7659 HW5

Guannan Shen

October 22, 2018

Contents

1 HW5 **1**
1.1 1. Next Generation Sequencing: Sample Size Estimates 1
1.1.1 (a) Using rnapower(), recreate Figure 3 from the journal club paper, Hart. 1
1.1.2 (b) For the Montgomery data, create a row in Table 1 in the Hart et al. paper. 3
1.1.3 (c) Calculate the biological coefficient of variations (CV) from the Montgomery 4
1.1.4 (d) Using rnapower(), recreate Figure 3 from Hart et al. again 5
1.1.5 (e) Using rnapower(), recreate the curve (not the histogram) in the top Figure 4 . . . 6
1.2 2. Next Generation Sequencing: Pre-Processing 7
1.2.1 (a) Within geneLevelData, how many genes have all zeros as counts? How many have at least one sample with a zero? 7
1.2.2 (b) For the following plots, use the log scale 8
1.2.3 (c) Apply withinLaneNormalization() to normalize by GC content. 9
1.2.4 (d) Using the within-lane normalized data from the previous part 11

```
## set up workspace
library(knitr)
library(tidyverse)
library(RNASeqPower)
library(edgeR)
library(cqn)
library(EDASeq)
library(yeastRNASeq)
options(stringsAsFactors = F)
options(dplyr.width = Inf)
getwd()
```

```
## [1] "/home/guanshim/Documents/Stats/CIDA_OMICs/7659Stats_Genetics/HW5"
```

```
## not in function
"%nin%" <- Negate("%in%")
```

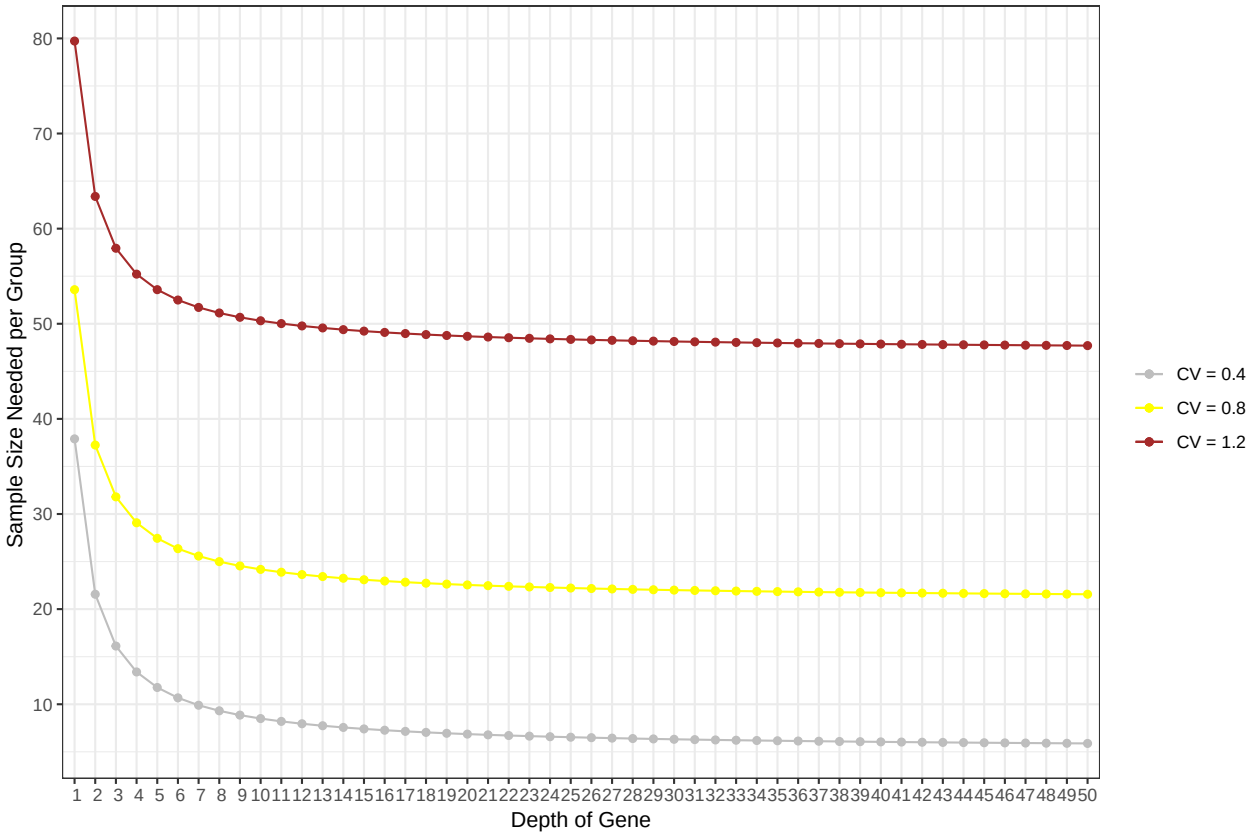
1 HW5

1.1 1. Next Generation Sequencing: Sample Size Estimates

1.1.1 (a) Using rnapower(), recreate Figure 3 from the journal club paper, Hart.

```
## montgomery data from cqn
data(montgomery.subset)
## GC and gene length of montgomery
data(uCovar)
## vector of length 10 containing the number of mapped reads
## for each sample
```

```r
data(sizeFactors.subset)

########## Understand the dataset ######## help(montgomery) number of
########## genes genes that have zero counts in all 10 samples were
########## already excluded
ng_mont <- nrow(montgomery.subset)

############## Question 1 figure 3 ################ sample size (ss) vs
############## depth sample size per group
ssize_depth <- sapply(c(0.4, 0.8, 1.2), function(y) {
    sapply(1:50, function(x) {
        rnapower(depth = x, cv = y, effect = 2, alpha = 0.05,
            power = 0.8)
    })
})
ssize_depth <- data.frame(ssize_depth)
colnames(ssize_depth) <- c("V1", "V2", "V3")

### Plot
ggplot(ssize_depth, aes(x = 1:50)) + geom_line(aes(y = V1, color = "CV = 0.4")) +
    geom_point(aes(y = V1, color = "CV = 0.4")) + geom_line(aes(y = V2,
    color = "CV = 0.8")) + geom_point(aes(y = V2, color = "CV = 0.8")) +
    geom_line(aes(y = V3, color = "CV = 1.2")) + geom_point(aes(y = V3,
    color = "CV = 1.2")) +
scale_x_discrete(name = "Depth of Gene", limits = c(1:50)) +
    scale_y_continuous(name = "Sample Size Needed per Group ",
        breaks = c(0, 10, 20, 30, 40, 50, 60, 70, 80)) + theme_bw() +
    scale_colour_manual("", breaks = c("CV = 0.4", "CV = 0.8",
        "CV = 1.2"), values = c(`CV = 0.4` = "grey", `CV = 0.8` = "yellow",
        `CV = 1.2` = "brown"))
```



1.1.2 (b) For the Montgomery data, create a row in Table 1 in the Hart et al. paper.

```
########## average of sequence reads aligning to the gene/ depth
########## ############ how many reads are assigned to a particular
########## gene / depth ## is a data frame with 23552 observations on
########## 10 different samples ##
N_total <- sum(sizeFactors.subset)
## number of genes genes that have zero counts in all 10
## samples were already excluded
ng_mont
```

```
## [1] 23552
```

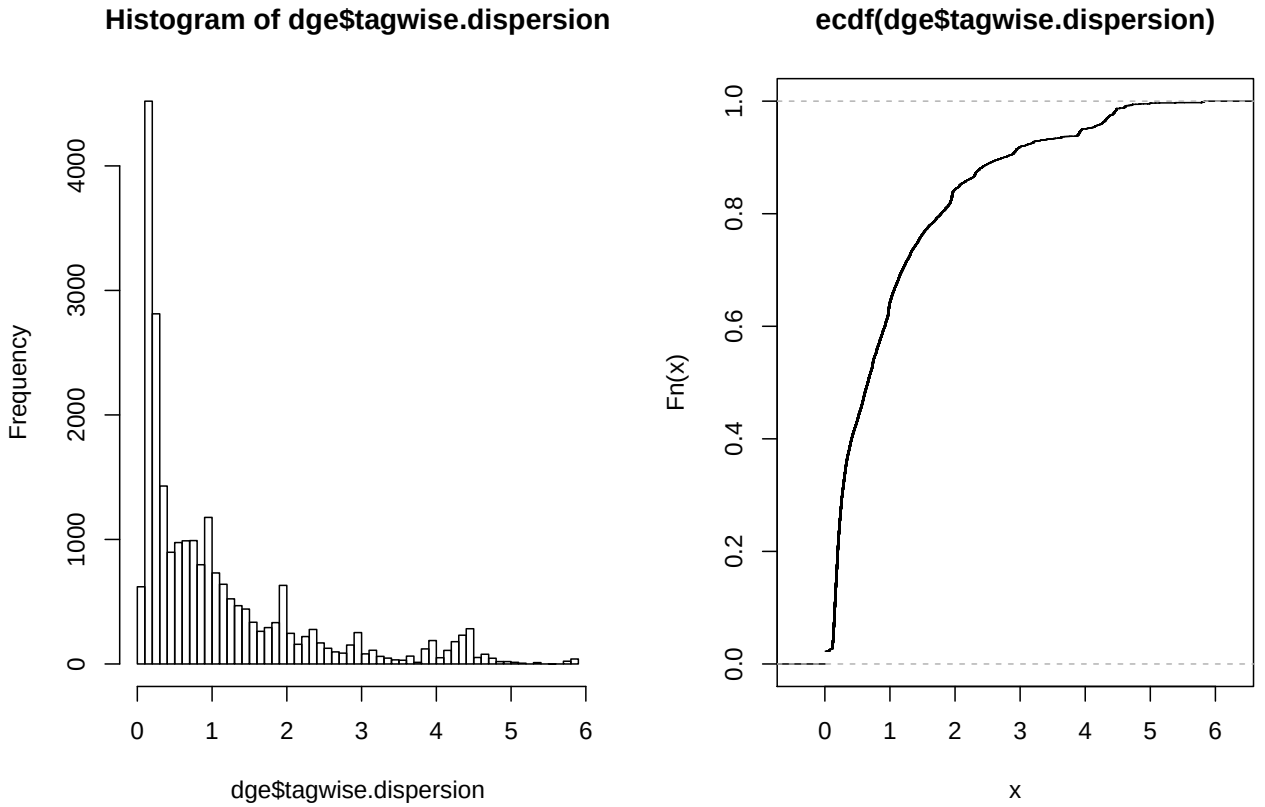
```
counts_gene_million <- rowSums(montgomery.subset)/N_total * 1e+06

mont_counts <- data.frame(Sample = "Montgomery", n = 10, Reads = round(N_total/(ng_mont *
    10), 2), mapped = "100%", a = round(sum(counts_gene_million <
    0.01)/ng_mont, 2), b = round(sum(0.01 <= counts_gene_million &
    counts_gene_million < 0.1)/ng_mont, 2), c = round(sum(0.1 <=
    counts_gene_million & counts_gene_million < 1)/ng_mont, 2),
    d = round(sum(1 <= counts_gene_million & counts_gene_million <
        10)/ng_mont, 2), e = round(sum(10 <= counts_gene_million &
        counts_gene_million < 100)/ng_mont, 2), f = round(sum(100 <=
        counts_gene_million & counts_gene_million < 1000)/ng_mont,
        2), g = round(sum(1000 <= counts_gene_million)/ng_mont,
        2))
```

```
colnames(mont_counts) <- c("Sample", "n", "Avg Reads", "% mapped",
    "<0.01", "0.01-0.1", "0.1-1", "1-10", "10-100", "100-1000",
    ">1000")
kable(mont_counts)
```

Sample	n	Avg Reads	% mapped	<0.01	0.01–0.1	0.1–1	1–10	10-100	100-1000	>1000
Montgomery	10	164.66	100%	0	0.23	0.22	0.18	0.27	0.09	0

1.1.3 (c) Calculate the biological coefficient of variations (CV) from the Montgomery

```
## edgeR DGElist
dge <- DGEList(counts = montgomery.subset, lib.size = sizeFactors.subset,
    group = rep(1, length(sizeFactors.subset)))
dge <- estimateCommonDisp(dge)
dge <- estimateTagwiseDisp(dge)

## the distribution of tagwise dispersion (genewise variation)
par(mfrow = c(1, 2))
hist(dge$tagwise.dispersion, breaks = 60)
plot(ecdf(dge$tagwise.dispersion), ylab = "Fn(x)")
```




```
## median and 90% quantile
med_mont <- median(dge$tagwise.dispersion)
med_mont
```

```
## [1] 0.6530353
```

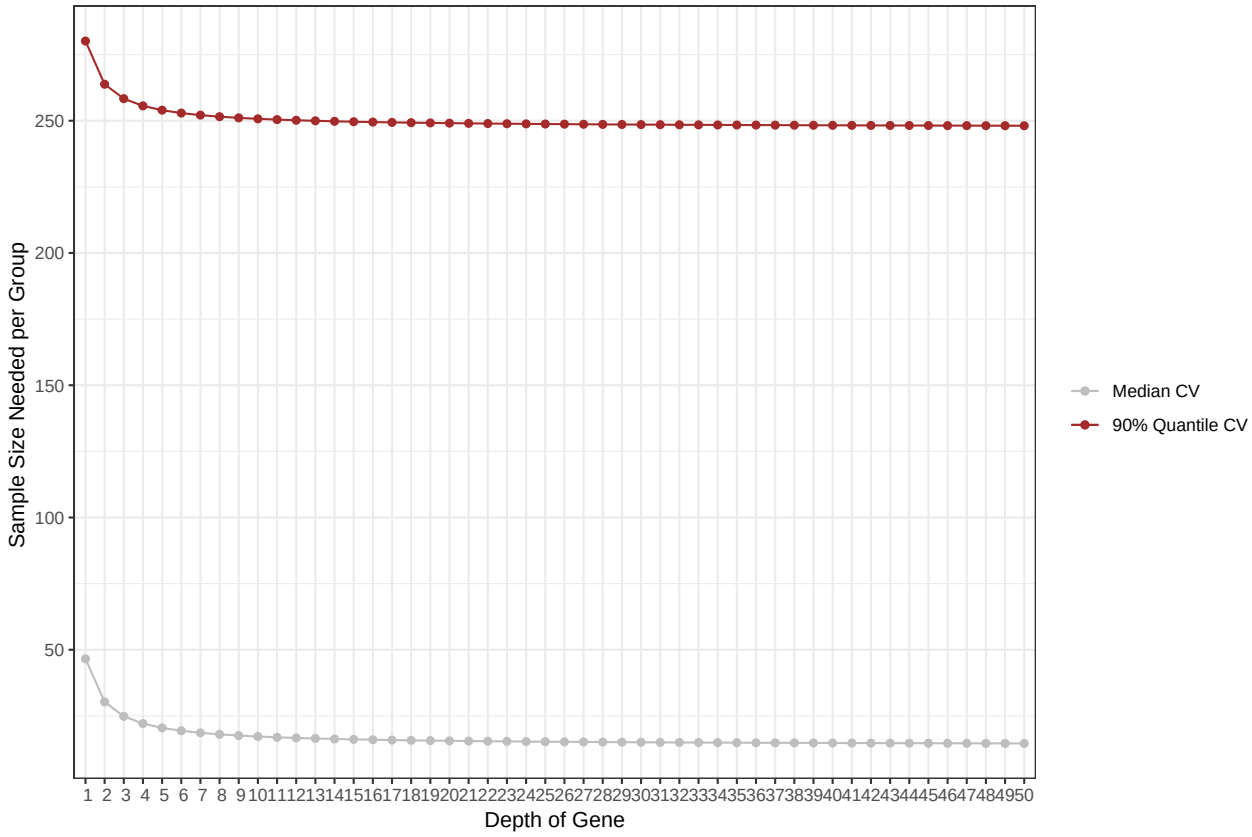
```
quan09_mont <- as.numeric(quantile(dge$tagwise.dispersion, probs = 0.9))
quan09_mont
```

```
## [1] 2.75203
```

The median and 90% percentile of the tagwise dispersions of whole genes in the Montgomery dataset are 0.653 and 2.752, respectively.

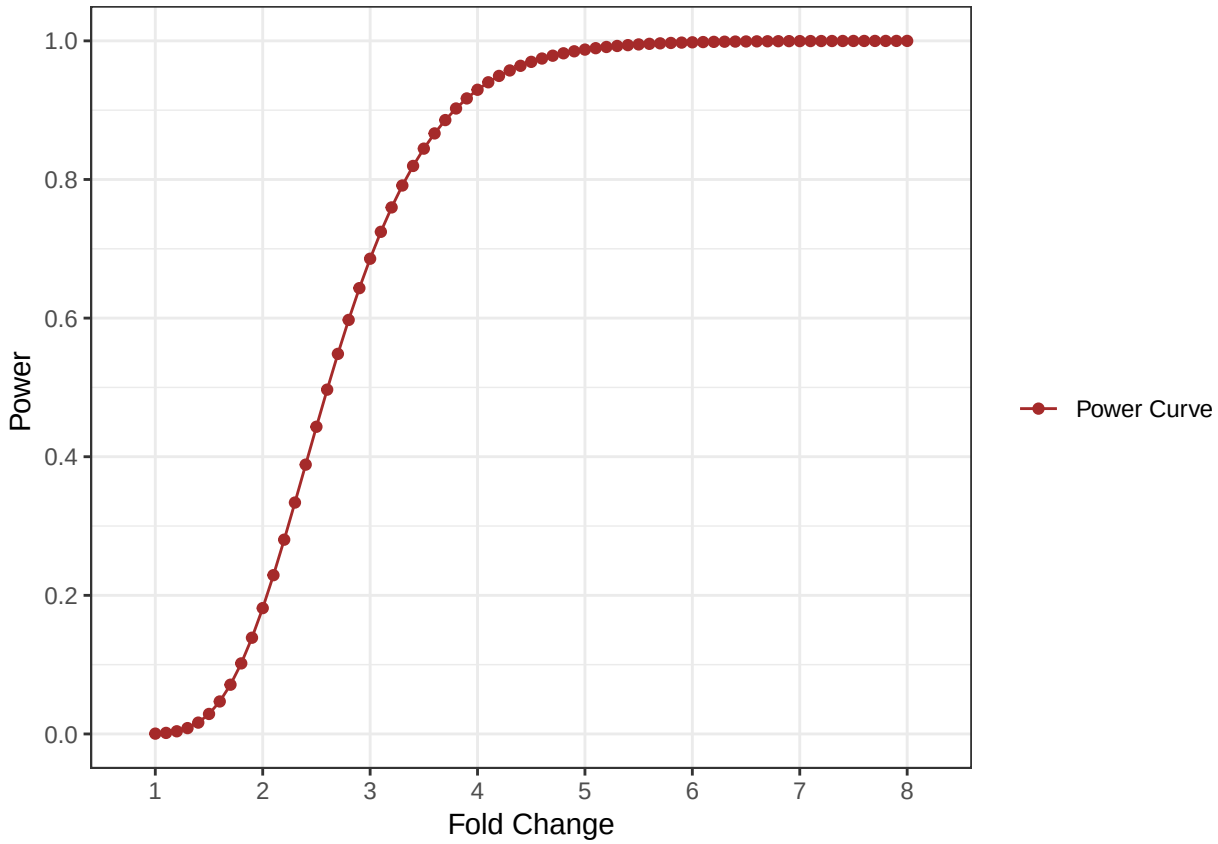
1.1.4 (d) Using rnapower(), recreate Figure 3 from Hart et al. again

```
## sample size (ss) vs depth sample size per group
ssize_depth_mont <- sapply(c(med_mont, quan09_mont), function(y) {
    sapply(1:50, function(x) {
        rnapower(depth = x, cv = y, effect = 2, alpha = 0.05,
            power = 0.8)
    })
})
ssize_depth_mont <- data.frame(ssize_depth_mont)
colnames(ssize_depth_mont) <- c("V1", "V2")

### Plot
ggplot(ssize_depth_mont, aes(x = 1:50)) + geom_line(aes(y = V1,
    color = "Median CV")) + geom_point(aes(y = V1, color = "Median CV")) +
    geom_line(aes(y = V2, color = "90% Quantile CV")) + geom_point(aes(y = V2,
    color = "90% Quantile CV")) + scale_x_discrete(name = "Depth of Gene",
    limits = c(1:50)) + scale_y_continuous(name = "Sample Size Needed per Group ",
    breaks = c(50, 100, 150, 200, 250, 300)) + theme_bw() + scale_colour_manual("",
    breaks = c("Median CV", "90% Quantile CV"), values = c(`Median CV` = "grey",
        `90% Quantile CV` = "brown"))
```



1.1.5 (e) Using rnapower(), recreate the curve (not the histogram) in the top Figure 4

```
## power curve for n = 20 per group, coverage of 100, s = 0.32
## (60th percentile of observed) and a = 0.001.
s06 <- as.numeric(quantile(dge$tagwise.dispersion, probs = 0.6))
p_curve <- sapply(seq(1, 8, by = 0.1), function(x) {
    rnapower(depth = 100, cv = s06, n = 20, effect = x, alpha = 0.001)
})
p_curve <- data.frame(p_curve)
colnames(p_curve) <- c("V1")

### Plot
ggplot(p_curve, aes(x = seq(1, 8, by = 0.1))) + geom_line(aes(y = V1,
    color = "Power Curve")) + geom_point(aes(y = V1, color = "Power Curve")) +
    scale_x_discrete(name = "Fold Change", limits = c(1:8)) +
    scale_y_continuous(name = "Power ", breaks = c(0, 0.2, 0.4,
        0.6, 0.8, 1)) + theme_bw() + scale_colour_manual("",
    breaks = c("Power Curve"), values = c(`Power Curve` = "brown"))
```



1.2 2. Next Generation Sequencing: Pre-Processing

1.2.1 (a) Within geneLevelData, how many genes have all zeros as counts? How many have at least one sample with a zero?

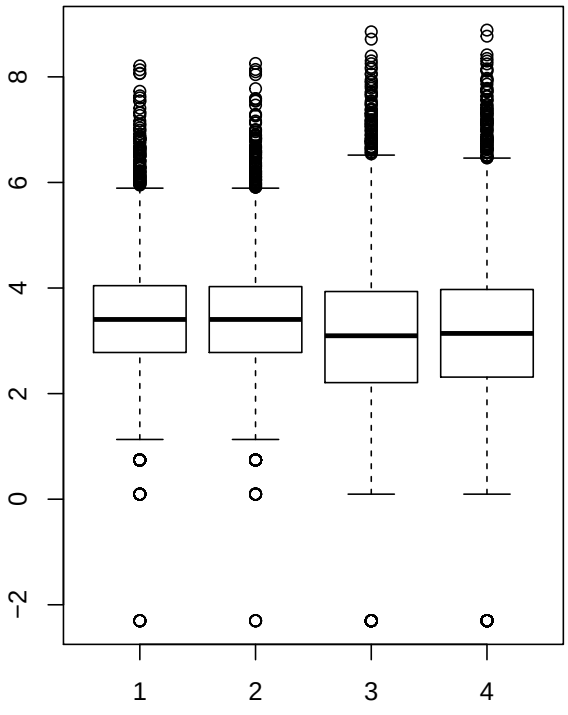
```
## yeast RNA-seq data set on two mutant and two wildtype
## strains
data(geneLevelData)
## also load the GC content and length
data(yeastGC)
data(yeastLength)
## GC and gene length of montgomery data(uCovar)

## genes have all zeros as counts
zero_yeast <- sum(rowSums(geneLevelData) == 0)
## at least one sample with a zero
onezero_yeast <- sum(apply(geneLevelData, 1, function(x) {
    any(x == 0)
}))
## only containing genes with  10 counts
geneLevelDataFilter <- geneLevelData[rowSums(geneLevelData) >=
    10, ]

## SeqExpressionSet object for the EDASeq functions.
exprs = as.matrix(geneLevelDataFilter)  # matrix of counts
```

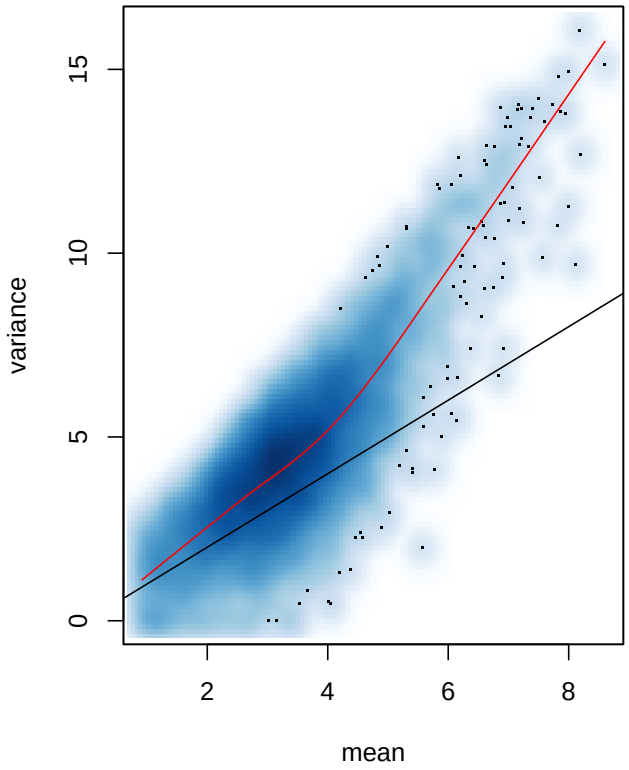
```
sub = intersect(rownames(geneLevelDataFilter), names(yeastGC))
exprs = exprs[sub, ]  #only examine genes with annotated GC content/length
row.names(exprs) = NULL  #remove row and column names
colnames(exprs) = NULL

# Create SeqExpressionSet, which contains counts, labels for
# the samples and GC content/length
counts <- newSeqExpressionSet(counts = exprs, phenoData = data.frame(conditions = factor(colnames(geneL
    featureData = AnnotatedDataFrame(data.frame(gc = yeastGC[sub],
        length = yeastLength[sub])))
```

557 genes have all zeros as counts. 1043 genes have at least one sample with a zero.

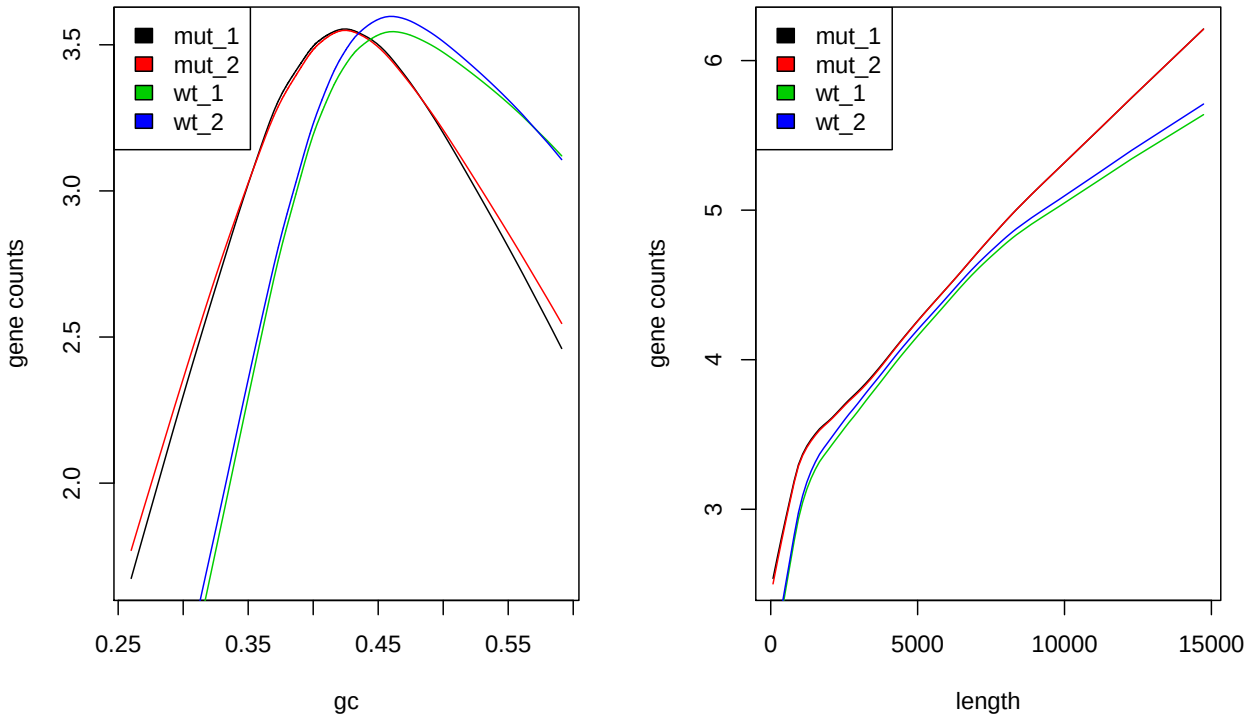
1.2.2 (b) For the following plots, use the log scale

```
par(mfrow = c(1, 2))
## To plot the counts by sample
boxplot(counts)

## plot the mean by variance plot a smoothScatter plot of the
## mean variance relation a lowess fit
meanVarPlot(counts, log = T)
```



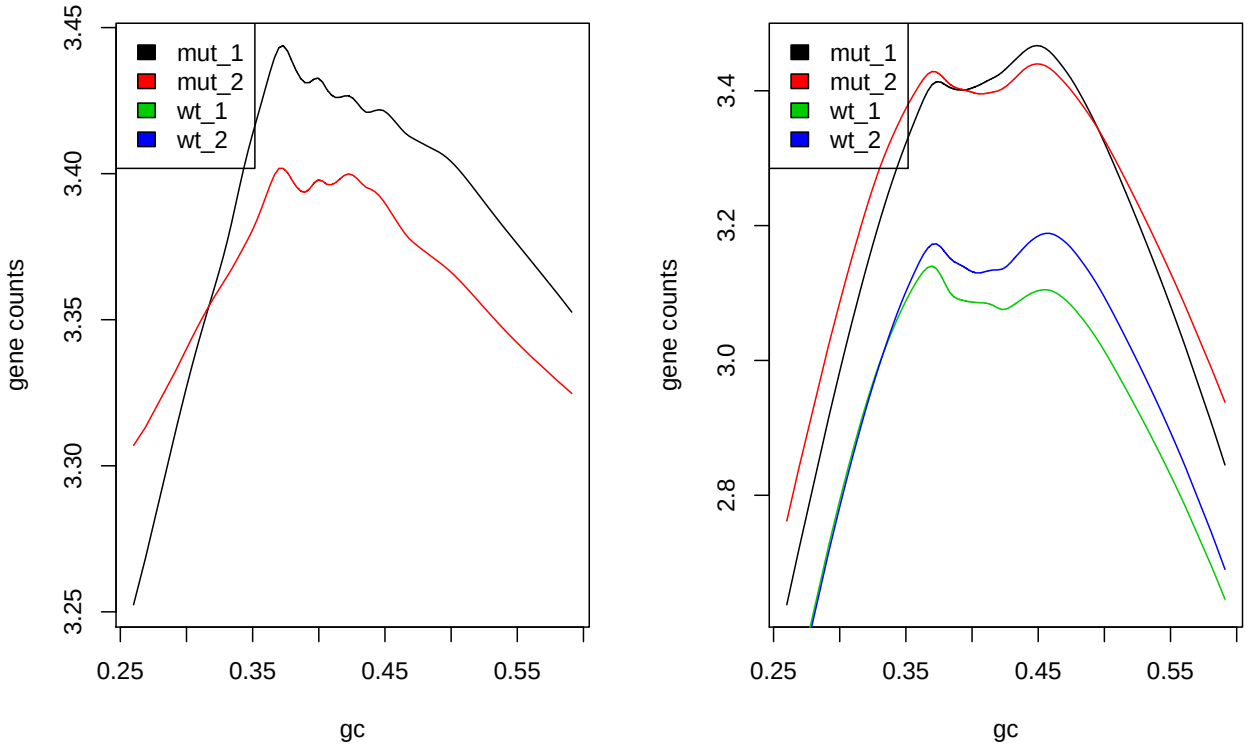



```
## To assess any biases by GC content
biasPlot(counts, "gc", log = TRUE)

## To assess any biases by length
biasPlot(counts, "length", log = TRUE)
```

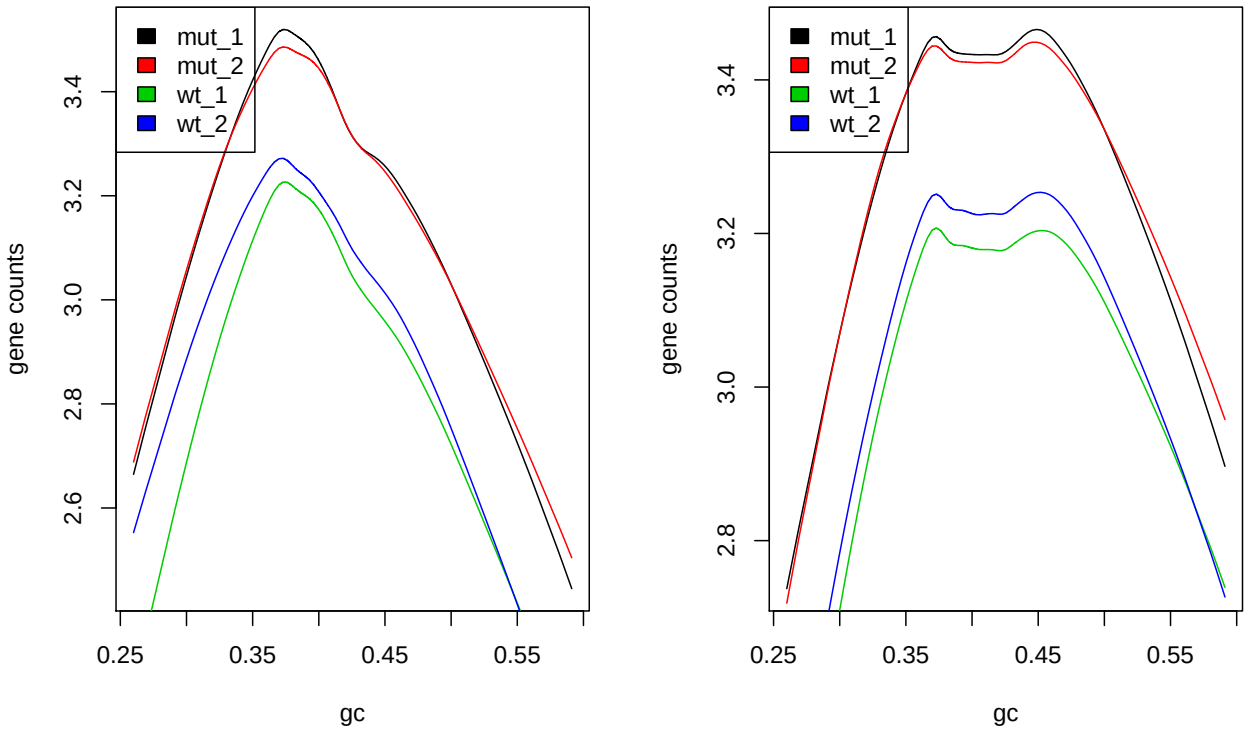


1.2.3 (c) Apply withinLaneNormalization() to normalize by GC content.

```
par(mfrow = c(1, 2))
## which=c('loess','median','upper','full') normalize by GC
## content which = 'loess'
norm_loess <- withinLaneNormalization(counts, "gc", which = "loess",
    offset = FALSE)
biasPlot(norm_loess, "gc", log = TRUE)

## which = 'median'
norm_median <- withinLaneNormalization(counts, "gc", which = "median",
    offset = FALSE)

biasPlot(norm_median, "gc", log = TRUE)
```

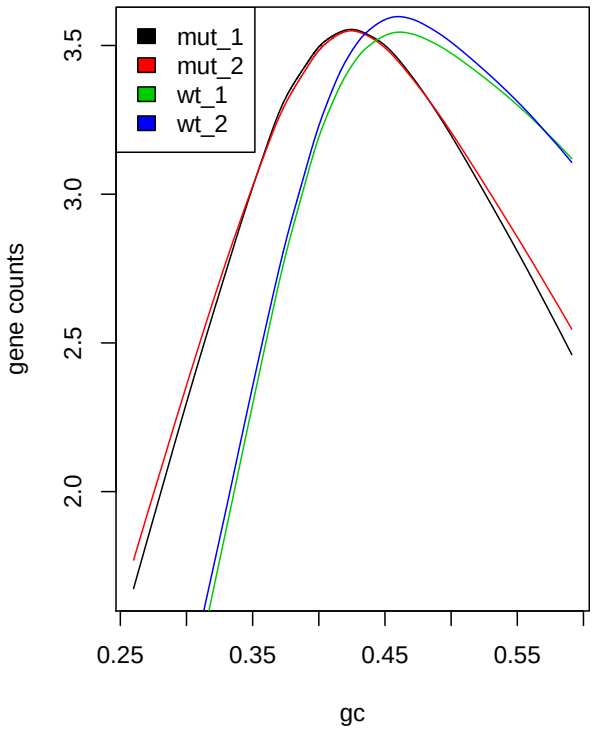


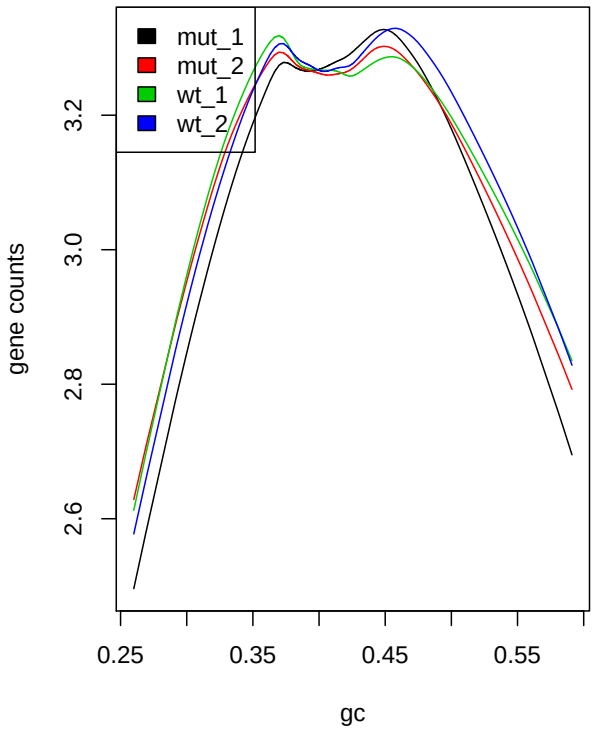

```
## which = 'upper'
norm_upper <- withinLaneNormalization(counts, "gc", which = "upper",
    offset = FALSE)

biasPlot(norm_upper, "gc", log = TRUE)


## which = 'full'
norm_full <- withinLaneNormalization(counts, "gc", which = "full",
    offset = FALSE)

biasPlot(norm_full, "gc", log = TRUE)
```



1.2.4 (d) Using the within-lane normalized data from the previous part

```
par(mfrow = c(1, 2))
biasPlot(counts, "gc", log = TRUE)
## 'median','upper','full'
bet_norm_median <- betweenLaneNormalization(norm_median, which = "median",
    offset = FALSE)
biasPlot(bet_norm_median, "gc", log = TRUE)
```






```
bet_norm_upper <- betweenLaneNormalization(norm_upper, which = "upper",
    offset = FALSE)
biasPlot(bet_norm_upper, "gc", log = TRUE)

bet_norm_full <- betweenLaneNormalization(norm_full, which = "full",
    offset = FALSE)
biasPlot(bet_norm_full, "gc", log = TRUE)
```