PSY 653 Module 07: ARIMA Modeling

Description of the dataset:

The covid2.csv is a time series dataset providing the number of new cases of COVID within the United States. There are three variables:

- **location** = location of event (filtered to only United States)
- **date** = date of the new COVID cases
- **new_cases =** number of new COVID cases on the corresponding day

1. Download the "covid2.csv" dataset from the module 07 lab module on canvas
2. Create a new R notebook from your project file and name it "ARIMA modeling notebook"
3. Create a first level header: "Load Libraries"
    a. In a new R chunk load in the tseries, forecast, psych & tidyverse packages
4. Create an R-chunk with the first level header: "Import data"
    a. Read in the datafile "covid2.csv" save it to an object named "covid"
        i. Note: make sure to use read_csv() (instead of read.csv()) so the date variable will be read as a date (You will ***not*** need to indicate the date format [i.e. %d/%m/%Y] like we did in lab, read_csv() should read it in automatically).
5. Create a new first level header: "Get variable descriptives"
    a. Use any method to get the dataset descriptives
6. Create a first level header: "Plot the data"
    a. Plot a line graph so that the date is on the x-axis and new_cases is on the y-axis
        i. Hint: ggplot(data, aes(x = x-variable, y = y-variable)) + geom_line()
    b. Does the data look stationary? Why do you think that?
7. Convert the new_cases variable to a timeseries object using ts(), and add a frequency of 30. Name the new time series object as "count_cases"
    a. Hint: object <- ts(data$var, frequency = 30)
8. Use adf.test() to statistically test if your time series object is stationary.
    a. Hint: adf.test(object)
    b. Statistically speaking, is your data stationary?
9. Smooth your data using the stl() function with an s.window argument set to "periodic". Name the new object "decomp".
    a. Hint: decomp <- stl(count_cases, s.window = "periodic")

10. Deseasonalize your data using the seasadj() function. Name the new object "deseasonal_cnt"
    a. Hint: deseasonal_cnt <- seasadj(decomp)
11. Take the deseasonalized data and give it a difference of 1. Name the new object "count_d1"
    a. Hint: count_d1 <- diff(deseasonal_cnt, difference = 1)
12. Plot the newly seasonally adjusted data using the plot() function
    a. Hint: plot(count_d1)
    b. How does it look now? Does the differenced data look stationary?
13. Re-check the stationarity of the data using the adf.test() function
    a. Hint: adf.test(count_d1)
    b. Is the data now stationary? How do you know?
14. Create a autocorrelation plot of your count_d1 object using the Acf() function.
    a. Hint: Acf(count_d1)
    b. Looking at this plot: How many Autocorrelations terms seem like they should be included in the model?
15. Create a partial autocorrelation plot of your count_d1 object using the Pacf() function.
    a. Hint: Pacf(count_d1)
    b. Looking at this plot: How many Moving average terms seem like they should be included in the model?
16. Use the arima() function and fit, what you view is, the best fitting arima model.
    a. This may take a few steps and different people may come up with different models. That is okay!
17. Use the auto.arima() function to have R fit the best arima model.
    a. Did you and the auto.arima() function come to the same conclusions?
18. Plot a forecast of the model!
    a. Hint: autoplot(forecast(deseasonal_cnt))
19. Take a step back and view the predicted forecast you just created. In layman's terms, What did we just create with this model? What is ARIMA modeling good for?