Homework Assignment # 3

Group 4: Ziheng Cheng, Krithika Krishnan, Ziqiang Chen
STA 525: Statistics For Bioinformatics

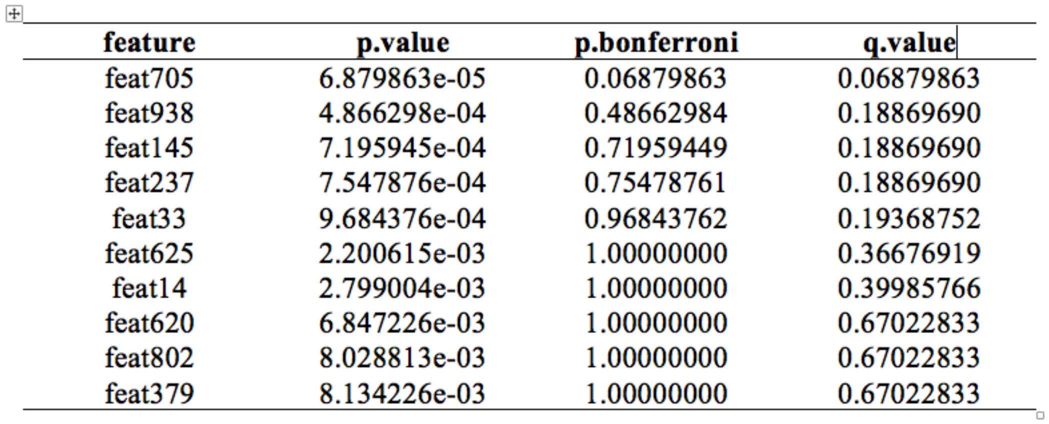
April 27, 2016

Problem 1

Summary of Biomarker Challenge

The Biomarker Challenge is basically a mimic experimental design activity through which the idea of balancing limited research funds with signi

cant statistical results has been realized. During the activity, each group had been assigned a given amount of funding used to purchase some array data on individual patients as well as validation assay data. The goal is to identify targets that are differentially expressed across two different patient samples from two populations (control and treatment). A total of 1000 features,have been compared.



feature	p.value	p.bonferroni	q.value
feat705	6.879863e-05	0.06879863	0.06879863
feat938	4.866298e-04	0.48662984	0.18869690
feat145	7.195945e-04	0.71959449	0.18869690
feat237	7.547876e-04	0.75478761	0.18869690
feat33	9.684376e-04	0.96843762	0.19368752
feat625	2.200615e-03	1.00000000	0.36676919
feat14	2.799004e-03	1.00000000	0.39985766
feat620	6.847226e-03	1.00000000	0.67022833
feat802	8.028813e-03	1.00000000	0.67022833
feat379	8.134226e-03	1.00000000	0.67022833

Figure 1: Top 10 feature selected

Principle components were used to visualize the array data for the 20 individuals per population. The two groups did not separate out completely with these three components.

The normalized array data was received and viewed using the first three principle components, as illustrated in the plot above. The scatter plot demonstrates that the two groups could not be completely separated.

Problem 2

For this problem, we first loaded the ArrayData and tried various distance metrics and clustering methods along with different K values.

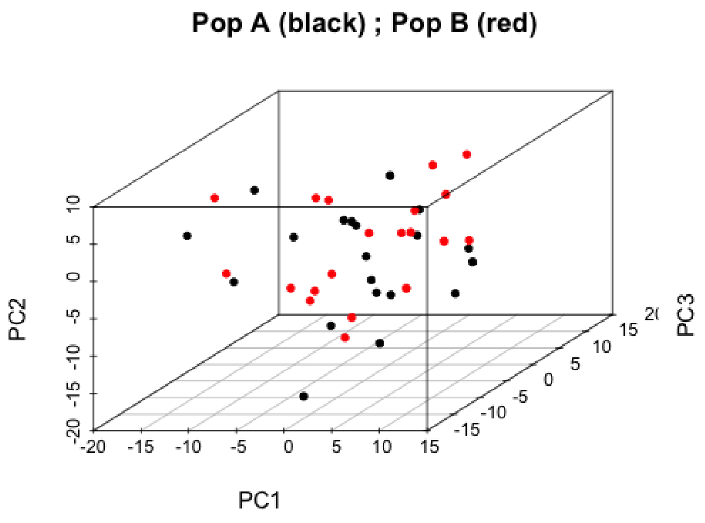


Figure 2: 3D scatter plot of PCA

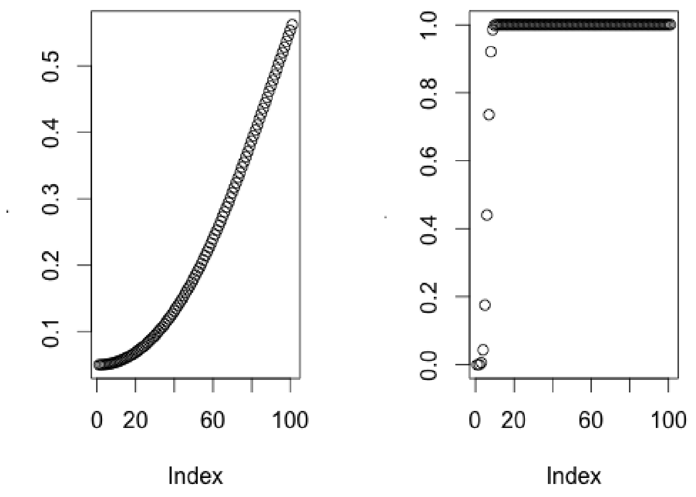


Figure 3: Power Analysis

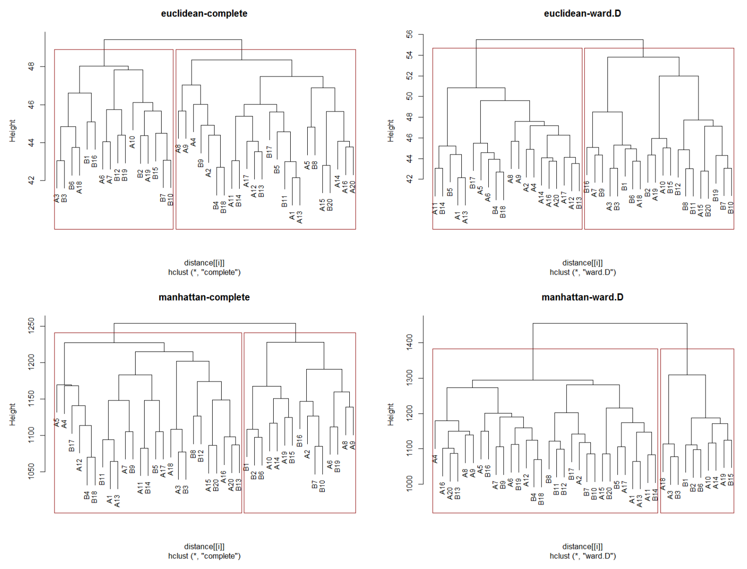


Figure 4: Heirarchial Clustering with two different distance metrics, Euclidean & Manhattan

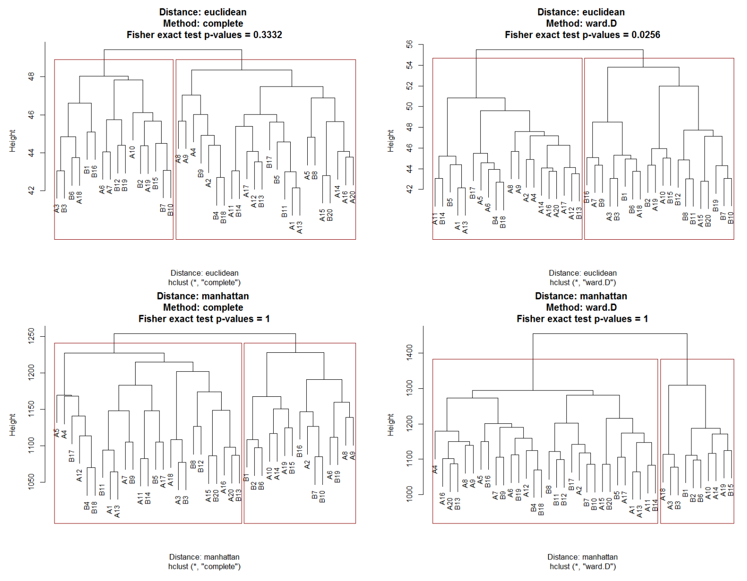


Figure 5: Heirarchial Clustering

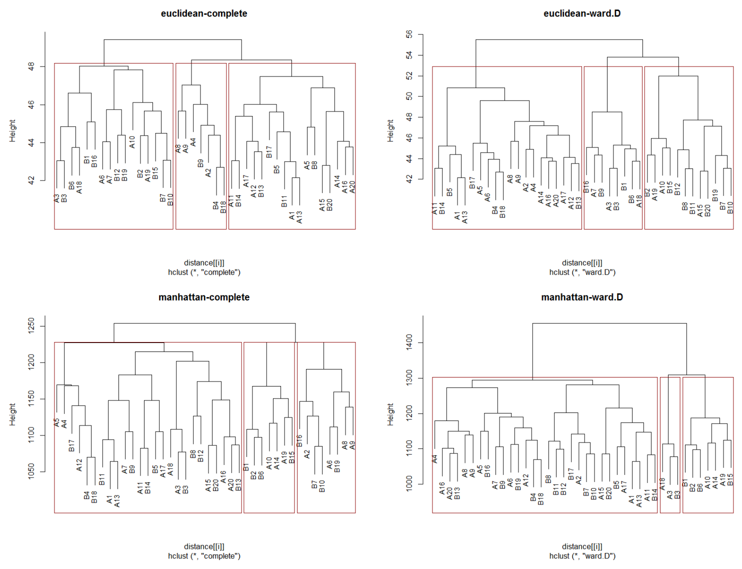


Figure 6: Heirarchial Clustering

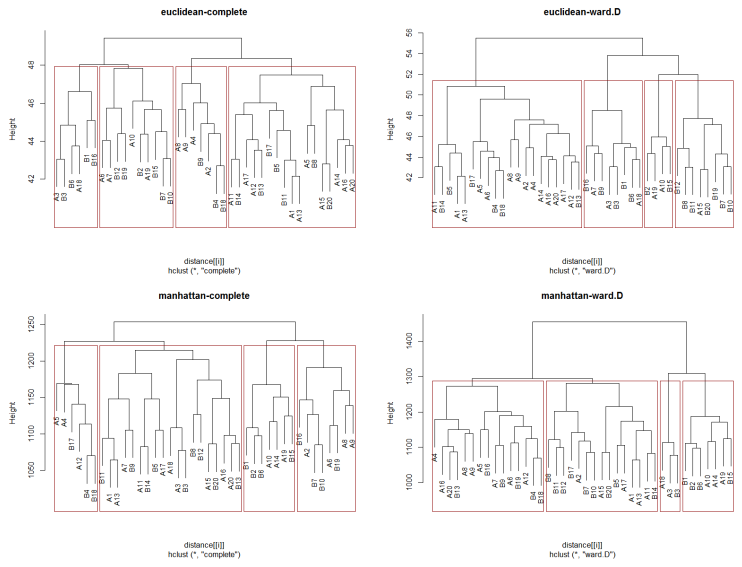


Figure 7: Heirarchial Clustering

Problem 3

The GeneMap was observed. On clustering the features and plotting them with different k values, we can observe that the gene 10 & 11 are largely distributed in cluster 1. Gene 12 was majorly located in cluster 4. Also, the other gene set was also located in certain cluster. With different k values, it was found that gene 10 was located in cluster 1 and gene 11 was located in cluster 2. Examining the features that we identified for validation, it is observed that the first validated feat is located in the 33rd set, and all other feats are located in the 1st set in GeneMap.

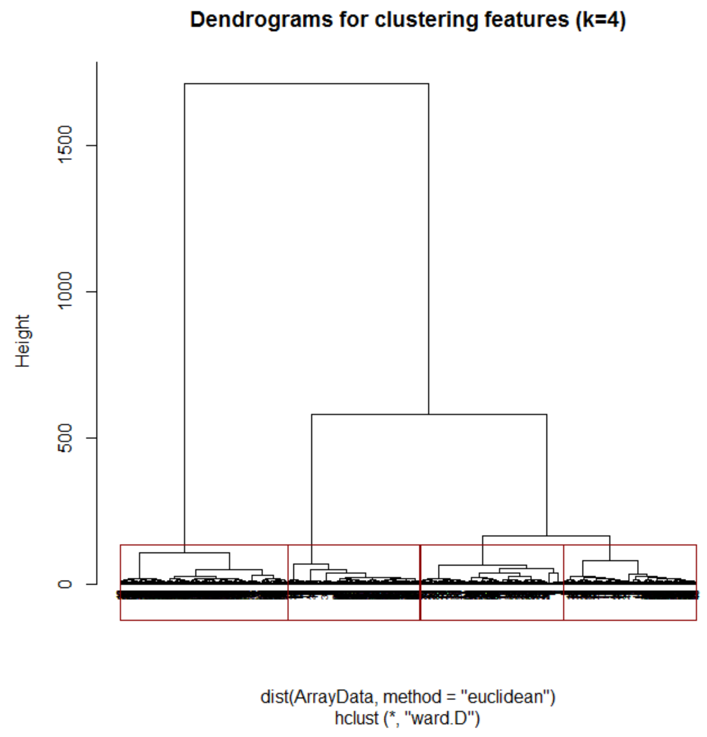
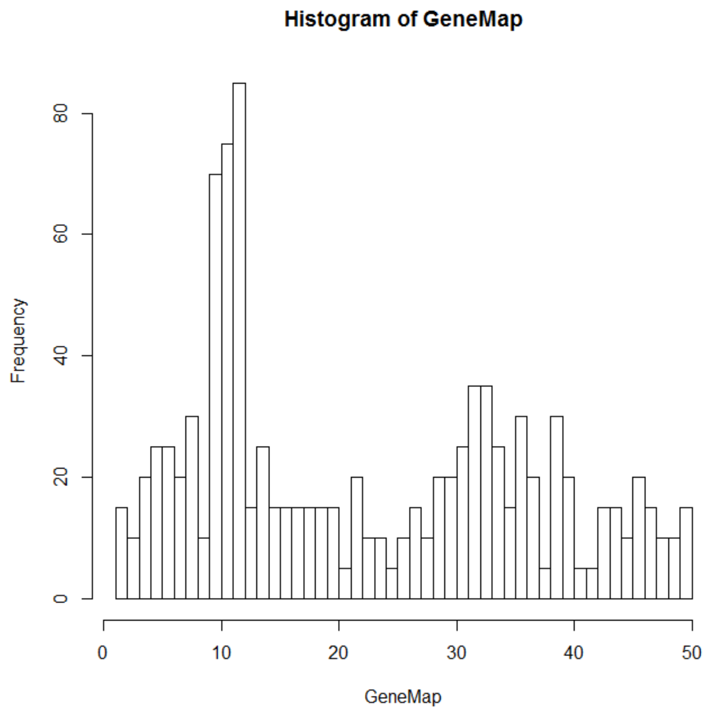


Figure 8: Heirarchial Clustering:dendogram



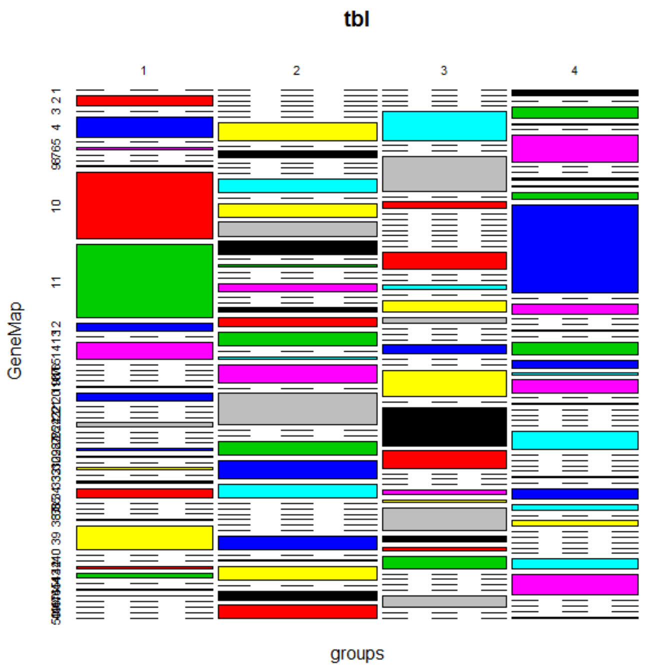


Figure 9: Mosaic Plot,k=4

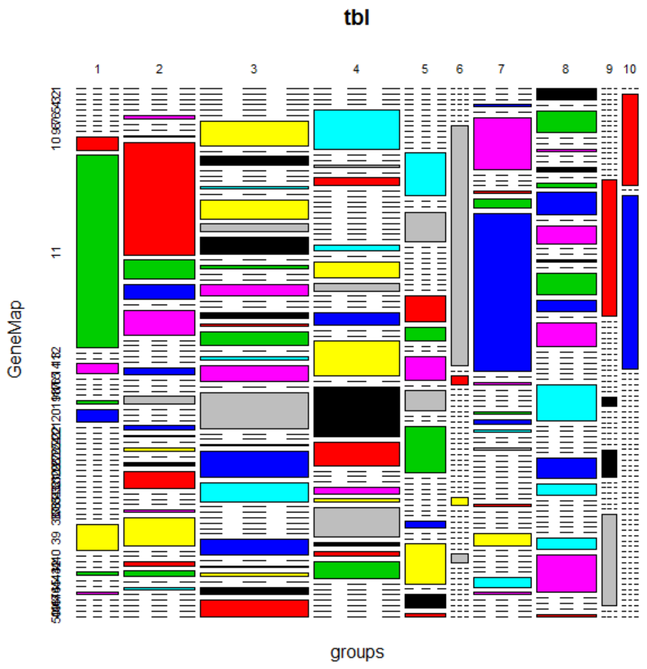
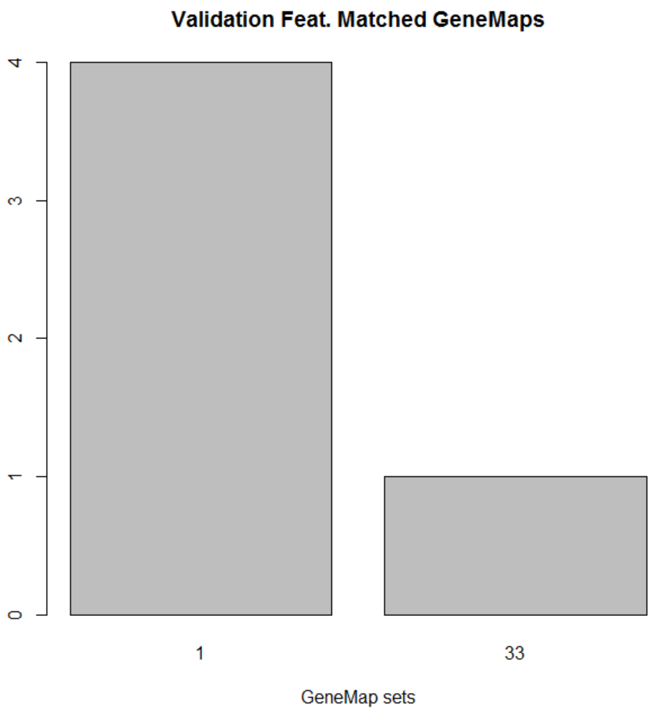
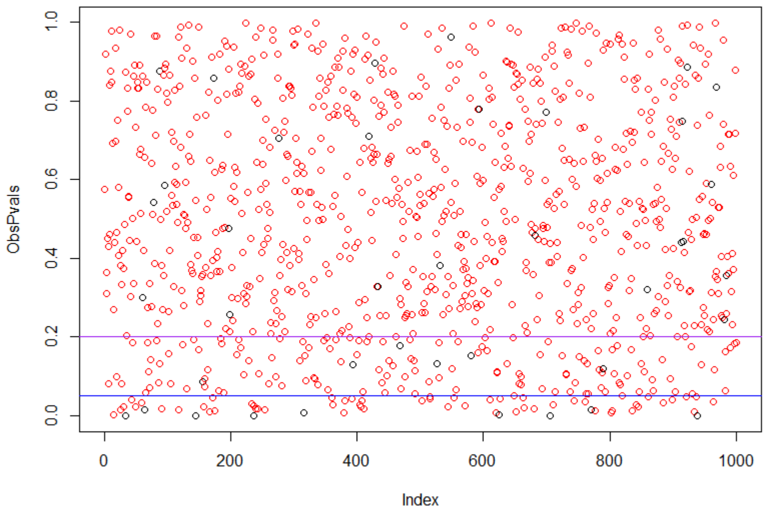


Figure 10: Mosaic Plot,k=10

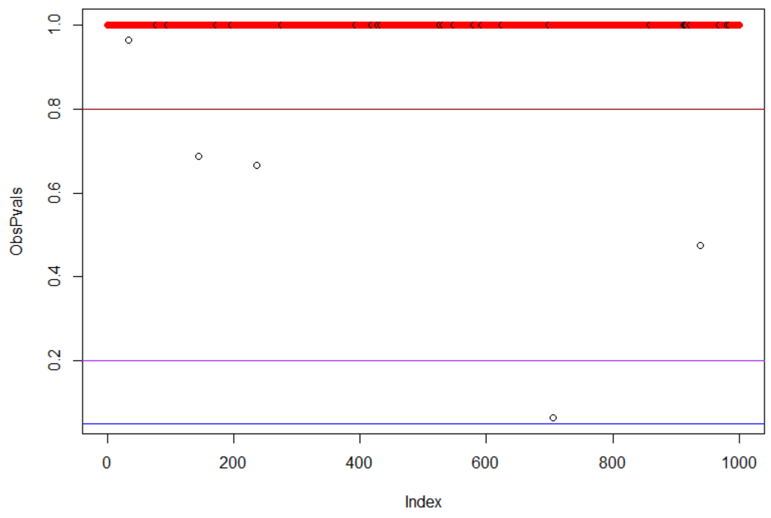


Problem 4

Using the GeneMap.RData, GSA analysis was performed and a p-value cutoff was selected to perform Fishers Exact Tests. In order ot run the runGSA command in piano, we mapped data for grouping and performed multiple test on our array data. The features for validation were identified. It was observed that there are only 5 validation features. biomaRt was used to accomplish to map multiple GO IDs. The p values for our selected genes were, 9.651155e-04, 6.877790e-04, 6.654895e-04, 6.482483e-05, & 4.750880e-04. The factor object for gene group wa set up and the p values were observed with a cutoff for significance of 0.05. As the fisher's exact test is more of a global test, it doesn't really reveal which gene group is significant. The unadjusted fisher exact p-value was observed to be 0.0005741. When the cutoff was chosen as 0.20, the unadjusted Fisher exact p-value varied to be 0.008633. It is found that the p- values change according to the value/choice of cutoff. Using Bonferroni correction, with cutoff being equal to

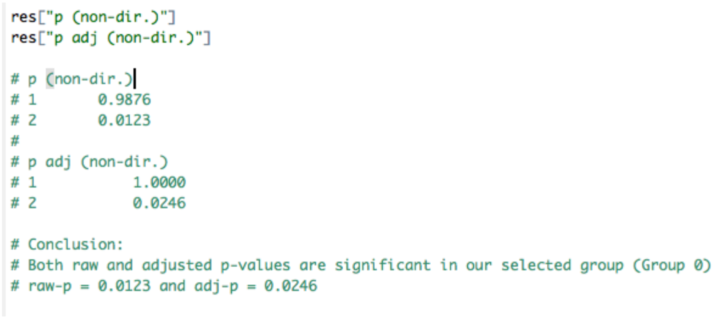


0.05, it was observed that the p value was 1 and it makes it clear that it doesnt work. When cutoff is 0.2 & 0.8, the p-values are 0.04, & 2.207e-06 respectively. Inspite of this, odds ratio was infinity so it is found that it is not correct and the results depend on the variation of the cut-off.



Problem 5

Using GeneMap.RData, GSA analysis was performed using piano A factor object for each gene group was created. For illustrative purposes, correlation were used instead of t-test values. On obtaining the p-values, we comapred t-test p-vals and correlation estimates. We plotted the correlation values & sorted them in descending order. None of the individual genes we validated was highly significant on its own. When the genes were on the other hand analyzed as a group, the set of genes were significant. We ran the GSA analysis using runGSA command. We observed that both raw and adjusted p-values are significant in the selected group(Group 0) raw-p being equal to 0.0123 & adjusted-p being equal to 0.0246



```
res["p (non-dir.)"]
res["p adj (non-dir.)"]

# p (non-dir.)
# 1       0.9876
# 2       0.0123
#
# p adj (non-dir.)
# 1           1.0000
# 2           0.0246

# Conclusion:
# Both raw and adjusted p-values are significant in our selected group (Group 0)
# raw-p = 0.0123 and adj-p = 0.0246
```

Problem 6

The factor object for gene group was set up. The t-test p-vals and correlation estimates were compared. Observed correlation values.

The correlation values were sorted in descending order.

It was made to look like a component in the GSEA plots that was observed earlier, It was ranked such that the highest correlation has rank = 1 There were 40 feats observed in group A. Analysis was done to assess whether the peak was found significant The decision was made to shuffle samples. P value was found to be 0.0006666667 None of the individual genes we validated was highly significant on its own. However, when analyzed as group, the set of genes is significant..

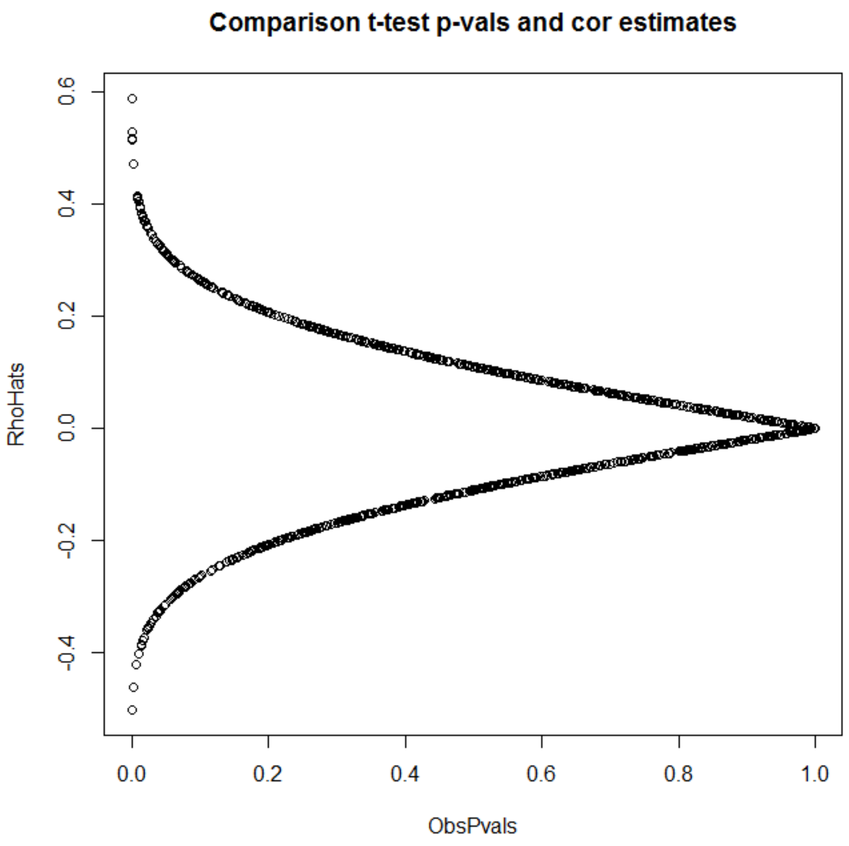


Figure 11: T-test component

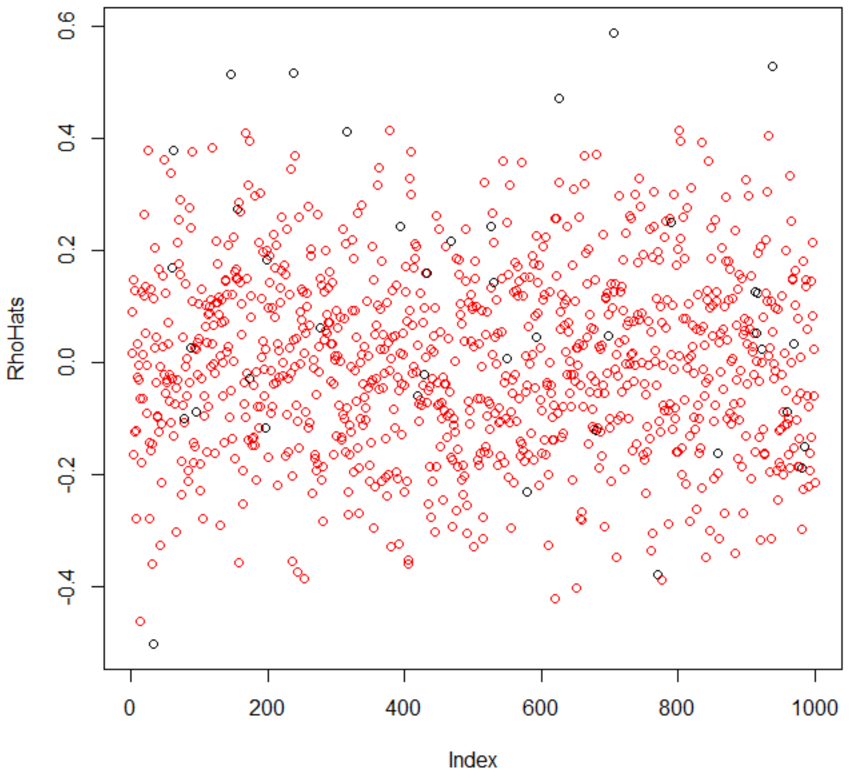


Figure 12: Unsorted correlation

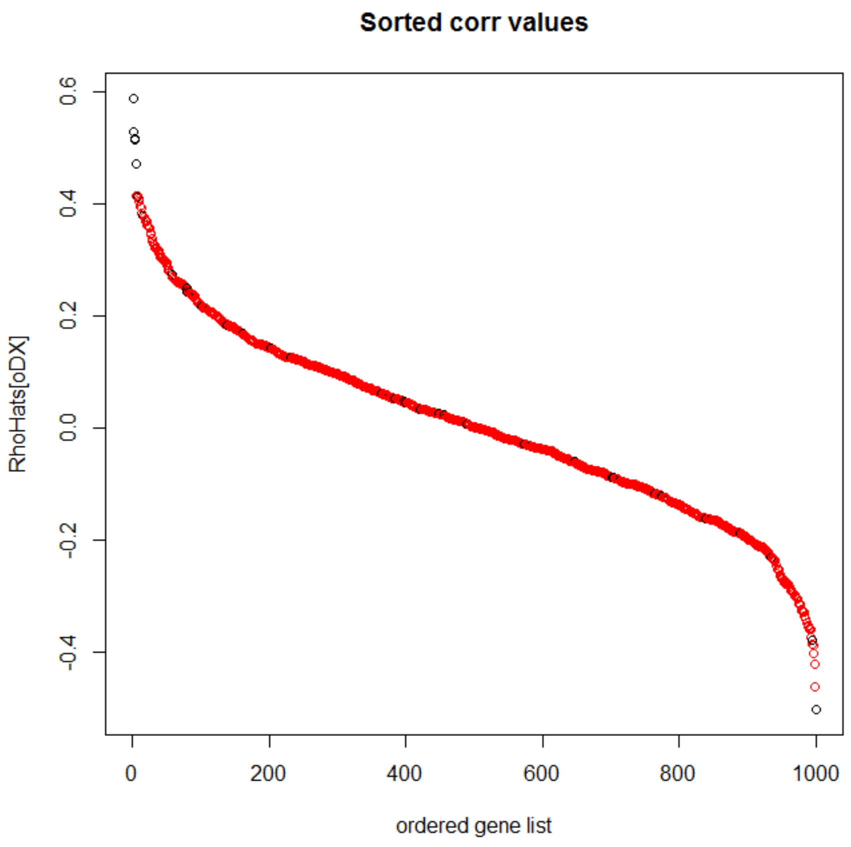


Figure 13: Sorted correlation

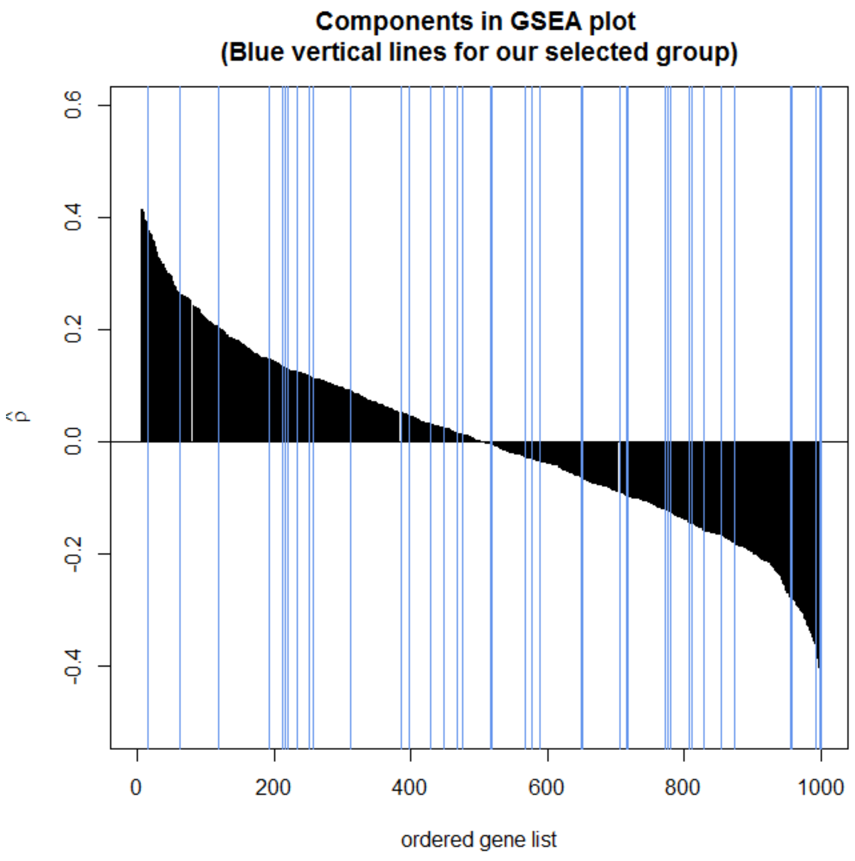


Figure 14: GSEA component

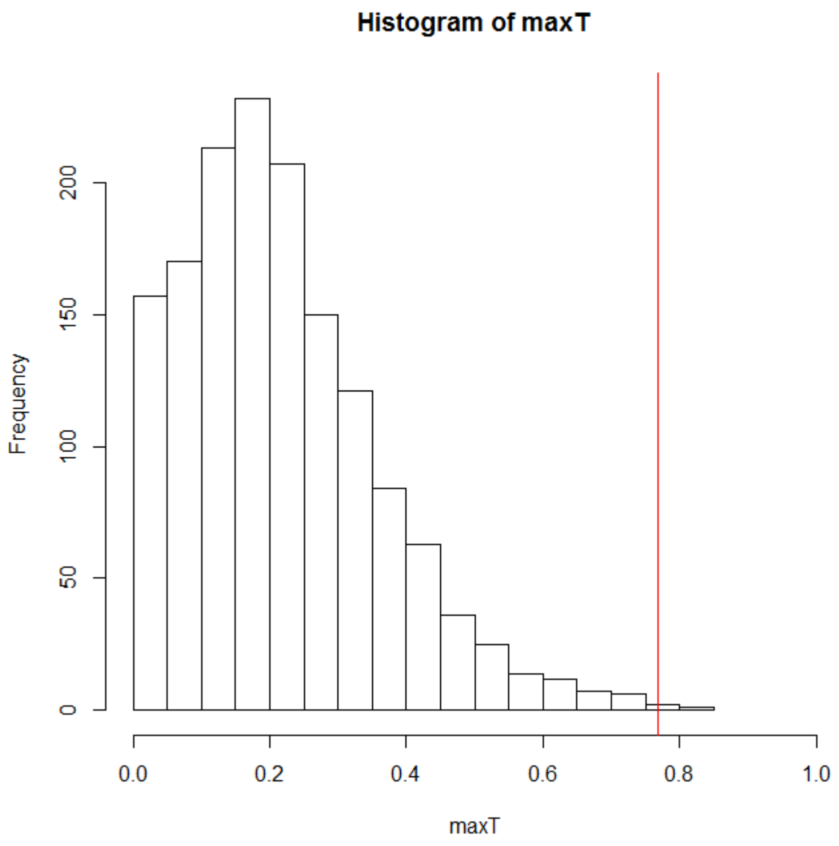
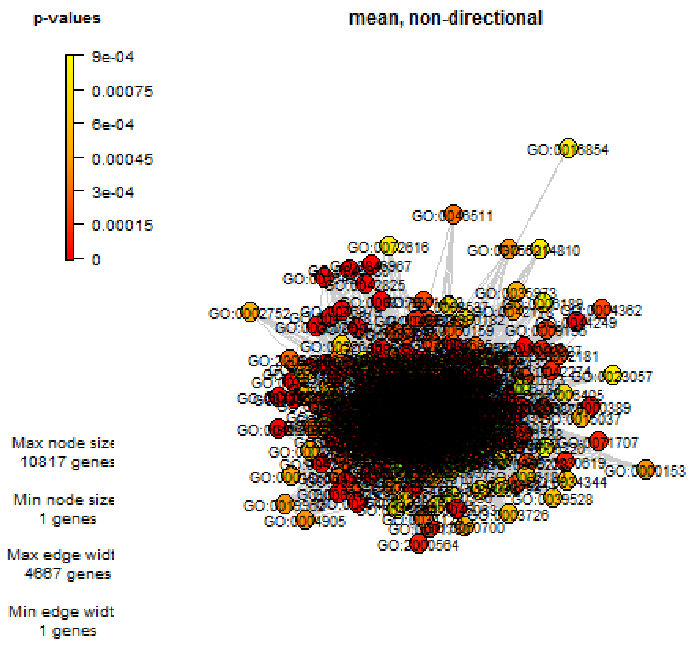


Figure 15: Histogram of MaxT

GSEA Problem

1. The gse19439 data was loaded and kruskal wallis scan was performed across all features. 2. The features of our exxpression set, observing what entrezIDs map to. We observed that EntrezIDs were not unique which means that there are many features mapping to the same EntrezIDs. We then, constructed a map from features to EntrezID. 3. The minimum p value across the set of features that map to the same EntrezID was used in the analysis. 4. In order to perform runGSA command in piano, we had to map each EntrezID to it's GO ID's. As EntrezIDs may map to mutliple GO IDs, we used biomaRt to accomplish that. The figure for the whole gene sets is very complicated as indicated. Therefore, use of filters is recommended in order to reduce the number of genes. We used the filtered goid data from group 3. As it cannot be done without Ensembl data, we made an approximate guess on the basis of group 3's analysis. It is observed that on application of filter, the number of gene sets gets reduced to a number enabling us to observe specific biological pathways.



Appendix

R code for this homework.

```
################################################
# STA 525: Statistics For Bioinfomatics
#       Homework Assignment #3
#
#-----------------------------------------------
# Group 4 members:
#         Ziheng Cheng
#         Krithika Krishnan
#         Ziqiang Chen
#################################################
#--------------------
# Problem 1
#--------------------
#============================================================================#
# Qestion 1 Array Data
#============================================================================#

setwd("")
load("Array.RData") # this provides: ArrayData,

#--------------------------------------------------------------#
# PCA
#--------------------------------------------------------------#

Adx= grep("A",colnames(ArrayData))
Bdx= grep("B",colnames(ArrayData))

ttl=c("Pop␣A␣(black)␣;␣Pop␣B␣(red)")

PCfit=prcomp(t(ArrayData),scale=F)

pcx1=t(ArrayData)%*%PCfit$rotation[,1]
pcx2=t(ArrayData)%*%PCfit$rotation[,2]
pcx3=t(ArrayData)%*%PCfit$rotation[,3]

clr=rep(1,(dim(ArrayData)[2]))
clr[Bdx]=2

s3d=scatterplot3d(pcx1,pcx3,pcx2,color=clr,pch=20,cex.symbols=1.1,
                xlab="PC1",ylab="PC3",zlab="PC2",
                main=ttl)



#####
### Power Analysis
###


dvals=seq(0,1,length=101)
pwr=rep(0,length(dvals))
for(i in 1:(length(dvals))) pwr[i]=pwr.t.test(n=10,d=dvals[i],sig.level=0.05)$power
pwr

dvals=seq(0,3.5,length=101)
pwr1=rep(0,length(dvals))
```

```
for(i in 1:(length(dvals))) pwr1[i]=pwr.t.test(n=1000,d=dvals[i],sig.level=0.05/1000)$power
pwr1

par(mfrow = c(1,2))
plot(pwr)
plot(pwr1)

#--------------------------------------------------------------#
# t-test
#--------------------------------------------------------------#
# A quick t-test scan

myT=function(i) t.test(ArrayData[i,Adx],ArrayData[i,Bdx])$p.value

p.vals=mapply(myT,1:(dim(ArrayData)[1]))
p.adj=p.adjust(p.vals,method="bonferroni")
q.adj=p.adjust(p.vals,method="fdr")

odx=order(p.vals)
p.df=data.frame(feature=rownames(ArrayData)[odx],
              p.value=p.vals[odx],
              p.bonferroni=p.adj[odx],
              q.value=q.adj[odx])

# look at best 10
p.df[1:10,]

# Above shows that feature 705, 938, 145, 237, 33, 625, 14, 620, 802, 379 are the best 10
# Or, we could look at minP adjusted p-values
#

# set up a factor object
Group=rep("A",dim(ArrayData)[2])
Group[Bdx]="B"
f=factor(Group,levels=c("A","B"))

# calculate minP adjusted p-values
resP=mt.minP(ArrayData, f, B = 10000)
# consider adding Bonferroni and BH adjustments as well
res <- mt.rawp2adjp(resP$rawp, c("Bonferroni", "BH"))
resP$Bonferroni=res$adjp[res$index,2]
resP$BH=res$adjp[res$index,3]
resP[1:10,]

# note: compare p.df adnd resP. Why is it not surprising that that they don't perfectly agree?



#============================================================================#
# Validation Data
#============================================================================#

load("Valid.RData") # provides: ValidData

#
# the object ValidData is a list of length 1000, corresponding to the 1000 features
# the list is non-empty only for features that have validation data..
#
lngths=unlist(lapply(ValidData,length))
nonZero=which(lngths!=0)
```

```
names(lngths)[nonZero]
#top feature 145, 705, 235, 63 , 754 and 954.

x=ValidData$feat705
Adx= grep("A",names(x))
Bdx= grep("B",names(x))
t.test(x[Adx],x[Bdx])

x=ValidData$feat705
Adx= grep("A",names(x))
Bdx= grep("B",names(x))
t.test(x[Adx],x[Bdx])

x=ValidData$feat235
Adx= grep("A",names(x))
Bdx= grep("B",names(x))
t.test(x[Adx],x[Bdx])

x=ValidData$feat63
Adx= grep("A",names(x))
Bdx= grep("B",names(x))
t.test(x[Adx],x[Bdx])

x=ValidData$feat754
Adx= grep("A",names(x))
Bdx= grep("B",names(x))
t.test(x[Adx],x[Bdx])

x=ValidData$feat954
Adx= grep("A",names(x))
Bdx= grep("B",names(x))
t.test(x[Adx],x[Bdx])

#-------------------
# Problem 2
#-------------------
rm(list=ls())

rm(list=ls())

# Question 2
load("HW3/RawData/Array.RData")

#--------------------------------
# Different distance metric
#--------------------------------
distance <- list()
distance[[1]] <- dist(t(ArrayData),method="euclidean")
dissimilarity <- 1 - cor(ArrayData)
distance[[2]] <- as.dist(dissimilarity)
distance[[2]] <- dist(t(ArrayData),method="manhattan")


#--------------------------------
# Different linkage method
#--------------------------------
method = c("complete","ward.D")
dist <- c("euclidean","manhattan")

#--------------------------------
```

```r
# Different K choice
#--------------------------------
# k=c(2,3,4)
# x11()
png("HW3/Figures/q2_hclust_k2.png",width=1600,height=1200,pointsize=20)
par(mfrow=c(2,2))
hc.store <- list()
for (i in 1:2) {
  for (j in 1:2) {
    hc <- hclust(distance[[i]],method=method[j])
    hc.store <- c(hc.store,hc)
    plot(hc,hang = 0.2, cex=.9, main=paste(dist[i],method[j], sep="-"))
    rect.hclust(hc, k=2, border="darkred")
  }
}
dev.off()

png("HW3/Figures/q2_hclust_k3.png",width=1600,height=1200,pointsize=20)
par(mfrow=c(2,2))
hc.store <- list()
for (i in 1:2) {
  for (j in 1:2) {
    hc <- hclust(distance[[i]],method=method[j])
    hc.store <- c(hc.store,hc)
    plot(hc,hang = 0.2, cex=.9, main=paste(dist[i],method[j], sep="-"))
    rect.hclust(hc, k=3, border="darkred")
  }
}
dev.off()

png("HW3/Figures/q2_hclust_k4.png",width=1600,height=1200,pointsize=20)
par(mfrow=c(2,2))
hc.store <- list()
for (i in 1:2) {
  for (j in 1:2) {
    hc <- hclust(distance[[i]],method=method[j])
    hc.store <- c(hc.store,hc)
    plot(hc,hang = 0.2, cex=.9, main=paste(dist[i],method[j], sep="-"))
    rect.hclust(hc, k=4, border="darkred")
  }
}
dev.off()

#-------------------------------------
# Perform Fisher exact test on k=2
#-------------------------------------

labels <- unlist(substr(colnames(ArrayData),1,1))

# for (i in 1:2) {
#   for (j in 1:2) {
#     hc <- hclust(distance[[i]],method=method[j])
#     grps <- cutree(hc, k=2)
#     groups <- rbind(groups, grps)
#   }
# }


png("HW3/Figures/q2_hclust_k2_fisher.png",width=1600,height=1200,pointsize=20)
par(mfrow=c(2,2))
```

```
for (i in 1:2) {
  for (j in 1:2) {
    hc <- hclust(distance[[i]],method=method[j])
    groups <- cutree(hc, k=2)
    pval <- fisher.test(table(groups, labels))$p.value
    plot(hc,hang = 0.2, cex=.9, xlab=paste("Distance:",dist[i]),
         main=paste(paste("Distance:",dist[i]), paste("\nMethod:",method[j]),
                 paste("\nFisher␣exact␣test␣p-values␣=", round(pval,4))))
    rect.hclust(hc, k=2, border="darkred")
  }
}
dev.off()




#--------------------
# Problem 3
#--------------------
rm(list=ls())
#################################
# Question 3
#################################
library(cluster)
library(fpc)
library(useful)
library(devtools)
library(ggbiplot)
library(wesanderson)

resetPar <- function() {
  dev.new()
  op <- par(no.readonly = TRUE)
  dev.off()
  op
}
par(resetPar())
rm(list=ls())

load("HW3/RawData/GeneMap.RData")
load("HW3/RawData/Array.RData")

#------------------------------
# Part 1
#------------------------------
# Looking at GeneMap
unique(GeneMap)
length(unique(GeneMap))
x11()
hist(GeneMap,breaks=50)


#------------------------------
# Part 2
#------------------------------

# Cluster the features
hc <- hclust(dist(ArrayData, method="euclidean"),method="ward.D")
```

```
# Dend plot first, k=4 looks good..
x11()
plot(hc,hang = -0.2, cex=.5, main="Dendrograms␣for␣clustering␣features␣(k=4)", labels=GeneMap)
rect.hclust(hc, k=4, border="darkred")

# rect.hclust(hc, k=5, border="darkred")
# rect.hclust(hc, k=6, border="darkred")
# rect.hclust(hc, k=7, border="darkred")
# rect.hclust(hc, k=8, border="darkred")

# Try k=4
groups <- cutree(hc, k=4)

# See if any of the clusters appear to be    enriched    for certain gene sets?
tbl <- table(groups,GeneMap)
print(tbl)

# We can see that gene 10 and 11 are largely distributed in cluster 1;
# and gene 12 are mostly located in cluster 4 (these are "top 3" gene sets);
# also you can see that other gene set also located in certain cluster

# We can also visulaize it
x11()
plot(tbl,col=1:50)

# we can try more cluster, for example k=10
groups <- cutree(hc, k=10)
tbl <- table(groups,GeneMap)
print(tbl)

x11()
plot(tbl,col=1:50)

# From above, we can further see that gene 10 located in cluster 1 and
gene 11 located in cluster 2, etc.


#------------------------------
# Part 3
#------------------------------
# Examine the features that you identified for validation.
load("HW3/RawData/Valid.RData")

# We have only 5 validation features
valid <- ValidData[!sapply(ValidData, is.null)]
dat.valid <- do.call("rbind", valid)

# See which gene we validated, and see where they located in the GeneMap
match(rownames(dat.valid),rownames(ArrayData))
geneset <- GeneMap[match(rownames(dat.valid),rownames(ArrayData))]
table(geneset)
x11()
barplot(table(geneset),xlab="GeneMap␣sets", main="Validation␣Feat.␣Matched␣GeneMaps")
# geneset
# 1 33
# 4  1
# We can see that our first validated feat located in the 33rd set, and all others feats
located in the 1st set in GeneMap

#-------------------
```

```
# Problem 4
#-------------------

rm(list=ls())

#########################
# Question 4
#########################

require('multtest')


# map data for grouping, perform multiple test on array data#
rm(list = ls())
load("HW3/RawData/GeneMap.RData")
load("HW3/RawData/Array.RData")

# Examine the features that you identified for validation.
load("HW3/RawData/Valid.RData")

# We have only 5 validation features
valid <- ValidData[!sapply(ValidData, is.null)]
dat.valid <- do.call("rbind", valid)
GeneGrp <- GeneMap

# save.image("HW3/RData/myDat.RData")
# Adopted Dr. Gaile's code...

###############
# Unadjusted #
###############
#-----------------------
# p-value cutoff=0.05
#-----------------------

X <- ArrayData
# Get P-values...
mightyMyT=function(i) t.test(X[i,c(1:10,21:30)],X[i,c(11:20,31:40)],var.equal=T)$p.value
ObsPvals=mapply(mightyMyT,1:1000)

# let's look at p-values for our selected genes:
# (Recall: in part 3, we selected 5 genes)
ObsPvals[match(rownames(dat.valid),rownames(ArrayData))]

# [1] 9.651155e-04 6.877790e-04 6.654895e-04 6.482483e-05 4.750880e-04

# Seems all the genes are significant, BUT..
# note: with Bonferroni correction, none of them are significant!
1000*ObsPvals[match(rownames(dat.valid),rownames(ArrayData))]

# [1] 0.96511550 0.68777898 0.66548953 0.06482483 0.47508801

# set up factor object for Gene Group..
GeneGrp[GeneGrp==1|GeneGrp==33] <- 0
GeneGrp[GeneGrp!=0] <- 1

# let's look at p-values..
plot(ObsPvals,col=as.factor(GeneGrp))
# Let's consider a cut-off for significance of 0.05
abline(h=0.05,col="blue")
```

```
# factor object for significant genes
SigFact=factor(ObsPvals<=0.05,levels=c(T,F))
 levels (SigFact)=c("Significant","Non-Sig")

# make a table..
table(SigFact,GeneGrp)

# test association of group with significance using Fisher's Exact Test
fisher.test(SigFact,GeneGrp)

# note: that was as "global" test - doesn't tell us which gene group is significant..

### (cutoff = 0.05) ###
# unadjusted Fisher exact p-value = 0.0005741
###

#-----------------------
# p-value cutoff=0.20
#-----------------------

# Let's consider a cut-off for significance of 0.20
abline(h=0.20,col="purple")
SigFact=factor(ObsPvals<=0.20,levels=c(T,F))
 levels (SigFact)=c("Significant","Non-Sig")
table(SigFact,GeneGrp)

fisher.test(SigFact,GeneGrp)


### (cutoff = 0.20) ###
# unadjusted Fisher exact p-value = 0.008633
###

# From above we can see that: Results depend upon choice of cut-off..



#######################################################
# what would the p-value be if we Bonf. corrected?
###############
# Bonferroni #
###############

# Do the exactly same thing except for Bonf. adj p-values
# Get P-values...
mightyMyT=function(i) t.test(X[i,c(1:10,21:30)],X[i,c(11:20,31:40)],var.equal=T)$p.value
ObsPvals=mapply(mightyMyT,1:1000)

ObsPvals=p.adjust(ObsPvals, method = "bonferroni")

# let's look at p-values..
plot(ObsPvals,col=as.factor(GeneGrp))
# Let's consider a cut-off for significance of 0.05
abline(h=0.05,col="blue")

# factor object for significant genes (0.05 cut-off)
SigFact=factor(ObsPvals<=0.05,levels=c(T,F))
 levels (SigFact)=c("Significant","Non-Sig")
```

```r
# make a table..
table(SigFact,GeneGrp)

# test association of group with significance using Fisher's Exact Test
fisher.test(SigFact,GeneGrp)

# p-value = 1
# Never works

#----------------------------------
# if we set cut-off = 0.2 and 0.8..

# Let's consider a cut-off for significance of 0.05
abline(h=0.2,col="purple")
abline(h=0.8,col="darkred")
# factor object for significant genes (0.2 cut-off)
SigFact=factor(ObsPvals<=0.2,levels=c(T,F))
levels(SigFact)=c("Significant","Non-Sig")

# make a table..
table(SigFact,GeneGrp)

# test association of group with significance using Fisher's Exact Test
fisher.test(SigFact,GeneGrp)

# cut-off = 0.2: p-value = 0.04
# But it is not correct, since odds ratio is infinity

# factor object for significant genes (0.8 cut-off)
SigFact=factor(ObsPvals<=0.8,levels=c(T,F))
levels(SigFact)=c("Significant","Non-Sig")

# make a table..
table(SigFact,GeneGrp)

# test association of group with significance using Fisher's Exact Test
fisher.test(SigFact,GeneGrp)


# cut-off = 0.8: p-value = 2.207e-06
# But it is not correct, since odds ratio is infinity

# Overall, Results depend upon choice of cut-off..

#--------------------
# Problem 5
#--------------------

rm(list = ls())
#########################
# Question 5
#########################
rm(list = ls())
library(piano)
load("HW3/RData/myDat.RData")

# set up factor object for Gene Group..
GeneGrp[GeneGrp==1|GeneGrp==33] <- 0
GeneGrp[GeneGrp!=0] <- 1
```

```
# Construct our own GeneMap
myMap = cbind(row.names(ArrayData), paste("GeneSet", GeneGrp))
# Now that we have a mapping, we can load it into the correct format using loadGSC
myGSC = loadGSC(myMap)

SmplGrp=c(rep(0,10),rep(1,10),rep(0,10),rep(1,10))

# Get P-values...
X <- ArrayData
mightyMyT=function(i) t.test(X[i,c(1:10,21:30)],X[i,c(11:20,31:40)],var.equal=T)$p.value
ObsPvals=mapply(mightyMyT,1:1000)

names(ObsPvals)=rownames(ArrayData)

# save(ObsPvals,myGSC,SmplGrp,myMap,GeneGrp,file="HW3/RData/myGSA.RData")

# We have all our input data and can move on with the runGSA function:
gsa = runGSA(ObsPvals, gsc=myGSC, adjMethod="bonferroni")

print(gsa)

res=GSAsummaryTable(gsa)

# See what we have got (Note: GeneSet 0 is our selected )
res

res["p␣(non-dir.)"]
res["p␣adj␣(non-dir.)"]

# p (non-dir.)
# 1       0.9876
# 2       0.0123
#
# p adj (non-dir.)
# 1           1.0000
# 2           0.0246

# Conclusion:
# Both raw and adjusted p-values are significant in our selected group (Group 0)
# raw-p = 0.0123 and adj-p = 0.0246
#--------------------
# Problem 6
#--------------------
#########################
# Question 6
#########################
rm(list = ls())
load("HW3/RawData/GeneMap.RData")
load("HW3/RawData/Array.RData")

X <- ArrayData

GeneGrp <- GeneMap
# set up factor object for Gene Group..
GeneGrp[GeneGrp==1|GeneGrp==33] <- 0
GeneGrp[GeneGrp!=0] <- 1

SmplGrp=c(rep(0,10),rep(1,10),rep(0,10),rep(1,10))

# We could/(should?) use t-test values instead of correlation but
```

```r
# for illustrative purposes, let's use correlation

MyCor=function(i) cor(SmplGrp,X[i,])
RhoHats=mapply(MyCor,1:1000)

# Get P-values...
mightyMyT=function(i) t.test(X[i,c(1:10,21:30)],X[i,c(11:20,31:40)],var.equal=T)$p.value
ObsPvals=mapply(mightyMyT,1:1000)

# aside, let's compare t-test p-vals and cor estimates
x11()
plot(ObsPvals,RhoHats, main="Comparison␣t-test␣p-vals␣and␣cor␣estimates")

# let's look at corr values
x11()
plot(RhoHats,col=as.factor(GeneGrp))

# let's sort the values in descending order
oDX=order(RhoHats,decreasing=T)
x11()
plot(1:1000,RhoHats[oDX],col=as.factor(GeneGrp[oDX]),xlab="ordered␣gene␣list",
      main="Sorted␣corr␣values")

# let's make this look like a component in the GSEA plots that we have seen:
x11()
plot(1:1000,RhoHats[oDX],col=as.factor(GeneGrp[oDX]),xlab="ordered␣gene␣list",
      ylab=expression(hat(rho)),pch="␣",
      main="Components␣in␣GSEA␣plot␣\n(Blue␣vertical␣lines␣for␣our␣selected␣group)")
abline(h=0)
for(i in 1:1000) lines(c(i,i),c(0,RhoHats[oDX[i]]),col=GeneGrp[oDX[i]])

ranks=1001-rank(RhoHats) # rank such that highest corr has rank 1

# See how many feats in group A
length(which(GeneGrp==0))

# 40

# put vertical lines for group A
abline(v=ranks[1:40], col="cornflowerblue")

#
# Calculate running sum statistic
#    use method II and p=1
#

RunSumA=rep(0,1001) # note: 51 not 50 - 1st place i init value
N=1001
Ns=40
Nr=sum(abs(RhoHats[1:40])^1) # note: I put ^1 just to emphasize that it is ^p with p=1


for(i in 1:1000){

  if(GeneGrp[oDX[i]]!=0){ # is the ith ordered gene not in group A?
    RunSumA[i+1]=RunSumA[i]-1/(N-Ns)
  }

  if(GeneGrp[oDX[i]]==0){ # is the ith ordered gene in group A?
    RunSumA[i+1]=RunSumA[i]+(abs(RhoHats[oDX[i]])^1)/Nr
  }

}

plot(1:1000,RunSumA[2:1001],type="l")
abline(v=ranks[1:40],col="gray49")

Tobs=max(RunSumA)

# T_observed = 0.768018

#
# How can we assess whether the peak is significant?
#

rho.original=RhoHats
SmplGrp.orig=c(rep(0,10),rep(1,10),rep(0,10),rep(1,10))

nperm=1500
maxT=rep(NA,1500)

for(h in 1:1500){

  cat(h,"...")

  # Do we shuffle genes or shuffle samples?
  # Let's discuss this...

  # Shuffle Samples:
  SmplGrp=sample(SmplGrp) # shuffle samples
  RhoHats=mapply(MyCor,1:1000)

  oDX=order(RhoHats,decreasing=T)

  Nr=sum(abs(RhoHats[1:40])^1) # note: I put ^1 just to emphasize that it is ^p with p=1


  for(i in 1:1000){

    if(GeneGrp[oDX[i]]!=0){ # is the ith ordered gene not in group A?
      RunSumA[i+1]=RunSumA[i]-1/(N-Ns)
    }

    if(GeneGrp[oDX[i]]==0){ # is the ith ordered gene in group A?
      RunSumA[i+1]=RunSumA[i]+(abs(RhoHats[oDX[i]])^1)/Nr
    }

  }

  if(h<10){ # plot the first 10 permutations
    Sys.sleep(0.1)
    plot(1:1000,RunSumA[2:1001],type="l",main=paste("Permuted␣Dataset",h))
    abline(v=ranks[1:40],col="gray49")
  }

  maxT[h]=max(RunSumA)

}

x11()
```

```
hist(maxT,xlim=c(min(maxT),1))
abline(v=Tobs,col=2)

pval=mean(maxT>=Tobs)

pval

# > pval
# [1] 0.0006666667
#
# Remember, none of the individual genes we validated was highly significant on its own.
# However, when analyzed as group, the set of genes is significant..

##########################
# GSEA problem
##########################
rm(list = ls())
# Load data

#----------------------------------------------
# 1. Load the gse19439 data and perform the Kruskal Wallis scan mentioned in HW#1
#----------------------------------------------

LoadScanFlag=F # flip this flag to T to load the data and perform the KW scan
if(LoadScanFlag){
  require(GEOquery)
  gse19439=getGEO("GSE19439",GSEMatrix=T)
  gse19439=gse19439[[1]]
  tmp=as.character(pData(phenoData(gse19439))[,1])
  J=length(tmp) # J=number of samples
  TBgroup=rep("",J)
  for(j in 1:J) TBgroup[j]=substring(tmp[j],1,3)
  # make a factor for TBgroup
  FTB=factor(TBgroup,levels=c("CON","LTB","PTB"))
  # get our expression set
  X=exprs(gse19439)
  #
  # Let's do a simple Kruskal-Wallis Scan across all features
  #
  # do a quick kruskal-wallis scan
  myKrusk=function(i){
    cat(i,"...",fill=F)
    kruskal.test(x=X[i,],g=FTB)$p.value
  }
  myPvals=mapply(myKrusk,1:(dim(X)[1])) ;save(file="myPvals.RData",myPvals)
  save(gse19439,myPvals,FTB,TBgroup,X,file="HW3/RawData/HW3loadscan.RData")
}
if(!LoadScanFlag) load("HW3/RawData/HW3loadscan.RData")

#------------------------------------------------------------
# 2. Now, we have to look at the "features" of our ExpressionSet. What EntrezIDs do they map to?
#------------------------------------------------------------
require(affy)


myfeat=featureData(gse19439)
# what annotation information do we have?
varLabels(myfeat)
# o.k., but what do those varLabels means?
myvarmdat=varMetadata(myfeat)
```

```r
# Now we know what info we have, how can we extract it and have a look?
mypdat=pData(myfeat)
Entrezs=mypdat[,11]

length(Entrezs)-length(unique(Entrezs))

# From above we can see that: EntrezIDs are not unique, which means there are many features
 mapping to the same EntrezIDs.
# Next, we will construct a map from featrues to EntrezID.
#-----------------------------------------------------------------
# 3. Use the minimum p-value across the set of features that map to the same EntrezID.
#------------------------------------------------------------------

UnqFlag=F # change this flag to T to run this code the first time..
 if(UnqFlag){
  # This is the EntrezIDs:
  unqEntrez=unique(Entrezs)
  mapEntrz=rep(NA,length(unqEntrez))
   for(i in 1:length(unqEntrez)){
    edx=which(Entrezs==unqEntrez[i])
    mapEntrz[i]=edx[order(myPvals[edx])[1]]
  }
  mapEntrz=mapEntrz[unqEntrez!=""]
  names(mapEntrz)=unqEntrez[unqEntrez!=""]
  myEntrezs=names(mapEntrz)
  myX=X[mapEntrz,]
  myP=myPvals[mapEntrz]
  save(mapEntrz,myEntrezs,myX,myP,file="HW3/RawData/UnqEntrezs.RData")
}
 if(!UnqFlag) load("HW3/RawData/UnqEntrezs.RData")

#-------------------------------------------
# 4. Perform runGSA in piano
#---------------------------------------------.
# In order to do that, we will need to map each EntrezID to its GO ID(s).
# Note that EntrezIDs may map to multiple GO IDs. We will use biomaRt to accomplish this..
 require("biomaRt")
 require("piano")

# Note: The BioMart community portal is temporarily unavailable. So here we use the
# RData from group 3
load("HW3/RawData/goids.RData")
head(goids)

myGsc = loadGSC(goids)
runGSAflag=F # trip to True to run the analysis
 if(runGSAflag){
  names(myP)=myEntrezs
  myP[is.na(myP)]=1.0 # a lazy work around to account for a missing P-value
  gsaRes <- runGSA(myP, gsc=myGsc)
  save(gsaRes,file="HW3/RawData/gsaRes.RData")
  GSAsummaryTable(gsaRes, save=TRUE, file="HW3/RawData/gsaResTab.xls")
  nw <- networkPlot(gsaRes,class="non")
}

# The figure above for the whole gene sets is very complicated.
## Thus we should use filters to reduce the number of genes.
# Use the filted goid data from group 3
load("filtered_GOID.RData")
```

```
# Unfortunately, it cannot be done without the ensembl data.
# By guess and looking at the group 3's result,
# after applying filter, the number of gene sets reduced so we can see some
    #specific biological pathways.
```