Explore Time Series Analysis

This is an exploration of time series analysis that includes moving average, holt-winters smoothing, and ARIMA models.

loading libraries

```
library(data.table)
library(dplyr)
```

```
## ------------------------------------------------------------------------------
```

```
## data.table + dplyr code now lives in dtplyr.
## Please library(dtplyr)!
```

```
## ------------------------------------------------------------------------------
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:data.table':
##
##     between, last
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(TTR)
library(forecast)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: timeDate
```

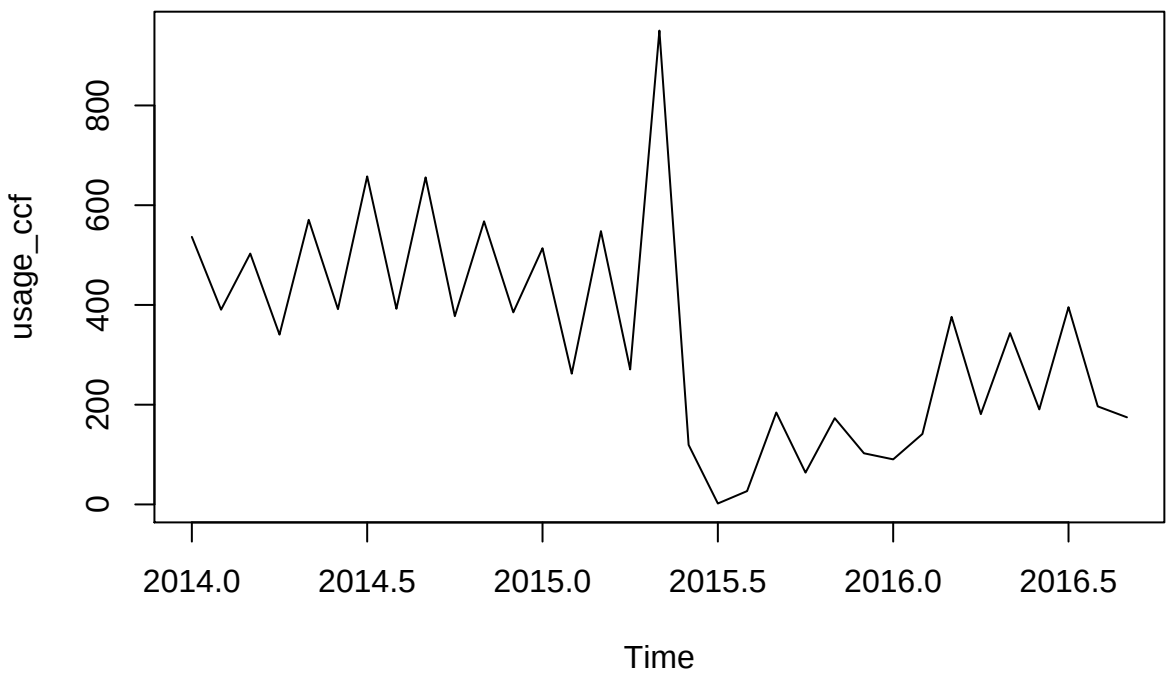
```
## This is forecast 7.3
```

Lets start with water usage data over time Read data from a csv and convert it to a time series. This is a randomly generated data for experimenting purposes.

```
smc_test <- fread('smc_test.csv', stringsAsFactors = FALSE)

smc_test <- arrange(smc_test, usage_date,cust_loc_id)

smc <- smc_test %>%  group_by(usage_date) %>% summarise(usage_ccf = sum(usage_ccf)/1000)

smc_ts_matrix <- as.matrix(smc[2])

smc_timeseries <- ts(smc_ts_matrix, frequency=12, start=c(2014,1))
```

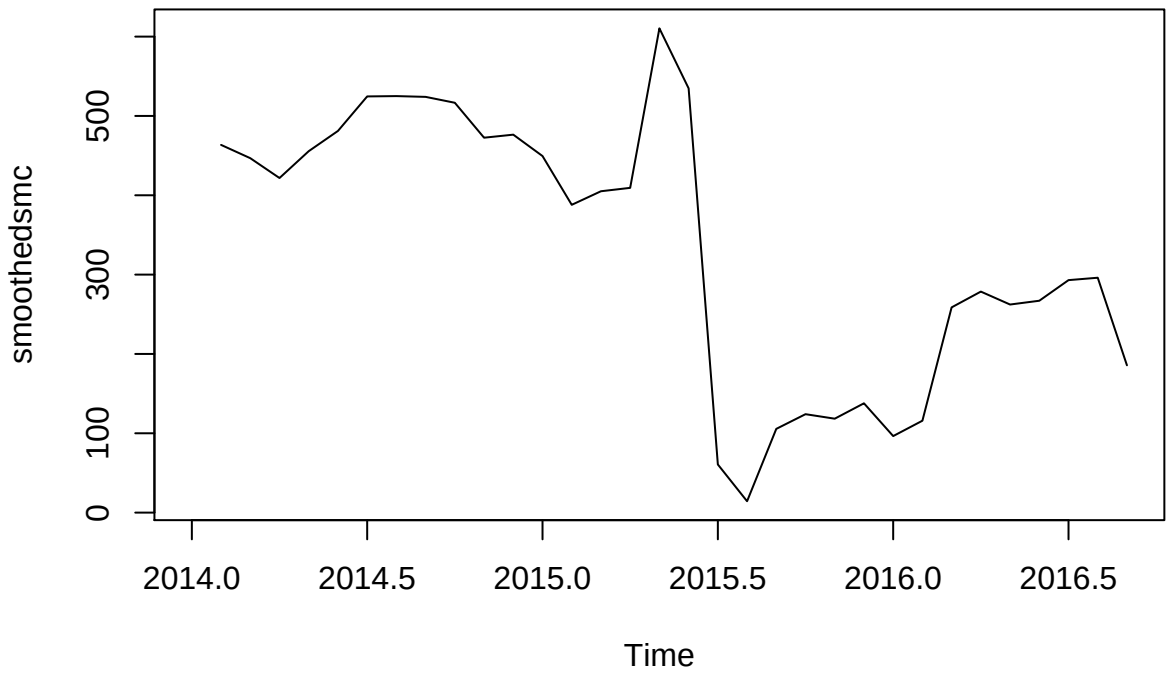
Plotting the timeseries

```
plot.ts(smc_timeseries)
```



Moving average of second order

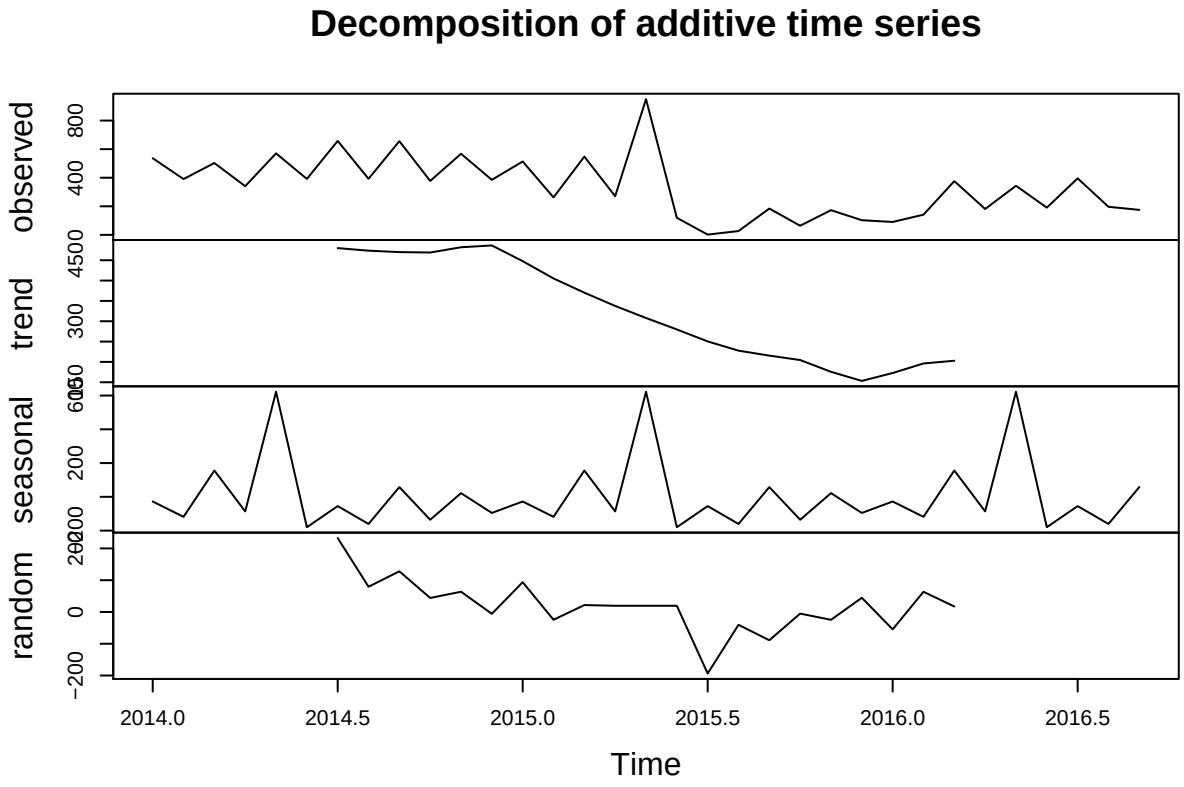
```
smoothedsmc <- SMA(smc_timeseries,n=2)
plot.ts(smoothedsmc)
```



We can observe that there is a severe downfall at mid 2015

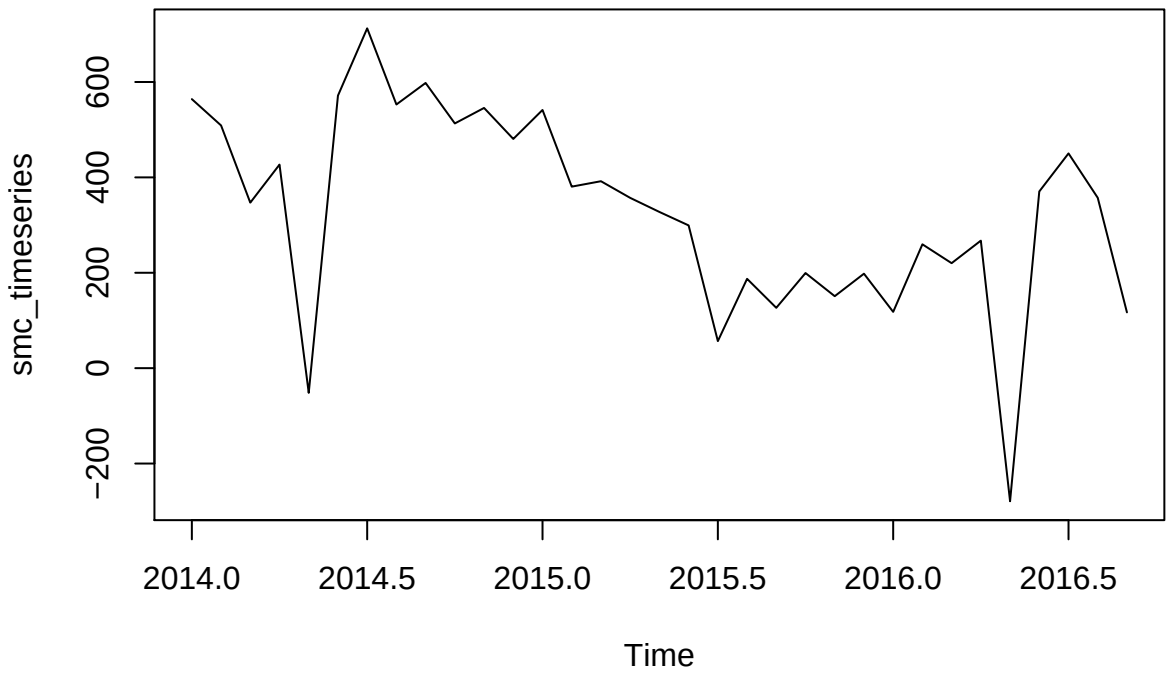
Here we can observe trend and seasonal components separately

```
smccomponents <- decompose(smc_timeseries)
plot(smccomponents)
```



Detach the seasonal component and see the plot

```
smcseasonallyadjusted <- smc_timeseries - smccomponents$seasonal
plot(smcseasonallyadjusted)
```



Simple exponential smoothing for default ts assuming no trend. In plot, Red line is the forecast, Black line is the original data

```
smc_notrend <- HoltWinters(smc_timeseries, beta=FALSE, gamma=FALSE)
smc_notrend
```

```
## Holt-Winters exponential smoothing without trend and without seasonal component.
##
## Call:
## HoltWinters(x = smc_timeseries, beta = FALSE, gamma = FALSE)
##
## Smoothing parameters:
##  alpha: 0.2562956
##  beta : FALSE
##  gamma: FALSE
##
## Coefficients:
##       [,1]
## a 230.3827
```

```
smc_notrend$SSE
```

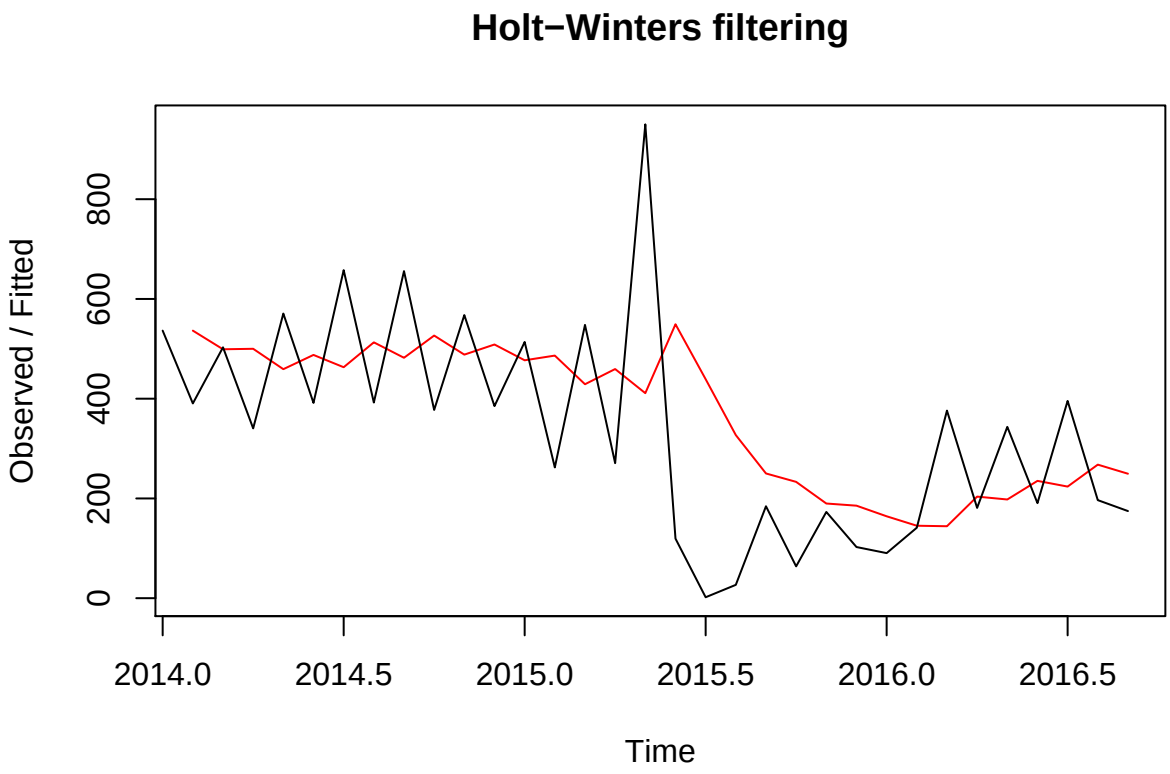
```
## [1] 1216009
```

```
smc_notrend$fitted
```

```
##              xhat    level
## Feb 2014 536.4960 536.4960
```

```
## Mar 2014 499.0786 499.0786
## Apr 2014 500.0857 500.0857
## May 2014 459.1889 459.1889
## Jun 2014 487.7726 487.7726
## Jul 2014 463.1109 463.1109
## Aug 2014 512.9863 512.9863
## Sep 2014 482.0616 482.0616
## Oct 2014 526.5754 526.5754
## Nov 2014 488.3672 488.3672
## Dec 2014 508.6866 508.6866
## Jan 2015 477.0278 477.0278
## Feb 2015 486.4349 486.4349
## Mar 2015 428.9734 428.9734
## Apr 2015 459.4643 459.4643
## May 2015 411.0977 411.0977
## Jun 2015 549.2380 549.2380
## Jul 2015 439.0786 439.0786
## Aug 2015 327.0344 327.0344
## Sep 2015 250.0785 250.0785
## Oct 2015 233.2261 233.2261
## Nov 2015 189.8119 189.8119
## Dec 2015 185.4931 185.4931
## Jan 2016 164.2464 164.2464
## Feb 2016 145.3109 145.3109
## Mar 2016 144.2799 144.2799
## Apr 2016 203.6974 203.6974
## May 2016 197.8989 197.8989
## Jun 2016 235.2225 235.2225
## Jul 2016 223.7857 223.7857
## Aug 2016 267.7989 267.7989
## Sep 2016 249.5635 249.5635
```

```r
plot(smc_notrend)
```



Simple exponential smoothing for seasonally adjusted ts assuming no trend - plotting seasonally adjusted data forecast gives us slightly better sum of squared errors

```
smc_adjusted <- HoltWinters(smcseasonallyadjusted, beta=FALSE, gamma=FALSE)
smc_adjusted
```

```
## Holt-Winters exponential smoothing without trend and without seasonal component.
##
## Call:
## HoltWinters(x = smcseasonallyadjusted, beta = FALSE, gamma = FALSE)
##
## Smoothing parameters:
##  alpha: 0.2935704
##  beta : FALSE
##  gamma: FALSE
##
## Coefficients:
##       [,1]
## a 231.4983
```

```
smc_adjusted$SSE
```

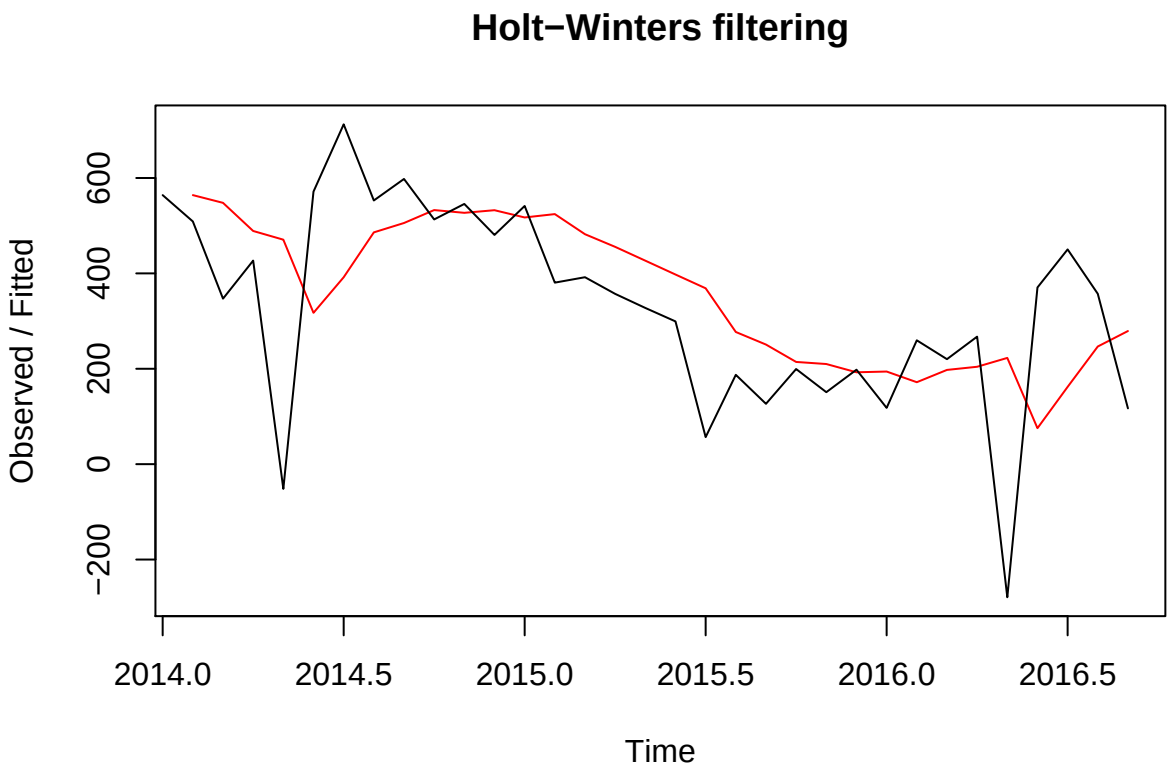
```
## [1] 1165723
```

```
smc_adjusted$fitted
```

```
##                xhat     level
## Feb 2014 564.10588 564.10588
```

```
## Mar 2014 547.88757 547.88757
## Apr 2014 488.91380 488.91380
## May 2014 470.67652 470.67652
## Jun 2014 317.27494 317.27494
## Jul 2014 391.88323 391.88323
## Aug 2014 485.99792 485.99792
## Sep 2014 505.62658 505.62658
## Oct 2014 532.73181 532.73181
## Nov 2014 526.97222 526.97222
## Dec 2014 532.43208 532.43208
## Jan 2015 517.21316 517.21316
## Feb 2015 524.29664 524.29664
## Mar 2015 482.10945 482.10945
## Apr 2015 455.63720 455.63720
## May 2015 426.68742 426.68742
## Jun 2015 397.57184 397.57184
## Jul 2015 368.71948 368.71948
## Aug 2015 277.11059 277.11059
## Sep 2015 250.74656 250.74656
## Oct 2015 214.28264 214.28264
## Nov 2015 209.92842 209.92842
## Dec 2015 192.59454 192.59454
## Jan 2016 194.18826 194.18826
## Feb 2016 171.81427 171.81427
## Mar 2016 197.59901 197.59901
## Apr 2016 204.20669 204.20669
## May 2016 222.74296 222.74296
## Jun 2016  75.43137  75.43137
## Jul 2016 162.04480 162.04480
## Aug 2016 246.65967 246.65967
## Sep 2016 279.10599 279.10599
```

```
plot(smc_adjusted)
```

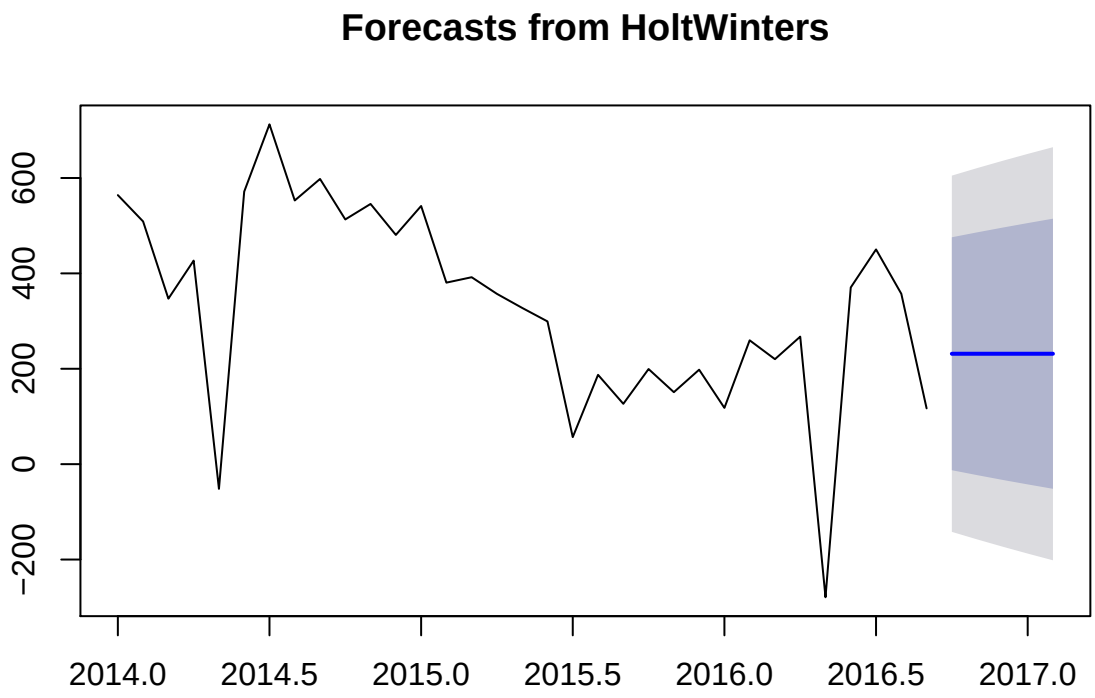


Predicting for next 5 months

```
smc_adjusted_forecasts <- forecast.HoltWinters(smc_adjusted, h=5)
smc_adjusted_forecasts
```

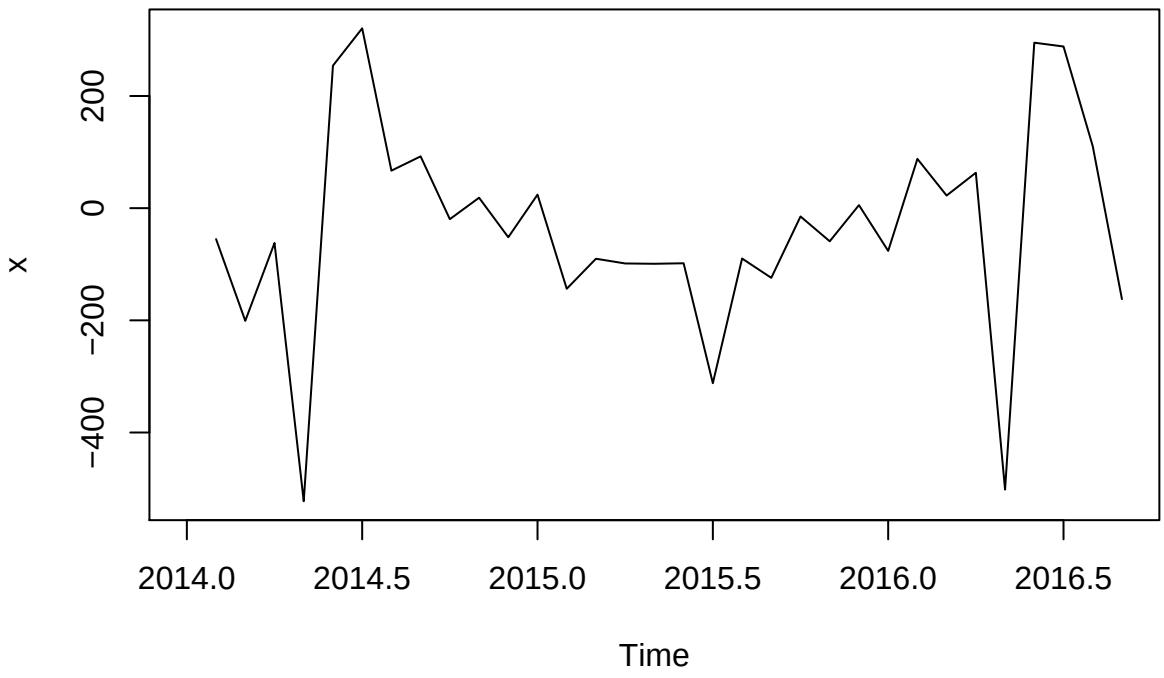
```
##          Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## Oct 2016       231.4983 -12.70362 475.7002 -141.9763 604.9729
## Nov 2016       231.4983 -23.00926 486.0059 -157.7374 620.7340
## Dec 2016       231.4983 -32.91354 495.9101 -172.8847 635.8813
## Jan 2017       231.4983 -42.45998 505.4566 -187.4847 650.4813
## Feb 2017       231.4983 -51.68479 514.6814 -201.5929 664.5895
```

```
plot.forecast(smc_adjusted_forecasts)
```



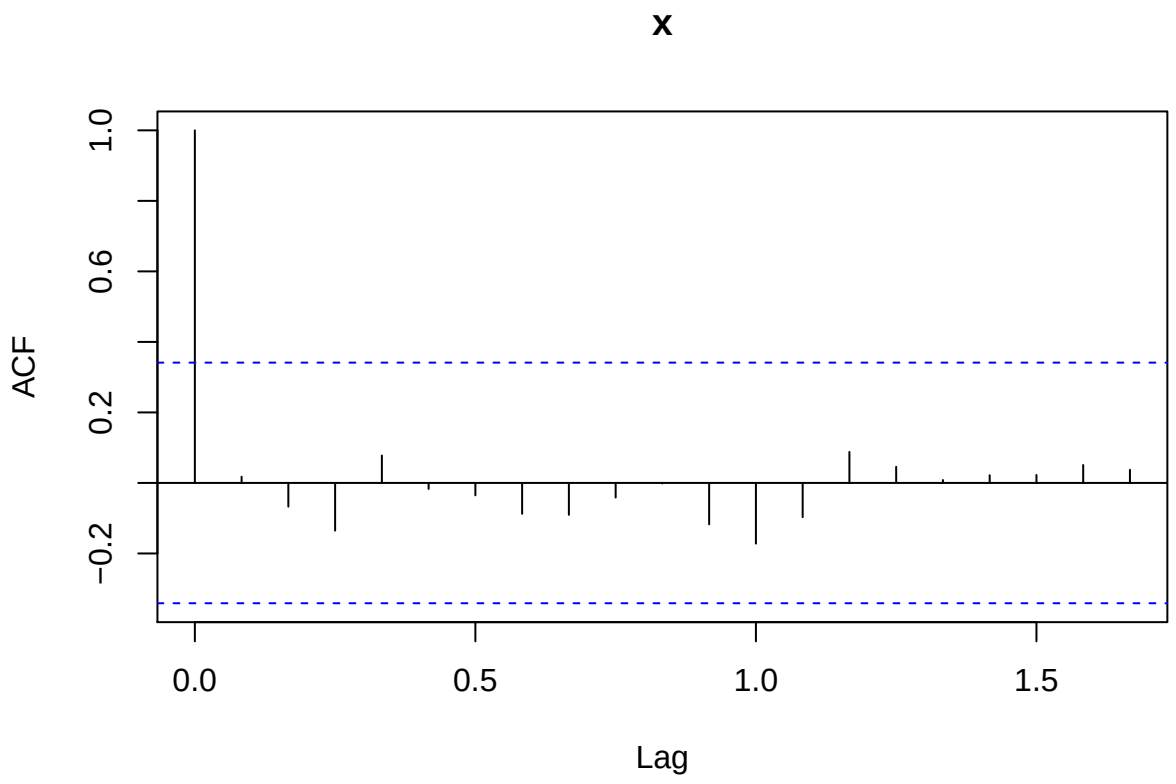
Lets see residuals(errors)

```
plot(smc_adjusted_forecasts$residuals)
```



With acf and Ljung-Box tests, we can see that there are no non-zero auto correlations as the p value is very high

```
acf(smc_adjusted_forecasts$residuals, lag.max = 20,na.action = na.pass)
```



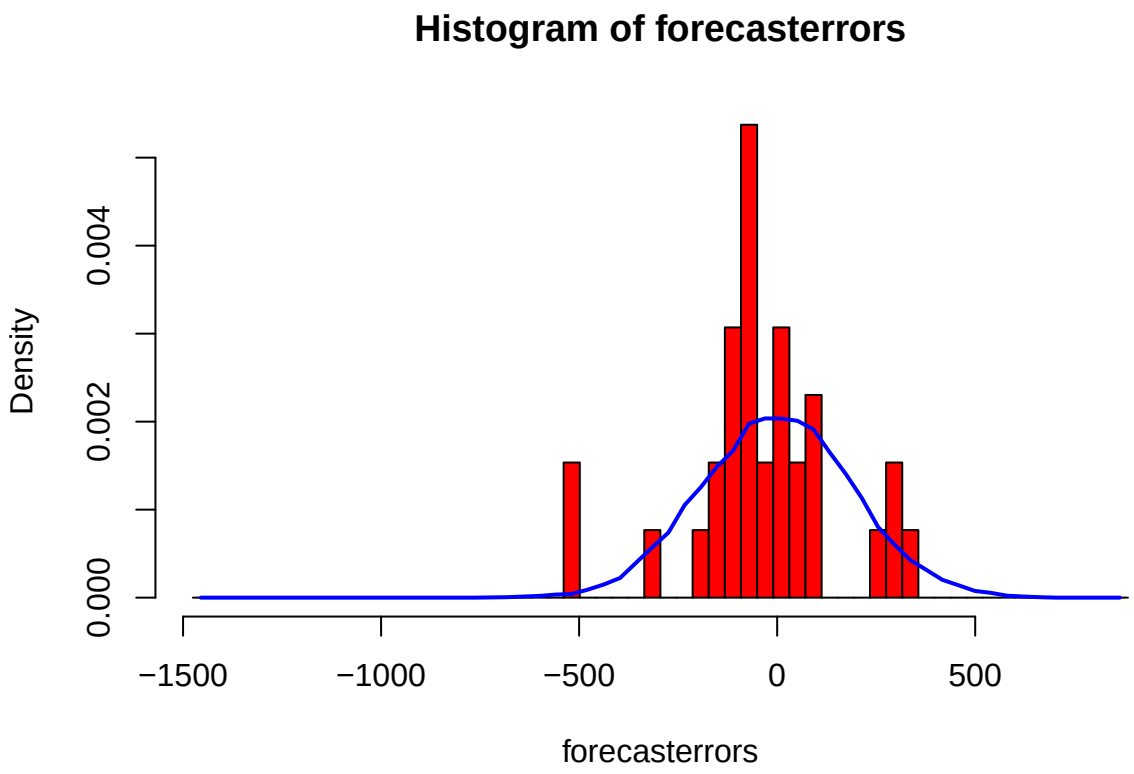

```
Box.test(smc_adjusted_forecasts$residuals, lag=20, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  smc_adjusted_forecasts$residuals
## X-squared = 5.8327, df = 20, p-value = 0.9991
```

Function to check whether forecast errors are normally distributed with mean zero

```
plotForecastErrors <- function(forecasterrors)
{
  # make a histogram of the forecast errors:
  mybinsize <- IQR(forecasterrors,na.rm=TRUE)/4
  mysd   <- sd(forecasterrors,na.rm=TRUE)
  mymin  <- min(forecasterrors,na.rm=TRUE) - mysd*5
  mymax  <- max(forecasterrors,na.rm=TRUE) + mysd*3
  # generate normally distributed data with mean 0 and standard deviation mysd
  mynorm <- rnorm(10000, mean=0, sd=mysd)
  mymin2 <- min(mynorm)
  mymax2 <- max(mynorm)
  if (mymin2 < mymin) { mymin <- mymin2 }
  if (mymax2 > mymax) { mymax <- mymax2 }
  # make a red histogram of the forecast errors, with the normally distributed data overlaid:
  mybins <- seq(mymin, mymax, mybinsize)
  hist(forecasterrors, col="red", freq=FALSE, breaks=mybins)
  # freq=FALSE ensures the area under the histogram = 1
```

```
  # generate normally distributed data with mean 0 and standard deviation mysd
  myhist <- hist(mynorm, plot=FALSE, breaks=mybins)
  # plot the normal curve as a blue line on top of the histogram of forecast errors:
  points(myhist$mids, myhist$density, type="l", col="blue", lwd=2)
}
plotForecastErrors(smc_adjusted_forecasts$residuals)
```



The plot shows that the distribution of forecast errors is roughly centred on zero, and is more or less normally distributed. The Ljung-Box test showed that there is little evidence of non-zero autocorrelations in the in-sample forecast errors, and the distribution of forecast errors seems to be normally distributed with mean zero. This suggests that the simple exponential smoothing method provides an adequate predictive model for water usage. Furthermore, the assumptions that the 80% and 95% predictions intervals were based upon (that there are no autocorrelations in the forecast errors, and the forecast errors are normally distributed with mean zero and constant variance) are probably valid.

Holts exponential model(assuming there is a trend and no seasonality)

```
smc_trend <- HoltWinters(smc_timeseries, gamma=FALSE)
smc_trend
```

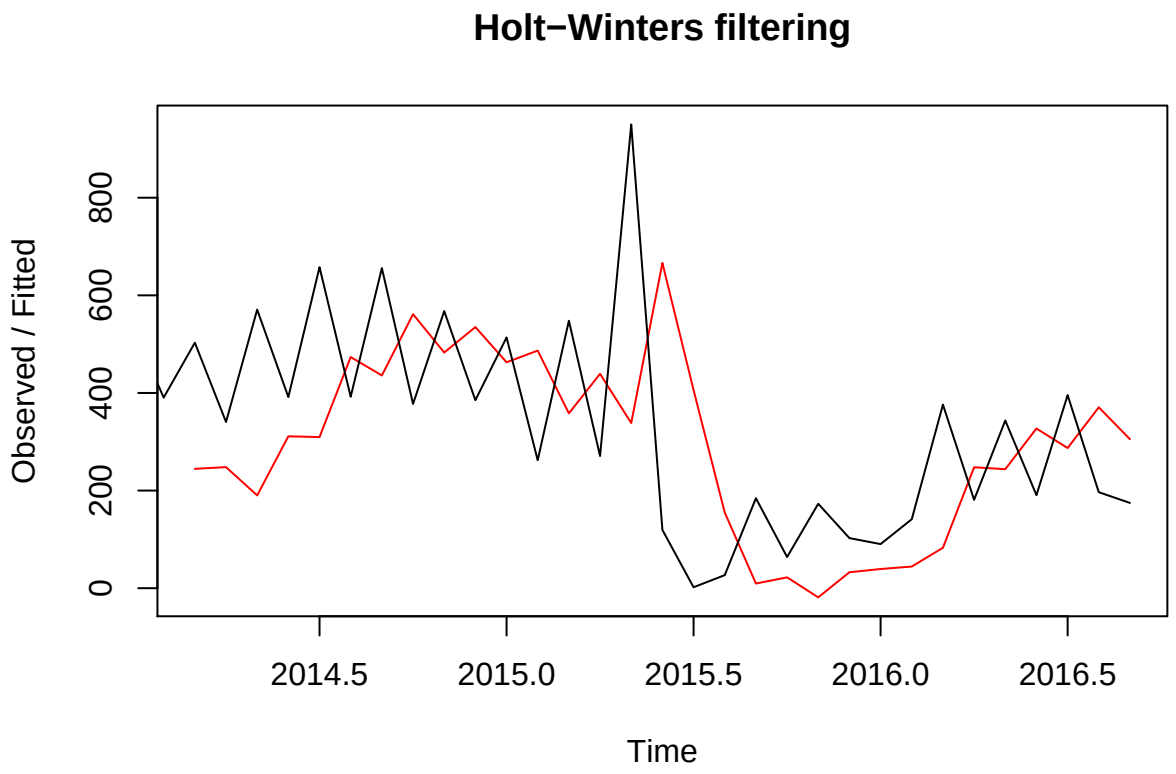
```
## Holt-Winters exponential smoothing with trend and without seasonal component.
##
## Call:
## HoltWinters(x = smc_timeseries, gamma = FALSE)
##
## Smoothing parameters:
##  alpha: 0.4445594
```

```
##  beta : 0.3009736
##  gamma: FALSE
##
## Coefficients:
##         [,1]
## a 247.206318
## b  -5.495393
```

```
smc_trend$SSE
```

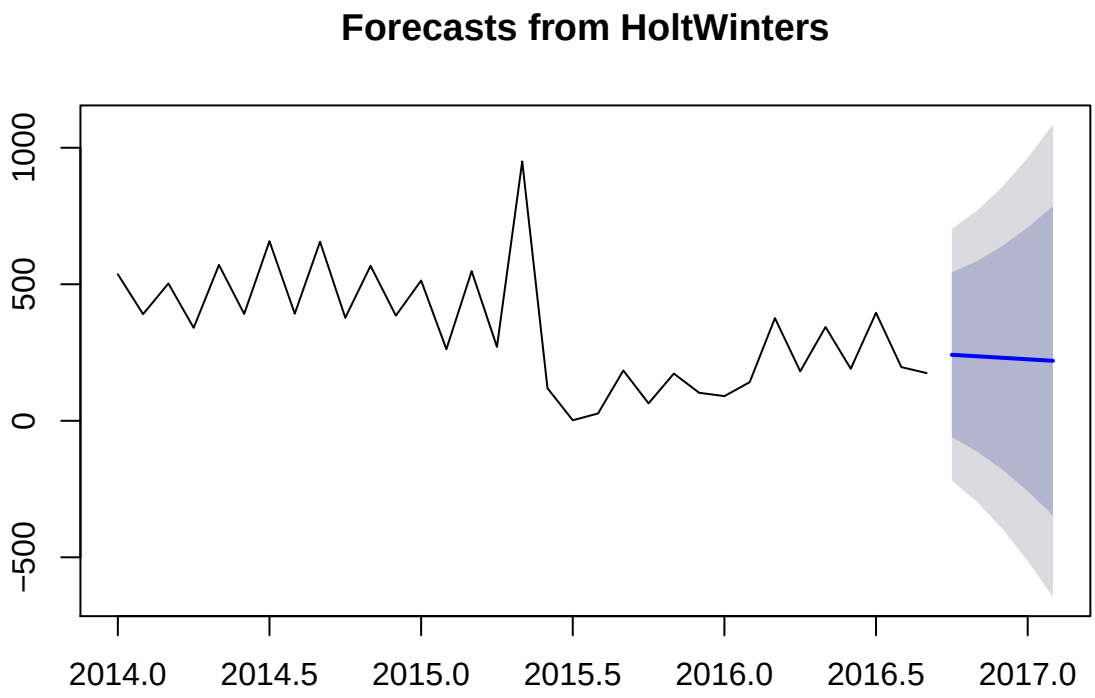
```
## [1] 1700652
```

```
plot(smc_trend)
```



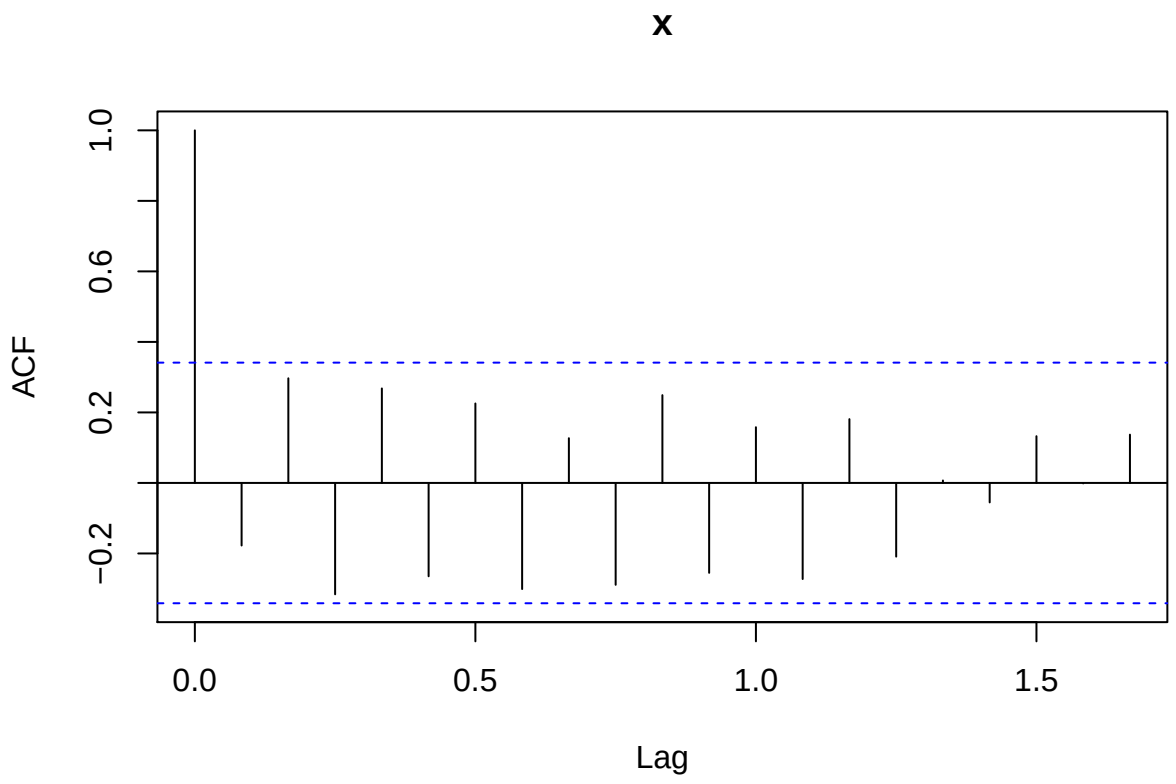
Lets predict for the next 5 months

```
smc_trend <- forecast.HoltWinters(smc_trend, h=5)
plot.forecast(smc_trend)
```



Lets check the residuals

```
acf(smc_trend$residuals, lag.max = 20,na.action = na.pass)
```



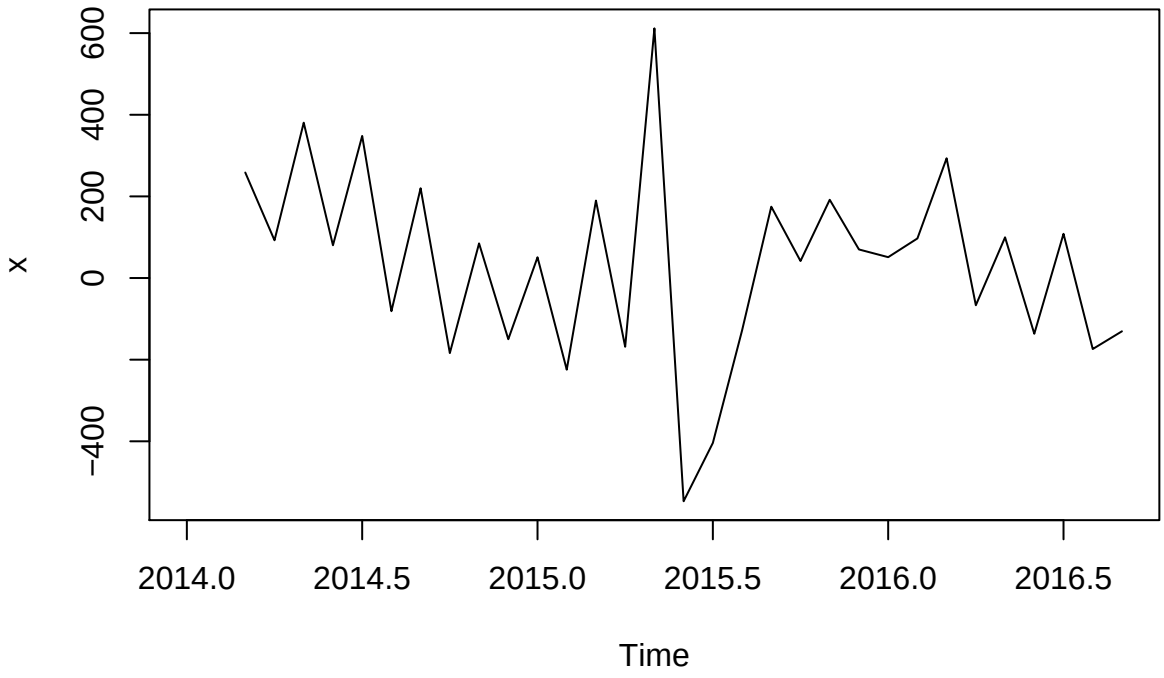

```
Box.test(smc_trend$residuals, lag=20, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  smc_trend$residuals
## X-squared = 43.882, df = 20, p-value = 0.001561
```

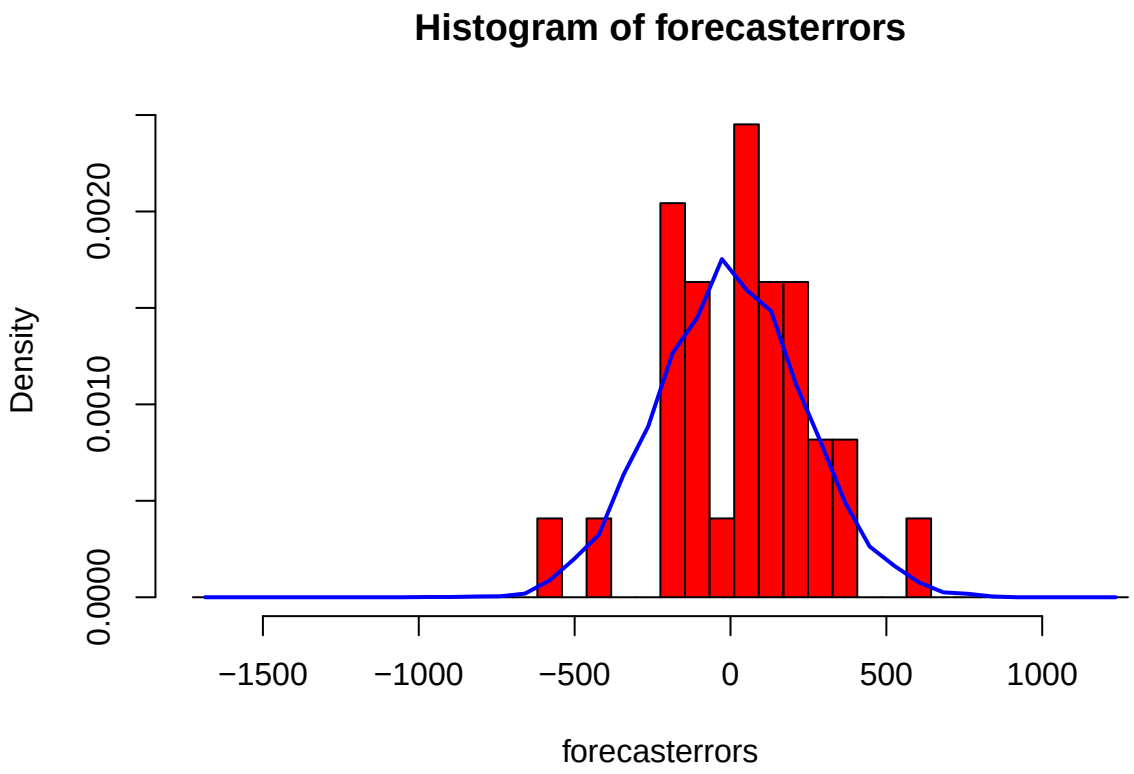
p-value is very low - indicates evidence of non-zero auto correlations. Model could be improved.

Lets check for constant variance and normal distribution with mean zero in residuals

```
plot.ts(smc_trend$residuals)            # make a time plot
```




```
plotForecastErrors(smc_trend$residuals) # make a histogram
```




```
mean(smc_trend$residuals,na.rm=TRUE)
```

```
## [1] 33.87264
```

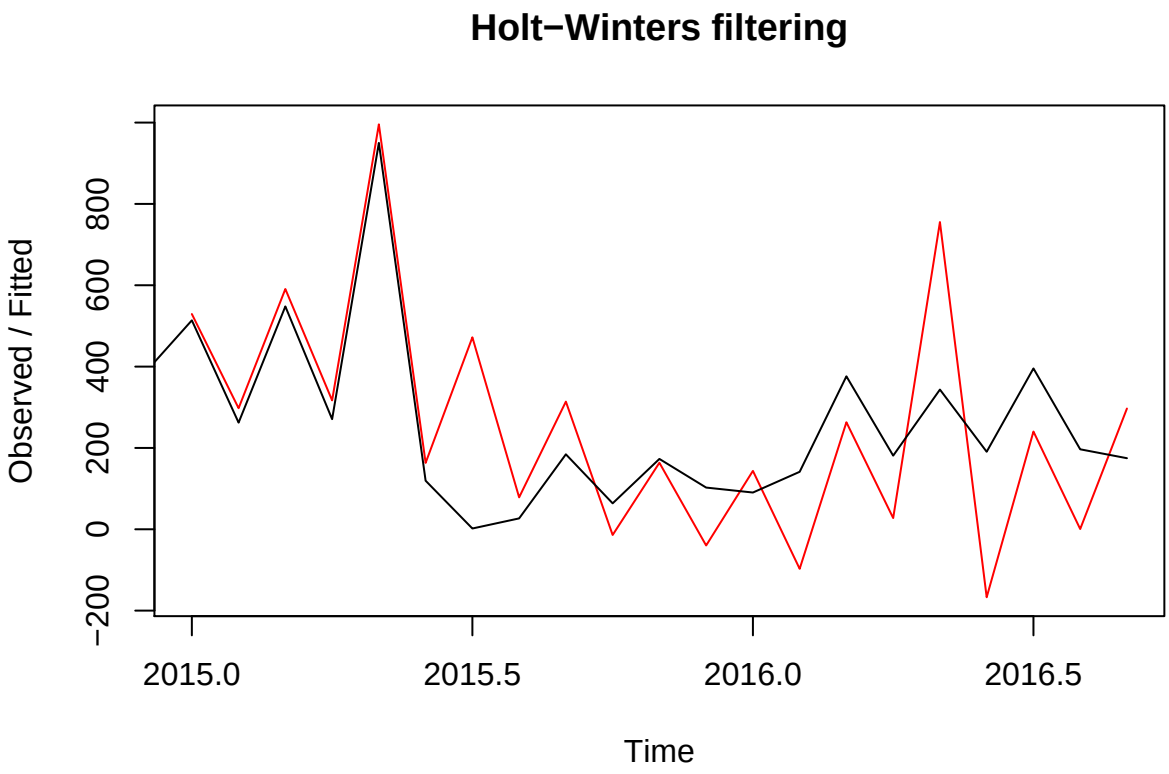
If we observe carefully, mean is to the right of zero. This model shoud not be considered.

Holt-Winter's Exponential smoothing (assuming there is trend and seasonality)

```
smc_trend_seasonality <- HoltWinters(smc_timeseries)

smc_trend_seasonality$SSE
```

```
## [1] 747350.6
```

```
plot(smc_trend_seasonality)
```

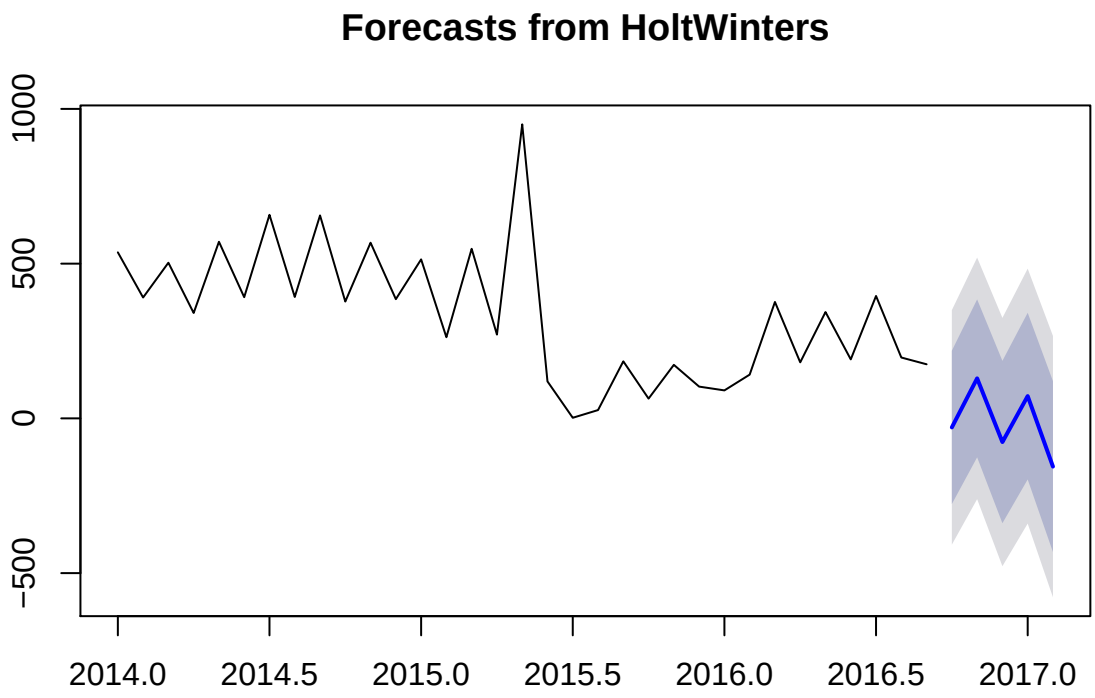


Predicting for the next 5 months

```
smc_trend_seasonality <- forecast.HoltWinters(smc_trend_seasonality, h=5)
smc_trend_seasonality
```

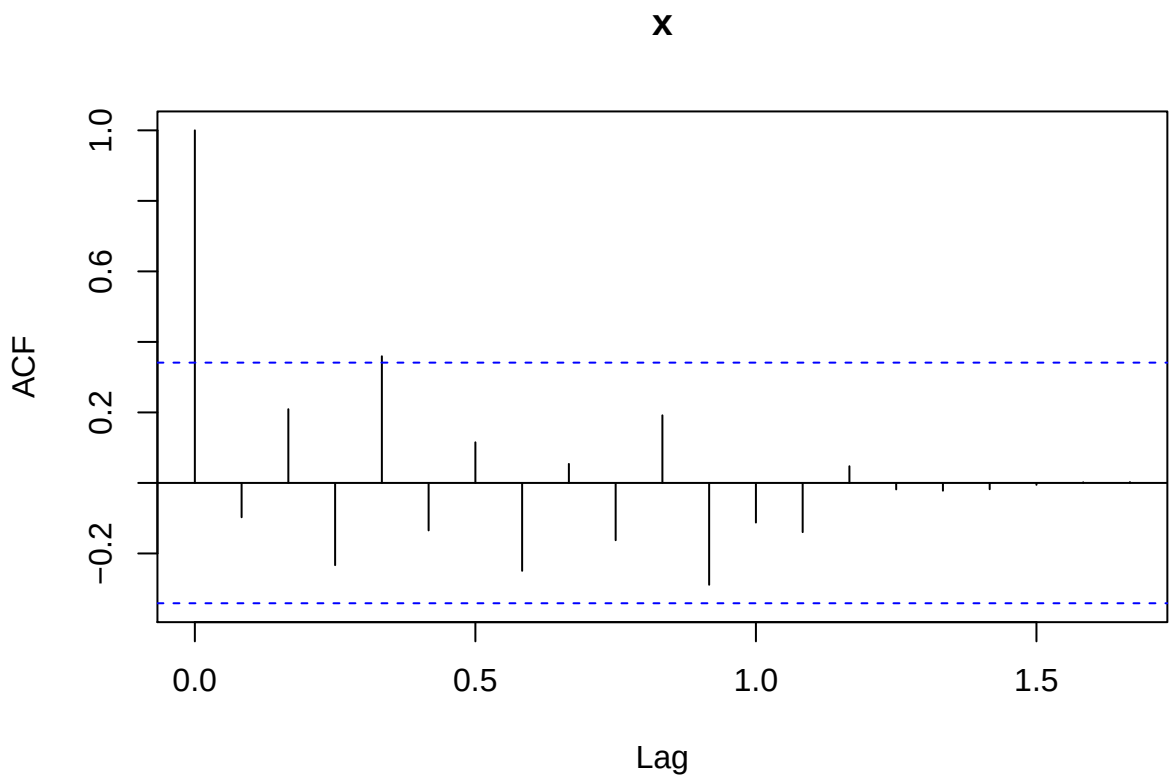
```
##          Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## Oct 2016      -29.17233 -276.9003 218.5556 -408.0395 349.6949
## Nov 2016      129.27206 -125.8318 384.3759 -260.8757 519.4198
## Dec 2016      -76.48897 -338.7614 185.7835 -477.6001 324.6221
## Jan 2017       71.93817 -197.3121 341.1884 -339.8445 483.7209
## Feb 2017     -155.80882 -431.8605 120.2429 -577.9935 266.3758
```

```
plot(smc_trend_seasonality)
```



Checking Residuals

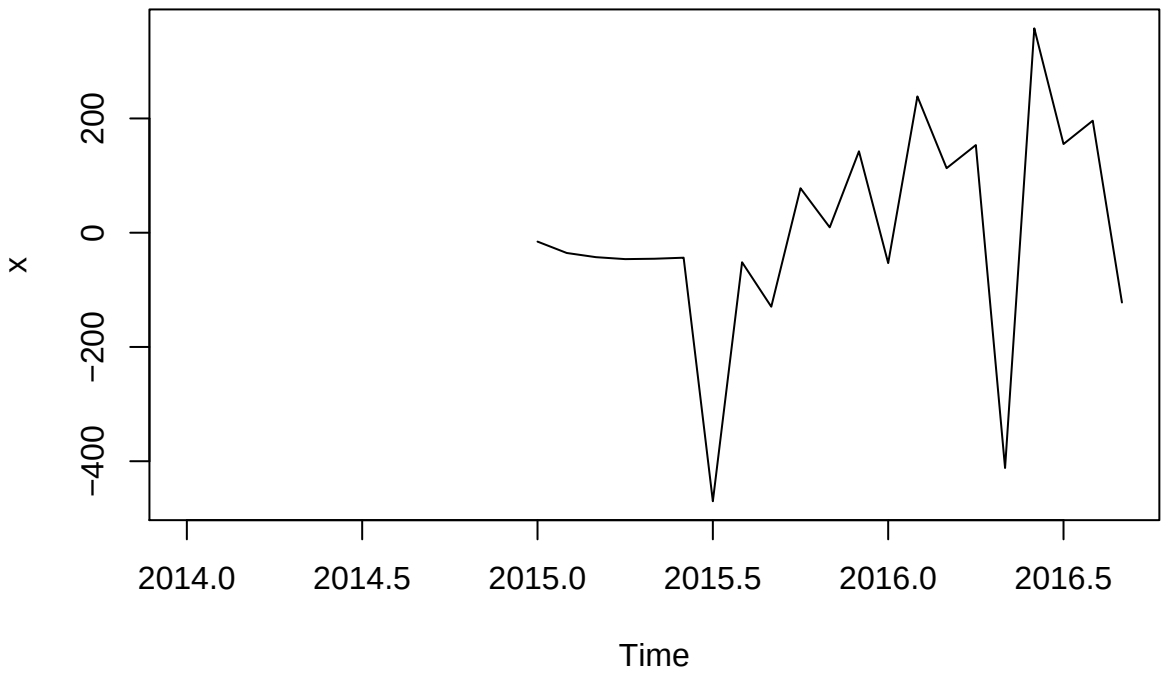
```
acf(smc_trend_seasonality$residuals, lag.max=20,na.action = na.pass)
```



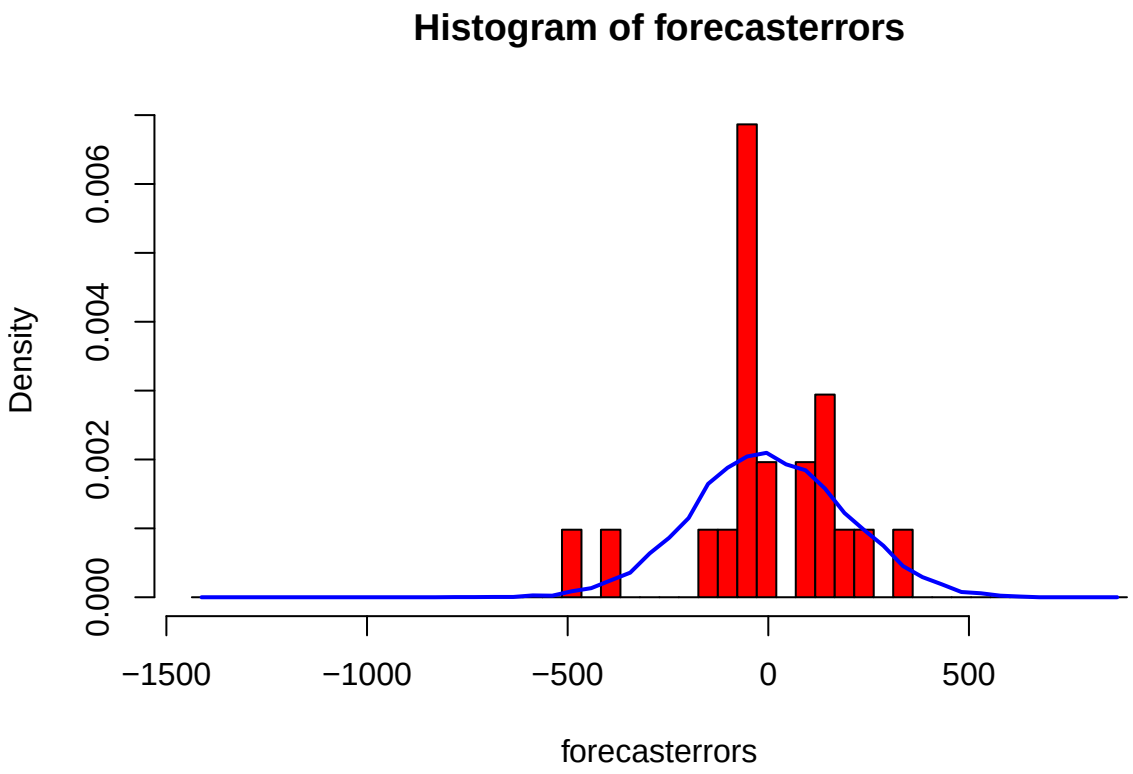

```
Box.test(smc_trend_seasonality$residuals, lag=20, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  smc_trend_seasonality$residuals
## X-squared = 18.524, df = 20, p-value = 0.5529
```

```
plot.ts(smc_trend_seasonality$residuals)            # make a time plot
```




```
plotForecastErrors(smc_trend_seasonality$residuals) # make a histogram
```

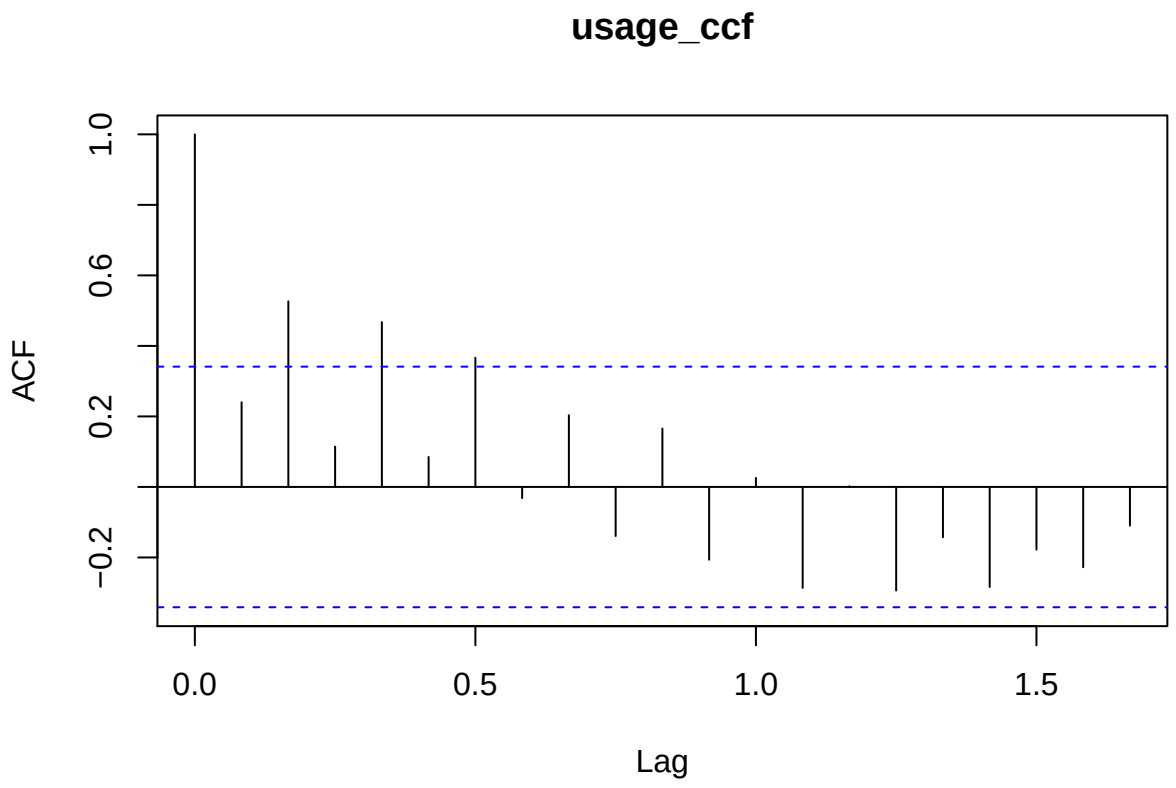


p-value is high and residuals are normally distributed with mean 0 - residuals are independent

ARIMA Model

ARIMA allows for non-zero auto correlations to exist

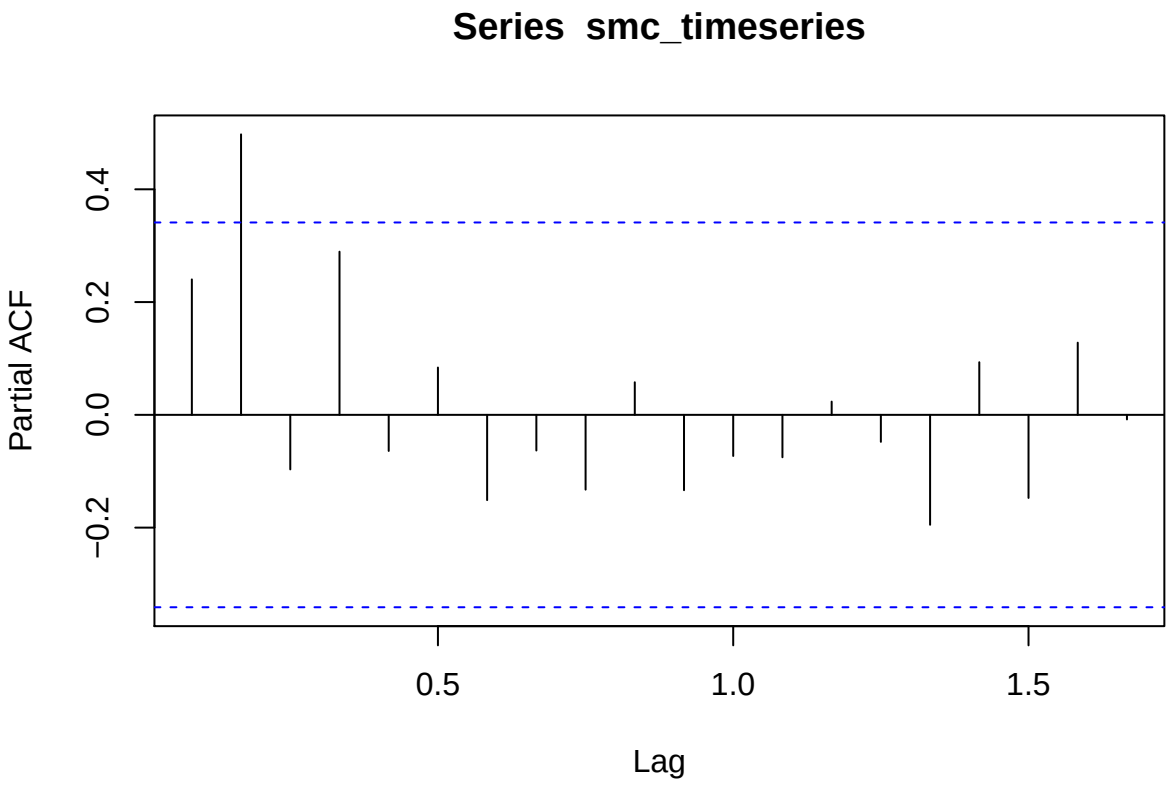
```
acf(smc_timeseries, lag.max=20)             # plot a correlogram
```




```
acf(smc_timeseries, lag.max=20, plot=FALSE) # get the autocorrelation values
```

```
##
## Autocorrelations of series 'smc_timeseries', by lag
##
## 0.0000 0.0833 0.1667 0.2500 0.3333 0.4167 0.5000 0.5833 0.6667 0.7500
##  1.000  0.240  0.526  0.114  0.467  0.085  0.366 -0.032  0.204 -0.139
## 0.8333 0.9167 1.0000 1.0833 1.1667 1.2500 1.3333 1.4167 1.5000 1.5833
##  0.166 -0.206  0.026 -0.286  0.002 -0.294 -0.143 -0.284 -0.178 -0.228
## 1.6667
## -0.110
```

```
pacf(smc_timeseries, lag.max=20)
```




```
pacf(smc_timeseries, lag.max=20, plot=FALSE) #get partial autocorrelation values
```

```
##
## Partial autocorrelations of series 'smc_timeseries', by lag
##
## 0.0833 0.1667 0.2500 0.3333 0.4167 0.5000 0.5833 0.6667 0.7500 0.8333
##  0.240  0.497 -0.097  0.289 -0.064  0.084 -0.151 -0.063 -0.133  0.058
## 0.9167 1.0000 1.0833 1.1667 1.2500 1.3333 1.4167 1.5000 1.5833 1.6667
## -0.134 -0.073 -0.075  0.024 -0.048 -0.195  0.093 -0.147  0.128 -0.008
```

We can select a model(p,d,q) based on the above information or go with a nice function auto.arima() to select optimal model

Passing original timeseries without any adjustments or differences. Auto Arima takes care of differences and selects optimal model

```
smctrain <- auto.arima(smc_ts_matrix, trace = TRUE)
```

```
##
##  ARIMA(2,1,2) with drift         : Inf
##  ARIMA(0,1,0) with drift         : 451.4597
##  ARIMA(1,1,0) with drift         : 433.0462
##  ARIMA(0,1,1) with drift         : Inf
##  ARIMA(0,1,0)                    : 449.2391
##  ARIMA(2,1,0) with drift         : 435.6649
##  ARIMA(1,1,1) with drift         : 435.3071
##  ARIMA(2,1,1) with drift         : Inf
##  ARIMA(1,1,0)                    : 430.8259
```

```
##  ARIMA(2,1,0)                    : 433.2664
##  ARIMA(1,1,1)                    : 433.0409
##  ARIMA(2,1,1)                    : 433.9095
##
##  Best model: ARIMA(1,1,0)
```

```
smctrain
```

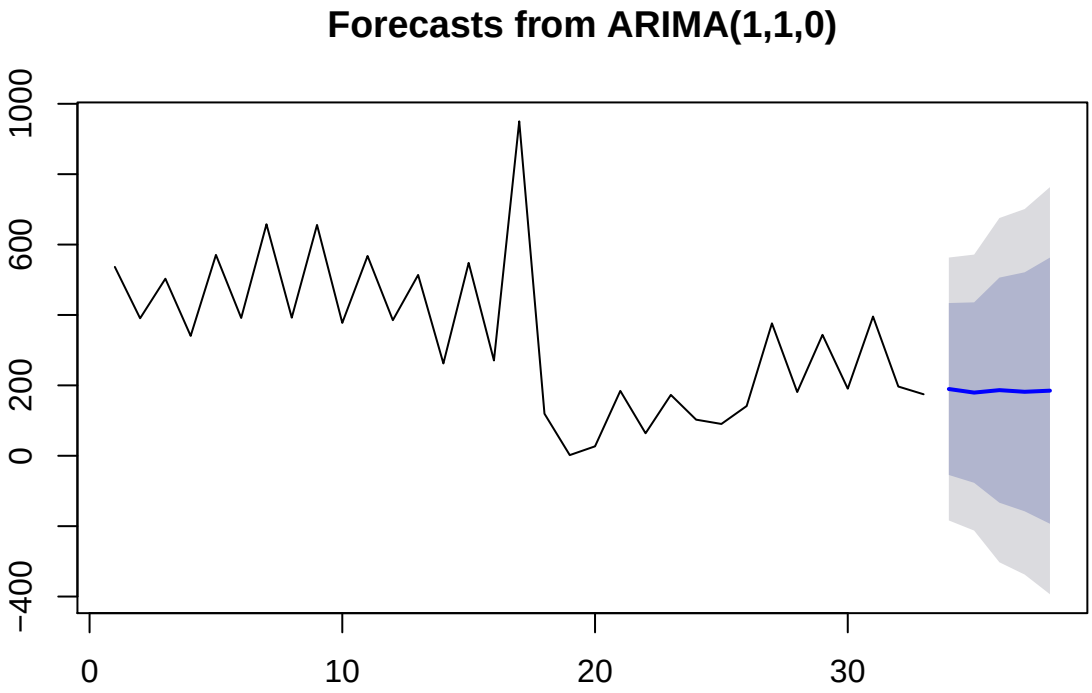
```
## Series: smc_ts_matrix
## ARIMA(1,1,0)
##
## Coefficients:
##           ar1
##       -0.6806
## s.e.   0.1226
##
## sigma^2 estimated as 36309:  log likelihood=-213.21
## AIC=430.41   AICc=430.83   BIC=433.34
```

Prediction

```
smcprediction <- forecast.Arima(smctrain, h=5)
smcprediction
```

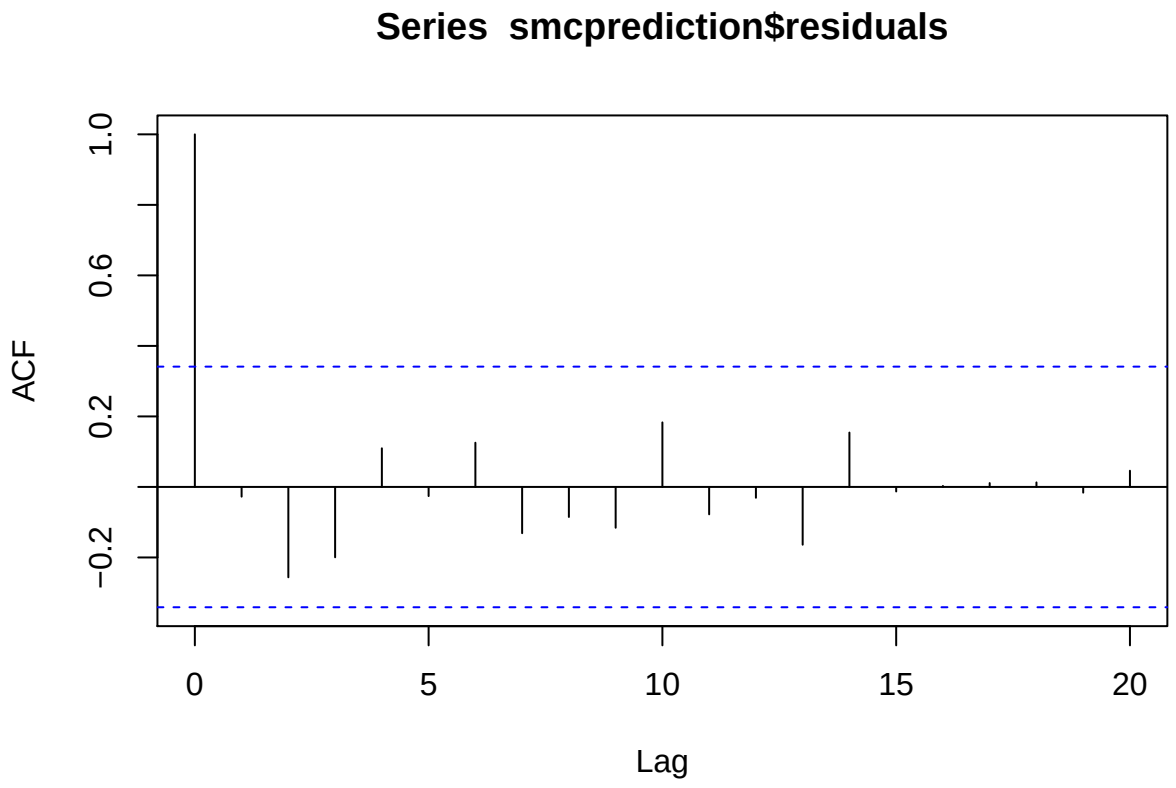
```
##    Point Forecast      Lo 80    Hi 80     Lo 95    Hi 95
## 34       189.6463  -54.55090 433.8436 -183.8211 563.1138
## 35       179.4910  -76.86038 435.8423 -212.5646 571.5465
## 36       186.4026 -133.34706 506.1523 -302.6123 675.4176
## 37       181.6986 -157.80836 521.2056 -337.5325 700.9297
## 38       184.9001 -193.24641 563.0467 -393.4251 763.2254
```

```
plot.forecast(smcprediction)
```



Checking residuals

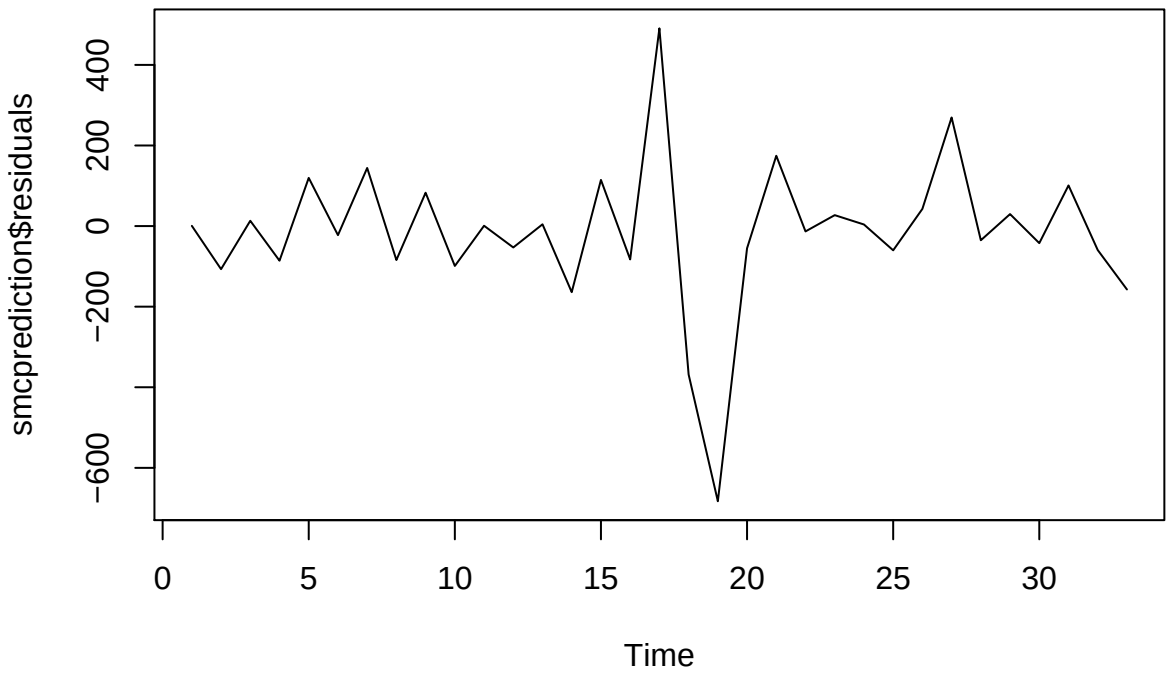
```
acf(smcprediction$residuals, lag.max=20)
```



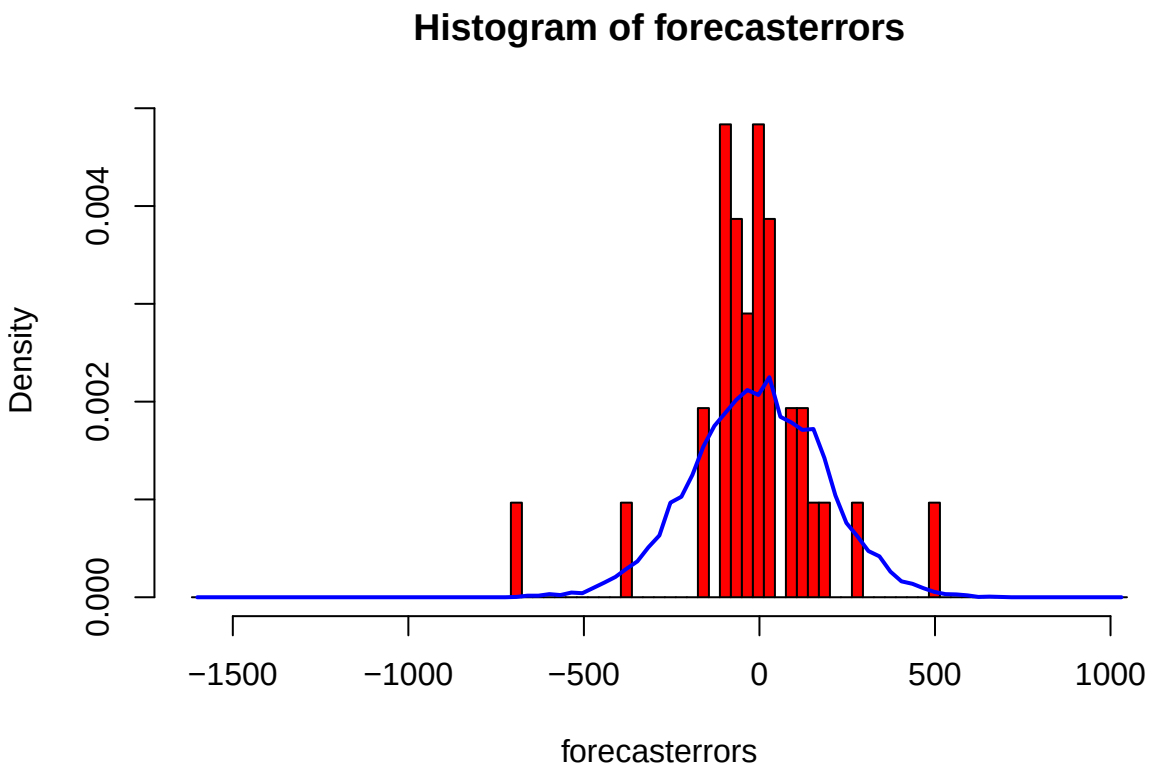

```
Box.test(smcprediction$residuals, lag=20, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  smcprediction$residuals
## X-squared = 12.222, df = 20, p-value = 0.9082
```

```
plot.ts(smcprediction$residuals)            # make time plot of forecast errors
```




```
plotForecastErrors(smcprediction$residuals) # make a histogram
```




```
mean(smcprediction$residuals)
```

```
## [1] -16.77279
```

The successive forecast errors do not seem to be correlated, but the forecast errors do not seem to be normally distributed with mean zero and constant variance, the ARIMA(1,1,0) does not seem to be an adequate predictive model for the water usage and could be improved. This time series is too random to predict and might need fourier series transformation.