TSA Competition

https://github.com/vivianzzzzz/ZhangLiuGupta_ENV797_TSA_Competition_S2024.git

Chenjia Liu, Xiyue Zhang, Shubhangi Gupta

2024-04-22

```r
library(readxl)
```

```
## Warning: package 'readxl' was built under R version 4.3.2
```

```r
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 4.3.2

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

```
library(smooth)
```

```
## Warning: package 'smooth' was built under R version 4.3.3

## Loading required package: greybox

## Warning: package 'greybox' was built under R version 4.3.3

## Package "greybox", v2.0.0 loaded.

##
## Attaching package: 'greybox'

## The following object is masked from 'package:lubridate':
##
##     hm

## This is package "smooth", v4.0.0
```

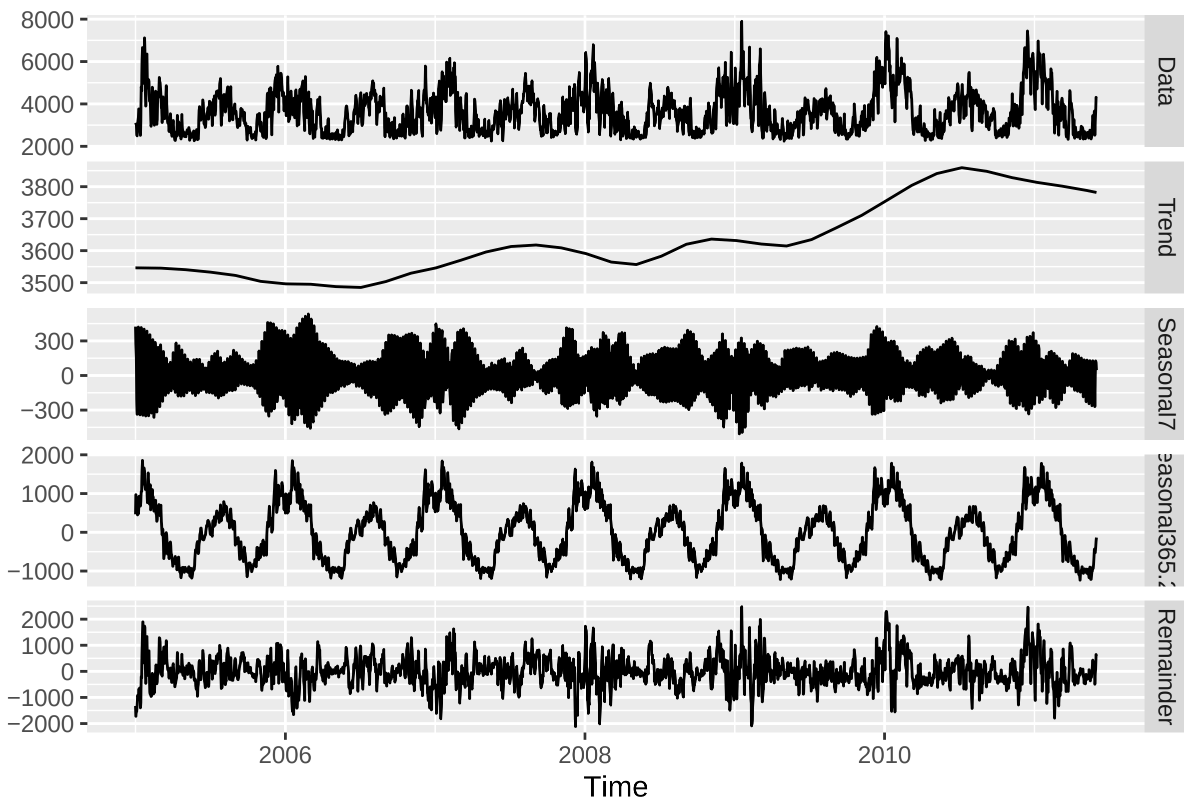
#Data Wrangling We wrangle the load, temperature and humidity data by (1) importing it, (2) converting the date column into a date object using lubridate, (3) calculating the mean load for each day using rowMeans(). For temperature and humidity, we first calculate the hourly values as the average of the value across all workstations, and then use rowmeans() to average the hourly values to daily values. Thus, temperature and humidity daily values are averages of the hourly values across all workstations. (4) checking for NAs, and (5) subsetting the data to only have the columns on the meter_id, date and daily mean. The data for all three variables extends from 1st January 2005 to 30th June 2011.

```
#Load Data
load_raw <- read_excel("./data/load.xlsx")
load <- load_raw %>%
  mutate(date = ymd(date)) %>%
  mutate(daily_mean = rowMeans(select(., 3:26), na.rm = TRUE)) %>%
  filter(!is.na(daily_mean)) %>%
  select(meter_id,date,daily_mean)

#Humidity Data
humidity_raw <- read_excel("./data/relative_humidity.xlsx")

humidity <- humidity_raw %>%
  group_by(date) %>%
  summarise(across(starts_with('rh_ws'), mean))%>%
  mutate(daily_mean = rowMeans(select(., 2:29), na.rm = TRUE)) %>%
  filter(!is.na(daily_mean)) %>%
  select(date,daily_mean)

#Temperature Data
temperature_raw <- read_excel("./data/temperature.xlsx")

temperature <- temperature_raw %>%
  group_by(date) %>%
  summarise(across(starts_with('t_ws'), mean))%>%
```

```
  mutate(daily_mean = rowMeans(select(., 2:29), na.rm = TRUE)) %>%
  filter(!is.na(daily_mean)) %>%
  select(date,daily_mean)

temperature <- temperature[-2373,]
```

#Creating a time series, training and testing data We convert the load, temperature and humidity daily mean datasets into time series objects using hte msts() function. Seasonal periods are taken to be 365.25. The training data extends from 1st January 2005 to 31st May 2011 and the testing data extends from 1st-30th June 2011.

```
#Load Data
ts_load <- msts(load$daily_mean,
                           seasonal.periods =c(7,365.25),
                           start=c(2005,01,01))

ts_load_training <- subset(ts_load,end = length(ts_load)-30)
ts_load_testing <- subset(ts_load,start = length(ts_load)-30)

ts_load_training %>% mstl() %>%
  autoplot()
```



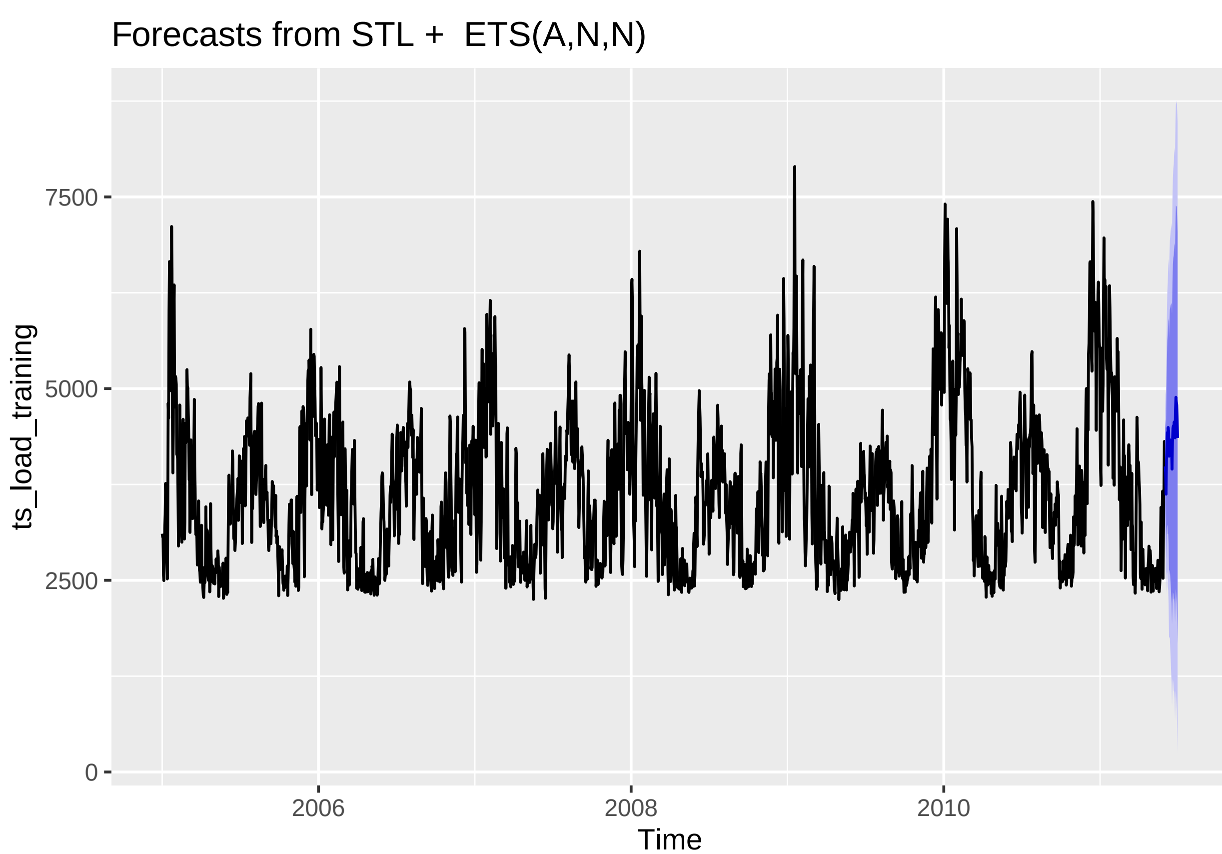

```
#Temperature Data
ts_temperature <- msts(temperature$daily_mean,
                           seasonal.periods =c(7,365.25),
```

```
                             start=c(2005,01,01))

ts_temperature_training <- subset(ts_load,end = length(ts_load)-30)
ts_temperature_testing <- subset(ts_load,start = length(ts_load)-30)

#Humidity Data
ts_humidity <- msts(humidity$daily_mean,
                             seasonal.periods =c(7,365.25),
                             start=c(2005,01,01))

ts_humidity_training <- subset(ts_load,end = length(ts_load)-30)
ts_humidity_testing <- subset(ts_load,start = length(ts_load)-30)
```

#Fitting models to the training data and forecasting on testing data In this section, we fit the following models to the load training data and forecast them for the next 30 days that make up the testing data: (1) STL + ETS (2) TBATS (3) 7 versions of Neural Networks - 4 using different combinations of p and P, 1 with temperature, 1 with humidity, and 1 with temperature+humidity (4) 2 versions of ARIMA - one with temperature, and one with humidity, and (5) SS Exponential smoothing The results of this data have been printed in a table at the end of this section.

##Fitting models only on load data

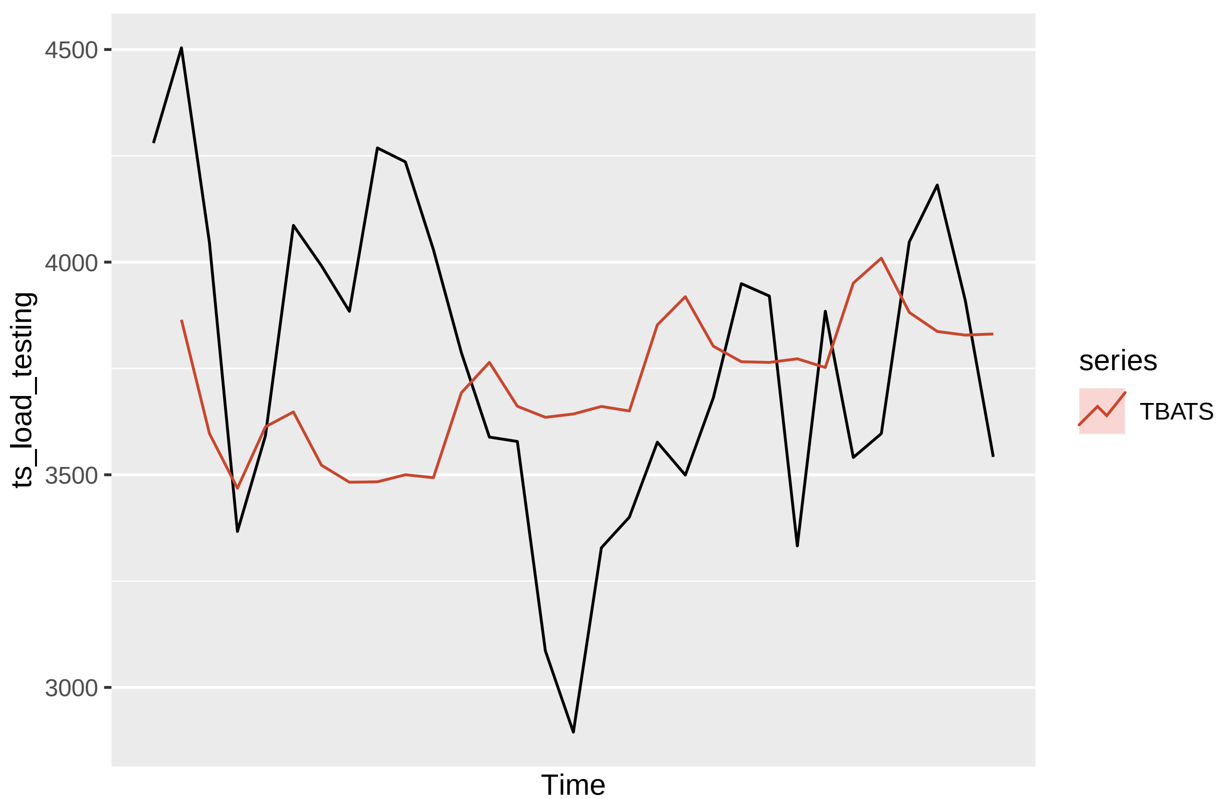
```
#LOAD ONLY
#(1) STL + ETS model
ETS_fit <-  stlf(ts_load_training,h=30)
autoplot(ETS_fit)
```



```r
ETS_scores <- accuracy(ETS_fit$mean,ts_load_testing)
print(ETS_scores)
```

```
##                   ME     RMSE      MAE      MPE     MAPE     ACF1 Theil's U
## Test set -609.6557 736.2245 653.4374 -17.2509 18.24626 0.555879   2.35369
```

```r
# (2) TBATS model
TBATS_fit <- tbats(ts_load_training)

TBATS_forcast <- forecast(TBATS_fit, h=30)

autoplot(ts_load_testing) +
  autolayer(TBATS_forcast, series="TBATS",PI=FALSE)
```

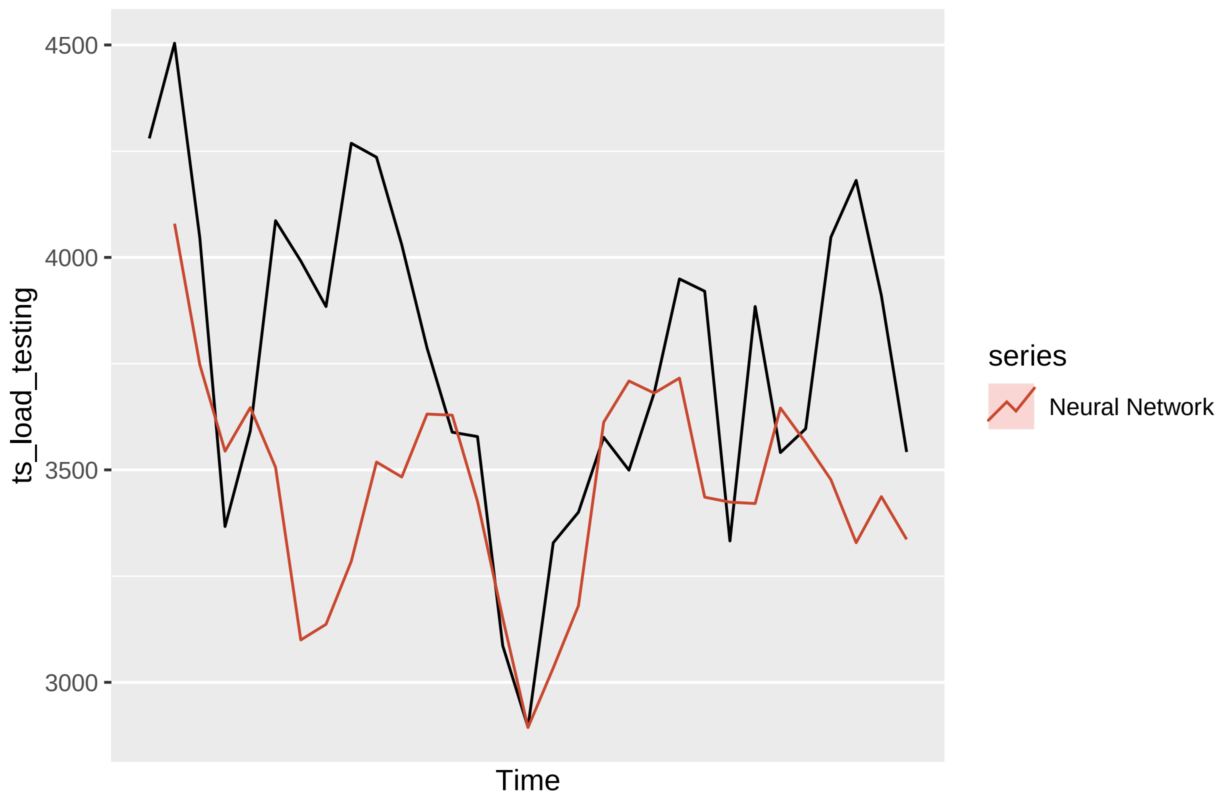



```r
TBATS_scores <- accuracy(TBATS_forcast$mean,ts_load_testing)

print(TBATS_scores)
```

```
##                  ME     RMSE      MAE      MPE     MAPE      ACF1 Theil's U
## Test set 32.82197 401.3228 341.3879 -0.10145 9.177415 0.6154517  1.187443
```

```r
# (3.1) Neural Network (p=1, P=1)
NN_fit1 <- nnetar(ts_load_training,p=1,P=1,xreg=fourier(ts_load_training, K=c(2,12)))
```

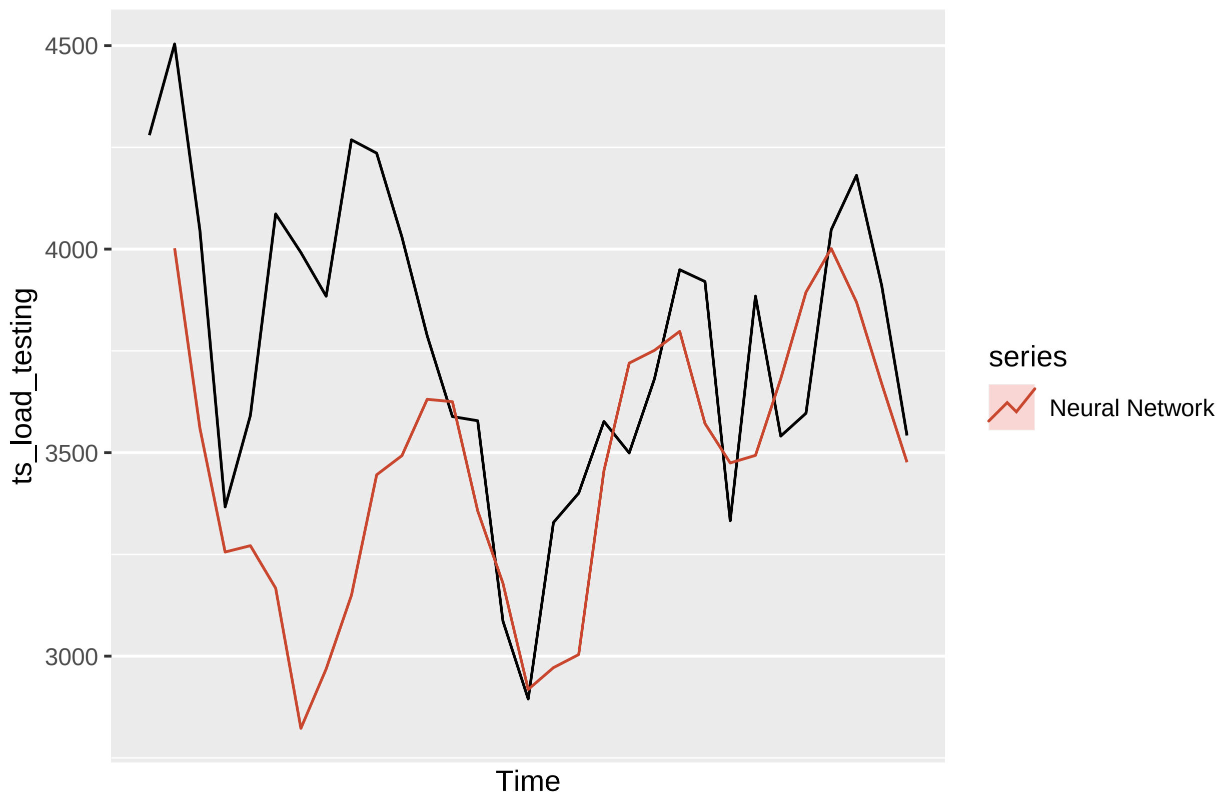
```r
NN_for1 <- forecast(NN_fit1,h=30, xreg=fourier(ts_load_training, K=c(2,12),h=30))

autoplot(ts_load_testing) +
  autolayer(NN_for1, series="Neural Network",PI=FALSE)
```




```r
NN_scores1 <- accuracy(NN_for1$mean,ts_load_testing)

print(NN_scores1)
```

```
##                 ME     RMSE     MAE      MPE    MAPE      ACF1 Theil's U
## Test set 285.0872 441.7436 337.018 7.023042 8.53726 0.5379985  1.280341
```

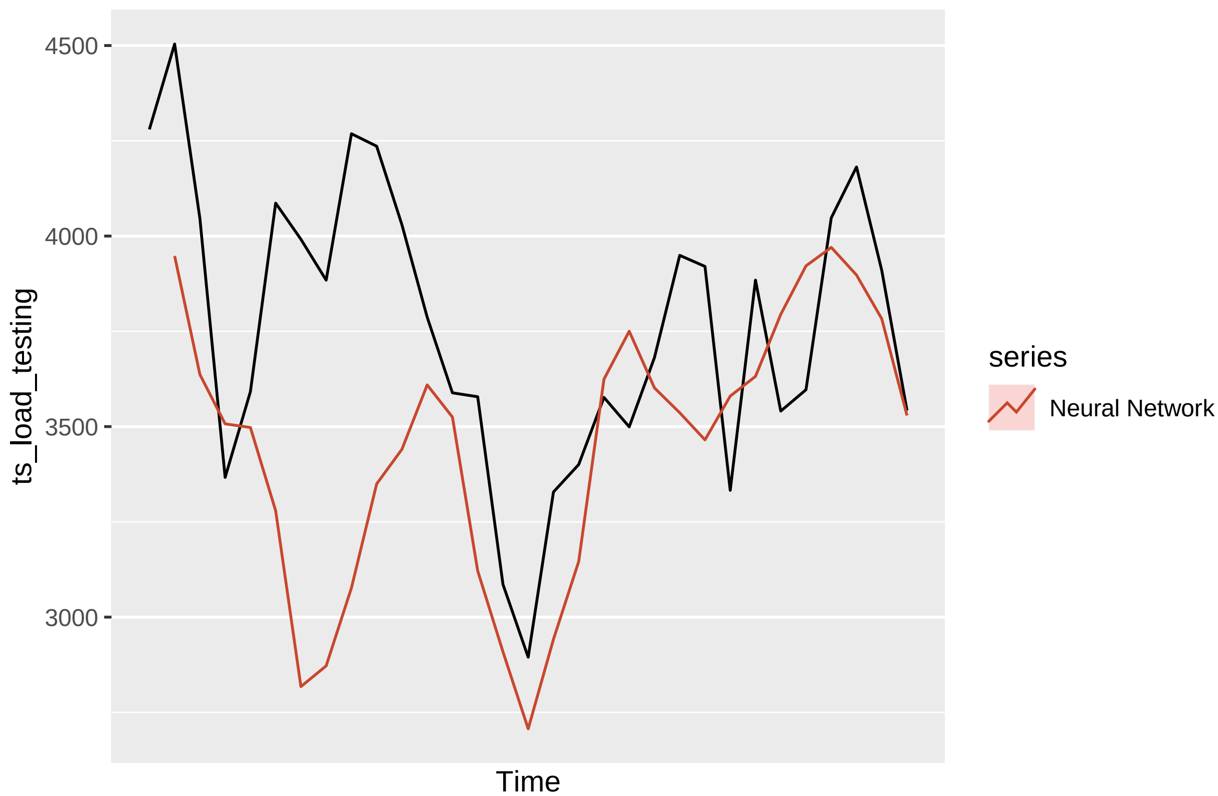
```r
#(3.2) Neural Network (p=2, P=2)
NN_fit2 <- nnetar(ts_load_training,p=2,P=1,xreg=fourier(ts_load_training, K=c(2,12)))
NN_for2 <- forecast(NN_fit2,h=30, xreg=fourier(ts_load_training, K=c(2,12),h=30))

autoplot(ts_load_testing) +
  autolayer(NN_for2, series="Neural Network",PI=FALSE)
```




```
NN_scores2 <- accuracy(NN_for2$mean,ts_load_testing)

print(NN_scores2)
```

```
##                 ME     RMSE     MAE      MPE     MAPE      ACF1 Theil's U
## Test set 288.4378 477.592 356.676 7.161719 9.131435 0.6735996  1.391056
```

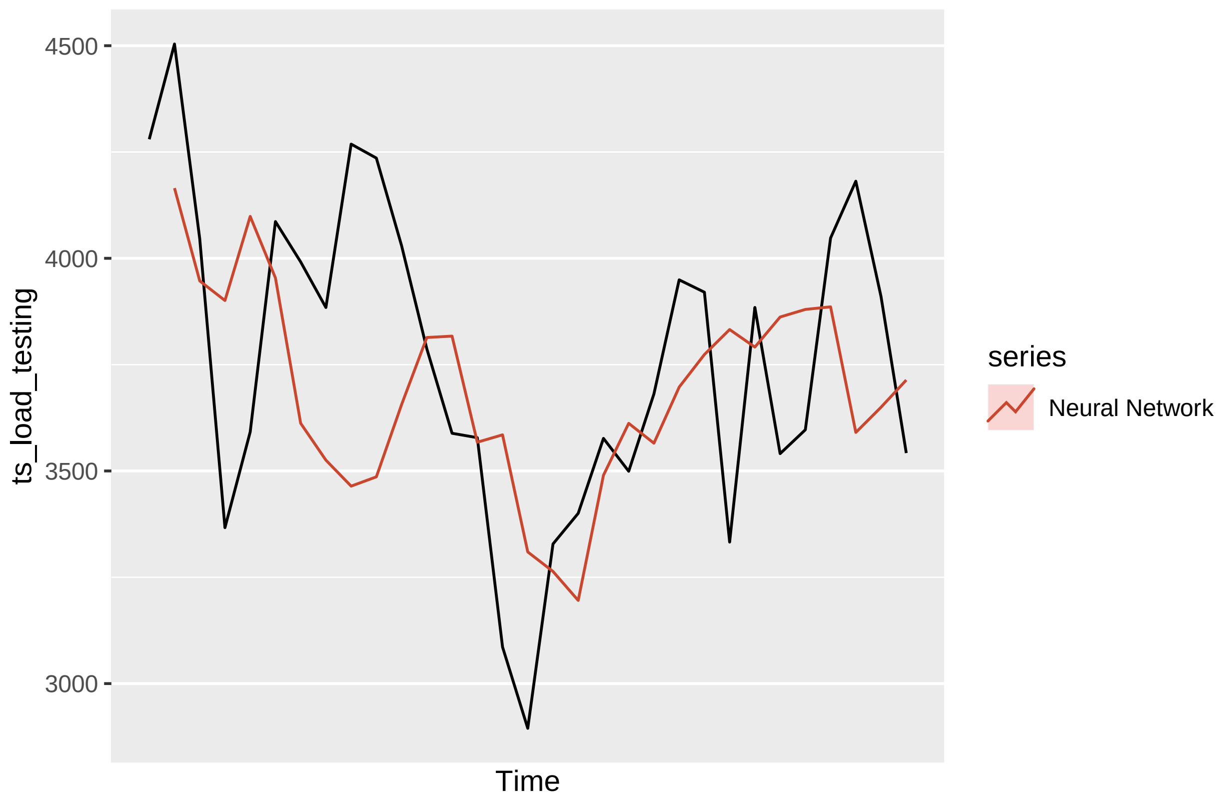
```
#(3.3) Neural Network (p=3, P=1)
NN_fit3 <- nnetar(ts_load_training,p=3,P=1,xreg=fourier(ts_load_training, K=c(2,12)))
NN_for3 <- forecast(NN_fit3,h=30, xreg=fourier(ts_load_training, K=c(2,12),h=30))

autoplot(ts_load_testing) +
  autolayer(NN_for3, series="Neural Network",PI=FALSE)
```




```r
NN_scores3 <- accuracy(NN_for3$mean,ts_load_testing)

print(NN_scores3)
```

```
##                 ME     RMSE      MAE      MPE     MAPE      ACF1 Theil's U
## Test set 295.361 499.5234 379.7224 7.369221 9.789742 0.6421115  1.444447
```

```r
#(3.4) Neural Network (p=1, P=0)
NN_fit4 <- nnetar(ts_load_training,p=1,P=0,xreg=fourier(ts_load_training, K=c(2,12)))
NN_for4 <- forecast(NN_fit4,h=30, xreg=fourier(ts_load_training, K=c(2,12),h=30))

autoplot(ts_load_testing) +
  autolayer(NN_for4, series="Neural Network",PI=FALSE)
```




```
NN_scores4 <- accuracy(NN_for4$mean,ts_load_testing)

print(NN_scores4)
```

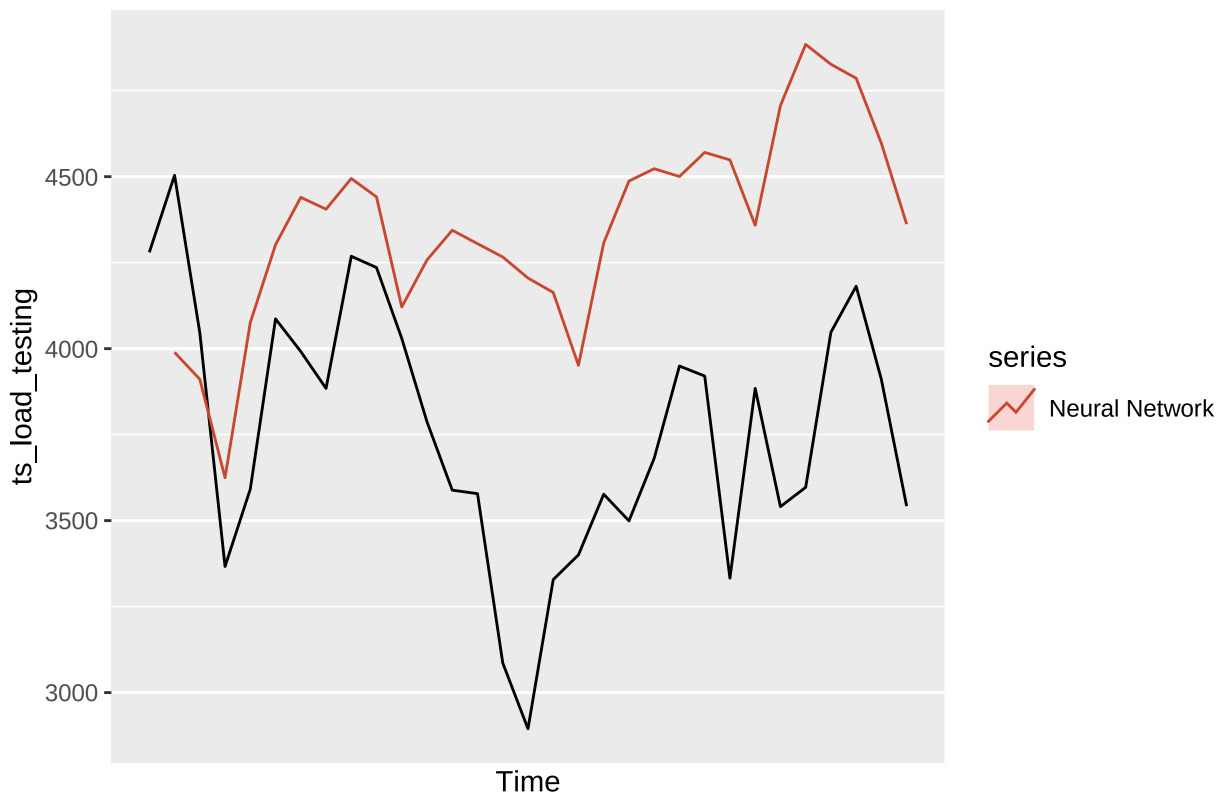
```
##                 ME     RMSE      MAE      MPE     MAPE      ACF1 Theil's U
## Test set 54.16279 358.6393 294.0842 0.711521 7.882517 0.4781256  1.058454
```

##Forecasting with temperature and humidity

```
#With temperature data
temp_regressors<- as.matrix(data.frame(fourier(ts_load_training, K=c(2,12)), "temp"= ts_temperature_tra

temp_for<-forecast(ts_temperature_training,h=30)

temp_regressors_for<-as.matrix(data.frame(fourier(ts_load_training, K=c(2,12),h=30), "temp"= temp_for$me

#with humidity data
hum_regressors<- as.matrix(data.frame(fourier(ts_load_training, K=c(2,12)), "hum"= ts_humidity_training

hum_for<-forecast(ts_humidity_training,h=30)

hum_regressors_for<-as.matrix(data.frame(fourier(ts_load_training, K=c(2,12),h=30), "hum"= hum_for$mean

#With both temperature and humidity
temp_hum_regressors<- as.matrix(data.frame(fourier(ts_load_training, K=c(2,12)), "temp"= ts_temperature
```

```
temp_hum_regressors_for<-as.matrix(data.frame(fourier(ts_load_training, K=c(2,12),h=30), "temp"= temp_fo
```

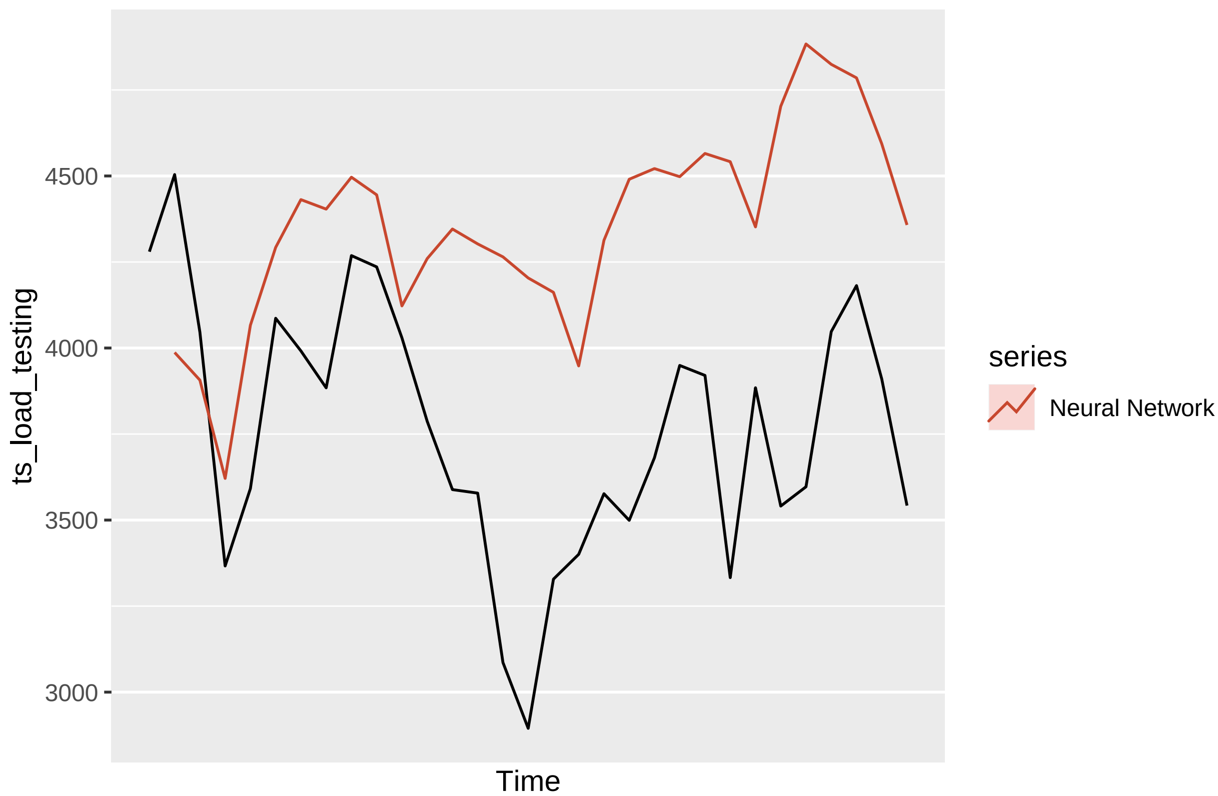
```
# (3.5) Neural Network + Temperature
NN_fit5 <- nnetar(ts_load_training,p=1,P=0,xreg=temp_regressors)
NN_for5 <- forecast(NN_fit5,h=30, xreg=temp_regressors_for)

autoplot(ts_load_testing) +
  autolayer(NN_for5, series="Neural Network",PI=FALSE)
```




```
NN_scores5 <- accuracy(NN_for5$mean,ts_load_testing)

print(NN_scores5)
```

```
##                   ME     RMSE      MAE      MPE     MAPE      ACF1 Theil's U
## Test set -614.2002 739.8185 657.5193 -17.3744 18.35878 0.5545055  2.365307
```

```
# (3.6) Neural Network + Humidity
NN_fit6 <- nnetar(ts_load_training,p=1,P=0,xreg=hum_regressors)
NN_for6 <- forecast(NN_fit6,h=30, xreg=hum_regressors_for)

autoplot(ts_load_testing) +
  autolayer(NN_for6, series="Neural Network",PI=FALSE)
```




```
NN_scores6 <- accuracy(NN_for6$mean,ts_load_testing)

print(NN_scores6)
```

```
##                  ME     RMSE      MAE       MPE     MAPE      ACF1 Theil's U
## Test set -611.8558 737.9602 655.6123 -17.31099 18.30581 0.5545949  2.359444
```

```
# (3.7) Neural Network + Temperature + Humidity
NN_fit7 <- nnetar(ts_load_training,p=1,P=0,xreg=temp_hum_regressors)
NN_for7 <- forecast(NN_fit7,h=30, xreg=temp_hum_regressors_for)

autoplot(ts_load_testing) +
  autolayer(NN_for7, series="Neural Network",PI=FALSE)
```


```
NN_scores7 <- accuracy(NN_for7$mean,ts_load_testing)

print(NN_scores7)
```

```
##                    ME     RMSE      MAE       MPE     MAPE      ACF1 Theil's U
## Test set -609.3594 736.1251 653.8423 -17.24395 18.25565 0.5571227  2.353904
```

```
# (4.1) Arima + Temperature
ARIMA_fit1<-auto.arima(ts_load_training,seasonal= FALSE, lambda=0,xreg=temp_regressors)
ARIMA_for1<-forecast(ARIMA_fit1,xreg=temp_regressors_for,h=30)

autoplot(ts_load_testing) +
  autolayer(ARIMA_for1, series="ARIMA",PI=FALSE)
```


```
ARIMA_scores1 <- accuracy(ARIMA_for1$mean,ts_load_testing)

print(ARIMA_scores1)
```

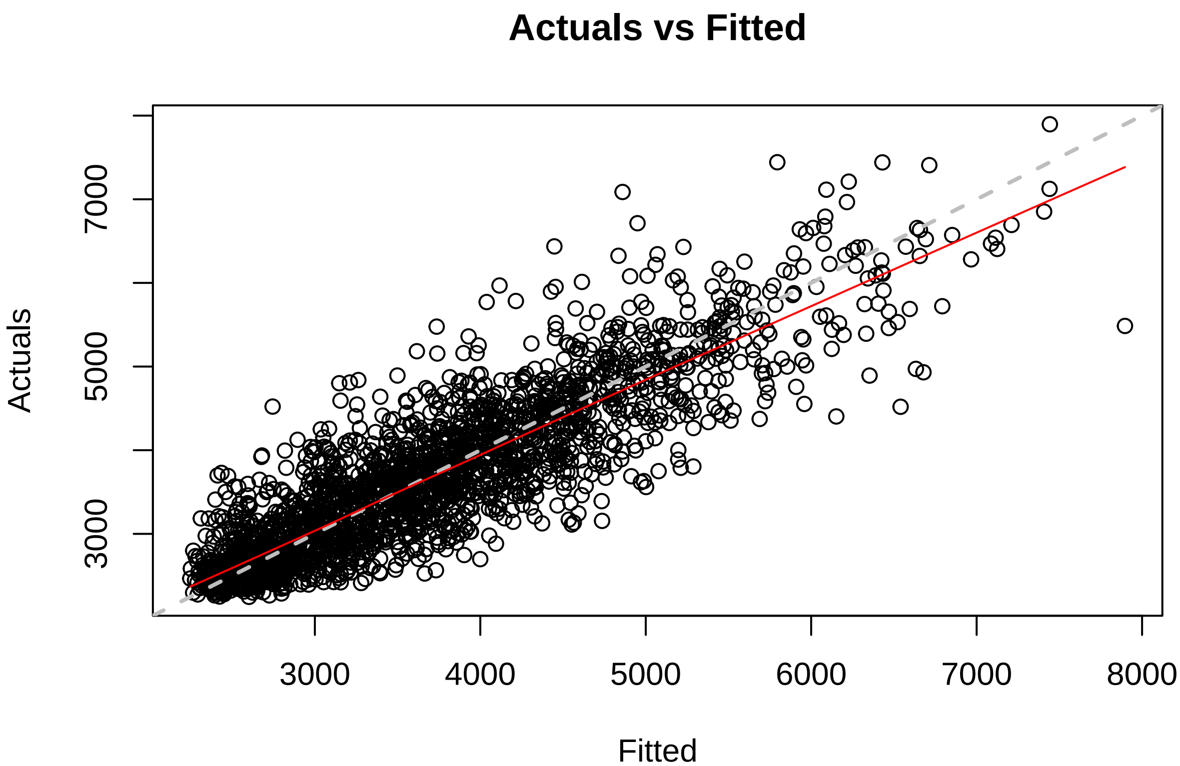
```
##                 ME     RMSE      MAE       MPE     MAPE      ACF1 Theil's U
## Test set -553.068 710.4597 616.3782 -15.72947 17.18329 0.5768567  2.249589
```

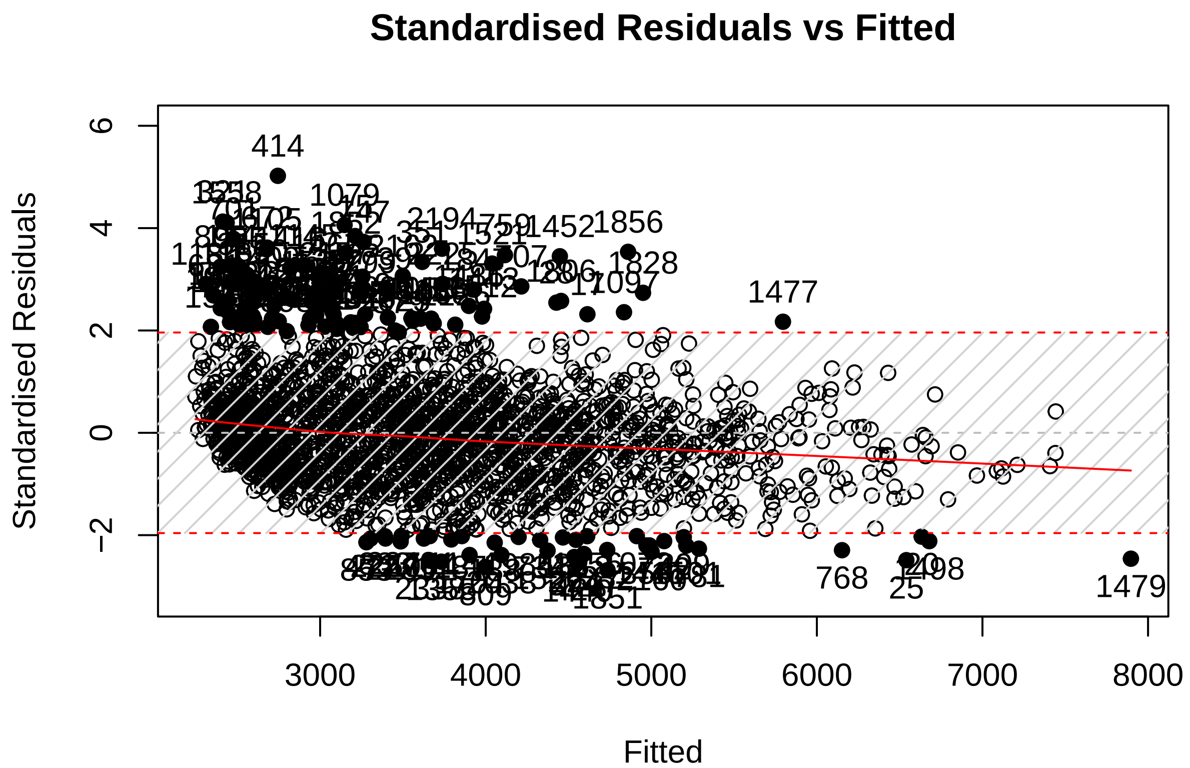
```
# (4.2) Arima + humidity
ARIMA_fit2<-auto.arima(ts_load_training,seasonal= FALSE, lambda=0,xreg=hum_regressors)
ARIMA_for2<-forecast(ARIMA_fit2,xreg=hum_regressors_for,h=30)

autoplot(ts_load_testing) +
  autolayer(ARIMA_for2, series="ARIMA",PI=FALSE)
```

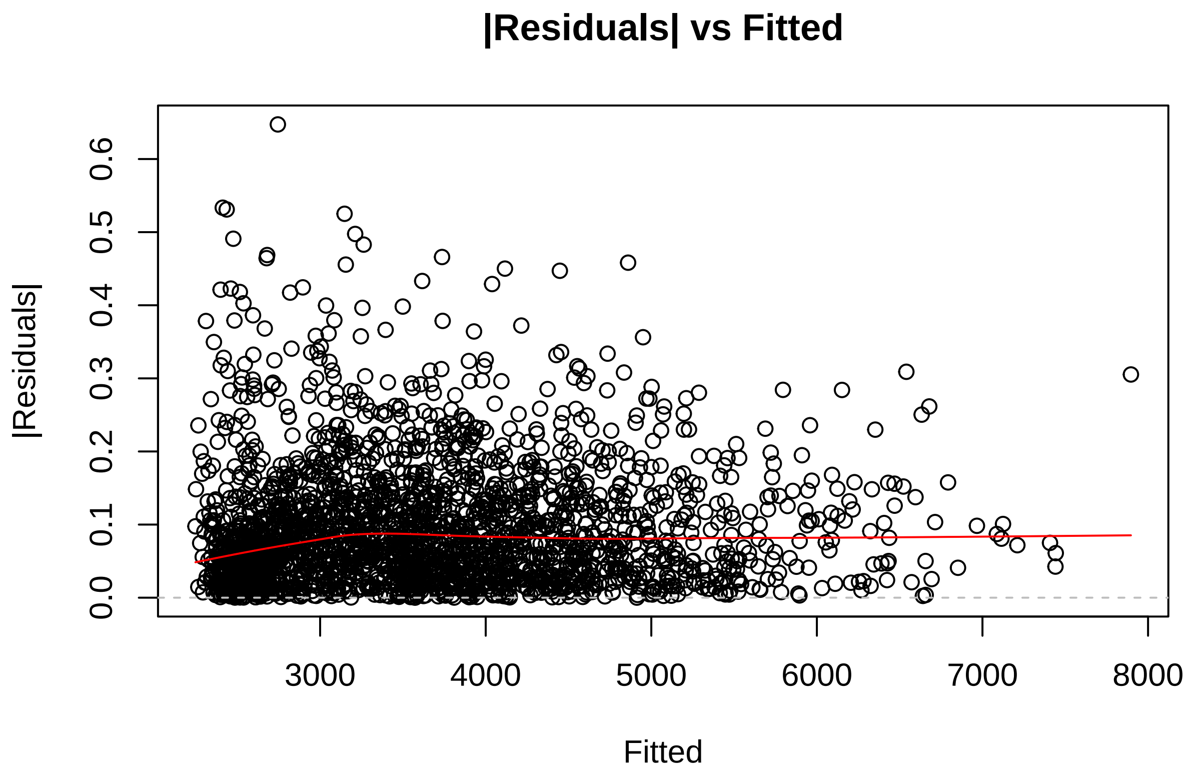

```r
ARIMA_scores2 <- accuracy(ARIMA_for2$mean,ts_load_testing)

print(ARIMA_scores2)
```

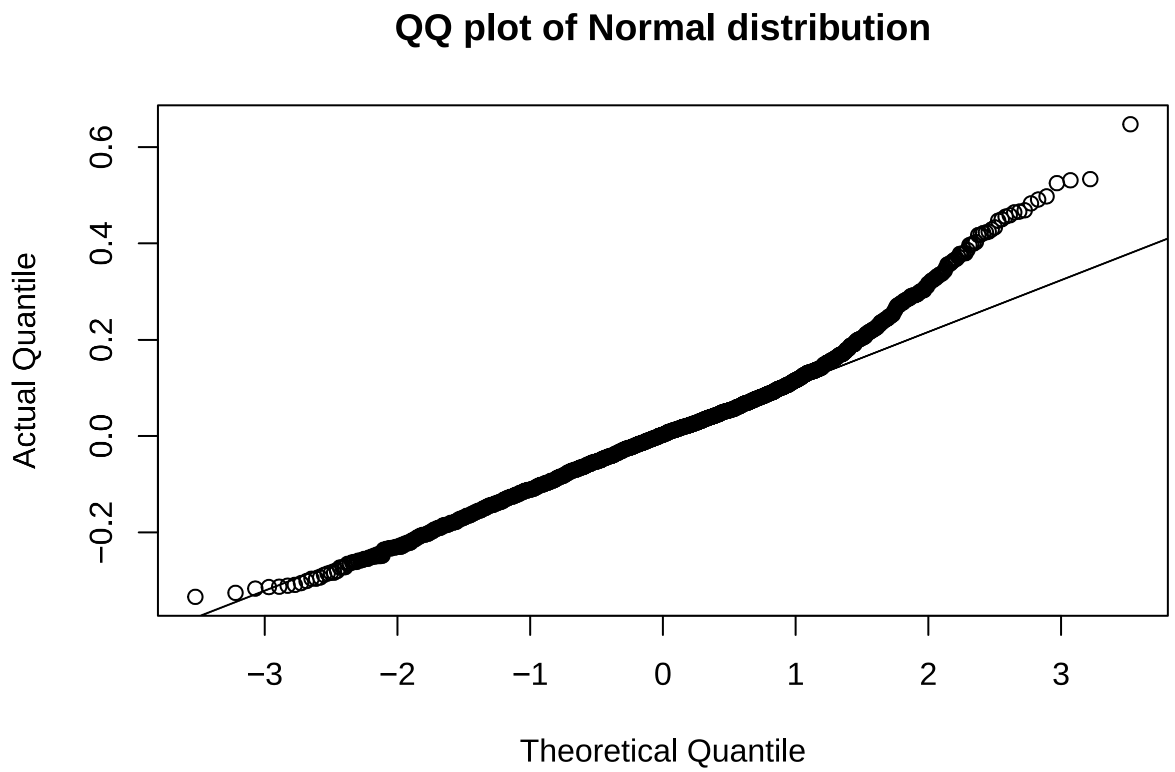
```
##                ME     RMSE      MAE       MPE     MAPE      ACF1 Theil's U
## Test set -553.068 710.4597 616.3782 -15.72947 17.18329 0.5768567  2.249589
```

```r
#(5) SS Exponential smoothing

SSES_fit1 <- es(ts_load_training,model="ZZZ",h=30,holdout=FALSE)
plot(SSES_fit1)
```


Actuals vs Fitted
Actuals
3000
5000
7000
3000
4000
5000
6000
7000
8000
Fitted


Standardised Residuals vs Fitted
Standardised Residuals
Fitted
6
4
2
0
−2
3000
4000
5000
6000
7000
8000
414
1477
768
1479


|Residuals| vs Fitted
|Residuals|
0.0
0.1
0.2
0.3
0.4
0.5
0.6
3000
4000
5000
6000
7000
8000
Fitted




```
SSES_scores1 <- accuracy(SSES_fit1$forecast,ts_load_testing)
print(SSES_scores1)
```

```
##                  ME     RMSE      MAE       MPE     MAPE      ACF1 Theil's U
## Test set -535.6292 645.6398 550.5681 -15.41089 15.74257 0.5240066  2.126168
```

##Printing the scores of all the above fitted models in one table

```
scores <- as.data.frame(
  rbind(TBATS_scores, NN_scores1,NN_scores2, NN_scores3,NN_scores4, NN_scores5, NN_scores6, ARIMA_score
  )
row.names(scores) <- c("TBATS_scores", "NN_scores1","NN_scores2","NN_scores3","NN_scores4","NN_scores5"
scores
```

```
##                        ME     RMSE      MAE        MPE      MAPE      ACF1
## TBATS_scores     32.82197 401.3228 341.3879  -0.101450  9.177415 0.6154517
## NN_scores1      285.08722 441.7436 337.0180   7.023042  8.537260 0.5379985
## NN_scores2      288.43776 477.5920 356.6760   7.161719  9.131435 0.6735996
## NN_scores3      295.36104 499.5234 379.7224   7.369221  9.789742 0.6421115
## NN_scores4       54.16279 358.6393 294.0842   0.711521  7.882517 0.4781256
## NN_scores5     -614.20016 739.8185 657.5193 -17.374395 18.358781 0.5545055
## NN_scores6     -611.85579 737.9602 655.6123 -17.310986 18.305815 0.5545949
## ARIMA_scores1 -553.06799 710.4597 616.3782 -15.729466 17.183291 0.5768567
##               Theil's U
## TBATS_scores   1.187443
```

```
## NN_scores1     1.280341
## NN_scores2     1.391056
## NN_scores3     1.444447
## NN_scores4     1.058454
## NN_scores5     2.365307
## NN_scores6     2.359444
## ARIMA_scores1  2.249589
```

#Forecasting daily demand for July 2011 In this section, we use the fitted models to forecast daily demand for July 2011. These results have been uploaded on Kaggle. We forecast the following models: (1) SS Exponential Smoothing (2) TBATS (3) 5 Neural Network Models - 2 with different combinations of p and P, one with temperature, one with humidity, and one with temperature and humidity (4) two ARIMA models - one with temperature, and one with humidity

##Forecasting only load data

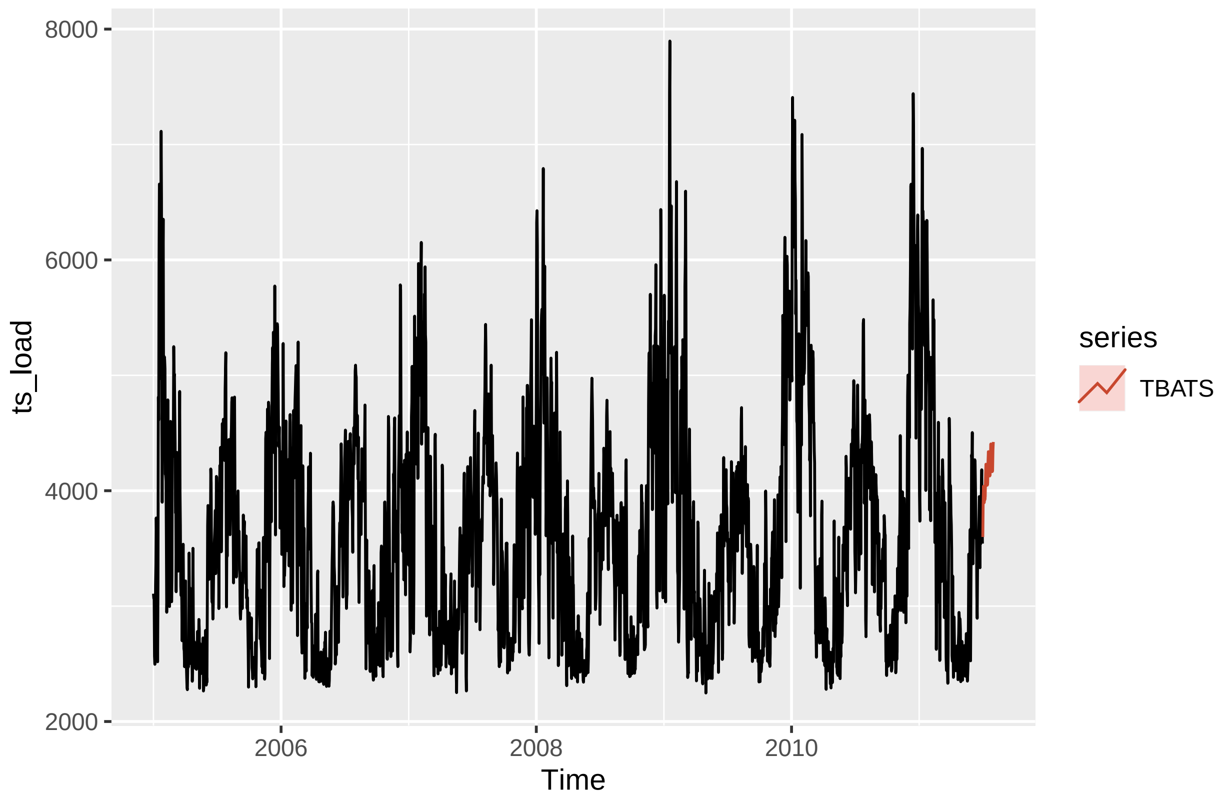
```r
# (1) SS Exponential smoothing
SSES_fit_load <- es(ts_load,model="ZZZ",h=31,holdout=FALSE)

#Exporting into a CSV
date <- seq(ymd("2011-07-01"), ymd("2011-07-31"), by = "days")
load<-SSES_fit_load$forecast
July_SSES <-data.frame(date=date, load=load)
July_SSES
```

```
##          date     load
## 1  2011-07-01 3542.083
## 2  2011-07-02 3542.083
## 3  2011-07-03 3542.083
## 4  2011-07-04 3542.083
## 5  2011-07-05 3542.083
## 6  2011-07-06 3542.083
## 7  2011-07-07 3542.083
## 8  2011-07-08 3542.083
## 9  2011-07-09 3542.083
## 10 2011-07-10 3542.083
## 11 2011-07-11 3542.083
## 12 2011-07-12 3542.083
## 13 2011-07-13 3542.083
## 14 2011-07-14 3542.083
## 15 2011-07-15 3542.083
## 16 2011-07-16 3542.083
## 17 2011-07-17 3542.083
## 18 2011-07-18 3542.083
## 19 2011-07-19 3542.083
## 20 2011-07-20 3542.083
## 21 2011-07-21 3542.083
## 22 2011-07-22 3542.083
## 23 2011-07-23 3542.083
## 24 2011-07-24 3542.083
## 25 2011-07-25 3542.083
## 26 2011-07-26 3542.083
## 27 2011-07-27 3542.083
## 28 2011-07-28 3542.083
```

```
## 29 2011-07-29 3542.083
## 30 2011-07-30 3542.083
## 31 2011-07-31 3542.083
```

```
write.csv(July_SSES, file = "July_SSES.csv", row.names = FALSE)
```

```
# (2) TBATS
TBATS_fit_load <- tbats(ts_load)
TBATS_forcast_load <- forecast(TBATS_fit_load, h=31)

autoplot(ts_load) +
  autolayer(TBATS_forcast_load, series="TBATS",PI=FALSE)
```



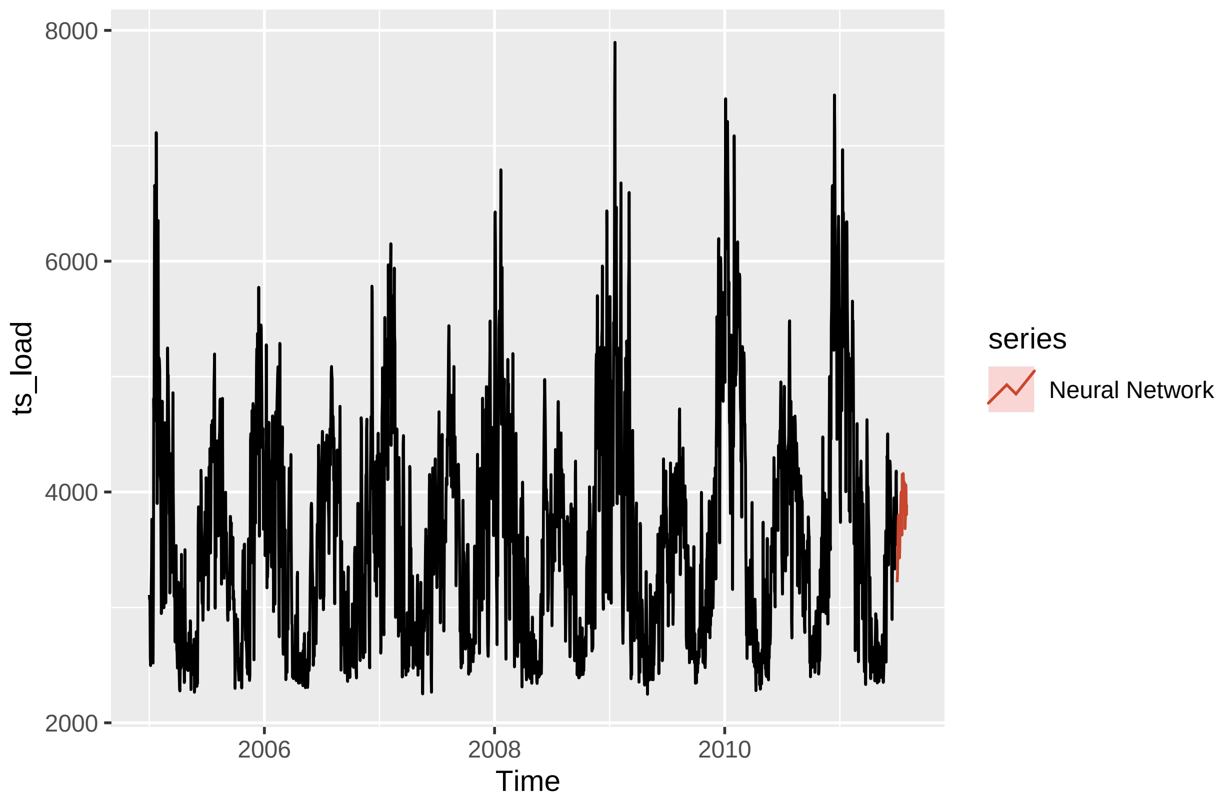

```
#Exporting into a CSV
date <- seq(ymd("2011-07-01"), ymd("2011-07-31"), by = "days")
load<-TBATS_forcast_load$mean
July_TBATS<-data.frame(date=date, load=load)
July_TBATS
```

```
##         date     load
## 1  2011-07-01 3598.771
## 2  2011-07-02 3886.334
## 3  2011-07-03 4038.799
## 4  2011-07-04 3944.307
```

```
## 5  2011-07-05 3896.595
## 6  2011-07-06 3943.398
## 7  2011-07-07 3922.083
## 8  2011-07-08 3951.288
## 9  2011-07-09 4141.061
## 10 2011-07-10 4229.657
## 11 2011-07-11 4106.773
## 12 2011-07-12 4045.933
## 13 2011-07-13 4082.449
## 14 2011-07-14 4048.286
## 15 2011-07-15 4068.249
## 16 2011-07-16 4255.097
## 17 2011-07-17 4338.693
## 18 2011-07-18 4206.031
## 19 2011-07-19 4137.537
## 20 2011-07-20 4168.842
## 21 2011-07-21 4128.072
## 22 2011-07-22 4142.534
## 23 2011-07-23 4326.580
## 24 2011-07-24 4405.148
## 25 2011-07-25 4264.095
## 26 2011-07-26 4188.257
## 27 2011-07-27 4213.343
## 28 2011-07-28 4165.435
## 29 2011-07-29 4173.131
## 30 2011-07-30 4351.156
## 31 2011-07-31 4422.473
```

```r
write.csv(July_TBATS, file = "July_TBATS.csv", row.names = FALSE)
```

```r
# (3.1) Neural Network (p=1, P=1)
NN_fit1_load <- nnetar(ts_load,p=1,P=1,xreg=fourier(ts_load, K=c(2,12)))
NN_for1_load <- forecast(NN_fit1_load,h=31, xreg=fourier(ts_load, K=c(2,12),h=31))

autoplot(ts_load) +
  autolayer(NN_for1_load, series="Neural Network",PI=FALSE)
```



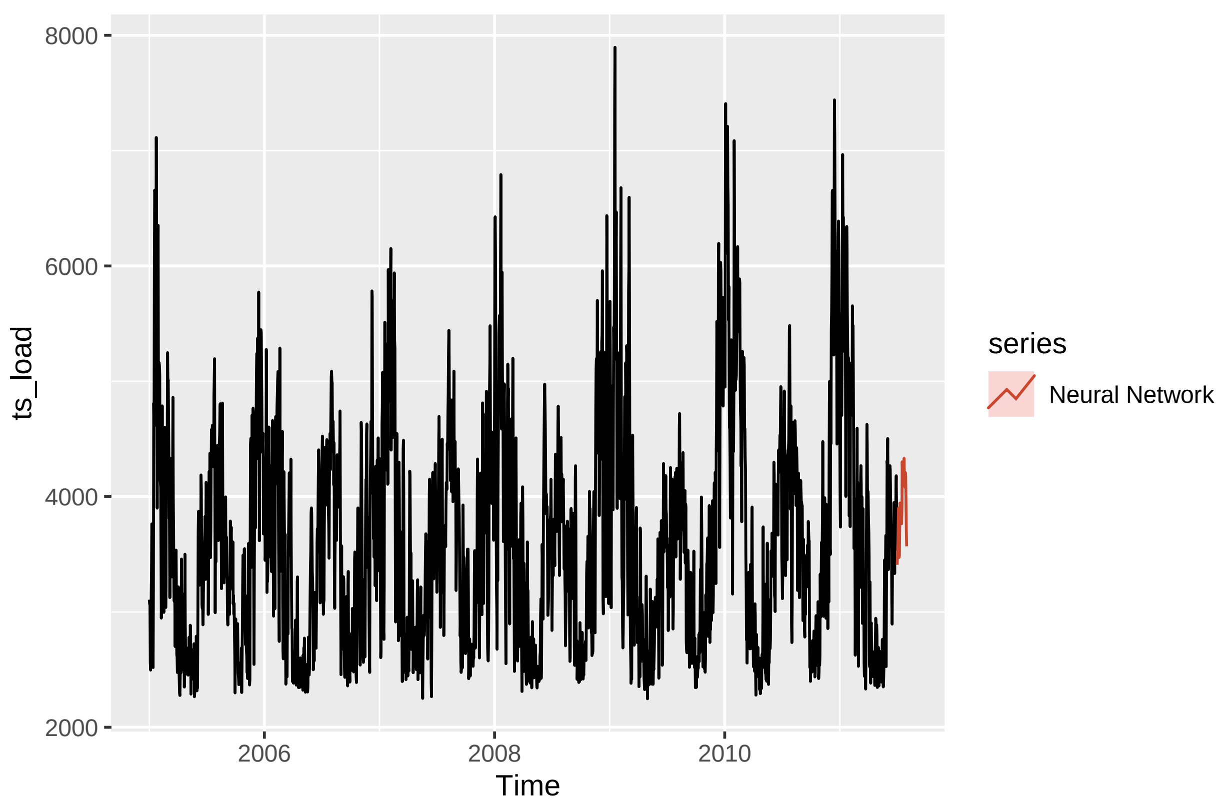

```
#Exporting into a CSV
date <- seq(ymd("2011-07-01"), ymd("2011-07-31"), by = "days")
load<-NN_for1_load$mean
July_NN1<-data.frame(date=date, load=load)
July_NN1
```

```
##          date     load
## 1  2011-07-01 3217.808
## 2  2011-07-02 3284.205
## 3  2011-07-03 3477.532
## 4  2011-07-04 3716.971
## 5  2011-07-05 3799.361
## 6  2011-07-06 3757.837
## 7  2011-07-07 3618.873
## 8  2011-07-08 3428.540
## 9  2011-07-09 3474.402
## 10 2011-07-10 3704.394
## 11 2011-07-11 3645.076
## 12 2011-07-12 3866.825
## 13 2011-07-13 3964.888
## 14 2011-07-14 4000.641
## 15 2011-07-15 3689.772
## 16 2011-07-16 3626.660
## 17 2011-07-17 4152.683
## 18 2011-07-18 4134.493
## 19 2011-07-19 4090.823
```

```
## 20 2011-07-20 4159.642
## 21 2011-07-21 4101.557
## 22 2011-07-22 4056.978
## 23 2011-07-23 4088.177
## 24 2011-07-24 4070.809
## 25 2011-07-25 3996.130
## 26 2011-07-26 3682.428
## 27 2011-07-27 3774.370
## 28 2011-07-28 4064.085
## 29 2011-07-29 4009.852
## 30 2011-07-30 3801.524
## 31 2011-07-31 3891.080
```

```
write.csv(July_NN1, file = "July_NN1.csv", row.names = FALSE)
```

```
# (3.2) Neural Network (p=1, P=0)
NN_fit4_load <- nnetar(ts_load,p=1,P=0,xreg=fourier(ts_load, K=c(2,12)))
NN_for4_load <- forecast(NN_fit4_load,h=31, xreg=fourier(ts_load, K=c(2,12),h=31))

autoplot(ts_load) +
  autolayer(NN_for4_load, series="Neural Network",PI=FALSE)
```




```
#Exporting into a CSV
date <- seq(ymd("2011-07-01"), ymd("2011-07-31"), by = "days")
load<-NN_for4_load$mean
```

```r
July_NN4<-data.frame(date=date, load=load)
July_NN4
```

```
##          date     load
## 1  2011-07-01 3408.872
## 2  2011-07-02 3487.946
## 3  2011-07-03 3588.096
## 4  2011-07-04 3715.087
## 5  2011-07-05 3900.320
## 6  2011-07-06 3831.936
## 7  2011-07-07 3472.343
## 8  2011-07-08 3487.517
## 9  2011-07-09 3715.726
## 10 2011-07-10 3945.821
## 11 2011-07-11 3781.993
## 12 2011-07-12 3878.113
## 13 2011-07-13 3923.474
## 14 2011-07-14 3835.773
## 15 2011-07-15 3764.783
## 16 2011-07-16 3893.368
## 17 2011-07-17 4302.281
## 18 2011-07-18 4206.596
## 19 2011-07-19 4207.894
## 20 2011-07-20 4286.607
## 21 2011-07-21 4245.448
## 22 2011-07-22 4291.647
## 23 2011-07-23 4332.310
## 24 2011-07-24 4146.067
## 25 2011-07-25 4194.133
## 26 2011-07-26 4081.493
## 27 2011-07-27 4209.103
## 28 2011-07-28 4188.173
## 29 2011-07-29 3942.556
## 30 2011-07-30 3714.785
## 31 2011-07-31 3569.291
```

```r
write.csv(July_NN4, file = "July_NN4.csv", row.names = FALSE)
```

##Forecasting load data with regressors for temperature and humidity

```r
#Temperature
temp_regressors_load<- as.matrix(data.frame(fourier(ts_load, K=c(2,12)), "temp"= ts_temperature))

temp_for_load<-forecast(ts_temperature,h=31)
temp_regressors_for_load<-as.matrix(data.frame(fourier(ts_load, K=c(2,12),h=31), "temp"= temp_for_load$m

#Humidity

hum_regressors_load<- as.matrix(data.frame(fourier(ts_load, K=c(2,12)), "hum"= ts_humidity))

hum_for_load<-forecast(ts_humidity,h=31)
hum_regressors_for_load<-as.matrix(data.frame(fourier(ts_load, K=c(2,12),h=31), "hum"= hum_for_load$mean
```

```r
#Temperature & Humidity
temp_hum_regressors_load<- as.matrix(data.frame(fourier(ts_load, K=c(2,12)), "temp"= ts_temperature, "hu

temp_hum_regressors_for_load<-as.matrix(data.frame(fourier(ts_load, K=c(2,12),h=31), "temp"= temp_for_lo
```

```r
# (3.3) Neural Network + Temperature
NN_fit5_load <- nnetar(ts_load,p=1,P=0,xreg=temp_regressors_load)
NN_for5_load <- forecast(NN_fit5_load,h=31, xreg=temp_regressors_for_load)

#Exporting into a CSV
date <- seq(ymd("2011-07-01"), ymd("2011-07-31"), by = "days")
load<- NN_for5_load$mean
July_NN5 <-data.frame(date=date, load=load)
write.csv(July_NN5, file = "July_NN5.csv", row.names = FALSE)
```

```r
# (3.4) Neural Network + Humidity
NN_fit6_load <- nnetar(ts_load,p=1,P=0,xreg=hum_regressors_load)
NN_for6_load <- forecast(NN_fit6_load,h=31, xreg=hum_regressors_for_load)

#Exporting into a CSV
date <- seq(ymd("2011-07-01"), ymd("2011-07-31"), by = "days")
load<-NN_for6_load$mean
July_NN6 <-data.frame(date=date, load=load)
write.csv(July_NN6, file = "July_NN6.csv", row.names = FALSE)
```

```r
# (3.5) Neural Network + Temperature + Humidity
NN_fit7_load <- nnetar(ts_load,p=1,P=0,xreg=temp_hum_regressors_load)
NN_for7_load <- forecast(NN_fit7_load,h=31, xreg=temp_hum_regressors_for_load)

#Exporting into a CSV
date <- seq(ymd("2011-07-01"), ymd("2011-07-31"), by = "days")
load<-NN_for7_load$mean
July_NN7 <-data.frame(date=date, load=load)
write.csv(July_NN7, file = "July_NN7.csv", row.names = FALSE)
```

```r
# (4.1) Arima+Temperature
ARIMA_fit1_load<-auto.arima(ts_load,seasonal= FALSE, lambda=0,xreg=temp_regressors_load)
ARIMA_for1_load<-forecast(ARIMA_fit1_load,xreg=temp_regressors_for_load,h=31)

#Exporting into a CSV
date <- seq(ymd("2011-07-01"), ymd("2011-07-31"), by = "days")
load<-ARIMA_for1_load$mean
ARIMA_for1_load <-data.frame(date=date, load=load)
write.csv(ARIMA_for1_load, file = "July_ARIMA1.csv", row.names = FALSE)
```

```r
# (4.2) Arima+humidity
ARIMA_fit2_load<-auto.arima(ts_load,seasonal= FALSE, lambda=0,xreg=hum_regressors_load)
ARIMA_for2_load<-forecast(ARIMA_fit2_load,xreg=hum_regressors_for_load,h=31)

#Exporting into a CSV
date <- seq(ymd("2011-07-01"), ymd("2011-07-31"), by = "days")
```

```r
load<-ARIMA_for2_load$mean
ARIMA_fit2_load <-data.frame(date=date, load=load)
write.csv(ARIMA_fit2_load, file = "July_ARIMA2.csv", row.names = FALSE)
```