Crimes Time Series

Kristin Fesmire

2023-08-25

```
rm(list = ls())

library(stringr)
library(ggplot2)
library(forecast)
library(reshape2)

df <- read.csv("C:/Users/krtfe/Downloads/Crimes_-_2023 (ret. 082023).csv")

cat('Pre-cleaning summary:\n\n')
```

```
## Pre-cleaning summary:
```

```
df %>% summary %>% print
```

```
##        ID           Case.Number           Date              Block
##  Min.   :   27279   Length:150712      Length:150712      Length:150712
##  1st Qu.:12997210   Class :character   Class :character   Class :character
##  Median :13054382   Mode  :character   Mode  :character   Mode  :character
##  Mean   :13021795
##  3rd Qu.:13111126
##  Max.   :13171025
##
##      IUCR           Primary.Type       Description        Location.Description
##  Length:150712      Length:150712      Length:150712      Length:150712
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##    Arrest            Domestic             Beat         District
##  Length:150712      Length:150712      Min.   : 111   Min.   : 1.00
##  Class :character   Class :character   1st Qu.: 532   1st Qu.: 5.00
##  Mode  :character   Mode  :character   Median :1031   Median :10.00
##                                        Mean   :1148   Mean   :11.25
##                                        3rd Qu.:1724   3rd Qu.:17.00
##                                        Max.   :2535   Max.   :31.00
##
##       Ward       Community.Area    FBI.Code          X.Coordinate
##  Min.   : 1.00   Min.   : 1.00   Length:150712      Min.   :1091242
```

```
##  1st Qu.: 9.00   1st Qu.:22.00   Class :character   1st Qu.:1153761
##  Median :23.00   Median :32.00   Mode  :character   Median :1167054
##  Mean   :23.03   Mean   :36.53                      Mean   :1165353
##  3rd Qu.:34.00   3rd Qu.:53.00                      3rd Qu.:1176910
##  Max.   :50.00   Max.   :77.00                      Max.   :1205114
##  NA's   :3                                          NA's   :7489
##   Y.Coordinate          Year       Updated.On          Latitude
##  Min.   :1813897   Min.   :2023   Length:150712      Min.   :41.65
##  1st Qu.:1859491   1st Qu.:2023   Class :character   1st Qu.:41.77
##  Median :1892280   Median :2023   Mode  :character   Median :41.86
##  Mean   :1886838   Mean   :2023                      Mean   :41.84
##  3rd Qu.:1910139   3rd Qu.:2023                      3rd Qu.:41.91
##  Max.   :1951503   Max.   :2023                      Max.   :42.02
##  NA's   :7489                                        NA's   :7489
##    Longitude        Location
##  Min.   :-87.94   Length:150712
##  1st Qu.:-87.71   Class :character
##  Median :-87.66   Mode  :character
##  Mean   :-87.67
##  3rd Qu.:-87.63
##  Max.   :-87.53
##  NA's   :7489
```

```r
# remove duplicate rows, removed 0 rows
df <- dplyr::distinct(df)

# remove duplicate rows by case number, removed 12 rows, from 150712 to 150700
df <- df[!duplicated(df$Case.Number),]

# simplify data, remove columns that aren't useful for current project
df <- df[c(3, 6, 7:10, 12:14)]

# remove rows with NA values, removed 3 rows, 150700 rows -> 150679 rows
df <- na.omit(df)

# add the useful columns and transform data types
df['Time'] <- str_split(df$Date,
                        pattern = ' ',
                        simplify = TRUE)[,2]
df['Time of Day'] <- str_split(df$Date,
                               pattern = ' ',
                               simplify = TRUE)[,3]
df['Date'] <- str_split(df$Date,
                        pattern = ' ',
                        simplify = TRUE)[,1] %>%
              as.Date(format = '%m/%d/%Y')

# set dataframe such that it only includes months from january to july
# from 150679 rows -> 147596
df <- df[df$Date < lubridate::ymd("2023-08-01"),]

cat('\n\nPost-cleaning summary:\n\n')
```

```
##
```

```
##
## Post-cleaning summary:
```

```
df %>% summary %>% print
```

```
##       Date             Primary.Type       Description
##  Min.   :2023-01-01   Length:147596      Length:147596
##  1st Qu.:2023-02-25   Class :character   Class :character
##  Median :2023-04-21   Mode  :character   Mode  :character
##  Mean   :2023-04-19
##  3rd Qu.:2023-06-12
##  Max.   :2023-07-31
##  Location.Description    Arrest            Domestic            District
##  Length:147596        Length:147596      Length:147596      Min.   : 1.00
##  Class :character     Class :character   Class :character   1st Qu.: 5.00
##  Mode  :character     Mode  :character   Mode  :character   Median :10.00
##                                                             Mean   :11.25
##                                                             3rd Qu.:17.00
##                                                             Max.   :31.00
##       Ward       Community.Area      Time           Time of Day
##  Min.   : 1.00   Min.   : 1.00   Length:147596      Length:147596
##  1st Qu.: 9.00   1st Qu.:22.00   Class :character   Class :character
##  Median :23.00   Median :32.00   Mode  :character   Mode  :character
##  Mean   :23.03   Mean   :36.54
##  3rd Qu.:34.00   3rd Qu.:53.00
##  Max.   :50.00   Max.   :77.00
```

```
# data frame representation
df %>% head
```

```
##         Date    Primary.Type         Description Location.Description Arrest
## 1 2023-06-28        HOMICIDE FIRST DEGREE MURDER                ALLEY   true
## 2 2023-06-29        HOMICIDE FIRST DEGREE MURDER               STREET  false
## 3 2023-03-30 CRIMINAL DAMAGE         TO PROPERTY          GAS STATION  false
## 4 2023-03-07           THEFT       FROM BUILDING            RESIDENCE  false
## 5 2023-06-29        HOMICIDE FIRST DEGREE MURDER               STREET  false
## 6 2023-06-29        HOMICIDE FIRST DEGREE MURDER               STREET  false
##   Domestic District Ward Community.Area     Time Time of Day
## 1    false       17   33             16 11:04:00          PM
## 2    false        7    6             68 07:40:00          PM
## 3    false        1    4             32 02:16:00          PM
## 4    false        3   20             42 10:57:00          AM
## 5    false        8   14             57 07:00:00          AM
## 6    false        7   16             67 04:39:00          PM
```

Separate data frame for (specific variable) counts by dates, from 2001 to 2022:

```r
# Crimes from 2001 to 2023 (Jan-July)
dfTotal <- read.csv("C:/Users/krtfe/Downloads/Crimes_-_2001_to_Present.csv")

# adding useful columns, dates, times, time of day
dfTotal['Date'] <- substr(dfTotal$Date,
                          1,
                          10)

# transform data types
dfTotal['Date'] <- as.Date(dfTotal$Date,
                           format = '%m/%d/%Y')

# set dataframe such that it only includes months from january to july
# from 150679 rows -> 147596
dfTotal <- dfTotal[dfTotal$Date < lubridate::ymd("2023-01-01"),] %>%
    sort

# data frame representation
dfTotal %>% head
```

```
## [1] "2001-01-01" "2001-01-01" "2001-01-01" "2001-01-01" "2001-01-01"
## [6] "2001-01-01"
```

```r
dfTotal %>% tail
```

```
## [1] "2022-12-31" "2022-12-31" "2022-12-31" "2022-12-31" "2022-12-31"
## [6] "2022-12-31"
```

Frequency table for total crimes in 2023:

```r
# second data frame, number of crimes

# start with the unique dates and their counts
counts2023 <- table(df$Date) %>%
              data.frame
colnames(counts2023) <- c('Date', 'Number of Crimes')

# printing the counts dataset
counts2023 %>% head
```

```
##         Date Number of Crimes
## 1 2023-01-01              970
## 2 2023-01-02              649
## 3 2023-01-03              733
## 4 2023-01-04              680
## 5 2023-01-05              654
## 6 2023-01-06              722
```

Frequency table for total crimes from 2001 to 2022:

```
# start with the unique dates and their counts
numCrimesTotal <- table(dfTotal)
countsTotal <- data.frame(numCrimesTotal)
colnames(countsTotal) <- c('Date',
                           'Number of Crimes')

countsTotal %>% head
```

```
##         Date Number of Crimes
## 1 2001-01-01             1825
## 2 2001-01-02             1143
## 3 2001-01-03             1151
## 4 2001-01-04             1166
## 5 2001-01-05             1267
## 6 2001-01-06             1292
```

Add a variable for when there is a leap day:

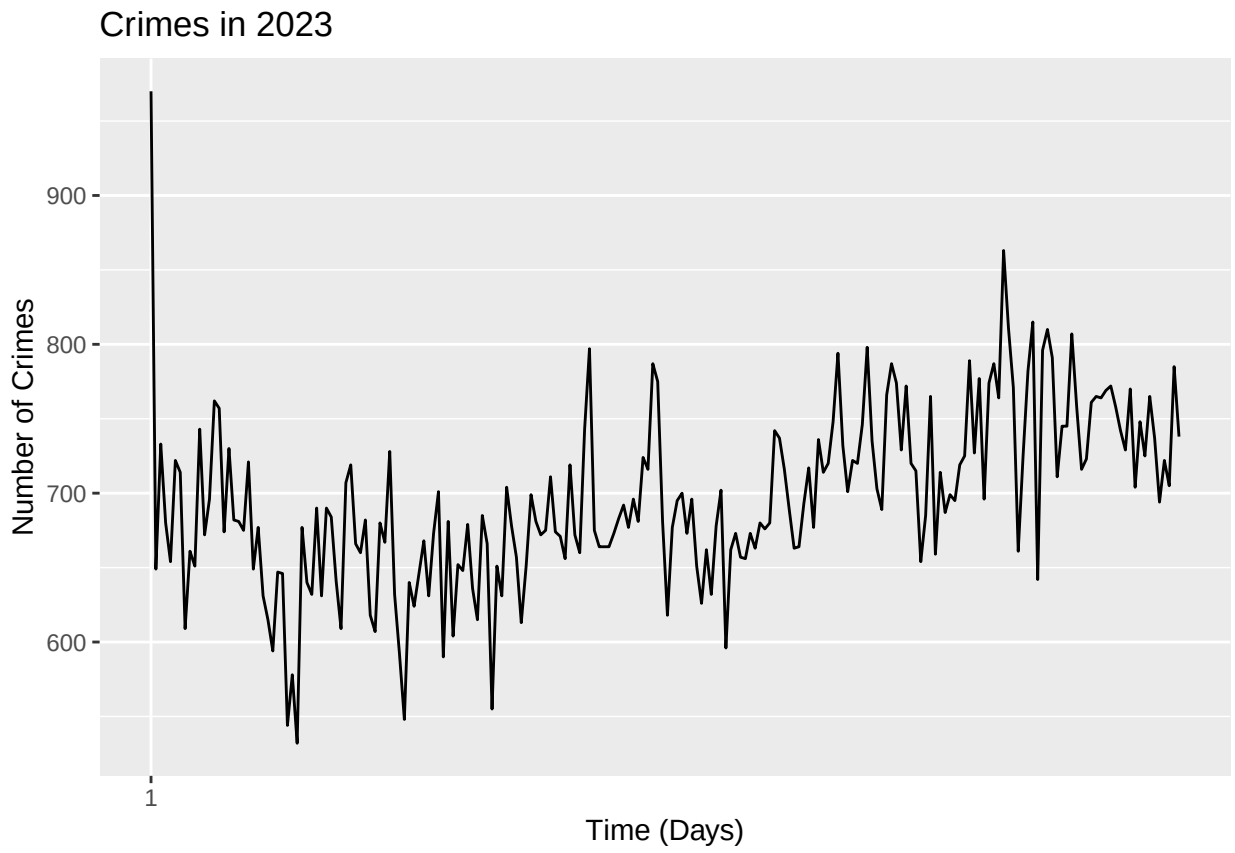
```
# 1 for a leap day, 0 for not a leap day
countsTotal['Leap Days'] <- grepl('02-29', countsTotal$Date) %>%
                                ifelse(1, 0)
counts2023['Leap Days'] <- rep(0, nrow(counts2023))
```

Add a variable for when it is new years:

```
# 1 for new years, 0 for not new years
countsTotal['New Years'] <- grepl('-01-01', countsTotal$Date) %>%
                                ifelse(1, 0)
counts2023['New Years'] <- c(1, rep(0, nrow(counts2023)-1))
```

Visualization of 2023 (Jan through August) crimes time series:

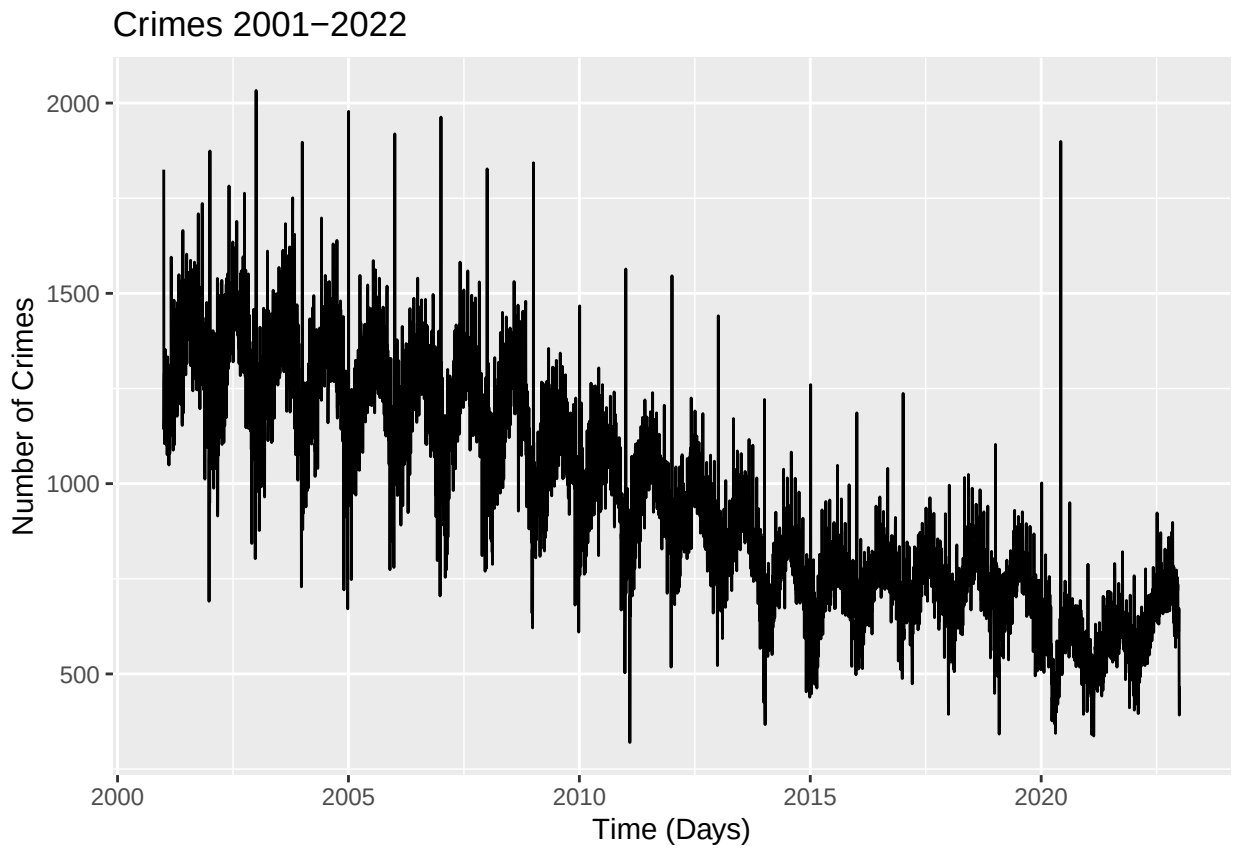
```
# ts for visualization
crimes23 <- ts(counts2023,
               frequency = 365,
               deltat = 1/365)

crimes23[,2] %>% autoplot(xlab = 'Time (Days)',
                     ylab = 'Number of Crimes',
                     main = 'Crimes in 2023')
```



Visualization of 2001 - 2022 crimes time series:

```
# ts for visualization
crimes01_22 <- ts(countsTotal,
                  start = c(2001, 1),
                  end = c(2022, 365),
                  frequency = 365,
                  deltat = 1/365)
```

```
autoplot(crimes01_22[,2],
         ylab = 'Number of Crimes',
         xlab = 'Time (Days)',
         main = 'Crimes 2001-2022')
```



The initial time series visually shows a downward trend and seasonality. The seasonality seems to be yearly. The variance in the data also seems to be non-constant. A box-cox transformation may help this.

Setting other time series:

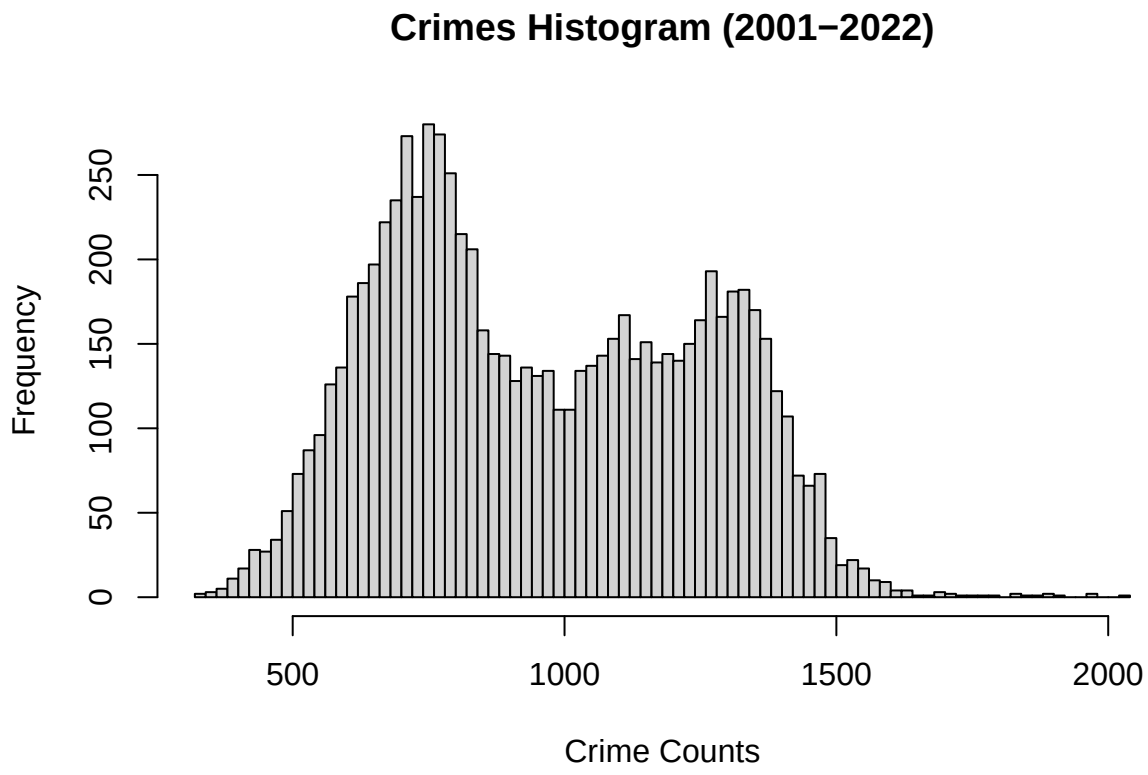
```
# time series, 2001-2019
crimes01_19 <- ts(countsTotal,
                  start = c(2001, 1),
                  end = c(2019, 365),
                  frequency = 365,
                  deltat = 1/365)

# time series, 2020-2022
crimes20_22 <- ts(countsTotal[6940:nrow(countsTotal),],
                  start = c(2020, 1),
```

```
                  end = c(2022, 365),
                  frequency = 365,
                  deltat = 1/365)

# crimes from 2001-2019, multiple seasonality
crimes01_19MS <- msts(countsTotal,
                      start = c(2001, 1),
                      end = c(2019, 365),
                      seasonal.periods = c(365,
                                           7))

# time series, 2020-2022, multiple seasonality
crimes20_22MS <- msts(countsTotal[6940:nrow(countsTotal),],
                      start = c(2020, 1),
                      end = c(2022, 365),
                      seasonal.periods = c(365,
                                           7))

# time series, 2001-2023
counts01_23 <- rbind(countsTotal, counts2023)

crimes01_23 <- ts(counts01_23,
                  start = c(2001, 1),
                  frequency = 365)

# time series, 2020-2023
crimes20_23 <- ts(counts01_23[6940:nrow(counts01_23),],
                  start = c(2020, 1),
                  frequency = 365,
                  deltat = 1/365)

# time series, 2001-2023, multiple seasonality
crimes01_23MS <- msts(counts01_23,
                      start = c(2001, 1),
                      seasonal.periods = c(365,
                                           7))

# time series, 2020-2023, multiple seasonality
crimes20_23MS <- msts(counts01_23[6940:nrow(counts01_23),],
                      start = c(2020, 1),
                      seasonal.periods = c(365,
                                           7))
```

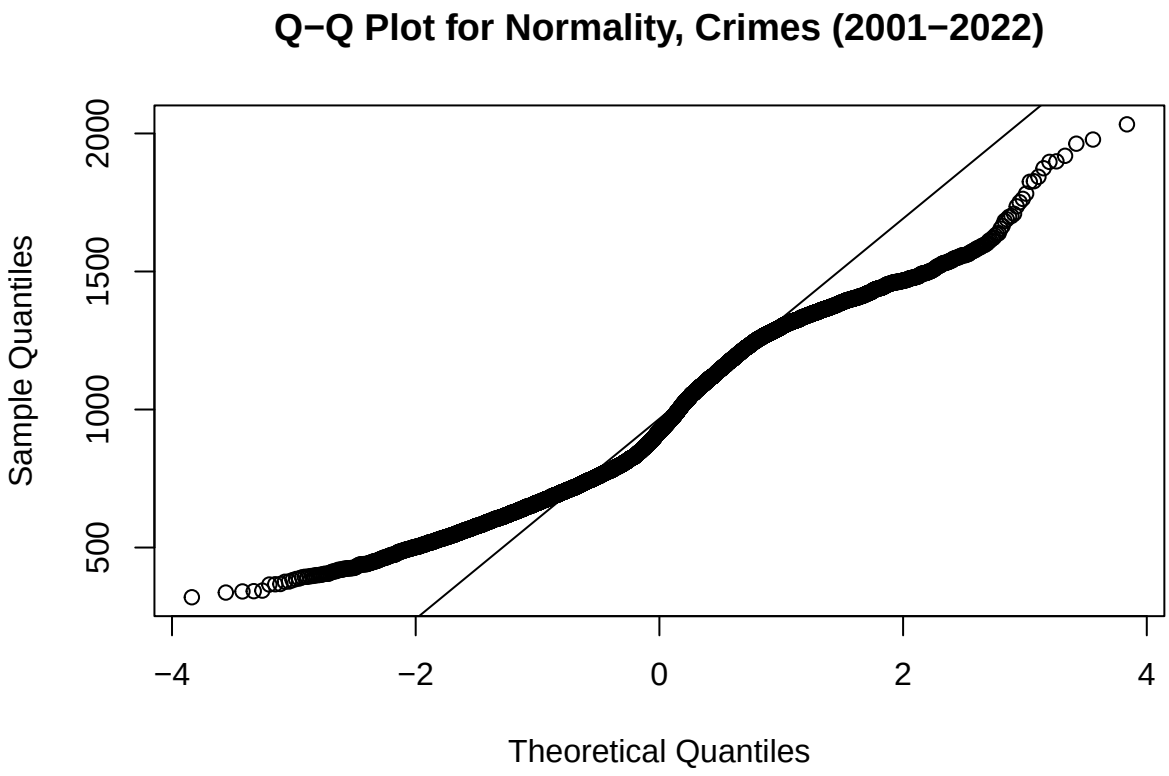
Is the distribution normally distributed?

```
# Histogram for number of crimes, 2001-2022
hist(countsTotal$`Number of Crimes`,
     breaks = 100,
```

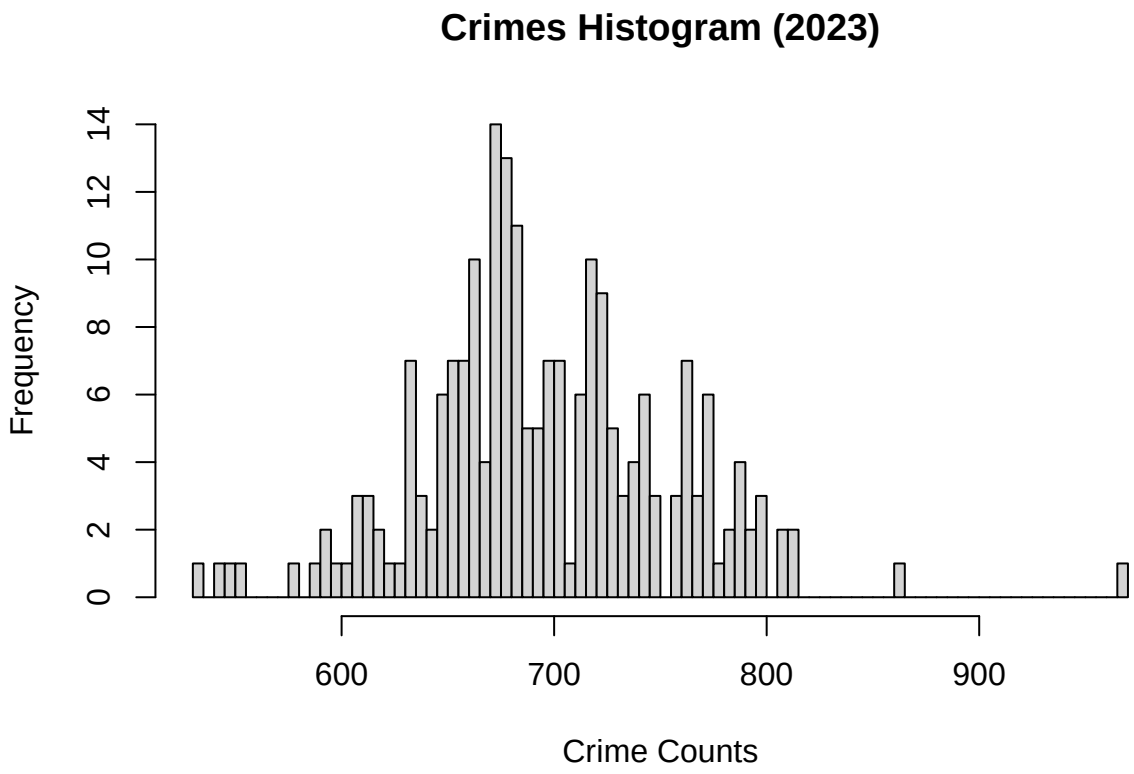
```
     main = 'Crimes Histogram (2001-2022)',
     xlab = 'Crime Counts')
```



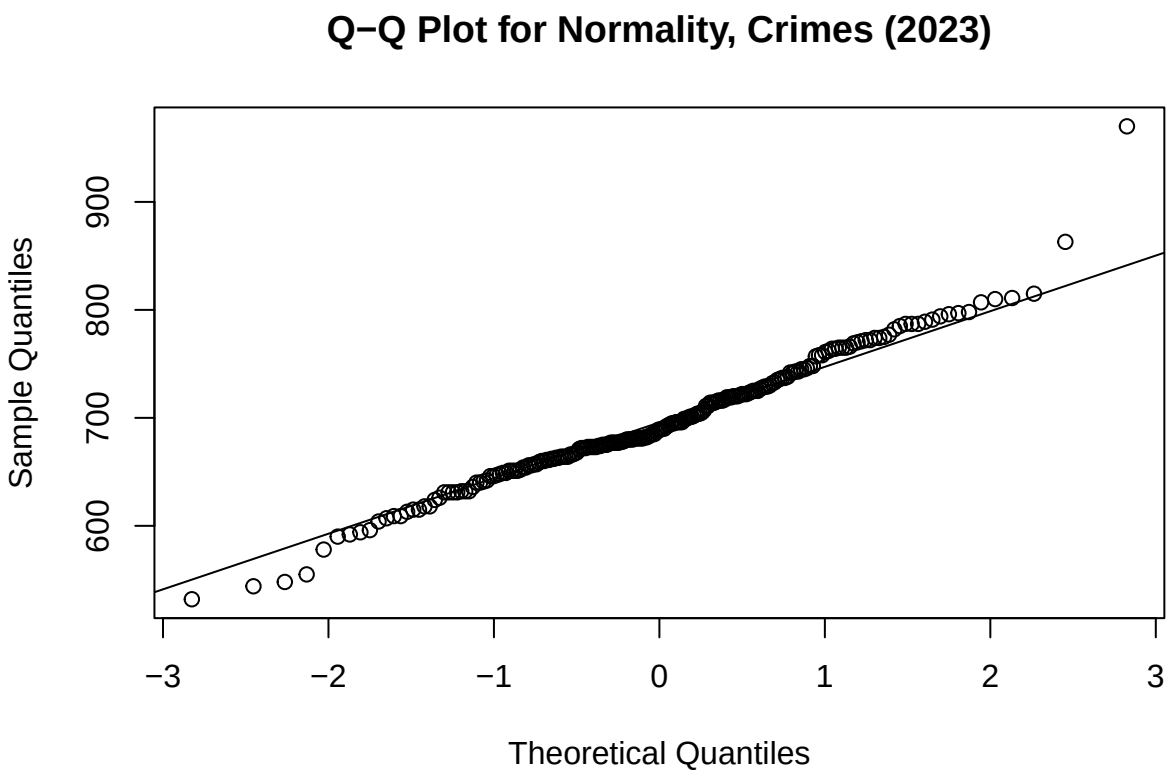

```
# Normality plots for number of crimes, 2001-2022
qqnorm(countsTotal$`Number of Crimes`,
       main = 'Q-Q Plot for Normality, Crimes (2001-2022)')
qqline(countsTotal$`Number of Crimes`)
```



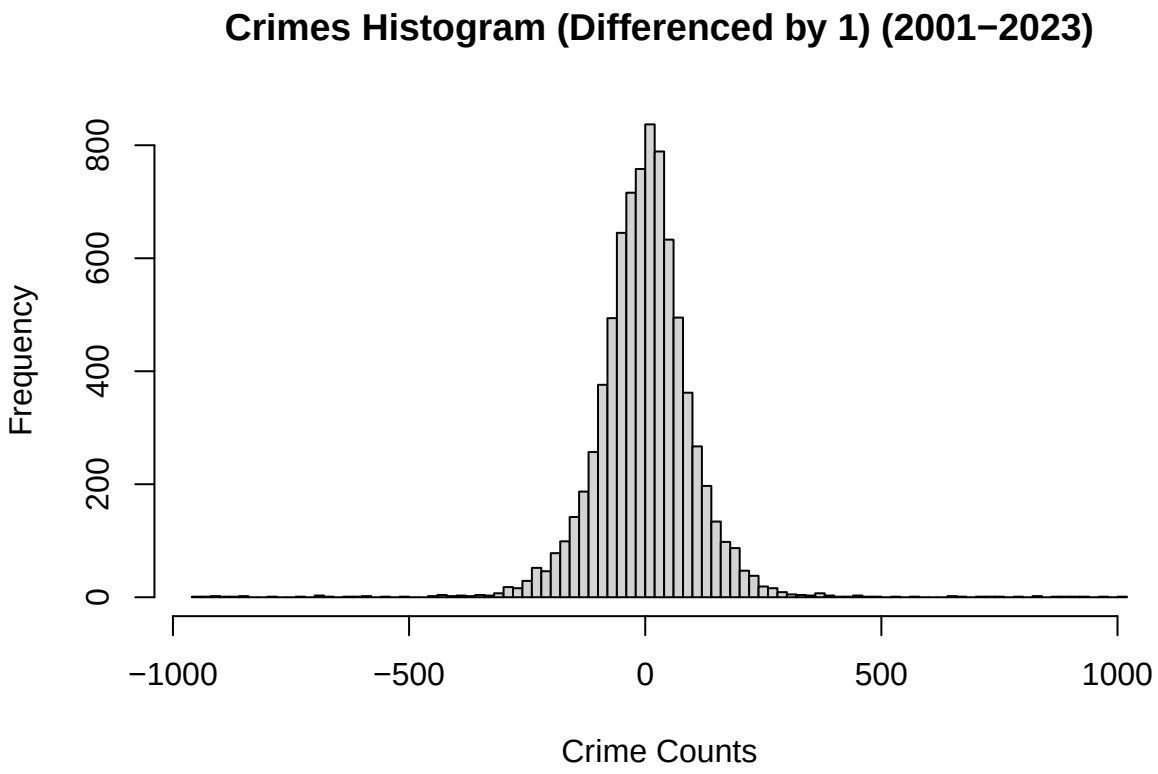

```
# Histogram for number of crimes, 2001-2022
hist(counts2023$`Number of Crimes`,
     breaks = 100,
     main = 'Crimes Histogram (2023)',
     xlab = 'Crime Counts')
```



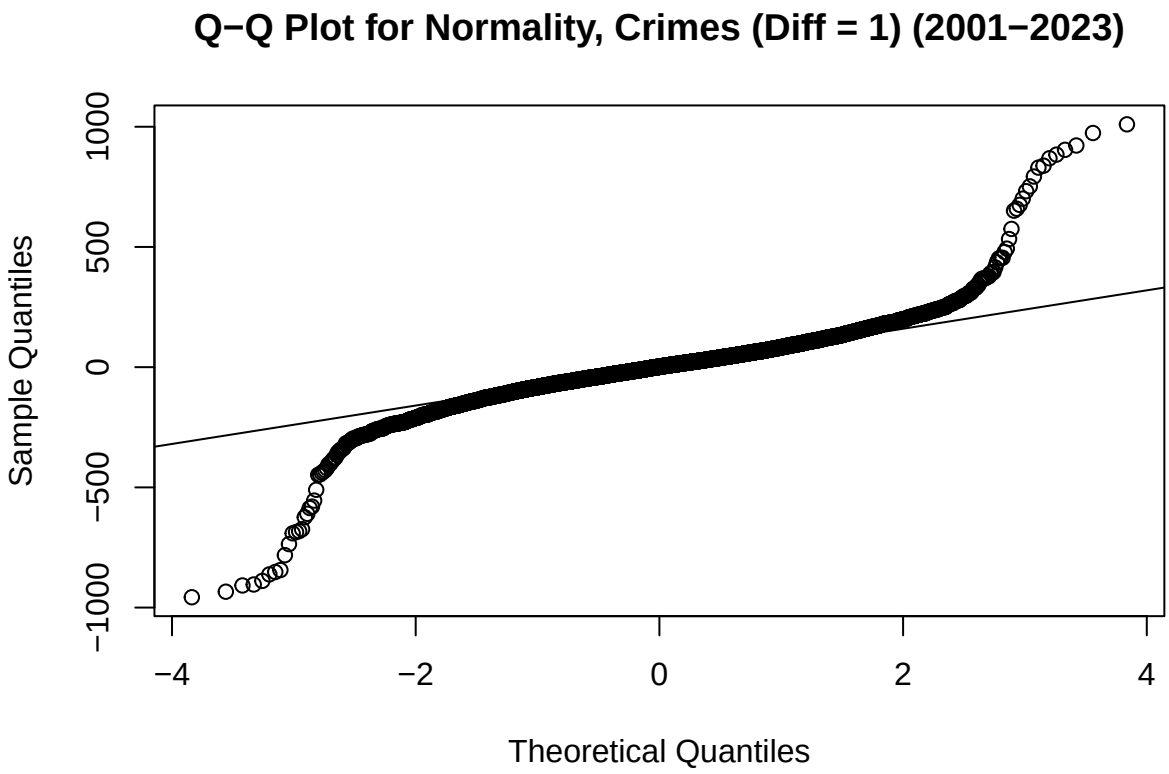

```
# Normality plots for number of crimes, 2023
qqnorm(counts2023$`Number of Crimes`,
       main = 'Q-Q Plot for Normality, Crimes (2023)')
qqline(counts2023$`Number of Crimes`)
```




```
# Histogram for number of crimes (after differencing), 2001-2022
hist(diff(countsTotal$`Number of Crimes`),
     breaks = 100,
     main = 'Crimes Histogram (Differenced by 1) (2001-2023)',
     xlab = 'Crime Counts')
```




```r
# Normality plots for number of crimes (after differencing), 2001-2022
qqnorm(diff(countsTotal$`Number of Crimes`),
       main = 'Q-Q Plot for Normality, Crimes (Diff = 1) (2001-2023)')
qqline(diff(countsTotal$`Number of Crimes`))
```

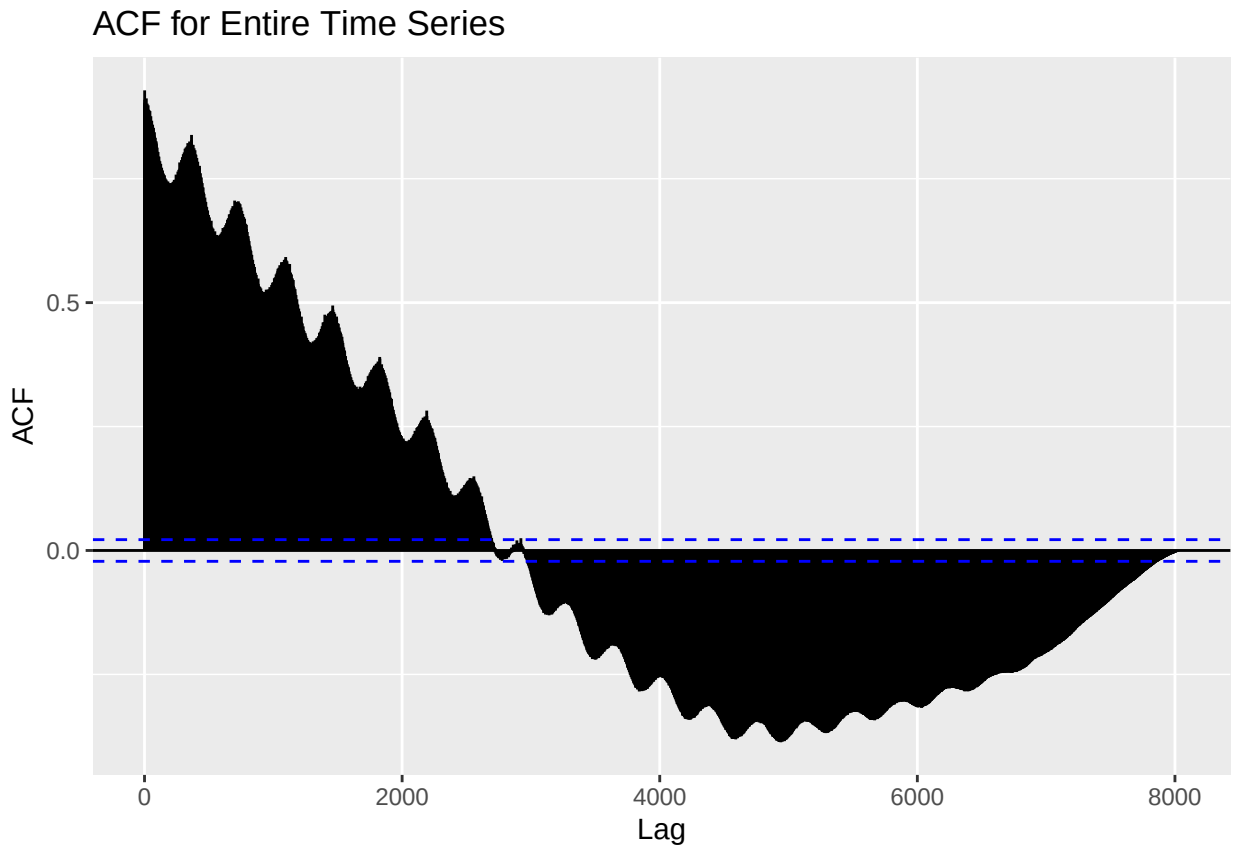


By the histogram of the 2023 crime counts, the crime counts seem to be normally distributed.

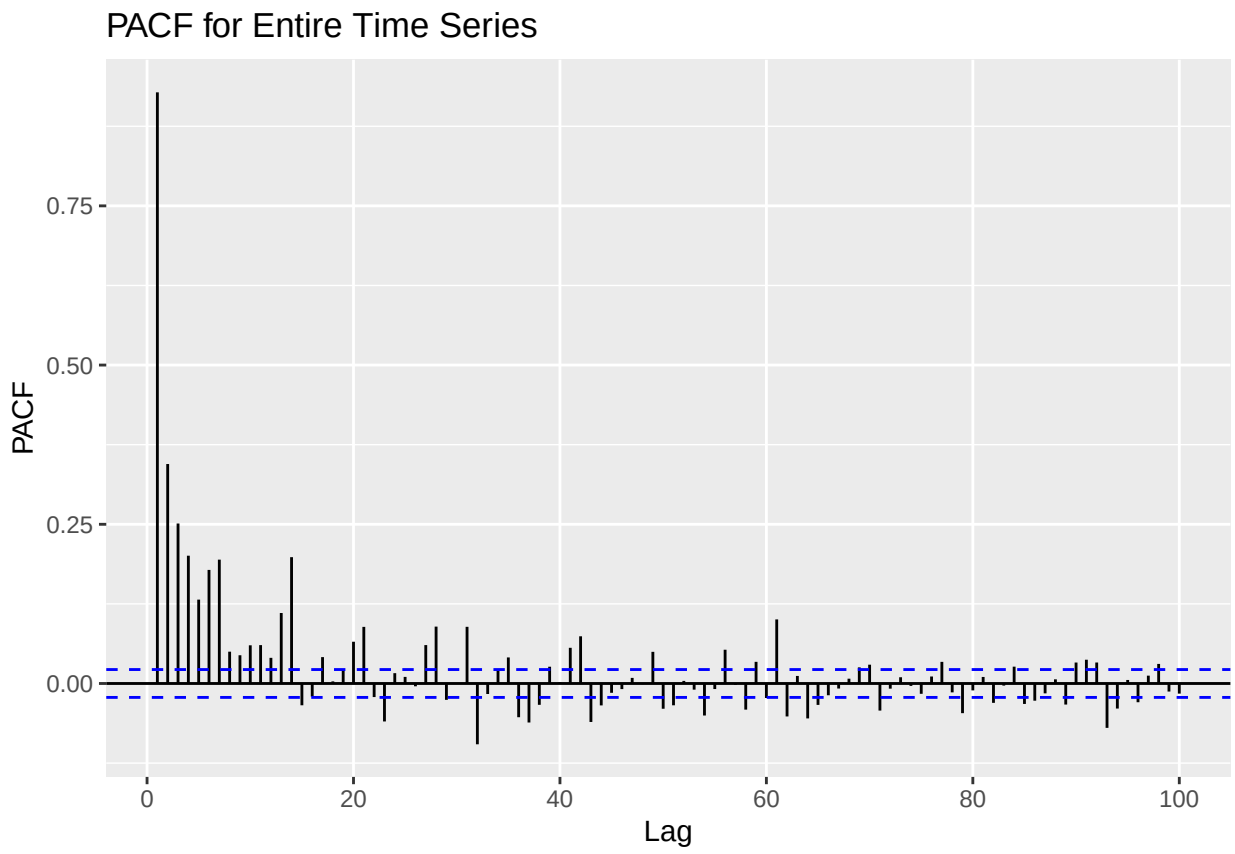
After differencing the total time series by one, the time series seems to represent a normal distribution.

ACF of Total Time Series:

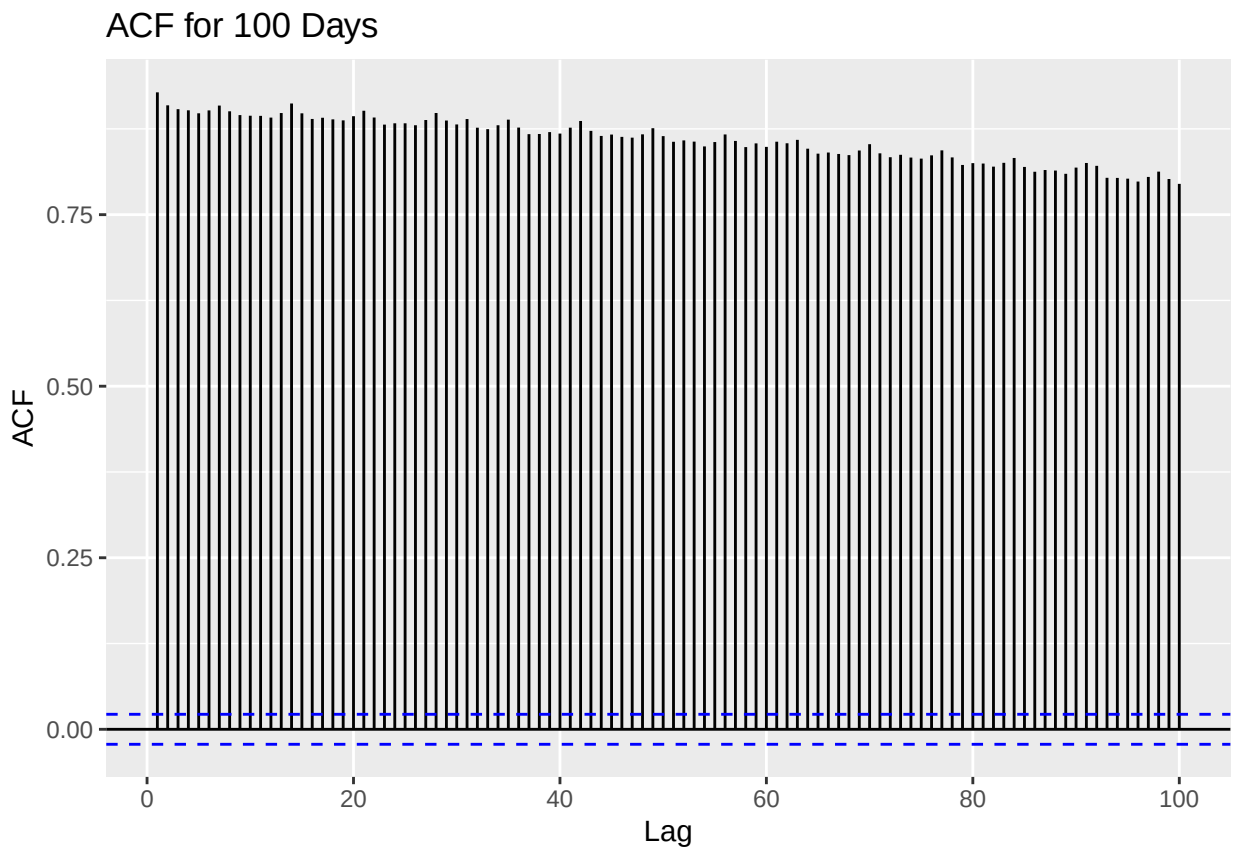
```
# acf for entire time series
ggAcf(crimes01_22[,2],
      lag.max = nrow(crimes01_22)) +
    ggtitle('ACF for Entire Time Series')
```




```
# PACF for entire time series
ggAcf(crimes01_22[,2],
      lag.max = 100,
      type = 'partial') +
    ggtitle('PACF for Entire Time Series')
```




```
# ACF for first 100 days of time series (to look at time series closer)
ggAcf(crimes01_22[,2],
      lag.max = 100) +
    ggtitle('ACF for 100 Days')
```



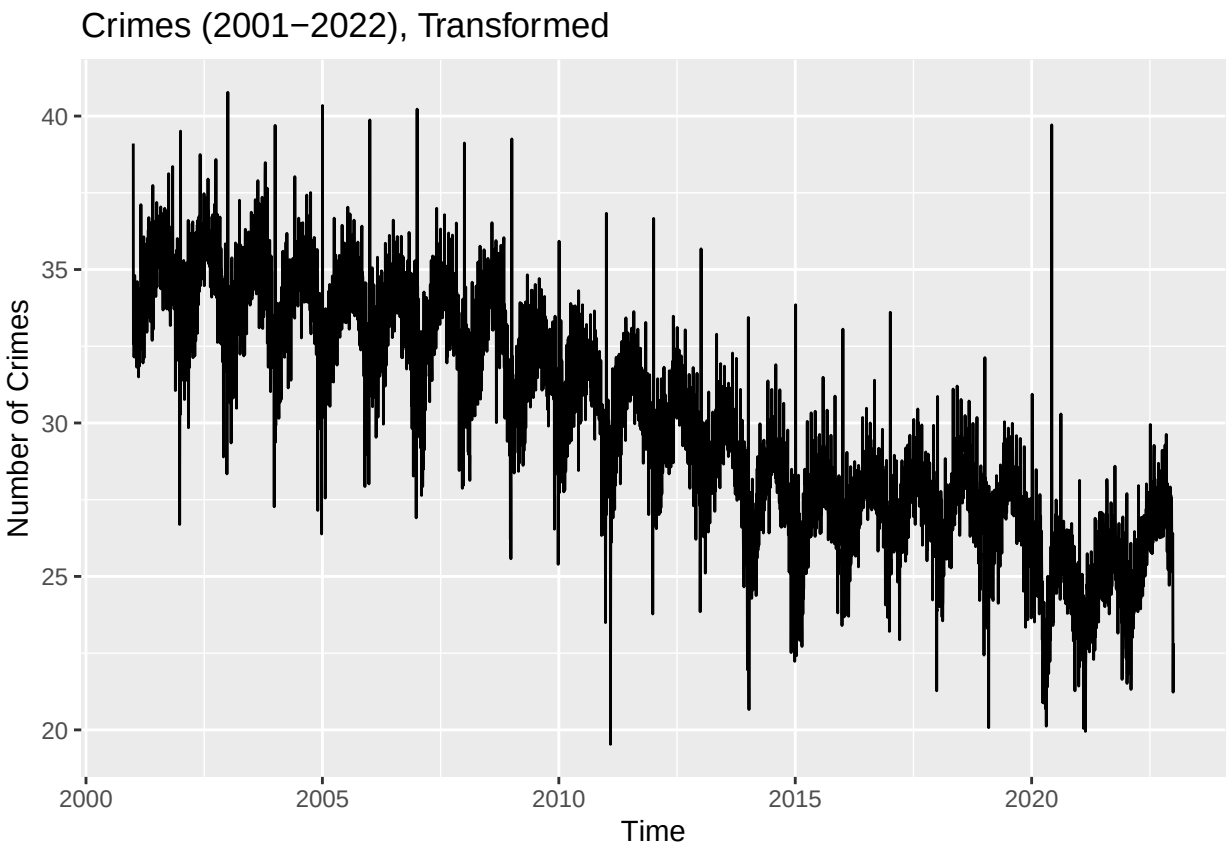
The ACF shows a clear downward trend and yearly seasonality in the data in the ACF plot for the entire time series.

From the 100 days ACF plot, it seems that there is a type of weekly seasonality in the data as well.

By the above, the ACF plot shows non-stationarity in the time series.

Application of a Box-Cox transformation, which reduces variance in the model:

```
# ideal lambda value calculated for boxcox transformation
lambdaCrimes <- BoxCox.lambda(crimes01_22[,2])

# Presenting the transformed values
BoxCox(crimes01_22[,2],
       lambda = lambdaCrimes) %>%
    autoplot(ylab = 'Number of Crimes',
             main = 'Crimes (2001-2022), Transformed')
```



Applying a box-cox transformation appeared to remove the non-constant variance in the model.

KPSS Test for Stationarity:

```
# KPSS test for stationarity, hospitalizations
cat('KPSS Test for Cases, Segment 1:\n')
```

```
## KPSS Test for Cases, Segment 1:
```

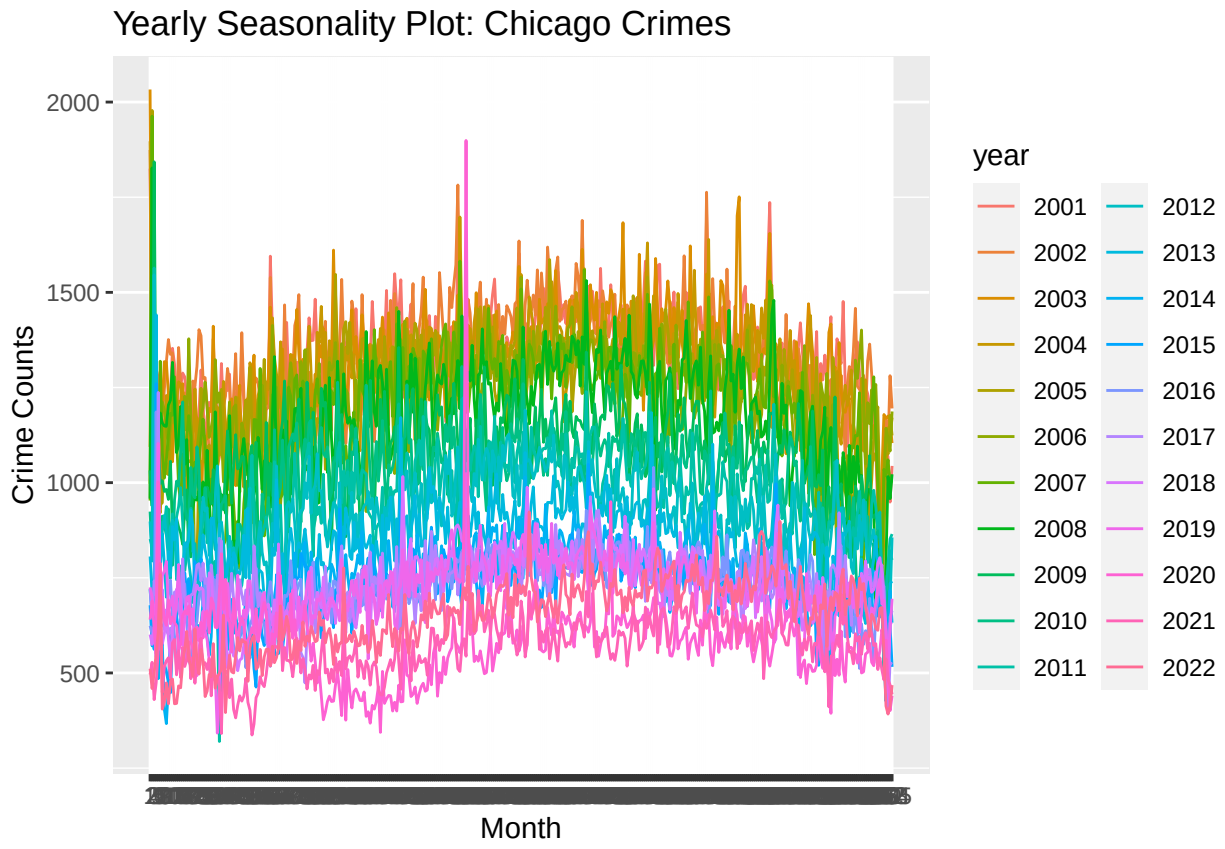
```
tseries::kpss.test(crimes01_22[,2], null = 'Trend')
```

```
## Warning in tseries::kpss.test(crimes01_22[, 2], null = "Trend"): p-value smaller
## than printed p-value

##
##  KPSS Test for Trend Stationarity
##
## data:  crimes01_22[, 2]
## KPSS Trend = 1.7668, Truncation lag parameter = 11, p-value = 0.01
```

Since the p-value from the KPSS test is less than 0.05, the null hypothesis that the time series is trend stationary can be rejected. This follows my previous hypothesis of non-stationarity in the data.

Seasonality (yearly):

```
ggseasonplot(crimes01_22[,2]) +
  ylab('Crime Counts') +
  xlab('Month') +
  ggtitle("Yearly Seasonality Plot: Chicago Crimes")
```

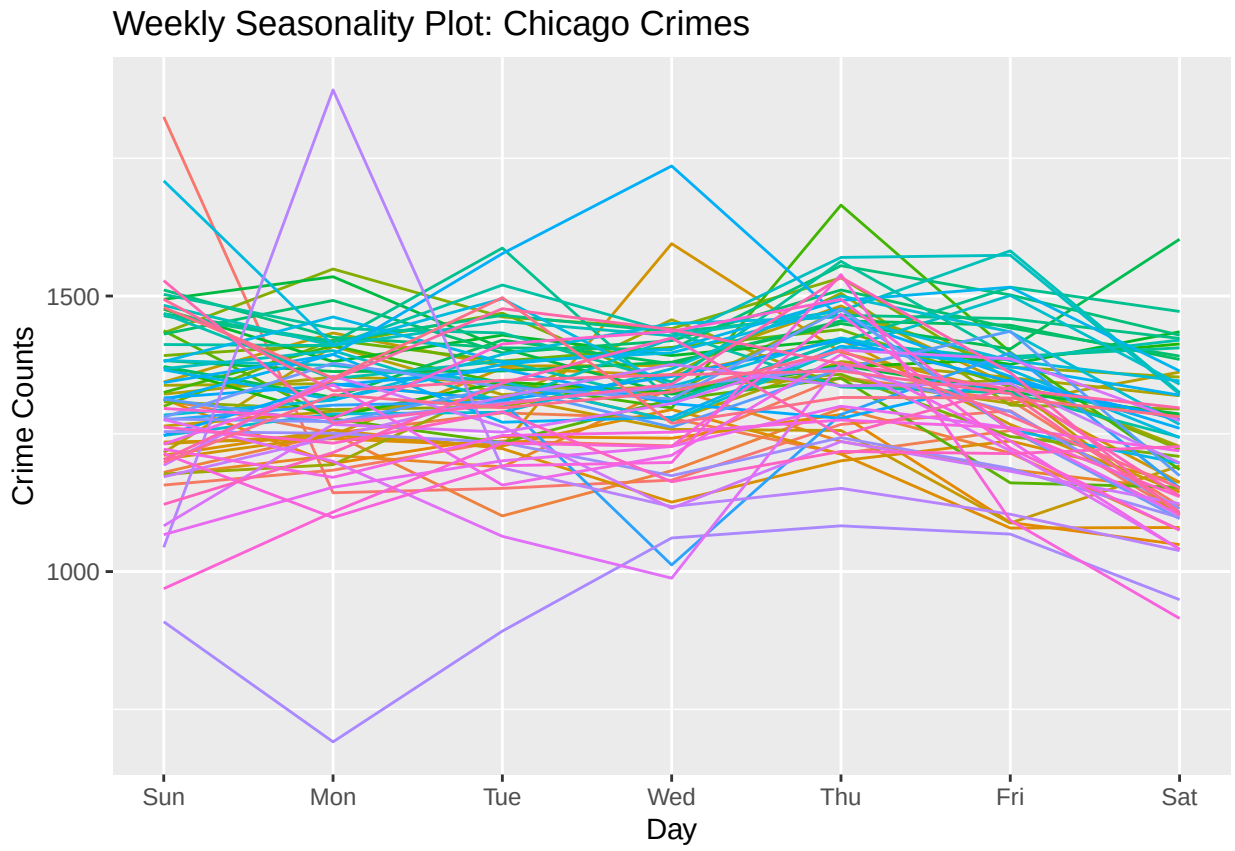


The seasonality plot shows a clear curve throughout the course of the year.

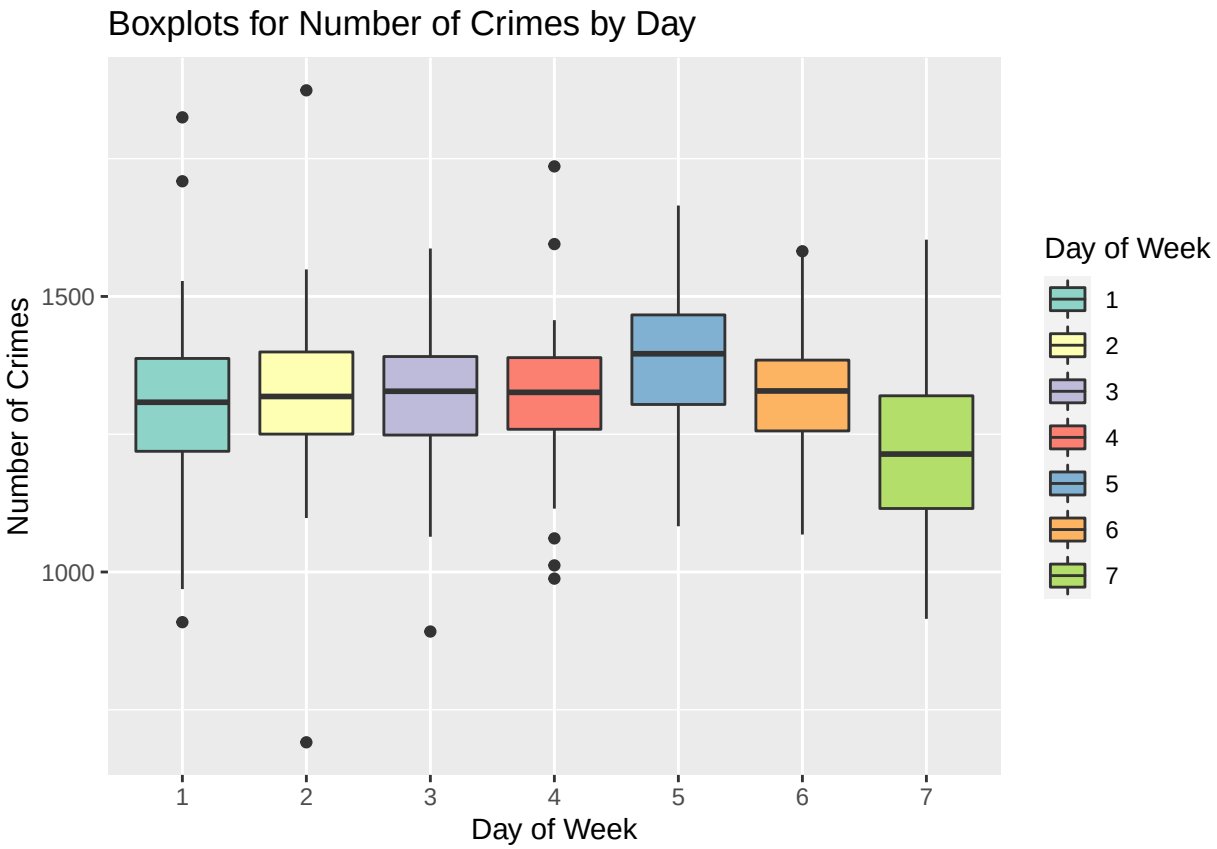
Seasonality (weekly):

```
# time series for just weekly seasonality visualization
crimesWeekly <- ts(countsTotal,
```

```
                    start = c(2001, 1),
                    end = c(2019, 365),
                    frequency = 7,
                    deltat = 1/7)

crimesWeekly[,2] %>%
   ggseasonplot() +
        ylab('Crime Counts') +
        xlab('Day') +
        ggtitle("Weekly Seasonality Plot: Chicago Crimes") +
        theme(legend.position = 'none')
```



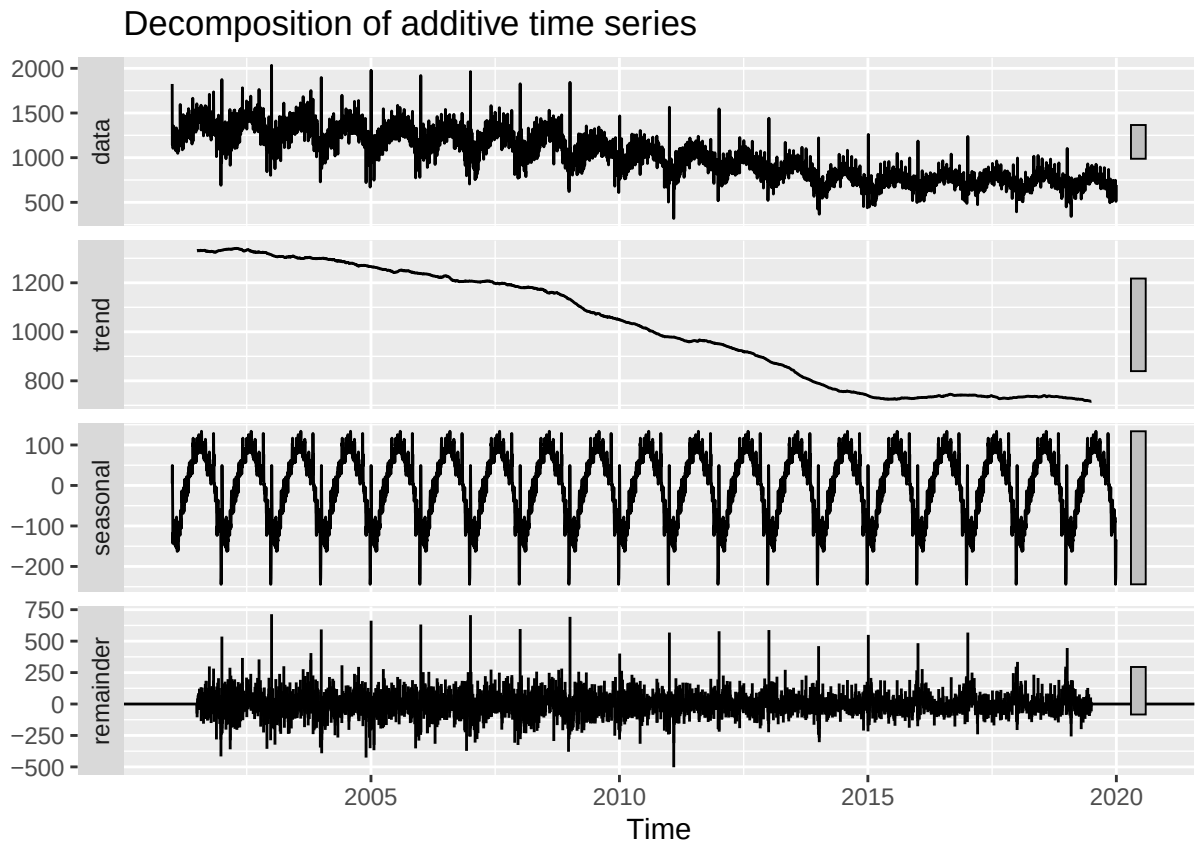

```
# boxplots by day
crimesWeekly[,2] %>%
    split(cycle(crimesWeekly)) %>%
    melt %>%
    ggplot(aes(x = L1,
               y = value,
               fill = L1)) +
      geom_boxplot() +
      scale_fill_brewer(palette="Set3", name='Day of Week') +
      ylab('Number of Crimes') +
      xlab('Day of Week') +
      ggtitle('Boxplots for Number of Crimes by Day')
```



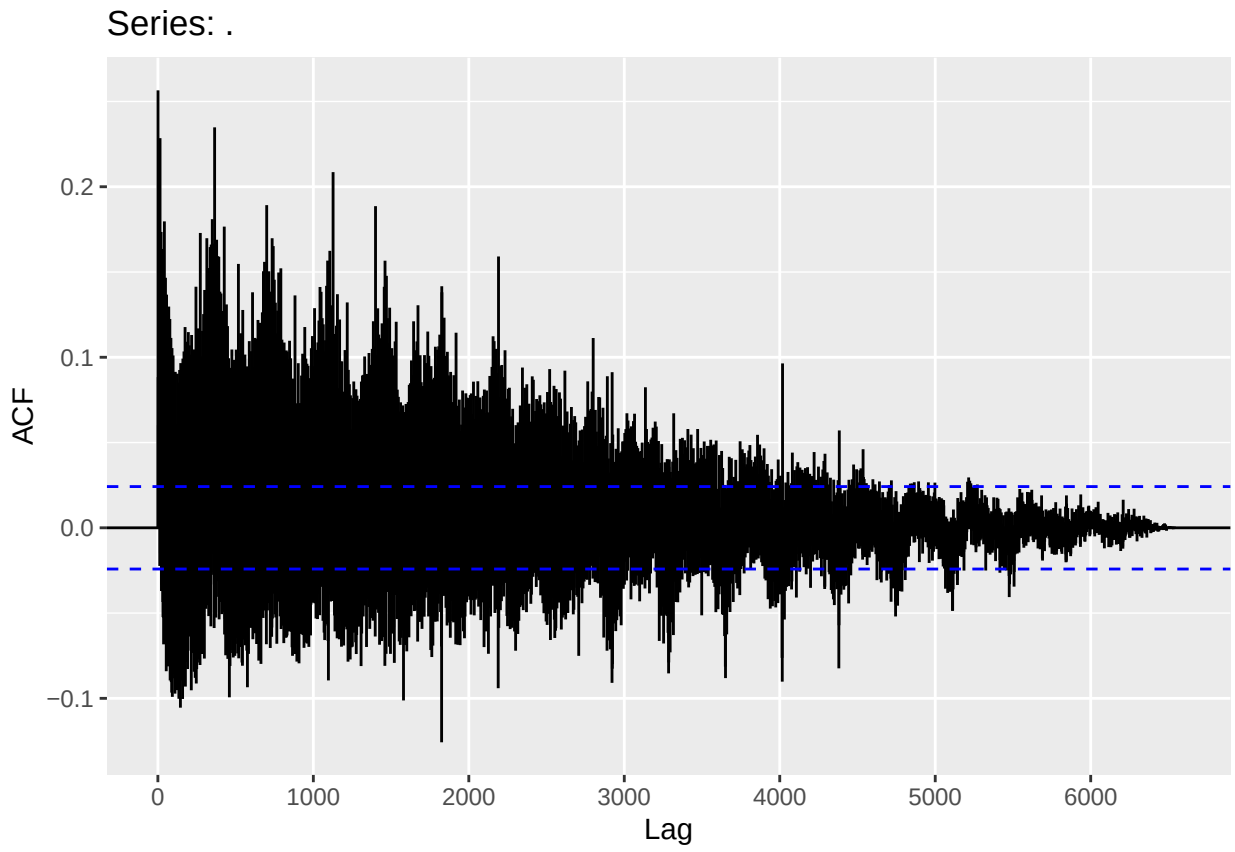
After plotting seasonality of weekly seasonality, there seems to be a slight pattern in weekly seasonality.

Decomposition (solely yearly seasonality):

```
# decomposed time series
decompTS <- decompose(crimes01_19[,2])

# plotting decomposed time series
decompTS %>% autoplot
```




```
# ACF for decomposed time series
na.omit(decompTS$random) %>%
    ggAcf(lag.max = length(decompTS$random))
```

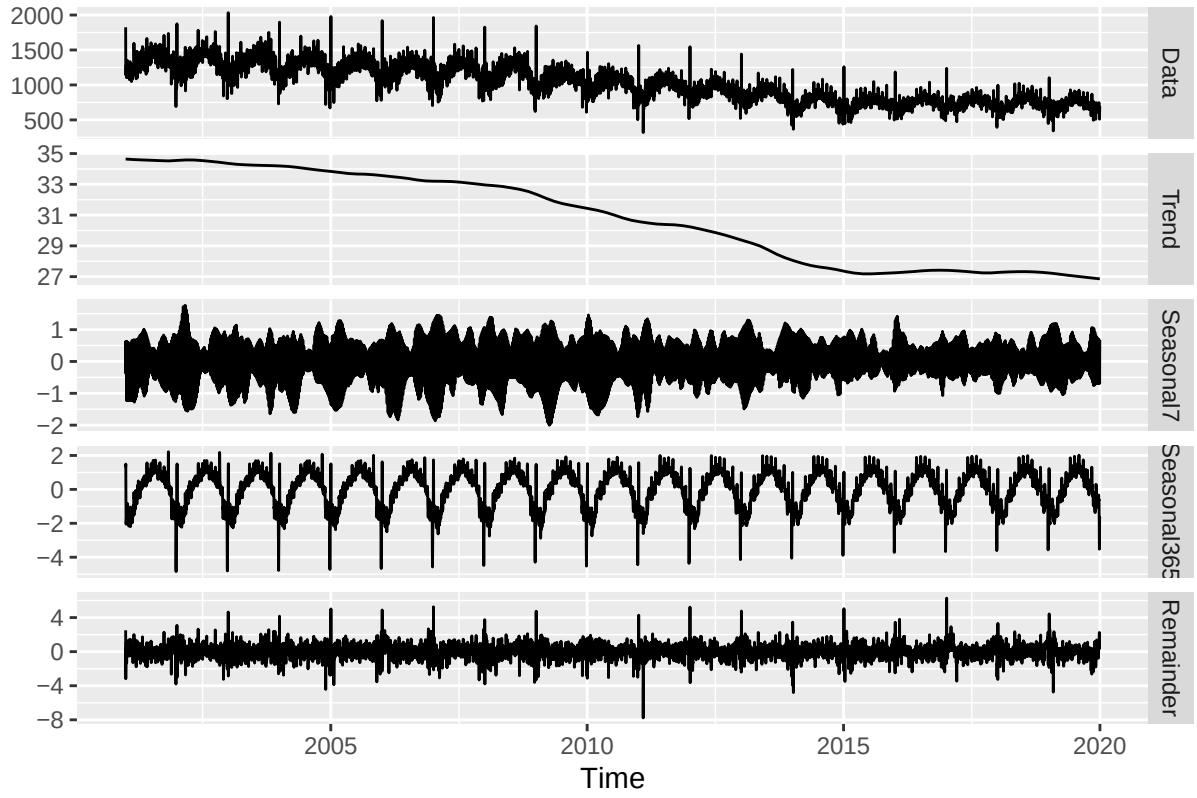


Removing the trend and stationarity from the time series seems to leave some potential non-constant variance, as the variance is larger at the beginning of the time series and smaller at the end. The lambda value from the box-cox transformation hopefully will help with this.

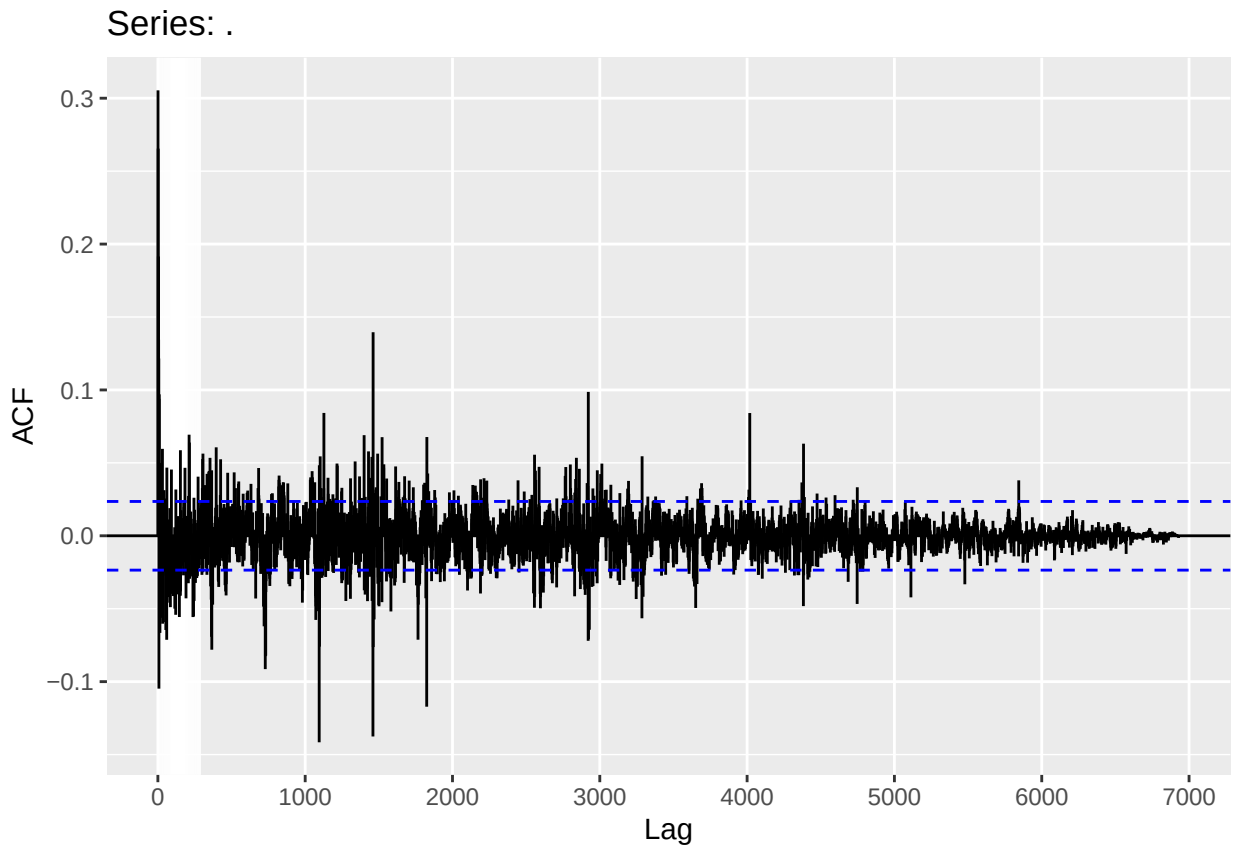
After computing the time series after seasonality and trend were removed, there still seems to be some kind of pattern in the data because of the large correlations in the ACF.

Decomposition (weekly and yearly seasonality together):

```
# decomposition of time series with multiple seasonality
decompTSMS <- crimes01_19MS[,2] %>%
               mstl(lambda = lambdaCrimes)

# plot of decomposed time series
decompTSMS %>%
    autoplot
```




```
# ACF of decomposed time series
decompTSMS[,5 ] %>%
    ggAcf(lag.max = length(decompTSMS[,5]))
```



After using decomposition that removes weekly seasonality alongside yearly seasonality and the trend, the time series shows much smaller correlations in the ACF.

Now that basic information about the time series is understood, the models for the time series will be built. A number of different strategies will be used. These strategies include the use of ARIMA, TBATS, STLM ARIMA, and STLM ETS to build the models. Also, the ARIMA models will be built with just yearly seasonality for one model and both yearly and weekly seasonality (MS = multiple seasonality) for other models.

The models will be built using data from 2001 to 2019. Furthermore, the models will be compared based on accuracy measures from building the models. The models will also be compared using accuracy measures from comparing forecasted values from 2020 to 2023 with actual values from 2020 to 2023. By the above, the best model will be chosen.

Model Creation:

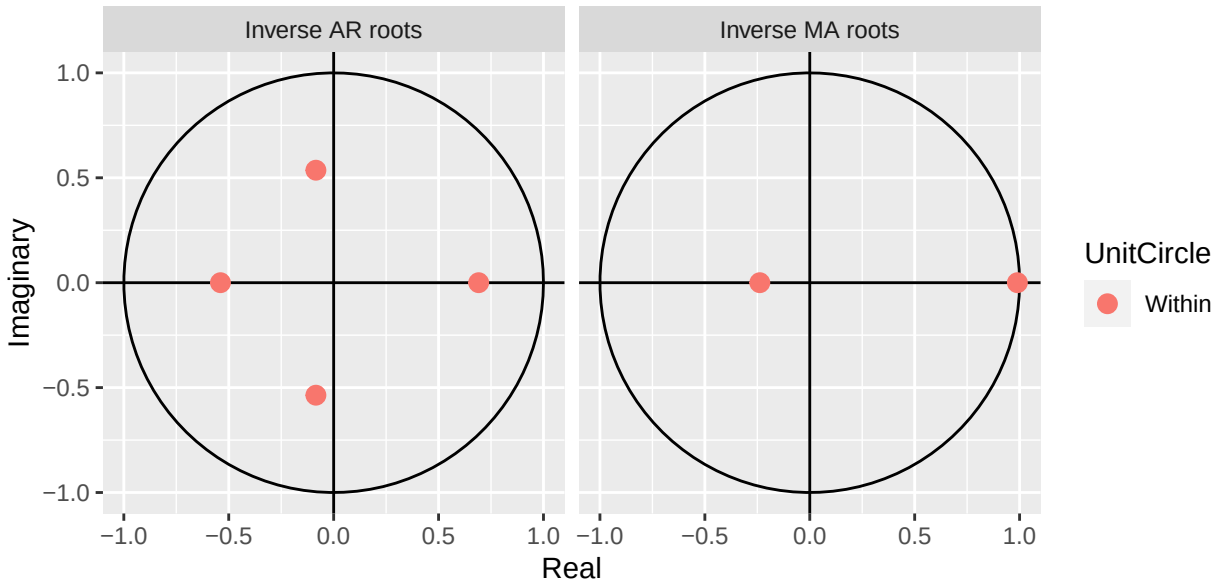
Fitting the model to find the ideal Arima model (just yearly seasonality):

```
# ARIMA model of time series with just yearly seasonality
crimesArima <- auto.arima(crimes01_19[,2],
                          stationary = FALSE,
```

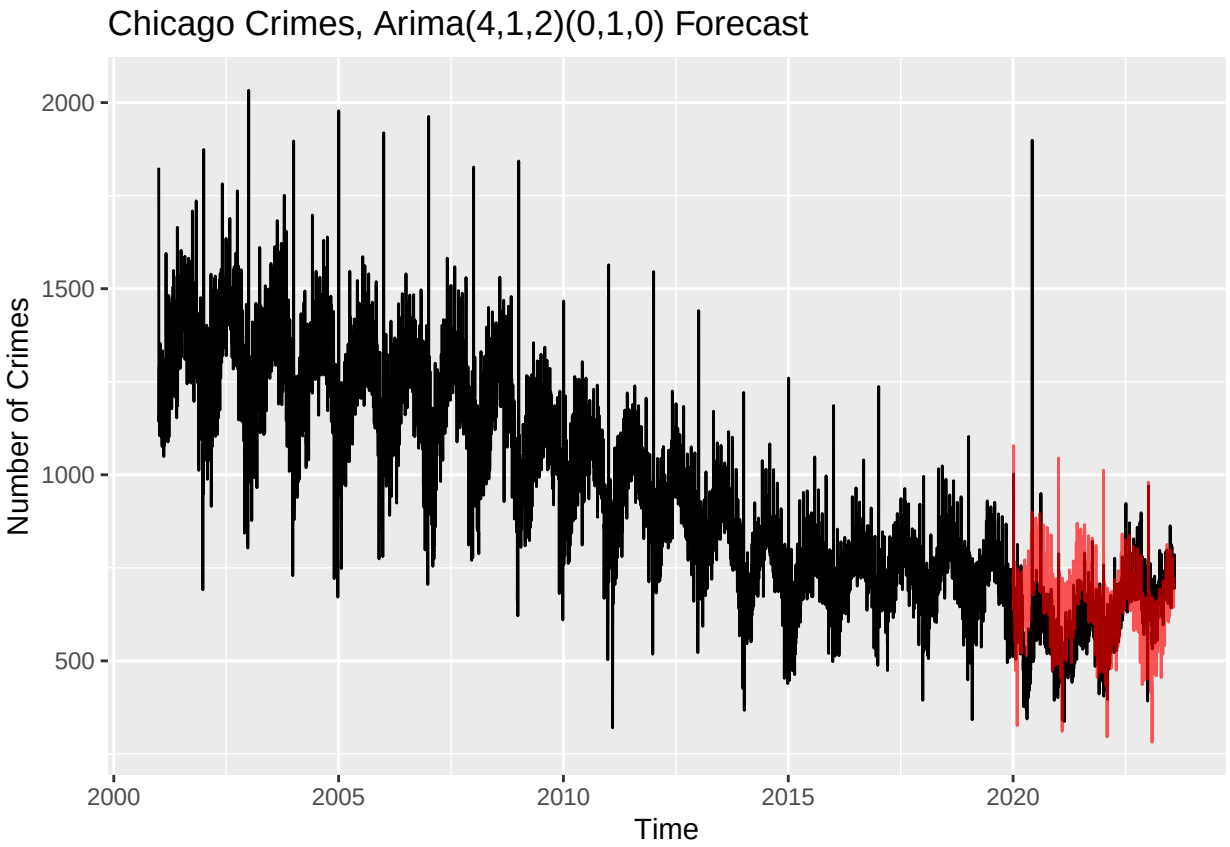
```
                          lambda = lambdaCrimes,
                          biasadj = FALSE,
                          d = 1,
                          D = 1,
                          xreg = crimes01_19[,3])

# forecast of arima model
foreArima <- forecast(crimesArima,
                      biasadj = FALSE,
                      lambda = lambdaCrimes,
                      xreg = crimes20_23[,3])

# summary of the model
summary(crimesArima)
```

```
## Series: crimes01_19[, 2]
## Regression with ARIMA(4,1,2)(0,1,0)[365] errors
## Box Cox transformation: lambda= 0.3620037
##
## Coefficients:
##           ar1     ar2     ar3     ar4      ma1      ma2     xreg
##       -0.0174  0.1039  0.1078  0.1098  -0.7513  -0.2356  -0.9361
## s.e.   0.0712  0.0202  0.0131  0.0140   0.0708   0.0698   0.4364
##
## sigma^2 = 1.641:  log likelihood = -10948.27
## AIC=21912.55   AICc=21912.57   BIC=21966.87
##
## Training set error measures:
##                      ME     RMSE     MAE        MPE    MAPE      MASE
## Training set -0.4908575 101.9996 73.5724 -0.5650656 7.63688 0.8150121
##                      ACF1
## Training set -0.02287071
```

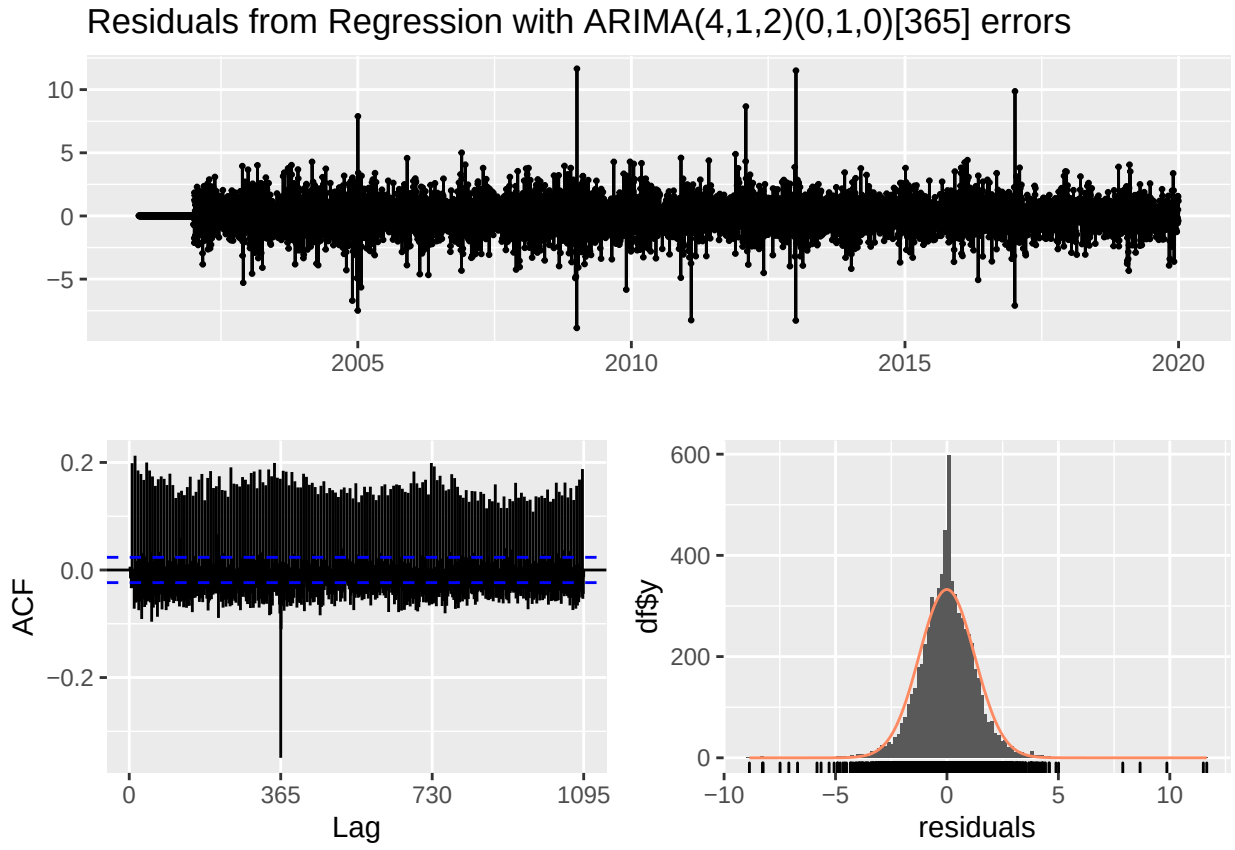
```
# plot of the model
autoplot(crimesArima)
```




```
# plot of the model with
autoplot(crimes01_23[,2],
         ylab = 'Number of Crimes',
         main = 'Chicago Crimes, Arima(4,1,2)(0,1,0) Forecast') +
    autolayer(foreArima$mean,
              series = 'Arima(4,1,2)(0,1,0)',
              alpha = 0.65,
              color = 'red')
```




```
# check the residuals of the model to determine whether they are normal
residArima <- checkresiduals(crimesArima)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(4,1,2)(0,1,0)[365] errors
## Q* = 28515, df = 724, p-value < 2.2e-16
##
## Model df: 6.   Total lags used: 730
```

Fitting the model to find the ideal Arima model (yearly seasonality + weekly seasonality):

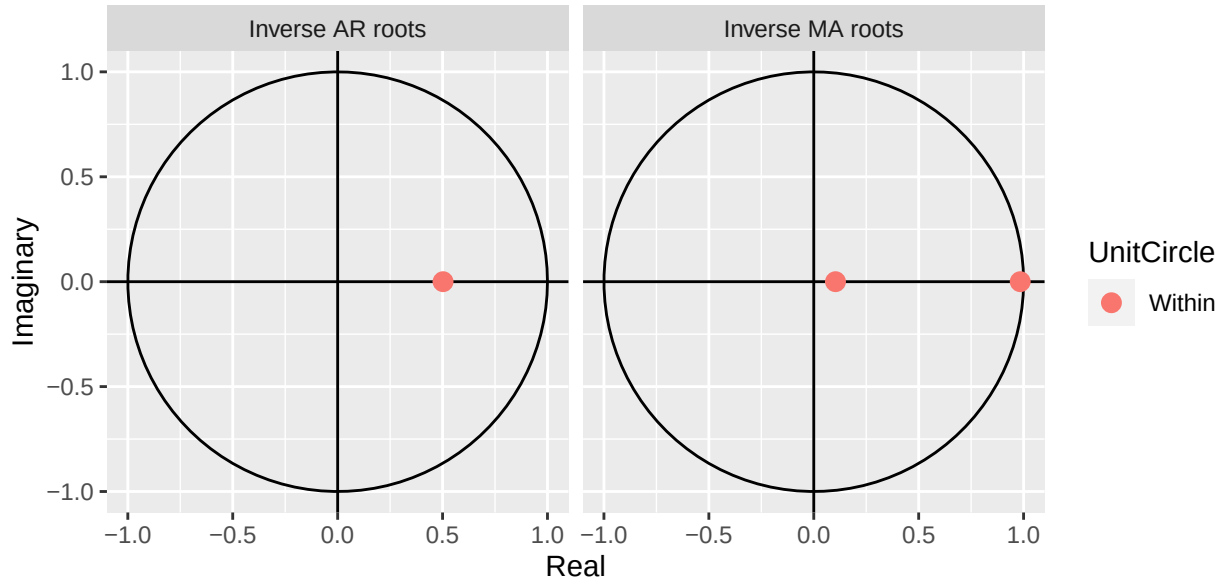
```
#
crimesArimaMS <- auto.arima(crimes01_19MS[,2],
                            stationary = FALSE,
                            seasonal = FALSE,
                            lambda = lambdaCrimes,
                            biasadj = FALSE,
                            xreg = cbind(crimes01_19MS[,3],
                                         crimes01_19MS[,4],
                                         fourier(crimes01_19MS[,2],
```

```
                                            K=c(3,3))))

foreArimaMS <- forecast(crimesArimaMS,
                        lambda = lambdaCrimes,
                        biasadj = FALSE,
                        xreg = cbind(crimes20_23MS[,3],
                                     crimes20_23MS[,4],
                                     fourier(crimes20_23MS[,2],
                                             K=c(3,3))))

summary(crimesArimaMS)
```

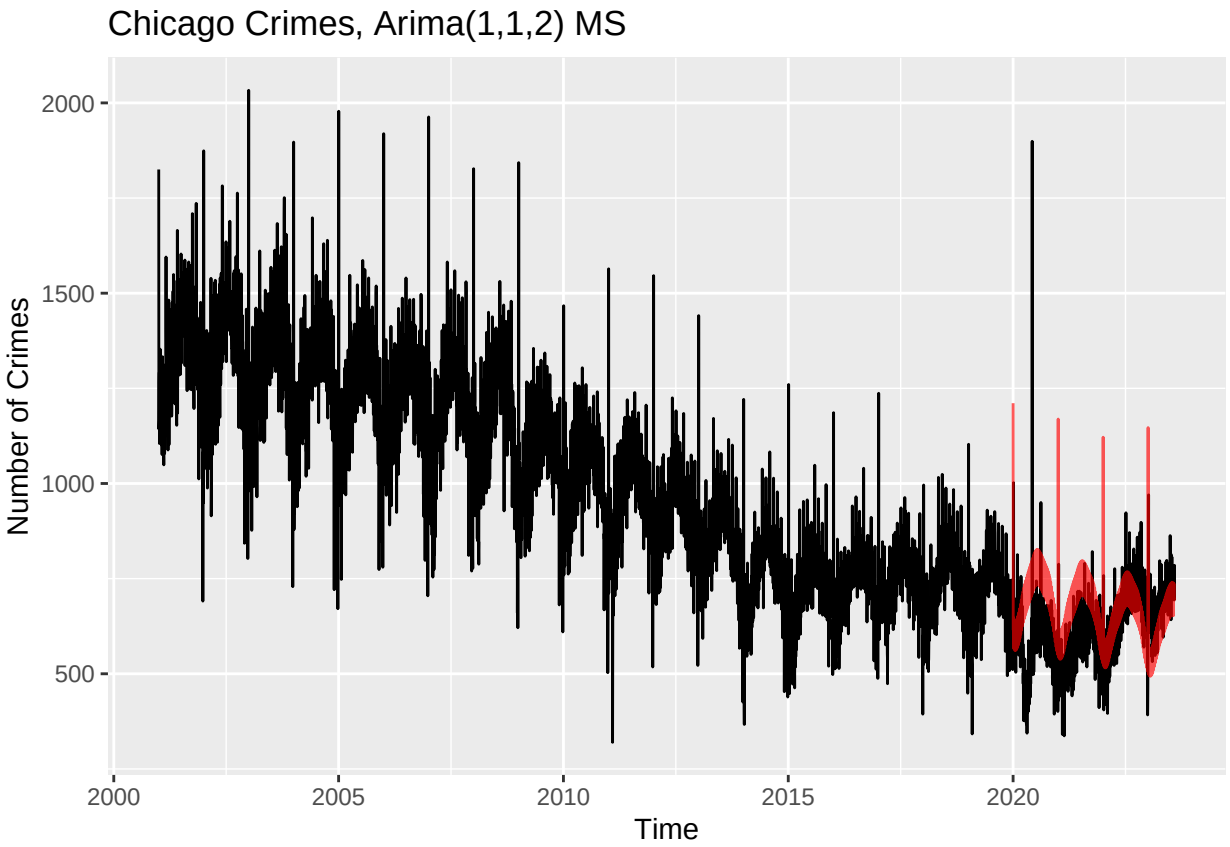
```
## Series: crimes01_19MS[, 2]
## Regression with ARIMA(1,1,2) errors
## Box Cox transformation: lambda= 0.3620037
##
## Coefficients:
##          ar1      ma1     ma2    drift  crimes01_19MS[, 3]  crimes01_19MS[, 4]
##       0.5024  -1.0876  0.1020  -0.0011             -0.3466              7.8299
## s.e.  0.0278   0.0317  0.0304   0.0003              0.4166              0.1923
##       fourier(crimes01_19MS[, 2], K = c(3, 3)).S1-7
##                                            -0.2135
## s.e.                                        0.0180
##       fourier(crimes01_19MS[, 2], K = c(3, 3)).C1-7
##                                            -0.2643
## s.e.                                        0.0180
##       fourier(crimes01_19MS[, 2], K = c(3, 3)).S2-7
##                                             0.0034
## s.e.                                        0.0129
##       fourier(crimes01_19MS[, 2], K = c(3, 3)).C2-7
##                                            -0.3391
## s.e.                                        0.0129
##       fourier(crimes01_19MS[, 2], K = c(3, 3)).S3-7
##                                             0.1593
## s.e.                                        0.0113
##       fourier(crimes01_19MS[, 2], K = c(3, 3)).C3-7
##                                            -0.0414
## s.e.                                        0.0113
##       fourier(crimes01_19MS[, 2], K = c(3, 3)).S1-365
##                                              -0.5488
## s.e.                                          0.0379
##       fourier(crimes01_19MS[, 2], K = c(3, 3)).C1-365
##                                              -1.2637
## s.e.                                          0.0376
##       fourier(crimes01_19MS[, 2], K = c(3, 3)).S2-365
##                                              -0.1522
## s.e.                                          0.0303
##       fourier(crimes01_19MS[, 2], K = c(3, 3)).C2-365
##                                              -0.2045
## s.e.                                          0.0302
##       fourier(crimes01_19MS[, 2], K = c(3, 3)).S3-365
##                                              -0.0786
## s.e.                                          0.0286
```

```
##       fourier(crimes01_19MS[, 2], K = c(3, 3)).C3-365
##                                              -0.1112
## s.e.                                            0.0286
##
## sigma^2 = 0.8153:  log likelihood = -9123.16
## AIC=18284.32   AICc=18284.43   BIC=18414.36
##
## Training set error measures:
##                     ME     RMSE     MAE        MPE     MAPE      MASE       ACF1
## Training set 1.728643 73.88187 54.9581 -0.3913793 5.651131 0.6088088 -0.0296849
```

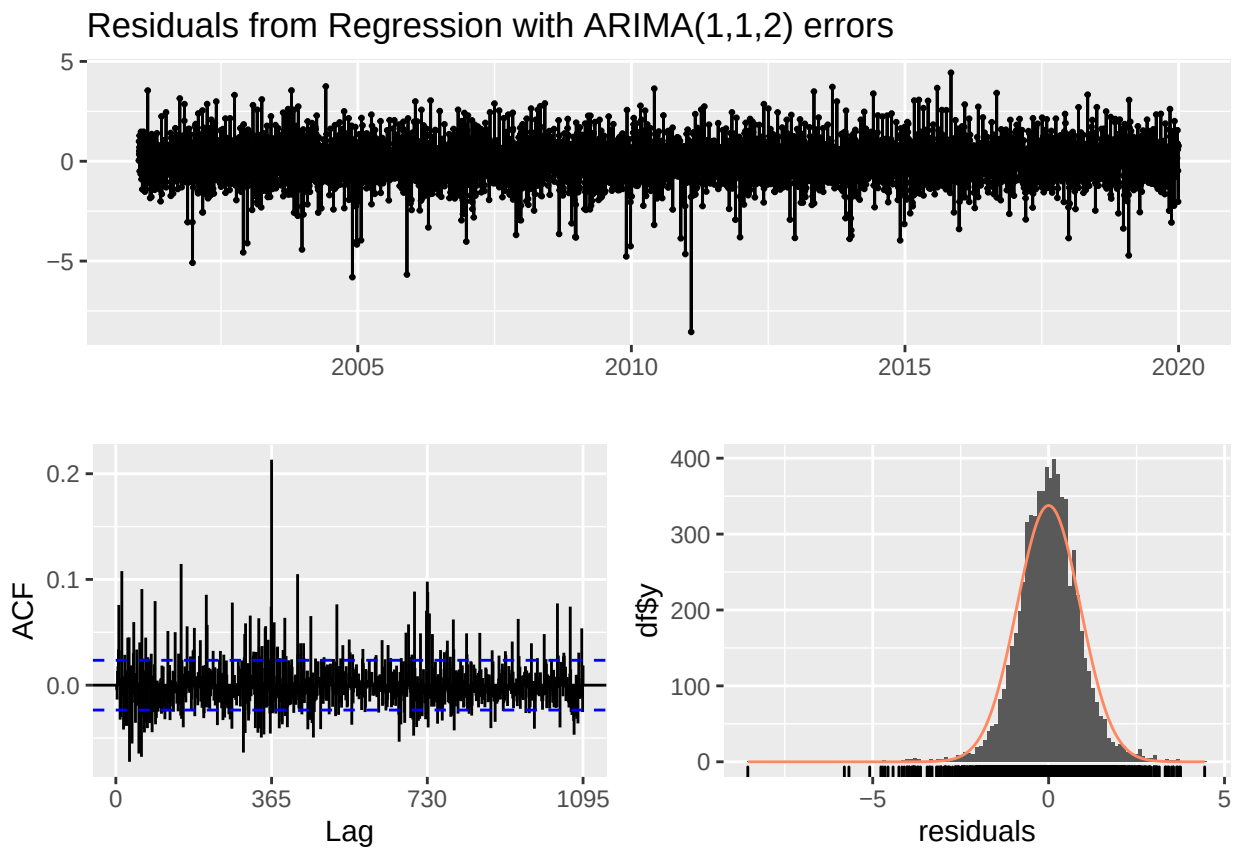
```
autoplot(crimesArimaMS)
```




```
autoplot(crimes01_23[,2],
         ylab = 'Number of Crimes',
         main = 'Chicago Crimes, Arima(1,1,2) MS') +
    autolayer(foreArimaMS$mean,
              series = 'Arima(1,1,2) MS',
              alpha = 0.65,
              color = 'red')
```




```
residArimaMS <- checkresiduals(crimesArimaMS)
```



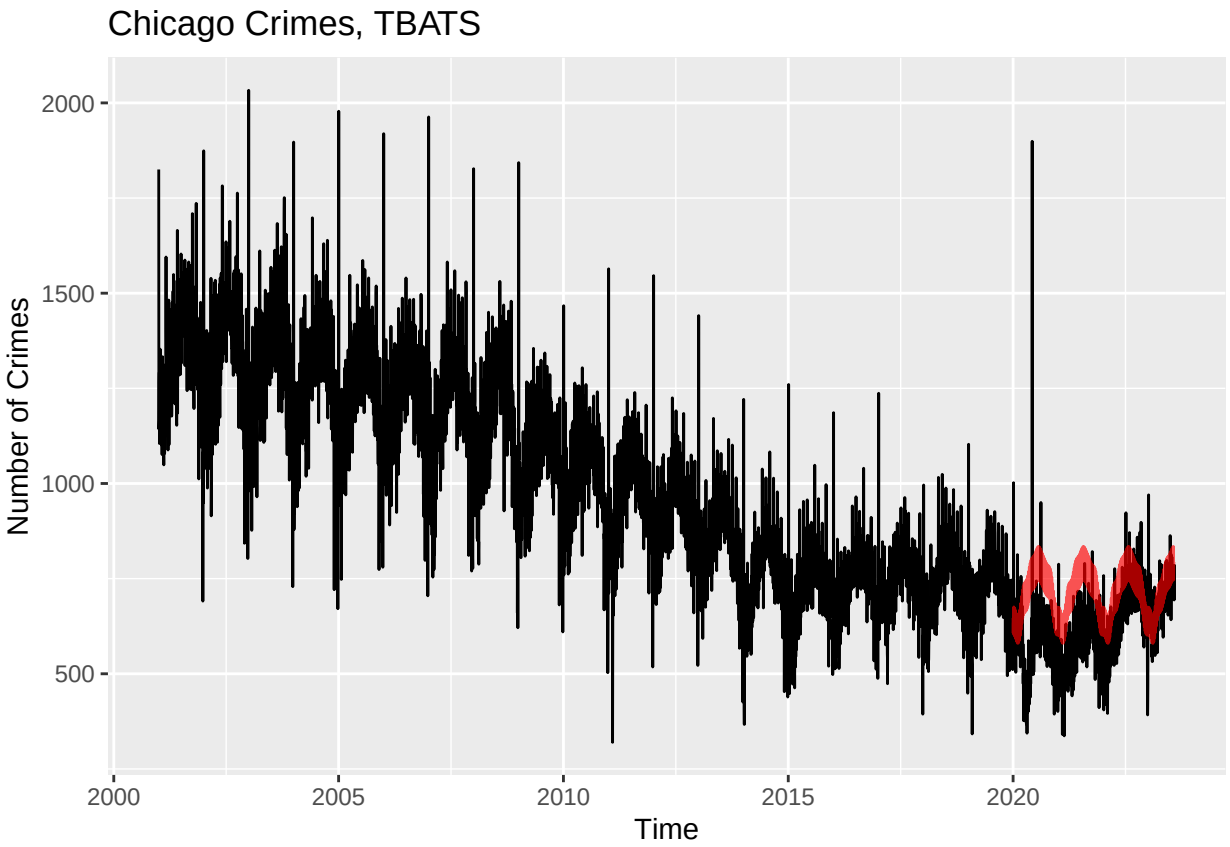

```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(1,1,2) errors
## Q* = 3475.4, df = 727, p-value < 2.2e-16
##
## Model df: 3.   Total lags used: 730
```

TBATS model:

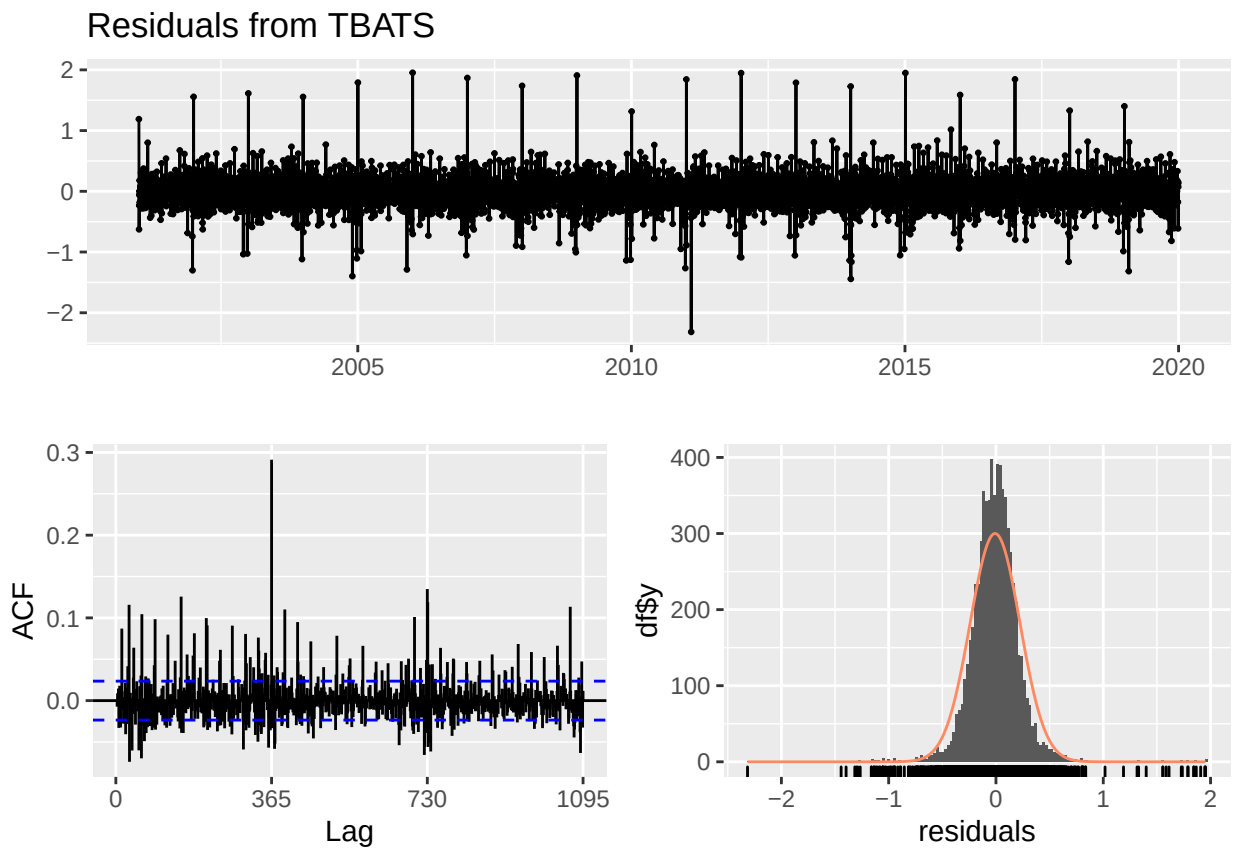
```
crimesTbats <- tbats(crimes01_19[,2],
                     use.trend = TRUE,
                     seasonal.periods = c(7, 365),
                     use.box.cox = TRUE)

foreTbats <- forecast(crimesTbats,
                      h=365*3+212)

crimesTbats
```

```
## TBATS(0.151, {3,1}, 0.879, {<7,3>, <365,6>})
##
## Call: tbats(y = crimes01_19[, 2], use.box.cox = TRUE, use.trend = TRUE,
##     seasonal.periods = c(7, 365))
##
## Parameters
##   Lambda: 0.150509
##   Alpha: 0.7365125
##   Beta: -0.08763729
##   Damping Parameter: 0.879261
##   Gamma-1 Values: -3.337665e-05 -0.0003091495
##   Gamma-2 Values: 7.091915e-05 -1.331799e-05
##   AR coefficients: 0.521586 -0.015786 -0.027453
##   MA coefficients: -0.844303
##
## Seed States:
##               [,1]
##  [1,] 13.082466244
##  [2,] -0.021887889
##  [3,] -0.075418431
##  [4,]  0.018617912
##  [5,]  0.023881708
##  [6,]  0.016346386
##  [7,]  0.076424862
##  [8,] -0.028139758
##  [9,] -0.272984199
## [10,] -0.035926226
## [11,] -0.015495065
## [12,]  0.008319564
## [13,]  0.023520147
## [14,]  0.001824881
## [15,] -0.123742889
## [16,] -0.034627888
## [17,] -0.016405082
## [18,]  0.012963663
## [19,]  0.019322134
## [20,]  0.018322765
## [21,]  0.000000000
## [22,]  0.000000000
## [23,]  0.000000000
## [24,]  0.000000000
## attr(,"lambda")
## [1] 0.1505087
##
## Sigma: 0.2330789
## AIC: 122385.6
```

```
autoplot(crimes01_23[,2],
         ylab = 'Number of Crimes',
         main = 'Chicago Crimes, TBATS') +
    autolayer(foreTbats$mean,
              series = 'TBATS',
              alpha = 0.65,
              color = 'red')
```




```
residTbats <- checkresiduals(crimesTbats)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from TBATS
## Q* = 4159.2, df = 730, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 730
```

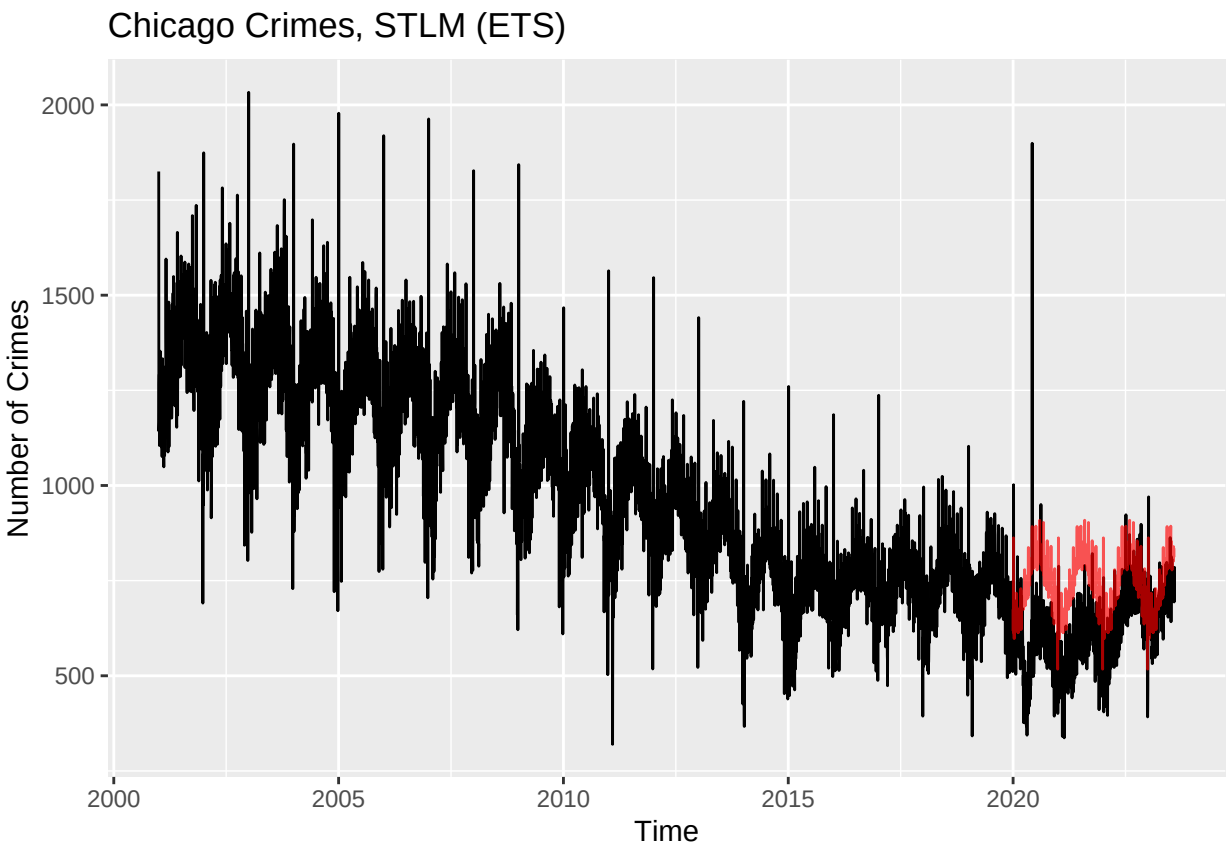
ETS model:

```
crimesETS <- stlm(crimes01_19[,2],
                  method = 'ets',
                  allow.multiplicative.trend = TRUE,
                  biasadj = FALSE,
                  lambda = lambdaCrimes)

foreETS <- forecast(crimesETS,
                    biasadj = FALSE,
                    lambda = lambdaCrimes,
```

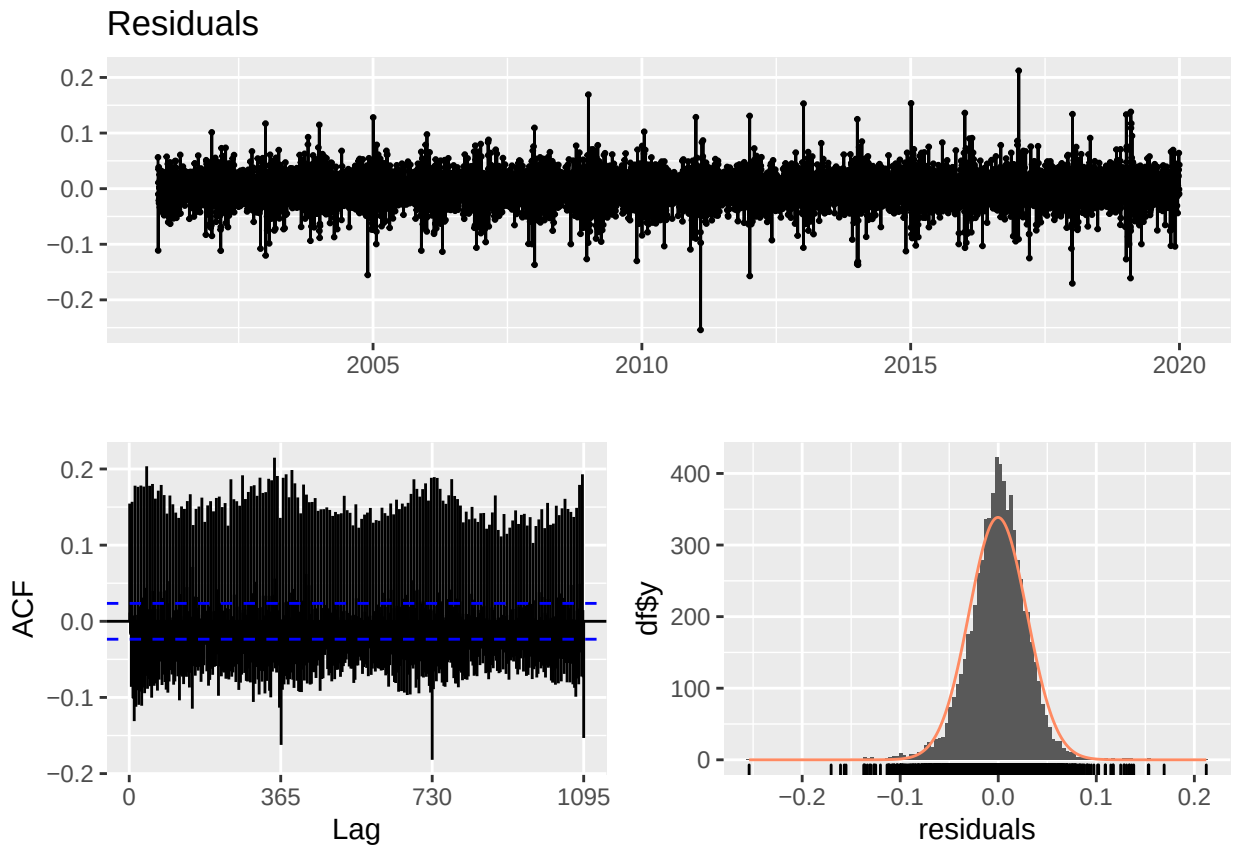
```
                    h=365*3+212)

crimesETS$model
```

```
## ETS(M,N,N)
##
## Call:
##  ets(y = x, model = etsmodel, allow.multiplicative.trend = allow.multiplicative.trend)
##
##   Smoothing parameters:
##     alpha = 0.1026
##
##   Initial states:
##     l = 34.73
##
##   sigma:  0.0295
##
##      AIC     AICc      BIC
## 59991.59 59991.59 60012.12
```

```
autoplot(crimes01_23[,2],
         ylab = 'Number of Crimes',
         main = 'Chicago Crimes, STLM (ETS)') +
    autolayer(foreETS$mean,
              series = 'STLM (ETS)',
              alpha = 0.65,
              color = 'red')
```




```
residETS <- checkresiduals(crimesETS)
```




```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 30432, df = 730, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 730
```

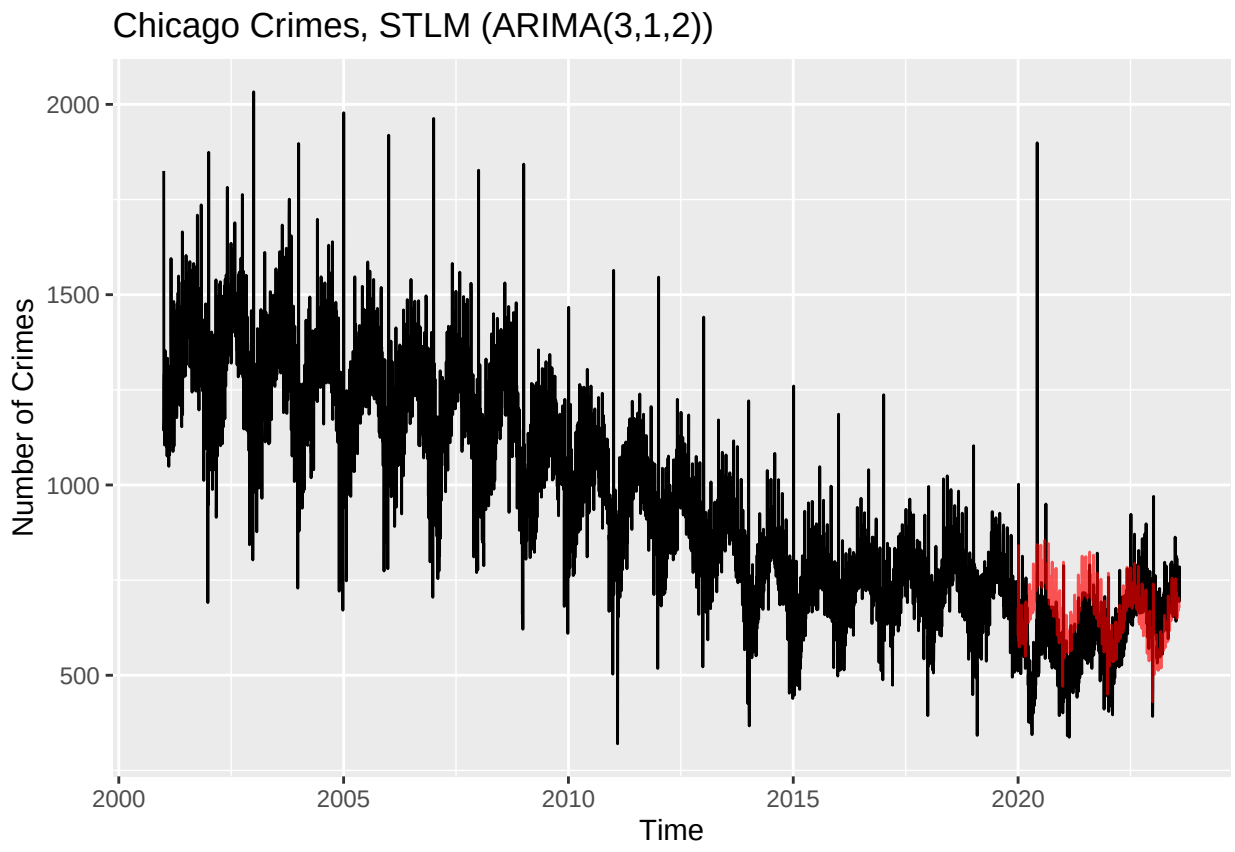
STLM model (just yearly seasonality):

```
crimesSTLM <- stlm(crimes01_19[,2],
                   method = 'arima',
                   xreg = crimes01_19[,3],
                   biasadj = FALSE,
                   lambda = lambdaCrimes)

foreSTLM <- forecast(crimesSTLM,
                     biasadj = FALSE,
                     xreg = crimes20_23[,3],
```

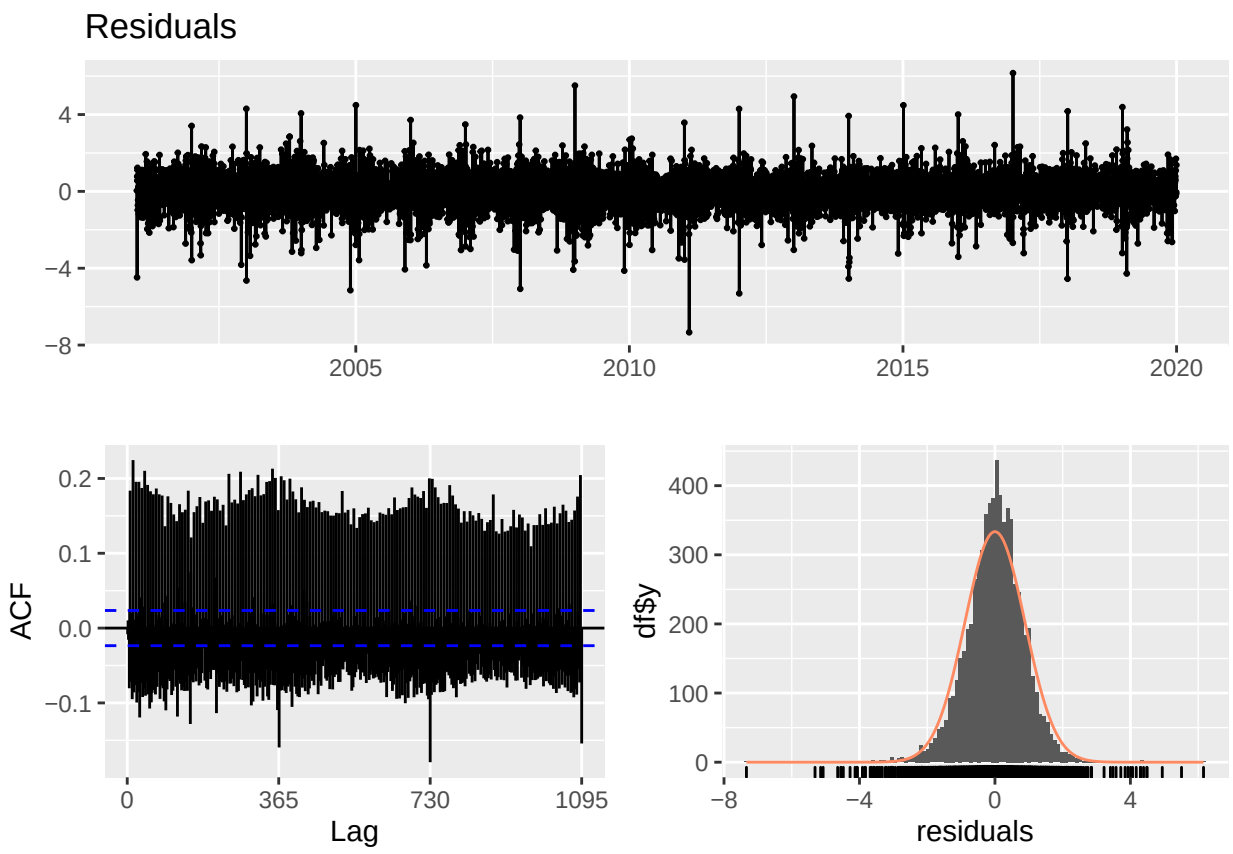
```
                    lambda = lambdaCrimes)

crimesSTLM$model
```

```
## Series: x
## Regression with ARIMA(3,1,2) errors
##
## Coefficients:
##          ar1      ar2     ar3      ma1     ma2    drift     xreg
##       1.0152  -0.1653  0.0287  -1.7650  0.7679  -0.0011  -0.7115
## s.e.  0.0295   0.0180  0.0130   0.0268  0.0266   0.0003   0.4291
##
## sigma^2 = 0.7904:  log likelihood = -9021.14
## AIC=18058.28   AICc=18058.3   BIC=18113.03
```

```
autoplot(crimes01_23[,2],
         ylab = 'Number of Crimes',
         main = 'Chicago Crimes, STLM (ARIMA(3,1,2))') +
    autolayer(foreSTLM$mean,
              series = 'STLM (ARIMA)',
              alpha = 0.65,
              color = 'red')
```



```
residSTLM <- checkresiduals(crimesSTLM)
```




```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 32343, df = 730, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 730
```

STLM model (yearly seasonality + weekly seasonality):

```
crimesSTLM_MS <- stlm(crimes01_19MS[,2],
                      method = 'arima',
                      lambda = lambdaCrimes,
                      biasadj = FALSE,
                      xreg = cbind(crimes01_19MS[,3],
                                   crimes01_19MS[,4],
                                   fourier(crimes01_19MS[,2],
```

```
                                          K=c(3,3))))

foreSTLM_MS <- forecast(crimesSTLM_MS,
                        lambda = lambdaCrimes,
                        biasadj = FALSE,
                        xreg = cbind(crimes20_23MS[,3],
                                     crimes20_23MS[,4],
                                     fourier(crimes20_23MS[,2],
                                             K=c(3,3))))
```

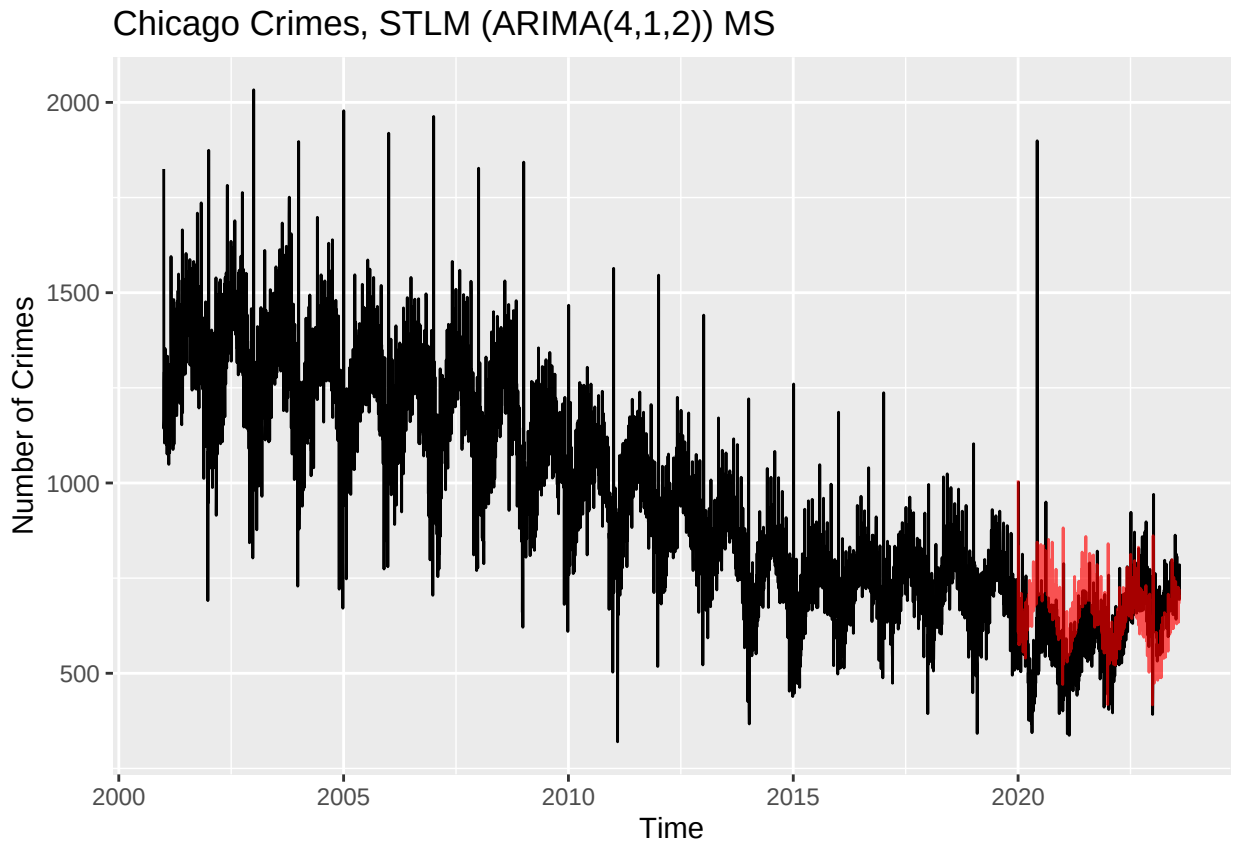
```
## Warning in forecast.forecast_ARIMA(object$model, h = h, level = level, ...):
## xreg contains different column names from the xreg used in training. Please
## check that the regressors are in the same order.
```

```
crimesSTLM_MS$model
```

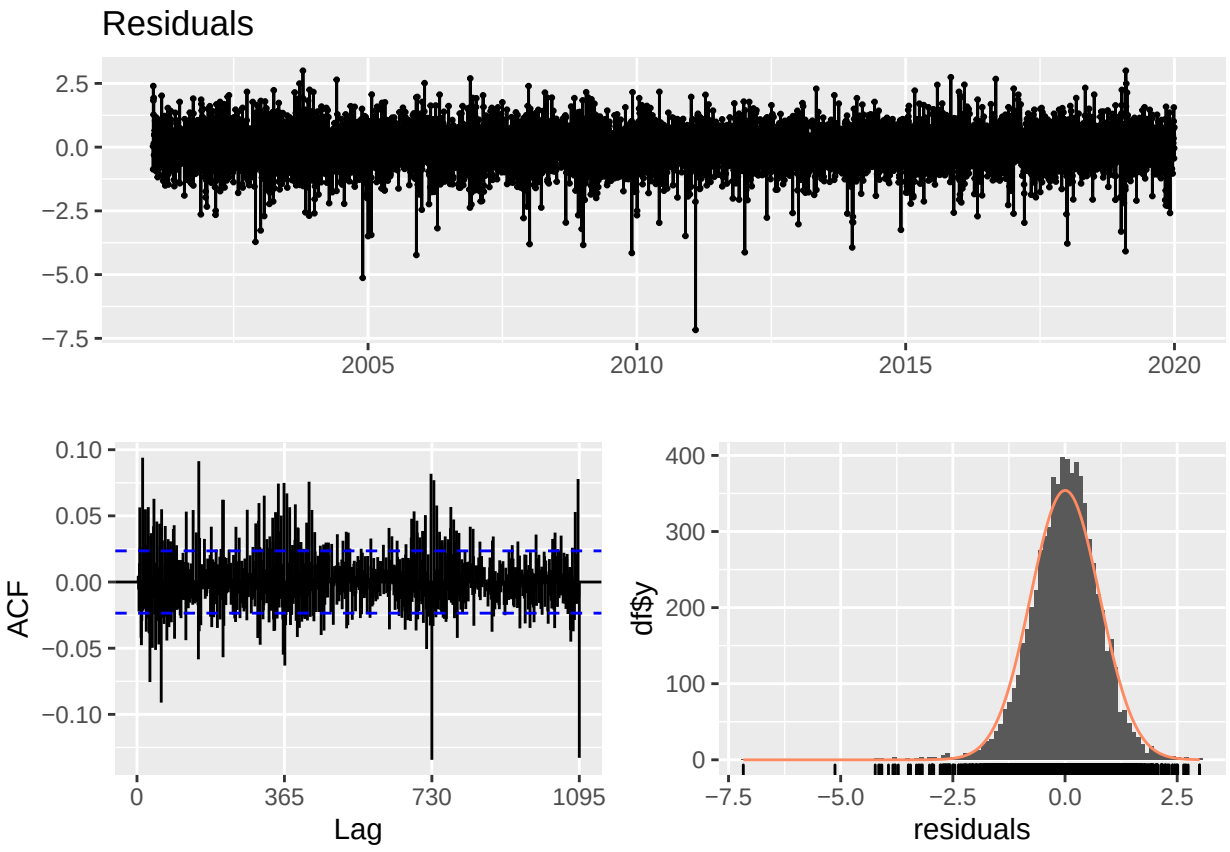
```
## Series: x
## Regression with ARIMA(4,1,2) errors
##
## Coefficients:
##          ar1      ar2      ar3     ar4      ma1     ma2    drift
##       0.9969  -0.1495  -0.0555  0.0493  -1.6534  0.6577  -0.0011
## s.e.  0.0506   0.0239   0.0173  0.0128   0.0494  0.0488   0.0003
##       crimes01_19MS[, 3]  crimes01_19MS[, 4]
##                  -0.8606              4.7384
## s.e.              0.3633              0.1725
##       fourier(crimes01_19MS[, 2], K = c(3, 3)).S1-7
##                                           -0.1873
## s.e.                                       0.0150
##       fourier(crimes01_19MS[, 2], K = c(3, 3)).C1-7
##                                           -0.2441
## s.e.                                       0.0150
##       fourier(crimes01_19MS[, 2], K = c(3, 3)).S2-7
##                                            0.0015
## s.e.                                       0.0110
##       fourier(crimes01_19MS[, 2], K = c(3, 3)).C2-7
##                                           -0.3413
## s.e.                                       0.0110
##       fourier(crimes01_19MS[, 2], K = c(3, 3)).S3-7
##                                            0.1579
## s.e.                                       0.0104
##       fourier(crimes01_19MS[, 2], K = c(3, 3)).C3-7
##                                           -0.0390
## s.e.                                       0.0104
##       fourier(crimes01_19MS[, 2], K = c(3, 3)).S1-365
##                                             -0.0012
## s.e.                                         0.0345
##       fourier(crimes01_19MS[, 2], K = c(3, 3)).C1-365
##                                             -0.0060
## s.e.                                         0.0343
##       fourier(crimes01_19MS[, 2], K = c(3, 3)).S2-365
##                                              0.0089
## s.e.                                         0.0290
```

```
##       fourier(crimes01_19MS[, 2], K = c(3, 3)).C2-365
##                                              -0.0120
## s.e.                                          0.0289
##       fourier(crimes01_19MS[, 2], K = c(3, 3)).S3-365
##                                              -0.0003
## s.e.                                          0.0276
##       fourier(crimes01_19MS[, 2], K = c(3, 3)).C3-365
##                                              -0.0205
## s.e.                                          0.0275
##
## sigma^2 = 0.5972:  log likelihood = -8042.26
## AIC=16128.51   AICc=16128.66   BIC=16279.08
```

```
autoplot(crimes01_23[,2],
         ylab = 'Number of Crimes',
         main = 'Chicago Crimes, STLM (ARIMA(4,1,2)) MS') +
    autolayer(foreSTLM_MS$mean,
              series = 'STLM (ARIMA) MS',
              alpha = 0.65,
              color = 'red')
```




```
residSTLM_MS <- checkresiduals(crimesSTLM_MS)
```




```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 2962.4, df = 730, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 730
```

Table comparing the different models:

```
errorStats <- data.frame(rbind(round(c(accuracy(crimesArima),
                                       residArima$statistic,
                                       crimesArima$aic,
                                       crimesArima$loglik), 5),
                               round(c(accuracy(crimesArimaMS),
                                       residArimaMS$statistic,
                                       crimesArimaMS$aic,
                                       crimesArimaMS$loglik), 5),
                               round(c(accuracy(crimesTbats),
```

```
                                        residTbats$statistic,
                                        crimesTbats$AIC,
                                        crimesTbats$likelihood/-2), 5),
                                round(c(accuracy(crimesETS),
                                        residETS$statistic,
                                        crimesETS$model$aic,
                                        crimesETS$model$loglik), 5),
                                round(c(accuracy(crimesSTLM),
                                        residSTLM$statistic,
                                        crimesSTLM$model$aic,
                                        crimesSTLM$model$loglik), 5),
                                round(c(accuracy(crimesSTLM_MS),
                                        residSTLM_MS$statistic,
                                        crimesSTLM_MS$model$aic,
                                        crimesSTLM_MS$model$loglik), 5)),
                            row.names = c('ARIMA',
                                          'ARIMA (MS)',
                                          'TBATS',
                                          'STLM ETS',
                                          'STLM ARIMA',
                                          'STLM ARIMA (MS)'))

colnames(errorStats) <- c(colnames(accuracy(crimesArima)),
                          'Ljung-Box Stat. Q*',
                          'AIC',
                          'LogLik')

print(errorStats)
```

```
##                       ME      RMSE      MAE      MPE    MAPE    MASE     ACF1
## ARIMA           -0.49086 101.99963 73.57240 -0.56507 7.63688 0.81501 -0.02287
## ARIMA (MS)       1.72864  73.88187 54.95810 -0.39138 5.65113 0.60881 -0.02968
## TBATS           -0.22185  83.30707 56.70070 -0.70080 5.80230 0.73568 -0.04527
## STLM ETS         0.96380  75.52534 55.72516 -0.48881 5.68963 0.61731  0.12180
## STLM ARIMA       1.83725  74.22028 54.29271 -0.39310 5.52409 0.60144 -0.03785
## STLM ARIMA (MS)  1.49610  63.31815 47.97048 -0.27227 4.92584 0.53140 -0.02181
##                  Ljung-Box Stat. Q*       AIC     LogLik
## ARIMA                     28514.825  21912.55 -10948.275
## ARIMA (MS)                 3475.368  18284.32  -9123.158
## TBATS                      4159.199 122385.64 -61156.818
## STLM ETS                  30431.985  59991.59 -29992.793
## STLM ARIMA                32342.925  18058.28  -9021.140
## STLM ARIMA (MS)            2962.435  16128.51  -8042.256
```

By this table, the STLM ARIMA (MS) model seems to perform the best. This model has the lowest ME, RMSE, MAE, MASE, AIC, and Log Likelihood values and the smallest absolute values for MPE and ACF1. Along with this, this model has the smallest test statistic among all models for the Ljung-Box test. By this, though the residuals of the model still show some correlation, this model provides the best residuals. By this, a model using STLM ARIMA with multiple seasonality provides the most accurate model.

Testing dataset accuracy:

```
# crimesArima, crimesArimaMS, crimesTbats, crimesETS, crimesSTLM, crimesSTLM_MS

data.frame(rbind(round(accuracy(crimes01_23[,2],
                                foreArima$mean), 5),
                 round(accuracy(crimes01_23[,2],
                                foreArimaMS$mean), 5),
                 round(accuracy(crimes01_23[,2],
                                foreTbats$mean), 5),
                 round(accuracy(crimes01_23[,2],
                                foreETS$mean), 5),
                 round(accuracy(crimes01_23[,2],
                                foreSTLM$mean), 5),
                 round(accuracy(crimes01_23[,2],
                                foreSTLM_MS$mean), 5)),
           row.names = c('ARIMA',
                         'ARIMA (MS)',
                         'TBATS',
                         'STLM ETS',
                         'STLM ARIMA',
                         'STLM ARIMA (MS)')) %>%
    print()
```

```
##                        ME     RMSE       MAE      MPE     MAPE    ACF1
## ARIMA            41.48780 131.8577 104.89534  4.78084 15.98963 0.64313
## ARIMA (MS)       47.62028 131.5493 101.86845  6.14799 15.11555 0.70782
## TBATS            93.51371 137.2320 110.63691 12.92156 15.49224 0.70563
## STLM ETS        120.21212 157.0351 131.90340 16.03000 17.70653 0.66145
## STLM ARIMA       41.26051 121.4710  96.76225  5.49729 14.58298 0.74179
## STLM ARIMA (MS)  41.98952 125.6887  99.44759  5.43122 14.94995 0.70164
##                  Theil.s.U
## ARIMA              1.69956
## ARIMA (MS)         2.66399
## TBATS              4.64720
## STLM ETS           4.88176
## STLM ARIMA         3.99479
## STLM ARIMA (MS)    2.67309
```

After calculating the forecasts of each model from 2020 to 2023, the forecasts were used to calculate the accuracy of the model using the test data (actual 2020 to 2023 crime values).

From these calculations, the STLM ARIMA had the best ME, RMSE, MAE, MAPE, and ACF1 values. ARIMA had the best MPE value and the best Theil.s.U value.

By this, STLM ARIMA seemed to give the most accurate predictions when testing. STLM ARIMA (MS) also gave relatively accurate predictions.

STLM ETS and TBATS performed the worst of all the models.

From the above observations, the STLM ARIMA model with multiple seasonality gave the best overall results, and that model will be used for time series predictions for crime counts in Chicago.