Time Series Forcasting and Model Evaluation

Dr. Nishat Mohammad

01/24/2024

Track Set 1

Question 1

The built-in dataset USArrests contains statistics about violent crime rates in the US States. Determine which states are outliers in terms of murders. Outliers, for the sake of this question, are defined as values that are more than 1.5 standard deviations from the mean.

```
library(datasets)
usarr_data<- USArrests
# Look at the data
dimensions_usarr <- dim(usarr_data)
str(usarr_data)
```

Answers:

```
## 'data.frame':    50 obs. of  4 variables:
##  $ Murder  : num  13.2 10 8.1 8.8 9 7.9 3.3 5.9 15.4 17.4 ...
##  $ Assault : int  236 263 294 190 276 204 110 238 335 211 ...
##  $ UrbanPop: int  58 48 80 50 91 78 77 72 80 60 ...
##  $ Rape    : num  21.2 44.5 31 19.5 40.6 38.7 11.1 15.8 31.9 25.8 ...
```

```
head(usarr_data)
```

```
##            Murder Assault UrbanPop Rape
## Alabama      13.2     236       58 21.2
## Alaska       10.0     263       48 44.5
## Arizona       8.1     294       80 31.0
## Arkansas      8.8     190       50 19.5
## California    9.0     276       91 40.6
## Colorado      7.9     204       78 38.7
```

```
n_na <- any(is.na(usarr_data))

# Get the outlier states with sd of 1.5 away from the mean for MURDER
# Get basic stats
murd_mn <- mean(usarr_data$Murder)
```

```r
murd_sd <- sd(usarr_data$Murder)

# Get Z score
z <- abs((usarr_data$Murder - murd_mn) / murd_sd)
# Outliers  for sd of 1.5 away from mean
outliers <- which(z > 1.5)
outlier_states <- rownames(usarr_data[outliers,])
```

The data shows the details of crime across states in the US. The first 6 rows and the columns can be seen in the table above. The dimensions are: "50", "4".
Are there any missing values? "FALSE".
The outlier states are : "Florida", "Georgia", "Louisiana", "Mississippi", "North Dakota", "South Carolina".

Question 2

For the same dataset, is there a correlation between urban population and murder, i.e., as one goes up, does the other statistic as well? Comment on the strength of the correlation. Which correlation algorithm is appropriate? Pearson? Spearman, Kendall? How would you decide between them? What if you choose an incorrect algorithm; what would the effect be?

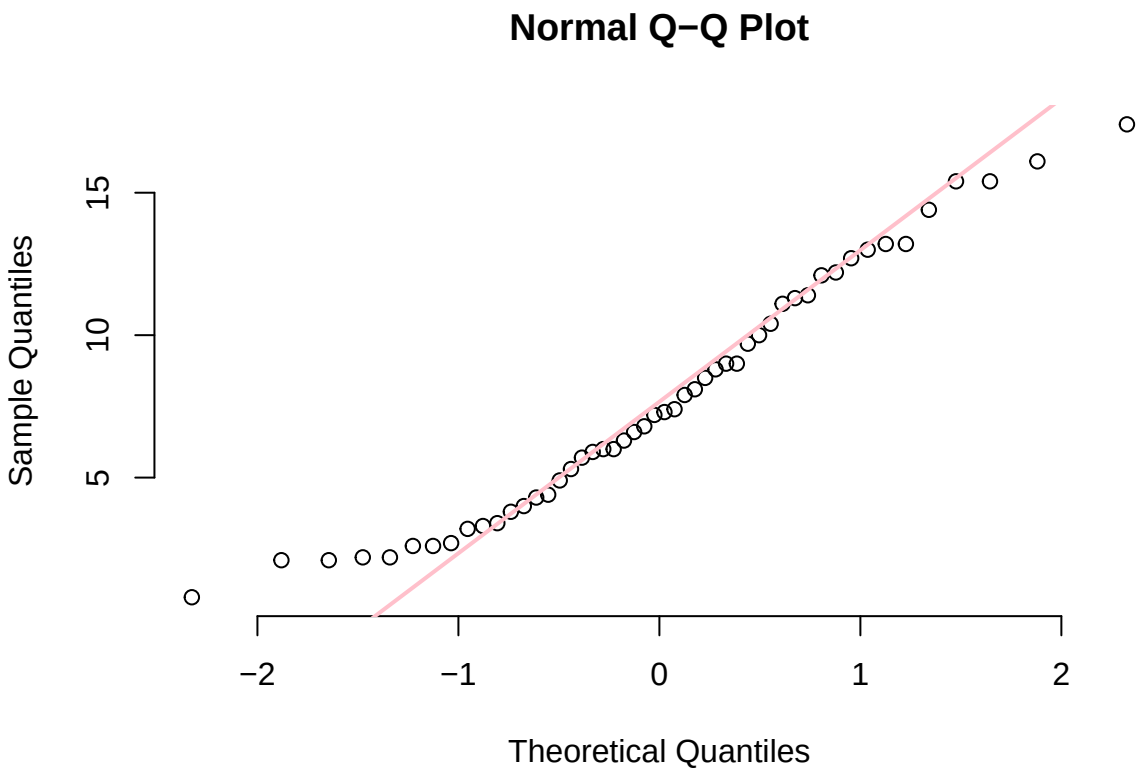
```r
# Check if the data in murder variable is normally distributed
murd_shap_wilk<-shapiro.test(usarr_data$Murder)
murd_shap_score<- murd_shap_wilk$statistic
murd_shap_p <- murd_shap_wilk$p.value
murd_shap_score
```

Answers:

```
##        W
## 0.957027
```

A Shapiro-Wilk test will check for normality of distribution of the Murder data, the Shapiro_Wilk score is "0.957026981656104", close to 1, with a p-value of "0.0667428225446598". This shows that the data is significantly normally distributed with a small p-value.

```r
# Visualize with a scatter plot to see how far outliers are form the general trend
qqnorm(usarr_data$Murder, pch = 1, frame = FALSE)
qqline(usarr_data$Murder, col = "pink", lwd = 2)
```



The q-q plot shows an S-shaped plot, which is in accordance with normally distributed data, so we could use the Pearsons Algorithm (great for normal data), but considering Pearsons correlation is sensitive to outliers, we could compare what we get with Spearman correlation as well which is less senstive to outliers when compared with Pearsons correlation.

```
# Pearsons correlation
murd_pcc <- cor(usarr_data$Murder, usarr_data$UrbanPop, method = "pearson")

murd_scc <- cor(usarr_data$Murder, usarr_data$UrbanPop, method = "spearman")
```

The Pearsons Correlation coefficient is “0.0695726217359934” which is remote from 1 and thus low, implies that the Urban population and Murder do not have a strong correlation.
I tried to verify this with Spearman test considering Pearsons is more sensitive to outliers and got a Spearman coefficient of “0.106716341834532”, which is also low and thus a verification that there is weak correlation between Urban Population and Murder.

Using the correlation algos in the wrong scenario:
The properties of each algo have to be known, the distribution of the data also has to be known, the presence of outliers have to be considered before choosing the algo for that scenario.
If we use the wrong algo for the scenario we can generally have incorrect interpretation of linearity,eg. if Kendall is used on normally distributed data can not only lead to incorrect interpretation but also loss of power because Kendall focuses on the monotonicity of the data. in cases where the data variables have tied values Kendall can over-emphasize this and again there will be incorrect interpretaion resulting from this.
For data with outliers Spearman is less sensitive to outliers than Pearson which can be strongly influenced by the outliers to give a biased correlation coefficient. On the other hand Pearson is better with linear data than Spearman which if used can lead to loss of power due to the inability of Spearman to work for linear normally distributed data.

The choice of the algo will be dependent on both the charcateristics of the data and the capabilities of the algo in relation to those characteristics.

Task Set II.

Question 1.

Based on 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 or use the gsheet2tbl() function from the gsheet package), forecast phone use for the next time period using a 2-year weighted moving average (with weights of 5 for the most recent year, and 2 for other), exponential smoothing (alpha of 0.4), and linear regression trendline.

```
mob_growth_data <- as.data.frame(read.csv("RawDataMobilePhoneGrowthBrazilMobilePhoneSubscriptions.csv",
head(mob_growth_data)
```

Answers:

```
##   Year Subscribers
## 1    1    23188171
## 2    2    28745769
## 3    3    34880964
## 4    4    46373266
## 5    5    65605000
## 6    6    86210336
```

```
str(mob_growth_data)
```

```
## 'data.frame':    12 obs. of  2 variables:
##  $ Year       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Subscribers: int  23188171 28745769 34880964 46373266 65605000 86210336 99918621 120980103 150641
```

```
# Remove the empty 12th row
mob_growth_data <- mob_growth_data[1:11,]
#mob_growth_data
```

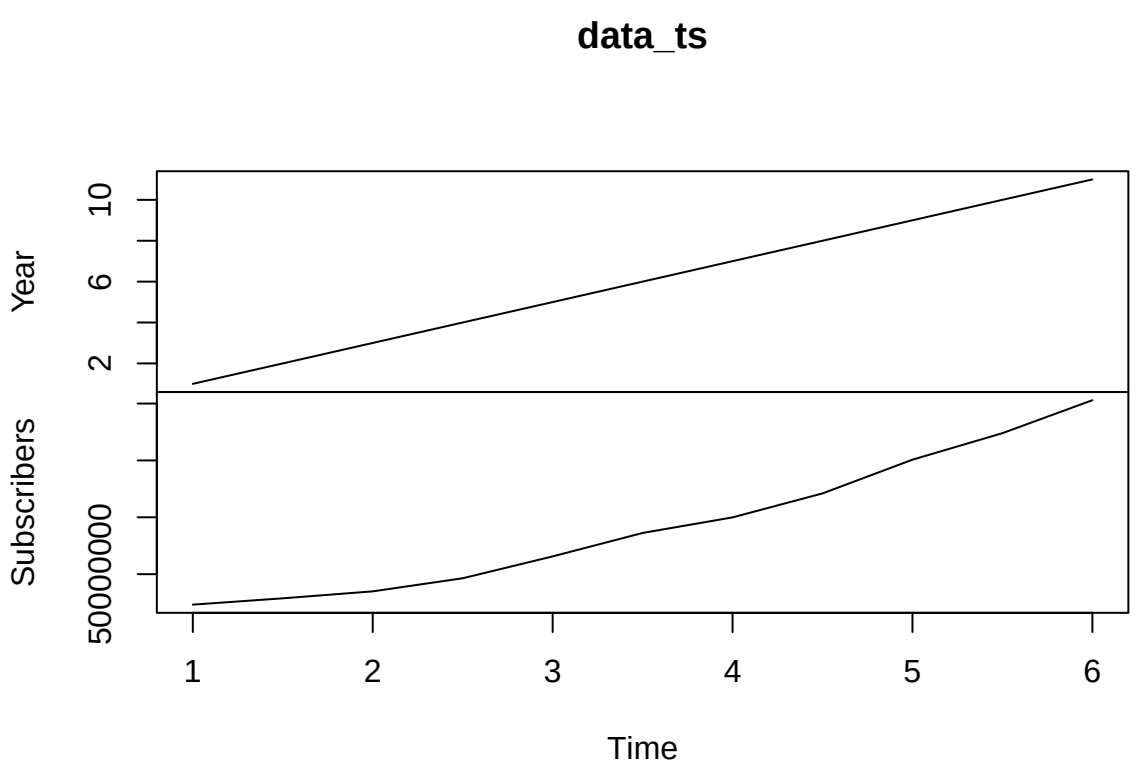
Fixing the data a bit

```
library(forecast)
```

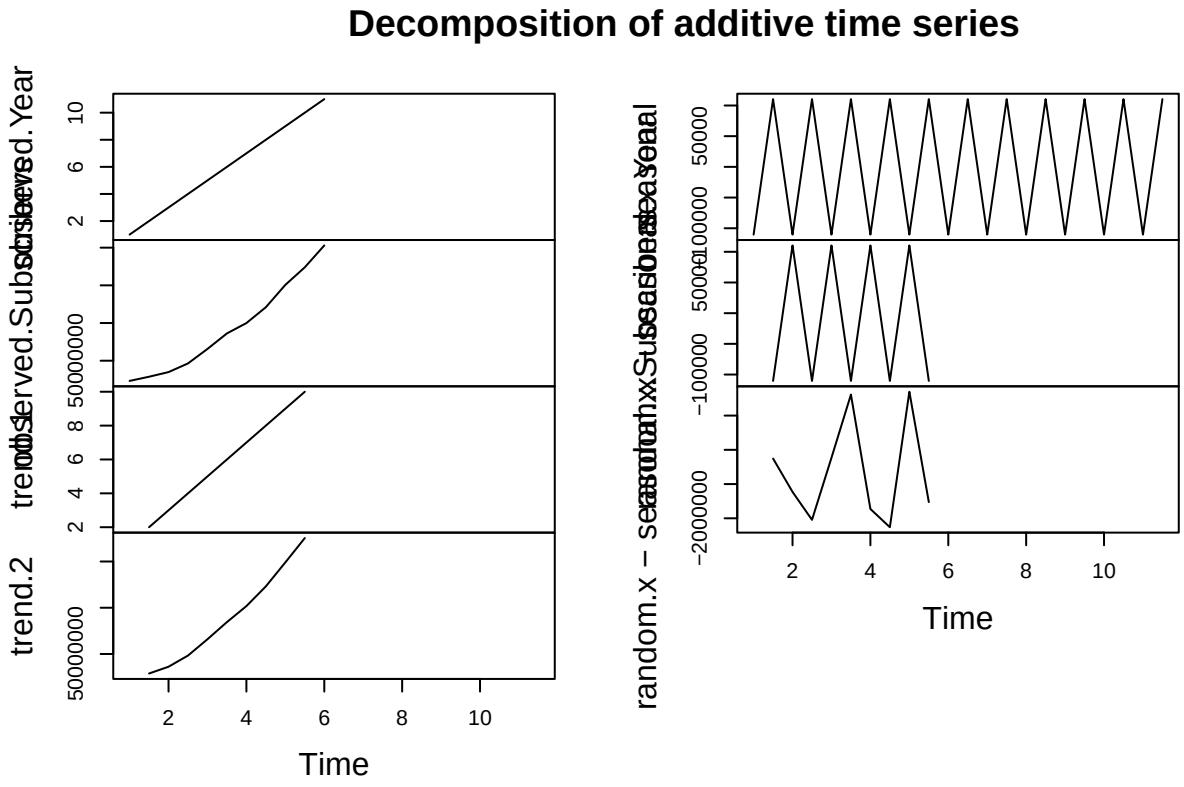
Test forcast without WMA or SES

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```r
options(scipen = 999)
# Time series
data_ts <- ts(mob_growth_data, frequency = 2)
plot(data_ts)
```




```r
# decompose the data
forcast_data <- decompose(data_ts)
plot(forcast_data)
```



Just testing to look at the trend, seasonality observed without weighted moving average, exponential smoothing or linear regression. Now we can get the weighted moving average.

```
# Qualify the weights
# Weight for most recent yr is 5
wt_factor1 <- 5
# Weight for second most recent yr is 2
wt_factor2 <- 2

# the total number of yrs in the data
tot_obs_yr <- nrow(mob_growth_data)
#tot_obs_yr
# Get the weighted average
nxt_yr_forcast <- (wt_factor1 * mob_growth_data$Subscribers[tot_obs_yr-1] +
            wt_factor2 * mob_growth_data$Subscribers[tot_obs_yr-2])/
  sum(wt_factor1,wt_factor2)
```

Weighted Moving Average The Weighted Moving average is designed to give more weight to the recent most occurrences than the previous ones, here we used weight factor of 5 for the previous year and weight factor of 2 for the 2nd most recent year to get the average moving average we divided the sum of the weighted values (subscribers multiplied by weight factor) by the sum of the weight factors.
The forecast for the next year for the number of subscribers is “167297092.285714”.

```
#create a new data frame
nw_mobgr_df <- as.data.frame(n = 1: nrow(mob_growth_data), x = mob_growth_data, forcast = 0, error = 0)

# first values of the forcast and error columns
nw_mobgr_df$forcast <- c(mob_growth_data$Subscribers[1], rep(0, nrow(mob_growth_data)-1))
nw_mobgr_df$error = c(rep(0,11))
nw_mobgr_df
```

Simple Exponential Smoothing

```
##    Year Subscribers  forcast error
## 1     1    23188171 23188171     0
## 2     2    28745769        0     0
## 3     3    34880964        0     0
## 4     4    46373266        0     0
## 5     5    65605000        0     0
## 6     6    86210336        0     0
## 7     7    99918621        0     0
## 8     8   120980103        0     0
## 9     9   150641403        0     0
## 10   10   173959368        0     0
## 11   11   202944033        0     0
```

```
# Define some values
alpha <- 0.4
r <- nrow(nw_mobgr_df)

# Loop to find the forcast using error and alpha
for (x in 2:r){
  nw_mobgr_df$forcast[x] <- nw_mobgr_df$forcast[x-1] + (alpha*nw_mobgr_df$error[x-1])
  nw_mobgr_df$error[x] <- nw_mobgr_df$Subscribers[x] - nw_mobgr_df$forcast[x]
}
nw_mobgr_df
```

```
##    Year Subscribers   forcast    error
## 1     1    23188171  23188171        0
## 2     2    28745769  23188171  5557598
## 3     3    34880964  25411210  9469754
## 4     4    46373266  29199112 17174154
## 5     5    65605000  36068773 29536227
## 6     6    86210336  47883264 38327072
## 7     7    99918621  63214093 36704528
## 8     8   120980103  77895904 43084199
## 9     9   150641403  95129584 55511819
## 10   10   173959368 117334311 56625057
## 11   11   202944033 139984334 62959699
```

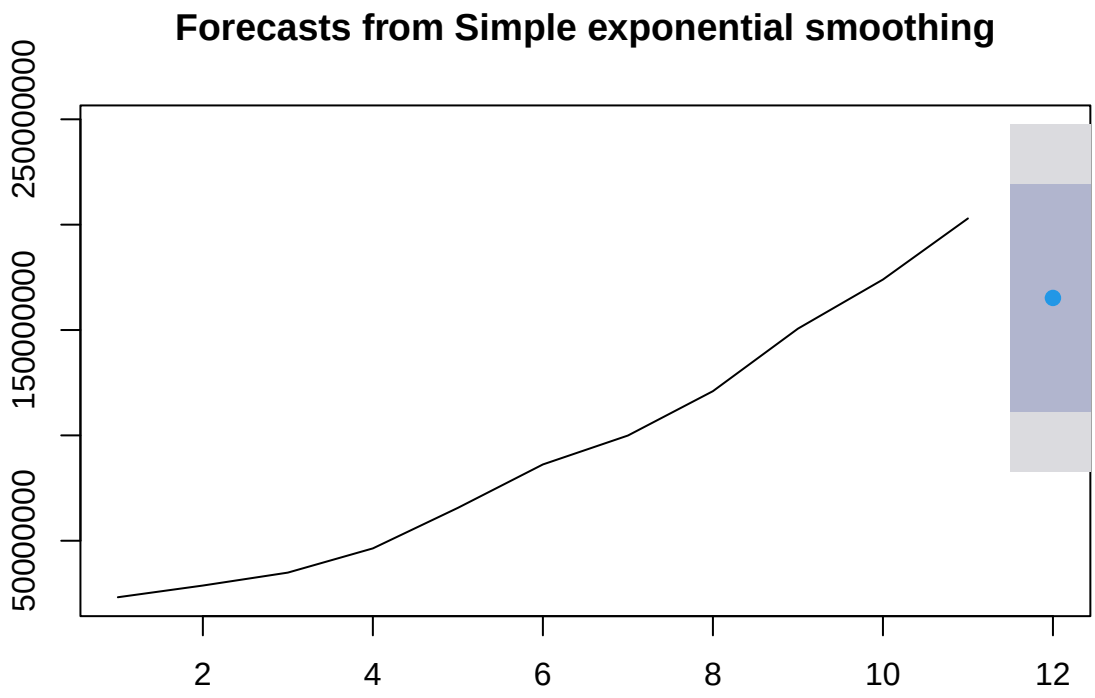
```
# Forecast the 12th year
yr = 12
forcast12 <- nw_mobgr_df$forcast[yr-1] + (alpha*nw_mobgr_df$error[yr-1])
forcast12
```

```
## [1] 165168214
```

```
# Or
# Using the ses() function
forcast_ses <- ses(nw_mobgr_df$Subscribers, alpha = 0.4, h = 1, trace=TRUE)
forcast_ses$mean
```

```
## Time Series:
## Start = 12
## End = 12
## Frequency = 1
## [1] 165219439
```

```
plot(forcast_ses)
```



Forcasting with exponential smoothing requires the addition to the current forcast value the multiple of the error and a constant. in this case the constant alpha is 0.4. the error is th differnce between the actual number of subscribers and the forcasted number of subscribers.
The exponential forecast with alpha of 0.4 for the 12th year is "165168213.62273".

In the second method, I used the ses() function which forecasts the 12th year to be the same value and the plot is seen above.

```r
# get the linear model
lrm <- lm(data = nw_mobgr_df,Subscribers~Year )
summary(lrm)
```

Linear Regression Trendline

```
##
## Call:
## lm(formula = Subscribers ~ Year, data = nw_mobgr_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 0.000000089 ***
## ---
## 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: 0.00000008903
```

```r
# Forcast the next yr, 12th year
f12 <- lrm$coefficients[[2]] * 12 + lrm$coefficients[[1]]
f12
```

```
## [1] 203610220
```

The Linear regression Trendline forcasts that the 12th year will have "203610220.436364" subscribers. The summary of the linear regression can be seen above.

Question 2.

Calculate the squared error for each model, i.e., use the model to calculate a forecast for each provided time period in the data set and then the square the error.

```r
# FOR WMA
# Make a function for WMA forecast
forcst_wma <- function (data, n, wt) {
  ind <- length(data):(length(data) - n + 1)
  time_per <- sum(data[ind] * wt)
  # Get forecast
  wma_forcst <- time_per / sum(wt)
  return (wma_forcst)
```

```
}
# Use function to make forecast and get the Square Error
nw_mobgr_df$WMA_SE <- 0

for(i in 3:nrow(nw_mobgr_df)) {
  frcst_wma <- forcst_wma(nw_mobgr_df$Subscribers[1:(i-1)], 2, c(5,2))
  #print(frcst_wma)
  nw_mobgr_df$WMA_SE[i] <- (nw_mobgr_df$Subscribers[i] - frcst_wma)^2
  #print(frcst_wma)
}
WMA_se <- nw_mobgr_df$WMA_SE
WMA_se
```

Answers:

```
##  [1]                0                0   59645966892994  175435716611878
##  [5]  506936431099073  681216114870865  383984555235855  623907220941615
##  [9] 1272981499433879 1010770822718490 1270704382287800
```

```
# FOR exponential smoothing
# Make a function
forcst_se <- function (data, alpha) {
  df_se <- data.frame(t = 1:length(data), x = data, frct = 0, err = 0)
  df_se$frct[1] <- df_se$x[1]
  df_se$err[1] <- 0
  for (v in 2:(length(data))) {
    df_se$frct[v] <- df_se$frct[v-1] + (alpha * df_se$err[v-1])
    df_se$err[v] <- df_se$x[v] - df_se$frct[v]
  }
  return (df_se$frct[v] + (alpha * df_se$err[v]))
}

# Use function to make forecast and get the Square Error
nw_mobgr_df$ES_SE <- 0

for (x in 3:nrow(nw_mobgr_df)) {
  # forecast ith value with E/S
  frcst_se <- forcst_se(nw_mobgr_df$Subscribers[1:(x-1)], alpha = 0.4)

  # calculate MSE
  nw_mobgr_df$ES_SE[x] <- (nw_mobgr_df$Subscribers[x] - frcst_se)^2

}
SE_se <- nw_mobgr_df$ES_SE

# FOR LM

# Make a function
forcst_lm <- function (data, t) {
  ind = 1:length(data)
  lrm_df <- data.frame(x = ind, y = data)
  lrm_mod <- lm(y ~ x, data = lrm_df)
  lrm_forc <- lrm_mod$coefficients[[2]] * (t) + lrm_mod$coefficients[[1]]
```

```
  return (lrm_forc)
}
# Use function to make forecast and get the Square Error
nw_mobgr_df$LRM_SE <- 0

for (x in 1:nrow(nw_mobgr_df)) {
  frcst_lrm <- forcst_lm(nw_mobgr_df$Subscribers, x)
  nw_mobgr_df$LRM_SE[x] <- (nw_mobgr_df$Subscribers[x] - frcst_lrm)^2
}
LRM_se <- nw_mobgr_df$LRM_SE
```

The Squared error for weighted moving average forecast is "0", "0", "59645966892994.3", "175435716611878", "506936431099073", "681216114870865", "383984555235855", "623907220941615", "1272981499433879", "1010770822718490", "1270704382287800".

The Squared error for exponantial smoothing forecast is "0", "0", "89676237032614.5", "294951575233242", "872388679876229", "1468964443555259", "1347222387777106", "1856248194741708", "3081562086349401", "3206397035352714", "3963923693400346".

The Squared error for Linear regression Trendline forecast is "425274405439071", "62457915080405.8", "17965063041746.3", "121505816313992", "101364261918434", "59898225115886.9", "151483361387229", "90689891758262.7", "3464912710041.63", "47646497704808.9", "310131855532801".

Question 3.

Calculate the average (mean) squared error for each model.

```
WMA_mse <- mean(nw_mobgr_df$WMA_SE)
SE_mse<- mean(nw_mobgr_df$ES_SE)
LRM_mse<- mean(nw_mobgr_df$LRM_SE)
```

Answers: For each forecasted year:

The Mean Squared error for weighted moving average forecast is "544143882735677" respectively.
The Mean Squared error for exponential smoothing forecast is "1471030393938056" respectively.
The Mean Squared error for Linear regression Trendline forecast is "126534746000244" respectively.

Question 4.

Which model has the smallest mean squared error (MSE)?

```
MSE_list <- c(WMA_mse, SE_mse, LRM_mse)
lowest_mse <- min(MSE_list)
```

Answers: The lowest MSE is "126534746000244" which is for Linear Regression model.

Question 5.

Write a function called ensembleForecast() that calculates a weighted average forecast by averaging out the three forecasts calculated with the following weights: 4 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.

```
ensembleForecast <- function (data, n, alpha, t, WMA_wt, ens_wt_list) {
  wma_frct <- forcst_wma(data, n, WMA_wt)
  #print(wma_frct)
  se_frct <- forcst_se(data, alpha)
  #print(se_frct)
  lrm_frct <- forcst_lm(data, t)
  #print(lrm_frct)
  f <- (ens_wt_list[1]*lrm_frct + ens_wt_list[2]*se_frct + ens_wt_list[3]*wma_frct) / sum(ens_wt_list)
}
ens_forcst12 <-  ensembleForecast(nw_mobgr_df$Subscribers, 2, 0.4, 12, c(5,2), c(4,2,1))
#ens_forcst12
```

Answers: The forecast for the next year is "191348572.733396" when all three forecast models are collective considered.