# Homework 8

*Chandrima Bhattacharya*

*7 March 2019*

```r
suppressPackageStartupMessages(library(DESeq2))
```

```
## Warning: package 'GenomicRanges' was built under R version 3.4.4

## Warning: package 'SummarizedExperiment' was built under R version 3.4.3

## Warning: package 'matrixStats' was built under R version 3.4.4
```

```r
setwd("F:/Annie/CornellMS/Semester2/ANGSD/ANGSD/")
```

## Question 1

Read count defines the number of reads or fragments overlapping with the union of exons of a gene. Properties of read counts are influenced by the follow.

- They are influenced by the sequencing depth. Usually, deeper sequencing gives higher expression value.
- The length of the gene and GC-bias associated with them are also responsible for difference in read counts value.
- The RNA-composition of the different individual gene abundances are responsible for affecting read counts. Highly-expressed transcripts usually can sometimes dominate reducing the reads count available for the rest of the transcripts.

Correction techniques for the following is defined below.

- Sequencing depth needs to be corrected when we are comparing the same gene between different samples. If we are using DESeq2, we can use one of it's function *estimateSizeFactors()* to correct the depth. It uses multiple statistical normalization steps for the following. There are other functions which are defined also for the same!
- The gene length and the GC% in the transcript sequence needs to be corrected while comparing different genes. For this we can use TPM or Transcript per Million normalization technique. TPM normalizes for gene length first, and then normalize for sequencing depth second, and hence performs better than RPKM.
- Individual gene abundances needs to be corrected when we are comparing the same gene between different samples. *rlog* is a DESeq2 function which does that by transforming the the count data to the log2 scale in a way which minimizes differences between samples for rows with small counts, and which normalizes with respect to library size.

## Question 2

```
?rlog()
?DESeqDataSetFromMatrix
```

The input format for rlog() is given below.

```
rlog(object, blind = TRUE, intercept, betaPriorVar, fitType = "parametric")
```

The input format for DESeqDataSetFromMatrix() is given below.

```
DESeqDataSetFromMatrix(countData, colData, design, tidy = FALSE,
  ignoreRank = FALSE, ...)
```

To view S4 objects, we can use showClass as follows.

```r
DESeq.rlog <- readRDS('rlogData.rds')
DESeqDS <- readRDS('DESeq.rds')
showClass(class (DESeq.rlog))
```

```
## Class "DESeqTransform" [package "DESeq2"]
##
## Slots:
##
## Name:                      rowRanges                        colData
## Class: GenomicRanges_OR_GRangesList                       DataFrame
##
## Name:                         assays                          NAMES
## Class:                        Assays               character_OR_NULL
##
## Name:                elementMetadata                       metadata
## Class:                     DataFrame                           list
##
## Extends:
## Class "RangedSummarizedExperiment", directly
## Class "SummarizedExperiment", by class "RangedSummarizedExperiment", distance 2
## Class "Vector", by class "RangedSummarizedExperiment", distance 3
## Class "Annotated", by class "RangedSummarizedExperiment", distance 4
```

```r
showClass(class (DESeqDS))
```

```
## Class "DESeqDataSet" [package "DESeq2"]
##
## Slots:
##
## Name:                         design             dispersionFunction
## Class:                       formula                       function
##
## Name:                      rowRanges                        colData
## Class: GenomicRanges_OR_GRangesList                       DataFrame
##
## Name:                         assays                          NAMES
## Class:                        Assays               character_OR_NULL
##
## Name:                elementMetadata                       metadata
## Class:                     DataFrame                           list
##
## Extends:
## Class "RangedSummarizedExperiment", directly
## Class "SummarizedExperiment", by class "RangedSummarizedExperiment", distance 2
## Class "Vector", by class "RangedSummarizedExperiment", distance 3
## Class "Annotated", by class "RangedSummarizedExperiment", distance 4
```

### a

DESeqDataSetFromMatrix() helps perform transformations and exploratory data analysis. The rlog() transform uses a an empirical Bayesian prior on inter-sample differences in the form of a ridge penalty. It uses log2scale. The DESeqDataSetFromMatrix creates a DESeq object from a raw counts matrix and colData. The rlog() function transforms the count data.

Similarities:
* Both are S4 objects. They also have associated slots. * They are DESeq2 function and thus creates DESeq objects as output.

Differences: * rlog() uses a logarithmic transformation whereas DESeqDataSetFromMatrix() doesn't. * The function rlog() returns a SummarizedExperiment object which contains the rlog-transformed values in its assay slot. Also it needs SummarizedExperiment object as input. DESeqDataSetFromMatrix() can be used in cases where the data not in the form of a SummarizedExperiment object, but either as a simple matrix of count values or as output files.

### b

The assay function will help us extract the expression values stored in either object.
The function rlog returns a SummarizedExperiment object which contains the rlog-transformed values in its assay slot.

```r
head(assay(DESeq.rlog))
```

```
##                  SNF2_1      SNF2_2      SNF2_3      SNF2_4      SNF2_5
## YAL012W   12.362790082 12.56467131 12.62759344 12.41684200 12.59001236
## YAL068C    0.005301111  0.01623866  0.01653841 -0.01798827 -0.03567399
## YAL067C    5.676560939  5.39000672  5.31347813  5.56282559  5.53604197
## YAL066W   -1.850419670 -1.87158639 -1.87154934 -1.87322146 -1.87036008
## YAL065C    2.228386441  2.35749366  2.28606049  2.18032395  2.16684913
## YAL064W.B  3.050999694  3.00284519  3.04880810  2.97013126  2.99244001
##                   WT_1        WT_2       WT_3        WT_4        WT_5
## YAL012W   12.65751928 12.14632332 12.1393616 12.32283797 12.27003301
## YAL068C   -0.03157676 -0.03622162 -0.0341577  0.01501276  0.02605988
## YAL067C    5.04963780  5.09250818  5.1448729  5.11699987  5.20977955
## YAL066W   -1.86812547 -1.87067489 -1.8695124 -1.87173887 -1.87041411
## YAL065C    2.51285290  2.28137504  2.2118784  2.23089053  2.22855898
## YAL064W.B  3.15824928  3.12911859  2.9588491  2.90451310  3.04307338
```

```r
head(assay(DESeqDS))
```

```
##           SNF2_1 SNF2_2 SNF2_3 SNF2_4 SNF2_5 WT_1 WT_2 WT_3 WT_4 WT_5
## YAL012W     7351   7180   7648   8119   5944 4312 3767 3040 5604 4167
## YAL068C        2      2      2      1      0    0    0    0    2    2
## YAL067C      103     51     44     90     53   12   23   21   30   29
## YAL066W        2      0      0      0      0    0    0    0    0    0
## YAL065C        5      9      6      3      1   10    5    2    4    3
## YAL064W.B     13      8     10      9      6    9   12    4    4    8
```

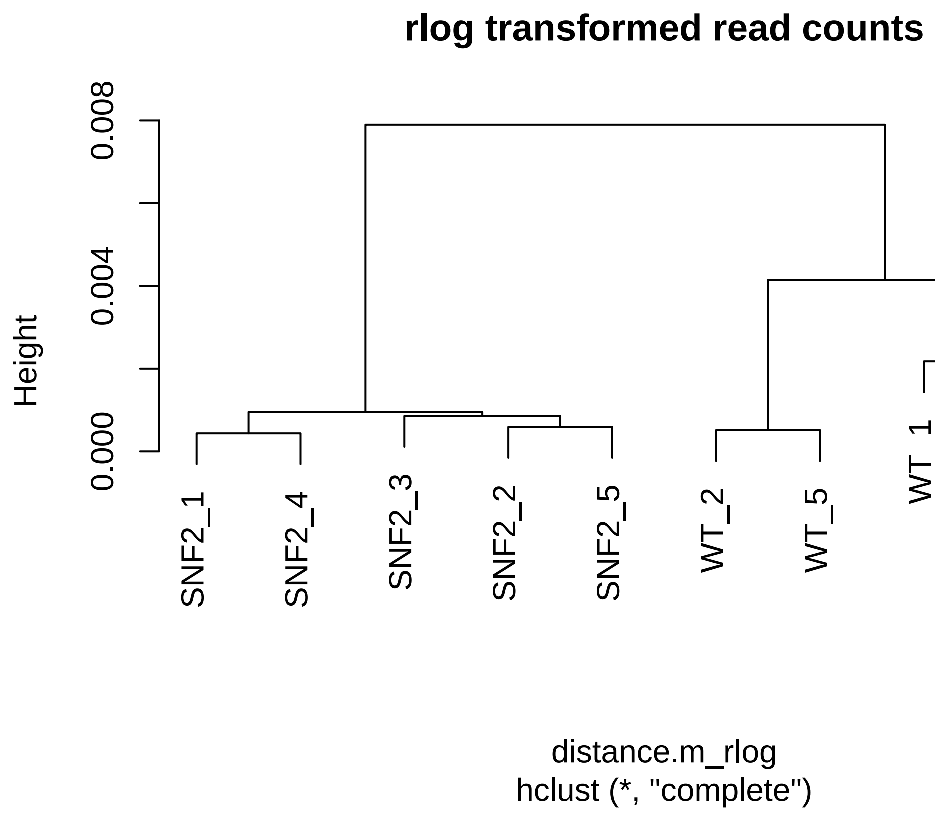
**c**

I have added a new matrix to my DESeq object under the metadata slot. Now my_normalization can be called.

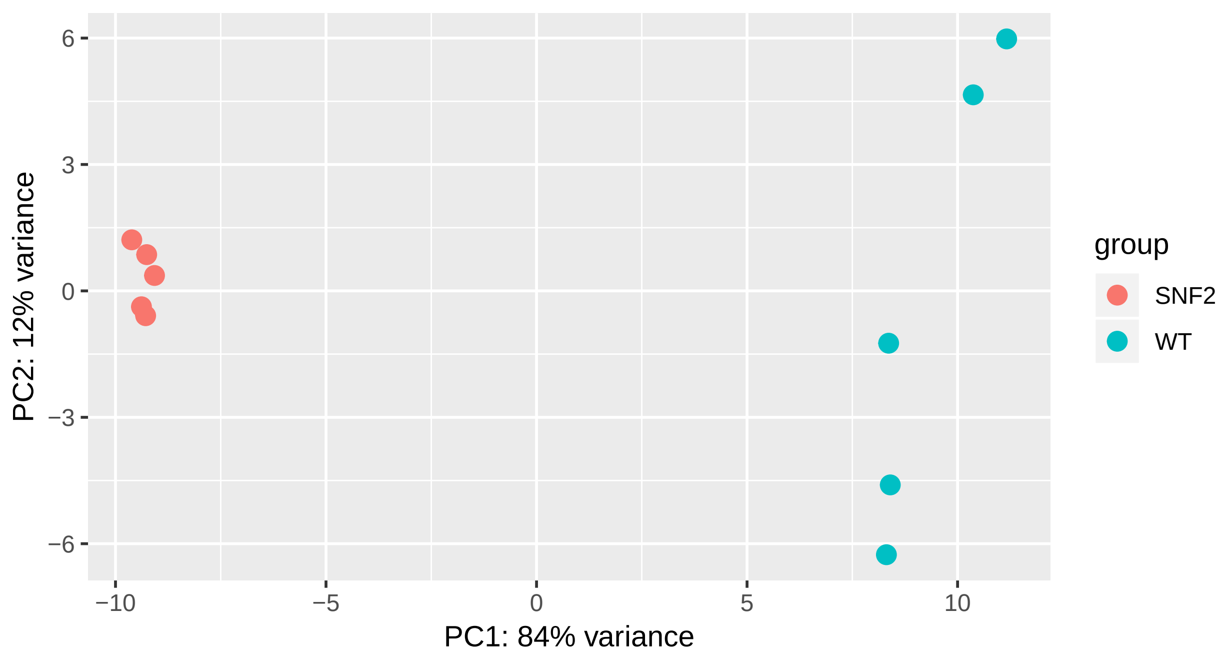
```r
DESeq.rlog@metadata$my_normalization <- assay(DESeq.rlog)
```

**Question 3**

```r
# Dendrogram
rlog.norm.counts <- readRDS("rlog.norm.counts.rds")
pearson_cor <- cor(rlog.norm.counts, method = "pearson")
distance.m_rlog <- as.dist(1-pearson_cor)
plot(hclust(distance.m_rlog),labels=colnames(rlog.norm.counts),main="rlog transformed read counts")
```



```r
# PCA plot
plotPCA(DESeq.rlog, intgroup=c("condition"))
```



Both the dendrogram and PCA shows clear data trend. We see that the knock outs cluster together as they are similar gene-knockouts. The wild-type is seen in the different side of the graph but are more spread out. We can see clustering of wild type 2,5 and 3,4 which is represented also by dendogram. The WT and KO conditions are definitely distinct from each other which is why they are spread out so distantly. This is confirmed in the sample distance heatmap in the pcaExplorer.

PCA Explorer allows to view top and bottom loadings plot. I think the various Principal Components which get formed while PCA is shown in the diagram.

```
#PCA Explorer
pcaExplorer::pcaExplorer(dds=DESeqDS ,dst=DESeq.rlog)
```