HW3

Anindita Deb
adeb3

04-03-2022

1 Answer 1

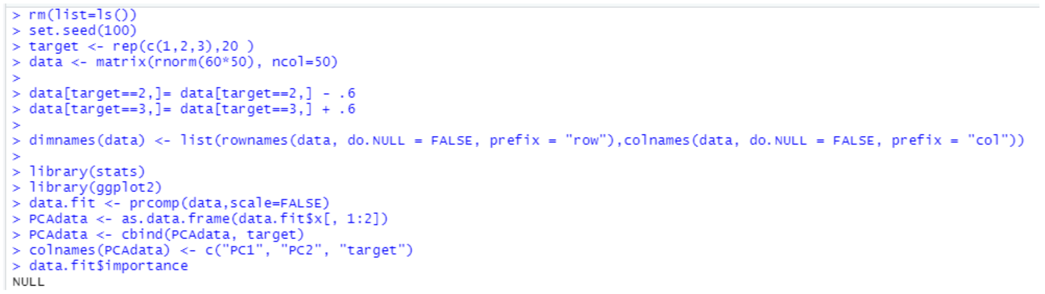
1. There are two parts to this question - First we need to answer in which linkage the two clusters will merge faster and then at what height they would converge.In order to answer the first part we need to have the distance matrix between this two clusters -When we converge two clusters we take this decision on the basis of minimum distance linkage does not play role **directly** what matters whether the clusters are having minimum distance compared to other clusters .However when we update the distance matrix we taking into ***consideration the underlying rule of different kinds of linkages***.The values in the distance matrix finally would decide when the clusters will converge and if at all they would converge or not and if at all they are converging the **height** would be again determined by the distance matrix which could be ***different in case of complete and single linkage***

2. There are two parts to this question - First we need to answer in which linkage the two clusters will merge faster and then at what height they would converge.In order to answer the first part we need to have the distance matrix between this two clusters -When we converge two clusters we take this decision on the basis of minimum distance linkage does not play role **directly** what matters whether the clusters are having minimum distance compared to other clusters .However when we update the distance matrix we taking into ***consideration the underlying rule of different kinds of linkages***.The values in the distance matrix finally would decide when the clusters will converge and if at all they

would converge or not and if at all they are converging the **height** would be again determined by the distance matrix which could be ***different in case of complete and single linkage*** We don't have sufficient information about the distance matrices and hence we cannot reach to any conclusion regarding convergence or depth of convergence.

2 Answer 2

1. **Part A** The below code snapshot shows a simulated data set with 20 observations in each of three classes (i.e. 60 observations total), and 50 variables has been generated.Also added a mean shift to the observations in each class so that there are three distinct classes.Here I am performing principal components analysis using the prcomp() where I have set the scale to **False** since variables are already scaled.



```
> rm(list=ls())
> set.seed(100)
> target <- rep(c(1,2,3),20 )
> data <- matrix(rnorm(60*50), ncol=50)
>
> data[target==2,]= data[target==2,] - .6
> data[target==3,]= data[target==3,] + .6
>
> dimnames(data) <- list(rownames(data, do.NULL = FALSE, prefix = "row"),colnames(data, do.NULL = FALSE, prefix = "col"))
>
> library(stats)
> library(ggplot2)
> data.fit <- prcomp(data,scale=FALSE)
> PCAdata <- as.data.frame(data.fit$x[, 1:2])
> PCAdata <- cbind(PCAdata, target)
> colnames(PCAdata) <- c("PC1", "PC2", "target")
> data.fit$importance
NULL
```

Figure 1: "Code Snapshot"

2. **Part B** Prinicipal Component analysis to check the usefulness of the randomly generated data - whether it could be used for modelling with Kmeans.

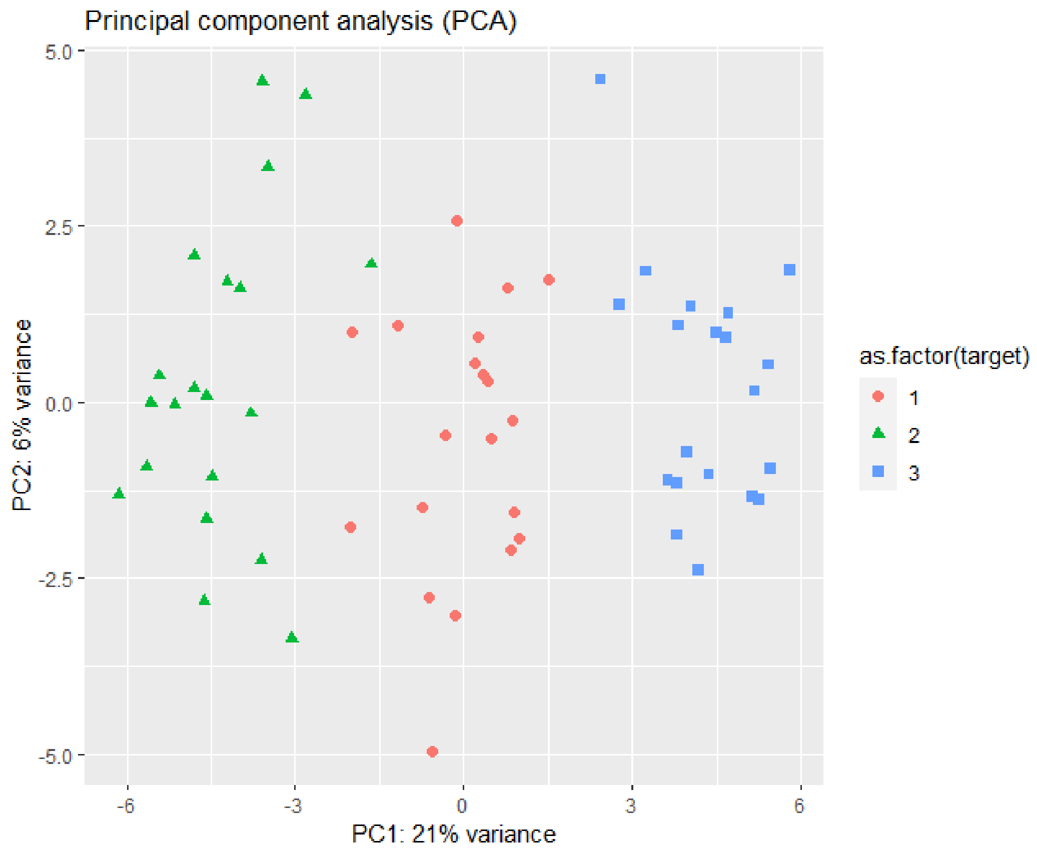
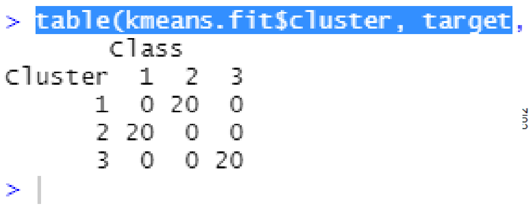


Figure 2: "Principal Component Score Plot"

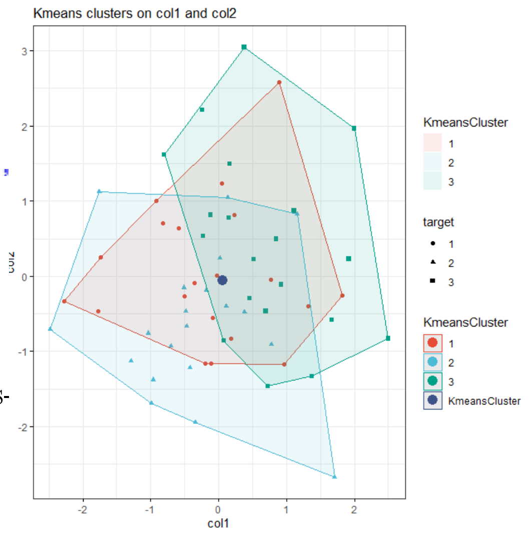
From the above plot we can see the first component does a good job of separating the classes, however the second class does not so good.As a clear separation between the classes could be obtained using first two principal components **thus for modelling purpose it is desirable to use this simulated dataset.** .

3. **Part C** K means Clustering with K = 3



```
> table(kmeans.fit$cluster, target,
       Class
Cluster  1  2  3
      1  0 20  0
      2 20  0  0
      3  0  0 20
> 
```

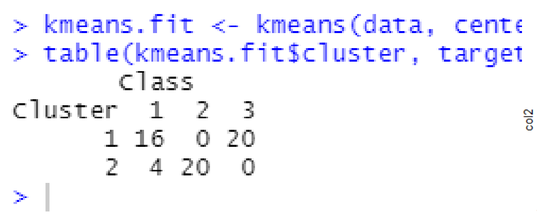
"Contigency table of the assigned cluster and the actual class for k=3"



"Kmeans plot on first two variable"

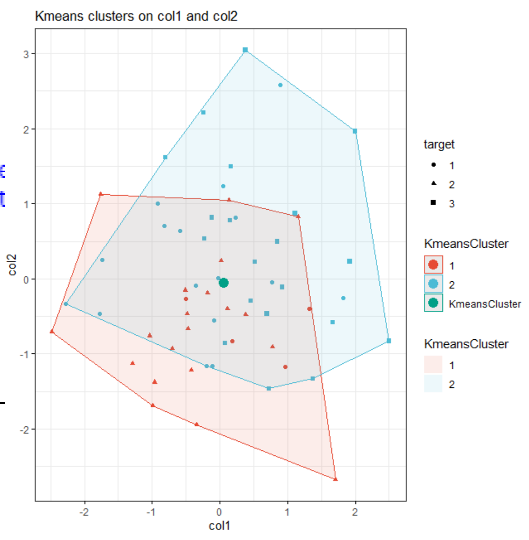
From the table above- although K-means clustering will arbitrarily number the clusters, we see that each arbitrary cluster is assigned to one class only. K-means performs perfectly in this case. From the Kmeans plot for k=3- it seems for all the clusters have some overalppings with only one centroid,thus data is not separated clearly.

4. **Part D** K means Clustering with K = 2



```
> kmeans.fit <- kmeans(data, cente
> table(kmeans.fit$cluster, target
       Class
Cluster  1  2  3
      1 16  0 20
      2  4 20  0
> 
```

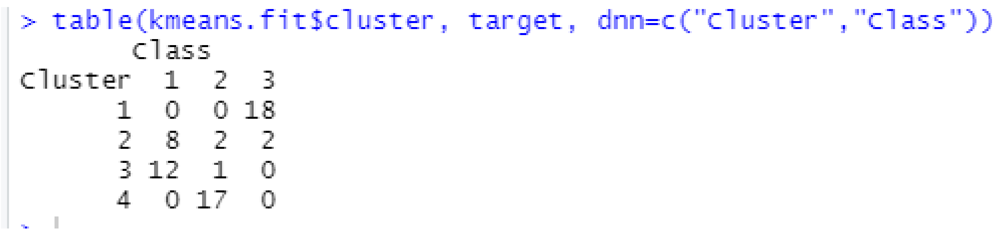
"Contigency table of the assigned cluster and the actual class for k=2"



"Kmeans plot on first two variable"

In the table above we see that two of the classes are perfectly assigned to two clusters. ie. class 2 to cluster 2 and class 3 to cluster 1.However as we only have two classes K-means has forced the class 1 to be split among the two clusters - however, majority of class1 observations go to cluster 1.

5. **Part E** K-means clustering with K = 4

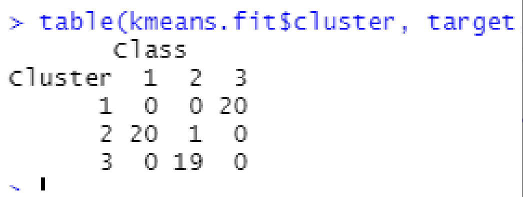


```
> table(kmeans.fit$cluster, target, dnn=c("Cluster","Class"))
       Class
Cluster  1  2  3
      1  0  0 18
      2  8  2  2
      3 12  1  0
      4  0 17  0
```

Figure 3: "Contigency table of the assigned cluster and the actual class for k=4"

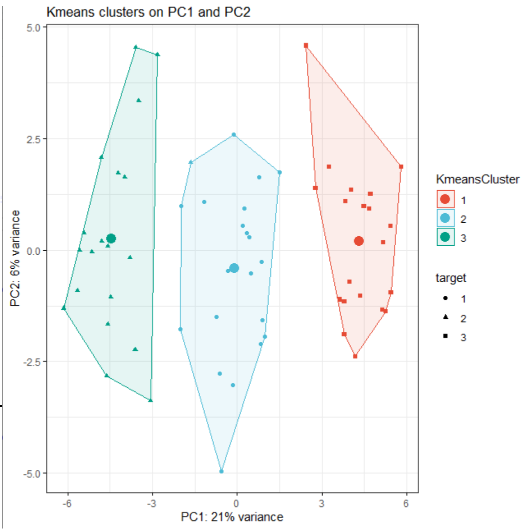
In the table above, we see that the majority of observations in each class are assigned to one cluster only. Class 1 to cluster 1. Class 2 to cluster 2. Class 3 to cluster 3.

6. **Part F** K-means clustering with K = 3 on the first two Principal component score vectors



```
> table(kmeans.fit$cluster, target
       Class
Cluster  1  2  3
      1  0  0 20
      2 20  1  0
      3  0 19  0
```

"Contigency table of the assigned cluster and the actual class for k=3"



"Kmeans clusters on first two principal components for k=3"

From the above plots it can be seen that only the first two principal components almost perfectly separate each class into unique clusters.Thus on comparing with part c results its clear that kmeans combined with pca does a good job in separating the data into three distinct classes.

7. **Part G** K-means clustering with K = 3 on the data after scaling
First lets check if the data (data.scale) is scaled to standard deviation of 1.From below boxplots it appears like it is. We do notice the the columnwise means have shifted slightly.

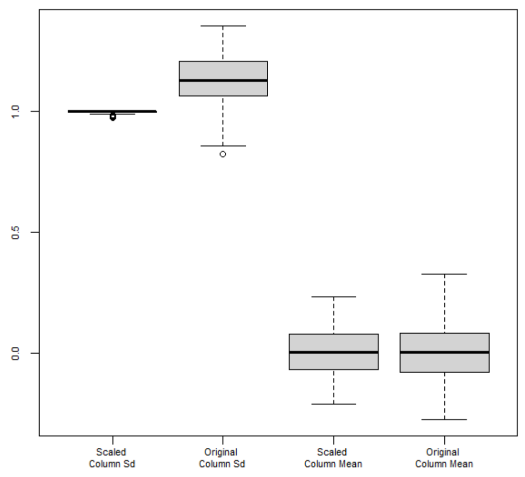


Figure 4: "Comparing scaled data with original data using BoxPlots after taking column wise means"

Lets examine the rowwise means to see how they compare after the scaling. The below plot shows a slight change in rowwise mean of each class however the general mean shift performed in part (a) of the question is very much kept intact.

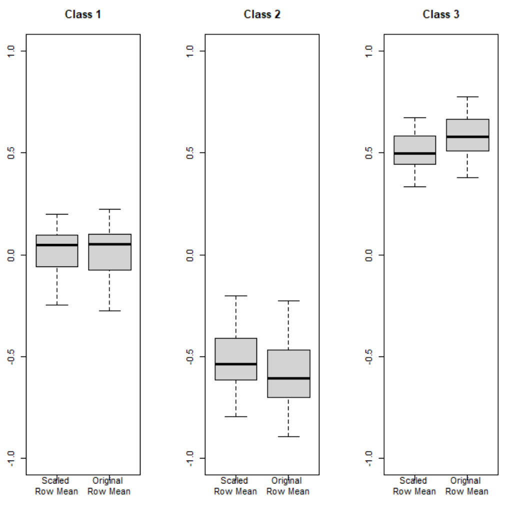


Figure 5: "Comparing scaled data with original data for each class using BoxPlots after taking rowise means"

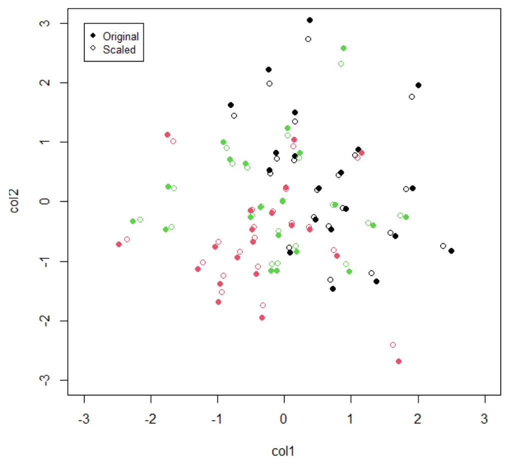


Figure 6: "Comparing scaled data with original data distribution for first two columns"

Given the above boxplots it can be seen that the data within each cluster does not move much with the scaling, so we would expect that the clustering should be able to get similar results with the columnwise scaling of SD as without the scaling of SD in part (c).

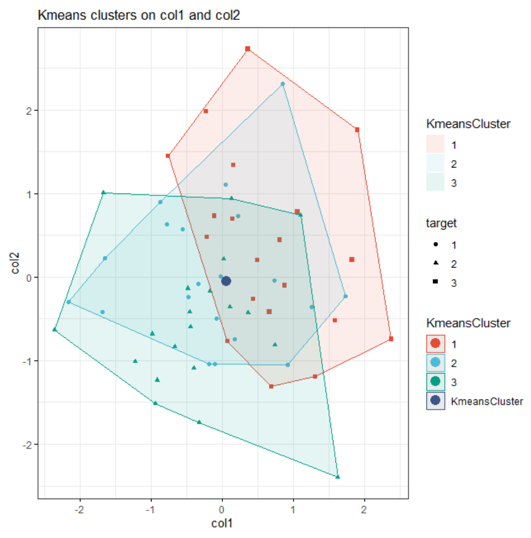
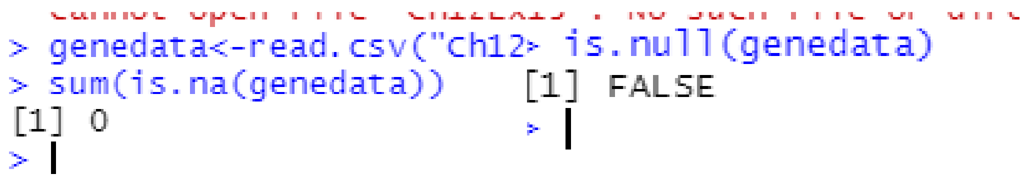


Figure 7: "Kmeans clusters on first two columns"

From the above plot it seems the results are quite similar to what we obtained without scaling data in part b

3 Answer 3

1. Checking null and missing values in the gene expression data.



```
> genedata<-read.csv("Ch12
> sum(is.na(genedata))
[1] 0
>
```

```
> is.null(genedata)
[1] FALSE
>
```

"No missing values" "No null data"

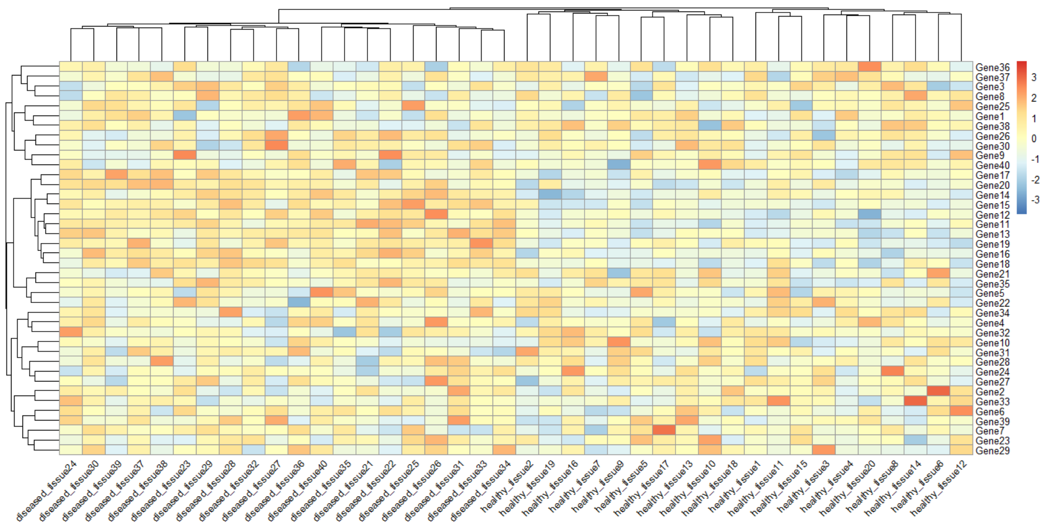


Figure 8: "Heat Map for the healthy and diseased tissues for first 40 genes"

2. **Part B**

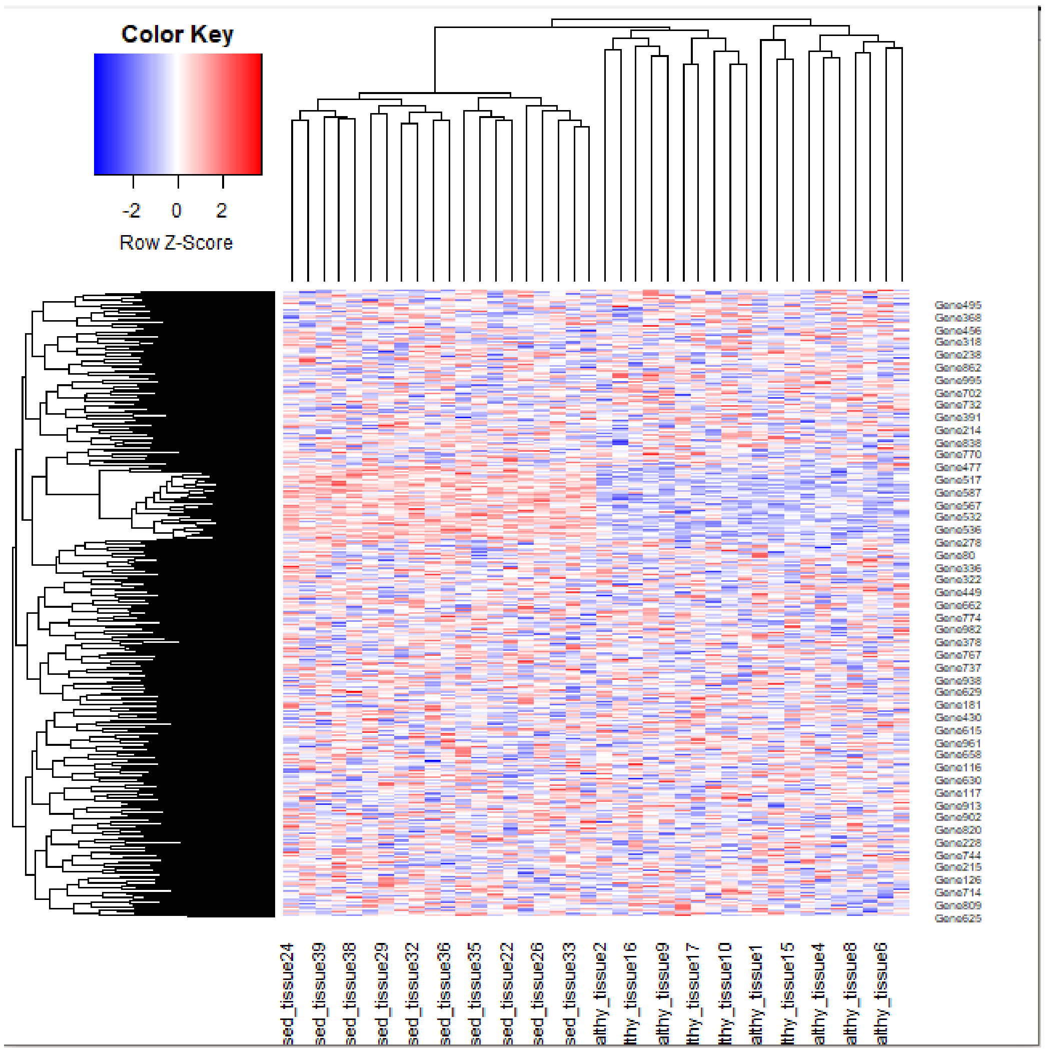


Figure 9: "Heat Map of hierarchial clustering on features and genes using complete linkage"

The above plot shows hierarchial Clustering on genes as well as tissues with correlation as distance matrix and complete linkage

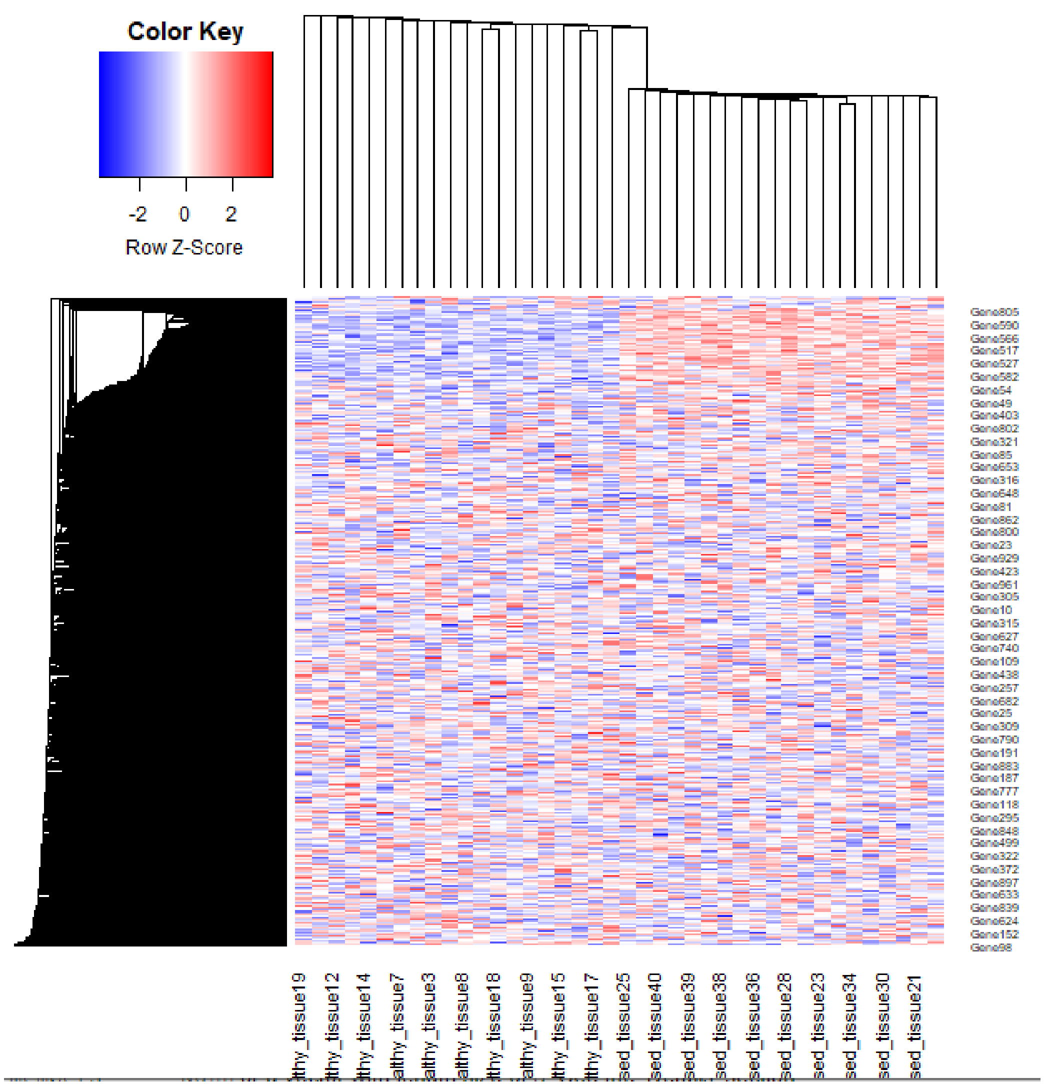


Figure 10: "Heat Map of hierarchial clustering on features and genes using single linkage"

The above plot shows hierarchial Clustering on genes as well as tissues with correlation as distance matrix and single linkage

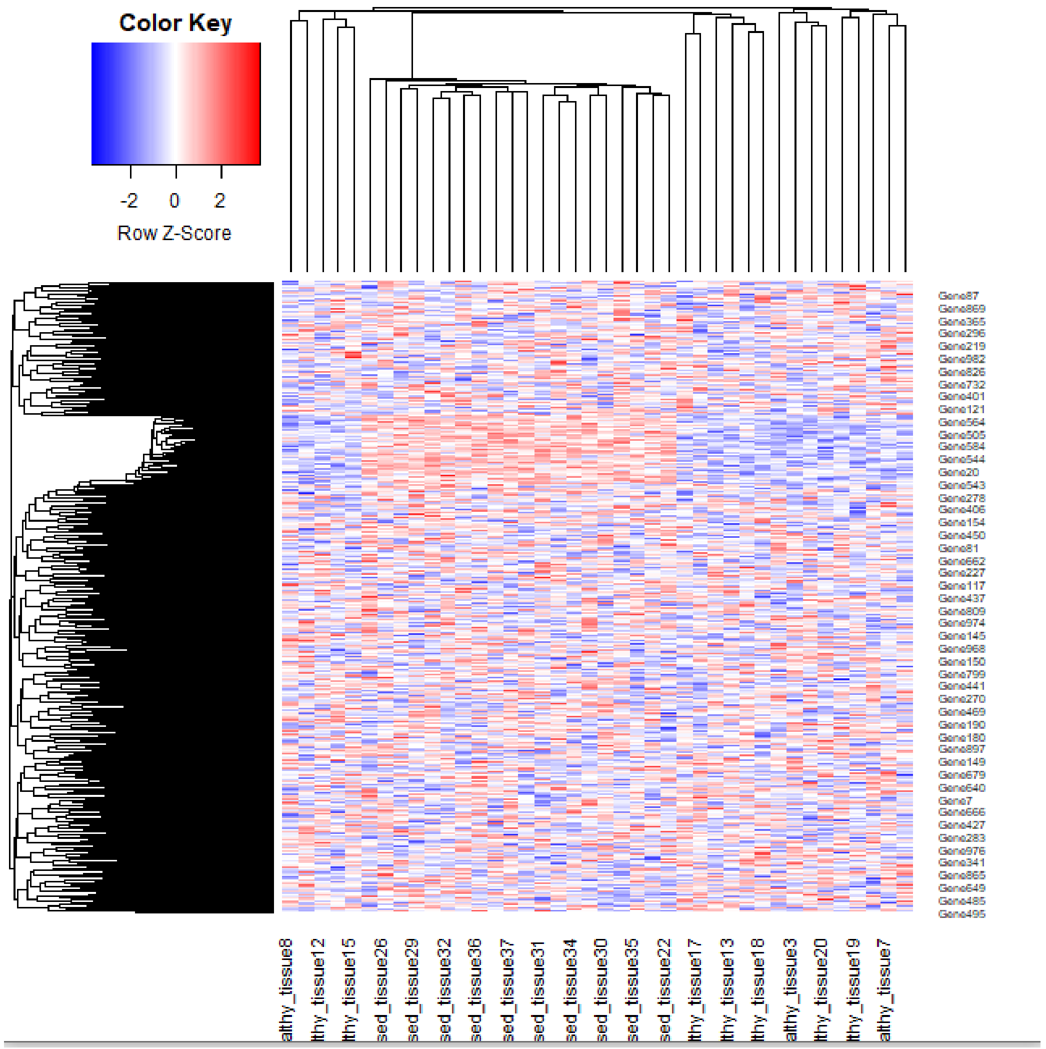


Figure 11: "Heat Map of hierarchial clustering on features and genes using average linkage"

The above plot shows hierarchial Clustering on genes as well as tissues with correlation as distance matrix and average linkage

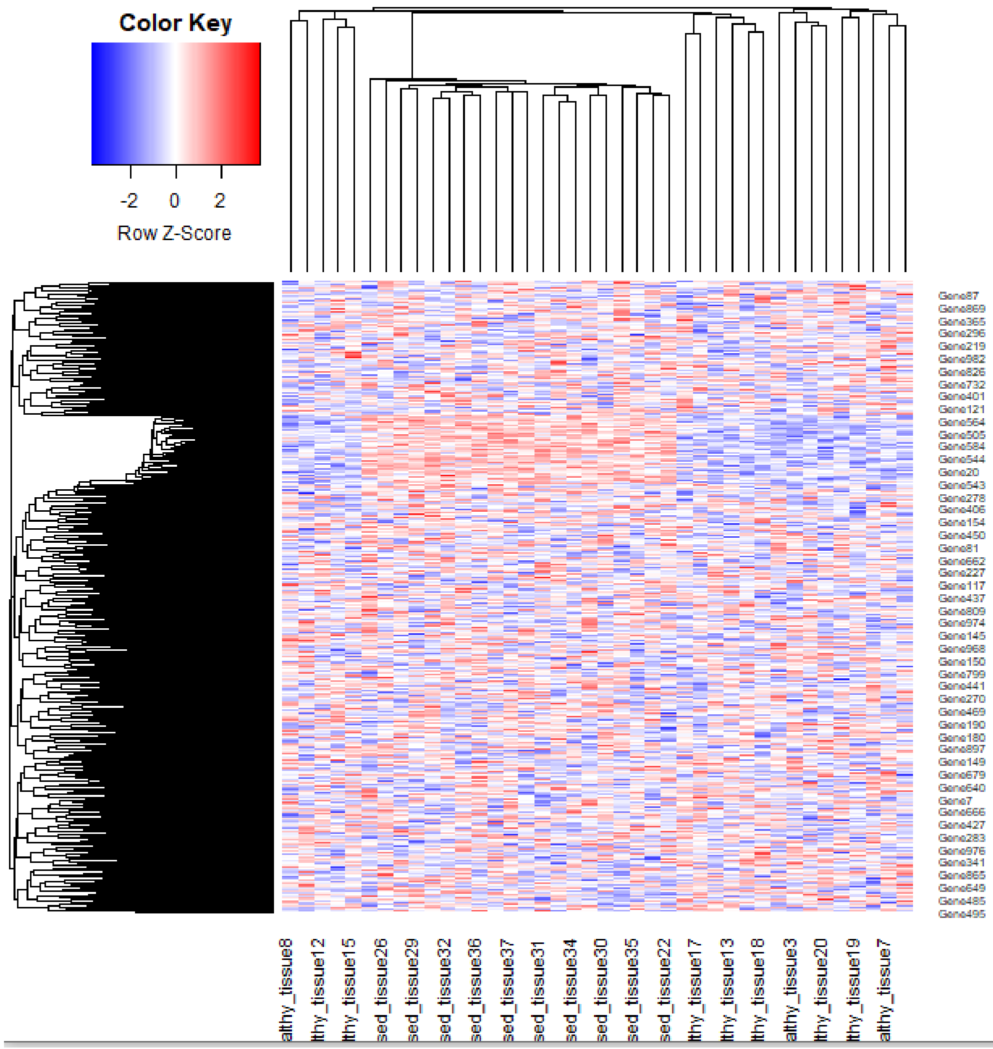


Figure 12: "Heat Map of hierarchial clustering on features and genes using average linkage"

The above plot shows hierarchial Clustering on genes as well as tissues with correlation as distance matrix and average linkage

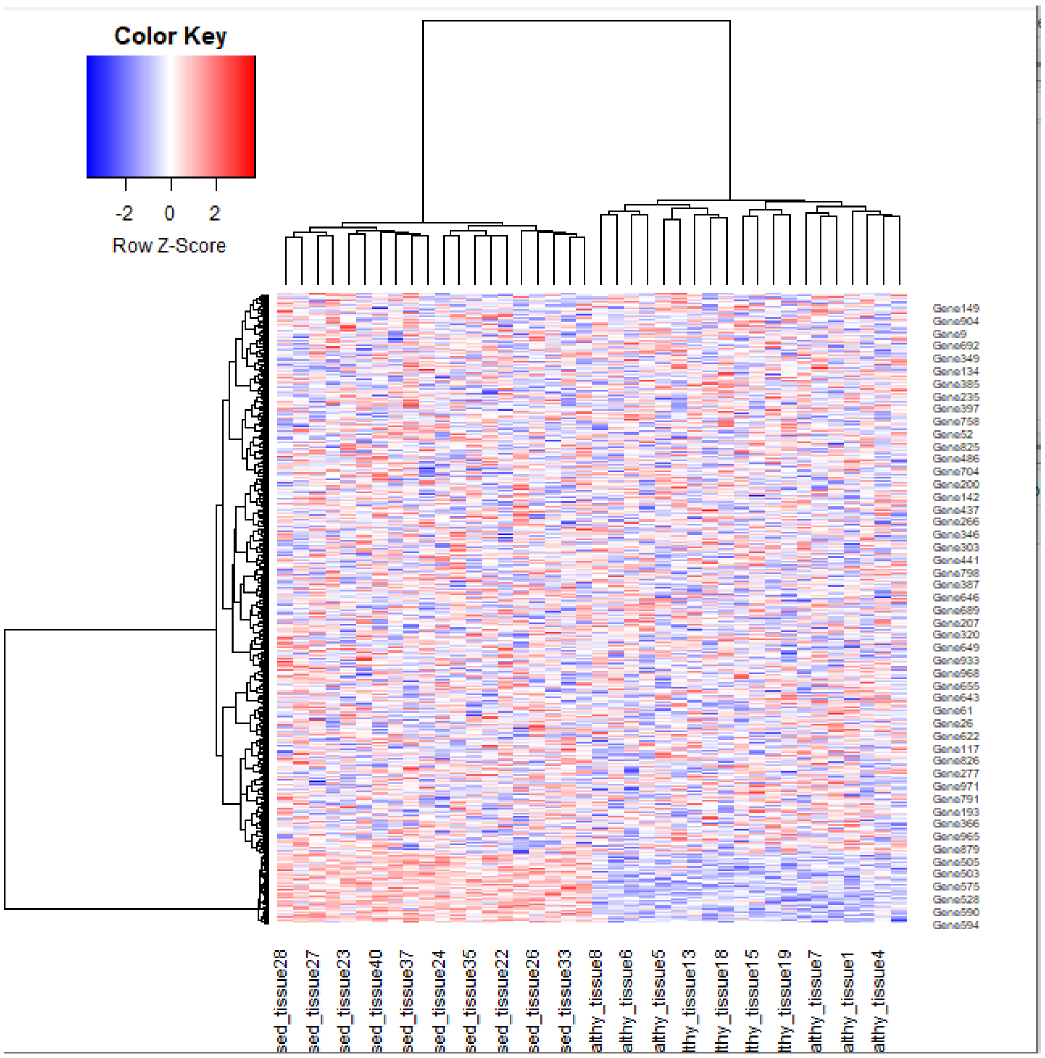


Figure 13: "Heat Map of hierarchial clustering on features and genes using ward.D linkage"

The above plot shows hierarchial Clustering on genes as well as tissues with correlation as distance matrix and ward.D linkage

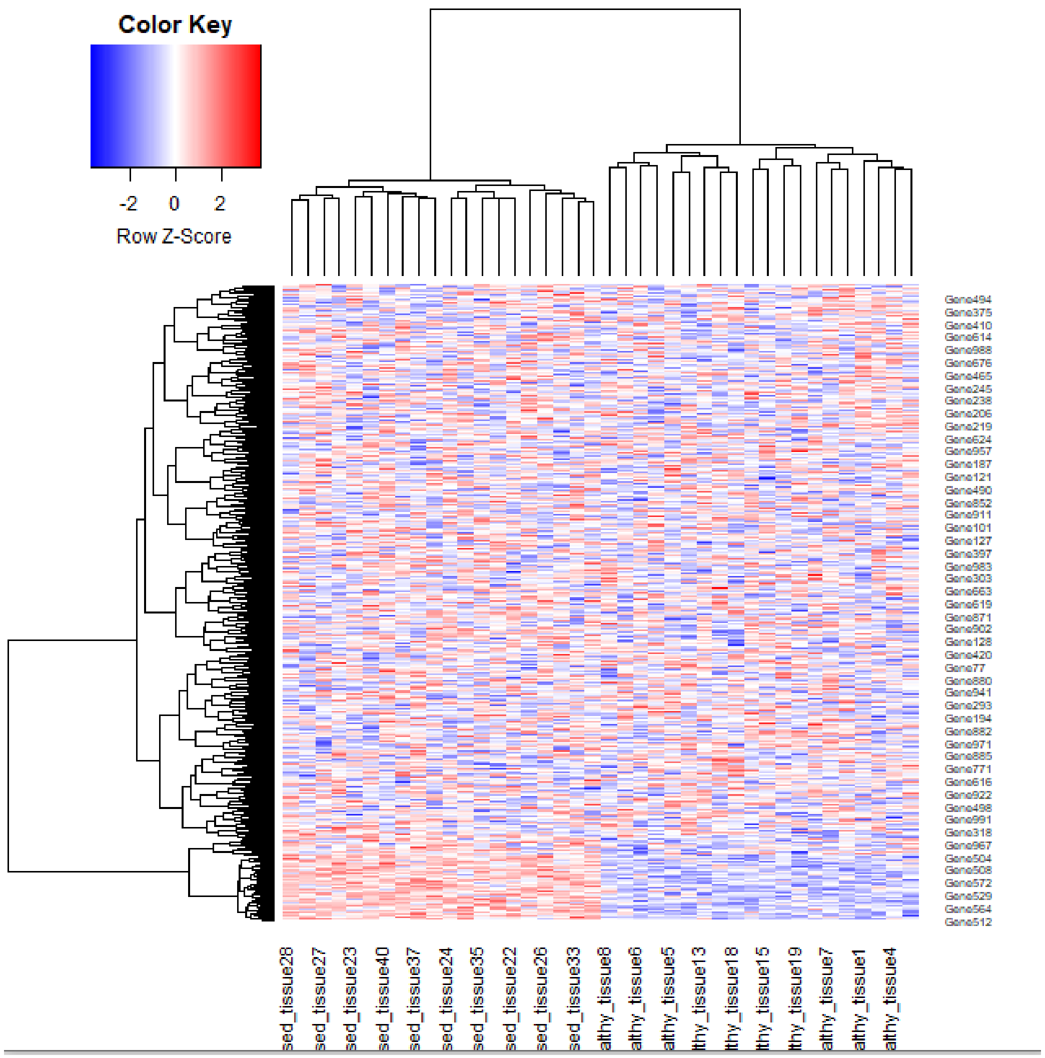


Figure 14: "Heat Map of hierarchial clustering on features and genes using ward.D2 linkage"

The above plot shows hierarchial Clustering on genes as well as tissues with correlation as distance matrix and ward.D2 linkage

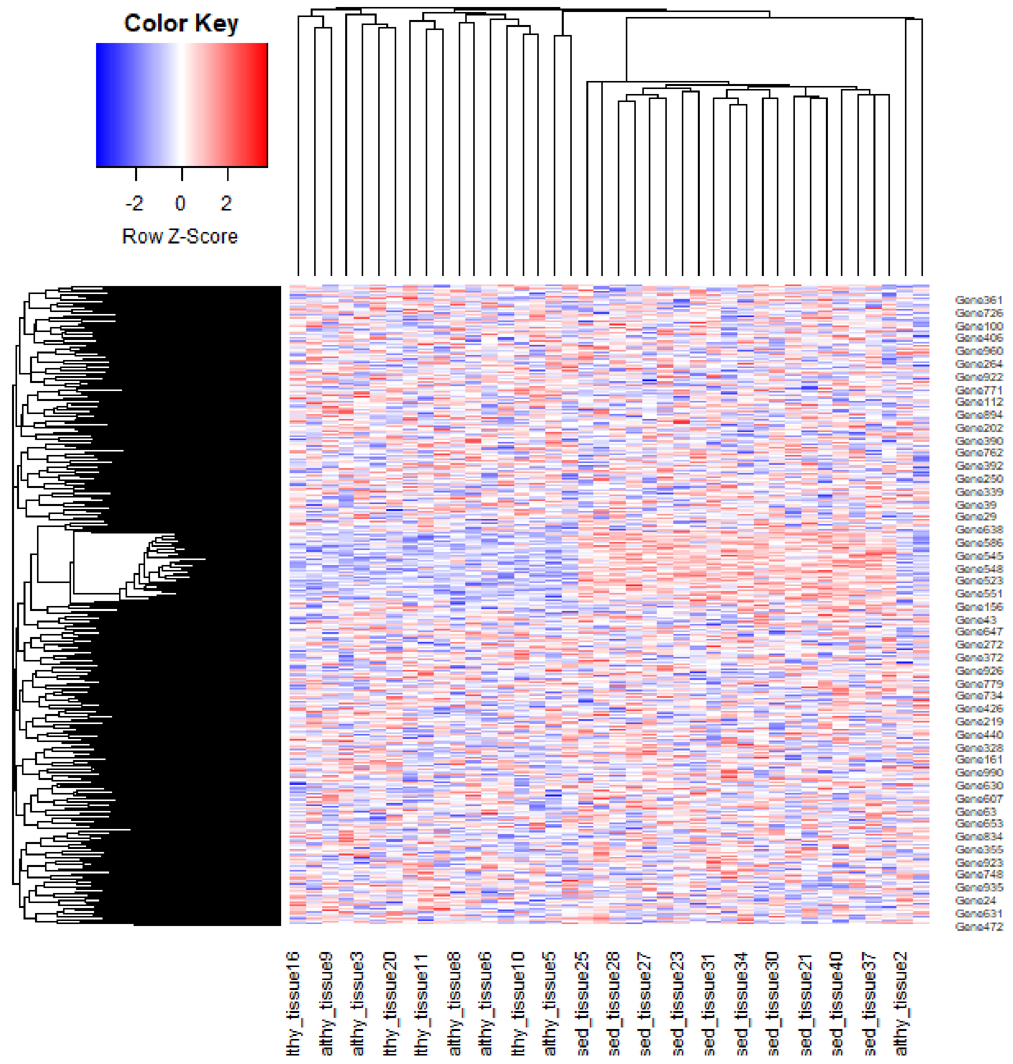


Figure 15: "Heat Map of hierarchial clustering on features and genes using mcquitty linkage"

The above plot shows hierarchial Clustering on genes as well as tissues with correlation as distance matrix and mcquitty linkage

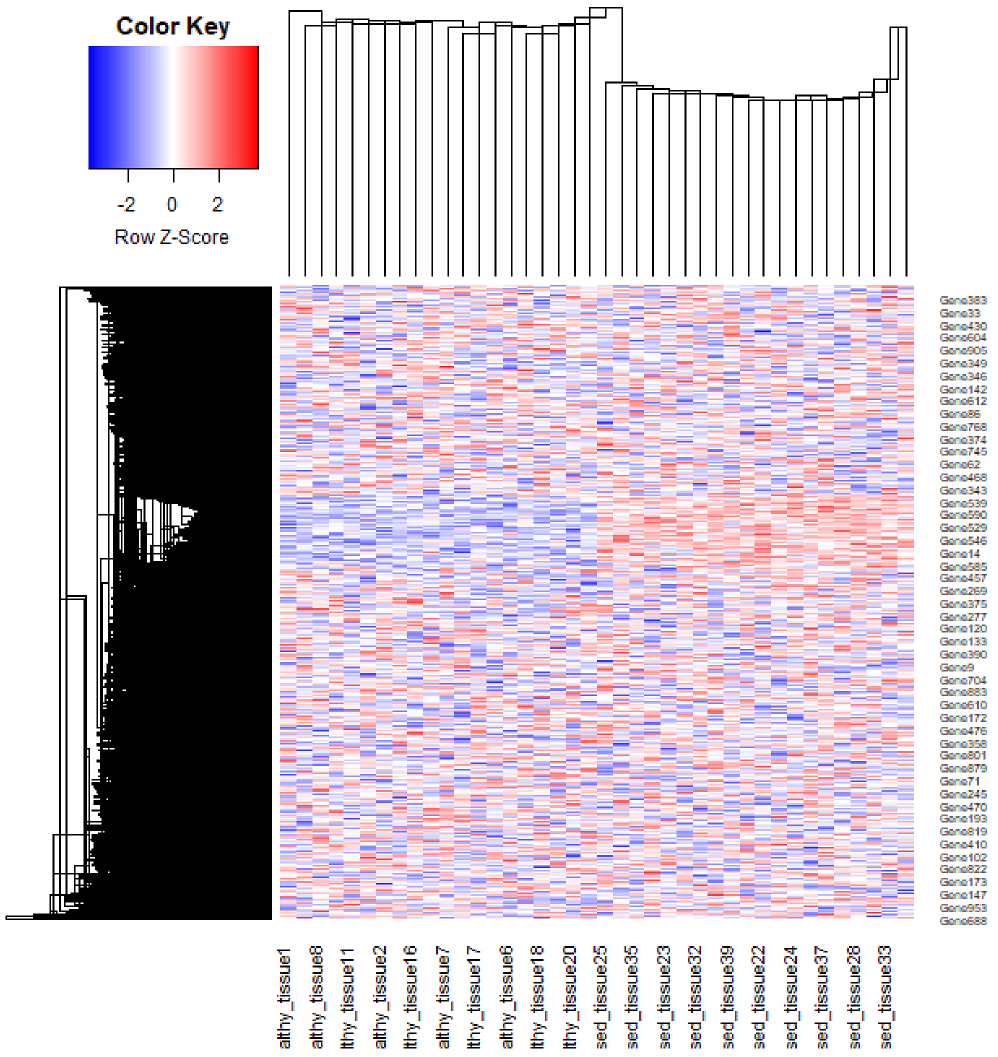


Figure 16: "Heat Map of hierarchial clustering on features and genes using median linkage"

The above plot shows hierarchial Clustering on genes as well as tissues with correlation as distance matrix and median linkage

The below plot shows hierarchial Clustering on genes as well as tissues with correlation as distance matrix and centroid linkage .

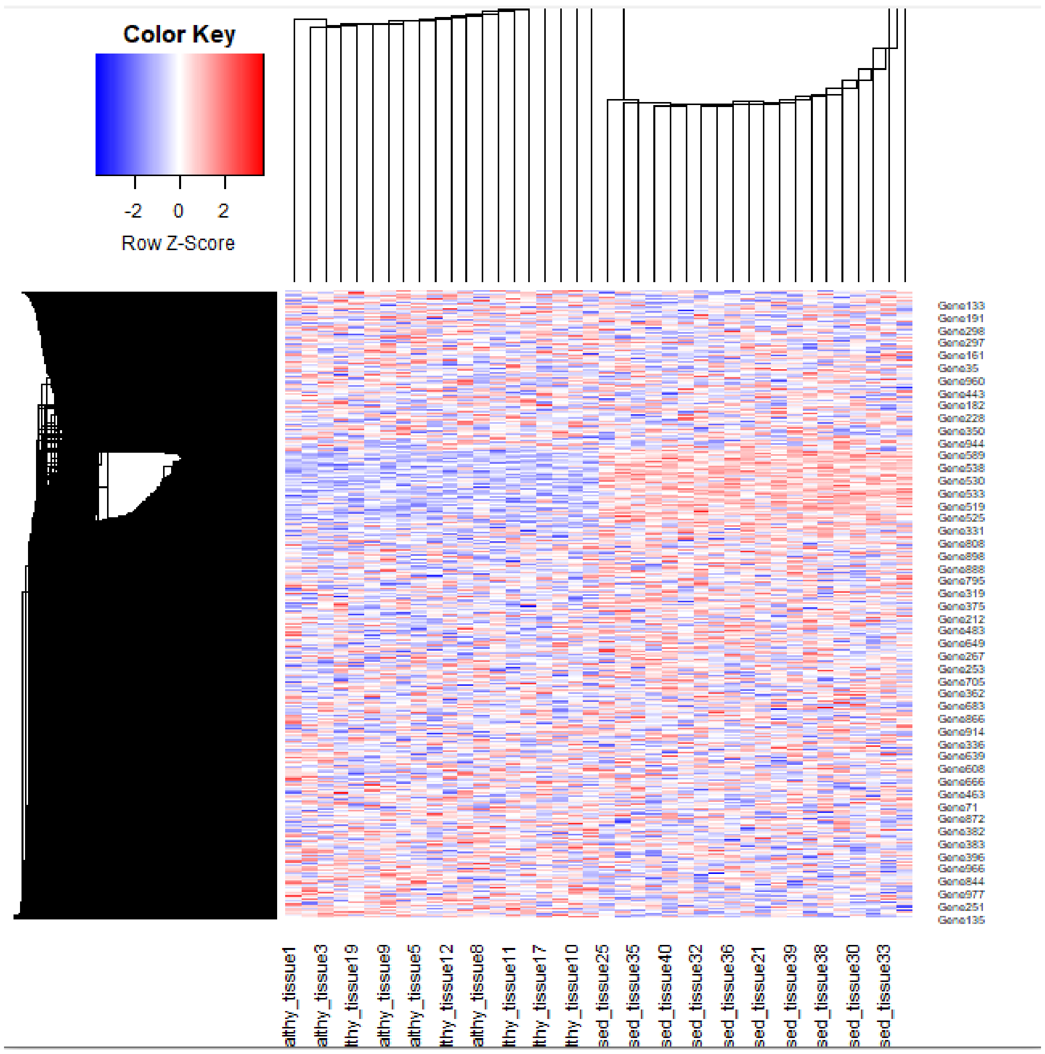


Figure 17: "Heat Map of hierarchial clustering on features and genes using centroid linkage"

We see that on varying the linkage method we get different hierarchial dendograms on genes as well as tissues.Also on looking further it seems ward linkage is able to provide us healthy dendograms for both genes and tissues which could be split into specific groups.Single linkage performs very badly in terms of clustering both genes as well as tissues. As stated above on comparing different linkages we see that if we consider

wardD or ward D2 linkage then the clustering on genes is able to separate the samples in two groups

3. **Part C**

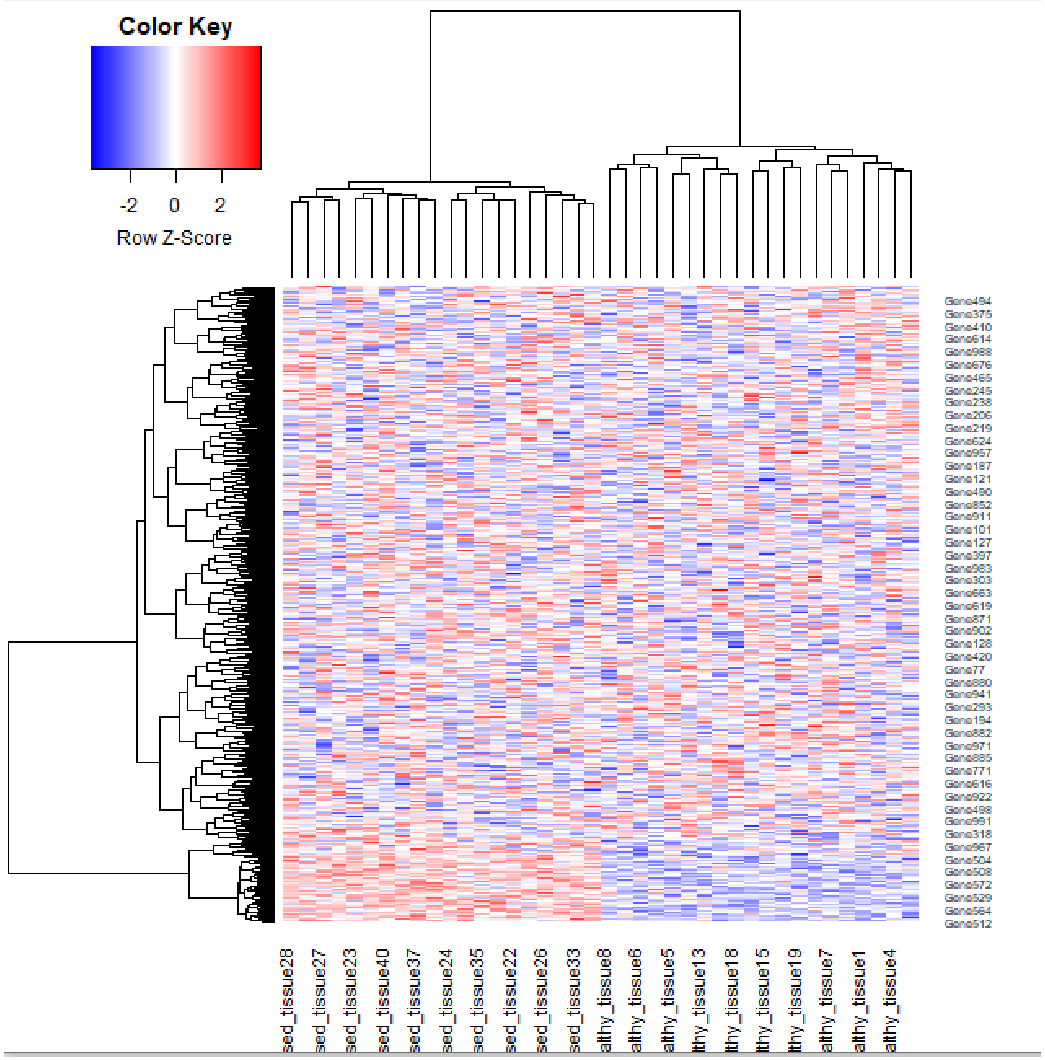


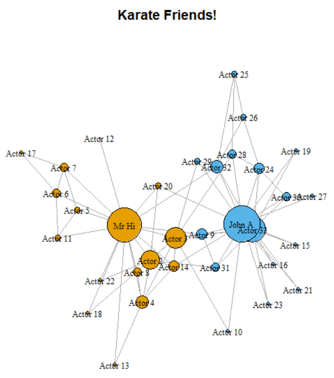
Figure 18: "Heat Map of hierarchial clustering on features and genes using ward.D linkage"

As we see from the above figure which involves hierarchial clustering on genes using ward.D linkage that genes 500-594

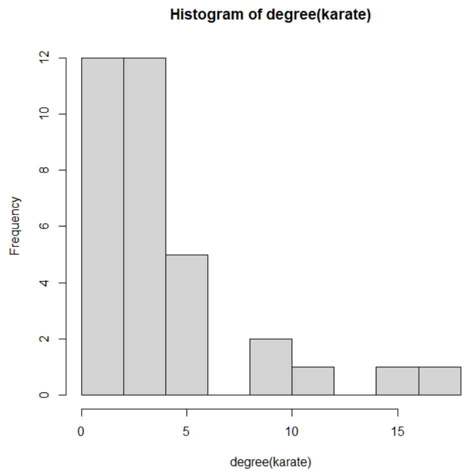
differs lot from all other genes , though their population is less but if we are to split this dendogram into two groups and this species of genes forms a totally different group.

4 Answer 4

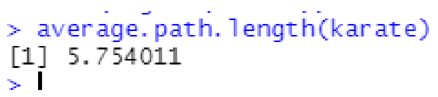
Visualizing the Karate network according to the degree distribution of the vertices where vertices represents the actors



"Karate Plot"

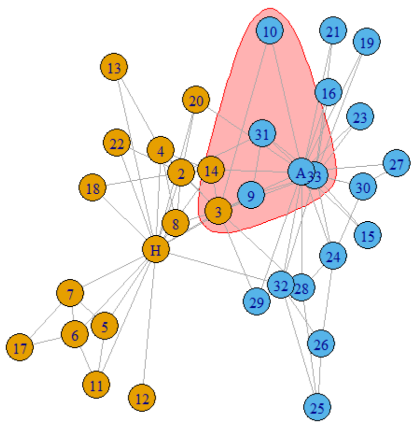


"Histogram plot of Degree Distribution"



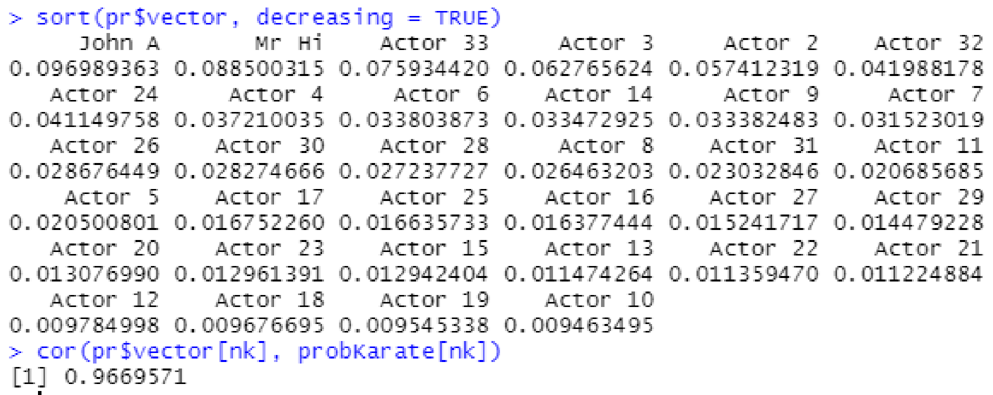
```
> average.path.length(karate)
[1] 5.754011
> 
```

"Average path length between the vertices"

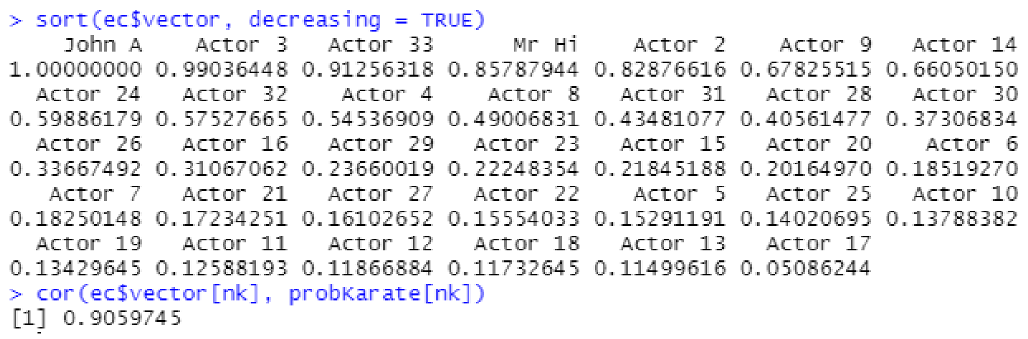


"Plotting the shortest path from a random actor which in thi case is**Actor 33**"

Who are the influencers (or hubs) in this social network? Let's use some network centrality measures to see. Are there any differences?Who gives the highest correlation with our drunken walk probabilities?

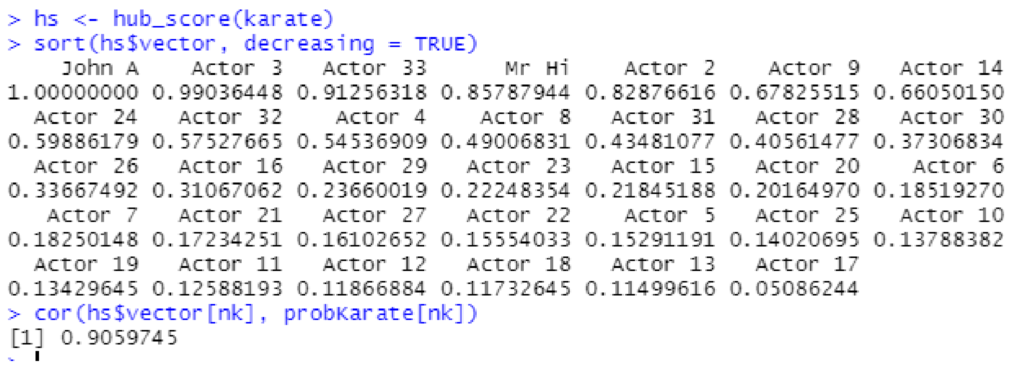


```
> sort(pr$vector, decreasing = TRUE)
     John A       Mr Hi    Actor 33     Actor 3     Actor 2    Actor 32 
0.096989363 0.088500315 0.075934420 0.062765624 0.057412319 0.041988178 
   Actor 24     Actor 4     Actor 6    Actor 14     Actor 9     Actor 7 
0.041149758 0.037210035 0.033803873 0.033472925 0.033382483 0.031523019 
   Actor 26    Actor 30    Actor 28     Actor 8    Actor 31    Actor 11 
0.028676449 0.028274666 0.027237727 0.026463203 0.023032846 0.020685685 
    Actor 5    Actor 17    Actor 25    Actor 16    Actor 27    Actor 29 
0.020500801 0.016752260 0.016635733 0.016377444 0.015241717 0.014479228 
   Actor 20    Actor 23    Actor 15    Actor 13    Actor 22    Actor 21 
0.013076990 0.012961391 0.012942404 0.011474264 0.011359470 0.011224884 
   Actor 12    Actor 18    Actor 19    Actor 10 
0.009784998 0.009676695 0.009545338 0.009463495 
> cor(pr$vector[nk], probKarate[nk])
[1] 0.9669571
```

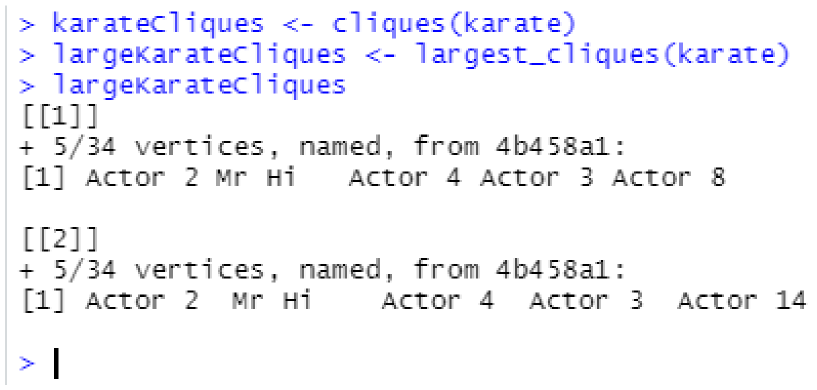


```
> sort(ec$vector, decreasing = TRUE)
    John A     Actor 3    Actor 33       Mr Hi     Actor 2     Actor 9    Actor 14 
1.00000000 0.99036448 0.91256318 0.85787944 0.82876616 0.67825515 0.66050150 
  Actor 24    Actor 32     Actor 4     Actor 8    Actor 31    Actor 28    Actor 30 
0.59886179 0.57527665 0.54536909 0.49006831 0.43481077 0.40561477 0.37306834 
  Actor 26    Actor 16    Actor 29    Actor 23    Actor 15    Actor 20     Actor 6 
0.33667492 0.31067062 0.23660019 0.22248354 0.21845188 0.20164970 0.18519270 
   Actor 7    Actor 21    Actor 27    Actor 22     Actor 5    Actor 25    Actor 10 
0.18250148 0.17234251 0.16102652 0.15554033 0.15291191 0.14020695 0.13788382 
  Actor 19    Actor 11    Actor 12    Actor 18    Actor 13    Actor 17 
0.13429645 0.12588193 0.11866884 0.11732645 0.11499616 0.05086244 
> cor(ec$vector[nk], probKarate[nk])
[1] 0.9059745
```

Figure 19: ***Eigenvector Centrality Eigenvector Centrality is used to see the nodes that connect to most connected nodes. From the graph we can conclude that the member with higher eigenvector centrality value are the members that connected with the most connected member of the karate club.***



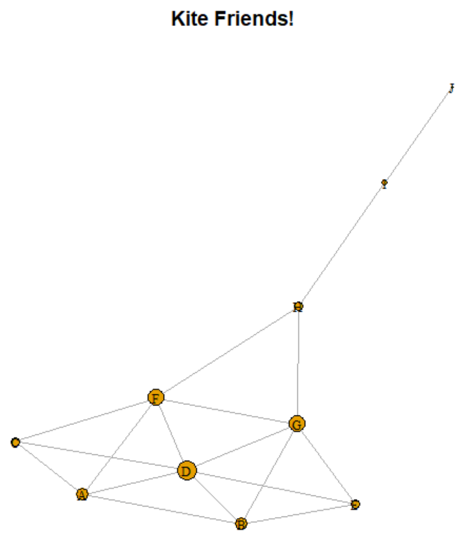
```
> hs <- hub_score(karate)
> sort(hs$vector, decreasing = TRUE)
    John A     Actor 3    Actor 33       Mr Hi     Actor 2     Actor 9    Actor 14 
1.00000000 0.99036448 0.91256318 0.85787944 0.82876616 0.67825515 0.66050150 
  Actor 24    Actor 32     Actor 4     Actor 8    Actor 31    Actor 28    Actor 30 
0.59886179 0.57527665 0.54536909 0.49006831 0.43481077 0.40561477 0.37306834 
  Actor 26    Actor 16    Actor 29    Actor 23    Actor 15    Actor 20     Actor 6 
0.33667492 0.31067062 0.23660019 0.22248354 0.21845188 0.20164970 0.18519270 
   Actor 7    Actor 21    Actor 27    Actor 22     Actor 5    Actor 25    Actor 10 
0.18250148 0.17234251 0.16102652 0.15554033 0.15291191 0.14020695 0.13788382 
  Actor 19    Actor 11    Actor 12    Actor 18    Actor 13    Actor 17 
0.13429645 0.12588193 0.11866884 0.11732645 0.11499616 0.05086244 
> cor(hs$vector[nk], probKarate[nk])
[1] 0.9059745
```



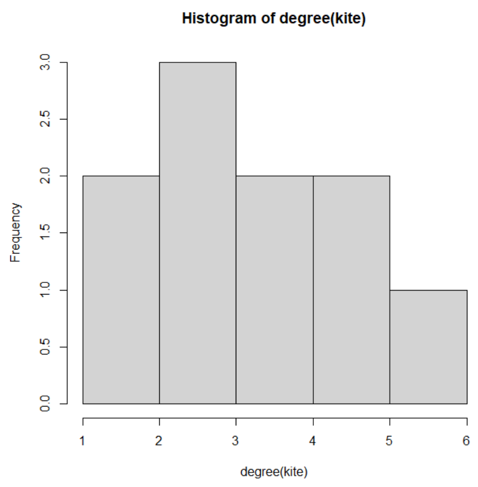
```
> karateCliques <- cliques(karate)
> largeKarateCliques <- largest_cliques(karate)
> largeKarateCliques
[[1]]
+ 5/34 vertices, named, from 4b458a1:
[1] Actor 2 Mr Hi   Actor 4 Actor 3 Actor 8

[[2]]
+ 5/34 vertices, named, from 4b458a1:
[1] Actor 2  Mr Hi    Actor 4  Actor 3  Actor 14

> 
```

Figure 20: "Who's the most connected groups to each other?"

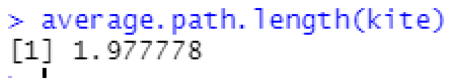
Visualizing the Kite network according to the degree distribution of the vertices where vertices represents the actors



"Kite Plot"

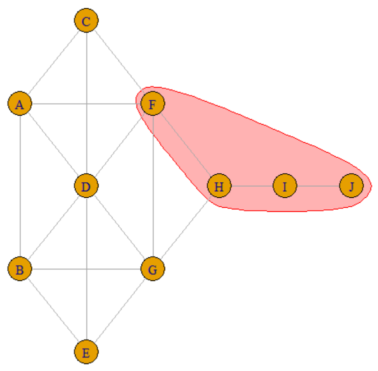


"Histogram plot of Degree Distribution"



```
> average.path.length(kite)
[1] 1.977778
```

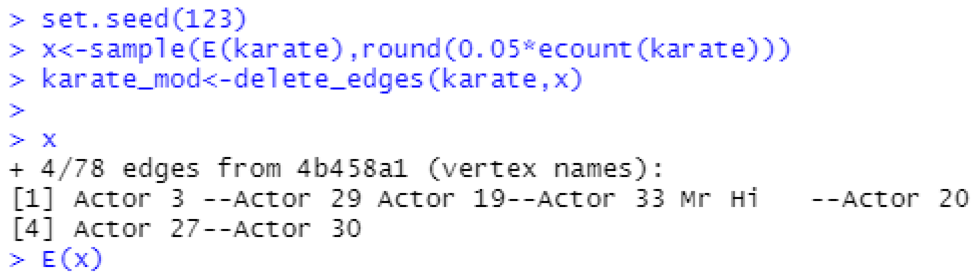
"Average path length between the vertices"



"Plotting the shortest path from a random node which in this case is Actor F"

1. Part A

   We have randomly removed 5 percent of the edges from the Karate Network and we are storing it in a variable x



```
> set.seed(123)
> x<-sample(E(karate),round(0.05*ecount(karate)))
> karate_mod<-delete_edges(karate,x)
>
> x
+ 4/78 edges from 4b458a1 (vertex names):
[1] Actor 3 --Actor 29 Actor 19--Actor 33 Mr Hi   --Actor 20
[4] Actor 27--Actor 30
> E(x)
```

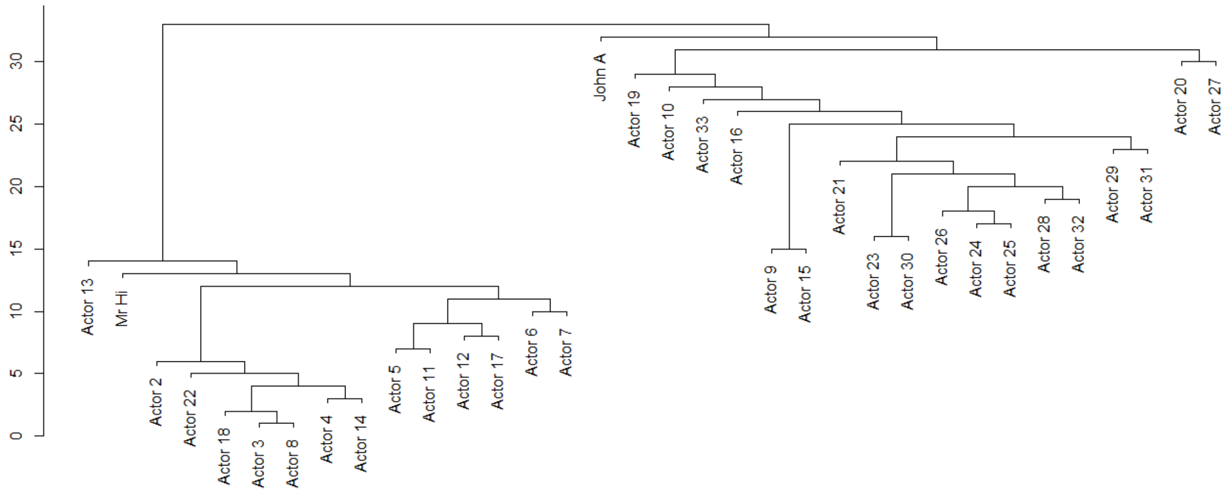
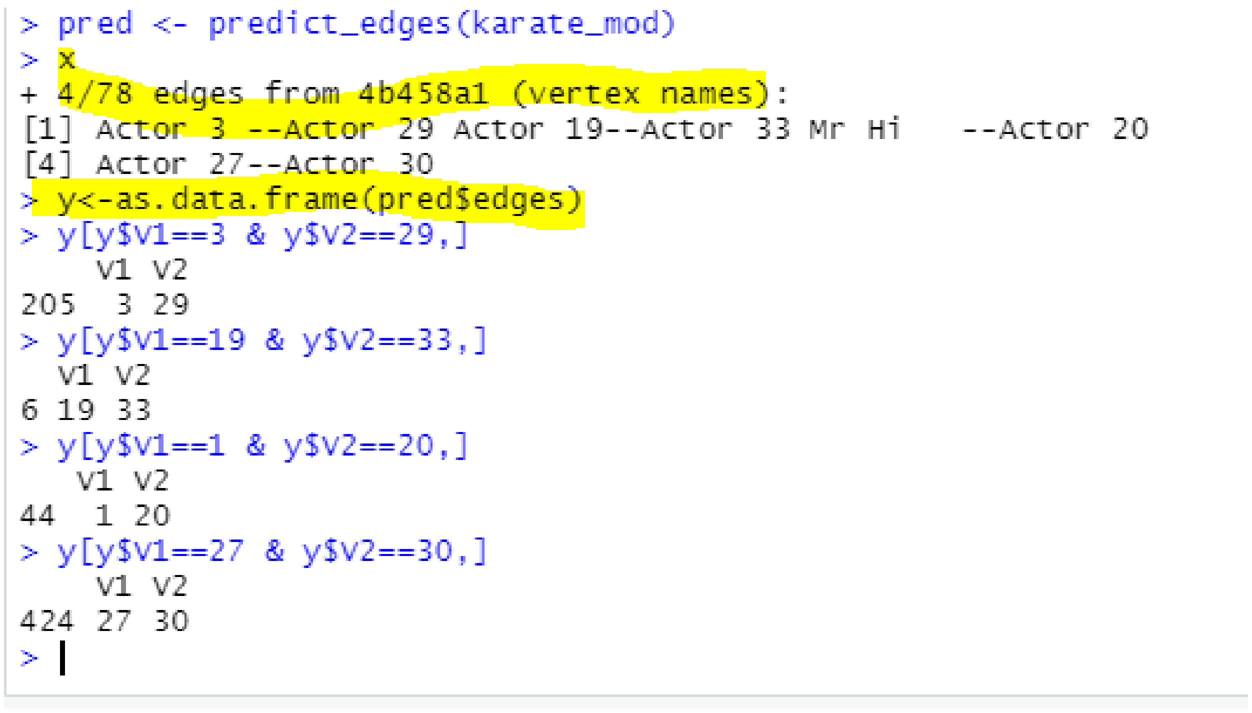


Figure 21: "Hierarchial Dendogram plot on Karate Network"

The dendogram reveals plot reveals clearly the actual graph structure that we visulaized from our earlier data that indeed there two separate groups or clubs in Karate Network

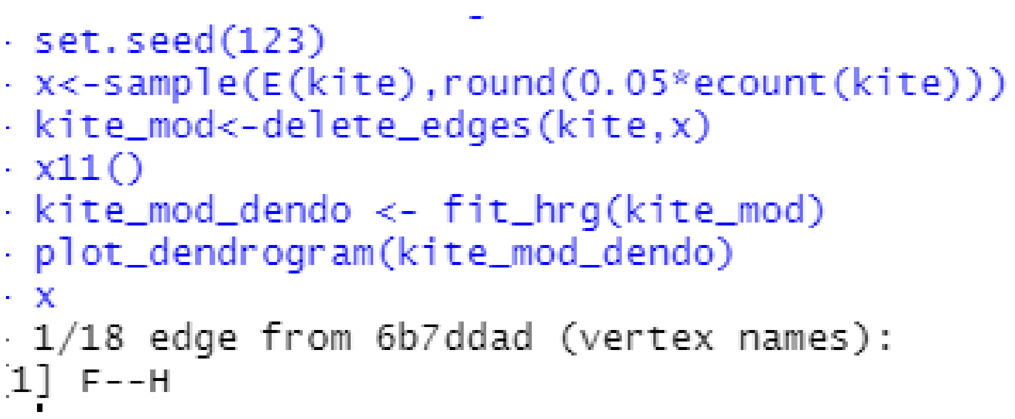
I have predicted the missing edges, I have mentioned earlier that my x variable contains randomly removed 5 percent of the edges from Karate Network.I have extracted the predicted edges from the variable pred and converted to a dataframe for my convinienece.Then I am matching the predicted edges with those present in the x variable(the missing ones).Here I am able to recover the missing edges.Below is the code snapshot for that.



```
> pred <- predict_edges(karate_mod)
> x
+ 4/78 edges from 4b458a1 (vertex names):
[1] Actor 3 --Actor 29 Actor 19--Actor 33 Mr Hi   --Actor 20
[4] Actor 27--Actor 30
> y<-as.data.frame(pred$edges)
> y[y$V1==3 & y$V2==29,]
    V1 V2
205  3 29
> y[y$V1==19 & y$V2==33,]
  V1 V2
6 19 33
> y[y$V1==1 & y$V2==20,]
   V1 V2
44  1 20
> y[y$V1==27 & y$V2==30,]
    V1 V2
424 27 30
> 
```

Figure 22: "A code snapshot with evidence that the randomly removed edges from Karate Network edges were recovered"

1. Part B ***We have randomly removed 5 percent of the edges from the Kite Network and we are storing it in a variable x***



```
set.seed(123)
x<-sample(E(kite),round(0.05*ecount(kite)))
kite_mod<-delete_edges(kite,x)
x11()
kite_mod_dendo <- fit_hrg(kite_mod)
plot_dendrogram(kite_mod_dendo)
x
1/18 edge from 6b7ddad (vertex names):
1] F--H
```

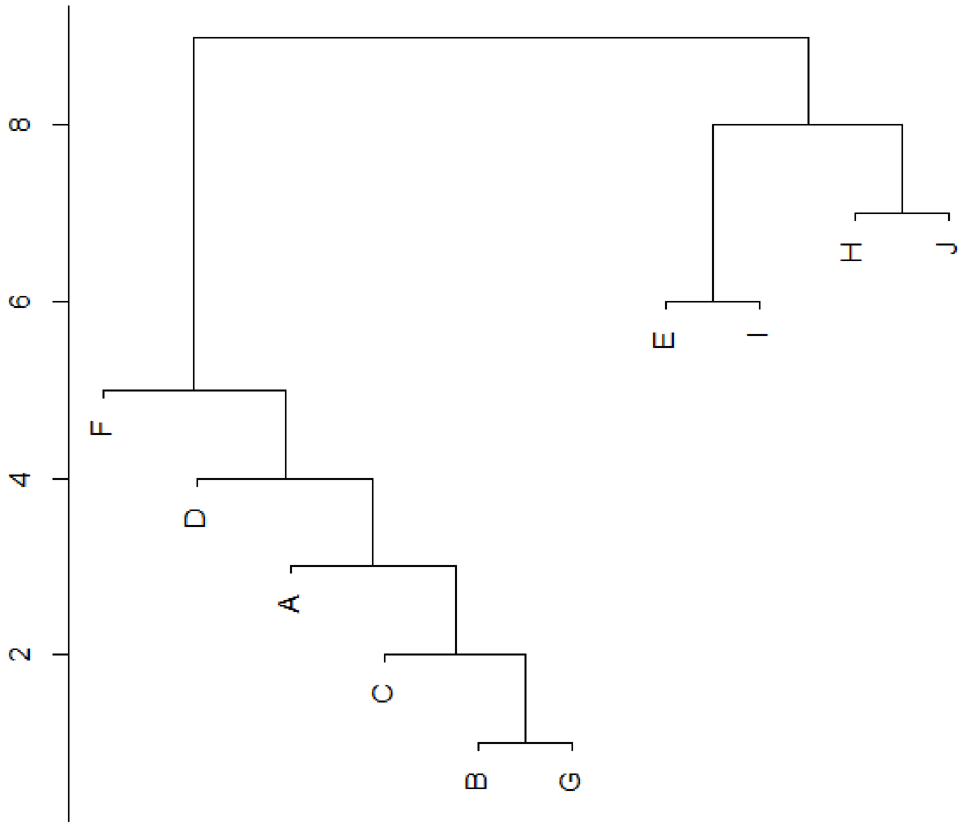
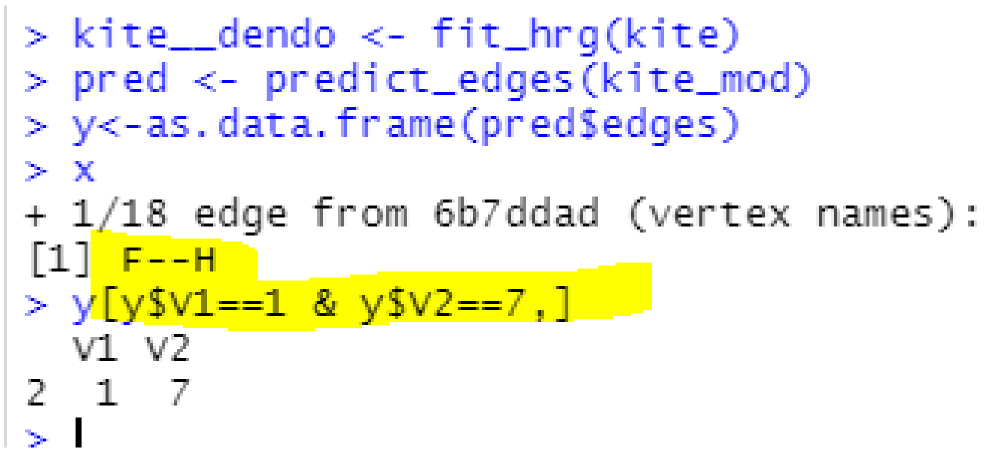


Figure 23: "Hierarchial Dendogram plot on Kite Network"

The dendogram reveals plot reveals clearly the actual graph structure that we visulaized from our earlier data that indeed there two separate groups of actors in Kite Network

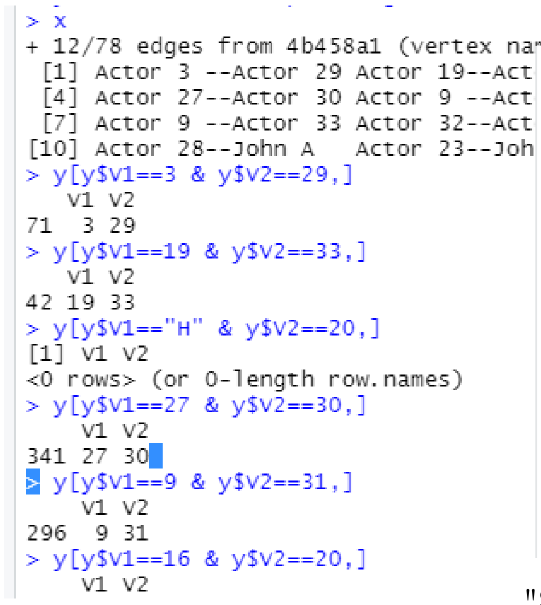
I have predicted the missing edges, I have mentioned earlier that my x variable contains randomly removed 5 percent of the edges from Kite Network.I have extracted the predicted edges from the variable pred and converted to a dataframe for my convinienece.Then I am matching the predicted edges with those present in the x variable(the missing ones).Here I am able to recover the missing edges.Below is the code snapshot for that.



```
> kite__dendo <- fit_hrg(kite)
> pred <- predict_edges(kite_mod)
> y<-as.data.frame(pred$edges)
> x
+ 1/18 edge from 6b7ddad (vertex names):
[1] F--H
> y[y$V1==1 & y$V2==7,]
  V1 V2
2  1  7
> 
```

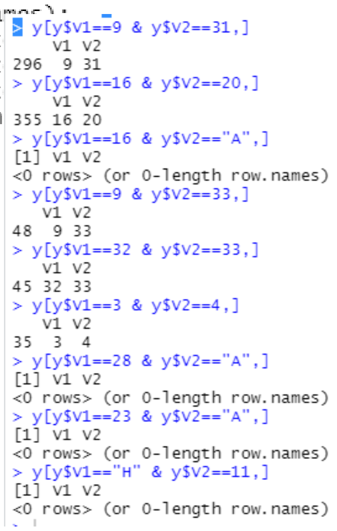
Figure 24: "A code snapshot with evidence that the randomly removed edges from Kite Network edges were recovered"

1. Part C ***Randomly removed 15 percent of the edges from karate network and testing on whether the removed edges are getting recovered after predicting all the significant edges that resulted from the prediction as shown in below code snapshot***



```
> x
+ 12/78 edges from 4b458a1 (vertex names):
 [1] Actor 3 --Actor 29 Actor 19--Act
 [4] Actor 27--Actor 30 Actor 9 --Act
 [7] Actor 9 --Actor 33 Actor 32--Act
[10] Actor 28--John A   Actor 23--Joh
> y[y$V1==3 & y$V2==29,]
   V1 V2
71  3 29
> y[y$V1==19 & y$V2==33,]
   V1 V2
42 19 33
> y[y$V1=="H" & y$V2==20,]
[1] V1 V2
<0 rows> (or 0-length row.names)
> y[y$V1==27 & y$V2==30,]
    V1 V2
341 27 30
> y[y$V1==9 & y$V2==31,]
    V1 V2
296  9 31
> y[y$V1==16 & y$V2==20,]
    V1 V2
```

"Snapshot of the code showing that all the removed edges except John A – Actor 23 were recovered after prediction"

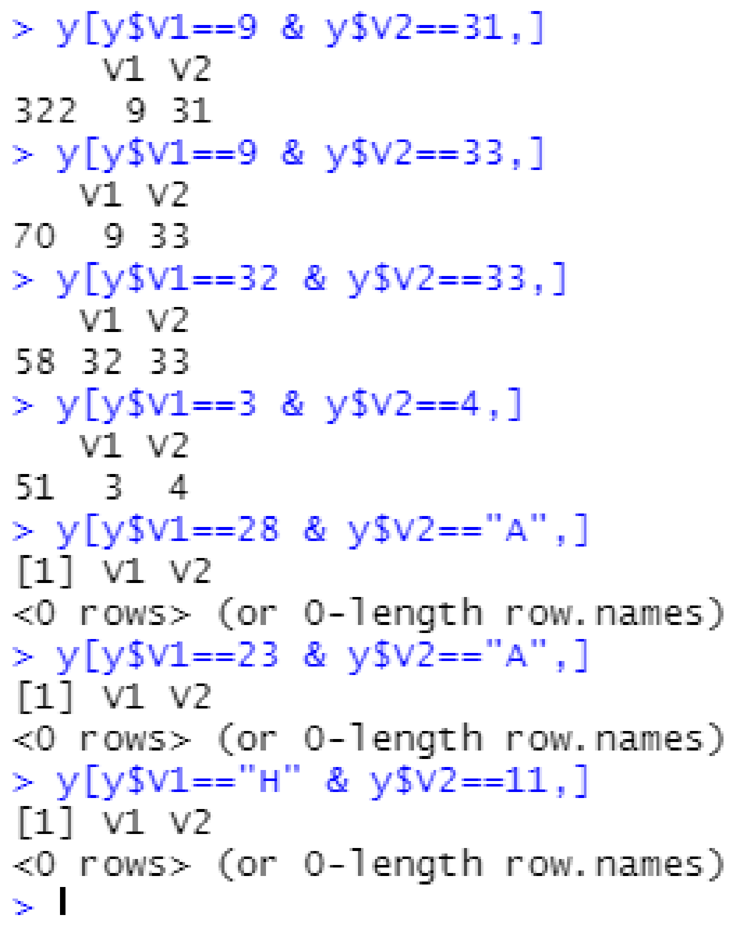


```
> y[y$V1==9 & y$V2==31,]
    V1 V2
296  9 31
> y[y$V1==16 & y$V2==20,]
    V1 V2
355 16 20
> y[y$V1==16 & y$V2=="A",]
[1] V1 V2
<0 rows> (or 0-length row.names)
> y[y$V1==9 & y$V2==33,]
   V1 V2
48  9 33
> y[y$V1==32 & y$V2==33,]
   V1 V2
45 32 33
> y[y$V1==3 & y$V2==4,]
   V1 V2
35  3  4
> y[y$V1==28 & y$V2=="A",]
[1] V1 V2
<0 rows> (or 0-length row.names)
> y[y$V1==23 & y$V2=="A",]
[1] V1 V2
<0 rows> (or 0-length row.names)
> y[y$V1=="H" & y$V2==11,]
[1] V1 V2
<0 rows> (or 0-length row.names)
```

"Snapshot of the code showing that all the removed edges except "Mr Hi – Actor 11" were recovered after prediction"

Except "Mr Hi – Actor 11" and "John A – Actor 23" edges I was able to recover the other edges after prediction.

Randomly removed 40 percent of the edges from karate network and testing on whether the removed edges are getting recovered after predicting all the significant edges that resulted from the prediction as shown in below code snapshot

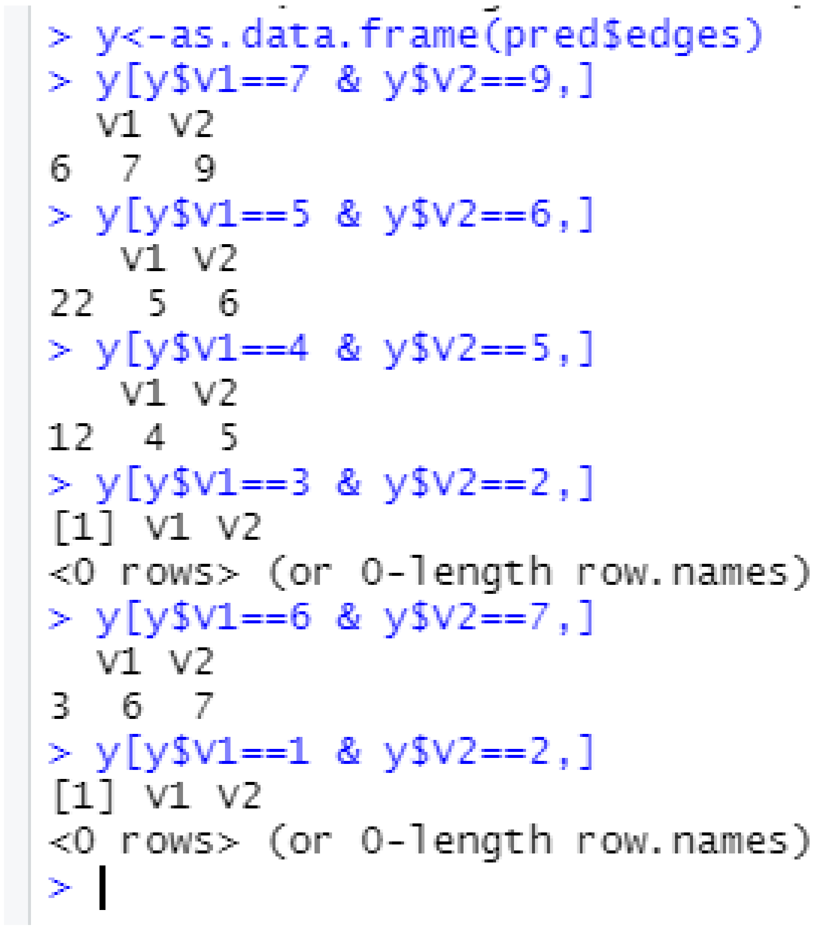


```
> y[y$V1==9 & y$V2==31,]
    V1 V2
322  9 31
> y[y$V1==9 & y$V2==33,]
   V1 V2
70  9 33
> y[y$V1==32 & y$V2==33,]
   V1 V2
58 32 33
> y[y$V1==3 & y$V2==4,]
   V1 V2
51  3  4
> y[y$V1==28 & y$V2=="A",]
[1] V1 V2
<0 rows> (or 0-length row.names)
> y[y$V1==23 & y$V2=="A",]
[1] V1 V2
<0 rows> (or 0-length row.names)
> y[y$V1=="H" & y$V2==11,]
[1] V1 V2
<0 rows> (or 0-length row.names)
> 
```

Figure 25: "Most of the edges were recovered except few such as "Mr Hi – Actor 11"

Except "Mr Hi – Actor 11" edge I was able to recover the other edges after prediction.

Randomly removed 15 percent of the edges from kite network and testing on whether the removed edges are getting recovered after predicting all the significant edges that resulted from the prediction as shown in below code snapshot

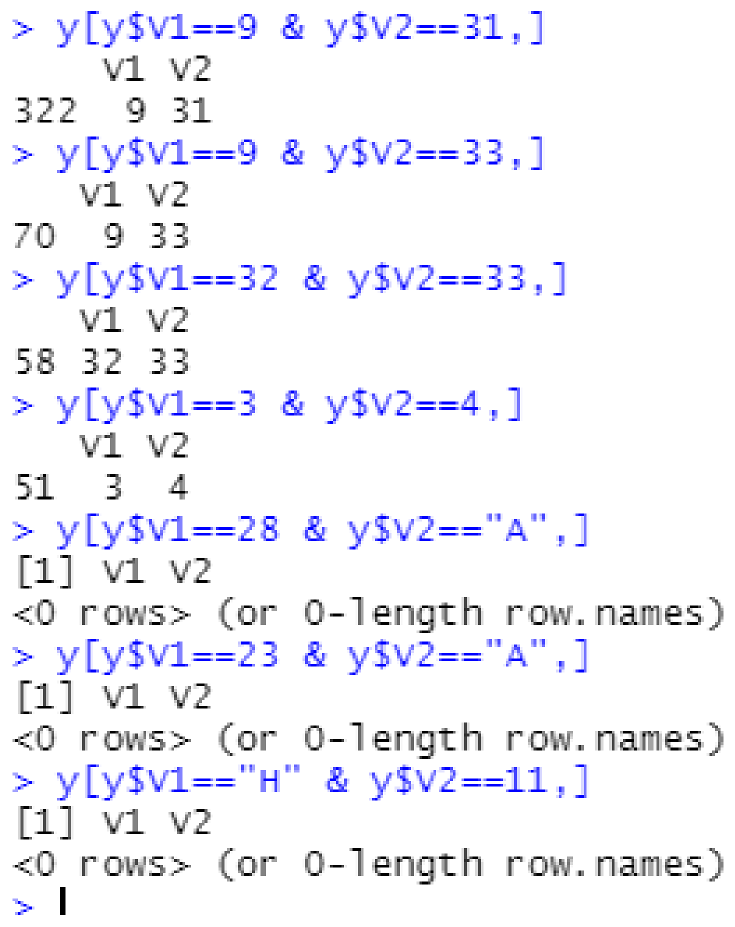


```
> y<-as.data.frame(pred$edges)
> y[y$V1==7 & y$V2==9,]
  V1 V2
6  7  9
> y[y$V1==5 & y$V2==6,]
   V1 V2
22  5  6
> y[y$V1==4 & y$V2==5,]
   V1 V2
12  4  5
> y[y$V1==3 & y$V2==2,]
[1] V1 V2
<0 rows> (or 0-length row.names)
> y[y$V1==6 & y$V2==7,]
  V1 V2
3  6  7
> y[y$V1==1 & y$V2==2,]
[1] V1 V2
<0 rows> (or 0-length row.names)
> 
```

Figure 26: "Again most of the edges except Actor 1—Actor2 were recovered"

Except "Actor 1—Actor2" edge I was able to recover the other edges after prediction.

Randomly removed 40 percent of the edges from kite network and testing on whether the removed edges are getting recovered after predicting all the significant edges that resulted from the prediction as shown in below code snapshot



```
> y[y$V1==9 & y$V2==31,]
    V1 V2
322  9 31
> y[y$V1==9 & y$V2==33,]
   V1 V2
70  9 33
> y[y$V1==32 & y$V2==33,]
   V1 V2
58 32 33
> y[y$V1==3 & y$V2==4,]
   V1 V2
51  3  4
> y[y$V1==28 & y$V2=="A",]
[1] V1 V2
<0 rows> (or 0-length row.names)
> y[y$V1==23 & y$V2=="A",]
[1] V1 V2
<0 rows> (or 0-length row.names)
> y[y$V1=="H" & y$V2==11,]
[1] V1 V2
<0 rows> (or 0-length row.names)
> 
```

Figure 27: "Most of the edges were recovered except few such as "Mr Hi – Actor 11"

Mostly all the edges were recovered for Karate and Kite networks except few.