Julie Garcia
October 16, 2017

Gene Expression Data Analysis and Visualization
410.671
HW #2

For this assignment, we will be evaluating different normalization methods on 2-channel arrays in which 4 biological samples were run. The study is from GEO and the description of the experiment is provided as follows.

Series GSE12050: Subcutaneous adipose tissue from lean and obese subjects

Obtaining adipose tissue samples are paramount to the understanding of human obesity. We have examined the impact of needle-aspirated and surgical biopsy techniques on the study of subcutaneous adipose tissue (scAT) gene expression in both obese and lean subjects. Biopsy sampling methods have a significant impact on data interpretation and revealed that gene expression profiles derived from surgical tissue biopsies better capture the significant changes in molecular pathways associated with obesity. We hypothesize that this is because needle biopsies do not aspirate the fibrotic fraction of scAT; which subsequently results in an under-representation of the inflammatory and metabolic changes that coincide with obesity. This analysis revealed that the biopsy technique influences the gene expression underlying the biological themes commonly discussed in obesity (e.g. inflammation, extracellular matrix, metabolism, etc.), and is therefore a caveat to consider when designing microarray experiments. These results have crucial implications for the clinical and physiopathological understanding of human obesity and therapeutic approaches.

We will be working with 4 lean subjects from which a needle biopsy was taken.

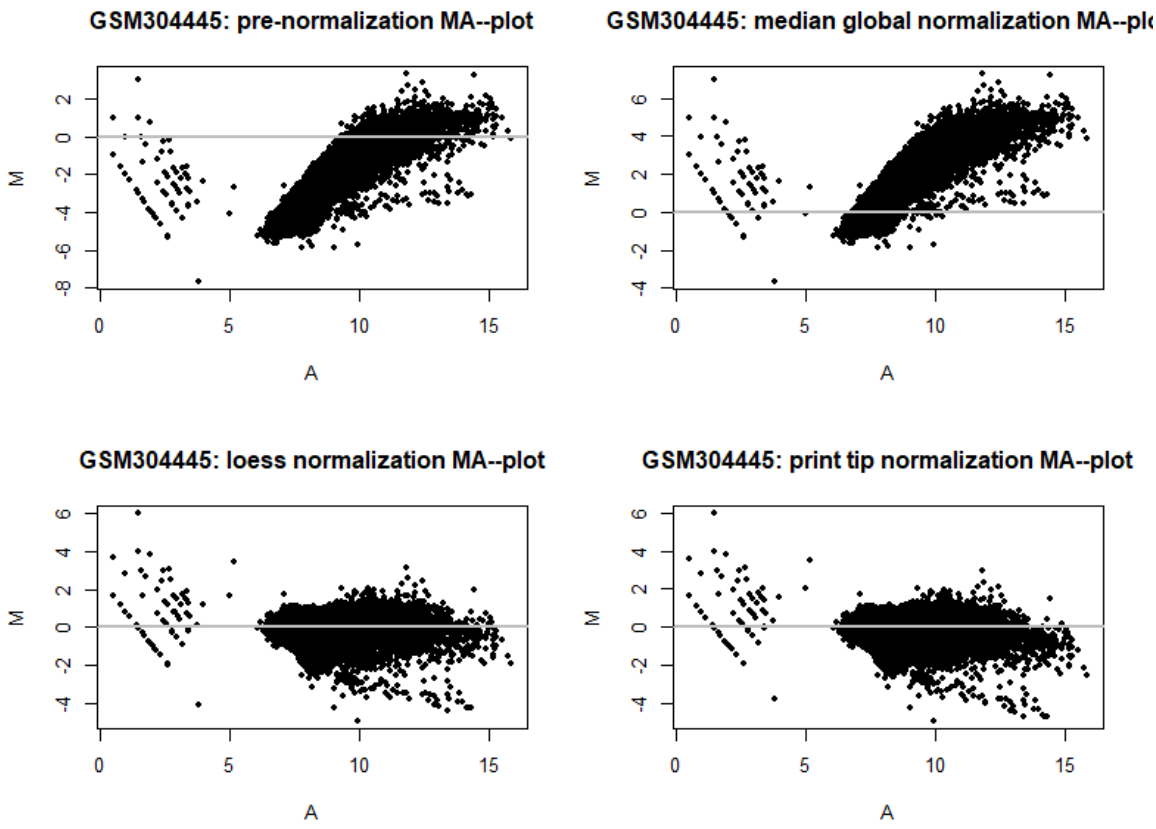
1.) First load the marray library, then load the 4 GenePix files, making sure to extract the foreground and background median values from the Cy5 and Cy3 channels. (2.5 pts)

```
source("https://bioconductor.org/biocLite.R")
biocLite("marray")
library(marray)

# extract raw marray data from genepix
help(read.GenePix)
targets.GSM304445 <- read.GenePix("GSM304445.gpr",
"c:\\temp\\datasets\\")
targets.GSM304446 <- read.GenePix("GSM304446.gpr",
"c:\\temp\\datasets\\")
targets.GSM304447 <- read.GenePix("GSM304447.gpr",
"c:\\temp\\datasets\\")
targets.GSM304448 <- read.GenePix("GSM304448.gpr",
"c:\\temp\\datasets\\")

class?marrayRaw
GSM304445.Cy5foreground.median <- median(targets.GSM304445@maRf)
GSM304445.Cy5background.median <- median(targets.GSM304445@maRb)
GSM304445.Cy3foreground.median <- median(targets.GSM304445@maGf)
GSM304445.Cy3background.median <- median(targets.GSM304445@maGb)
```

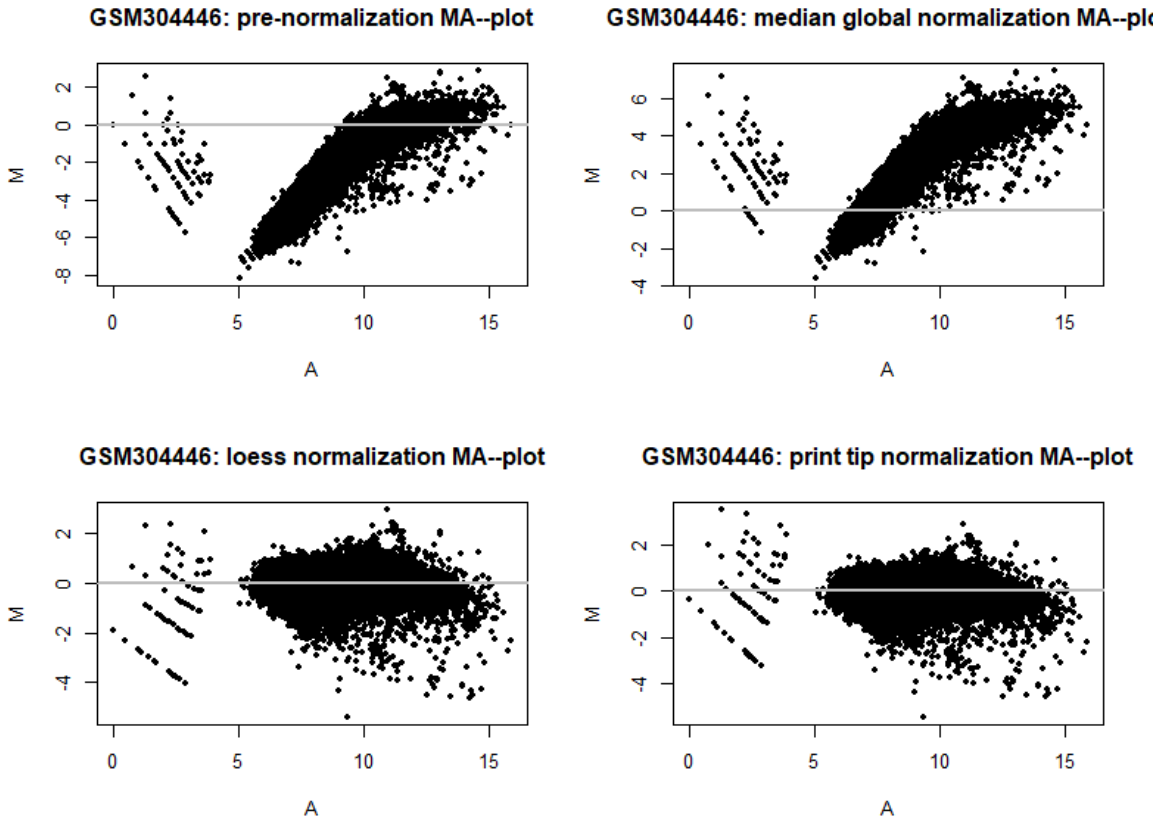
```
GSM304446.Cy5foreground.median <- median(targets.GSM304446@maRf)
GSM304446.Cy5background.median <- median(targets.GSM304446@maRb)
GSM304446.Cy3foreground.median <- median(targets.GSM304446@maGf)
GSM304446.Cy3background.median <- median(targets.GSM304446@maGb)

GSM304447.Cy5foreground.median <- median(targets.GSM304447@maRf)
GSM304447.Cy5background.median <- median(targets.GSM304447@maRb)
GSM304447.Cy3foreground.median <- median(targets.GSM304447@maGf)
GSM304447.Cy3background.median <- median(targets.GSM304447@maGb)

GSM304448.Cy5foreground.median <- median(targets.GSM304448@maRf)
GSM304448.Cy5background.median <- median(targets.GSM304448@maRb)
GSM304448.Cy3foreground.median <- median(targets.GSM304448@maGf)
GSM304448.Cy3background.median <- median(targets.GSM304448@maGb)
```

2.) Normalize each array using median global, loess, and print-tip-group loess methods. Then plot MvA plots of all 4 arrays comparing no normalization to the other 3 normalization approaches. (2 pts)

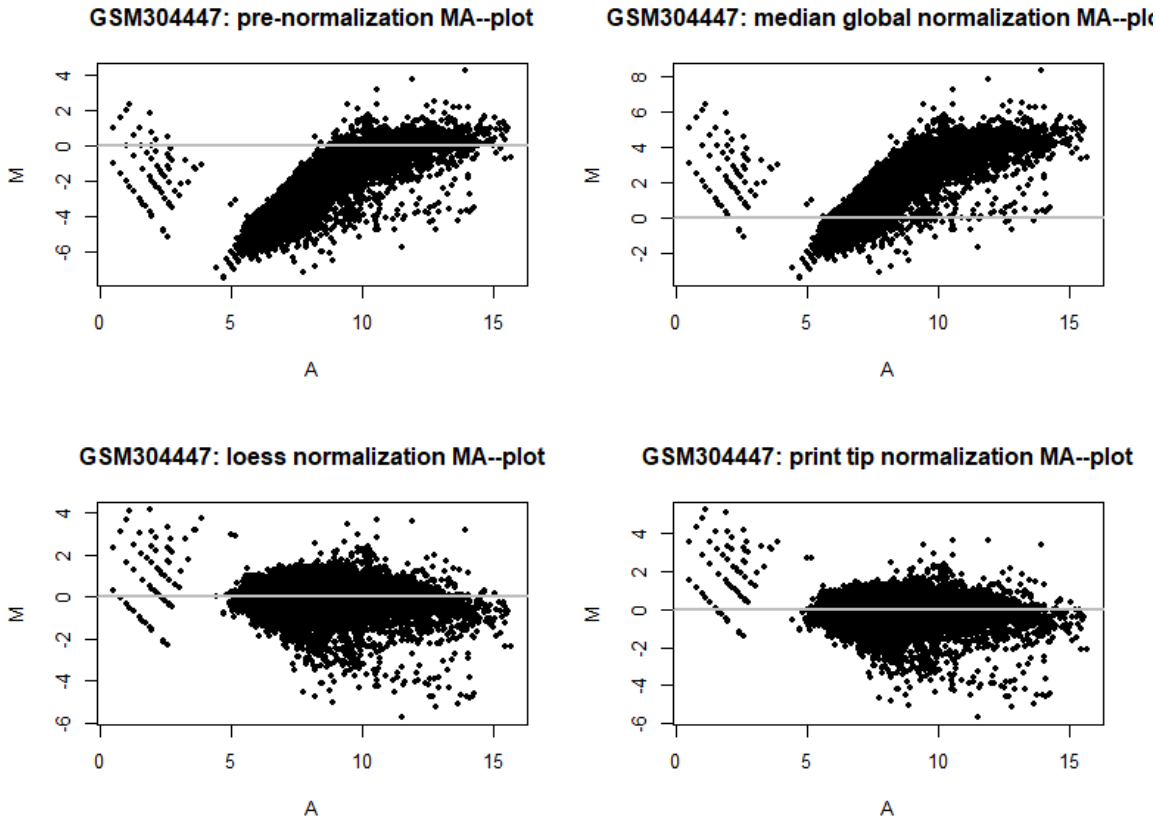
```
# GSM304445 - 3 types of normalization
GSM304445 <- targets.GSM304445
GSM304445.norm.median <- maNorm(GSM304445, norm="median")
GSM304445.norm.loess <- maNorm(GSM304445, norm="loess")
GSM304445.norm.print <- maNorm(GSM304445,
norm="scalePrintTipMAD")

# GSM304445 -plots
par(mfrow=c(2,2))
plot(GSM304445, lines.func=NULL, legend=NULL,
     main="GSM304445: pre-normalization MA--plot")
plot(GSM304445.norm.median, lines.func=NULL, legend=NULL,
     main="GSM304445: median global normalization MA--plot")
plot(GSM304445.norm.loess, lines.func=NULL, legend=NULL,
     main="GSM304445: loess normalization MA--plot")
plot(GSM304445.norm.print, lines.func=NULL, legend=NULL,
     main="GSM304445: print tip normalization MA--plot")
```



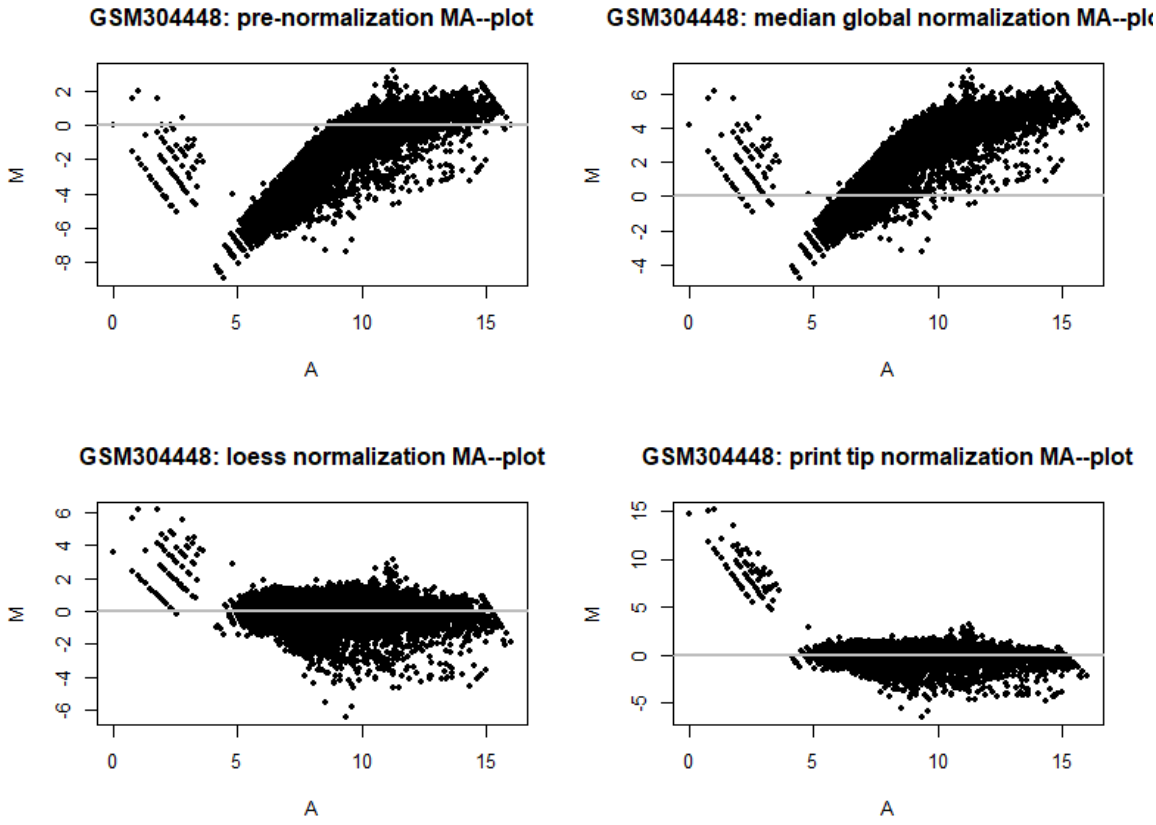

```
#GSM304446 -3 types of normalization
GSM304446 <- targets.GSM304446
GSM304446.norm.median <- maNorm(GSM304446, norm="median")
GSM304446.norm.loess <- maNorm(GSM304446, norm="loess")
GSM304446.norm.print <- maNorm(GSM304446,
norm="scalePrintTipMAD")

#GSM304446 - plots
par(mfrow=c(2,2))
plot(GSM304446, lines.func=NULL, legend=NULL,
     main="GSM304446: pre-normalization MA--plot")
plot(GSM304446.norm.median, lines.func=NULL, legend=NULL,
     main="GSM304446: median global normalization MA--plot")
plot(GSM304446.norm.loess, lines.func=NULL, legend=NULL,
     main="GSM304446: loess normalization MA--plot")
plot(GSM304446.norm.print, lines.func=NULL, legend=NULL,
     main="GSM304446: print tip normalization MA--plot")
```




```
#GSM304447 -3 types of normalization
GSM304447 <- targets.GSM304447
GSM304447.norm.median <- maNorm(GSM304447, norm="median")
GSM304447.norm.loess <- maNorm(GSM304447, norm="loess")
GSM304447.norm.print <- maNorm(GSM304447,
norm="scalePrintTipMAD")

#GSM304447 - plots
par(mfrow=c(2,2))
plot(GSM304447, lines.func=NULL, legend=NULL,
     main="GSM304447: pre-normalization MA--plot")
plot(GSM304447.norm.median, lines.func=NULL, legend=NULL,
     main="GSM304447: median global normalization MA--plot")
plot(GSM304447.norm.loess, lines.func=NULL, legend=NULL,
     main="v: loess normalization MA--plot")
plot(GSM304447.norm.print, lines.func=NULL, legend=NULL,
     main="GSM304447: print tip normalization MA--plot")
```

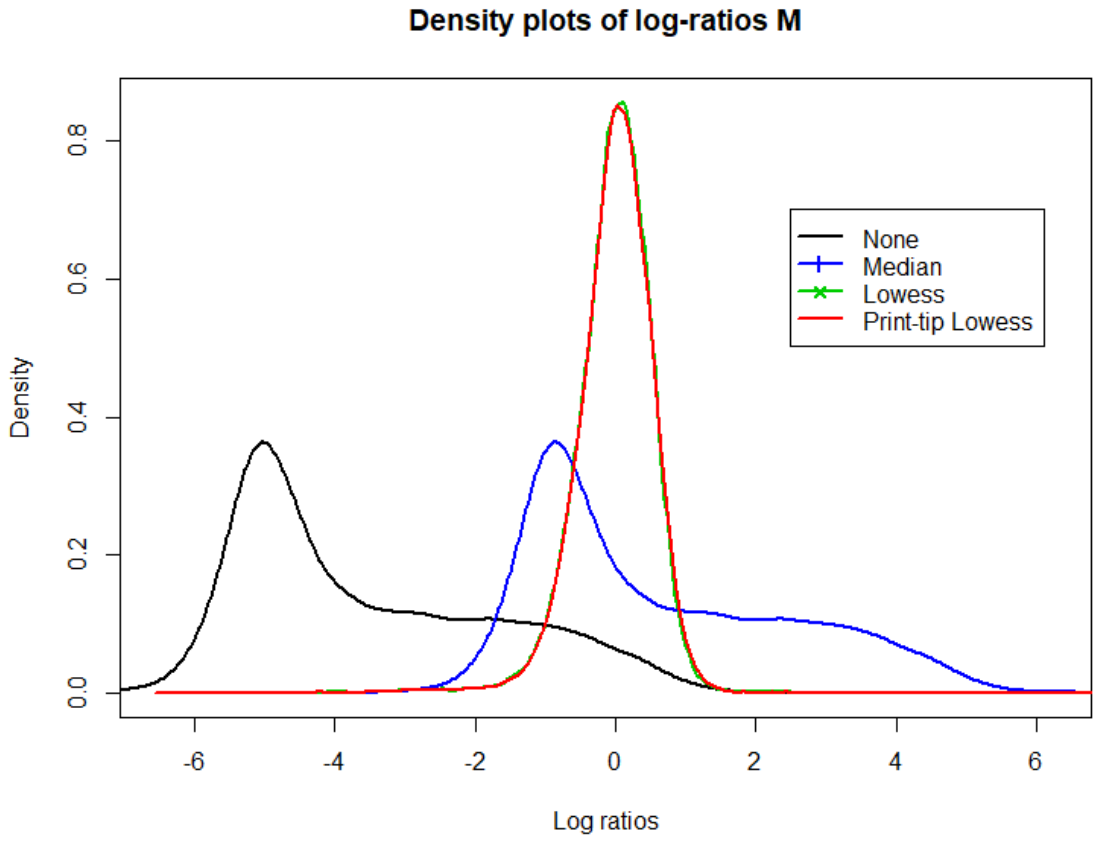



```
#GSM304448 -3 types of normalization
GSM304448 <- targets.GSM304448
GSM304448.norm.median <- maNorm(GSM304448, norm="median")
GSM304448.norm.loess <- maNorm(GSM304448, norm="loess")
GSM304448.norm.print <- maNorm(GSM304448,
norm="scalePrintTipMAD")

#GSM304448 - plots
par(mfrow=c(2,2))
plot(GSM304448, lines.func=NULL, legend=NULL,
     main="GSM304448: pre-normalization MA--plot")
plot(GSM304448.norm.median, lines.func=NULL, legend=NULL,
     main="GSM304448: median global normalization MA--plot")
plot(GSM304448.norm.loess, lines.func=NULL, legend=NULL,
     main="GSM304448: loess normalization MA--plot")
plot(GSM304448.norm.print, lines.func=NULL, legend=NULL,
     main="GSM304448: print tip normalization MA--plot")
```



3.) Plot density plots of the log ratio values for each normalization (and pre normalization) for only array #4. Put them all on the same plot. Make sure to label the axes and provide a legend. (2 pts)

```
par(mfrow=c(1,1))
density <- density(maM(GSM304448.norm.loess), na.rm=TRUE)
plot(x=NULL, y=NULL,
     xlim=c(min(density$x), max(density$x)),
     ylim=c(min(density$y), max(density$y)), type='n',
     main="Density plots of log-ratios M",
     xlab="Log ratios", ylab="Density")
lines(density(maM(GSM304448), na.rm=TRUE), lwd=2, col=1)
lines(density(maM(GSM304448.norm.median), na.rm=TRUE), lwd=2,
col=4 )
lines(density(maM(GSM304448.norm.loess), na.rm=TRUE), lwd=2,
col=3 )
lines(density(maM(GSM304448.norm.print), na.rm=TRUE), lwd=2,
col=2 )
help(legend)
legend(x=2.5, y=0.7, legend=c("None", "Median", "Lowess", "Print-
tip Lowess"),
       col=c(1, 4, 3, 2), pch = c(NA, 3, 4), lty=1, lwd=2)
```



4.) Based on the plots generated so far, which normalization do you think is most preferred for this dataset? (2 pts)

Based on the plots so far, the Lowess or Print-Tip Lowess methods of normalization seem to be the best for this set of data. In the MvA plots, the data is better distributed around the 0 line and again in the density plot you can see that the data is also more symmetric

5.) Research has demonstrated that often a single channel, background subtracted provides as good a normalization as using both channels. To test this, we will be utilizing the fact that these 4 samples are replicates and calculate the correlation between them. So, first extract the Cy5 foreground and background values for each of the 4 arrays and subtract the background from the foreground values, then $\log_2$ transform these values. Then calculate global median normalization on these 4 arrays using these background subtracted Cy5 values. Hint, you need to use the median of each array to scale, such that after normalization, all arrays will have a median of 1. (4 pts)

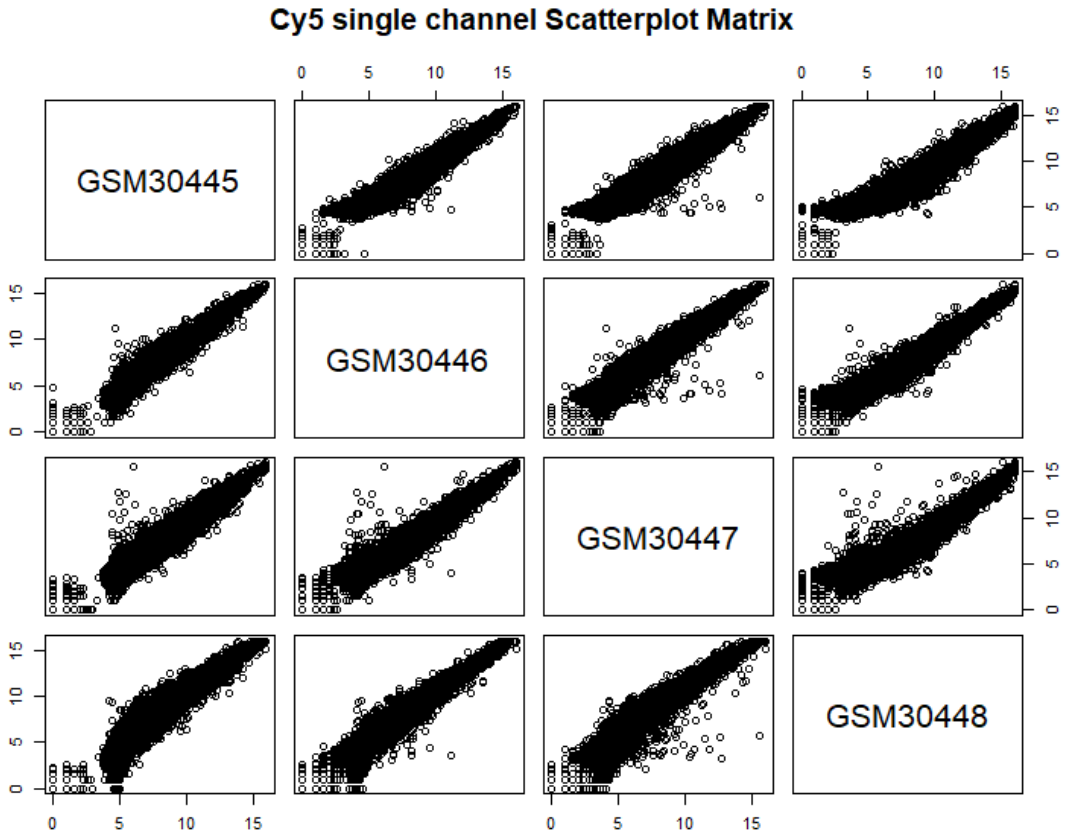
```
help(limma)
limmaUsersGuide(view=TRUE)

GSM304445.Cy5.f <- GSM304445@maRf
GSM304445.Cy5.b <- GSM304445@maRb
```

```
GSM304445.Cy5.subtract <- GSM304445.Cy5.f - GSM304445.Cy5.b
GSM304445.Cy5.subtract.log <- log2(GSM304445.Cy5.subtract)
GSM304445.Cy5.subtract.log
GSM304445.Cy5.norm <-
normalizeBetweenArrays(GSM304445.Cy5.subtract.log,
method="scale")

GSM304446.Cy5.f <- GSM304446@maRf
GSM304446.Cy5.b <- GSM304446@maRb
GSM304446.Cy5.subtract <- GSM304446.Cy5.f - GSM304446.Cy5.b
GSM304446.Cy5.subtract.log <- log2(GSM304446.Cy5.subtract)
GSM304446.Cy5.subtract.log
GSM304446.Cy5.norm <-
normalizeBetweenArrays(GSM304446.Cy5.subtract.log,
method="scale")

GSM304447.Cy5.f <- GSM304447@maRf
GSM304447.Cy5.b <- GSM304447@maRb
GSM304447.Cy5.subtract <- GSM304447.Cy5.f - GSM304447.Cy5.b
GSM304447.Cy5.subtract.log <- log2(GSM304447.Cy5.subtract)
GSM304447.Cy5.subtract.log
GSM304447.Cy5.norm <-
normalizeBetweenArrays(GSM304447.Cy5.subtract.log,
method="scale")

GSM304448.Cy5.f <- GSM304448@maRf
GSM304448.Cy5.b <- GSM304448@maRb
GSM304448.Cy5.subtract <- GSM304448.Cy5.f - GSM304448.Cy5.b
GSM304448.Cy5.subtract.log <- log2(GSM304448.Cy5.subtract)
GSM304448.Cy5.subtract.log
GSM304448.Cy5.norm <-
normalizeBetweenArrays(GSM304448.Cy5.subtract.log,
method="scale")
```

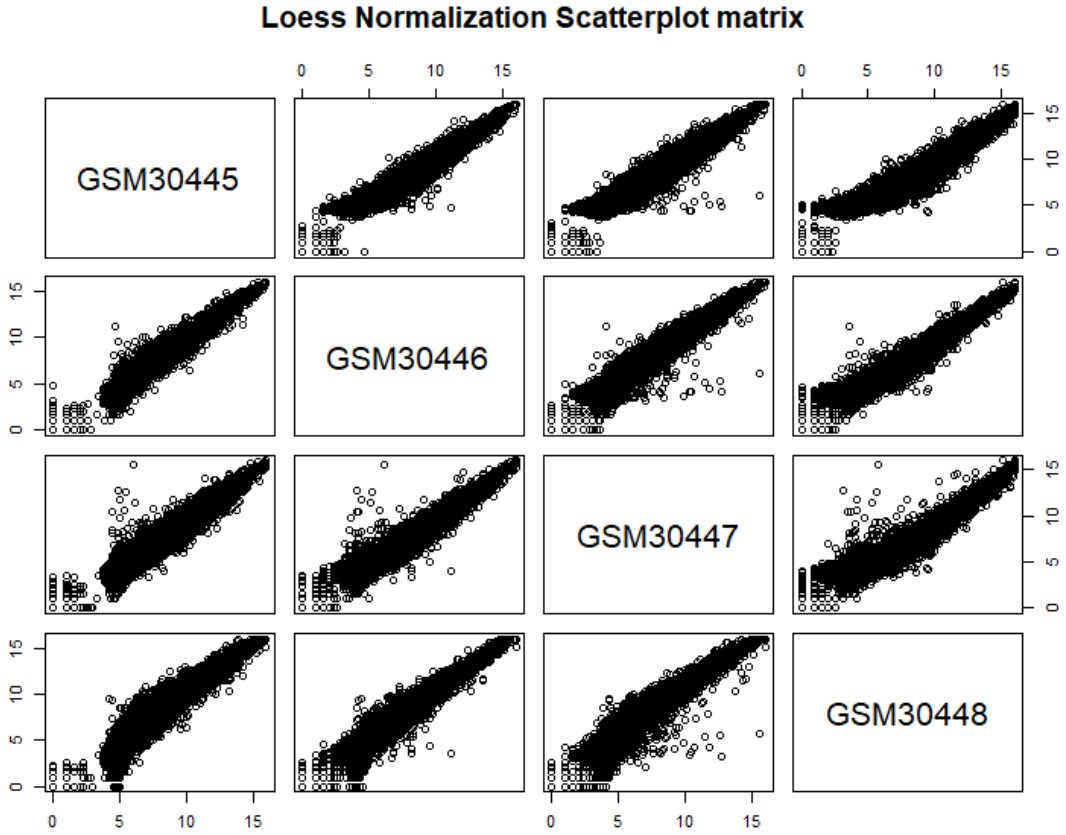
6.) Next calculate a Spearman's rank correlation between all 4 arrays that you normalized in #5 and do the same with the M values from loess normalized data that you generated in #2. Plot a scatter plot matrix for each of the two normalizations (pairs() function), and be sure to label the arrays and title the plot. Print the correlation coefficients to the screen. (4 pts)

```
GSM56.cor <- cor(as.numeric(GSM304445.Cy5.norm),
as.numeric(GSM304446.Cy5.norm), method = "spearman",
use="pairwise.complete.obs")
GSM57.cor <- cor(as.numeric(GSM304445.Cy5.norm),
as.numeric(GSM304447.Cy5.norm), method = "spearman",
use="pairwise.complete.obs")
```

```
GSM58.cor <- cor(as.numeric(GSM304445.Cy5.norm),
as.numeric(GSM304448.Cy5.norm), method = "spearman",
use="pairwise.complete.obs")
GSM67.cor <- cor(as.numeric(GSM304446.Cy5.norm),
as.numeric(GSM304447.Cy5.norm),
                 method = "spearman", use="pairwise.complete.obs")
GSM68.cor <- cor(as.numeric(GSM304446.Cy5.norm),
as.numeric(GSM304448.Cy5.norm),
                 method = "spearman", use="pairwise.complete.obs")
GSM78.cor <- cor(as.numeric(GSM304447.Cy5.norm),
as.numeric(GSM304448.Cy5.norm),
                 method = "spearman", use="pairwise.complete.obs")

help(pairs)
Cy5.norm.matrix <- cbind(GSM304445.Cy5.norm, GSM304446.Cy5.norm,
                         GSM304447.Cy5.norm, GSM304448.Cy5.norm)
Cy5.norm.matrix[1:4, 1:4]
colnames(Cy5.norm.matrix) <- c("GSM30445", "GSM30446",
"GSM30447", "GSM30448")
loess.norm.matrix <- cbind(maM(GSM304445.norm.loess),
maM(GSM304446.norm.loess),
                         maM(GSM304447.norm.loess),
maM(GSM304448.norm.loess))
colnames(loess.norm.matrix) <- c("GSM30445", "GSM30446",
"GSM30447", "GSM30448")

loess.norm.matrix[1:4, 1:4]
pairs(Cy5.norm.matrix, labels=colnames(Cy5.norm.matrix),
      main="Cy5 single channel Scatterplot Matrix")
```




```
pairs(Cy5.norm.matrix, labels=colnames(Cy5.norm.matrix),
      main="Loess Normalization Scatterplot matrix")
```




```
cor.Cy5.table <- rbind(c(1.0, GSM56.cor, GSM57.cor, GSM58.cor),
                  c(GSM56.cor, 1.0, GSM67.cor, GSM68.cor),
                  c(GSM57.cor, GSM67.cor, 1.0, GSM78.cor),
                  c(GSM58.cor, GSM68.cor, GSM78.cor, 1.0))
rownames(cor.Cy5.table) <- c("GSM304445", "GSM304446",
"GSM304447", "GSM304448")
colnames(cor.Cy5.table) <- c("GSM304445", "GSM304446",
"GSM304447", "GSM304448")
cor.Cy5.table
```

```
GSM304445 GSM304446 GSM304447 GSM304448
GSM304445 1.0000000 0.8962752 0.8790831 0.8990696
GSM304446 0.8962752 1.0000000 0.8766078 0.9080043
GSM304447 0.8790831 0.8766078 1.0000000 0.8854758
GSM304448 0.8990696 0.9080043 0.8854758 1.0000000
```

```
cor.norm.table <- rbind(c(1.0, GSM56.loess.cor, GSM57.loess.cor,
GSM58.loess.cor),c(GSM56.loess.cor, 1.0, GSM67.loess.cor,
GSM68.loess.cor),c(GSM57.loess.cor, GSM67.loess.cor, 1.0,
GSM78.loess.cor),c(GSM58.loess.cor, GSM68.loess.cor,
GSM78.loess.cor, 1.0))
```

```
rownames(cor.norm.table) <- c("GSM304445", "GSM304446",
"GSM304447", "GSM304448")
colnames(cor.norm.table) <- c("GSM304445", "GSM304446",
"GSM304447", "GSM304448")
cor.norm.table
```

```
GSM304445 GSM304446 GSM304447 GSM304448
GSM304445 1.0000000 0.6907476 0.7498902 0.6845716
GSM304446 0.6907476 1.0000000 0.7280997 0.7037628
GSM304447 0.7498902 0.7280997 1.0000000 0.7383481
GSM304448 0.6845716 0.7037628 0.7383481 1.0000000
```

7.) Now we want to compare these normalizations to quantile normalized data to see if we gain anything by leveraging the distributions across all 4 arrays. Carry out the steps in the lecture or use the paper from Bolstad *et al*. entitled: "A comparison of normalization methods for high density oligonucleotide array data based on variance and bias" (on the course website), but we are only going to conduct this on the Cy5 channel. The basic steps are as follows (these 6 steps are calculated on non-logged data; the data is logged after these steps are carried out): (8 pts)

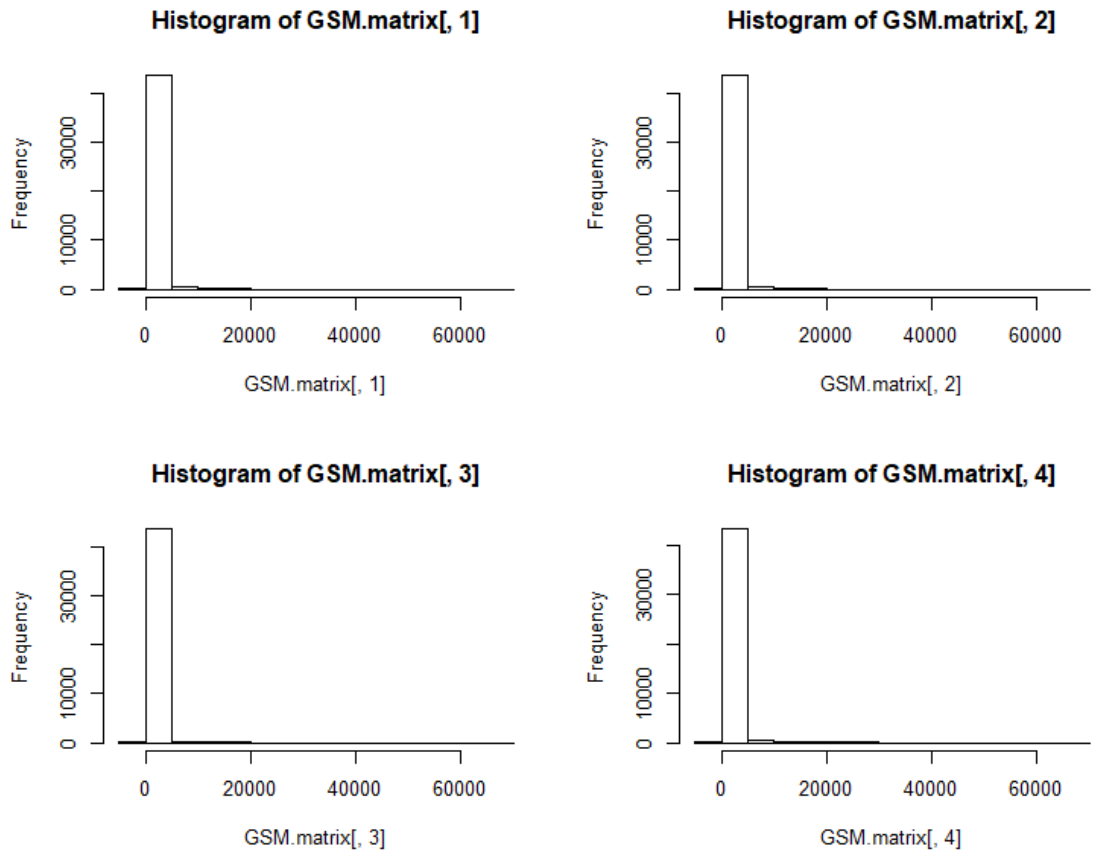
1. Subtract the foreground – background for each of the 4 chips for only the Cy5 channel. This should all be on the linear or raw scale (no logging yet).
2. Sort each column independently in this new matrix
3. Calculate row means for the sorted matrix
4. Create a new matrix with each row having the same values as the sorted row mean vectors from step #3 (you should have a new R matrix)
5. Rank the columns independently on the original background subtracted matrix (from step #1)
Hint: use the rank() function with the argument ties="first" or order()
6. Reorder the columns in the new matrix from step #4 using the ranks from step #5

```
GSM304445.sub <- GSM304445@maRf - GSM304445@maRb
GSM304446.sub <- GSM304446@maRf - GSM304446@maRb
GSM304447.sub <- GSM304447@maRf - GSM304447@maRb
GSM304448.sub <- GSM304448@maRf - GSM304448@maRb
GSM304445.sorted <- sort(GSM304445.sub)
GSM304446.sorted <- sort(GSM304446.sub)
GSM304447.sorted <- sort(GSM304447.sub)
GSM304448.sorted <- sort(GSM304448.sub)

GSM.matrix <- cbind(GSM304445.sorted, GSM304446.sorted,
                    GSM304447.sorted, GSM304448.sorted)

rowmeans <- rowMeans(GSM.matrix, na.rm = FALSE, dims = 1)
GSM.matrix <- cbind(GSM.matrix, rowmeans)
GSM304445.rank <- rank(GSM304445.sub, ties="first")
GSM304446.rank <- rank(GSM304446.sub, ties="first")
GSM304447.rank <- rank(GSM304447.sub, ties="first")
GSM304448.rank <- rank(GSM304448.sub, ties="first")

GSM.matrix <- cbind(GSM.matrix, GSM304445.rank,
              GSM304446.rank, GSM304447.rank,GSM304448.rank)
```

```
GSM.matrix[1:4,]
GSM.matrix[,1]<- GSM.matrix[order(GSM304445.rank),][,1]
GSM.matrix[,2]<- GSM.matrix[order(GSM304446.rank),][,2]
GSM.matrix[,3]<- GSM.matrix[order(GSM304447.rank),][,3]
GSM.matrix[,4]<- GSM.matrix[order(GSM304448.rank),][,4]
GSM.matrix[1:4,]
```

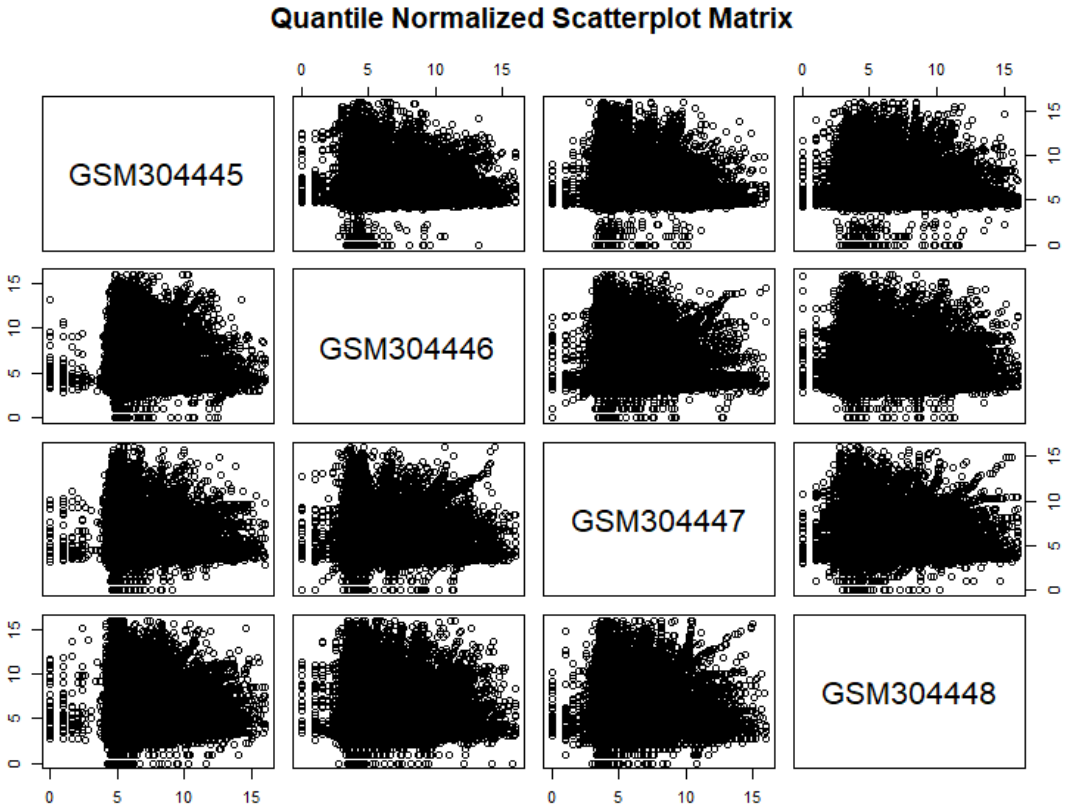
To verify that each array has the same distribution, use the hist() function to look at various arrays (e.g., hist(c5.norm[,1]); hist(c5.norm[,2]); etc.). Slight differences in distributions are a result of the ties in the ranking.

```
par(mfrow=c(2,2))
hist(GSM.matrix[,1])
hist(GSM.matrix[,2])
hist(GSM.matrix[,3])
hist(GSM.matrix[,4])
```



8.) Now log (base 2) the new R matrix you created from step 6 (question #7) and calculate a Spearman's rank correlation between the 4 arrays and plot a scatter plot matrix as you did before. Print the correlation coefficients to the screen. (5 pts)

```
GSM.matrix.log <- log2(GSM.matrix[,1:4])
GSM.matrix.log[1:4,]

GSM56.cor <- cor(as.numeric(GSM.matrix.log[,1]),
as.numeric(GSM.matrix.log[,2]),
                 method = "spearman",
use="pairwise.complete.obs")
GSM57.cor <- cor(as.numeric(GSM.matrix.log[,1]),
as.numeric(GSM.matrix.log[,3]),
                 method = "spearman",
use="pairwise.complete.obs")
GSM58.cor <- cor(as.numeric(GSM.matrix.log[,1]),
as.numeric(GSM.matrix.log[,4]),
                 method = "spearman",
use="pairwise.complete.obs")
GSM67.cor <- cor(as.numeric(GSM.matrix.log[,2]),
as.numeric(GSM.matrix.log[,3]),
                 method = "spearman",
use="pairwise.complete.obs")
GSM68.cor <- cor(as.numeric(GSM.matrix.log[,2]),
as.numeric(GSM.matrix.log[,4]),
                 method = "spearman",
use="pairwise.complete.obs")
GSM78.cor <- cor(as.numeric(GSM.matrix.log[,3]),
as.numeric(GSM.matrix.log[,4]),
                 method = "spearman",
use="pairwise.complete.obs")
""
colnames(GSM.matrix.log) <- c("GSM304445", "GSM304446",
"GSM304447", "GSM304448")
pairs(GSM.matrix.log, labels=colnames(GSM.matrix.log),
      main="Quantile Normalized Scatterplot Matrix")
```




```
quantile.norm.table <- rbind(c(1.0, GSM56.cor, GSM57.cor, GSM58.cor),
                             c(GSM56.cor, 1.0, GSM67.cor, GSM68.cor),
                             c(GSM57.cor, GSM67.cor, 1.0, GSM78.cor),
                             c(GSM58.cor, GSM68.cor, GSM78.cor, 1.0))
```

```
quantile.norm.table
```

```
           [,1]        [,2]        [,3]        [,4]
[1,]  1.00000000 -0.03025796 -0.01047010 -0.04891640
[2,] -0.03025796  1.00000000  0.09018573  0.05795112
[3,] -0.01047010  0.09018573  1.00000000  0.04390790
[4,] -0.04891640  0.05795112  0.04390790  1.00000000
```

9.) Of the 4 normalization methods, which do you suggest as optimal and why? (2.5 pts)

I would still use the loess or print loess method that we performed in #2. The normalization seems to be the most accurate and the one that we performed by hand in #8 is time consuming and prone to error, as you can see.

10.) Now we want to work with a qRT-PCR dataset from patients with an inflammatory disease. The genes measured for this experiment included a set of proinflammatory chemokines and cytokines that are related to the disease. Download the raw qRT-PCR file called Inflammation_qRT-PCR.csv. Then change the

normalization script from the lecture notes to include the housekeeping genes beta actin, GAPH, and 18S. Look at the file to make sure the housekeepers are spelled correctly.

Run the normalization script and output a data matrix of fold change values.

```
# qt-PCR
f.parse <- function(path=pa,file=fi,out=out.fi) {
  d <-
read.table(paste(path,file,sep=""),skip=11,sep=",",header=T)
  u <- as.character(unique(d$Name))
  u <- u[u!=""]; u <- u[!is.na(u)];
  ref <- unique(as.character(d$Name[d$Type=="Reference"]))
  u <- unique(c(ref,u))
  hg <- c("B-ACTIN","GAPDH","18S")
  hg <- toupper(hg)
  p <- unique(toupper(as.character(d$Name.1)))
  p <- sort(setdiff(p,c("",hg)))

  mat <- matrix(0,nrow=length(u),ncol=length(p))
  dimnames(mat) <- list(u,p)
  for (i in 1:length(u)) {
    print(paste(i,": ",u[i],sep=""))
    tmp <- d[d$Name %in% u[i],c(1:3,6,9)]
    g <- toupper(unique(as.character(tmp$Name.1)))
    g <- sort(setdiff(g,c("",hg)))

    for (j in 1:length(g)) {
      v <- tmp[toupper(as.character(tmp$Name.1)) %in% g[j],5]
      v <- v[v!=999]
      v <- v[((v/mean(v))<1.5) & ((v/mean(v))>0.67)]   #gene j
vector

      hv3 <- NULL
      for (k in 1:length(hg)) {  #housekeeping gene vector (each
filtered by reps)
        hv <- tmp[toupper(as.character(tmp$Name.1)) %in% hg[k],5]
        hv <- hv[hv!=999]
        hv3 <- c(hv3,hv[((hv/mean(hv))<1.5) &
((hv/mean(hv))>0.67)])
      }
      sv <- mean(as.numeric(v)) - mean(as.numeric(hv3))
     #scaled value for gene j

      if(i==1) { #reference sample only
        mat[u[i],g[j]] <- sv
        next
      }
```

```
      mat[u[i],g[j]] <- sv - mat[u[1],g[j]]
    }
  }

  mat[1,][!is.na(mat[1,])] <- 0
  fc <- 2^(-1 * mat)

write.table(t(c("Subject",dimnames(mat)[[2]])),paste(path,out,sep
=""),quote=F,sep="\t",col.names=F,row.names=F)

write.table(round(fc,3),paste(path,out,sep=""),quote=F,sep="\t",a
ppend=T,col.names=F)
}


# run function
pa <- "C:\\temp\\datasets"
fi <- "qRT-PCR.CSV"
out.fi <- "foldchange.txt“

f.parse(pa,fi,out.fi)
```

I attached the output file to the course message.

11.) Read the normalized qRT-PCR data matrix into R, using a Spearman's rank correlation, which two patients are most correlated? Plot these two patients against each other in a scatter plot. (3 pts)

```
data = read.table("c:\\temp\\datasets\\foldchange.txt", header=T,
row.names=1)
```

I could not read in this data and did not have time to debug the f.parse function. I keep getting this error on the foldchange.txt data..

```
Error in scan(file = file, what = what, sep = sep, quote = quote, dec =
dec,  : line 1 did not have 36 elements
```