ENV 790.30 - Time Series Analysis for Energy Data | Spring 2021

Assignment 6 - Due date 03/26/21

Rajat Khandelwal

Directions

You should open the .rmd file corresponding to this assignment on RStudio. The file is available on our class repository on Github. And to do so you will need to fork our repository and link it to your RStudio.

Once you have the project open the first thing you will do is change "Student Name" on line 3 with your name. Then you will start working through the assignment by **creating code and output** that answer each question. Be sure to use this assignment document. Your report should contain the answer to each question and any plots/tables you obtained (when applicable).

When you have completed the assignment, **Knit** the text and code into a single PDF file. Rename the pdf file such that it includes your first and last name (e.g., "LuanaLima_TSA_A06_Sp21.Rmd"). Submit this pdf using Sakai.

Set up

```
#Load/install required package here
library(forecast)
library(tseries)
library(Kendall)
library(readxl)
library(dplyr)
library(lubridate)
library(ggplot2)
library(tidyverse)
library(readr)
```

Importing and processing the data set

Consider the data from the file "Net_generation_United_States_all_sectors_monthly.csv". The data corresponds to the monthly net generation from January 2001 to December 2020 by source and is provided by the US Energy Information and Administration. **You will work with the natural gas column only**.

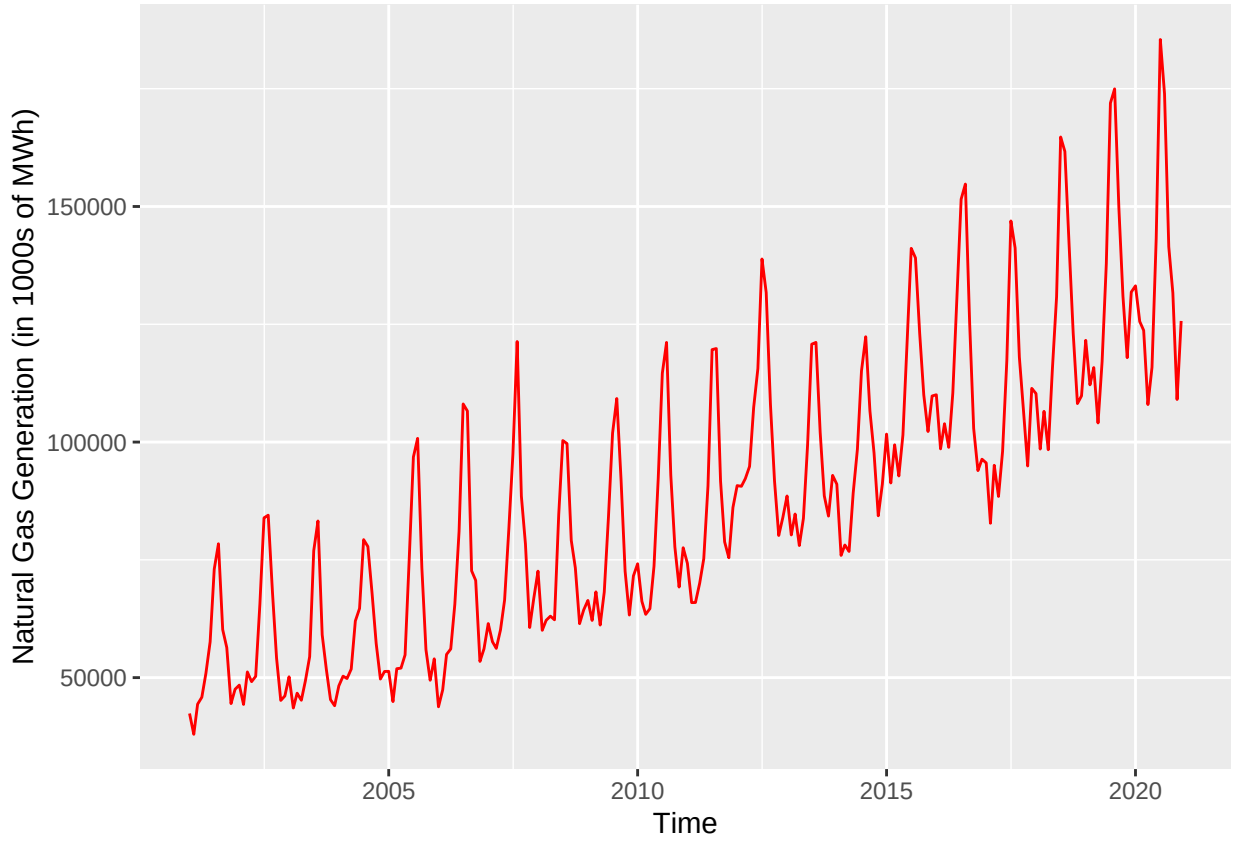
Packages needed for this assignment: "forecast","tseries". Do not forget to load them before running your script, since they are NOT default packages.\

Q1

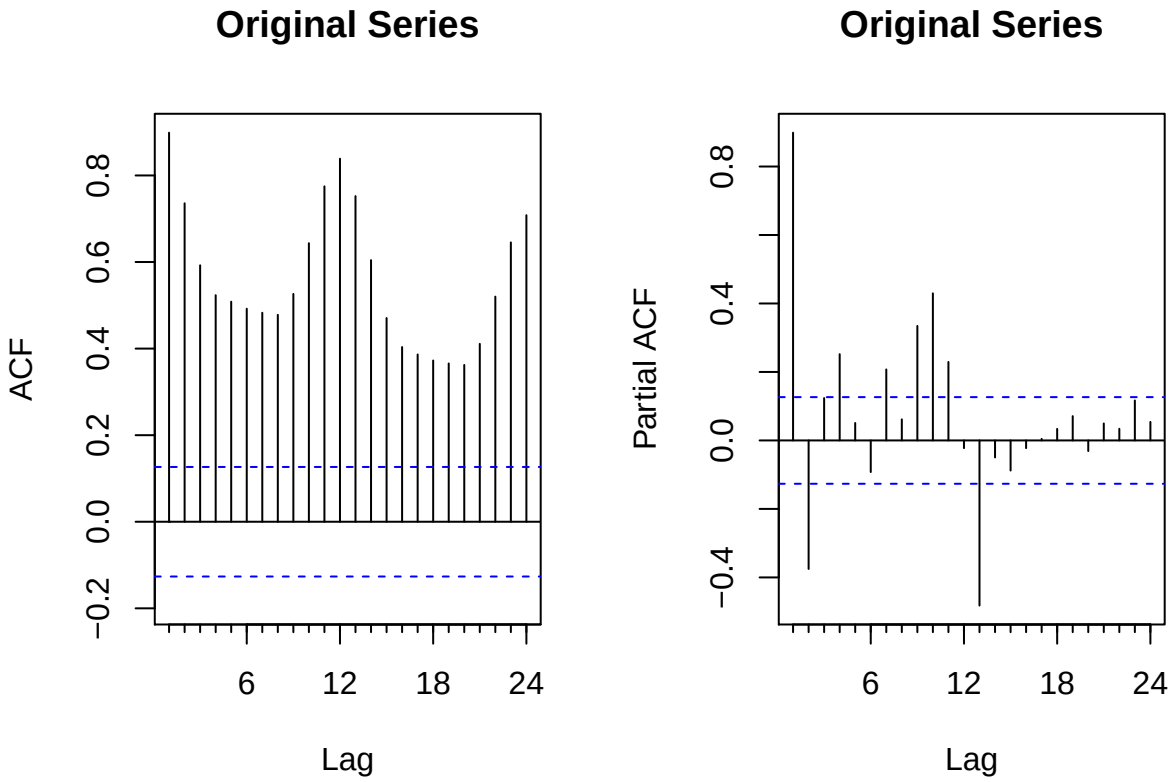
Import the csv file and create a time series object for natural gas. Make you sure you specify the **start=** and **frequency=** arguments. Plot the time series over time, ACF and PACF.

```
#Reading input CSV file and manipulating the data
setwd("/Users/rajatkhandelwal/Documents/GitHub/ENV790_30_TSA_S2021/Data")
data <- read_csv("Net_generation_United_States_all_sectors_monthly.csv", skip = 4)
```

```
data <- data %>% rename ("Natural Gas Generation (in 1000s of MWh)"
                         = "natural gas thousand megawatthours") %>% select(1,4)
data$Month <- my(data$Month)
data <- data %>% arrange(data, data$Month)

#Setting parameters for future use
ncols = ncol(data) - 1
nobs = nrow(data)

#Converting dataframe to time-series object
data_ts <- ts(data[2], start = c(year(data$Month[1]), month(data$Month[1])), end
              = c(year(data$Month[nobs]),month(data$Month[nobs])),frequency = 12)

#Plotting the time-series, it's ACF and PACF
ggplot(data, aes(data$Month,data_ts)) +
      geom_line(color = "red") +
      xlab("Time") +
      ylab("Natural Gas Generation (in 1000s of MWh)")
```



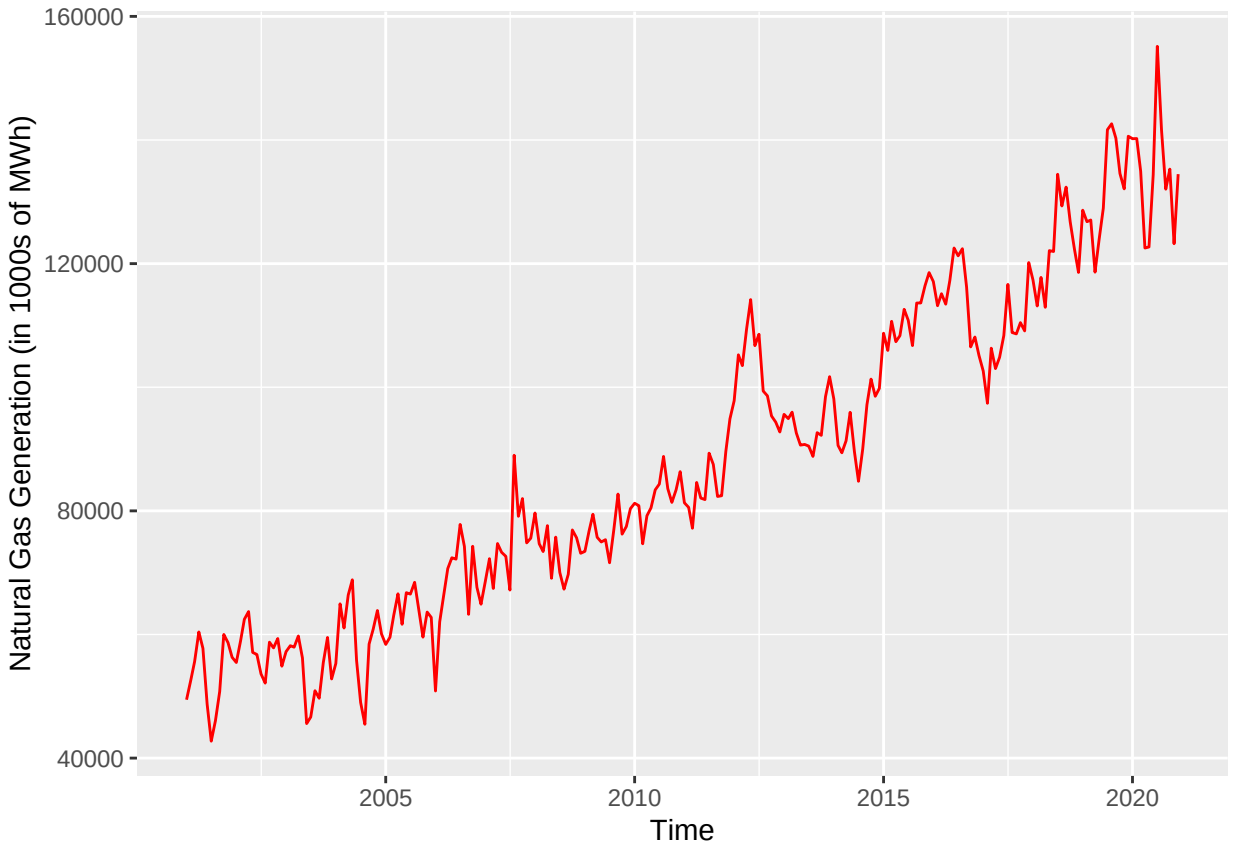

```
par(mfrow = c(1,2))
Acf(data_ts, main = "Original Series")
Pacf(data_ts, main = "Original Series")
```



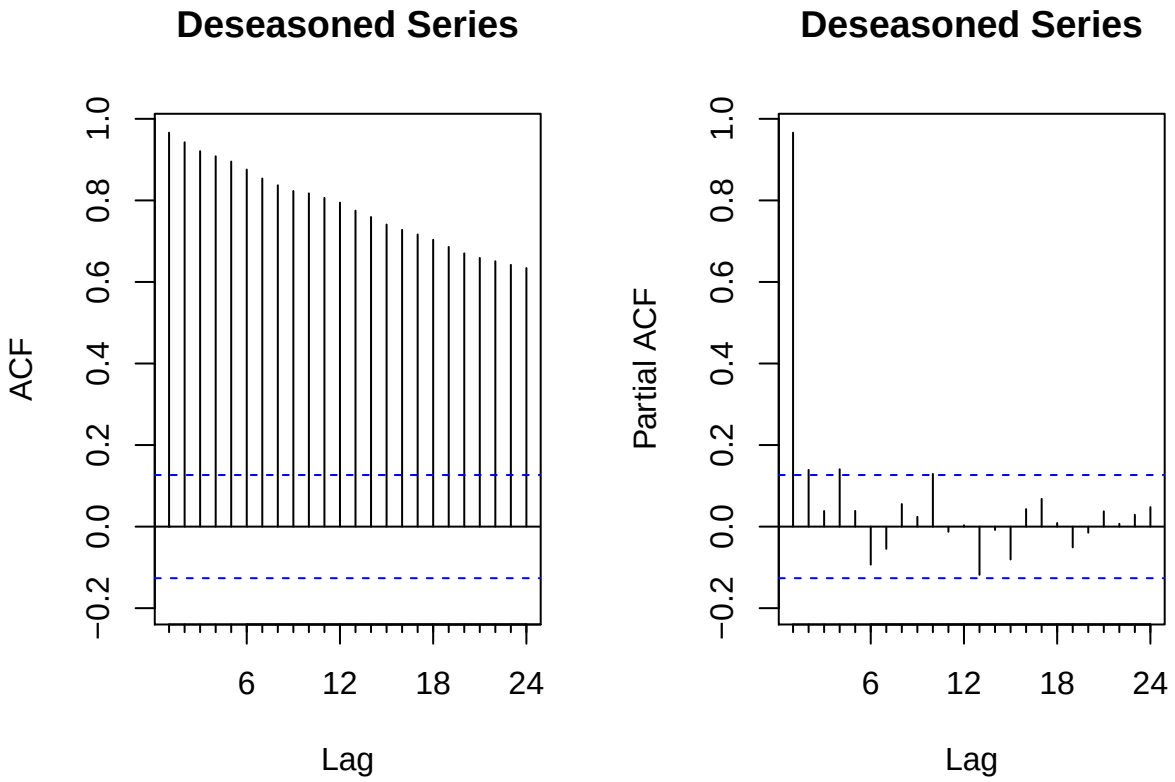
Q2

Using the *decompose*() or *stl*() and the *seasadj*() functions create a series without the seasonal component, i.e., a deseasonalized natural gas series. Plot the deseasonalized series over time and corresponding ACF and PACF. Compare with the plots obtained in Q1.

```
#Decomposing time-series using the default "additive" method
data_ts_decomp <- decompose(data_ts)

#Deseasoning the time-series using seasadj function
data_ts_deseasoned <- seasadj(data_ts_decomp)

#Plotting the deseasoned time-series, it's ACF and PACF
ggplot(data, aes(data$Month,data_ts_deseasoned)) +
       geom_line(color = "red") +
       xlab("Time") +
       ylab("Natural Gas Generation (in 1000s of MWh)")
```




```
par(mfrow = c(1,2))
Acf(data_ts_deseasoned, main = "Deseasoned Series")
Pacf(data_ts_deseasoned, main = "Deseasoned Series")
```



We observe peaks at lags of 12 and 24 in the ACF of the original series which are **absent** in the ACF of the deseasoned series. Further, a sharp negative peak at lag = 12 in the PACF of the original series is no longer observed in the deseasoned series. Lastly, the time-series plot of the deseasoned series is down-shifted relative to the original time-series plot **due to removal of seasonality**. This indicates the loss of seasonality for after using the decompose() and seasadj() functions.

Modeling the seasonally adjusted or deseasonalized series

Q3

Run the ADF test and Mann Kendall test on the deseasonalized data from Q2. Report and explain the results.

```
print("ADF Test")
```

```
## [1] "ADF Test"
```

```
print(adf.test(data_ts_deseasoned))
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  data_ts_deseasoned
## Dickey-Fuller = -4.0271, Lag order = 6, p-value = 0.01
## alternative hypothesis: stationary
```

```
print("Mann-Kendall test")
```

```
## [1] "Mann-Kendall test"
```

```
print(MannKendall(data_ts_deseasoned))
```

```
## tau = 0.843, 2-sided pvalue =< 2.22e-16
```

Since the p-value for for the ADF test is < 0.05, we **reject the null hypothesis** that the time-series has a unit root. Hence, we conclude that the time-series does not have a stochastic trend. Further, since the p-value of the Mann-Kendall test is < 0.05 hence we again **reject the null hypothesis** that there is no deterministic trend in the series.

Therefore, based on these results, we can state that the deseasonal time-series **does not have a stochastic trend but it does have a deterministic trend.**

Q4

Using the plots from Q2 and test results from Q3 identify the ARIMA model parameters p, d and q. Note that in this case because you removed the seasonal component prior to identifying the model you don't need to worry about seasonal component. Clearly state your criteria and any additional function in R you might use. DO NOT use the *auto.arima*() function. You will be evaluated on ability to can read the plots and interpret the test results.

Answer: On observing the ACF of the deseasoned series, we observe a **slow decay in the ACF value with increasing lag**, which is indicative of an Auto-Regressive (AR) process. The PACF shows a **cut-off value of 2** which leads us to conclude that these two plots together indicate an AR(2) process, hence $p = 2$. Further, there is no indication of a Moving Average (MA) process, therefore $q = 0$. Since the ACF has high positive values even upto lag = 24, it indicates that the series **likely needs some differencing**.This was also reflected by the Mann-Kendall & ADF tests done previously, which led us to conclude that the series has a deterministic trend. To remove this trend, we will have to difference the series. To calculate the number of differencing cycles required, we can use the **ndiffs()** function in R.

```
ndiff <- ndiffs(data_ts_deseasoned)
print(ndiff)
```

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

Therefore, since ndiff = 1, we need to difference the series once to remove the trend component.

$p = 2$
$d = 1$
$q = 0$

Q5

Use *Arima*() from package "forecast" to fit an ARIMA model to your series considering the order estimated in Q4. Should you allow for constants in the model, i.e., $include.mean = TRUE$ or $include.drift = TRUE$. **Print the coefficients** in your report. Hint: use the *cat*() function to print.

```
#Fitting an ARIMA model for this time-series. Including drift since d + D < 2.
#Since d = 1, include.mean = FALSE.
data_ts_deseasoned_arima <- Arima(data_ts_deseasoned, order = c(2,1,0), seasonal
                                  = c(0,0,0), include.drift = TRUE, include.mean = FALSE)
print(data_ts_deseasoned_arima)
```

```
## Series: data_ts_deseasoned
## ARIMA(2,1,0) with drift
##
## Coefficients:
##           ar1      ar2     drift
##       -0.1647  -0.1283  346.8289
## s.e.   0.0645   0.0650  272.1672
##
## sigma^2 estimated as 29891418:  log likelihood=-2394.61
## AIC=4797.21   AICc=4797.39   BIC=4811.12
```

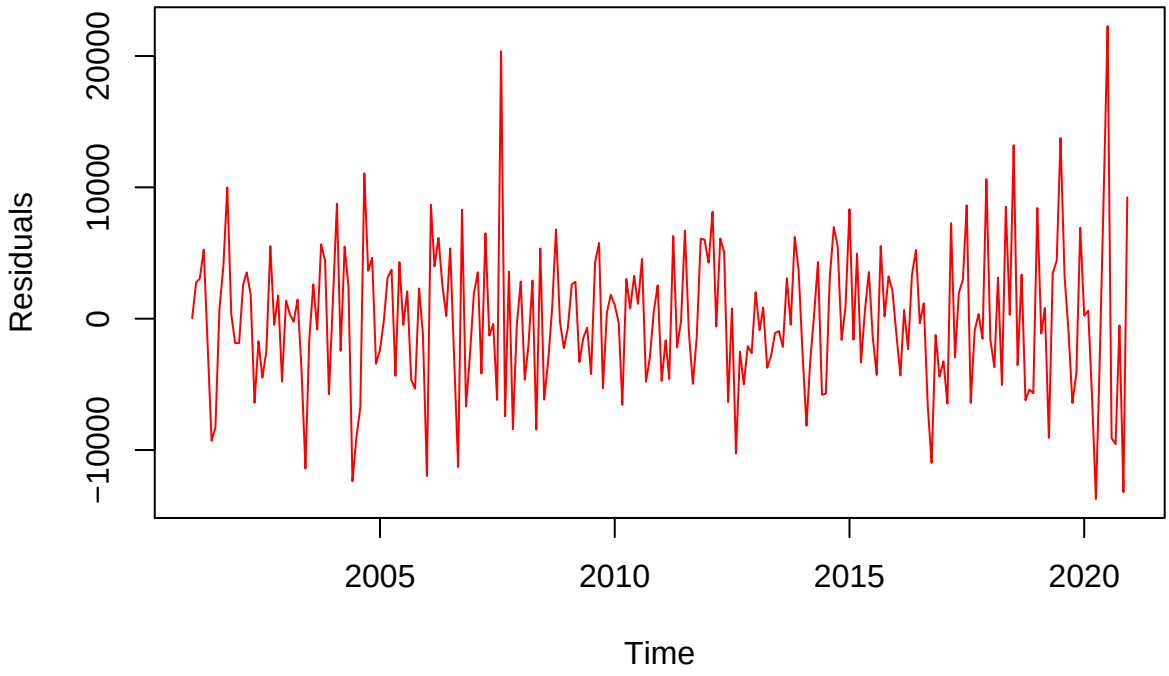
```
cat(" ar1", data_ts_deseasoned_arima[[1]][1], "\n", "ar2" , data_ts_deseasoned_arima[[1]][2],
    "\n", "drift", data_ts_deseasoned_arima[[1]][3], "\n")
```

```
##  ar1 -0.1647241
##  ar2 -0.1283009
##  drift 346.8289
```

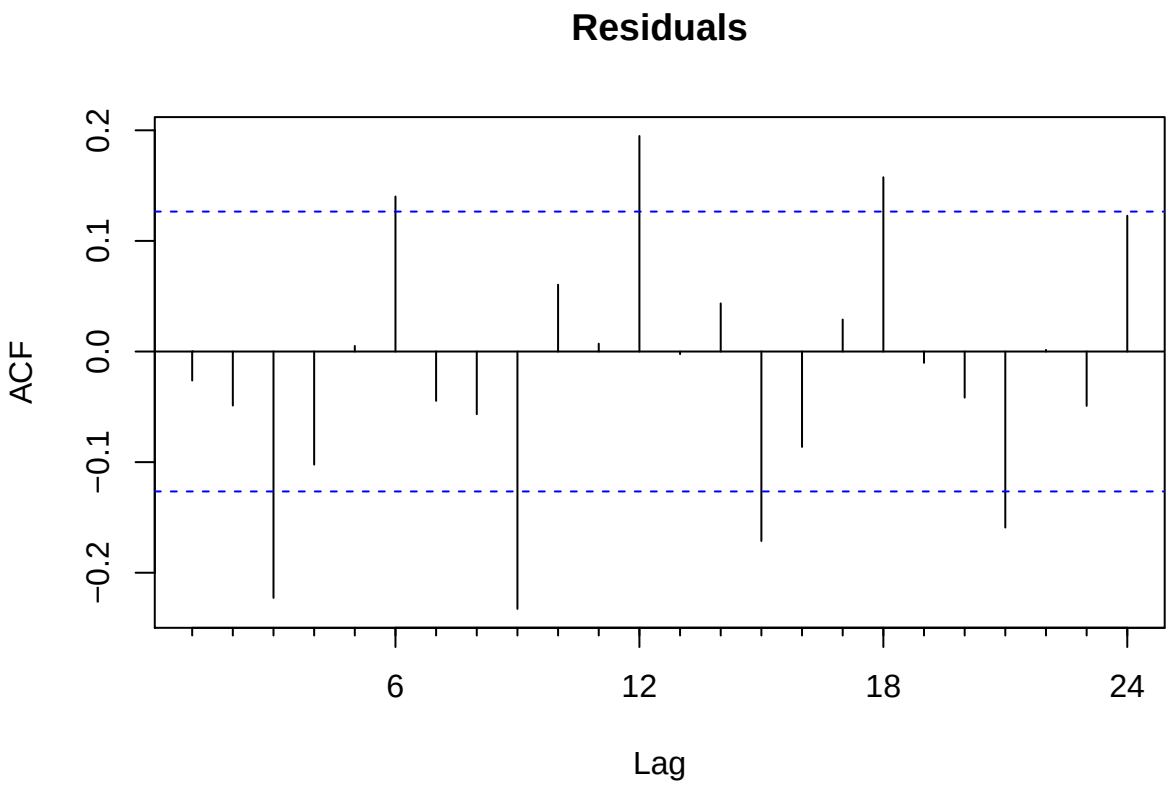
Q6

Now plot the residuals of the ARIMA fit from Q5 along with residuals ACF and PACF on the same window. You may use the *checkresiduals*() function to automatically generate the three plots. Do the residual series look like a white noise series? Why?

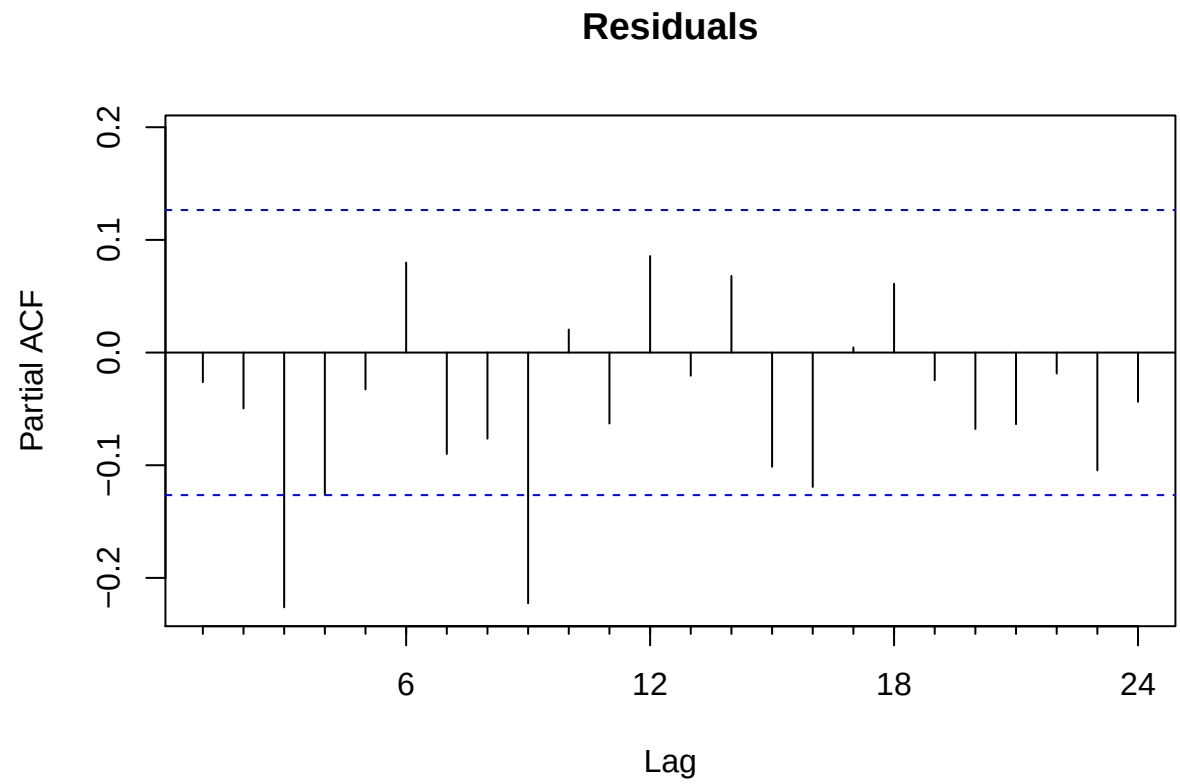
```
plot(x = data$Month, y = data_ts_deseasoned_arima[[8]], type = "l", col = "red",
     ylab = "Residuals", xlab = "Time")
```



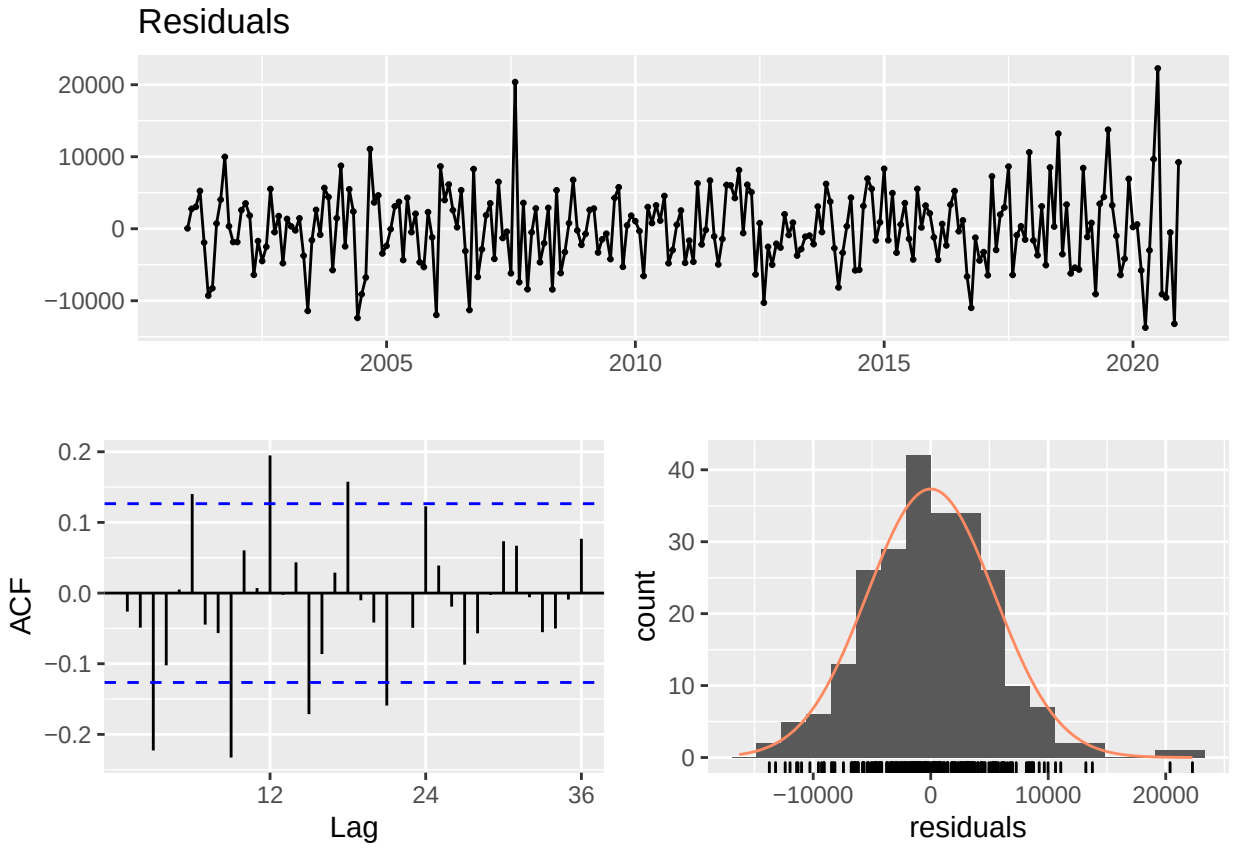

```
Acf(data_ts_deseasoned_arima[[8]], main = "Residuals")
```




```
Pacf(data_ts_deseasoned_arima[[8]], main = "Residuals")
```




```
checkresiduals(data_ts_deseasoned_arima[[8]])
```




```
mean_residuals_deseasoned <- print(mean(data_ts_deseasoned_arima[[8]]))
```

```
## [1] 4.511745
```

The residual series is close to, **but is not a perfect white noise series**. A white noise series is definied as a time-series which has no autocorrelation. It is effectively a sequence of independent, identically distribubted (IID) random variables. An ideal white noise series will have it's mean = 0. **If a given time-series model identifies the data trend and seasonality perfectly, the residual series should be a white noise series.**

For the fitted ARIMA model, we see that the residual series shows some autocorrelation (refer: ACF & PACF) and the mean of the series is 4.511745. This shows that the given model fitting was not able to accurately estimate the data trend and seasonality due to which the residual series has non-zero correlation and mean. Therefore, the residual series is **not a white noise series.**However, it must be noted that due to the relatively small values of the mean and autocorrelation, the residual series is quite close to being a white noise series.

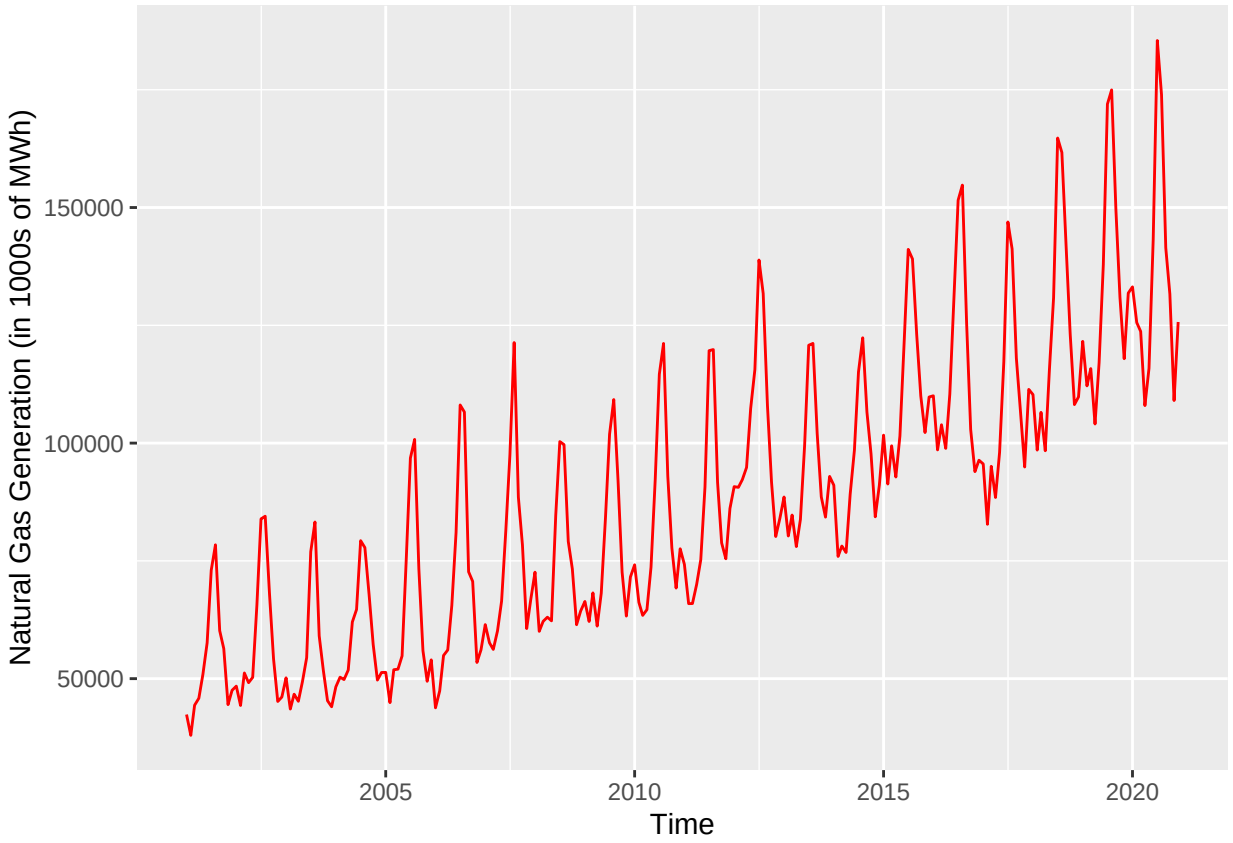
Modeling the original series (with seasonality)

Q7

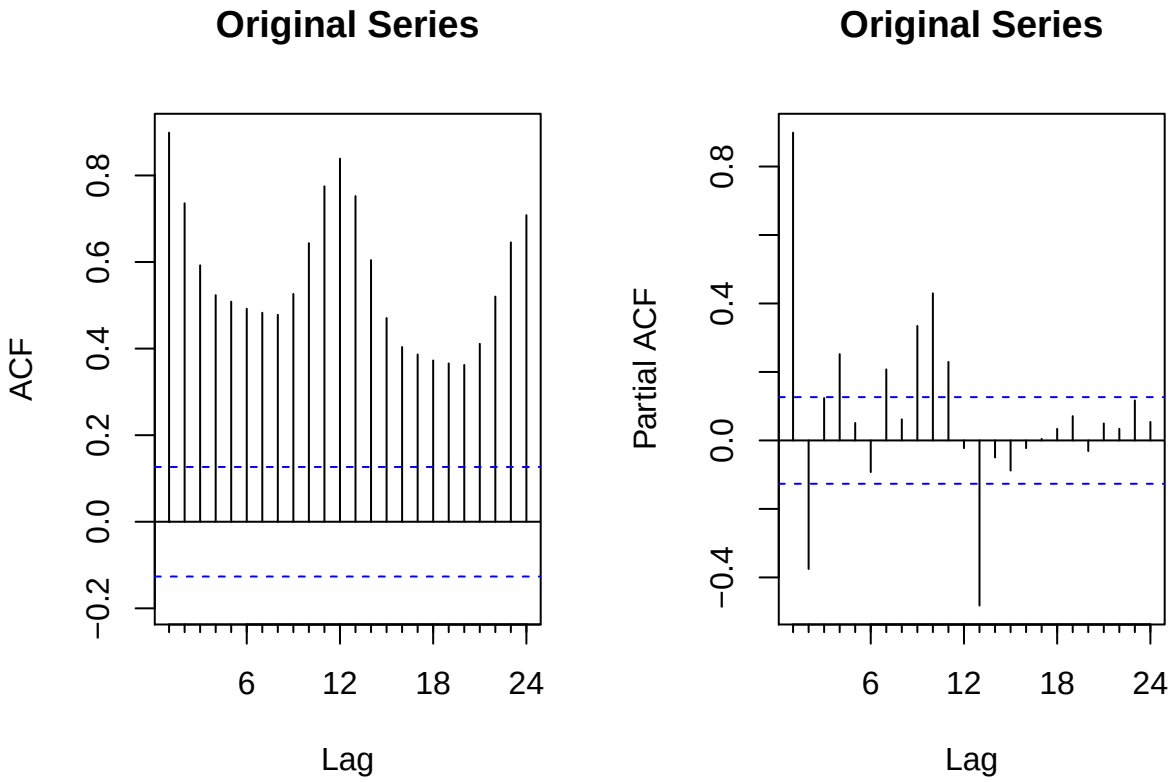
Repeat Q4-Q6 for the original series (the complete series that has the seasonal component). Note that when you model the seasonal series, you need to specify the seasonal part of the ARIMA model as well, i.e., P, D and Q.

```
#Plotting the original time-series, it's ACF and PACF
ggplot(data, aes(data$Month,data_ts)) +
      geom_line(color = "red") +
      xlab("Time") +
```

```
      ylab("Natural Gas Generation (in 1000s of MWh)")
```




```
par(mfrow = c(1,2))
Acf(data_ts, main = "Original Series")
Pacf(data_ts, main = "Original Series")
```



Looking at the ACF and PACF, we note that the seasonality is strong and stable over time. Further, the ACFs are have high values upto high number of lags. This indicates that the series probably **needs to be differenced seasonally**. To find out the number of seasonal differencing cycles required, we use the nsdiffs() function.

```
ndiff_seasonal <- nsdiffs(data_ts)
print(ndiff_seasonal)
```

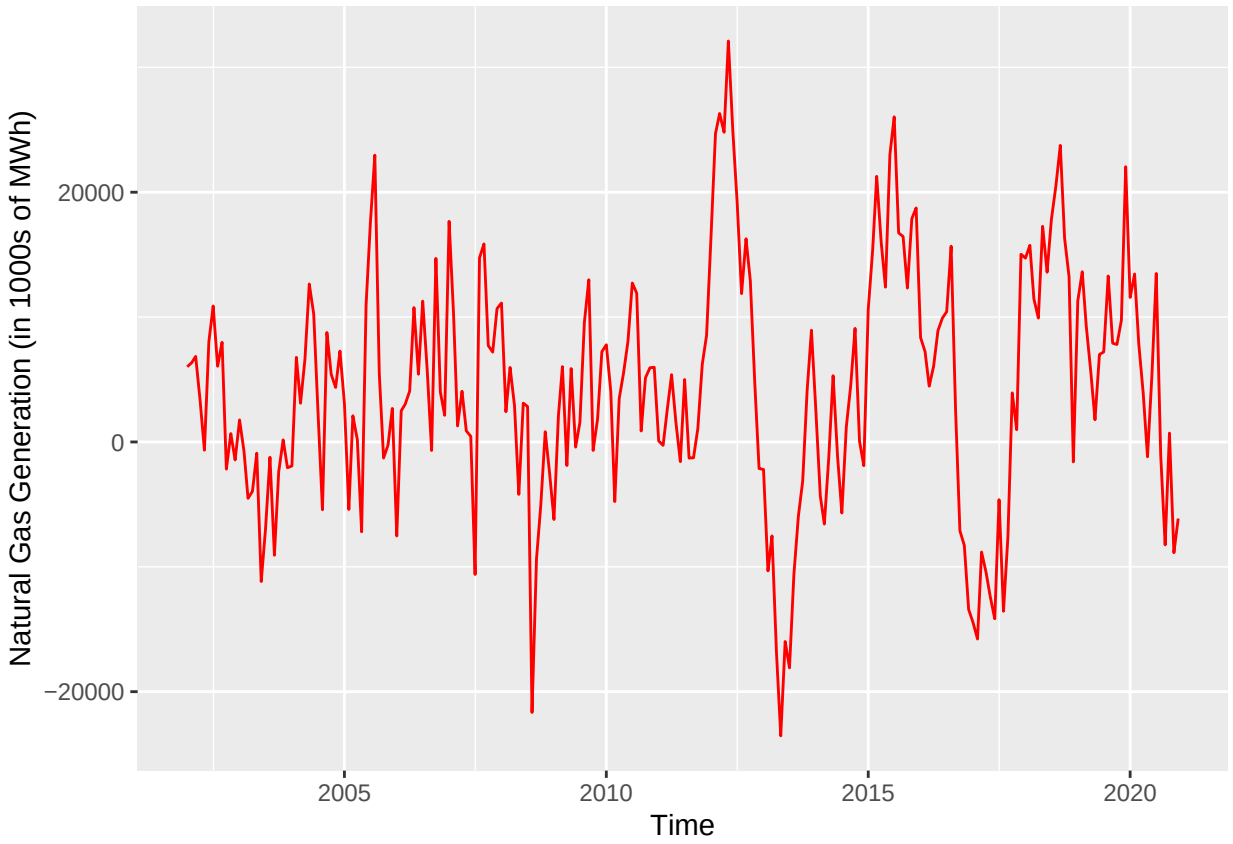
```
## [1] 1
```

We get ndiff_seasonal = 1 and hence D = 1, i.e. we will seasonally difference the series once.

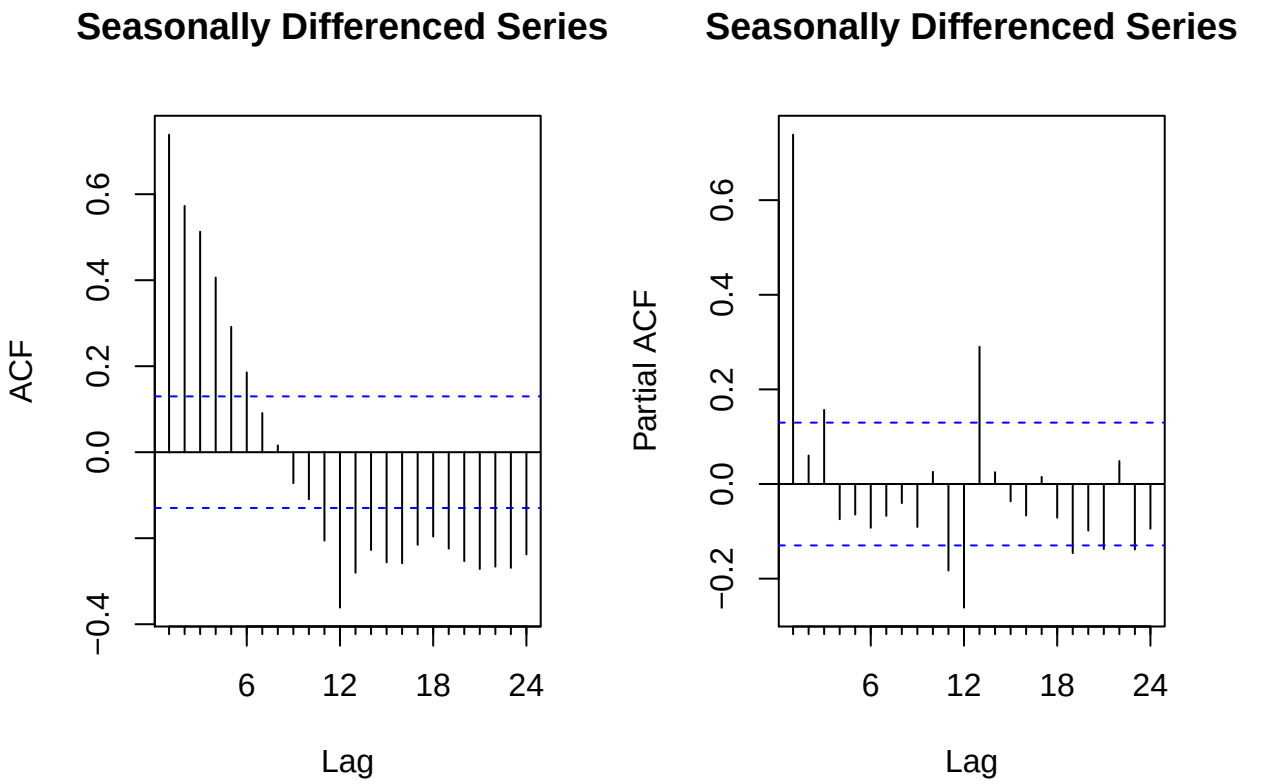
```
data_ts_diff <- diff(data_ts, lag = 12, difference = 1)
```

Now, we will plot the seasonally differenced time-series, it's ACF and PACF to estimate values of P & Q.

```
ggplot(data[13:240,], aes(data$Month[13:240],data_ts_diff)) +
    geom_line(color = "red") +
    xlab("Time") +
    ylab("Natural Gas Generation (in 1000s of MWh)")
```




```
par(mfrow = c(1,2))
Acf(data_ts_diff, main = "Seasonally Differenced Series ")
Pacf(data_ts_diff, main = "Seasonally Differenced Series")
```



Answer: On observing the ACF of the original series, we observe negative spikes in the ACF value at lags pf 12 and 24, which is indicative of an **Seasonal Moving Average (SMA)** process. The PACF shows a sharp neagtive peak at lag = 12 which leads us to conclude that these two plots together indicate an SMA(1) process, hence $Q = 1$. Further, there is no indication of a Seasonal Auto-Regressive (SAR) process, therefore $P = 0$. We already know that $D = 1$.

Therefore, in summary:
$P = 0$
$D = 1$
$Q = 1$

Using the above information, we use the Arima() function to fit an ARIMA model with the estimated values of (p,d,q) and (P, D, Q). **Since d + D = 2, we will not include drift in our model.**

```
data_ts_arima <- Arima(data_ts, order = c(2,1,0), seasonal = c(0,1,1))
print(data_ts_arima)
```

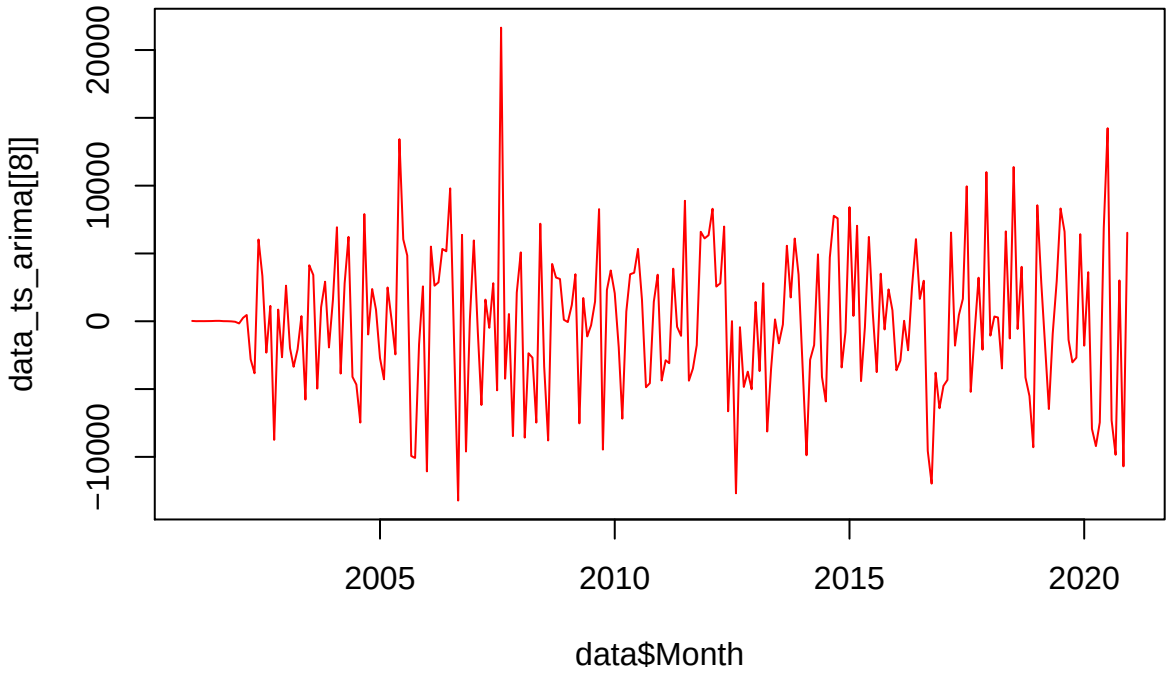
```
## Series: data_ts
## ARIMA(2,1,0)(0,1,1)[12]
##
## Coefficients:
##           ar1      ar2     sma1
##       -0.2100  -0.1645  -0.6741
## s.e.   0.0657   0.0663   0.0583
##
## sigma^2 estimated as 30007100:  log likelihood=-2278.39
## AIC=4564.78   AICc=4564.96   BIC=4578.48
```

```
cat(" ar1", data_ts_arima[[1]][1], "\n", "ar2" , data_ts_arima[[1]][2],
    "\n", "sma1", data_ts_arima[[1]][3], "\n")
```

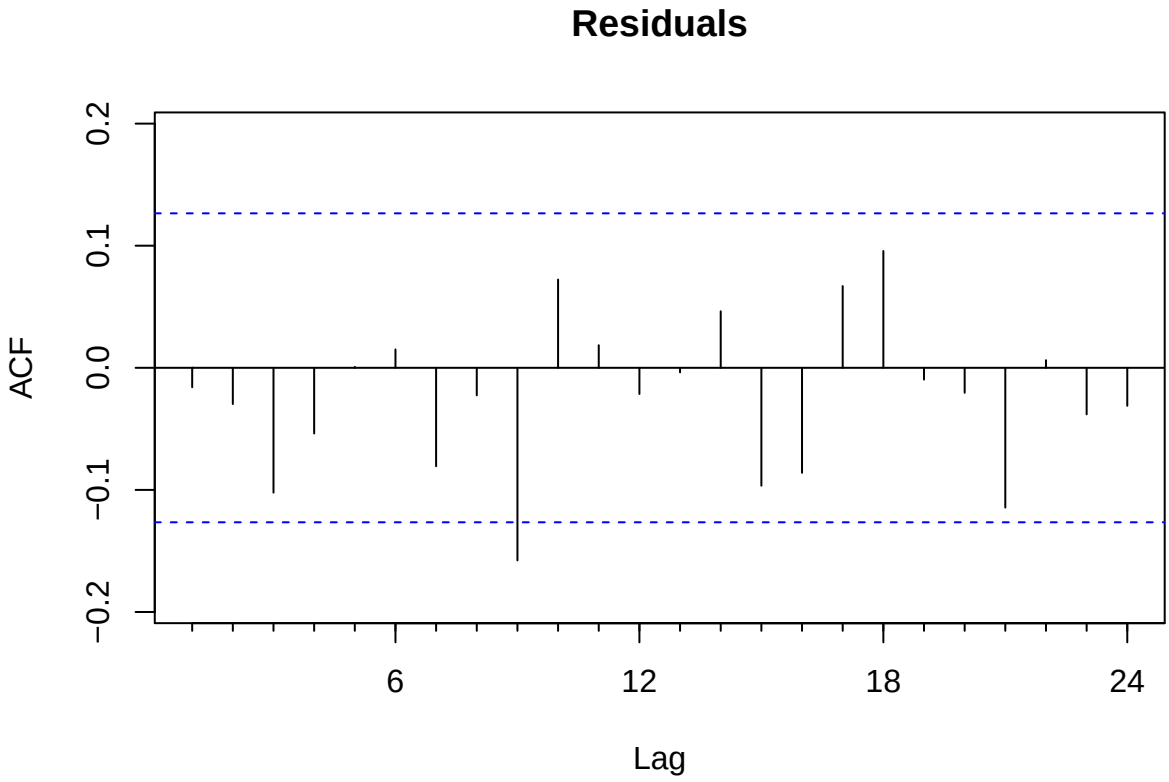
```
##  ar1 -0.2100211
##  ar2 -0.1645098
##  sma1 -0.6740642
```

Checking residuals.

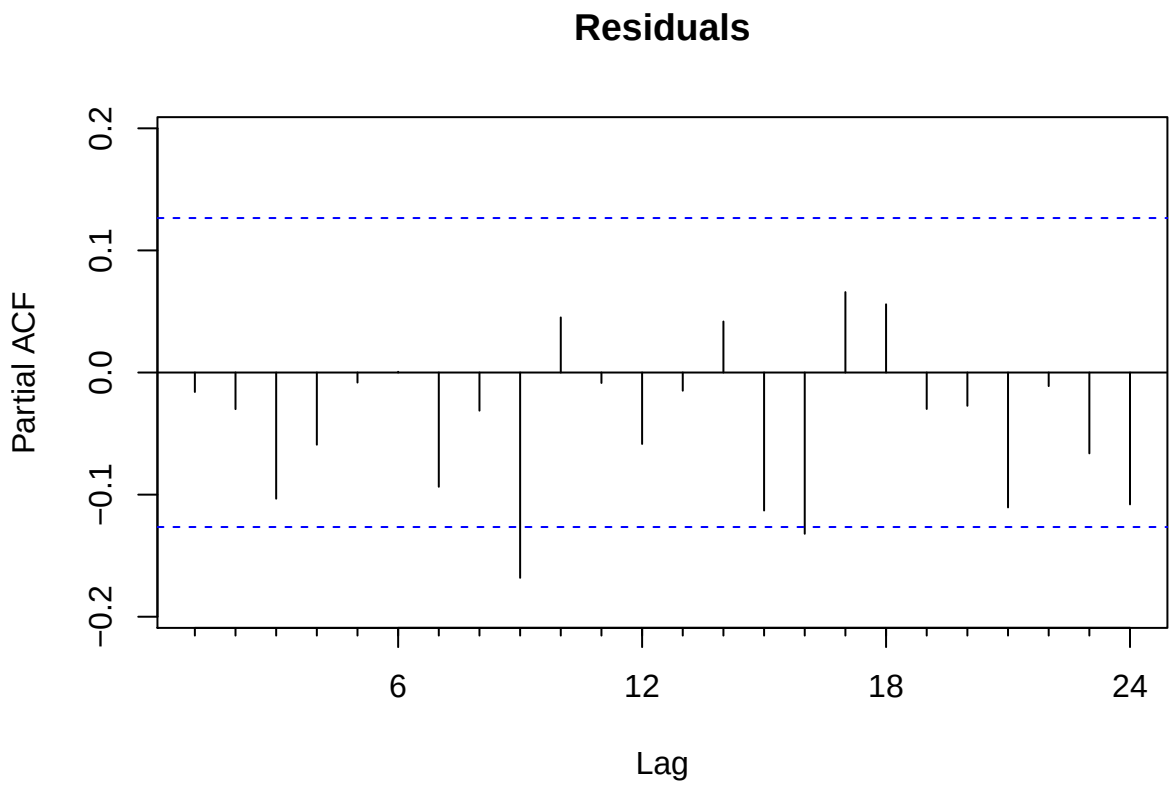
```
plot(x = data$Month, y = data_ts_arima[[8]], type = "l", col = "red")
```



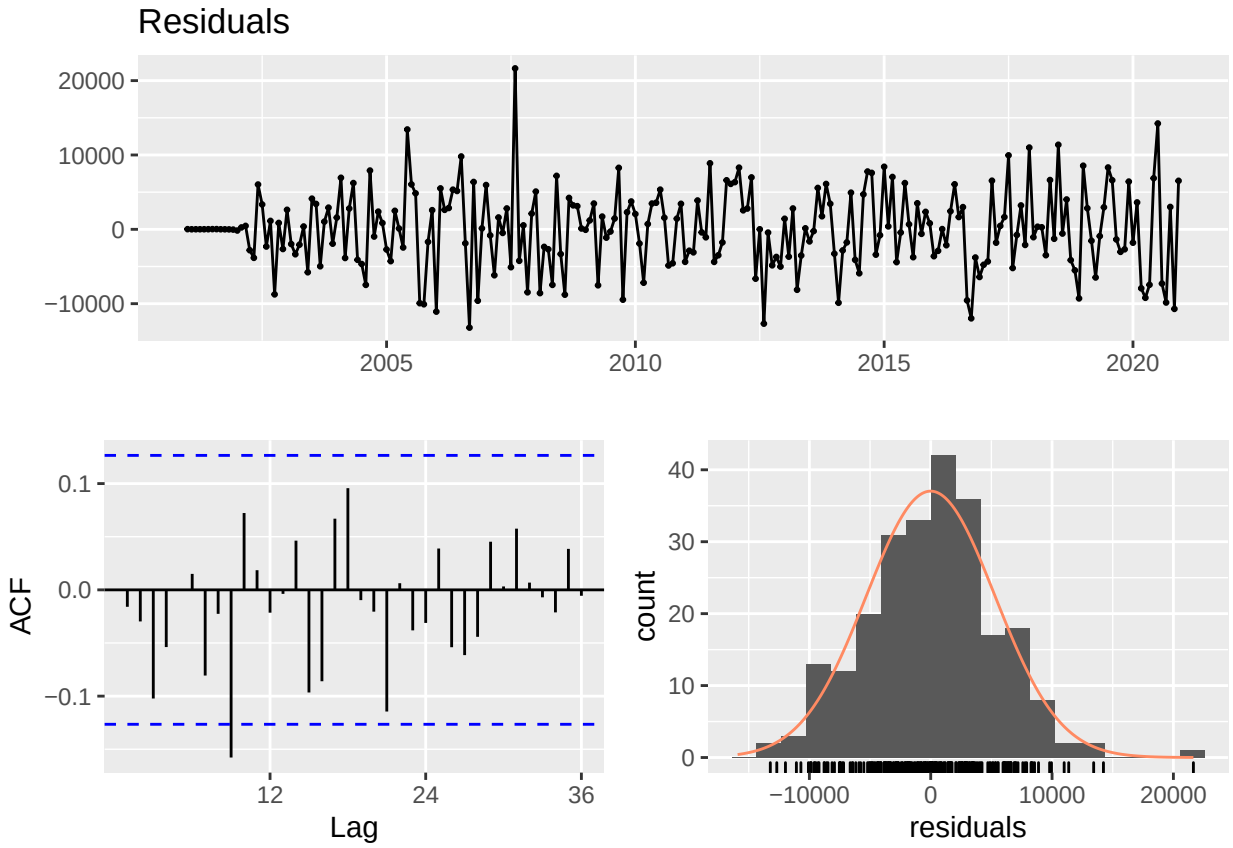

```
Acf(data_ts_arima[[8]], main = "Residuals")
```



```
Pacf(data_ts_arima[[8]], main = "Residuals")
```




```
checkresiduals(data_ts_arima[[8]])
```




```
mean_residuals_original<- print(mean(data_ts_arima[[8]]))
```

```
## [1] -5.434581
```

```
sd_residuals_original<- print(sd(data_ts_arima[[8]]))
```

```
## [1] 5303.186
```

The residual series is close to, **but is not a perfect white noise series**.

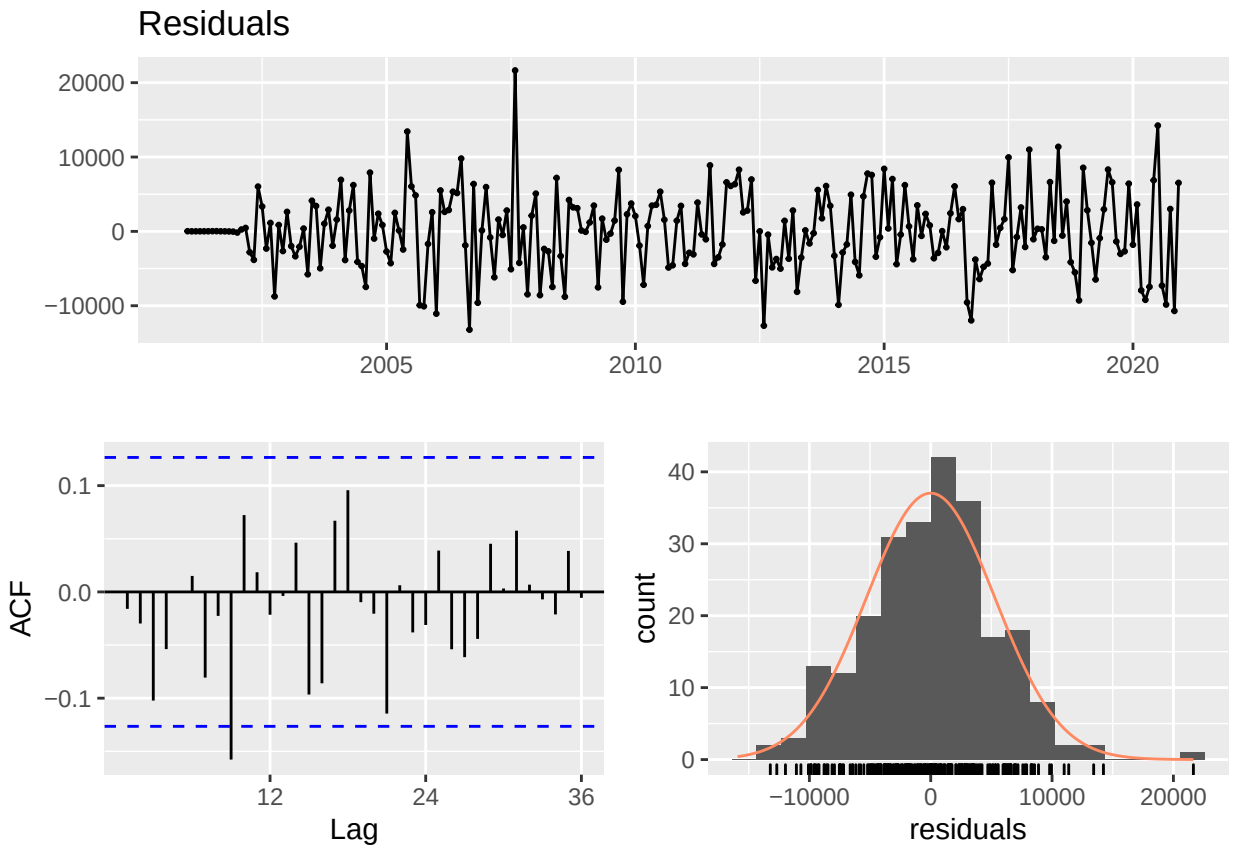
For the fitted ARIMA model, we see that the residual series shows some autocorrelation and the mean of the series is -5.434581. This shows that the given model fitting was not able to accurately estimate the data trend and seasonality due to which the residuals have non-zero correlation (Refer: ACF & PACF) and mean. Therefore, the residual series is **not a white noise series.**However, it must be noted that due to the relatively small values of the mean and autocorrelation, the residual series is quite close to being a white noise series.

Q8

Compare the residual series for Q7 and Q6. Can you tell which ARIMA model is better representing the Natural Gas Series? Is that a fair comparison? Explain your response.

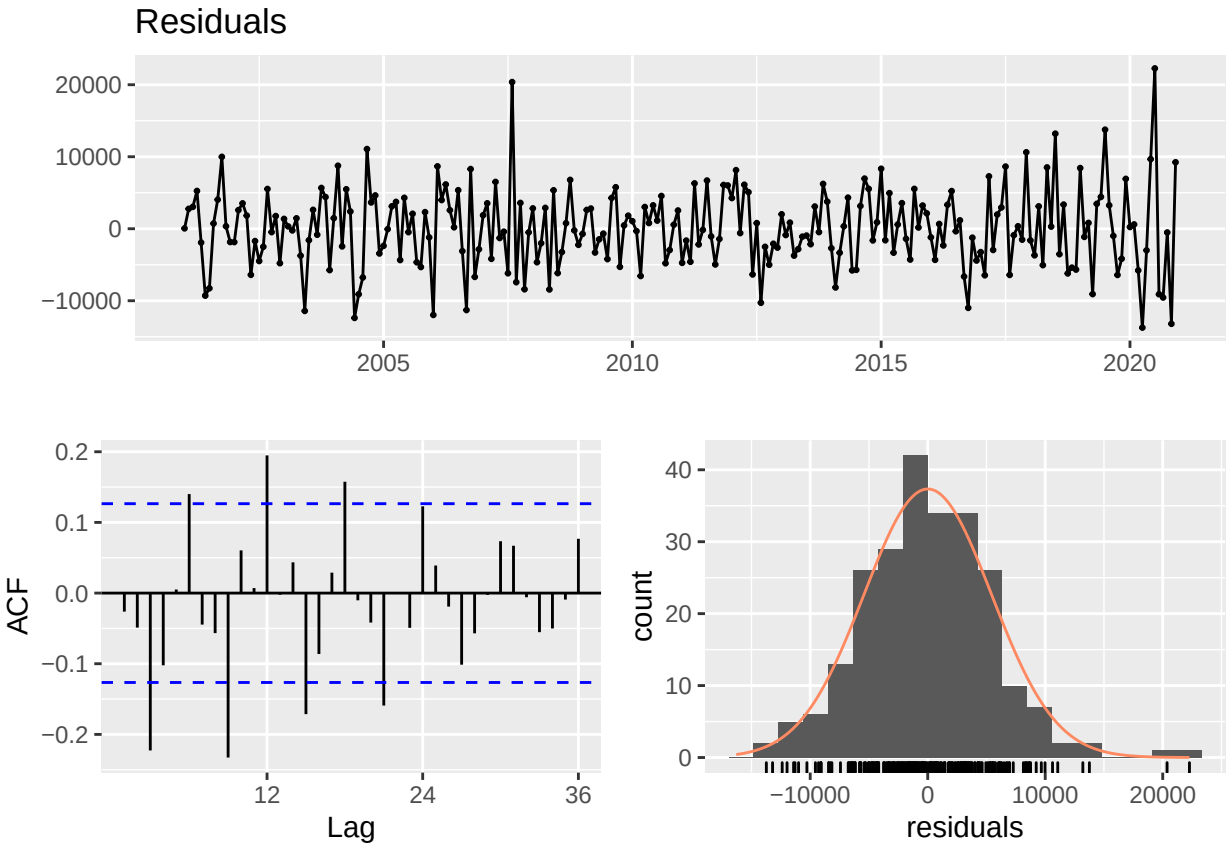
Residual analysis for original series fitted to ARIMA model (2,1,0)(0,1,1)

```
checkresiduals(data_ts_arima[[8]])
```



Residual analysis for deseasoned series fitted to ARIMA model (2,1,0)

```
checkresiduals(data_ts_deseasoned_arima[[8]])
```

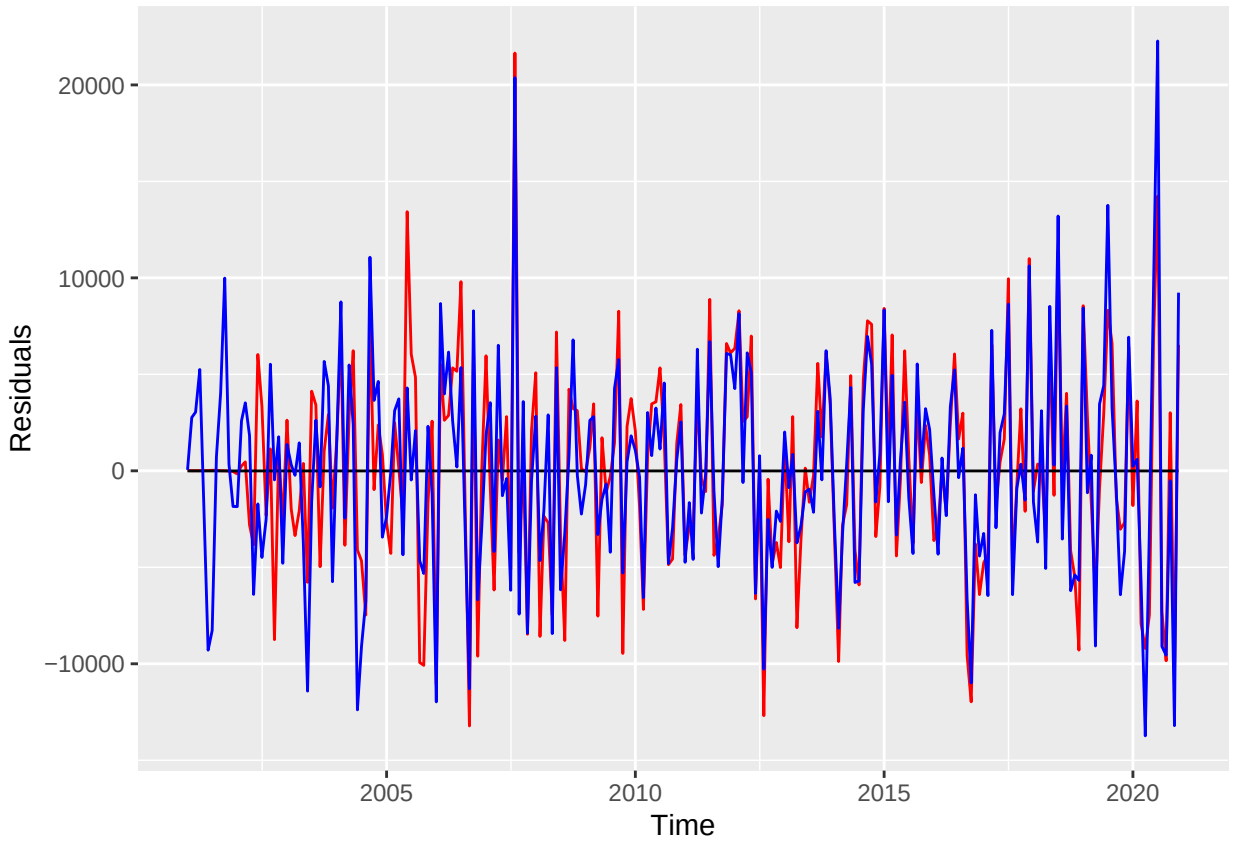


On comparing the two residual series, we observe that the that the residuals for the ARIMA model fitted to the original series show **lesser autocorrelation** as compared to that of the residuals for the ARIMA model fitted to the deseasoned series. Means for both the residual series are quite close to zero and hence we can't draw any conclusions for comparative purposes.

No, it is not a fair comparision as both the series being fitted are different. We can only fairly compare two models if the input time-series are the same.

Both residuals plotted together.
Red: Original series
Blue: Deseasoned series

```
ggplot(data, aes(data$Month, data_ts_arima[[8]])) +
  geom_line(color = "red") +
  geom_line(aes(y = mean_residuals_original)) +
  geom_line(aes(y = data_ts_deseasoned_arima[[8]]), color = "blue") +
  xlab("Time") +
  ylab("Residuals")
```



Checking your model with the auto.arima()

Please do not change your answers for Q4 and Q7 after you ran the *auto.arima*(). It is **ok** if you didn't get all orders correctly. You will not loose points for not having the correct orders. The intention of the assignment is to walk you to the process and help you figure out what you did wrong (if you did anything wrong!).

Q9

Use the *auto.arima*() command on the **deseasonalized series** to let R choose the model parameter for you. What's the order of the best ARIMA model? Does it match what you specified in Q4?

```
arima_fit_deseasoned <- auto.arima(data_ts_deseasoned, max.P = 0, max.D = 0,
                                   max.Q = 0)
print(arima_fit_deseasoned)
```

```
## Series: data_ts_deseasoned
## ARIMA(1,1,1) with drift
##
## Coefficients:
##          ar1      ma1     drift
##       0.7065  -0.9795  359.5052
## s.e.  0.0633   0.0326   29.5277
##
## sigma^2 estimated as 26980609:  log likelihood=-2383.11
## AIC=4774.21   AICc=4774.38   BIC=4788.12
```

The order of the best fit ARIMA model for the deseasoned series is (1,1,1) which does not match with our

estimate of (2,1,0).

Q10

Use the *auto.arima*() command on the **original series** to let R choose the model parameters for you. Does it match what you specified in Q7?

```
#Fitting model using auto.arima. We set allowdrift = TRUE since we are open to
#considering models with drift.
arima_fit_original <- auto.arima(data_ts, allowdrift = TRUE)
print(arima_fit_original)
```

```
## Series: data_ts
## ARIMA(1,0,0)(0,1,1)[12] with drift
##
## Coefficients:
##          ar1     sma1     drift
##       0.7416  -0.7026  358.7988
## s.e.  0.0442   0.0557   37.5875
##
## sigma^2 estimated as 27569124:  log likelihood=-2279.54
## AIC=4567.08   AICc=4567.26   BIC=4580.8
```

The order of the best fit ARIMA model for the deseasoned series is (1,0,0)(0,1,1) with drift which does not match with our estimate of (2,1,0),(0,1,1) (without drift, since d + D = 2).