Linear Time Series Analysis:

the Growing Trend of Probiotics

Andre Starr

#912249700

December 2017

**Abstract**

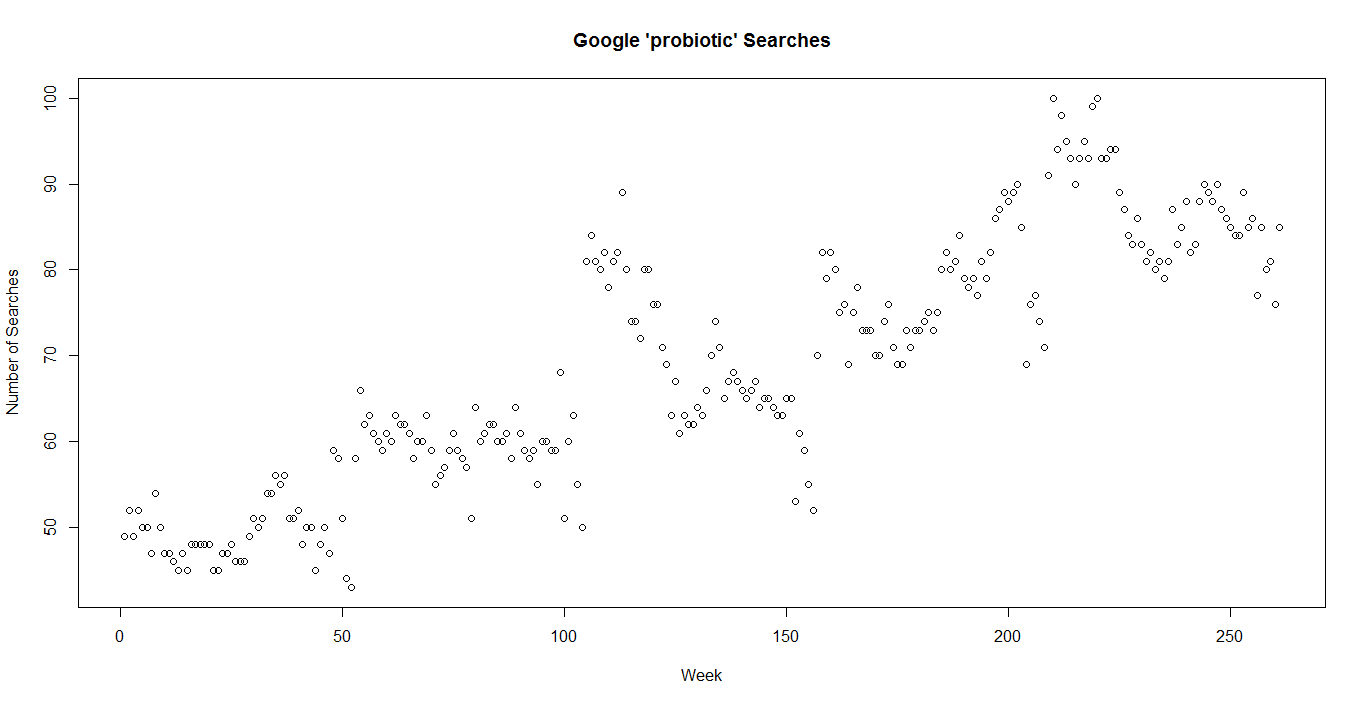
In this paper we conduct a linear time series analysis on the number of times the term ‘probiotics’ is searched weekly on Google. We will explain how we collected the data, and then provide some basic analysis on what the data looks like. We will proceed to explain our process of decomposing the time series data into its relevant components—trend, seasonality, and the remaining random component. We will then develop models for the random component, and also for the prediction of future observations from time series with all components included. Along the way, we will utilize various diagnostic graphs and procedures to assess the accuracy and quality of our results. Much of our analysis will be done using built-in R functions and packages. In the Discussion section we will summarize our findings, evaluating potential errors and shortcomings. We will conclude by explaining how our findings might have value to someone like a product manager.

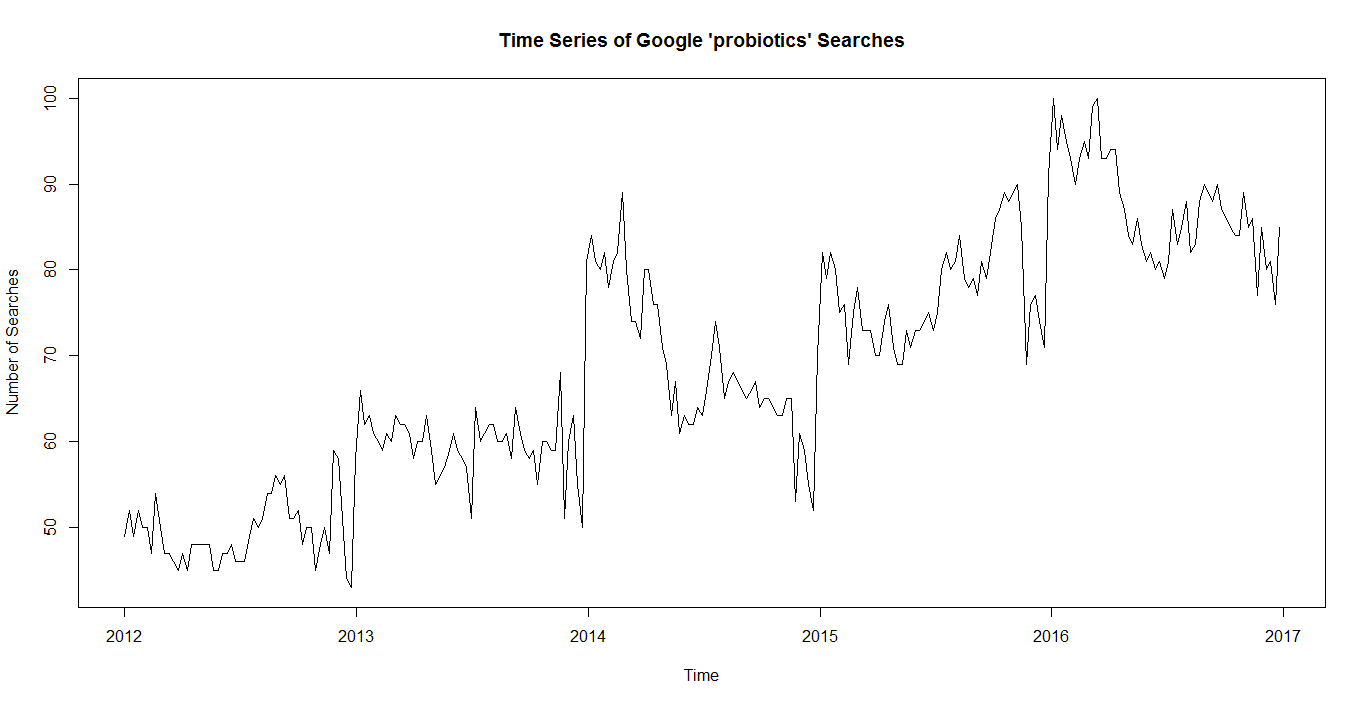
**Introduction**

Personal health has becoming an increasingly popular topic over the years, and for good reason—we like being healthy. This has led people to take up all sorts of diets and exercise regiments. Some pursue the fitness regimen known as CrossFit, while others may put themselves on a strict low-carbohydrate diet known as a ketogenic diet. Others will avoid eating genetically modified food. Some of these trends spur debate as to whether they actually improve one’s health. But regardless of how one attempts to better themselves, the goal is the same—to be healthier, and as a result, happier. One of the many health trends that seems to be gaining popularity as of late is probiotics, which help regulate the bacteria in your gut.

**Data Description**

Data was acquired using *google.com/trends* and searching for the phrase ‘probiotics’. The resulting search data was stored as a .csv file and then loaded into R. The number of searches was reported weekly for five years, ranging from January 1, 2012 to December 31, 2016, which is a sample size of 261. Five years multiplied by 52 weeks per year would be 260 data points, so I’m not sure why there is an additional week being recorded (nor where in the dataset it is located)— it is probably because 2012 and 2016 were leap years with one additional day. We are interested in how the number of weekly searches varies over the course of the year, indicating that a single cycle is 52 weeks long (or 52 weeks plus one day in the case of a leap year.) Thus, our ‘frequency’ of our cycle is 52, but this isn’t precisely correct due to the extra observation. To remedy this, in R the leap years can apparently be accommodated for by setting the ‘frequency’ equal to 365.25/7—I’m not sure about the specifics of how this works, but I’ve read online that it does (website referenced on last page in ‘References’.) Using 365.25/7 as the frequency when creating the time series object sets the frequency equal to 52.1785714285714. I’m not sure if this step is necessary, or if we could just use 52 and not worry about the extra day in leap years. The results don’t seem to change much either way. Regardless, we choose to use 365.25/7. Provided below is a plot of the data observations, with the corresponding time series below it.





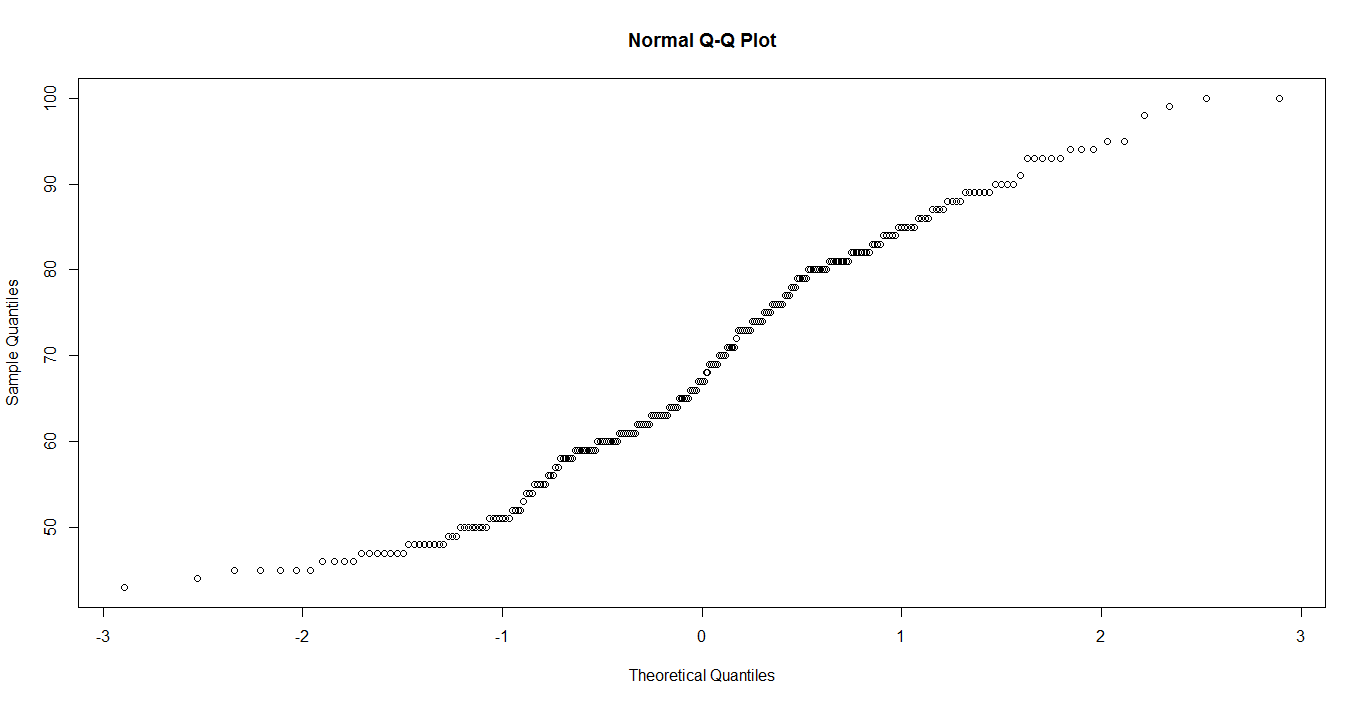
Some basic summary statistics for the data are as follows:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Minimum | 1st Quarter | Median | Mean | 3rd Quarter | Maximum | Variance |
| 43.00 | 58.00 | 67.00 | 68.63 | 81.00 | 100.00 | 210.05 |

Looking at the plot of the data, there appears to be a consistent, linear, upward trend, and a rough yearly seasonality. There appears to be a large decrease around December and a large increase around January every year. The data seems to be roughly normally distributed, and we will check with a q-q plot later. The data also appears to have roughly constant variance.

**Data Analysis**

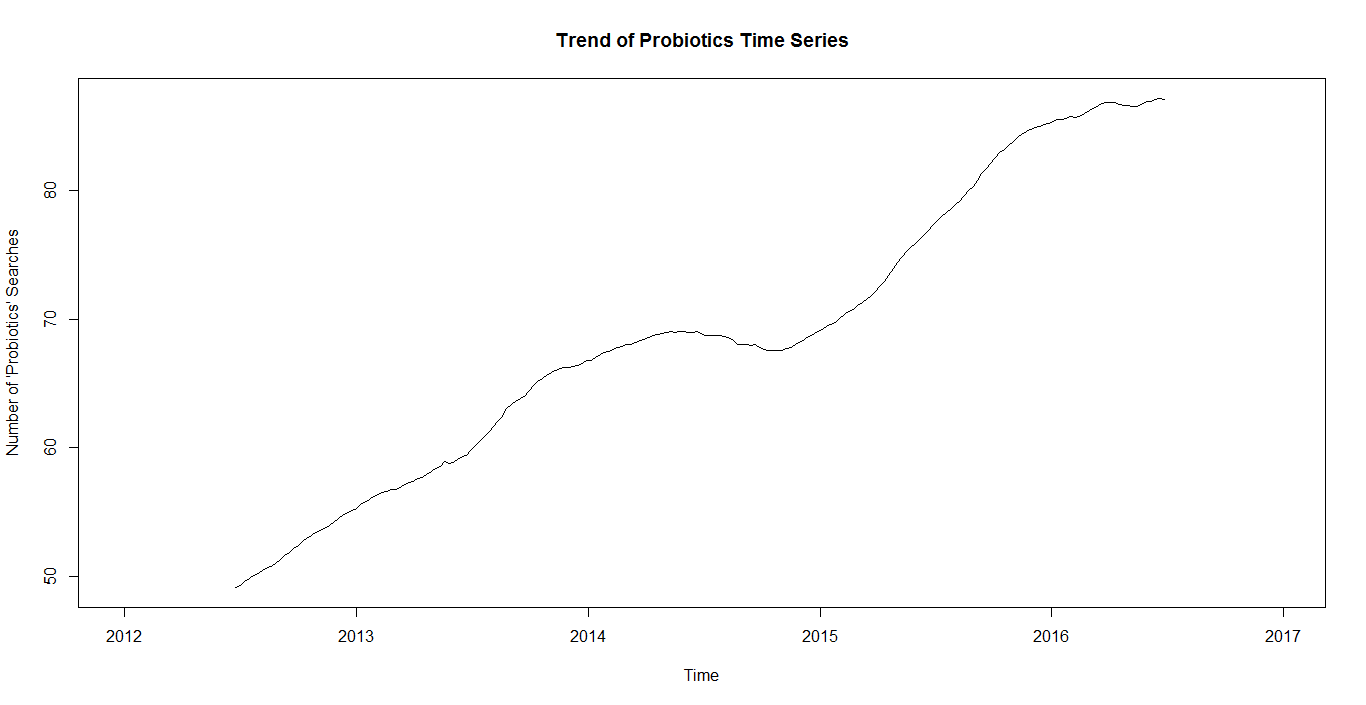
It appears we do not need to transform our data in order to stabilize variance, or for any other reason; we said earlier that the data appears roughly normally distributed with roughly constant variance. The q-q plot below appears to support the idea of normality; the shape is appropriate and the tails at the end aren’t too extreme—they’re quite common among q-q plots.



We use the ‘decompose’ function to estimate and remove the smooth components of our time series—trend and seasonality. We also use the ‘decompose’ function to obtain the rough part of our time series, which is the random fluctuations that are left over after removing trend and seasonality. However, we will attempt to explain how the ‘decompose’ function goes about accomplishing these tasks.

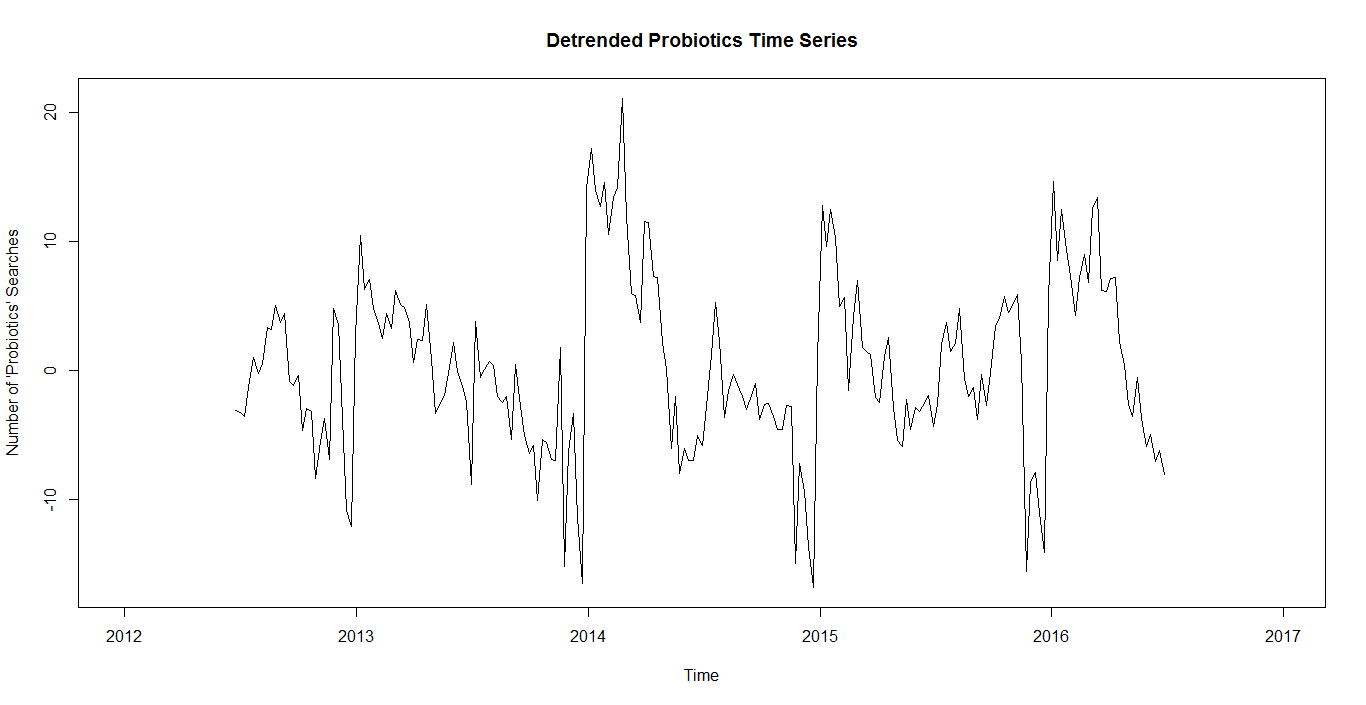
There are a few different methods for estimating the trend component, one of which is smoothing by means of moving averages. The two-sided moving average version of this method is briefly discussed in Section 1.3 of the Lecture Notes. Describing the two-sided moving average in more simple terms, for every data point *t,* we take the average of the surrounding *q* number of data points. This smooths the fluctuations to help show the underlying trend.

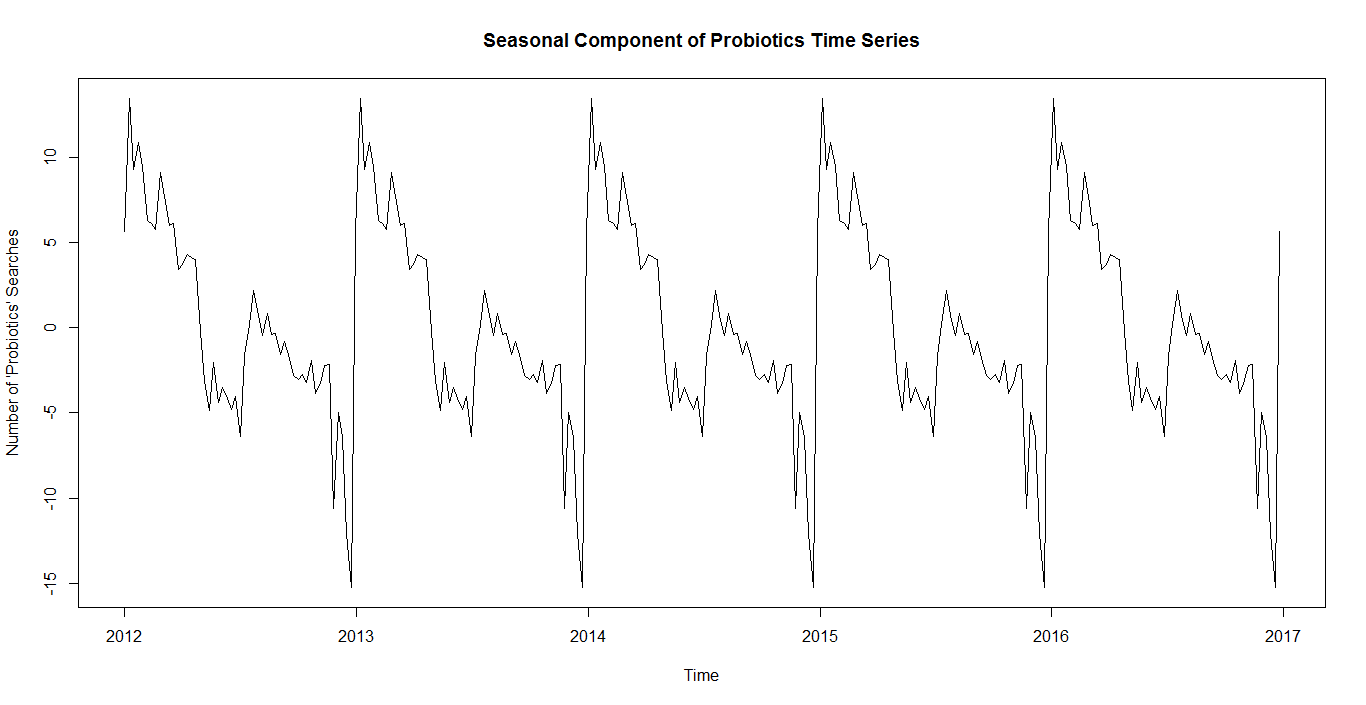
Section 1.3 of the Lecture Notes also states that “there is no possibility of estimating the ﬁrst *q* and last *n* - *q* drift terms due to the two-sided nature of the moving averages.” Since our data is reported weekly, we average the 52 data points around every data point *t*, with 26 being in either direction of our data point *t*. This means we are not able to estimate the first 26 terms in the trend, nor the last 26 terms in the trend. However, this shouldn’t be much of a problem since we still have plenty of data points available to distinguish the trend. A plot of the estimated trend component is pictured below.



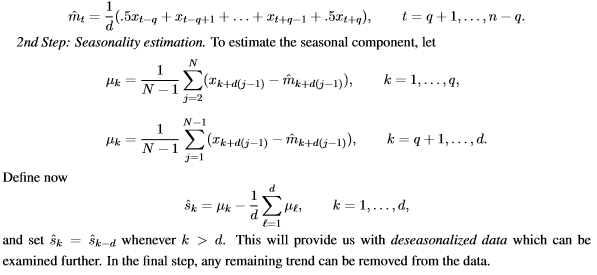
The consistent, upward trend is more easily seen here. It has a bit more curvature than a truly linear pattern, but it appears consistent enough to proceed with our data analysis.

If we subtract our estimated trend from our original time series, we are left with a detrended time series that highlights the underlying seasonality. However, this is not our final display of the estimated seasonality—mostly because it still includes the random component. This can be seen if we compare our detrended estimate with the ‘decompose’ function’s seasonality estimate (pictured below.) The detrended estimate is on top, and the seasonality from the ‘decompose’ function is below it.

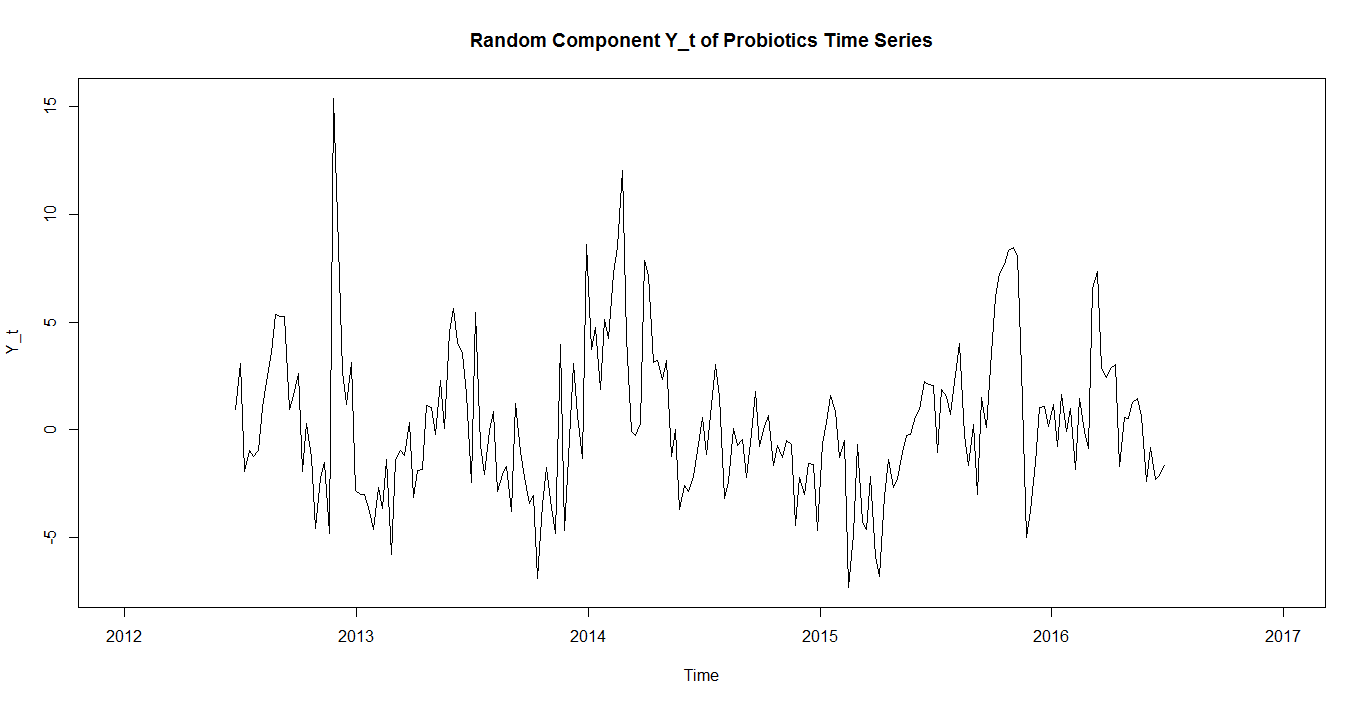




The spike in 2014 appears to be a bit larger than others, but it doesn’t look significantly larger. To transition from the detrended estimate to the ‘decompose’ function’s estimate, the ‘decompose’ function performs some mathematical computations that average the seasonality and spread it across the entire dataset. This seasonality calculation is for moving average estimation. Roughly explained, one creates a data matrix. In this data matrix, each column is a cycle (in this case roughly 52 weeks or data points). We then calculate the mean for every row. Each data entry is then changed to the data entry minus the row mean. Hopefully that is explained somewhat correctly. The formulas describing this process can be seen below (they are given in Section 1.4 of the Lecture Notes.)

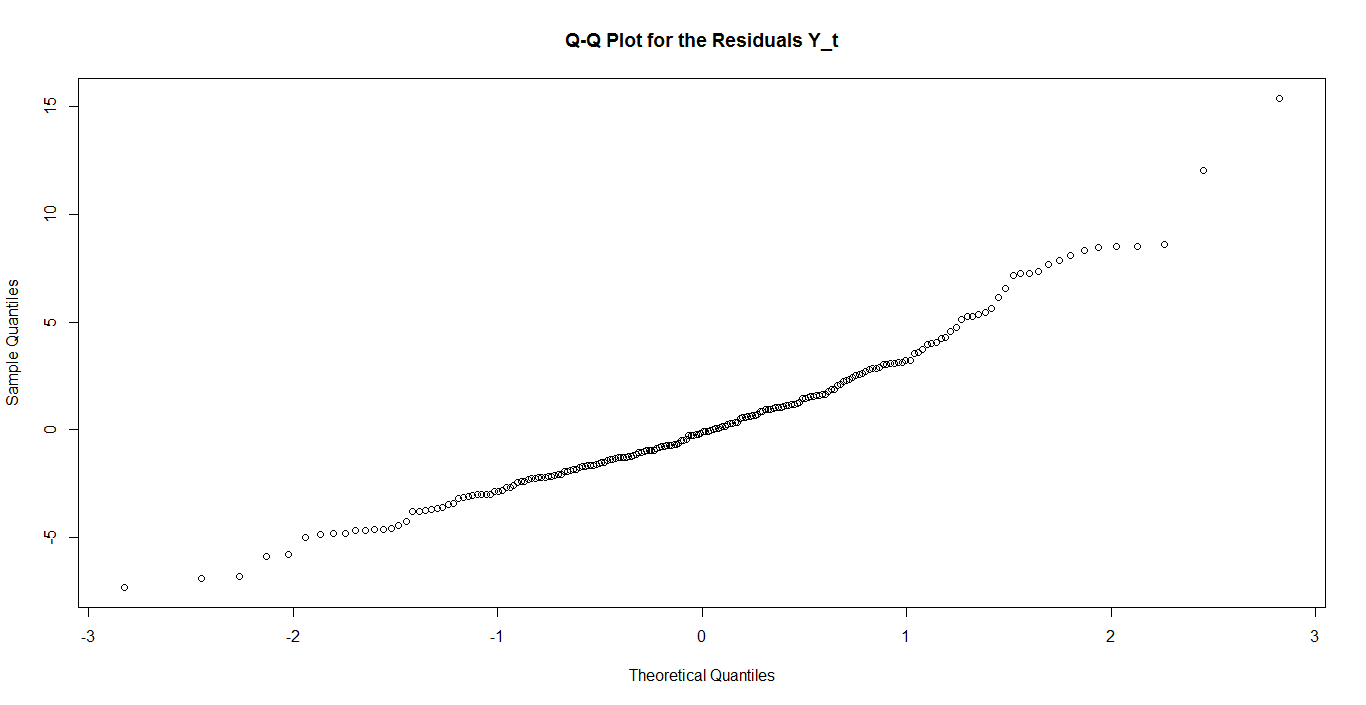


The rough part, or random part, of our time series can then be retrieved by subtracting the smooth components (the trend and seasonality) from our time series. This can be seen below, retrieved from the ‘decompose’ function.

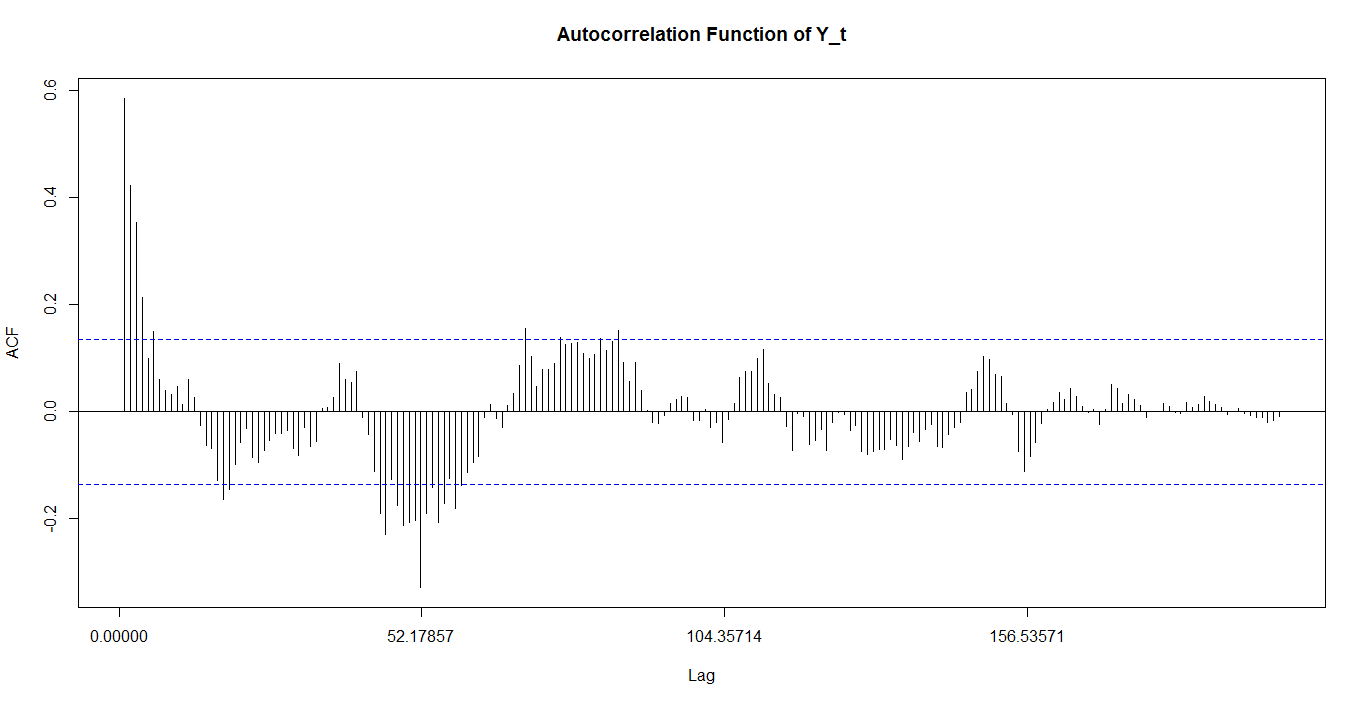


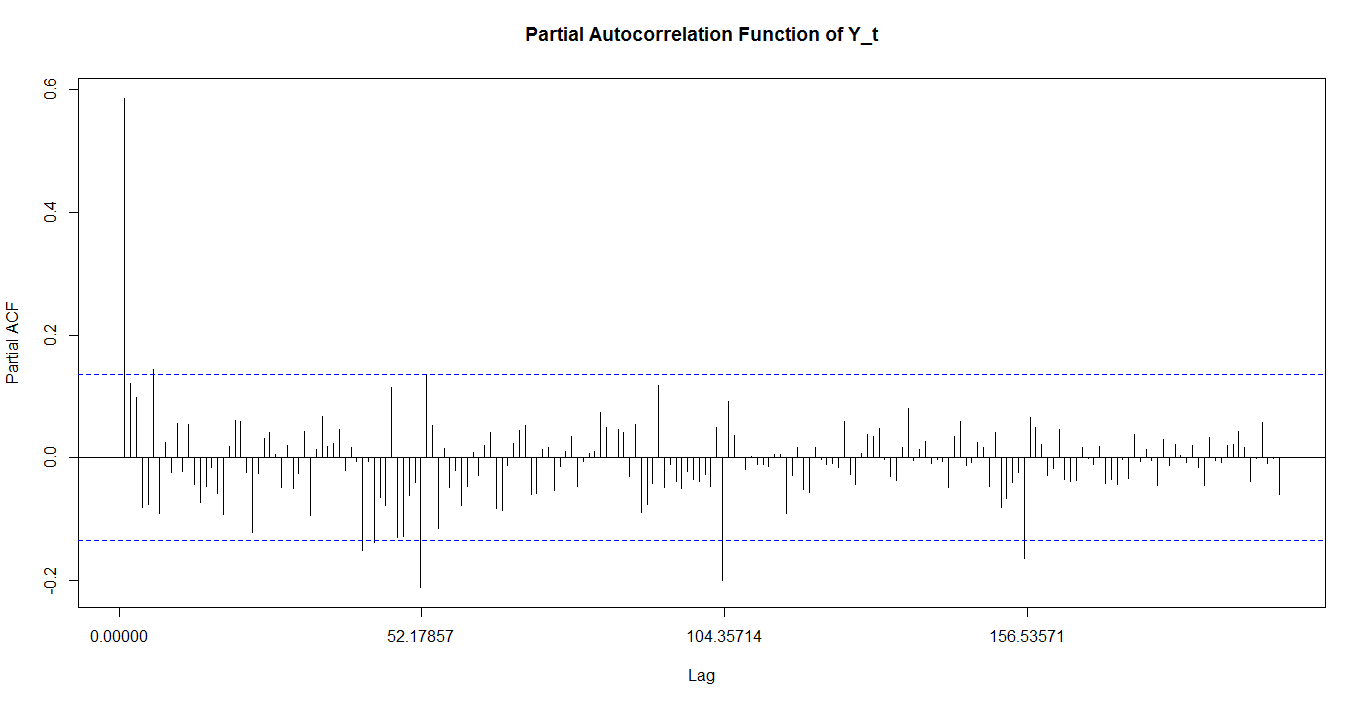
Our next goal is to determine if Y-hat\_t is weakly stationary, which we refer to as just ‘stationary’. We will also check for dependence. To be stationary, the mean and variance of the time series must not depend on time *t*. In other words, the mean and variance should stay constant throughout the entire time series. It’s a bit difficult to tell whether there is still underlying seasonality, as there seems to be a spike around the beginning of every year. However, that doesn’t appear too obvious or noticeable; the variance appears to be relatively constant throughout. The mean also appears relatively constant. Thus, we conclude that this time series of the random components is stationary.

We can further analyze the plot of residuals above with a q-q plot to check for normality. This plot is pictured below, and supports the idea of normality. The tails have a couple points quite a bit separated from the rest, but they don’t seem significantly far away.



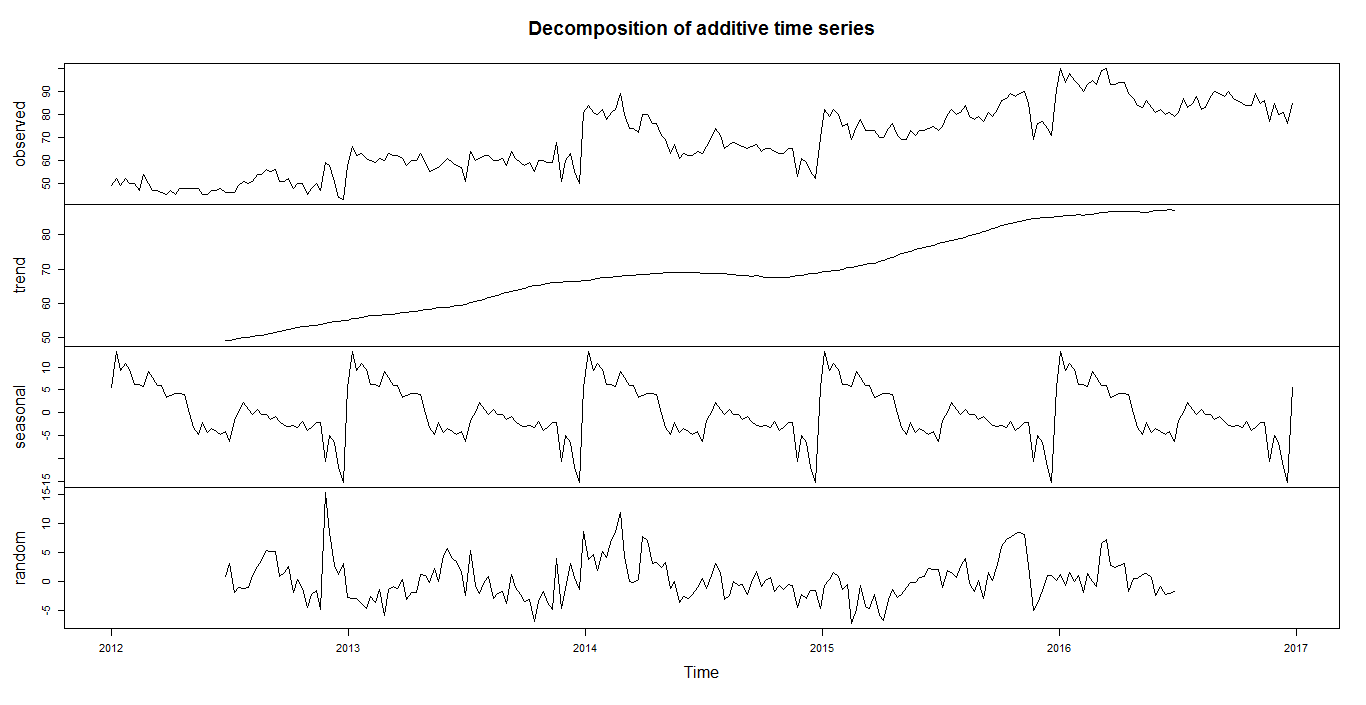
We can double check this conclusion of stationarity by looking at a plot of the autocorrelation (ACF) of the random component. We want to see that the values decrease to 0 as the lag increases, and we do see that. However, the decrease to 0 does not appear to happen quickly, which might be a sign of dependence, which is not what we want; I am unsure of how to assess the significance of how quickly the ACF model is supposed to approach 0. The spike of significance around a lag of 52, which is roughly the beginning of the next year, is a bit worrying. However, it doesn’t seem too significant, since the values continue to decrease as lag increases. Perhaps there is another way to assess the significance of that 52-lag, but I am unsure of how to go about doing that. The PACF tells a similar story.





Another way to check for stationarity is using the Ljung-Box test. It will check the first 20 lags to check for significance indicating dependence among the residuals. It reports a p-value, and if that p-value is below roughly 0.05, then we can say that the rough component appears stationary. We report a p-value of 2.2e-16, indicating quite strongly that the random component residuals appear stationary.

As a way to consolidate and view the entire decomposition of our time series, below is a plot of the entire ‘decompose’ function, reiterating what we have found thus far.



**Model Selection**

Our next step is to fit stationary ARMA(p, q) models to the residuals by analyzing ACF and PACF plots. ARMA models are a combination of autoregressive models and moving average models—p and q are the orders for each, respectively. We should use those plots to support what we decide to use as our ARMA model, and our choices of p and q. There is an R function called ‘auto.arima()’ which can pick a model for us, although it is an ARIMA(p, d, q) model and might be more general than what we’ve learned in class—thus a bit difficult to explain in-depth. The d parameter signifies the amount of differencing has been done to the data. Basically, ‘auto.arima()’ tries all potential combinations of p, d, and q, and then reports the model found with the lowest AIC; I could guess that the model probably averages the other predictors AICc and BIC with AIC, but it would only be a guess.

The model chosen seems a bit more complicated than what we have covered in class, and I am unsure of how to properly and accurately explain it. Thus, here is the R output for the model, in case the reader is better able to understand it than I am:

Series: Y\_t

ARIMA(1,0,0)(1,0,0)[52] with zero mean

Coefficients:

ar1 sar1

0.5832 -0.4408

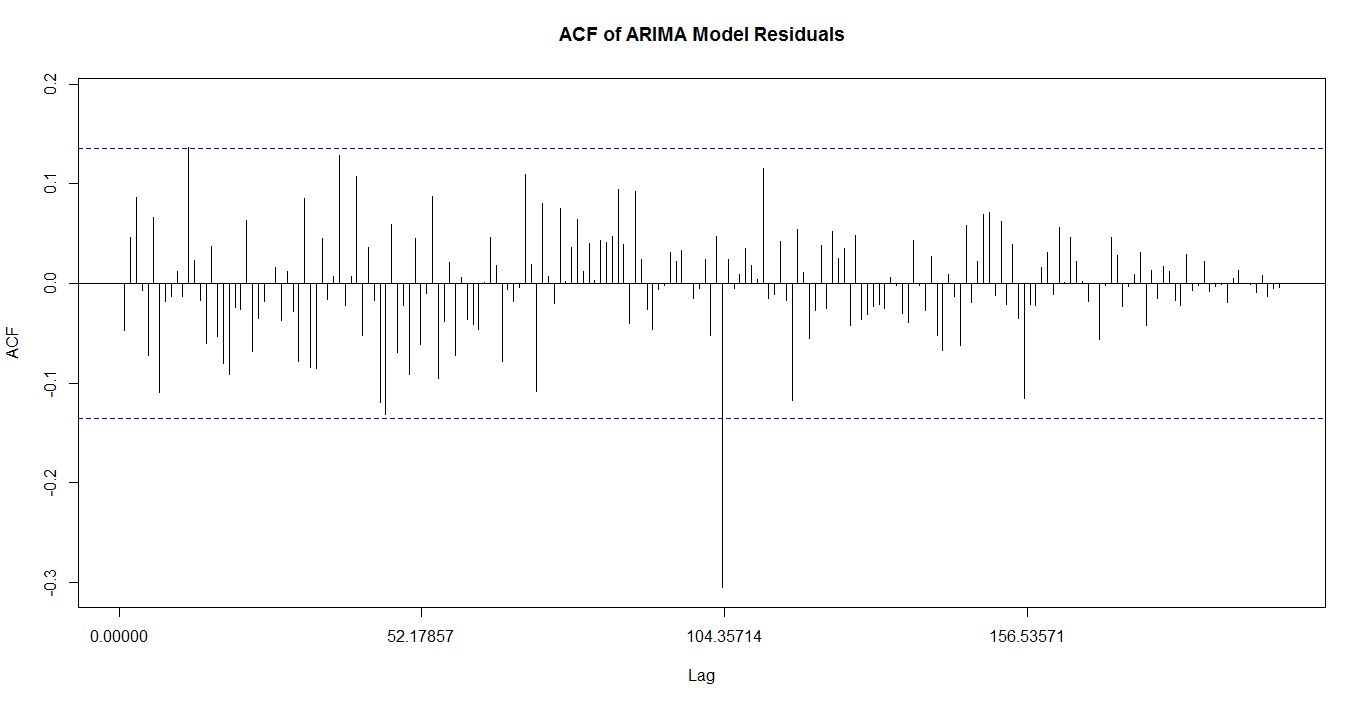
s.e. 0.0557 0.0689

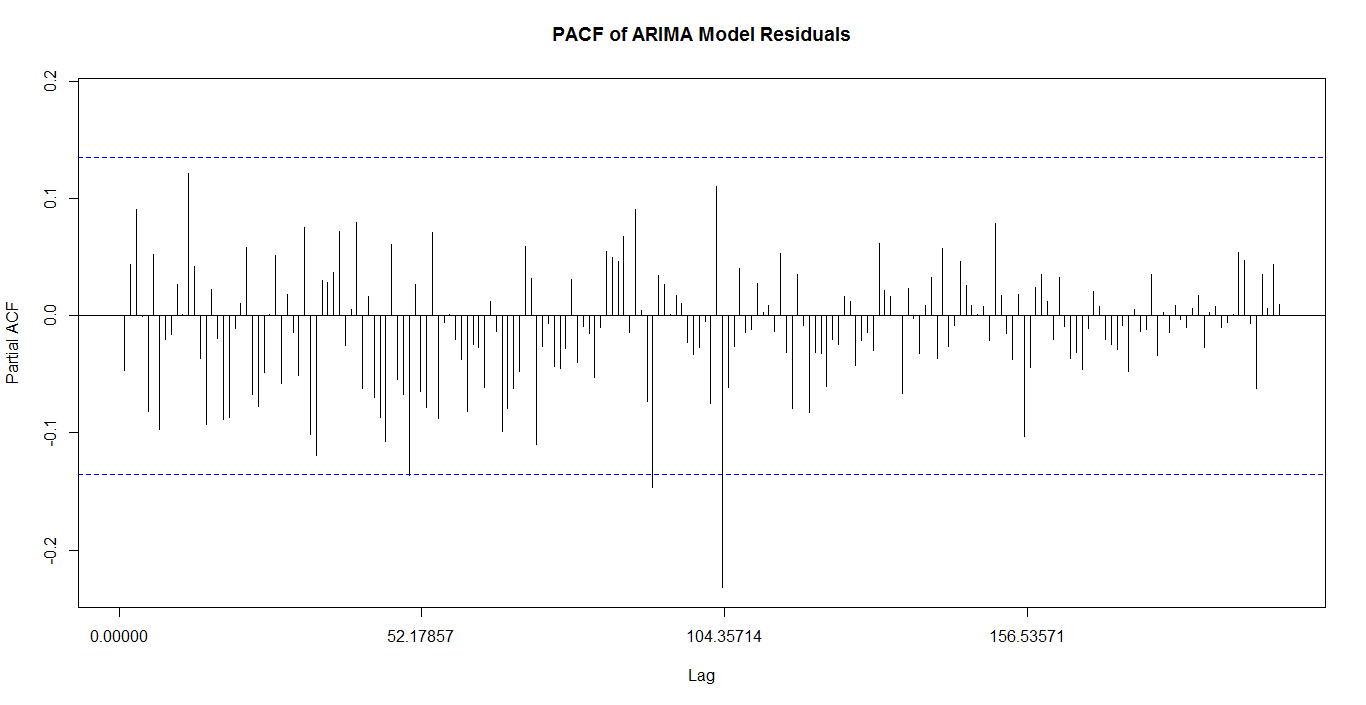
sigma^2 estimated as 5.32: log likelihood=-501.33

AIC=1008.66 AICc=1008.76 BIC=1019.36

The model appears to have two coefficients, 0.5832 and -0.4408. I think the 0.5832 is a *p* parameter. I am unsure if the -0.4408 is a p or q parameter. It looks like a combination of two AR(1) models. The negative parameter makes sense because our ACF and PACF show the values bouncing from negative to positive at different lags. It seems a model with an AIC of 1008.66 is the lowest AIC we can find.

After the model is selected, we look at the ARIMA model’s residuals to make sure that they look like a white noise process, as this is our goal. We look ACF and PACF of the model’s residuals to check this, and also conduct the Ljung-Box Test again.



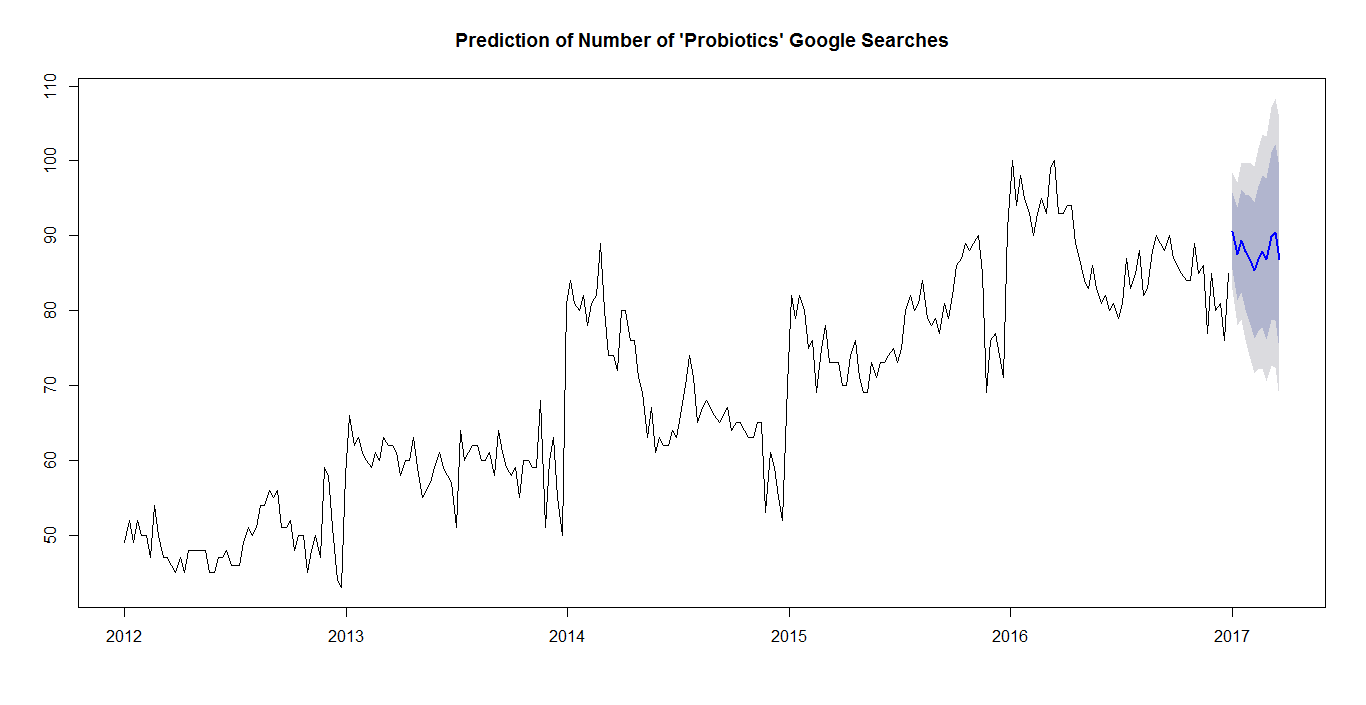


Values below the blue dotted-line indicate that they are most likely not significant, i.e. that they are not dependent. Most of the values for the ACF and PACF fall under the blue dotted-line, but they do not approach 0 very quickly. This is again a sign that there may be a problem: there may still be underlying seasonality that has been unaccounted for, or perhaps some other unsolved issue with dependence amidst the data. Furthermore, the significant lag at 104, or roughly two years, is odd and might be another indicator of this problem. We take note of these behaviors and will continue to proceed with caution; we also are aware that the stationarity may not be as clear as we would hope it to be.

For the first 20 lags, the Ljung-Box test for the model residuals reports a p-value of 0.6442, which is well above 0.05. This indicates that there is likely some significant correlation at these lags. This seems to also indicate the potential problems briefly mentioned above. Again, we take note of these potential problems in our conclusion and discussion.

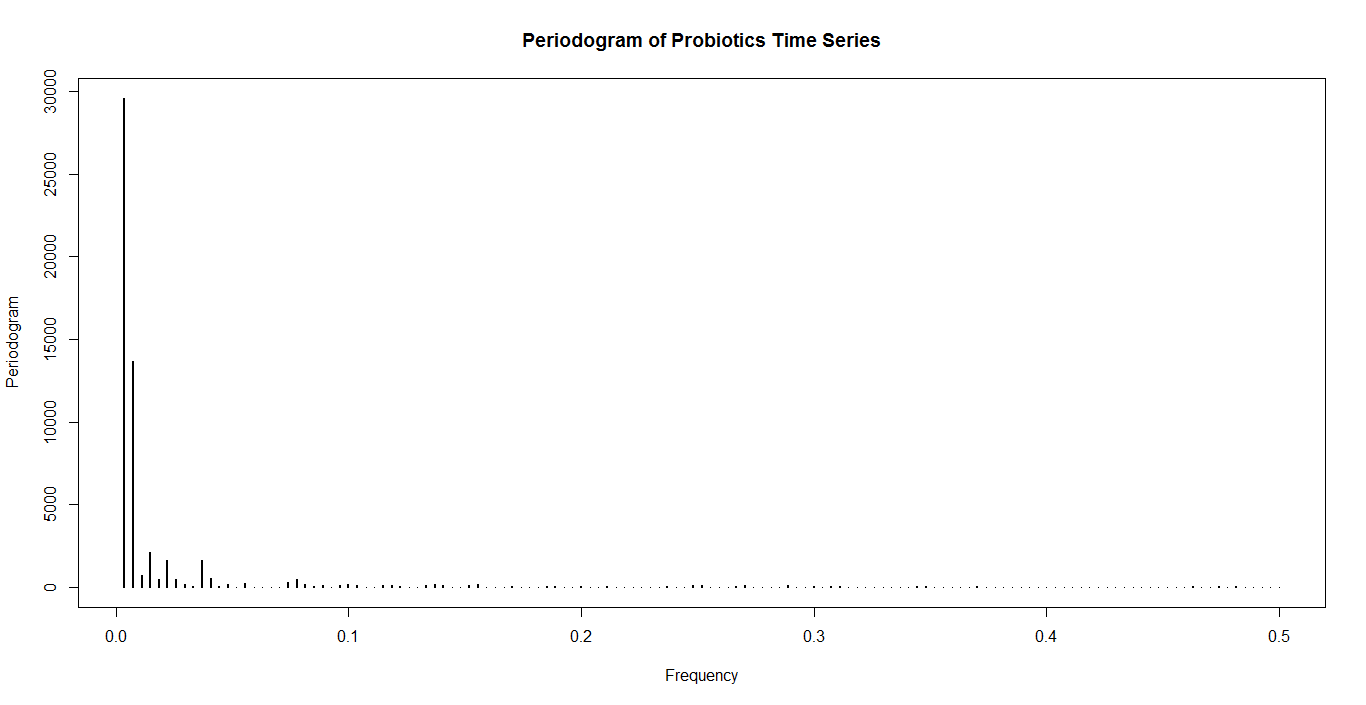
**Prediction**

To predict future values, we use the ‘auto.arima()’ function to fit a model of our time series that includes both the rough component (random) and the smooth components (trend and seasonality). We then use the ‘forecast()’ function on this model to predict future values, and we chose to predict the next 12 data points—which is roughly three months. A plot of these predictions is provided below.



The shaded regions are confidence levels for the prediction interval, with the darker region being an 80% confidence interval and the lighter region being a 95% confidence interval. The dark line is a point estimate for the prediction. The predictions appear to emphasize our idea of an upward trend.

We briefly run the ‘periodogram’ function from the TSA package on our original time series to create a periodogram. When doing a spectral analysis, we attempt to break down the time series into a combination of sin and cosine functions. This periodogram helps to show which frequencies are the most significant. It is pictured below. It seems to support the idea that a significant frequency is very close to 0.



**Discussion**

Our findings were not perfect, but there is still useful information to takeaway. We found a bit of evidence to support the appearance of an upward trend of searches for ‘probiotics.’ However, the linearity and strength of this trend can be questioned due to the curvature of the trend seen when decomposed. Our estimate of the seasonal component looked promising at first, but ACF’s and PACF’s led us to believe that there may be more underlying seasonality or dependence among the data. This is because the values did not decrease to 0 quickly as the lag increased. There were also a couple of significant lags very far out. Perhaps the seasonality simply isn’t very strong, and is therefore difficult to estimate. Similarly, estimated random component of our time series seemed to be relatively similar to that of a white noise process, but the ACF and PACF made us doubt that as well, and for the same reasons as it did for the estimated seasonality. Again, our ARIMA model had the same problems with the ACF and PACF. We are unsure as to how we would go about improving these estimates, but at least they are not complete rubbish. Further research could lead us to improved results. Using the ‘forecast’ function with an ARIMA model of our undecomposed time series, we were able to make a prediction interval of 12 future values that appeared very reasonable. These predictions also agreed with our notion of an upward trend, reassuring us that there is some legitimacy and use for our findings and analysis. After all, some would argue that perfect models and data analyses don’t truly exist anyway.

**Conclusion**

We have shown that there is a growing interest in probiotics, which can be seen in the growing number of Google searches for the term ‘probiotics’ over the years. The number of searches may not seem significant, but the data analysis shows that this upward trend seems likely to continue. Our forecasted future values support the idea that this trend appears it will continue, and forecasting later can continue to provide information on where the market is heading. Furthermore, Google searches could be considered representative of our population’s interests. To create a profit out of this information, one could invest in probiotics, either by producing them or buying shares in the market of those who do produce probiotics. It’s important to consider that the earlier one gets into a market, the larger their share of that market is likely to be. Investing now could prove to be very beneficial. This idea doesn’t have to stop at probiotics either—there may be a growing trend of interest in personal well-being overall.

There is also a seasonal aspect to this probiotics trend, where the number of searches increases around January 1st of every year. This could help identify when to focus on marketing probiotics, and in turn further increase profits or develop a consumer base.

Probiotics arguably offer potential health benefits for humanity at large. There is an increasing number of studies supporting their benefits. If companies can promote further growth of this trend, then they could potentially help to create a healthier population. A healthier population is a more productive population. And if what you care about is the profits, then consider that this promotion of probiotics and caring about people could help promote the company’s public image if marketed correctly.

**Appendix**

# STA 137 Project

# Probiotics google search trend

# Linear Time Series Analysis

library(astsa)

library(forecast)

# Read in the data

library(readr)

ProBioData <- read\_csv("C:/Users/Andre/Desktop/Classes/STA137/project/FinalProBioData.csv",

skip = 2)

View(ProBioData)

# Get a list of only the number of searches; dates (the first column) not included

ProBioData[,2]

ProBioNumbers = ProBioData[,2]

typeof(ProBioNumbers)

# unlist the 'list' type to proceed with computation -> coerce to type 'integer';

# Not sure why this is necessary, must be some coding semantics I don't understand

# Get errors later if in 'list' form

ProBioNumbers = unlist(ProBioNumbers)

typeof(ProBioNumbers)

# 5 years \* 52 weeks/year = 260

# It somehow adds 1 observation, probably because 2012 and 2016 were leap years, with 366 days

# Sample size is 261

length(ProBioNumbers)

# Plot of the data observations (i.e. the number of searches every week)

par(mfrow = c(1,1))

plot(ProBioNumbers, xlab = "Week", ylab = "Number of Searches",

main = "Google 'probiotic' Searches")

#### Summary Statistics of the data

summary(ProBioNumbers)

var(ProBioNumbers)

# Turn the list into a TimeSeries object

###################### How to deal with leap years?

# Set frequency to 365.25/7, rather than 52 -> I'm not sure why this works, but it does.

ProBioTS = ts(ProBioNumbers, frequency = 365.25/7, start = c(2012, 1, 1))

# Plot the data in a time series format

plot.ts(ProBioTS, ylab = "Number of Searches", main = "Time Series of Google 'probiotics' Searches")

# QQ-plot looks roughly normally distributed

par(mfrow = c(1,1))

qqnorm(ProBioNumbers)

# Comparing with log transformation, we see little difference

#logP = log(ProBioNumbers)

#qqnorm(logP)

#plot(logP)

#plot(ProBioNumbers)

#### Estimating Trend component

# I'm just going to use decompose

dec = decompose(ProBioTS)

plot(dec)

plot(dec$trend, main = "Trend of Probiotics Time Series",

ylab = "Number of 'Probiotics' Searches")

#### Estimating Seasonality (seasonal component)

plot(ProBioTS)

plot(ProBioTS - dec$trend, main = "Detrended Probiotics Time Series",

ylab = "Number of 'Probiotics' Searches")

# the spike was a bit larger in 2014, but it doesn't look significantly larger

plot(dec$seasonal, main = "Seasonal Component of Probiotics Time Series",

ylab = "Number of 'Probiotics' Searches")

#### Assess the random component

smooth = dec$seasonal + dec$trend

Y\_t = ProBioNumbers-smooth

plot(dec$random) # same as the plot of Y\_t below

ts.plot(Y\_t, main = "Random Component Y\_t of Probiotics Time Series")

Y\_t = dec$random

plot(dec$random)

plot(dec)

# q-q plot of the residuals supports normality

qqnorm(Y\_t, main = "Q-Q Plot for the Residuals Y\_t")

# Acf() and Pacf() are from forecast package and shows larger lags

# They also automatically accomodate for NA values

# # The NA values come about because of the NA's in our moving average trend estimation.

#acf() and pacf() are not from forecast and it shows very small lags (smaller than 1.)

Acf(dec$random, lag.max = 200)

Acf(Y\_t, lag.max = 200, main = "Autocorrelation Function of Random Component")

Pacf(Y\_t, lag.max = 200, main = "Partial Autocorrelation Function of Y\_t")

Box.test(Y\_t, lag = 20, type="Ljung-Box")

#### Model

fit = auto.arima(Y\_t)

summary(fit)

fit

plot(fit)

Acf(fit$residuals, lag.max = 200, main = "ACF of ARIMA Model Residuals")

Pacf(fit$residuals, main = "PACF of ARIMA Model Residuals", lag.max = 200)

Box.test(fit$residuals, lag = 20, type="Ljung-Box")

#### Prediction

fit.log = auto.arima(ProBioTS)

plot(forecast(fit.log, h=12), main="Prediction of Number of 'Probiotics' Google Searches")

# Spectral Analysis (just a periodogram)

#install.packages("TSA")

library(TSA)

per = periodogram(ProBioTS, main = "Periodogram of Probiotics Time Series")

**References**

Mostly used as referential reading

Mainly used to find out that 365.25/7 can accommodate leap years in a time series:

<http://manishbarnwal.com/blog/2017/05/03/time_series_and_forecasting_using_R/>

<https://anomaly.io/seasonal-trend-decomposition-in-r/>

<https://onlinecourses.science.psu.edu/stat510/?q=book/export/html/41>

<https://rstudio-pubs-static.s3.amazonaws.com/269797_8a8da7911da74ea38e2708a954aeaf29.html>