Homework 2

Kevin Chen

7/10/2020

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)

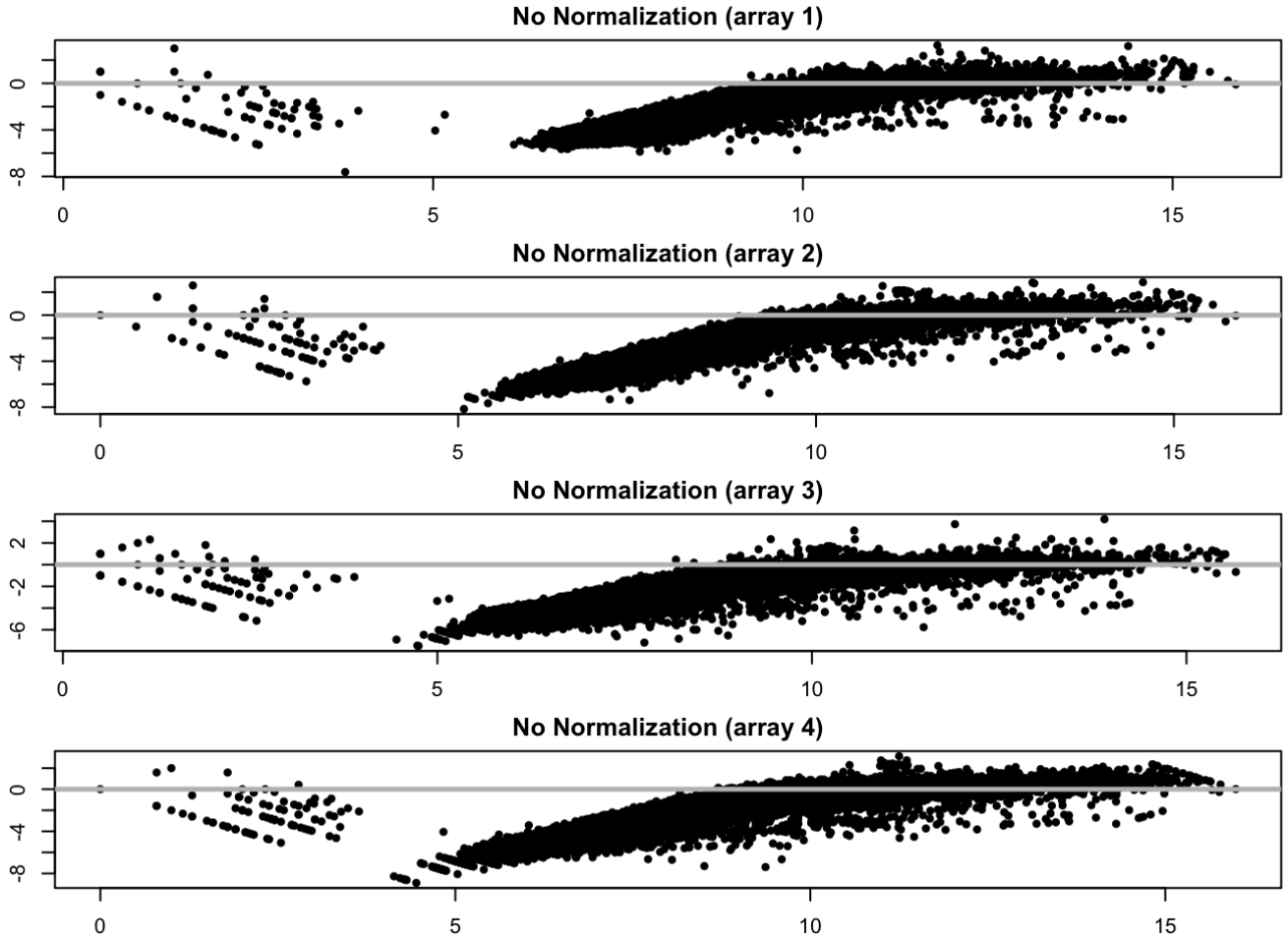
```
library(marray)
dir.path <- "/Users/kevinchen/Documents/MS Bioinformatic program courses and supplement courses/summer 2020/Gene
 Expression Data Analysis and Visualization/Homework Assignments/Homework 2/GSE12050_amend.gpix"
gse <- read.GenePix(path=dir.path,name.Gf = "F532 Median",name.Gb ="B532 Median", name.Rf = "F635 Median", name.R
b = "B635 Median",name.W ="Flags")
```

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)

```
library(marray)
```

```
## Loading required package: limma
```

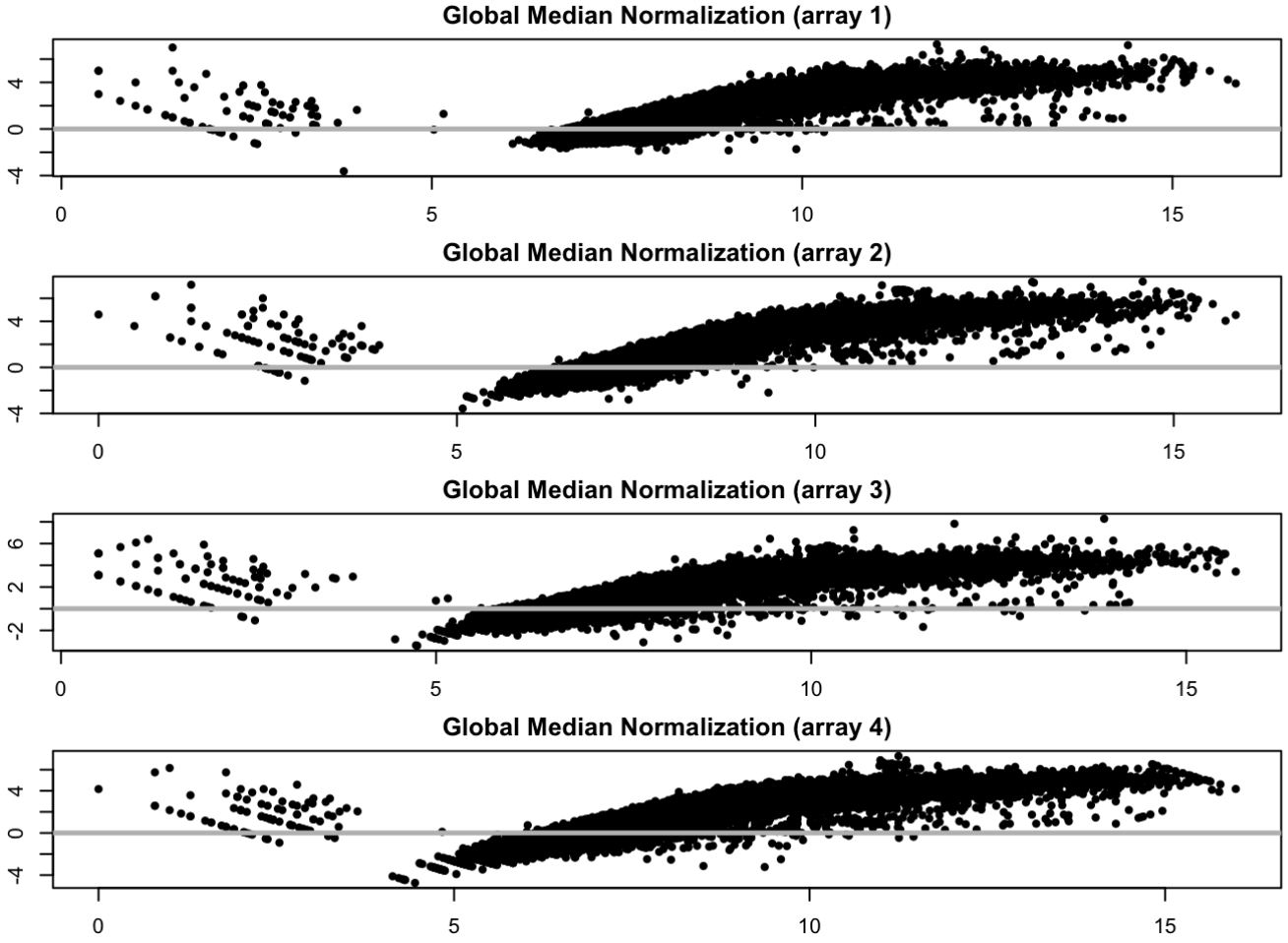
```
load("/Users/kevinchen/Documents/MS Bioinformatic program courses and supplement courses/summer 2020/Gene Expression Data Analysis and Visualization/Homework Assignments/Homework 2/gse.rda")
median.global<-maNorm(gse,norm="median",span=0.45)
loess<-maNorm(gse,norm="loess",span=0.45)
print.tip<-maNorm(gse,norm="printTipLoess",span=0.45)
par(mfrow=c(4,1),mar=c(2,2,2,2))
sapply(1:4,function(x) {maPlot(gse[,x],main=paste0("No Normalization (array ",x,")"),lines.func = NULL, legend.func=NULL)})
```




```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## [[3]]
## NULL
##
```

```
## [[4]]
## NULL
```

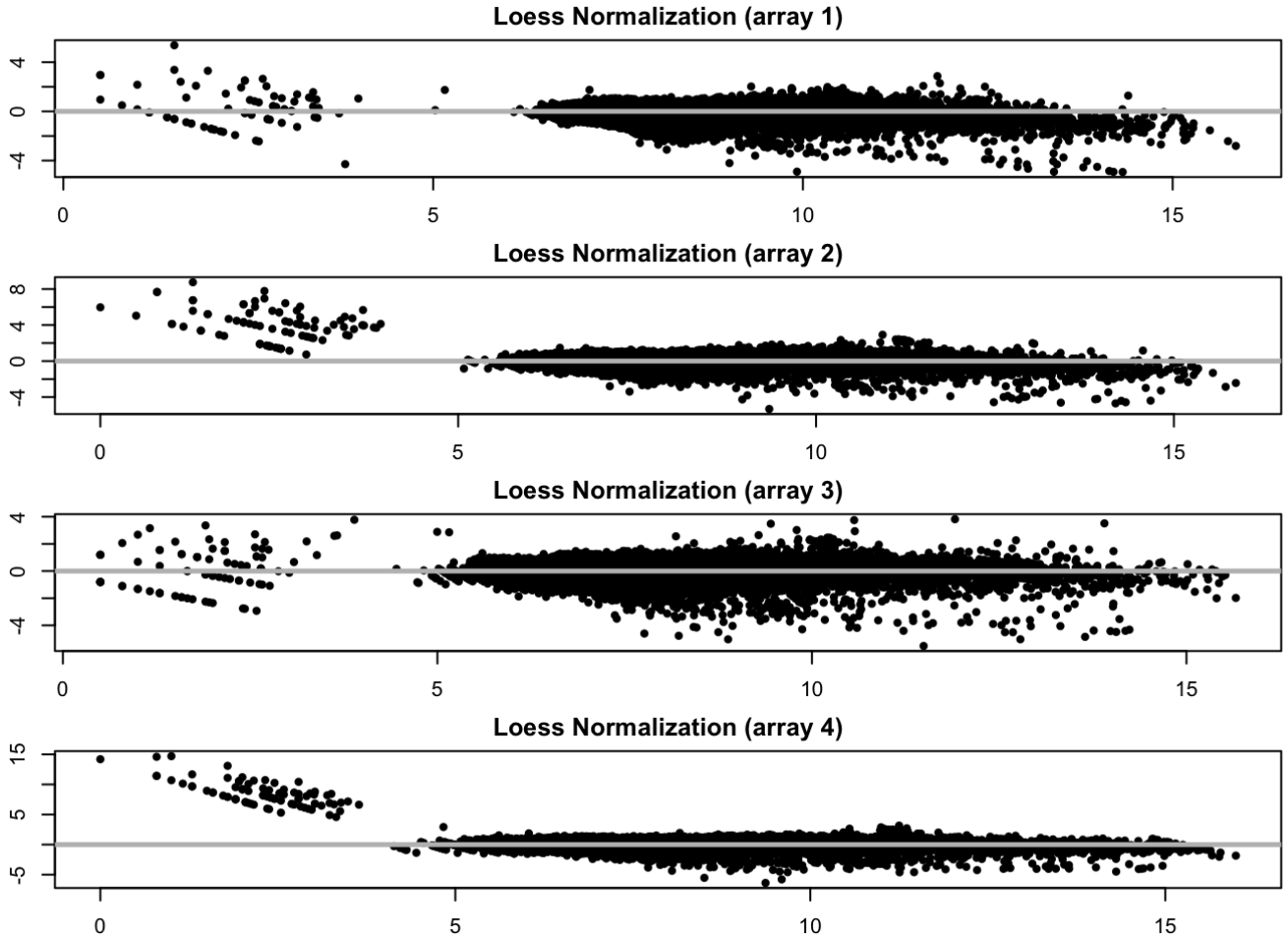
```
par(mfrow=c(4,1))
sapply(1:4,function(x) {maPlot(median.global[,x],main=paste0("Global Median Normalization (array ",x,")"),lines.f
unc = NULL,legend.func=NULL)})
```




```
## [[1]]
## NULL
```

```
##
## [[2]]
## NULL
##
## [[3]]
## NULL
##
## [[4]]
## NULL
```

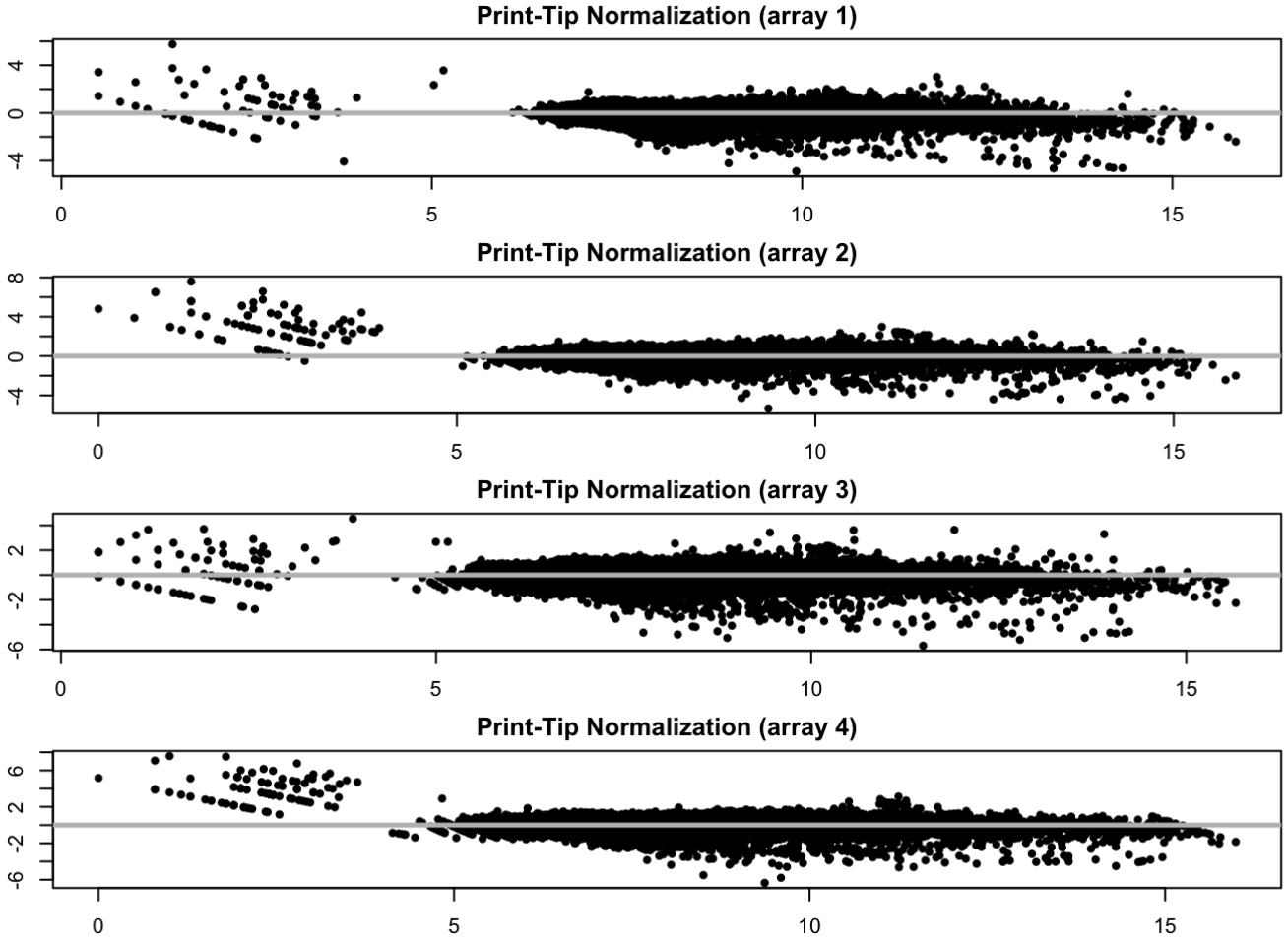
```
par(mfrow=c(4,1))
sapply(1:4,function(x) {maPlot(loess[,x],main=paste0("Loess Normalization (array ",x,")"),lines.func = NULL,legend.func=NULL)})
```




```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## [[3]]
## NULL
##
```

```
## [[4]]
## NULL
```

```
par(mfrow=c(4,1))
sapply(1:4,function(x) {maPlot(print.tip[,x],main=paste0("Print-Tip Normalization (array ",x,")"),lines.func = NU
LL,legend.func=NULL)})
```

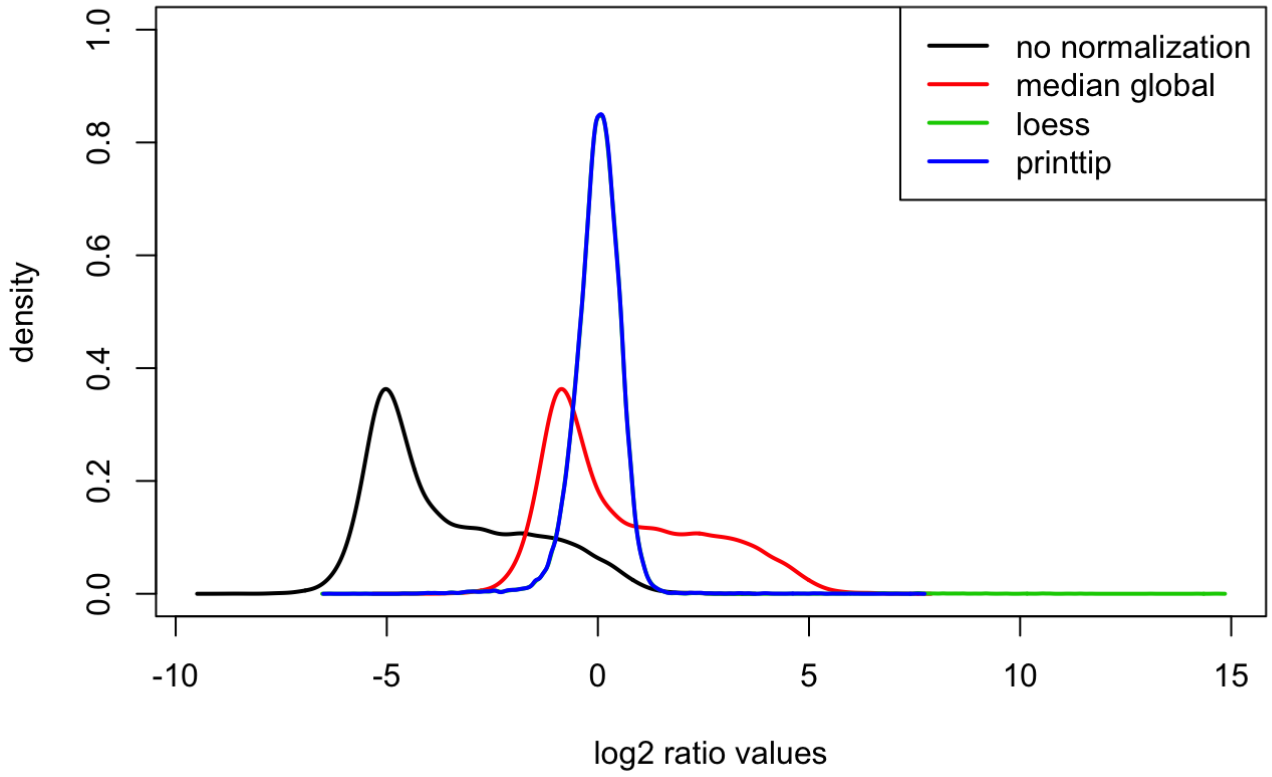



```
## [[1]]
## NULL
```

```
##
## [[2]]
## NULL
##
## [[3]]
## NULL
##
## [[4]]
## NULL
```

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)

```
log.ratio.raw<-as.numeric(maM(gse[,4]))
log.ratio.median<-as.numeric(maM(median.global[,4]))
log.ratio.loess<-as.numeric(maM(loess[,4]))
log.ratio.printtip<-as.numeric(maM(print.tip[,4]))
raw.density<-density(log.ratio.raw,na.rm=T)
median.density<-density(log.ratio.median,na.rm=T)
loess.density<-density(log.ratio.loess,na.rm=T)
printtip.density<-density(log.ratio.printtip,na.rm=T)
dat<-list(raw.density,median.density,loess.density,printtip.density)
plot(x=range(raw.density$x,median.density$x,loess.density$x,printtip.density$xty$x),y=range(0,1),xlab="log2 ratio
 values", ylab="density",type="n")
for(i in 1:length(dat)){
  dat.y<-dat[[i]]$y
  dat.x<-dat[[i]]$x
  lines(dat.x,dat.y,col=i,lwd=2)
}
legend(x="topright",legend=c("no normalization","median global","loess","printtip"),col=1:4,lwd=2)
```



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

```
# both Loess and Print-tip are preferred over raw and median global normalization. Loess and Print-tip seem to be close in performance according to the plots in question 3. However, Print-Tip seems to be doing slightly better than Loess in normalizing values outside of the "main cluster". Thus, the most preferred normalization is Print-Tip
```

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 log2 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)

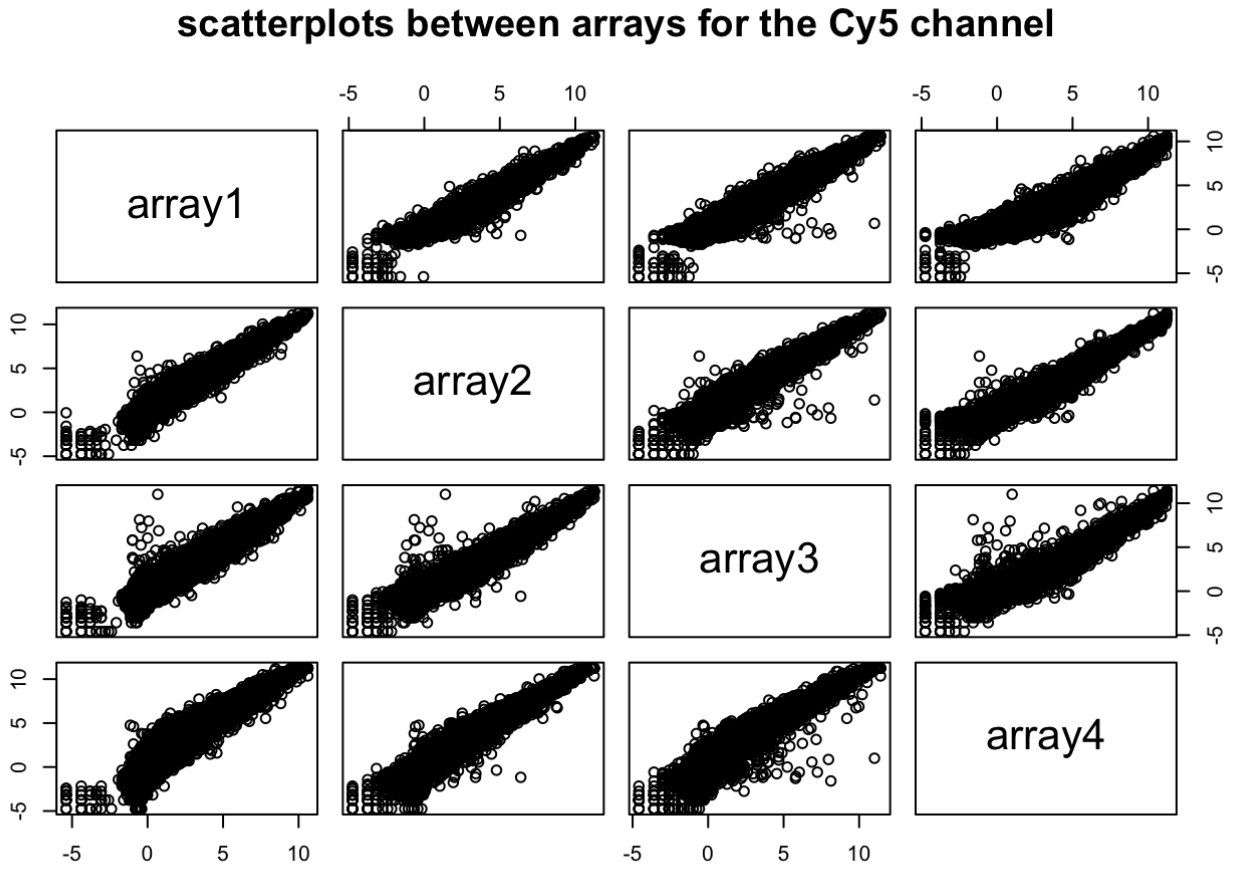
```
Cy5.foreground<-maRf(gse)
Cy5.background<-maRb(gse)
Cy5.background.corrected<-Cy5.foreground-Cy5.background
new.colnames<-gsub("/Users/kevinchen/Documents/MS Bioinformatic program courses and supplement courses/summer 202
0/Gene Expression Data Analysis and Visualization/Homework Assignments/Homework 2/GSE12050_amend.gpix/","",colnam
es(Cy5.background.corrected))
colnames(Cy5.background.corrected)<-new.colnames
Cy5.log2<-log2(Cy5.background.corrected)
```

```
## Warning: NaNs produced
```

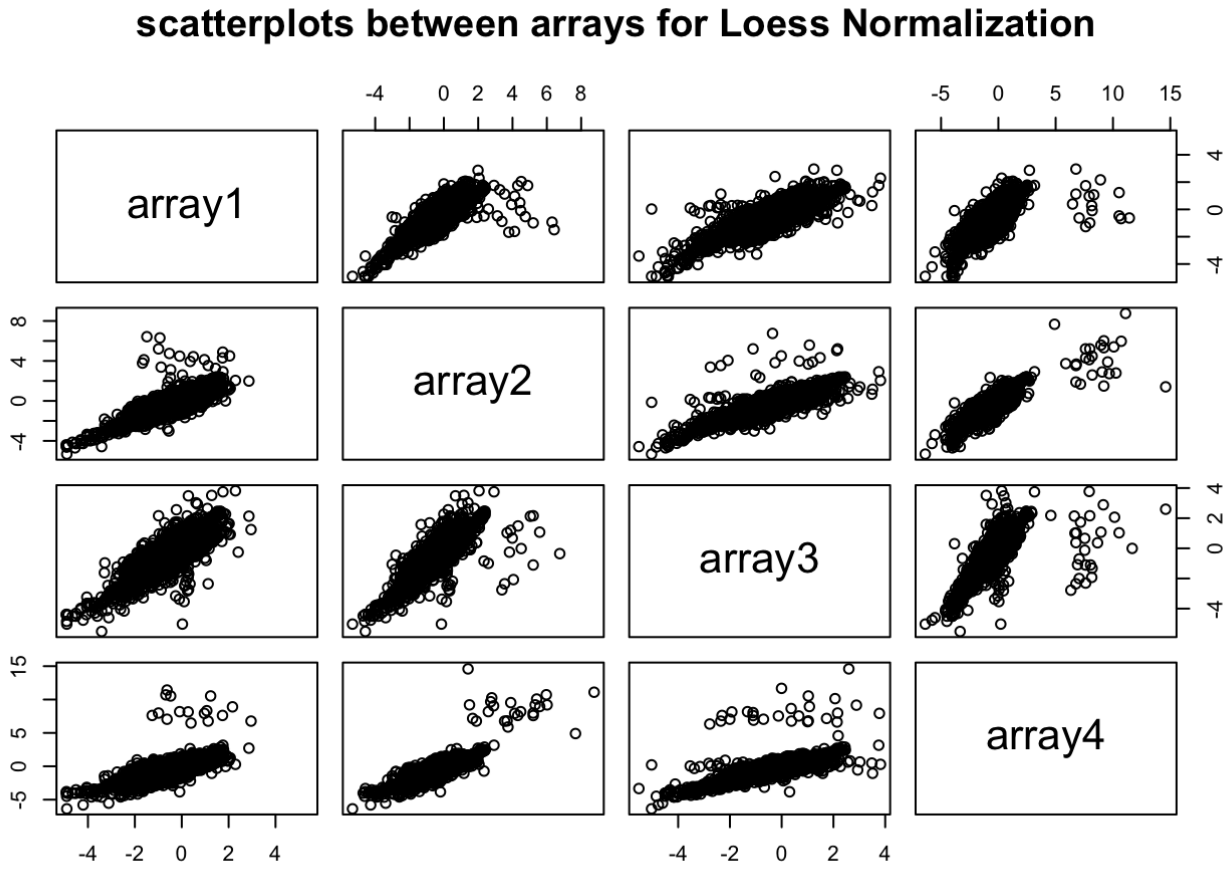
```
medians<-apply(Cy5.log2,2,median, na.rm=T)
single.channel.result<-sweep(Cy5.log2,2,medians,"-")
```

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)

```
single.channel.corr<-cor(single.channel.result,method="spearman",use="pairwise.complete.obs")
loess.m.values<-maM(loess)
new.colnames<-gsub("/Users/kevinchen/Documents/MS Bioinformatic program courses and supplement courses/summer 202
0/Gene Expression Data Analysis and Visualization/Homework Assignments/Homework 2/GSE12050_amend.gpix/","",colnam
es(loess.m.values))
colnames(loess.m.values)<-new.colnames
loess.corr<-cor(loess.m.values,method="spearman", use="pairwise.complete.obs")
pairs(single.channel.result[,1:4], label=paste0("array",c(1:4)),main="scatterplots between arrays for the Cy5 cha
nnel")
```




```
pairs(loess.m.values[,1:4],label=paste0("array",c(1:4)),main="scatterplots between arrays for Loess Normalizatio
n")
```




```
print(single.channel.corr)
```

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

```
print(loess.corr)
```

```
##               GSM304445.gpr GSM304446.gpr GSM304447.gpr GSM304448.gpr
## GSM304445.gpr     1.0000000     0.6989116     0.7531842     0.6960390
## GSM304446.gpr     0.6989116     1.0000000     0.7274441     0.7116478
## GSM304447.gpr     0.7531842     0.7274441     1.0000000     0.7478431
## GSM304448.gpr     0.6960390     0.7116478     0.7478431     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).

```
Cy5.foreground<-maRf(gse)
Cy5.background<-maRb(gse)
Cy5.background.corrected<-Cy5.foreground-Cy5.background
new.colnames<-gsub("/Users/kevinchen/Documents/MS Bioinformatic program courses and supplement courses/summer 202
0/Gene Expression Data Analysis and Visualization/Homework Assignments/Homework 2/GSE12050_amend.gpix/","",colnam
es(Cy5.background.corrected))
```

2. Sort each column independently in this new matrix

```
Cy5.sorted<-apply(Cy5.background.corrected,2,sort)
```

3. Calculate row means for the sorted matrix

```
Cy5.means<-apply(Cy5.sorted,1,mean)
```

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)

```
Cy5.means.matrix<-matrix(rep(Cy5.means,each=ncol(Cy5.background.corrected)),nrow = length(Cy5.means),byrow = T)
colnames(Cy5.means.matrix)<-new.colnames
```

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()

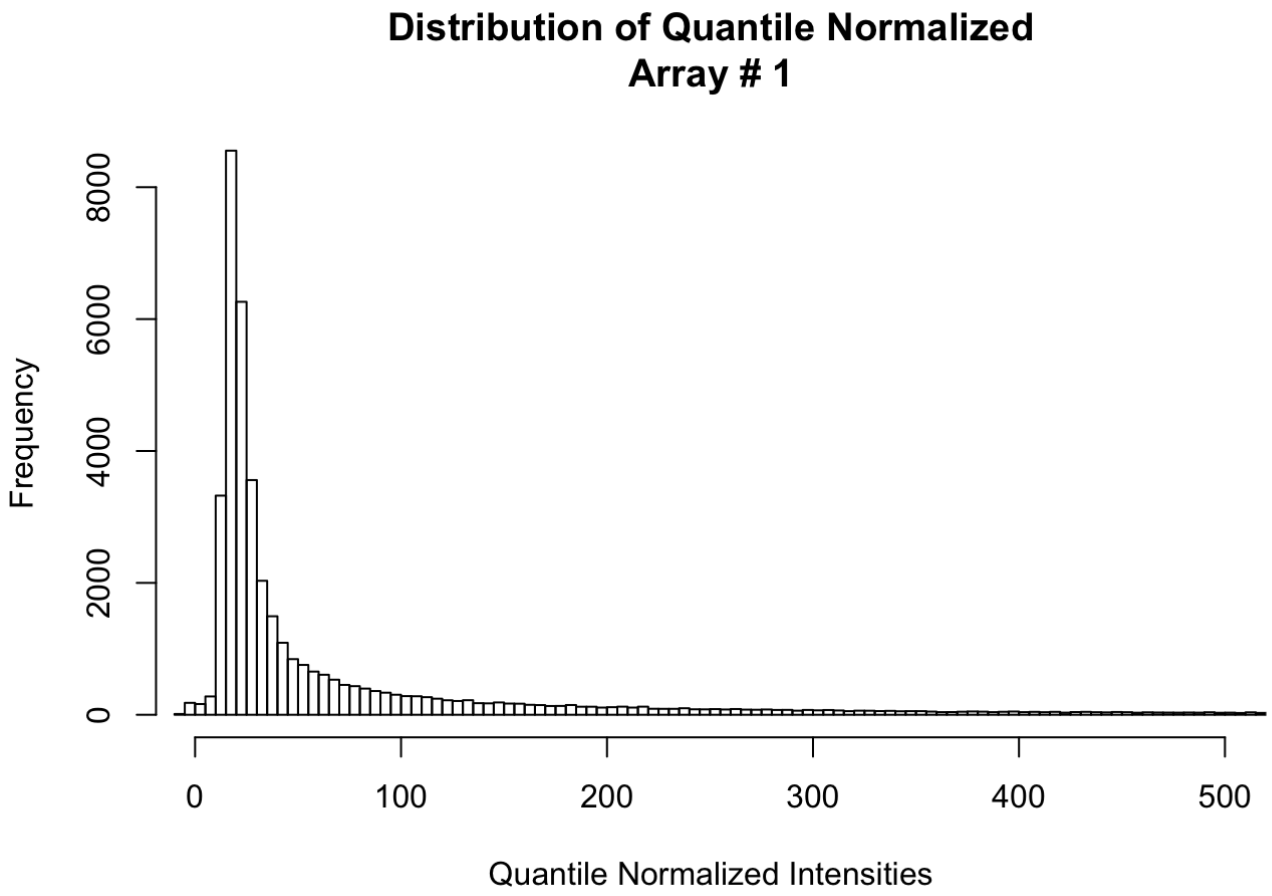
```
Cy5.ranked<-apply(Cy5.background.corrected,2,rank,ties="first")
```

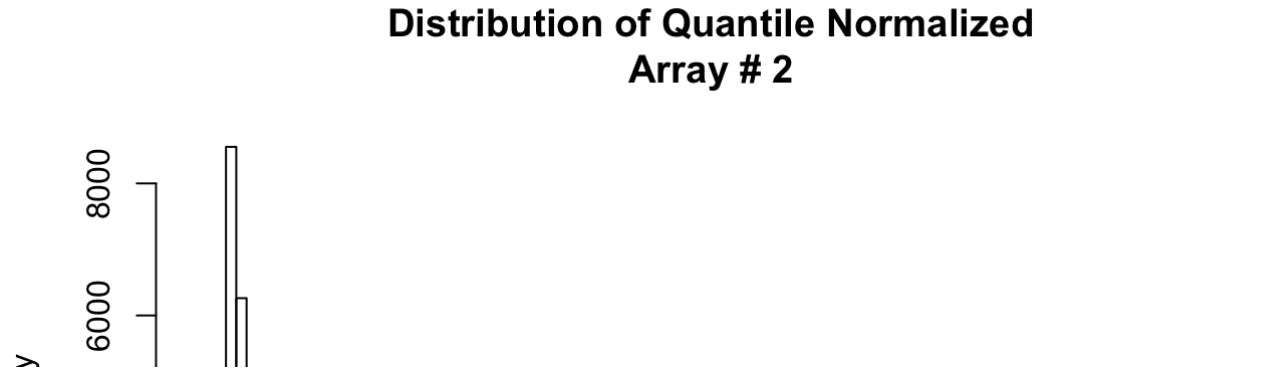
6. Reorder the columns in the new matrix from step #4 using the ranks from step #5

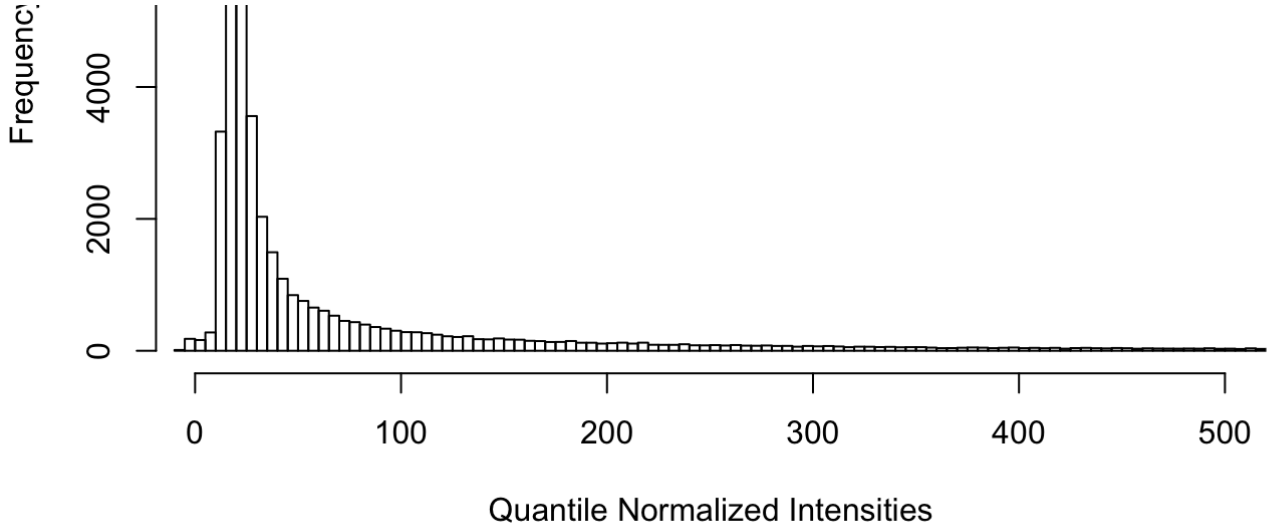
```
for(i in 1:ncol(Cy5.means.matrix)){
  Cy5.means.matrix[,i]<-Cy5.means.matrix[Cy5.ranked[,i],i]
}
```

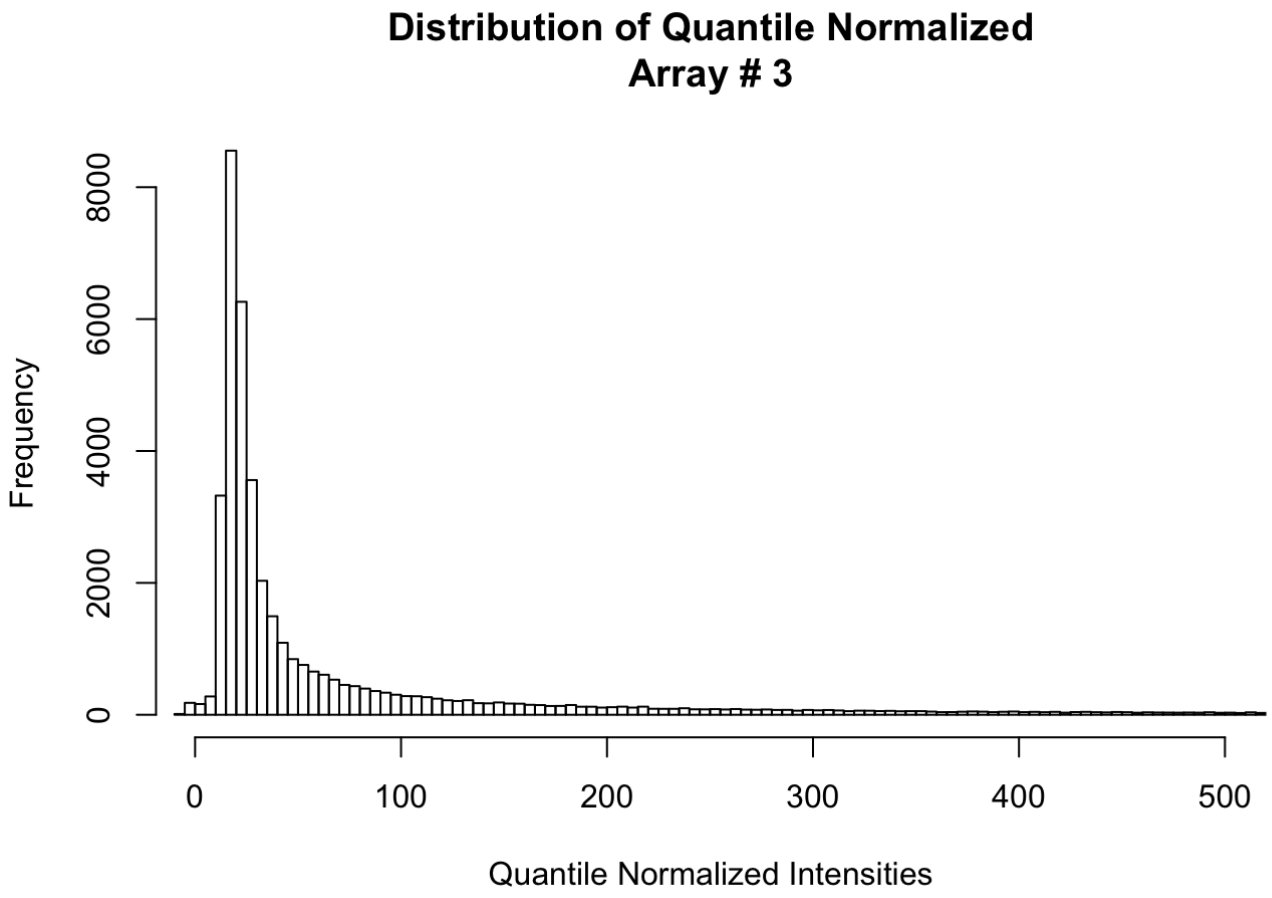
To verify that each array has the same distribution, use the hist() function to look at various arrays (e.g., hist(c5.norm[,1]); hist(c5.norm[,2]); etc.). Slight differences in distributions are a result of the ties in the ranking.

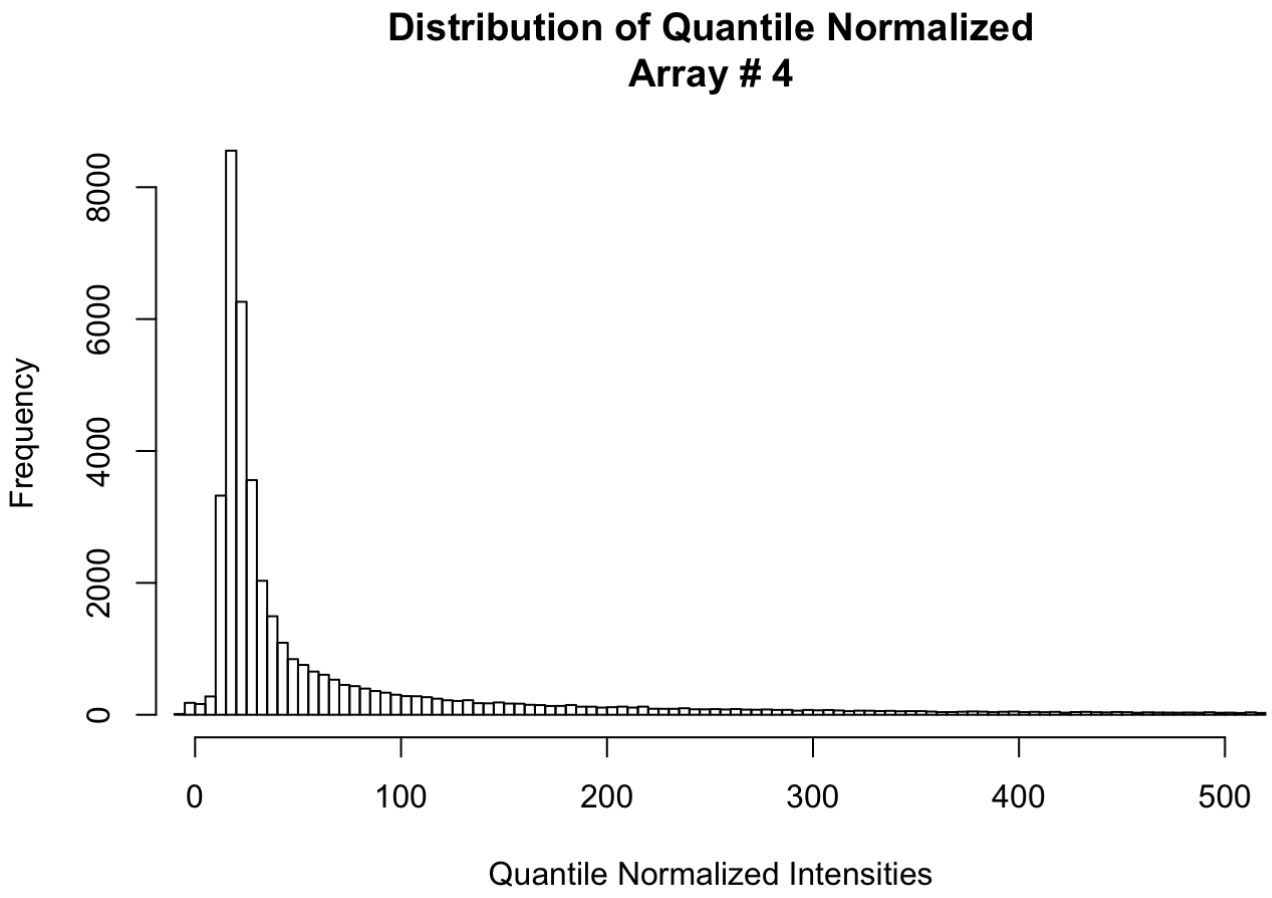
```
for(i in 1:ncol(Cy5.means.matrix)){
  hist(Cy5.means.matrix[,i],xlab="Quantile Normalized Intensities",main=paste0("Distribution of Quantile Normaliz
ed\nArray # ",i),freq=T,xlim=range(1,500),n=20000)}
```









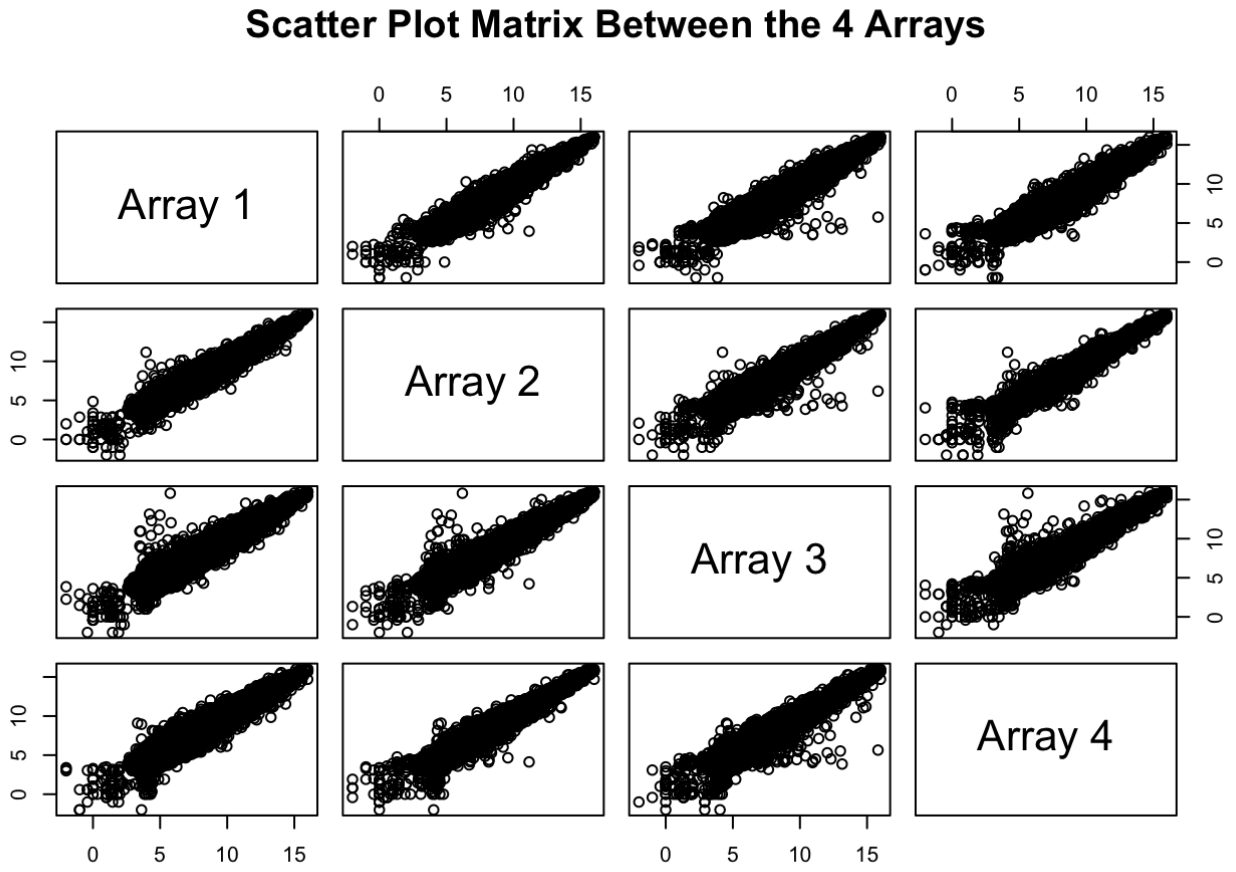


8.) Now log (base 2) the new R matrix you created from step 6 (question #7) and calculate a Spearman's rank correlation between the 4 arrays and plot a scatter plot matrix as you did before. Print the correlation coefficients to the screen. (5 pts)

```
log2.Cy5.q.norm<-log2(Cy5.means.matrix)
```

```
## Warning: NaNs produced
```

```
q.norm.corr<-cor(log2.Cy5.q.norm,method="spearman",use="pairwise.complete.obs")
pairs(log2.Cy5.q.norm,label=paste0("Array ",1:ncol(Cy5.means.matrix)),main="Scatter Plot Matrix Between the 4 Arr
ays")
```




```
print(q.norm.corr)
```

```
##               GSM304445.gpr GSM304446.gpr GSM304447.gpr GSM304448.gpr
## GSM304445.gpr     1.0000000     0.8957944     0.8784166     0.8984337
```

```
## GSM304446.gpr     0.8957944     1.0000000     0.8772443     0.9077429
## GSM304447.gpr     0.8784166     0.8772443     1.0000000     0.8846636
## GSM304448.gpr     0.8984337     0.9077429     0.8846636     1.0000000
```

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

```
# Of the 4 normalization methods, quantile normalization is the most optimal in my opinion because it allows for
 each array to have the same distribution and it allows for better comparison of biological differences between a
rrays. It is very effective in normalizing the arrays and reducing the variances across arrays. As the plots in q
uestion 8 have shown, the correlations between the replicates were very high after quantile normalization and it
 seems to do better than Loess and global median normalizaiton based on the plots. It is also a computationally e
fficient method.
```

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.

```
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))
    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",
    append = T,
    col.names = F
  )
}
# run function
pa <- "/Users/kevinchen/Documents/MS Bioinformatic program courses and supplement courses/summer 2020/Gene Expres
sion Data Analysis and Visualization/Homework Assignments/Homework 2/"
fi <- "qRT-PCR (1).csv"
out.fi <- "fold_chg_matrix.txt"
f.parse(pa, fi, out.fi)
```

```
## [1] "1: 1"
## [1] "2: 8"
## [1] "3: 9"
## [1] "4: 2"
## [1] "5: 7"
## [1] "6: 10"
## [1] "7: 3"
## [1] "8: 6"
## [1] "9: 4"
## [1] "10: 5"
```

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)

```
pcr.norm<-read.table("fold_chg_matrix.txt",header=T,sep="\t",row.names=1)
pcr.norm<-t(pcr.norm)
pcr.norm.corr<-cor(pcr.norm,method="spearman",use="pairwise.complete.obs")
```

```
## Warning in cor(pcr.norm, method = "spearman", use = "pairwise.complete.obs"):
## the standard deviation is zero

## Warning in cor(pcr.norm, method = "spearman", use = "pairwise.complete.obs"):
## the standard deviation is zero

## Warning in cor(pcr.norm, method = "spearman", use = "pairwise.complete.obs"):
## the standard deviation is zero

## Warning in cor(pcr.norm, method = "spearman", use = "pairwise.complete.obs"):
## the standard deviation is zero

## Warning in cor(pcr.norm, method = "spearman", use = "pairwise.complete.obs"):
## the standard deviation is zero

## Warning in cor(pcr.norm, method = "spearman", use = "pairwise.complete.obs"):
## the standard deviation is zero

## Warning in cor(pcr.norm, method = "spearman", use = "pairwise.complete.obs"):
## the standard deviation is zero

## Warning in cor(pcr.norm, method = "spearman", use = "pairwise.complete.obs"):
## the standard deviation is zero

## Warning in cor(pcr.norm, method = "spearman", use = "pairwise.complete.obs"):
## the standard deviation is zero

## Warning in cor(pcr.norm, method = "spearman", use = "pairwise.complete.obs"):
## the standard deviation is zero
```

```
pcr.norm.corr
```

```
##     1         8         9         2         7        10         3         6
## 1  NA        NA        NA        NA        NA        NA        NA        NA
## 8  NA 1.0000000 0.9578446 0.8706012 0.9578446 0.9374026 0.9747067 0.9615103
```

```
## 9  NA 0.9578446 1.0000000 0.9255865 0.9556452 0.9694803 0.9644428 0.9571114
## 2  NA 0.8706012 0.9255865 1.0000000 0.9318182 0.9687471 0.9167889 0.9321848
## 7  NA 0.9578446 0.9556452 0.9318182 1.0000000 0.9617817 0.9875367 0.9956012
## 10 NA 0.9374026 0.9694803 0.9687471 0.9617817 1.0000000 0.9588489 0.9588489
## 3  NA 0.9747067 0.9644428 0.9167889 0.9875367 0.9588489 1.0000000 0.9882698
## 6  NA 0.9615103 0.9571114 0.9321848 0.9956012 0.9588489 0.9882698 1.0000000
## 4  NA 0.8845308 0.9417155 0.9692082 0.9384164 0.9645312 0.9255865 0.9435484
## 5  NA 0.9028592 0.9325513 0.9695748 0.9560117 0.9634314 0.9486804 0.9582111
##            4         5
## 1         NA        NA
## 8  0.8845308 0.9028592
## 9  0.9417155 0.9325513
## 2  0.9692082 0.9695748
## 7  0.9384164 0.9560117
## 10 0.9645312 0.9634314
## 3  0.9255865 0.9486804
## 6  0.9435484 0.9582111
## 4  1.0000000 0.9791056
## 5  0.9791056 1.0000000
```

```
# Patient 6 and 7 are the most correlated.
plot(pcr.norm[,6],pcr.norm[,7])
```