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

Assignment 6 - Due date 03/26/21

Benjamin Culberson

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(lubridate)
```

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

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

```
library(ggplot2)
library(Kendall)
library(outliers)
library(tidyverse)
```

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

```
## v tibble  3.0.5     v dplyr   1.0.3
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0
## 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", v0.6.8 loaded.

##
## Attaching package: 'greybox'

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

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

## This is package "smooth", v3.1.0
```

```
#New package for M9 to assist with tables
#install.packages("kableExtra")
library(kableExtra)
```

```
##
## Attaching package: 'kableExtra'

## The following object is masked from 'package:dplyr':
##
##     group_rows
```

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.

```
net_generation <- read.csv(
  file="../Data/Net_generation_United_States_all_sectors_monthly.csv",
  header=TRUE,
  skip=3)

#Inspect data
head(net_generation)
```

```
##    Month all.fuels..utility.scale..thousand.megawatthours
## 1 Dec-20                                         344970.4
## 2 Nov-20                                         302701.8
## 3 Oct-20                                         313910.0
## 4 Sep-20                                         334270.1
## 5 Aug-20                                         399504.2
## 6 Jul-20                                         414242.5
##   coal.thousand.megawatthours natural.gas.thousand.megawatthours
## 1                    78700.33                           125703.7
## 2                    61332.26                           109037.2
## 3                    59894.57                           131658.2
## 4                    68448.00                           141452.7
## 5                    91252.48                           173926.6
## 6                    89831.36                           185444.8
##   nuclear.thousand.megawatthours
## 1                       69870.98
## 2                       61759.98
## 3                       59362.46
## 4                       65727.32
## 5                       68982.19
## 6                       69385.44
##   conventional.hydroelectric.thousand.megawatthours
## 1                                          23086.37
## 2                                          21831.88
## 3                                          18320.72
## 4                                          19161.97
## 5                                          24081.57
## 6                                          27675.94
```

```
nvar <- ncol(net_generation) - 1
nobs <- nrow(net_generation)

net_generation_processed <-
  net_generation %>%
  mutate( Month = my(Month) ) %>%
  rename( All.Fuels = all.fuels..utility.scale..thousand.megawatthours ) %>%
  rename( Coal = coal.thousand.megawatthours ) %>%
  rename( Natural.Gas = natural.gas.thousand.megawatthours ) %>%
  rename( Nuclear = nuclear.thousand.megawatthours ) %>%
  rename( Conventional.Hydroelectric = conventional.hydroelectric.thousand.megawatthours ) %>%
```

```r
  arrange( Month )

summary(net_generation_processed)
```

```
##      Month               All.Fuels           Coal          Natural.Gas
##  Min.   :2001-01-01   Min.   :276127   Min.   : 40624   Min.   : 37967
##  1st Qu.:2005-12-24   1st Qu.:309142   1st Qu.:113769   1st Qu.: 62245
##  Median :2010-12-16   Median :328297   Median :141665   Median : 84415
##  Mean   :2010-12-16   Mean   :336382   Mean   :135391   Mean   : 88028
##  3rd Qu.:2015-12-08   3rd Qu.:357671   3rd Qu.:161751   3rd Qu.:108385
##  Max.   :2020-12-01   Max.   :421797   Max.   :190135   Max.   :185445
##     Nuclear       Conventional.Hydroelectric
##  Min.   :54547   Min.   :14743
##  1st Qu.:62651   1st Qu.:19568
##  Median :65775   Median :22307
##  Mean   :66037   Mean   :22639
##  3rd Qu.:70438   3rd Qu.:25475
##  Max.   :74649   Max.   :32607
```

```r
ts_net_generation <- ts(
  net_generation_processed[,2:(nvar+1)],
  start=c(year(net_generation_processed$Month[1]),month(net_generation_processed$Month[1])),
  frequency=12)

#note that we are only transforming columns with electricity price, not the date columns
head(ts_net_generation,15)
```

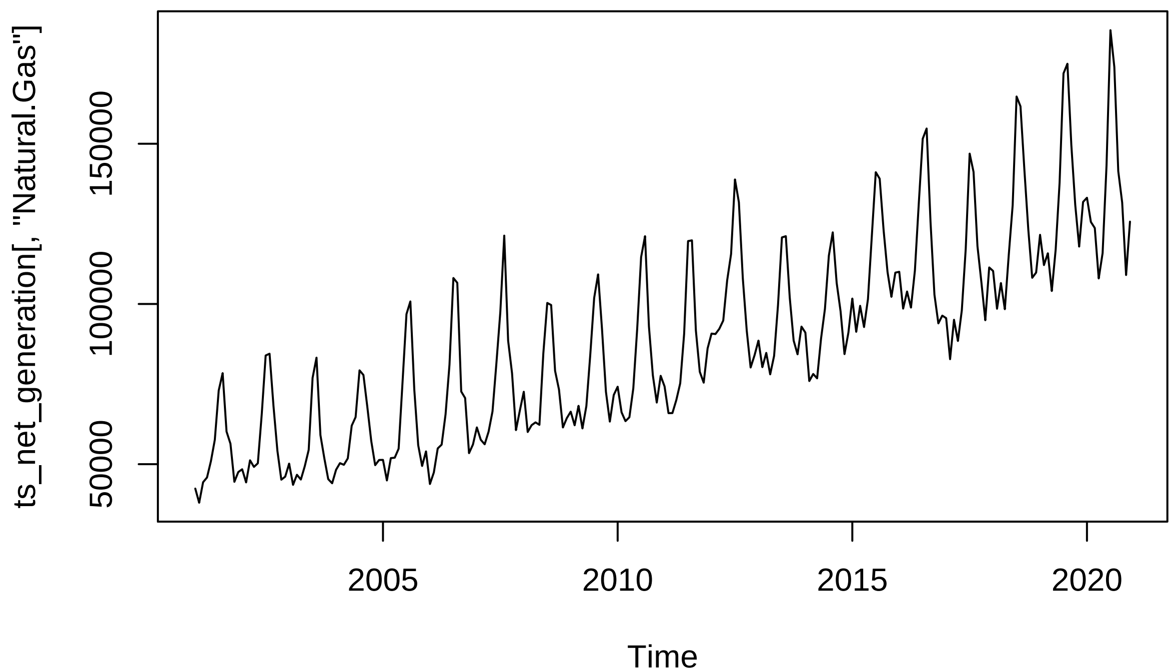
```
##          All.Fuels     Coal Natural.Gas  Nuclear Conventional.Hydroelectric
## Jan 2001  332493.2 177287.1    42388.66 68707.08                   18852.05
## Feb 2001  282940.2 149735.5    37966.93 61272.41                   17472.89
## Mar 2001  300706.5 155269.0    44364.41 62140.71                   20477.19
## Apr 2001  278078.9 140670.7    45842.75 56003.03                   18012.99
## May 2001  300491.6 151592.9    50934.21 61512.44                   19175.63
## Jun 2001  327694.0 162615.8    57603.15 68023.10                   20727.63
## Jul 2001  357613.7 179060.4    73030.14 69166.04                   18079.12
## Aug 2001  370532.8 183116.1    78409.80 68389.50                   18913.77
## Sep 2001  306928.9 154158.3    60181.14 63378.45                   15256.03
## Oct 2001  294733.6 148930.8    56376.44 60460.97                   15234.50
## Nov 2001  278933.9 144117.0    44490.62 62341.71                   15412.93
## Dec 2001  305496.3 157402.4    47540.86 67430.88                   19346.31
## Jan 2002  319941.5 164358.0    48412.83 70925.86                   21794.93
## Feb 2002  281825.7 143048.8    44308.43 61658.27                   20191.68
## Mar 2002  302549.0 151485.7    51214.46 63040.65                   21008.81
```

```r
tail(ts_net_generation,15)
```

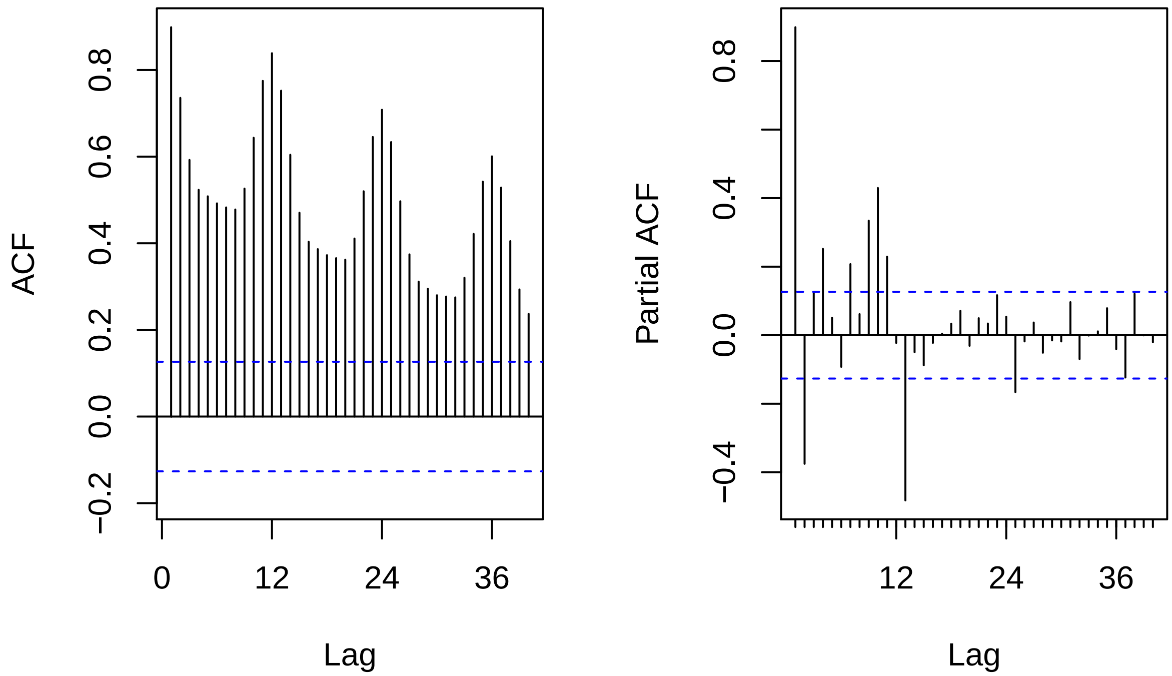
```
##          All.Fuels     Coal Natural.Gas  Nuclear Conventional.Hydroelectric
## Oct 2019  320351.9 66777.23    130947.6 62032.62                   18305.81
## Nov 2019  315849.1 75549.34    117910.5 64125.43                   20217.60
## Dec 2019  338401.6 72580.74    131838.9 73073.57                   21478.18
## Jan 2020  340668.7 65099.96    133157.6 74169.65                   25331.54
```

```
## Feb 2020  318167.6 56114.39    125593.9 65950.34                     26370.50
## Mar 2020  307479.1 50643.86    123697.0 63997.21                     23594.46
## Apr 2020  276127.3 40623.51    107960.0 59170.02                     22112.08
## May 2020  304277.2 46529.54    115870.9 64337.97                     30485.00
## Jun 2020  352766.1 65335.08    143245.4 67205.08                     29058.82
## Jul 2020  414242.5 89831.36    185444.8 69385.44                     27675.94
## Aug 2020  399504.2 91252.48    173926.6 68982.19                     24081.57
## Sep 2020  334270.1 68448.00    141452.7 65727.32                     19161.97
## Oct 2020  313910.0 59894.57    131658.2 59362.46                     18320.72
## Nov 2020  302701.8 61332.26    109037.2 61759.98                     21831.88
## Dec 2020  344970.4 78700.33    125703.7 69870.98                     23086.37
```

```r
plot(ts_net_generation[,"Natural.Gas"])
```




```r
#ACF and PACF plots
par(mfrow=c(1,2))
ACF_Plot <- Acf(ts_net_generation[,"Natural.Gas"], lag = 40, plot = TRUE,main="")
PACF_Plot <- Pacf(ts_net_generation[,"Natural.Gas"], lag = 40, plot = TRUE,main="")
```

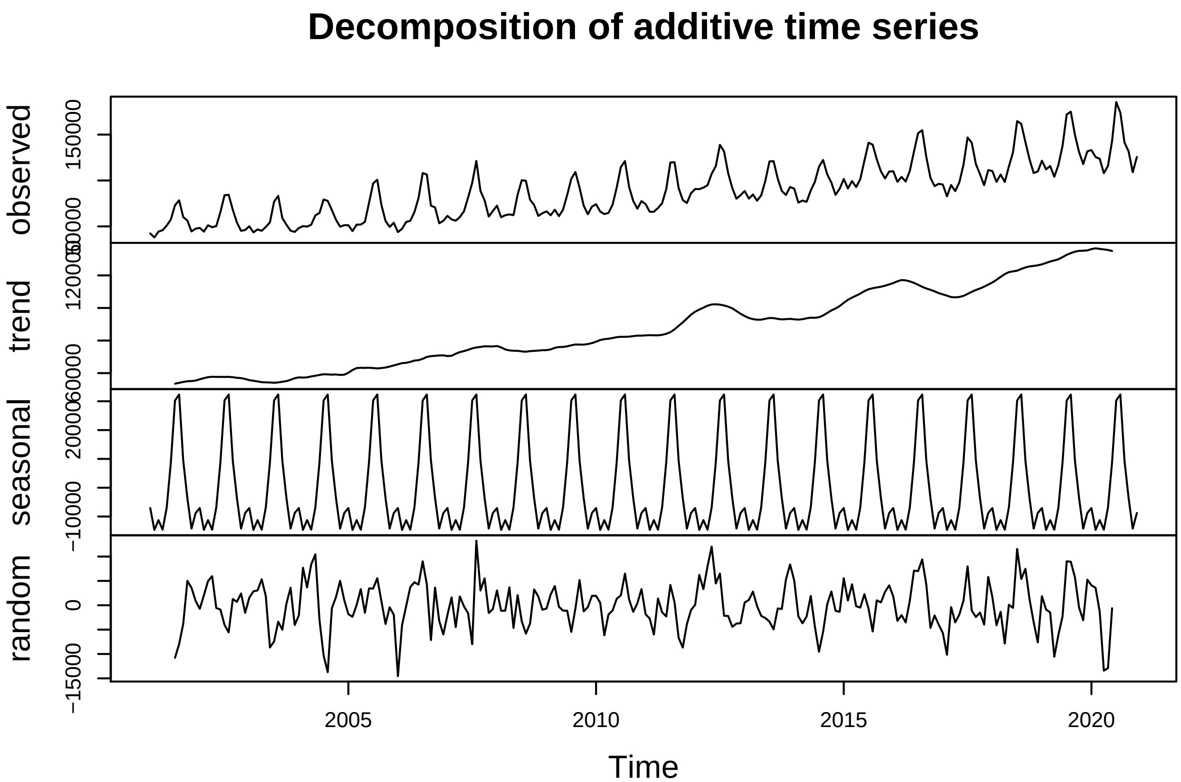



```r
par(mfrow=c(1,1))
```

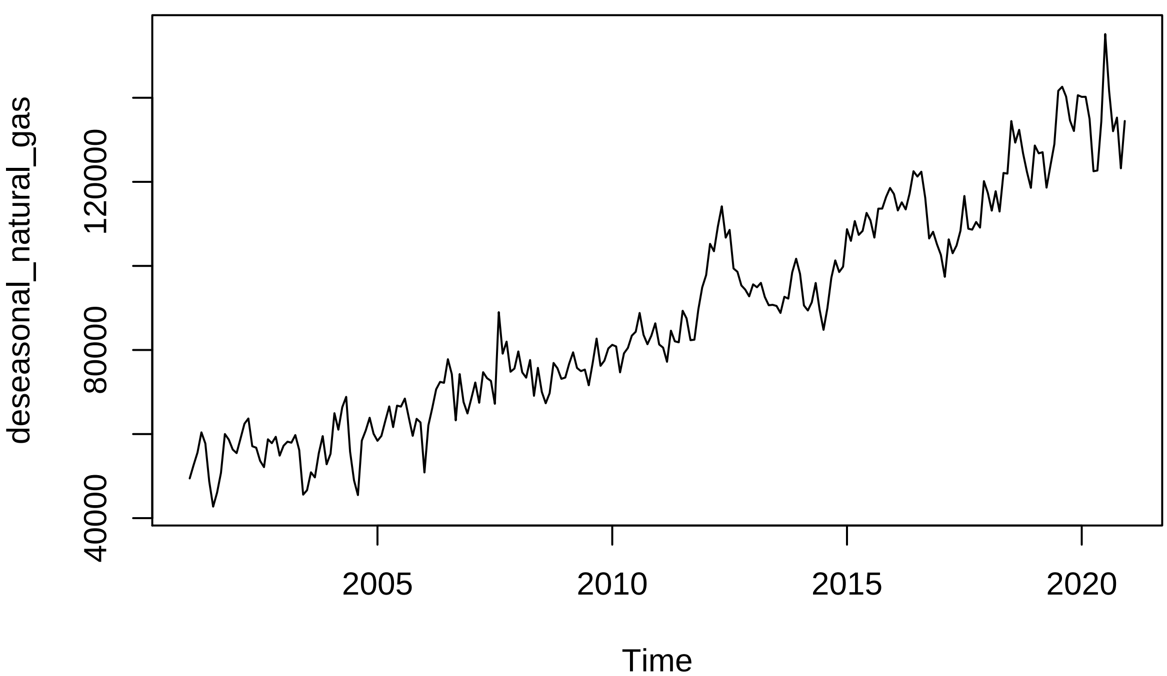
Q2

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.

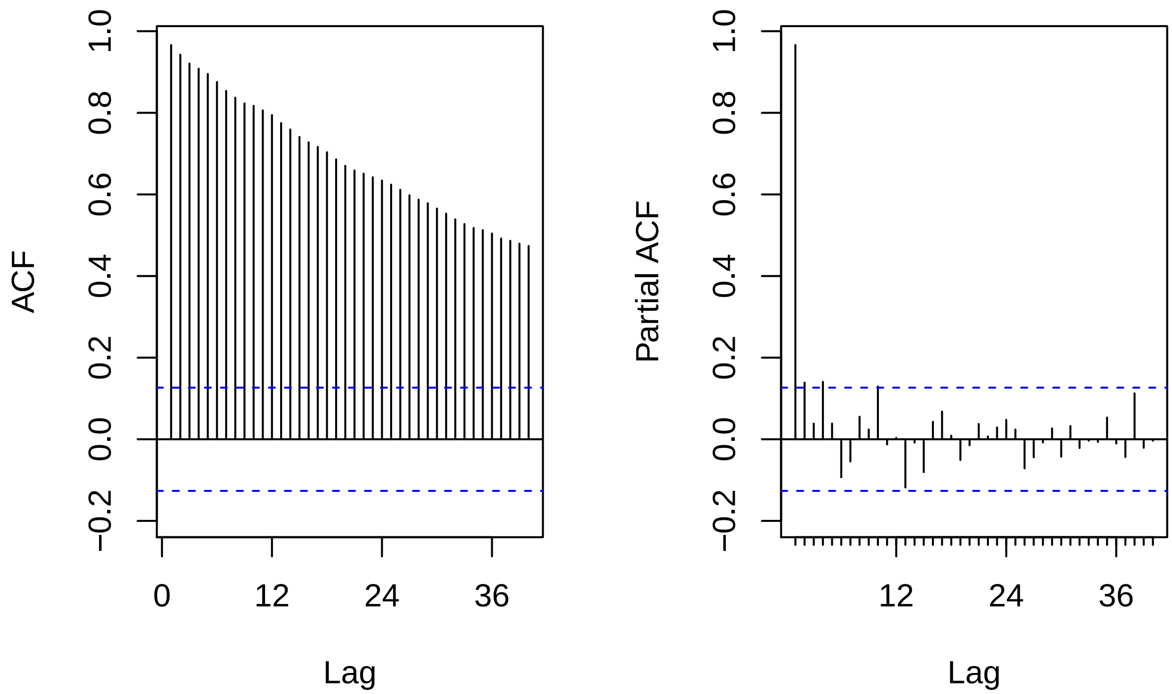
```r
#Using R decompose function
decompose_natural_gas <- decompose(ts_net_generation[,"Natural.Gas"],"additive")
plot(decompose_natural_gas)
```




```r
deseasonal_natural_gas <- seasadj(decompose_natural_gas)
plot(deseasonal_natural_gas)
```




```r
#ACF and PACF plots
par(mfrow=c(1,2))
ACF_Plot <- Acf(deseasonal_natural_gas, lag = 40, plot = TRUE,main="")
PACF_Plot <- Pacf(deseasonal_natural_gas, lag = 40, plot = TRUE,main="")
```




```r
par(mfrow=c(1,1))
```

The plots in Q2 clearly show the deseasoned data from Q1. The overall trend appears the same but most the large jumps I would associate with seasonality have been removed (there still are some here or there). The ACF also has the sinusoidal variation removed in the Q2 data (compared to the Q1) and the spikes in the PACF that exist in the Q1 data do not exist in the Q2 data.

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.

```r
print("Results for ADF test/n")
```

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

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

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

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

```r
print("Results of Mann Kendall on deseasonalized Natural Gas data")
```

```
## [1] "Results of Mann Kendall on deseasonalized Natural Gas data"
```

```r
print(summary(MannKendall(deseasonal_natural_gas)))
```

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

The ADF test has a null hypothesis that states that the data has a unit root and that its trend will not eventually return to a stationary trend. The alternative hypothesis states that there is not a unit root and is not dependent entirely on the previous observation. This ADF test shows a p-value with 0.01 which means we reject the null hypothesis, suggesting that we have a deterministic trend.

The Mann Kendall test's null hypothesis states that the data is stationary and its alternate hypothesis states that the data follows a trend. It's two sided p-value of =< 2.22e-16 tells us to reject the null hypothesis of stationary data.

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.

Based on the ACF and PACF plots from Q2 and the test results from Q3, this ARIMA model has p = 1, d = 1, and q = 0. This is an autoregressive model with a deterministic trend. The ADF and Mann Kendall tests show us that this model has a deterministic trend and must be differenced (d=1) and the slow decay of the ACF along with the lag = 1 cutoff on the PACF show that this is an AR model with p = 1. This is not an MA model so 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.

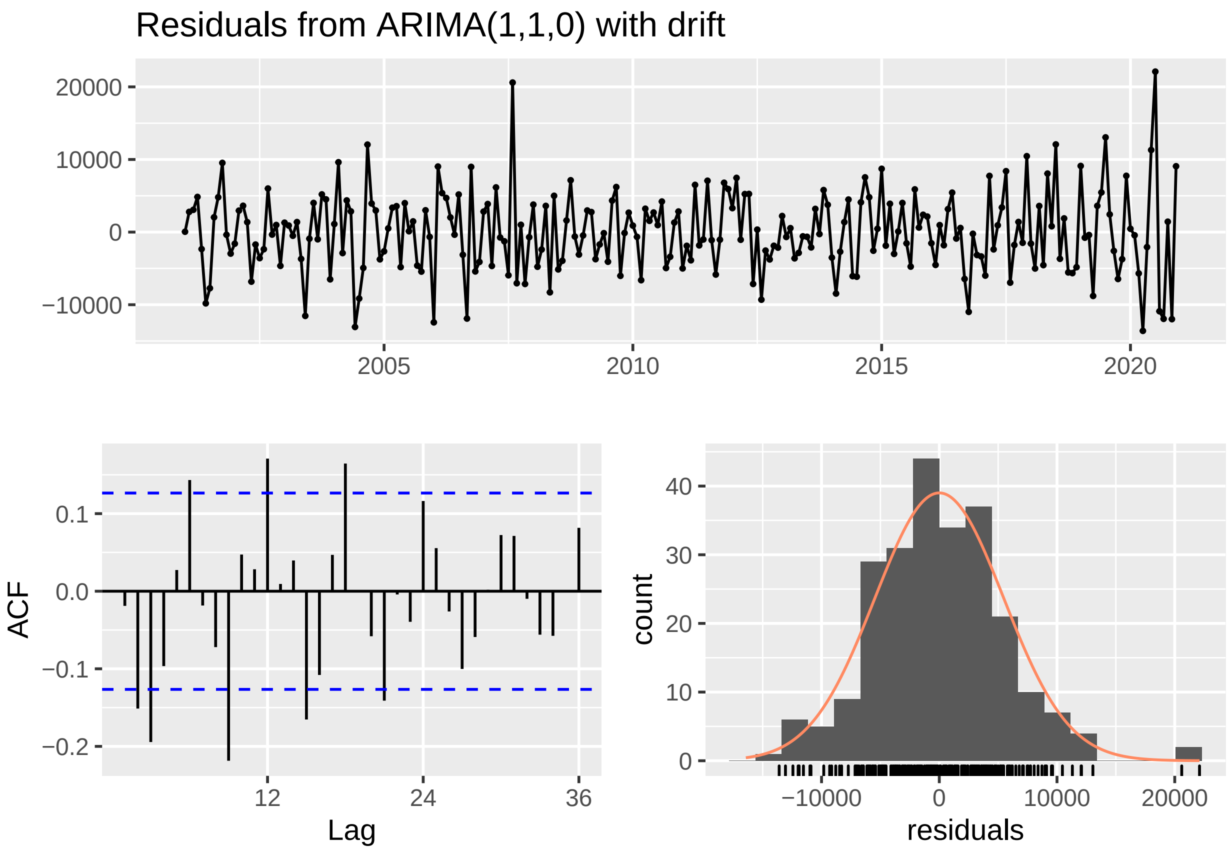
```r
ARIMA_NG <- Arima(deseasonal_natural_gas,order=c(1,1,0),seasonal=c(0,0,0),include.mean=TRUE,include.dri
print(ARIMA_NG)
```

```
## Series: deseasonal_natural_gas
## ARIMA(1,1,0) with drift
##
## Coefficients:
##           ar1     drift
##       -0.1479  348.3927
## s.e.   0.0644  308.8385
##
## sigma^2 estimated as 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?

```
checkresiduals(ARIMA_NG)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,0) with drift
## Q* = 72.475, df = 22, p-value = 2.683e-07
##
## Model df: 2.   Total lags used: 24
```

After running the checkresiduals function, these 3 plots appear to show a reasonable residual white noise series. The residual series seems random, normally distributed and the ACF does not appear to show a significant self-correlation.

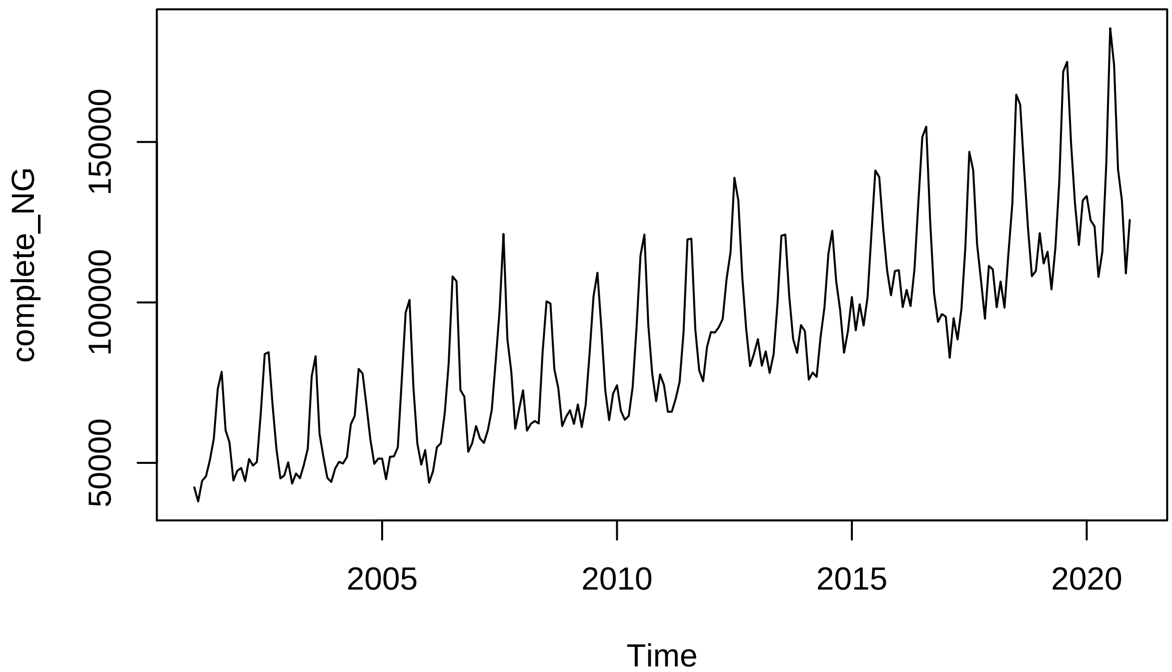
Modeling the original series (with seasonality)

Q7

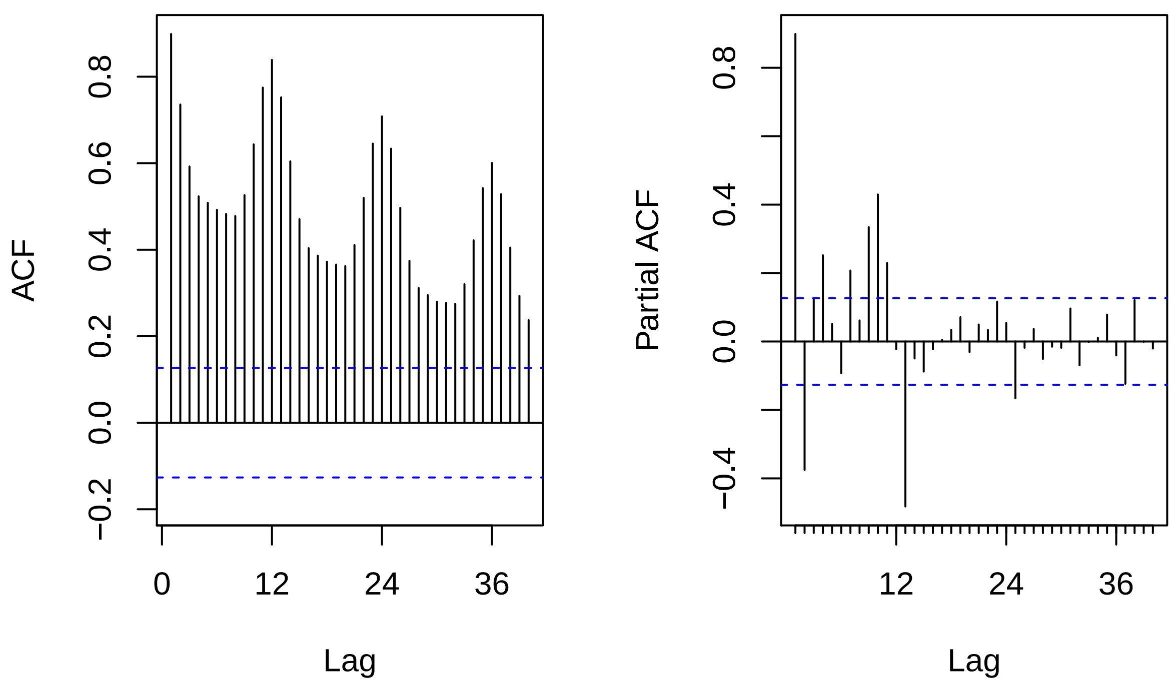
Repeat Q4-Q6 for the original series (the complete series that has the seasonal component). Note that when you model the seasonal series, you need to specify the seasonal part of the ARIMA model as well, i.e., P, D and Q.

```r
complete_NG<-ts_net_generation[,"Natural.Gas"]

plot(complete_NG)
```




```r
#ACF and PACF plots
par(mfrow=c(1,2))
ACF_Plot_2 <- Acf(complete_NG, lag = 40, plot = TRUE,main="")
PACF_Plot_2 <- Pacf(complete_NG, lag = 40, plot = TRUE,main="")
```



```r
par(mfrow=c(1,1))

print("Results for ADF test/n")
```

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

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

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

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

```r
print("Results of Mann Kendall on Seasonal Natural Gas data")
```

```
## [1] "Results of Mann Kendall on Seasonal Natural Gas data"
```

```r
print(summary(MannKendall(complete_NG)))
```

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

After plotting the seasonal trend and its ACF and PACF, and after running the ADF and Mann Kendall tests, the associated SARIMA model has p = 2, d = 1, and q = 0. This is an autoregressive seasonal model with a deterministic trend. The ADF and Mann Kendall tests show us that this model has a deterministic trend and must be differenced both seasonally and non-seasonally (d=D=1, because I know that this has a seasonal trend) and the slow decay of the ACF along with the lag = 2 cutoff on the PACF show that this is an AR model with p = 2. This is not an MA model so q = 0. The small lag spikes in the PACF likely indicate that P = 1 and the lag spikes in the ACF at lag = 12, 24, 36 indicate that Q = 1 as well.

```r
ARIMA_NG_seasonal <- Arima(complete_NG,order=c(2,1,0),seasonal=c(1,1,1),include.mean=TRUE,include.drift
```

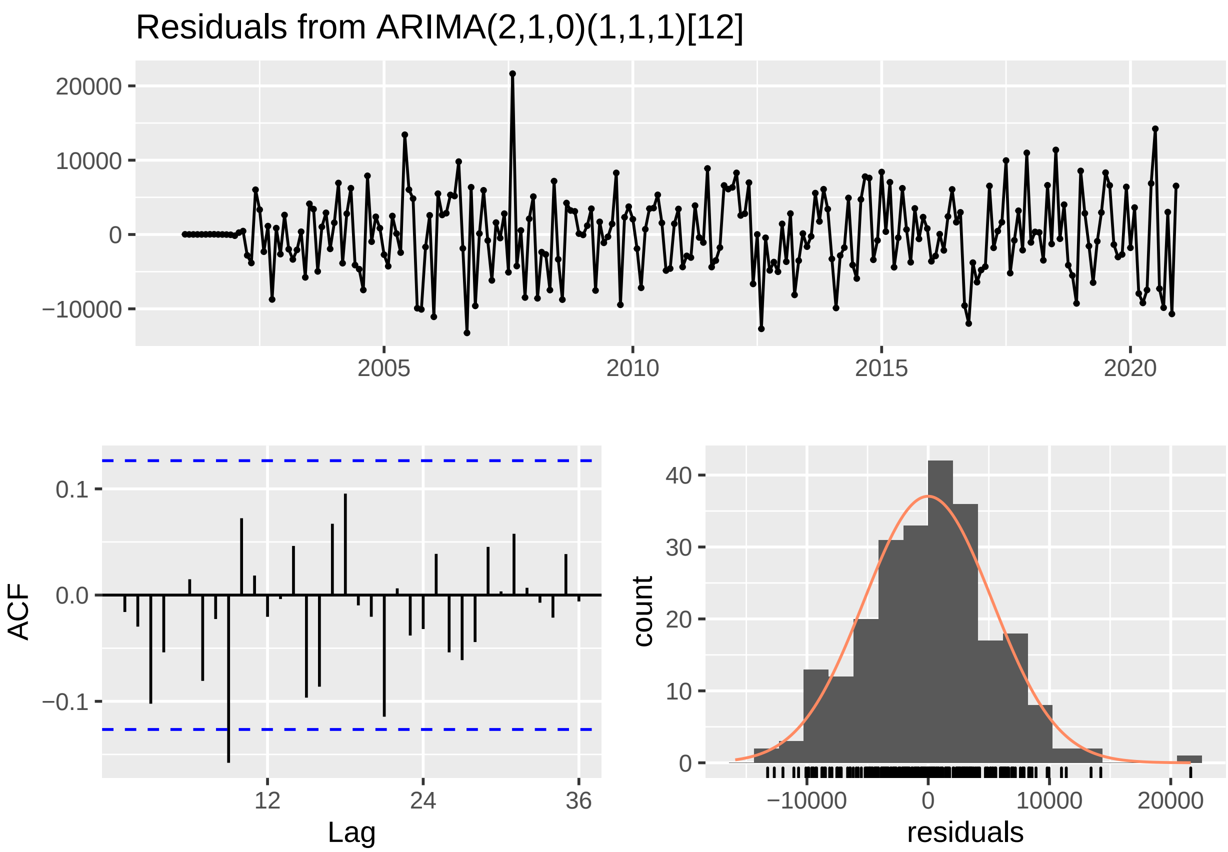
```
## Warning in Arima(complete_NG, order = c(2, 1, 0), seasonal = c(1, 1, 1), : No
## drift term fitted as the order of difference is 2 or more.
```

```r
print(ARIMA_NG_seasonal)
```

```
## Series: complete_NG
## ARIMA(2,1,0)(1,1,1)[12]
##
## Coefficients:
##          ar1      ar2     sar1     sma1
```

```
##       -0.2100  -0.1643  -0.0026  -0.6726
## s.e.   0.0658   0.0671   0.1023   0.0835
##
## sigma^2 estimated as 30141883:  log likelihood=-2278.39
## AIC=4566.78   AICc=4567.05   BIC=4583.91
```

```
checkresiduals(ARIMA_NG_seasonal)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,1,0)(1,1,1)[12]
## Q* = 25.874, df = 20, p-value = 0.17
##
## Model df: 4.   Total lags used: 24
```

After running the ARIMA model as (2,1,0)x(1,1,1), the residual series does once again appear to be a reasonable white noise series. The residuals appear to be random, somewhat normally distributed, and there does not seem to be any serious self-correlation.

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.

After running the two separate models, both seem to have relatively normally distributed residuals with no significant self correlation - they are both reasonably look like white noise series. However, if I had to pick one of these models I would pick the SARIMA (Q7) model. The ACF of the ARIMA model, while showing only a small level of self correlation, does seem to have a pattern. There are noticeable and seemingly recurring spikes throughout the ACF as we move from lag = 1 to lag = 40. Again, these spikes have low values and I'm not that concerned with them, but the SARIMA model only has a single spike at lag = 8 and so I am more comfortable picking the SARIMA model.

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(deseasonal_natural_gas)
```

```
## Series: deseasonal_natural_gas
## 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
```

I chose ARIMA(1,1,0) while the auto.arima went with ARIMA(1,1,1). Evidently the function identified a moving average component to the model that I did not. I'm not sure why it would select this order. The PACF of the deseasonal data showed a clear cutoff at lag = 1 and a clear slow decay in the ACF. Based on this information I chose to just include an AR process in my model. There may be another reason to choose an MA process that I am missing, but for nowI remain confident in my reasoning.

The ADF and Mann Kendall tests showed a deterministic trend that I decided needed to be differenced. I chose d = 1 for my model and the auto.arima agreed.

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(complete_NG)
```

```
## Series: complete_NG
## 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 27569123:  log likelihood=-2279.54
## AIC=4567.08   AICc=4567.26   BIC=4580.8
```

I chose ARIMA(2,1,0)(1,1,1) while the auto.arima went with ARIMA(1,0,0)(0,1,1)

I chose to difference the equation both seasonally and non-seasonally and while the auto.arima function agreed with my seasonal difference, it did not with my non-seasonal difference. The reason I chose to include both differences is because the non-seasonal data had to be differenced in the previous question, so I felt like just seasonally differencing the data would be insufficient, but I suppose I may have been wrong.

I chose a non-seasonal AR order of 2, because the PACF of the data cuts off at lag = 2, not 1. The auto.arima function does not agree and felt that an order of 1 was better. I'm not sure why.

The auto.arima function agreed that the spikes in the ACF at lag = 12, 24, etc. made a strong case for the seasonal MA process to have an order of 1. However, the auto.arima function did not agree that the small spikes along the PACF at roughly lag =12, 24, 36 indicate a seasonal autoregressive component to the model. This is somewhat understandable given that the spikes were actually at lag = 13, 25, 37 but I was under the impression that this might have had something to do with p = 2 (which was not the correct value for p).