ENV 797 - Time Series Analysis for Energy and Environment Applications | Spring 2025

Assignment 7 - Due date 03/06/25

Alex Lopez

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_Sp25.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
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 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(cowplot)
```

```
##
## Attaching package: 'cowplot'
##
## The following object is masked from 'package:lubridate':
##
##     stamp
```

Importing and processing the data set

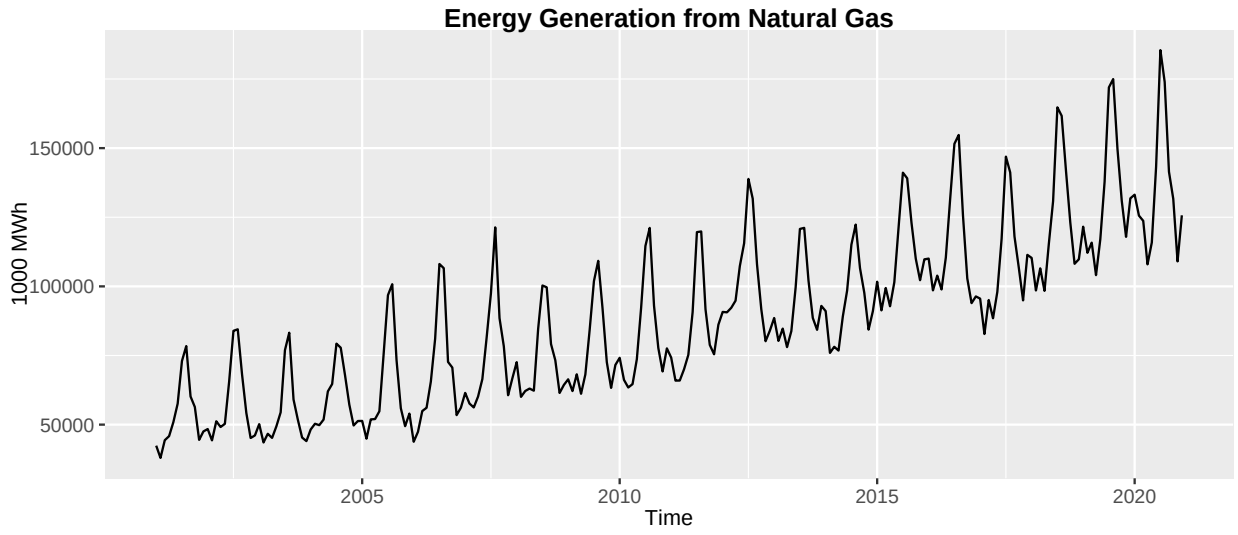
Consider the data from the file "Net_generation_United_States_all_sectors_monthly.csv". The data corresponds to the monthly net generation from January 2001 to December 2020 by source and is provided by the US Energy Information and Administration. **You will work with the natural gas column only**.

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.

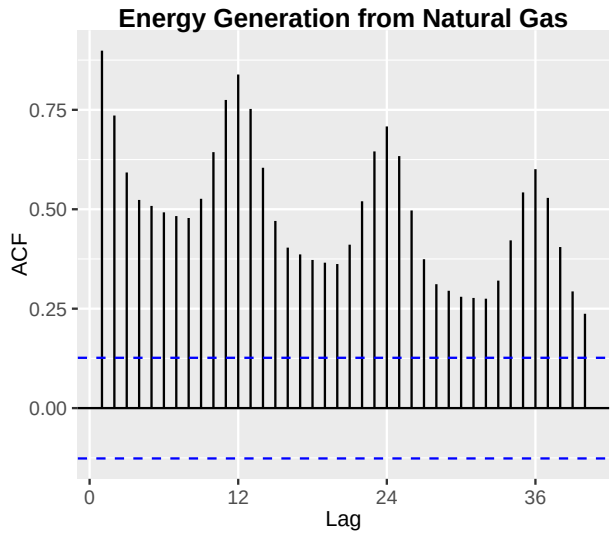
```
#set theme
theme_set(
  theme(plot.title = element_text(hjust = 0.5, face = "bold", size = 12),
          axis.title.x = element_text(size = 10),
          axis.title.y = element_text(size = 10, angle = 90, vjust = 0.5))
  )

net_generation <- read.csv(file="./Data/Net_generation_United_States_all_sectors_monthly.csv",
                           header=TRUE,skip=4)

#create date object and rename columns
net_gen_processed <-
  net_generation %>%
  mutate( Month = my(Month) ) %>%
```

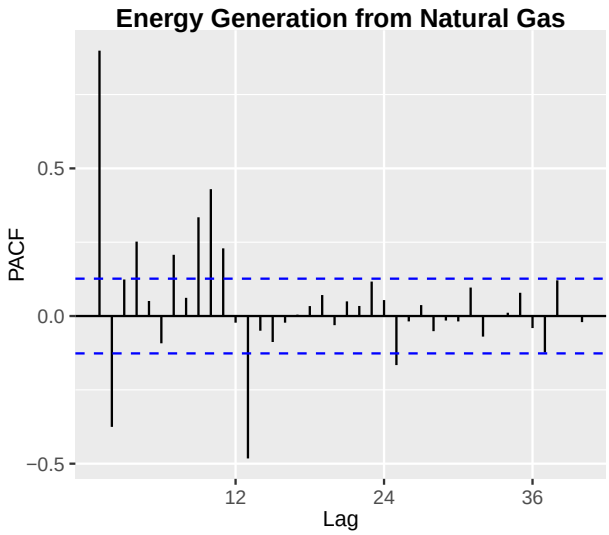
```
  rename( AllFuels = all.fuels..utility.scale..thousand.megawatthours ) %>%
  rename( Coal = coal.thousand.megawatthours ) %>%
  rename( NG = natural.gas.thousand.megawatthours ) %>%
  rename( Nuclear = nuclear.thousand.megawatthours ) %>%
  rename( Hydro = conventional.hydroelectric.thousand.megawatthours ) %>%
  arrange( Month )

net_gen_processed <- net_gen_processed[, c('Month', 'NG')]

#convert to ts object
ts_net_gen_processed <- ts(net_gen_processed[,2],
                           start = c(year(net_gen_processed$Month[1]),
                                     month(net_gen_processed$Month[1])),
                           frequency = 12)

#plotting time series over time
autoplot(ts_net_gen_processed) +
  ggtitle('Energy Generation from Natural Gas') +
  ylab("1000 MWh")
```




```
#ACF plot and PACF plot
plot_grid(
  autoplot(Acf(ts_net_gen_processed, lag = 40, plot=FALSE),
           main = "Energy Generation from Natural Gas"),
  autoplot(Pacf(ts_net_gen_processed, lag = 40, plot=FALSE),
           main = "Energy Generation from Natural Gas"),
  nrow=1
)
```

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

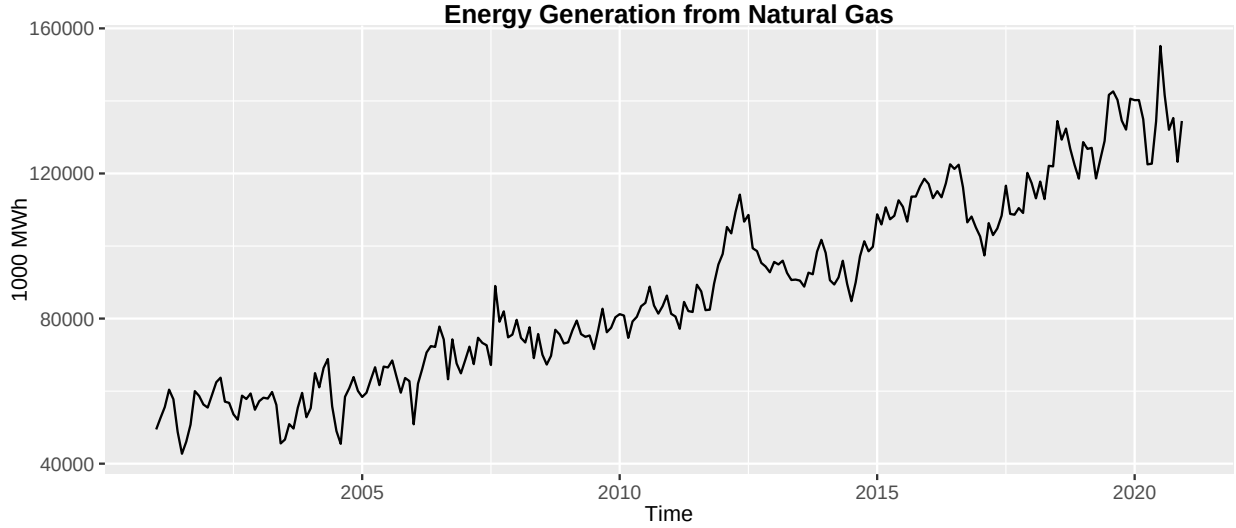




Q2

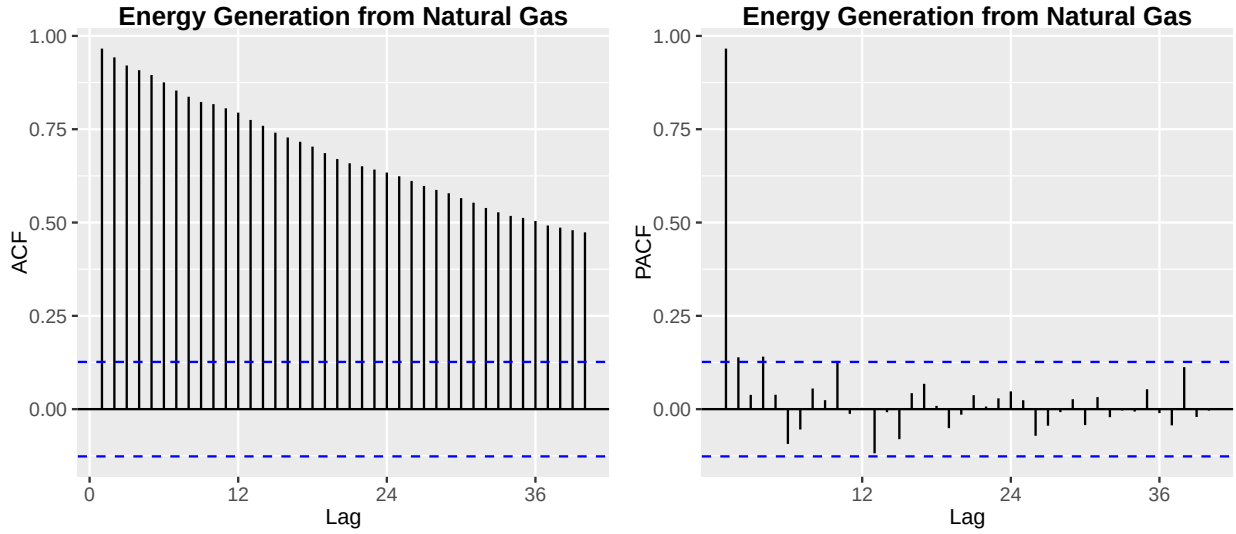
Using the *decompose*() and the *seasadj*() functions create a series without the seasonal component, i.e., a deseasonalized natural gas series. Plot the deseasonalized series over time and corresponding ACF and PACF. Compare with the plots obtained in Q1.

```
#decompose
decompose_net_gen <- decompose(ts_net_gen_processed, 'additive')

#deseason
deseasonal_net_gen <- seasadj(decompose_net_gen)

#plotting time series over time
autoplot(deseasonal_net_gen) +
  ggtitle('Energy Generation from Natural Gas') +
  ylab("1000 MWh")
```



```r
#ACF plot and PACF plot
plot_grid(
  autoplot(Acf(deseasonal_net_gen, lag = 40, plot=FALSE),
           main = "Energy Generation from Natural Gas"),
  autoplot(Pacf(deseasonal_net_gen, lag = 40, plot=FALSE),
           main = "Energy Generation from Natural Gas"),
  nrow=1
)
```

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



>Answer: Comparing the plots over time, we can obersve that the seasonality from the plot in Q1 was removed, with no observable seasonal fluctuations in the plot for Q2. Whereas the ACF plot in Q1 showed seasonal spikes at 12-month intervals, the ACF plot in Q2 shows no seasonal spikes. In Q1, the PACF plot shows a spike in signifcant correlation at lag 12, but the PACF plot in Q2 shows no spikes in correlation at any 12-month interval or any significant correlation at all after lag 1.

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
#ADF test
print(adf.test(deseasonal_net_gen, alternative = 'stationary'))
```

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

```
##
##  Augmented Dickey-Fuller Test
##
```

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

```
#Mann Kendall test
summary(MannKendall(deseasonal_net_gen))
```

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

> Answer: Regarding the ADF test, the p-value is less than 0.05, so we reject the null hypothesis. This means that the deseasonalized data is stationary. Regarding the Mann Kendall test, the p-value is less than 0.05, so we reject the null hypothesis in favor of the alternative and conclude that there is a statistically significant trend. Moreover, the tau is 0.843, which signifies a strong increasing trend over time. The results from both these tests suggest that even after removing seasonality, there is a strong increasing trend present in the deaseasonalized data. In other words, there is no stochastic trend according to the ADF test, but there is a deterministic trend according to the Mann Kendall test.

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.

```
#find out how many times we need to difference
n_differencing <- ndiffs(deseasonal_net_gen)
cat("Number of differencing needed: ", n_differencing)
```

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

> Answer: To determine the ARIMA model parameters p and q, we need to look at the ACF anf PACF plots, because the ACF provides the value of q and the PACF provides the value of p. Since the ACF plot shows a gradual decay over time, we can say that the deseasonalized series appears to be an AR model. For AR models, the PACF determines the order, and the order will be the last lag in the PACF plot after which there is a cut-off. In our case, it is lag 1, so p will be 1. Since there doesn't seem to be a cutoff in the ACF plot, which would determine q, we can say q is 0. To help determine how many times we should difference the deseasonalized series, the ARIMA model parameter d, we use the ndiffs() from package 'forecast'. The Mann Kendall test confirming there is a deterministic trend should also make us set d to 1. The ARIMA model should therefore be ARIMA(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()` or `print()` function to print.

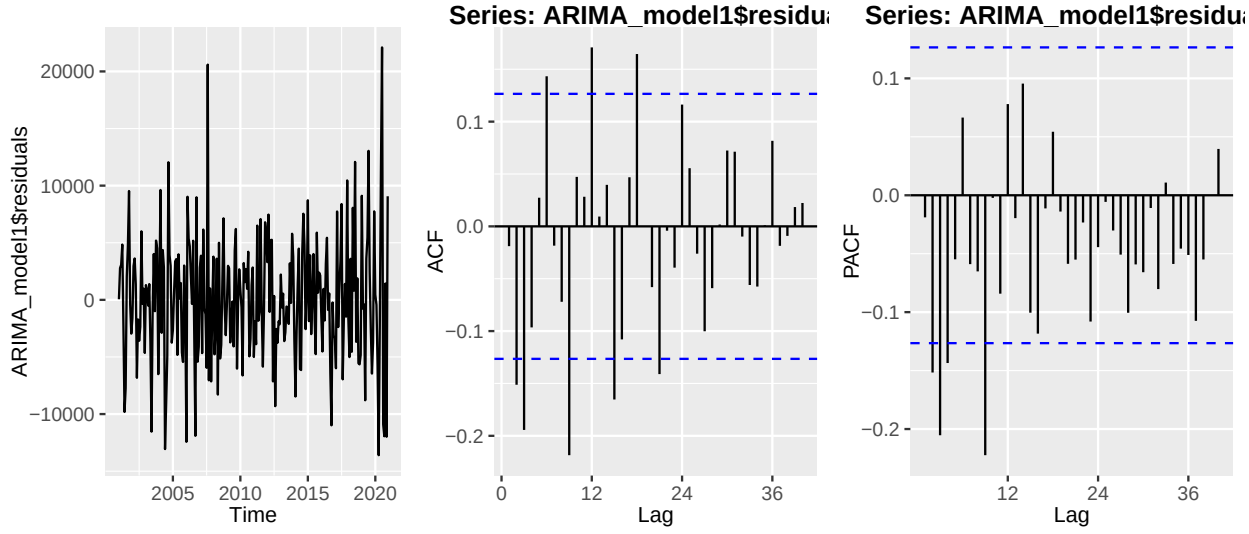
```
ARIMA_model1 <- Arima(deseasonal_net_gen, order=c(1,1,0), include.drift=TRUE)
print(ARIMA_model1)
```

```
## Series: deseasonal_net_gen
## 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?

```
plot_grid(
  autoplot(ARIMA_model1$residuals),
  autoplot(Acf(ARIMA_model1$residuals,lag.max=40, plot = FALSE)),
  autoplot(Pacf(ARIMA_model1$residuals,lag.max=40, plot = FALSE)),
  nrow=1
)
```



Answer: The residual series do not really look like a white noise series for a couple of reasons. From just looking at the time plot, it is hard to tell, because the values do look somewhat random and there is no discernable pattern. However, in the ACF and PACF plots, there are a few values outside the blue significance bounds, which confirms that the model may not have completely captured all the patterns in the series.

Modeling the original series (with seasonality)

Q7

Repeat Q3-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
print(adf.test(ts_net_gen_processed, alternative = 'stationary'))
```

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

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

```
#Mann Kendall test
summary(MannKendall(ts_net_gen_processed))
```

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

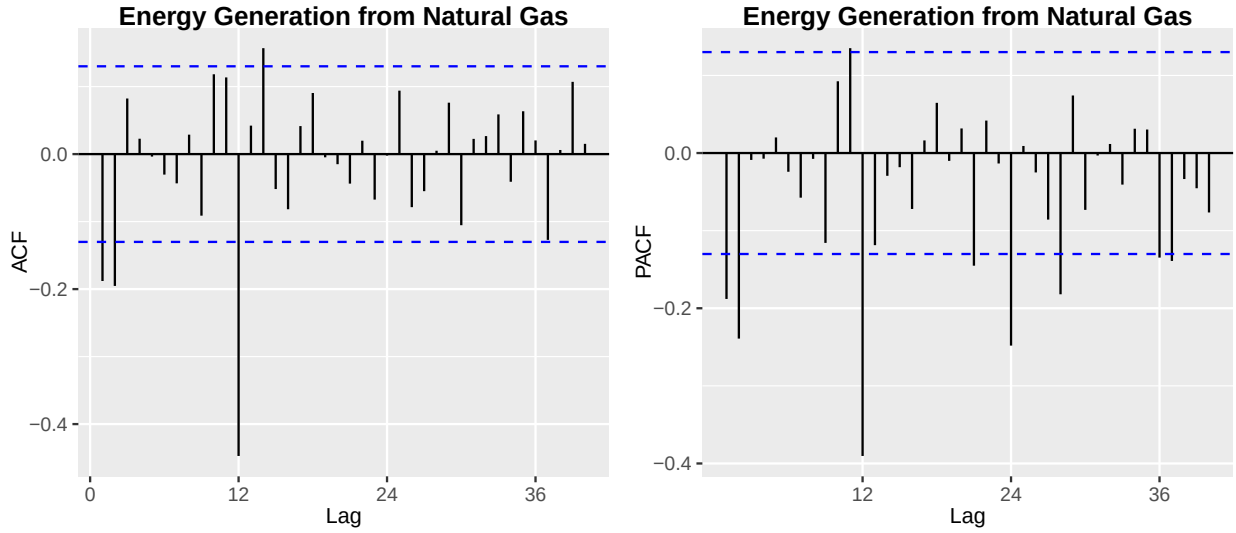
> Regarding the ADF test, the p-value is less than 0.05, so we reject the null hypothesis. This means that the original series is stationary. Regarding the Mann Kendall test, the p-value is less than 0.05, so we reject the null hypothesis in favor of the alternative and conclude that there is a statistically significant trend in the original series. Moreover, the tau is 0.651, which signifies a an increasing trend over time. There is no stochastic trend according to the ADF test, but there is a deterministic trend according to the Mann Kendall test, just like in the deseasonalized data.

```
ns_diff <- nsdiffs(ts_net_gen_processed)
cat("Number of seasonal differencing needed: ", ns_diff)
```

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

```
#diff done on orig series
ts_net_gen_diff <- diff(ts_net_gen_processed, lag = 1, differences = 1)
ts_net_gen_diff <- diff(ts_net_gen_diff, lag = 12, differences = 1)

#ACF and PACF plots of differenced original series
plot_grid(
  autoplot(Acf(ts_net_gen_diff, lag = 40, plot=FALSE),
           main = "Energy Generation from Natural Gas"),
  autoplot(Pacf(ts_net_gen_diff, lag = 40, plot=FALSE),
           main = "Energy Generation from Natural Gas"),
  nrow=1
)
```

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

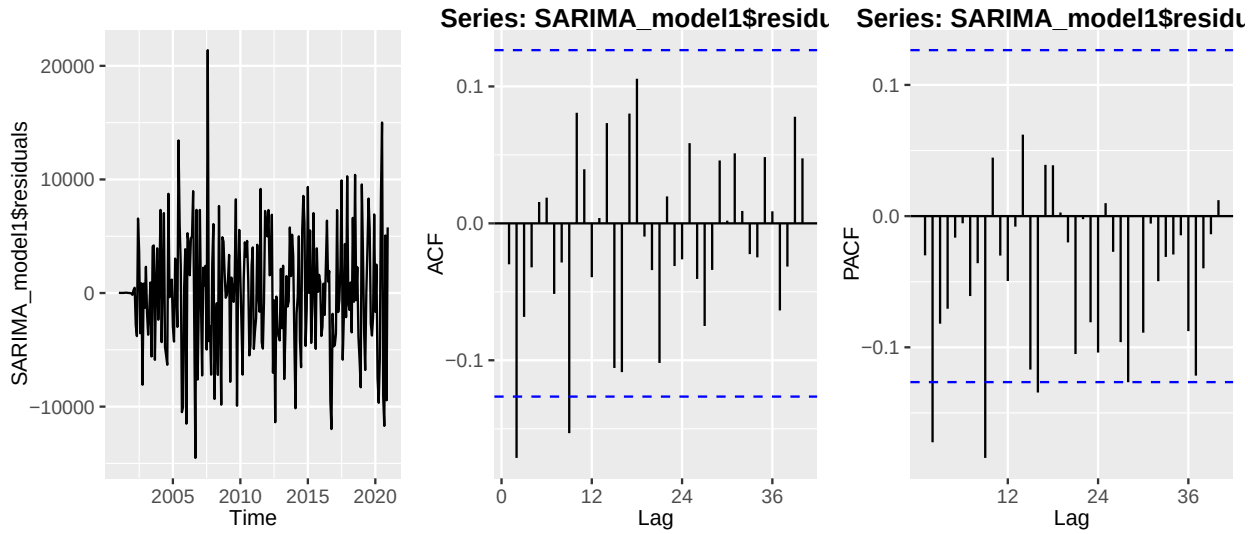


To determine the non-seasonal model parameters p, d, and q, we can use the same ones we used from the deaseasonalized series. To determine how many times we should difference the original series, the seasonal parameter D, we use the nsdiffs() from package 'forecast'. This gives us D = 1. To determine the seasonal model parameters P and Q, we need to look at the seasonal lags only in the ACF anf PACF plots after twice differencing. Because there is a negative spike in the ACF at lag 12 and negative spikes at lags 12, 24, and 36 in the PACF, we can conclude that Q = 1. There are no positive spikes in lags 12, 24, and 36 in the ACF plot, so P can't be 1; it is 0. P and Q can't be more than 1, so this supports our conclusion. The model should therefore be ARIMA(1,1,0)(0,1,1)[12].

```
SARIMA_model1 <- Arima(ts_net_gen_processed,
                       order=c(1,1,0),
                       seasonal=c(0,1,1),
                       include.drift=FALSE)
print(SARIMA_model1)
```

```
## Series: ts_net_gen_processed
## 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
```

```
plot_grid(
  autoplot(SARIMA_model1$residuals),
  autoplot(Acf(SARIMA_model1$residuals,lag.max=40, plot = FALSE)),
  autoplot(Pacf(SARIMA_model1$residuals,lag.max=40, plot = FALSE)),
  nrow=1
)
```



>This time, the residual series look more like a white noise series, because the values still look somewhat random in the time plot and in the ACF and PACF plots, there are fewer values outside the blue significance bounds, which confirms that this model, although not completely capturing all the patterns in the series, fits the data better.

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: Since there are fewer values outside the blue significance bounds and the AIC decreased with the second model, the second model does a better job at representing the natural gas series. This is not entirely a fair comparison, because whereas we twice differenced the original series for the second model to remove trend and seasonality, we only removed seasonality from the original series for the first 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 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?

```r
ARIMA_autofit <- auto.arima(deseasonal_net_gen, max.D=0, max.P = 0,max.Q=0)
print(ARIMA_autofit)
```

```
## Series: deseasonal_net_gen
## 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 model automatically chosen, ARIMA(1,1,1) does not match with the model I specified in Q4, ARIMA(1,1,0). Parameter q was different.

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

```
## Series: ts_net_gen_processed
## 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: The model automatically chosen, ARIMA(1,0,0)(0,1,1)[12] does not match with the model I specified in Q7, ARIMA(1,1,0)(0,1,1)[12]. Parameter d was different.