Homework Assignment # 2

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

March 31, 2016

Problem 1

(a) Details about our groups's dataset, GSE19442

Berry South Africa (array)
gse19442=getGEO("GSE19442",GSEMatrix=T)
show(gse19442)
pdat19442=pData(phenoData(gse19442[[1]]))[,c(1,11:16)]

It corresponds to a dataset called Berry South Africa titled, "Blood Trancriptional Profiles of TB in South Africa." It records the expression profiling by array. It aims to compare transcriptional response to Tuberculosis in regions of different incidence or prevalence. It is coded with the different spectra of TB disease which is as follows,

>ATB - Active TB,
>LTB - Latent TB,
>PTB - Pulmonary TB.
Controls for PTB are from LTB. The platform used was GPL6947 Illumina Human HT-12 V30 expression beadchip

(b) We cleaned up the data by removing variable names in the data and by providing headers.
PCA was first performed for the entire dataset.

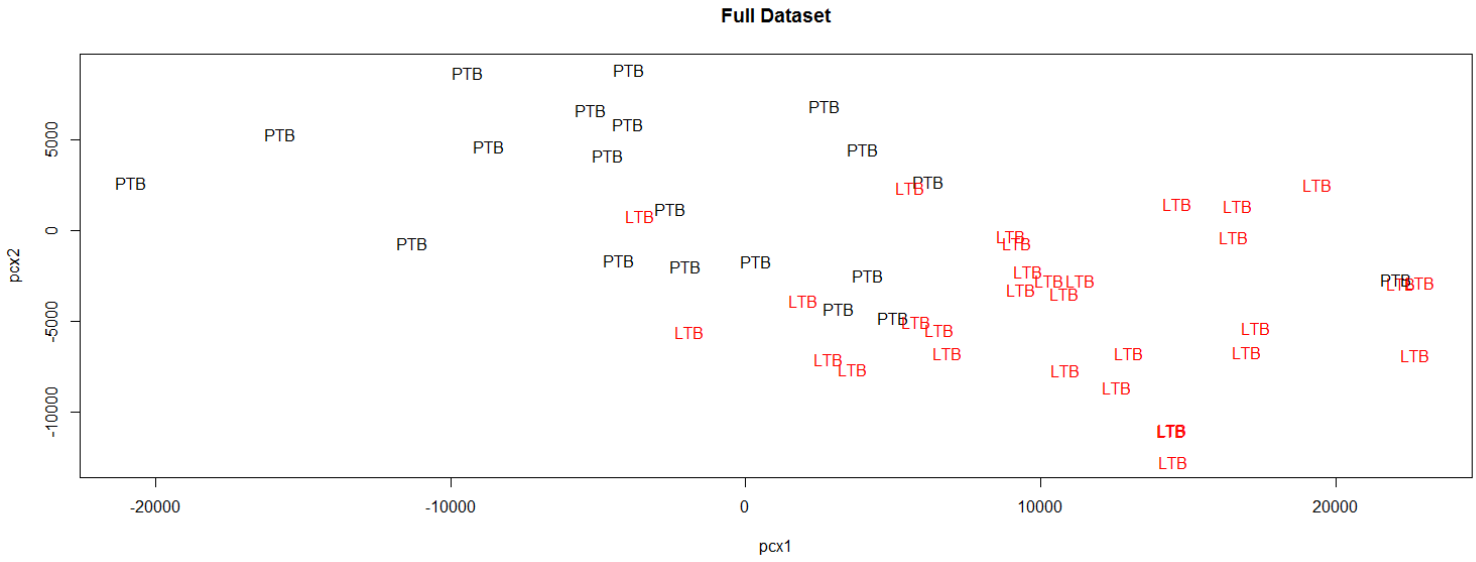


Figure 1: PCA of full Dataset

(c)PCA was performed to examine each of the factor levels assigned.
(d)Feature selection step was performed using diptest library package Histogram for the process,
Then, PCA was done for each factor level, It started from a feature selection in the range of the following across all the 6 variables,

!htbp]

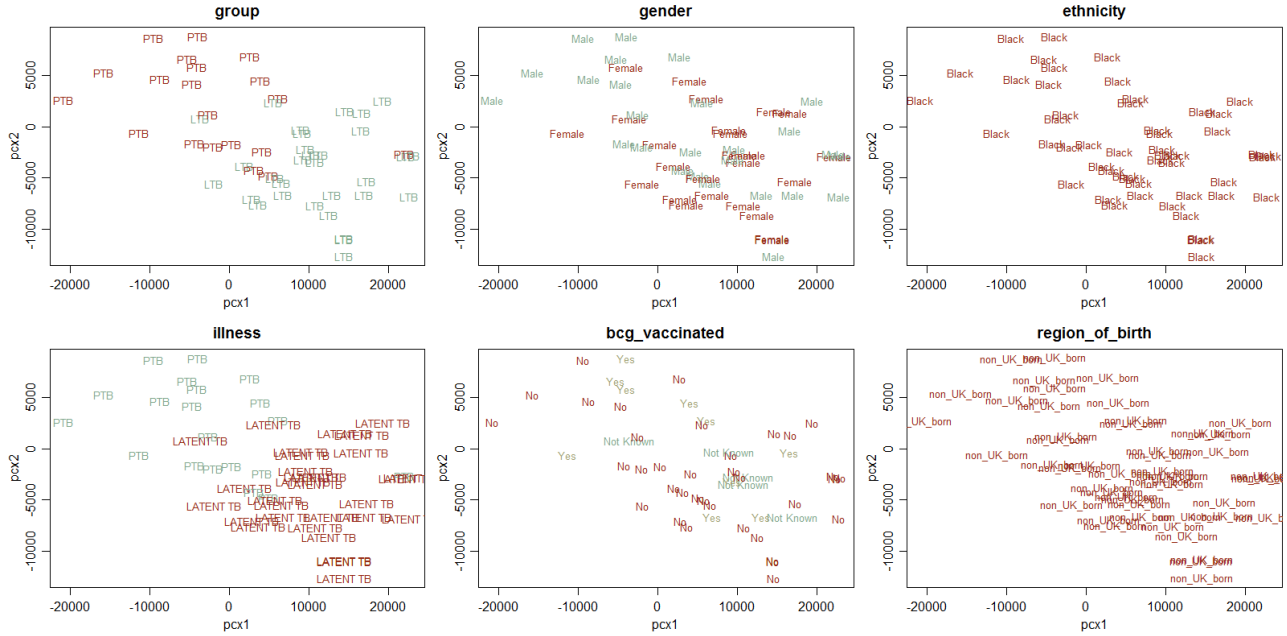


Figure 2: PCA plotted for PCA1 and PCA2 for all factor levels

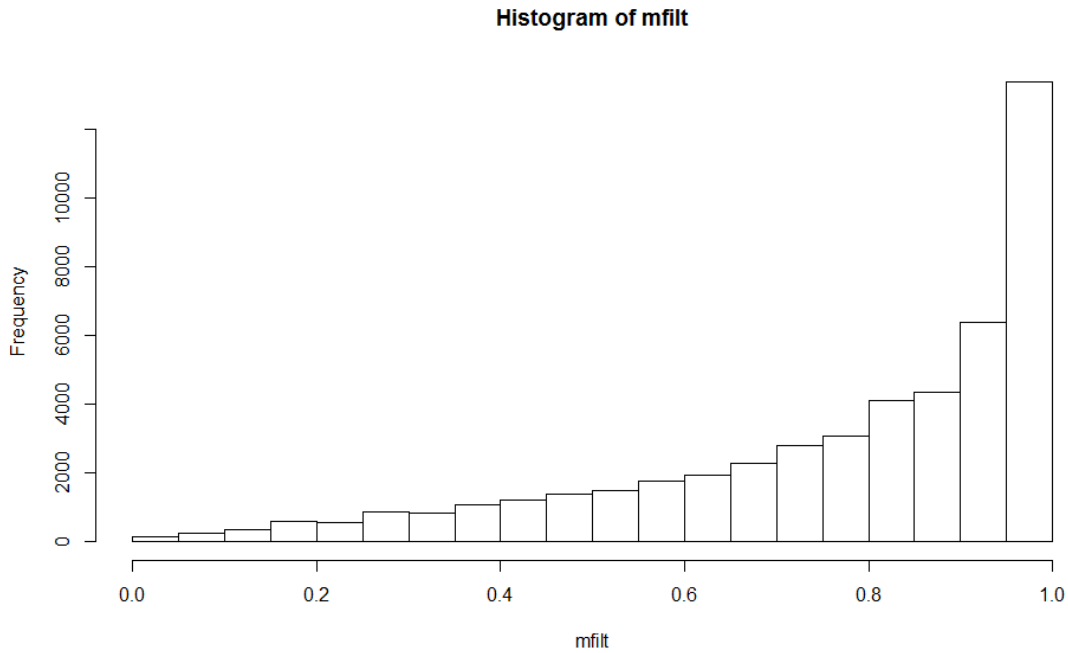


Figure 3: Histogram : feature selection

1. 5000
2. 2500
3. 1500
4. 1000
5. 500
6. 100

It was observed that the removal of features and selecting specfic features leads to loss in variability due to which there is a risk involved in losing signal.

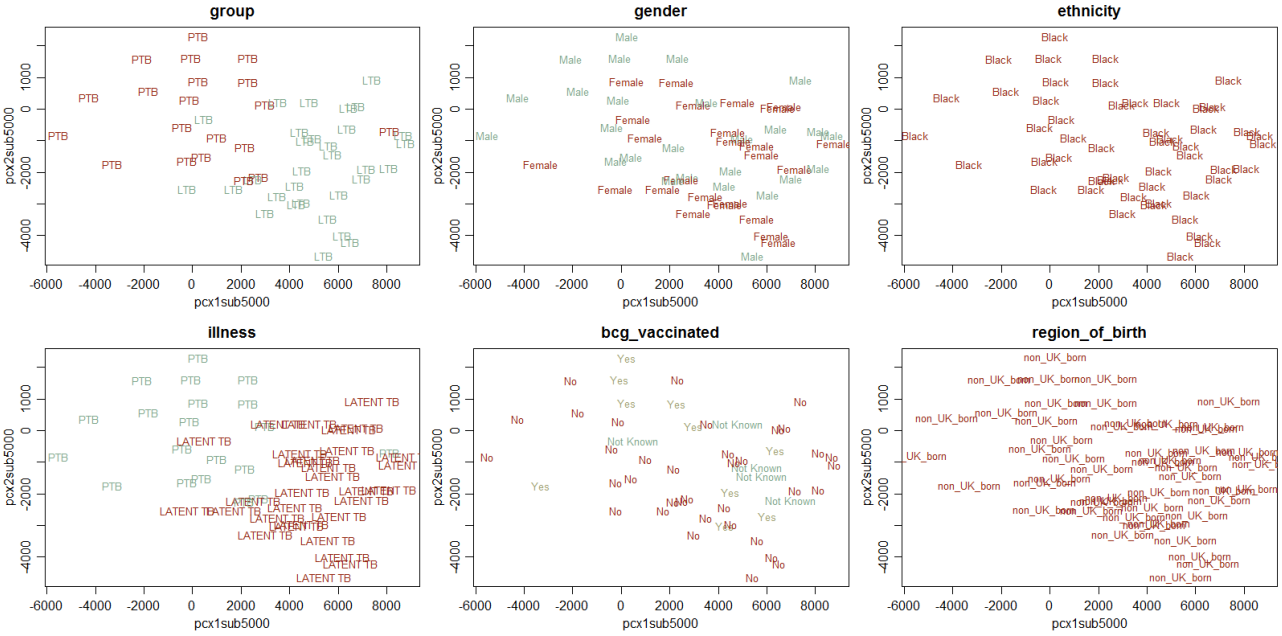


Figure 4: PCA plotted for PCA1 and PCA2 for 5000 features

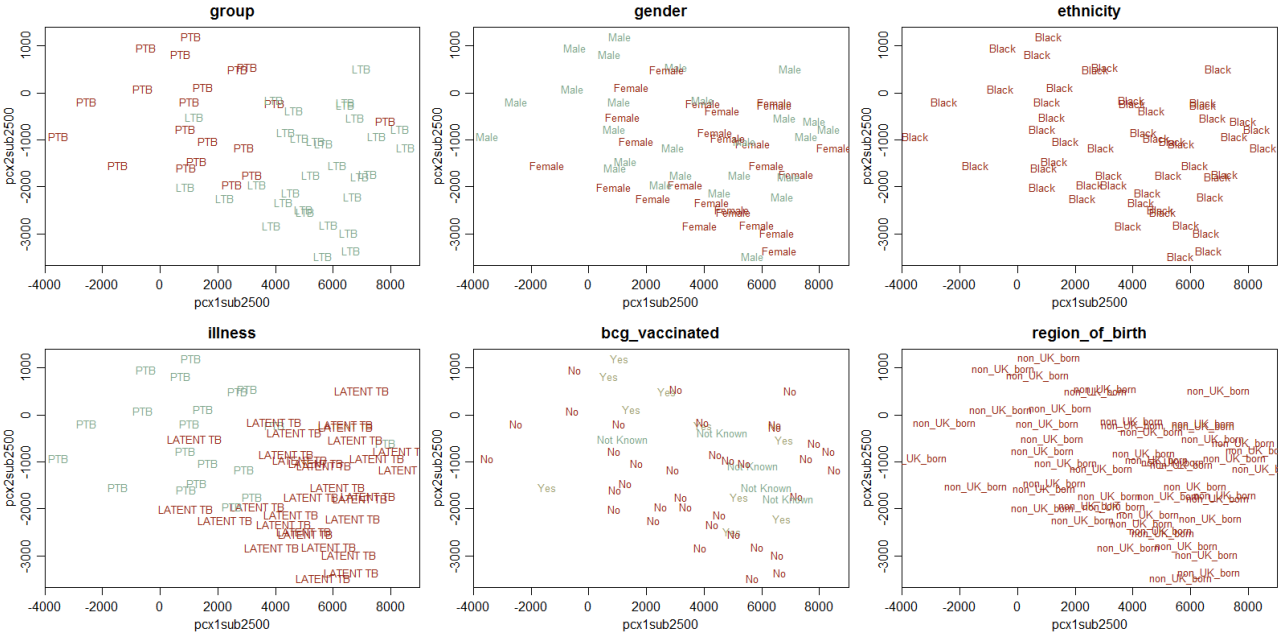


Figure 5: PCA plotted for PCA1 and PCA2 for 2500 features

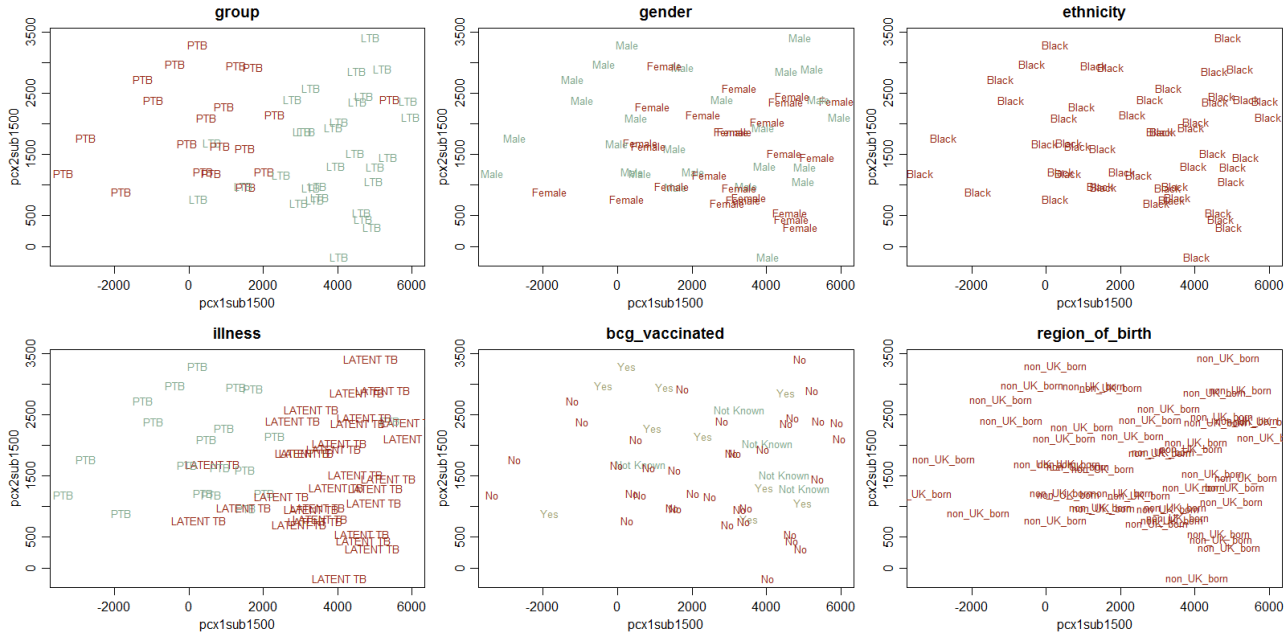


Figure 6: PCA plotted for PCA1 and PCA2 for 1500 features

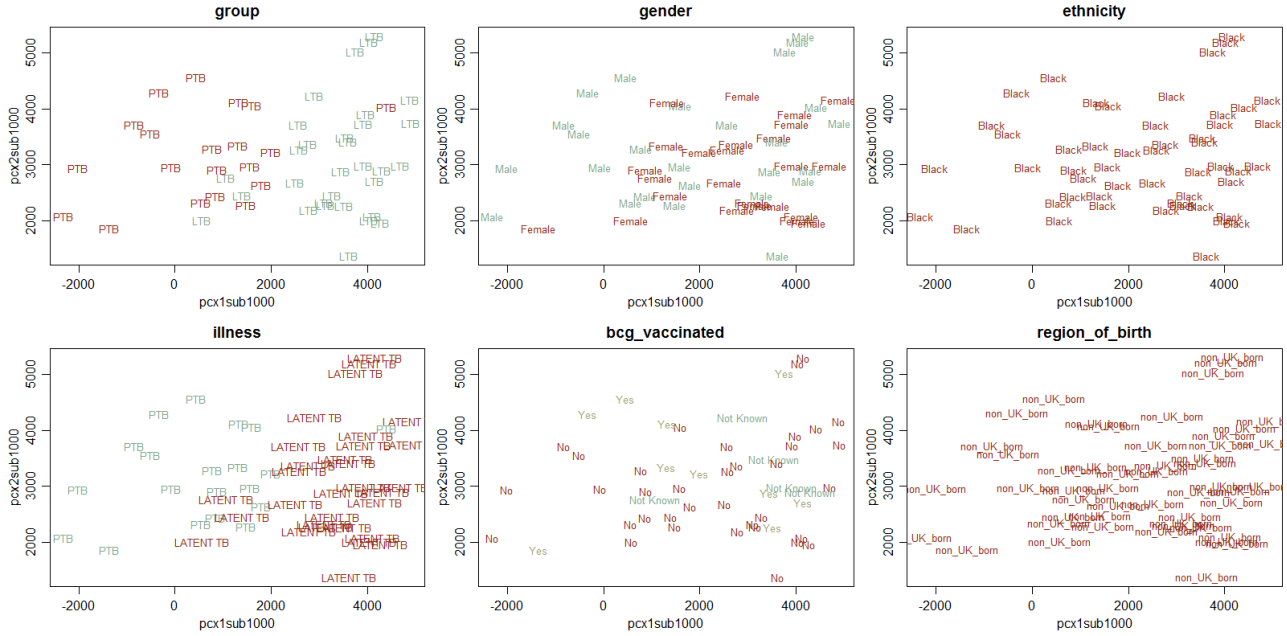


Figure 7: PCA plotted for PCA1 and PCA2 for 1000 features

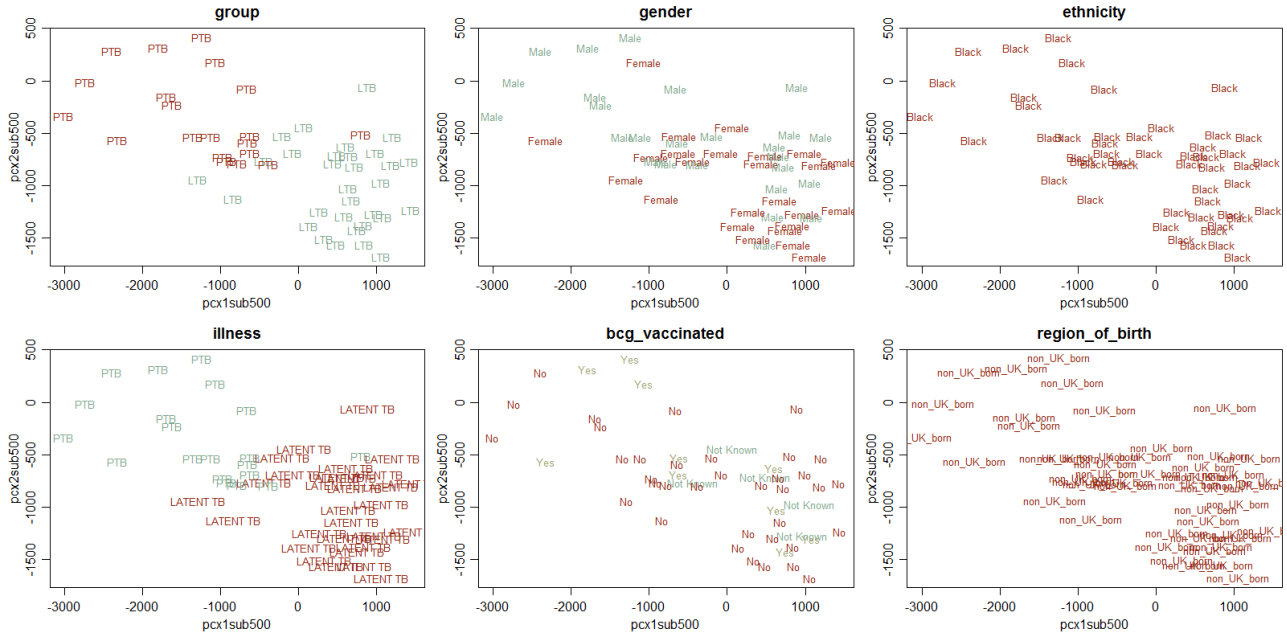


Figure 8: PCA plotted for PCA1 and PCA2 for 500 features

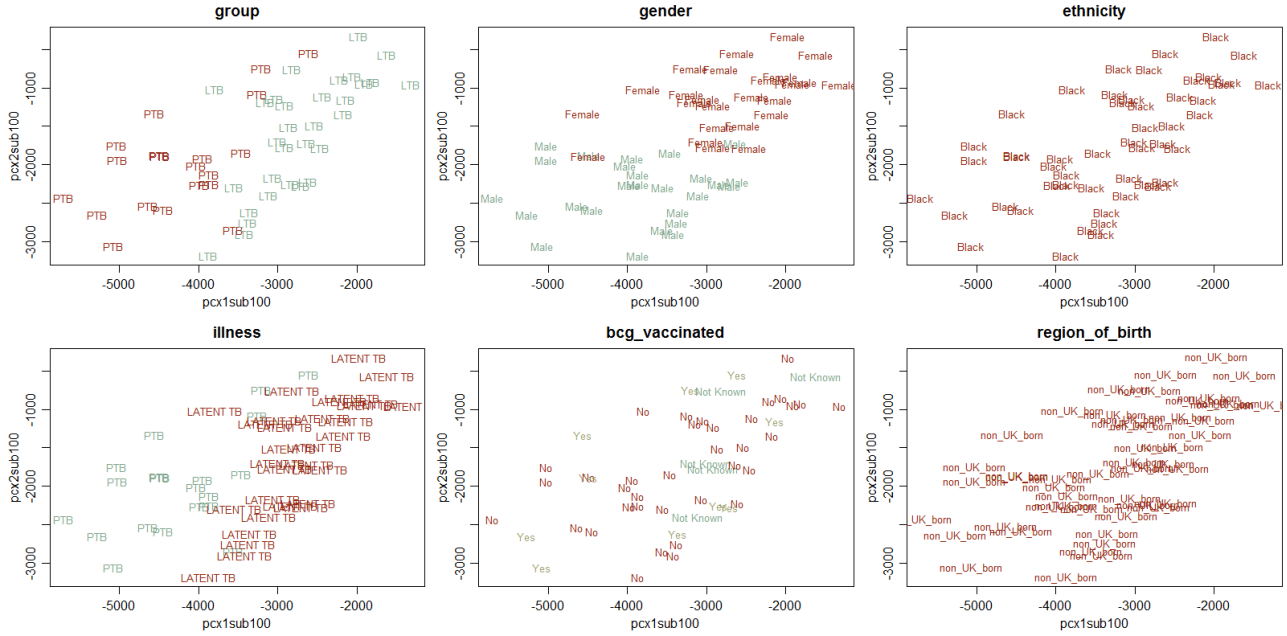


Figure 9: PCA plotted for PCA1 and PCA2 for 100 features

Problem 2

(a) The group's assigned Berry, 2010 dataset was loaded (b) Using hclust function, clustering was performed of the given samples in order to test for any association among the clinical variables that are chosen as subjects. The following were used as different distance metrics & clustering methods

1. distance metrics : correlation-based, euclidean & manhattan
2. clustering methods : single, complete, & average

It can observed that the clustering perfomed using distance metric, manhattan & clustering method, complete linkage gave a good cluster tree with a better divide into 2 groups.

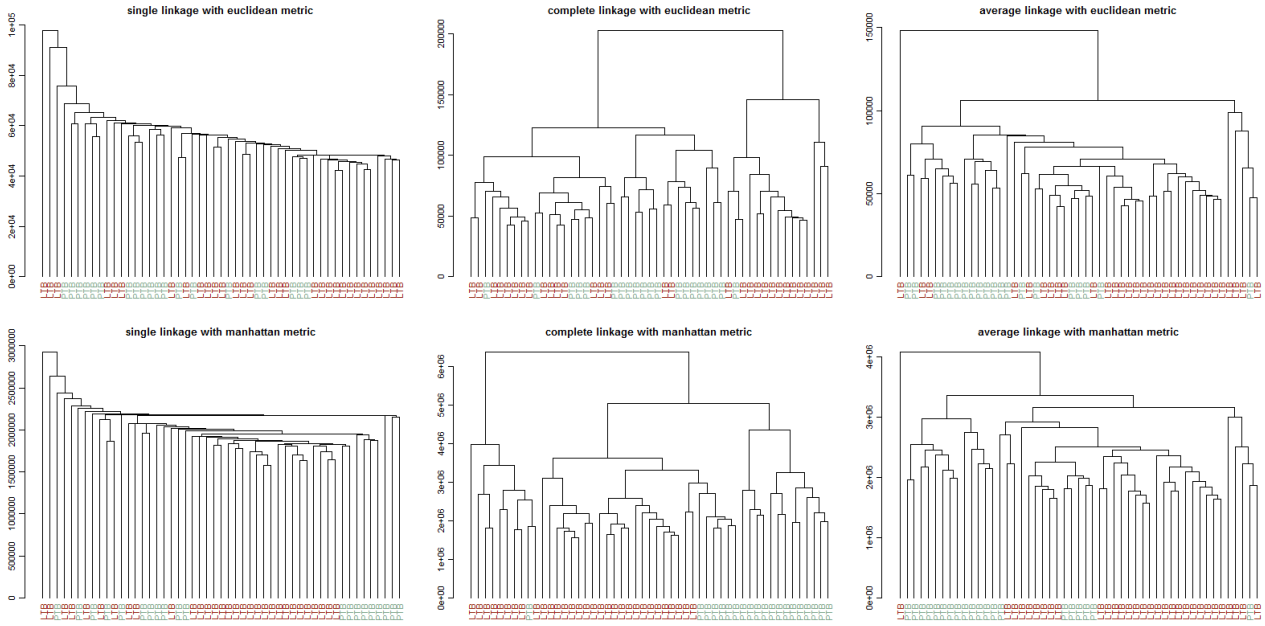


Figure 10: clustering of the samples using three different distance metrics & clustering methods for the full data

(c) The above analysis was conducted using feature selected subsets from problem 1 of the Berry, 2010. The clustering done for the selected subsets of the dataset proved to be more scattered and varied. The subset having 5000 features proved to be the better of them.

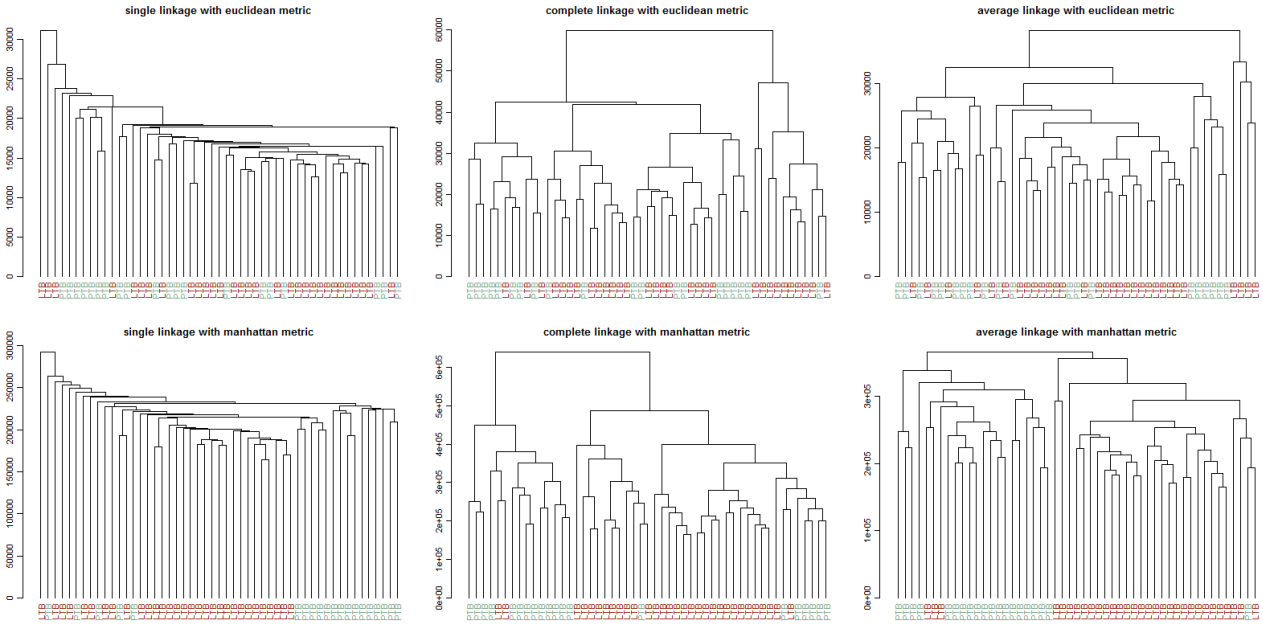


Figure 11: clustering of 5000 features

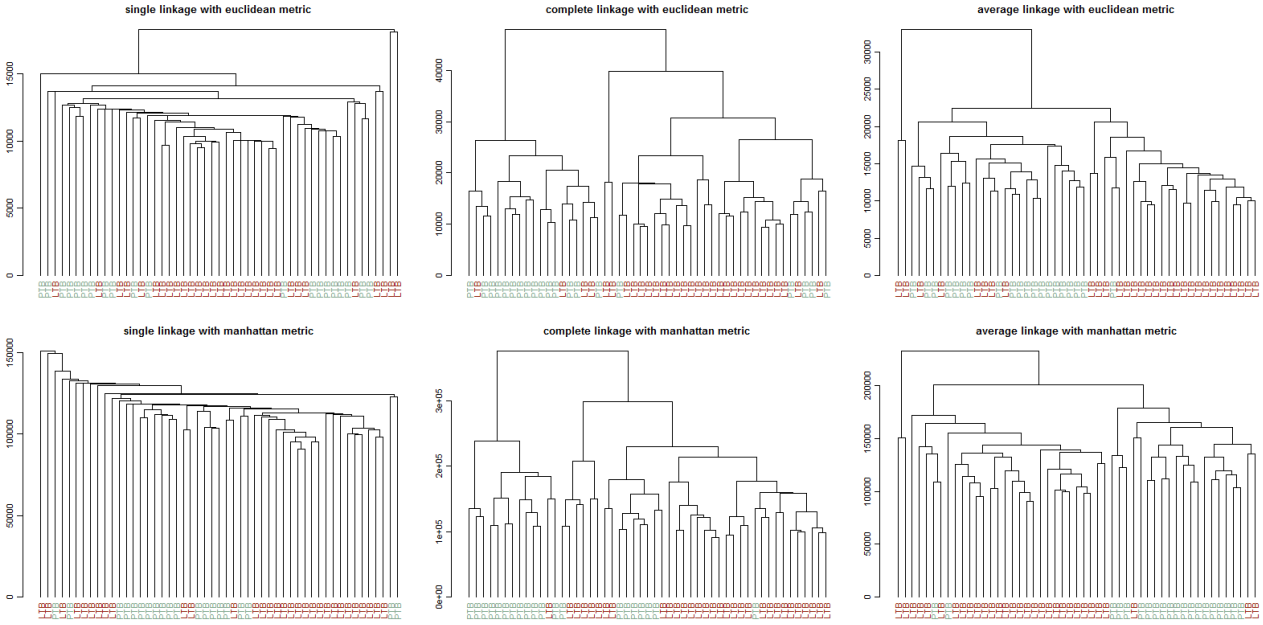


Figure 12: clustering of 2500 features

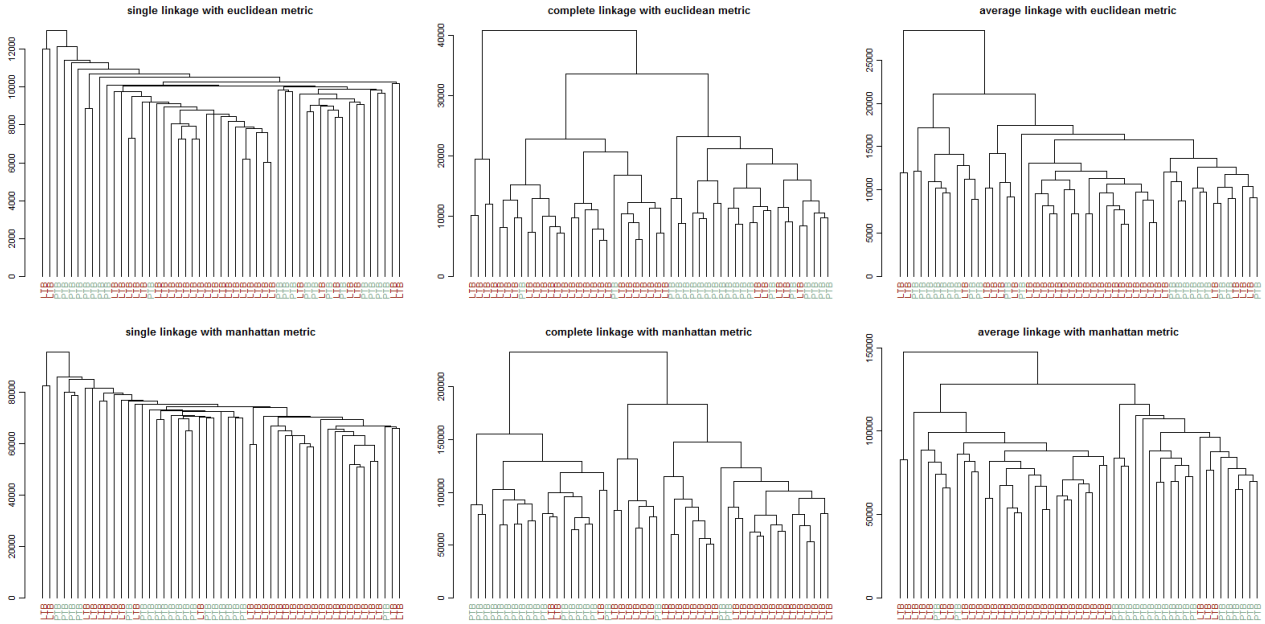


Figure 13: clustering of 1500 features

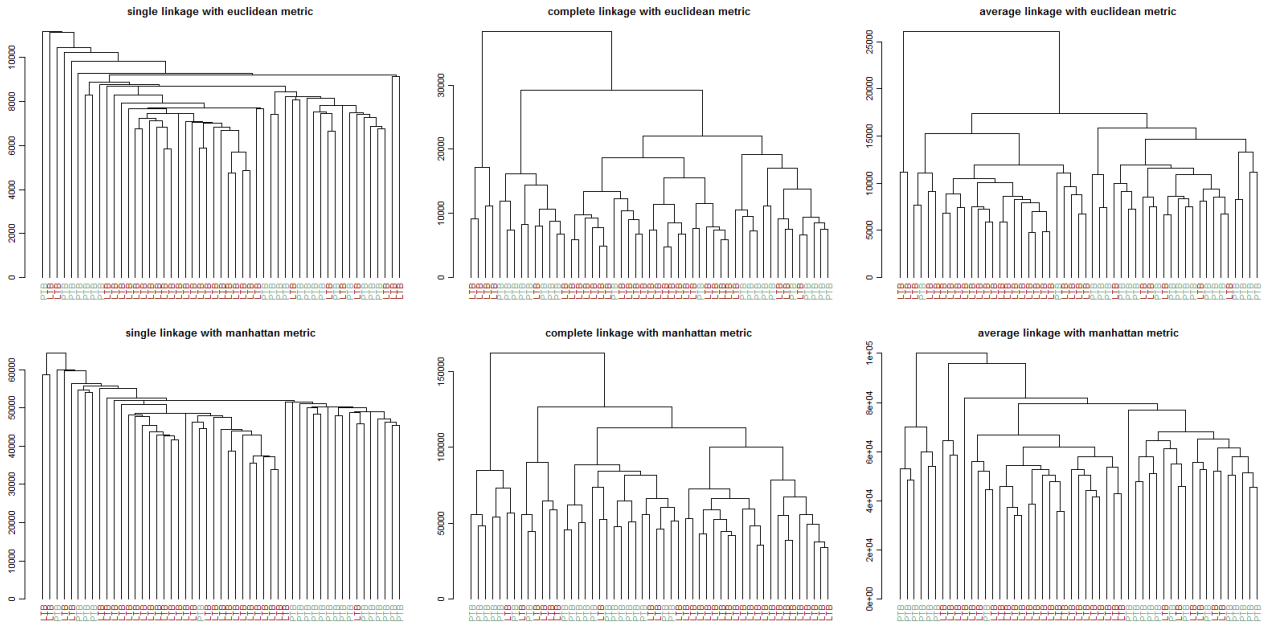


Figure 14: clustering of 1000 features

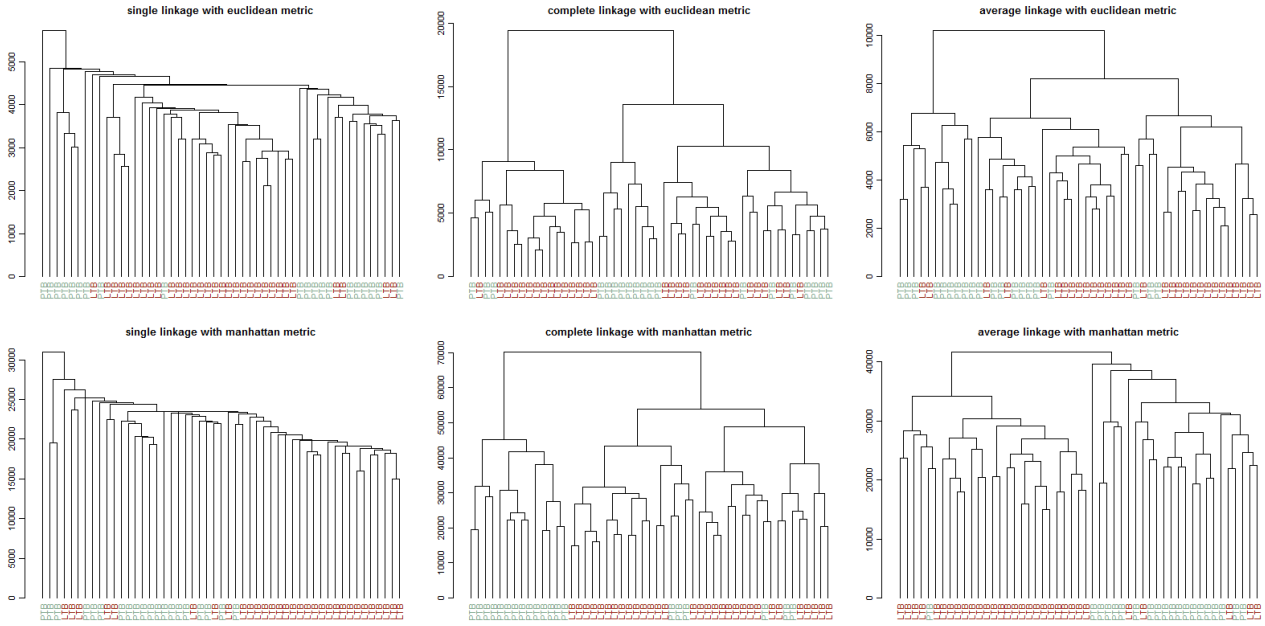


Figure 15: clustering of 500 features

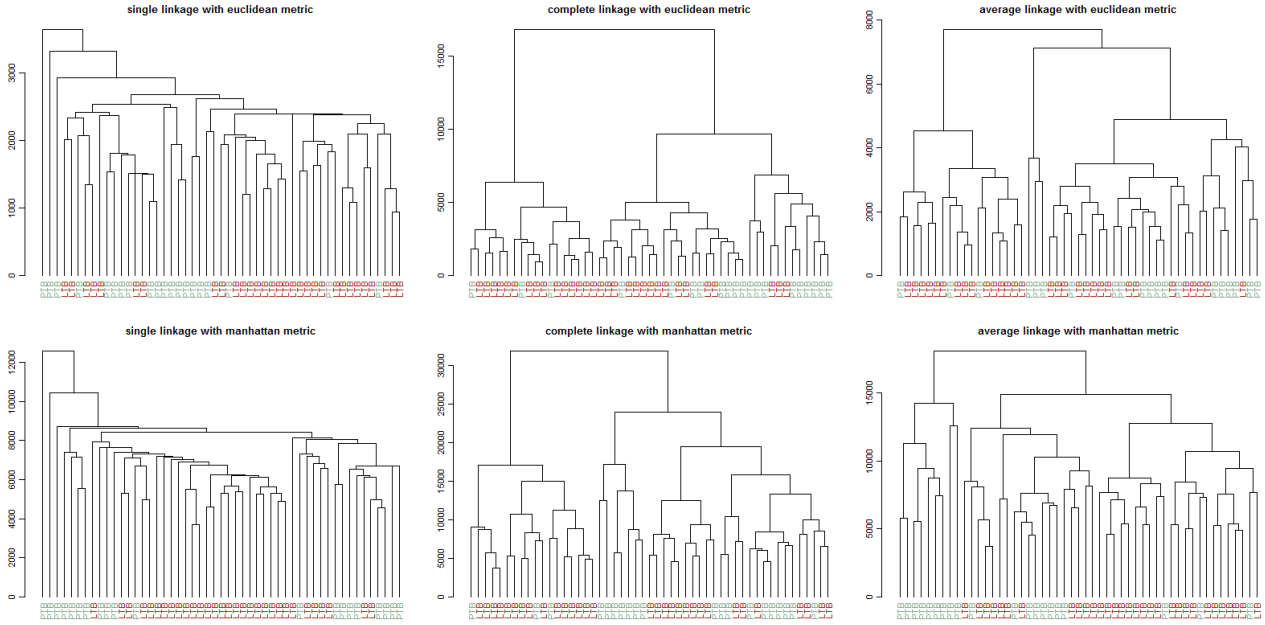


Figure 16: clustering of 100 features

Problem 3

Theoretically, the power of a binary hypothesis test is the probability that the test correctly rejects the null hypothesis (H0) when the alternative hypothesis (H1) is true.

Statistical power may depend on a number of factors. Some factors may be particular to a specific testing situation, but at a minimum, power nearly depends on the following three factors:

1. The statistical significance criterion used in the test 2. The magnitude of the effect of interest in the population 3. The sample size used to detect the effect

We have run the simulation and conducted power analysis, as is shown below:

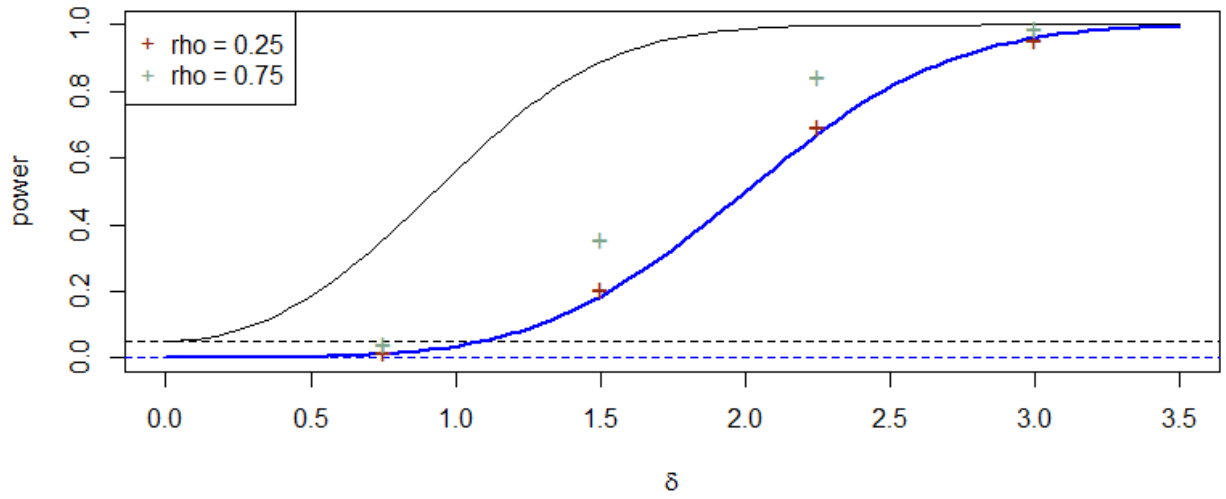


Figure 17: Power Curves delta=0.75,1.50,2.25,3.0 & rho 0.25,0.75

The figure gives information of:

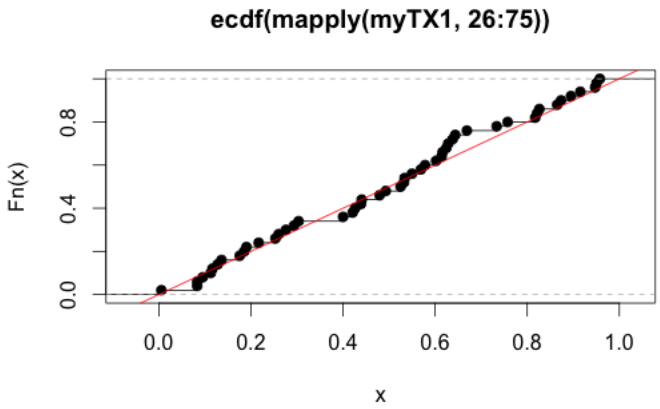
1. As the correlation increases, the power has gained a optimization. 2. When the correlation of the null group increases, maxT works better than Bonferroni adjustment. 3. Bonferroni is valid under independent scenario.

Problem 4

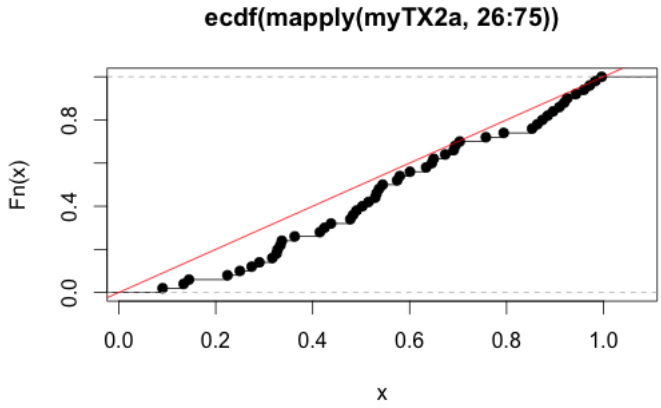
We expect to explore the effects that centering error can have on estimation and inference. We will investigate the p-value and correlation distributions for the following three datasets:

Dataset X1 - generated with no centering/batch errors - correlation among alternative targets, none among the nulls Dataset X2a- generated with centering/batch errors - correlation among alternative targets. none among the nulls Dataset X2b - re-centered about each sample (i.e., column mean)

Here we use ks test for uniform distribution (Note: reject H0 when p¡0.05, H0:uniformly distributed, that is if p¿0.05, then data is uniformly distributed) Empirical cumulative distribution function for different data set is shown as below:

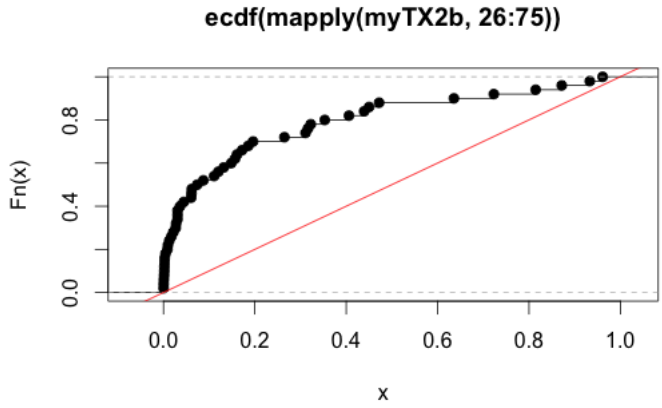


empirical cdf for X1 data



x2a.png

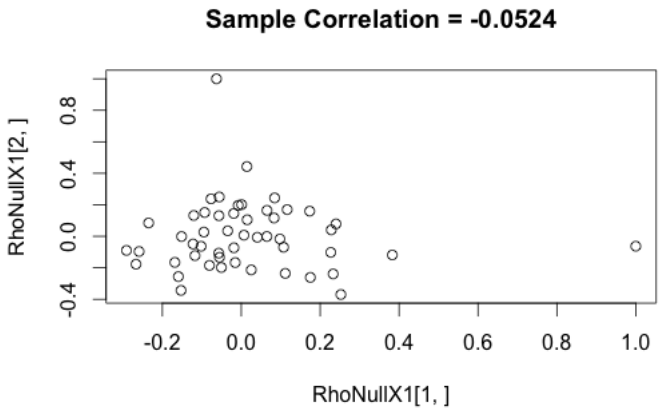
empirical cdf for X2a data



x2b.png

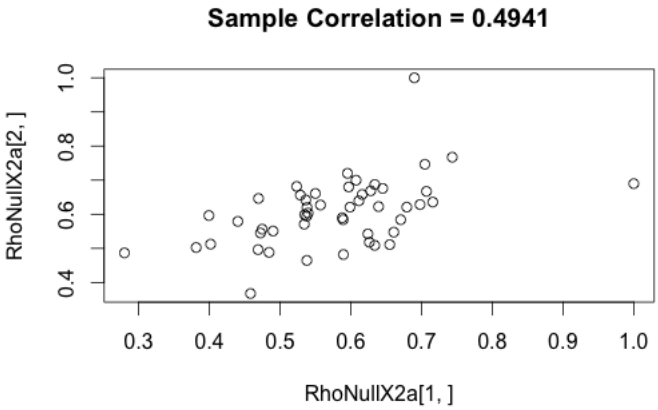
empirical cdf for X2b data

Basically X1 are uniformly distributed. Correlation estimate The number of pairwise correlation with null target is 1225. Correlation plot is shown as below:



x1.png

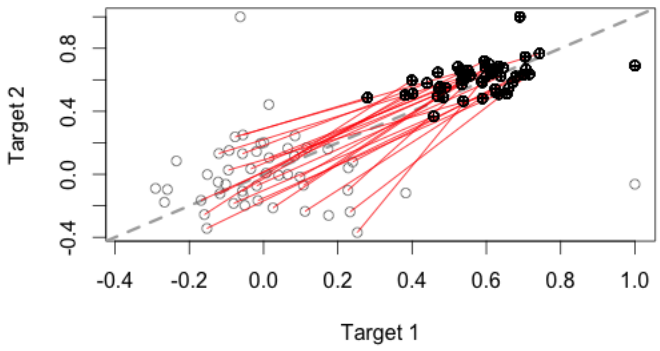
Correlation of X1 data



x2a.png

Correlation of X2a data

As is shown, X1 are centered as zero. X1 to X2a centering error: We constructed a visualized plot:

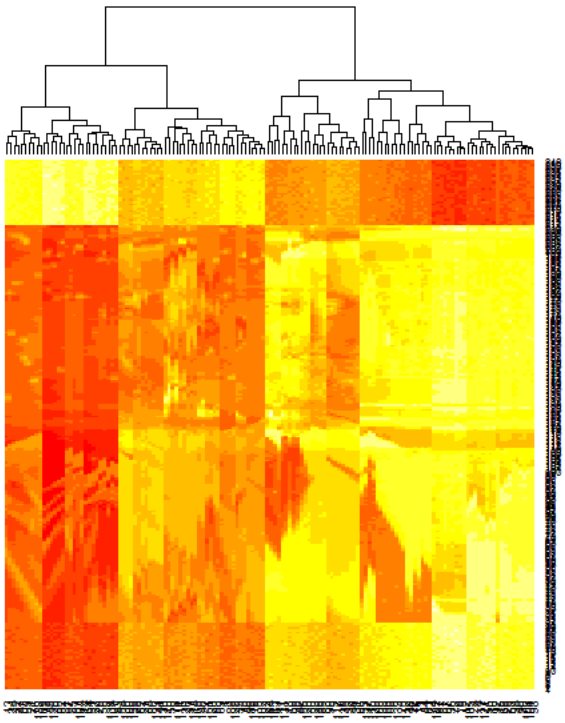


X1 to X2a centering error

Since the random error following normal distribution is added, which can be treated as a random noise. We can see from plot "X1 to X2a centering error" that the error addition does not result in a spurious inference.

Problem 5

In order to restore the proper ordering of the columns of the matrix, X, there was analysis done. The top & bottom 30 rows seemed to have a gradient pattern. By sorting the rows according to their mean, we can hope to get an order. The heatmap after sorting the matrix by their mean indicated a clear image of the picture and were reversed photos of each other.



column-wise scrambled data matrix, X

ordered data by their mean

ordered data by their mean

In order to obtain a clearer image of the picture, the correlation using pearson method between the column that has the smallest mean and the other columns. The matrix is sorted to indicate of any improvement from the previous images.

reversed heatmap : ordered data by the correlation between column having smallest mean & others

An alternative way of the attempt would be using the Principal Component Analysis. From the graph plotted with the various Principal components (fit graph), we can observe that Principal Component 1 contributes maximum

variance. First we play with whole data set, the solution seems to be in a mess. Then try out with only the top 30 rows, the solution seems to be better. Finally we treat 30 rows from both the top and bottom, we get much better solution.

rreversed heatmap : PCA with all the pigments

reversed heatmap : PCA with top 30 rows

rreversed heatmap : PCA with both top 30 rows and bottom

Appendix

R code for this homework.

```
##################################################
# STA 525: Statistics For Bioinfomatics
#      Homework Assignment #2
#
#------------------------------------------------
# Group 4 members:
#        Ziheng Cheng
#        Krithika Krishnan
#        Ziqiang Chen
##################################################

rm(list=ls())
#install.packages("diptest")
#install.packages("GEOquery")
#install.packages("wesanderson")
 library(diptest)
 library(GEOquery)
 library(wesanderson)
#-------------------
# Problem 1
#-------------------
gse19442=getGEO("GSE19442",GSEMatrix=T)
show(gse19442)
pdat19442=pData(phenoData(gse19442[[1]]))[,c(1,11:16)]


# let's extract those classes
tmp=as.character(pData(phenoData(gse19442[[1]]))[,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("PTB","LTB"))
# integer version
iFTB=as.integer(FTB)

# Create an expressionset:
eset=exprs(gse19442[[1]])

# Manipulate data set
colnames(pdat19442) <- c("group", "gender", "ethnicity", "illness", "geographical_region",
"bcg_vaccinated", "region_of_birth")

 for (j in 1:ncol(pdat19442)-1){
  tmp <- strsplit(as.character(pdat19442[,j+1]), ": ")
  tmp.lst <- c()
  for (i in 1:51) {
    tmp.lst[i] <- tmp[[i]][2]
  }
  pdat19442[,j+1] <- as.factor(tmp.lst)
}
pdat19442[,1] <-FTB


save.image("HW2/RData/q1data.RData")
```

```r
boxplot(t(x))

## Perform PCA with full data set

# Let's look at the full dataset
x=eset
fit <- prcomp(t(x),scale=T,center=T)
summary(fit) # print variance accounted for

pcx1=t(x)%*%fit$rotation[,1]
pcx2=t(x)%*%fit$rotation[,2]

plot(pcx1,pcx2,pch="␣",main="Full␣Dataset")
text(pcx1,pcx2,label=TBgroup,col=iFTB)


# For all factor levels
png("HW2/Figures/q1_1.png",width=1600,height=800,pointsize=24)
par(mfrow=c(2,3))
par(mgp=c(1.875,.75,0))
par(mar=c(3.25,3.00,2.25,0.5))

for(i in c(1,2,3,4,6,7)){
  plot(pcx1,pcx2, pch="␣", main=colnames(pdat19442)[i])
  text(pcx1,pcx2,label=pdat19442[,i],cex=0.8,
       col=wes_palettes$Cavalcanti[6-as.integer(pdat19442[,i])])
}

dev.off()


## Feature selection

MyDip=function(i) dip.test(x=eset[i,])$p.value
I=dim(eset)[1] # number of features

#this might take awhile to run.... (maybe even 15 minutes to an hour on your laptop)
mfilt=as.vector(mapply(MyDip,1:I))


hist(mfilt)



mycut=sort(mfilt)[2001]
subDX=which(mfilt<mycut)
length(subDX)
SubE=eset[subDX,]

# Let's try some other feature selection sizes.
SubE100=eset[which(mfilt<sort(mfilt)[101]),]
SubE500=eset[which(mfilt<sort(mfilt)[501]),]
SubE1000=eset[which(mfilt<sort(mfilt)[1001]),]
SubE1500=eset[which(mfilt<sort(mfilt)[1501]),]
SubE2500=eset[which(mfilt<sort(mfilt)[2501]),]
SubE5000=eset[which(mfilt<sort(mfilt)[5001]),]

save.image("HW2/RData/q1data.RData")
```

```
# let's look at subset with just 5000 features..
x=SubE5000
fit <- prcomp(t(x),scale=T,center=T)

pcx1sub5000=t(x)%*%fit$rotation[,1]
pcx2sub5000=t(x)%*%fit$rotation[,2]
plot(pcx1sub5000,pcx2sub5000,pch="␣",main="Filtered␣Dataset␣(p=5000)")
text(pcx1sub5000,pcx2sub5000,label=TBgroup,col=iFTB)

png("HW2/Figures/q1_2.png",width=1600,height=800,pointsize=24)
par(mfrow=c(2,3))
par(mgp=c(1.875,.75,0))
par(mar=c(3.25,3.00,2.25,0.5))

for(i in c(1,2,3,4,6,7)){
  plot(pcx1sub5000,pcx2sub5000, pch="␣", main=colnames(pdat19442)[i])
  text(pcx1sub5000,pcx2sub5000,label=pdat19442[,i],cex=0.8,
       col=wes_palettes$Cavalcanti[6-as.integer(pdat19442[,i])])
}

dev.off()

# let's look at subset with just 2500 features..
x=SubE2500
fit <- prcomp(t(x),scale=T,center=T)

pcx1sub2500=t(x)%*%fit$rotation[,1]
pcx2sub2500=t(x)%*%fit$rotation[,2]
plot(pcx1sub2500,pcx2sub2500,pch="␣",main="Filtered␣Dataset␣(p=2500)")
text(pcx1sub2500,pcx2sub2500,label=TBgroup,col=iFTB)

png("HW2/Figures/q1_3.png",width=1600,height=800,pointsize=24)
par(mfrow=c(2,3))
par(mgp=c(1.875,.75,0))
par(mar=c(3.25,3.00,2.25,0.5))

for(i in c(1,2,3,4,6,7)){
  plot(pcx1sub2500,pcx2sub2500, pch="␣", main=colnames(pdat19442)[i])
  text(pcx1sub2500,pcx2sub2500,label=pdat19442[,i],cex=0.8,
       col=wes_palettes$Cavalcanti[6-as.integer(pdat19442[,i])])
}

dev.off()

# let's look at subset with just 1500 features..
x=SubE1500
fit <- prcomp(t(x),scale=T,center=T)

pcx1sub1500=t(x)%*%fit$rotation[,1]
pcx2sub1500=t(x)%*%fit$rotation[,2]
plot(pcx1sub1500,pcx2sub1500,pch="␣",main="Filtered␣Dataset␣(p=1500)")
text(pcx1sub1500,pcx2sub1500,label=TBgroup,col=iFTB)

png("HW2/Figures/q1_4.png",width=1600,height=800,pointsize=24)
par(mfrow=c(2,3))
par(mgp=c(1.875,.75,0))
par(mar=c(3.25,3.00,2.25,0.5))
```

```
for(i in c(1,2,3,4,6,7)){
  plot(pcx1sub1500,pcx2sub1500, pch="␣", main=colnames(pdat19442)[i])
  text(pcx1sub1500,pcx2sub1500,label=pdat19442[,i],cex=0.8,
       col=wes_palettes$Cavalcanti[6-as.integer(pdat19442[,i])])
}

dev.off()

# let's look at subset with just 1000 features..
x=SubE1000
fit <- prcomp(t(x),scale=T,center=T)

pcx1sub1000=t(x)%*%fit$rotation[,1]
pcx2sub1000=t(x)%*%fit$rotation[,2]
plot(pcx1sub1000,pcx2sub1000,pch="␣",main="Filtered␣Dataset␣(p=1000)")
text(pcx1sub1000,pcx2sub1000,label=TBgroup,col=iFTB)

png("HW2/Figures/q1_5.png",width=1600,height=800,pointsize=24)
par(mfrow=c(2,3))
par(mgp=c(1.875,.75,0))
par(mar=c(3.25,3.00,2.25,0.5))

for(i in c(1,2,3,4,6,7)){
  plot(pcx1sub1000,pcx2sub1000, pch="␣", main=colnames(pdat19442)[i])
  text(pcx1sub1000,pcx2sub1000,label=pdat19442[,i],cex=0.8,
       col=wes_palettes$Cavalcanti[6-as.integer(pdat19442[,i])])
}

dev.off()

# let's look at subset with just 500 features..
x=SubE500
fit <- prcomp(t(x),scale=T,center=T)

pcx1sub500=t(x)%*%fit$rotation[,1]
pcx2sub500=t(x)%*%fit$rotation[,2]
plot(pcx1sub500,pcx2sub500,pch="␣",main="Filtered␣Dataset␣(p=500)")
text(pcx1sub500,pcx2sub500,label=TBgroup,col=iFTB)

png("HW2/Figures/q1_6.png",width=1600,height=800,pointsize=24)
par(mfrow=c(2,3))
par(mgp=c(1.875,.75,0))
par(mar=c(3.25,3.00,2.25,0.5))

for(i in c(1,2,3,4,6,7)){
  plot(pcx1sub500,pcx2sub500, pch="␣", main=colnames(pdat19442)[i])
  text(pcx1sub500,pcx2sub500,label=pdat19442[,i],cex=0.8,
       col=wes_palettes$Cavalcanti[6-as.integer(pdat19442[,i])])
}

dev.off()

# let's look at subset with just 100 features..
x=SubE100
fit <- prcomp(t(x),scale=T,center=T)

pcx1sub100=t(x)%*%fit$rotation[,1]
pcx2sub100=t(x)%*%fit$rotation[,2]
plot(pcx1sub100,pcx2sub100,pch="␣",main="Filtered␣Dataset␣(p=100)")
text(pcx1sub100,pcx2sub100,label=TBgroup,col=iFTB)
```

```
png("HW2/Figures/q1_7.png",width=1600,height=800,pointsize=24)
par(mfrow=c(2,3))
par(mgp=c(1.875,.75,0))
par(mar=c(3.25,3.00,2.25,0.5))

for(i in c(1,2,3,4,6,7)){
  plot(pcx1sub100,pcx2sub100, pch="␣", main=colnames(pdat19442)[i])
  text(pcx1sub100,pcx2sub100,label=pdat19442[,i],cex=0.8,
       col=wes_palettes$Cavalcanti[6-as.integer(pdat19442[,i])])
}

dev.off()

#
# We remove features, we lose variability and we run the risk of losing
# some "signal"
#

#-------------------
# Problem 2
#-------------------
rm(list=ls())
library(dendextend)
library(wesanderson)
# Load data
load("HW2/RData/q1data.RData")


#-------------------------------------------------
# using ?hcluster and ?dist
#    to see what options available
#-------------------------------------------------
# methods:
#   "single", "complete", "average"
# distances:
#   corelation-based, "euclidean", "manhattan"
#-------------------------------------------------

# part b
# Cluster with full data (hcluster)

clusters = c("single", "complete", "average")
dists = c("euclidean", "manhattan")
x=eset
colnames(x) <- FTB

png("HW2/Figures/q2_1.png",width=1600,height=800,pointsize=16)
par(mfrow=c(2,3))
par(mgp=c(1.875,.75,0))
par(mar=c(3.25,3.00,2.25,0.5))
for (j in 1:2) {
  for (i in 1:3) {
    hcex <- hclust(d=dist(t(x),method=dists[j]),method=clusters[i])
    dend <- as.dendrogram(hcex,hang=-1)
    labels_colors(dend) <- wes_palettes$Cavalcanti[6-as.integer(as.factor(labels(dend)))]
    plot(dend,main=paste(clusters[i],"linkage","with",dists[j],"metric"),cex=0.9,xlab="",sub="")
  }
}
```

```
dev.off()

## part c
## Feature selection

# let's look at subset with just 5000 features..
x=SubE5000

colnames(x) <- FTB

png("HW2/Figures/q2_2.png",width=1600,height=800,pointsize=16)
par(mfrow=c(2,3))
par(mgp=c(1.875,.75,0))
par(mar=c(3.25,3.00,2.25,0.5))
for (j in 1:2) {
  for (i in 1:3) {
    hcex <- hclust(d=dist(t(x),method=dists[j]),method=clusters[i])
    dend <- as.dendrogram(hcex,hang=-1)
    labels_colors(dend) <- wes_palettes$Cavalcanti[6-as.integer(as.factor(labels(dend)))]
    plot(dend,main=paste(clusters[i],"linkage","with",dists[j],"metric"),cex=0.9,xlab="",sub="")
  }
}

dev.off()


# let's look at subset with just 2500 features..
x=SubE2500

colnames(x) <- FTB

png("HW2/Figures/q2_3.png",width=1600,height=800,pointsize=16)
par(mfrow=c(2,3))
par(mgp=c(1.875,.75,0))
par(mar=c(3.25,3.00,2.25,0.5))
for (j in 1:2) {
  for (i in 1:3) {
    hcex <- hclust(d=dist(t(x),method=dists[j]),method=clusters[i])
    dend <- as.dendrogram(hcex,hang=-1)
    labels_colors(dend) <- wes_palettes$Cavalcanti[6-as.integer(as.factor(labels(dend)))]
    plot(dend,main=paste(clusters[i],"linkage","with",dists[j],"metric"),cex=0.9,xlab="",sub="")
  }
}

dev.off()


# let's look at subset with just 1500 features..
x=SubE1500

colnames(x) <- FTB

png("HW2/Figures/q2_4.png",width=1600,height=800,pointsize=16)
par(mfrow=c(2,3))
par(mgp=c(1.875,.75,0))
par(mar=c(3.25,3.00,2.25,0.5))
for (j in 1:2) {
  for (i in 1:3) {
    hcex <- hclust(d=dist(t(x),method=dists[j]),method=clusters[i])
    dend <- as.dendrogram(hcex,hang=-1)
```

```
    labels_colors(dend) <- wes_palettes$Cavalcanti[6-as.integer(as.factor(labels(dend)))]
    plot(dend,main=paste(clusters[i],"linkage","with",dists[j],"metric"),cex=0.9,xlab="",sub="")
  }
}

dev.off()

# let's look at subset with just 1000 features..
x=SubE1000


colnames(x) <- FTB

png("HW2/Figures/q2_5.png",width=1600,height=800,pointsize=16)
par(mfrow=c(2,3))
par(mgp=c(1.875,.75,0))
par(mar=c(3.25,3.00,2.25,0.5))
for (j in 1:2) {
  for (i in 1:3) {
    hcex <- hclust(d=dist(t(x),method=dists[j]),method=clusters[i])
    dend <- as.dendrogram(hcex,hang=-1)
    labels_colors(dend) <- wes_palettes$Cavalcanti[6-as.integer(as.factor(labels(dend)))]
    plot(dend,main=paste(clusters[i],"linkage","with",dists[j],"metric"),cex=0.9,xlab="",sub="")
  }
}

dev.off()



# let's look at subset with just 500 features..
x=SubE500

colnames(x) <- FTB

png("HW2/Figures/q2_6.png",width=1600,height=800,pointsize=16)
par(mfrow=c(2,3))
par(mgp=c(1.875,.75,0))
par(mar=c(3.25,3.00,2.25,0.5))
for (j in 1:2) {
  for (i in 1:3) {
    hcex <- hclust(d=dist(t(x),method=dists[j]),method=clusters[i])
    dend <- as.dendrogram(hcex,hang=-1)
    labels_colors(dend) <- wes_palettes$Cavalcanti[6-as.integer(as.factor(labels(dend)))]
    plot(dend,main=paste(clusters[i],"linkage","with",dists[j],"metric"),cex=0.9,xlab="",sub="")
  }
}

dev.off()




# let's look at subset with just 100 features..
x=SubE100

colnames(x) <- FTB

png("HW2/Figures/q2_7.png",width=1600,height=800,pointsize=16)
par(mfrow=c(2,3))
```

```
par(mgp=c(1.875,.75,0))
par(mar=c(3.25,3.00,2.25,0.5))
for (j in 1:2) {
  for (i in 1:3) {
    hcex <- hclust(d=dist(t(x),method=dists[j]),method=clusters[i])
    dend <- as.dendrogram(hcex,hang=-1)
    labels_colors(dend) <- wes_palettes$Cavalcanti[6-as.integer(as.factor(labels(dend)))]
    plot(dend,main=paste(clusters[i],"linkage","with",dists[j],"metric"),cex=0.9,xlab="",sub="")
  }
}

dev.off()

#------------------
# Problem 3
#------------------
rm(list=ls())
library(multtest)
library(MASS)
library(pwr)

# calculate power for n=10, sig.level=0.05, and d=1
pwr.t.test(n=10,d=1,sig.level=0.05)

# what if we wanted to know the minimum detectable effect size for 0.80 power
pwr.t.test(n=10,sig.level=0.05,power=0.8)

# now, let's calculate power for a collection of d values
dvals=seq(0,3.5,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
# make figure
plot(dvals,pwr,type="l",xlab=expression(delta),ylab="power",lwd=2,ylim=c(0,1))
abline(h=0.05,lty=2)



######################
# FWER: Bon
######################
pwrBon=rep(0,length(dvals))
for(i in 1:(length(dvals))) pwrBon[i]=pwr.t.test(n=10,d=dvals[i],sig.level=0.05/200)$power
# make figure
plot(dvals,pwr,type="l",xlab=expression(delta),ylab="power",ylim=c(0,1))
abline(h=0.05,lty=2)
lines(dvals,pwrBon,col=4,lwd=2)
abline(h=0.05/200,lty=2,col=4)




######################
# FWER: maxT
######################

#==========================================================#
#
# Simple Multivariate Normal Simulation
#
#==========================================================#
```

```
# A simulation function to generate a matrix of data:
# the simulated data will be a matrix that has:
#
# 2*n columns with the first n columns corresponding to n samples from a 'control' group
# and the second n columns correspnding to n samples from a 'treatment' group
#
# p.alt+p.null rows with the first p.alt rows corresponding to the the p.alt features (e.g. genes)
# that are diferrentially expressed and the last p.null rows corresponding to
# p.null features that aren't differentiall expressed
#
# the matrix values will all be generated from a multivariate normal distribution with covariance
# matrix:
# | 1 rho.alt rho.alt ... 0 0 0 |
# | rho.alt 1 rho.alt ... 0 0 0 |
# | :   :    :      :    :   |
# | 0 0 0 ... 1 rho.nul rho.nul |
# | 0 0 0 ... rho.nul 1 rho.nul |
# | 0 0 0 ... rho.nul rho.nul 1 |
#
# The p.null rows will all have mean zero while the p.alt rows will have mean zero for
# the first n columns and mean delta for the last n columns
# (i.e., the columns corresponding to 'treatment').
#
# Additionally, the function allows for sample specific centering error
# from a normal distribution with mean 0 and sd = sdC
SimDatHW2=function(p.alt=10, # the number of differentially expressed features
                   p.null=90, # the number of non-differentially expressed features
                   n=20, # the number of samples in each of the treatment and control groups
                   rho.alt=0.2, # correlation of the alt hyp variables
                   rho.null=0.1, # correlation of the null hyp variables
                   delta=2,# the mean of the p.alt features in the "treatment" group
                   sdC=0 # the variance of the sample specific centering error
){
  p=p.alt+p.null
  Sigma=array(rep(0,p^2),dim=c(p,p))
  Sigma[1:p.alt,1:p.alt]=rho.alt
  Sigma[(p.alt+(1:p.null)),(p.alt+(1:p.null))]=rho.null
  diag(Sigma)=1
  Xc=mvrnorm(n,mu=rep(0,p),Sigma=Sigma)
  Xt=mvrnorm(n,mu=c(rep(delta,p.alt),rep(0,p.null)),Sigma=Sigma)
  x=t(rbind(Xc,Xt))
  colnames(x)=c(paste("cntrl",1:n,sep=""),paste("trt",1:n,sep=""))
  rownames(x)=c(paste("DEgene",1:(p.alt)),paste("nonDEgene",1:p.null,sep=""))
  if(sdC>0){
    CentError=rnorm(2*n,sd=sdC)
    for(j in 1:(2*n)) x[,j]=x[,j]+CentError[j]
  }
  return(x)
}

if(F){ # I ran this as if(T) the first time, then changed it
  # so I don't have to rerun it every time
  f <- factor(c(rep("ctrl", 10), rep("trt", 10)))
  nreps=1000
  PhiVec=rep(NA,nreps)
  for(k in 1:nreps){
    X=SimDatHW2(p.alt=1,p.null=199,n=10,rho.alt=0.0,rho.null=0.75,delta=2)
    # calculate minP adjusted p-values
    resT <- mt.maxT(X, f, B = 2500)
    PhiVec[k]=as.integer(resT$adjp[resT$index==1]<=0.05)
```

```
  }
  save( file ="HW2/PhiVec.RData",PhiVec)
} # end if(F)
load( file ="HW2/PhiVec.RData") # provides PhiVec



plot(dvals,pwr,type="l",xlab=expression(delta),ylab="power",ylim=c(0,1))
abline(h=0.05,lty=2)
lines(dvals,pwrBon,col=4,lwd=2)
abline(h=0.05/200,lty=2,col=4)
# add our simulation point
points(2,mean(PhiVec),pch="+",col="darkgreen")



# Add some more points to Figure we have above

delta <- c(0.75, 1.50, 2.25, 3.0)
rho.null <- c(0.25, 0.75)

# Make a function to do that
myPhiVec <- function(rho.null,delta){
  f <- factor(c(rep("ctrl", 10), rep("trt", 10)))
  nreps <- 1000
  PhiVec <- rep(NA,nreps)
  for(k in 1:nreps){
    X <- SimDatHW2(p.alt=1,p.null=199,n=10,rho.alt=0,rho.null=rho.null,delta=delta)
    # calculate minP adjusted p-values
    resT <- mt.maxT(X, f, B = 2500)
    PhiVec[k] <- as.integer(resT$adjp[resT$index==1]<=0.05)
  }
  return(PhiVec)
}

myphivecs.rho25 <- mapply(myPhiVec, rho.null[1], delta[1:4])
myphivecs.rho75 <- mapply(myPhiVec, rho.null[2], delta[1:4])

# save(myphivecs.rho25,myphivecs.rho75,file= "HW2/myPhiVec.RData")
load("HW2/myPhiVec.RData")

png("HW2/Figures/q3_1.png",width=800,height=400,pointsize=16)
plot(dvals,pwr,type="l",xlab=expression(delta),ylab="power",ylim=c(0,1))
abline(h=0.05,lty=2)
lines(dvals,pwrBon,col=4,lwd=2)
abline(h=0.05/200,lty=2,col=4)
# add our simulation point
for(i in 1:length(deltas)){
  points(deltas[i], mean(myphivecs.rho25[,i]), pch="+", col="#972D15", cex=1.2)
  points(deltas[i], mean(myphivecs.rho75[,i]), pch="+", col="#81A88D", cex=1.2)
}
legend("topleft", pch="+", paste(expression(rho), "=", rho.null), col=c("#972D15","#81A88D"))
dev.off()

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

rm(list=ls())
library(MASS)
```

```
# Set the previous function
SimDatHW2=function(p.alt=10, # the number of differentially expressed features
p.null=90, # the number of non-differentially expressed features
n=20, # the number of samples in each of the treatment and control groups
rho.alt=0.2, # correlation of the alt hyp variables
rho.null=0.1, # correlation of the null hyp variables
delta=2,# the mean of the p.alt features in the "treatment" group
sdC=0 # the variance of the sample specific centering error
){
p=p.alt+p.null
Sigma=array(rep(0,p^2),dim=c(p,p))
Sigma[1:p.alt,1:p.alt]=rho.alt
Sigma[(p.alt+(1:p.null)),(p.alt+(1:p.null))]=rho.null
diag(Sigma)=1
Xc=mvrnorm(n,mu=rep(0,p),Sigma=Sigma)
Xt=mvrnorm(n,mu=c(rep(delta,p.alt),rep(0,p.null)),Sigma=Sigma)
x=t(rbind(Xc,Xt))
colnames(x)=c(paste("cntrl",1:n,sep=""),paste("trt",1:n,sep=""))
rownames(x)=c(paste("DEgene",1:(p.alt)),paste("nonDEgene",1:p.null,sep=""))
if(sdC>0){
CentError=rnorm(2*n,sd=sdC)
for(j in 1:(2*n)) x[,j]=x[,j]+CentError[j]
}
return(x)
}


# Dataset 1:
# generated with no centering/batch errors.
# correlation among the "alternative" targets but none among the nulls
X1= SimDatHW2(p.alt=25,p.null=50,n=20,rho.alt=.3,rho.null=0.0,delta=2,sdC=0)

# Dataset 2a:
# generated with centering/batch errors.
# correlation among the "alternative" targets but none among the nulls
X2a= SimDatHW2(p.alt=25,p.null=50,n=20,rho.alt=.3,rho.null=0.0,delta=2,sdC=1)

# Dataset 2b:
# Dataset 2a but re-centered about each sample (i.e., column) mean.
# Mean centering of this type is not an uncommon practice among small panel arrays
X2b= X2a
for(j in 1:40) X2b[,j]=X2b[,j]-mean(X2b[,j])



## Part I distribution (ecdf) of p-calues --- here we do 3 plots and 3 tests

# here we use ks test for uniform distribution
# (Note: reject H0 when p<0.05, H0:uniformly distributed, that is if p>0.05, then data is uniformly distrib

# For X1
myTX1=function(i) t.test(X1[i,1:20],X1[i,21:40],var.equal=T)$p.value
plot(ecdf(mapply(myTX1,26:75)));abline(0,1,col=2)
ks.test(mapply(myTX1,26:75),"punif")
# For X2a
myTX2a=function(i) t.test(X2a[i,1:20],X2a[i,21:40],var.equal=T)$p.value
plot(ecdf(mapply(myTX2a,26:75)));abline(0,1,col=2)
ks.test(mapply(myTX2a,26:75),"punif")
# For X2b
```

```
myTX2b=function(i) t.test(X2b[i,1:20],X2b[i,21:40],var.equal=T)$p.value
plot(ecdf(mapply(myTX2b,26:75)));abline(0,1,col=2)
ks.test(mapply(myTX2b,26:75),"punif")

# From above, basically X1 are uniformly distributed


## Part II correlation estimate

#The number of pairwise correlation with null target:
choose(50,2)
#1225


# The pairwise correlation for each group --- here we do 3 plots
# For X1
RhoNullX1=cor(t(X1[26:75,]))
hist(RhoNullX1[upper.tri(RhoNullX1)])
# For X2a
RhoNullX2a=cor(t(X2a[26:75,]))
hist(RhoNullX2a[upper.tri(RhoNullX2a)])
# For X2b
RhoNullX2b=cor(t(X2b[26:75,]))
hist(RhoNullX2b[upper.tri(RhoNullX2b)])

# From above, X1 are centered as zero


## More visualization ## --- 1 plot here from Dr. Gaile's code.

# let's look at the correlation plot of null data without centering error
plot(RhoNullX1[1,],RhoNullX1[2,],main=paste("Sample␣Correlation␣=",round(cor(RhoNullX1[1,],RhoNullX1[2,]),4)

# Now,let's look at the plot after adding centering error to each column
plot(RhoNullX2a[1,],RhoNullX2a[2,],main=paste("Sample␣Correlation␣=",round(cor(RhoNullX2a[1,],RhoNullX2a[2,]

# Let's make a more illustrative plot
xylim=range(c(RhoNullX1[1,],RhoNullX2a[1,],RhoNullX1[2,],RhoNullX2a[2,]))
plot(c(RhoNullX1[1,],RhoNullX2a[1,]),c(RhoNullX1[2,],RhoNullX2a[2,]),
xlim=xylim,ylim=xylim,type="n",xlab="Target␣1", ylab="Target␣2")
abline(0,1,lty=2,lwd=3,col="gray67")
# now,let's add original points
points(RhoNullX1[1,],RhoNullX1[2,],col="gray47")
# now, let's add the points with centering error and connect with lines
for(j in 26:dim(X2a)[1]){
lines(c(RhoNullX1[1,j],RhoNullX2a[1,j]),c(RhoNullX1[2,j],RhoNullX2a[2,j]),col=2)
points(RhoNullX2a[1,],RhoNullX2a[2,],pch=10)
}


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

rm(list = ls())
load("HW2/HW2X.RData")
dim(X)
colnames(X) = as.character(c(1:ncol(X))) # name the columns
heatmap(X,Rowv = NA)
```

```
# The bottom and top 30 rows seems to have some gradient pattern
# so we can try to order these rows by their mean.
colmean.t30 = colMeans(X[c(1:30),])
ordDX <- order(colmean.t30)
heatmap(X[,ordDX],Rowv=NA,Colv=NA)

colmean.b30 = colMeans(X[c(211:240),])
ordDX <- order(colmean.b30)
heatmap(X[,ordDX],Rowv=NA,Colv=NA)


# Seems that the two heatmaps above are two reversed version of the photo for some guy..

# Next we try to make these figures more clear..

# We can use the correlation between the column that has the smallest mean and other columns.
# Then we order the correlation to see whether there is an improvement.

n = ncol(X)
ordDX =rep(NA,n)
ordDX[1] = order(colmean.t30)[1] # assign the first is the minimum column mean
others = X[,-ordDX[1]]
myCol = X[,ordDX[1]]

for (i in 1:(n - 1)) {
  myCor = 0
  for (j in 1:(n - i)){
    myCor[j] = cor(myCol,others[,j],method = "pearson")
  }
  maxDX = which.max(myCor)
  myCol = others[,maxDX,drop = F]
  others = others[,-maxDX,drop = F]
  ordDX[i + 1] = as.numeric(colnames(myCol))

}
heatmap(X[,ordDX],Rowv = NA,Colv = NA)

# Then we have more clear image for/of our advisor - Dr. Gaile


##====================================
## Another attempt (PCA)
##====================================

# Play with whole data set at first
fit <- prcomp(t(X),scale=T,center=T)

# Form the following plot, the first component attribute most variation
plot(fit,type="l")

# Now we can see the order of "items" in the first component
ordDX <- order(fit$x[,1])

# then we see what happens
heatmap(X[,ordDX],Rowv=NA,Colv=NA)

## not good..

##----------------------------
## (inspired by Jinchen)
```

```
## Try only the first 30 rows..
##----------------------------
fit <- prcomp(t(X[1:30,]),scale=T,center=T)

# Form the following plot, the first component attribute most variation
plot(fit,type="l")

# Now we can see the order of "items" in the first component
ordDX <- order(fit$x[,1])

# then we see what happens
heatmap(X[,ordDX],Rowv=NA,Colv=NA)

# Looks better


##----------------------------
## Try both top and bottom
##----------------------------

fit <- prcomp(t(X[c(1:30,211:240),]),scale=T,center=T)

# Form the following plot, the first component attribute most variation
plot(fit,type="l")

# Now we can see the order of "items" in the first component
ordDX <- order(fit$x[,1])

# then we see what happens
heatmap(X[,ordDX],Rowv=NA,Colv=NA)

## Much better..
```