MACHINE LEARNING

NAME: ADITYA RANJAN
REG. NO.: 16BCE2310

REPORT,LAB ASSESMENT 4, LAB ASSESMENT 5

REPORT

ABSTRACT

Networks has become an invaluable tool to describe any complex system architecture. It is typically based on link prediction model to observe various application natures such as technological, biological and social behaviour. The objective of prediction modeling is to reduce the experimental effort considerably in order to accelerate the network topological interaction. The link prediction modeling already proves its significance in several application systems[1] namely biomedicine, recommendation system and social media to infer the feature interaction. Numerous methods such as similarity-based, maximum likelihood and probabilistic models are widely used and considered important to solve the link-prediction problem. Moreover, these models use undirected graph-networks to analyse the performance in different application domains. To improve the performance and working of various link prediction model, the standard statistics known as area under the curve (AUC) is employed. It is defined to be a probability to choose the random links in the given probe set $\varepsilon\ P$ that has a higher score (i.e. True Positive) than a random non-existent chosen link (i.e. True Negative). However, it is still lacking of powerful prediction model to synchronize with directed or undirected complex networks regardless of their application systems. To overcome the issue of link prediction model, a heuristic deepauto-encoder framework is proposed. A multiplex network and foursquare (i.e. locationbased social network) is considered to develop a auto-encoder link prediction model. The main goal of this model is not only to improve the prediction ratio of cross-layer communication but also to examine the features such as reputation and optimism of multiplex networks.

Keywords:

INTRODUCTION

Social networks are nowadays becoming important source of data to study the interactions between the people in groups or community.These can be visulaized as many forms like graphs in which vertex acts to person and an edge represents some association between them.Social networks are dynamic in nature i.e changes with time ,forming new users (nodes) and forming new links between different users .There are two types of social network -complete and ego centric. Any social networks which are egocentric can be seen as a collection of egonets connected to each other.

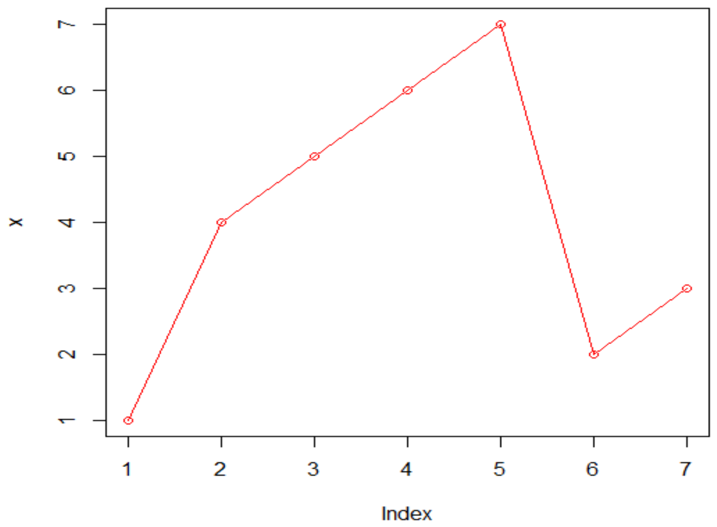
Egonets[2] consists of two types of nodes ego and alter .Ego node is the focal node(respondent) and actors that has ties with them are called alter.ego networks serves many purposes such as social support ,sense-making i.e how to interpret the world ,social control ensuring behaviour is according to norms ,access to resources ,influence(normative) pressure and helps in studying mixing patterns between groups.

LAB ASSESSMENT 4

1. Line Charts
2. Bar Charts
3. Histograms
4. Pie Charts
5. Dotcharts
6. Misc
7. Strip Charts
8. Histograms
9. Boxplots
10. Scatter Plots
11. Normal QQ Plots

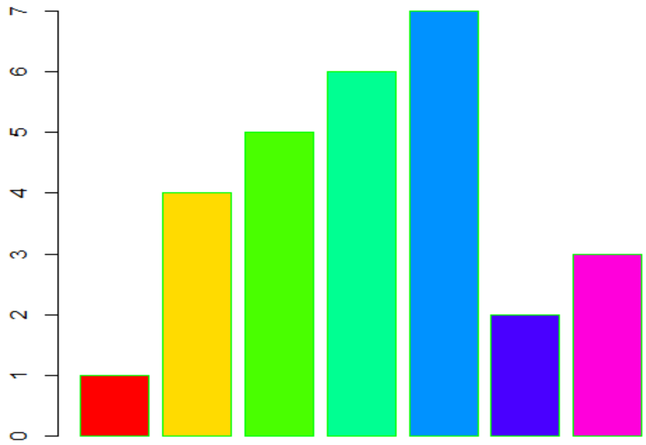
LINE CHART

```
> plot(x)

> plot(x, type="o", col="red")

> x<-c(1,4,5,6,7,2,3)
```

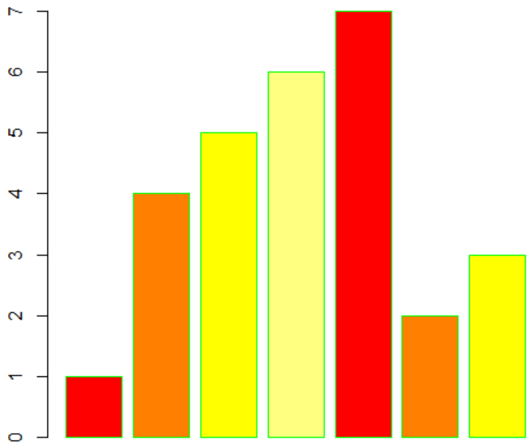


Bar Graph

> barplot(x, border="green", col=rainbow(7))

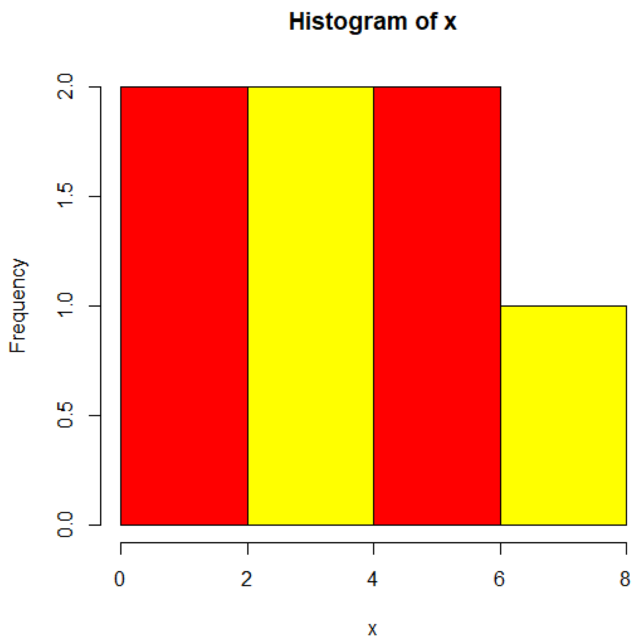


> barplot(x, border="green", col=heat.colors(4))



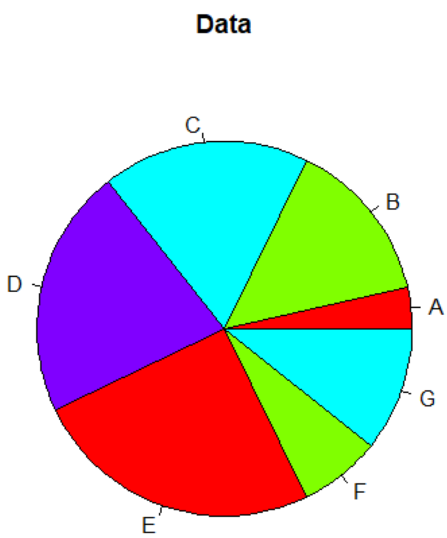
HISTOGRAM

hist(x, col=heat.colors(2))



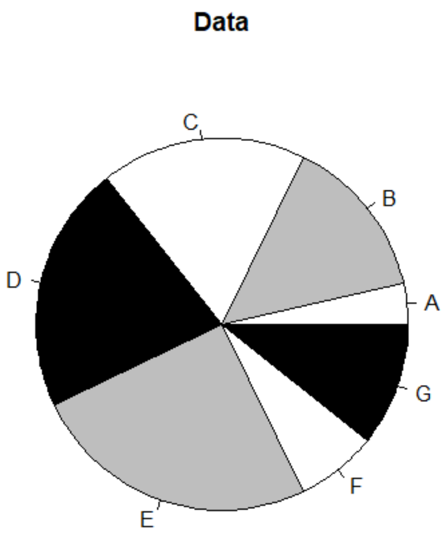
Pie Chart

pie(x, main="Data", col=rainbow(4),labels=c("A","B","C","D","E","F","G"))



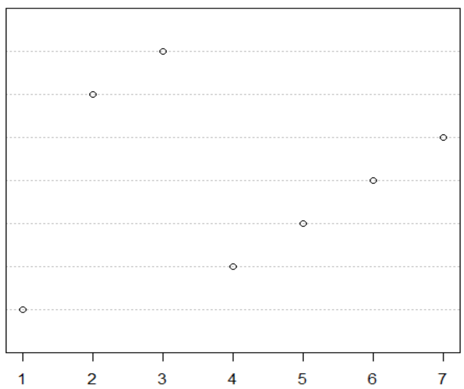
> colors<-c("white","grey","white","black","grey","white","black")

> pie(x, main="Data", col=colors,labels=c("A","B","C","D","E","F","G"))



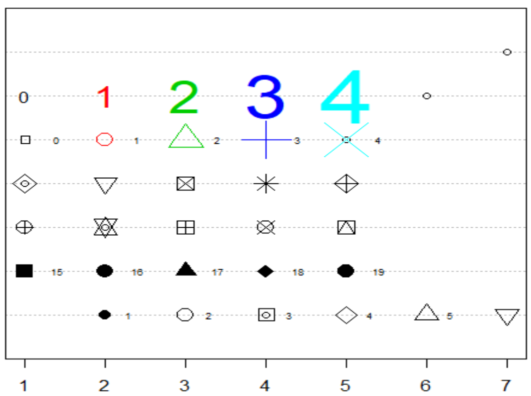
Dotchart

> dotchart(x)



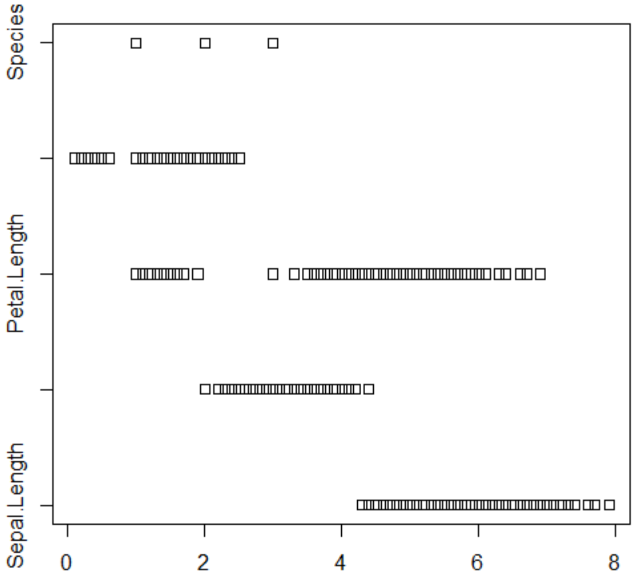
Misc.

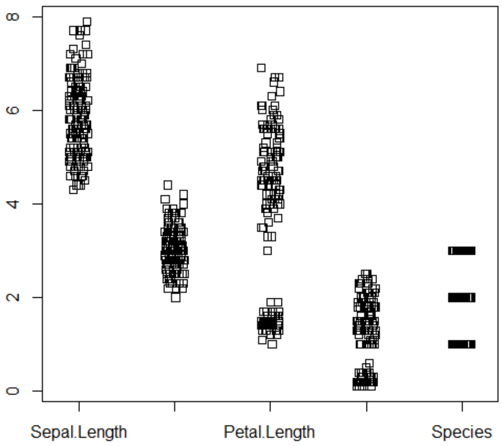
```
> text(1:5, rep(6,5), labels=c(0:4), cex=1:5, col=1:5)

> points(1:5, rep(5,5), cex=1:5, col=1:5, pch=0:4)

> text((1:5)+0.4, rep(5,5), cex=0.6, (0:4))

> points(1:5, rep(4,5), cex=2, pch=(5:9))

> points(1:5, rep(3,5), cex=2, pch=(10:14))

> points(1:5, rep(2,5), cex=2, pch=(15:19))

> text((1:5)+0.4, rep(2,5), cex=0.6, (15:19))

> points((1:6)+0.8+0.2, rep(1,6), cex=2, pch=(20:25))

> text((1:6)+0.8+0.5, rep(1,6), cex=0.6, pch=(20:25))
```



Stripchart

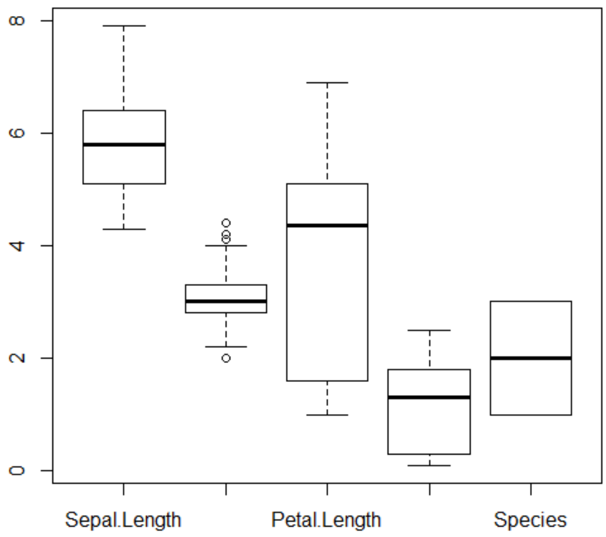
```
> stripchart(iris)

> stripchart(iris, method="jitter")

> stripchart(iris, method="jitter", vertical=TRUE)
```


Species
Petal.Length
Sepal.Length
0
2
4
6
8


0
2
4
6
8
Sepal.Length
Petal.Length
Species

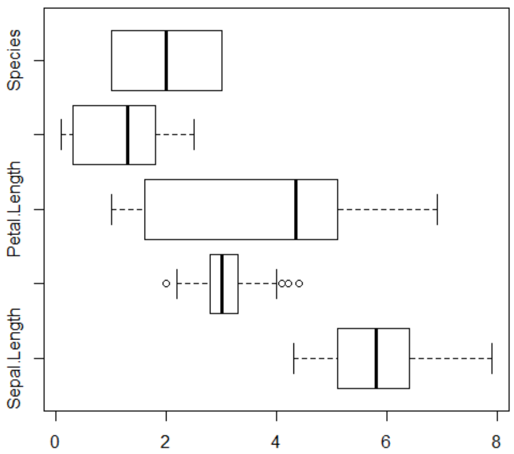
Boxplot

boxplot(iris)

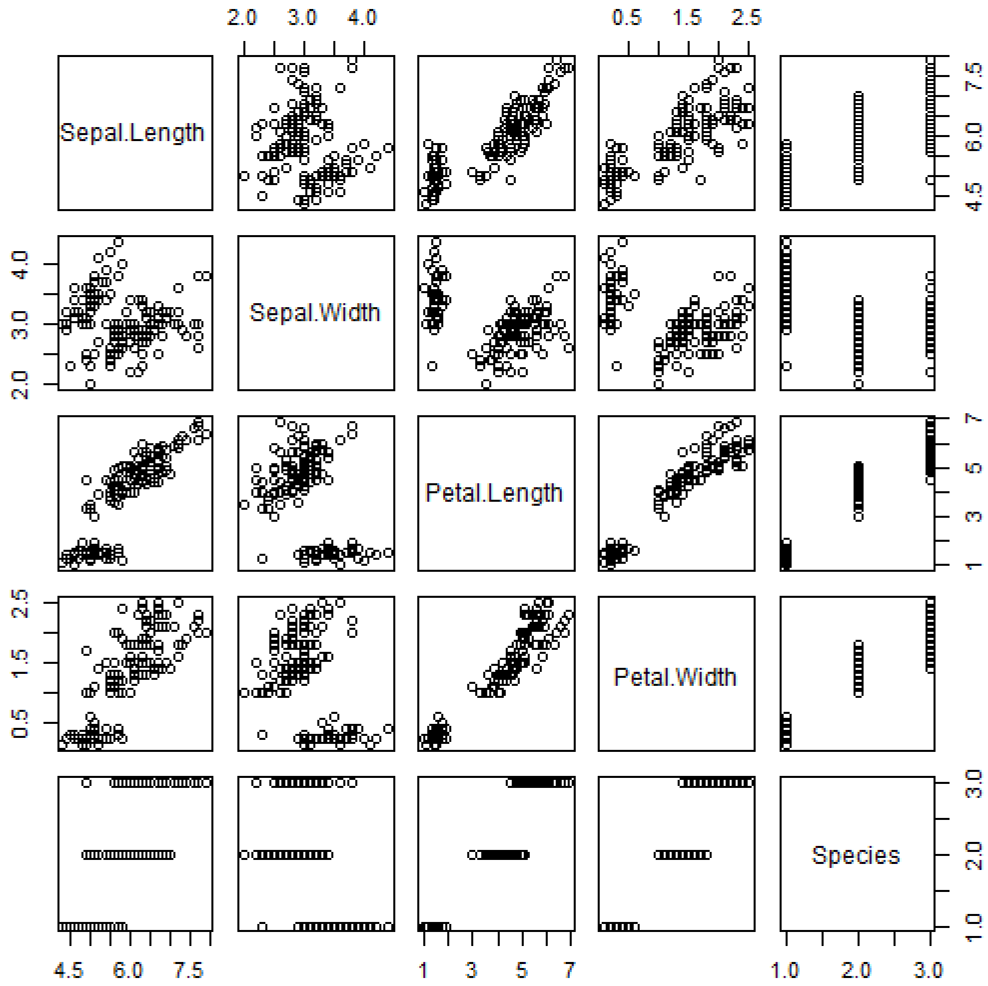


boxplot(iris, horizontal=TRUE)

>

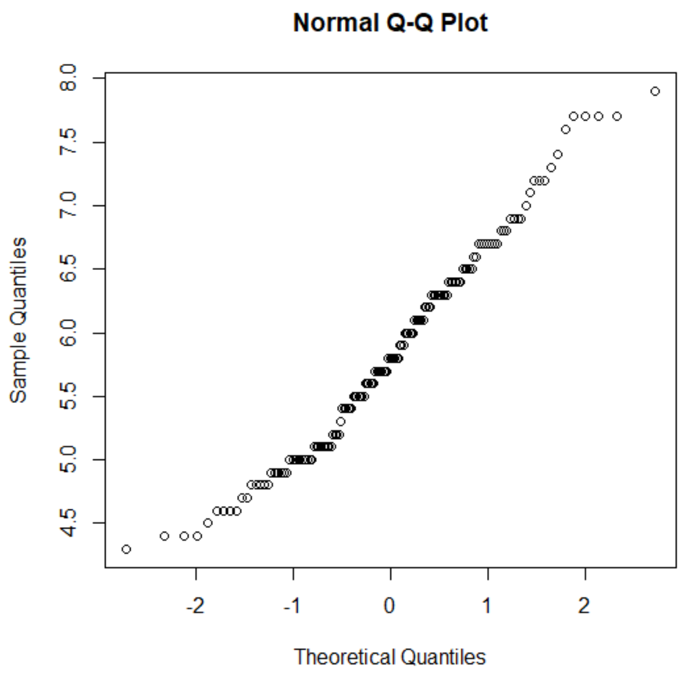


Scatter plot

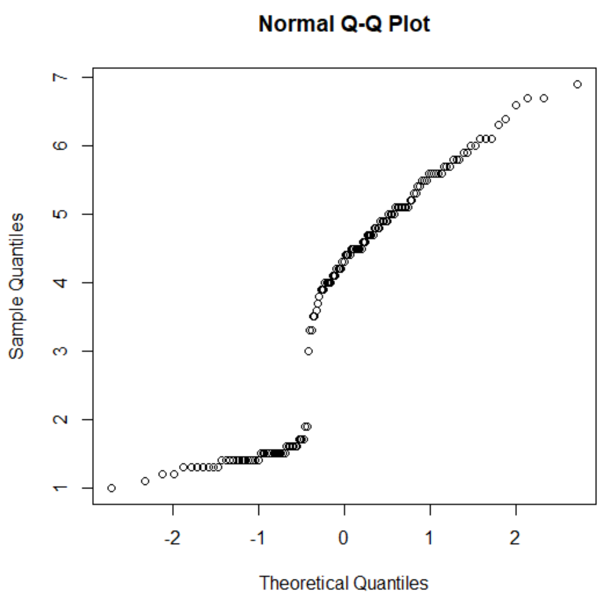


QQPlot

qqnorm(iris$Sepal.Length)



qqnorm(iris$Petal.Length)



LAB ASSESMENT 5

1. Implement kNN(k-nearest neighbors) in R for classification(Consider binary class of predictors of any data sets of your choice)

CODE

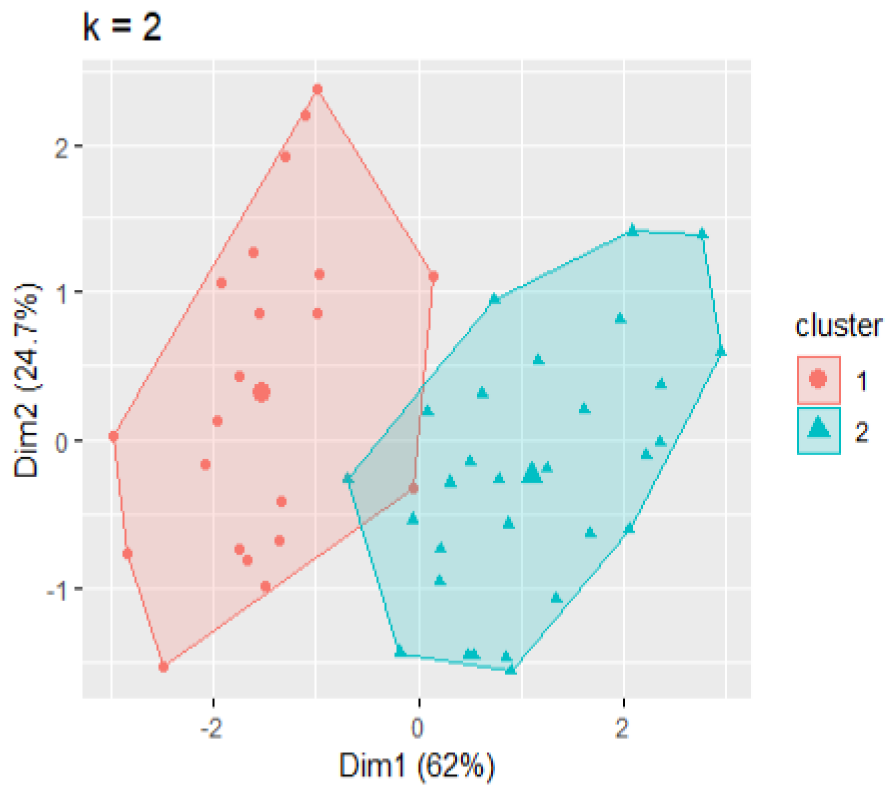
```
> dt=sample(nrow(heart), nrow(heart)*.8)
> heartTrain<- heart[dt,]
> heartTest<-heart[-dt,]
> testlabels=heartTest$target
> trainlabels=heartTrain$target
> pred <- knn(train=heartTrain,test=heartTest,cl=trainlabels,k=10)
> pred
[1] 0 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 1 1 1 1
[30] 0 1 1 1 1 1 1 0 1 0 1 1 1 0 1 0 0 0 1 0 1 0 0 1 0 0 1 1 1
[59] 1 0 0
Levels: 0 1
> accuracy(heartTest$target,pred,threshold=0.5)
threshold AUC omission.rate sensitivity specificity
1 0.5 0.5876344 0.2580645 0.7419355 0.4333333
prop.correct Kappa
1 0.5901639 0.176121
```

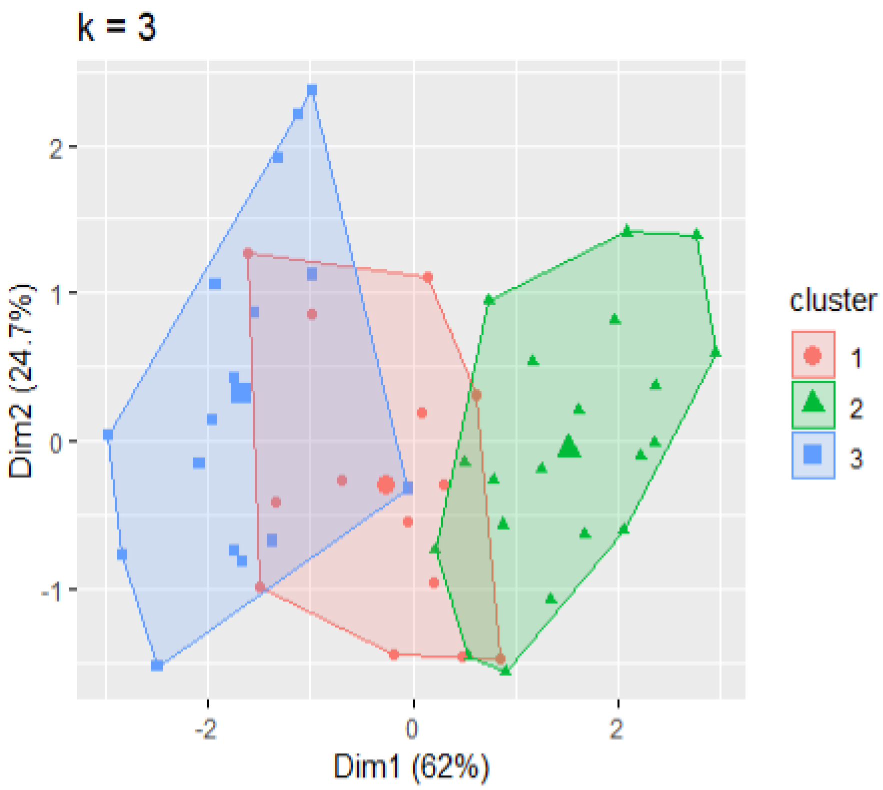
3. Implement K-means clustering

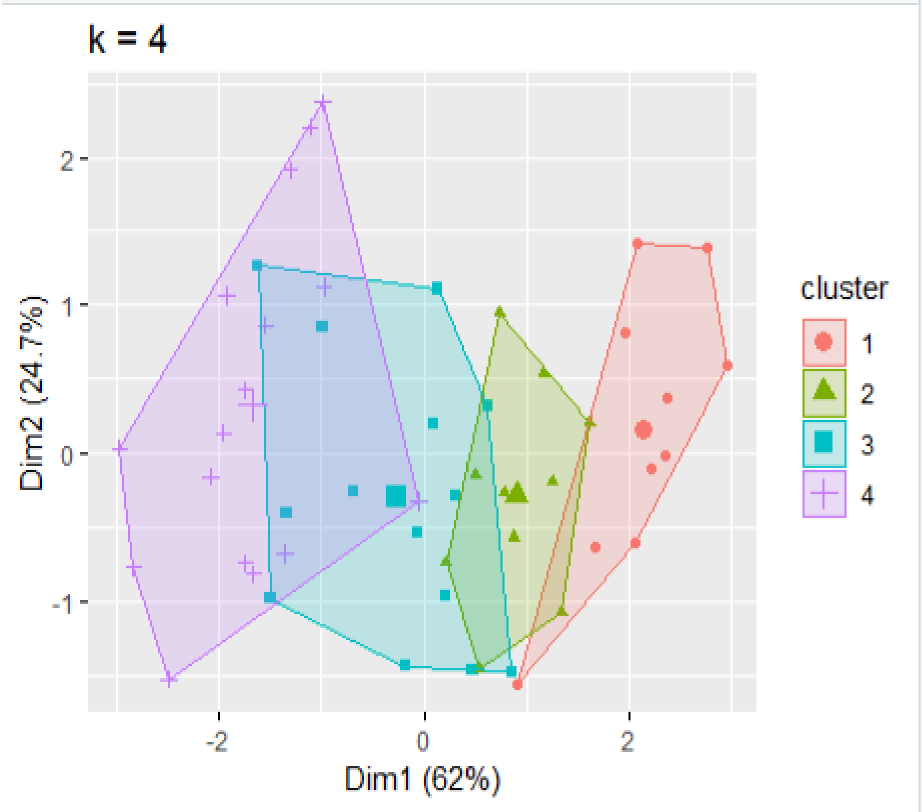
```
library(tidyverse)  # data manipulation

library(cluster)    # clustering algorithms

library(factoextra) # clustering algorithms & visualization

df <- USArrests

df <- na.omit(df)

distance <- get_dist(df)

fviz_dist(distance, gradient = list(low = "#00AFBB", mid = "white", high = "#FC4E07"))

k2 <- kmeans(df, centers = 2, nstart = 25)

str(k2)

k2
```

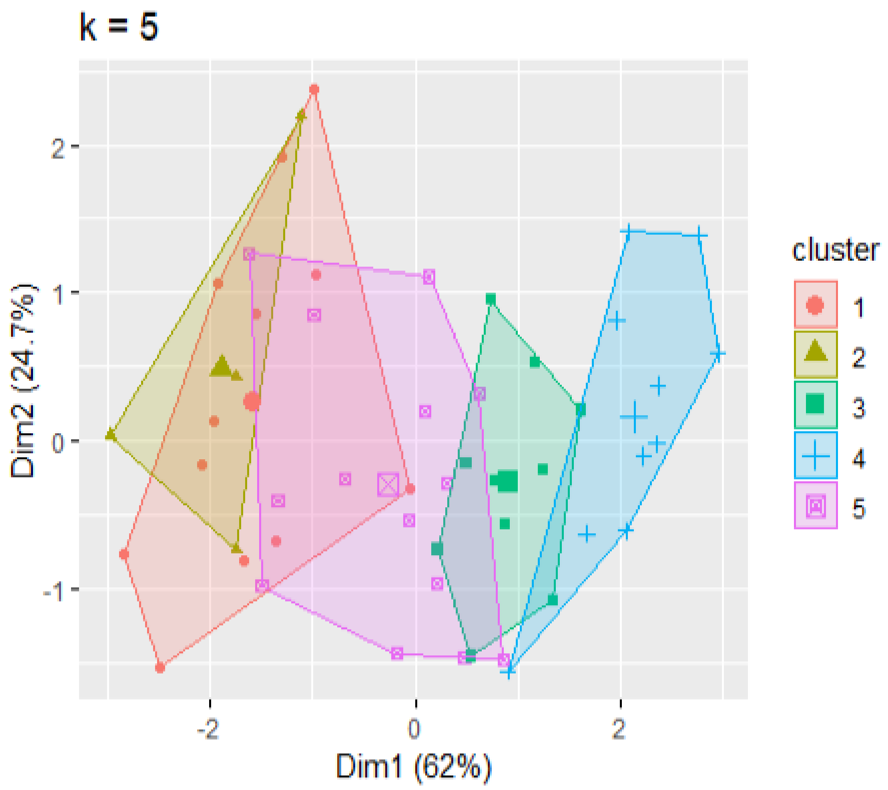
```
fviz_cluster(k2, data = df)
df %>%
  as_tibble() %>%
  mutate(cluster = k2$cluster,
         state = row.names(USArrests)) %>%
  ggplot(aes(UrbanPop, Murder, color = factor(cluster), label = state)) +
  geom_text()
k3 <- kmeans(df, centers = 3, nstart = 25)
k4 <- kmeans(df, centers = 4, nstart = 25)
k5 <- kmeans(df, centers = 5, nstart = 25)

# plots to compare
p1 <- fviz_cluster(k2, geom = "point", data = df) + ggtitle("k = 2")
p2 <- fviz_cluster(k3, geom = "point",  data = df) + ggtitle("k = 3")
p3 <- fviz_cluster(k4, geom = "point",  data = df) + ggtitle("k = 4")
p4 <- fviz_cluster(k5, geom = "point",  data = df) + ggtitle("k = 5")

library(gridExtra)
grid.arrange(p1, p2, p3, p4, nrow = 2)p
```

Output


k = 2
Dim2 (24.7%)
Dim1 (62%)
cluster
1
2


k = 3
Dim2 (24.7%)
Dim1 (62%)
cluster
1
2
3


k = 4
Dim2 (24.7%)
Dim1 (62%)
2
1
0
-1
-2
0
2
cluster
1
2
3
4


k = 5
Dim2 (24.7%)
Dim1 (62%)
2
1
0
-1
-2
0
2
cluster
1
2
3
4
5

4. Spam classification using any Ensemble classifier? Find AUC, ROC, Confusion Matrix, and accuracy?

```
require(caret)
require(tm)
require(wordcloud)
require(e1071)
require(MLmetrics)
rawData <-  SMSSpamCollection
colnames(rawData) <- c("type", "text")

#Converting the text to utf-8 format
rawData$text <- iconv(rawData$text, to = "utf-8")

rawData$type <- factor(rawData$type)
summary(rawData)
table(rawData$type)
prop.table(table(rawData$type)) * 100
set.seed(1234)
trainIndex <- createDataPartition(rawData$type, p = .75,
                                  list = FALSE,
                                  times = 1)
trainData <- rawData[trainIndex,]
testData <- rawData[-trainIndex,]
prop.table(table(trainData$type)) * 100
prop.table(table(testData$type)) * 100
trainData_ham <- trainData[trainData$type == "ham",]
head(trainData_ham$text)
tail(trainData_ham$text)
trainData_spam <- trainData[trainData$type == "spam",]
head(trainData_spam$text)
tail(trainData_spam$text)
trainData_spam <- NULL
trainData_ham <- NULL
corpus <- Corpus(VectorSource(trainData$text))
print(corpus)
corpus[[1]]$content
corpus[[2]]$content
corpus[[50]]$content
corpus[[100]]$content
corpus <- tm_map(corpus, content_transformer(tolower))
corpus <- tm_map(corpus, removeNumbers)
corpus <- tm_map(corpus, removeWords, stopwords())
corpus <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus, stripWhitespace)
corpus[[1]]$content
corpus[[2]]$content
```

```
corpus[[50]]$content
corpus[[100]]$content
pal1 <- brewer.pal(9,"YlGn")
pal1 <- pal1[-(1:4)]

pal2 <- brewer.pal(9,"Reds")
pal2 <- pal2[-(1:4)]
par(mfrow = c(1,2))
wordcloud(corpus[trainData$type == "ham"], min.freq = 40, random.order = FALSE, colors =
pal1)
wordcloud(corpus[trainData$type == "spam"], min.freq = 40, random.order = FALSE, colors =
pal2)
sms_dtm <- DocumentTermMatrix(corpus, control = list(global = c(2, Inf)))
print(sms_dtm)
inspect(sms_dtm[1:10, 5:13])
sms_features <- findFreqTerms(sms_dtm, 5) #find words that appears at least 5 times
summary(sms_features)
head(sms_features)
sms_dtm_train <- DocumentTermMatrix(corpus, list(global = c(2, Inf), dictionary =
sms_features))
print(sms_dtm_train)
convert_counts <- function(x){
 x <- ifelse(x > 0, 1, 0)
 x <- factor(x, levels = c(0,1), labels = c("No", "Yes"))
 return (x)
}
sms_dtm_train <- apply(sms_dtm_train, MARGIN = 2, convert_counts)
sms_classifier <- naiveBayes(sms_dtm_train, trainData$type)

sms_classifier[[2]][1:5]
corpus <- Corpus(VectorSource(testData$text))
corpus <- tm_map(corpus, content_transformer(tolower))
corpus <- tm_map(corpus, removeNumbers)
corpus <- tm_map(corpus, removeWords, stopwords())
corpus <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus, stripWhitespace)

sms_dtm_test <- DocumentTermMatrix(corpus, list(global = c(2, Inf), dictionary =
sms_features))
print(sms_dtm_test)

sms_dtm_test <- apply(sms_dtm_test, MARGIN = 2, convert_counts)
sms_dtm_test[1:10, 5:12]
sms_test_pred <- predict(sms_classifier, sms_dtm_test)
table(testData$type, sms_test_pred)

ConfusionMatrix(sms_test_pred, testData$type)
```

Accuracy(sms_test_pred, testData$type)
F1_Score(sms_test_pred, testData$type)

```
> sms_dtm_test <- apply(sms_dtm_test, MARGIN = 2, convert_counts
)
> sms_dtm_test[1:10, 5:12]
    Terms
Docs like  now   send  std   still weeks word  xxx
  1  "Yes" "Yes" "Yes" "Yes" "Yes" "Yes" "Yes" "Yes"
  2  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
  3  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
  4  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
  5  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
  6  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
  7  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
  8  "Yes" "No"  "No"  "No"  "No"  "No"  "No"  "No"
  9  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
  10 "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
> sms_test_pred <- predict(sms_classifier, sms_dtm_test)
> table(testData$type, sms_test_pred)
      sms_test_pred
       ham spam
  ham  685    1
  spam  19   90
>
> ConfusionMatrix(sms_test_pred, testData$type)
      y_pred
y_true ham spam
  ham  685    1
  spam  19   90
>
> Accuracy(sms_test_pred, testData$type)
[1] 0.9748428
> F1_Score(sms_test_pred, testData$type)
[1] 0.9856115
```

5. Implement polynomial regression and find all the necessary errors (Take any regression data from UCI machine learning repository)(if possible in MS EXCEL; R_2 and RMSE are expected to be calculated as I have demonstrated in both F1 and F2 slot classes)

```
> Position_Salaries <- read.csv("C:/Users/HP/Desktop/Polynomial-
Regression-master/Position_Salaries.csv")
>   View(Position_Salaries)
> # Polynomial Regression
>
> # Importing the dataset
> dataset = Position_Salaries
> dataset = dataset[2:3]
>
> # Splitting the dataset into the Training set and Test set
> # # install.packages('caTools')
> # library(caTools)
> # set.seed(123)
> # split = sample.split(dataset$Salary, SplitRatio = 2/3)
> # training_set = subset(dataset, split == TRUE)
> # test_set = subset(dataset, split == FALSE)
>
> # Feature Scaling
> # training_set = scale(training_set)
```

```
> # test_set = scale(test_set)
>
> # Fitting Linear Regression to the dataset
> lin_reg = lm(formula = Salary ~ .,
+              data = dataset)
>
> # Fitting Polynomial Regression to the dataset
> dataset$Level2 = dataset$Level^2
> dataset$Level3 = dataset$Level^3
> dataset$Level4 = dataset$Level^4
> poly_reg = lm(formula = Salary ~ .,
+               data = dataset)
>
> # Visualising the Linear Regression results
> # install.packages('ggplot2')
> library(ggplot2)
> ggplot() +
+   geom_point(aes(x = dataset$Level, y = dataset$Salary),
+              colour = 'red') +
+   geom_line(aes(x = dataset$Level, y = predict(lin_reg, newdat
a = dataset)),
+             colour = 'blue') +
+   ggtitle('Truth or Bluff (Linear Regression)') +
+   xlab('Level') +
+   ylab('Salary')
>
> # Visualising the Polynomial Regression results
> # install.packages('ggplot2')
> library(ggplot2)
> ggplot() +
+   geom_point(aes(x = dataset$Level, y = dataset$Salary),
+              colour = 'red') +
+   geom_line(aes(x = dataset$Level, y = predict(poly_reg, newda
ta = dataset)),
+             colour = 'blue') +
+   ggtitle('Truth or Bluff (Polynomial Regression)') +
+   xlab('Level') +
+   ylab('Salary')
>
> # Visualising the Regression Model results (for higher resolut
ion and smoother curve)
> # install.packages('ggplot2')
> library(ggplot2)
> x_grid = seq(min(dataset$Level), max(dataset$Level), 0.1)
> ggplot() +
+   geom_point(aes(x = dataset$Level, y = dataset$Salary),
+              colour = 'red') +
+   geom_line(aes(x = x_grid, y = predict(poly_reg,
+                                         newdata = data.frame(L
evel = x_grid,
+                                                              L
evel2 = x_grid^2,
+                                                              L
evel3 = x_grid^3,
+                                                              L
evel4 = x_grid^4))),
+             colour = 'blue') +
+   ggtitle('Truth or Bluff (Polynomial Regression)') +
+   xlab('Level') +
+   ylab('Salary')
>
> # Predicting a new result with Linear Regression
> predict(lin_reg, data.frame(Level = 6.5))
       1
330378.8
>
> # Predicting a new result with Polynomial Regression
> predict(poly_reg, data.frame(Level = 6.5,
+                              Level2 = 6.5^2,
```

```
+                                    Level3 = 6.5^3,
+                                    Level4 = 6.5^4))
       1
158862.5

>
> summary(poly_reg)

Call:
lm(formula = Salary ~ ., data = dataset)

Residuals:
     1      2      3      4      5      6      7      8
 -8357  18240   1358 -14633 -11725   6725  15997  10006
     9     10
-28695  11084

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  184166.7    67768.0   2.718  0.04189 *
Level       -211002.3    76382.2  -2.762  0.03972 *
Level2        94765.4    26454.2   3.582  0.01584 *
Level3       -15463.3     3535.0  -4.374  0.00719 **
Level4          890.2      159.8   5.570  0.00257 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 20510 on 5 degrees of freedom
Multiple R-squared:  0.9974,   Adjusted R-squared:  0.9953
F-statistic: 478.1 on 4 and 5 DF,  p-value: 1.213e-06


>
```

6. Implement PCA with high dimension data set.

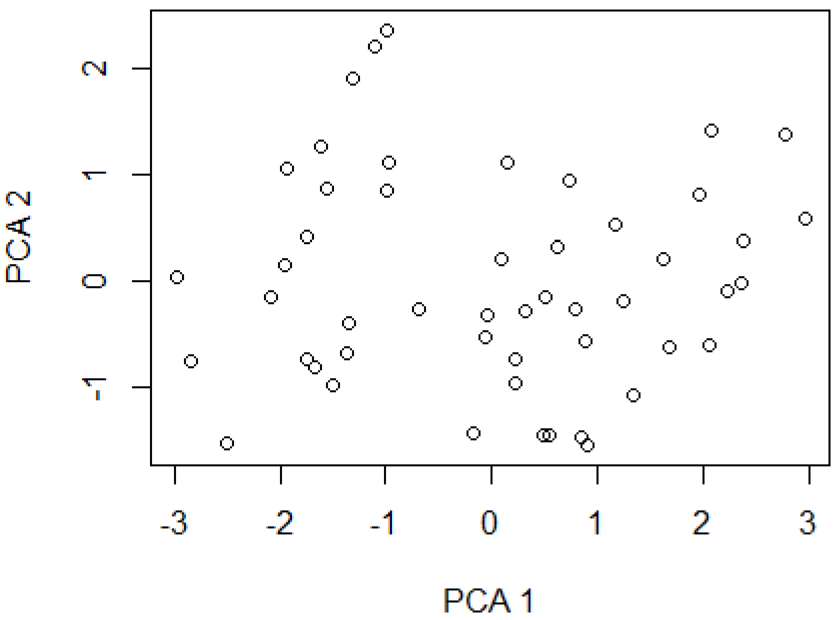
```
> p1 <- princomp(USArrests, cor = TRUE)  ## using correlation ma
trix
> ## p1 <- princomp(USArrests)  ## using covariance matrix
>
> summary(p1)
Importance of components:
                          Comp.1    Comp.2    Comp.3
Standard deviation     1.5748783 0.9948694 0.5971291
Proportion of Variance 0.6200604 0.2474413 0.0891408
Cumulative Proportion  0.6200604 0.8675017 0.9566425
                           Comp.4
Standard deviation     0.41644938
Proportion of Variance 0.04335752
Cumulative Proportion  1.00000000
> loadings(p1)

Loadings:
         Comp.1 Comp.2 Comp.3 Comp.4
Murder    0.536  0.418  0.341  0.649
Assault   0.583  0.188  0.268 -0.743
UrbanPop  0.278 -0.873  0.378  0.134
Rape      0.543 -0.167 -0.818

               Comp.1 Comp.2 Comp.3 Comp.4
SS loadings      1.00   1.00   1.00   1.00
Proportion Var   0.25   0.25   0.25   0.25
Cumulative Var   0.25   0.50   0.75   1.00
> plot(p1)
```

```
> biplot(p1)
> p1$scores
                    Comp.1      Comp.2      Comp.3
Alabama         0.98556588  1.13339238  0.44426879
Alaska          1.95013775  1.07321326 -2.04000333
Arizona         1.76316354 -0.74595678 -0.05478082
Arkansas       -0.14142029  1.11979678 -0.11457369
California      2.52398013 -1.54293399 -0.59855680
Colorado        1.51456286 -0.98755509 -1.09500699
Connecticut    -1.35864746 -1.08892789  0.64325757
Delaware        0.04770931 -0.32535892  0.71863294
Florida         3.01304227  0.03922851  0.57682949
Georgia         1.63928304  1.27894240  0.34246008
Hawaii         -0.91265715 -1.57046001 -0.05078189
Idaho          -1.63979985  0.21097292 -0.25980134
Illinois        1.37891072 -0.68184119  0.67749564
Indiana        -0.50546136 -0.15156254 -0.22805484
Iowa           -2.25364607 -0.10405407 -0.16456432
Kansas         -0.79688112 -0.27016470 -0.02555331
Kentucky       -0.75085907  0.95844029  0.02836942
Louisiana       1.56481798  0.87105466  0.78348036
Maine          -2.39682949  0.37639158  0.06568239
Maryland        1.76336939  0.42765519  0.15725013
Massachusetts  -0.48616629 -1.47449650  0.60949748
Michigan        2.10844115 -0.15539682 -0.38486858
Minnesota      -1.69268181 -0.63226125 -0.15307043
Mississippi     0.99649446  2.39379599  0.74080840
Missouri        0.69678733 -0.26335479 -0.37744383
Montana        -1.18545191  0.53687437 -0.24688932
Nebraska       -1.26563654 -0.19395373 -0.17557391
Nevada          2.87439454 -0.77560020 -1.16338049
New Hampshire  -2.38391541 -0.01808229 -0.03685539
New Jersey      0.18156611 -1.44950571  0.76445355
New Mexico      1.98002375  0.14284878 -0.18369218
New York        1.68257738 -0.82318414  0.64307509
North Carolina  1.12337861  2.22800338  0.86357179
North Dakota   -2.99222562  0.59911882 -0.30127728
Ohio           -0.22596542 -0.74223824  0.03113912
Oklahoma       -0.31178286 -0.28785421  0.01530979
Oregon          0.05912208 -0.54141145 -0.93983298
Pennsylvania   -0.88841582 -0.57110035  0.40062871
Rhode Island   -0.86377206 -1.49197842  1.36994570
South Carolina  1.32072380  1.93340466  0.30053779
South Dakota   -1.98777484  0.82334324 -0.38929333
Tennessee       0.99974168  0.86025130 -0.18808295
Texas           1.35513821 -0.41248082  0.49206886
Utah           -0.55056526 -1.47150461 -0.29372804
Vermont        -2.80141174  1.40228806 -0.84126309
Virginia       -0.09633491  0.19973529 -0.01171254
Washington     -0.21690338 -0.97012418 -0.62487094
West Virginia  -2.10858541  1.42484670 -0.10477467
Wisconsin      -2.07971417 -0.61126862  0.13886500
Wyoming        -0.62942666  0.32101297  0.24065923
                     Comp.4
Alabama         0.156267145
Alaska         -0.438583440
Arizona        -0.834652924
Arkansas       -0.182810896
California     -0.341996478
Colorado        0.001464887
Connecticut    -0.118469414
Delaware       -0.881977637
Florida        -0.096284752
Georgia         1.076796812
Hawaii          0.902806864
Idaho          -0.499104101
Illinois       -0.122021292
Indiana         0.424665700
Iowa            0.017555916
```

```
Kansas          0.206496428
Kentucky        0.670556671
Louisiana       0.454728038
Maine          -0.330459817
Maryland       -0.559069521
Massachusetts  -0.179598963
Michigan        0.102372019
Minnesota       0.067316885
Mississippi     0.215508013
Missouri        0.225824461
Montana         0.123742227
Nebraska        0.015892888
Nevada          0.314515476
New Hampshire  -0.033137338
New Jersey      0.243382700
New Mexico     -0.339533597
New York       -0.013484369
North Carolina -0.954381667
North Dakota   -0.253987327
Ohio            0.473915911
Oklahoma        0.010332321
Oregon         -0.237780688
Pennsylvania    0.359061124
Rhode Island   -0.613569430
South Carolina -0.131466685
South Dakota   -0.109571764
Tennessee       0.652864291
Texas           0.643195491
Utah           -0.082314047
Vermont        -0.144889914
Virginia        0.211370813
Washington     -0.220847793
West Virginia   0.131908831
Wisconsin       0.184103743
Wyoming        -0.166651801
> screeplot(p1) ## identical with plot()
> screeplot(p1, npcs=4, type="lines")
>
> ## Formula interface
> princomp(~ ., data = USArrests, cor = TRUE) ## identical with
princomp(USArrests, cor = TRUE)
Call:
princomp(formula = ~., data = USArrests, cor = TRUE)

Standard deviations:
   Comp.1    Comp.2    Comp.3    Comp.4
1.5748783 0.9948694 0.5971291 0.4164494

 4  variables and  50 observations.
> p2 <- princomp(~ Murder + Assault + UrbanPop, data = USArrests
, cor = TRUE)
> p2$scores
                    Comp.1      Comp.2       Comp.3
Alabama         1.25080553  0.93412066  0.201506279
Alaska          0.80065917  1.39419228 -0.653266658
Arizona         1.35427654 -0.83689479 -0.848878451
Arkansas        0.03474106  1.11766672 -0.187653014
California      1.54281817 -1.51784467 -0.423165230
Colorado        0.52626351 -0.79155189 -0.124964459
Connecticut    -0.99819894 -1.12546259 -0.048898386
Delaware        0.39419522 -0.50665329 -0.804464605
Florida         2.83047216 -0.28976223 -0.052070426
Georgia         1.72618027  1.07670445  1.103637238
Hawaii         -1.11961472 -1.43477571  0.890180117
Idaho          -1.44303362  0.35902584 -0.513469084
Illinois        1.38491701 -0.91063839 -0.061782006
Indiana        -0.58892601 -0.05257008  0.400735395
Iowa           -1.99079223  0.07979666  0.012105645
Kansas         -0.73943810 -0.20172825  0.205767472
```

```
Kentucky       -0.47377123  0.98242635  0.678453680
Louisiana       1.84755969  0.56888502  0.529154745
Maine          -1.88795776  0.49271848 -0.307103791
Maryland        1.66225987  0.25550504 -0.547262233
Massachusetts  -0.35199751 -1.54695578 -0.119741339
Michigan        1.53397296 -0.18913136  0.049753945
Minnesota      -1.61348203 -0.46897279  0.057259253
Mississippi     1.63769688  2.08557281  0.297067826
Missouri        0.33038301 -0.20464547  0.180441559
Montana        -1.02808890  0.65569326  0.106419367
Nebraska       -1.18559433 -0.06662758  0.003536029
Nevada          1.65222989 -0.64885267  0.172936370
New Hampshire  -2.01612318  0.13990369 -0.023957507
New Jersey      0.27000176 -1.59300531  0.313521179
New Mexico      1.60590656  0.05134235 -0.366791164
New York        1.59041067 -1.05629739  0.040286012
North Carolina  1.83529537  1.86454481 -0.855221027
North Dakota   -2.53835810  0.83461860 -0.264827504
Ohio           -0.33126193 -0.70314655  0.472298727
Oklahoma       -0.30579132 -0.26286570  0.012193981
Oregon         -0.51959624 -0.30966821 -0.336835287
Pennsylvania   -0.65922961 -0.58592558  0.400779623
Rhode Island   -0.25891514 -1.72965149 -0.470533931
South Carolina  1.61680635  1.71688057 -0.097614409
South Dakota   -1.70933775  1.01279073 -0.134702980
Tennessee       0.86353518  0.82872633  0.628467325
Texas           1.28038415 -0.58883678  0.681458225
Utah           -0.87426806 -1.32335563 -0.116292344
Vermont        -2.51599180  1.73156383 -0.209565337
Virginia       -0.06101570  0.20657886  0.210474972
Washington     -0.66735028 -0.78238932 -0.287858451
West Virginia  -1.56736097  1.54042823  0.138468455
Wisconsin      -1.78949820 -0.49147321  0.205884014
Wyoming        -0.33677733  0.28999712 -0.135867810
>
>
> ## prcomp()
>
> ## USArrests data vary by orders of magnitude, so scaling is a
ppropriate
> p3 <- prcomp(USArrests, scale = TRUE) ## using correlation mat
rix
> ## p3 <- prcomp(USArrests)  ## using covariance matrix
>
> print(p3)
Standard deviations (1, .., p=4):
[1] 1.5748783 0.9948694 0.5971291 0.4164494

Rotation (n x k) = (4 x 4):
                PC1        PC2        PC3         PC4
Murder   -0.5358995  0.4181809 -0.3412327  0.64922780
Assault  -0.5831836  0.1879856 -0.2681484 -0.74340748
UrbanPop -0.2781909 -0.8728062 -0.3780158  0.13387773
Rape     -0.5434321 -0.1673186  0.8177779  0.08902432
> summary(p3)
Importance of components:
                          PC1    PC2     PC3     PC4
Standard deviation     1.5749 0.9949 0.59713 0.41645
Proportion of Variance 0.6201 0.2474 0.08914 0.04336
Cumulative Proportion  0.6201 0.8675 0.95664 1.00000
> plot(p3) ## Scree plot
> biplot(p3)
>
> ## Formula interface
> p4 <- prcomp(~ Murder + Assault + UrbanPop, data = USArrests,
scale = TRUE)
>
>
>
```

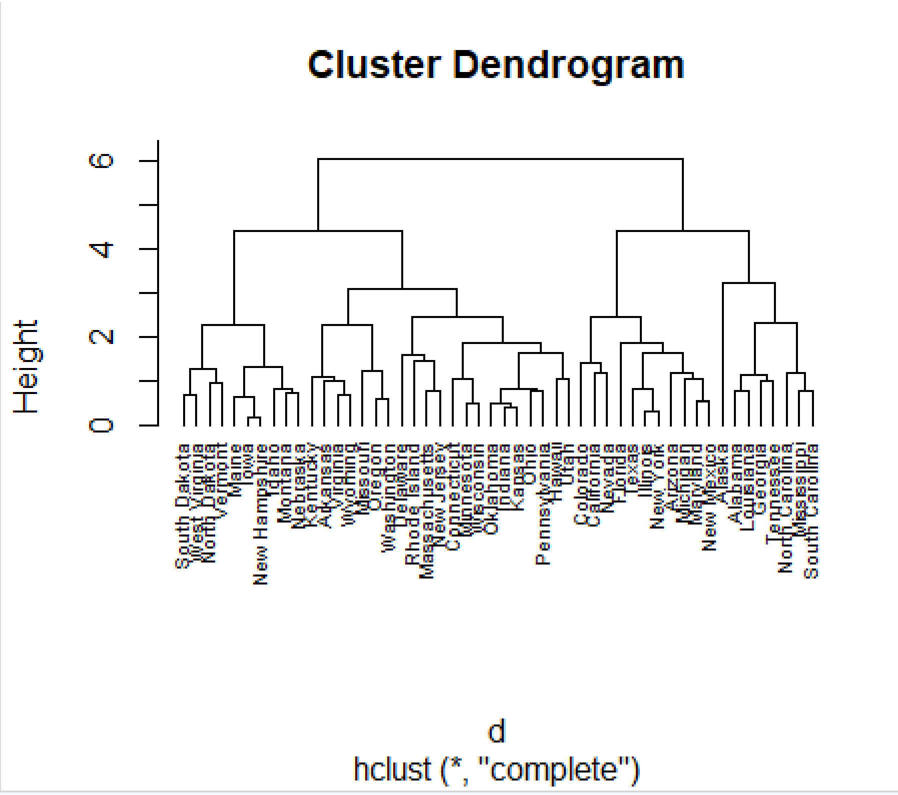
```
>
> ## pca() in “labdsv” package
>
> library(labdsv)  ## You first have to load the LabDSV library.
> p5 <- pca(USArrests, dim=4, cor = TRUE) ## using correlation m
atrix
> ## p5 <-pca(USArrests, dim=4) ## using covariance matrix
>
> summary(p5)
Importance of components:
                            [,1]      [,2]      [,3]
Standard deviation     1.5748783 0.9948694 0.5971291
Proportion of Variance 0.6200604 0.2474413 0.0891408
Cumulative Proportion  0.6200604 0.8675017 0.9566425
                             [,4]
Standard deviation     0.41644938
Proportion of Variance 0.04335752
Cumulative Proportion  1.00000000
> varplot.pca(p5)  ## scree plot and cumulative variances plot
Hit Return to Continue
loadings.pca(p5)
> plot(p5)

>
```



7. Hierarchical clustering with any data set of your choice.

```
> library(tidyverse)  # data manipulation
> library(cluster)    # clustering algorithms
> library(factoextra) # clustering visualization
```

```
> library(dendextend) # for comparing two dendrograms

---------------------
Welcome to dendextend version 1.10.0
Type citation('dendextend') for how to cite the package.

Type browseVignettes(package = 'dendextend') for the package vignette.
The github page is: https://github.com/talgalili/dendextend/

Suggestions and bug-reports can be submitted at: https://github.com/talgali
li/dendextend/issues
Or contact: <tal.galili@gmail.com>

        To suppress this message use:  suppressPackageStartupMessages(libra
ry(dendextend))
---------------------


Attaching package: ‘dendextend’

The following object is masked from ‘package:stats’:

    cutree

Warning message:
package ‘dendextend’ was built under R version 3.5.3
> df <- USArrests
> df <- USArrests
> df <- scale(df)
> head(df)
               Murder   Assault   UrbanPop         Rape
Alabama    1.24256408 0.7828393 -0.5209066 -0.003416473
Alaska     0.50786248 1.1068225 -1.2117642  2.484202941
Arizona    0.07163341 1.4788032  0.9989801  1.042878388
Arkansas   0.23234938 0.2308680 -1.0735927 -0.184916602
California 0.27826823 1.2628144  1.7589234  2.067820292
Colorado   0.02571456 0.3988593  0.8608085  1.864967207
> d <- dist(df, method = "euclidean")
> hc1 <- hclust(d, method = "complete" )
> plot(hc1, cex = 0.6, hang = -1)
> hc2 <- agnes(df, method = "complete")
> hc2$ac
[1] 0.8531583
```



8. Mention one catchy and contemporary problem statement that can be solved by machine learning with 10 sentences (No coding/ program is required only problem statement and technical requirements are to be written)

Image recognition (Computer Vision)

Computer vision produces numerical or symbolic information from images and high-dimensional data. It involves machine learning, data mining, database knowledge discovery and pattern recognition. Potential business uses of image recognition technology are found in healthcare, automobiles – driverless cars, marketing campaigns, etc. Baidu has developed a prototype of DuLight for visually impaired which incorporates computer vision technology to capture surrounding and narrate the interpretation through an earpiece. Image recognition based marketing campaigns such as Makeup Genius by L'Oreal drive social sharing and user engagement.

1. Image classification

The problem of image classification goes like this: Given a set of images that are all labeled with a single category, we're asked to predict these categories for a novel set of test images and measure the accuracy of the predictions. There are a variety of challenges associated with this task, including viewpoint variation, scale variation, intra-class variation, image deformation, image occlusion, illumination conditions, and background clutter.

2. Object detection

The task to define objects within images usually involves outputting bounding boxes and labels for individual objects. This differs from the classification / localization task by applying classification and localization to many objects instead of just a single dominant object. You only have 2 classes of object classification, which means object bounding boxes and non-object bounding boxes. For example, in car detection, you have to detect all cars in a given image with their bounding boxes.
If we use the Sliding Window technique like the way we classify and localize images, we need to apply a CNN to many different crops of the image. Because CNN classifies each crop as object or background, we need to apply CNN to huge numbers of locations and scales, which is very computationally expensive

3. Object tracking

Object Tracking refers to the process of following a specific object of interest, or multiple objects, in a given scene. It traditionally has applications in video and real-world interactions where observations are made following an initial object detection. Now, it's crucial to autonomous driving systems such as self-driving vehicles from companies like Uber and Tesla.Object Tracking methods can be divided into 2 categories according to the observation model: generative method and discriminative method. The generative method uses the generative model to describe the apparent characteristics and minimizes the reconstruction error to search the object, such as PCA. There are 2 kinds of basic network models that can be used: **stacked auto encoders (SAE)** and **convolutional neural network (CNN).**