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

Assignment 6 - Due date 03/25/22

Ben Joseph

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  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.1     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()
```

```
library(smooth)
```

```
## Loading required package: greybox

## Package "greybox", v1.0.4 loaded.

##
## Attaching package: 'greybox'

## The following object is masked from 'package:tidyr':
##
##     spread

## The following object is masked from 'package:forecast':
##
##     forecast

## The following object is masked from 'package:lubridate':
##
##     hm

## This is package "smooth", v3.1.5
```

```
#install.packages("sarima")
library(sarima)
```

```
## Loading required package: stats4

##
## Attaching package: 'sarima'

## The following object is masked from 'package:stats':
##
##     spectrum
```

```
library(readxl)
```

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.

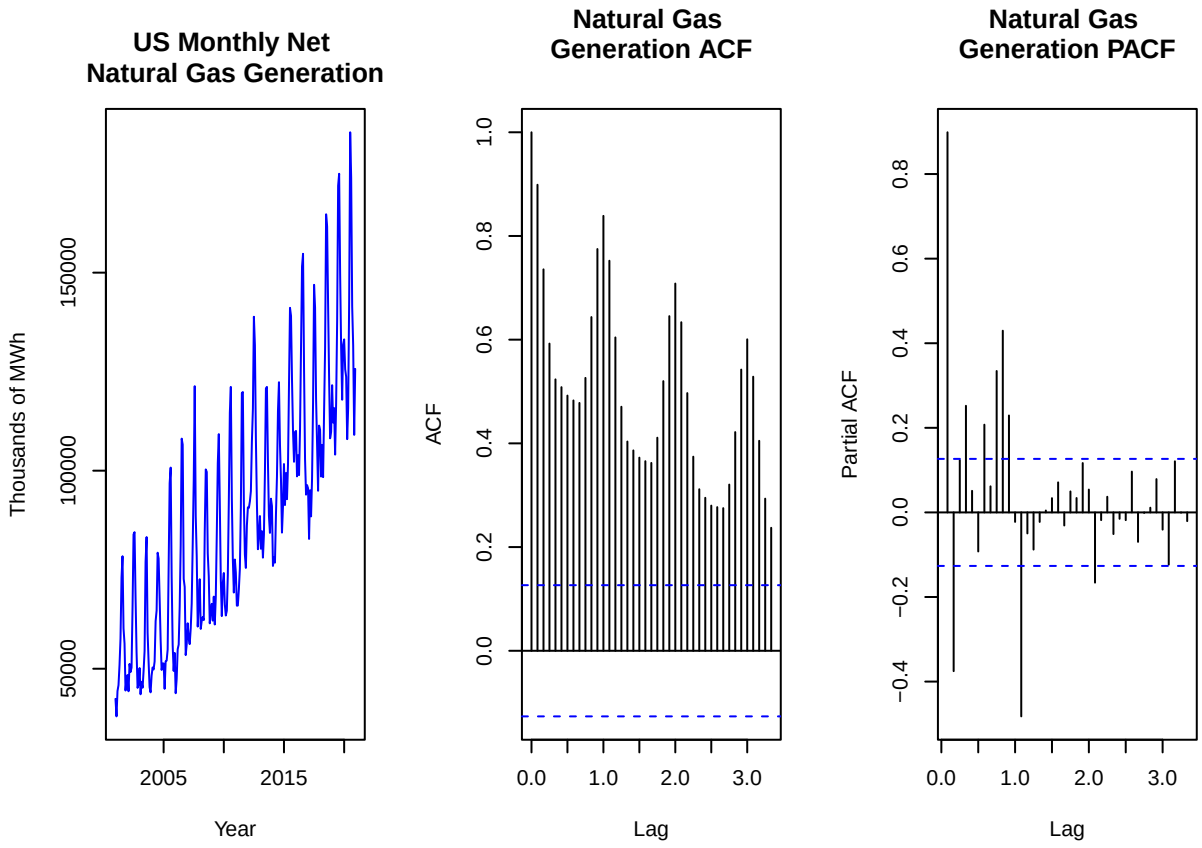
```
df <- read_csv("./Data/Net_generation_United_States_all_sectors_monthly.csv"
   , col_names = TRUE, skip =4, na="Not Available")
```

```
## 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.
```

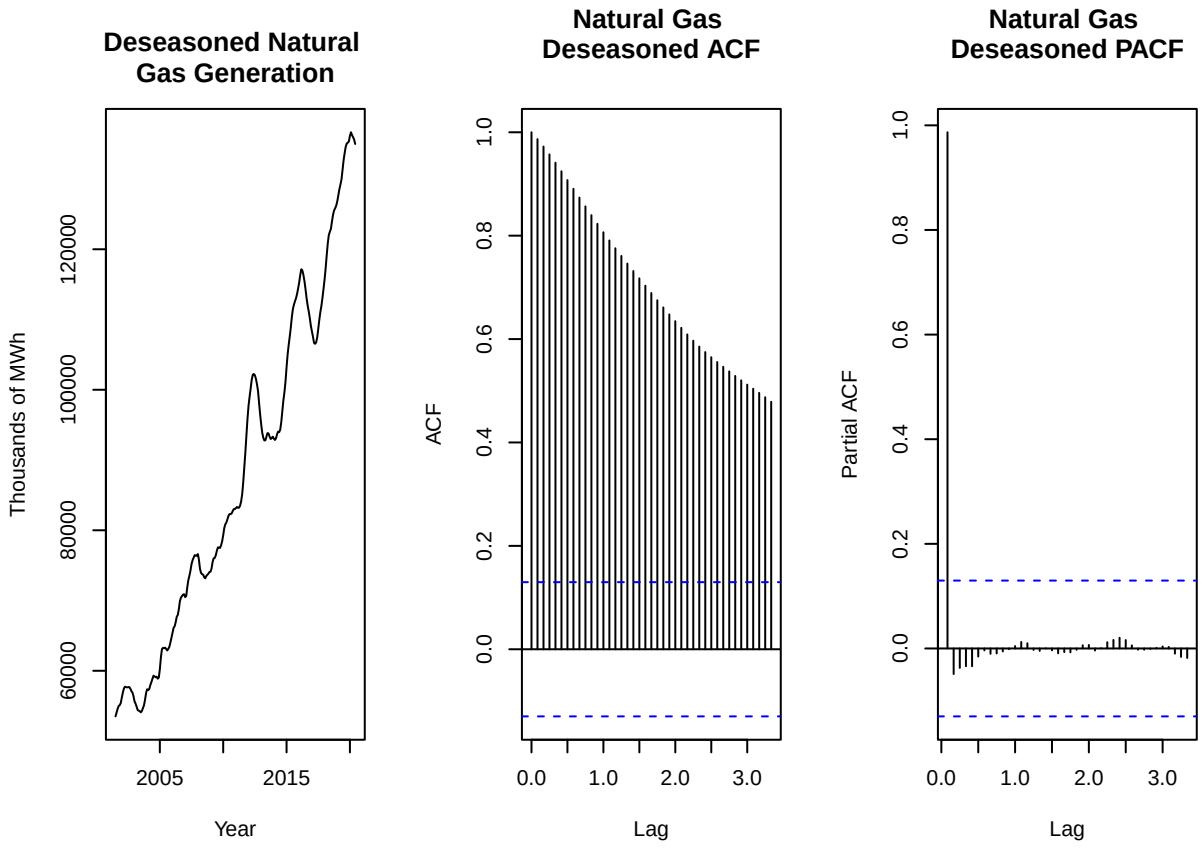
```
df2 <- df[order(nrow(df):1),] ## Reverse Rows
ts <- ts(data = df2$`natural gas thousand megawatthours`, start = 2001, frequency = 12)
par(mfrow=c(1,3))
plot(ts, col="blue", ylab="Thousands of MWh", main="US Monthly Net \nNatural Gas Generation",
     xlab="Year")
acfNG <- acf(ts, lag.max = 40, type = "correlation",plot=TRUE, main="Natural Gas \nGeneration ACF")
pacfNG <- pacf(ts, lag.max = 40, plot = T, main="Natural Gas \nGeneration 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.

```
ts_decomposed_additive <- decompose(ts,"additive")
ts_decomposed_trend <- na.omit(ts_decomposed_additive$trend)
par(mfrow=c(1,3))
plot(ts_decomposed_trend, main="Deseasoned Natural \nGas Generation", xlab = "Year", ylab = "Thousands
acfNG_deseas <- acf(ts_decomposed_trend, lag.max = 40, type = "correlation",plot=TRUE, main="Natural Ga
pacfNG <- pacf(ts_decomposed_trend, lag.max = 40, plot = T, main="Natural Gas \nDeseasoned PACF")
```



Answer: Deseasoning the time series and plotting the trend took out all seasonality from the ACF and PACF, and made all PACF values very close to zero.

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.

```
#seasonal mann kendall
MKtest <- MannKendall(ts_decomposed_trend)
print("Results for Mann Kendall")
```

```
## [1] "Results for Mann Kendall"
```

```
print(summary(MKtest))
```

```
## Score =  23468 , Var(Score) = 1325529
## denominator =  25878
## tau = 0.907, 2-sided pvalue =< 2.22e-16
## NULL
```

```r
#augmented dickey fuller
ADFtest <- adf.test(ts_decomposed_trend, alternative = "stationary")
print("Results for ADF test")
```

```
## [1] "Results for ADF test"
```

```r
print(adf.test(ts_decomposed_trend,alternative = "stationary"))
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ts_decomposed_trend
## Dickey-Fuller = -3.8176, Lag order = 6, p-value = 0.01896
## alternative hypothesis: stationary
```

> Answer: The Mann Kendall test has a very small p-value, below the standard significance level of .05. This low p value indicates a rejection of the null hypothesis that the deseasoned natural gas generation in stationary, in support of it following a deterministic trend.
>
> The ADF test signifies that the time series is stationary and doesn't follow a stochastic trend because it has a phi value of -3.8176 that is less than 1.

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: Since the ACF decays slowly, i think it's an AR process and since the PACF cuts off after 1 lag, i'm going to use p=1 and q=0. Since the Mann Kendall test returned a deterministic trend, i use 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.

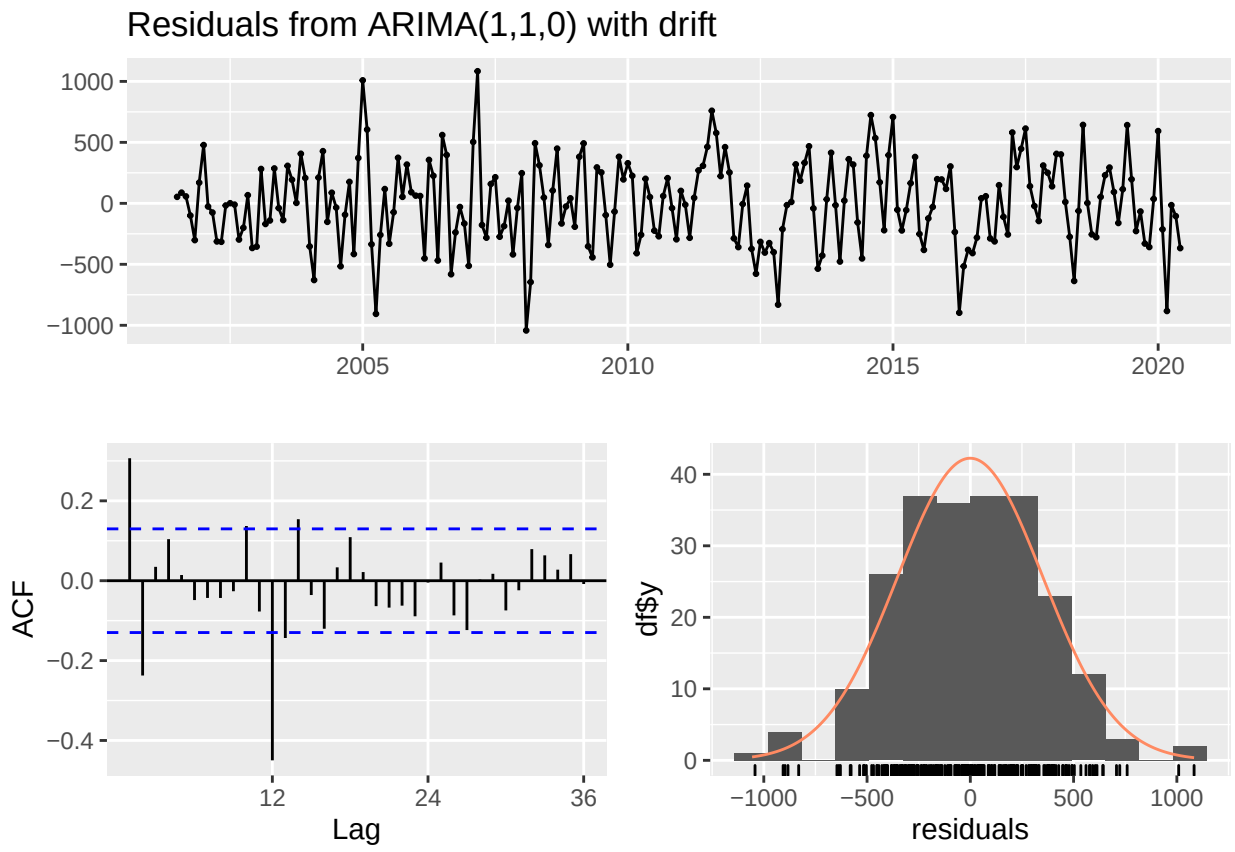
```r
ng_deseas_arima <- Arima(ts_decomposed_trend,order=c(1,1,0),include.drift=TRUE)
print(ng_deseas_arima)
```

```
## Series: ts_decomposed_trend
## ARIMA(1,1,0) with drift
##
## Coefficients:
##          ar1    drift
##       0.8781  332.034
## s.e.  0.0313  186.037
##
## sigma^2 = 125070:  log likelihood = -1653.93
## AIC=3313.85   AICc=3313.96   BIC=3324.13
```

Q6

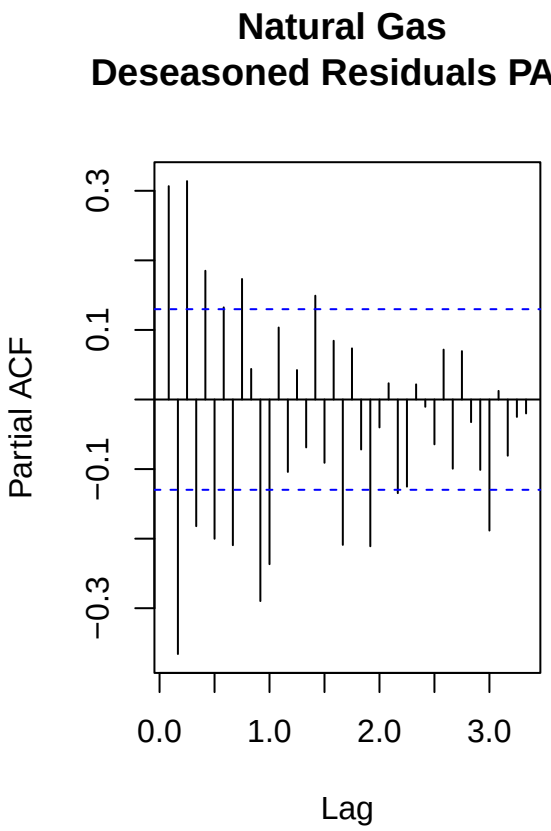
Now plot the residuals of the ARIMA fit from Q5 along with residuals ACF and PACF on the same window. You may use the *checkresiduals*() function to automatically generate the three plots. Do the residual series look like a white noise series? Why?

```
par(mfrow=c(1,2))
checkresiduals(ng_deseas_arima)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,0) with drift
## Q* = 117.6, df = 22, p-value = 4.774e-15
##
## Model df: 2.   Total lags used: 24
```

```
pacfResid <- pacf(ng_deseas_arima$residuals, lag.max = 40, plot = T, main="Natural Gas \nDeseasoned Resi
```



Answer: the residuals look like white noise because there is no obvious trend in the residuals and the ACf is more or less insignificant. There are a few lags that stray outside the band of significance, but i think they are random enough to ignore. Also the residuals are normally distributed. The PACF is slightly concerning that many of the lags seem significant. They do seem to trail off a little bit which may indicate a moving average term. If i were to guess it would be the order of 2 because the ACF sort of cuts off after 2.

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.

Answer: Since the seasonal component ACF decays slowly, i think it's an AR process and since the PACF cuts off after about lag 3, i'm going to use P=3 and Q=0. Since the Mann Kendall test returned a deterministic trend, i use D=1.

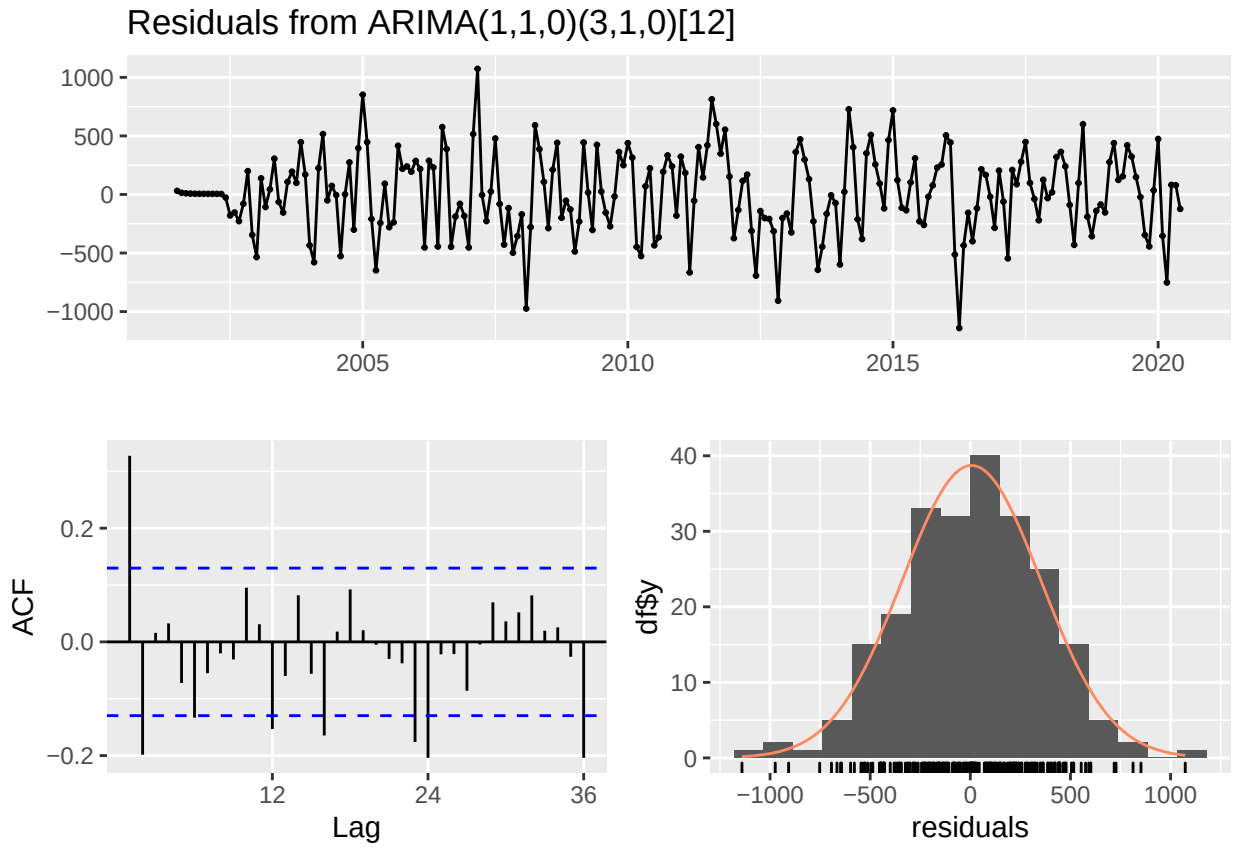
```
ng_deseas_sarima <- Arima(ts_decomposed_trend,order=c(1,1,0),seasonal=c(3,1,0),include.drift=TRUE)
```

```
## Warning in Arima(ts_decomposed_trend, order = c(1, 1, 0), seasonal = c(3, : No
## drift term fitted as the order of difference is 2 or more.
```

```
print(ng_deseas_sarima)
```

```
## Series: ts_decomposed_trend
## ARIMA(1,1,0)(3,1,0)[12]
##
## Coefficients:
##          ar1     sar1     sar2     sar3
##       0.8756  -1.1732  -0.9041  -0.4425
## s.e.  0.0338   0.0612   0.0801   0.0628
##
## sigma^2 = 129371:  log likelihood = -1579.06
## AIC=3168.12   AICc=3168.41   BIC=3184.98
```

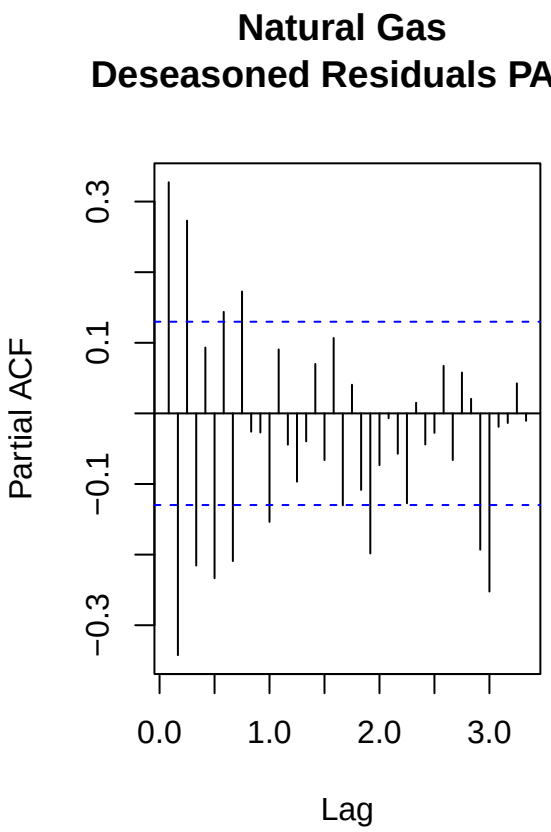
```
par(mfrow=c(1,2))
checkresiduals(ng_deseas_sarima)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,0)(3,1,0)[12]
## Q* = 80.387, df = 20, p-value = 3.374e-09
```

```
##
## Model df: 4.   Total lags used: 24
```

```
pacfResidSarima <- pacf(ng_deseas_sarima$residuals, lag.max = 40, plot = T, main="Natural Gas \nDeseaso
```



Answer: The results looked fairly similar. I cant tell any bias or triend in the residuals, it's mean appears to be zero. The residuals are normally distributed and most of the lags in the ACF are insigificant after 1. I still see the same issue in the PACF where there seems to be a lot of lags that are significant. Also, there seems to be a lot of significance in the lags with multiples of 12 which may indicate it requires more differencing.

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.

```
print(ng_deseas_arima)
```

```
## Series: ts_decomposed_trend
## ARIMA(1,1,0) with drift
##
## Coefficients:
##          ar1    drift
##       0.8781  332.034
```

```
## s.e.  0.0313  186.037
##
## sigma^2 = 125070:  log likelihood = -1653.93
## AIC=3313.85   AICc=3313.96   BIC=3324.13
```

```
print(ng_deseas_sarima)
```

```
## Series: ts_decomposed_trend
## ARIMA(1,1,0)(3,1,0)[12]
##
## Coefficients:
##          ar1     sar1     sar2     sar3
##       0.8756  -1.1732  -0.9041  -0.4425
## s.e.  0.0338   0.0612   0.0801   0.0628
##
## sigma^2 = 129371:  log likelihood = -1579.06
## AIC=3168.12   AICc=3168.41   BIC=3184.98
```

> Answer: the AIC for the arima was 3313.85 and the AIC for the SARIMA was 3168. It is impossible to compare these numbers though because they are different models with different data sets.

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?

```
auto.arima(ts_decomposed_trend)
```

```
## Series: ts_decomposed_trend
## ARIMA(2,1,2)(1,0,1)[12] with drift
##
## Coefficients:
##          ar1      ar2     ma1      ma2     sar1     sma1     drift
##       1.2171  -0.3237  0.4512  -0.5050  -0.0193  -0.7019  358.0152
## s.e.  0.4071   0.3153  0.3930   0.3747   0.0947   0.0768   42.2676
##
## sigma^2 = 49062:  log likelihood = -1551.62
## AIC=3119.23   AICc=3119.89   BIC=3146.63
```

> Answer: The auto.arima function produced an aroma of ARIMA(2,1,2)(1,0,1). My autoregressive order was off by 1. My differencing order was correct. My assumption of zero for the moving average term was incorrect but my analysis after examining the ACF and PACF of the residuals was correct. I thought there might be a moving average term of order 2.

Q10