# STAT 5703

# Assignment 2

**Vidhi ghiya**

**StudentID#101088148**

# Question 1

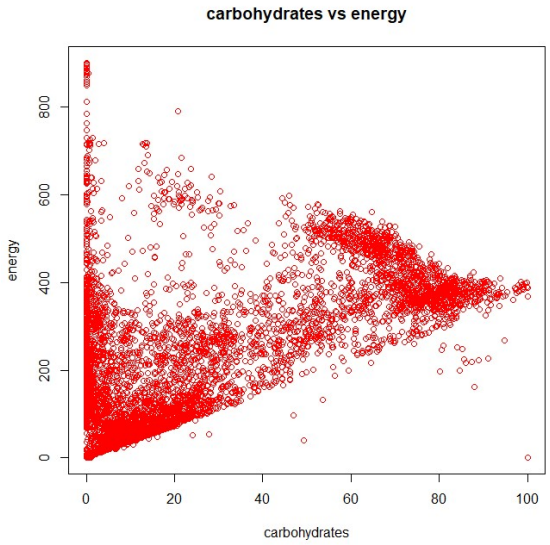
For the nutrient dataset given, to perform PCA we will be following these steps:

1. Cleaning of the data
2. Visualizing data using histogram and plot function
3. Perform PCA

Here, we read the dataset named USDA nutrient Dataset using the read.csv(). We use the paste function here. After that we start cleaning of the data and for that we remove null values from the file. We use na.omit() on the given dataset.

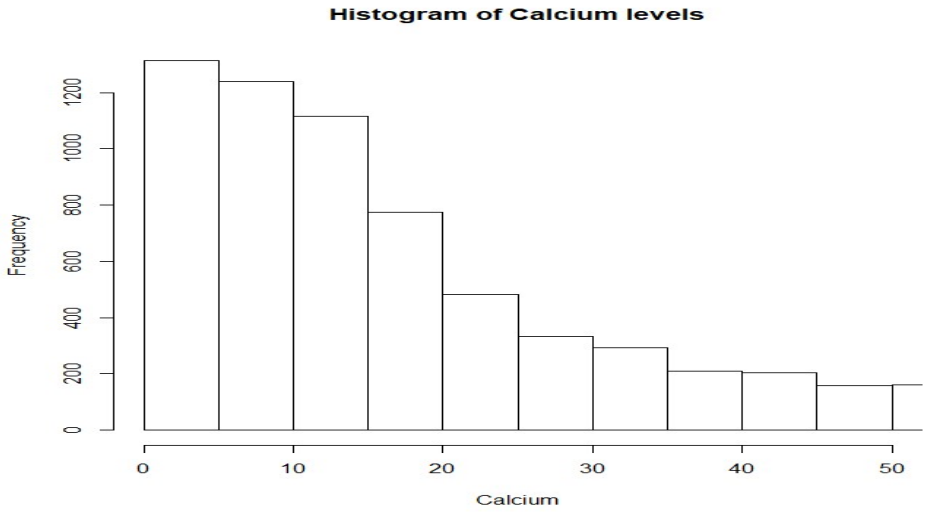
Thereafter, **for cleaning the dataset** we are reducing the number of columns and keep the eye on nutrient content and thereafter making the subset of the data which will be containing only the nutrient content

For **the visualization**, I am visualizing the carbohydrate content. Therefore, I am making the subset of the main original dataset with respect to Carbohydrates which is half of the maximum value and a random selected number, that is it should be greater than 50. Using **Carbohydrates_subset$Descrip** we check the description that have the content of carbohydrate greater than 50, we also see the energy content corresponding to it as we will be plotting the graph of carbohydrates vs energy.




From the diagram above, we can say that the level of carbohydrate is equal in all the food components and has the maximum value as 100. For the energy when compared to carbohydrate most of the food components have energies between 200-600. So, for the food to have high energy, the food should have very low level of the carbohydrates that is the food with highest carbohydrates has the level of energy near to 400. We can say that with the increase in carbohydrates the energy content keeps decreasing. It saturates near 400.

There is some food which should not be consumed as it low carbohydrate level and also have negligible low energy content.



For the figure above, that is the histogram of calcium, most of the food products have very less amount of calcium in it. As from the figure, we can see that 1200products have 0 calcium. The mean of values is good as there are some content in the food which provides calcium which is 73.41 (mean =73.41). we can also say that there are few products with more calcium value as the graph is continuously degreasing as we increase the calcium content and on an average frequency of products with any calcium content is near to 200.

We can perform the similar comparison with other contents and look for the number of products containing them.

We then perform the PCA that is Principal component analysis. For that we will use the standard library and its function named prcomp() on the subset that we created after cleaning the data. The results were not the one which we were looking for.

```
Importance of components:
                            PC1       PC2       PC3       PC4       PC5      PC6      PC7      PC8      PC9     PC10     PC11     PC12     PC13     PC14     PC15     PC16
Standard deviation     779.8274 261.12291 184.47420 162.71048 131.88656 57.51117 47.38853 28.31842 23.48721  9.99967 8.99951 5.83276 4.77222 3.59787 3.43407 3.33879
Proportion of Variance   0.7987   0.08955   0.04469   0.03477   0.02284  0.00434  0.00295  0.00105  0.00072  0.00013 0.00011 0.00004 0.00003 0.00002 0.00002 0.00001
Cumulative Proportion    0.7987   0.88824   0.93293   0.96770   0.99055  0.99489  0.99784  0.99889  0.99962  0.99975 0.99986 0.99990 0.99993 0.99995 0.99996 0.99998
                           PC17    PC18  PC19   PC20   PC21   PC22   PC23      PC24      PC25      PC26      PC27      PC28      PC29      PC30      PC31      PC32
Standard deviation     2.90024 2.56447 1.424 0.5159 0.3771 0.3236 0.2829 2.619e-10 2.369e-10 2.275e-10 2.176e-10 2.149e-10 2.068e-10 1.974e-10 1.92e-10 5.324e-14
Proportion of Variance 0.00001 0.00001 0.000 0.0000 0.0000 0.0000 0.0000 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.00e+00 0.000e+00
Cumulative Proportion  0.99999 1.00000 1.000 1.0000 1.0000 1.0000 1.0000 1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.00e+00 1.000e+00
                           PC33
Standard deviation     1.786e-15
Proportion of Variance 0.000e+00
Cumulative Proportion  1.000e+00
```

# Question 2

Here we are going to perform clustering on the president election data. For that we are following the k-means and the distance clustering method.
After setting the drives and the work directory we are going to going to perform the following operations:

1. Data Cleaning
2. K-means
3. Euclidean clustering

## Data Cleaning

It will be easy to use the variables directly, so we start with reading the files into variables. We are using the library named **dplyr** for the functions like **innerjoin()** for joining the **county_facts** and votes file. But before that we need to clean the file of county_facts.

In county_fact file there were some unwanted data, like having null values. Therefore we removed such rows using **facts<- facts[facts[,3]!="",]** and thereafter **innerjoin()** was used as mentioned before on county_facts and votes file.

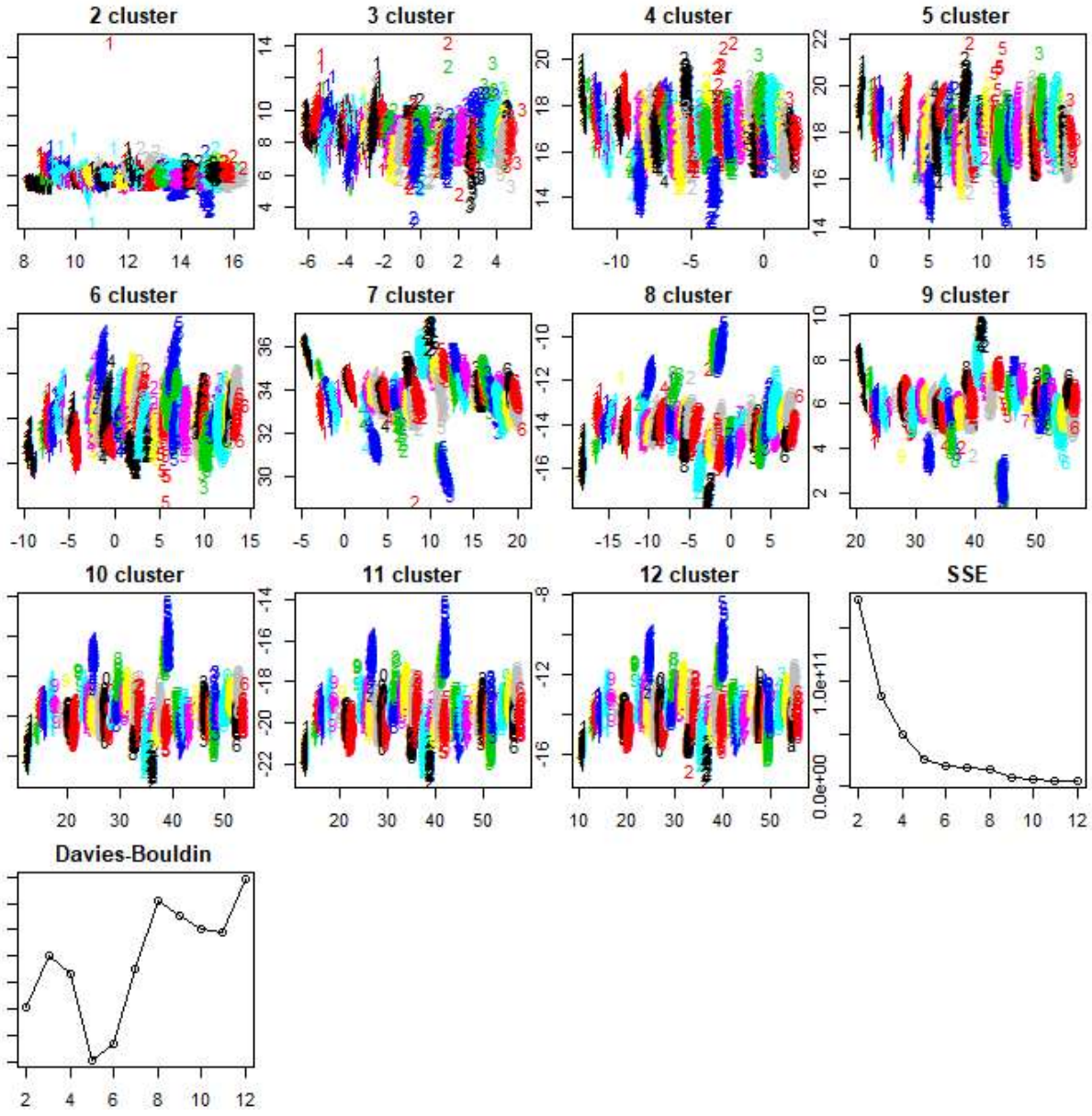
Using the as.numeric() we converted the data into numeric form then adding column named winning which was the comparison of the 2 people from the republic party who were being voted for figuring it out who was voted the most in that particular state.

We are here performing the clustering on only few columns as I removed few of them having the value 0 (because while performing the k means it gives the eigen value error).

## K-means

For we perform the K-means clustering. We set the value of seeds to set to 654321 and doing the for loop with the iteration from 2-12. For finding the value of center(k), we are also calculating the number of misclassifications and the value of DBI.

We got the following results when we used K from 2-12. We can see that the cluster with value of K=5, it is not that clear but still we can see the clusters and they are separated, and we can even look for it in the SSE plot, which also gives k as 5 as the best value.



```
---------Analysis of cluster having K= 5 ---------

Size of all cluster
Votes:

      1   2   3   4   5
  1 129  76 131  54  98
  2 388 482 695 615 444

Total misclassifications= 488
SSE = 26104363005.8061
DBI = 0.52093147164756
```

For looking for the best seed for this center, we need to find the minimum misclassification value. The results below show that it is 1. And therefore, we now calculate the optimal results for the better result using **K=5** and **Seed=1.**

```
Minimum misplacements = 488
Seed of minimum misplacements = 1
```

Let's see the centers first.

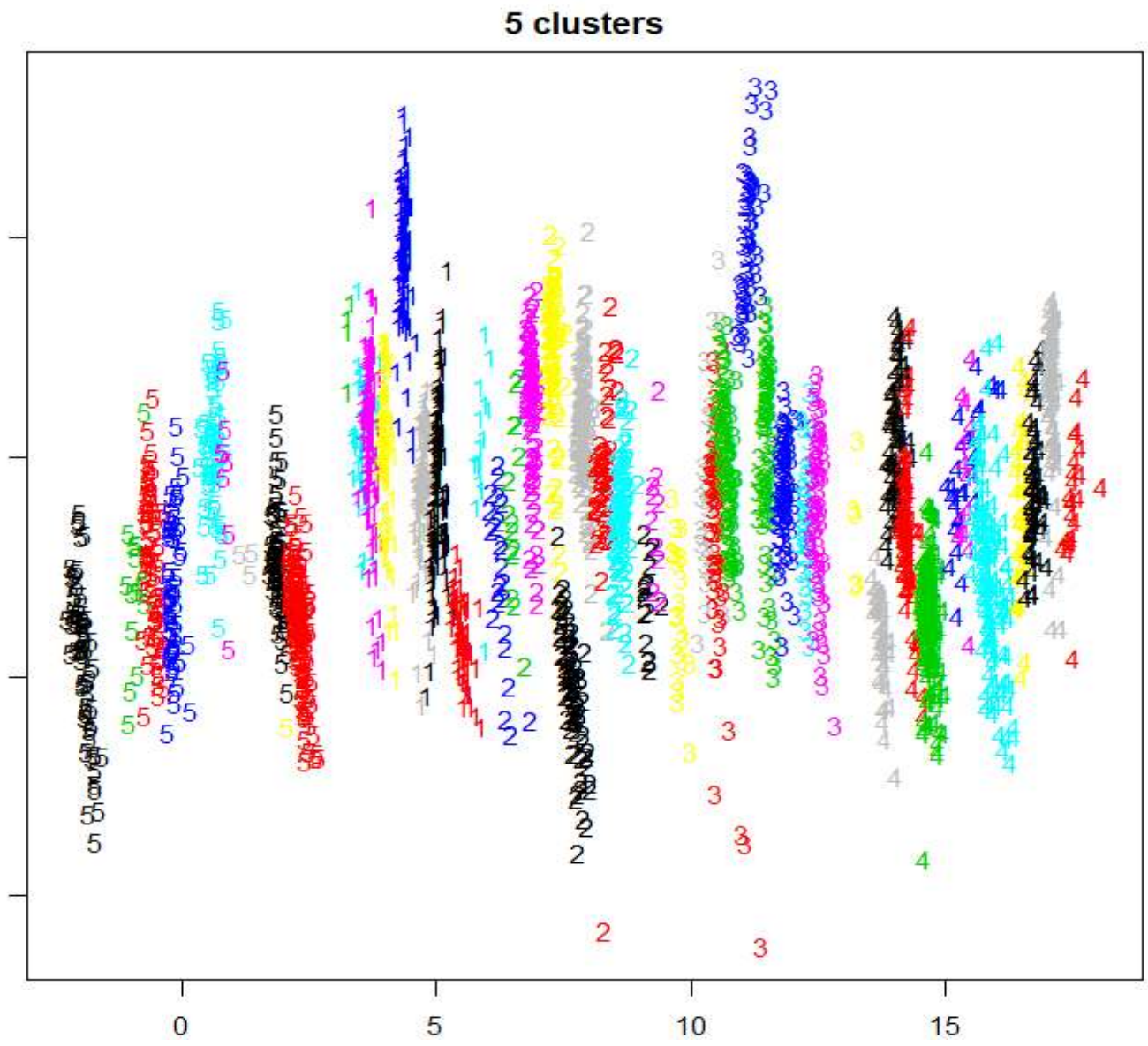
```
> print(Centre$centers)
     fips area_name.x  Clinton     Trump state_abbr     Obama    Romney Clinton_Obama Trump_Romney Trump_Prediction Clinton_Prediction Trump_Deviation  winning
1 19336.536   957.2337 0.2793709 0.6682448  15.201238 0.3621926 0.6185602   -0.08282168   0.04968466        0.6837960         0.2686487     0.015551198 1.925697
2 28484.773   961.8830 0.3099603 0.6412846  24.779690 0.3930654 0.5880927   -0.08310506   0.05319191        0.6449178         0.3054954     0.003633203 1.858864
3 38513.970   971.7048 0.3379798 0.6136919  33.070111 0.4171334 0.5669330   -0.07915361   0.04675893        0.6240795         0.3253110     0.010387608 1.819188
4 49731.834   961.7712 0.3084325 0.6455536  44.185230 0.3650472 0.6191162   -0.05661471   0.02643741        0.6337773         0.3203603    -0.011776296 1.841404
5  8498.975   959.1663 0.3640460 0.5988108   5.969052 0.4022801 0.5812682   -0.03823408   0.01754256        0.5832212         0.3766858    -0.015589608 1.750484
```

This is how the data is divided into clusters.

```
> kmclusterPlots <- bestResult(votes.new,votes.new,5,1)
[1] "best Cluster=  5  Analysis= "

      1   2   3   4   5
  1  48  82  98 131 129
  2 598 499 444 695 388
```

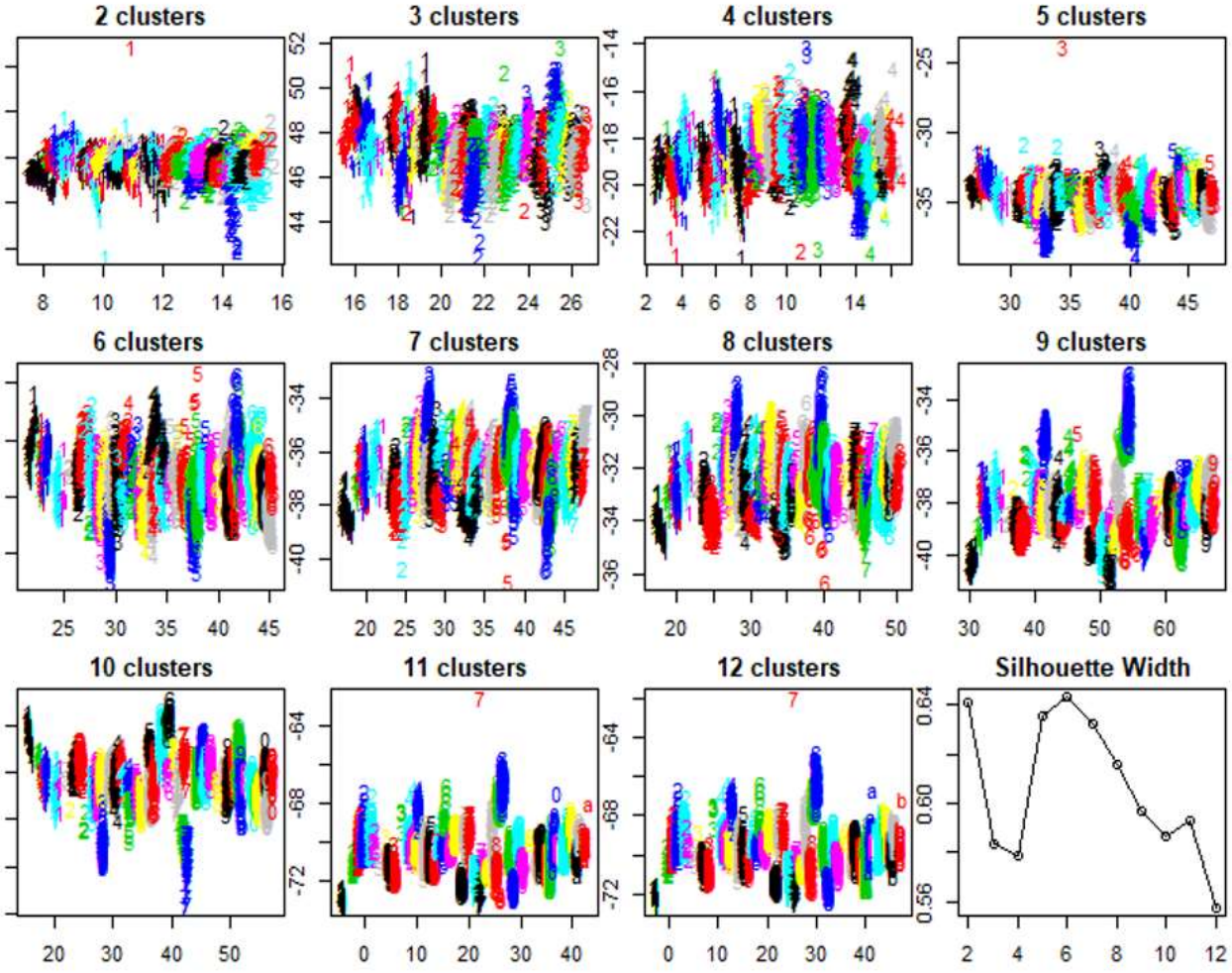
This is how the data is divided into 5 clusters. Color states various states of USA and number shows the cluster centers. It is not clear and while forming the clusters it has many misclassifications


5 clusters
0
5
10
15

## Euclidean clustering

Here we use the method called Euclidean clustering. Euclidean clustering method forms cluster using the distance (in any direction or angle). We use the pam() where we assign the set metric variable as Euclidean. To find the center (k ) value, using the same number of iterations in for loop we can calculate the number of misclassifications and the average silhouette width. In silhouette graph, maximum value at any point gives the best cluster.

The picture below shows the result for the value of K from 2-12. The maximum value is with 6 clusters.

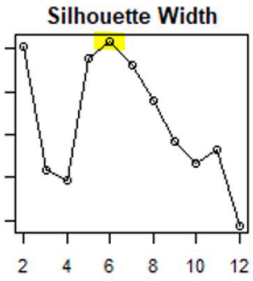


```
---------Analysis of Clusters whose K= 6 ---------

Size of all clusters
Votes:

     1   2   3   4   5   6
  1 87  48  49  75  94 135
  2 200 230 573 482 443 696

Total misclassifications= 488
Average silhoutte width= 0.643326189881483
```




The best K value from Silhouette width value is for **K=6** as shown above (for me). After finding the best cluster using Euclidean distance, then after for finding the best seed for this center we are getting the value by calculating the minimum misclassification. Here I get the value of seed of minimum misplacements as 1

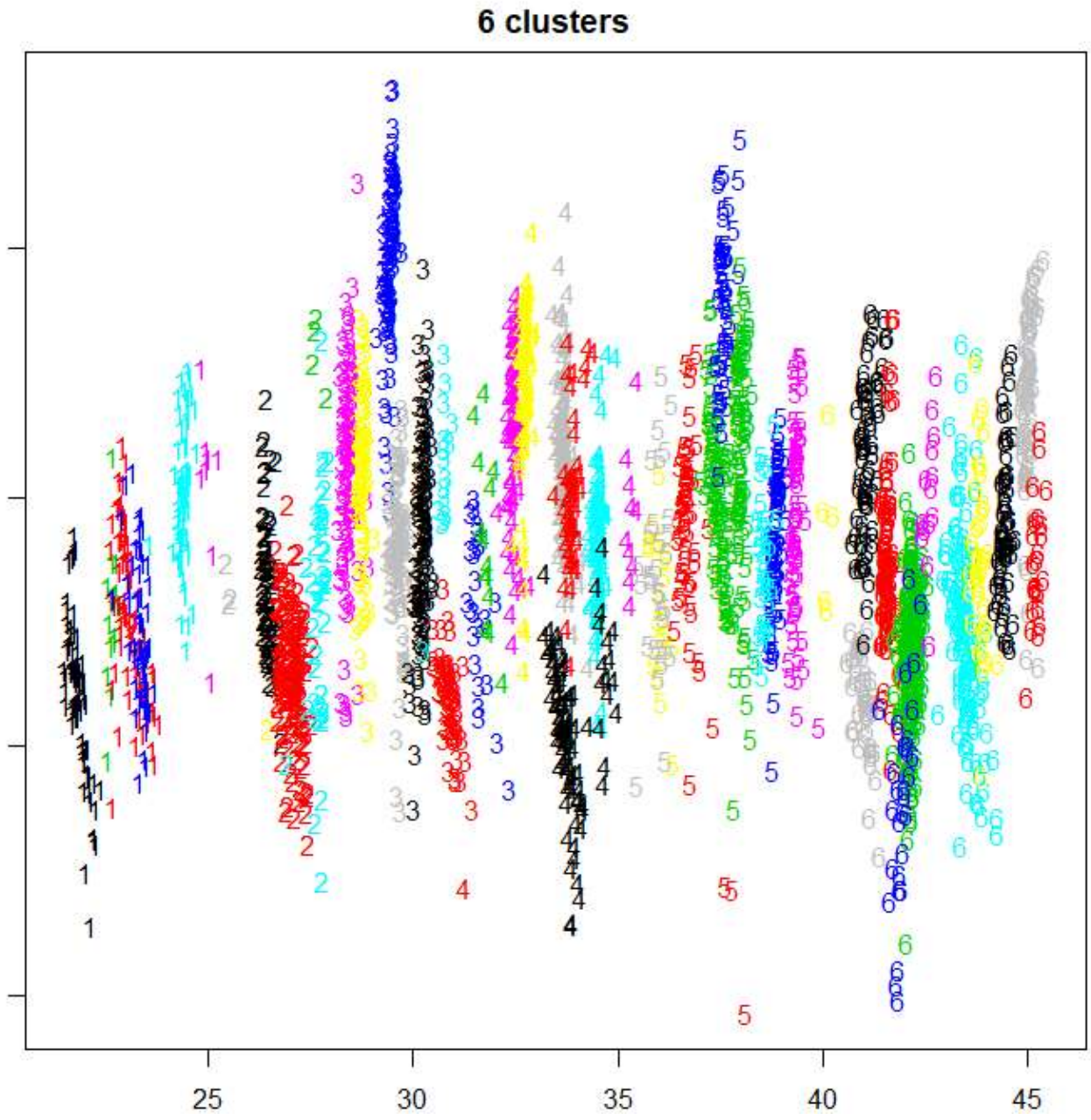
```
Minimum misplacements =  488
Seed of Minimum misplacements=  1
```

Now to find the optimal solution we use K=6 and value of seed=1.

```
1] "best Cluster= 6  Analysis="

     1   2   3   4   5   6
 1 201 201 534 498 456 629
 2  86  77  88  59  81 202
```

The figure below shows how the data is divided into 6-cluesters. Here the colors represent states of USA and the number shows the center of the cluster. From the picture, it is not clear and while forming the cluster there are many misplacements.




We saw that both methods: K-means and Euclidean gave almost same results.

# Appendix for question1

```
windows(record=TRUE)

# Installing required packages
#install.packages("arules")
#install.packages("rggobi")

# Calling necessary libraries
#library(arules)
#library(rggobi)

#--------------------------------------#
# Setting up the directory and database #
#--------------------------------------#

Student.Number <- "101088148"
Initials <- "VAG"
ASorLAB <- "Assignment"
Assignment.Number <- "2"
Student.info <- paste(Student.Number, Initials, ASorLAB, Assignment.Number, sep="-")

# "drive", AND "path.up" SHOULD BE THE ONLY PARTS THAT REQUIRE YOUR
PROFESSOR
# OR TA TO BE ABLE TO RUN YOUR CODE

drive="S:"
path.upto <- paste("Grad 3 term", "Data Mining", sep="/" )
code.dir <- paste(drive, path.upto, Student.info, "Code", sep="/")
data.dir <- paste(drive, path.upto, Student.info, "Data", sep="/")
work.dir <- paste(drive, path.upto, Student.info, "Work", sep="/")
```

```
setwd(work.dir)


# Reading data into nutrition variable from the data folder from the directory
nutrition <- paste(data.dir,"nndb_flat.csv", sep="/")
nutrition <- read.csv(nutrition)


#REmoving the null values from the databse
nutrition <- na.omit(nutrition)


#reducing data by just considering the nutrients and removing every other thing
nutrition1 <-nutrition[,c(8:40)]
summary(nutrition1)


#creating subset of original data by checking where Protein nutrient is greater than 50
carbohydrates_subset<-subset(nutrition, Carb_g>50)
carbohydrates_subset$Descrip
carbohydrates_subset$Energy_kcal


#plotting the graph for protein and fat and visualizing accordingly
plot(nutrition$Carb_g, nutrition$Energy_kcal, xlab="carbohydrates", ylab="energy",
main="carbohydrates vs energy", col="Red")

#plotting histogram for Vitc_mg to check the frequency of occurences
hist(nutrition$Calcium_mg, xlab = "Calcium", main = "Histogram of Calcium levels", xlim =
c(0,50), breaks = 2000)
```

```
#performing PCA on the data
nutrition_pca <- prcomp(nutrition1)
head(nutrition_pca)
summary(nutrition_pca)
```

## Appendix for question 2

```
windows(record=TRUE)

# Installing required packages
#install.packages("rggobi")

# Calling necessary libraries
#library(ggplot2)
#library(rggobi)

#---------------------------------------#
# Setting up the directory and database #
#---------------------------------------#

Student.Number <- "101088148"
Initials <- "VAG"
ASorLAB <- "Assignment"
Assignment.Number <- "2"
Student.info <- paste(Student.Number, Initials, ASorLAB, Assignment.Number, sep="-")

# "drive", AND "path.up" SHOULD BE THE ONLY PARTS THAT REQUIRE YOUR PROFESSOR
# OR TA TO BE ABLE TO RUN YOUR CODE

drive="S:"
path.upto <- paste("Grad3term", "Data Mining", sep="/" )
code.dir <- paste(drive, path.upto, Student.info, "Code", sep="/")
data.dir <- paste(drive, path.upto, Student.info, "Data", sep="/")
work.dir <- paste(drive, path.upto, Student.info, "Work", sep="/")
setwd(work.dir)
```

```
# Reding the data
vote <- paste(data.dir,"votes.csv", sep="/")
votes <- read.csv(vote)
head(votes)
fact <- paste(data.dir,"county_facts.csv", sep="/")
facts <- read.csv(fact)
head(facts)

library(dplyr)
# Cleaning of data
facts[1:10,1:10]
facts <-facts[facts[,3]!="",]
dim(facts)
facts[1:2,]
votes_facts<-inner_join(facts[,1:3],votes, by="fips")
dim(votes_facts)
votes_facts = lapply(votes_facts, as.numeric)
write.csv(votes_facts,file="votes.new.csv")
votesfactss<- paste(work.dir,"votes.new.csv", sep="/")
votes_factss <- read.csv(votesfactss)
head(votes_factss)
summary(votes_factss)

votes.new<-votes_factss[ ,c(2:3, 11:12, 15, 23:24, 80:84)]
head(votes.new)
dim(votes.new)
winning<-ifelse(votes.new$Trump > votes.new$Romney,1,ifelse(votes.new$Trump <
votes.new$Romney,2,NA))
summary(winning)
```

```
head(winning)
str(winning)
votes.new$winning <- winning
head(votes.new)

#Davies Bouldin

Davies.Bouldin <- function(A, SS, m) {
  # A - the centres of the clusters
  # SS - the within sum of squares
  # m - the sizes of the clusters
  N <- nrow(A) # number of clusters
  # intercluster distance
  S <- sqrt(SS/m)
  # Get the distances between centres
  M <- as.matrix(dist(A))
  # Get the ratio of intercluster/centre.dist
  R <- matrix(0, N, N)
  for (i in 1:(N-1)) {
    for (j in (i+1):N) {
      R[i,j] <- (S[i] + S[j])/M[i,j]
      R[j,i] <- R[i,j]
    }
  }
  return(mean(apply(R, 1, max)))
}

#K-Means Clustering

kmcluster <- function(data.new,data.select)
{
```

```
min.err <- 100
errs <- rep(0, 10)
DBI <- rep(0, 10)
size.cluster <- rep(0,10)
max.cluster <- rep(0,12)
library(cluster)
library(fpc)
oldpar <- par(mfrow = c(4,4))
par(mar=c(2,1,2,1))
for (i in 2:12) {
 set.seed(654321)
 k_m <- kmeans(data.new, i, 12)
 plotcluster(data.new,col=data.select$state_abbr, k_m$cluster, main=paste(i,"cluster"))
 errs[i-1] <- sum(k_m$withinss)
 DBI[i-1] <- Davies.Bouldin(k_m$centers, k_m$withinss, k_m$size)
 size <- "\n\nSize of all cluster"
 an <- "\n\n---------Analysis of cluster having K="
 end <- "---------"
 cat(paste(an,i,end))
 cat(size)
 cat("\nVotes:\n")
 con <- table(data.select$winning, k_m$cluster)
 print(con)
 misclassify<-0
 for (n in 1:i)
 {
  misclassify=misclassify+k_m$size[n]-max(con[,n])
 }
 mis<-("\nTotal misclassifications=")
 cat(paste(mis,misclassify))
 if(misclassify<min.err)
```

```
    {
      min.err<-misclassify
      min.err.cluster<-i
    }
    ss <- "\nSSE ="
    db <- "\nDBI ="
    cat(paste(ss,errs[i-1]))
    cat(paste(db,DBI[i-1]))
  }
  plot(2:12, errs, main="SSE")
  lines(2:12, errs)
  plot(2:12, DBI, main="Davies-Bouldin")
  lines(2:12, DBI)
  par(oldpar)
  cat(paste("\n\n\nLeast DBI=",min(DBI)))
}

kmclusterPlot <- kmcluster(votes.new,votes.new)

#best cluster is 5

#best seed

min.err <- function(data.new,data.select,ke)
{
  min.err <- 550
  for(j in 1:4000)
  {
    size.cluster <- rep(0,10)
    max.cluster <- rep(0,12)
    library(cluster)
```

```
   library(fpc)
   set.seed(j)
   k_m <- kmeans(data.new, ke, 12)
   con <- table(data.select$winning, k_m$cluster)
   misclassify<-0
   for (n in 1:ke)
   {
    misclassify = misclassify+k_m$size[n]-max(con[,n])
   }
   if(misclassify < min.err)
   {
    min.err <- misclassify
    min.err.seed<-j
   }
  }
  cat(paste("\nMinimum misplacements =",min.err))
  cat(paste("\nSeed of minimum misplacements =",min.err.seed))
}

kmclusterPlot2 <- min.err(votes.new,votes.new,5)

#Seed of minimum misplacements is 1

set.seed(1)
Centre <- kmeans(votes.new, 5, 12)
print(Centre$centers)
bestresult<-function(data.new,data.select,k,seed)
{
  set.seed(seed)
  k_m <- kmeans(data.new, k, 12)
  plotcluster(data.new,col=data.select$state_abbr, k_m$cluster, main=paste(k,"clusters"))
```

```
  print(paste("best Cluster= ",k," Analysis= "))
  incorrect <- rep(0,k)
  con <- table(data.select$winning, k_m$cluster)
  print(con)
  cat("\n\n")
}

kmclusterPlot3 <- bestresult(votes.new,votes.new,5,1)



#Euclidean Clustering

mancluster <- function(data.new,data.select)
{
  min.err<-100
  errs <- rep(0, 10)
  avg.width <- rep(0, 10)
  size.cluster <- rep(0,10)
  max.cluster <- rep(0,12)
  library(cluster)
  library(fpc)
  oldpar <- par(mfrow = c(4,4))
  par(mar=c(2,1,2,1))
  for (i in 2:12) {
    set.seed(654321)
    library(fpc)
    man <- pam(data.new, i, metric = "euclidean")
    plotcluster(data.new,col=data.select$state_abbr, man$cluster, main=paste(i,"clusters"))
    avg.width[i-1] <- man$silinfo$avg.width
    size <- "\n\nSize of all clusters"
```

```
    analysis <- "\n\n---------Analysis of Clusters whose K="
    en <-"---------"
    cat(paste(analysis,i,en))
    cat(size)
    cat("\nVotes:\n")
    con <- table(data.select$winning, man$cluster)
    print(con)
    misclassify<-0
    for (n in 1:i)
    {
      misclassify=misclassify+ man$clusinfo[n,1]-max(con[,n])
    }
    mis<-("\nTotal misclassifications=")
    cat(paste(mis,misclassify))
    if(misclassify<min.err)
    {
      min.err<-misclassify
      min.err.cluster<-i
    }

    db <- "\nAverage silhoutte width="
    cat(paste(db,avg.width[i-1]))
  }
  plot(2:12, avg.width, main="Silhouette Width")
  lines(2:12, avg.width)
  par(oldpar)
  cat(paste("\n\n\nHighest Average Silhoutte Width = ",max(avg.width)))
}
manPlot <- mancluster(votes.new,votes.new)

#best cluster is 6
```

```
#best seed
min.err <- function(data.new,data.select,ke)
{
 min.err<-550
 for(j in 1:10)
 {
  size.cluster <- rep(0,10)
  max.cluster <- rep(0,12)
  library(cluster)
  library(fpc)
  set.seed(j)
  man <- pam(data.new, ke, metric = "manhattan")
  con <- table(data.select$winning, man$clustering)
  misclassify<-0
  for (n in 1:ke)
  {
   misclassify=misclassify+man$clusinfo[n,1]-max(con[,n])
  }
  if(misclassify<min.err)
  {
   min.err<-misclassify
   min.err.seed<-j
  }
 }
 cat(paste("\nMinimum misplacements = ",min.err))
 cat(paste("\nSeed of Minimum misplacements= ",min.err.seed))
}
manPlot2 <- min.err(votes.new,votes.new,2)

#best result
```

```
bestresults<-function(data.new,data.select,k,seed)
{
  set.seed(seed)
  man <- pam(data.new, k, metric = "euclidean")

  plotcluster(data.new,col=data.select$state_abbr, man$cluster, main=paste(k,"clusters"))
  print(paste("best Cluster=",k," Analysis="))
  incorrect <- rep(0,k)
  con <- table(data.select$winning, man$clustering)
  print(con)
  cat("\n\n")
}
bestresults1 <-bestresults(votes.new,votes.new,6,1)
```