ENV797_Competition_LinPace

Repository: https://github.com/sp636/LinPace_TSA_competition.git

Samantha Pace, Will Lin

2024-04-25

```
#install.packages("readxl")
#Install library
library(lubridate)
library(ggplot2)
library(forecast)
library(Kendall)
library(tseries)
library(outliers)
library(tidyverse)
library(smooth)
library(zoo)
library(kableExtra)
library(readxl)
```

```
getwd()
```

```
## [1] "/Users/lzh/Desktop/LinPace_TSA_competition"
```

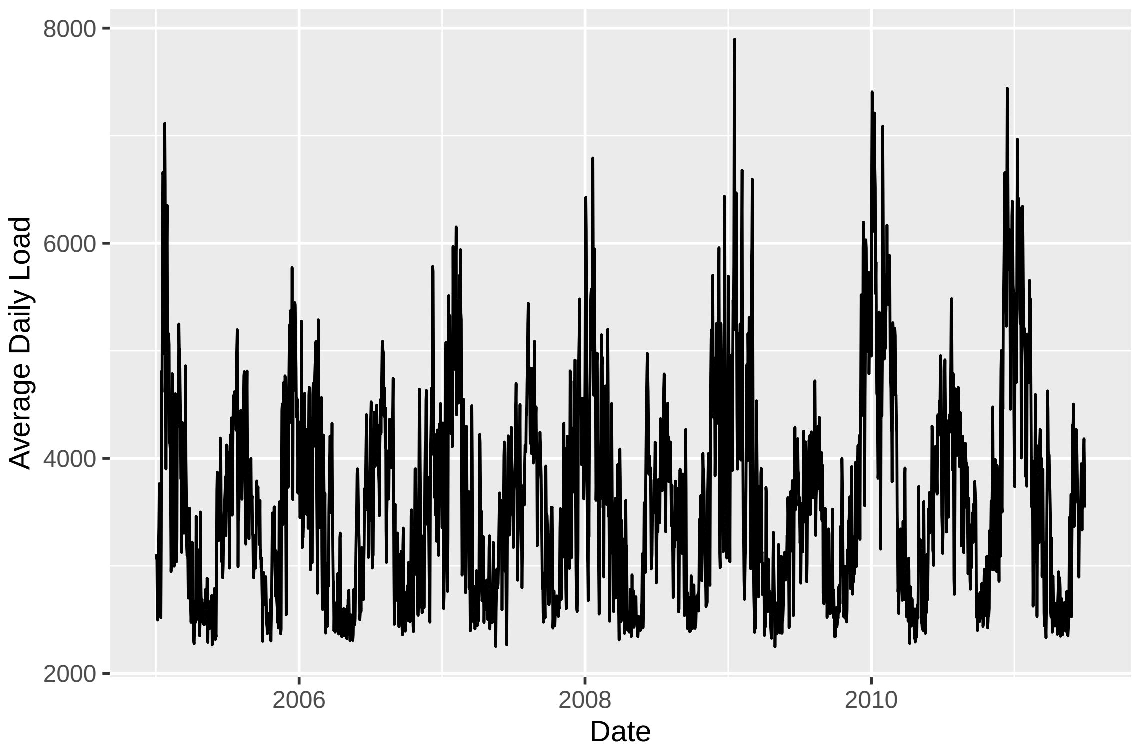
```
#Import datasets
load <- read_excel("./Data/load.xlsx")
hum <- read_excel("./Data/relative_humidity.xlsx")
temp <- read_excel("./Data/temperature.xlsx")
```

Competition Part 1

```
#Date & time objects conversion
#load
#The dataset "load" use hour as the columns and date as the rows.
#Adjust the dataset to be more suitable to convert into time series dataset.
load <- pivot_longer(load, cols = starts_with("h"), names_to = "hour",
                     values_to = "load")
#Convert hour object as numeric
load$hour <- as.numeric(gsub("h", "", load$hour))
#Fix date columns and select needed columns
load <- load %>%
  mutate(Date = ymd(date)) %>%
```

```r
  mutate(Year = year(date),
          Month = month(date),
          Day = day(date),
          Hour = hour) %>%
  select(Date, Year, Month, Day, Hour, load)

#relative_humidity
#Fix date columns and select needed columns
hum <- hum %>%
  mutate(Date = ymd(date)) %>%
  mutate(Year = year(date),
         Month = month(date),
         Day = day(date),
         Hour = hr) %>%
  select(Date, Year, Month, Day, Hour, rh_ws1:rh_ws28)

#temperature
#Fix date columns and select needed columns
temp <- temp %>%
  mutate(Date = ymd(date)) %>%
  mutate(Year = year(date),
         Month = month(date),
         Day = day(date),
         Hour = hr) %>%
  select(Date, Year, Month, Day, Hour, t_ws1:t_ws28)
```

```r
#Creating data frames with daily observations
#load
load_daily <- load %>%
  filter(!is.na(load)) %>%
  group_by(Date,Year,Month,Day) %>%
  summarise( daily_mean_load = mean(load))

ggplot(load_daily, aes(x=Date,y=daily_mean_load)) +
  geom_line() +
  ylab("Average Daily Load")
```



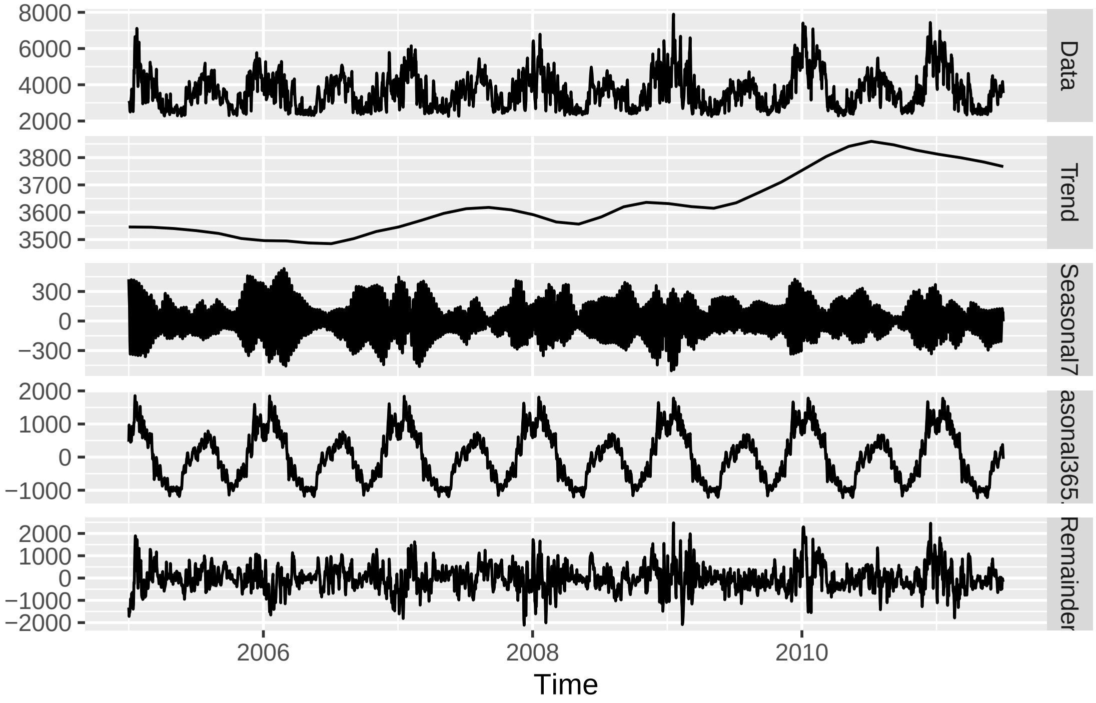
```
summary(load_daily$daily_mean_load)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    2247    2798    3506    3629    4211    7897 
```

```
#relative_humidity
hum_daily <- hum %>%
  group_by(Date,Year,Month,Day) %>%
  summarise(daily_mean_hum = mean(c_across(rh_ws1:rh_ws28)))

#temperature
temp_daily <- temp %>%
  group_by(Date,Year,Month,Day) %>%
  summarise(daily_mean_temp = mean(c_across(t_ws1:t_ws28))) %>%
  na.omit()
```

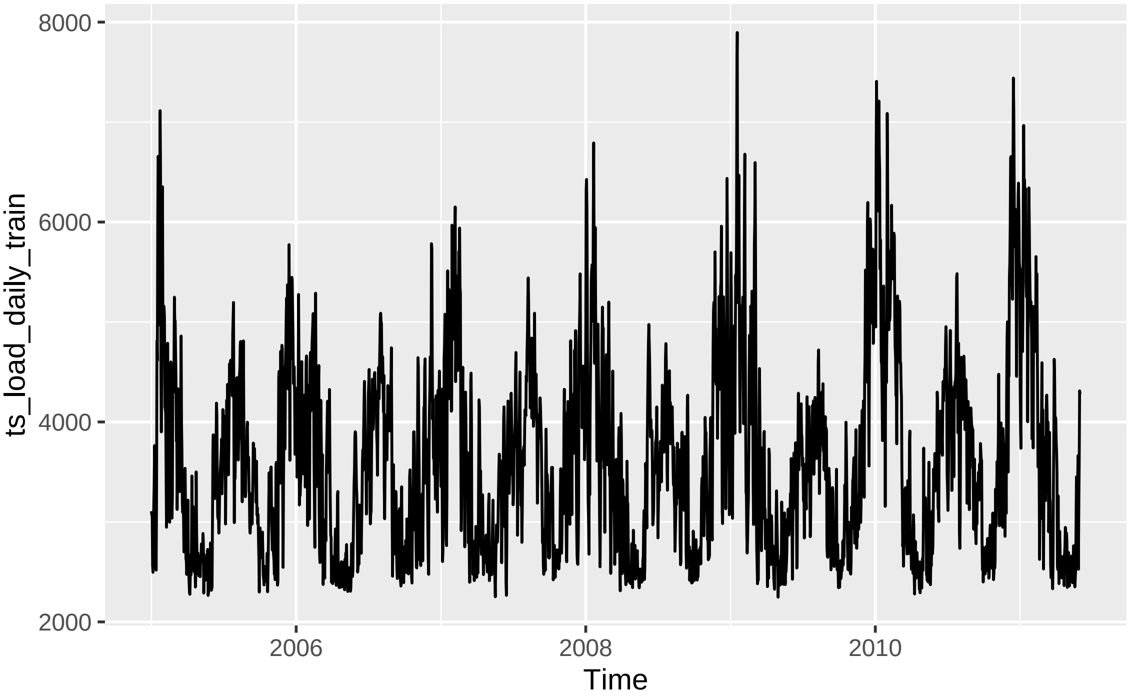
```
#Transform daily dataframes into time series objects
ts_load_daily <- msts(load_daily$daily_mean_load,
                           seasonal.periods =c(7,365.25),
                           start=c(2005,1,1))

ts_hum_daily <- msts(hum_daily$daily_mean_hum,
                           seasonal.periods =c(7,365.25),
                           start=c(2005,1,1))

ts_temp_daily <- msts(temp_daily$daily_mean_temp,
                           seasonal.periods =c(7,365.25),
                           start=c(2005,1,1))
```

```r
# Decompose time series objects
ts_load_daily %>% mstl() %>%
  autoplot()
```

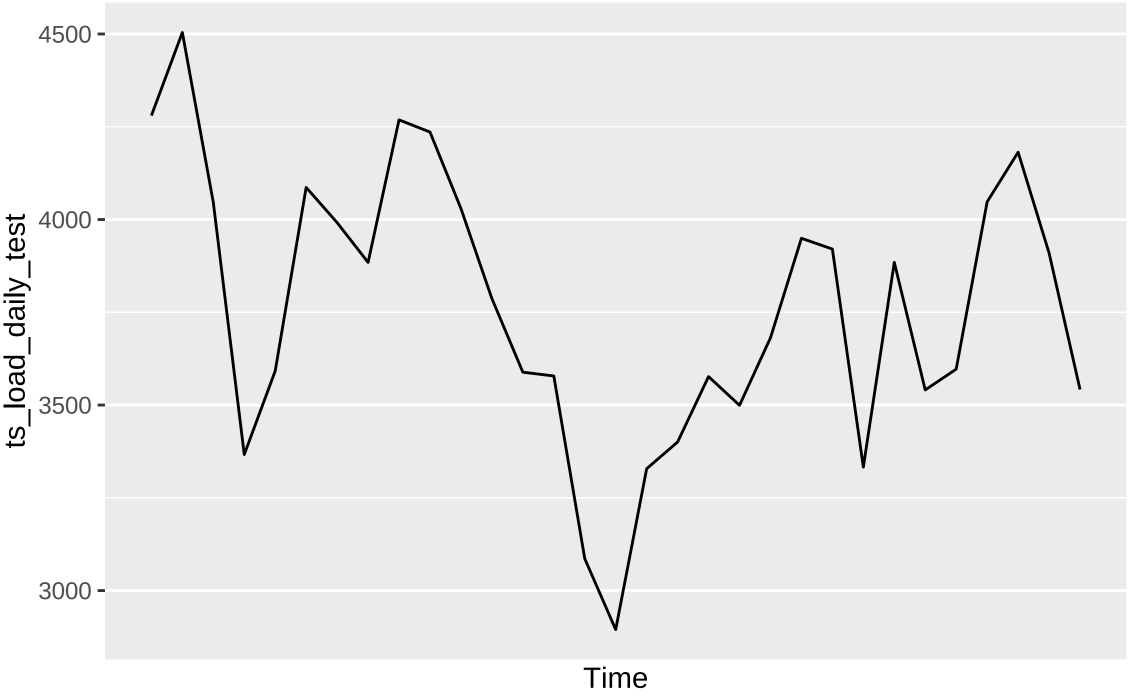


Competition Part 2

```r
#create the training subset considering period 01/01/2005-12/31/2009
total_days = length(ts_load_daily)
days_june_2011 = 30
days_july_2011 = 31

ts_load_daily_train <- subset(ts_load_daily,
                              end = total_days-days_june_2011)

#create testing subset
ts_load_daily_test <- subset(ts_load_daily,
                                  start = total_days-days_june_2011)

#Creat dataframe subset for plot fitting comparison
ts_load_daily_fit_july11 <- subset(ts_load_daily,
                                end = total_days+days_july_2011)


autoplot(ts_load_daily_train)
```

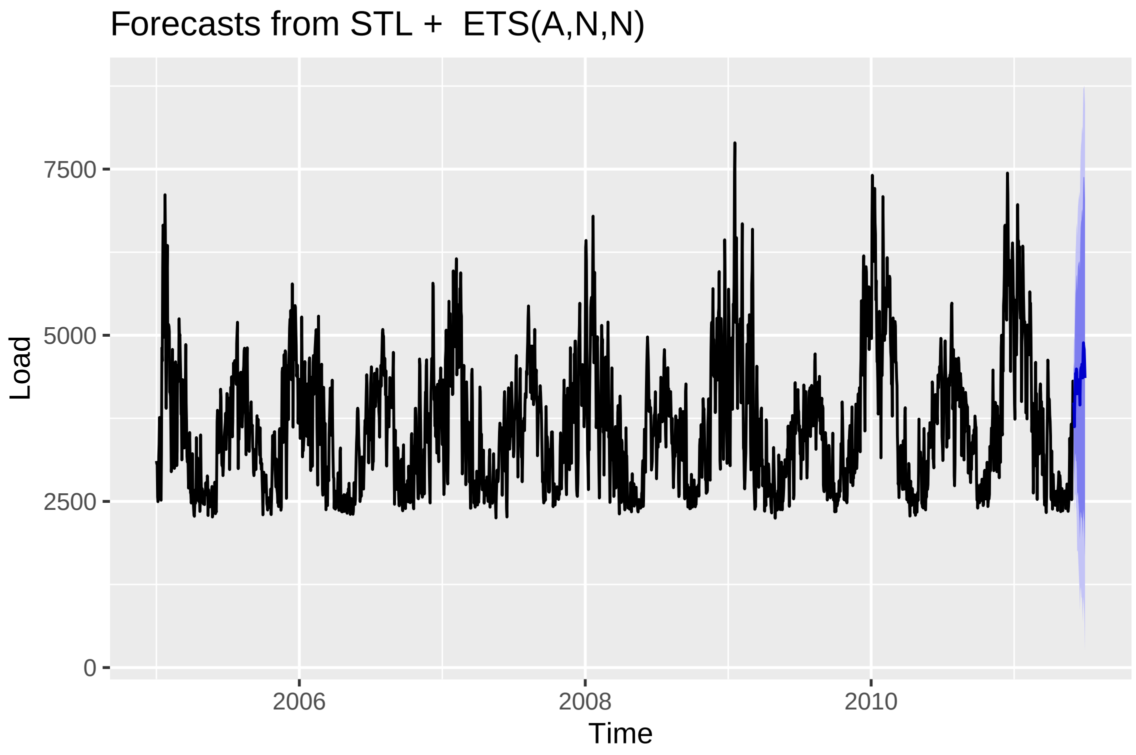


```
autoplot(ts_load_daily_test)
```

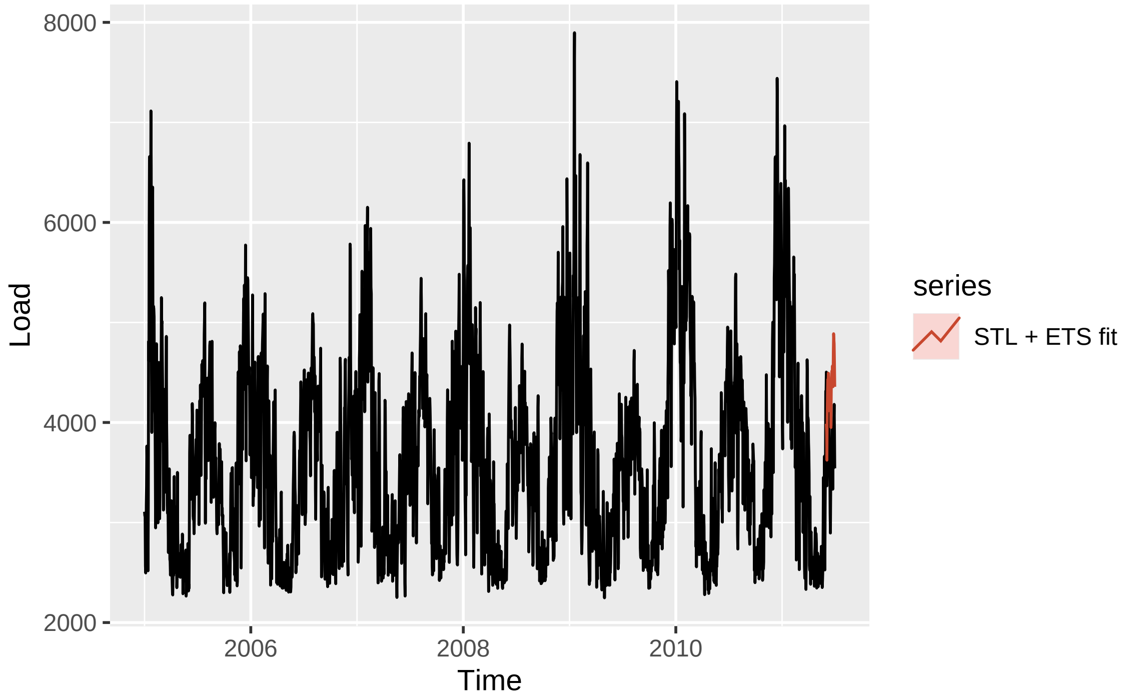


```
#Hum Train & Test
ts_hum_daily_train <- subset(ts_hum_daily,
                              end = total_days-days_june_2011)

#create testing subset
ts_hum_daily_test <- subset(ts_hum_daily,
                                  start = total_days-days_june_2011)


#Temp Train & Test
ts_temp_daily_train <- subset(ts_temp_daily,
                              end = total_days-days_june_2011)

#ts_temp_daily_train <- subset()
#create testing subset
ts_temp_daily_test <- subset(ts_temp_daily,
                                  start = total_days-days_june_2011)
```

```
#Model 1: STL + ETS
#Fit and forecast STL + ETS model to data

#STL on train (all but june 2011)
ETS_fit_train <-  stlf(ts_load_daily_train,h=30) # changed to 30 for 30 days of june

#Plot foresting results
autoplot(ETS_fit_train) + ylab("Load")
```

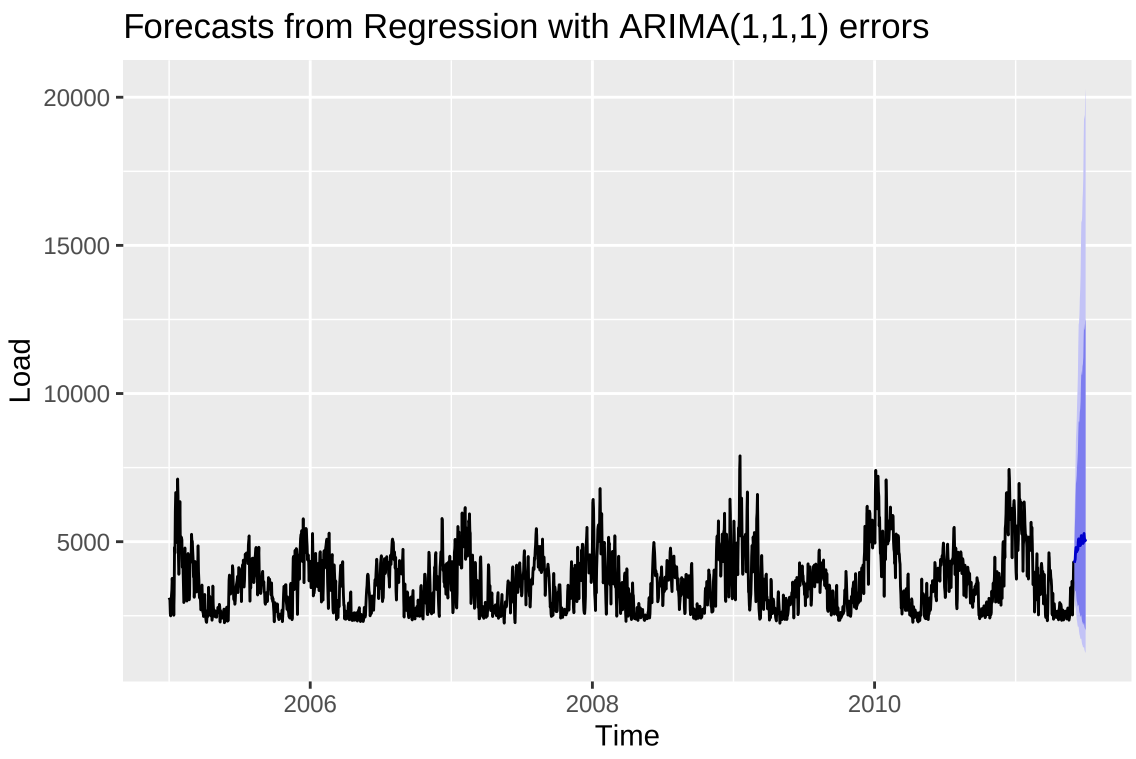


```
#Plot model + observed data
autoplot(ts_load_daily) +
  autolayer(ETS_fit_train, series="STL + ETS fit",PI=FALSE) +
  ylab("Load")
```

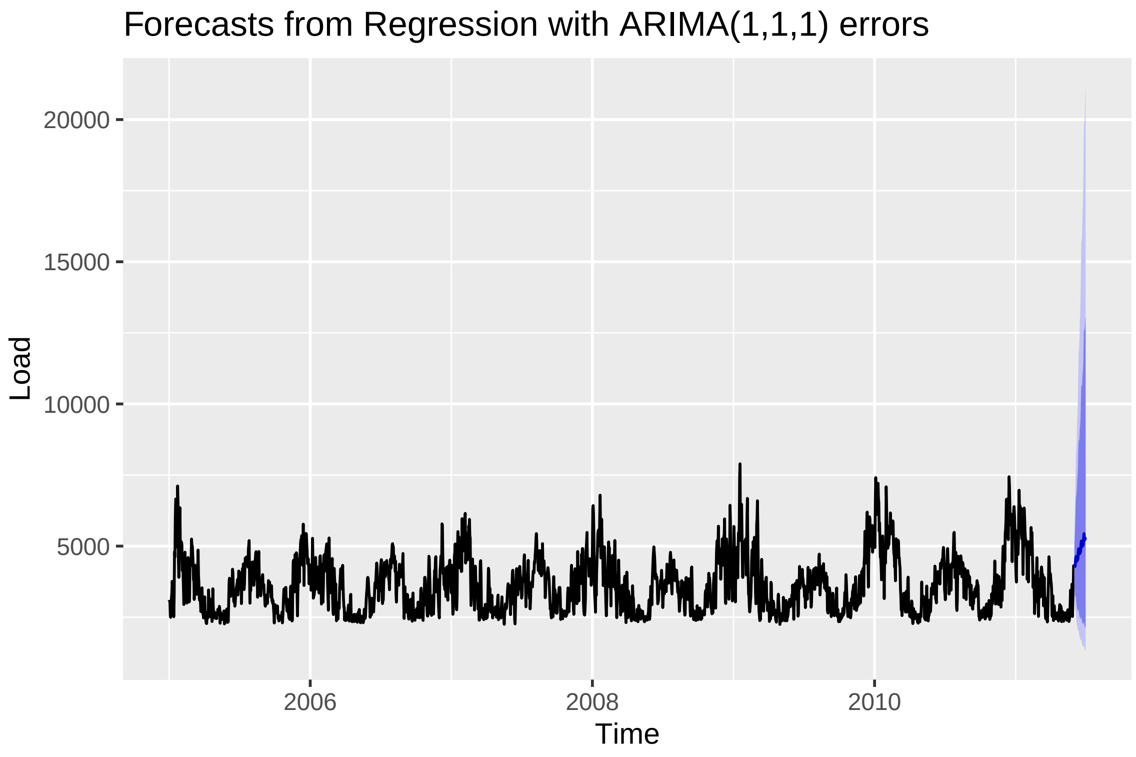


```
#Model 2: ARIMA + FOURIER terms
#Fit arima model with fourier terms as exogenous regressors
ARIMA_Four_fit <- auto.arima(ts_load_daily_train,
                             seasonal=FALSE,
                             lambda=0,
                             xreg=fourier(ts_load_daily_train,
                                          K=c(2,12))
                             )
#Forecast with ARIMA fit
ARIMA_Four_for <- forecast(ARIMA_Four_fit,
                           xreg=fourier(ts_load_daily_train,
                                        K=c(2,12),
                                        h=30),
                           h=30
                           )

#Fit arima model with fourier terms as exogenous regressors. version 2
ARIMA_Four_fit2 <- auto.arima(ts_load_daily_train,
                             seasonal=FALSE,
                             lambda=0,
                             xreg=fourier(ts_load_daily_train,
                                          K=c(2,2))
                             )
```

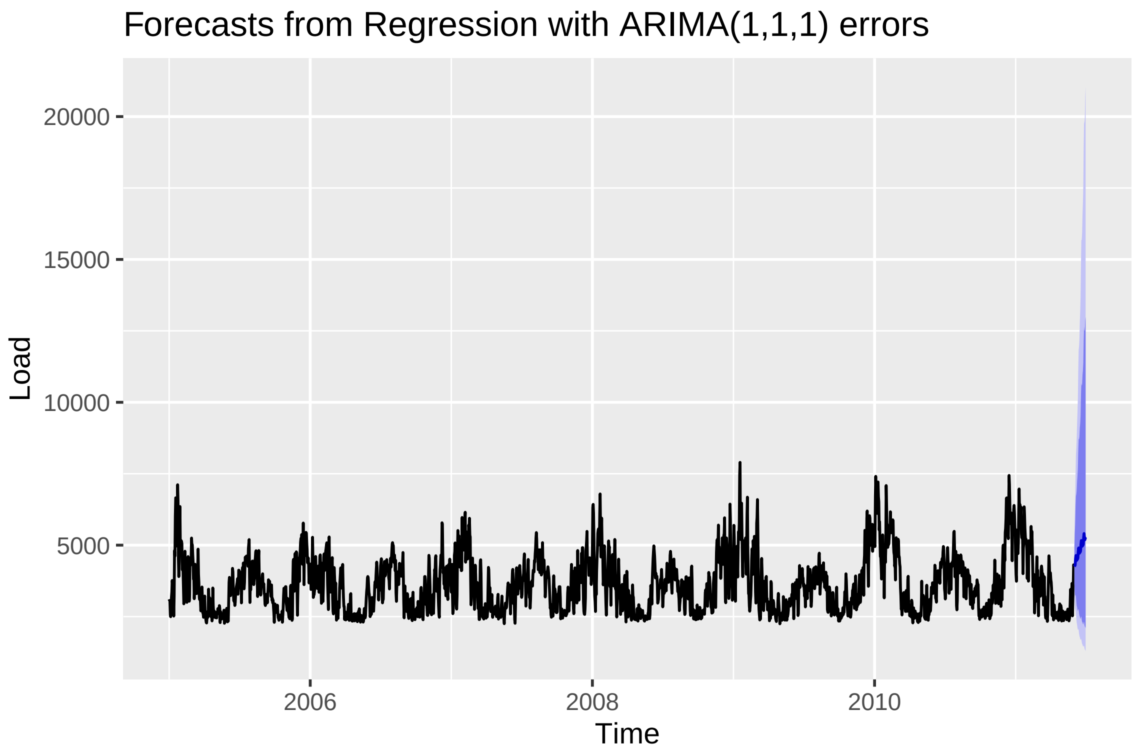
```
ARIMA_Four_for2 <- forecast(ARIMA_Four_fit2,
                           xreg=fourier(ts_load_daily_train,
                                        K=c(2,2),
                                        h=30),
                           h=30
                           )

#Fit arima model with fourier terms as exogenous regressors. version 3
ARIMA_Four_fit3 <- auto.arima(ts_load_daily_train,
                             seasonal=FALSE,
                             lambda=0,
                             xreg=fourier(ts_load_daily_train,
                                          K=c(2,4))
                             )

ARIMA_Four_for3 <- forecast(ARIMA_Four_fit3,
                           xreg=fourier(ts_load_daily_train,
                                        K=c(2,4),
                                        h=30),
                           h=30
                           )

#Fit arima model with fourier terms as exogenous regressors. version 4
ARIMA_Four_fit4 <- auto.arima(ts_load_daily_train,
                             seasonal=FALSE,
                             lambda=0,
                             xreg=fourier(ts_load_daily_train,
                                          K=c(2,6))
                             )

ARIMA_Four_for4 <- forecast(ARIMA_Four_fit4,
                           xreg=fourier(ts_load_daily_train,
                                        K=c(2,6),
                                        h=30),
                           h=30
                           )

#Plot foresting results
autoplot(ARIMA_Four_for) + ylab("Load")
```



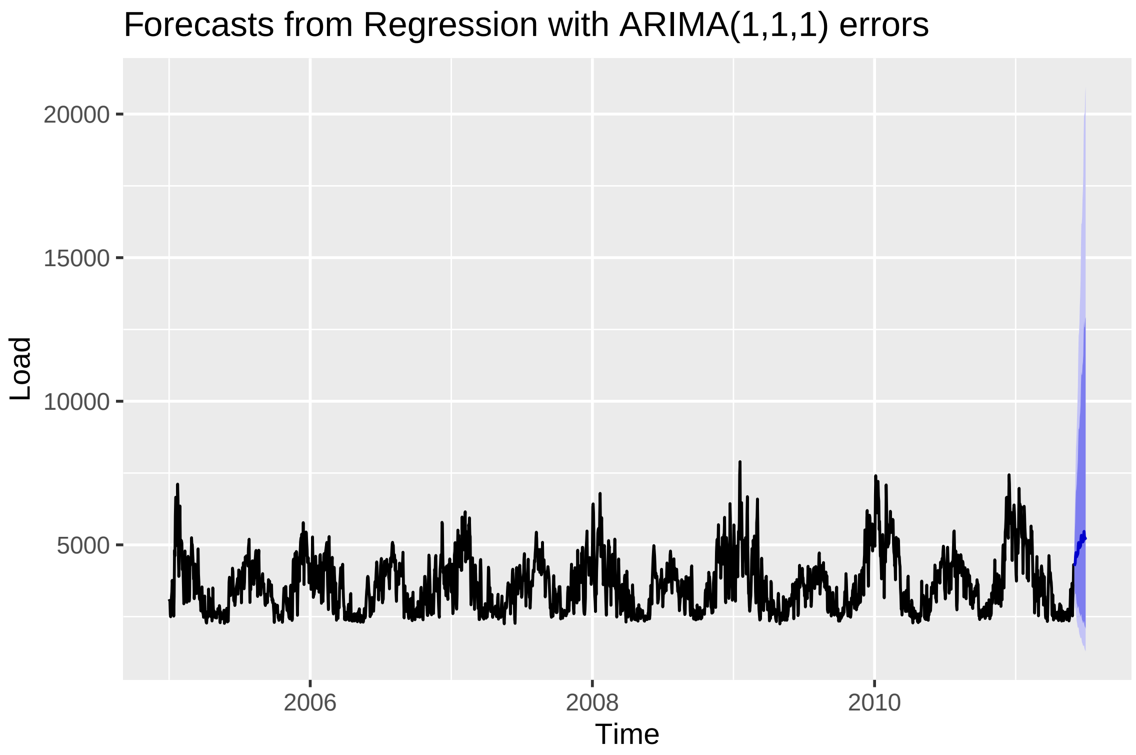
```r
autoplot(ARIMA_Four_for2) + ylab("Load")
```



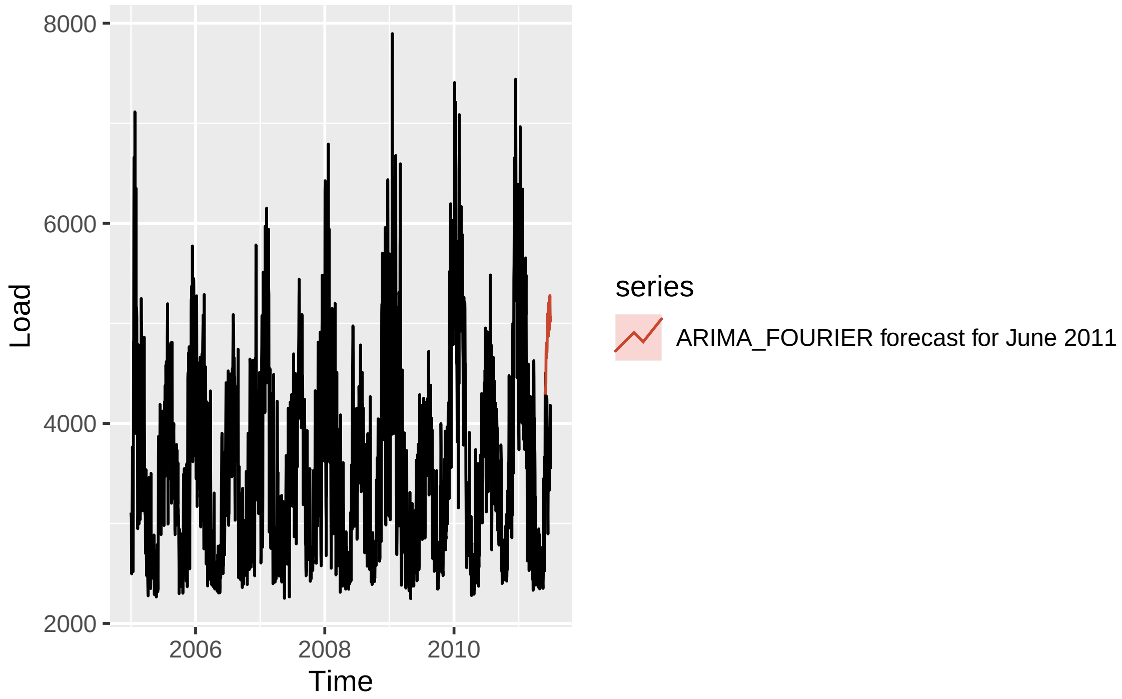
```
autoplot(ARIMA_Four_for3) + ylab("Load")
```



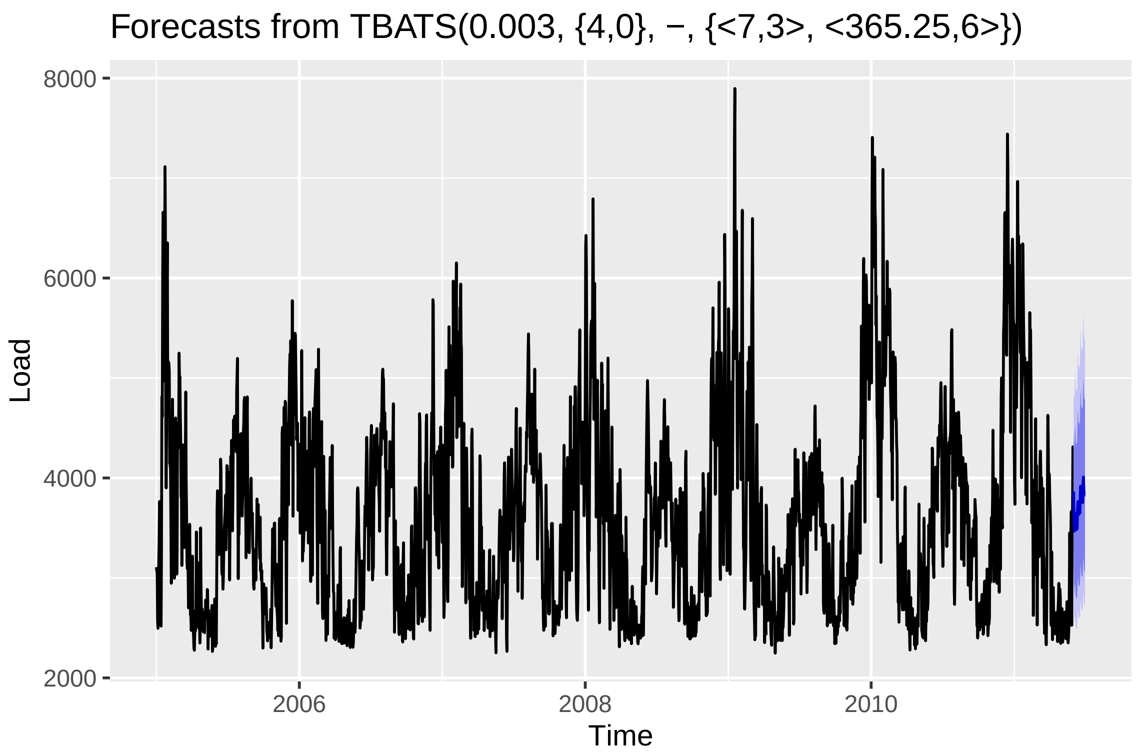
```
autoplot(ARIMA_Four_for4) + ylab("Load")
```



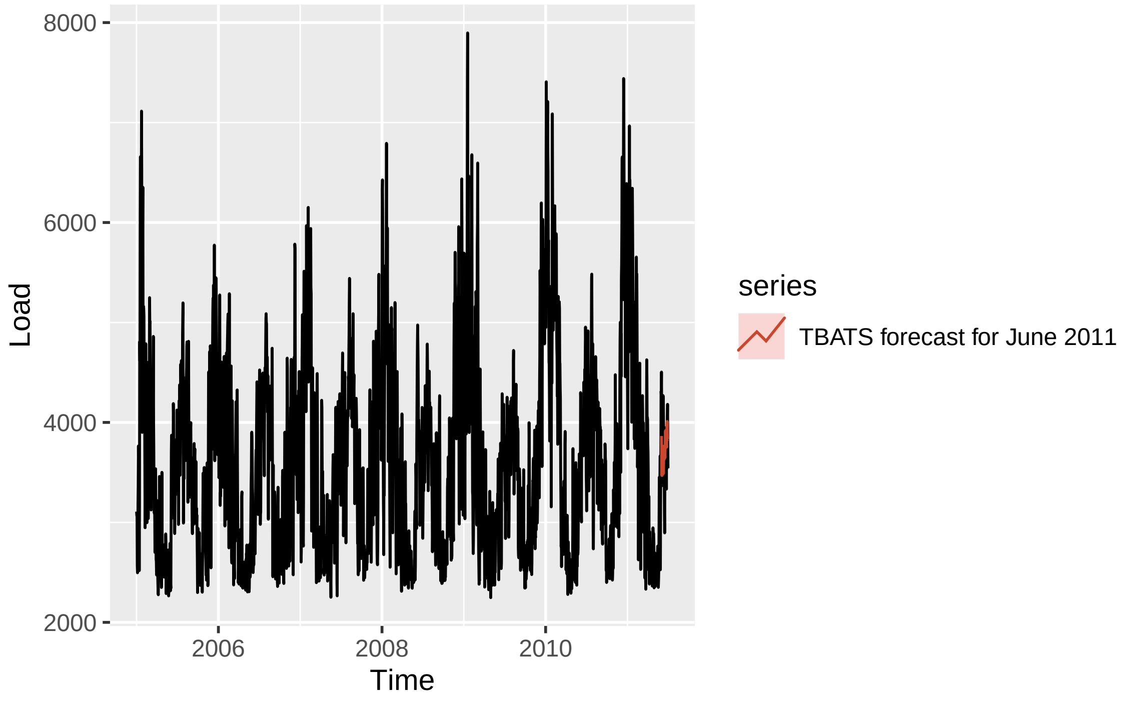
```
#Plot model + observed data
autoplot(ts_load_daily) +
  autolayer(ARIMA_Four_for, series="ARIMA_FOURIER forecast for June 2011",
            PI=FALSE) +
  ylab("Load")
```



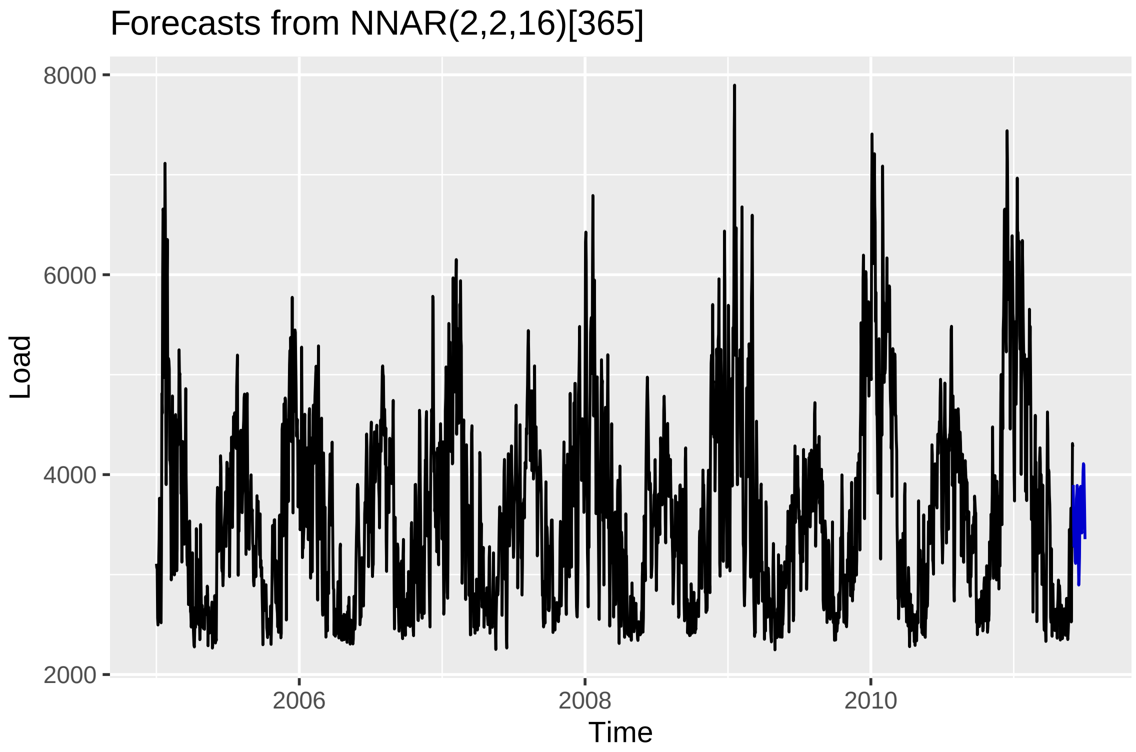
```
#Model 3: TBATS
TBATS_fit <- tbats(ts_load_daily_train)

TBATS_for <- forecast(TBATS_fit, h=30)

#Plot foresting results
autoplot(TBATS_for) +
  ylab("Load")
```



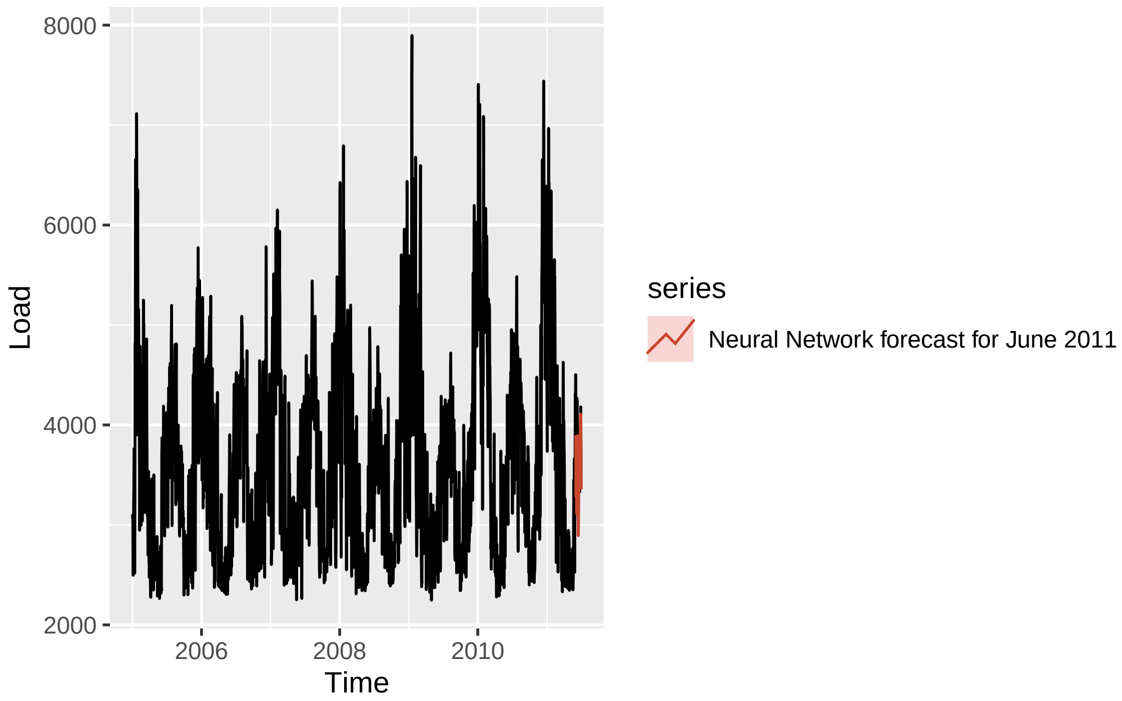
```r
#Plot model + observed data
autoplot(ts_load_daily) +
  autolayer(TBATS_for, series="TBATS forecast for June 2011",PI=FALSE)+
  ylab("Load")
```



```r
# Model 4: Neural Network Time Series Forecasts
# can change the P and p; like ARIMA
NN_fit <- nnetar(ts_load_daily_train,
                 p=2,
                 P=2,
                 xreg=fourier(ts_load_daily_train, K=c(2,12)))

NN_for <- forecast(NN_fit, h=30,xreg=fourier(ts_load_daily_train,
                                         K=c(2,12),h=31))

#Plot foresting results
autoplot(NN_for) +
  ylab("Load")
```



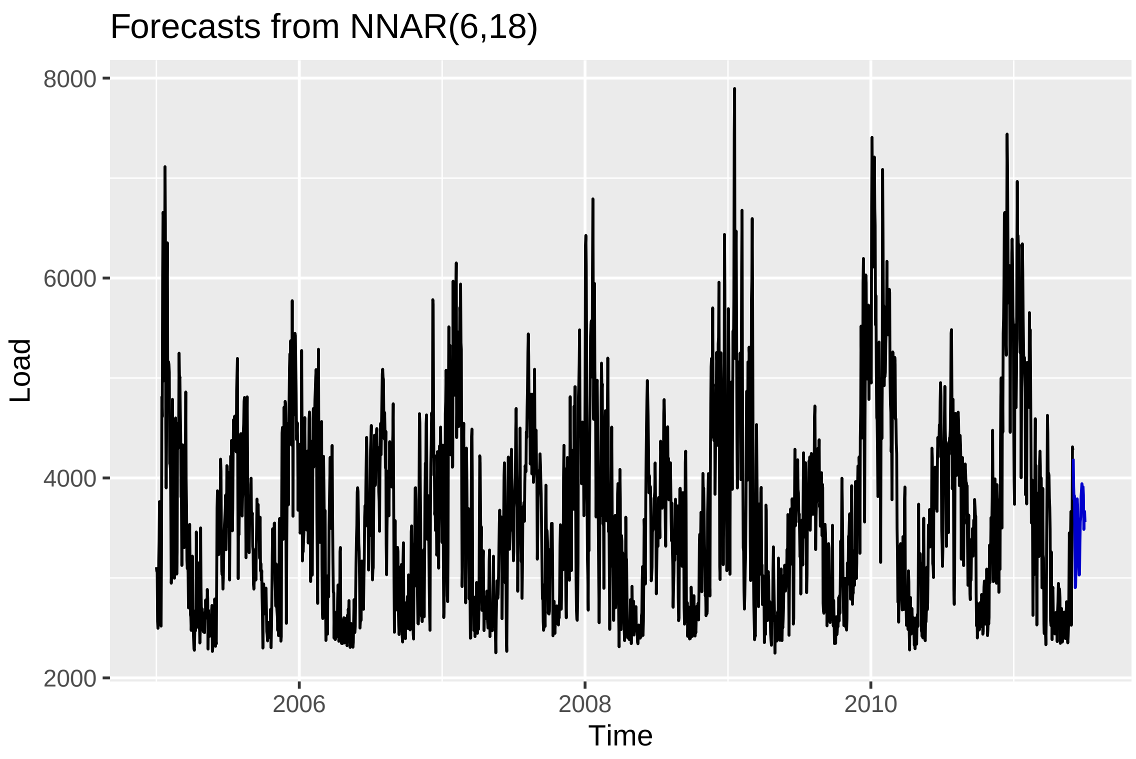
```r
#Plot model + observed data
autoplot(ts_load_daily) +
  autolayer(NN_for, series="Neural Network forecast for June 2011",PI=FALSE)+
  ylab("Load")
```



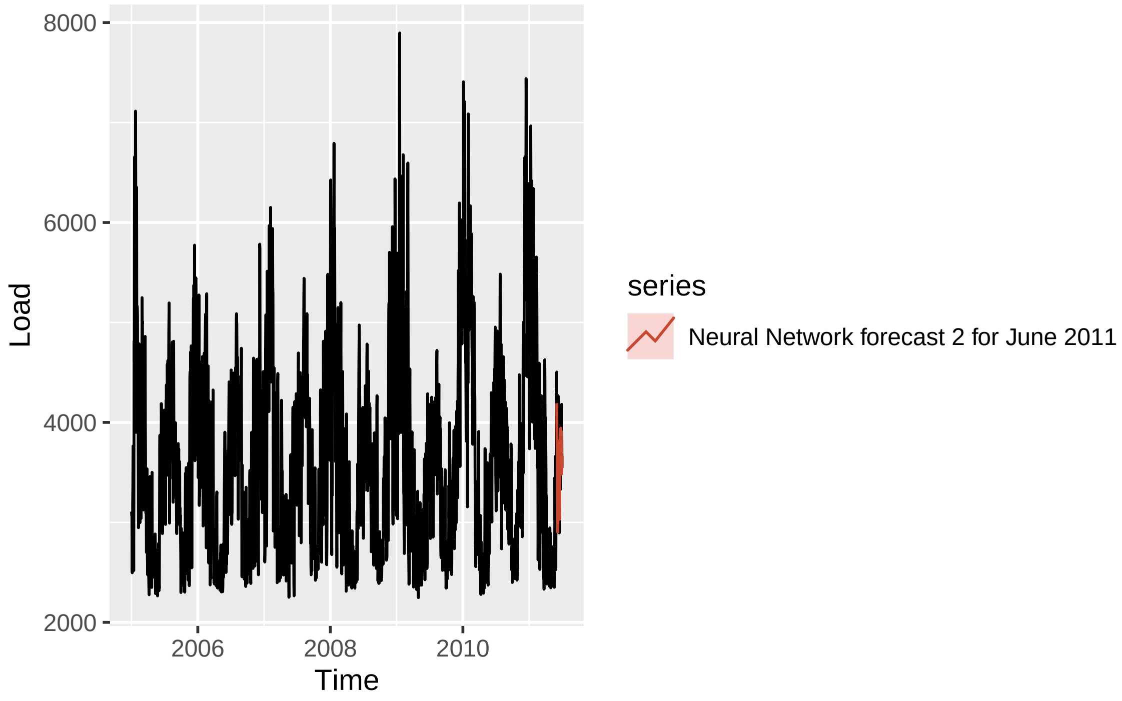
```
# Model 5: Neural Network Time Series Forecasts
# can change the P and p; like ARIMA
NN_fit2 <- nnetar(ts_load_daily_train,
                 p=6,
                 P=10,
                 xreg=fourier(ts_load_daily_train, K=c(2,12)))
```

```
## Warning in nnetar(ts_load_daily_train, p = 6, P = 10, xreg =
## fourier(ts_load_daily_train, : Series too short for seasonal lags
```

```
NN_for2 <- forecast(NN_fit2, h=30,xreg=fourier(ts_load_daily_train,
                                          K=c(2,12),h=31))

#Plot foresting results
autoplot(NN_for2) +
  ylab("Load")
```



```
#Plot model + observed data
autoplot(ts_load_daily) +
  autolayer(NN_for2, series="Neural Network forecast 2 for June 2011",PI=FALSE)+
  ylab("Load")
```



```r
#Check accuracy of the models
#Model 1: STL + ETS
ETS_scores <- accuracy(ETS_fit_train$mean,ts_load_daily_test)

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

#Model: ARIMA + Fourier 2
ARIMA_scores2 <- accuracy(ARIMA_Four_for2$mean,ts_load_daily_test)

#Model: ARIMA + Fourier 3
ARIMA_scores3 <- accuracy(ARIMA_Four_for3$mean,ts_load_daily_test)

#Model: ARIMA + Fourier 4
ARIMA_scores4 <- accuracy(ARIMA_Four_for4$mean,ts_load_daily_test)

# Model 3:  TBATS
TBATS_scores <- accuracy(TBATS_for$mean,ts_load_daily_test)

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

# Model 5:  Neural Network 2
NN_scores2 <- accuracy(NN_for2$mean,ts_load_daily_test)
```

Comparing scores; TBATS is the best with 9.17 score

```r
#Compare performance metrics
scores_fit <- as.data.frame(
  rbind(ETS_scores, ARIMA_scores, ARIMA_scores2, ARIMA_scores3, ARIMA_scores4,
        TBATS_scores, NN_scores, NN_scores2)
  )
row.names(scores_fit) <- c("STL+ETS", "ARIMA+Fourier", "ARIMA+Fourier2",
                           "ARIMA+Fourier3", "ARIMA+Fourier4", "TBATS","NN",
                           "NN2")

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

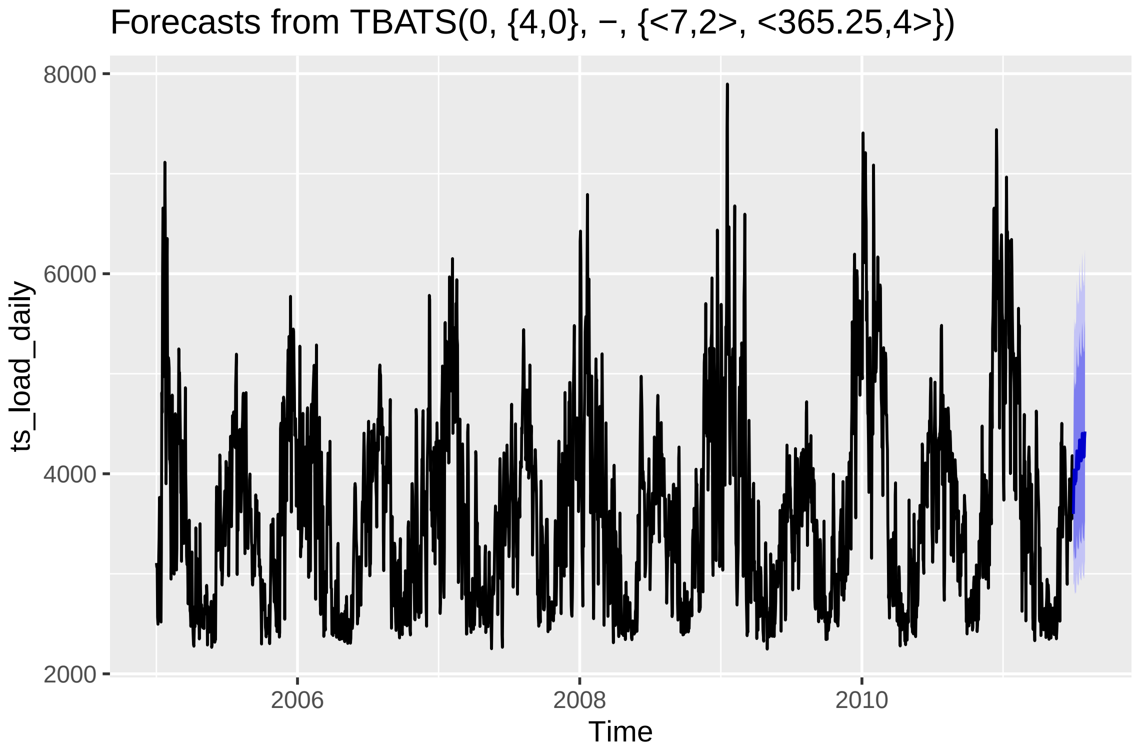
```
## The best model to forecast Summer by MAPE is: NN
```

```r
#Compare results in a table format
kbl(scores_fit,
      caption = "Forecast Accuracy for Daily Load",
      digits = array(5,ncol(scores_fit))) %>%
  kable_styling(full_width = FALSE, position = "center",
                latex_options = "hold_position") %>%
  #highlight model with lowest MAPE
  kable_styling(latex_options="striped",
                stripe_index = which.min(scores_fit[,"MAPE"]))
```

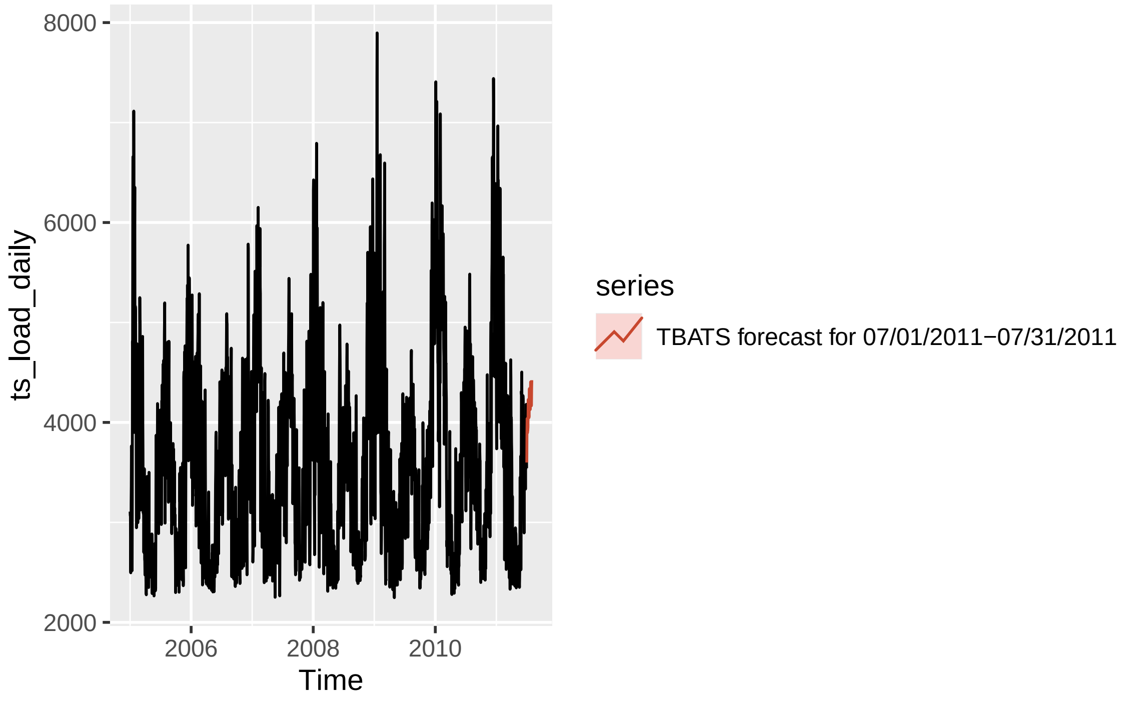
Table 1: Forecast Accuracy for Daily Load

	ME	RMSE	MAE	MPE	MAPE	ACF1	Theil's U
STL+ETS	-609.65571	736.2245	653.4375	-17.25090	18.24626	0.55588	2.35369
ARIMA+Fourier	-1137.92574	1240.3155	1152.3575	-31.83072	32.15113	0.61229	4.00568
ARIMA+Fourier2	-1099.09897	1225.9469	1116.0276	-30.76618	31.14203	0.66791	3.94117
ARIMA+Fourier3	-1086.06318	1210.4212	1102.9134	-30.41115	30.78526	0.66100	3.89341
ARIMA+Fourier4	-1195.42951	1315.6559	1211.1347	-33.42627	33.77496	0.66977	4.24254
TBATS	32.82197	401.3228	341.3879	-0.10145	9.17741	0.61545	1.18744
NN	167.90020	404.6472	309.2715	3.98210	7.95630	0.62495	1.16192
NN2	173.59547	473.1787	358.5093	3.90461	9.41208	0.61518	1.38432

```
# fit TBATS model to whole dataset and predict July 2011 (31 days)
TBATS_fit_all <- tbats(ts_load_daily)

#forecast July
TBATS_for_july2011 <- forecast(TBATS_fit_all, h=31)

#generate df
TBATS.df <- data.frame(TBATS_for_july2011$mean)

write.csv(TBATS.df, file = "Data/Submission2", row.names=F)


#Plot foresting results
autoplot(TBATS_for_july2011)
```

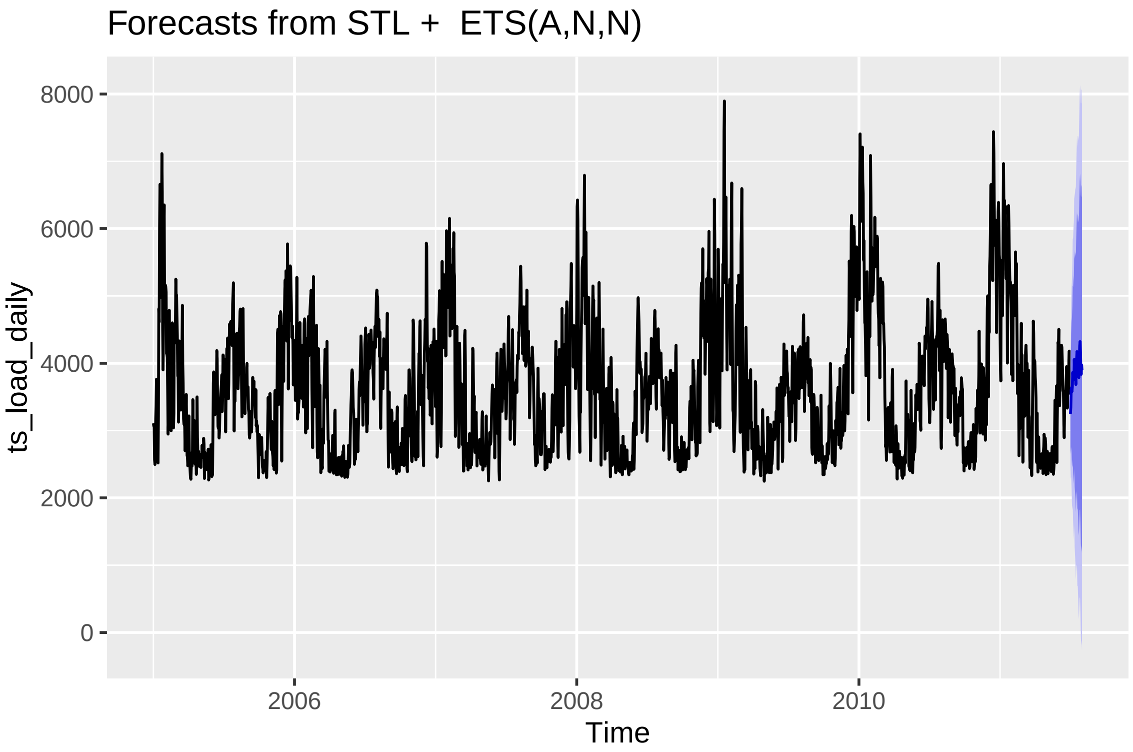


```
#Plot model + observed data
autoplot(ts_load_daily) +
  autolayer(TBATS_for_july2011,
            series="TBATS forecast for 07/01/2011-07/31/2011",PI=FALSE)
```

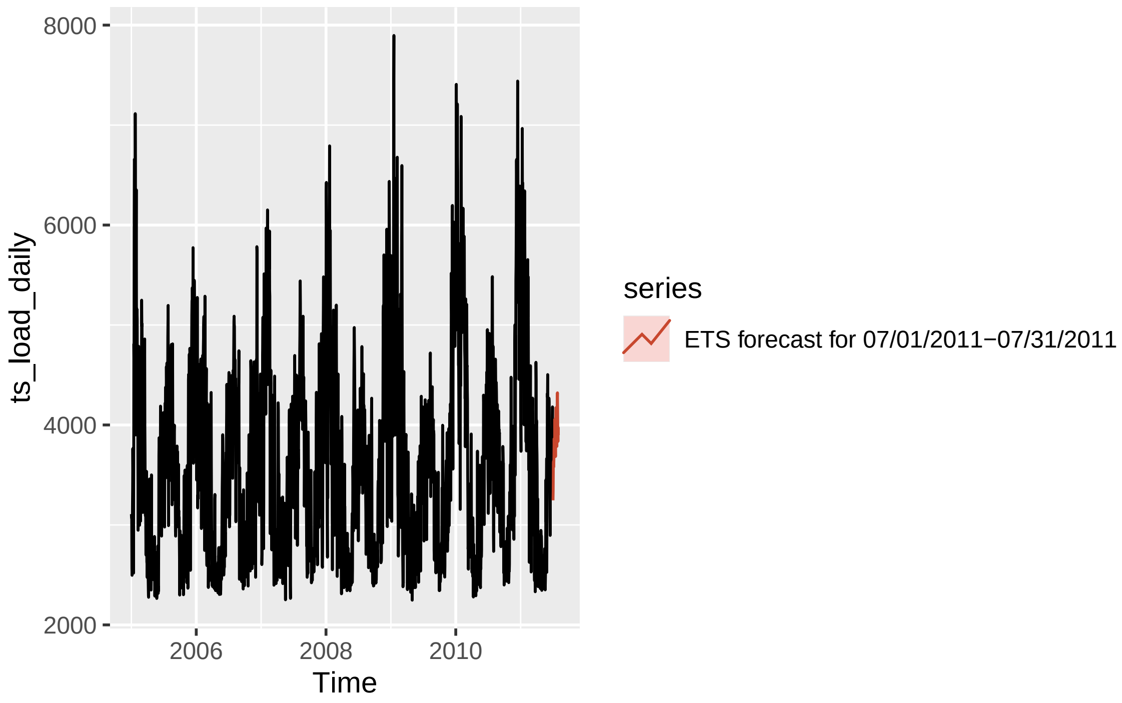


Projecting out all models

```
# fit ETS model to whole dataset and predict July 2011 (31 days)
ETS_fit_all <- stlf(ts_load_daily, h=31)

#Plot foresting results
autoplot(ETS_fit_all)
```

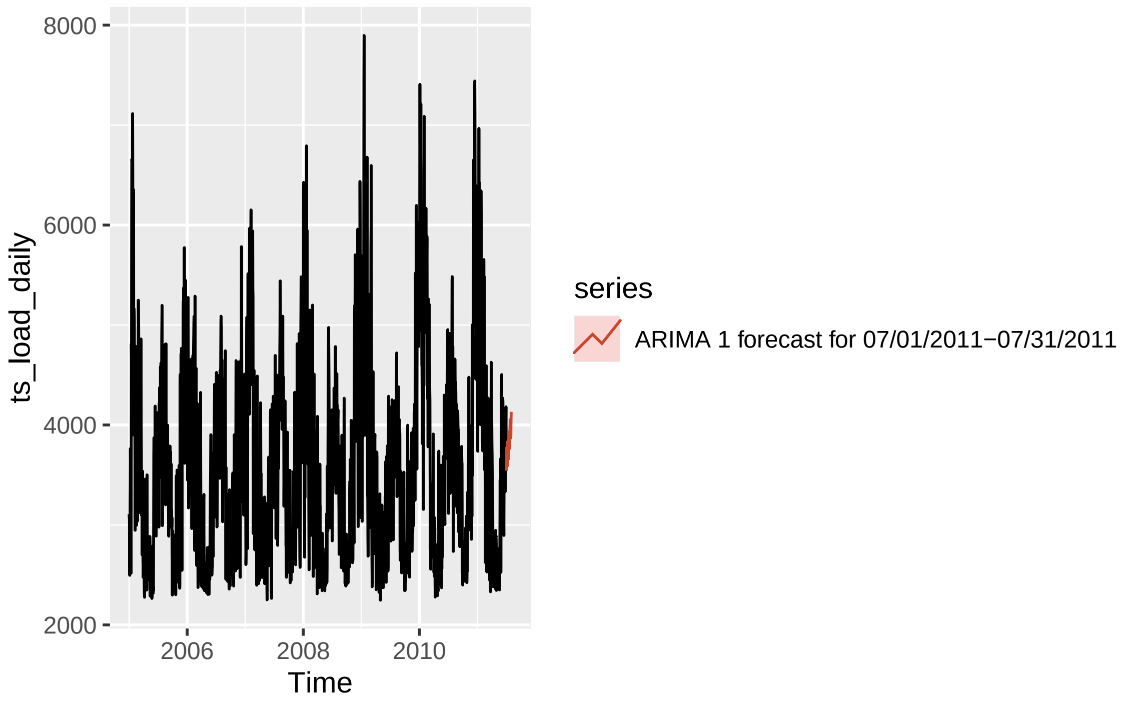


```
#Plot model + observed data
autoplot(ts_load_daily) +
  autolayer(ETS_fit_all, series="ETS forecast for 07/01/2011-07/31/2011",
            PI=FALSE)
```



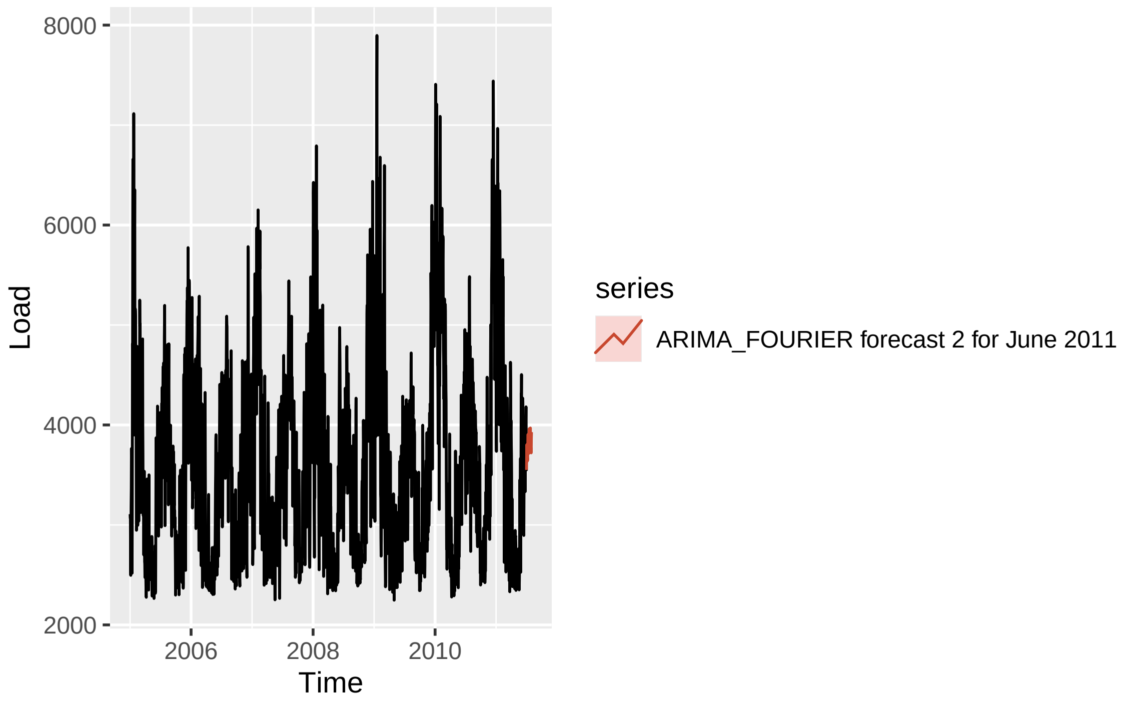
```
# creating df and submission
date <- seq(as.Date("2011-07-01"), as.Date("2011-07-31"), by = 1)
ETS.df <- data.frame(date, ETS_fit_all$mean)
write.csv(ETS.df, file = "./submission02.csv")
```

```
#Model 2: ARIMA + FOURIER terms
#Fit arima model with fourier terms as exogenous regressors
ARIMA_Four_fit_all <- auto.arima(ts_load_daily,
                             seasonal=FALSE,
                             lambda=0,
                             xreg=fourier(ts_load_daily,
                                          K=c(2,12))
                             )
#Forecast with ARIMA fit
ARIMA_Four_for_all <- forecast(ARIMA_Four_fit_all,
                           xreg=fourier(ts_load_daily,
                                        K=c(2,12),
                                        h=31),
                           h=31
                           )
#plot
autoplot(ts_load_daily) +
  autolayer(ARIMA_Four_for_all,
            series="ARIMA 1 forecast for 07/01/2011-07/31/2011",PI=FALSE)
```



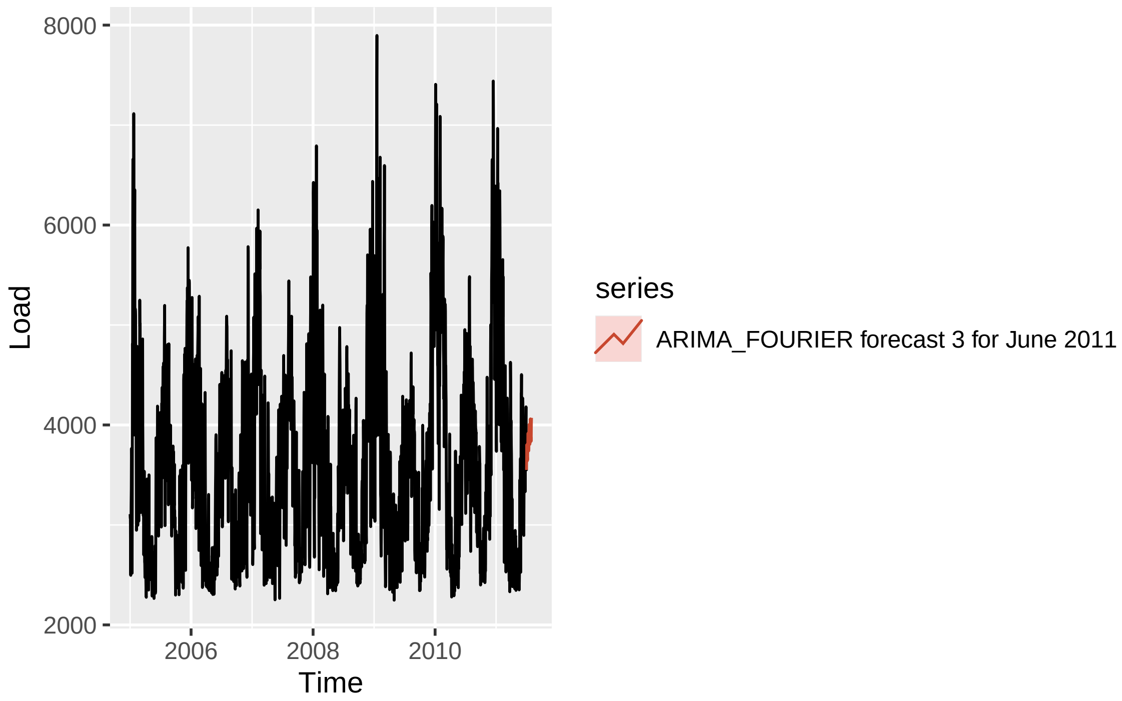
```r
# creating df and submission
date <- seq(as.Date("2011-07-01"), as.Date("2011-07-31"), by = 1)
ARIMA1.df <- data.frame(date, ARIMA_Four_for_all$mean)
write.csv(ARIMA1.df, file = "./submission04.csv")
```

```r
#Fit arima model with fourier terms as exogenous regressors. version 2
ARIMA_Four_fit2_all <- auto.arima(ts_load_daily,
                             seasonal=FALSE,
                             lambda=0,
                             xreg=fourier(ts_load_daily,
                                          K=c(2,2))
                             )
ARIMA_Four_for2_all <- forecast(ARIMA_Four_fit2_all,
                           xreg=fourier(ts_load_daily,
                                        K=c(2,2),
                                        h=31),
                           h=31
                           )

#Plot model + observed data
autoplot(ts_load_daily) +
  autolayer(ARIMA_Four_for2_all,
          series="ARIMA_FOURIER forecast 2 for June 2011",PI=FALSE) +
  ylab("Load")
```



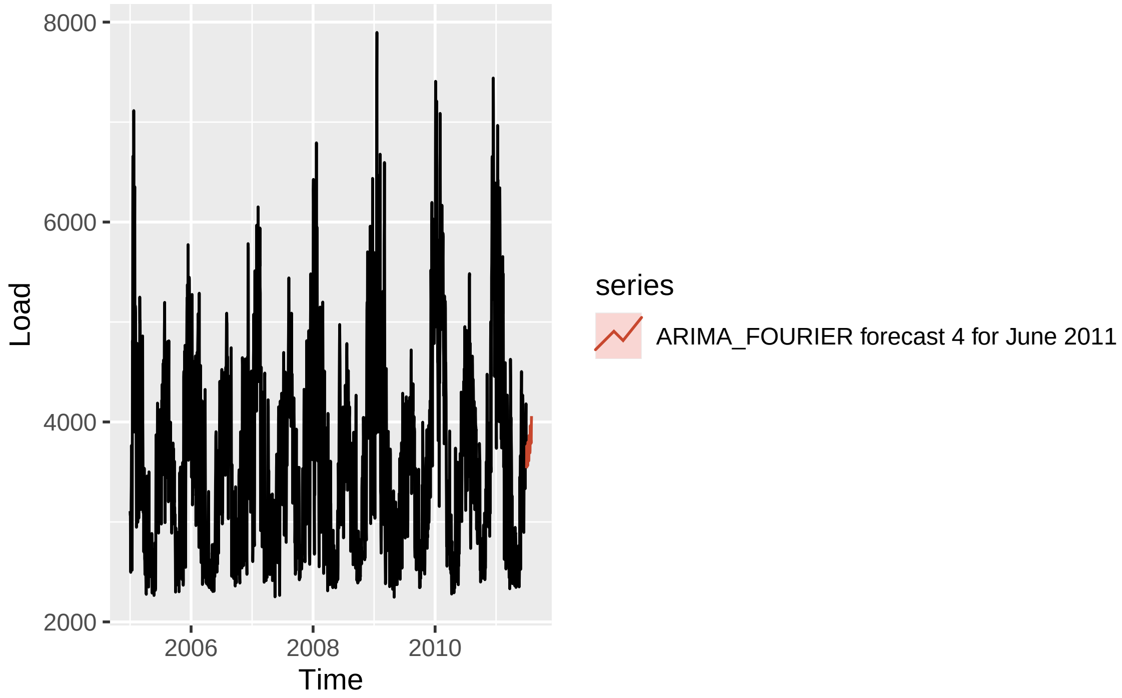
```
# creating df and submission
ARIMA2.df <- data.frame(date, ARIMA_Four_for2_all$mean)
write.csv(ARIMA2.df, file = "./LinPace_05.csv")
```

```
#Fit arima model with fourier terms as exogenous regressors. version 3
ARIMA_Four_fit3_all <- auto.arima(ts_load_daily,
                             seasonal=FALSE,
                             lambda=0,
                             xreg=fourier(ts_load_daily,
                                          K=c(2,4))
                             )

ARIMA_Four_for3_all <- forecast(ARIMA_Four_fit3_all,
                           xreg=fourier(ts_load_daily,
                                        K=c(2,4),
                                        h=31),
                           h=31
                           )
#Plot model + observed data
autoplot(ts_load_daily) +
  autolayer(ARIMA_Four_for3_all,
            series="ARIMA_FOURIER forecast 3 for June 2011",PI=FALSE) +
  ylab("Load")
```



```r
# creating df and submission
ARIMA3.df <- data.frame(date, ARIMA_Four_for3_all$mean)
write.csv(ARIMA3.df, file = "./LinPace_06.csv")
```

```r
#Fit arima model with fourier terms as exogenous regressors. version 4
ARIMA_Four_fit4_all <- auto.arima(ts_load_daily,
                             seasonal=FALSE,
                             lambda=0,
                             xreg=fourier(ts_load_daily,
                                          K=c(2,6))
                             )

ARIMA_Four_for4_all <- forecast(ARIMA_Four_fit4_all,
                           xreg=fourier(ts_load_daily,
                                        K=c(2,6),
                                        h=31),
                           h=31
                           )

#Plot model + observed data
autoplot(ts_load_daily) +
  autolayer(ARIMA_Four_for4_all,
          series="ARIMA_FOURIER forecast 4 for June 2011",PI=FALSE) +
  ylab("Load")
```



```r
# creating df and submission
ARIMA4.df <- data.frame(date, ARIMA_Four_for4_all$mean)
write.csv(ARIMA4.df, file = "./LinPace_07.csv")
```

Exogenous

```r
#Model 2: ARIMA + FOURIER terms with temp
#Write a function for the forecast
ARIMA <- function(x, y, train_data, temp_data, hum_data, test_data) {
  regressors <- as.matrix(data.frame(fourier(train_data, K=c(x, y)),
                                     temp = temp_data,
                                     hum = hum_data))

  temp_forecast <- forecast(temp_data, h=30)
  hum_forecast <- forecast(hum_data, h=30)

  regressors_forecast <- as.matrix(data.frame(fourier(train_data, K=c(x, y),
                                                       h=30),
                                              temp=temp_forecast$mean,
                                              hum=hum_forecast$mean))

  ARIMA_Four_fit <- auto.arima(train_data,
                               seasonal=FALSE,
                               lambda=0,
                               xreg=regressors)
  ARIMA_Four_for <- forecast(ARIMA_Four_fit,
                              xreg=regressors_forecast,
                              h=30)
  scores <- accuracy(ARIMA_Four_for$mean, test_data)
  return(scores[,"MAPE"])
}
```

```
x_value <- 2
y_values <- c(2, 4, 6, 12)
for (y in y_values) {
      mape <- ARIMA(x_value, y, ts_load_daily_train, ts_temp_daily_train,
                    ts_hum_daily_train, ts_load_daily_test)

      cat(sprintf("x: %d, y: %d, MAPE: %.4f\n", x_value, y, mape))
}
```

```
## x: 2, y: 2, MAPE: 10.7632
## x: 2, y: 4, MAPE: 12.9976
## x: 2, y: 6, MAPE: 11.2320
## x: 2, y: 12, MAPE: 10.2572
```

Among all ARIMA forecasts, when x = 2, y = 12, MAPE is the lowest. It’s 10.2572.

```
NN <- function(p, P, x, y, train_data, temp_data, hum_data, test_data) {
  regressors <- as.matrix(data.frame(fourier(train_data, K=c(x, y)),
                                     temp = temp_data,
                                     hum = hum_data))

  temp_forecast <- forecast(temp_data, h=30)
  hum_forecast <- forecast(hum_data, h=30)

  regressors_forecast <- as.matrix(data.frame(fourier(train_data, K=c(x, y),
                                                      h=30),
                                              temp=temp_forecast$mean,
                                              hum=hum_forecast$mean))
  NN_fit <- nnetar(train_data,
                 p=p,
                 P=P,
                 xreg=regressors)

  NN_for <- forecast(NN_fit, h=30, xreg=regressors_forecast)

  scores <- accuracy(NN_for$mean, test_data)
  return(scores[,"MAPE"])
}

best_score <- Inf
best_params <- NULL

p_values <- 1:2
P_values <- 1:2
x_value <- 2
y_values <- c(2, 4, 6, 12)
```

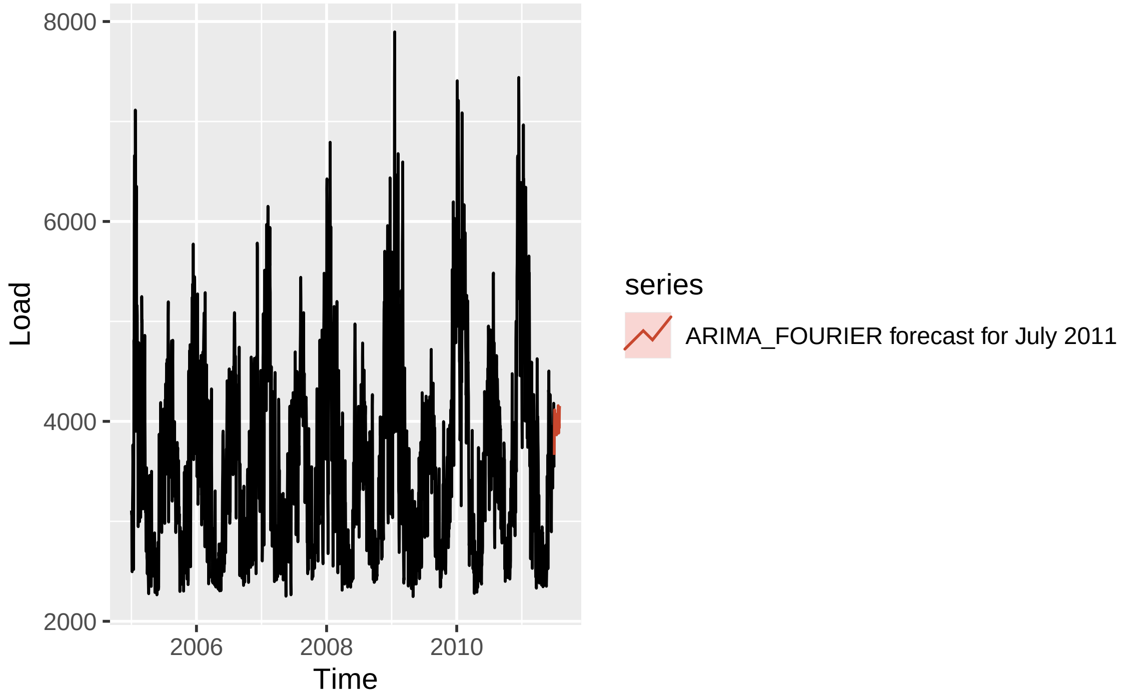
```r
for (p in p_values) {
  for (P in P_values) {
    for (y in y_values) {
      # Calculate MAPE for the current configuration
      mape <- NN(p, P, x_value, y, ts_load_daily_train, ts_temp_daily_train,
                 ts_hum_daily_train, ts_load_daily_test)

      cat(sprintf("p: %d, P: %d, x: %d, y: %d, MAPE: %.4f\n", p, P, x_value, y,
                  mape))

    }
  }
}
```

```
## p: 1, P: 1, x: 2, y: 2, MAPE: 25.6438
## p: 1, P: 1, x: 2, y: 4, MAPE: 23.9728
## p: 1, P: 1, x: 2, y: 6, MAPE: 20.9066
## p: 1, P: 1, x: 2, y: 12, MAPE: 19.4879
## p: 1, P: 2, x: 2, y: 2, MAPE: 25.2760
## p: 1, P: 2, x: 2, y: 4, MAPE: 21.6041
## p: 1, P: 2, x: 2, y: 6, MAPE: 22.2119
## p: 1, P: 2, x: 2, y: 12, MAPE: 20.2192
## p: 2, P: 1, x: 2, y: 2, MAPE: 24.9950
## p: 2, P: 1, x: 2, y: 4, MAPE: 21.7954
## p: 2, P: 1, x: 2, y: 6, MAPE: 20.5538
## p: 2, P: 1, x: 2, y: 12, MAPE: 19.2856
## p: 2, P: 2, x: 2, y: 2, MAPE: 25.1643
## p: 2, P: 2, x: 2, y: 4, MAPE: 20.8346
## p: 2, P: 2, x: 2, y: 6, MAPE: 21.4397
## p: 2, P: 2, x: 2, y: 12, MAPE: 19.9568
```

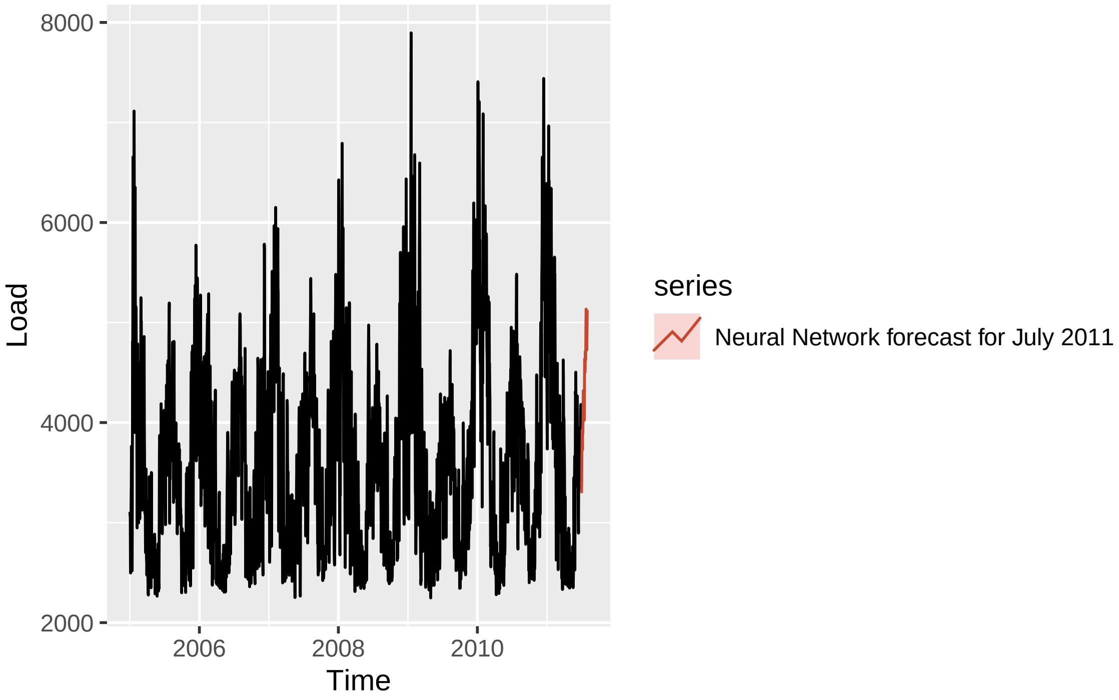
When p = 2, P = 1, x = 2, y = 12, the MAPE of NN forecast is the lowest. It's 19.9353.

When including both exogenous regressors, temperature and relative humidity, ARIMA with x = 2 and y = 12 has the lowest MAPE, if not considering STL+ETS and TBATS.

```r
## Forecast of the best ARIMA model
ARIMA_All_Forecast<- function(x, y, train_data, temp_data, hum_data) {
  regressors <- as.matrix(data.frame(fourier(train_data, K=c(x, y)),
                                     temp = temp_data,
                                     hum = hum_data))

  temp_forecast <- forecast(temp_data, h=31)
  hum_forecast <- forecast(hum_data, h=31)

  regressors_forecast <- as.matrix(data.frame(fourier(train_data, K=c(x, y),
                                                      h=31),
                                              temp=temp_forecast$mean,
                                              hum=hum_forecast$mean))

  ARIMA_Four_fit <- auto.arima(train_data,
                               seasonal=FALSE,
                               lambda=0,
                               xreg=regressors)

  ARIMA_Four_for <- forecast(ARIMA_Four_fit,
                              xreg=regressors_forecast,
                              h=31)

  ARIMA_all_df <- data.frame(ARIMA_Four_for$mean)

  write.csv(ARIMA_all_df, file = "./Data/Submission3", row.names=F)

  autoplot(train_data) +
  autolayer(ARIMA_Four_for, series="ARIMA_FOURIER forecast for July 2011",
            PI=FALSE) +
  ylab("Load")


}

ARIMA_All_Forecast(2, 12, ts_load_daily, ts_temp_daily, ts_hum_daily)
```



```
#Forecast of best NN model
NN_All_Forecast <- function(p, P, x, y, train_data, temp_data, hum_data) {
  regressors <- as.matrix(data.frame(fourier(train_data, K=c(x, y)),
                                     temp = temp_data,
                                     hum = hum_data))

  temp_forecast <- forecast(temp_data, h=31)
  hum_forecast <- forecast(hum_data, h=31)
  regressors_forecast <- as.matrix(data.frame(fourier(train_data, K=c(x, y),
                                                      h=31),
                                              temp=temp_forecast$mean,
                                              hum=hum_forecast$mean))

  NN_fit <- nnetar(train_data,
                 p=p,
                 P=P,
                 xreg=regressors)

  NN_for <- forecast(NN_fit, h=31, xreg=regressors_forecast)

  NN_all_df <- data.frame(NN_for$mean)

  write.csv(NN_all_df, file = "./Data/Submission4", row.names=F)

  autoplot(train_data) +
  autolayer(NN_for, series="Neural Network forecast for July 2011",PI=FALSE) +
  ylab("Load")
}

NN_All_Forecast(2, 1, 2, 12, ts_load_daily, ts_temp_daily, ts_hum_daily)
```


Load
Time
series
Neural Network forecast for July 2011