ENV 797 - Time Series Analysis for Energy and Environment Applications | Spring 2024

Assignment 7 - Due date 03/07/24

Samantha Pace

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 file open on your local machine the first thing you will do is rename the file such that it includes your first and last name (e.g., "LuanaLima_TSA_A07_Sp24.Rmd"). Then change "Student Name" on line 4 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. Submit this pdf using Sakai.

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

Set up

```
#Load/install required package here
#install.packages("forecast")
#install.packages("tseries")
#install.packages("lubridate")
#install.packages("tidyverse")
#nstall.packages("Kendall")
#nstall.packages("tinytex")

library(forecast)
```

```
## Warning: package 'forecast' was built under R version 4.3.3
```

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

```
library(tseries)
```

```
## Warning: package 'tseries' was built under R version 4.3.3
```

```
library(lubridate)
```

```
## Warning: package 'lubridate' was built under R version 4.3.3
```

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

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

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

```
## Warning: package 'tidyverse' was built under R version 4.3.3

## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr   1.1.4     v stringr 1.5.1
## v forcats 1.0.0     v tibble  3.2.1
## v purrr   1.0.2     v tidyr   1.3.1
## v readr   2.1.5

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(Kendall)
library(tinytex)
```

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**.

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.

```
# working directory
getwd()
```

```
## [1] "C:/Users/saman/OneDrive/Desktop/Duke Spring 24/GITHUB/TSA_Sp24"
```

```
# import data
generation_data <-
  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.
```

```
# select the month and nat gas columns, fix date, arrange by month
generation_data_ng <-
  generation_data %>%
  select(1,4) %>%
```

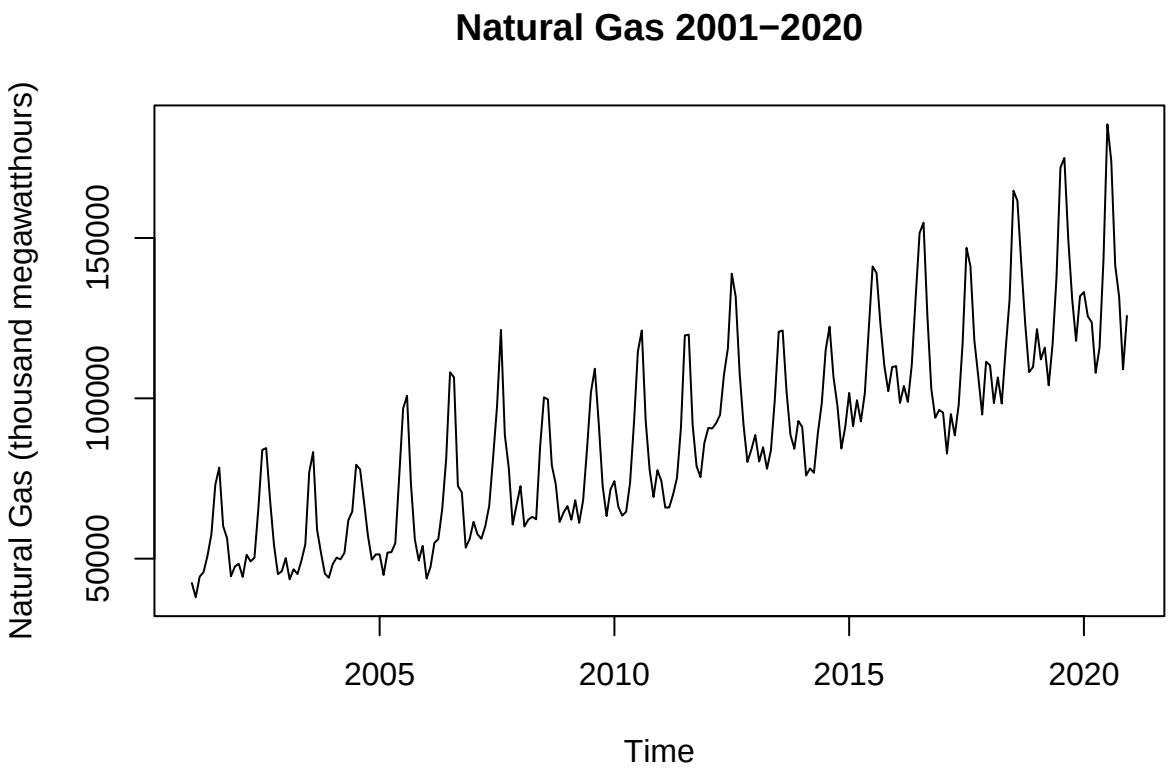
```r
  mutate(Month = my(Month))

generation_data_ng <- generation_data_ng %>%
  arrange(generation_data_ng$Month)

# create time series object
natgas_ts <- ts(generation_data_ng$`natural gas thousand megawatthours`,
                start = c(year(generation_data_ng$Month[1]),
                          month(generation_data_ng$Month[1])),
                frequency = 12)

head(natgas_ts, 15)
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
## 2001 42388.66 37966.93 44364.41 45842.75 50934.21 57603.15 73030.14 78409.80
## 2002 48412.83 44308.43 51214.46
##           Sep      Oct      Nov      Dec
## 2001 60181.14 56376.44 44490.62 47540.86
## 2002
```

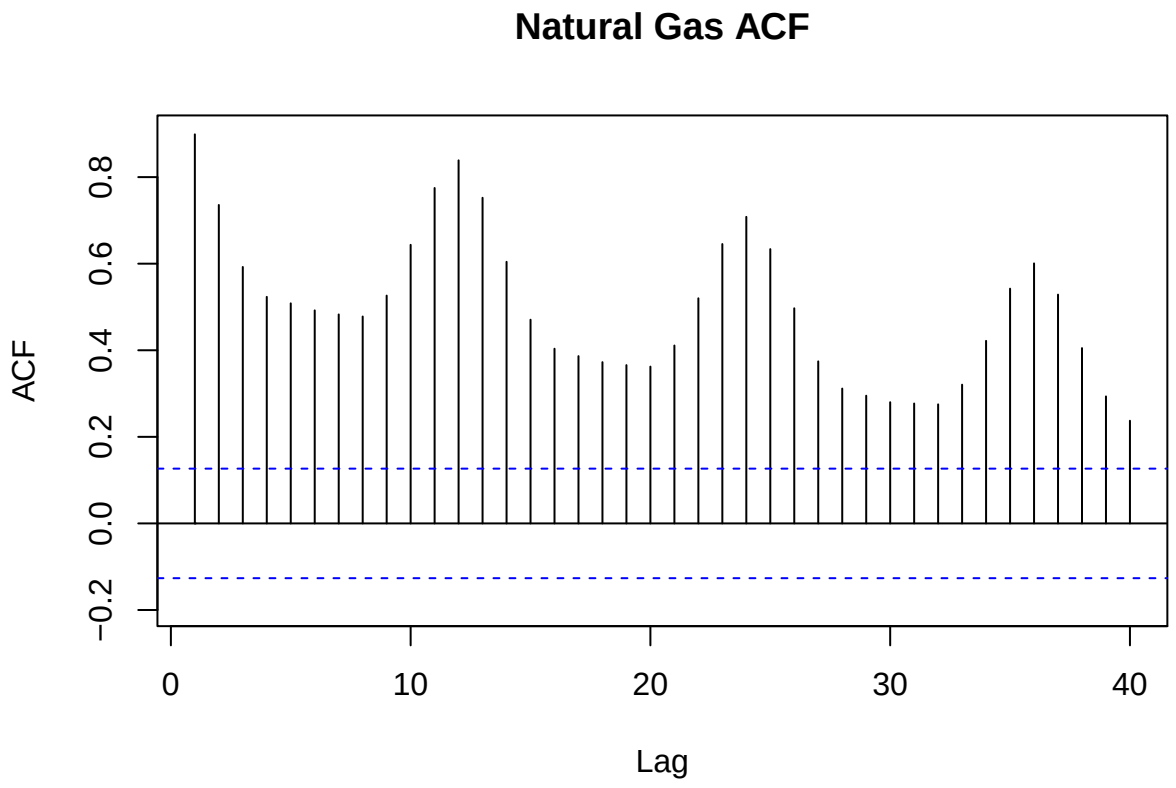
```r
tail(natgas_ts, 15)
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
## 2019
## 2020 133157.6 125593.9 123697.0 107960.0 115870.9 143245.4 185444.8 173926.6
##           Sep      Oct      Nov      Dec
## 2019          130947.6 117910.5 131838.9
## 2020 141452.7 131658.2 109037.2 125703.7
```

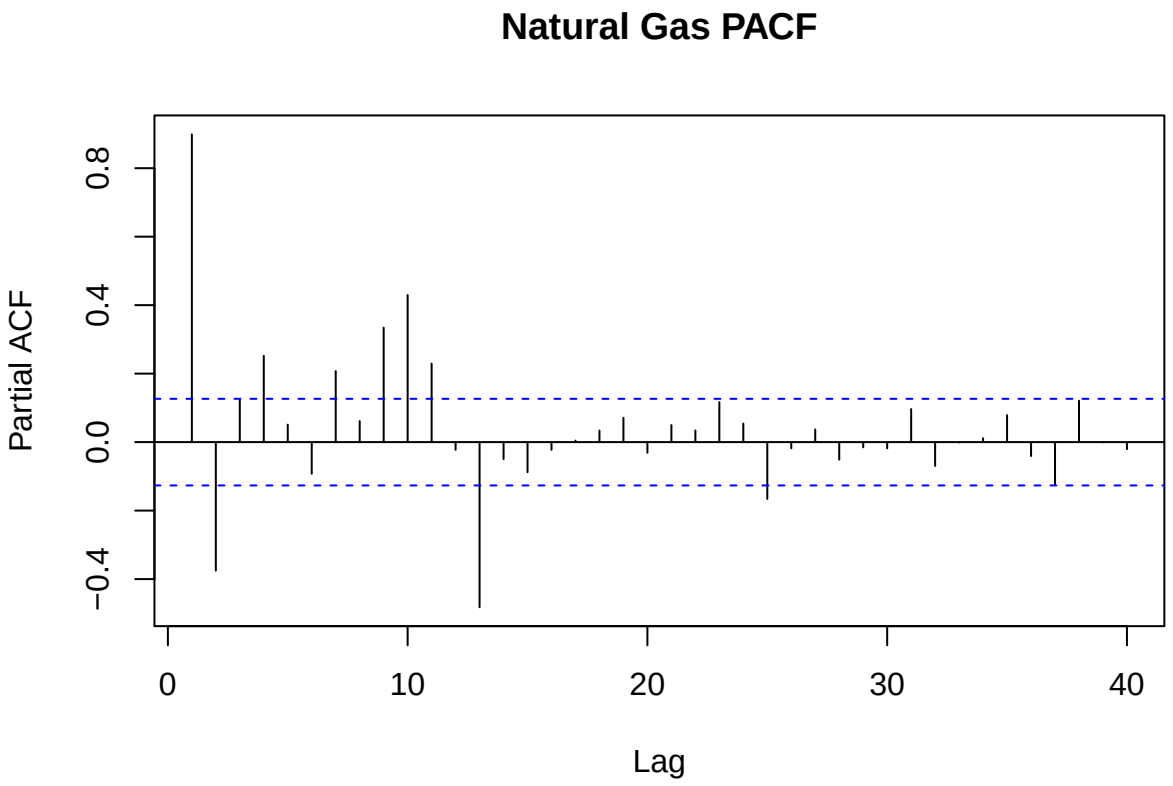
```r
# plot ts
plot(natgas_ts,
ylab="Natural Gas (thousand megawatthours)",
main = "Natural Gas 2001-2020")
```




```
#ACF
ACF_plot <- Acf(generation_data_ng$`natural gas thousand megawatthours`,
                lag = 40, plot = TRUE, main = "Natural Gas ACF")
```



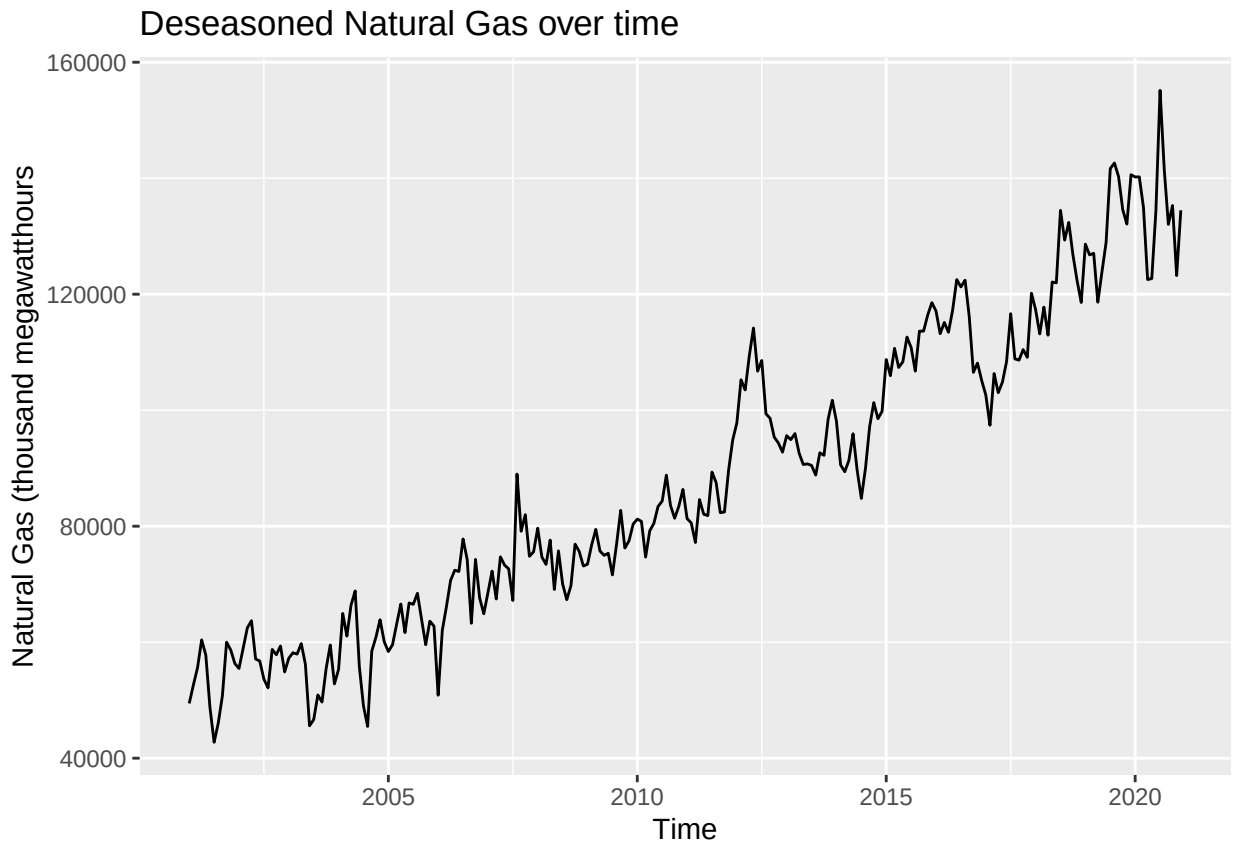

```
# PACF
PACF_plot <-Pacf(generation_data_ng$`natural gas thousand megawatthours`,
                 lag = 40, plot = TRUE, main = "Natural Gas PACF")
```



Q2

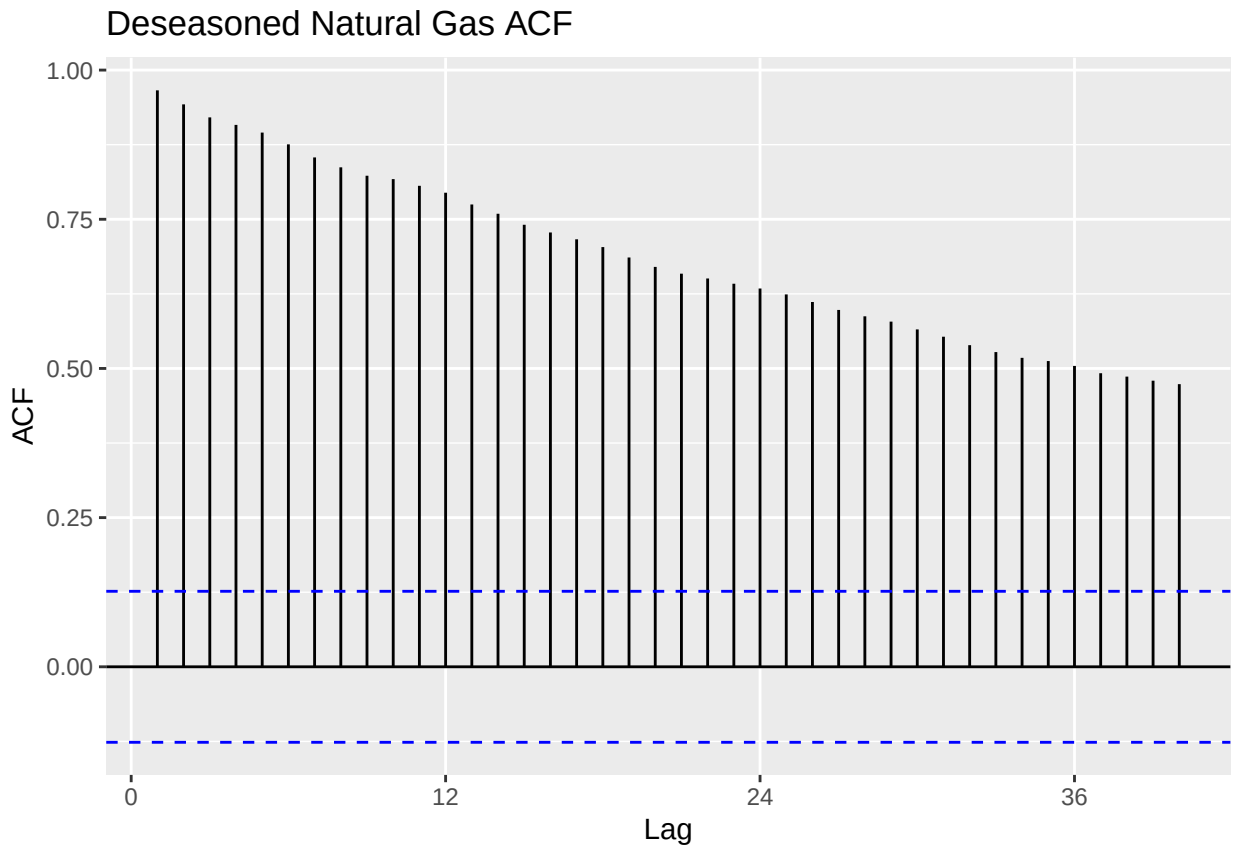
Using the *decompose*() or *stl*() and the *seasadj*() functions create a series without the seasonal component, i.e., a deseasonalized natural gas series. Plot the deseasonalized series over time and corresponding ACF and PACF. Compare with the plots obtained in Q1.

```
# decomposing and seasadj
decompose_natgas <-decompose(natgas_ts, "additive")
deseasonal_natgas <- seasadj(decompose_natgas)

#plot
autoplot(deseasonal_natgas,
         main = "Deseasoned Natural Gas over time",
         ylab = "Natural Gas (thousand megawatthours")
```



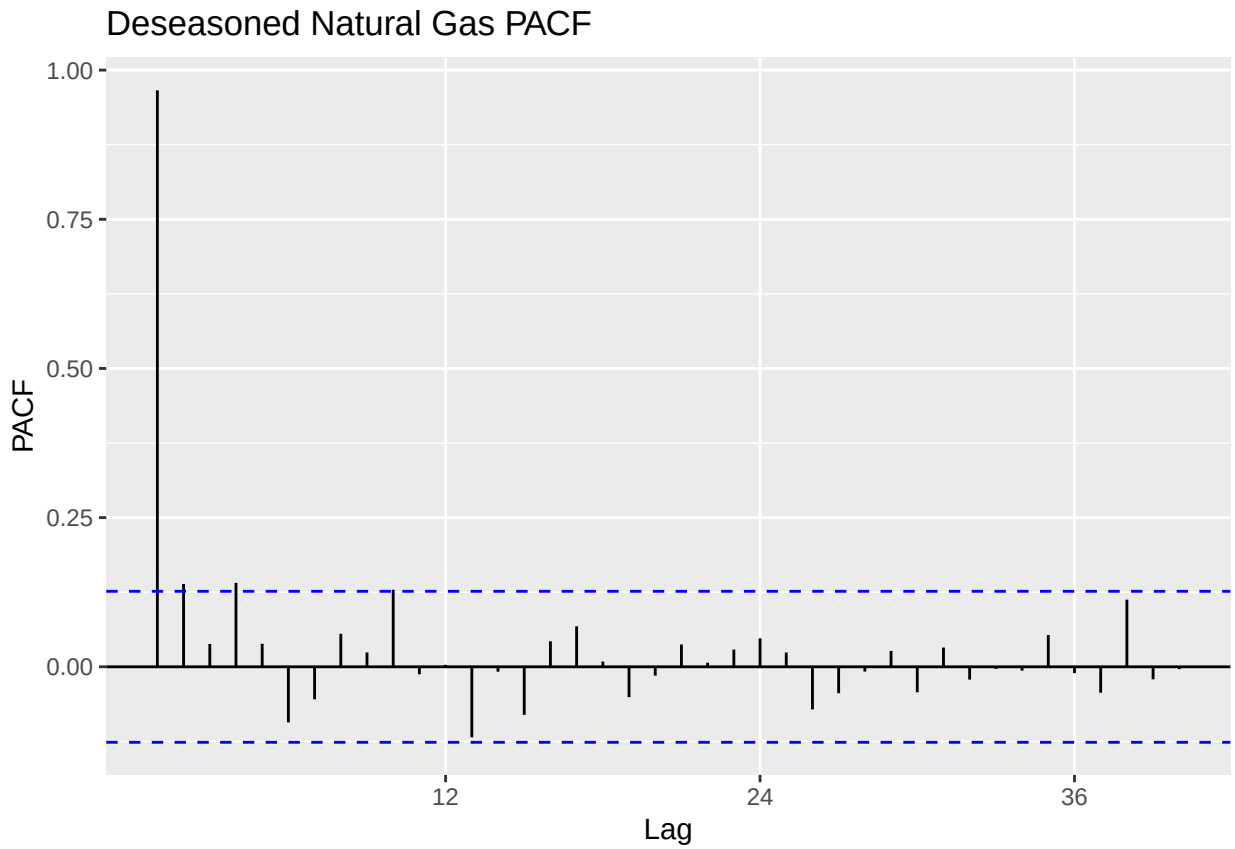

```
# ACF
autoplot(Acf(deseasonal_natgas, lag = 40, plot = F),
                main = "Deseasoned Natural Gas ACF")
```

```
## Warning in ggplot2::geom_segment(lineend = "butt", ...): Ignoring unknown
## parameters: `main`
```




```
#PACF
autoplot(Pacf(deseasonal_natgas, lag = 40, plot =F),
                main = "Deseasoned Natural Gas PACF")
```

```
## Warning in ggplot2::geom_segment(lineend = "butt", ...): Ignoring unknown
## parameters: `main`
```



Answer: The original data series showed a clear seasonal pattern with regular wave-like patterns; and this was gone from the deseasoned data set. The deseasoned data set still has a clear upward trend, and now seemingly random variation of movements along the trend rather than a wave-like pattern. The trend was clear through the original data's ACF, with declining peaks around lags 12, 24, and 36. The ACF for the deseasoned data shows no waves; only decay over time but with strong dependence on time. The PACF for the original data showed there were significant coefficients around lag 10-12. For the deseasoned PACF, aside from lag 1, nearly all the coefficients are within bounds of being insignificant, showing an effective elimination of the seasonal component.

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.

```
# ADF test: unit root/stochastic trend
print(adf.test(deseasonal_natgas))
```

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

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

```r
# Mann-Kendall Test
MKtest <- MannKendall(deseasonal_natgas)
print(summary(MKtest))
```

```
## Score =  24186 , Var(Score) = 1545533
## denominator =  28680
## tau = 0.843, 2-sided pvalue =< 2.22e-16
## NULL
```

Answer: For the Mann Kendall test, the test statistic is the tau value, which is 0.843, and has a p-value of less than 0.05, which means the null hypothesis that there is no trend. Since we can reject this, the Mann Kendall Test indicates there is a trend present in the data; and the s value is positive, so we expect a positive trend. The ADF is testing for a unit root, and the p-value is less than 0.05, so can reject the null hypothesis and conclude that the deseasoned data is stationary relative to the unit root.

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 understand the ACF/PACF plots and interpret the test results.

Answer: Because the ACF of the deseasoned data shows a slow decay and the PACF shows a cut off at lag 1, I believe this suggests it will use an AR model. The p model parameter corresponds to the order of the AR, which based on the PACF, is 1. Based on the ACF and PACF, it doesn't look like there will be an MA order, so q=0. While the ADF showed that there isn't a stochastic trend, there is still a significant upward trend that is visually notable in the deseasoned time series, and the MK test aligns with the finding that there is a trend, so I will identify the differencing parameter, d, to be 1. Using the ndiffs() function to determine how many times to difference the series would be function in R that I might use to determine the differencing parameter.

in summary: p = 1, d = 1, q = 0. (1, 1, 0)

Q5

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

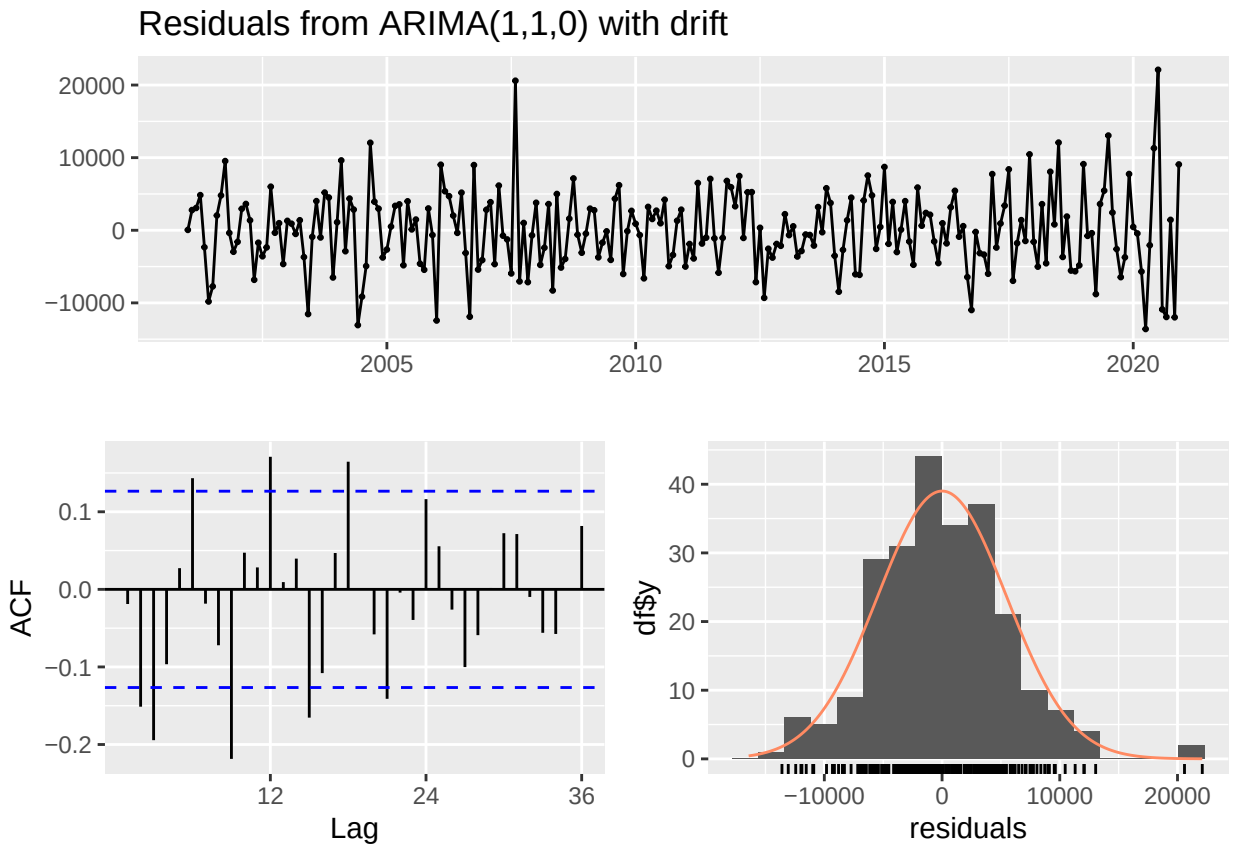
```r
# arima (1,1,0)
Model_110 <- Arima(deseasonal_natgas, order = c(1,1,0),
                   include.drift=TRUE)
print(Model_110)
```

```
## Series: deseasonal_natgas
## ARIMA(1,1,0) with drift
##
## Coefficients:
##           ar1     drift
##       -0.1479  348.3927
## s.e.   0.0644  308.8385
##
## sigma^2 = 30254066:  log likelihood = -2396.54
## AIC=4799.07   AICc=4799.18   BIC=4809.5
```

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?

```
# check residuals
checkresiduals(Model_110)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,0) with drift
## Q* = 72.475, df = 23, p-value = 5.015e-07
##
## Model df: 1.   Total lags used: 24
```

Answer: The residuals look to be fairly centered around the mean, although it looks like there may be a couple of outliers, particularly on the positive side of the mean. The ACF shows there are a few coefficients that are just outside the boundaries of significance, which may be something to improve, this may mean there is still more autocorrelation to model. The residuals look to be fairly normally distributed based on the histogram, but there is note of the higher end potential outliers here as well. In addition to the couple potential outliers, there looks to also be a wave-like pattern with increasing magnitude from 2017 through 2021 that is centered around the mean. Considering these components, the residuals look to be otherwise white noise.

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.

```
# ADF test: unit root/stochastic trend
print(adf.test(natgas_ts))
```

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

```
##
##  Augmented Dickey-Fuller Test
##
## data:  natgas_ts
## Dickey-Fuller = -8.9602, Lag order = 6, p-value = 0.01
## alternative hypothesis: stationary
```

```
# Mann-Kendall Test
MKtest_complete <- MannKendall(natgas_ts)
print(summary(MKtest_complete))
```

```
## Score =  18658 , Var(Score) = 1545533
## denominator =  28680
## tau = 0.651, 2-sided pvalue =< 2.22e-16
## NULL
```

Answer: The time series plot over time shows a clear seasonal component and upward trend. The ACF plot shows peaks or spikes at the seasonal lags, suggesting that a seasonal moving average component (Q) of 1 would be suitable. Besides the peaks at the lags, there is general decay in the ACF, which is indicative of an AR model, so I will set p=1. The residuals of the deseasoned data didn't fully reflect that the trend/seasonality was fully eliminated, so I will set both d and D to be 1 for each. Also, the ADF and MK tests suggest a trend present.

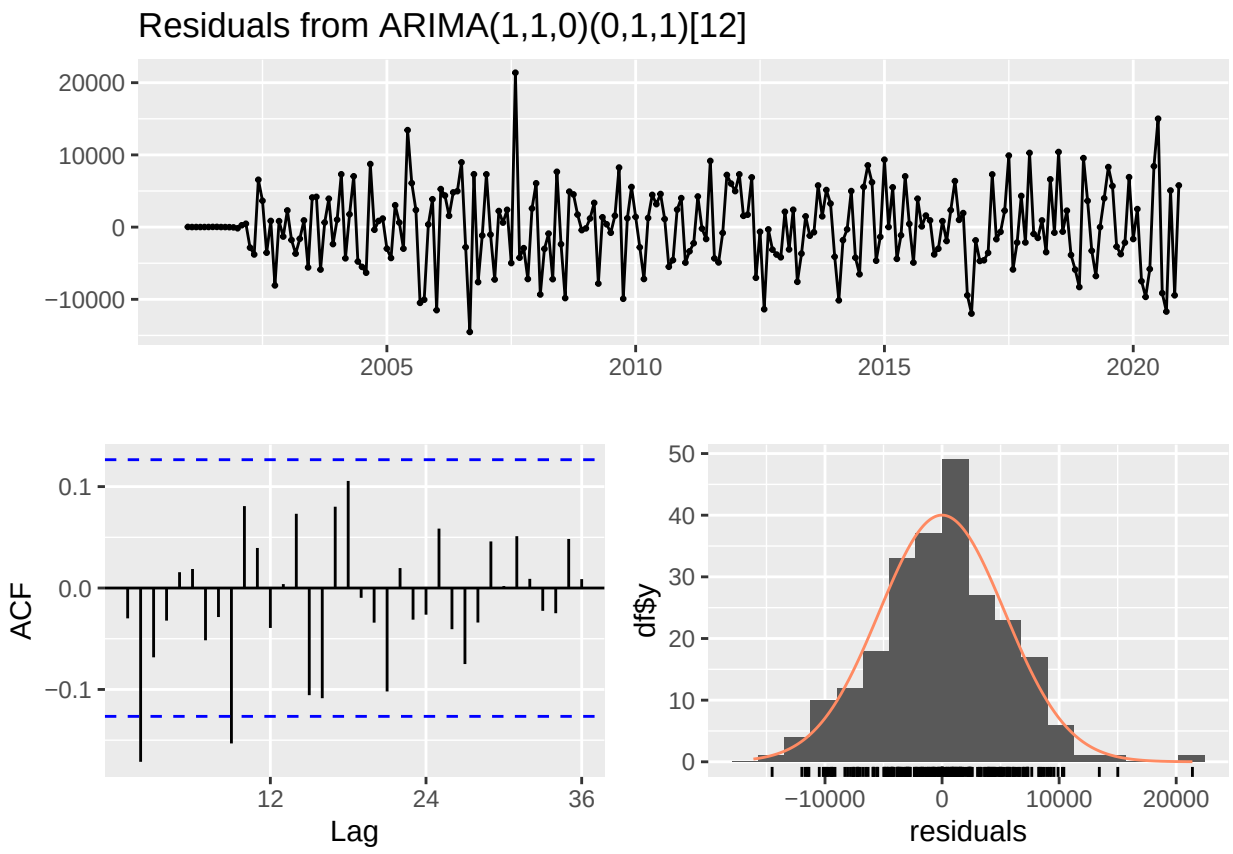
The plots do not reflect a non-seasonal moving average part from my interpretation, so that will remain 0. Since the sum of P and Q shouldn't be more than 1 and Q will be equal to 1, therefore P will be set to 0.

In summary, the model will be (1,1,0)(0,1,1)

```
# arima function
Model_110_011 <- Arima(natgas_ts, order = c(1,1,0),
                       seasonal = c(0,1,1),
                       include.mean = TRUE)
print(Model_110_011)
```

```
## Series: natgas_ts
## ARIMA(1,1,0)(0,1,1)[12]
##
## Coefficients:
##           ar1     sma1
##       -0.1808  -0.6898
## s.e.   0.0655   0.0557
##
## sigma^2 = 30626308:  log likelihood = -2281.43
## AIC=4568.86   AICc=4568.96   BIC=4579.13
```

```
# check residuals
checkresiduals(Model_110_011)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,0)(0,1,1)[12]
## Q* = 33.681, df = 22, p-value = 0.05291
##
## Model df: 2.   Total lags used: 24
```

Answer: The residuals of this fitted arima model look similar to those of the deseasoned fitted model in Q6. The residuals look to be centered around the mean, with some potential outliers still. The ACF plot shows a couple of lines that may indicate still some time dependence, but not much. Overall, this does look a little more like white noise than the previous residuals.

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.

Answer: These residual series are fairly similar. However, I can't tell which ARIMA model is better at representing the Natural Gas Series. It is not a fair comparison because one models and accounts for the seasonality and seasonal parameters while the other model does not. As a result, the models are effectively fitting to two different data sets and it is not a fair comparison then.

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 same order as the *auto.arima*().

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?

```
# auto.arima on deseason data
auto_Model <- auto.arima(deseasonal_natgas)
print(auto_Model)
```

```
## Series: deseasonal_natgas
## 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
```

Answer: The order of the best arima model based on the auto.arima function was (1, 1, 1). What I specified in Q4 was (1,1,0), which is very close but missing the MA order of 1.

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?

```
# auto.arima on original series
auto_orig_data <- auto.arima(natgas_ts)
print(auto_orig_data)
```

```
## Series: natgas_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 = 27569124:  log likelihood = -2279.54
## AIC=4567.08   AICc=4567.26   BIC=4580.8
```

Answer: R specified ARIMA(1,0,0)(0,1,1) as the best model. I specified ARIMA(1,1,0)(0,1,1), which had an additional non-seasonal differencing component. They were close but not quite the same.