Economics 144 Project 1: Forecasting Number of Poice Calls in Eugene, Oregon

Jae Kum (Jackie) Kim, Edward(Jun Jie) Li, Luna Xu, and Austin Steinhart

1/31/2019

Contents

I. Introduction

The data we are analyzing is the daily number of police calls in Eugene, OR from 09/22/2008 up through approximately 2pm on 12/19/2018. The mean number of calls is 238.2, the maximum is 482 and the minimum is 3.

The number of calls per day had a clear downward trend and what appeared to be the season variation. From our previous knowledge of crime rates, we know that there are the seasonal differences in crimes rates (e.g. homicides spike in the summer) which we would expect would lead to seasonal differences in police calls, making the data ideal to analysis for this project. One caveat is that there is a break in the data in late 2013. Upon further research, we discovered that Eugene Police converted to a new format for categorizing and reporting a crime that is expected to be mandated by the federal government in the near future. Thus we expect this to have some effect on the number of reported police calls. This will be discussed further in Part III.

In order to clean the data, we first sequenced it by date using the "dplyr" package. We then removed the first and last data points (09/22/08 and 12/19/18) as they were both stated to be incomplete according to the owner of the data file. Next, we manipulated the data to produce weekly averages. We decided to use weekly averages rather than daily totals in order to determine a stronger seasonal trend as we would be creating 52 seasonal variable rather than 365. The mean, maximum, and minimum of the weekly averages are 238.3, 360, and 118.1, respectively.

II. Results

```r
# setup
rm(list = ls(all = TRUE))
library(readr)
library(lubridate)
library(dplyr)
library(stats)
library(forecast)
library(car)
library(MASS)
library(forecast)

library(xts)
library(zoo)
```

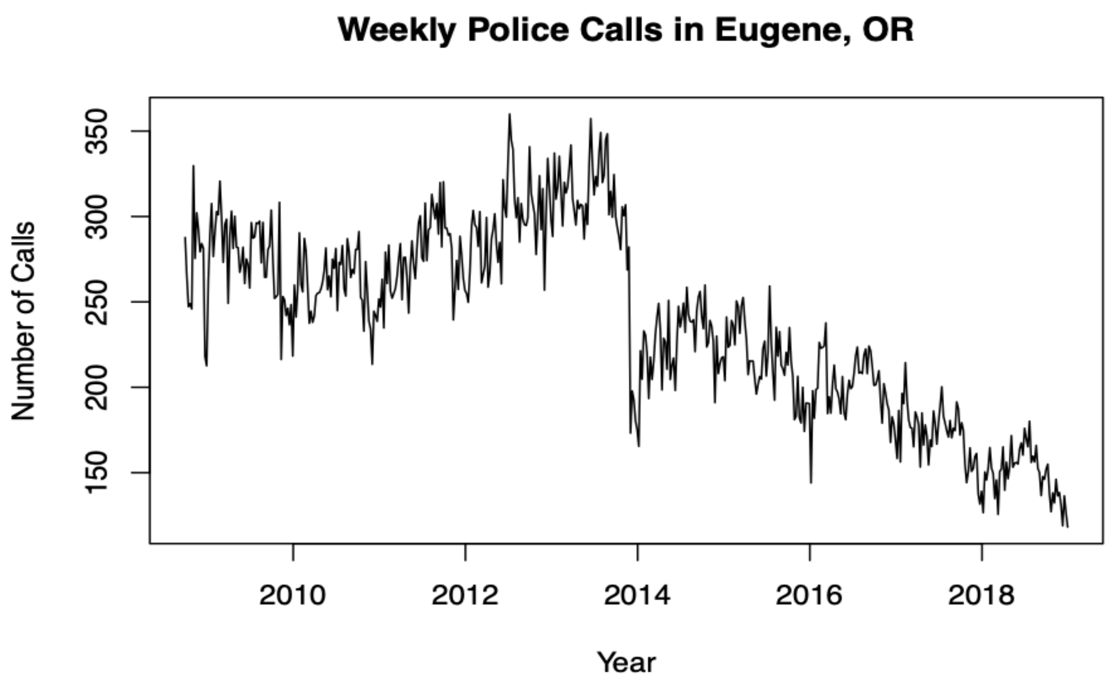
```r
require(stats)
library(foreign)
library(nlts)
```

```r
setwd("~/Downloads")
data <- read.table("Eugene_Police_Calls.csv", header = TRUE, sep = ",")
data = data %>% arrange(date)
date1 = seq(as.Date("2008-09-23"), as.Date("2018-12-18"), by = "day")
police = zoo(data$calls[(c(-1, -3740))], date1)
police_w = apply.weekly(police, mean)
date2 = seq(as.Date("2008-09-28"), as.Date("2018-12-16"), by = "week")
police_w = zoo(police_w, date2)
```

1. Modeling and Forecasting Trend

1a. Time-Series Plot

```r
police_ts = ts(police_w, start = 2008 + 38.7143/52, frequency = 52)
t<-seq(2008 + 38.7143/52, 2018 + 49/52, length=length(police_ts))
plot(police_ts, type ="l", main = 'Weekly Police Calls in Eugene, OR',
     xlab = 'Year',
     ylab = 'Number of Calls' )
```

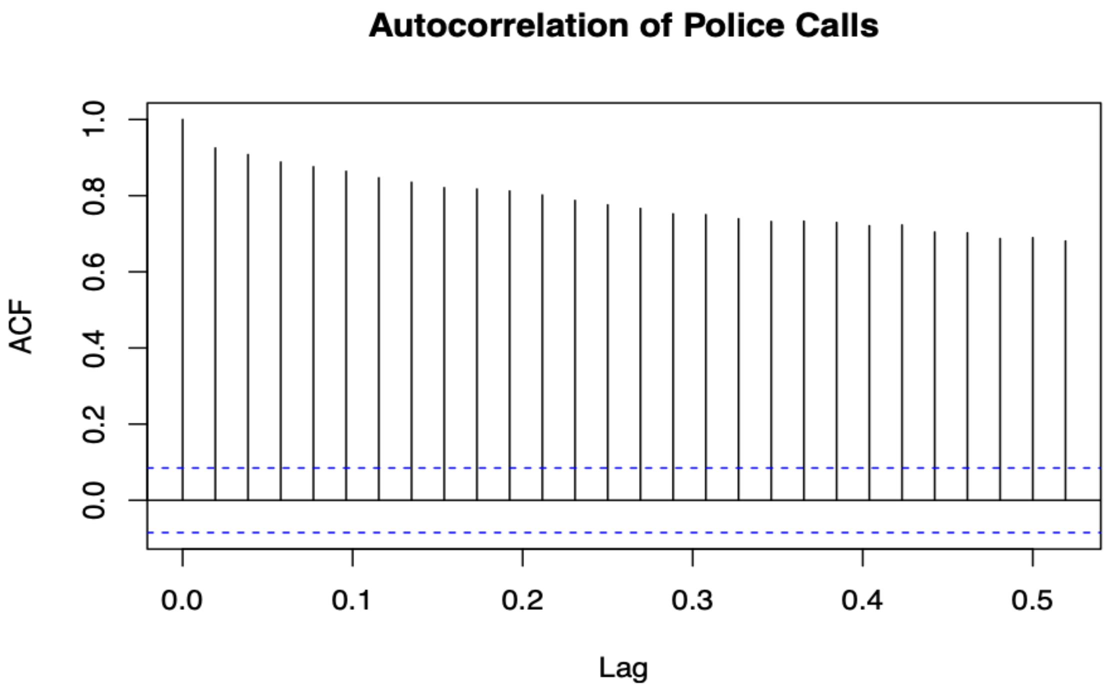


1b. Covariance Stationary

The data does not appear to be covariance stationary. It has a structural break during 2013 due to a Federal policy change regarding police reports. In addition, there is clearly a downward trend since 2014.

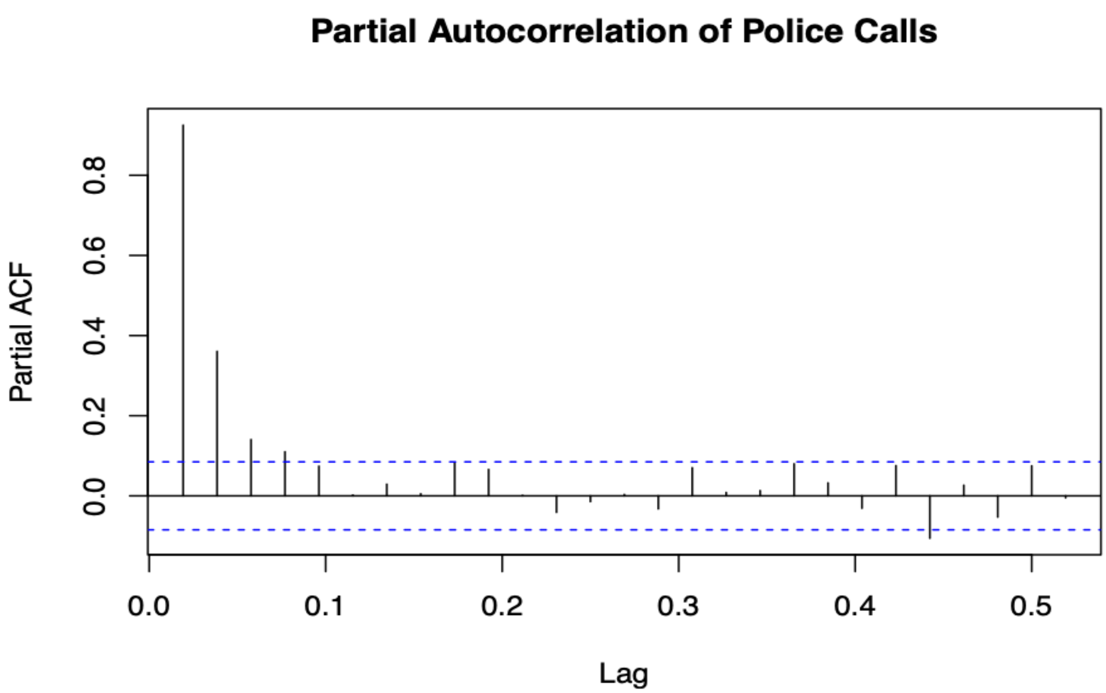
1c. ACF and PACF Plots

```r
acf(police_ts, main = 'Autocorrelation of Police Calls')
```



The plot shows a slowly decaying ACF, but it is far from decaying to zero. This confirms that our data is not covariance stationary.

```
pacf(police_ts, main = 'Partial Autocorrelation of Police Calls')
```



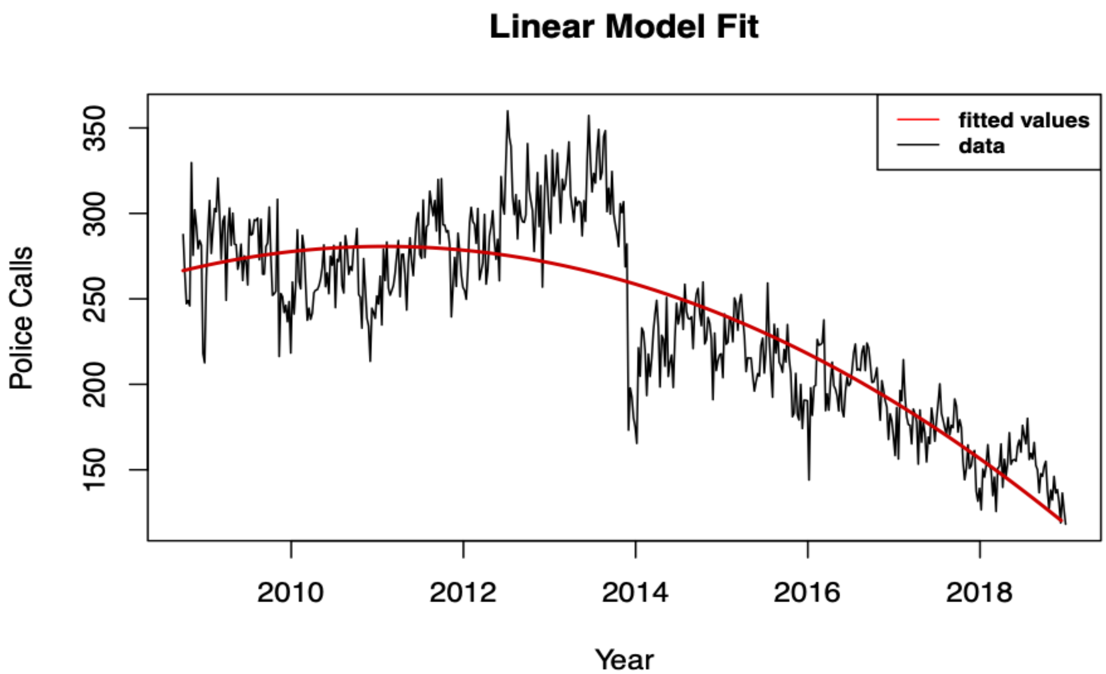
PACF shows significant spikes at the first 3 to 4 lags and quickly decays to zero. This may suggest an autoregressive process.

1d. Linear and Nonlinear Models

```
# Linear (Quadratic)
y1=tslm(police_ts~t+I(t^2))
```

Linear Fit Plot

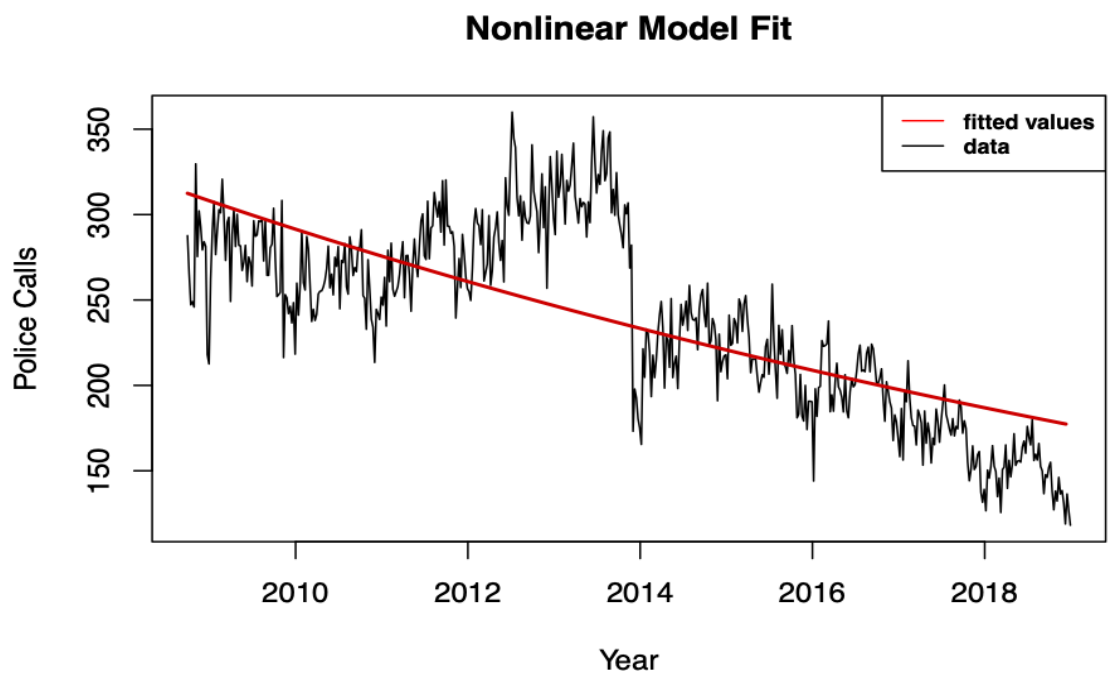
```
plot(police_ts, main = "Linear Model Fit", xlab = "Year", ylab = "Police Calls")
lines(t, y1$fit, col = "red3", lwd = 2)
legend(x = "topright", legend = c("fitted values", "data"), col = c("red", "black"),
    lty = c(1, 1), cex = 0.75, text.font = 2)
```




```
# Nonlinear (exponential)
ds = data.frame(x = t, y = police_ts)
y2 = nls(y ~ exp(a + b * t), data = ds, start = list(a = 0, b = 0))
```

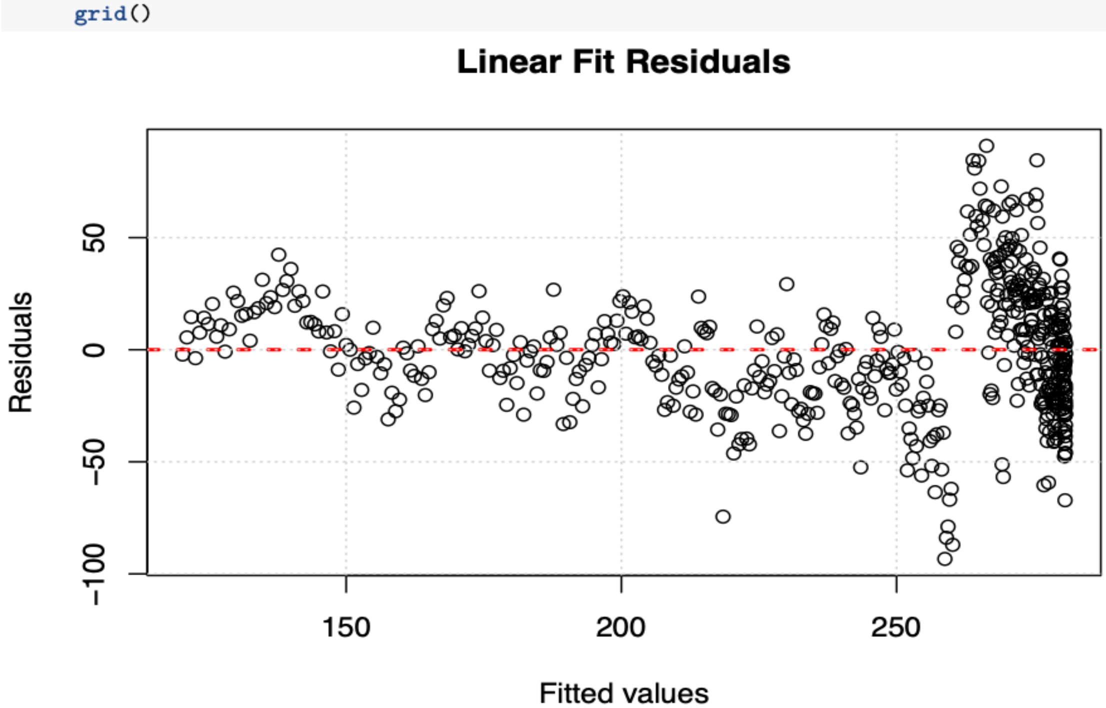
Nonlinear Fit Plot

```
plot(police_ts, main = "Nonlinear Model Fit", xlab = "Year", ylab = "Police Calls")
lines(t, fitted(y2), col = "red3", lwd = 2)
legend(x = "topright", legend = c("fitted values", "data"), col = c("red", "black"),
    lty = c(1, 1), cex = 0.75, text.font = 2)
```



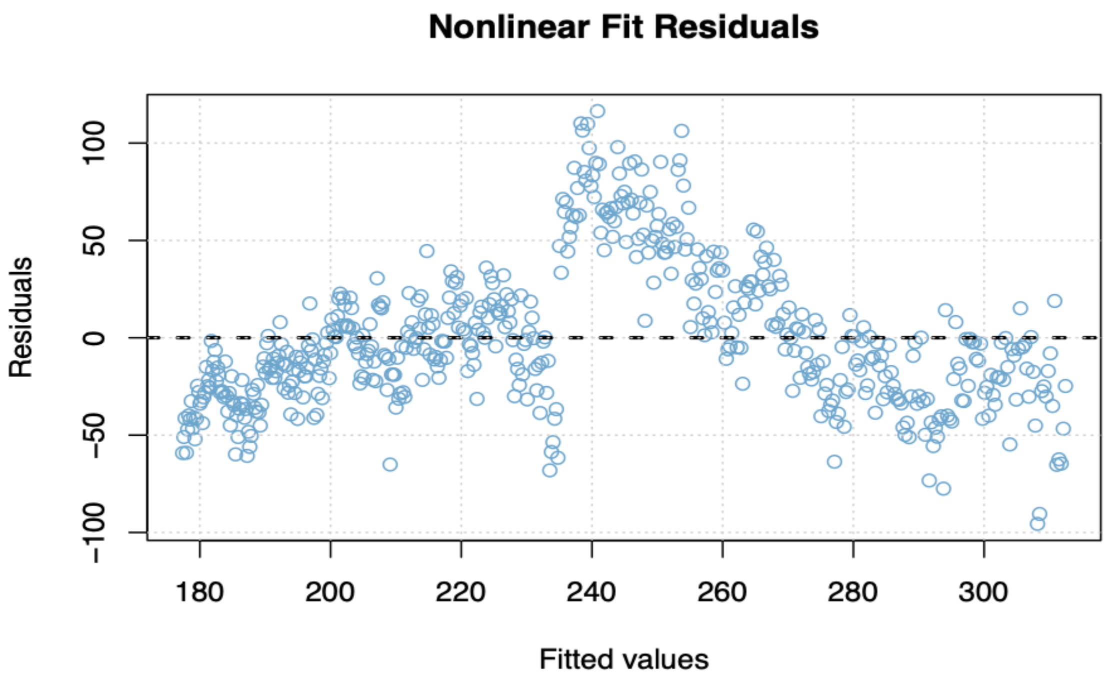
1e. Residuals

```
plot(y1$fit, y1$res, main = "Linear Fit Residuals",
     ylab="Residuals", xlab="Fitted values")
     abline(h = 0, lty = 2, col = 'red', lwd = 2)
     grid()
```



High residuals for the linear fit seems to be clustered around fitted values that are greater than 250. This indicates that in the periods of 2012 to 2014 of the data, when police calls demonstrated a sudden increase and decrease, the model is an unperfect fitt.

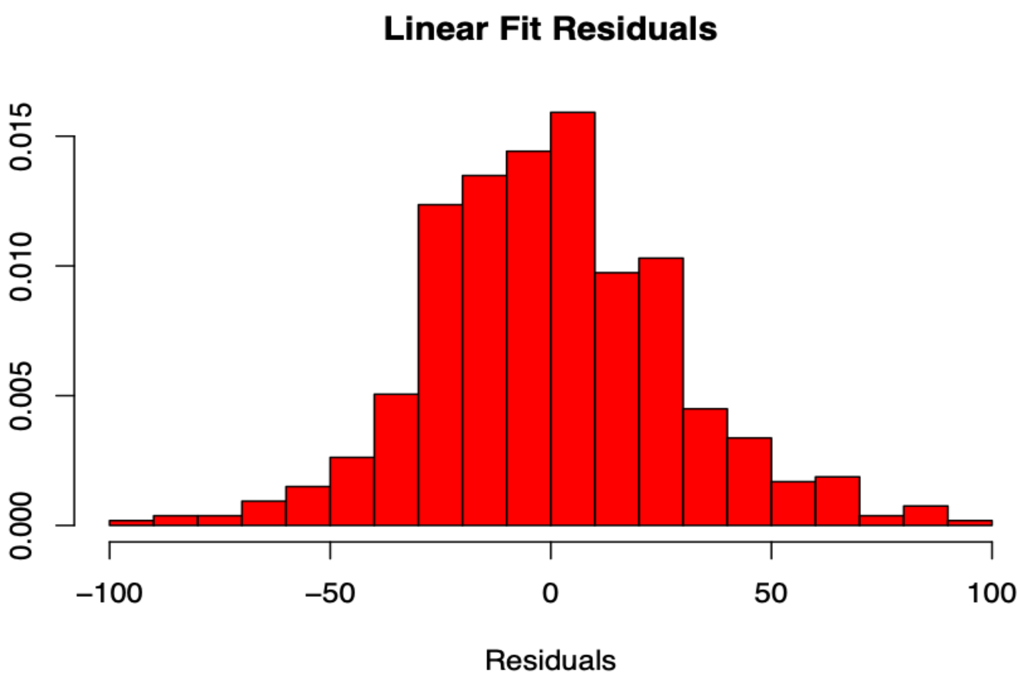
```
plot(fitted(y2),residuals(y2), main = 'Nonlinear Fit Residuals',
     ylab="Residuals", xlab="Fitted values",
     col = 'skyblue3')
abline(h = 0, lty = 2, col = 'black', lwd = 2)
grid()
```



For the nonlinear fit, the residuals are mostly positive in the cluster between 2012 and 2014. After 2014, the residuals go into the negative. It seems that due to the sudden increase in calls between 2012 and 2014 and the sudden drop at the end of 2014, the model is unable to fit well. The nonlinear fit plot showcases a clear pattern in trend.

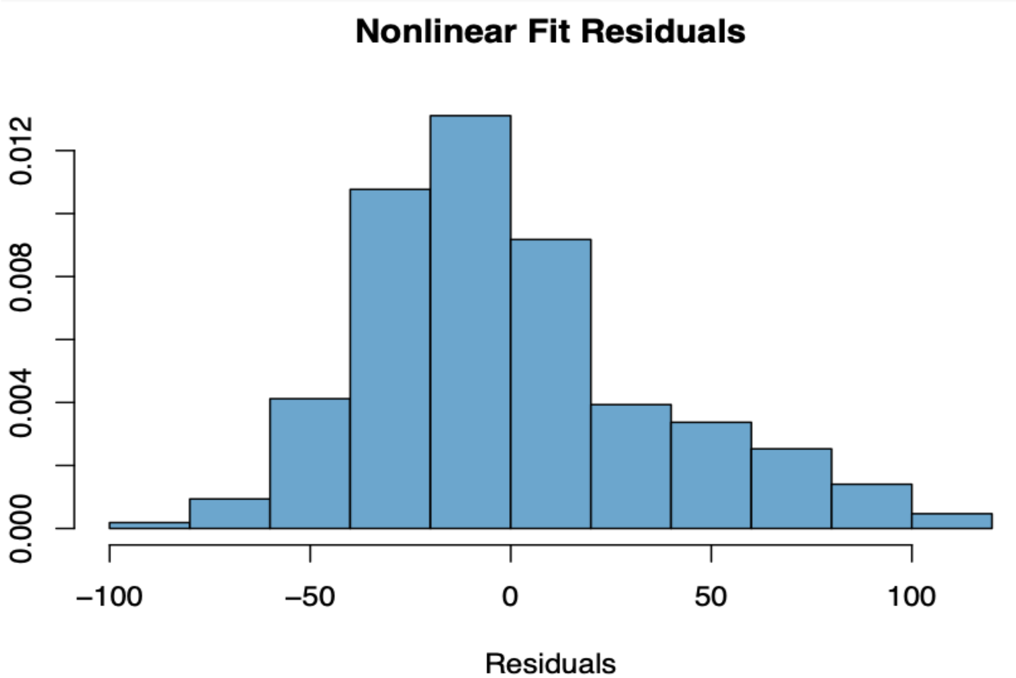
1f. Histogram of Residuals

```
truehist(residuals(y1),
         main = 'Linear Fit Residuals',
         xlab = 'Residuals',
         col = 'red')
```



The histogram of residuals for the linear fit model is mostly normal with a mean at 0. Skewness is also minimal. This satisfies the residuals requirement for linear regression.

```
truehist(residuals(y2),
         main = 'Nonlinear Fit Residuals',
         xlab = 'Residuals',
         col = 'skyblue3')
```



The histogram for nonlinear fit residuals is right skewed and the mean is a little bit less than zero. Its normality is definitely less convincing than that of the linear fit model.

1g. Diagnostic

```r
# Linear Fit
summary(y1)
```

```
##
## Call:
## tslm(formula = police_ts ~ t + I(t^2))
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -93.33 -19.14  -1.15  17.04  90.95
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.050e+07  6.426e+05  -16.33   <2e-16 ***
## t            1.044e+04  6.382e+02   16.36   <2e-16 ***
## I(t^2)      -2.595e+00  1.584e-01  -16.38   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 28.49 on 531 degrees of freedom
## Multiple R-squared:  0.7313, Adjusted R-squared:  0.7303
## F-statistic: 722.5 on 2 and 531 DF,  p-value: < 2.2e-16
```

Linear model y1 produces an adjusted R^2 of 0.7303, F statistic of 722.54, a p-value that is practically zero. Given these statistics, this model is highly significant. For the most part, the regression line reflects the overall trend of the data.

```r
# Nonlinear Fit
summary(y2)
```

```
##
## Formula: y ~ exp(a + b * t)
##
## Parameters:
##     Estimate Std. Error t value Pr(>|t|)
## a 117.279127   4.642261   25.26   <2e-16 ***
## b  -0.055525   0.002306  -24.08   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 36.82 on 532 degrees of freedom
##
## Number of iterations to convergence: 10
## Achieved convergence tolerance: 4.284e-06
```

The non-linear model does not provide R^2 or F-statistic as they are only calculated for linear regression models. However, the residuals standard error is 36.82. Residual standard error for the non-linear model is greater than the same error for the linear model, which suggests that the sum errors of the non-linear model is greater. Visually speaking, the non-linear model does look like a worse fit. Low p values for both predictors show high model significance.

1h. Model Selection

```r
AIC(y1,y2)
```

```
##    df      AIC
## y1  4 5097.613
## y2  3 5370.580
```

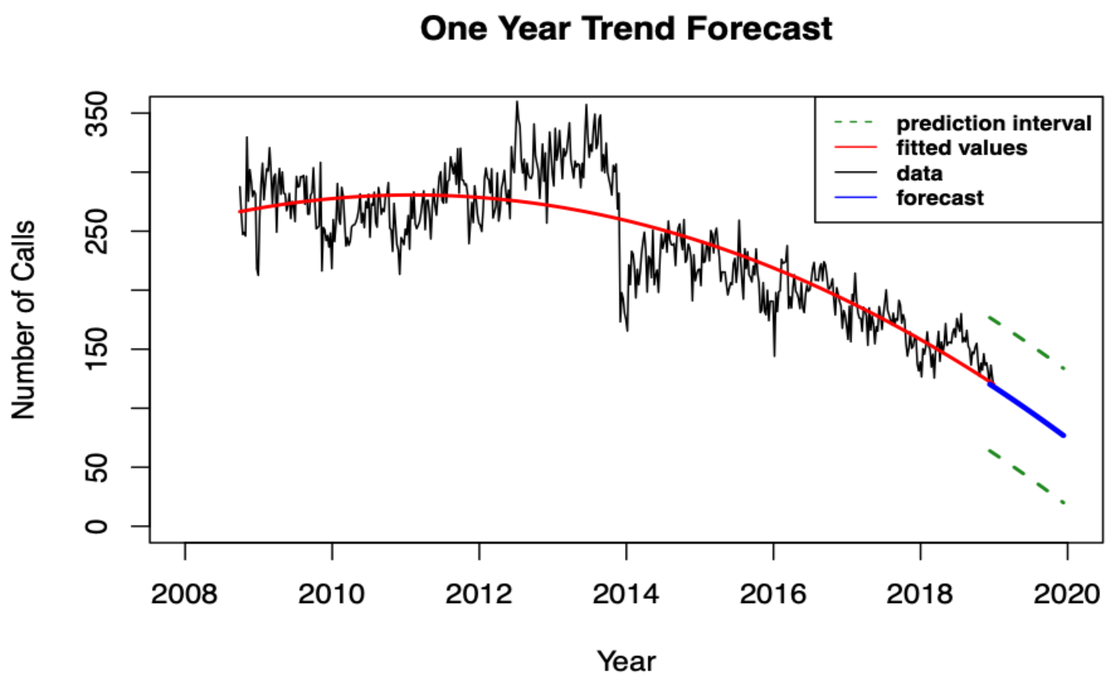
```r
BIC(y1,y2)
```

```
##    df      BIC
## y1  4 5114.734
## y2  3 5383.422
```

The models both agree on y1. We will be selecting y1.

1i. Forecasting One Year Ahead

```r
t_pred <- seq(t[length(t)], t[length(t)] + 1, length = 52)
tn = data.frame(t = t_pred)
pred = predict(tslm(police_ts ~ t + I(t^2)), tn, interval = "predict")
plot(police_ts, main = "One Year Trend Forecast", xlab = "Year", ylab = "Number of Calls",
    ylim = c(0, 350), xlim = c(2008, 2020))
lines(y1$fit, col = "red", lwd = 2)
lines(t_pred, pred[, 1], col = "blue", lwd = 3)
lines(t_pred, pred[, 2], col = "forestgreen", lwd = 2, lty = 2)
lines(t_pred, pred[, 3], col = "forestgreen", lwd = 2, lty = 2)
legend(x = "topright", legend = c("prediction interval", "fitted values", "data",
    "forecast"), col = c("forestgreen", "red", "black", "blue"), lty = c(2,
    1, 1, 1), cex = 0.75, text.font = 2)
```



2. Modeling and Forecasting Seasonality

2a. Seasonal Model

```r
y3<- tslm(police_ts ~ season)
summary(y3)
```

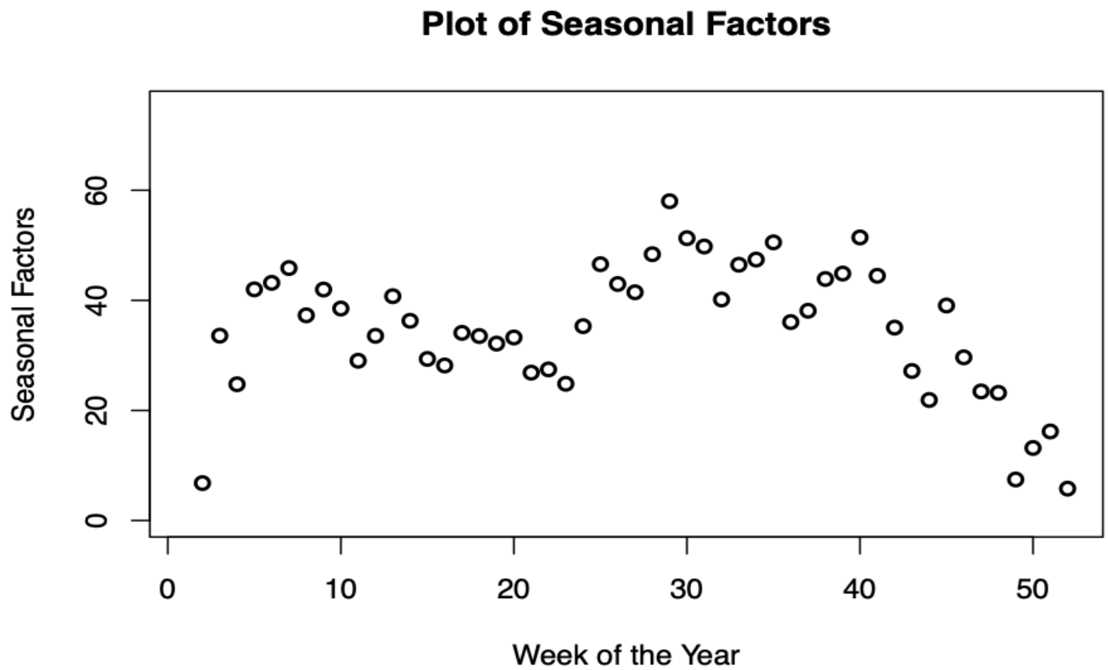
```
##
## Call:
## tslm(formula = police_ts ~ season)
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -111.400  -43.183    5.793   44.469  113.532
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  204.286     16.909  12.081   <2e-16 ***
## season2        6.800     24.504   0.278   0.7815
## season3       33.586     24.504   1.371   0.1711
## season4       24.757     24.504   1.010   0.3128
## season5       42.000     24.504   1.714   0.0872 .
## season6       43.186     24.504   1.762   0.0786 .
## season7       45.871     24.504   1.872   0.0618 .
## season8       37.286     24.504   1.522   0.1288
## season9       41.971     24.504   1.713   0.0874 .
## season10      38.514     24.504   1.572   0.1167
## season11      29.029     24.504   1.185   0.2367
## season12      33.557     24.504   1.369   0.1715
## season13      40.757     24.504   1.663   0.0969 .
## season14      36.286     24.504   1.481   0.1393
## season15      29.357     24.504   1.198   0.2315
## season16      28.171     24.504   1.150   0.2509
## season17      34.100     24.504   1.392   0.1647
## season18      33.514     24.504   1.368   0.1720
## season19      32.143     24.504   1.312   0.1902
## season20      33.257     24.504   1.357   0.1754
## season21      26.871     24.504   1.097   0.2734
## season22      27.457     24.504   1.121   0.2631
## season23      24.843     24.504   1.014   0.3112
## season24      35.314     24.504   1.441   0.1502
## season25      46.571     24.504   1.901   0.0580 .
## season26      42.971     24.504   1.754   0.0801 .
## season27      41.471     24.504   1.692   0.0912 .
## season28      48.386     24.504   1.975   0.0489 *
## season29      58.014     24.504   2.368   0.0183 *
## season30      51.300     24.504   2.094   0.0368 *
## season31      49.786     24.504   2.032   0.0427 *
## season32      40.157     24.504   1.639   0.1019
## season33      46.471     24.504   1.896   0.0585 .
## season34      47.414     24.504   1.935   0.0536 .
## season35      50.543     24.504   2.063   0.0397 *
## season36      36.057     24.504   1.471   0.1418
## season37      38.114     24.504   1.555   0.1205
## season38      43.886     24.504   1.791   0.0739 .
## season39      44.871     24.504   1.831   0.0677 .
## season40      51.437     23.914   2.151   0.0320 *
## season41      44.468     23.914   1.860   0.0636 .
## season42      35.052     23.914   1.466   0.1434
## season43      27.169     23.914   1.136   0.2565
## season44      21.896     23.914   0.916   0.3603
## season45      39.065     23.914   1.634   0.1030
## season46      29.649     23.914   1.240   0.2156
## season47      23.455     23.914   0.981   0.3272
## season48      23.195     23.914   0.970   0.3326
## season49       7.455     23.914   0.312   0.7554
## season50      13.182     23.914   0.551   0.5817
## season51      16.182     23.914   0.677   0.4989
## season52       5.792     23.914   0.242   0.8087
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 56.08 on 482 degrees of freedom
## Multiple R-squared:  0.05458,	Adjusted R-squared:  -0.04546
## F-statistic: 0.5456 on 51 and 482 DF,  p-value: 0.9957
```

2b. Plotting Seasonal Factors

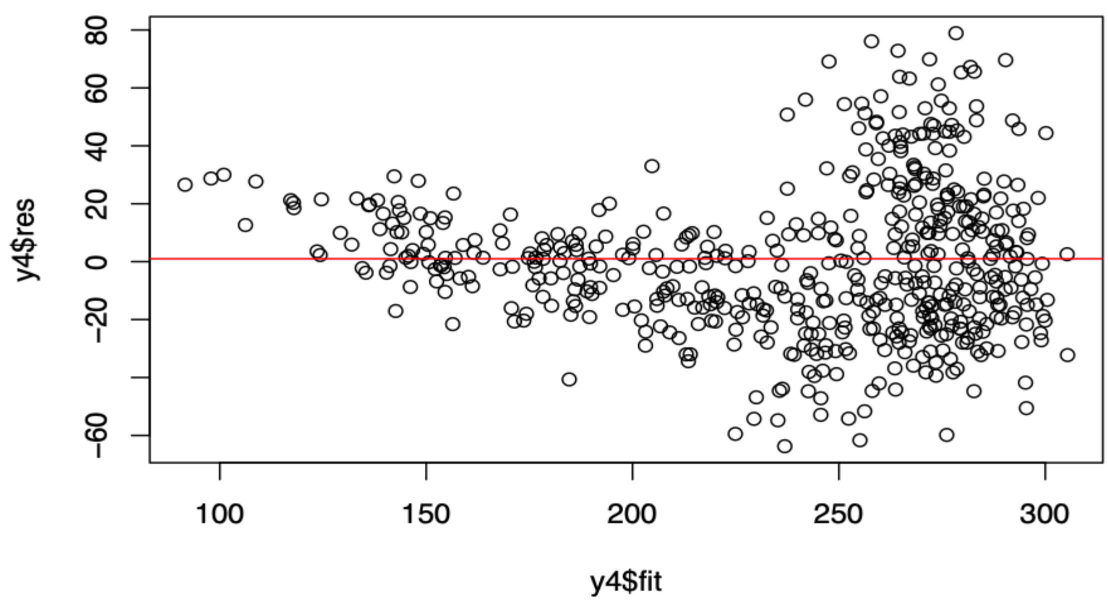
```r
plot(y3$coef, ylab = "Seasonal Factors", xlab = "Week of the Year", lwd = 2,
    main = "Plot of Seasonal Factors", ylim = c(0, 75))
```



Several of the seasonal factors are indeed significant at either the 0.05 or 0.01 level. The plot demonstrates a few week trend when the number of police calls go from highest to lowest. One interesting note: At around the 30th week of the year which would correspond the the beginning of the spring, there is spike in police calls. The opposite analysis goes for fall when the number of police calls drop. This may be able to be explained by the rising crime rate in the spring and summer due to increasing temperatures while the decrease can be explained by the lowering temperatures in the fall.

2c. Trend + Seasonal Model

```r
y4 <- tslm(police_ts~poly(trend,2) + season)
plot(y4$fit, y4$res)
abline(1,0, col ='red')
```



Again, the residual plot exhibits higher variance at higher fit values. The heteroskedasticity is likely caused by the lack of fitness between year 2012 and 2014. Compared to the residuals vs. fitted plot in part 1, the overall magnitudes of the residuals are smaller here, suggesting a better fit.

2d. Summary Statistics

```
summary(y4)
```

```
##
## Call:
## tslm(formula = police_ts ~ poly(trend, 2) + season)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -63.742 -17.189  -1.575  14.683  78.912
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      209.168      8.078  25.892  < 2e-16 ***
## poly(trend, 2)1 -984.245     26.837 -36.675  < 2e-16 ***
## poly(trend, 2)2 -456.716     26.917 -16.967  < 2e-16 ***
## season2           -4.229     11.708  -0.361 0.718107
## season3           22.800     11.708   1.947 0.052065 .
## season4           14.216     11.707   1.214 0.225248
## season5           31.705     11.707   2.708 0.007008 **
## season6           33.139     11.707   2.831 0.004840 **
## season7           36.075     11.707   3.081 0.002178 **
## season8           27.742     11.707   2.370 0.018200 *
## season9           32.681     11.707   2.792 0.005454 **
## season10          29.480     11.707   2.518 0.012122 *
## season11          20.252     11.707   1.730 0.084288 .
## season12          25.040     11.707   2.139 0.032946 *
## season13          32.502     11.707   2.776 0.005713 **
## season14          28.294     11.707   2.417 0.016028 *
## season15          21.630     11.707   1.848 0.065268 .
## season16          20.711     11.707   1.769 0.077499 .
## season17          26.909     11.707   2.299 0.021958 *
```

```
## season18          26.594     11.707   2.272 0.023548 *
## season19          25.495     11.707   2.178 0.029906 *
## season20          26.884     11.707   2.296 0.022080 *
## season21          20.774     11.707   1.775 0.076601 .
## season22          21.638     11.707   1.848 0.065162 .
## season23          19.304     11.707   1.649 0.099805 .
## season24          30.057     11.707   2.568 0.010543 *
## season25          41.598     11.706   3.553 0.000418 ***
## season26          38.284     11.706   3.270 0.001152 **
## season27          37.071     11.706   3.167 0.001640 **
## season28          44.275     11.706   3.782 0.000175 ***
## season29          54.195     11.706   4.629 4.73e-06 ***
## season30          47.773     11.706   4.081 5.25e-05 ***
## season31          46.554     11.706   3.977 8.06e-05 ***
## season32          37.222     11.706   3.180 0.001570 **
## season33          43.835     11.706   3.745 0.000203 ***
## season34          45.078     11.706   3.851 0.000134 ***
## season35          48.509     11.706   4.144 4.04e-05 ***
## season36          34.328     11.706   2.932 0.003524 **
## season37          36.691     11.706   3.134 0.001828 **
## season38          42.770     11.706   3.654 0.000287 ***
## season39          44.066     11.706   3.764 0.000188 ***
## season40          47.845     11.422   4.189 3.34e-05 ***
## season41          41.141     11.422   3.602 0.000349 ***
## season42          31.992     11.422   2.801 0.005301 **
## season43          24.378     11.422   2.134 0.033321 *
## season44          19.376     11.422   1.696 0.090457 .
## season45          36.817     11.422   3.223 0.001353 **
## season46          27.676     11.422   2.423 0.015756 *
## season47          21.758     11.422   1.905 0.057383 .
## season48          21.776     11.422   1.907 0.057173 .
## season49           6.316     11.422   0.553 0.580537
## season50          12.325     11.422   1.079 0.281084
## season51          15.609     11.422   1.367 0.172388
## season52           5.505     11.422   0.482 0.630052
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26.79 on 480 degrees of freedom
## Multiple R-squared:  0.7852, Adjusted R-squared:  0.7615
## F-statistic: 33.11 on 53 and 480 DF,  p-value: < 2.2e-16
```

```r
accuracy(y4)
```

```
##               ME     RMSE      MAE        MPE     MAPE      MASE
## Training set  0 25.39543 19.83021 -0.8992238 8.283254 0.4254125
```

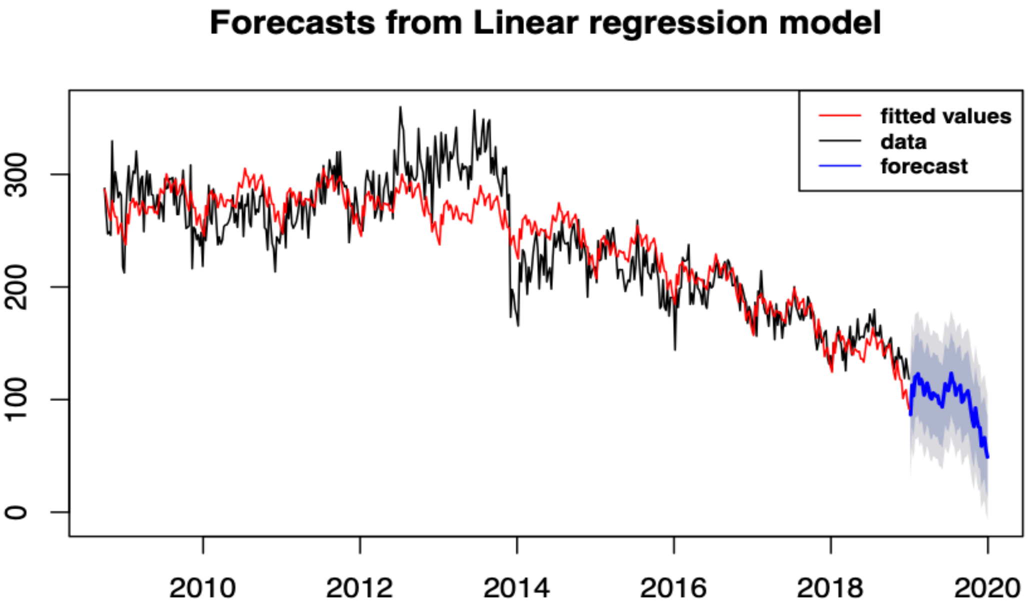
This model has a high R^2 of 0.7852. A highly significant F-stat of 33.11 and a p-value that is approximately 0. Most of the seasonal variables are highly significant to the 0.001 level. The peak of police calls in terms of season is week 29, which is around the beginning of the spring. This is probably due to increased outdoor activities due to increased temperatures leading to more police activity.

Looking at the residuals, it seems residuals mostly cluster at the higher call values of >250. This indicates a similar analysis to part I of the project where our linear and non-linear models have a hard time fitting the high call volumes between 2012 and 2014 and the sudden structural break before 2014.

MAE is around 19.78, showing that the model has a problem in certain areas. While there may be a possibility of outliers, we already removed possible outliers in Part 1. This means the Break in 2013-2014 is contributing some factors of high MAE.

2e. Forecasting One Year Ahead

```
y4 <- tslm(police_ts ~ poly(trend, 2) + season)
plot(forecast(y4, h = 52))
lines(y4$fit, col = "red")
legend(x = "topright", legend = c("fitted values", "data", "forecast"), col = c("red",
    "black", "blue"), lty = c(1, 1, 1), cex = 0.75, text.font = 2)
```



III. Conclusion

Our final model y4 successfully incorporates trend and seasonal factors that underpins our data. For most of the data, we observe a downward trend and a five-week seasonal cycles with spike in Summer as well as a dip in winter. y4 follows the trend adequately for the periods before 2013 and after 2014. Predicting one year ahead, we expect to see a continual drop in police calls while retaining a seasonal cycle. The downward trend can be most likely explained by high-quality police work and better economic conditions. However, if the model continues on its current path, it will hit a level of no police calls before the end of 2021. This certainly will not be true. Due to the structural break, in order to improve our model, we should create two separate models, one before the break and one after the break. Due to the break, our current model have a strong downward trend as discussed above. If we instead created two models we would see a much less dramatic trend for the second half of the data.

Regarding the structural break, after a brief news search we discovered that the Eugene Police converted to a new format for categorizing and reporting crime. The switch was made from the Uniform Crime Reporting (UCR) format to Oregon National Incident Based Reporting System (NIBRS). The reporting systems use divergent rules that if compared might result in inaccurate conclusions about crime rate changes. Under UCR, the top most serious crime is the one the agency reports (with a couple exceptions), and with NIBRS, all of an incident's crimes (up to a total of 22) are reported. Thus, due to the differences in the way that crime was reported, we suspect that the data regarding police calls may also be affected.

IV. References

Data is extracted from Kaggle. Click on the right to go to the link: (https://www.kaggle.com/warrenhendricks/police-call-data-for-eugene-and-springfield-or)