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

Assignment 6 - Due date 03/26/21

Xueying Feng

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(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 1.3.0 --
```

```
## v tibble  3.1.0     v dplyr   1.0.5
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     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", 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
```

```
#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.

```
NetGen <- read.csv("../Data/Net_generation_United_States_all_sectors_monthly.csv",skip = 4)
head(NetGen)
```

```
##      Month all.fuels..utility.scale..thousand.megawatthours
## 1 Dec 2020                                          344970.4
## 2 Nov 2020                                          302701.8
## 3 Oct 2020                                          313910.0
## 4 Sep 2020                                          334270.1
## 5 Aug 2020                                          399504.2
```

```
## 6 Jul 2020                                          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
```

```
#Inspect data
head(NetGen)
```

```
##      Month all.fuels..utility.scale..thousand.megawatthours
## 1 Dec 2020                                          344970.4
## 2 Nov 2020                                          302701.8
## 3 Oct 2020                                          313910.0
## 4 Sep 2020                                          334270.1
## 5 Aug 2020                                          399504.2
## 6 Jul 2020                                          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
```

```r
ncol <- ncol(NetGen)
nobs <- nrow(NetGen)

# Change column names
#colnames(NetGen)[1] <- "Date"
colnames(NetGen)=c("Date","AllFuels","Coal","NG", "Nuclear","ConventionalHydroelectric")
str(NetGen)
```

```
## 'data.frame':    240 obs. of  6 variables:
##  $ Date                     : Factor w/ 240 levels "Apr 2001","Apr 2002",..: 60 200 220 240 40 120 14
##  $ AllFuels                 : num  344970 302702 313910 334270 399504 ...
##  $ Coal                     : num  78700 61332 59895 68448 91252 ...
##  $ NG                       : num  125704 109037 131658 141453 173927 ...
##  $ Nuclear                  : num  69871 61760 59362 65727 68982 ...
##  $ ConventionalHydroelectric: num  23086 21832 18321 19162 24082 ...
```

```r
head(NetGen)
```

```
##       Date AllFuels     Coal       NG  Nuclear ConventionalHydroelectric
## 1 Dec 2020 344970.4 78700.33 125703.7 69870.98                   23086.37
## 2 Nov 2020 302701.8 61332.26 109037.2 61759.98                   21831.88
## 3 Oct 2020 313910.0 59894.57 131658.2 59362.46                   18320.72
## 4 Sep 2020 334270.1 68448.00 141452.7 65727.32                   19161.97
## 5 Aug 2020 399504.2 91252.48 173926.6 68982.19                   24081.57
## 6 Jul 2020 414242.5 89831.36 185444.8 69385.44                   27675.94
```

```r
#convert to numeric
#NetGen[,2:ncol] <- sapply(NetGen[,2:ncol],as.numeric)

# change character format to numeric format
#library(lubridate)
NetGen$Date <- parse_date_time(NetGen$Date,"by")

#arrange date from 2001 to 2020
NetGen<- NetGen %>%
  arrange(NetGen$Date)

str(NetGen)
```

```
## 'data.frame':    240 obs. of  6 variables:
##  $ Date                     : POSIXct, format: "2001-01-01" "2001-02-01" ...
##  $ AllFuels                 : num  332493 282940 300707 278079 300492 ...
##  $ Coal                     : num  177287 149735 155269 140671 151593 ...
##  $ NG                       : num  42389 37967 44364 45843 50934 ...
##  $ Nuclear                  : num  68707 61272 62141 56003 61512 ...
##  $ ConventionalHydroelectric: num  18852 17473 20477 18013 19176 ...
```

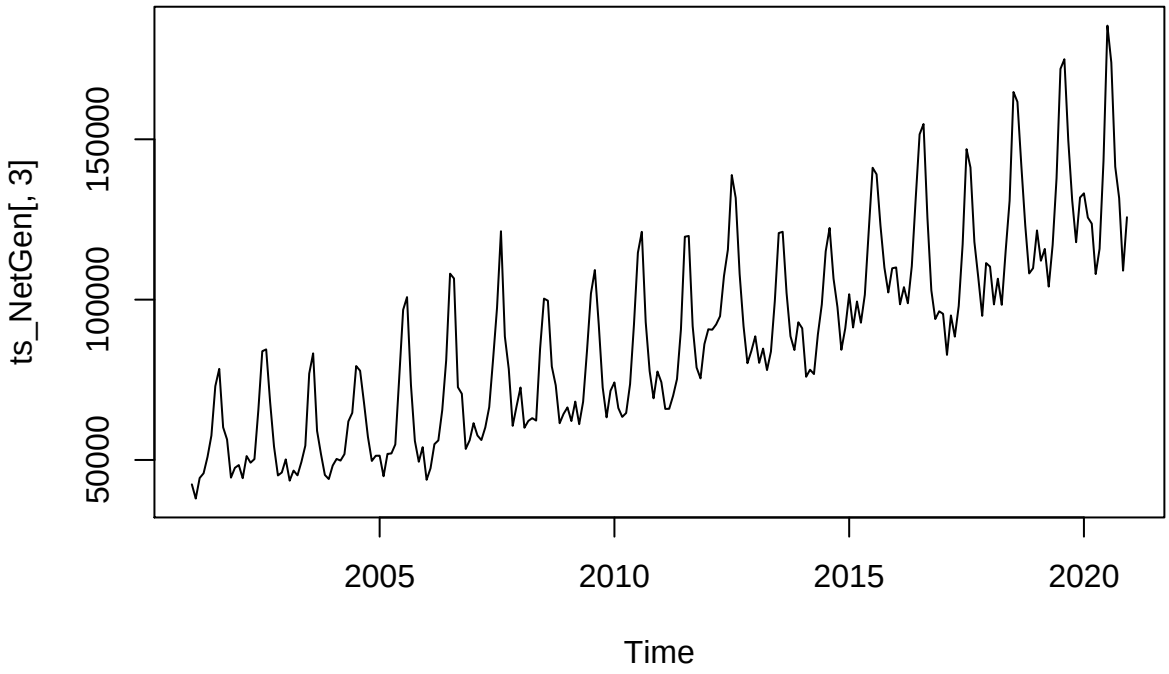
```r
head(NetGen)
```

```
##         Date AllFuels     Coal       NG  Nuclear ConventionalHydroelectric
## 1 2001-01-01 332493.2 177287.1 42388.66 68707.08                   18852.05
## 2 2001-02-01 282940.2 149735.5 37966.93 61272.41                   17472.89
## 3 2001-03-01 300706.5 155269.0 44364.41 62140.71                   20477.19
## 4 2001-04-01 278078.9 140670.7 45842.75 56003.03                   18012.99
## 5 2001-05-01 300491.6 151592.9 50934.21 61512.44                   19175.63
## 6 2001-06-01 327694.0 162615.8 57603.15 68023.10                   20727.63
```

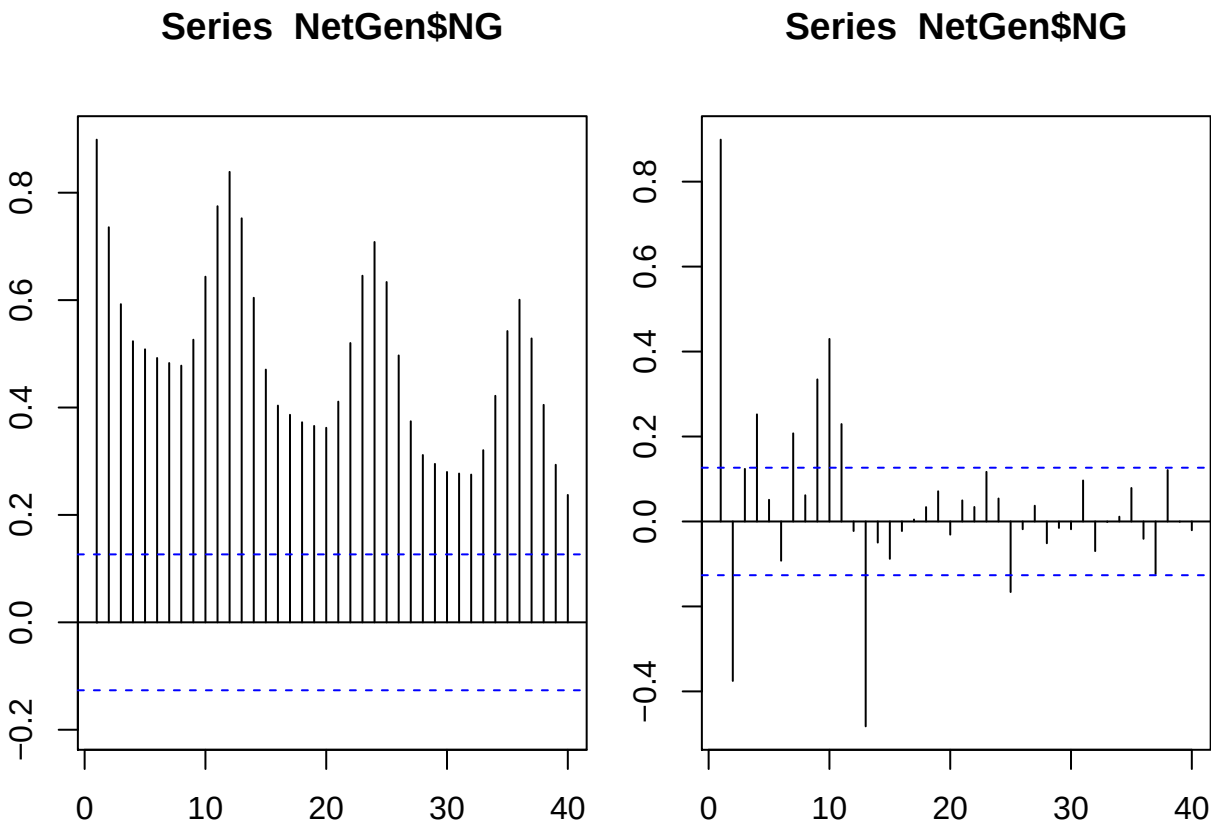
```
#Transforming data into time series object
ts_NetGen <- ts(NetGen[,2:(ncol)],
                start=c(year(NetGen$Date[1]),
                        month(NetGen$Date[1])),
                frequency=12)
str(ts_NetGen)
```

```
##  Time-Series [1:240, 1:5] from 2001 to 2021: 332493 282940 300707 278079 300492 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:5] "AllFuels" "Coal" "NG" "Nuclear" ...
```

```
#plot NG trend
plot(ts_NetGen[,3])
```




```
#ACF and PACF plots
par(mar=c(3,3,3,0));par(mfrow=c(1,2))
ACF_Plot <- Acf(NetGen$NG, lag = 40, plot = TRUE)
  ## ACF should seasonality, and has correlation, it is not stationary
PACF_Plot <- Pacf(NetGen$NG, lag = 40)
```

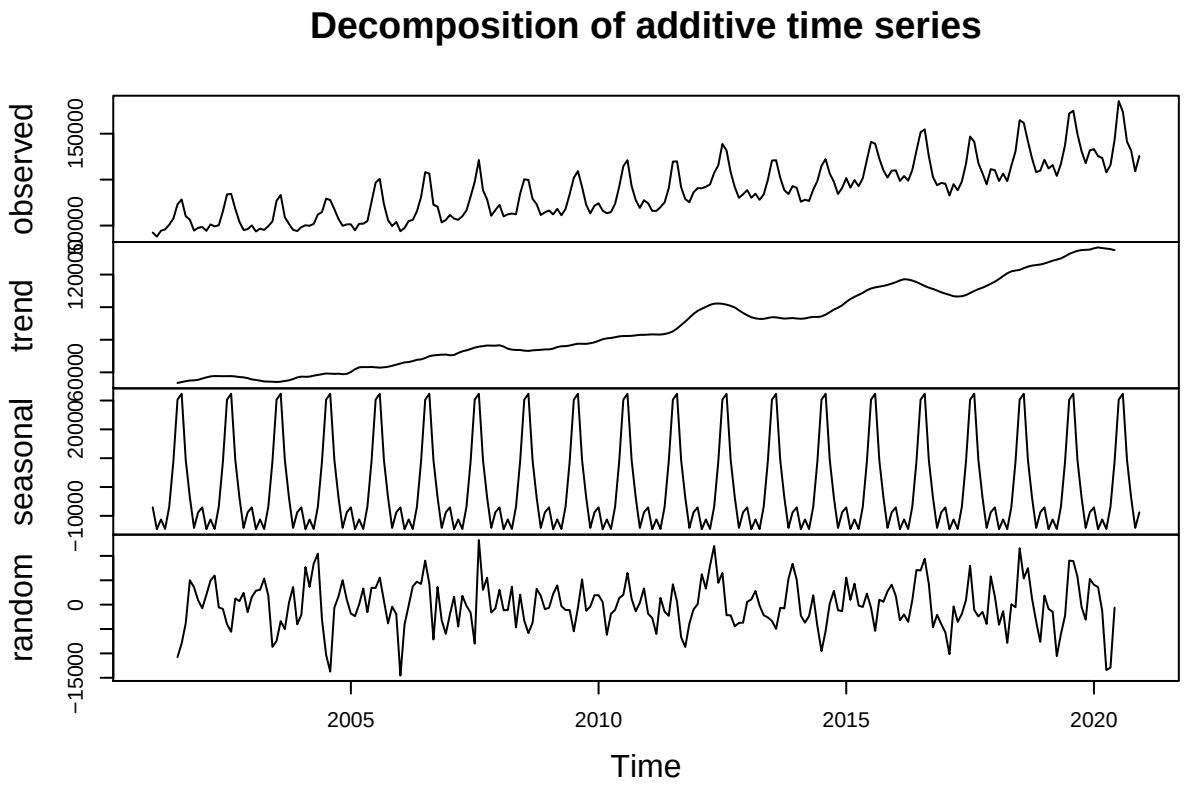



```
## Pacf plot shows we might dealing with stochastic or unit root (means random)
```

Q2

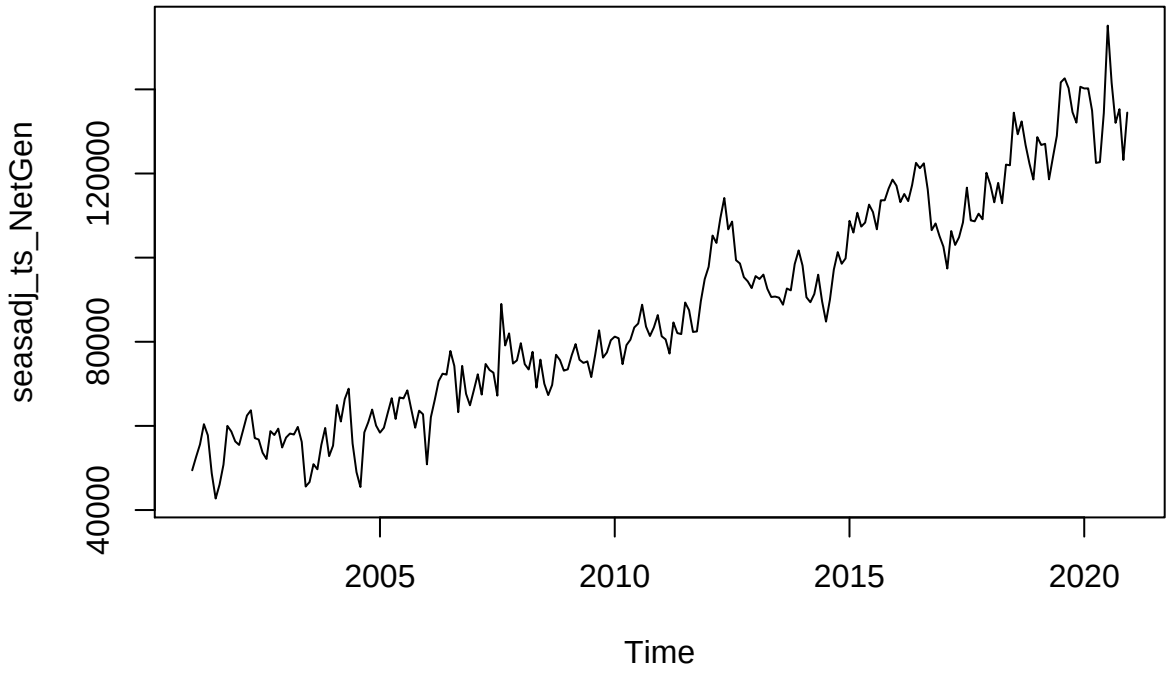
Using the *decompose*() or *stl*() and the *seasadj*() functions create a series without the seasonal component, i.e., a deseasonalized natural gas series. Plot the deseasonalized series over time and corresponding ACF and PACF. Compare with the plots obtained in Q1.

```
#Using R decompose function
decompose_ts_NetGen <- decompose(ts_NetGen[,"NG"],"additive")
plot(decompose_ts_NetGen)
```

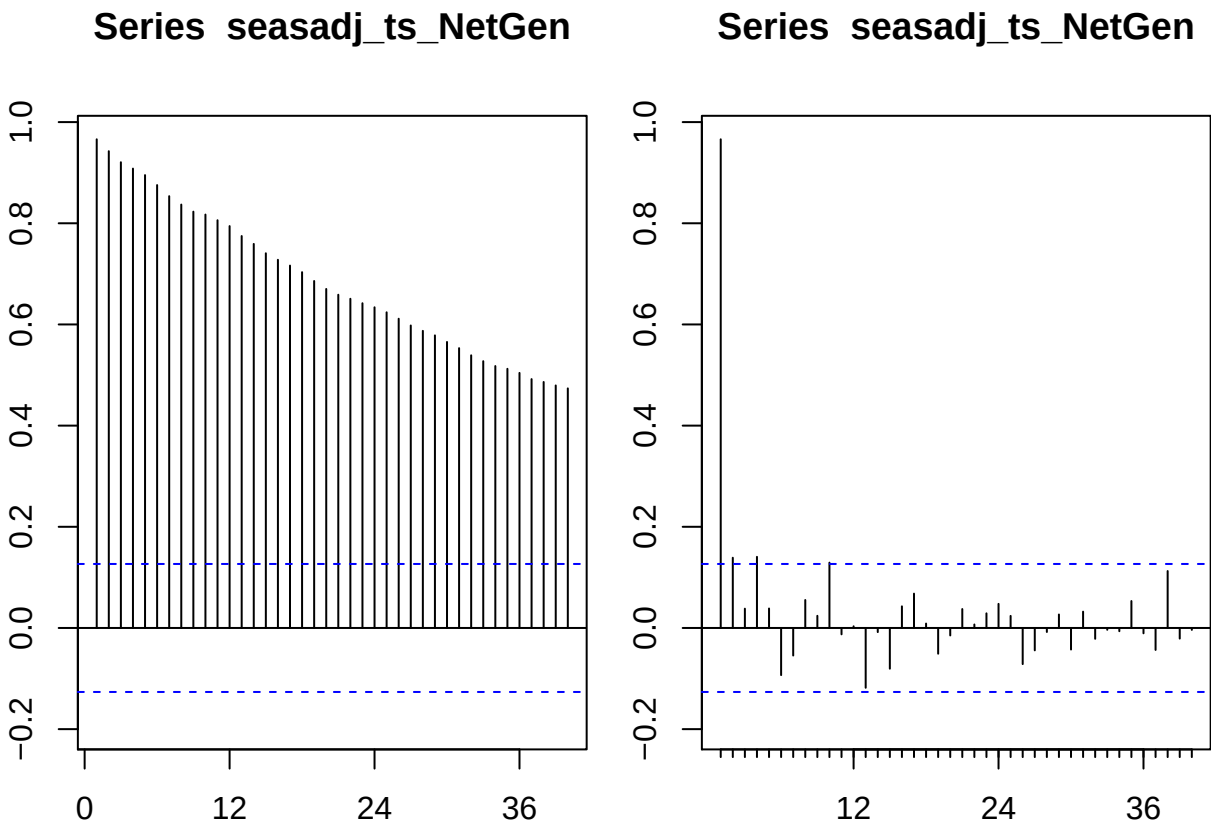



```
#decompose_ts_NetGen2 <- decompose(ts_NetGen[,"NG"],"multiplicative")
#plot(decompose_ts_NetGen2)

##Using R seasadj function
seasadj_ts_NetGen <- seasadj(decompose_ts_NetGen)
plot(seasadj_ts_NetGen)
```



```
par(mar=c(3,3,3,0));par(mfrow=c(1,2))
seasadj_ACF_Plot <- Acf(seasadj_ts_NetGen, lag = 40, plot = TRUE)
seasadj_PACF_Plot <- Pacf(seasadj_ts_NetGen, lag = 40)
```



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
ADFtest <- adf.test(seasadj_ts_NetGen,alternative="stationary")
```

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

```
print(ADFtest)
```

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

```
#Run MannKendall
library(Kendall)
print("Results of Mann Kendall on average yearly series")
```

```
## [1] "Results of Mann Kendall on average yearly series"
```

```
print(summary(MannKendall(seasadj_ts_NetGen)))
```

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

> Augmented Dickey-Fuller test: the p-value of the test is lower than significance level 0.05, then it is stationary. Mann Kendall: the p-value of the test is lower than significance level 0.05 , then there is statistically significant evidence that a trend is present in the time series 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.

> The stationary series has positive autocorrelation at lag 1, then autoregressive model(AR) terms work best. It is a autoregressive model with p=2 also because of a slow decay in the ACF plot and a clear cut off at lag 2 in the PACF plot (p=1). We assume we are working with a zero-mean process,leading to the conclusion that d=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.

```
Arima.model <- Arima(seasadj_ts_NetGen,order=c(1,0,0),include.mean=TRUE)
print(Arima.model)
```

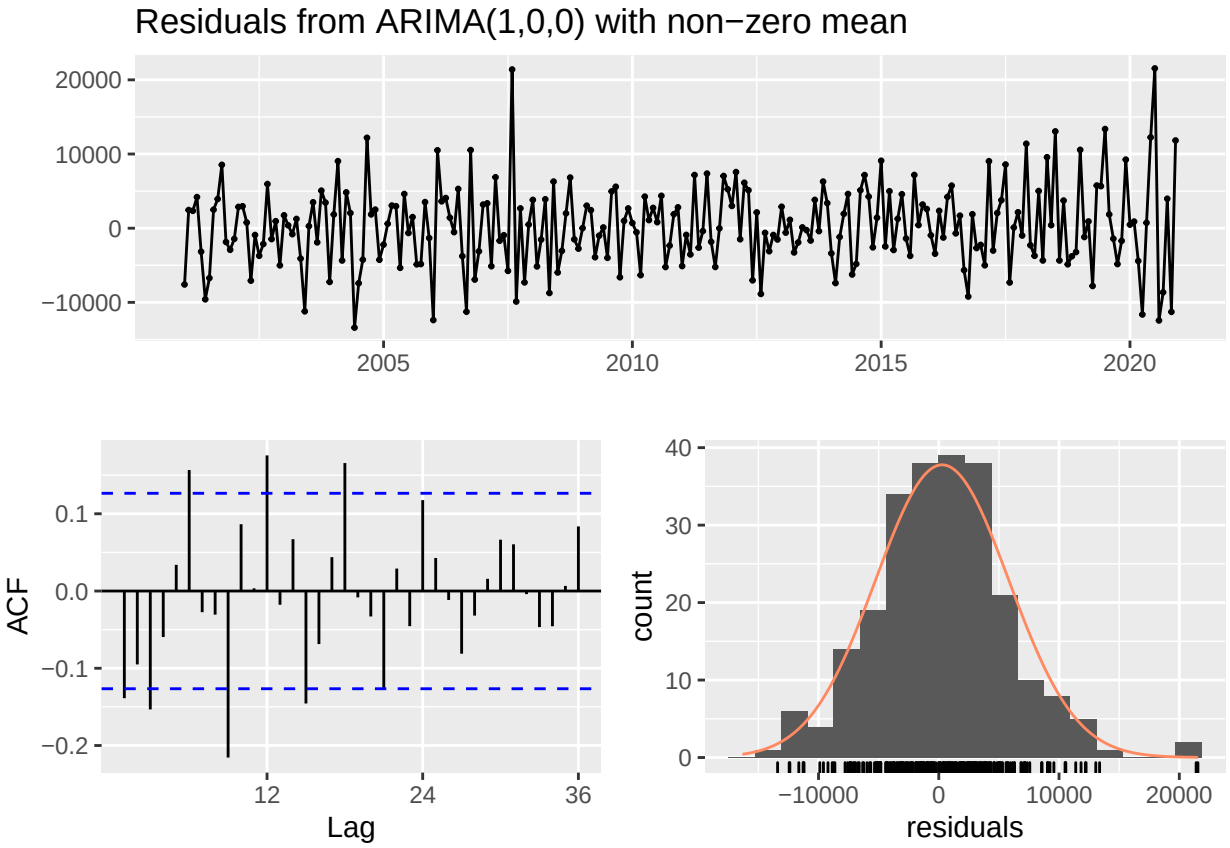
```
## Series: seasadj_ts_NetGen
## ARIMA(1,0,0) with non-zero mean
##
## Coefficients:
##          ar1      mean
##       0.9825  90230.35
## s.e.  0.0120  16958.50
##
## sigma^2 estimated as 30851494:  log likelihood=-2410.59
## AIC=4827.17   AICc=4827.27   BIC=4837.61
```

```
#cat("coefficients is:", )
```

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.model)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,0) with non-zero mean
## Q* = 66.317, df = 22, p-value = 2.479e-06
##
## Model df: 2.   Total lags used: 24
```

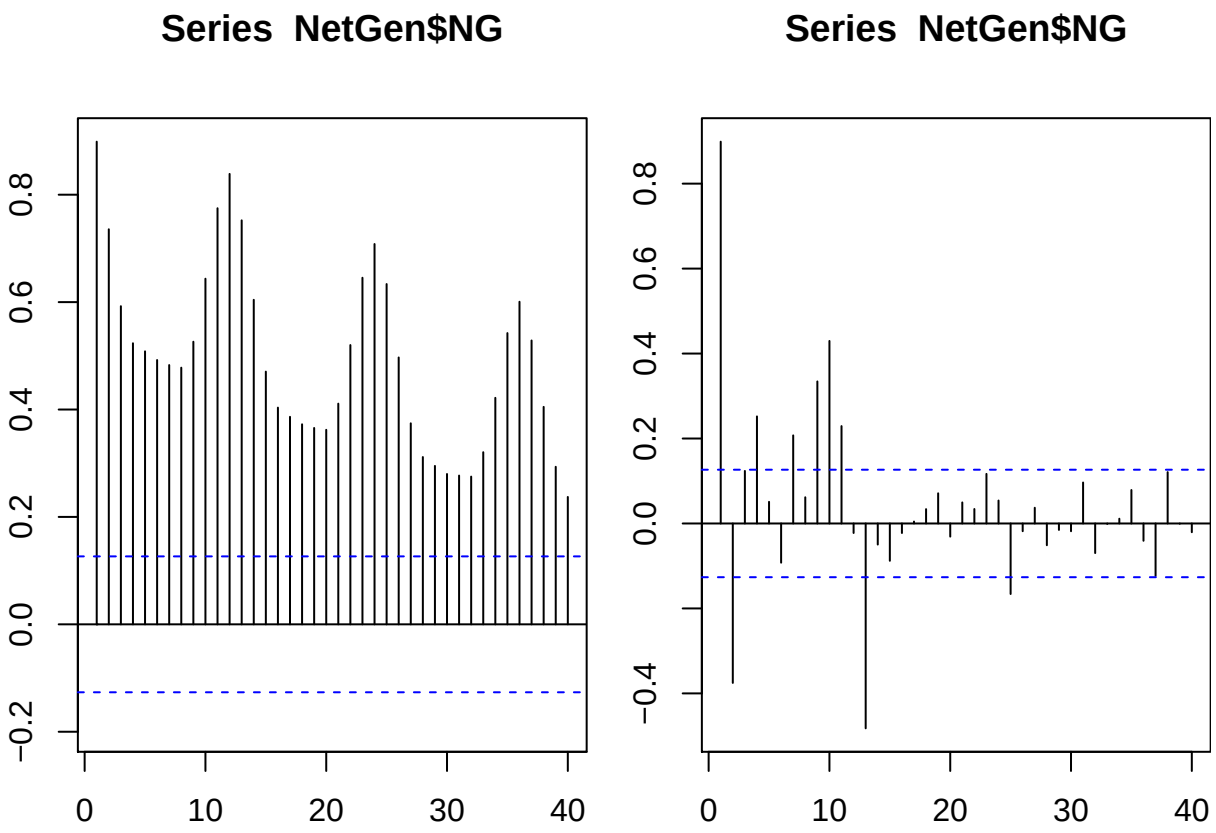
```
#The residual series do not look like a white noise series, because it is not between two blue dotted l
```

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.

```
#ACF and PACF plots
par(mar=c(3,3,3,0));par(mfrow=c(1,2))
ACF_Plot <- Acf(NetGen$NG, lag = 40, plot = TRUE)
PACF_Plot <- Pacf(NetGen$NG, lag = 40)
```

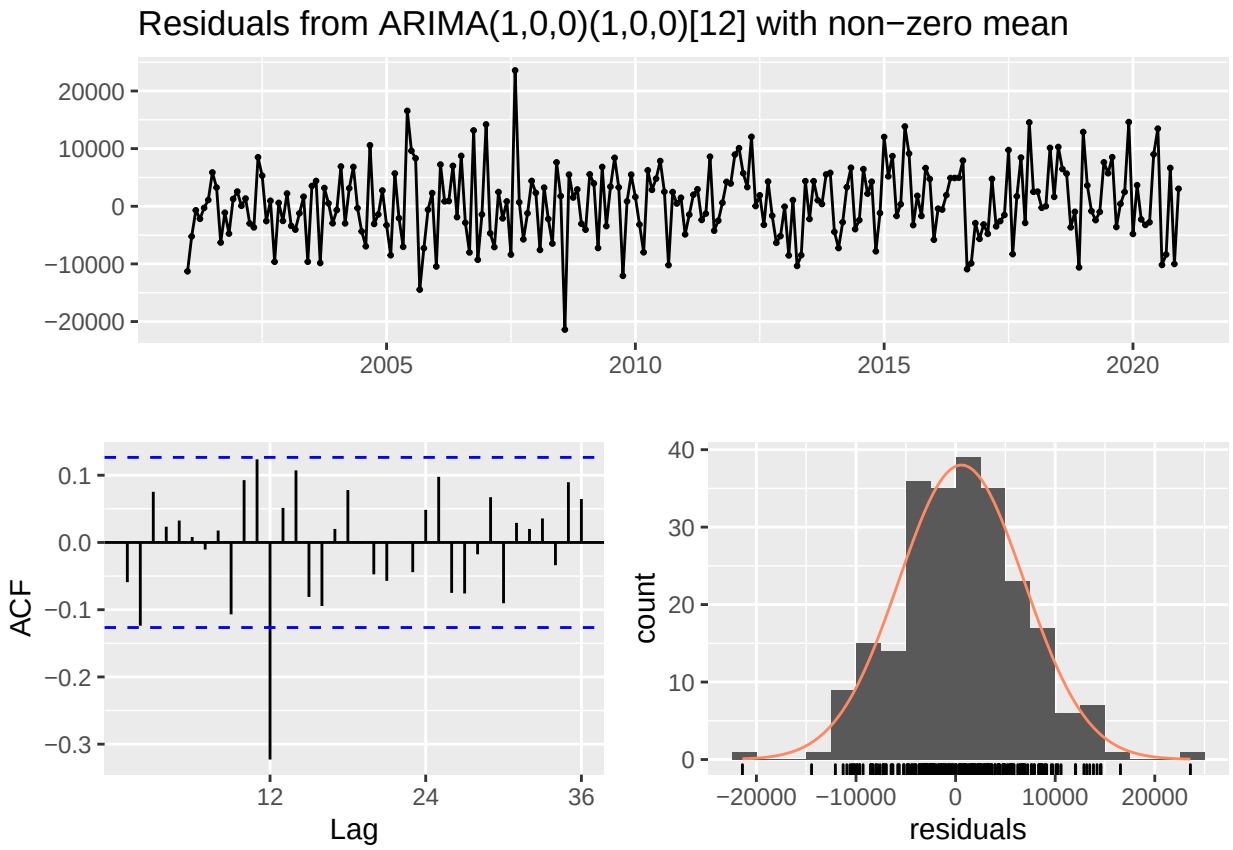



```
#For the order of the non-seasonal component, we focus on the early lags (ex.before lag=12). We can see
#As for the seasonal component, we are only intrested at seasonal lags 12, 24, 36 and so forth. We can
```

```
Arima.Sea.odel <-Arima(ts_NetGen[,"NG"], order=c(1,0,0), seasonal=list(order=c(1,0,0),include.mean=TRUE
summary(Arima.Sea.odel)
```

```
## Series: ts_NetGen[, "NG"]
## ARIMA(1,0,0)(1,0,0)[12] with non-zero mean
##
## Coefficients:
##          ar1    sar1      mean
##       0.8148  0.8979  90174.75
## s.e.  0.0374  0.0269  15570.78
##
## sigma^2 estimated as 40310701:  log likelihood=-2450.96
## AIC=4909.92   AICc=4910.09   BIC=4923.85
##
## Training set error measures:
##                     ME     RMSE     MAE        MPE     MAPE      MASE
## Training set 586.1283 6309.264 4974.36 0.02322463 6.056062 0.6081554
##                      ACF1
## Training set -0.05906034
```

```
checkresiduals(Arima.Sea.odel)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,0)(1,0,0)[12] with non-zero mean
## Q* = 53.911, df = 21, p-value = 0.0001017
##
## Model df: 3.   Total lags used: 24
```

```
#The residual series do look like a white noise series, because only one spike is not in two dotted lin
```

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.

```
#I can tell which ARIMA model is better representing the Natural Gas Series, and second one is better.B
```

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!).

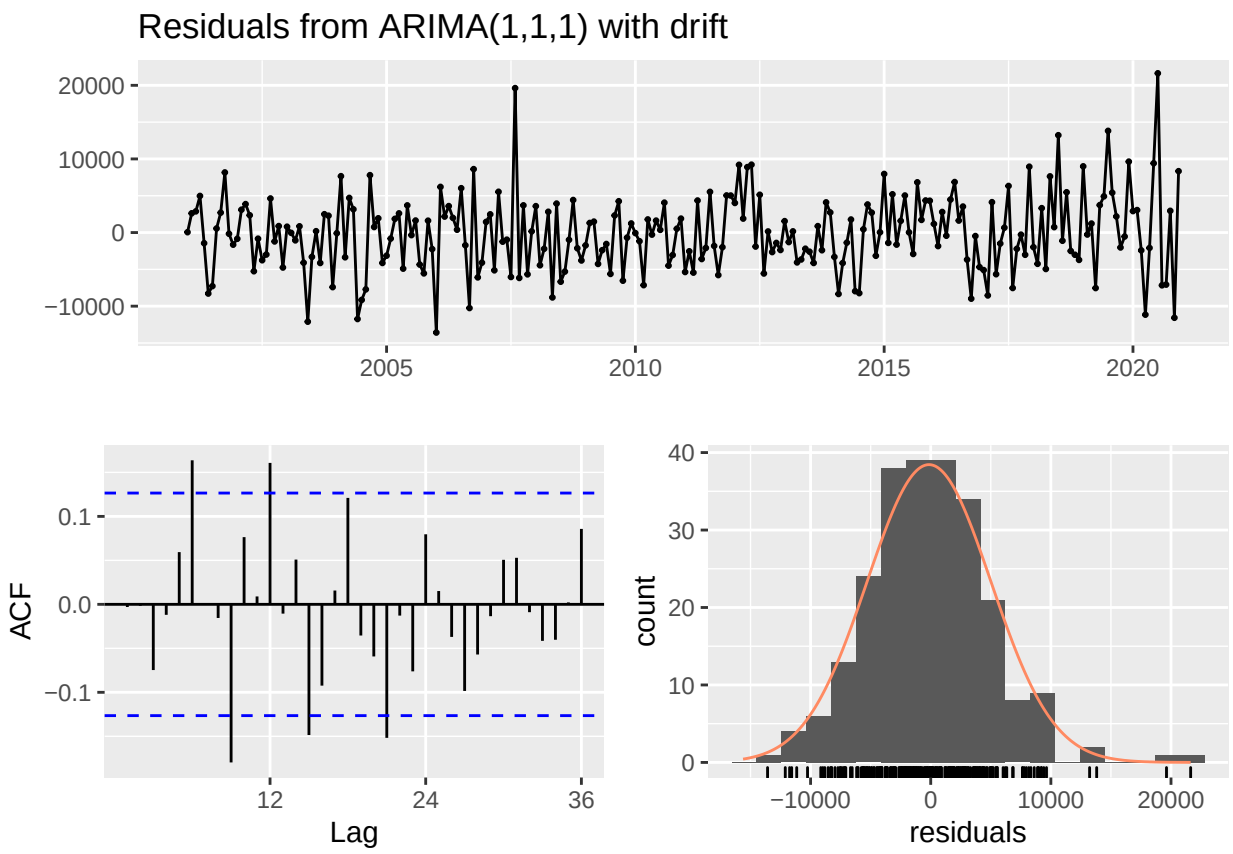
Q9

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?

```
SARIMA_autofit <- auto.arima(seasadj_ts_NetGen)
print(SARIMA_autofit)
```

```
## Series: seasadj_ts_NetGen
## 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
```

```
checkresiduals(SARIMA_autofit)
```



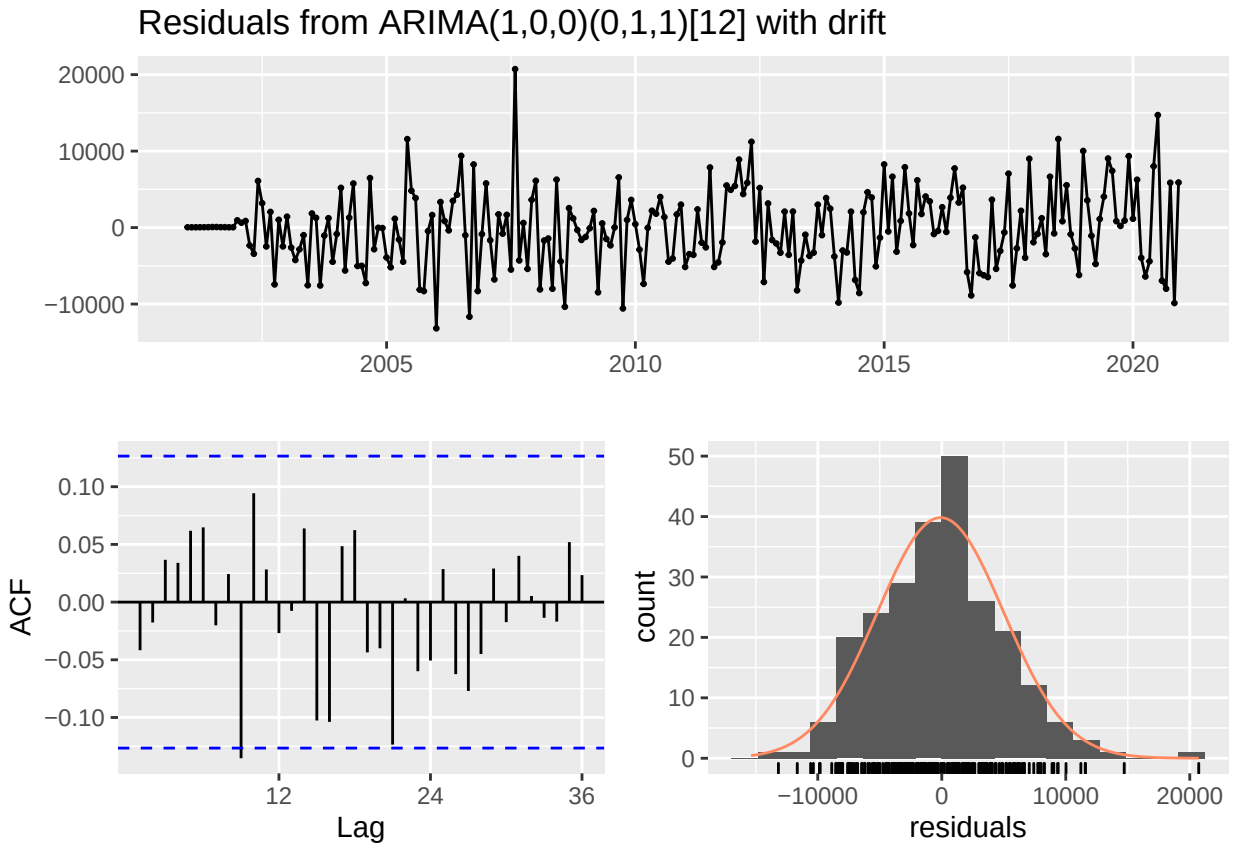

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,1) with drift
## Q* = 48.356, df = 21, p-value = 0.000615
##
## Model df: 3.   Total lags used: 24
```

```
# order is 1, and it only matches the p value 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?

```
SARIMA_autofit <- auto.arima(ts_NetGen[,"NG"])
checkresiduals(SARIMA_autofit)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,0)(0,1,1)[12] with drift
## Q* = 25.414, df = 21, p-value = 0.2297
##
## Model df: 3.   Total lags used: 24
```

```
#It does not matches what I specified in Q4.
```