624. HW3. Fall 2019

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

9/25/2019

Table of Contents

# Project 1

## 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)

## Warning: package 'forecast' was built under R version 3.6.1

## Registered S3 methods overwritten by 'ggplot2':  
## method from   
## [.quosures rlang  
## c.quosures rlang  
## print.quosures rlang

## Registered S3 method overwritten by 'xts':  
## method from  
## as.zoo.xts zoo

## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

## Registered S3 methods overwritten by 'forecast':  
## method from   
## fitted.fracdiff fracdiff  
## residuals.fracdiff fracdiff

library(ggplot2)  
  
library(Hmisc)

## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula

##   
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':  
##   
## format.pval, units

library(lubridate)

##   
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':  
##   
## date

library(fma)

## Warning: package 'fma' was built under R version 3.6.1

library(readxl)

## Warning: package 'readxl' was built under R version 3.6.1

library(knitr)  
  
library(seasonal)

## Warning: package 'seasonal' was built under R version 3.6.1

library(openxlsx)

## Warning: package 'openxlsx' was built under R version 3.6.1

### Step 2. Read in 2 Excel files

#install.packages("readxl")  
  
# xlsx 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

str(mdata1)

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

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 |

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

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

dim(mdata2)

## [1] 1000 2

str(mdata2)

## Classes 'tbl\_df', 'tbl' and 'data.frame': 1000 obs. of 2 variables:  
## $ 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 ...

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

head(mdata2)

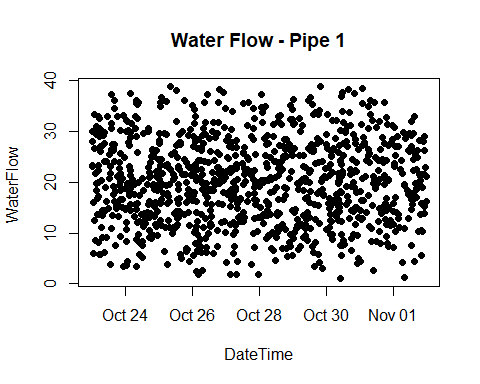
## # A tibble: 6 x 2  
## DateTime WaterFlow  
## <dttm> <dbl>  
## 1 2015-10-23 01:00:00 18.8  
## 2 2015-10-23 02:00:00 43.1  
## 3 2015-10-23 03:00:00 38.0  
## 4 2015-10-23 04:00:00 36.1  
## 5 2015-10-23 05:00:00 31.9  
## 6 2015-10-23 06:00:00 28.2

tail(mdata2)

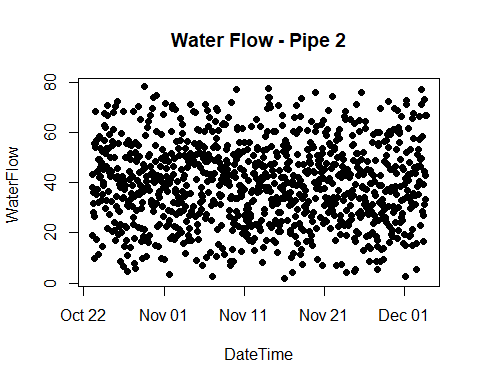
## # A tibble: 6 x 2  
## DateTime WaterFlow  
## <dttm> <dbl>  
## 1 2015-12-03 11:00:00 73.0  
## 2 2015-12-03 12:00:00 31.5  
## 3 2015-12-03 13:00:00 66.8  
## 4 2015-12-03 14:00:00 42.9  
## 5 2015-12-03 15:00:00 33.4  
## 6 2015-12-03 16:00:00 66.7

Some basis scatter plots of our data.

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



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$mhourp,(mdata1$WaterFlow+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

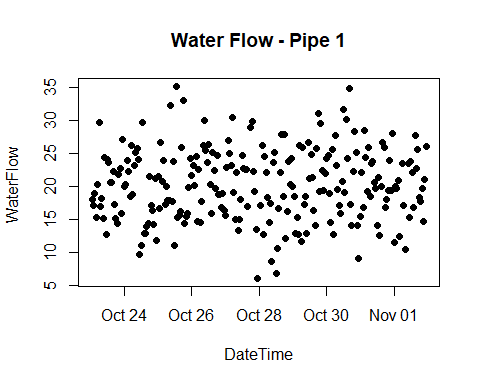
head(mdata1N)

## # 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

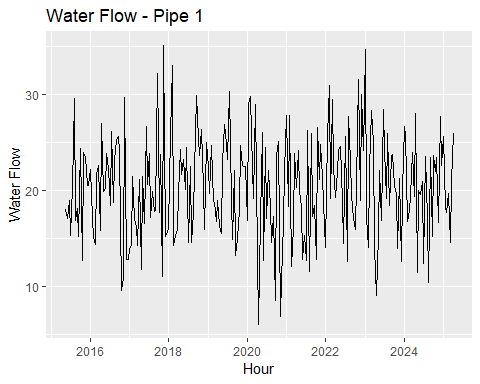
mdata1N <- rbind(mdata1N, nr1)  
  
#4 hour difference  
nr2<-data.frame(as.POSIXct("2015-11-01 01:00:00 -0400"),19.998079)  
names(nr2)<-c("DateTimeN","WaterFlowN")  
  
#4 hour difference  
nr2<-data.frame(as.POSIXct("2015-10-28 00:00:00 -0400"),17.089225)  
names(nr2)<-c("DateTimeN","WaterFlowN")  
  
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="DateTime")  
  
#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)  
  
#autoplot(ts3) +  
# ggtitle("Water Flow - Pipe 1 and 2") +  
# xlab("Hour") +  
# ylab("Water Flow")

### Step 6. Looking at seasonality and trend.

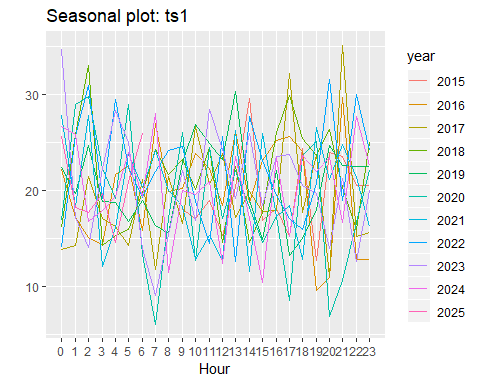
Maximum value

which.max(ts1)/24

## [1] 2.541667

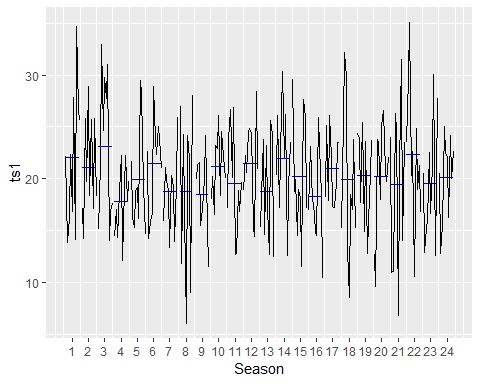
The spike in water flow was on 3 day, we can see on the graph.

ggseasonplot(ts1)



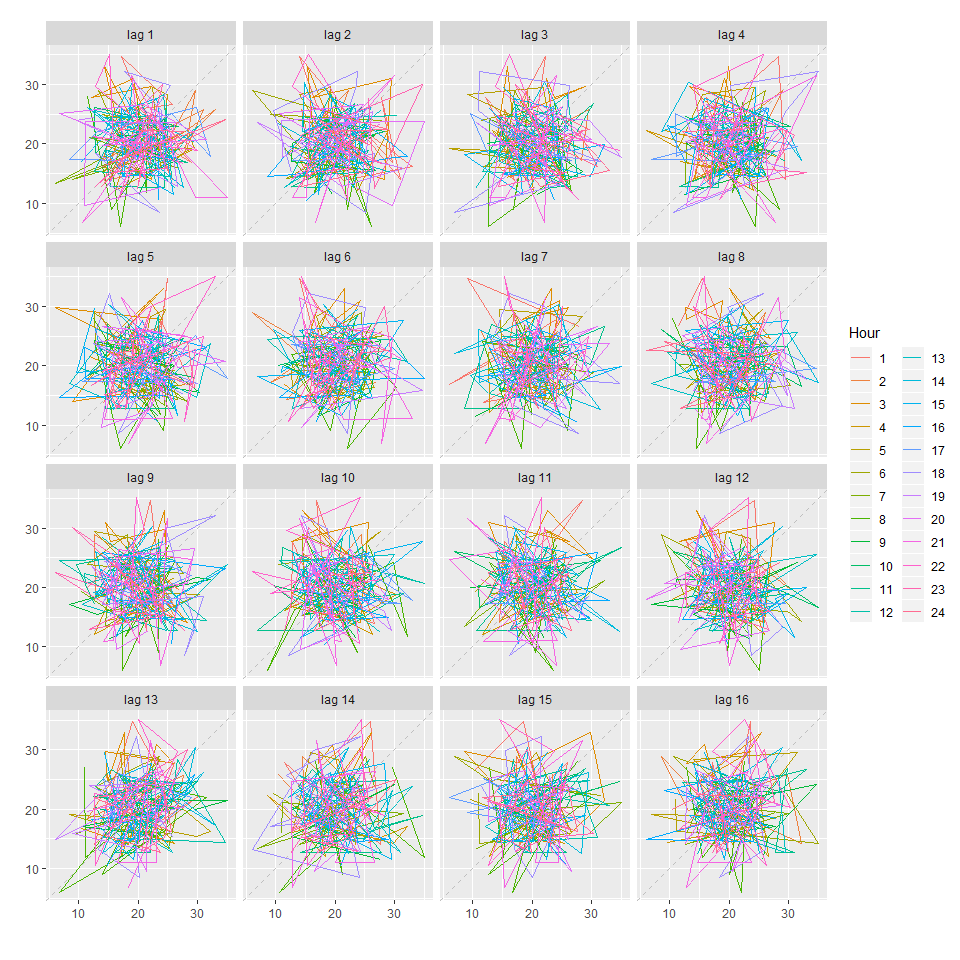
No clear patern in water use by time of day. Even though, more water seems to be used in late hours.

ggsubseriesplot(ts1)

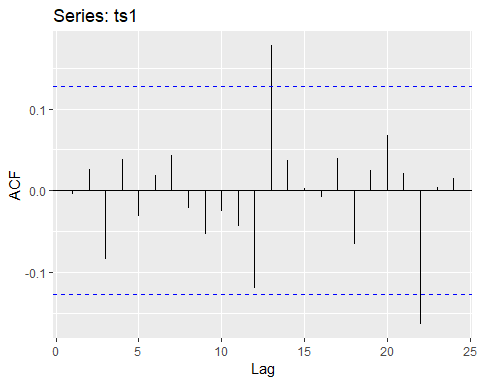


No clear picture. But the top hours were evening and night. The lowest water use was in mornings.

gglagplot(ts1)



ggAcf(ts1, lag=24)

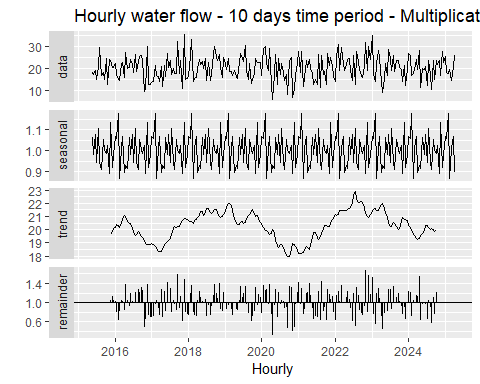


Again no clear pattern

### Step 8. Applying decomposition.

ts\_decomp<-decompose(ts1,type="multiplicative")

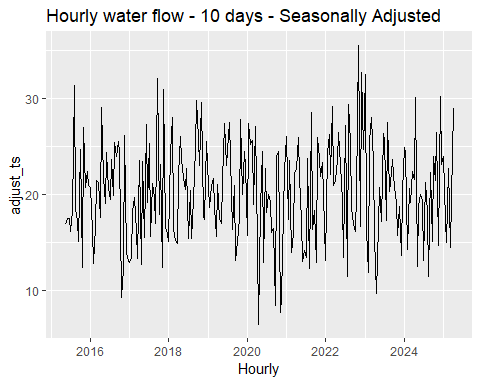
autoplot(ts\_decomp) +  
 ggtitle("Hourly water flow - 10 days time period - Multiplicative Decomposition") +  
 xlab("Hourly")



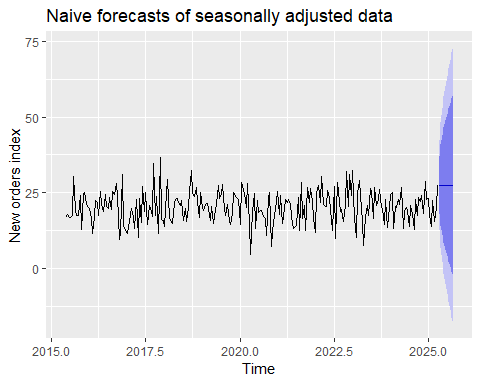
There is some type of sesonality - pattern repeats daily. But no clear trend.

Seasonaly adjusted data

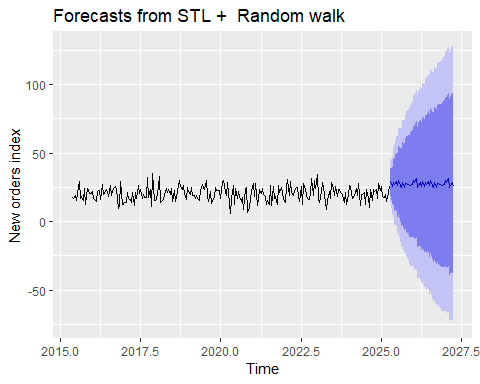
adjust\_ts<-ts1/ts\_decomp$seasonal  
  
autoplot(adjust\_ts) +  
 ggtitle("Hourly water flow - 10 days - Seasonally Adjusted") +  
 xlab("Hourly")



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')

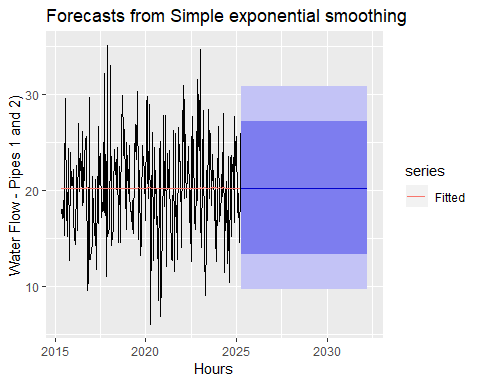
### Step 9. Exponential Forecasting.

Simple exponential forecast.

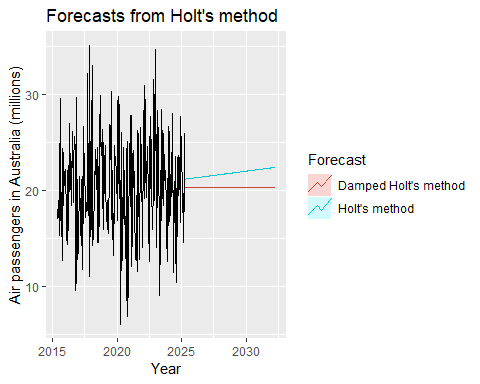
fc <- ses(ts1, h=24\*7)  
# Accuracy of one-step-ahead training errors  
round(accuracy(fc),2)

## 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 and 2)") + 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("Year") +  
 ylab("Air passengers in Australia (millions)") +  
 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)

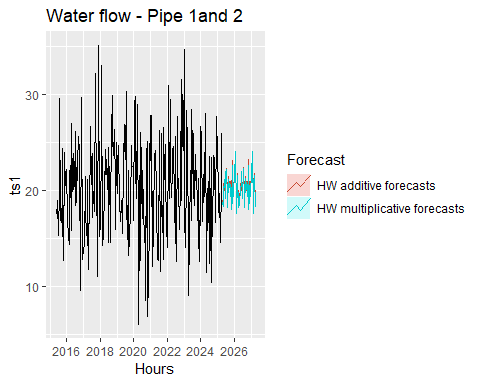
## [1] 4.668705

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

## [1] 4.561972

The simple exponential forecast appears to be the best.

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 1and 2") +  
 guides(colour=guide\_legend(title="Forecast"))



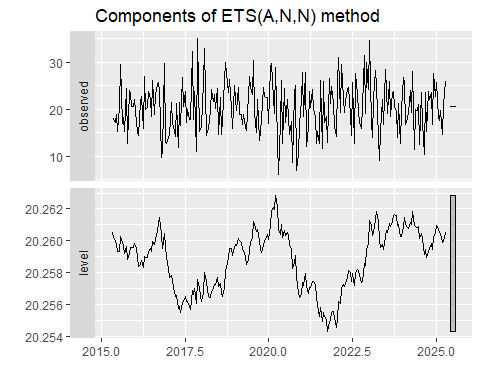
### Step 9. Selecting Forecasting Method.

fit<-ets(ts1)  
  
summary(fit)

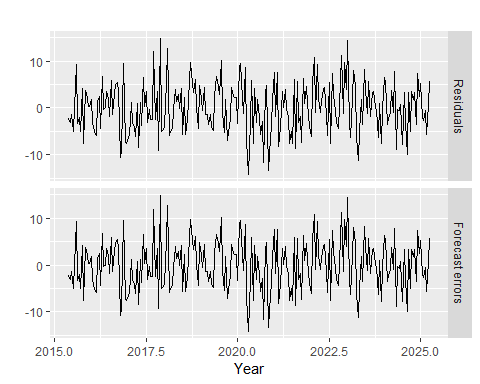
## ETS(A,N,N)   
##   
## Call:  
## ets(y = ts1)   
##   
## Smoothing parameters:  
## alpha = 1e-04   
##   
## Initial states:  
## l = 20.2605   
##   
## sigma: 5.3919  
##   
## AIC AICc BIC   
## 2108.400 2108.502 2118.817   
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set 0.001743637 5.369158 4.397655 -8.679627 25.24513 0.7132159  
## ACF1  
## Training set -0.004125927

Model selected is A and N and N

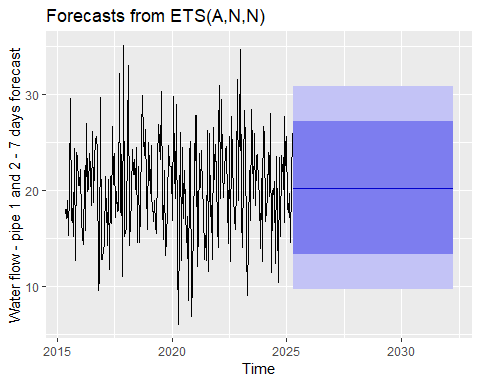
autoplot(fit)



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



fit1 <- fit%>%forecast(h=24\*7,level=c(80,95))   
  
fit1%>%  
 autoplot() +  
 ylab("Water flow - pipe 1 and 2 - 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)  
  
#install.packages("openxlsx")  
  
  
  
#write.xlsx(fdata, "fdata.xlsx")