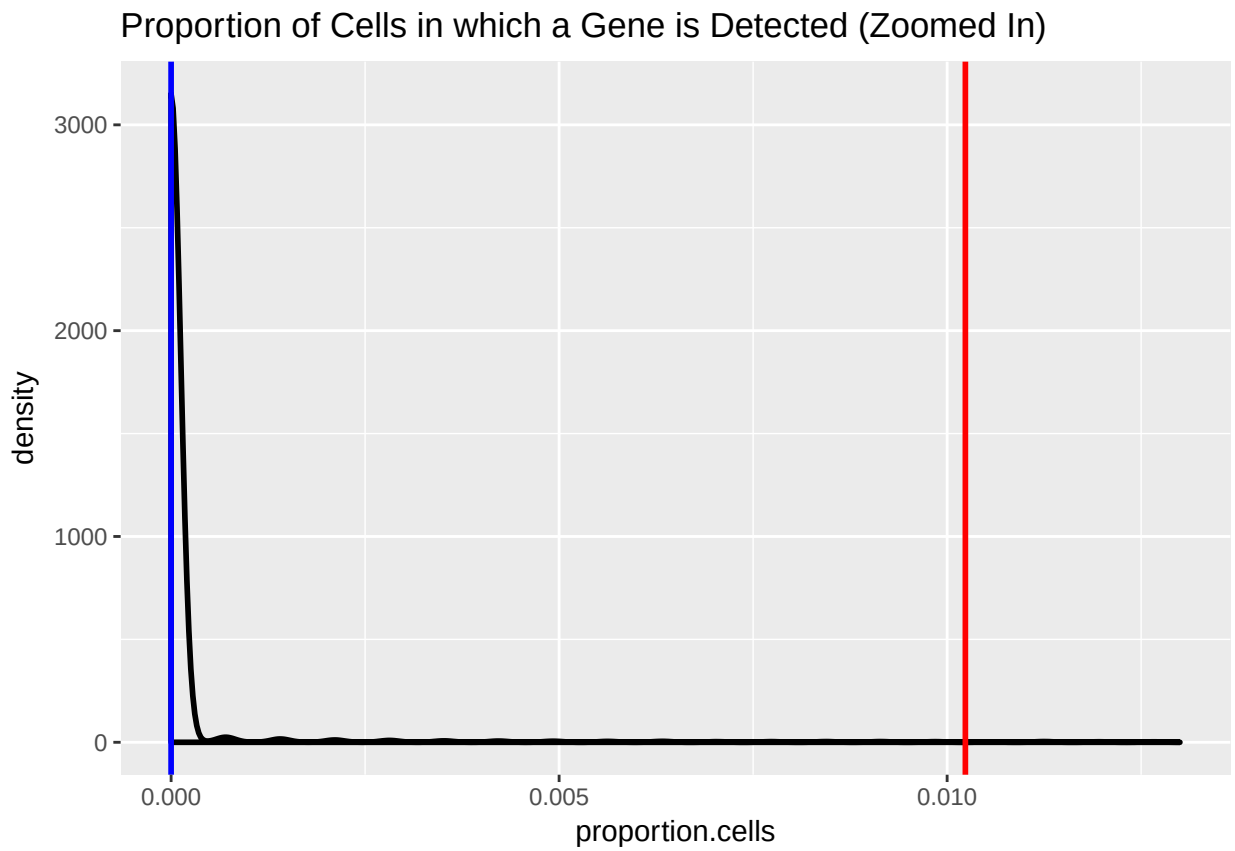
The plot above shows that the proportion of genes with zero counts in each cell is very high.

```
#proportion of cells in which a gene is detected

proportion.cells <- colSums(geneLevelCountsTranspose >= 20) /
  nrow(geneLevelCountsTranspose)

mean.proportion <- mean(proportion.cells)
median.proportion <- median(proportion.cells)

proportion.cells <- data.frame(proportion.cells)

ggplot(data = proportion.cells,
       mapping = aes(x = proportion.cells)) +
  geom_density(size = 1) +
  labs(title = "Proportion of Cells in which a Gene is Detected") +
  geom_vline(xintercept = mean.proportion, col = "red", size = 1) +
  geom_vline(xintercept = median.proportion, col = "blue", size = 1)
```




```
ggplot(data = proportion.cells,
       mapping = aes(x = proportion.cells)) +
  geom_density(size = 1) +
  labs(title = "Proportion of Cells in which a Gene is Detected (Zoomed In)") +
  geom_vline(xintercept = mean.proportion, col = "red", size = 1) +
  geom_vline(xintercept = median.proportion, col = "blue", size = 1) +
  xlim(0, 0.013)
```

```
## Warning: Removed 1530 rows containing non-finite values (stat_density).
```

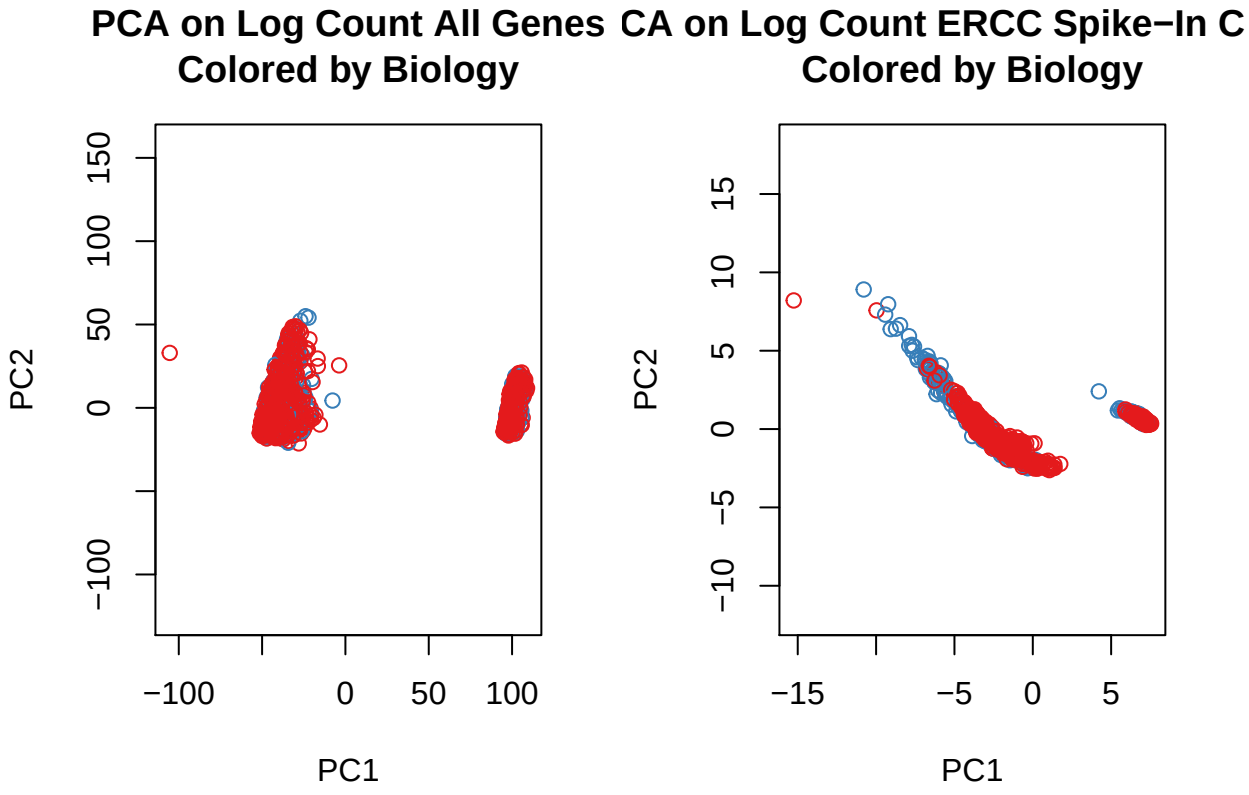


The plot above shows the density of the proportion of cells in which a gene is detected for a particular gene. The red line indicates the mean value. Blue line indicates median value. The median is zero. The plot shows that most genes have counts less than 20 in most of the cells.

```
# PCA by Biology

pca.all.genes <- prcomp(log.count)$x
```

```
pca.ercc <- prcomp(log.ercc)$x
```

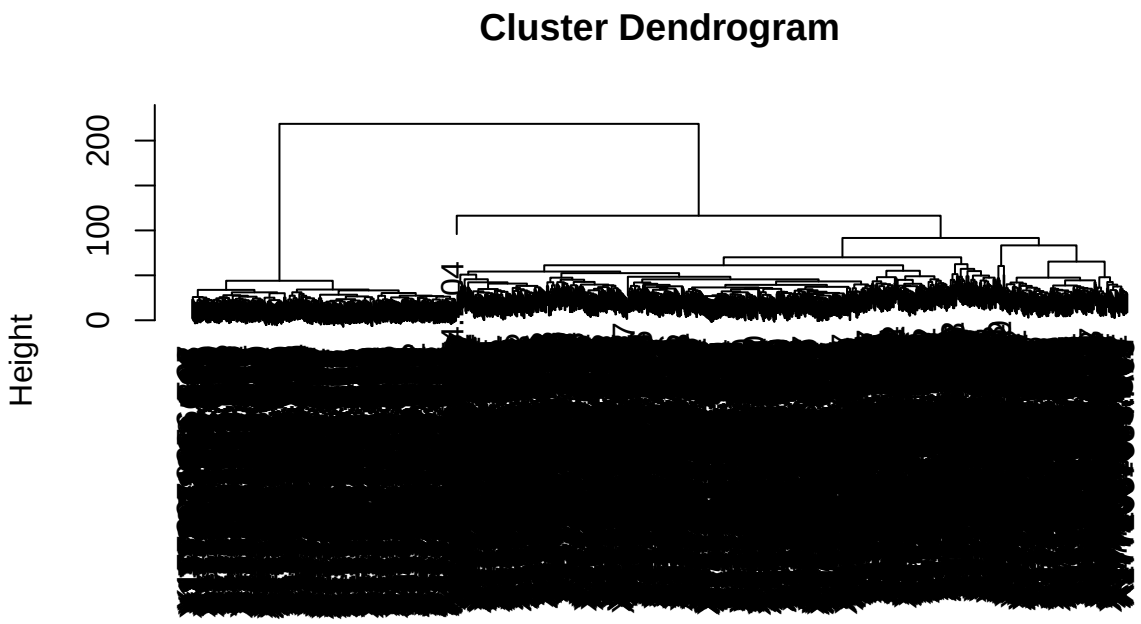
```
par(mfrow = c(1, 2))

plot(pca.all.genes[,1:2],
     asp=1,
     col=set1[meta.transpose$cell_fraction],
     main = "PCA on Log Count All Genes\nColored by Biology")

plot(pca.ercc[,1:2],
     asp=1,
     col=set1[meta.transpose$cell_fraction],
     main = "PCA on Log Count ERCC Spike-In Control\nColored by Biology")
```



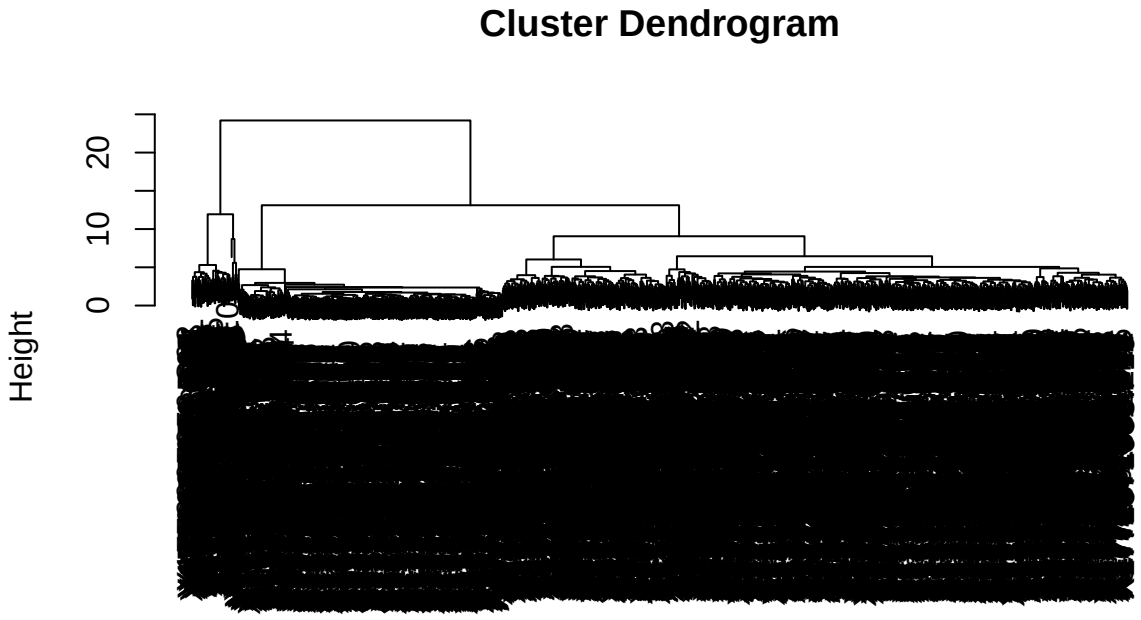

```
par(mfrow = c(1, 1))
```

Above plots of the first principal component vs the second principal component reveal that there are two main clusters of cells for both all cells and ERCC spike-in controls. The clustering is not by biology. I tried to color the points by many different categories but could not figure out the cause for the clustering.

```
#Hierarchical Clustering

dist.log.all.genes <- dist(log.count)
hclust.log.all.genes <- hclust(dist.log.all.genes, "complete")
plot(hclust.log.all.genes, xlab=NA, sub=NA)
```




```
dist.log.ercc.genes <- dist(log.ercc)
hclust.log.ercc.genes <- hclust(dist.log.ercc.genes, "complete")
plot(hclust.log.ercc.genes, xlab=NA, sub=NA)
```



The dendrograms further confirm the PCA results that there are two main clusterings for all the genes as well as for ERCC spike-in controls.

Question 3. Normalization

a. Choose a filtering scheme to reduce the number of genes in the dataset. What percentage of the reads do you filter out?

As my filtering scheme, I will only select the genes that have at least 20 reads in at least 40 of the samples.

```
unwanted.gene.total <- sum(colSums(geneLevelCountsTranspose >= 20) < 40)

unwanted.gene.total

unwanted.gene.percentage <-  unwanted.gene.total /
  ncol(geneLevelCountsTranspose)

unwanted.gene.percentage

wanted.columns <- colSums(geneLevelCountsTranspose >= 20) >= 40

filtered.genes <- geneLevelCountsTranspose[ , which(wanted.columns == 1)]
```

Using this scheme, I filter out 94.9% percent of genes, or 24708 / 26024 of the genes. I keep 1316 / 26024 of the genes using this scheme.