ENV 790.30 - Time Series Analysis for Energy Data | Spring 2023

Assignment 7 - Due date 03/20/23

Hugh Cipparone

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_Sp23.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.

Set up

```
#Load/install required package here

library(lubridate)
```

```
## Loading required package: timechange
```

```
##
## 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 1.3.2 --
```

```
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.5.0
```

```
## v readr   2.1.3      v forcats 0.5.2
## v purrr   0.3.5
## -- 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()
```

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**.

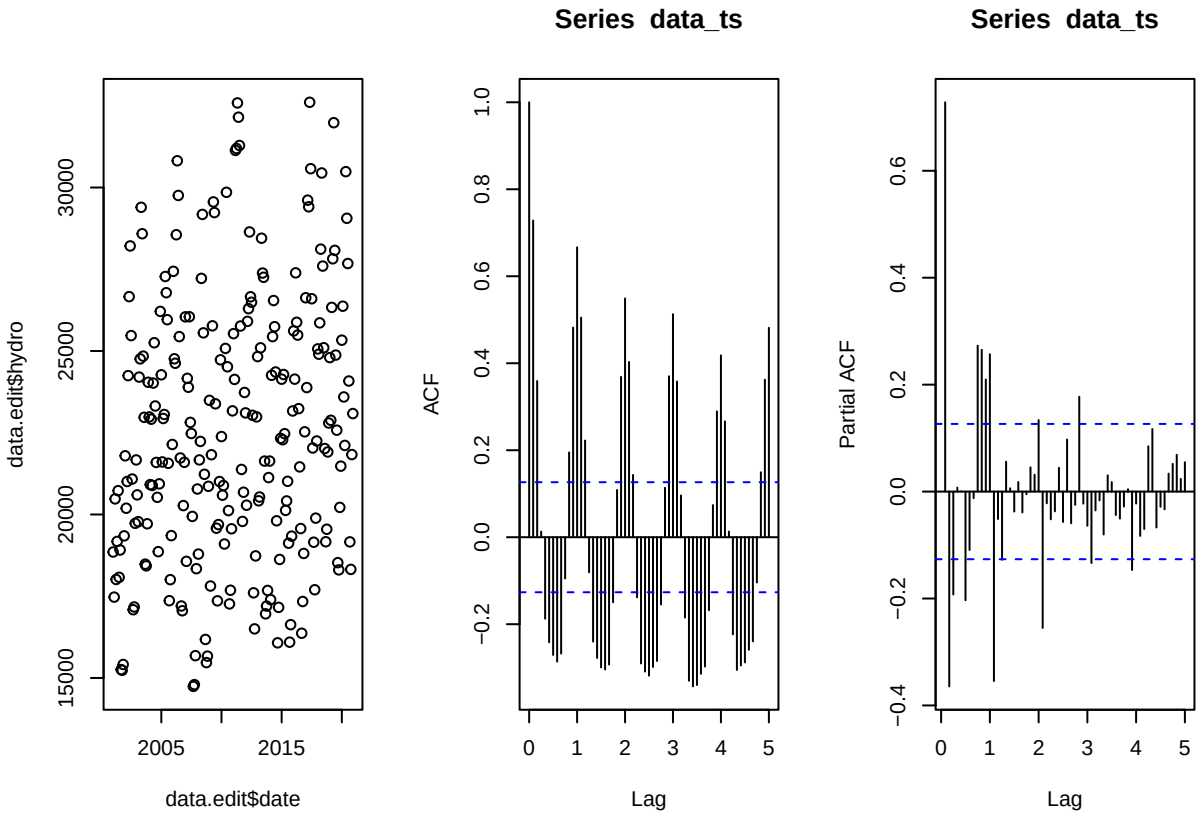
Packages needed for this assignment: "forecast","tseries". Do not forget to load them before running your script, since they are NOT default packages.\

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.

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

date<-my(data$Month)

data.edit<-cbind(date,data) %>%
  select(date,conventional.hydroelectric.thousand.megawatthours) %>%
  rename(hydro=conventional.hydroelectric.thousand.megawatthours)


data_ts <- ts(data.edit$hydro,start=c(2001,1),frequency = 12)
```

```
par(mfrow=c(1,3))

plot(data.edit$date, data.edit$hydro)

acf(data_ts,lag.max=60, plot=TRUE)
pacf(data_ts,lag.max=60, plot=TRUE)
```

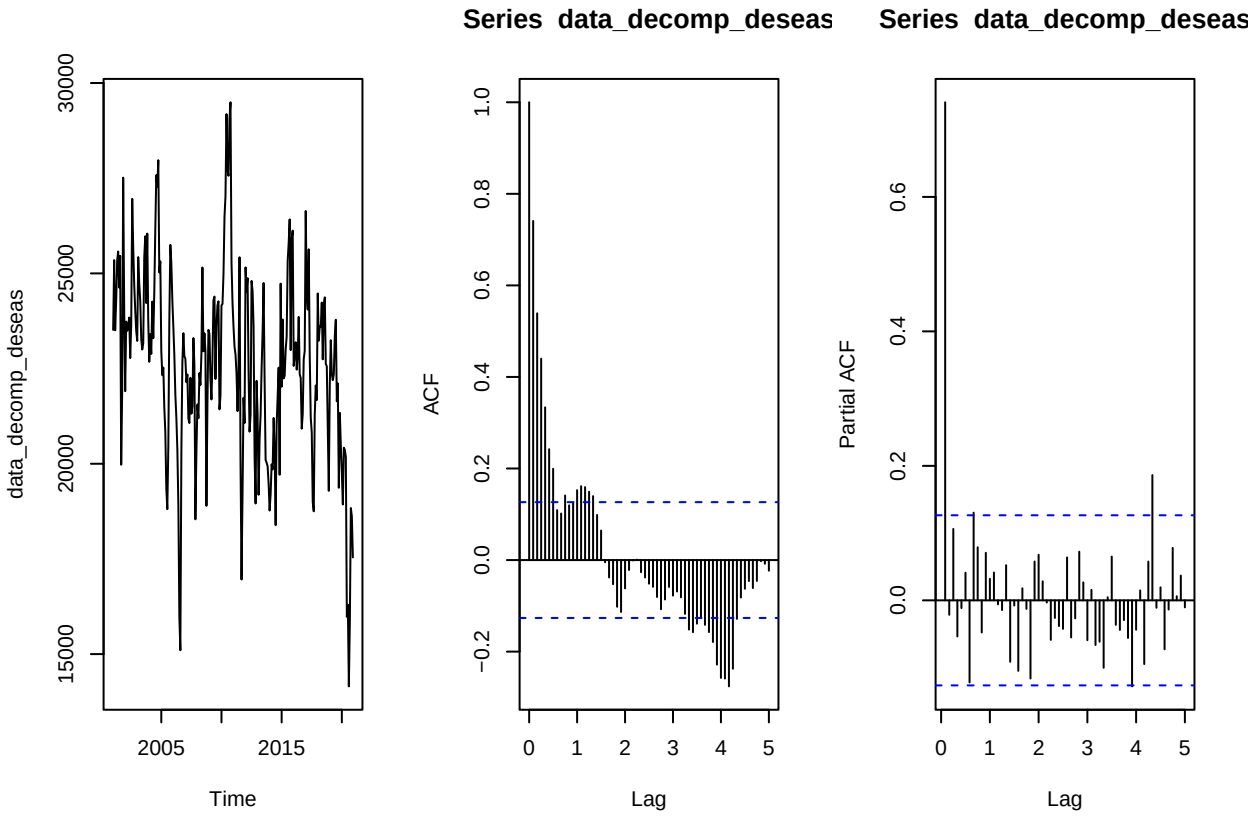


Q2

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.

```
data_decomp <- stl(data_ts, s.window = "periodic")

data_decomp_deseas <- seasadj(data_decomp)
```

```
par(mfrow=c(1,3))
plot(data_decomp_deseas)
acf(data_decomp_deseas,lag.max=60, plot=TRUE)
pacf(data_decomp_deseas,lag.max=60, plot=TRUE)
```



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.

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

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

```
##
##  Augmented Dickey-Fuller Test
##
## data:  data_decomp_deseas
## Dickey-Fuller = -4.459, Lag order = 6, p-value = 0.01
## alternative hypothesis: stationary
```

```
print(summary(MannKendall(data_decomp_deseas)))
```

```
## Score =  -6792 , Var(Score) = 1545533
## denominator =  28680
## tau = -0.237, 2-sided pvalue =4.6937e-08
## NULL
```

Interpretation: The ADF test shows that we reject the null - which is that these data are stochastic - meaning that these data are not stochastic. Looking at the Mann Kendall, we see that the null hypothesis is once again rejected ($p<0.05$) meaning that these data follow a deterministic trend. Overall, these results suggest a deterministic trend in the 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.

Answer: p=1 because p is the order of the AR component and the lag of the PACF for the series looks like it drops off vastly after the first value. d=1 because it is a deterministic trend and therefore requires differencing. q=5 because the MA section of the model comes from the ACF plot and the first spike is 1 so doesn't count (no actual lag comparison there) and there appears to be 5 lags before any real drop off in value of the lags.

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*() function to print.

```
arima.1<-Arima(data_decomp_deseas, order=c(1,1,5), include.drift = TRUE)

print("Hand-made ARIMA Output")
```

```
## [1] "Hand-made ARIMA Output"
```

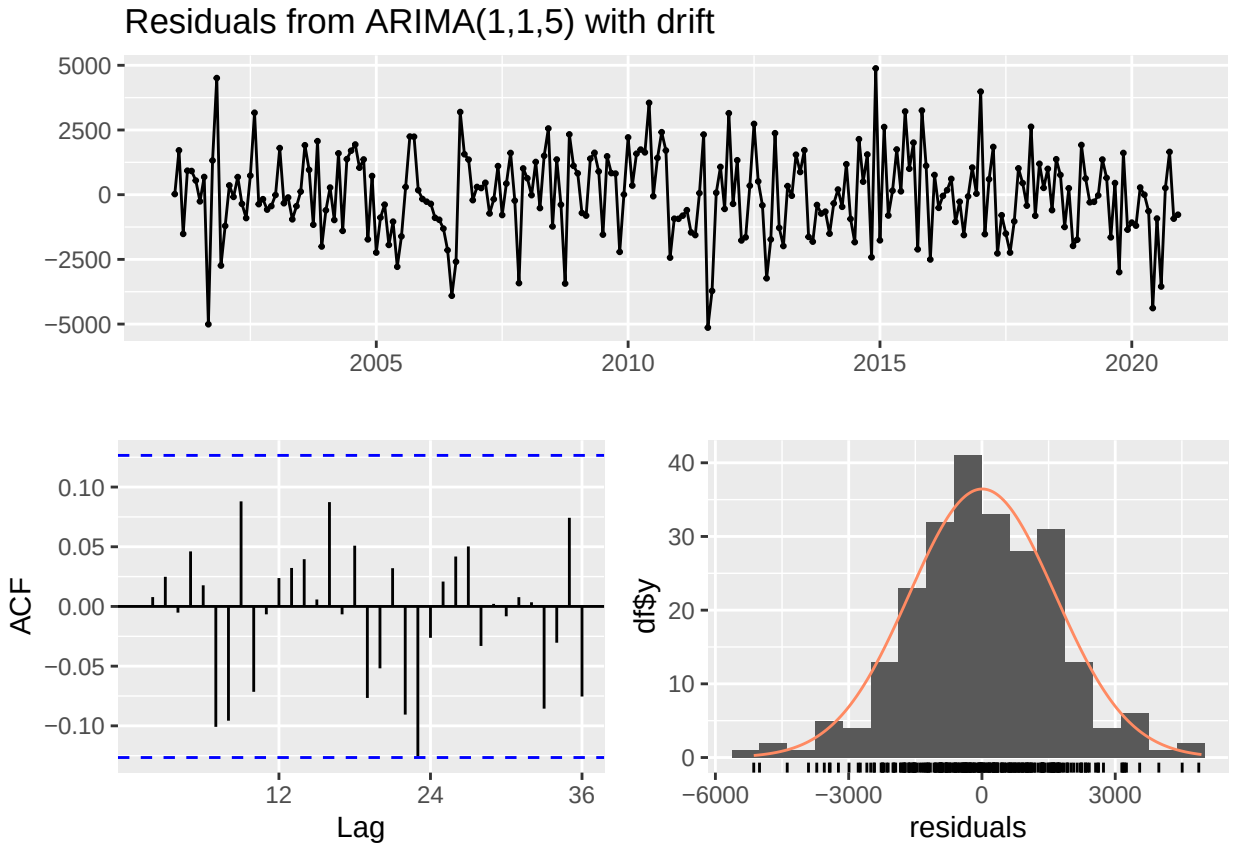
```
print(arima.1$coef)
```

```
##          ar1          ma1          ma2          ma3          ma4          ma5
##  -0.31944256   0.11605780  -0.35880211  -0.19401937  -0.06558832  -0.19035632
##        drift
## -23.43851319
```

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.1)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,5) with drift
## Q* = 21.468, df = 18, p-value = 0.2565
##
## Model df: 6.   Total lags used: 24
```

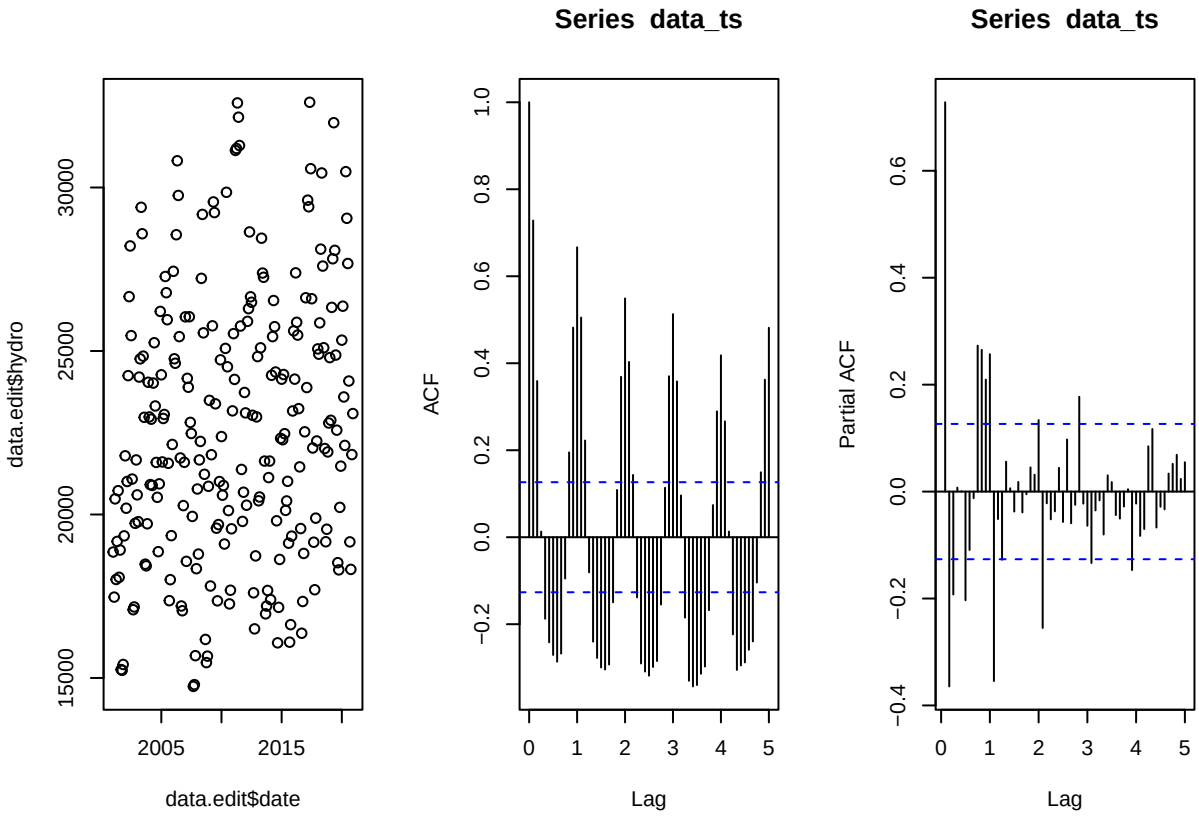
Answer: These residuals do more or less look like white noise. Their median - and seeming mean - is 0 at least. The plot does look like it groups around 0, although there does appear to be some wavering along that value (some amount of periodic shapes). But overall looks pretty good frankly.

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.

```
par(mfrow=c(1,3))

plot(data.edit$date, data.edit$hydro)
acf(data_ts,lag.max=60, plot=TRUE)
pacf(data_ts,lag.max=60, plot=TRUE)
```




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

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

```
##
##  Augmented Dickey-Fuller Test
##
## data:  data_ts
## Dickey-Fuller = -8.0077, Lag order = 6, p-value = 0.01
## alternative hypothesis: stationary
```

```
print(summary(SeasonalMannKendall(data_ts)))
```

```
## Score =  -526 , Var(Score) = 11400
## denominator =  2280
## tau = -0.231, 2-sided pvalue =8.3741e-07
## NULL
```

Determine Order: Based on the results above, I think that D = 1 because there is evidence of a deterministic trend from the Seasonal Mann Kendall. This appears to be an SAR process. Q = 3 because at the point of seasonality (lag = 1) there are 3 values before it drops below significance. P = 0 because there doesn't appear to be a strong spike at the point of seasonality (the positive values around Lag 1)

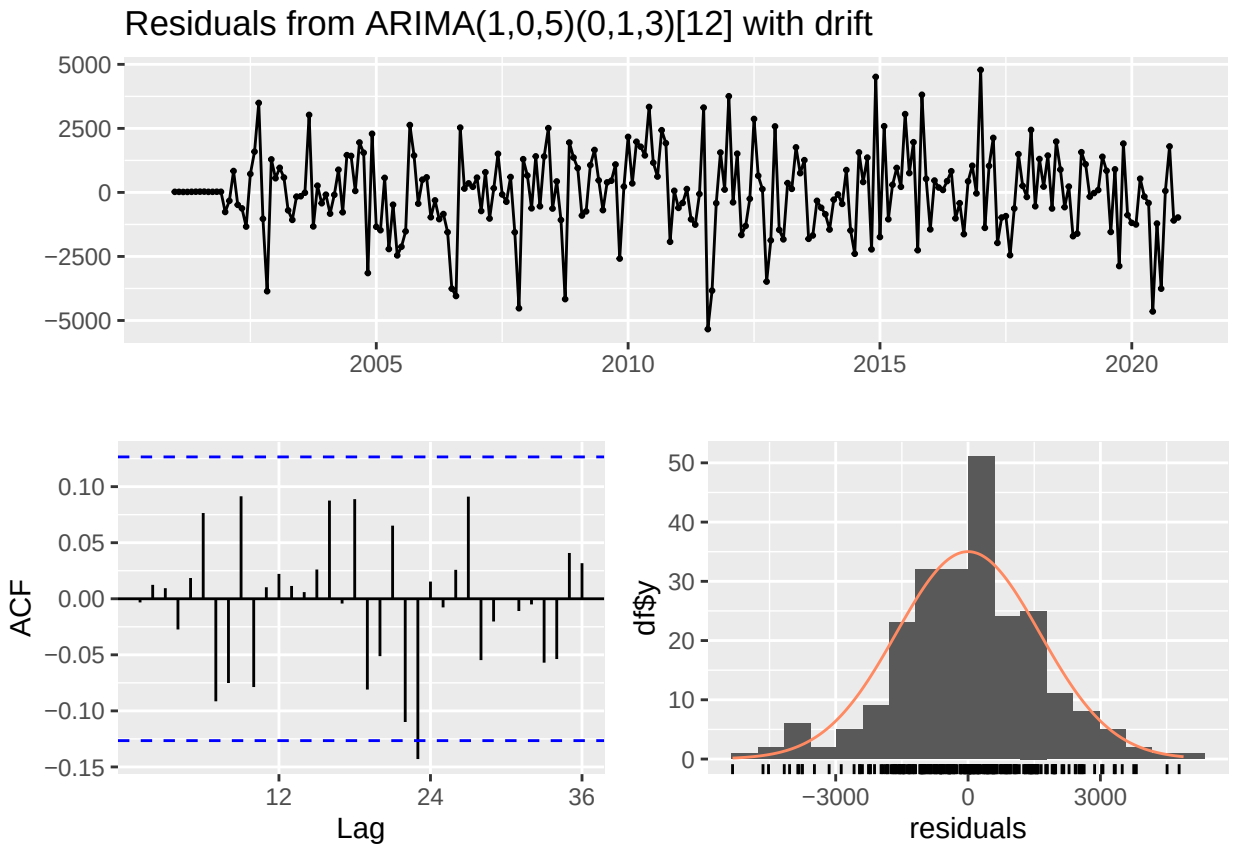
```
arima.seas<-Arima(data_ts, order=c(1,0,5), seasonal = c(0,1,3),include.drift = TRUE)

print("Hand-made Seasonal ARIMA Output")
```

```
## [1] "Hand-made Seasonal ARIMA Output"
```

```
print(arima.seas$coef)
```

```
##          ar1          ma1          ma2          ma3          ma4          ma5
##   0.85400459  -0.08685016  -0.20827952  -0.03253005   0.01614867  -0.11129114
##         sma1         sma2         sma3        drift
##  -0.89375933  -0.04761249  -0.05852568 -14.04164163
```

```
checkresiduals(arima.seas)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,5)(0,1,3)[12] with drift
## Q* = 25.671, df = 15, p-value = 0.04163
##
## Model df: 9.   Total lags used: 24
```

Interpretation: This does appear to be white noise - mean around zero, although some random spikes here and there. My only concern is the large spike just above 0 in the histogram of the value of the residuals. And the fact that the ACF lag value spikes into significance around lag 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.

Answer: I'm not entirely sure, but I think that Q6 - the deseasoned ARIMA model - is a little better than my seasonal ARIMA model. While the median for each appears to be 0, the spike just above 0 in Q7 is suspicious to me.

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(data_decomp_deseas)
```

```
## Series: data_decomp_deseas
## ARIMA(1,1,2)(1,0,0)[12]
##
## Coefficients:
##          ar1      ma1      ma2    sar1
##       0.6303  -0.8463  -0.0945  0.0674
## s.e.  0.3938   0.4280   0.2222  0.0847
##
## sigma^2 = 2796813:  log likelihood = -2111.41
## AIC=4232.82   AICc=4233.07   BIC=4250.2
```

Answer: This matches some aspects = p=1 and d=1 - but my q was all off

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(data_ts)
```

```
## Series: data_ts
## ARIMA(1,0,0)(2,1,0)[12] with drift
##
## Coefficients:
##          ar1     sar1     sar2     drift
##       0.7312  -0.5628  -0.2583  -21.3603
## s.e.  0.0453   0.0677   0.0663   21.9532
##
## sigma^2 = 3706816:  log likelihood = -2048.36
## AIC=4106.73   AICc=4107   BIC=4123.88
```

Answer: It absolutely does not. I only got the D component correct!