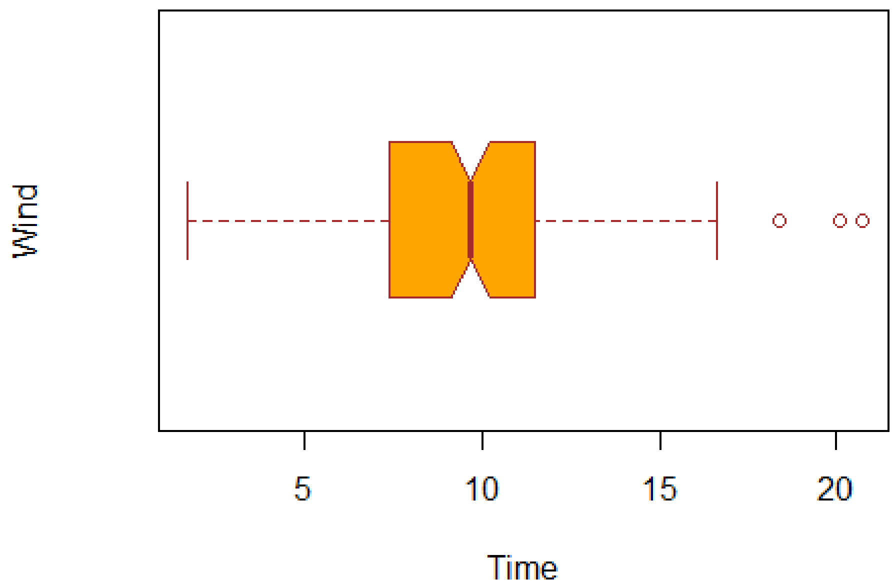
```
#Question 1
data(airquality)
head(airquality)

##   Ozone Solar.R Wind Temp Month Day
## 1    41     190  7.4   67     5   1
## 2    36     118  8.0   72     5   2
## 3    12     149 12.6   74     5   3
## 4    18     313 11.5   62     5   4
## 5    NA      NA 14.3   56     5   5
## 6    28      NA 14.9   66     5   6

colnames(airquality)

## [1] "Ozone"   "Solar.R" "Wind"    "Temp"    "Month"   "Day"

airq=airquality
#Replacing NA values
airq$Solar.R[which(is.na(airq$Solar.R))]=mean(airq$Solar.R,na.rm=TRUE)
airq$Ozone[which(is.na(airq$Ozone))]=mean(airq$Ozone,na.rm=TRUE)
#Central,variational measures
mean(airq$Ozone)

## [1] 42.12931

sd(airq$Ozone)

## [1] 28.69337

median(airq$Ozone)

## [1] 42.12931

mean(airq$Solar.R)

## [1] 185.9315

sd(airq$Solar.R)

## [1] 87.96027

median(airq$Ozone)

## [1] 42.12931

mean(airq$Temp)

## [1] 77.88235

sd(airq$Temp)

## [1] 9.46527

median(airq$Temp)
```

```
## [1] 79

mean(airq$Wind)

## [1] 9.957516

sd(airq$Wind)

## [1] 3.523001

median(airq$Wind)

## [1] 9.7

help(airquality)

## starting httpd help server ... done

#Boxplot
boxplot(airq$Wind,main = "Average wind speed in miles per hour at 0700 and 10
00 hours at LaGuardia Airport",
        xlab = "Time",
        ylab = "Wind",
        col = "orange",
        border = "brown",
        horizontal = TRUE,
        notch = TRUE)
tail(sort(airq$Wind), 3)

## [1] 18.4 20.1 20.7

#Question 2
library(caret)

## Loading required package: lattice
## Loading required package: ggplot2
```

I speed in miles per hour at 0700 and 1000 hours at L




```
library(e1071)
df <- read.csv("http://users.stat.ufl.edu/~winner/data/nfl2008_fga.csv")


head(df)

##   GameDate AwayTeam HomeTeam qtr min sec kickteam def down togo kicker ydl
ine
## 1 20081130      IND      CLE   1  47   2      IND CLE    4   11     15
12
## 2 20081005      IND      HOU   1  54  47      IND HOU    4    3     15
28
## 3 20081228      TEN      IND   1  45  20      IND TEN    4    3     15
10
## 4 20081012      BAL      IND   1  45  42      IND BAL    4    1     15
19
## 5 20080907      CHI      IND   1  50  56      IND CHI    4   21     15
21
## 6 20081116      HOU      IND   1  50  43      IND HOU    4    7     15
22
##          name distance homekick kickdiff timerem offscore defscore season
GOOD
## 1 A.Vinatieri       30        0       -3    2822        0        3   2008
1
## 2 A.Vinatieri       46        0        0    3287        0        0   2008
1
## 3 A.Vinatieri       28        1        7    2720        7        0   2008
```

```
1
## 4 A.Vinatieri       37         1        14    2742       14         0   2008
1
## 5 A.Vinatieri       39         1         0    3056        0         0   2008
1
## 6 A.Vinatieri       40         1        -3    3043        0         3   2008
1
##   Missed Blocked
## 1      0       0
## 2      0       0
## 3      0       0
## 4      0       0
## 5      0       0
## 6      0       0
```

```
names(df)
```

```
##  [1] "GameDate" "AwayTeam" "HomeTeam" "qtr"      "min"      "sec"
##  [7] "kickteam" "def"      "down"     "togo"     "kicker"   "ydline"
## [13] "name"     "distance" "homekick" "kickdiff" "timerem"  "offscore"
## [19] "defscore" "season"   "GOOD"     "Missed"   "Blocked"
```

```
str(df)
```

```
## 'data.frame':    1039 obs. of  23 variables:
##  $ GameDate: int  20081130 20081005 20081228 20081012 20080907 20081116 20
081123 20081207 20081130 20090118 ...
##  $ AwayTeam: Factor w/ 32 levels "ARI","ATL","BAL",..: 14 14 31 3 6 13 14
16 16 24 ...
##  $ HomeTeam: Factor w/ 32 levels "ARI","ATL","BAL",..: 8 13 14 14 14 14 26
10 23 1 ...
##  $ qtr     : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ min     : int  47 54 45 45 50 50 46 52 46 49 ...
##  $ sec     : int  2 47 20 42 56 43 45 34 12 46 ...
##  $ kickteam: Factor w/ 32 levels "ARI","ATL","BAL",..: 14 14 14 14 14 14 1
4 16 16 24 ...
##  $ def     : Factor w/ 32 levels "ARI","ATL","BAL",..: 8 13 31 3 6 13 26 1
0 23 1 ...
##  $ down    : int  4 4 4 4 4 4 4 4 4 4 ...
##  $ togo    : int  11 3 3 1 21 7 5 7 7 9 ...
##  $ kicker  : int  15 15 15 15 15 15 15 18 18 29 ...
##  $ ydline  : int  12 28 10 19 21 22 5 8 20 27 ...
##  $ name    : Factor w/ 39 levels "A.Vinatieri",..: 1 1 1 1 1 1 1 2 2 3 ...
##  $ distance: int  30 46 28 37 39 40 23 26 38 45 ...
##  $ homekick: int  0 0 1 1 1 1 0 0 0 0 ...
##  $ kickdiff: int  -3 0 7 14 0 -3 0 0 -3 -7 ...
##  $ timerem : int  2822 3287 2720 2742 3056 3043 2805 3154 2772 2986 ...
##  $ offscore: int  0 0 7 14 0 0 0 0 0 0 ...
##  $ defscore: int  3 0 0 0 0 3 0 0 3 7 ...
##  $ season  : int  2008 2008 2008 2008 2008 2008 2008 2008 2008 2008 ...
##  $ GOOD    : int  1 1 1 1 1 1 1 1 1 1 ...
```

```
##  $ Missed  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Blocked : int  0 0 0 0 0 0 0 0 0 0 ...

# (a) Train the model using 80% of this dataset and suggest an
# appropriate GLM to model homekick to togo, ydline and kicker
# variables.

df <- na.omit(df)

# Split the dataset into 80 train and 20 test.
n <- nrow(df)
indexes <- sample(n,n*(80/100))
trainset <- df[indexes,]
testset <- df[-indexes,]


# To determine what #type of model to use we need to look
# at the ourcome variable homekick
str(df$homekick)

##  int [1:1037] 0 0 1 1 1 1 0 0 0 0 ...

max(df$homekick) #1

## [1] 1

min(df$homekick) #0

## [1] 0

# As the outcome is binary, we create a glm model using binomial
# family
model <- glm(homekick ~ togo + ydline + kicker, data = trainset, family = 'bi
nomial')

# (b) Specify the significant variables on homekick at the level
# of ????=0.05, and estimate the parameters of your model

summary(model)

##
## Call:
## glm(formula = homekick ~ togo + ydline + kicker, family = "binomial",
##     data = trainset)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3942  -1.1672  -0.9659   1.1731   1.3866
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept)  0.031754   0.208305   0.152   0.8788
## togo        -0.035086   0.017553  -1.999   0.0456 *
## ydline       0.014969   0.007627   1.963   0.0497 *
## kicker      -0.003967   0.006183  -0.642   0.5211
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1149.2  on 828  degrees of freedom
## Residual deviance: 1142.7  on 825  degrees of freedom
## AIC: 1150.7
##
## Number of Fisher Scoring iterations: 3
```

```r
# at alpha of 0.05 togo,ydline is significant
# Using this we can estimate the parameters as
# Intercept = 0 (as non-sig), beta for togo is - 0.033084
#beta for ydline is 0.012182

# homekick =  0 - 0.033084 * togo
#homekick=0+0.012182*ydline




# Let's rerun the model removing the other non-signifcant items

model2 <- glm(homekick ~ togo+ydline, data=trainset, family='binomial')
summary(model2)
```

```
##
## Call:
## glm(formula = homekick ~ togo + ydline, family = "binomial",
##     data = trainset)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.362  -1.167  -0.978   1.173   1.382
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.044112   0.171456  -0.257   0.7970
## togo        -0.035195   0.017548  -2.006   0.0449 *
## ydline       0.014979   0.007626   1.964   0.0495 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1149.2  on 828  degrees of freedom
```

```
## Residual deviance: 1143.1  on 826  degrees of freedom
## AIC: 1149.1
##
## Number of Fisher Scoring iterations: 3
```

```
#Togo and ydline are still significant.


# (c) Predict the test dataset using the trained model.

predicted_data <- predict(model2,testset, type='response')
# convert that to 1s and 0s
p_data <- as.integer(predicted_data > 0.5)


# Confusion matrix
confusionMatrix(data = as.factor(p_data),
                reference = as.factor(testset$homekick))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 58 46
##          1 52 52
##
##                Accuracy : 0.5288
##                  95% CI : (0.4586, 0.5982)
##     No Information Rate : 0.5288
##     P-Value [Acc > NIR] : 0.5282
##
##                   Kappa : 0.0577
##
##  Mcnemar's Test P-Value : 0.6135
##
##             Sensitivity : 0.5273
##             Specificity : 0.5306
##          Pos Pred Value : 0.5577
##          Neg Pred Value : 0.5000
##              Prevalence : 0.5288
##          Detection Rate : 0.2788
##    Detection Prevalence : 0.5000
##       Balanced Accuracy : 0.5289
##
##        'Positive' Class : 0
##
```

```
tab <- table(p_data, testset$homekick) # Confusion matrix
tab
```

```
##
## p_data  0  1
##      0 58 46
##      1 52 52
```

```r
model_accuracy <- sum( tab[row(tab) == col(tab)] ) / sum(tab)
model_accuracy
```

```
## [1] 0.5288462
```

```r
#3
pacman::p_load('psych', 'caret', 'e1071')
pacman::p_load('quantmod', 'xts', 'ggplot2', 'tseries', 'forecast')
data=getSymbols("GAW.L",src="yahoo",from = "2012-01-01", to = "2019-07-11",
                auto.assign = FALSE)
```

```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.
```

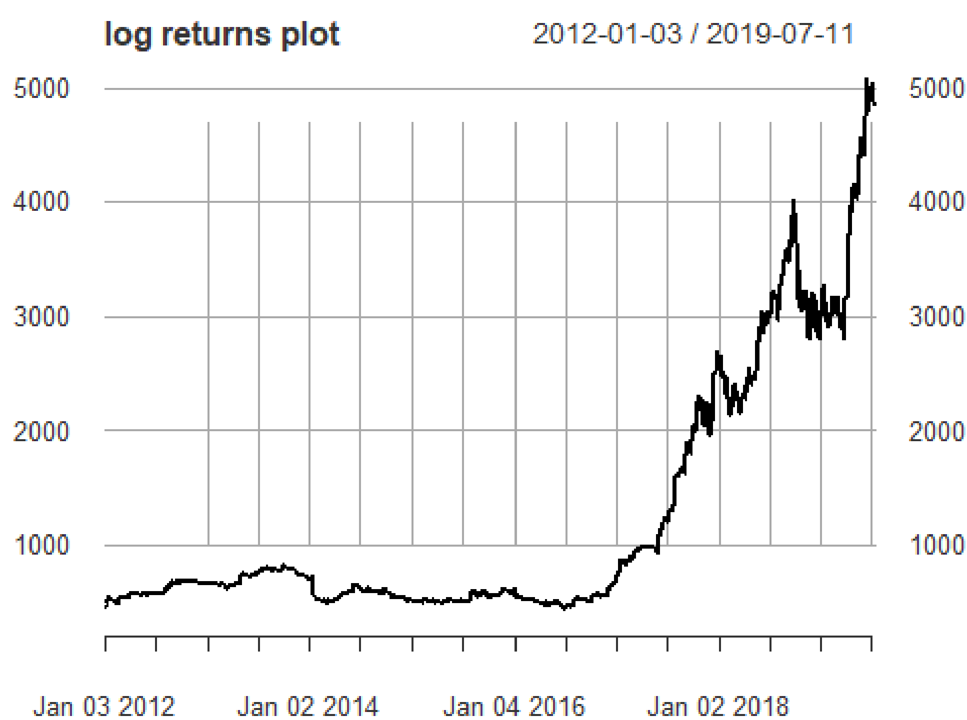
```
## Warning: GAW.L contains missing values. Some functions will not work if ob
jects
## contain missing values in the middle of the series. Consider using na.omit
(),
## na.approx(), na.fill(), etc to remove or replace them.
```

```
## Warning: 'indexClass<-' is deprecated.
## Use 'tclass<-' instead.
## See help("Deprecated") and help("xts-deprecated").
```

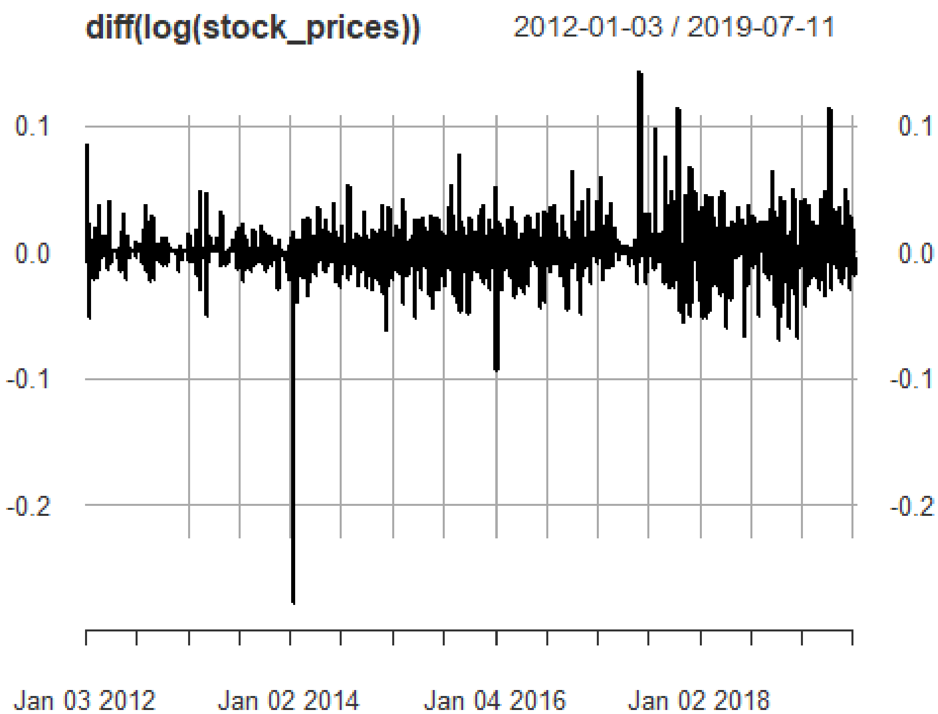
```r
head(data)
```

```
##            GAW.L.Open GAW.L.High GAW.L.Low GAW.L.Close GAW.L.Volume
## 2012-01-03     453.96        458    444.00         450         6525
## 2012-01-04     450.00        458    450.00         450         5016
## 2012-01-05     480.00        505    480.00         490        30451
## 2012-01-06     497.50        530    497.50         515        15985
## 2012-01-09     525.00        535    525.00         525        19048
## 2012-01-10     535.00        535    521.25         520        11803
##            GAW.L.Adjusted
## 2012-01-03       252.5661
## 2012-01-04       252.5661
## 2012-01-05       275.0164
## 2012-01-06       289.0479
## 2012-01-09       294.6605
## 2012-01-10       291.8541
```

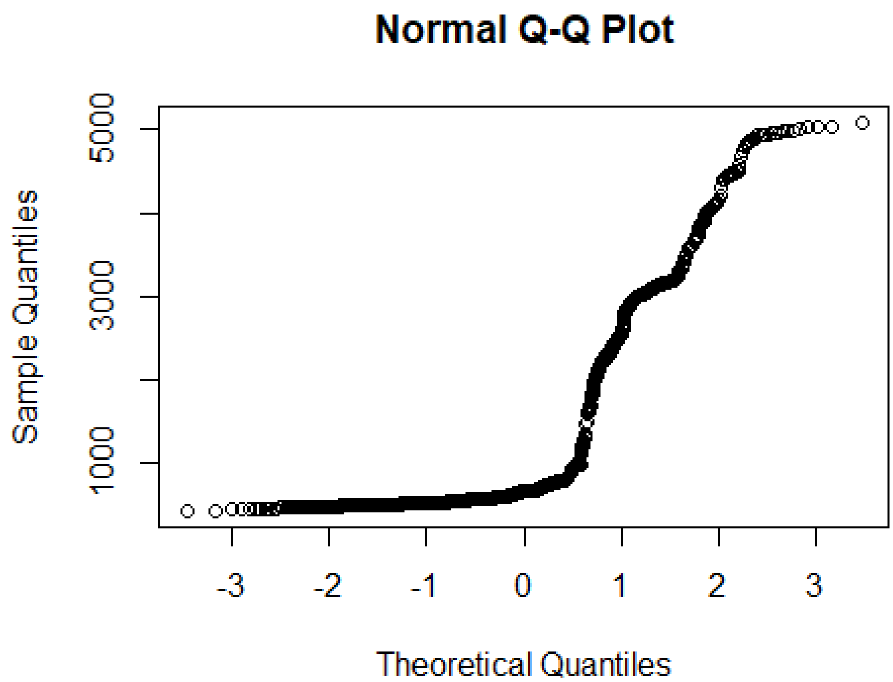
```r
dim(data)

## [1] 1901    6

str(data)

## An 'xts' object on 2012-01-03/2019-07-11 containing:
##   Data: num [1:1901, 1:6] 454 450 480 498 525 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:6] "GAW.L.Open" "GAW.L.High" "GAW.L.Low" "GAW.L.Close" ...
##   Indexed by objects of class: [Date] TZ: UTC
##   xts Attributes:
## List of 2
##  $ src    : chr "yahoo"
##  $ updated: POSIXct[1:1], format: "2020-03-06 09:37:07"

data=na.omit(data)
# Remove all but the closing price
stock_prices <- data[,4]

# (a) Validate the assumptions using graphical visualization.

# plot the data
plot(stock_prices, type = 'l', main = 'log returns plot')
```



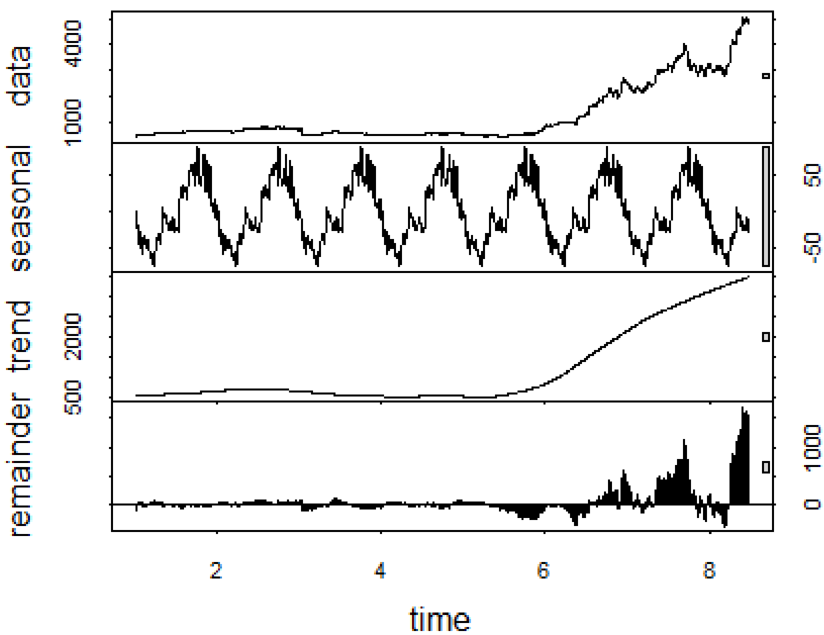
```r
# From the plot we can see the mean and variance is not stationary
# This is especially true when we compare to a normalised version
plot(diff(log(stock_prices)))
```




```r
# We can also see from the qqnorm plot that it is not normal
qqnorm(stock_prices, main = "Normal Q-Q Plot",
       xlab = "Theoretical Quantiles",
       ylab = "Sample Quantiles", plot.it = TRUE)
```

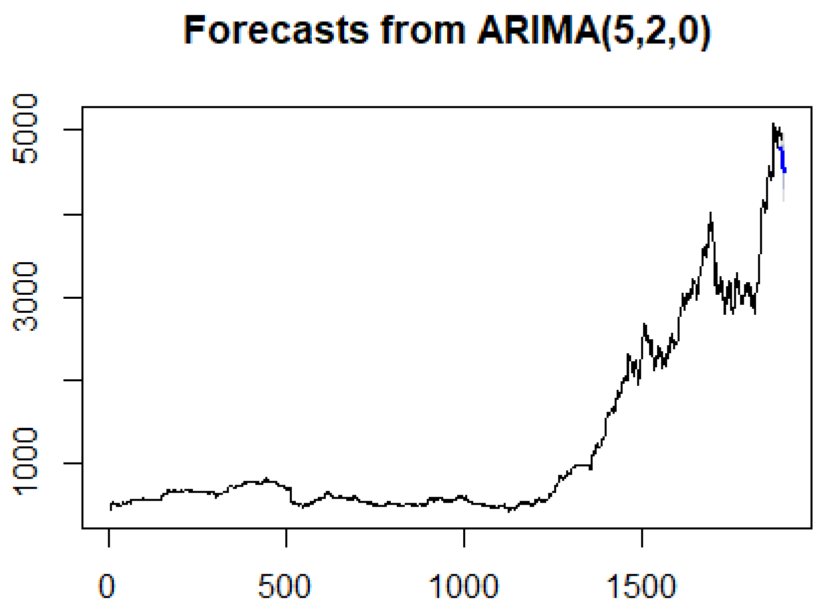



```
periodicity(stock_prices)

## Daily periodicity from 2012-01-03 to 2019-07-11

# Daily periodicity from 2012-01-03 to 2019-07-11
# Adjust it to have yearly frequency, there are ~253 trading days in the year
yearly_stock_prices <- ts(as.numeric(stock_prices), frequency = 253)
fit <- stl(yearly_stock_prices, s.window = "periodic", robust = TRUE)
plot(fit)
```




```
# it seems there is no seasonal apparent as the data is very similar to the t
rend
# meaning the seasonal didn't account for much and the remainder is quite sim
ilar
# to the original data also, suggesting the forced seasonal trend accounts fo
r very little

# (b) Fit the optimized model for 'close price' and provide
#     the coefficient estimates for the fitted model.

auto.fit <- auto.arima(stock_prices, seasonal = TRUE)
auto.fit

## Series: stock_prices
## ARIMA(5,2,0)
##
## Coefficients:
##           ar1      ar2      ar3      ar4      ar5
##       -0.8653  -0.6398  -0.4684  -0.3841  -0.2153
## s.e.   0.0225   0.0287   0.0304   0.0287   0.0224
##
## sigma^2 estimated as 1439:  log likelihood=-9566.58
## AIC=19145.16   AICc=19145.2   BIC=19178.43

# (c) What is the estimated order for AR and MA?

# The estimated order is ARIMA(5,2,0)
```

```
# p = 5
# d = 2
# q = 0


# (d) Forecast h=10 step ahead prediction of close price on the plot
#     of the original time series.

auto.fcast <- forecast(auto.fit, h =  10)
plot(auto.fcast)
```




```
#Question 4
pacman::p_load('psych', 'MASS', 'car', 'ggplot2', 'GGally', 'CCA', 'sem', 'cl
uster')


# Question 4
# Use dataset available on
# http://users.stat.ufl.edu/~winner/data/nfl2008_fga.csv

df <- read.csv('http://users.stat.ufl.edu/~winner/data/nfl2008_fga.csv')
names(df)

##  [1] "GameDate" "AwayTeam" "HomeTeam" "qtr"      "min"      "sec"
##  [7] "kickteam" "def"      "down"     "togo"     "kicker"   "ydline"
```

```
## [13] "name"     "distance" "homekick" "kickdiff" "timerem"  "offscore"
## [19] "defscore" "season"   "GOOD"     "Missed"   "Blocked"
```

```r
head(df)
```

```
##   GameDate AwayTeam HomeTeam qtr min sec kickteam def down togo kicker ydl
ine
## 1 20081130      IND      CLE   1  47   2      IND CLE    4   11     15
12
## 2 20081005      IND      HOU   1  54  47      IND HOU    4    3     15
28
## 3 20081228      TEN      IND   1  45  20      IND TEN    4    3     15
10
## 4 20081012      BAL      IND   1  45  42      IND BAL    4    1     15
19
## 5 20080907      CHI      IND   1  50  56      IND CHI    4   21     15
21
## 6 20081116      HOU      IND   1  50  43      IND HOU    4    7     15
22
##          name distance homekick kickdiff timerem offscore defscore season
GOOD
## 1 A.Vinatieri       30        0       -3    2822        0        3   2008
1
## 2 A.Vinatieri       46        0        0    3287        0        0   2008
1
## 3 A.Vinatieri       28        1        7    2720        7        0   2008
1
## 4 A.Vinatieri       37        1       14    2742       14        0   2008
1
## 5 A.Vinatieri       39        1        0    3056        0        0   2008
1
## 6 A.Vinatieri       40        1       -3    3043        0        3   2008
1
##   Missed Blocked
## 1      0       0
## 2      0       0
## 3      0       0
## 4      0       0
## 5      0       0
## 6      0       0
```

```r
describe(df)
```

```
## [1]  23893     NA     NA     NA     NA     NA     NA     NA -23870
```

```r
str(df)
```

```
## 'data.frame':    1039 obs. of  23 variables:
##  $ GameDate: int  20081130 20081005 20081228 20081012 20080907 20081116 20
081123 20081207 20081130 20090118 ...
##  $ AwayTeam: Factor w/ 32 levels "ARI","ATL","BAL",..: 14 14 31 3 6 13 14
```

```
16 16 24 ...
##  $ HomeTeam: Factor w/ 32 levels "ARI","ATL","BAL",..: 8 13 14 14 14 14 26
10 23 1 ...
##  $ qtr     : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ min     : int  47 54 45 45 50 50 46 52 46 49 ...
##  $ sec     : int  2 47 20 42 56 43 45 34 12 46 ...
##  $ kickteam: Factor w/ 32 levels "ARI","ATL","BAL",..: 14 14 14 14 14 14 1
4 16 16 24 ...
##  $ def     : Factor w/ 32 levels "ARI","ATL","BAL",..: 8 13 31 3 6 13 26 1
0 23 1 ...
##  $ down    : int  4 4 4 4 4 4 4 4 4 4 ...
##  $ togo    : int  11 3 3 1 21 7 5 7 7 9 ...
##  $ kicker  : int  15 15 15 15 15 15 15 18 18 29 ...
##  $ ydline  : int  12 28 10 19 21 22 5 8 20 27 ...
##  $ name    : Factor w/ 39 levels "A.Vinatieri",..: 1 1 1 1 1 1 1 2 2 3 ...
##  $ distance: int  30 46 28 37 39 40 23 26 38 45 ...
##  $ homekick: int  0 0 1 1 1 1 0 0 0 0 ...
##  $ kickdiff: int  -3 0 7 14 0 -3 0 0 -3 -7 ...
##  $ timerem : int  2822 3287 2720 2742 3056 3043 2805 3154 2772 2986 ...
##  $ offscore: int  0 0 7 14 0 0 0 0 0 0 ...
##  $ defscore: int  3 0 0 0 0 3 0 0 3 7 ...
##  $ season  : int  2008 2008 2008 2008 2008 2008 2008 2008 2008 2008 ...
##  $ GOOD    : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Missed  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Blocked : int  0 0 0 0 0 0 0 0 0 0 ...

#There are two rows incomplete
df[!complete.cases(data),]

##  [1] GameDate AwayTeam HomeTeam qtr      min      sec      kickteam def
##  [9] down     togo     kicker   ydline   name     distance homekick kickdi
ff
## [17] timerem  offscore defscore season   GOOD     Missed   Blocked
## <0 rows> (or 0-length row.names)

# As this is only 2 out of 1039 observations, we choose to exclude these
data <- df[complete.cases(df),]

# 1. Use LDA to classify the dataset into few classes so that
#    at least 90% of information of dataset is explained through
#    new classification. (Hint: model the variable "qtr" to
#    variables "togo", "kicker", and "ydline"). How many LDs do
#    you choose? Explain the reason.

qtrlda <- lda(qtr ~ togo + kicker + ydline, data = data)
qtrlda

## Call:
## lda(qtr ~ togo + kicker + ydline, data = data)
##
## Prior probabilities of groups:
```

```
##          1          2          3          4          5
## 0.20636451 0.35969142 0.17550627 0.24590164 0.01253616
##
## Group means:
##       togo   kicker   ydline
## 1 6.481308 19.64486 17.22897
## 2 6.973190 18.77212 19.30027
## 3 6.543956 19.96703 19.03297
## 4 6.792157 20.20000 18.53725
## 5 5.923077 22.61538 19.53846
##
## Coefficients of linear discriminants:
##                LD1         LD2         LD3
## togo    0.06665269  0.12498308  0.20996464
## kicker -0.04134867 -0.06009657  0.05013225
## ydline  0.07726467 -0.07173243 -0.02257770
##
## Proportion of trace:
##   LD1   LD2   LD3
## 0.615 0.322 0.063
```

```
0.615 + 0.322 # 0.937
```

```
## [1] 0.937
```

```
# LD1 and LD2 combined explain 93.7% of the information and are selected


# 2. Apply PCA, and identify the important principle components
#    involving at least 90% of dataset variation. Explain your
#    decision strategy? Plot principle components versus their
#    variance (Hint: to sketch the plot use the Scree plot).

# We start by removing any of the non-continuous data
# We also remove season as it has 0 variance
keep_columns <- c('qtr', 'min', 'sec', 'down', 'togo','kicker','ydline','dist
ance',
                  'homekick', 'kickdiff', 'timerem', 'offscore', 'defscore',
                  'GOOD', 'Missed', 'Blocked')
data <- data[,keep_columns]

names(data)
```

```
##  [1] "qtr"      "min"      "sec"      "down"     "togo"     "kicker"
##  [7] "ydline"   "distance" "homekick" "kickdiff" "timerem"  "offscore"
## [13] "defscore" "GOOD"     "Missed"   "Blocked"
```

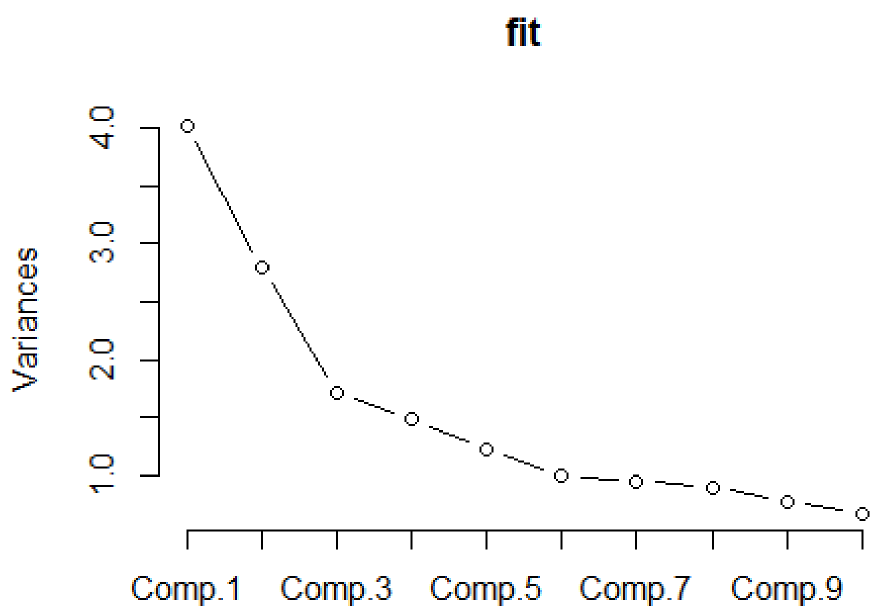
```
describe(data)
```

```
##                                                          25%         50%
##  16592.0000    116.7663    480.9318   -887.0000      1.0000      4.0000
##         75%
##     23.0000   3507.0000 -16576.0000

fit <- princomp(data, cor = TRUE)

summary(fit)

## Importance of components:
##                           Comp.1    Comp.2    Comp.3     Comp.4     Comp.5
## Standard deviation     2.0048406 1.6721636 1.3083384 1.22081818 1.11060318
## Proportion of Variance 0.2512116 0.1747582 0.1069843 0.09314981 0.07708996
## Cumulative Proportion  0.2512116 0.4259698 0.5329542 0.62610397 0.70319393
##                            Comp.6     Comp.7     Comp.8     Comp.9    Comp
.10
## Standard deviation     1.00083350 0.97841761 0.95110310 0.88207039 0.82066
687
## Proportion of Variance 0.06260423 0.05983131 0.05653732 0.04862801 0.04209
338
## Cumulative Proportion  0.76579816 0.82562948 0.88216680 0.93079481 0.97288
819
##                           Comp.11     Comp.12      Comp.13      Comp.14
## Standard deviation     0.62552037 0.203651798 3.223577e-02 3.041048e-08
## Proportion of Variance 0.02445473 0.002592128 6.494657e-05 5.779982e-17
## Cumulative Proportion  0.99734293 0.999935053 1.000000e+00 1.000000e+00
##                             Comp.15      Comp.16
## Standard deviation     2.446684e-08 8.914025e-09
## Proportion of Variance 3.741415e-17 4.966240e-18
## Cumulative Proportion  1.000000e+00 1.000000e+00

# Importance of components:
#                           Comp.1    Comp.2    Comp.3     Comp.4     Comp.5
Comp.6
# Standard deviation     2.0048406 1.6721636 1.3083384 1.22081818 1.11060318
1.00083350
# Proportion of Variance 0.2512116 0.1747582 0.1069843 0.09314981 0.07708996
0.06260423
# Cumulative Proportion  0.2512116 0.4259698 0.5329542 0.62610397 0.70319393
0.76579816
#                            Comp.7     Comp.8     Comp.9    Comp.10     Comp.
11     Comp.12
# Standard deviation     0.97841761 0.95110310 0.88207039 0.82066687 0.625520
37 0.203651798
# Proportion of Variance 0.05983131 0.05653732 0.04862801 0.04209338 0.024454
73 0.002592128
# Cumulative Proportion  0.82562948 0.88216680 0.93079481 0.97288819 0.997342
93 0.999935053
#                             Comp.13      Comp.14      Comp.15      Comp.16
# Standard deviation     3.223577e-02 3.041048e-08 2.446684e-08 8.914025e-09
```

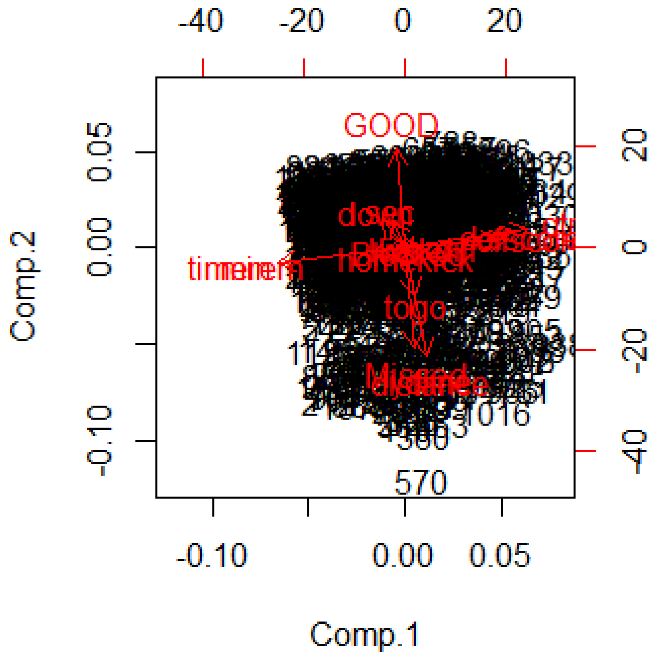
```
# Proportion of Variance 6.494657e-05 5.779982e-17 3.741415e-17 4.966240e-18
# Cumulative Proportion  1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00

loadings(fit)

##
## Loadings:
##          Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9 Co
mp.10
## qtr       0.472
## min      -0.483
## sec              0.128         0.196  0.662                      -0.280 -0
.652
## down             0.132         0.240  0.612               -0.209  0.101  0
.673
## togo            -0.221  0.265 -0.290                             -0.846  0
.270
## kicker                         0.101        -0.917  0.369
## ydline          -0.491        -0.308  0.246        -0.121         0.257
## distance        -0.491        -0.307  0.248        -0.119         0.255
## homekick               -0.329         0.151  0.112  0.108  0.905         0
.130
## kickdiff               -0.696 -0.211                      -0.240 -0.162
## timerem  -0.484
## offscore  0.389        -0.349 -0.120                      -0.142
## defscore  0.357         0.411  0.109                       0.117  0.100
## GOOD             0.455  0.122 -0.485               -0.113
## Missed          -0.455 -0.122  0.485                0.113
## Blocked                       -0.260  0.111  0.315  0.886 -0.176
##          Comp.11 Comp.12 Comp.13 Comp.14 Comp.15 Comp.16
## qtr       0.301   0.811
## min      -0.298   0.410           0.691   0.146
## sec
## down     -0.148
## togo
## kicker
## ydline                   -0.707
## distance                  0.707
## homekick
## kickdiff                         -0.126   0.594
## timerem  -0.297   0.410          -0.692  -0.146
## offscore -0.567                   0.123  -0.580
## defscore -0.617                  -0.109   0.514
## GOOD                                             -0.706
## Missed                                           -0.706
## Blocked
##
##                Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Com
p.9
## SS loadings     1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.
```

```
000
## Proportion Var  0.062  0.063  0.062  0.062  0.063  0.062  0.062  0.062  0.
062
## Cumulative Var  0.062  0.125  0.188  0.250  0.312  0.375  0.437  0.500  0.
562
##                Comp.10 Comp.11 Comp.12 Comp.13 Comp.14 Comp.15 Comp.16
## SS loadings      1.000   1.000   1.000   1.000   1.000   1.000   1.000
## Proportion Var   0.062   0.062   0.062   0.062   0.062   0.062   0.062
## Cumulative Var   0.625   0.687   0.750   0.812   0.875   0.937   1.000
```

```
plot(fit, type = 'lines')
```



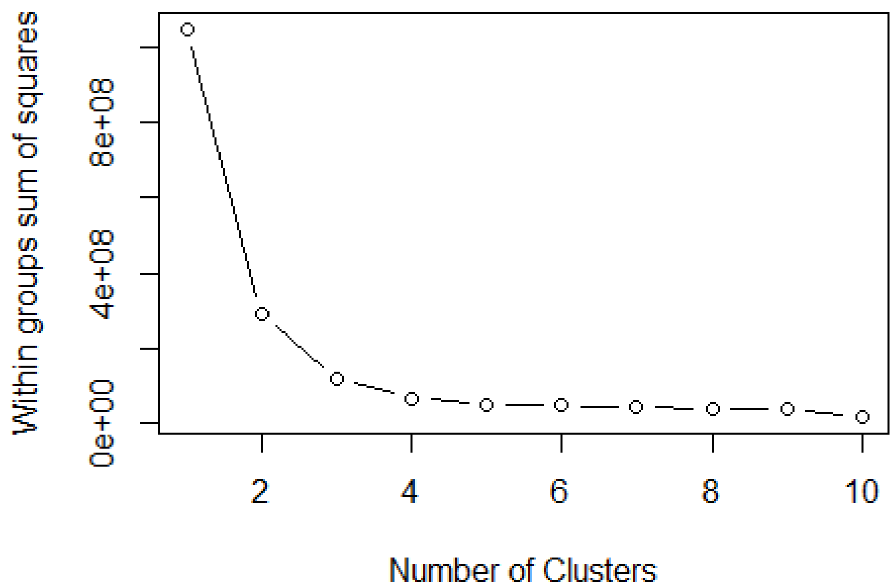

```
biplot(fit)
```



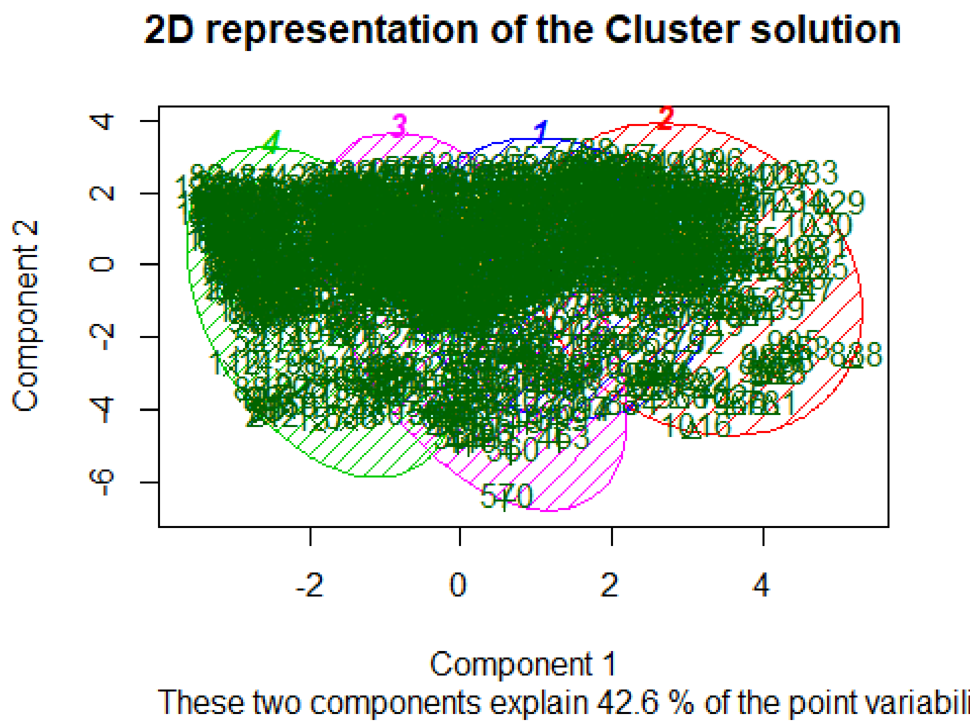

```
# From the screeplot we have a couple of options we could choose from
# Comp1 + Comp2 + Comp3 explains 53% of the data and the increase to 4 is onl
y 9%
# Another possibile point of inflection could be 6 which explains 76% and the
inrease
# to 7 is only 6%

# in this quesiton however, we're being asked to explain 90% of the dataset v
ariation
# for that we need a total of 9 components which explain 93%



# 3. Split the dataset into two sets of variables so that
#    X=( togo,kicker,ydline) and Y=( distance, homekick).

names(data)

##  [1] "qtr"      "min"      "sec"      "down"     "togo"     "kicker"
##  [7] "ydline"   "distance" "homekick" "kickdiff" "timerem"  "offscore"
## [13] "defscore" "GOOD"     "Missed"   "Blocked"

x <- data[,5:7]
y <- data[,8:9]
names(x)

## [1] "togo"   "kicker" "ydline"
```

```r
# [1] "togo"   "kicker" "ydline"
names(y)

## [1] "distance" "homekick"

# [1] "distance" "homekick"

#    Apply
#    canonical correlation analysis to find the cross-correlation
#    between X and Y. What is the correlation between ydline and
#    distance?


# display the canonical correlations
cc1 <- cc(x, y)
cc1$cor

## [1] 0.99894989 0.06975549

# [1] 0.99894989 0.06975549

cor(x, y)  # correlation between two set of variables

##            distance    homekick
## togo    0.315641454 -0.04838438
## kicker -0.001951722 -0.02363159
## ydline  0.998947222  0.04295427

#            distance    homekick
# togo    0.315641454 -0.04838438
# kicker -0.001951722 -0.02363159
# ydline  0.998947222  0.04295427


# The correlation betweel ydline and distance is 0.998947222
# So positively correlated and almost 1!

#    Use K-means clustering analysis to identify the most
#    important classes. How many classes do you select? Why?


# We use this wssplot function is to work out the lowest number of clusters
# with the highest amount of variation (information)
wssplot <- function(data, nc=10, seed=1234){
        wss <- (nrow(data)-1)*sum(apply(data,2,var))
        for (i in 2:nc) {
                set.seed(seed)
                wss[i] <- sum(kmeans(data, centers = i)$withinss)}
        plot(1:nc, wss, type = "b", xlab = "Number of Clusters",
             ylab = "Within groups sum of squares")}
```

```
wssplot(data, nc = 10)
```




```
# This scree plot gives us an indication of the number of clusters we should
be
# looking for. In this example, after 4 there is no major change.
# We could also make a case for 3, but 4 appears to be a better point of infl
ection

# generate the clusters
k.means.fit <- kmeans(data, 4) # the 4 indicates the number of groups previou
s selected

# We can also plot this to visualise the 4 groups
library(cluster)
clusplot(data, k.means.fit$cluster, main = '2D representation of the Cluster
solution',
         color = TRUE, shade = TRUE,
         labels = 2, lines = 0)
```


2D representation of the Cluster solution
Component 2
Component 1
These two components explain 42.6 % of the point variabili