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

Assignment 7 - Due date 03/07/24

Jenn McNeill

## 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

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
```

```r
library(Kendall)
library(tseries)
library(outliers)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr   1.1.3     v stringr 1.5.0
## v forcats 1.0.0     v tibble  3.2.1
## v purrr   1.0.2     v tidyr   1.3.0
## v readr   2.1.4

## -- 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 error
```

```r
library(smooth)
```

```
## Loading required package: greybox
## Package "greybox", v2.0.0 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", v4.0.0
```

```r
library(dplyr)
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.

```
getwd()
```

```
## [1] "/Users/jennifermcneill/TSA_Spring2024/TSA_Spring2024"
```

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

#Inspect data
head(net_generation)
```

```
##      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
```

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

#Create a processed dataframe
natural_gas_processed <-
  net_generation %>%
  mutate( Month = my(Month) ) %>%
```

```r
  select( Month, natural.gas.thousand.megawatthours) %>%
  rename( natural_gas = natural.gas.thousand.megawatthours ) %>%
  arrange( Month )

#Check for NA
head(natural_gas_processed)
```

```
##        Month natural_gas
## 1 2001-01-01    42388.66
## 2 2001-02-01    37966.93
## 3 2001-03-01    44364.41
## 4 2001-04-01    45842.75
## 5 2001-05-01    50934.21
## 6 2001-06-01    57603.15
```

```r
summary(natural_gas_processed)
```

```
##      Month             natural_gas
##  Min.   :2001-01-01   Min.   : 37967
##  1st Qu.:2005-12-24   1st Qu.: 62245
##  Median :2010-12-16   Median : 84415
##  Mean   :2010-12-16   Mean   : 88028
##  3rd Qu.:2015-12-08   3rd Qu.:108385
##  Max.   :2020-12-01   Max.   :185445
```

```r
#No NA detected

#Create a time series object
ts_natural_gas <- ts(natural_gas_processed[,2],
                     start=c(year(natural_gas_processed$Month[1]),
                             month(natural_gas_processed$Month[1])),
                             frequency=12)

#Check head and tail of the time series object
head(ts_natural_gas,15)
```

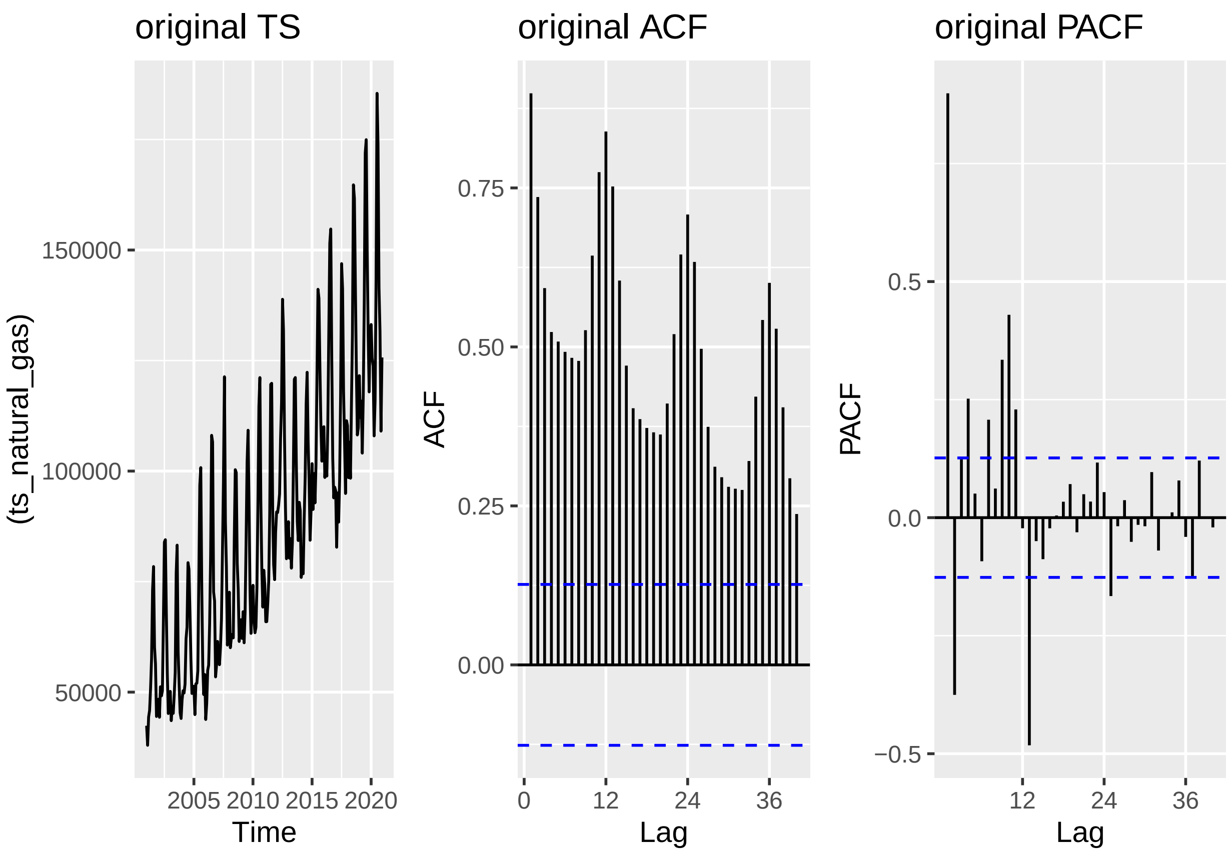
```
##           Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
## 2001 42388.66 37966.93 44364.41 45842.75 50934.21 57603.15 73030.14 78409.80
## 2002 48412.83 44308.43 51214.46
##           Sep      Oct      Nov      Dec
## 2001 60181.14 56376.44 44490.62 47540.86
## 2002
```

```r
tail(ts_natural_gas,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
```

```
#Plot the time series over time, ACF, and PACF
plot_grid(
  autoplot((ts_natural_gas), main = "original TS"),
  autoplot(Acf(ts_natural_gas, lag = 40, plot=FALSE), main = "original ACF"),
  autoplot(Pacf(ts_natural_gas, lag = 40, plot=FALSE), main = "original PACF"),
  ncol=3
)
```

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




**Q2**

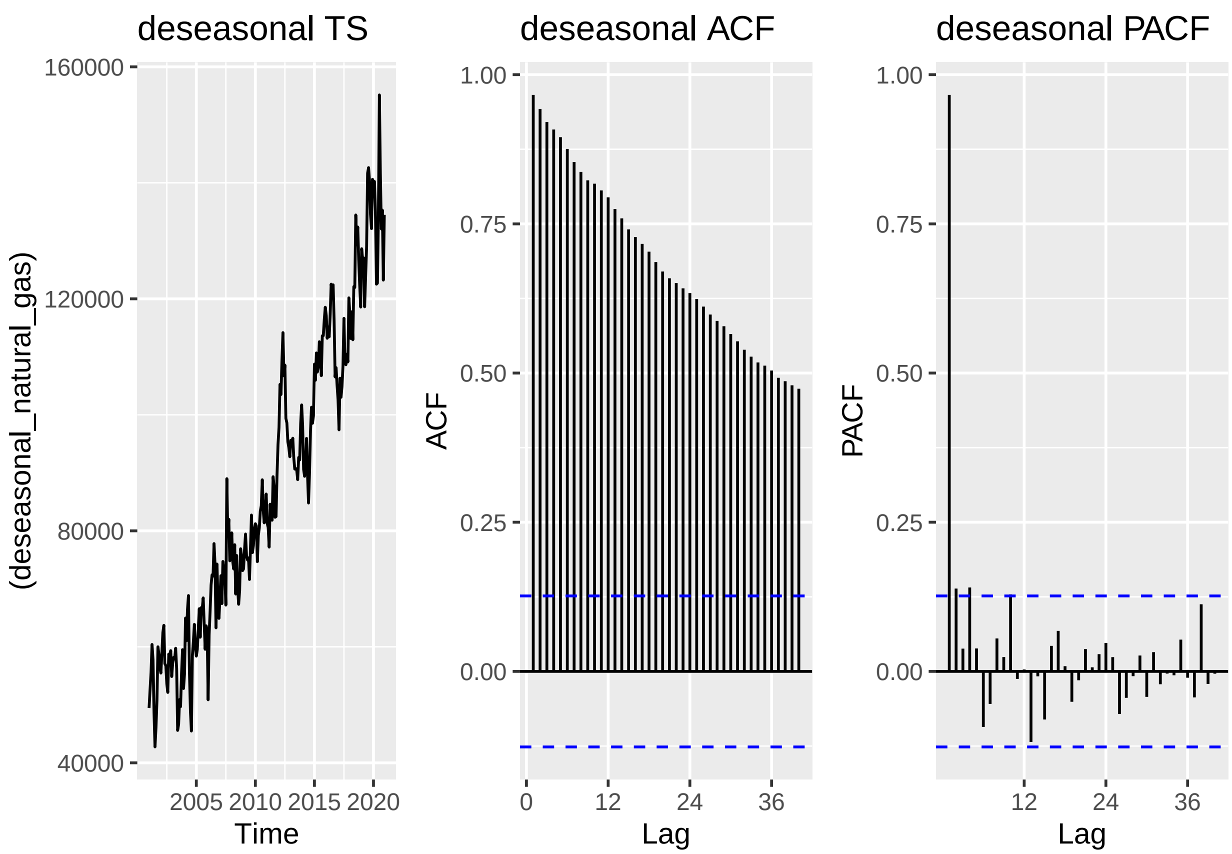
Using the *decompose*() or *stl*() and the *seasadj*() functions create a series without the seasonal component, i.e., a deseasonalized natural gas series. Plot the deseasonalized series over time and corresponding ACF and PACF. Compare with the plots obtained in Q1.

```
decompose_natural_gas <- decompose(ts_natural_gas,"additive")
deseasonal_natural_gas <- seasadj(decompose_natural_gas)

plot_grid(
  autoplot((deseasonal_natural_gas), main = "deseasonal TS"),
  autoplot(Acf(deseasonal_natural_gas, lag = 40, plot=FALSE), main = "deseasonal ACF"),
  autoplot(Pacf(deseasonal_natural_gas, lag = 40, plot=FALSE), main = "deseasonal PACF"),
```

```
  ncol=3
)
```

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




These plots are different than the plots obtained in Q1 because they have removed seasonality as a factor. The time series no longer shows evenly spaced fluctuations, the ACF plot shows a steady decay, and the PACF plot shows one significant lag.

## 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.

```
MK_deseasonal_natural_gas <- MannKendall(deseasonal_natural_gas)
print(summary(MK_deseasonal_natural_gas))
```

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

```r
ADF_deseasonal_natural_gas <- adf.test(deseasonal_natural_gas,alternative="stationary")
```

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

```r
print(ADF_deseasonal_natural_gas)
```

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

The results of the Seasonal Mann Kendall show that there is a trend in the data because the p-value is 2.22e-16, which is less than .05. According to this p-value, we can reject the null hypothesis that there is no trend. Additionally, the score is positive, which means that the trend is increasing. This matches what I observed when plotting the time series, which is that the series has a positive trend over time. The ADF results in a p-value of 0.01, which is less than .05. This means we reject the null hypothesis that the series is non-stationary and we know that we will have to difference the series to achieve stationarity.

### 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.

```r
n_diff <- ndiffs(ts_natural_gas)
```

The results from the ADF test tell me that I have to difference the series to achieve stationarity, so I know that my d order will be non-zero in my ARIMA model. I will run the ndiffs() function to see how many times I should difference. Since the deseasonal ACF has non-zero autocorrelation values that decay with the lag and the deseasonal PACF has one significant lag, I am assuming that this is an AR process and will assume that my p order will also be non-zero. The MA term will be trial and error.

### 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
Model_111 <- Arima(deseasonal_natural_gas,order=c(1,1,1),include.drift=TRUE)
print(Model_111)
```

```
## 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 = 26980609:  log likelihood = -2383.11
## AIC=4774.21   AICc=4774.38   BIC=4788.12
```

```r
compare_aic <- data.frame(Model_111$aic)

Model_011 <- Arima(deseasonal_natural_gas,order=c(0,1,1),include.drift=TRUE)
print(Model_011)
```

```
## Series: deseasonal_natural_gas
## ARIMA(0,1,1) with drift
##
## Coefficients:
##           ma1     drift
##       -0.2296  344.5131
## s.e.   0.0866  271.9198
##
## sigma^2 = 29946343:  log likelihood = -2395.33
## AIC=4796.66   AICc=4796.77   BIC=4807.09
```

```r
compare_aic <- data.frame(compare_aic,Model_011$aic)

Model_211 <- Arima(deseasonal_natural_gas,order=c(2,1,1),include.drift=TRUE)
print(Model_211)
```

```
## Series: deseasonal_natural_gas
## ARIMA(2,1,1) with drift
##
## Coefficients:
##          ar1     ar2      ma1     drift
##       0.7057  0.0017  -0.9798  359.4921
## s.e.  0.0710  0.0707   0.0360   29.3046
##
## sigma^2 = 27094287:  log likelihood = -2383.11
## AIC=4776.21   AICc=4776.47   BIC=4793.59
```

```r
compare_aic <- data.frame(compare_aic,Model_211$aic)

Model_112 <- Arima(deseasonal_natural_gas,order=c(1,1,2),include.drift=TRUE)
print(Model_112)
```

```
## Series: deseasonal_natural_gas
## ARIMA(1,1,2) with drift
##
## Coefficients:
##          ar1      ma1     ma2     drift
##       0.7081  -0.9823  0.0025  359.4980
## s.e.  0.0939   0.1189  0.0982   29.3122
```

```
##
## sigma^2 = 27094333:  log likelihood = -2383.11
## AIC=4776.21   AICc=4776.47   BIC=4793.59
```

```
compare_aic <- data.frame(compare_aic,Model_112$aic)

Model_212 <- Arima(deseasonal_natural_gas,order=c(2,1,2),include.drift=TRUE)
print(Model_212)
```

```
## Series: deseasonal_natural_gas
## ARIMA(2,1,2) with drift
##
## Coefficients:
##           ar1     ar2      ma1      ma2     drift
##       -0.2337  0.7179  -0.0574  -0.9425  359.3697
## s.e.   0.0541  0.0478   0.0493   0.0484   17.2642
##
## sigma^2 = 26504431:  log likelihood = -2381.07
## AIC=4774.13   AICc=4774.49   BIC=4794.99
```

```
compare_aic <- data.frame(compare_aic,Model_212$aic)

print(compare_aic)
```

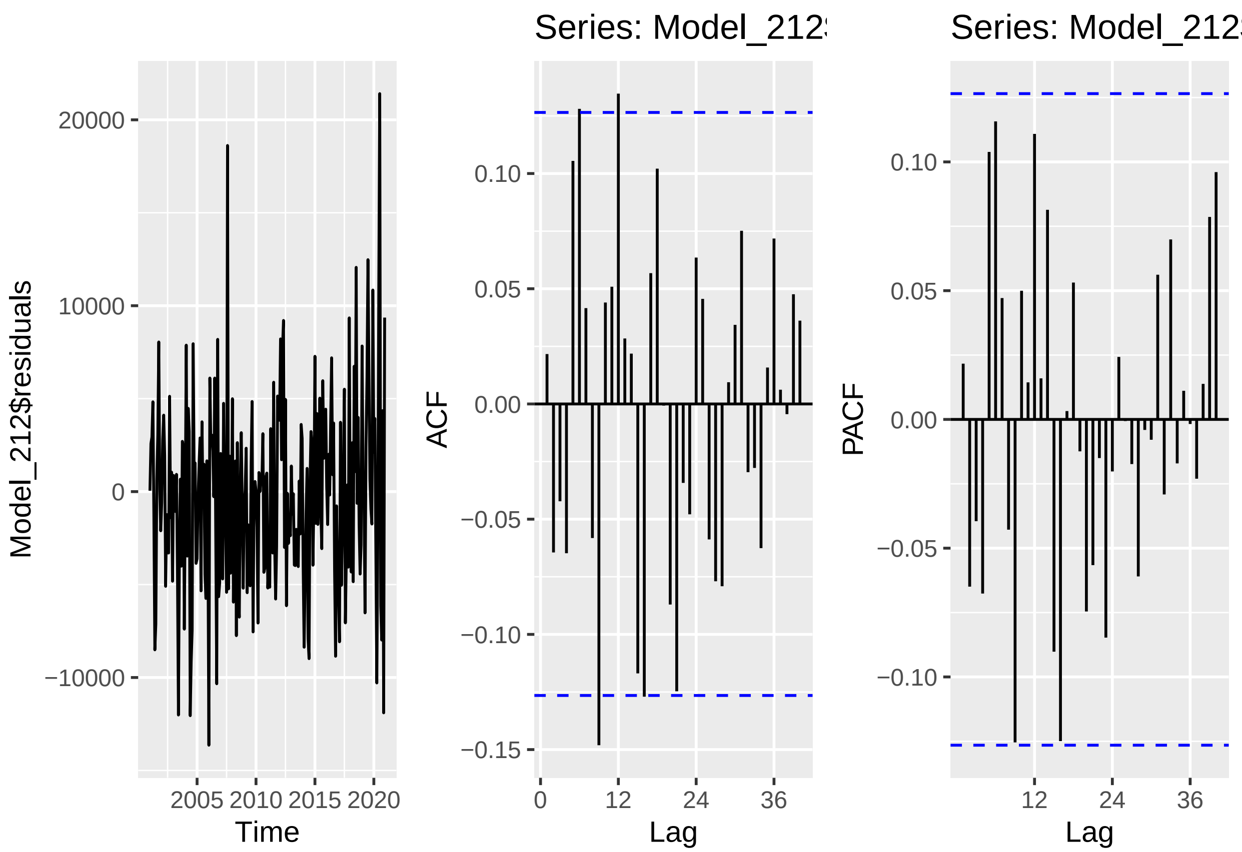
```
##   Model_111.aic Model_011.aic Model_211.aic Model_112.aic Model_212.aic
## 1      4774.213      4796.664      4776.212      4776.212      4774.131
```

In the ARIMA that is the best fit, we want the residuals to be white noise centered at 0 with no time dependence between the error. The error at t should not depend on error at t-1. The ACF and PACF plots should have insignificant coefficients always between the two dashed lines. The value for the AIC will be the lowest in the ARIMA that is the best fit. I predict that order (2,1,2) will be the best fit in this case because it best meets the above criteria the best and has the lowest AIC value of the five orders that I tested.

### 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(Model_212$residuals),
  autoplot(Acf(Model_212$residuals,lag.max=40, plot = FALSE)),
  autoplot(Pacf(Model_212$residuals,lag.max=40, plot = FALSE)),
  nrow=1)
```




The residual series does look like a white noise series! It is centered at zero and appears to have minimums and maximums at randomly spaced time increments.

## 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 $Q$.

The results from the ADF test tell me that I have to difference the series to achieve stationarity. I will use ndiffs() and nsdiffs() to determine the number of times I should difference seasonally and non-seasonally, and I assume that my d order and D order will both be non-zero. Since the original ACF has non-zero autocorrelation values that decay with the lag and show seasonal patterns and the original PACF has significant lags with a cutoff, I am assuming that this is an SAR process and will assume that my p order and P order will also be non-zero. The seasonal lags occur at increments of 12 lags. The MA terms q and Q will be trial and error. I remember that with seasonal ARIMAs you can only have a P term or a Q term. I expect to see a P term instead of a Q term because the ACF shows multiple seasonal lags and decays over time.

```
ns_diff <- nsdiffs(ts_natural_gas)

Model_111011 <- Arima(ts_natural_gas,order=c(1,1,1),seasonal=c(0,1,1),include.drift=FALSE)
print(Model_111011)
```

```
## Series: ts_natural_gas
```

```
## ARIMA(1,1,1)(0,1,1)[12]
##
## Coefficients:
##          ar1      ma1     sma1
##       0.7323  -0.9819  -0.7017
## s.e.  0.0504   0.0183   0.0563
##
## sigma^2 = 27922085:  log likelihood = -2272.2
## AIC=4552.39   AICc=4552.57   BIC=4566.09
```

```
compare_aic_seas <- data.frame(Model_111011$aic)

Model_111110 <- Arima(ts_natural_gas,order=c(1,1,1),seasonal=c(1,1,0),include.drift=FALSE)
print(Model_111110)
```

```
## Series: ts_natural_gas
## ARIMA(1,1,1)(1,1,0)[12]
##
## Coefficients:
##          ar1      ma1     sar1
##       0.7722  -1.0000  -0.4526
## s.e.  0.0432   0.0213   0.0595
##
## sigma^2 = 32606981:  log likelihood = -2287.56
## AIC=4583.12   AICc=4583.3   BIC=4596.82
```

```
compare_aic_seas <- data.frame(compare_aic_seas,Model_111110$aic)

Model_110110 <- Arima(ts_natural_gas,order=c(1,1,0),seasonal=c(1,1,0),include.drift=FALSE)
print(Model_110110)
```

```
## Series: ts_natural_gas
## ARIMA(1,1,0)(1,1,0)[12]
##
## Coefficients:
##           ar1     sar1
##       -0.1897  -0.4659
## s.e.   0.0653   0.0589
##
## sigma^2 = 35370456:  log likelihood = -2295.37
## AIC=4596.74   AICc=4596.84   BIC=4607.01
```

```
compare_aic_seas <- data.frame(compare_aic_seas,Model_110110$aic)

Model_011110 <- Arima(ts_natural_gas,order=c(0,1,1),seasonal=c(1,1,0),include.drift=FALSE)
print(Model_011110)
```

```
## Series: ts_natural_gas
## ARIMA(0,1,1)(1,1,0)[12]
##
## Coefficients:
##           ma1     sar1
```

```
##       -0.2698  -0.4594
## s.e.   0.0728   0.0591
##
## sigma^2 = 34791709:  log likelihood = -2293.47
## AIC=4592.94   AICc=4593.05   BIC=4603.21
```

```r
compare_aic_seas <- data.frame(compare_aic_seas,Model_011110$aic)

Model_212011 <- Arima(ts_natural_gas,order=c(2,1,2),seasonal=c(0,1,1),include.drift=FALSE)
print(Model_212011)
```

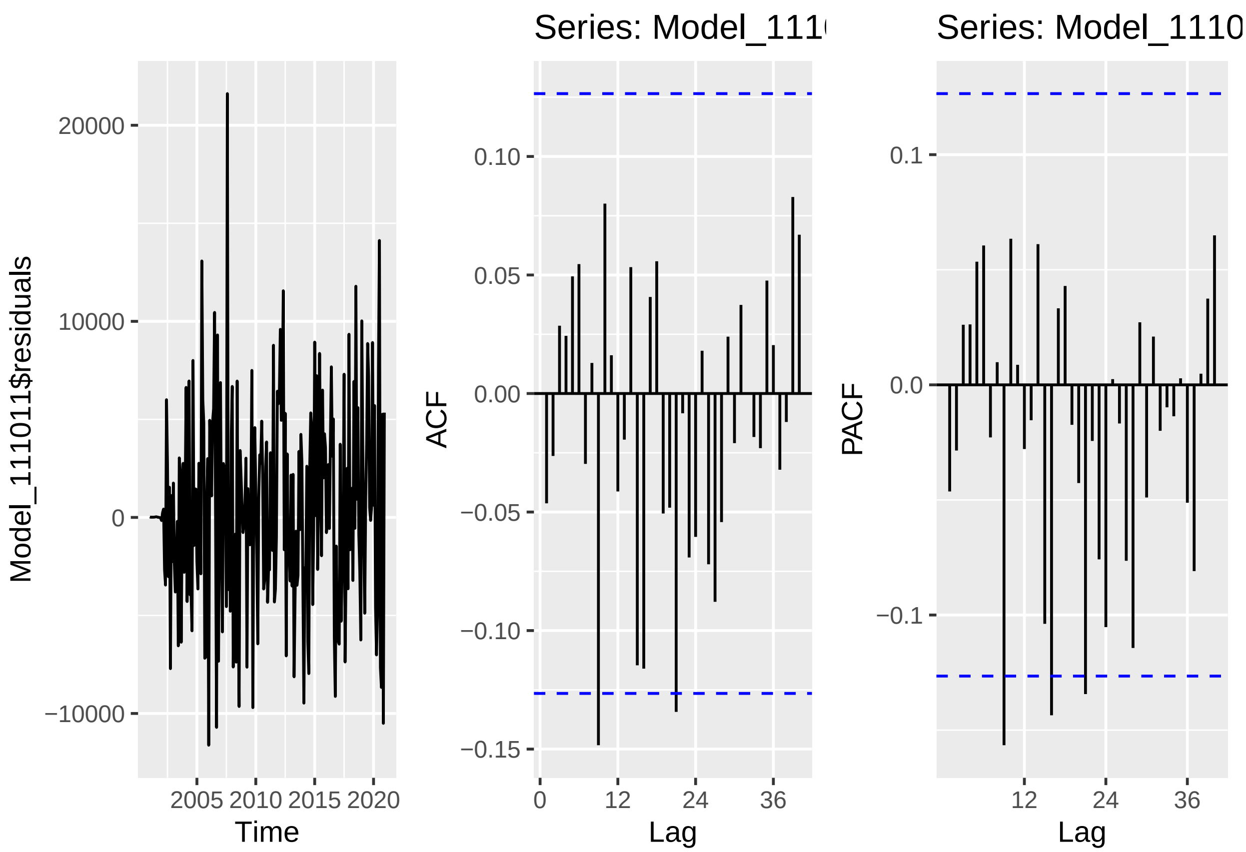
```
## Series: ts_natural_gas
## ARIMA(2,1,2)(0,1,1)[12]
##
## Coefficients:
##           ar1     ar2      ma1      ma2     sma1
##       -0.2162  0.7057  -0.0489  -0.9156  -0.7165
## s.e.   0.0834  0.0648   0.0767   0.0737   0.0563
##
## sigma^2 = 28013060:  log likelihood = -2271.66
## AIC=4555.33   AICc=4555.71   BIC=4575.88
```

```r
compare_aic_seas <- data.frame(compare_aic_seas,Model_212011$aic)

print(compare_aic_seas)
```

```
##   Model_111011.aic Model_111110.aic Model_110110.aic Model_011110.aic
## 1         4552.393         4583.117         4596.737         4592.939
##   Model_212011.aic
## 1         4555.329
```

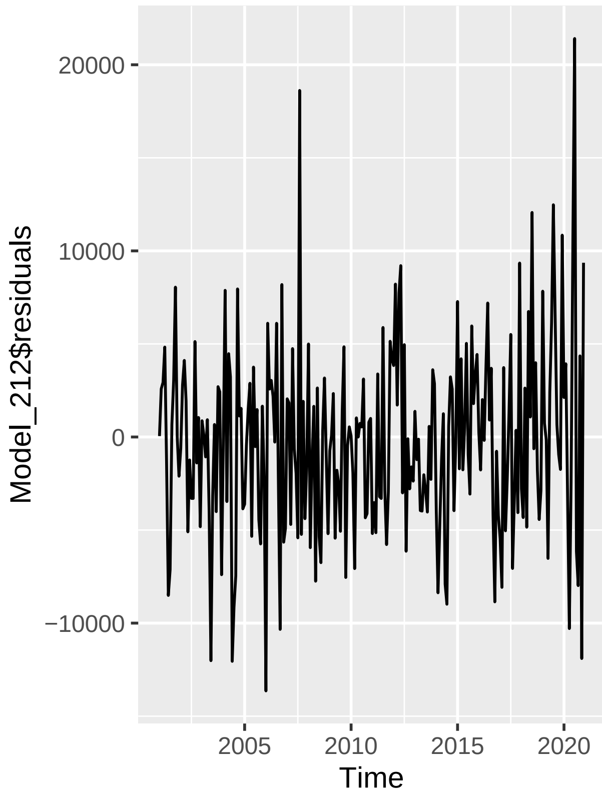
```r
plot_grid(
  autoplot(Model_111011$residuals),
  autoplot(Acf(Model_111011$residuals,lag.max=40, plot = FALSE)),
  autoplot(Pacf(Model_111011$residuals,lag.max=40, plot = FALSE)),
  nrow=1)
```

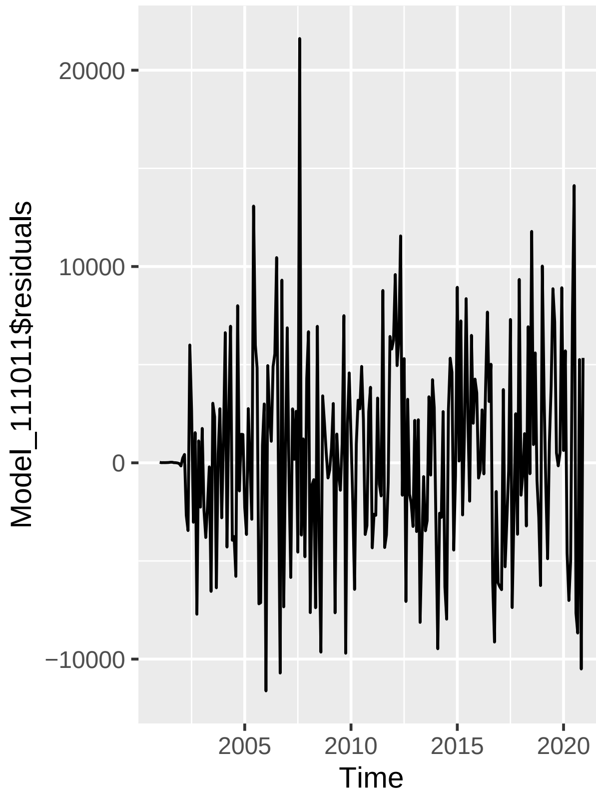



### Q8

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.

```
plot_grid(
  autoplot(Model_212$residuals),
  autoplot(Model_111011$residuals),
  nrow=1)
```




I cannot tell which ARIMA model is better representing the natural gas series because both of the residuals are white noise. Both plots show residuals centered around zero and no patterns of detected seasonality. It is difficult to compare the seasonal Arima and non-seasonal Arima because both of these models can be good fits for the data depending on whether the seasonality is a term that can be reliably removed. Removing the seasonality before modeling the Arima has benefits if you feel confident in your ability to remove it accurately. Keeping the seasonality in and modeling with it as a component may be a safer option, but if the seasonality is not all perfectly spaced in 12 month increments, it might actually be introducing more error into the 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?

```
deseasonal_autofit <- auto.arima(deseasonal_natural_gas,max.D=0,max.P=0,max.Q=0)
print(deseasonal_autofit)
```

```
## 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 = 26980609:  log likelihood = -2383.11
## AIC=4774.21   AICc=4774.38   BIC=4788.12
```

The order of the best ARIMA model is (1,1,1) with drift. I had modeled that the AIC values for orders (1,1,1) with drift and (2,1,2) with drift were almost exactly the same, so I am not surprised that the Auto ARIMA gave (1,1,1) as the best order.

### 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
seasonal_autofit <- auto.arima(ts_natural_gas)
print(seasonal_autofit)
```

```
## Series: ts_natural_gas
## 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
```

The order of the best seasonal ARIMA model is (1,0,0)(0,1,1)[12] with drift. This differs from the order that I specified in Q7, which was (1,1,1)(0,1,1)[12]. I am surprised to see that the moving average non-seasonal term has been removed when the auto ARIMA fits the model. I am also surprised to see that the auto ARIMA does not detect a seasonal autoregressive term.