Group 4 Project - LA crime

Group 4

November 22, 2020

NOTE: THIS CODE CONSISTS OF BOTH MAIN CODE ALONG WITH THE SHINY APP IN 3 PARTS

MAIN CODE - PART A (Crime only) | PART B (External Variables) | PART C (Shiny APP)

To run the code successfully add 'Monthly data','GDP_Jan2010_Aug2020',

#'RentCPI_Jan2010_Aug2020','Temperature_Jan2010_Aug2020','Unemployment_Jan2010_Aug2020' excel files in the same folder of this # RMD file and select the go to the menu options and selesct 'Session' -> 'Set Working Directory' -> 'To source file location'. Then go to top right of this window and Under 'Run' select 'Run All' or use shortcult Cntrl+Alt+R.

The Shiny App gives a dashboard that shows the crime and external variable time series and MAPE 2020 results based on training till 2019. A user can also select a time range to focus on using the slider provided at the bottom left.

Introduction to the problem

Lately, there has been a spike in the number of crime alert warnings that have been issued by USC DPS. Also, with the 2020 elections being around the corner, it has been noticed that the Republican party and President Trump's campaign has been using the "law and order" message to attack their political rival, claiming that cities and states governed by Democrats experience constant security threats. In addition, in the midst of COVID-19 pandemic and the national economic distress, as well as the recent nation-wide protests stimulated by police brutality towards African Americans, more cases of violent crimes have been reported in news than in ordinary times. However, politicians' campaign tactics and our personal perceptions of crime might be misleading and give a biased view. We want to use this final project opportunity to identify the trends present in crime occurrences. Therefore, we are planning to investigate Los Angeles crime data as a time series to help us understand the pattern of crime occurrences and its influencing factors.

Objectives and Motivation 1. Understand the various components of LA crime time series data. 2. Understand the influencing factors of LA crime occurrences. 3. Provide a one-step forward forecast of LA crime occurrences.

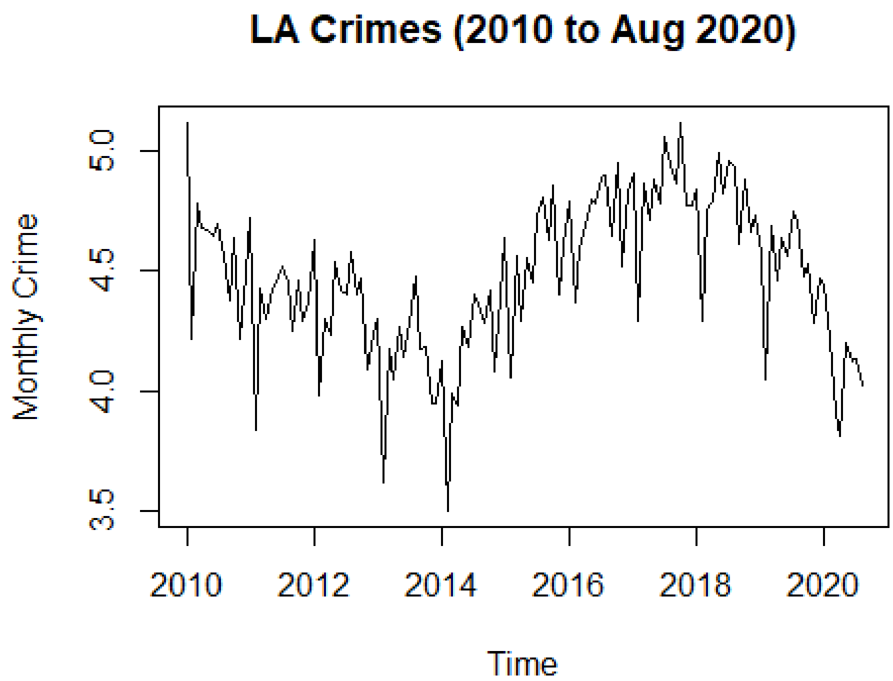
Questions to Investigate 1. What is the trend of LA crime occurrences? 2. Is there any seasonality in LA crime occurrences? 3. How do economic factors influence LA crime occurrences? 4. How does population change influence LA crime occurrences? 5. How does external variables like homelessness influence LA crime occurrences? 6. Does climate/weather influence crime occurrences? 7. How does COVID-19 pandemic influence LA crime occurrences? 8. What are some of the other possible factors?

— PART A

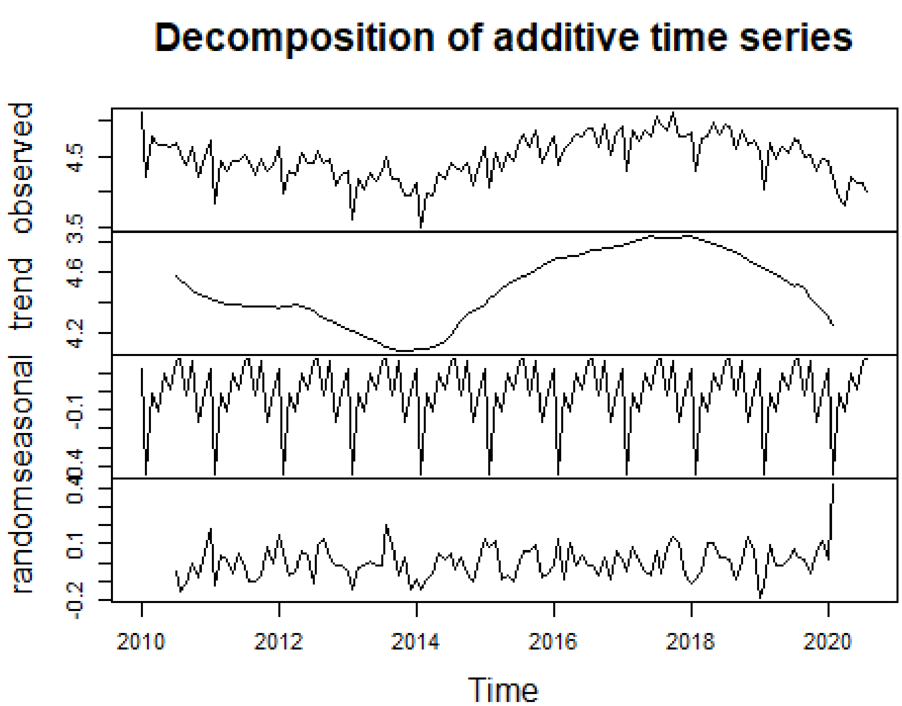
FITTING MODELS ON CRIME DATA

```
# Loading relevant packages
library(forecast)

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

library(ggplot2)
library(zoo)

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

library(readxl)
library(tseries)
library(astsa)

##
## Attaching package: 'astsa'

## The following object is masked from 'package:forecast':
##
##     gas

library(tree)

## Warning: package 'tree' was built under R version 4.0.3

library(cvTools)

## Warning: package 'cvTools' was built under R version 4.0.3
```

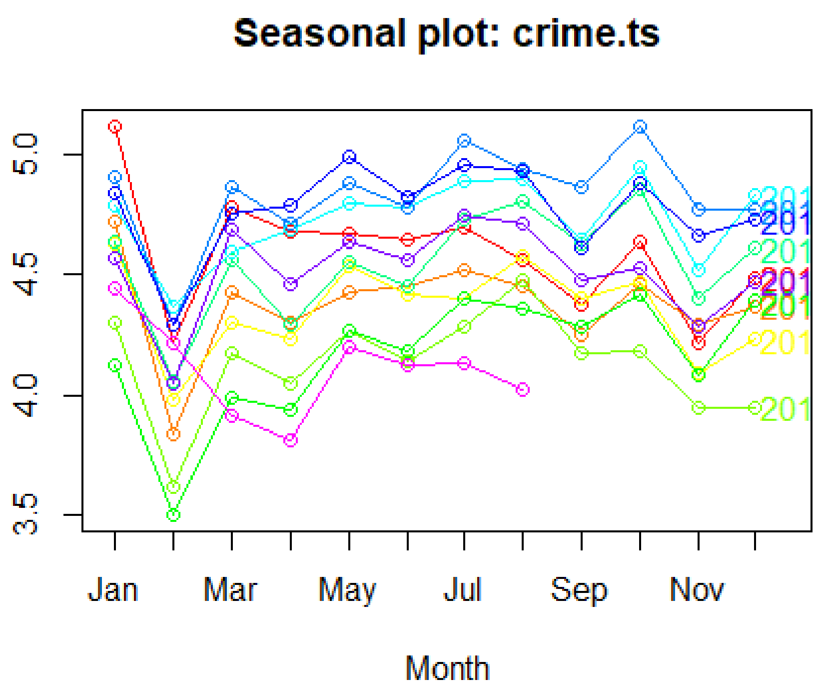
```r
## Loading required package: lattice

## Loading required package: robustbase

## Warning: package 'robustbase' was built under R version 4.0.3

# Reading data
crime <- read_excel("Monthly data.xlsx")

head(crime)

## Registered S3 method overwritten by 'cli':
##   method     from
##   print.tree tree

## # A tibble: 6 x 4
##   Time         Crime `Population (in 1000s)` `Crime/Population`
##   <chr>        <dbl>                   <dbl>              <dbl>
## 1 2010 Month 1 19433                    3795               5.12
## 2 2010 Month 2 16016                    3795               4.22
## 3 2010 Month 3 18125                    3795               4.78
## 4 2010 Month 4 17766                    3795               4.68
## 5 2010 Month 5 17713                    3795               4.67
## 6 2010 Month 6 17662                    3795               4.65

tail(crime)

## # A tibble: 6 x 4
##   Time         Crime `Population (in 1000s)` `Crime/Population`
##   <chr>        <dbl>                   <dbl>              <dbl>
## 1 2020 Month 3 15684                    4012               3.91
## 2 2020 Month 4 15301                    4012               3.81
## 3 2020 Month 5 16856                    4012               4.2
## 4 2020 Month 6 16534                    4012               4.12
## 5 2020 Month 7 16587                    4012               4.13
## 6 2020 Month 8 16131                    4012               4.02

# Converting data to time series object
crime.ts <- ts(crime$`Crime/Population`, start = c(2010,1), freq = 12)

# Basic plot
par(mfrow = c(1,1))
plot(crime.ts, xlab = "Time", ylab = "Monthly Crime", main = "LA Crimes (2010
to Aug 2020)")
```




```r
# Looking for trend and seasonality
plot(decompose(crime.ts))
```



```r
# Looking for seasonality pattern
seasonplot(crime.ts, col=rainbow(12), year.labels=TRUE)
```



There is a potential polynomial trend present in the data along with a strong annual seasonality. Based on the season plot it could be seen that summer months show higher crime rates compared to other months.

```r
# splitting data into train and test
train.ts <- window(crime.ts,end = c(2018,12)) # Till 2018
test.ts  <- window(crime.ts,start = c(2019,1),end = c(2019,12)) # 2019 year
valid.ts <- window(crime.ts, start = c(2020,1)) # 2020 till August
traintest.ts <- window(crime.ts,end = c(2019,12)) # Till 2019
```

*** BUILDING MODELS***

Model1 - Fitting Linear Trend

```r
# Fitting linear trend
lr_trend <- tslm(train.ts~trend)

# Model performance
summary(lr_trend)

##
## Call:
## tslm(formula = train.ts ~ trend)
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -0.97916 -0.16738  0.03127  0.19462  0.86477 
## 
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 4.2506646  0.0560615  75.821  < 2e-16 ***
## trend       0.0045699  0.0008929   5.118 1.39e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.2893 on 106 degrees of freedom
## Multiple R-squared:  0.1982, Adjusted R-squared:  0.1906 
## F-statistic: 26.19 on 1 and 106 DF,  p-value: 1.386e-06
```

```
# Train, test and validation accuracy
accuracy(lr_trend$fitted.values,train.ts) # train accuracy
```

```
##                    ME      RMSE       MAE        MPE     MAPE      ACF1
## Test set 1.642835e-17 0.2865923 0.2237848 -0.4307351 5.105621 0.4062563
##          Theil's U
## Test set 0.9274398
```

```
accuracy(forecast(lr_trend, h=12)$mean,test.ts) # test accuracy
```

```
##                  ME      RMSE       MAE       MPE     MAPE       ACF1
Theil's U
## Test set -0.2580808 0.3192631 0.2580808 -5.906454 5.906454 -0.1511182
1.0572
```

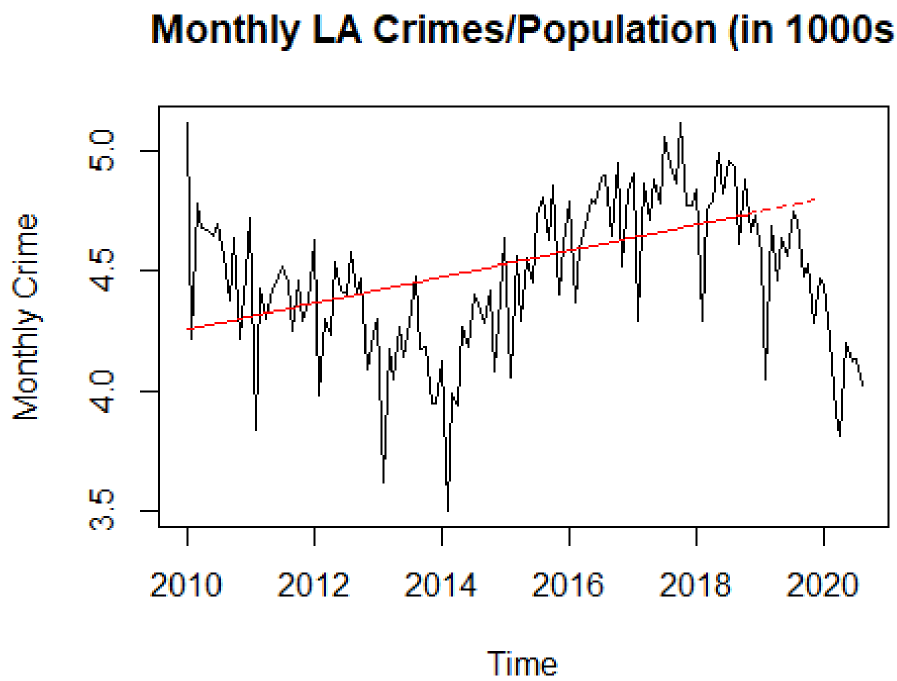
```
accuracy(forecast(lr_trend, h=24)$mean,valid.ts) # 2019 accuracy (train till
2018)
```

```
##                  ME      RMSE       MAE       MPE     MAPE      ACF1
Theil's U
## Test set -0.7133628 0.7374268 0.7133628 -17.60907 17.60907 0.1729896
3.548223
```

```
accuracy(forecast(tslm(traintest.ts~trend), h=12)$mean,valid.ts) # 2019
accuracy (train till 2019)
```

```
##                  ME      RMSE       MAE       MPE     MAPE      ACF1
Theil's U
## Test set -0.6130096 0.6405457 0.6130096 -15.15938 15.15938 0.1720557
3.09773
```

```
# Plotting the linear trend graph
plot(crime.ts, xlab = "Time", ylab = "Monthly Crime", main = "Monthly LA
Crimes/Population (in 1000s")+
lines(lr_trend$fitted.values, col="red")+
lines(forecast(lr_trend, h=12)$mean,col = "red",lty = 2)
```




```
## integer(0)
```

Linear trend gives a poor fit with 0.19 R-squared. The plot shows that changing trend leads to over-prediction for the test period.

Model2 - Fitting Quadratic trend

```
# Fitting quadratic trend
quad_trend <- tslm(train.ts ~ poly(trend, 2, raw=TRUE))

# Model performance
summary(quad_trend)

##
## Call:
## tslm(formula = train.ts ~ poly(trend, 2, raw = TRUE))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.81261 -0.12191  0.02779  0.17893  0.53394
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  4.600e+00  7.245e-02  63.499  < 2e-16 ***
## poly(trend, 2, raw = TRUE)1 -1.451e-02  3.068e-03  -4.728 7.08e-06 ***
## poly(trend, 2, raw = TRUE)2  1.750e-04  2.727e-05   6.418 4.08e-09 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2463 on 105 degrees of freedom
## Multiple R-squared:  0.4241, Adjusted R-squared:  0.4131
## F-statistic: 38.66 on 2 and 105 DF,  p-value: 2.629e-13
```

```
# Train, test and validation accuracy
accuracy(quad_trend$fitted.values,train.ts) # train accuracy
```

```
##                        ME      RMSE       MAE        MPE     MAPE      ACF1
## Test set 1.643219e-17 0.2428889 0.1895828 -0.3133995 4.302202 0.2018948
##          Theil's U
## Test set 0.7820619
```

```
accuracy(forecast(quad_trend, h=12)$mean,test.ts) # test accuracy
```

```
##                    ME      RMSE       MAE       MPE     MAPE        ACF1
## Test set -0.7201092 0.7490983 0.7201092 -16.15402 16.15402 -0.02562482
##          Theil's U
## Test set  2.473015
```

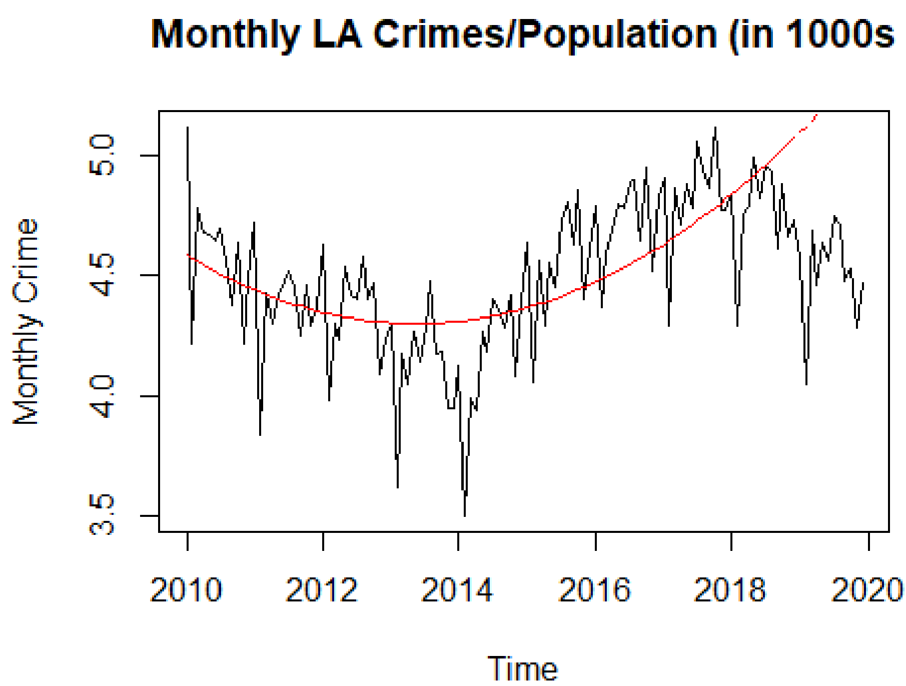
```
accuracy(forecast(quad_trend, h=24)$mean,valid.ts) # 2020 accuracy (train
till 2018)
```

```
##                  ME     RMSE      MAE       MPE     MAPE      ACF1 Theil's
U
## Test set -1.401738 1.418175 1.401738 -34.42544 34.42544 0.2188959
6.714741
```

```
accuracy(forecast(tslm(traintest.ts~poly(trend, 2, raw=TRUE)),
h=12)$mean,valid.ts) # 2020 accuracy (train till 2019)
```

```
##                    ME      RMSE       MAE       MPE     MAPE      ACF1
Theil's U
## Test set -0.7991342 0.8222888 0.7991342 -19.70745 19.70745 0.1814234
3.949495
```

```
# Plotting the quadratic trend graph
plot(traintest.ts, xlab = "Time", ylab = "Monthly Crime", main = "Monthly LA
Crimes/Population (in 1000s")+
lines(quad_trend$fitted.values, col="red")+
lines(forecast(quad_trend, h=12)$mean,col = "red",lty = 2)
```




```
## integer(0)
```

Quadratic model gives a better training accuracy than the linear trend however it performs weaker on the test set as there is a change in the direction which model is unable to predict.

Model3 - Fitting Cubic trend

```
# Fitting cubic trend
cubic_trend <- tslm(train.ts ~ poly(trend, 3, raw=TRUE))

# Model performance
summary(cubic_trend)

##
## Call:
## tslm(formula = train.ts ~ poly(trend, 3, raw = TRUE))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.77814 -0.07583  0.03135  0.12968  0.39774
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  4.895e+00  8.890e-02  55.062  < 2e-16 ***
## poly(trend, 3, raw = TRUE)1 -4.623e-02  7.031e-03  -6.575 1.99e-09 ***
## poly(trend, 3, raw = TRUE)2  8.993e-04  1.495e-04   6.014 2.72e-08 ***
```

```
## poly(trend, 3, raw = TRUE)3 -4.430e-06  9.019e-07  -4.911 3.37e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.223 on 104 degrees of freedom
## Multiple R-squared:  0.5325, Adjusted R-squared:  0.519
## F-statistic: 39.49 on 3 and 104 DF,  p-value: < 2.2e-16
```

```
# Train, test and validation accuracy
accuracy(cubic_trend$fitted.values,train.ts) # train accuracy
```

```
##                    ME      RMSE       MAE        MPE     MAPE       ACF1
## Test set 4.935178e-17 0.2188331 0.1639091 -0.2572882 3.760204 0.05184614
##          Theil's U
## Test set  0.715638
```

```
accuracy(forecast(cubic_trend, h=12)$mean,test.ts) # test accuracy
```

```
##                  ME      RMSE       MAE     MPE     MAPE        ACF1
Theil's U
## Test set -0.2187178 0.2913233 0.2212927 -5.0331 5.087307 -0.08623509
0.9468461
```

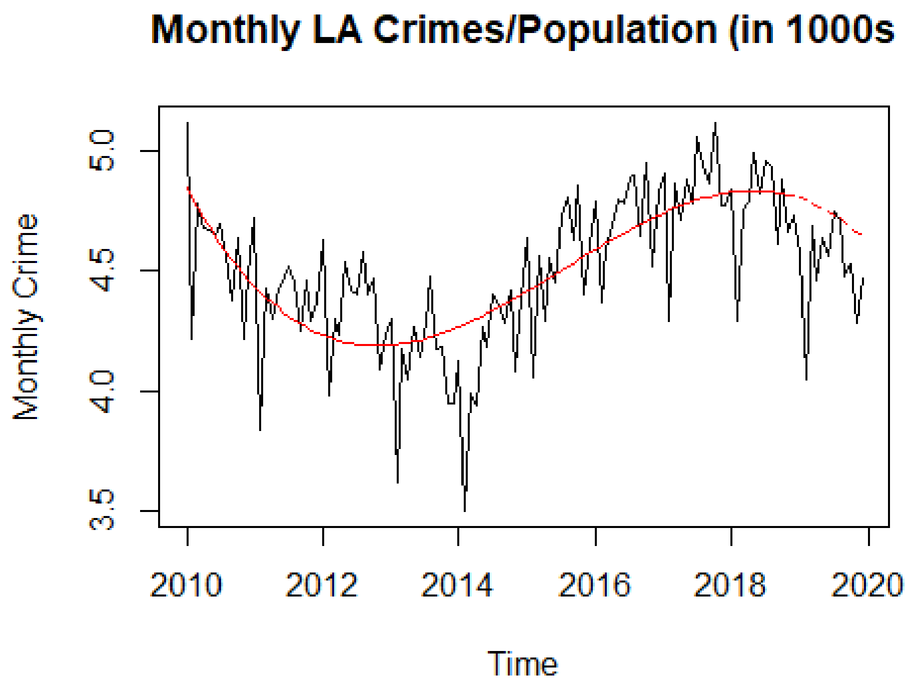
```
accuracy(forecast(cubic_trend, h=24)$mean,valid.ts) # 2020 accuracy (train
till 2018)
```

```
##                  ME      RMSE       MAE       MPE     MAPE      ACF1
Theil's U
## Test set -0.4198857 0.4541193 0.4198857 -10.42355 10.42355 0.2316293
2.199697
```

```
accuracy(forecast(tslm(traintest.ts~poly(trend, 3, raw=TRUE)),
h=12)$mean,valid.ts) # 2020 accuracy (train till 2019)
```

```
##                  ME      RMSE       MAE      MPE     MAPE      ACF1
Theil's U
## Test set -0.1489571 0.2305317 0.1607796 -3.80279 4.069061 0.2956765
1.132647
```

```
# Plotting the cubic trend graph
plot(traintest.ts, xlab = "Time", ylab = "Monthly Crime", main = "Monthly LA
Crimes/Population (in 1000s")+
lines(cubic_trend$fitted.values, col="red")+
lines(forecast(cubic_trend, h=12)$mean,col = "red",lty = 2)
```




```
## integer(0)
```

Cubic model gives a better training and test accuracy than the above model and captures the changing trend better. However, it is still over-predicting the values.

Model4 - Fitting cubic trend with season

```
# Fitting cubic trend with season
cubic_trend_season <- tslm(train.ts ~ poly(trend, 3, raw=TRUE) + season)

# Model performance
summary(cubic_trend_season)

##
## Call:
## tslm(formula = train.ts ~ poly(trend, 3, raw = TRUE) + season)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.311806 -0.065322 -0.000789  0.087050  0.249387
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  5.085e+00  6.523e-02  77.963  < 2e-16 ***
## poly(trend, 3, raw = TRUE)1 -4.813e-02  4.216e-03 -11.417  < 2e-16 ***
## poly(trend, 3, raw = TRUE)2  9.388e-04  8.965e-05  10.471  < 2e-16 ***
```

```
## poly(trend, 3, raw = TRUE)3 -4.667e-06  5.410e-07  -8.626 1.64e-13 ***
## season2                     -6.526e-01  6.217e-02 -10.498  < 2e-16 ***
## season3                     -1.735e-01  6.218e-02  -2.790  0.00639 **
## season4                     -2.581e-01  6.220e-02  -4.150 7.36e-05 ***
## season5                     -6.545e-02  6.224e-02  -1.052  0.29571
## season6                     -1.454e-01  6.228e-02  -2.334  0.02174 *
## season7                     -3.423e-03  6.233e-02  -0.055  0.95633
## season8                      4.848e-03  6.239e-02   0.078  0.93823
## season9                     -1.916e-01  6.245e-02  -3.068  0.00282 **
## season10                     1.599e-03  6.253e-02   0.026  0.97966
## season11                    -3.321e-01  6.262e-02  -5.303 7.65e-07 ***
## season12                    -1.782e-01  6.272e-02  -2.842  0.00551 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1319 on 93 degrees of freedom
## Multiple R-squared:  0.8538, Adjusted R-squared:  0.8318
## F-statistic:  38.8 on 14 and 93 DF,  p-value: < 2.2e-16
```

```
# Train, test and validation accuracy
accuracy(cubic_trend_season$fitted.values,train.ts) # train accuracy
```

```
##                        ME      RMSE        MAE         MPE     MAPE      ACF1
## Test set 4.106227e-18 0.1223644 0.09654151 -0.08096159 2.204365 0.6621193
##          Theil's U
## Test set 0.4179956
```

```
accuracy(forecast(cubic_trend_season, h=12)$mean,test.ts) # test accuracy
```

```
##                   ME      RMSE       MAE       MPE     MAPE       ACF1
Theil's U
## Test set -0.1865122 0.2029199 0.1865122 -4.158649 4.158649 0.05331183
0.5726204
```

```
accuracy(forecast(cubic_trend_season, h=24)$mean,valid.ts) # 2020 accuracy
(train till 2018)
```

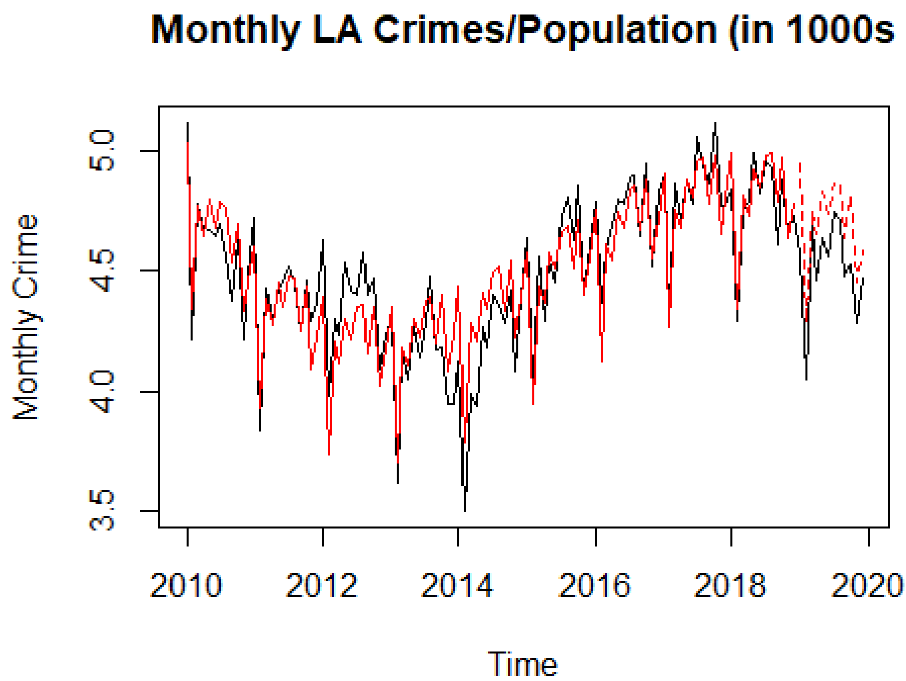
```
##                   ME      RMSE       MAE       MPE    MAPE       ACF1
Theil's U
## Test set -0.3661441 0.4303129 0.4063398 -9.089899 10.0424 -0.0799162
2.046349
```

```
accuracy(forecast(tslm(traintest.ts~poly(trend, 3, raw=TRUE) + season),
h=12)$mean,valid.ts) # 2020 accuracy (train till 2019)
```

```
##                   ME      RMSE       MAE       MPE     MAPE       ACF1
Theil's U
## Test set -0.1178082 0.2513739 0.2150643 -3.029247 5.333894 -0.1210161
1.196915
```

```
# Plotting the cubic trend with season graph
plot(traintest.ts, xlab = "Time", ylab = "Monthly Crime", main = "Monthly LA
```

```
Crimes/Population (in 1000s")+
lines(cubic_trend_season$fitted.values, col="red")+
lines(forecast(cubic_trend_season, h=12)$mean,col = "red",lty = 2)
```




```
## integer(0)
```

Cubic trend with season is able to capture both the polynomial trend as well as season and gives a good performance overall.

Model5 - Naive and Seasonal Naive model

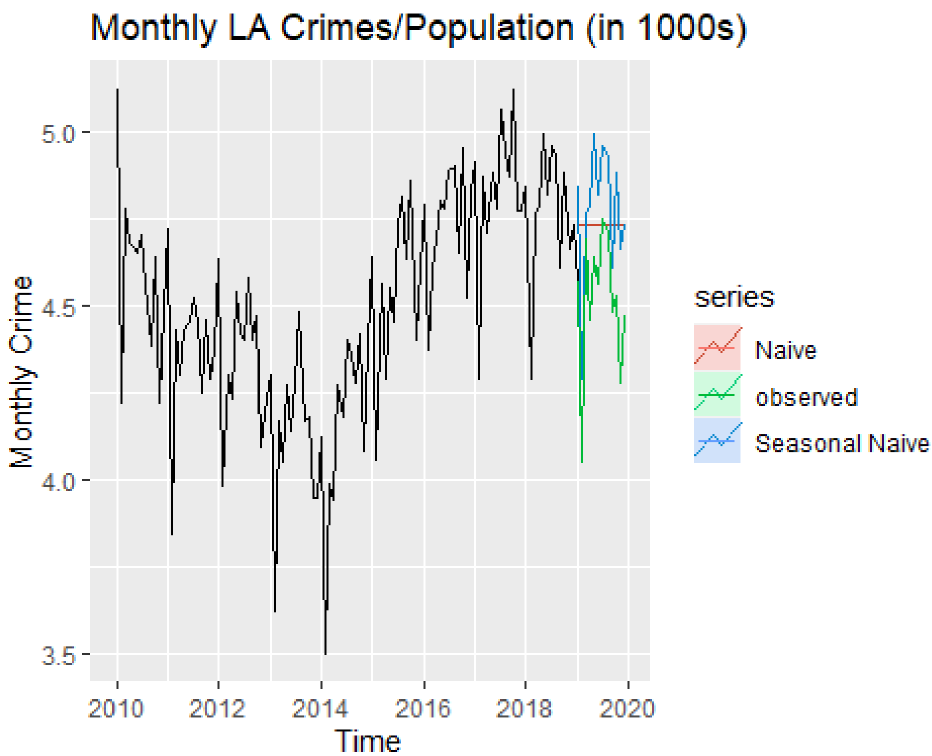
```
# Fitting naive & seasonal naive model
naivemod <- naive(train.ts, h=12)
snaivemod <- snaive(train.ts,h=frequency(train.ts))

# Model performance
summary(naivemod)

##
## Forecast method: Naive method
##
## Model Information:
## Call: naive(y = train.ts, h = 12)
##
## Residual sd: 0.3037
##
## Error measures:
```

```
##                       ME      RMSE       MAE        MPE    MAPE     MASE
## Training set -0.00364486 0.3023259 0.2346729 -0.3185593 5.36765 1.248122
##                    ACF1
## Training set -0.6068029
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Jan 2019           4.73 4.342554 5.117446 4.137452 5.322548
## Feb 2019           4.73 4.182068 5.277932 3.892011 5.567989
## Mar 2019           4.73 4.058923 5.401077 3.703677 5.756323
## Apr 2019           4.73 3.955108 5.504892 3.544904 5.915096
## May 2019           4.73 3.863644 5.596356 3.405023 6.054977
## Jun 2019           4.73 3.780955 5.679045 3.278560 6.181440
## Jul 2019           4.73 3.704914 5.755086 3.162266 6.297734
## Aug 2019           4.73 3.634137 5.825863 3.054022 6.405978
## Sep 2019           4.73 3.567661 5.892339 2.952357 6.507643
## Oct 2019           4.73 3.504788 5.955212 2.856199 6.603801
## Nov 2019           4.73 3.444986 6.015014 2.764741 6.695259
## Dec 2019           4.73 3.387847 6.072153 2.677354 6.782646
```

```
summary(snaivemod)
```

```
##
## Forecast method: Seasonal naive method
##
## Model Information:
## Call: snaive(y = train.ts, h = frequency(train.ts))
##
## Residual sd: 0.2291
##
## Error measures:
##                      ME      RMSE       MAE       MPE     MAPE MASE
ACF1
## Training set 0.02239583 0.2290492 0.1880208 0.3485976 4.237108    1
0.7856895
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Jan 2019           4.84 4.546462 5.133538 4.391072 5.288928
## Feb 2019           4.29 3.996462 4.583538 3.841072 4.738928
## Mar 2019           4.76 4.466462 5.053538 4.311072 5.208928
## Apr 2019           4.79 4.496462 5.083538 4.341072 5.238928
## May 2019           4.99 4.696462 5.283538 4.541072 5.438928
## Jun 2019           4.82 4.526462 5.113538 4.371072 5.268928
## Jul 2019           4.96 4.666462 5.253538 4.511072 5.408928
## Aug 2019           4.93 4.636462 5.223538 4.481072 5.378928
## Sep 2019           4.61 4.316462 4.903538 4.161072 5.058928
## Oct 2019           4.88 4.586462 5.173538 4.431072 5.328928
## Nov 2019           4.66 4.366462 4.953538 4.211072 5.108928
## Dec 2019           4.73 4.436462 5.023538 4.281072 5.178928
```

```
# Train, test and validation accuracy

# Naive
accuracy(naivemod$mean,test.ts) # test accuracy

##                    ME      RMSE    MAE       MPE     MAPE       ACF1 Theil's
U
## Test set -0.2141667 0.2847367 0.2175 -4.932929 5.003105 -0.1460992
0.9413283

accuracy(naive(train.ts, h=20)$mean,valid.ts) # 2020 accuracy (train till
2018)

##                  ME      RMSE     MAE       MPE     MAPE      ACF1 Theil's U
## Test set -0.62375 0.6500096 0.62375 -15.41876 15.41876 0.1703123  3.137791

accuracy(naive(traintest.ts, h=8)$mean,valid.ts) # 2020 accuracy (train till
2019)

##                  ME      RMSE     MAE       MPE     MAPE      ACF1 Theil's U
## Test set -0.36375 0.4071394 0.36375 -9.074391 9.074391 0.1703123  1.995943

# SNaive
accuracy(snaivemod$mean,test.ts) # test accuracy

##                    ME      RMSE       MAE     MPE   MAPE       ACF1 Theil's
U
## Test set -0.2558333 0.2705396 0.2558333 -5.6988 5.6988 0.02834096
0.8713505

accuracy(snaive(train.ts, h=20)$mean,valid.ts) # 2020 accuracy (train till
2018)

##                  ME      RMSE     MAE       MPE     MAPE      ACF1 Theil's U
## Test set -0.69125 0.7482396 0.69125 -17.08277 17.08277 0.2881335  3.624459

accuracy(snaive(traintest.ts, h=8)$mean,valid.ts) # 2020 accuracy (train till
2019)

##                 ME      RMSE  MAE      MPE     MAPE      ACF1 Theil's U
## Test set -0.4475 0.5390269 0.49 -11.1551 12.16221 0.1344829  2.624033

# Plotting the Naive and SNaive graph
autoplot(traintest.ts,ylab = "Monthly Crime", main = "Monthly LA
Crimes/Population (in 1000s)") +
autolayer(naivemod,series = "Naive",PI = FALSE)+
autolayer(snaivemod, series = "Seasonal Naive", PI = FALSE)+
autolayer(test.ts, series = "observed")
```



Naive and SNaive models are over-predicting and performing weak.

Model6 - Trailing Moving Average

```
# Using same forecast for all

# Defining training and test length
nValid = 12

# Fitting moving average (choosing k = 12, to suppress seasonality) for
train.ts and traintest.ts
ma.trailing <- rollmean(train.ts,k =12,align = "right")
ma.trailing.1 <- rollmean(traintest.ts,k =12,align = "right")

# Store the last value
last.ma = tail(ma.trailing,1)
last.ma.1 = tail(ma.trailing.1,1)

# Repeat the last value as predicted value for test period
ma.trailing.pred = ts(rep(last.ma,nValid),start = c(2019,1), end =
c(2019,nValid),frequency = 12)
ma.trailing.pred.1 = ts(rep(last.ma.1,nValid),start = c(2020,1), end =
c(2020,8),frequency = 12)

# Train, test and validation accuracy
accuracy(ma.trailing,train.ts) # train accuracy
```

```
##                   ME      RMSE       MAE        MPE     MAPE       ACF1
Theil's U
## Test set 0.0141323 0.2145219 0.1614003 0.06873065 3.691975 0.07449844
0.7180517
```

```r
accuracy(ma.trailing.pred,test.ts) # test accuracy
```

```
##                    ME     RMSE       MAE       MPE     MAPE       ACF1
Theil's U
## Test set -0.2558333 0.317267 0.2558333 -5.857286 5.857286 -0.1460992
1.045613
```

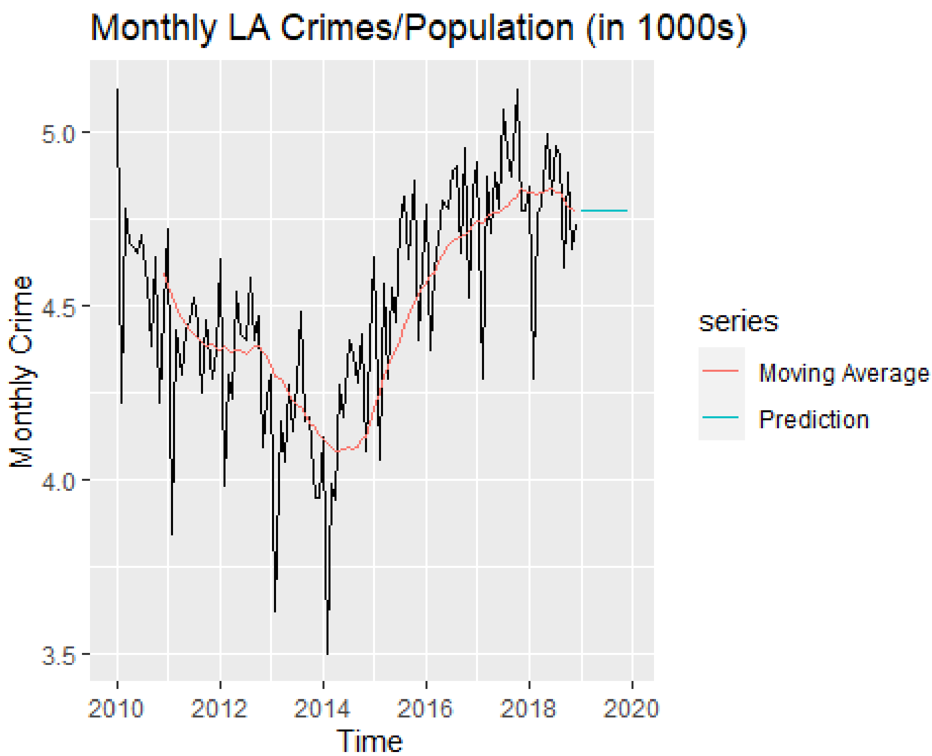
```r
accuracy(ts(rep(last.ma,nValid+8),start = c(2019,1), end =
c(2019,nValid+8),frequency = 12),valid.ts) # 2020 accuracy (train till 2018)
```

```
##                    ME      RMSE       MAE       MPE     MAPE      ACF1
Theil's U
## Test set -0.6654167 0.6900926 0.6654167 -16.43549 16.43549 0.1703123
3.323464
```

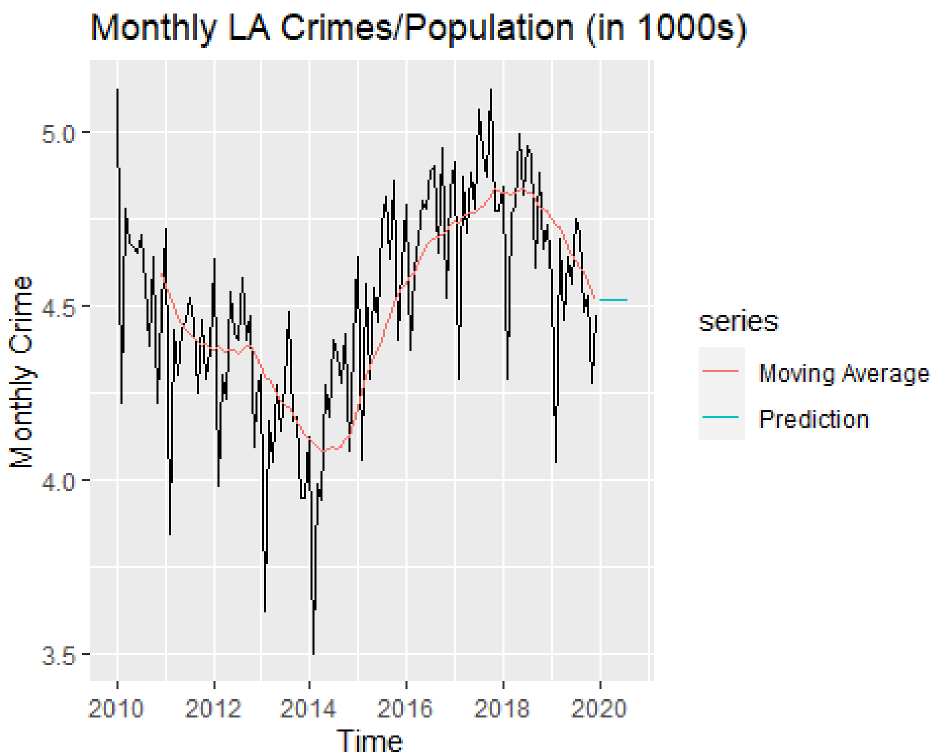
```r
accuracy(ma.trailing.pred.1,valid.ts) # 2020 accuracy (train till 2019)
```

```
##                    ME     RMSE       MAE       MPE     MAPE      ACF1
Theil's U
## Test set -0.4095833 0.448561 0.4095833 -10.19279 10.19279 0.1703123
2.194057
```

```r
# Plotting the moving average graph
autoplot(train.ts,ylab = "Monthly Crime", main = "Monthly LA
Crimes/Population (in 1000s)")+
autolayer(ma.trailing,series = "Moving Average")+
autolayer(ma.trailing.pred,series = "Prediction")
```



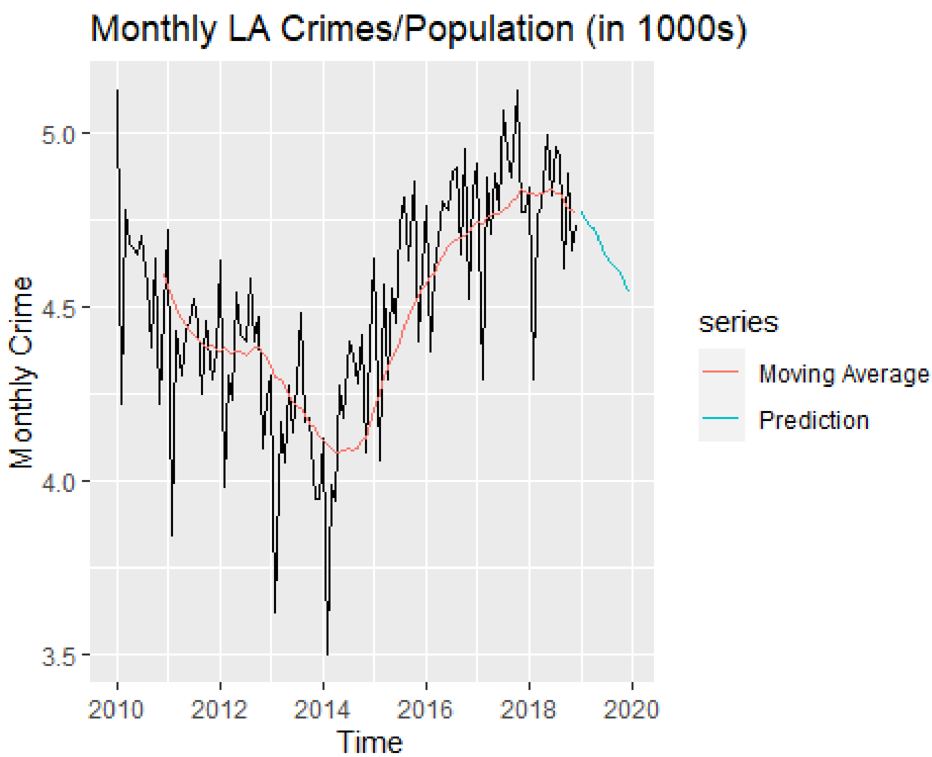

```
autoplot(traintest.ts,ylab = "Monthly Crime", main = "Monthly LA
Crimes/Population (in 1000s)")+
autolayer(ma.trailing.1,series = "Moving Average")+
autolayer(ma.trailing.pred.1,series = "Prediction")
```



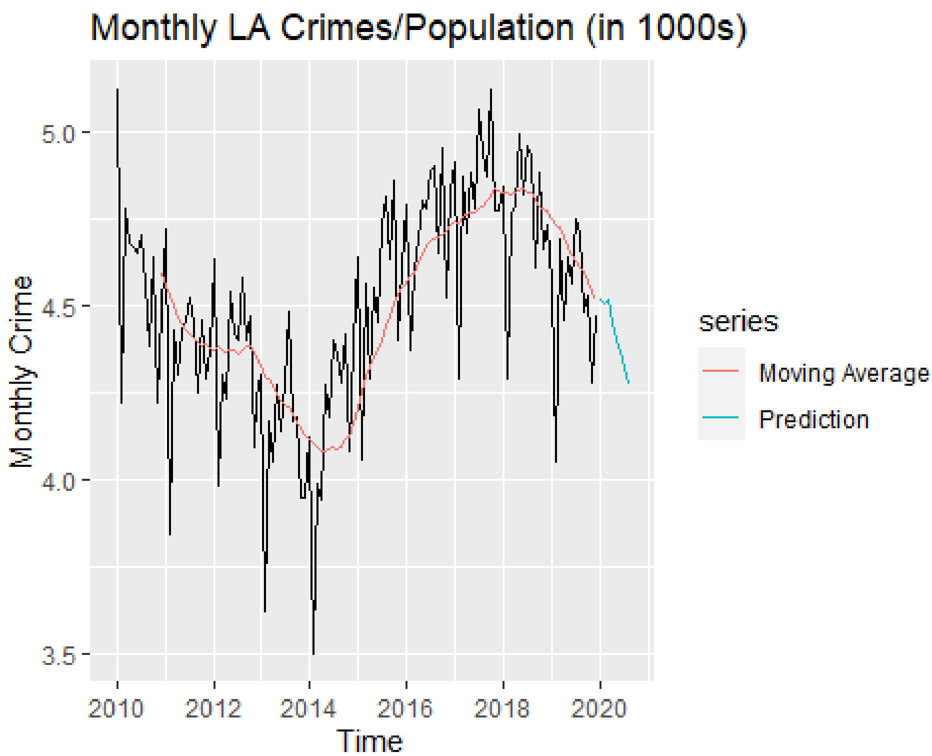

```
# Rolling forward with 1 step
nTrain = length(train.ts)
nTrain.1 = length(traintest.ts)
nValid.1 = 8

# Create a one-step-ahead rolling forecast
step = 1

# Create an empty vector to store our prediction results
ma.trailing.pred.r <- rep(NA,nValid)
ma.trailing.pred.r.1 <- rep(NA,nValid.1)


# Start the for loop for train.ts
for (i in 1:nValid)
{

  # Split the data into training and validation
  nTrain = length(traintest.ts) -nValid + (i-1)
  nTrain.1 = length(crime.ts) -nValid + (i-1)
  train_ma.ts = window(crime.ts,start = c(2010,1),end=c(2010,nTrain))

  # Fit a trailing average smoother
  ma.trailing.roll <- rollmean(train_ma.ts,k=12,align = "right")

  # Find the last moving average in the training period
```

```
  last.ma = tail(ma.trailing.roll,1)

  # Use the last moving average as prediction for each month in the test
period
  ma.trailing.pred.r[i] = last.ma
}

# Start the for loop for traintest.ts
for (i in 1:nValid.1)
{

  # Split the data into training and validation
  nTrain.1 = length(crime.ts) -nValid.1 + (i-1)
  train_ma.1.ts = window(crime.ts,start = c(2010,1),end=c(2010,nTrain.1))

  # Fit a trailing average smoother
  ma.trailing.roll.1 <- rollmean(train_ma.1.ts,k=12,align = "right")

  # Find the last moving average in the training period
  last.ma.1 = tail(ma.trailing.roll.1,1)

  # Use the last moving average as prediction for each month in the test
period
  ma.trailing.pred.r.1[i] = last.ma.1
}


# Converting to time series
ma.trailing.roll = ts(ma.trailing.roll,start
=c(2010,12),end=c(2018,12),frequency = 12)
ma.trailing.roll.1 = ts(ma.trailing.roll.1,start
=c(2010,12),end=c(2019,12),frequency = 12)
ma.trailing.pred.r = ts(ma.trailing.pred.r,start = c(2019,1),end =
c(2019,12),frequency = 12)
ma.trailing.pred.r.1 = ts(ma.trailing.pred.r.1,start = c(2020,1),end =
c(2020,8),frequency = 12)

# Train, test and validation accuracy
accuracy(ma.trailing.roll,train.ts) # train accuracy

##                   ME      RMSE       MAE        MPE     MAPE       ACF1
Theil's U
## Test set 0.0141323 0.2145219 0.1614003 0.06873065 3.691975 0.07449844
0.7180517

accuracy(ma.trailing.pred.r,test.ts) # test accuracy

##                   ME      RMSE      MAE        MPE     MAPE        ACF1
Theil's U
```

```
## Test set -0.144375 0.2487025 0.175625 -3.387831 4.048184 0.004375242
0.8056098

accuracy(ma.trailing.pred.r.1,valid.ts) # 2020 accuracy (train till 2019)

##                    ME      RMSE       MAE       MPE      MAPE      ACF1
Theil's U
## Test set -0.3136458 0.3667295 0.3136458 -7.840216 7.840216 0.2821972
1.790323

# Plotting the moving average graph
autoplot(train.ts,ylab = "Monthly Crime", main = "Monthly LA
Crimes/Population (in 1000s)")+
autolayer(ma.trailing.roll,series = "Moving Average")+
autolayer(ma.trailing.pred.r,series = "Prediction")
```




```
autoplot(traintest.ts,ylab = "Monthly Crime", main = "Monthly LA
Crimes/Population (in 1000s)")+
autolayer(ma.trailing.roll.1,series = "Moving Average")+
autolayer(ma.trailing.pred.r.1,series = "Prediction")
```



Moving average methods give good accuracy. Rolling forward with 1-step is more robust since both train and test values are closer.

Model7 - Exponential Smoothing

```
# Fitting exponential smoothing model
modelopt <-  ets(train.ts,model = "ZZZ")
modelopt.1 <- ets(traintest.ts,model = "ZZZ")

# Model performance
summary(modelopt) # AAdA

## ETS(A,Ad,A)
##
## Call:
##  ets(y = train.ts, model = "ZZZ")
##
##   Smoothing parameters:
##     alpha = 0.4957
##     beta  = 0.0289
##     gamma = 1e-04
##     phi   = 0.9747
##
##   Initial states:
##     l = 4.8499
##     b = -0.0275
```

```
##     s = -0.0088 -0.1684 0.164 -0.0099 0.172 0.1628
##            0.0033 0.1041 -0.0905 -0.011 -0.481 0.1633
##
##   sigma:  0.1022
##
##      AIC     AICc      BIC
## 30.48012 38.16551 78.75848
##
## Training set error measures:
##                        ME       RMSE        MAE        MPE     MAPE
MASE
## Training set 0.005259183 0.09379717 0.07673553 0.09947785 1.709382
0.4081225
##                     ACF1
## Training set 0.03182859
```

```
summary(modelopt.1) # AAdA
```

```
## ETS(A,Ad,A)
##
## Call:
##  ets(y = traintest.ts, model = "ZZZ")
##
##   Smoothing parameters:
##     alpha = 0.4661
##     beta  = 0.0415
##     gamma = 1e-04
##     phi   = 0.975
##
##   Initial states:
##     l = 4.8978
##     b = -0.0366
##     s = -1e-04 -0.1602 0.1737 -0.0099 0.1798 0.1763
##            0.0209 0.0947 -0.1055 -0.0034 -0.5014 0.1351
##
##   sigma:  0.1035
##
##      AIC     AICc      BIC
## 47.82418 54.59646 97.99903
##
## Training set error measures:
##                        ME      RMSE        MAE        MPE     MAPE
MASE
## Training set 0.001776786 0.0958972 0.07567869 0.02637627 1.683675
0.3869933
##                     ACF1
## Training set 0.06250074
```

```
# Train, test and validation accuracy
accuracy(modelopt$fitted, train.ts) # train accuracy
```

```
##                     ME       RMSE        MAE        MPE     MAPE       ACF1
## Test set 0.005259183 0.09379717 0.07673553 0.09947785 1.709382 0.03182859
##          Theil's U
## Test set 0.3057293
```

```
accuracy(forecast(modelopt, h = length(test.ts))$mean,test.ts) # test
accuracy
```

```
##                    ME      RMSE       MAE       MPE     MAPE      ACF1
Theil's U
## Test set -0.2291524 0.2436011 0.2291524 -5.107122 5.107122 0.3234059
0.7483415
```

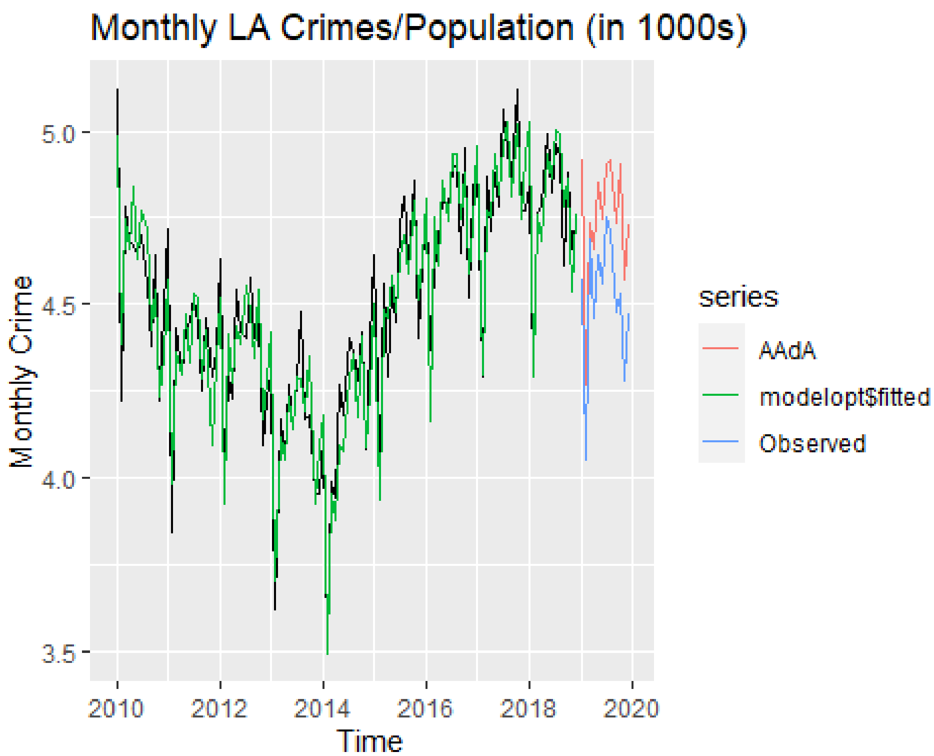
```
accuracy(forecast(modelopt, h =
length(test.ts)+length(valid.ts))$mean,valid.ts) # 2020 accuracy (train till
2018)
```

```
##                    ME      RMSE       MAE       MPE     MAPE      ACF1
Theil's U
## Test set -0.6322584 0.6836349 0.6322584 -15.60584 15.60584 0.1159685
3.261395
```

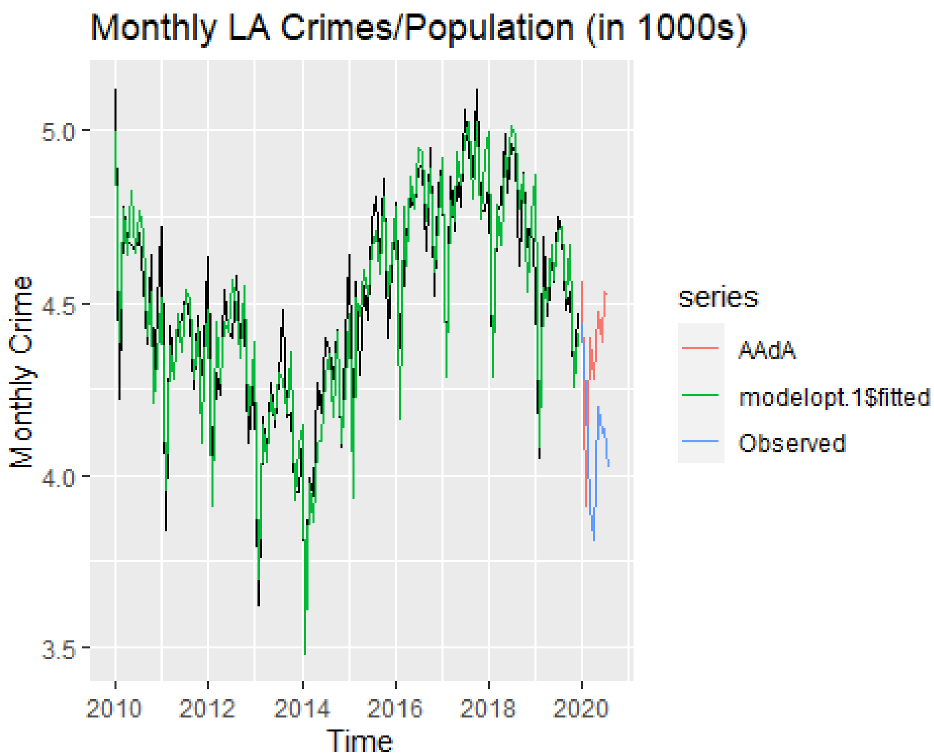
```
accuracy(forecast(modelopt.1, h = length(valid.ts))$mean,valid.ts) # 2020
accuracy (train till 2019)
```

```
##                    ME      RMSE      MAE       MPE    MAPE      ACF1 Theil's
U
## Test set -0.2759697 0.3750366 0.353227 -6.909419 8.74016 0.0677085
1.813177
```

```
# Plotting the exponential smoothing graph
autoplot(train.ts,ylab = "Monthly Crime", main = "Monthly LA
Crimes/Population (in 1000s)")+
autolayer(modelopt$fitted)+
autolayer(forecast(modelopt, h = length(test.ts))$mean,series = "AAdA")+
autolayer(test.ts,series = "Observed")
```



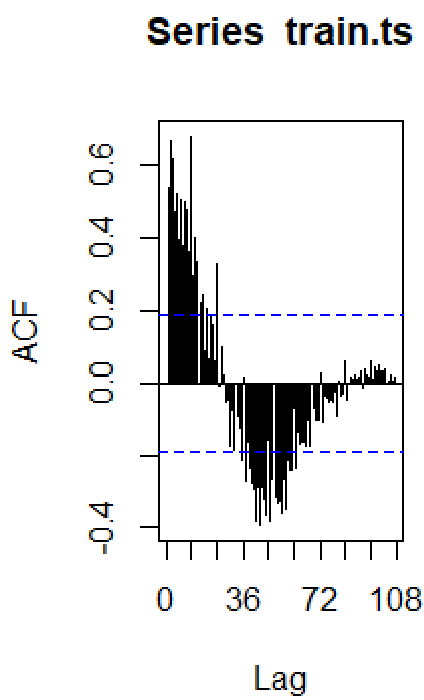

```
autoplot(traintest.ts,ylab = "Monthly Crime", main = "Monthly LA
Crimes/Population (in 1000s)")+
autolayer(modelopt.1$fitted)+
autolayer(forecast(modelopt.1, h = length(valid.ts))$mean,series = "AAdA")+
autolayer(valid.ts,series = "Observed")
```

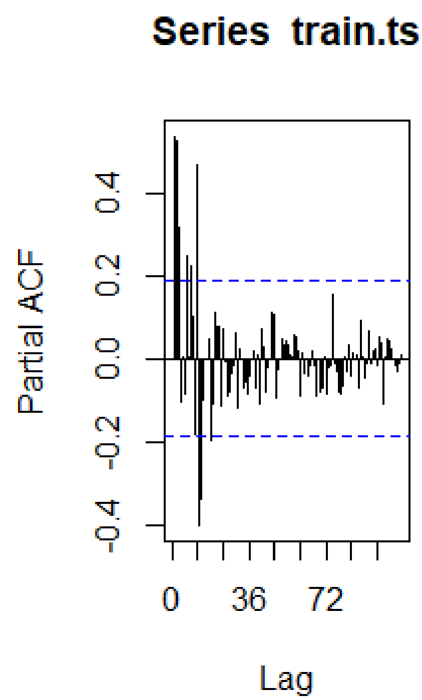


The chosen optimal ets model is AAdA and it can be seen that test accuracy is much higher than that of training accuracy suggesting that this model is trying to overfit.

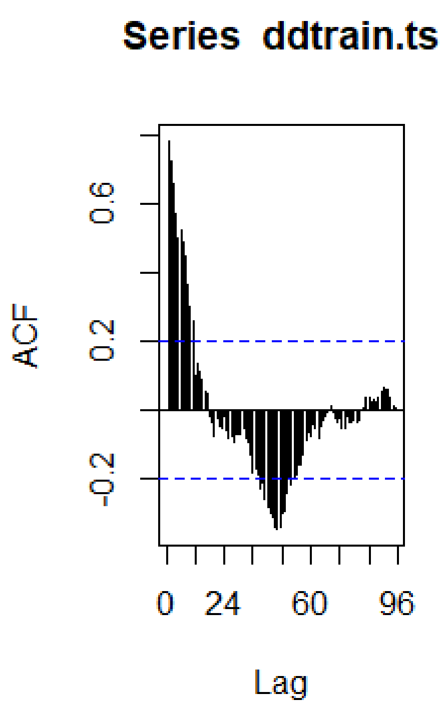
Model8 - SARIMA (Box-Jenkins)

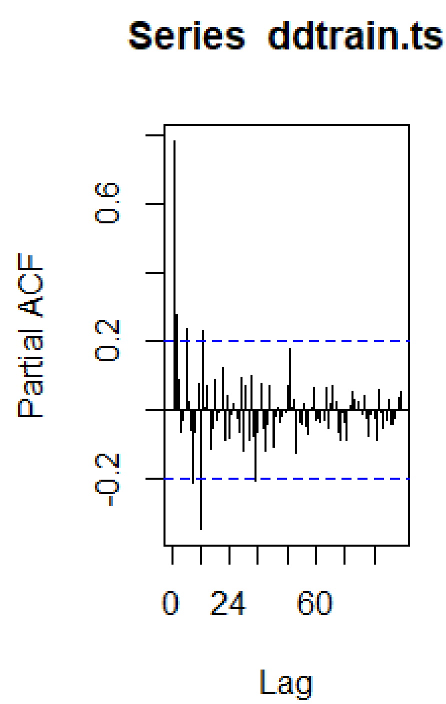
```
# Fitting exponential smoothing model

# differencing the time series to remove seasonality
ddtrain.ts <- diff(train.ts,lag = 12)

# Acf and Pacf for train.ts
par(mfrow = c(1,2))
Acf(train.ts,lag.max = 150)
Pacf(train.ts,lag.max = 150)
```






```r
# Acf and Pacf for ddtrain
Acf(ddtrain.ts,lag.max = 150)
Pacf(ddtrain.ts,lag.max = 150)
```





```r
par(mfrow = c(1,1))

# Fitting best SARIMA Models for train.ts and traintest.ts
modelsarima.1 <- Arima(train.ts, order = c(2,0,0),seasonal = list(order =
c(0,0,1),period = 12))
modelsarima.2 <- Arima(train.ts, order = c(2,0,1),seasonal = list(order =
c(0,0,0),period = 12))
modelsarima.3 <- Arima(train.ts, order = c(2,0,1),seasonal = list(order =
c(0,0,1),period = 12))
modelsarima.4 <- Arima(train.ts, order = c(1,0,1),seasonal = list(order =
c(1,1,1),period = 12))

modelsarima.1.1 <- Arima(traintest.ts, order = c(2,0,0),seasonal = list(order
= c(0,0,1),period = 12))
modelsarima.1.2 <- Arima(traintest.ts, order = c(2,0,1),seasonal = list(order
= c(0,0,0),period = 12))
modelsarima.1.3 <- Arima(traintest.ts, order = c(2,0,1),seasonal = list(order
= c(0,0,1),period = 12))
modelsarima.1.4 <- Arima(traintest.ts, order = c(1,0,1),seasonal = list(order
= c(1,1,1),period = 12))


# Model performance
summary(modelsarima.1)

## Series: train.ts
## ARIMA(2,0,0)(0,0,1)[12] with non-zero mean
##
## Coefficients:
##          ar1     ar2    sma1    mean
##       0.2899  0.4703  0.5901  4.5439
## s.e.  0.0871  0.0884  0.0732  0.1035
##
## sigma^2 estimated as 0.03181:  log likelihood=31.9
## AIC=-53.81   AICc=-53.22   BIC=-40.4
##
## Training set error measures:
##                        ME      RMSE       MAE        MPE     MAPE
MASE
## Training set -0.007049779 0.1750322 0.1299367 -0.3450317 2.951586
0.6910762
##                    ACF1
## Training set -0.1289881

summary(modelsarima.2)

## Series: train.ts
## ARIMA(2,0,1) with non-zero mean
##
## Coefficients:
```

```
##          ar1     ar2      ma1    mean
##       0.4873  0.4217  -0.3843  4.5443
## s.e.  0.1390  0.1166   0.1382  0.1288
##
## sigma^2 estimated as 0.04939:  log likelihood=10.62
## AIC=-11.24   AICc=-10.66   BIC=2.17
##
## Training set error measures:
##                        ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -0.006102593 0.218074 0.166222 -0.388544 3.793845 0.8840618
##                     ACF1
## Training set -0.01309687
```

```
summary(modelsarima.3)
```

```
## Series: train.ts
## ARIMA(2,0,1)(0,0,1)[12] with non-zero mean
##
## Coefficients:
##          ar1     ar2      ma1    sma1    mean
##       0.5644  0.3257  -0.3787  0.5817  4.5608
## s.e.  0.1686  0.1350   0.1670  0.0746  0.1321
##
## sigma^2 estimated as 0.03055:  log likelihood=34.52
## AIC=-57.04   AICc=-56.21   BIC=-40.95
##
## Training set error measures:
##                        ME      RMSE       MAE        MPE     MAPE
MASE
## Training set -0.007736085 0.1707016 0.1294295 -0.3467505 2.936445
0.6883787
##                     ACF1
## Training set -0.01617884
```

```
summary(modelsarima.4)
```

```
## Series: train.ts
## ARIMA(1,0,1)(1,1,1)[12]
##
## Coefficients:
##          ar1     ma1    sar1     sma1
##       0.9845  -0.406  0.1993  -0.9995
## s.e.  0.0276   0.096  0.1245   0.2362
##
## sigma^2 estimated as 0.01063:  log likelihood=72.83
## AIC=-135.67   AICc=-135   BIC=-122.84
##
## Training set error measures:
##                       ME       RMSE        MAE        MPE     MAPE
MASE
## Training set 0.002026469 0.09515483 0.07497228 0.03122664 1.677612
```

```
0.3987445
##                      ACF1
## Training set -0.03353291
```

```r
summary(modelsarima.1.1)
```

```
## Series: traintest.ts
## ARIMA(2,0,0)(0,0,1)[12] with non-zero mean
##
## Coefficients:
##          ar1    ar2    sma1    mean
##       0.2815  0.461  0.5665  4.5144
## s.e.  0.0826  0.084  0.0657  0.0915
##
## sigma^2 estimated as 0.03229:  log likelihood=34.95
## AIC=-59.91   AICc=-59.38   BIC=-45.97
##
## Training set error measures:
##                        ME      RMSE       MAE        MPE     MAPE
MASE
## Training set -0.006773256 0.1766807 0.1306777 -0.3416893 2.966172
0.6682383
##                     ACF1
## Training set -0.1289679
```

```r
summary(modelsarima.1.2)
```

```
## Series: traintest.ts
## ARIMA(2,0,1) with non-zero mean
##
## Coefficients:
##          ar1     ar2      ma1    mean
##       0.4777  0.4114  -0.3576  4.5110
## s.e.  0.1362  0.1111   0.1356  0.1066
##
## sigma^2 estimated as 0.04973:  log likelihood=11.29
## AIC=-12.59   AICc=-12.06   BIC=1.35
##
## Training set error measures:
##                        ME      RMSE     MAE        MPE     MAPE     MASE
## Training set -0.005914833 0.2192487 0.16721 -0.3855092 3.811056 0.855051
##                      ACF1
## Training set -0.01477205
```

```r
summary(modelsarima.1.3)
```

```
## Series: traintest.ts
## ARIMA(2,0,1)(0,0,1)[12] with non-zero mean
##
## Coefficients:
##          ar1     ar2      ma1    sma1    mean
```

```
##       0.5698  0.3160  -0.3912  0.5672  4.5214
## s.e.  0.1604  0.1265   0.1588  0.0664  0.1183
##
## sigma^2 estimated as 0.0309:  log likelihood=37.94
## AIC=-63.87   AICc=-63.13   BIC=-47.15
##
## Training set error measures:
##                        ME      RMSE       MAE        MPE     MAPE
MASE
## Training set -0.007833573 0.1720731 0.1298033 -0.3510562 2.943662
0.6637671
##                     ACF1
## Training set -0.02017621
```

```
summary(modelsarima.1.4)
```

```
## Series: traintest.ts
## ARIMA(1,0,1)(1,1,1)[12]
##
## Coefficients:
##          ar1      ma1    sar1     sma1
##       0.9795  -0.4180  0.2195  -0.8872
## s.e.  0.0283   0.0951  0.1716   0.2925
##
## sigma^2 estimated as 0.01197:  log likelihood=81.07
## AIC=-152.14   AICc=-151.56   BIC=-138.73
##
## Training set error measures:
##                        ME      RMSE        MAE         MPE     MAPE
MASE
## Training set -0.001925158 0.1018629 0.07925708 -0.05714377 1.770274
0.4052919
##                     ACF1
## Training set -0.01421168
```

```
# Train, test and validation accuracy

#(2,0,0)X(0,0,1)12
accuracy(modelsarima.1$fitted, train.ts) # train accuracy
```

```
##                    ME      RMSE       MAE        MPE     MAPE       ACF1
## Test set -0.007049779 0.1750322 0.1299367 -0.3450317 2.951586 -0.1289881
##          Theil's U
## Test set  0.555668
```

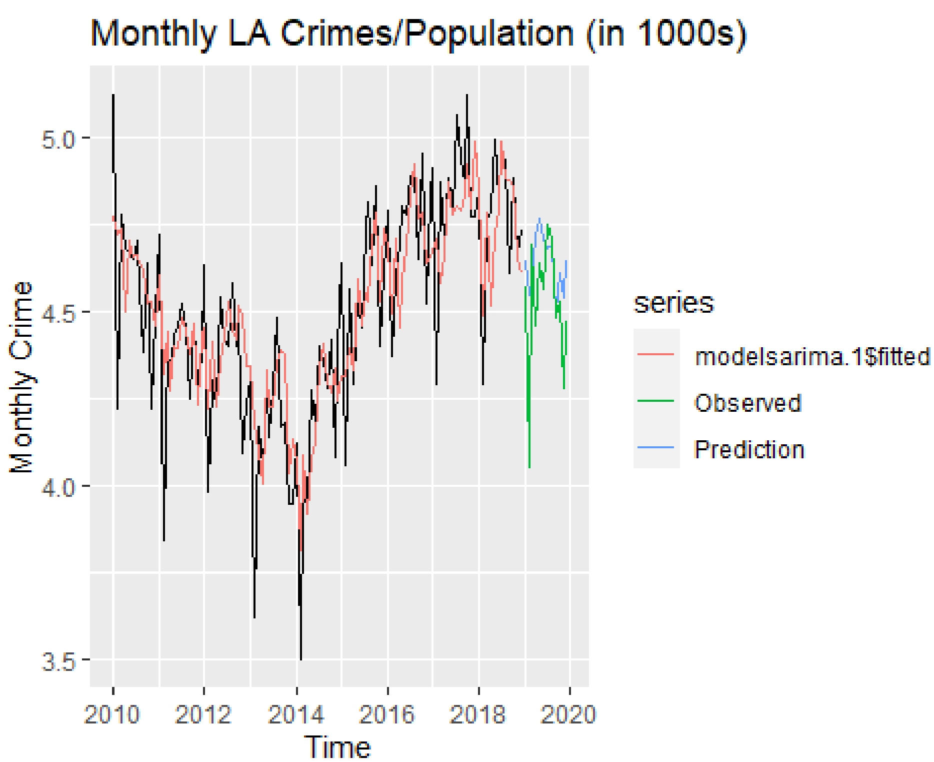
```
accuracy(forecast(modelsarima.1, h = length(test.ts))$mean,test.ts) # test
accuracy
```

```
##                  ME      RMSE       MAE       MPE     MAPE       ACF1
Theil's U
```

```
## Test set -0.1184505 0.1987609 0.1521983 -2.768268 3.484407 -0.2966039
0.6637339

accuracy(forecast(modelsarima.1, h =
length(test.ts)+length(valid.ts))$mean,valid.ts) # 2020 accuracy (train till
2018)

##                  ME      RMSE       MAE       MPE     MAPE      ACF1 Theil's
U
## Test set -0.466034 0.5004733 0.466034 -11.56951 11.56951 0.1565645
2.438525

accuracy(forecast(modelsarima.1.1, h = length(valid.ts))$mean,valid.ts) #
2020 accuracy (train till 2019)

##                   ME      RMSE       MAE       MPE     MAPE      ACF1
Theil's U
## Test set -0.2994051 0.3832406 0.3299067 -7.537992 8.235259 0.2097242
1.864586

#(2,0,1)X(0,0,0)12
accuracy(modelsarima.2$fitted, train.ts) # train accuracy

##                     ME     RMSE      MAE       MPE     MAPE        ACF1
## Test set -0.006102593 0.218074 0.166222 -0.388544 3.793845 -0.01309687
##          Theil's U
## Test set 0.6984745

accuracy(forecast(modelsarima.2, h = length(test.ts))$mean,test.ts) # test
accuracy

##                   ME      RMSE       MAE       MPE    MAPE       ACF1
Theil's U
## Test set -0.1426726 0.2391578 0.1707786 -3.350052 3.94398 -0.1087419
0.7896264

accuracy(forecast(modelsarima.2, h =
length(test.ts)+length(valid.ts))$mean,valid.ts) # 2020 accuracy (train till
2018)

##                   ME      RMSE       MAE       MPE     MAPE     ACF1
Theil's U
## Test set -0.4971953 0.5286662 0.4971953 -12.32729 12.32729 0.169324
2.568575

accuracy(forecast(modelsarima.1.2, h = length(valid.ts))$mean,valid.ts) #
2020 accuracy (train till 2019)

##                   ME      RMSE       MAE       MPE     MAPE     ACF1
Theil's U
## Test set -0.3403194 0.3906756 0.3481795 -8.511107 8.688139 0.168984
1.91621
```

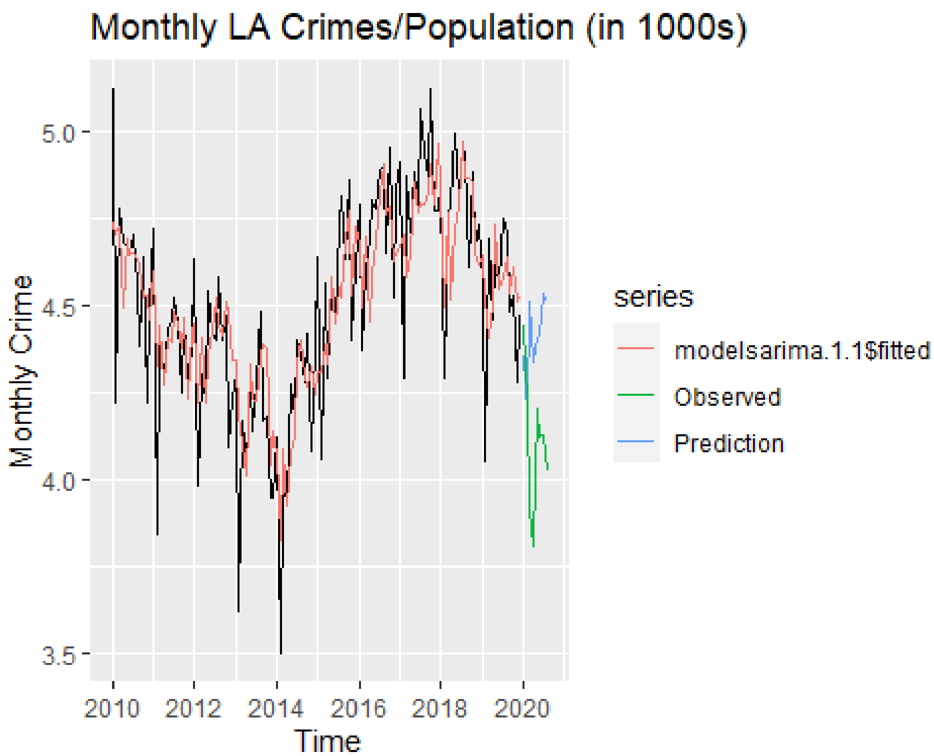
```
#(2,0,1)X(0,0,1)12
accuracy(modelsarima.3$fitted, train.ts) # train accuracy

##                       ME      RMSE       MAE        MPE     MAPE        ACF1
## Test set -0.007736085 0.1707016 0.1294295 -0.3467505 2.936445 -0.01617884
##          Theil's U
## Test set 0.5435921

accuracy(forecast(modelsarima.3, h = length(test.ts))$mean,test.ts) # test
accuracy

##                  ME      RMSE       MAE       MPE     MAPE       ACF1
Theil's U
## Test set -0.13108 0.2019163 0.1563507 -3.043553 3.579994 -0.3013802
0.6742021

accuracy(forecast(modelsarima.3, h =
length(test.ts)+length(valid.ts))$mean,valid.ts) # 2020 accuracy (train till
2018)

##                    ME      RMSE       MAE       MPE     MAPE      ACF1
Theil's U
## Test set -0.4905066 0.5236929 0.4905066 -12.16794 12.16794 0.1664772
2.548771

accuracy(forecast(modelsarima.1.3, h = length(valid.ts))$mean,valid.ts) #
2020 accuracy (train till 2019)

##                    ME      RMSE       MAE       MPE     MAPE      ACF1
Theil's U
## Test set -0.2890992 0.3692619 0.3138331 -7.277003 7.846011 0.1793986
1.798637

#(1,0,1)X(1,1,1)12
accuracy(modelsarima.4$fitted, train.ts) # train accuracy

##                     ME       RMSE        MAE        MPE     MAPE        ACF1
## Test set 0.002026469 0.09515483 0.07497228 0.03122664 1.677612 -0.03353291
##          Theil's U
## Test set 0.3158059

accuracy(forecast(modelsarima.4, h = length(test.ts))$mean,test.ts) # test
accuracy

##                    ME      RMSE       MAE       MPE     MAPE      ACF1
Theil's U
## Test set -0.2197465 0.2336608 0.2197465 -4.897721 4.897721 0.2313509
0.7231835

accuracy(forecast(modelsarima.4, h =
length(test.ts)+length(valid.ts))$mean,valid.ts) # 2020 accuracy (train till
2018)
```

```
##                     ME      RMSE       MAE       MPE     MAPE      ACF1
Theil's U
## Test set -0.6226681 0.6754007 0.6226681 -15.37314 15.37314 0.1135854
3.223363

accuracy(forecast(modelsarima.1.4, h = length(valid.ts))$mean,valid.ts) #
2020 accuracy (train till 2019)

##                     ME      RMSE       MAE     MPE     MAPE      ACF1 Theil's
U
## Test set -0.3151786 0.4222984 0.3912147 -7.8944 9.696203 0.1556587
2.053325

# Plotting the SARIMA model graph for (2,0,0)X(0,0,1)12 (best results among
4)
autoplot(train.ts,ylab = "Monthly Crime", main = "Monthly LA
Crimes/Population (in 1000s)")+
autolayer(modelsarima.1$fitted)+
autolayer(forecast(modelsarima.1, h = length(test.ts))$mean,series =
"Prediction")+
autolayer(test.ts,series = "Observed")
```

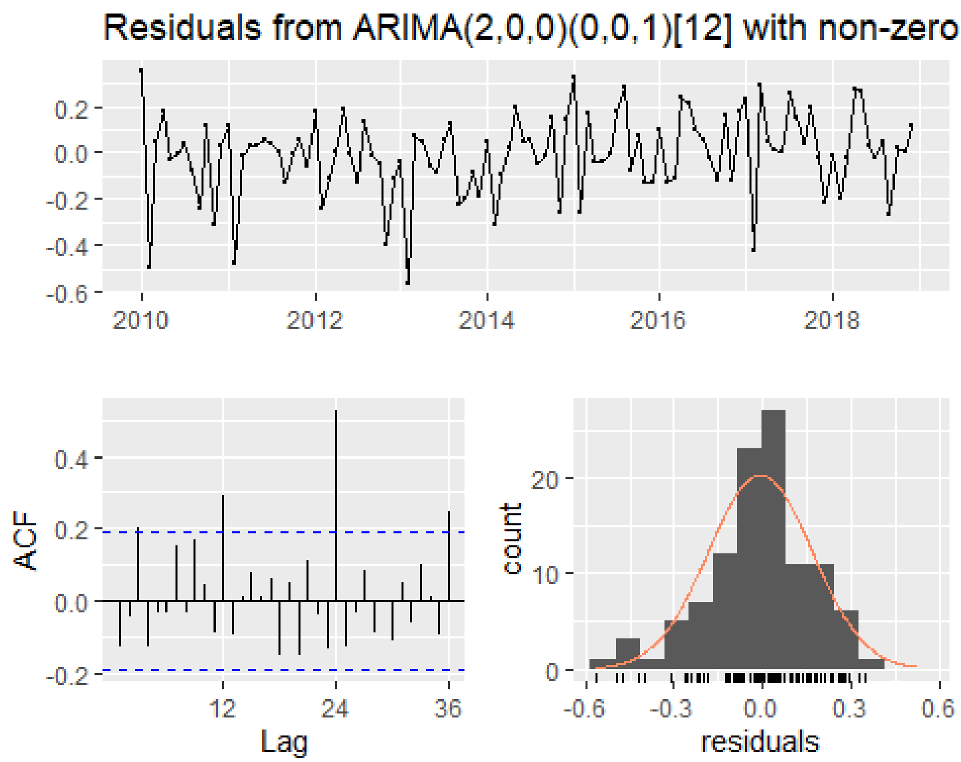



```
autoplot(traintest.ts,ylab = "Monthly Crime", main = "Monthly LA
Crimes/Population (in 1000s)")+
autolayer(modelsarima.1.1$fitted)+
autolayer(forecast(modelsarima.1.1, h = length(valid.ts))$mean,series =
```

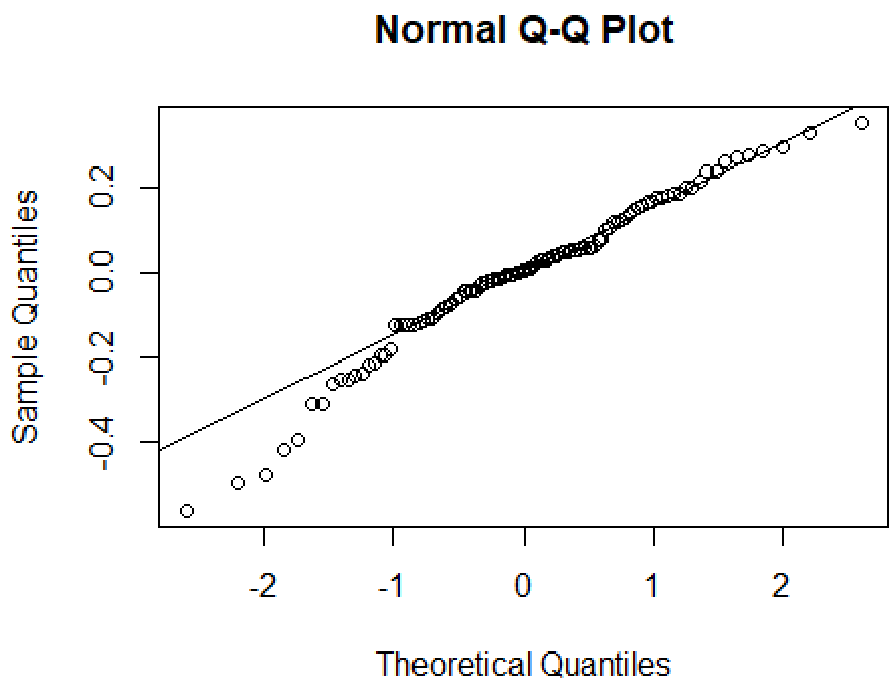
```
"Prediction")+
autolayer(valid.ts,series = "Observed")
```



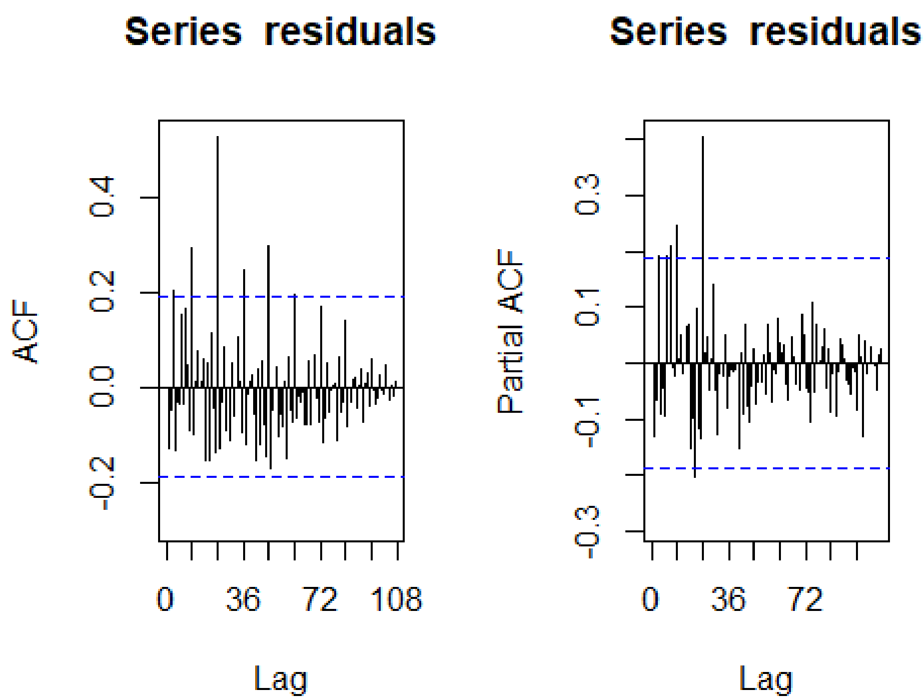

```
# Checking for model assumptions
residuals <- modelsarima.1$residuals

checkresiduals(modelsarima.1) # can directly add model
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,0,0)(0,0,1)[12] with non-zero mean
## Q* = 37.924, df = 18, p-value = 0.003964
##
## Model df: 4.   Total lags used: 22

# Normality & Spread Check
qqnorm(residuals)
qqline(residuals)
```




```r
# Autocorrelation check (Independence)
par(mfrow = c(1,2))
Acf(residuals,lag.max = 150)
Pacf(residuals,lag.max = 150)
```




```
par(mfrow = c(1,1))
```

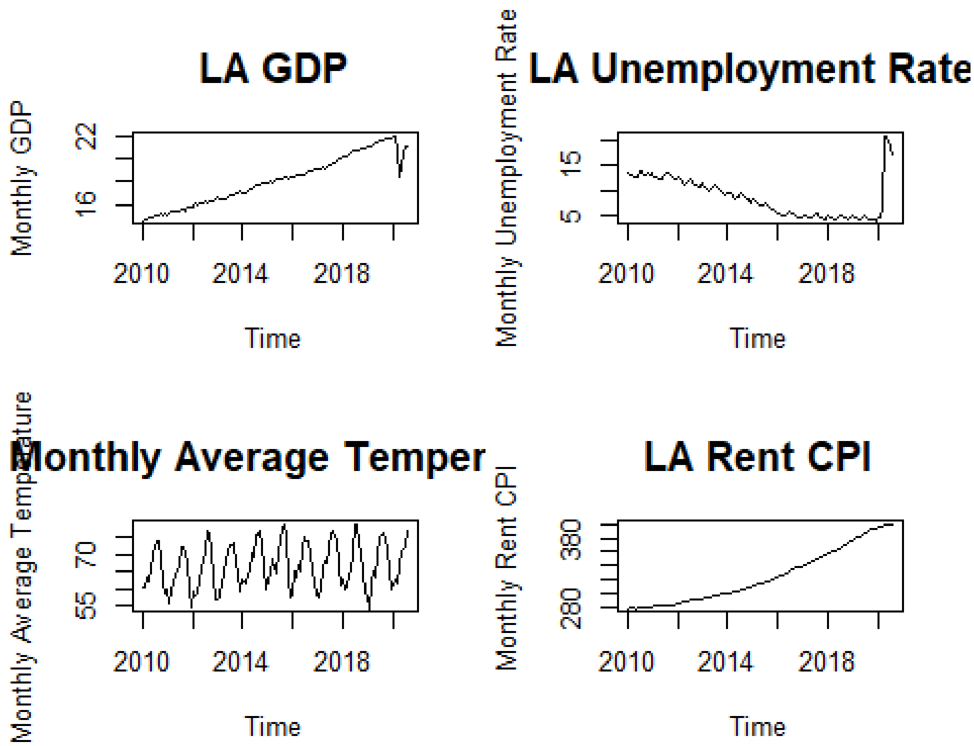
Based on the ACF and PACF plot and accuracy (2,0,0)X(0,0,1) model has been identified as the best model. The accuracy is fairly decent however the residuals show that some seasonality is present. Even after removing that in model 4 the peformance is poor. Thus, this method is not suitable for prediction with this data.

—PART B

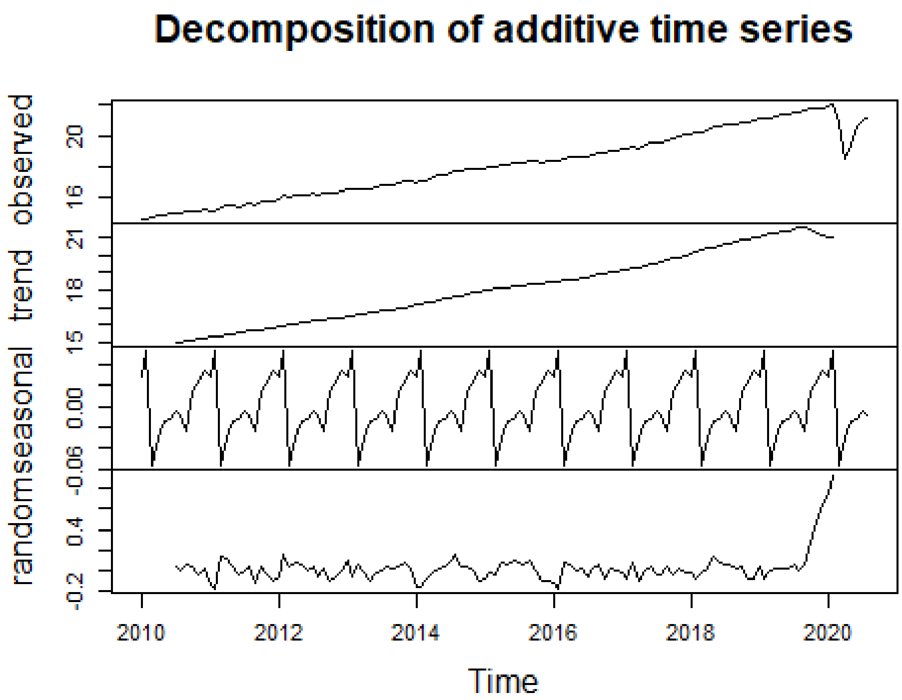
ADDING EXTERNAL DATA

```
# Reading external data for same time duration from Jan 2010 to Aug 2020
gdp=read_excel('GDP_Jan2010_Aug2020.xlsx')
ur=read_excel('Unemployment_Jan2010_Aug2020.xlsx')
rent=read_excel('RentCPI_Jan2010_Aug2020.xlsx')
temp=read_excel('Temperature_Jan2010_Aug2020.xlsx')

# Converting to time series
gdp.ts=ts(gdp$`GDP (Trillion Dollars)`, start=c(2010,1), frequency=12)
ur.ts=ts(ur$`Unemployment Rate`, start=c(2010,1), frequency=12)
rent.ts=ts(rent$RentCPI, start=c(2010,1), frequency=12)
temp.ts=ts(temp$`AVG Temp (Fahrenheit)`, start=c(2010,1), frequency=12)

# Plotting the data
par(mfrow = c(2,2))

#gdp
```

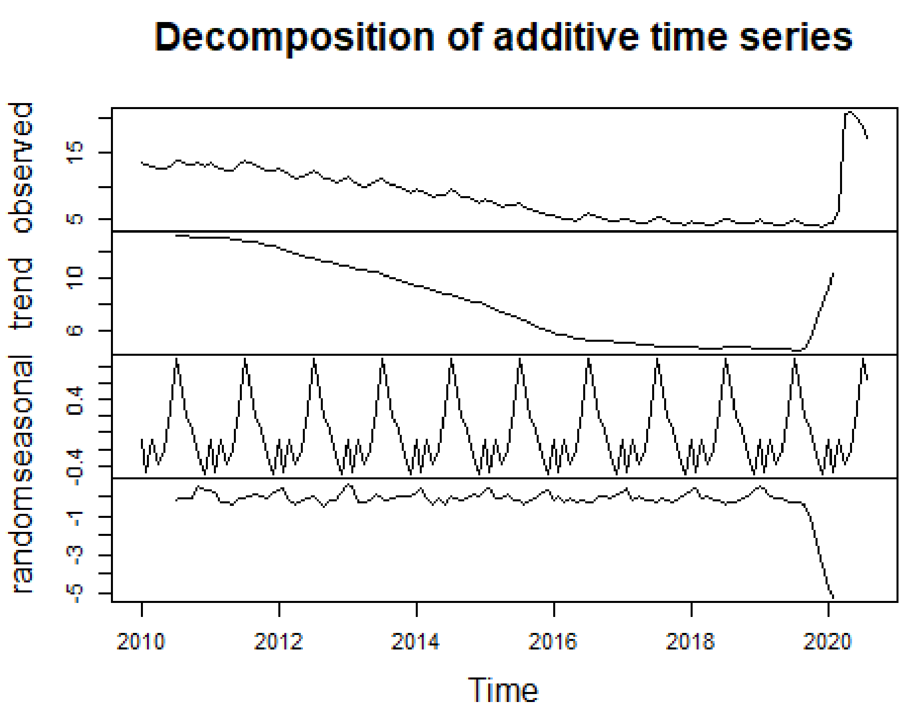
```
plot(gdp.ts, xlab = "Time", ylab = "Monthly GDP", main = "LA GDP", cex.lab =
2, cex.axis=2, cex.main=1.5)

#ur
plot(ur.ts, xlab = "Time", ylab = "Monthly Unemployment Rate", main = "LA
Unemployment Rate ", cex.lab = 2, cex.axis=2, cex.main=1.5)

#temp
plot(temp.ts, xlab = "Time", ylab = "Monthly Average Temperature", main = "LA
Monthly Average Temperature ", cex.lab = 2, cex.axis=2, cex.main=1.5)

#rent
plot(rent.ts, xlab = "Time", ylab = "Monthly Rent CPI", main = "LA Rent CPI",
cex.lab = 2, cex.axis=2, cex.main=1.5)
```



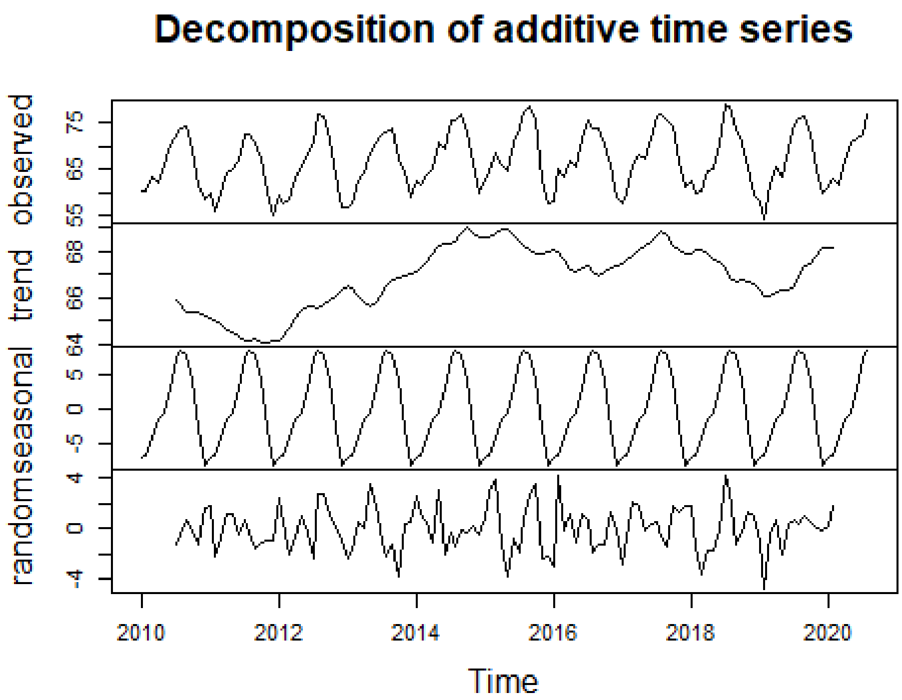

```
par(mfrow = c(1,2))

# Decomposing each time series
plot(decompose(gdp.ts)) # linear rising trend and seasonality
```



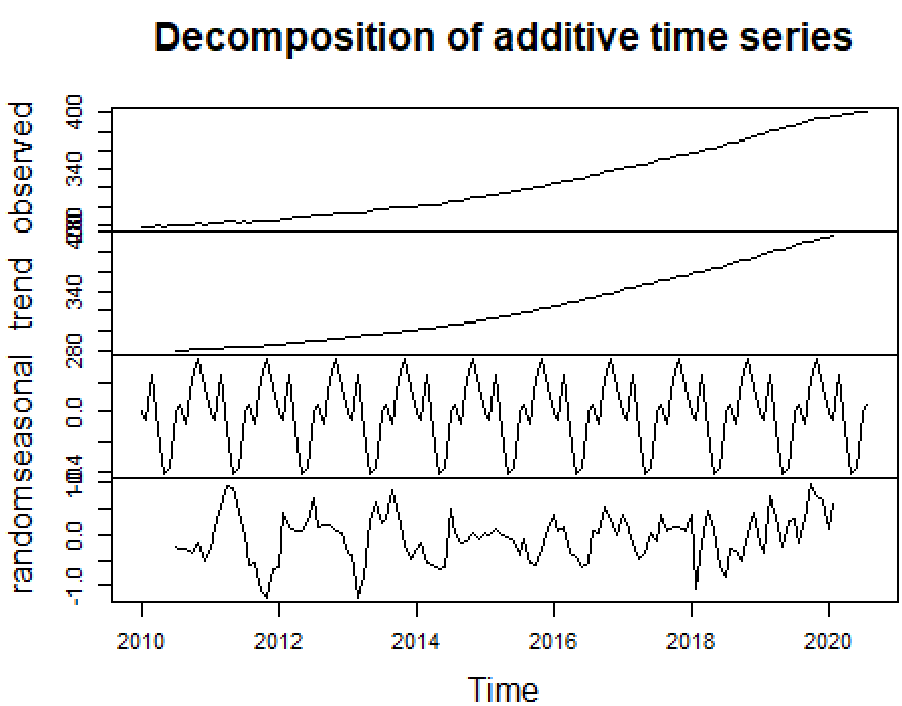

```r
plot(decompose(ur.ts)) # declining trend till 2020 and seasonality
```




```r
plot(decompose(temp.ts)) # no clear trend with seasonality
```




```r
plot(decompose(rent.ts)) # linear rising trend with seasonality
```

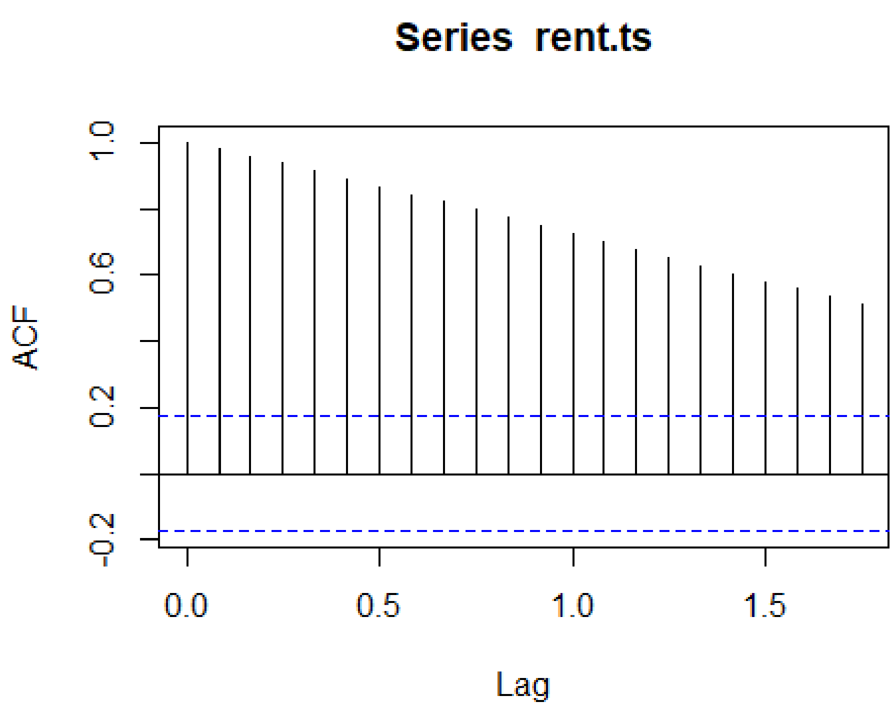


Checking if time series are stationary

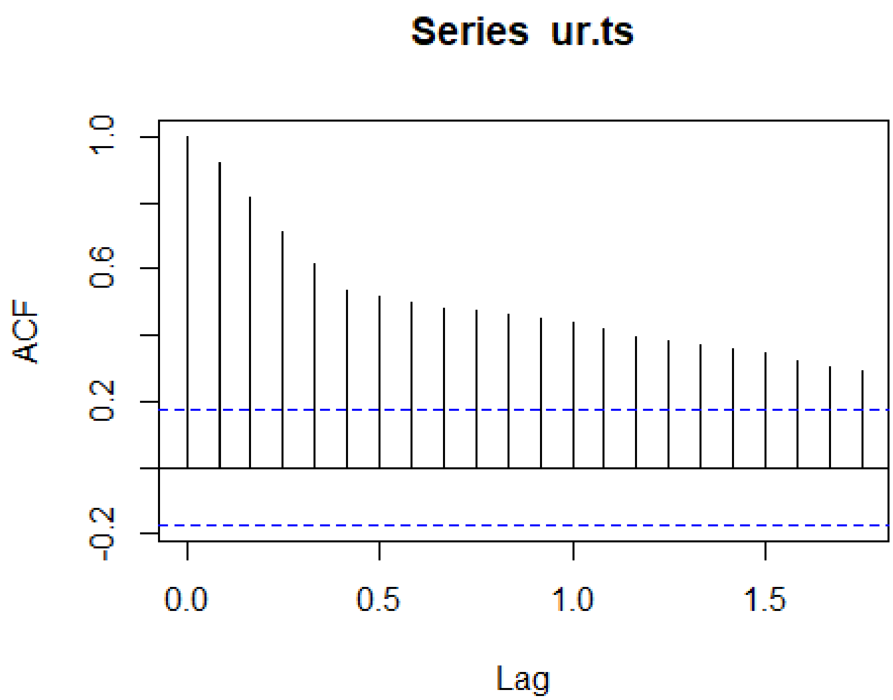
```r
# adf (Augmented Dickey Fuller test)
adf.test(crime.ts,alternative = "stationary",k=0)

## Warning in adf.test(crime.ts, alternative = "stationary", k = 0): p-value
## smaller than printed p-value

##
##  Augmented Dickey-Fuller Test
##
## data:  crime.ts
## Dickey-Fuller = -6.2571, Lag order = 0, p-value = 0.01
## alternative hypothesis: stationary

adf.test(gdp.ts,alternative = "stationary",k=0)

##
##  Augmented Dickey-Fuller Test
##
## data:  gdp.ts
## Dickey-Fuller = -3.9141, Lag order = 0, p-value = 0.01569
## alternative hypothesis: stationary

adf.test(rent.ts,alternative = "stationary",k=0)

##
##  Augmented Dickey-Fuller Test
##
## data:  rent.ts
## Dickey-Fuller = -2.7084, Lag order = 0, p-value = 0.2821
## alternative hypothesis: stationary

adf.test(temp.ts,alternative = "stationary",k=0)

##
##  Augmented Dickey-Fuller Test
##
## data:  temp.ts
## Dickey-Fuller = -3.8082, Lag order = 0, p-value = 0.02084
## alternative hypothesis: stationary

adf.test(ur.ts,alternative = "stationary",k=0)

##
##  Augmented Dickey-Fuller Test
##
## data:  ur.ts
## Dickey-Fuller = -0.72858, Lag order = 0, p-value = 0.9656
## alternative hypothesis: stationary
```

```r
# pp (Phillips-Perron Unit Root test)

pp.test(crime.ts,alternative = "stationary")

## Warning in pp.test(crime.ts, alternative = "stationary"): p-value smaller
than
## printed p-value

##
##  Phillips-Perron Unit Root Test
##
## data:  crime.ts
## Dickey-Fuller Z(alpha) = -67.008, Truncation lag parameter = 4, p-value
## = 0.01
## alternative hypothesis: stationary

pp.test(gdp.ts,alternative = "stationary")

## Warning in pp.test(gdp.ts, alternative = "stationary"): p-value smaller
than
## printed p-value

##
##  Phillips-Perron Unit Root Test
##
## data:  gdp.ts
## Dickey-Fuller Z(alpha) = -29.745, Truncation lag parameter = 4, p-value
## = 0.01
## alternative hypothesis: stationary

pp.test(rent.ts,alternative = "stationary")

##
##  Phillips-Perron Unit Root Test
##
## data:  rent.ts
## Dickey-Fuller Z(alpha) = -2.1542, Truncation lag parameter = 4, p-value
## = 0.9633
## alternative hypothesis: stationary

pp.test(temp.ts,alternative = "stationary")

## Warning in pp.test(temp.ts, alternative = "stationary"): p-value smaller
than
## printed p-value

##
##  Phillips-Perron Unit Root Test
##
## data:  temp.ts
## Dickey-Fuller Z(alpha) = -43.739, Truncation lag parameter = 4, p-value
```

```
## = 0.01
## alternative hypothesis: stationary

pp.test(ur.ts,alternative = "stationary")

##
##  Phillips-Perron Unit Root Test
##
## data:  ur.ts
## Dickey-Fuller Z(alpha) = -4.6533, Truncation lag parameter = 4, p-value
## = 0.8473
## alternative hypothesis: stationary

# rent and ur and are not stationary and hence need to be transformed. Also
seasonality is present in crime that needs to be adjusted.

acf(rent.ts) # trend
```



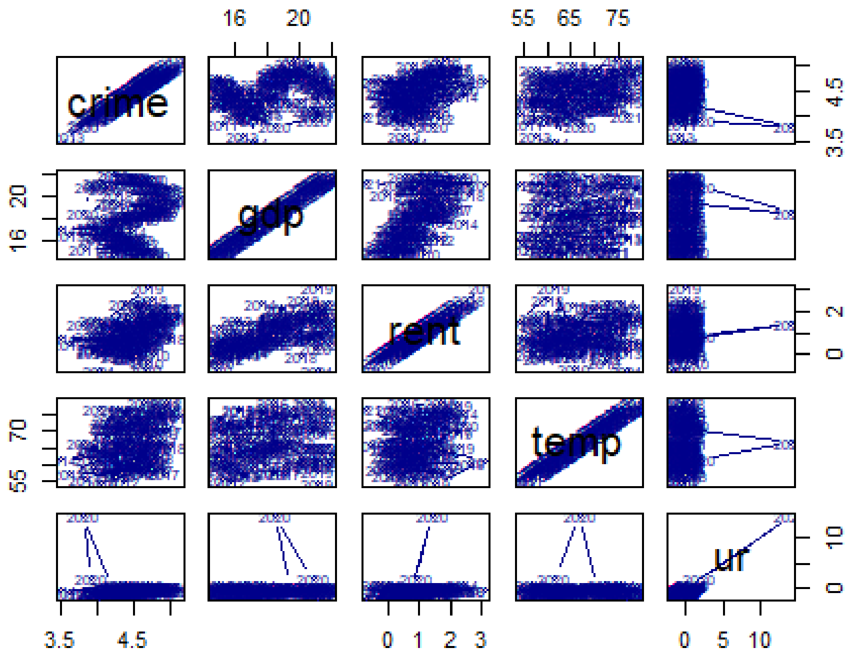

```
acf(ur.ts) # trend
```



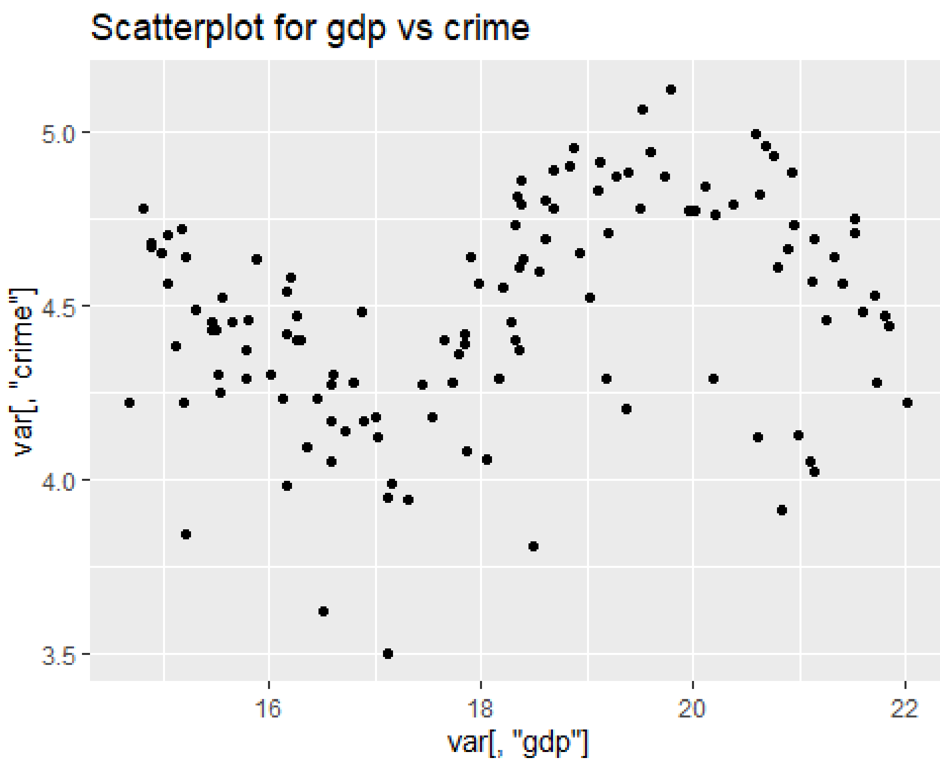

```
par(mfrow = c(1,1))

# Differencing rent and ur to remove trend
newrent.ts <- diff(rent.ts,lag = 1)
newur.ts <- diff(ur.ts,lag = 1)


# Knit the regular and transformed variables
var = ts.intersect(crime=crime.ts,
                   gdp=gdp.ts,
                   rent=newrent.ts,
                   temp=temp.ts,
                   ur=newur.ts)
```

Plotting scatterplot and correlations between variables

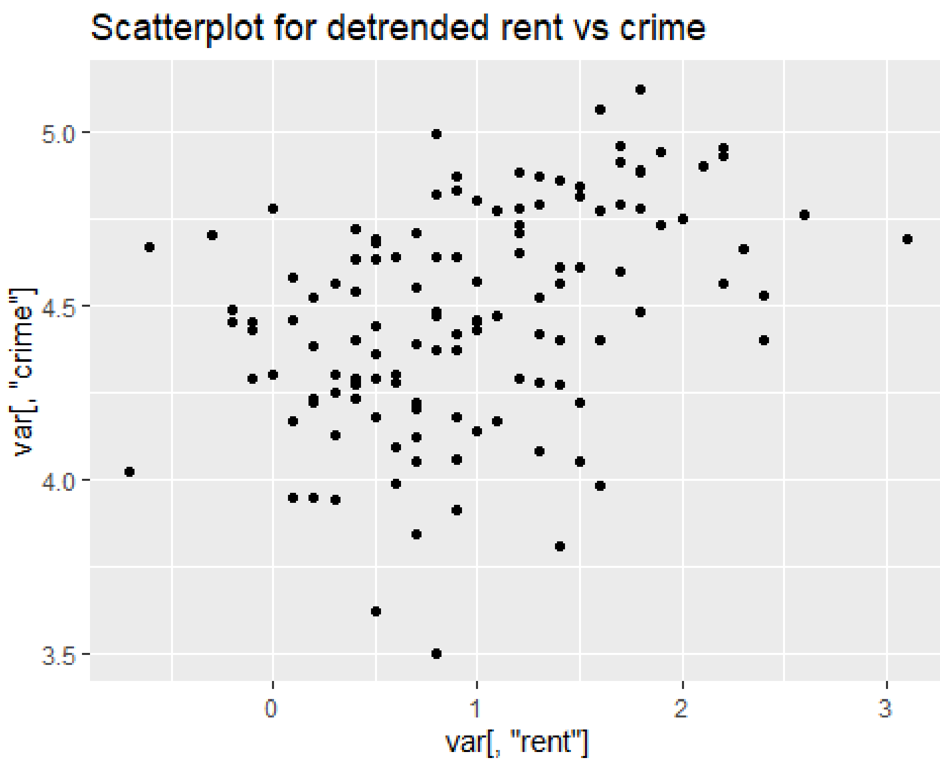
```
# Examine scatterplot of data
pairs(var, cex = 0.65, col = "darkblue")
```



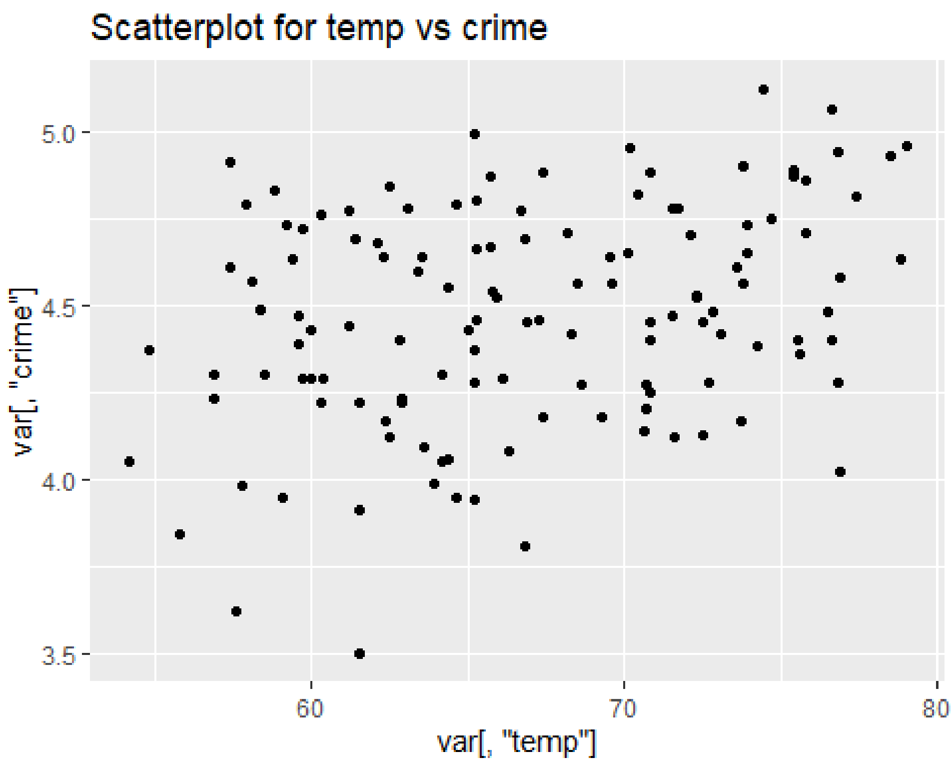

```r
# Examine correlations of data
var.corr <- cor(as.matrix(var))
var.corr.line <- var.corr["crime",]
var.corr.line

##       crime         gdp        rent        temp          ur
##  1.00000000  0.29078249  0.39730053  0.31228182 -0.09255462

# Individual scatterplots
qplot(var[,'gdp'],var[,'crime'],main = "Scatterplot for gdp vs crime")

## Don't know how to automatically pick scale for object of type ts.
Defaulting to continuous.
## Don't know how to automatically pick scale for object of type ts.
Defaulting to continuous.
```



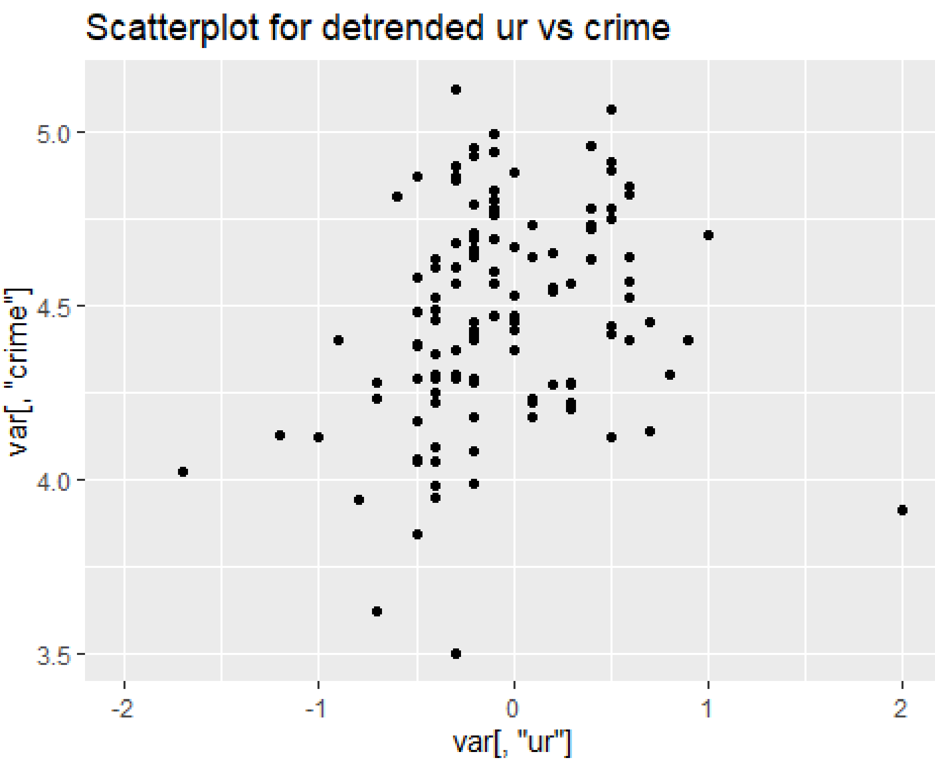

```
qplot(var[,'rent'],var[,'crime'],main = "Scatterplot for detrended rent vs
crime")

## Don't know how to automatically pick scale for object of type ts.
Defaulting to continuous.
## Don't know how to automatically pick scale for object of type ts.
Defaulting to continuous.
```




```
qplot(var[,'temp'],var[,'crime'],main = "Scatterplot for temp vs crime")

## Don't know how to automatically pick scale for object of type ts.
Defaulting to continuous.
## Don't know how to automatically pick scale for object of type ts.
Defaulting to continuous.
```




```
qplot(var[,'ur'],var[,'crime'],main = "Scatterplot for detrended ur vs
crime",xlim = c(-2,2))

## Don't know how to automatically pick scale for object of type ts.
Defaulting to continuous.

## Warning: Removed 1 rows containing missing values (geom_point).
```



The above scatterplots show that increased rent and temperature are showing some positive association with crime. For the UR it can be seen the correlation is weak due to presence of a outliers which when removed will eventually give a positive linear association. The gdp shows polynomial relation with crime.

2020 is a pandemic year that might affect the correlation patterns between variables

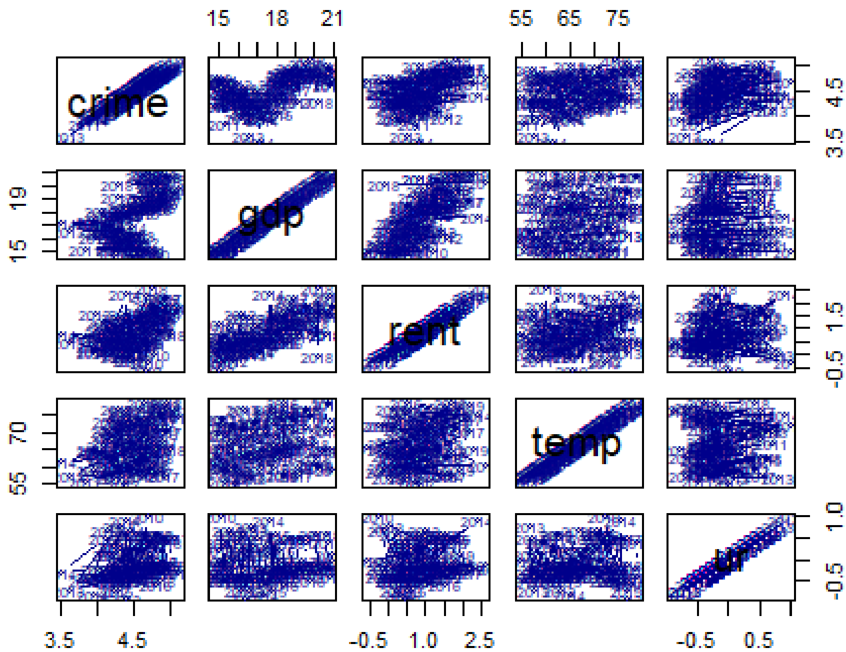
What if we only look at the normal years? Is there a more significant relationship?

Creating train, test and validation set

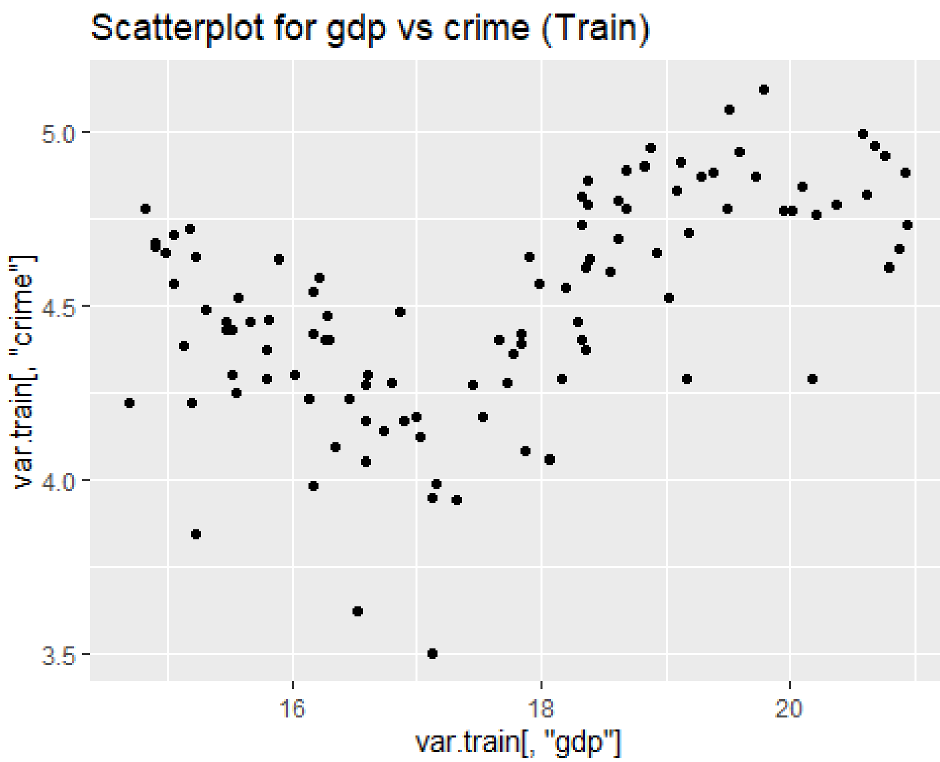
```
# Separating into train, test and validation
var.train = window(var, end=c(2018,12)) # training data
var.test = window(var, start=c(2019,1), end=c(2019,12)) # testing data
var.valid = window(var, start=c(2020,1)) # 2020 validation data
var.traintest = window(var, end=c(2019,12)) #training data till 2019
```

Plotting scatterplot and correlations between variables again for training (normal years)

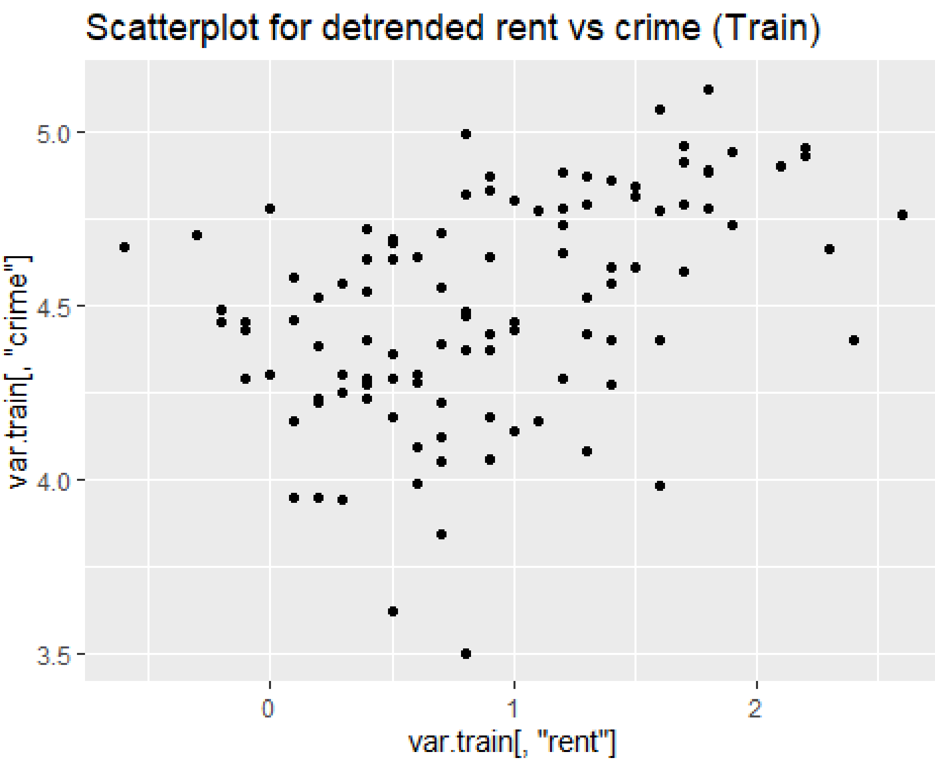
```
# Examine scatter plots of training data
pairs(var.train, cex = 0.65, col = "darkblue")
```



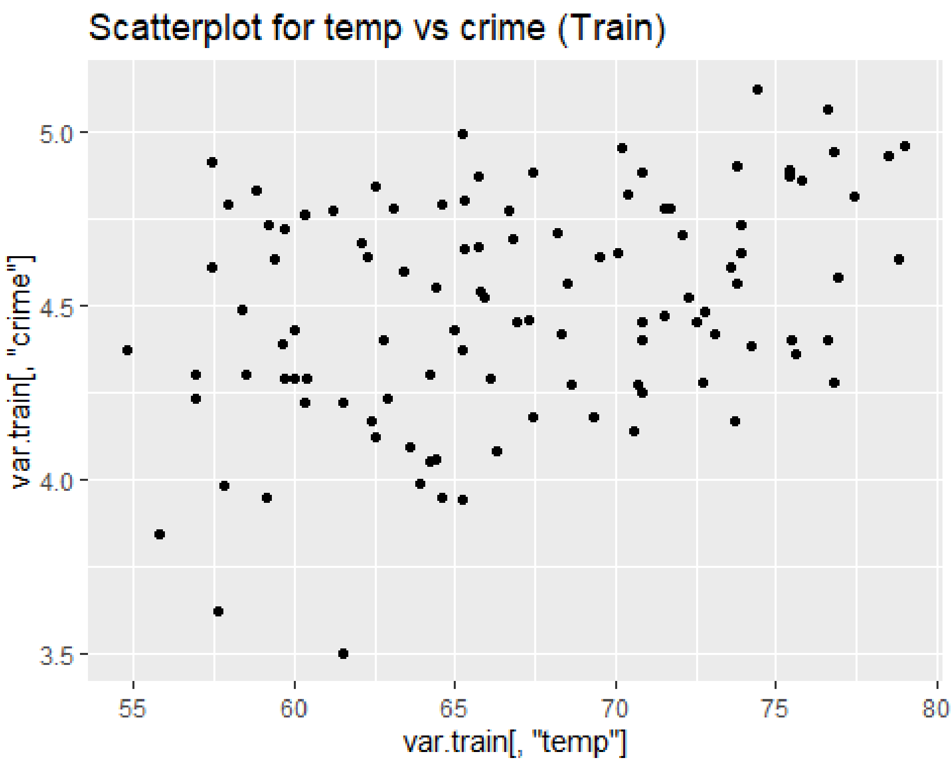

```
# Examine correlations of training data
var.corr <- cor(as.matrix(var.train))
var.corr.line <- var.corr["crime",]
var.corr.line

##     crime       gdp      rent      temp        ur
## 1.0000000 0.5062431 0.4353158 0.3555681 0.2867371

# Individual scatterplots
qplot(var.train[,'gdp'],var.train[,'crime'],main = "Scatterplot for gdp vs
crime (Train)")

## Don't know how to automatically pick scale for object of type ts.
Defaulting to continuous.
## Don't know how to automatically pick scale for object of type ts.
Defaulting to continuous.
```



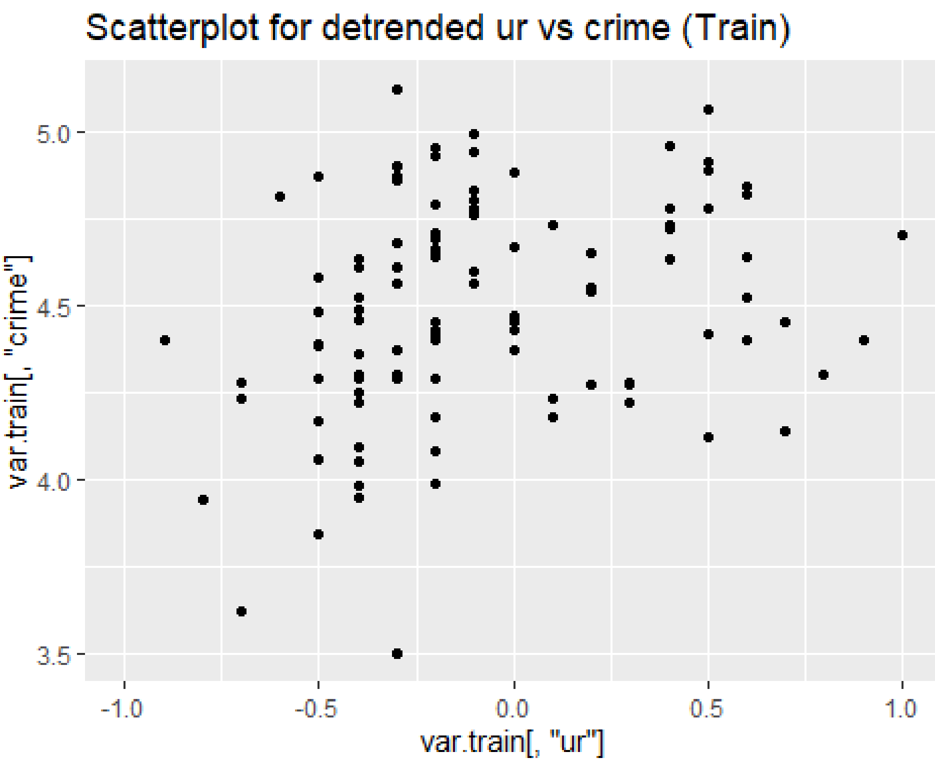

```
qplot(var.train[,'rent'],var.train[,'crime'],main = "Scatterplot for
detrended rent vs crime (Train)")

## Don't know how to automatically pick scale for object of type ts.
Defaulting to continuous.
## Don't know how to automatically pick scale for object of type ts.
Defaulting to continuous.
```




```
qplot(var.train[,'temp'],var.train[,'crime'],main = "Scatterplot for temp vs
crime (Train)")

## Don't know how to automatically pick scale for object of type ts.
Defaulting to continuous.
## Don't know how to automatically pick scale for object of type ts.
Defaulting to continuous.
```



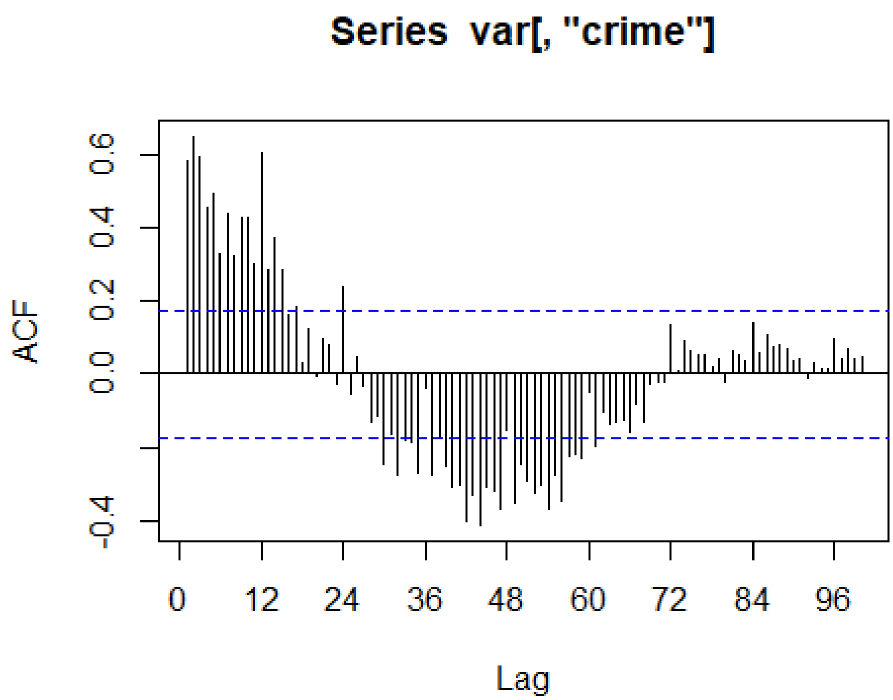

```
qplot(var.train[,'ur'],var.train[,'crime'],main = "Scatterplot for detrended
ur vs crime (Train)",xlim = c(-1,1))

## Don't know how to automatically pick scale for object of type ts.
Defaulting to continuous.
```



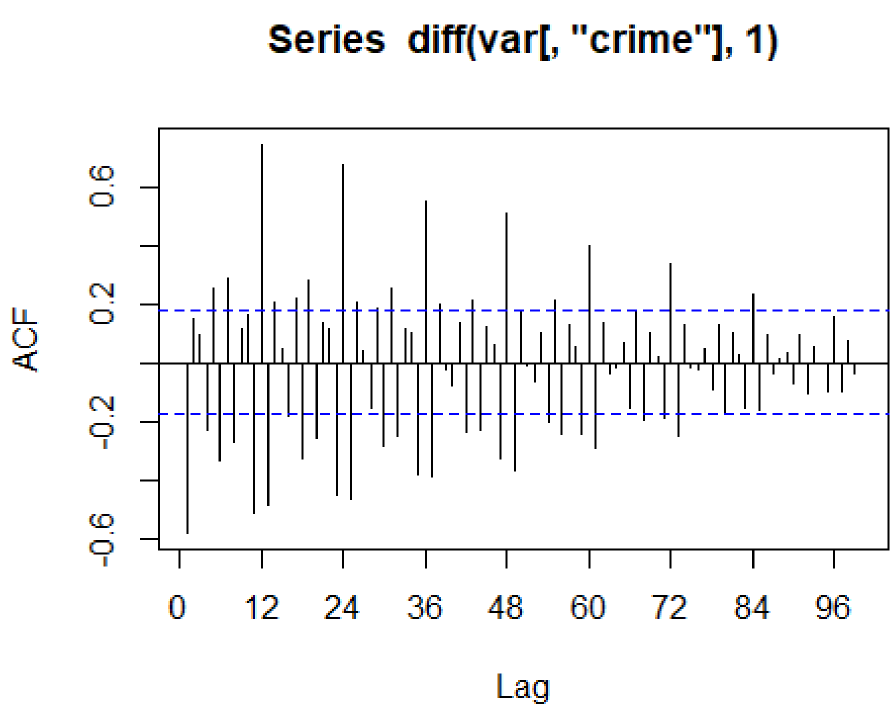
For normal years, stronger association between the external variables and crime is revealed although the pattern is similar to last time. Also, unemployment rate now has slightly positive correlation.

Examine Autocorrelations and cross-correlations between target and external variables

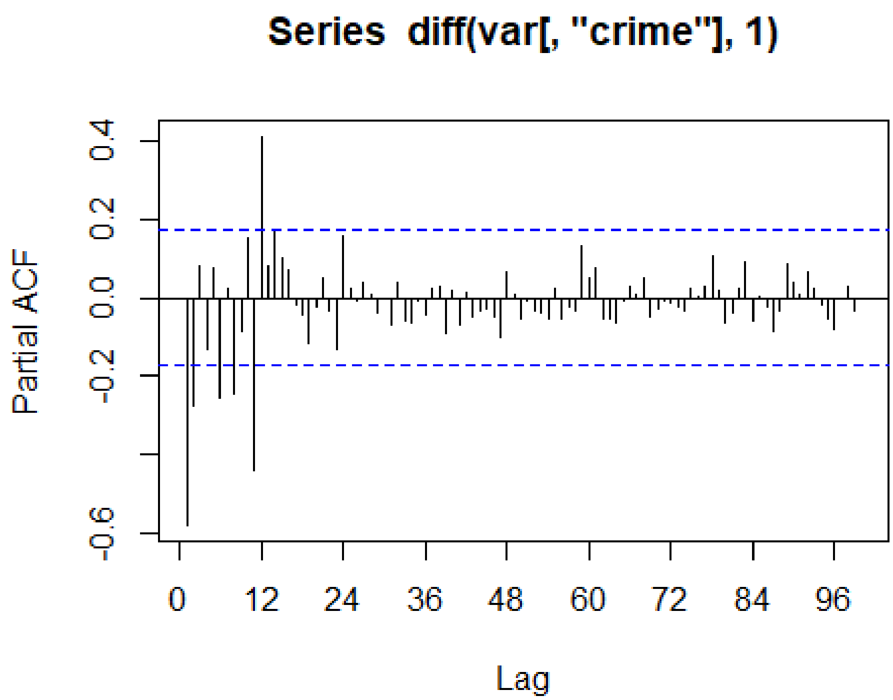
```
# Checking for Autocorrelations for crime

# crime
Acf(var[,'crime'], lag.max = 100) # trend present
```



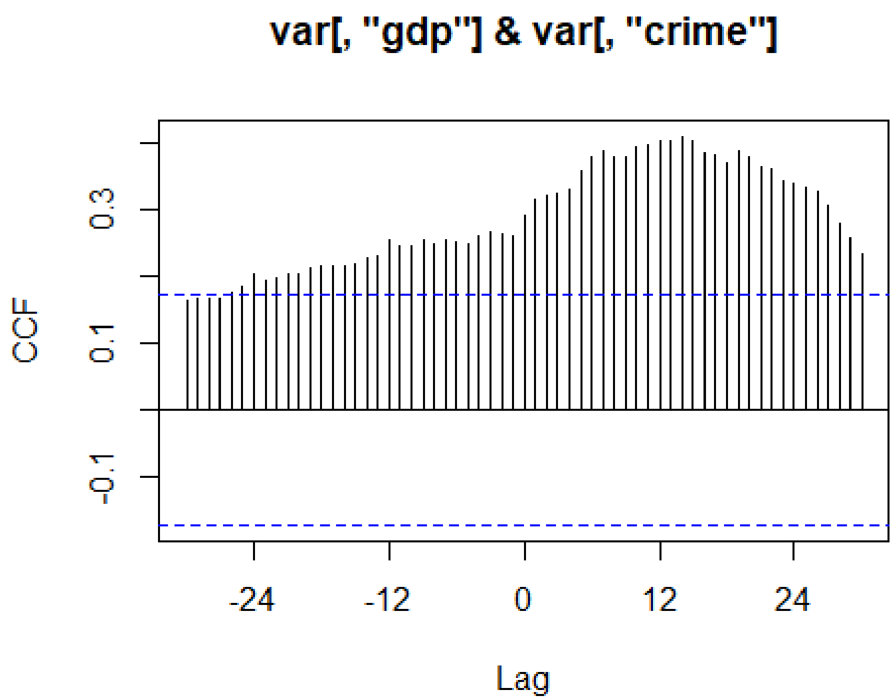

```r
Acf(diff(var[,'crime'],1), lag.max = 100) # Annual seasonality present
```



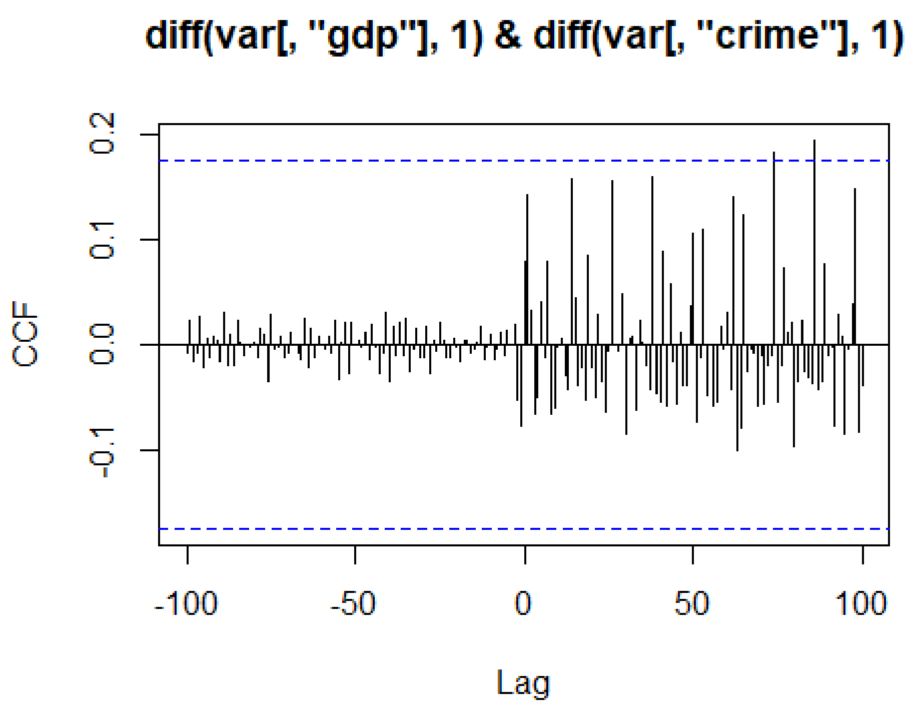

```r
Pacf(diff(var[,'crime'],1), lag.max = 100)
```



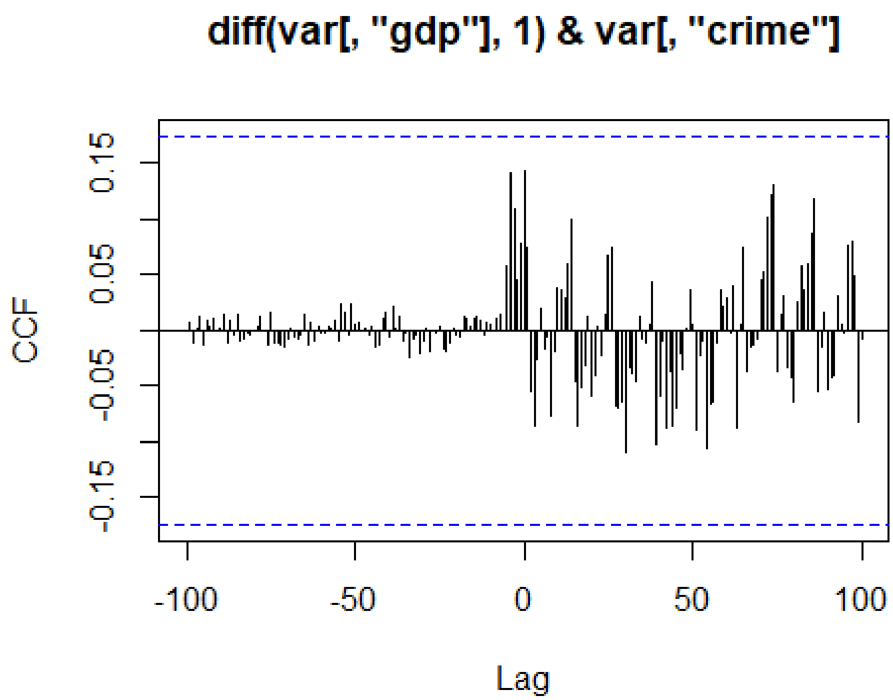

```
# Checking for cross-correlations

# gdp and crime
Ccf(var[,'gdp'], var[,'crime'], lag.max = 30) # lag 12 observed
```



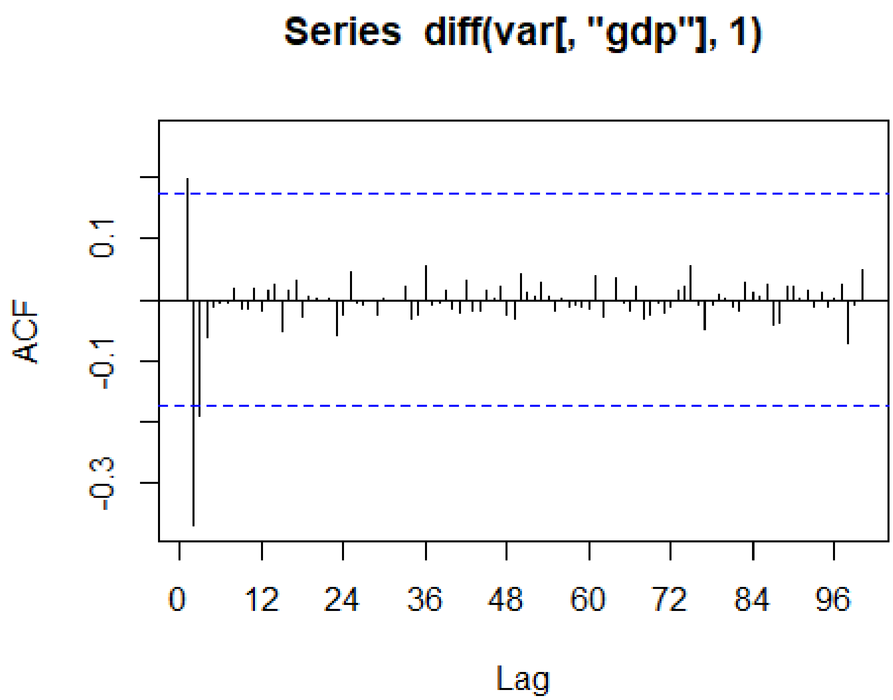

```
Ccf(diff(var[,'gdp'],1), diff(var[,'crime'],1), lag.max = 100) # detrended variables
```



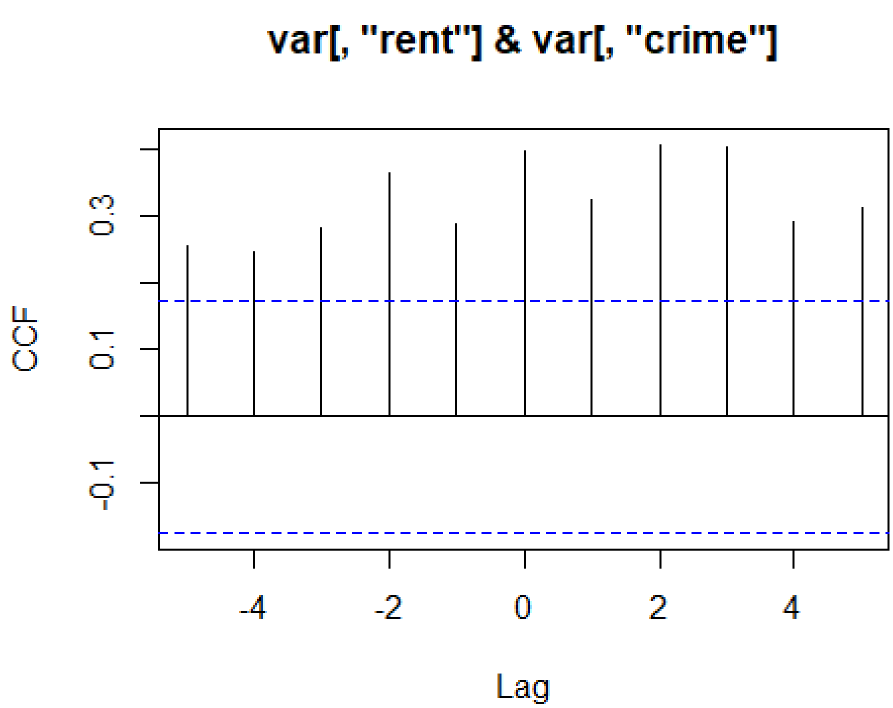
```r
Ccf(diff(var[,'gdp'],1), var[,'crime'], lag.max = 100) # crime vs detrended gdp
```



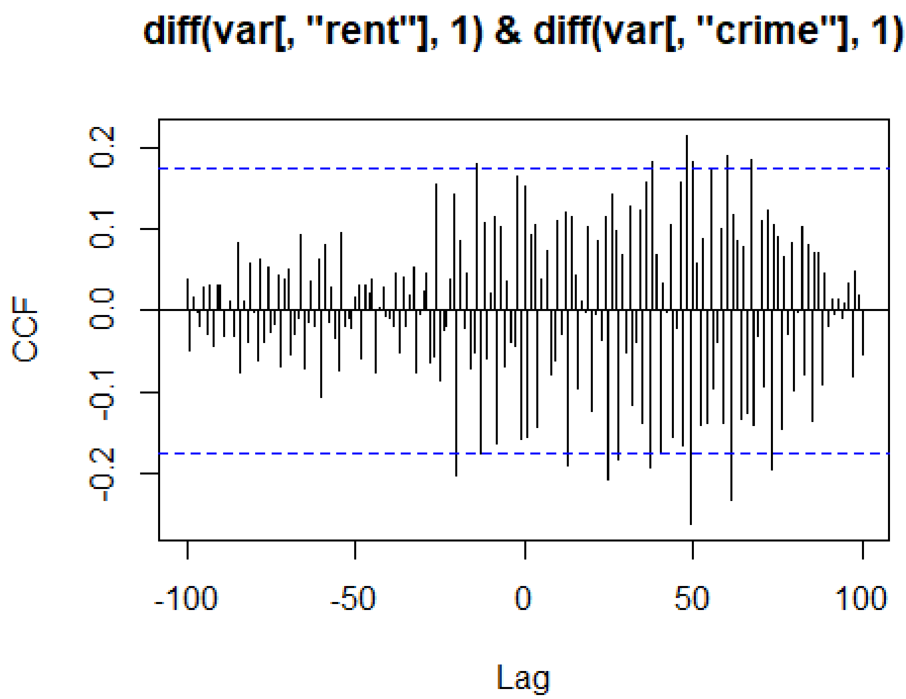

```r
# After removing trend from GDP and crime, there is still some patterns left
Acf(diff(var[,'gdp'],1), lag.max = 100) # a bit seasonality left
```



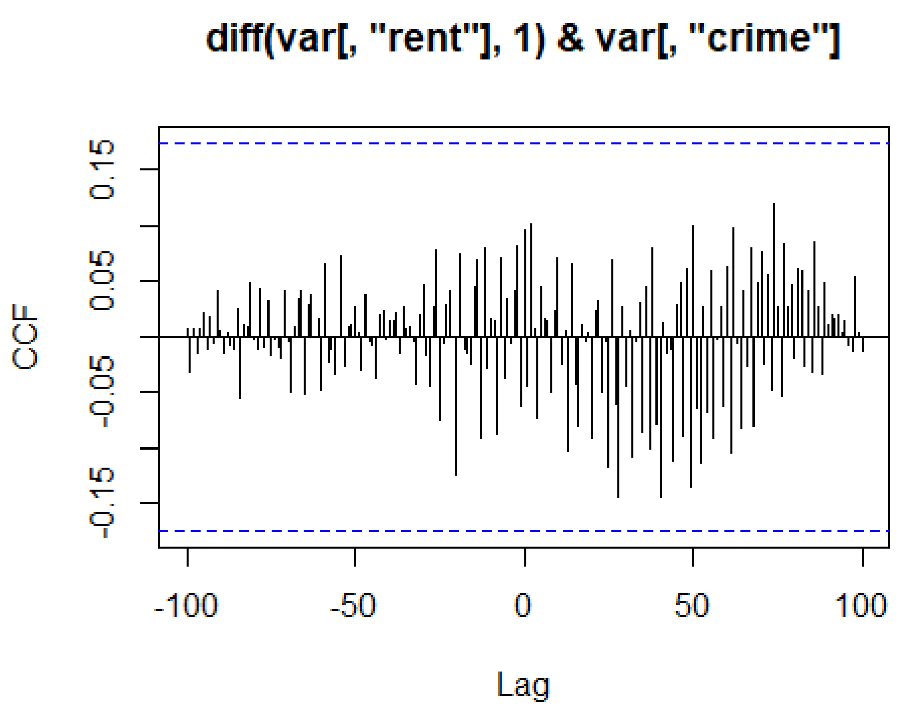

```r
# rent and crime
Ccf(var[,'rent'], var[,'crime'], lag.max = 5) # lag 2
```



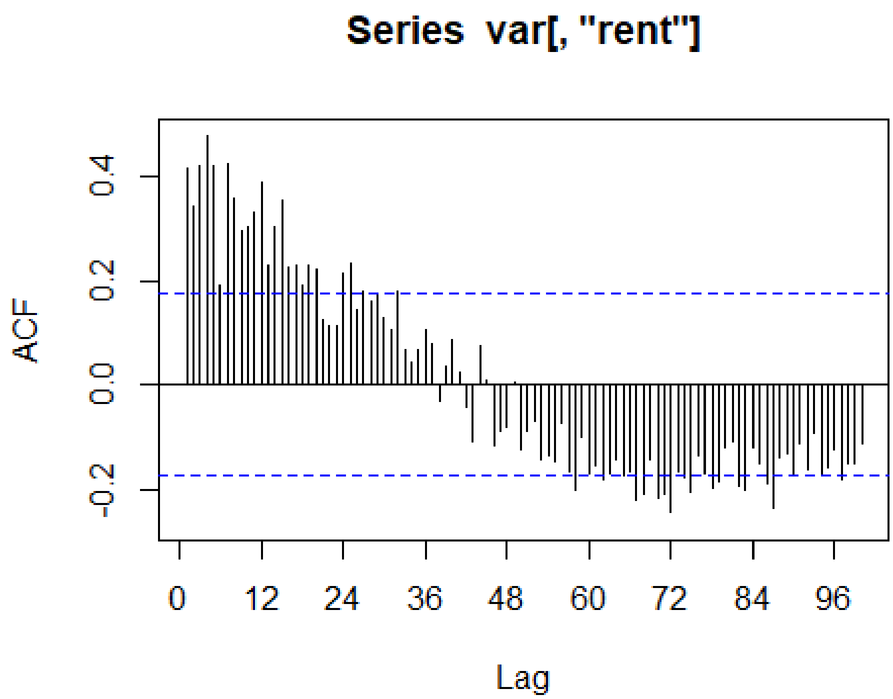
```
Ccf(diff(var[,'rent'],1), diff(var[,'crime'],1), lag.max = 100)
```



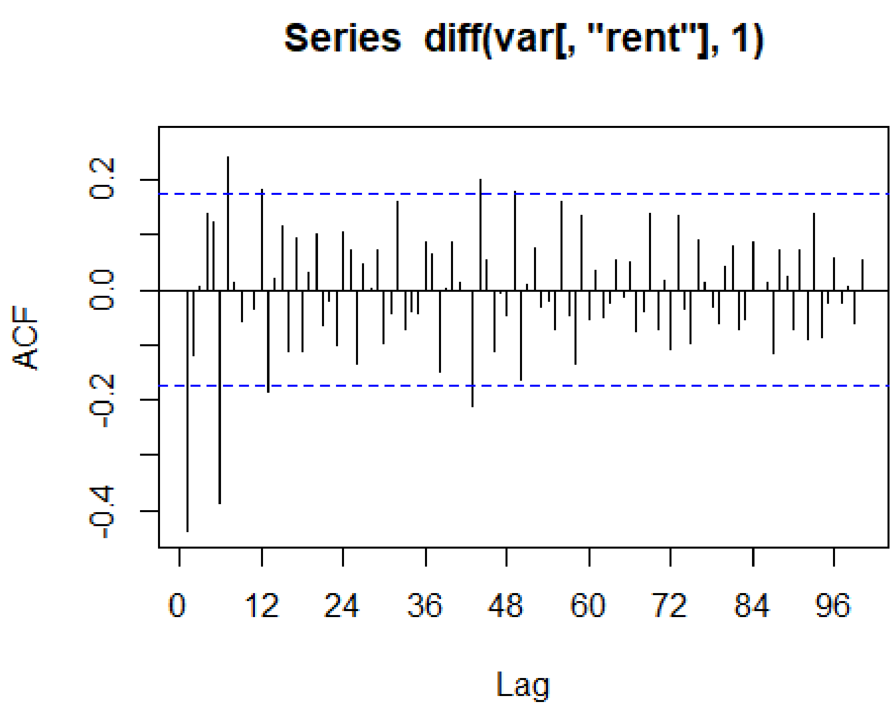

```
Ccf(diff(var[,'rent'],1), var[,'crime'], lag.max = 100)
```



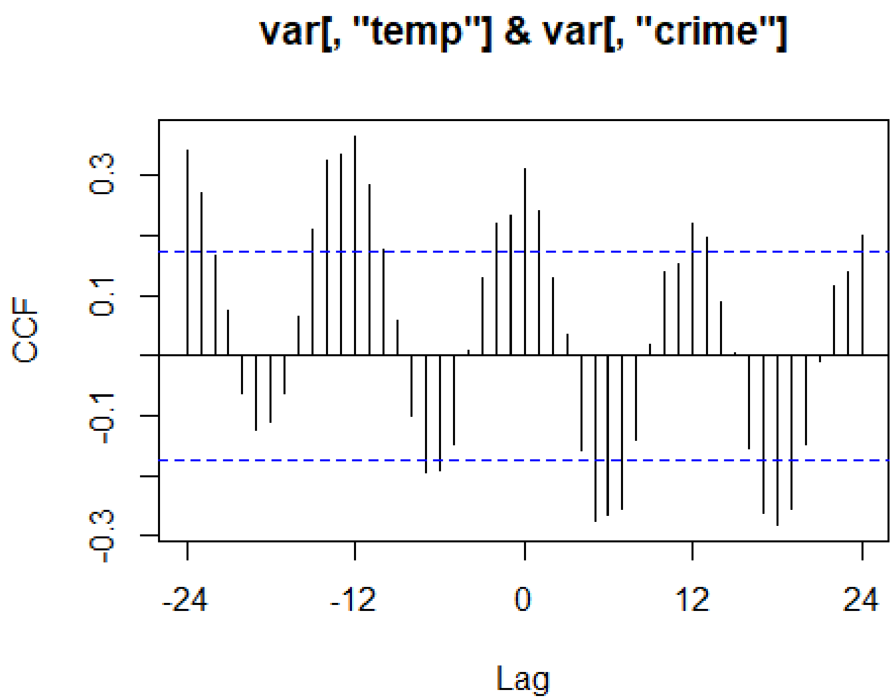
```r
Acf(var[,'rent'], lag.max = 100) # trend present
```



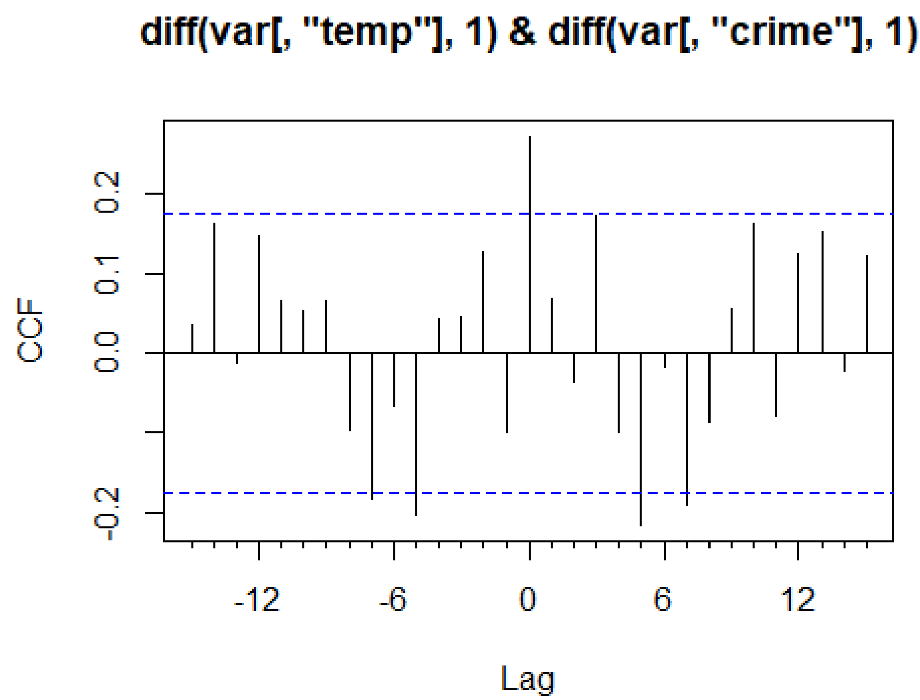

```r
Acf(diff(var[,'rent'],1), lag.max = 100)
```



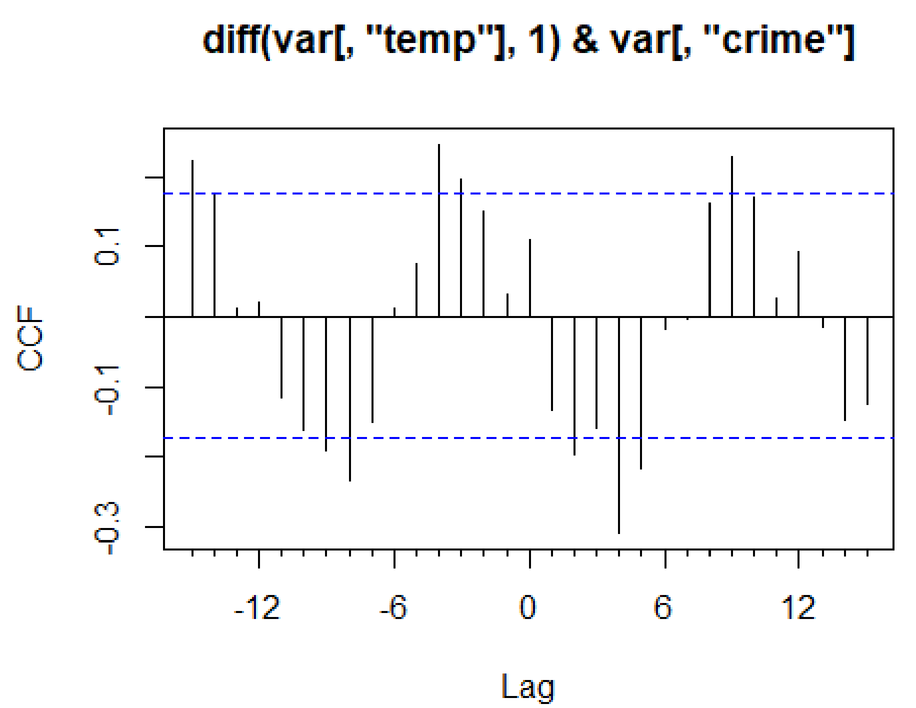
```r
# temp and crime
Ccf(var[,'temp'], var[,'crime'], lag.max = 24) # lag 12
```



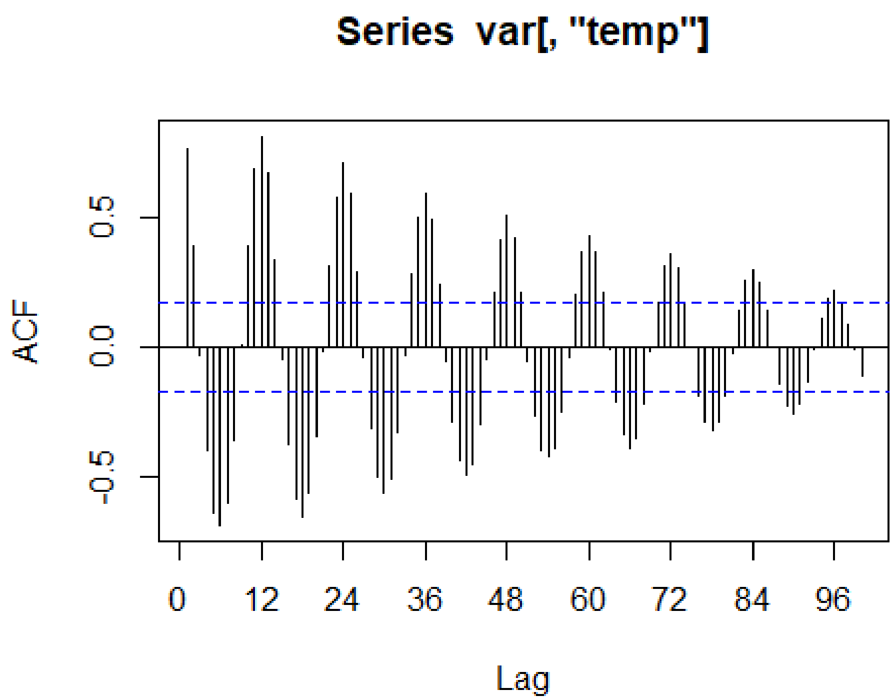

```r
Ccf(diff(var[,'temp'],1), diff(var[,'crime'],1), lag.max = 15)
```



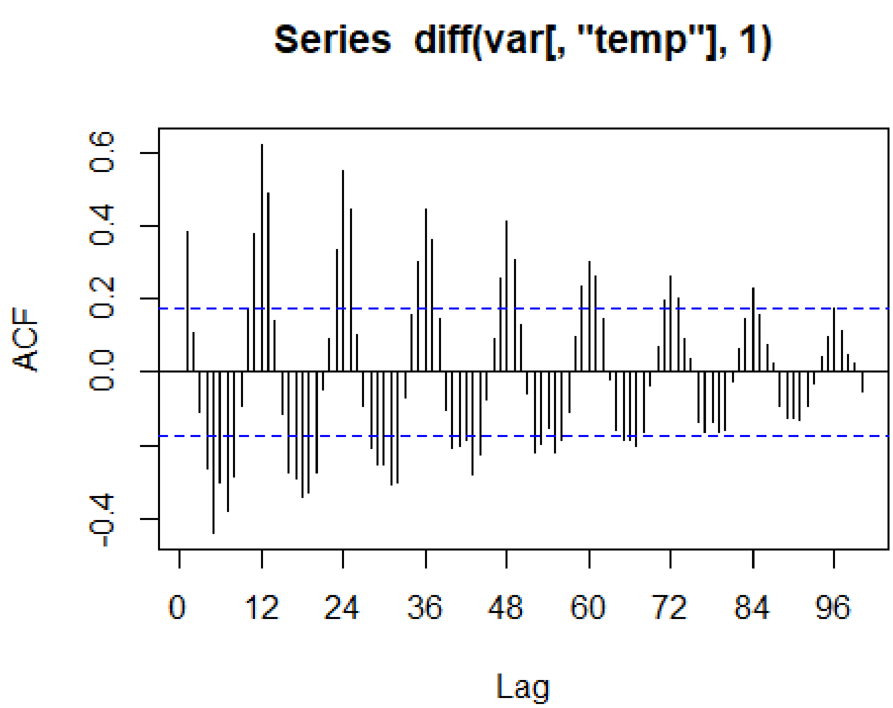

```r
Ccf(diff(var[,'temp'],1), var[,'crime'], lag.max = 15) # differenced temp
```



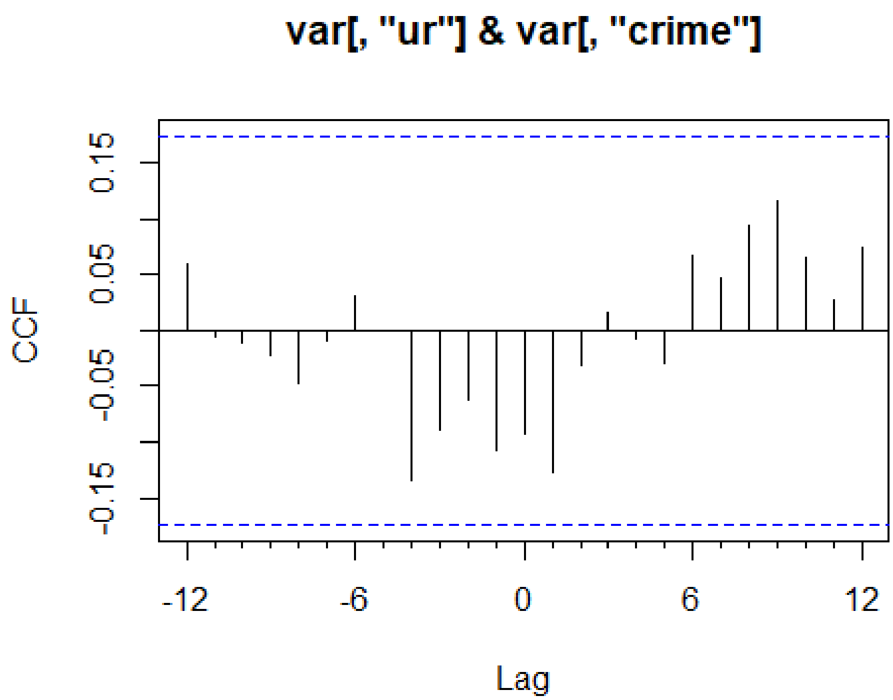

```r
Acf(var[,'temp'], lag.max = 100)
```



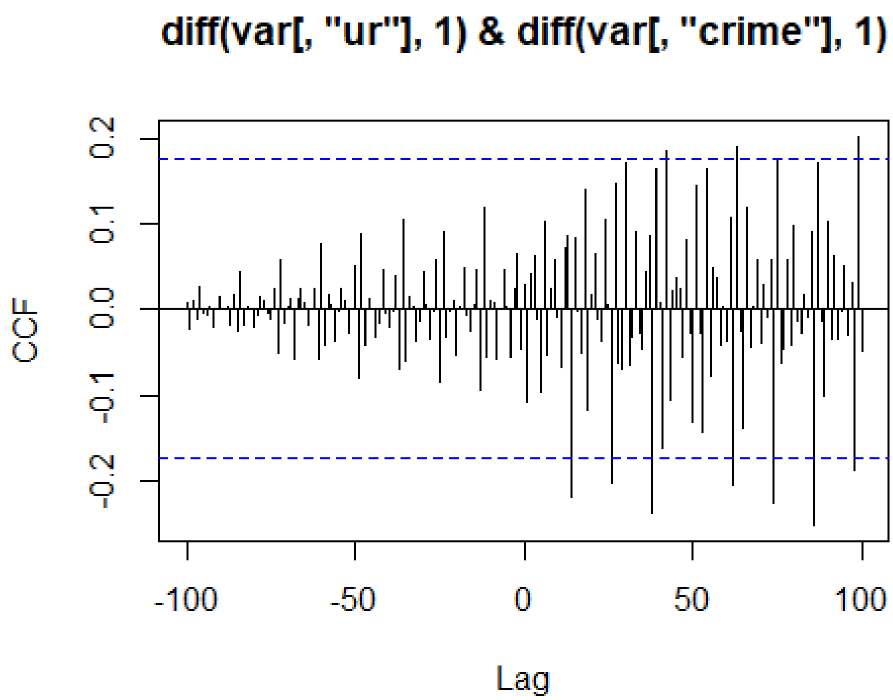

```r
Acf(diff(var[,'temp'],1), lag.max = 100) # seasonality present
```



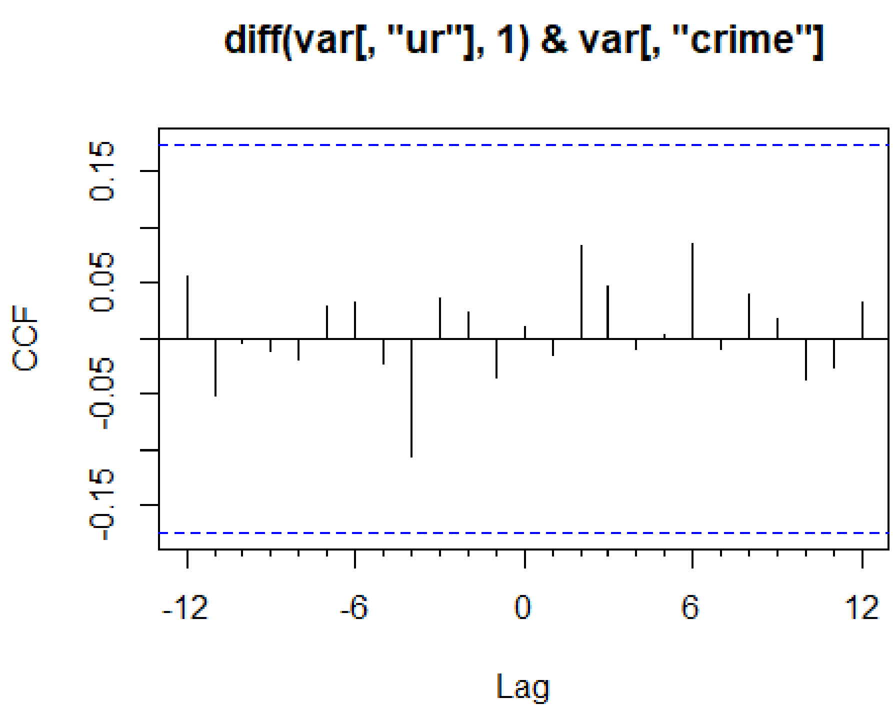
```r
# ur and crime
Ccf(var[,'ur'], var[,'crime'], lag.max = 12) # No Significant correlation
```



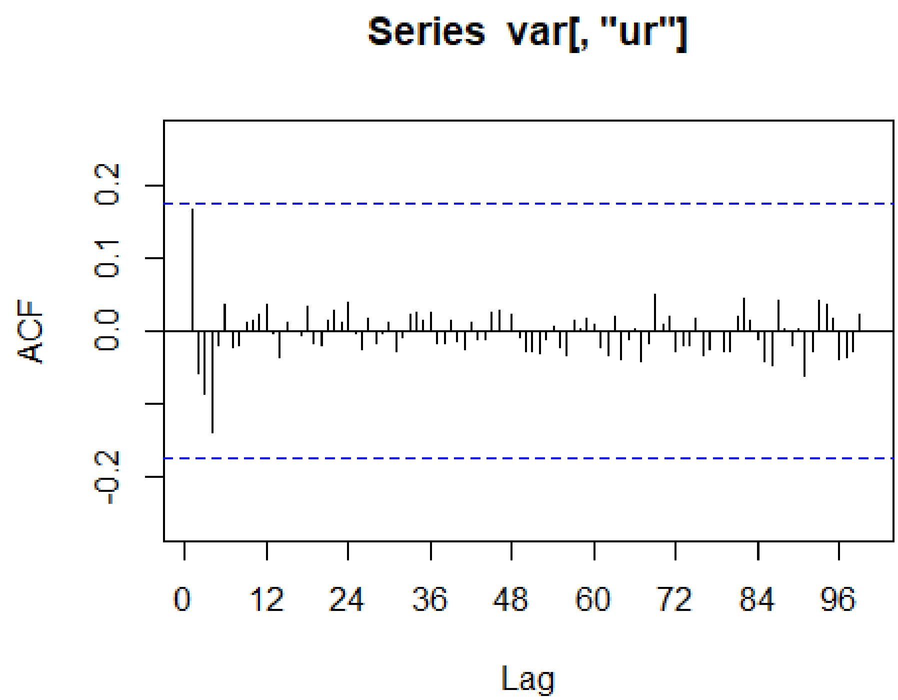

```r
Ccf(diff(var[,'ur'],1), diff(var[,'crime'],1), lag.max = 100)
```



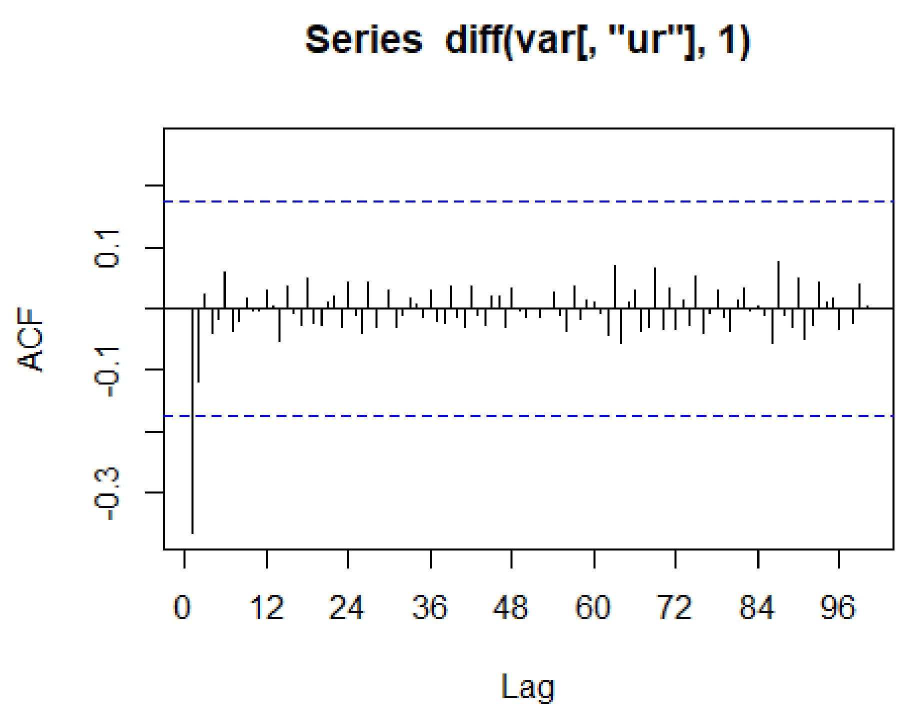

```
Ccf(diff(var[,'ur'],1), var[,'crime'], lag.max = 12)
```



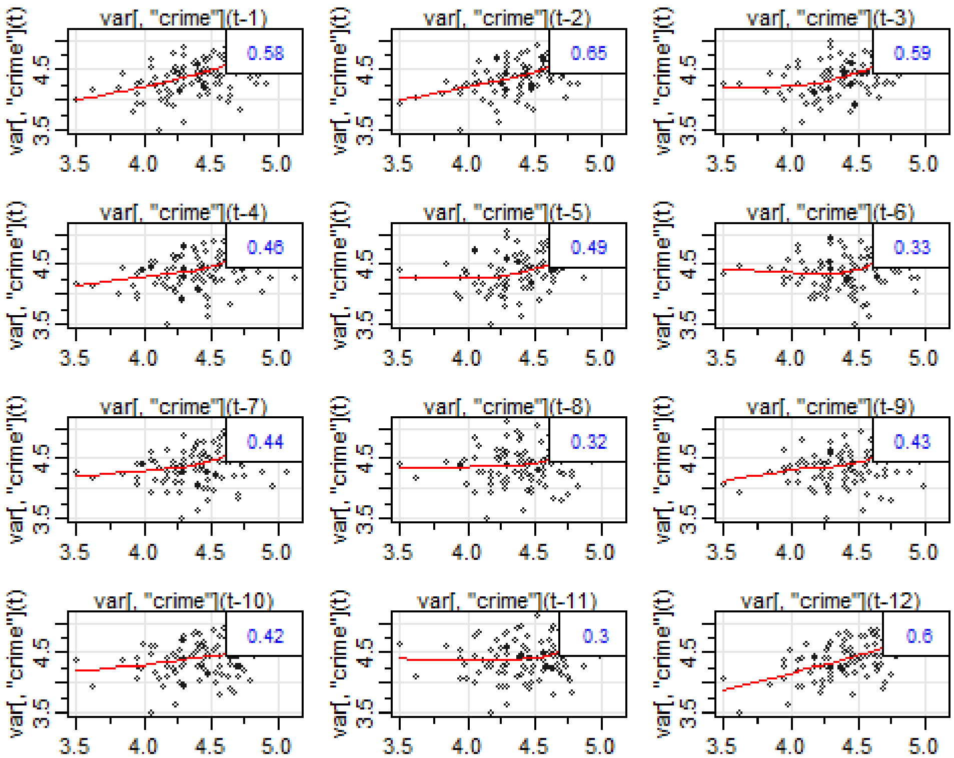

```
Acf(var[,'ur'], lag.max = 100)
```



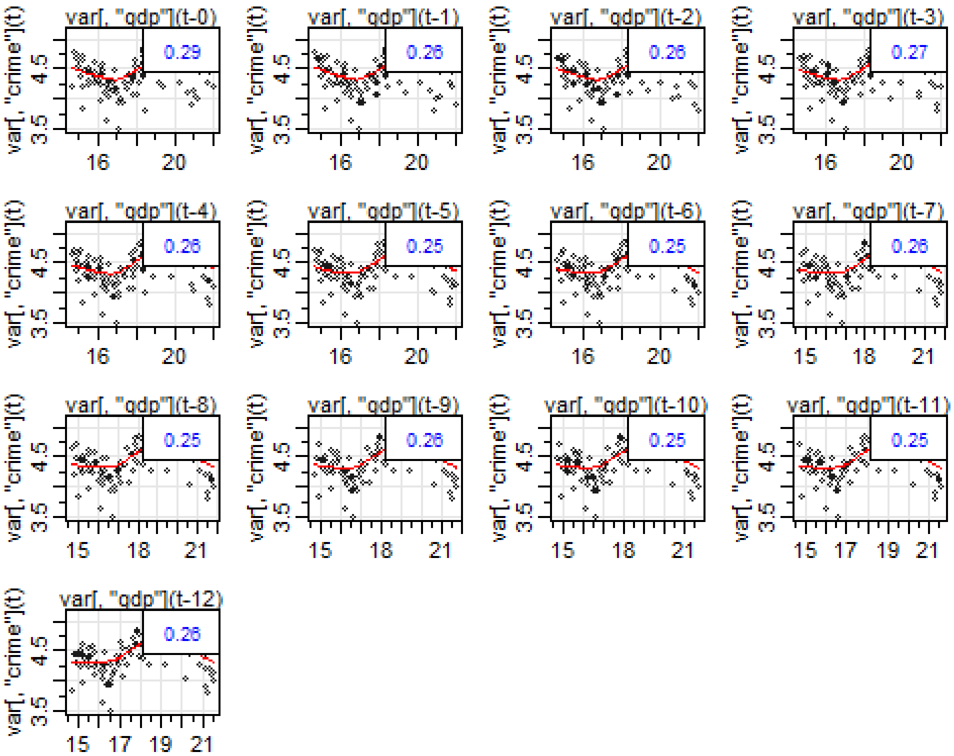

```
Acf(diff(var[,'ur'],1), lag.max = 100)
```



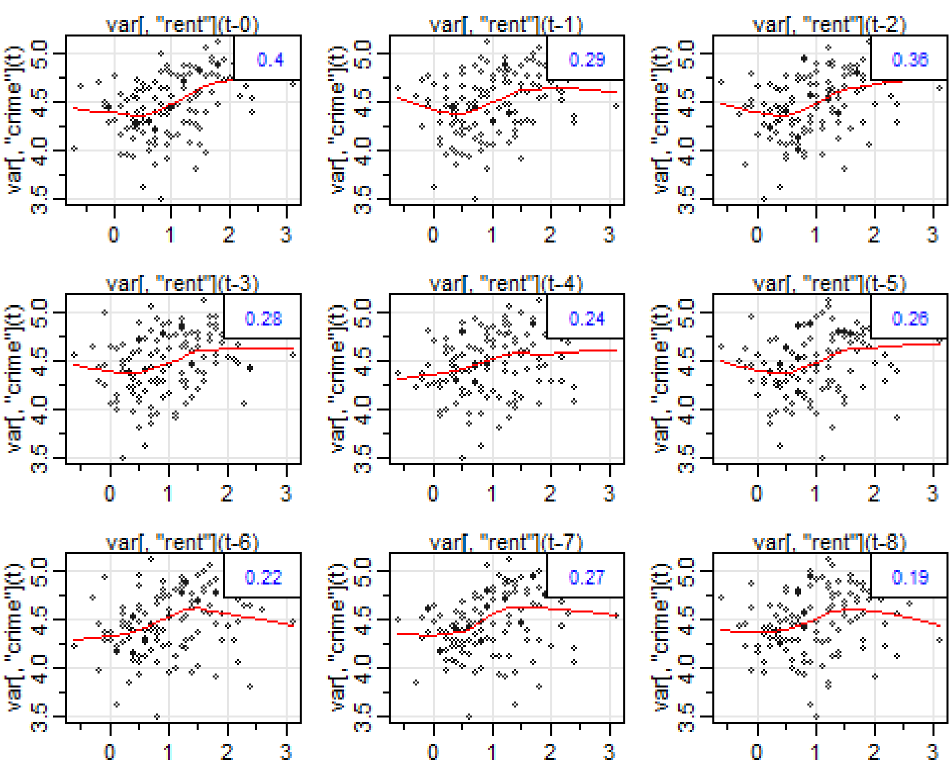
```
# Examining the lagged correlations
lag1.plot(var[,'crime'],12) # high correlations at t-2 and seasonal
(lag12,24)
```



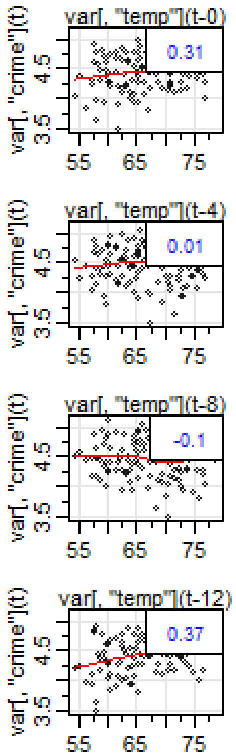

```
lag2.plot(var[,'gdp'], var[,'crime'], 12) # t-12
```




```
lag2.plot(var[,'rent'], var[,'crime'], 8) # t-2
```




```
lag2.plot(var[,'temp'], var[,'crime'], 12) # seasonal (lag12,24)
```



*** BUILDING REGRESSION MODELS ***

Using gdp only

```
m1 = tslm(crime ~ gdp, data=var.train)
summary(m1)$adj.r.squared

## [1] 0.2491991

m1.2019.pred=predict(m1, newdata=var.test)

accuracy(m1$fitted.values, var.train[,'crime']) # training accuracy

##                         ME      RMSE       MAE        MPE     MAPE      ACF1
## Test set -8.340694e-18 0.2723849 0.2158582 -0.3935836 4.941707 0.4341312
##          Theil's U
## Test set 0.9438629

accuracy(m1.2019.pred, var.test[,'crime']) # testing accuracy

##                     ME     RMSE       MAE       MPE     MAPE       ACF1
Theil's U
## Test set -0.3266486 0.376727 0.3266486 -7.426626 7.426626 -0.1481748
1.243103
```

Using rent only

```
m2 = tslm(crime ~ rent, data=var.train)
summary(m2)$adj.r.squared

## [1] 0.1817808

m2.2019.pred=predict(m2, newdata=var.test)

accuracy(m2$fitted.values, var.train[,'crime']) # training accuracy

##                          ME      RMSE      MAE        MPE     MAPE      ACF1
## Test set -4.146454e-17 0.2843514 0.229493 -0.4263338 5.236649 0.4197819
##           Theil's U
## Test set 0.9758805

accuracy(m2.2019.pred, var.test[,'crime']) # testing accuracy

##                      ME      RMSE       MAE       MPE     MAPE       ACF1
Theil's U
## Test set -0.127727 0.2422041 0.1947251 -2.985188 4.421762 0.00834406
0.818078
```

Using temp only

```
m3 = tslm(crime ~ temp, data=var.train)
summary(m3)$adj.r.squared

## [1] 0.118109

m3.2019.pred=predict(m3, newdata=var.test)

accuracy(m3$fitted.values, var.train[,'crime']) # training accuracy

##                          ME      RMSE       MAE        MPE     MAPE      ACF1
## Test set -1.667037e-17 0.2952079 0.2468748 -0.4524952 5.588578 0.5432081
##           Theil's U
## Test set  1.012578

accuracy(m3.2019.pred, var.test[,'crime']) # testing accuracy

##                       ME      RMSE       MAE       MPE     MAPE       ACF1
Theil's U
## Test set 0.03229881 0.1663518 0.1404787 0.5995584 3.128798 -0.3584152
0.5305206
```

Using ur only

```
m4 = tslm(crime ~ ur, data=var.train)
summary(m4)$adj.r.squared

## [1] 0.07347737
```

```
m4.2019.pred=predict(m4, newdata=var.test)

accuracy(m4$fitted.values, var.train[,'crime']) # training accuracy

##                       ME      RMSE       MAE        MPE     MAPE      ACF1
## Test set -1.667012e-17 0.3025858 0.2488187 -0.4746256 5.635219 0.6350411
##          Theil's U
## Test set  1.033936

accuracy(m4.2019.pred, var.test[,'crime']) # testing accuracy

##                    ME      RMSE       MAE       MPE     MAPE       ACF1
Theil's U
## Test set 0.01231549 0.1575012 0.1218608 0.1344477 2.743575 0.04229078
0.5267305
```

GDP+UR

```
m5 = tslm(crime ~ ur+gdp, data=var.train)
summary(m5)$adj.r.squared

## [1] 0.3173708

m5.2019.pred=predict(m5, newdata=var.test)

accuracy(m5$fitted.values, var.train[,'crime']) # training accuracy

##                       ME      RMSE       MAE        MPE     MAPE     ACF1
## Test set -3.321712e-17 0.2584848 0.2060427 -0.3539239 4.708619 0.503192
##          Theil's U
## Test set 0.8976684

accuracy(m5.2019.pred, var.test[,'crime']) # testing accuracy

##                    ME      RMSE       MAE       MPE     MAPE       ACF1
Theil's U
## Test set -0.3302436 0.3656462 0.3302436 -7.465629 7.465629 0.02243306
1.172483
```

GDP+rent

```
m6 = tslm(crime ~ rent+gdp, data=var.train)
summary(m6)$adj.r.squared

## [1] 0.2573203

m6.2019.pred=predict(m6, newdata=var.test)

accuracy(m6$fitted.values, var.train[,'crime']) # training accuracy
```

```
##                         ME      RMSE       MAE        MPE     MAPE      ACF1
## Test set -8.340694e-18 0.2696146 0.2153545 -0.3863243 4.929097 0.4315399
##          Theil's U
## Test set 0.9331662
```

```
accuracy(m6.2019.pred, var.test[,'crime']) # testing accuracy
```

```
##                   ME      RMSE       MAE       MPE     MAPE        ACF1
## Test set -0.3054196 0.3557007 0.3054196 -6.944429 6.944429 -0.07161175
##          Theil's U
## Test set  1.184665
```

GDP+temp

```
m7 = tslm(crime ~ temp+gdp, data=var.train)
summary(m7)$adj.r.squared
```

```
## [1] 0.3117762
```

```
m7.2019.pred=predict(m7, newdata=var.test)
```

```
accuracy(m7$fitted.values, var.train[,'crime']) # training accuracy
```

```
##                        ME      RMSE       MAE        MPE     MAPE      ACF1
## Test set 8.340821e-18 0.2595419 0.2062615 -0.3569174 4.713992 0.4163062
##          Theil's U
## Test set  0.902694
```

```
accuracy(m7.2019.pred, var.test[,'crime']) # testing accuracy
```

```
##                 ME      RMSE     MAE       MPE     MAPE       ACF1 Theil's
U
## Test set -0.28295 0.3281038 0.28295 -6.408765 6.408765 -0.3205334
1.090483
```

GDP+rent+UR

```
m8 = tslm(crime ~ ur+rent+gdp, data=var.train)
summary(m8)$adj.r.squared
```

```
## [1] 0.3177159
```

```
m8.2019.pred=predict(m8, newdata=var.test)
```

```
accuracy(m8$fitted.values, var.train[,'crime']) # training accuracy
```

```
##                         ME      RMSE       MAE        MPE     MAPE     ACF1
## Test set -8.342214e-18 0.2571741 0.2044851 -0.3508414 4.676241 0.495573
##          Theil's U
## Test set 0.8928892
```

```r
accuracy(m8.2019.pred, var.test[,'crime']) # testing accuracy
```

```
##                     ME    RMSE       MAE       MPE     MAPE       ACF1 Theil's
U
## Test set -0.3156657 0.3506 0.3156657 -7.136491 7.136491 0.06023038
1.135582
```

model 7 & 8 are able to explain around 31% of the variation in crime. Now, we look for lead relationships based on the acf and ccf seen above and incorporate those in the model to see if it boosts the performance.

*** BUILDING ADL MODELS ***

```r
# Adding lead variables based on correlations obtained from acf and ccf

var.adl = ts.intersect(crime=crime.ts,
                       gdp=gdp.ts,
                       rent=newrent.ts,
                       temp=temp.ts,
                       ur=newur.ts,
                       crime.lead2=stats::lag(crime.ts,-2),
                       gdp.lead12=stats::lag(gdp.ts, -12),
                       rent.lead2=stats::lag(newrent.ts, -2),
                       temp.lead12=stats::lag(temp.ts, -12)
                      )

var.adl.train=window(var.adl, end=c(2018,12))
var.adl.test=window(var.adl, start=c(2019, 1), end=c(2019,12))
var.adl.valid=window(var.adl, start=c(2020,1))
var.adl.traintest=window(var.adl, end=c(2019,12))
```

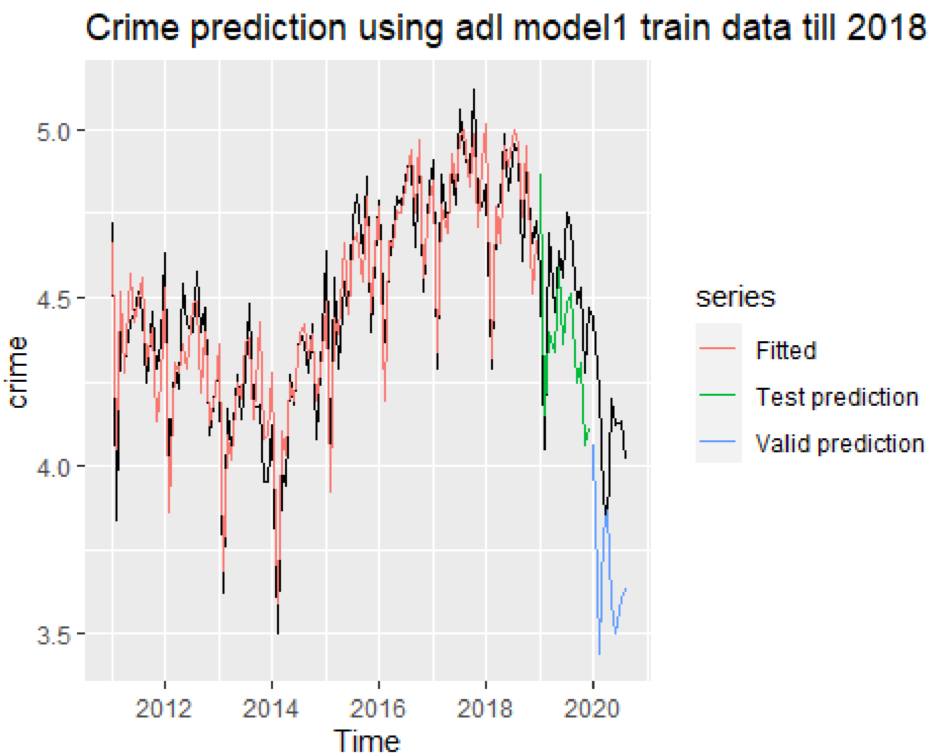
trying trend, season and all variables in the adl (gdp had a cubic relationship with crime)

```r
# Fitting model on train (till 2018)
adl.m1 = tslm(crime ~ trend + season +
crime.lead2+gdp.lead12+I(gdp.lead12^2)+I(gdp.lead12^3)
+rent.lead2+temp.lead12,data=var.adl.train)
summary(adl.m1)
```

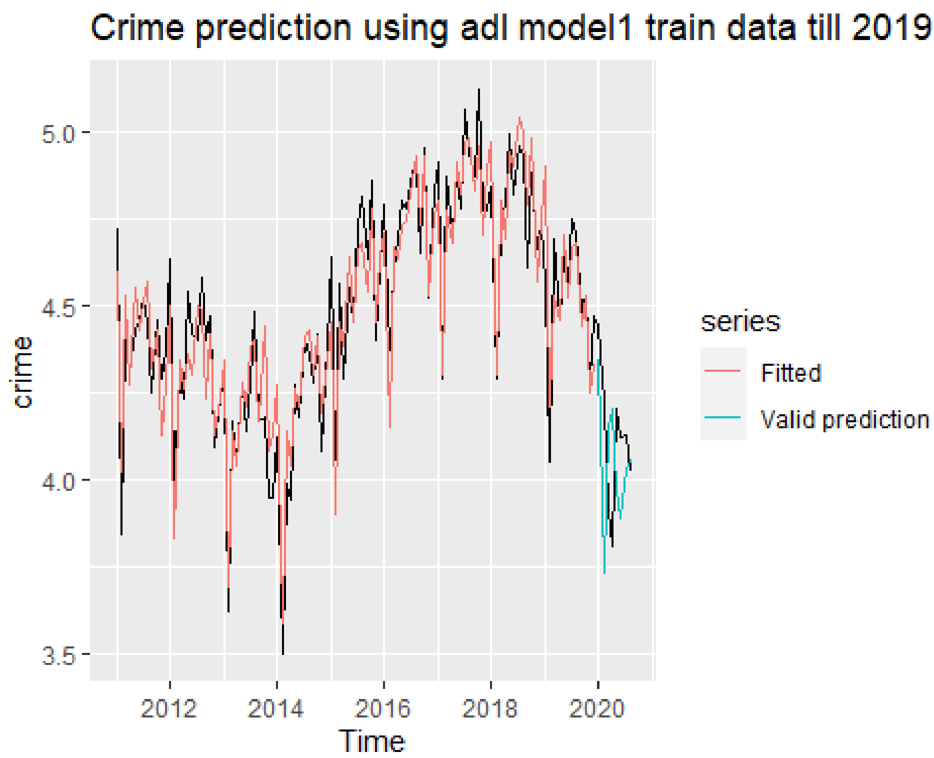
```
##
## Call:
## tslm(formula = crime ~ trend + season + crime.lead2 + gdp.lead12 +
##     I(gdp.lead12^2) + I(gdp.lead12^3) + rent.lead2 + temp.lead12,
##     data = var.adl.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.24865 -0.06546  0.01235  0.06153  0.18207
##
```

```
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     120.772251  29.259501   4.128 9.20e-05 ***
## trend             0.003989   0.006219   0.641 0.523111
## season2          -0.692454   0.054963 -12.599  < 2e-16 ***
## season3          -0.321561   0.065420  -4.915 4.89e-06 ***
## season4          -0.124689   0.065472  -1.904 0.060586 .
## season5          -0.156245   0.067815  -2.304 0.023922 *
## season6          -0.202259   0.079415  -2.547 0.012865 *
## season7          -0.180294   0.101047  -1.784 0.078322 .
## season8          -0.132197   0.104095  -1.270 0.207923
## season9          -0.421152   0.109195  -3.857 0.000237 ***
## season10         -0.207634   0.092702  -2.240 0.027989 *
## season11         -0.375463   0.061128  -6.142 3.32e-08 ***
## season12         -0.300723   0.064352  -4.673 1.24e-05 ***
## crime.lead2       0.467634   0.096586   4.842 6.50e-06 ***
## gdp.lead12      -20.467061   5.025049  -4.073 0.000112 ***
## I(gdp.lead12^2)   1.170566   0.289697   4.041 0.000125 ***
## I(gdp.lead12^3)  -0.022229   0.005527  -4.022 0.000134 ***
## rent.lead2        0.046518   0.024824   1.874 0.064736 .
## temp.lead12       0.006566   0.005791   1.134 0.260364
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1032 on 77 degrees of freedom
## Multiple R-squared:  0.9202, Adjusted R-squared:  0.9015
## F-statistic:  49.3 on 18 and 77 DF,  p-value: < 2.2e-16

# Predicting values for test(2019) using train
adl.m1.pred.test=round(forecast(adl.m1, newdata =
data.frame(var.adl.test))$mean,2)

# Predicting values for valid(2020) using train
adl.m1.pred.valid.train=ts(as.numeric(round(forecast(adl.m1, newdata =
data.frame(var.adl.valid))$mean,2)),start = c(2020,1),end =
c(2020,8),frequency = 12)

# Fitting model on traintest (till 2019)
adl.m1.traintest=tslm(crime ~ trend + season +
crime.lead2+gdp.lead12+I(gdp.lead12^2)+I(gdp.lead12^3)
+rent.lead2+temp.lead12,data=var.adl.traintest)

# Predicting values for valid(2020) using traintest
adl.m1.pred.valid.traintest=round(forecast(adl.m1.traintest, newdata =
data.frame(var.adl.valid))$mean,2)

# Finding accuracies
accuracy(round(adl.m1$fitted.values,2), var.adl.train[,'crime']) # training
accuracy
```

```
##                      ME       RMSE         MAE         MPE      MAPE
ACF1
## Test set -0.0002083333 0.09259005 0.07708333 -0.05147531 1.743972
0.3166125
##          Theil's U
## Test set 0.3128217

accuracy(adl.m1.pred.test, var.adl.test[,'crime']) # testing accuracy

##                  ME     RMSE       MAE      MPE     MAPE      ACF1 Theil's U
## Test set 0.1466667 0.22971 0.2133333 3.201552 4.707167 0.3016945 0.7335004

accuracy(adl.m1.pred.valid.train, var.adl.valid[,'crime']) #valid accuracy
(train)

##             ME      RMSE   MAE      MPE     MAPE        ACF1 Theil's U
## Test set 0.42 0.4947221 0.435 10.04841 10.44211 -0.02139722  2.284137

accuracy(adl.m1.pred.valid.traintest, var.adl.valid[,'crime']) #valid
accuracy (traintest)

##             ME      RMSE    MAE      MPE     MAPE        ACF1 Theil's U
## Test set 0.06 0.2686075 0.2275 1.246787 5.589125 -0.03938731  1.276484

# plotting

# train,test and valid (train)
autoplot(var.adl[,'crime'],ylab = "crime",main = "Crime prediction using adl
model1 train data till 2018") +
  autolayer(round(adl.m1$fitted.values,2),series ="Fitted") +
  autolayer(adl.m1.pred.test,series ="Test prediction")+
  autolayer(adl.m1.pred.valid.train,series ="Valid prediction")
```




```
# valid (traintest)
autoplot(var.adl[,'crime'],ylab = "crime",main = "Crime prediction using adl
model1 train data till 2019") +
  autolayer(round(adl.m1.traintest$fitted.values,2),series ="Fitted") +
  autolayer(adl.m1.pred.valid.traintest,series ="Valid prediction")
```

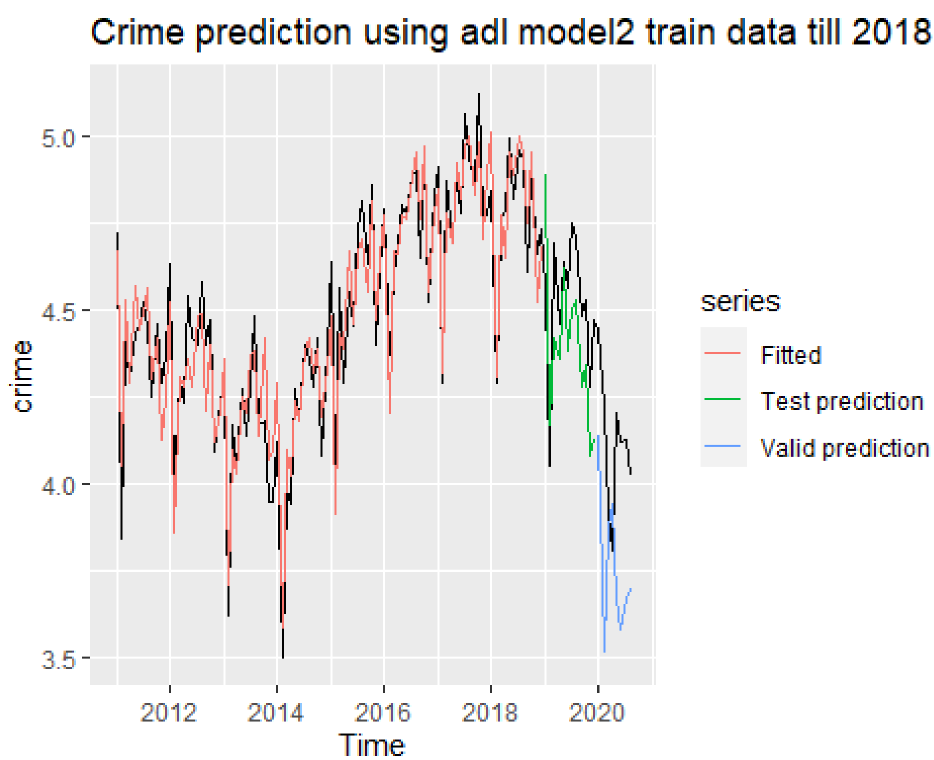


removing trend due to high p value

```
# Fitting model on train (till 2018)
adl.m2 = tslm(crime ~ season +
crime.lead2+gdp.lead12+I(gdp.lead12^2)+I(gdp.lead12^3)
+rent.lead2+temp.lead12,data=var.adl.train)
summary(adl.m2)

##
## Call:
## tslm(formula = crime ~ season + crime.lead2 + gdp.lead12 + I(gdp.lead12^2)
+
##     I(gdp.lead12^3) + rent.lead2 + temp.lead12, data = var.adl.train)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.238108 -0.069213  0.009543  0.061379  0.186787
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     120.677551  29.148542   4.140 8.71e-05 ***
## season2          -0.691950   0.054750 -12.638  < 2e-16 ***
## season3          -0.319594   0.065101  -4.909 4.92e-06 ***
## season4          -0.126847   0.065139  -1.947 0.055093 .
## season5          -0.154727   0.067518  -2.292 0.024624 *
## season6          -0.199822   0.079025  -2.529 0.013471 *
```

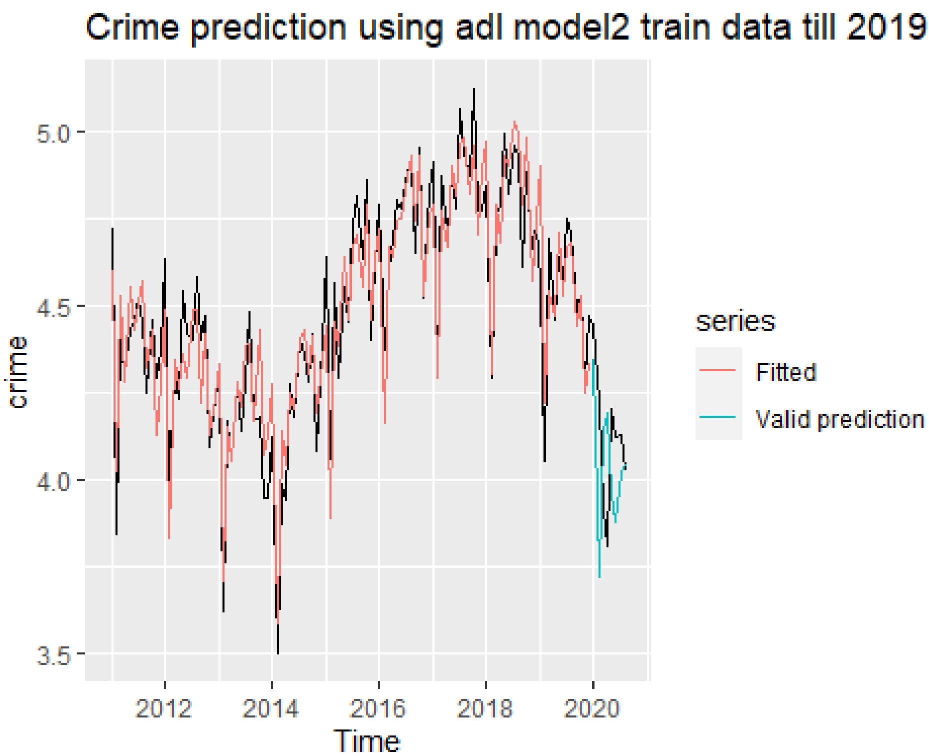
```
## season7          -0.175410   0.100379  -1.747 0.084490 .
## season8          -0.127838   0.103481  -1.235 0.220396
## season9          -0.414884   0.108346  -3.829 0.000258 ***
## season10         -0.204105   0.092189  -2.214 0.029751 *
## season11         -0.376536   0.060875  -6.185 2.66e-08 ***
## season12         -0.303716   0.063940  -4.750 9.09e-06 ***
## crime.lead2       0.462061   0.095831   4.822 6.90e-06 ***
## gdp.lead12      -20.575545   5.003220  -4.112 9.61e-05 ***
## I(gdp.lead12^2)   1.182329   0.288023   4.105 9.87e-05 ***
## I(gdp.lead12^3)  -0.022478   0.005493  -4.092 0.000103 ***
## rent.lead2        0.047373   0.024694   1.918 0.058721 .
## temp.lead12       0.006035   0.005710   1.057 0.293758
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1028 on 78 degrees of freedom
## Multiple R-squared:  0.9197, Adjusted R-squared:  0.9022
## F-statistic: 52.57 on 17 and 78 DF,  p-value: < 2.2e-16

# Predicting values for test(2019) using train
adl.m2.pred.test=round(forecast(adl.m2, newdata =
data.frame(var.adl.test))$mean,2)

# Predicting values for valid(2020) using train
adl.m2.pred.valid.train=ts(as.numeric(round(forecast(adl.m2, newdata =
data.frame(var.adl.valid))$mean,2)),start = c(2020,1),end =
c(2020,8),frequency = 12)

# Fitting model on traintest (till 2019)
adl.m2.traintest=tslm(crime ~ season +
crime.lead2+gdp.lead12+I(gdp.lead12^2)+I(gdp.lead12^3)
+rent.lead2+temp.lead12,data=var.adl.traintest)

# Predicting values for valid(2020) using traintest
adl.m2.pred.valid.traintest=round(forecast(adl.m2.traintest, newdata =
data.frame(var.adl.valid))$mean,2)

# Finding accuracies
accuracy(round(adl.m2$fitted.values,2), var.adl.train[,'crime']) # training
accuracy

##                       ME       RMSE        MAE         MPE     MAPE
ACF1
## Test set -0.0002083333 0.09296057 0.07666667 -0.05183841 1.735411
0.3245857
##          Theil's U
## Test set 0.3143852

accuracy(adl.m2.pred.test, var.adl.test[,'crime']) # testing accuracy
```

```
##                    ME      RMSE    MAE      MPE     MAPE      ACF1 Theil's U
## Test set 0.1241667 0.215851 0.1975 2.702905 4.363763 0.2958035 0.6737725

accuracy(adl.m2.pred.valid.train, var.adl.valid[,'crime']) #valid accuracy
(train)

##                ME      RMSE    MAE     MPE     MAPE        ACF1 Theil's U
## Test set 0.345 0.4308422 0.3775 8.22243 9.075448 -0.02181869  2.001933

accuracy(adl.m2.pred.valid.traintest, var.adl.valid[,'crime']) #valid
accuracy (traintest)

##               ME      RMSE    MAE       MPE     MAPE        ACF1 Theil's U
## Test set 0.07 0.2704163 0.2275 1.493743 5.582147 -0.04672041  1.283847

# plotting

# train,test and valid (train)
autoplot(var.adl[,'crime'],ylab = "crime",main = "Crime prediction using adl
model2 train data till 2018") +
  autolayer(round(adl.m2$fitted.values,2),series ="Fitted") +
  autolayer(adl.m2.pred.test,series ="Test prediction")+
  autolayer(adl.m2.pred.valid.train,series ="Valid prediction")
```




```
# valid (traintest)
autoplot(var.adl[,'crime'],ylab = "crime",main = "Crime prediction using adl
model2 train data till 2019") +
```

```
autolayer(round(adl.m2.traintest$fitted.values,2),series ="Fitted") +
autolayer(adl.m2.pred.valid.traintest,series ="Valid prediction")
```

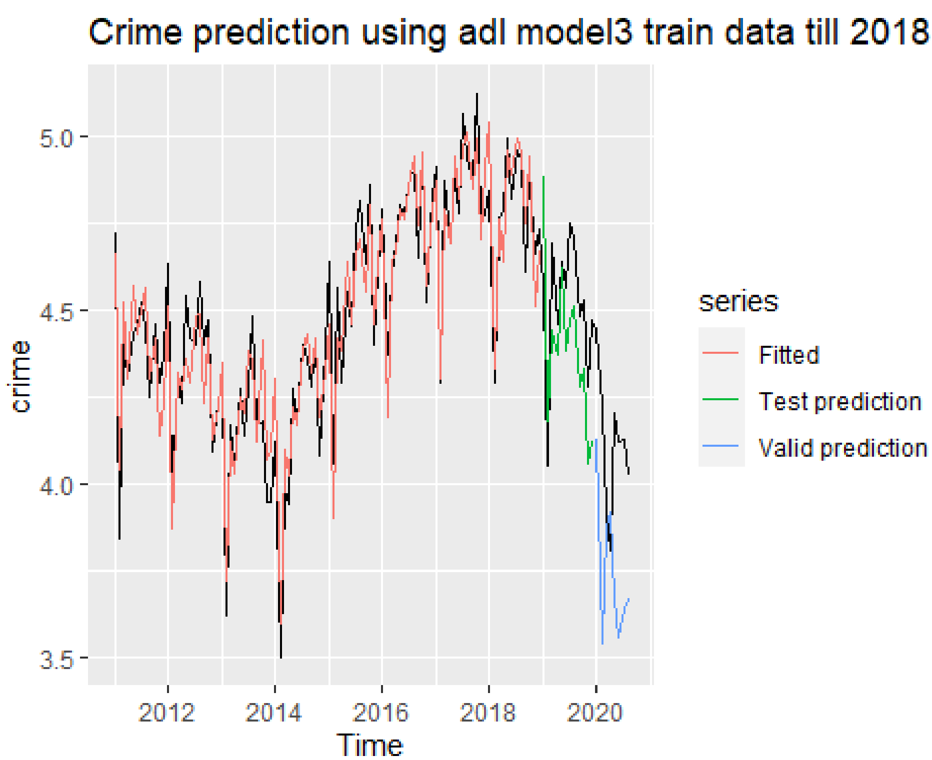


removing temp.lead12 due to high p value

```
# Fitting model on train (till 2018)
adl.m3 = tslm(crime ~ season +
crime.lead2+gdp.lead12+I(gdp.lead12^2)+I(gdp.lead12^3)
+rent.lead2,data=var.adl.train)
summary(adl.m3)

##
## Call:
## tslm(formula = crime ~ season + crime.lead2 + gdp.lead12 + I(gdp.lead12^2)
+
##     I(gdp.lead12^3) + rent.lead2, data = var.adl.train)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.227927 -0.063740  0.004235  0.064070  0.183462
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     126.045121  28.724127   4.388 3.50e-05 ***
## season2          -0.686994   0.054589 -12.585  < 2e-16 ***
## season3          -0.296325   0.061312  -4.833 6.49e-06 ***
```

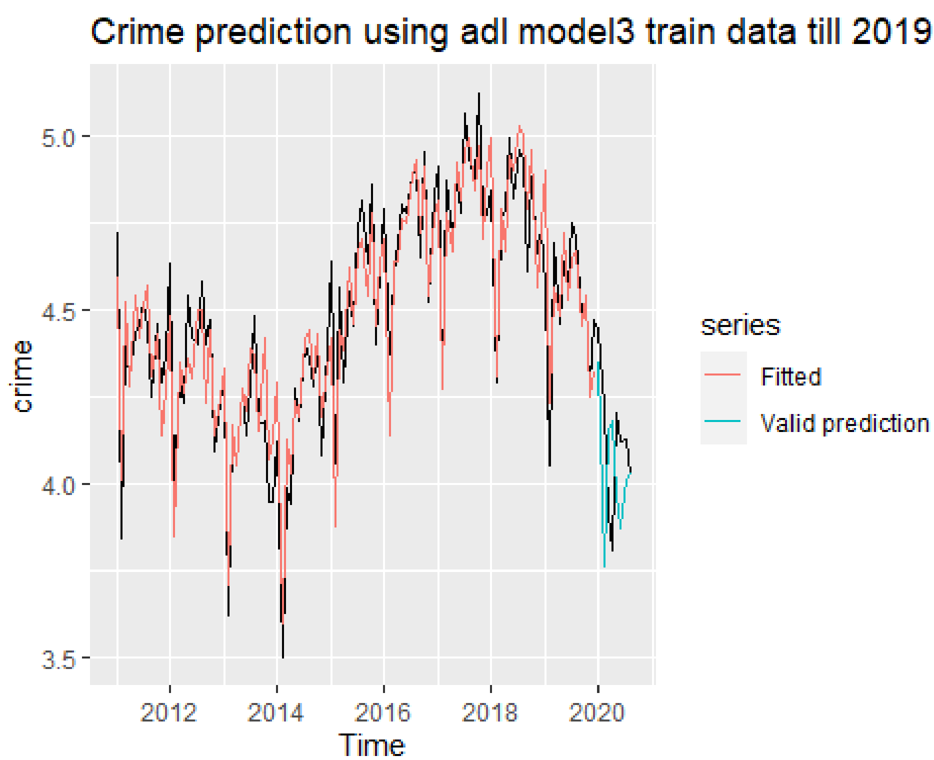
```
## season4          -0.096266   0.058404  -1.648   0.1033
## season5          -0.112006   0.054125  -2.069   0.0418 *
## season6          -0.138160   0.053350  -2.590   0.0114 *
## season7          -0.090214   0.059875  -1.507   0.1359
## season8          -0.035427   0.055404  -0.639   0.5244
## season9          -0.321043   0.062154  -5.165 1.76e-06 ***
## season10         -0.133801   0.063885  -2.094   0.0394 *
## season11         -0.350672   0.055782  -6.286 1.67e-08 ***
## season12         -0.311156   0.063598  -4.893 5.15e-06 ***
## crime.lead2       0.457311   0.095796   4.774 8.16e-06 ***
## gdp.lead12      -21.457192   4.936869  -4.346 4.08e-05 ***
## I(gdp.lead12^2)   1.233711   0.284103   4.342 4.14e-05 ***
## I(gdp.lead12^3)  -0.023465   0.005417  -4.332 4.31e-05 ***
## rent.lead2        0.046919   0.024709   1.899   0.0612 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1029 on 79 degrees of freedom
## Multiple R-squared:  0.9186, Adjusted R-squared:  0.9021
## F-statistic:  55.7 on 16 and 79 DF,  p-value: < 2.2e-16

# Predicting values for test(2019) using train
adl.m3.pred.test=round(forecast(adl.m3, newdata =
data.frame(var.adl.test))$mean,2)

# Predicting values for valid(2020) using train
adl.m3.pred.valid.train=ts(as.numeric(round(forecast(adl.m3, newdata =
data.frame(var.adl.valid))$mean,2)),start = c(2020,1),end =
c(2020,8),frequency = 12)

# Fitting model on traintest (till 2019)
adl.m3.traintest=tslm(crime ~ season +
crime.lead2+gdp.lead12+I(gdp.lead12^2)+I(gdp.lead12^3)
+rent.lead2,data=var.adl.traintest)

# Predicting values for valid(2020) using traintest
adl.m3.pred.valid.traintest=round(forecast(adl.m3.traintest, newdata =
data.frame(var.adl.valid))$mean,2)

# Finding accuracies
accuracy(round(adl.m3$fitted.values,2), var.adl.train[,'crime']) # training
accuracy

##                     ME       RMSE      MAE         MPE     MAPE      ACF1
## Test set 4.626035e-18 0.09327379 0.075625 -0.04764648 1.714415 0.3429119
##          Theil's U
## Test set 0.3151248

accuracy(adl.m3.pred.test, var.adl.test[,'crime']) # testing accuracy
```

```
##                    ME      RMSE    MAE      MPE    MAPE      ACF1 Theil's U
## Test set 0.1291667 0.2209261 0.2025 2.810339 4.47588 0.3407967 0.6947537

accuracy(adl.m3.pred.valid.train, var.adl.valid[,'crime']) #valid accuracy
(train)

##                  ME      RMSE     MAE     MPE     MAPE      ACF1 Theil's U
## Test set 0.35625 0.4372213 0.38375 8.50172 9.223505 0.0212037  2.031476

accuracy(adl.m3.pred.valid.traintest, var.adl.valid[,'crime']) #valid
accuracy (traintest)

##                  ME      RMSE     MAE      MPE     MAPE        ACF1 Theil's U
## Test set 0.06625 0.2607921 0.22125 1.409884 5.434421 -0.02593691  1.242358

# plotting

# train,test and valid (train)
autoplot(var.adl[,'crime'],ylab = "crime",main = "Crime prediction using adl
model3 train data till 2018") +
  autolayer(round(adl.m3$fitted.values,2),series ="Fitted") +
  autolayer(adl.m3.pred.test,series ="Test prediction")+
  autolayer(adl.m3.pred.valid.train,series ="Valid prediction")
```




```
# valid (traintest)
autoplot(var.adl[,'crime'],ylab = "crime",main = "Crime prediction using adl
model3 train data till 2019") +
```

```
autolayer(round(adl.m3.traintest$fitted.values,2),series ="Fitted") +
autolayer(adl.m3.pred.valid.traintest,series ="Valid prediction")
```



removing rent.lead2 due to high p value

```
# Fitting model on train (till 2018)
adl.m4 = tslm(crime ~ season +
crime.lead2+gdp.lead12+I(gdp.lead12^2)+I(gdp.lead12^3),data=var.adl.train)
summary(adl.m4)
```

```
##
## Call:
## tslm(formula = crime ~ season + crime.lead2 + gdp.lead12 + I(gdp.lead12^2)
+
##     I(gdp.lead12^3), data = var.adl.train)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.229239 -0.059711  0.003901  0.066367  0.203115
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)      125.262303  29.185153   4.292 4.92e-05 ***
## season2           -0.705346   0.054594 -12.920  < 2e-16 ***
## season3           -0.308372   0.061968  -4.976 3.64e-06 ***
## season4           -0.100285   0.059309  -1.691  0.09475 .
```

```
## season5          -0.108350   0.054965  -1.971  0.05215 .
## season6          -0.159373   0.053010  -3.006  0.00353 **
## season7          -0.110726   0.059844  -1.850  0.06797 .
## season8          -0.044919   0.056069  -0.801  0.42543
## season9          -0.313618   0.063033  -4.975 3.66e-06 ***
## season10         -0.139367   0.064849  -2.149  0.03465 *
## season11         -0.367166   0.055992  -6.558 4.96e-09 ***
## season12         -0.311524   0.064625  -4.820 6.71e-06 ***
## crime.lead2       0.475237   0.096870   4.906 4.81e-06 ***
## gdp.lead12      -21.361496   5.016362  -4.258 5.57e-05 ***
## I(gdp.lead12^2)   1.229693   0.288684   4.260 5.54e-05 ***
## I(gdp.lead12^3)  -0.023406   0.005504  -4.252 5.69e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1045 on 80 degrees of freedom
## Multiple R-squared:  0.9149, Adjusted R-squared:  0.8989
## F-statistic: 57.31 on 15 and 80 DF,  p-value: < 2.2e-16
```

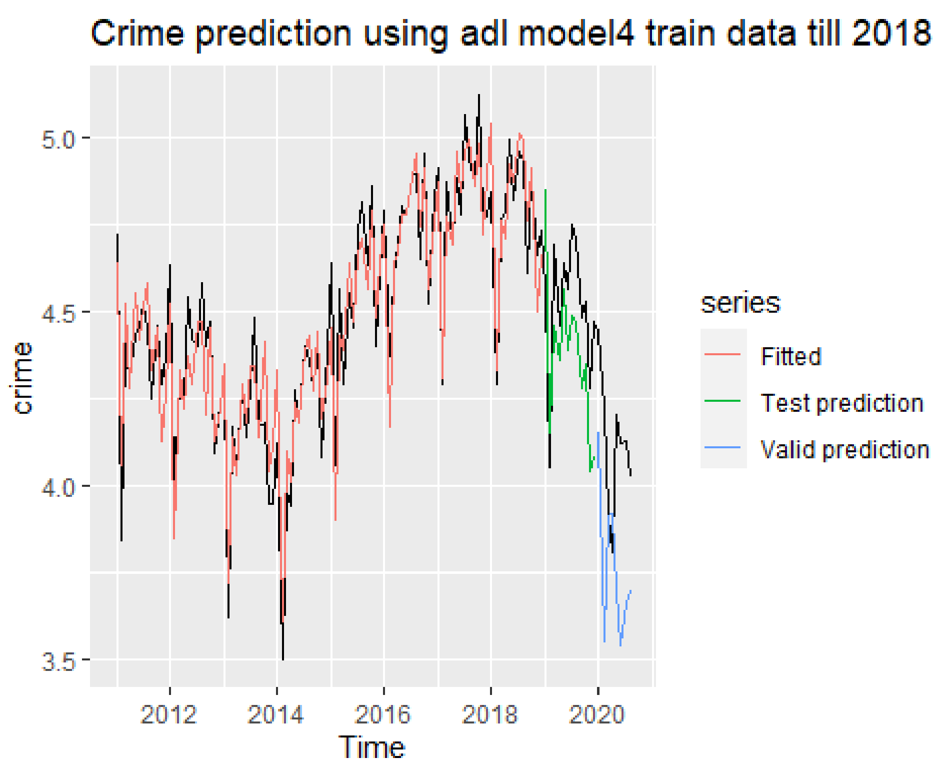
```r
# Predicting values for test(2019) using train
adl.m4.pred.test=round(forecast(adl.m4, newdata =
data.frame(var.adl.test))$mean,2)

# Predicting values for valid(2020) using train
adl.m4.pred.valid.train=ts(as.numeric(round(forecast(adl.m4, newdata =
data.frame(var.adl.valid))$mean,2)),start = c(2020,1),end =
c(2020,8),frequency = 12)

# Fitting model on traintest (till 2019)
adl.m4.traintest=tslm(crime ~ season + crime.lead2+gdp.lead12+I(gdp.lead12^2)
+I(gdp.lead12^3),data=var.adl.traintest)

# Predicting values for valid(2020) using traintest
adl.m4.pred.valid.traintest=round(forecast(adl.m4.traintest, newdata =
data.frame(var.adl.valid))$mean,2)

# Finding accuracies
accuracy(round(adl.m4$fitted.values,2), var.adl.train[,'crime']) # training
accuracy
```

```
##                     ME      RMSE        MAE         MPE     MAPE      ACF1
## Test set -0.0002083333 0.0954485 0.07645833 -0.05421467 1.730841 0.3393162
##          Theil's U
## Test set 0.3214045
```

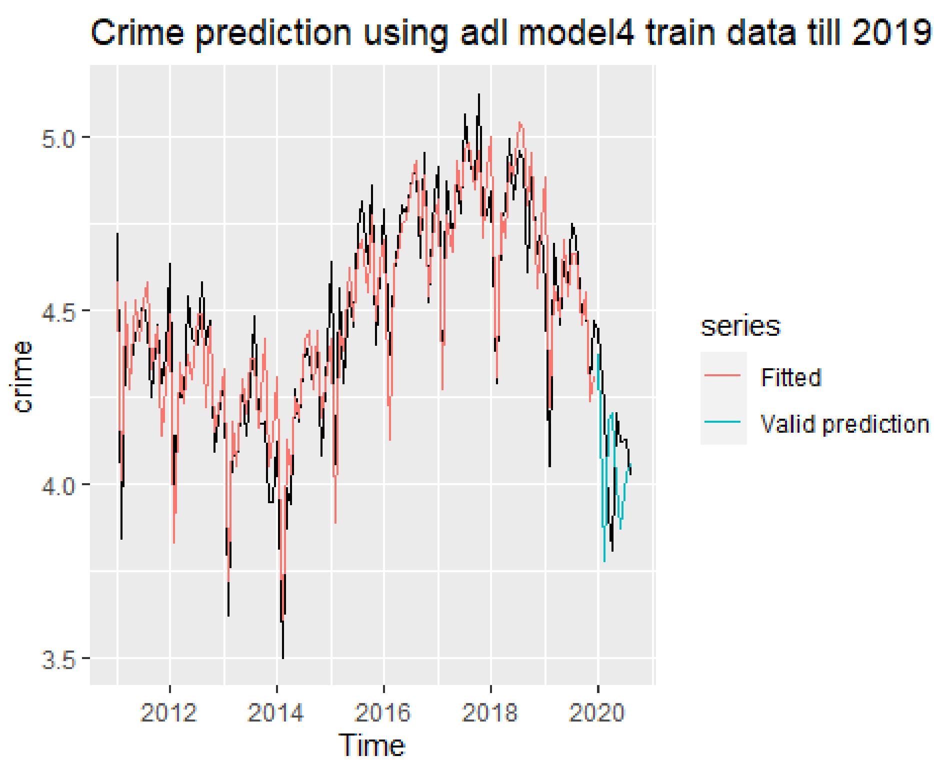
```r
accuracy(adl.m4.pred.test, var.adl.test[,'crime']) # testing accuracy
```

```
##              ME      RMSE       MAE      MPE     MAPE      ACF1 Theil's U
## Test set 0.1425 0.2224672 0.2058333 3.111808 4.544483 0.3674602 0.7120614
```

```r
accuracy(adl.m4.pred.valid.train, var.adl.valid[,'crime']) #valid accuracy
(train)

##              ME      RMSE    MAE      MPE     MAPE        ACF1 Theil's U
## Test set 0.34 0.4264387 0.3675 8.105493 8.827278 0.03641509  1.982797

accuracy(adl.m4.pred.valid.traintest, var.adl.valid[,'crime']) #valid
accuracy (traintest)

##               ME      RMSE  MAE        MPE     MAPE        ACF1 Theil's U
## Test set 0.045 0.2592296 0.22 0.8897093 5.423863 -0.01261028  1.239956

# plotting

# train,test and valid (train)
autoplot(var.adl[,'crime'],ylab = "crime",main = "Crime prediction using adl
model4 train data till 2018") +
  autolayer(round(adl.m4$fitted.values,2),series ="Fitted") +
  autolayer(adl.m4.pred.test,series ="Test prediction")+
  autolayer(adl.m4.pred.valid.train,series ="Valid prediction")
```



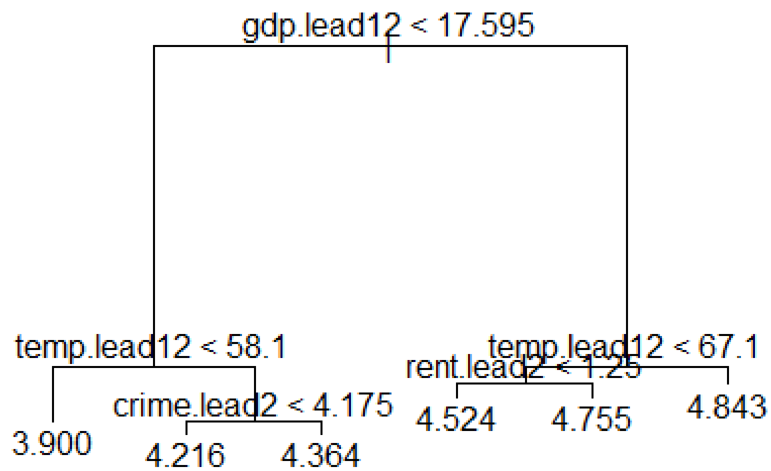

```r
# valid (traintest)
autoplot(var.adl[,'crime'],ylab = "crime",main = "Crime prediction using adl
model4 train data till 2019") +
  autolayer(round(adl.m4.traintest$fitted.values,2),series ="Fitted") +
  autolayer(adl.m4.pred.valid.traintest,series ="Valid prediction")
```



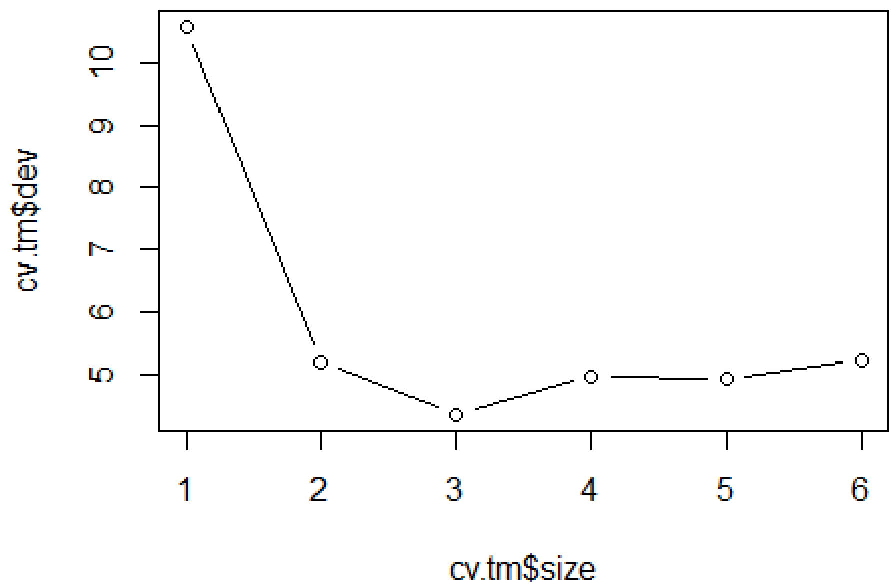
All the variables are significant now in the model and based on accuracies we stop removing more variables.

*** BUILDING DECISION TREE MODELS***

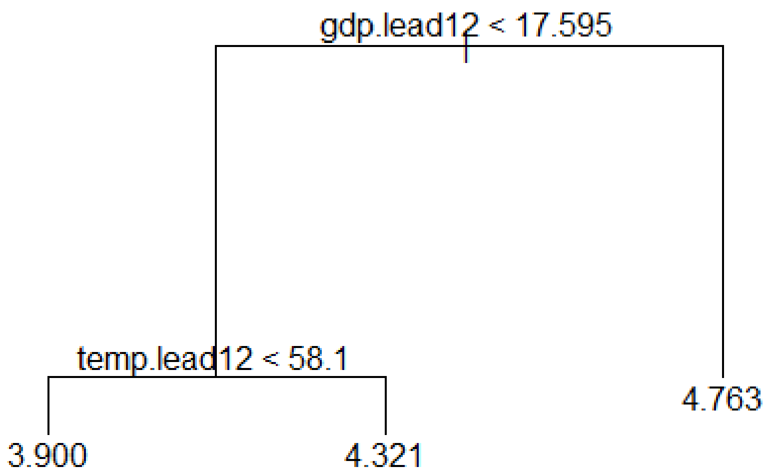
```
# Fitting decision tree model on train (till 2018)
tree.model = tree(crime ~
crime.lead2+gdp.lead12+I(gdp.lead12^2)+I(gdp.lead12^3)
+rent.lead2+temp.lead12,data=var.adl.train)
summary(tree.model)

##
## Regression tree:
## tree(formula = crime ~ crime.lead2 + gdp.lead12 + I(gdp.lead12^2) +
##     I(gdp.lead12^3) + rent.lead2 + temp.lead12, data = var.adl.train)
## Variables actually used in tree construction:
## [1] "gdp.lead12"  "temp.lead12" "crime.lead2" "rent.lead2"
## Number of terminal nodes:  6
## Residual mean deviance:  0.03253 = 2.928 / 90
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.52440 -0.09983  0.03559  0.00000  0.09590  0.42360

# Plot the tree
plot(tree.model)
text(tree.model, pretty=0)
```



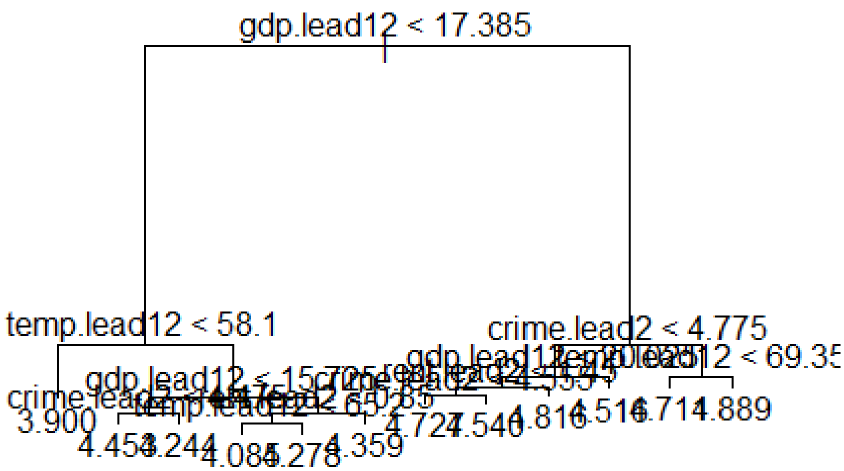

```
# Prune the tree
cv.tm = cv.tree(tree.model)
plot(cv.tm$size, cv.tm$dev, type='b') # there is not much improvement after 3 splits.
```



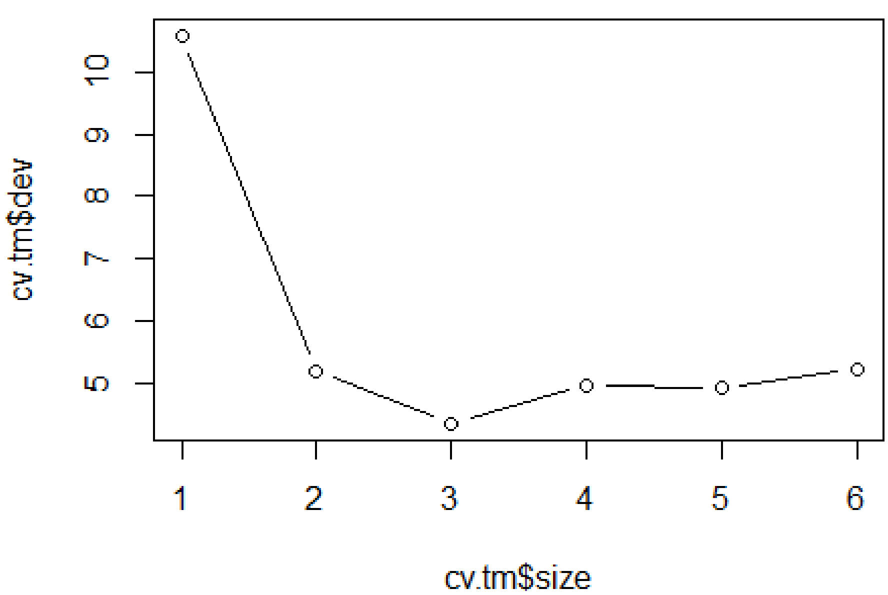

```
# can prune the tree to 3 branches
prune.tree=prune.tree(tree.model, best=3)
plot(prune.tree)
text(prune.tree, pretty=0)
```



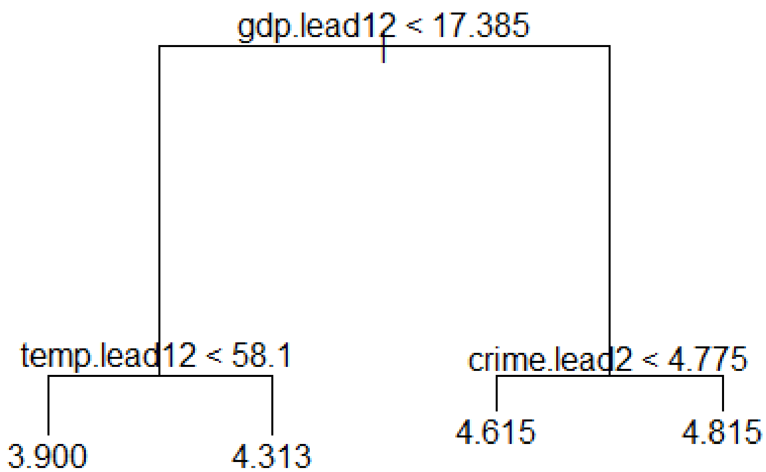

```
## Fitting decision tree model on traintest (till 2019)
tree.model.traintest=tree(crime ~
crime.lead2+gdp.lead12+I(gdp.lead12^2)+I(gdp.lead12^3)
+rent.lead2+temp.lead12,data=var.adl.traintest)
summary(tree.model.traintest)

##
## Regression tree:
## tree(formula = crime ~ crime.lead2 + gdp.lead12 + I(gdp.lead12^2) +
##     I(gdp.lead12^3) + rent.lead2 + temp.lead12, data = var.adl.traintest)
## Variables actually used in tree construction:
## [1] "gdp.lead12"  "temp.lead12" "crime.lead2" "rent.lead2"
## Number of terminal nodes:  12
## Residual mean deviance:  0.02864 = 2.749 / 96
## Distribution of residuals:
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.465800 -0.078890  0.003929  0.000000  0.097250  0.330000

# plot the tree
plot(tree.model.traintest)
text(tree.model.traintest, pretty=0)
```



```r
# prune the tree
cv.tm.traintest=cv.tree(tree.model.traintest)
plot(cv.tm$size, cv.tm$dev, type='b')
```

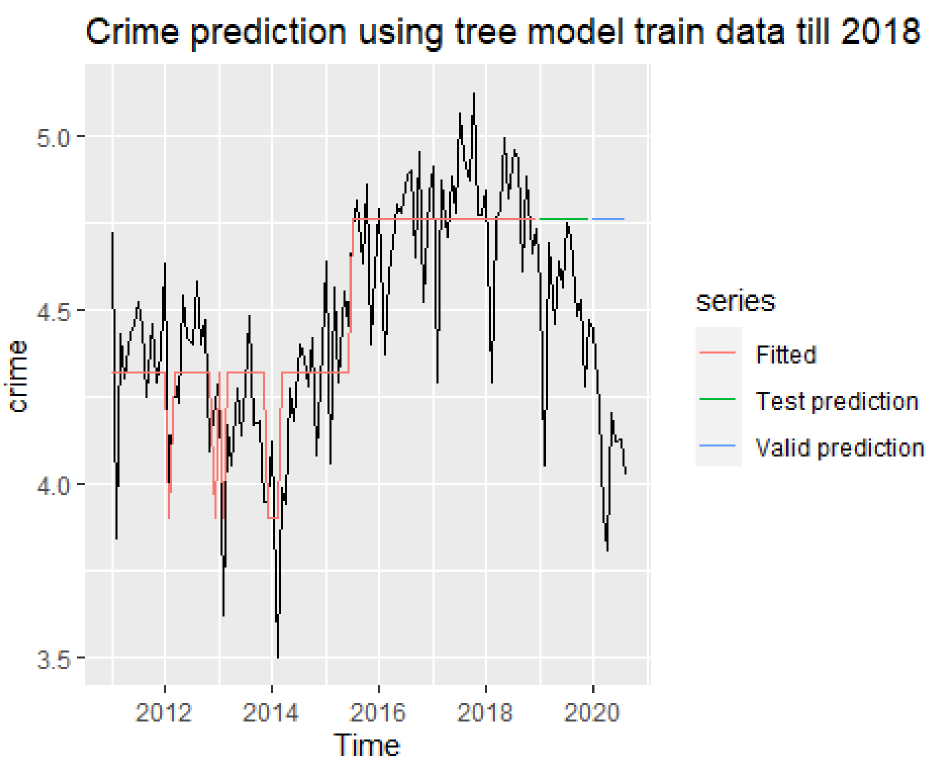


```
prune.tree.traintest=prune.tree(tree.model.traintest, best=4)
plot(prune.tree.traintest)
text(prune.tree.traintest, pretty=0)
```

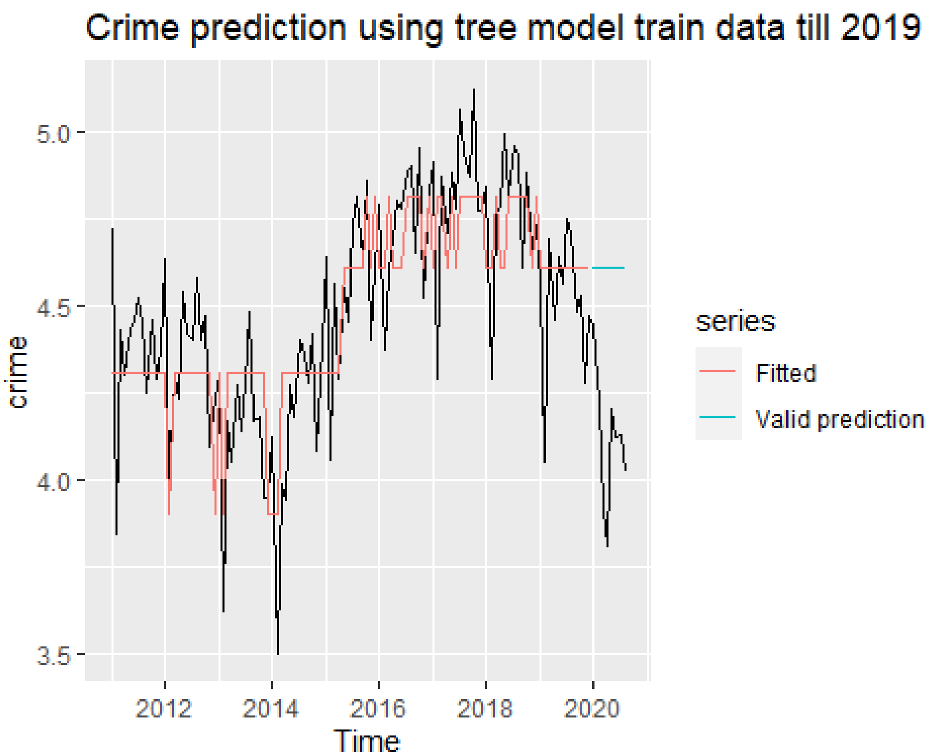



```
# Finding accuracies
accuracy(round(predict(prune.tree, newdata=var.adl.train),2),
var.adl.train[,'crime']) # training accurcy

##                 ME      RMSE       MAE        MPE     MAPE        ACF1
Theil's U
## Test set 0.001875 0.1959592 0.1554167 -0.1613801 3.546908 -0.02258433
0.6460426

accuracy(round(predict(prune.tree, newdata=var.adl.test),2),
var.adl.test[,'crime']) # testing accuracy

##                   ME      RMSE       MAE       MPE     MAPE       ACF1
Theil's U
## Test set -0.2441667 0.3079367 0.2441667 -5.598466 5.598466 -0.1460992
1.015683

accuracy(round(predict(prune.tree, newdata=var.adl.valid),2),
var.adl.valid[,'crime']) # valid accuracy (train)

##                 ME      RMSE     MAE       MPE     MAPE      ACF1 Theil's U
## Test set -0.65375 0.6788501 0.65375 -16.15081 16.15081 0.1703123   3.27143
```

```
accuracy(round(predict(prune.tree.traintest, newdata=var.adl.valid),2),
var.adl.valid[,'crime']) # valid accuracy (traintest)

##                 ME      RMSE     MAE       MPE     MAPE      ACF1 Theil's U
## Test set -0.50375 0.5359221 0.50375 -12.49059 12.49059 0.1703123  2.606214

# plotting

# train,test and valid (train)
autoplot(var.adl[,'crime'],ylab = "crime",main = "Crime prediction using tree
model train data till 2018") +
  autolayer(ts(round(predict(prune.tree, newdata=var.adl.train),2),end =
c(2018,12),freq = 12),series ="Fitted") +
  autolayer(ts(round(predict(prune.tree, newdata=var.adl.test),2),end =
c(2019,12),freq = 12),series ="Test prediction")+
  autolayer(ts(round(predict(prune.tree, newdata=var.adl.valid),2),end =
c(2020,8),freq = 12),series ="Valid prediction")
```




```
# valid (traintest)
autoplot(var.adl[,'crime'],ylab = "crime",main = "Crime prediction using tree
model train data till 2019") +
  autolayer(ts(round(predict(prune.tree.traintest,
newdata=var.adl.traintest),2),end = c(2019,12),freq = 12),series ="Fitted")+
  autolayer(ts(round(predict(prune.tree.traintest,
newdata=var.adl.valid),2),end = c(2020,8),freq = 12),series ="Valid
prediction")
```



Based on the all the above models that have been tried, adl.m4 has the best performance and thus this is our chosen model.

*** CHECKING FOR STRUCTURAL BREAK ***

There is a structural break expected starting around late Dec 2019 when pandemic was announced and lot of economic activities slowed down and sudden increased unemployment rate. To analyze this we are using training data till 2019 and applying our model on the 2020 data till August to see if the predicted vs. actual results are significantly different using a paired t-test.

Hypothesis test

H0 - Ud = 0 i.e. Crime predicted and crime actual are not different and crime rate was not impacted due to covid Ha - Ud != 0 Crime predicted and crime actual are different and crime rate was indeed impacted due to covid

alpha here is chosen at 5%

```
# Paired t-test

t.test(adl.m4.pred.valid.traintest, var.adl.valid[,'crime'],paired = TRUE)
```

```
##
##  Paired t-test
##
## data:  adl.m4.pred.valid.traintest and var.adl.valid[, "crime"]
```

```
## t = -0.46636, df = 7, p-value = 0.6551
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.2731674  0.1831674
## sample estimates:
## mean of the differences
##                  -0.045
```

Given that p-value is 0.665 which is much higher than 0.05, we cannot reject the null hypothesis that crime rate has changed due to covid-19.

— PART C

#######R SHINY APP#######

```
# Reading Library
library(shiny)
library(shinydashboard)

## Warning: package 'shinydashboard' was built under R version 4.0.3

##
## Attaching package: 'shinydashboard'

## The following object is masked from 'package:graphics':
##
##     box

library(mapproj)

## Warning: package 'mapproj' was built under R version 4.0.3

## Loading required package: maps

## Warning: package 'maps' was built under R version 4.0.3

##
## Attaching package: 'maps'

## The following object is masked from 'package:astsa':
##
##     unemp

library(maps)

# Creating User Interface

  ui <-

           dashboardPage(
                        dashboardHeader(title = "LA Crime Rate"),
```

```
                        dashboardSidebar(
                                        #sliderInput("bins","Number of
breaks",1,100,50),
                                        sidebarMenu(
                                                    menuItem("Data",tabName =
"timeseries",icon = icon("chart-line")),
                                                    menuSubItem("Crime
Data",tabName = "crime"),
                                                    menuSubItem("External
Variables",tabName = "external"),
                                                    menuItem("Prediction
Models",tabName = "model",icon = icon("sistrix")),

menuSubItem("Linear",tabName = "linear"),

menuSubItem("Quadratic",tabName = "quadratic"),

menuSubItem("Cubic",tabName = "cubic"),
                                                    menuSubItem("Moving
Average",tabName = "ma"),
                                                    menuSubItem("Exponential
Smoothing",tabName = "exponential"),

menuSubItem("SARIMA",tabName = "sarima"),
                                                    menuSubItem("ADL",tabName
= "adl"),
                                                    menuSubItem("Decision
Tree",tabName = "tree")

                                                    ),
                                        sliderInput("obs1", "Year Range:",min
= 2010, max = 2020, value = c(2010,2020),sep = "")




                        ),




            dashboardBody(
                        tabItems(
                                tabItem(tabName = "timeseries"),

                                tabItem(tabName = "crime",
                                        fluidRow(splitLayout(cellWidths =
c("70%", "30%"), plotOutput("crime.ts"), plotOutput("map")))
```

```
                                        ),

                                tabItem(tabName = "external",
                                        fluidRow(splitLayout(cellWidths = 
c("50%", "50%"), plotOutput("gdp.ts"), plotOutput("ur.ts"))),
                                        fluidRow(splitLayout(cellWidths = 
c("50%", "50%"), plotOutput("temp.ts"), plotOutput("rent.ts"))),

                                        ),



                                tabItem(tabName = "model"),

                                tabItem(tabName = "linear",
                                        fluidRow(splitLayout(cellWidths = 
c("70%", "60%"), plotOutput("linear"), infoBox("2020 PREDICTION 
ERROR(%)",value = tags$p(15.16, style = "font-size: 190%;"),icon = 
icon("bullseye"),color = "green")))
                                        ),
                                tabItem(tabName = "quadratic",
                                        fluidRow(splitLayout(cellWidths = 
c("70%", "60%"), plotOutput("quadratic"), infoBox("2020 PREDICTION 
ERROR(%)",value = tags$p(19.71, style = "font-size: 190%;"),icon = 
icon("bullseye"),color = "green")))
                                        ),
                                tabItem(tabName = "cubic",
                                        fluidRow(splitLayout(cellWidths = 
c("70%", "60%"), plotOutput("cubic"), infoBox("2020 PREDICTION 
ERROR(%)",value = tags$p(4.07, style = "font-size: 190%;"),icon = 
icon("bullseye"),color = "green")))
                                        ),
                                tabItem(tabName = "ma",
                                        fluidRow(splitLayout(cellWidths = 
c("70%", "60%"), plotOutput("ma"), infoBox("2020 PREDICTION ERROR(%)",value = 
tags$p(7.84, style = "font-size: 190%;"),icon = icon("bullseye"),color = 
"green")))
                                        ),
                                tabItem(tabName = "exponential",
                                        fluidRow(splitLayout(cellWidths = 
c("70%", "60%"), plotOutput("exponential"), infoBox("2020 PREDICTION 
ERROR(%)",value = tags$p(8.74, style = "font-size: 190%;"),icon = 
icon("bullseye"),color = "green")))
                                        ),
                                tabItem(tabName = "sarima",
                                        fluidRow(splitLayout(cellWidths = 
c("70%", "60%"), plotOutput("sarima"), infoBox("2020 PREDICTION 
ERROR(%)",value = tags$p(8.24, style = "font-size: 190%;"),icon = 
icon("bullseye"),color = "green")))
```

```
                                        ),
                                tabItem(tabName = "adl",
                                        fluidRow(splitLayout(cellWidths =
c("70%", "60%"), plotOutput("adl"), infoBox("2020 PREDICTION ERROR(%)",value
= tags$p(5.42, style = "font-size: 190%;"),icon = icon("bullseye"),color =
"green")))
                                        ),
                                tabItem(tabName = "tree",
                                        fluidRow(splitLayout(cellWidths =
c("70%", "60%"), plotOutput("tree"), infoBox("2020 PREDICTION ERROR(%)",value
= tags$p(12.49, style = "font-size: 190%;"),icon = icon("bullseye"),color =
"green")))
                                        )
                                )


                        )









)




# Creating Server

server <- function(input, output)
{


   output$linear <- renderPlot({ ts1 <- window(crime.ts,start =
input$obs1[1],end=input$obs1[2])
                                 plot(ts1, xlab = "Time", ylab = "Monthly
Crime", main = "Monthly LA Crimes/Population (in 1000s)-LINEAR MODEL")+
                                 lines(lr_trend$fitted.values, col="red")+
                                 lines(forecast(lr_trend, h=12)$mean,col =
"green",lty = 2)
                             })

   output$quadratic <- renderPlot({ ts1 <- window(crime.ts,start =
input$obs1[1],end=input$obs1[2])
                                   plot(ts1, xlab = "Time", ylab = "Monthly
Crime", main = "Monthly LA Crimes/Population (in 1000s)-QUADRATIC MODEL")+
                                   lines(quad_trend$fitted.values,
col="red")+
```

```
                                    lines(forecast(quad_trend, h=12)$mean,col
= "green",lty = 2)
                               })

   output$cubic <- renderPlot({ts1 <- window(crime.ts,start =
input$obs1[1],end=input$obs1[2])
                               plot(ts1, xlab = "Time", ylab = "Monthly
Crime", main = "Monthly LA Crimes/Population (in 1000s)-CUBIC MODEL")+
                               lines(cubic_trend$fitted.values, col="red")+
                               lines(forecast(cubic_trend, h=12)$mean,col =
"green",lty = 2)
                               })

   output$ma <- renderPlot({ts1 <- window(crime.ts,start =
input$obs1[1],end=input$obs1[2])
                           plot(ts1,ylab = "Monthly Crime", main = "Monthly
LA Crimes/Population (in 1000s)-MOVING AVERAGE MODEL")+
                           lines(ma.trailing.roll.1,series = "Fitted",col =
"red")+
                           lines(ma.trailing.pred.r.1,series =
"Prediction",col = "green",lty = 2)
                           })

   output$exponential <- renderPlot({ts1 <- window(crime.ts,start =
input$obs1[1],end=input$obs1[2])
                                     plot(ts1,ylab = "Monthly Crime", main =
"Monthly LA Crimes/Population (in 1000s)-EXPONENTIAL SMOOTHING MODEL")+
                                     lines(modelopt.1$fitted,series =
"Fitted",col = "red")+
                                     lines(forecast(modelopt.1, h =
length(valid.ts))$mean,series = "Predicted",col = "green",lty = 2)+
                                     lines(valid.ts,series = "Observed",col =
"blue")
                                     })

   output$sarima <- renderPlot({ts1 <- window(crime.ts,start =
input$obs1[1],end=input$obs1[2])
                               plot(ts1,ylab = "Monthly Crime", main =
"Monthly LA Crimes/Population (in 1000s)-SARIMA MODEL")+
                               lines(modelsarima.1.1$fitted,series =
"Fitted",col = "red")+
                               lines(forecast(modelsarima.1.1, h =
length(valid.ts))$mean,series = "Prediction",col = "green",lty = 2)+
                               lines(valid.ts,series = "Observed",col =
"blue")
                               })

   output$adl <- renderPlot({ts1 <- window(var.adl[,'crime'],start =
input$obs1[1],end=input$obs1[2])
```

```
                              plot(ts1,ylab = "Monthly Crime", main = "Monthly
LA Crimes/Population (in 1000s)-ADL MODEL")+

lines(round(adl.m4.traintest$fitted.values,2),series ="Fitted",col = "red") +
                              lines(adl.m4.pred.valid.traintest,series
="Predicted",col = "green",lty = 2)
                            })

   output$tree <- renderPlot({ts1 <- window(var.adl[,'crime'],start =
input$obs1[1],end=input$obs1[2])
                               plot(ts1,ylab = "Monthly Crime", main =
"Monthly LA Crimes/Population (in 1000s)- TREE MODEL") +
                               lines(ts(round(predict(prune.tree.traintest,
newdata=var.adl.traintest),2),end = c(2019,12),freq = 12),series
="Fitted",col = "red")+
                               lines(ts(round(predict(prune.tree.traintest,
newdata=var.adl.valid),2),end = c(2020,8),freq = 12),series ="Predicted",col
= "green",lty = 2)
                             })



   output$map <- renderPlot({
                                ggplot(map_data("state", region="california"),
aes(x=long, y=lat)) +
                                geom_polygon() +
                                coord_map() +
                                geom_point(aes(x=-118.41, y=34.11),
color="green",size = 10)
                            })

   output$crime.ts <- renderPlot({ ts1 <- window(crime.ts,start =
input$obs1[1],end=input$obs1[2])
                                     plot(ts1, xlab = "Time", ylab = "Monthly
Crime", main = "LA Crimes")
                                })

   output$gdp.ts <- renderPlot({ ts1 <- window(gdp.ts,start =
input$obs1[1],end=input$obs1[2])
                                   plot(ts1, xlab = "Time", ylab = "Monthly
GDP", main = "LA GDP", cex.lab = 2, cex.axis=2, cex.main=1.5)
                                })

   output$temp.ts <- renderPlot({ ts1 <- window(temp.ts,start =
input$obs1[1],end=input$obs1[2])
                                    plot(ts1, xlab = "Time", ylab = "Monthly
Average Temperature", main = "LA Monthly Average Temperature ", cex.lab = 2,
cex.axis=2, cex.main=1.5)
                               })
```

```
   output$ur.ts <- renderPlot({ ts1 <- window(ur.ts,start =
input$obs1[1],end=input$obs1[2])
                                plot(ts1, xlab = "Time", ylab = "Monthly
Unemployment Rate", main = "LA Unemployment Rate ", cex.lab = 2, cex.axis=2,
cex.main=1.5)
                             })

   output$rent.ts <- renderPlot({ ts1 <- window(rent.ts,start =
input$obs1[1],end=input$obs1[2])
                                  plot(ts1, xlab = "Time", ylab = "Monthly
Rent CPI", main = "LA Rent CPI", cex.lab = 2, cex.axis=2, cex.main=1.5)
                               })




}




# Launching the Shiny Application
shinyApp(ui,server)
```

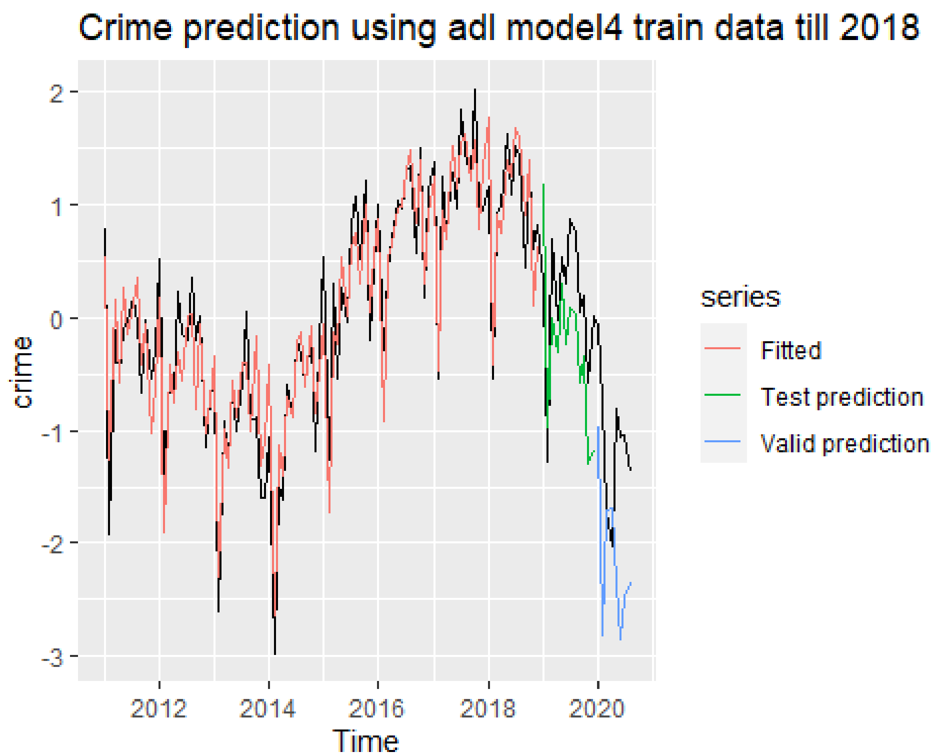
Shiny applications not supported in static R Markdown documents

try if standarization helps in improving the performance

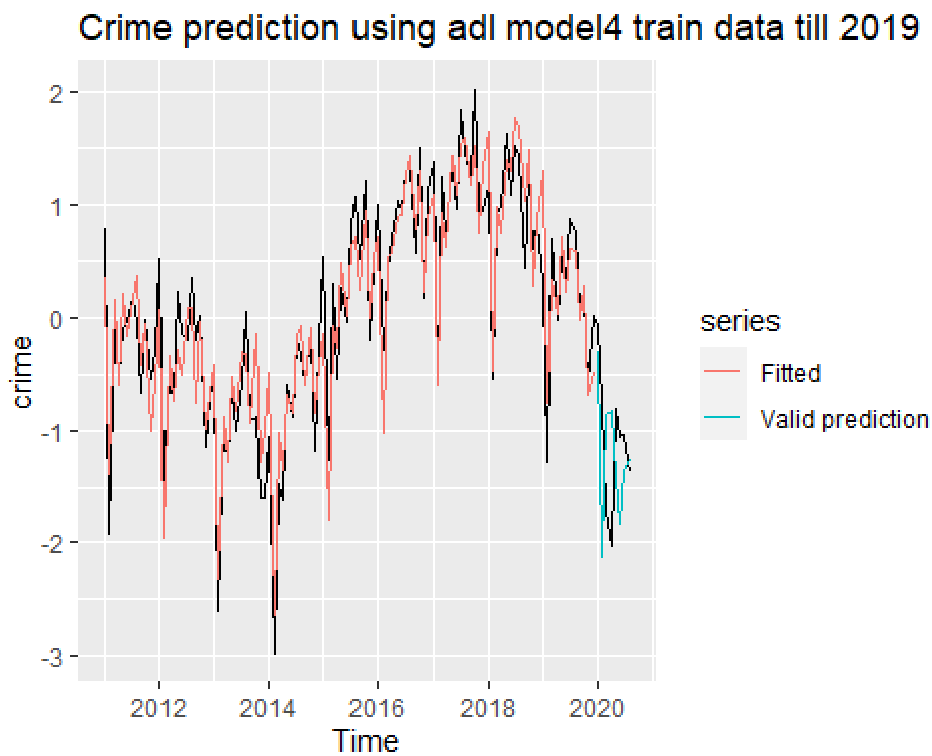
```
# standardizing the data
for(i in seq_len(ncol(var.adl))) var.adl[,i] <- scale(var.adl[,i])

# creating train, test, valid and traintest again
var.adl.train=window(var.adl, end=c(2018,12))
var.adl.test=window(var.adl, start=c(2019, 1), end=c(2019,12))
var.adl.valid=window(var.adl, start=c(2020,1))
var.adl.traintest=window(var.adl, end=c(2019,12))

adl.m4 = tslm(crime ~ season +
crime.lead2+gdp.lead12+I(gdp.lead12^2)+I(gdp.lead12^3),data=var.adl.train)
summary(adl.m4)

##
## Call:
## tslm(formula = crime ~ season + crime.lead2 + gdp.lead12 + I(gdp.lead12^2)
+
##     I(gdp.lead12^3), data = var.adl.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.70759 -0.18431  0.01204  0.20486  0.62695
```

```
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       0.93457    0.12439   7.513 7.21e-11 ***
## season2          -2.17719    0.16852 -12.920  < 2e-16 ***
## season3          -0.95185    0.19128  -4.976 3.64e-06 ***
## season4          -0.30955    0.18307  -1.691  0.09475 .
## season5          -0.33444    0.16966  -1.971  0.05215 .
## season6          -0.49194    0.16362  -3.006  0.00353 **
## season7          -0.34178    0.18472  -1.850  0.06797 .
## season8          -0.13865    0.17307  -0.801  0.42543
## season9          -0.96804    0.19456  -4.975 3.66e-06 ***
## season10         -0.43018    0.20017  -2.149  0.03465 *
## season11         -1.13333    0.17283  -6.558 4.96e-09 ***
## season12         -0.96158    0.19948  -4.820 6.71e-06 ***
## crime.lead2       0.47025    0.09585   4.906 4.81e-06 ***
## gdp.lead12        1.01167    0.20621   4.906 4.80e-06 ***
## I(gdp.lead12^2) -0.25978    0.09862  -2.634  0.01012 *
## I(gdp.lead12^3) -0.54859    0.12901  -4.252 5.69e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3227 on 80 degrees of freedom
## Multiple R-squared:  0.9149, Adjusted R-squared:  0.8989
## F-statistic: 57.31 on 15 and 80 DF,  p-value: < 2.2e-16

# Predicting values for test(2019) using train
adl.m4.pred.test=round(forecast(adl.m4, newdata =
data.frame(var.adl.test))$mean,2)

# Predicting values for valid(2020) using train
adl.m4.pred.valid.train=ts(as.numeric(round(forecast(adl.m4, newdata =
data.frame(var.adl.valid))$mean,2)),start = c(2020,1),end =
c(2020,8),frequency = 12)

# Fitting model on traintest (till 2019)
adl.m4.traintest=tslm(crime ~ season + crime.lead2+gdp.lead12+I(gdp.lead12^2)
+I(gdp.lead12^3),data=var.adl.traintest)

# Predicting values for valid(2020) using traintest
adl.m4.pred.valid.traintest=round(forecast(adl.m4.traintest, newdata =
data.frame(var.adl.valid))$mean,2)

# Finding accuracies
accuracy(round(adl.m4$fitted.values,2), var.adl.train[,'crime']) # training
accuracy

##                     ME      RMSE       MAE      MPE     MAPE      ACF1
## Test set -0.0003682979 0.2942329 0.2356335 27.68191 75.85841 0.3379518
```

```
##          Theil's U
## Test set 0.4097492

accuracy(adl.m4.pred.test, var.adl.test[,'crime']) # testing accuracy

##                 ME      RMSE       MAE      MPE     MAPE      ACF1 Theil's
U
## Test set 0.4354714 0.6809929 0.6295958 576.2323 984.2954 0.3651764
0.5328314

accuracy(adl.m4.pred.valid.train, var.adl.valid[,'crime']) #valid accuracy
(train)

##                ME     RMSE      MAE       MPE     MAPE       ACF1 Theil's
U
## Test set 1.044963 1.311994 1.130656 -249.7691 254.0144 0.03417074
3.016125

accuracy(adl.m4.pred.valid.traintest, var.adl.valid[,'crime']) #valid
accuracy (traintest)

##                 ME      RMSE       MAE       MPE     MAPE        ACF1
Theil's U
## Test set 0.1387125 0.7984144 0.6775343 -66.68269 95.94159 -0.01184276
1.991303

# plotting

# train,test and valid (train)
autoplot(var.adl[,'crime'],ylab = "crime",main = "Crime prediction using adl
model4 train data till 2018") +
  autolayer(round(adl.m4$fitted.values,2),series ="Fitted") +
  autolayer(adl.m4.pred.test,series ="Test prediction")+
  autolayer(adl.m4.pred.valid.train,series ="Valid prediction")
```




```
# valid (traintest)
autoplot(var.adl[,'crime'],ylab = "crime",main = "Crime prediction using adl
model4 train data till 2019") +
  autolayer(round(adl.m4.traintest$fitted.values,2),series ="Fitted") +
  autolayer(adl.m4.pred.valid.traintest,series ="Valid prediction")
```

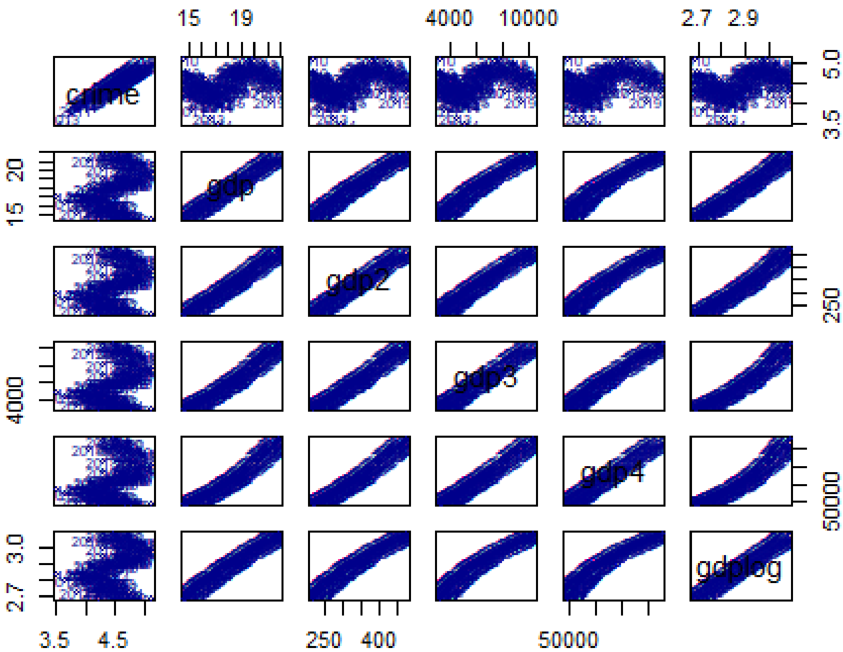


The standardization does not help in improving the model performance.

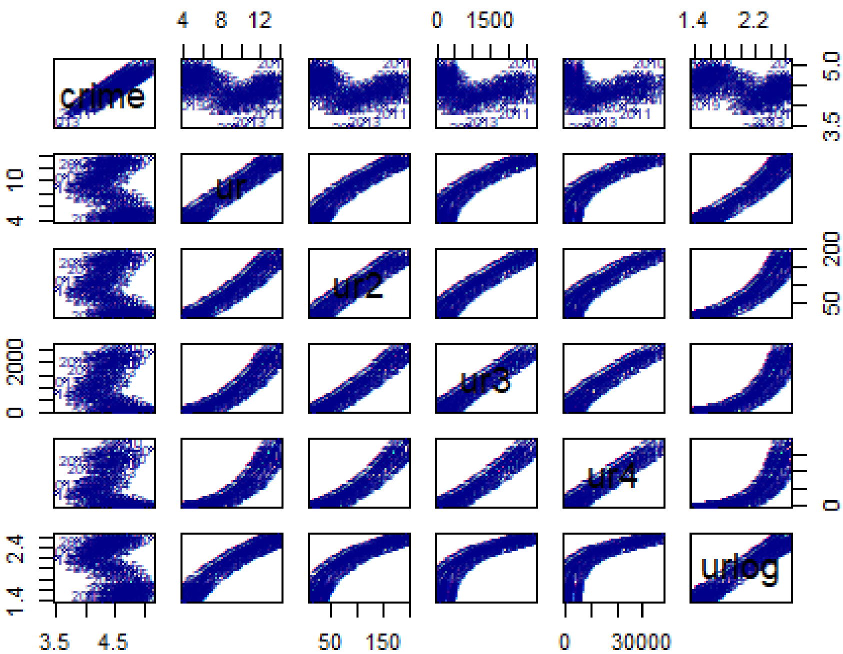
Polynomial transformation of external variables

```
gdp.ts.2=gdp.ts^2
gdp.ts.3=gdp.ts^3
gdp.ts.4=gdp.ts^4
gdp.ts.log=log(gdp.ts)

rent.ts.2=rent.ts^2
rent.ts.3=rent.ts^3
rent.ts.4=rent.ts^4
rent.ts.log=log(rent.ts)
ur.ts.2=ur.ts^2
ur.ts.3=ur.ts^3
ur.ts.4=ur.ts^4
ur.ts.log=log(ur.ts)
temp.ts.2=temp.ts^2
temp.ts.3=temp.ts^3
temp.ts.4=temp.ts^4
temp.ts.log=log(temp.ts)

var.poly=ts.intersect(crime=crime.ts,
                      gdp=gdp.ts,
                      gdp2=gdp.ts.2,
                      gdp3=gdp.ts.3,
```

```
                      gdp4=gdp.ts.4,
                      gdplog=gdp.ts.log,
                      rent=rent.ts,
                      rent2=rent.ts.2,
                      rent3=rent.ts.3,
                      rent4=rent.ts.4,
                      rentlog=rent.ts.log,
                      ur=ur.ts,
                      ur2=ur.ts.2,
                      ur3=ur.ts.3,
                      ur4=ur.ts.4,
                      urlog=ur.ts.log,
                      temp=temp.ts,
                      temp2=temp.ts.2,
                      temp3=temp.ts.3,
                      temp4=temp.ts.4,
                      templog=temp.ts.log)

# get data before 2020
var.poly.2019=window(var.poly, end=c(2019,12))

# examine linear relationship after transformation
var.poly.corr <- cor(as.matrix(var.poly))
var.corr.line <- var.poly.corr["crime",]
var.corr.line

##      crime        gdp       gdp2       gdp3       gdp4     gdplog
rent
##  1.0000000  0.2568808  0.2582808  0.2572168  0.2538585  0.2529394
0.2008331
##      rent2      rent3      rent4    rentlog         ur        ur2
ur3
##  0.1833415  0.1639982  0.1431685  0.2161645 -0.4572438 -0.4099269 -
0.3578787
##        ur4      urlog       temp      temp2      temp3      temp4
templog
## -0.3146669 -0.4879355  0.2894994  0.2918178  0.2938221  0.2955027
0.2868833

# just look at the linear relationship between GDP related variables and
crime
var.poly.gdp.2019=window(ts.intersect(crime=crime.ts,
                      gdp=gdp.ts,
                      gdp2=gdp.ts.2,
                      gdp3=gdp.ts.3,
                      gdp4=gdp.ts.4,
                      gdplog=gdp.ts.log),
                    end=c(2019,12))
pairs(var.poly.gdp.2019, cex = 0.65, col = "darkblue")
```

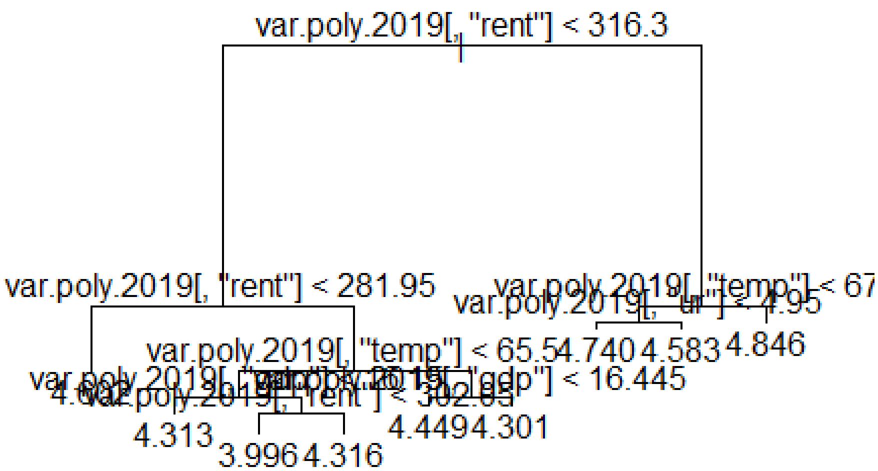



```
# just look at the linear relationship between UR related variables and crime
var.poly.ur.2019=window(ts.intersect(crime=crime.ts,
                                      ur=ur.ts,
                                     ur2=ur.ts.2,
                                     ur3=ur.ts.3,
                                     ur4=ur.ts.4,
                                     urlog=ur.ts.log),
                         end=c(2019,12))
pairs(var.poly.ur.2019, cex = 0.65, col = "darkblue")
```

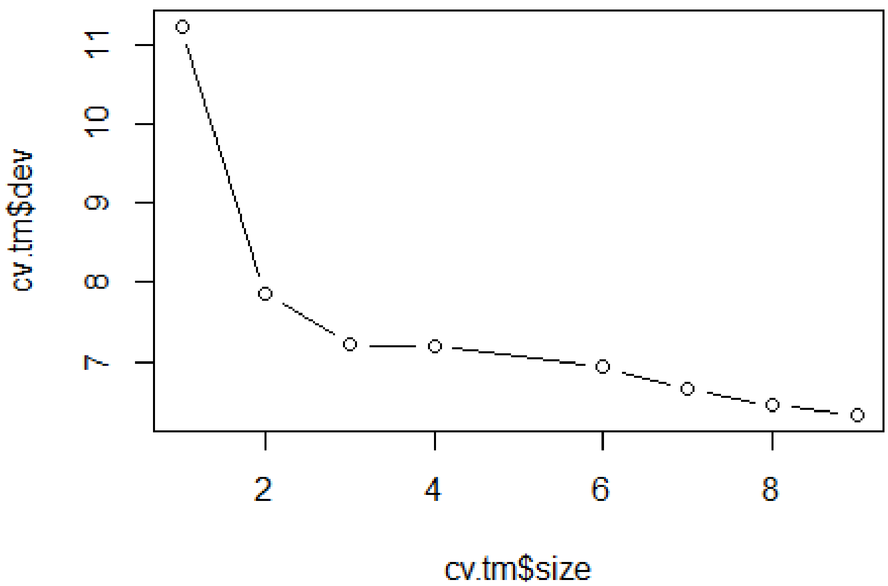



```r
# get training data
var.poly.2019=window(var.poly, end=c(2018,12))

# try a new tree model with all original and transformed variables
tree.poly=tree(var.poly.2019[,'crime']~var.poly.2019[,'gdp'] +
var.poly.2019[,'gdp2'] + var.poly.2019[,'gdp3'] + var.poly.2019[,'gdp4'] +
var.poly.2019[,'gdplog'] +
                 var.poly.2019[,'rent'] + var.poly.2019[,'rent2'] +
var.poly.2019[,'rent3'] + var.poly.2019[,'rent4'] + var.poly.2019[,'rentlog']
+
                 var.poly.2019[,'ur'] + var.poly.2019[,'ur2'] +
var.poly.2019[,'ur3'] + var.poly.2019[,'ur4'] + var.poly.2019[,'urlog'] +
                 var.poly.2019[,'temp'] + var.poly.2019[,'temp2'] +
var.poly.2019[,'temp3'] + var.poly.2019[,'temp4'] + var.poly.2019[,'templog']
                )
summary(tree.poly)

##
## Regression tree:
## tree(formula = var.poly.2019[, "crime"] ~ var.poly.2019[, "gdp"] +
##     var.poly.2019[, "gdp2"] + var.poly.2019[, "gdp3"] + var.poly.2019[,
##     "gdp4"] + var.poly.2019[, "gdplog"] + var.poly.2019[, "rent"] +
##     var.poly.2019[, "rent2"] + var.poly.2019[, "rent3"] + var.poly.2019[,
##     "rent4"] + var.poly.2019[, "rentlog"] + var.poly.2019[, "ur"] +
##     var.poly.2019[, "ur2"] + var.poly.2019[, "ur3"] + var.poly.2019[,
##     "ur4"] + var.poly.2019[, "urlog"] + var.poly.2019[, "temp"] +
##     var.poly.2019[, "temp2"] + var.poly.2019[, "temp3"] + var.poly.2019[,
```

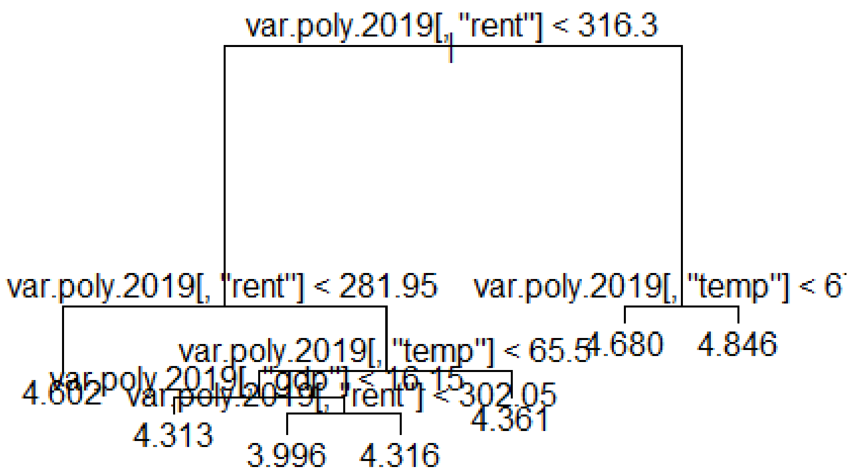
```
##     "temp4"] + var.poly.2019[, "templog"])
## Variables actually used in tree construction:
## [1] "var.poly.2019[, \"rent\"]" "var.poly.2019[, \"temp\"]"
## [3] "var.poly.2019[, \"gdp\"]"  "var.poly.2019[, \"ur\"]"
## Number of terminal nodes:  9
## Residual mean deviance:  0.0345 = 3.416 / 99
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.49580 -0.06929  0.02045  0.00000  0.09953  0.51770

plot(tree.poly)
text(tree.poly, pretty=0)
```




```
cv.tm=cv.tree(tree.poly)
plot(cv.tm$size, cv.tm$dev, type='b')
```

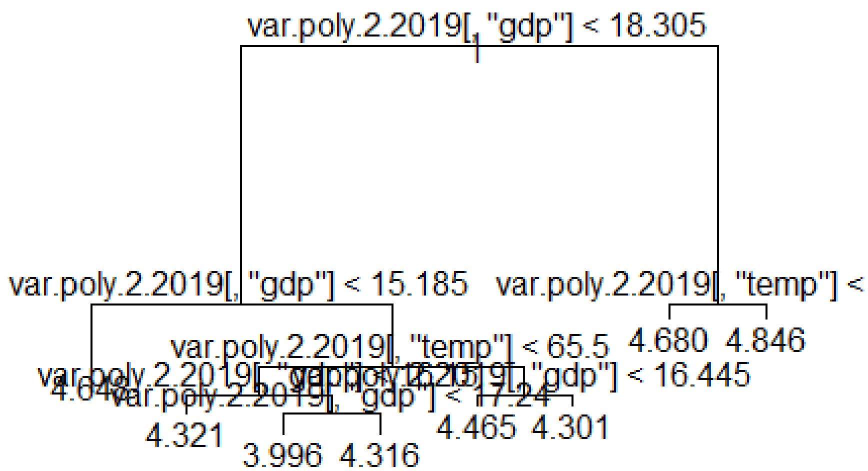



```
# there is not much improvement after 7 splits.

# can prune the tree to 7 branches
prune.tree=prune.tree(tree.poly, best=7)
plot(prune.tree)
text(prune.tree, pretty=0)
```




```
# Seems that the tree still only picked up the linear variables...
# exactly the same tree as the last one


# What if we add polynomial variables with linear variables?
# Will the linear variables still outperform?
# This time we only look at GDP and temperature first
var.poly.2=ts.intersect(crime=crime.ts,
                        gdp=gdp.ts,
                        gdp2=gdp.ts.2,
                        gdp3=gdp.ts.3,
                        gdp4=gdp.ts.4,
                        gdplog=gdp.ts.log,
                        gdp2l=gdp.ts+gdp.ts.2,
                        gdp3l=gdp.ts+gdp.ts.3,
                        gdp4l=gdp.ts+gdp.ts.4,
                        temp=temp.ts,
                        temp2=temp.ts.2,
                        temp3=temp.ts.3,
                        temp4=temp.ts.4,
                        templog=temp.ts.log,
                        temp2l=temp.ts+temp.ts.2,
                        temp3l=temp.ts+temp.ts.3,
                        temp4l=temp.ts+temp.ts.4)
var.poly.2.2019=window(var.poly.2, end=c(2018,12))
```

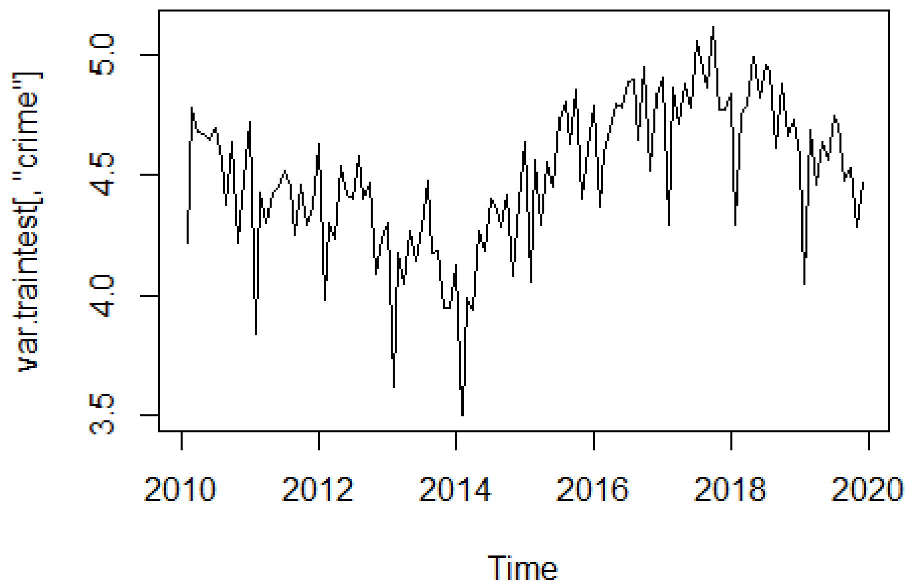
```r
tree.poly.2=tree(var.poly.2.2019[,'crime']~var.poly.2.2019[,'gdp'] +
var.poly.2.2019[,'gdp2'] + var.poly.2.2019[,'gdp3'] +
var.poly.2.2019[,'gdp4'] + var.poly.2.2019[,'gdplog'] +
                 var.poly.2.2019[,'gdp2l'] + var.poly.2.2019[,'gdp3l'] +
var.poly.2.2019[,'gdp4l'] +
                 var.poly.2.2019[,'temp'] + var.poly.2.2019[,'temp2'] +
var.poly.2.2019[,'temp3'] + var.poly.2.2019[,'temp4'] +
var.poly.2.2019[,'templog'] +
                var.poly.2.2019[,'temp2l'] + var.poly.2.2019[,'temp3l'] +
var.poly.2.2019[,'temp4l']
)
summary(tree.poly.2)

##
## Regression tree:
## tree(formula = var.poly.2.2019[, "crime"] ~ var.poly.2.2019[,
##     "gdp"] + var.poly.2.2019[, "gdp2"] + var.poly.2.2019[, "gdp3"] +
##     var.poly.2.2019[, "gdp4"] + var.poly.2.2019[, "gdplog"] +
##     var.poly.2.2019[, "gdp2l"] + var.poly.2.2019[, "gdp3l"] +
##     var.poly.2.2019[, "gdp4l"] + var.poly.2.2019[, "temp"] +
##     var.poly.2.2019[, "temp2"] + var.poly.2.2019[, "temp3"] +
##     var.poly.2.2019[, "temp4"] + var.poly.2.2019[, "templog"] +
##     var.poly.2.2019[, "temp2l"] + var.poly.2.2019[, "temp3l"] +
##     var.poly.2.2019[, "temp4l"])
## Variables actually used in tree construction:
## [1] "var.poly.2.2019[, \"gdp\"]"  "var.poly.2.2019[, \"temp\"]"
## Number of terminal nodes:  8
## Residual mean deviance:  0.0343 = 3.43 / 100
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.49580 -0.07250  0.01214  0.00000  0.10930  0.47200

plot(tree.poly.2)
text(tree.poly.2, pretty=0)
```



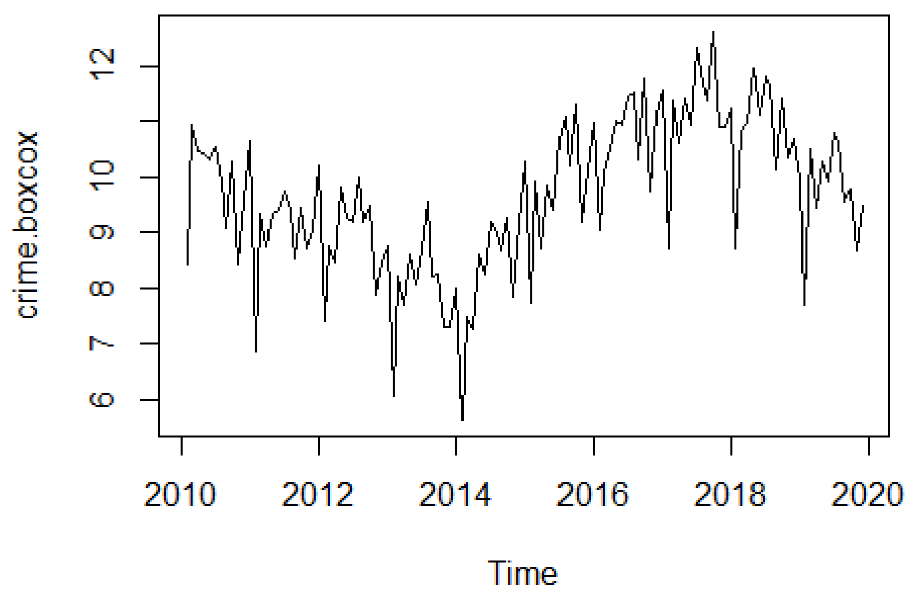

```r
# Still only used GDP and temperature...
```

Trying boxcox transformation for the variables

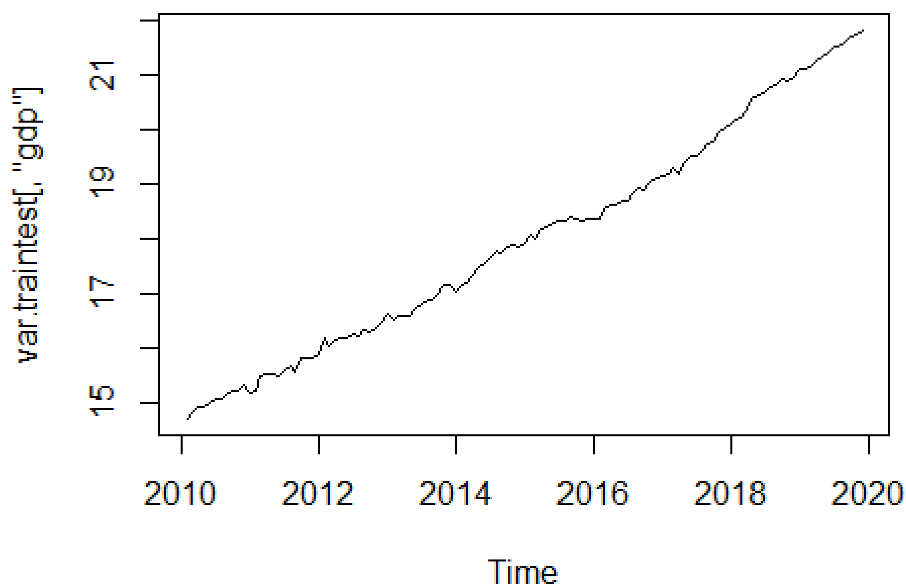
```r
plot(var.traintest[,'crime'])
```



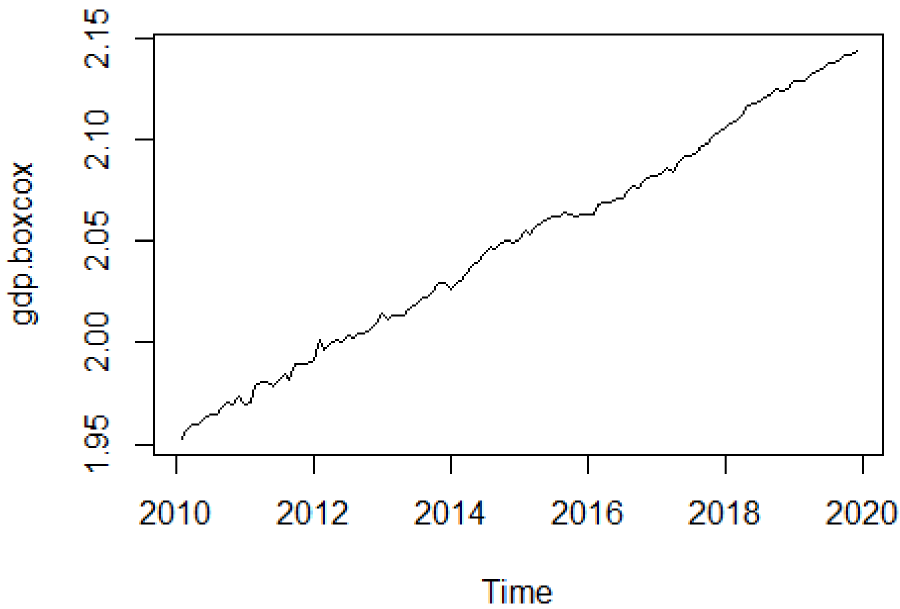

```r
crime.boxcox=BoxCox(var.traintest[,'crime'], lambda = 'auto')
plot(crime.boxcox)
```



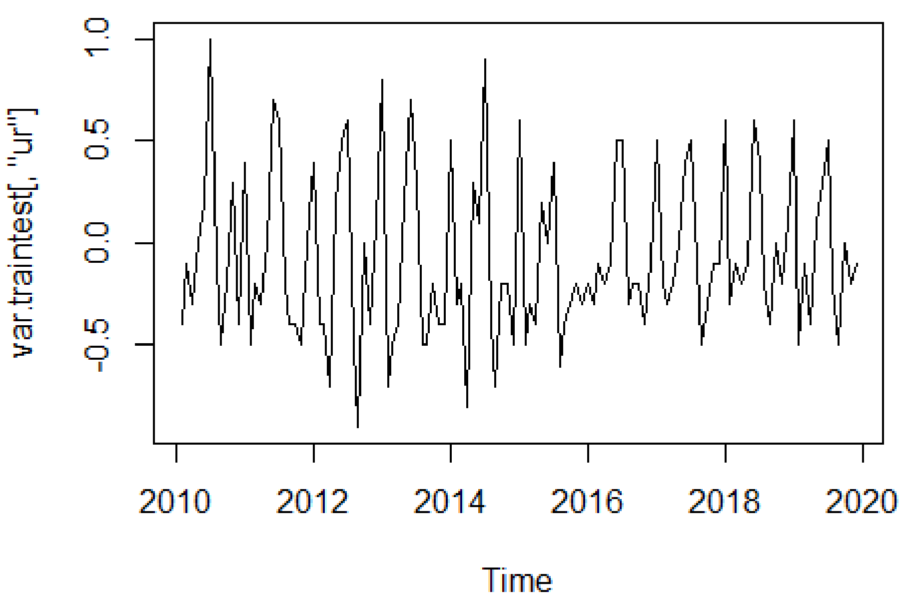
```r
plot(var.traintest[,'gdp'])
```




```r
gdp.boxcox=BoxCox(var.traintest[,'gdp'], lambda = 'auto')
plot(gdp.boxcox)
```




```r
plot(var.traintest[,'ur'])
```

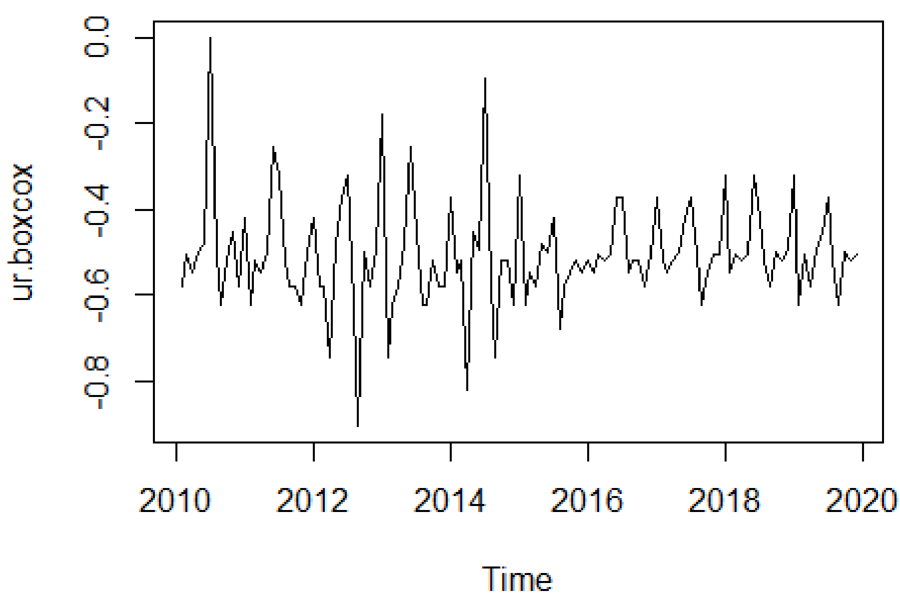



```r
ur.boxcox=BoxCox(var.traintest[,'ur'], lambda = 'auto')
```

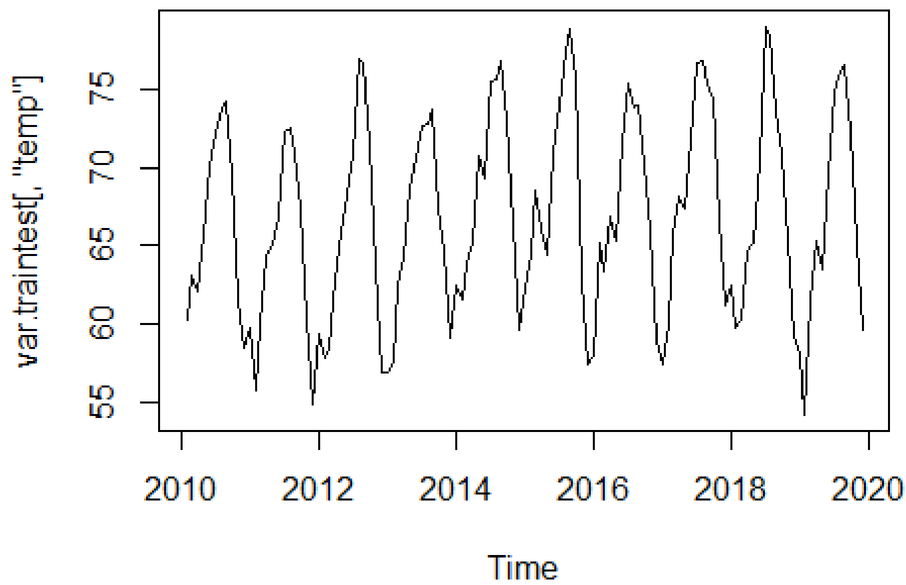
```
## Warning in optimize(guer.cv, c(lower, upper), x = x, nonseasonal.length =
## nonseasonal.length): NA/Inf replaced by maximum positive value

## Warning in optimize(guer.cv, c(lower, upper), x = x, nonseasonal.length =
## nonseasonal.length): NA/Inf replaced by maximum positive value

## Warning in optimize(guer.cv, c(lower, upper), x = x, nonseasonal.length =
## nonseasonal.length): NA/Inf replaced by maximum positive value

## Warning in optimize(guer.cv, c(lower, upper), x = x, nonseasonal.length =
## nonseasonal.length): NA/Inf replaced by maximum positive value

## Warning in optimize(guer.cv, c(lower, upper), x = x, nonseasonal.length =
## nonseasonal.length): NA/Inf replaced by maximum positive value

## Warning in optimize(guer.cv, c(lower, upper), x = x, nonseasonal.length =
## nonseasonal.length): NA/Inf replaced by maximum positive value

## Warning in optimize(guer.cv, c(lower, upper), x = x, nonseasonal.length =
## nonseasonal.length): NA/Inf replaced by maximum positive value

## Warning in optimize(guer.cv, c(lower, upper), x = x, nonseasonal.length =
## nonseasonal.length): NA/Inf replaced by maximum positive value

## Warning in optimize(guer.cv, c(lower, upper), x = x, nonseasonal.length =
## nonseasonal.length): NA/Inf replaced by maximum positive value

## Warning in optimize(guer.cv, c(lower, upper), x = x, nonseasonal.length =
## nonseasonal.length): NA/Inf replaced by maximum positive value

## Warning in optimize(guer.cv, c(lower, upper), x = x, nonseasonal.length =
## nonseasonal.length): NA/Inf replaced by maximum positive value

## Warning in optimize(guer.cv, c(lower, upper), x = x, nonseasonal.length =
## nonseasonal.length): NA/Inf replaced by maximum positive value

## Warning in optimize(guer.cv, c(lower, upper), x = x, nonseasonal.length =
## nonseasonal.length): NA/Inf replaced by maximum positive value

## Warning in optimize(guer.cv, c(lower, upper), x = x, nonseasonal.length =
## nonseasonal.length): NA/Inf replaced by maximum positive value

## Warning in optimize(guer.cv, c(lower, upper), x = x, nonseasonal.length =
## nonseasonal.length): NA/Inf replaced by maximum positive value

## Warning in optimize(guer.cv, c(lower, upper), x = x, nonseasonal.length =
## nonseasonal.length): NA/Inf replaced by maximum positive value
```

```
## Warning in optimize(guer.cv, c(lower, upper), x = x, nonseasonal.length =
## nonseasonal.length): NA/Inf replaced by maximum positive value

## Warning in optimize(guer.cv, c(lower, upper), x = x, nonseasonal.length =
## nonseasonal.length): NA/Inf replaced by maximum positive value

## Warning in optimize(guer.cv, c(lower, upper), x = x, nonseasonal.length =
## nonseasonal.length): NA/Inf replaced by maximum positive value

## Warning in optimize(guer.cv, c(lower, upper), x = x, nonseasonal.length =
## nonseasonal.length): NA/Inf replaced by maximum positive value

## Warning in optimize(guer.cv, c(lower, upper), x = x, nonseasonal.length =
## nonseasonal.length): NA/Inf replaced by maximum positive value

## Warning in optimize(guer.cv, c(lower, upper), x = x, nonseasonal.length =
## nonseasonal.length): NA/Inf replaced by maximum positive value
```

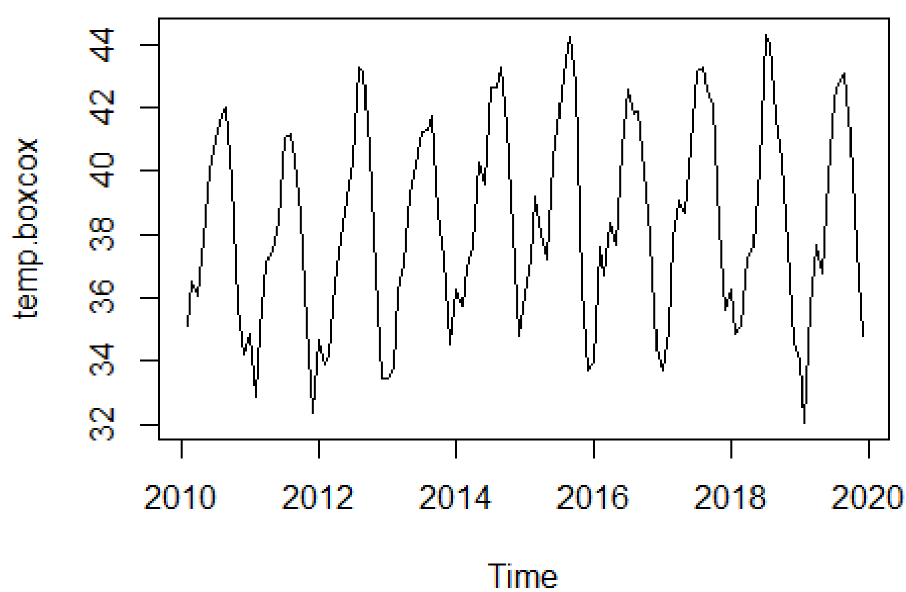
```
plot(ur.boxcox)
```



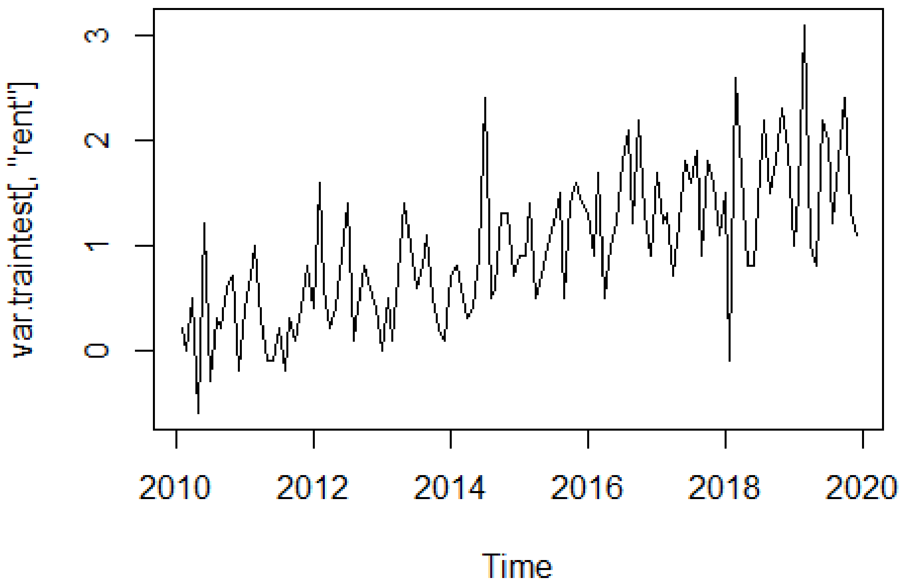

```
plot(var.traintest[,'temp'])
```



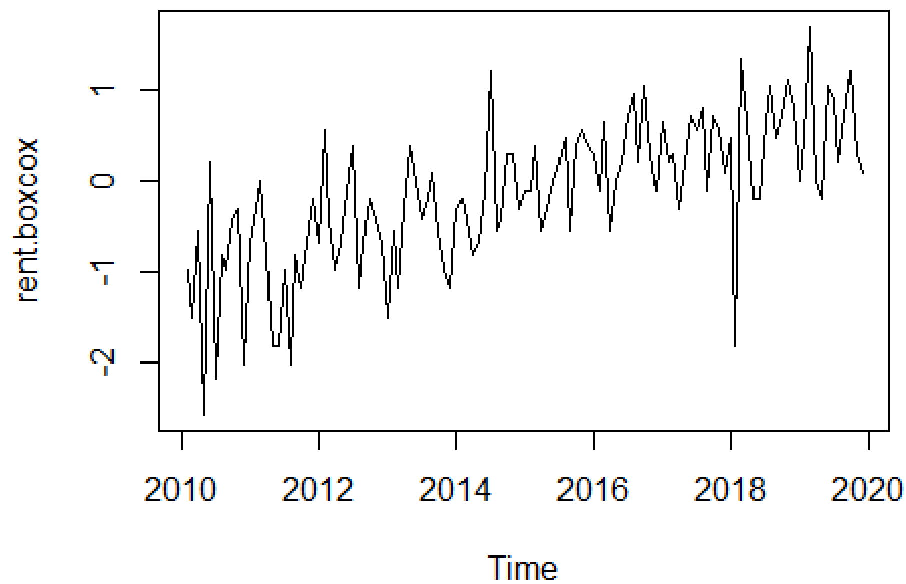

```
temp.boxcox=BoxCox(var.traintest[,'temp'], lambda = 'auto')
plot(temp.boxcox)
```



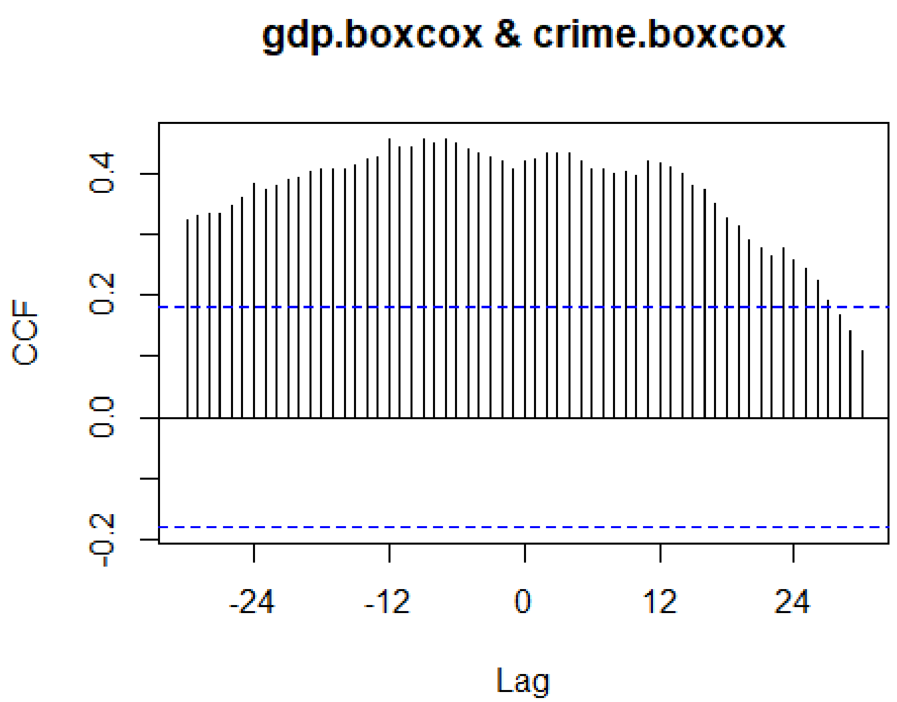
```r
plot(var.traintest[,'rent'])
```



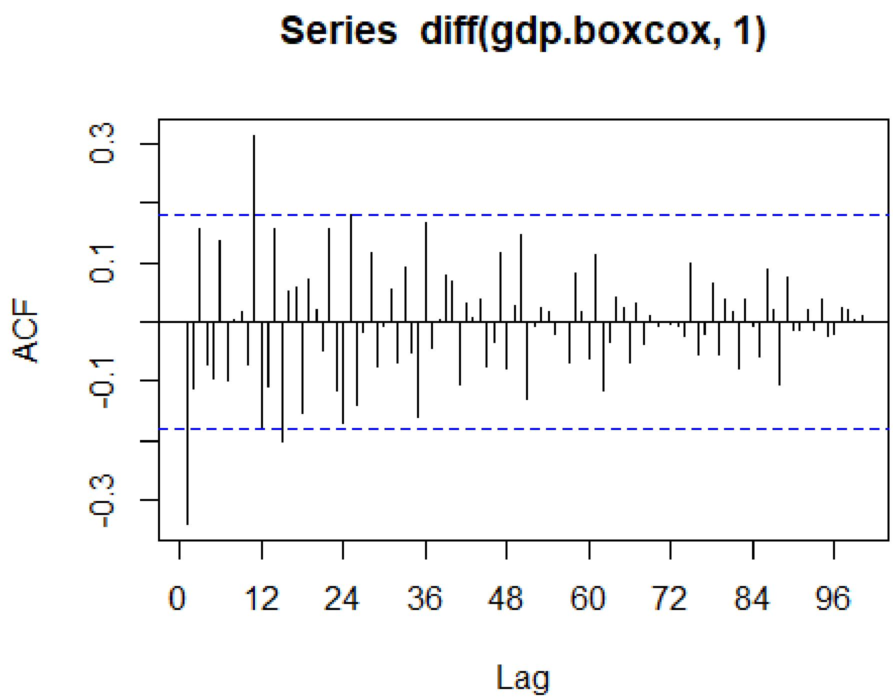

```r
rent.boxcox=BoxCox(var.traintest[,'rent'], lambda = 'auto')
plot(rent.boxcox)
```



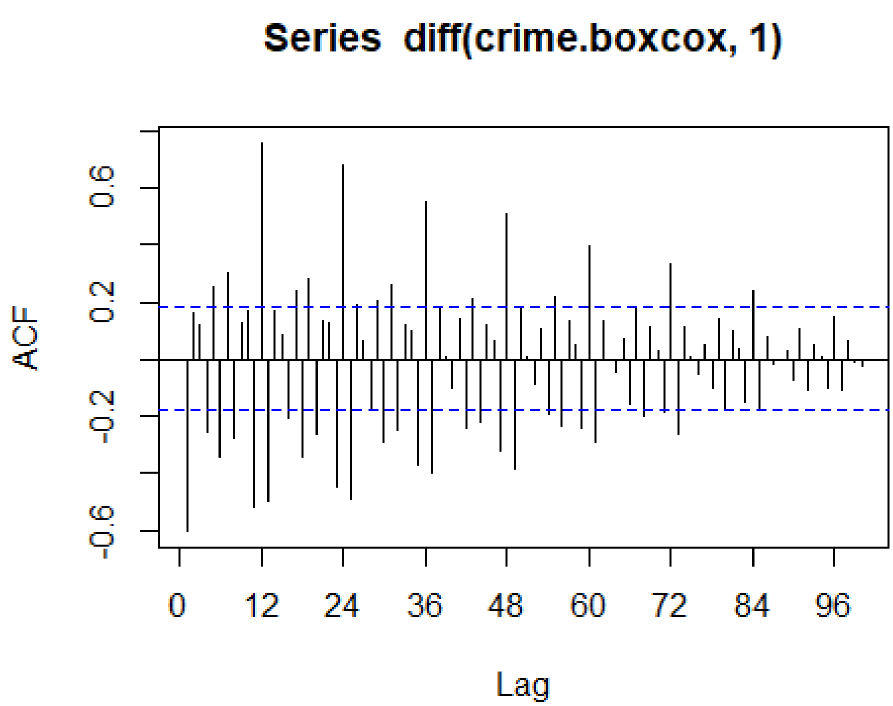

```r
# check cross correlation

Ccf(gdp.boxcox, crime.boxcox, lag.max = 30) # original variables, lag 13
```



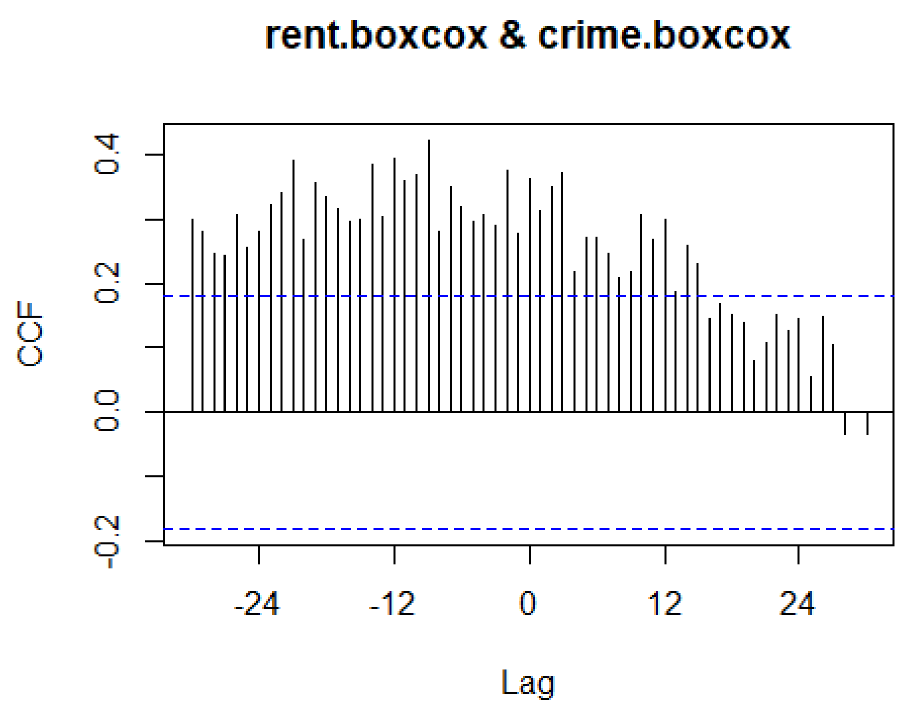
```r
Acf(diff(gdp.boxcox,1), lag.max = 100) # a bit seasonality left
```



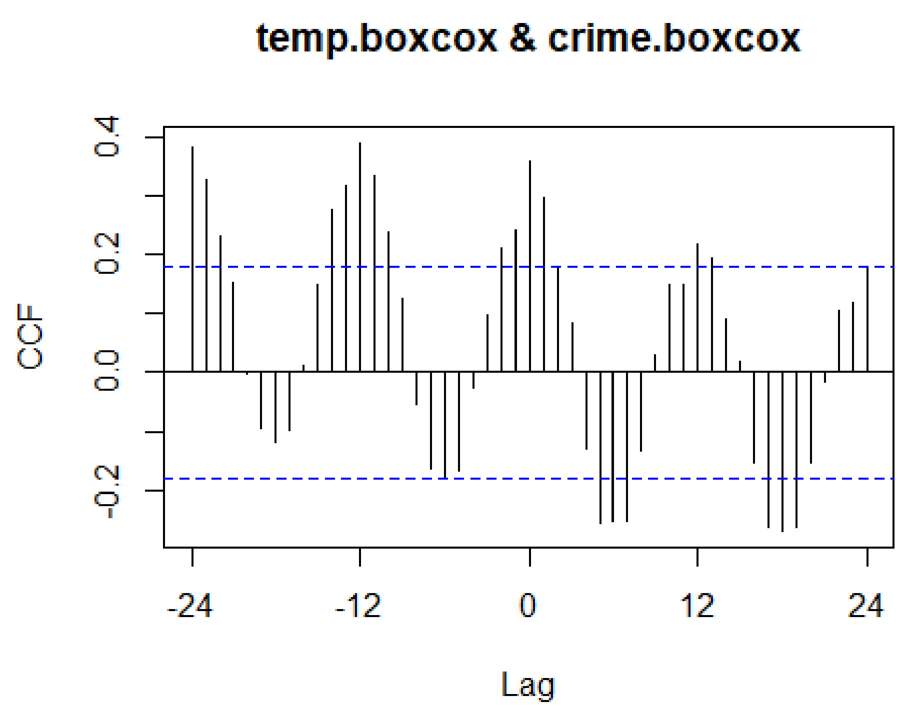

```r
Acf(diff(crime.boxcox,1), lag.max = 100) # seasonality left
```



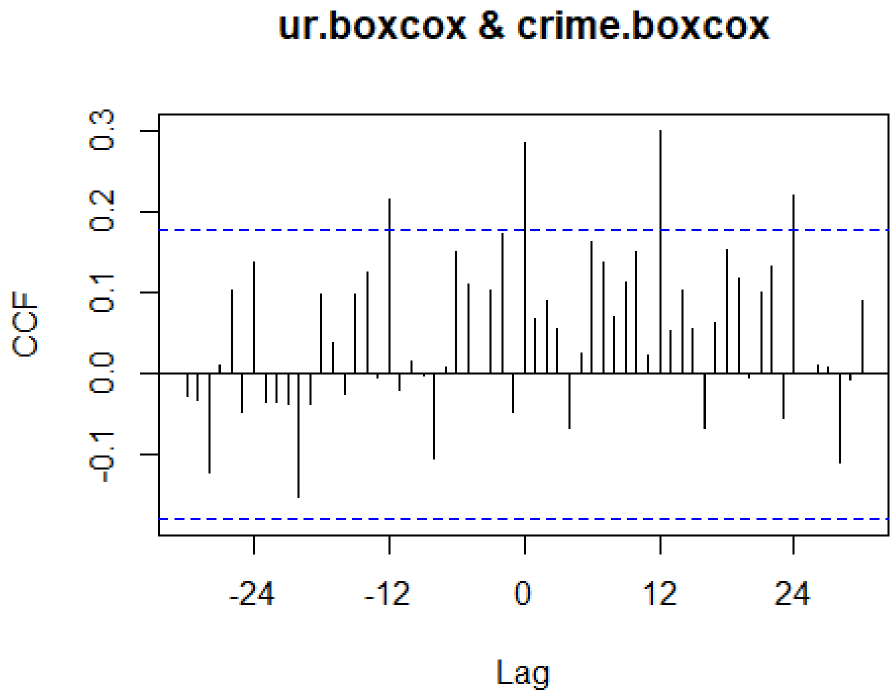
```r
Ccf(rent.boxcox, crime.boxcox, lag.max = 30) # original variables, lag 1
```




```r
Ccf(temp.boxcox, crime.boxcox, lag.max = 24) # original variables, lag 12
```



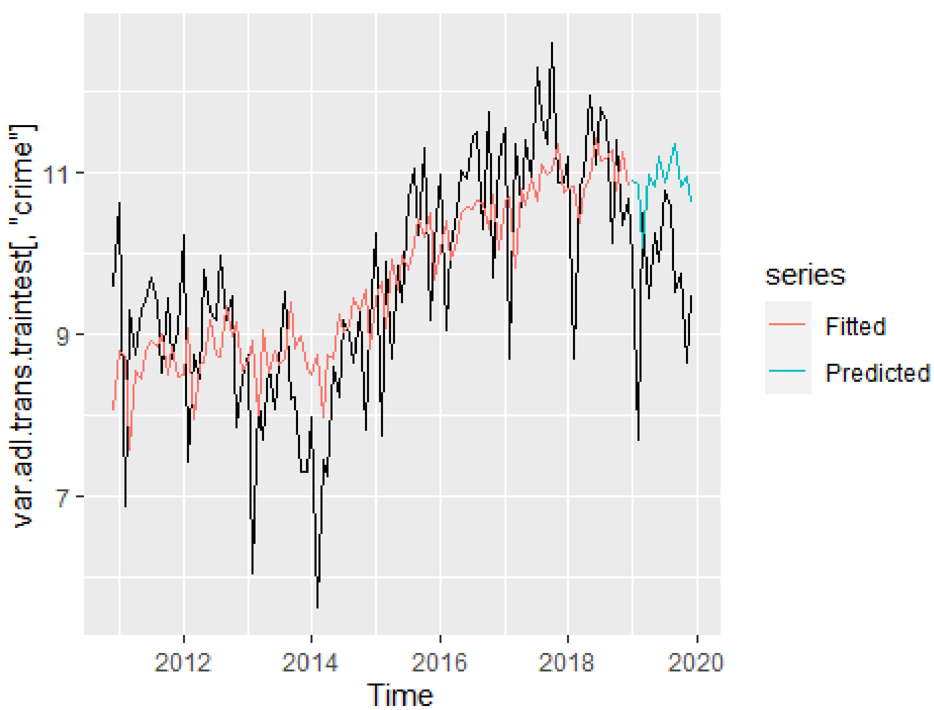
```r
Ccf(ur.boxcox, crime.boxcox, lag.max = 30) # original variables, lag 7
```




```r
var.adl.trans = ts.intersect(crime=crime.boxcox,
                             crime.lag=stats::lag(crime.boxcox, -1),
                             gdp.lag=stats::lag(gdp.boxcox, -10),
                             ur.lag=stats::lag(ur.boxcox, -7),
                             temp=temp.boxcox,
                             rent.lag=stats::lag(rent.boxcox, -1))

var.adl.trans.train=window(var.adl.trans, end=c(2018,12))
var.adl.trans.test=window(var.adl.trans, start=c(2019,1), end=c(2019,12))
var.adl.trans.traintest=window(var.adl.trans, end=c(2019,12))

adl.model = lm(crime~crime.lag+gdp.lag+rent.lag+ur.lag,
data=var.adl.trans.train)
accuracy(adl.model$fitted.values, var.adl.trans.train[,'crime'])

##                    ME     RMSE       MAE      MPE    MAPE       ACF1
Theil's U
## Test set 1.831998e-16 1.046452 0.8317969 -1.39865 9.30801 -0.1586785
0.7992504

adl.model.pred=predict(adl.model, newdata=var.adl.trans.test)
accuracy(adl.model.pred, var.adl.trans.test[,'crime'])

##                 ME      RMSE       MAE        MPE      MAPE       ACF1 Theil's
U
```

```
## Test set -1.177173 1.497225 1.245042 -13.06619 13.71273 -0.4446272
1.042856
```

```
autoplot(var.adl.trans.traintest[,'crime']) +
  autolayer(ts(adl.model$fitted.values, end=c(2018,12), frequency =
12),series = "Fitted") +
  autolayer(ts(adl.model.pred, end=c(2019,12), frequency = 12),series =
"Predicted")
```



There was not major improvement seen from box cox transformation

END OF CODE