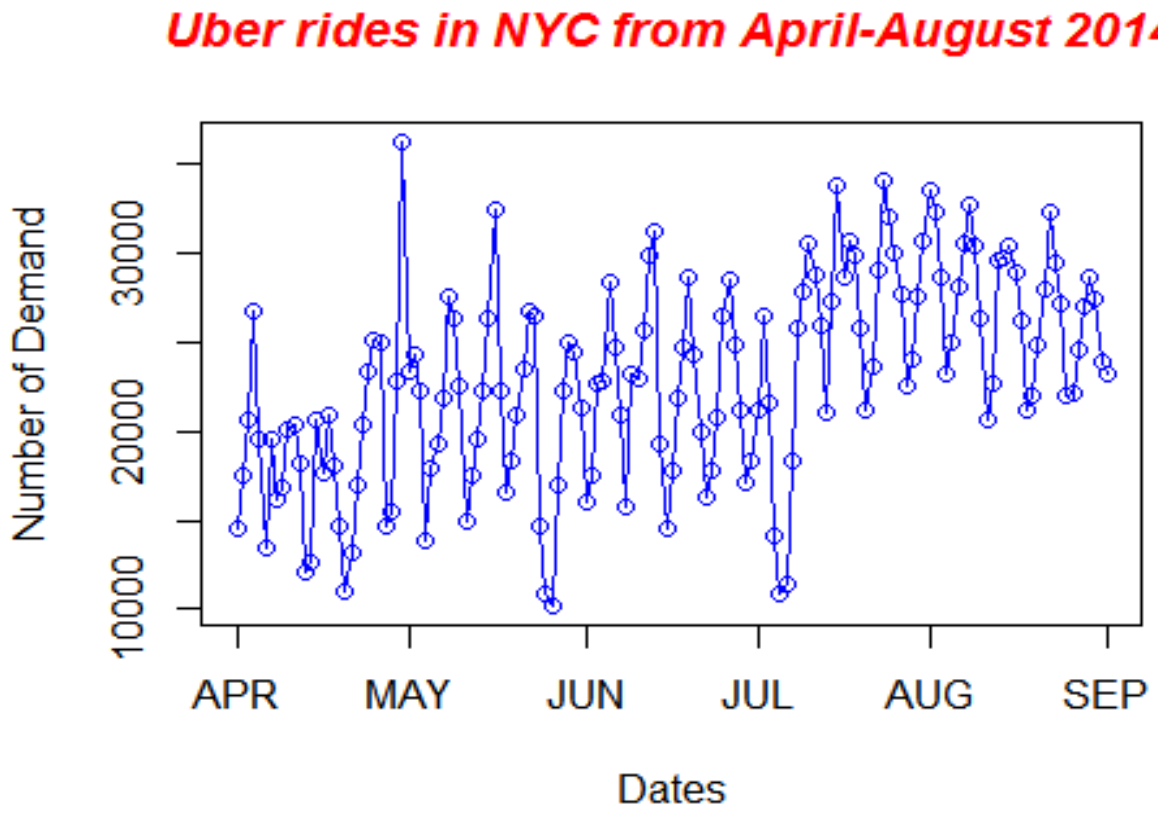
Final Project Uber Data Analysis.R

Soowhan Park

Fri Dec 04 23:53:54 2015

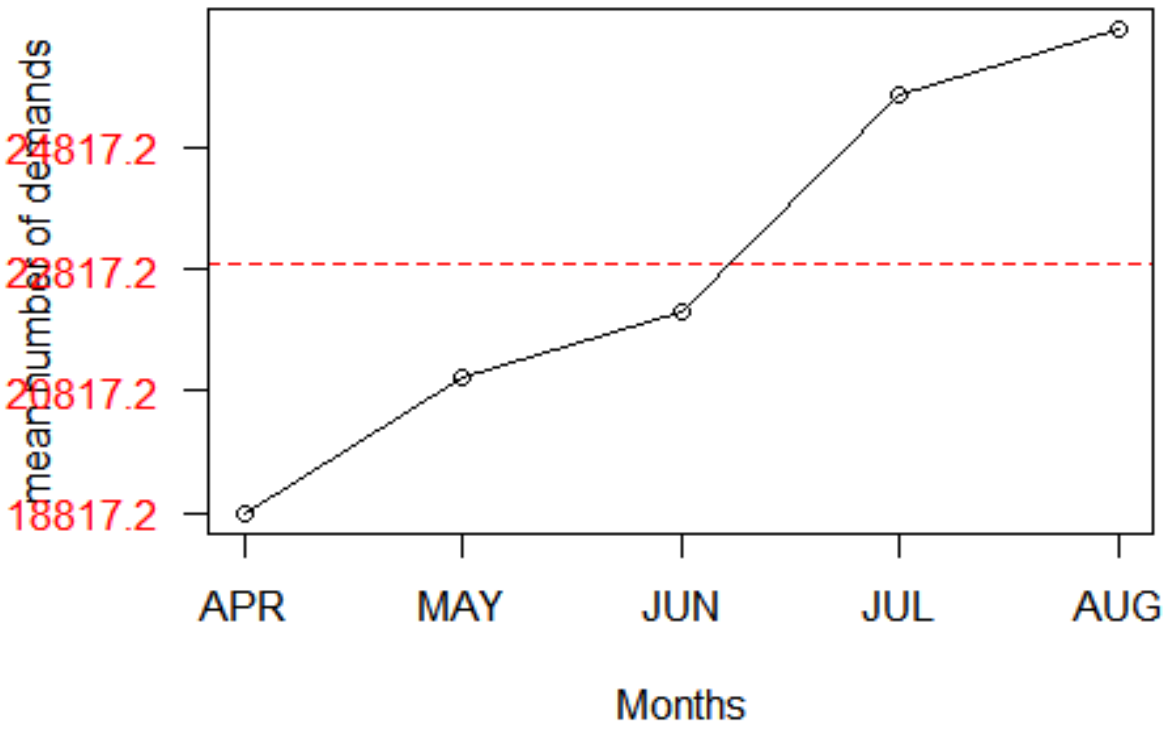
```
### Calling required libraries ###
library(astsa)
library(tseries)

### Data Mining ###
APR14 <- read.csv("uber-raw-data-apr14.csv.txt")
MAY14 <- read.csv("uber-raw-data-may14.csv.txt")
JUN14 <- read.csv("uber-raw-data-jun14.csv.txt")
JUL14 <- read.csv("uber-raw-data-jul14.csv.txt")
AUG14 <- read.csv("uber-raw-data-aug14.csv.txt")
SEP14 <- read.csv("uber-raw-data-sep14.csv.txt")


### countfun ###
countfun <- function(x) {
  q=1
  i=which(x==FALSE) #indexes of changing date (turn i)
  demand = matrix(, length(i), 1) #in case of 31 day months. This matrix cont
ains demand number in each day
  demand[1,] = i[1] #number of demand in first day
  ilag=c(0, diff(i)) # first 0 is to ignore 1st case
  for (q in 2:length(i)) {
    demand[q,] = ilag[q]
  }
  return(demand)
}

### sameday fun ###
samedayfun <- function(x) { #this function counts if the next ride is still o
n the same day
  a = data.matrix(x$Date.Time) #mine out date.time data and set it to matrix
  p=1 #start
  tfmat = matrix(, length(a), 1) #true false matrix
  for(i in 1:length(a)) {
    ini=substr(a[i], 1, 8) #setting initial counting date
    j=i+1                  #the next date
    nex=substr(a[j], 1, 8) #setting which to count
    if (is.na(nex)==TRUE) {
      result=rbind(countfun(tfmat), length(a)-tail(which(tfmat==F),1)) #techn
ical revision for my code
    } else {
      if (ini==nex) {
```

```
        tfmat[p,]=TRUE
        p=p+1
      } else {
        tfmat[p,]=FALSE
        p=p+1
      }
    }
  }
  return(result)
}

### demand fun ###
demandfun <- function(x, m) {
  if (any(c(1,3,5,7,8,10,12) == m)) { #meaning if the given month has 31 days
    dim(x) <- c(31,5)
  } else {
    dim(x) <- c(30,5)
  }
  #as you can see, my function disregards lunar calendar april since my
   analysis doesnt take special aprial into account (28 days)
  result=rowSums(x)
  return(result)
}


### Analysis begins ###
# The below data is what I am analyzing and using to predict which day or per
iods of days hit the high number of demands
APR14DATA=demandfun(samedayfun(APR14), 4)
MAY14DATA=demandfun(samedayfun(MAY14), 5)
JUN14DATA=demandfun(samedayfun(JUN14), 6)
JUL14DATA=demandfun(samedayfun(JUL14), 7)
AUG14DATA=demandfun(samedayfun(AUG14), 8)

# The below data is the actual result, which I want to compare my result to s
ee how close the prediction values are
SEP14DATA=demandfun(samedayfun(SEP14), 9)
# merging apr, may, jun, july, aug data.
APRtoAUG14 = c(APR14DATA, MAY14DATA, JUN14DATA,
               JUL14DATA, AUG14DATA)

# plotting to visualize the first glance of merged data
plot(APRtoAUG14, type="o", col="blue", xlab="Dates", ylab="Number of Demand",
xaxt='n')
title(main="Uber rides in NYC from April-August 2014",
      col.main="red", font.main=4)
xticks <- c("APR","MAY","JUN","JUL","AUG", "SEP")
axis(1, at= c(1, 31, 62, 92, 122, 153), lab=xticks)
```



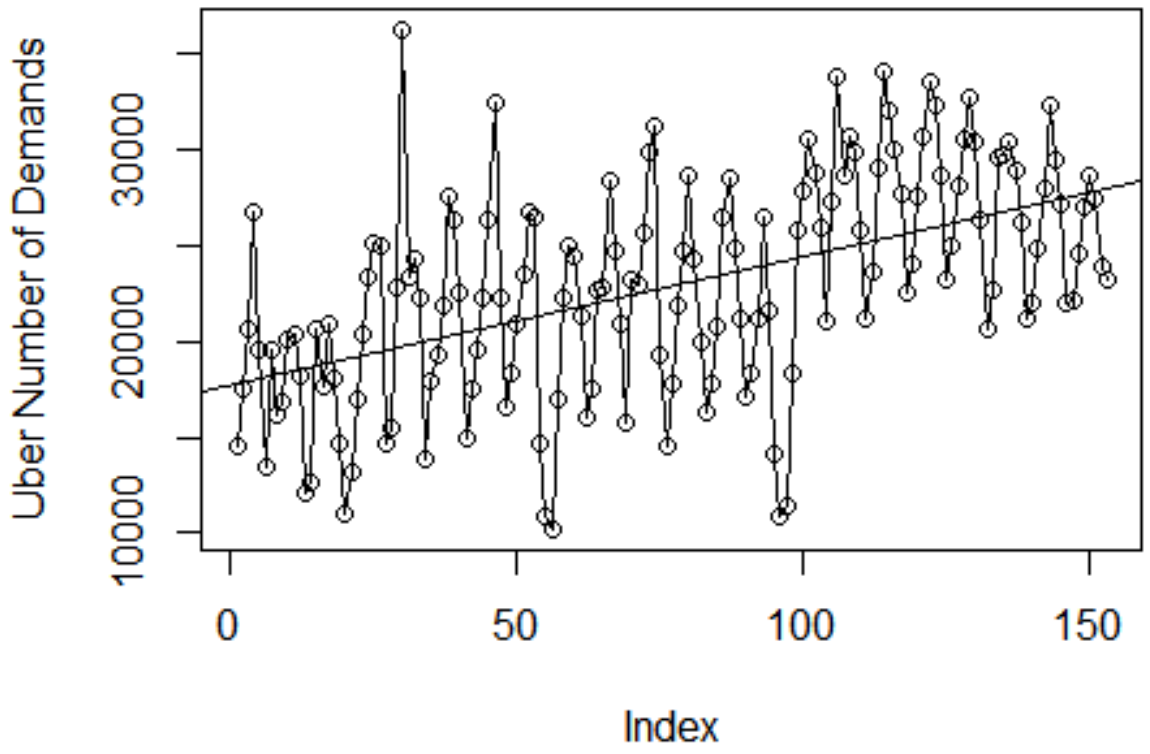

```
# Just by looking at first glance, the time series looks great for analysis
# because of seemingly randomness with some seasonal patterns.
# The demand graph looks like it has increasing average value implying non-st
ationarity but we can always take detrending or differencing. I prefer detren
ding in this case because unlike differencing, detrending keeps the neccesary
dependencies between data elements for estimation/prediction. Differencing is
good for forcefully coercing the data to stationarity for any further analysi
s, but worse than detrending in terms of estimating, which I am conducting.

### Checking Stationarity ###
# Let's check if the Time Series is stationary.
# compare mean of the total time series to means of each monthly data.

# We can always use adf.test or kpss.test to check the stationarity.
# Important thing to note is that the data is stationary if properties of
# stationary data are invariant of time. [Properties=mean, autocorrelations]
# I know that this data has an increasing trend (As Uber is fast-growing company)
# Random walk with drift model could be a candidate but the randomness
# entails seasonality/signal more than the pure randomness, so let's put that
  random walk aside.
# Proving manually that the current data is non-stationary;
  Mean is not constant.
```

```r
monthly_u = c(mean(APR14DATA),
              mean(MAY14DATA),
              mean(JUN14DATA),
              mean(JUL14DATA),
              mean(AUG14DATA))

plot(monthly_u, type="o", axes=F, xlab="Months", ylab="mean number of demands
")
abline(h=mean(APRtoAUG14), pch=22, lty=2, col="red")
yticks <- seq(mean(APR14DATA), mean(AUG14DATA), 2000)
xticks <- c("APR","MAY","JUN","JUL","AUG")
axis(2, at = yticks, labels = yticks, col.axis="red", las=2)
axis(1, at= 1:5, lab=xticks)
box()
```

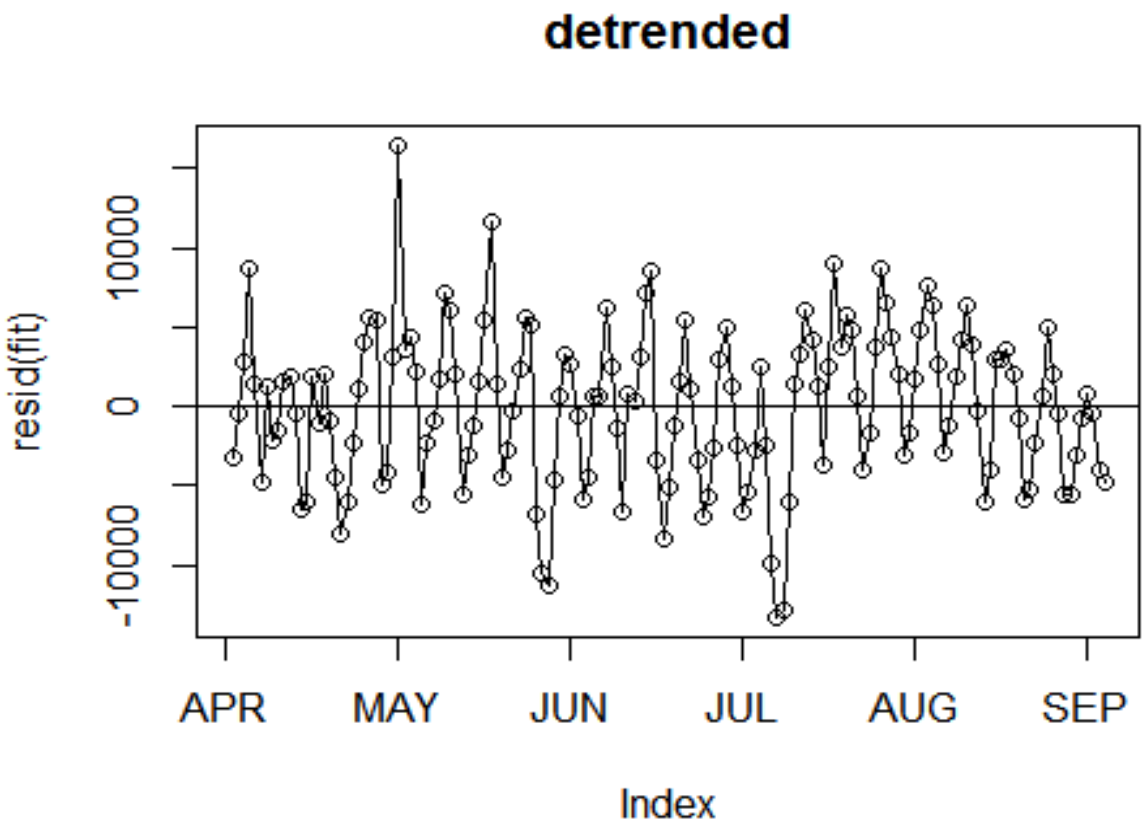



```r
# As we can see, the mean function is not constant.
# Let's see if we can find increasing mean via linear regression as well.

fit <- lm(APRtoAUG14~time(APRtoAUG14))
```

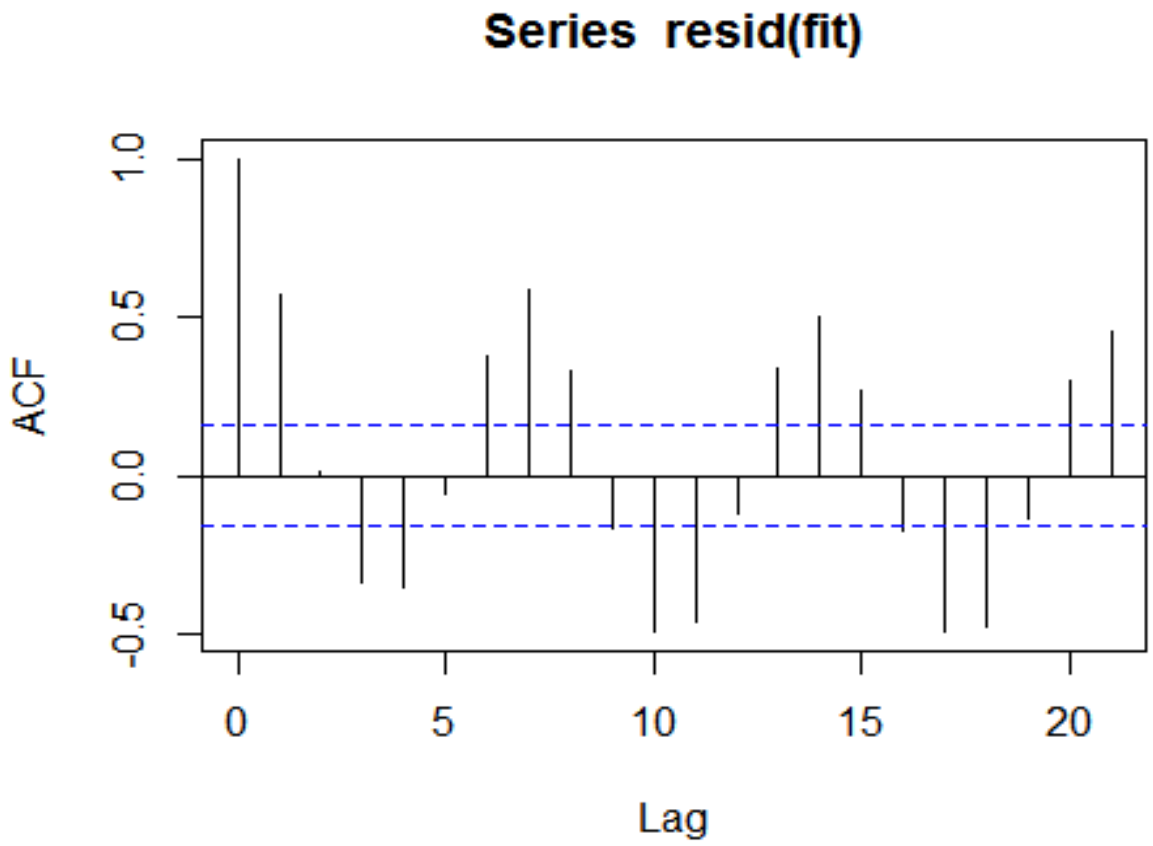
```r
plot(APRtoAUG14,type="o",ylab="Uber Number of Demands")
abline(fit)
```




```r
# Definitely, the line is not horizontal, meaning mean through time can't be
constant.
# Now, let's play with the data. First, let's make it stationary for analysis

plot(resid(fit), type="o", main="detrended", xaxt="n")
axis(1, at= seq(0,150,30), lab=c(xticks, "SEP"))
# Check that the mean is constant via linear regression.
# [You can always subgroup data by time and get means in each subgroups
# and compare if they are consistent (may not be perfect but must show consta
ncy over time)]

fitt <- lm(resid(fit)~time(resid(fit)))
abline(fitt)
```

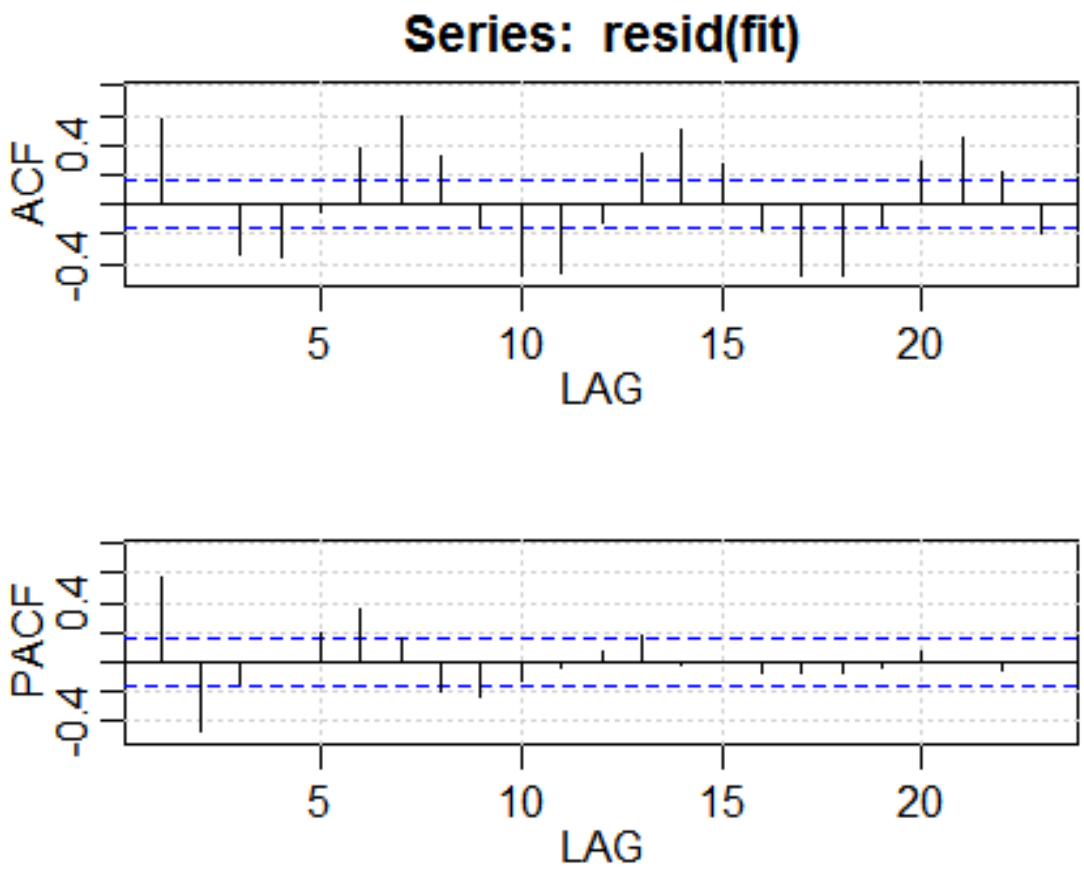



```
# Usually, you can take log here to get rid of any abnormal peaks
#(seemingly due to holiday season or national event requiring a lot of taxis)
# Since I want to predict when or which period in september Uber's demand is
high, the answer that I want is the such :
# 1) Will Weekends on september have high demands than weekdays?
# 2) If yes, which day? Friday, Saturday, or Sunday?
# 3) Which weeks have higher demand than other weeks?
# Therefore, I do not have to worry about MAY's very high demand (May day hol
iday)


# and July's very low demand (Independence day holiday)
# Now, let's look at dependencies between each data points.
# Check ACF


acf(resid(fit))
```




```
# For autocorrelations, weakly stationary time series show that the ACF
# only depends on the time distance between the data points, not on the actua
l time.
# So, we see that the tick marks at lag 1 for ACF, and after that it shows pe
riodicity.
# Other than that periodicity, autocorrelations are all in the boundaries.
# These periodicity can be interpreted as the following :
# In that distance (here it is 7), autocorrelations should be almost 0 becaus
e
# uber ride demand correlations within a week should be 0.
# We know that people who order Uber on weekends a lot will likely order on n
ext weekend, not on weekdays,
# because of, for example, party people (may have to watch out for drinking s
o always Uber)
# and people who order uber usually on weekdays will likely order again on an
other weekdays (working)

# So I conclude that the data is stationary. Now, let's build ARMA model.
# Note: I was thikning about smoothing the data but this would be good
#       if we have a large amount of data, then we can see which period
#       will likely to result in high demand than other periods.
#       Since we have only apr to august data, and since I have sep data to c
```

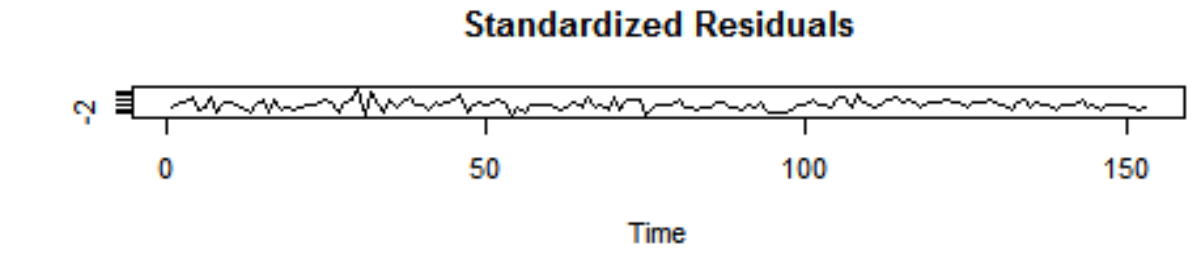
```
heck,
#       I will try to estimate the likely demand in each day and for this pur
pose,
#       I won't do smoothing.

# Let's check ACF and PACF to see which p, q would be appropriate
# for ARMA model

acf2(resid(fit))
```

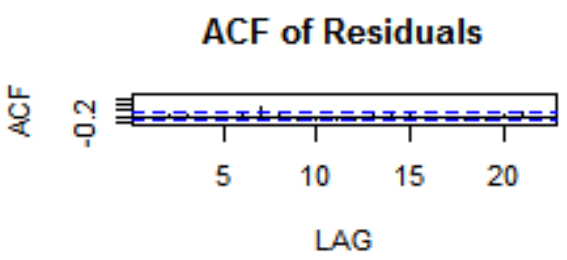


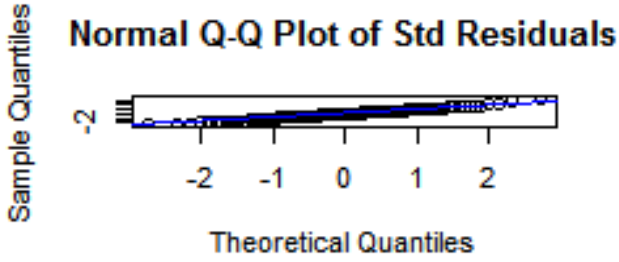

```
# ACF decays exponentially and PACF has large ticks at lag 1 and 2 -> AR(2) s
uggested

# Check if AR(2) is good

sarima(resid(fit), 2,0,0)
```

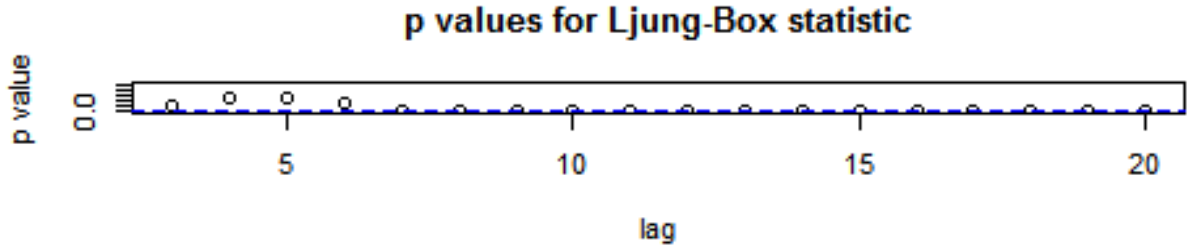
```
## initial  value 8.495610
## iter   2 value 8.298802
## iter   3 value 8.178177
```

```
## iter   4 value 8.170870
## iter   5 value 8.163677
## iter   6 value 8.163660
## iter   7 value 8.163658
## iter   7 value 8.163658
## iter   7 value 8.163658
## final  value 8.163658
## converged
## initial  value 8.162039
## iter   2 value 8.162022
## iter   3 value 8.162018
## iter   4 value 8.162017
## iter   4 value 8.162017
## iter   4 value 8.162017
## final  value 8.162017
## converged
```



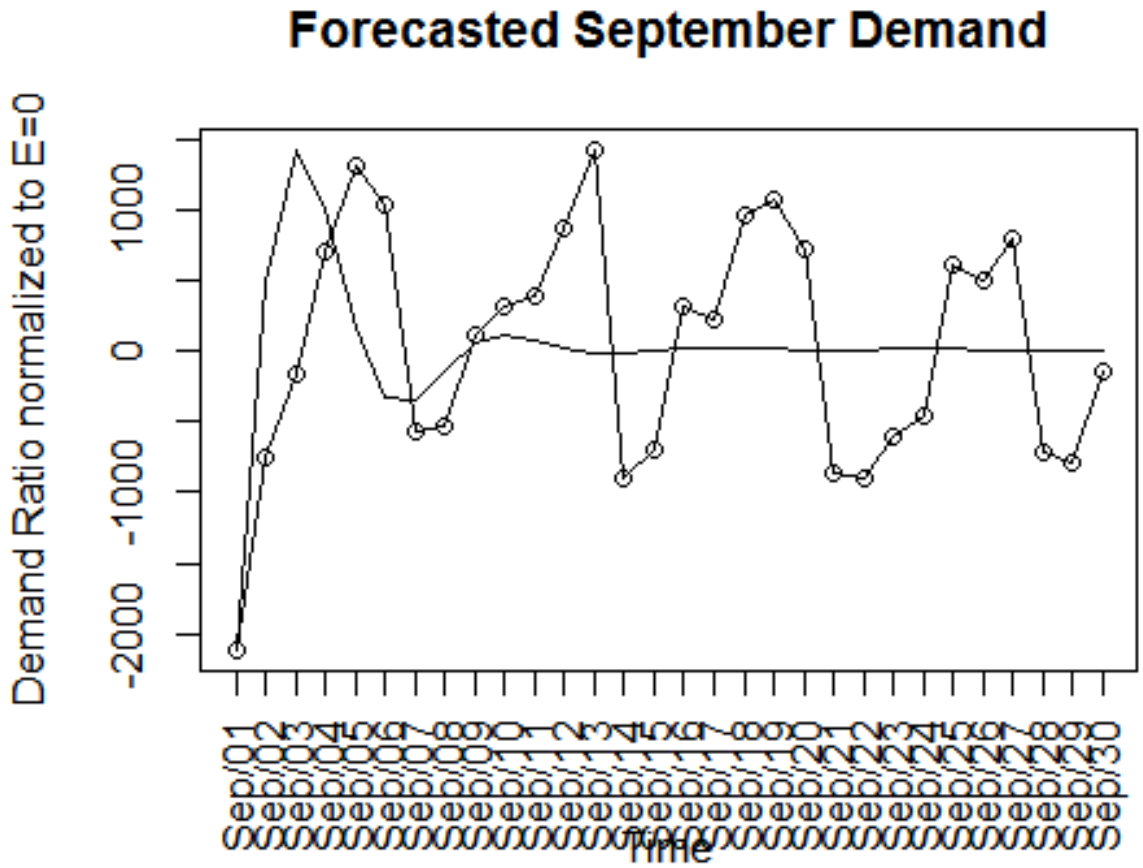




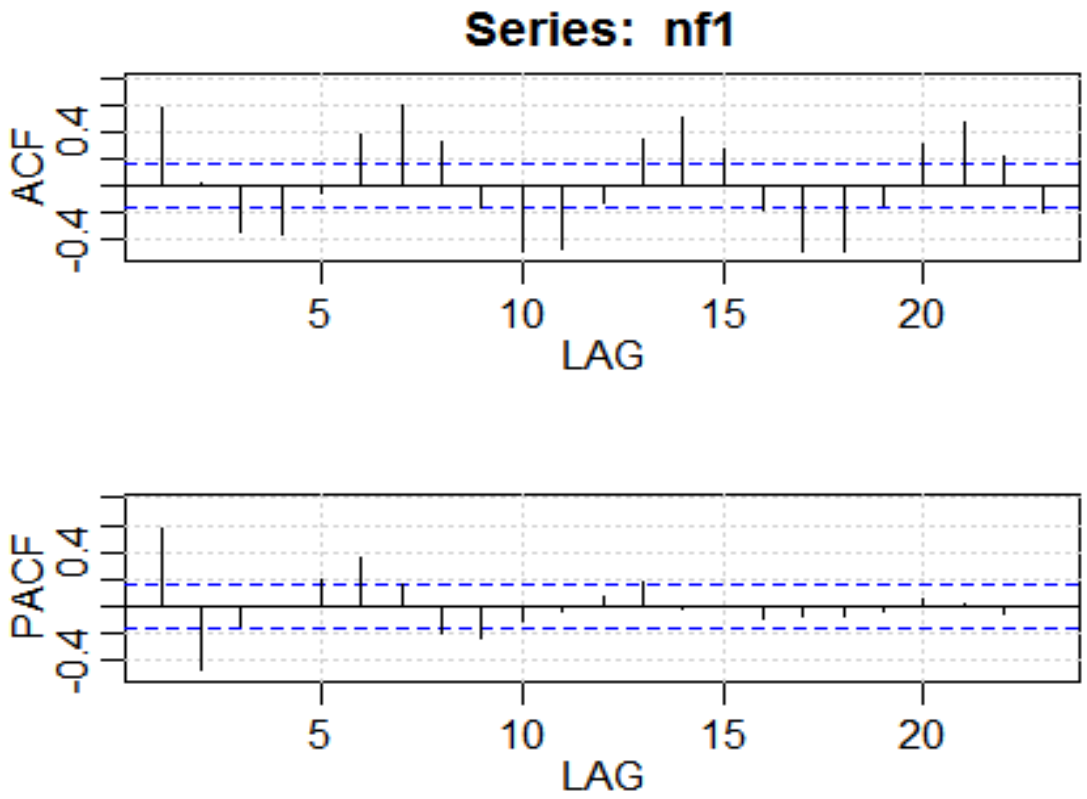



```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D
```

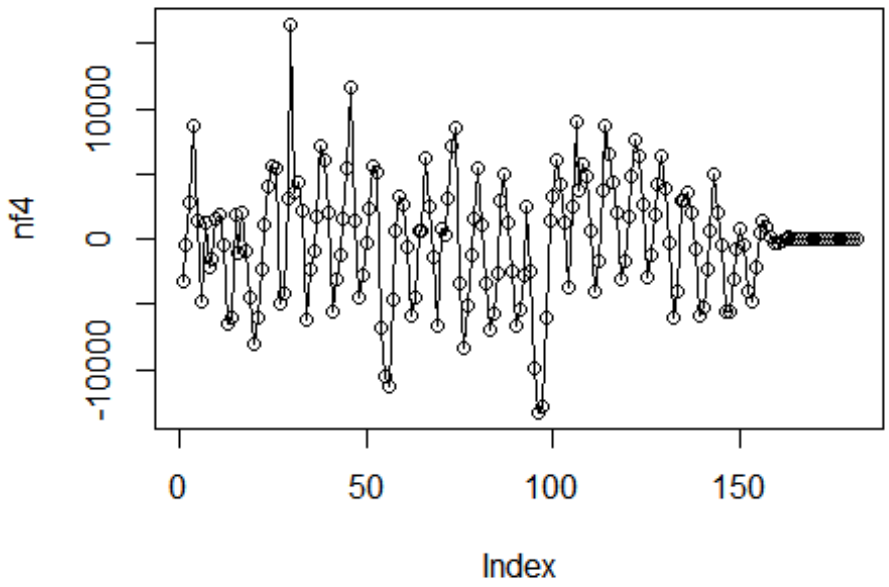
```
,
##     Q), period = S), xreg = xmean, include.mean = FALSE, optim.control = l
ist(trace = trc,
##     REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1      ar2     xmean
##       0.8489  -0.4745   -9.2104
## s.e.  0.0709   0.0707  452.1807
##
## sigma^2 estimated as 12213700:  log likelihood = -1465.89,  aic = 2939.77
##
## $AIC
## [1] 17.35728
##
## $AICc
## [1] 17.37212
##
## $BIC
## [1] 16.4167

# Inspecting the result, there are some outliers in standardized residuals
# but with a few values exceeding 3 standard deviations in magnitude.
# ACF shows no apprarent departure from the model assumption (at lag 7, ignor
able spike)
# QQ plot shows departures at the tail (almost at the end of august)
# In general, I would say that AR(2) is appropirate.

# Let's try fitting it to AR(2)

# Use ar.ols -> fitting an autoregressive time series model to the data
# by ordinary least squares, by default selecting the complexity by AIC.
regr_pre = ar.ols(resid(fit), order=2, demean=F, intercept=T)
fore = predict(regr_pre, n.ahead=30) #estimate the next 30 days (september)


plot(fore$pred, main="Forecasted September Demand", xaxt='n', ylab='Demand Ra
tio normalized to E=0')
x=seq(as.Date("2014/9/1"), as.Date("2014/9/30"), "days")
axis(1, at=154:183, labels=format(x, '%b/%d'), las=2, font=0.5)
par(new=TRUE) # To put actual september data
plot(resid(lm(SEP14DATA~time(SEP14DATA))), axes = FALSE, xlab = "", ylab = ""
, type='o')
```




```
# As we can see, forecasting this way did not predict which day
# will have high demand. Only thing we can get out of here is that
# there is some high demand peak in the first week.

# Let's try forecasting a week by week.
fore1= predict(regr_pre, n.ahead=7) #estimate the next 7 days (september 1st
week)
nf1= c(resid(fit), fore1$pred)
acf2(nf1)
```

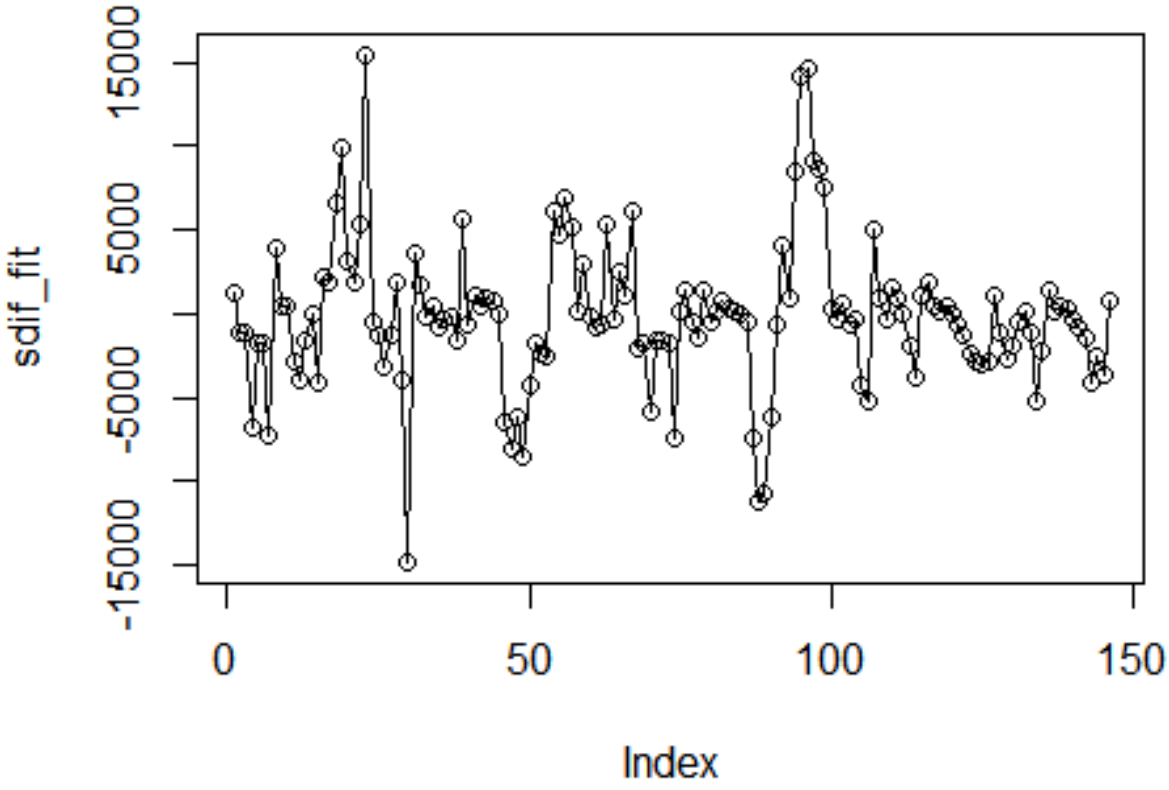



```
# Intuitively, the forecasted values come from our model, so we can keep p,q
s in our models.
regr_pre1 = ar.ols(nf1, order=2, demean=F, intercept=T)
fore2 = predict(regr_pre1, n.ahead=7) #estimate the next 7 days (september 2n
d week)
nf2= c(nf1, fore2$pred)

# conduct two more times to get 4 weeks of prediction.
regr_pre2 = ar.ols(nf2, order=2, demean=F, intercept=T)
fore3 = predict(regr_pre2, n.ahead=7) #estimate the next 7 days (september 2n
d week)
nf3= c(nf2, fore3$pred)

regr_pre3 = ar.ols(nf3, order=2, demean=F, intercept=T)
fore4 = predict(regr_pre3, n.ahead=7) #estimate the next 7 days (september 2n
d week)
nf4= c(nf3, fore4$pred)

# Now, plot the entire data (with 28 predicted values at the end)
plot(nf4, type="o")
```

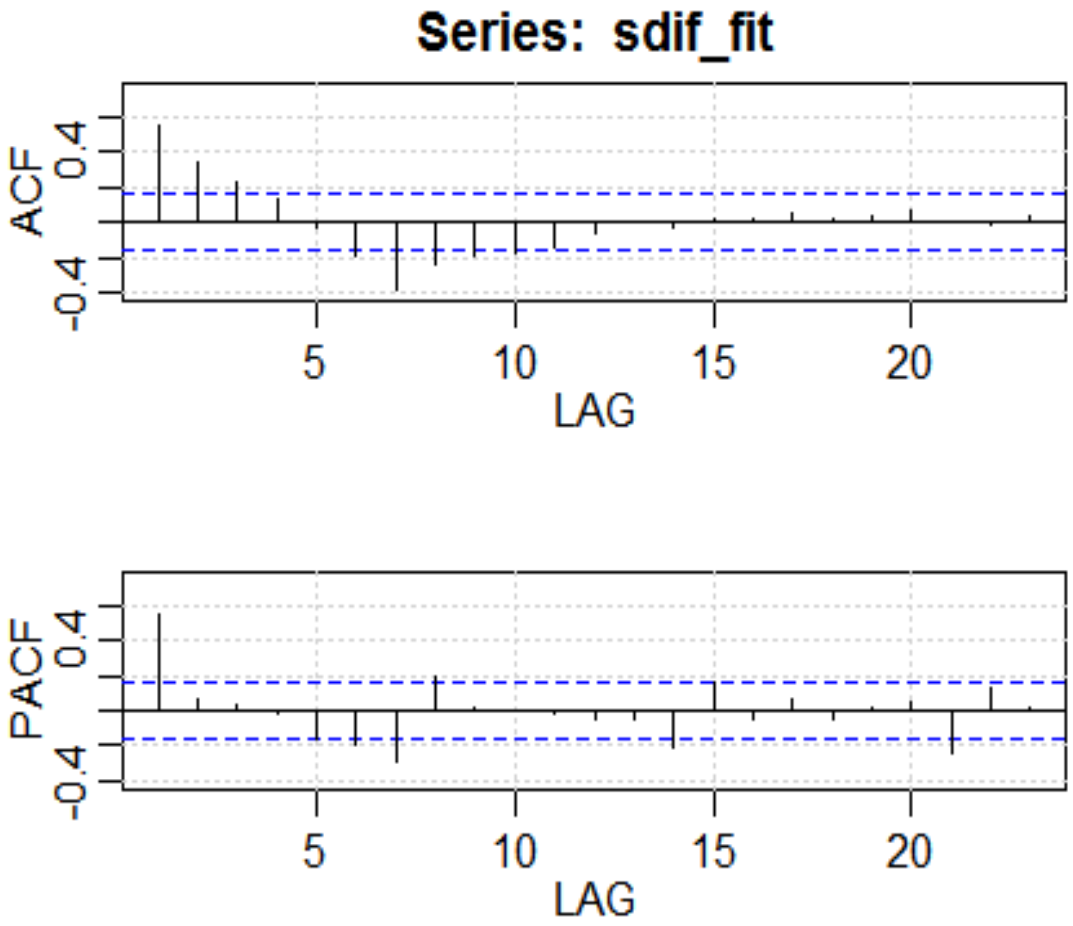



```
## It output the same result. This means that R knows that the data's varianc
e
## decreasing in general as the time goes. That is why our functions predict
that
## the demand eventually converges. This makes sense if we think about it,
## because Uber is an new company and as the time goes, there are going to be
## people who always use, and not always use Uber. However, in reality,
## obviously, it is not going to converge to 0. Though , we can probably
## guess that the first week (or 2nd) will have the highest peaks
## Now let's take a look at seasonal ARIMA analysis.
```

```
### Seasonal Analsysis ###

sdif_fit = diff(resid(fit), 7)
plot(sdif_fit,type="o")
```

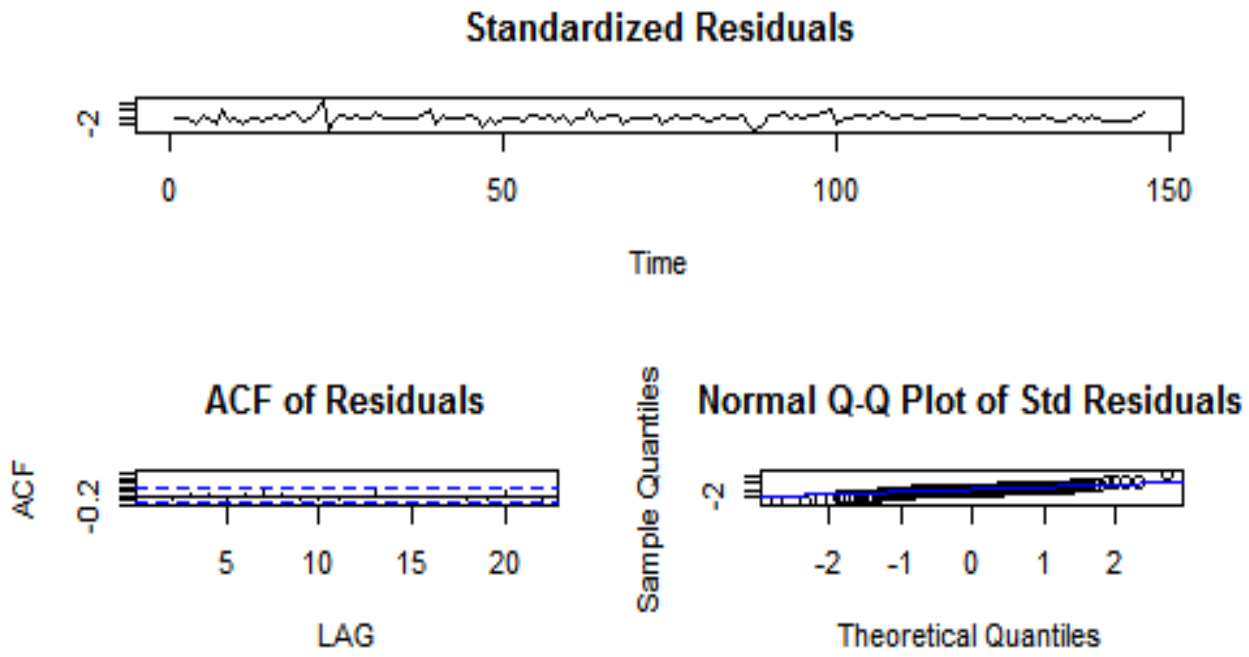


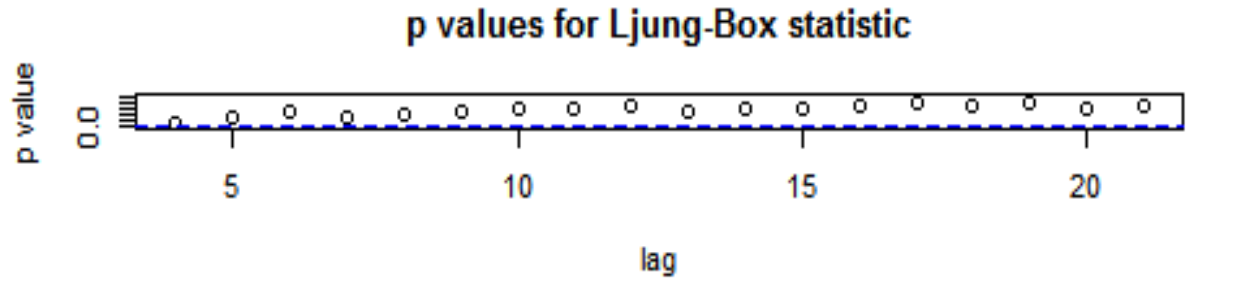
```
acf2(sdif_fit)
```




```
# Case: ACF is cutting off after lag1s and PACF is taling off -> SMA(1)
# Case: ACF is cutting off after lag2s and PACF is taling off -> SMA(2)
# Case: ACF is cutting off after lag3s and PACF is taling off -> SMA(3)

# Case: ACF and PACF both tailing off
# because of a spike in PACF          -> SARMA(1,1), SARMA(1,2) or SARMA(1,3)
```

```
sarima(sdif_fit, 2,0,0,0,0,1,7)
```

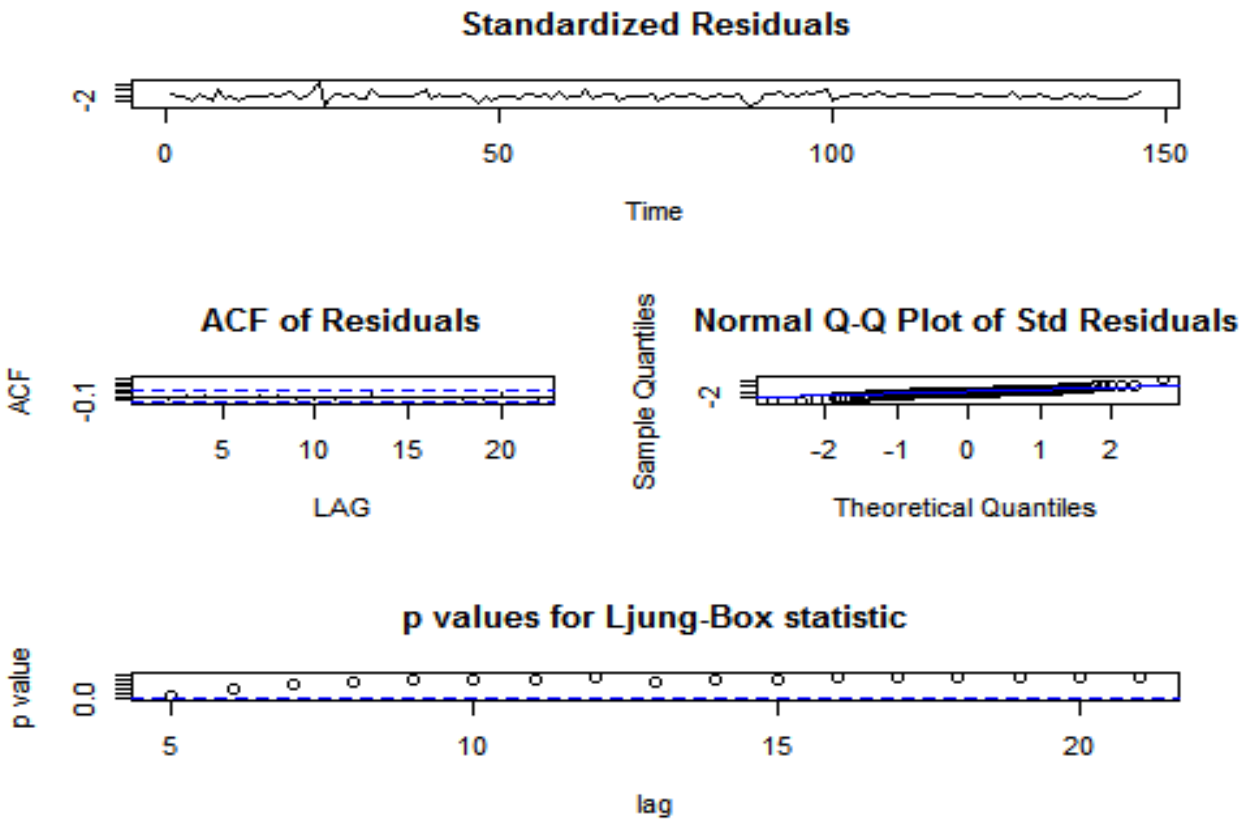





```
## $AIC
## [1] 16.92449
##
## $AICc
## [1] 16.94112
##
## $BIC
## [1] 16.00623
```

```
#16.92449, 16.94112, 16.00623
```

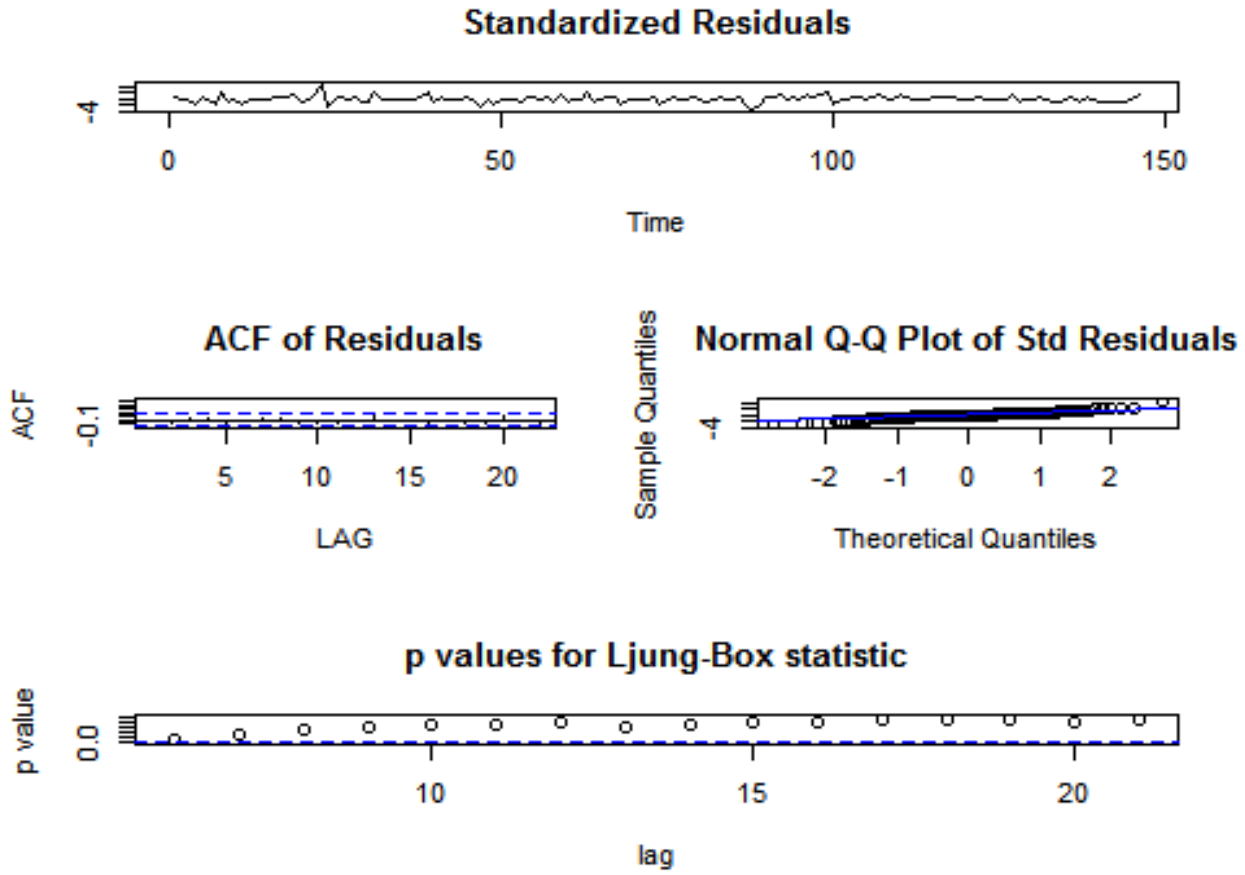
```
sarima(sdif_fit, 2,0,0,0,0,2,7)
```




```
## $fit
## $AIC
## [1] 16.88511
##
## $AICc
## [1] 16.90295
##
## $BIC
## [1] 15.98729
```

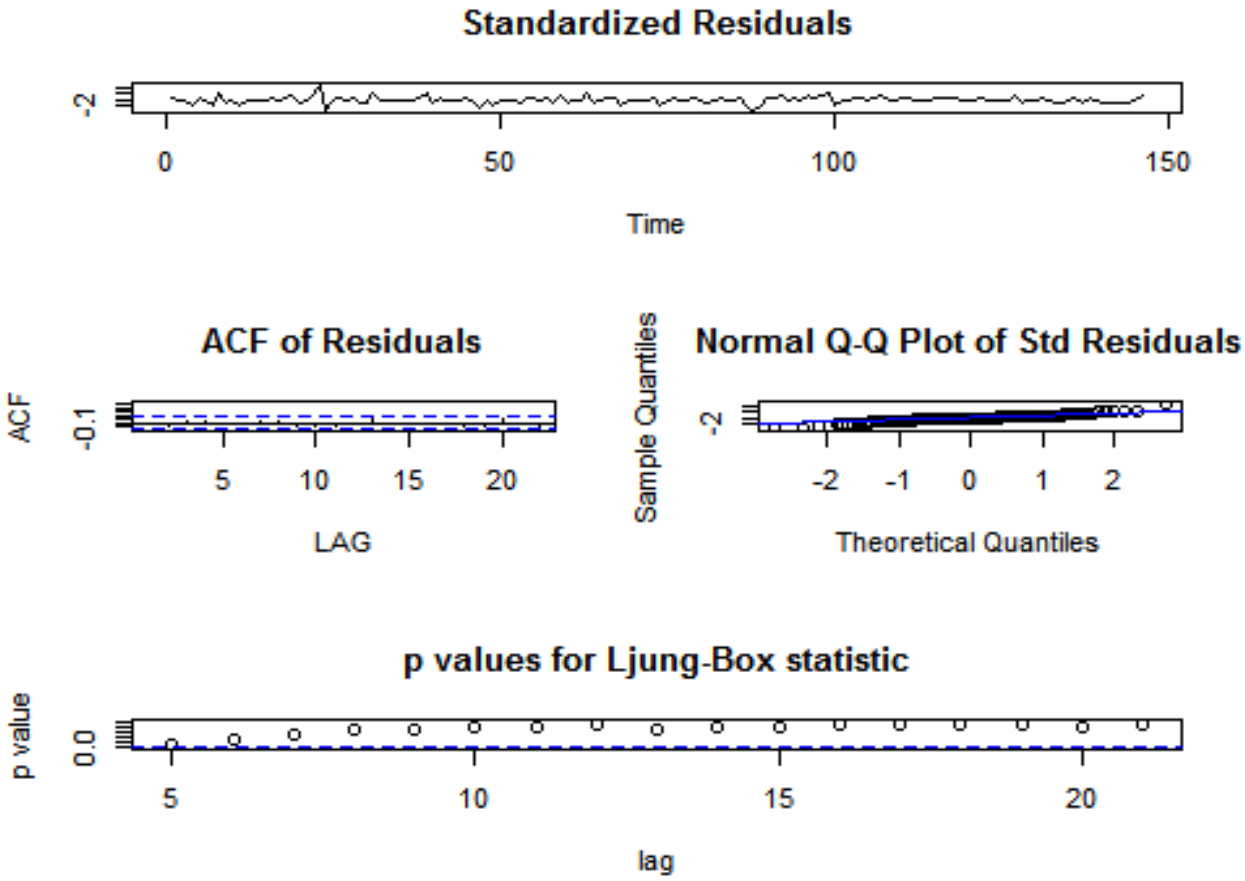
```
#16.88511, 16.90295, 15.98729
```

```
sarima(sdif_fit, 2,0,0,0,0,3,7)
```




```
## $AIC
## [1] 16.92015
##
## $AICc
## [1] 16.93941
##
## $BIC
## [1] 16.04276
```

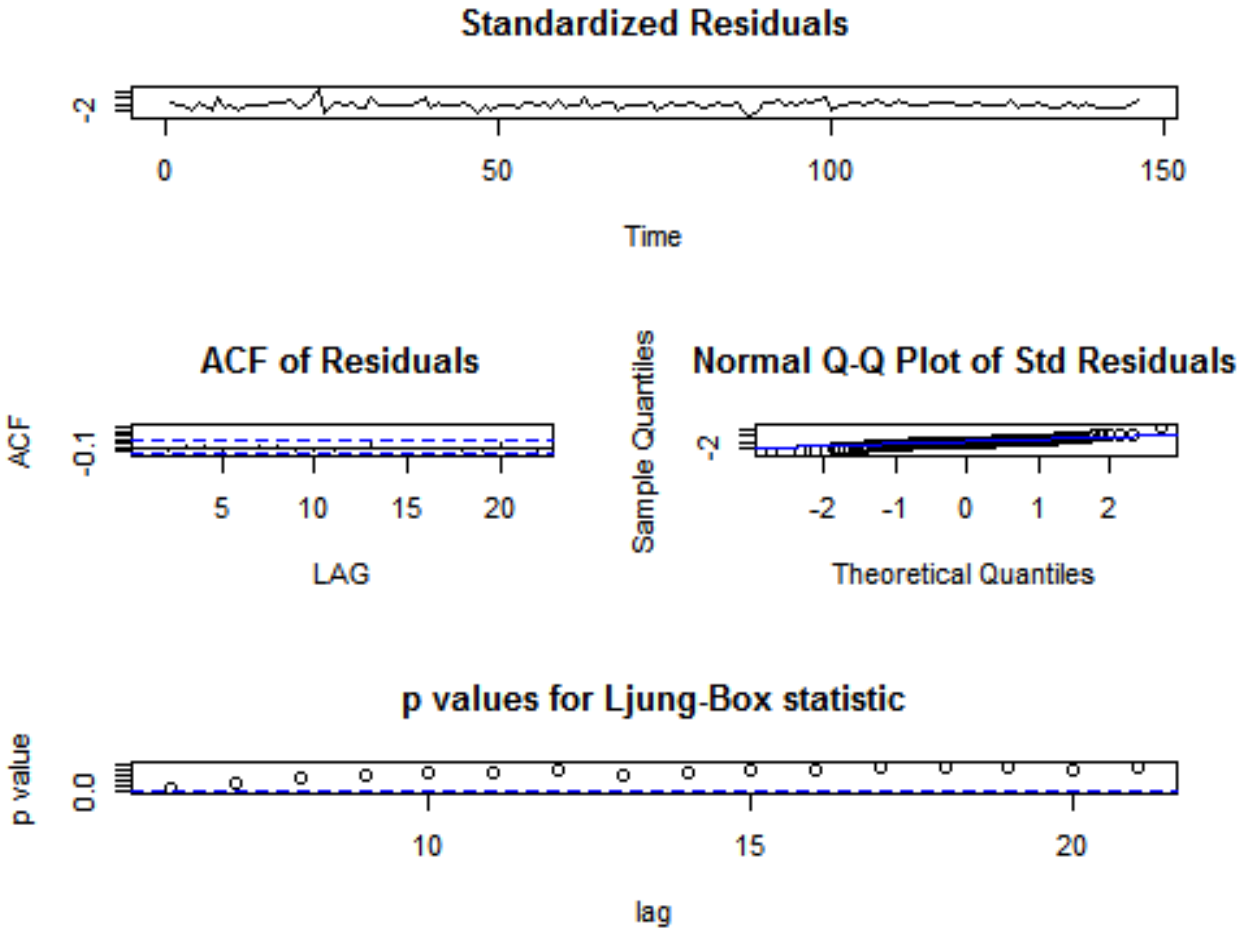
```
sarima(sdif_fit, 2,0,0,1,0,1,7)
```




```
## $AIC
## [1] 16.88651
##
## $AICc
## [1] 16.90435
##
## $BIC
## [1] 15.98869
```

```
#16.88651, 16.90435, 15.98869
```

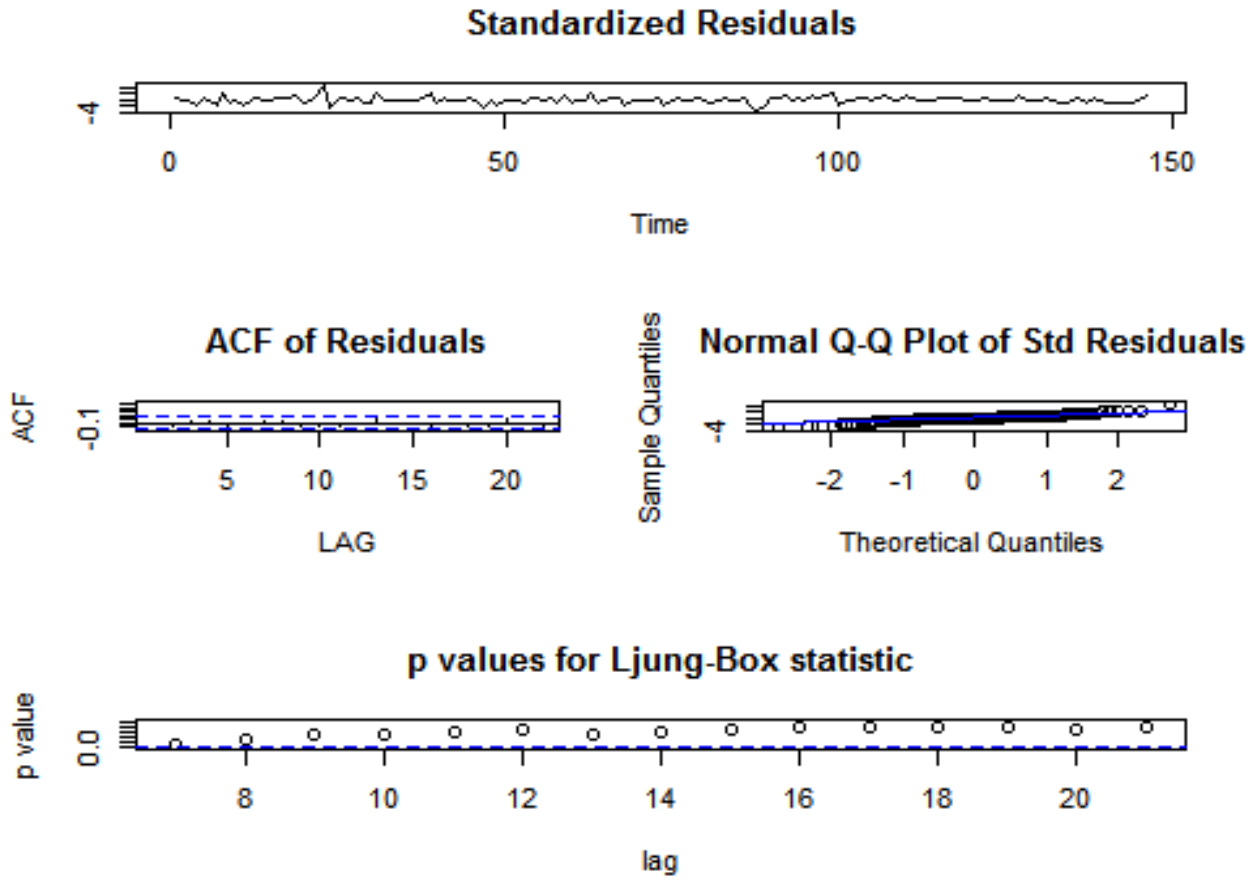
```r
sarima(sdif_fit, 2,0,0,1,0,2,7)
```




```
## $AIC
## [1] 16.89746
##
## $AICc
## [1] 16.91671
##
## $BIC
## [1] 16.02007
```

```r
#16.89746, 16.91671, 16.02007
```

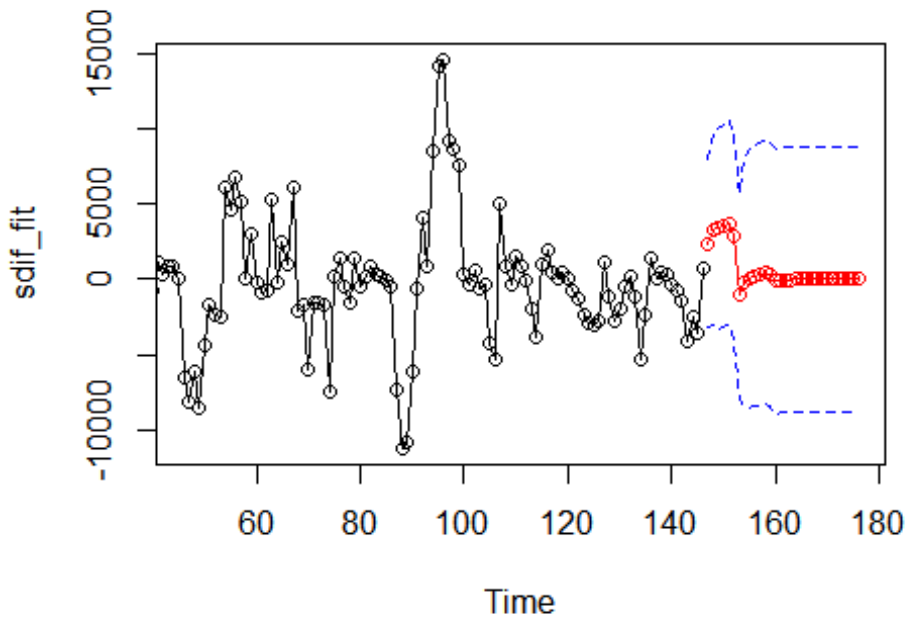
```
sarima(sdif_fit, 2,0,0,1,0,3,7)
```




```
## $AIC
## [1] 16.94464
##
## $AICc
## [1] 16.96554
##
## $BIC
## [1] 16.08769
```

```
#16.94464, 16.96554, 16.08769

# By comparing values here, the lowest values were focused on
# arima(2,0,0) and sarima(0,0,2)
```

```r
# So this is the model that I am going to fit now for estimation.
sarima.for(sdif_fit, 30, 2,0,0,0,0,2,7)
```

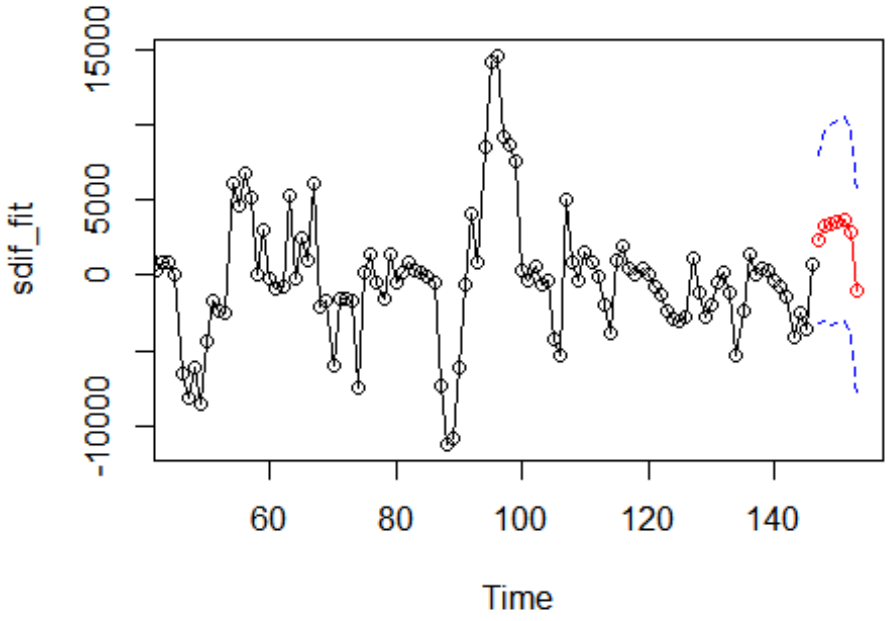



```
## $pred
## Time Series:
## Start = 147
## End = 176
## Frequency = 1
##  [1]  2367.1457679  3317.2924619  3409.2284033  3525.6352200  3710.6107160
##  [6]  2835.8589619 -1036.7750934  -270.6583901    59.9001601   222.0438081
## [11]   361.4950381   408.2020397   350.8559183  -142.0359817   -64.7471906
## [16]   -32.5741369   -12.7367848    -0.8586452     6.2667760    10.5406534
## [21]    13.1041738    14.6418019    15.5640884    16.1172860    16.4491002
## [26]    16.6481260    16.7675039    16.8391081    16.8820571    16.9078184
##
## $se
## Time Series:
## Start = 147
## End = 176
## Frequency = 1
##  [1] 2782.510 3189.235 3324.870 3372.298 3389.171 3395.141 3397.065
##  [8] 4080.798 4273.078 4340.816 4364.915 4373.553 4376.656 4377.769
## [15] 4393.270 4398.257 4400.063 4400.712 4400.946 4401.030 4401.060
## [22] 4401.071 4401.075 4401.076 4401.077 4401.077 4401.077 4401.077
## [29] 4401.077 4401.077
```

```r
# The above estimation also gave us the beginning parts, but then
# again, we could not estimate thereafter. Let's try the same method
# used above.

sarima.for(sdif_fit, 7, 2,0,0,0,0,2,7)$pred
```

```
## Time Series:
## Start = 147
## End = 153
## Frequency = 1
## [1]  2367.146  3317.292  3409.228  3525.635  3710.611  2835.859 -1036.775
```

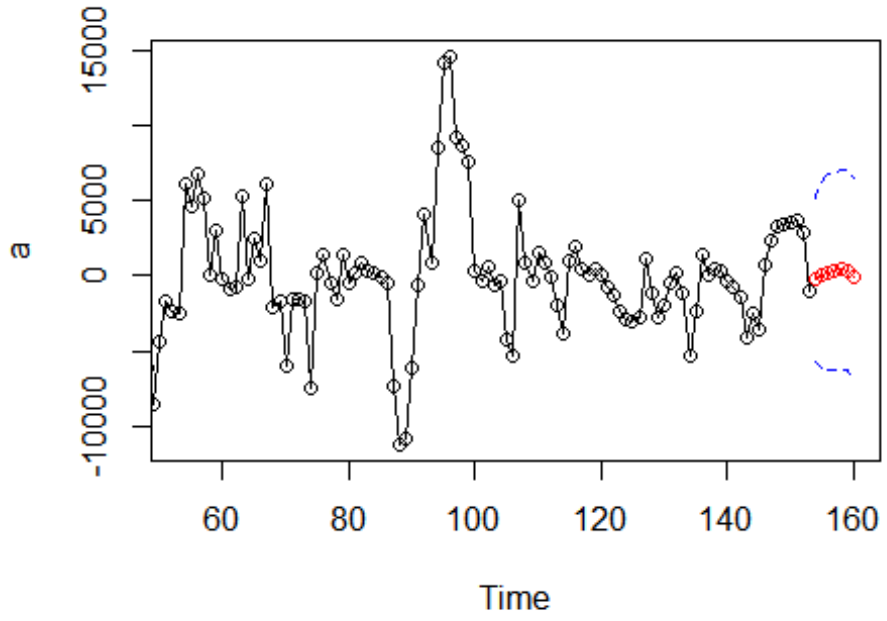
```r
a=c(sdif_fit, sarima.for(sdif_fit, 7, 2,0,0,0,0,2,7)$pred)
```



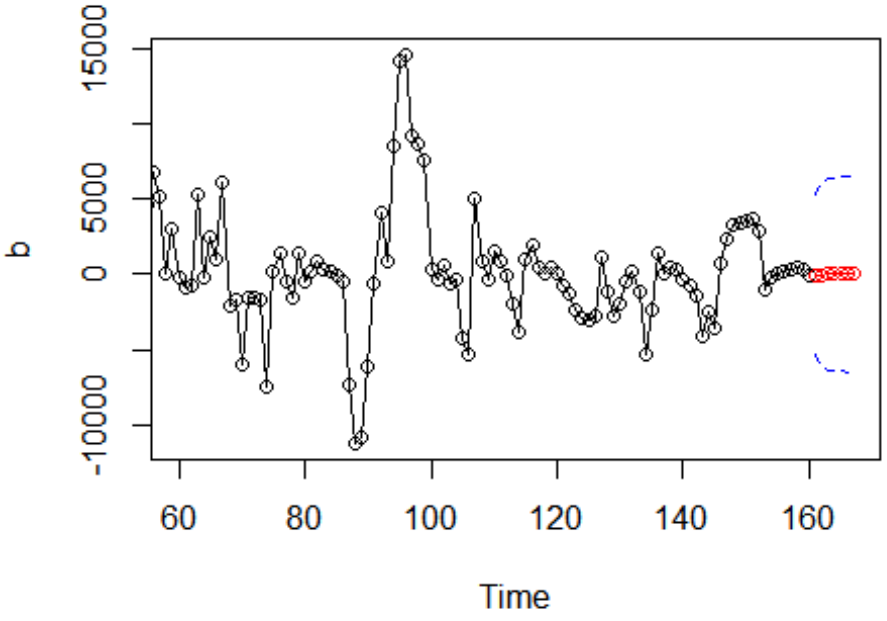

```r
sarima.for(a, 7, 2,0,0,0,0,2,7)$pred
```

```
## Time Series:
## Start = 154
## End = 160
## Frequency = 1
## [1] -271.18979   59.90075  221.97822  361.19651  407.50349  350.50917
## [7] -141.38480
```

```r
b=c(a, sarima.for(a, 7, 2,0,0,0,0,2,7)$pred)
```



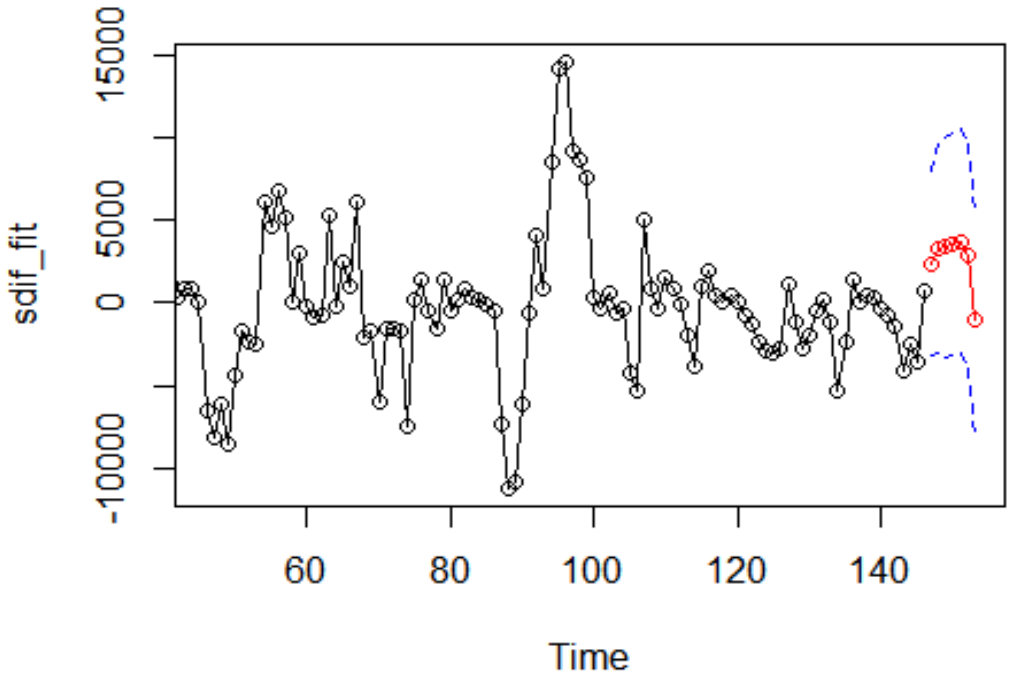

```
sarima.for(b, 7, 2,0,0,0,0,2,7)
```




```
## $pred
## Time Series:
```

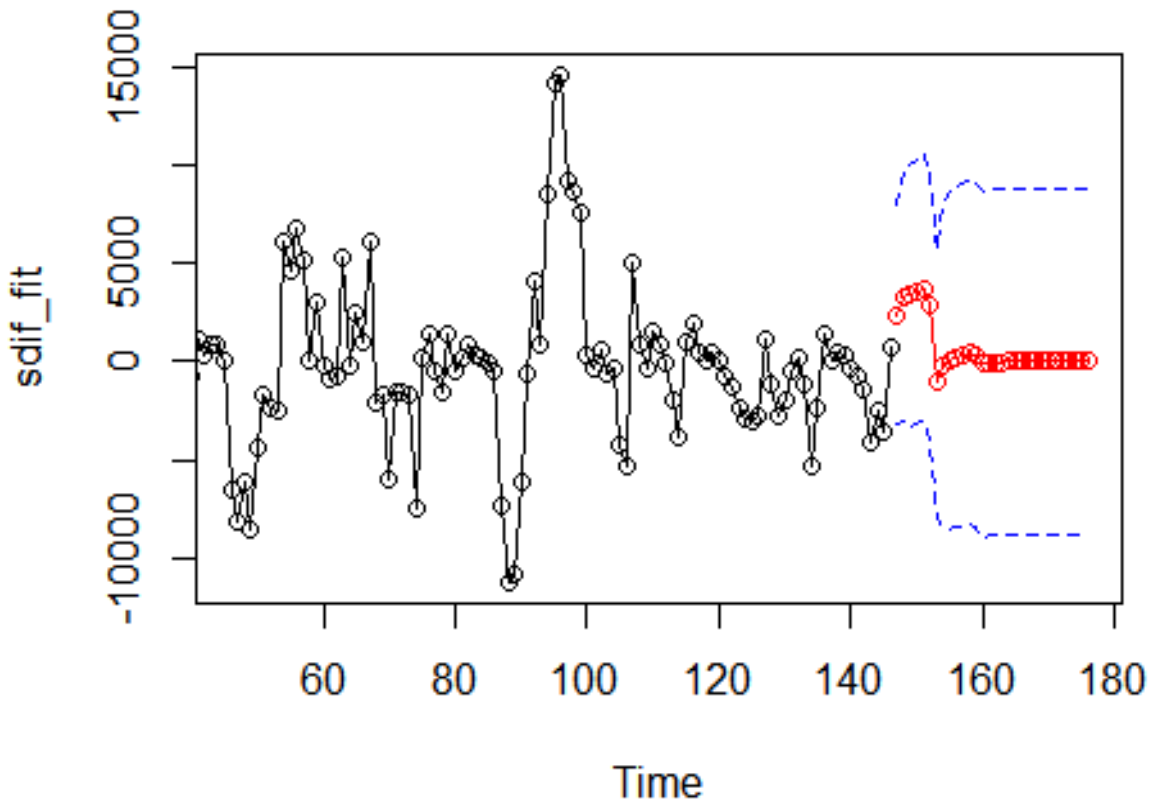
```
## Start = 161
## End = 167
## Frequency = 1
## [1] -64.025633 -31.054120 -11.167959   0.821056   7.451094  11.814714
## [7]  13.427507
##
## $se
## Time Series:
## Start = 161
## End = 167
## Frequency = 1
## [1] 2652.620 3041.577 3171.295 3216.694 3232.863 3238.600 3240.484
```

```r
# stop doing it. It results the same as 30 day prediction.

# Take a look at first seasonal prediction.
sarima.for(sdif_fit, 7, 2,0,0,0,0,2,7)$pred
```

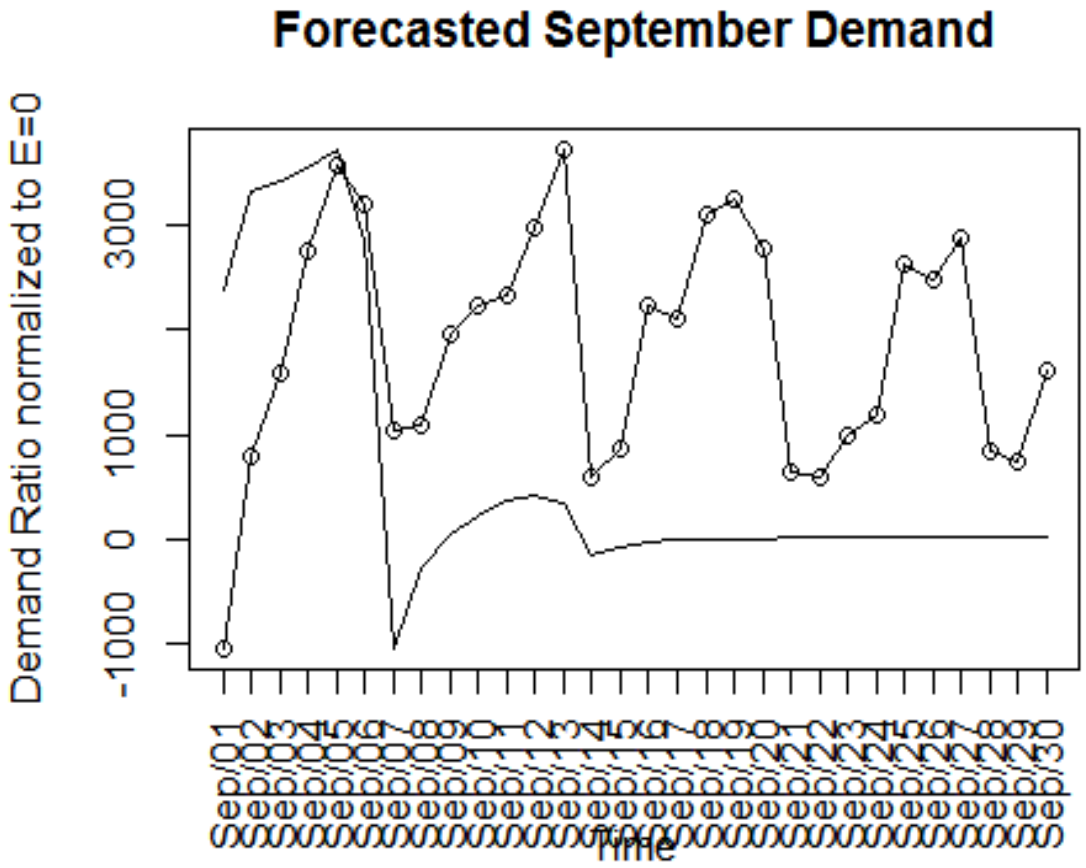



```
## Time Series:
## Start = 147
## End = 153
## Frequency = 1
## [1]  2367.146  3317.292  3409.228  3525.635  3710.611  2835.859 -1036.775
```

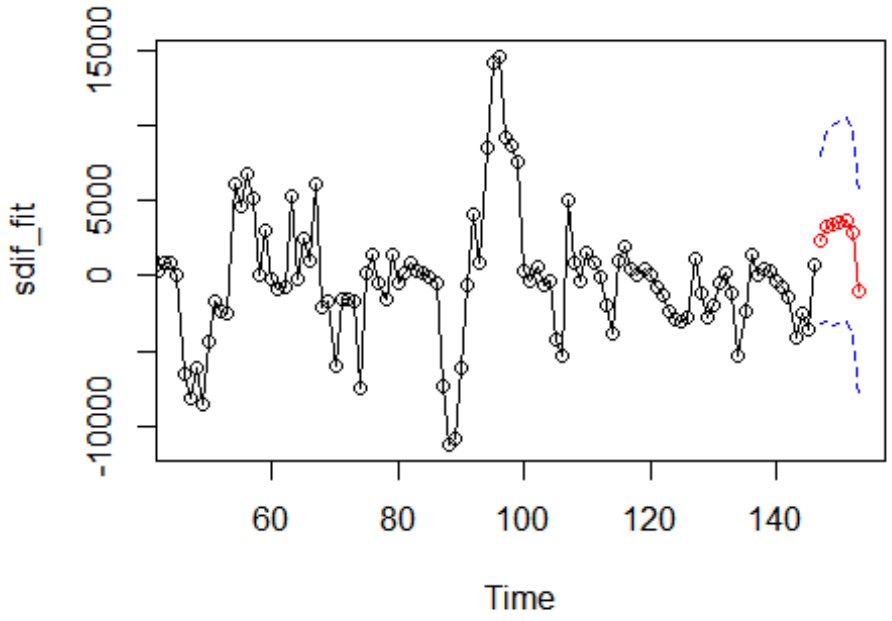
```r
# Recall September's data to see if the prediction is correct
# plot detrended september's and prediction

plot(sarima.for(sdif_fit, 30, 2,0,0,0,0,2,7)$pred, main="Forecasted September
Demand", xaxt='n', ylab='Demand Ratio normalized to E=0')
```




```r
x=seq(as.Date("2014/9/1"), as.Date("2014/9/30"), "days")
axis(1, at=147:176, labels=format(x, '%b/%d'), las=2, font=0.5)
par(new=TRUE) # To put actual september data
plot(resid(lm(SEP14DATA~time(SEP14DATA))), axes = FALSE, xlab = "", ylab = ""
, type='o')
```



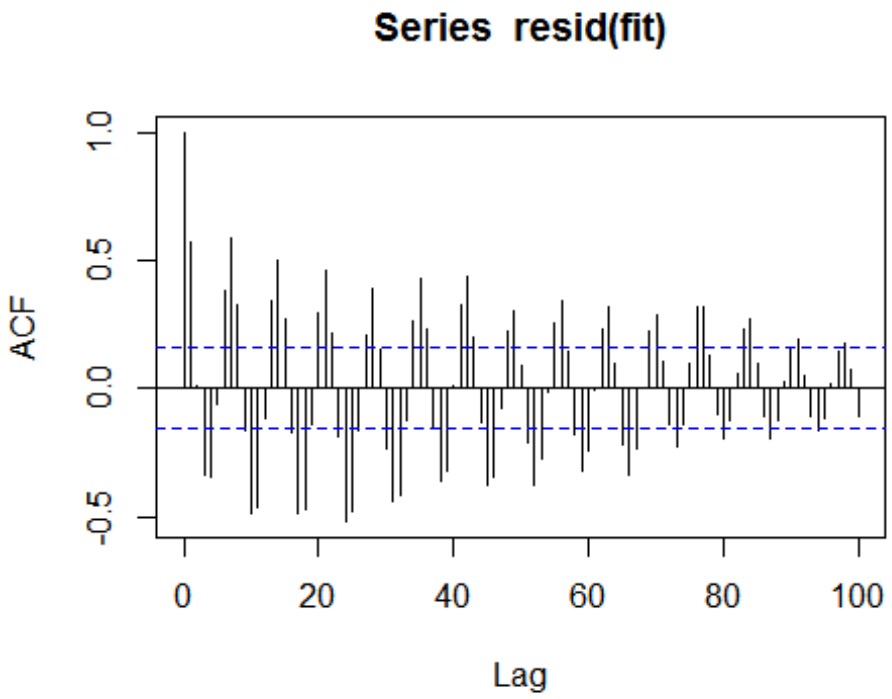

```
# As we can see here, the prediction that SEP 5th is going to have the
# highest demand in the first week is CORRECT.
# This is definitely better than non-seasonal differenced analysis.
# September 5th is Friday, and even though the second week's prediction
# is not as good as first week's, IT DOES have the peak and that peak
# shows that in the second week, SEP 12, 13th will likely to have highest
# demand (which is Friday and Saturday), and this prediction is correct again
.

# Also, thereafter, the variances decrease and the predictions converge.
# recall again

sarima.for(sdif_fit, 7, 2,0,0,0,0,2,7)$pred
```



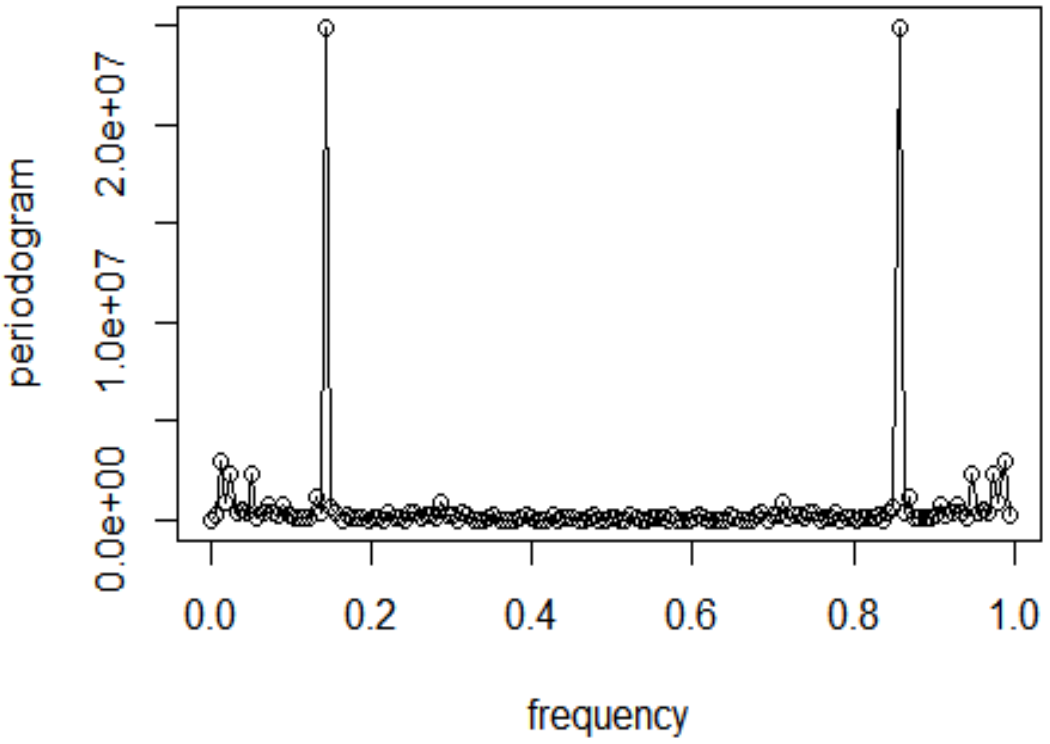

```
## Time Series:
## Start = 147
## End = 153
## Frequency = 1
## [1]  2367.146  3317.292  3409.228  3525.635  3710.611  2835.859 -1036.775
```

```
# We can see that in APR through July, high level of variances can be detecte
d
# but as the months get into July and August, the seasonaly differenced data
# tells us that the variance (or people's randomness in order uber) is
# decreasing. So, again we can probably think that the first week or second w
eek's
# demand peaks are going to be high. [Here, SEPTEMBER 5TH]

# What we have gained so far is that
# 1) SEP 5th and 12th(13th) will likely to have highest demands in those week
s.
# 2) They are fridays (and 13th -> saturday), so we can see that people are l
ikely
#    to order Uber on weekedns in September.
# 3) Variances decrease from April to August, so during September, beginning
weekends
#    will likely to have higher demand than those after.

# Let's confirm or get another results through spectral analysis.
# Let's see if we can find the general periodicity instead of looking at seas
onal difference
```

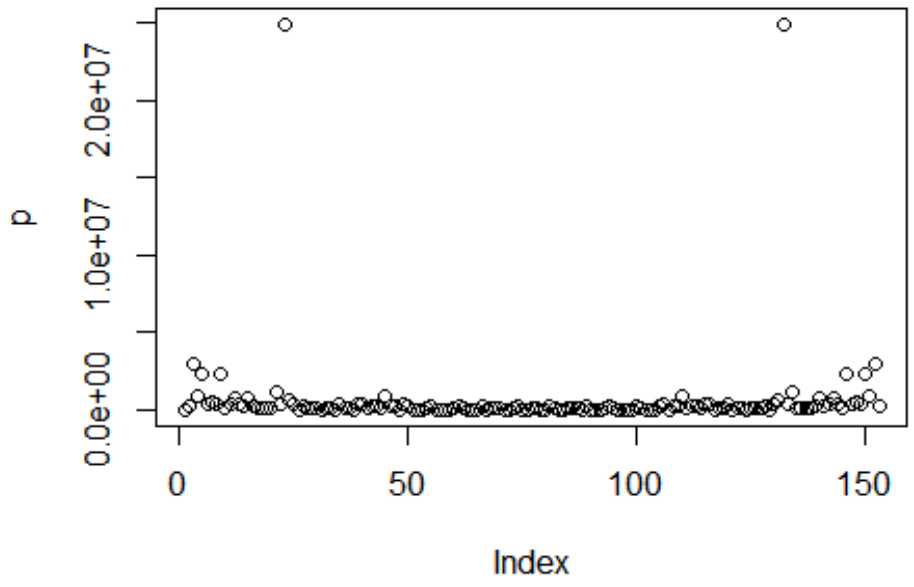
```
### SPECTRAL ANALYSIS ###

# I believe spectral analysis and estimation would be
# very good for this data because I am trying to look for where the peaks
# are going to be for entire month using previous 5 months data,
# and that is all about examining periodic cycles and
# analysis of variances. (How much it will increase, which potentially yields
peak)

# It would be good start to use linear regression to discover a signal
# in noise but we do not know the frequency.
# So, use non-linear regression with frequency as a parameter.

# Let's find frequency through using scaled periodogram.

# We still use detrended data as spectral analysis requires non-trended -> re
sid(fit)
# because if not, the spectral analysis might take a half cycle(possibly from
trend) into consideration

# Let's see if the ACF satisfies the condition that the sum of all the AC are
# bounded (ACF should look like it converges)

acf(resid(fit),100)
```



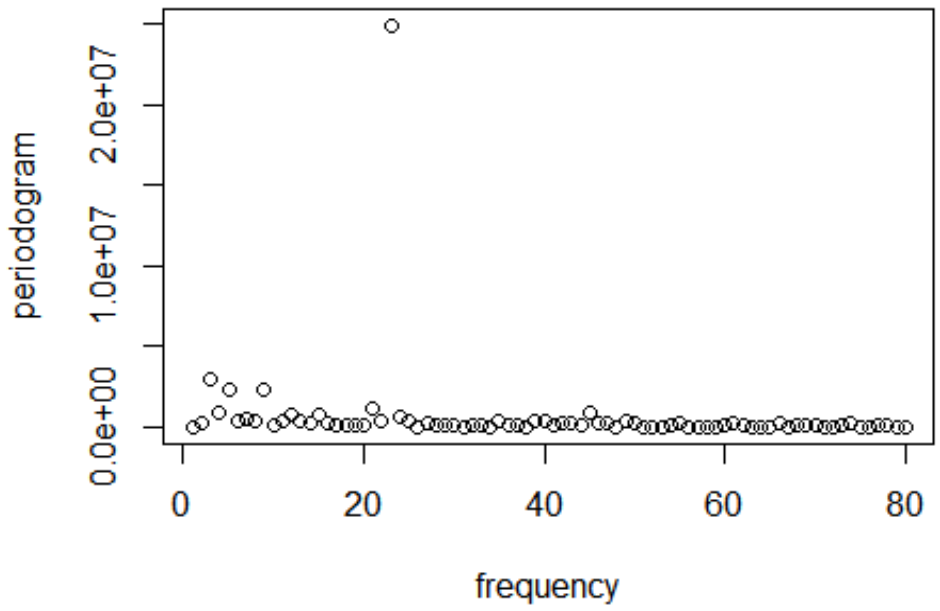
```r
# And yes, ACF does looks like it converges. Spectral density analysis is pos
sible.

# Use periodogram to estimate the spectral density using DFT and squaring it.

p=abs(2*fft(resid(fit)/(153)))^2 # by the definitions
Fr=(0:152)/153
plot(Fr, p, type="o", xlab="frequency", ylab="periodogram")
```




```r
# we can see a significant peak at around 0.15, and minor peak around 0.05
# We can also count only the peaks upto 0.5 because this periodogram is symme
tric
# and definitions and properties of periodogram (because of cosine property)
# peaks at around 0.15 and 0.85 can be assumed equal.

plot(p)
```




```r
plot(head(p,80), xlab="frequency", ylab="periodogram")
```



```r
# Also check by max functions
m1=max(head(p,5)) # to get the n between 0 and 5 (for second major frequency)
m2=max(p[11:length(p)/2])
which(p==m1)
```

```
## 3
## 3
```

```r
which(p==m2)
```

```
## 23
## 23
```

```r
# result shows that at n=3 and 23 will have the frequency level.
# One thing to note here is that at n=3, the periodogram value is not that
# different from those at n=5 and 8. So, I might take these into accounts
# later when I fit the model to find periodicity.

# Let's look at frequency now.

c(Fr[3],Fr[23]) # this is w in (1/w) form where (1/w) is the actual frequency
.
```

```
## [1] 0.0130719 0.1437908
```

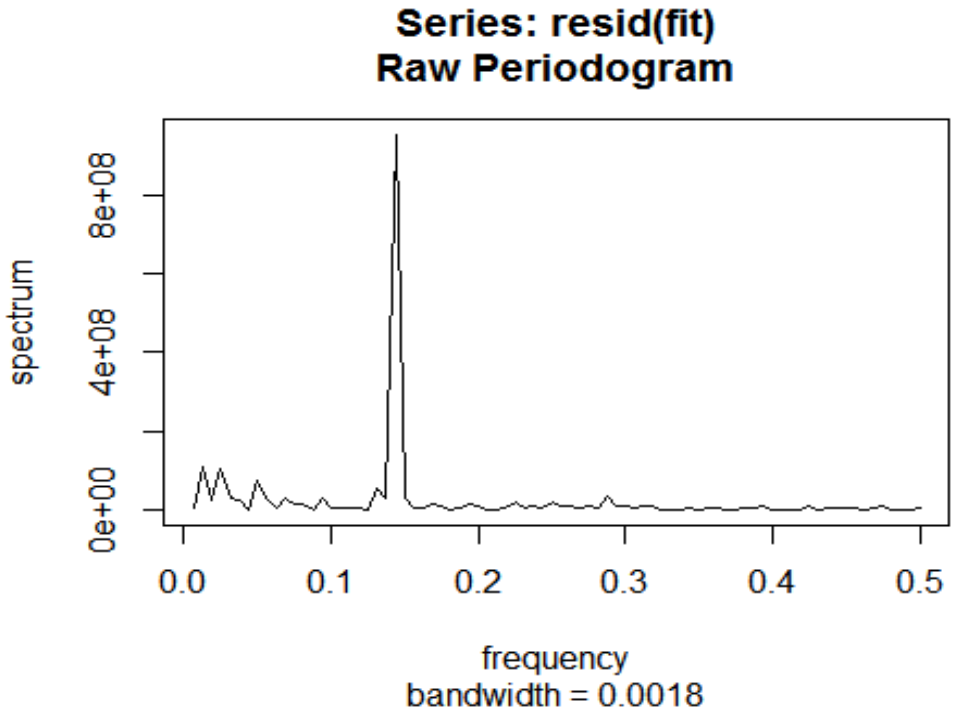
```r
# Therefore the estimated frequency level in this data is
1/c(Fr[3],Fr[23])
```

```
## [1] 76.500000  6.954545
```

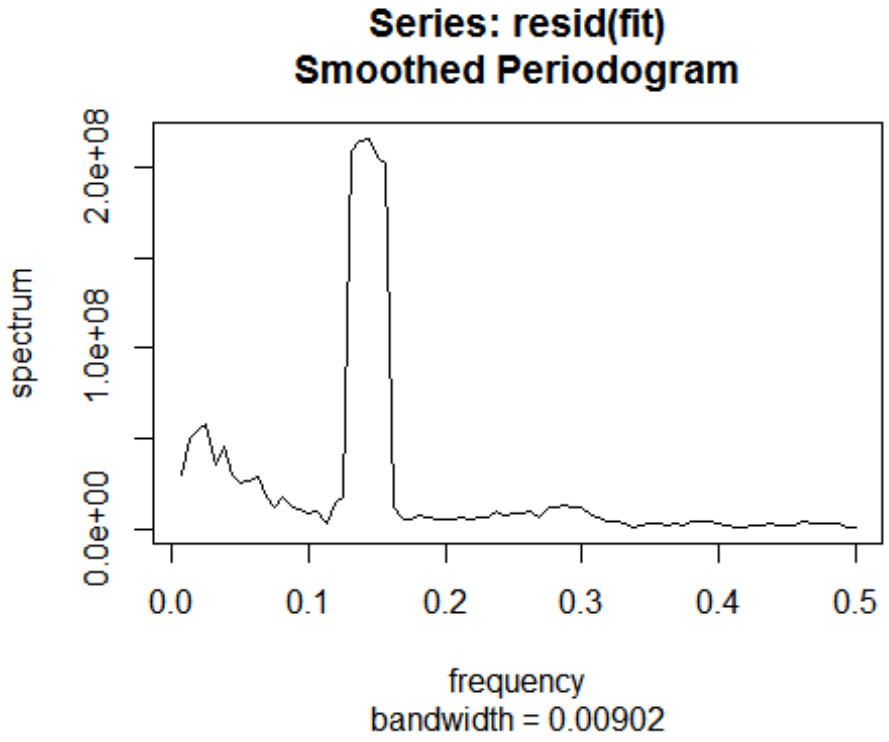
```r
# Which is what we would expect, around 7 (Weekly). But about that
# 76.50000 --> meaning there is some kind of periodicity for 2 months and a h
alf.
# I am going to try putting in n=3,n=5,n=7 components in the final analysis.

# so let's also get n=5, n=7 frequencies. [Disregard these values until the l
ast analysis]
1/c(Fr[5], Fr[7])
```

```
## [1] 38.25 25.50
```

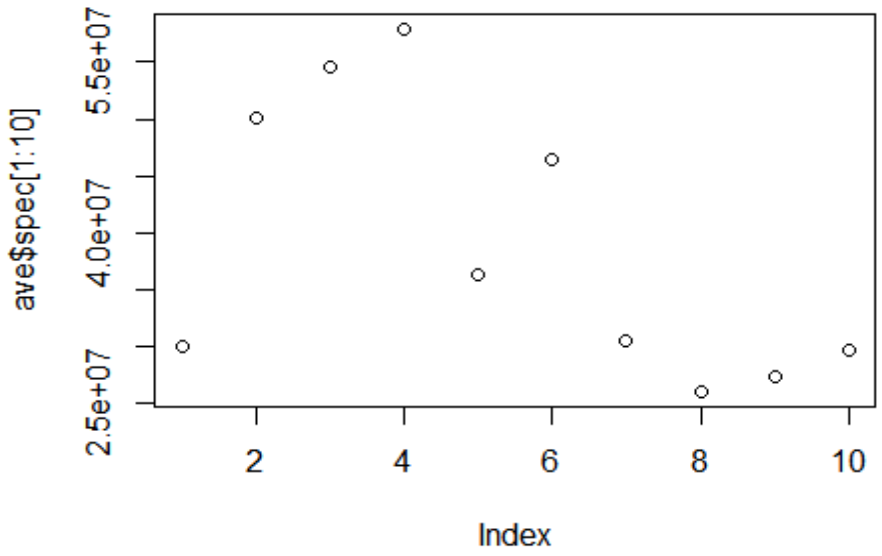
```r
# 38.25 and 25.50. This can be interpreted as there is some periodicity
# for uber orders at the end of month's beginning week(38.25) and
# at the start of month's ending week(25.50)


# We can also use build-in R functino to conduct the same analysis
spec.pgram(resid(fit), taper=0, log="no")
```



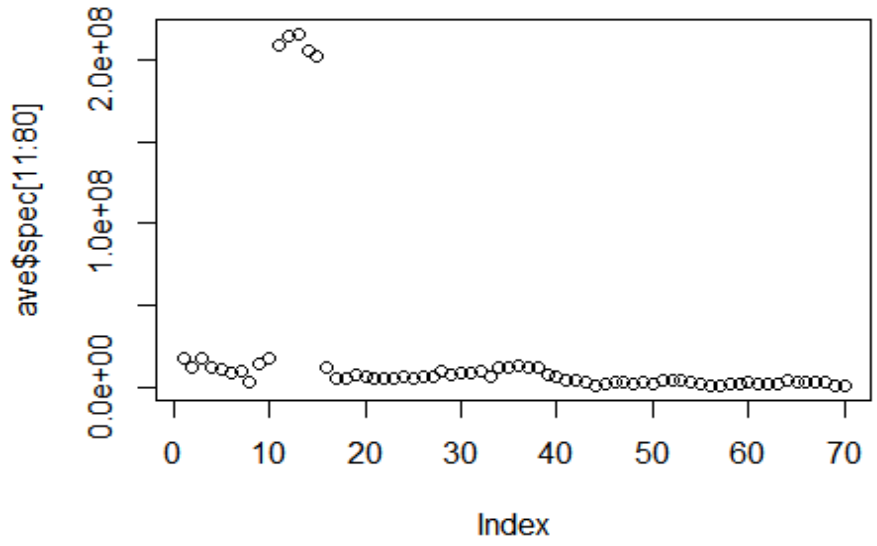

```r
# Let's actually smooth periodogram and see what happens to the small
# peaks around for n=0~5 range.[Came out n=3 when manually done]
k=kernel("daniell", 2) # use danielle kernel
ave=spec.pgram(resid(fit), k, taper=0, log='no')
```



```r
# to draw vertical lines to signify where the frequency would be
plot(ave$spec[1:10])
```




```r
which(ave$spec[1:10]==max(ave$spec[1:10]))
## [1] 4
#4th n
plot(ave$spec[11:80])
```



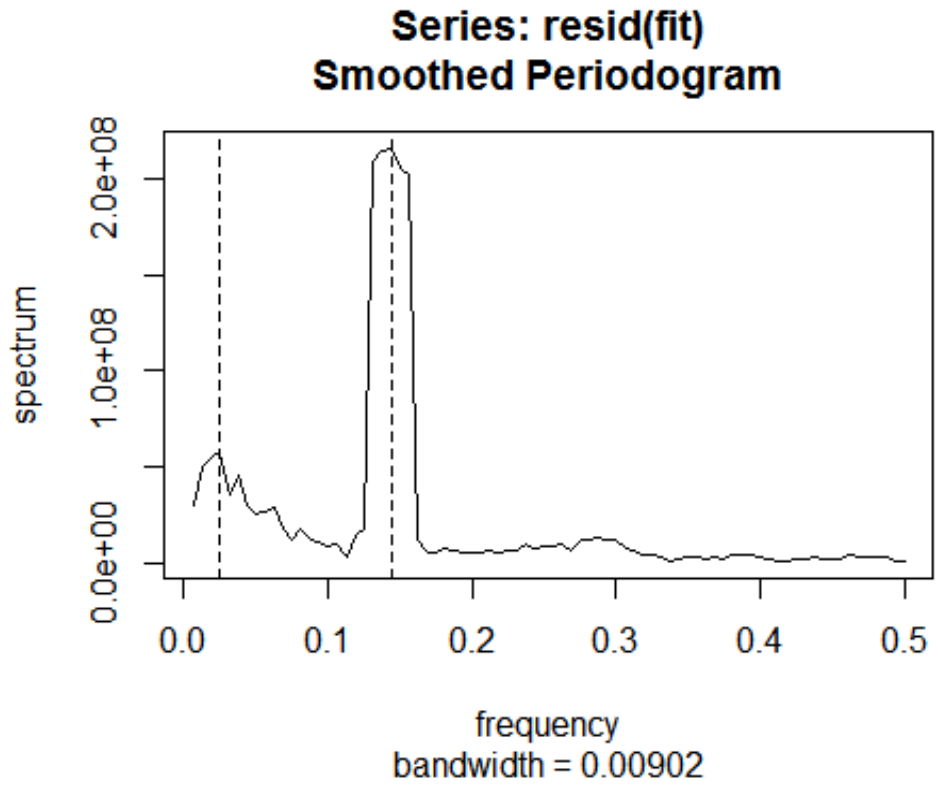
```r
which(ave$spec[11:80]==max(ave$spec[11:80]))
```

```
## [1] 13
```

```r
#10+13 = 23rd n

# The estimated frequencies are
1/ave$freq[c(4,23)]
```

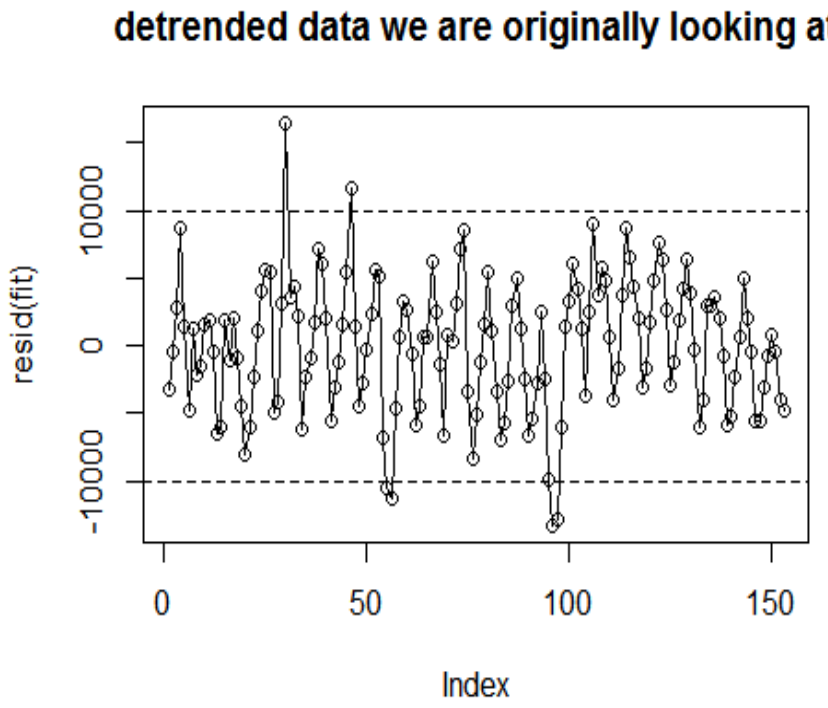
```
## [1] 40.000000  6.956522
```

```r
# So this is basically saying that people do usually
# order Ubers (almost) monthly (very little periodicity but still there is)
# as I mentioned above, since
# Still, we can conclude weekly for sure.

# 4:80 = x:.5
v1=4*0.5/80
# 23:80 = x:.5
v2=23*0.5/80

ave=spec.pgram(resid(fit), k, taper=0, log='no')
abline(v=c(v1,v2), lty=2)
```




```r
### So, using spectral analysis, we found out frequencies and general periodi
city.
```

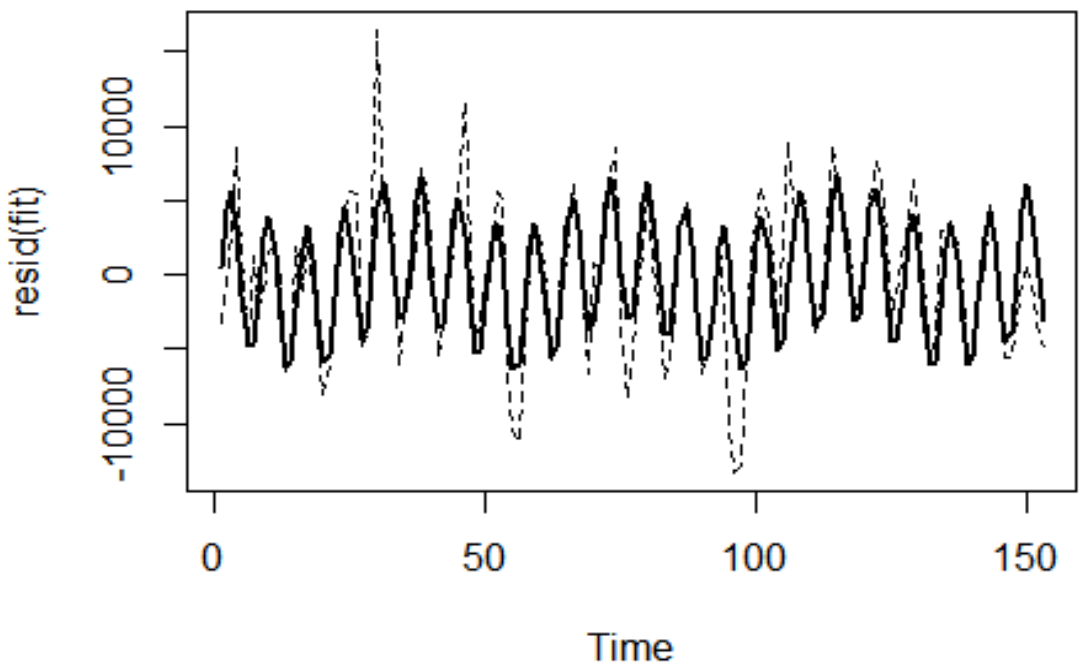
```r
# Let's use regression to discover the signal with known frequency.
plot(resid(fit),type="o", main="detrended data we are originally looking at")
# We know by looking at the detrended data, the upward and downward do not go
# beyond, mostly, 10000 for each in magnitude. So, A should be around 10000
abline(h=c(-10000,10000), lty=2)
```




```r
# We know that w=1/7 and 1/40 [REMEMBER THAT 1/40 was obtained through SMOOTH
ING]
# [USE 1/38.25, 1/25.50, 1/76.5, 1/6.95 as well for non-smoothed case]

# FIRST, SMOOTHED periodo.
z1=cos(2*pi*1:length(resid(fit))/7)
z2=sin(2*pi*1:length(resid(fit))/7)
z3=cos(2*pi*1:length(resid(fit))/40)
z4=sin(2*pi*1:length(resid(fit))/40)

summary(finalfit <- lm(resid(fit)~0+z1+z2+z3+z4))

##
## Call:
## lm(formula = resid(fit) ~ 0 + z1 + z2 + z3 + z4)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11040.5  -1842.4     54.5   1796.6  11976.8
##
```

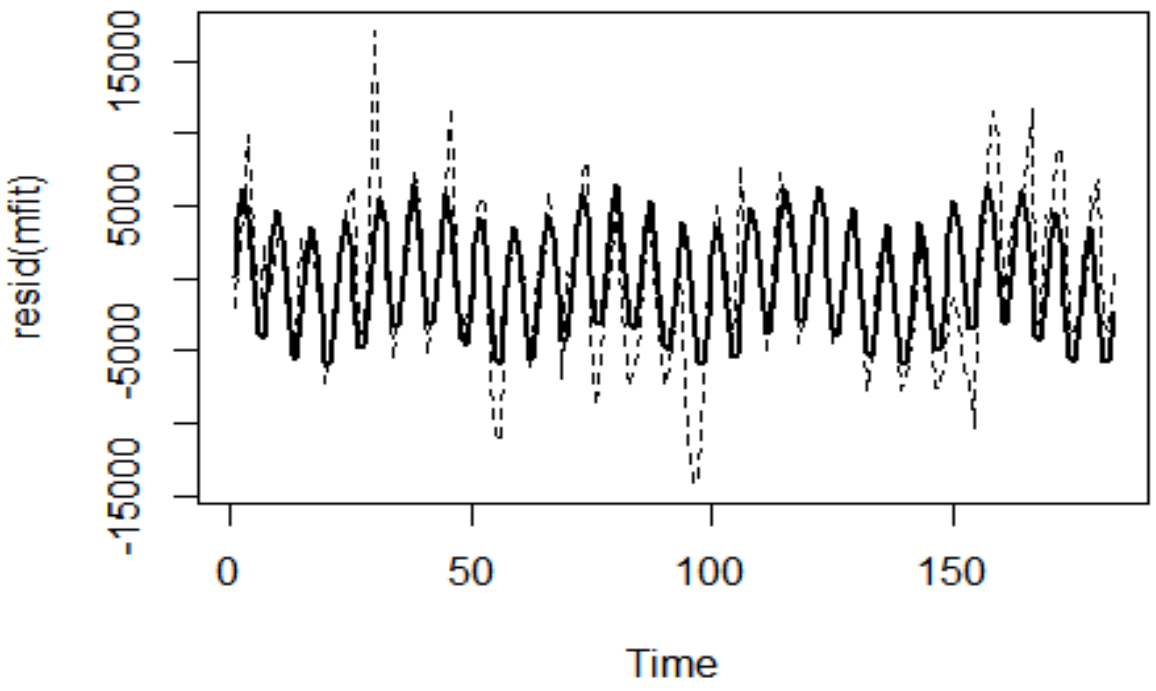
```
## Coefficients:
##    Estimate Std. Error t value Pr(>|t|)
## z1  -4273.4      367.2 -11.639  < 2e-16 ***
## z2   2527.4      364.8   6.928  1.2e-10 ***
## z3   1293.3      370.2   3.493 0.000628 ***
## z4  -1094.3      362.2  -3.021 0.002965 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3201 on 149 degrees of freedom
## Multiple R-squared:  0.579,  Adjusted R-squared:  0.5677
## F-statistic: 51.24 on 4 and 149 DF,  p-value: < 2.2e-16
```

```
plot.ts(resid(fit), lty="dashed")
lines(fitted(finalfit), lwd=2)
```



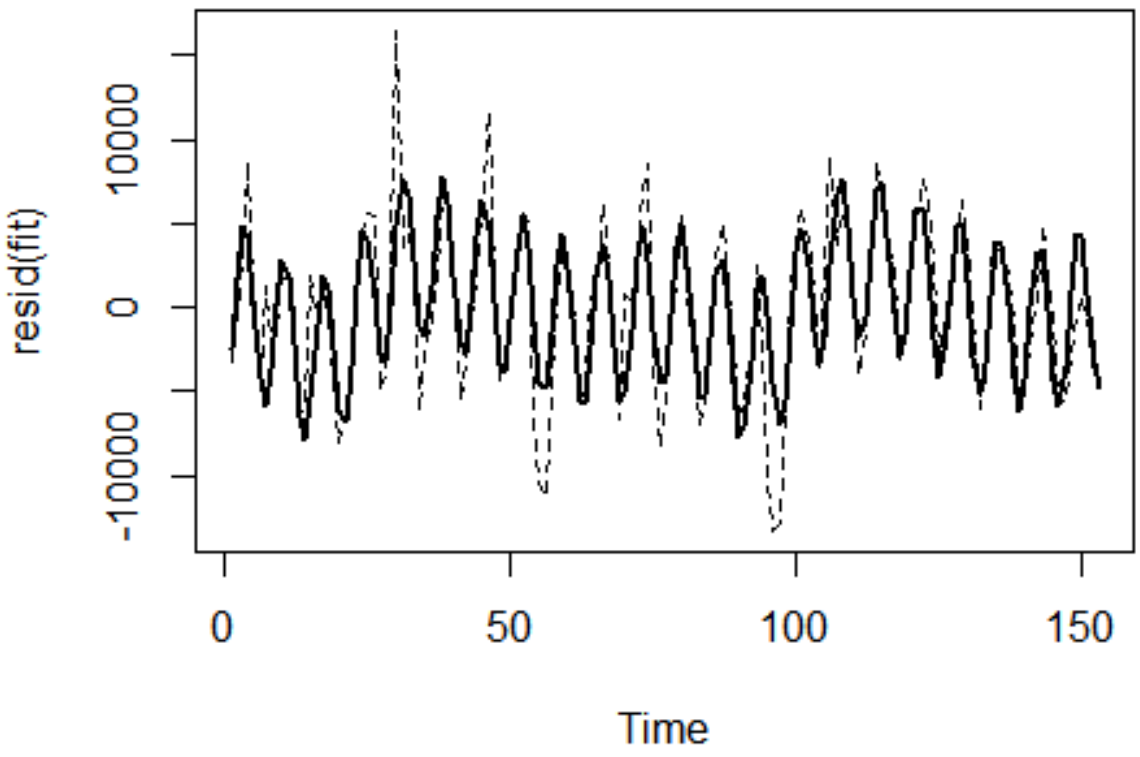

```
# merge original data with september's data to see if prediction is corrrect.
APRtoSEP14=c(APRtoAUG14, SEP14DATA)
mfit=lm(APRtoSEP14~time(APRtoSEP14))
z1=cos(2*pi*1:length(resid(mfit))/7)
z2=sin(2*pi*1:length(resid(mfit))/7)
z3=cos(2*pi*1:length(resid(mfit))/40)
z4=sin(2*pi*1:length(resid(mfit))/40)
pmfit=lm(resid(mfit)~0+z1+z2+z3+z4)
```

```r
plot.ts(resid(mfit), lty="dashed")
lines(fitted(pmfit), lwd=2)
```

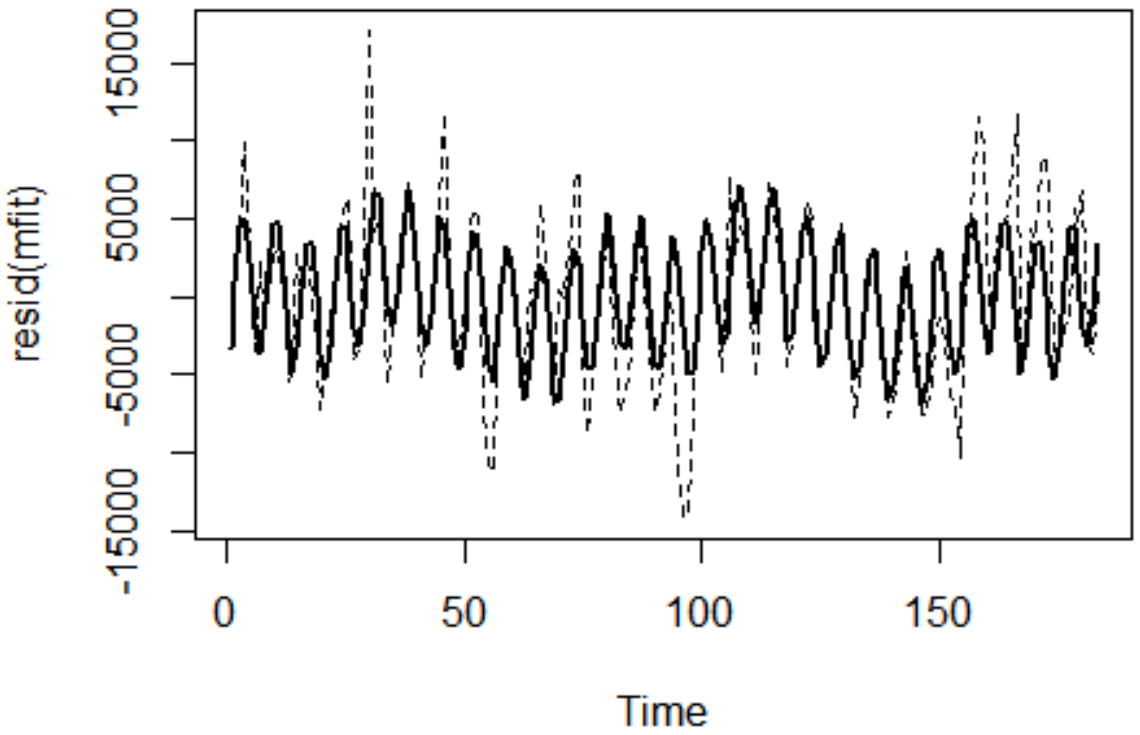



```r
# THE RESULT IS GOOD in terms of predicing that first and second week's deman
d
# is going to be higher than 3rd and 4th
# (the amount of how-higher-prediction is good too)

# Although I am satisfied with the result above,
# [for most part, estimates do tell us when the demand is going to be high]
# if we look at around n=30, there is a immense peak that is estimated
# lower than the next week's high demand day, which could be awful
# If this didn't happen due to holiday, then my assumption is wrong to
# not take those into consideration. So, let's try without smoothing.

# Let's recall what I mentioned about adding periodicities from periodogram.
# instead of using kernel smoothed periodogram.
# Recall values.
```

```r
noSM = 1/c(Fr[3], Fr[5], Fr[7], Fr[23])

# Put in
z1=cos(2*pi*1:length(resid(fit))/noSM[1])
z2=sin(2*pi*1:length(resid(fit))/noSM[1])
z3=cos(2*pi*1:length(resid(fit))/noSM[2])
z4=sin(2*pi*1:length(resid(fit))/noSM[2])
z5=cos(2*pi*1:length(resid(fit))/noSM[3])
z6=sin(2*pi*1:length(resid(fit))/noSM[3])
z7=cos(2*pi*1:length(resid(fit))/noSM[4])
z8=sin(2*pi*1:length(resid(fit))/noSM[4])

summary(finalfit <- lm(resid(fit)~0+z1+z2+z3+z4+z5+z6+z7+z8))

##
## Call:
## lm(formula = resid(fit) ~ 0 + z1 + z2 + z3 + z4 + z5 + z6 + z7 +
##     z8)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9520.9 -1257.3  -103.6  1602.0 11691.3
##
## Coefficients:
##    Estimate Std. Error t value Pr(>|t|)
## z1  -1656.1      338.9  -4.887 2.68e-06 ***
## z2   -466.1      338.9  -1.375   0.1711
## z3   1470.0      338.9   4.338 2.68e-05 ***
## z4   -397.0      338.9  -1.172   0.2433
## z5    405.1      338.9   1.195   0.2339
## z6    621.7      338.9   1.835   0.0686 .
## z7  -4971.4      338.9 -14.670  < 2e-16 ***
## z8    455.4      338.9   1.344   0.1811
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2964 on 145 degrees of freedom
## Multiple R-squared:  0.6487, Adjusted R-squared:  0.6293
## F-statistic: 33.47 on 8 and 145 DF,  p-value: < 2.2e-16

plot.ts(resid(fit), lty="dashed")
lines(fitted(finalfit), lwd=2)
```




```
# merge original data with september's data to see if prediction is corrrect.
APRtoSEP14=c(APRtoAUG14, SEP14DATA)
mfit=lm(APRtoSEP14~time(APRtoSEP14))
z1=cos(2*pi*1:length(resid(mfit))/noSM[1])
z2=sin(2*pi*1:length(resid(mfit))/noSM[1])
z3=cos(2*pi*1:length(resid(mfit))/noSM[2])
z4=sin(2*pi*1:length(resid(mfit))/noSM[2])
z5=cos(2*pi*1:length(resid(mfit))/noSM[3])
z6=sin(2*pi*1:length(resid(mfit))/noSM[3])
z7=cos(2*pi*1:length(resid(mfit))/noSM[4])
z8=sin(2*pi*1:length(resid(mfit))/noSM[4])

pmmfit=lm(resid(mfit)~0+z1+z2+z3+z4+z5+z6+z7+z8)


plot.ts(resid(mfit), lty="dashed")
lines(fitted(pmmfit), lwd=2)
```

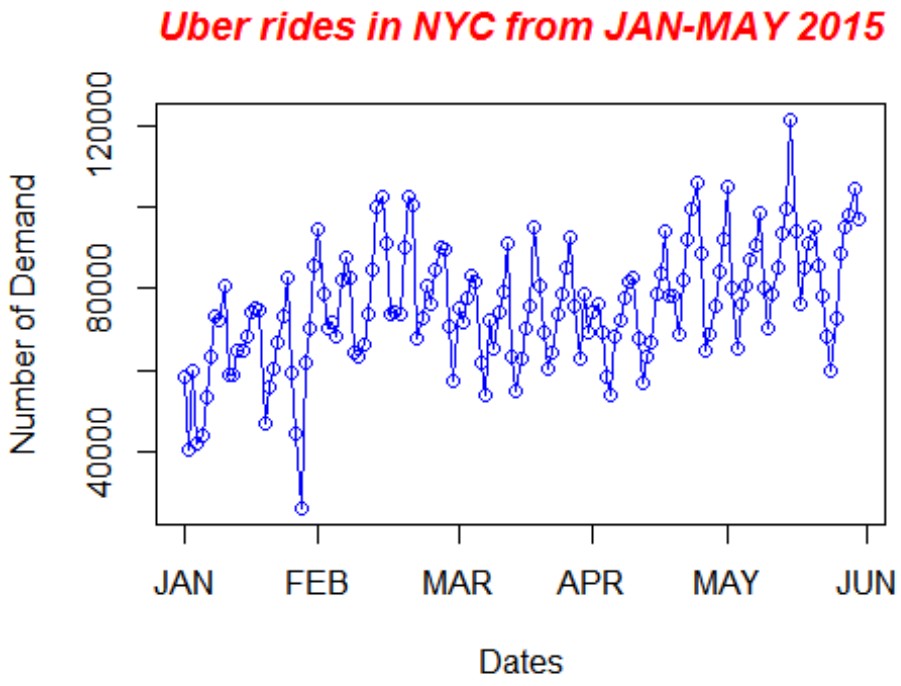



```
# THE RESULT IS WORSE than the SMOOTHED version here.
# 4th week of september's demand is predicted to be higher than
# that of 3rd week, but that is not the case.
# first week's high demand and 2nd week's high demand are still
# predicted high. [about the same magnitude]

### Conclusion ###

# through seasonal ARIMA analysis, we confirmed that
# 1) SEP 5th and 12th(13th) will likely to have highest demands in those week
s.

# through both ARIMA analysis and Spectral analysis, we confirmed that
# 2) They are fridays (and 13th -> saturday), so we can see that people are l
ikely
#    to order Uber on weekedns in September.

# through Spectral analysis, we confirmed that
# 3) Variances decrease from April to August, so during September, beginning
weekends
```

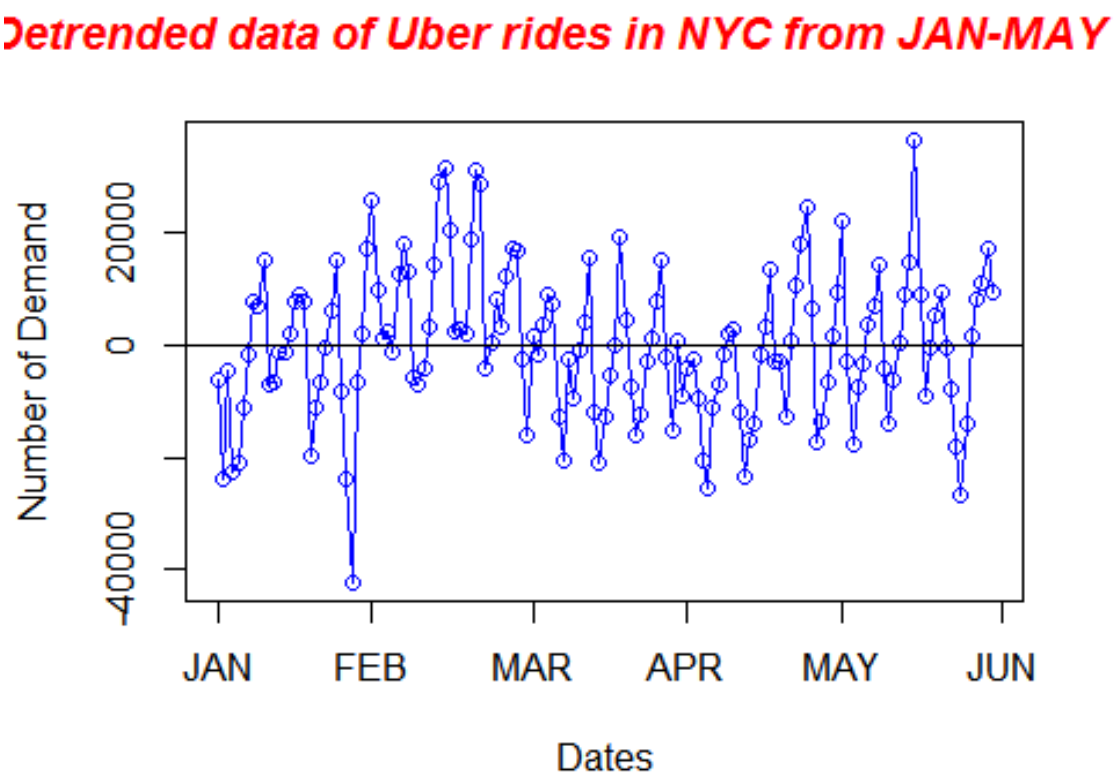
```
#    will likely to have higher demand than those after.

# We can now prepare for outrageous demand and avoid high amount of
# surge pricing for customers. --- end ---


#### Because of the company growth, it would be nice (professor tip)
#### to consider different user behavior of 2014 and 2015.

# So, let's start same analysis on 2015 (JAN through JUN)

# Source
# https://raw.githubusercontent.com/fivethirtyeight/uber-tlc-foil-response/ma
ster/uber-trip-data/uber-raw-data-janjune-15.csv.zip
# The above source is in 'zip' file, so I downloaded it.

# The raw csv file's capacity is 526.1 MB once you download it.
# my R or Excel can't open files that are over 5 MB and 30 MB respectively
# , so the follwing is what I did to get the data I want

## Unlike previous apr14~sep14 data, the dates are not in chronological order
.
## So, I wanted to call the data in EXCEL first and order it chronologically.
## Since R can only take 5MB data and EXCEL can take up to almost 30 MB (EXCE
L works in a way that it counts the number of rows and columns in determining
the capacity)

## STEP 1 : Call the file in NOTEPAD
## STEP 2 : Divide the data by using (ctrl+shift+home (or end)) from the midd
le of the entire data
## STEP 3 : Repeat this process and save every divided notepad.

# Note : The initial file capacity was 526.1 MB. So, if we take step 2, it wi
ll produce two 213 MB files.
#        Then divide those two files again into four files, each with 106 MB.
#        Then divide those four files into 8 files, each with 53 MB, and so o
n.
#        The divided capacities don't have to be exactly the same as I am man
ually 'blocking' the data and
#        dividing them. This is just to CALL THE ENTIRE DATA into EXCEL and
#        put them in the chronological order so I can use my functions to get
the number of demand each month.


## STEP 4 : every time step 2 is repeated, mark the file with new number afte
r dash
##          ex) p1-1-1-1, p1-1-1-2, p1-1-2-1-, p1-1-2-2,.... etc

##          In total, I have 16 files that I am going to work on EXCEL to ord
```

```
er the data by chronological order
##          These 16 files add up to be the same capacity as the original. Go
od. I will attach these as a zipped file.

##          The files are still more than 30.0MB and can't handle the data in
R, so I will play around with it in EXCEL using countif function.
##          And get the demand number.

### IN EXCEL ###

# STEP 1: Use Find and Replace function to get rid the data other than place
and date.

# For example, original format -> B02682,2015-06-18 10:18:00,B02682,114
#            , After STEP 1    -> B02682,2015-06-18 10

# STEP 2: Now get rid of first unnecessary data (Uber Code for regions)
# Function used in excel : =RIGHT(A2, LEN(A2)-7)
# For example, original format -> B02598,2015-01-01 00
#            , After STEP 2    -> 2015-01-01 00

# Step 3: Now get rid of last 2 digits
# Function used in excel : =LEFT(D2, LEN(D2)-3)  ## It's 3 here to count the
space.
# For example, original format -> 2015-01-01 00
#            , After Step 3    -> 2015-01-01

# Step 4: Make it into a date format. Currently, it's just number with dash.
# Function used in excel : =DATE(LEFT(F2,4),MID(F2,6,2),RIGHT(F2,2))
# For example, original format -> 2015-01-01 (NUMBERS. THESE DON'T MEAN ANYTH
ING YET)
#            , After Step 4    -> 1/1/2015 (IN DATE FORMAT)

# Step 5: We can do chronological ordering and move it to R and use functions
that I made.
#         Though, I will just use COUNTIF function to get the actual demand r
ight here.
# for example, =COUNTIF(H:H,"1/1/2015") command gives me (in p-1-1-1 file) 11
171
#            , =COUNTIF(H:H,"1/1/2015") gives me 6733
#            , continues until =COUNTIF(H:H, "6/31/2015")
# This might seem like a lot of work, but with copy and paste, we can do it f
ast and
# once we do it, for other file, it's just a copy paste because we are lookin
g at the same time for each file.


# Step 6: Do the same thing (by copy and paste the functions) on every 16 fil
e.
```

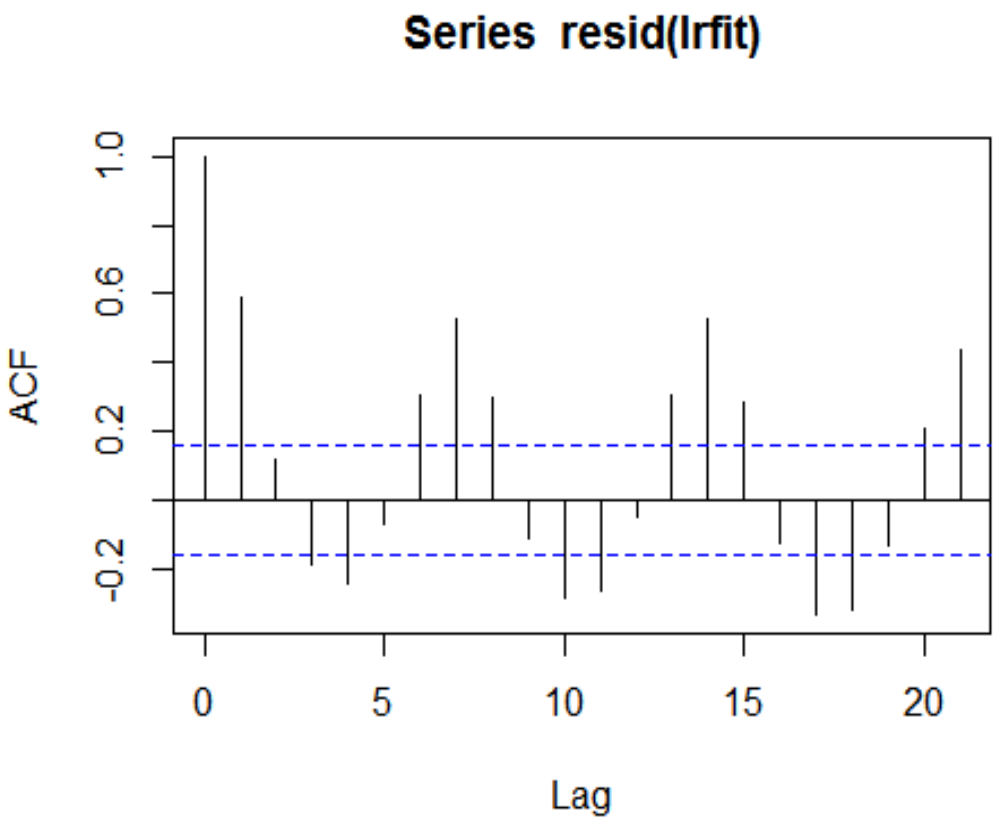
```r
# Step 7: Merge after using copy-paste special in Excel.

# Demand data that I am analyzing : jan15-may15 uber data demand
JUN15 <- read.csv("jun15 uber data demand(numbers only).csv", head=TRUE, sep=
",")
JUN15DATA <- JUN15$X.1

JANtoMAY15 <- read.csv("jan15-may15 uber data demand (numbers only).csv", hea
d=TRUE, sep=",")
JANtoMay15DATA <- JANtoMAY15$X.1
JANtoMay15DATA <- JANtoMay15DATA[!is.na(JANtoMay15DATA)]

# Conduct Same Analysis with the data. JANtoMAY15DATA for analysis and JUN15D
ATA to compare the results

# 1) Check stationarity, and if non-stationary, then make it stationary.


plot(JANtoMay15DATA, type="o", col="blue", xlab="Dates", ylab="Number of Dema
nd", xaxt='n')
title(main="Uber rides in NYC from JAN-MAY 2015",
      col.main="red", font.main=4)

xticks <- c("JAN","FEB","MAR","APR","MAY", "JUN")
axis(1, at= c(1, 31, 62, 92, 122, 153), lab=xticks)
```



```r
# visually, there is a bit of trend. Take it out of data.
fit15 <- lm(JANtoMay15DATA~time(JANtoMay15DATA))
detr_data <- resid(fit15)
plot(detr_data, type="o", col="blue", xlab="Dates", ylab="Number of Demand",
xaxt='n')
title(main="Detrended data of Uber rides in NYC from JAN-MAY 2015",
      col.main="red", font.main=4)
xticks <- c("JAN","FEB","MAR","APR","MAY", "JUN")
axis(1, at= c(1, 31, 62, 92, 122, 153), lab=xticks)

# Now let's see if the mean is constant by doing linear regression.
lrfit <- lm(detr_data~time(detr_data))
abline(lrfit)
```




```r
# As we can see the straight line, the mean is constant.
# Let's look at ACF now to see variance.

acf(resid(lrfit))
```




```
# ACF shows ticks at h=0 and h=1 and periodicity and this
# says that the autocorrelations are functions of lags. Stationarity confirme
d.
# Let's try using R-functions this time to recheck the stationarity.

# Let's do unit root test if another transformation is required from here.
adf.test(detr_data, alternative="stationary")

##
##  Augmented Dickey-Fuller Test
##
## data:  detr_data
## Dickey-Fuller = -3.3109, Lag order = 5, p-value = 0.07208
## alternative hypothesis: stationary

# The null-hypothesis for an ADF test is that the data are non-stationary.
# So large p-values are indicative of non-stationarity, and small p-values
# suggest stationarity. Using the usual 5% threshold, p-value is 0.01 so
# more differencing from here is not required.
# p-value is 0.01<0.05

# we can also do Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test.
# This will suggest differencing if p-value is higher than 0.05 (5%) (vice-ve
```

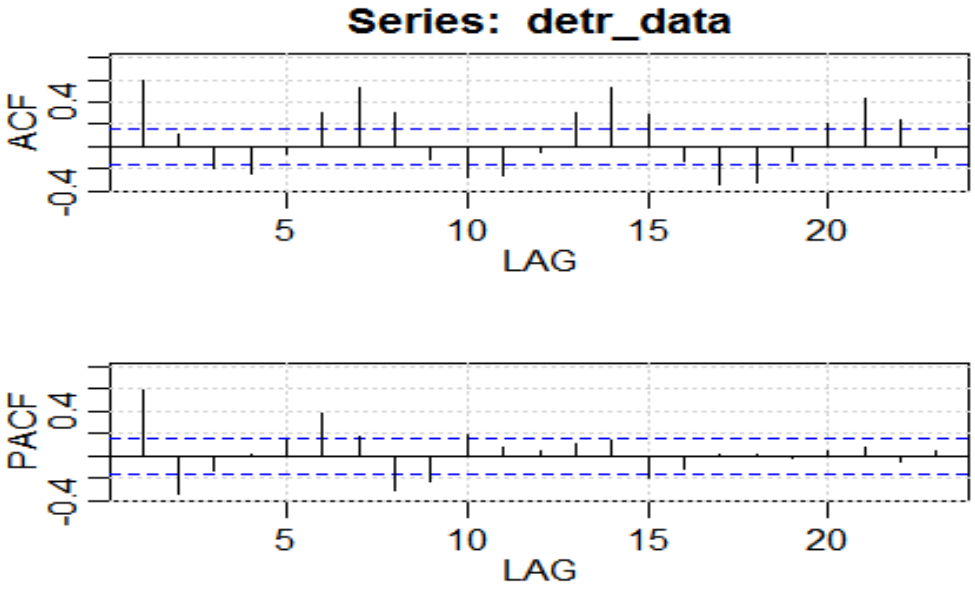
```
rsa from ADF test)

kpss.test(detr_data)
```

```
## Warning in kpss.test(detr_data): p-value greater than printed p-value

##
##  KPSS Test for Level Stationarity
##
## data:  detr_data
## KPSS Level = 0.13784, Truncation lag parameter = 2, p-value = 0.1
```

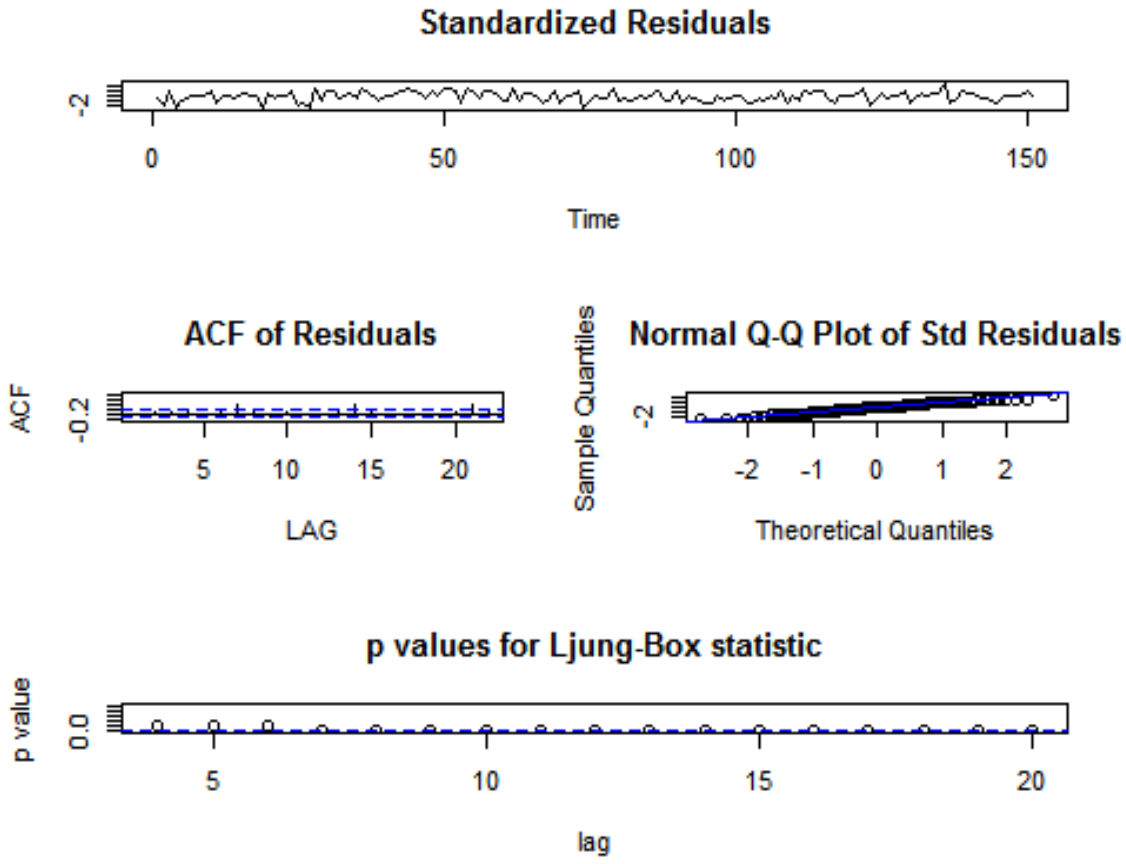
```
# p-value is 0.1>0.05 --> suggesting stationarity.

adf.test(detr_data)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  detr_data
## Dickey-Fuller = -3.3109, Lag order = 5, p-value = 0.07208
## alternative hypothesis: stationary
```

```
# p-value is 0.07208 --> suggesting stationarity (no more differencing or tra
snformation is needed)

# Let's find ARMA modeling.

# check which p,q would be appropriate for this data
acf2(detr_data)
```



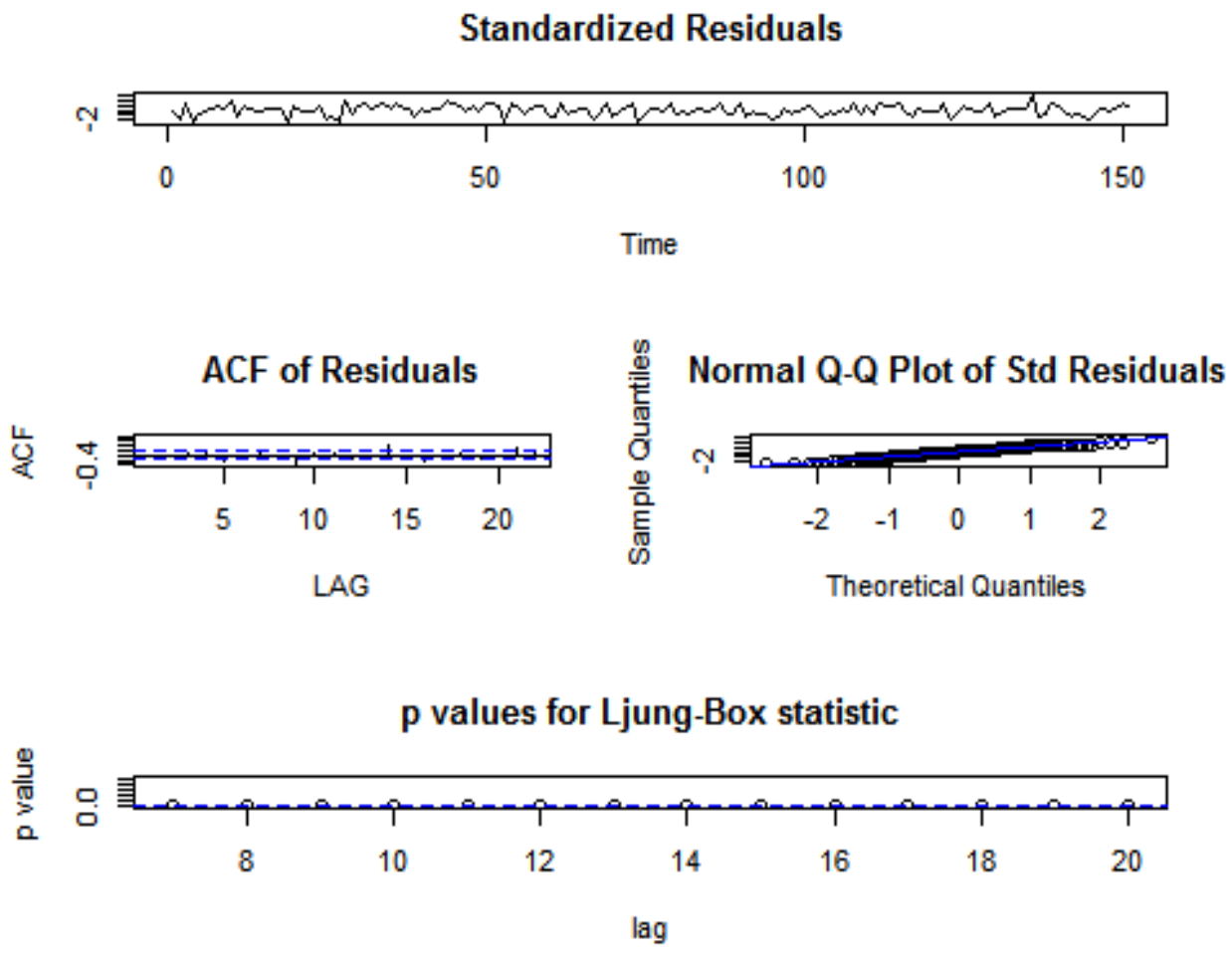
```r
# ACF suggests MA(q) with q=1 (one tickmark and decay rapidly)
# PACF suggests AR(p) with p=2 (two tickmarks and decay rapidly)

# Check whether ARMA(2,1) is good
sarima(detr_data, 2,0,1)
```




```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D
,
##     Q), period = S), xreg = xmean, include.mean = FALSE, optim.control = l
ist(trace = trc,
##     REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1      ar2      ma1      xmean
##       1.0032  -0.4784  -0.2414    40.4233
## s.e.  0.1373   0.0948   0.1428  1288.8696
##
## sigma^2 estimated as 98135344:  log likelihood = -1603.96,  aic = 3217.91
##
```

```
## $AIC
## [1] 19.45484
##
## $AICc
## [1] 19.47082
##
## $BIC
## [1] 18.53477
```

```
# Here, you can always use 'forecast' package and do auto.arima for finding
# which p,q to use but sometimes, it will give you more errors due to functio
nal limits.
# For example, if run in this case, it will give us ARMA(5,1) but when we exa
mine
# the pvalues for Ljung Box statistic, it is terrible.
sarima(detr_data, 5,0,1)
```

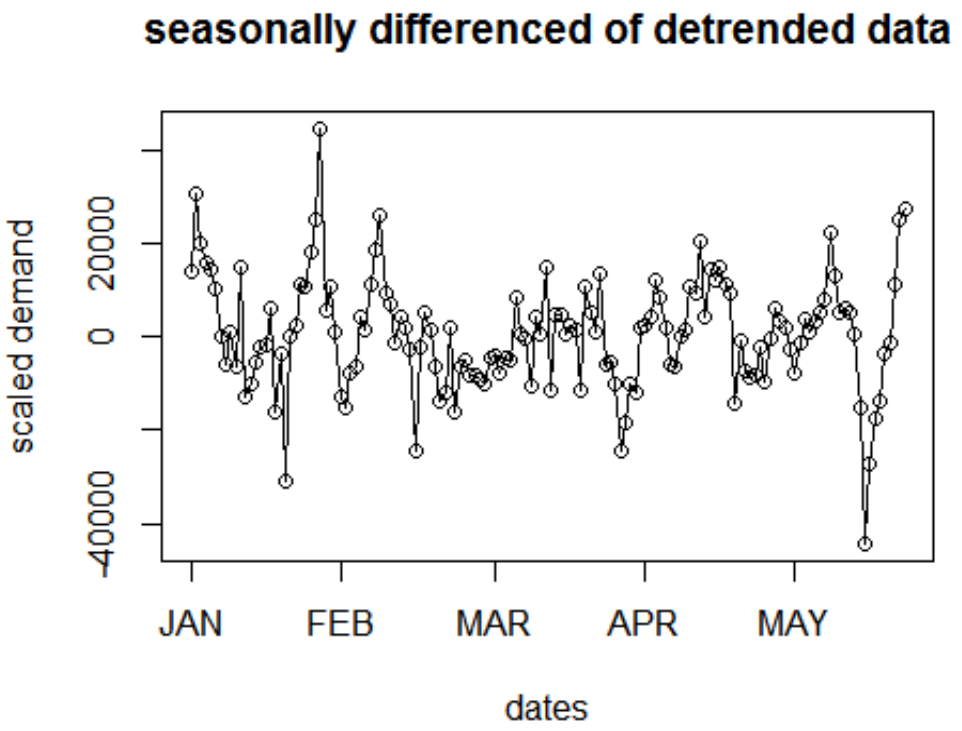



```
## $AIC
## [1] 19.39923
##
```

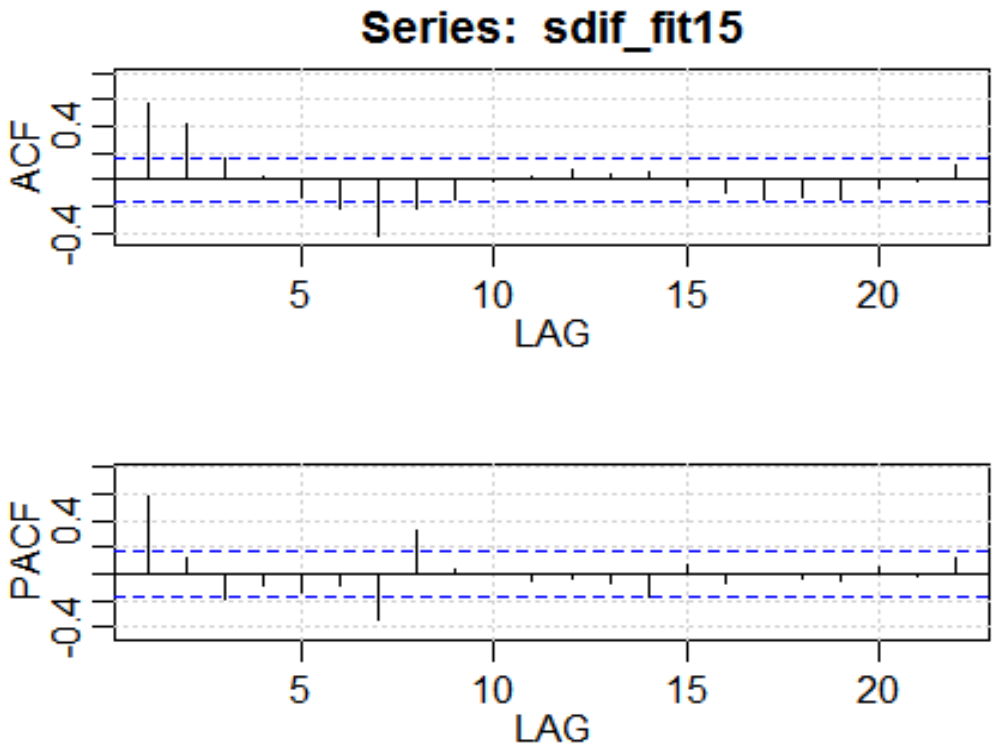
```
## $AICc
## [1] 19.41919
##
## $BIC
## [1] 18.5391
```

```
# ACF of residuals show us that there is a periodicity of 7 (do seasonal diff
erencing),
# p values for Ljung-box does have many values that are under the 0.05 level
# which shows the dependencies. (obviously, the data is non-stationary in ter
ms of periodicity)
# normal q-q plot looks good though.
```

```
## Seasonal Analsysis

sdif_fit15 = diff(detr_data, 7)
plot(sdif_fit15,type="o", main="seasonally differenced of detrended data",
     xlab="dates", ylab="scaled demand", xaxt='n')


xticks <- c("JAN","FEB","MAR","APR","MAY", "JUN")
axis(1, at= c(1, 31, 62, 92, 122, 152), lab=xticks)
```

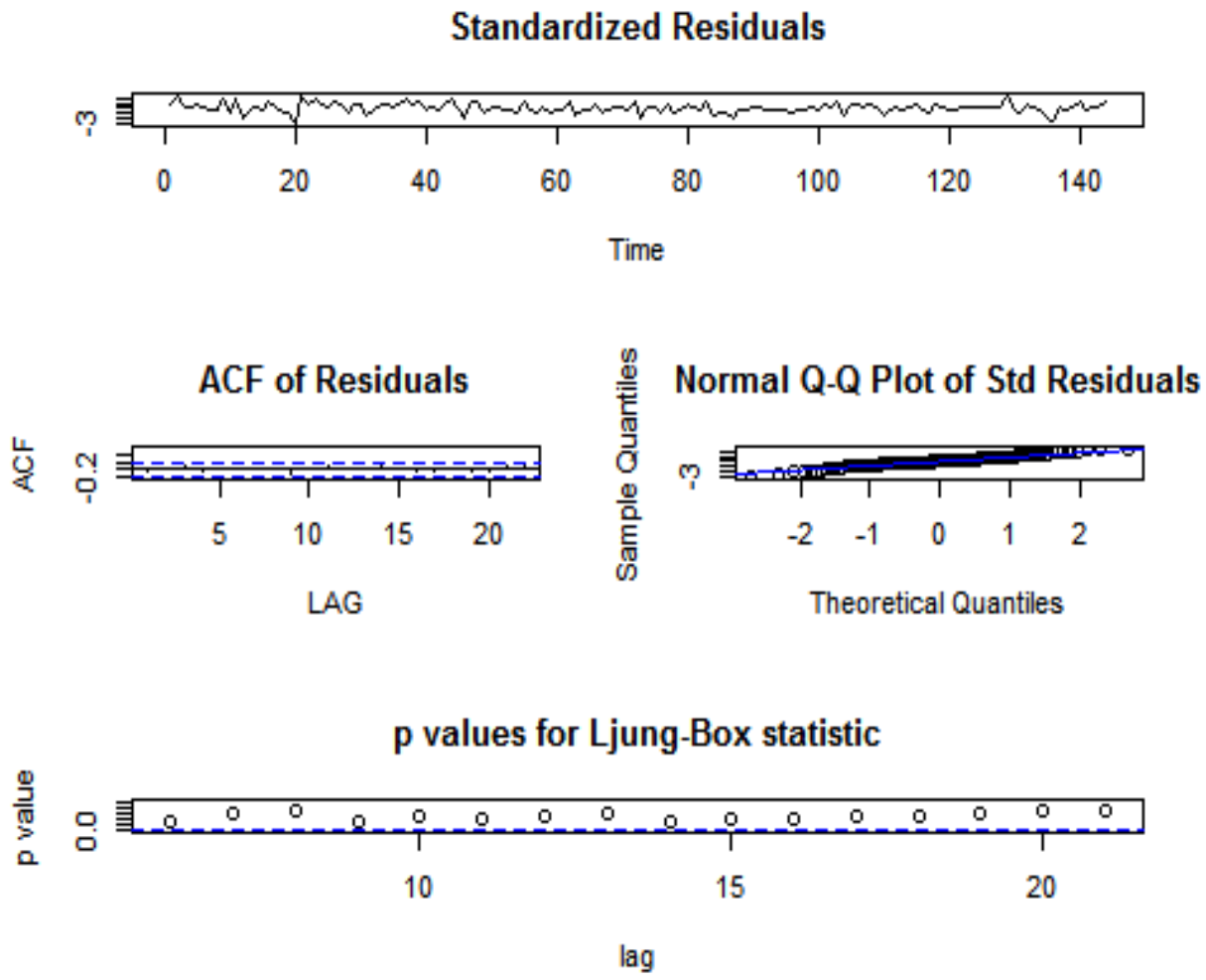


```
# There are some great differences near FEB(high demand), and near mid May(lo
w demand)
# If I can, GARCH or ARCH model would be great to take these into account.
# for now, let's proceed into SARIMA building.

acf2(sdif_fit15)
```




```
# Case: ACF is cutting off after lag1s and PACF is cutting off after lag1s ->
SARMA(1,1)
# Case: ACF is cutting off after lag2s and PACF is cutting off after lag1s->
SARMA(1,2)

# Case: ACF tailing off and PACF cutting off after lag1s -> SAR(1)

# based on ARMA(2,1)
```

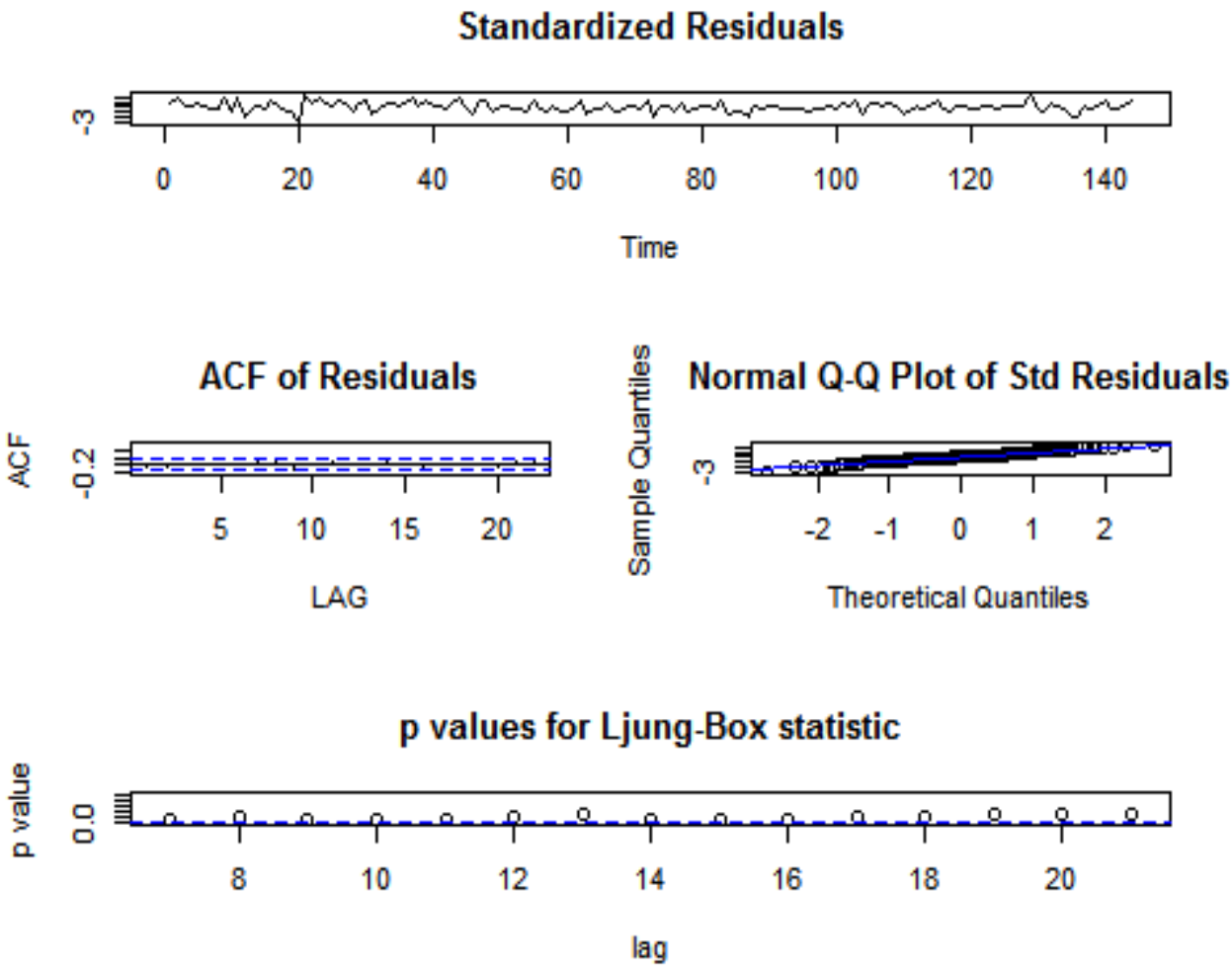
```r
# CASES:
sarima(sdif_fit15, 2,0,1,1,0,1,7)
```




```
## $AIC
## [1] 18.82543
##
## $AICc
## [1] 18.84504
##
## $BIC
## [1] 17.94917
```

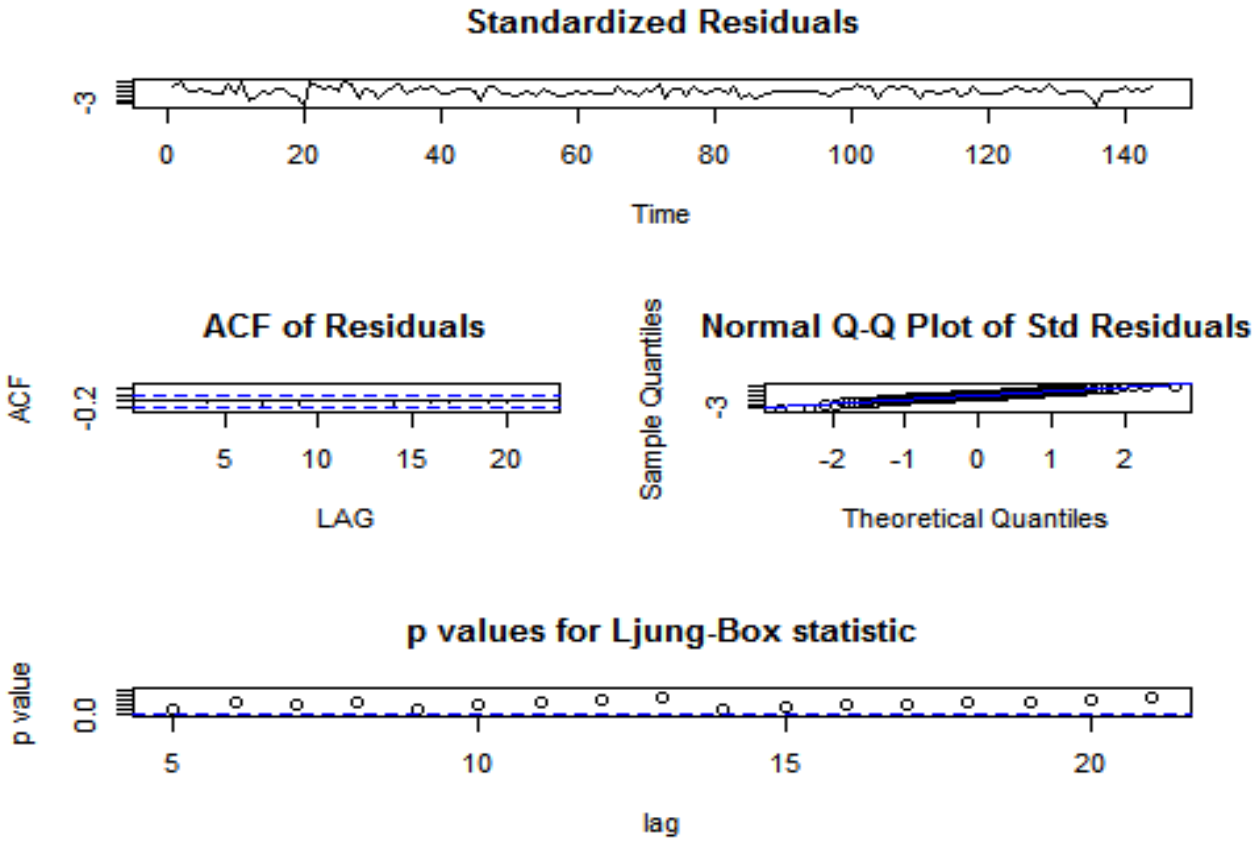
```r
#18.82543, 18.84504, 17.94917
```

```
sarima(sdif_fit15, 2,0,1,1,0,2,7)
```




```
## $AIC
## [1] 18.83967
##
## $AICc
## [1] 18.86096
##
## $BIC
## [1] 17.98403

#18.83967, 18.86096, 17.98403
```

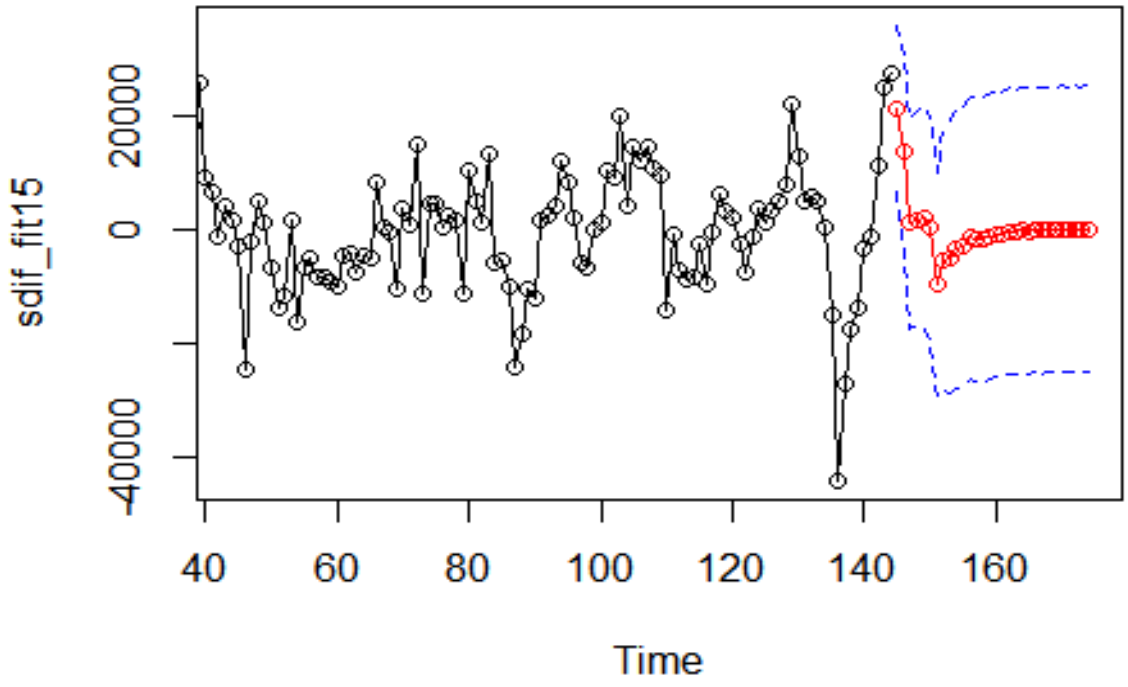
```
sarima(sdif_fit15, 2,0,1,1,0,0,7)
```




```
## $AIC
## [1] 19.08233
##
## $AICc
## [1] 19.10048
##
## $BIC
## [1] 18.18545
```

```
#19.08233, 19.10048, 18.18545
```

```r
# By comparing values here, the lowest values were focused on
# arima(2,0,1) and sarima(1,0,1)(7)

# So this is the model that I am going to fit now for estimation.
sarima.for(sdif_fit15, 30, 2,0,1,1,0,1,7)
```




```
## $pred
## Time Series:
## Start = 145
## End = 174
## Frequency = 1
##  [1] 21499.928653 13698.653671  1186.576458  1856.985510  2087.956924
##  [6]   254.967018 -9552.582192 -5456.264127 -4850.576573 -3406.347570
## [11] -2976.465909 -1341.309460 -1595.124067 -1542.392535 -1372.861032
## [16]  -628.164617  -760.406116  -237.786817  -337.668067   -13.402210
## [21]  -237.113930    41.091942   -77.254921   109.137051     3.449636
## [26]   147.797030    52.595409   150.745553    76.164575   158.796046
##
## $se
## Time Series:
## Start = 145
## End = 174
## Frequency = 1
```

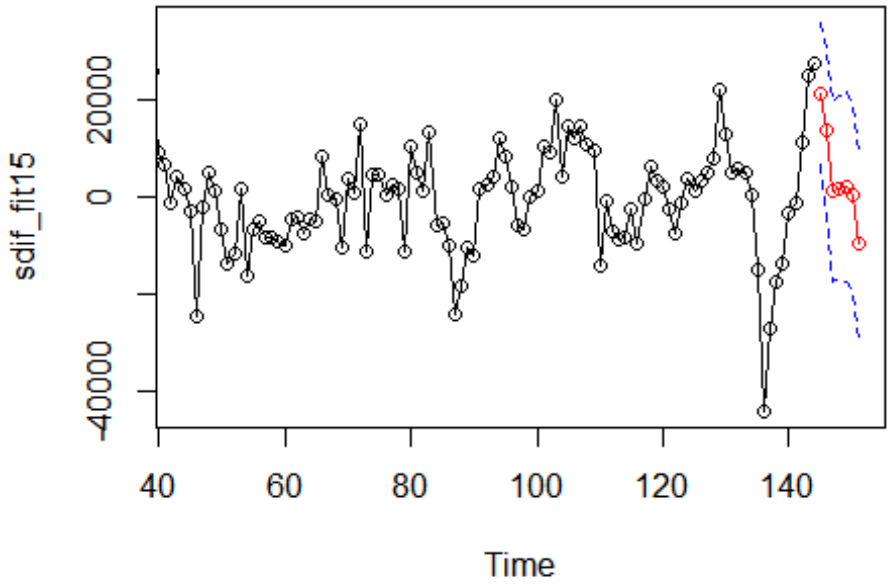
```
##  [1]  7288.516  8562.337  9336.106  9567.844  9761.589  9809.143  9862.963
##  [8] 11455.661 11915.942 12297.322 12387.042 12493.384 12509.247 12541.686
## [15] 12570.059 12602.479 12607.634 12617.492 12618.193 12621.470 12621.505
## [22] 12623.706 12623.728 12624.540 12624.546 12624.880 12624.903 12625.056
## [29] 12625.073 12625.163
```

```
# The above estimation also gave us the beginning parts, but then
# again, we could not estimate thereafter. Let's try the same method
# used above.
```

```
sarima.for(sdif_fit15, 7, 2,0,1,1,0,1,7)$pred
```

```
## Time Series:
## Start = 145
## End = 151
## Frequency = 1
## [1] 21499.929 13698.654  1186.576  1856.986  2087.957   254.967 -9552.582
```

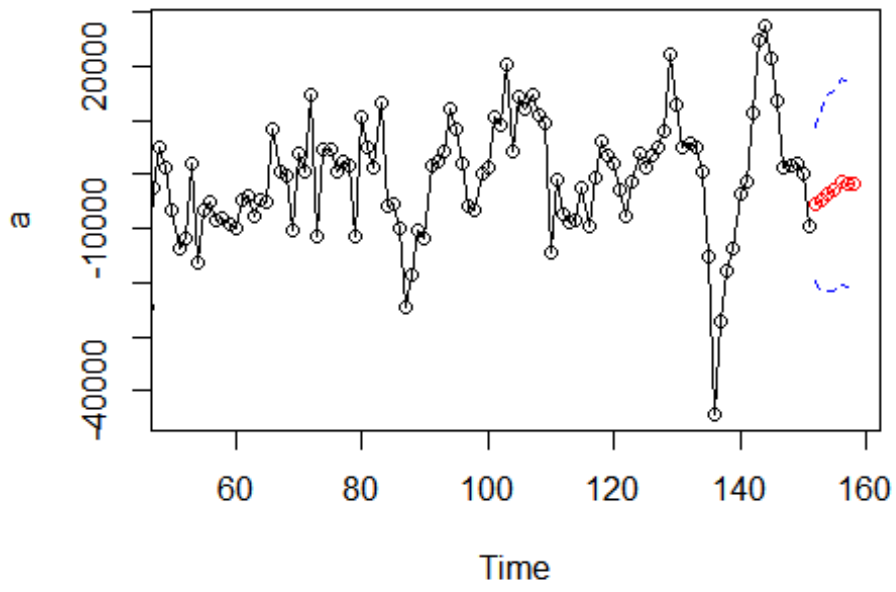
```
a=c(sdif_fit15, sarima.for(sdif_fit15, 7, 2,0,1,1,0,1,7)$pred)
```



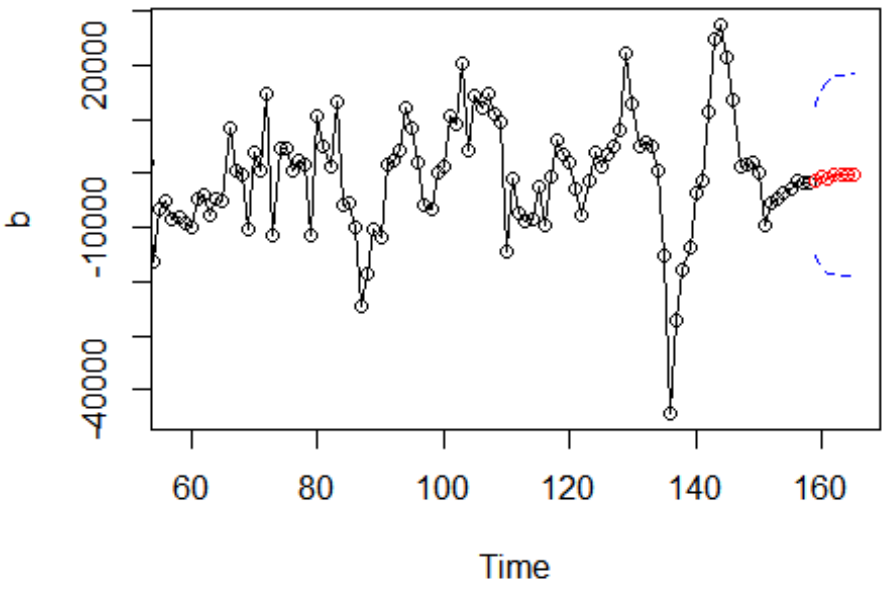

```
sarima.for(a, 7, 2,0,1,1,0,1,7)$pred
```

```
## Time Series:
## Start = 152
## End = 158
## Frequency = 1
## [1] -5458.812 -4855.839 -3415.019 -2985.150 -1347.069 -1603.372 -1553.018
```

```
b=c(a, sarima.for(a, 7, 2,0,1,1,0,1,7)$pred)
```

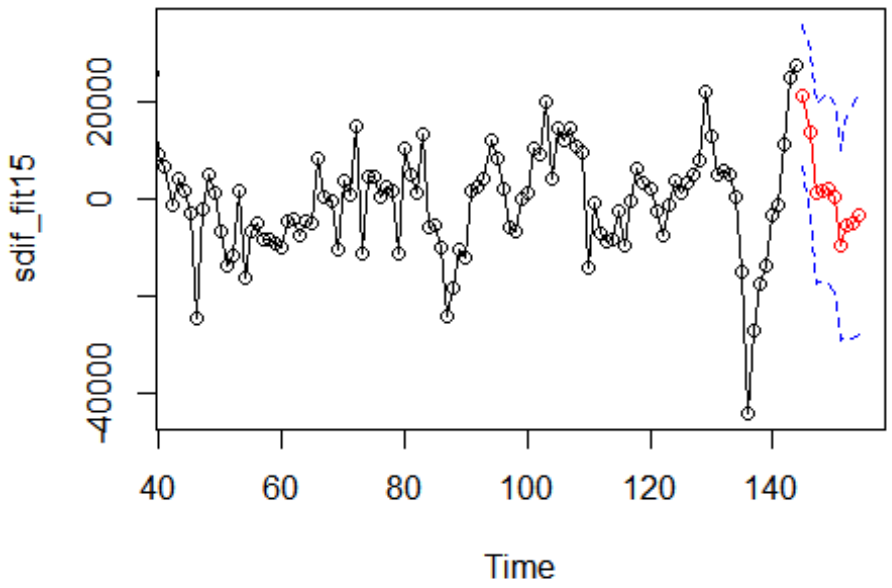



```
sarima.for(b, 7, 2,0,1,1,0,1,7)
```



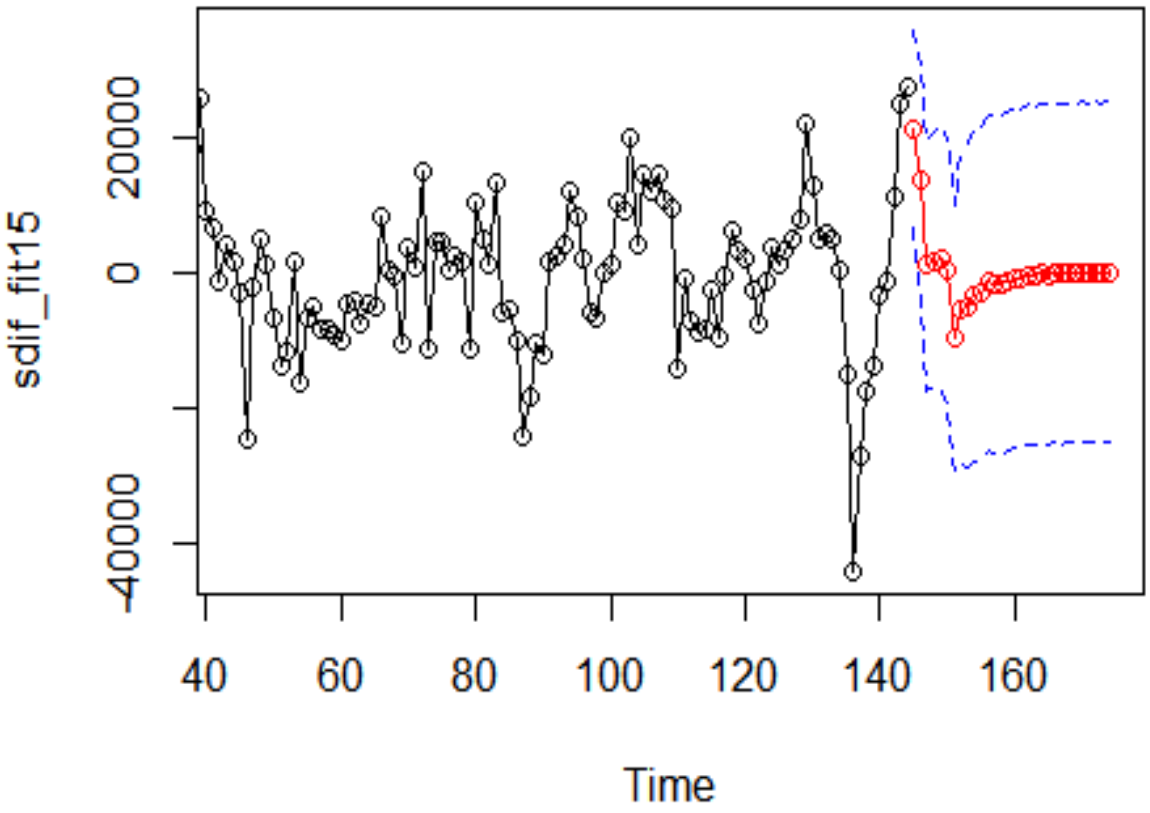
```
## $pred
## Time Series:
## Start = 159
## End = 165
## Frequency = 1
## [1] -1381.92512  -636.12183  -765.19357  -242.43082  -338.57423   -16.4521
8
## [7]  -237.21089
##
## $se
## Time Series:
## Start = 159
## End = 165
## Frequency = 1
## [1] 6943.245 8158.571 8898.275 9121.250 9306.939 9352.799 9404.519
```

```
# stop doing it. It results the same as 30 day prediction.

# Take a look at first seasonal prediction.
sarima.for(sdif_fit15, 10, 2,0,1,1,0,1,7)$pred
```

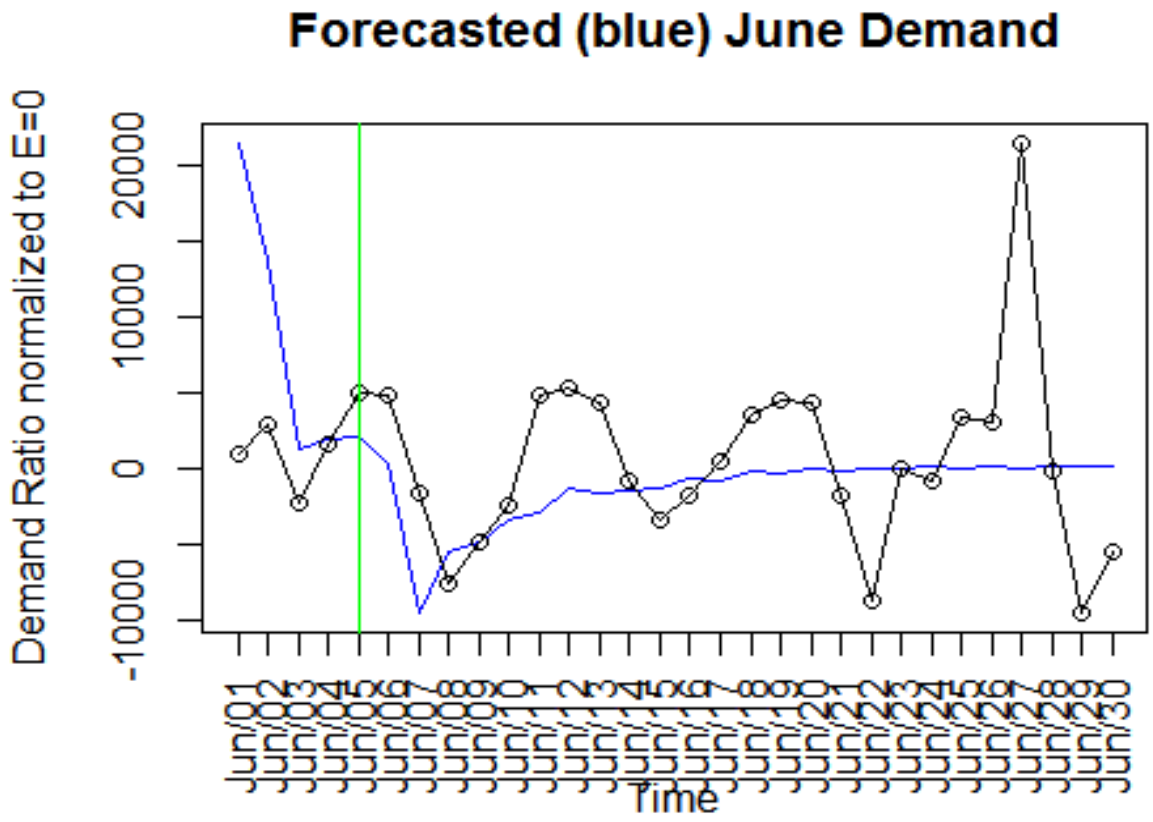



```
## Time Series:
## Start = 145
## End = 154
## Frequency = 1
##  [1] 21499.929 13698.654  1186.576  1856.986  2087.957   254.967 -9552.582
##  [8] -5456.264 -4850.577 -3406.348
```

```
# Recall June's data to see if the prediction is correct
# plot detrended June's and prediction

plot(sarima.for(sdif_fit15, 30, 2,0,1,1,0,1,7)$pred, col="blue", main="Foreca
sted (blue) June Demand", xaxt='n', ylab='Demand Ratio normalized to E=0')
```

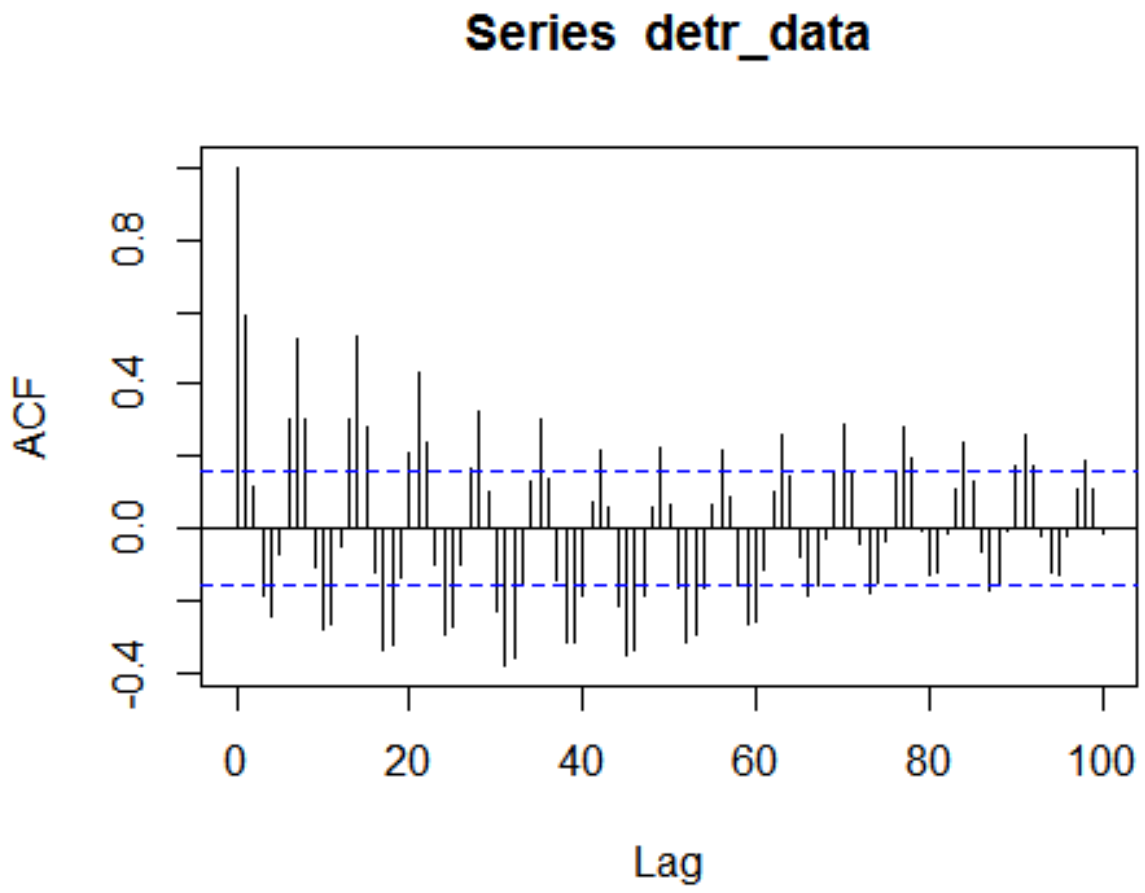



```
x=seq(as.Date("2015/6/1"), as.Date("2015/6/30"), "days")
axis(1, at=145:174, labels=format(x, '%b/%d'), las=2, font=0.5)
par(new=TRUE) # To put actual september data
plot(resid(lm(JUN15DATA~time(JUN15DATA))), axes = FALSE, xlab = "", ylab = ""
, type='o')
abline(v=5, col="green")
```

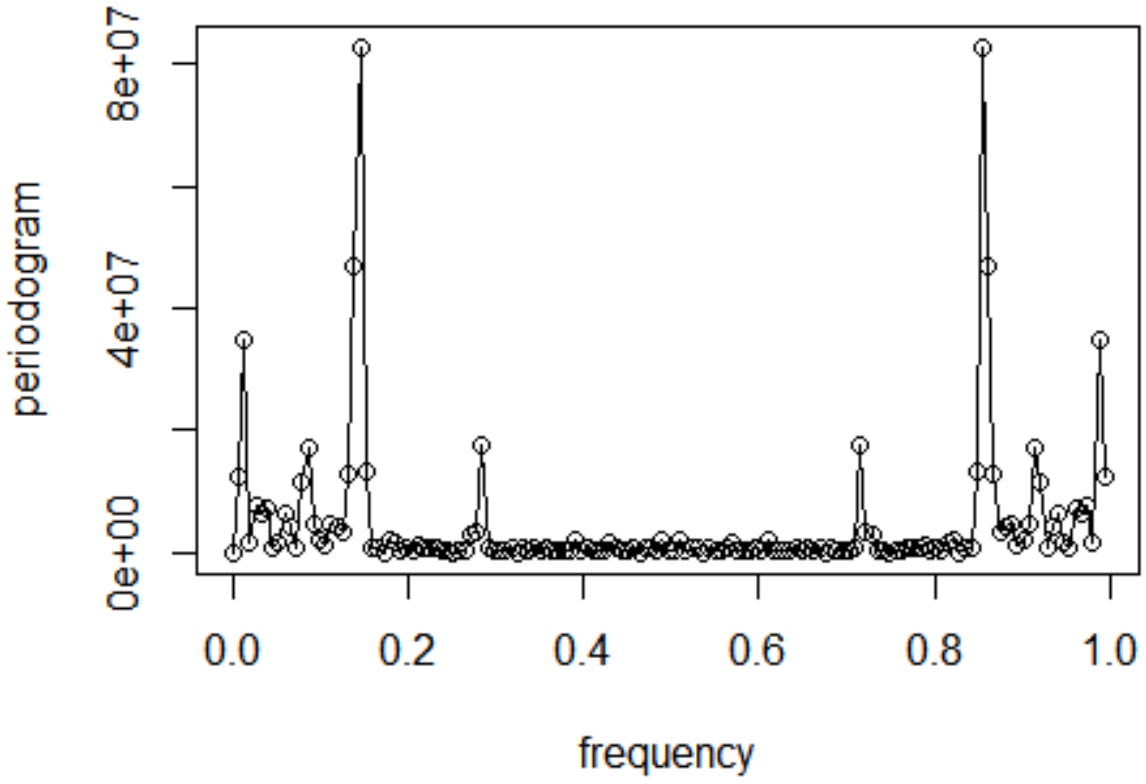



```
# As we can see here, the prediction that JUN 5th is going to have the
# highest demand in the first week is CORRECT.

# Not much findings other than that is possible here. Let's turn to periodici
ty
# using spectral analysis.

# What we have gained so far is that
# 1) JUNE 5th in the first week (Friday) will have the highest demand.
```

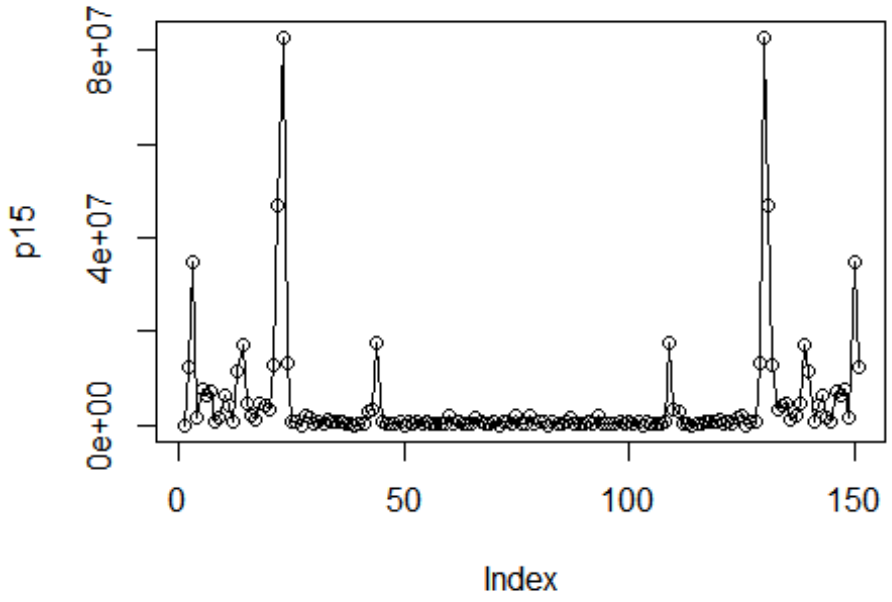
```r
# Checking whether spectral analysis is possible.
acf(detr_data,100)
```



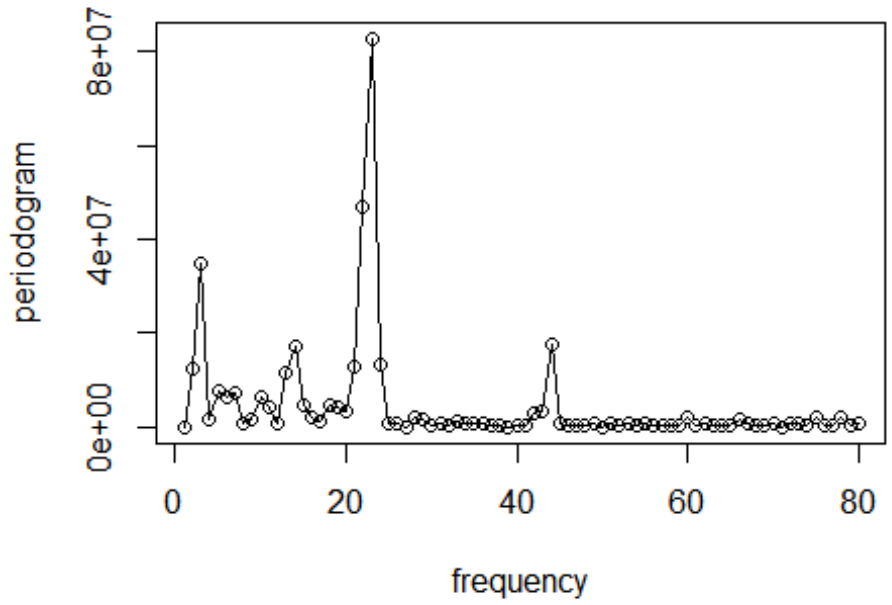

```r
# Yes, the values converge. The sum is bounded. Let's carry out the analysis.

p15=abs(2*fft(detr_data/(151)))^2 # by the definitions
Fr15=(0:150)/151
plot(Fr15, p15, type="o", xlab="frequency", ylab="periodogram")
```




```
# We only have to analyze until 1/2 frequency. (symmetric)
# There seems to be 6 major peaks in the preriodogram.
# Let's try fitting it to 6 sines and cosines waves and fitting it to smoothe
d version with less peaks

plot(p15, type="o")
```

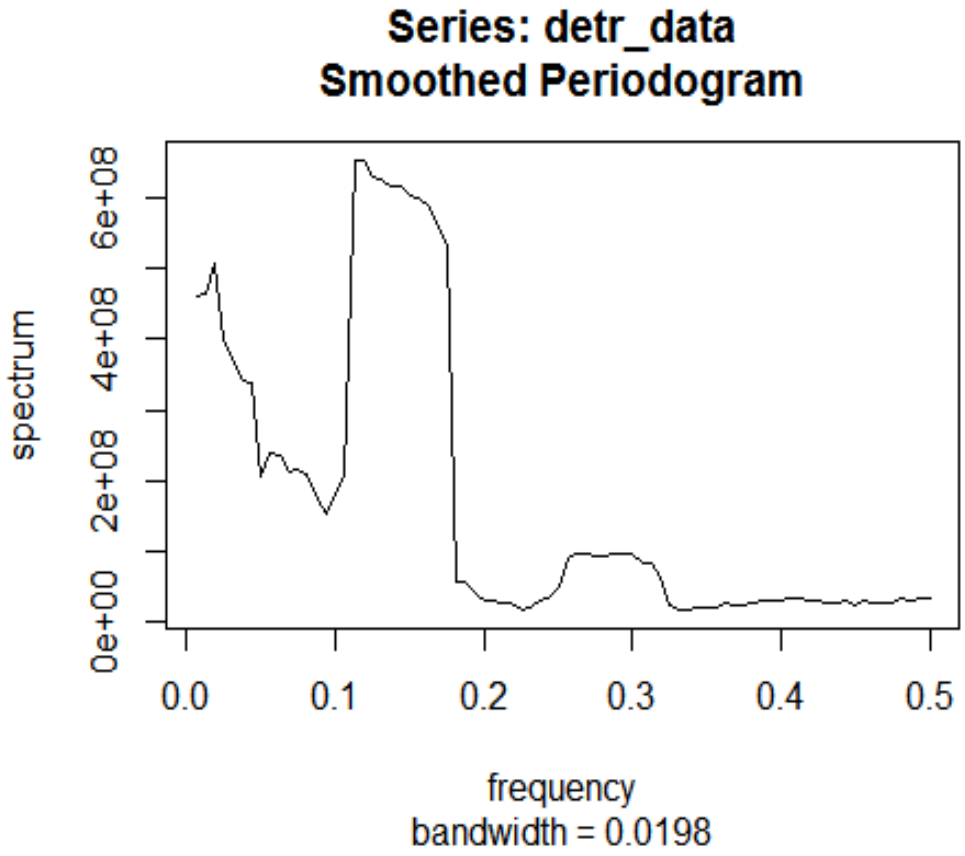



```r
plot(head(p15,80), xlab="frequency", ylab="periodogram", type="o")
```




```r
# Visually, n=3,6(between 5,7),10,14,23,44
# 6 peaks --> This results in bad estimation. (results below)
```

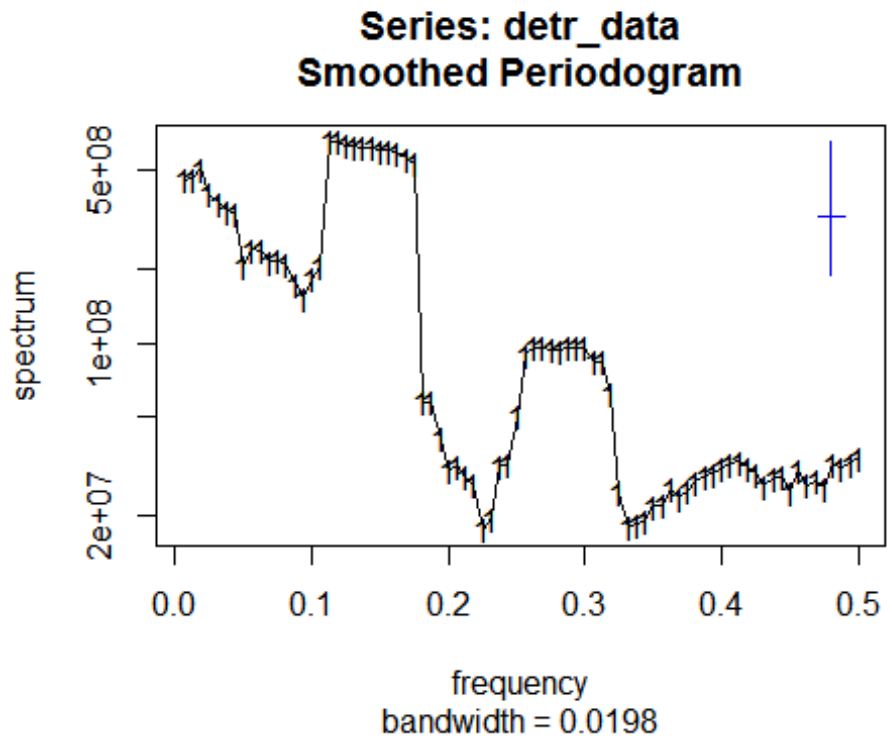
```r
# So take 4 that are outstanding
# n= 3, 14, 23, 44

# Let's try smoothing
k15=kernel("daniell", 5) # use danielle kernel
ave15=spec.pgram(detr_data, k15, taper=0, log='no')
```



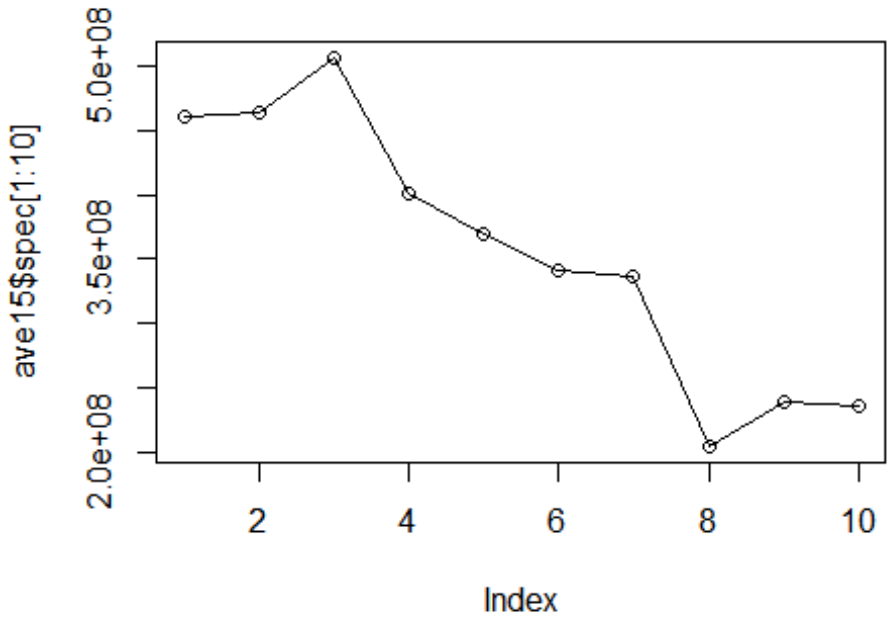

```r
# RECALL BELOW RESULT LATER (***)
1/ave15$freq[c(3,14,23,44)]
```

```
## [1] 53.333333 11.428571  6.956522  3.636364
```

```r
# Find frequency based on smoothing (skip n=1)
plot(ave15,type="o")
```

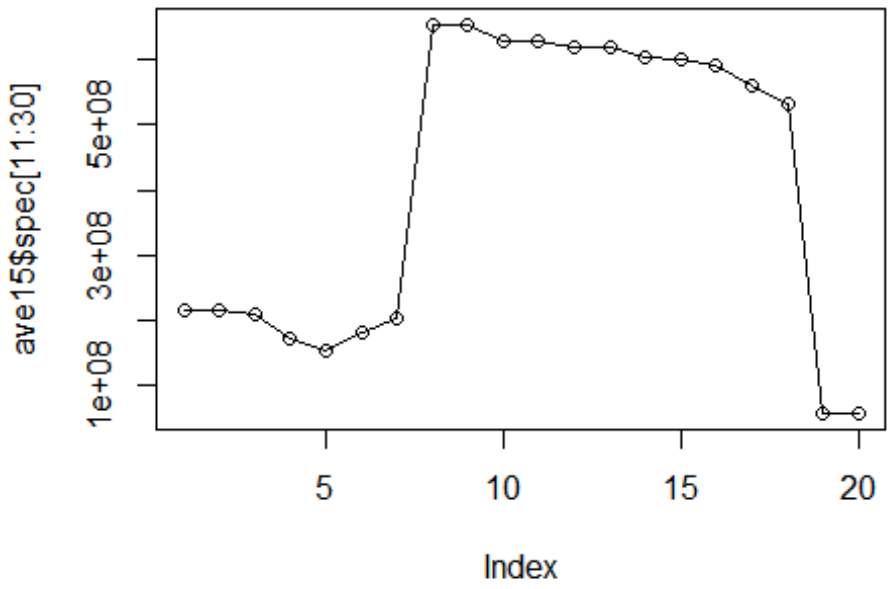



```
plot(ave15$spec[1:10],type="o")
```

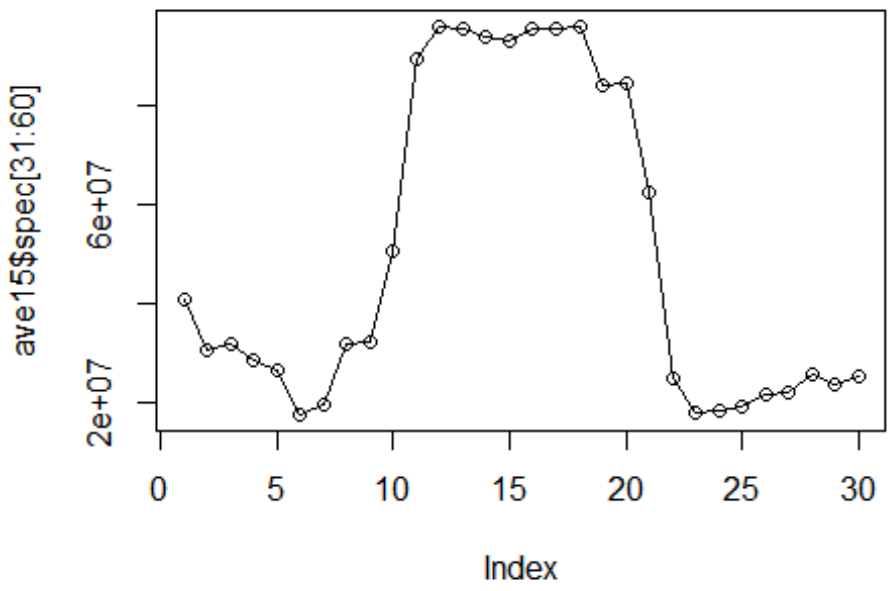



```
which(ave15$spec[1:10]==max(ave15$spec[1:10]))
```

```
## [1] 3

#3rd n

plot(ave15$spec[11:30],type="o")
```




```
which(ave15$spec[11:30]==max(ave15$spec[11:30]))

## [1] 8

#The values gives me 8, we add to 10 and it's 18, but as we can see the smoot
hed
#result, we can set it to mid-point of the box. --> 12+10 = 22

plot(ave15$spec[31:60],type="o")
```

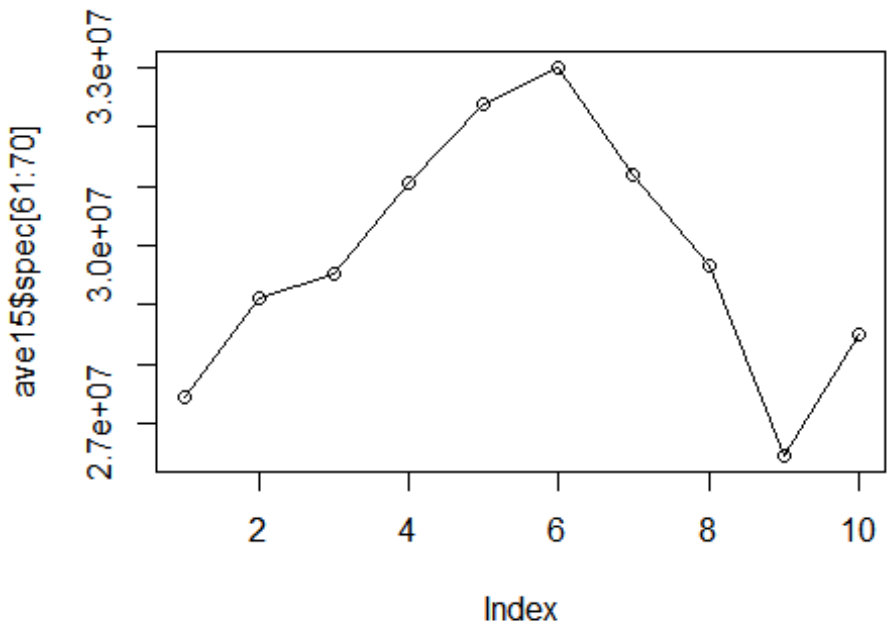



```
which(ave15$spec[31:60]==max(ave15$spec[31:60]))
```

```
## [1] 12
```

```
#The value gives me 12 but we know that we can take mid-point between
#12 and 18, which is 15 by visually inspecting. So add 15 to 30, 45.
```

```
plot(ave15$spec[61:70], type="o")
```

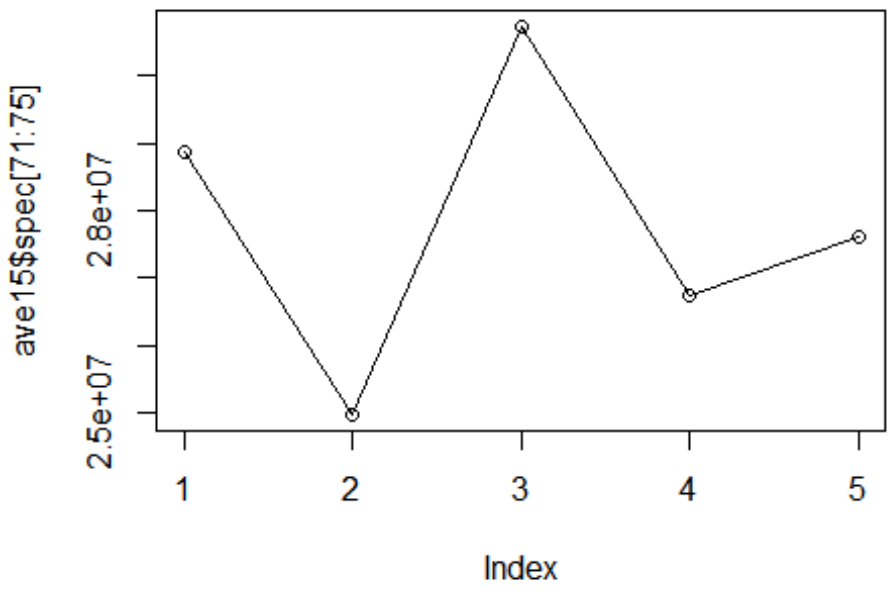



```
which(ave15$spec[61:70]==max(ave15$spec[61:70]))
```

```
## [1] 6
```

```
#The value gives me 6, add that to 60, it's 66.
```

```
plot(ave15$spec[71:75], type="o")
```

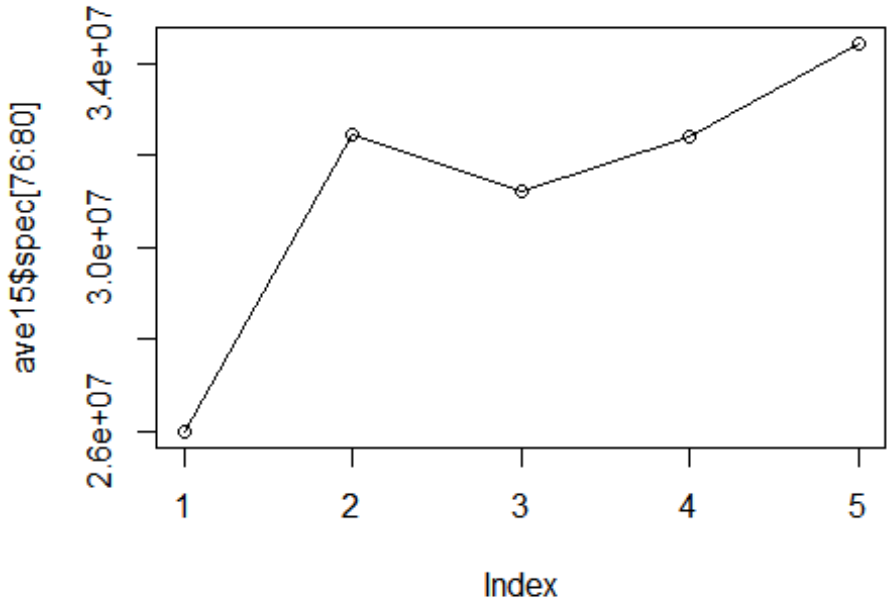



```
which(ave15$spec[71:75]==max(ave15$spec[71:75]))
```

```
## [1] 3
```

```
#The value gives me 3, add that to 70, it's 73.
```

```
plot(ave15$spec[76:80], type="o")
```




```
which(ave15$spec[76:80]==max(ave15$spec[76:80]))
```

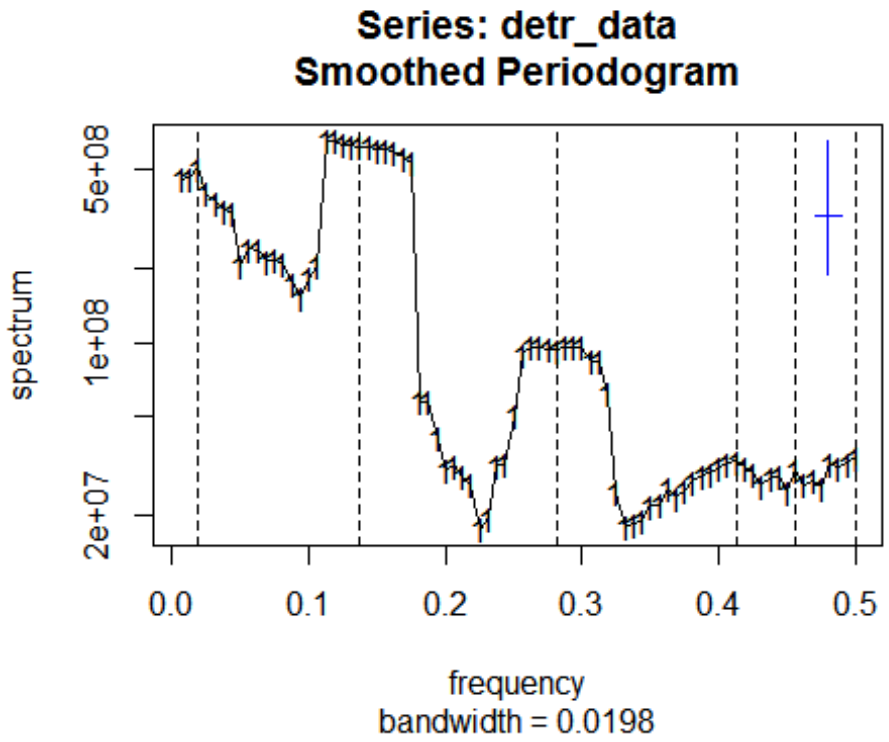
```
## [1] 5
```

```
#The value gives me 5, add that to 75, it's 80.

#Results: 3, 22, 45, 66, 73, 80
#store these to a
a =c(3,22,45,66,73,80)

# The estimated frequencies are
1/ave15$freq[a]
```

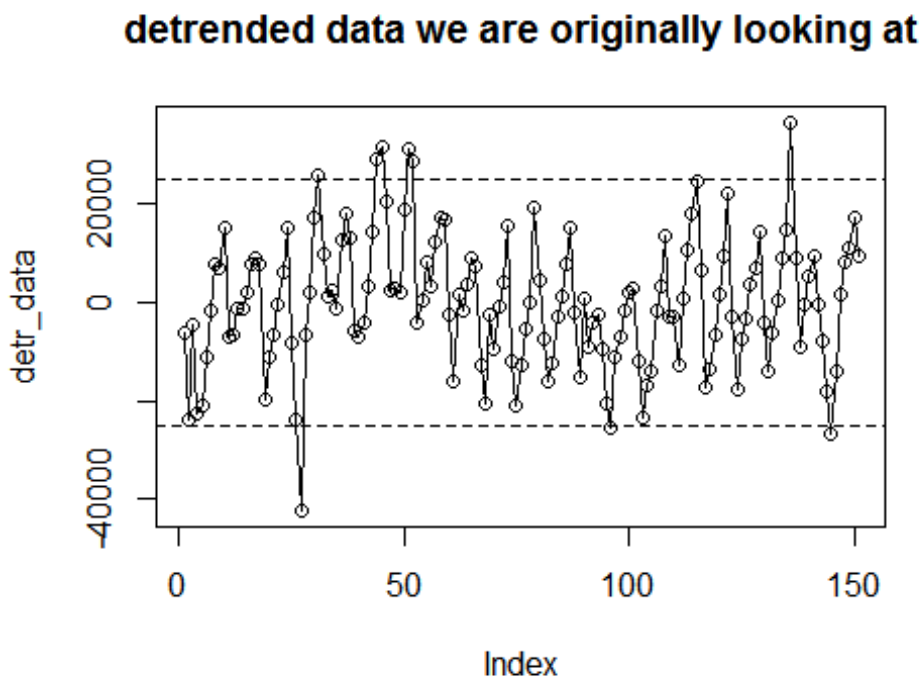
```
## [1] 53.333333  7.272727  3.555556  2.424242  2.191781  2.000000
```

```
# Let's try fitting after all.

# 3:80 = x:.5
v1=a[1]*0.5/80
# 22:80 = x:.5
v2=a[2]*0.5/80
# ...
v3=a[3]*0.5/80
v4=a[4]*0.5/80
v5=a[5]*0.5/80
v6=a[6]*0.5/80
```

```r
plot(ave15, type="o")
abline(v=c(v1,v2,v3,v4,v5,v6), lty=2)
```




```r
### So, using spectral analysis, we found out frequencies and general periodi
city.
# Let's use regression to discover the signal with known frequency.
plot(detr_data,type="o",main="detrended data we are originally looking at")
# We know by looking at the detrended data, the upward and downward do not go
# beyond, mostly, 25000 for each in magnitude. So, A should be around 25000
abline(h=c(-25000,25000), lty=2)
```



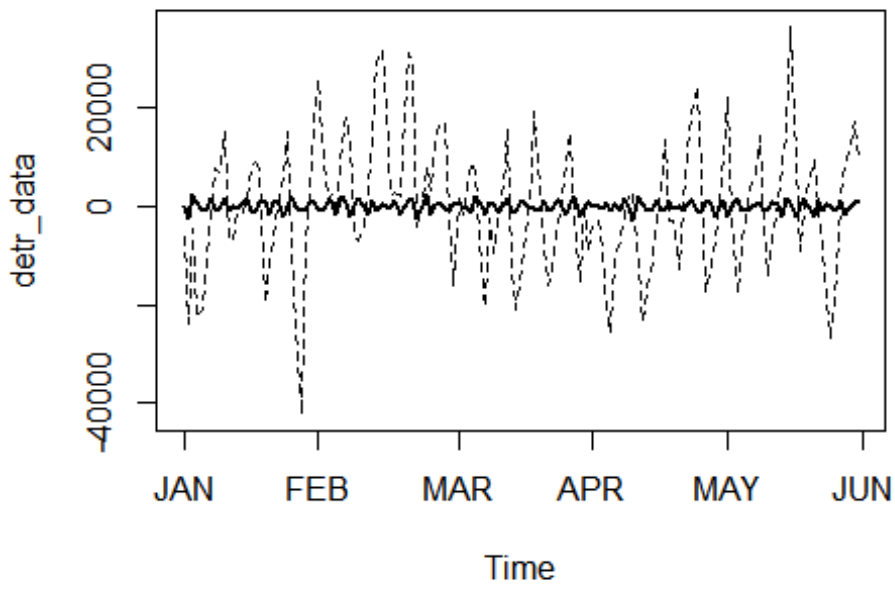

```
# We know that w=1/a where a=3,6(between 5,7),10,14,23,44 for non-smooth
# and for smooth, w=1/a where a equals
a
```

```
## [1]  3 22 45 66 73 80
```

```
# FIRST, NON-SMOOTHED periodo.
v=c(v1,v2,v3,v4,v5,v6)

z1=cos(2*pi*1:length(detr_data)/v[1])
z2=sin(2*pi*1:length(detr_data)/v[1])
z3=cos(2*pi*1:length(detr_data)/v[2])
z4=sin(2*pi*1:length(detr_data)/v[2])
z5=sin(2*pi*1:length(detr_data)/v[3])
z6=sin(2*pi*1:length(detr_data)/v[3])
z7=sin(2*pi*1:length(detr_data)/v[4])
z8=sin(2*pi*1:length(detr_data)/v[4])
z9=sin(2*pi*1:length(detr_data)/v[5])
z10=sin(2*pi*1:length(detr_data)/v[5])
z11=sin(2*pi*1:length(detr_data)/v[6])
z12=sin(2*pi*1:length(detr_data)/v[6])


summary(finalfit15 <- lm(detr_data~0+z1+z2+z3+z4
                         +z5+z6+z7+z8+z9+z10+z11+z12))
```

```
##
## Call:
## lm(formula = detr_data ~ 0 + z1 + z2 + z3 + z4 + z5 + z6 + z7 +
##     z8 + z9 + z10 + z11 + z12)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -41180  -8475     19   9082  35617
##
## Coefficients: (4 not defined because of singularities)
##       Estimate Std. Error t value Pr(>|t|)
## z1   3.371e+02  1.555e+03   0.217    0.829
## z2   8.182e+02  1.550e+03   0.528    0.598
## z3   1.017e+03  1.555e+03   0.654    0.514
## z4  -4.242e+02  1.546e+03  -0.274    0.784
## z5  -3.438e+02  1.543e+03  -0.223    0.824
## z6          NA         NA      NA       NA
## z7   5.332e+02  1.547e+03   0.345    0.731
## z8          NA         NA      NA       NA
## z9  -5.036e+02  1.648e+03  -0.306    0.760
## z10         NA         NA      NA       NA
## z11  3.546e+15  1.793e+16   0.198    0.843
## z12         NA         NA      NA       NA
##
## Residual standard error: 13470 on 143 degrees of freedom
## Multiple R-squared:  0.007592,   Adjusted R-squared:  -0.04793
## F-statistic: 0.1367 on 8 and 143 DF,  p-value: 0.9974
```

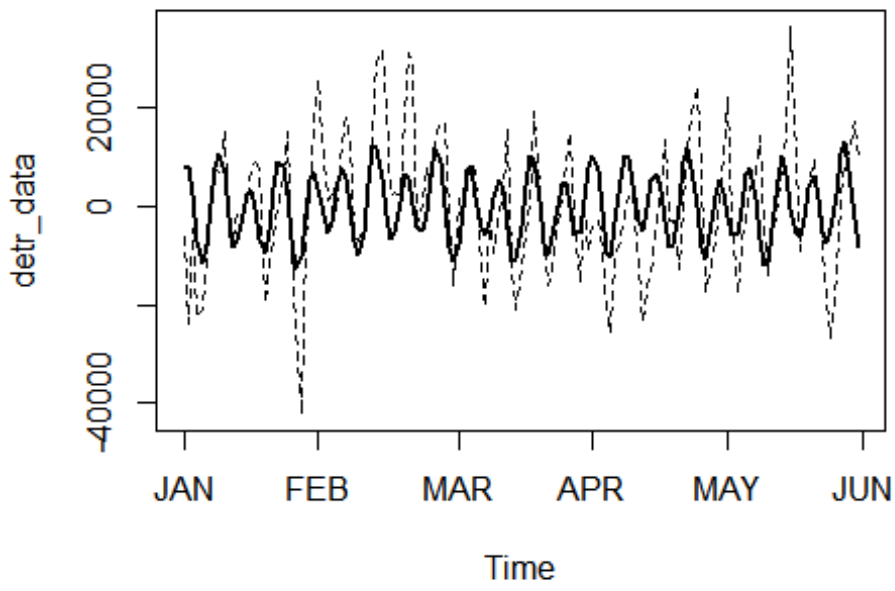
```r
plot.ts(detr_data, lty="dashed", xaxt='n')
lines(fitted(finalfit15), lwd=2)
xticks <- c("JAN","FEB","MAR","APR","MAY", "JUN")
axis(1, at= c(1, 31, 62, 92, 122, 152), lab=xticks)
```



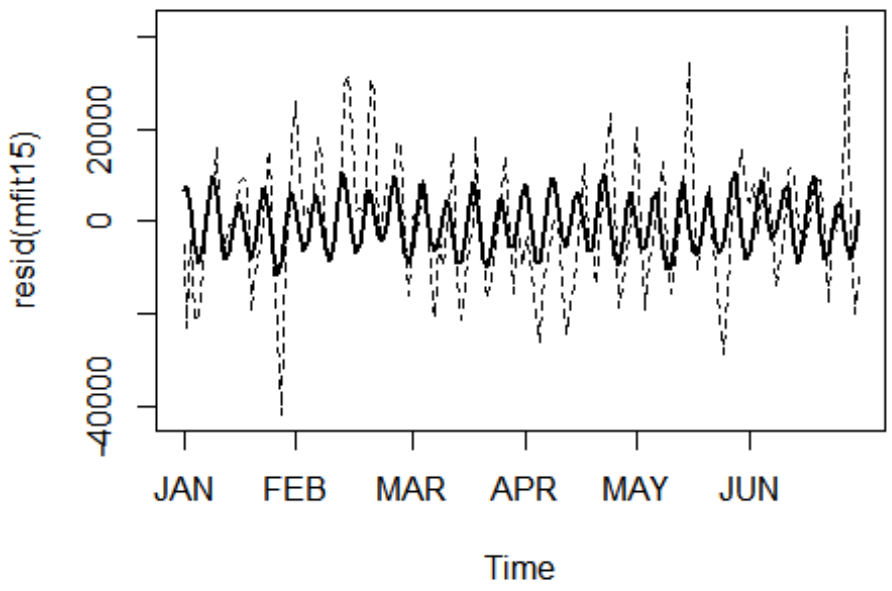

```
# Whether the next demand is higher or less than the previous day's demand ca
n be
# observed through this fitted estimate. However, it is hard to see and very
# bad in terms of estimating the quantities. This is because there are too ma
ny
# parameters for frequency. Let's take out the high frequency levels.
# It's not worth to go over smoothed, since there are still too many frequenc
ies.
# Call less frequencies.

## RECALL (***)
v=1/ave15$freq[c(3,14,23,44)]
z1=cos(2*pi*1:length(detr_data)/v[1])
z2=sin(2*pi*1:length(detr_data)/v[1])
z3=cos(2*pi*1:length(detr_data)/v[2])
z4=sin(2*pi*1:length(detr_data)/v[2])
z5=sin(2*pi*1:length(detr_data)/v[3])
z6=sin(2*pi*1:length(detr_data)/v[3])
z7=sin(2*pi*1:length(detr_data)/v[4])
z8=sin(2*pi*1:length(detr_data)/v[4])



summary(finalfit15_1 <- lm(detr_data~0+z1+z2+z3+z4
                           +z4+z5+z6+z7+z8))
```

```
##
## Call:
## lm(formula = detr_data ~ 0 + z1 + z2 + z3 + z4 + z4 + z5 + z6 +
##     z7 + z8)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -32201  -8164  -1206   6293  36717
##
## Coefficients: (2 not defined because of singularities)
##    Estimate Std. Error t value Pr(>|t|)
## z1   1065.8     1357.3   0.785   0.4336
## z2  -1005.1     1318.4  -0.762   0.4471
## z3   2343.8     1339.8   1.749   0.0823 .
## z4  -2345.0     1333.9  -1.758   0.0809 .
## z5   8492.9     1333.5   6.369 2.37e-09 ***
## z6       NA         NA      NA       NA
## z7   -592.3     1335.7  -0.443   0.6581
## z8       NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11610 on 145 degrees of freedom
## Multiple R-squared:  0.2529, Adjusted R-squared:  0.222
## F-statistic:  8.18 on 6 and 145 DF,  p-value: 1.254e-07
```

```r
plot.ts(detr_data, lty="dashed",xaxt='n')
lines(fitted(finalfit15_1), lwd=2)
axis(1, at= c(1, 31, 62, 92, 122, 152), lab=xticks)
```




```
# merge original data with september's data to see if prediction is corrrect.
JANtoJUN15DATA=c(JANtoMay15DATA, JUN15DATA)
mfit15=lm(JANtoJUN15DATA~time(JANtoJUN15DATA))
z1=cos(2*pi*1:length(resid(mfit15))/v[1])
z2=sin(2*pi*1:length(resid(mfit15))/v[1])
z3=cos(2*pi*1:length(resid(mfit15))/v[2])
z4=sin(2*pi*1:length(resid(mfit15))/v[2])
z5=sin(2*pi*1:length(resid(mfit15))/v[3])
z6=sin(2*pi*1:length(resid(mfit15))/v[3])
z7=sin(2*pi*1:length(resid(mfit15))/v[4])
z8=sin(2*pi*1:length(resid(mfit15))/v[4])

pmfit15=lm(resid(mfit15)~0+z1+z2+z3+z4+z5+z6+z7+z8)



plot.ts(resid(mfit15), lty="dashed", xaxt='n')
lines(fitted(pmfit15), lwd=2)
axis(1, at= c(1, 31, 62, 92, 122, 152), lab=xticks)
```




```
# From This Result - not good in terms of estimating which peaks are higher
# than the other peaks like the 2014 Results above - I believe that finding
# general trends through spectral analysis is NOT ENOUGH, or NOT GOOD at all.
# This is ONLY GOOD if we are trying to see if it will have upward trend or
# downward trend. We can see all the peaks do match to peaks and low-peaks to
# low-peaks. We can confirm this observation.

# Through doing linear ARIMA Modeling, both seasonal and non-seasonal,
# We were able to get first one to two weeks amount of "quantitative"
# (not like general trends) predictions using Y-W equations and Innovations.
# However, we could not predict further due to complextity of the data.
# The complexity entails, for example, at the end of June, there is a high de
mand
# (Which would be OPTIMAL for me to find that because that's when the 'surge-
pricing'
# is going to be very high if not enough supply (available drivers) is met)

## Conclusion 2 ##
# I think the data originally was unstable in a way that, because Uber is
# an emerging (and big) company with unpredictable demand in holidays or even
ts,
# the stationarity properties did not "satisfy" enough in order for
# linear ARIMA models or frequency analysis (or spectral analysis) to predict
# well. Though, we coerced the data into stationary as much as possible and
# still predicted 2014 results very well. (First two weeks have the highest d
emand)
```

```
# While this was a fruitful result (for me at least), I think simple combinat
ions of
# sines and cosines or simple linear ARIMA (even seasonal) modelings are not
enough
# To suggest further improvement, study of volatility (GARCH, ARCH Models) or
wavelet
# analysis would improve such long-stationary process.(high demand at the end
of June)
# However, I only have one such high peak in the data. This means I have too
'little' of
# a data to predict such. --End of analysis--
```