R Notebook

Problem 1

1.The built-in dataset USArrests contains statistics about violent crime rates in the US States. Determine which states are outliers in terms of assaults. Outliers, for the sake of this question, are defined as values that are more than 1.5 standard deviations from the mean. Only identify the outliers, do not remove them.

```
library(tidyverse)
```

```
## -- Attaching packages ------------------------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.1.0     v purrr   0.2.5
## v tibble  1.4.2     v dplyr   0.7.8
## v tidyr   0.8.1     v stringr 1.3.1
## v readr   1.1.1     v forcats 0.3.0
```

```
## -- Conflicts ---------------------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ggplot2)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
library(dplyr)

data("USArrests")
mean_assaults <- mean(USArrests$Assault)
std_dev <- sd(USArrests$Assault)
USArrests$z_score <- abs((mean_assaults - USArrests$Assault)/std_dev)
outliers <- USArrests[which(USArrests$z_score > 1.5), ]
outliers
```

```
##                Murder Assault UrbanPop Rape  z_score
## Florida          15.4     335       80 31.9 1.970778
## Maryland         11.3     300       67 27.8 1.550799
## North Carolina   13.0     337       45 16.1 1.994776
## North Dakota      0.8      45       44  7.3 1.509042
```

```
#output: Florida, Maryland, North Carolina, North Dakota
```

2.For the same dataset as in (1), is there a correlation between murder and assault, i.e., as one goes up, does the other statistic as well? Comment on the strength of the correlation. Calculate the Pearson coefficient of correlation in R using a function – you do not have to do the calculation yourself.

```
cor(x = USArrests$Murder, y = USArrests$Assault, method = "pearson")
```

```
## [1] 0.8018733
```

```
#output : 0.8018733
```

Problem 2

1.Download the data on the growth of mobile phone use in Brazil (you'll need to copy the data and create a CSV that you can load into R) and load it into an R data frame.

```
mobile_sub <- data.frame(read_csv("~/Downloads/Raw Data Mobile Phone Growth (Brazil) - Mobile Phone Subs
```

```
## Parsed with column specification:
## cols(
##   Year = col_integer(),
##   Subscribers = col_integer()
## )
```

2.Forecast phone subscriptions for the next time period using a simple moving average of the prior four time periods.

```
#moving average
n <- nrow(mobile_sub)
sub1 <- mobile_sub[8:11, 2]
mean(sub1)
```

```
## [1] 162131227
```

```
#output :Forecast phone subscriptions for the next time period is 162131227
```

3.Forecast phone subscriptions for the next time period using a 3-year weighted moving average (with weights of 7 for the most recent year, 4 for the one prior and 1 for the one prior to that).

```
weight_1 <- c(7,4,1)
last_3 <- mobile_sub[9:11, 2]
new_1 <- weight_1 * last_3
weighted_mvavg <- sum(new_1)/ sum(weight_1)
#output : Forecast phone subscriptions for the next time period using a 3-year weighted moving average
```

4.Forecast phone subscriptions for the next time period using exponential smoothing (alpha of 0.2)

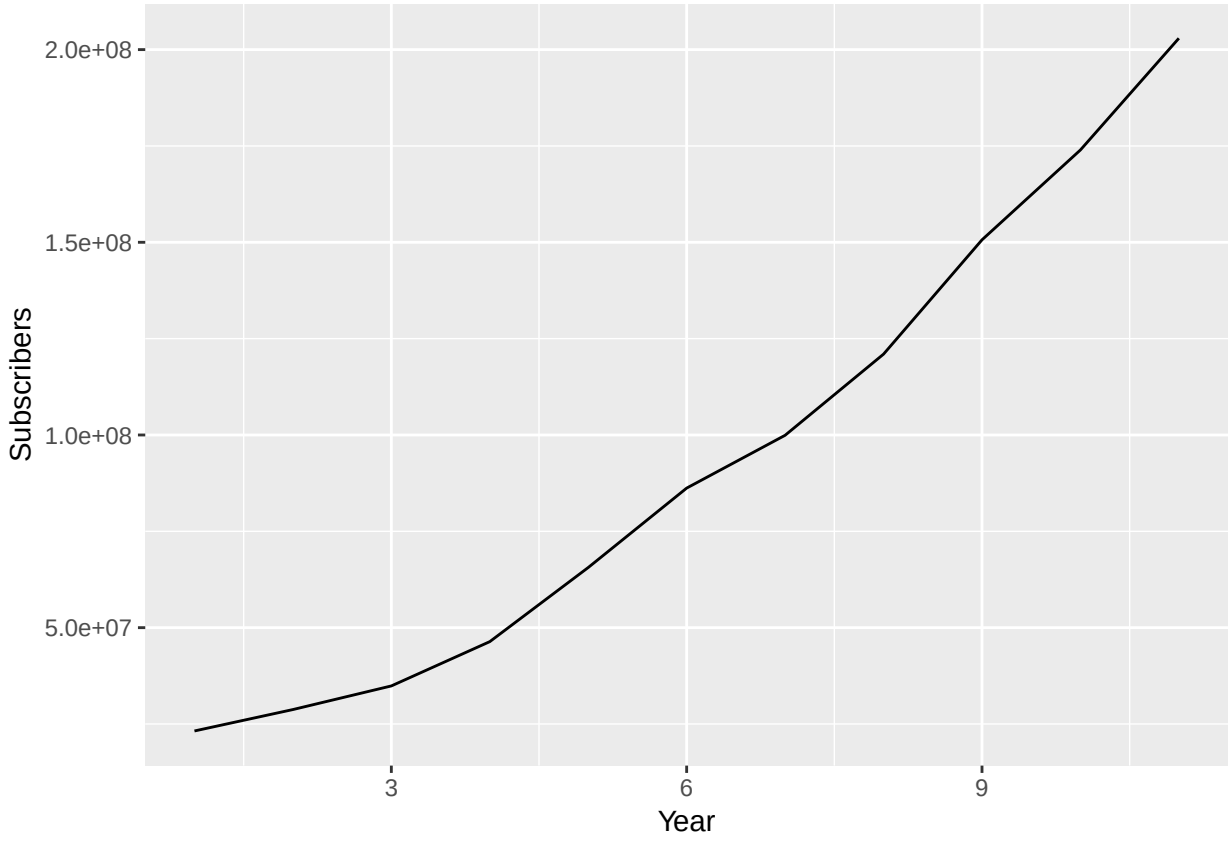
```
alpha <- 0.2
mobile_sub$Ft <- 0
mobile_sub$Error_val  <- 0

#F(t) <- F(t-1) + a * E(t-1)
mobile_sub$Ft[1] <- mobile_sub$Subscribers[1]
mobile_sub$Ft[2] <- mobile_sub$Ft[1] + alpha * mobile_sub$Error_val[1]
mobile_sub$Error_val[2] <- mobile_sub[2,2] - mobile_sub$Ft[2]

for (i in 3 : nrow(mobile_sub))
{
  mobile_sub$Ft[i] <- mobile_sub$Ft[i-1] + alpha * mobile_sub$Error_val[[i-1]]
  mobile_sub$Error_val[i] <- mobile_sub[i,2] - mobile_sub$Ft[i]
}
exp_forecast <- mobile_sub$Ft[n] + alpha * mobile_sub$Error_val[[n]]
#output: Forecast phone subscriptions for the next time period is 123744469
```

5. Forecast phone subscriptions for the next time period using a linear regression trend line.

```
ggplot(mobile_sub, aes(Year, Subscribers)) + geom_line()
```




```
model <- lm(mobile_sub$Subscribers ~ mobile_sub$Year)
print(model)
```

```
##
## Call:
## lm(formula = mobile_sub$Subscribers ~ mobile_sub$Year)
##
## Coefficients:
##     (Intercept)  mobile_sub$Year
##       -15710760         18276748
```

```
forecast <- -15710760 + 18276748*(12)
#output : Forecast phone subscriptions for the next time period is 203610216
```

6.Calculate the average mean squared error for models from (3), (4), and (5), i.e., use the model to calculate a forecast for each given time period and then calculate the squared error for each observation; then average the squared errors. Which model has the smallest mean squared error (MSE)?

```
mobile_sub$Ft <- 0
mobile_sub$Error_val  <- 0
moving_average <- rbind(mobile_sub, c(n+1, weighted_mvavg))
exp_smooth <- rbind(mobile_sub, c(n+1, exp_forecast))
linear_trend <- rbind(mobile_sub, c(n+1, forecast))

weighted_forecast <- lm(moving_average$Subscribers ~ moving_average$Year)
moving_average$F_1 <- 0
```

```
moving_average$error <- 0

for(i in 1: (n+1))
{
  moving_average$F_1[i] <- 17711891 * moving_average$Year[i] - 13263044
  moving_average$error[i] <- abs(moving_average[i,2] - moving_average$F_1[i]) ^ 2
}

mean(moving_average$error, na.rm = TRUE)
```

```
## [1] 2.30791e+14
```

```
forecast_exp <- lm(exp_smooth$Subscribers ~ exp_smooth$Year)
exp_smooth$F_1 <- 0
exp_smooth$error <- 0

for (i in 1:(n-1))
{
  exp_smooth$F_1[i] <- 15204989 * exp_smooth$Year[i] -2399801
  exp_smooth$error[i] <- abs(exp_smooth[i,2] - exp_smooth$F_1[i]) ^ 2
}

mean(exp_smooth$error, na.rm = TRUE)
```

```
## [1] 1.056129e+14
```

```
forecast_lr <- lm(linear_trend$Subscribers ~ linear_trend$Year)
linear_trend$F_1 <- 0
linear_trend$error <- 0

for (i in 1 : (n-1))
{
  linear_trend$F_1[i] <- 18276748 * linear_trend$Year[i] - 15710759
  linear_trend$error[i] <- abs(linear_trend[i,2] - linear_trend$F_1[i]) ^ 2
}

mean(linear_trend$error, na.rm = TRUE)
```

```
## [1] 9.014585e+13
```

8.Calculate a weighted average forecast by averaging out the three forecasts calculated in (3) with the following weights: 5 for trend line, 2 for exponential smoothing, 1 for weighted moving average. Remember to divide by the sum of the weights in a weighted average.

```
weight_2 <- c(5,2,1)
forecast_avg <- c(forecast,exp_forecast, weighted_mvavg)
new_2 <- weight_2 * forecast_avg
average_forecast <- sum(new_2)/sum(weight_2)
```