Bioinformatics HW 4

Ryan Gallagher

2023-10-30

## 1) Download and read in a data package “golubEsets” into R. Read in the dataset “Golub\_Merge”. This is a leukemia dataset.

### a) How many genes and how many samples are in the data set? How many acute lymphoblastic leukemia (ALL) patients and how many acute myeloid leukemia (ALM) patients in the dataset? How many male, female, and unknown patients?

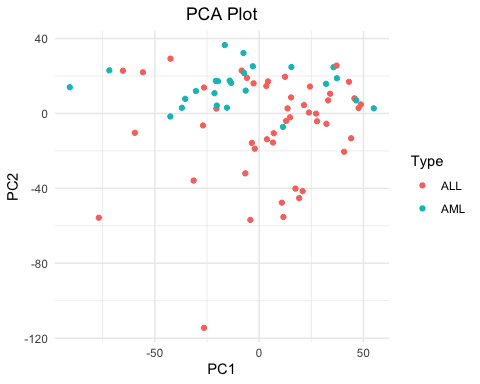
* There are 7129 genes in the dataset.
* There are 72 samples in the dataset.
* There are 47 patients with acute lymphoblastic leukemia (ALL)
* There are 25 patients with acute myeloid leukemia (AML)
* Gender is distributed as: 23, 26
* With ‘NA’ accounting for 23

table(Golub\_Merge$Gender)

##   
## F M   
## 23 26

### b) Use all genes and do dimension redution PCA. Color ALL and AML patients on the plot.

pheno\_data = pData(Golub\_Merge)  
expression\_mat = exprs(Golub\_Merge)  
  
pca = prcomp(t(expression\_mat), scale.=TRUE)  
  
pca\_dat = data.frame(PC1 = pca$x[,1],  
 PC2 = pca$x[,2],  
 Type = factor(pheno\_data$ALL.AML))  
  
ggplot(pca\_dat, aes(x=PC1, PC2, color=Type)) + geom\_point() +  
 theme\_minimal() +   
 labs(title="PCA Plot", x="PC1", y="PC2", color="Type") +  
 theme(plot.title = element\_text(hjust=0.5))



### c) Use “samr” package in R to detect differentially expressed genes (use random seed 12345) between ALL and AML patients. You will have the read the document from the package to learn how to use the function correctly. How many ALL up-regulated and down-regulated genes are identified when FDR threshold is 5%? How about when FDR is 1%? List the top 10 up-regulated and down-regulared DE genes respectively. Plot a heatmap for identified DE genes at 1% threshold.

#install.packages('samr')  
library(samr)  
set.seed(12345)  
class\_vec = factor(pData(Golub\_Merge)$ALL.AML)  
  
sam\_dat = list(x = expression\_mat,  
 y = class\_vec,  
 logged2=TRUE)  
  
samr\_o = samr(data = sam\_dat,  
 resp.type = "Two class unpaired",  
 nperms = 100)

Finding how many ALL up-regulated and down-regulated genes are identified when FDR threshold is 5%:

delta.table = samr.compute.delta.table(samr\_o)  
  
### use delta.table to find which delta yields median FDR of 0.01 or 0.05  
  
#delta = 0.88  
siggenes.table = samr.compute.siggenes.table(samr\_o, del=0.88, sam\_dat, delta.table)  
  
low\_tab = siggenes.table$genes.lo  
low\_tab = low\_tab[as.numeric(low\_tab[,8])<1,]  
  
high\_tab = siggenes.table$genes.up  
high\_tab = high\_tab[as.numeric(high\_tab[,8])<1,]  
  
up = nrow(high\_tab)  
lo = nrow(low\_tab)

There are 484 up-regulated genes for 5% threshold.  
There are 467 down-regulated genes for 5% threshold.

Top 10 up\_regulated

head(as.data.frame(high\_tab)$`Gene ID`, 10)

## [1] "g4847" "g4196" "g2288" "g1882" "g1834" "g6041" "g3252" "g2121" "g1745"  
## [10] "g1829"

Top 10 down-regulared

head(as.data.frame(low\_tab)$`Gene ID`, 10)

## [1] "g4328" "g2354" "g6855" "g2642" "g6225" "g6281" "g804" "g1685" "g5772"  
## [10] "g1144"

delta.table = samr.compute.delta.table(samr\_o)  
siggenes.table = samr.compute.siggenes.table(samr\_o, del=1.23, sam\_dat, delta.table)  
  
low\_tab = siggenes.table$genes.lo  
low\_tab = low\_tab[as.numeric(low\_tab[,8])<1,]  
  
high\_tab = siggenes.table$genes.up  
high\_tab = high\_tab[as.numeric(high\_tab[,8])<1,]  
  
up = nrow(high\_tab)  
lo = nrow(low\_tab)

There are 484 up-regulated genes for 1% threshold.  
There are 467 down-regulated genes for 1% threshold.

Top 10 up\_regulated

head(as.data.frame(high\_tab)$`Gene ID`, 10)

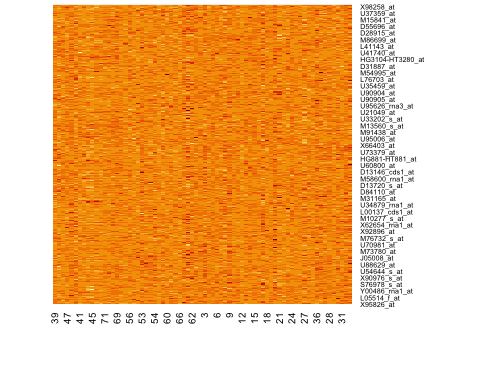
## [1] "g4847" "g4196" "g2288" "g1882" "g1834" "g6041" "g3252" "g2121" "g1745"  
## [10] "g1829"

Top 10 down-regulared

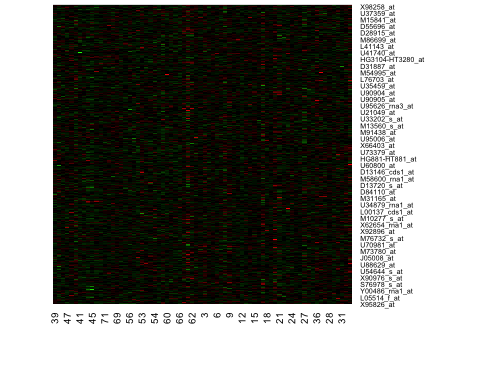
head(as.data.frame(low\_tab)$`Gene ID`, 10)

## [1] "g4328" "g2354" "g6855" "g2642" "g6225" "g6281" "g804" "g1685" "g5772"  
## [10] "g1144"

heatmap.data=sam\_dat$x[as.numeric(c(high\_tab[,1], low\_tab[,1])),]  
heatmap(heatmap.data, Rowv=NA, Colv=NA)



B=16  
heatmap(heatmap.data, Rowv=NA, Colv=NA, col= rgb(c(rep(0, B), (0:B)/B), c((B:0)/16, rep(0, B)), rep(0, 2\*B+1)))

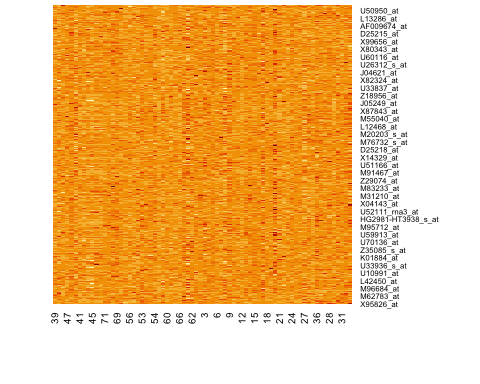


### d) Use t-test with Benjamini-Hochberg correction to detect differentially expressed genes under FDR=1%. How many differentially expressed genes are detected? Also plot a heatmap for these DE genes. How does the result compare to (c) and which method is more powerful?

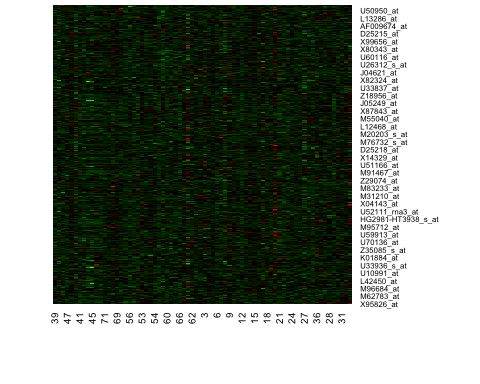
low\_tab = siggenes.table$genes.lo  
low\_tab = as.data.frame(low\_tab) %>%   
 mutate(q = as.numeric(`q-value(%)`)) %>%   
 arrange(q) %>%  
 mutate(benj = p.adjust(q, 'hochberg'))  
   
low\_tab = low\_tab[as.numeric(low\_tab[,10])<1,]  
  
high\_tab = siggenes.table$genes.up  
high\_tab = as.data.frame(high\_tab) %>%   
 mutate(q = as.numeric(`q-value(%)`)) %>%   
 arrange(q) %>%  
 mutate(benj = p.adjust(q, 'hochberg'))   
  
high\_tab = high\_tab[as.numeric(high\_tab[,10])<1,]  
  
up = nrow(high\_tab)  
lo = nrow(high\_tab)

There are 502 DE genes expressed in this method

heatmap.data=sam\_dat$x[as.numeric(c(high\_tab[,1], low\_tab[,1])),]  
heatmap(heatmap.data, Rowv=NA, Colv=NA)

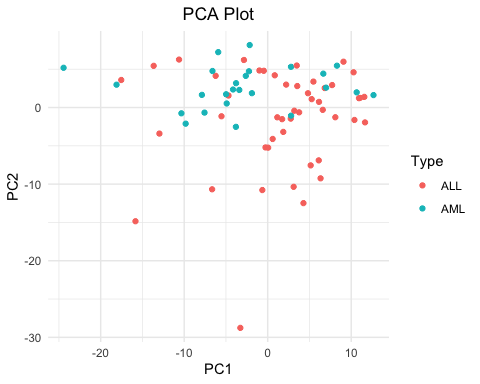


B=16  
heatmap(heatmap.data, Rowv=NA, Colv=NA, col= rgb(c(rep(0, B), (0:B)/B), c((B:0)/16, rep(0, B)), rep(0, 2\*B+1)))

 These results are much more conservative than (c). We find that using the Hochberg method yields a less powerful results.

### (e) Use DE genes detected by either sam or t-test at FDR=1% and do dimension reduction by PCA. Compare to (b). What’s your observation?

pca = prcomp(t(sam\_dat$x[as.numeric(c(high\_tab[,1], low\_tab[,1])),]), scale.=TRUE)  
  
pca\_dat = data.frame(PC1 = pca$x[,1],  
 PC2 = pca$x[,2],  
 Type = factor(pheno\_data$ALL.AML))  
  
ggplot(pca\_dat, aes(x=PC1, PC2, color=Type)) + geom\_point() +  
 theme\_minimal() +   
 labs(title="PCA Plot", x="PC1", y="PC2", color="Type") +  
 theme(plot.title = element\_text(hjust=0.5))

 These appear quite similar

## 2) Download R package ‘impute’. Use data(khanmiss) to access the R object khanmiss. This dataset includes random missing data values. We are going to use it to try imputation methods. To save time running this dataset, subset the first 100 genes and save it as a new dataset.

### a) How many missing values in the dataset? How many genes have missing values?

library(impute)  
  
data(khanmiss)  
babykhan = as.data.frame(khanmiss[2:100,-c(1,2)])  
babykhan = data.frame(lapply(babykhan, function(x) as.numeric(as.character(x))))  
  
rows\_with\_na = apply(babykhan, 1, function(x) any(is.na(x)))  
babykhan[rows\_with\_na, 1:5]

## sample1 sample2 sample3 sample4 sample5  
## 19 -1.0463994 -1.2990164 -0.9043622 NA NA  
## 28 1.2651432 1.3307817 NA -0.08904982 0.2911760  
## 40 -1.5950420 -1.9351681 NA -0.33813362 -0.6737368  
## 44 -0.2201480 -0.3928945 NA NA -0.8335593  
## 60 NA 0.7689965 1.4276522 0.83286564 1.3110588  
## 67 -2.7135654 -2.5626520 -1.9951004 -2.02798826 -1.9011280  
## 84 -0.3319823 0.1814879 0.8091063 0.62165118 0.9190868

print(paste("There are", sum(is.na(babykhan)), "missing values in this dataset"))

## [1] "There are 43 missing values in this dataset"

We find that there are 43 missing values in our dataset. We also find that 7 genes have missing data.

### b) Write your own code to define K-nearest neighbors for genes that you found in (a). Use the neighbors you defined to do imputation. Try different selection of K and summarize what you observe.

knn\_impute = function(dataset, k = 3) {  
  
   
 #dataset = data.frame(lapply(dataset, function(x) as.numeric(as.character(x))))  
  
 distance = function(row1, row2) {  
 valid\_columns = !is.na(row1) & !is.na(row2)  
 sum((row1[valid\_columns] - row2[valid\_columns])^2)  
 }  
 imputed\_data = dataset  
 for (i in 1:nrow(dataset)) {  
 # Find rows with missing data  
 na\_columns = which(is.na(dataset[i, ]))  
 for (col in na\_columns) {  
 distances = apply(dataset[-i, , drop = FALSE], 1, function(x) distance(dataset[i, ], x))  
   
 nearest\_indices = order(distances)[1:k]  
 imputed\_data[i, col] = mean(dataset[nearest\_indices, col], na.rm = TRUE)  
 }  
 }  
  
 return(imputed\_data)  
}

khan\_5 = knn\_impute(babykhan, k = 2)  
print(paste("There are", sum(is.na(khan\_5)), "missing values in this dataset"))

## [1] "There are 0 missing values in this dataset"

khan\_5[c(19, 28, 40, 44,60,67,84), 1:5]

## sample1 sample2 sample3 sample4 sample5  
## 19 -1.0463994 -1.2990164 -0.9043622 -1.39124875 -0.6887713  
## 28 1.2651432 1.3307817 0.6174580 -0.08904982 0.2911760  
## 40 -1.5950420 -1.9351681 0.1782314 -0.33813362 -0.6737368  
## 44 -0.2201480 -0.3928945 -1.3206421 -1.16183210 -0.8335593  
## 60 1.4887607 0.7689965 1.4276522 0.83286564 1.3110588  
## 67 -2.7135654 -2.5626520 -1.9951004 -2.02798826 -1.9011280  
## 84 -0.3319823 0.1814879 0.8091063 0.62165118 0.9190868

khan\_5 = knn\_impute(babykhan, k = 3)  
print(paste("There are", sum(is.na(khan\_5)), "missing values in this dataset"))

## [1] "There are 0 missing values in this dataset"

khan\_5[c(19, 28, 40, 44,60,67,84), 1:5]

## sample1 sample2 sample3 sample4 sample5  
## 19 -1.0463994 -1.2990164 -0.9043622 -0.60111488 -0.02532074  
## 28 1.2651432 1.3307817 0.4756193 -0.08904982 0.29117596  
## 40 -1.5950420 -1.9351681 -0.2347586 -0.33813362 -0.67373679  
## 44 -0.2201480 -0.3928945 -1.1025965 -1.06087662 -0.83355933  
## 60 1.4942923 0.7689965 1.4276522 0.83286564 1.31105883  
## 67 -2.7135654 -2.5626520 -1.9951004 -2.02798826 -1.90112801  
## 84 -0.3319823 0.1814879 0.8091063 0.62165118 0.91908682

khan\_5 = knn\_impute(babykhan, k = 5)  
print(paste("There are", sum(is.na(khan\_5)), "missing values in this dataset"))

## [1] "There are 0 missing values in this dataset"

khan\_5[c(19, 28, 40, 44,60,67,84), 1:5]

## sample1 sample2 sample3 sample4 sample5  
## 19 -1.0463994 -1.2990164 -0.9043622 -0.22672356 0.2477416  
## 28 1.2651432 1.3307817 0.4005431 -0.08904982 0.2911760  
## 40 -1.5950420 -1.9351681 -0.1360074 -0.33813362 -0.6737368  
## 44 -0.2201480 -0.3928945 -1.0419293 -0.84195457 -0.8335593  
## 60 1.5605787 0.7689965 1.4276522 0.83286564 1.3110588  
## 67 -2.7135654 -2.5626520 -1.9951004 -2.02798826 -1.9011280  
## 84 -0.3319823 0.1814879 0.8091063 0.62165118 0.9190868

khan\_5 = knn\_impute(babykhan, k = 10)  
print(paste("There are", sum(is.na(khan\_5)), "missing values in this dataset"))

## [1] "There are 0 missing values in this dataset"

khan\_5[c(19, 28, 40, 44,60,67,84), 1:5]

## sample1 sample2 sample3 sample4 sample5  
## 19 -1.0463994 -1.2990164 -0.9043622 -0.02052477 0.4053586  
## 28 1.2651432 1.3307817 0.3505320 -0.08904982 0.2911760  
## 40 -1.5950420 -1.9351681 -0.7195615 -0.33813362 -0.6737368  
## 44 -0.2201480 -0.3928945 -0.8653962 -0.64686052 -0.8335593  
## 60 1.4220402 0.7689965 1.4276522 0.83286564 1.3110588  
## 67 -2.7135654 -2.5626520 -1.9951004 -2.02798826 -1.9011280  
## 84 -0.3319823 0.1814879 0.8091063 0.62165118 0.9190868

### c) Use the function impute.knn and compare with results using your own ocde. Are they the same?

khan\_5 = impute.knn(as.matrix(babykhan), k = 2)$data  
print(paste("There are", sum(is.na(khan\_5)), "missing values in this dataset"))

## [1] "There are 0 missing values in this dataset"

khan\_5[c(19, 28, 40, 44, 60, 67, 84), 1:5]

## sample1 sample2 sample3 sample4 sample5  
## [1,] -1.0463994 -1.2990164 -0.9043622 -0.75467124 -0.4205634  
## [2,] 1.2651432 1.3307817 0.8739697 -0.08904982 0.2911760  
## [3,] -1.5950420 -1.9351681 -0.4130867 -0.33813362 -0.6737368  
## [4,] -0.2201480 -0.3928945 -0.4434955 -0.61472993 -0.8335593  
## [5,] 1.6875211 0.7689965 1.4276522 0.83286564 1.3110588  
## [6,] -2.7135654 -2.5626520 -1.9951004 -2.02798826 -1.9011280  
## [7,] -0.3319823 0.1814879 0.8091063 0.62165118 0.9190868

khan\_5 = impute.knn(as.matrix(babykhan), k = 3)$data  
print(paste("There are", sum(is.na(khan\_5)), "missing values in this dataset"))

## [1] "There are 0 missing values in this dataset"

khan\_5[c(19, 28, 40, 44, 60, 67, 84), 1:5]

## sample1 sample2 sample3 sample4 sample5  
## [1,] -1.0463994 -1.2990164 -0.9043622 -0.90343992 -0.1687901  
## [2,] 1.2651432 1.3307817 0.6466271 -0.08904982 0.2911760  
## [3,] -1.5950420 -1.9351681 -0.6797290 -0.33813362 -0.6737368  
## [4,] -0.2201480 -0.3928945 -0.5396263 -0.46102578 -0.8335593  
## [5,] 1.6249554 0.7689965 1.4276522 0.83286564 1.3110588  
## [6,] -2.7135654 -2.5626520 -1.9951004 -2.02798826 -1.9011280  
## [7,] -0.3319823 0.1814879 0.8091063 0.62165118 0.9190868

khan\_5 = impute.knn(as.matrix(babykhan), k = 5)$data  
print(paste("There are", sum(is.na(khan\_5)), "missing values in this dataset"))

## [1] "There are 0 missing values in this dataset"

khan\_5[c(19, 28, 40, 44, 60, 67, 84), 1:5]

## sample1 sample2 sample3 sample4 sample5  
## [1,] -1.0463994 -1.2990164 -0.9043622 -0.89942783 0.1000996  
## [2,] 1.2651432 1.3307817 0.5031478 -0.08904982 0.2911760  
## [3,] -1.5950420 -1.9351681 -0.5930291 -0.33813362 -0.6737368  
## [4,] -0.2201480 -0.3928945 -0.5794432 -0.31282303 -0.8335593  
## [5,] 1.6257192 0.7689965 1.4276522 0.83286564 1.3110588  
## [6,] -2.7135654 -2.5626520 -1.9951004 -2.02798826 -1.9011280  
## [7,] -0.3319823 0.1814879 0.8091063 0.62165118 0.9190868

khan\_5 = impute.knn(as.matrix(babykhan), k = 10)$data  
print(paste("There are", sum(is.na(khan\_5)), "missing values in this dataset"))

## [1] "There are 0 missing values in this dataset"

khan\_5[c(19, 28, 40, 44, 60, 67, 84), 1:5]

## sample1 sample2 sample3 sample4 sample5  
## [1,] -1.0463994 -1.2990164 -0.9043622 -0.42106279 0.1596587  
## [2,] 1.2651432 1.3307817 0.2999865 -0.08904982 0.2911760  
## [3,] -1.5950420 -1.9351681 -0.6394597 -0.33813362 -0.6737368  
## [4,] -0.2201480 -0.3928945 -0.6189699 -0.29042917 -0.8335593  
## [5,] 1.4684643 0.7689965 1.4276522 0.83286564 1.3110588  
## [6,] -2.7135654 -2.5626520 -1.9951004 -2.02798826 -1.9011280  
## [7,] -0.3319823 0.1814879 0.8091063 0.62165118 0.9190868

Comparing my code with the function, and looking at k=2. We see that the imputed values from my code is 1.488, while the function yields 1.68. These are clearly not the same!