Post02: An Introduction to Time Series in R

Alex Yi

November 27, 2017

Introduction

In Statistics 133, the two main models we used included Linear Regression, and Loess Regression. However, other models can be used, such as time series models. A time series is a series of data points that are listed in time order, such that it is usually a series of discrete time data points. As the name suggests, the time periods can include years, days, hours, or minutes. Through the use of time series analysis, one can analyze time series data to extract meaningful characteristics of the data, and through the use of time series forecasting, one can use a model to predict future values based on previously observed values.

Motivation

One may ask why I chose to select this as a topic, and not just study it further in classes such as Statistics 153. Through my readings and prior research experiences, time series has reared its head again and again. The presence of time series data in our world is overwhelming, and includes applications to fields such as finance, signal processing, and ecology. In this post, I hope to introduce the concept of time series, and how to analyze and forecast in R.

Background

Time series analysis and forecasting is an extensive field of study, and for the sake of this blog post, I will do my best to summarize the core concepts. As aforementioned, the study of time series can largely be broken down into analysis and forecasting, where analysis describes the underlying statistics and forces behind the data, and forecasting allows for prediction of future values.

When analyzing time series, most patterns can be described in terms of trend and seasonality. Trend shows the generalized linear portion that changes over time and does not repeat within the captured data, where seasonality shows the repetition within the systematic intervals over time.

In trend analysis, one can identify trend via smoothing (a.k.a filtering in signal processing). This involves applying a function to approximate the data yet largely preserve trend and seasonality. Perhaps the most common smoothing technique is known as the moving average, in which each element in the time series is replaced by the average of n surrounding elements, where n is the size of the moving “window.” After smoothing the data, the trend can be easily identified, and then detrended (removal of trend) to further study the seasonality.

Seasonality can be detected using Fourier Transform, which decomposes the data (signal) into the frequencies that make it up. It can also be measured by autocorrelation, where autocorrelation is defined by the correlation between the ith element and the (i-k) element, given that the seasonality repeats every k elements. As a result, autocorrelations between consecutive seasonalities are dependent, and this can be defined as serial dependence. Serial dependence can be removed by converting each ith element to the difference from the (i-k) element. This is meant to make the series stationary for forecasting. A stationary series can be defined as having a constant mean, having homoscedasticity (variance should not be function of time), and having a constant autocorrelation and covariance.

Perhaps the most popular forecasting method is known as ARMA, or Auto-Regressive Moving Average. This consists of two common processes: Autogressive process, and moving average process. In an autoregressive process, each observation is made up of a random error, as well as a linear combination of a given number of prior observations. Furthermore, the process will only be stable if the series is stationary. The other method, moving average, is defined such that each observation is made up of a random error component, and a linear combination of prior random shocks. This requires invertibility, where the moving average can be rewritten as an autoregressive form (of infinite order), which furthermore depends on the series being stationary.

Example:

Trend Analysis:

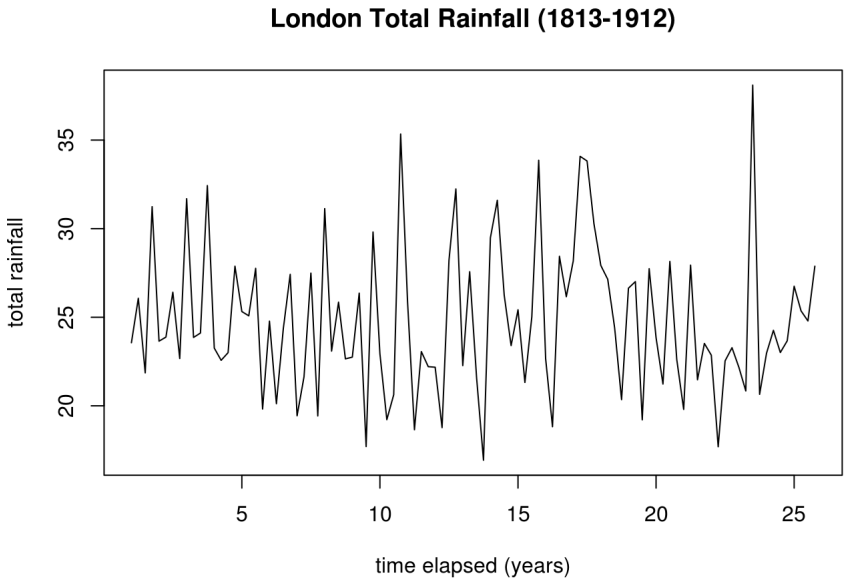
With the cursory overview of the theory out of the way, we can begin to look at the dataset. For simplicity’s sake, the dataset here is the total annual rainfall in inches of London, England from 1813-1912.

```
#load data set
london <- read.csv("total-annual-rainfall-in-inches-.csv")
colnames(london)[2] <- 'total-rainfall'
head(london)
```

```
##   Year total-rainfall
## 1 1813          23.56
## 2 1814          26.07
## 3 1815          21.86
## 4 1816          31.24
## 5 1817          23.65
## 6 1818          23.88
```

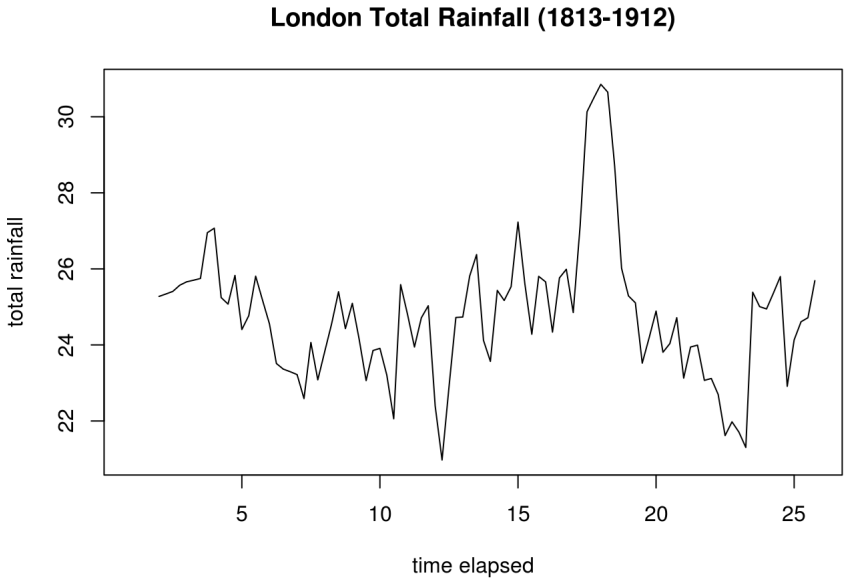
Once the csv is loaded, the next step is to store the data as a time series object in R.

```
#convert data frame into a time series object. Because only one column represents
#the time series, we isolate that.
londonts<- ts(london, frequency = 4)[,2]
plot.ts(londonts, xlab = "time elapsed (years)", ylab = "total rainfall", main = "London Total Rainfall (1813-1912
)")
```



In accordance to the information given in the background, we can observe that there is largely no trend here, as there is no apparently long term movement. Nevertheless, we can decompose the trend by applying a Moving Average Filter via the TTR package.

```
library("TTR")
londontsMA3 <- SMA(londonts, n = 5)
plot.ts(londontsMA3, xlab = "time elapsed", ylab = "total rainfall",
        main = "London Total Rainfall (1813-1912)")
```



As seen in the data after a moving average is applied, characteristics of the time series data become more prominent. Relative peaks and valleys can be smoothed out, and larger periods of drought and/or rainfall can be further emphasized. In the case of this filter, there is a prominent period of rain between 65-70 elapsed years, which translates to around the 1880s.

Using decompose(), we can identify the trend component, seasonal component, and irregular component. We can then plot this data.

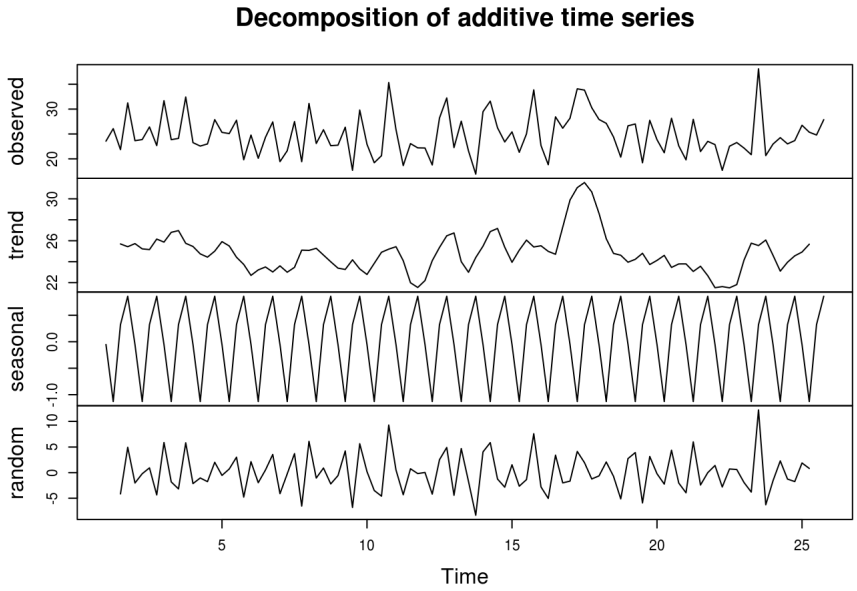
```
compLondonts <- decompose(londonts)
compLondonts
```

```
## $x
##      Qtr1  Qtr2  Qtr3  Qtr4
## 1   23.56 26.07 21.86 31.24
## 2   23.65 23.88 26.41 22.67
## 3   31.69 23.86 24.11 32.43
## 4   23.26 22.57 23.00 27.88
## 5   25.32 25.08 27.76 19.82
## 6   24.78 20.12 24.34 27.42
## 7   19.44 21.63 27.49 19.43
## 8   31.13 23.09 25.85 22.65
## 9   22.75 26.36 17.70 29.81
## 10 22.93 19.22 20.63 35.34
## 11 25.89 18.65 23.06 22.21
## 12 22.18 18.77 28.21 32.24
```

```
## 13 22.27 27.57 21.59 16.93
## 14 29.48 31.60 26.25 23.40
## 15 25.42 21.32 25.02 33.86
## 16 22.67 18.82 28.44 26.16
## 17 28.17 34.08 33.82 30.28
## 18 27.92 27.14 24.40 20.35
## 19 26.64 27.01 19.21 27.74
## 20 23.85 21.23 28.15 22.61
## 21 19.80 27.94 21.47 23.52
## 22 22.86 17.69 22.54 23.28
## 23 22.17 20.84 38.10 20.65
## 24 22.97 24.26 23.01 23.67
## 25 26.75 25.36 24.79 27.88
##
## $seasonal
##           Qtr1        Qtr2        Qtr3        Qtr4
## 1  -0.05259115 -1.12951823  0.32313802  0.85897135
## 2  -0.05259115 -1.12951823  0.32313802  0.85897135
## 3  -0.05259115 -1.12951823  0.32313802  0.85897135
## 4  -0.05259115 -1.12951823  0.32313802  0.85897135
## 5  -0.05259115 -1.12951823  0.32313802  0.85897135
## 6  -0.05259115 -1.12951823  0.32313802  0.85897135
## 7  -0.05259115 -1.12951823  0.32313802  0.85897135
## 8  -0.05259115 -1.12951823  0.32313802  0.85897135
## 9  -0.05259115 -1.12951823  0.32313802  0.85897135
## 10 -0.05259115 -1.12951823  0.32313802  0.85897135
## 11 -0.05259115 -1.12951823  0.32313802  0.85897135
## 12 -0.05259115 -1.12951823  0.32313802  0.85897135
## 13 -0.05259115 -1.12951823  0.32313802  0.85897135
## 14 -0.05259115 -1.12951823  0.32313802  0.85897135
## 15 -0.05259115 -1.12951823  0.32313802  0.85897135
## 16 -0.05259115 -1.12951823  0.32313802  0.85897135
## 17 -0.05259115 -1.12951823  0.32313802  0.85897135
## 18 -0.05259115 -1.12951823  0.32313802  0.85897135
## 19 -0.05259115 -1.12951823  0.32313802  0.85897135
## 20 -0.05259115 -1.12951823  0.32313802  0.85897135
## 21 -0.05259115 -1.12951823  0.32313802  0.85897135
## 22 -0.05259115 -1.12951823  0.32313802  0.85897135
## 23 -0.05259115 -1.12951823  0.32313802  0.85897135
## 24 -0.05259115 -1.12951823  0.32313802  0.85897135
## 25 -0.05259115 -1.12951823  0.32313802  0.85897135
##
## $trend
##        Qtr1     Qtr2     Qtr3     Qtr4
## 1        NA       NA 25.69375 25.43125
## 2  25.72625 25.22375 25.15750 26.16000
## 3  25.87000 26.80250 26.96875 25.75375
## 4  25.45375 24.74625 24.43500 25.00625
## 5  25.91500 25.50250 24.42750 23.74000
## 6  22.69250 23.21500 23.49750 23.01875
## 7  23.60125 22.99625 23.45875 25.10250
## 8  25.08000 25.27750 24.63250 23.99375
## 9  23.38375 23.26000 24.17750 23.30750
## 10 22.78125 23.83875 24.90000 25.19875
## 11 25.43125 24.09375 21.98875 21.54000
## 12 22.19875 24.09625 25.36125 26.47250
## 13 26.74500 24.00375 22.99125 24.39625
## 14 25.48250 26.87375 27.17500 25.38250
## 15 23.94375 25.09750 26.06125 25.40500
## 16 25.52000 24.98500 24.71000 27.30500
## 17 29.88500 31.07250 31.55625 30.65750
## 18 28.61250 26.19375 24.79250 24.61625
## 19 23.95125 24.22625 24.80125 23.73000
## 20 24.12500 24.60125 23.45375 23.78625
## 21 23.79000 23.06875 23.56500 22.66625
## 22 21.51875 21.62250 21.50625 21.81375
## 23 24.15250 25.76875 25.54000 26.06750
## 24 24.60875 23.10000 23.95000 24.56000
## 25 24.92000 25.66875       NA       NA
##
## $random
##            Qtr1         Qtr2         Qtr3         Qtr4
## 1            NA           NA -4.156888021  4.949778646
## 2  -2.023658854 -0.214231771  0.929361979 -4.348971354
## 3   5.872591146 -1.812981771 -3.181888021  5.817278646
## 4  -2.141158854 -1.046731771 -1.758138021  2.014778646
## 5  -0.542408854  0.707018229  3.009361979 -4.778971354
## 6   2.140091146 -1.965481771  0.519361979  3.542278646
## 7  -4.108658854 -0.236731771  3.708111979 -6.531471354
## 8   6.102591146 -1.057981771  0.894361979 -2.202721354
## 9  -0.581158854  4.229518229 -6.800638021  5.643528646
## 10  0.201341146 -3.489231771 -4.593138021  9.282278646
## 11  0.511341146 -4.314231771  0.748111979 -0.188971354
## 12  0.033841146 -4.196731771  2.525611979  4.908528646
## 13 -4.422408854  4.695768229 -1.724388021 -8.325221354
```

```
## 14  4.050091146  5.855768229 -1.248138021 -2.841471354
## 15  1.528841146 -2.647981771 -1.364388021  7.596028646
## 16 -2.797408854 -5.035481771  3.406861979 -2.003971354
## 17 -1.662408854  4.137018229  1.940611979 -1.236471354
## 18 -0.639908854  2.075768229 -0.715638021 -5.125221354
## 19  2.741341146  3.913268229 -5.914388021  3.151028646
## 20 -0.222408854 -2.241731771  4.373111979 -2.035221354
## 21 -3.937408854  6.000768229 -2.418138021 -0.005221354
## 22  1.393841146 -2.802981771  0.710611979  0.607278646
## 23 -1.929908854 -3.799231771 12.236861979 -6.276471354
## 24 -1.586158854  2.289518229 -1.263138021 -1.748971354
## 25  1.882591146  0.820768229           NA           NA
##
## $figure
## [1] -0.05259115 -1.12951823  0.32313802  0.85897135
##
## $type
## [1] "additive"
##
## attr(,"class")
## [1] "decomposed.ts"
```

```
plot(compLondonts)
```

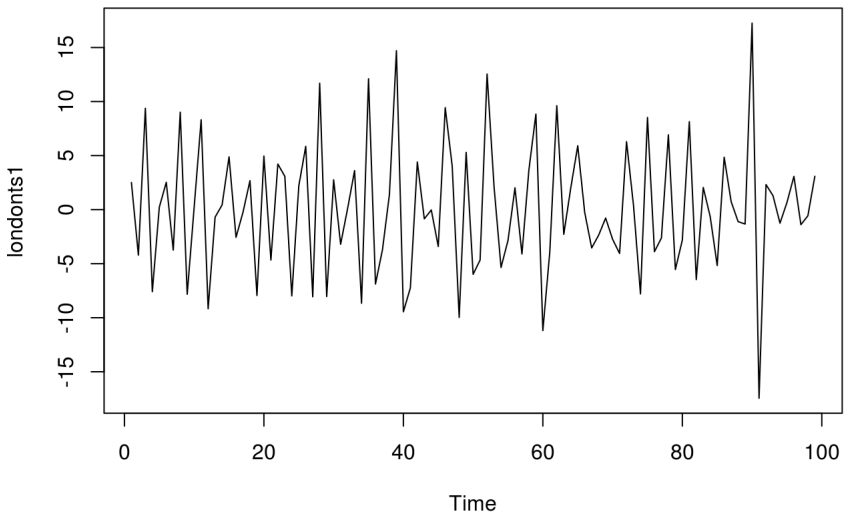


Upon inspection of the dataset, the rain levels approximately repeated every 4 years. However, one can merely observe the above graph to recognize that this is not always the case, as sometimes there may be longer dry and/or wet periods than usual, resulting in differing bouts of seasonality.

Forecast Analysis:

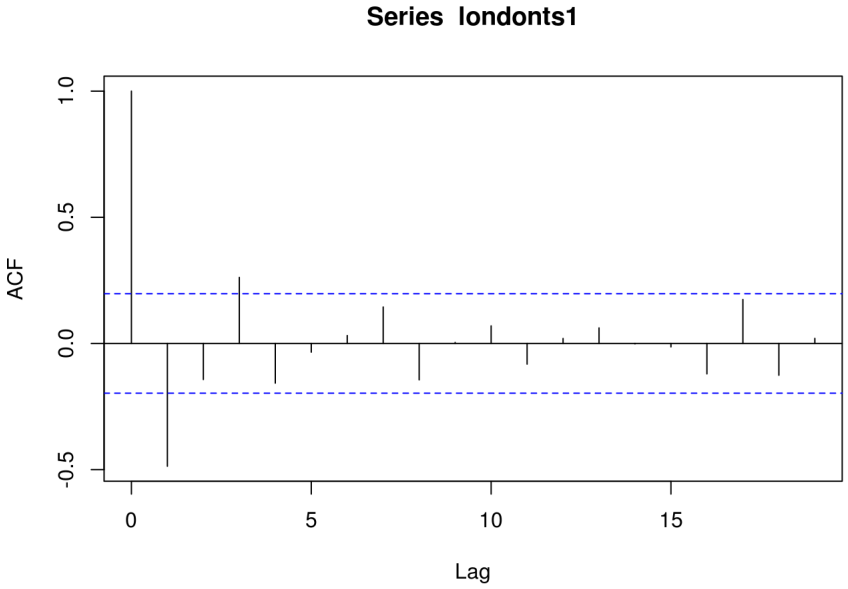
In this example I will choose to use ARIMA for the model at hand. Because ARIMA requires the time series to be stationary, I can use diff() to obtain “suitably lagged and differenced” values.

```
#calculate lagged and iterated differences to make series stationary
londonts1 <- diff(london[[2]], differences = 1)
plot.ts(londonts1)
```



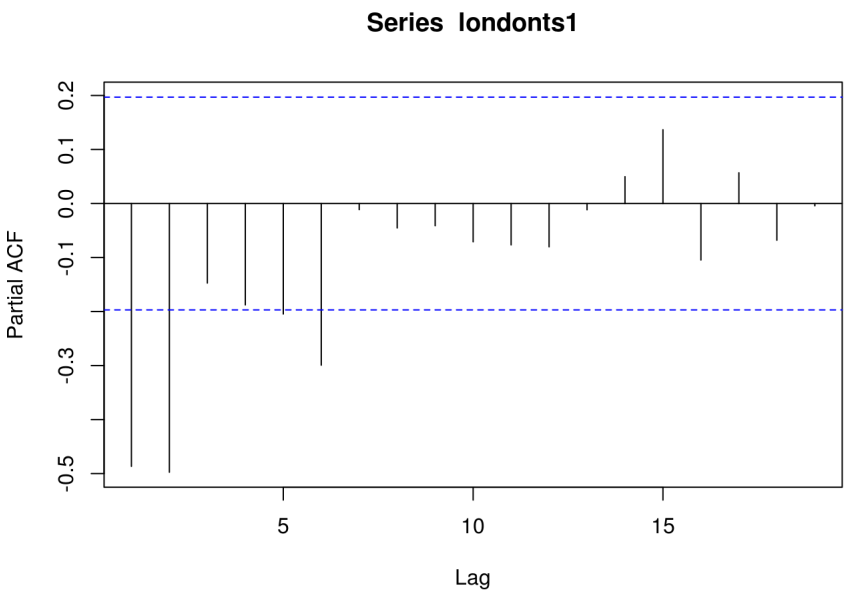
This resultant time series appears to be very stationary in mean, so we can proceed further. We then examine the autocorrelations between values.

```
#autocorrelation of differenced time series.
acf(londonts1)
```



From the correlogram above, we can observe there is a correlation of 1 at lag 0, and the other significant bounds are at lag 1 (-0.487), and lag 3(0.262). We can further this analysis by examining the partial autocorrelations:

```
#examination of partial autocorrelation
pacf(londonts1)
```



As seen in this diagram, the lags of 0, 1, 5, and 6 exceed the significance bounds.

Although we COULD use the above visualizations to determine the appropriate coefficients, the library("forecast") allows us to shortcut all of this via the method auto.arima(), which returns the best fit for the given time series. Woops. This is implemented below as follows:

```
#find the best arima model:
library("forecast")
auto.arima(londonts1)
```

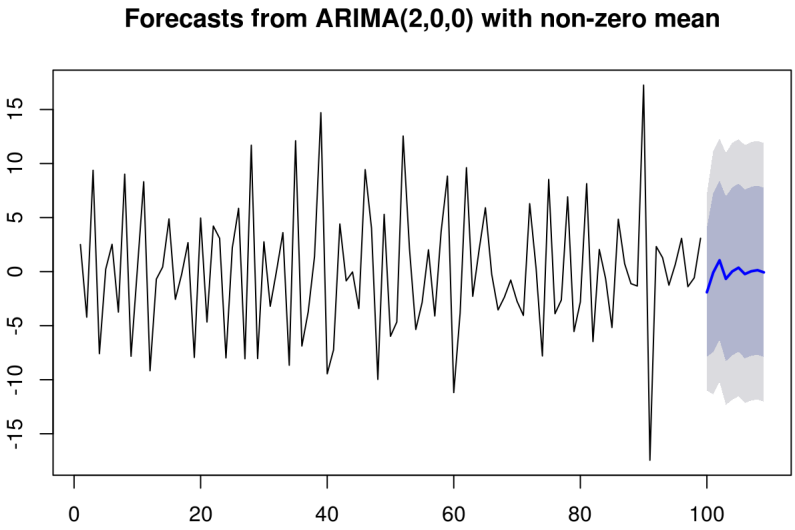
```
## Series: londonts1
## ARIMA(2,0,0) with zero mean
##
## Coefficients:
##           ar1      ar2
##       -0.7247  -0.4913
## s.e.   0.0868   0.0861
##
## sigma^2 estimated as 22.06:  log likelihood=-293.01
## AIC=592.01   AICc=592.27   BIC=599.8
```

```
#build the arima, and forecast the next 10 years?
london_arima <- arima(londonts1, order = c(2,0,0))
london_forecasts <- forecast(london_arima, h = 10)
london_forecasts
```

```
##     Point Forecast     Lo 80    Hi 80     Lo 95     Hi 95
## 100    -1.90677222 -7.863884 4.050340 -11.01739  7.203844
## 101    -0.08391198 -7.440802 7.272979 -11.33531 11.167481
## 102     1.05008666 -6.309567 8.409740 -10.20553 12.305705
## 103    -0.66730397 -8.287323 6.952715 -12.32112 10.986510
## 104     0.02010848 -7.752043 7.792260 -11.86637 11.906589
## 105     0.36573395 -7.407654 8.139122 -11.52264 12.254105
## 106    -0.22247192 -8.023102 7.578158 -12.15251 11.707562
## 107     0.03397968 -7.785329 7.853289 -11.92462 11.992581
## 108     0.13712855 -7.682506 7.956764 -11.82197 12.096228
## 109    -0.06362054 -7.886173 7.758932 -12.02718 11.899941
```

With these forecasts, along with the 80% and 95% confidence intervals for these predictions, we can plot as follows:

```
plot(london_forecasts)
```



For further testing of the validity of these results, one could further examine the forecast errors via tests such as the Ljung-Box test.

Conclusion:

In this post, I strove to introduce you to the concept of a time series, and how one can analyze the current data, and use these preexisting values to predict, or “forecast”, future values. In the example I provided above, I selected a data set of 100 years of total annual rainfall in London, and conducted analysis on both the current data via trend analysis and decompose(). I then used ARIMA to forecast the total rain in London for the next ten years. Overall, the study of time series obviously extends way beyond this post, but the overall goal of this post was to be a cursory introduction. I hope you found this information useful, and can now run basic scripts on time series to not only better yourself as a statistician, but better your understanding of how the world around you works.

References:

https://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/ https://cran.r-project.org/web/views/TimeSeries.html https://a-little-book-of-r-for-time-series.readthedocs.io/en/latest/ http://r-statistics.co/Time-Series-Analysis-With-R.html https://ac.els-cdn.com/S0925400507007691/1-s2.0-S0925400507007691-main.pdf?_tid=68fffd1c-d4b5-11e7-a11f-00000aab0f01&acdnat=1511926307_009a6aa59be2807bb6ca6569de963856 http://www.statsoft.com/Textbook/Time-Series-Analysis

DATA:

https://datamarket.com/data/set/22np/total-annual-rainfall-in-inches-london-england-1813-1912#!ds=22np&display=line