BIOL\*3300 Lab9 F21

**Normalizing and Transforming Read Counts**

# Background review

In the last lab, we performed the RNAseq read alignment analysis with a replicate in the Mov10 overexpression group using STAR, a splice-aware alignment tool. After we have our reads aligned to the genome, we also counted how many reads have been mapped to each gene with featureCounts program. Our final output was 2 columns file in which each row represents a gene (or feature) and its corresponding counts.

In our dataset, we have two groups (control and Mov10oe) and in this lab we want to assess the difference in expression between these groups on a gene-by-gene basis using R software.

# Statistical models in R

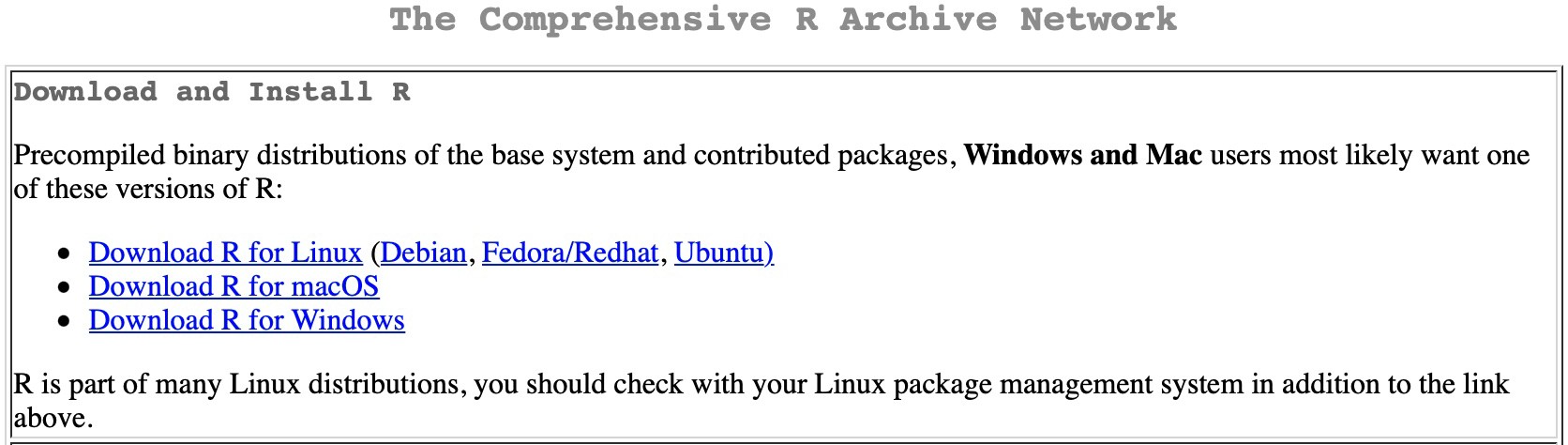
R is a software environment for statistical computing and graphics. R is widely used in the field of bioinformatics, amongst various other disciplines. It can be locally installed on almost all operating systems (and it's free!), with numerous packages available that help in increasing efficency of data handling, data manipulation and data analysis.

R is a powerful language that can be very useful for NGS data analysis, and there are many popular packages for working with RNA-Seq count data. Some of these packages include edgeR, DESeq2, and limma-voom. All of these tools use statistical modeling of the count data to test each gene against the null hypothesis and evaluate whether or not it is significantly differentially expressed.

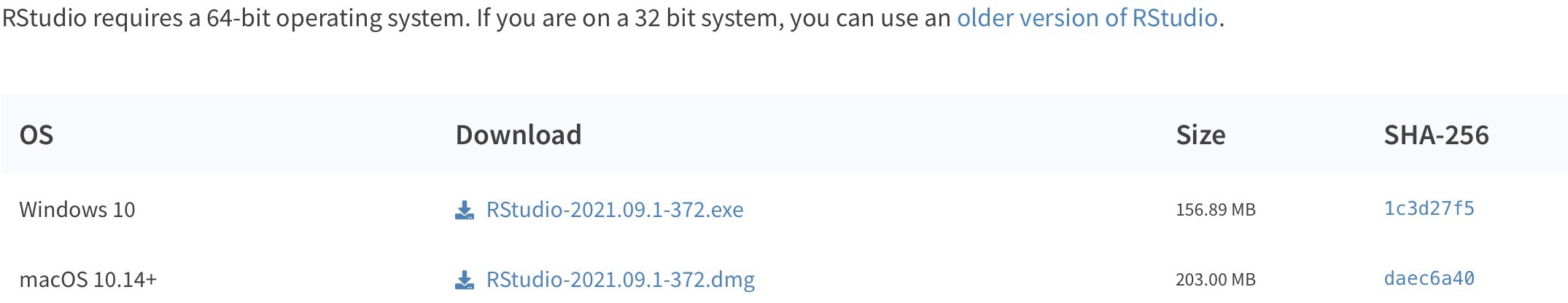
## Setup Your RStudio

RStudio is a separate open-source project that brings many powerful coding tools together into an intuitive easy-to-learn interface. RStudio runs in all major platforms (Windows, Mac, Linux) and through a web browser (using the server installation).

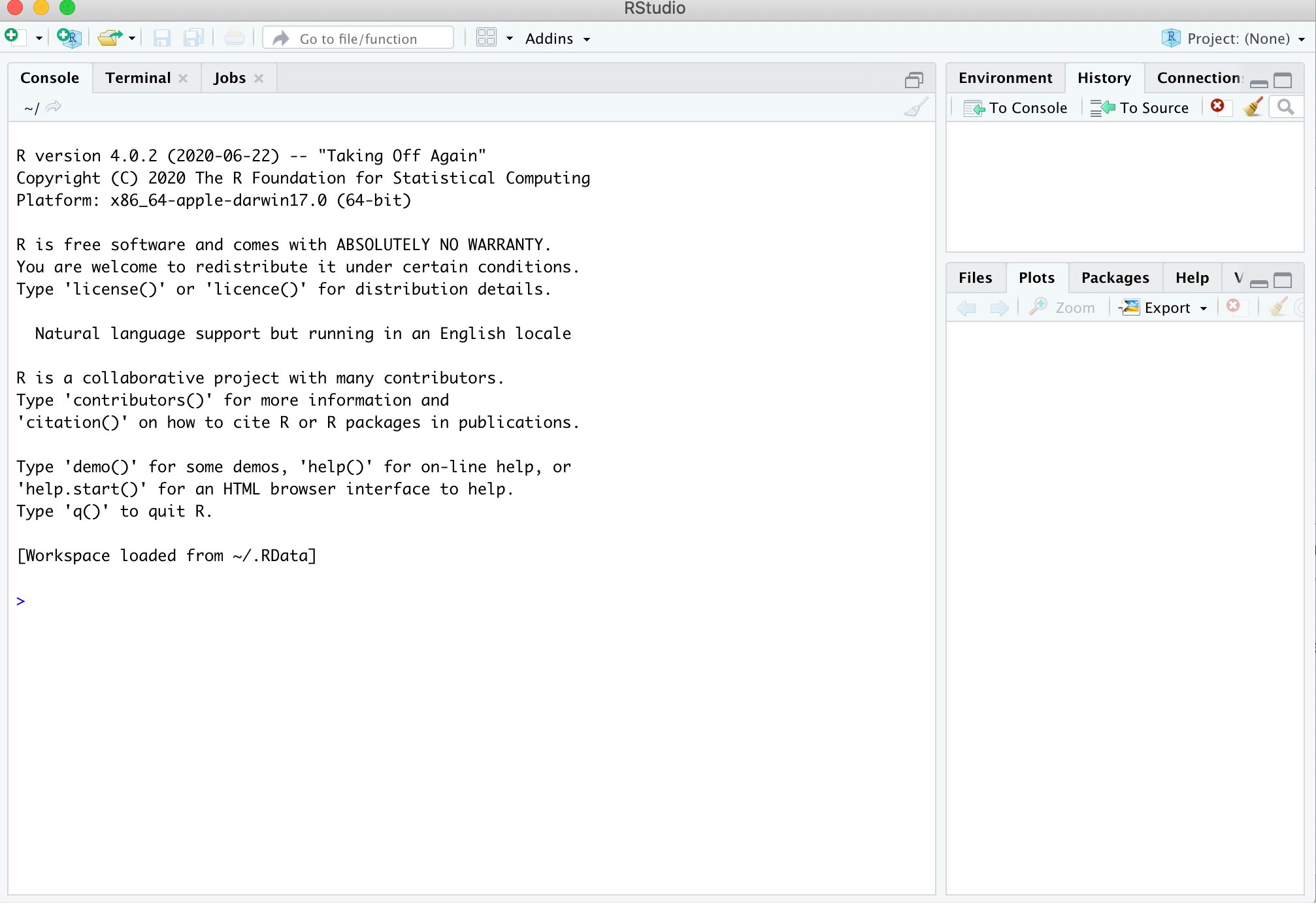
Step1: Install R [https://cran.rstudio.com](https://cran.rstudio.com/)



Step2: Install RStudio <https://www.rstudio.com/products/rstudio/download/#download>



Desktop Version:



RStudio on initial startup has three main components: the **Console** is where you can type commands and see output. An **Environment** tab show all the active objects. The **History** tab shows a list of commands used so far. The files tab shows all the **files** and folders in your default workspace as if you were on a PC/Mac window. The **plots** tab will show all your

graphs. The **packages** tab will list a series of packages or add-ons needed to run certainprocess. For additional info see the **help** tab.

When you launch R, you will be greeted with a prompt (>) and a blinking cursor: **>**

* In this tutorial, copy the commands or codes after the **">"** sign in your Console.
* In your Console, if your commands or codes have no responds, please press "control+ c" at the same time to quit the process.
* Contents after **"#"** are the annotations for R commands. There is no need to copy to theConsole but RStudio won’t report errors if you do.

## Getting the data

We will now use the output file (**RNA\_seq\_featurecounts.txt** in Courselink lab 10 folder already) based on 2 groups (control and Mov10oe) of 6 samples. The featureCounts file was generated by the following command:

Note all the 6 aligned bam files should be created with STAR, so then you can use

\*Aligned.sortedByCoord.out.bam wildcard.

\*\* DO NOT RUN \*\* featureCounts -s 2 \

-a ./genome/chr1-hg19\_genes.gtf \

-o ./results/RNA\_seq\_featurecounts.txt \

./results/\*Aligned.sortedByCoord.out.bam

## Importing the data to R

First, download the file **RNA\_seq\_featurecounts.txt** from courselink. Create a Lab10 folder on your desktop and move your downloaded file into Lab10 folder.

For MAC users:

# get the table of read counts by indicating the path to the file

* readcounts <- read.table("~/Desktop/Lab10/RNA\_seq\_featurecounts.txt",header=TRUE)

For Windows users:

* getwd()

[1] "C:/Users/KellerLb/Documents"

Note here KellerLb should be your **user name** and remember to **substitute KellerLb with your user name** in the following command lines.

TRUE)

# get the table of read counts by indicating the path to the file

* readcounts <- read.table("C:/Users/KellerLb/Desktop/Lab10/RNA\_seq\_featurecounts.txt",header =

Now we are finally in the same page:

# One of the requirements of the assay () slots is that the row.names # correspond to the gene IDs and the col.names to the sample names

* row.names(readcounts) <- readcounts$Geneid

# In addition , we need to exclude all columns that do not contain read counts

* readcounts <- readcounts[,-c(1:6)]

# give meaningful sample names - this can be achieved via numerous approaches # the one shown here is the least generic and most error - prone one!

* names(readcounts) <- c("Control\_1","Control\_2","Control\_3","Mov10\_1","Mov10\_2","Mov10\_3")

# ALWAYS CHECK YOUR DATA AFTER YOU MANIPULATED IT!

* str(readcounts)
* head(readcounts,n = 3)

# make a data.frame with meta-data where row.names should match the individual # sample names

* sample\_info <- data.frame(condition = gsub("\_[0-9]+"," ",names(readcounts)), row.names = names(readcounts))
* sample\_info

condition Control\_1 Control Control\_2 Control Control\_3 Control Mov10\_1 Mov10

Mov10\_2 Mov10

Mov10\_3 Mov10

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Control\_1 | Control\_2 | Control\_3 | Mov10\_1 | Mov10\_2 | Mov10\_3 |
| WASH7P | 3 | 1 | 2 | 2 | 0 | 0 |
| FAM138F | 0 | 0 | 0 | 0 | 0 | 0 |
| FAM138A | 0 | 0 | 0 | 0 | 0 | 0 |

## Normalizing and Transforming Read Counts

The numbers (or estimates) of reads overlapping with a given gene cannot be directly interpreted as absolute proxies of individual gene expression levels. The value that is obtained for a single gene in a single sample is based on the number of reads corresponding to that gene (or transcript), but there are numerous factors that influence the efficiency of amplification and sequencing of DNA fragments (e.g. GC-bias).

In order to compare the gene expression between two conditions, we must calculate the fraction of reads assigned to each gene relative to the total number of reads and with respect to the entire RNA repertoire, which may vary drastically from sample to sample. The purpose of normalization is to eliminate systematic effects that are not associated with the biological differences of interest.

# IF NEEDED, install DESeq2,which is not available via install.packages(), # but through bioconductor

* if (!requireNamespace("BiocManager", quietly = TRUE)) install.packages("BiocManager")
* BiocManager::install("DESeq2")
* library(DESeq2)

# generate the DESeqDataSet

* DESeq. <- DESeqDataSetFromMatrix(countData = readcounts,

colData = sample\_info, design = ~ condition)

# test what counts () returns

* counts(DESeq.ds)

Control\_1 Control\_2 Control\_3 Mov10\_1 Mov10\_2 Mov10\_3

# remove genes without any counts

* DESeq.ds <- DESeq.ds[rowSums(counts(DESeq.ds))>0,]

# investigate different sample sizes

* colSums(counts(DESeq.ds)) Control\_1 Control\_2 Control\_3

164087 140232 107047

Mov10\_1 Mov10\_2 Mov10\_3

224925 205092 125930

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| WASH7P 3 1 | 2 | 2 |  | 0 | 0 |
| LOC729737 0 0 | 1 | 1 |  | 1 | 0 |
| LOC100133331 1 0 | 0 | 0 |  | 0 | 1 |
| ... |  |  |  |  |  |
| SH3BGRL3 0 0 | 0 | 0 |  | 1 | 0 |
| DHDDS 0 1 | 0 | 0 |  | 0 | 1 |
| [ reached getOption("max.print") | -- omitted | 773 rows | ] |  |  |

DESeq2’s default method to normalize read counts to account for differences in sequencing depths is implemented in estimateSizeFactors().

# calculate the size factor and add it to the data set

* DESeq.ds <- estimateSizeFactors(DESeq.ds)
* sizeFactors(DESeq.ds)

Control\_1 Control\_2 Control\_3 Mov10\_1 Mov10\_2 Mov10\_3 1.1744444 1.0046460 0.7943334 1.2825146 1.1775918 0.7454880

# if you check colData () again , you see that this now contains the sizeFactors

* colData(DESeq.ds)

DataFrame with 6 rows and 2 columns condition sizeFactor

<factor> <numeric> Control\_1 Control 1.174444

Control\_2 Control 1.004646

Control\_3 Control 0.794333

Mov10\_1 Mov10 1.282515

Mov10\_2 Mov10 1.177592

Mov10\_3 Mov10 0.745488

# counts () allows you to immediately retrieve the normalized read counts

* counts.sf\_normalized <- counts(DESeq.ds, normalized = TRUE)

## Log2 transformation of read counts

Due to the relatively large dynamic range of expression values that RNA-seq data can cover, many downstream analyses (including clustering) work much better if the read counts are transformed to the log scale following normalization. While you will occasionally see log10 transformed read counts, log2 is more commonly used because it is easier to think about doubled values rather than powers of 10. The transformation should be done in addition to sequencing depth normalization.

# transform size - factor normalized read counts to log2 scale using a # pseudocount of 1

* log.norm.counts <- log2(counts.sf\_normalized + 1)

You can see how the log2 transformation makes even simple graphs more easily interpretable by generating boxplots of read counts.

# to plot the following two images next to each other

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

# first, boxplots of non-transformed read counts (one per sample )

* boxplot(counts.sf\_normalized,

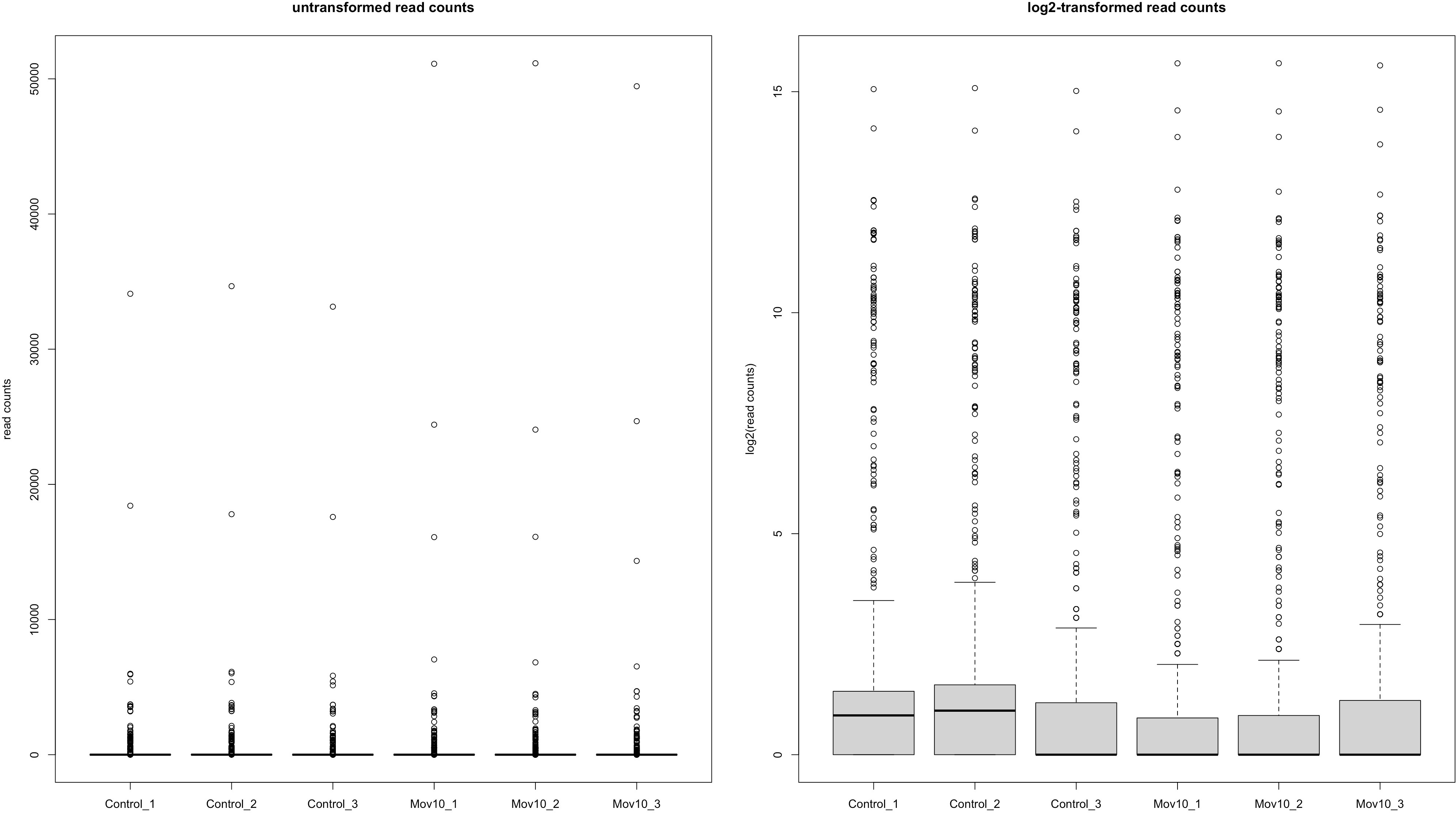
main = "untransformed read counts", ylab = "read counts")

# box plots of log2-transformed read counts

* boxplot(log.norm.counts,

main = "log2-transformed read counts", ylab = "log2(read counts)")

You will see the two boxplots like the following:



## Principal Components Analysis (PCA)

A complementary approach to determine whether samples display greater variability between experimental conditions than between replicates of the same treatment is principal components analysis. It is a typical example of dimensionality reduction approaches that have become very popular in the field of machine learning. The goal is to find groups of features (e.g., genes) that have something in common (e.g., certain patterns of expression across different samples), so that the information from thousands of features is captured and

represented by a reduced number of groups.

# obtain regularized log - transformed values

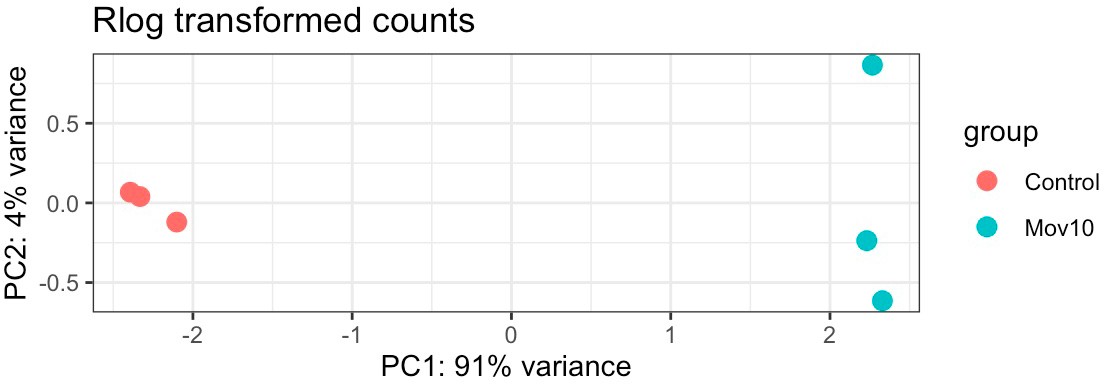
* DESeq.rlog <- rlog(DESeq.ds,blind = TRUE)

# DESeq2 also offers a convenience function based on ggplot2 to do PCA directly on a DESeqDataSet:

* install.packages("ggplot2")
* library(ggplot2)

# PCA

* P <- plotPCA(DESeq.rlog)
* P <- P + theme\_bw() + ggtitle("Rlog transformed counts")
* print(P)



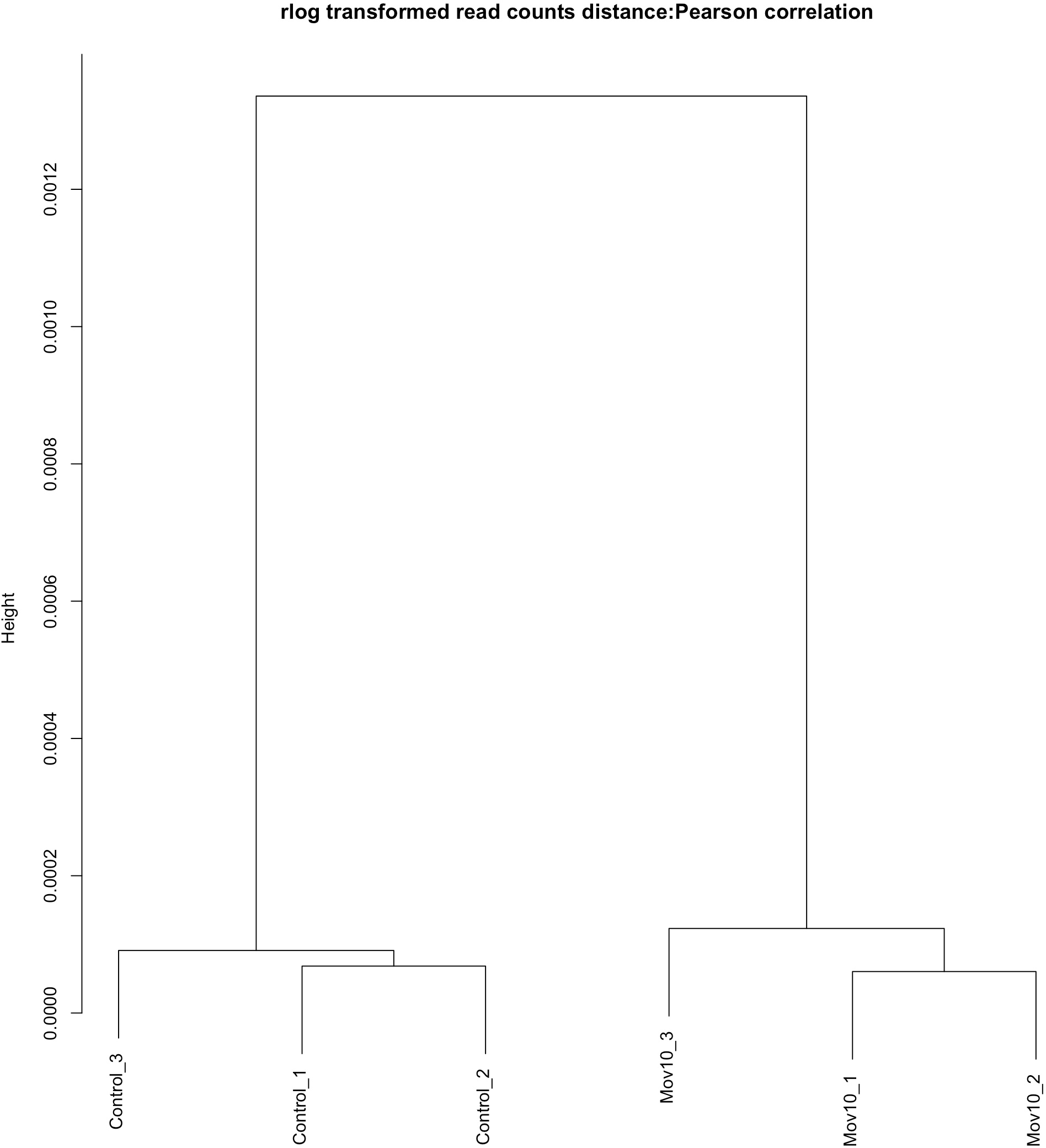
## Hierarchical clustering

To determine whether the different sample types can be separated in an unsupervised fashion (i.e., samples of different conditions are more dissimilar to each other than replicates within the same condition), hierarchical clustering can be used. Hierarchical clustering is typically based on pairwise comparisons of individual samples, which are grouped into "neighborhoods" of similar samples. The basis of hierarchical clustering is therefore a matrix of similarity metrics (which is different from the actual gene expression values!).

A common way to assess the (dis)similarity is the Pearson correlation coefficient, r, that we just described. The corresponding distance measure is d = 1 − r. Alternatively, the Euclidean distance is often used as a measure of distance between two vectors of read counts. The Euclidean distance is strongly influenced by differences of the scale: if two samples show

large differences in sequencing depth, this will affect the Euclidean distance more than the distance based on the Pearson correlation coefficient.

The result of hierarchical clustering is a dendrogram (Figure 22); clusters are obtained by cutting the dendrogram at a level where the jump between two consecutive nodes is large: connected components then form individual clusters. It must be noted that there is no consensus on how to decide the "correct" number of clusters.



# cor () calculates the correlation between columns of a matrix

* distance.m\_rlog <- as.dist(1 - cor(rlog.norm.counts,method = "pearson"))

# plot () can directly interpret the output of hclust ()

* plot(hclust(distance.m\_rlog),

labels = colnames(rlog.norm.counts),

main = "rlog transformed read counts distance:Pearson correlation")

# Assignment for Lab 10

**Please answer the following questions:**

1. Explain the differences of the boxplots using untransformed and transformed read counts. What is the median expression value in both boxplots? What is the maximum? Can you guess the mean?
2. Interpret the PCA analysis results. Are the control and Mov10 samples quite different?

# Reference

These lab materials are from the following website:

1. <http://www.ncbi.nlm.nih.gov/pubmed/25464849>
2. [https://github.com/hbctraining/Intro-to-rnaseq-hpc-salmon-flipped/tree/main/](https://github.com/hbctraining/Intro-to-rnaseq-hpc-salmon-flipped/tree/main/lessons) [lessons](https://github.com/hbctraining/Intro-to-rnaseq-hpc-salmon-flipped/tree/main/lessons)
3. <https://chagall.med.cornell.edu/RNASEQcourse/Intro2RNAseq.pdf>