7659 HW5

Guannan Shen

October 22, 2018

Table of Contents

[HW5 2](#_Toc528057525)

[1. Next Generation Sequencing: Sample Size Estimates 2](#_Toc528057526)

[(a) Using rnapower(), recreate Figure 3 from the journal club paper, Hart. 2](#_Toc528057527)

[(b) For the Montgomery data, create a row in Table 1 in the Hart et al. paper. 3](#_Toc528057528)

[(c) Calculate the biological coefficient of variations (CV) from the Montgomery 4](#_Toc528057529)

[(d) Using rnapower(), recreate Figure 3 from Hart et al. again 5](#_Toc528057530)

[(e) Using rnapower(), recreate the curve (not the histogram) in the top Figure 4 6](#_Toc528057531)

[2. Next Generation Sequencing: Pre-Processing 8](#_Toc528057532)

[(a) Within geneLevelData, how many genes have all zeros as counts? How many have at least one sample with a zero? 8](#_Toc528057533)

[(b) For the following plots, use the log scale 8](#_Toc528057534)

[(c) Apply withinLaneNormalization() to normalize by GC content. 10](#_Toc528057535)

[(d) Using the within-lane normalized data from the previous part 12](#_Toc528057536)

## 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] "C:/Users/hithr/Documents/Stats/CIDA\_OMICs/7659Stats\_Genetics/HW5"

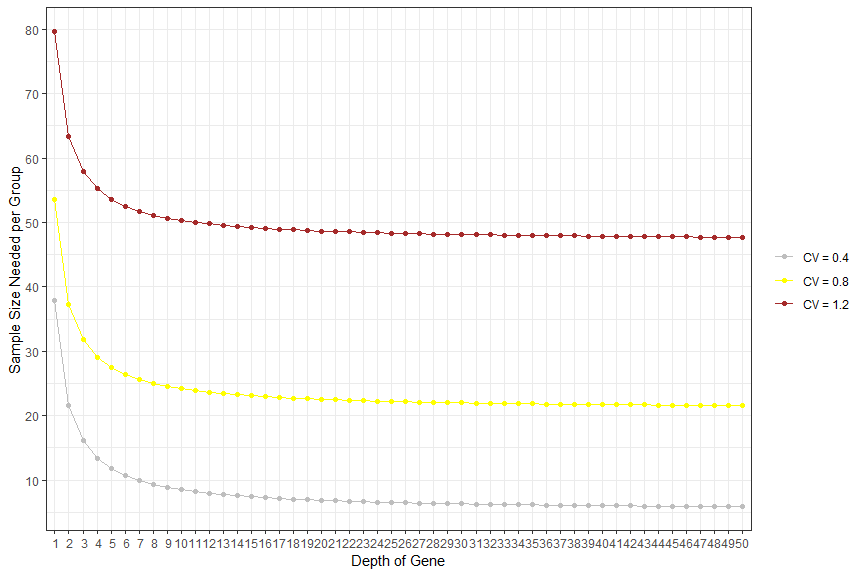
## not in function  
"%nin%" <- Negate("%in%")

# HW5

## 1. Next Generation Sequencing: Sample Size Estimates

### (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  
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"))



### 

### The above figure shows the required sample size needed to detect a two-fold difference in expression with 80% power at alpha = 0.05 for three different CV values and a range of sequencing depths. The alpha = 0.01 in the Hart et al., (2013) paper, but 0.05 here to match the curves in the original plot. As mentioned in the paper, “once technical variation becomes 1/10 the biological variation (roughly Depth > 10), any gain in precision through increased depth will be negligible”.

### (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

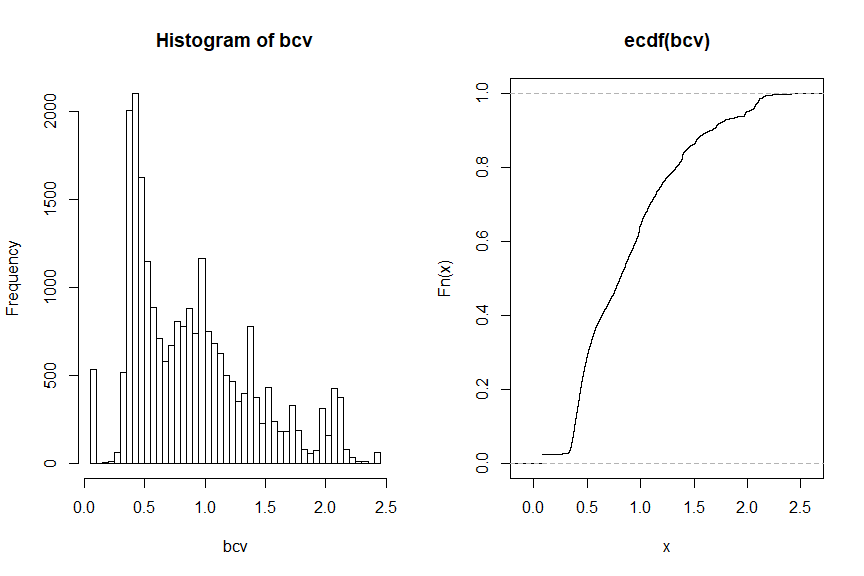
counts\_gene\_million <- rowSums(montgomery.subset)/N\_total \* 1e+06  
  
mont\_counts <- data.frame(Sample = "Montgomery", n = 10, Reads = round(N\_total/(ng\_mont \*   
 10), 2), mapped = "100%", a = round(sum(counts\_gene\_million <   
 0.01)/ng\_mont, 2), b = round(sum(0.01 <= counts\_gene\_million &   
 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)

### 

### Compared to the other data sets in Table 1, the results for Montgomery data here are different. The overall “Counts per million mapped” here is less than the datasets in Table 1, left shifted. The peaks of the other datasets are located within the range of 10-100. But here there is a 2-fold higher proportion for range 0.01-0.1, thus the “Counts per million mapped” distributes almost evenly across 0.01-0.1, 0.1-1 and 10-100. In summary, the table suggests the average coverage in Montgomery dataset is relatively low.

### (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))  
## bcv is estimated as the sqrt of dispersion  
bcv <- sqrt(dge$tagwise.dispersion)  
## plots  
hist(bcv, breaks = 60)  
plot(ecdf(bcv), ylab = "Fn(x)")



## median and 90% quantile  
med\_mont <- median(bcv)  
med\_mont

## [1] 0.808106

quan09\_mont <- as.numeric(quantile(bcv, probs = 0.9))  
quan09\_mont

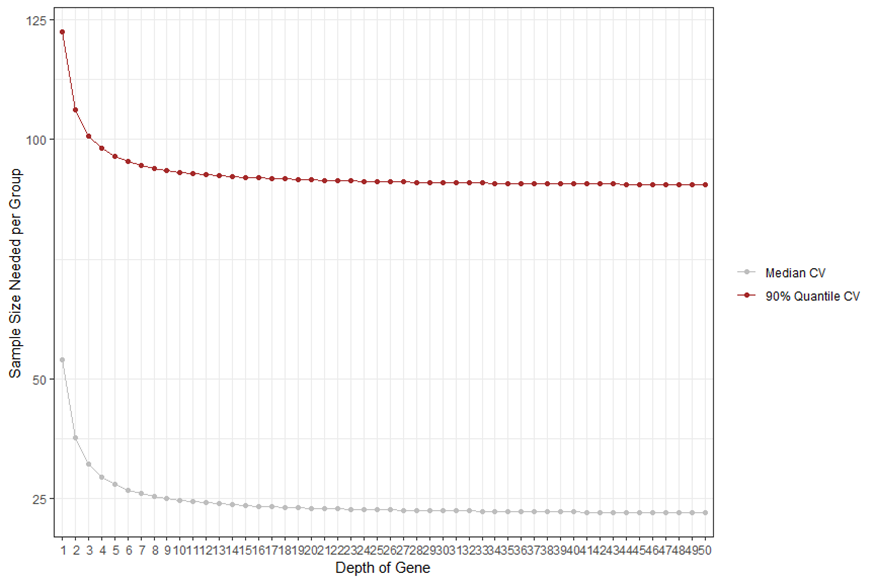
## [1] 1.658924

The PMC3378882 paper suggests the BCV (biological coefficient of variations) is the square root of the tagwise dispersion. Thus, the median and 90% percentile of the BCV of whole genes in the Montgomery dataset are 0.8081 and 1.6589, respectively.

The 90% quantiles of the human samples in the Hart et al., (2013) paper range from 0.32 to 0.74 with a median of 0.43. In summary, 10 human samples here have a larger mode of BCV and BCV is greater, generally.

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

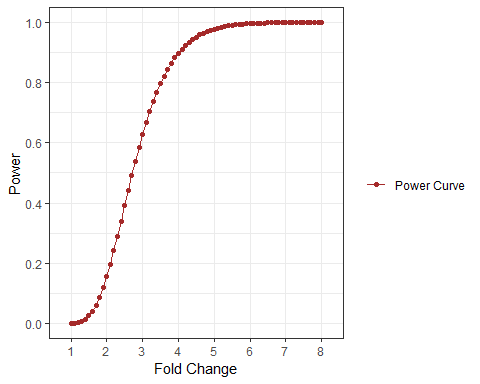
## sample size (ss) vs depth sample size per group  
ssize\_depth\_mont <- sapply(c(med\_mont, quan09\_mont), function(y) {  
 sapply(1:50, function(x) {  
 rnapower(depth = x, cv = y, effect = 2, alpha = 0.05,   
 power = 0.8)  
 })  
})  
ssize\_depth\_mont <- data.frame(ssize\_depth\_mont)  
colnames(ssize\_depth\_mont) <- 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"))



### The above figure shows the required sample size needed to detect a two-fold difference in expression with 80% power at alpha = 0.05. Usually, the depth is greater than 10. Based on the median of the biological CV from those 10 samples, I would recommend 25 samples per group to conduct this study.

### (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(bcv, 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"))



The above figure shows for n = 20 per group, coverage of 100, 60% quantile of BCV and alpha = 0.01. The power is less than 0.2 with two-fold change. But eventually the power can reach 1 with six-fold change.

## 2. Next Generation Sequencing: Pre-Processing

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

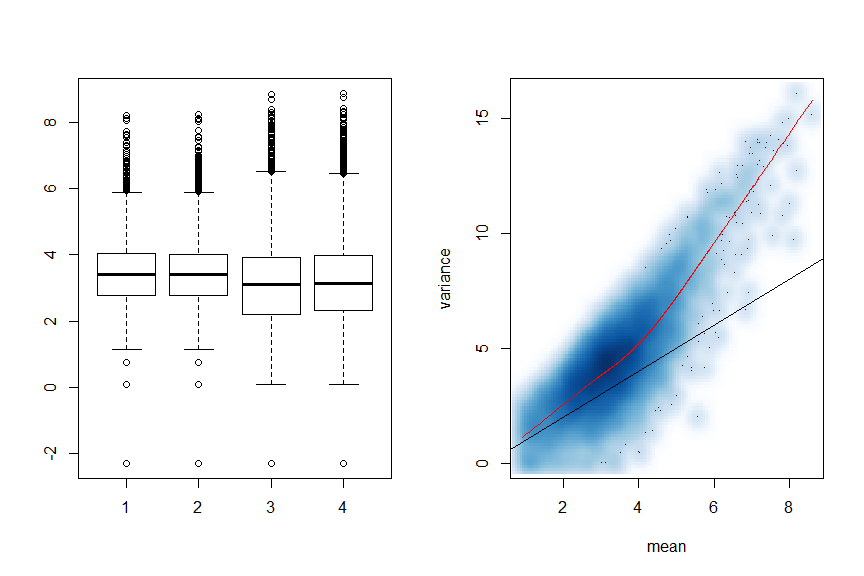
## yeast RNA-seq data set on two mutant and two wildtype  
## strains  
data(geneLevelData)  
## also load the GC content and length  
data(yeastGC)  
data(yeastLength)  
## GC and gene length of montgomery data(uCovar)  
  
## genes have all zeros as counts  
zero\_yeast <- sum(rowSums(geneLevelData) == 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(geneLevelDataFilter))),   
 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.

### 

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

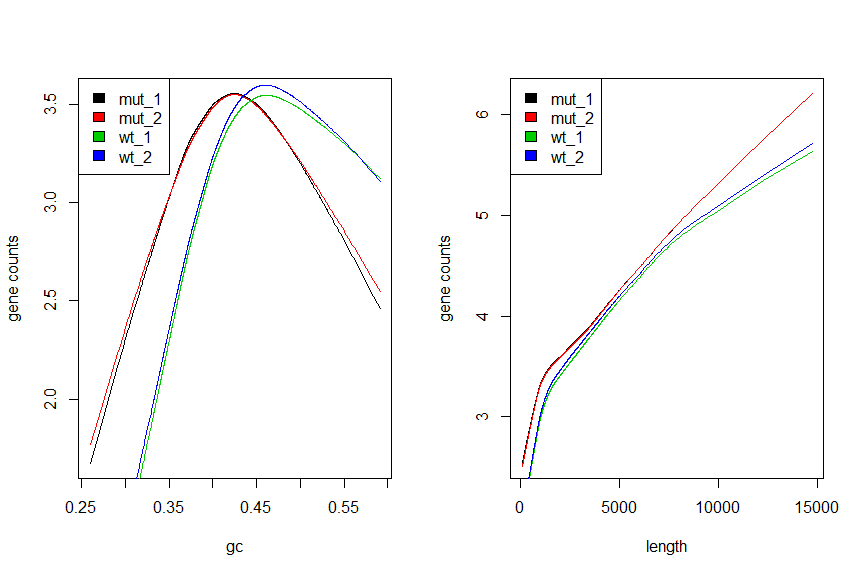
par(mfrow = c(1, 2))  
## To plot the counts by sample  
boxplot(counts)  
  
## plot the mean by variance plot a smoothScatter plot of the  
## mean variance relation a lowess fit  
meanVarPlot(counts, log = T)



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

The above boxplot shows there is an obvious overall intergroup difference on median and variance. The wild type group has a smaller median but a greater variance. Thus, this dataset should be normalized.

The mean by variance plot shows the relation is not linear. Particularly, genes with larger means have greater variance. The dataset shows an overdispersion pattern, suggesting negative binomial model with a dispersion parameter is better than basic Poisson model fit.



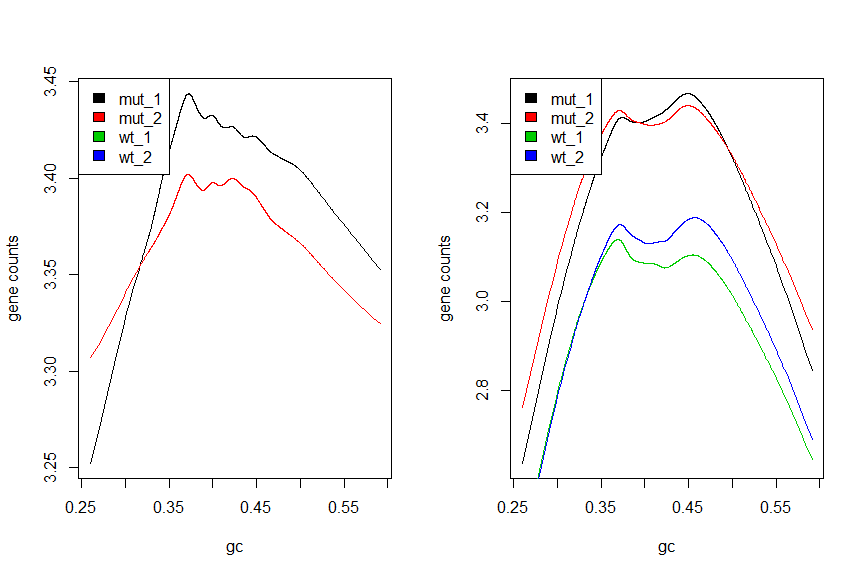
It was mentioned in the paper, read counts depend not only on length but also on sequence features such as GC-content and mappability.

The gc plot also shows a systematic difference between two groups. There is a shift in the peak, as well as the whole distribution. Thus, both median and variance are different. This is evidence of the existence of strong sample-specific GC-content effects on RNA-Seq read counts.

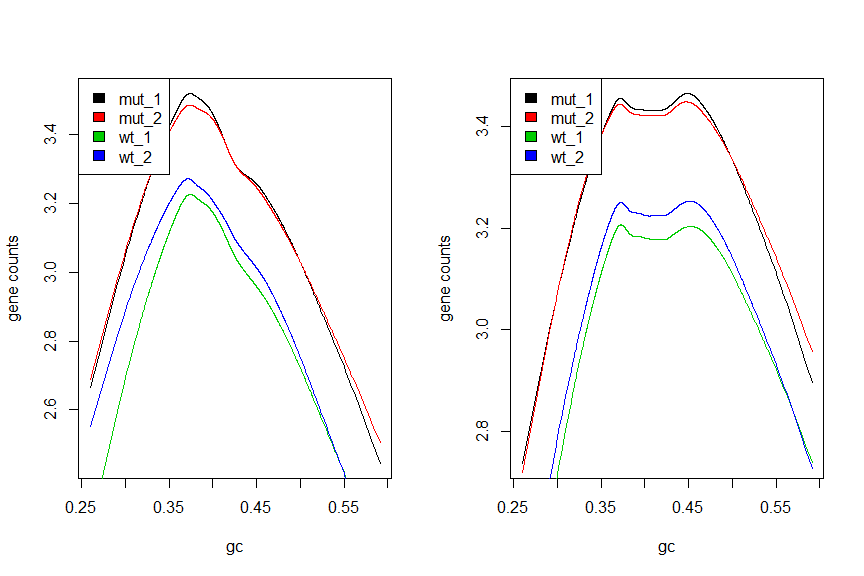
The length plot demonstrates there is a difference of counts on different gene length between two groups. However, for this dataset, the bias introduced by gene length is less than the one introduced by GC-content effects. This plot also shows “mut” group has a larger library size.

### (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)



There are 4 different methods available, which are four normalizations described in Risso et al. (2011). Here, I did the normalization by GC content.

The loess normalization transforms the data by regressing the counts on GC content and subtracting the loess fit from the counts to remove the GC-content effects.

The median, upper and full normalizations are similar. All of them stratify the genes based on the GC content. Particularly, the genes are stratified in 10 strata/bins, by default.

For each bin, different methods work differently:

“Median” scales the data to have the same median in each bin.

“Upper” scales the data to have the same upper in each bin.

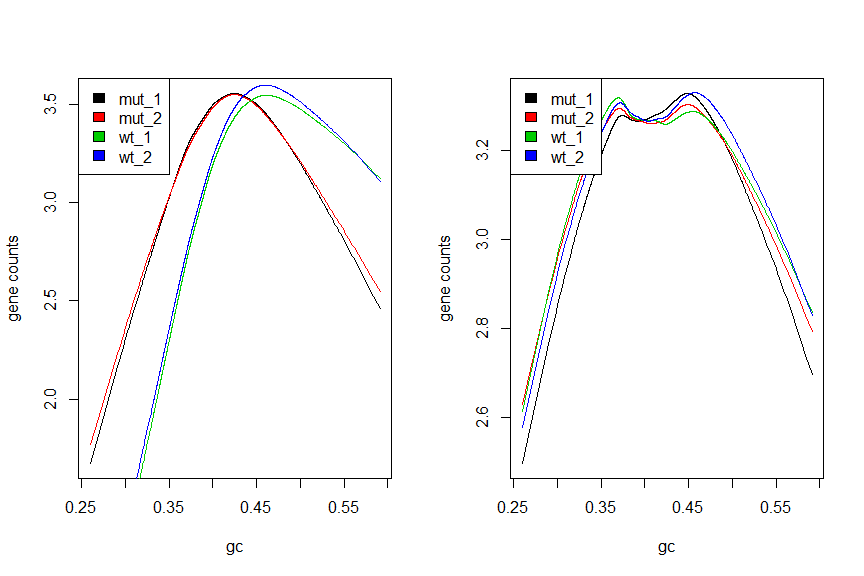
“Full” forces the distribution of each stratum to be the same using a non-linear full quantile normalization. This method is the most aggressive and can be over-normalized.

Basically, the loess method is worst. With normalization, there is still an obvious inter-replicates difference. There are differences not only in scale (median) but also in the location of curves (variance).

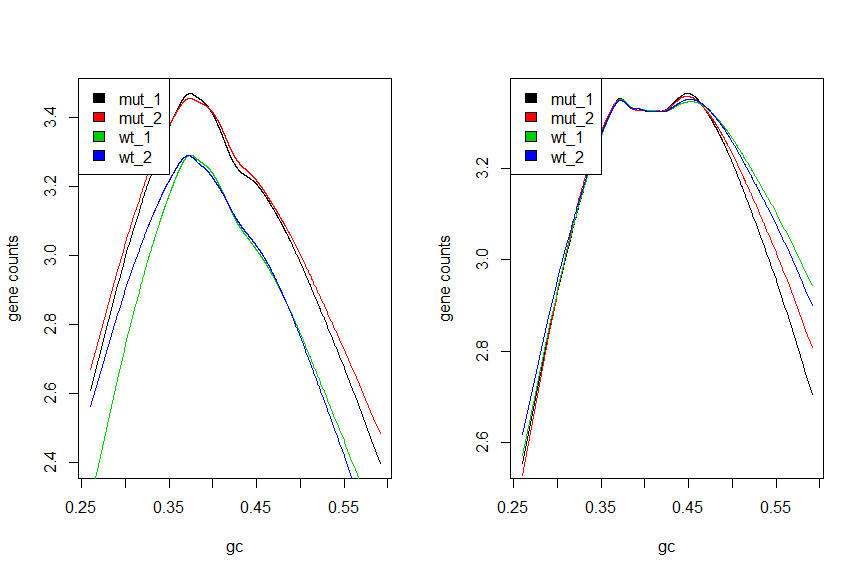
The results from 3 different quantile methods are similar. All these methods removed the difference in shapes of curves (variance). In addition, both the median and full method resulted in two-peaks curves. Now after within-lane normalization by these methods, the only bias left is the difference in the scale (median).

### (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)



There are 3 different methods available: median, upper and full normalizations. For between-lane normalization, there is no stratification.

Similarly, “median” forces the median of each lane to be the same. “Upper” scales the data of each lane to have the same upper quantile. “Full” forces the distribution of each lane to be the same using a non-linear full quantile normalization.

In my analysis, the following between-lane normalization using the same method as the initial within-lane normalization. Generally, the “upper” method only removed the difference in variance. The “median” and “full” method removed the difference both in the variance and median. In addition, it is obvious that the “full” method is more aggressive. Basically, there is no common rule to judge the method and it all depends on the dataset. In term of this yeast dataset, “median” and “full” work better but one should be cautious that the “full” method might be too aggressive.