Practice Problems 2

Jordan Lian

2/07/2021

1. (15 pts) 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.

```
# New Data Frame
df <- USArrests

# Reference Statistics
mean(df$Murder)
```

```
## [1] 7.788
```

```
1.5 * sd(df$Murder)
```

```
## [1] 6.533265
```

```
# Use row.names()
row.names(df)[abs(df$Murder - mean(df$Murder)) > (1.5*sd(df$Murder))]
```

```
## [1] "Florida"        "Georgia"        "Louisiana"      "Mississippi"
## [5] "North Dakota"   "South Carolina"
```

2. (15 pts) For the same dataset as in (1), 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. Calculate the Pearson coefficient of correlation in R.

```
# Libraries
library(ggplot2)
library(ggpubr)

# Correlation Calculation
cor.test(df$Murder, df$UrbanPop)
```

```
##
##  Pearson's product-moment correlation
##
## data:  df$Murder and df$UrbanPop
## t = 0.48318, df = 48, p-value = 0.6312
## alternative hypothesis: true correlation is not equal to 0
```

```
## 95 percent confidence interval:
##  -0.2128979  0.3413107
## sample estimates:
##        cor
## 0.06957262
```

```r
# Plot: Urban Population vs Murder
ggplot(df, aes(x=Murder, y=UrbanPop)) +
  geom_point() +
  geom_smooth() +
  stat_cor(method = "pearson", label.x = 7.5, label.y = 35) +
  ylab("Urban Population") +
  ggtitle("Urban Population vs Murder") +
  theme(plot.title = element_text(face = "bold", hjust = 0.5))
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

The correlation value is 0.07, which shows basically no correlation between urban population and murder. From what I have read in my spare time, law enforcement and public policy have more to do with murder rather than urban population.

3. (3 x 10 pts) Based on the data on the growth of mobile phone use in Brazil (you will 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.

I created separate data frames for each forecasting technique to show my results.

Using gsheet2tbl() to load dataset into R, store link as variable URL

[1] "https://docs.google.com/spreadsheets/d/1tOnM9XceK4Ak8tzWQ2vDelWlJexzJiS3LbT6MN6_rW0/edit?usp=sharing"

```
# Import dataset Using gsheet2tbl()
library(gsheet)
brazil <- gsheet2tbl(url)
brazil
```

```
## # A tibble: 12 x 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
```

2-year weighted moving average

$F_t = \frac{W_1 * D_{t-1} + W_2 * D_{t-2}}{W_1 + W_2}$, where F_t is the forecast, D_t is the demand (subscribers in this case), and W_t is the assigned weight to the demand value.

```
# New Data Frame, Initialize New Column
weight_MA <- brazil
weight_MA$Forecast <- rep(NA, 12)

# Store Weights Into a Vector
weights <- c(2, 5)

# Use weighted.mean() to get the weighted averages
for(i in 3:12){
  prev_2 <- c(brazil$Subscribers[(i-2): (i-1)])
  weight_MA$Forecast[i] <- weighted.mean(prev_2, weights)
}

# Forecast Data
weight_MA
```

```
## # A tibble: 12 x 3
##     Year Subscribers   Forecast
##    <dbl>       <dbl>      <dbl>
```

```
## 1      1     23188171        NA
## 2      2     28745769        NA
## 3      3     34880964  27157884.
## 4      4     46373266  33128051.
## 5      5     65605000  43089751.
## 6      6     86210336  60110219.
## 7      7     99918621  80323097.
## 8      8    120980103  96001968.
## 9      9    150641403 114962537.
## 10    10    173959368 142166746.
## 11    11    202944033 167297092.
## 12    12           NA 194662700.
```

```r
# Forecast for the Next Time Period
weight_MA$Forecast[12]
```

```
## [1] 194662700
```

Exponential Smoothing (alpha = 0.4)

$F_{t+1} = F_t + \alpha(D_t - F_t)$, where F_t is the forecast and D_t is the demand (subscribers in this case). Since this forecast uses previous forecast values to calculate the current forecast, we must initialize one value to start out.

```r
# New Data Frame, Initialize New Column
expo_smooth <- brazil
expo_smooth$Forecast <- rep(NA, 12)

# Initialize First Forecast Value using Weighted Moving Average at Year 3
expo_smooth$Forecast[3] = weight_MA$Forecast[3]

# Get Forecast Values
for(i in 4:12){
  smoothing <- 0.4*(brazil$Subscribers[i-1] - expo_smooth$Forecast[i-1])
  expo_smooth$Forecast[i] <- expo_smooth$Forecast[i-1] + smoothing
}

# Forecast Data
expo_smooth
```

```
## # A tibble: 12 x 3
##     Year Subscribers   Forecast
##    <dbl>       <dbl>      <dbl>
## 1      1    23188171         NA
## 2      2    28745769         NA
## 3      3    34880964  27157884.
## 4      4    46373266  30247116.
## 5      5    65605000  36697576.
## 6      6    86210336  48260546.
## 7      7    99918621  63440462.
## 8      8   120980103  78031725.
## 9      9   150641403  95211076.
```

```
## 10    10   173959368 117383207.
## 11    11   202944033 140013671.
## 12    12          NA 165185816.
```

```
# Forecast for the Next Time Period
expo_smooth$Forecast[12]
```

```
## [1] 165185816
```

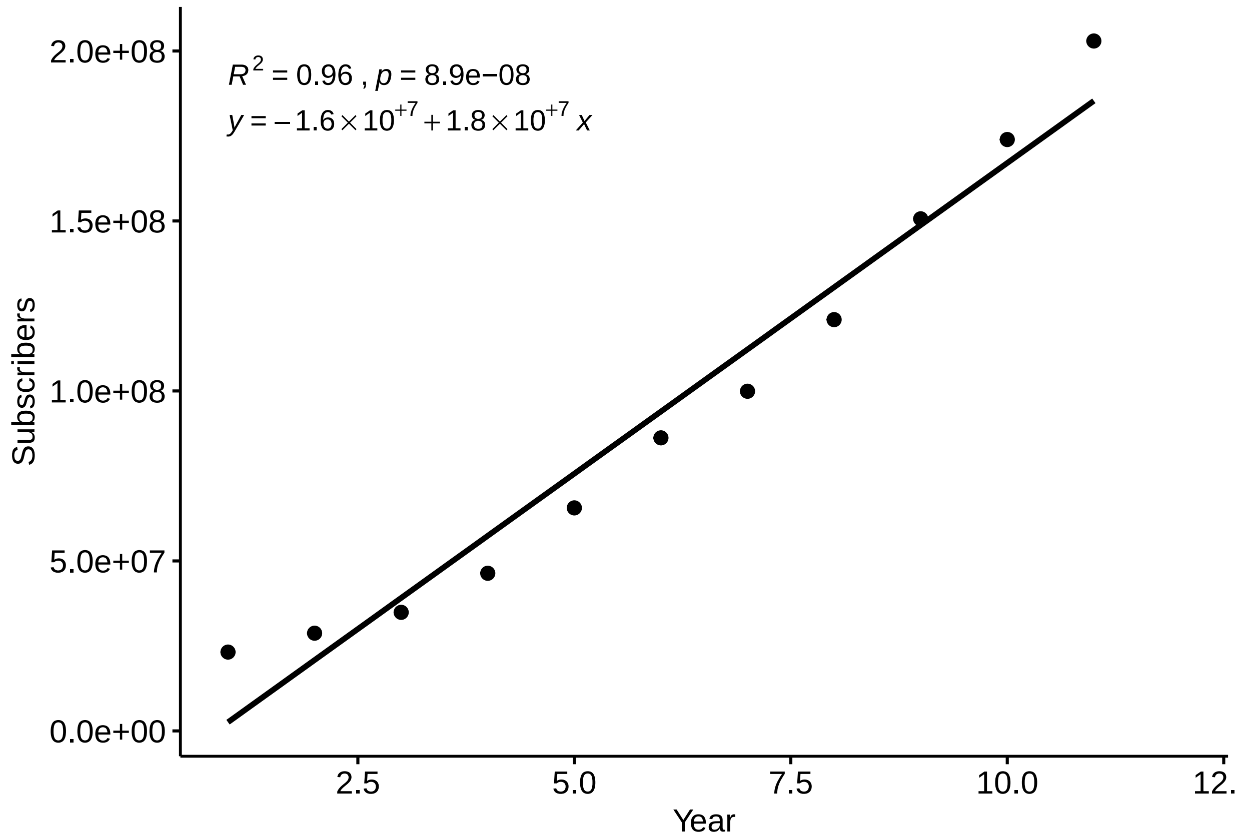
Linear Regression Trendline

```
# Get Regression Equation for Forecast Values
lm(Subscribers ~ Year, data = brazil)
```

```
##
## Call:
## lm(formula = Subscribers ~ Year, data = brazil)
##
## Coefficients:
## (Intercept)         Year
##   -15710760     18276748
```

```
# Plot the Data to Visualize
ggscatter(brazil, x = "Year", y = "Subscribers", add = "reg.line") +
  stat_cor(aes(label = paste(..rr.label.., ..p.label.., sep = "~`,`~"))) +
  stat_regline_equation(label.y = 1.8*100000000)
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```r
# New Data Frame, Initialize New Column
lin_reg <- brazil
lin_reg$Forecast <- rep(NA, 12)

# Get Forecast Values
for(i in 1:12){
  lin_reg$Forecast[i] <- -15710760 + (18276748 * i)
}

# Forecast Data
lin_reg
```

```
## # A tibble: 12 x 3
##     Year Subscribers  Forecast
##    <dbl>       <dbl>     <dbl>
##  1     1    23188171   2565988
##  2     2    28745769  20842736
##  3     3    34880964  39119484
##  4     4    46373266  57396232
##  5     5    65605000  75672980
##  6     6    86210336  93949728
##  7     7    99918621 112226476
##  8     8   120980103 130503224
##  9     9   150641403 148779972
## 10    10   173959368 167056720
## 11    11   202944033 185333468
```

```
## 12    12           NA 203610216
```

```r
# Forecast for the Next Time Period
lin_reg$Forecast[12]
```

```
## [1] 203610216
```

4. (20 pts) Calculate the squared error for each model, i.e., use the model to calculate a forecast for each given time period and then the squared error. Finally, calculate the average (mean) squared error for each model. Which model has the smallest mean squared error (MSE)?

$E_t = F_t - D_t$, where E_t is the Error value

$$MSE = \frac{1}{n} * \sum_{i=1}^{n} (E_i)^2$$

Weighted Moving Average

```r
# Create Error and Square Error Columns
weight_MA$Error <- weight_MA$Forecast - weight_MA$Subscribers
weight_MA$Sq_Error <- weight_MA$Error^2

# Square Error for Time Periods
weight_MA
```

```
## # A tibble: 12 x 5
##     Year Subscribers   Forecast      Error Sq_Error
##    <dbl>       <dbl>      <dbl>      <dbl>    <dbl>
##  1     1    23188171         NA         NA  NA
##  2     2    28745769         NA         NA  NA
##  3     3    34880964  27157884.  -7723080.  5.96e13
##  4     4    46373266  33128051. -13245215.  1.75e14
##  5     5    65605000  43089751. -22515249.  5.07e14
##  6     6    86210336  60110219. -26100117.  6.81e14
##  7     7    99918621  80323097. -19595524.  3.84e14
##  8     8   120980103  96001968. -24978135.  6.24e14
##  9     9   150641403 114962537. -35678866.  1.27e15
## 10    10   173959368 142166746. -31792622.  1.01e15
## 11    11   202944033 167297092. -35646941.  1.27e15
## 12    12          NA 194662700.         NA  NA
```

```r
# MSE
weight_MA_MSE <- mean(weight_MA$Sq_Error, na.rm = T)
weight_MA_MSE
```

```
## [1] 6.650647e+14
```

Exponential Smoothing

```r
# Create Error and Square Error Columns
expo_smooth$Error <- expo_smooth$Forecast - expo_smooth$Subscribers
expo_smooth$Sq_Error <- expo_smooth$Error^2

# Square Error for Time Periods
expo_smooth
```

```
## # A tibble: 12 x 5
##     Year Subscribers   Forecast      Error Sq_Error
##    <dbl>       <dbl>      <dbl>      <dbl>    <dbl>
##  1     1    23188171        NA          NA  NA
##  2     2    28745769        NA          NA  NA
##  3     3    34880964  27157884.  -7723080.  5.96e13
##  4     4    46373266  30247116. -16126150.  2.60e14
##  5     5    65605000  36697576. -28907424.  8.36e14
##  6     6    86210336  48260546. -37949790.  1.44e15
##  7     7    99918621  63440462. -36478159.  1.33e15
##  8     8   120980103  78031725. -42948378.  1.84e15
##  9     9   150641403  95211076. -55430327.  3.07e15
## 10    10   173959368 117383207. -56576161.  3.20e15
## 11    11   202944033 140013671. -62930362.  3.96e15
## 12    12          NA 165185816.        NA   NA
```

```r
# MSE
expo_smooth_MSE <- mean(expo_smooth$Sq_Error, na.rm = T)
expo_smooth_MSE
```

```
## [1] 1.778262e+15
```

Linear Regression

```r
# Create Error and Square Error Columns
lin_reg$Error <- lin_reg$Forecast - lin_reg$Subscribers
lin_reg$Sq_Error <- lin_reg$Error^2

# Square Error for Time Periods
lin_reg
```

```
## # A tibble: 12 x 5
##     Year Subscribers  Forecast     Error Sq_Error
##    <dbl>       <dbl>     <dbl>     <dbl>    <dbl>
##  1     1    23188171   2565988 -20622183  4.25e14
##  2     2    28745769  20842736  -7903033  6.25e13
##  3     3    34880964  39119484   4238520  1.80e13
##  4     4    46373266  57396232  11022966  1.22e14
##  5     5    65605000  75672980  10067980  1.01e14
##  6     6    86210336  93949728   7739392  5.99e13
##  7     7    99918621 112226476  12307855  1.51e14
##  8     8   120980103 130503224   9523121  9.07e13
##  9     9   150641403 148779972  -1861431  3.46e12
```

```
## 10    10   173959368 167056720  -6902648  4.76e13
## 11    11   202944033 185333468 -17610565  3.10e14
## 12    12          NA 203610216        NA NA
```

```
# MSE
lin_reg_MSE <- mean(lin_reg$Sq_Error, na.rm = T)
lin_reg_MSE
```

```
## [1] 1.265347e+14
```

Smallest Mean Squared Error (MSE)

Using the min() function, I found that the linear regression forecast produced the smallest MSE.

```
min(weight_MA_MSE, expo_smooth_MSE, lin_reg_MSE)
```

```
## [1] 1.265347e+14
```

5. (20 pts) Calculate a weighted average forecast by averaging out the three forecasts calculated in (3) 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.

```
# New Data Frame, Initialize New Column
weight_AF <- brazil
weight_AF$Forecast <- rep(NA, 12)

# Store Weights Into a Vector
forecast_weights <- c(4, 2, 1)

# Use weighted.mean() to get the weighted averages
for(i in 3:12){
  forecast_vals <- c(lin_reg$Forecast[i], expo_smooth$Forecast[i], weight_MA$Forecast[i])
  weight_AF$Forecast[i] <- weighted.mean(forecast_vals, forecast_weights)
}

# Forecast Data
weight_AF
```

```
## # A tibble: 12 x 3
##     Year Subscribers   Forecast
##    <dbl>       <dbl>      <dbl>
##  1     1    23188171         NA
##  2     2    28745769         NA
##  3     3    34880964  33993084.
##  4     4    46373266  46172459.
##  5     5    65605000  59882403.
##  6     6    86210336  76061460.
##  7     7    99918621  93729989.
##  8     8   120980103 110582616.
##  9     9   150641403 128643511.
## 10    10   173959368 149308577.
## 11    11   202944033 169808330.
## 12    12          NA 191353599.
```