Understanding Home Remodeling Seasons Through Time Series Modeling and Frequency Analysis of Google Trends Data

Laura Kim

March 16, 2017

1 Introduction

In this report, we plan on outlining the process of understanding the home remodeling trends in our country by studying the patterns of Google Search queries that include the term General Contractor. Searches made on Google may indicate what people are curious about. For example, the sudden spike in searches over the word "immigration" in Figure 1, may indicate an acceleration of interestest in immigration policies when the spikes occured.

The dataset for this project was collected from the Google Trends website and the paper outlines the necessary time series analysis performed on the dataset. We were able to create a time series model of the data and find an annual trend which coincided with actual realestate market cycles.

2 Google Trends Data Set

Google Trends shows the Google search index for the searches entered into the Google Search Bar. The Google Search index will be described later, but we can consider it as the change in interest or search traffic over a search term.

Google Trends Data may be retrieved from the website https://trends.google.com/trends/?cat=. We enter the search term that we would like to study and also enter the date range and narrow the geographic origins of searches. The website then obtains a time series data set which shows the Google search index over the range of specified time. The dataset may be downloaded in various formats including .csv, which is the format used for this project.

For the analysis, the search term General Contractor was used and the date range was set to 5 years, from January 1, 2012 to January 1, 2017. We also limited the dataset to searches in the United States.

2.1 Google Trends Applications

The reason we chose Google Trends data to analyze remodeling cycles is because Google Searches have been used to monitor marketing trends, general interests such as Oscar Awards, flu outbreaks (although there are controversies on how effective it is).

The idea behind this research is that people tend to query what is on their mind. For instance, one would search for flu remedies at the onset of flu symptoms. This lead to creating an entirely independent platform for the sole purpose of keeping track of flu outbreaks. Since CDC data has a 2 week lag, google search trends ended up providing a 2 week lead time in the prediction of flu. In short, google trends could be a good way to guage the public interest in a topic. It is a window into the collective mind set of google searchers. Given that one can isolate searches with resolution up to city level, local and regional trends can be studied as well as seasonal trends such as fashion. It is an invaluable tool for marketers in the modern data driven economy.

2.2 How is a Google Search Index Calculated?

Google Trends manipulates the actual search counts for many reasons. The search data is adjusted to make comparisons between days easier. Each data observation is divided by the total number of searches within the geographic region and date range selected and then scaled to be between 0 to 100. This data processing is done to make it easier to compare relative popularity of the search terms rather than assess the raw search volume. The raw search volume would generally be highest in cities with denser populations and is not a good indicator of relative change in interest. Also, the number of people on Google changes yearly. For example, there are more people on Google now than there were 10 years ago. Therefore it is important to have divide by the number of searches in time so that comparisons between years can be properly made.

For this project, we were interested in relative search interest over time. We were not making comparisons across location so this data manipulation did not necessarily help our analysis. It also did not harm our analysis because we were interested in analyzing general interest in a topic (as manifested by search requests) over time.

3 Stationarity and Detrending

The top panel in Figure 1 shows the the time series plot of the google search index for the term general contractor. It looks almost stationary but there is a slight but noticeable upward trend. The Augmented Dicky Fuller Test for stationarity had a p-value of 0.01, so we rejected the null hypothesis that it had a unit root and accepted the alternative hypothesis, that it is a stationary sequence. However, despite the result of the ADF Test, we decided to find the trend. After all, if the trend did not really exist, the series would not change much after detrending. The benefit of finding a cleaner model far outweighs the small trouble of detrended. The top panel of Figure 1 shows the trend line, which has an

intercept of -3181.304 and a slope of 1.616, both of which turned out to be highly significant in the model. The residuals after removing the trend is shown in the bottom panel of the Figure 1.

4 ACF and PACF

The sample autocorrelation and partial autocorrelation of the detrended series appears to show some periodicity. Recall that the ACF provides a lot of information about the order of a MA process but it does not tell a lot about the ARMA and AR process alone. This is why we also have to consider the PACF for the AR models.

For the series we are studying, the sample PACF appears to be cutting off after 2 lags and the ACF looks like it is tailing off. This may be an AR(2) model. However, if the ACF is cutting off after lag 6 and the PACF is tailing, the model is a MA(6). It could also be an ARMA(p, q) if the ACF and PACF are tailing off. Figure 9 displays the ACF and the PACF and the reader can consider what the pattern means. All we could tell from the ACF and PACF is that there is some seasonality in the data.

In the next section, we look at the frequency domain to determine the seasonality of the time series. We then attempt to remove the seasonality before fitting the model.

5 Frequency Analysis

To identify any seasonality or periodicity in the series, we first created a periodogram. According to Property 4.1 of the text, any stationary series can be represented in sines and cosines. The book says that trends may add low frequency components into the spectrum so we used the detrended series with a mean of 0 for the frequency analysis. The spectrum appears to have one main peak at a frequency of 1.038462 or a period of 54 weeks. This approximately corresponds to an annual cycle. There were also peaks at frequencies of 2.8888, 2.596154, 0.5192308, and 0.1442308 which had corresponding periods of 18 weeks, 135 weeks, 27 weeks, and 7.5 weeks. The latter two frequencies are harmonics of the fundamental frequency of 54 weeks due to the imperfect nature of the sinuoids. However, we could not find a reason to explain the first two frequencies (18 weeks and 135 weeks). The confidence interval for the main peak is $9.434306 \leq f(\omega) \leq 1374.606$ which is too wide to make a conclusion about the significance of the peak. Figure 3 shows the periodogram with vertical lines identifying peaks that were discussed here.

Next we attempted to compute the average periodogram to smooth out some of the noise. This is a nonparametric spectral estimation because there is no assumption about the parametric form of the spectrum. First, we used the daneill filter with m=1 or L=3. The image is shown in Figure 4. We can see that already, the freqeuncy of the first peak is being averaged out. Figure 5 shows an attempt with a Daniell filter applied twice, both with L=3. It appears to smooth out the periodogram even more. Figure 6 shows the Daniell Filter with m=2 or L=5 applied once. The peaks have changed completely and the

first peak has been smoothed out. Also, the peaks have flattened out making it difficult to determine the frequency at which the peaks occur.

In attempting to bring back some of the frequencies that have leaked or averaged with nearby frequencies, we try to fit a parametric spectral estimator. We fit an AR(p) model to the data and used the AIC to find the best fitting order. The book claims that often, this method will give a spectral density with higher resolution for nearby peaks. The spec.ar function in R picks the best fitting AR model. In our case, it fit an AR(3) which smooths out all peaks. We just know that the high power spectrum are dominating the lower frequencies as shown in Figure 7. The resolution is far worse than the nonparametric methods in this case. To bring back some of the peaks, we try to fit some higher order AR models. AR(33) looks closer to the original periodogram. Although not all of the peaks are present, Figure 8 shows that the main peak for the annual cycle is present as are some of the harmonics. The peaks are not as high or sharp as the raw periodogram but is sharper than some of the nonparametric spectral plots. The parametric method may be a good compromise between the noisy raw periodogram and the flattend averaged periodogram.

6 Time Series Modeling

Before fitting a time series model to the data, it is a good idea to make sure that the trend and seasonality is removed. We are then left with a stationary series that might look like noise. We have thus far used detrended data but have not removed the seasonality. The next sections will discuss removing the seasonality and then model the remaining signal.

6.1 Removing the Seasonality

The frequency domain analysis revealed a clear yearly cycle for searches of the term general contractor. To remove this yearly pattern we subtracted seasonal means from the data. The resulting plot is shown in Figure 11. The plot is centered around 0 and the seasonality is much decreased, even if there still appears to be a slight seasonal pattern.

The sample ACF and PACF of the detrended and deseasonalized data appears to look a lot more stationary than just the detrended series. It appears to look like an AR(1) or a MA(1) but it is unclear what will fit the model best. In the next sections, we disuss how we attempted to fit various models to this data and try to forcast a few future points using the best fitting model.

6.2 Time Series Models

We fitted the detrended and deseasonalized data to a time series model. Fitting all possible models for ARMA(p, q) from values of p and q between 0 and 20, we found that ARMA(2, 3) had the lowest AIC of 1736.427. Other potentially well fitting models were ARMA(1, 2) with AIC 1736.427, MA(1) with AIC 1741.238, AR(6) with AIC 1740.320 and AR(1) with

AIC 1740.238. From looking at the residuals plots, there was not a huge difference between the models except in the QQplots. The residuals for all models appeared to look like stationary white noise except for few potential outliers. There were not significant correlations between the lags and the ljung-box statistics all had high p-values. Choosing the best model came down to picking the model with the most normally distributed residuals as shown by the QQplot and the model with easiest interpretation.

After much consideration, we decided to use the AR(1) model for simplicity and the residual properties in the QQ plot. For some reason, the residuals from all of these models appeared to be heavy tailed. AR(1) seemed to be the least heavy tailed so we chose to use this model for the forecast.

7 Forecasting

For forecasting, we used the AR(1) model of the residuals $x_t = .4104x_{t-1} - 0.0311$. First, we used the forecast function in R and forecasted 2 months ahead. However, the result was not very accurate and the standard error bars too wide. Also, the outcome was not very interesting because the prediction started to plateau after a couple of points. This phenomenon is due to that fact that in long range forecasts, the forecast settles quickly to the mean. In figure 13, we show a plot of the forecast using this method.

Next, we computed the forecast by hand but used the one step point forecast technique by plugging in observations from the previous time stamp. Because we deseasonalized and detrended the data, we added the seasonality and trend back in to get the final forecast. The plot is shown in the Figure 14 and appears to be quite accurate and follows the real data closely. The forecast is shown in red and the actual data is shown in blue. The test data was: 84, 73, 78, 84, 91, 82, 90, 77. The forecast was: 82.85246, 77.32508, 76.36506, 85.41448, 89.07431, 86.96773, 88.83891, 77.29073.

8 The Google Trends Data Confusion and Conclusion

For the presentation, 5 years worth of data was collected from the Google Trends website and analyzed and modeled in a similar way as outlined above. However, the data for forecasting was collected later. To be specific, 5 years and 2 months worth of data was collected from the same website and then the last 2 months of data was subsetted to be used for analysis.

During the forecasting portion of the analysis, it turned out to be hard to calculate very accurate predictions. Despite formulating very good models with nearly ideal residual properties that seemed to have the characteristics of white noise with zero lag and perfect normal distribution, it was difficult to obtain results that resembled the real data as we did for this report in the one step ahead prediction.

Recall that the data is normalized and scaled by date range and location. Since we added 2 months worth of extra data, the entire dataset changed. The yearly cycle remained the

same but the ARMA model changed because the data was renormalized with the extra months. Although Google does not exactly state how normalization is done, we suppose that if there were two weeks worth of data, with 2 searches one week and 10 the next, the numbers would be normalized and scaled to 20 and 100. If another week of 3 searches were added, we would get normalized and scaled values of 20, 100 and 30. However, if we have a number that is larger than 10, lets say 20 on the third week, then the normalization and scaling changes the original search values to 10, 50, and 100 for the three weeks.

Google Trends explains their motivation behind the normalization and scaling on this website. $https://medium.com/google-news-lab/what-is-google-trends-data-and-what-does-it-mean-b48f07342ee8\#.8h3yx2gc1$

Google explains it best so we quote here,"That normalization is really important: the number of people searching on Google changes constantlyin 2004 search volume was much smaller than it is today, so raw search numbers wouldnt give you any way to compare searches then and now. By normalizing our data, we can make deeper insights: comparing different dates, different countries or different cities. The context of our numbers also matters. We index our data to 100, where 100 is the maximum search interest for the time and location selected. That means that if we look at search interest in the 2016 elections since the start of 2012, well see that March 2016 had the highest search interest, with a value of 100. "

In conclusion, 5 years of google trends data for the search term general contractor was downloaded from the google trends website. The AR(1) model appeared to fit the detrended and deseasonalized data well. Detrending removed the line $y_t = -3181.304 + 1.616 * t$ from the raw data and then we removed a yearly seasonality.

The annual cycle reveals that the number of searches for contractors are genrally low in the winter and high during the summer. This pattern is well explained by the real estate market where home sales are stronger in the summer months than the winter months. It seems likely that contractors are hired to fix home either before or after homes are bought and sold during the summer months.

After the data manipulation, the data appeared stationary and we fitted many models and chose the AR(1) model for easy interpretation and relative low AIC (but not the lowest). The one step ahead forecast of the AR(1) model seemed to follow the data well.

9 Code

```
# This is with a new data set that covers 1/2012 to 1/2017 with extra data points

setwd("C:/Users/Laura/Dropbox/classes/stat221 - time series/project")
load("tsa3.rda")
data.google <- read.csv("multiTimeline_la.csv", header = TRUE, sep = ",",strip.white = T

dat <- data.google[1:262,]
dat_w <- data.google
```

```
#test of stationarity: adf
#augmented dickey-fuller test
library(tseries)
adf.test(dat[,2], k=6)

# turn "training" data into a time series

library(lubridate)
dat_ts <-ts(dat$General,
            freq=364/7,
            start = decimal_date(ymd("2012-01-01")),
            end = decimal_date(ymd("2017-01-01"))
)


# time series of entire dataset
dat_all_ts <-ts(dat_w$General,
                freq=52,
                start = 2012.000)


# augmented dickey fuller test p value 0.01
adf.test(dat_ts, k= trunc((length(dat_ts)-1)^(1/3)))


library(forecast)
# only training set
t<-time(dat_ts)
fit_ts = lm(dat_ts ~ t)

# whole data set
tw<-time(dat_all_ts)
fit_tsw = lm(dat_all_ts ~ tw)

a <- stl(dat_ts, s.window="periodic", t.window =10)

fitted_ts <-ts(fit_ts$fitted.values,
               freq=364/7,
               start = decimal_date(ymd("2012-01-01")),
               end = decimal_date(ymd("2017-01-01"))
)

fitted_tsw <-ts(fit_tsw$fitted.values,
               freq=364/7,
```

```
               start = decimal_date(ymd("2012-01-01"))
)


# detrended plot
dev.new()
par(mfrow=c(2,1))
plot.ts(dat_ts,ylab="Google Index", xlab="Time", lwd=2, col='skyblue3', ylim=c(40, 100))
lines(fitted_ts ,col="red3", lwd=2)
plot(t, fit_ts$residuals, ylab="Residuals",type='l',xlab="Time")

# create time series of detrended series
detrended_ts = resid(fit_ts)
detrended = ts(detrended_ts,
               freq=364/7,
               start = decimal_date(ymd("2012-01-01")),
               end = decimal_date(ymd("2017-01-01"))
)

# detrended time series including test points
detrended_tsw = resid(fit_tsw)
detrendedw = ts(detrended_tsw,
               freq=364/7,
               start = decimal_date(ymd("2012-01-01")),
               end = decimal_date(ymd("2017-01-01"))
)


library(data.table)
# creating data table
data = data.table(detrended = as.numeric(detrended),index1 = seq(1,54,1))
data_pred = data.table()

# obtaining seasonal components
seasonality1 = data[,mean(detrended),by = index1]
data_merged = merge(data,seasonality1, all.x = TRUE,by = 'index1', sort = 'FALSE')
data_merged = as.data.frame(data_merged)

# data with seasonality removed
residual = data_merged$detrended - data_merged[,3]

## creating data table (for the whole dataset)
#dataw = data.table(detrendedw = as.numeric(detrendedw),index1 = seq(1,54,1))
#data_pred = data.table()
```

```
## obtaining seasonal components
#seasonality12 = dataw[,mean(detrendedw),by = index1]
#data_mergedw = merge(dataw,seasonality12, all.x = TRUE,by = 'index1', sort = 'FALSE')
#data_mergedw = as.data.frame(data_mergedw)

## data with seasonality removed
#residual2 = data_mergedw$detrended - data_mergedw[,3]

# there is a trend with an upward slope of 0.044 which is an increase in the search term
# who knows what that really means because we don't know the total number of searches
# also we have correlated errors here

# acf and pacf
acf2(dat_ts, 100)
acf2(detrended, 100)
acf2(residual)

residual = ts(residual,
              freq=364/7,
              start = decimal_date(ymd("2012-01-01")),
              end = decimal_date(ymd("2017-01-01"))
)

adf.test(residual, k=6)

# AIC  minimum at 1740.507 p = 1or AR(1) or 1740.320 p = 8 AAR(8) and
#1741.238 q = 1 or MA(1) respectively
# This is only for the ar model
n= length(dat[,2])
AIC.ar = rep(0,50) -> AIC.ma -> BIC.ar -> BIC.ma
for (k in 1:50){
  fit.ar = arima(residual, order = c(k, 0, 0))
  fit.ma = arima(residual, order = c(0, 0, k))
  AIC.ar[k] = AIC(fit.ar)
  BIC.ar[k] = AIC(fit.ar, k = log(length(n)))
  AIC.ma[k] = AIC(fit.ma)
  BIC.ma[k] = AIC(fit.ma, k = log(length(n)))

}
IC = cbind(AIC.ar, AIC.ma)
ts.plot(IC, type = "o")

# AIC minimum 1736.427 ARMA(1,2)
# 1732.326 ARMA(2,3)
#
```

```
# the detrended one gets min at AIC 1863.811 ARMA[1,2]
# This is only for the ARMA model
n= length(dat[,2])
AIC = matrix(cbind(rep(0,20), 20), nrow = 20, ncol = 20) -> AICc -> BIC
for (p in 1:20){
  for (q in 1:20){
    fit = arima(residual, order = c(p, 0, q))
    AIC[p,q] = AIC(fit)
    BIC[p,q] =  AIC(fit, k = log(length(n)))

  }
}


# sarima makes a lot of outputs regarding residuals
# if pvalue is high, we fail to reject null
# the null is that there is correlation?
arma_Values = sarima(residual, 2, 0, 3)
sarima(detrended, 1, 0, 2)
arma_raw = sarima(dat_ts, 6, 0, 4)
for_resid = arima(detrended, order=c(6,0,4))
for_resid_ts = ts(resid(for_resid),
                  freq=364/7,
                  start = decimal_date(ymd("2012-01-01"))
)
acf2(for_resid_ts, 50)

# forecasting

# long term forecast
m1=Arima(residual,order=c(1,0,0))
plot(forecast(m1,h=9),shadecols="oldstyle")

# forecasting used in project
t_test <- c(2017.000,2017.019, 2017.038, 2017.058, 2017.077, 2017.096, 2017.115, 2017.13
linear <- -3181.304 + 1.616 * t_test
dat.test <- dat_w[262:270,]$General
season <- seasonality1[1:9, 2]
residual_new <- dat.test-linear
fore_res <- 0.4104*residual_new -0.0311
real_forecast <- linear+season+fore_res


se <- sd(real_forecast)/sqrt(length(real_forecast))
real_forecast_ts <- ts(real_forecast,
```

```
                      freq = 52,
                      start = decimal_date(ymd("2017-01-08")))
test.dat_ts <- ts(dat.test[-1],
                  freq = 52,
                  start = decimal_date(ymd("2017-01-08")))

ts.plot(dat_ts,real_forecast_ts, col = 1:2) # plot of prediction
lines(test.dat_ts, type="p", col = 4)


# periodogram
dat.per = spec.pgram(detrended, taper=0, log="no")
dat.per = spec.pgram(dat_ts, taper=0, log="no")

abline(v=(f1, f3, f3))

# confidence interval is too wide
U = qchisq(.025, 2)
L = qchisq(.975, 2)
2*dat.per$spec[5]/L
2*dat.per$spec[5]/U
2*dat.per$spec[10]/L
2*dat.per$spec[10]/U
2*dat.per$spec[36]/L
2*dat.per$spec[36]/U


# everything else is a harmonic of the 54 week cycle
dat.per$spec[5]
f1 <-dat.per$freq[5] # period of 54 weeks
dat.per$spec[10]
f2 <- dat.per$freq[10] #period of 27 weeks
dat.per$spec[36]
f3 <- dat.per$freq[36] # period of 7.5 weeks

abline(v=c(f1, f2, f3), lty=2)


spaic=spec.ar(detrended, log="no")

# nonparametric
# averaged periodogram: can't seem to get a much smoother spectral density
k = kernel("daniell", 8) # sort of a monthly average L=5, m=2
gt.ave = spec.pgram(detrended, k, taper=0, log="no")
abline(v=c(f1, f2, f3), lty=2)
```

```
# nonparametric
# averaged periodogram with modified daniell kernel
dev.new()
k=kernel("modified.daniell", c(1, 1))
gt.smo=spec.pgram(detrended, k, taper=0, log="no")
abline(v=c(f1, f2, f3), lty=2)

# parametric spectral estimation


gtaic = spec.ar(detrended, log = "no")
spec.ar(detrended, order = 33, log = "no")
abline(v=c(f1), lty=2)

n= length(dat[,2])
AIC.ar = rep(0,50) -> AIC.ma -> BIC.ar -> BIC.ma
for (k in 1:50){
  #fit.ar = arima(dat_ts, order = c(k, 0, 0))
  #fit.ma = arima(dat_ts, order = c(0, 0, k))
  fit.ar = arima(detrended, order = c(k, 0, 0))
  #fit.ma = arima(detrended, order = c(0, 0, k))
  sigma2 = var(fit$resid, na.rm=TRUE)
  AIC.ar[k] = AIC(fit.ar)
  #BIC.ar[k] = AIC(fit.ar, k = log(length(n)))
  #AIC.ma[k] = AIC(fit.ma)
  #BIC.ma[k] = AIC(fit.ma, k = log(length(n)))

}

dat.per = spec.pgram(detrended, taper=0, log="no")
dat.per = spec.pgram(dat_ts, taper=0, log="no")
dat.per = spec.pgram(residual, taper=0, log="no")
dat.per$spec[5]
f1 <-dat.per$freq[5] # period of 54 weeks
dat.per$spec[10]
f2 <- dat.per$freq[10] #period of 27 weeks
dat.per$spec[36]
f3 <- dat.per$freq[36] # period of 7.5 weeks
f4 <- dat.per$freq[2]  #135 weeks
f5 <- dat.per$freq[15] #18 weeks


abline(v=c(f1, f2, f3, f4, f5), lty=2)
```

```
U = qchisq(.025, 2)
L = qchisq(.975, 2)
2*dat.per$spec[5]/L
2*dat.per$spec[5]/U
2*dat.per$spec[10]/L
2*dat.per$spec[10]/U
2*dat.per$spec[36]/L
2*dat.per$spec[36]/U


spaic=spec.ar(detrended, log="no")

abline(v=(f1, f3, f3))



# to see the cycles better
dev.new()
par(mfrow=c(2,1))
plot(detrended)
plot(residual)

ker=kernel("modified.daniell", 6) # modified danielle kernel applied twice
L=round(1/sum(ker$coef^2)) # the value of L or sppans
plot(soif <- kernapply(detrended, ker), main = "Low Pass Filtered")
spec.ts <-spectrum(soif, spans=13, taper = 0)
abline(v=f1, lty=2)

ker=kernel("modified.daniell", c(3,3)) # modified danielle kernel applied twice
L=round(1/sum(ker$coef^2)) # the value of L or sppans
plot(f <- kernapply(detrended, ker), main = "Low Pass Filtered")
spec.ts <-spectrum(f, spans=c(7,7), taper = 0)
abline(v=f1, lty=2)


## componets
components <- decompose(dat_ts)
plot(components)
```

10 Plots

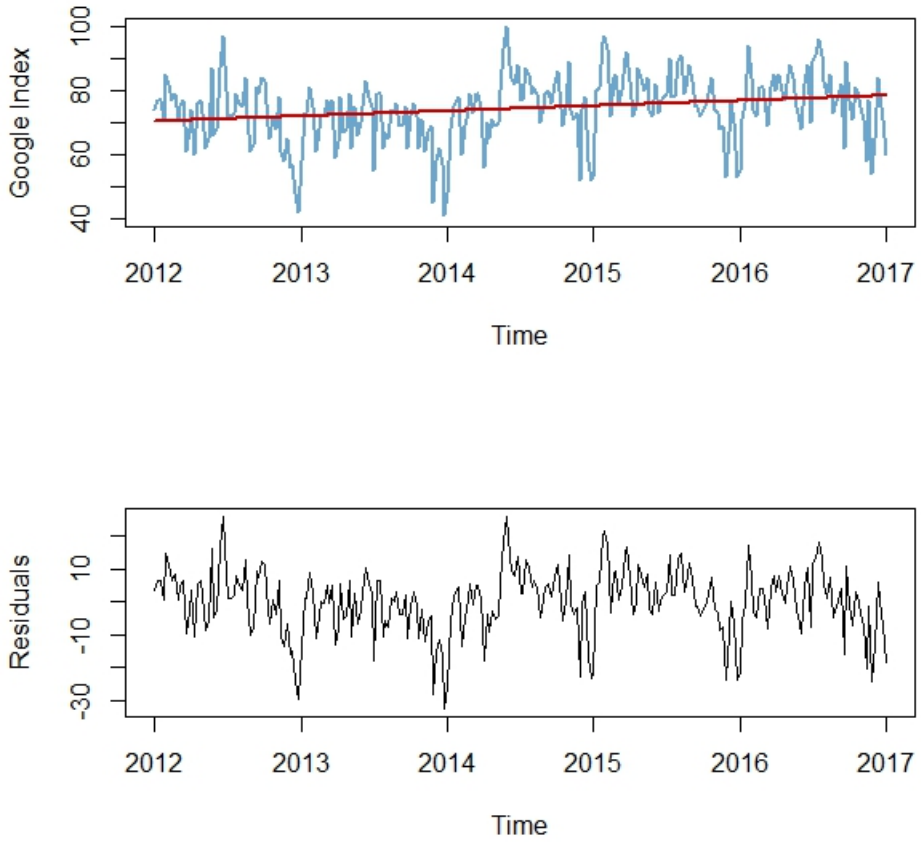

Figure 1: Time Series Plot of Google Index for General Contractor

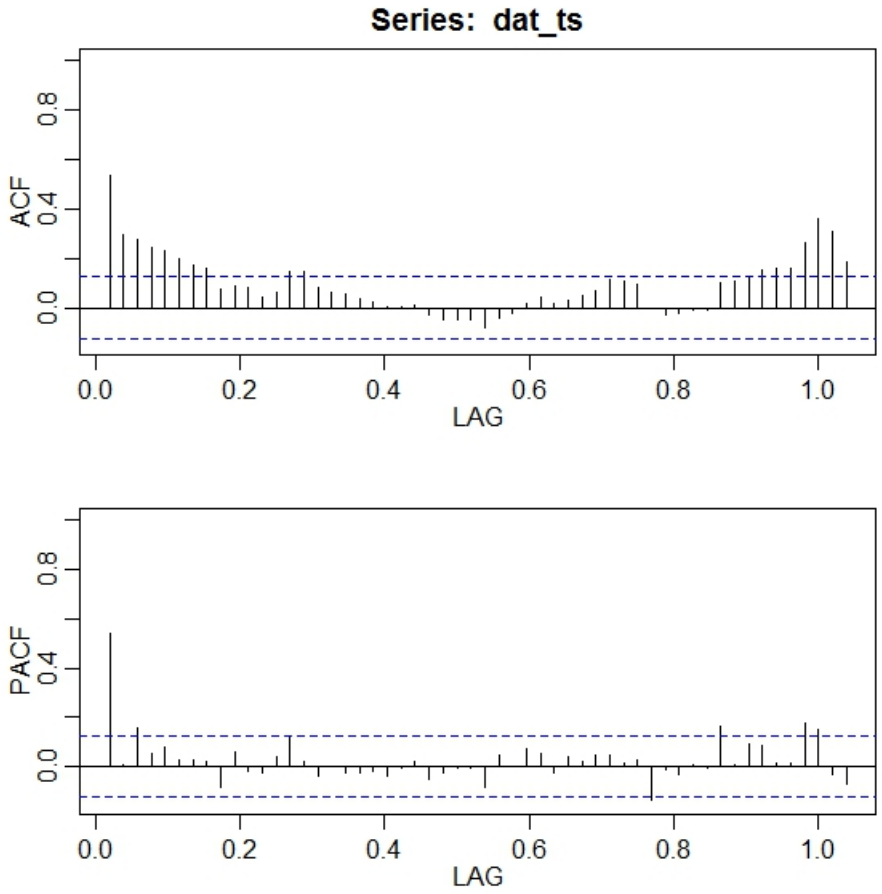

Figure 2: ACF and PACF

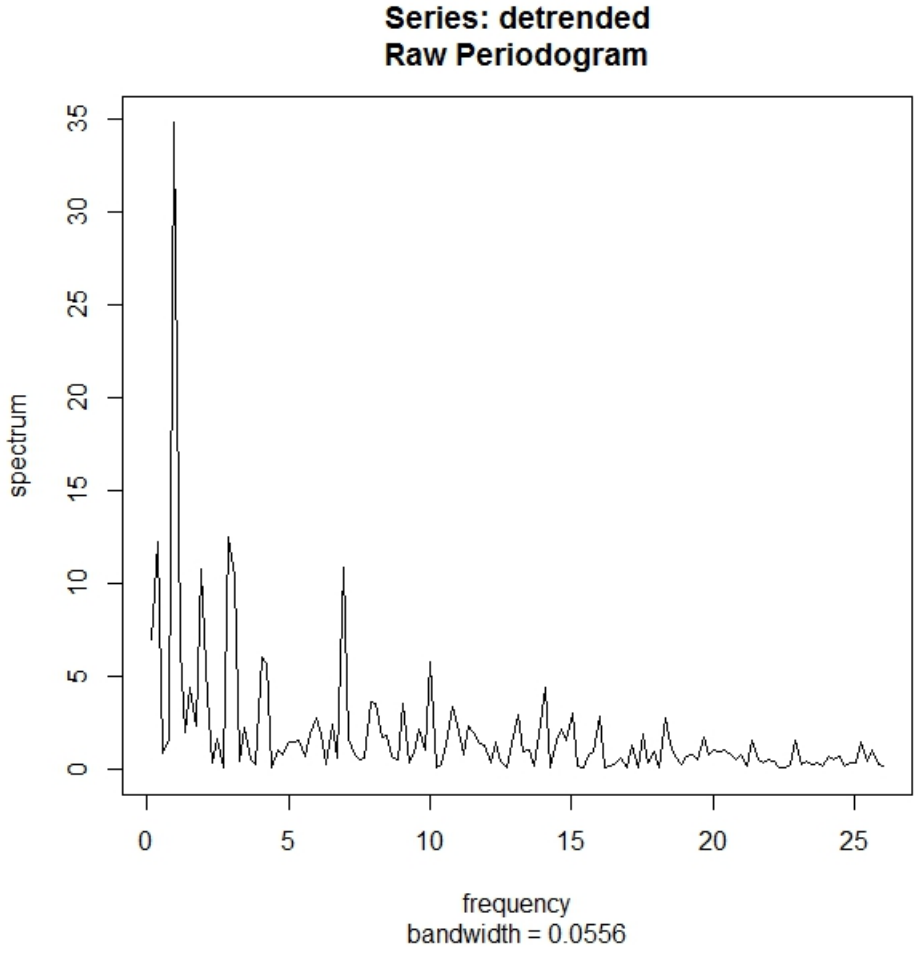

Figure 3: Periodogram with peak at annual cycle

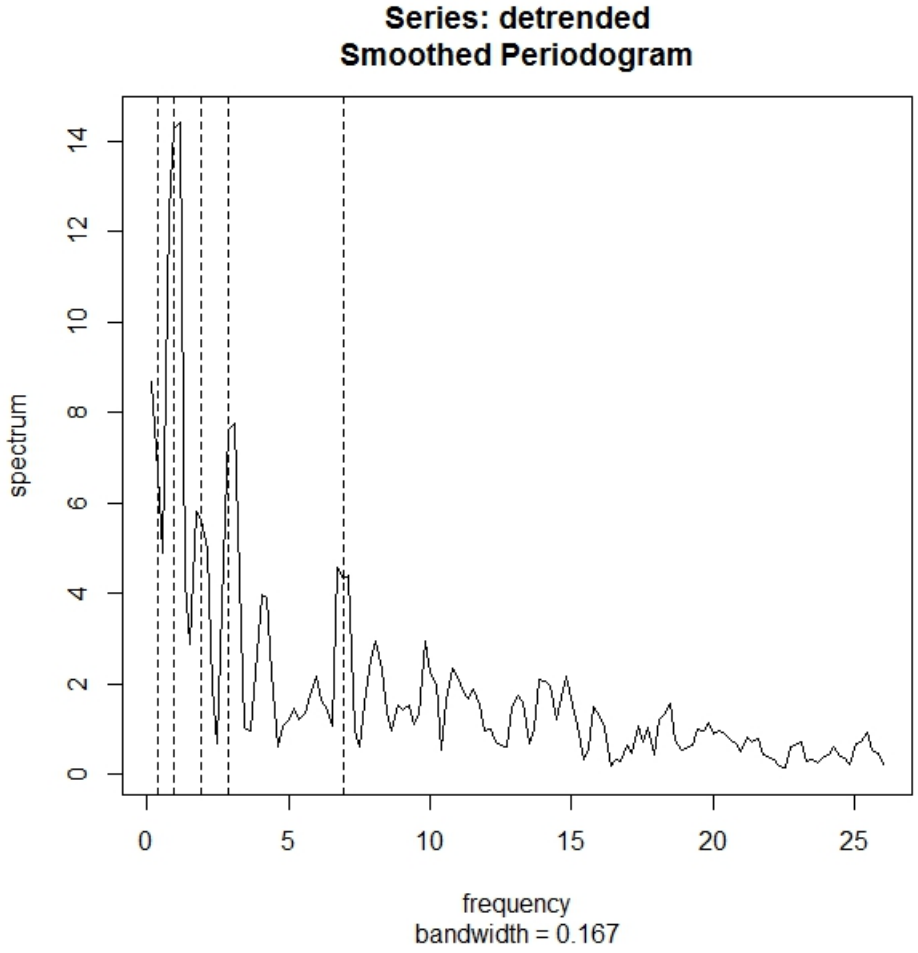

Figure 4: Average Periodogram with Daniell filter m=1 or L=3

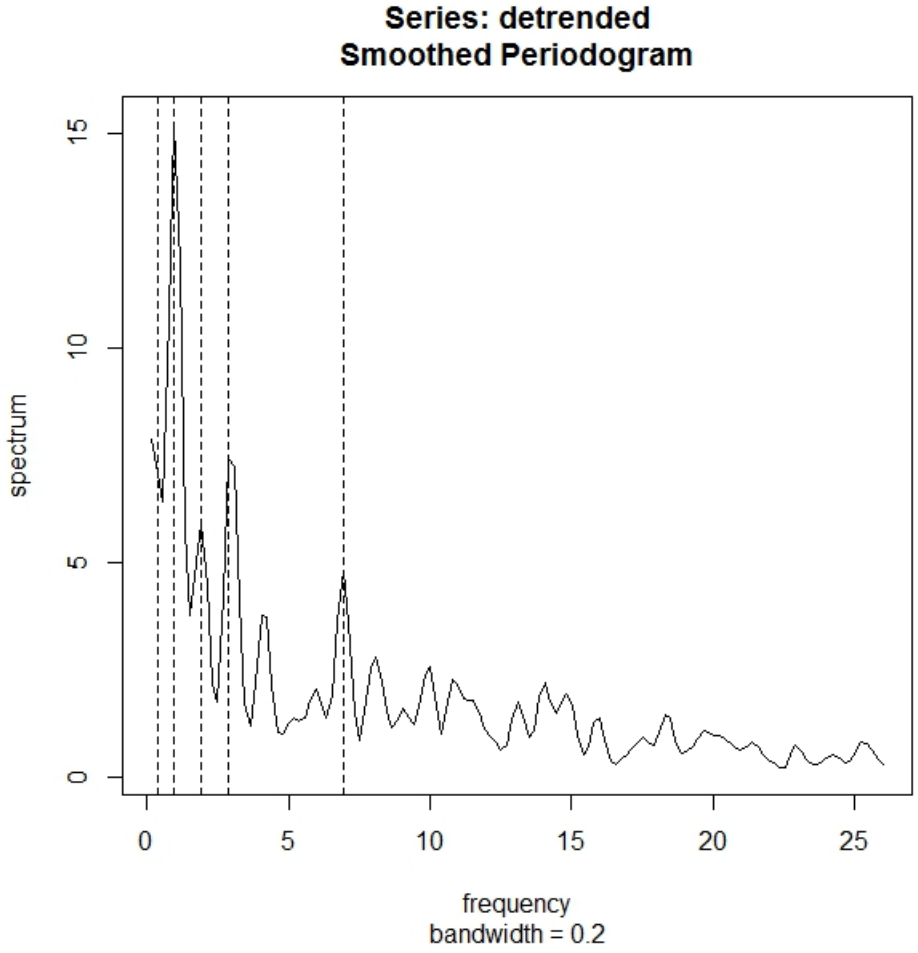

Figure 5: Average Periodogram with Daniell filter m=1 or L=3 applied twice

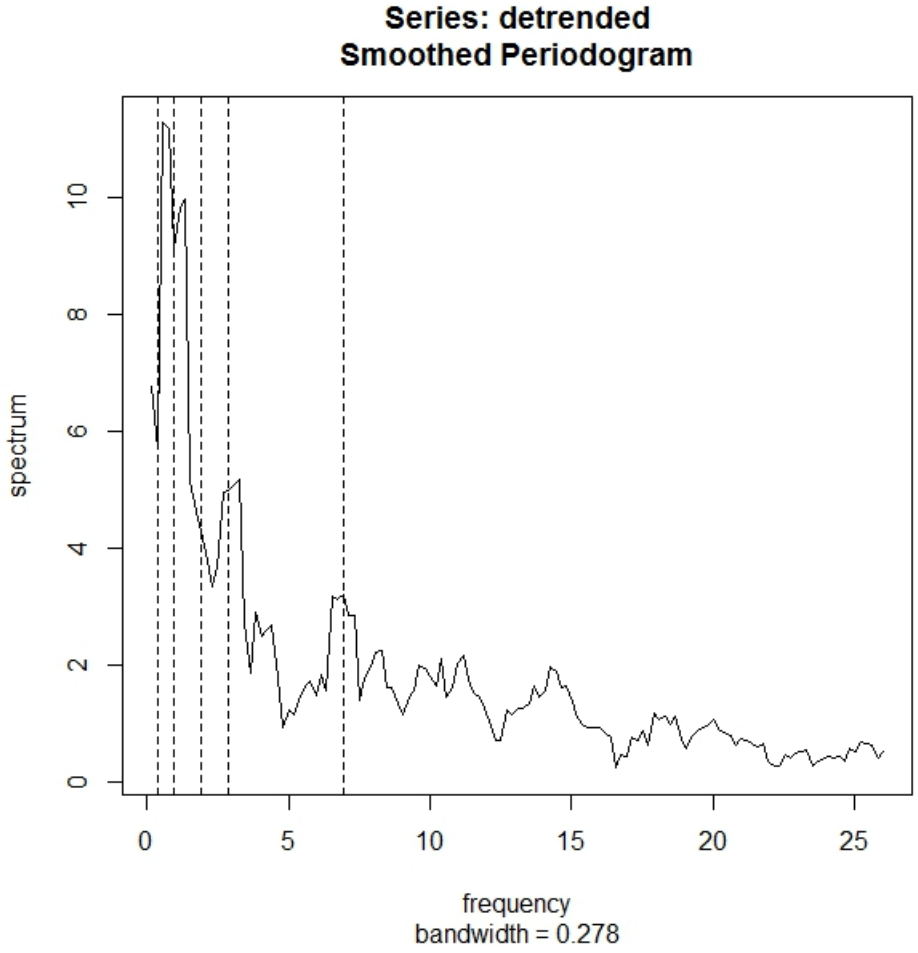

Figure 6: Average Periodogram with Daniell filter m=2 or L=5 applied once

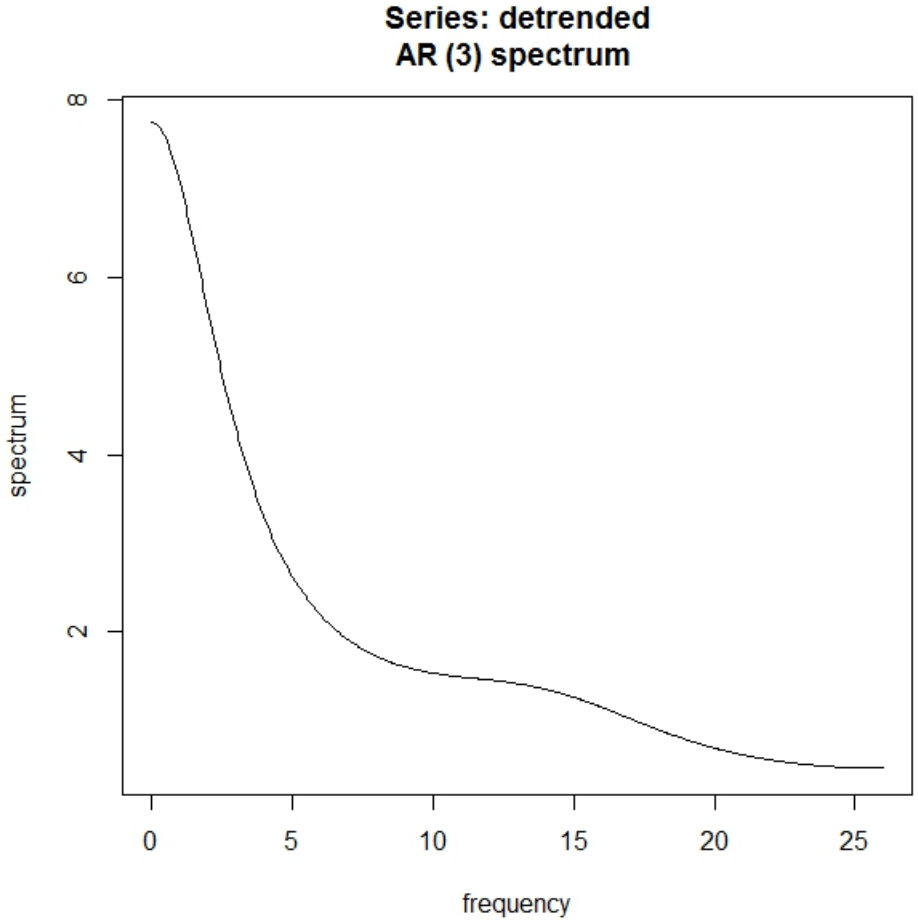

Figure 7: Parametic Spectral Estimation

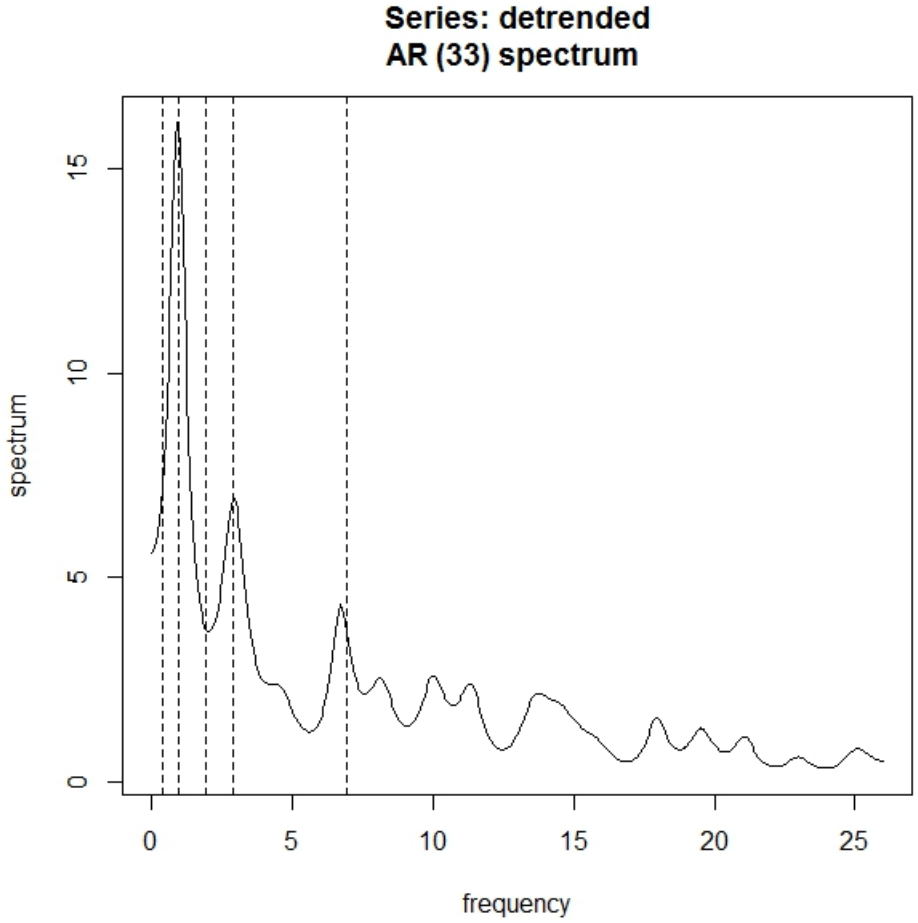

Figure 8: Order 33 Parametic Spectral Estimation

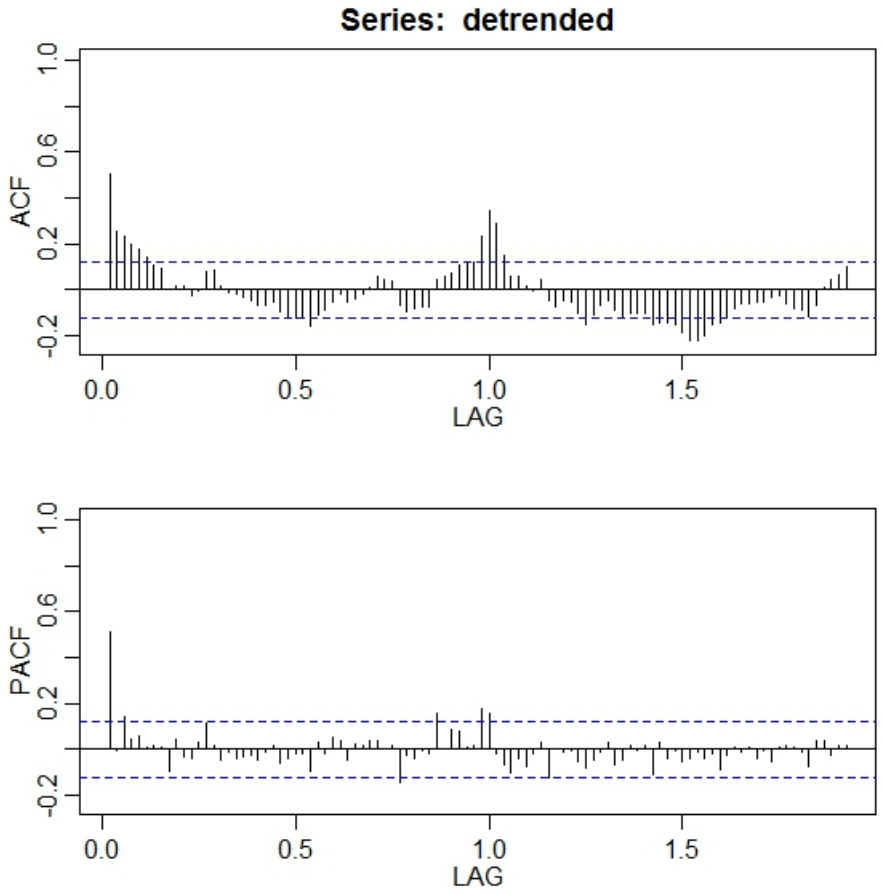

Figure 9: ACF and PACF of the detrended series

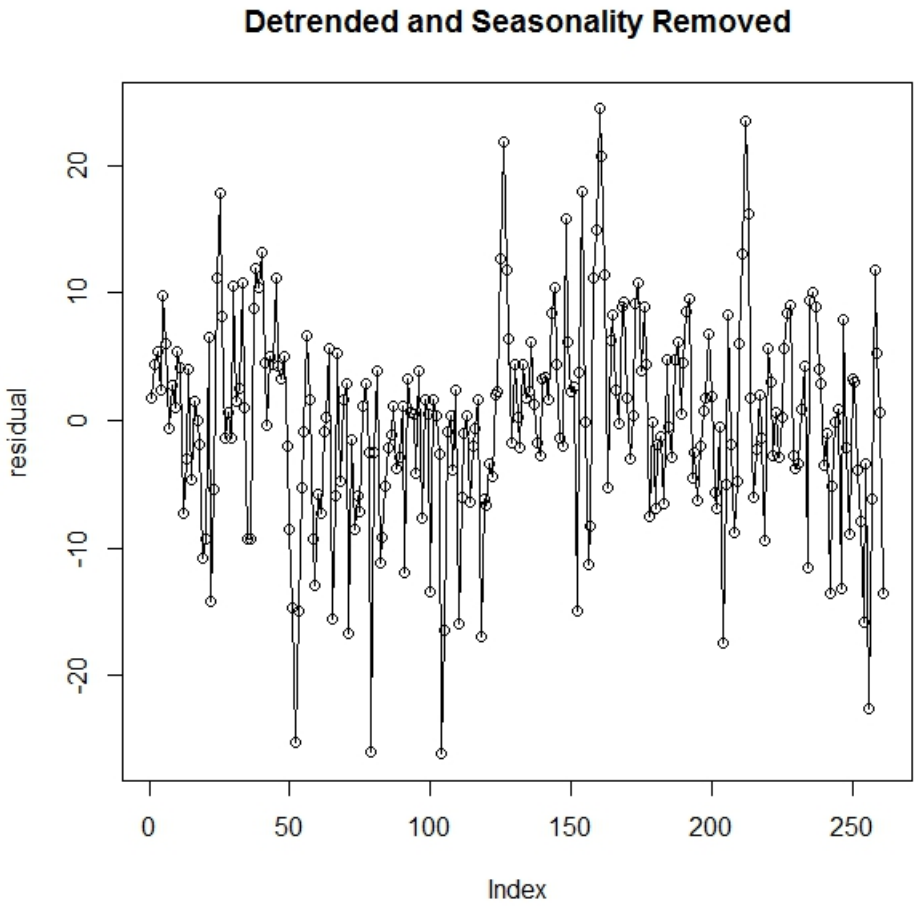

Figure 10: Plot of time series with seasonality removed

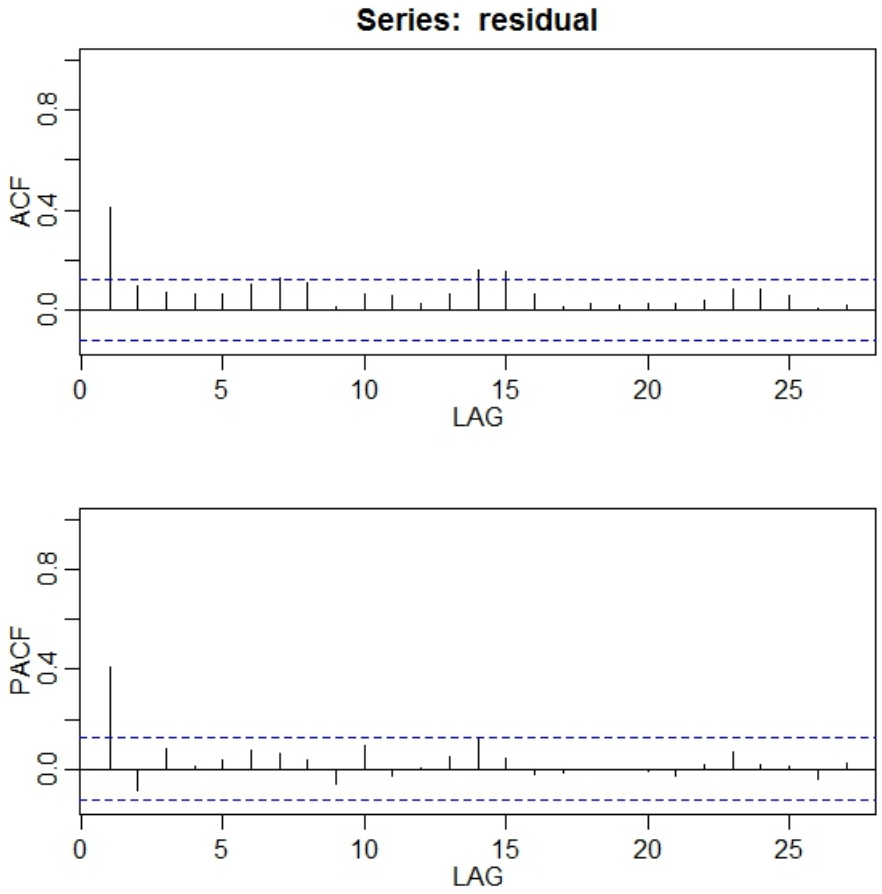

Figure 11: Sample ACF and PACF of time series with seasonality removed

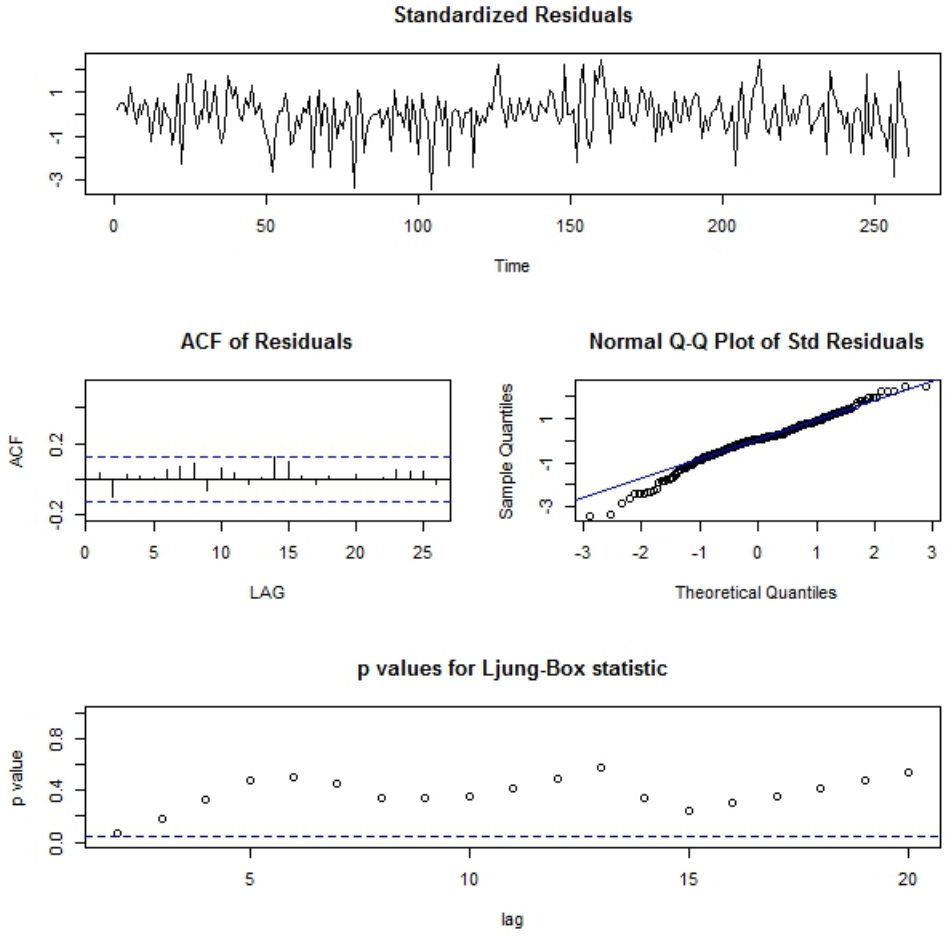

Figure 12: Sample ACF and PACF of time series with seasonality removed

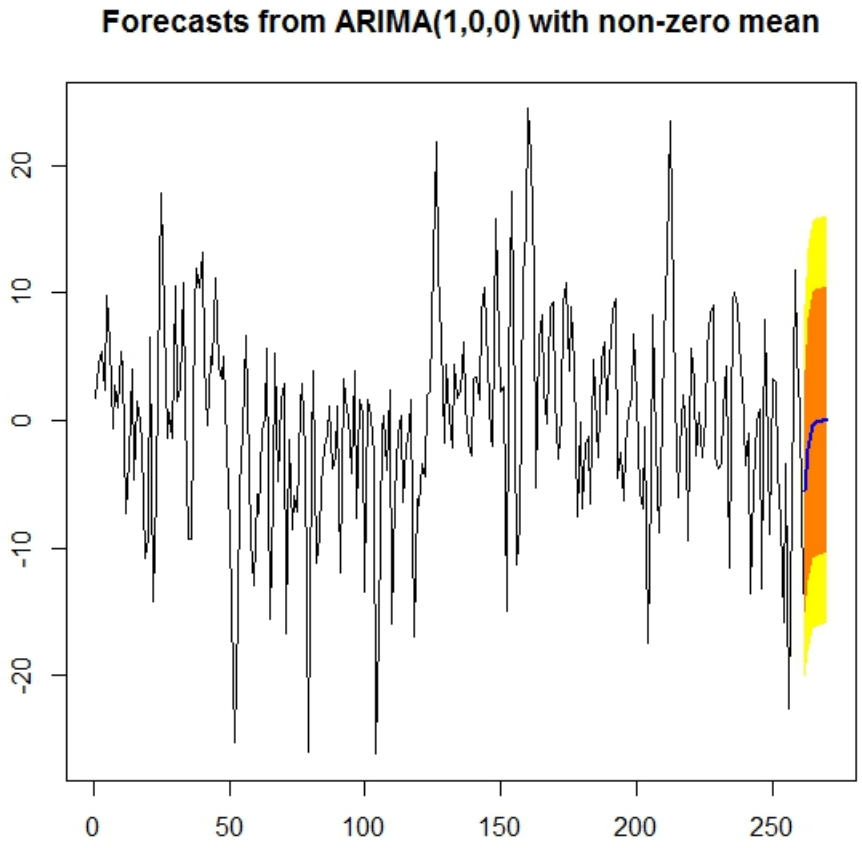

Figure 13: Forecast residuals 2 months using the Long Range Forecast

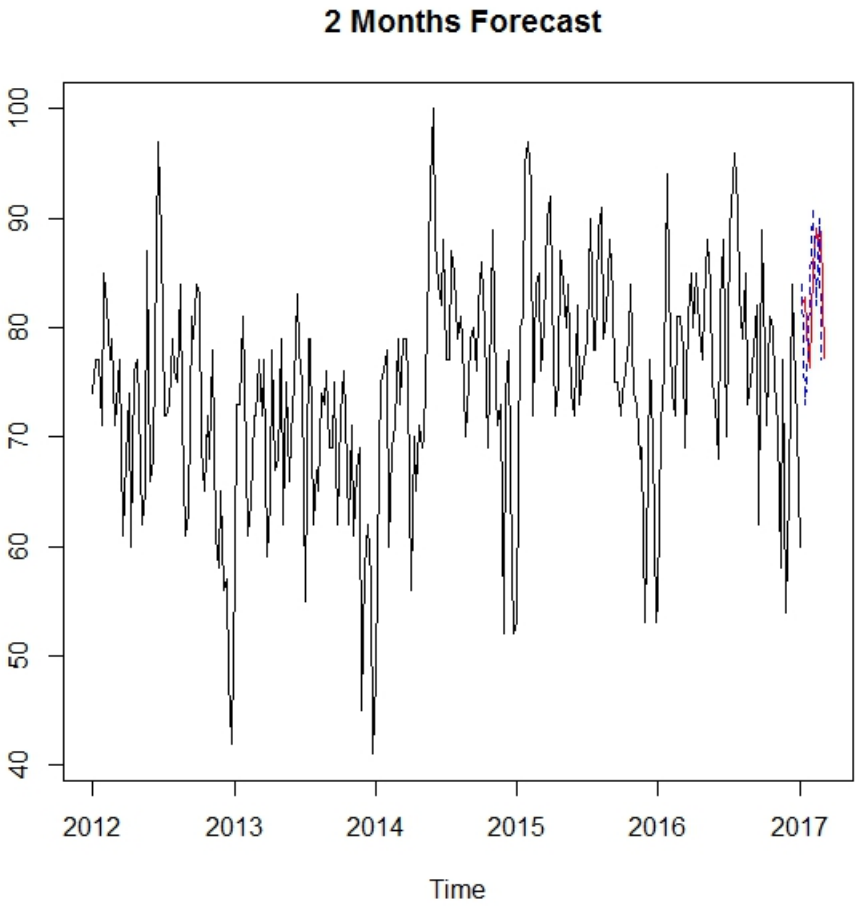

Figure 14: Forecast residuals 2 months using the One Step Ahead Forecast