İlknur Baş
21601847
MBG-326
Term Project

Term Project

(Note: when I have convert to PDF, the quality of images has changed.)

Since I have the latest R versions, I have used the following code instead of biocLite

```
>BiocManager::install("multtest")

>library("multtest")
```

Then, loaded the golub dataset which includes 3051 genes and 38 tumor mRNA samples. Also, I can easily see the gene expression datas by clicking "**golub**" in the R-programming environment. It is a large matrix and row indicates genes, columns indicates samples (patients).

```
>data(golub)
```

1)

golub : The golub table contains gene expression values from 3051 genes taken from 38 from the leukemia micro patients.

golub.gnames: The golub.gnames matrix contains the names of 3051 genes. (we can access it from the environment section of R program) There are 3 columns which corresponds to including gene index, ID, and biological name.

golub.cl: A vector which indicates the tumor classes by using 0 and 1. When I have compiled ">golub.cl" , it shows 27 zeros for acute lymphoblastic leukemia patients and 11 ones for cute myeloid leukemia patients, in total 38 tumor mRNA samples.

2)

a) The probe set of my is gene is 9.

b) In order to get the name the gene, I have written following codes

```
> rowName<-golub.gnames[9,]
> rowName
[1] "45"                            "AFFX-HSAC07/X00351_5_at (endogenous control)"
[3] "AFFX-HSAC07/X00351_5_at"

#Then, to reach the name of my gene,
> rowName[3]
[1] "AFFX-HSAC07/X00351_5_at"
```

c) Following code shows all the gene expression data of the specific row,9.

```
> golub[9,]
 [1] 3.22372 3.09954 2.99977 3.34097 3.27515 3.49405 3.21234 3.25724 2.84099 3.45407
3.31366 3.59116 3.04257
[14] 3.28765 3.08083 3.29544 2.66927 3.68131 3.12728 2.17590 0.53551 3.10338
3.70837 2.95165 2.86640 2.72070
[27] 3.28996 3.16946 3.04535 2.89050 3.27934 3.09505 2.88576 3.09109 3.20545
2.81781 3.01414 3.19656
```

In order to find the average expression of 9th row, I have used mean function.

```
> averageExpression <-mean(golub[9,])
> averageExpression
[1] 3.0613
```

d) Class 0 and class 1 is not differentially expressed because in part c I have found the average expression which is approximately 3. Then I have compared it with the gene expression values for 9th row.(Note: first 27 column indicates class 0 and last 11 column indicates class 1). The values are pretty similar to the average expression value.

e) In this part, we can analyze from the box-plot whether the gene is differently expressed between class 0 and class 1 or not.

In r programming factors are used to categorize the data and store it as levels. Since we need to categorize 2 data(class0 and class1), I have used this function.

```
> categorize <- factor(golub.cl, levels=0:1, labels = c("class 0","class 1"))
```

To reach class 0 which indicates the first 27 value, I have compiled this following code.

```
> golub[9, categorize =="class 0"]
 [1] 3.22372 3.09954 2.99977 3.34097 3.27515 3.49405 3.21234 3.25724 2.84099 3.45407
3.31366 3.59116 3.04257
[14] 3.28765 3.08083 3.29544 2.66927 3.68131 3.12728 2.17590 0.53551 3.10338
3.70837 2.95165 2.86640 2.72070
[27] 3.28996
```

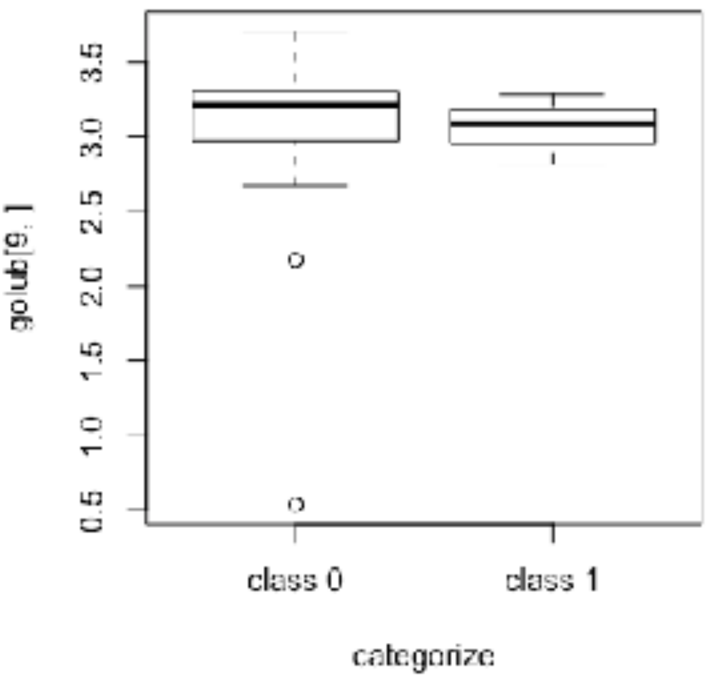
To reach class 1 which indicates the last 11 value, I have compiled this following code.

```
 > golub[9, categorize =="class 1"]
 [1] 3.16946 3.04535 2.89050 3.27934 3.09505 2.88576 3.09109 3.20545 2.81781 3.01414
3.19656
```

In order to draw boxplots,

```
> boxplot(golub[9,] ~ categorize)
```

I can also see from the box plot, the classes have similar gene expression values since they placed similarly. So, we can say that the gene is not differently expressed between class 0 and class 1.



f)
Since it is asked to find the top 10 most positively correlated genes with my gene, I need to compare my gene with other genes 3050 genes. I have written a while loop for the comparison and put the values into a vector. (variable c is used for intention purposes.)

```
> myList<-c()
> i<-1
> while (i<=3051){
+ if(i==9) c<-1
+ else myList<-c(myList,cor(golub[9,],golub[i,]))
+ i<-i+1
+ }
```

Now myList vector has the correlated genes. I want to reach positive ones so sort function is used.

```
> maximum<-sort(myList,decreasing = TRUE)
> maximum[1:10]
 [1] 0.9486591 0.9425543 0.9342749 0.9308348 0.8987936 0.8973011 0.8960285
0.7981863 0.7653650 0.7447439
```

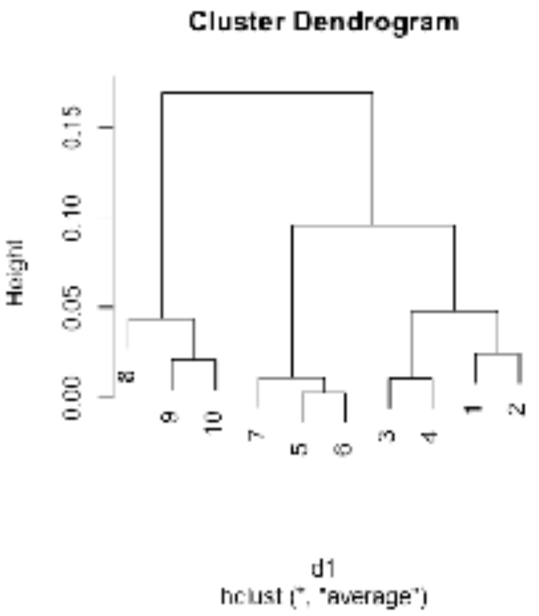
g)
This below values are negatively correlated genes.

```
> minimum<-sort(myList,decreasing = FALSE)
> minimum[1:10]
 [1] -0.7301470 -0.7063955 -0.6768836 -0.6671904 -0.6135381 -0.6117644 -0.6022769
-0.5949451 -0.5919169
[10] -0.5894360
```

h)
To perform clustering we need dimensions. However maximum and minimum are vectors. So I have put them into data frame

```
> frameNew<-data.frame(maximum[1:10],minimum[1:10])
> frameNew
  maximum.1.10. minimum.1.10.
1      0.9486591    -0.7301470
2      0.9425543    -0.7063955
3      0.9342749    -0.6768836
4      0.9308348    -0.6671904
5      0.8987936    -0.6135381
6      0.8973011    -0.6117644
7      0.8960285    -0.6022769
8      0.7981863    -0.5949451
9      0.7653650    -0.5919169
10     0.7447439    -0.5894360
```

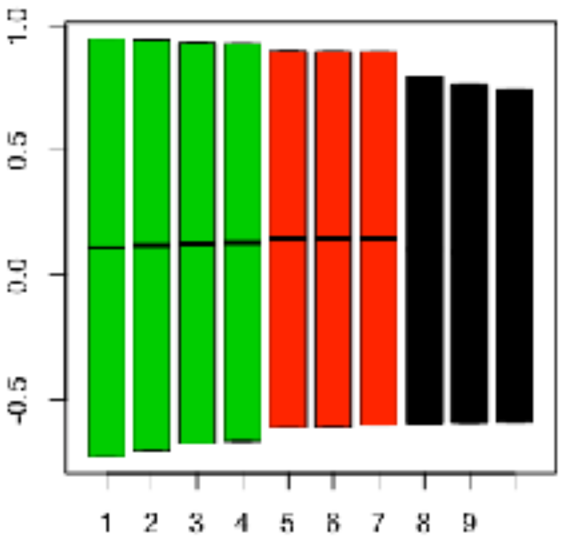
a) hierarchical clustering



I have used euclidean method because it is most appropriate for my data.

```
> d1<-dist(frameNew,method="euclidean")
> dendogram<hclust(d1,method="average",member=NULL)
>plot( dendogram)
```

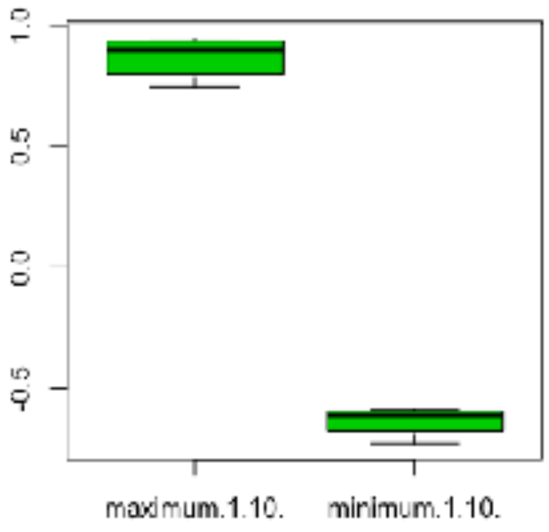
I have used the average linkage because it calculates the distance between any two clusters A and B is the average of all distances from an object in cluster A to an object in cluster B.(topology and branch lengths can be changed when we have changed the method)



b) k-means clustering

```
>km<-kmeans(frameNew,centers = 3 )
>boxplot (t(frameNew),col=km$cluster)
```

I want to use 3 clusters for k-means because when I have see the cluster dendrogram in previous part, we can say that there are 3 clusters(8-9-10, 7-6-5, 1-2-3-4).

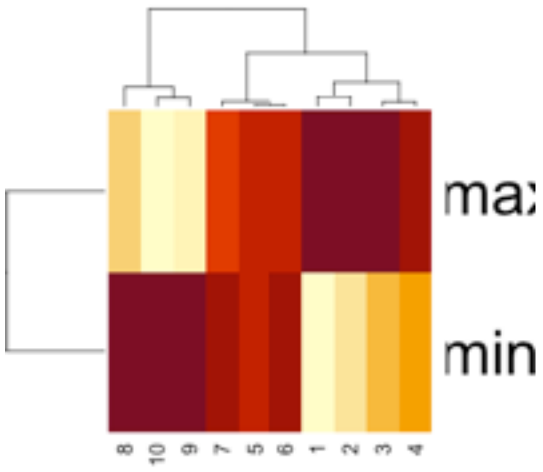


And also from the 2nd box plot (we can find it from writing following codes

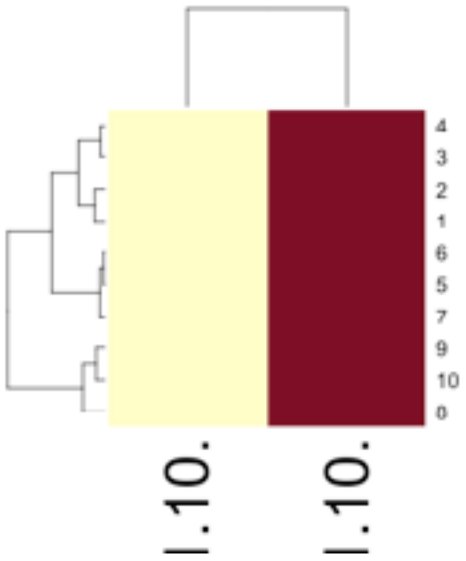
```
>km<-kmeans(frameNew,centers = 3 )
>boxplot (frameNew,col=km$cluster)
```

),we can cluster the columns. Since the only difference between them their signs(positive, negative), the plots are in the opposite side of the graph.

i) > heatmap(as.matrix(t(frameNew)))



The darker parts shows that they have more correlation. For instance, 1,2, and 3 in max part is darker so that means they have more correlated with my gene. For instance, 1,2, and 3 in min part is less correlated with my gene



Also, we can see the heatmap of maximum and minimum expression value clusters, by writing the following code;

> heatmap(as.matrix(frameNew))

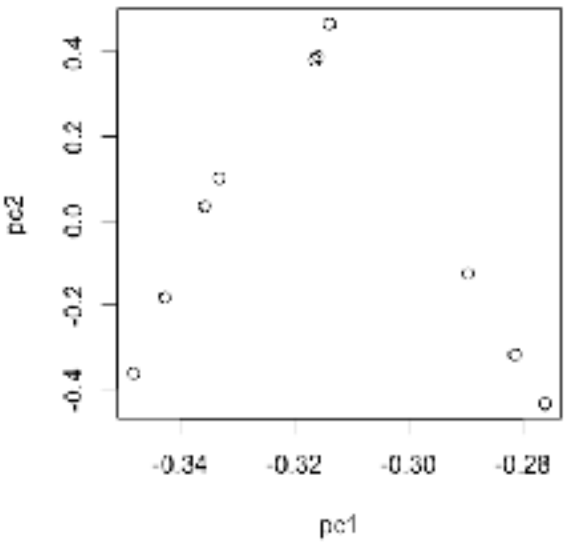
j)

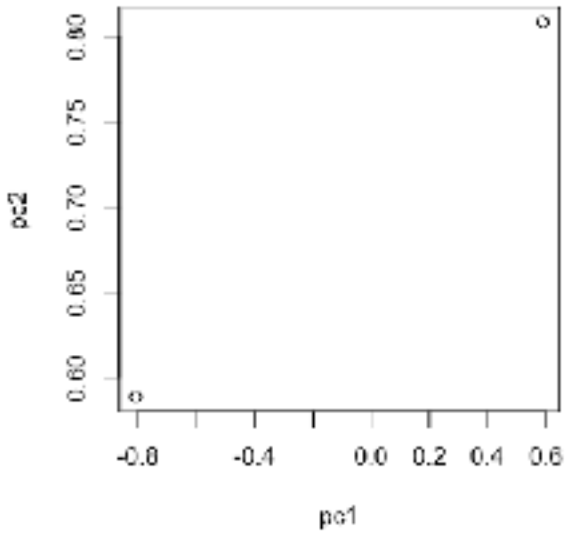
a)

```
> result<-prcomp(t(frameNew),center=FALSE)
> scores<-result$rotation
> plot(scores[,1],scores[,2] ,xlab=“pc1",ylab="pc2")
```

This PCA shows us that, there is less space between similar genes, so that we can understand they can form a cluster together.
Also, we can see from the 2nd PCA plot that maximum and minimum expression value clusters, by changing the following code:

```
> result<-prcomp( frameNew),center=FALSE)
```

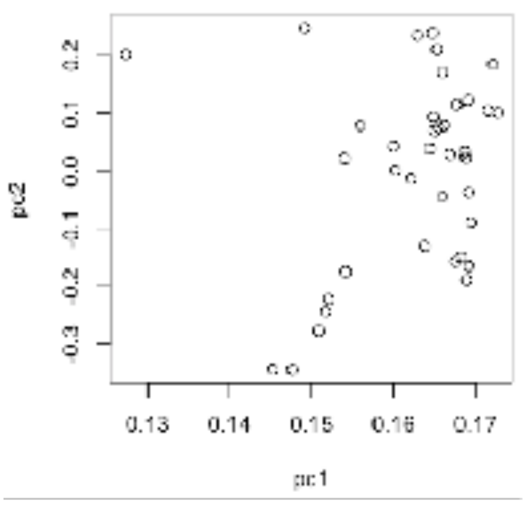


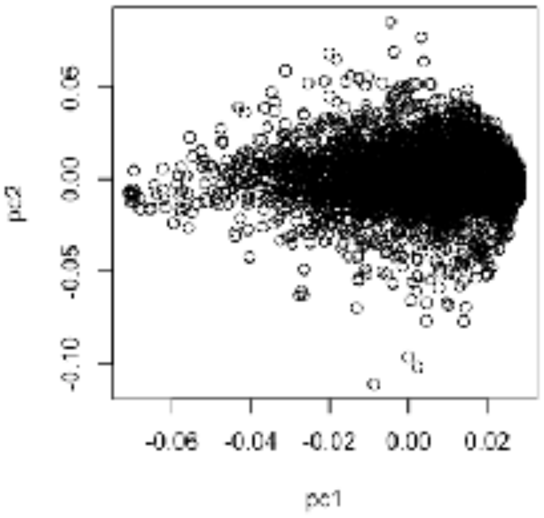


b)

>result2<-prcomp(golub,center=FALSE)
> scores<-result2$rotation
> plot(scores[,1],scores[,2] ,xlab="pc1",ylab="pc2")

The plots are different from each other because the values are making the difference. The PCA plot that contains all gene expression values for 38 samples, however the j part-a PC contains 20 genes 10 ten samples. They have similar clusters but in the end there is difference between PCA plots.
Also the 2nd PCA plot shows the transpose of golub.


pc2
pc1
0.13
0.14
0.15
0.16
0.17
-0.3
-0.2
-0.1
0.0
0.1
0.2


pc2
pc1
-0.06
-0.04
-0.02
0.00
0.02
-0.10
-0.05
0.00
0.05