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)
```

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

```
library(tseries)
library(Kendall)
library(readxl)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
```

```
##     date, intersect, setdiff, union
```

```
library(ggplot2)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v tibble  3.0.6     v purrr   0.3.4
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x lubridate::as.difftime() masks base::as.difftime()
## x lubridate::date()        masks base::date()
## x dplyr::filter()          masks stats::filter()
## x lubridate::intersect()   masks base::intersect()
## x dplyr::lag()             masks stats::lag()
## x lubridate::setdiff()     masks base::setdiff()
## x lubridate::union()       masks base::union()
```

```
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)
```

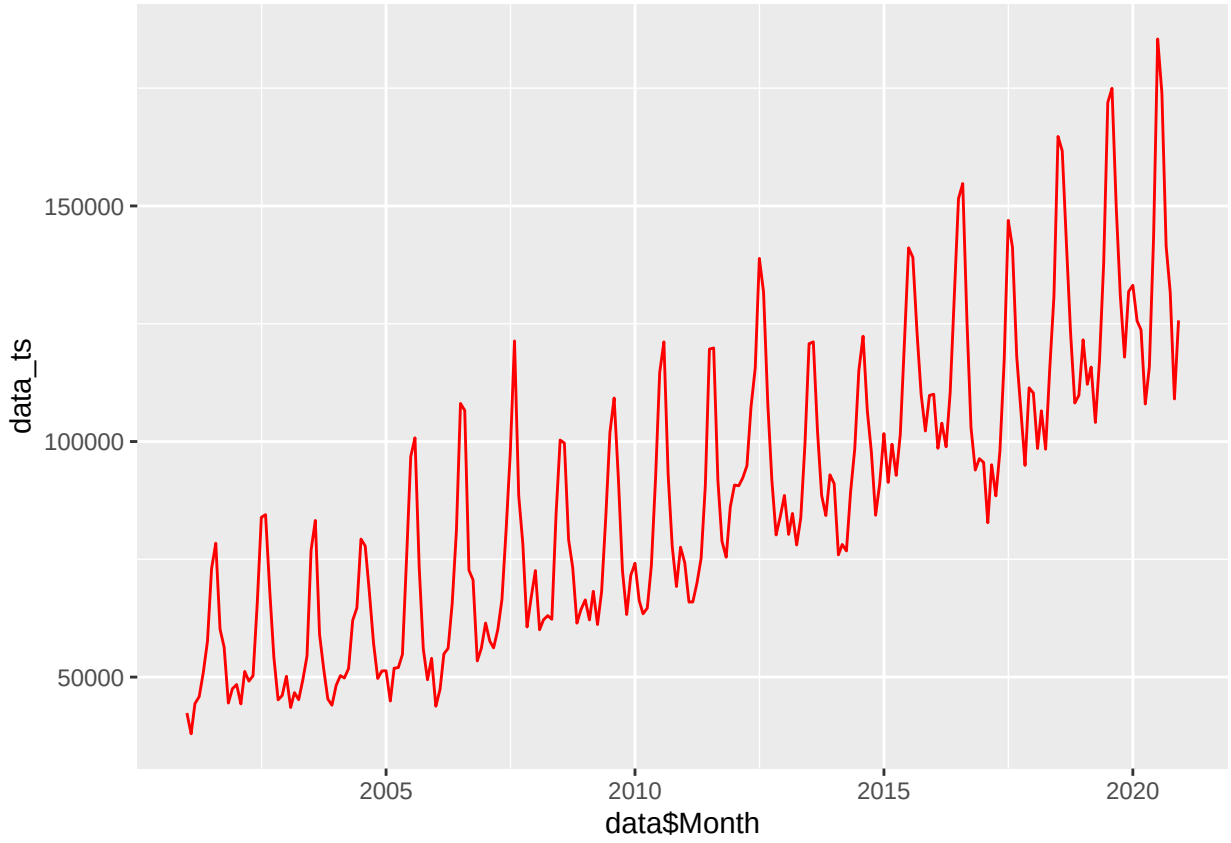
```
##
## -- Column specification --------------------------------------------------------
## cols(
##   Month = col_character(),
##   `all fuels (utility-scale) thousand megawatthours` = col_double(),
##   `coal thousand megawatthours` = col_double(),
##   `natural gas thousand megawatthours` = col_double(),
##   `nuclear thousand megawatthours` = col_double(),
##   `conventional hydroelectric thousand megawatthours` = col_double()
## )
```

```
data <- data %>% rename ("Natural Gas Generation (in 1000s of MWh)" = "natural gas thousand megawatthou
data$Month <- my(data$Month)

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

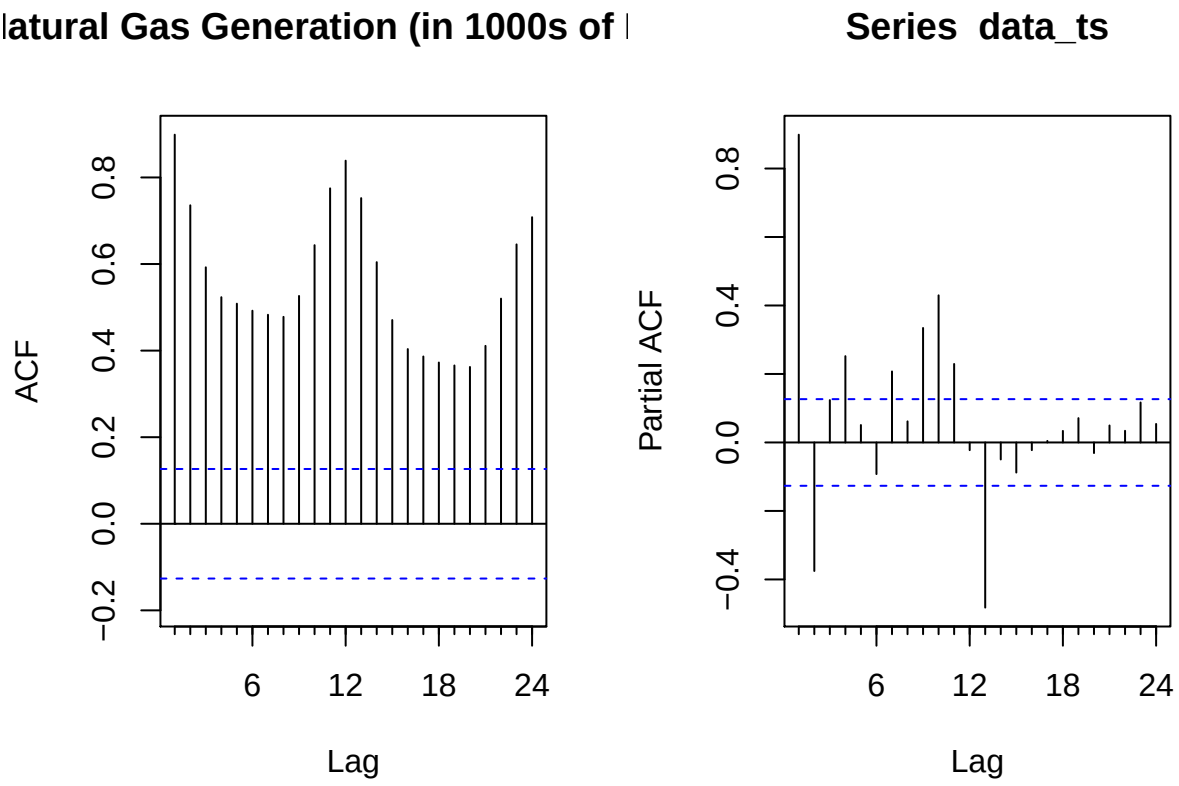
```r
#Converting dataframe to time-series object
data_ts <- ts(data[2], end = c(year(data$Month[1]), month(data$Month[1])), start = c(year(data$Month[no

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

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

```
## Warning: Use of `data$Month` is discouraged. Use `Month` instead.
```




```r
par(mfrow = c(1,2))
Acf(data_ts)
Pacf(data_ts)
```



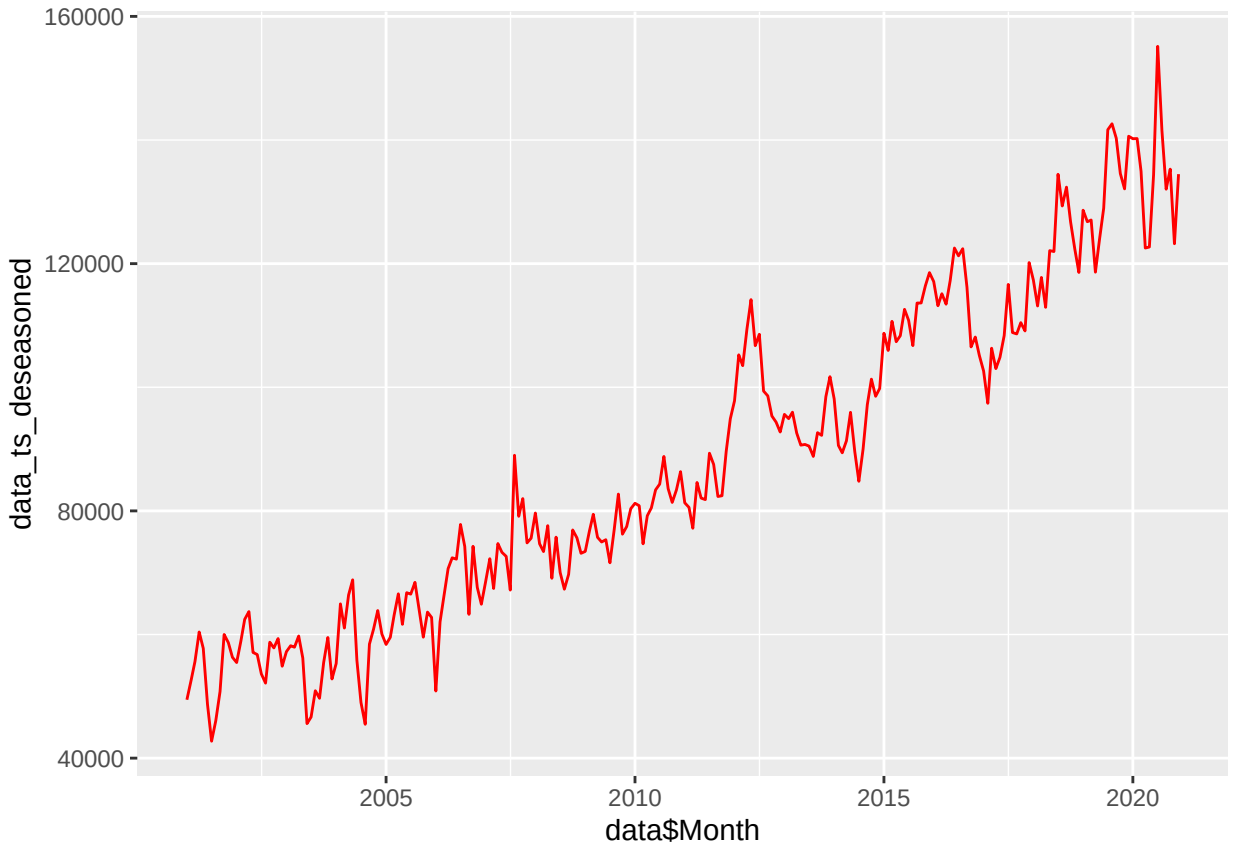
Q2

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.

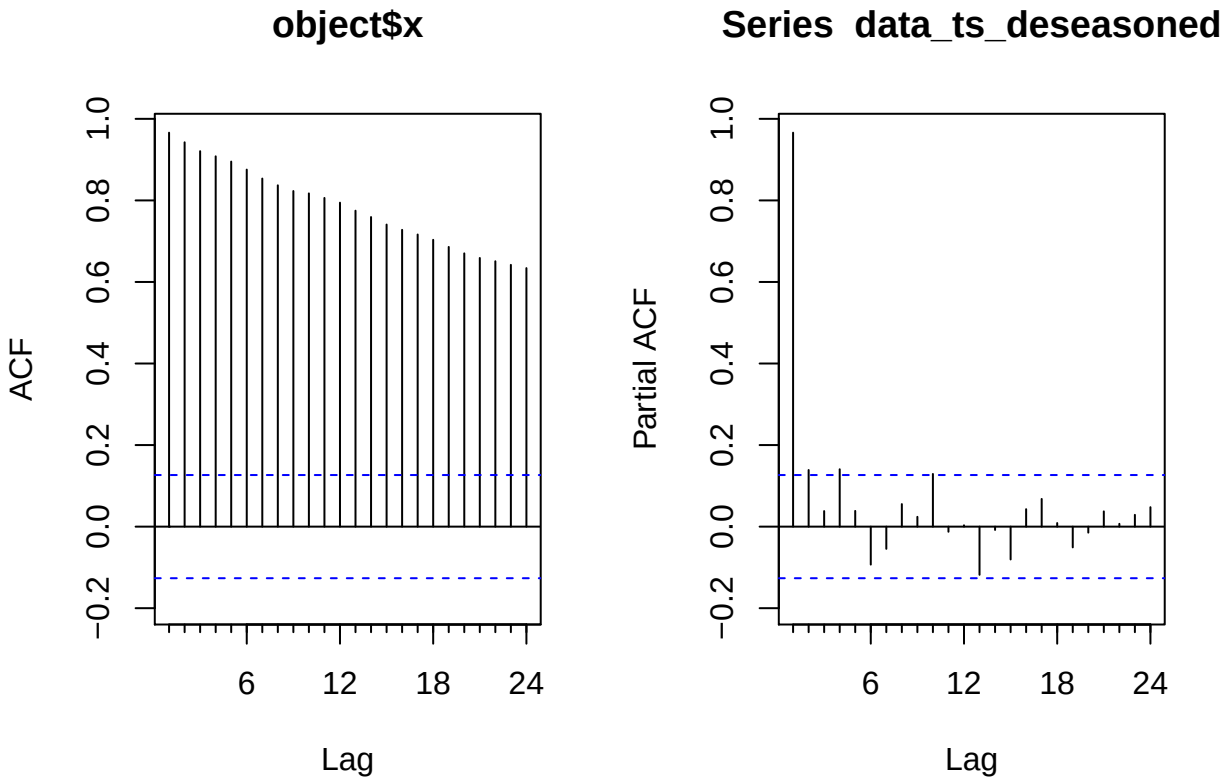
```
#Decomposing time-series using the default "additive" method
data_ts_decomp <- decompose(data_ts)

#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")
```

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

```
## Warning: Use of `data$Month` is discouraged. Use `Month` instead.
```




```
par(mfrow = c(1,2))
Acf(data_ts_deseasoned)
Pacf(data_ts_deseasoned)
```



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))
```

```
## Warning in adf.test(data_ts_deseasoned): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  data_ts_deseasoned
## Dickey-Fuller = -4.0574, 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 is stationary. 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 test done previously. 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.

```
data_ts_deseasoned_arima <- Arima(data_ts_deseasoned, order = c(2,1,0), seasonal = c(0,0,0), include.dri
print(data_ts_deseasoned_arima)
```

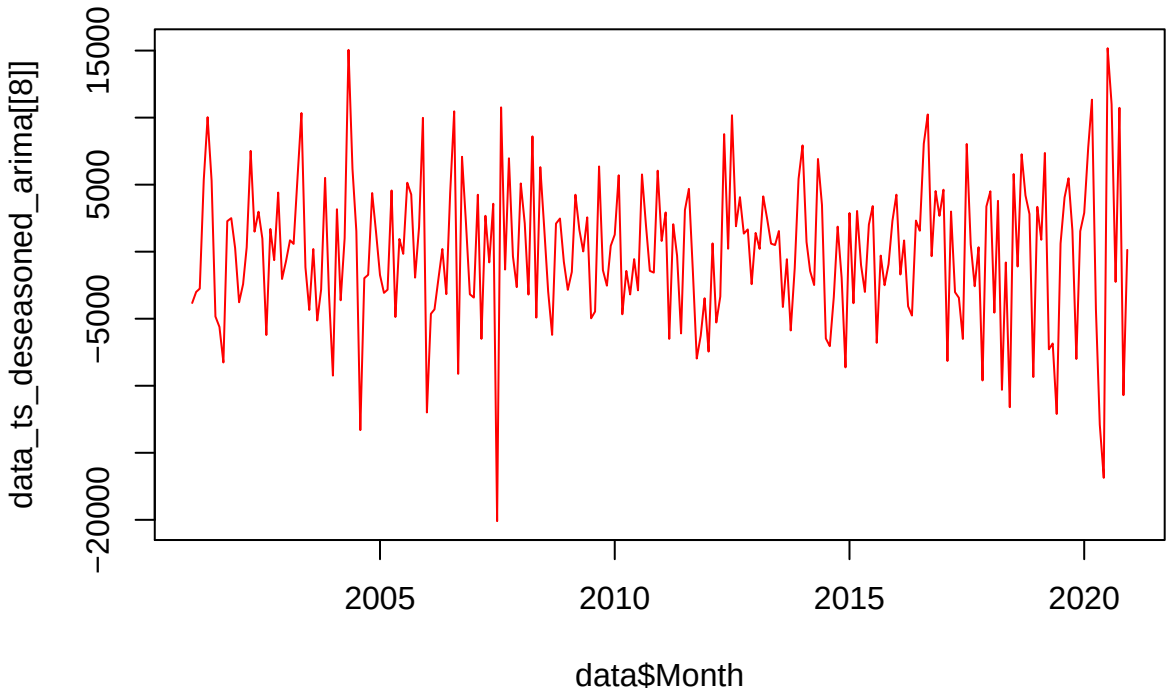
```
## Series: data_ts_deseasoned
## ARIMA(2,1,0) with drift
##
## Coefficients:
##           ar1      ar2      drift
##       -0.1647  -0.1283  -346.9085
## s.e.   0.0645   0.0650   272.1655
##
## sigma^2 estimated as 29891484:  log likelihood=-2394.61
## AIC=4797.21   AICc=4797.39   BIC=4811.12
```

«» cat(sprintf("ϕ_1 =", data_ts_deseasoned_arima[[1]][1]))

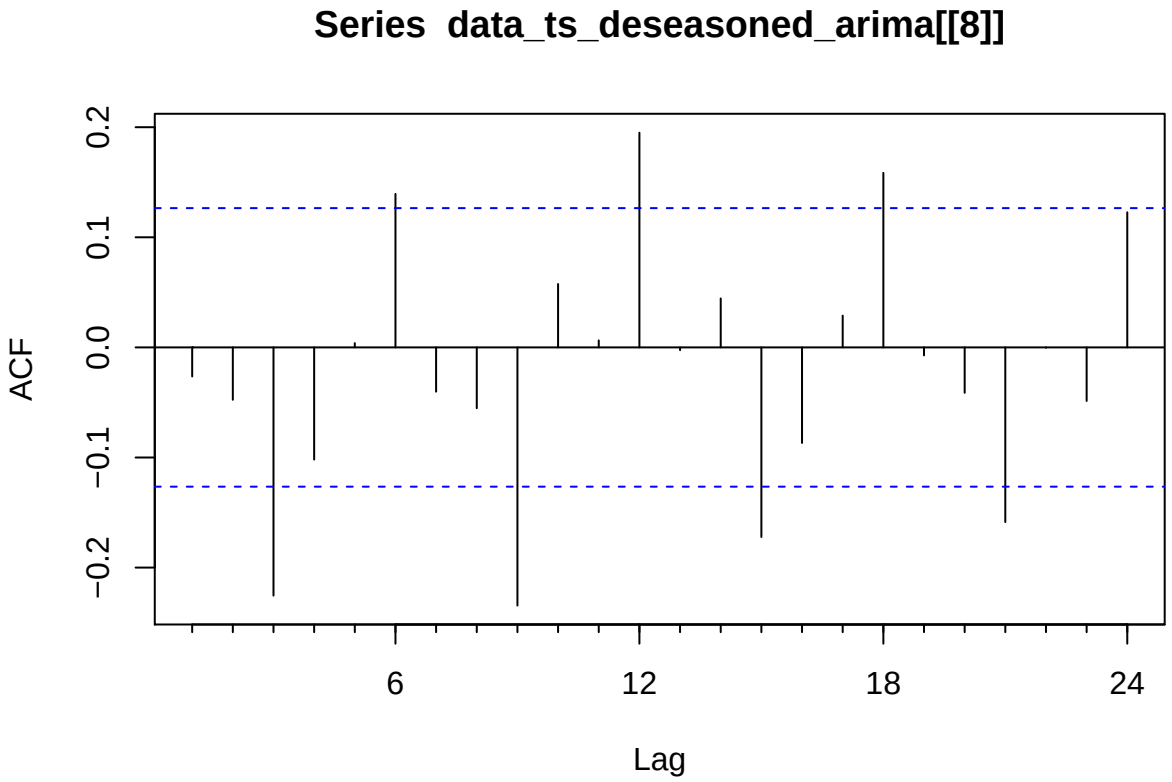
Q6

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?

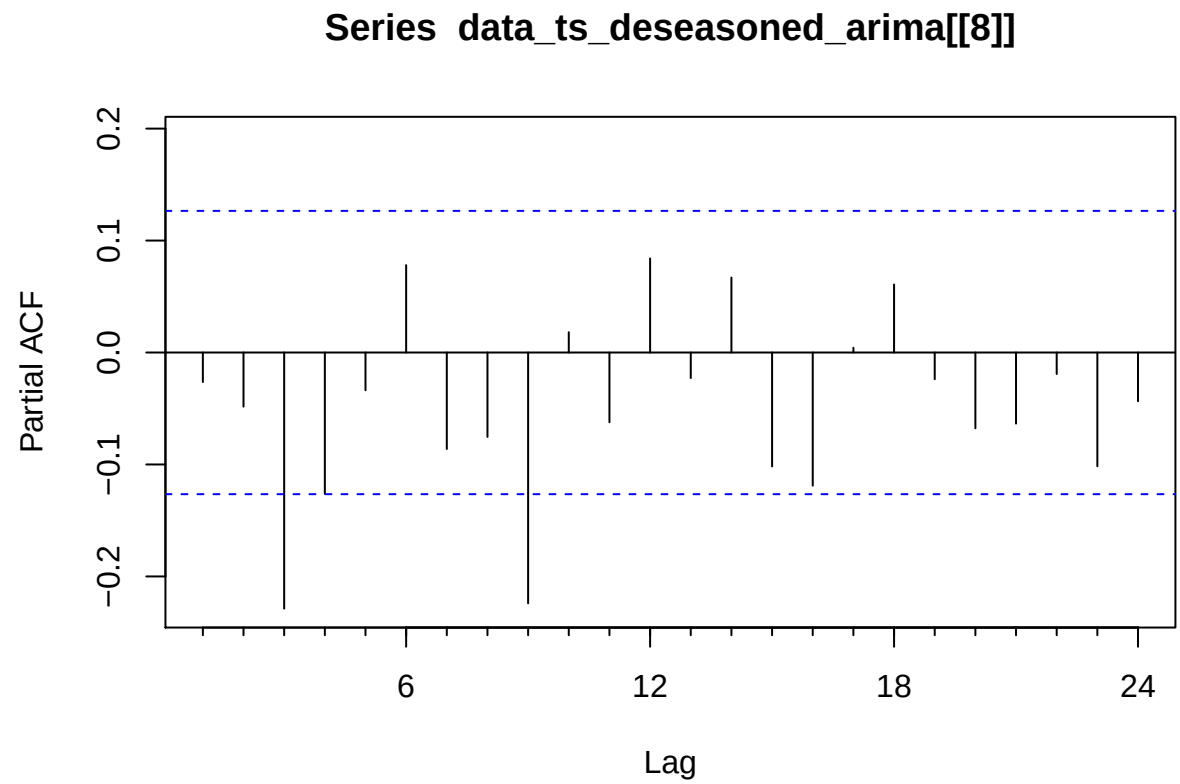
```
plot(x = data$Month, y = data_ts_deseasoned_arima[[8]], type = "l", col = "red")
```




```
Acf(data_ts_deseasoned_arima[[8]])
```



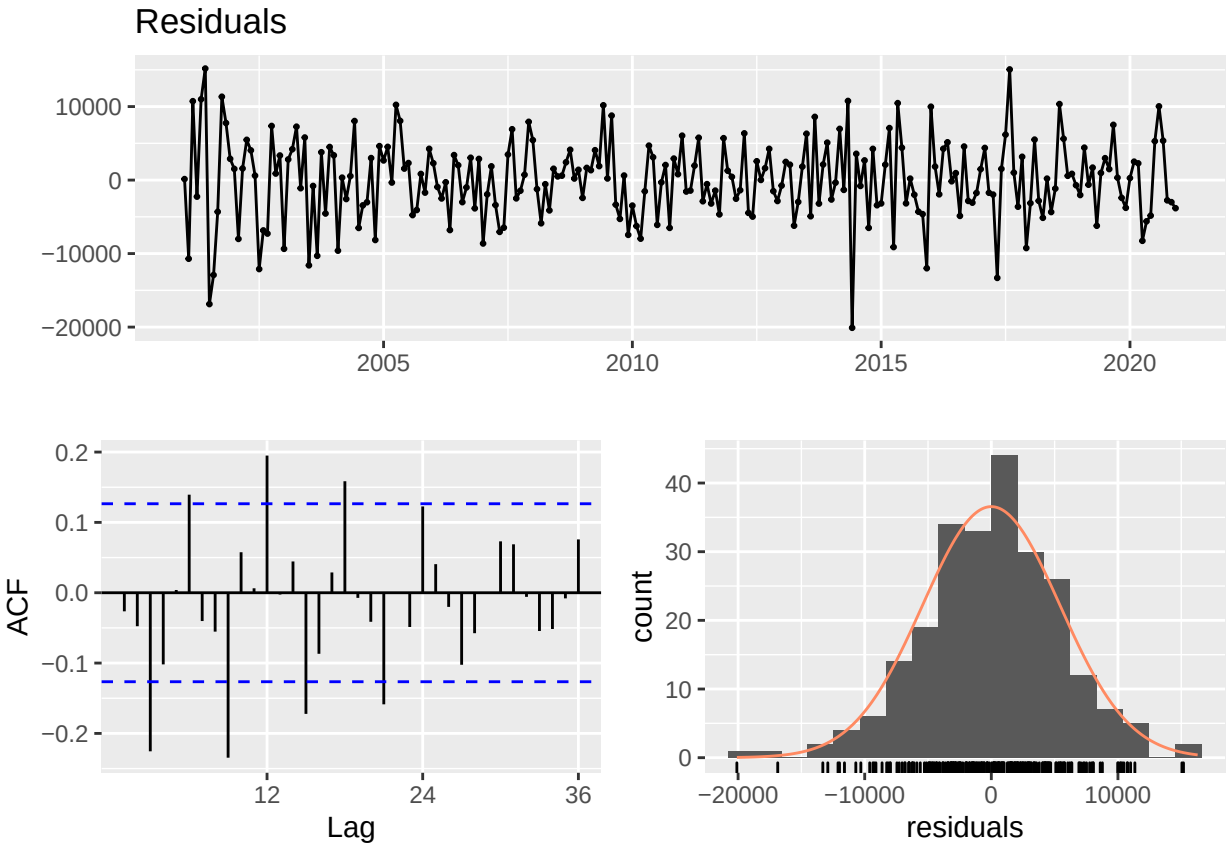

```
Pacf(data_ts_deseasoned_arima[[8]])
```




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

```
## Warning in modeldf.default(object): Could not find appropriate degrees of
```

```
## freedom for this model.
```



Modeling the original series (with seasonality)

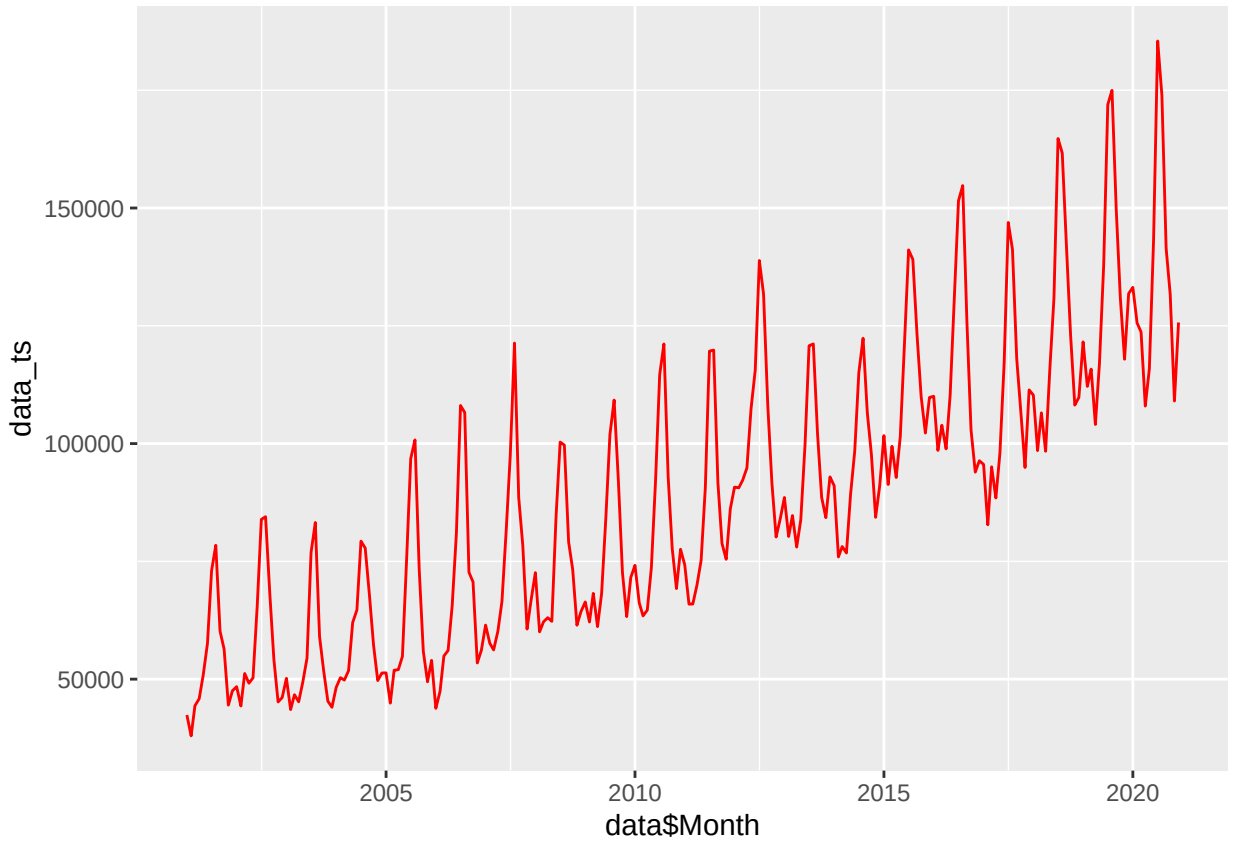
Q7

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.

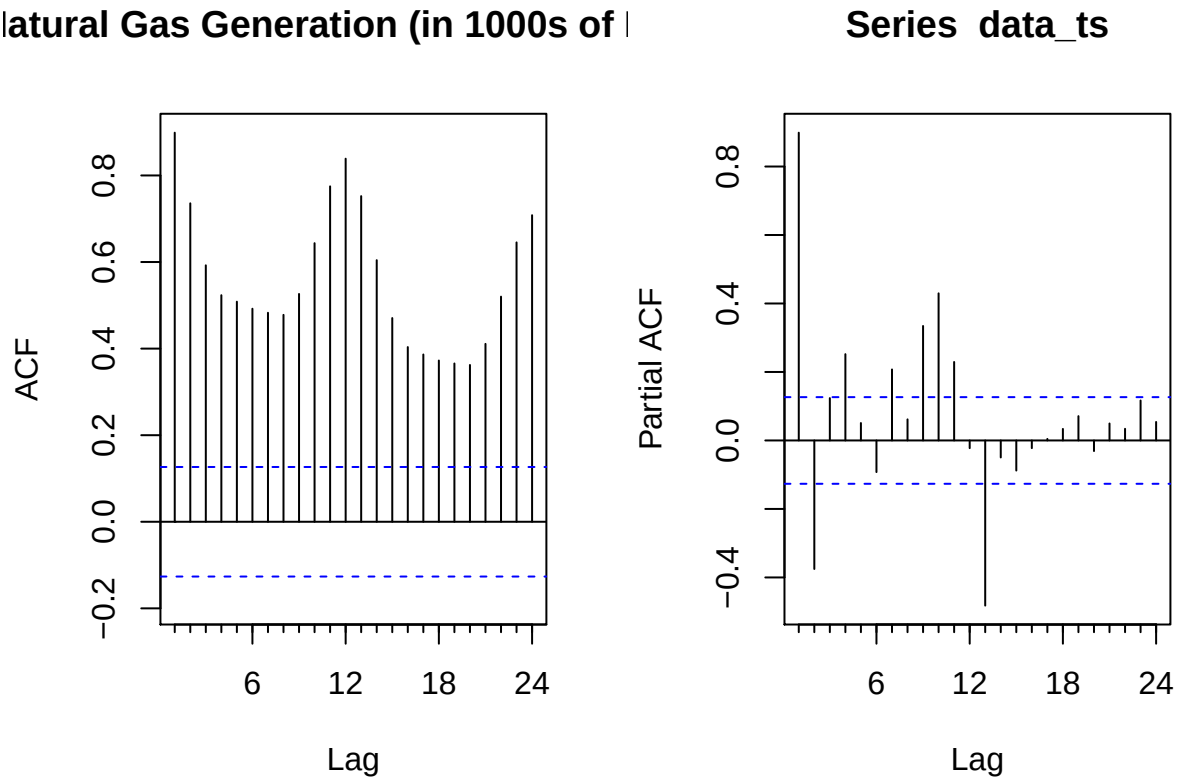
```
ggplot(data, aes(data$Month,data_ts)) +
  geom_line(color = "red")
```

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

```
## Warning: Use of `data$Month` is discouraged. Use `Month` instead.
```




```
par(mfrow = c(1,2))
Acf(data_ts)
Pacf(data_ts)
```



unable to figure out P & Q from the ACF & PCF ache se

```
#ndiff <- ndiffs(data_ts_deseasoned)
#ndiff
ndiff_seasonal <- nsdiffs(data_ts)
print(ndiff_seasonal)
```

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

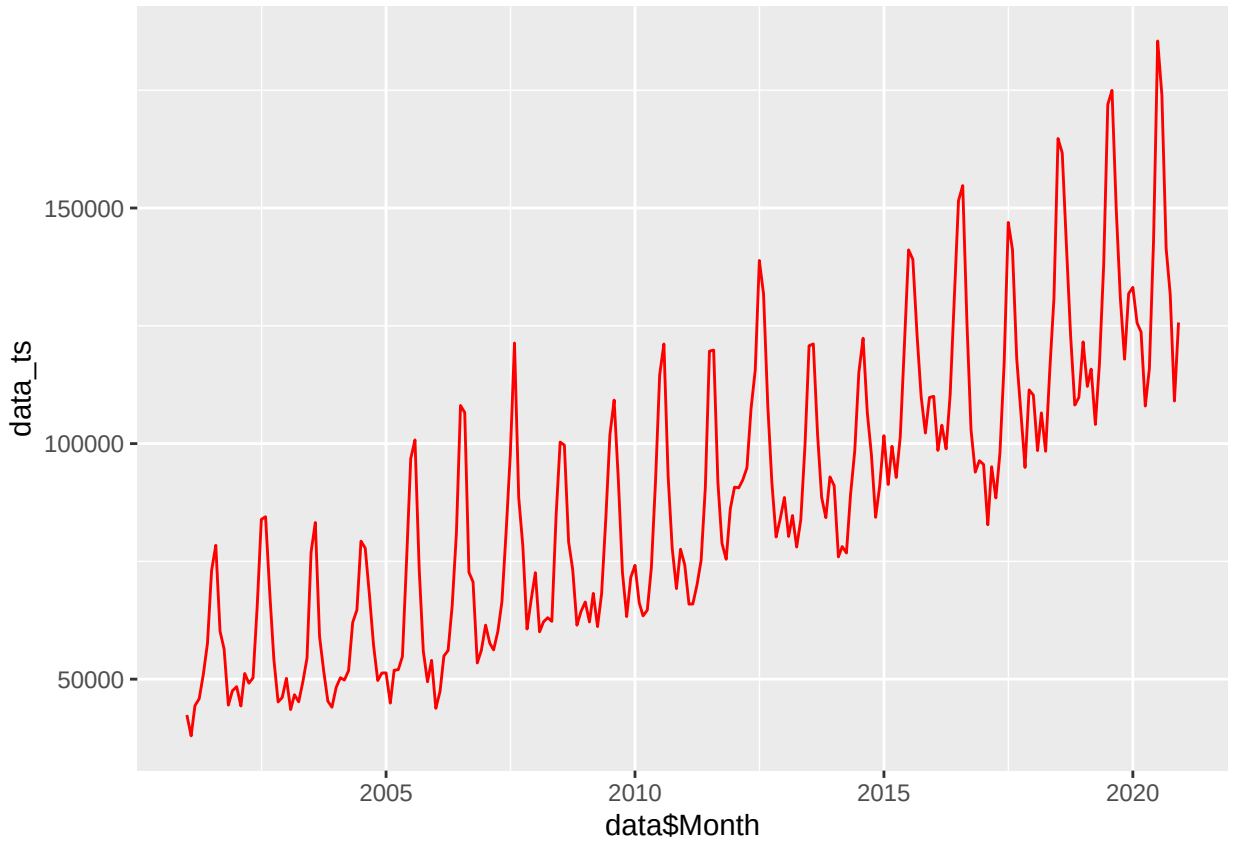
- ndiff_seasonal = 1 and hence we differenced.

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

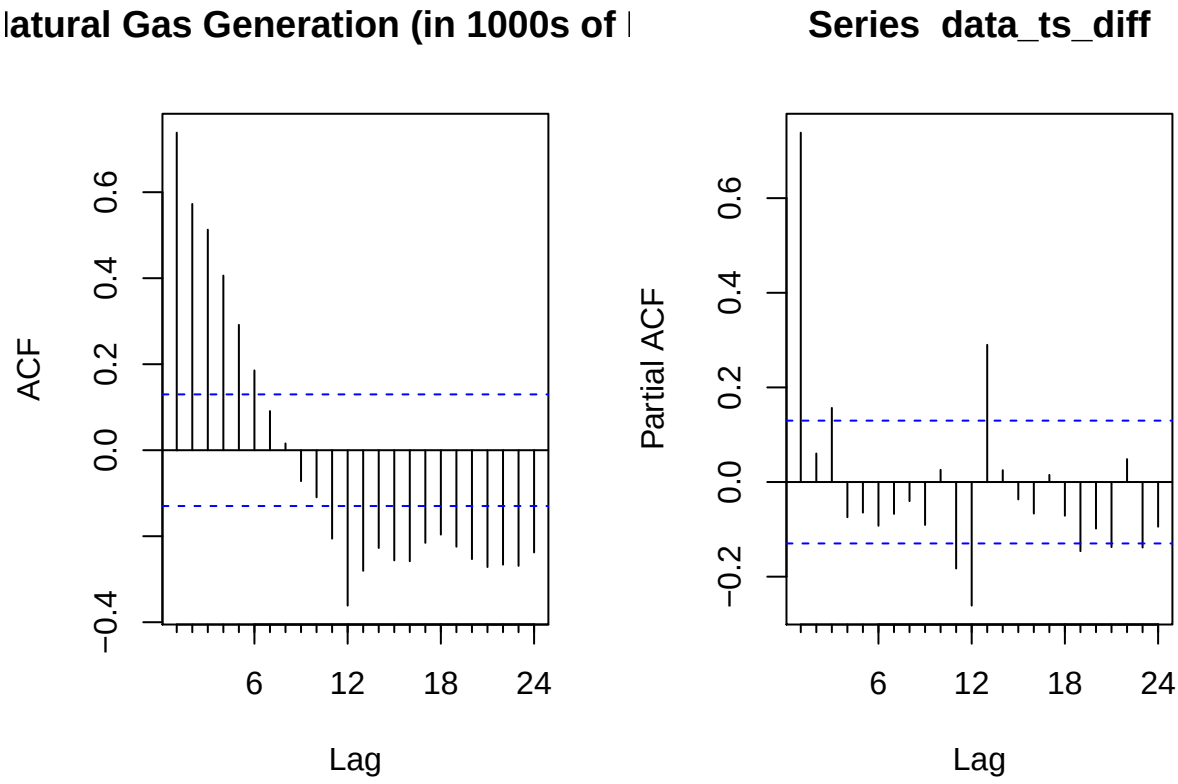
```
ggplot(data, aes(data$Month,data_ts)) +
  geom_line(color = "red")
```

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

```
## Warning: Use of `data$Month` is discouraged. Use `Month` instead.
```



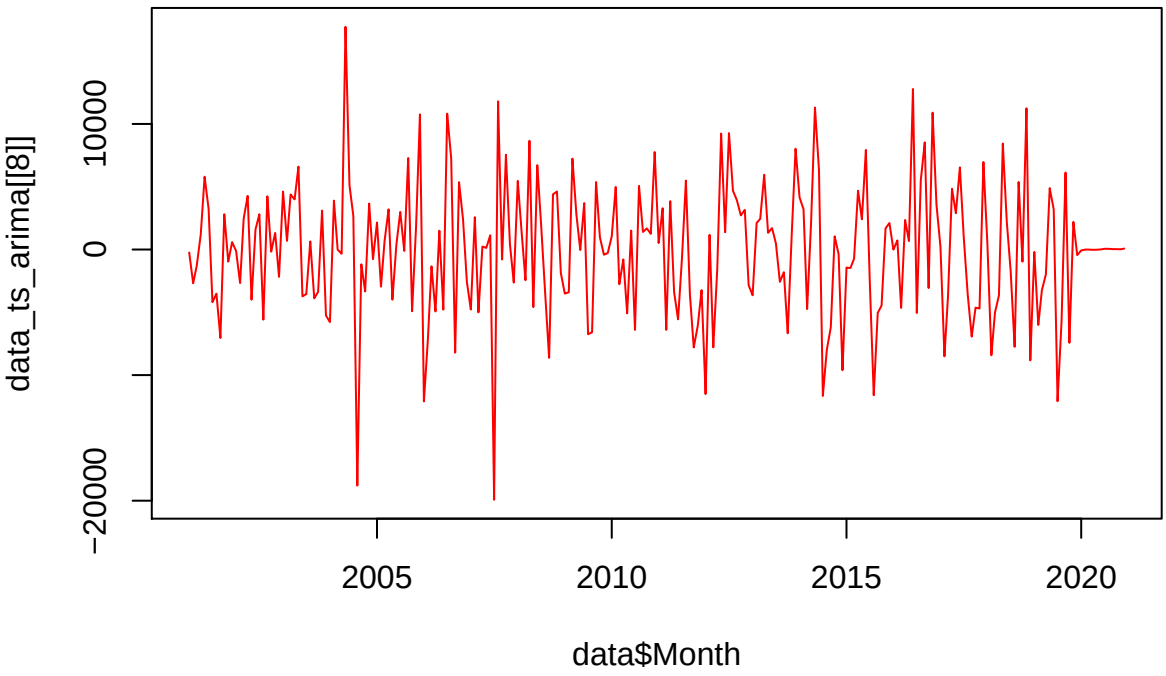

```
par(mfrow = c(1,2))
Acf(data_ts_diff)
Pacf(data_ts_diff)
```



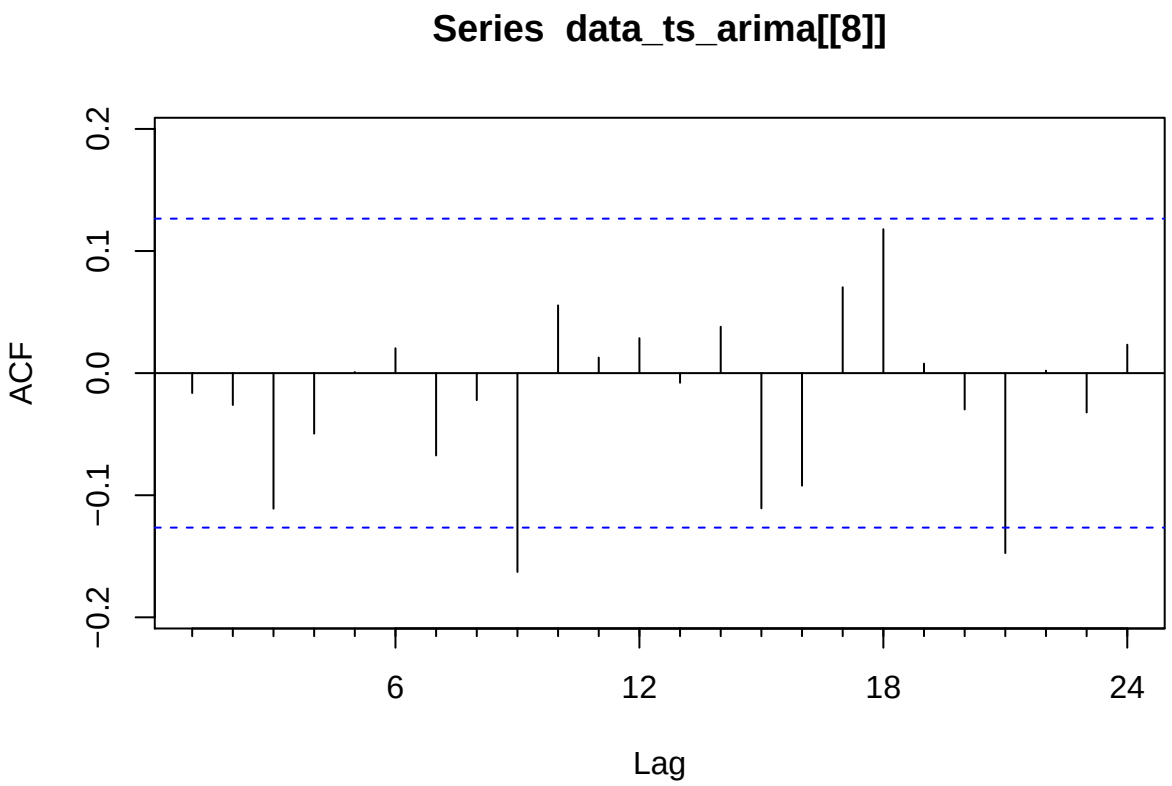
$D = 1, Q = 1$ $P = 0,$

```
data_ts_arima <- Arima(data_ts, order = c(2,1,0), seasonal = c(0,1,1), include.mean = FALSE, include.co
```

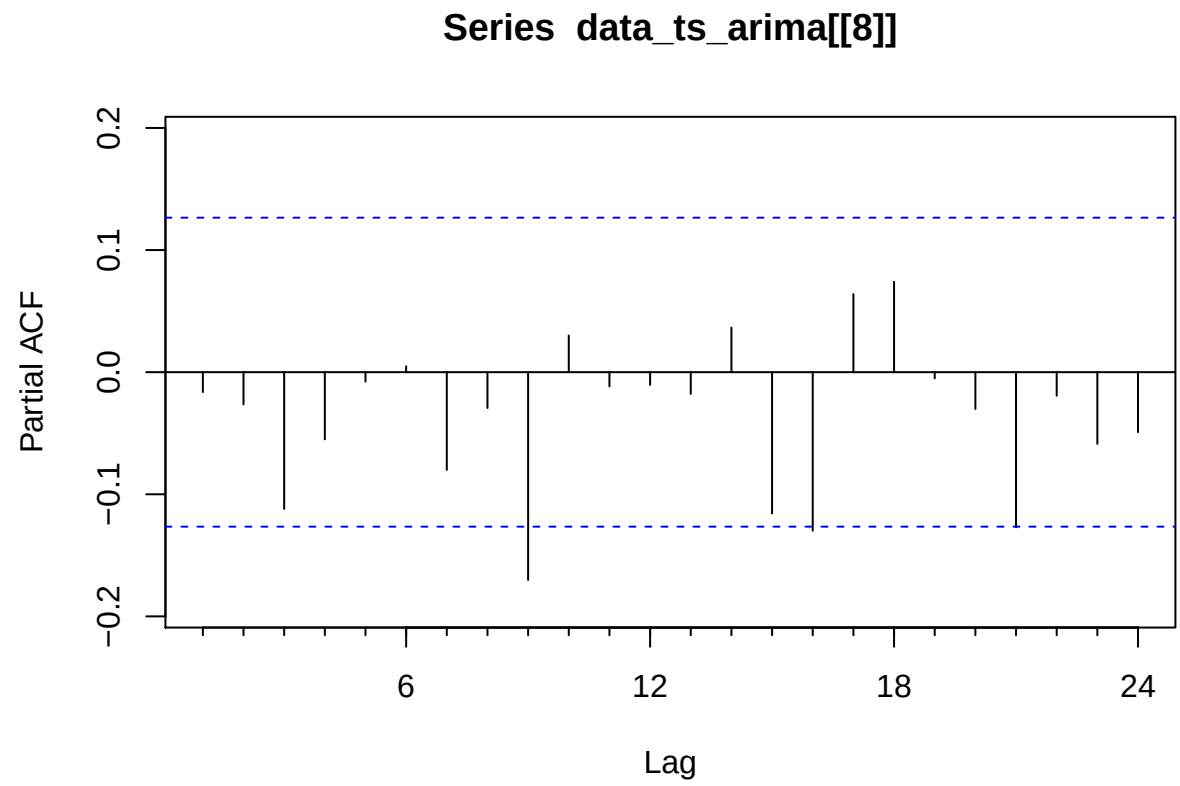
```
plot(x = data$Month, y = data_ts_arima[[8]], type = "l", col = "red")
```




```
Acf(data_ts_arima[[8]])
```



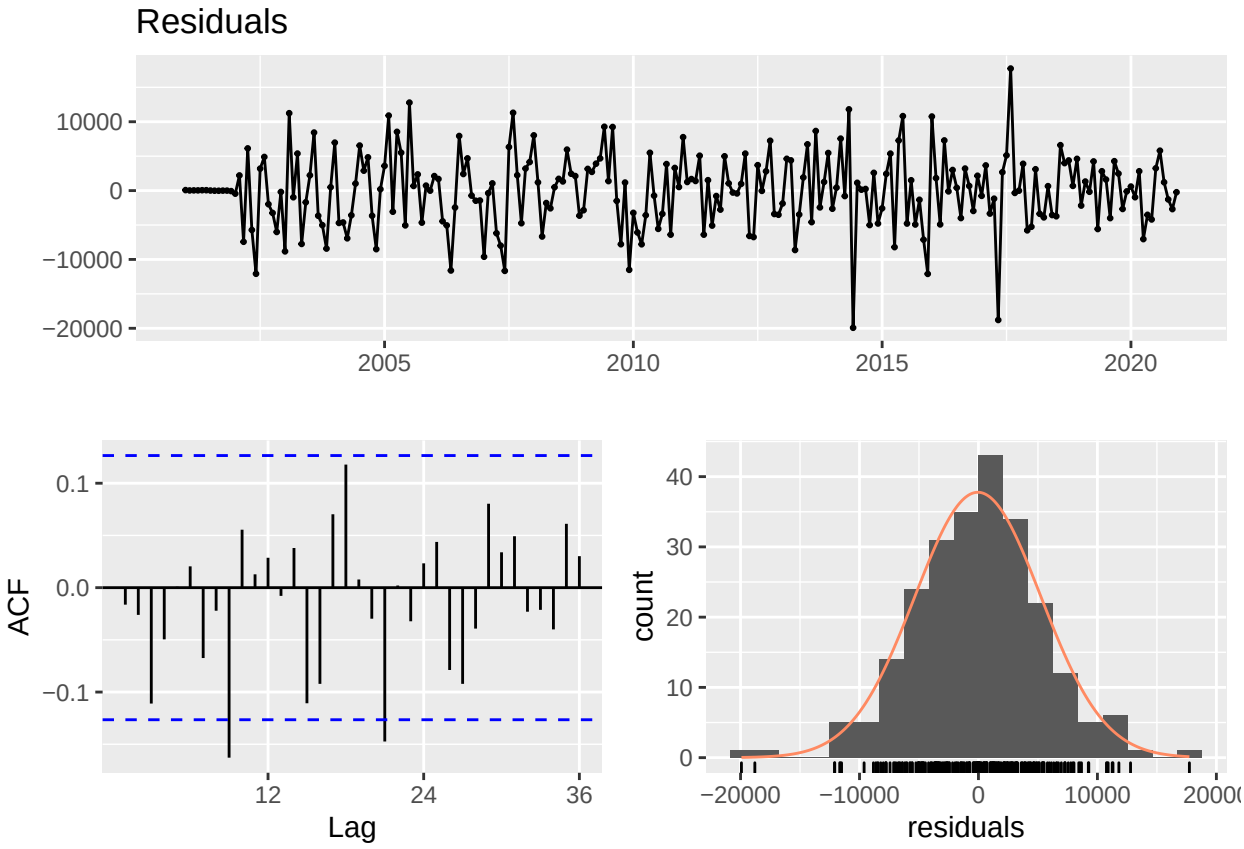

```
Pacf(data_ts_arima[[8]])
```




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

```
## Warning in modeldf.default(object): Could not find appropriate degrees of
```

```
## freedom for this model.
```

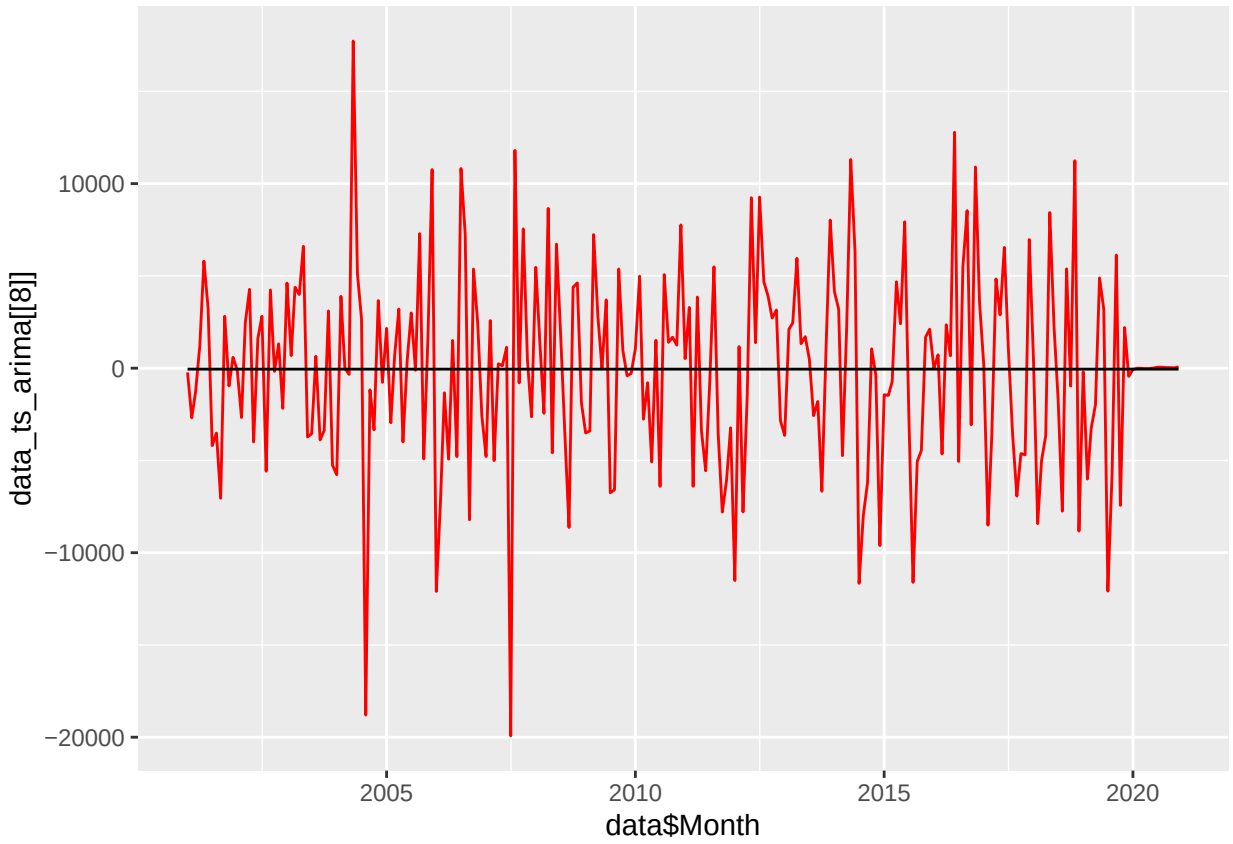


Q8

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.

```
### use summarise and group by to claculate mean, sd, var
residual_av_original <- mean(data_ts_arima[[8]][1:240])
residual_av_deseasoned <- mean(data_ts_deseasoned_arima[[8]][1:240])

ggplot(data, aes(data$Month, data_ts_arima[[8]])) +
  geom_line(color = "red") +
  geom_line(aes(y = residual_av_original))
```

```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
## Warning: Use of `data$Month` is discouraged. Use `Month` instead.

## Warning: Use of `data$Month` is discouraged. Use `Month` instead.
```




```
  geom_line(aes(y = data_ts_deseasoned_arima[[8]]), color = "green")
```

```
## mapping: y = ~data_ts_deseasoned_arima[[8]]
## geom_line: na.rm = FALSE, orientation = NA
## stat_identity: na.rm = FALSE
## position_identity
```

```
  #geom_line(aes(x = data$Month,y = residual_av_deseasoned), color = "yellow")
```

«»

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(2,1,3) with drift
##
```

```
## Coefficients:
##           ar1     ar2     ma1      ma2      ma3      drift
##       -0.3017  0.6483  0.0544  -0.9125  -0.0861  -360.0058
## s.e.   0.0964  0.0930  0.1209   0.0496   0.0999    33.5640
##
## sigma^2 estimated as 26790430:  log likelihood=-2380.74
## AIC=4775.49   AICc=4775.97   BIC=4799.82
```

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?

```
##With or without P,D & Q?
arima_fit_original <- auto.arima(data_ts,max.P = 0, max.Q = 0, max.D = 0, allowdrift = FALSE)
print(arima_fit_deseasoned)
```

```
## Series: data_ts_deseasoned
## ARIMA(2,1,3) with drift
##
## Coefficients:
##           ar1     ar2     ma1      ma2      ma3      drift
##       -0.3017  0.6483  0.0544  -0.9125  -0.0861  -360.0058
## s.e.   0.0964  0.0930  0.1209   0.0496   0.0999    33.5640
##
## sigma^2 estimated as 26790430:  log likelihood=-2380.74
## AIC=4775.49   AICc=4775.97   BIC=4799.82
```