Wrangled_Final_TSA_Project

Narissa Jimenez-Petchumrus, Kristen Pulley, Soman Ul-Haq

4/12/2022

##Setup

##Import All of the Data

```
#Import French Science Chilled Water Data
chilled_water <- read_xlsx(path="./CHW Interval Data.xlsx",col_names=TRUE)
```

```
## New names:
## * `` -> ...9
## * `` -> ...10
```

```
chilled_water #need to chop off some columns

#Import French Science Steam Data
steam_data <- read_xlsx(path="./Steam Interval Data.xlsx",col_names=TRUE, sheet="Steam_all_dates")
```

```
## New names:
## * `` -> ...9
## * `` -> ...10
## * `` -> ...11
## * `` -> ...12
## * `` -> ...13
```

```
steam_data #need to chop off some columns

#Import Temperature Data
Temp_Data <- read_xlsx(path='./Temp_Data.xlsx',col_names=TRUE, skip=1)
Temp_Data

#Delete Bottom Row of Temp Data to match steam and chw
tail(Temp_Data,5)
Temp_Data<-head(Temp_Data,-1)
tail(Temp_Data,5)
```

##Clean & Wrangle Data

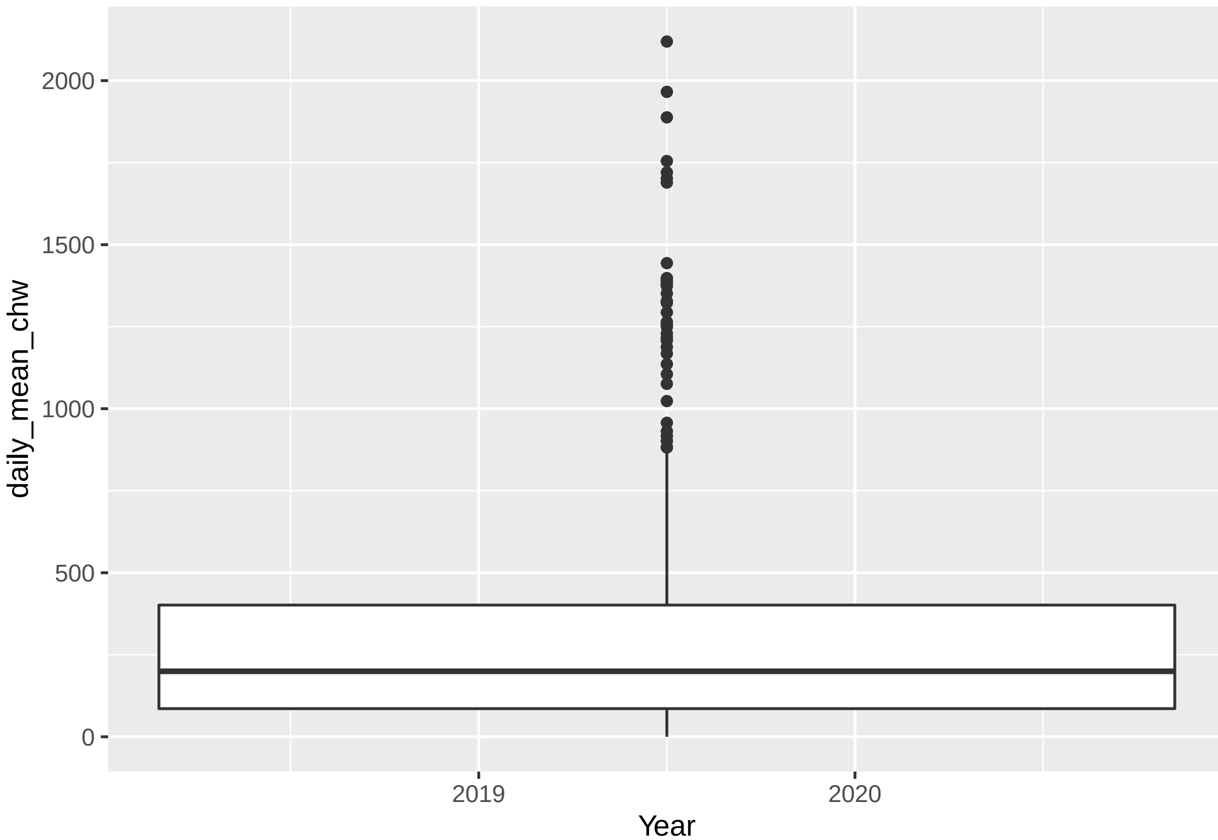
```
#Clean Chilled Water Data
chilled_water2 <- chilled_water[, c(1,7)] %>% as.data.frame()
chilled_water2

sapply(chilled_water2, class) #check data type
names(chilled_water2) <- c('Date','ton_hrs_cummulative') #change column names
head(chilled_water2)
sapply(chilled_water2,class) #check data type again
```

```r
chilled_water3 <- chilled_water2 %>%
   mutate( Day = day(Date)) %>%
   mutate( Year = year(Date)) %>%
   mutate( Month = month(Date)) %>%
   mutate(date = date(Date))

 chw_daily <- aggregate(chilled_water3$ton_hrs_cummulative, list(chilled_water3$date), mean)

 names(chw_daily) <- c('Date','avg_chw')


 chw_daily2 <- chw_daily %>%
   mutate( Day = day(Date)) %>%
   mutate( Year = year(Date)) %>%
   mutate( Month = month(Date))


chw_daily3 <- chw_daily2 %>%
   filter( !is.na(avg_chw)) %>%
   group_by(Year,Month,Day) %>%
   summarise( daily_mean_chw = mean(avg_chw))
```

```
## `summarise()` has grouped output by 'Year', 'Month'. You can override using the
## `.groups` argument.
```

```r
chw_daily3

#Boxplot
ggplot(chw_daily3, aes(x=Year, y=daily_mean_chw)) +
              geom_boxplot() #there are definitely outliers
```

```
## Warning: Continuous x aesthetic -- did you forget aes(group=...)?
```




```
# missing values detection
sum(is.na(chw_daily3$daily_mean_chw)) #no NAs

#cleaning up bottom zero for chw
tail(chw_daily3,5)
chw_daily3<-head(chw_daily3,-1)

tail(chw_daily3,5)
```

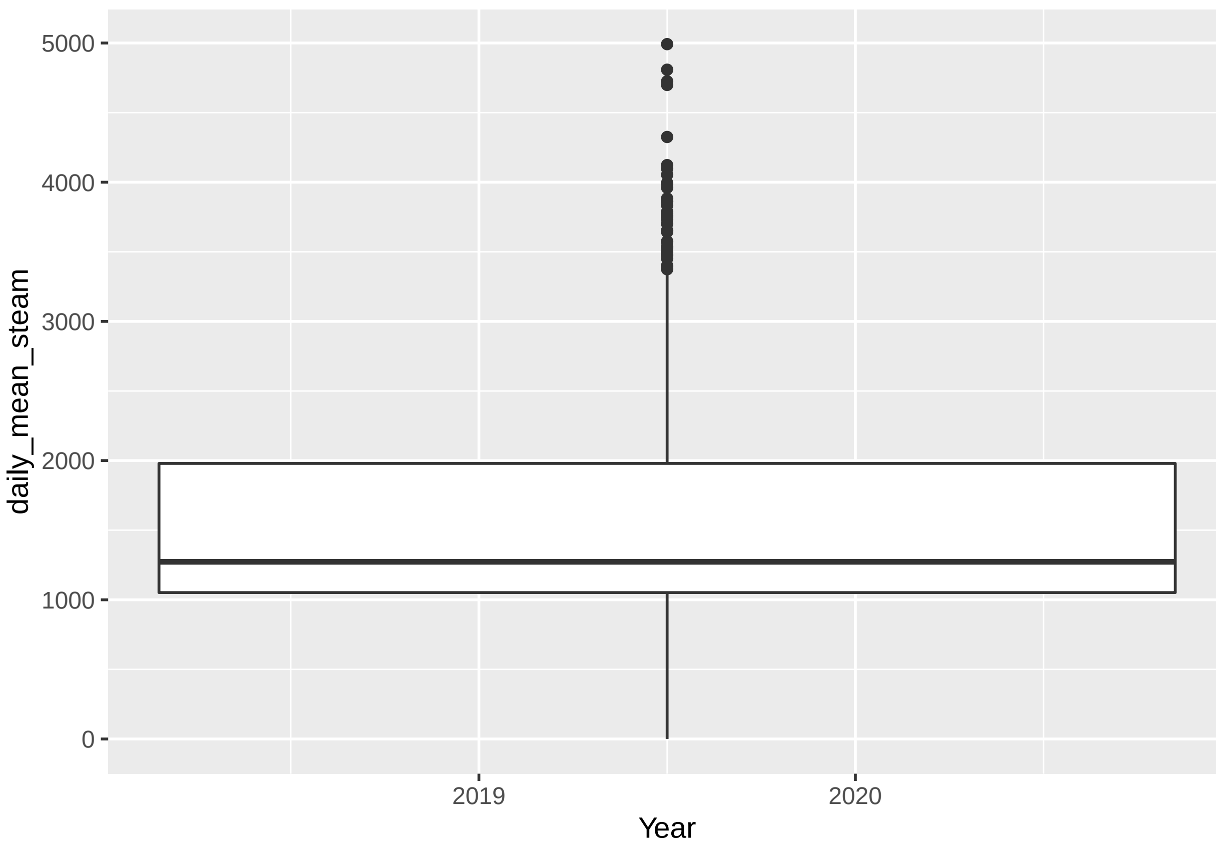
```
#Cleaning Steam Data
steam_data <- steam_data[,c(1,7)]

steam_data <- steam_data %>%
  mutate(date = date(Date))

names(steam_data) <- c('Date','Steam','date')

steam_daily <- aggregate(steam_data$Steam, list(steam_data$date), mean)

names(steam_daily) <- c('Date','avg_steam')


steam_daily2 <- steam_daily %>%
  mutate( Day = day(Date)) %>%
  mutate( Year = year(Date)) %>%
  mutate( Month = month(Date))
```

```
steam_daily3 <- steam_daily2 %>%
  filter( !is.na(avg_steam)) %>%
  group_by(Year,Month,Day) %>%
  summarise( daily_mean_steam = mean(avg_steam))
```

```
## `summarise()` has grouped output by 'Year', 'Month'. You can override using the
## `.groups` argument.
```

```
steam_daily3


#boxplot
ggplot(steam_daily3, aes(x=Year, y=daily_mean_steam)) +
             geom_boxplot() #there are definitely outliers
```

```
## Warning: Continuous x aesthetic -- did you forget aes(group=...)?
```

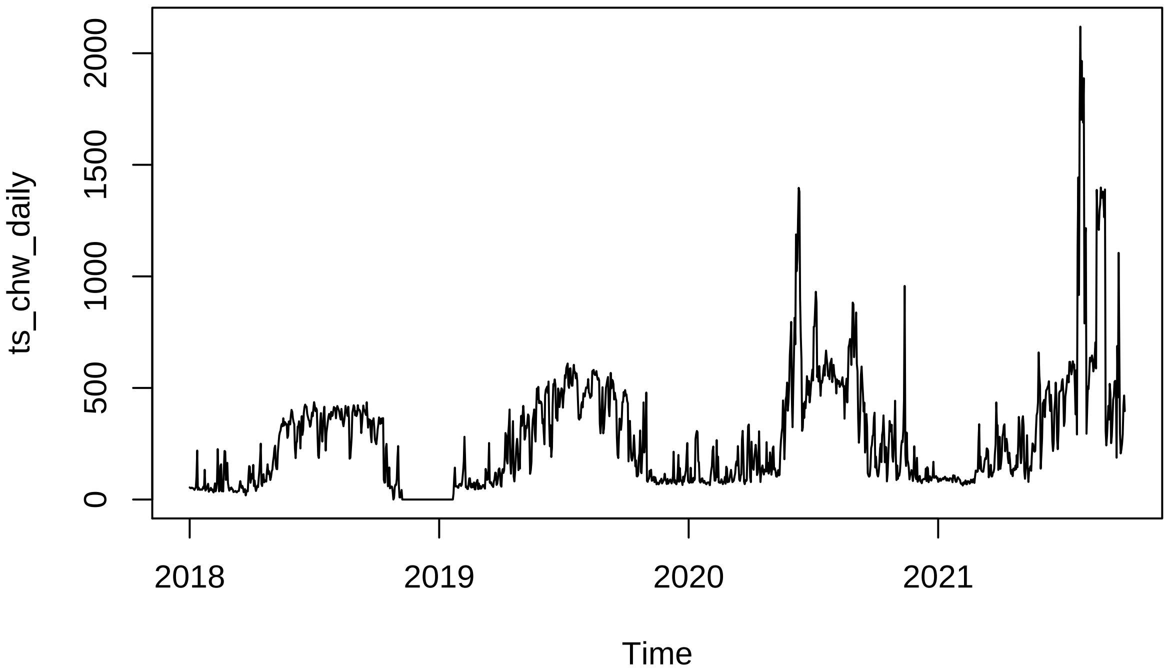



```
#note about outliers, didn't delete as didn't want uneven TS objects, didn't impute either as outliers

# missing values detection
sum(is.na(steam_daily3$daily_mean_steam)) #no NAs



#cleaning up bottom zero for steam
tail(steam_daily3,5)
steam_daily3<-head(steam_daily3,-1)
```

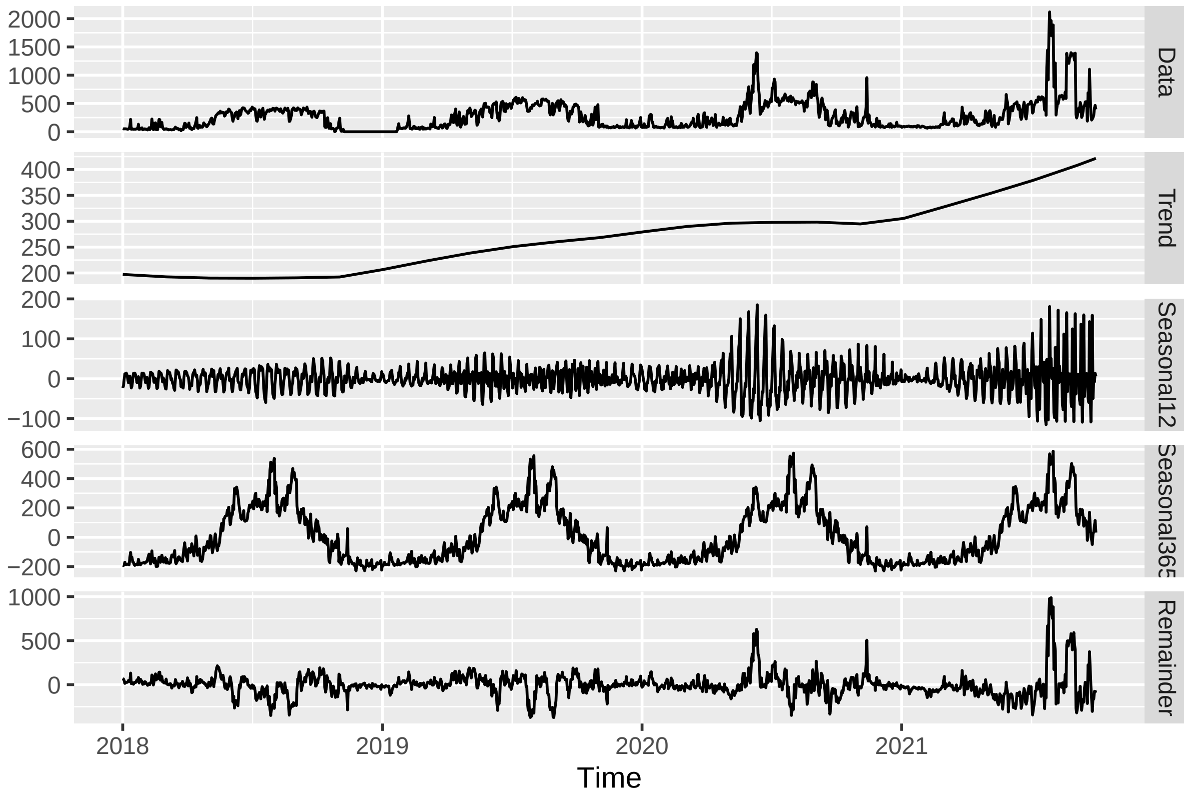
```r
tail(steam_daily3,5)
```

##Create time series object

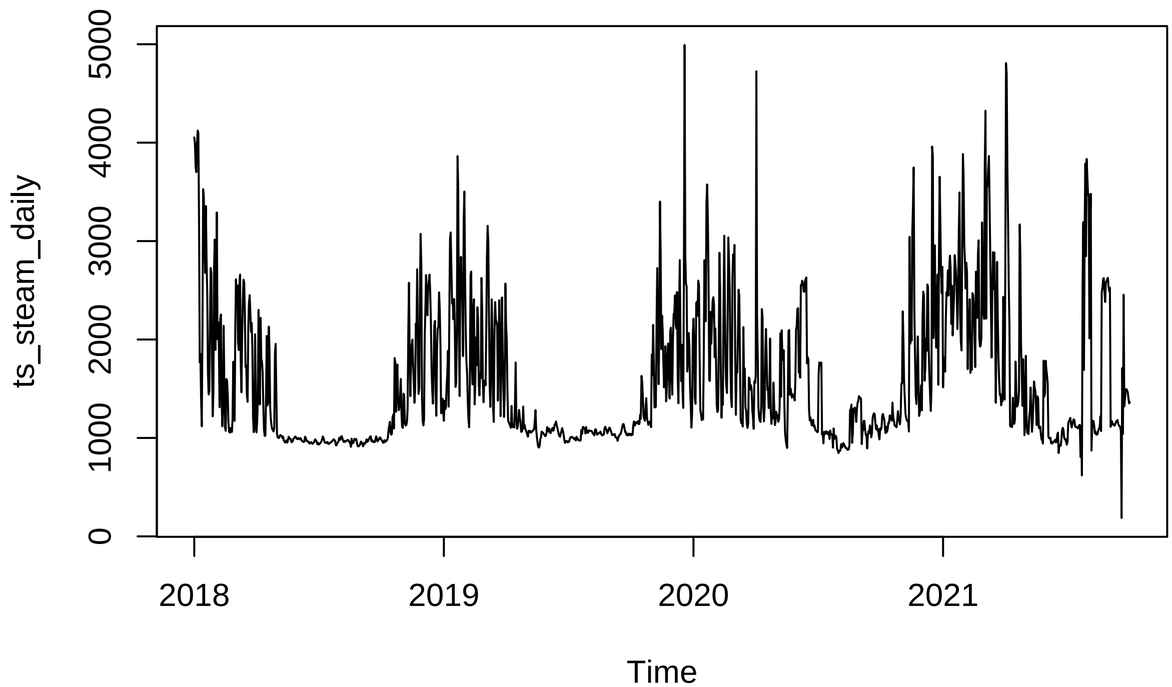
```r
#Plotting Chilled Water Consumption time series
ts_chw_daily <- msts(chw_daily3$daily_mean_chw,
                     seasonal.periods =c(12,365),
                     start=c(2018))

plot(ts_chw_daily)
```



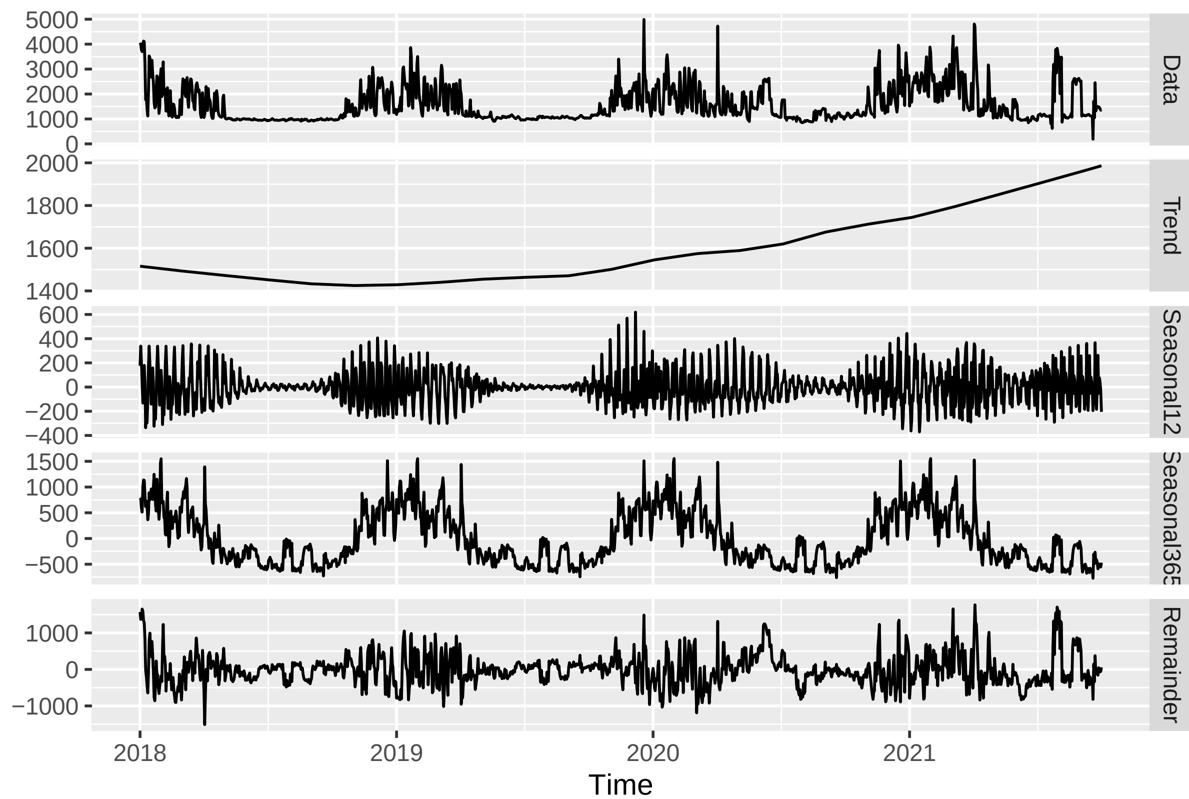

```r
ts_chw_daily %>% mstl() %>%
  autoplot()
```




```r
#Plotting Steam consumption time series and decomposing

ts_steam_daily <- msts(steam_daily3$daily_mean_steam,
                      seasonal.periods =c(12,365),
                      start=c(2018))

ts_steam_daily

plot(ts_steam_daily)
```

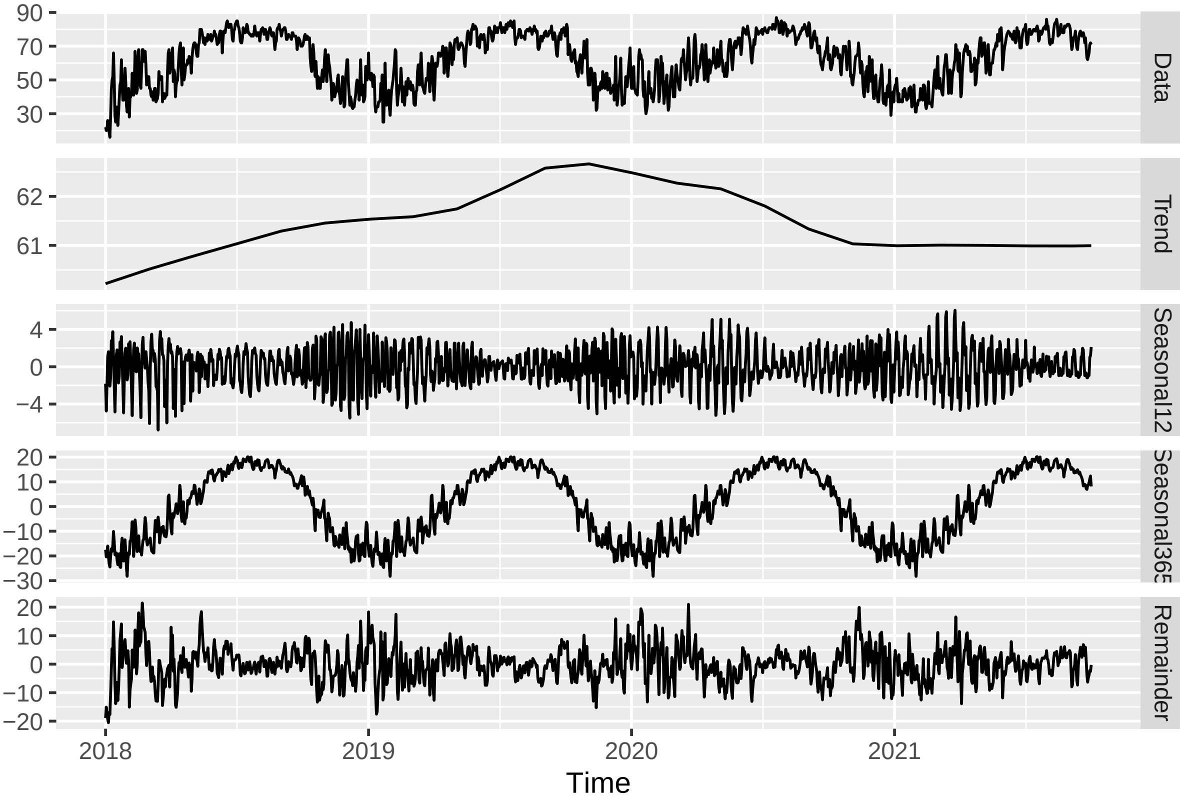



```r
ts_steam_daily %>% mstl() %>%
  autoplot()
```




```r
#Making temperature data time series and decomposing

ts_temp_daily <- msts(Temp_Data$TAVG,
                      seasonal.periods =c(12,365),
```

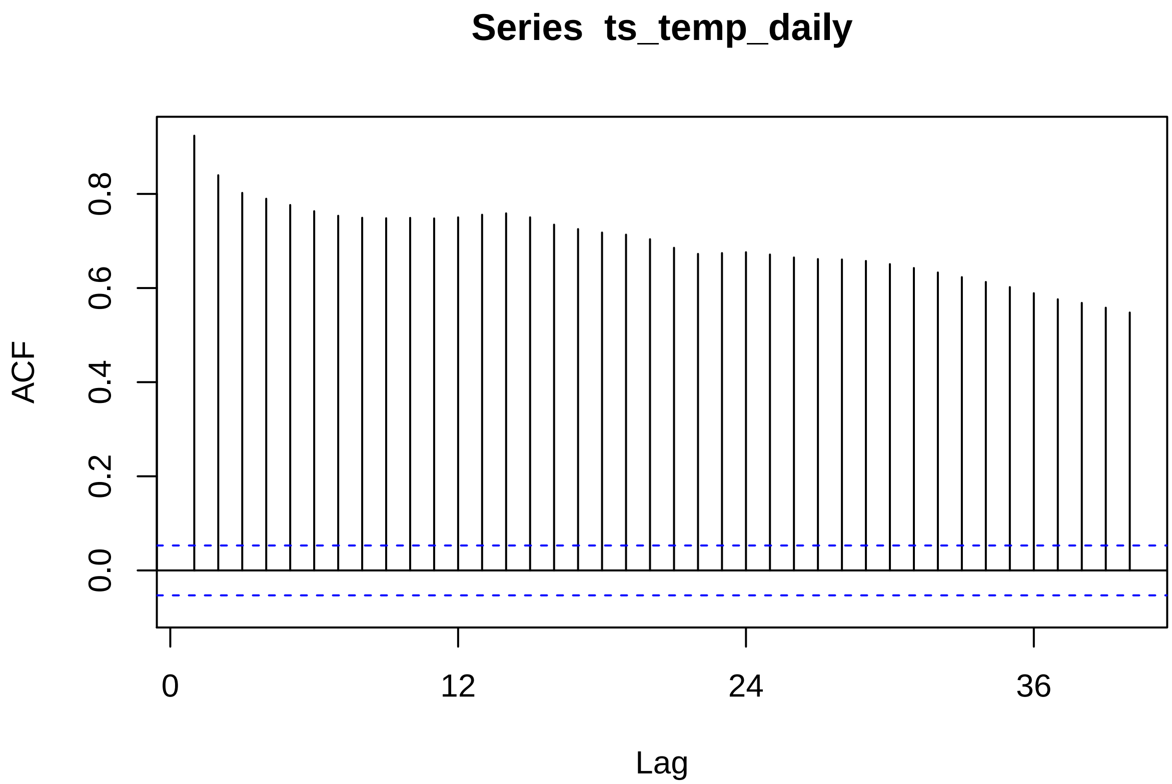
```
                    start=c(2018))

ts_temp_daily %>% mstl() %>%
  autoplot()
```



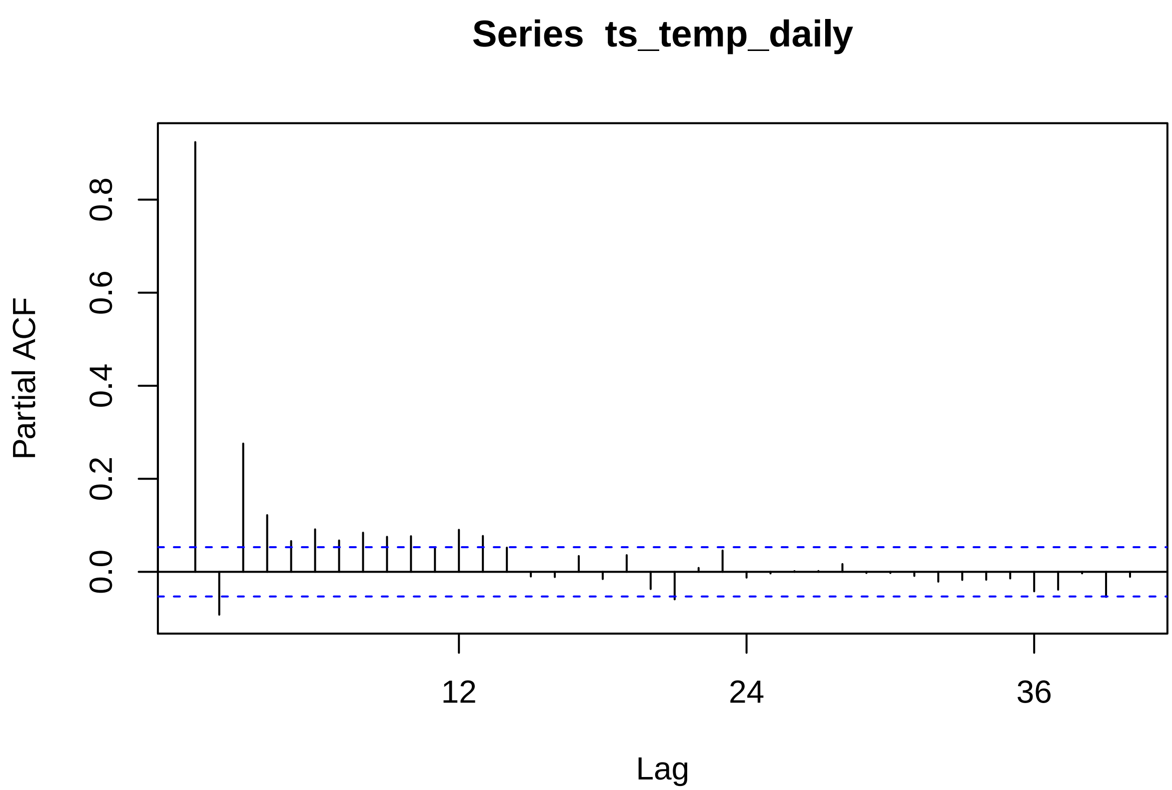

```
summary(ts_temp_daily)
```

##Plot ACF and PACF

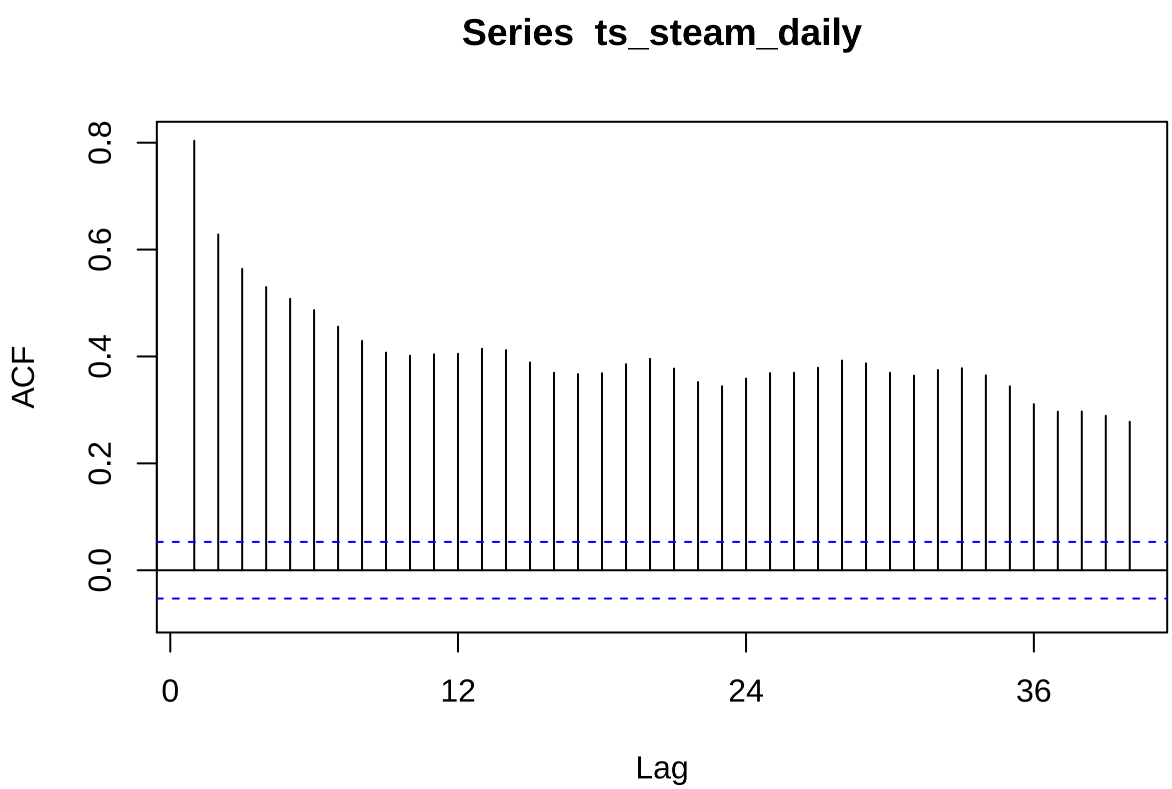
```
#Acf and pacf for temperature series
Acf(ts_temp_daily, lag.max = 40)
```



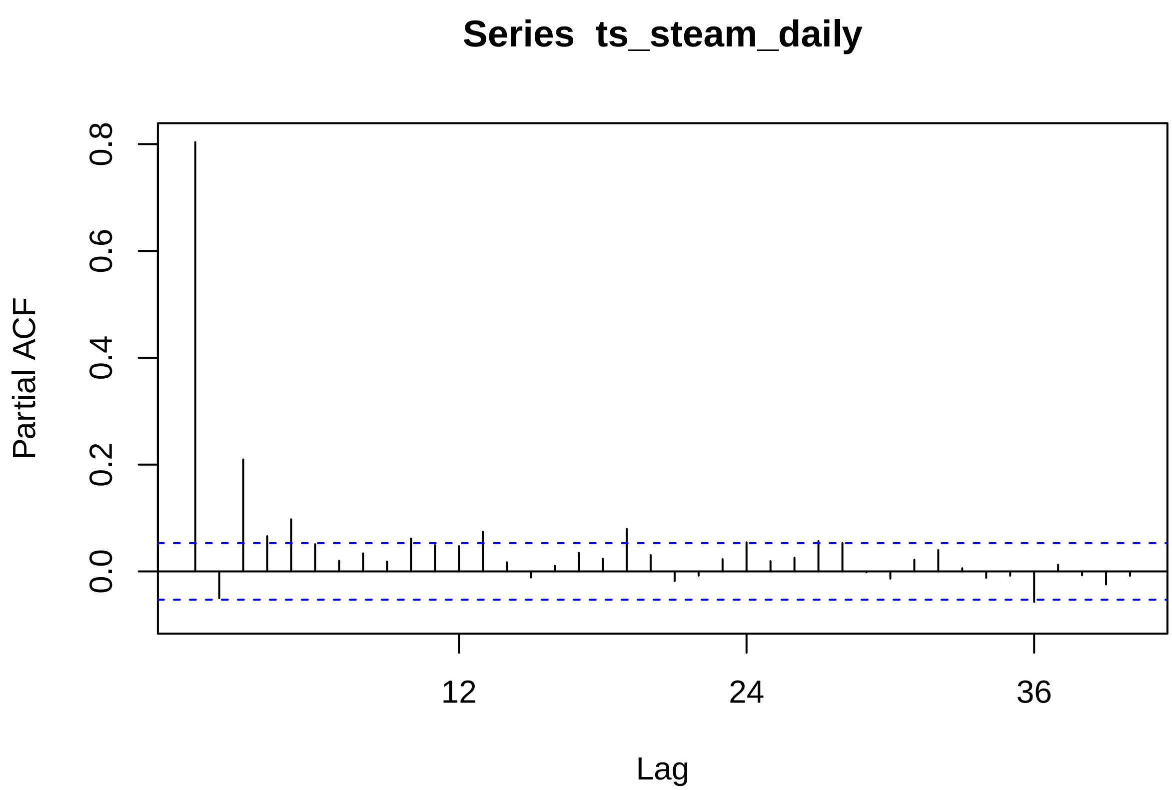

```r
Pacf(ts_temp_daily, lag.max = 40)
```



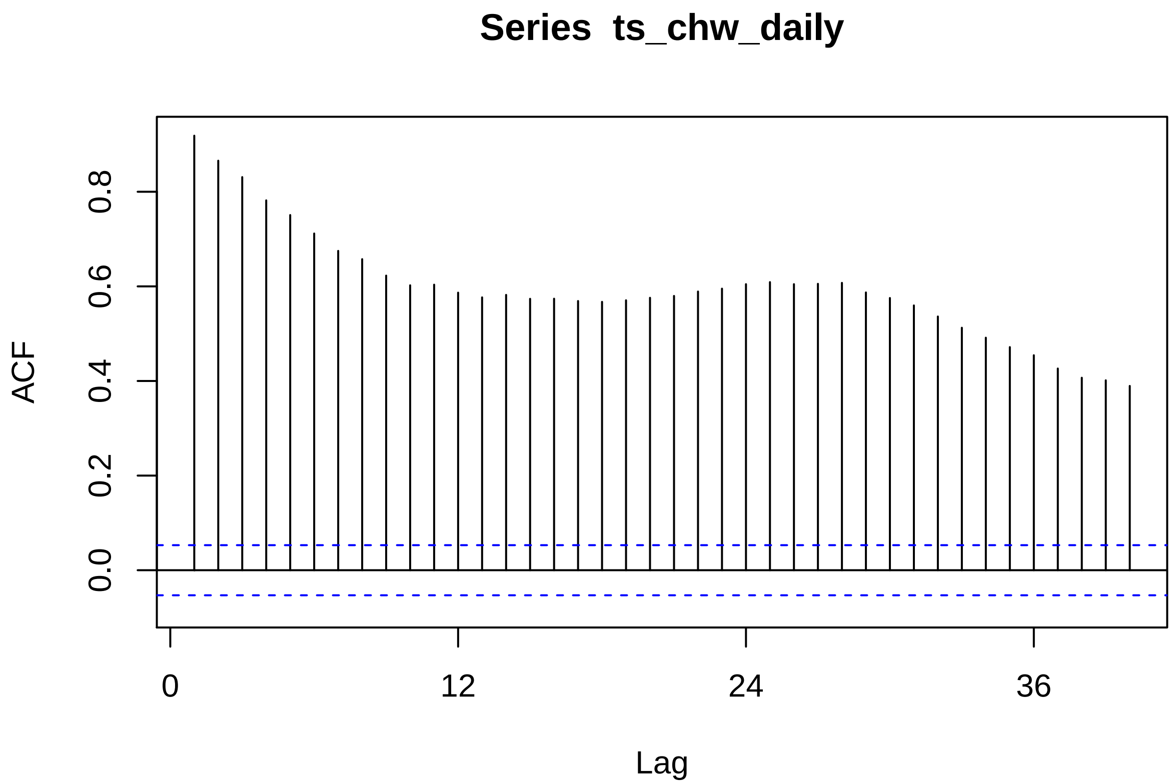

```r
#Acf and pacf for steam consumption series
Acf(ts_steam_daily, lag.max = 40)
```



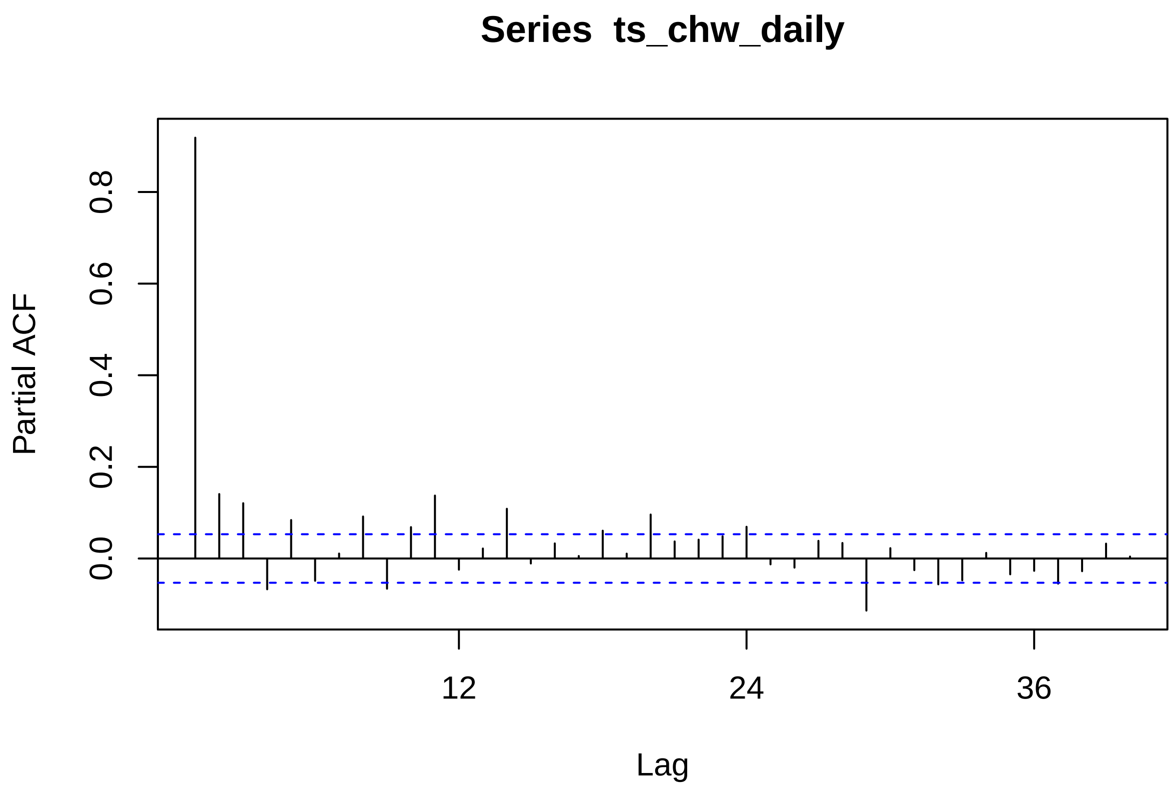

```
Pacf(ts_steam_daily, lag.max = 40)
```




```
#Acf and pacf for chilled water consumption series
Acf(ts_chw_daily, lag.max = 40)
```

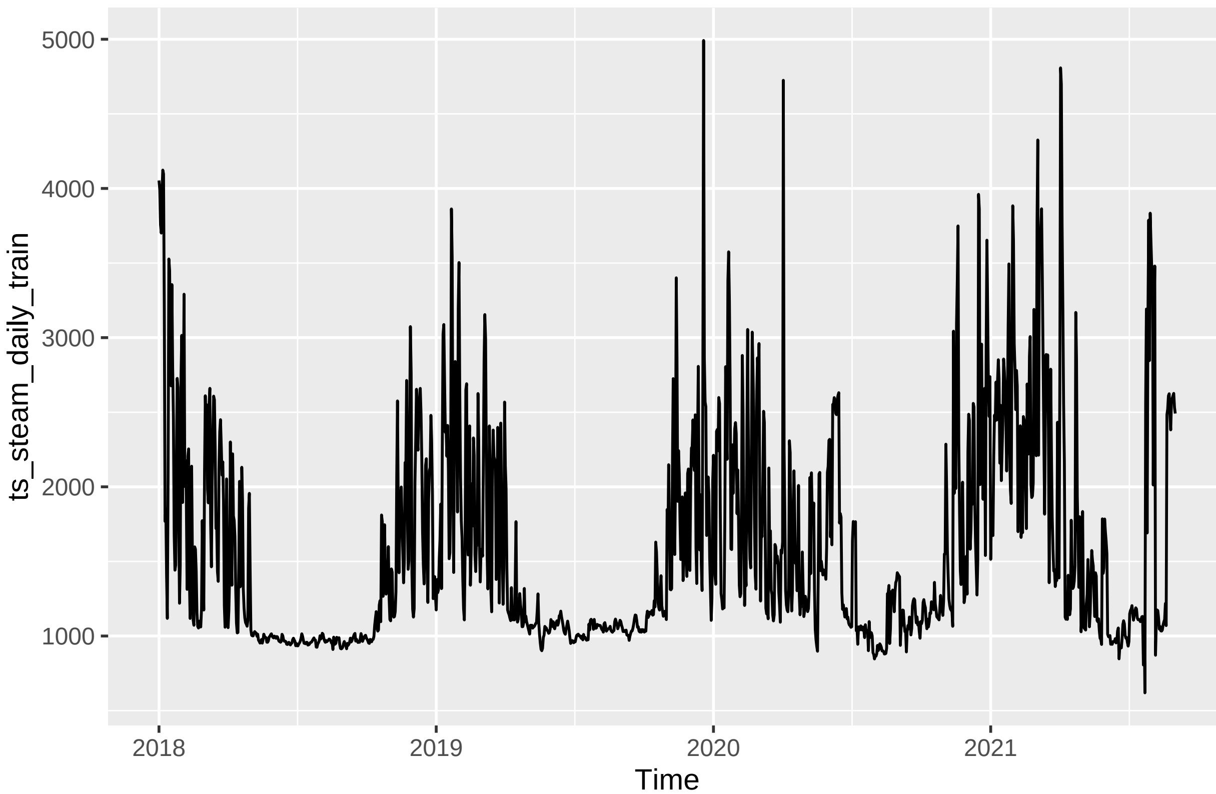



```
Pacf(ts_chw_daily, lag.max = 40)
```

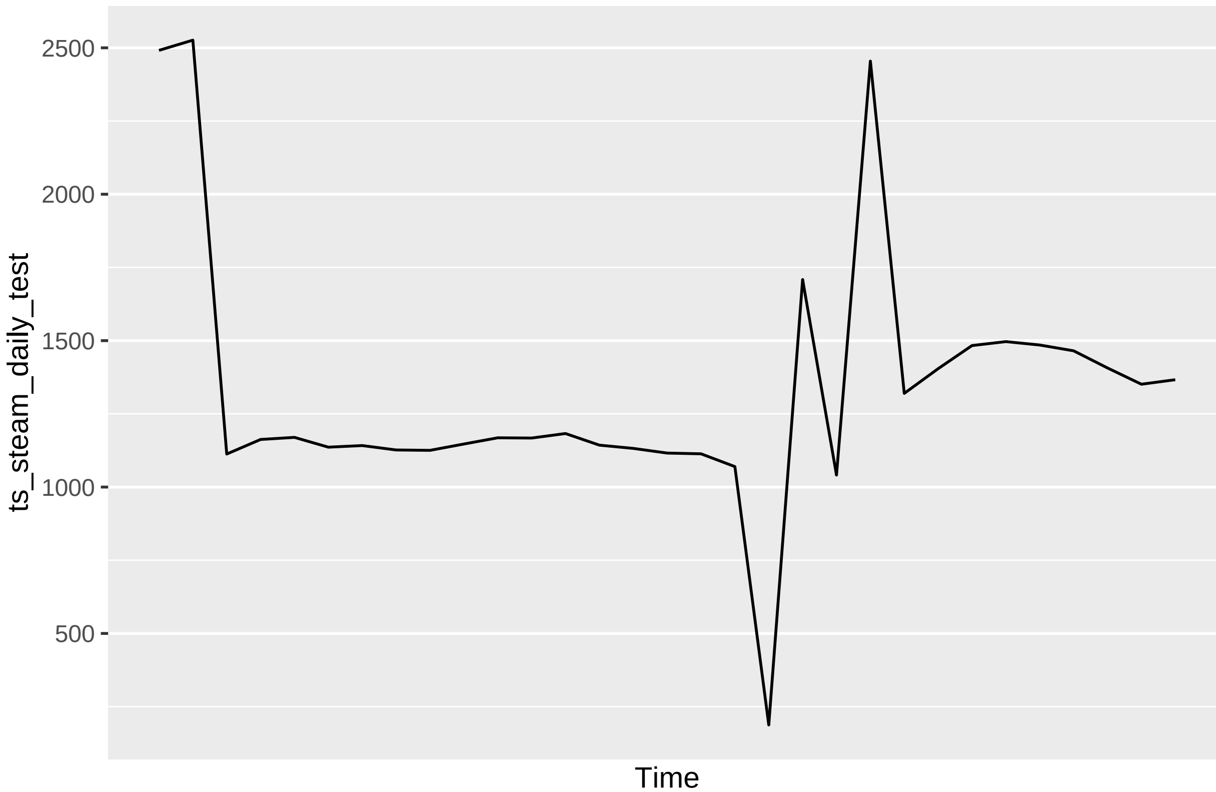


##Make train and test sets

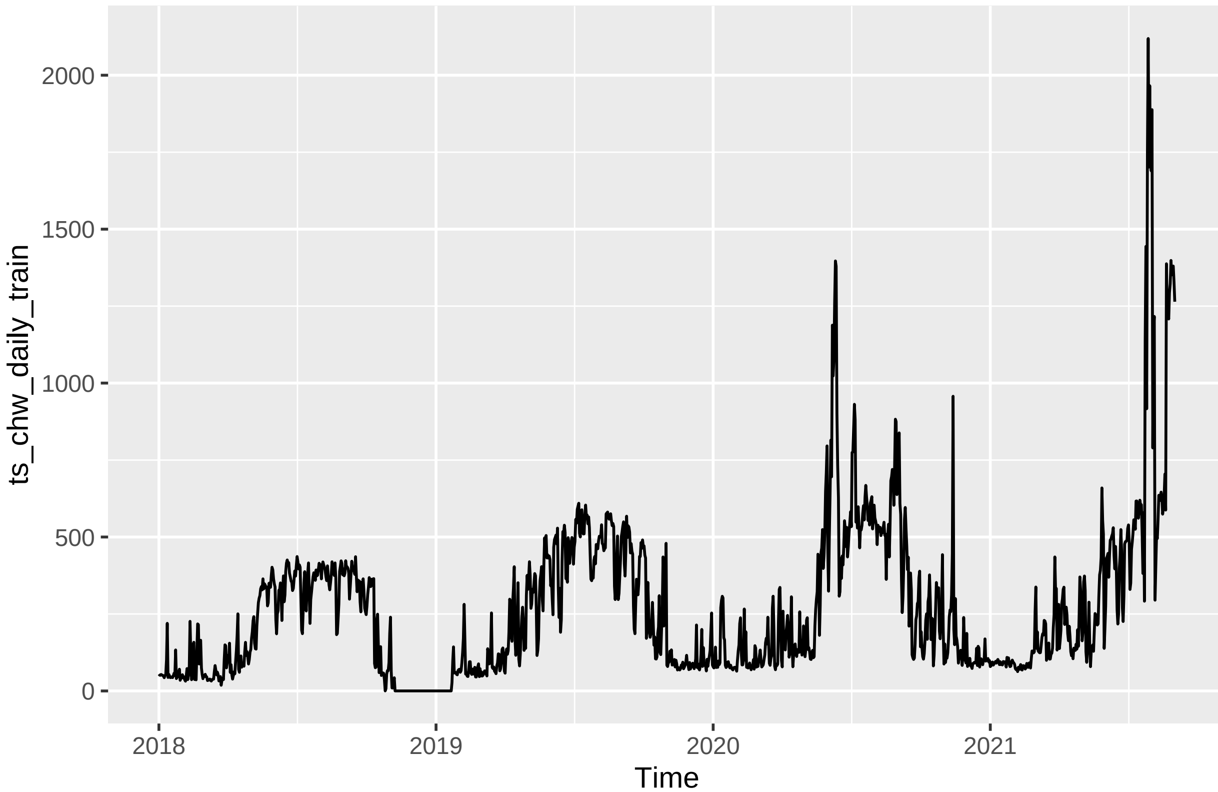
```r
#Making training and testing dataset for steam consumption
#regular file has h=121, try playing with smaller time horizon like 1 month

n_for = 30



ts_steam_daily_train <- subset(ts_steam_daily,
                                  end = length(ts_steam_daily)-n_for)

ts_steam_daily_test <- subset(ts_steam_daily,
                                 start = length(ts_steam_daily)-n_for)

autoplot(ts_steam_daily_train)
```



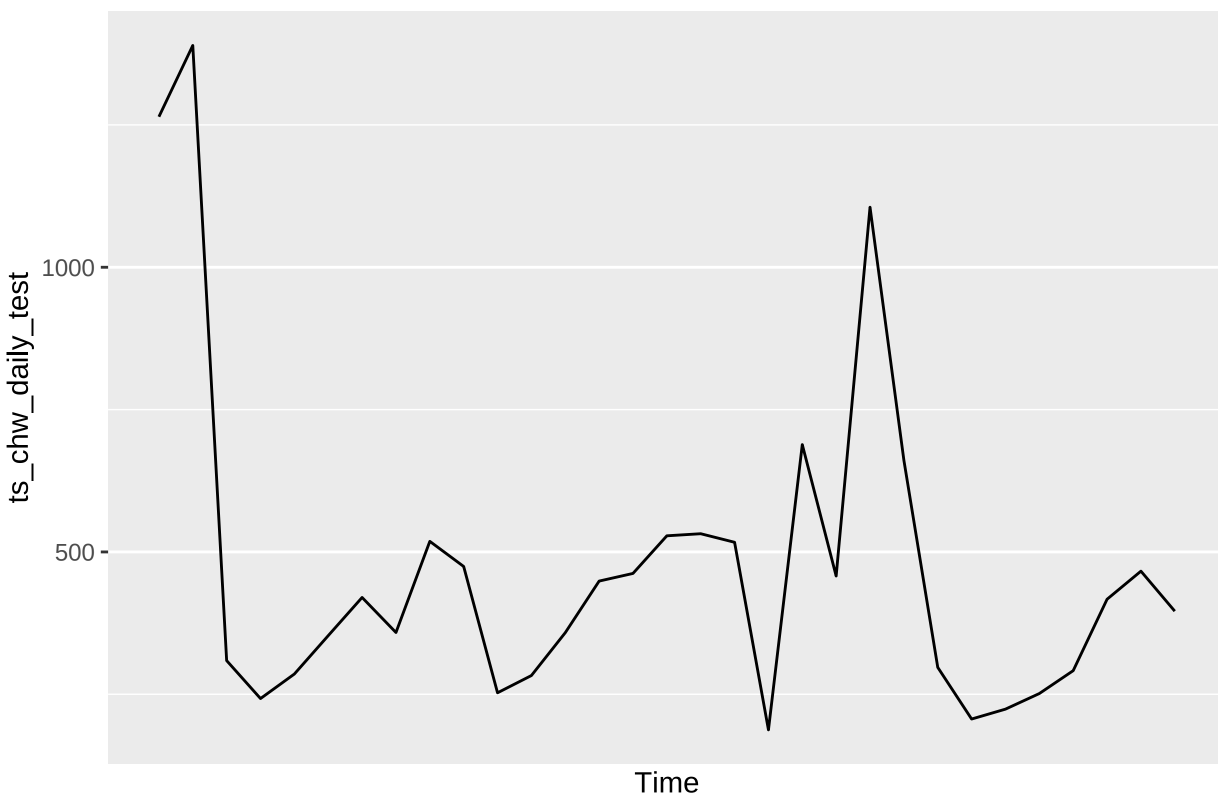

```r
autoplot(ts_steam_daily_test)
```




```
#Making training and testing datasets for chilled water consumption

ts_chw_daily_train <- subset(ts_chw_daily,
                                   end = length(ts_chw_daily)-n_for)

ts_chw_daily_test <- subset(ts_chw_daily,
                                   start = length(ts_chw_daily)-n_for)

autoplot(ts_chw_daily_train)
```

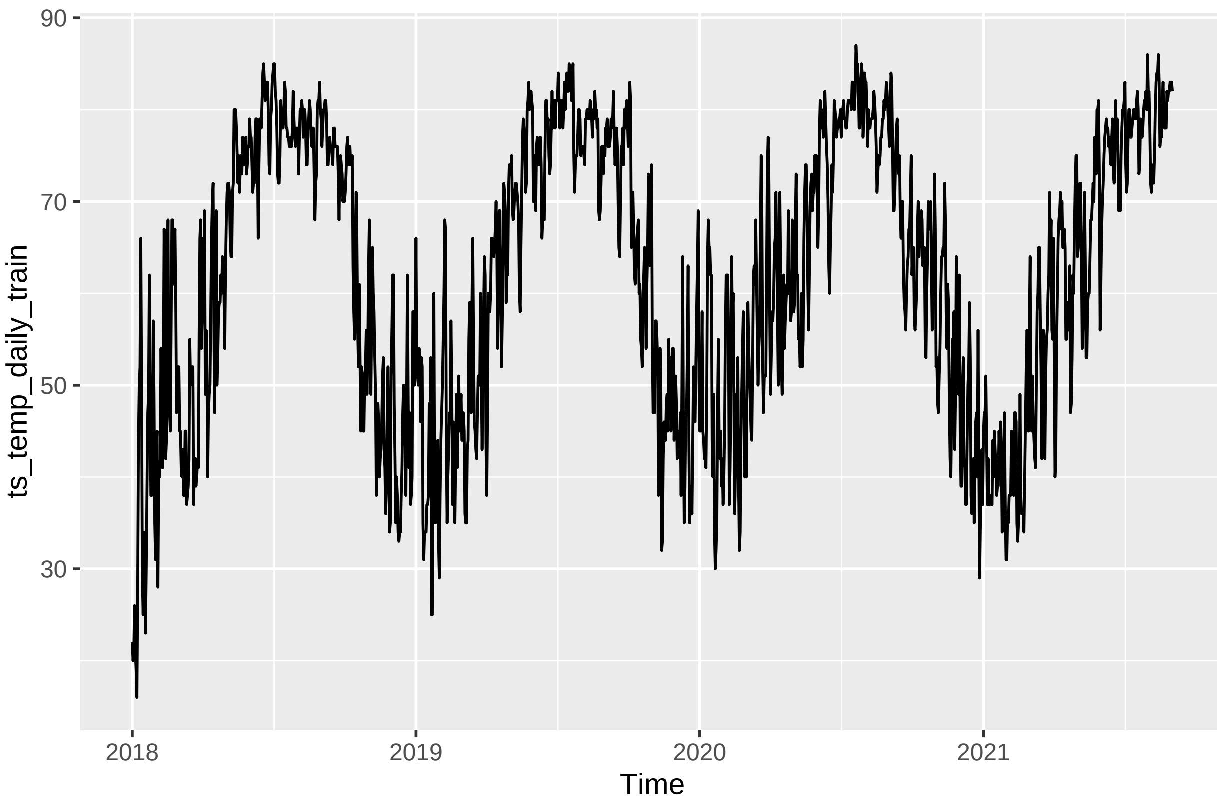



```
autoplot(ts_chw_daily_test)
```

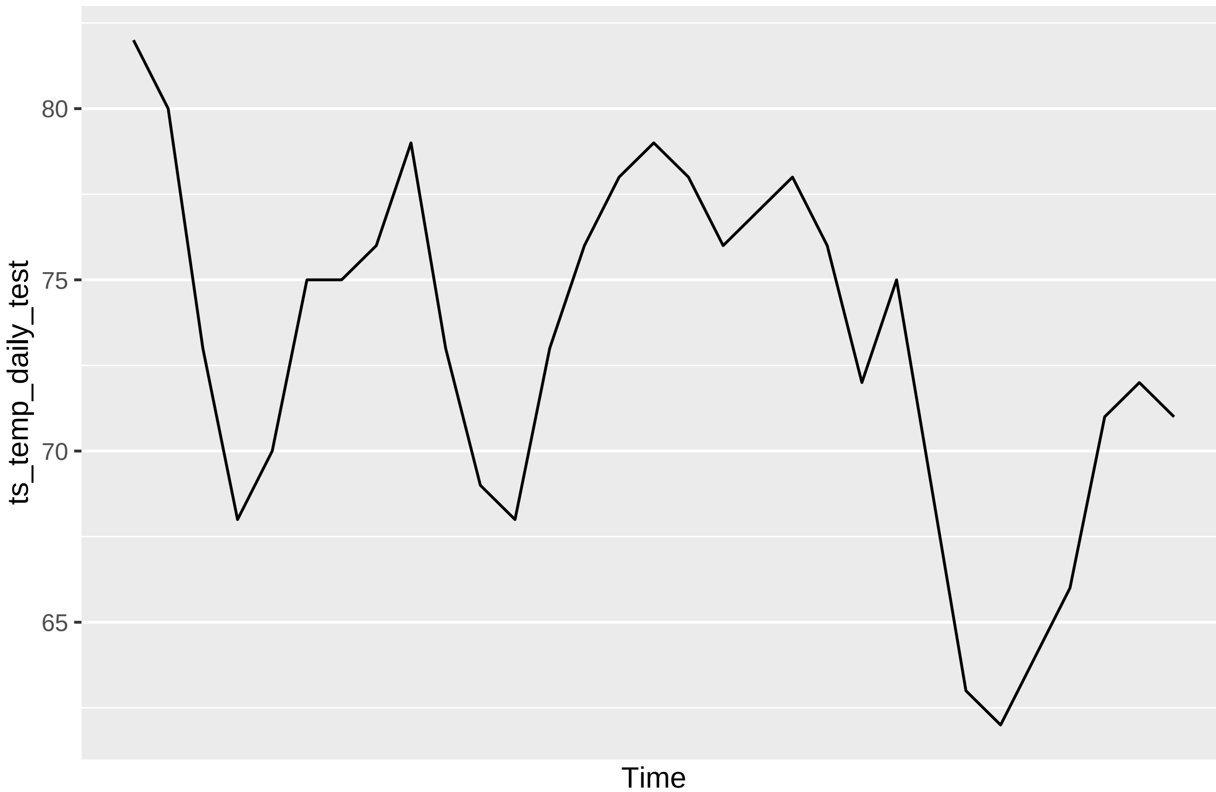



```
#Making training and testing datasets for temperature
```

```
ts_temp_daily_train <- subset(ts_temp_daily,
                              end = length(ts_temp_daily)-n_for)

ts_temp_daily_test <- subset(ts_temp_daily,
                              start = length(ts_temp_daily)-n_for)

autoplot(ts_temp_daily_train)
```

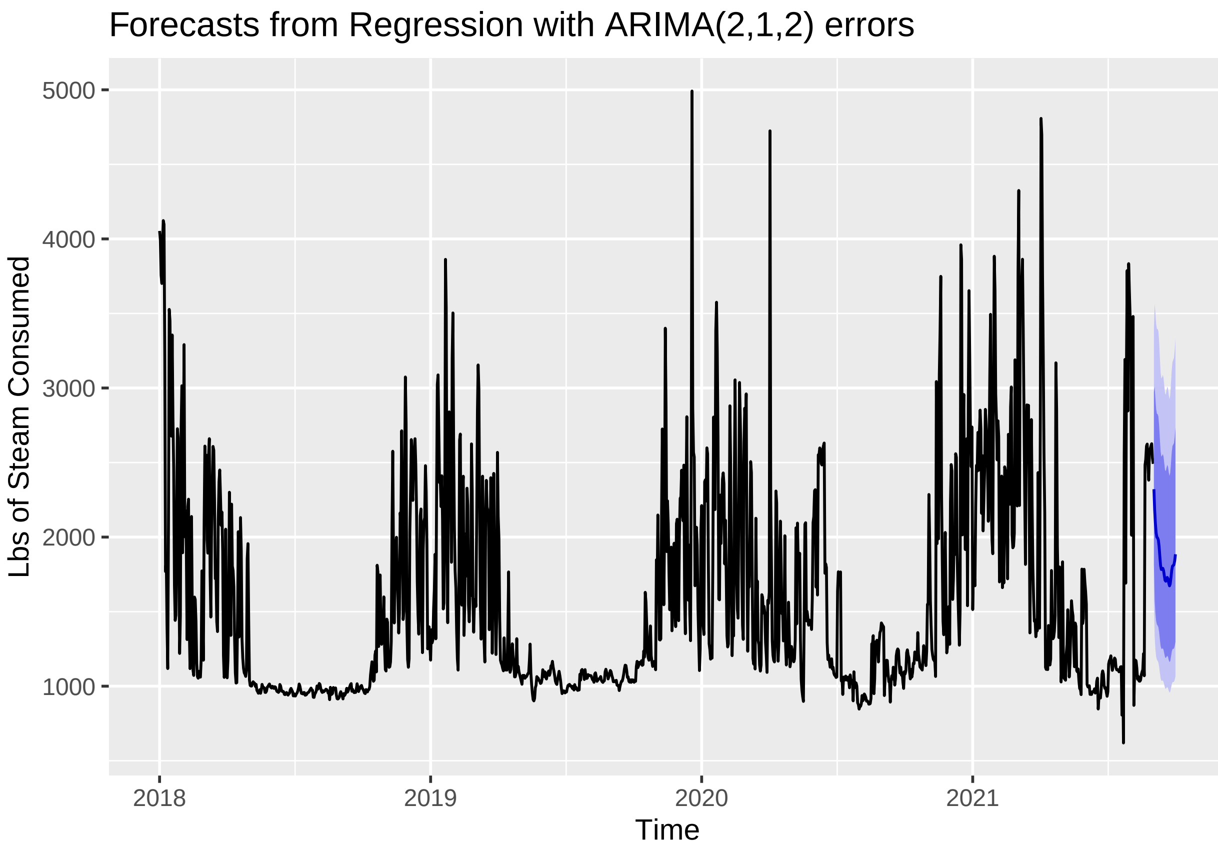



```
autoplot(ts_temp_daily_test)
```

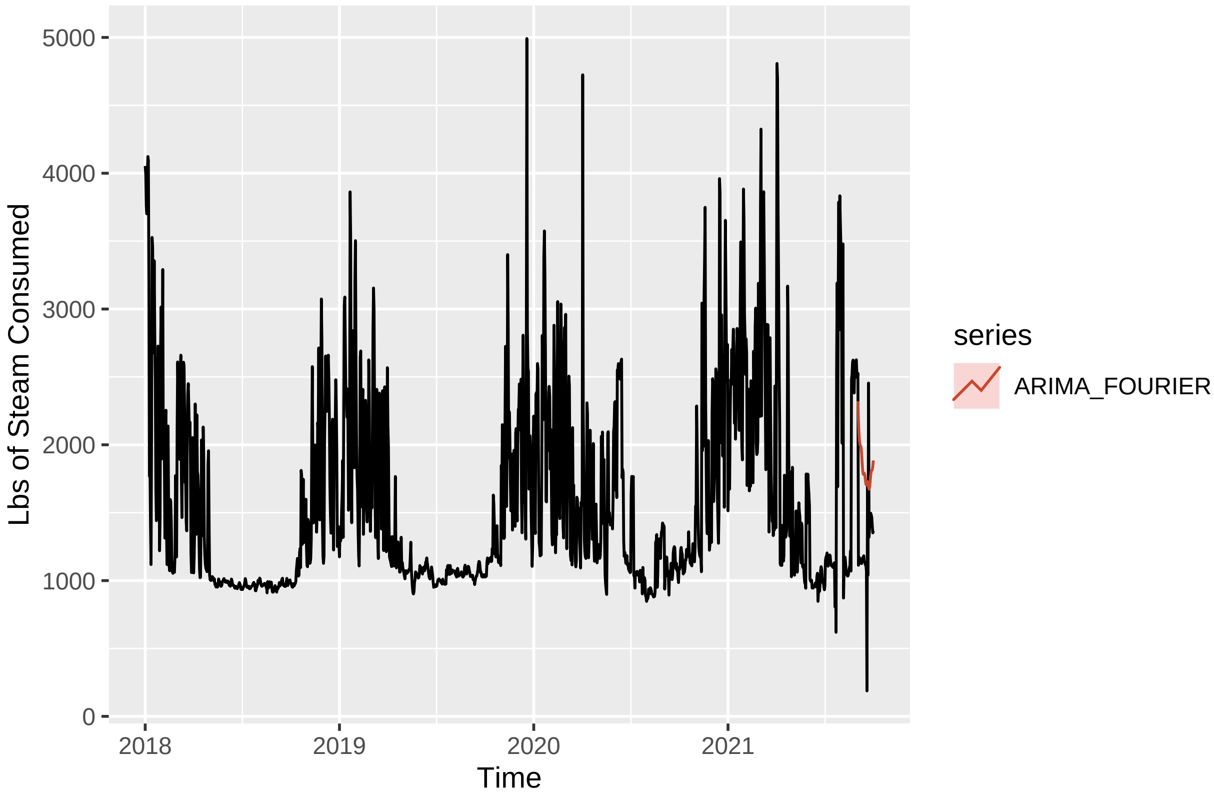


##Try models on training and testing sets for steam

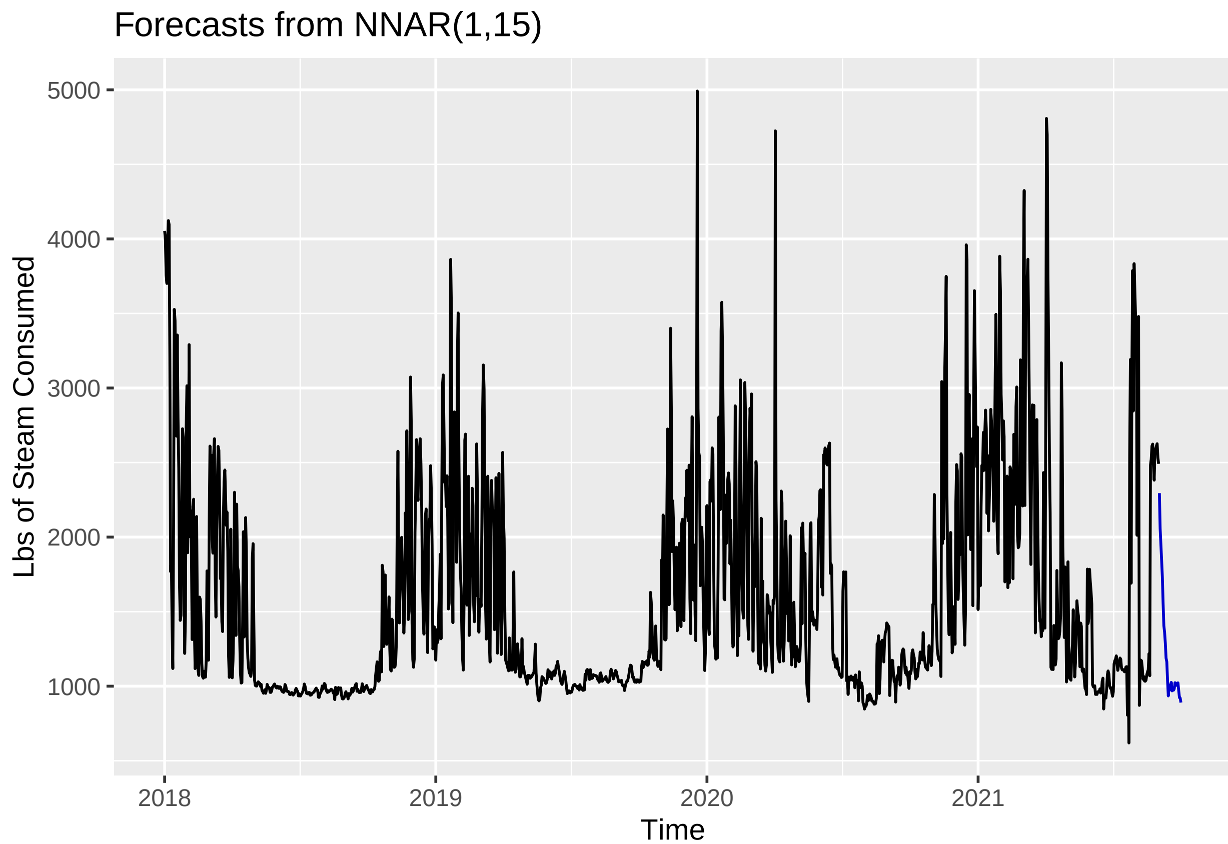
```
ARIMA_Four_fit <- auto.arima(ts_steam_daily_train,
                             seasonal=FALSE,
                             lambda=0,
                             xreg=fourier(ts_steam_daily_train,
                                          K=c(2,12))
)

#Forecast with ARIMA fit
#also need to specify h for fourier terms
ARIMA_Four_for <- forecast::forecast(ARIMA_Four_fit,
                                     xreg=fourier(ts_steam_daily_train,
                                                  K=c(2,12),
                                                  h=30),
                                     h=30
                                     )

#Plot forecasting results
autoplot(ARIMA_Four_for) + ylab("Lbs of Steam Consumed")
```



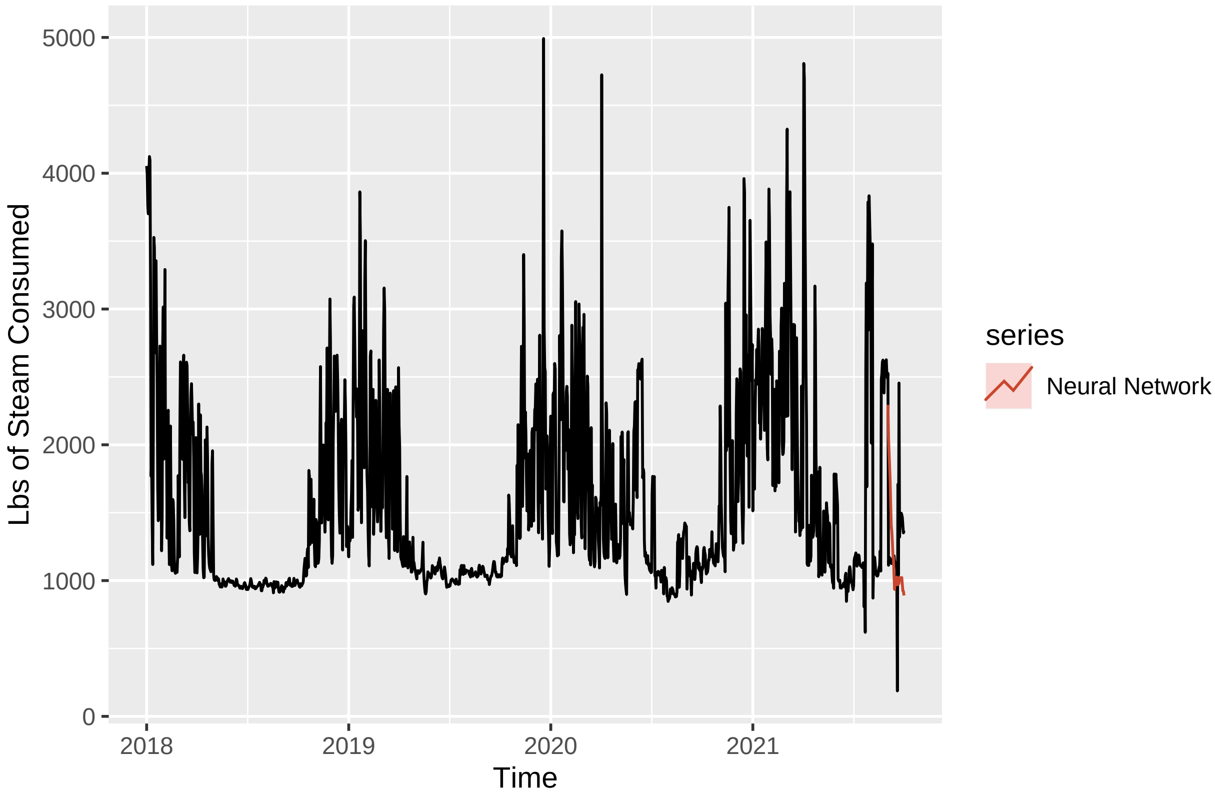

```r
#Plot model + observed data
autoplot(ts_steam_daily) +
  autolayer(ARIMA_Four_for, series="ARIMA_FOURIER",PI=FALSE) +
  ylab("Lbs of Steam Consumed")
```



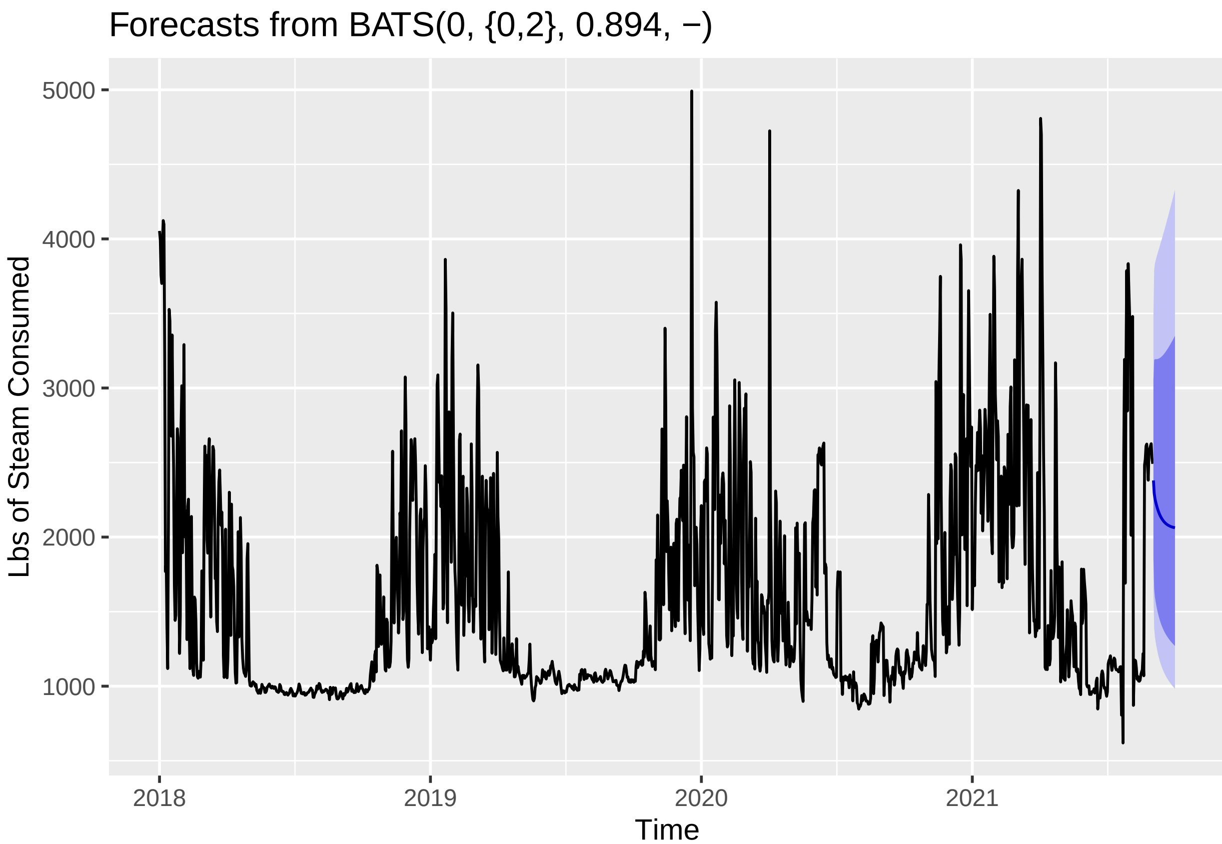

```
NN_fit <- nnetar(ts_steam_daily_train, p=1,P=0, xreg=fourier(ts_steam_daily_train, K=c(2,12)))

#NN_for <- forecast(NN_fit, h=30)
NN_for <- forecast::forecast(NN_fit, h=30,xreg=fourier(ts_steam_daily_train,
                                                        K=c(2,12),h=30))

#Plot forecasting results
autoplot(NN_for) +
  ylab("Lbs of Steam Consumed")
```



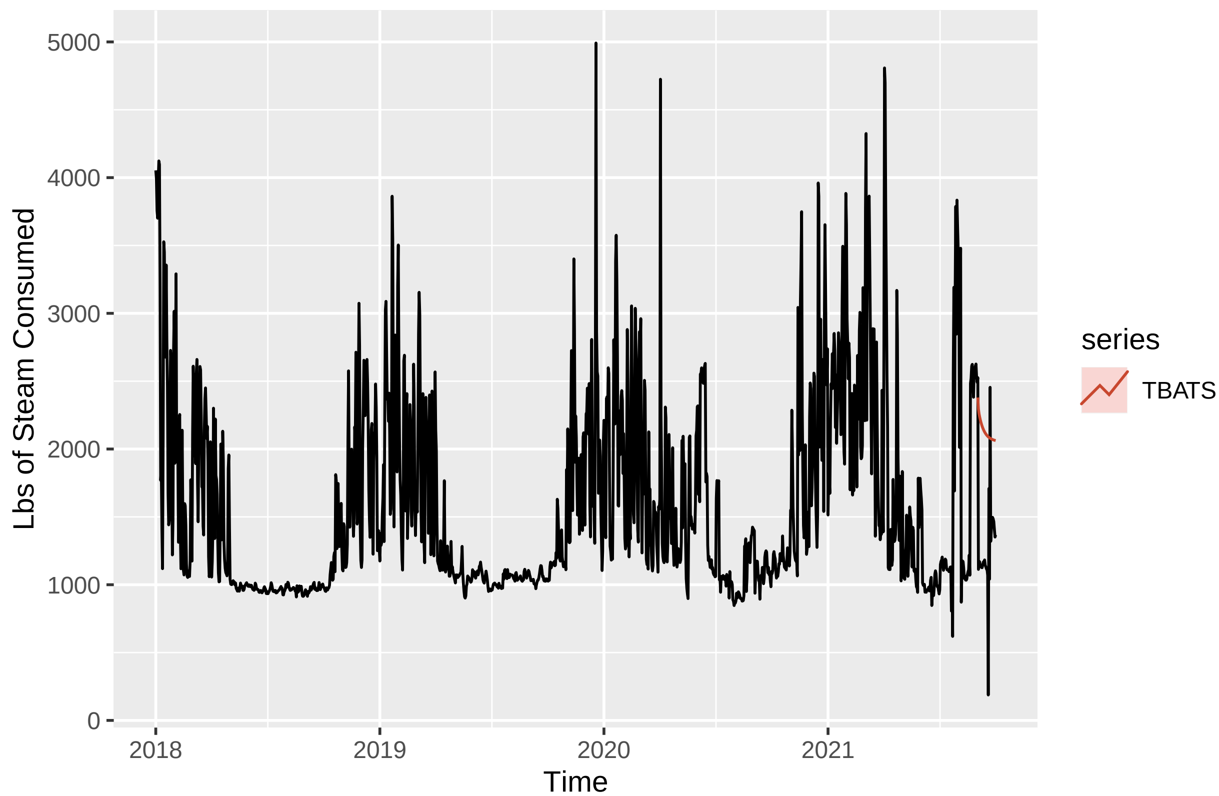

```r
#Plot model + observed data
autoplot(ts_steam_daily) +
  autolayer(NN_for, series="Neural Network",PI=FALSE)+
  ylab("Lbs of Steam Consumed")
```



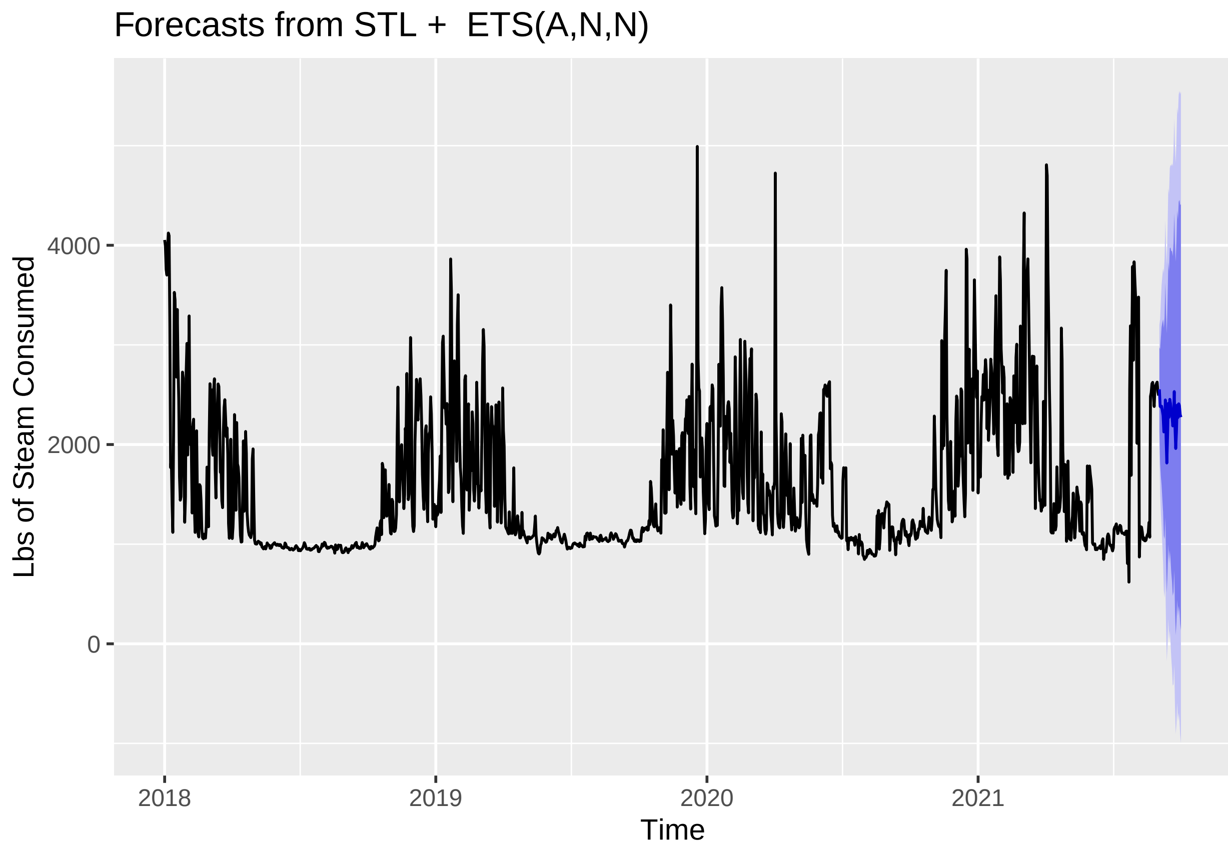

```
#TBATS
TBATS_fit <- tbats(ts_steam_daily_train)

TBATS_for <- forecast::forecast(TBATS_fit, h=30)

#Plot forecasting results
autoplot(TBATS_for)+
  ylab("Lbs of Steam Consumed")
```



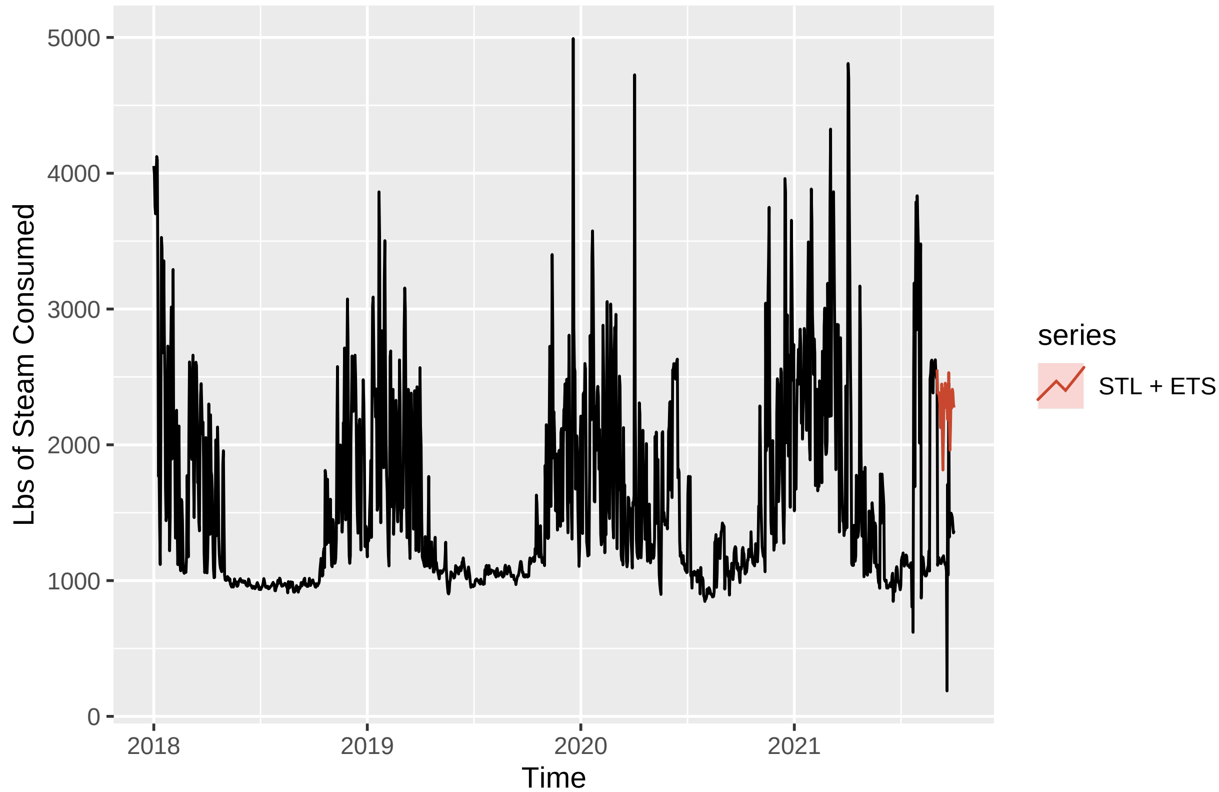

```r
#Plot model + observed data
autoplot(ts_steam_daily) +
  autolayer(TBATS_for, series="TBATS",PI=FALSE)+
  ylab("Lbs of Steam Consumed")
```




```
#fit and forecast STL + ETS model to data
ETS_fit <- stlf(ts_steam_daily_train,h=30)

#Plot forecasting results
autoplot(ETS_fit) + ylab("Lbs of Steam Consumed")
```




```
#Plot model + observed data
autoplot(ts_steam_daily) +
  autolayer(ETS_fit, series="STL + ETS",PI=FALSE)+
  ylab("Lbs of Steam Consumed")
```



##Check for model accuracy for steam

```r
#Model 1: STL + ETS
ETS_scores <- accuracy(ETS_fit$mean,ts_steam_daily_test)
ETS_scores
```

```
##                 ME     RMSE      MAE       MPE     MAPE        ACF1 Theil's U
## Test set -985.5099 1059.749 985.5099 -113.2218 113.2218 0.007853172 0.6483164
```

```r
#Model 2: ARIMA + Fourier
ARIMA_scores <- accuracy(ARIMA_Four_for$mean,ts_steam_daily_test)
ARIMA_scores
```

```
##                 ME     RMSE      MAE      MPE     MAPE       ACF1 Theil's U
## Test set -549.0109 681.3117 613.5782 -70.8607 73.47526 0.06865719 0.3755341
```

```r
#Model 3: TBATS
TBATS_scores <- accuracy(TBATS_for$mean,ts_steam_daily_test)
TBATS_scores
```

```
##                 ME     RMSE      MAE       MPE     MAPE        ACF1 Theil's U
## Test set -835.6644 926.7022 870.4477 -99.14358 100.5493 0.001476104 0.5587452
```

```r
#Model 4: Neural Network
NN_scores <-accuracy(NN_for$mean,ts_steam_daily_test)
NN_scores
```

```
##                ME     RMSE     MAE       MPE     MAPE      ACF1 Theil's U
## Test set 79.32179 514.1481 404.824 -10.30675 41.43956 0.4530624 0.5351867
```

```r
#create data frame
scores <- as.data.frame(
  rbind(ETS_scores, ARIMA_scores, TBATS_scores, NN_scores)
```

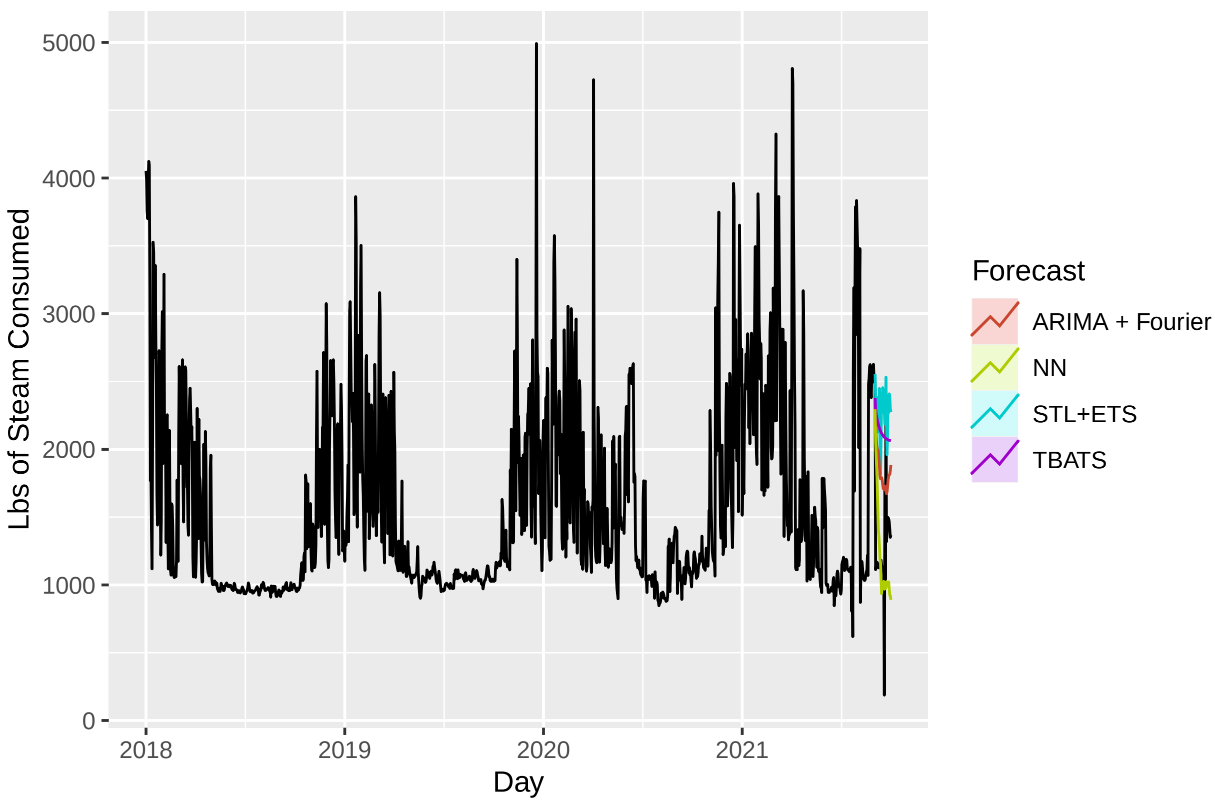
```
  )
row.names(scores) <- c("STL+ETS", "ARIMA+Fourier","TBATS","NN")

#choose model with lowest RMSE
best_model_index <- which.min(scores[,"RMSE"])
cat("The best model by RMSE is:", row.names(scores[best_model_index,]))
```

```
## The best model by RMSE is: NN
```

Table 1: Forecast Accuracy for Daily Lbs of Steam Consumption

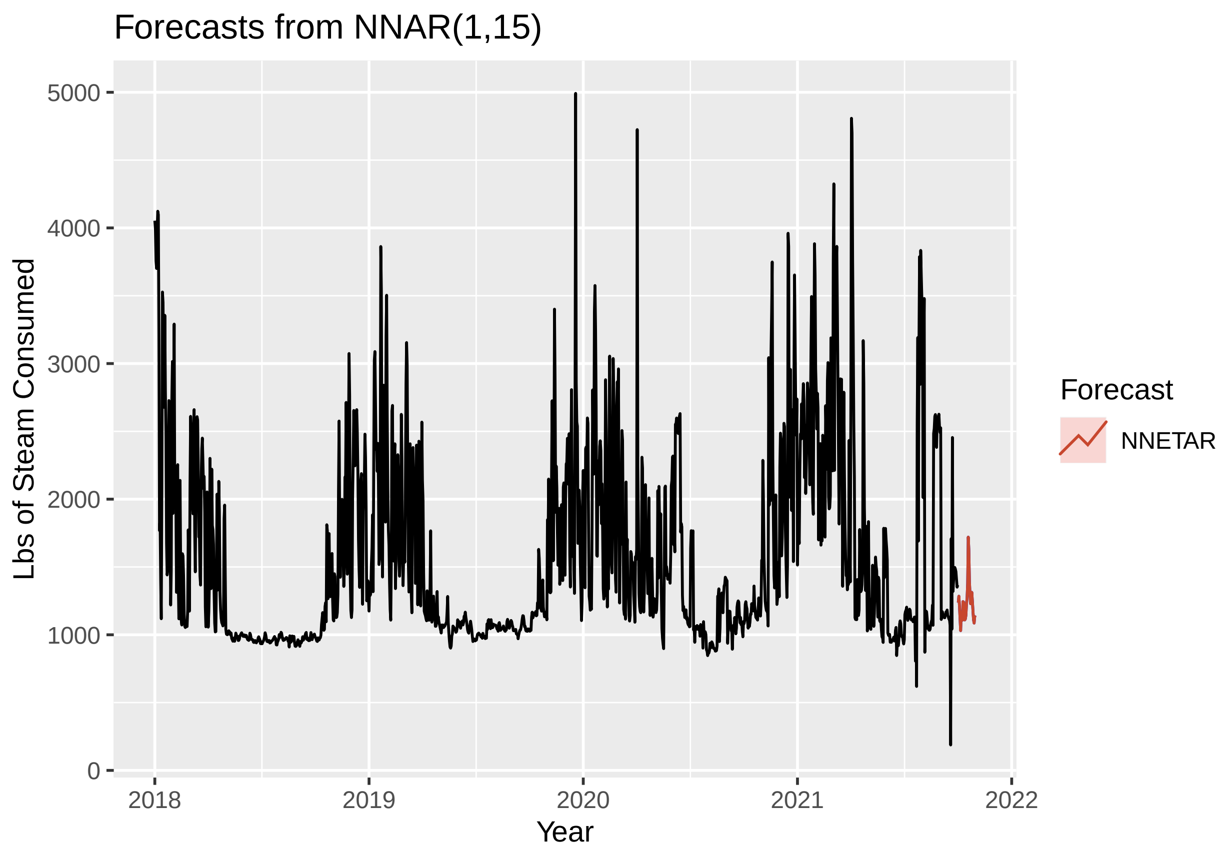
	ME	RMSE	MAE	MPE	MAPE	ACF1	Theil's U
STL+ETS	-985.50991	1059.7487	985.5099	-113.22179	113.22179	0.00785	0.64832
ARIMA+Fourier	-549.01090	681.3117	613.5782	-70.86070	73.47526	0.06866	0.37553
TBATS	-835.66438	926.7022	870.4477	-99.14358	100.54932	0.00148	0.55875
NN	79.32179	514.1481	404.8240	-10.30675	41.43956	0.45306	0.53519

```
autoplot(ts_steam_daily) +
  autolayer(ETS_fit, PI=FALSE, series="STL+ETS") +
  autolayer(ARIMA_Four_for, PI=FALSE, series="ARIMA + Fourier") +
  autolayer(TBATS_for,PI=FALSE, series="TBATS") +
  autolayer(NN_for,PI=FALSE,series="NN") +
  xlab("Day") + ylab("Lbs of Steam Consumed") +
  guides(colour=guide_legend(title="Forecast"))
```



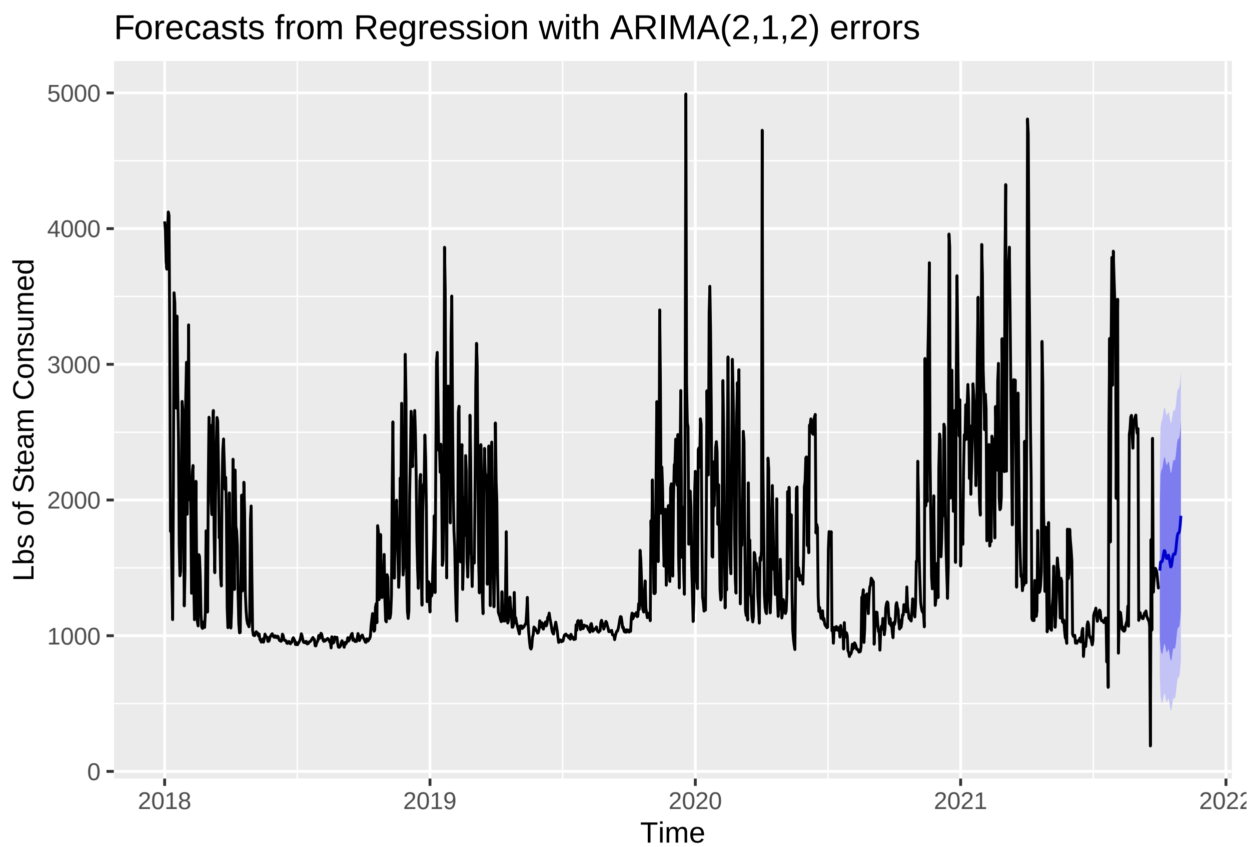
##Forecasting for steam

```r
#lowest RMSE

NN_fit2 <- nnetar(ts_steam_daily, p=1,P=0, xreg=fourier(ts_temp_daily, K=c(2,12)))

#NN_for <- forecast(NN_fit, h=30)
NN_for2 <- forecast::forecast(NN_fit2, h=30,xreg=fourier(ts_temp_daily,
                                                          K=c(2,12),h=30))


#Plot forecasting results
autoplot(NN_for2) +
  autolayer(NN_for2, series="NNETAR",PI=FALSE) +
  xlab("Year") + ylab("Lbs of Steam Consumed") +
  guides(colour=guide_legend(title="Forecast"))
```

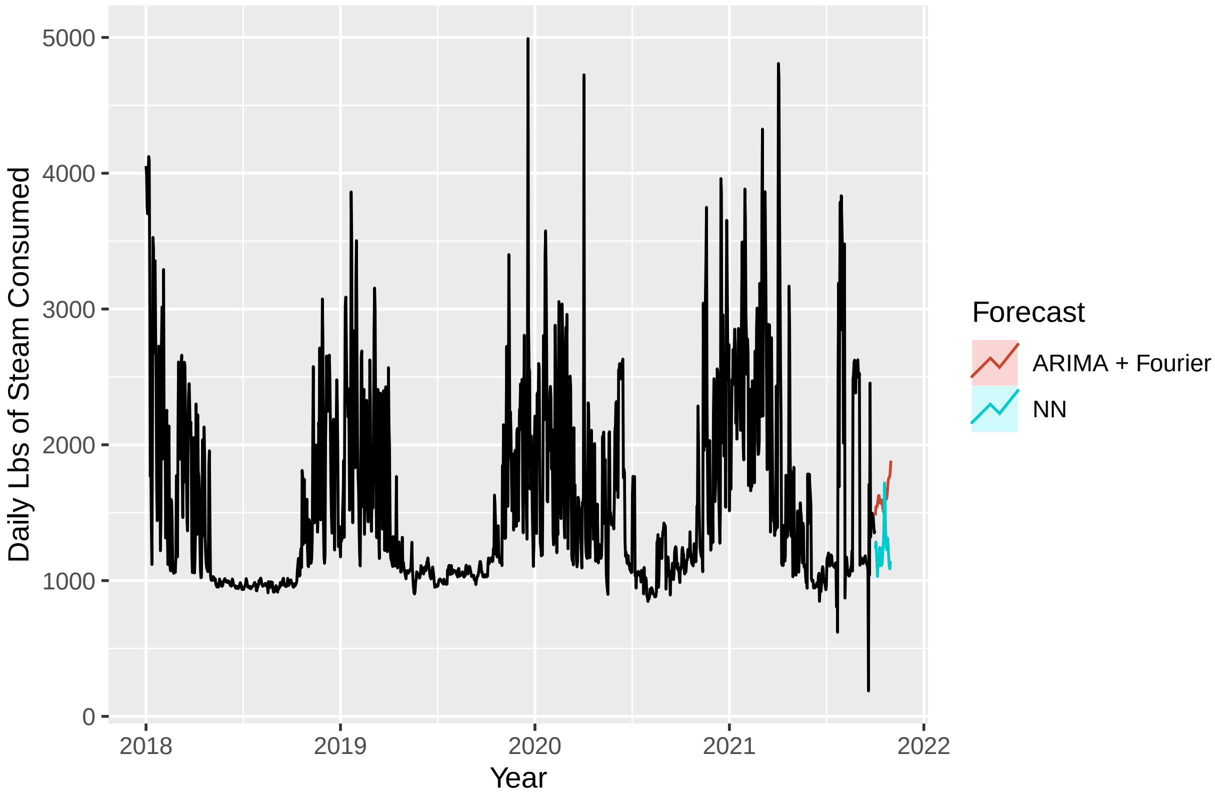



```r
ARIMA_Four_fit2 <- auto.arima(ts_steam_daily,
                              seasonal=FALSE,
                              xreg=fourier(ts_temp_daily,
                                           K=c(2,12))
) #use a log transformation (lambda=0) in the `auto.arima()` to ensure the forecasts and prediction int

#Forecast with ARIMA fit
#also need to specify h for fourier terms
ARIMA_Four_for2 <- forecast::forecast(ARIMA_Four_fit2,
                                      xreg=fourier(ts_temp_daily,
                                                   K=c(2,12),
                                                   h=30),
```

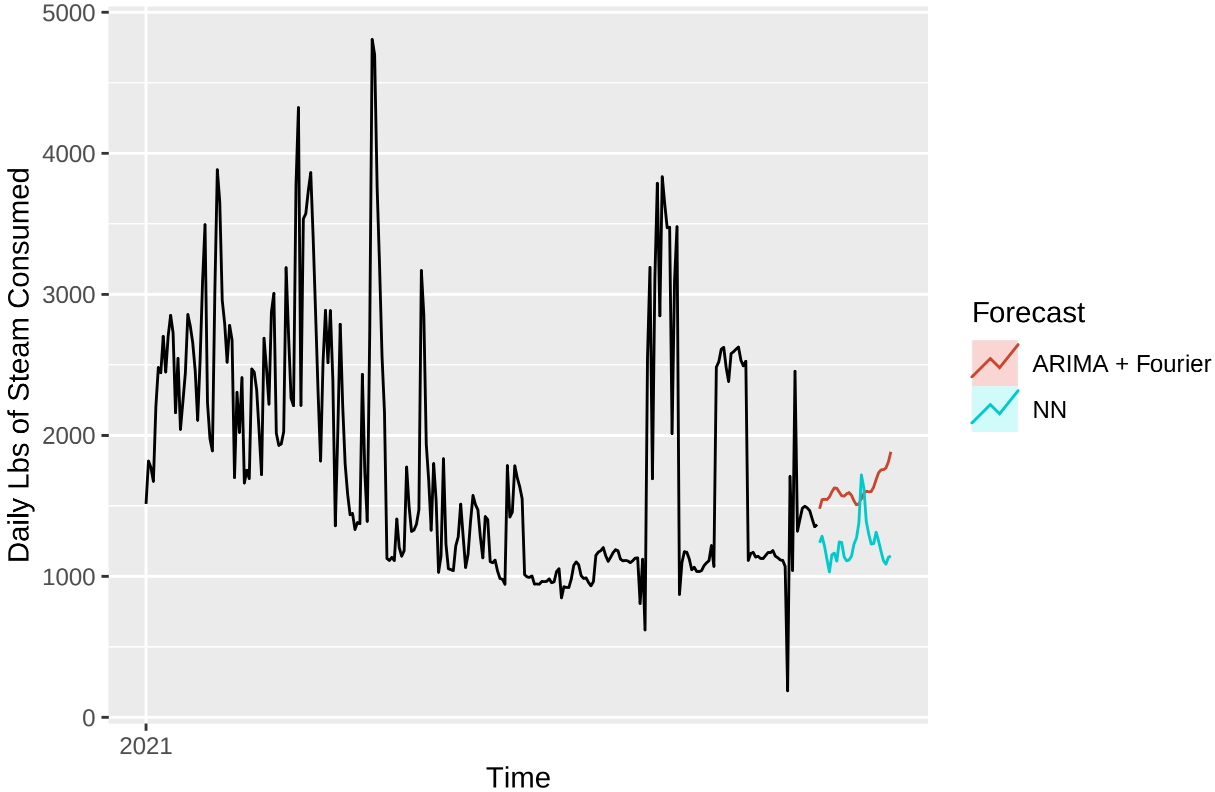
```r
                                    h=30
                                    )

#Plot forecasting results
autoplot(ARIMA_Four_for2) + ylab("Lbs of Steam Consumed")
```




```r
autoplot(ts_steam_daily) +
  autolayer(ARIMA_Four_for2, PI=FALSE, series="ARIMA + Fourier") +
  autolayer(NN_for2,PI=FALSE,series="NN") +
  xlab("Year") + ylab("Daily Lbs of Steam Consumed") +
  guides(colour=guide_legend(title="Forecast"))
```

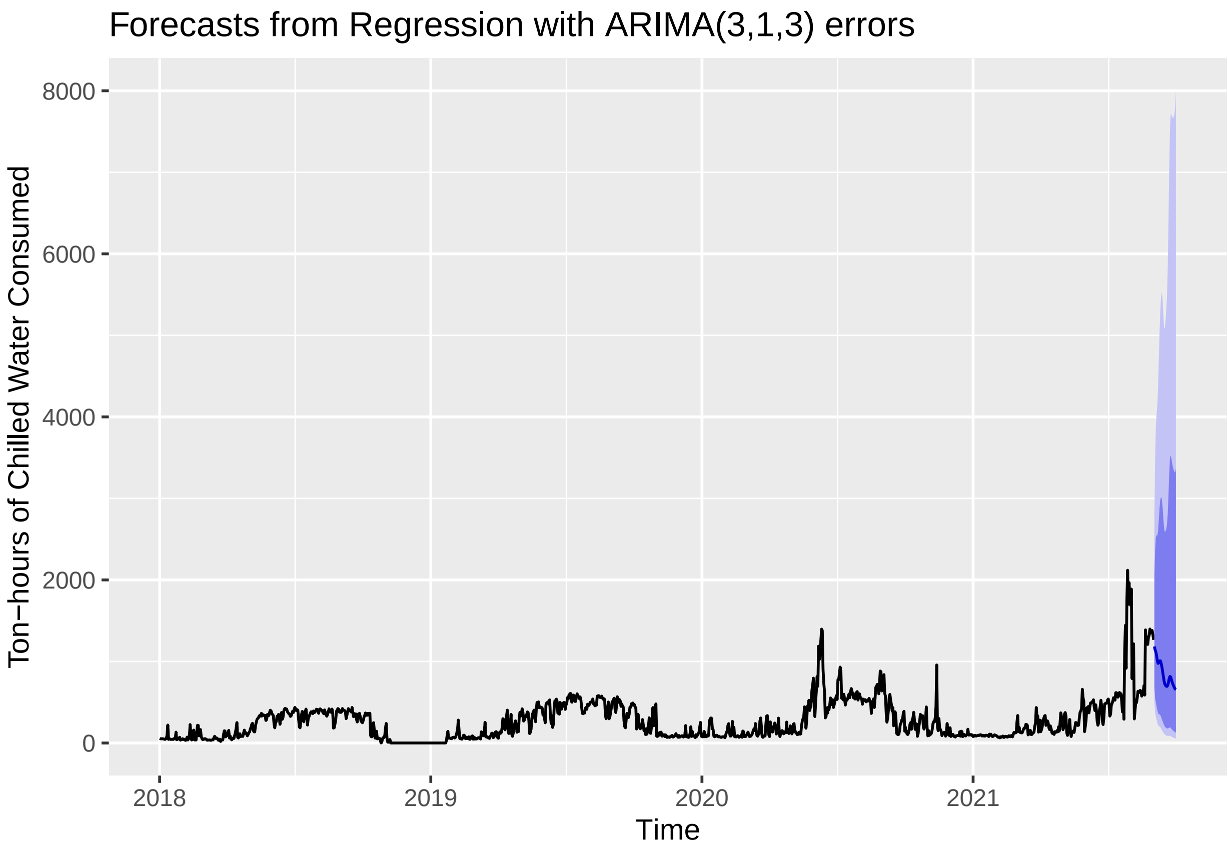



```r
autoplot(window(ts_steam_daily,start=2021)) +
  autolayer(ARIMA_Four_for2, PI=FALSE, series="ARIMA + Fourier") +
  autolayer(NN_for2,PI=FALSE, series="NN") +
  ylab("Daily Lbs of Steam Consumed") +
  guides(colour=guide_legend(title="Forecast"))
```

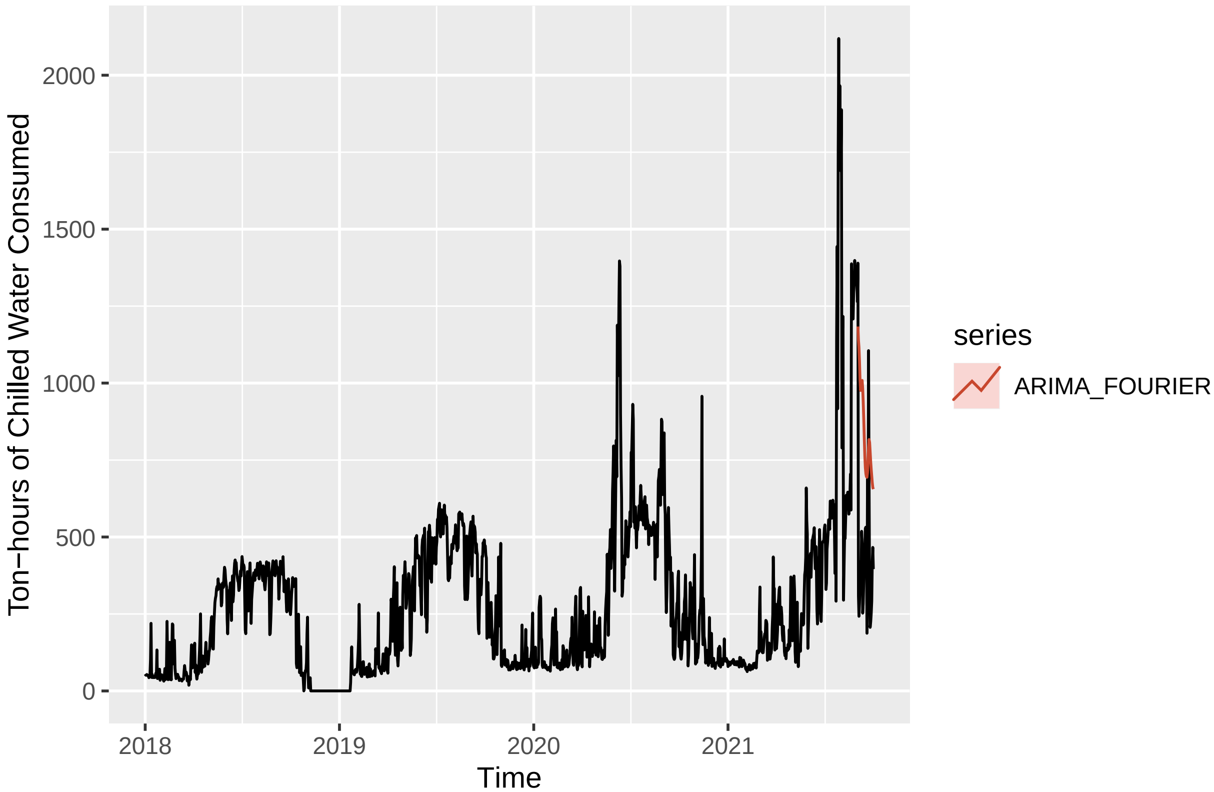


##Try models on training and testing sets for chilled water

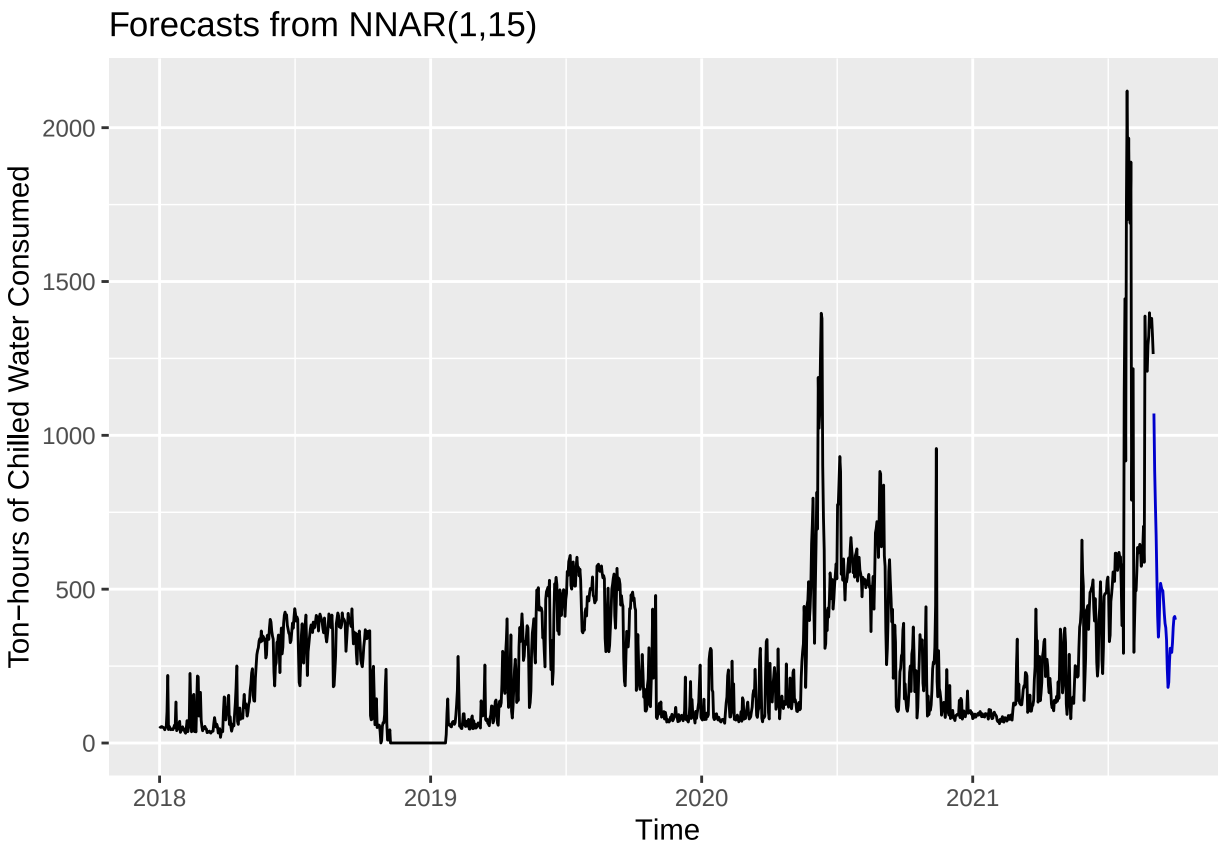
```
ARIMA_Four_fit3 <- auto.arima(ts_chw_daily_train,
                              seasonal=FALSE,
                              lambda=0,
                              xreg=fourier(ts_chw_daily_train,
                                           K=c(2,12))
)

#Forecast with ARIMA fit
#also need to specify h for fourier terms
ARIMA_Four_for3 <- forecast::forecast(ARIMA_Four_fit3,
                                      xreg=fourier(ts_chw_daily_train,
                                                   K=c(2,12),
                                                   h=30),
                                      h=30
                                      )

#Plot forecasting results
autoplot(ARIMA_Four_for3) + ylab("Ton-hours of Chilled Water Consumed")
```



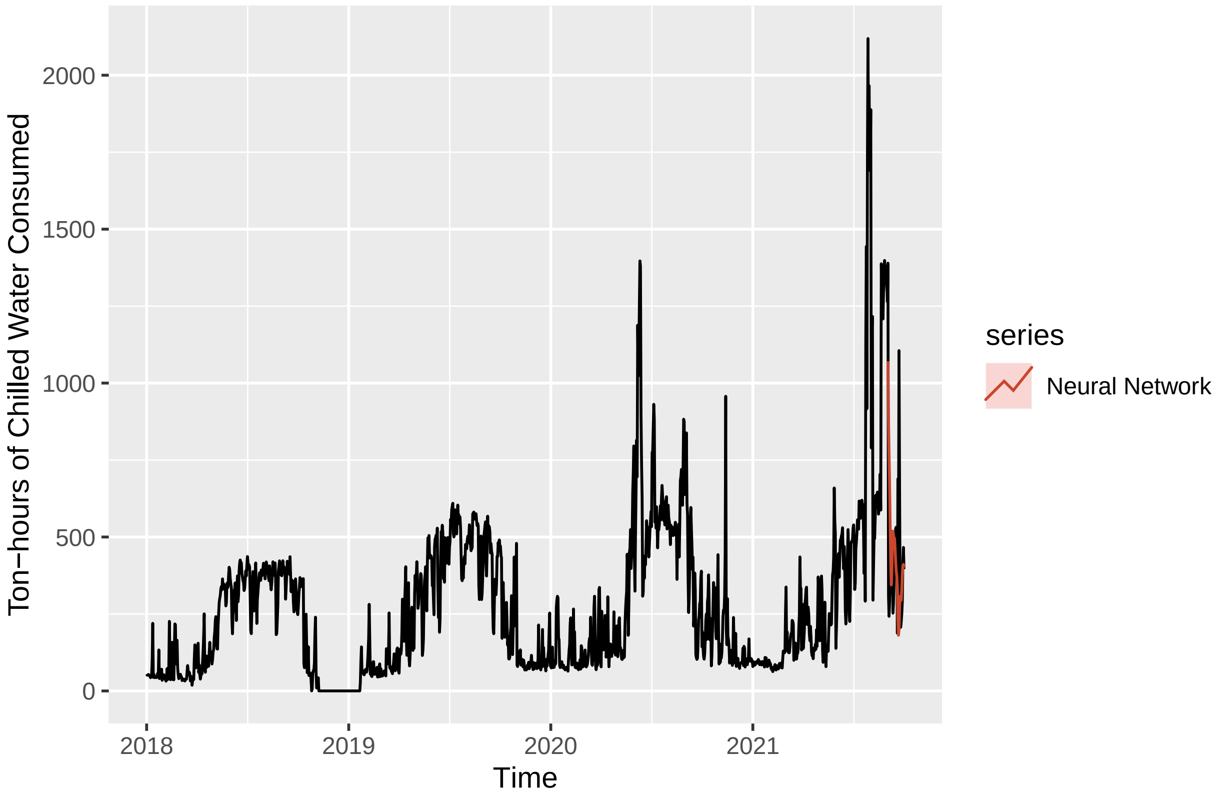

```
#Plot model + observed data
autoplot(ts_chw_daily) +
  autolayer(ARIMA_Four_for3, series="ARIMA_FOURIER",PI=FALSE) +
  ylab("Ton-hours of Chilled Water Consumed")
```



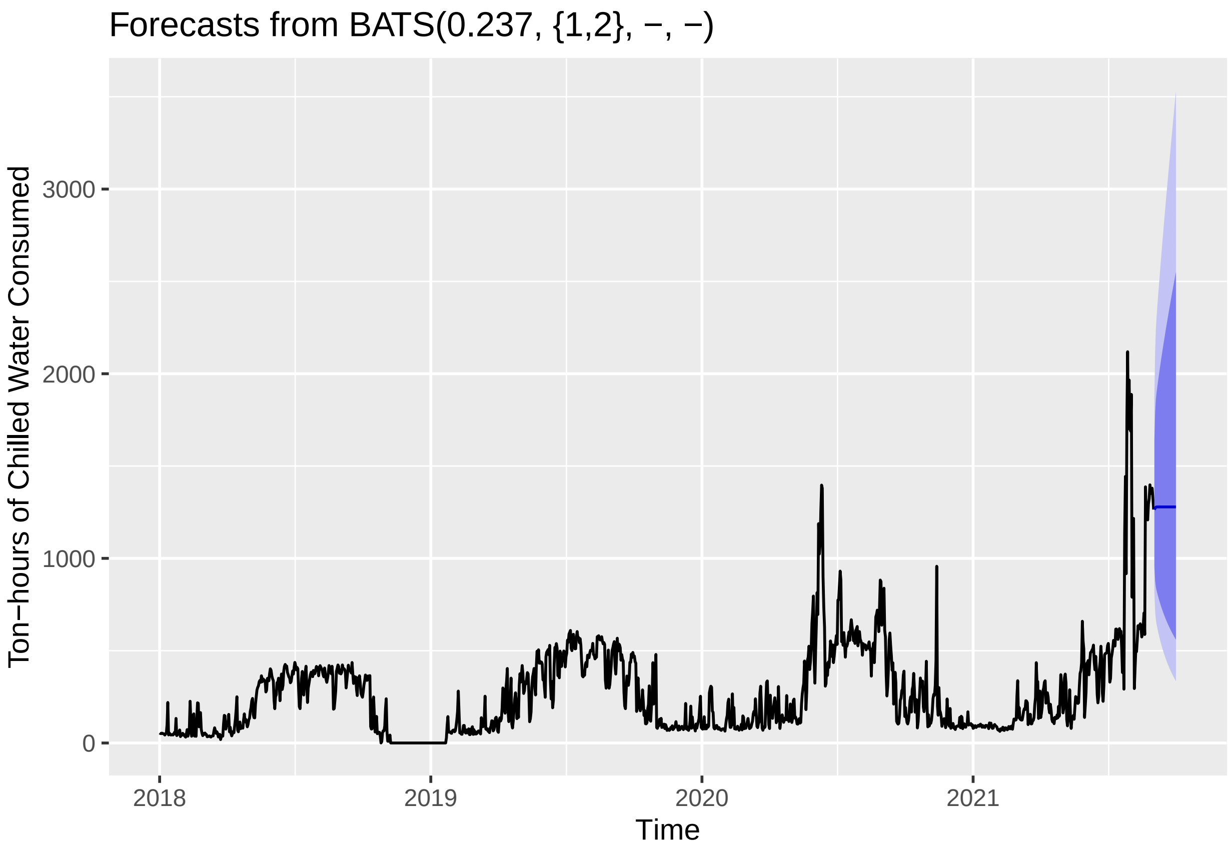

```r
NN_fit3 <- nnetar(ts_chw_daily_train, p=1,P=0, xreg=fourier(ts_chw_daily_train, K=c(2,12)))

#NN_for <- forecast(NN_fit, h=30)
NN_for3 <- forecast::forecast(NN_fit3, h=30,xreg=fourier(ts_chw_daily_train,
                                                          K=c(2,12),h=30))

#Plot forecasting results
autoplot(NN_for3) +
  ylab("Ton-hours of Chilled Water Consumed")
```



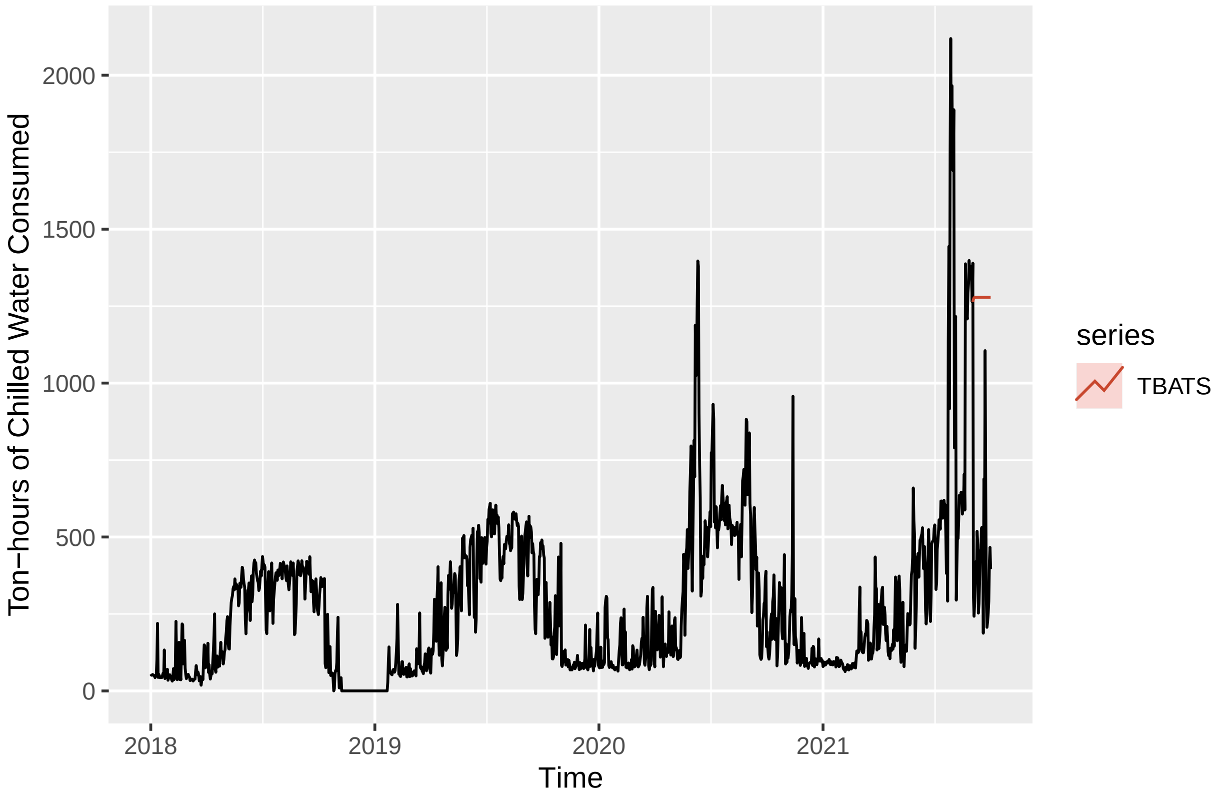

```
#Plot model + observed data
autoplot(ts_chw_daily) +
  autolayer(NN_for3, series="Neural Network",PI=FALSE)+
  ylab("Ton-hours of Chilled Water Consumed")
```



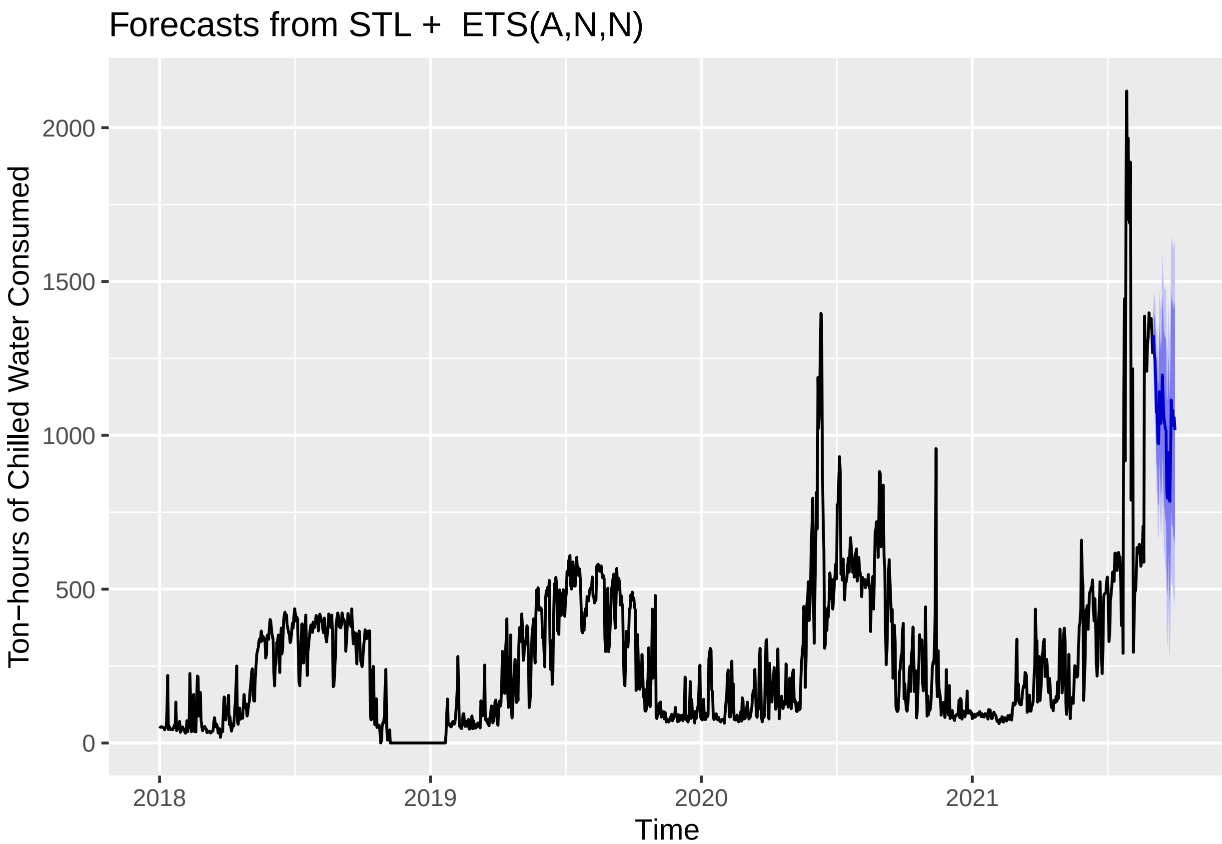

```
#TBATS
TBATS_fit2 <- tbats(ts_chw_daily_train)

TBATS_for2 <- forecast::forecast(TBATS_fit2, h=30)

#Plot forecasting results
autoplot(TBATS_for2)+
  ylab("Ton-hours of Chilled Water Consumed")
```



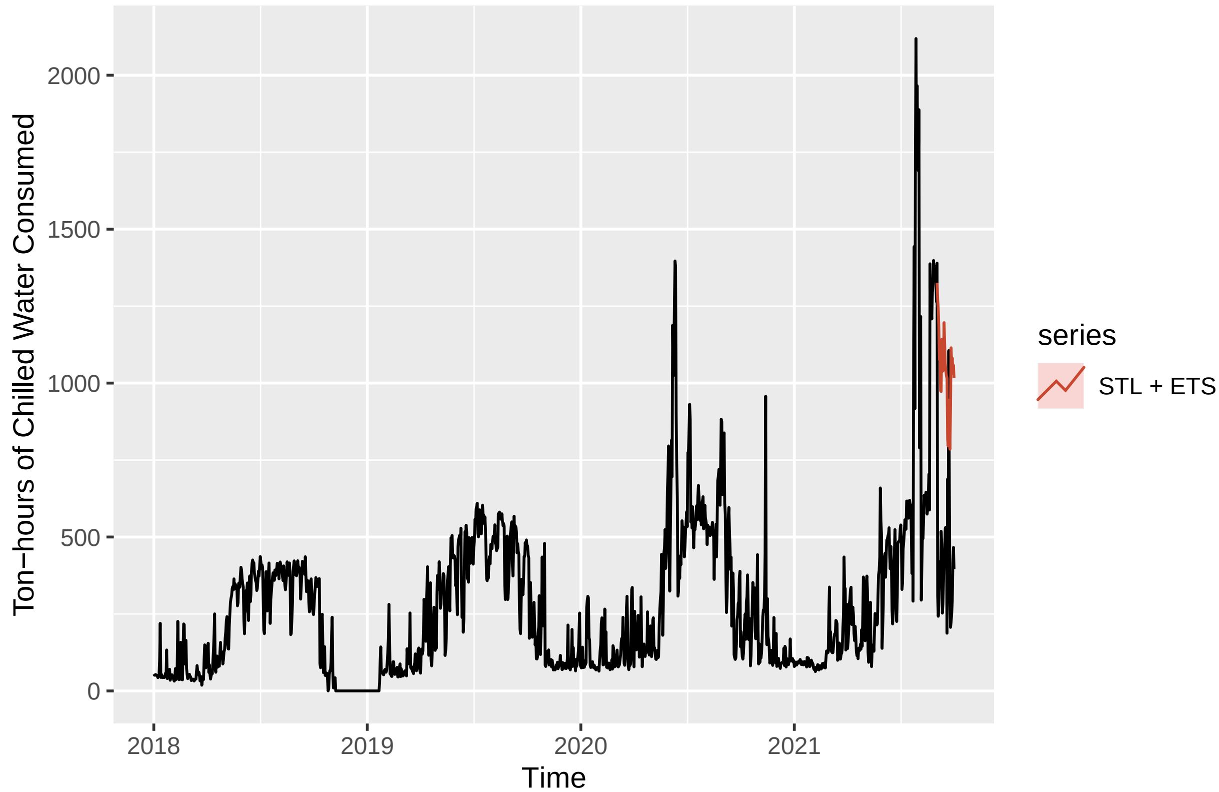

```
#Plot model + observed data
autoplot(ts_chw_daily) +
  autolayer(TBATS_for2, series="TBATS",PI=FALSE)+
  ylab("Ton-hours of Chilled Water Consumed")
```




```
#fit and forecast STL + ETS model to data
ETS_fit2 <- stlf(ts_chw_daily_train,h=30)

#Plot forecasting results
autoplot(ETS_fit2) + ylab("Ton-hours of Chilled Water Consumed")
```




```r
#Plot model + observed data
autoplot(ts_chw_daily) +
  autolayer(ETS_fit2, series="STL + ETS",PI=FALSE)+
  ylab("Ton-hours of Chilled Water Consumed")
```



##Check for model accuracy for chilled water

```
#Model 1: STL + ETS
ETS_scores2 <- accuracy(ETS_fit2$mean,ts_chw_daily_test)
ETS_scores2
```

```
##                  ME     RMSE      MAE       MPE     MAPE      ACF1 Theil's U
## Test set -600.2523 660.9109 615.2337 -189.1489 190.4252 0.3683979  3.274696
```

```
#Model 2: ARIMA + Fourier
ARIMA_scores2 <- accuracy(ARIMA_Four_for3$mean,ts_chw_daily_test)
ARIMA_scores2
```

```
##                  ME    RMSE      MAE       MPE     MAPE      ACF1 Theil's U
## Test set -404.1935 490.401 438.8154 -133.8777 136.7561 0.3954729  2.371499
```

```
#Model 3: TBATS
TBATS_scores2 <- accuracy(TBATS_for2$mean,ts_chw_daily_test)
TBATS_scores2
```

```
##                 ME     RMSE      MAE       MPE    MAPE      ACF1 Theil's U
## Test set -831.772 869.3554 840.1744 -251.7013 252.306 0.1088153  4.232442
```

```
#Model 4: Neural Network
NN_scores2 <-accuracy(NN_for3$mean,ts_chw_daily_test)
NN_scores2
```

```
##                 ME     RMSE      MAE       MPE     MAPE      ACF1 Theil's U
## Test set 3.598657 284.5008 194.2376 -21.29257 48.78501 0.4947084   1.24258
```

```
#create data frame
scores2 <- as.data.frame(
  rbind(ETS_scores2, ARIMA_scores2, TBATS_scores2, NN_scores2)
```

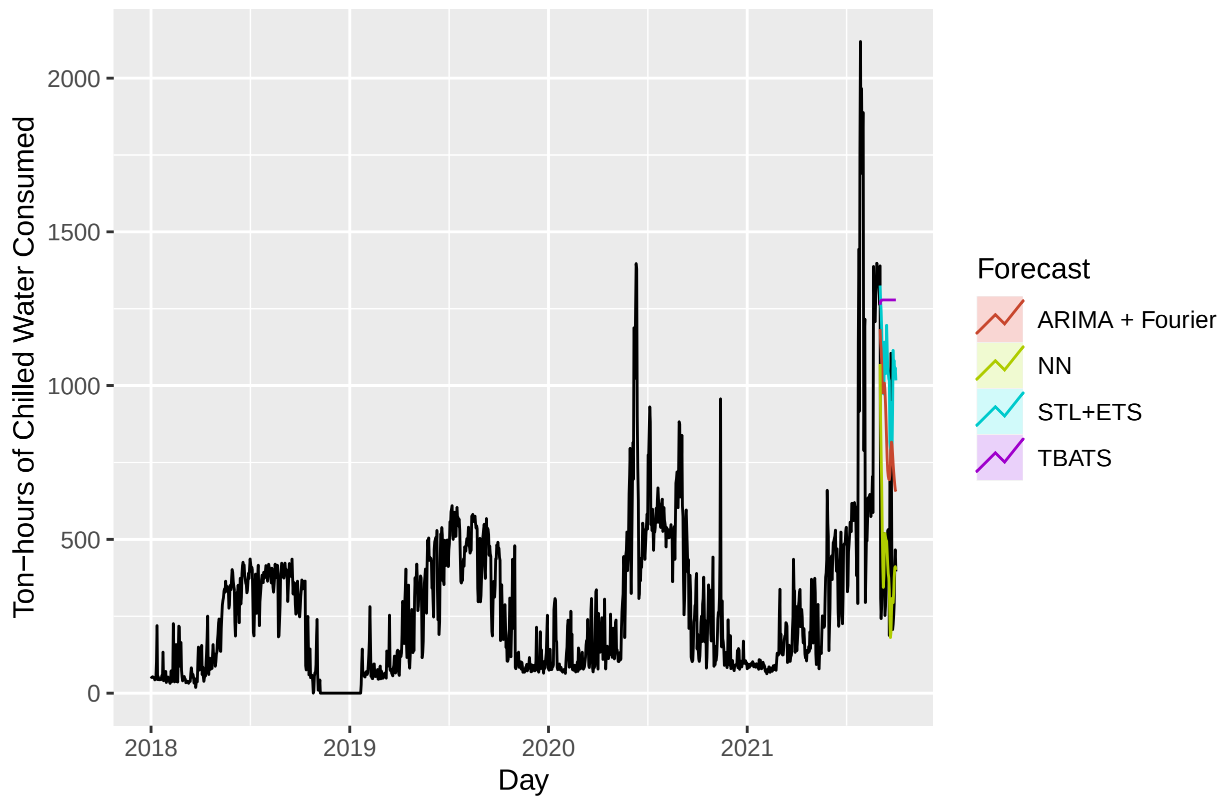
```
  )
row.names(scores2) <- c("STL+ETS", "ARIMA+Fourier","TBATS","NN")

#choose model with lowest RMSE
best_model_index <- which.min(scores2[,"RMSE"])
cat("The best model by RMSE is:", row.names(scores2[best_model_index,]))
```

```
## The best model by RMSE is: NN
```

Table 2: Forecast Accuracy for Daily Ton-hours of Chilled Water Consumption

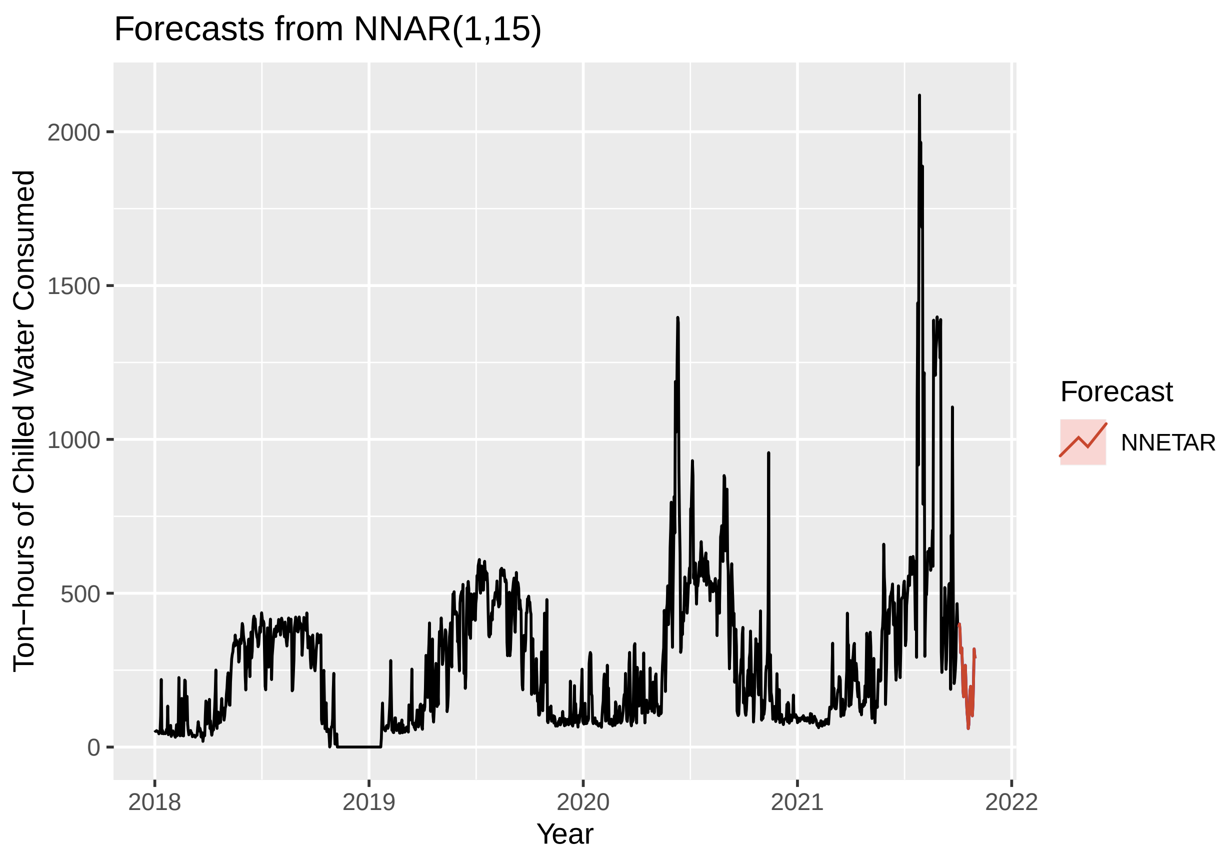
	ME	RMSE	MAE	MPE	MAPE	ACF1	Theil's U
STL+ETS	-600.25226	660.9109	615.2337	-189.14888	190.42519	0.36840	3.27470
ARIMA+Fourier	-404.19353	490.4010	438.8154	-133.87773	136.75607	0.39547	2.37150
TBATS	-831.77198	869.3554	840.1744	-251.70132	252.30600	0.10882	4.23244
NN	3.59866	284.5008	194.2376	-21.29257	48.78501	0.49471	1.24258

```
autoplot(ts_chw_daily) +
  autolayer(ETS_fit2, PI=FALSE, series="STL+ETS") +
  autolayer(ARIMA_Four_for3, PI=FALSE, series="ARIMA + Fourier") +
  autolayer(TBATS_for2,PI=FALSE, series="TBATS") +
  autolayer(NN_for3,PI=FALSE,series="NN") +
  xlab("Day") + ylab("Ton-hours of Chilled Water Consumed") +
  guides(colour=guide_legend(title="Forecast"))
```



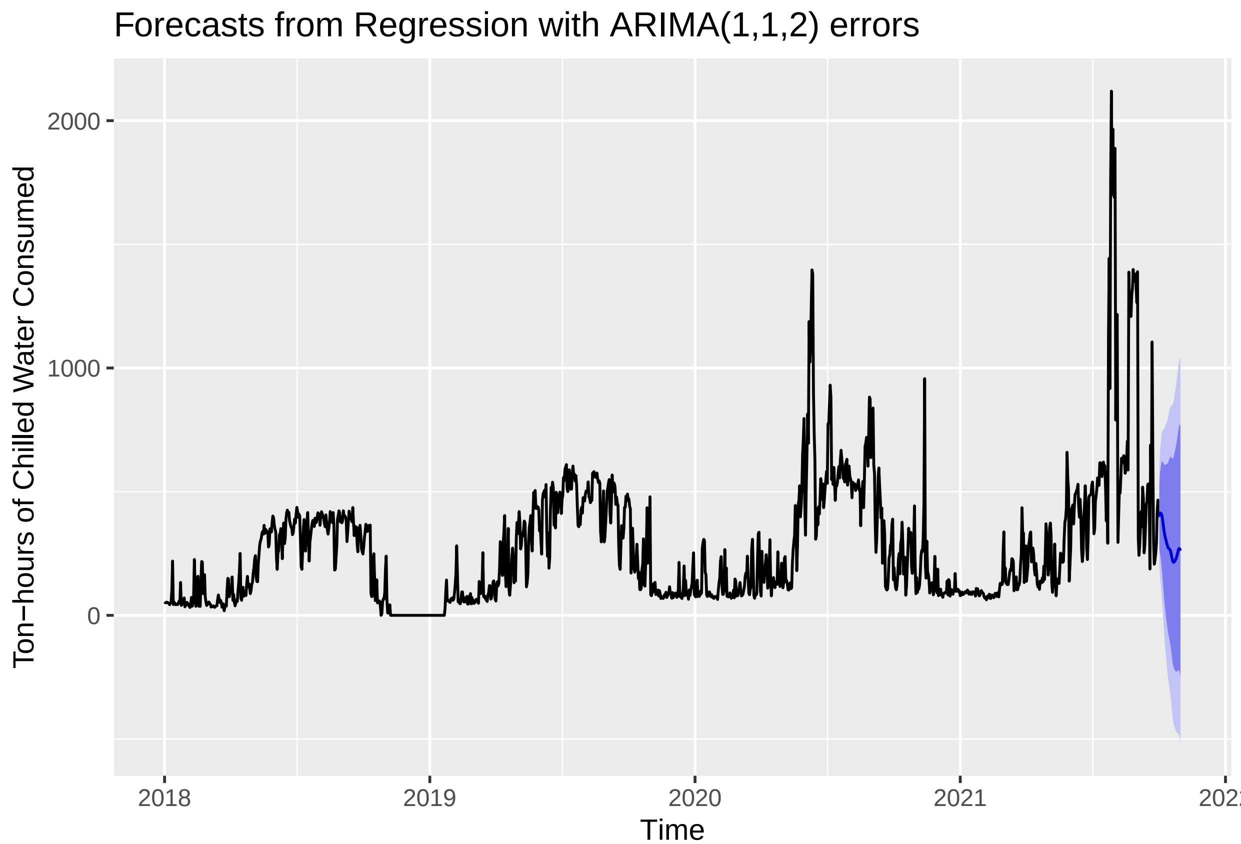
##Forecasting for chilled water

```r
#lowest RMSE
NN_fit4 <- nnetar(ts_chw_daily, p=1,P=0, xreg=fourier(ts_temp_daily, K=c(2,12)))

#NN_for <- forecast(NN_fit, h=30)
NN_for4 <- forecast::forecast(NN_fit4, h=30,xreg=fourier(ts_temp_daily,
                                                         K=c(2,12),h=30))


#Plot forecasting results
autoplot(NN_for4) +
  autolayer(NN_for4, series="NNETAR",PI=FALSE) +
  xlab("Year") + ylab("Ton-hours of Chilled Water Consumed") +
  guides(colour=guide_legend(title="Forecast"))
```

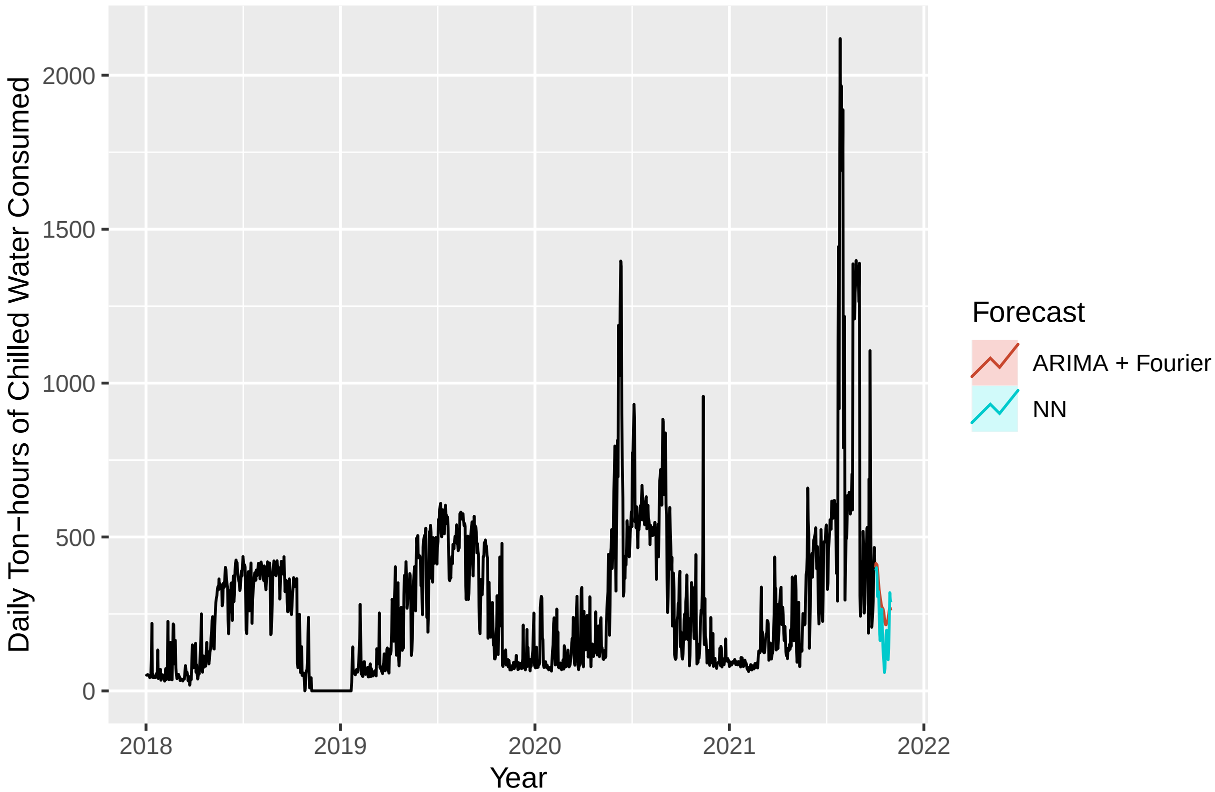



```r
ARIMA_Four_fit4 <- auto.arima(ts_chw_daily,
                              seasonal=FALSE,
                              xreg=fourier(ts_temp_daily,
                                           K=c(2,12))
) #use a log transformation (lambda=0) in the `auto.arima()` to ensure the forecasts and prediction int

#Forecast with ARIMA fit
#also need to specify h for fourier terms
ARIMA_Four_for4 <- forecast::forecast(ARIMA_Four_fit4,
                                      xreg=fourier(ts_temp_daily,
                                                   K=c(2,12),
                                                   h=30),
                                      h=30
```

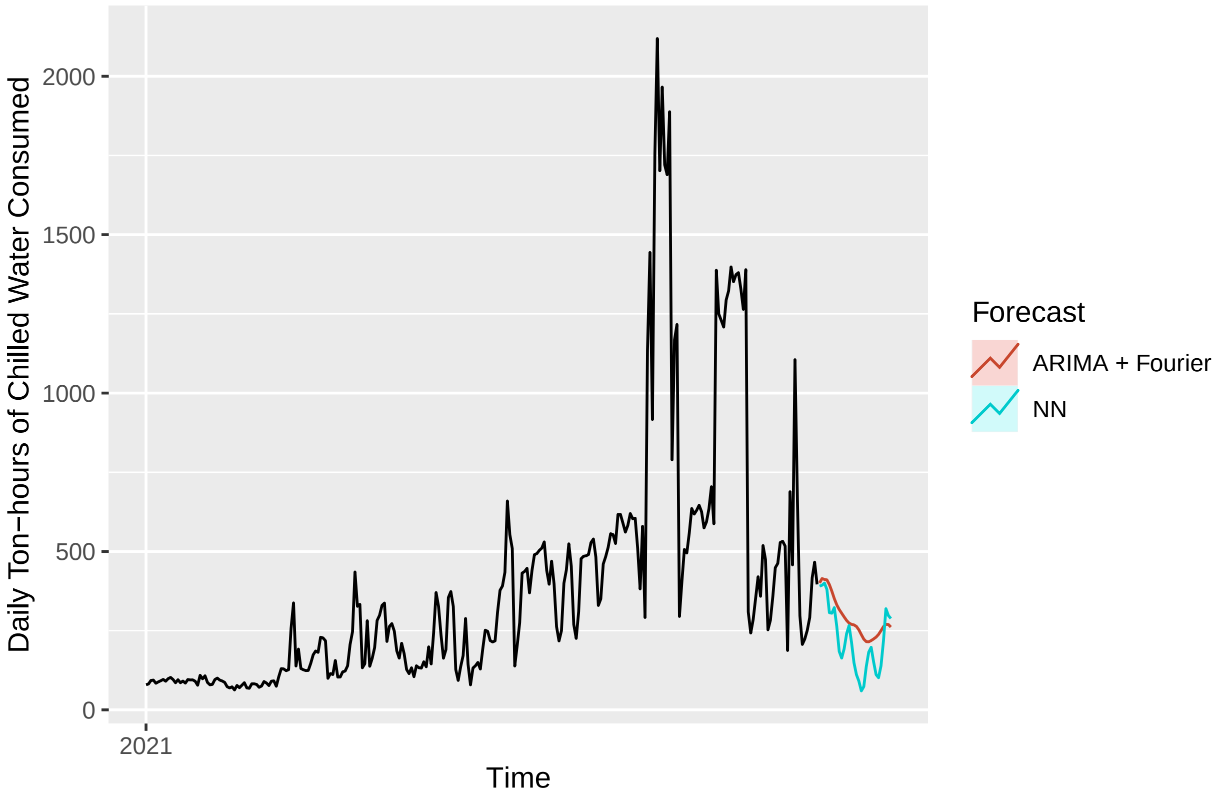
```
                                      )

#Plot forecasting results
autoplot(ARIMA_Four_for4) + ylab("Ton-hours of Chilled Water Consumed")
```




```
autoplot(ts_chw_daily) +
  autolayer(ARIMA_Four_for4, PI=FALSE, series="ARIMA + Fourier") +
  autolayer(NN_for4,PI=FALSE,series="NN") +
  xlab("Year") + ylab("Daily Ton-hours of Chilled Water Consumed") +
  guides(colour=guide_legend(title="Forecast"))
```




```r
autoplot(window(ts_chw_daily,start=2021)) +
  autolayer(ARIMA_Four_for4, PI=FALSE, series="ARIMA + Fourier") +
  autolayer(NN_for4,PI=FALSE, series="NN") +
  ylab("Daily Ton-hours of Chilled Water Consumed") +
  guides(colour=guide_legend(title="Forecast"))
```


Daily Ton−hours of Chilled Water Consumed
2000
1500
1000
500
0
2021
Time
Forecast
ARIMA + Fourier
NN