ENV 790.30 - Time Series Analysis for Energy Data | Spring 2022

Assignment 6 - Due date 03/25/22

Aasha Reddy

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_A07_Sp22.Rmd"). Submit this pdf using Sakai.

Set up

```
#Load/install required package here
library(lubridate)
```

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

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

```
library(ggplot2)
library(forecast)
```

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

```
library(Kendall)
library(tseries)
library(outliers)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v tibble  3.1.5     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1
## v purrr   0.3.4
```

```
## -- 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()
```

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.

```
# load data
ng <- read_csv("../Data/Net_generation_United_States_all_sectors_monthly.csv",
               skip = 4)
```

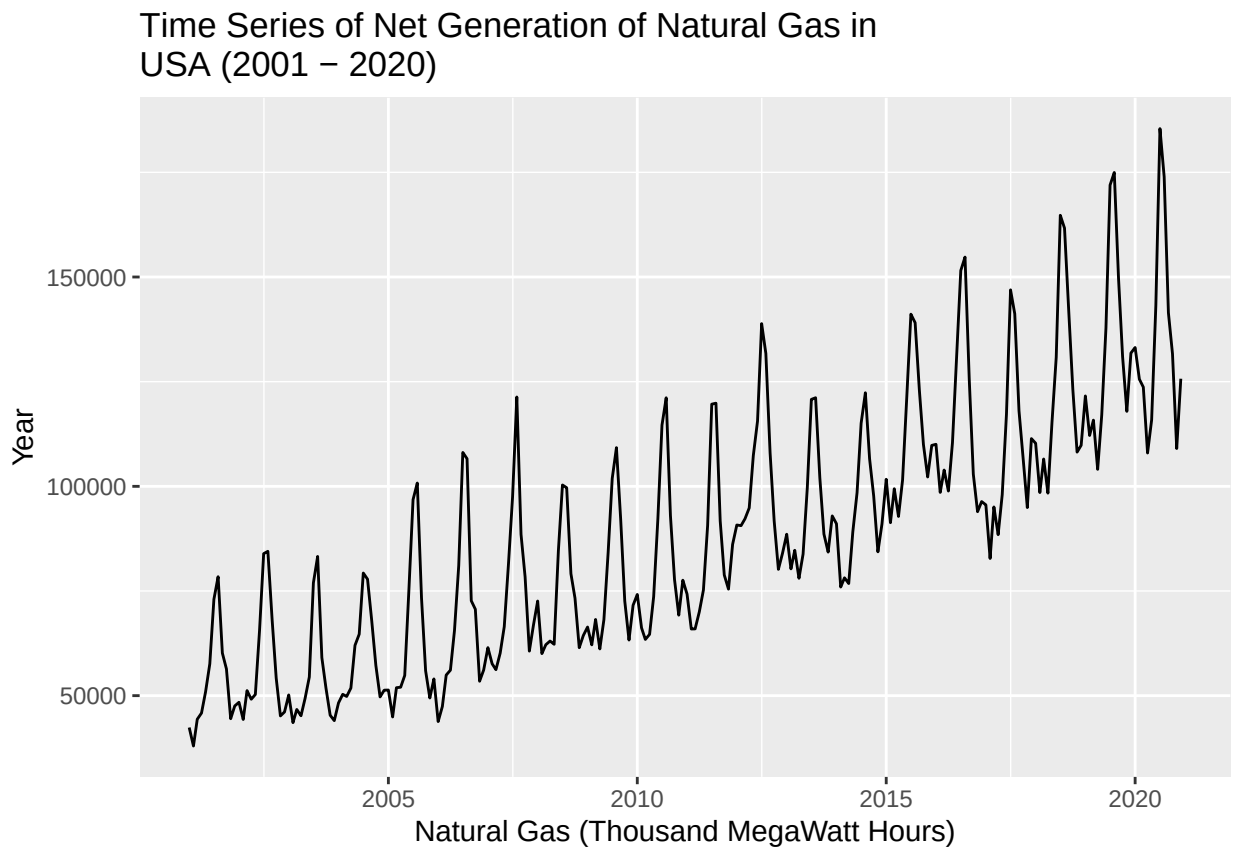
```
## Rows: 240 Columns: 6
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): Month
## dbl (5): all fuels (utility-scale) thousand megawatthours, coal thousand meg...
```

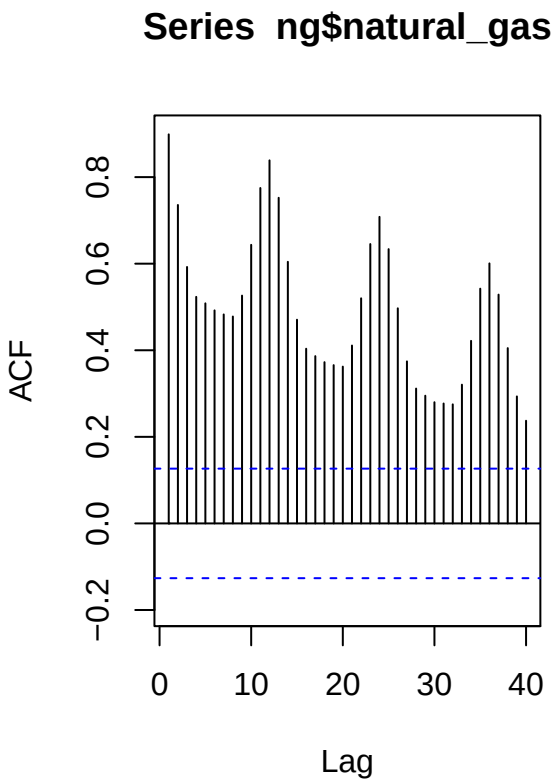
```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

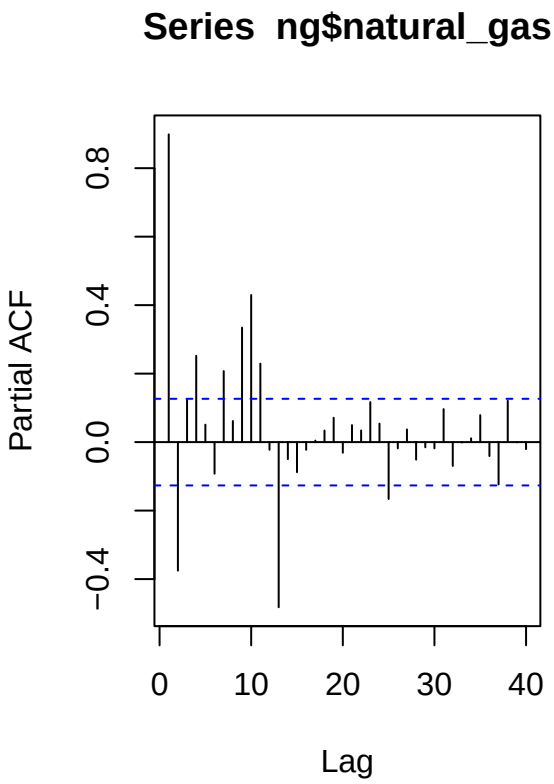
```
# clean data
ng <- ng %>%
  janitor::clean_names() %>%
  mutate(date = my(month)) %>%
  select(-month) %>%
  select(date, natural_gas_thousand_megawatthours) %>%
  arrange(date) %>%
  rename(natural_gas = natural_gas_thousand_megawatthours)


# create time series object
ng_ts <- ts(ng, start = 1, frequency = 12)
```

```
# plot time series
ggplot(data = ng, aes(x = date, y = natural_gas)) +
  geom_line() +
  labs(y = "Year",
       x = "Natural Gas (Thousand MegaWatt Hours)",
       title = "Time Series of Net Generation of Natural Gas in \nUSA (2001 - 2020)")
```




```
# plot acf and pacf
par(mfrow=c(1,2))

Acf(ng$natural_gas, lag = 40, plot = TRUE)
Pacf(ng$natural_gas, lag = 40, plot = TRUE)
```

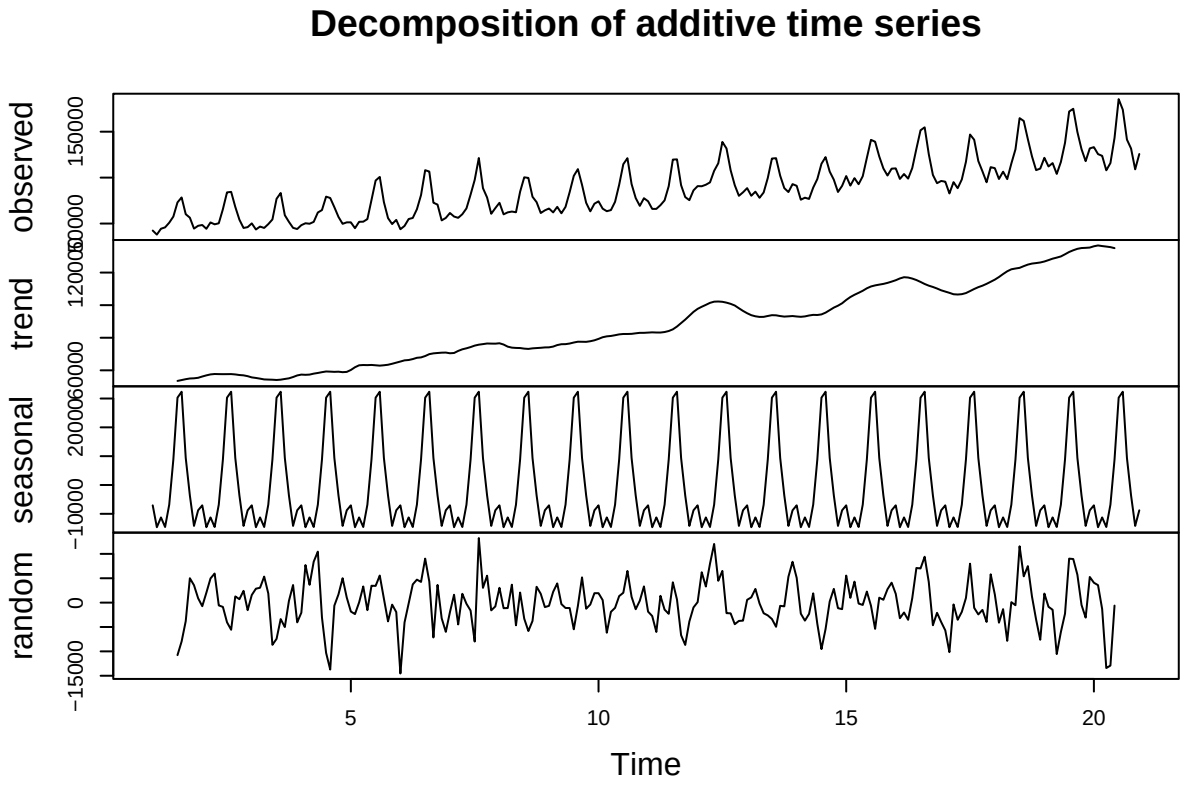




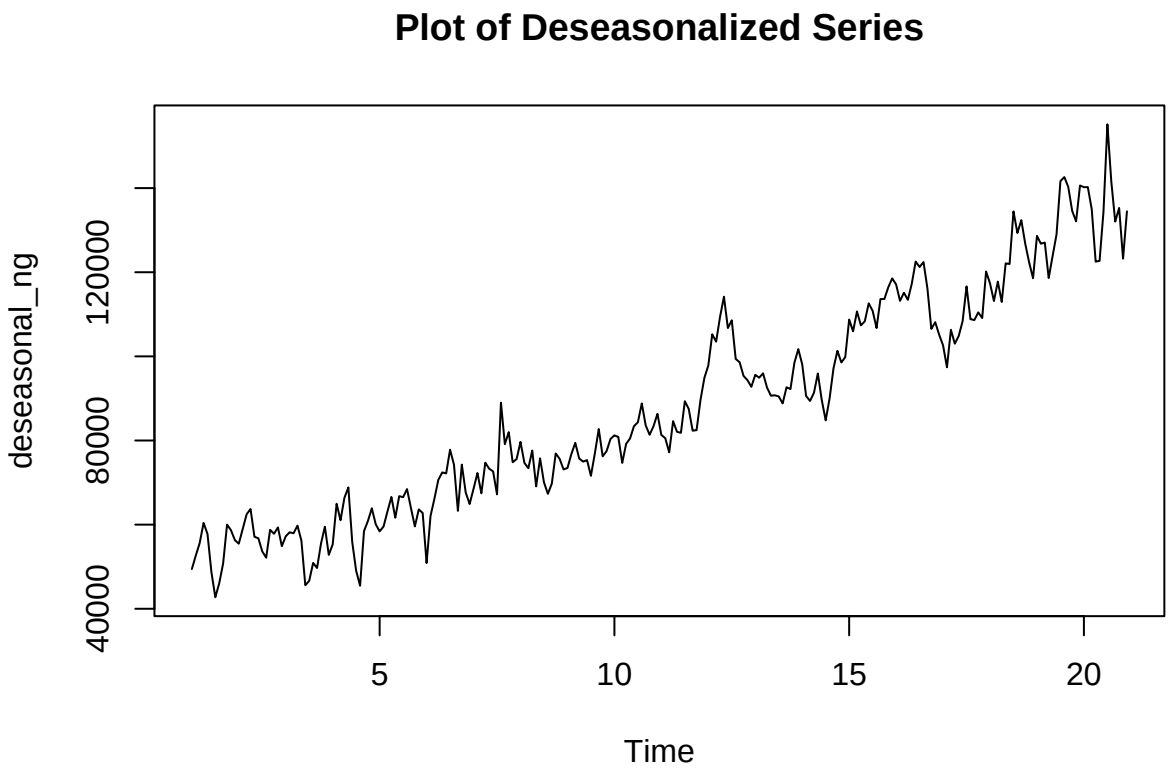
Q2

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.

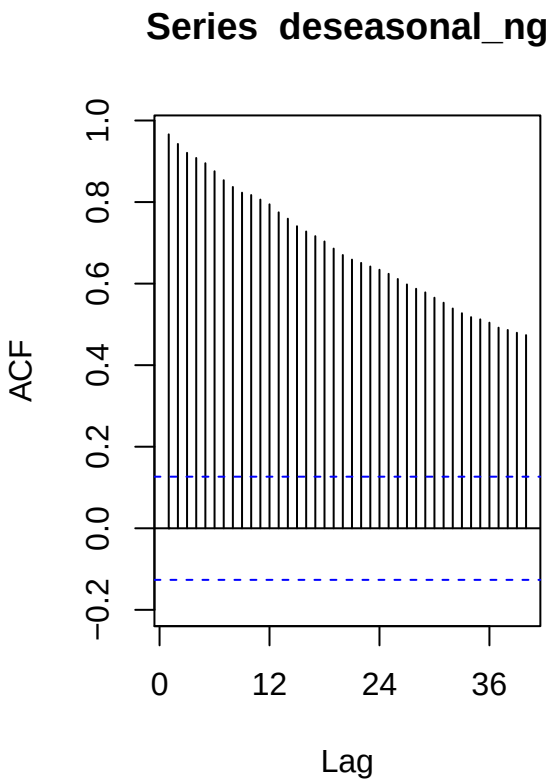
```
#Using R decompose function
decompose_ng <- decompose(ng_ts[,"natural_gas"],"additive")
plot(decompose_ng)
```

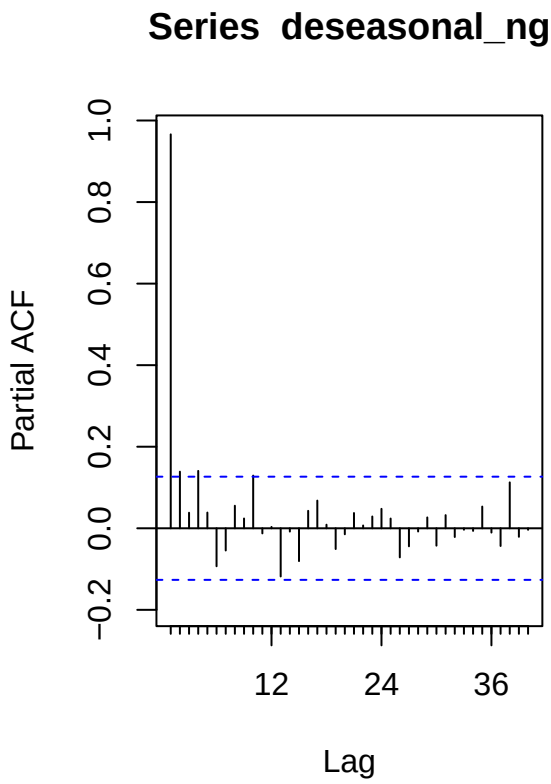



```
# deseason
deseasonal_ng <- seasadj(decompose_ng)
plot(deseasonal_ng, main = "Plot of Deseasonalized Series")
```




```
# plot acf and pacf
par(mfrow=c(1,2))

Acf(deseasonal_ng, lag = 40, plot = TRUE)
Pacf(deseasonal_ng, lag = 40, plot = TRUE)
```





We can see that for the untransformed data, the ACF has a clear scalloping pattern that indicates seasonality. After decomposing and de-seasoning the natural gas series, we can see the ACF no longer has the scalloped pattern indicating we dealt with the seasonality. This is also evident in the PACF, we see that in the de-seasoned series, the PACF no longer has any cutoffs after lag 1 whereas we could see many cutoffs following the yearly pattern in the untransformed PACF plot.

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.

```
# Run adf
print((adf.test(deseasonal_ng, alternative="stationary")))
```

```
## Warning in adf.test(deseasonal_ng, alternative = "stationary"): p-value smaller
## than printed p-value
```

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

```
# Run mann-kendall
print(MannKendall(deseasonal_ng))
```

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

The p-value for the Mann Kendall test is almost 0, and thus we reject the null hypothesis that there is no deterministic trend. This suggests that the natural gas series has a deterministic trend.

The p-value for the ADF test is 0.01 (small), thus we reject the null hypothesis that there is a stochastic trend. This suggests that the natural gas series does not have a stochastic trend. This means we can fit ARIMA on this data and do not need to difference the data.

Overall, there is a deterministic trend.

Q4

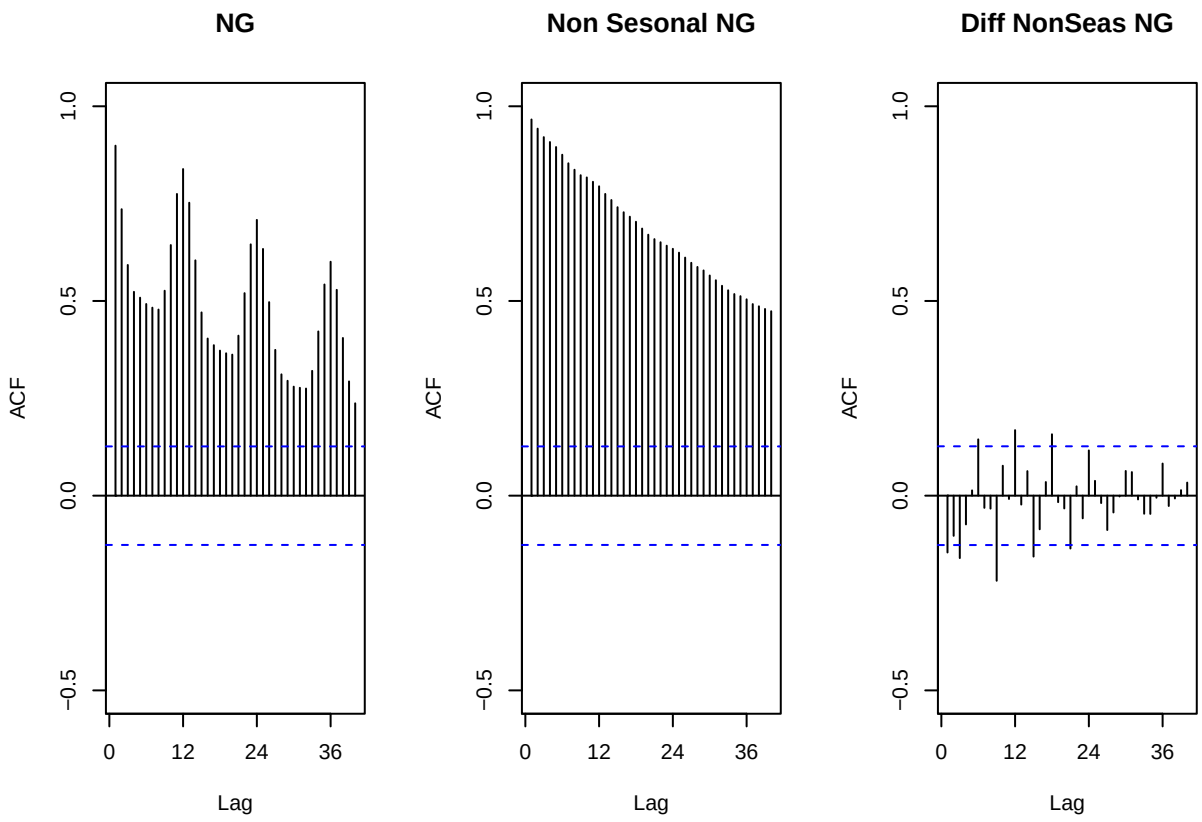
Using the plots from Q2 and test results from Q3 identify the ARIMA model parameters p, d and q. Note that in this case because you removed the seasonal component prior to identifying the model you don't need to worry about seasonal component. Clearly state your criteria and any additional function in 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.

```
n_diff <- ndiffs(deseasonal_ng)
cat("Number of differencing needed: ",n_diff)
```

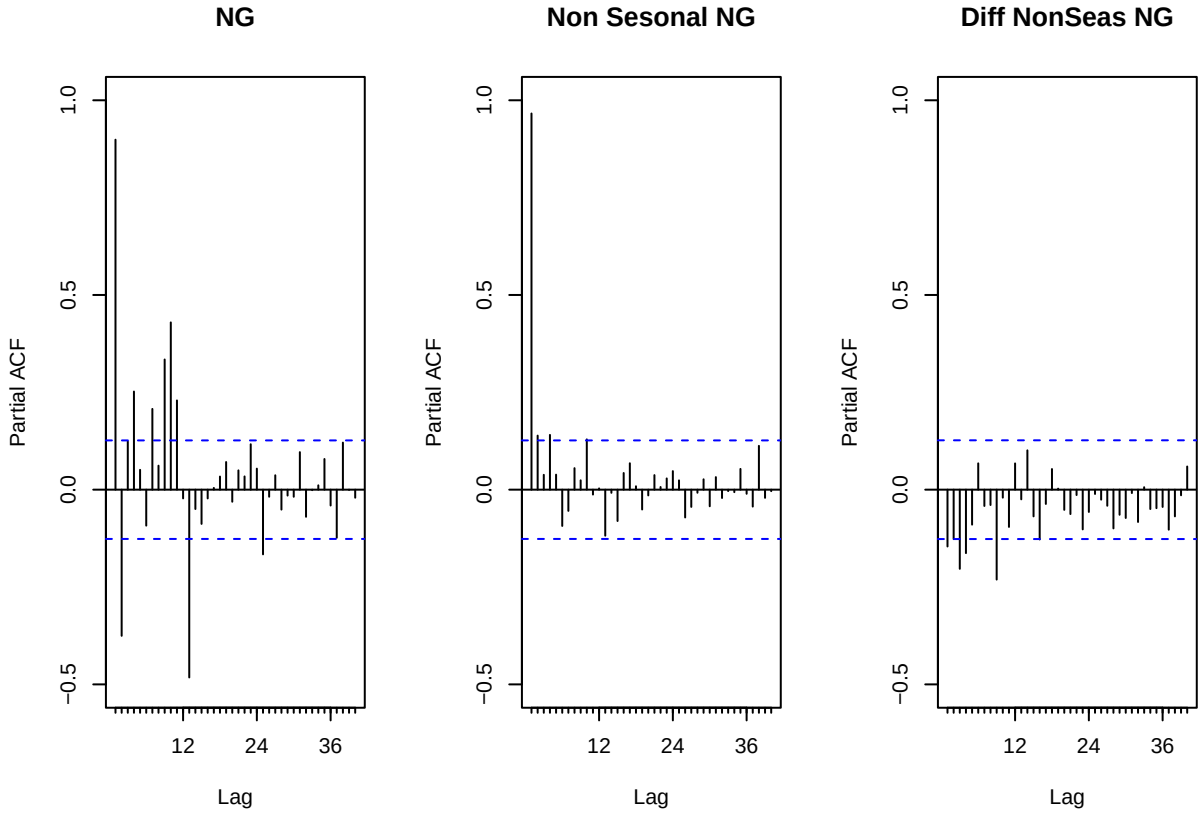
```
## Number of differencing needed:  1
```

```
# difference once
deseasonal_ng_diff <- diff(deseasonal_ng,differences=1,lag=1)
```

```
# we can then check the acfs again
par(mfrow=c(1,3))

Acf(ng_ts[, "natural_gas"],lag.max=40,main="NG",ylim=c(-.5,1))
Acf(deseasonal_ng,lag.max=40,main="Non Sesonal NG",ylim=c(-.5,1))
Acf(deseasonal_ng_diff,lag.max=40,main="Diff NonSeas NG",ylim=c(-.5,1))
```




```
#Comparing PACFs
par(mfrow=c(1,3))
Pacf(ng_ts[, "natural_gas"],lag.max=40,main="NG",ylim=c(-.5,1))
Pacf(deseasonal_ng,lag.max=40,main="Non Sesonal NG",ylim=c(-.5,1))
Pacf(deseasonal_ng_diff,lag.max=40,main="Diff NonSeas NG",ylim=c(-.5,1))
```



Because we have stochastic deterministic trend, we have to difference. Based on the plots of ACF and PACF after differencing and results from Q3, we should use an ARMA(1,1,1) model. First of all, based on the test results, after de-seasoning, we have a stochastic deterministric trend telling us that an ARMA model is appropriate. The PACF plot tells us the AR process term, p, and it cuts off at lag 1, meaning p = 1. The ACF plot tells us the MA process term, q, and there is no cutoff, meaning that q = 1. We also need to difference once according to the ndiffs function, so d = 1.

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.

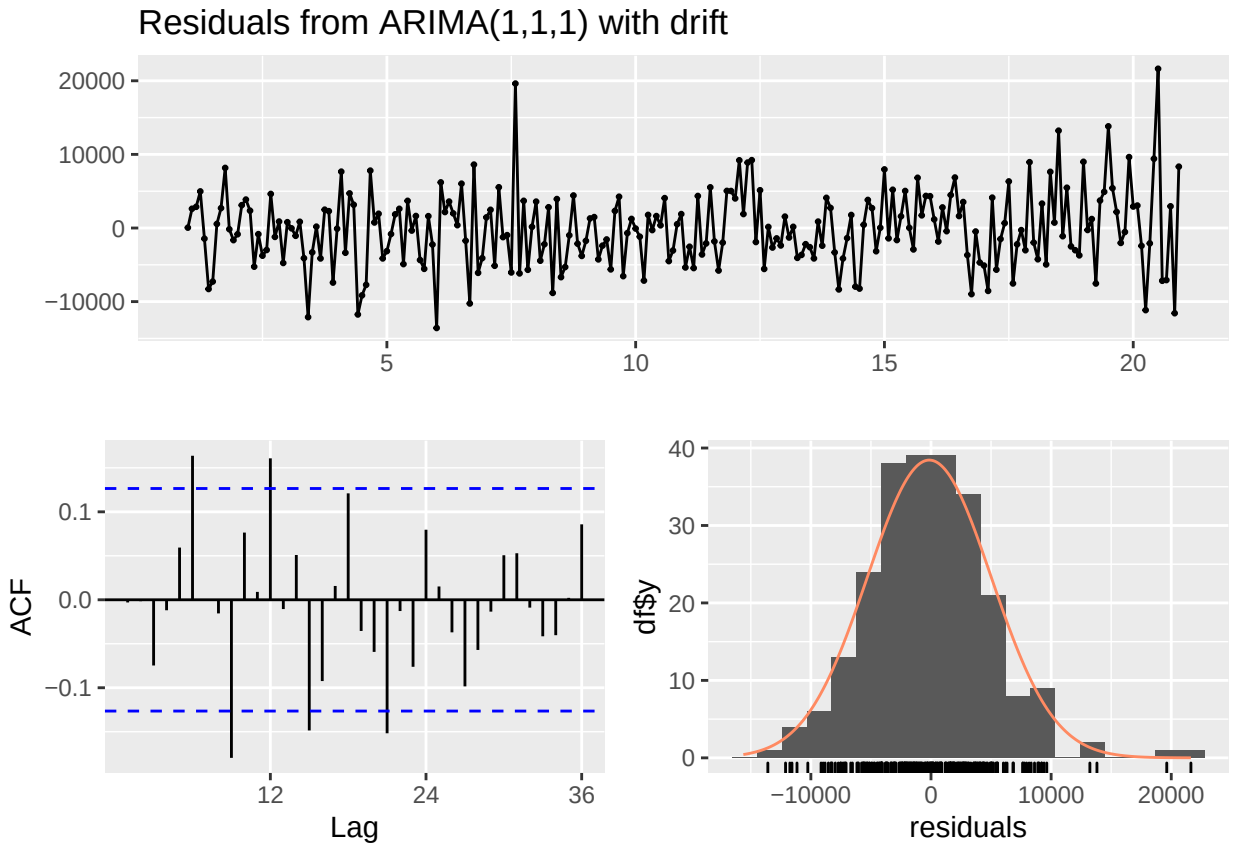
```
arma_10 <- Arima(deseasonal_ng, order = c(1,1,1), include.drift = TRUE)

# print coefficients
arma_10$coef
```

```
##         ar1         ma1       drift
##   0.7065237  -0.9794655 359.5051904
```

Q6

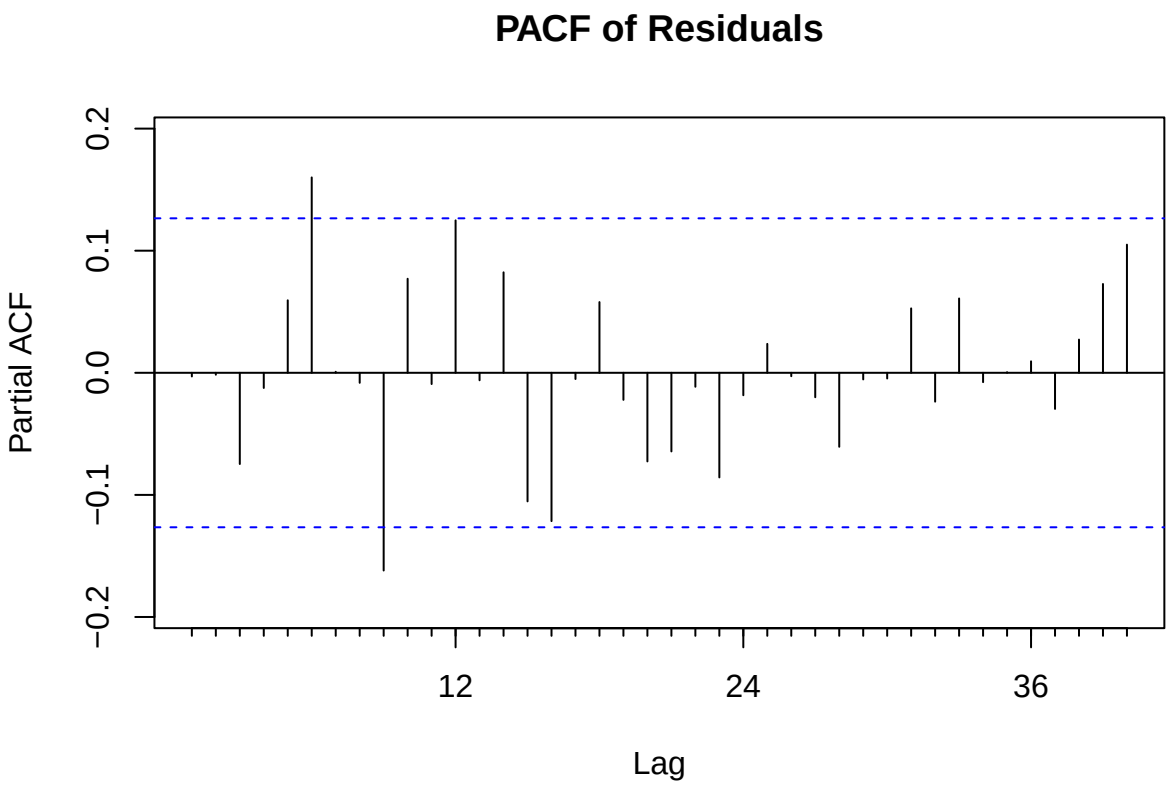
Now plot the residuals of the ARIMA fit from Q5 along with residuals ACF and PACF on the same window. You may use the *checkresiduals*() function to automatically generate the three plots. Do the residual series look like a white noise series? Why?

```
checkresiduals(arma_10)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,1) with drift
## Q* = 48.356, df = 21, p-value = 0.000615
##
## Model df: 3.   Total lags used: 24
```

```
Pacf(arma_10$residuals, lag = 40, main = "PACF of Residuals")
```



Yes, the residual series does look relatively like a white noise series. There seems to be no apparent pattern in the residuals in the first plot, they are also relatively balanced around 0. They also seem to be normally distributed according to the 3rd plot. The ACF plot shows us that there may still be some seasonal pattern though. However, there are some residual ACFs that are over the cutoff. The ACF plot also seems to show a slightly seasonal trend. I would conclude that this model is not the best fit.

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.

First we need to difference our series, we check how many differences we need below and find we need to difference 1 time.

```
# Find out how many non seasonal diffs
n_diff <- ndiffs(ng_ts)
cat("Number of differencing needed for non-seasonal compontent: ", n_diff)
```

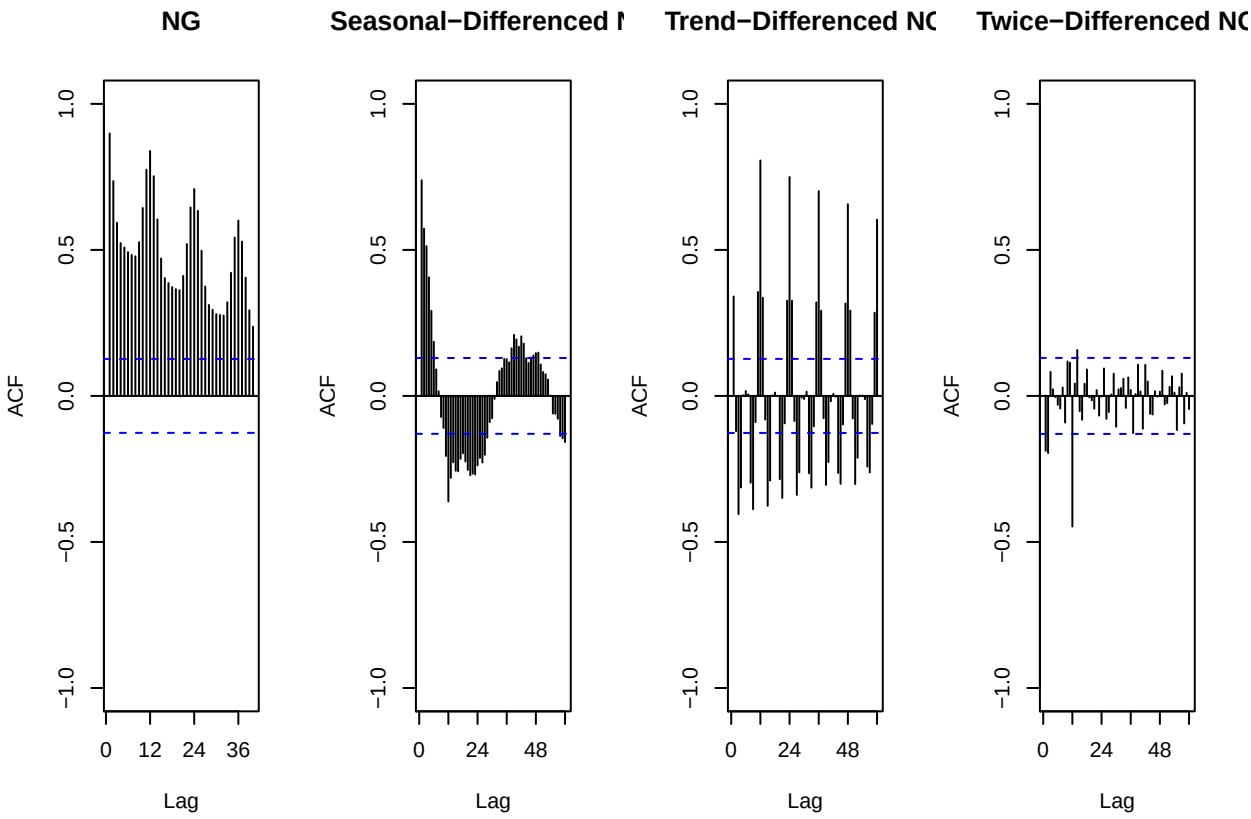
```
## Number of differencing needed for non-seasonal compontent:  1
```

```
# Find out how many time we need to difference
ns_diff <- nsdiffs(ng_ts[,"natural_gas"])
cat("Number of seasonal differencing needed: ",ns_diff)
```

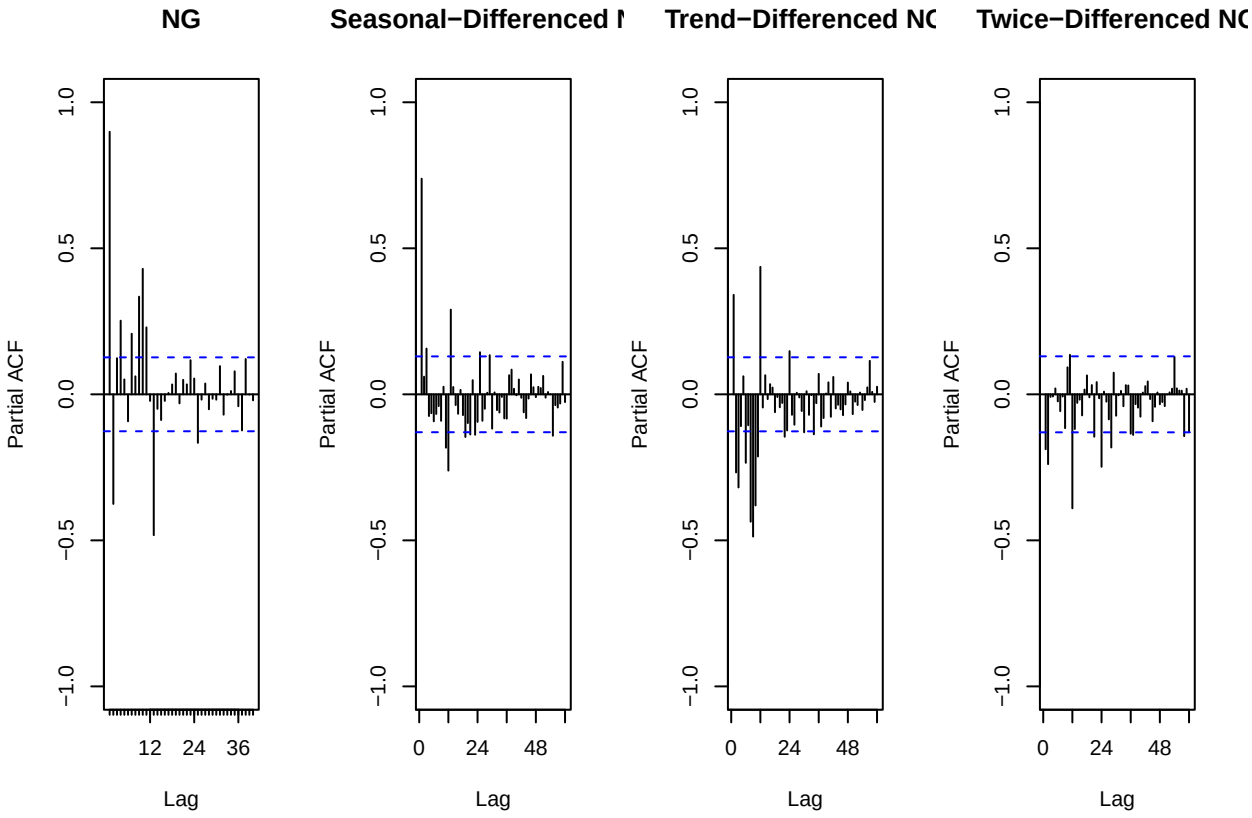
```
## Number of seasonal differencing needed:  1
```

```r
#Lets difference the series once at lag 12 to remove the seasonal trend.
ng_seas_diff <- diff(ng_ts[,"natural_gas"],lag=12, differences=1)
ng_trend_diff <- diff(ng_ts[,"natural_gas"],lag =1, differences=1) #diff
ng_both_diff <- diff(ng_trend_diff,lag =12, differences=1)
```

```r
#Check autocorrelation plots for differenced series
#Comparing ACFs
par(mfrow=c(1,4))
Acf(ng_ts[,"natural_gas"],lag.max=40,main="NG",ylim=c(-1,1))
Acf(ng_seas_diff,lag.max=60,main="Seasonal-Differenced NG",ylim=c(-1,1))
Acf(ng_trend_diff,lag.max=60,main="Trend-Differenced NG",ylim=c(-1,1))
Acf(ng_both_diff,lag.max=60,main="Twice-Differenced NG",ylim=c(-1,1))
```

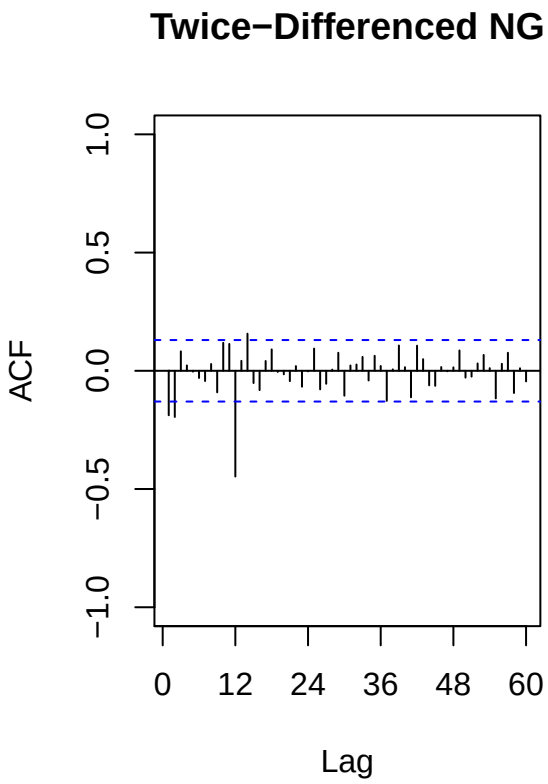


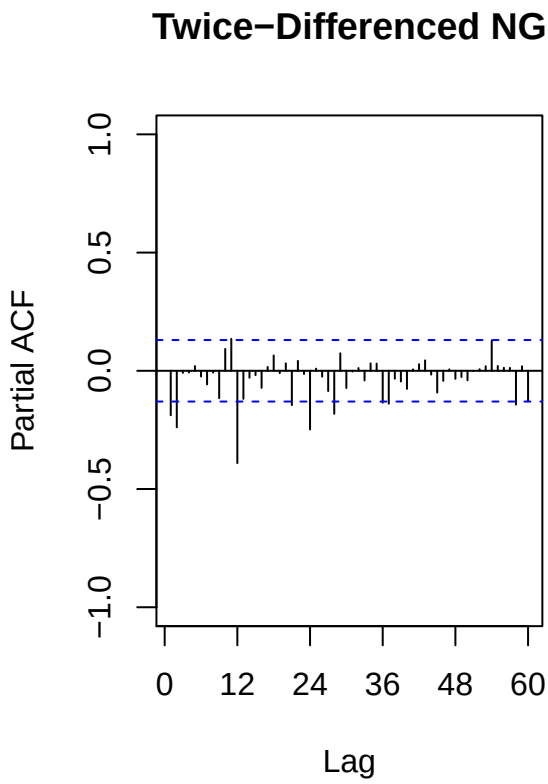

```r
#Comparing PACFs
par(mfrow=c(1,4))
Pacf(ng_ts[,"natural_gas"],lag.max=40,main="NG",ylim=c(-1,1))
Pacf(ng_seas_diff,lag.max=60,main="Seasonal-Differenced NG",ylim=c(-1,1))
Pacf(ng_trend_diff,lag.max=60,main="Trend-Differenced NG",ylim=c(-1,1))
Pacf(ng_both_diff,lag.max=60,main="Twice-Differenced NG",ylim=c(-1,1))
```



We look at the twice-differenced series to identify the model order:

```
par(mfrow=c(1,2))
Acf(ng_both_diff,lag.max=60,main="Twice-Differenced NG",ylim=c(-1,1))
Pacf(ng_both_diff,lag.max=60,main="Twice-Differenced NG",ylim=c(-1,1))
```





We look at the ACF and PACF plot of the differenced series to try to find the order of the model. Here when we look at the first 12 lags for ACF & PACF we don't see slow decays but we do see a cutoff at lag 2 for both plots, indicating ARMA(2, 2), and we know from ndiffs that d=1.

We see that ACF has one spike at 12 and PACF has 2 spikes one at 12 and one at 24. This suggests a seasonal moving average so the order of the seasonal component is P = 0, Q = 1 and D = 1 which we know from the nsdiffs function.

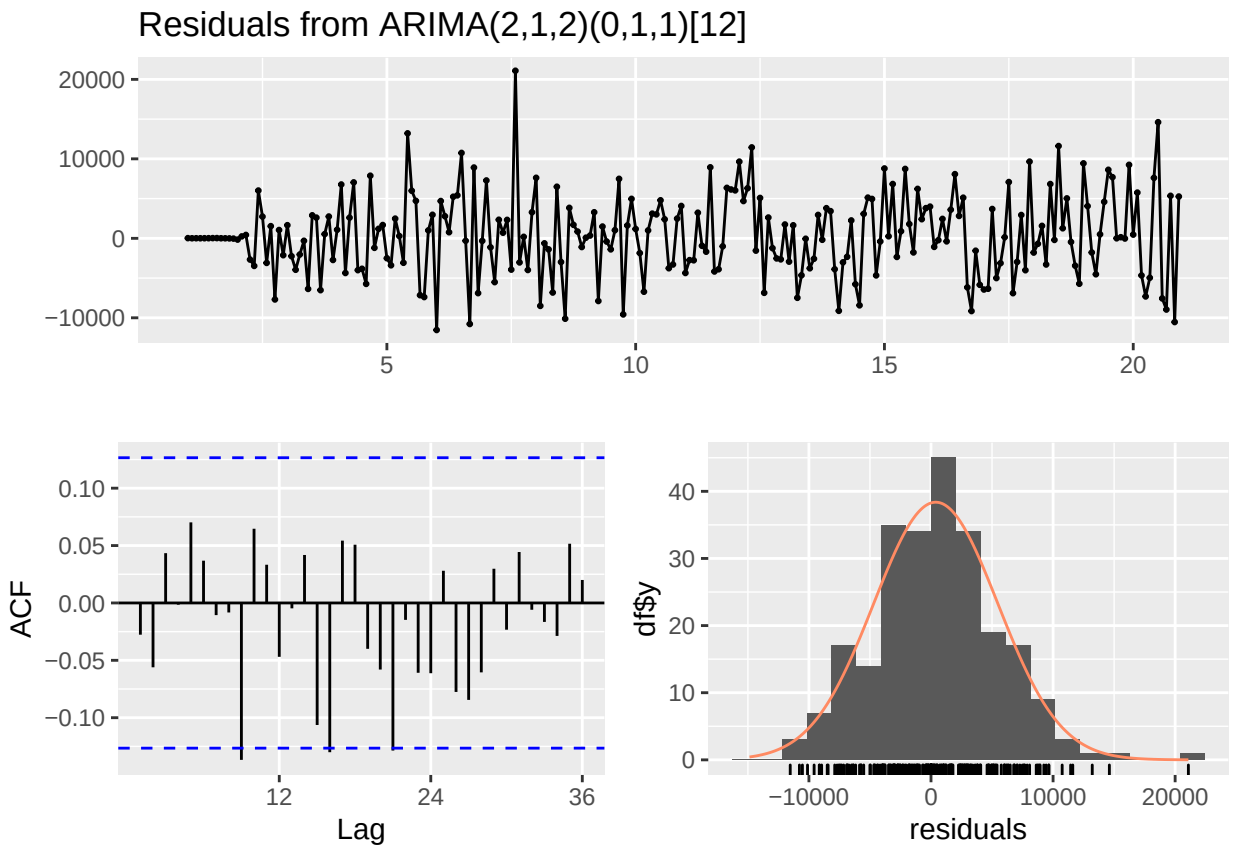
Thus the final model would be ARIMA(2, 1, 2)(0, 1, 1)_[12].

```
arima_111_011 <- Arima(ng_ts[,"natural_gas"],order=c(2,1,2),seasonal=c(0,1,1),
                       include.drift = FALSE)

# print coefficients
print(arima_111_011)
```

```
## Series: ng_ts[, "natural_gas"]
## ARIMA(2,1,2)(0,1,1)[12]
##
## Coefficients:
##           ar1     ar2      ma1      ma2     sma1
##       -0.2162  0.7057  -0.0489  -0.9156  -0.7165
## s.e.   0.0834  0.0648   0.0767   0.0737   0.0563
##
## sigma^2 = 28013060:  log likelihood = -2271.66
## AIC=4555.33   AICc=4555.71   BIC=4575.88
```

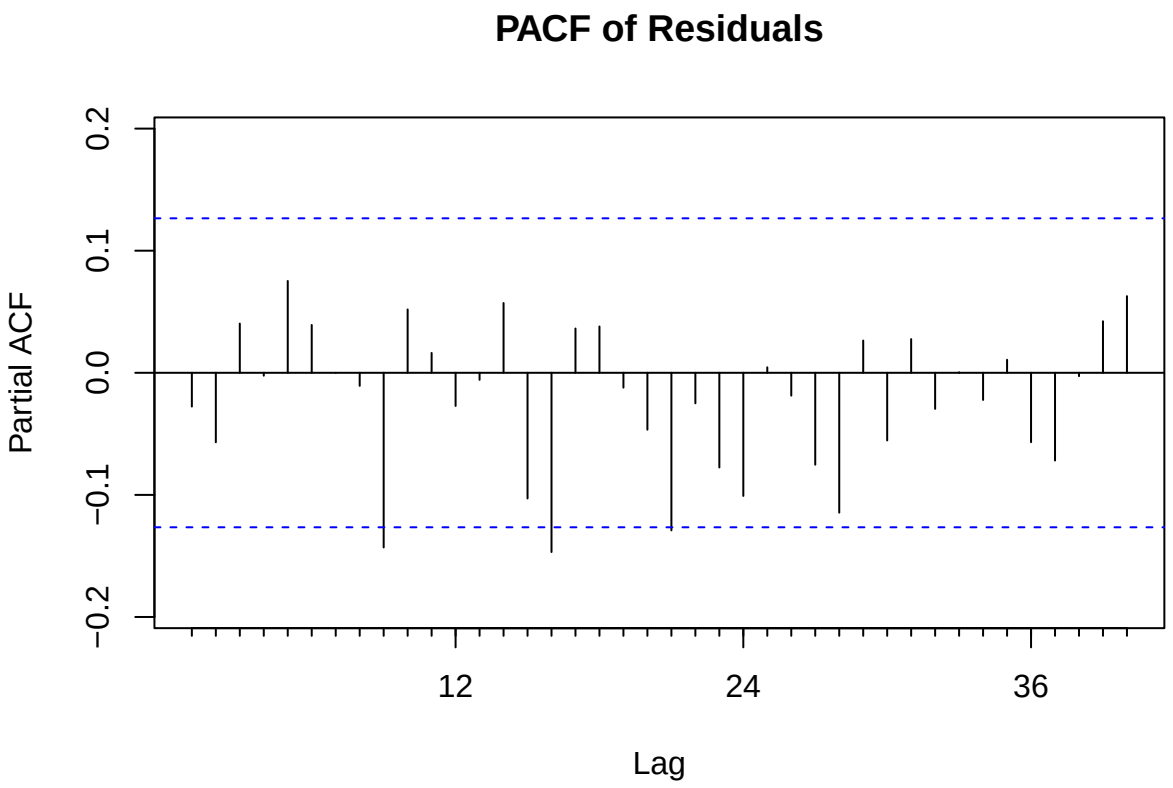
Do the residual series look like a white noise series? Why?

```
checkresiduals(arima_111_011)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,1,2)(0,1,1)[12]
## Q* = 26.571, df = 19, p-value = 0.115
##
## Model df: 5.   Total lags used: 24
```

```
Pacf(arima_111_011$residuals, lag = 40, main = "PACF of Residuals")
```



Yes, the residual series does look relatively like a white noise series. It looks more like a white noise series than the deseasonal model residuals. The ACF plot shows that the residuals are all relatively within the cutoff. They also seem to be normally distributed according to the 3rd plot.

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.

Just from looking at the residuals it is very difficult to tell which ARIMA model is better. However, I would say that the seasonal ARIMA model, ARIMA(2, 1, 2)(0, 1, 1)_[12], seems to be better representing the Natural Gas series. This is because the residuals look more normally distributed and the residual series seems to resemble a white noise series slightly more than the SARIMA model. However, this is not a fair comparison because we are fitting to deseasoned data vs. seasonal data, and they are using different parameters.

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?

```r
autoarima <- auto.arima(deseasonal_ng)
summary(autoarima)
```

```
## Series: deseasonal_ng
## 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 = 26980609:  log likelihood = -2383.11
## AIC=4774.21   AICc=4774.38   BIC=4788.12
##
## Training set error measures:
##                      ME     RMSE     MAE        MPE     MAPE      MASE
## Training set -141.3123 5150.819 3984.38 -0.7171368 4.850437 0.4871225
##                      ACF1
## Training set -0.00301446
```

We see that auto arima on the deseasonalized series chooses an ARIMA(1, 1, 1) model. This matches what I specified in Q4.

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?

```r
autoarima_2 <- auto.arima(ng_ts[, "natural_gas"])
summary(autoarima_2)
```

```
## Series: ng_ts[, "natural_gas"]
## 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 = 27569124:  log likelihood = -2279.54
## AIC=4567.08   AICc=4567.26   BIC=4580.8
##
## Training set error measures:
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -97.32578 5083.901 3950.295 -0.7114711 4.673706 0.4829553
##                      ACF1
## Training set -0.04171074
```

We see that auto arima on the original series chooses an ARIMA(1, 0, 0)(0, 1, 1)*[12] model. This does not match what I specified in Q7, which was ARIMA(2, 1, 2)(0, 1, 1)*[12].