MTH 325, Lab #9, 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.

Complex time series are typically composed of three components: a seasonal component, a trend component, and error or noise. We are going to look at how to separate these three components from each other. We previously looked at differencing as a way of obtaining the noise component. Now we are going to let a package do this work for us.

Recall from the last lab, we looked at the dataset Seatbelts and pulled out the time series of the number of drivers killed. Let's look at it again.

```
data("Seatbelts")
killed<-Seatbelts[,1]
killed
plot(killed)
```

There is a seasonal pattern to the data (most likely related to winter weather). Let's decompose the data into its component parts using decompose(). Then plot the result.

```
kdecomp<-decompose(killed)
plot(kdecomp)
```

We get four subgraphs. The top one is the original plot. The second is the trend, which can be nonlinear. Here it looks like the random walk we looked at in the last lab. The third plot is the seasonal component. And the final plot is the error or noise.

Let's try this again with the sunspots data set.

```
data(sunspots)
sunspots
plot(sunspots)

sundecomp<-decompose(sunspots)
plot(sundecomp)
```

There is a lot more data here, and so it's much harder to see the structure of the seasonal component. It may be worthwhile for a dataset like this to break it down into smaller components before decomposing it. But, we'll set that aside for now.

It's important to note here that you may encounter an error if you apply the decompose() function to a time series without any seasonal component. For example, if we apply decompose() to the uspop data from the tasks in the last lab, you will get an error. In such cases, you will need to obtain the trend through regression methods. In the last lab next week, we'll look at other modeling techniques that can also be used to obtain trends when automatic decomposition fails.

Once we have the decomposition, we can look at individual pieces and produce separate graphs.

```
summary(sundecomp)
sundecomp$trend
plot(sundecomp$trend)
plot(sundecomp$seasonal)
plot(sundecomp$random)
```

Summary() will give us the list of components, and then we can look at the time series on the console, or plot each one separately. The seasonal component still looks like a mess here. And the "trend" looks pretty seasonal to me. The decomposition function may be having an issue with the 11-year sunspot cycle.

We can extract the trend or the seasonal component from the data by subtracting it off the original (for instance, if we wanted to see a seasonally adjusted plot.

```
sunadj<-sunspots-sundecomp$trend
plot(sunadj)

sunadjs<-sunspots-sundecomp$seasonal
plot(sunadjs)
```

The resulting plot confirms my suspicion that the "trend" here is the seasonal 11-year sunspot cycle.

We can also use LOESS to model the trend and decompose the time series. Here we'll look at the sunspot data and motor vehicle deaths from the Seatbelts dataset.

```
sundecomp2<-stl(sunspots,s.window="periodic")
plot(sundecomp2)

kdecomp2<-stl(killed,s.window="periodic")
plot(kdecomp2)
```

The order of the plots is a little different, and the plotting style for the errors also, but basically, this gives the same information as the decompose() function. For even more methods, check out [3] in the resource list.

One of the differences between time series and cross-sectional data is that errors may not be independent. Values in the series may be correlated with each other. We can plot graphs of these autocorrelation relationships with two kinds of special plots: an ACF graph, and a PACF graph. Let's look at the Seatbelts death data to start.

```
acf(killed)
pacf(killed)
```

On the horizontal axis is the lag (in years), and the lines indicate the correlation values. 0-lag corresponds to the original value, which is obviously 1 since this is measuring correlation with itself. But the correlation with the previous value in the sequence is still 0.6, which is pretty high. The graphs plot lines of the graph that show where statistical significance is. Lines that are taller than the horizontal line should be included in any auto-regressive model, as are any negative correlations that extend beyond the lower line. Correlations in this data increase again around lag-1 (equal to a year) which is a sign of the seasonal component of the data.

The pacf() graph is similar, but it calculated a little differently. If the autocorrelation is especially strong, you get something that looks like the sunspot data.

```
acf(sunspots)
pacf(sunspots)
```

Here, you can see the strong relationship extends at a significant level even to the three-year mark.

In the next lab we'll look at building models of time series data that will allow us to make predictions. Some of them will be autoregressive and relate future values to previous values. The number of those values we'll need will be selected from looking at these autocorrelation graphs.

Tasks

1. For the built-in datasets UKgas and UKDriverDeaths, plot the time series. Then decompose them into their trend, seasonal and noise components using the method of your choice. Describe both the seasonal and trend components. Remove the seasonal component and plot the resulting seasonally adjusted data. Create ACF and PACF plots. Describe what you see. Create numerical plots (histograms, boxplots, qqplots) of the full time series data and the decomposed elements. What do you notice about them and their relationships to each other.

Resources:

1. https://rpubs.com/odenipinedo/time-series-analysis-in-R
2. https://www.datascienceinstitute.net/blog/time-series-decomposition-in-r
3. https://cran.r-project.org/web/packages/stR/vignettes/stRvignette.html
4. https://www.dominodatalab.com/blog/time-series-with-r