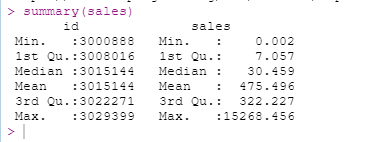
**STUDENT NAME:**

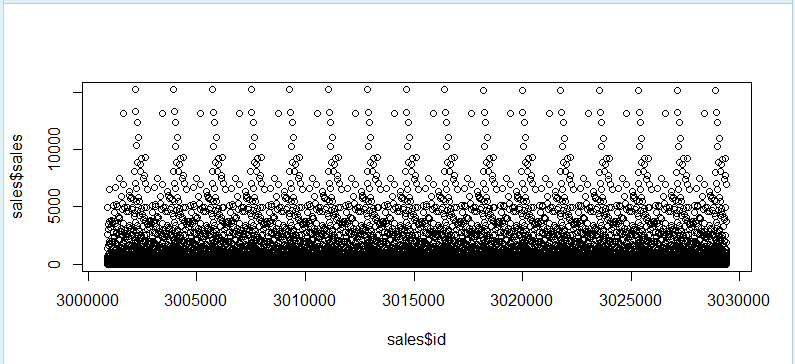
**DATE:**

1. **Linear Regression Model**

Summary of the Sales data



***Simple linear regression***

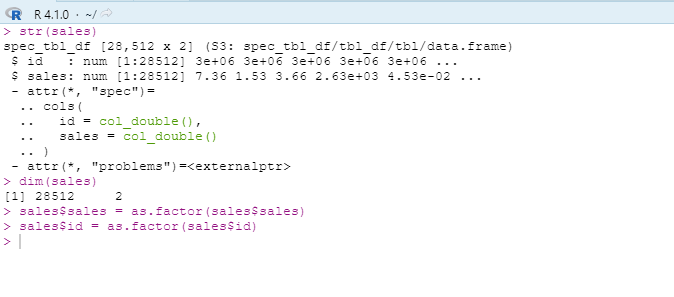


***Linear correlation***



1. **Random forest**

Load the dataset and make id as dependent variable

****

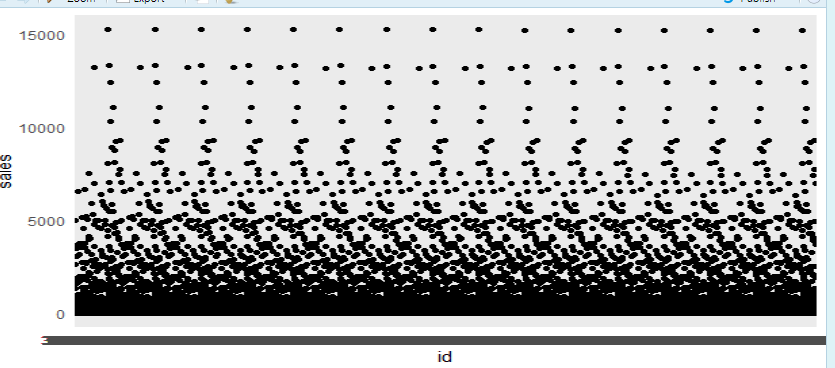
****

1. **CORRELATIONAL MODEL**
2. Getting the correlation between the ***store\_nbr*** and the***cluster*** they belong to:
3. > stores <- read\_csv("class/hello/papers/Data analysis with R/stores.csv")
4. Rows: 54 Columns: 5
5. -- Column specification -----------------------------------------------------------------------
6. Delimiter: ","
7. chr (3): id, sale
8. i Use `spec()` to retrieve the full column specification for this data.
9. i Specify the column types or set `show\_col\_types = FALSE` to quiet this message.
10. > View(stores)
11. > cor(sales$id, stores$sales)
12. [1] -0.05928399

Plotting the Correlation graphs:

> library(ggplot2)

> ggplot(sales) + aes(x = store\_nbr, y = cluster) + geom\_point(colur = "#0c4c8a")+ theme\_minimal()



**Interpretation of the model**

In this lab, we have conducted a correlational analysis between two variables provided on the dataset. The correlation result produced is -0.05928399 meaning that there is no correlation whatsoever between the two values and that the values are independent of each other, the existent of one and a change in one does not affect the other. In this case, if store number changes, it does not affect in any way the relationship of the cluster of the rela*t*ionship

1. **TIME SERIES MODEL WITH ARIMA**

> View(sales)

> plot.ts(sales)

# GET THE TIMES SERIES

> library(ggplot2) ggplot(sales, aes(id, sales)) + geom\_line()

> library(ggplot2)

> ggplot(sales, aes(id, sales)) + geom\_line()

#PREDICT THE NEXT CLUSTER OF SHOPS OVERTIME

plot(diff(log(sales$sales)),type='l', main='log returns plot')

#STEP 4 STATIONARISE THE TIME SERIES

>adf.test(diff(log(as.numeric(sales$sales))), alternative="stationary", k=0)

#STEP 5 CALCULATE THE ACF PACF ON OUR DATASET BASED ON SHOP CLUSTERS

>acf(diff(log(sales$sales)))

>pacf(diff(log(sales$sales)))

#STEP 6

#CALL OUR FORECAST LIBRARY INSIDE ARIMA MODEL

> library(forecast)

> (fit <- arima(diff(log(sale$sales)), c(3, 0, 1)))

#FITTING OUR ARIMA

> fitARIMA <- auto.arima(diff(log(sales$sales)), trace=TRUE)

#CHECKNG TO SEE HOW OUR ARIMA MODEL FITTED WITH THE TRAINING DATASET

> plot(as.ts(diff(log(sales$sales))) )

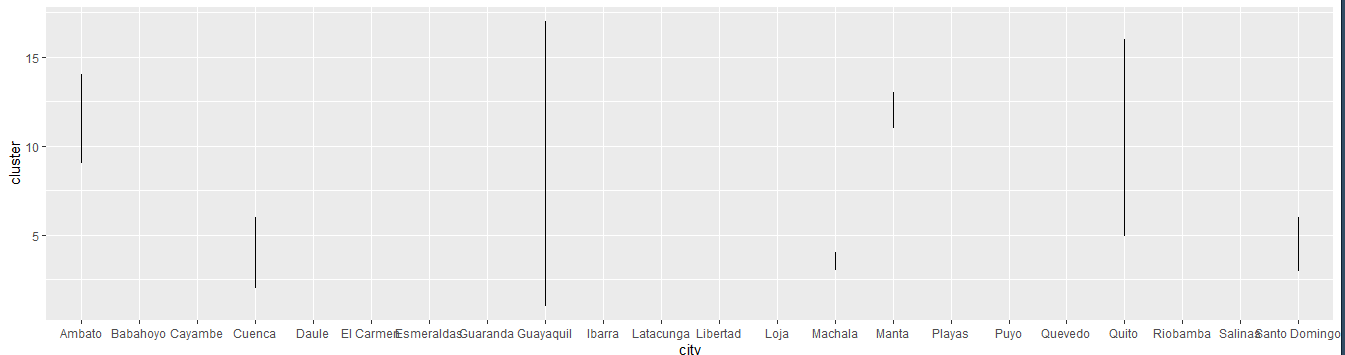
> lines(fitted(fitARIMA), col="red")

#STEP 7

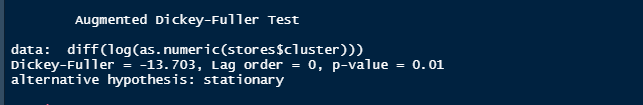
#MAKING A PREDICTION BASED ON OUR ARIMA MODEL

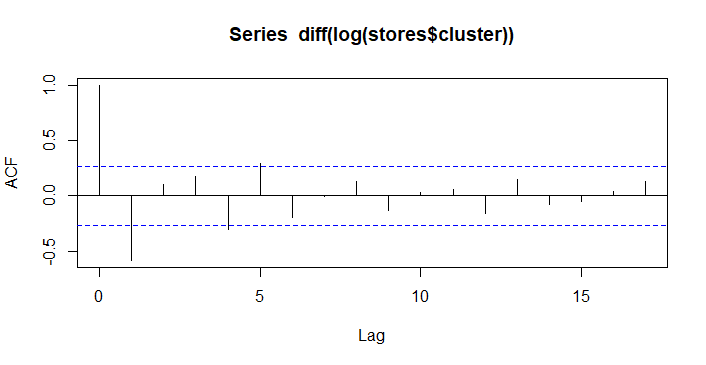
 >futurVal <- forecast(fitARIMA,h=5, level=c(99))

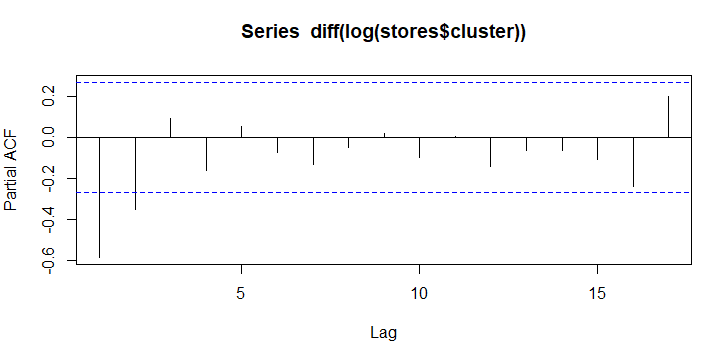
> plot(forecast(futurVal))

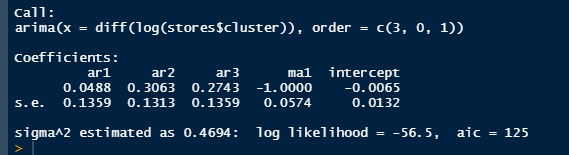


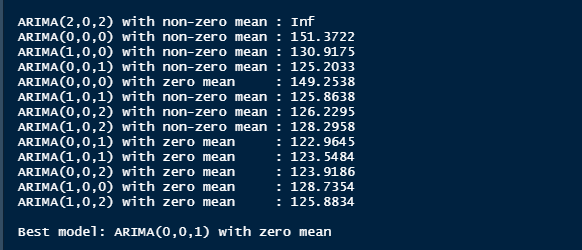


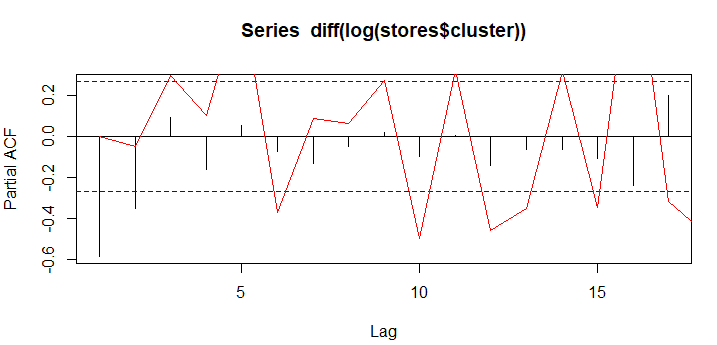


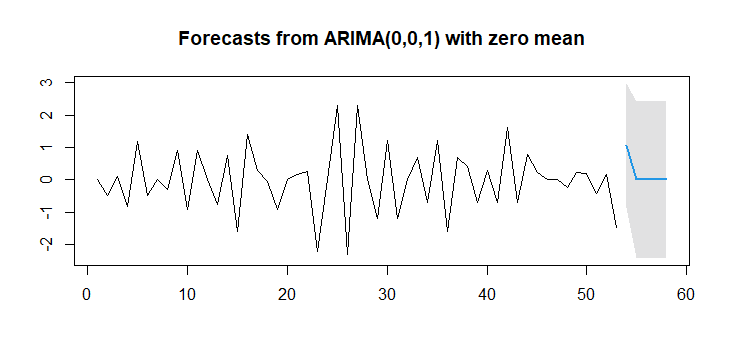


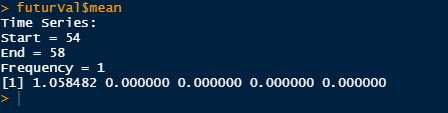












1. **RNN MODEL**

## STEP 3

 ##WE USE MOVING BLOCK SUB SAMPLING TO ENABLE US CUT OUR VECTOR INTO SMALL BITS FOR EASIER SAMPLING

> start\_indexes <- seq(1, length(sales\_type)- (max\_len + 1), by = 3)

> cluster\_matrix <- matrix(nrow = length(start\_indexes), ncol = max\_len + 1)

> for (i in 1:length(start\_indexes)){cluster\_matrix [i,] + max\_len)]}

> for (i in 1:length(start\_indexes))

{cluster\_matrix [i,] <-store\_type[start\_indexes[i]:(start\_indexes[i] + max\_len)]}

#REMOVE WARNING MESSAGES

> dev.off()

null device

## STEP 4

#WE REMOVE N/A values and converting our matrix to NUMERIC

> for (i in 1:length(start\_indexes))

{cluster\_matrix [i,] <-store\_type[start\_indexes[i]:(start\_indexes[i] + max\_len)]}

> cluster\_matrix <- cluster\_matriX \* 1

> cluster\_matrix <- suppressWarnings(as.numeric(cluster\_matrix) \* 1)

> if(anyNA(cluster\_matrix)){

+     cluster\_matrix <- na.omit(cluster\_matrix)

+ }

## STEP 5

#SEPARATE OUR DATA INTO PREVIOUS DAYS AND THEN DEFINE DAYS WE WANT TO PREDICT FOR THE STORE TYPE IN Y VARIABLE

X <- cluster\_matrix[,-ncol(cluster\_matrix)]

y <- cluster\_matrix[,ncol(cluster\_matrix)]

## STEP 6

# THIS INDEXING WILL SEPARATE OUR DATA INTO TRAINING AND TESTING UNITS

training\_index <- createDataPartition(y, p = .9,

                                  list = FALSE,

                                  times = 1)

## STEP 7

# THEN TRAIN THE DATA

X\_train <- array(X[training\_index,], dim = c(length(training\_index), max\_len, 1))

y\_train <- y[training\_index]

## STEP 7

# THEN TEST THE DATA

X\_test <- array(X[-training\_index,], dim = c(length(y) - length(training\_index), max\_len, 1))

y\_test <- y[-training\_index]

## STEP 8

# DEFINE A NEW MODEL FOR THE STORES DATASET

stores\_cluster\_model <- keras\_model\_sequential()

# DEFINE NEW DIMENSIONS FOR INPUT DATA

dim(X\_train)

## STEP 9

# THEN DEFINE INPUT LAYER OF THE MODEL

stores\_cluster\_model %>%

    layer\_dense(input\_shape = dim(X\_train)[2:3], units = max\_len)

stores\_cluster\_model %>%

    layer\_simple\_rnn(units = 6)

stores\_cluster\_model %>%

    layer\_dense(units = 1, activation = 'sigmoid')

# TO GET A SUMMARY OF THE MODEL STRUCTURE USE summary command

summary(stores\_cluster\_model)

## STEP 10

# TRAINING THE MODEL

stores\_cluster\_training\_model <- stores\_cluster\_model %>% fit(

    x = X\_train,

    y = y\_train,

    batch\_size = batch\_size,

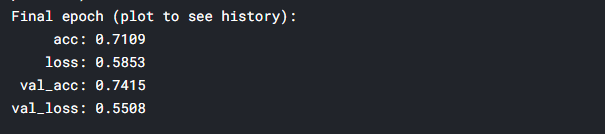
    epochs = total\_epochs,

    validation\_split = 0.1)

## STEP 11

# PREVIEW THE MODEL

stores\_cluster\_training\_model



# PLOT THE RESULTING MODEL AS TRAINED

plot(stores\_cluster\_training\_model)

