CDA 532 Homework 2

UB (University at Buffalo) ID: 50413349

UB (University at Buffalo) Name: SHREERAM G S

Q1)

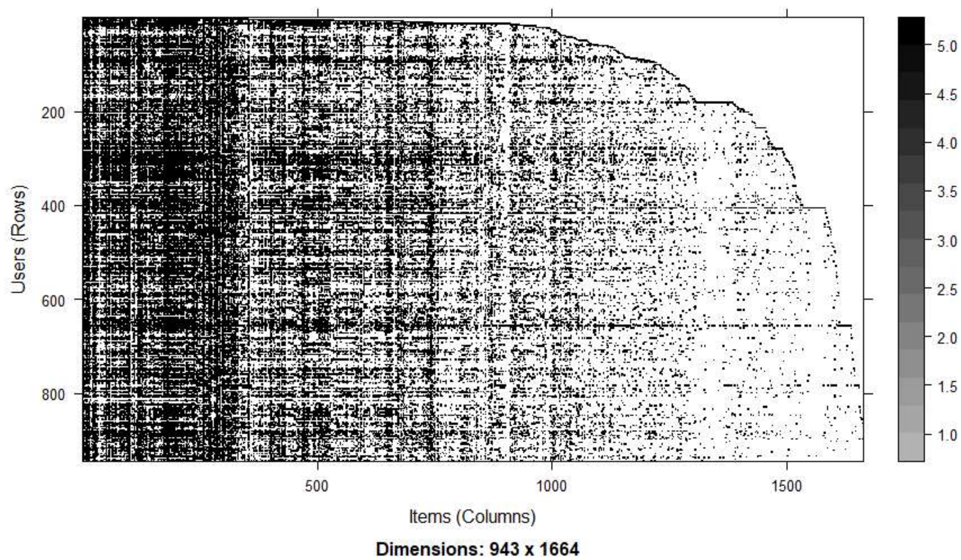
This problem statement requires us to use the recommenderlab package. We use **install.packages("recommenderlab")** to install the package before performing further operations on the residing data set from the library. Next, we call the libraries below and use the data() function to access the MovieLense data set. The movie lens data set is a real rating matrix object, and this has 3 different data. MovieLense's real rating matrix, a user data set, and meta data set. We use the bellow operations to view all the different data, but we are focused on the real rating matrix which gives user-item relationship.

```
library("devtools")
library("dplyrL")
library("recommenderlab")
data("MovieLense")
dim(MovieLense)
#visualizing the movie lens data matrix
image(MovieLense)

#head of first 5 movies and their ratings
head(as(MovieLense[1,], "list")[[1]])
#lets view the meta data
head(MovieLenseMeta)
#viewing the user data, this is a part of the movieLens data set
head(MovieLenseUser)
#store real rating matrix to a variable
mlens <- MovieLense

head(mlens_df)
#get the row and column names
dimnames(mlens)
 # number of ratings per user
rowCounts(mlens)
## number of ratings per item
colCounts(mlens)
# average item rating
colMeans(mlens)
#total number of ratings
nratings(mlens)
# user-item combinations with ratings
hasRating(mlens)
```

Below we see the image of the matrix.



Viewing the first 5 movie rating by user 1 for its respective movies in a list format by using the head() function.

```
> #head of first 5 movies and their ratings
> head(as(MovieLense[1,], "list")[[1]])
                                   Toy Story (1995)                              GoldenEye (1995)
                                                  5                                             3
                                  Four Rooms (1995)                            Get Shorty (1995)
                                                  4                                             3
                                     Copycat (1995) Shanghai Triad (Yao a yao yao dao waipo qiao) (1995)
                                                  3                                             5
```

Head() of metadata

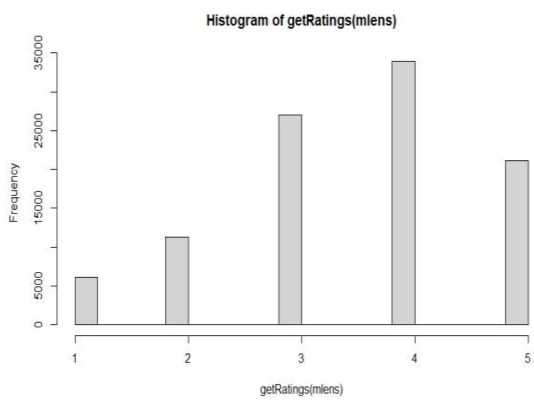
```
> #lets view the meta data
> head(MovieLenseMeta)
                                         title year                                                     url
1                             Toy Story (1995) 1995         http://us.imdb.com/M/title-exact?Toy%20Story%20(1995)
2                             GoldenEye (1995) 1995          http://us.imdb.com/M/title-exact?GoldenEye%20(1995)
3                            Four Rooms (1995) 1995        http://us.imdb.com/M/title-exact?Four%20Rooms%20(1995)
4                            Get Shorty (1995) 1995        http://us.imdb.com/M/title-exact?Get%20Shorty%20(1995)
5                               Copycat (1995) 1995            http://us.imdb.com/M/title-exact?Copycat%20(1995)
6 Shanghai Triad (Yao a yao yao dao waipo qiao) (1995) 1995 http://us.imdb.com/Title?Yao+a+yao+yao+dao+waipo+qiao+(1995)
  unknown Action Adventure Animation Children's Comedy Crime Documentary Drama Fantasy Film-Noir Horror Musical Mystery
1       0      0         0         1          1      1     0           0     0       0         0      0       0       0
2       0      1         1         0          0      0     0           0     0       0         0      0       0       0
3       0      0         0         0          0      0     0           0     0       0         0      0       0       0
4       0      1         0         0          0      1     0           0     1       0         0      0       0       0
5       0      0         0         0          0      0     1           0     1       0         0      0       0       0
6       0      0         0         0          0      0     0           0     1       0         0      0       0       0
  Romance Sci-Fi Thriller War Western
1       0      0        0   0       0
2       0      0        1   0       0
3       0      0        1   0       0
4       0      0        0   0       0
5       0      0        1   0       0
6       0      0        0   0       0
```

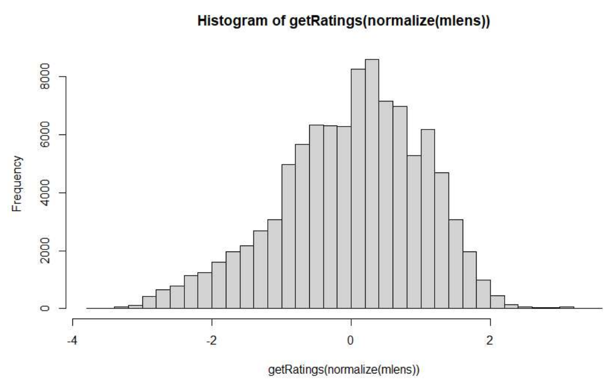
Head of the user data in the MovieLense Data package.

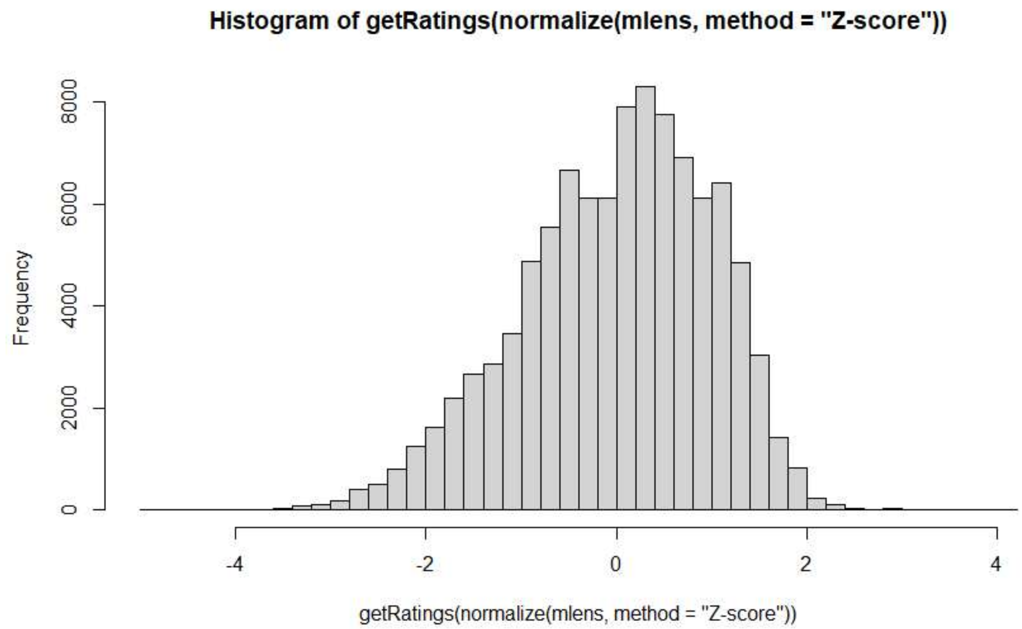
```
> #viewing the user data, this is a part of the movieLens data set
> head(MovieLenseUser)
  id age sex occupation zipcode
1  1  24   M technician   85711
2  2  53   F      other   94043
3  3  23   M     writer   32067
4  4  24   M technician   43537
5  5  33   F      other   15213
6  6  42   M  executive   98101
```

We use rowcounts, colcounts and other functions to see the data set in a list format and then use the information obtained from it to realize further operations. We now move forward to visualize the dataset before we start figuring out which recommender is right based on the principles mentioned in the problem statement.

```
> #Histogram of rating distribution
> hist(getRatings(mlens))
> #Histogram of normalized ratings using row centering
> mlen_norm<- hist(getRatings(normalize(mlens)),breaks = 50)
> #Histogram of Z-score normalization
> hist(getRatings(normalize(mlens, method="Z-score")),breaks = 50)
```







Next, we create a recommender registry using the code below, and this registry has all the various techniques such as ALS, UBCF, SVD etc. We choose the appropriate method based on the problem statement. But before we do that, we shall also compare the various techniques and hence determine the most appropriate method for its respective dataset.

```
> #Creating a recommender
> R_sys <- recommenderRegistry$get_entries(dataType = "realRatingMatrix")
> R_sys
$HYBRID_realRatingMatrix
Recommender method: HYBRID for realRatingMatrix
Description: Hybrid recommender that aggegates several recommendation strategies using weighted averages.
Reference: NA
Parameters:
  recommenders weights
1         NULL    NULL

$ALS_realRatingMatrix
Recommender method: ALS for realRatingMatrix
Description: Recommender for explicit ratings based on latent factors, calculated by alternating least squares algorithm.
Reference: Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, Rong Pan (2008). Large-Scale Parallel Collaborative Filtering fo
r the Netflix Prize, 4th Int'l Conf. Algorithmic Aspects in Information and Management, LNCS 5034.
Parameters:
  normalize lambda n_factors n_iterations min_item_nr seed
1      NULL    0.1        10           10           1 NULL

$ALS_implicit_realRatingMatrix
Recommender method: ALS_implicit for realRatingMatrix
Description: Recommender for implicit data based on latent factors, calculated by alternating least squares algorithm.
Reference: Yifan Hu, Yehuda Koren, Chris Volinsky (2008). Collaborative Filtering for Implicit Feedback Datasets, ICDM '08 Pr
```

Next, we must choose an evaluation scheme, and here we use "*split*" method, by setting the best rating score to 5.

```
> #creating "all-but-5" evaluation scheme
> scheme <- evaluationScheme(mlens, method="split", train = .9, k=1, given=-5, goodRating=5)
> scheme
Evaluation scheme using all-but-5 items
Method: 'split' with 1 run(s).
Training set proportion: 0.900
Good ratings: >=5.000000
Data set: 943 x 1664 rating matrix of class 'realRatingMatrix' with 99392 ratings.
```

Now we use user based collaborative filtering and item based collaborative filtering to determine predictors which are further compared with one another to view the best recommender system.

```
#UBCF recommender and predictor
r1 <- Recommender(getData(scheme, "train"), "UBCF")
p1 <- predict(r1, getData(scheme, "known"), type="ratings")

#IBCF recommender and predictor
r2 <- Recommender(getData(scheme, "train"), "IBCF")
p2 <- predict(r2, getData(scheme, "known"), type="ratings")
```

```
> error <- rbind(
+   UBCF = calcPredictionAccuracy(p1, getData(scheme, "unknown")),
+   IBCF = calcPredictionAccuracy(p2, getData(scheme, "unknown"))
+   )
> error
         RMSE      MSE      MAE
UBCF 1.402553 1.967155 1.128884
IBCF 1.323743 1.752296 1.031111
>
```

We see that the RMSE value in the IBCF is lower than UBCF. But according to the following problem statement principles, we use the UBCF model.

Now we predict the top 10 recommendations using the UBCF recommender of random 3 users.

```
> Top_Pred <- predict(r1,sample(getData(scheme, "known"),3));
> (as(Top_Pred,"list"))
$`62`
 [1] "Boot, Das (1981)"                "Big Lebowski, The (1998)"       "Ridicule (1996)"
 [4] "187 (1997)"                      "Fire Down Below (1997)"         "Close Shave, A (1995)"
 [7] "Nikita (La Femme Nikita) (1990)" "Kolya (1996)"                   "Breaking the Waves (1996)"
[10] "Amistad (1997)"

$`670`
 [1] "Excess Baggage (1997)"                       "Close Shave, A (1995)"
 [3] "Midnight in the Garden of Good and Evil (1997)" "Anna Karenina (1997)"
 [5] "Back to the Future (1985)"                   "101 Dalmatians (1996)"
 [7] "Matilda (1996)"                              "Big Lebowski, The (1998)"
 [9] "Fallen (1998)"                               "Primal Fear (1996)"

$`491`
 [1] "Clerks (1994)"                "Heavy Metal (1981)"          "Fish Called Wanda, A (1988)"
 [4] "Grand Day Out, A (1992)"      "Fear of a Black Hat (1993)"  "Serial Mom (1994)"
 [7] "Brady Bunch Movie, The (1995)" "Heathers (1989)"            "Forbidden Planet (1956)"
[10] "Andre (1994)"
```

.

Q2)

a)

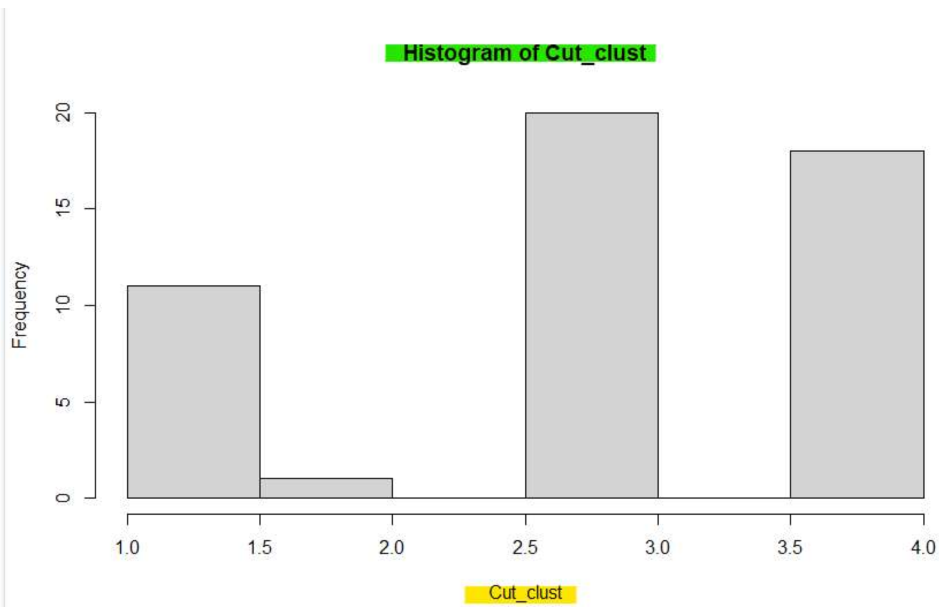
For this problem statement we shall import ISLR and kohonen packages, where we use the kohonen library to utilize the state data set. Next, we shall perform data exploration techniques to vie the dimensionality and obtain basic statistical inference required to perform further knowledge discovery operations. As we see below, the data set is composed of 50rows with 8 variables, and they are population, Income, Illiteracy, life expectancy, Murder, Highschool graduates, Frost and finally Area. These items are related to different states in the USA. We use the summary() function to generate statistical data on all the variables present in our data frame.

In the first sub part we are required to cluster the data using the hierarchical clustering technique.

```
> library(kohonen)
> require(ISLR)
> require(kohonen)
> data(state)
>
> #a) Hierarchical clustering=>
>
> #making a dataframe from the matrix=> state.x77
> df<- data.frame(state.x77)
>
> #view statistics of df
> dim(df)
[1] 50  8
> head(df)
           Population Income Illiteracy Life.Exp Murder HS.Grad Frost   Area
Alabama          3615   3624        2.1    69.05   15.1    41.3    20  50708
Alaska            365   6315        1.5    69.31   11.3    66.7   152 566432
Arizona          2212   4530        1.8    70.55    7.8    58.1    15 113417
Arkansas         2110   3378        1.9    70.66   10.1    39.9    65  51945
California      21198   5114        1.1    71.71   10.3    62.6    20 156361
Colorado         2541   4884        0.7    72.06    6.8    63.9   166 103766
> summary(df)
   Population        Income       Illiteracy       Life.Exp         Murder          HS.Grad          Frost
 Min.   :  365   Min.   :3098   Min.   :0.500   Min.   :67.96   Min.   : 1.400   Min.   :37.80   Min.   :  0.00
 1st Qu.: 1080   1st Qu.:3993   1st Qu.:0.625   1st Qu.:70.12   1st Qu.: 4.350   1st Qu.:48.05   1st Qu.: 66.25
 Median : 2838   Median :4519   Median :0.950   Median :70.67   Median : 6.850   Median :53.25   Median :114.50
 Mean   : 4246   Mean   :4436   Mean   :1.170   Mean   :70.88   Mean   : 7.378   Mean   :53.11   Mean   :104.46
 3rd Qu.: 4968   3rd Qu.:4814   3rd Qu.:1.575   3rd Qu.:71.89   3rd Qu.:10.675   3rd Qu.:59.15   3rd Qu.:139.75
 Max.   :21198   Max.   :6315   Max.   :2.800   Max.   :73.60   Max.   :15.100   Max.   :67.30   Max.   :188.00
      Area
 Min.   :  1049
 1st Qu.: 36985
 Median : 54277
 Mean   : 70736
 3rd Qu.: 81163
 Max.   :566432
>
```

Next, we shall scale our data frame and store the resultant in a new data frame variable named scaled_df. This variable is later used to calculate the Euclidean distance and hence the hierarchical cluster using the function hclust(). We later also use the cutree() function to show a simpler and easier to comprehend resultant. We later plot the cut tree into a histogram.

```
> scaled_df<- scale(df)
> #distance for hierarchical clustering
> clust_d <- dist(scaled_df, method = "euclidean")
>
> #using hierarchical clustering
> df_cluster<-hclust(clust_d, method="complete")
>
> #plotting the dendrogram
> plot(df_cluster)
>
> #plotting a histograms with 4 clusters
> Cut_clust=cutree(df_cluster,k=4)
> Cut_clust
       Alabama         Alaska        Arizona       Arkansas     California       Colorado    Connecticut       Delaware
             1              2              3              1              3              4              4              3
       Florida        Georgia         Hawaii          Idaho       Illinois        Indiana           Iowa         Kansas
             3              1              3              4              3              3              4              4
      Kentucky      Louisiana          Maine       Maryland  Massachusetts       Michigan      Minnesota    Mississippi
             1              1              4              3              3              3              4              1
      Missouri        Montana       Nebraska         Nevada  New Hampshire     New Jersey     New Mexico       New York
             3              4              4              4              4              3              1              3
North Carolina   North Dakota           Ohio       Oklahoma         Oregon   Pennsylvania   Rhode Island South Carolina
             1              4              3              3              3              3              4              1
  South Dakota      Tennessee          Texas           Utah        Vermont       Virginia     Washington  West Virginia
             4              1              3              4              4              3              3              1
     Wisconsin        Wyoming
             4              4
> hist(Cut_clust)
```



b) Here we shall be using self-organizing maps, so first let us convert the data set into a data frame, using the function as.data.frame() and specifying *"dataframe"* as the type of object to be converted to.

Later we shall use the scale function to scale our given data set and store it in a variable named sstate_scaled.

```
> #b)Self-Organising Maps (SOM)
>
> #converting the matrix data into dataframe
> df<- as.data.frame(state.x77,"dataframe")
>
> state_scaled <- scale(df)
```

After scaling, we see the data as this:

```
> state_scaled
       Population      Income Illiteracy     Life Exp       Murder     HS Grad       Frost        Area
 [1,] -0.14143156 -1.32113867   1.525758 -1.362193670  2.091810096 -1.46192933 -1.62482920 -0.23471832
 [2,] -0.86939802  3.05824562   0.541398 -1.168509784  1.062429318  1.68280347  0.91456761  5.80934967
 [3,] -0.45568908  0.15330286   1.033578 -0.244786635  0.114315444  0.61805142 -1.72101848  0.50020474
 [4,] -0.47853603 -1.72148373   1.197638 -0.162843452  0.737361704 -1.63526106 -0.75912574 -0.22022120
 [5,]  3.79697895  1.10371551  -0.114842  0.619341473  0.791539640  1.17518912 -1.62482920  1.00349033
 [6,] -0.38199648  0.72940916  -0.771082  0.880069781 -0.156574234  1.33614002  1.18389757  0.38709909
 [7,] -0.25678625  1.48453153  -0.114842  1.192943751 -1.158866044  0.35805383  0.66447550 -0.77201412
 [8,] -0.82146423  0.60735274  -0.442962 -0.609806266 -0.319108041  0.18472210 -0.02808727 -0.80576651
 [9,]  0.90280832  0.61711725   0.213278 -0.162843452  0.899895511 -0.06289466 -1.79796989 -0.19508270
[10,]  0.15333885 -0.56113404   1.361698 -1.742112063  1.766742482 -1.54859519 -0.85531502 -0.14840362
[11,] -0.75673121  0.85797525   1.197638  2.027274338 -0.319108041  1.08852326 -2.00958630 -0.75369642
[12,] -0.76905064 -0.51556631  -0.935142  0.738531557 -0.562908752  0.79138315  0.41438339  0.13994490
[13,]  1.55685818  1.09232357  -0.442962 -0.550211224  0.791539640 -0.06289466  0.43362124 -0.17565164
[14,]  0.23890291  0.03612870  -0.771082  0.001042913 -0.075307331 -0.02575214  0.33743197 -0.40595308
[15,] -0.31031978  0.31278991  -1.099202  1.252538793 -1.375577787  0.72947896  0.68371335 -0.17338976
[16,] -0.44045778  0.37951409  -0.935142  1.267437554 -0.779620494  0.84090650  0.18352913  0.12951447
```

Next, we shall move forward to set the seed(). Here we have used a random 500as our seed number.

The somgrid() function is used to record the coordinates of the grid to be used for a batch.We store the grid of the SOM, and the topology is set to hexagonal.

```
> #fiting an SOM(self organized maps)
> set.seed(500)
>
> #setting the grid for SOM and the topology
> S.grid <- somgrid(xdim= 5, ydim =5, topo ='hexagonal')
> S.grid
$pts
       x         y
 [1,] 1.5 0.8660254
 [2,] 2.5 0.8660254
 [3,] 3.5 0.8660254
 [4,] 4.5 0.8660254
 [5,] 5.5 0.8660254
 [6,] 1.0 1.7320508
 [7,] 2.0 1.7320508
 [8,] 3.0 1.7320508
 [9,] 4.0 1.7320508
[10,] 5.0 1.7320508
[11,] 1.5 2.5980762
[12,] 2.5 2.5980762
[13,] 3.5 2.5980762
[14,] 4.5 2.5980762
[15,] 5.5 2.5980762
[16,] 1.0 3.4641016
[17,] 2.0 3.4641016
[18,] 3.0 3.4641016
[19,] 4.0 3.4641016
[20,] 5.0 3.4641016
[21,] 1.5 4.3301270
[22,] 2.5 4.3301270
[23,] 3.5 4.3301270
[24,] 4.5 4.3301270
[25,] 5.5 4.3301270

$xdim
[1] 5

$ydim
[1] 5

$topo
[1] "hexagonal"

$neighbourhood.fct
[1] bubble
Levels: bubble gaussian

$toroidal
[1] FALSE

attr(,"class")
[1] "somgrid"
```

Finally, we shall train the data set and perform various visualization techniques on it.

We use the function SOM() to create Self Organizing maps, and this generated model is now stored in a variable named "*model*". Then we perform plotting as shown below.

```
> #training the SOM Model
> model <- som(state_scaled, grid = state_grid,  rlen=500)
>
> #plotting the trained SOM Model with the type distance neighbors
> clr <- function(n , alpha = 1){rainbow(n, end = 4/6, alpha = alpha)[n:1]}
>
> plot(model , type = "dist.neighbours" , palette.name = clr)
>
> plot(model, type="mapping")
```

Spatial weights are necessary in using the areal data hence above use type = dist.neighbours . We use mapping to map the model and consequently plot it.

c) **Advantages of hierarchical clustering:**

- Implementation is easy.
- The dendrogram is easier to analyze and is very informative as the data is in detailed hierarchy.
- The number of clusters need not be specified.

Advantages of SOM:

- SOMs are easy to interpret as they represent the similarity in data points.
- Several types of SOM can be used to perform in detailed analysis of the data.
- Easier to implement.

Q3)

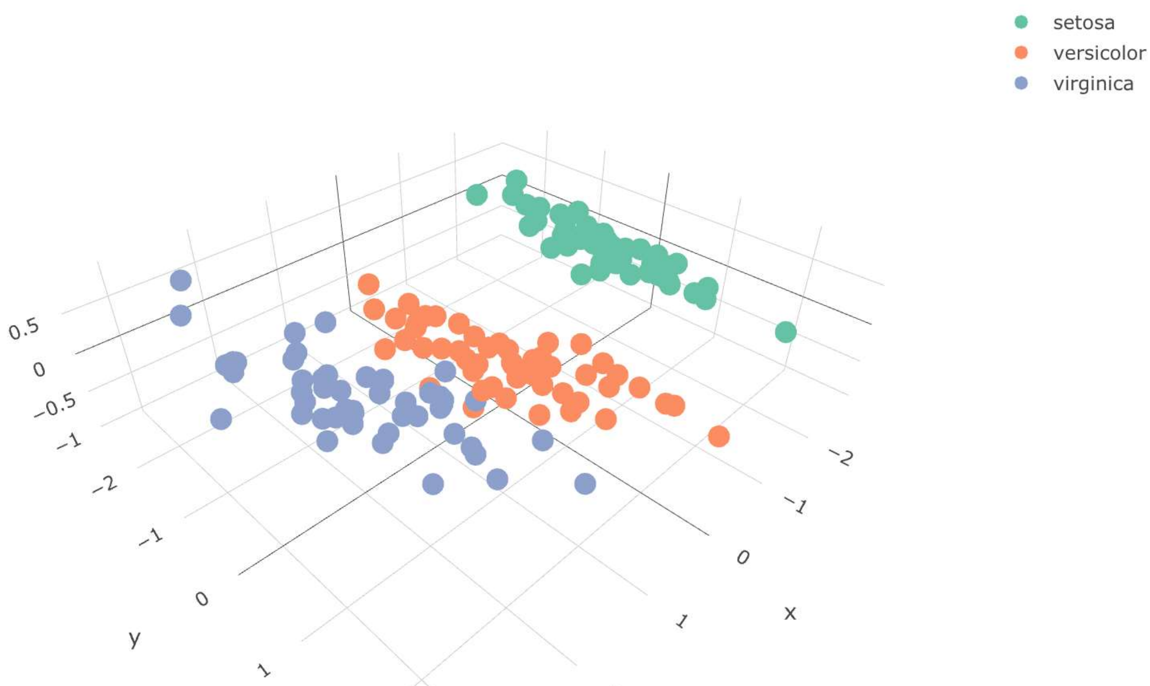
We have used the following libraries

```
library('cluster')
library('fpc')
library('multtest')
library('bootcluster')
library('fossil')
```

a)The plot using two principal components is plotted using the code

```
# alternatively
library(plotly)
plot_ly(x=pc_ex$x[,1], y=pc_ex$x[,2], z=pc_ex$x[,3], type="scatter3d", mode="markers", color=YY)
colors_str
```

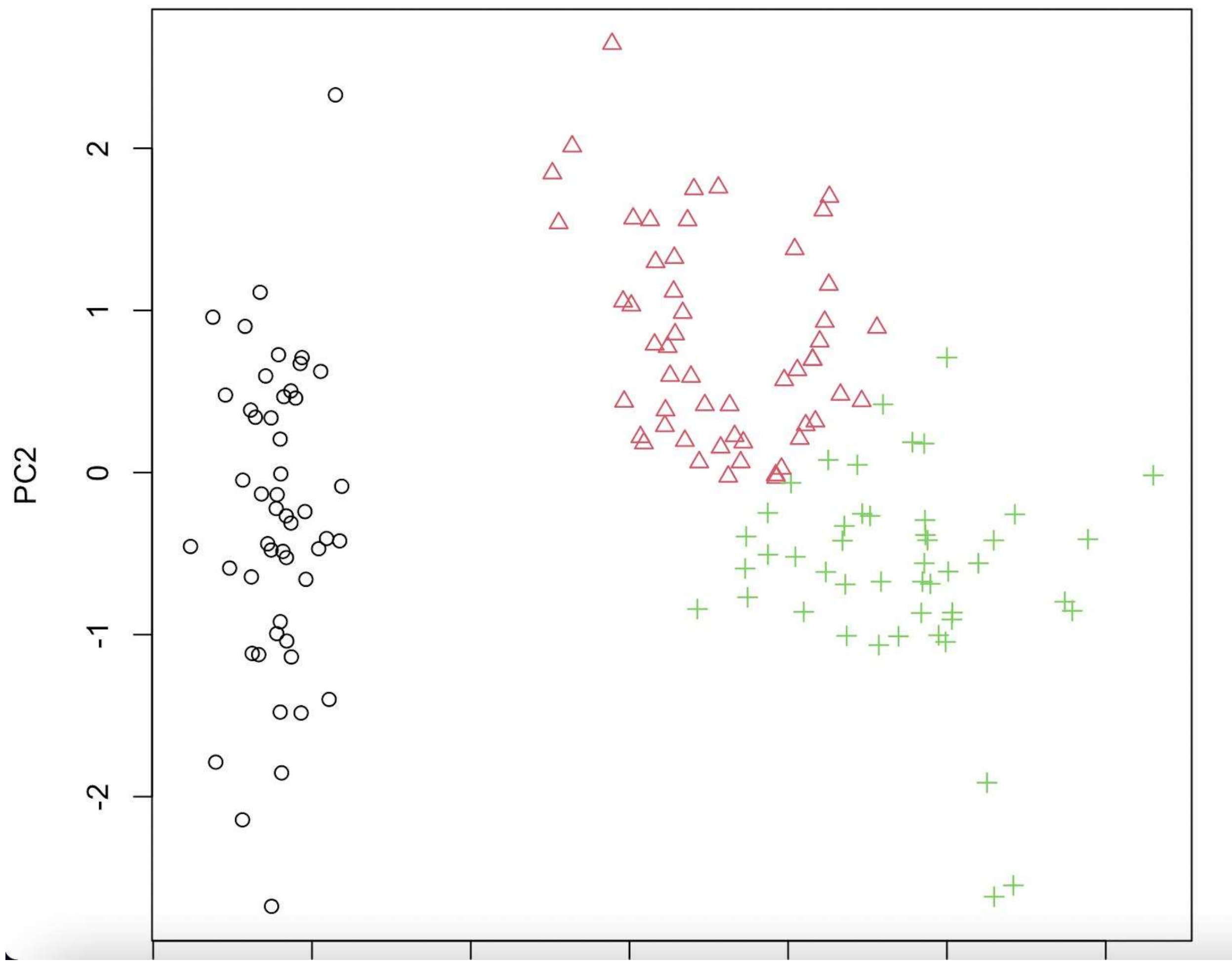
The output plot is



b) K-means clustering is achieved using the following code,

```
#b)K-means clustering
k=kmeans(reduced_data, centers=3,nstart=10)
k
quartz()
plot(reduced_data,  col = k$cluster, pch=k$cluster)
```

The output is



c) rand index and adjusted rand index is computed using the code,

```
#c)rand.index and adj.rand.index
rand.index(k$cluster,as.numeric(iris$Species))
adj.rand.index(k$cluster,as.numeric(iris$Species))
```

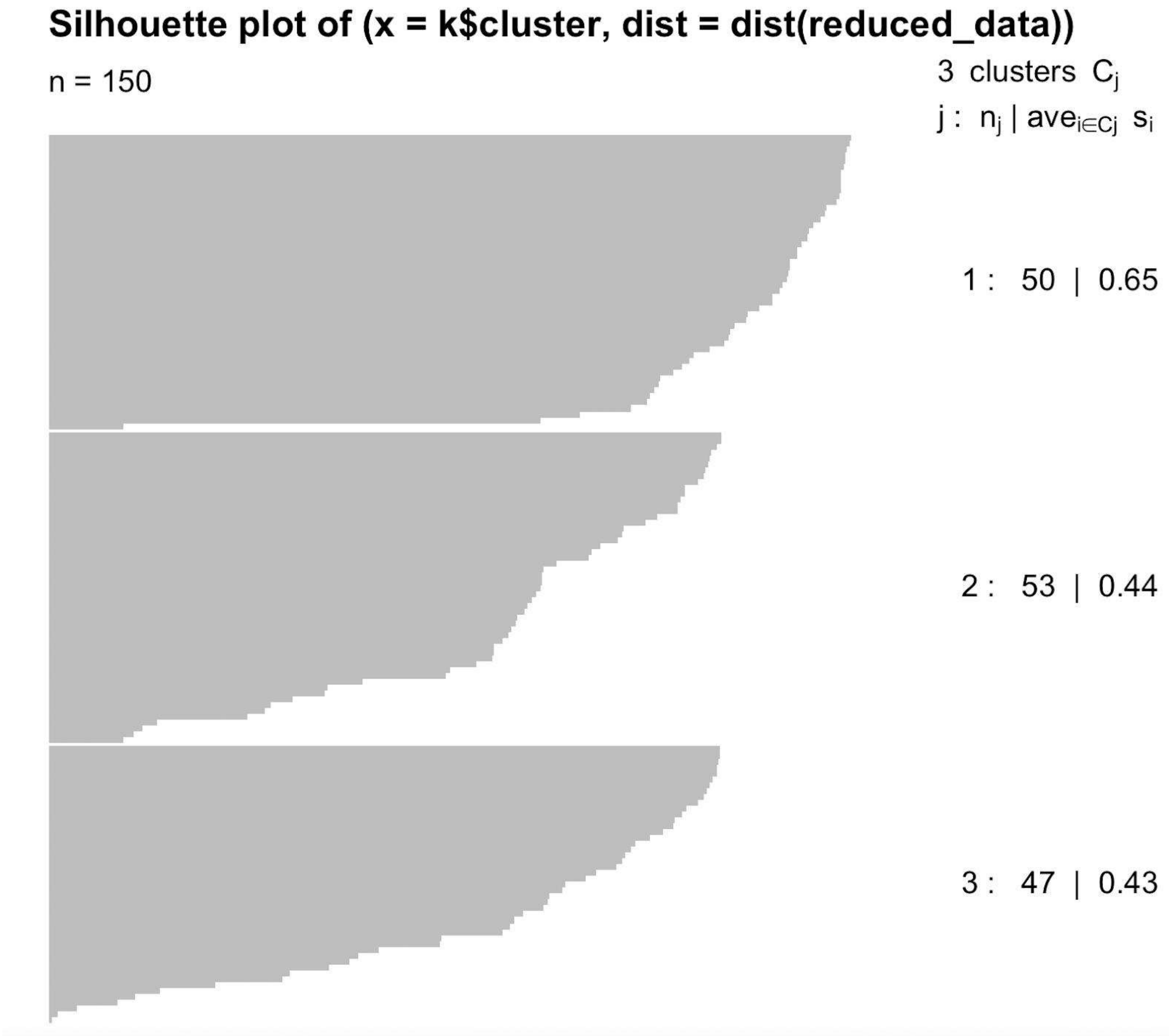
The output of the code is as follows,

```
> rand.index(k$cluster,as.numeric(iris$Species))
[1] 0.8322148
> adj.rand.index(k$cluster,as.numeric(iris$Species))
[1] 0.6201352
```

d)The silhouette is plotted using the code,

```
#d)silhouette plots and gap statistics
si = silhouette (k$cluster,dist(reduced_data))
quartz()
plot(si)
```

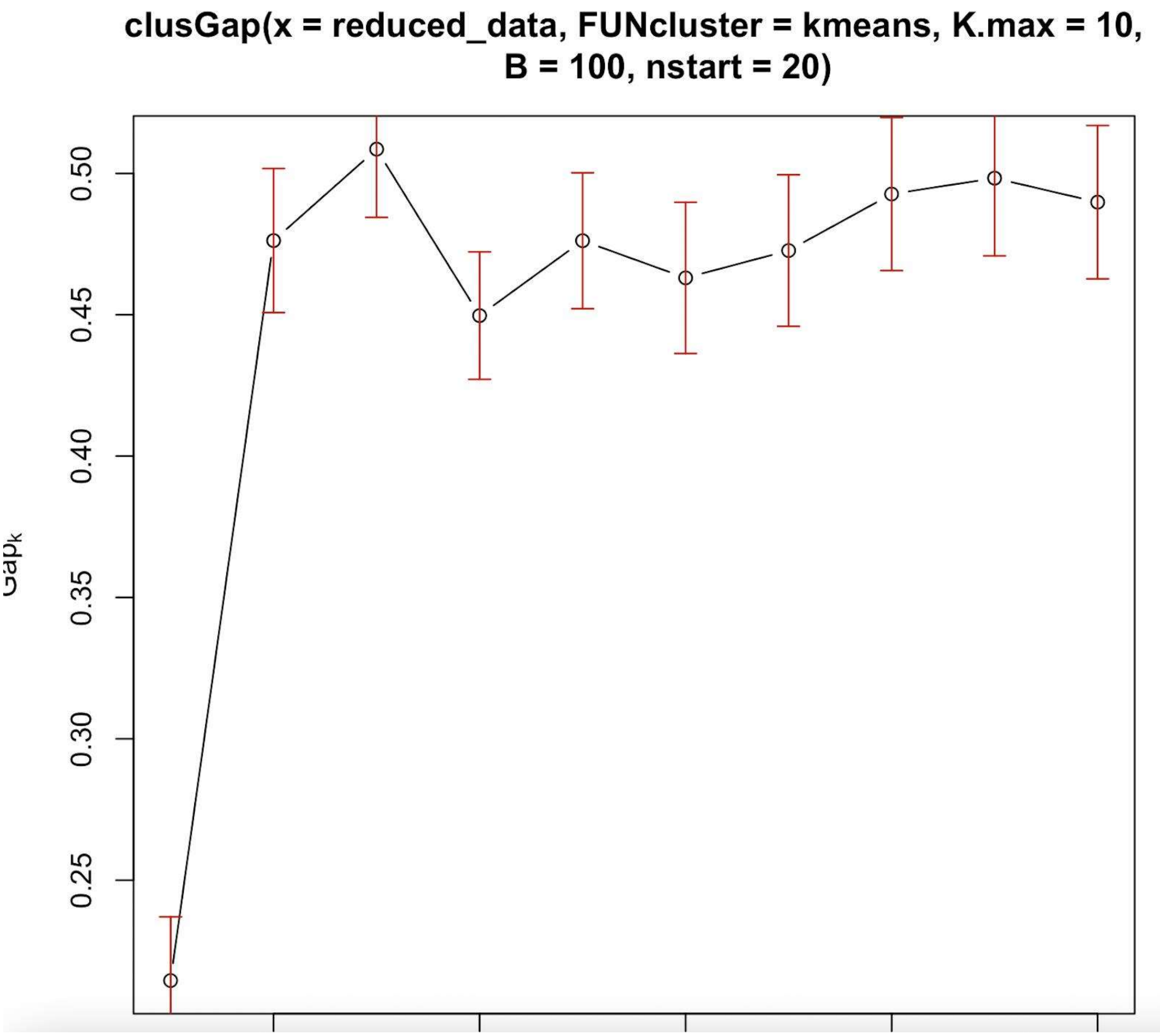
The output is as shown below



The gap statistics is computed using the code,

```
g=clusGap(reduced_data,kmeans,nstart=20,K.max = 10,B=100)
quartz()
plot(g)
```

The output is as follows,



e) Rand index is computed to correlate the results of two clustering techniques, in our task we compare the results k means clustering and the value in the species column of the dataset used.

Rand index is 0.8322 which means that the clustered values and the actual species are highly correlated.

The adjusted rand index is the correction of the rand index. The value is 0.6201.

The silhouette is plotted for the k means clustering with 3 clusters. Cluster 1 has 50 data points with cluster score 0.65,Cluster 2 has 53 data points with cluster score 0.44 and cluster 3 has 47 data points with cluster score 0.43

The average silhouette score for 3 clusters is around 0.5 which indicates that the k-means clustering for 3 clusters is quite efficient.

Observing the output of gap statistics , we see that the x-axis shows the number of clusters and the y-axis shows the gap value.10 is the maximum number of clusters. The gap value is highest ie around 0.5 for 3 as number of clusters .The gap value is the lowest for 4 clusters ie around 0.45. 3 is the ideal number of clusters for k-means clustering. The gap value for 9 clusters is close enough to the gap value of 3 clusters.