M6: ARIMA Models in R

Luana Lima

02/17/2021

Setting R code chunk options

First R code chunk is used for setting the options for all R code chunks. The choice echo=TRUE means both code and output will appear on report, include = FALSE neither code nor output is printed.

Loading packages and initializing

Second R code chunk is for loading packages. By setting message = FALSE, the code will appear but not the output.

```
library(lubridate)
library(ggplot2)
library(forecast)
library(Kendall)
library(tseries)
library(outliers)
library(tidyverse)

#New packages for M6
#install.packages("cowplot")
library(cowplot)
```

Importing data

For this module we will work with monthly average for electricity retail price in US. The data is from the U.S. Energy Information Administration and can be download [here][https://www.eia.gov/electricity/data/browser/#/topic/7?agg=2,0,1&geo=g&freq=M%2013:41:41%20GMT-0500%20(EST)].

```
#Importing time series data from text file#
electricity_price <- read.csv(file="./Data/Average_retail_price_of_electricity_United_States_monthly.csv

#Inspect data
head(electricity_price)
```

```
##      Month all.sectors.cents.per.kilowatthour
## 1 Nov 2020                              10.45
## 2 Oct 2020                              10.64
## 3 Sep 2020                              11.07
## 4 Aug 2020                              11.11
## 5 Jul 2020                              11.14
## 6 Jun 2020                              10.96
##   residential.cents.per.kilowatthour commercial.cents.per.kilowatthour
## 1                              13.35                             10.59
## 2                              13.60                             10.73
```

```
## 3                              13.55                            11.07
## 4                              13.31                            10.95
## 5                              13.26                            10.90
## 6                              13.28                            10.95
##   industrial.cents.per.kilowatthour
## 1                              6.48
## 2                              6.72
## 3                              7.01
## 4                              7.09
## 5                              7.17
## 6                              6.94
```

```r
nvar <- ncol(electricity_price) - 1
nobs <- nrow(electricity_price)

#Preparing the data - create date object and rename columns
electricity_price_processed <-
  electricity_price %>%
  mutate( Month = my(Month) ) %>%
  rename( All.sectors = all.sectors.cents.per.kilowatthour ) %>%
  rename( Residential = residential.cents.per.kilowatthour ) %>%
  rename( Commercial = commercial.cents.per.kilowatthour ) %>%
  rename( Industrial = industrial.cents.per.kilowatthour ) %>%
  arrange( Month )

head(electricity_price_processed)
```

```
##        Month All.sectors Residential Commercial Industrial
## 1 2001-01-01        6.75        7.73       7.25       4.73
## 2 2001-02-01        6.87        8.04       7.51       4.80
## 3 2001-03-01        7.01        8.32       7.70       4.86
## 4 2001-04-01        7.02        8.46       7.73       4.87
## 5 2001-05-01        7.17        8.83       7.77       5.00
## 6 2001-06-01        7.58        9.07       8.13       5.23
```

```r
summary(electricity_price_processed)
```

```
##      Month             All.sectors      Residential      Commercial
##  Min.   :2001-01-01   Min.   : 6.750   Min.   : 7.73   Min.   : 7.250
##  1st Qu.:2005-12-16   1st Qu.: 8.520   1st Qu.: 9.82   1st Qu.: 9.070
##  Median :2010-12-01   Median : 9.720   Median :11.77   Median :10.080
##  Mean   :2010-11-30   Mean   : 9.381   Mean   :11.23   Mean   : 9.746
##  3rd Qu.:2015-11-16   3rd Qu.:10.305   3rd Qu.:12.64   3rd Qu.:10.540
##  Max.   :2020-11-01   Max.   :11.140   Max.   :13.60   Max.   :11.170
##    Industrial
##  Min.   :4.71
##  1st Qu.:5.99
##  Median :6.58
##  Mean   :6.37
##  3rd Qu.:6.89
##  Max.   :7.72
```

```r
#No NAs so we don't need to worry about missing values
```

Transforming data into time series object

Many of the functions we will use require a time series object. You can transform your data in a time series using the function *ts()*.

```
ts_electricity_price <- ts(electricity_price_processed[,2:(nvar+1)],
                           start=c(year(electricity_price_processed$Month[1]),month(electricity_price_pr
                           frequency=12)
#note that we are only transforming columns with electricity price, not the date columns
head(ts_electricity_price,15)
```

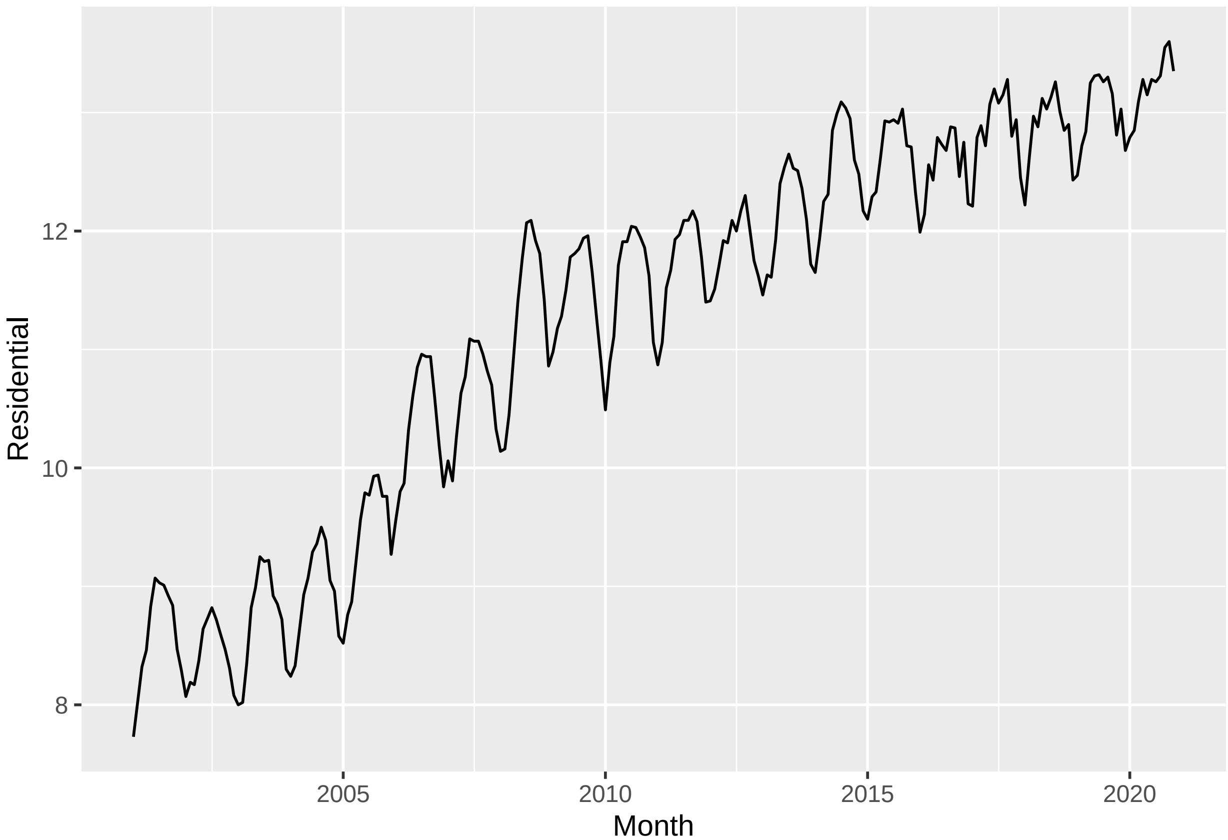
```
##          All.sectors Residential Commercial Industrial
## Jan 2001        6.75        7.73       7.25       4.73
## Feb 2001        6.87        8.04       7.51       4.80
## Mar 2001        7.01        8.32       7.70       4.86
## Apr 2001        7.02        8.46       7.73       4.87
## May 2001        7.17        8.83       7.77       5.00
## Jun 2001        7.58        9.07       8.13       5.23
## Jul 2001        7.88        9.03       8.41       5.57
## Aug 2001        7.84        9.01       8.35       5.50
## Sep 2001        7.62        8.92       8.22       5.31
## Oct 2001        7.43        8.84       8.27       5.07
## Nov 2001        7.02        8.47       7.73       4.78
## Dec 2001        7.03        8.29       7.66       4.78
## Jan 2002        6.95        8.07       7.49       4.73
## Feb 2002        6.97        8.19       7.68       4.76
## Mar 2002        6.95        8.17       7.72       4.73
```

```
tail(ts_electricity_price,15)
```

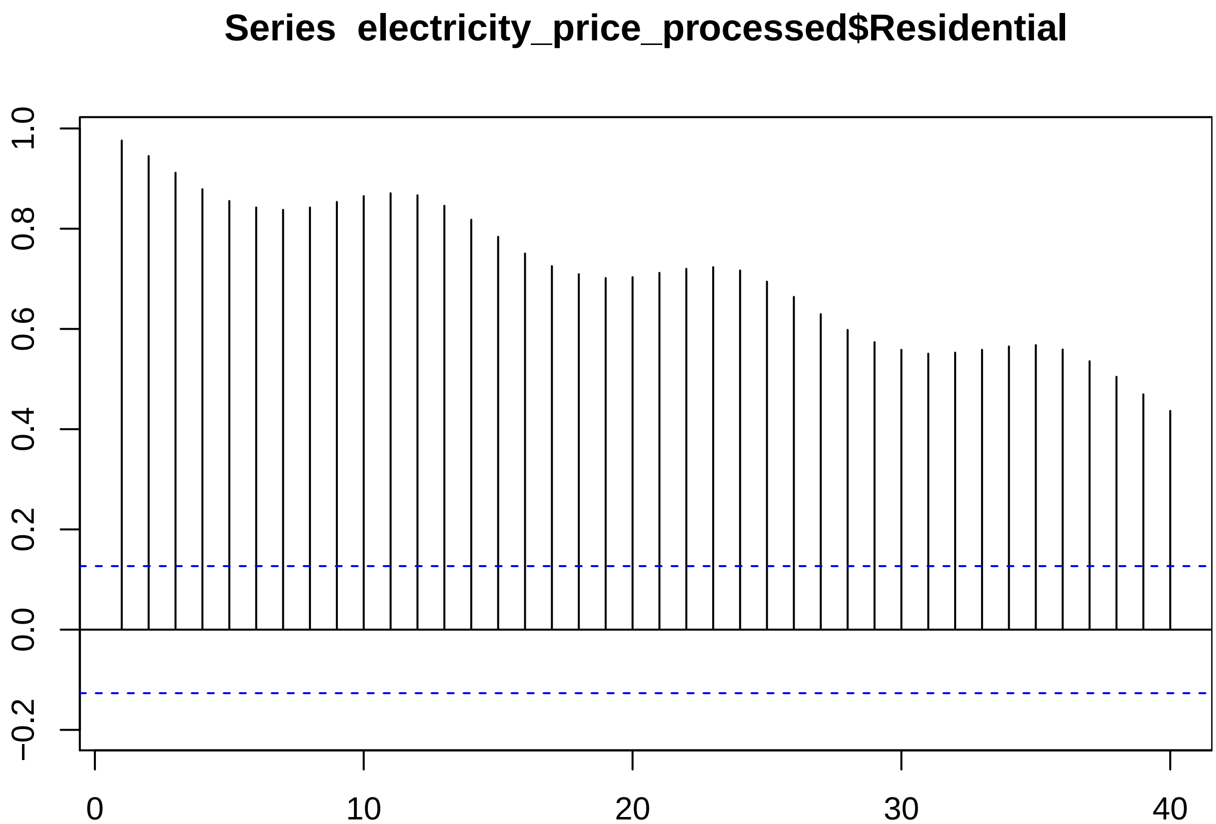
```
##          All.sectors Residential Commercial Industrial
## Sep 2019       10.82       13.16      10.96       7.06
## Oct 2019       10.39       12.81      10.74       6.84
## Nov 2019       10.38       13.03      10.57       6.72
## Dec 2019       10.22       12.68      10.32       6.38
## Jan 2020       10.28       12.79      10.24       6.33
## Feb 2020       10.29       12.85      10.36       6.41
## Mar 2020       10.29       13.09      10.41       6.38
## Apr 2020       10.42       13.28      10.42       6.40
## May 2020       10.47       13.15      10.46       6.53
## Jun 2020       10.96       13.28      10.95       6.94
## Jul 2020       11.14       13.26      10.90       7.17
## Aug 2020       11.11       13.31      10.95       7.09
## Sep 2020       11.07       13.55      11.07       7.01
## Oct 2020       10.64       13.60      10.73       6.72
## Nov 2020       10.45       13.35      10.59       6.48
```

Initial Plots

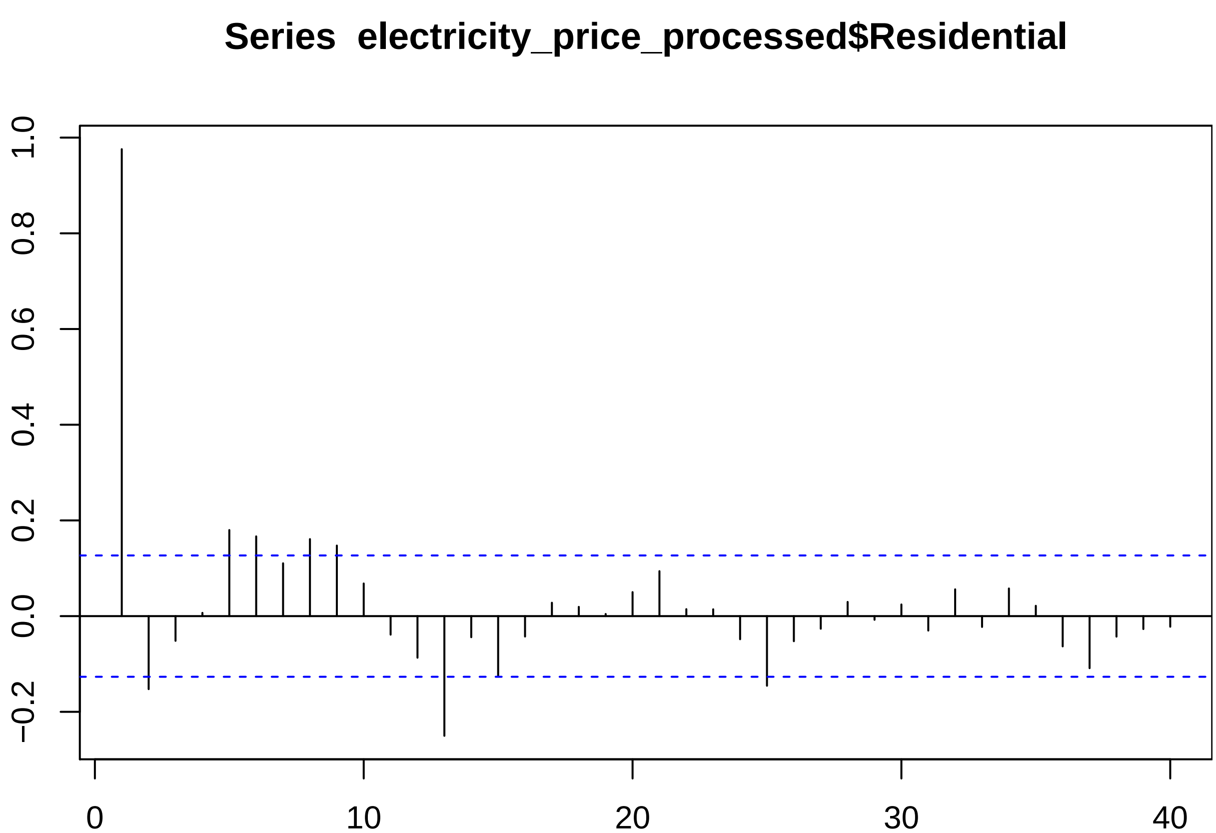
```
#Generating a box plot by factor where factor is month of the year

TS_Plot <-
  ggplot(electricity_price_processed, aes(x=Month, y=Residential)) +
      geom_line()
plot(TS_Plot)
```




```
#Note that although the date is reversed on the data frame, since we are using the ggplot and a date ob

#ACF and PACF plots
par(mar=c(3,3,3,0))

ACF_Plot <- Acf(electricity_price_processed$Residential, lag = 40, plot = TRUE)
```



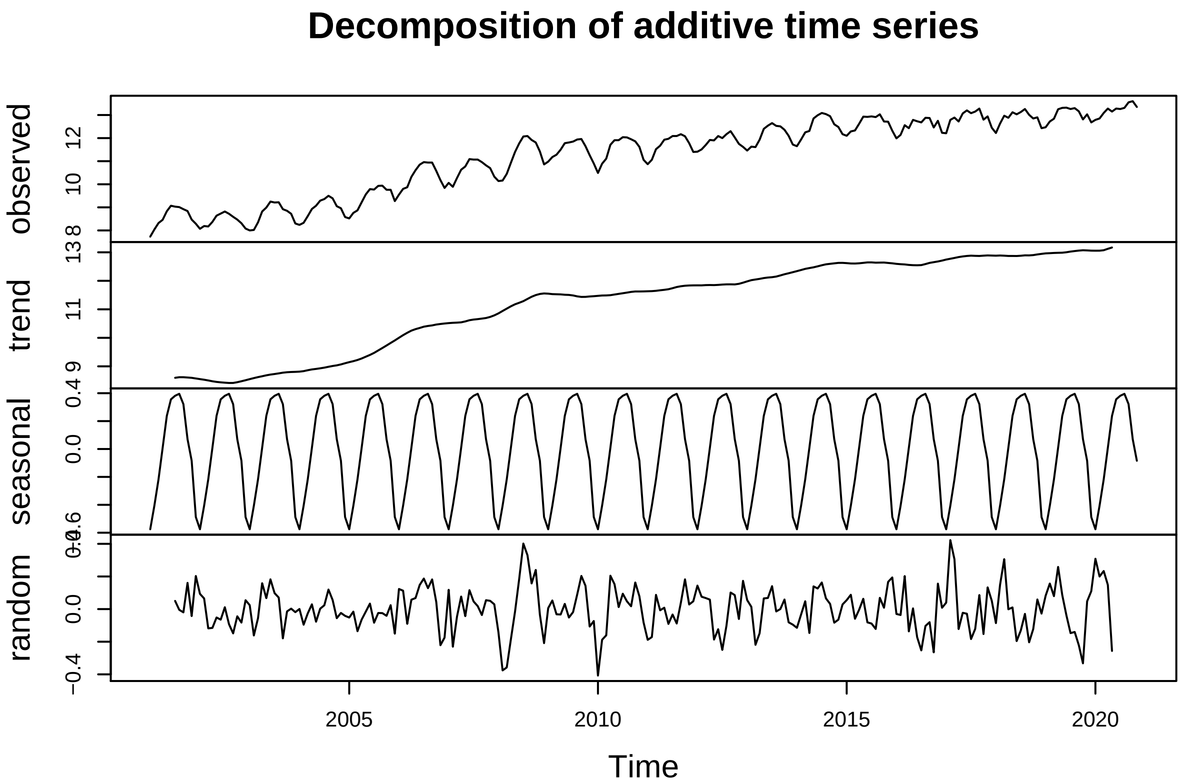

```
PACF_Plot <- Pacf(electricity_price_processed$Residential, lag = 40)
```



Decomposing the time series and removing seasonality

The plots from the previous section show the data has a seasonal component. Since we are working with non-seasonal ARIMA, we need to decompose the series and eliminate the seasonality.

```
#Using R decompose function
decompose_residential_price <- decompose(ts_electricity_price[,"Residential"],"additive")
plot(decompose_residential_price)
```

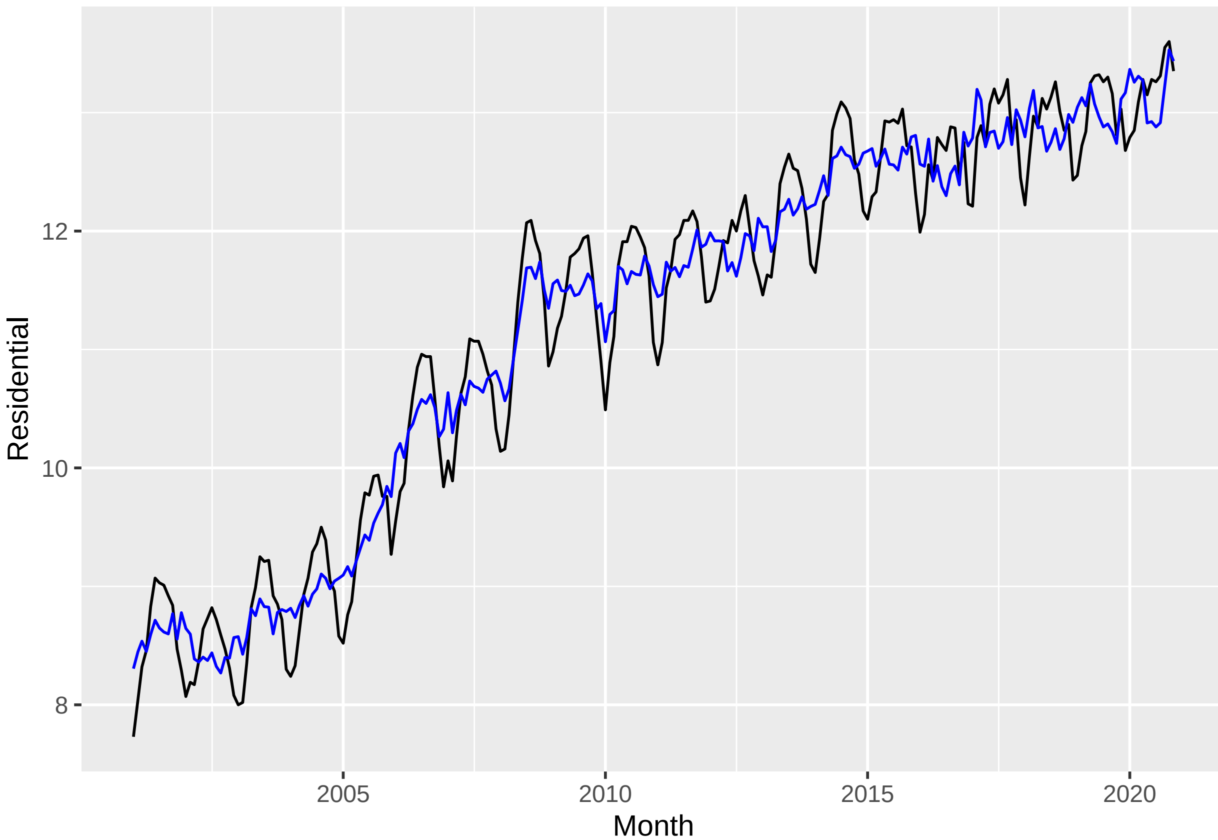



```
#Note the time is reversed on this plot. Price should be increasing over time
```

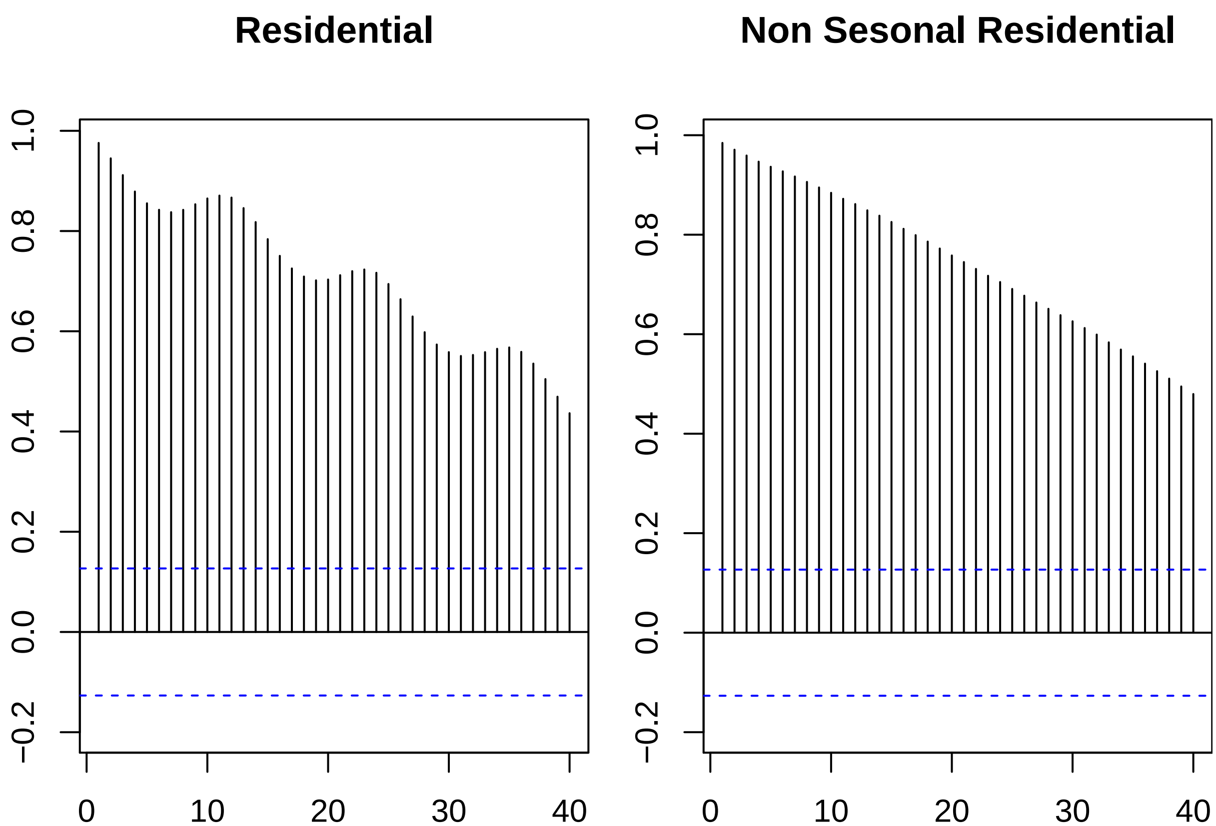
To take seasonality only out of the data set, we will use function seasadj() from package forecast. The function returns seasonally adjusted data constructed by removing the seasonal component. It takes one main object that should be created using decompose() function.

The ACF and PACF from the seasonal adjusted series will help you specify components **p** and **q** of the ARIMA(p,d,q).

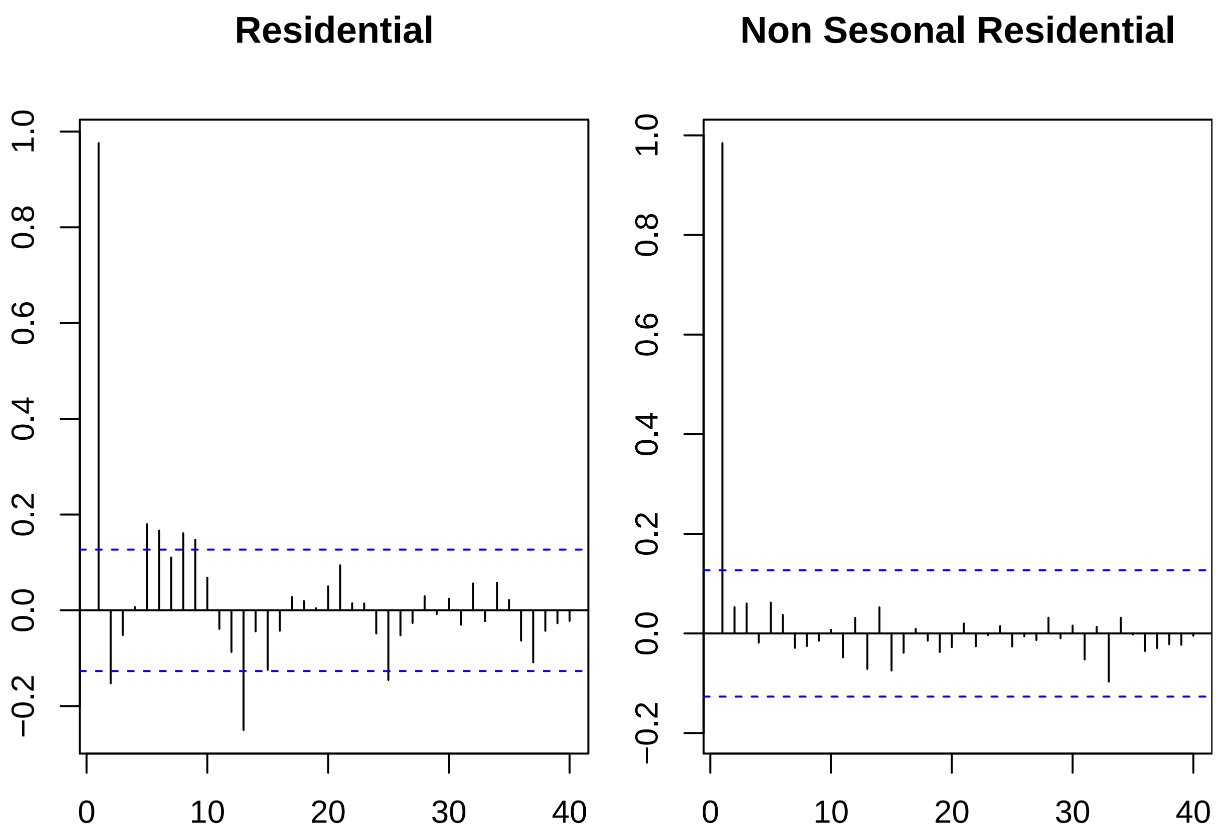
```
#Creating non-seasonal residential price time series
deseasonal_residential_price <- seasadj(decompose_residential_price)

df_residential <- data.frame( Month = electricity_price_processed$Month,
                              Residential = electricity_price_processed$Residential,
                              NonSeasonalResidential = as.numeric(deseasonal_residential_price))

ggplot(df_residential, aes(x=Month)) +
      geom_line(aes(x=Month,y=Residential),color="black") +
      geom_line(aes(x=Month,y=NonSeasonalResidential),color="blue")
```




```
#Comparing ACFs
par(mar=c(3,3,3,0));par(mfrow=c(1,2))
Acf(df_residential$Residential,lag.max=40,main="Residential")
Acf(df_residential$NonSeasonalResidential,lag.max=40,main="Non Sesonal Residential")
```




```
#Note seasonality is gone!

#Comparing PACFs
par(mar=c(3,3,3,0));par(mfrow=c(1,2))
Pacf(df_residential$Residential,lag.max=40,main="Residential")
Pacf(df_residential$NonSeasonalResidential,lag.max=40,main="Non Sesonal Residential")
```



The new ACF plot show a slow decay which is a sign of non-stationarity.

Run stationarity test

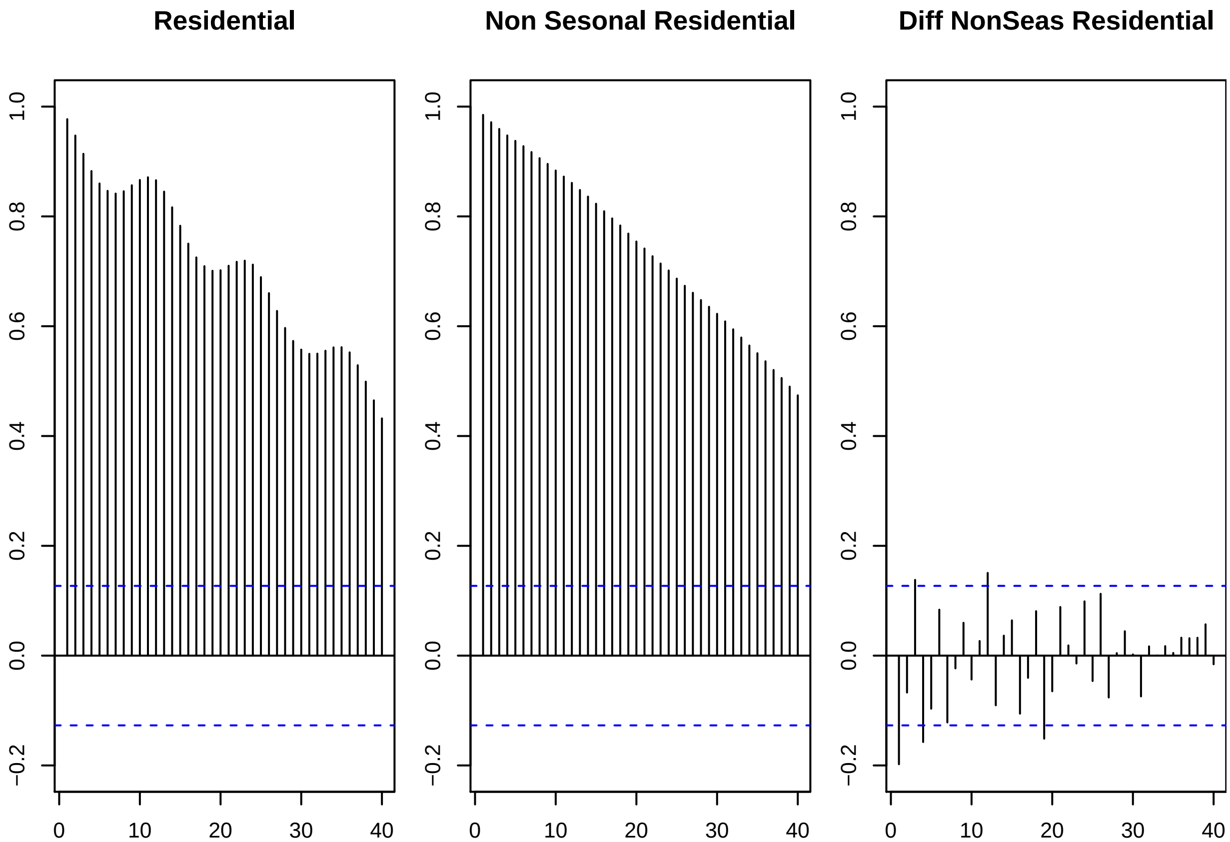
Always check for stationarity before fitting ARIMA models. This will help specify component **d** of the ARIMA(p,d,q). If there is a trend you need to set **d=1**.

```
#Run ADF
#adf.test(deseasonal_price,alternative="stationary")
print((adf.test(deseasonal_residential_price,alternative="stationary")))
```

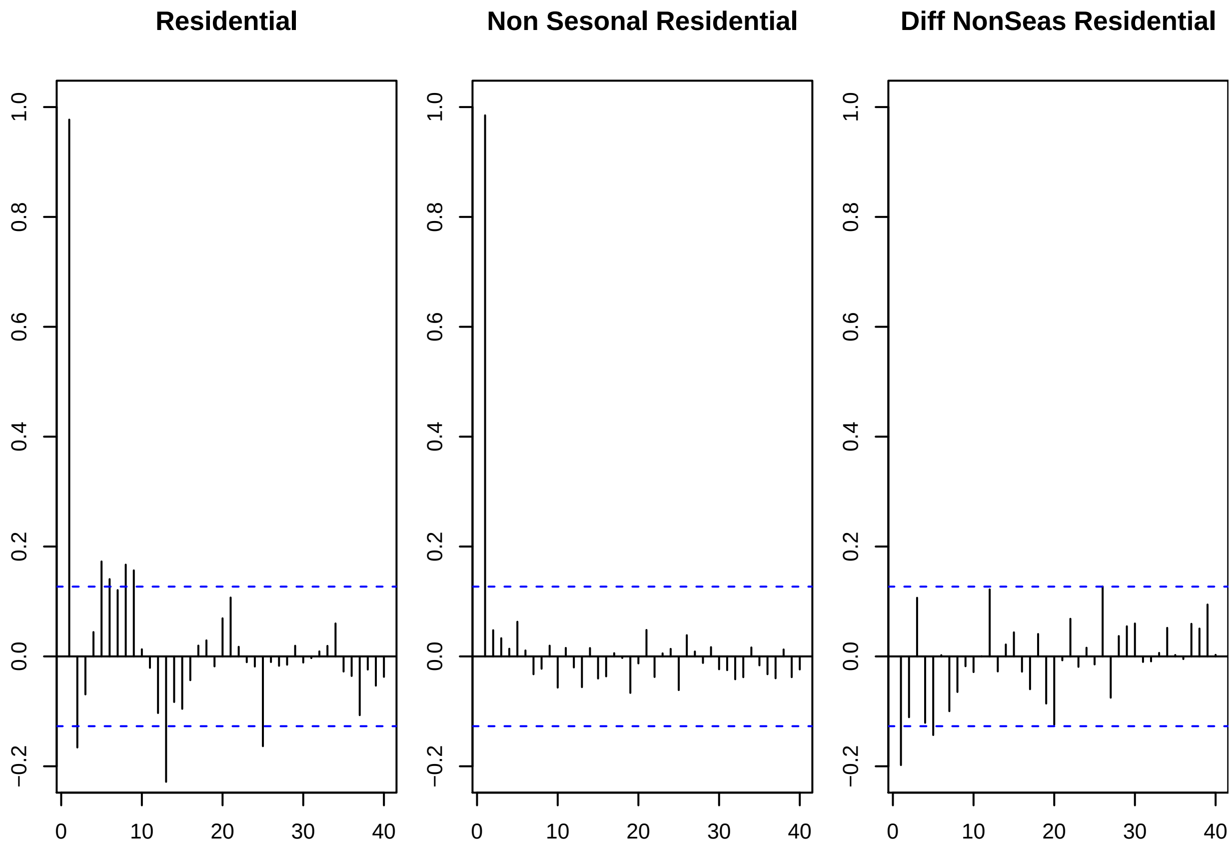
```
##
##  Augmented Dickey-Fuller Test
##
## data:  deseasonal_residential_price
## Dickey-Fuller = -1.4098, Lag order = 6, p-value = 0.824
## alternative hypothesis: stationary
```

```
#Note that p-value greater then 0.05 so we accept H0. Data has stochastic trend
#Lets difference the series to remove the trend.
#Difference the data at lag 1
deseasonal_residential_price_diff <- diff(deseasonal_residential_price,differences=1)

#Add the new series to our data frame
df_residential_full <-
  df_residential %>%
  cbind(ResidentialDiff = c(NA,as.numeric(deseasonal_residential_price_diff))) %>%
  na.omit(residentialDiff)
```

```r
#Check autocorrelation plot again


#Comparing ACFs
par(mar=c(3,3,3,0));par(mfrow=c(1,3))
Acf(df_residential_full$Residential,lag.max=40,main="Residential",ylim=c(-.2,1))
Acf(df_residential_full$NonSeasonalResidential,lag.max=40,main="Non Sesonal Residential",ylim=c(-.2,1))
Acf(df_residential_full$ResidentialDiff,lag.max=40,main="Diff NonSeas Residential",ylim=c(-.2,1))
```




```r
#Comparing PACFs
par(mar=c(3,3,3,0));par(mfrow=c(1,3))
Pacf(df_residential_full$Residential,lag.max=40,main="Residential",ylim=c(-.2,1))
Pacf(df_residential_full$NonSeasonalResidential,lag.max=40,main="Non Sesonal Residential",ylim=c(-.2,1))
Pacf(df_residential_full$ResidentialDiff,lag.max=40,main="Diff NonSeas Residential",ylim=c(-.2,1))
```



Manually fitting ARIMA models to series

In the section we will manually fit ARIMA models to the residential electricity price series using function Arima() from package *forecast*. Some important arguments for Arima() are:

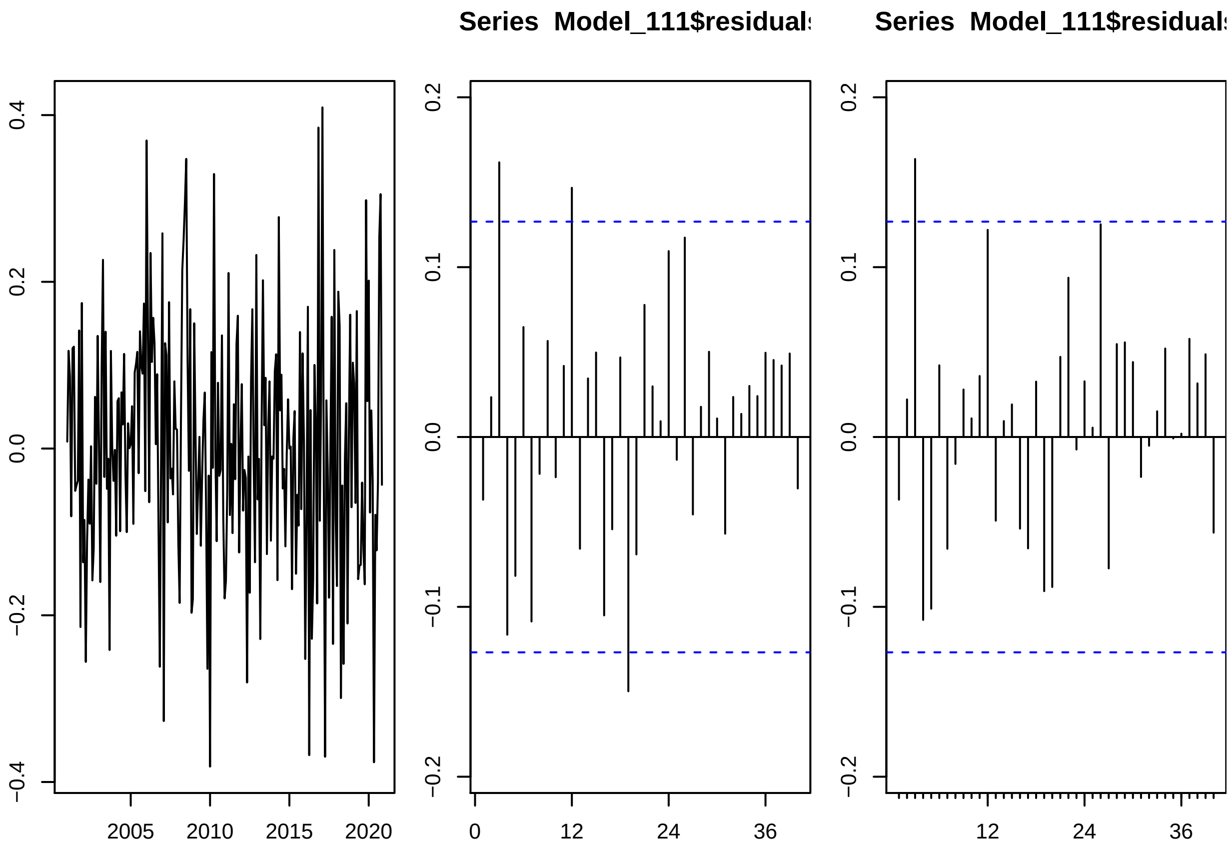
y: univariate (single vector) ts object *order=c(, ,)*: three orders (p,d,q) of non-seasonal part of the ARIMA in this order *include.mean*: the default is TRUE for undifferenced series, which means the model will include a mean term, and FALSE when d > 0 *include.drift*: the default is FALSE, but changing to TRUE might lead to better fits. The drift will be necessary when the series mean is not zero even after differencing

```
#Remember the order d=1 will perform the differencing,
#so lets try ARIMA(1,1,1) on the non-seasonal residential data before differencing
Model_111 <- Arima(deseasonal_residential_price,order=c(1,1,1),include.drift=TRUE)
print(Model_111)
```

```
## Series: deseasonal_residential_price
## ARIMA(1,1,1) with drift
##
## Coefficients:
##          ar1      ma1   drift
##       0.5798  -0.7702  0.0209
## s.e.  0.1673   0.1324  0.0052
##
## sigma^2 estimated as 0.02115:  log likelihood=122.61
## AIC=-237.22   AICc=-237.05   BIC=-223.33
```

```
compare_aic <- data.frame(Model_111$aic)

#Check residuals series, if white noise we got a good fit
```

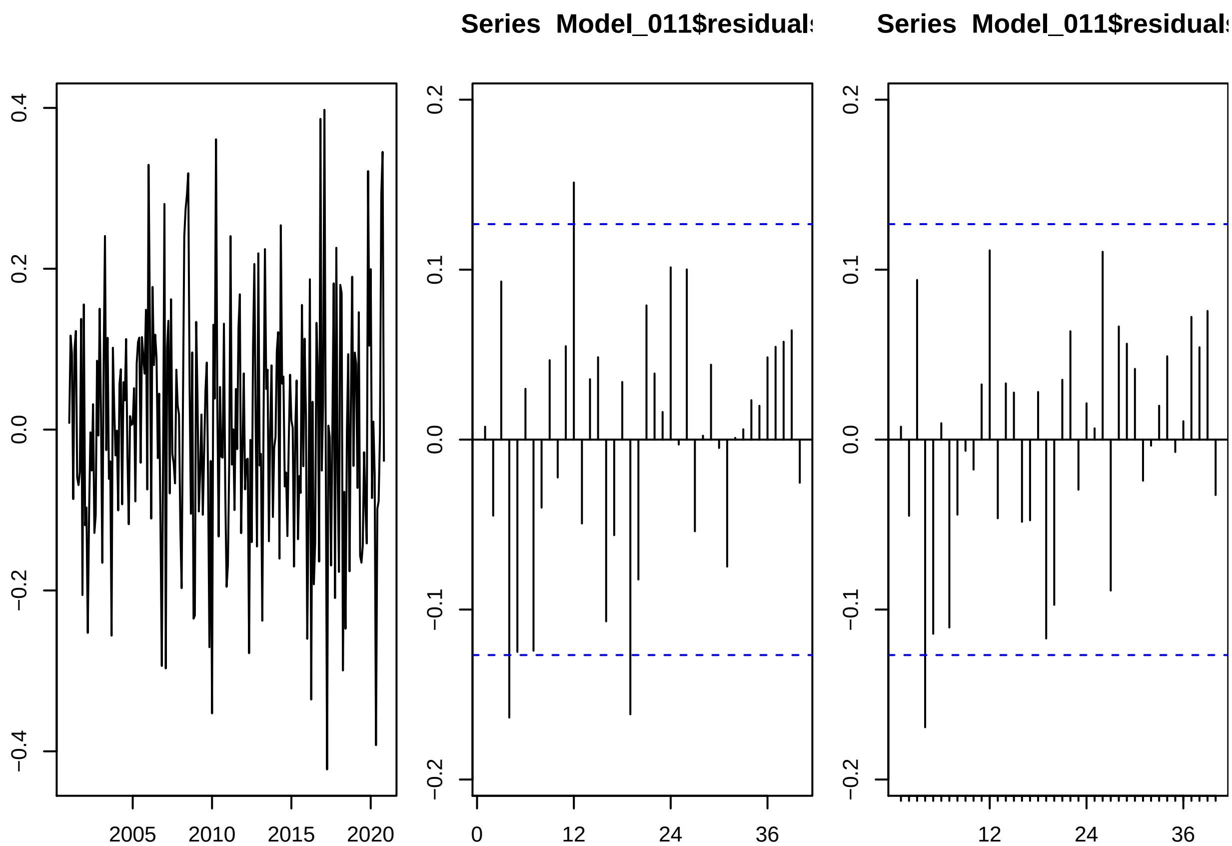
```r
par(mar=c(3,3,3,0));par(mfrow=c(1,3))
ts.plot(Model_111$residuals)
Acf(Model_111$residuals,lag.max=40)
Pacf(Model_111$residuals,lag.max=40)
```




```r
#Now let's try ARIMA(0,1,1)
Model_011 <- Arima(deseasonal_residential_price,order=c(0,1,1),include.drift=TRUE)
print(Model_011)
```

```
## Series: deseasonal_residential_price
## ARIMA(0,1,1) with drift
##
## Coefficients:
##           ma1   drift
##       -0.2254  0.0215
## s.e.   0.0653  0.0073
##
## sigma^2 estimated as 0.0212:  log likelihood=121.85
## AIC=-237.69   AICc=-237.59   BIC=-227.27
```

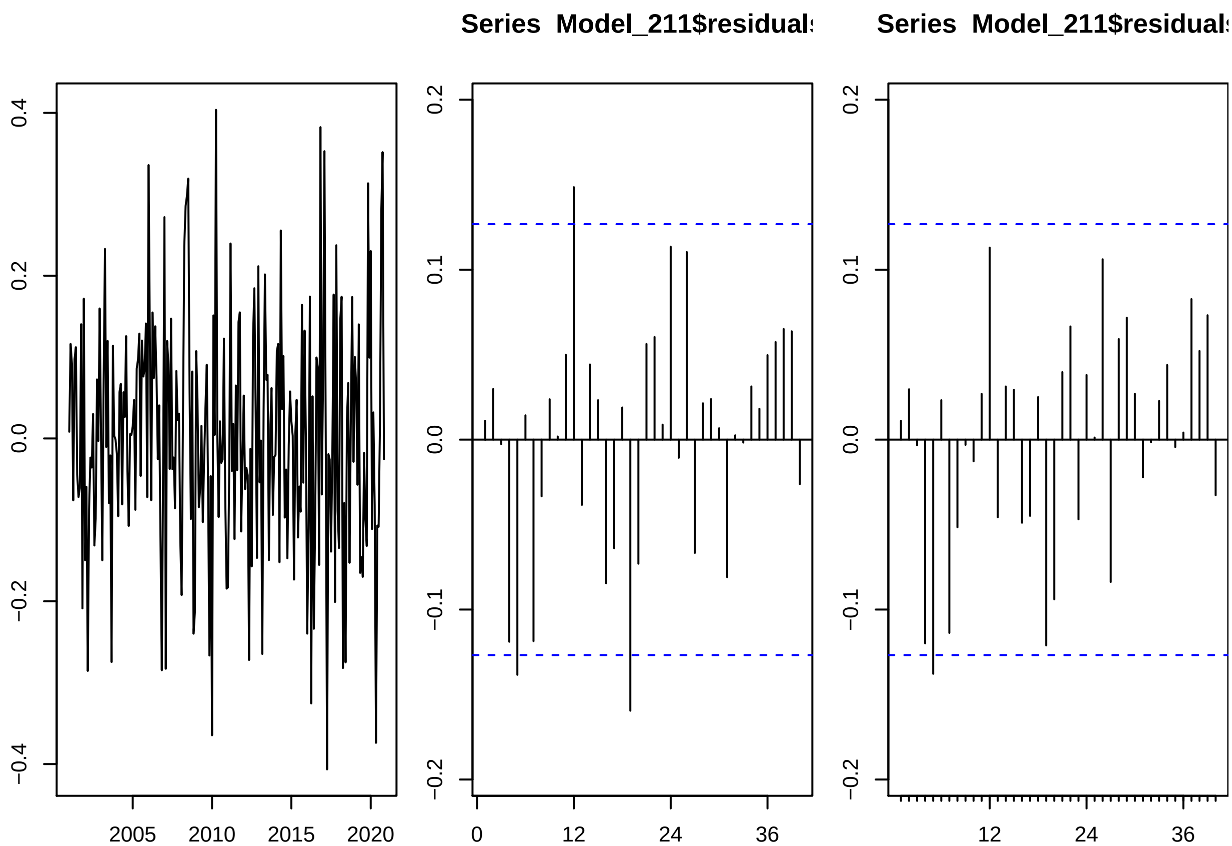
```r
compare_aic <- data.frame(compare_aic,Model_011$aic)

par(mar=c(3,3,3,0));par(mfrow=c(1,3))
ts.plot(Model_011$residuals)
Acf(Model_011$residuals,lag.max=40)
Pacf(Model_011$residuals,lag.max=40)
```




```r
#Now let's try ARIMA(2,1,1)
Model_211 <- Arima(deseasonal_residential_price,order=c(2,1,1),include.drift=TRUE)
print(Model_211)
```

```
## Series: deseasonal_residential_price
## ARIMA(2,1,1) with drift
##
## Coefficients:
##           ar1      ar2     ma1   drift
##       -0.6900  -0.2252  0.4764  0.0215
## s.e.   0.2087   0.0675  0.2067  0.0072
##
## sigma^2 estimated as 0.02103:  log likelihood=123.82
## AIC=-237.65   AICc=-237.39   BIC=-220.29
```

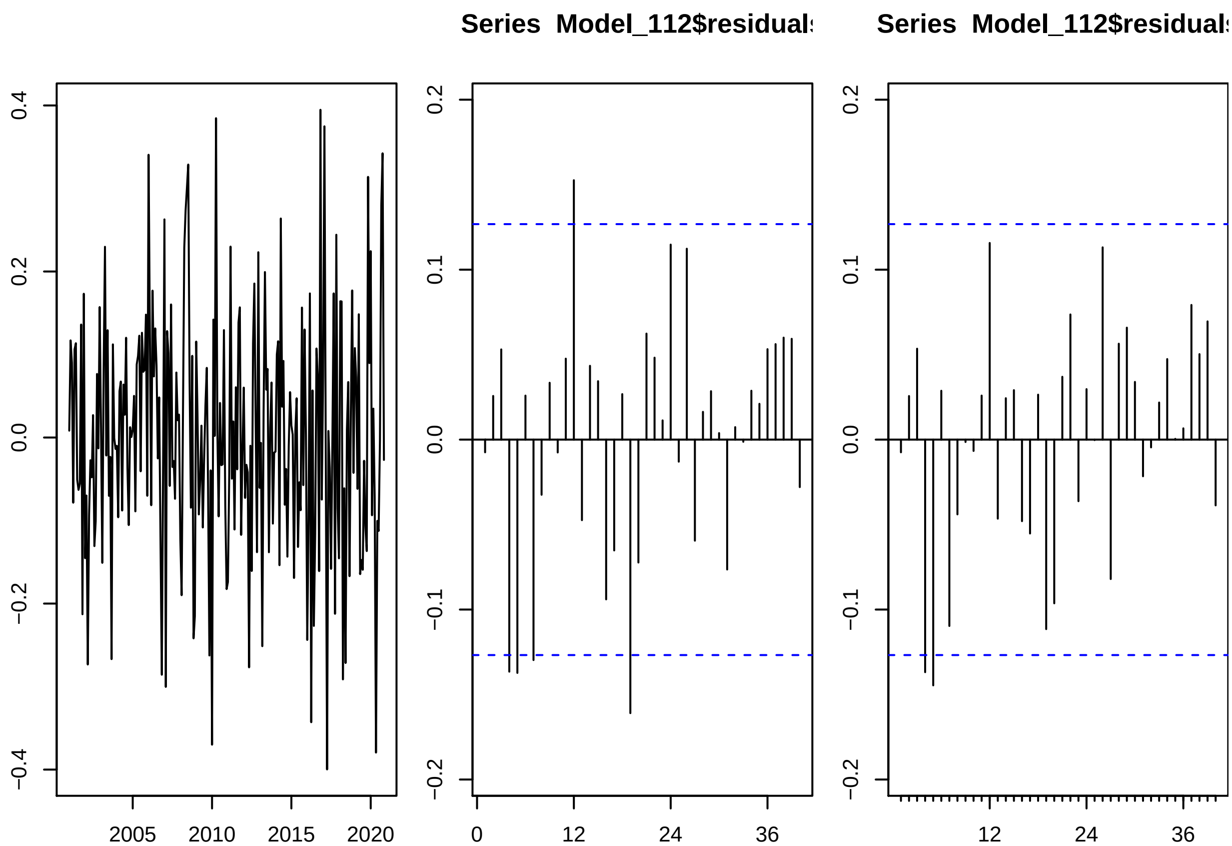
```r
compare_aic <- data.frame(compare_aic,Model_211$aic)

par(mar=c(3,3,3,0));par(mfrow=c(1,3))
ts.plot(Model_211$residuals)
Acf(Model_211$residuals,lag.max=40)
Pacf(Model_211$residuals,lag.max=40)
```




```r
#Now let's try ARIMA(1,1,2)
Model_112 <- Arima(deseasonal_residential_price,order=c(1,1,2),include.drift=TRUE)
print(Model_112)
```

```
## Series: deseasonal_residential_price
## ARIMA(1,1,2) with drift
##
## Coefficients:
##          ar1     ma1      ma2   drift
##       -0.543  0.3423  -0.1964  0.0214
## s.e.   0.261  0.2596   0.0687  0.0070
##
## sigma^2 estimated as 0.0212:  log likelihood=122.89
## AIC=-235.78   AICc=-235.52   BIC=-218.42
```

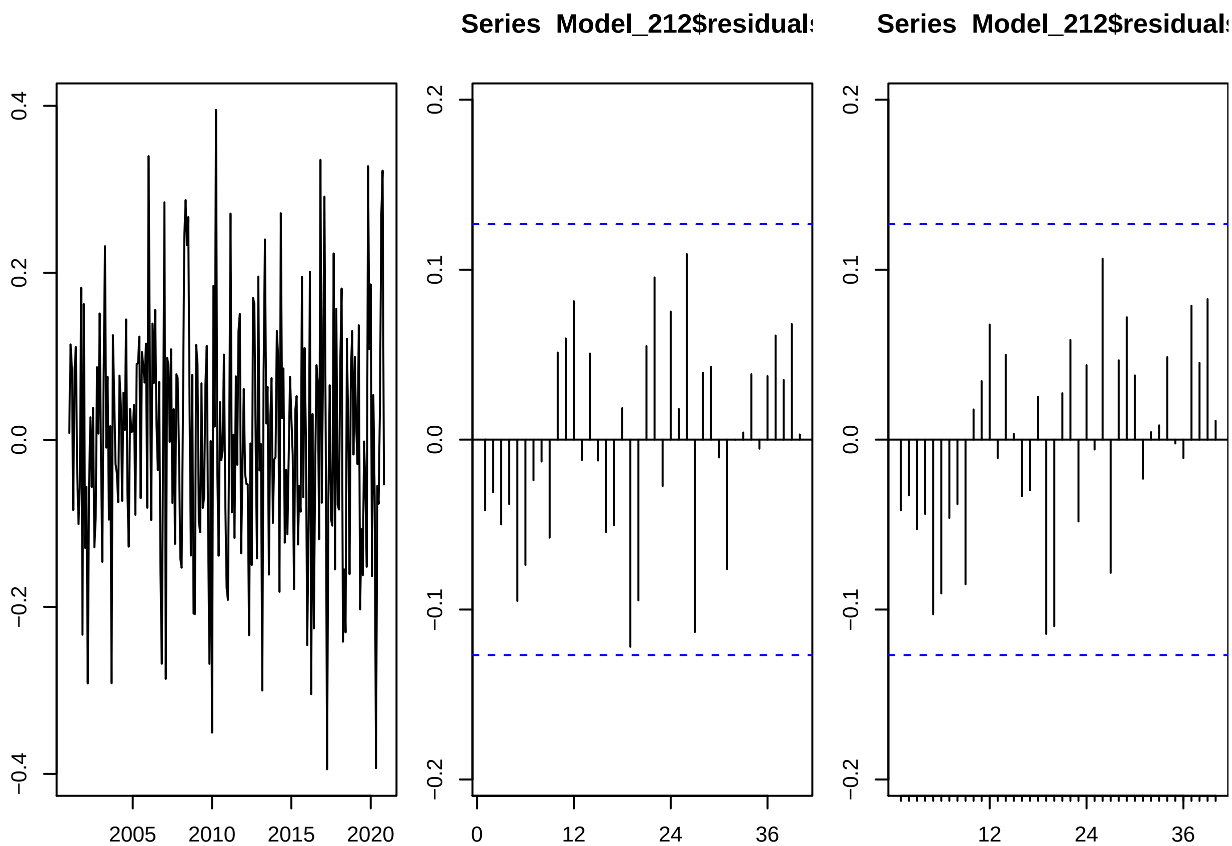
```r
compare_aic <- data.frame(compare_aic,Model_112$aic)

par(mar=c(3,3,3,0));par(mfrow=c(1,3))
ts.plot(Model_112$residuals)
Acf(Model_112$residuals,lag.max=40)
Pacf(Model_112$residuals,lag.max=40)
```




```r
#Now let's try ARIMA(2,1,2)
Model_212 <- Arima(deseasonal_residential_price,order=c(2,1,2),include.drift=TRUE)
print(Model_212)
```

```
## Series: deseasonal_residential_price
## ARIMA(2,1,2) with drift
##
## Coefficients:
##           ar1      ar2     ma1     ma2   drift
##       -0.9488  -0.8484  0.8040  0.7078  0.0217
## s.e.   0.0867   0.1001  0.1206  0.1391  0.0083
##
## sigma^2 estimated as 0.02052:  log likelihood=127.14
## AIC=-242.29   AICc=-241.93   BIC=-221.46
```

```r
compare_aic <- data.frame(compare_aic,Model_212$aic)

par(mar=c(3,3,3,0));par(mfrow=c(1,3))
ts.plot(Model_212$residuals)
Acf(Model_212$residuals,lag.max=40)
Pacf(Model_212$residuals,lag.max=40)
```




```r
print(compare_aic)
```

```
##   Model_111.aic Model_011.aic Model_211.aic Model_112.aic Model_212.aic
## 1     -237.2213     -237.6905     -237.6465     -235.7771     -242.2898
```

Automatically fitting ARIMA

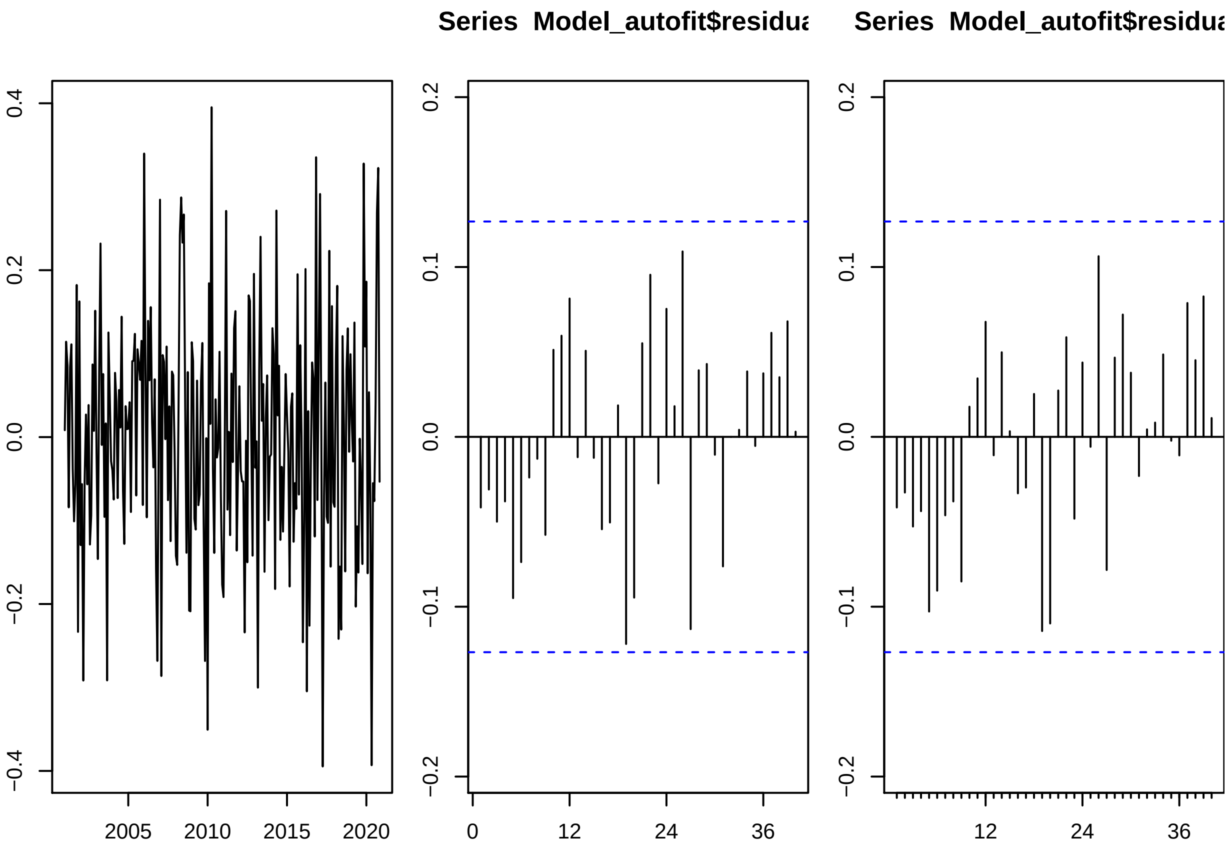
Now that you have played with different order, let's try the auto.arima() function from the base package *stats*. The best fit for this time series is a ARIMA(2,1,2) with drift.

```r
Model_autofit <- auto.arima(deseasonal_residential_price,max.D=0,max.P = 0,max.Q=0)
print(Model_autofit)
```

```
## Series: deseasonal_residential_price
## ARIMA(2,1,2) with drift
##
## Coefficients:
##           ar1      ar2     ma1     ma2   drift
##       -0.9488  -0.8484  0.8040  0.7078  0.0217
## s.e.   0.0867   0.1001  0.1206  0.1391  0.0083
##
## sigma^2 estimated as 0.02052:  log likelihood=127.14
## AIC=-242.29   AICc=-241.93   BIC=-221.46
```

```r
compare_aic <- cbind(compare_aic,Model_autofit$aic)

par(mar=c(3,3,3,0));par(mfrow=c(1,3))
ts.plot(Model_autofit$residuals)
Acf(Model_autofit$residuals,lag.max=40)
```

```r
Pacf(Model_autofit$residuals,lag.max=40)
```




```r
print(compare_aic)
```

```
##   Model_111.aic Model_011.aic Model_211.aic Model_112.aic Model_212.aic
## 1     -237.2213     -237.6905     -237.6465     -235.7771     -242.2898
##   Model_autofit$aic
## 1         -242.2898
```

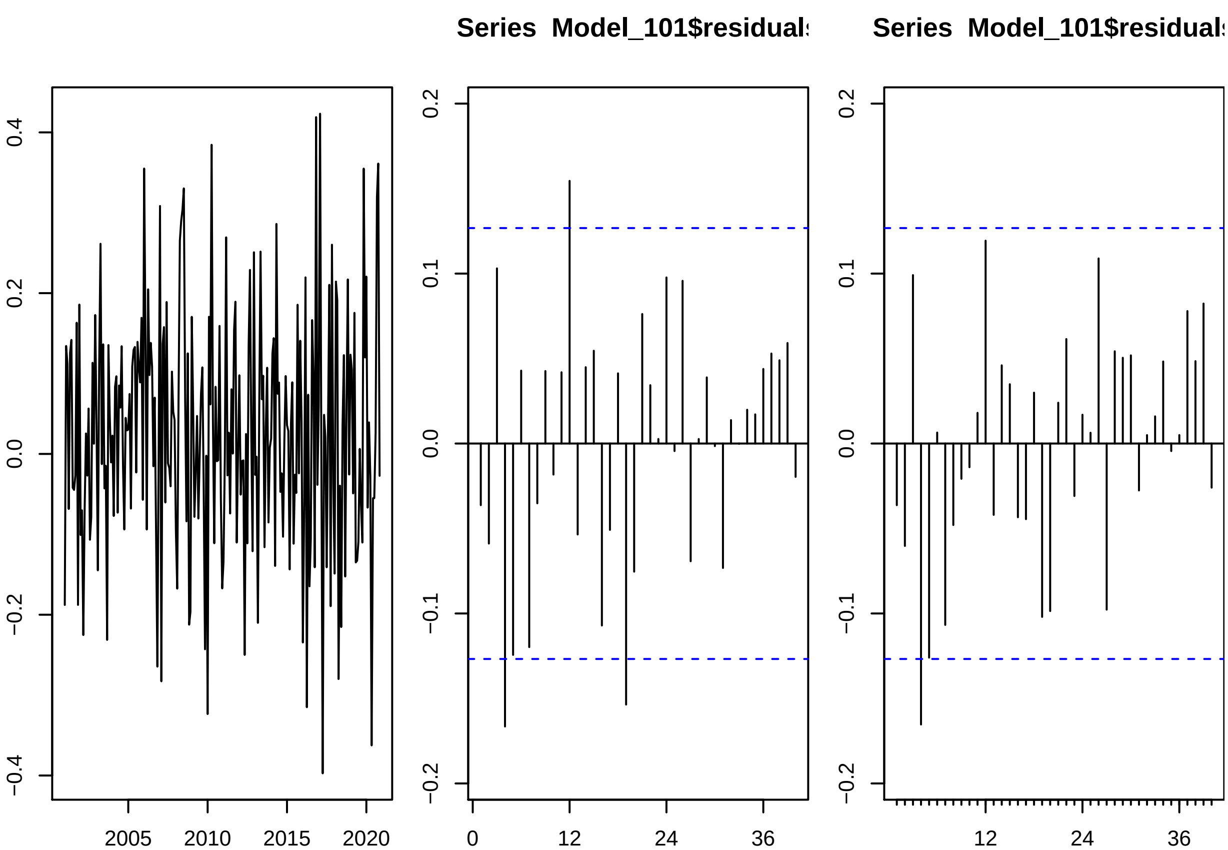
What happens if you don't differenciate?

If you don't differenciate the series, i.e., if you input the non-stationarity series, you should specify d = 1. Otherwise, Arima will be fitting a model to a non-stationary series. Note the difference between AIC for Model_101 and Model_101_diff

```r
Model_101 <- Arima(deseasonal_residential_price,order=c(1,0,1))
print(Model_101)
```

```
## Series: deseasonal_residential_price
## ARIMA(1,0,1) with non-zero mean
##
## Coefficients:
##          ar1      ma1     mean
##       0.9992  -0.1851  12.0816
## s.e.  0.0017   0.0634   3.8150
##
## sigma^2 estimated as 0.02201:  log likelihood=115.42
## AIC=-222.84   AICc=-222.67   BIC=-208.93
```

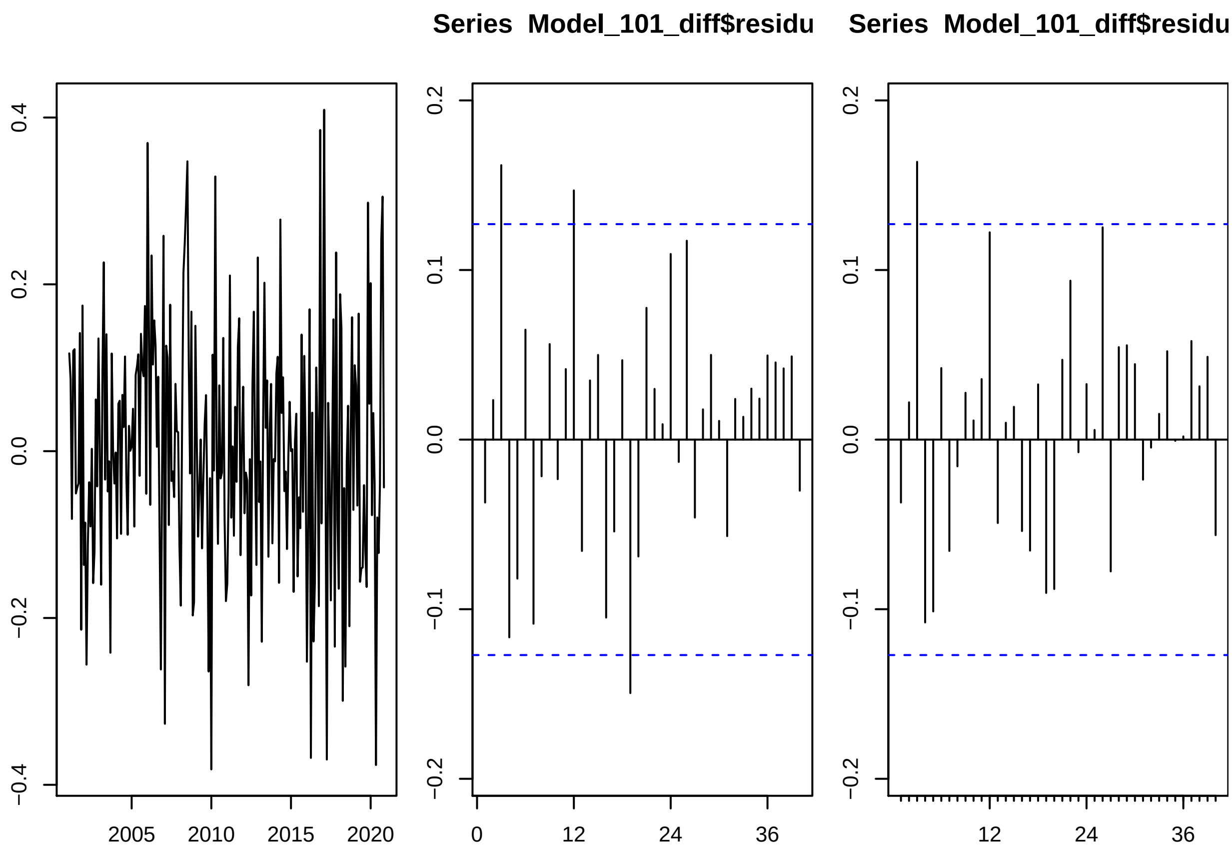
```r
compare_aic <- data.frame(compare_aic,Model_101$aic)

par(mar=c(3,3,3,0));par(mfrow=c(1,3))
ts.plot(Model_101$residuals)
Acf(Model_101$residuals,lag.max=40)
Pacf(Model_101$residuals,lag.max=40)
```




```r
#Remember the order d=1 will perform the differencing, so lets also try ARIMA(1,0,1) on the non-seasona
Model_101_diff=Arima(deseasonal_residential_price_diff,order=c(1,0,1))
print(Model_101_diff)
```

```
## Series: deseasonal_residential_price_diff
## ARIMA(1,0,1) with non-zero mean
##
## Coefficients:
##          ar1      ma1    mean
##       0.5798  -0.7702  0.0209
## s.e.  0.1673   0.1324  0.0052
##
## sigma^2 estimated as 0.02115:  log likelihood=122.61
## AIC=-237.22   AICc=-237.05   BIC=-223.33
```

```r
par(mar=c(3,3,3,0));par(mfrow=c(1,3))
ts.plot(Model_101_diff$residuals)
Acf(Model_101_diff$residuals,lag.max=40)
Pacf(Model_101_diff$residuals,lag.max=40)
```




```
compare_aic <- data.frame(compare_aic,Model_101_diff$aic)

print(compare_aic)
```

```
##   Model_111.aic Model_011.aic Model_211.aic Model_112.aic Model_212.aic
## 1     -237.2213     -237.6905     -237.6465     -235.7771     -242.2898
##   Model_autofit.aic Model_101.aic Model_101_diff.aic
## 1         -242.2898     -222.8398          -237.2214
```

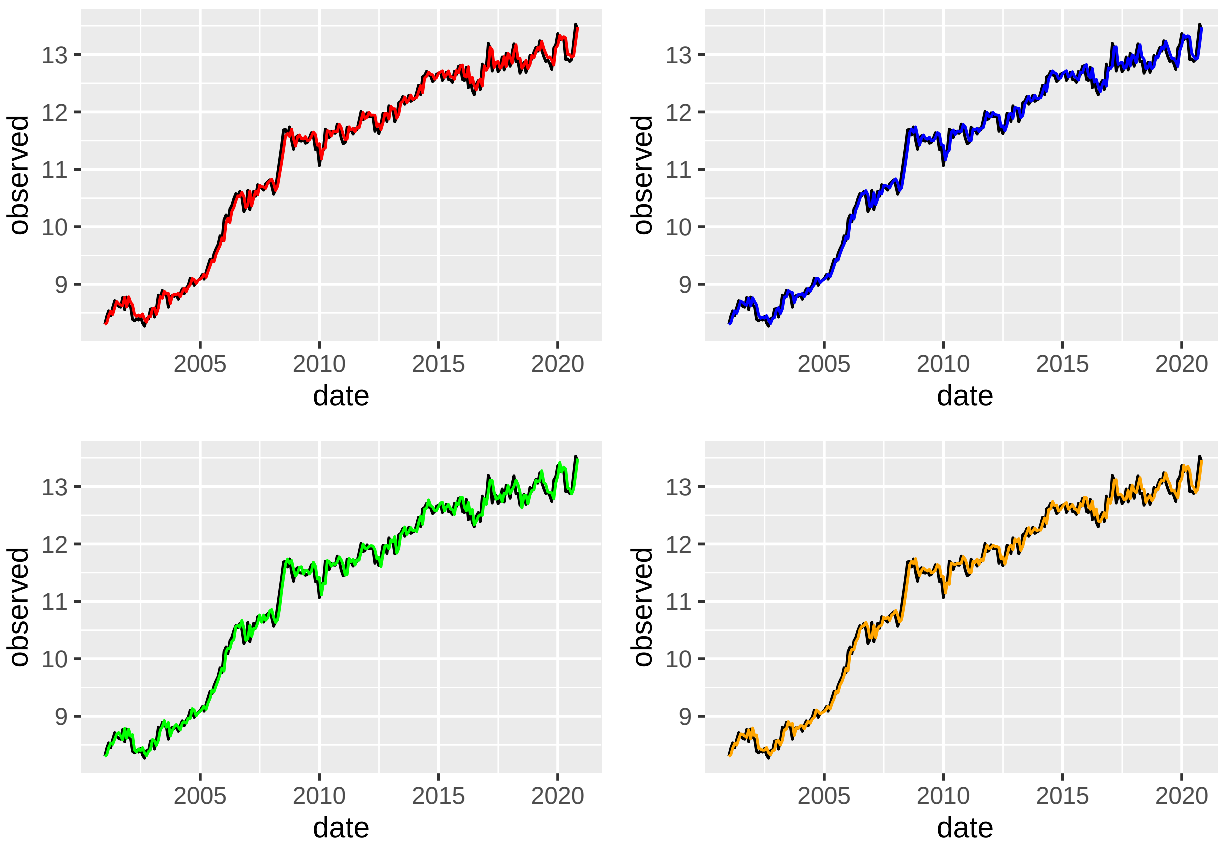
Note that AIC is worse for the ARIMA(1,0,1) with the non-diferrenced series.

Comparing models

One way of checking goodness of fit is by plotting observed versus fitted value over time. Here we will do it for some of the models we created only. But it can be generalized for all of them.

```
df_models <- data.frame(
  date = electricity_price_processed$Month,
  observed = as.numeric(deseasonal_residential_price),
  ARIMA_111 = as.numeric(Model_111$fitted),
  ARIMA_011 = as.numeric(Model_011$fitted),
  ARIMA_auto = as.numeric(Model_autofit$fitted),
  ARIMA_211 = as.numeric(Model_211$fitted)
)

Plot1 <-
ggplot(df_models) +
  geom_line(aes(x=date,y=observed),color="black") +
```

```
  geom_line(aes(x=date,y=ARIMA_111),color="red")

Plot2 <-
ggplot(df_models) +
  geom_line(aes(x=date,y=observed),color="black") +
  geom_line(aes(x=date,y=ARIMA_011),color="blue")

Plot3 <-
ggplot(df_models) +
  geom_line(aes(x=date,y=observed),color="black") +
    geom_line(aes(x=date,y=ARIMA_auto),color="green")

Plot4 <-
ggplot(df_models) +
  geom_line(aes(x=date,y=observed),color="black") +
  geom_line(aes(x=date,y=ARIMA_211),color="orange")

cowplot::plot_grid(Plot1,Plot2,Plot3,Plot4,nrow=2)
```



This is still non-seasonal data. If you want to compare to original series, you need to add seasonal component back.