Assignment 1

10515229

## Introduction:

My answers to Assignment 1 for B9IS106 are found below.

### Question 1:

#### 1.(a)

In 1.(a), To explore the general features of dataset, the commands are as follows:

#Quick inspection of raw data  
head(airquality) #To view the first 6 rows of the dataset

## 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(airquality) #To view the last 6 rows of the dataset

## 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

View(airquality) #To view the dataset  
#Type of data & its dimensions  
class(airquality) #To prints the vector of names of classes

## [1] "data.frame"

nrow(airquality) #To find the number of rows in the dataset

## [1] 153

ncol(airquality) #To find the number of columns in the dataset

## [1] 6

length(airquality[,1]) #To find the number of rows

## [1] 153

length(airquality)#To find the number of columns

## [1] 6

names(airquality) #To view names of columns

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

str(airquality) #To view the structure of the dataset

## '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 ...

help("airquality")#To view the Description, Usage, Format, Details, Source, Reference, Examples in R Documentation

#### 1.(b)

In 1.(b), To perform data Cleansing the commands are as follows:

my.data<-airquality  
  
head(my.data) #To view the first 6 rows of the dataset

## 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(my.data) #To view the last 6 rows of the dataset

## 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

is.na(my.data) #To check the dataset has null values #if TRUE= null, FALSE= not null

## Ozone Solar.R Wind Temp Month Day  
## [1,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [2,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [3,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [4,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [5,] TRUE TRUE FALSE FALSE FALSE FALSE  
## [6,] FALSE TRUE FALSE FALSE FALSE FALSE  
## [7,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [8,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [9,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [10,] TRUE FALSE FALSE FALSE FALSE FALSE  
## [11,] FALSE TRUE FALSE FALSE FALSE FALSE  
## [12,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [13,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [14,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [15,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [16,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [17,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [18,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [19,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [20,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [21,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [22,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [23,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [24,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [25,] TRUE FALSE FALSE FALSE FALSE FALSE  
## [26,] TRUE FALSE FALSE FALSE FALSE FALSE  
## [27,] TRUE TRUE FALSE FALSE FALSE FALSE  
## [28,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [29,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [30,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [31,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [32,] TRUE FALSE FALSE FALSE FALSE FALSE  
## [33,] TRUE FALSE FALSE FALSE FALSE FALSE  
## [34,] TRUE FALSE FALSE FALSE FALSE FALSE  
## [35,] TRUE FALSE FALSE FALSE FALSE FALSE  
## [36,] TRUE FALSE FALSE FALSE FALSE FALSE  
## [37,] TRUE FALSE FALSE FALSE FALSE FALSE  
## [38,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [39,] TRUE FALSE FALSE FALSE FALSE FALSE  
## [40,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [41,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [42,] TRUE FALSE FALSE FALSE FALSE FALSE  
## [43,] TRUE FALSE FALSE FALSE FALSE FALSE  
## [44,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [45,] TRUE FALSE FALSE FALSE FALSE FALSE  
## [46,] TRUE FALSE FALSE FALSE FALSE FALSE  
## [47,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [48,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [49,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [50,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [51,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [52,] TRUE FALSE FALSE FALSE FALSE FALSE  
## [53,] TRUE FALSE FALSE FALSE FALSE FALSE  
## [54,] TRUE FALSE FALSE FALSE FALSE FALSE  
## [55,] TRUE FALSE FALSE FALSE FALSE FALSE  
## [56,] TRUE FALSE FALSE FALSE FALSE FALSE  
## [57,] TRUE FALSE FALSE FALSE FALSE FALSE  
## [58,] TRUE FALSE FALSE FALSE FALSE FALSE  
## [59,] TRUE FALSE FALSE FALSE FALSE FALSE  
## [60,] TRUE FALSE FALSE FALSE FALSE FALSE  
## [61,] TRUE FALSE FALSE FALSE FALSE FALSE  
## [62,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [63,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [64,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [65,] TRUE FALSE FALSE FALSE FALSE FALSE  
## [66,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [67,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [68,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [69,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [70,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [71,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [72,] TRUE FALSE FALSE FALSE FALSE FALSE  
## [73,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [74,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [75,] TRUE FALSE FALSE FALSE FALSE FALSE  
## [76,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [77,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [78,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [79,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [80,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [81,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [82,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [83,] TRUE FALSE FALSE FALSE FALSE FALSE  
## [84,] TRUE FALSE FALSE FALSE FALSE FALSE  
## [85,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [86,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [87,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [88,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [89,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [90,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [91,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [92,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [93,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [94,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [95,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [96,] FALSE TRUE FALSE FALSE FALSE FALSE  
## [97,] FALSE TRUE FALSE FALSE FALSE FALSE  
## [98,] FALSE TRUE FALSE FALSE FALSE FALSE  
## [99,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [100,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [101,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [102,] TRUE FALSE FALSE FALSE FALSE FALSE  
## [103,] TRUE FALSE FALSE FALSE FALSE FALSE  
## [104,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [105,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [106,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [107,] TRUE FALSE FALSE FALSE FALSE FALSE  
## [108,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [109,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [110,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [111,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [112,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [113,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [114,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [115,] TRUE FALSE FALSE FALSE FALSE FALSE  
## [116,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [117,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [118,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [119,] TRUE FALSE FALSE FALSE FALSE FALSE  
## [120,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [121,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [122,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [123,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [124,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [125,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [126,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [127,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [128,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [129,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [130,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [131,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [132,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [133,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [134,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [135,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [136,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [137,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [138,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [139,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [140,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [141,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [142,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [143,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [144,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [145,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [146,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [147,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [148,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [149,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [150,] TRUE FALSE FALSE FALSE FALSE FALSE  
## [151,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [152,] FALSE FALSE FALSE FALSE FALSE FALSE  
## [153,] FALSE FALSE FALSE FALSE FALSE FALSE

any(is.na(my.data)) #To check the dataset has null values

## [1] TRUE

sum(is.na(my.data)) #To find total of null values in a dataset

## [1] 44

str(my.data) #To view the structure of the dataset

## '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 ...

colSums(is.na(my.data)) #To view the which column has null values and its total per column

## Ozone Solar.R Wind Temp Month Day   
## 37 7 0 0 0 0

nrow(my.data) #No. of rows before cleansing

## [1] 153

my.data.clean<-na.omit(my.data) #To remove all rows containing NA  
  
any(is.na(my.data.clean)) #To check the dataset has any null values after cleansing

## [1] FALSE

nrow(my.data.clean) #To check the number of rows not affected after cleansing

## [1] 111

#### 1.(c)

In 1.(c) Computation of central and variational measures. The commands are as follows:

head(my.data.clean)

## 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

View(my.data.clean)  
temp<-my.data.clean$Temp #To extract temp column from the cleansed dataset  
  
temp\_mean= mean(temp) #To find mean of temp  
temp\_mean

## [1] 77.79279

temp\_median=median(temp) #To find median of temp  
temp\_median

## [1] 79

temp\_occurence<-table(as.vector(temp)) #To find occurence of temp  
temp\_occurence

##   
## 57 58 59 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82   
## 1 1 2 3 2 1 2 2 2 3 4 2 1 3 3 4 2 2 6 4 4 3 3 10 7   
## 83 84 85 86 87 88 89 90 91 92 93 94 96 97   
## 3 3 3 5 3 2 2 3 1 3 2 2 1 1

names(temp\_occurence)[temp\_occurence==max(temp\_occurence)] #To find mode of temp

## [1] "81"

temp\_sd=sd(temp) #To find standard deviation of Temp  
temp\_sd

## [1] 9.529969

temp\_var=var(temp) #To find variance of temp  
temp\_var

## [1] 90.82031

temp\_sd\_sqrt\_var=sqrt(temp\_var) #sqrt of var = sd  
temp\_sd\_sqrt\_var

## [1] 9.529969

temp\_range=range(temp) #To find Range of temp  
temp\_range

## [1] 57 97

quantile(temp) #To find quantile of temp

## 0% 25% 50% 75% 100%   
## 57.0 71.0 79.0 84.5 97.0

summary(temp) #To find Mininum & Maximum value, 1st, 2nd, 3rd quantile , mean of temp

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 57.00 71.00 79.00 77.79 84.50 97.00

#### 1.(d)

In 1.(d) boxplot technique to detect outlier of ‘wind’ attribute

Wind=my.data.clean$Wind  
   
sort\_Wind= Wind[order(Wind)] #To sort the column wind   
  
fivedata\_Wind=fivenum(sort\_Wind) #fivenum returns a vector of length five (minimum, lower-hinge, median, upper-hinge, maximum)  
  
fivedata\_Wind

## [1] 2.3 7.4 9.7 11.5 20.7

IQR=fivedata\_Wind[4]-fivedata\_Wind[2] #InterQuartile Range  
IQR

## [1] 4.1

QLw=fivedata\_Wind[2]-1.5\*IQR #Lower Quartile Range  
QLw

## [1] 1.25

QUw=fivedata\_Wind[4]+1.5\*IQR #Upper Quartile Range  
QUw

## [1] 17.65

Int=c(QLw,QUw)  
vect=c()  
length(Wind) #number of rows in column Wind

## [1] 111

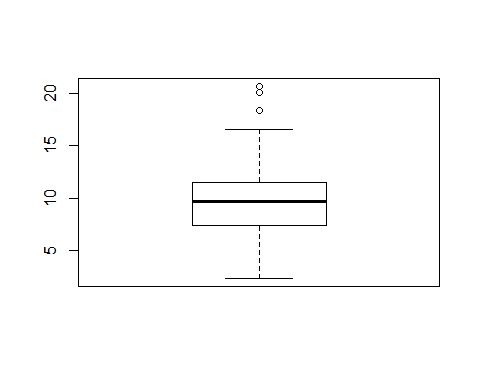
#The values outside Inter & Outer Quartile range are outliers  
for(i in Wind)  
{if((i>QUw)&(i>QLw))  
{vect=c(vect,i)}  
 }  
vect

## [1] 20.1 18.4 20.7

length(vect) #Number of outliers in column Wind

## [1] 3

View(vect) #To view the outliers  
#Alternatively we can find the outliers using the statistics of boxplot  
boxplot(Wind)



boxplot.stats(Wind)

## $stats  
## [1] 2.3 7.4 9.7 11.5 16.6  
##   
## $n  
## [1] 111  
##   
## $conf  
## [1] 9.085135 10.314865  
##   
## $out  
## [1] 20.1 18.4 20.7

#The last line of the statistics has the outliers

#### 2(a)

In 2(a) Generalised Linear Model

dataset <- read.csv("http://users.stat.ufl.edu/~winner/data/nfl2008\_fga.csv")  
library(caTools) # useful to split data to training and test datasets  
  
set.seed(1456)  
n=nrow(dataset)  
n

## [1] 1039

indexes = sample(n,n\*(80/100)) #To split the dataset  
trainset=dataset[indexes,]# training dataset  
testset =dataset[-indexes,]# test dataset  
#(a)   
table(trainset$homekick)

##   
## 0 1   
## 417 414

By identifing the output variable the appropriate glm to be used is logistic regression as the homekick column contains binary outcomes #### 2(b)

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

##   
## Call:  
## glm(formula = homekick ~ togo + ydline + kicker, family = "binomial",   
## data = trainset)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.3834 -1.1695 -0.9385 1.1692 1.4355   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 0.140714 0.209486 0.672 0.5018   
## togo -0.042133 0.017288 -2.437 0.0148 \*  
## ydline 0.011670 0.007518 1.552 0.1206   
## kicker -0.004444 0.006221 -0.714 0.4750   
## ---  
## 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: 1141.9 on 825 degrees of freedom  
## (2 observations deleted due to missingness)  
## AIC: 1149.9  
##   
## Number of Fisher Scoring iterations: 4

To consider the significanse of the variable p-value should be less than alpha since p-value for togo is less than alpha =0.05 therefore, only the input variable togo is significant

#### 2(C)

Prediction of the test dataset using the trained model

pred=predict(fit,testset)  
predictedvalues=rep(0,length(pred))  
predictedvalues[pred>0.5]=1  
actual=testset$homekick  
df=data.frame(actual,predictedvalues)  
View(df)  
head(df)

## actual predictedvalues  
## 1 0 0  
## 2 0 0  
## 3 1 0  
## 4 1 0  
## 5 1 0  
## 6 1 0

#### 2(D)

confusion\_matrix=table(predictedvalues, actual)  
confusion\_matrix

## actual  
## predictedvalues 0 1  
## 0 108 100

View(confusion\_matrix)

The tabular representation of Actual vs Predicted values can be obtained in confusion matrix Accuracy of the model can be obtained by the below formula

accuracy=mean(predictedvalues == actual) # correctness of prediction  
accuracy

## [1] 0.5192308

#### 3(A)

Assumption Validation using graphical visualization.

library(forecast)  
link='http://www.stat.ufl.edu/~winner/data/wage\_cpi.csv'  
data=read.csv(link)  
head(data)

## Year yr\_month ser\_month wage cpi  
## 1 2000 1 1 172.6 373.06  
## 2 2000 2 2 174.8 365.15  
## 3 2000 3 3 190.7 358.72  
## 4 2000 4 4 213.6 342.34  
## 5 2000 5 5 203.0 336.22  
## 6 2000 6 6 210.4 326.95

str(data)

## 'data.frame': 129 obs. of 5 variables:  
## $ Year : int 2000 2000 2000 2000 2000 2000 2000 2000 2000 2000 ...  
## $ yr\_month : int 1 2 3 4 5 6 7 8 9 10 ...  
## $ ser\_month: int 1 2 3 4 5 6 7 8 9 10 ...  
## $ wage : num 173 175 191 214 203 ...  
## $ cpi : num 373 365 359 342 336 ...

x=data$wage  
class(x)

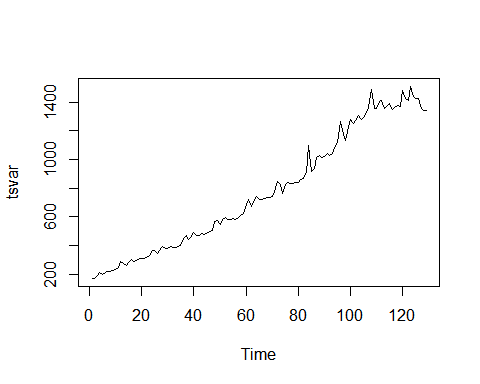
## [1] "numeric"

tsvar <- ts(x,frequency=1)#to convert to timeseries data  
class(tsvar)

## [1] "ts"

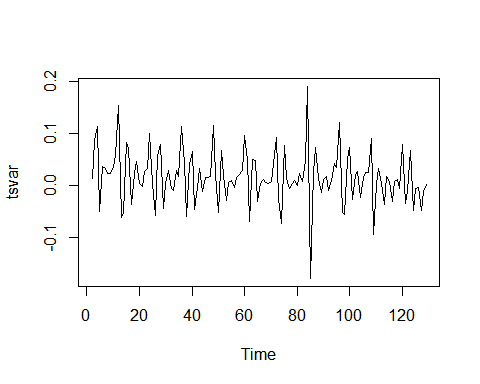
Validate the assumption 1)Stationarity 2)Normality

plot(tsvar)

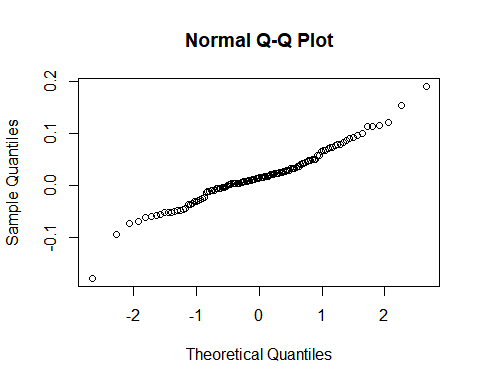
 By plotting the timeseries data we can see that it is non-stationary in mean

To make the timeseries data to stationary in mean diff function is applied

tsvar=diff(log(tsvar)) # to make ts stationary in mean and variance use diff & log  
plot(tsvar)



qqnorm(tsvar) # if the graph is straight slope line the time series data is normal

 From the above plots the assumption of timeseries data to be 1)Stationarity 2)Normality is true

#### 3(b)

Fit the optimized model for ‘wage’ and provide the coefficient estimates for the fitted model.

By using auto arima The one with the lowest BIC and AIC should be our choice of the optimized model

#Seasonal =F & T test is performed to check AIC & BIC   
auto.fit<-auto.arima(tsvar,seasonal=T) #automatic way to find   
auto.fit

## Series: tsvar   
## ARIMA(2,1,1)   
##   
## Coefficients:  
## ar1 ar2 ma1  
## -0.3022 -0.4719 -0.9350  
## s.e. 0.0794 0.0787 0.0323  
##   
## sigma^2 estimated as 0.001986: log likelihood=214.55  
## AIC=-421.1 AICc=-420.77 BIC=-409.72

auto.fit<-auto.arima(tsvar,seasonal=F) #automatic way to find   
auto.fit

## Series: tsvar   
## ARIMA(2,1,1)   
##   
## Coefficients:  
## ar1 ar2 ma1  
## -0.3022 -0.4719 -0.9350  
## s.e. 0.0794 0.0787 0.0323  
##   
## sigma^2 estimated as 0.001986: log likelihood=214.55  
## AIC=-421.1 AICc=-420.77 BIC=-409.72

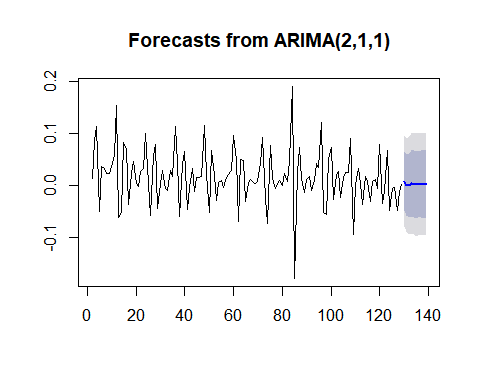
Akaike information criterion(AIC) & Bayesian information criterion(BIC) is same for both the tests

#### 3 (c)

Based on the output of auto arima The order of AR is 2 The order of AR is 1 #### 3 (d) Forecast h=10 step ahead prediction of wage on the plot of the original time series

Visualization the prediction for 10 years

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



#### 4(a)

library(MASS)  
dataset <- read.csv("http://users.stat.ufl.edu/~winner/data/nfl2008\_fga.csv")  
View(dataset)  
dataset.lda<-lda(qtr~togo+ydline+kicker,data=dataset)  
dataset.lda

## Call:  
## lda(qtr ~ togo + ydline + kicker, data = dataset)  
##   
## Prior probabilities of groups:  
## 1 2 3 4 5   
## 0.20636451 0.35969142 0.17550627 0.24590164 0.01253616   
##   
## Group means:  
## togo ydline kicker  
## 1 6.481308 17.22897 19.64486  
## 2 6.973190 19.30027 18.77212  
## 3 6.543956 19.03297 19.96703  
## 4 6.792157 18.53725 20.20000  
## 5 5.923077 19.53846 22.61538  
##   
## Coefficients of linear discriminants:  
## LD1 LD2 LD3  
## togo 0.06665269 0.12498308 0.20996464  
## ydline 0.07726467 -0.07173243 -0.02257770  
## kicker -0.04134867 -0.06009657 0.05013225  
##   
## Proportion of trace:  
## LD1 LD2 LD3   
## 0.615 0.322 0.063

Two LDA’s as most important can be chosen as the sum of LD1+LD2 is >90 variance so select LD1 and LD2

No. of LDA = min(# of categories in output-1,# of input variable) = min(5-1,3) No. of LDA =3

LDA1 = 0.066togo + 0.077ydline - 0.041kicker LDA2 = 0.124togo - 0.071ydline - 0.060kicker

#### 4(b)

PCA

head(dataset)

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

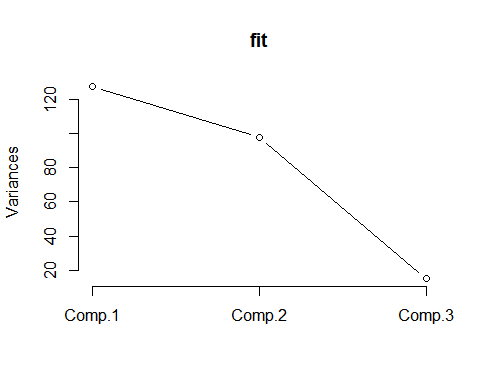
data=cbind(dataset$togo,dataset$ydline,dataset$kicker)  
data1=na.omit(data)  
fit<-princomp(data1)  
summary(fit) #print variance acconted for,

## Importance of components:  
## Comp.1 Comp.2 Comp.3  
## Standard deviation 11.2926696 9.8769392 3.90137351  
## Proportion of Variance 0.5306904 0.4059689 0.06334073  
## Cumulative Proportion 0.5306904 0.9366593 1.00000000

loadings(fit) #pc loadings

##   
## Loadings:  
## Comp.1 Comp.2 Comp.3  
## [1,] 0.157 0.988  
## [2,] 0.987 -0.157  
## [3,] -1.000   
##   
## 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

plot(fit,type="lines") #scree plot

 This shows that first principal component explains 33.33% variance. Second component explains 66.7% variance. Third component explains 100% variance. So, we select first two components as important principle components involving at least 90% of dataset variation

It also can be answered based on scree plot. Based on the scree plot the second component exceeds 90%.Therefore first 2 components are chosen.