Time-Series Forecasting and Model Evaluation

Prachi Sardana

2023-09-29

## Task Set 1 Question 1

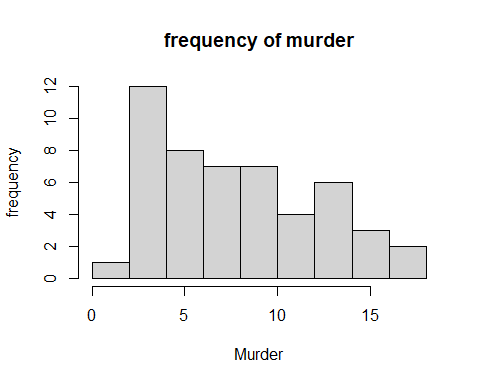
To load in the built in data set [USArrests] obtained from (<https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/USArrests.html>) which constitute the statistics about the violent crime rate in US

To calculate the outliers which are defined as values that are more than 1.5 standard deviations from the mean.

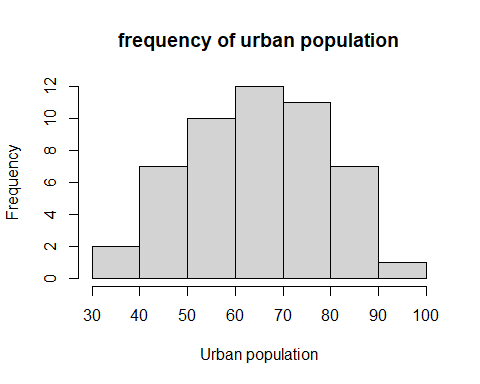
## Task set 1 Question 2

To calculate the correlation, there is a need to check for normal distribution of data which can be checked by plotting a histogram, by testing normality through Shapiro wilk test and plotting Q-Q Plot where the data points in the straight line indicate that the data is normally distributed. Based on checking the normality we select correlation algorithms.

# Histogram to check if the data is normally distributed  
hist(df$Murder, main = "frequency of murder",xlab = "Murder", ylab = "frequency")



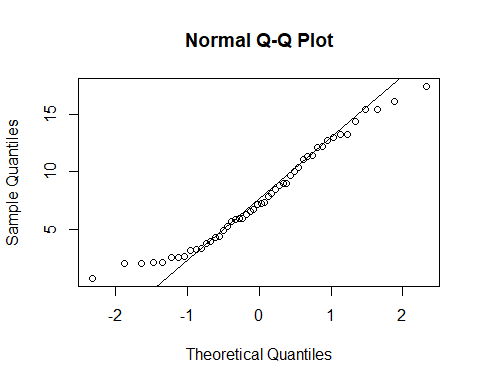
hist(df$UrbanPop, main = "frequency of urban population", xlab = "Urban population", ylab = "Frequency")



# Based on the shapiro wilk test the p value is greated than 0.05 which indicates that the data is normally distributed.   
shapiro.test(df$Murder)

##   
## Shapiro-Wilk normality test  
##   
## data: df$Murder  
## W = 0.95703, p-value = 0.06674

# Create a Q-Q plot  
qqnorm(df$Murder)  
qqline(df$Murder)



# Based on the Q-Q plot most of the data points lie in the straight line indicating that data is normally distributed.  
  
# Hence, we can use parametric correlation coefficient Pearson to check the relationship between the 2 variables  
  
correlation <- cor(df$UrbanPop, df$Murder, method = "pearson")  
print(correlation)

## [1] 0.06957262

# Since the correlation coefficient is 0.06 , it means that both variables are weak and positively related to each other.

Since the correlation is close to 0.06 which means that there is no correlation between the urban population and murder. Since a correlation of 1 indicates a positive correlation while close to 0 indicates less correlation and - 1 indicates negative correlation.Pearson correlation algorithm is appropriate since the data is normally distributed based on shapiro wilk test and Q-Q Plot and hence,parametric.The Spearman and Kendall correlation coefficients are non parametric and used when the data is not normally distributed. Choosing a wrong correlation coefficient can impact the statistical significance of the data and detect the outliers in it .

## Task Set 2 Question 1

Used google sheet package to load the data on growth of mobile phone use in Brazil (“<https://docs.google.com/spreadsheets/d/1tOnM9XceK4Ak8tzWQ2vDelWlJexzJiS3LbT6MN6_rW0/edit?usp=sharing>”) and calculate the weighted average mean.Used gsheet2tbl()function to copy the data from google sheets using *gsheet* package

## Warning: package 'gsheet' was built under R version 4.2.3

## # A tibble: 12 × 2  
## Year Subscribers  
## <dbl> <dbl>  
## 1 1 23188171  
## 2 2 28745769  
## 3 3 34880964  
## 4 4 46373266  
## 5 5 65605000  
## 6 6 86210336  
## 7 7 99918621  
## 8 8 120980103  
## 9 9 150641403  
## 10 10 173959368  
## 11 11 202944033  
## 12 12 NA

## [1] 194662700

Using function wma forcast for a time series dataframe taking 3 arguments data frame, num\_coloumns(Year), weight. For each period it multiplies the data by weight and sum the results in p returning weighted average mean

num\_coloumn <- nrow(df)  
# used function weighted average mean and the formula   
wmaForecast <- function (df, num\_coloumn, weight) {  
   
 # creating a sequence of indices for periods  
 periods <- length(df):(length(df) - num\_coloumn + 1)  
   
 # multiply the periods by the weights and sum  
 p <- sum(df[periods] \* weight)  
   
 # divide the sum by the sum of the weights  
 f <- p / sum(weight)  
   
 # return forecast  
 return (f)  
}

Calculated the forecast for next time period next.year.wma using a 2 year weighted average mean based on the function of wma and the weights for 5 most recent year , 2 for the other and printed the forecast of next 2 years weighted mean average.

# calculated the forecast for next time period using a 2 year weighted average mean based on the function of wma and the weights for 5 most recent year , 2 for the other   
next.year.wma <- wmaForecast(df$Subscribers, 2, c(5,2))  
print(next.year.wma)

## [1] 194662700

Calculated Exponential smoothing which is a forecasting methd to calculate the weighted average of past observation and determine the future points

# Exponential smoothing   
  
# Forecast of time period +1 = Forecast of prev time period + alpha \* Error which is equal to (Yt - Ft ) Yt is actual - Forecasted  
  
  
# Smoothing parameter = 0.4  
alpha = 0.4   
  
# Forecast for first time period called as observed value   
df$f[1] <- df$Subscribers[1]

## Warning: Unknown or uninitialised column: `f`.

# since for the first coloumn error remains 0   
df$error[1] <- 0

## Warning: Unknown or uninitialised column: `error`.

# calculate the forecast for all period > 1 starting from 2nd coloumn  
for (j in 2:(nrow(df))) {  
 # calculate the forecast as forecast(t-1) + alpha\*error(t-1)   
 df$f[j] <- df$f[j-1] + (alpha \* df$error[j-1])  
   
 # Error(t) = Observed value(t) - Forecast(t)  
 df$error[j] <- df$Subscribers[j] - df$f[j]  
 }  
  
# Exponential smoothing for next year =   
exp\_s.next\_year <- df$f[j] + (alpha \* df$error[j])  
print(exp\_s.next\_year)

## [1] 165168214

# exponential smoothing forecast function that take arguments(dataframe and alpha smoothing constant)  
esForecast <- function (df, alpha) {  
 # created a dataframe data.es to have columns for forecast (f) and error (e)  
 data.es <- data.frame(t = 1:length(df),  
 x = df,  
 f = 0,  
 error = 0)  
   
 # the "forecast" for the first time period is the observed value (by convention)  
 data.es$f[1] <- data.es$x[1]  
 # the error is (by definition) then 0  
 data.es$error[1] <- 0  
   
 # calculate the forecast for all period > 1  
 for (i in 2:(length(df))) {  
 # calculate the forecast as forecast(i-1) + alpha\*error(i-1)  
 data.es$f[i] <- data.es$f[i-1] + (alpha \* data.es$error[i-1])  
   
 # calculate the error for the forecast (Y - f)  
 data.es$error[i] <- data.es$x[i] - data.es$f[i]  
 }  
   
 return (data.es$f[i] + (alpha \* data.es$error[i]))  
}  
  
next.year.es <- esForecast(df$Subscribers, 0.4)  
print(next.year.es)

## [1] 165168214

Trendline linear progression using linear model lm() function.

# Linear regression trendline   
# model <- lm(y ~ x, data = your\_data)  
  
linear\_model <- lm (Subscribers ~ Year, data = df)  
summary(linear\_model)

##   
## Call:  
## lm(formula = Subscribers ~ Year, data = df)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -12307858 -9795553 -4238521 7402838 20622182   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -15710760 8041972 -1.954 0.0825 .   
## Year 18276748 1185724 15.414 8.9e-08 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 12440000 on 9 degrees of freedom  
## Multiple R-squared: 0.9635, Adjusted R-squared: 0.9594   
## F-statistic: 237.6 on 1 and 9 DF, p-value: 8.903e-08

The linear equation is the form of **y = mx+b** and calculate the forecast by using *x = 12*, the “next year”

# Based on the formula y= mx + b where m is the slope and x is the time period and b is the intercept  
  
#forecast\_linearmodel <- Intercept + 12 \* Slope  
#print(forecast\_linearmodel)  
# Based on the formula y = mx + b where m is the slope and x is the time period, and b is the intercept  
  
forecast\_linear\_model <- linear\_model$coefficients[[2]] \* 12 + linear\_model$coefficients[[1]]  
print(forecast\_linear\_model)

## [1] 203610220

Applied the functions linear regression trend line

# function linear regression trendline which takes argument as datafame and   
  
forecast\_tl <- function(df, t){  
 period = 1:length(df)  
 lin\_model <- data.frame(x = period,  
 y = df)  
 m = lm(y ~ x, data = lin\_model)  
 forecast\_linear\_model <- m$coefficients[[2]] \* (t) + m$coefficients[[1]]  
 return(forecast\_linear\_model)  
}  
  
next.year.tl <- forecast\_tl(df$Subscribers, 12)  
print(next.year.tl)

## [1] 203610220

## Task set 2 Question 2 ,3 and 4

*MSE* calculates the square of the difference between the actual and the predicted value Mean squared error for trendline is less compared to exponential smoothing and weighted moving average

# Initialize a vector to store MSE values  
mse\_values <- numeric(length = nrow(df))  
  
# Calculate MSE for E/S (Exponential Smoothing)  
for (i in 3:nrow(df)) {  
 f.ES <- esForecast(df$Subscribers[1:(i-1)], alpha = 0.4)  
 mse\_values[i] <- (df$Subscribers[i] - f.ES)^2  
}  
  
# Calculate the average mean of the MSE values for Exponential smoothing  
MSE.ES <- mean(mse\_values)  
print(paste0("MSE for Exponential smoothing: ", MSE.ES))

## [1] "MSE for Exponential smoothing: 1471030393938056"

# Calculate MSE for (Trendline /Linear)  
for (i in 1:nrow(df)) {  
 f.TL <- forecast\_tl(df$Subscribers, i)  
 mse\_values[i] <- (df$Subscribers[i] - f.TL)^2  
}  
  
# Calculate the average mean of the MSE values for Trendline   
MSE.TL <- mean(mse\_values)  
print(paste0("MSE for Trendline: ", MSE.TL))

## [1] "MSE for Trendline: 126534746000244"

# Initialize a vector to store MSE values for MA (Moving Average)  
mse\_values\_MA <- numeric(length = nrow(df))  
  
# Calculate mean MSE for MA (Moving Average)  
for (i in 3:nrow(df)) {  
 f.MA <- wmaForecast(df$Subscribers[1:(i-1)], 2, c(5, 2))  
 mse\_values\_MA[i] <- (df$Subscribers[i] - f.MA)^2  
}  
  
# Calculate the average mean of the MSE values for Moving average  
MSE.MA <- mean(mse\_values\_MA)  
print(paste0("MSE for moving average: ", MSE.MA))

## [1] "MSE for moving average: 544143882735677"

# Task Set2 Question 5

Used ensembleForecast() function to calculate the weighted average of the three forecasts moving average, trendline and exponential smoothing with the weights of 4 for trend line, 2 for exponential smoothing and 1 for weighted moving average.Dividing the sum of weights in a weighted average

ensembleForecast <- function (df, t, alpha, n, weights.WMA, weights) {  
 f.WMA <- wmaForecast(df, n, weights.WMA)  
 f.ES <- esForecast(df, alpha)  
 f.TL <- forecast\_tl(df, t)  
 # Calculate the weighted average forecast  
 weighted\_forecast <- (weights[1]\*f.TL + weights[2]\*f.ES + weights[3]\*f.WMA) / sum(weights)  
 }

# Assuming df is your dataframe  
next.year.ensemble <- ensembleForecast(df$Subscribers, 12, 0.4, 2, c(5,2), c(4,2,1))  
print(next.year.ensemble)

## [1] 191348573