Project1

Group 4

21/10/2019

# PART A: ATM Forecast

## Overview

The purpouse of the Part A is to forecast how much cash is taken out of 4 different ATM machines for May 2010.  
The data is given in a single file which includes 3 columns: Date, ATM, Cash. ‘Cash’ is provided in hundreds of dollars. There are 4 different ATM: ATM1, ATM2, ATM3, ATM4

The following steps will be taken to forecast how much cash is taken out of 4 different ATM machines:

* data exploration and Preparation (converting data from long to wide format in order to assess and analyse cash withdrawal of each ATM separately; checking missing values, outliers)
* checking assumptions
* building model
* checking residuals
* assessing accuracy
* making predictions and saving them in scv file.

library(ggplot2)  
library(urca)  
library(fpp2)  
library(datetime)  
library(readxl)  
library(dplyr)  
library(tidyr)  
library(forecast)

# reading the data  
ATM\_df <- read\_excel("project1/ATM624Data.xlsx")  
head(ATM\_df)

## # A tibble: 6 x 3  
## DATE ATM Cash  
## <dttm> <chr> <dbl>  
## 1 2009-05-01 00:00:00 ATM1 96  
## 2 2009-05-01 00:00:00 ATM2 107  
## 3 2009-05-02 00:00:00 ATM1 82  
## 4 2009-05-02 00:00:00 ATM2 89  
## 5 2009-05-03 00:00:00 ATM1 85  
## 6 2009-05-03 00:00:00 ATM2 90

summary(ATM\_df)

## DATE ATM Cash   
## Min. :2009-05-01 00:00:00 Length:1474 Min. : 0.00   
## 1st Qu.:2009-08-01 00:00:00 Class :character 1st Qu.: 1.00   
## Median :2009-11-01 00:00:00 Mode :character Median : 73.00   
## Mean :2009-10-31 19:11:48 Mean : 58.53   
## 3rd Qu.:2010-02-01 00:00:00 3rd Qu.: 96.50   
## Max. :2010-05-14 00:00:00 Max. :1123.00   
## NA's :15

Data contains 3 columns: Date, ATM, Cash. We have 15 missing values (it is about 1% of data) which can be omitted or replaced with mean or median (in case of existing outliers) without compromising the quality of the forecast. We will handle missing values for each ATM separately.

Converting dataset to a wide format:

ATM\_df <- ATM\_df %>%  
 # converting data to wide format  
 spread(ATM, Cash) %>%   
 # converting DATE to format %Y-%m-%d  
 mutate(DATE = as.Date(DATE, tryFormats = "%Y-%m-%d")) %>%   
 select(-`<NA>`)  
  
head(ATM\_df)

## # A tibble: 6 x 5  
## DATE ATM1 ATM2 ATM3 ATM4  
## <date> <dbl> <dbl> <dbl> <dbl>  
## 1 2009-05-01 96 107 0 96  
## 2 2009-05-02 82 89 0 82  
## 3 2009-05-03 85 90 0 85  
## 4 2009-05-04 90 55 0 90  
## 5 2009-05-05 99 79 0 99  
## 6 2009-05-06 88 19 0 88

summary(ATM\_df)

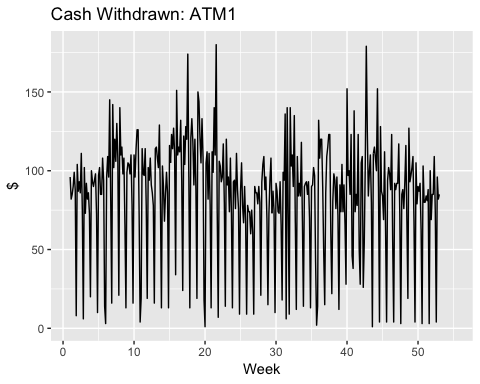
## DATE ATM1 ATM2 ATM3   
## Min. :2009-05-01 Min. : 1.0 Min. : 0.00 Min. : 0.0000   
## 1st Qu.:2009-08-03 1st Qu.: 73.0 1st Qu.: 25.00 1st Qu.: 0.0000   
## Median :2009-11-06 Median : 91.0 Median : 66.50 Median : 0.0000   
## Mean :2009-11-06 Mean : 84.1 Mean : 62.46 Mean : 0.7206   
## 3rd Qu.:2010-02-08 3rd Qu.:108.0 3rd Qu.: 93.00 3rd Qu.: 0.0000   
## Max. :2010-05-14 Max. :180.0 Max. :147.00 Max. :96.0000   
## NA's :14 NA's :15 NA's :14   
## ATM4   
## Min. : 1.00   
## 1st Qu.: 73.00   
## Median : 91.00   
## Mean : 86.84   
## 3rd Qu.: 108.00   
## Max. :1123.00   
## NA's :14

Missing values of ATM1, ATM2 will be replaced with mean.

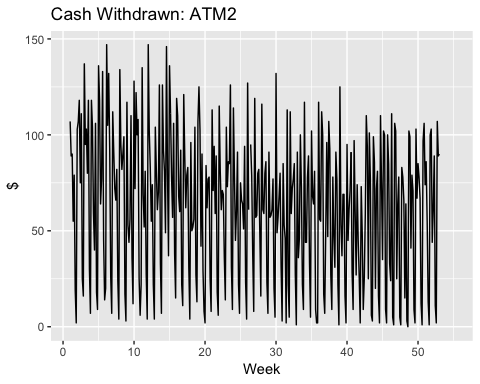
Missing values of ATM4 will be replaced with mean after replacing outlier with the median.

Converting dataset to time series object for each ATM

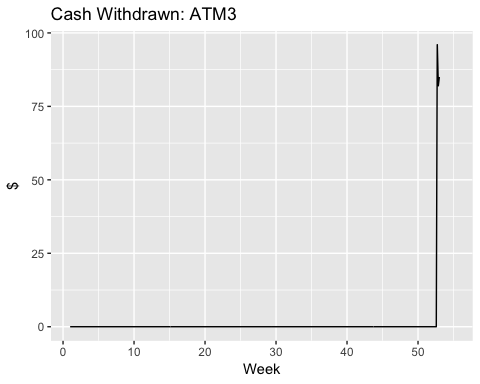
# converting to ts, using frequency = 7 (weekly).  
ATM\_ts <- ts(ATM\_df %>% select(-DATE), frequency = 7)  
  
ATM1 <- ATM\_ts[, "ATM1"]  
ATM2 <- ATM\_ts[, "ATM2"]  
ATM3 <- ATM\_ts[, "ATM3"]  
ATM4 <- ATM\_ts[, "ATM4"]  
  
autoplot(ATM1) + labs(title = "Cash Withdrawn: ATM1", x = "Week", y = "$")



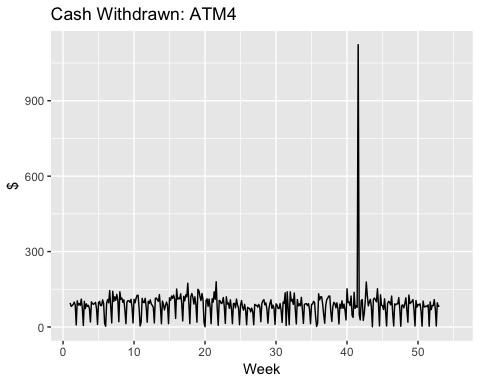
autoplot(ATM2) + labs(title = "Cash Withdrawn: ATM2", x = "Week", y = "$")



autoplot(ATM3) + labs(title = "Cash Withdrawn: ATM3", x = "Week", y = "$")



autoplot(ATM4) + labs(title = "Cash Withdrawn: ATM4", x = "Week", y = "$")



From the graph above we can conclude the following:

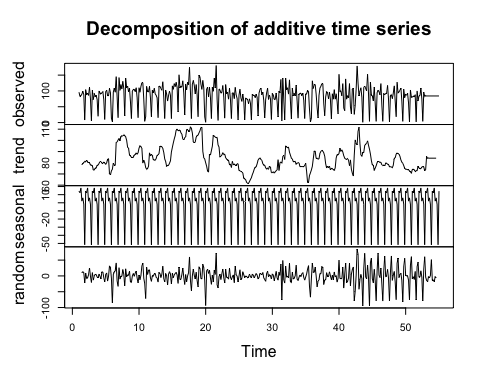
* ATM1, ATM2, ATM4 showed high frequency usage during a year.
* ATM3 showed activity only for 3 days.
* ATM4 has an outlier which should be deleted or replaced (with median).

ARIMA model will be used for making forecast for ATM1, ATM2, ATM4. ATM3 does not have enough data to make a forecast based on ARIMA model as it is considered as a rule of thumb that at least 50-100 observations is required in order to apply ARIMA model. Possible methods that can be applied for ATM3 forecast are:

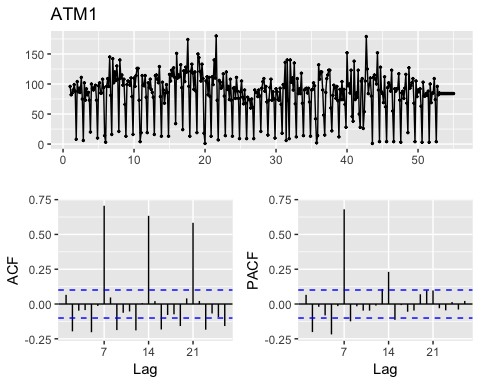
* the historical mean
* the historical median for added robustness
* the random walk (forecast the last observation out)

### Forecasting ATM1

# replacing missing values with mean  
ATM1[which(is.na(ATM1))] <- mean(ATM1, na.rm = TRUE)  
# checking components of ts  
ATM1.components.ts = decompose(ATM1)  
plot(ATM1.components.ts)



# checking stationarity of data  
ggtsdisplay(ATM1)



Data looks stationary as there is no clear trend and ACF plot shows that the autocorrelation values drops quickly, although some values are bigger than critical value.

Checking stationarity of data further using unit root test.

# checking with unit root test  
ATM1 %>% ur.kpss() %>% summary()

##   
## #######################   
## # KPSS Unit Root Test #   
## #######################   
##   
## Test is of type: mu with 5 lags.   
##   
## Value of test-statistic is: 0.4784   
##   
## Critical value for a significance level of:   
## 10pct 5pct 2.5pct 1pct  
## critical values 0.347 0.463 0.574 0.739

We should reject the null hypothesis of stationarity if the value of the test statistic is more extreme than the 10%, 5% and 1% critical values. In our case we can not reject the null hypothesis.

# checking the number of first differences required   
ATM1 %>% ndiffs()

## [1] 0

ndiffs() indicates the differences are not required.

We can conclude that data is stationary and apply arima model.

Using auto.arima function to select the best model based on the lowest AIC and BIC.

auto.fit.ATM1<-auto.arima(ATM1,trace=TRUE, seasonal = TRUE)

##   
## Fitting models using approximations to speed things up...  
##   
## ARIMA(2,0,2)(1,1,1)[7] with drift : 3383.141  
## ARIMA(0,0,0)(0,1,0)[7] with drift : 3477.918  
## ARIMA(1,0,0)(1,1,0)[7] with drift : 3419.172  
## ARIMA(0,0,1)(0,1,1)[7] with drift : 3374.301  
## ARIMA(0,0,0)(0,1,0)[7] : 3475.903  
## ARIMA(0,0,1)(0,1,0)[7] with drift : 3468.075  
## ARIMA(0,0,1)(1,1,1)[7] with drift : 3373.906  
## ARIMA(0,0,1)(1,1,0)[7] with drift : 3417.451  
## ARIMA(0,0,1)(2,1,1)[7] with drift : 3386.533  
## ARIMA(0,0,1)(1,1,2)[7] with drift : 3375.763  
## ARIMA(0,0,1)(0,1,2)[7] with drift : 3369.339  
## ARIMA(0,0,0)(0,1,2)[7] with drift : 3377.741  
## ARIMA(1,0,1)(0,1,2)[7] with drift : 3371.715  
## ARIMA(0,0,2)(0,1,2)[7] with drift : 3370.45  
## ARIMA(1,0,0)(0,1,2)[7] with drift : 3371.768  
## ARIMA(1,0,2)(0,1,2)[7] with drift : 3373.731  
## ARIMA(0,0,1)(0,1,2)[7] : 3367.284  
## ARIMA(0,0,1)(0,1,1)[7] : 3372.261  
## ARIMA(0,0,1)(1,1,2)[7] : 3373.718  
## ARIMA(0,0,1)(1,1,1)[7] : 3371.865  
## ARIMA(0,0,0)(0,1,2)[7] : 3375.698  
## ARIMA(1,0,1)(0,1,2)[7] : 3369.648  
## ARIMA(0,0,2)(0,1,2)[7] : 3368.384  
## ARIMA(1,0,0)(0,1,2)[7] : 3369.713  
## ARIMA(1,0,2)(0,1,2)[7] : 3371.653  
##   
## Now re-fitting the best model(s) without approximations...  
##   
## ARIMA(0,0,1)(0,1,2)[7] : 3417.315  
##   
## Best model: ARIMA(0,0,1)(0,1,2)[7]

auto.fit.ATM1

## Series: ATM1   
## ARIMA(0,0,1)(0,1,2)[7]   
##   
## Coefficients:  
## ma1 sma1 sma2  
## 0.1780 -0.5676 -0.1324  
## s.e. 0.0543 0.0498 0.0507  
##   
## sigma^2 estimated as 557.5: log likelihood=-1704.6  
## AIC=3417.21 AICc=3417.31 BIC=3432.88

The best model is ARIMA(0,0,1)(0,1,2)[7].

(0,0,1) - values for a trend part with no differencing, where

p=0 as the lags in PACF graph are not significant and this is interpreted as trend component of the time series which is not playing a significant role.

d=0, this means that the time series provided is stationary and does not require differencing.

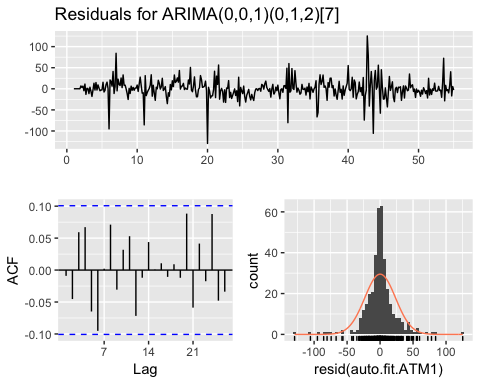
q = 1, this means that the lag after which the ACF value becomes zero in the graph is 1. This means that the time series has a randomness component and the ACF after the lag 2 becomes close to 0.

(0,1,2) - values for a seasonal part.

[7] - weekly seasonal cycle

Checking residuals of the suggested model by auto.arima() to make sure that there is no more information left for extraction.

ggtsdisplay(resid(auto.fit.ATM1), points = FALSE, plot.type = "histogram", main = "Residuals for ARIMA(0,0,1)(0,1,2)[7]")



The residuals look normally distributed and random, that means that there is no useful information is hidden in residuals to be extracted by ARIMA models.

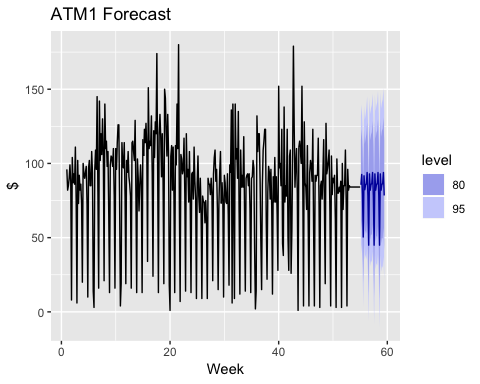
# iit is suggested to assign the value of lag as twice of natural seasonal pattern  
Box.test(resid(auto.fit.ATM1), lag = 14)

##   
## Box-Pierce test  
##   
## data: resid(auto.fit.ATM1)  
## X-squared = 15.328, df = 14, p-value = 0.3561

p-value is more than 0.05, suggesting that the residuals most likely are white noise.

Making forecast for 2 weeks (2\*7)

ATM1\_forecast <- forecast(auto.fit.ATM1, h = 31, level = c(80, 95))  
autoplot(ATM1\_forecast) + labs(title = "ATM1 Forecast", x = "Week", y = "$")



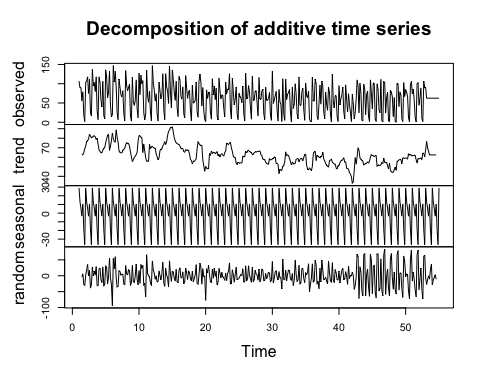
ATM1\_forecast

## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 55.14286 85.60663 55.347017 115.86625 39.328543 131.88473  
## 55.28571 92.55299 61.817592 123.28839 45.547255 139.55873  
## 55.42857 79.21076 48.475361 109.94616 32.205024 126.21650  
## 55.57143 50.47116 19.735764 81.20656 3.465427 97.47690  
## 55.71429 91.65971 60.924308 122.39511 44.653971 138.66544  
## 55.85714 82.30549 51.570090 113.04089 35.299753 129.31122  
## 56.00000 85.61957 54.884169 116.35497 38.613832 132.62530  
## 56.14286 86.26277 52.858599 119.66694 35.175499 137.35004  
## 56.28571 93.88569 60.400417 127.37097 42.674382 145.09701  
## 56.42857 78.41698 44.931698 111.90225 27.205663 129.62829  
## 56.57143 44.94378 11.458503 78.42906 -6.267532 96.15509  
## 56.71429 92.93934 59.454058 126.42461 41.728023 144.15065  
## 56.85714 81.98315 48.497871 115.46843 30.771836 133.19446  
## 57.00000 85.88543 52.400153 119.37071 34.674118 137.09674  
## 57.14286 86.33291 51.639446 121.02637 33.273837 139.39198  
## 57.28571 93.88569 59.154629 128.61676 40.769113 147.00228  
## 57.42857 78.41698 43.685910 113.14804 25.300394 131.53356  
## 57.57143 44.94378 10.212715 79.67485 -8.172801 98.06036  
## 57.71429 92.93934 58.208270 127.67040 39.822754 146.05592  
## 57.85714 81.98315 47.252083 116.71421 28.866567 135.09973  
## 58.00000 85.88543 51.154365 120.61650 32.768849 139.00201  
## 58.14286 86.33291 50.435562 122.23025 31.432656 141.23316  
## 58.28571 93.88569 57.952005 129.81938 38.929859 148.84153  
## 58.42857 78.41698 42.483286 114.35067 23.461139 133.37281  
## 58.57143 44.94378 9.010091 80.87747 -10.012056 99.89962  
## 58.71429 92.93934 57.005646 128.87303 37.983500 147.89517  
## 58.85714 81.98315 46.049459 117.91684 27.027313 136.93898  
## 59.00000 85.88543 49.951741 121.81912 30.929595 140.84127  
## 59.14286 86.33291 49.270763 123.39505 29.651250 143.01456  
## 59.28571 93.88569 56.788347 130.98304 37.150198 150.62119  
## 59.42857 78.41698 41.319628 115.51432 21.681479 135.15247

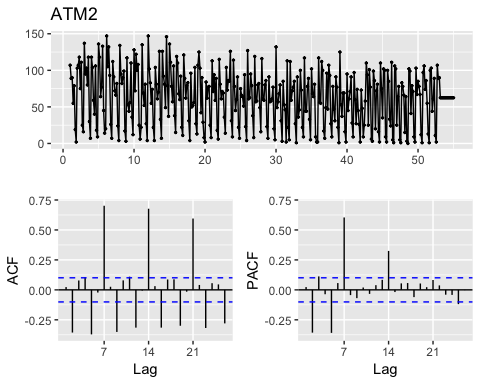
### Forecasting ATM2

Same approach is taken for forecasting ATM2.

# replacing NA with mean  
ATM2[which(is.na(ATM2))] <- mean(ATM2, na.rm = TRUE)  
# checking components of ts  
ATM2.components.ts = decompose(ATM2)  
plot(ATM2.components.ts)



# checking stationarity of data  
ggtsdisplay(ATM2)



# checking with unit root test  
ATM2 %>% ur.kpss() %>% summary()

##   
## #######################   
## # KPSS Unit Root Test #   
## #######################   
##   
## Test is of type: mu with 5 lags.   
##   
## Value of test-statistic is: 1.9267   
##   
## Critical value for a significance level of:   
## 10pct 5pct 2.5pct 1pct  
## critical values 0.347 0.463 0.574 0.739

The value of the test statistic is more extreme than the 10%, 5% and 1% critical values. In our case we can reject the null hypothesis.

# checking the number of first differences required to make the time series non-seasonal  
ATM2 %>% ndiffs()

## [1] 1

ndiffs() indicates that 1 difference is required.

auto.fit.ATM2<-auto.arima(ATM2,trace=TRUE, seasonal = TRUE)

##   
## Fitting models using approximations to speed things up...  
##   
## ARIMA(2,0,2)(1,1,1)[7] with drift : 3398.856  
## ARIMA(0,0,0)(0,1,0)[7] with drift : 3523.777  
## ARIMA(1,0,0)(1,1,0)[7] with drift : 3435.667  
## ARIMA(0,0,1)(0,1,1)[7] with drift : 3406.682  
## ARIMA(0,0,0)(0,1,0)[7] : 3521.755  
## ARIMA(2,0,2)(0,1,1)[7] with drift : 3391.583  
## ARIMA(2,0,2)(0,1,0)[7] with drift : 3532.074  
## ARIMA(2,0,2)(0,1,2)[7] with drift : 3393.474  
## ARIMA(2,0,2)(1,1,0)[7] with drift : 3432.742  
## ARIMA(2,0,2)(1,1,2)[7] with drift : 3394.898  
## ARIMA(1,0,2)(0,1,1)[7] with drift : 3407.559  
## ARIMA(2,0,1)(0,1,1)[7] with drift : 3406.649  
## ARIMA(3,0,2)(0,1,1)[7] with drift : 3407.612  
## ARIMA(2,0,3)(0,1,1)[7] with drift : 3393.481  
## ARIMA(1,0,1)(0,1,1)[7] with drift : 3408.617  
## ARIMA(1,0,3)(0,1,1)[7] with drift : 3403.391  
## ARIMA(3,0,1)(0,1,1)[7] with drift : 3405.812  
## ARIMA(3,0,3)(0,1,1)[7] with drift : 3396.444  
## ARIMA(2,0,2)(0,1,1)[7] : 3389.78  
## ARIMA(2,0,2)(0,1,0)[7] : 3495.53  
## ARIMA(2,0,2)(1,1,1)[7] : 3397.608  
## ARIMA(2,0,2)(0,1,2)[7] : 3391.639  
## ARIMA(2,0,2)(1,1,0)[7] : 3430.697  
## ARIMA(2,0,2)(1,1,2)[7] : 3393.486  
## ARIMA(1,0,2)(0,1,1)[7] : 3405.597  
## ARIMA(2,0,1)(0,1,1)[7] : 3404.743  
## ARIMA(3,0,2)(0,1,1)[7] : 3405.873  
## ARIMA(2,0,3)(0,1,1)[7] : 3391.677  
## ARIMA(1,0,1)(0,1,1)[7] : 3406.631  
## ARIMA(1,0,3)(0,1,1)[7] : 3401.421  
## ARIMA(3,0,1)(0,1,1)[7] : 3404.601  
## ARIMA(3,0,3)(0,1,1)[7] : 3395.828  
##   
## Now re-fitting the best model(s) without approximations...  
##   
## ARIMA(2,0,2)(0,1,1)[7] : 3433.13  
##   
## Best model: ARIMA(2,0,2)(0,1,1)[7]

auto.fit.ATM2

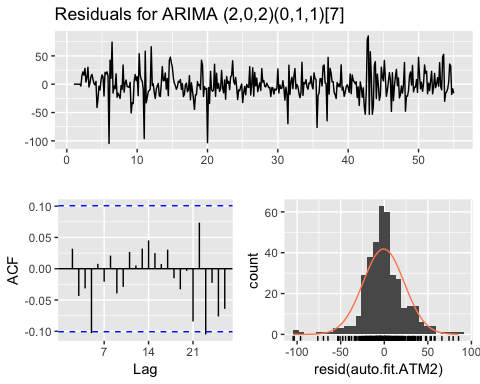
## Series: ATM2   
## ARIMA(2,0,2)(0,1,1)[7]   
##   
## Coefficients:  
## ar1 ar2 ma1 ma2 sma1  
## -0.4345 -0.9117 0.4899 0.7942 -0.7629  
## s.e. 0.0521 0.0397 0.0802 0.0564 0.0396  
##   
## sigma^2 estimated as 577.3: log likelihood=-1710.45  
## AIC=3432.9 AICc=3433.13 BIC=3456.41

Suggested best model is ARIMA(2,0,2)(0,1,1)[7].

Although auto.arima() has not applied differencing, a model with 1 time differencing should also be tested as there is not clear trend and unit root test indicates that data is not perfectly stationary.

Checking residuals of the suggested model by auto.arima() to make sure that there is no more information is left for extraction.

ggtsdisplay(resid(auto.fit.ATM2), points = FALSE, plot.type = "histogram", main = "Residuals for ARIMA (2,0,2)(0,1,1)[7]")



The residuals look normally distributed and random, that means that there is no useful information is hidden in residuals to be extracted by ARIMA models.

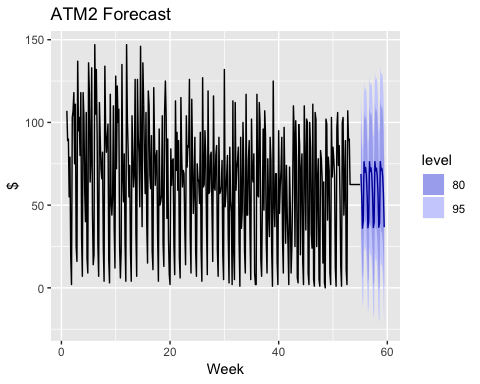
Box.test(resid(auto.fit.ATM2), lag = 14)

##   
## Box-Pierce test  
##   
## data: resid(auto.fit.ATM2)  
## X-squared = 8.2127, df = 14, p-value = 0.878

p-value is more than 0.05, suggesting that the residuals most likely is white noise.

Making forecast for 2 weeks (2\*7)

ATM2\_forecast <- forecast(auto.fit.ATM2, h = 31, level = c(80, 95))  
autoplot(ATM2\_forecast) + labs(title = "ATM2 Forecast", x = "Week", y = "$")



### Forecasting ATM3

There are only three observations at ATM3, and only these observations are used for the forecast.

As suggested above mean, median or random walk can be used for ATM3 forecasting.

# only 3 last non-zero vlues of ATM3 is considered  
ATM3<-c(96,82,85)  
mean(ATM3)

## [1] 87.66667

median(ATM3)

## [1] 85

mean - 87

meadin - 85

random walk - 85

All results look quite similar.

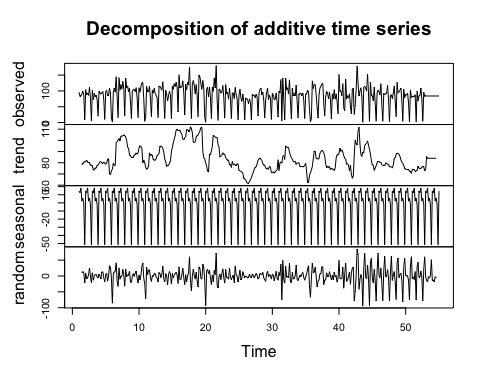
### Forecasting ATM4

The main issue with ATM4 data set is the outlier. We are going to replace it with median and then handle missing values with mean.

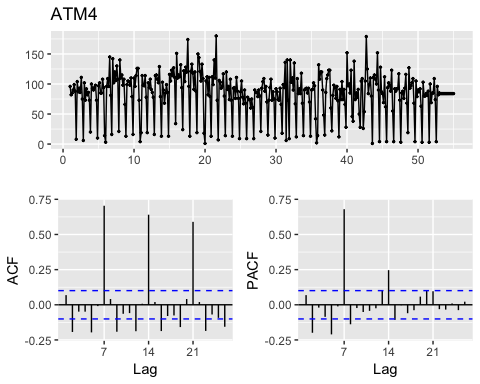
# replacing outlier with median  
ATM4[which.max(ATM4)] <- median(ATM4, na.rm = TRUE)  
# replacing missing values with mean  
ATM4[which(is.na(ATM4))] <- mean(ATM4, na.rm = TRUE)

Checking assumption before applying ARIMA model.

# checking components of ts  
ATM4.components.ts = decompose(ATM4)  
plot(ATM4.components.ts)



# checking stationarity of data  
ggtsdisplay(ATM4)



# checking with unit root test  
ATM4 %>% ur.kpss() %>% summary()

##   
## #######################   
## # KPSS Unit Root Test #   
## #######################   
##   
## Test is of type: mu with 5 lags.   
##   
## Value of test-statistic is: 0.4968   
##   
## Critical value for a significance level of:   
## 10pct 5pct 2.5pct 1pct  
## critical values 0.347 0.463 0.574 0.739

We can not reject the null hypothesis as the value of the test statistic is more extreme than the 10%, 5% and 1% critical values. Data is stationary.

# checking the number of first differences required to make the time series non-seasonal  
ATM4 %>% ndiffs()

## [1] 0

ndiffs() indicates tha differencig is not required.

auto.fit.ATM4<-auto.arima(ATM4,trace=TRUE, seasonal = TRUE)

##   
## Fitting models using approximations to speed things up...  
##   
## ARIMA(2,0,2)(1,1,1)[7] with drift : 3379.099  
## ARIMA(0,0,0)(0,1,0)[7] with drift : 3479.071  
## ARIMA(1,0,0)(1,1,0)[7] with drift : 3413.418  
## ARIMA(0,0,1)(0,1,1)[7] with drift : 3368.45  
## ARIMA(0,0,0)(0,1,0)[7] : 3477.055  
## ARIMA(0,0,1)(0,1,0)[7] with drift : 3467.631  
## ARIMA(0,0,1)(1,1,1)[7] with drift : 3370.098  
## ARIMA(0,0,1)(0,1,2)[7] with drift : 3365.198  
## ARIMA(0,0,1)(1,1,2)[7] with drift : 3371.874  
## ARIMA(0,0,0)(0,1,2)[7] with drift : 3374.742  
## ARIMA(1,0,1)(0,1,2)[7] with drift : 3367.567  
## ARIMA(0,0,2)(0,1,2)[7] with drift : 3366.409  
## ARIMA(1,0,0)(0,1,2)[7] with drift : 3367.68  
## ARIMA(1,0,2)(0,1,2)[7] with drift : 3369.637  
## ARIMA(0,0,1)(0,1,2)[7] : 3363.144  
## ARIMA(0,0,1)(0,1,1)[7] : 3366.41  
## ARIMA(0,0,1)(1,1,2)[7] : 3369.828  
## ARIMA(0,0,1)(1,1,1)[7] : 3368.054  
## ARIMA(0,0,0)(0,1,2)[7] : 3372.7  
## ARIMA(1,0,1)(0,1,2)[7] : 3365.501  
## ARIMA(0,0,2)(0,1,2)[7] : 3364.343  
## ARIMA(1,0,0)(0,1,2)[7] : 3365.625  
## ARIMA(1,0,2)(0,1,2)[7] : 3367.559  
##   
## Now re-fitting the best model(s) without approximations...  
##   
## ARIMA(0,0,1)(0,1,2)[7] : 3413.163  
##   
## Best model: ARIMA(0,0,1)(0,1,2)[7]

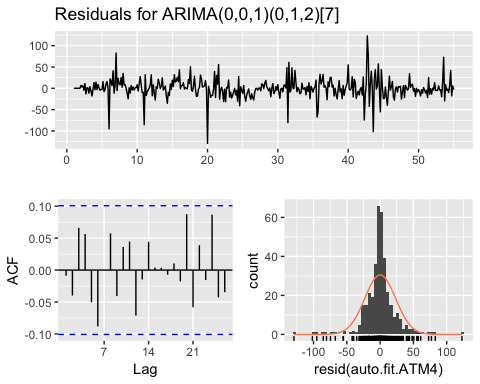
auto.fit.ATM4

## Series: ATM4   
## ARIMA(0,0,1)(0,1,2)[7]   
##   
## Coefficients:  
## ma1 sma1 sma2  
## 0.1871 -0.5802 -0.1148  
## s.e. 0.0541 0.0499 0.0509  
##   
## sigma^2 estimated as 551.4: log likelihood=-1702.53  
## AIC=3413.05 AICc=3413.16 BIC=3428.73

Suggested best model is ARIMA(0,0,1)(0,1,2)[7]

Checking residuals of the suggested model by auto.arima() to make sure that there is no more information is left for extraction.

ggtsdisplay(resid(auto.fit.ATM4), points = FALSE, plot.type = "histogram", main = "Residuals for ARIMA(0,0,1)(0,1,2)[7]")



The residuals look normally distributed and random, that means that there is no useful information is hidden in residuals to be extracted by ARIMA models.

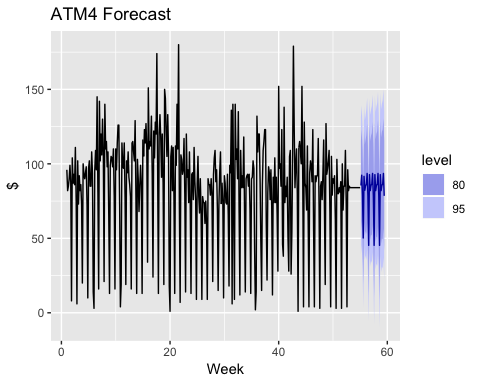
Box.test(resid(auto.fit.ATM4), lag = 14)

##   
## Box-Pierce test  
##   
## data: resid(auto.fit.ATM4)  
## X-squared = 13.255, df = 14, p-value = 0.5065

p-value is more than 0.05, suggesting that the residuals may be white noise.

Making forecast for 2 weeks (2\*7)

ATM4\_forecast <- forecast(auto.fit.ATM4, h = 31, level = c(80, 95))  
autoplot(ATM4\_forecast) + labs(title = "ATM4 Forecast", x = "Week", y = "$")



Saving forecast to excel:

# write.csv(ATM1\_forecast, file = "ATM1\_pred.csv")  
# write.csv(ATM2\_forecast, file = "ATM2\_pred.csv")  
# write.csv(ATM4\_forecast, file = "ATM4\_pred.csv")

# PART B: Power Consumption Forecast

Part B consists of a simple dataset of residential power usage for January 1998 until December 2013. The task is to make a monthly forecast of power consumption for 2014. The data is given in a single file. The variable ‘KWH’ is power consumption in Kilowatt hours.

library(urca)  
library(dplyr)  
library(tidyr)  
library(plotly)  
library(readxl)  
library(forecast)  
library(fpp2)  
library(tidyverse)  
library(lubridate)

Reading data

resident\_data = read\_excel("project1/ResidentialCustomerForecastLoad-624.xlsx")  
summary(resident\_data)

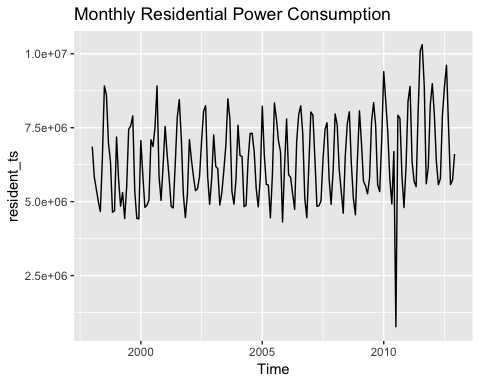
## CaseSequence YYYY-MMM KWH   
## Min. :733.0 Length:192 Min. : 770523   
## 1st Qu.:780.8 Class :character 1st Qu.: 5429912   
## Median :828.5 Mode :character Median : 6283324   
## Mean :828.5 Mean : 6502475   
## 3rd Qu.:876.2 3rd Qu.: 7620524   
## Max. :924.0 Max. :10655730   
## NA's :1

KWH column has 1 missing value, which can be replaced with median (as data set has an outlier).

# replcing 1 missing value with median  
resident\_data$KWH[is.na(resident\_data$KWH)]<-median(resident\_data$KWH,na.rm = TRUE)

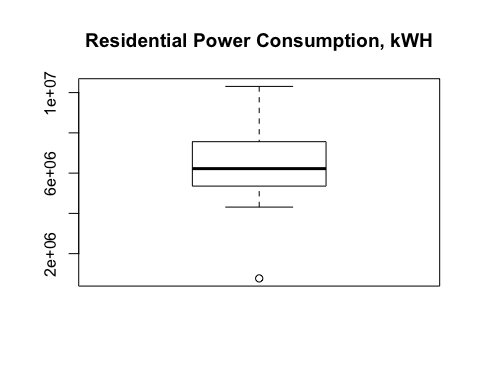
Creating ts object and plotting it.

# making ts object with frequency = 12 as we work with monthly data  
resident\_ts <- ts(resident\_data[, "KWH"], start =c(1998,1), end=c(2012,12), frequency = 12)  
autoplot(resident\_ts) + labs(title = "Monthly Residential Power Consumption")



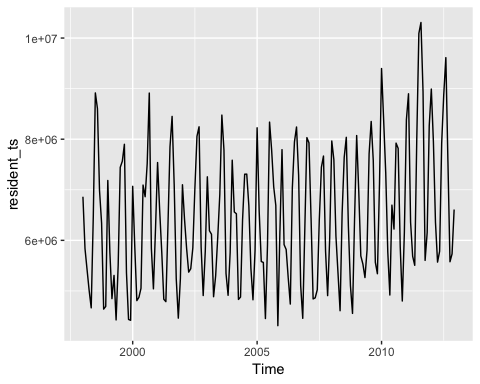
It seems that data has an outlier, checking it with the boxplot.

boxplot(resident\_ts, main = "Residential Power Consumption, kWH")



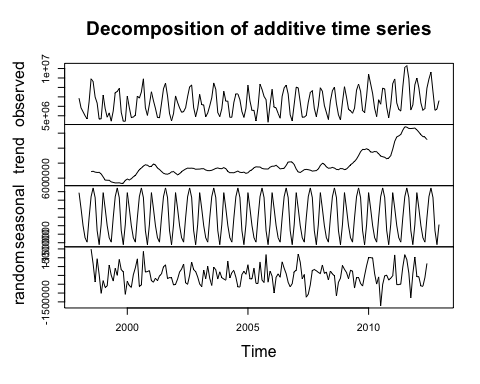
The outlier (July 2010) looks like a mistake or typo and seems unusually low. Outlier will be replaced with the median. Although in real life it is better to investigate this issue to get the correct value.

# replacing the outlier with the meadin  
resident\_ts[which.min(resident\_ts)] <- median(resident\_ts, na.rm = TRUE)  
autoplot(resident\_ts)

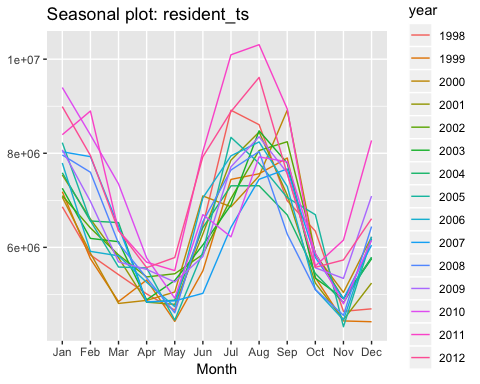


Checking assumptions

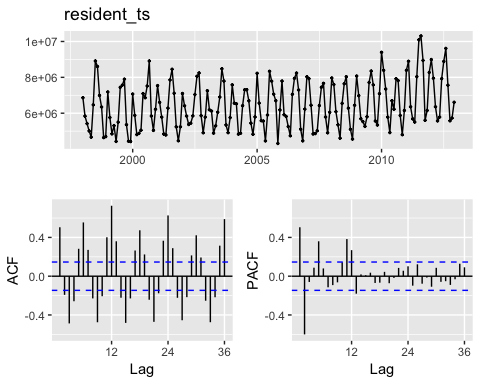
# checking components of ts  
components.ts = decompose(resident\_ts)  
plot(components.ts)



# checking seasonality  
ggseasonplot(resident\_ts)



# checking stationarity of data  
ggtsdisplay(resident\_ts)



Data has trend and seasonality with peaks in summer and winter months due to high usage of conditioners and heating.

# checking with unit root test  
resident\_ts %>% ur.kpss() %>% summary()

##   
## #######################   
## # KPSS Unit Root Test #   
## #######################   
##   
## Test is of type: mu with 4 lags.   
##   
## Value of test-statistic is: 1.0777   
##   
## Critical value for a significance level of:   
## 10pct 5pct 2.5pct 1pct  
## critical values 0.347 0.463 0.574 0.739

# checking the number of first differences required   
resident\_ts %>% ndiffs()

## [1] 1

Data is not stationary, tt is suggested that 1 difference is required.

Building ARIMA models and selecting the best one.

resident\_auto<-auto.arima(resident\_ts,trace=TRUE, seasonal = TRUE)

##   
## Fitting models using approximations to speed things up...  
##   
## ARIMA(2,0,2)(1,1,1)[12] with drift : 4642.773  
## ARIMA(0,0,0)(0,1,0)[12] with drift : 4738.1  
## ARIMA(1,0,0)(1,1,0)[12] with drift : 4668.517  
## ARIMA(0,0,1)(0,1,1)[12] with drift : 4651.18  
## ARIMA(0,0,0)(0,1,0)[12] : 4737.272  
## ARIMA(2,0,2)(0,1,1)[12] with drift : 4654.441  
## ARIMA(2,0,2)(1,1,0)[12] with drift : 4665.271  
## ARIMA(2,0,2)(2,1,1)[12] with drift : 4631.313  
## ARIMA(2,0,2)(2,1,0)[12] with drift : 4629.608  
## ARIMA(1,0,2)(2,1,0)[12] with drift : 4629.795  
## ARIMA(2,0,1)(2,1,0)[12] with drift : 4631.859  
## ARIMA(3,0,2)(2,1,0)[12] with drift : 4623.863  
## ARIMA(3,0,2)(1,1,0)[12] with drift : 4660.481  
## ARIMA(3,0,2)(2,1,1)[12] with drift : 4625.698  
## ARIMA(3,0,2)(1,1,1)[12] with drift : 4642.479  
## ARIMA(3,0,1)(2,1,0)[12] with drift : 4627.336  
## ARIMA(4,0,2)(2,1,0)[12] with drift : 4626.769  
## ARIMA(3,0,3)(2,1,0)[12] with drift : 4625.296  
## ARIMA(2,0,3)(2,1,0)[12] with drift : 4627.503  
## ARIMA(4,0,1)(2,1,0)[12] with drift : 4628.248  
## ARIMA(4,0,3)(2,1,0)[12] with drift : 4629.569  
## ARIMA(3,0,2)(2,1,0)[12] : 4626.938  
##   
## Now re-fitting the best model(s) without approximations...  
##   
## ARIMA(3,0,2)(2,1,0)[12] with drift : 4967.75  
##   
## Best model: ARIMA(3,0,2)(2,1,0)[12] with drift

resident\_auto

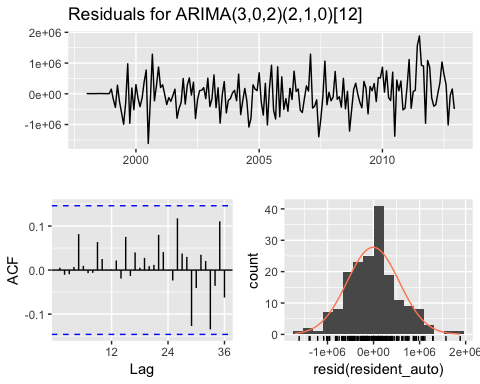
## Series: resident\_ts   
## ARIMA(3,0,2)(2,1,0)[12] with drift   
##   
## Coefficients:  
## ar1 ar2 ar3 ma1 ma2 sar1 sar2 drift  
## -0.5194 -0.2156 0.3829 0.9070 0.5112 -0.7963 -0.4722 7983.078  
## s.e. 0.2044 0.1814 0.0890 0.2135 0.2309 0.0763 0.0846 3121.675  
##   
## sigma^2 estimated as 3.581e+11: log likelihood=-2474.31  
## AIC=4966.61 AICc=4967.75 BIC=4994.73

ARIMA(3,0,2)(2,1,0)[12] was suggested as the best model.

Although auto.arima() has not applied differencing, a model with 1 time differencing should also be tested as there is not clear trend and ndiffs() suggests one.

Checking residuals of the suggested model by auto.arima() to make sure that there is no more information is left for extraction.

ggtsdisplay(resid(resident\_auto), points = FALSE, plot.type = "histogram", main = "Residuals for ARIMA(3,0,2)(2,1,0)[12]")



Examining the null hypothesis of residuals independence.

Box.test(resid(resident\_auto), lag = 24)

##   
## Box-Pierce test  
##   
## data: resid(resident\_auto)  
## X-squared = 5.2552, df = 24, p-value = 1

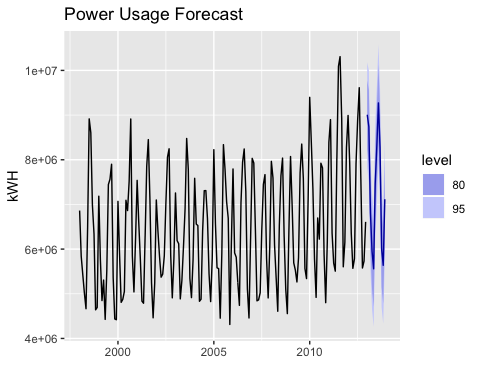
p-value is more than 0.05, suggesting that the residuals may be white noise.

Making predictions for 2013

resident\_forecast <- forecast(resident\_auto, h = 1\*12, level = c(80, 95))  
resident\_forecast

## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## Jan 2013 9007596 8240746 9774446 7834801 10180392  
## Feb 2013 8740635 7918187 9563084 7482809 9998462  
## Mar 2013 6812241 5986619 7637863 5549561 8074921  
## Apr 2013 5952630 5104985 6800275 4656269 7248991  
## May 2013 5558582 4710936 6406228 4262219 6854945  
## Jun 2013 7475594 6627849 8323338 6179080 8772107  
## Jul 2013 8301542 7449981 9153103 6999191 9603892  
## Aug 2013 9272166 8419684 10124648 7968408 10575924  
## Sep 2013 8279407 7426923 9131890 6975646 9583168  
## Oct 2013 6001133 5147699 6854567 4695918 7306348  
## Nov 2013 5637290 4783108 6491471 4330932 6943647  
## Dec 2013 7116527 6262297 7970758 5810094 8422960

autoplot(resident\_forecast) + labs(title = "Power Usage Forecast", x = " ", y = "kWH")



# PART C: Water Flow Forecast

Part C consists of two data sets. These are simple 2 columns sets, however they have different time stamps. Your optional assignment is to time-base sequence the data and aggregate based on hour (example of what this looks like, follows). Note for multiple recordings within an hour, take the mean. Then to test appropriate assumptions and forecast a week forward with confidence bands (80 and 95%).

The following steps will be taken:

1. Data preparation and exploration
2. Checking assumptions
3. Building model
4. Checking Residuals
5. Assessing accuracy
6. Making Predictions

### Data Preparation

Waterflow1 and Waterflow2 files have similar structure, but different time stamps: pipeline 1 has readings in the middle of hours while pipeline 2 has reading at the end of every hour. In order to join two files together we need to convert readings of WaterFlow\_Pipe1 to hourly and then join 2 data sets. Also data for WaterFlow\_Pipe1 is available till 2015-11-01 23:35, whereas data for WaterFlow\_Pipe2 is available till 2015-12-03 16:00.

Please note: as no instructions given we assume that WaterFlow\_Pipe1 measures from 2015-11-01 23:35 till 2015-12-03 16:00 are missing values. Which are going to be replaced with mean.

library(urca)  
library(fpp2)  
library(datetime)  
library(readxl)  
library(lubridate)  
library(dplyr)  
library(tidyr)  
library(forecast)

# reading files  
water1 = read\_excel("project1/Waterflow\_Pipe1.xlsx",col\_types =c("date", "numeric"))  
water2 = read\_excel("project1/Waterflow\_Pipe2.xlsx",col\_types =c("date", "numeric"))  
# renaming column names  
colnames(water1)= c("DateTime","WaterFlow")  
colnames(water2)= c("DateTime","WaterFlow")

## transforming WaterFlow\_Pipe1  
  
water1 <- water1 %>%  
 # splitting DateTime column to date and hour columns  
 mutate(Date = date(DateTime),  
 # convert minutes to hours  
 Hour = hour(DateTime)+1) %>%   
 # getting back to DateTime column  
 mutate(DateTime = ymd\_h(paste(Date, Hour))) %>%   
 # grouping by DateTime  
 group\_by(DateTime) %>%  
 summarize(WaterFlow = mean(WaterFlow)) %>%   
 select(DateTime,WaterFlow)  
  
summary(water1)

## DateTime WaterFlow   
## Min. :2015-10-23 01:00:00 Min. : 8.923   
## 1st Qu.:2015-10-25 11:45:00 1st Qu.:17.033   
## Median :2015-10-27 23:30:00 Median :19.784   
## Mean :2015-10-27 23:38:53 Mean :19.893   
## 3rd Qu.:2015-10-30 11:15:00 3rd Qu.:22.789   
## Max. :2015-11-02 00:00:00 Max. :31.730

## transforming WaterFlow\_Pipe2  
water2 <- water2 %>%  
 # grouping by DateTime  
 group\_by(DateTime) %>%   
 summarize(WaterFlow = mean(WaterFlow))  
summary(water2)

## DateTime WaterFlow   
## Min. :2015-10-23 01:00:00 Min. : 1.885   
## 1st Qu.:2015-11-02 10:45:00 1st Qu.:28.140   
## Median :2015-11-12 20:30:00 Median :39.682   
## Mean :2015-11-12 20:30:00 Mean :39.556   
## 3rd Qu.:2015-11-23 06:15:00 3rd Qu.:50.782   
## Max. :2015-12-03 16:00:00 Max. :78.303

# joining water1 and water2 data sets  
water\_df <- full\_join(water1, water2, by = "DateTime", suffix = c("\_1", "\_2")) %>%   
 select (DateTime, WaterFlow\_1, WaterFlow\_2) %>%   
 # calculating total waterflow  
 mutate (Total = WaterFlow\_1 + WaterFlow\_2) %>%   
 select(DateTime,WaterFlow\_1, WaterFlow\_2, Total)  
  
summary(water\_df)

## DateTime WaterFlow\_1 WaterFlow\_2   
## Min. :2015-10-23 01:00:00 Min. : 8.923 Min. : 1.885   
## 1st Qu.:2015-11-02 10:45:00 1st Qu.:17.033 1st Qu.:28.140   
## Median :2015-11-12 20:30:00 Median :19.784 Median :39.682   
## Mean :2015-11-12 20:30:00 Mean :19.893 Mean :39.556   
## 3rd Qu.:2015-11-23 06:15:00 3rd Qu.:22.790 3rd Qu.:50.782   
## Max. :2015-12-03 16:00:00 Max. :31.730 Max. :78.303   
## NA's :764   
## Total   
## Min. : 18.01   
## 1st Qu.: 47.23   
## Median : 60.74   
## Mean : 59.50   
## 3rd Qu.: 70.67   
## Max. :106.50   
## NA's :764

Replacing missing values with mean for Pipe 1.

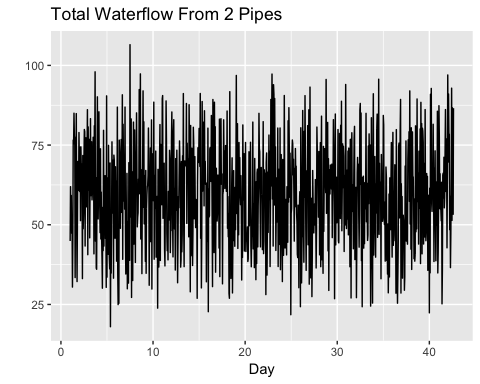
# replcing missing values with mean for pipe1  
water\_df$WaterFlow\_1[which(is.na(water\_df$WaterFlow\_1))] <- mean(water\_df$WaterFlow\_1, na.rm = TRUE)  
# updating Total flow  
water\_df <- water\_df %>%   
 mutate (Total = WaterFlow\_1 + WaterFlow\_2)   
summary(water\_df)

## DateTime WaterFlow\_1 WaterFlow\_2   
## Min. :2015-10-23 01:00:00 Min. : 8.923 Min. : 1.885   
## 1st Qu.:2015-11-02 10:45:00 1st Qu.:19.893 1st Qu.:28.140   
## Median :2015-11-12 20:30:00 Median :19.893 Median :39.682   
## Mean :2015-11-12 20:30:00 Mean :19.893 Mean :39.556   
## 3rd Qu.:2015-11-23 06:15:00 3rd Qu.:19.893 3rd Qu.:50.782   
## Max. :2015-12-03 16:00:00 Max. :31.730 Max. :78.303   
## Total   
## Min. : 18.01   
## 1st Qu.: 47.59   
## Median : 59.71   
## Mean : 59.45   
## 3rd Qu.: 70.73   
## Max. :106.50

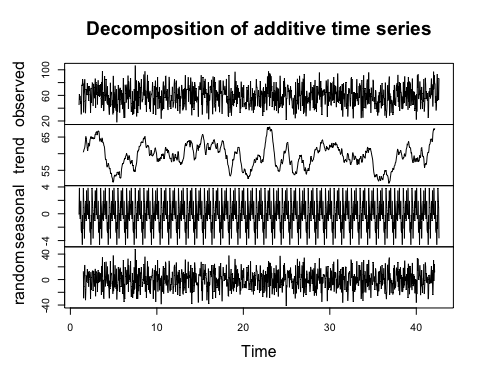
Now data does not have missing values.

### Checking assumptions

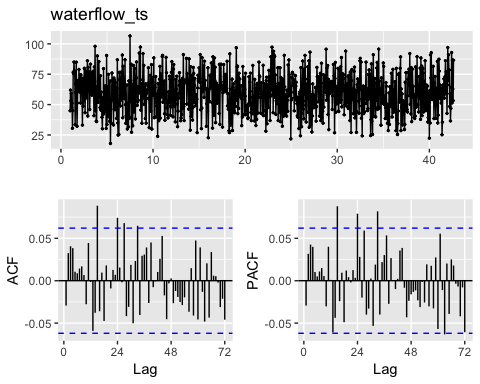
# we'll use frequency = 24 h as we want to get the results in the unit of hours  
waterflow\_ts <- ts(water\_df$Total, frequency = 24)  
autoplot(waterflow\_ts) + labs (title = "Total Waterflow From 2 Pipes", x = "Day", y = " ")



# checking components of ts  
components.ts = decompose(waterflow\_ts)  
plot(components.ts)



# checking stationarity of data  
ggtsdisplay(waterflow\_ts)



The data is stationary.

# checking with unit root test  
waterflow\_ts %>% ur.kpss() %>% summary()

##   
## #######################   
## # KPSS Unit Root Test #   
## #######################   
##   
## Test is of type: mu with 7 lags.   
##   
## Value of test-statistic is: 0.0995   
##   
## Critical value for a significance level of:   
## 10pct 5pct 2.5pct 1pct  
## critical values 0.347 0.463 0.574 0.739

The results of unit root test confirm that data is stationary.

We will try to use auto.arima function in forecast package which helps us identify the best fit ARIMA model on the fly.

auto.fit<-auto.arima(waterflow\_ts,trace=TRUE, seasonal = TRUE)

##   
## Fitting models using approximations to speed things up...  
##   
## ARIMA(2,0,2)(1,0,1)[24] with non-zero mean : 8406.99  
## ARIMA(0,0,0) with non-zero mean : 8407.396  
## ARIMA(1,0,0)(1,0,0)[24] with non-zero mean : 8405.063  
## ARIMA(0,0,1)(0,0,1)[24] with non-zero mean : 8404.82  
## ARIMA(0,0,0) with zero mean : 11081.41  
## ARIMA(0,0,1) with non-zero mean : 8408.602  
## ARIMA(0,0,1)(1,0,1)[24] with non-zero mean : 8405.028  
## ARIMA(0,0,1)(0,0,2)[24] with non-zero mean : 8406.755  
## ARIMA(0,0,1)(1,0,0)[24] with non-zero mean : 8404.497  
## ARIMA(0,0,1)(2,0,0)[24] with non-zero mean : 8410.154  
## ARIMA(0,0,1)(2,0,1)[24] with non-zero mean : 8412.166  
## ARIMA(0,0,0)(1,0,0)[24] with non-zero mean : 8404.009  
## ARIMA(0,0,0)(2,0,0)[24] with non-zero mean : 8409.584  
## ARIMA(0,0,0)(1,0,1)[24] with non-zero mean : 8404.651  
## ARIMA(0,0,0)(0,0,1)[24] with non-zero mean : 8403.74  
## ARIMA(0,0,0)(0,0,2)[24] with non-zero mean : 8405.663  
## ARIMA(0,0,0)(1,0,2)[24] with non-zero mean : 8405.992  
## ARIMA(1,0,0)(0,0,1)[24] with non-zero mean : 8404.861  
## ARIMA(1,0,1)(0,0,1)[24] with non-zero mean : 8406.426  
## ARIMA(0,0,0)(0,0,1)[24] with zero mean : 10298.19  
##   
## Now re-fitting the best model(s) without approximations...  
##   
## ARIMA(0,0,0)(0,0,1)[24] with non-zero mean : 8403.737  
##   
## Best model: ARIMA(0,0,0)(0,0,1)[24] with non-zero mean

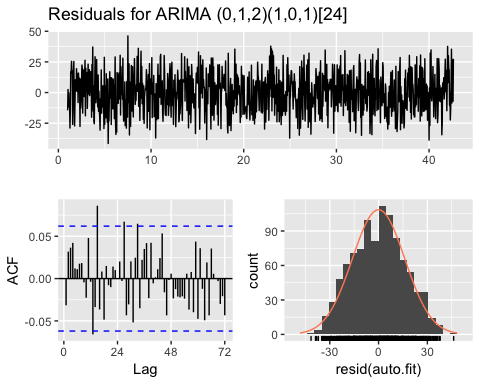
auto.fit

## Series: waterflow\_ts   
## ARIMA(0,0,0)(0,0,1)[24] with non-zero mean   
##   
## Coefficients:  
## sma1 mean  
## 0.0760 59.4647  
## s.e. 0.0317 0.5475  
##   
## sigma^2 estimated as 260.3: log likelihood=-4198.86  
## AIC=8403.71 AICc=8403.74 BIC=8418.44

auto.arima() has built different models with different p, q values; with and without seasonal term; with and without drift. For all models 1 time differencing was applied.

auto.arima() suggests that ARIMA(0,0,0)(0,0,1)[24] is the best models. It indicates that data is white noise and the errors are uncorrelated across time.

ggtsdisplay(resid(auto.fit), points = FALSE, plot.type = "histogram", main = "Residuals for ARIMA (0,1,2)(1,0,1)[24]")



The residuals look normally distributed and random, that means that there is no useful information is hidden in residuals to be extracted by ARIMA models.

Box.test(resid(auto.fit), lag = 24)

##   
## Box-Pierce test  
##   
## data: resid(auto.fit)  
## X-squared = 26.227, df = 24, p-value = 0.3418

p-value is more than 0.05, suggesting that the residuals may be white noise.

Performing forecast for a one week (24\*7)

water\_forecast <- forecast(auto.fit, h = 24\*7, level = c(80, 95))  
water\_forecast

## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 42.66667 59.45137 38.77646 80.12628 27.83182 91.07091  
## 42.70833 60.62281 39.94790 81.29771 29.00326 92.24235  
## 42.75000 61.53377 40.85886 82.20868 29.91422 93.15331  
## 42.79167 59.51824 38.84333 80.19315 27.89869 91.13778  
## 42.83333 60.84134 40.16643 81.51625 29.22180 92.46088  
## 42.87500 60.04288 39.36797 80.71779 28.42333 91.66242  
## 42.91667 58.33704 37.66214 79.01195 26.71750 89.95659  
## 42.95833 60.07592 39.40101 80.75083 28.45637 91.69546  
## 43.00000 62.32474 41.64983 82.99965 30.70519 93.94428  
## 43.04167 60.08099 39.40608 80.75590 28.46144 91.70053  
## 43.08333 61.74872 41.07381 82.42363 30.12918 93.36827  
## 43.12500 60.93955 40.26464 81.61446 29.32001 92.55910  
## 43.16667 60.87354 40.19863 81.54845 29.25400 92.49309  
## 43.20833 58.66841 37.99350 79.34332 27.04886 90.28795  
## 43.25000 59.45142 38.77651 80.12632 27.83187 91.07096  
## 43.29167 57.76703 37.09212 78.44193 26.14748 89.38657  
## 43.33333 58.48660 37.81169 79.16151 26.86706 90.10615  
## 43.37500 60.01932 39.34441 80.69423 28.39977 91.63886  
## 43.41667 62.15205 41.47714 82.82696 30.53250 93.77159  
## 43.45833 58.81566 38.14075 79.49057 27.19612 90.43521  
## 43.50000 61.48880 40.81390 82.16371 29.86926 93.10835  
## 43.54167 59.72954 39.05464 80.40445 28.11000 91.34909  
## 43.58333 58.92523 38.25032 79.60014 27.30569 90.54478  
## 43.62500 61.55661 40.88170 82.23152 29.93706 93.17615  
## 43.66667 59.46466 38.73006 80.19925 27.75383 91.17549  
## 43.70833 59.46466 38.73006 80.19925 27.75383 91.17549  
## 43.75000 59.46466 38.73006 80.19925 27.75383 91.17549  
## 43.79167 59.46466 38.73006 80.19925 27.75383 91.17549  
## 43.83333 59.46466 38.73006 80.19925 27.75383 91.17549  
## 43.87500 59.46466 38.73006 80.19925 27.75383 91.17549  
## 43.91667 59.46466 38.73006 80.19925 27.75383 91.17549  
## 43.95833 59.46466 38.73006 80.19925 27.75383 91.17549  
## 44.00000 59.46466 38.73006 80.19925 27.75383 91.17549  
## 44.04167 59.46466 38.73006 80.19925 27.75383 91.17549  
## 44.08333 59.46466 38.73006 80.19925 27.75383 91.17549  
## 44.12500 59.46466 38.73006 80.19925 27.75383 91.17549  
## 44.16667 59.46466 38.73006 80.19925 27.75383 91.17549  
## 44.20833 59.46466 38.73006 80.19925 27.75383 91.17549  
## 44.25000 59.46466 38.73006 80.19925 27.75383 91.17549  
## 44.29167 59.46466 38.73006 80.19925 27.75383 91.17549  
## 44.33333 59.46466 38.73006 80.19925 27.75383 91.17549  
## 44.37500 59.46466 38.73006 80.19925 27.75383 91.17549  
## 44.41667 59.46466 38.73006 80.19925 27.75383 91.17549  
## 44.45833 59.46466 38.73006 80.19925 27.75383 91.17549  
## 44.50000 59.46466 38.73006 80.19925 27.75383 91.17549  
## 44.54167 59.46466 38.73006 80.19925 27.75383 91.17549  
## 44.58333 59.46466 38.73006 80.19925 27.75383 91.17549  
## 44.62500 59.46466 38.73006 80.19925 27.75383 91.17549  
## 44.66667 59.46466 38.73006 80.19925 27.75383 91.17549  
## 44.70833 59.46466 38.73006 80.19925 27.75383 91.17549  
## 44.75000 59.46466 38.73006 80.19925 27.75383 91.17549  
## 44.79167 59.46466 38.73006 80.19925 27.75383 91.17549  
## 44.83333 59.46466 38.73006 80.19925 27.75383 91.17549  
## 44.87500 59.46466 38.73006 80.19925 27.75383 91.17549  
## 44.91667 59.46466 38.73006 80.19925 27.75383 91.17549  
## 44.95833 59.46466 38.73006 80.19925 27.75383 91.17549  
## 45.00000 59.46466 38.73006 80.19925 27.75383 91.17549  
## 45.04167 59.46466 38.73006 80.19925 27.75383 91.17549  
## 45.08333 59.46466 38.73006 80.19925 27.75383 91.17549  
## 45.12500 59.46466 38.73006 80.19925 27.75383 91.17549  
## 45.16667 59.46466 38.73006 80.19925 27.75383 91.17549  
## 45.20833 59.46466 38.73006 80.19925 27.75383 91.17549  
## 45.25000 59.46466 38.73006 80.19925 27.75383 91.17549  
## 45.29167 59.46466 38.73006 80.19925 27.75383 91.17549  
## 45.33333 59.46466 38.73006 80.19925 27.75383 91.17549  
## 45.37500 59.46466 38.73006 80.19925 27.75383 91.17549  
## 45.41667 59.46466 38.73006 80.19925 27.75383 91.17549  
## 45.45833 59.46466 38.73006 80.19925 27.75383 91.17549  
## 45.50000 59.46466 38.73006 80.19925 27.75383 91.17549  
## 45.54167 59.46466 38.73006 80.19925 27.75383 91.17549  
## 45.58333 59.46466 38.73006 80.19925 27.75383 91.17549  
## 45.62500 59.46466 38.73006 80.19925 27.75383 91.17549  
## 45.66667 59.46466 38.73006 80.19925 27.75383 91.17549  
## 45.70833 59.46466 38.73006 80.19925 27.75383 91.17549  
## 45.75000 59.46466 38.73006 80.19925 27.75383 91.17549  
## 45.79167 59.46466 38.73006 80.19925 27.75383 91.17549  
## 45.83333 59.46466 38.73006 80.19925 27.75383 91.17549  
## 45.87500 59.46466 38.73006 80.19925 27.75383 91.17549  
## 45.91667 59.46466 38.73006 80.19925 27.75383 91.17549  
## 45.95833 59.46466 38.73006 80.19925 27.75383 91.17549  
## 46.00000 59.46466 38.73006 80.19925 27.75383 91.17549  
## 46.04167 59.46466 38.73006 80.19925 27.75383 91.17549  
## 46.08333 59.46466 38.73006 80.19925 27.75383 91.17549  
## 46.12500 59.46466 38.73006 80.19925 27.75383 91.17549  
## 46.16667 59.46466 38.73006 80.19925 27.75383 91.17549  
## 46.20833 59.46466 38.73006 80.19925 27.75383 91.17549  
## 46.25000 59.46466 38.73006 80.19925 27.75383 91.17549  
## 46.29167 59.46466 38.73006 80.19925 27.75383 91.17549  
## 46.33333 59.46466 38.73006 80.19925 27.75383 91.17549  
## 46.37500 59.46466 38.73006 80.19925 27.75383 91.17549  
## 46.41667 59.46466 38.73006 80.19925 27.75383 91.17549  
## 46.45833 59.46466 38.73006 80.19925 27.75383 91.17549  
## 46.50000 59.46466 38.73006 80.19925 27.75383 91.17549  
## 46.54167 59.46466 38.73006 80.19925 27.75383 91.17549  
## 46.58333 59.46466 38.73006 80.19925 27.75383 91.17549  
## 46.62500 59.46466 38.73006 80.19925 27.75383 91.17549  
## 46.66667 59.46466 38.73006 80.19925 27.75383 91.17549  
## 46.70833 59.46466 38.73006 80.19925 27.75383 91.17549  
## 46.75000 59.46466 38.73006 80.19925 27.75383 91.17549  
## 46.79167 59.46466 38.73006 80.19925 27.75383 91.17549  
## 46.83333 59.46466 38.73006 80.19925 27.75383 91.17549  
## 46.87500 59.46466 38.73006 80.19925 27.75383 91.17549  
## 46.91667 59.46466 38.73006 80.19925 27.75383 91.17549  
## 46.95833 59.46466 38.73006 80.19925 27.75383 91.17549  
## 47.00000 59.46466 38.73006 80.19925 27.75383 91.17549  
## 47.04167 59.46466 38.73006 80.19925 27.75383 91.17549  
## 47.08333 59.46466 38.73006 80.19925 27.75383 91.17549  
## 47.12500 59.46466 38.73006 80.19925 27.75383 91.17549  
## 47.16667 59.46466 38.73006 80.19925 27.75383 91.17549  
## 47.20833 59.46466 38.73006 80.19925 27.75383 91.17549  
## 47.25000 59.46466 38.73006 80.19925 27.75383 91.17549  
## 47.29167 59.46466 38.73006 80.19925 27.75383 91.17549  
## 47.33333 59.46466 38.73006 80.19925 27.75383 91.17549  
## 47.37500 59.46466 38.73006 80.19925 27.75383 91.17549  
## 47.41667 59.46466 38.73006 80.19925 27.75383 91.17549  
## 47.45833 59.46466 38.73006 80.19925 27.75383 91.17549  
## 47.50000 59.46466 38.73006 80.19925 27.75383 91.17549  
## 47.54167 59.46466 38.73006 80.19925 27.75383 91.17549  
## 47.58333 59.46466 38.73006 80.19925 27.75383 91.17549  
## 47.62500 59.46466 38.73006 80.19925 27.75383 91.17549  
## 47.66667 59.46466 38.73006 80.19925 27.75383 91.17549  
## 47.70833 59.46466 38.73006 80.19925 27.75383 91.17549  
## 47.75000 59.46466 38.73006 80.19925 27.75383 91.17549  
## 47.79167 59.46466 38.73006 80.19925 27.75383 91.17549  
## 47.83333 59.46466 38.73006 80.19925 27.75383 91.17549  
## 47.87500 59.46466 38.73006 80.19925 27.75383 91.17549  
## 47.91667 59.46466 38.73006 80.19925 27.75383 91.17549  
## 47.95833 59.46466 38.73006 80.19925 27.75383 91.17549  
## 48.00000 59.46466 38.73006 80.19925 27.75383 91.17549  
## 48.04167 59.46466 38.73006 80.19925 27.75383 91.17549  
## 48.08333 59.46466 38.73006 80.19925 27.75383 91.17549  
## 48.12500 59.46466 38.73006 80.19925 27.75383 91.17549  
## 48.16667 59.46466 38.73006 80.19925 27.75383 91.17549  
## 48.20833 59.46466 38.73006 80.19925 27.75383 91.17549  
## 48.25000 59.46466 38.73006 80.19925 27.75383 91.17549  
## 48.29167 59.46466 38.73006 80.19925 27.75383 91.17549  
## 48.33333 59.46466 38.73006 80.19925 27.75383 91.17549  
## 48.37500 59.46466 38.73006 80.19925 27.75383 91.17549  
## 48.41667 59.46466 38.73006 80.19925 27.75383 91.17549  
## 48.45833 59.46466 38.73006 80.19925 27.75383 91.17549  
## 48.50000 59.46466 38.73006 80.19925 27.75383 91.17549  
## 48.54167 59.46466 38.73006 80.19925 27.75383 91.17549  
## 48.58333 59.46466 38.73006 80.19925 27.75383 91.17549  
## 48.62500 59.46466 38.73006 80.19925 27.75383 91.17549  
## 48.66667 59.46466 38.73006 80.19925 27.75383 91.17549  
## 48.70833 59.46466 38.73006 80.19925 27.75383 91.17549  
## 48.75000 59.46466 38.73006 80.19925 27.75383 91.17549  
## 48.79167 59.46466 38.73006 80.19925 27.75383 91.17549  
## 48.83333 59.46466 38.73006 80.19925 27.75383 91.17549  
## 48.87500 59.46466 38.73006 80.19925 27.75383 91.17549  
## 48.91667 59.46466 38.73006 80.19925 27.75383 91.17549  
## 48.95833 59.46466 38.73006 80.19925 27.75383 91.17549  
## 49.00000 59.46466 38.73006 80.19925 27.75383 91.17549  
## 49.04167 59.46466 38.73006 80.19925 27.75383 91.17549  
## 49.08333 59.46466 38.73006 80.19925 27.75383 91.17549  
## 49.12500 59.46466 38.73006 80.19925 27.75383 91.17549  
## 49.16667 59.46466 38.73006 80.19925 27.75383 91.17549  
## 49.20833 59.46466 38.73006 80.19925 27.75383 91.17549  
## 49.25000 59.46466 38.73006 80.19925 27.75383 91.17549  
## 49.29167 59.46466 38.73006 80.19925 27.75383 91.17549  
## 49.33333 59.46466 38.73006 80.19925 27.75383 91.17549  
## 49.37500 59.46466 38.73006 80.19925 27.75383 91.17549  
## 49.41667 59.46466 38.73006 80.19925 27.75383 91.17549  
## 49.45833 59.46466 38.73006 80.19925 27.75383 91.17549  
## 49.50000 59.46466 38.73006 80.19925 27.75383 91.17549  
## 49.54167 59.46466 38.73006 80.19925 27.75383 91.17549  
## 49.58333 59.46466 38.73006 80.19925 27.75383 91.17549  
## 49.62500 59.46466 38.73006 80.19925 27.75383 91.17549

autoplot(water\_forecast) + labs(title = "Water Flow Forecast", y = "", x = "Day")

