|  |  |
| --- | --- |
|  | |
| Data Advanced Data Analytics  Advanced Data Aanlyics Exam | |
| Module code : B8IT109 | |
| Ciaran Finnegan  Student No : 10524150  15/06/2020 |  |
|  |  |

Table of Contents

1 ADA – Exam Paper Submission 3

1.1 Course Details 3

1.2 Exam Declaration 3

1.3 Notes on Exam Submission Format 3

2 Question Four – Time Series 4

2.1 Question 1 – from PDF 4

2.2 Output From RStudio Cloud Console – Question Four 5

3 Question One – Probability Models 16

3.1 Question 1 – from PDF 16

3.2 Output from RStudio Cloud Console 17

4 Question Three 20

4.1 Question 3 – from PDF 20

4.2 Output from RStudio Cloud Console 21

5 Question Four 22

5.1 Question 4 – from PDF 22

5.2 Output from RStudio Cloud Console 23

# ADA – Exam Paper Submission

## Course Details

|  |  |
| --- | --- |
| Module Code | B8IT109 (B8IT109\_1920\_SME3\_BHD08DNW) |
| Module Name | Advanced Data Analytics – Fri/Sat Part-time March 2019 Intake |
| Date | 15th June 2020 |
| Student number | 10524150 |
| Student name | Ciaran Finnegan |

## Exam Declaration

*By uploading this exam from my Moodle account I Ciaran Finnegan am confirming that this document is all my own work.*

*I understand that DBS will carry out checks such as text-matching (via Urkund), bench-marking and viva voce exams in order to verify the authenticity of submissions.*

## Notes on Exam Submission Format

* The exam answers are in two parts;
  + Part 1 is a reproduction of the exam question.
  + Part 2 is the answer to the exam question and is a copy/paste of the contents of my RStudio console.

# Question Four – Time Series

## Question 1 – from PDF

Use dataset available on

http://www.stat.ufl.edu/~winner/data/clotthes\_expend.csv , apply time

series analysis, consider **sales.b** as your time series variable:

1. Validate the assumptions using graphical visualization.

**(5 Marks)**

b) Fit the optimized model for **sales.b** and provide the

coefficient estimates for the fitted model. **(7.5 Marks)**

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

**(5 Marks)**

d) Forecast h=10 step ahead prediction of **sales.b** on the plot of

the original time series.

**(7.5 Marks)**

**(Total: 25 Marks)**

## Output From RStudio Cloud Console – Question Four

## Exam Advanced Data Analytics : Module Code B8IT109

## Advanced Data Analytics : Module Code B8IT109

> ## Student Name : Ciaran Finnegan 10524150

>

> ## Exam Submission

>

> ## June 15th 2020

>

> ## Question 4 : Time Series Analysis

>

> ## Use dataset available on

> ## http://www.stat.ufl.edu/~winner/data/clotthes\_expend.csv ,

> ## apply time series analysis,

> ## consider sales.b as your time series variable:

>

>

>

> ## Read in the CEP dataset

> datasetCEP <- read.csv("clotthes\_expend.csv")

>

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

> ## the and structure of dataset

> nrow(datasetCEP)

[1] 85

> head(datasetCEP)

year sales.b price.index sales.index pop.m realgdp.b ad.gdppct

1 1929 9.0 27.859 32.30554 121.77 1,056.60 2.8

2 1930 7.8 26.591 29.33323 123.08 966.7 2.7

3 1931 6.7 23.231 28.84077 124.04 904.8 2.7

4 1932 4.9 19.081 25.68000 124.84 788.2 2.8

5 1933 4.5 19.446 23.14101 125.58 778.3 2.3

6 1934 5.5 22.545 24.39565 126.37 862.2 2.5

> tail(datasetCEP)

year sales.b price.index sales.index pop.m realgdp.b ad.gdppct

80 2008 319.5 99.130 322.3040 304.09 14,830.40 2.7

81 2009 306.5 100.000 306.5000 306.77 14,418.70 2.6

82 2010 320.6 99.347 322.7073 309.33 14,783.80 2.6

83 2011 338.9 101.089 335.2491 311.59 15,020.60 2.6

84 2012 353.7 104.744 337.6804 313.91 15,369.20 2.6

85 2013 360.7 105.732 341.1455 316.16 15,710.30 2.6

> str(datasetCEP)

'data.frame': 85 obs. of 7 variables:

$ year : int 1929 1930 1931 1932 1933 1934 1935 1936 1937 1938 ...

$ sales.b : num 9 7.8 6.7 4.9 4.5 5.5 5.8 6.3 6.6 6.5 ...

$ price.index: num 27.9 26.6 23.2 19.1 19.4 ...

$ sales.index: num 32.3 29.3 28.8 25.7 23.1 ...

$ pop.m : num 122 123 124 125 126 ...

$ realgdp.b : chr "1,056.60" "966.7" "904.8" "788.2" ...

$ ad.gdppct : num 2.8 2.7 2.7 2.8 2.3 2.5 2.3 2.3 2.3 2.2 ...

>

>

> ## Minor Clean up of CEP dataset

> sum(is.na(datasetCEP))

[1] 0

> datasetCEP <- na.omit(datasetCEP)

> sum(is.na(datasetCEP))

[1] 0

> nrow(datasetCEP) # Confirm rows after missing data removed = 0

[1] 85

> ## No missing rows in CEP dataset

>

> ## Review 'sales.b' attribute

> table(datasetCEP$sales.b)

4.5 4.9 5.5 5.8 6.3 6.5 6.6 6.7 6.9 7.2 7.8 8.5 9 10.6 12.9 14.1 15.9 17.5 18 18.6 18.9 19.3 20.5 21.2 21.3 21.4

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

22.4 23.4 23.6 23.9 25.4 25.9 26.6 27.9 28.6 31.1 32.7 35.8 37.5 41.3 44.3 45.5 49 53.5 59.2 62.4 66.9 72.2 79.3 89.3 96.4 103

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

113.2 116.7 126.4 137.6 146.8 157.2 167.7 178.2 190.4 195.2 199.1 211.2 219.1 227.4 231.2 239.5 247.5 257.8 271.1 277.9 278.8 280.8 285.3 297.5 306.5 310.7

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

319.5 320.2 320.6 323.7 338.9 353.7 360.7

1 1 1 1 1 1 1

>

> sales\_price <- datasetCEP$sales.b

>

>

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

> ## Q.4 (Part a)

> >

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

>

> ## Run functions to look at the structure of the closing price

> ## dataset for our chosen stock

> View(sales\_price)

>

> ## <Screen shot of 'View' output...here>



>

>

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

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

> >

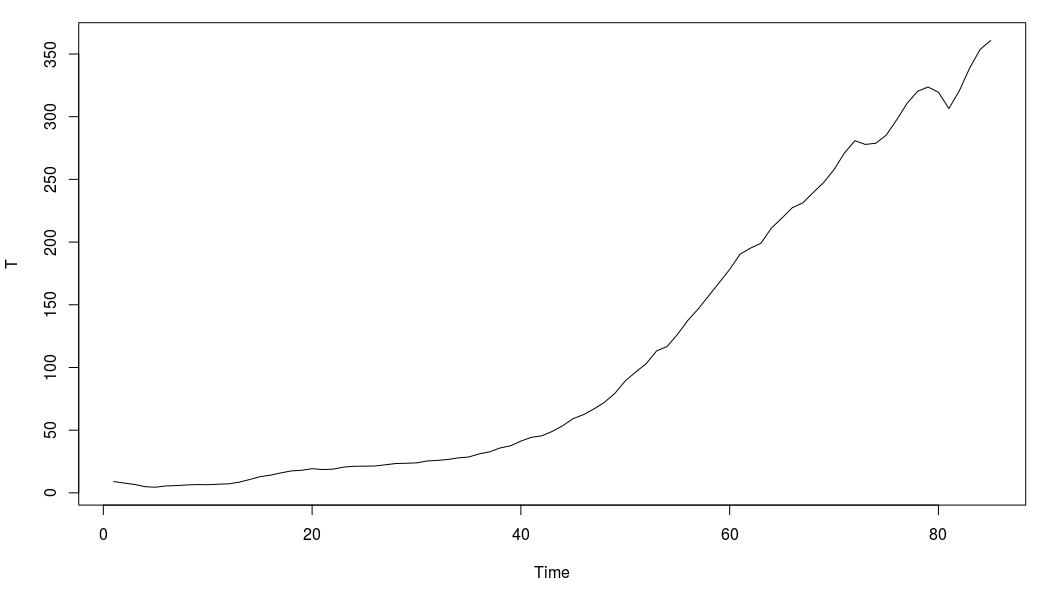
> ## Generate the plot of the time series variable- the range

> ## represents the sales prices extracted from the time range

> ## of data (frequency = 1 so every daily sales price is plotted).

> plot(T)

>



>

> ## I can see that the time series is not particularly stationary

> ## in terms of mean or variance

>

>

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

>

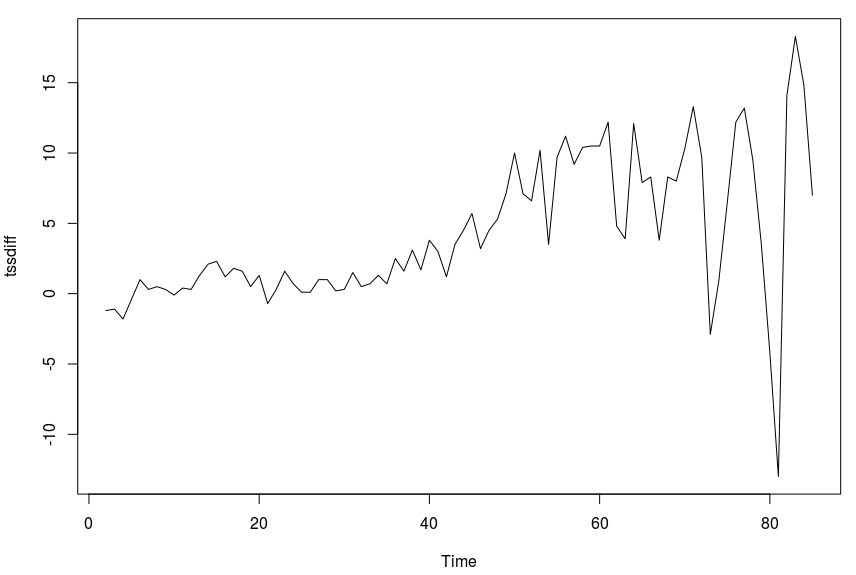
> ## Apply 'diff' function

> tssdiff=diff(T) # Stationary in mean

> plot(tssdiff)

>

> ## <Put diff plot graph here..>



>

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

> ## somewhat around zero.

>

>

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

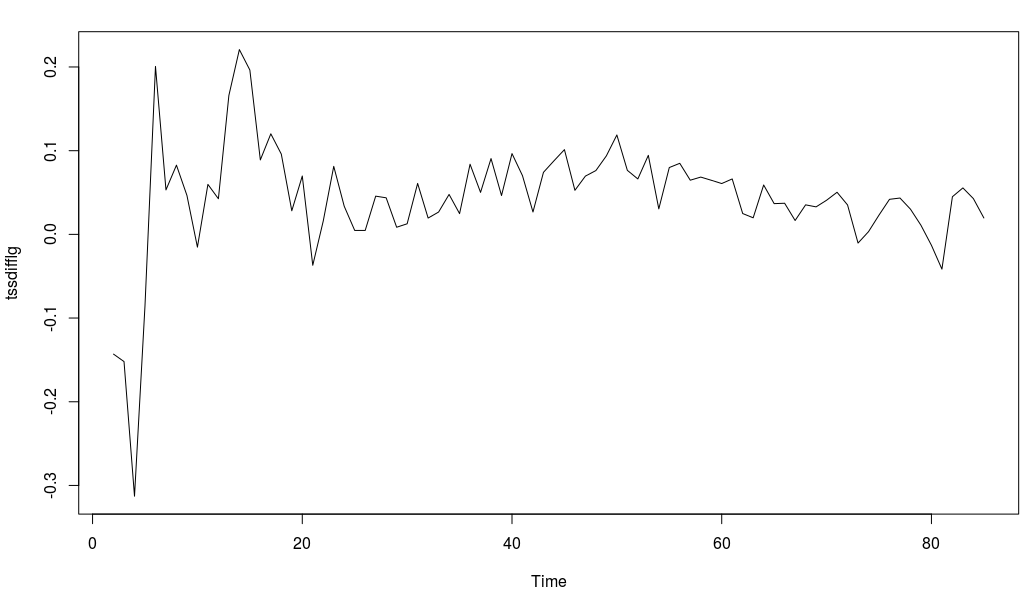
> ## stationary visualisation in mean and variance

> tssdifflg = diff(log(T))

> plot(tssdifflg)

>

> ## <Put log plot graph here..>



>

>

> ## This graph shows mean as approximately stationary and the

> ## variance also stationary between -0.1 and 0.02,

> ## apart from a few outliers

>

>

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

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

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

>

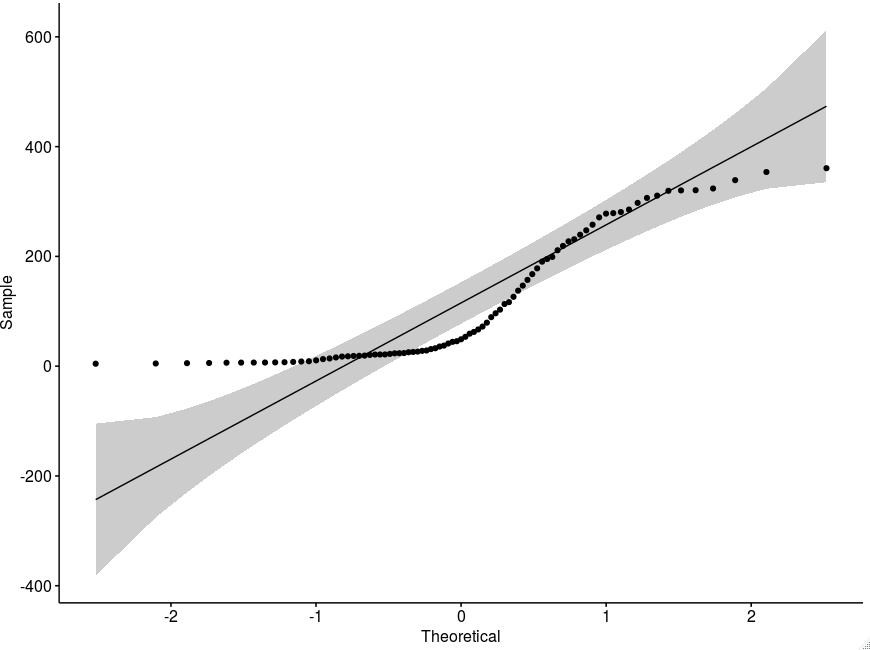
>

> ## ggqqlog plot graph

> ggqqplot(T)

>

> ## <Put ggqqlog plot graph here..>



>

>

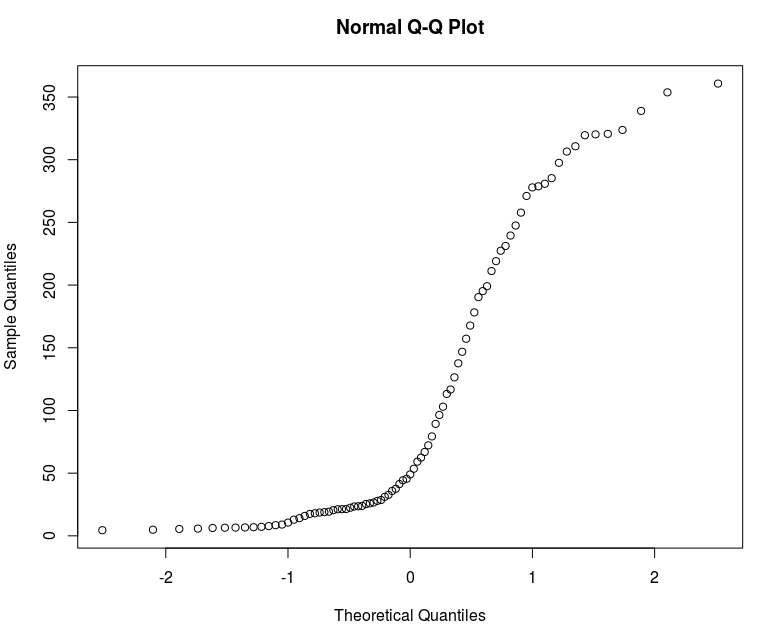
> ## qqnorm plot graph

> qqnorm(T)

>

> ## <Put qqnorm plot graph here..>

>



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

> ## Q.4 (Part b)

>

> ## Fit the optimized model for ‘sales.b’ 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

>

>

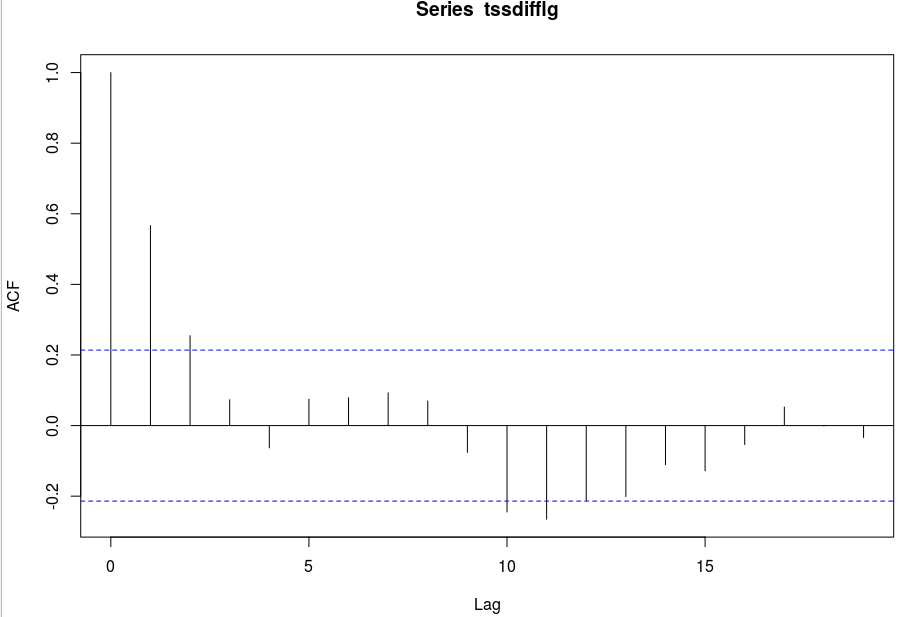
> ## 1 - Apply 'acf' and 'pacf' to get estimations of 'q' and 'p'

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

> acf(tssdifflg)

>

> ## <Put acf plot graph here..>



>

> ## There are two initial lags outside the boundary, therefore q = 3.

> ## (Above or below boundary line is not important).

>

>

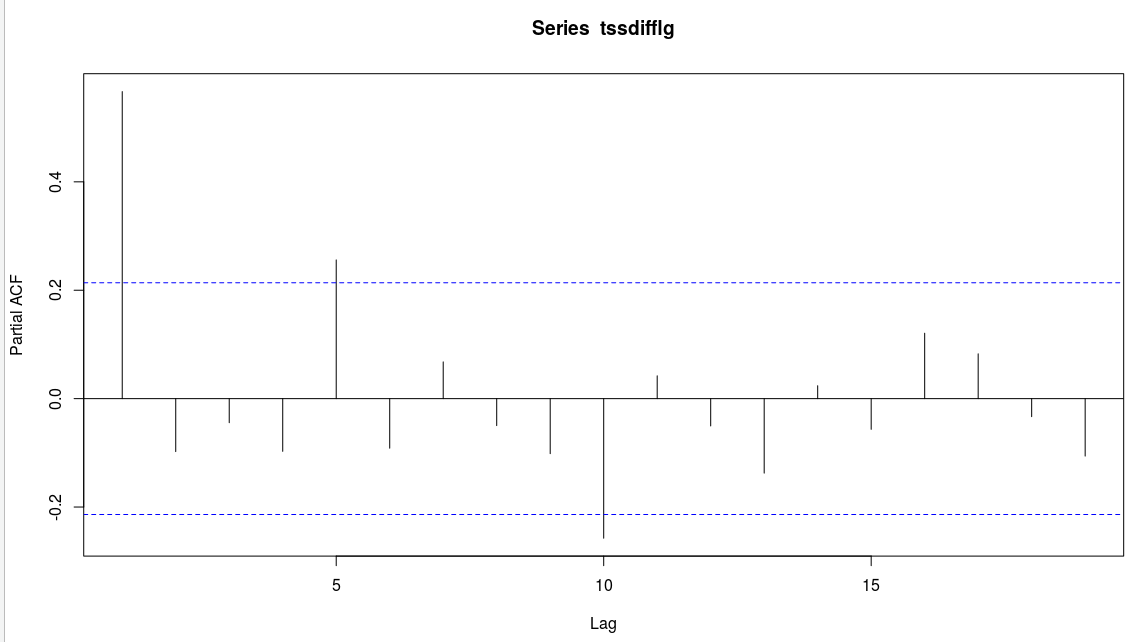
>

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

> pacf(tssdifflg)

>

> ## <Put pacf plot graph here..>



>

> ## 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 = 3.

> ## The original time series with sales.b is used.

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

> ## Parameter Estimation

> manual.fit <- arima(T, c(1,1,3), method="ML") # Fitted Model

> ## Display value of 'manual.fit'

> manual.fit

Call:

arima(x = T, order = c(1, 1, 3), method = "ML")

Coefficients:

ar1 ma1 ma2 ma3

0.9918 -0.3247 -0.3709 -0.1285

s.e. 0.0114 0.1150 0.1056 0.1215

sigma^2 estimated as 14.19: log likelihood = -231.46, aic = 472.93

>

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

> ## With q = 3, 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 ma3

> ## 0.9918 -0.3247 -0.3709 -0.1285

>

>

> ## We can see the aic (Akaike Information Criterion) value = 472.93

>

>

>

>

>

> ## 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(3,2,2)

Coefficients:

ar1 ar2 ar3 ma1 ma2

0.8457 -0.5780 -0.2651 -1.4855 0.7370

s.e. 0.1409 0.1398 0.1353 0.1086 0.0905

sigma^2 estimated as 10.27: log likelihood=-213.96

AIC=439.92 AICc=441.03 BIC=454.43

>

> ## AIC = 439.92

>

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

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

> auto.fit.T

Series: T

ARIMA(3,2,2)

Coefficients:

ar1 ar2 ar3 ma1 ma2

0.8457 -0.5780 -0.2651 -1.4855 0.7370

s.e. 0.1409 0.1398 0.1353 0.1086 0.0905

sigma^2 estimated as 10.27: log likelihood=-213.96

AIC=439.92 AICc=441.03 BIC=454.43

> # 'seasonal' flag makes no difference to result

>

> ## AIC = 439.92

>

>

> ## Automated coefficient are lower as 472.93 (Manual) > 439.92 (Auto).

> ## Therefore Auto ARIMA is better than manual fitting.

>

>

>

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

> ## Q.4 (Part c)

>

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

> auto.fit

Series: T

ARIMA(3,2,2)

Coefficients:

ar1 ar2 ar3 ma1 ma2

0.8457 -0.5780 -0.2651 -1.4855 0.7370

s.e. 0.1409 0.1398 0.1353 0.1086 0.0905

sigma^2 estimated as 10.27: log likelihood=-213.96

AIC=439.92 AICc=441.03 BIC=454.43

> ## The best model shows Series: T ARIMA(3,2,2)

> ## Therefore the estimated order for AR is p = 3, and MA is q = 2

>

>

>

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

> ## Q.4 (Part d)

>

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

> ## original time series.

>

>

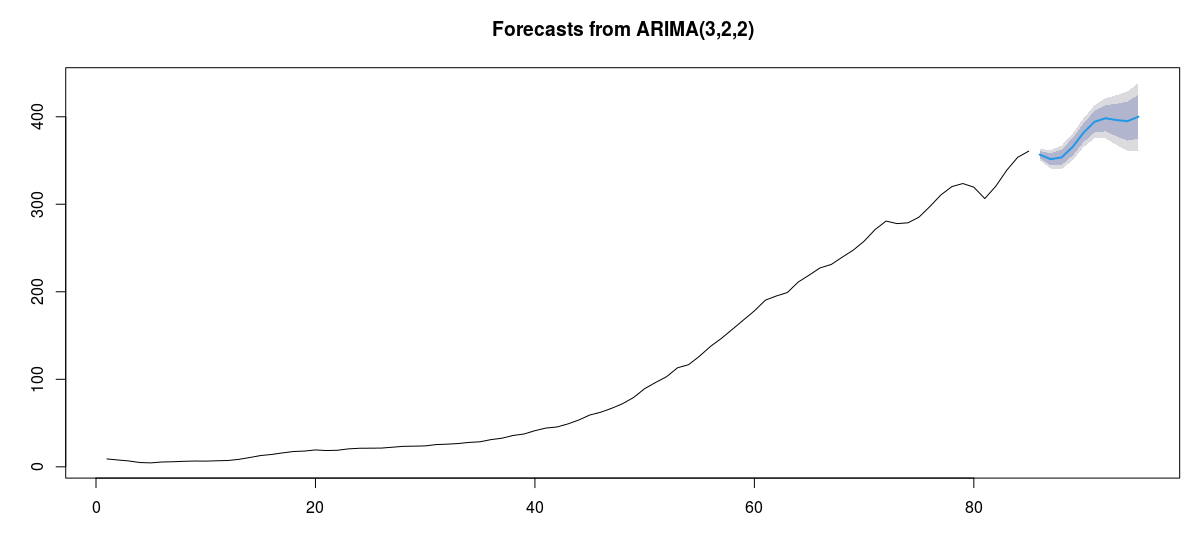
> # The best model to use is the auto fitting - as determined in the analysis in the previous

> ## steps in the question.

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

> ## Plot this forecast

> plot(auto.fcast)



|  |
| --- |
|  |
|  |
|  |
|  |
| |  | | --- | |  | |

# Question One – Probability Models

## Question 1 – from PDF

Use **mtcars** dataset and consider **disp** and **am** as the

attributes of interest.

a) Use the appropriate probability models to quantify the uncertainty

in **disp** and **am** .

**(5 Marks)**

b) Estimate the parameters of your proposed models using the dataset.

**(5 Marks)**

c) Predict the future values of **disp** and **am** using (a) and (b).

**(10 Marks)**

d) Using (a), (b), find P( **disp** > 0.7). **(5 Marks)**

**(Total: 25 Marks)**

## Output from RStudio Cloud Console

> ## Exam Advanced Data Analytics : Module Code B8IT109

|  |
| --- |
| > ## Advanced Data Analytics : Module Code B8IT109  > ## Student Name : Ciaran Finnegan 10524150  > > ## June 15th 2020  >  > ## Question 1 : Probability Models  >  > ## Use mtcars dataset and consider disp and am as  > ## the attributes of interest.  > > ## Review content and structure of mtcars dataset  >  > nrow(mtcars)  [1] 32  > head(mtcars)  mpg cyl disp hp drat wt qsec vs am gear carb  Mazda RX4 21.0 6 160 110 3.90 2.620 16.46 0 1 4 4  Mazda RX4 Wag 21.0 6 160 110 3.90 2.875 17.02 0 1 4 4  Datsun 710 22.8 4 108 93 3.85 2.320 18.61 1 1 4 1  Hornet 4 Drive 21.4 6 258 110 3.08 3.215 19.44 1 0 3 1  Hornet Sportabout 18.7 8 360 175 3.15 3.440 17.02 0 0 3 2  Valiant 18.1 6 225 105 2.76 3.460 20.22 1 0 3 1  > tail(mtcars)  mpg cyl disp hp drat wt qsec vs am gear carb  Porsche 914-2 26.0 4 120.3 91 4.43 2.140 16.7 0 1 5 2  Lotus Europa 30.4 4 95.1 113 3.77 1.513 16.9 1 1 5 2  Ford Pantera L 15.8 8 351.0 264 4.22 3.170 14.5 0 1 5 4  Ferrari Dino 19.7 6 145.0 175 3.62 2.770 15.5 0 1 5 6  Maserati Bora 15.0 8 301.0 335 3.54 3.570 14.6 0 1 5 8  Volvo 142E 21.4 4 121.0 109 4.11 2.780 18.6 1 1 4 2  > str(mtcars)  'data.frame': 32 obs. of 11 variables:  $ mpg : num 21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...  $ cyl : num 6 6 4 6 8 6 8 4 4 6 ...  $ disp: num 160 160 108 258 360 ...  $ hp : num 110 110 93 110 175 105 245 62 95 123 ...  $ drat: num 3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...  $ wt : num 2.62 2.88 2.32 3.21 3.44 ...  $ qsec: num 16.5 17 18.6 19.4 17 ...  $ vs : num 0 0 1 1 0 1 0 1 1 1 ...  $ am : num 1 1 1 0 0 0 0 0 0 0 ...  $ gear: num 4 4 4 3 3 3 3 4 4 4 ...  $ carb: num 4 4 1 1 2 1 4 2 2 4 ...  >  > attach(mtcars)  >  > #consider disp and am as the attributes of interest  > table(disp)  disp  71.1 75.7 78.7 79 95.1 108 120.1 120.3 121 140.8 145 146.7 160 167.6 225 258 275.8 301 304 318 350 351 360 400 440 460  1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 3 1 1 1 1 1 2 1 1 1  472  1  > table(am)  am  0 1  19 13  > ###################################################  > ## a) Use the appropriate probability models to quantify  > ## the uncertainty in disp and am.  >  > ## disp - This is a continous variable - hence using the  > ## Normal Model would be the standard approach  >  > ## am - This is a binary output of '0' and '1' -  > ## hence use the Bernouli Model  >  >  >  >  > ###################################################  > ## b) Estimate the parameters of your proposed models  > ## using the dataset.  >  > # disp - parameters are mu and sigma  > # model x using N(mu, sigma), mu and sigma are parameters  > # disp - mu : mean  > disp.mu = mean(disp)  > disp.mu  [1] 230.7219  >  > # mpg - sigma : standard deviation  > disp.sigma = sd(disp)  > disp.sigma  [1] 123.9387  >  >  >  > # am - parameters are sum of '1' values over row count  > y <- am  > am.p = sum(y)/length(y)  > am.p  [1] 0.40625  >  >  >  >  > ###################################################  > ## c) Predict the future values of disp and am using  > ## (a) and (b).  >  > # disp - predict using rnorm formula  > disp.preds = rnorm(1000, disp.mu, disp.sigma)  > disp.pred = mean(disp.preds)  > disp.pred  [1] 237.6375  >  >  > # vs - predict using bnorm formula  > am.preds = rbinom(1000, 1, am.p)  > table(am.preds)  am.preds  0 1  586 414  >  > ## As the count of '0' values is higher than the count  > ## of '1' values I predict that the future value of  > ## am is '0'  >  >  >  > ###################################################  > ## d) Using (a), (b), find P(disp > 0.7).  > # P(X>0.7) = 1 - P(X<=0.3) = 1 - pnorm(0.3, mu, sigma)  > # disp - use pnorm formula  > disp.prob = 1 - pnorm(0.3, disp.mu, disp.sigma)  > disp.prob  [1] 0.9684978 |
|  |
| |  | | --- | | > | |

# Question Three SVM and LR

## Question 3 – from PDF

Use the dataset ‘ **quakes’** , and consider **‘mag’** as the output variable and

select the set of input variables from the remaining columns. Split the

dataset into 80% trainset and 20% as the testset.

a) Perform linear regression (LR) analysis and derive the optimal

predictive model based on the trainset. ( **Hint** : Use α = 0.05 for

the attribute selection). Predict the values of testset using the

predictive model. **(7.5 Marks)**

b) Apply support vector regression (SVR) to predict the values of

testset. **(7.5 Marks)**

c) Use RMSE to evaluate the accuracy of two models in 1000 Monte

Carlo runs. Which method does provide a better prediction?

**(10 Marks)**

**(Total: 25 Marks)**

## Output from RStudio Cloud Console

## CA Two Advanced Data Analytics : Module Code B8IT109

## CA Two Advanced Data Analytics : Module Code B8IT109

> ## Student Name : Ciaran Finnegan 10524150

>

> ## Exam Submission

>

> ## June 15th 2020

>

> ## Question 3 : Regression analysis and support vector machine

>

> ## Use the dataset ‘ quakes’ , and consider ‘mag’ as the output variable and

> ## select the set of input variables from the remaining columns. Split the

> ## dataset into 80% trainset and 20% as the testset.

>

>

>

> library(datasets)

>

>

>

> data("quakes")

> ds <- data.frame(quakes)

>

> ## Minor Clean up of dataset

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

[1] 0

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

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

[1] 0

>

>

>

> # Standard analysis of the 'Quakes' dataset

> nrow(ds)

[1] 1000

> head(ds)

lat long depth mag stations

1 -20.42 181.62 562 4.8 41

2 -20.62 181.03 650 4.2 15

3 -26.00 184.10 42 5.4 43

4 -17.97 181.66 626 4.1 19

5 -20.42 181.96 649 4.0 11

6 -19.68 184.31 195 4.0 12

> tail(ds)

lat long depth mag stations

995 -17.70 188.10 45 4.2 10

996 -25.93 179.54 470 4.4 22

997 -12.28 167.06 248 4.7 35

998 -20.13 184.20 244 4.5 34

999 -17.40 187.80 40 4.5 14

1000 -21.59 170.56 165 6.0 119

> summary(ds)

lat long depth mag stations

Min. :-38.59 Min. :165.7 Min. : 40.0 Min. :4.00 Min. : 10.00

1st Qu.:-23.47 1st Qu.:179.6 1st Qu.: 99.0 1st Qu.:4.30 1st Qu.: 18.00

Median :-20.30 Median :181.4 Median :247.0 Median :4.60 Median : 27.00

Mean :-20.64 Mean :179.5 Mean :311.4 Mean :4.62 Mean : 33.42

3rd Qu.:-17.64 3rd Qu.:183.2 3rd Qu.:543.0 3rd Qu.:4.90 3rd Qu.: 42.00

Max. :-10.72 Max. :188.1 Max. :680.0 Max. :6.40 Max. :132.00

> str(ds)

'data.frame': 1000 obs. of 5 variables:

$ lat : num -20.4 -20.6 -26 -18 -20.4 ...

$ long : num 182 181 184 182 182 ...

$ depth : int 562 650 42 626 649 195 82 194 211 622 ...

$ mag : num 4.8 4.2 5.4 4.1 4 4 4.8 4.4 4.7 4.3 ...

$ stations: int 41 15 43 19 11 12 43 15 35 19 ...

>

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

> ## ensure consistency of results

> set.seed(1234)

> # The purpose of this is to ensure consistency in the initial prediction

>

>

> ## Split the dataset in 80/20 ratio - 'mag' is the output variable

> sample = sample.split(ds$mag, SplitRatio=0.80)

> trainset = subset(ds, sample==TRUE)

> testset = subset(ds, sample==FALSE)

>

> # Display the number of rows in each set after

> ## splitting the 'quakes' data

> nrow(ds) # Original dataset

[1] 1000

> nrow(trainset) # Training set

[1] 800

> nrow(testset) # Test set

[1] 200

> dim(trainset)

[1] 800 5

>

>

>

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

> ## Question 3 a) Perform linear regression (LR) analysis and derive the optimal

> ## predictive model based on the trainset. ( Hint : Use α = 0.05 for

> ## the attribute selection). Predict the values of testset using the

> ## predictive model.

>

>

> fit = glm(mag~., trainset, family = "gaussian")

> summary(fit)

Call:

glm(formula = mag ~ ., family = "gaussian", data = trainset)

Deviance Residuals:

Min 1Q Median 3Q Max

-0.62484 -0.12706 -0.00191 0.12563 0.79065

Coefficients:

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

(Intercept) 5.780e+00 2.100e-01 27.531 < 2e-16 \*\*\*

lat -8.111e-03 1.445e-03 -5.614 2.74e-08 \*\*\*

long -9.778e-03 1.218e-03 -8.026 3.62e-15 \*\*\*

depth -2.723e-04 3.149e-05 -8.647 < 2e-16 \*\*\*

stations 1.538e-02 3.095e-04 49.673 < 2e-16 \*\*\*

---

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

(Dispersion parameter for gaussian family taken to be 0.03641281)

Null deviance: 129.868 on 799 degrees of freedom

Residual deviance: 28.948 on 795 degrees of freedom

AIC: -372.98

Number of Fisher Scoring iterations: 2

>

>

> ## Predictive Model

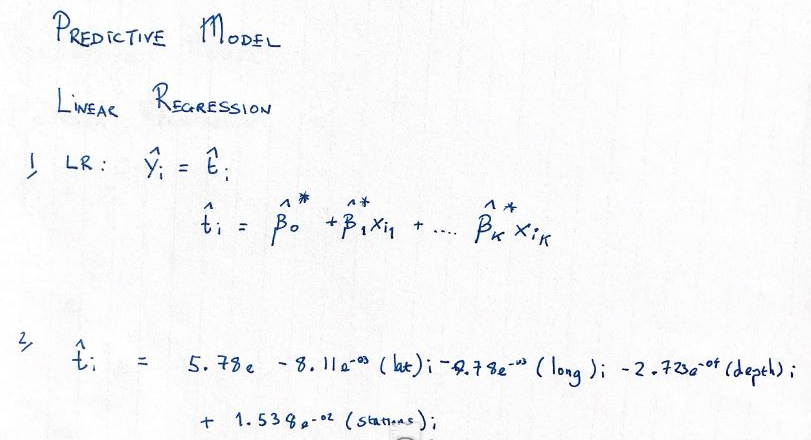
> ## yhat(i) = that(i) = 5.78e - 8.11(e-03)(lat)i - 9.71(e-03)(long)i - 2.723(e-04)(depth)i

> ## + 1.538(e-02)(stations)i

>

> ## All attributes are significant

>



> # Predict against testset

> pred=predict(fit, testset)

> pred

3 5 10 17 23 27 28 34 38 45 46 57 63 68 78 82 94

4.840773 4.159156 4.288756 5.530725 4.345045 4.496360 5.225098 4.262198 4.255930 4.627956 4.749439 4.334834 4.977855 4.723268 4.526514 4.519167 4.422314

97 113 118 119 123 124 125 126 137 143 145 168 171 175 177 185 191

4.453864 4.267045 4.402761 5.012950 4.445339 4.350567 4.455242 4.931918 5.113694 4.901086 4.386354 5.200007 4.341666 4.164913 4.818286 4.237914 5.170590

196 203 204 214 224 228 246 247 250 257 260 264 269 276 281 284 299

4.353966 4.555023 4.706021 5.412337 4.395126 4.392107 4.418791 4.380948 4.421942 4.542956 4.732223 4.413037 4.598668 4.388388 4.494269 4.164426 4.455421

307 310 315 316 322 326 342 344 346 353 355 360 367 370 373 376 381

4.302288 4.654972 4.933345 4.236269 5.166695 4.365277 4.403834 4.385202 4.361969 4.254351 4.523573 4.503731 4.764132 4.446238 4.944231 5.956691 5.264151

383 384 385 386 389 400 401 402 406 407 418 423 425 431 433 450 452

4.550155 4.958159 4.751817 5.146344 4.406388 5.479570 4.606819 4.773971 4.235745 4.383812 4.826437 4.411202 4.936883 4.196640 4.257017 4.481323 4.395191

457 469 470 473 480 485 487 490 500 501 503 511 512 532 537 541 542

4.340387 4.336426 4.827656 4.258323 4.294258 4.191170 4.870057 4.358043 4.617502 4.775149 4.621327 4.704838 5.546669 4.864516 4.508931 5.583627 4.475799

543 545 552 554 559 565 566 567 575 578 585 592 596 599 601 608 612

4.592620 4.567419 4.868941 4.696842 4.465284 4.429765 4.425368 4.452639 4.381723 4.386746 4.321900 4.357502 4.757419 4.616523 5.582013 4.706570 4.812797

613 617 620 621 625 626 629 632 636 640 644 646 647 650 660 665 670

4.640746 4.860874 4.705469 4.768625 4.837511 4.428165 5.031159 5.065938 6.021161 4.546294 4.764331 4.894176 5.007251 4.221387 4.343901 4.312651 4.536495

679 703 707 710 716 725 726 728 732 736 743 745 751 757 760 764 772

4.323845 5.037298 4.239399 4.286951 4.307921 4.459589 4.548546 4.515979 4.613303 4.733288 4.472354 5.064642 4.289749 5.219844 4.374827 5.191274 4.266150

787 799 806 819 825 827 828 835 845 848 852 859 860 867 869 872 876

5.309179 4.441666 4.407925 4.489386 4.513058 4.425616 4.453241 4.297108 4.405664 4.511684 4.875422 4.456645 4.437480 4.314752 5.436732 4.695010 4.303532

877 882 888 891 893 896 899 902 908 912 914 915 920 928 932 942 943

4.368876 4.210567 4.613851 4.615889 5.328752 4.407998 4.883598 4.721464 4.713740 4.469177 4.496107 5.209686 5.758738 5.153941 4.411645 4.588135 4.471775

945 953 955 956 963 972 975 980 986 990 992 998 999

4.573374 4.603174 4.419537 4.567863 4.522298 5.291971 4.550865 4.593570 4.415189 4.583741 4.435479 4.598807 4.289512

>

>

>

>

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

> # Q3 (b) Apply support vector regression (SVR) with the kernel ‘poly’ and

> ## predict the output variable of the testset.

> fit.svm = svm(mag~.,data = trainset, kernel='poly')

> summary(fit.svm)

Call:

svm(formula = mag ~ ., data = trainset, kernel = "poly")

Parameters:

SVM-Type: eps-regression

SVM-Kernel: polynomial

cost: 1

degree: 3

gamma: 0.25

coef.0: 0

epsilon: 0.1

Number of Support Vectors: 697

> pred.svm = predict(fit.svm, testset)

> pred.svm

3 5 10 17 23 27 28 34 38 45 46 57 63 68 78 82 94

4.845120 4.064160 4.172201 5.762958 4.474178 4.530126 4.846575 4.547960 4.245926 4.631684 4.693074 4.516332 4.926029 4.777328 4.696934 4.549054 4.501154

97 113 118 119 123 124 125 126 137 143 145 168 171 175 177 185 191

4.534148 4.415335 4.146281 4.955176 4.471892 4.360995 4.464412 4.984182 5.498400 4.888346 4.140373 4.675184 4.219638 4.386485 4.753559 4.365228 4.688768

196 203 204 214 224 228 246 247 250 257 260 264 269 276 281 284 299

4.319302 4.524200 4.562888 5.309140 4.522716 4.431443 4.370910 4.302205 4.450359 4.531137 4.674105 4.395066 4.554726 4.404043 4.493057 4.346441 4.457651

307 310 315 316 322 326 342 344 346 353 355 360 367 370 373 376 381

4.375726 4.581405 4.652190 4.393440 4.689925 4.356585 4.423985 4.397752 4.413585 4.241527 4.510910 4.519652 4.640420 4.502048 4.762238 6.453500 5.145337

383 384 385 386 389 400 401 402 406 407 418 423 425 431 433 450 452

4.537767 4.915492 4.763037 4.662026 4.594064 6.054106 4.568150 4.761548 4.195469 4.475158 4.508209 4.407330 4.875070 4.452821 4.280997 4.542963 4.478816

457 469 470 473 480 485 487 490 500 501 503 511 512 532 537 541 542

4.282474 4.325135 4.587554 4.209231 4.342941 4.152251 4.531432 3.908894 4.568272 4.952478 4.569562 4.671729 5.484681 4.869734 4.471850 5.835903 4.427535

543 545 552 554 559 565 566 567 575 578 585 592 596 599 601 608 612

4.554401 4.462595 4.959658 4.686026 4.497510 4.544727 4.516518 4.493884 4.320522 4.432267 4.347386 4.524660 4.746976 4.548806 5.574644 4.743133 4.844773

613 617 620 621 625 626 629 632 636 640 644 646 647 650 660 665 670

4.650341 4.899299 4.677586 4.676768 4.801002 4.507892 5.030608 5.030713 5.224014 4.570356 4.763758 4.678497 4.940190 4.322696 4.252550 4.231201 4.541886

679 703 707 710 716 725 726 728 732 736 743 745 751 757 760 764 772

4.389831 4.982178 4.417562 4.453889 4.013595 4.541706 4.572602 4.535238 4.545584 4.874550 4.479575 4.594650 4.215284 5.114614 4.349592 4.820015 4.442610

787 799 806 819 825 827 828 835 845 848 852 859 860 867 869 872 876

5.377181 4.445513 4.486669 4.514883 4.473102 4.583676 4.448654 4.515105 4.528937 4.550792 4.842441 4.475610 4.617881 4.535330 5.344337 4.673791 4.321726

877 882 888 891 893 896 899 902 908 912 914 915 920 928 932 942 943

4.436137 4.325569 4.585087 4.722222 5.424685 4.499719 4.689244 4.801903 4.702802 4.470381 4.496223 5.171680 6.013318 5.081553 4.486410 4.526524 4.485094

945 953 955 956 963 972 975 980 986 990 992 998 999

4.574113 4.582998 4.498078 4.572118 4.543539 5.270940 4.557615 4.582198 4.630136 4.581336 4.500543 4.558252 4.169799

>

>

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

> ## Q3 (c) Use RMSE measure to evaluate the accuracy of two models in 100

> ## Monte Carlo runs.

> ## Which method does provide a better prediction?

>

> a=0; mc=1000

>

> for(i in 1:mc){

+

+ sample.mc = sample.split(ds$mag, SplitRatio=0.80)

+ trainset.mc = subset(ds, sample==TRUE)

+ testset.mc = subset(ds, sample==FALSE)

+

+

+

+ # SVM

+ fit.svm.mc = svm(mag~.,data = trainset.mc, kernel='poly')

+ yhat.svm = predict(fit.svm.mc, testset.mc)

+

+ # RMSE - SVM

+ mse.svm = sum((yhat.svm-testset.mc$mag)^2)/nrow(testset.mc)

+ rmse.svm = sqrt(mse.svm)

+

+

+ # Linear Regression

+ fit.lm.mc = glm(mag~., trainset.mc, family = "gaussian")

+ yhat.lm = predict(fit.lm.mc, testset.mc)

+

+ # RMSE - Linear Regression

+ mse.lm = sum((yhat.lm-testset.mc$mag)^2)/nrow(testset.mc)

+ rmse.lm = sqrt(mse.lm)

+

+

+

+

+

+ # Evaluate the above code in 100 MC runs

+ a=a+rmse.svm/mc

+

+ a=a+(1/mc)\*c(rmse.svm,rmse.lm)

+

+ }

>

> ## Display RMSE values for the SVM and Linear Regression (in that order)

> a

[1] 0.5196329 0.4602642

>

> ## The lower the RMSE value the better. Therefore the Linear Regression

> ## gives a better prediction (0.4602642 < 0.5196329)

# Question Four

## Question 4 – from PDF

## Output from RStudio Cloud Console

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