PROJECT 1 FALL 2019

Group 1: Angrand, Burke, Deboch, Groysman, Karr

Data 624

Table of Contents

Data 624. Project 1: Part A

Angrand, Burke, Deboch, Groysman, Karr

October 22, 2019

Part A - ATM Forecast, ATM624Data.xlsx

Data: ATM624Data.xlsx

In part A, I want you to forecast how much cash is taken out of 4 different ATM machines for May 2010. The data is given in a single file. The variable 'Cash' is provided in hundreds of dollars, other than that it is straight forward. I am being somewhat ambiguous on purpose. I am giving you data, please provide your written report on your findings, visuals, discussion and your R code all within a Word readable document, except the forecast which you will put in an Excel readable file. I must be able to cut and paste your R code and run it in R studio. Your report must be professional - most of all - readable, EASY to follow. Let me know what you are thinking, assumptions you are making! Your forecast is a simple CSV or Excel file that MATCHES the format of the data I provide.

```
#Upload library
library(tidyverse)
library(readxl)
library(fpp2)
library(forecast)
```

Read in File/EDA/Data Adjustments

- drop nulls
- restructure dataset
- convert to time series object
- plot time series object for each ATM

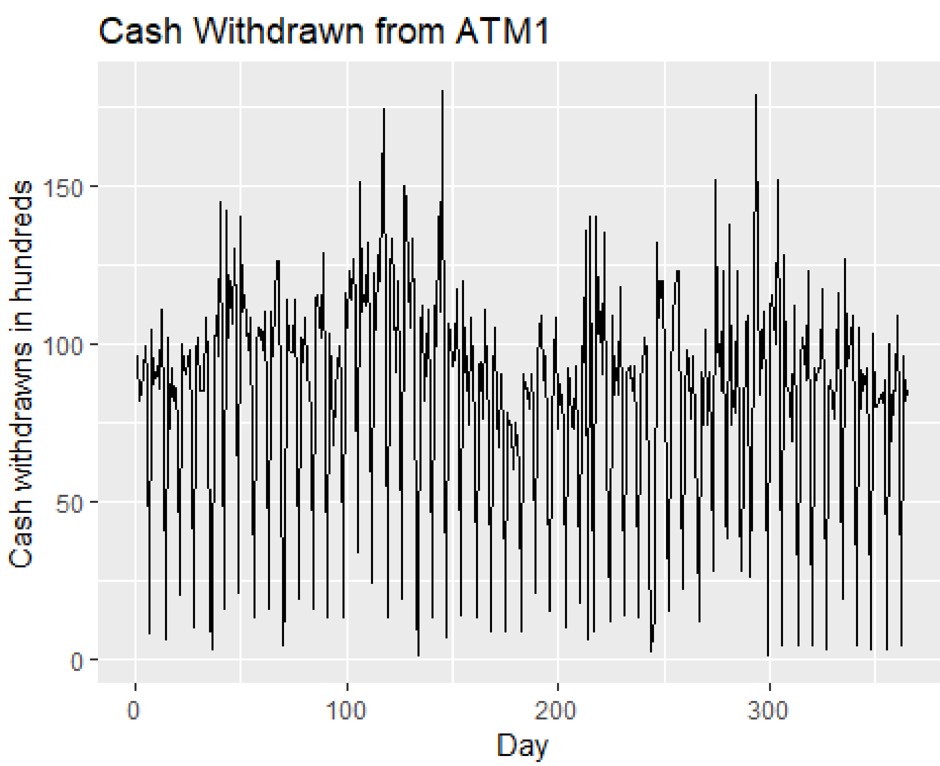
```
temp = tempfile(fileext = ".xlsx")
dataURL <- "https://raw.githubusercontent.com/mburke65/CUNY_Data624/ma
ster/ProjectFolder/Provided_Files/ATM624Data.xlsx"
download.file(dataURL, destfile=temp, mode='wb')

atm <- readxl::read_excel(temp, sheet =1)

head(atm,5)

## # A tibble: 5 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

#Drop null values
atm<-atm %>%
  drop_na()

#Convert each ATM to Column
atm<- atm %>%
  spread(ATM,Cash)
head(atm,5)

## # A tibble: 5 x 5
##   DATE                 ATM1  ATM2  ATM3  ATM4
##   <dttm>              <dbl> <dbl> <dbl> <dbl>
## 1 2009-05-01 00:00:00    96   107     0    96
## 2 2009-05-02 00:00:00    82    89     0    82
## 3 2009-05-03 00:00:00    85    90     0    85
## 4 2009-05-04 00:00:00    90    55     0    90
## 5 2009-05-05 00:00:00    99    79     0    99

#Fix the date column
atm <- atm %>%
  mutate(DATE =as.Date(DATE))
head(atm)

## # 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

#Convert to a time series
ts_atm <- ts(atm %>% select(-DATE))

head(ts_atm)

## Time Series:
## Start = 1
## End = 6
## Frequency = 1
##   ATM1 ATM2 ATM3 ATM4
## 1   96  107    0   96
```

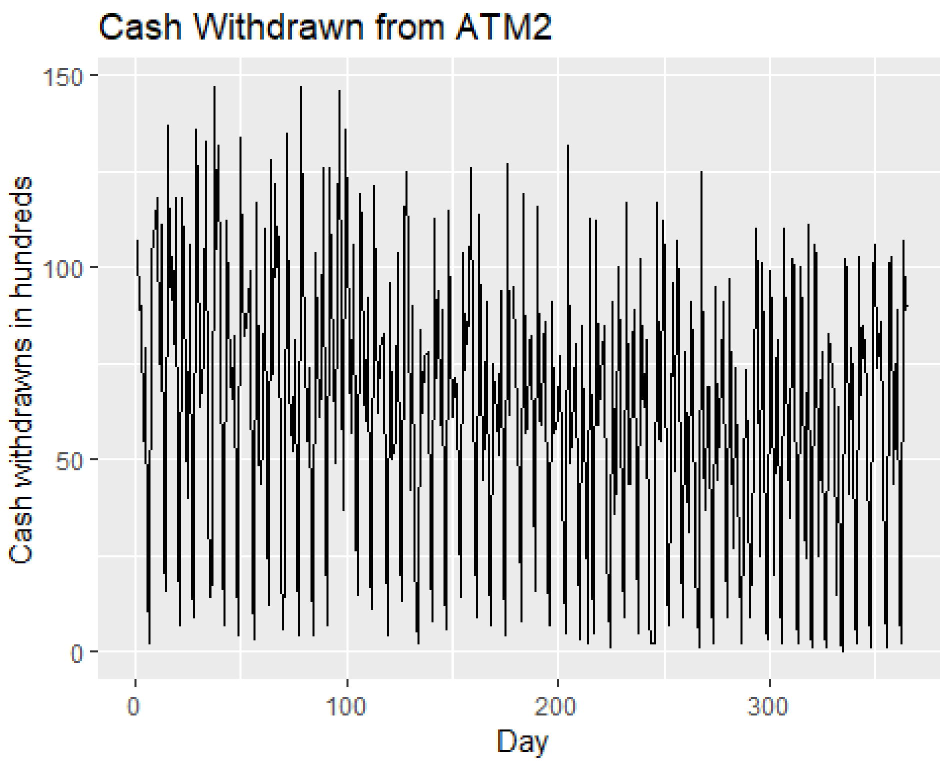
```
## 2   82   89    0   82
## 3   85   90    0   85
## 4   90   55    0   90
## 5   99   79    0   99
## 6   88   19    0   88
```

- ATM1, ATM2, and ATM4 are a big deal of variation. buT ATM3 shows no cash withdrawn for most of the year. One assumption we can do about ATM3 is that it has just opened. we will use the entire time series of ATM1 and ATM2. ATM3 will be used to forecast future prediction.

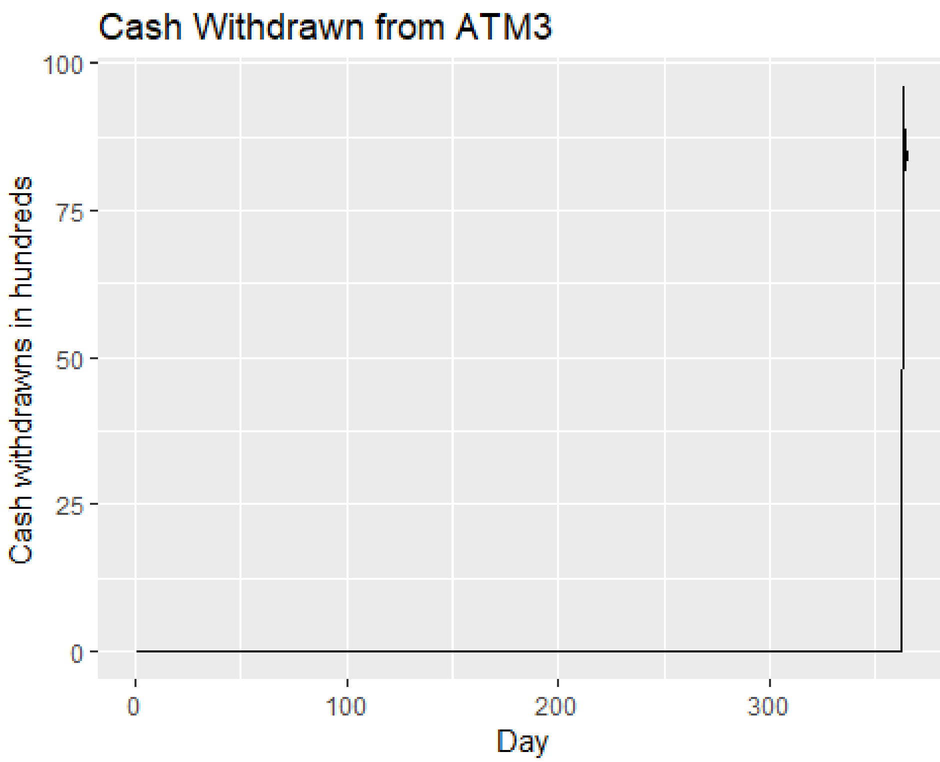
```
#Separate each ATM from the dataset and graph each dataset
atm1<-ts_atm[,"ATM1"]
autoplot(atm1) +
  labs(title ="Cash Withdrawn from ATM1", x="Day") +
  scale_y_continuous("Cash withdrawns in hundreds") +
  scale_color_discrete(NULL)
```



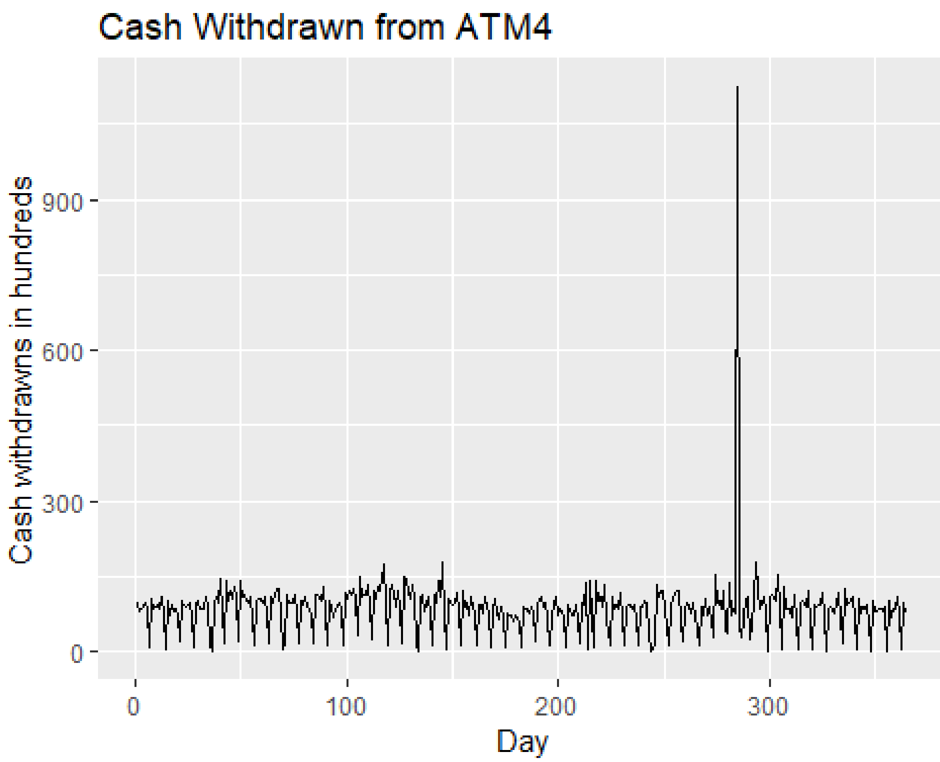

```
#Separate each ATM from the dataset and graph each dataset
atm2<-ts_atm[,"ATM2"]
autoplot(atm2) +
  labs(title ="Cash Withdrawn from ATM2", x="Day") +
  scale_y_continuous("Cash withdrawns in hundreds") +
  scale_color_discrete(NULL)
```




```
#Separate each ATM from the dataset and graph each dataset
atm3<-ts_atm[,"ATM3"]
autoplot(atm3) +
  labs(title ="Cash Withdrawn from ATM3", x="Day") +
  scale_y_continuous("Cash withdrawns in hundreds") +
  scale_color_discrete(NULL)
```

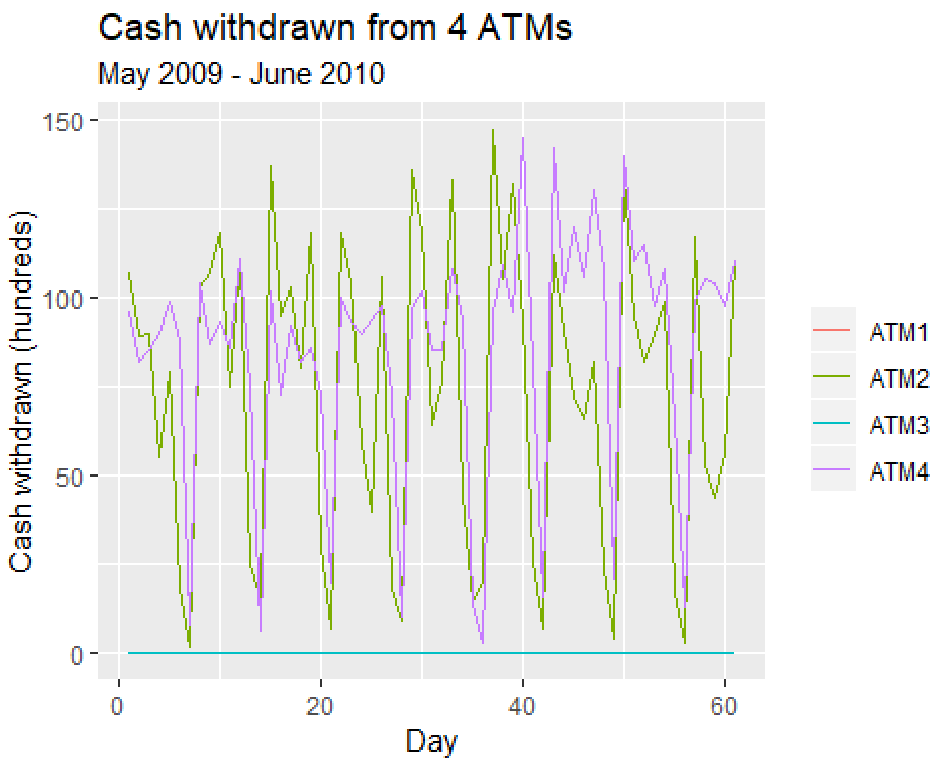



```
#Separate each ATM from the dataset and graph each dataset
atm4<-ts_atm[,"ATM4"]
autoplot(atm4) +
  labs(title ="Cash Withdrawn from ATM4", x="Day") +
  scale_y_continuous("Cash withdrawns in hundreds") +
  scale_color_discrete(NULL)
```



- ATM1, ATM2 and ATM4 show a lot of deal of seasonality in the withdrawn from those ATM.We can further analyze it by selecting the first 2 months of the data.

```
autoplot(ts(ts_atm[1:61, ])) +

  labs(title = "Cash withdrawn from 4 ATMs",

       subtitle = "May 2009 - June 2010",

       x = "Day") +

  scale_y_continuous("Cash withdrawn (hundreds)") +

  scale_color_discrete(NULL)
```

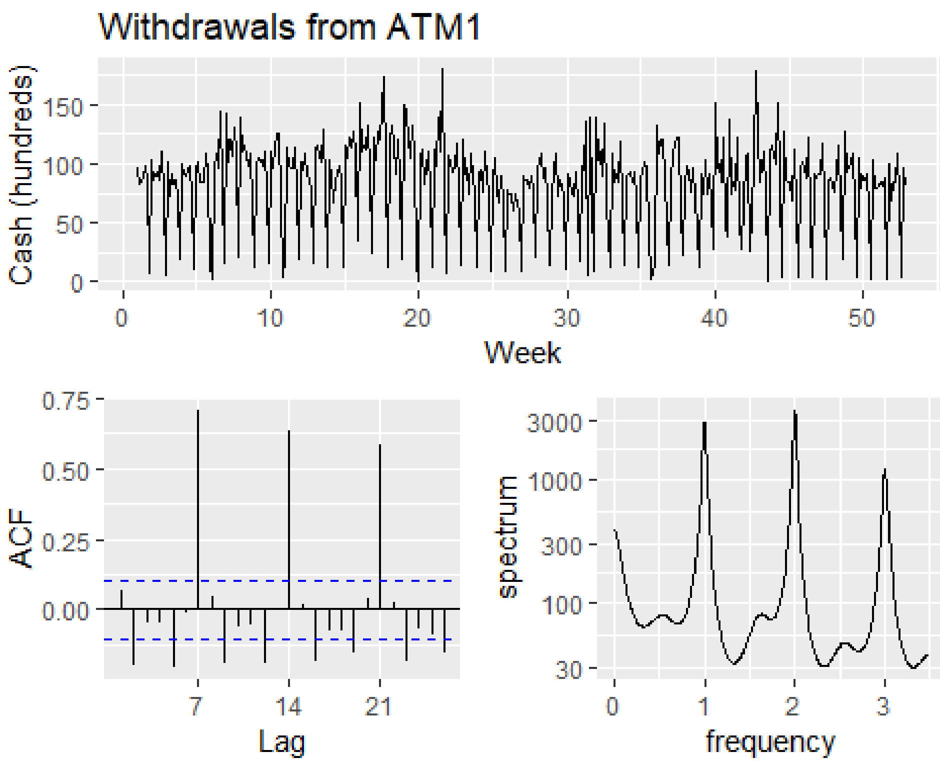


- The data presents a sort of weekly seasonnality. To capture the seasonnality of this data we will set the frequency to 7.

```
atm1_freq<-ts(atm1, frequency =7)
atm2_freq<-ts(atm2, frequency=7)
atm4_freq<-ts(atm4, frequency=7)
```

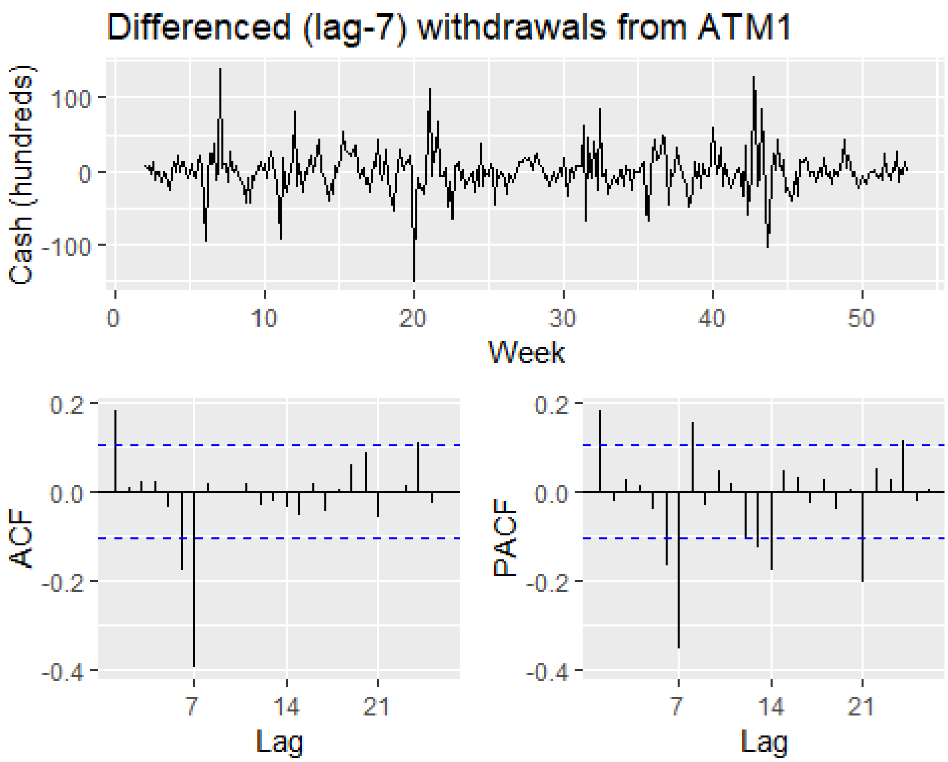
ATM1

```
#ACF and spectrum plot
ggtsdisplay(atm1_freq, points = FALSE, plot.type = "spectrum",

            main = "Withdrawals from ATM1", xlab = "Week", ylab = "Cas
h (hundreds)")
```



- In 7, 14 and 21 there are large spikes. the frequency 1,2,3 show the spike as well. Both suggest a seasonnal ARIMA model.

```
ggtsdisplay(diff(atm1_freq, 7), points = FALSE,

           main = "Differenced (lag-7) withdrawals from ATM1",

           xlab = "Week", ylab = "Cash (hundreds)")
```



- BoxCox transformation to estimate lambda

```r
# get optimal lambda for Box-cox transformation

lambda_atm1<- BoxCox.lambda(atm1_freq)

# define function to create models & return AIC values for timeseries

aic_atm<- function(p, d, q, P, D, Q) {

  # create model with Box-Cox and specified ARIMA parameters; extract AIC

  AIC(Arima(atm1_freq, order = c(p, d, q), seasonal = c(P, D, Q), lambda = lambda_atm1))

}

# create possible combinations of p, q, P, Q except all zero

expand.grid(p = 0:1, q = 0:1, P = 0:1, Q = 0:1) %>%

  filter(p > 0 | q > 0 | P > 0 | Q > 0) %>%

  # calc AIC for models
```

```r
  mutate(aic = pmap_dbl(list(p, 0, q, P, 1, Q), aic_atm)) %>%

  # return best AIC

  slice(which.min(aic))
```

```
##   p q P Q     aic
## 1 1 1 0 1 1221.26
```

- The minimum aic value is for non-seasonality AR(1) and MA(1). AR(0) and AM(1) is for seasonality. Let's fit the model using arima model arima(1,0,1)(0,1,1)

```r
fit_atm1 <- Arima(atm1_freq, order = c(1, 0, 1), seasonal = c(0, 1, 1)
, lambda = lambda_atm1)
summary(fit_atm1)
```

```
## Series: atm1_freq
## ARIMA(1,0,1)(0,1,1)[7]
## Box Cox transformation: lambda= 0.2584338
##
## Coefficients:
##           ar1     ma1     sma1
##       -0.4894  0.6125  -0.6385
## s.e.   0.2309  0.2081   0.0432
##
## sigma^2 estimated as 1.732:  log likelihood=-606.63
## AIC=1221.26   AICc=1221.37   BIC=1236.78
##
## Training set error measures:
##                    ME     RMSE      MAE       MPE     MAPE     MASE
## Training set 2.293003 24.81988 15.66437 -89.57546 108.1682 0.892827
##                      ACF1
## Training set -0.008839946
```

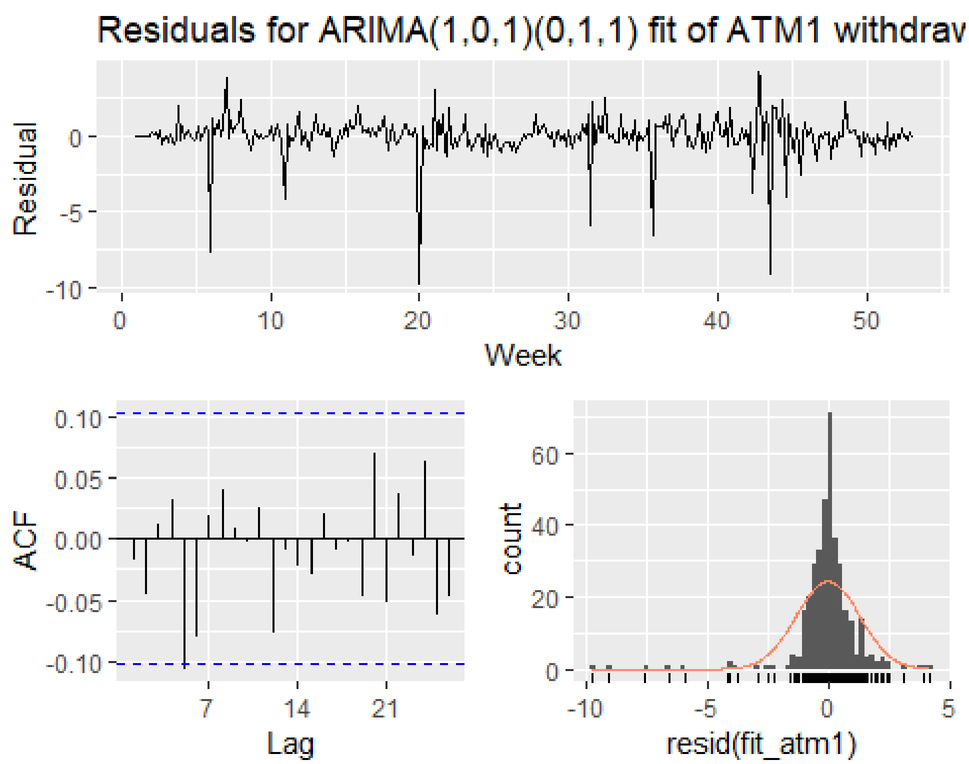
Let's diagnostic the residuals with Ljung-Box.

```r
Box.test(resid(fit_atm1), type = "L", fitdf = 3, lag = 7)
```

```
##
##  Box-Ljung test
##
## data:  resid(fit_atm1)
## X-squared = 8.0497, df = 4, p-value = 0.08977
```

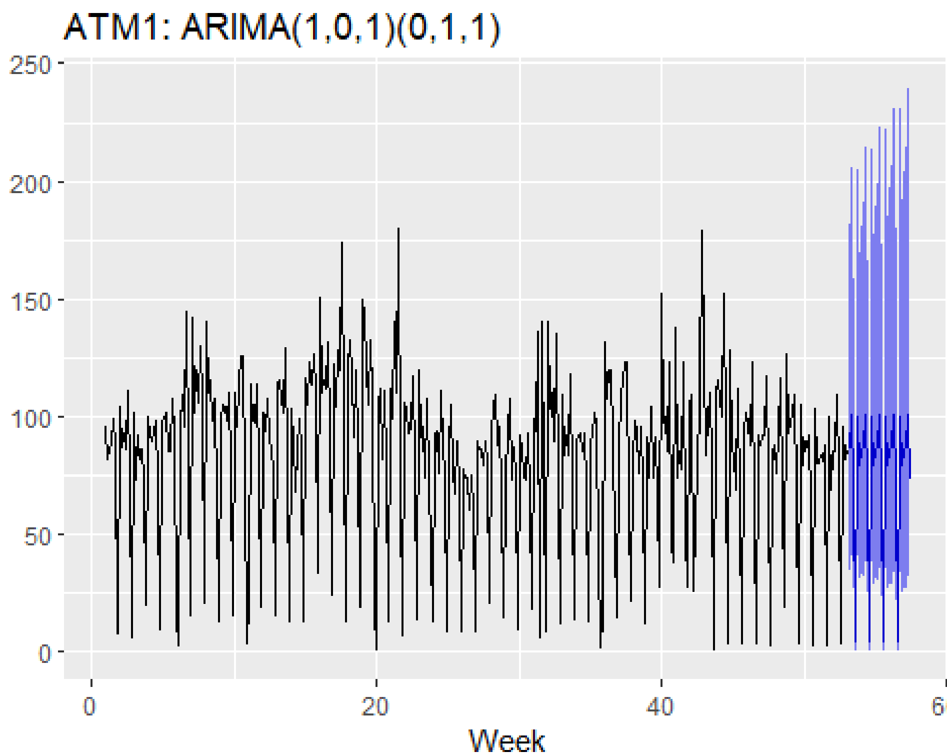
```r
ggtsdisplay(resid(fit_atm1), points = FALSE, plot.type = "histogram",

            main = "Residuals for ARIMA(1,0,1)(0,1,1) fit of ATM1 with
drawals",
```

```
           xlab = "Week", ylab = "Residual")
```



- The p_value is greater than 0.05 meaning that the residual is white noise. The residuals are not correlated and there is a normal distribution around the mean 0. We can use that model for forecasting.

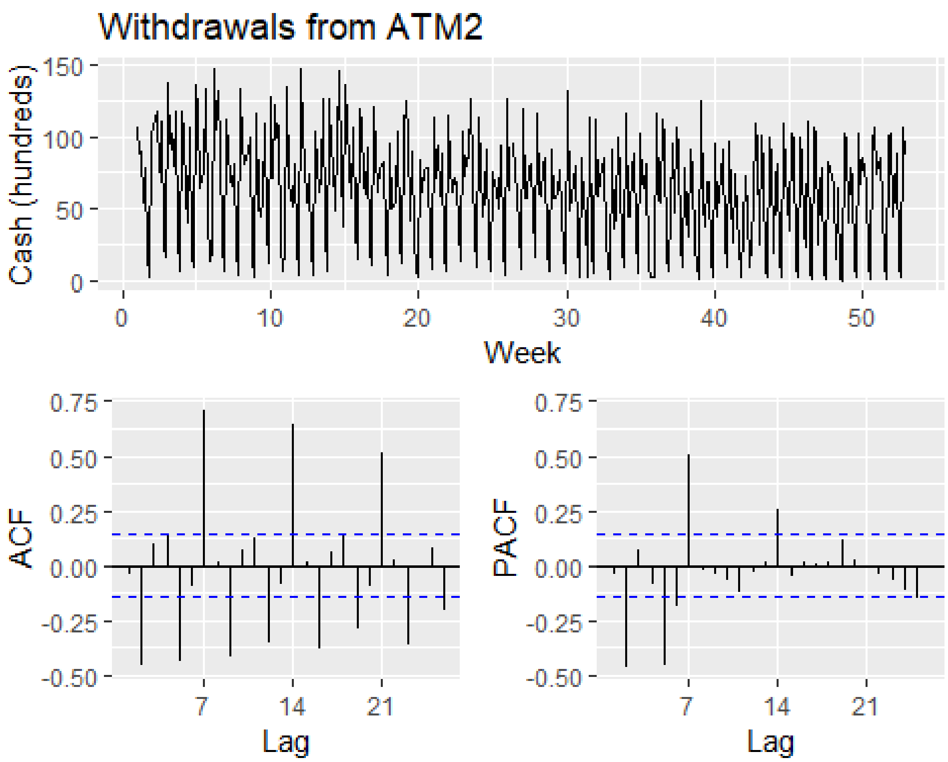
```
forecast_atm1 <- forecast(fit_atm1, 31, level = 95)
autoplot(forecast_atm1) +

    labs(title = "ATM1: ARIMA(1,0,1)(0,1,1)", x = "Week", y = NULL) +

    theme(legend.position = "none")
```



ATM2

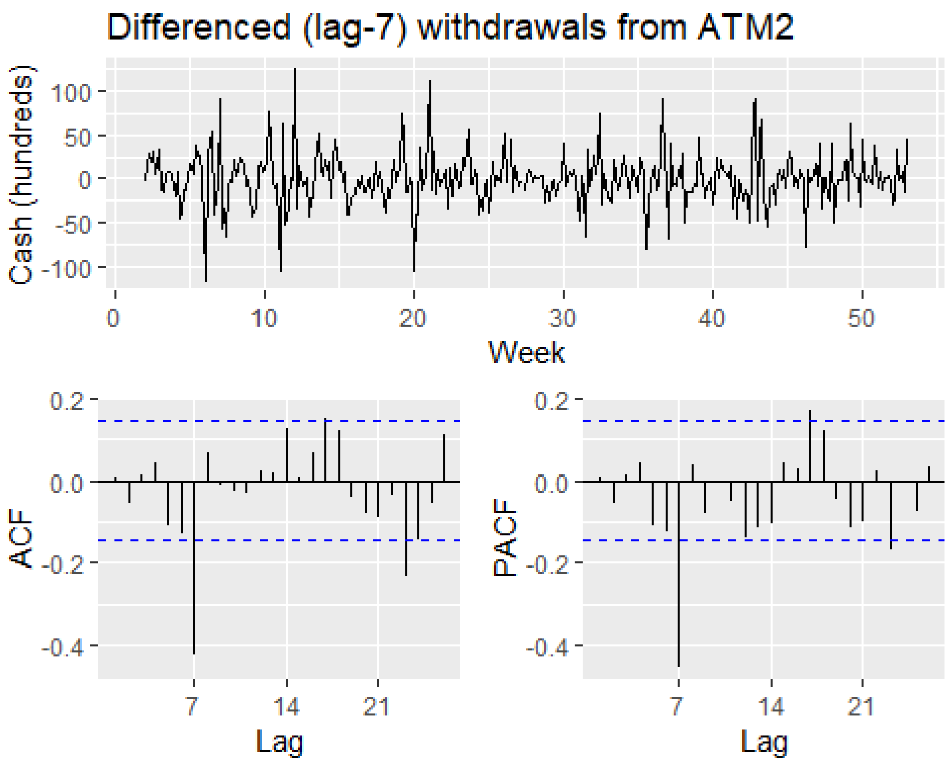
- We can repeat the same stepp for ATM2.

```
ggtsdisplay(atm2_freq, points = FALSE,

           main = "Withdrawals from ATM2", xlab = "Week", ylab = "Cas
h (hundreds)")
```



- The lag difference is 7.

```
ggtsdisplay(diff(atm2_freq, 7), points = FALSE,

            main = "Differenced (lag-7) withdrawals from ATM2",

            xlab = "Week", ylab = "Cash (hundreds)")
```



- The spikes in ACF & PACF in the non-differenced series at $k = 2$ & $k = 5$ suggest $p, q \in [0,2,5]$. using the same aic function we can evaluate the minimum aic

```
# get optimal lambda for Box-cox transformation

lambda_atm2 <- BoxCox.lambda(atm2_freq)

# Evaluate aic

aic_atm <- function(p, d, q, P, D, Q) {

  # create model with Box-Cox and specified ARIMA parameters; extract
AIC

  AIC(Arima(atm2_freq, order = c(p, d, q), seasonal = c(P, D, Q), lamb
da = lambda_atm2))

}

# create possible combinations of p, q, P, Q except all zero

expand.grid(p = c(0, 2, 5), q = c(0, 2, 5), P = 0:1, Q = 0:1) %>%

  filter(p > 0 | q > 0 | P > 0 | Q > 0) %>%

  # calculate AIC for models
```

```
  mutate(aic = pmap_dbl(list(p, 0, q, P, 1, Q), aic_atm)) %>%

  # return minimum AIC

  slice(which.min(aic))
```

```
##   p q P Q      aic
## 1 2 2 0 1 2323.517
```

- the model arima used is arima(5,0,5)(0,1,1). Let’s fit that model

```
fit_atm2<-Arima(atm2_freq, order = c(5, 0, 5), seasonal = c(0, 1, 1),
lambda = lambda_atm2)
summary(fit_atm2)
```

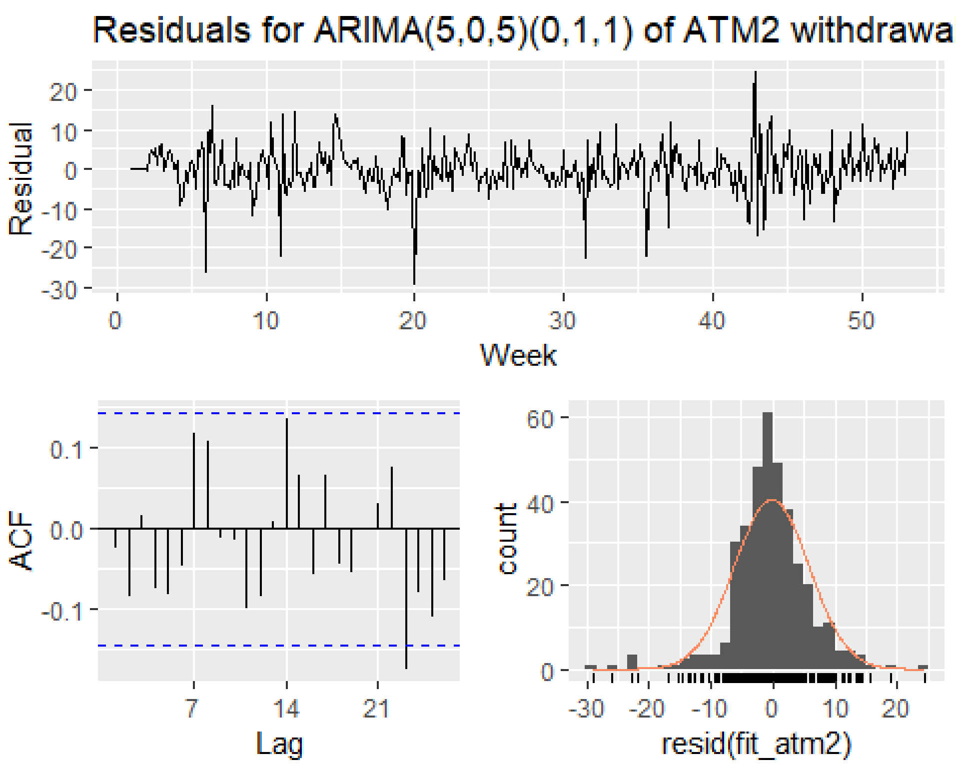
```
## Series: atm2_freq
## ARIMA(5,0,5)(0,1,1)[7]
## Box Cox transformation: lambda= 0.6584081
##
## Coefficients:
##          ar1      ar2     ar3     ar4      ar5      ma1     ma2
  ma3
##       0.2055  -0.1209  0.2260  0.3032  -0.4312  -0.1448  0.0114  -0
.2213
## s.e.  0.4529   0.4033  0.2176  0.2419   0.4136   0.4787  0.4200   0
.2100
##          ma4     ma5     sma1
##      -0.2466  0.2470  -0.6905
## s.e.  0.2463  0.4176   0.0595
##
## sigma^2 estimated as 37.91:  log likelihood=-1152.1
## AIC=2328.19   AICc=2329.1   BIC=2374.76
##
## Training set error measures:
##                     ME     RMSE      MAE  MPE MAPE      MASE
 ACF1
## Training set 0.2238867 23.87153 16.65107 -Inf  Inf 0.8279025 -0.030
50682
```

- Let’s evaluate the residual to check the validity of the model

```
Box.test(resid(fit_atm2), type = "L", fitdf = 11, lag = 14)
```

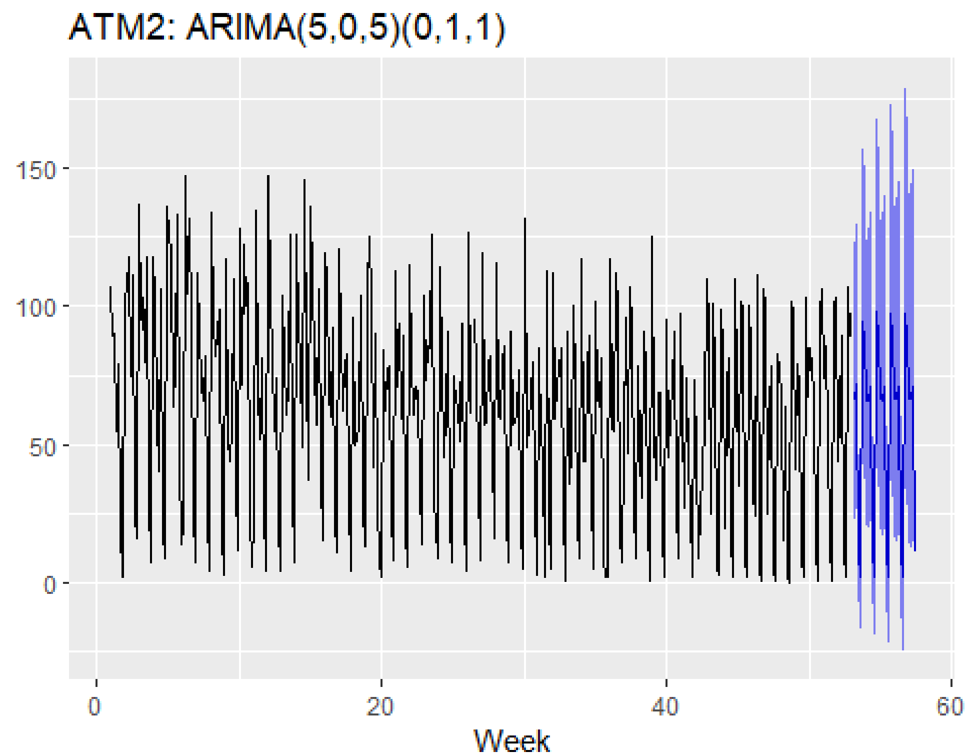
```
##
##  Box-Ljung test
##
## data:  resid(fit_atm2)
## X-squared = 2.1119, df = 3, p-value = 0.5495
```

```r
ggtsdisplay(resid(fit_atm2), points = FALSE, plot.type = "histogram",
            main = "Residuals for ARIMA(5,0,5)(0,1,1) of ATM2 withdraw
als",
            xlab = "Week", ylab = "Residual")
## Warning: Removed 1 rows containing non-finite values (stat_bin).
```



- P-value is greater than 0.05 and the residual appear to be normally distributed with a mean of 0. It can be used for forecast ATM2.

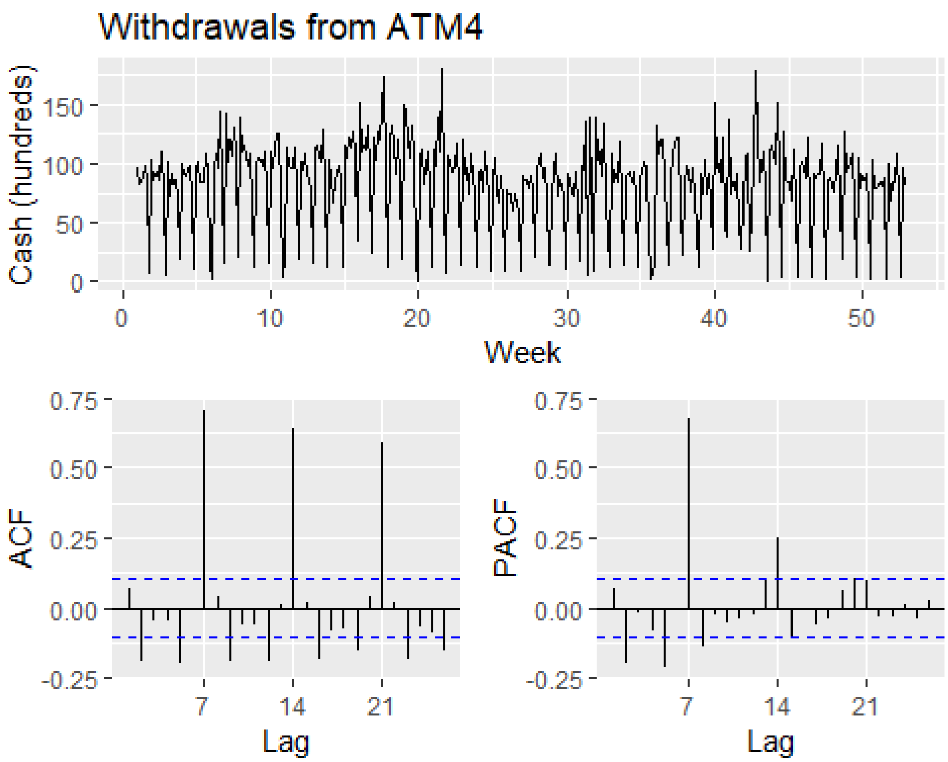
```r
forecast_atm2<- forecast(fit_atm2, 31, level = 95)
autoplot(forecast_atm2) +
    labs(title = "ATM2: ARIMA(5,0,5)(0,1,1)", x = "Week", y = NULL) +
    theme(legend.position = "none")
```



ATM4

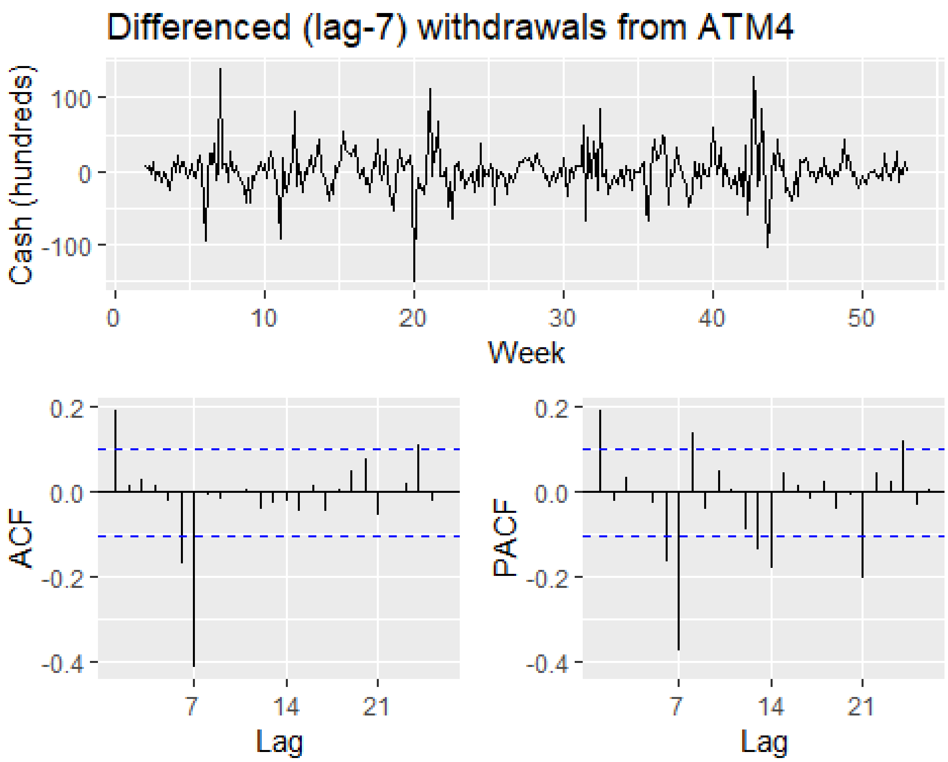
- ATM4 has the same seasonality as ATM1 and ATM2. We will use the previous step to evaluate ATM4 model.

```
#Minimze the effect of the big withdraw in the day by using the median
 of the ATM4 dataset

atm4_freq[which.max(atm4_freq)] <- median(atm4_freq, na.rm = TRUE)

ggtsdisplay(atm4_freq, points = FALSE,

            main = "Withdrawals from ATM4", xlab = "Week", ylab = "Cas
h (hundreds)")
```



- We notice a difference lag of 7.

```r
ggtsdisplay(diff(atm4_freq, 7), points = FALSE,
            main = "Differenced (lag-7) withdrawals from ATM4",
            xlab = "Week", ylab = "Cash (hundreds)")
```



- ARIMA model for ATM4 will be evaluated.

```r
# get optimal lambda for Box-cox transformation

lambda_atm4 <- BoxCox.lambda(atm4_freq)

aic_atm(0,2,5,0,2,5)

## [1] 2365.837

# create possible combinations of p, q, P, Q except all zero

expand.grid(p = c(0, 2, 5), q = c(0, 2, 5), P = 0:1, Q = 0:1) %>%

  filter(p > 0 | q > 0 | P > 0 | Q > 0) %>%

  # calculate AIC for models

  mutate(aic = pmap_dbl(list(p, 0, q, P, 1, Q), aic_atm)) %>%

  # return minimum AIC

  slice(which.min(aic))

##   p q P Q      aic
## 1 2 2 0 1 2323.517
```

- Let’s fit the ARIMA model with the values (0,0,2)(0,1,1)

```
fit_atm4<-Arima(atm4_freq, order = c(0, 0, 2), seasonal = c(0, 1, 1),
lambda = lambda_atm4)
summary(fit_atm4)
```

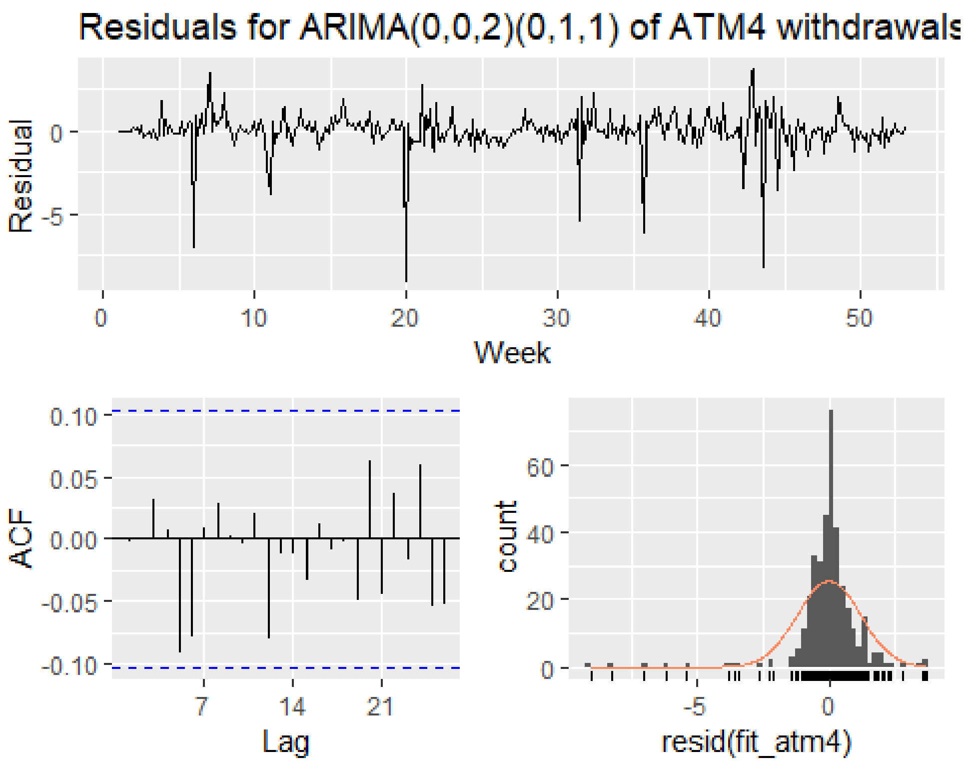
```
## Series: atm4_freq
## ARIMA(0,0,2)(0,1,1)[7]
## Box Cox transformation: lambda= 0.2355973
##
## Coefficients:
##          ma1      ma2     sma1
##       0.1094  -0.1089  -0.6468
## s.e.  0.0524   0.0523   0.0422
##
## sigma^2 estimated as 1.467:  log likelihood=-576.96
## AIC=1161.92   AICc=1162.03   BIC=1177.44
##
## Training set error measures:
##                    ME     RMSE      MAE       MPE     MAPE      MAS
E
## Training set 2.356651 24.88094 15.90136 -85.71176 104.5953 0.902312
3
##                    ACF1
## Training set 0.02127326
```

- Let’s investigate the residuals using Ljung-box test

```
Box.test(resid(fit_atm4), type = "L", fitdf = 3, lag = 7)
```

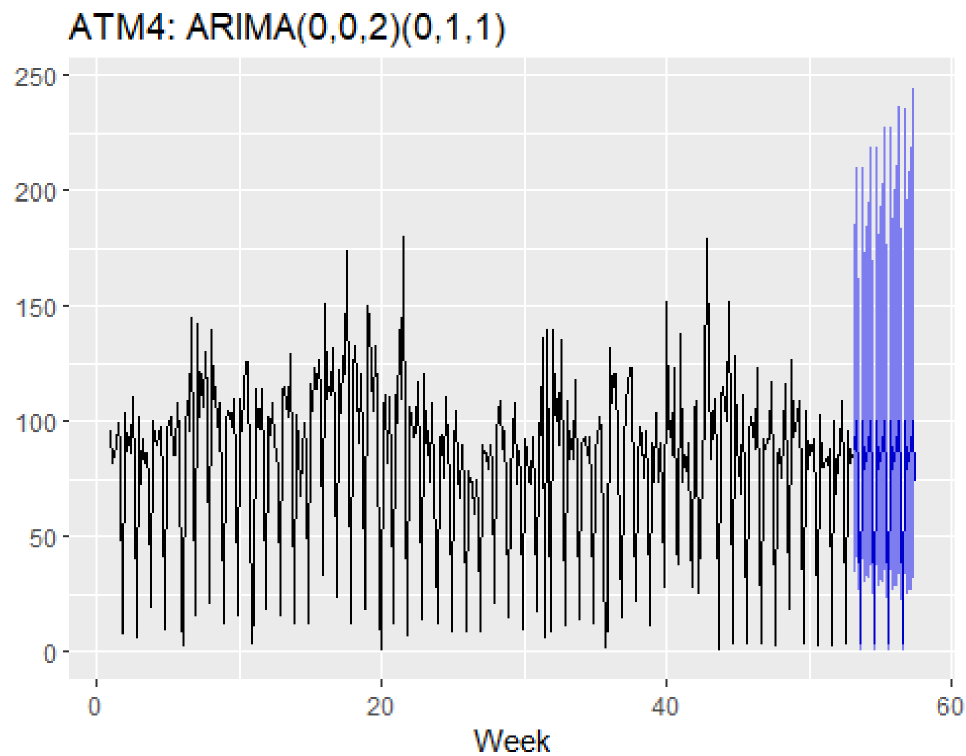
```
##
##  Box-Ljung test
##
## data:  resid(fit_atm4)
## X-squared = 5.7899, df = 4, p-value = 0.2154
```

```
ggtsdisplay(resid(fit_atm4), points = FALSE, plot.type = "histogram",

            main = "Residuals for ARIMA(0,0,2)(0,1,1) of ATM4 withdraw
als",

            xlab = "Week", ylab = "Residual")
```



- It is normally distributed around a mean of 0.p-value is also greater than 0.05. We can use the model to forecast.

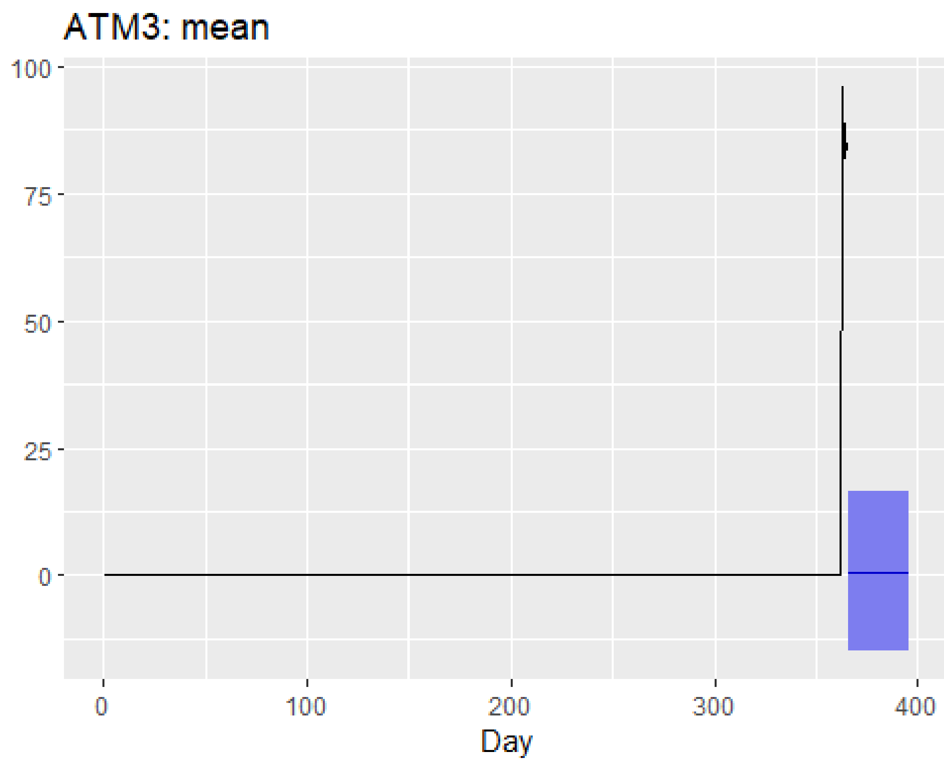
```
forecast_atm4<- forecast(fit_atm4, 31, level = 95)
autoplot(forecast_atm4) +

    labs(title = "ATM4: ARIMA(0,0,2)(0,1,1)", x = "Week", y = NULL) +

    theme(legend.position = "none")
```



ATM3

- Since ATM3 contains limited data we will use the mean forecast method.

```
forecast_atm3 <- meanf(atm3, 31, level = 95)
autoplot(forecast_atm3) +

    labs(title = "ATM3: mean", x = "Day", y = NULL) +

    theme(legend.position = "none")
```



Writing the forecast to a CSV file

```
data_frame(DATE = rep(max(atm$DATE) + 1:31, 4),

           atm = rep(names(atm)[-1], each = 31),

           Cash = c(forecast_atm1$mean, forecast_atm2$mean,

                    forecast_atm3$mean, forecast_atm4$mean)) %>%

  write_csv("project1_forecast_atm_1.csv")

## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

Data 624. Project 1: Part B

Angrand, Burke, Deboch, Groysman, Karr

October 22, 2019

Part B - Forecasting Power, ResidentialCustomerForecastLoad-624.xlsx

Data: ResidentialCustomerForecastLoad-624.xlsx

Part B consists of a simple dataset of residential power usage for January 1998 until December 2013. Your assignment is to model these data and a monthly forecast for 2014. The data is given in a single file. The variable 'KWH' is power consumption in Kilowatt hours, the rest is straight forward. Add these to your existing files above - clearly labeled.

```
library(httr)
library(xlsx)
library(ggplot2)
library(gridExtra)
library(forecast)
```

a. Read in File

```
temp = tempfile(fileext = ".xlsx")
dataURL <- "https://raw.githubusercontent.com/mburke65/CUNY_Data624/ma
ster/ProjectFolder/Provided_Files/ResidentialCustomerForecastLoad-624.
xlsx"
download.file(dataURL, destfile=temp, mode='wb')

power.data <- readxl::read_excel(temp, sheet =1)
```

b. EDA Analysis

```
- Check/Fill in null values
- Convert to time series
- Graph the monthly data
  - General plot & seasonal plot: seasonality can be observed in the b
elow plot. There are spikes each year from May to August (air conditio
ning?) and again in December (holiday season?). There is a slight dip
in Jul 2010 maybe due to an unseasonably cold month.
  - Seasonal Box Plot: provides a similar visual to the seasonal plot
with usage spikes in the summer months and December. IT also highlight
s the flucuations in consumption within each month.
  - Decomposition components graph: this plot again shows that there i
s a general upwards trend in the data with an observed outlier in July
 2010.
```

```r
head(power.data)
```

```
## # A tibble: 6 x 3
##   CaseSequence `YYYY-MMM`     KWH
##          <dbl> <chr>        <dbl>
## 1          733 1998-Jan   6862583
## 2          734 1998-Feb   5838198
## 3          735 1998-Mar   5420658
## 4          736 1998-Apr   5010364
## 5          737 1998-May   4665377
## 6          738 1998-Jun   6467147
```

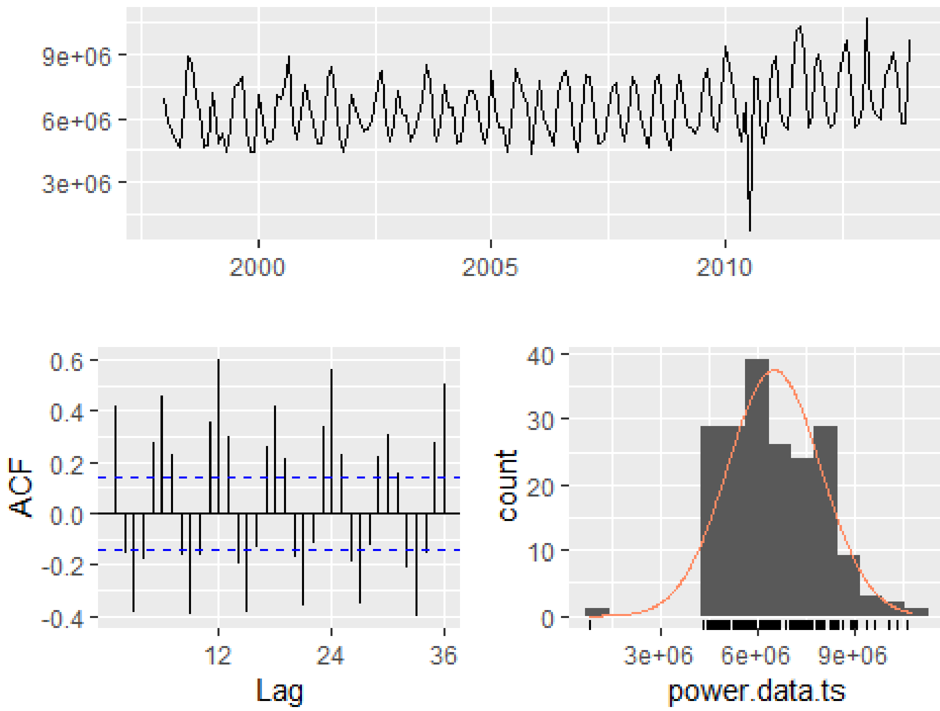
```r
summary(power.data)
```

```
##   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
```

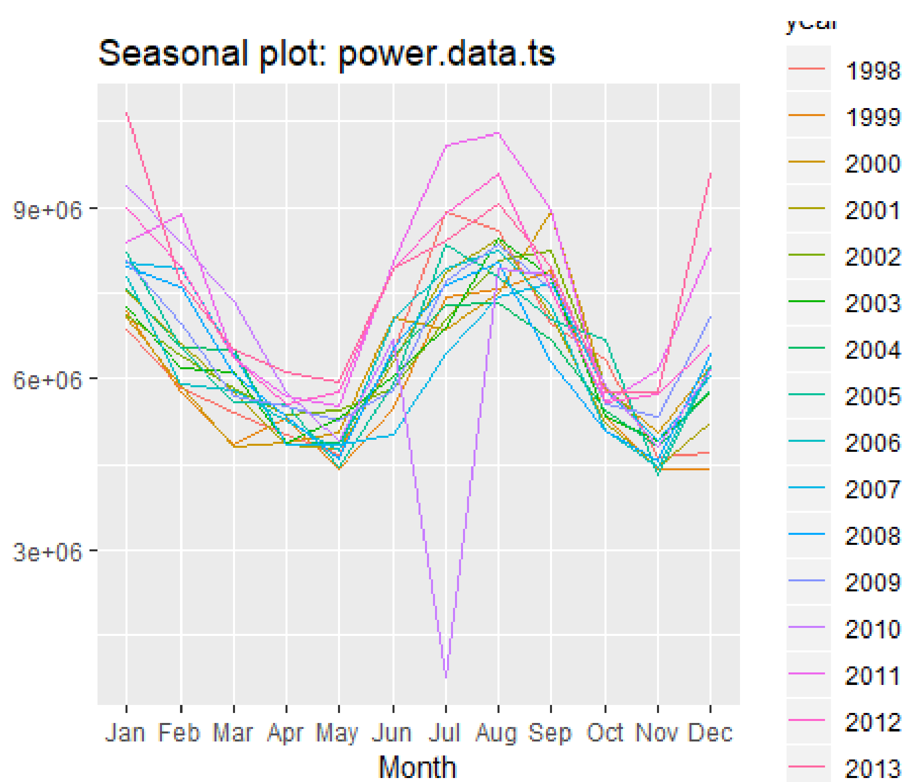
```r
print(paste("Check for nulls: ",sum(is.na(power.data)), " Row of Nulls
"))
```

```
## [1] "Check for nulls:  1  Row of Nulls"
```

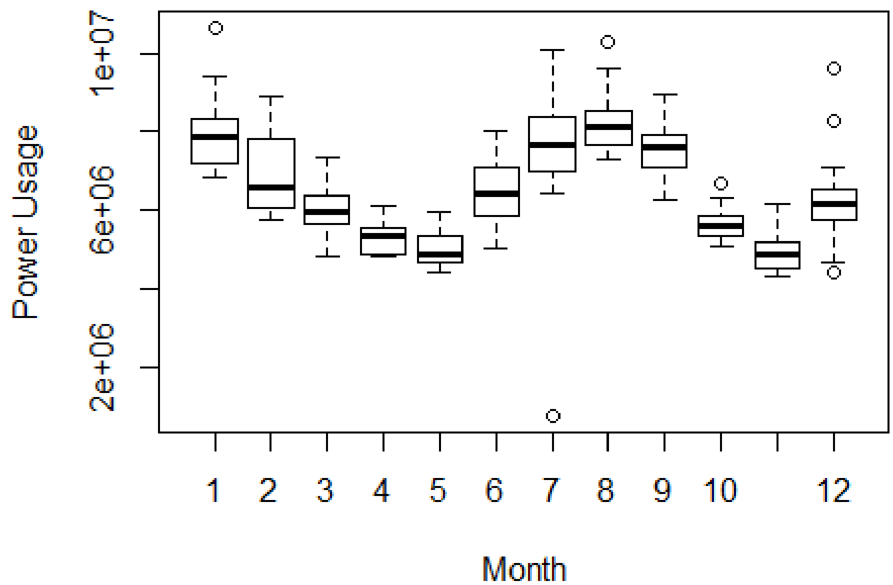
```r
power.data[is.na(power.data)] <- median(power.data$KWH,na.rm = TRUE)
power.data.ts <-ts(power.data[,"KWH"],start = c(1998,1),frequency = 12
)
ggtsdisplay(power.data.ts, points = FALSE, plot.type = "histogram")
```



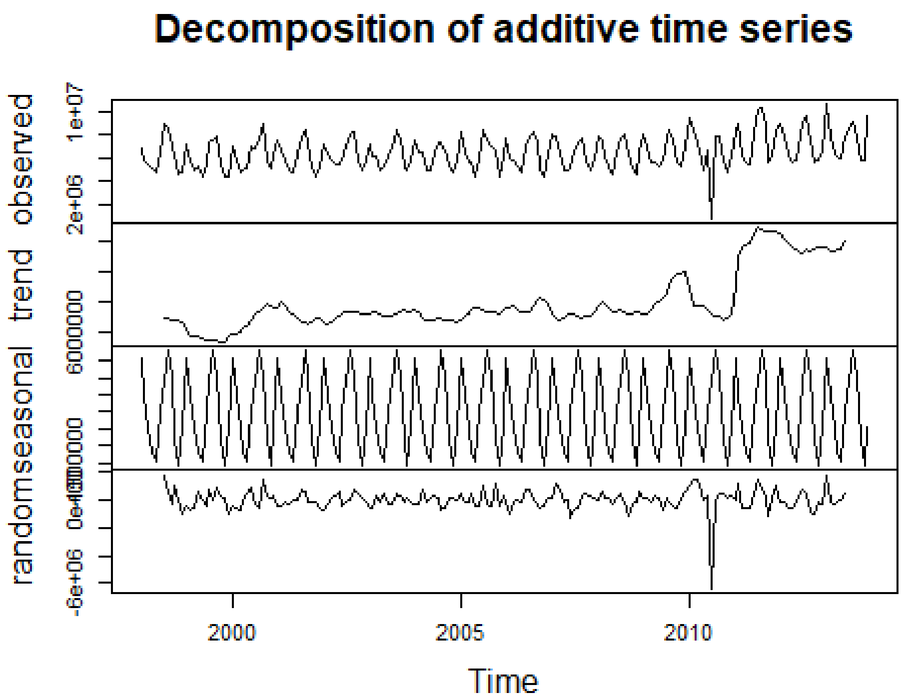

```
ggseasonplot(power.data.ts)
```



```
boxplot(power.data.ts~cycle(power.data.ts),xlab="Month", ylab = "Power
 Usage")
```




```
plot(decompose(power.data.ts))
```



C. Model 1: Arima W/ Box-Cox Transformation

```
- Replace outlier with tsoutlier suggestion (utilizes a box-cox transf
ormation)
- Use an auto arima model on the box-cox adjusted data
  - Suggested model: ARIMA(0,0,3)(2,1,0)[12] with drift. RSME(595389)
& AICc (5332.67)
- Check the residuals to make sure the model is satisfactory:
  - ACF /PACF Plots: the residual appears normal residuals mostly arou
nd 0, suggesting stationarity of the residuals
  - The Box Ljung tests presents a p-value of 0.6951 which indicates w
hite noise
- Forecast 2014 power values & plot forecasted values
```

```r
#outlier detection/suggestion/replacement
find.outlier<- tsoutliers(power.data.ts, iterate = 2, lambda = "auto")
power.data.ts.bc<- power.data.ts
power.data.ts.bc[find.outlier$`index`[1]] <- find.outlier$replacements
[1]
print(paste("Suggested/Implemented Change for Outlier: ",power.data.ts
.bc[151], " Original Value",power.data.ts[151]))
```

```
## [1] "Suggested/Implemented Change for Outlier:  7757388.48810024  O
riginal Value 770523"
```

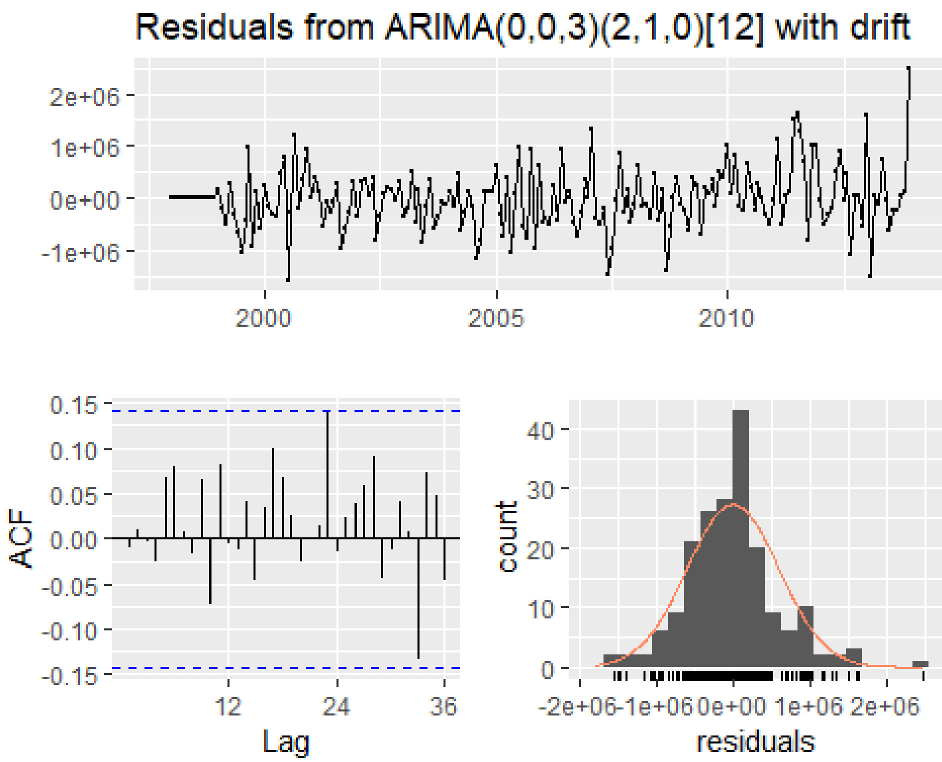
```
#auto arima model
power.model <- auto.arima(power.data.ts.bc, seasonal = TRUE, stepwise
= FALSE)
summary.arima<- summary(power.model)
```

```
## Series: power.data.ts.bc
## ARIMA(0,0,3)(2,1,0)[12] with drift
##
## Coefficients:
##          ma1     ma2     ma3     sar1     sar2     drift
##       0.3492  0.0587  0.2303  -0.7222  -0.4251  9027.233
## s.e.  0.0788  0.0892  0.0741   0.0765   0.0784  3057.838
##
## sigma^2 estimated as 3.912e+11:  log likelihood=-2659.01
## AIC=5332.02   AICc=5332.67   BIC=5354.37
##
## Training set error measures:
##                     ME   RMSE      MAE        MPE     MAPE      MAS
E
## Training set -8181.906 595389 434520.9 -0.8060555 6.610412 0.697749
1
##                     ACF1
## Training set -0.01026567
```

```
summary.arima
```

```
##                     ME   RMSE      MAE        MPE     MAPE      MAS
E
## Training set -8181.906 595389 434520.9 -0.8060555 6.610412 0.697749
1
##                     ACF1
## Training set -0.01026567
```

```
#check residuals
checkresiduals(power.model)
```

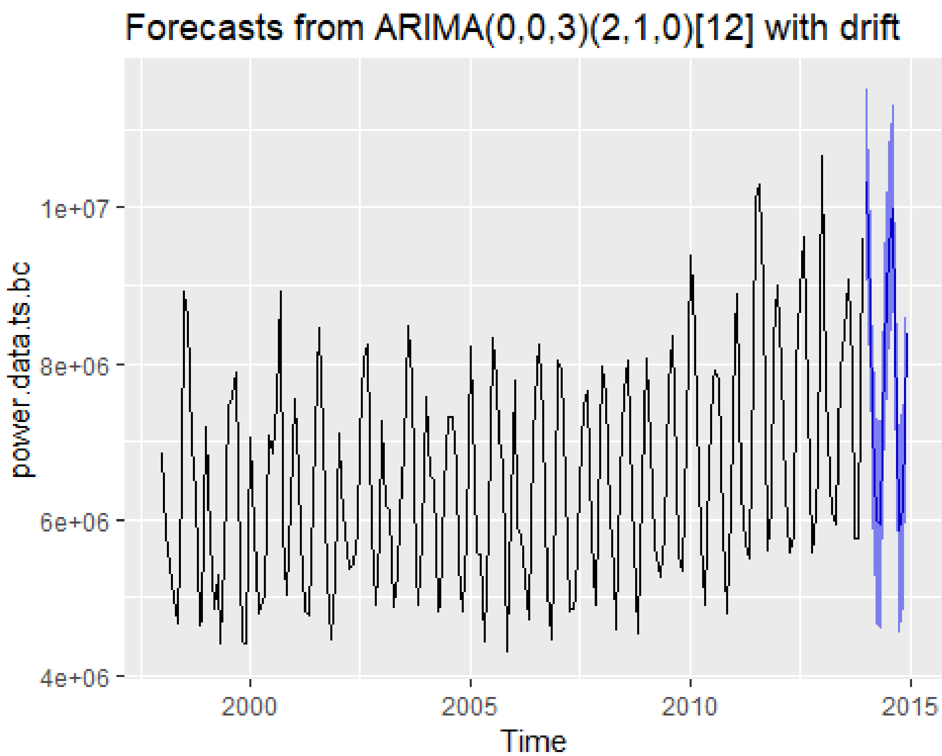



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,0,3)(2,1,0)[12] with drift
## Q* = 14.513, df = 18, p-value = 0.6951
##
## Model df: 6.   Total lags used: 24
```

```
#forecast model @ 95%
forecast.power <- forecast(power.model, level = c(95), h =12)
forecast.power
```

```
##          Point Forecast    Lo 95    Hi 95
## Jan 2014       10312755 9086940 11538570
## Feb 2014        8685531 7387130  9983933
## Mar 2014        7203085 5902687  8503482
## Apr 2014        6000251 4669575  7330927
## May 2014        5941905 4611229  7272581
## Jun 2014        8204931 6874255  9535607
## Jul 2014        9501418 8170742 10832094
## Aug 2014        9992966 8662290 11323642
## Sep 2014        8493959 7163283  9824635
## Oct 2014        5871672 4540996  7202348
## Nov 2014        6154352 4823676  7485028
## Dec 2014        8381806 7051130  9712482
```

```r
autoplot(forecast.power)
```



D. Model 2: ETS W/ Box-Cox Transformation

```
- The ets function automatically selects the best method for forecasti
ng data. the ets function selected ETS(M,N,M) exponential smoothing:
  - The first letter denotes the error type: multiplicative errors
  - The second letter denotes the trend type: no trend
  - The third letter denotes the season type: multiplicative seasonali
ty
- Utilize the transformed data & ETS model
- Model Results: RMSE (630869.7) & AICc (6148.032)
- Check the residuals to make sure the model is satisfactory:
  - ACF /PACF Plots: the residual appears normal residuals mostly arou
nd 0, suggesting stationarity of the residuals
  - The Box Ljung tests presents a p-value of 0.0002921 which may indi
cate that there's dependency issues with the lags
```

```r
#model w/ previously transformed data
power.model.ets <- ets(power.data.ts.bc)
summary.ets<- summary(power.model.ets)
```

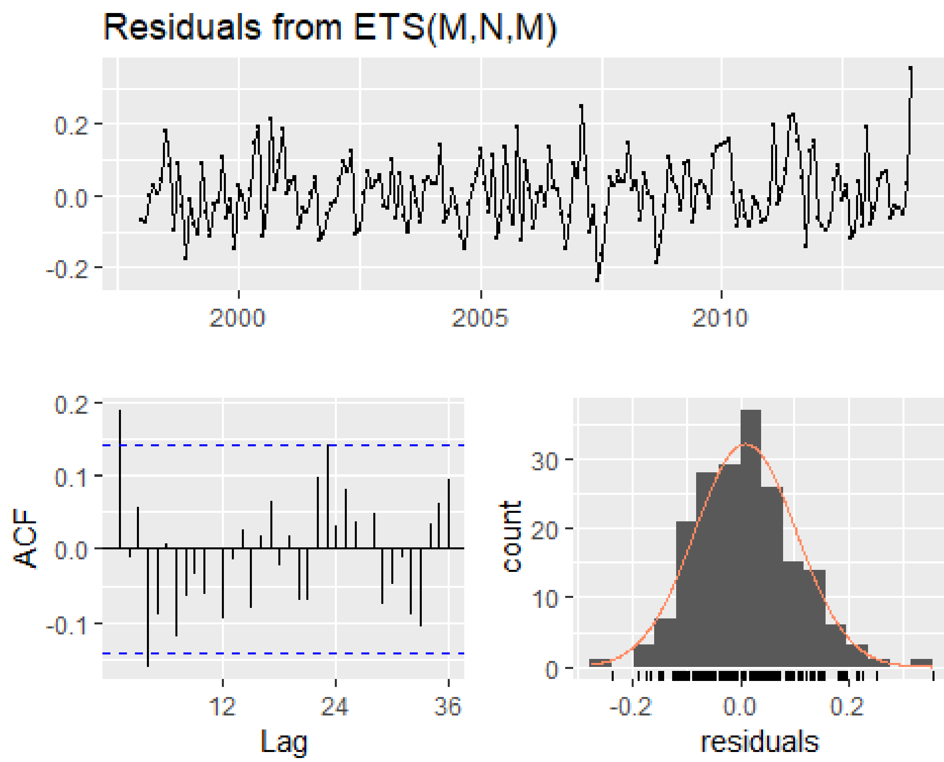
```
## ETS(M,N,M)
##
## Call:
```

```
##  ets(y = power.data.ts.bc)
##
##   Smoothing parameters:
##     alpha = 0.1206
##     gamma = 0.203
##
##   Initial states:
##     l = 6188160.6435
##     s = 0.9017 0.755 0.9295 1.223 1.2676 1.2298
##            1.0165 0.7614 0.8029 0.8903 1.029 1.1935
##
##   sigma:  0.0971
##
##      AIC     AICc      BIC
## 6145.305 6148.032 6194.167
##
## Training set error measures:
##                    ME     RMSE      MAE        MPE     MAPE      MA
SE
## Training set 51458.11 630869.7 482886.1 0.04287257 7.292037 0.77541
35
##                   ACF1
## Training set 0.2096574
```

```
summary.ets
```

```
##                    ME     RMSE      MAE        MPE     MAPE      MA
SE
## Training set 51458.11 630869.7 482886.1 0.04287257 7.292037 0.77541
35
##                   ACF1
## Training set 0.2096574
```

```
#check residuals
checkresiduals(power.model.ets)
```

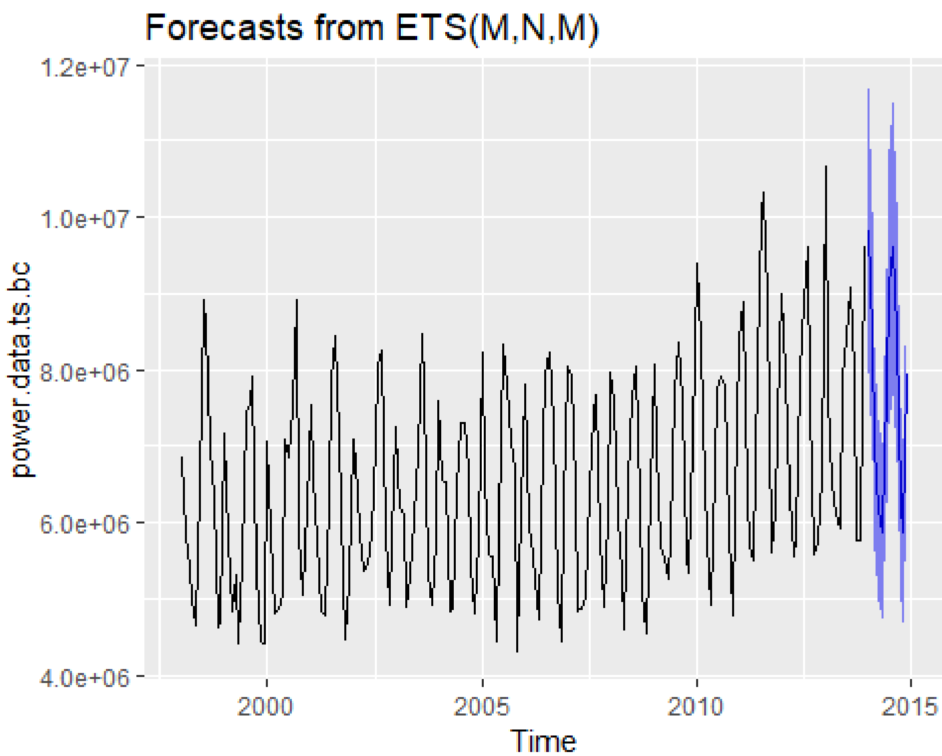



```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,N,M)
## Q* = 32.819, df = 10, p-value = 0.0002921
##
## Model df: 14.   Total lags used: 24
```

```
#forecast model @ 95%
forecast.power.ets <- forecast(power.model.ets, level = c(95), h =12)
forecast.power.ets
```

```
##          Point Forecast   Lo 95    Hi 95
## Jan 2014        9825114 7955694 11694534
## Feb 2014        8460361 6838842 10081879
## Mar 2014        6974291 5627960  8320623
## Apr 2014        6167737 4968643  7366830
## May 2014        5886368 4733958  7038779
## Jun 2014        7783200 6248904  9317496
## Jul 2014        9070884 7270556 10871212
## Aug 2014        9599368 7681334 11517403
## Sep 2014        8501578 6791613 10211542
## Oct 2014        6241977 4978271  7505684
## Nov 2014        5885873 4686553  7085194
## Dec 2014        7933193 6306380  9560005
```

```r
autoplot(forecast.power.ets)
```



E. Model 3: STLF

```
- STLF model will be the third model as it provides the user more cont
rol and can be robust when dealing with outliers. the STLF utilizes a
local weighted regression to fit the points (Loess smoothing) and fore
cast future values.
- Model summary: RMSE (843670.1) & AICc (6255.445)
- Check residuals:
  - ACF/PACF: most lags are within the error bounds, suggesting statio
narity of the residuals
  - Box Ljung:p-value = 0.1457 which indicates white noise
- Forecast 2014 power values & plot forecasted values

  power.model.stl <- stlf(power.data.ts,  s.window='periodic', robust=
TRUE)
summary.stl<- summary(power.model.stl)

##
## Forecast method: STL +  ETS(A,N,N)
##
## Model Information:
## ETS(A,N,N)
##
```

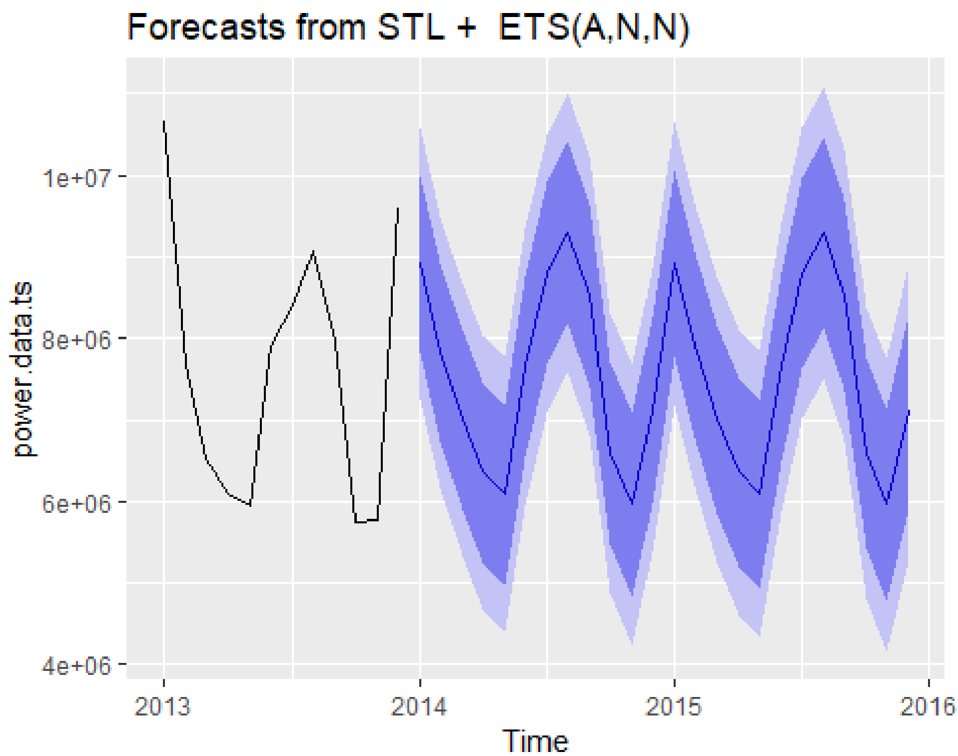
```
## Call:
##  ets(y = x, model = etsmodel, allow.multiplicative.trend = allow.mu
ltiplicative.trend)
##
##   Smoothing parameters:
##     alpha = 0.0892
##
##   Initial states:
##     l = 6317161.2015
##
##   sigma:  848098.8
##
##      AIC     AICc      BIC
## 6255.318 6255.445 6265.090
##
## Error measures:
##                    ME     RMSE      MAE       MPE     MAPE      MAS
E
## Training set 69834.05 843670.1 512067.7 -4.243142 12.03155 0.731642
2
##                   ACF1
## Training set 0.209786
##
## Forecasts:
##          Point Forecast   Lo 80    Hi 80   Lo 95    Hi 95
## Jan 2014        8919230 7832347 10006112 7256987 10581473
## Feb 2014        7833393 6742199  8924586 6164556  9502230
## Mar 2014        7005154 5909665  8100642 5329749  8680558
## Apr 2014        6358706 5258940  7458473 4676759  8040654
## May 2014        6086344 4982317  7190371 4397880  7774808
## Jun 2014        7653295 6545023  8761567 5958339  9348251
## Jul 2014        8801193 7688692  9913693 7099770 10502616
## Aug 2014        9301580 8184867 10418293 7593714 11009445
## Sep 2014        8524478 7403568  9645387 6810194 10238761
## Oct 2014        6593256 5468165  7718347 4872577  8313934
## Nov 2014        5961092 4831835  7090349 4234043  7688141
## Dec 2014        7113767 5980360  8247174 5380371  8847164
## Jan 2015        8919230 7781688 10056772 7179509 10658950
## Feb 2015        7833393 6691730  8975055 6087371  9579415
## Mar 2015        7005154 5859386  8150921 5252853  8757454
## Apr 2015        6358706 5208848  7508565 4600150  8117263
## May 2015        6086344 4932409  7240279 4321553  7851135
## Jun 2015        7653295 6495299  8811292 5882292  9424298
## Jul 2015        8801193 7639149  9963237 7023999 10578386
## Aug 2015        9301580 8135502 10467658 7518218 11084942
## Sep 2015        8524478 7354380  9694575 6734968 10313987
```

```
## Oct 2015        6593256 5419152  7767359 4797619  8388892
## Nov 2015        5961092 4782996  7139188 4159350  7762834
## Dec 2015        7113767 5931693  8295842 5305940  8921594
```

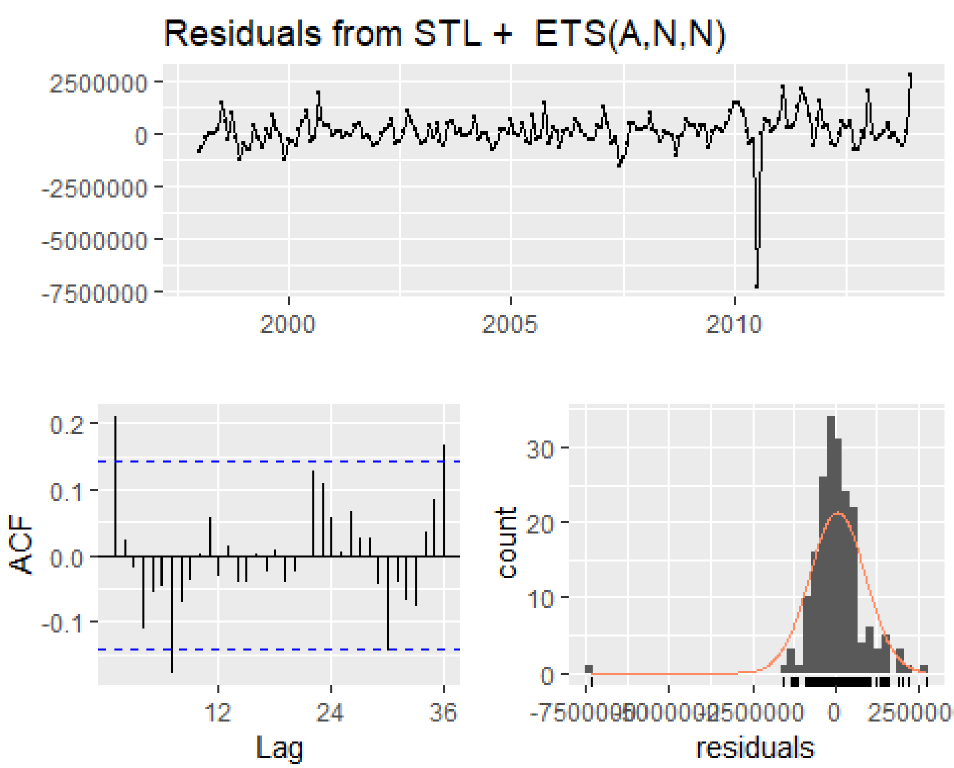
```r
summary.stl
```

```
##          Point Forecast   Lo 80    Hi 80   Lo 95    Hi 95
## Jan 2014        8919230 7832347 10006112 7256987 10581473
## Feb 2014        7833393 6742199  8924586 6164556  9502230
## Mar 2014        7005154 5909665  8100642 5329749  8680558
## Apr 2014        6358706 5258940  7458473 4676759  8040654
## May 2014        6086344 4982317  7190371 4397880  7774808
## Jun 2014        7653295 6545023  8761567 5958339  9348251
## Jul 2014        8801193 7688692  9913693 7099770 10502616
## Aug 2014        9301580 8184867 10418293 7593714 11009445
## Sep 2014        8524478 7403568  9645387 6810194 10238761
## Oct 2014        6593256 5468165  7718347 4872577  8313934
## Nov 2014        5961092 4831835  7090349 4234043  7688141
## Dec 2014        7113767 5980360  8247174 5380371  8847164
## Jan 2015        8919230 7781688 10056772 7179509 10658950
## Feb 2015        7833393 6691730  8975055 6087371  9579415
## Mar 2015        7005154 5859386  8150921 5252853  8757454
## Apr 2015        6358706 5208848  7508565 4600150  8117263
## May 2015        6086344 4932409  7240279 4321553  7851135
## Jun 2015        7653295 6495299  8811292 5882292  9424298
## Jul 2015        8801193 7639149  9963237 7023999 10578386
## Aug 2015        9301580 8135502 10467658 7518218 11084942
## Sep 2015        8524478 7354380  9694575 6734968 10313987
## Oct 2015        6593256 5419152  7767359 4797619  8388892
## Nov 2015        5961092 4782996  7139188 4159350  7762834
## Dec 2015        7113767 5931693  8295842 5305940  8921594
```

```r
power.model.stl<- forecast(power.model.stl)
autoplot(power.model.stl, 12)
```




```r
checkresiduals(power.model.stl)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from STL +  ETS(A,N,N)
## Q* = 28.969, df = 22, p-value = 0.1457
##
## Model df: 2.   Total lags used: 24
```

F. Compare Model Results/Export Data

```
- After comparing the RMSE of in the accuracy test, the ARIMA model wi
ll be used as the final model due to the lower RSME and better predict
ion capabilities. The ARIMA model also has the lowest AICc score and b
est score from the Box Ljung tests.

rmse.list <- data.frame(list(accuracy(power.model)[2], accuracy(power.
model.ets)[2], accuracy(power.model.stl)[2]))
names(rmse.list)<- list('Arima', 'ETS','STL')
rmse.list

##    Arima      ETS      STL
## 1 595389 630869.7 843670.1
```

G. Send Results to excel

```
- Send to a .csv file, will manually merge into the project's consolid
ated file for project submission

write.csv(forecast.power,"Power_Forecasts_ARIMA.csv")
```

Data 624. Project 1: Part C

Team 1. Angrand, Burke, Deboch, Groysman, Karr

10/22/2019

Part C – Waterflow_Pipe1.xlsx and Waterflow_Pipe2.xlsx

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%). Add these to your existing files above – clearly labeled.

Step 1. Load Libraries

```
library(forecast)

library(ggplot2)

library(Hmisc)

library(lubridate)

library(fma)

library(readxl)

library(knitr)

library(seasonal)

library(openxlsx)
```

Step 2. Read in 2 Excel files

```
mdata1 <- read_excel("Waterflow_Pipe1.xlsx")

mdata2 <- read_excel("Waterflow_Pipe2.xlsx")
```

Step 3. Exploratory Analysis.

Let's see domensions, top/bottom records, data types

```
dim(mdata1)

## [1] 1000    2
```

```r
str(mdata1)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    1000 obs. of  2 variab
les:
##  $ DateTime : POSIXct, format: "2015-10-23 00:24:06" "2015-10-23 00
:40:02" ...
##  $ WaterFlow: num  23.4 28 23.1 30 6 ...
```

```r
kable(summary(mdata1))
```

DateTime	WaterFlow
Min. :2015-10-23 00:24:06	Min. : 1.067
1st Qu.:2015-10-25 11:21:35	1st Qu.:13.683
Median :2015-10-27 20:07:30	Median :19.880
Mean :2015-10-27 20:49:15	Mean :19.897
3rd Qu.:2015-10-30 08:24:51	3rd Qu.:26.159
Max. :2015-11-01 23:35:43	Max. :38.913

```r
head(mdata1)
```

```
## # A tibble: 6 x 2
##   DateTime            WaterFlow
##   <dttm>                  <dbl>
## 1 2015-10-23 00:24:06     23.4
## 2 2015-10-23 00:40:02     28.0
## 3 2015-10-23 00:53:51     23.1
## 4 2015-10-23 00:55:40     30.0
## 5 2015-10-23 01:19:17      6.00
## 6 2015-10-23 01:23:58     15.9
```

```r
tail(mdata1)
```

```
## # A tibble: 6 x 2
##   DateTime            WaterFlow
##   <dttm>                  <dbl>
## 1 2015-11-01 22:09:14      15.3
## 2 2015-11-01 22:09:18      26.3
## 3 2015-11-01 22:25:31      29.1
## 4 2015-11-01 23:08:20      22.8
## 5 2015-11-01 23:34:10      16.2
## 6 2015-11-01 23:35:43      21.2
```

```r
dim(mdata2)
```

```
## [1] 1000    2
```

```r
str(mdata2)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    1000 obs. of  2 variab
les:
##  $ DateTime : POSIXct, format: "2015-10-23 01:00:00" "2015-10-23 02
:00:00" ...
##  $ WaterFlow: num  18.8 43.1 38 36.1 31.9 ...
```

```r
summary(mdata2)
```

```
##     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
```

```r
head(mdata2)
```

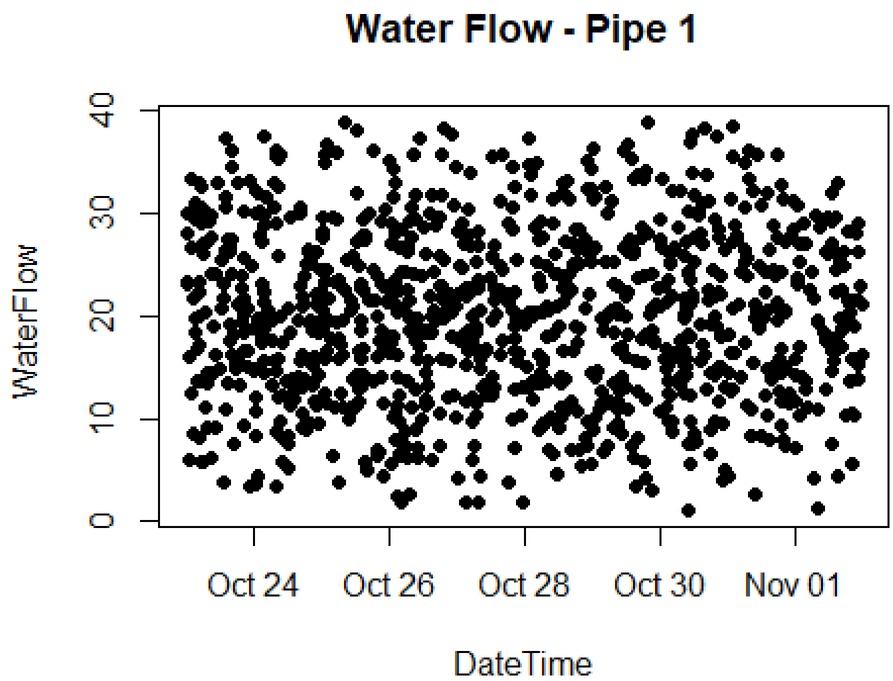
```
## # A tibble: 6 x 2
##   DateTime            WaterFlow
##   <dttm>                  <dbl>
## 1 2015-10-23 01:00:00      18.8
## 2 2015-10-23 02:00:00      43.1
## 3 2015-10-23 03:00:00      38.0
## 4 2015-10-23 04:00:00      36.1
## 5 2015-10-23 05:00:00      31.9
## 6 2015-10-23 06:00:00      28.2
```

```r
tail(mdata2)
```

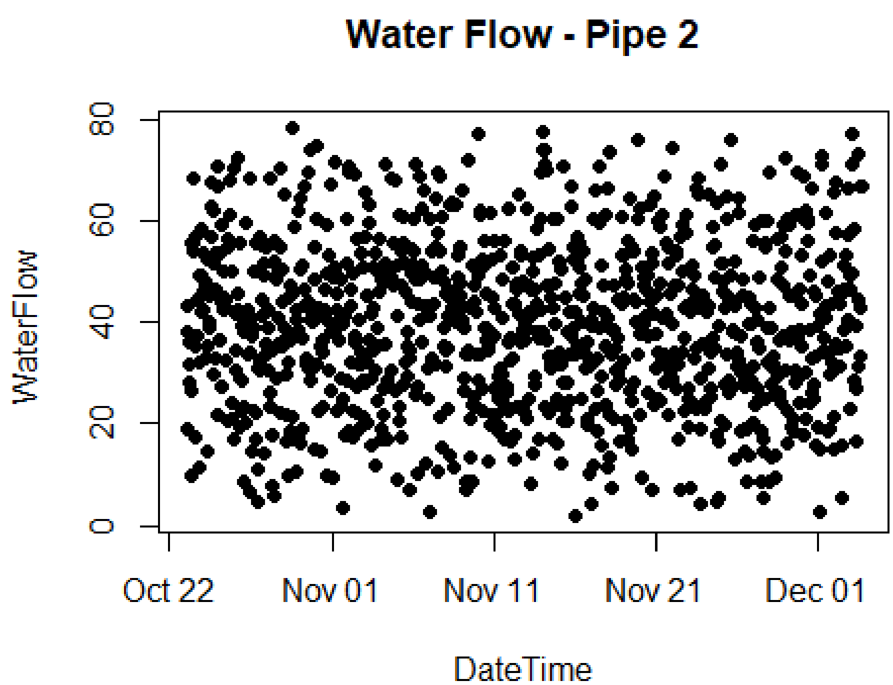
```
## # A tibble: 6 x 2
##   DateTime            WaterFlow
##   <dttm>                  <dbl>
## 1 2015-12-03 11:00:00      73.0
## 2 2015-12-03 12:00:00      31.5
## 3 2015-12-03 13:00:00      66.8
## 4 2015-12-03 14:00:00      42.9
## 5 2015-12-03 15:00:00      33.4
## 6 2015-12-03 16:00:00      66.7
```

Some basis scatter plots of our data.

```r
plot(mdata1$DateTime, mdata1$WaterFlow, main="Water Flow - Pipe 1",
   xlab= "DateTime", ylab="WaterFlow ", pch=19)
```




```r
plot(mdata2$DateTime, mdata2$WaterFlow, main="Water Flow - Pipe 2",
   xlab="DateTime", ylab="WaterFlow ", pch=19)
```



Step 4. Data Cleaning.

Let’s get the first dataset in the right format. One record per an hour.

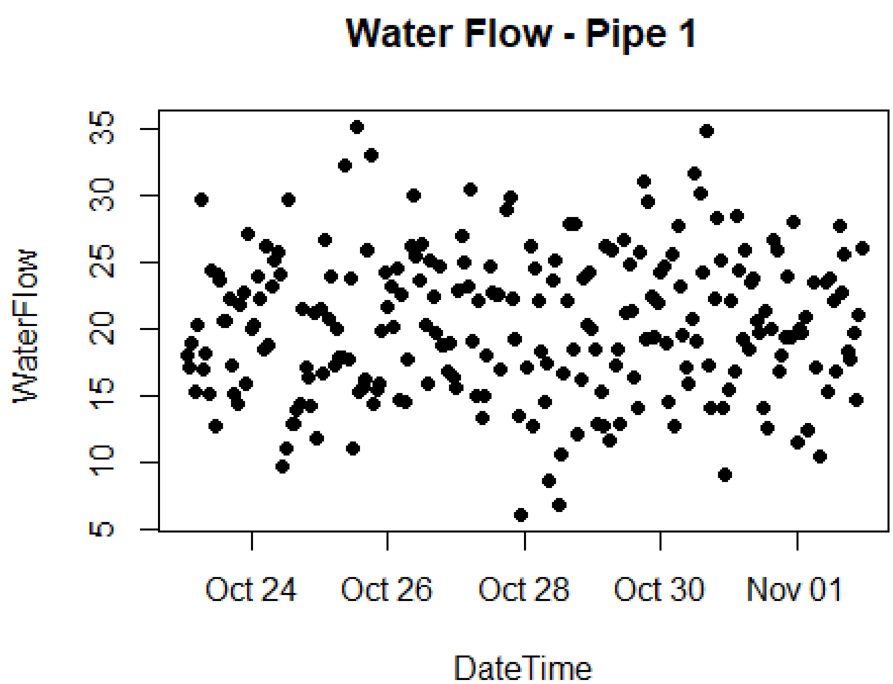
```
mdata1$WFp<-Lag(mdata1$WaterFlow,shift=1)

mdata1$DateTimep<-Lag(mdata1$DateTime)

#mydata1$myhour<-hour(mdata1$DateTime)



mdata1$mhour<-hour(mdata1$DateTime)

mdata1$mhourp<-hour(mdata1$DateTimep)

mdata1$WaterFlowN<-ifelse(mdata1$mhour!=mdata1$mhourp,(mdata1$WaterFlo
w+mdata1$WFp)/2,NA)


mdata1N<-mdata1[complete.cases(mdata1), ]



mdata1N$DateTimeN<-floor_date(mdata1N$DateTime,"hour")

mdata1N<-mdata1N[,c(8,7)]
```

Let’s see domensions, top and bottom records, and a plot of transformed data.

```
dim(mdata1N)

## [1] 235   2

head(mdata1N)

## # A tibble: 6 x 2
##   DateTimeN           WaterFlowN
##   <dttm>                   <dbl>
## 1 2015-10-23 01:00:00       18.0
## 2 2015-10-23 02:00:00       17.1
## 3 2015-10-23 03:00:00       18.9
## 4 2015-10-23 04:00:00       15.3
## 5 2015-10-23 05:00:00       20.3
## 6 2015-10-23 06:00:00       29.6

tail(mdata1N)
```

```
## # A tibble: 6 x 2
##   DateTimeN           WaterFlowN
##   <dttm>                   <dbl>
## 1 2015-11-01 18:00:00       18.3
## 2 2015-11-01 19:00:00       17.7
## 3 2015-11-01 20:00:00       19.7
## 4 2015-11-01 21:00:00       14.6
## 5 2015-11-01 22:00:00       21.0
## 6 2015-11-01 23:00:00       26.0
```

```
plot(mdata1N$DateTimeN, mdata1N$WaterFlowN, main="Water Flow - Pipe 1"
,
   xlab= "DateTime", ylab="WaterFlow ", pch=19)
```



Fixing missing data (4 records were missing), taking into account time zone and daylight saving time.

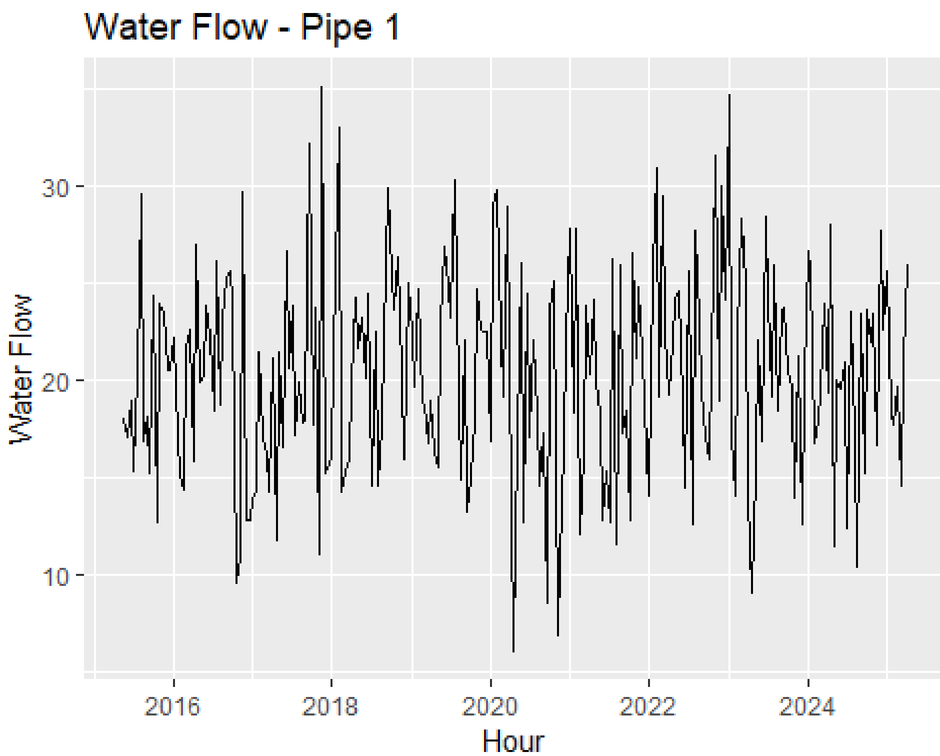
```
#4 hour difference

nr1<-data.frame(as.POSIXct("2015-10-27 17:00:00 -0400"),28.944308)
names(nr1)<-c("DateTimeN","WaterFlowN")
nr1

##             DateTimeN WaterFlowN
## 1 2015-10-27 17:00:00   28.94431
```

```
mdata1N <- rbind(mdata1N, nr1)

#4 hour difference
nr2<-data.frame(as.POSIXct("2015-11-01 01:00:00 -0400"),19.998079)
names(nr2)<-c("DateTimeN","WaterFlowN")

#4 hour difference
nr2<-data.frame(as.POSIXct("2015-10-28 00:00:00 -0400"),17.089225)
names(nr2)<-c("DateTimeN","WaterFlowN")

mdata1N <- rbind(mdata1N, nr2)
#5 hour difference - time change
nr3<-data.frame(as.POSIXct("2015-11-01 08:00:00 -0500"),23.474922)
names(nr3)<-c("DateTimeN","WaterFlowN")


mdata1N <- rbind(mdata1N, nr3)


mdata1N1 <- mdata1N[order(mdata1N$DateTimeN),]
```

Step 5. Converting data into time series.

Our first dataset only covers time period from 10/23/2019, 1AM to 11/1/2019 11PM, while the second dataset covers from 10/23/2019, 1AM to 12/3/2019, 4PM. We are required to forecast one week of data flow for both pipes. So, the correct way would be to predict one week from the earliest data set, or from 11/1/2019, 11PM. For that time period, we only need to forecast first dataset and for the second we have actual data.

```
ts1<-ts(mdata1N1$WaterFlowN,start=c(2015,10,23,1),freq=24)

autoplot(ts1) +
  ggtitle("Water Flow - Pipe 1") +
  xlab("Hour") +
  ylab("Water Flow")
```




```
#ts2<-ts(mdata2$WaterFlow,start=c(2015,10,23,1),freq=24*365)

#ts2

#autoplot(ts2) +
#  ggtitle("Water Flow - Pipe 2") +
#  xlab("Hour") +
#  ylab("Water Flow")

#mdataM=merge(x = mdata1N1, y = mdata2, by.x = "DateTimeN", by.y="Date
Time")

#dim(mdataM)

#dim(mdataM)

#mdataM$WaterFlowC<-mdataM$WaterFlowN+mdataM$WaterFlow

#mdataM<-mdataM[,c(1,4)]

#mdataM

#strftime(mdataM$DateTimeN,"%Y-%m-%d %H:%M:%S %z")

#ts3<-ts(mdataM$WaterFlowC,start=c(2015,10,23,2),freq=24)
```

```r
#autoplot(ts3) +
#  ggtitle("Water Flow - Pipe 1 and 2") +
#  xlab("Hour") +
#  ylab("Water Flow")
```

Step 6. Looking at seasonality and trend.

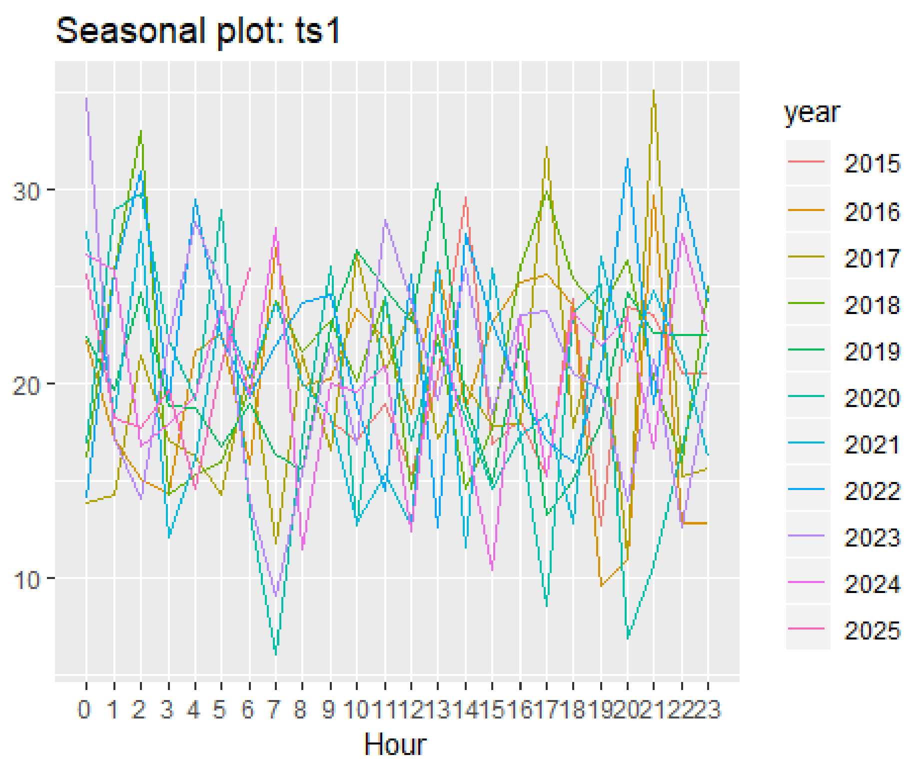
Maximum value

```r
which.max(ts1)/24
```

```
## [1] 2.541667
```

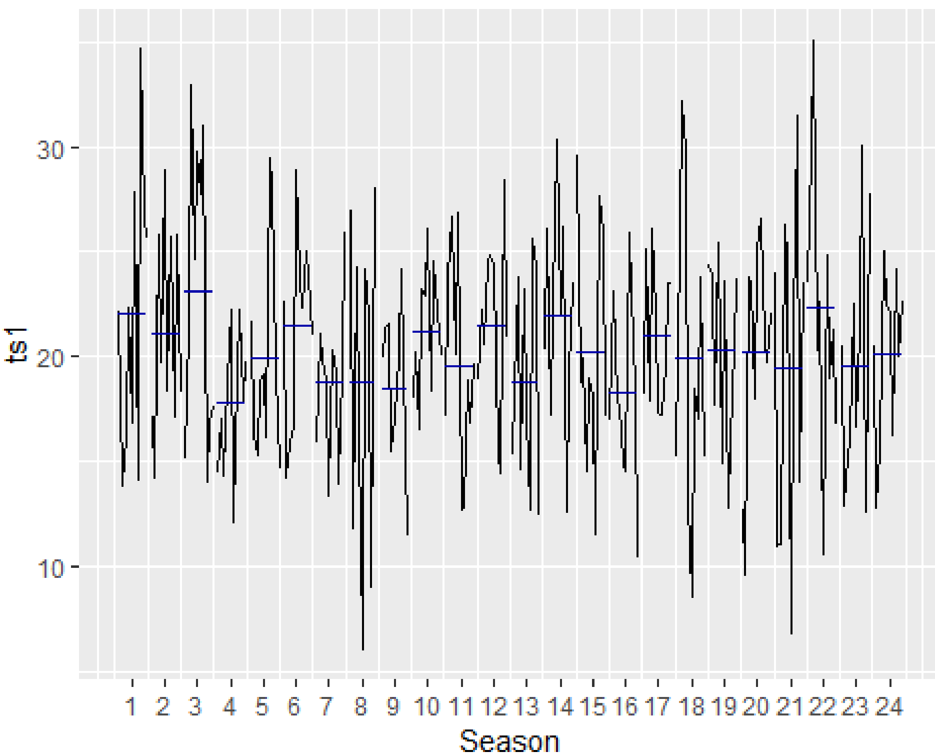
The spike in water flow was on 3 day, we can see on the graph.

```r
ggseasonplot(ts1)
```



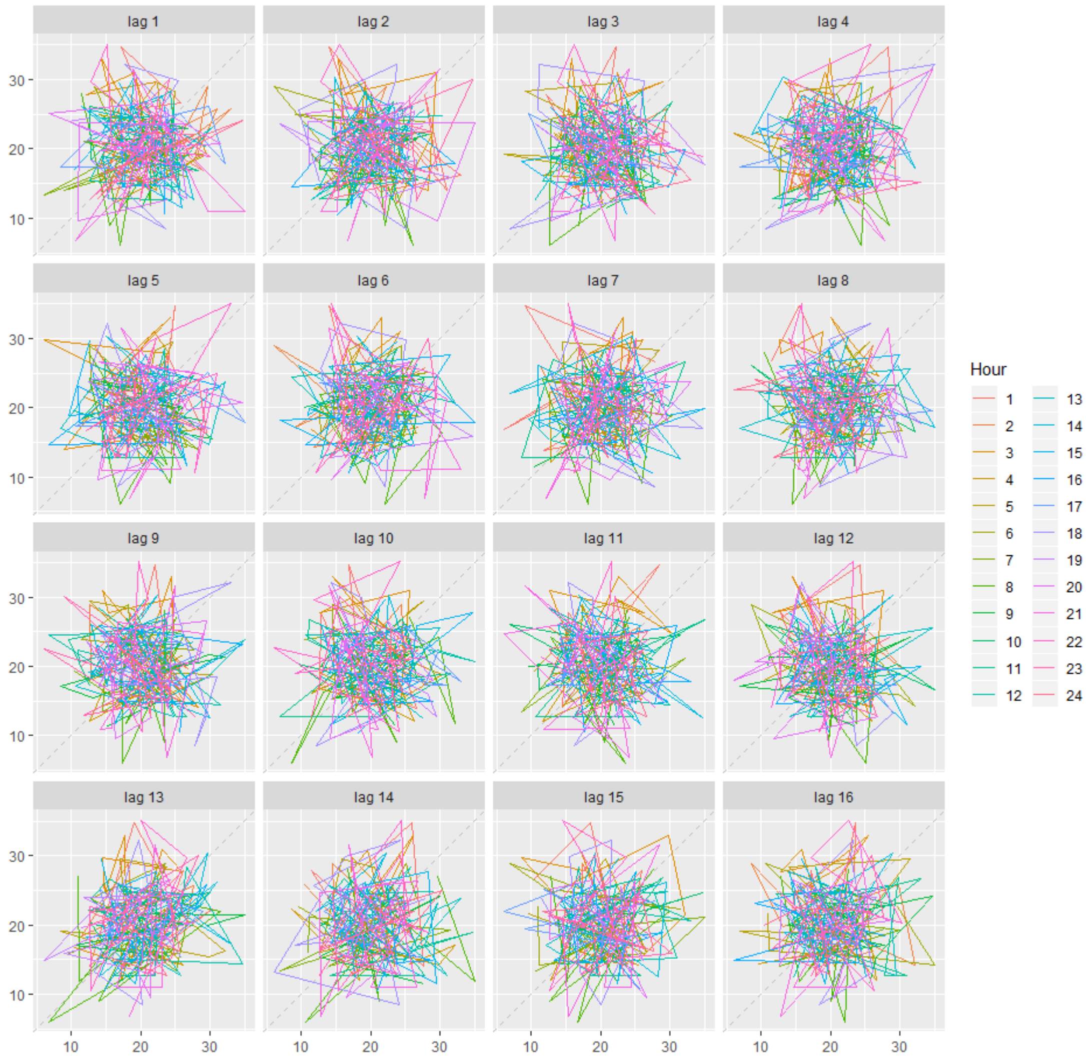
No clear patern in water use by time of day. Even though, more water seems to be used in late hours.

```r
ggsubseriesplot(ts1)
```

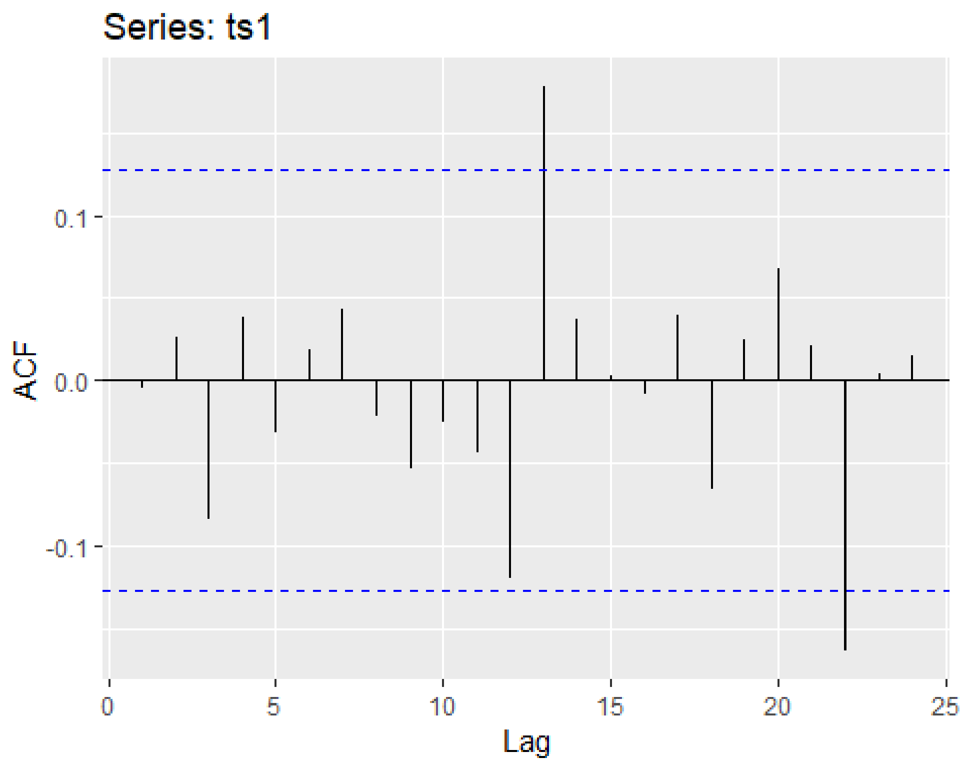


No clear picture. But the top hours were evening and night. The lowest water use was in mornings.

```
gglagplot(ts1)
```



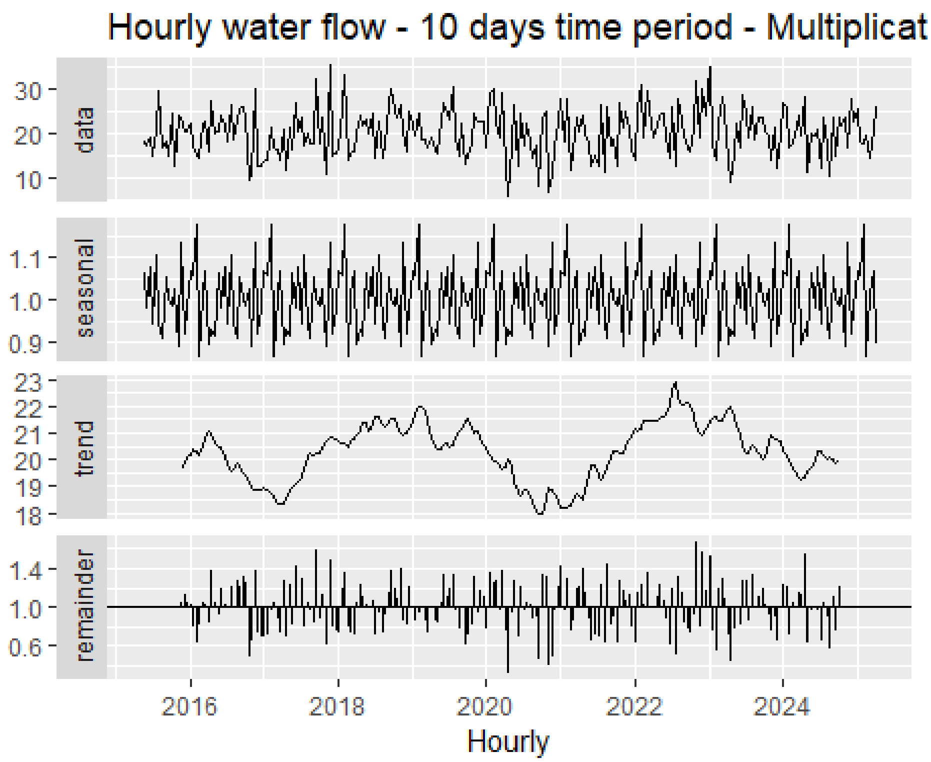

```
ggAcf(ts1, lag=24)
```



Again no clear pattern

Step 8. Applying decomposition.

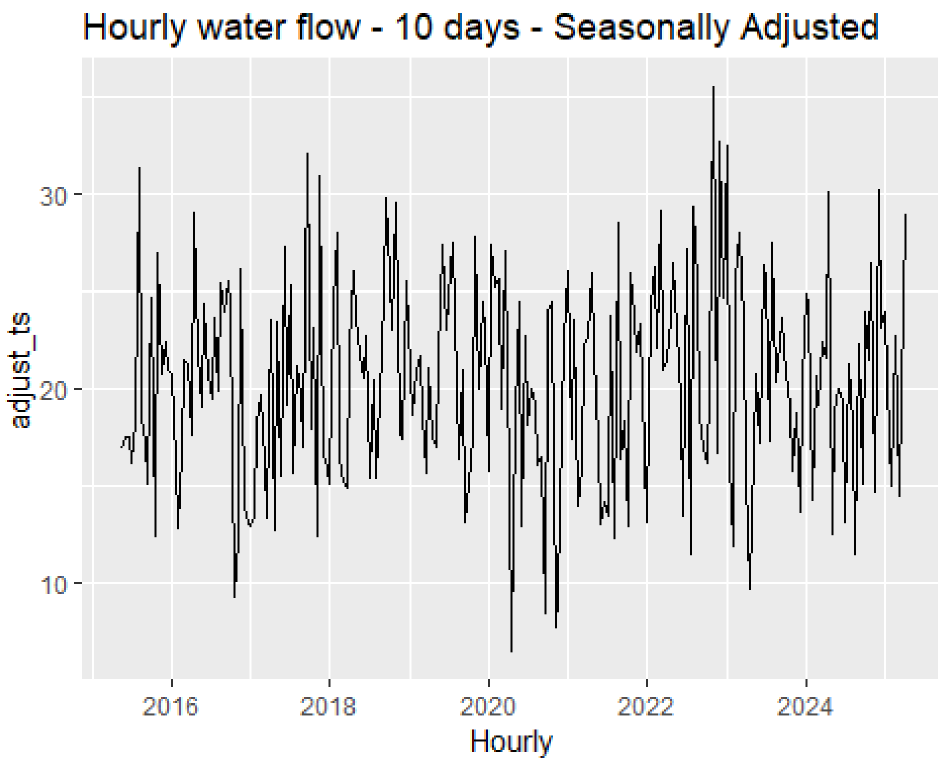
```
ts_decomp<-decompose(ts1,type="multiplicative")

autoplot(ts_decomp) +
  ggtitle("Hourly water flow - 10 days time period - Multiplicative De
composition") +
  xlab("Hourly")
```



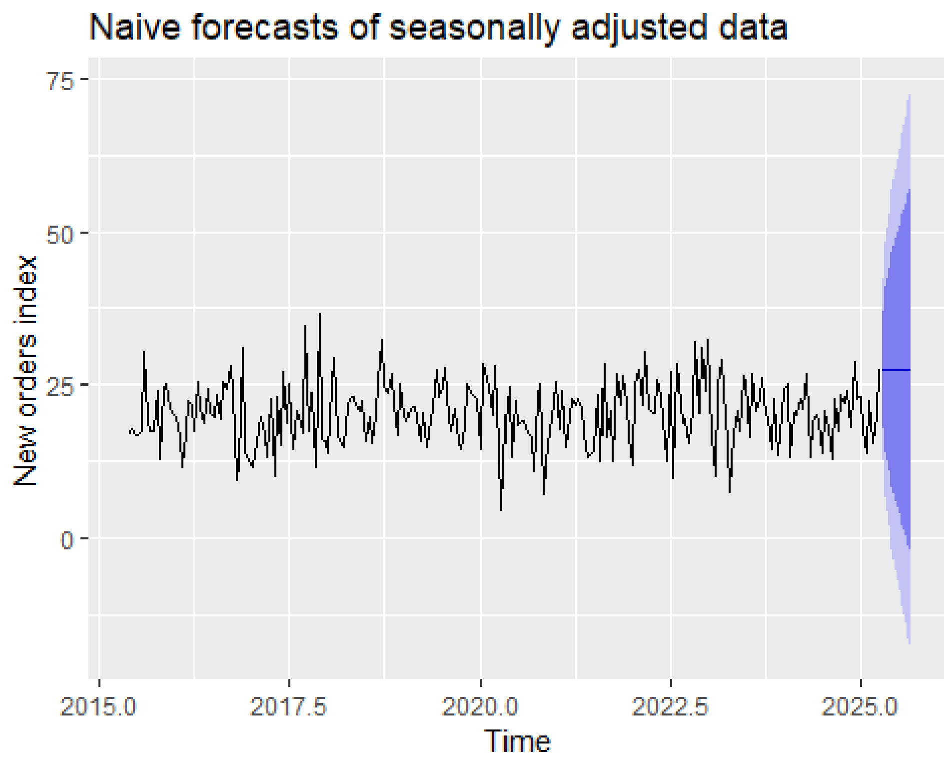
There is some type of sesonality - pattern repeats daily. But no clear trend.

Seasonaly adjusted data

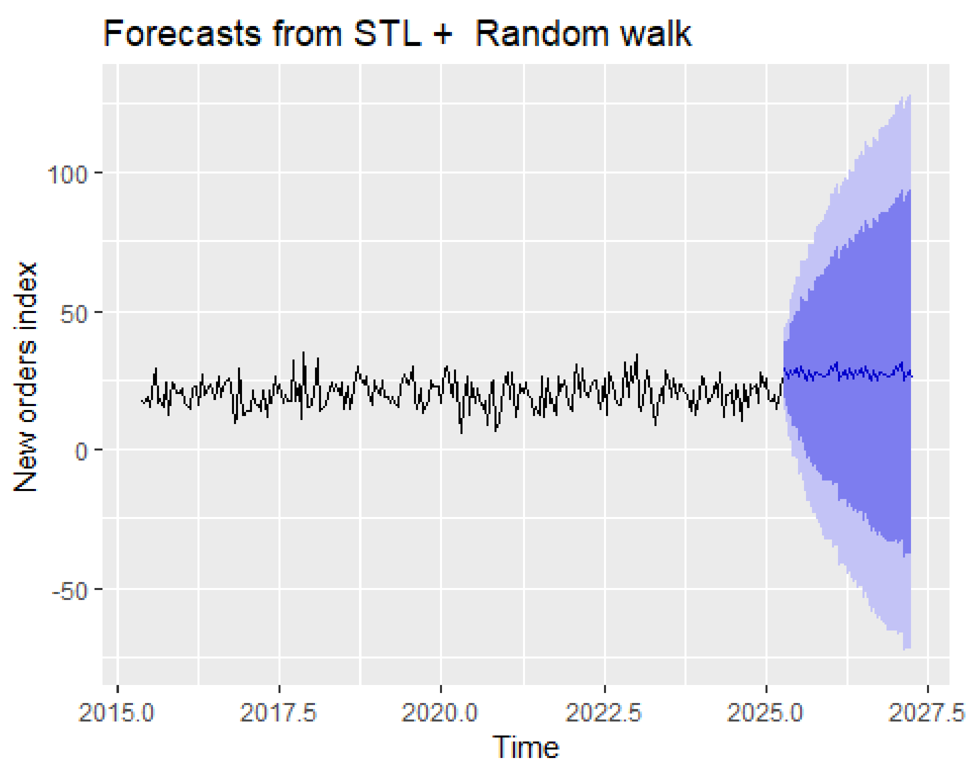
```
adjust_ts<-ts1/ts_decomp$seasonal

autoplot(adjust_ts) +
  ggtitle("Hourly water flow - 10 days - Seasonally Adjusted") +
  xlab("Hourly")
```




```
fit <- stl(ts1, t.window=13, s.window="periodic",
  robust=TRUE)

fit1<-fit %>% seasadj() %>% naive()

fit1%>%autoplot() + ylab("New orders index") +
  ggtitle("Naive forecasts of seasonally adjusted data")
```




```
fit2<-fit %>% forecast(method="naive")

fit2%>%autoplot() + ylab("New orders index")
```

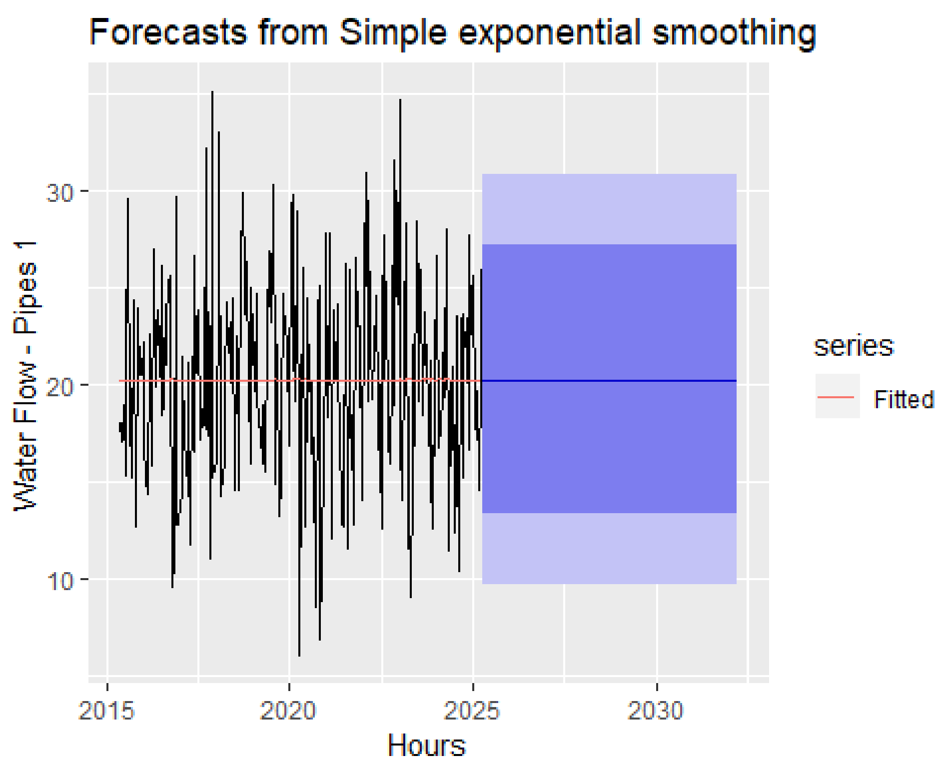


```
fcast <- stlf(ts1, method='naive')
```

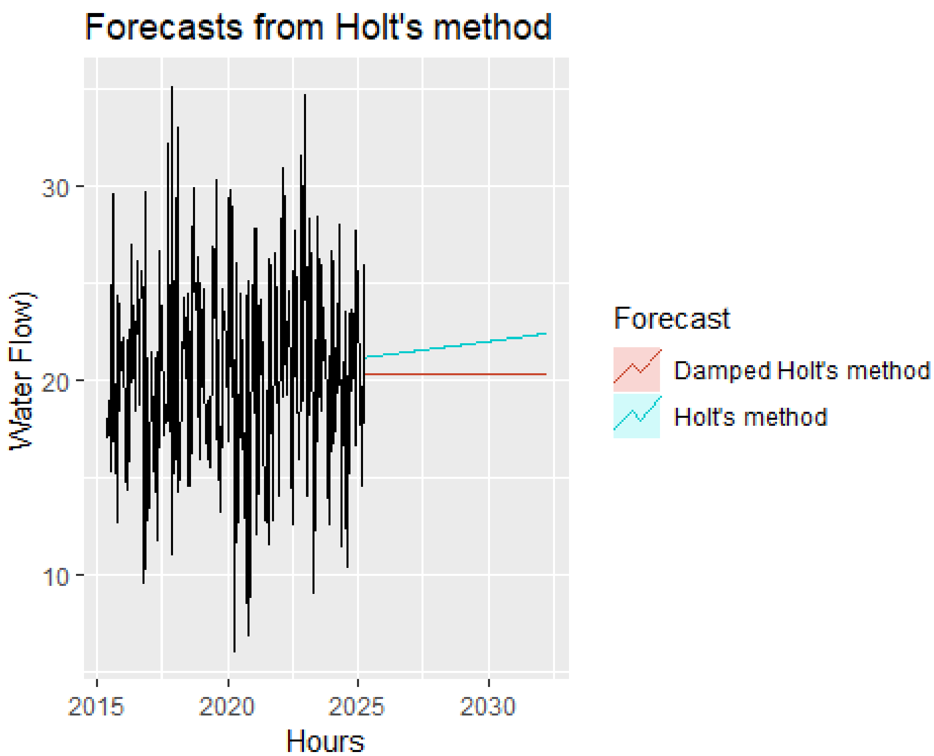
Step 9. Exponential Forecasting.

Simple exponential forecast.

```
fc <- ses(ts1, h=24*7)
# Accuracy of one-step-ahead training errors
round(accuracy(fc),2)

##               ME RMSE MAE   MPE  MAPE MASE ACF1
## Training set  0 5.37 4.4 -8.68 25.25 0.71    0

autoplot(fc) +
  autolayer(fitted(fc), series="Fitted") +
  ylab("Water Flow - Pipes 1") + xlab("Hours")
```



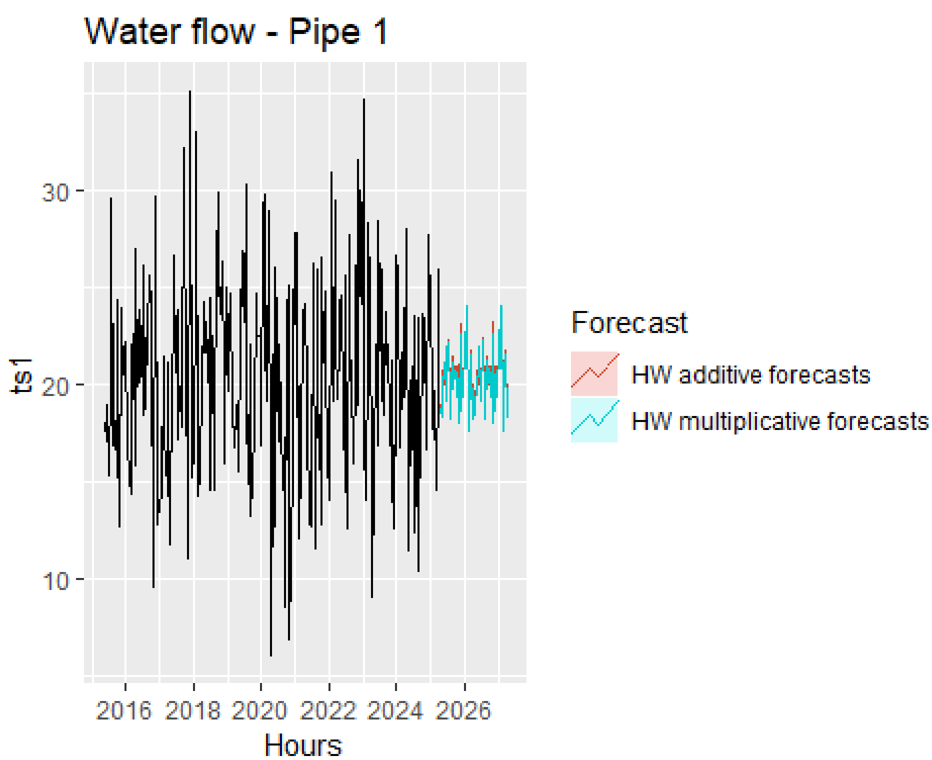

```
fc <- holt(ts1, h=24*7)

fc2 <- holt(ts1, damped=TRUE, phi = 0.9, h=24*7)
autoplot(ts1) +
  autolayer(fc, series="Holt's method", PI=FALSE) +
  autolayer(fc2, series="Damped Holt's method", PI=FALSE) +
  ggtitle("Forecasts from Holt's method") + xlab("Hours") +
  ylab("Water Flow)") +
  guides(colour=guide_legend(title="Forecast"))
```




```
e1 <- tsCV(ts1, ses, h=1)
e2 <- tsCV(ts1, holt, h=1)
e3 <- tsCV(ts1, holt, damped=TRUE, h=1)
mean(e1^2, na.rm=TRUE)

## [1] 29.51763

mean(e2^2, na.rm=TRUE)

## [1] 33.01115

mean(e3^2, na.rm=TRUE)

## [1] 31.76159

mean(abs(e1), na.rm=TRUE)

## [1] 4.435647

mean(abs(e2), na.rm=TRUE)

## [1] 4.668705

mean(abs(e3), na.rm=TRUE)

## [1] 4.561972
```

The simple exponential forecast appears to be the best.

```r
fit1 <- hw(ts1,seasonal="additive")
fit2 <- hw(ts1,seasonal="multiplicative")
autoplot(ts1) +
  autolayer(fit1, series="HW additive forecasts", PI=FALSE) +
  autolayer(fit2, series="HW multiplicative forecasts",
    PI=FALSE) +
  xlab("Hours") +
  ggtitle("Water flow - Pipe 1") +
  guides(colour=guide_legend(title="Forecast"))
```



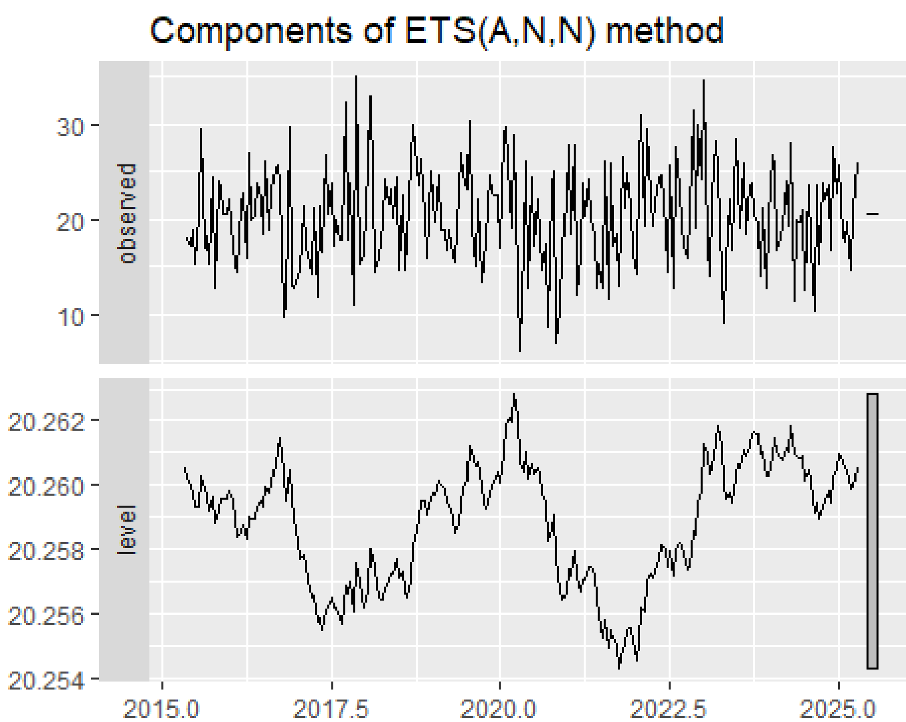
Step 9. Selecting Forecasting Method.

```r
fit<-ets(ts1)

summary(fit)

## ETS(A,N,N)
##
## Call:
##  ets(y = ts1)
##
##   Smoothing parameters:
##     alpha = 1e-04
##
##   Initial states:
##     l = 20.2605
```

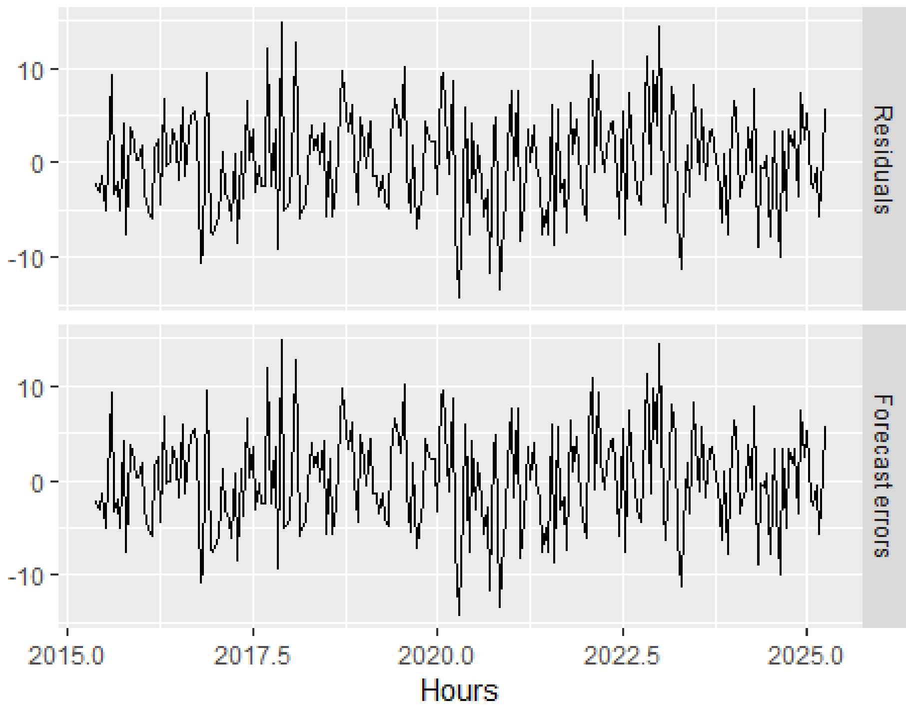
```
##
##   sigma:  5.3919
##
##      AIC     AICc      BIC
## 2108.400 2108.502 2118.817
##
## Training set error measures:
##                       ME     RMSE      MAE       MPE     MAPE
MASE
## Training set 0.001743637 5.369158 4.397655 -8.679627 25.24513 0.713
2159
##                     ACF1
## Training set -0.004125927
```

Model selected is A and N and N

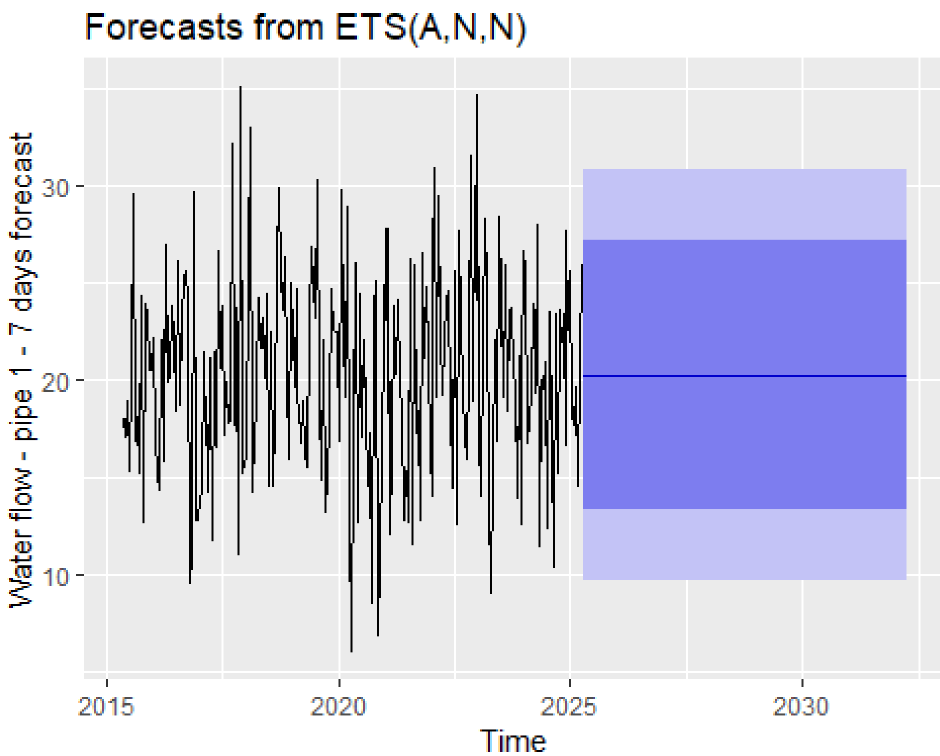
```
autoplot(fit)
```




```
cbind('Residuals' = residuals(fit),
      'Forecast errors' = residuals(fit,type='response')) %>%
  autoplot(facet=TRUE) + xlab("Hours") + ylab("")
```




```
fit1 <- fit%>%forecast(h=24*7,level=c(80,95))

fit1%>%
  autoplot() +
  ylab("Water flow - pipe 1 - 7 days forecast")
```



Step 10. Preparing the final file to be ouputed in the Excel

```
mdata1A<-mdata2[240:(239+24*7),]

fdata<-cbind(fit1,mdata1A)

#write.xlsx(fdata, "fdata.xlsx")
```