|  |  |
| --- | --- |
|  | |
| Data Advanced Data Analytics  CA TWO | |
| Module code : B8IT109 | |
| Ciaran Finnegan  Student No : 10524150  07/06/2020 |  |
|  |  |

Table of Contents

1 Question One 3

1.1 Question 1 – from PDF 3

1.2 Output From RStudio Cloud Console 4

2 Question Two 13

2.1 Question 2 – from PDF 13

2.2 Output from RStudio Cloud Console 14

3 Question Three 24

3.1 Question 3 – from PDF 24

3.2 Output from RStudio Cloud Console 25

4 Question Four 38

4.1 Question 4 – from PDF 38

4.2 Output from RStudio Cloud Console 39

# Question One

## Question 1 – from PDF

Use in-built dataset ‘airquality’,

a) Explore the general feature of dataset using appropriate R functions.

(**5 Marks**)

b) Perform data cleansing if required. (**5 Marks**)

c) Consider ‘Temp’ attributes and compute the central and variational measures. (**10 Marks**)

d) Apply boxplot technique to detect outlier of ‘wind’ attribute if any.

(**10 Marks**)

**(Total: 30 Marks)**

## Output From RStudio Cloud Console

## CA Two Advanced Data Analytics : Module Code B8IT109

> ## Student Name : Ciaran Finnegan

>

> ## Student Number : 10524150

>

> ## June 2020

>

>

**> ## Question One**

>

>

> ## Library Calls

> library(DT)

> library(ggplot2)

> library(reshape2)

> library(dplyr)

>

> ## References used.

> ## http://www.datasciencemadesimple.com/melting-casting-r/

> ## http://www.datasciencemadesimple.com/drop-variables-columns-r-using-dplyr/

>

>

>

> ########################################################

> ## Use in-built dataset ‘airquality’

> ########################################################

>

> ## Read in airquality dataset

> aq <- data.frame(airquality)

>

>

> #########################################################################

>

> # Below are a set of R functions to perform relatively straightforward

actions to generate central and variational measures.

>

> # I am re-using functions I wrote for the CA One solutions.

>

>

> # Calculate mean of attribute list - pass an attribute list from air

quality dataset

> dataset.attr.mean <- function(y) {

+

+ return(mean(y))

+

+ }

>

> # Calculate Median of attribute list - pass an attribute list from air

quality dataset

> dataset.attr.median <- function(y) {

+

+ return(median(y))

+

+ }

>

> # Calculate Mode of attribute list - pass an attribute list from air

quality dataset

> # Start by creating a dedicated 'Mode' function - as shown in lectures

> dataset.attr.mode <- function(v) {

+

+ uniqv <- unique(v)

+ uniqv[which.max(tabulate(match(v, uniqv)))]

+

+ }

>

>

> # Calculate range of values of an attribute list - pass an attribute list from air quality dataset

> dataset.attr.range <- function(v) {

+

+ # Range of values in <> attribute

+ rDPF = range(v)

+ RangeDPF = rDPF[2] - rDPF[1]

+ RangeDPF

+

+ }

>

> # Calculate variance of values of an attribute list –

pass an attribute list from air quality dataset

> dataset.attr.var <- function(v) {

+

+ V=var(v)

+

+ }

>

> # Calculate standard deviation of values of an attribute list –

pass an attribute list from air quality dataset

> dataset.attr.stddev <- function(v) {

+

+ s=sd(v)

+

+ }

>

> #########################################################################

> #########################################################################

**> ## Q.1 (Part a)**

> ## Explore the general feature of dataset using appropriate R functions.

>

>

> # Display number of rows in airquality dataset

> nrow(aq)

[1] 153

>

> # Display first and last five rows of airquality dataset

> head(aq)

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

> tail(aq)

Ozone Solar.R Wind Temp Month Day

148 14 20 16.6 63 9 25

149 30 193 6.9 70 9 26

150 NA 145 13.2 77 9 27

151 14 191 14.3 75 9 28

152 18 131 8.0 76 9 29

153 20 223 11.5 68 9 30

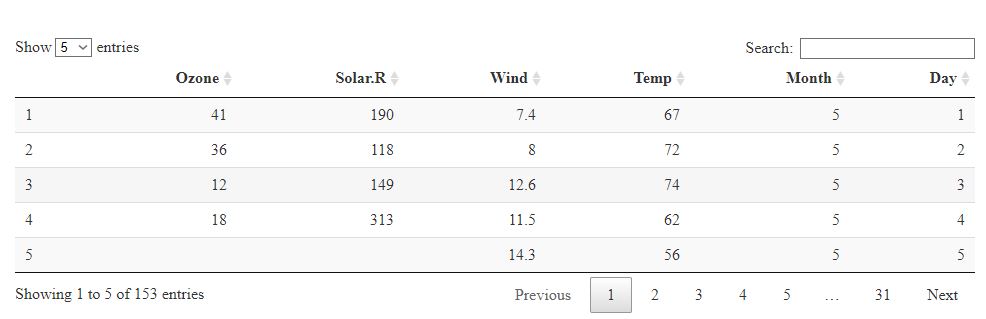
>

> ## Display table view of records - opens in the 'viewer' and can be

exported as a web page

> DT::datatable(aq, options = list(lengthMenu = c(5, 30, 50), pageLength = 5))

<Data Table View>



> ## Display summary statistics for the variables in the airquality dataset

> ## This will help show where missing data entries are in the dataset

> summary(aq)

Ozone Solar.R Wind Temp Month Day

Min. : 1.00 Min. : 7.0 Min. : 1.700 Min. :56.00 Min. :5.000 Min. : 1.0

1st Qu.: 18.00 1st Qu.:115.8 1st Qu.: 7.400 1st Qu.:72.00 1st Qu.:6.000 1st Qu.: 8.0

Median : 31.50 Median :205.0 Median : 9.700 Median :79.00 Median :7.000 Median :16.0

Mean : 42.13 Mean :185.9 Mean : 9.958 Mean :77.88 Mean :6.993 Mean :15.8

3rd Qu.: 63.25 3rd Qu.:258.8 3rd Qu.:11.500 3rd Qu.:85.00 3rd Qu.:8.000 3rd Qu.:23.0

Max. :168.00 Max. :334.0 Max. :20.700 Max. :97.00 Max. :9.000 Max. :31.0

NA's :37 NA's :7

> ## Display structure of the airquality dataset

> str(aq)

'data.frame': 153 obs. of 6 variables:

$ Ozone : int 41 36 12 18 NA 28 23 19 8 NA ...

$ Solar.R: int 190 118 149 313 NA NA 299 99 19 194 ...

$ Wind : num 7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...

$ Temp : int 67 72 74 62 56 66 65 59 61 69 ...

$ Month : int 5 5 5 5 5 5 5 5 5 5 ...

$ Day : int 1 2 3 4 5 6 7 8 9 10 ...

>

>

> ## -------------------------------------------------------------------

> ## Generate total Histogram view of airquality dataset.

> ## -------------------------------------------------------------------

> ## However, remove the 'month' and 'day' columns from this visualisation

> ## as they will distort the scale of the output.

> ## Their values are obvious to understand ;

> ## 'month' is just the number of days in each month in the dataset

> ## 'day' is the day of the particular month

>

>

> ## Read in a copy of the full air quality dataset to another dataset

> ## variable to be used to generate the histogram

> airqHist <- aq

>

> ## Drop the 'Month' and 'Day' columns as they add no value to this

> ## histogram view and distort the scales of the visualisations

> airqHist = select(airqHist, -c(Month, Day))

>

> ## Reshape the data to fit into a single histogram

> d <- melt(airqHist)

No id variables; using all as measure variables

>

> ## Generate Histogram visualisation for the airquality dataset

> ggplot(d,aes(x = value, fill=variable)) +

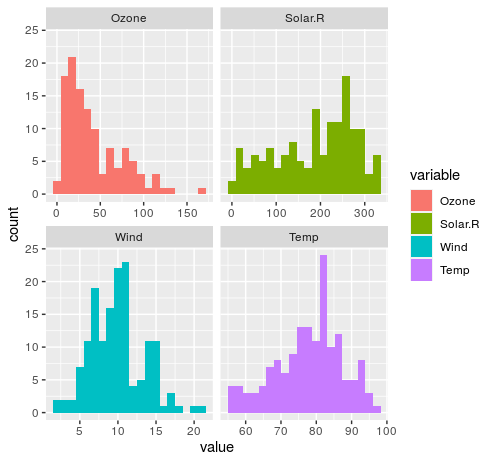
+ facet\_wrap(~variable,scales = "free\_x", nrow=2,ncol=2) +

+ geom\_histogram(position="identity", bins=20)

Warning message:

Removed 44 rows containing non-finite values (stat\_bin).

>



> ## The histogram generates a warning message about 44 rows in the dataset

> ## with missing entries, which are ignored in the graph.

> ## I will address these data issues in the next section of this solution to Question 1 of ADA CA Two.

> #########################################################################

**> ## Q.1 (Part b)**

> ## Perform data cleansing if required.

>

>

> ## Some entries are missing - generate a count by column to focus in these values as shown at the end of the output from the 'summary' function.

>

> ## Other options exist to deal with missing entries but, for simplicity, I have chosen to just delete these rows from the dataset.

>

> ## Show initial row count before data clean up

> nrow(aq)

[1] 153

> ## Show where missing entries are in the dataset

> colSums(is.na(aq))

Ozone Solar.R Wind Temp Month Day

37 7 0 0 0 0

> ## Remove the rows with missing data

> aqClean <- na.omit(aq)

> ## Check row count after rows with missing data removed.

> nrow(aqClean)

[1] 111

> >

> ## 'Month' and 'Day' are showing as numeric values. These should be adjusted to categorical. We use a 'factor' function to alter this in the cleaned

dataset.

> aqClean$Month = factor(aqClean$Month)

> aqClean$Day = factor(aqClean$Day)

> head(aqClean)

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

7 23 299 8.6 65 5 7

8 19 99 13.8 59 5 8

> summary(aqClean)

Ozone Solar.R Wind Temp Month Day

Min. : 1.0 Min. : 7.0 Min. : 2.30 Min. :57.00 5:24 7 : 5

1st Qu.: 18.0 1st Qu.:113.5 1st Qu.: 7.40 1st Qu.:71.00 6: 9 9 : 5

Median : 31.0 Median :207.0 Median : 9.70 Median :79.00 7:26 13 : 5

Mean : 42.1 Mean :184.8 Mean : 9.94 Mean :77.79 8:23 16 : 5

3rd Qu.: 62.0 3rd Qu.:255.5 3rd Qu.:11.50 3rd Qu.:84.50 9:29 17 : 5

Max. :168.0 Max. :334.0 Max. :20.70 Max. :97.00 18 : 5

(Other):81

>

###########################################################################

**> ## Q.1 (Part c)**

> ## Consider ‘Temp’ attributes and compute the central and variational

measures.

>

>

> #########################################################################

> # Central Measures - mean, median, and mode

> #########################################################################

>

> # Return Mean of 'Temp' attribute value

> attrMean = dataset.attr.mean(aqClean$Temp)

> attrMean

[1] 77.79279

>

> # Return Median of 'Temp' attribute value

> attrMedian = dataset.attr.median(aqClean$Temp)

> attrMedian

[1] 79

>

> # Return Mode of 'Temp' attribute value

> attrMode = dataset.attr.mode(aqClean$Temp)

> attrMode

[1] 81

>

>

> #########################################################################

> # Variational Measures : Range/variance/standard deviation

> #########################################################################

>

> # Return Range of 'Temp' attribute values

> attrRange = dataset.attr.range(aqClean$Temp)

> attrRange

[1] 40

>

> # Return variance

> attrVar = dataset.attr.var(aqClean$Temp)

> attrVar

[1] 90.82031

>

> # Return standard deviation

> attrSD = dataset.attr.stddev(aqClean$Temp)

> attrSD

[1] 9.529969

>

>

>

>

|  |
| --- |
| #################################################################################  **> ## Q.1 (Part d)**  > ## Apply box plot technique to detect outlier of ‘wind’ attribute if any.  >  >  > ## Taking the entire air quality data set into account we can see that there  > ## three outliers when we represent the 'wind' attribute values through the  > ## BoxPlot technique.  >  > ## These outlier values are represented as red stars in the BoxPlot  >  > ggplot(aqClean, aes(y = aqClean$Wind)) +  + geom\_boxplot(outlier.colour="red", outlier.shape=8,  + outlier.size=4) +  + ylab("WIND") +  + theme(axis.title.x = element\_text(color = "blue", size = 14, face = "bold"),  + axis.text.x = element\_blank(),  + axis.ticks.y = element\_blank()) +  + scale\_y\_continuous(limits=c(1,23), breaks=seq(0,23,1))  >  >  <BoxPlot Wind Graph>    >    > ## Breaking down the wind measurements by month we can see the following  > ## range of data across the time frame for the air quality dataset  >  > ## Two outlier values appear in the 5th month, and one appears in the 6th  >  > ggplot(aqClean, aes(x = Month, y = aqClean$Wind, col = Month, fill = Month)) +  + geom\_boxplot(alpha = 0.2,  + outlier.shape=4,  + outlier.size=6) +  + ylab("WIND") +  + scale\_color\_manual(values = c("red", "black", "green", "orange", "blue")) +  + scale\_fill\_manual(values = c("red", "black", "green", "orange", "blue"))  <BoxPlot Wind by Month Graph> |
|  |
| |  | | --- | |  | |

# Question Two

## Question 2 – from PDF

Use dataset available on http://users.stat.ufl.edu/~winner/data/nfl2008\_fga.csv , then:

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

**(5 Marks)**

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

**(5 Marks)**

(c) Predict the test dataset using the trained model. **(5 Marks)**

(d) Provide the confusion matrix and obtain the probability of correctness of predictions. **(10 Marks)**

(Total: 25 Marks)

## Output from RStudio Cloud Console

> ## CA Two Advanced Data Analytics : Module Code B8IT109

> ## Student Name : Ciaran Finnegan

>

> ## Student Number : 10524150

>

> ## June 2020

>

>

**> ## Question Two**

>

> ## Use dataset available on http://users.stat.ufl.edu/~winner/data/nfl2008\_fga.csv

> ## (a) Train the model using 80% of this dataset and suggest an

appropriate GLM to model homekick to togo, ydline and kicker variables.

>

>

> library(caTools) # useful to split data to training and test datasets

> library(MASS)

>

> #############################################################

> ## Read in the NFL dataset

> link='http://users.stat.ufl.edu/~winner/data/nfl2008\_fga.csv'

> datasetNFL=read.csv(link)

> ## Present head and tail and structure of NFL dataset

> head(datasetNFL)

GameDate AwayTeam HomeTeam qtr min sec kickteam def down togo kicker ydline name distance homekick kickdiff timerem offscore

1 20081130 IND CLE 1 47 2 IND CLE 4 11 15 12 A.Vinatieri 30 0 -3 2822 0

2 20081005 IND HOU 1 54 47 IND HOU 4 3 15 28 A.Vinatieri 46 0 0 3287 0

3 20081228 TEN IND 1 45 20 IND TEN 4 3 15 10 A.Vinatieri 28 1 7 2720 7

4 20081012 BAL IND 1 45 42 IND BAL 4 1 15 19 A.Vinatieri 37 1 14 2742 14

5 20080907 CHI IND 1 50 56 IND CHI 4 21 15 21 A.Vinatieri 39 1 0 3056 0

6 20081116 HOU IND 1 50 43 IND HOU 4 7 15 22 A.Vinatieri 40 1 -3 3043 0

defscore season GOOD Missed Blocked

1 3 2008 1 0 0

2 0 2008 1 0 0

3 0 2008 1 0 0

4 0 2008 1 0 0

5 0 2008 1 0 0

6 3 2008 1 0 0

> tail(datasetNFL)

GameDate AwayTeam HomeTeam qtr min sec kickteam def down togo kicker ydline name distance homekick kickdiff timerem offscore

1034 20081102 TB KC 5 -5 27 TB KC 4 1 36 16 M.Bryant 34 0 0 -273 27

1035 20081102 GB TEN 5 -6 41 TEN GB 3 2 37 23 R.Bironas 41 1 0 -319 16

1036 20081211 NO CHI 5 -3 19 CHI NO 2 13 7 18 R.Gould 35 1 0 -161 24

1037 20081222 GB CHI 5 -4 33 CHI GB 3 10 7 20 R.Gould 38 1 0 -207 17

1038 20081116 PHI CIN 5 -15 13 CIN PHI 4 5 8 29 S.Graham 47 1 0 -887 13

1039 20081019 NYJ OAK 5 -13 35 OAK NYJ 4 11 28 39 S.Janikowski 57 1 0 -745 13

defscore season GOOD Missed Blocked

1034 27 2008 1 0 0

1035 16 2008 1 0 0

1036 24 2008 1 0 0

1037 17 2008 1 0 0

1038 13 2008 0 1 0

1039 13 2008 1 0 0

> str(datasetNFL)

'data.frame': 1039 obs. of 23 variables:

$ GameDate: int 20081130 20081005 20081228 20081012 20080907 20081116 20081123 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 14 16 16 24 ...

$ def : Factor w/ 32 levels "ARI","ATL","BAL",..: 8 13 31 3 6 13 26 10 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 ...

> #############################################################

>

>

> #########################################################################

**> ## Q.2 (Part a)**

> ## Train the model using 80% of this dataset and suggest an appropriate

GLM to model homekick to togo, ydline and kicker variables.

>

>

> ## Minor Clean up of NFL dataset

> sum(is.na(datasetNFL)) # Check how many rows have missing values

[1] 4

> datasetNFL <- na.omit(datasetNFL) # Clean the rows with missing values

> sum(is.na(datasetNFL)) # Check the missing values are removed

[1] 0

> ## For this NFL dataset four rows with missing date were found and

removed

>

> #

> ## Display the values for 'homekick' as this is the variable which the

question wants to model against togo, ydline and kicker variables.

> table(datasetNFL$homekick)

0 1

525 512

> ## I can see the range of binary values in the 'homekick' attribute in

the NFL dataset

>

> ## The output variable is 'homekick'. As this is a binary outcome (0,1)

the best GLM is therefore **Logistic Regression**.

>

> ## The togo, ydline and kicker attributes in the NFK dataset are the

input variables in this question.

>

>

> ## For this section of the question use 'set.seed()' to ensure

consistency of results

> set.seed(42) # The purpose of this is to ensure consistency in the

initial prediction

>

> #

> ## Split NFL dataset in 80/20 ratio

> sample = sample.split(datasetNFL$homekick, SplitRatio=0.80)

> trainsetNFL = subset(datasetNFL, sample==TRUE)

> testsetNFL = subset(datasetNFL, sample==FALSE)

>

> #

> # Display the number of rows in each set after splitting the NFL data

> nrow(datasetNFL) # Original dataset

[1] 1037

> nrow(trainsetNFL) # Training set

[1] 830

> nrow(testsetNFL) # Test set

[1] 207

>

> #

> ## Model the trainset by fitting the Logistic Regression GLM - the input (independent) variables are used as required by the question

> ## The 'family' variable is set to 'binomial' because as we are using

Logistic Regression

> fit=glm(homekick~togo+ydline+kicker, data=trainsetNFL, family='binomial')

> summary(fit)

Call:

glm(formula = homekick ~ togo + ydline + kicker, family = "binomial",

data = trainsetNFL)

Deviance Residuals:

Min 1Q Median 3Q Max

-1.3739 -1.1628 -0.9582 1.1793 1.4378

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) -0.021338 0.208033 -0.103 0.9183

togo -0.044716 0.017652 -2.533 0.0113 \*

ydline 0.015585 0.007591 2.053 0.0401 \*

kicker 0.000503 0.006122 0.082 0.9345

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1150.5 on 829 degrees of freedom

Residual deviance: 1142.2 on 826 degrees of freedom

AIC: 1150.2

Number of Fisher Scoring iterations: 4

>

>

>

> #########################################################################

**> ## Q.2 (Part b)**

> ## Specify the significant variables on homekick at the level of 𝛼=0.05, and estimate the parameters of your model.

>

> #

> ## Re-display the outcome

> summary(fit)

Call:

glm(formula = homekick ~ togo + ydline + kicker, family = "binomial",

data = trainsetNFL)

Deviance Residuals:

Min 1Q Median 3Q Max

-1.3739 -1.1628 -0.9582 1.1793 1.4378

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) -0.021338 0.208033 -0.103 0.9183

togo -0.044716 0.017652 -2.533 0.0113 \*

ydline 0.015585 0.007591 2.053 0.0401 \*

kicker 0.000503 0.006122 0.082 0.9345

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1150.5 on 829 degrees of freedom

Residual deviance: 1142.2 on 826 degrees of freedom

AIC: 1150.2

Number of Fisher Scoring iterations: 4

>

> ## The significant variables at the level of α=0.05 are;

> ## (I used a 'seed' setting of '42' to ensure consistent results)

> ##

> ## 'togo'

> ## 'ydline'

>

> ## togo and ydline variables have a 'P' value less than 0.05 (0.0113 and 0.0401 respectively).

> ## kicker has a 'P' value of 0.9345, hence greater than 0.05

> ## The 'Intercept' also has a value greater than α=0.05 and is thus also not a significant parameter for the model.

>

>

> # The estimated parameters for the significant variables are;

(rounded slightly)

> #

> # togo : -0.0447

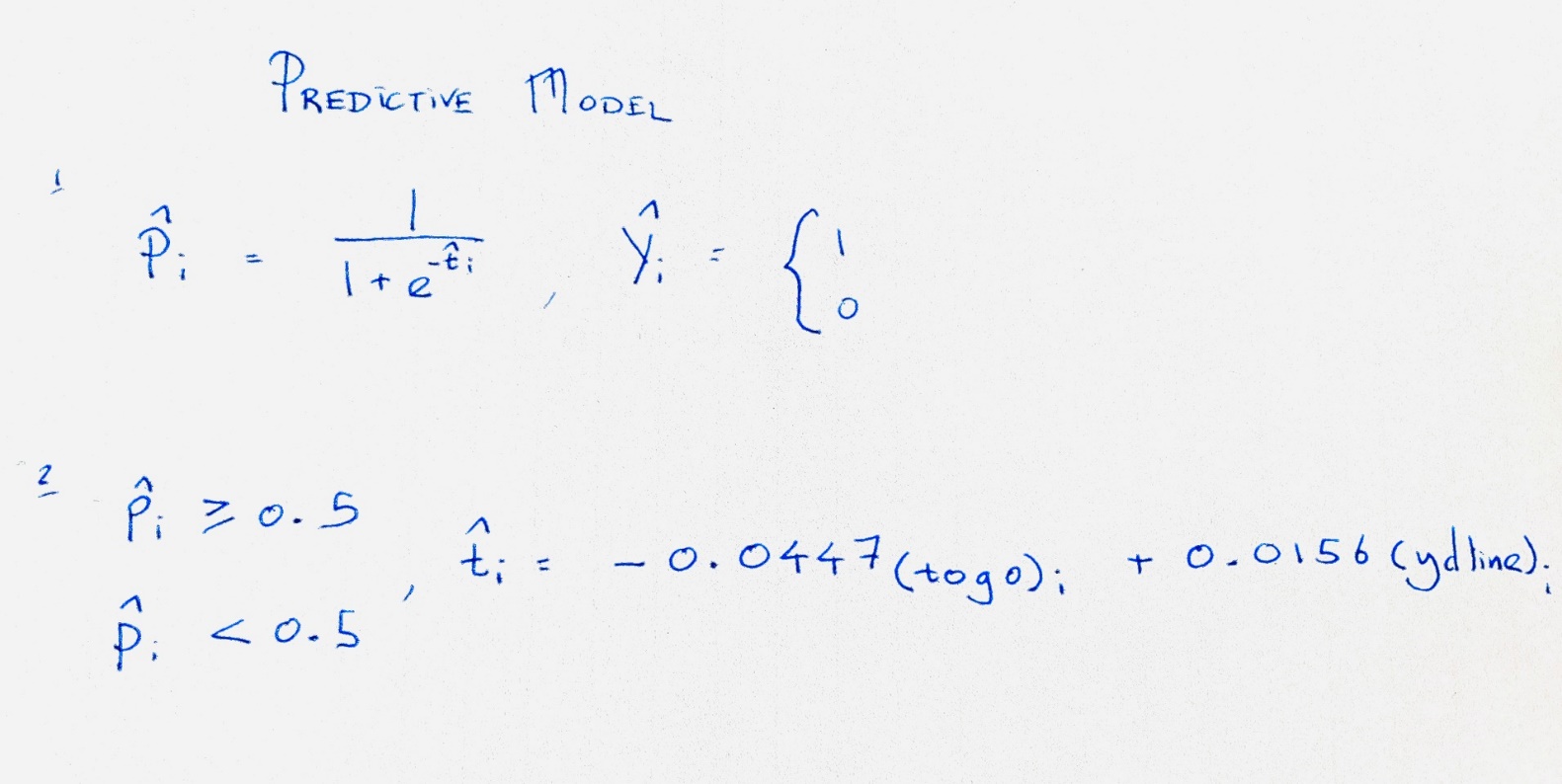
> # ydline : 0.0156

>

> #

>

> ## Photo image of Predictive Model provided here :



>

>

>

>

>

> #

> #########################################################################

**> ## Q.2 (Part c)**

> ## Predict the test dataset using the trained model.

>

> ## The solution could re-train the model with just togo and ydline input variables but it was found that this did not improve accuracy.

>

>

> #############################################

> ## This was discussed in the lecture on Friday 29th May as a means to

reduce the model to only significant attributes.

> ##

> #lr\_red = stepAIC(fit)

> #summary(lr\_red)

> #pred\_red=predict(lr\_red, testsetNFL)

> #predres=predict(lr\_red, testsetNFL, type='response')

>

> # I also ran a test to see if re-training the model with just the input

variables mentioned in the question would improve accuracy, but it did not.

>

> #datasetRed = data.frame(datasetNFL$togo, datasetNFL$ydline, datasetNFL$kicker, datasetNFL$homekick)

> #sample = sample.split(datasetRed$datasetNFL.homekick, SplitRatio=0.80)

> #trainsetRed = subset(datasetRed, sample==TRUE)

> #testsetRed = subset(datasetRed, sample==FALSE)

> #fit.Red = glm(datasetNFL.homekick~., data=trainsetRed, family='binomial')

> #predres=predict(fit.Red, testsetRed, type='response')

> #testsetNFL <- testsetRed

> #############################################

>

>

>

> ## Use the trained model to predict the results from the NFL test set

> predres=predict(fit, testsetNFL, type='response')

> ## Display the predictions using the test set

> predres

1 2 5 8 17 20 26 27 36 38 40 45 49

0.4210042 0.5716129 0.3485202 0.4500130 0.5281042 0.5094325 0.4997219 0.5655777 0.4658542 0.4043266 0.4480219 0.4974650 0.4400874

51 56 59 66 69 74 76 79 80 82 86 88 105

0.4826635 0.5278535 0.5293489 0.5081271 0.4625969 0.4520057 0.5058946 0.5007544 0.5574144 0.4912819 0.5273388 0.4919719 0.4672160

110 114 116 137 149 155 159 160 166 168 169 180 181

0.4703673 0.4452889 0.5170652 0.4973129 0.3789857 0.4383594 0.6011108 0.4959780 0.4717460 0.5340027 0.4664648 0.5444789 0.5032958

192 201 212 213 228 229 232 240 241 243 259 264 269

0.5618750 0.4546149 0.4962514 0.4612217 0.5661955 0.6099849 0.5265799 0.4910304 0.5267601 0.5544427 0.5016148 0.5735760 0.4668688

271 274 275 277 278 294 298 301 306 330 337 338 340

0.4552516 0.4759793 0.4542407 0.5071300 0.5449518 0.4829060 0.5111902 0.4762302 0.5515850 0.4808637 0.5061791 0.5602516 0.4748503

342 344 350 363 364 373 375 377 378 390 391 394 398

0.4188201 0.5274905 0.5158528 0.4052750 0.4727468 0.4827759 0.4848122 0.4739810 0.4851409 0.5105354 0.5294742 0.5761749 0.4599720

417 423 427 431 432 433 438 440 444 445 450 451 456

0.4481463 0.4804870 0.4786323 0.4543916 0.5067726 0.5518122 0.3581623 0.5308393 0.5051336 0.4978510 0.4572875 0.5096905 0.5732048

464 467 477 478 479 489 503 505 506 508 529 532 534

0.4552909 0.5324872 0.4378576 0.4832762 0.4871690 0.5628523 0.5624724 0.4807381 0.4594788 0.5056344 0.5192002 0.4837655 0.4852929

537 545 549 551 552 556 566 576 583 588 606 607 609

0.5177990 0.4819099 0.4811148 0.4092928 0.4719946 0.4792599 0.5713944 0.5432267 0.4458141 0.5711416 0.4915399 0.4401004 0.3841983

616 629 634 637 638 640 648 653 654 657 671 675 676

0.5041628 0.4629765 0.5148127 0.4168431 0.5352587 0.4935625 0.4210256 0.5540699 0.5148061 0.4912819 0.5963955 0.3889404 0.4858153

678 684 691 692 697 698 701 709 714 719 722 727 728

0.4749823 0.5219599 0.4157431 0.4936949 0.5719953 0.5116864 0.4768576 0.5670645 0.5218497 0.4898567 0.4852731 0.4576272 0.5395917

730 740 742 743 750 756 764 768 773 775 783 784 791

0.4581332 0.5494777 0.5449845 0.4662145 0.5227327 0.5579107 0.4763557 0.3909746 0.4913082 0.5895927 0.4596189 0.5221846 0.5159302

792 796 798 803 818 822 823 825 831 842 844 845 846

0.5613601 0.5299557 0.4874072 0.5596188 0.5002667 0.4990091 0.4832499 0.5177924 0.4639903 0.4687933 0.5106742 0.5368631 0.6072290

853 860 868 891 895 900 903 906 908 916 924 925 929

0.5767632 0.4675917 0.4948484 0.4393720 0.4886904 0.4847969 0.5130820 0.5145416 0.5108284 0.5053830 0.4223793 0.4373710 0.4695958

943 949 950 952 956 957 961 963 965 973 976 988 996

0.4843076 0.4650065 0.4408313 0.5358974 0.5105420 0.4812404 0.4857757 0.5373721 0.5270749 0.4782974 0.5061440 0.4320580 0.5133975

998 1001 1002 1005 1007 1023 1027 1028 1029 1030 1031 1039

0.4658608 0.5870428 0.4653536 0.5397516 0.5669281 0.5067397 0.4687189 0.4678684 0.5379689 0.5767632 0.4590074 0.5271428

>

>

>

> #########################################################################

**> ## Q.2 (Part d)**

> ## Provide the confusion matrix and obtain the probability of correctness of predictions.

>

> ## Review the layout of the head of the testset of the NFL dataset

> head(testsetNFL)

GameDate AwayTeam HomeTeam qtr min sec kickteam def down togo kicker ydline name distance homekick kickdiff timerem offscore

1 20081130 IND CLE 1 47 2 IND CLE 4 11 15 12 A.Vinatieri 30 0 -3 2822 0

2 20081005 IND HOU 1 54 47 IND HOU 4 3 15 28 A.Vinatieri 46 0 0 3287 0

5 20080907 CHI IND 1 50 56 IND CHI 4 21 15 21 A.Vinatieri 39 1 0 3056 0

8 20081207 KC DEN 1 52 34 KC DEN 4 7 18 8 C.Barth 26 0 0 3154 0

17 20081228 DAL PHI 1 53 54 PHI DAL 4 5 29 22 D.Akers 40 1 0 3234 0

20 20081102 MIA DEN 1 51 1 MIA DEN 4 9 19 29 D.Carpenter 47 0 3 3061 3

defscore season GOOD Missed Blocked

1 3 2008 1 0 0

2 0 2008 1 0 0

5 0 2008 1 0 0

8 0 2008 1 0 0

17 0 2008 1 0 0

20 0 2008 1 0 0

>

> ## Convert phat to yhat

> ## Set up so that predicted results from the model are compared against

the actual values in the testset data. It is necessary to change the values to represent the binary outcome

> predictedvalues=rep(0,nrow(testsetNFL))

> ## Assess the number of matching 'homekick' values in the testset against the output of the model

> predictedvalues[predres>=0.5]=1 # Probability of 'homekick' being 1, if p<0.5 then 'homekick'=0

>

> ## Compare the values predicted for the testset against the actual values for 'homekick'

> tab = table(predictedvalues, testsetNFL$homekick)

>

> ## Show a Confusion Matrix to represent the accuracy of the results

> tab

predictedvalues 0 1

0 59 53

1 46 49

> ## Calculate the accuracy as a percentage value

> ## This is a sum of the TP (True Pos) + TN (True Neg) / All Results

> accuracy=sum(tab[row(tab)==col(tab)])/sum(tab)

> #

> accuracy

[1] 0.5217391

> #

> ## Using the given seed of '42' the calculation is (59+49)/(59+49+46+53) = 52% (with rounding)

>

## These values are the ones provided by the Confusion Matrix above.

> ## Accuracy is poor

# Question Three

## Question 3 – from PDF

Using Yahoo Finance API, select a specific stock market price, apply time series analysis, consider ***‘close price’*** as your time series variable:

1. Validate the assumptions using graphical visualization. (5 Marks)
2. Fit the optimized model for ‘***close price’*** and provide the coefficient estimates for the fitted model. **(5 Marks)**
3. What is the estimated order for AR and MA? (5 Marks)

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

**(Total: 25 Marks)**

## Output from RStudio Cloud Console

## CA Two Advanced Data Analytics : Module Code B8IT109

## Student Name : Ciaran Finnegan

## Student Number : 10524150

## June 2020

**## Question Three**

## References used to call Yahoo Finance API

## http://statmath.wu.ac.at/~hornik/QFS1/quantmod-vignette.pdf

## https://stackoverflow.com/questions/26666254/retrieve-monthly-adjusted-stock-quotes-using-the-quantmod-package-in-r

###########################################################################

## Using Yahoo Finance API, select a specific stock market price, apply

time series analysis, consider ‘close price' as your time series variable:

###########################################################################

# 'Quantmod' Package required to access Yahoo Finance API

library(quantmod)

library(forecast)

# Library to plot time series to check for normality

library(ggpubr)

# Create a data structure that contains stock quote objects

ETF\_Data <- new.env()

# Assign dates to set range for daily stock closing prices.

# This range of values over 13 months is set to be large enough for

# accurate forecasting but not too large to degrade the quality of the

# graph visuals.

sDate <- as.Date("2016-12-01") # Start date for time series

eDate <- as.Date("2018-01-01") # End date for time series

# Use the stock price data over the given period of time (above) for chosen company

# This code is written to allow the selection of different company stock

Prices # to compare trends and choose different time series.

# The stock chosen is the AIG (American International Group) for the year

2017

ticker\_symbol = "AIG" # AIG

# Alternative company stocks - not used.

#ticker\_symbol = "IBM" # International Business Machines

#ticker\_symbol = "UA" # Under Armour, Inc

# Invoke 'getSymbols' function to retrieve to stock price data over the

time period for the chosen company through the Yahoo Finance API

getSymbols(ticker\_symbol, env=ETF\_Data, from=sDate, to=eDate, src = "yahoo", symbol.lookup = TRUE)

[1] "AIG"

# Load a dataset with the time series of the chosen company stock prices

CompanyStockPrices = get(ticker\_symbol, envir = ETF\_Data)

# Review the layout (head and tail) of the stock price information in

dataset

head(CompanyStockPrices)

AIG.Open AIG.High AIG.Low AIG.Close AIG.Volume AIG.Adjusted

2016-12-01 63.66 64.15 63.60 64.11 5942700 58.60310

2016-12-02 64.09 64.20 63.69 63.75 5651000 58.27402

2016-12-05 64.14 64.22 63.83 64.22 4706000 58.70365

2016-12-06 64.21 64.36 63.76 64.20 4942600 58.97925

2016-12-07 64.29 64.99 64.21 64.92 5113100 59.64070

2016-12-08 64.98 66.10 64.80 65.82 5196500 60.46751

tail(CompanyStockPrices)

AIG.Open AIG.High AIG.Low AIG.Close AIG.Volume AIG.Adjusted

2017-12-21 60.25 60.67 60.02 60.08 4941500 56.34938

2017-12-22 60.27 60.36 59.11 59.31 4266300 55.62718

2017-12-26 59.22 59.79 59.19 59.43 2358300 55.73974

2017-12-27 59.29 59.56 59.19 59.54 2841400 55.84291

2017-12-28 59.70 60.02 59.44 60.00 2373000 56.27434

2017-12-29 59.97 60.21 59.58 59.58 2837600 55.88041

# Consider ‘close\_price' as my time series variable in the solution to this question

# I use the quantmod function 'Cl' to isolate the time series for closing

prices

close\_price = Cl(CompanyStockPrices)

# Review initial and final records in 'close\_price' time series

head(close\_price)

AIG.Close

2016-12-01 64.11

2016-12-02 63.75

2016-12-05 64.22

2016-12-06 64.20

2016-12-07 64.92

2016-12-08 65.82

tail(close\_price)

AIG.Close

2017-12-21 60.08

2017-12-22 59.31

2017-12-26 59.43

2017-12-27 59.54

2017-12-28 60.00

2017-12-29 59.58

###########################################################################

###########################################################################

###########################################################################**## Q.3 (Part a)**

## Using Yahoo Finance API, select a specific stock market price, apply

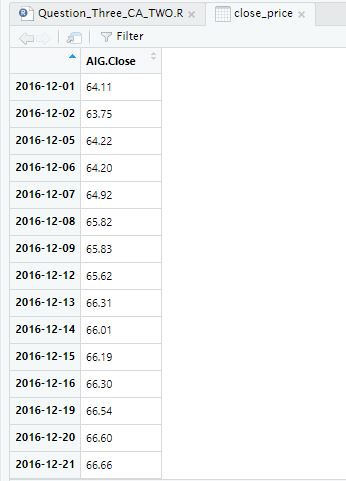
time series analysis, consider ‘close price' as your time series variable:

##(a) Validate the assumptions using graphical visualization.

## Run functions to look at the structure of the closing price dataset for our chosen stock

View(close\_price)

## <Screen shot of 'View' output>



#<Output.. of str function on 'xts' object>

str(close\_price)

An ‘xts’ object on 2016-12-01/2017-12-29 containing:

Data: num [1:272, 1] 64.1 63.8 64.2 64.2 64.9 ...

- attr(\*, "dimnames")=List of 2

..$ : NULL

..$ : chr "AIG.Close"

Indexed by objects of class: [Date] TZ: UTC

xts Attributes:

List of 2

$ src : chr "yahoo"

$ updated: POSIXct[1:1], format: "2020-06-07 16:32:33"

## Invoke the 'ts' function on the 'close\_price' time series.

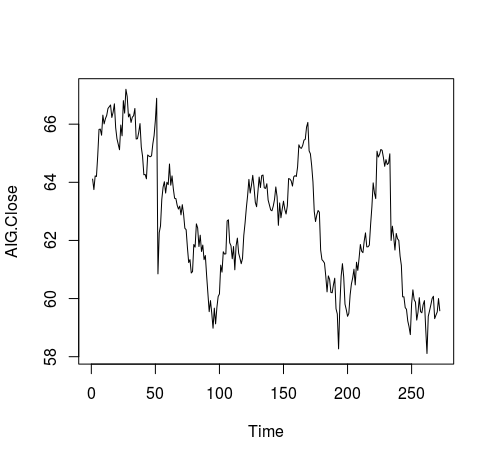
T <- ts(close\_price, frequency = 1)

## Generate the plot of the time series variable- the range represents the closing prices extracted from the time range of data (frequency = 1 so

every daily closing price is plotted).

plot(T)

## <Plot(T)>



## We can see that the time series is not particularly stationary in terms of mean or variance

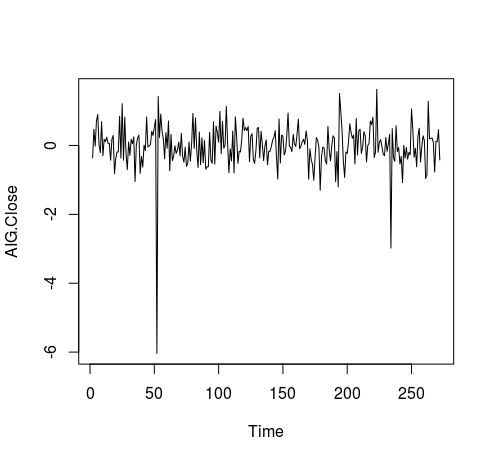
## We apply 'diff' and 'log' functions to smooth out the graph plot

## Apply 'diff' function

tssdiff=diff(T) # Stationary in mean

plot(tssdiff)

## <Diff plot graph>



## The plot of 'diff' is more stationary in mean, with an average

approximately around zero.

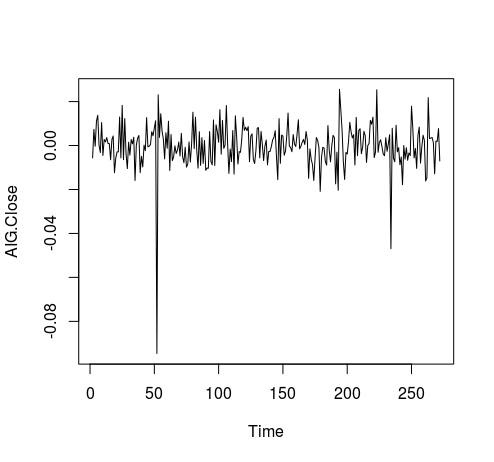
## Apply log function, then applying 'diff', to achieve a stationary

visualisation in mean and variance

tssdifflg = diff(log(T))

plot(tssdifflg)

## <Log/Diff Plot Graph>



## This graph shows mean as approximately stationary and the variance also stationary between -0.02 and 0.02, apart from a few outliers

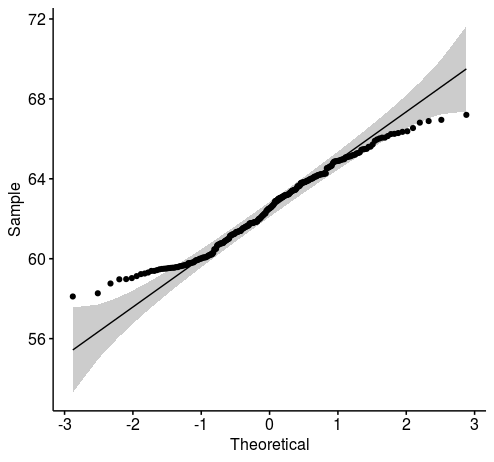
## ------------------------------------------------------------------------

## Also - run ggqqplot to graphing the data and show level of normality in the data set

## ------------------------------------------------------------------------

ggqqplot(T)

## <ggqqlog Plot Graph>



###########################################################################

**## Q.3 (Part b)**

## Fit the optimized model for ‘close price’ and provide the coefficient

estimates for the fitted model.

## To compute optimised coefficient estimates for fitted model we have two approaches :-

## 1:- Apply 'acf' and 'pacf' to get estimation of order, and also estimate parameters.

## 2:- Apply ARIMA manually

## 3:- Apply Auto ARIMA

## 4:- Select the model with the lowest AiC (Akaike Information Criterion) value and use those coefficient values

## It is necessary to apply both methods (manual and automatic) and see

which one has a lower AIC, then determine that method is optimised.

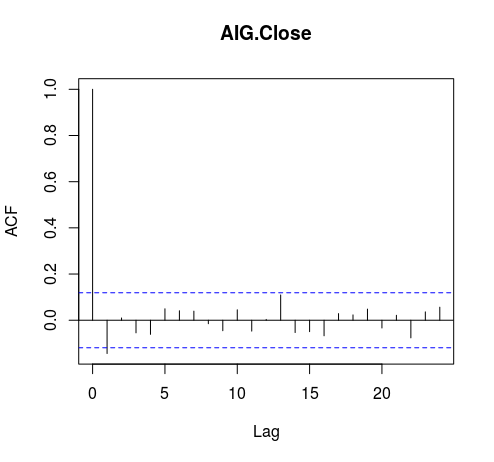
## Try and find as low a value of AIC as possible

## 1 - Apply 'acf' and 'pacf' to get estimations of 'q' and 'p'

## acf = autocorrelation function. Gives us the estimation for 'q'

acf(tssdifflg)

## <acf Plot Graph>

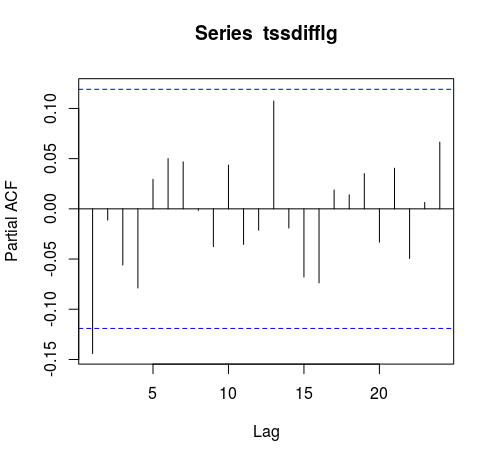


## There are two initial lags outside the boundary, therefore q = 2. (Above or below boundary line is not important).

## pacf = partial autocorrelation function. Gives is the estimation for 'p'

pacf(tssdifflg)

## <pacf Plot Graph>



## One initial lag is outside are outside the bounds, therefore p = 1

## Now use 'arima' function to fit a manual ARIMA; p = 1, (1 diff used), q = 2. The original time series with closing price is used.

## ARIMA (p,d,q) Model : Using original time series 'T'

## Parameter Estimation

manual.fit <- arima(T, c(1,1,2)) # Fitted Model

## Display value of 'manual.fit'

manual.fit

Call:

arima(x = T, order = c(1, 1, 2))

Coefficients:

ar1 ma1 ma2

-0.9986 0.8525 -0.1420

s.e. 0.0058 0.0621 0.0616

sigma^2 estimated as 0.4067: log likelihood = -262.82, aic = 533.63

## With p = 1, we see one value for the 'ar1' coefficient

## With q = 2, we see two values for the 'ma' (moving average) coefficients

## The values just under the 'ar1', 'ma1', and 'ma2' headings are the

Estimation of parameters

Coefficients:

ar1 ma1 ma2

-0.9986 0.8525 -0.1420

## We can see the aic (Akaike Information Criterion) value = 533.63

## Next we need to apply 'auto.arima' to generate a fitted model

## 'seasonal' = F - time series does not have a seasonality trend

auto.fit <- auto.arima(T, seasonal = FALSE)

auto.fit

Series: T

ARIMA(0,1,1)

Coefficients:

ma1

-0.1502

s.e. 0.0611

sigma^2 estimated as 0.4101: log likelihood=-263.27

AIC=530.54 AICc=530.58 BIC=537.74

## AIC = 530.54

## 'seasonal' = F - time series does not have a seasonality trend

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

auto.fit.T

Series: T

ARIMA(0,1,1)

Coefficients:

ma1

-0.1502

s.e. 0.0611

sigma^2 estimated as 0.4101: log likelihood=-263.27

AIC=530.54 AICc=530.58 BIC=537.74

# 'seasonal' flag makes no difference to result

## AIC = 530.54

## Automated coefficient are lower as 533.63 (Manual) > 530.54 (Auto,

non seasonal). Therefore Auto ARIMA is better than manual fitting.

**## Q.3 (Part c)**

## What is the estimated order for AR and MA?

auto.fit

Series: T

ARIMA(0,1,1)

Coefficients:

ma1

-0.1502

s.e. 0.0611

sigma^2 estimated as 0.4101: log likelihood=-263.27

AIC=530.54 AICc=530.58 BIC=537.74

## The best model shows Series: T ARIMA(0,1,1)

## Therefore the estimated order for **AR is p = 0, and MA is q = 1**

**## Q.3 (Part d)**

## Forecast h=10 step ahead prediction of wage on the plot of the

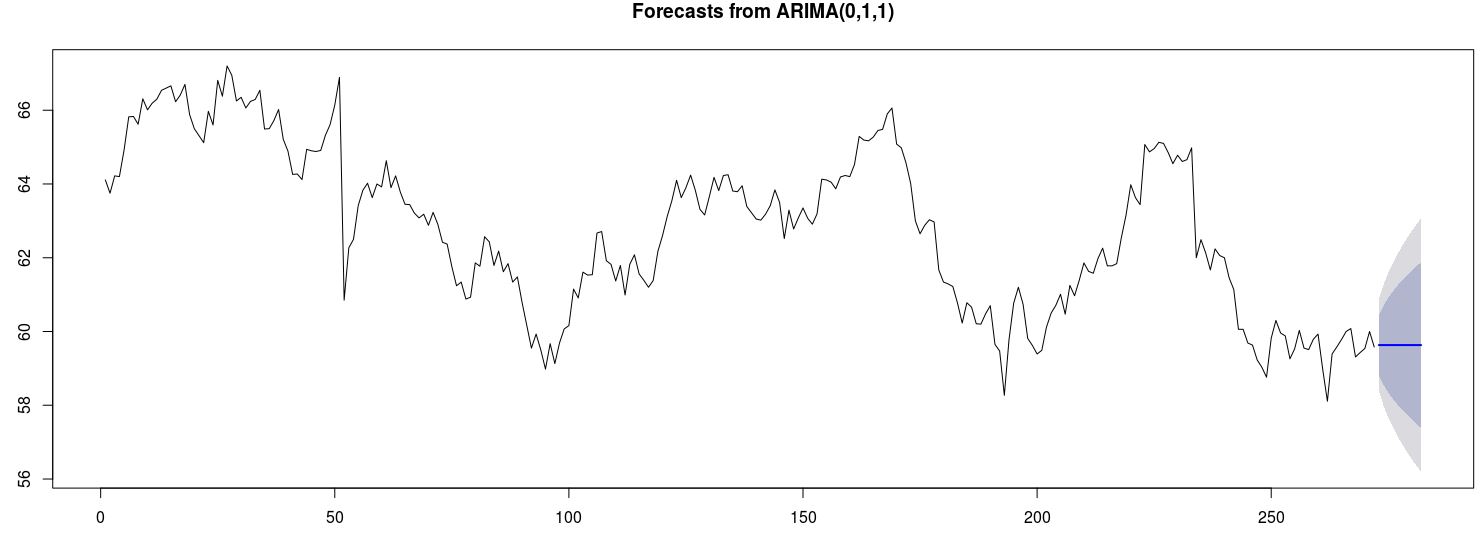
## original time series.

# The best model to use is the auto fitting - as determined in the analysis in the previous steps in the question.

auto.fcast <- forecast(auto.fit, h = 10) # Prediction for 10 step ahead

## Plot this forecast

plot(auto.fcast)



###########################################################################

## The forecast graph is still not particularly useful. However, if we

change the 'seasonal' = T parameter in the auto.arima function then the time series will incorporate a seasonality trend BUT it is necessary to 'force'

the closing price time series into a multi-seasonal time series in order to pick up the time pattern in the data.

#

## Set up multi-seasonal time series to match a working day period of 13 months (as defined when the stock data was first extracted via the Yahoo API)

stockSTS <- msts(T,seasonal.periods=c(5,270))

## Set 'D' parameter in auto.arima function so that the seasonal component in the data will be picked up

auto.fitT <- auto.arima(stockSTS, D=1, seasonal = TRUE)

## Check the aic value

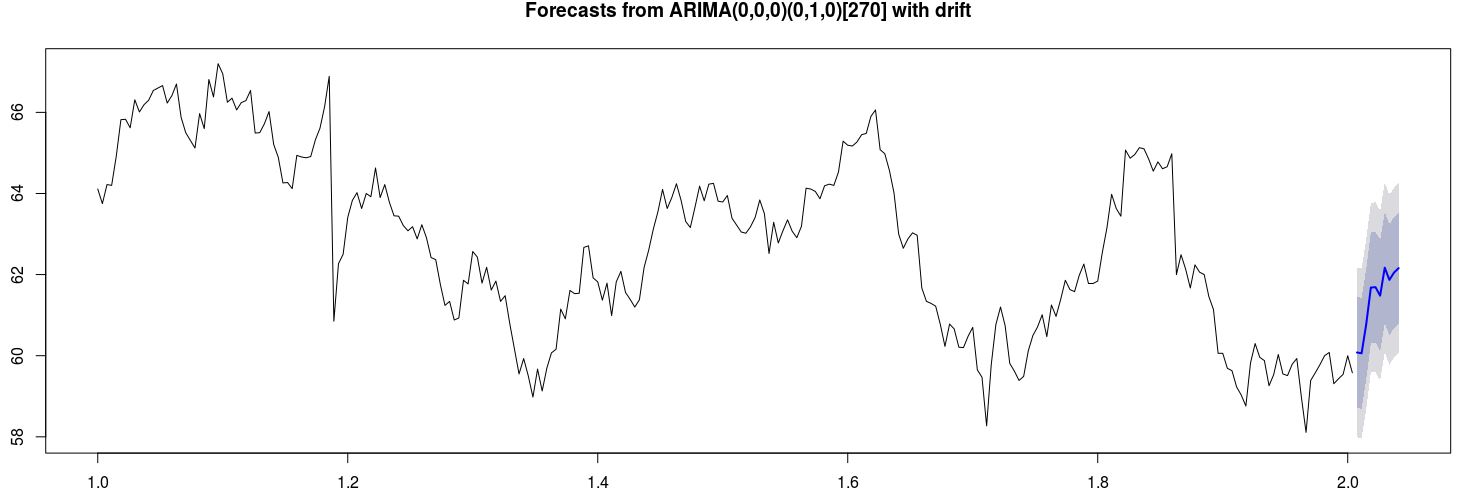
auto.fitT

auto.fcastT <- forecast(auto.fitT, h = 10) # Prediction for 10 steps ahead

## Plot this forecast

plot(auto.fcastT)

# Forecast is more realistic looking...



# Question Four

## Question 4 – from PDF

Use dataset available on http://users.stat.ufl.edu/~winner/data/nfl2008\_fga.csv

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.

**(5 Marks)**

*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). **(5 Marks)**

3. Split the dataset into two sets of variables so that **X**=( togo,kicker,ydline) and **Y**=( distance, homekick). Apply canonical correlation analysis to find the cross-correlation between **X** and **Y**. What is the correlation between ***ydline*** and ***distance***? **(5 Marks)**

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

**(6 Marks)**

**(Total: 20 Marks)**

## Output from RStudio Cloud Console

> ## CA Two Advanced Data Analytics : Module Code B8IT109

> ## Student Name : Ciaran Finnegan

>

> ## Student Number : 10524150

>

> ## June 2020

>

>

**> ## Question Four - LDA, PCA, K-Means, Canonical Correlation**

>

> ## Use dataset available on http://users.stat.ufl.edu/~winner/data/nfl2008\_fga.csv

> ## Perform initial data load, analysis, and clean up operations before

Starting Question 4 solution

>

>

>

> ## Load MASS library to use LDA function

> library(MASS)

> library(CCA)

>

> # Load 'factoextra' for visualization - Scree plot

> #install.packages("factoextra")

> library(factoextra)

>

>

>

> ## Read in the NFL dataset

> link='http://users.stat.ufl.edu/~winner/data/nfl2008\_fga.csv'

> datasetNFL=read.csv(link)

> ## Brief Review of number of rows, head and tail of dataset records

> ## the and structure of dataset

> nrow(datasetNFL)

[1] 1039

> head(datasetNFL)

GameDate AwayTeam HomeTeam qtr min sec kickteam def down togo kicker ydline name distance homekick kickdiff timerem offscore

1 20081130 IND CLE 1 47 2 IND CLE 4 11 15 12 A.Vinatieri 30 0 -3 2822 0

2 20081005 IND HOU 1 54 47 IND HOU 4 3 15 28 A.Vinatieri 46 0 0 3287 0

3 20081228 TEN IND 1 45 20 IND TEN 4 3 15 10 A.Vinatieri 28 1 7 2720 7

4 20081012 BAL IND 1 45 42 IND BAL 4 1 15 19 A.Vinatieri 37 1 14 2742 14

5 20080907 CHI IND 1 50 56 IND CHI 4 21 15 21 A.Vinatieri 39 1 0 3056 0

6 20081116 HOU IND 1 50 43 IND HOU 4 7 15 22 A.Vinatieri 40 1 -3 3043 0

defscore season GOOD Missed Blocked

1 3 2008 1 0 0

2 0 2008 1 0 0

3 0 2008 1 0 0

4 0 2008 1 0 0

5 0 2008 1 0 0

6 3 2008 1 0 0

> tail(datasetNFL)

GameDate AwayTeam HomeTeam qtr min sec kickteam def down togo kicker ydline name distance homekick kickdiff timerem offscore

1034 20081102 TB KC 5 -5 27 TB KC 4 1 36 16 M.Bryant 34 0 0 -273 27

1035 20081102 GB TEN 5 -6 41 TEN GB 3 2 37 23 R.Bironas 41 1 0 -319 16

1036 20081211 NO CHI 5 -3 19 CHI NO 2 13 7 18 R.Gould 35 1 0 -161 24

1037 20081222 GB CHI 5 -4 33 CHI GB 3 10 7 20 R.Gould 38 1 0 -207 17

1038 20081116 PHI CIN 5 -15 13 CIN PHI 4 5 8 29 S.Graham 47 1 0 -887 13

1039 20081019 NYJ OAK 5 -13 35 OAK NYJ 4 11 28 39 S.Janikowski 57 1 0 -745 13

defscore season GOOD Missed Blocked

1034 27 2008 1 0 0

1035 16 2008 1 0 0

1036 24 2008 1 0 0

1037 17 2008 1 0 0

1038 13 2008 0 1 0

1039 13 2008 1 0 0

> str(datasetNFL)

'data.frame': 1039 obs. of 23 variables:

$ GameDate: int 20081130 20081005 20081228 20081012 20080907 20081116 20081123 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 14 16 16 24 ...

$ def : Factor w/ 32 levels "ARI","ATL","BAL",..: 8 13 31 3 6 13 26 10 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 ...

>

>

> ## Minor Clean up of NFL dataset

> sum(is.na(datasetNFL))

[1] 4

> datasetNFL <- na.omit(datasetNFL)

> sum(is.na(datasetNFL))

[1] 0

> nrow(datasetNFL) # Confirm rows after missing data removed = 1037

[1] 1037

>

>

**> ## Q. 4(Part 1)**

>

> ## Use LDA to classify the dataset into a small number of classes so that at least 90% of the information of the 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.

>

>

>

> # Display the values for 'qtr'

> table(datasetNFL$qtr)

1 2 3 4 5

214 373 182 255 13

>

>

>

>

> ## Use LDA function to classify dataset. The output variable is 'qtr' and the input variables are > ##'togo', 'kicker', and 'ydline'.

> datasetNFL.lda <- lda(qtr~togo+kicker+ydline, data=datasetNFL)

> datasetNFL.lda

Call:

lda(qtr ~ togo + kicker + ydline, data = datasetNFL)

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

>

> ## Two LDs are required - LD1 and LD2 - to explain at least 90% of formation of the NFL dataset

> ## Reading values under the 'Proportion of trace:' output I can see...

> ## LD1 explains 61.5%. LD2 explains a further 32.2%. Hence **LD1 and LD2**

**will explain 93.7 % together**.

>

>

>

**> ## Q. 4(Part 2)**

>

> ## Apply PCA, and identify the important principle components involving

at least 90% of dataset variation.

> ## Explain your decision strategy?

>

> ## We only use the input variables for the PCA question. This analysis is a type of 'unsupervised' learning.

> ## Just focusing on raw data on 'togo', 'kicker', and 'ydline' from

> ## dataset and extracting PCs from the correlation matrix

>

> ## I could have used the cbind(datasetNFL$togo, datasetNFL$kicker, datasetNFL$ydline) function but I want to preserve the dataset column names

> datasetNFL2 = datasetNFL[10:12] # datasetNFL$togo, datasetNFL$kicker, datasetNFL$ydline

> fit <- princomp(datasetNFL2, cor = TRUE)

> summary(fit) # Print variance

Importance of components:

Comp.1 Comp.2 Comp.3

Standard deviation 1.146726 0.9998433 0.8278479

Proportion of Variance 0.438327 0.3332289 0.2284441

Cumulative Proportion 0.438327 0.7715559 1.0000000

>

>

> ## Looking at the 'Cumulative Proportion' output line we can see that

Comp1 captures 43.8% of dataset variation.

> ## Comp 1 and Comp2 together capture 77.2% (approx.) of dataset variation.

> ## However, **all three components (Comp1, Comp2, and Comp3) are important to capture 90% of the dataset variation.**

>

>

> ## Plot principle components versus their variance

> ## (Hint: to sketch the plot use the Scree plot).

>

> #

> ## Use function to extract loadings for factor analysis - small loadings are usually not printed.

> loadings(fit) # PC loadings

Loadings:

Comp.1 Comp.2 Comp.3

togo 0.707 0.707

kicker 0.999

ydline 0.706 -0.707

Comp.1 Comp.2 Comp.3

SS loadings 1.000 1.000 1.000

Proportion Var 0.333 0.333 0.333

Cumulative Var 0.333 0.667 1.000

>

> #

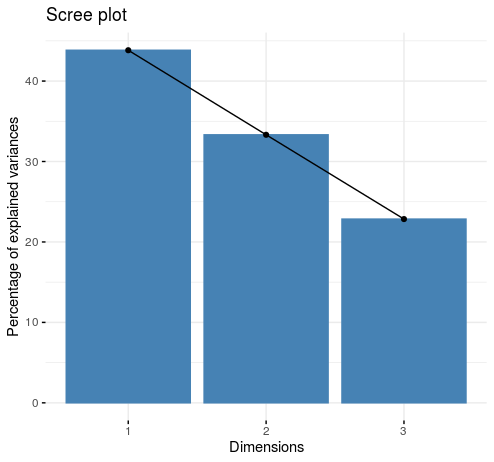
> ## Visualize eigenvalues (scree plot). Show the percentage of variances

explained by each principal component.

> fviz\_eig(fit)

>

> # <Scree Plot Graph>



>

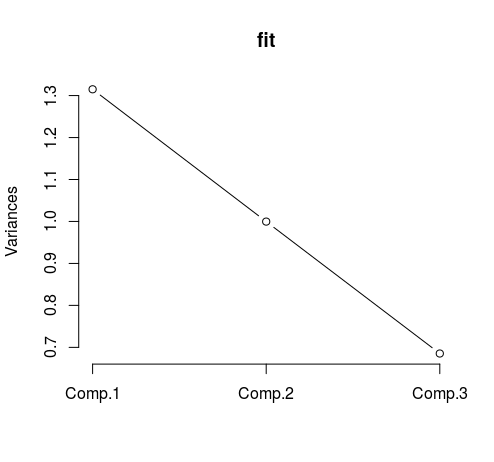
> #

> plot(fit, type = "lines") # Another Scree Plot view. A plot of variation

>

> #

> ## <Plot Graph>



> ## Component 2 is just on or over the Variance value of '1' so I can

determine that Component 1 and 2 are the most important components to

consider.

> ## However, in order to meet the 90% level of variance requested in this question we still need to consider Component 3.

>

>

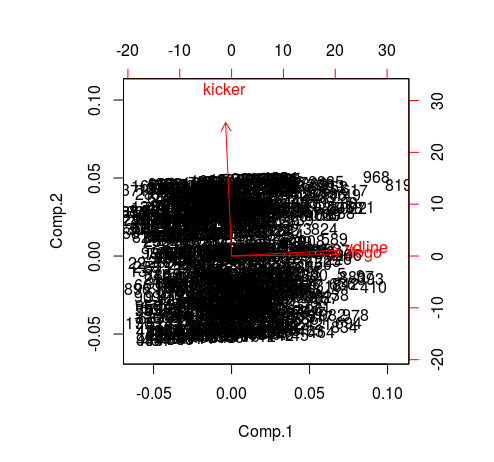
> biplot(fit) # Graph that shows two components and role of each variable

(relationship between components and variables)

>

> #

> # <BiPlot Graph>



> ## For example, as you increase 'ydline' there is an noticeable increase in Component 1

> ## An increase in 'ydline' shows a very minor increase in Component 2

>

> #

> ## The Plots confirms that all three components are important to capture 90%.

> ## There is no 'bend' in the line indicating that higher components

contribute less to the capture of dataset variation

**> ## Q. 4(Part 3)**

>

> ## Split the dataset into two sets of variables so that

X=(togo, kicker, ydline) and Y=(distance, homekick).

> ## Apply canonical correlation analysis to find the cross-correlation

between X and Y.

>

>

> ## Set up 'X' variable

> X <- cbind(datasetNFL$togo, datasetNFL$kicker, datasetNFL$ydline)

>

> ## Set up 'Y' variable

> Y <- cbind(datasetNFL$distance, datasetNFL$homekick)

>

> ## Run 'cor' function to produce Correlation Matrix

> cor(X, Y)

[,1] [,2]

[1,] 0.315641454 -0.04838438

[2,] -0.001951722 -0.02363159

[3,] 0.998947222 0.04295427

>

>

> ## What is the correlation between 'ydline' and 'distance'?

>

> ## Read three down the X value and one across the Y value

> ## The correlation between 'ydline' and 'distance' is equal to

'0.998947222'

> ## This value shows a high level of correlation between the 'ydline' and 'distance' values

>

>

>

>

>

>

>

**> ## Q. 4(Part 4)**

>

> ## Use K-means clustering analysis to identify the most important classes.

> ## How many classes do you select? Why?

>

> ## Again consider the input variables. We use the 'datasetNFL2' dataset

because I want to just consider the 'togo', 'kicker', and 'ydline' input

variables.

>

>

>

> # Generate the plot K-Means clustering

> ## Write function for plot generation

> wssplot <- function(datasetNFL2, nc=10, seed=2343){

+

+ wss <- (nrow(datasetNFL2)-1) \* sum(apply(datasetNFL2, 2, var))

+

+ for (i in 2 : nc){

+

+ set.seed(seed)

+ wss[i] <- sum(kmeans(datasetNFL2, centers = i)$withinss)

+

+ }

+

+ plot(1:nc, wss, type = "b", xlab = "Numbers of Clusters", ylab = "Within Groups Sum of Squares")

+

+

+ }

>

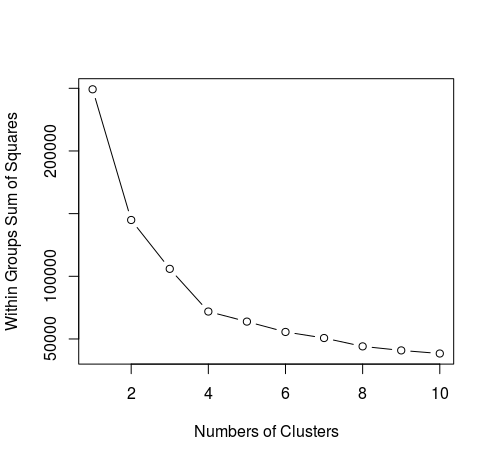
> # Invoke plot function

> wssplot(datasetNFL2, nc = 10)

> # Use a default of number of classes = 10 to start the analysis

>

> # <K-Mean Cluster Graph>



>

>

> #

> ## **In the Cluster graph we can see a definite 'elbow' at Number of**

**Clusters = 4.**

> ## After Cluster 4 the changes in variation are noticeably less

> ## Therefore the main cluster are clusters 1 through to cluster 4.

> ## We would select four classes as an answer to this question.

>

>

> ## K-Means : Clustering Analysis on NFL Dataset

> k.means.fit <- kmeans(datasetNFL2, 4) # k = 4, the number of classes in

type (see above)

> attributes(k.means.fit)

$names

[1] "cluster" "centers" "totss" "withinss" "tot.withinss" "betweenss" "size" "iter" "ifault"

$class

[1] "kmeans"

>

> ## Centroids(arithmetic mean)

> k.means.fit$centers

togo kicker ydline

1 5.268293 28.439024 10.22997

2 7.961373 30.841202 27.05579

3 5.952586 8.336207 10.53879

4 7.859649 10.536842 26.82807

>

> ## Cluster size - shows the breakdown of the number of data points in the NFL dataset into my chosen cluster grouping

> k.means.fit$size

[1] 287 233 232 285

>

> ## All value above sum to 1037, which is the size of the dataset (rows)