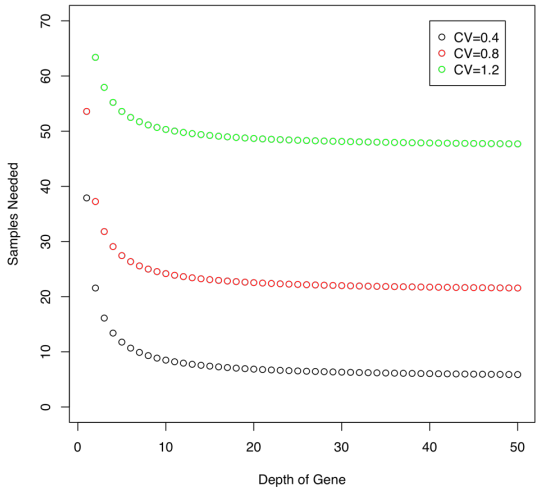
Homework 5 Solutions
BIOS-7659/CPBS-7659

1) Next Generation Sequencing Sample Size Estimates

We will explore the `RNASeqPower` (Hart et al., 2013) method for power and sample size calculations for RNA-Seq data using data from Montgomery et al., 2010 provided in the `cqn` R package of data. We took a subset of 10 samples of lymphoblastoid cell lines.

a) For a given power and type I error rate (alpha), Figure 3 from Hart et al., (2013) shows how many samples are needed for different sequencing depths (the average number of reads per gene) and coefficient of variations (CV), which is the variation between biological replicates. This figure could be recreated but using alpha=0.05, not alpha=0.01 as described in the text. As sequencing depth increases, the number of samples needed decreases and then remains constant. As the biological variation (CV) increases, more samples are needed. After a certain amount of depth, the CV has much more of an effect on the number of samples needed.




```
library(RNASeqPower)
```

```
plot(rnapower(depth = 1:50 , power = .80,  cv = 1.2, alpha = 0.05,
      effect = 2), ylim = c(0,70), col = "green",
      ylab = "Samples Needed",xlab = "Depth of Gene")
points(rnapower(depth = 1:50 , power = .80,  cv = .8, alpha = 0.05,
      effect = 2),  col = "red")
points(rnapower(depth = 1:50 , power = .80,  cv = .4, alpha = 0.05,
      effect = 2),  col = "black")
legend(40, 70, col = c("black","red","green"), legend = c("CV=0.4",
      "CV=0.8", "CV=1.2"), pch = 1 )
```

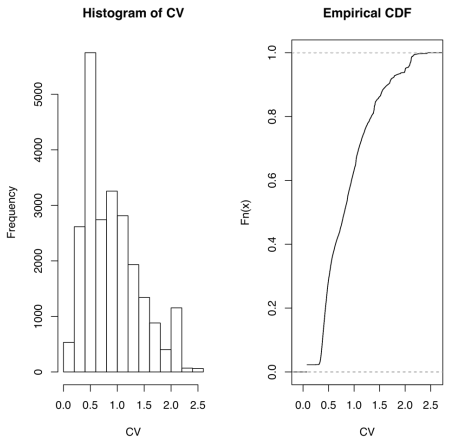
b) Below is a row for the Montgomery data set in the format of Table 1 in Hart et al. (2013). This data set has already been filtered for genes that have at least one count, therefore, the % mapped column in Table 1 is 100%. It is not clear from their manuscript how counts per million were calculated. Here, we aggregated across all samples and divided by the total count for that gene.

Compared to the other data sets, the Montgomery data set has many more genes with smaller counts per million mapped. In particular, the percentage of genes in the range "0.1-1" and "10-100" is much larger or smaller respectively compared to other data sets.

			Counts Per Million Mapped						
Sample	**n**	**Avg Reads**	**<.01**	**0.01-.1**	**0.1-1**	**1-10**	**10-100**	**100-1000**	**>1000**
Lymphoblastoid	10	147.84	0.0	23.0	21.3	18.3	26.7	10.4	0.3

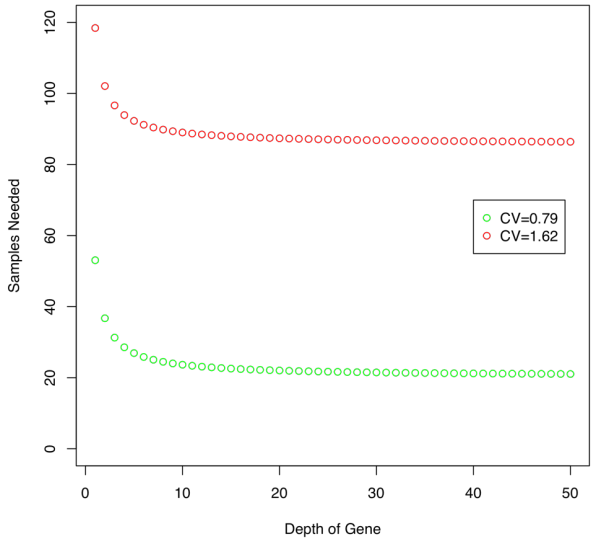
```
library(cqn)
data(montgomery.subset)
N = dim(montgomery.subset)[1]
y = apply(montgomery.subset,1,function(x) sum(x))#counts across samples
n = sum(montgomery.subset) # all counts
cpm = (y/n)*1000000 # counts/million for each gene
round(100*c(sum(cpm<=.01), sum(cpm>.01 & cpm <= .1)/N,
      sum(cpm>.1 & cpm <= 1)/N , sum(cpm> 1 & cpm <= 10)/N,
      sum(cpm> 10 & cpm <= 100)/N, sum(cpm> 100 & cpm <= 1000)/N,
      sum(cpm> 1000 )/N),1)
```

c) The biological coefficient variation was estimated by taking the square root of the tagwise dispersion estimates from edgeR. The empirical cumulative distribution function (CDF) is different than those reported in Figure 2 of Hart et al. (2013). The range for the CVs in the Montgomery data is larger, up to 2.5, while for the reported data sets in the paper, the range is only up to 1.0. The median is 0.79 and 90th percentile is 1.62, which are larger than the other human data sets in Figure 2, where the median and 90% percentile range from 0.2 and 0.75.



```
#Use edgeR for cvs
library(edgeR)
final = montgomery.subset
totalreads = apply(final, 2, sum)
d.mont = DGEList(counts = final, lib.size = totalreads, group =
      rep(c("grp1", "grp2"), each= 5))

tagwise = estimateTagwiseDisp(d.mont)
mont.cvs = sqrt(tagwise$tagwise.dispersion) #take square root
mont.cvmed = median(sqrt(tagwise$tagwise.dispersion)) #median

par(mfrow = c(1,2))
hist(mont.cvs, main = "Histogram of CV", xlab = "CV")
plot(ecdf(mont.cvs), xlab = "CV", main = "Empirical CDF")
```

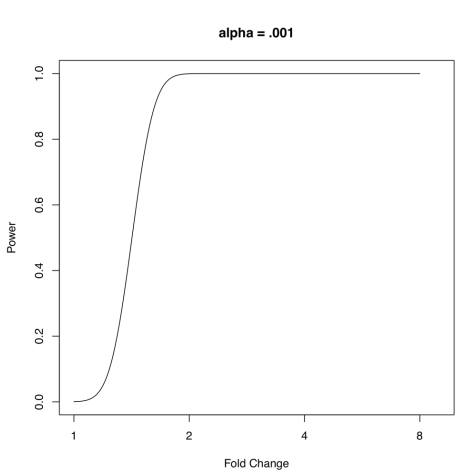
d) Based on the median and 90th percentile CV for the Montgomery data, we recreated Figure 3. For the median and 90th percentile CV, a sample size of 25 and 90 respectively per group is recommended to observe a fold change of 2 with 80% power and 0.05 Type I error. These were the smallest sample sizes, which were not sensitive to the depth of coverage.

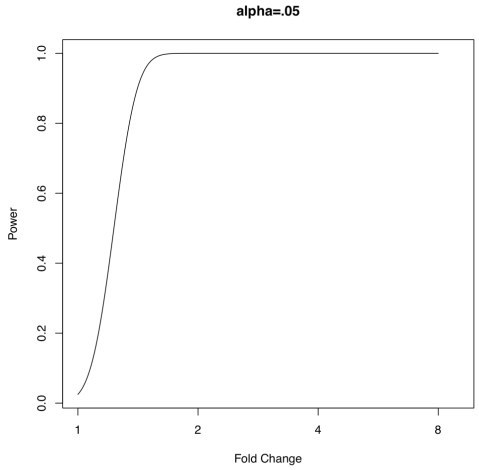



```
plot(rnapower(depth = 1:50 , power = .80,  cv = .79, alpha = 0.05,
      effect = 2), ylim = c(0,120), col = "green",
      ylab = "Samples Needed",xlab = "Depth of Gene")
points(rnapower(depth = 1:50 , power = .80,  cv = 1.62, alpha = 0.05,
      effect = 2),  col = "red")
legend(40, 70, col = c("green","red"), legend = c("CV=0.79",
      "CV=1.62"), pch = 1 )
```

e) There were challenges to recreating the power curve in Figure 4 from Hart et al. (2013). First, in the manuscript the y-axis for the power curve stopped at 0.60, which should be 1.0 since that is the maximum power possible. Second, there

was inconsistent description of the plot; the text and legend stated different parameters. Third, the x-axis is not evenly spaced, the tick marks are powers of two, which can be misleading. Below, two plots based on the descriptions offered in the paper are displayed. It seems that the power curve with alpha = 0.001 is closest to Figure 4 from the paper






```
vals = (100:800)/100

y=rnapower(depth = 100, n=20, cv = .32, alpha = 0.001, effect = vals)
plot(x, y, log="x", type="l", xaxt="n", xlim=c(1,9),
            xlab="Fold Change", ylab = "Power", main = "alpha = .001")
axis(1, at=c(1,2,4,8))

y=rnapower(depth = 100, n=20, cv = .32, alpha = 0.05, effect = vals)
plot(x, y, log="x", type="l", xaxt="n", xlim=c(1,9),
             xlab="Fold Change", ylab = "Power", main = "alpha=.05")
axis(1, at=c(1,2,4,8))
```

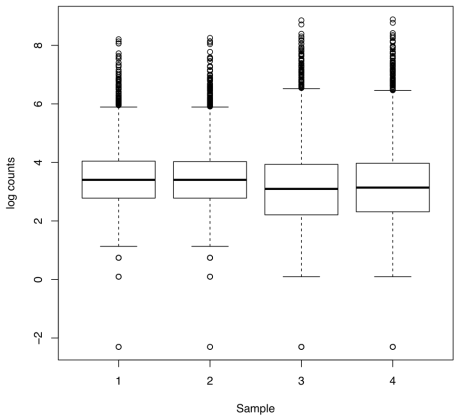
2) Next Generation Sequencing: Pre-Processing

Like other high-throughput data, there are several data QC and pre-processing steps that are recommended before performing statistical modeling on RNA-Seq data. We will explore biases related to GC content and gene length, and try different within and between lane normalization methods.

a) Filtering out genes with low counts is an important first step because many zeros and small counts may cause skewness or violation of distributional assumptions for common tests. In the provided data set, there are 7065 genes. Of those, 557 genes have 0 counts for all samples and 1043 genes have a 0 count for at least one sample. Filtering genes with at least 10 counts per sample, 6083 genes remain.

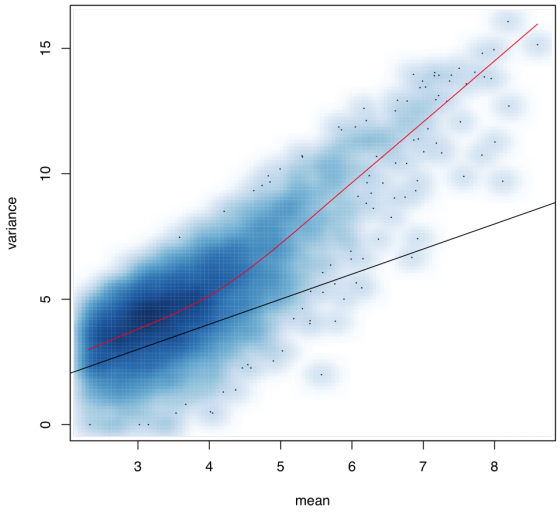
```
Code:
library(yeastRNASeq)
library(EDASeq)

data(geneLevelData)
data(yeastGC)
data(yeastLength)

dim(geneLevelData) #7065
allzero = apply(geneLevelData, 1, function(x) sum(x ==0))
sum(allzero == 4) # 557
sum(allzero>0) #1043

#Filter data with at least 10 counts
keep = rowSums(geneLevelData)
geneLevelDataFilter = geneLevelData[keep>=10,]
geneNamesFilter = row.names(geneLevelDataFilter)[keep >=10]

#Use code in assignment to create SeqExpressionSet object: 5988 genes
#with annotation
exprs = as.matrix(geneLevelDataFilter)
sub = intersect(rownames(geneLevelDataFilter), names(yeastGC))
exprs = exprs[sub,]
row.names(exprs) = NULL
colnames(exprs) = NULL
counts = newSeqExpressionSet(counts=exprs,phenoData=data.frame(conditions =
colnames(geneLevelDataFilter)),
        featureData=AnnotatedDataFrame(data.frame(gc=yeastGC[sub], length =
yeastLength[sub])))
```

b) The boxplot of the log counts by sample indicate that there is a need for normalization, since samples 3 and 4 (wild type) have larger variance and lower median than samples 1 and 2 (mutant).




```
Code:
boxplot(counts, xlab = "Sample", ylab = "log counts")
```

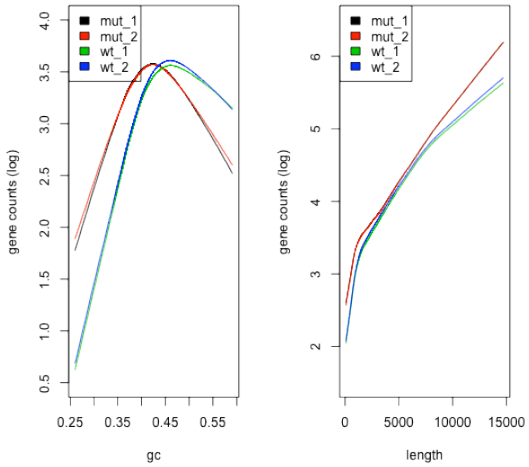
The mean by variance plot shows a smooth fit for the mean-variance relationship (red line) and a line for the mean equal to the variance (black line). The plot indicates that the variance is larger than the mean especially for large means. This is a violation of the Poisson assumption that the variance equals mean and suggests that the negative binomial or other distribution that accounts for over-dispersion may be appropriate.




```
Code:
meanVarPlot(counts, log = TRUE)
```

The plots for log counts by GC content indicate a quadratic shape for the plot, where the peak of log counts is centered around 0.40 to 0.45 GC content, but is lower for values outside this range. In addition, the wildtype samples have a shift towards larger GC content, while the mutant samples have a bias for lower GC content. The plot for log counts by length indicates that all samples have more counts for longer genes, which is expected. However, the wildtype samples have lower log counts than the mutant for a given length and a smaller slope for longer

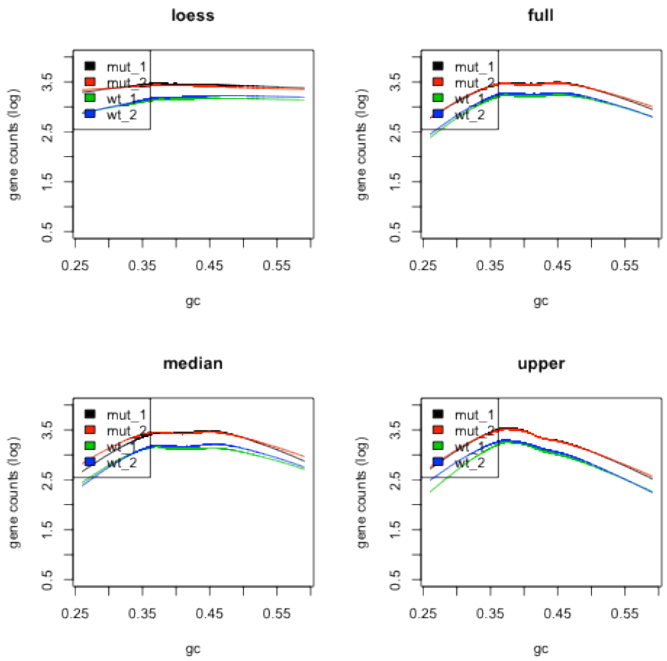
length. These plots support the need for GC content or length based normalization.




```
Code:
par(mfrow = c(1,2)
biasPlot(counts, "gc", log = TRUE, ylim = c(.5,4))
biasPlot(counts, "length", log = TRUE, ylim = c(1.5,6.5))
dev.off()
```

c) Within-lane normalization can be used to normalize each lane by features such as GC content and gene length. The “median” method bins the counts by the feature, counts are sorted within bins and then the counts are scaled within each bin to have the same median across bins. The “upper” method is similar to the median method but the counts are scaled within each bin to have the same upper quartile across bins. The “full” method, uses the quantile normalization method developed for microarrays, but now the counts within each bin (not array sample) are forced to have the same distribution. Finally, the “loess” method performs a locally weighted scatterplot smoothing of counts on the feature.

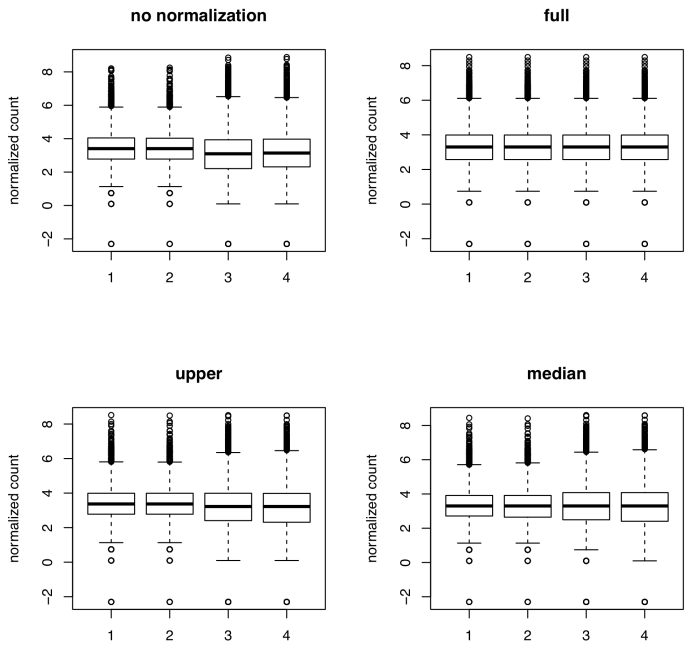
Below, results of within-lane normalization are displayed by GC content for each of the methods. All methods result in similarly shaped curves by sample, which corrects for the bias observed in the raw count data for shifted GC values in the wildtype (part b). Loess is the most aggressive method for correction since the lines are the flattest across GC content. Full quantile and median methods are quite similar with decreased counts for small and large GC content. The upper method has some decrease for small GC content but shows a more dramatic decreasing pattern for large GC content. However, for all methods, the lower counts in the wildtype is still apparent since the lines are always lower.




```
Code:
par(mfrow = c(2,2))
fl = withinLaneNormalization(counts, y = "gc",  which = "loess")
biasPlot(fl, "gc", log = TRUE, ylim = c(.5,4), main = "loess")
fl = withinLaneNormalization(counts, y = "gc",  which = "full")
biasPlot(fl, "gc", log = TRUE, ylim = c(.5,4), main = "full")
fl = withinLaneNormalization(counts, y = "gc",  which = "median")
biasPlot(fl, "gc", log = TRUE, ylim = c(.5,4), main= "median")
fl = withinLaneNormalization(counts, y = "gc",  which = "upper")
biasPlot(fl, "gc", log = TRUE, ylim = c(.5,4), main = "upper")
```

d) Between-lane normalization is used to adjust for sequencing depth and other biases across lanes. The “median” method forces the median to be the same across all lanes. The “upper” method forces the upper quartile to be the same across all lanes. The “full” method uses quantile normalization to forces the counts to have the same distribution across lanes.

The results of between-lane normalization following the loess within-lane normalization are displayed below. The boxplots indicate that the full quantile method is the most aggressive in standardizing the medians and variances across lanes. In the upper and median method, the median in the wildtype samples (3 & 4) are still lower and the variance is larger. Using the full quantile method, the wildtype samples have similar median and inter-quartile range (IQR) as the mutant samples.




```
Code:
boxplot(counts, main = "no normalization")
fq = betweenLaneNormalization(fl, which = "full")
boxplot(fq, main = "full")
fq = betweenLaneNormalization(fl, which = "upper")
boxplot(fq, main = "upper")
fq = betweenLaneNormalization(fl, which = "median")
boxplot(fq, main = "median")
```