Noah L. Schrick
1492657

Lab 3. Introduction to gene expression data: normalization and exploratory cluster analysis. There are two main technologies for generating high-throughput gene expression data: microarray and RNA-Seq. In this lab, we will work with an RNA-Seq based gene expression data set from a study of major depressive disorder. This data is also available at https://github.com/insilico/DepressionGeneModules/tree/master/secondary_expression_data. Include plots in your Word doc report and use a color to highlight your answers.

A. Load Data. Download the RNA-Seq data from Harvey (sense.filtered.cpm.Rdata). Load the data into an RStudio with an R script. **1.** How many genes and samples are there? The rows are genes and the columns are samples.

```
# load gene expression data
# set the current working directory first (setwd())
load("sense.filtered.cpm.Rdata")
dim(sense.filtered.cpm)
colnames(sense.filtered.cpm)
```

```
8923 genes, 157 samples
```

Download the demographic (sex, age,…) and clinical diagnostic (depression) data from Harvey or github (Demographic_symptom.csv).

```
# phenotype (mdd/hc) is in a separate file
# match phenotypes to expression data subject ids
subject.attrs <- read.csv("Demographic_symptom.csv", stringsAsFactors = FALSE)
dim(subject.attrs)  # 160 subjects x 40 attributes
colnames(subject.attrs)  # interested in X (sample ids) and Diag (diagnosis)
subject.attrs$X
subject.attrs$Diag
```

There are more samples in the demographic data than in the gene expression data, so we need to match gene expression samples with their diagnosis. **2.** How many cases and controls are there?

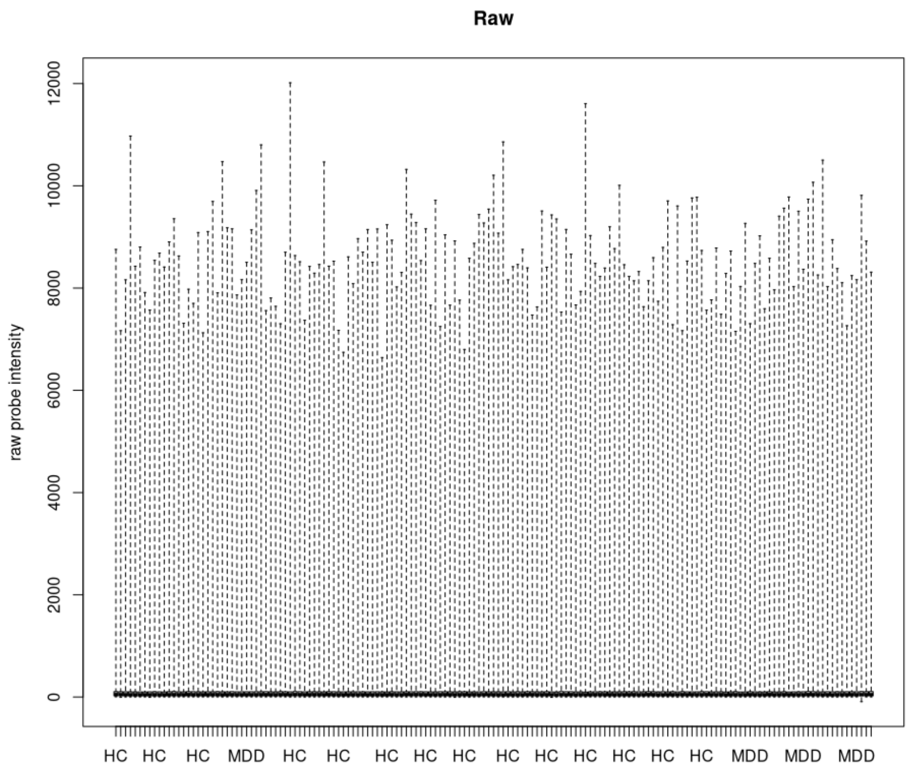
```
library(dplyr) # install.packages("dplyr")
# create a phenotype vector
# grab X (subject ids) and Diag (Diagnosis) from subject.attrs that
# intersect %in% with the RNA-Seq data
phenos.df <- subject.attrs %>%
  filter(X %in% colnames(sense.filtered.cpm)) %>%
  dplyr::select(X, Diag)
colnames(phenos.df) # $Diag is mdd diagnosis
# grab Diag column and convert character to factor
mddPheno <- as.factor(phenos.df$Diag)  # this is our phenotype/class vector

summary(mddPheno) # MDD -- major depressive disorder, HC -- healthy control
```

```
HC: 79
MDD: 78
```

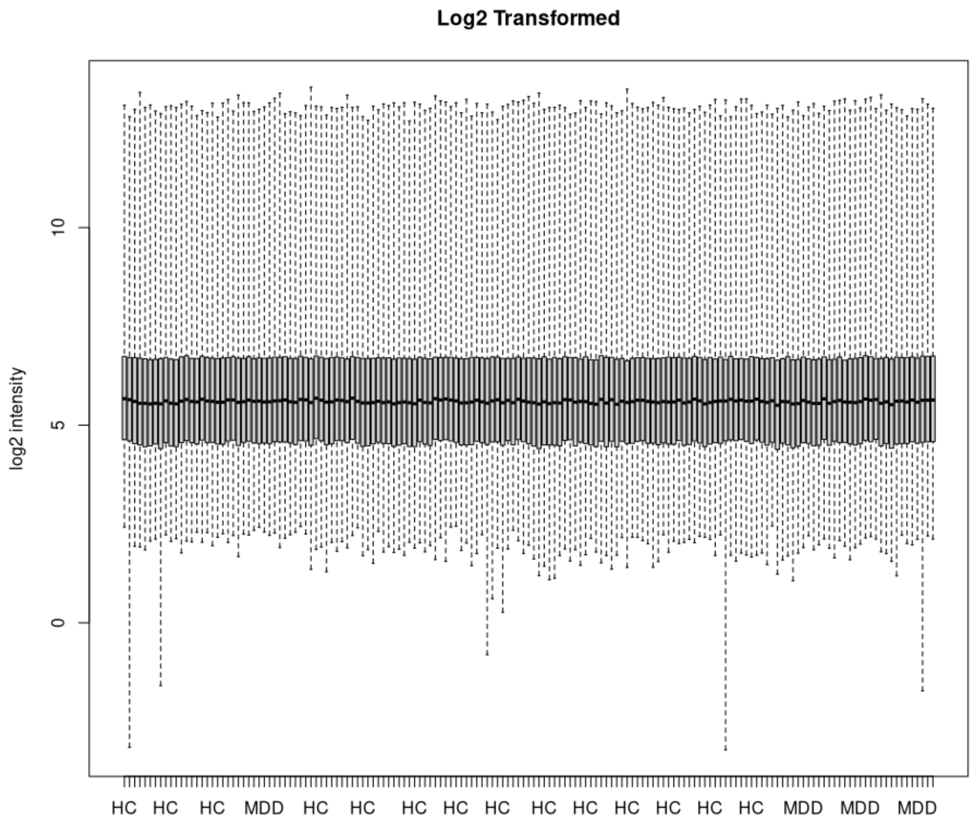
B. Normalization. The data has already been normalized by counts per million reads, but we would like to explore the data to determine whether additional normalization and transformations of the data are needed. The log2 transformation is meant to make the distribution of expression levels symmetric within a subject. Use the code below to visualize the distribution of probe intensities for all samples. The boxplots show expression of all genes for each sample, where samples are arranged across the x-axis. The histograms plot the density of expresson levels for one sample. **3.** Show your plots and discuss the differences between the original data (raw cpm) and data that has been log2 transformed.

```
### raw cpm boxplots and histogram of one sample
boxplot(sense.filtered.cpm,range=0,ylab="raw probe intensity", main="Raw",
names=mddPheno)

hist(sense.filtered.cpm[,1], freq=F, ylab="density", xlab="raw probe
intensity", main="Raw Data Density for Sample 1")

### log2 transformed boxplots and histogram
boxplot(log2(sense.filtered.cpm), range=0,ylab="log2 intensity", main="Log2
Transformed", names=mddPheno)

hist(log2(sense.filtered.cpm[,1]), freq=F, ylab="density", xlab="log2 probe
intensity", main="log2 Data Density for Sample 1")
```

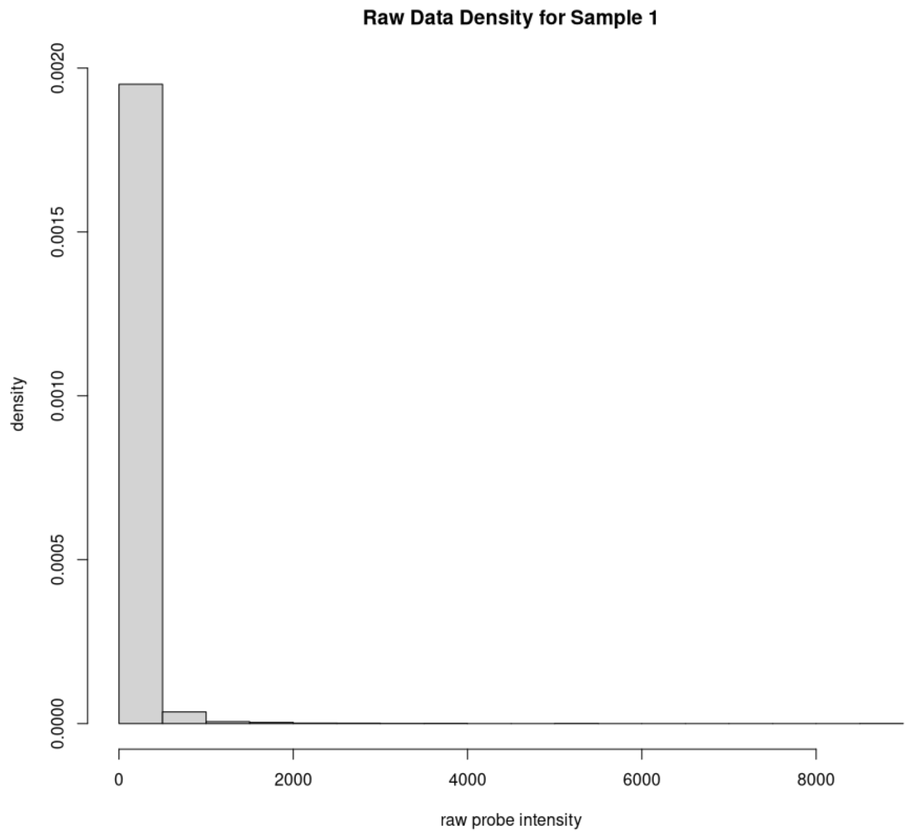
Original Boxplot



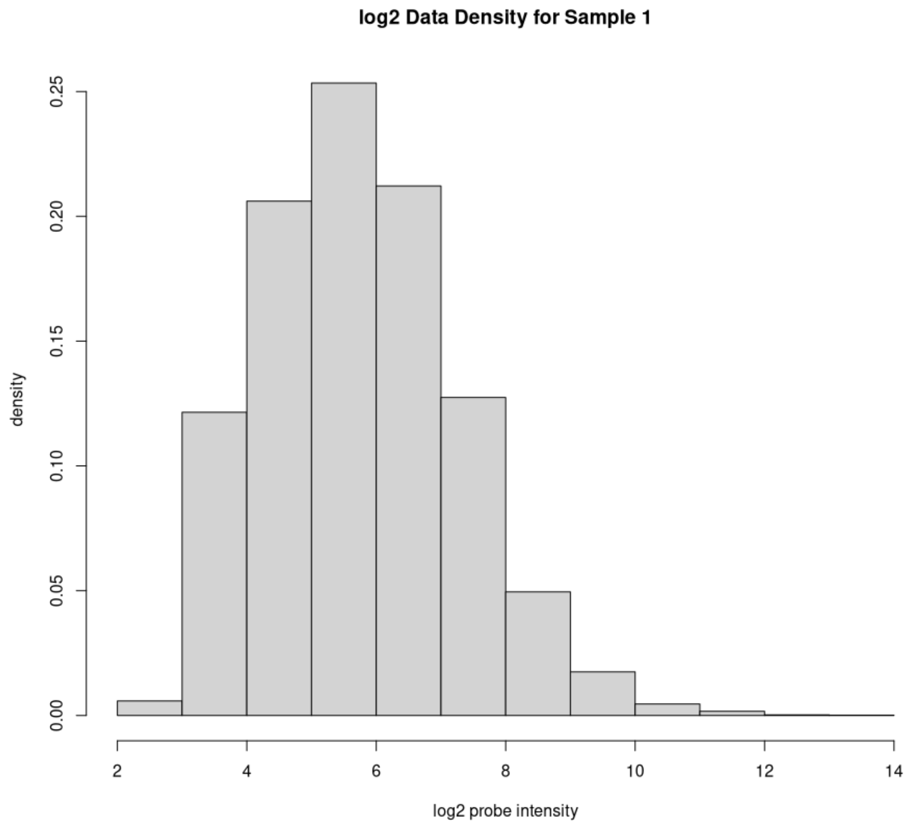
Log2 Transformed Boxplot



Original Histogram



Log2 Transformed Histogram



Discussion:

For the boxplots -

The original boxplot figure had its belt at or near the 0 value for probe intensity, demonstrating a skew in the data. With the log2 transformation, the belt fits closer toward the middle of the intensity results. This transformation also caused a shift in the whiskers, since the interquartile ranges also underwent a transformation.

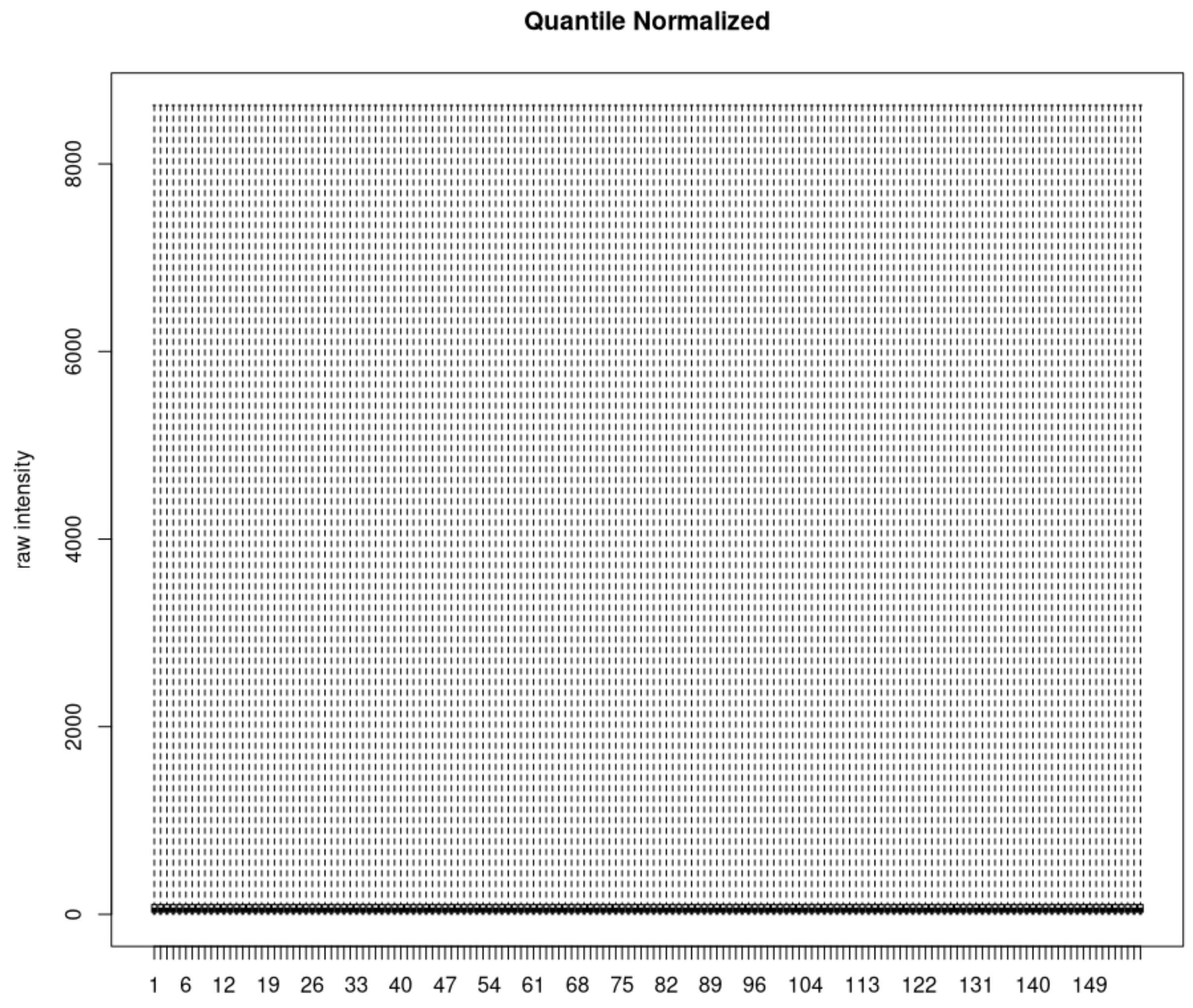
For the histogram -

The original histogram had a visually evident skew that did not fit a typical bell curve. This invalidates normal distribution simplifications where mean, median, and mode are the same value. The mean, median, and mode of the original values and the transformed values are below:

	Mean	Mode	Median
Original	103.086400	10.243872	50.93252
Log2 Transformed	5.757042	3.356689	5.670515

Run the code below to perform quantile normalization of the data. **4.** How does quantile normalization affect each sample's distribution of gene expression in comparison to the other samples in the boxplots (show the boxplots)?

```
# install quantile normalize
#install.packages("BiocManager")
library(BiocManager)
BiocManager::install("preprocessCore")
library(preprocessCore)    # replace with custom function?

# apply quantile normalization
mddExprData_quantile <- normalize.quantiles(sense.filtered.cpm)

boxplot(mddExprData_quantile,range=0,ylab="raw intensity", main="Quantile
Normalized")
```

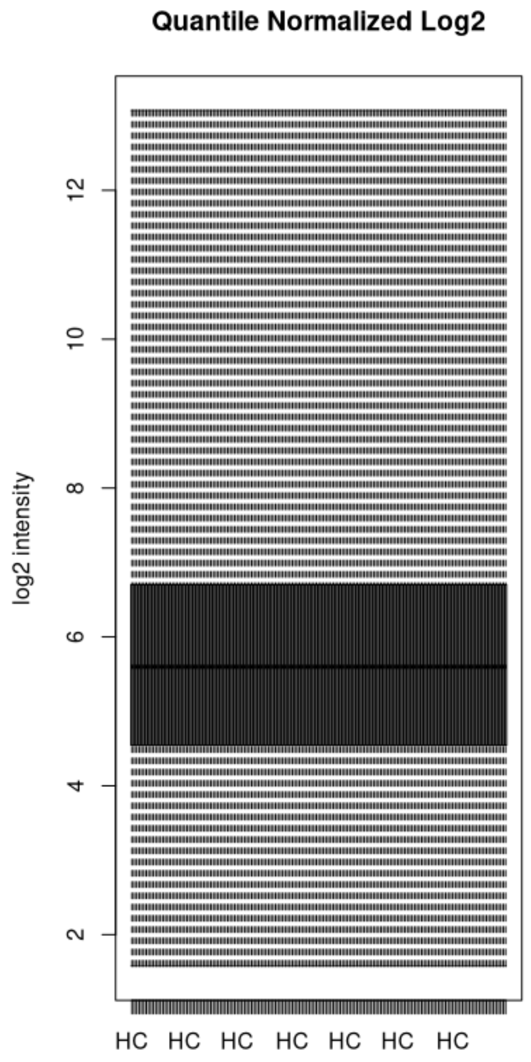


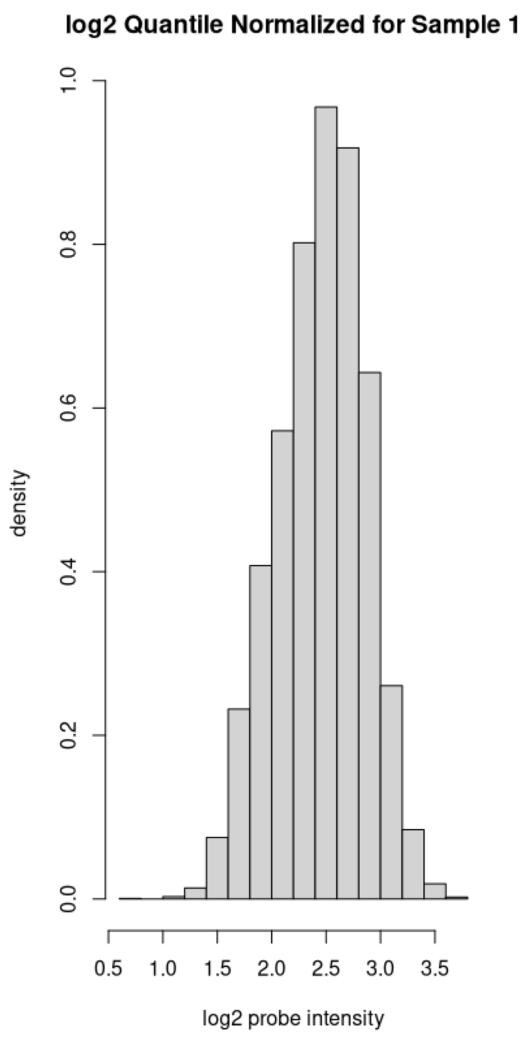
Discussion:
Quantile normalization will make distributions nearly identical. Looking at the maximum intensity for the new boxplot, all have seemingly identical values. When viewing the normalized data, we can confirm the data is nearly similar. Note: I am only showing the first 3 columns and first 6 rows for clarity.

```
           [,1]        [,2]        [,3]
[1,]   9.299333    9.350118    9.157426
[2,]  49.966026   50.071288   40.204923
[3,]  13.926249   14.806380   15.240474
[4,]   8.798341    5.960953   12.532579
[5,]  45.139933   41.850599   43.335455
[6,] 173.861687  219.696979  153.666608
```

Now let's apply the log2 transformation to the normalized data to make the normalized data more symmetric. **5**. Show the plots of the final processed data.

```
mddExprData_quantileLog2 <- log2(mddExprData_quantile)
# add phenotype names to matrix
colnames(mddExprData_quantileLog2) <- mddPheno

boxplot(mddExprData_quantileLog2,range=0,ylab="log2 intensity", main="Quantile
Normalized Log2")

hist(log2(mddExprData_quantileLog2[,1]), freq=F, ylab="density", xlab="log2
probe intensity", main="log2 Quantile Normalized for Sample 1")
```





6. Based on quantile normalization, why is the output from the following commands expected?

```
mean(mddExprData_quantileLog2[,1])
colMeans(mddExprData_quantileLog2)
```

This is expected since this is the primary function of quantile normalization – to make distributions of data identical. The non-log2 quantile normalization wasn't identical, but values were similar. By applying the log2 transformation, we were able to normalize the data further, and allow for the quantile normalization to make all data identical.

C. Exploratory clustering of microarray data. Below we run multi-dimensional scaling (MDS) and hierarchical clustering to get an idea of how the samples are related based on the similarity of their gene expression. Later we will swap dimensions to see how the genes are related. In addition to clustering samples, the first two dimensions of MDS explain the most variation in the data. Ideally this variation in the expression data is due to biological differences between the samples. **7.** Run the code below and show plots. What is the size of the mdd.mds (i.e., use dim to compute number of rows and columns) and what do the rows and columns represent? **8.** How are the MDD/HC samples distributed/mixed in the mds space? Are there any unusual samples?

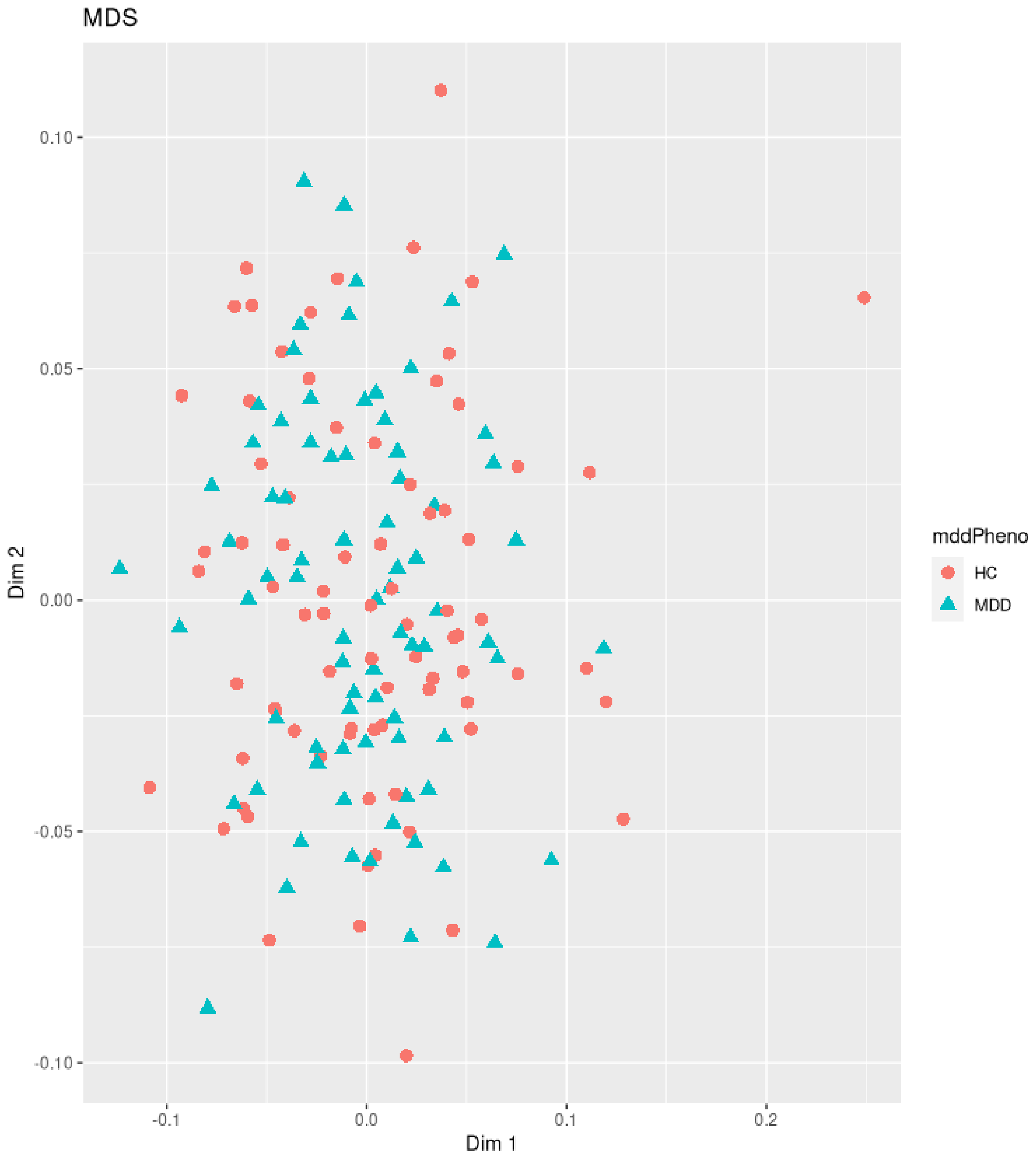
```
# transpose data matrix and convert to data frame
# ggplot wants data frame and subjects as rows
expr_SxG <- data.frame(t(mddExprData_quantileLog2)) # Subject x Gene
colnames(expr_SxG) <- rownames(sense.filtered.cpm)  # add gene names

## MDS of subjects
#d<-dist(expr_SxG)         # Euclidean metric
mddCorr<-cor(t(expr_SxG))  # distance based on correlation
d <- sqrt(1-mddCorr)
mdd.mds <- cmdscale(d, k = 2)
x <- mdd.mds[,1]
y <- mdd.mds[,2]
mdd.mds.df <- data.frame(mdd.mds)
colnames(mdd.mds.df) <- c("dim1","dim2")

#install.packages("ggplot2") # if not already installed
#BiocManager::install("ggplot2")
library(ggplot2)

p <- ggplot() # initialize empty ggplot object
p <- p + geom_point(data=mdd.mds.df, aes(x=dim1, y=dim2, color=mddPheno,
shape=mddPheno), size=3)
p <- p + ggtitle("MDS") + xlab("Dim 1") + ylab("Dim 2")
print(p)
```

7.

```
> dim(mdd.mds)
[1] 157   2
```
Rows: Subject
Cols: Genes

8.

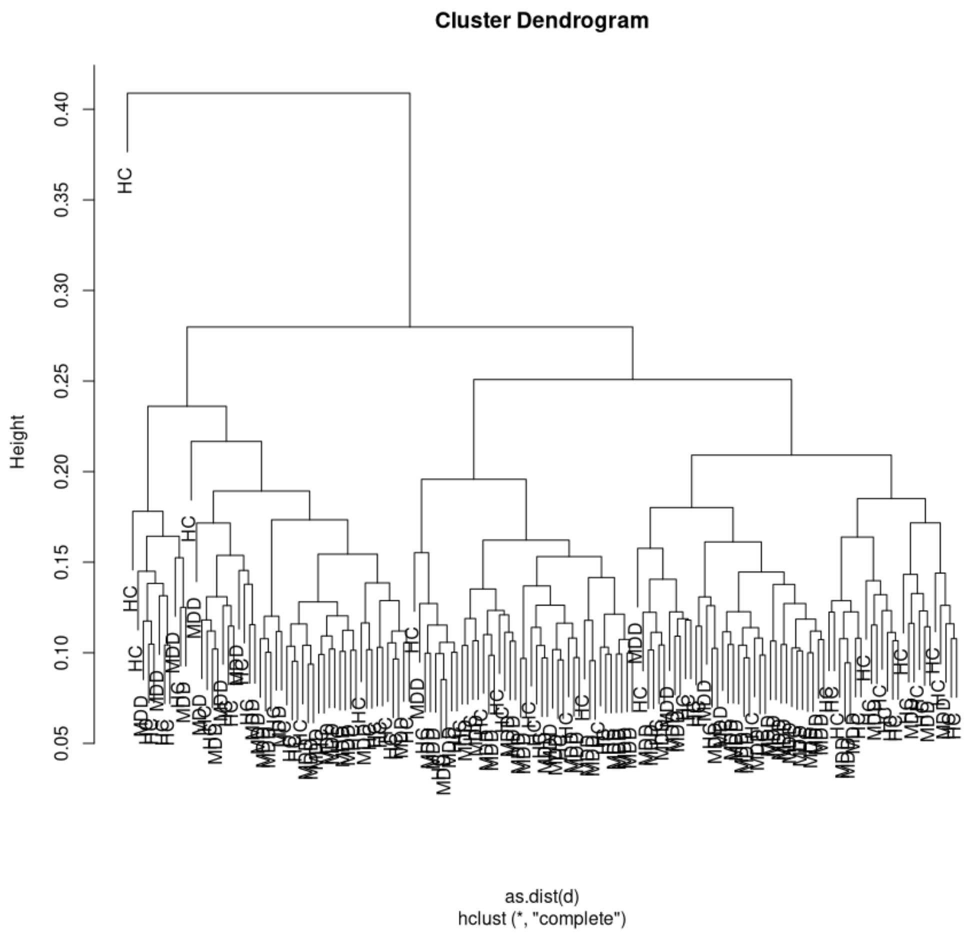
The MDD and HC samples are pretty well mixed in the MDS space. There are no visually evident ways to cleanly divide the space and identify clusters. There are a few samples that appear to be outliers. The most evident is the data point at roughly (0.25, 0.60), which is quite distant from all other points.

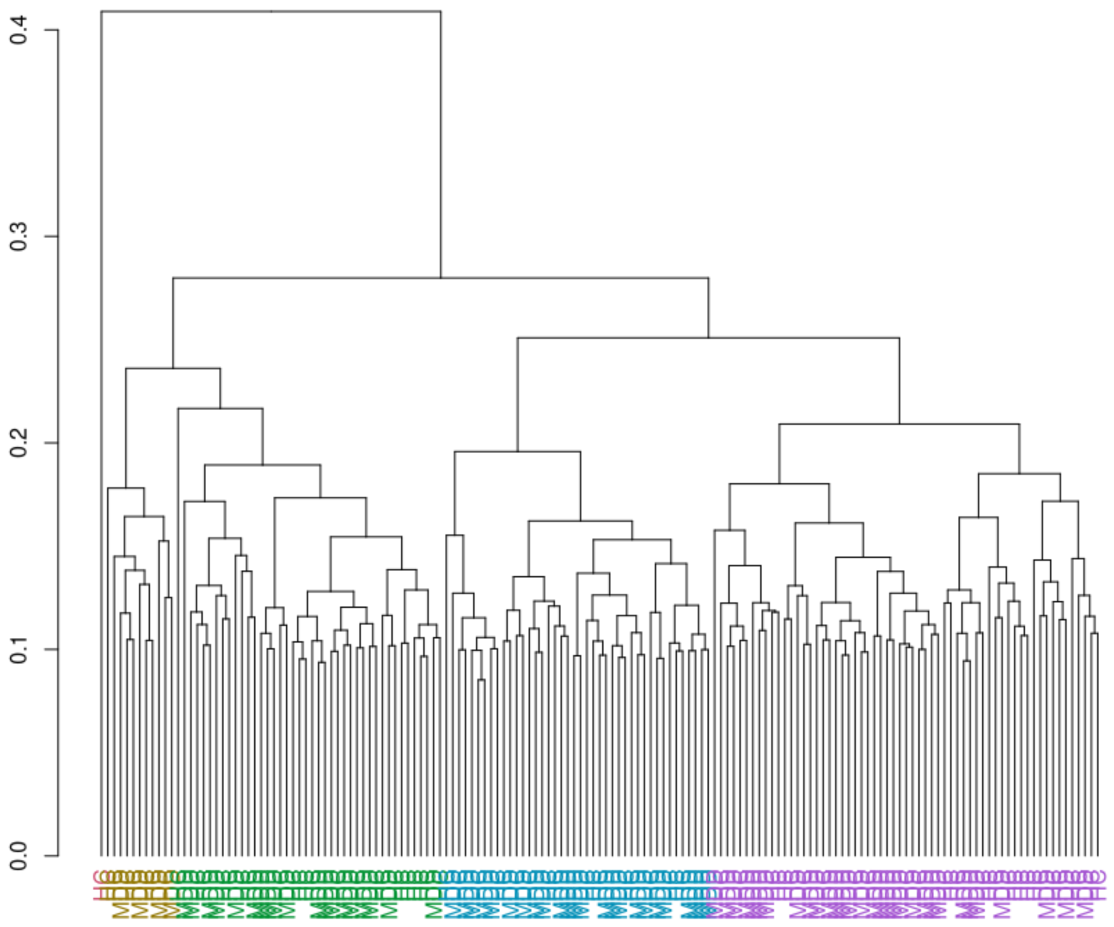


MDS
Dim 2
Dim 1
0.10
0.05
0.00
-0.05
-0.10
-0.1
0.0
0.1
0.2
mddPheno
HC
MDD

Below, use the distance matrix to create a dendrogram to visualize the relationships between samples. Show the plot. **9.** How are the relationships between samples that you observed in the mds plot reflected in the tree? In other words, how are the MDD/HC samples distributed in the tree (the table command might also help you interpret the distribution) and are there any unusual samples in the tree?

```
## hierarchical cluster of subjects
mddTree = hclust(as.dist(d))
mddTree$labels <- mddPheno
plot(mddTree)

num.clust <- 5
mddCuts <- cutree(mddTree,k=num.clust)
table(names(mddCuts),mddCuts)

#install.packages("dendextend")
library(dendextend)
dend <- as.dendrogram(mddTree)
dend=color_labels(dend, k=num.clust)
#dend <- dend %>% color_branches(k = 4)
plot(dend) # selective coloring of branches AND labels
```




0.4
0.3
0.2
0.1
0.0

9.

Similar to question 8, the samples are pretty well mixed. From the table command, we see:

```
         mddCuts
          1   2  3  4  5
  HC   18  30 23  7  1
 MDD  24 31 19  4  0
```

Converting this to a percentage rather than frequency, with the margin set to 2 to compute across columns (clusters) rather than rows, we see:

```
> prop.table(mddCutstable, margin=2)
          mddCuts
          1            2            3            4            5
HC    0.4285714    0.4918033    0.5476190    0.6363636    1.0000000
MDD 0.5714286    0.5081967    0.4523810    0.3636364    0.0000000
```

In clusters 1, 2, and 3, the mix is nearly 50/50. In cluster 4, it's a bit more spread, but still has a 60/40 mix. Only cluster 5 has a 100/0, but only one item is in that cluster due to that point being quite distant in comparison to the other data.

Use the code below to identify clusters using the dynamic cutree from WGCNA.

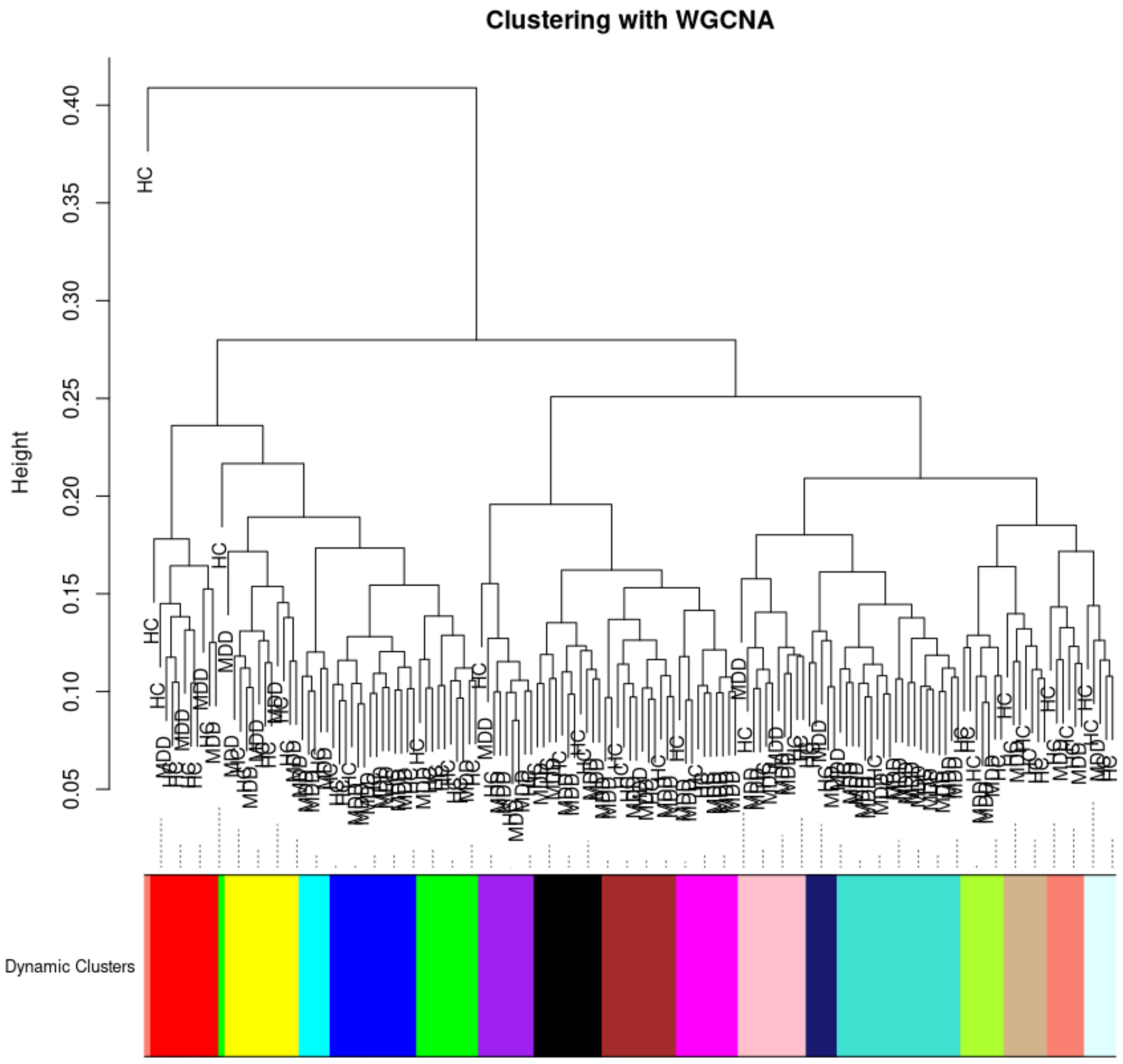
```
#library(BiocManager)
#BiocManager::install("WGCNA")
library(WGCNA)

# Plot the dendrogram and colors underneath
sizeGrWindow(8,6)
dynamicMods = cutreeDynamic(dendro = mddTree, distM = d,
                    deepSplit = 2, pamRespectsDendro = FALSE,
                    minClusterSize = 2)
mddColors = labels2colors(dynamicMods)
table(mddColors)
table(mddColors,names(mddCuts))
plotDendroAndColors(mddTree, mddColors, "Dynamic Clusters",
                    dendroLabels = NULL, # hang = -1,
                    addGuide = TRUE, #guideHang = 0.05,
                    main = "Clustering with WGCNA")
```

The second table above shows how often a dynamic cluster (color) co-occurs with a phenotype label (MDD or HC). **10.** Show the plot and second table. Which clusters (colors) are imbalanced towards one phenotype or the other? Which clusters are most evenly mixed?

10.

> table(mddColors,names(mddCuts))

mddColors	**HC**	**MDD**
black	**5**	**6**
blue	**7**	**7**
brown	**5**	**7**
cyan	**1**	**4**
green	**9**	**2**
greenyellow	**4**	**3**
lightcyan	**4**	**1**
magenta	**4**	**6**
midnightblue	**3**	**2**
pink	**4**	**7**
purple	**4**	**5**
red	**7**	**4**
salmon	**5**	**2**
tan	**6**	**1**
turquoise	**5**	**15**
yellow	**6**	**6**



Clustering with WGCNA
Height
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
HC
MDD
Dynamic Clusters

```
prop.table(mddColorstable, margin = 1)
```

```
mddColors          HC          MDD
black          0.4545455   0.5454545
blue           0.5000000   0.5000000
brown          0.4166667   0.5833333
cyan           0.2000000   0.8000000
green          0.8181818   0.1818182
greenyellow    0.5714286   0.4285714
lightcyan      0.8000000   0.2000000
magenta        0.4000000   0.6000000
midnightblue   0.6000000   0.4000000
pink           0.3636364   0.6363636
purple         0.4444444   0.5555556
red            0.6363636   0.3636364
salmon         0.7142857   0.2857143
tan            0.8571429   0.1428571
turquoise      0.2500000   0.7500000
yellow         0.5000000   0.5000000
```

Converting the original table into percentages rather than frequencies, and computing across clusters (rows), we see a few clusters that are pretty evenly mixed. Blue and yellow are 50/50 and perfectly mixed, black, brown, greenyellow, and purple are also fairly well mixed.

Cyan, green, lightcyan, pink, salmon, tan, and turquoise are fairly imbalanced.

Note: We expect some genes to show differences in expression between MDD/HC groups (differentially expressed), which we will investigate in the next lab. However, when clustering MDD/HC groups based on all of the genes, we expect the effect of a few differentially expressed genes to be washed out (hidden) among the many genes that are not related to differences between MDD/HC.

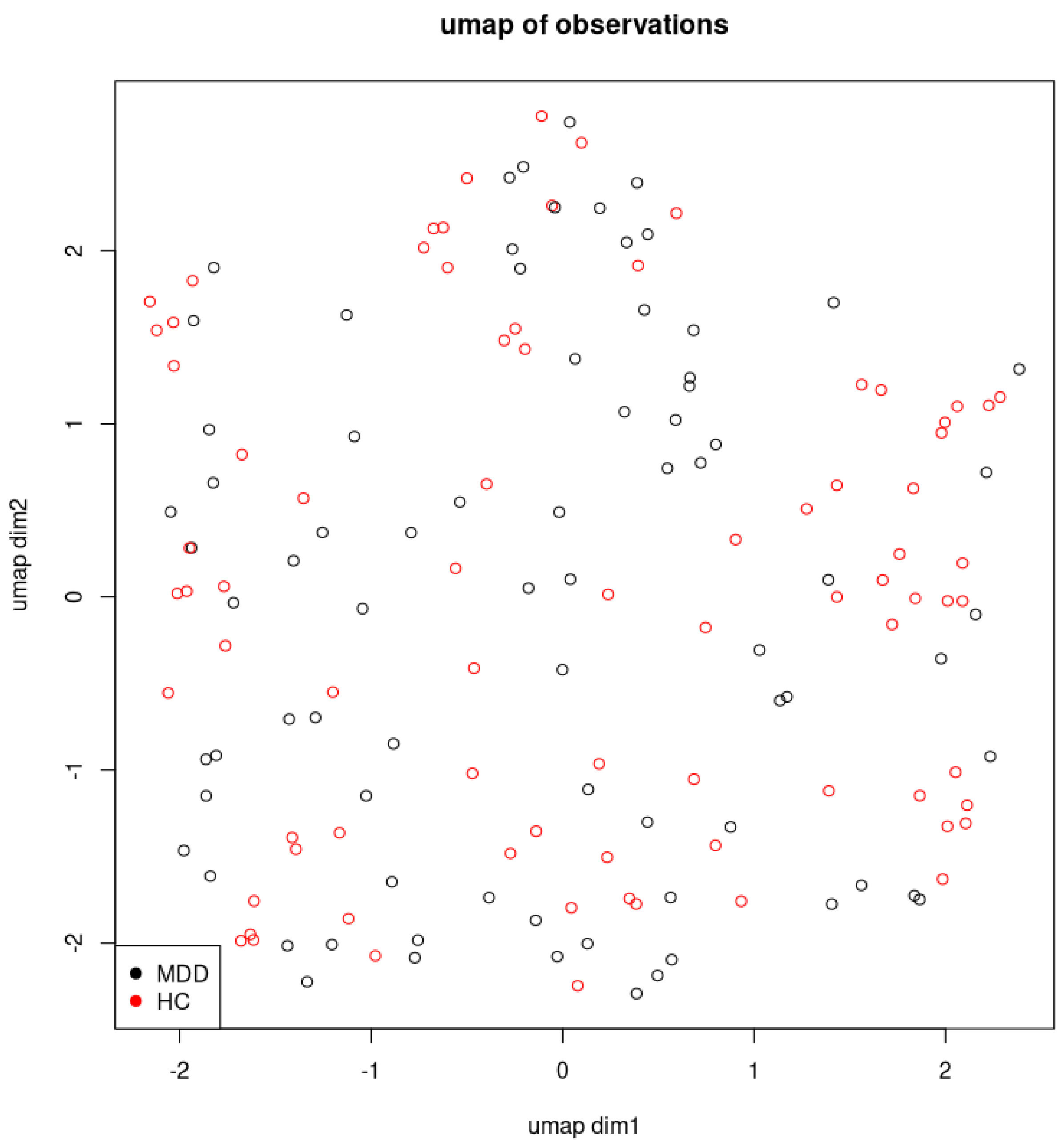
Optional. Use UMAP (Uniform Manifold Approximation and Projection) to cluster subjects. Show plot.

```
library(umap)
# change umap config parameters
custom.config = umap.defaults
custom.config$random_state = 123
custom.config$n_epochs = 500

# umap to cluster observations
obs_umap = umap(expr_SxG, config=custom.config)
#add colors for MDD/HC
colors = rep("black",nrow(expr_SxG))
#colors[dats[,ncol(dats)]==1]="red"
dim(obs_umap$layout)
plot(obs_umap$layout, col=colors,
     main="umap of observations", xlab="umap dim1", ylab="umap dim2")
#legend("bottomleft", legend = c("class -1","class +1"),
#       col=c("black","red"), pch=19)
```

NOTE: I changed the colors to the following:

```
        for (i in 1:nrow(expr_SxG)) {
                colors[i] <- ifelse(grepl("MDD", rownames(expr_SxG)[i]), "black", "red")
        }
```

This lets each data point in the UMAP plot to be colored according to its class. I found this easier to view, and the original coloring just had each data point as black.


umap of observations
umap dim2
umap dim1
2
1
0
-1
-2
-2
-1
0
1
2
● MDD
● HC