Time Series Visuaization

Jack Baumruk

December 3, 2017

Introduction

A lot of data that one will encounter in daily life are of a particular variety: Time Series Data. Time series data is data where the observations are strictly sequential, that is the are ordered in some many and shuffling them would eliminate their meaning. Most often, these data sets are ordered by time, hence the name, such as the daily price of a stock, or the average temperature in a location by month. However, sometimes this data is ordered by some other metric such as distance, but most often it will be in terms of time. Regardless, time series data has very particular behaviors and properties that make it unique from other kinds of datasets, and while we will not go into how all of these are precisely applied here, we will demonstrate how one can use R to visualize these analyses most elegantly and effectively.

Generating some Data

For this post, we will generate two sets of time series data to use. Each will contain 500 observations. Note, we will be setting a random seed that must be used in order to replicate the below datasets.

The first will be a series X with the following formula:

$$X_i = \epsilon_i$$

where ϵ_i are independent random variables with a Normal distribution of mean 0 and variance of 1. This is a simple series known as “white noise.”

```
# Set seed for reproducibility
set.seed(12345)

# generate x as specified
x <- rnorm(500, mean = 0, sd = 1)
```

The second series will be Y defined by:

$$Y_i - Y_{i-12} = \epsilon_i$$

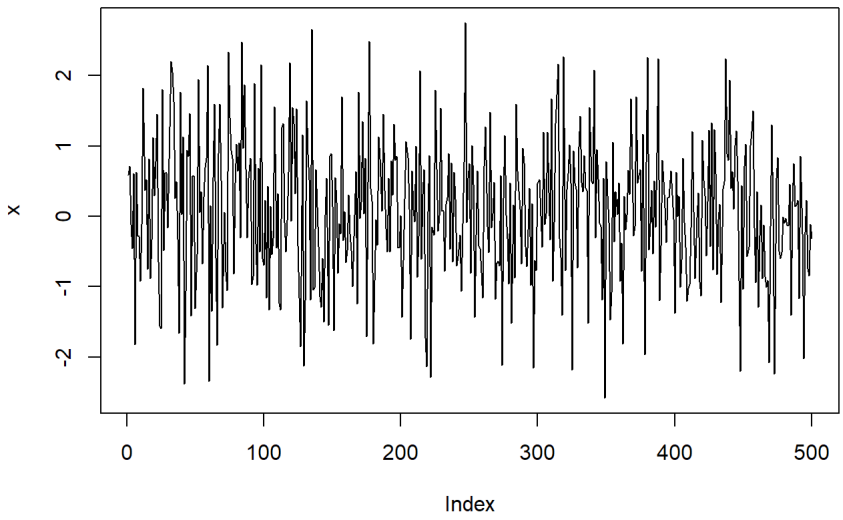
where $epsilon_i$ is defined as above. This series with a seasonal component that one might see if looking at monthly data (since there are 12 months in a year).

```
# Set seed for reproducibility
set.seed(11111)

# initialize 12 values to seed the series
y_initial <- rnorm(12, mean = 0, sd = 1)

# generate the random component for the series
epsilon <- rnorm(500, mean = 0, sd = 1)

# generate y as specified
y <- c(y_initial, rep(0, 500))
for(i in 13:512){
  y[i] <- y[i-12] + epsilon[i-12]
}
y <- y[-(1:12)]
```

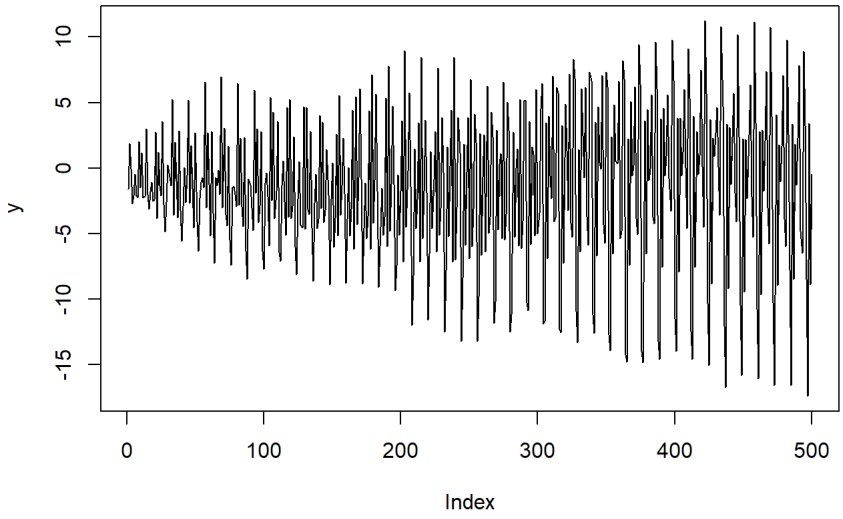
Basic Plots

The most straightforward way to plot the data against the index (time in this case) which we do here for our datasets. Note, typically time series will be plotting by not showing the points, but instead simply connecting them by lines, which we specify using `type = 'l'` in the `plot` function.

```
# Plot x
plot(x, type = 'l')
```




```
# Plot y
plot(y, type = 'l')
```

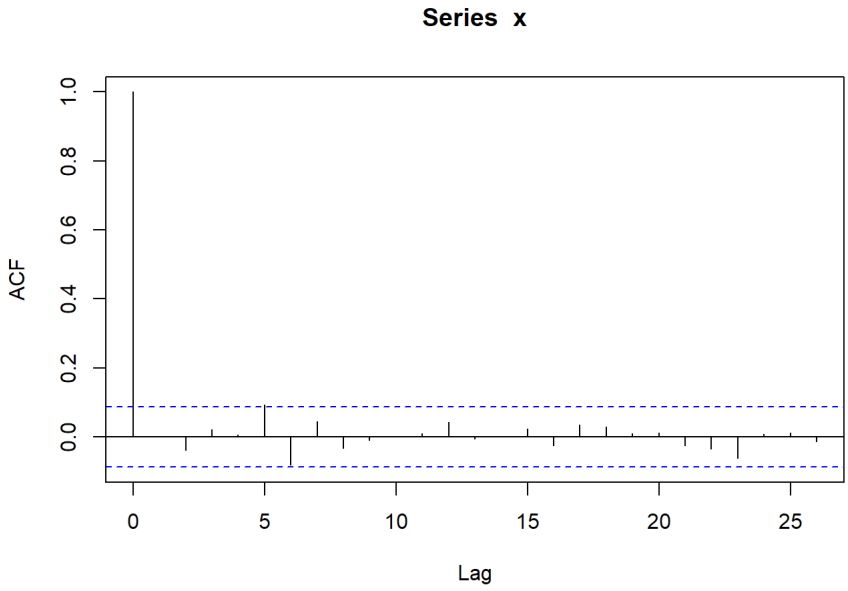


Notice the random “noisiness” in the first plot, and the periodicity in the second.

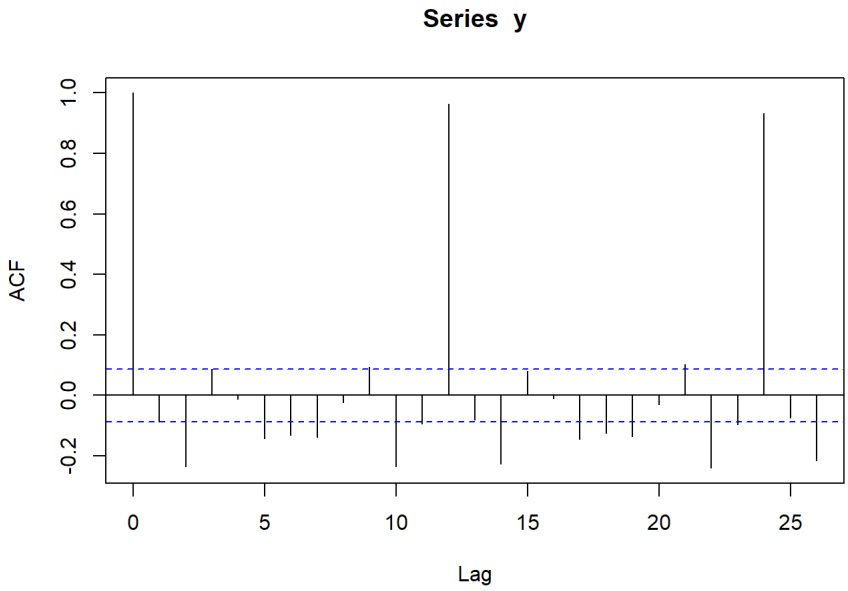
The `acf()` Function

The auto-correlation function (ACF) is one of the most important tools for analyzing time series data. This function, for some whole number “lag” k, is given by the correlation between points that are k apart from each other. That is, if we compare the data set to the same data set shifted backwards by k steps and take the correlation of these two, we get the ACF. R has a very usuful tool for generating and visualizing an ACF, unsurprisingly called `acf()`. It is easy to apply as well, only requiring one argument, the time series. The ACF for our X and Y data are shown here.

```
# ACF for x
acf(x)
```




```
# ACF for y
acf(y)
```



An important thing to note in these plots is that the `acf()` function shows a dotted blue line on either side of zero indicating the 95% confidence interval that the correlation is non-zero.

We notice that the first plot has a correlation of one at zero (as it always should since any data is perfectly correlated to itself) and the rest of the lags have correlations near zero, which it should as we chose each element of X independently and randomly.

The second plot looks much different. We notice spikes in correlation at lag 12 and 24 which is expected based on the forumla, but there are several other significant, though smaller, correlations in between. There are several reasons for this, which I won't go into here, but it is important to see how visualizations like this can help us learn more about a given series, figure out what it is doing, and better understand why it has certain properties.

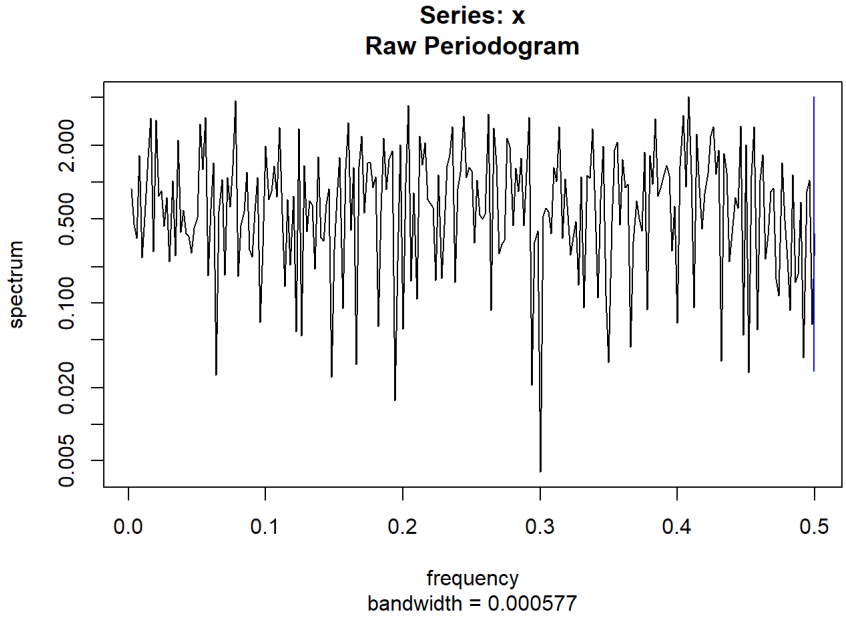
The `spectrum()` Function

Probably the next most important visual tool for time series analysis is the spectral density, also known as the periodogram. The spectral density depicts the presence of various frequencies at which the data oscillates, specifically in the form of sine functions. The idea is that if you tried to fit a sum of sine functions with various frequencies to the data, the spectral density depicts what the coefficient on each of these different frequency sine functions would be. The spectral density is an important tool for finding various patterns in the data, especially when frequency is known to be an important aspect of the data, such as in sound recordings or physical oscillations of an object.

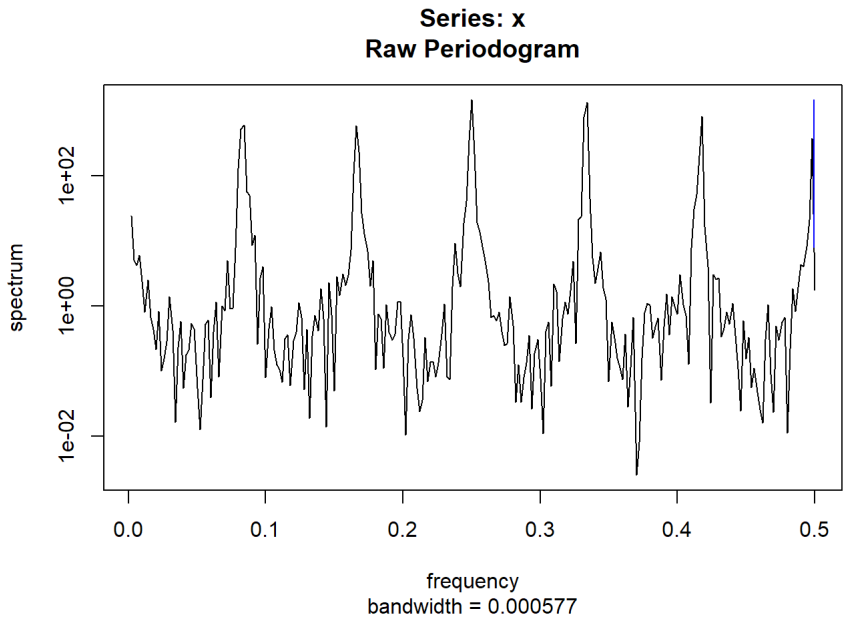
Like with the ACF, R supplies a usful function for plotting the spectral density, the `spectrum()` function. The name is referring to the concept of looking at the frequency "spectrum" of a time series to analyze patterns, which, clearly enough, is the root work for "spectral" in spectral density. The term "periodogram" is also very often used to describe the plot of the spectral density, and the name refers to the plotting of the "periods" of the function.

We display the spectral densities using `spectrum()` for our two datasets:

```
# Spectral density of x
spectrum(x)
```




```
# Spectral density of y
spectrum(y)
```



Notice here that the x-axis runs from 0 to 0.5 where the units are taken to be number of oscillations divided by the length of the series. For example, the value at 0.1 refers to the frequency that has 50 oscillations out of the 500 data points, indicating a period of 10 units. The periodogram only goes from 0 to 0.5, rather than to 1, because it happens to be the case that $f(x) = f(1-x)$ for x the axis of this plot so showing 0.5 to 1 would just be a mirror image and provide no extra information. I leave it to the reader to check why this would be true.

Also notice how again we have the first plot appearing very "random" with no real pattern, and the second showing a clear periodicity, the first spike of which (at about 0.083) corresponds to the period of 12 that we know is in the data.

Conclusion

We have looked at a couple of important tools for visualizing time series data and the way to create these in R. As we showed, many of these visualizations are already built into R and are very simple to use, you likely won't even need to install any packages to do most basic time series analysis. It is important to know how to use these tools, however, if you ever come across a time series analysis task, as the are the best and easiest way to visualize and understand your data, so make sure to know what functionality R has for you to do so.

References

- Bartlett, Peter. "Introduction to Time Series Analysis". Lecture as part of *STAT 153* at UC Berkeley. https://www.stat.berkeley.edu/~bartlett/courses/153-fall2010/lectures/3.pdf.
- Srivastava, Tavish. "A Complete Tutorial on Time Series Modeling in R". *Analytics Vidhya*. https://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/.
- Hyndman, Rob J. "CRAN Task View: Time Series Analysis." *The Comprehensive R Archive Network*. https://cran.r-

project.org/web/views/TimeSeries.html.

- R-core. “acf”. *RDocumentation*. https://www.rdocumentation.org/packages/stats/versions/3.4.1/topics/acf.
- Penn State Eberly College of Science. “1.2 Sample ACF and Properties of AR(1) Model”. Lesson as part of *STAT 510* at Penn State. https://onlinecourses.science.psu.edu/stat510/node/60.
- R-core. “specturm”. *RDocumentation*. https://www.rdocumentation.org/packages/stats/versions/3.4.1/topics/spectrum.
- Sparing, Helen J. *Spectral Analysis in R*. https://ms.mcmaster.ca/~bolker/eeid/2010/Ecology/Spectral.pdf.