ENV 790.30 - Time Series Analysis for Energy Data | Spring 2022

Assignment 7 - Due date 03/25/22

Narissa Jimenez-Petchumrus

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_A07_Sp22.Rmd"). Submit this pdf using Sakai.

Set up

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.\

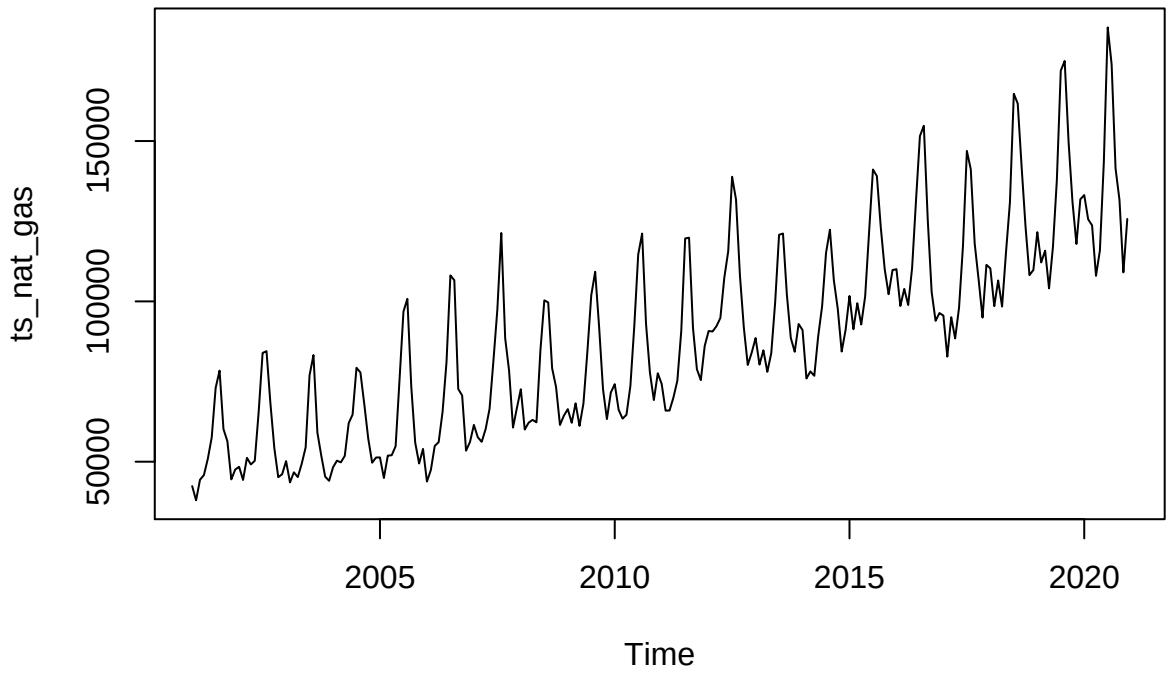
Q1

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.

```
net_gen <- read.csv("/Users/narissapetchumrus/Desktop/Duke Folder/ENV_790/ENV790_TimeSeriesAnalysis_Sp2
net_gen
```

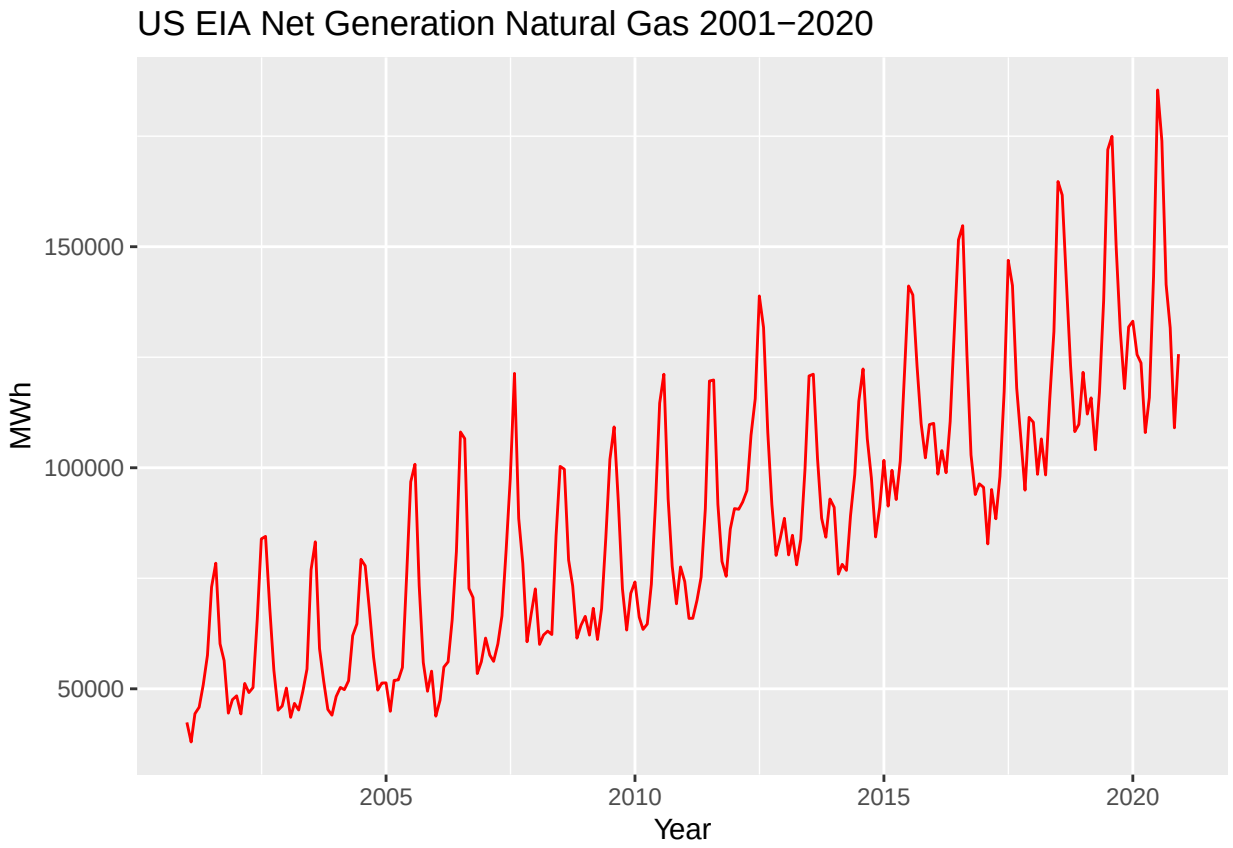
```
#need to flip the data to start at Jan 2001 as it starts at Dec 2020
net_gen_rev<-net_gen[order(nrow(net_gen):1),]
net_gen_rev

#Turn month into date object to make pretty graph because I'm extra
gen.df<-data.frame(net_gen_rev)
my_date <- paste(gen.df[,1], sep="-")
my_date <- my(my_date)
head(my_date)

nat_gas_gen<- data.frame(my_date,gen.df$natural.gas.thousand.megawatthours)
names(nat_gas_gen)<-c('Time','Natural_Gas')
head(nat_gas_gen)
```

```r
#time series object
ts_nat_gas<-ts(net_gen_rev$natural.gas.thousand.megawatthours,start=c(2001,1),end=c(2020,12),frequency=
plot(ts_nat_gas) #fugly plot
```

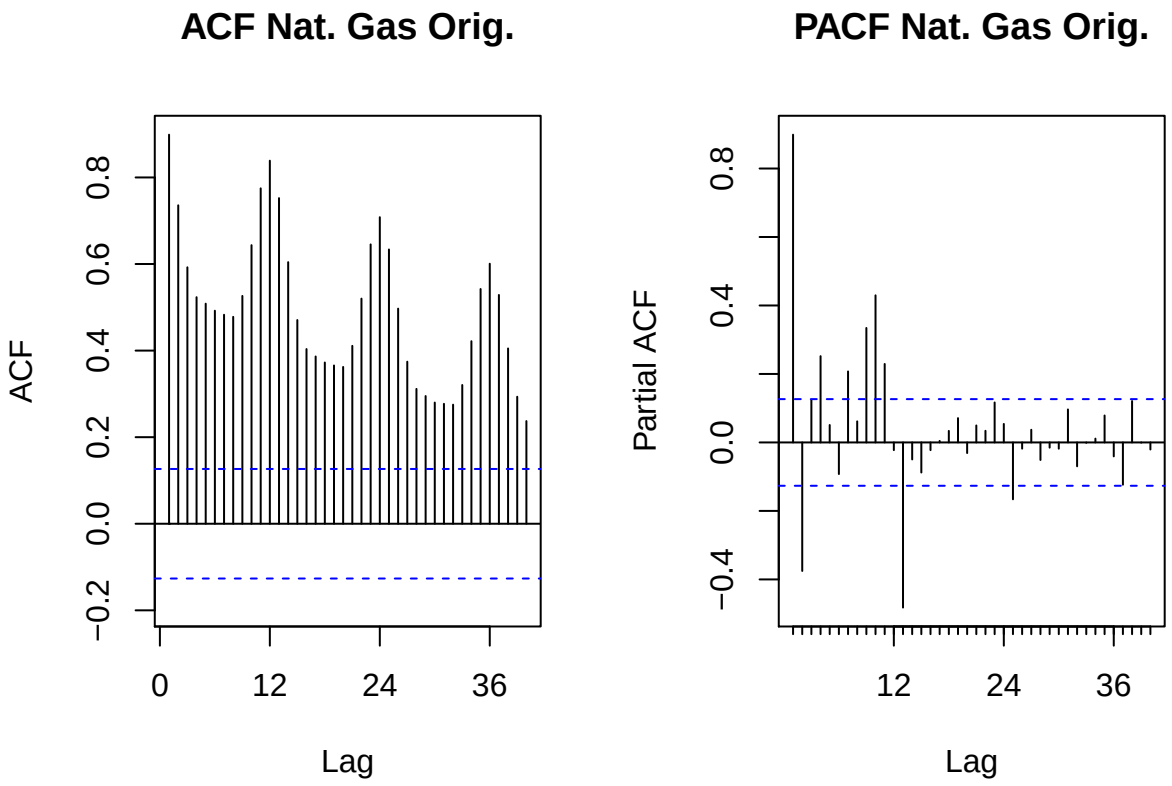



```r
ts_nat_gas
```

```r
#nice plot natural gas, doing red because methane isn't cute
ts_plot <-
  ggplot(nat_gas_gen, aes(x=Time, y=Natural_Gas)) +
      geom_line(color="Red") +
            labs(title="US EIA Net Generation Natural Gas 2001-2020",
                 x="Year",
                 y="MWh",
                 )
plot(ts_plot)
```



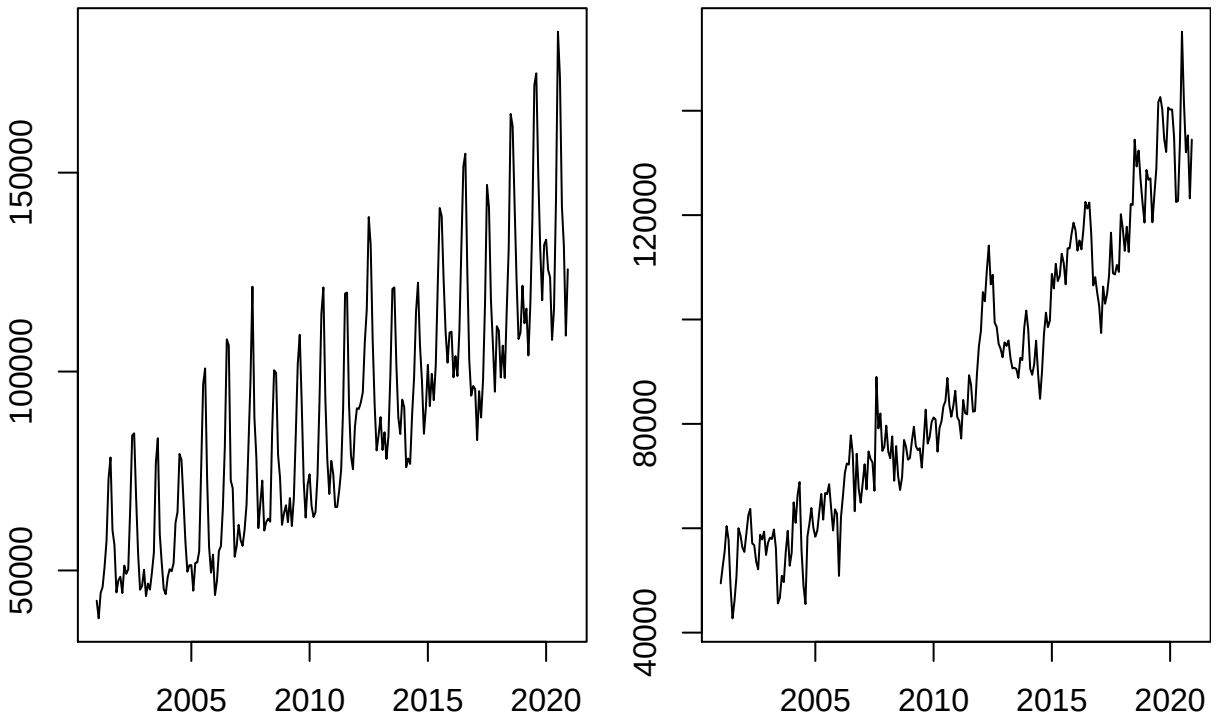

```
#ACF/PACF
par(mfrow=c(1,2))
Acf(ts_nat_gas,lag.max = 40, main="ACF Nat. Gas Orig.")
Pacf(ts_nat_gas,lag.max = 40, main="PACF Nat. Gas Orig.")
```



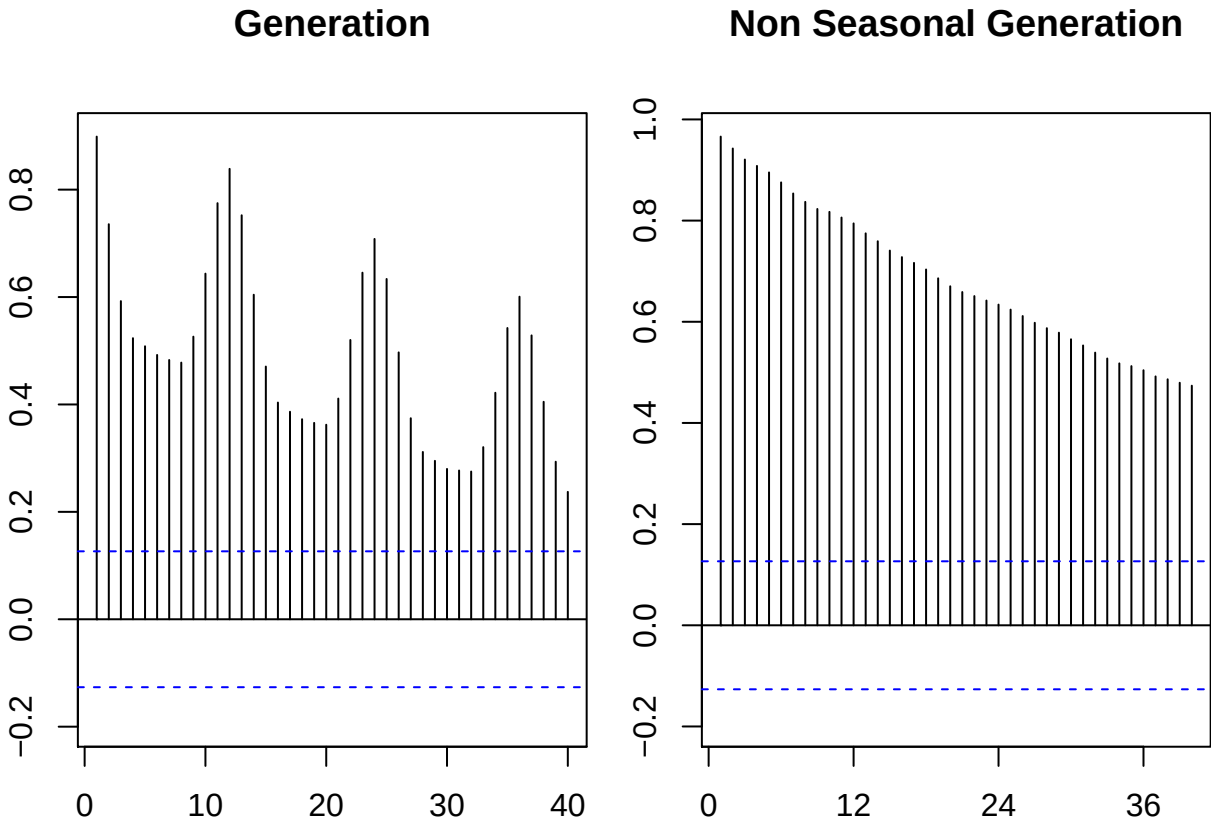
Q2

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.

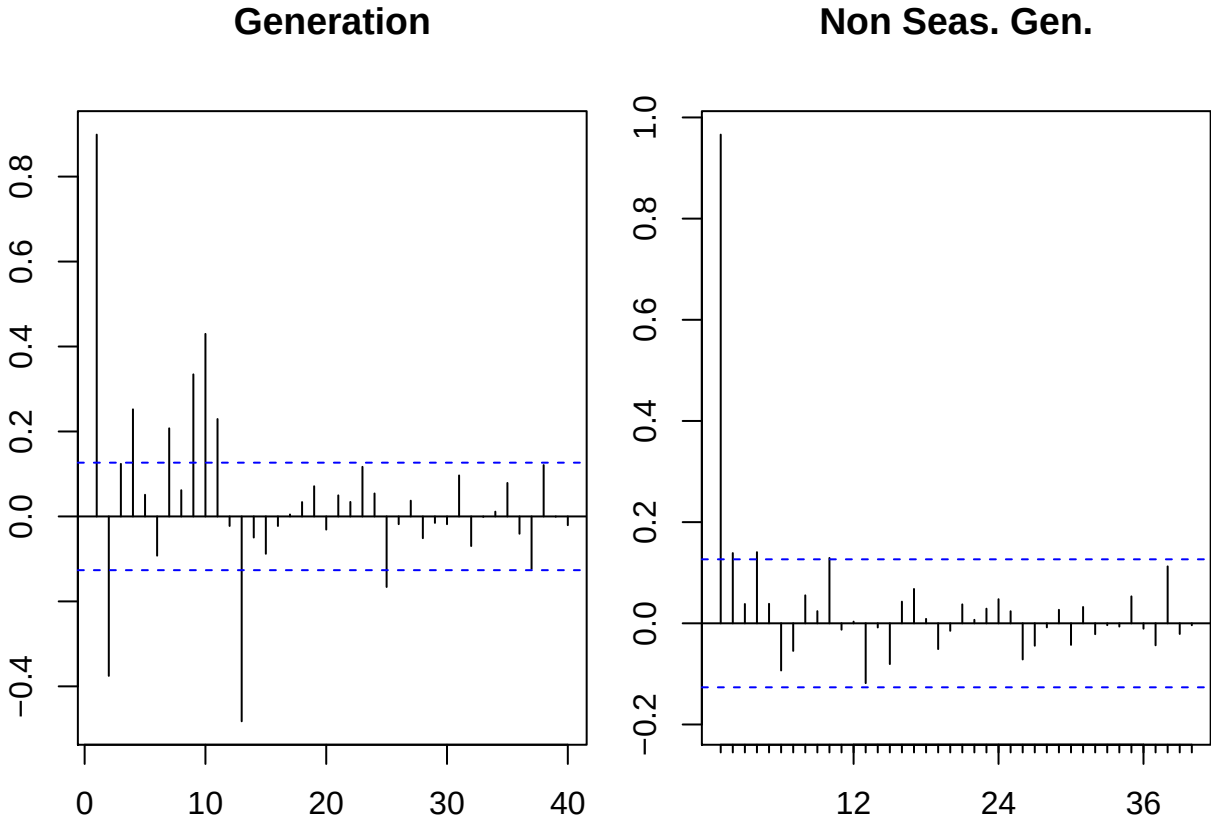
```
#decomposed series
decomp.nat_gas<-decompose(ts_nat_gas,"additive")
deseason.nat_gas<-seasadj(decomp.nat_gas)

#plot deaseasonalized and seasonal data side-by-side
par(mar=c(3,3,3,0));par(mfrow=c(1,2))
plot(ts_nat_gas)
plot(deseason.nat_gas)
```




```
#plot ACFs
par(mar=c(3,3,3,0));par(mfrow=c(1,2))
Acf(nat_gas_gen$Natural_Gas,lag.max=40,main="Generation")
Acf(deseason.nat_gas,lag.max=40,main="Non Seasonal Generation") #seasonality is gone!
```



```
#plot PACFs
par(mar=c(3,3,3,0));par(mfrow=c(1,2))
Pacf(nat_gas_gen$Natural_Gas,lag.max=40,main="Generation")
Pacf(deseason.nat_gas,lag.max=40,main="Non Seas. Gen.")
```



Answer: Compared with the plots in Q1, the time series plots between the seasonal/deseasonal data are different in the sense that the non-seasonal plot doesn't have as much spread/variance compared to the original seasonal plot (although both still have their upward trends in tact). The seasonal ACF clearly has seasonality as we can see increasing/decreasing lags while the non-seasonal ACF has a gradual decline in lags (but still a present trend as the lags gradually decrease). The PACF of the non-seasonal data has lags that are more statistically significant/within the blue bands/confidence intervals compared to the seasonal data.

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(deseason.nat_gas, alternative = "stationary")
```

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

```
##
##  Augmented Dickey-Fuller Test
##
## data:  deseason.nat_gas
## Dickey-Fuller = -4.0271, Lag order = 6, p-value = 0.01
```

```
## alternative hypothesis: stationary
```

```
MannKendall(deseason.nat_gas)
```

```
## tau = 0.843, 2-sided pvalue =< 2.22e-16
```

Answer: According to the Mann-Kendall test, the p-value being way below my self-declared critical value of 0.05 (it's less than or equal to 2.22e-16) means that we can reject the null hypothesis and embrace the alternative hypothesis. This means that there is a deterministic trend within the data. According to the ADF Test, I can reject the null hypothesis and embrace the alternative hypothesis since my p-value is below the critical value of 0.05 (it's 0.01). This implies that there's no unit root (stationary over a unit root but non-stationary over a deterministic trend though), thus the series does not have a stochastic trend. Therefore, to de-trend the data will require a linear regression as I will do below.

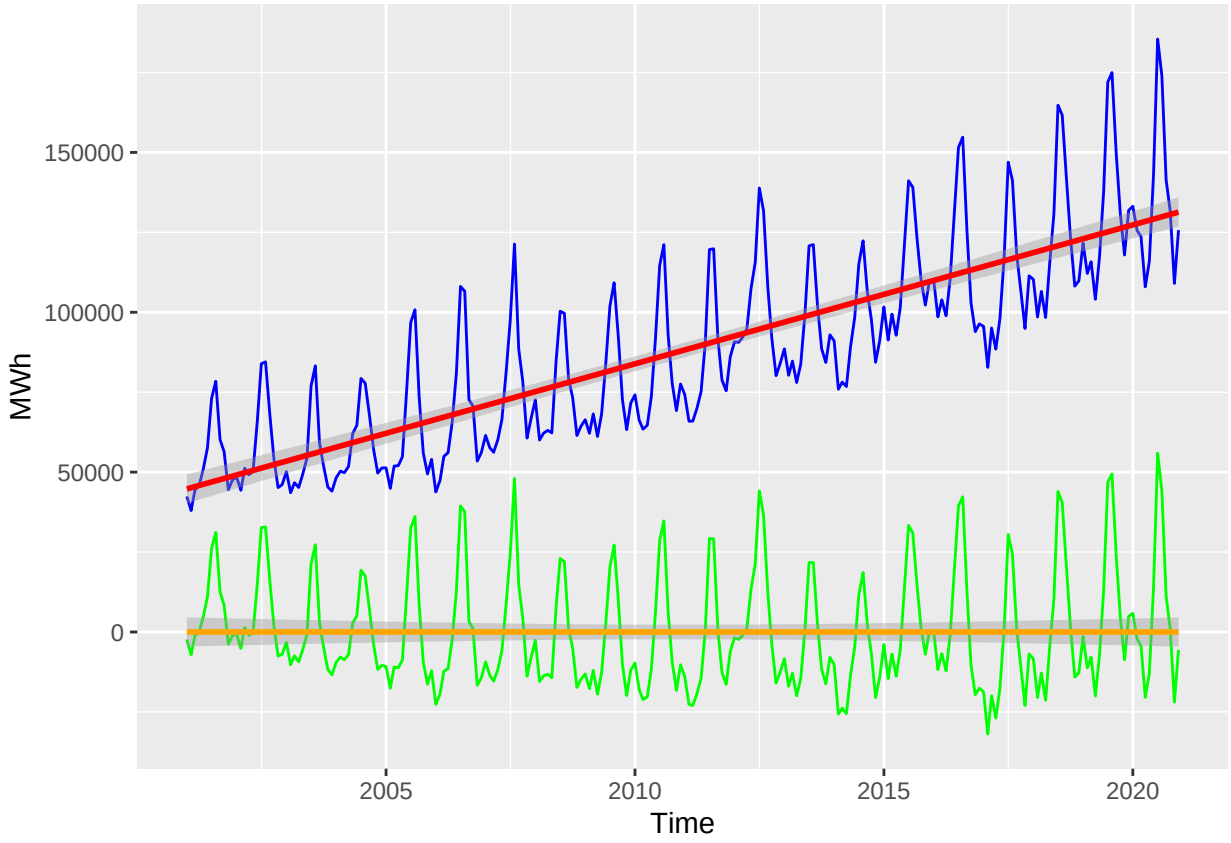
```
#Create vector t
nobs <- nrow(nat_gas_gen)
t <- c(1:nobs)

#Fit a linear trend to TS
lm_model=lm(Natural_Gas~t,data=nat_gas_gen)
summary(lm_model)
```

```
##
## Call:
## lm(formula = Natural_Gas ~ t, data = nat_gas_gen)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -31884 -12824  -5007   9880  55942
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 44379.52    2315.09   19.17   <2e-16 ***
## t             362.23      16.66   21.75   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17880 on 238 degrees of freedom
## Multiple R-squared:  0.6652, Adjusted R-squared:  0.6638
## F-statistic:   473 on 1 and 238 DF,  p-value: < 2.2e-16
```

```
beta0=as.numeric(lm_model$coefficients[1])  #first coefficient is the intercept term or beta0
beta1=as.numeric(lm_model$coefficients[2])  #second coefficient is the slope or beta1

#remove the trend from series
detrend_nat_gas <- nat_gas_gen[,2]-(beta0+beta1*t)

#Understanding what we did
ggplot(nat_gas_gen, aes(x=Time, y=Natural_Gas)) +
            geom_line(color="blue") +
            ylab(paste0("MWh")) +
            geom_smooth(color="red",method="lm") +
            geom_line(aes(y=detrend_nat_gas), col="green") +
            geom_smooth(aes(y=detrend_nat_gas),color="orange",method="lm")
```

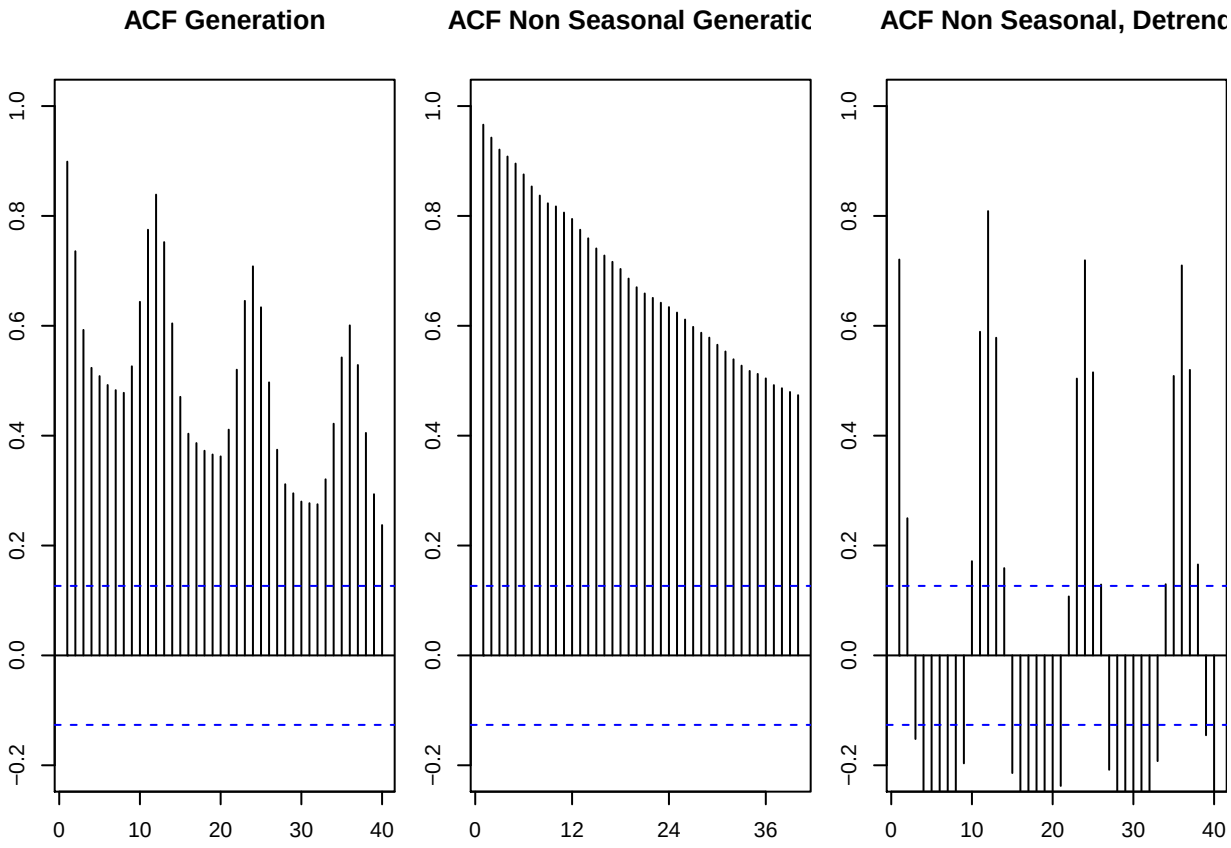
```
## `geom_smooth()` using formula 'y ~ x'
```

```
## `geom_smooth()` using formula 'y ~ x'
```

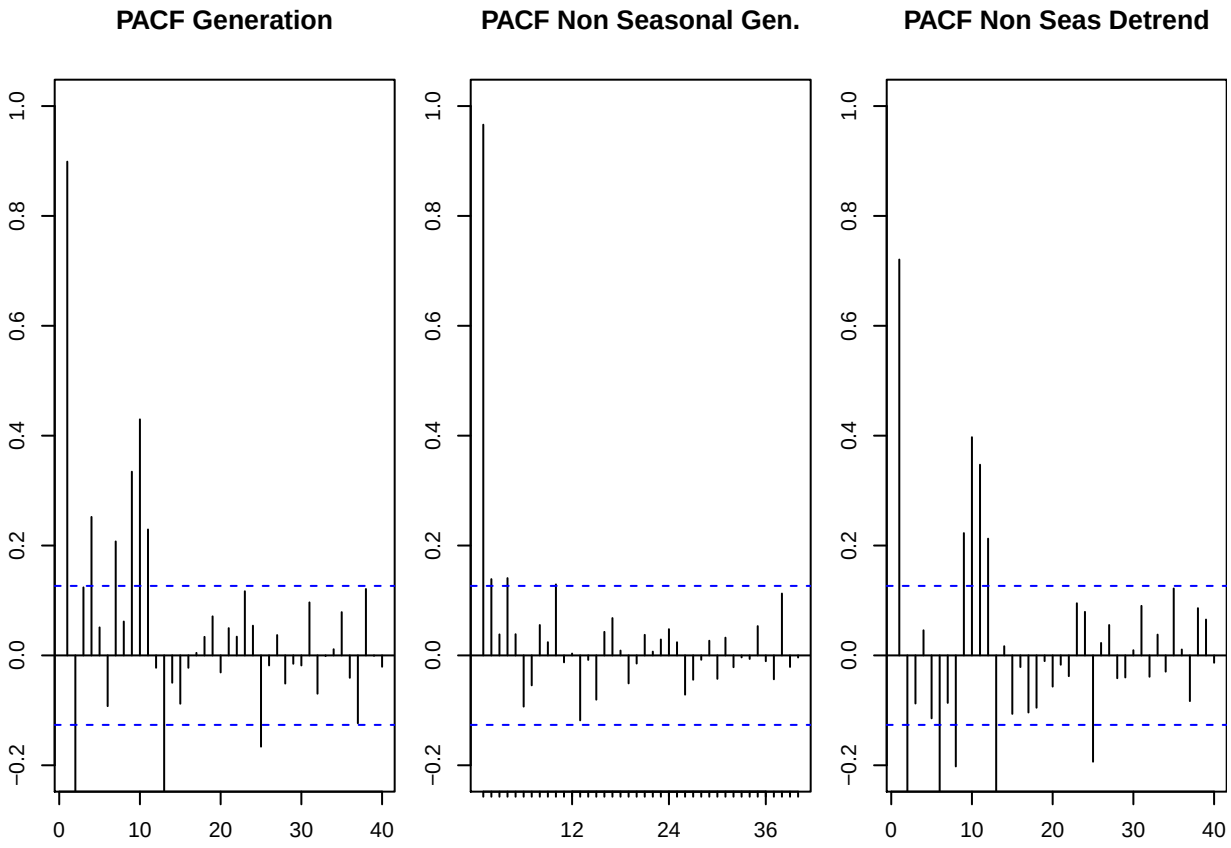


Answer: Great, now the natural gas generation data is both de-seasonalized and detrended! (I know I could've used differencing, but I wanted to review linear regression differencing.)

```
par(mar=c(3,3,3,0));par(mfrow=c(1,3))
Acf(nat_gas_gen$Natural_Gas,lag.max=40,main="ACF Generation",ylim=c(-.2,1))
Acf(deseason.nat_gas,lag.max=40,main="ACF Non Seasonal Generation",ylim=c(-.2,1)) #seasonality is gone!
Acf(detrend_nat_gas,lag.max=40,main="ACF Non Seasonal, Detrend",ylim=c(-.2,1)) #trend is gone!
```




```
par(mar=c(3,3,3,0));par(mfrow=c(1,3))
Pacf(nat_gas_gen$Natural_Gas,lag.max=40,main="PACF Generation",ylim=c(-.2,1))
Pacf(deseason.nat_gas,lag.max=40,main="PACF Non Seasonal Gen.",ylim=c(-.2,1)) #seasonality is gone!
Pacf(detrend_nat_gas,lag.max=40,main="PACF Non Seas Detrend",ylim=c(-.2,1))
```



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: Regarding model parameters and also actually using the ACF/PACF plots I did for Q3 since I detrended/de-seasonalized based off a deterministic trend being present: p=0, d=1 (since we removed the trend by differencing once with regression), q=3. To determine q, I looked at the PACF and the lags seem to kind of decrease gradually which could imply an MA model. For an MA model, we can determine the order by looking at the ACF and the cutoff after some initial lags. On the ACF, the lags cutoff at 3, thus q=3. I determined that p=0 as a safe bet since I'm guessing that this is an MA model vs. an AR model and it's hard to just look at the ACF/PACF and guess an ARMA(1,1) model (p=1, q=1). So I'm determining no AR terms.

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.

```
ARIMAmodel1 <- Arima(detrend_nat_gas,order=c(0,1,3),include.mean=TRUE, include.drift=TRUE)
print(ARIMAmodel1)
```

```
## Series: detrend_nat_gas
## ARIMA(0,1,3) with drift
##
```

```
## Coefficients:
##           ma1      ma2      ma3    drift
##       -0.0294  -0.3673  -0.6033   0.5676
## s.e.   0.0542   0.0521   0.0485  25.5107
##
## sigma^2 estimated as 117464024:  log likelihood=-2560.09
## AIC=5130.19   AICc=5130.45   BIC=5147.57
```

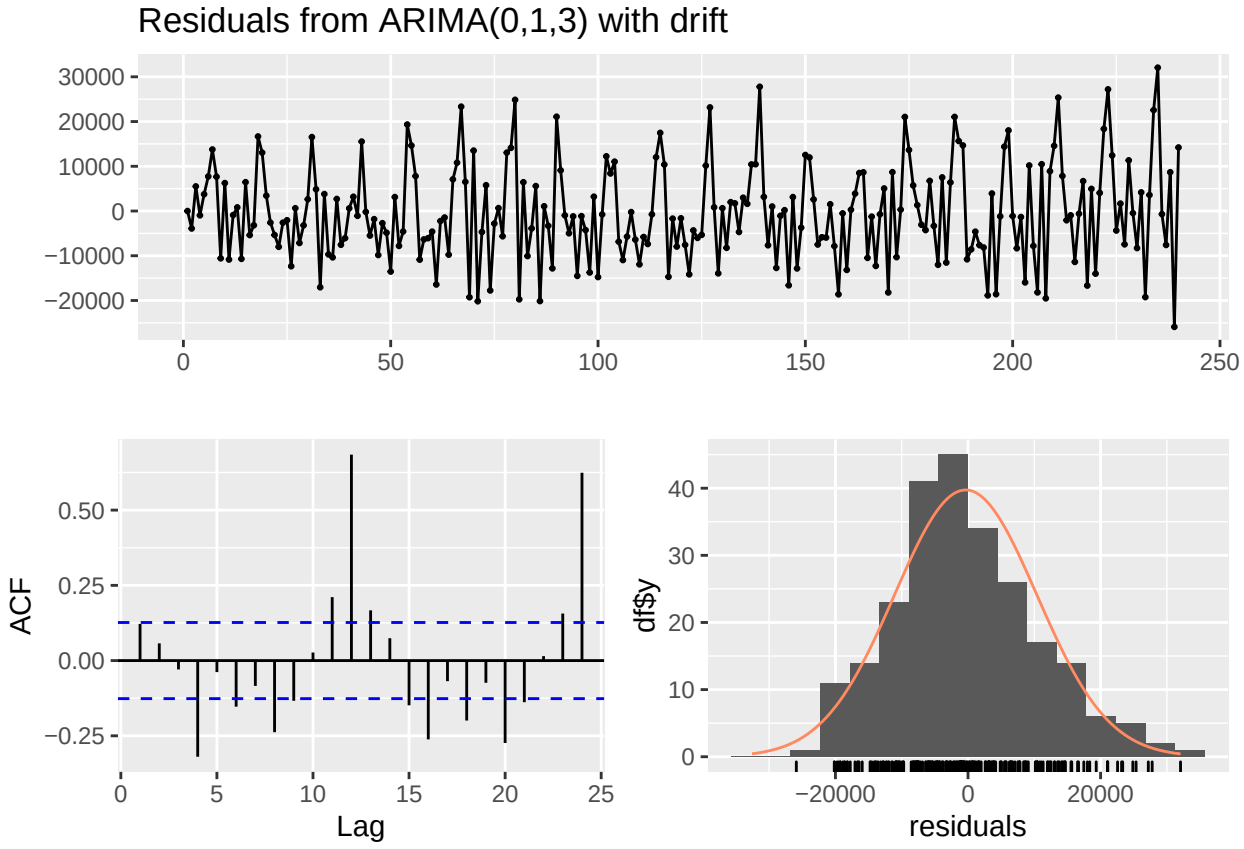
```
ARIMAmodel1coeff<- c(-0.0294,-0.3673,-0.6033)
cat("ARIMA model 1 constants ma1, ma2, ma3=",ARIMAmodel1coeff)
```

```
## ARIMA model 1 constants ma1, ma2, ma3= -0.0294 -0.3673 -0.6033
```

Q6

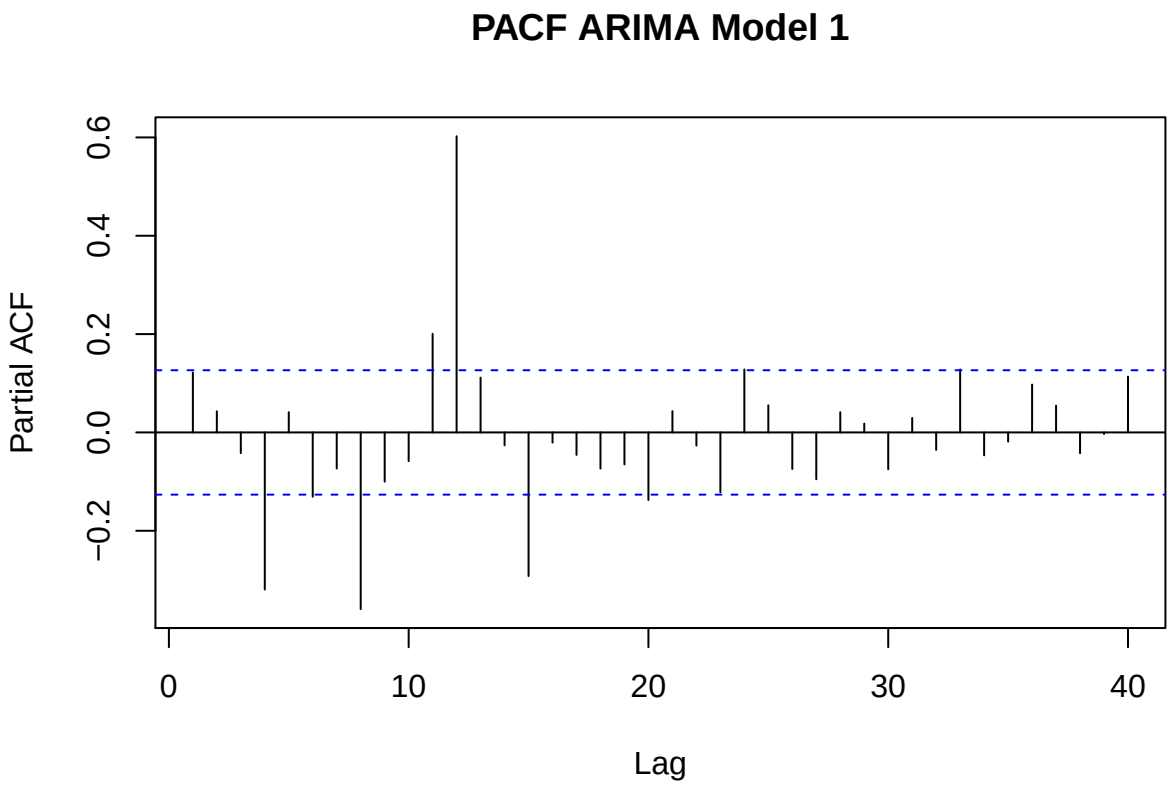
Now plot the residuals of the ARIMA fit from Q5 along with residuals ACF and PACF on the same window. You may use the *checkresiduals*() function to automatically generate the three plots. Do the residual series look like a white noise series? Why?

```
checkresiduals(ARIMAmodel1,lag=40,plot=TRUE)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,3) with drift
## Q* = 564.56, df = 36, p-value < 2.2e-16
##
## Model df: 4.   Total lags used: 40
```

```
Pacf(ARIMAmodel1$residuals,lag.max=40,main="PACF ARIMA Model 1")
```



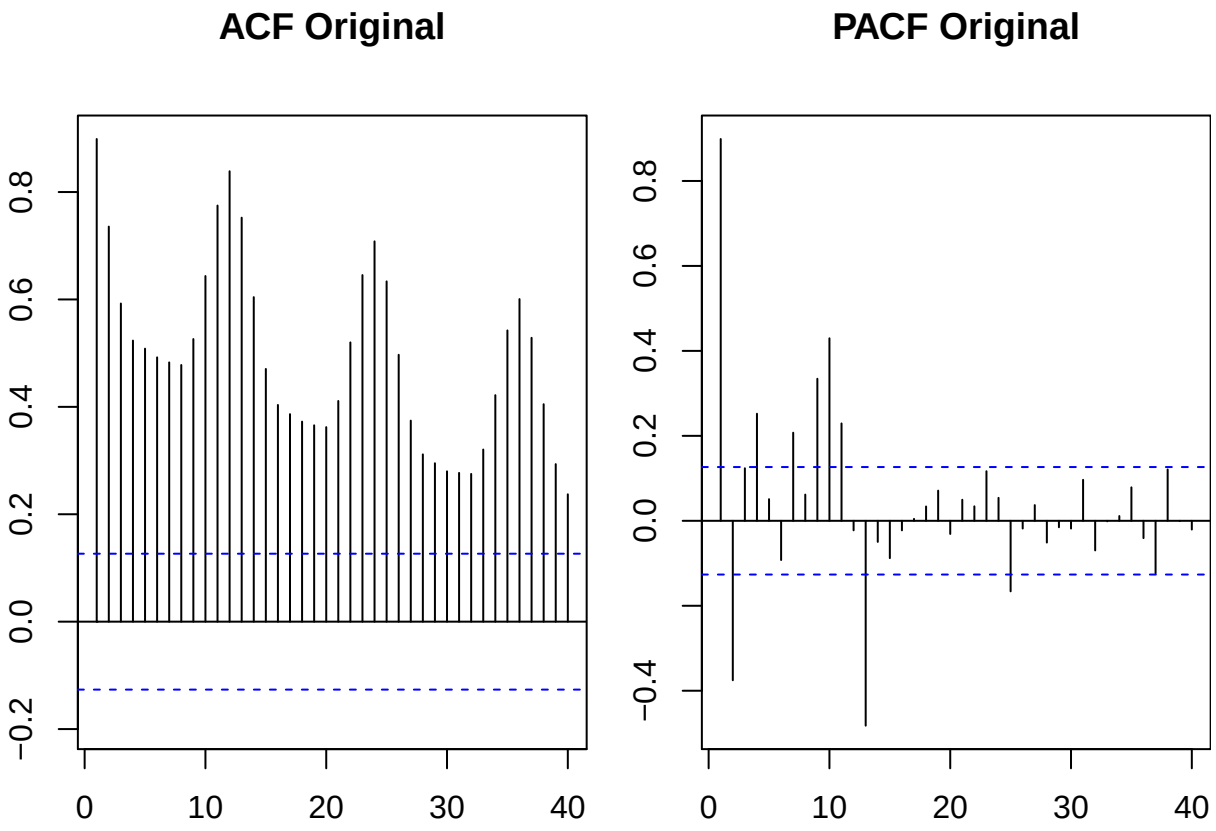
Answer: The residuals appear to be a white noise series because the mean of the residuals is near zero in the first plot. Also the lags of the ACF with the exception of a few are also close to zero/within the blue line confidence intervals in the second plot. Also the histogram plot appears to be nearly normally distributed with the residual mean being mostly concentrated at zero in the third plot. The PACF lags are also mostly close to zero/within the blue line confidence intervals.

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.

```
#plot ACF/PACF
par(mar=c(3,3,3,0));par(mfrow=c(1,2))
Acf(nat_gas_gen$Natural_Gas,lag.max=40,main="ACF Original")
Pacf(nat_gas_gen$Natural_Gas,lag.max=40,main="PACF Original")
```



Answer: Based on the ACF/PACFs of the original data (not de-seasonalized or even de-trended), the SARIMA model parameters are: P= 0, D= 1, Q=1. I picked these model parameters by looking at the PACF, which multiple spikes at seasonal lags implies an SMA process, thus P + Q <= 1, so Q would have to equal 1 and P would have to be 0. D= 1 because there is seasonality. For the non-seasonal part, p= 1, d= 0 (no differencing), q= 2. I'm guessing this likely poorly because the ACF lags display a slow decay, indicative of an AR model and the PACF lags cut off at lag 2 and doesn't really show a slow decay.

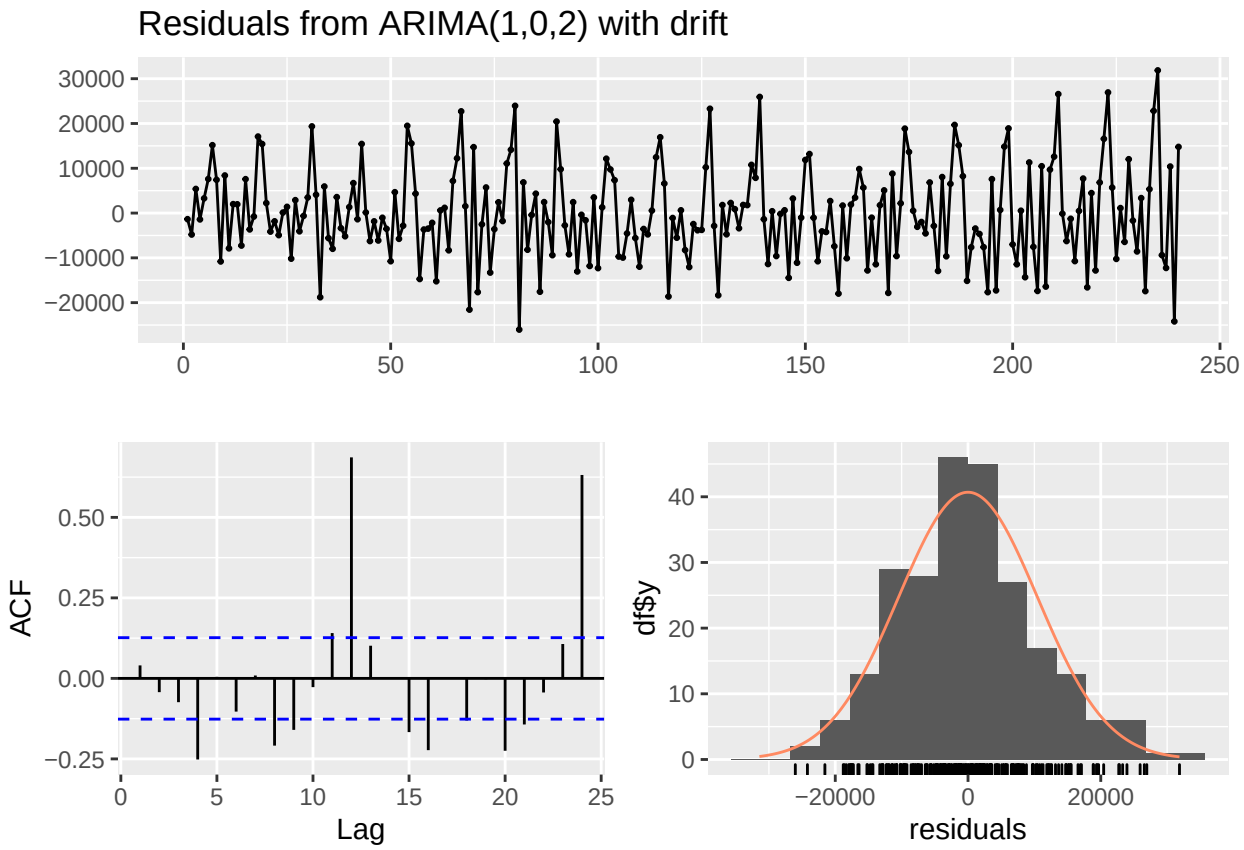
```
SARIMAmodel1 <- Arima(nat_gas_gen$Natural_Gas,order=c(1,0,2),seasonal=c(0,1,1),include.drift=TRUE)
print(SARIMAmodel1)
```

```
## Series: nat_gas_gen$Natural_Gas
## ARIMA(1,0,2) with drift
##
## Coefficients:
##          ar1     ma1     ma2  intercept     drift
##       0.3500  0.7280  0.4660  44375.615  362.5971
## s.e.  0.0855  0.0766  0.0675   4509.524   32.3582
##
## sigma^2 estimated as 111619595:  log likelihood=-2562.41
## AIC=5136.83   AICc=5137.19   BIC=5157.71
```

```
#store constants
SARIMAmodel1coeff<- c(0.3500,0.7280,0.4660, 44375.615)
cat("ARIMA model 1 constants ar1, ma1, ma2, intercept=",SARIMAmodel1coeff)
```

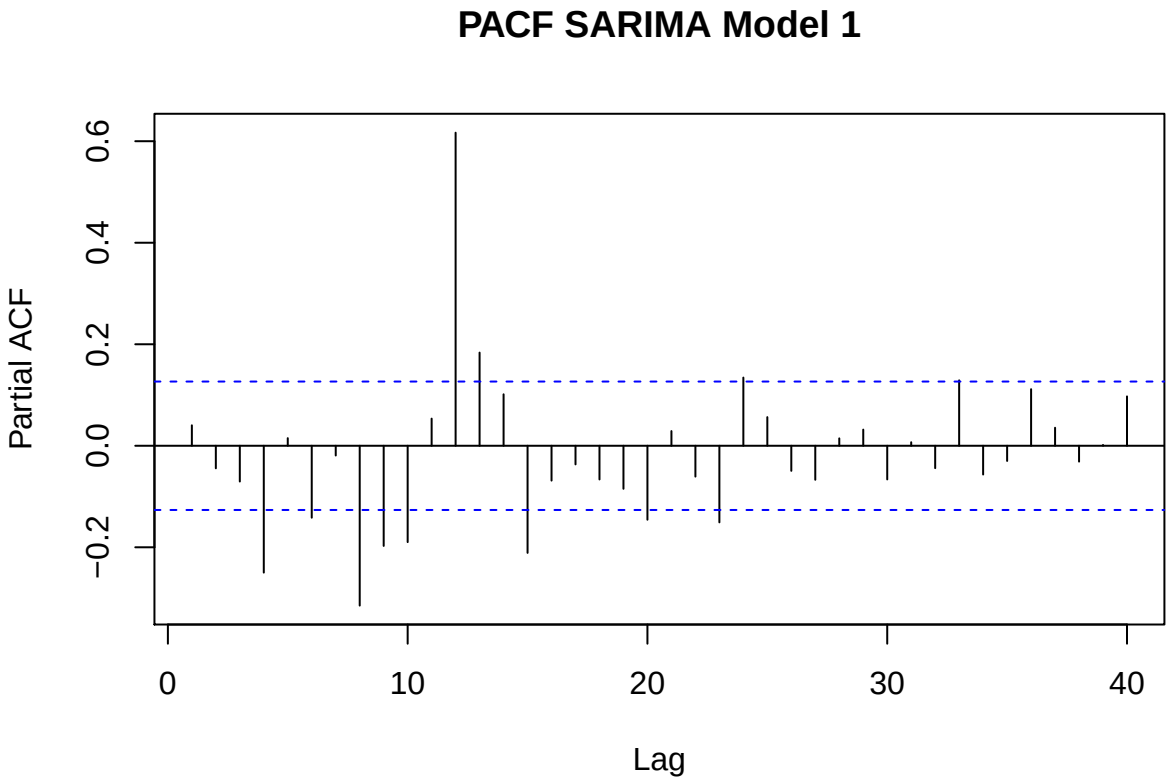
```
## ARIMA model 1 constants ar1, ma1, ma2, intercept= 0.35 0.728 0.466 44375.61
```

```
#residuals of SARIMA
checkresiduals(SARIMAmodel1,lag=40,plot=TRUE)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,2) with drift
## Q* = 485.37, df = 35, p-value < 2.2e-16
##
## Model df: 5.   Total lags used: 40
```

```
Pacf(SARIMAmodel1$residuals,lag.max=40,main="PACF SARIMA Model 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.

> Answer: The original model barely has a "lower AIC" (literally a difference of 6, AIC of 5130 compared to 5136), so it appears to be the better model only in theory. There also isn't a huge difference between the residual series plots for Q6/Q7 either. Both appear relatively white noise-like. However, it's not a fair comparison. We can't compare the AICs since they're not based on the same initial variable since one is ARIMA (can't handle seasonality) and one is SARIMA (can handle seasonality). It's not fair to compare ARIMA errors before we put back seasonality. We need to always compare/consider the same base dataset.

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?

```
auto.arima(deseason.nat_gas)
```

```
## Series: deseason.nat_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 estimated as 26980609:  log likelihood=-2383.11
## AIC=4774.21   AICc=4774.38   BIC=4788.12
```

Answer: This does not match the model parameters I picked in Q4. I originally picked ARIMA(0,1,3). According to the auto ARIMA function, the best order is ARIMA(1,1,1).

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(ts_nat_gas,include.mean=TRUE)
```

```
## Series: ts_nat_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 estimated as 27569124:  log likelihood=-2279.54
## AIC=4567.08   AICc=4567.26   BIC=4580.8
```

Answer: The output does not match what I selected in Q7. The order I picked was ARIMA(0,1,2)(0,1,1) s=12. I selected the correct P,D,Q but not the correct p,d,q. The model parameters R selects for me is ARIMA(1,0,0)(0,1,1) s=12.