MTH 325, Lab #8, Spring 2023   Name ____

Instructions: Follow along with the tutorial portion of the lab. Replicate the code examples in R on your own, along with the demonstration. Then use those examples as a model to answer the questions/perform the tasks that follow. Copy and paste the results of your code to answer questions where directed. Submit your response file and the code used (both for the tutorial and part two). Your code file and your lab response file should each include your name inside. Be sure to follow the write-up directions in the Lab Directions file.

In this lab, we'll begin to look at time series and their structure in R. We are going to use a couple of built-in datasets of varying complexities. For this lab, we'll look at very basic tasks like graphing and differencing. In future labs, we'll develop our analysis skills a bit further.

The three time series datasets we'll use in this lab are imported into R below.

```
data(nhtemp)
data("Seatbelts")
data("sunspots")
```

Print out the data set and compare the output to the View() function.

```
nhtemp
View(Seatbelts)
sunspots
```

One thing you should notice when you print out the dataset to the console rather than use View() is that time series display their datetime stamps upon print, but don't display those when you use View(). For complex time series data, it can be helpful to just look at a single timestamped numerical variable from the data. Pulling out the data column will also pull out the datetime stamp.

```
killed<-Seatbelts[,1]
killed
```

Moreover, as you can see, if there is monthly data, it will be displayed in the form of an array, rather than a single list.

We can use the plot() function to obtain a quick graph of the data. This function will automatically place Time on the horizontal axis, and the measured numerical variable on the vertical axis.

```
plot(nhtemp)
plot(killed)
plot(sunspots)
plot(Seatbelts)
```

Notice that the first three of these plots, there is only one time series plotted, but Seatbelts contains multiple plots, and the function plots each time series on a smaller subgraph.

This plot function can be adapted similarly to when we used plot() earlier in the course.

```
plot(nhtemp, main="New Haven Temperatures",type="b",ylab="temperature")
```

Here, I've added a title to graph, a new y-axis label and type "b" adds the circles where the observations are.

There are also packages that deal directly with time series, with their own plotting functions and analysis tools that we'll use later. The one we'll look at here is called TSstudio. You'll need to install it before we go on.

```
library(TSstudio)
```

The plotting functions are particular to time series graphs and are built on plotly in the background. Some of the graphs can be made interactive. Let's try out the basic plot on the three single series graphs.

```
ts_plot(nhtemp)
ts_plot(killed)
ts_plot(sunspots)
```

These graphs retain some interactive functions like zoom options. Play around with the interactive options that appear on the graph. We can add additional features if we want, such as a slider bar, as well as a title and axis labels.

```
ts_plot(sunspots,title = "Sunspot Observations, Monthly", Xtitle = "Time",
        Ytitle = "Number of Sunspots",slider=TRUE)
```

The slider is on the bottom, pull the bars from the edges to where you want it and it will narrow the range of the larger graph so that you can see more detail.

You can also change other features of the graph, like colors and thickness of the line.

```
ts_plot(sunspots,title = "Sunspot Observations, Monthly", Xtitle = "Time",
        Ytitle = "Number of Sunspots",color = "black", width = 3)
```

We can also make other style adjustments like points for observations or dashed lines.

```
ts_plot(nhtemp,title = "Temperatures in New Haven", Xtitle = "Time",
        Ytitle = "Temperature",dash="dash", line.mode =  "lines+markers")
```

The ts_plot() function works a little differently than plot() when there are multiple time series. In the default case, it will plot all the time series on a single graph. You can adjust it to "multiple" graphs so thee is one for each time series, but this works best when there are fewer series than there is available in the Seatbelts dataset. This function will plot all numerical series in the data.

```
ts_plot(Seatbelts)
ts_plot(Seatbelts,type = "multiple")
```

We can look at the properties of time series. Some basic elements are shown below. These functions give the start time (first observation), end time (last observation), delta-t (period between observations), frequency (observations per year), time (prints out decimal equivalents of observations in an array), cycle (prints out the observation number per year in an array), and if it's a time series or not.

```
start(sunspots)
end(sunspots)
deltat(sunspots)
frequency(sunspots)
time(sunspots)
cycle(sunspots)
is.ts(sunspots)
```

Some aspects of time series analysis requires a time series to be stationary (essentially just random noise). We'll first simulate white noise to see what a stationary time series looks like.

```
white_noise<- arima.sim(model = list(order = c(0, 0, 0)), n = 100)
ts.plot(white_noise)
```

You can try rerunning the code a couple times to see different versions of white noise. Since we didn't set a seed, each graph will be a little different. The important thing to note in all these graphs, like our residual plots, there should be no pattern, and the mean should be zero (though mean zero need not always be the case).

Another kind of basic time series is the random walk. This model arises when errors are correlated rather than independent of each other.

```
random_walk <- arima.sim(model = list(order = c(0, 1, 0)), n = 100)
ts.plot(random_walk)
```

Again, run it more than once to see variations on this model type. The random walk should reduce to white noise by finding what is called the first difference: essentially the change between successive steps in the time series. This is a common model of analysis to remove trends from time series data.

We can perform this analysis on our datasets.

```
dif1<-diff(nhtemp)
plot(dif1)
diff1<-diff(killed)
plot(diff1)
```

These graphs now look fairly random, so that's good. It's less clear that this works on the sunspot data.

```
diff2<-diff(sunspots)
plot(diff2)
```

There is a strong seasonal (11-year) component to the sunspot data and it's not clear to me we've removed it all. We can try removing a seasonal component by using a lag as shown below. However, we'll also look at dealing with seasonal decomposition in a future lab. I chose lag=132 since that is 11×12 months, but feel free to experiment with other lags to see if that improves the picture any.

```
diffs<-diff(sunspots,lag=132)
plot(diffs)
```

We can calculate second and third differences as well, as many as necessary to remove any trend. Just apply diff() again to the differences. Each time you apply it with the default lag=1 you'll lose one observation. If you do a seasonal lag, you'll lose that many observations, each time. These differences are similar to derivatives since the time series is regularly spaced, which means that first differences can remove a linear trend, second differences a quadratic trend, etc. You can apply other transformations like log() to eliminate an exponential trend.

In addition to operations specific to time series, we can also do other kinds of analysis on numerical variables such as correlations and pair plots when we have more than one time series, histograms, boxplots, and qqplots. Some examples are shown below, though this list is not exhaustive.

```
cor(Seatbelts)
pairs(Seatbelts)
hist(killed)

qqnorm(killed, pch = 1, frame = FALSE)
qqline(killed, col = "steelblue", lwd = 2)
```

Tasks

1. Load the built-in dataset Nile. Use the time series functions to find the start time, end time, time step size, frequency, and confirm that the dataset is a time series. Create a nice plot of the data with labels and other appropriate features. Create some basic numerical plots of the data such as a qqplot, histogram and boxplot of the data. Describe what you find. Calculate first differences. Is the data stationary?

2. The built-in dataset uspop is for US Population data from the census. Plot the time series in a nice graph. You can see that the growth trend is exponential. Apply a log transformation. Then replot the data. Calculate and plot first differences until the results are stationary. Discuss how many steps were needed. Create exploratory plots of the stationary data.

3. Use the built-in dataset austres for Australian residential data. Plot the time series in a nice graph using different features than previous graphs. This data trend is linear. Calculate and plot first differences until the results are stationary. Discuss how many steps were needed. Create exploratory plots of the stationary data.

Resources:

1. https://cran.r-project.org/web/packages/TSstudio/vignettes/Plotting_Time_Series.html
2. https://rpubs.com/odenipinedo/time-series-analysis-in-R
3. https://r-coder.com/plot-r/
4. http://www.sthda.com/english/wiki/qq-plots-quantile-quantile-plots-r-base-graphs