Time series analysis

Teaching note:

A special case of two-variable analysis is the study of a time series where one of the two variables is time. In this case we may wish to understand how the other variable of interest changes over time. We may look for a directional trend, and for evidence of a seasonal cycle, and use these to try to predict the future movements of the data.

Getting the data:

For time series analysis, you will be using historical monthly price data for the S&P 500 index from January 2000 to December 2012. This data and similar data is available from Yahoo! Finance (for example, http://finance.yahoo.com/q/hp?s=%5EGSPC+Historical+Prices).

Use the following code to read the CSV file and examine the structure of the dataset:

```
Index<-read.csv("SP500.csv")
head(index)
```

One problem with this data from Yahoo! Finance is that it is provided in reverse chronological order. We can flip it around with the following:

```
Index <- Index[rev(rownames(index)),]
head(Index)
```

It will be helpful to wrap the data in an R time series object, which offers some affordances for plotting and data analysis based on the assumption that the data points are sequential and equidistant in time. The `ts()` function creates the time series object:

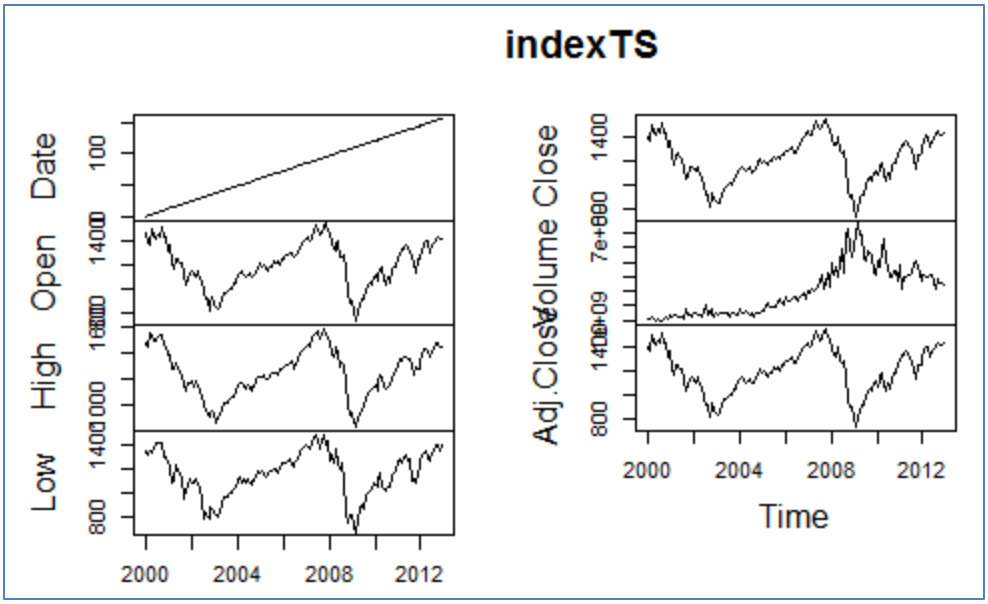
```
indexTS<-ts(Index,start=c(2000,1),end=c(2012,12),frequency=12)
```

The `frequency=12` parameter implies that there are 12 measurements per "time unit", and the other parameters tell R that that the data starts from the first measurement in time unit 2000 and ends with the 12th measurement in time unit 2012. We recognize these as months in a span of years, but the same `ts()` function with different parameters could identify days within weeks or quarters within years, etc.

Plot the data:

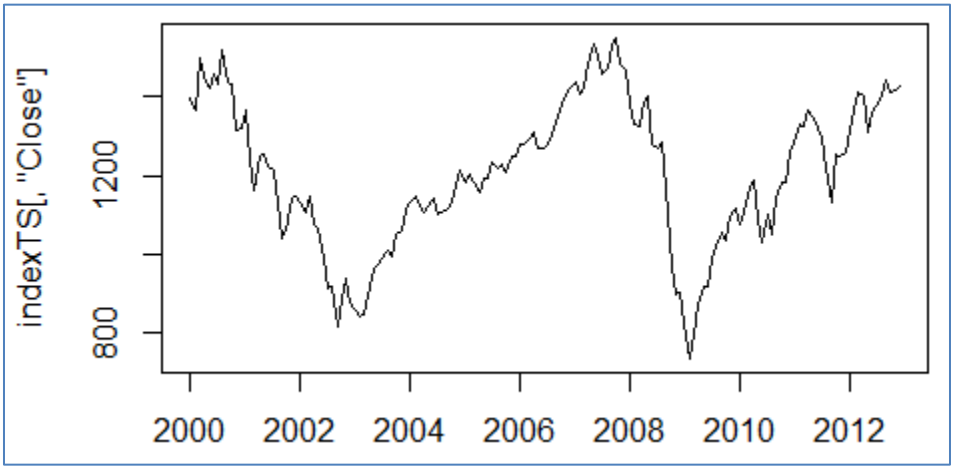
1. Let's take a look at the data. Because it is wrapped in an R time series object, we can make a rough plot of all the variables with the simple line:

   ```
   plot(indexTS)
   ```



2. That plot includes a little more information than we needed. We can plot a single variable, the closing price on the first trading day of each month, with the following:

```
plot(indexTS[,"Close"])
```



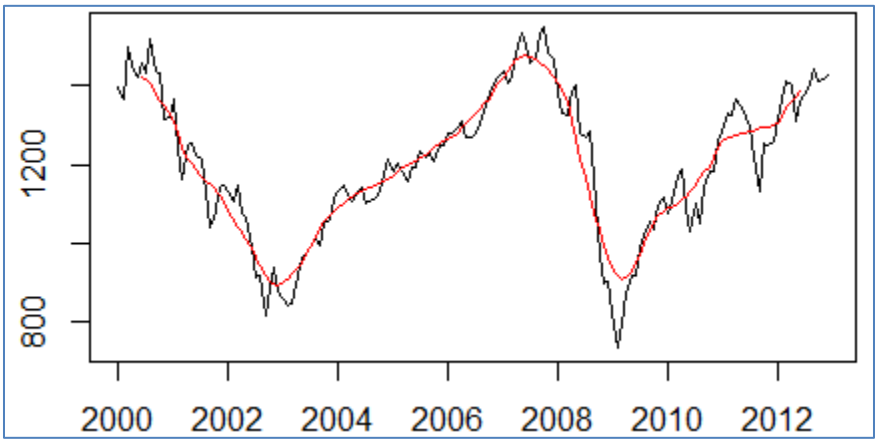
Determine trends and seasonality:

1. A trend line refers to a smoothing of the data to show its general direction and hopefully minimize seasonal effects and random noise. One of the simplest ways to determine a trend line is with a moving average.

   To implement a moving average, we take advantage of R's `filter()` function for time series. The second parameter of this function accepts a set of weights to be given to each of several data points in the moving average (a feature which would allow us to implement non-uniformly weighted moving averages if we wanted). By default, the

moving average is "centered", but a back-looking moving average is also possible. Consult `?filter` for more details. The expression `rep(1/12,12)` means to create a vector that repeats the value 1/12 twelve times. Twelve data points is a good choice for our moving average because it corresponds to the number of months in a year and may mitigate seasonality. Finally, `filter(indexTS[,"Close"],rep(1/12,12))` creates a 12-month moving average. We wrap it in the `lines()` function so that it will be added to the previous plot:

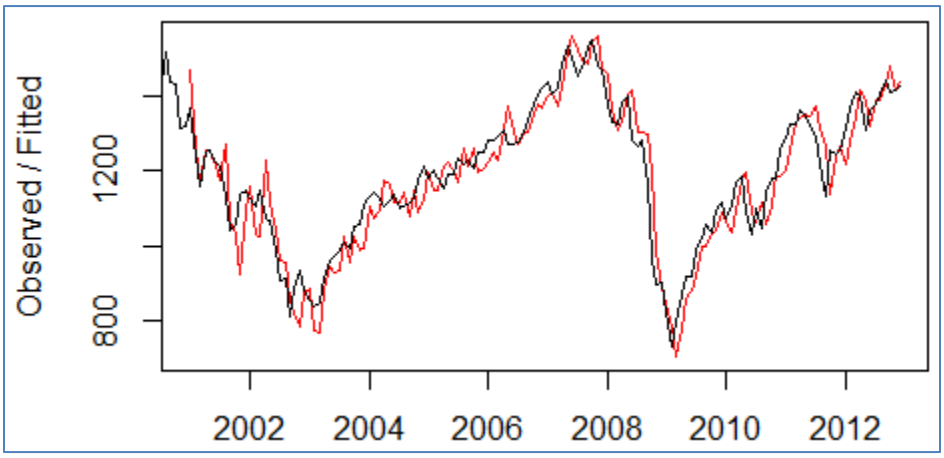
```
lines( filter(indexTS[,"Close"],rep(1/12,12)), col="red" )
```



Some of the problems with moving averages are that they miss the big peaks and troughs in the data, and that they cannot be calculated for the endpoints. Since you often want to use the trend line to predict the future, the latter is a big problem. Another subtler weakness of our particular trend line is that with an even number of data points, it is a bit lopsided and doesn't center perfectly on the data. An odd number would produce a trend line slightly more faithful to the data.

2. Another oft-used method for smoothing a time series is exponential smoothing or the "Holt-Winters method". You can compute and plot a quick Holt-Winters model with:

```
plot(HoltWinters(indexTS[,"Close"]))
```



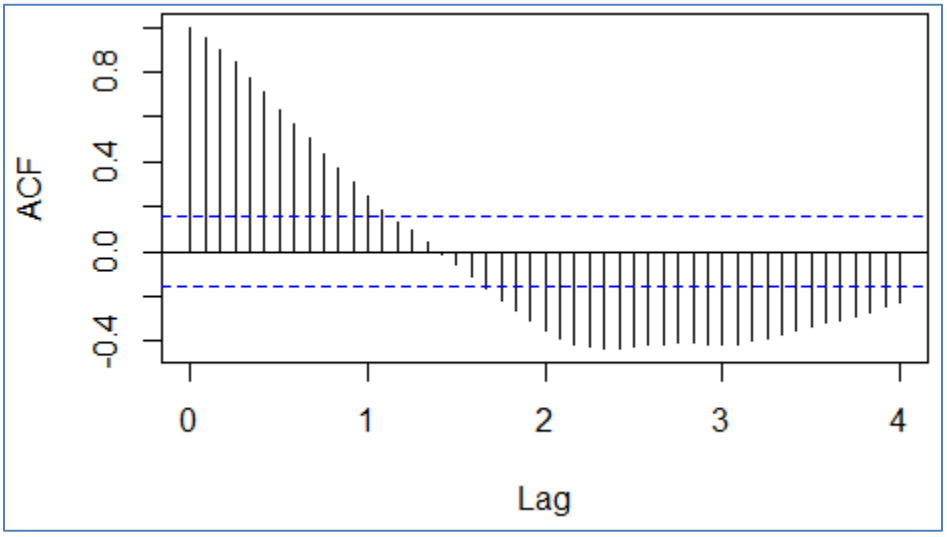
This code allows R to automatically determine what it deems to be optimal values for the α, β, and γ parameters. If you wish to specify your own, or to specify other variations like a multiplicative model, the `HoltWinters()` function can accommodate this. Type `?HoltWinters` for more details on the function. For example, you could specify:

```
plot(HoltWinters(indexTS[,"Close"],alpha=0.4,beta=0.4,gamma=0.4))
```

Consult a good textbook to understand what these parameters do.

3. The autocorrelation function is a tool that data scientists use to diagnose seasonality in a time series. It shows how well the data is correlated with itself at different amounts of lag. If our data goes through an annual cycle, for example, we might expect a relatively strong correlation between each data point and the 12th data point after it. We can view the autocorrelation function with `acf()`.

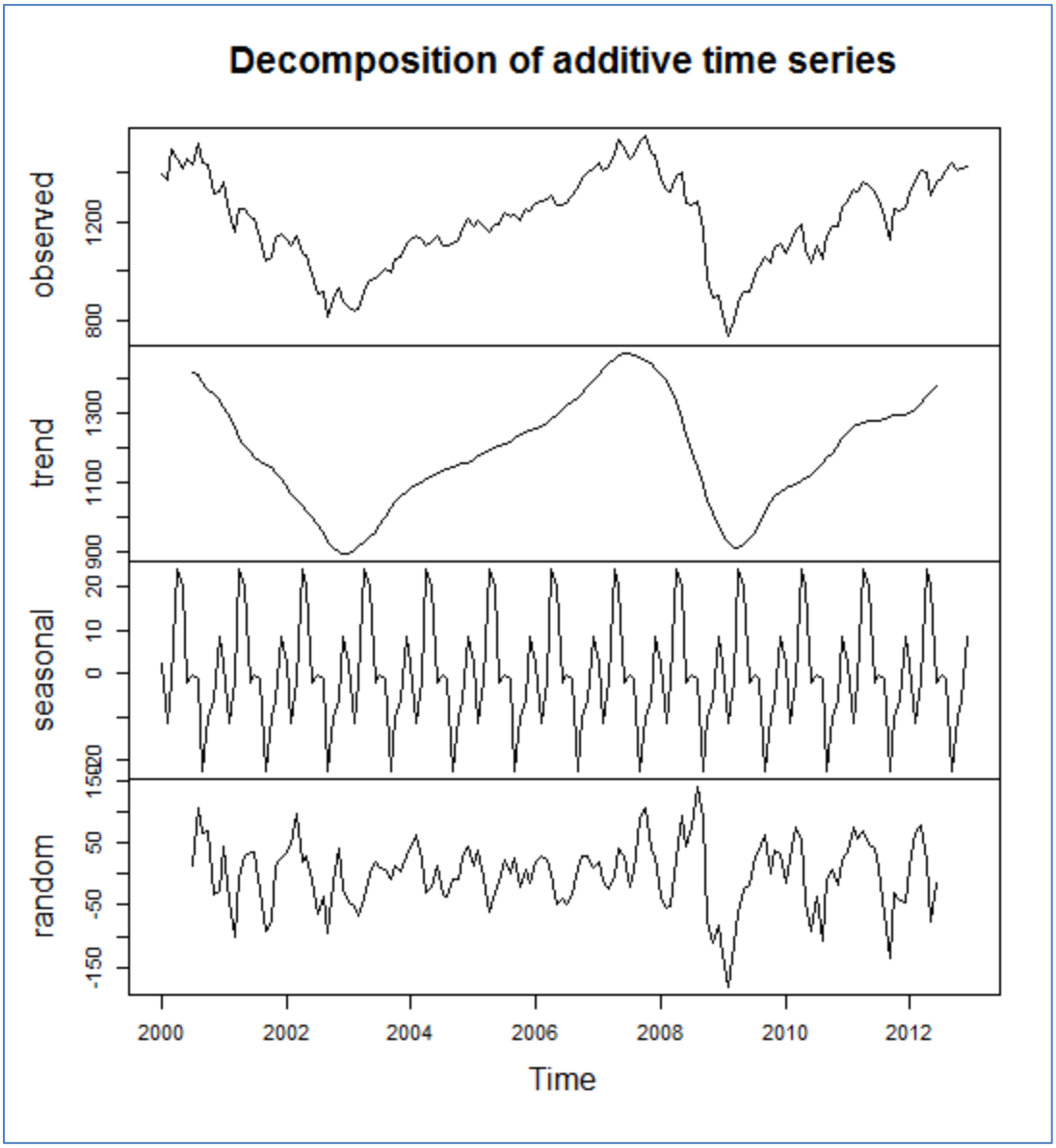
```
acf(indexTS[,"Close"])
acf(indexTS[,"Close"],lag.max=48)  # shows acf even further out
```



In the plot, the x variable is "time units", years in our case, so the x value of 1 actually corresponds to the 12th lagged data point. In this graph we see no spike of autocorrelation at 12 months or at any other point. Instead we see a strong and monotonically decreasing autocorrelation in each of the first several data points. This indicates that (as you would expect), the S&P500 index moves incrementally and in any given month will not be much different from its value in the previous month. In fact this effect probably swamps the influence of any possible seasonal effect.

4. We can let R do the work of separating out the trend line and seasonal component (and error component) of the data with the `decompose()` command. According to its help file (`?decompose`), this uses a classical method employing moving averages.

```
indexDTS <- decompose(indexTS[,"Close"])
plot(indexDTS)
```



Extend the time series with a prediction:

1. If you can build a model to describe the data as a function of time, you can generate predicted values to extend the time series into the future. It is unlikely that you can do this for the S&P500 index, but we'll show you the code just as an exercise. We will create a model based on the Holt-Winters method we used previously:

   ```
   HWmodel <- HoltWinters(indexTS[,"Close"])
   ```

   Then use the model to predict the next five years (60 values):

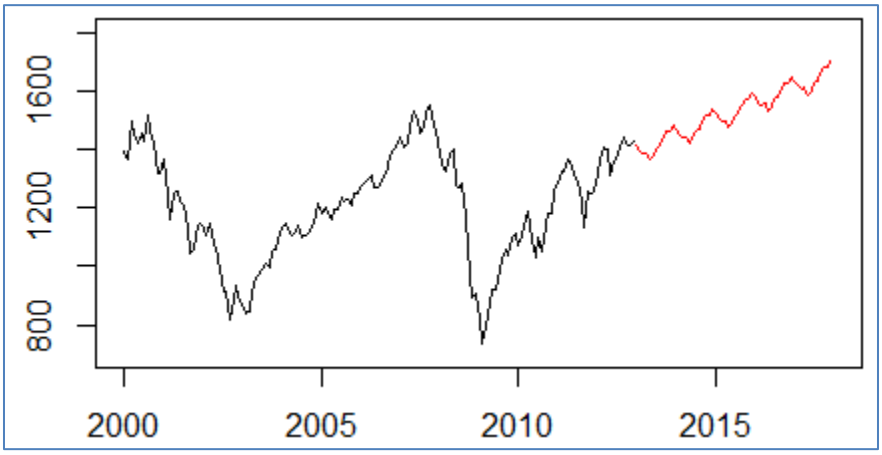
```
prediction <- predict(HWmodel,n.ahead=60)
```

Let's plot the original time series next to the prediction. Start by plotting the time series with `xlim` and `ylim` parameters that leaves space for the predicted values:

```
plot(indexTS[,"Close"],xlim=c(2000,2018),ylim=c(700,1800))
```

Now add the predicted values:

```
lines(prediction,col="red")
```



The result is a prediction that takes into account the trend line at the end of 2012 as well as the apparent seasonality in the time series.