Library Imports

```
library(glue)
library(DAAG)

options(repr.plot.width=10, repr.plot.height=5)

set.seed(20)
options(warn=-1)
```

```
Loading required package: lattice
```

Question 7.1

Describe a situation or problem from your job, everyday life, current events, etc., for which exponential smoothing would be appropriate. What data would you need? Would you expect the value of α (the first smoothing parameter) to be closer to 0 or 1, and why?

ANSWER: The first that comes to mind when thinking of exponential smoothing and its uses for short-term forecasting is the stock market. Nowadays, there are numerous firms that try to use machine learning/AI to predict the prices of stocks in order to make lots of profit.

The data I would need is stock price data in minute or maybe even second intervals. Seeing as how volatile stock prices are, I would say the alpha value would be near 1 so a lot of weight would be placed on the true current observation and little weight on the previous observation.

Question 7.2

Using the 20 years of daily high temperature data for Atlanta (July through October) from Question 6.2 (file temps.txt), build and use an exponential smoothing model to help make a judgment of whether the unofficial end of summer has gotten later over the 20 years. (Part of the point of this assignment is for you to think about how you might use exponential smoothing to answer this question. Feel free to combine it with other models if you'd like to. There's certainly more than one reasonable approach.) Note: in R, you can use either HoltWinters (simpler to use) or the smooth package's es function (harder to use, but more general). If you use es, the Holt-Winters model uses model="AAM" in the function call (the first and second constants are used "A"dditively, and the third (seasonality) is used "M"ultiplicatively; the documentation doesn't make that clear).

```
temp_data <- read.table('../data/7.2tempsSummer2018.txt', sep='', header=TRUE)

temp_data$datetime <- as.Date(temp_data$DAY, format = "%d-%b")
```

First, I transformed the data into a time series data frame with a frequency of 123 since there are 123 days in the original data set.

```
temp_ts  <- ts(as.vector(temp_data[,c(2:21)]), start=1996, frequency=123)
```

The transformed data set was then inputted into the Holt Winters function and the smoothed data was plotted as a function of time.

The plot is in the appendix.

```
hw <- HoltWinters(temp_ts)
plot.ts(fitted(hw)[,1], col=3, ylab="Temperature (F)", xlab="Year")
```

The resulting seasonal factors were transformed into a matrix and used in the CUSUM approach to find the date at which summer ended.

```
m <- matrix(hw$fitted[,4], ncol=123)
```

Custom CUSUM function that I coded in R is below.

```
# cusum_approach takes in the data of interest, threshold (T), critical value (C),
mean (mu),
# and a boolean value for is_inc.
# The is_inc determines whether the change that is being detected is an increase or
decrease.
cusum_approach <- function(data, T, C, mu, is_inc) {
    S_t <- 0
    end_index <- length(data)

    if (isTRUE(is_inc)) {
        for (i in 1:length(data)) {
            S_t <- S_t - mu + data[i] - C
            # Check if the S_t is less than 0, if it is then it should be
            # reset to 0
            if (S_t <= 0) {
                S_t <- 0
                }
            # Check if S_t is greater than the threshold, if it is then
            # the loop ends
            if (S_t > T) {
                end_index <- i
                break
            }
        }
    }

    else {
        for (i in 1:length(data)) {
            S_t <- S_t + mu - data[i] - C
            # Check if the S_t is less than 0, if it is then it should be
            # reset to 0
            if (S_t <= 0) {
                S_t <- 0
```

```
                }
            # Check if S_t is greater than the threshold, if it is then
            # the loop ends
            if (S_t > T) {
                end_index <- i
                break
            }
        }
    }
    return(end_index)
}
```

I tested a variety of C and T values and this combination of C = 1 and T = 25 seemed to give the most reasonable summer end dates.

```
C <- 1
T <- 20

for (i in 1:nrow(m)) {
    year_data <- m[i,]

    mu <- mean(year_data)
    end_index <- cusum_approach(year_data, T, C, mu, is_inc=FALSE)

    summer_end_date <- temp_data[end_index,1]
    year <- substr(colnames(temp_ts)[i], 2, 5)
    print(glue("Summer ends on {summer_end_date} in the year {year}"))
}
```

```
Summer ends on 7-Jul in the year 1996
Summer ends on 7-Jul in the year 1997
Summer ends on 7-Jul in the year 1998
Summer ends on 6-Jul in the year 1999
Summer ends on 26-Jul in the year 2000
Summer ends on 13-Jul in the year 2001
Summer ends on 6-Jul in the year 2002
Summer ends on 3-Sep in the year 2003
Summer ends on 26-Jul in the year 2004
Summer ends on 8-Aug in the year 2005
Summer ends on 13-Jul in the year 2006
Summer ends on 13-Jul in the year 2007
Summer ends on 12-Jul in the year 2008
Summer ends on 1-Aug in the year 2009
Summer ends on 6-Jul in the year 2010
Summer ends on 6-Jul in the year 2011
Summer ends on 6-Jul in the year 2012
Summer ends on 27-Aug in the year 2013
Summer ends on 14-Aug in the year 2014
```

For the same C and T values, I ran the original data through the CUSUM approach to compare the summer end dates before and after exponential smoothing.

```
for (i in 2:21) {
    # Initialize the mean (mu)
    mu <- mean(temp_data[,i])

    # Get the index at which the threshold is passed.
    end_index <- cusum_approach(temp_data[,i], T, C, mu, is_inc=FALSE)

    summer_end_date <- temp_data[end_index,1]
    year <- substr(colnames(temp_data)[i], 2, 5)
    print(glue("Summer ends on {summer_end_date} in the year {year}"))

    }
```

```
Summer ends on 21-Sep in the year 1996
Summer ends on 25-Sep in the year 1997
Summer ends on 30-Sep in the year 1998
Summer ends on 20-Sep in the year 1999
Summer ends on 6-Sep in the year 2000
Summer ends on 25-Sep in the year 2001
Summer ends on 25-Sep in the year 2002
Summer ends on 29-Sep in the year 2003
Summer ends on 19-Sep in the year 2004
Summer ends on 7-Oct in the year 2005
Summer ends on 21-Sep in the year 2006
Summer ends on 17-Sep in the year 2007
Summer ends on 21-Sep in the year 2008
Summer ends on 2-Oct in the year 2009
Summer ends on 28-Sep in the year 2010
Summer ends on 7-Sep in the year 2011
Summer ends on 30-Sep in the year 2012
Summer ends on 16-Aug in the year 2013
Summer ends on 26-Sep in the year 2014
Summer ends on 16-Sep in the year 2015
```

The exponential smoothing provided summer end dates that are much earlier than the end dates estimated on the original data set. This was actually counter intuitive for me. I expected the end dates to actually be later after exponential smoothing because the threshold would be harder to surpass if the data was more stable. However, the opposite was true. My hypothesis as to why this happened is that the volatility in the temperature data causes the S_t value to fluctuate a lot. So, the S_t value can come close to the threshold value but if the next day a sudden drop in temperature happened then S_t could now be very far from the threshold. The exponential smoothing prevents such sudden changes in S_t by smoothing out the temperature data.

Question 8.1

Describe a situation or problem from your job, everyday life, current events, etc., for which a linear regression model would be appropriate. List some (up to 5) predictors that you might use.

ANSWER:

Linear regression is useful when predicting house prices. Especially in the Bay Area where the housing market is so high, its important to be able to accurately predict the housing price so that you don't overpay on an already ridiculously highly priced house.

5 predictors to use: 1. Square footage 2. Distance of bus stop or subway stations 3. Year it was built 4. Number of bedrooms 5. Number of bathrooms

Question 8.2

Using crime data from http://www.statsci.org/data/general/uscrime.txt (file uscrime.txt, description at http://www.statsci.org/data/general/uscrime.html), use regression (a useful R function is lm or glm) to predict the observed crime rate in a city with the following data:

- M = 14.0
- So = 0
- Ed = 10.0
- Po1 = 12.0
- Po2 = 15.5
- LF = 0.640
- M.F = 94.0
- Pop = 150
- NW = 1.1
- U1 = 0.120
- U2 = 3.6
- Wealth = 3200
- Ineq = 20.1
- Prob = 0.04
- Time = 39.0

Show your model (factors used and their coefficients), the software output, and the quality of fit. Note that because there are only 47 data points and 15 predictors, you'll probably notice some overfitting. We'll see ways of dealing with this sort of problem later in the course.

```
crime_data <- read.table('../data/5.1uscrimeSummer2018.txt', sep='', header=TRUE)
```

```
M  <-  14.0
So <- 0
Ed <- 10.0
Po1 <- 12.0
Po2 <- 15.5
LF <- 0.640
M.F <- 94.0
Pop <- 150
NW <- 1.1
U1 <- 0.120
U2 <- 3.6
Wealth <- 3200
```

```
Ineq <- 20.1
Prob <- 0.04
Time <- 39.0

sample_data <- data.frame(M,So,Ed,Po1,Po2,LF,M.F,Pop,NW,U1,U2,Wealth,Ineq,Prob,Time)
```

Use cross-validation to determine the root mean squared error (RMSE) and the mean absolute error (MAE) for each fold in the cross-validation step.

```
# RMSE function
rmse <- function(resids) {
    sqrt(mean(resids^2))
}
# MAE function
mae <- function(resids) {
    mean(abs(resids))
}
```

```
# Shuffle data
shuffled_df <- crime_data[sample(nrow(crime_data)), ]

# 5 fold split
folds <- cut(seq(1, nrow(crime_data)), breaks=5, labels=FALSE)

# For each fold, train and test on Linear Regression algorithm
resids <- c(1:nrow(crime_data))
preds <- c(1:nrow(crime_data))
for (i in 1:5) {
    print(glue("Iteration #{i}"))
    test_indices <- which(folds==i, arr.ind=TRUE)
    test_data <- crime_data[test_indices, ]
    train_data <- crime_data[-test_indices, ]

    temp_fit <- lm(formula=Crime ~., data=train_data)
    print(summary(temp_fit))

    # Perform prediction step on test data since model was fit using the training
data
    preds[test_indices] <- predict(temp_fit, test_data)
    resids[test_indices] <- test_data$Crime - preds[test_indices]
}

overall_rmse <- round(rmse(resids), 3)
overall_mae <- round(mae(resids), 3)

print(glue("RMSE: {overall_rmse}"))
print(glue("MAE: {overall_mae}"))
```

```
Iteration #1

Call:
lm(formula = Crime ~ ., data = train_data)
```

```
Residuals:
    Min      1Q  Median      3Q     Max 
-354.88  -86.44   -4.82   77.37  467.47 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)   
(Intercept) -6104.9296  2068.7312  -2.951  0.00763 **
M             102.2760    60.1488   1.700  0.10382   
So           -114.2915   195.7346  -0.584  0.56550   
Ed            120.1692    77.2415   1.556  0.13471   
Po1           174.3027   118.6461   1.469  0.15663   
Po2          -112.9272   131.6819  -0.858  0.40081   
LF           -764.5198  1700.0301  -0.450  0.65753   
M.F            15.9682    24.5602   0.650  0.52263   
Pop            -1.2496     1.6571  -0.754  0.45919   
NW              4.3656     7.9608   0.548  0.58921   
U1          -5307.2621  5371.9166  -0.988  0.33442   
U2            148.8000   103.3013   1.440  0.16448   
Wealth          0.2596     0.1493   1.739  0.09666 . 
Ineq           77.4564    28.8980   2.680  0.01401 * 
Prob        -3938.1110  2897.3326  -1.359  0.18850   
Time           -2.5860     8.2228  -0.314  0.75625   
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 217.9 on 21 degrees of freedom
Multiple R-squared:  0.7718,	Adjusted R-squared:  0.6089 
F-statistic: 4.736 on 15 and 21 DF,  p-value: 0.0006456

Iteration #2

Call:
lm(formula = Crime ~ ., data = train_data)

Residuals:
     Min       1Q   Median       3Q      Max 
-231.958  -83.934   -2.402   64.632  259.599 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) -3.485e+03  1.436e+03  -2.427  0.02387 *  
M            1.295e+02  3.617e+01   3.581  0.00166 ** 
So          -1.494e+01  1.250e+02  -0.120  0.90589    
Ed           2.879e+02  5.360e+01   5.371 2.16e-05 ***
Po1          2.097e+02  8.736e+01   2.400  0.02530 *  
Po2         -1.139e+02  9.575e+01  -1.189  0.24708    
LF          -9.398e+02  1.164e+03  -0.807  0.42814    
M.F         -1.025e+01  1.726e+01  -0.594  0.55865    
Pop         -1.415e+00  1.026e+00  -1.380  0.18154    
NW           1.052e+00  6.287e+00   0.167  0.86859    
U1          -7.455e+03  3.994e+03  -1.867  0.07535 .  
U2           2.569e+02  7.525e+01   3.414  0.00248 ** 
Wealth      -7.612e-02  8.877e-02  -0.858  0.40041    
Ineq         7.050e+01  1.829e+01   3.854  0.00086 ***
Prob        -9.381e+03  2.691e+03  -3.486  0.00209 ** 
Time        -1.596e+01  6.722e+00  -2.375  0.02670 *  
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 156 on 22 degrees of freedom
Multiple R-squared:  0.9092,	Adjusted R-squared:  0.8473 
F-statistic: 14.68 on 15 and 22 DF,  p-value: 3.959e-08

Iteration #3

Call:
lm(formula = Crime ~ ., data = train_data)

Residuals:
    Min      1Q  Median      3Q     Max 
-381.60 -108.29   12.58  114.69  521.04 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)   
(Intercept) -6.988e+03  2.101e+03  -3.326  0.00307 **
M            6.918e+01  4.982e+01   1.388  0.17892   
So          -1.415e+01  1.940e+02  -0.073  0.94252   
Ed           2.481e+02  7.439e+01   3.335  0.00300 **
Po1          2.203e+02  1.378e+02   1.599  0.12411   
Po2         -1.373e+02  1.447e+02  -0.949  0.35298   
LF          -1.235e+03  1.797e+03  -0.687  0.49898   
M.F          3.043e+01  2.462e+01   1.236  0.22951   
Pop         -8.035e-01  1.555e+00  -0.517  0.61058   
NW           8.875e+00  8.069e+00   1.100  0.28330   
U1          -6.551e+03  5.050e+03  -1.297  0.20803   
U2           1.744e+02  1.055e+02   1.653  0.11258   
Wealth       1.143e-02  1.291e-01   0.089  0.93027   
Ineq         5.951e+01  3.137e+01   1.897  0.07099 . 
Prob        -2.784e+03  2.667e+03  -1.044  0.30789   
Time         5.304e+00  9.232e+00   0.575  0.57146   
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 224.6 on 22 degrees of freedom
Multiple R-squared:  0.763,	Adjusted R-squared:  0.6015 
F-statistic: 4.723 on 15 and 22 DF,  p-value: 0.0005457

Iteration #4

Call:
lm(formula = Crime ~ ., data = train_data)

Residuals:
    Min      1Q  Median      3Q     Max 
-400.37  -80.37    0.42   94.20  403.24 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)  
(Intercept) -5.098e+03  1.884e+03  -2.706   0.0129 *
M            6.709e+01  6.386e+01   1.051   0.3048  
So          -8.101e+01  1.816e+02  -0.446   0.6599  
Ed           1.496e+02  8.715e+01   1.716   0.1002  
Po1          2.475e+02  1.417e+02   1.747   0.0946 .
Po2         -1.702e+02  1.600e+02  -1.064   0.2989  
```

```
LF          -8.118e+02  2.644e+03  -0.307   0.7617
M.F          2.223e+01  2.878e+01   0.773   0.4480
Pop          1.416e+00  1.836e+00   0.771   0.4487
NW           8.377e+00  7.792e+00   1.075   0.2940
U1          -3.955e+03  4.912e+03  -0.805   0.4294
U2           1.342e+02  9.980e+01   1.345   0.1923
Wealth       4.299e-02  1.281e-01   0.335   0.7404
Ineq         4.937e+01  3.203e+01   1.541   0.1375
Prob        -4.536e+03  2.597e+03  -1.747   0.0946 .
Time        -4.215e+00  8.548e+00  -0.493   0.6268
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 218.4 on 22 degrees of freedom
Multiple R-squared:  0.8327,    Adjusted R-squared:  0.7187
F-statistic: 7.302 on 15 and 22 DF,  p-value: 1.958e-05

Iteration #5

Call:
lm(formula = Crime ~ ., data = train_data)

Residuals:
    Min      1Q  Median      3Q     Max
-304.60 -109.24  -38.76  124.87  401.79

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -7536.8930  2240.1291  -3.364  0.00293 **
M              72.1891    47.9392   1.506  0.14700
So             94.6490   182.1750   0.520  0.60881
Ed            147.4452    73.9349   1.994  0.05927 .
Po1           188.3669   164.1972   1.147  0.26420
Po2          -110.4119   183.1552  -0.603  0.55308
LF          -1139.1182  1740.3542  -0.655  0.51987
M.F            37.8183    24.1592   1.565  0.13244
Pop            -0.4180     1.5172  -0.275  0.78563
NW              2.3102     7.6550   0.302  0.76578
U1          -6183.4096  4799.1947  -1.288  0.21161
U2            129.5443    94.5294   1.370  0.18503
Wealth          0.1775     0.1133   1.567  0.13214
Ineq           81.8189    25.3099   3.233  0.00399 **
Prob        -5497.2661  2751.7377  -1.998  0.05886 .
Time           -1.2663    10.1290  -0.125  0.90170
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 209.9 on 21 degrees of freedom
Multiple R-squared:  0.8489,    Adjusted R-squared:  0.741
F-statistic: 7.867 on 15 and 21 DF,  p-value: 1.47e-05

RMSE: 289.558
MAE: 221.229
```

I also want to see the residual plot as well for the predicted values.

The plot is in the appendix.

```
xyplot(resids ~ preds,
  xlab = "Fitted Values",
  ylab = "Residuals",
  main = "Residual Diagnostic Plot",
  panel = function(x, y, ...)
  {
    panel.grid(h = -1, v = -1)
    panel.abline(h = 0)
    panel.xyplot(x, y, ...)
  }
)
```

From the summary of the models above, it seems that the features that seemed most useful were Ineq, Ed, M, Po1, U2, Prob, and Time. Using this subset of predictors, I'll build another linear regression model.

```
# Shuffle data
shuffled_df <- crime_data[sample(nrow(crime_data)), ]

# 5 fold split
folds <- cut(seq(1, nrow(crime_data)), breaks=5, labels=FALSE)

# For each fold, train and test on Linear Regression algorithm
resids <- c(1:nrow(crime_data))
preds <- c(1:nrow(crime_data))
for (i in 1:5) {
    print(glue("Iteration #{i}"))
    test_indices <- which(folds==i, arr.ind=TRUE)
    test_data <- crime_data[test_indices, ]
    train_data <- crime_data[-test_indices, ]

    temp_fit <- lm(formula=Crime ~ M + Ed + Po1 + U2 + Ineq + Prob + Time,
data=train_data)
    print(summary(temp_fit))

    # Perform prediction step on test data since model was fit using the training
data
    preds[test_indices] <- predict(temp_fit, test_data)
    resids[test_indices] <- test_data$Crime - preds[test_indices]
}

overall_rmse <- round(rmse(resids), 3)
overall_mae <- round(mae(resids), 3)

print(glue("RMSE: {overall_rmse}"))
print(glue("MAE: {overall_mae}"))
```

```
Iteration #1

Call:
```

```
lm(formula = Crime ~ M + Ed + Po1 + U2 + Ineq + Prob + Time,
    data = train_data)

Residuals:
    Min      1Q  Median      3Q     Max
-463.41 -113.86    0.58   68.57  592.99

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -4568.594   1241.738  -3.679 0.000949 ***
M             106.720     38.893   2.744 0.010304 *
Ed            176.508     59.748   2.954 0.006163 **
Po1           109.048     16.902   6.452 4.65e-07 ***
U2             81.040     48.031   1.687 0.102287
Ineq           58.293     17.056   3.418 0.001891 **
Prob        -4109.257   2127.567  -1.931 0.063255 .
Time           -1.312      6.202  -0.212 0.833895
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 215.3 on 29 degrees of freedom
Multiple R-squared:  0.6924,    Adjusted R-squared:  0.6182
F-statistic: 9.327 on 7 and 29 DF,  p-value: 5.151e-06

Iteration #2

Call:
lm(formula = Crime ~ M + Ed + Po1 + U2 + Ineq + Prob + Time,
    data = train_data)

Residuals:
     Min       1Q   Median       3Q      Max
-301.239 -132.051   -1.481  118.213  242.193

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -5213.611    809.713  -6.439 4.12e-07 ***
M             133.082     31.328   4.248 0.000192 ***
Ed            201.936     43.540   4.638 6.46e-05 ***
Po1           110.910     13.403   8.275 3.09e-09 ***
U2            110.216     36.692   3.004 0.005338 **
Ineq           72.327     13.255   5.457 6.42e-06 ***
Prob        -6561.410   2178.158  -3.012 0.005225 **
Time          -10.483      5.571  -1.882 0.069617 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 167.6 on 30 degrees of freedom
Multiple R-squared:  0.8571,    Adjusted R-squared:  0.8237
F-statistic:  25.7 on 7 and 30 DF,  p-value: 5.114e-11

Iteration #3

Call:
lm(formula = Crime ~ M + Ed + Po1 + U2 + Ineq + Prob + Time,
    data = train_data)
```

```
Residuals:
    Min      1Q  Median      3Q     Max 
-393.89  -92.37  -18.49  129.66  565.05 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) -5343.858   1205.693  -4.432 0.000115 ***
M             104.572     40.033   2.612 0.013922 *  
Ed            214.457     57.723   3.715 0.000830 ***
Po1           106.462     17.007   6.260 6.76e-07 ***
U2             80.575     53.943   1.494 0.145695    
Ineq           68.054     16.851   4.039 0.000343 ***
Prob        -2217.777   2264.673  -0.979 0.335267    
Time            5.124      6.864   0.746 0.461205    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 211.4 on 30 degrees of freedom
Multiple R-squared:  0.7138,	Adjusted R-squared:  0.6471 
F-statistic: 10.69 on 7 and 30 DF,  p-value: 1.117e-06

Iteration #4

Call:
lm(formula = Crime ~ M + Ed + Po1 + U2 + Ineq + Prob + Time, 
    data = train_data)

Residuals:
    Min      1Q  Median      3Q     Max 
-542.08 -102.08    4.98  135.19  444.37 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) -4353.1006  1012.5067  -4.299 0.000167 ***
M              82.6074    43.6773   1.891 0.068273 .  
Ed            158.3024    55.2603   2.865 0.007552 ** 
Po1           127.6608    15.6173   8.174    4e-09 ***
U2             86.4822    44.3536   1.950 0.060596 .  
Ineq           65.0294    16.8641   3.856 0.000566 ***
Prob        -3945.8976  2022.6503  -1.951 0.060469 .  
Time            0.1747     6.1475   0.028 0.977520    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 199.3 on 30 degrees of freedom
Multiple R-squared:  0.8101,	Adjusted R-squared:  0.7658 
F-statistic: 18.28 on 7 and 30 DF,  p-value: 3.182e-09

Iteration #5

Call:
lm(formula = Crime ~ M + Ed + Po1 + U2 + Ineq + Prob + Time, 
    data = train_data)

Residuals:
    Min      1Q  Median      3Q     Max 
-498.35  -90.91  -10.76  153.29  548.25 
```

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) -4652.173   1259.826  -3.693 0.000915 ***
M              95.819     42.592   2.250 0.032229 *  
Ed            176.717     62.219   2.840 0.008159 ** 
Po1           119.557     16.805   7.114 7.91e-08 ***
U2             82.956     54.423   1.524 0.138272    
Ineq           72.333     16.831   4.298 0.000177 ***
Prob        -4765.887   2180.953  -2.185 0.037097 *  
Time           -3.731      6.958  -0.536 0.595933    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 218.3 on 29 degrees of freedom
Multiple R-squared:  0.7744,    Adjusted R-squared:   0.72 
F-statistic: 14.22 on 7 and 29 DF,  p-value: 7.421e-08

RMSE: 227.23
MAE: 169.714
```

The plot is in the appendix.

```
xyplot(resids ~ preds,
  xlab = "Fitted Values",
  ylab = "Residuals",
  main = "Residual Diagnostic Plot",
  panel = function(x, y, ...)
  {
    panel.grid(h = -1, v = -1)
    panel.abline(h = 0)
    panel.xyplot(x, y, ...)
  }
)
```

The second model with the subset of predictors worked the better than the model built on all the predictors. RMSE reduced from 289.558 to 227.23 and MAE reducted from 221.229 to 169.714.

So, now the final model using all available data will be built using the formula below:

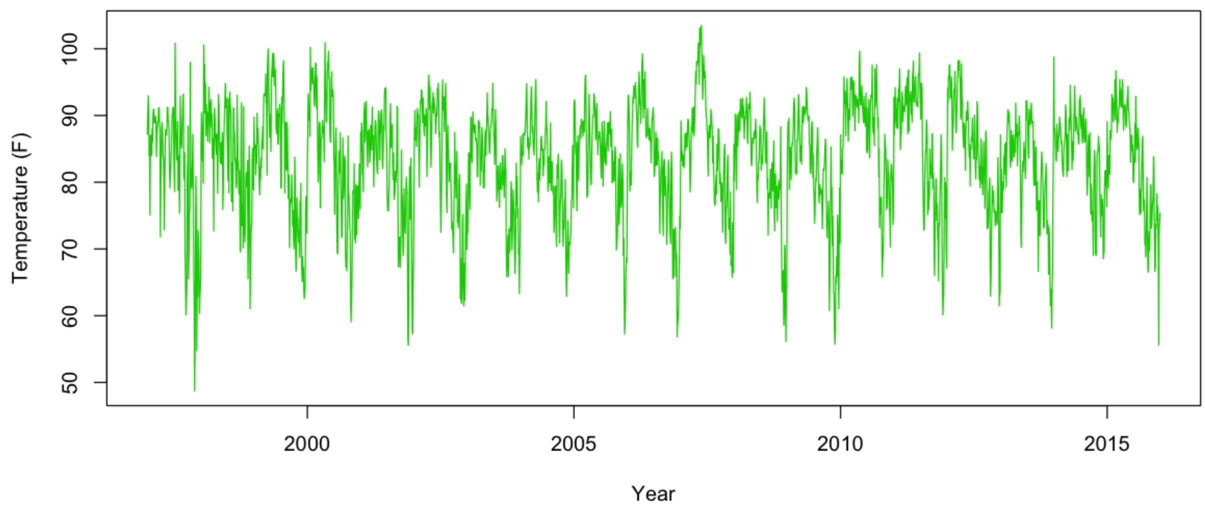
Crime ~ M + Ed + Po1 + U2 + Ineq + Prob + Time

```
final_model <- lm(formula=Crime ~ M + Ed + Po1 + U2 + Ineq + Prob + Time,
data=crime_data)

sample_prediction <- round(predict(final_model, sample_data), 3)

print(glue("Prediction for the crime rate for the sample data is
{sample_prediction}"))
print("Final Model")
print(summary(final_model))
```

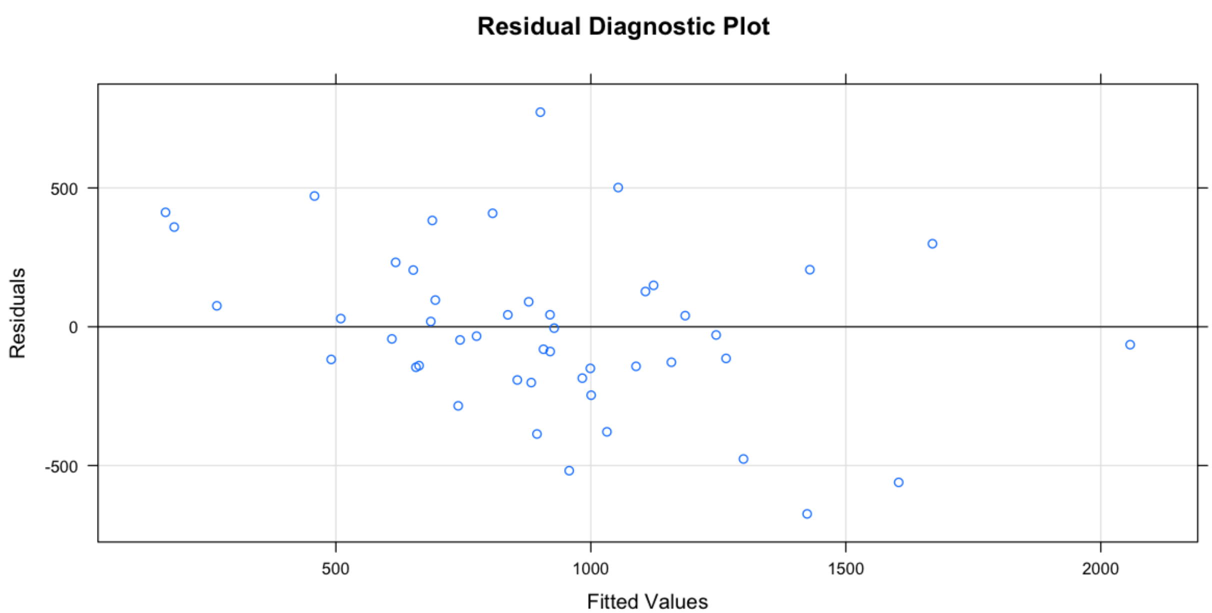
```
Prediction for the crime rate for the sample data is 1285.283
[1] "Final Model"

Call:
lm(formula = Crime ~ M + Ed + Po1 + U2 + Ineq + Prob + Time, 
    data = crime_data)

Residuals:
    Min      1Q  Median      3Q     Max 
-480.89  -89.12   -6.63  140.27  576.79 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) -4911.094    960.729  -5.112 8.79e-06 ***
M             106.659     33.877   3.148 0.003144 ** 
Ed            189.408     48.288   3.922 0.000345 ***
Po1           115.704     13.993   8.269 4.16e-10 ***
U2             88.720     41.364   2.145 0.038249 *  
Ineq           67.728     14.083   4.809 2.28e-05 ***
Prob        -4249.756   1880.672  -2.260 0.029502 *  
Time           -2.310      5.538  -0.417 0.678810    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 202.8 on 39 degrees of freedom
Multiple R-squared:  0.7669,	Adjusted R-squared:  0.7251 
F-statistic: 18.33 on 7 and 39 DF,  p-value: 1.553e-10
```

Appendix:

Holt-Winters plot



Linear regression model with every predictor



Linear regression model with M, Ed, Po1, U2, Ineq, Prob, and Time as the predictors