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

Assignment 7 - Due date 03/07/24

David Robinson

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

```r
#Load/install required package here

library(lubridate)
```

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

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

```r
library(ggplot2)
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 4.3.2

## 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(cowplot)
```

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

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

```
## Warning: package 'sarima' was built under R version 4.3.2
```

```
## Loading required package: stats4
##
## Attaching package: 'sarima'
##
## The following object is masked from 'package:stats':
##
##     spectrum
```

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**.

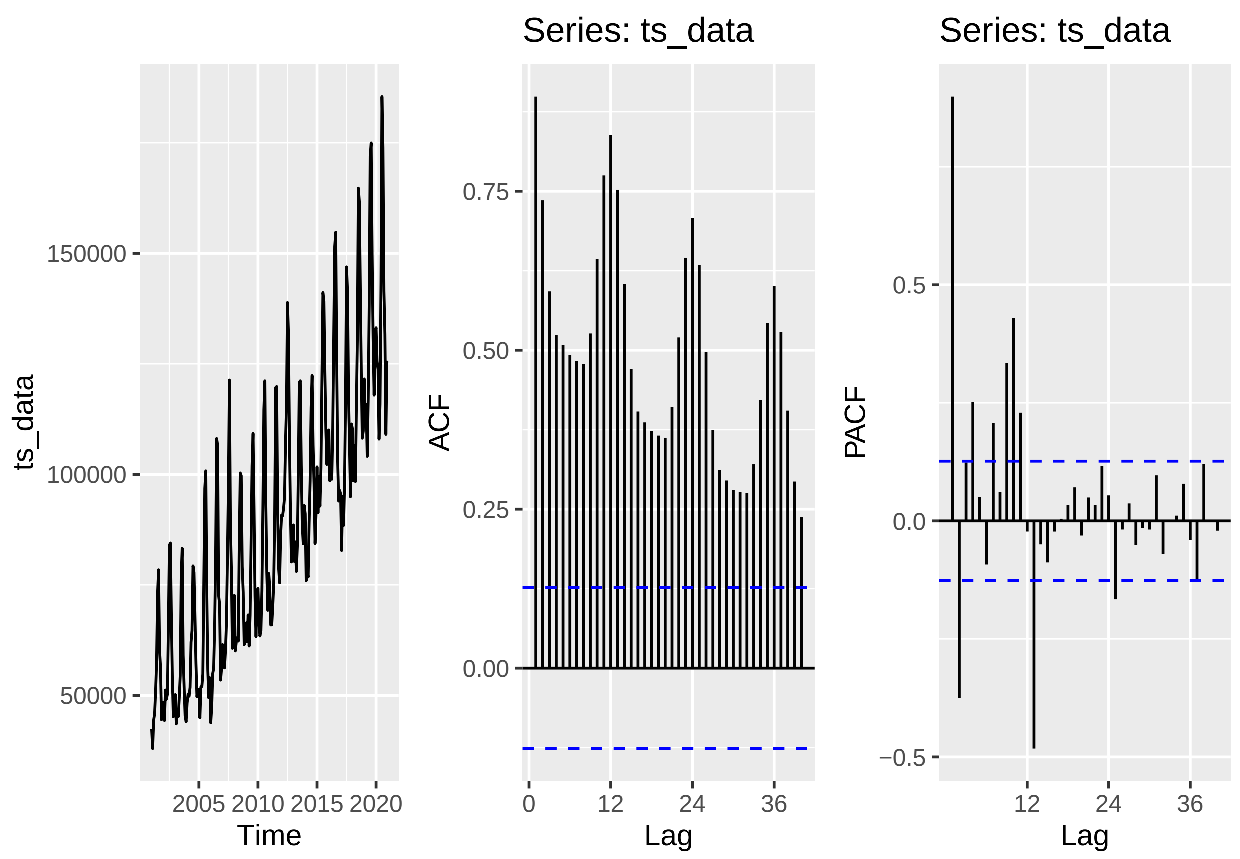
Q1

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.

```
#Importing data from text file
raw_data <- read.csv(file="./Data/Net_generation_United_States_all_sectors_monthly.csv",
                     header=TRUE,skip=4)

#Select columns for natural gas only
data <- raw_data %>%
mutate(Date = my(Month)) %>%
arrange(Date) %>%
mutate(NatGas = natural.gas.thousand.megawatthours) %>%
select(Date,NatGas)

#Inspect data
head(data)
```

```
##         Date   NatGas
## 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
```

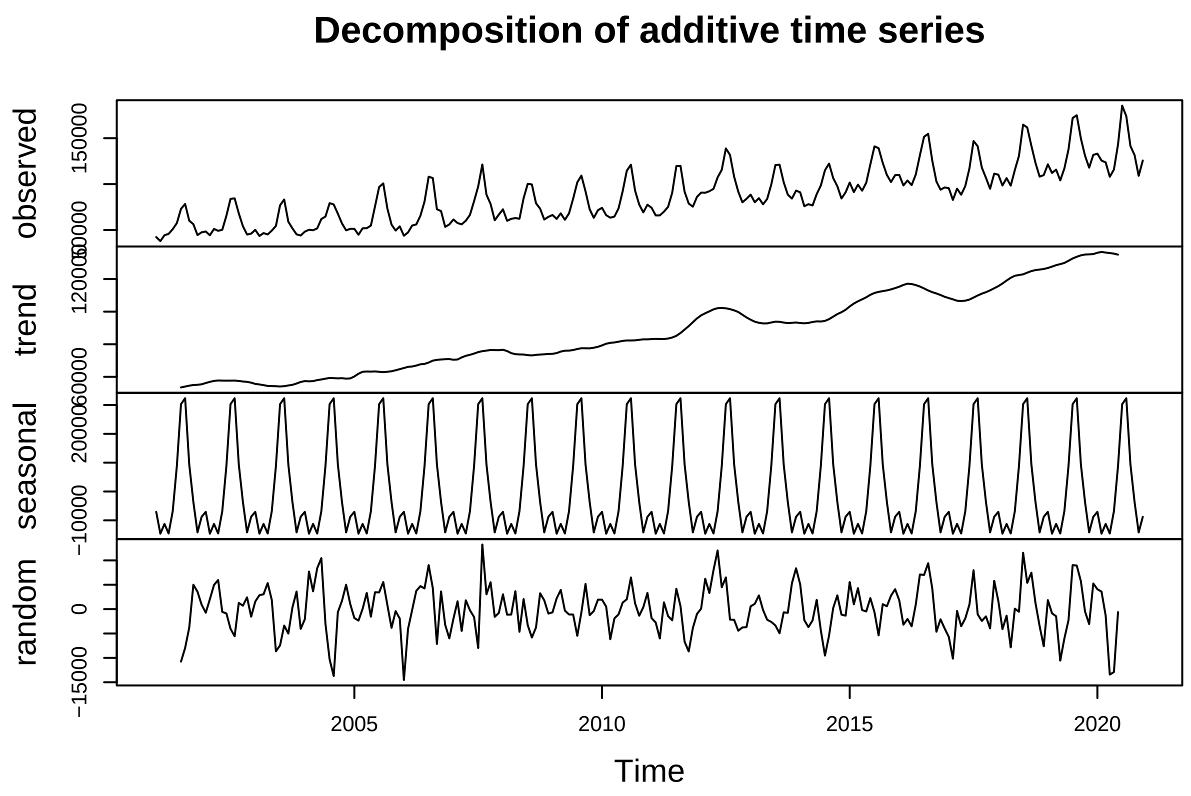
```
nvar <- ncol(data) - 1
nobs <- nrow(data)

#Transform data into a time series object
first_month <- month(first(data$Date))
first_year <- year(first(data$Date))

ts_data <- ts(data$NatGas,
              start=c(first_year,first_month),
              frequency=12)


ACF <- autoplot(Acf(ts_data,plot=FALSE,lag.max=40))
PACF <- autoplot(Pacf(ts_data,plot=FALSE,lag.max=40))

plot_grid(
  autoplot(ts_data),
  ACF,
  PACF,
  nrow=1
)
```



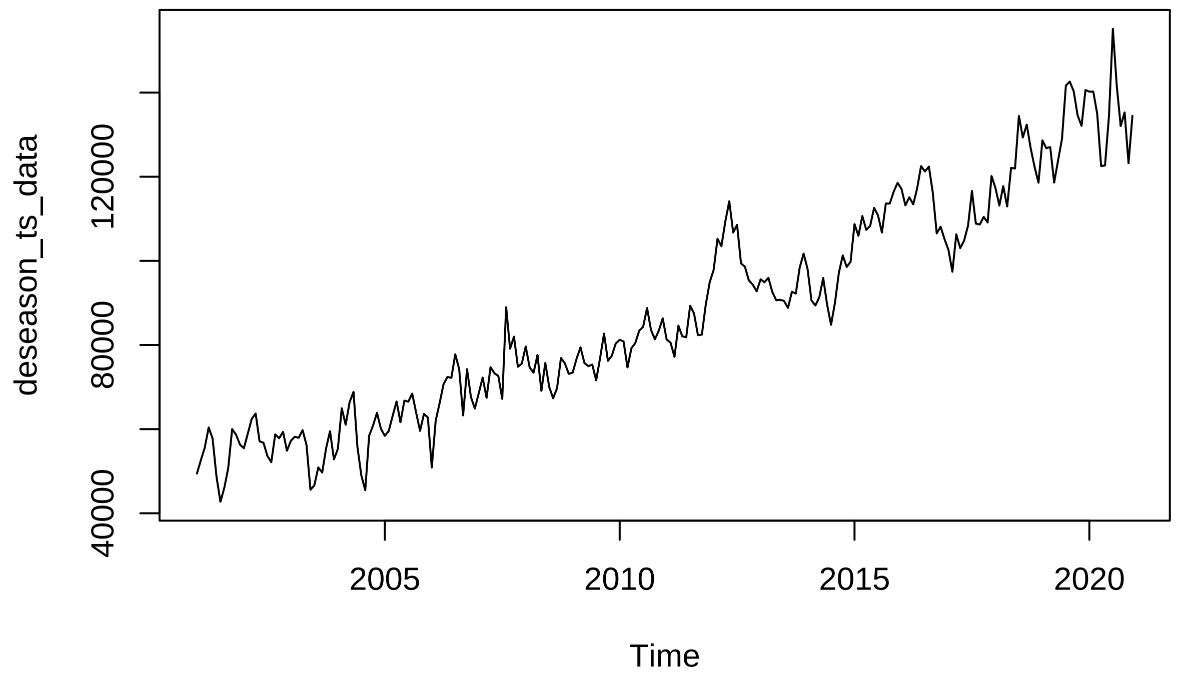
Q2

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.

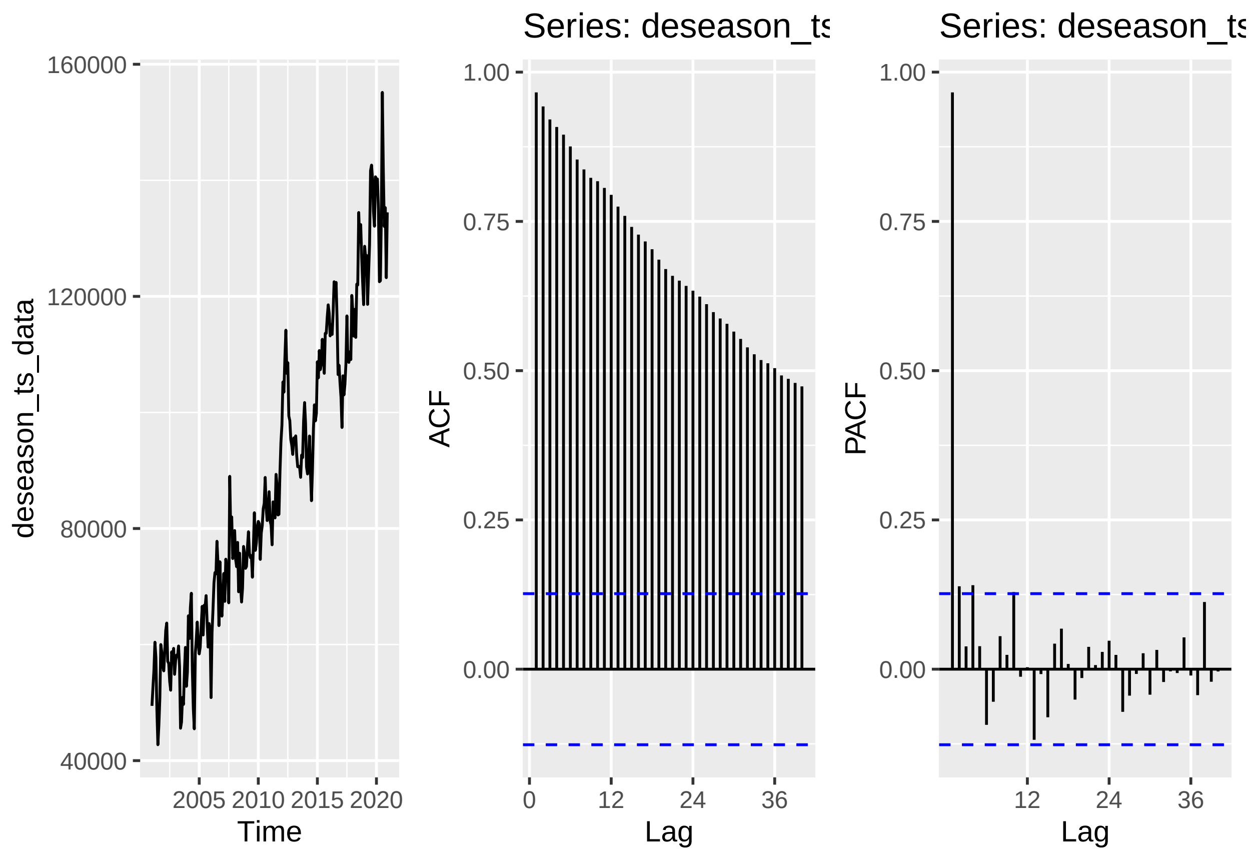
```r
#Decompose
decompose_ts_data <- decompose(ts_data,"additive")
plot(decompose_ts_data)
```



```
#Deseason
deseason_ts_data <- seasadj(decompose_ts_data)
plot(deseason_ts_data)
```



```
ACF_deseason <- autoplot(Acf(deseason_ts_data,plot=FALSE,lag.max=40))
PACF_deseason <- autoplot(Pacf(deseason_ts_data,plot=FALSE,lag.max=40))

plot_grid(
  autoplot(deseason_ts_data),
  ACF_deseason,
  PACF_deseason,
  nrow = 1)
```



```
#Compared to the plots obtained in Q1, the seasonal element is removed --
#the time series plot no longer has a wave-like patern, the ACF no longer has
#a wave-like pattern, and the PACF loses significant spikes at 12 month
#intervals.
```

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.

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

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

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

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

```
##
```

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

```
#Mann-Kendall
print("Results of Mann Kendall on average yearly series")
```

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

```
print(summary(MannKendall(deseason_ts_data)))
```

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

```
#The conclusion for the ADF test, given a p-value of 0.01, is that we reject
#the null hypothesis. This suggests that the time series is stationary / does
#not have a stochastic trend.

#The conclusion for the Mann-Kendall test, with a small p-value of 2.22e-16, is
#that we reject the null hypothesis and conclude that there is a deterministic
#trend. S is positive so its an increasing trend.
```

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.

```
#The cut off of the PACF indicates an AR process. So, p = 1. It seems that the
#series does not need to be differenced, so d = 0.
#The slow decay of the ACF indicates an AR process, so q = 0.

#Therefore we yield ARIMA(1,0,0).
```

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()` r `print()` function to print.

```
model_Q5 <- Arima(deseason_ts_data,
                  order=c(1,0,0),
                  include.mean = TRUE)

summary(model_Q5)
```

```
## Series: deseason_ts_data
## ARIMA(1,0,0) with non-zero mean
##
## Coefficients:
##          ar1      mean
##       0.9825  90230.35
## s.e.  0.0120  16958.50
##
## sigma^2 = 30851494:  log likelihood = -2410.59
## AIC=4827.17   AICc=4827.27   BIC=4837.61
##
## Training set error measures:
##                     ME    RMSE      MAE         MPE     MAPE      MASE
## Training set 281.0307 5531.22 4289.962 -0.09287034 5.240518 0.5244823
##                     ACF1
## Training set -0.1387296
```

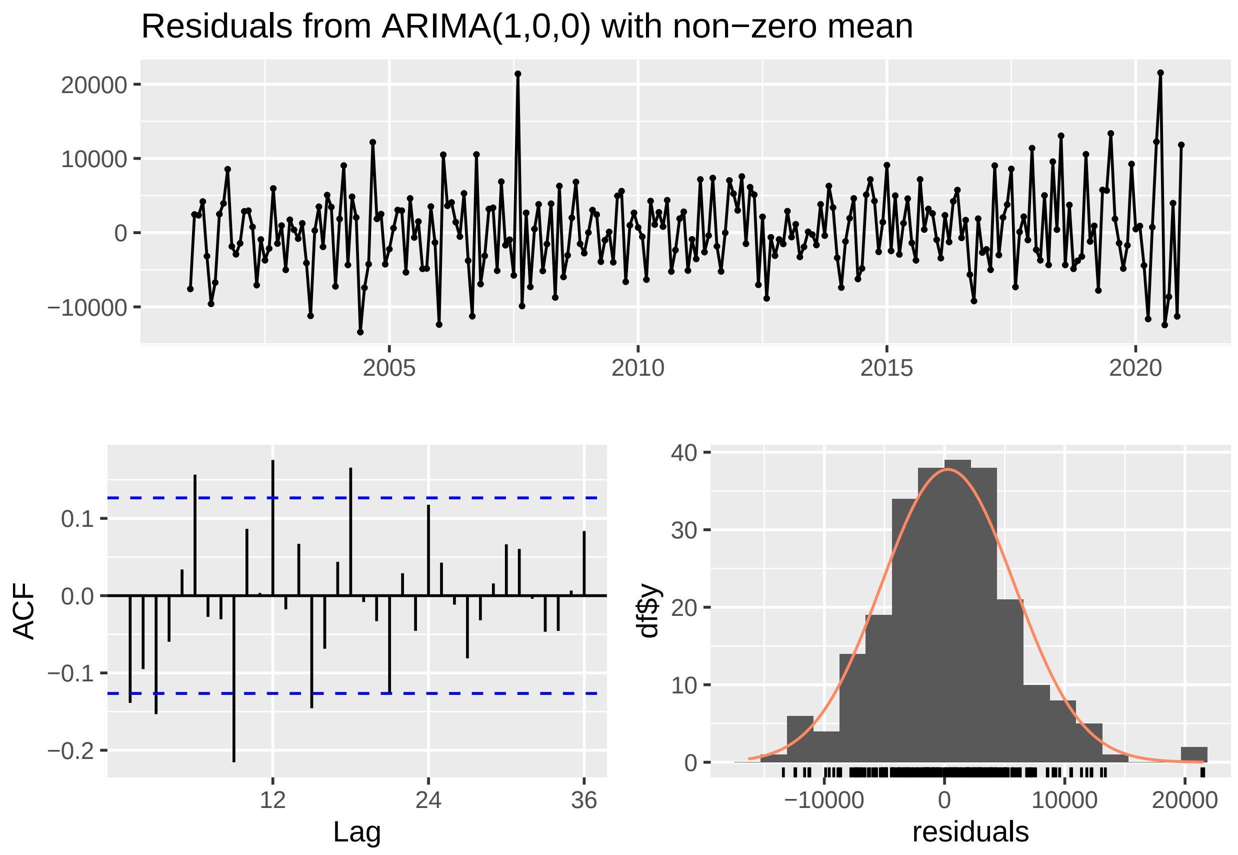
```
print(model_Q5$coef)
```

```
##          ar1    intercept
## 9.825447e-01 9.023035e+04
```

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(model_Q5)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,0) with non-zero mean
## Q* = 66.317, df = 23, p-value = 4.443e-06
##
## Model df: 1.   Total lags used: 24
```

```
#Yes, the residuals do look like a white noise series based on the normal
#distribution and the ACF which shows some values of significance.
```

Modeling the original series (with seasonality)

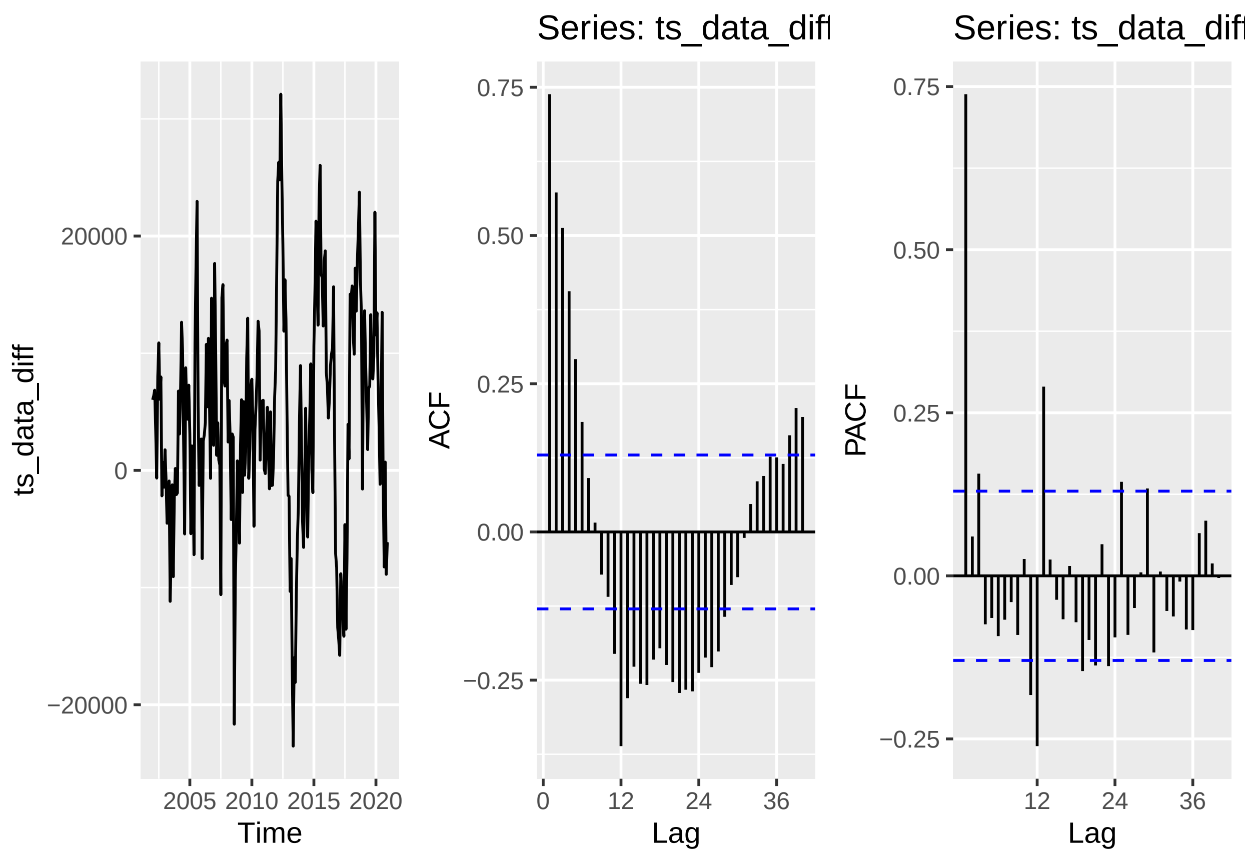
Q7

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.

```
nsdiffs(ts_data)
```

```
## [1] 1
```

```r
#This yields 1, so we will difference one time.

ts_data_diff <- diff(ts_data, lag = 12, differences = 1)

ACF_diff <- Acf(ts_data_diff, plot=FALSE, lag.max = 40)
PACF_diff <- Pacf(ts_data_diff, plot=FALSE, lag.max = 40)

plot_grid(autoplot(ts_data_diff),
          autoplot(ACF_diff),
          autoplot(PACF_diff),
          nrow = 1)
```



```r
#The seasonal wave-like pattern is no longer present. That said, there are
#still significant spikes at 12 which could indicate an MA seasonal term.
#So, P=0, Q=1, D=1.

#For the non-seasonal component, we note exponential decay in the ACF meaning
#that an AR term is needed. The PACF indicates first order because of the
#cutoff between lags 1 and 2. So, p=1, q=0, d=0.

model_Q7 <- Arima(ts_data,
                  order=c(1,0,0),
                  seasonal=c(0,1,1),
                  include.constant=TRUE)
```

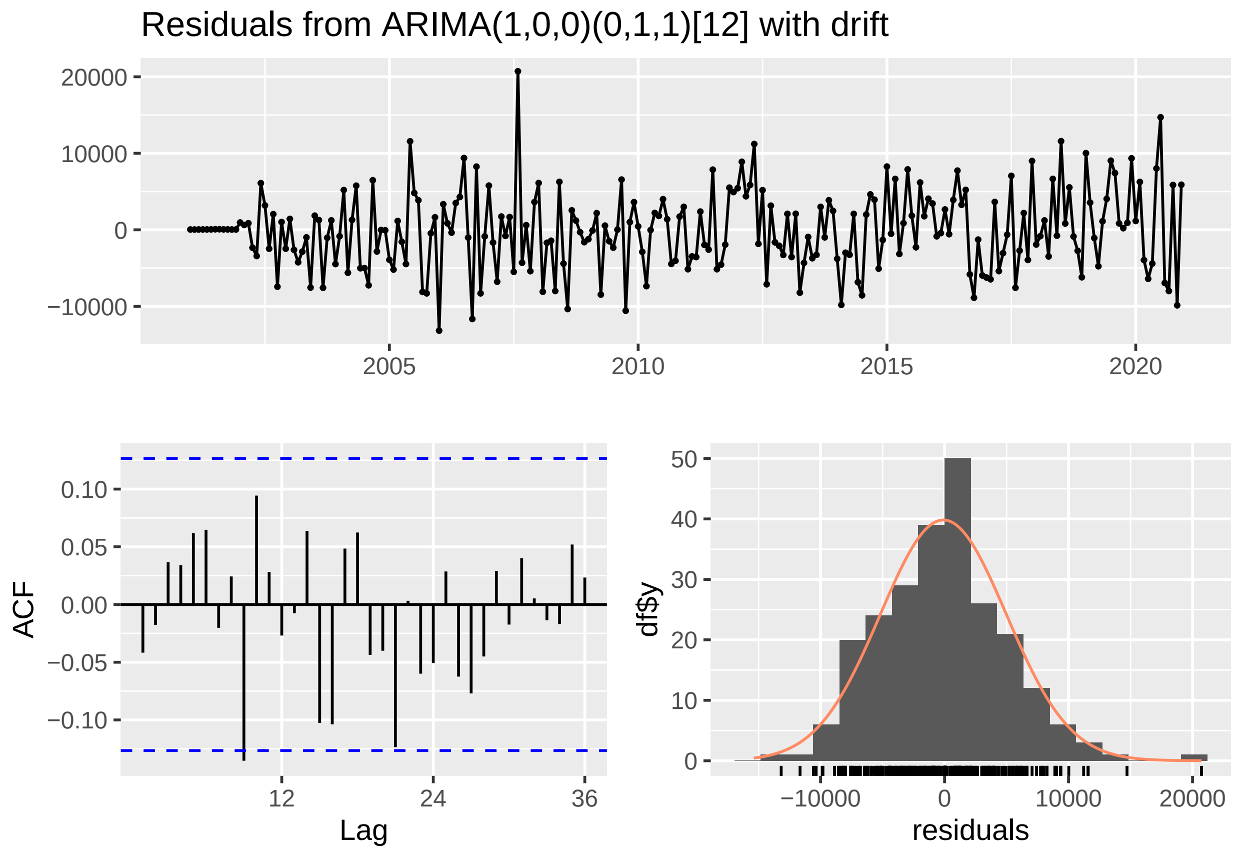
```r
summary(model_Q7)
```

```
## Series: ts_data
## 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
##
## Training set error measures:
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -97.32578 5083.901 3950.295 -0.7114711 4.673706 0.4829553
##                     ACF1
## Training set -0.04171074
```

```r
print(model_Q7$coef)
```

```
##         ar1        sma1       drift
##   0.7415607  -0.7025578 358.7988398
```

```r
checkresiduals(model_Q7)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,0)(0,1,1)[12] with drift
## Q* = 25.414, df = 22, p-value = 0.2777
##
## Model df: 2.   Total lags used: 24
```

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.

```
#Q7 looks like a better model based on the lack of significant spikes in the
#ACF and the results of the statistical test, which indicates that there is no
#correlation with time. It seems that the ARIMA is not modeling the seasonal
#component, while the SARIMA is -- therefore, this is perhaps not a fair
#comparison.
```

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?

```
auto.arima(deseason_ts_data)
```

```
## Series: deseason_ts_data
## 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 is ARIMA(1,1,1). This does not match ARIMA(1,0,0) as 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?

```r
auto.arima(ts_data)
```

```
## Series: ts_data
## 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
```

```r
#The order is SARIMA(1,0,0)(0,1,1). This does match as specified in Q7.
```