R Notebook

https://github.com/hxia5/XiaGupta_ENV797_TSA_Competition_S2024

Haochong Xia, Ayush Gupta

2024-04-26

```r
library(readxl)

# Load the Excel file into a data frame
data <- read_excel("data/load.xlsx")


# Read the Excel file
temperature_data <- read_excel('data/temperature.xlsx')


relative_humidity_data <- read_excel("data/relative_humidity.xlsx")
```

```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(magrittr)
library(lubridate)
```

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

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

```r
suppressPackageStartupMessages(library(lubridate))

load <- data %>%
  mutate(date = ymd(date)) %>% #converts date format
  mutate(d_mean = rowMeans(select(., 3:26), na.rm = TRUE)) %>% #Calculates the daily mean and ignore NA
  select(date,d_mean)
```

```r
#Filled in missing value in temp data with last hour's value
# Loop through each column of the dataframe
```

```r
for (i in 2:ncol(temperature_data)) {
  # Loop through each row of the column
  for (j in 2:nrow(temperature_data)) {
    # If the value is missing, replace it with the value from the row above
    if (is.na(temperature_data[j, i])) {
      temperature_data[j, i] <- temperature_data[j - 1, i]
    }
  }
}
```

```r
temp <- temperature_data %>%
  group_by(date) %>%
  summarise(across(starts_with('t_ws'), mean))%>% #Groups the data by date and calculates the mean
  mutate(d_mean = rowMeans(select(., 2:29), na.rm = TRUE)) %>% #Calculates the daily mean and ignore NA
  select(date,d_mean)
```

```r
hum <- relative_humidity_data %>%
  group_by(date) %>%
  summarise(across(starts_with('rh_ws'), mean))%>% #Groups the data by date and calculates the mean
  mutate(d_mean = rowMeans(select(., 2:29), na.rm = TRUE)) %>% #Calculates the daily mean and ignore NA
  select(date,d_mean)
```

```r
# Basic model for first try
library(forecast)
```

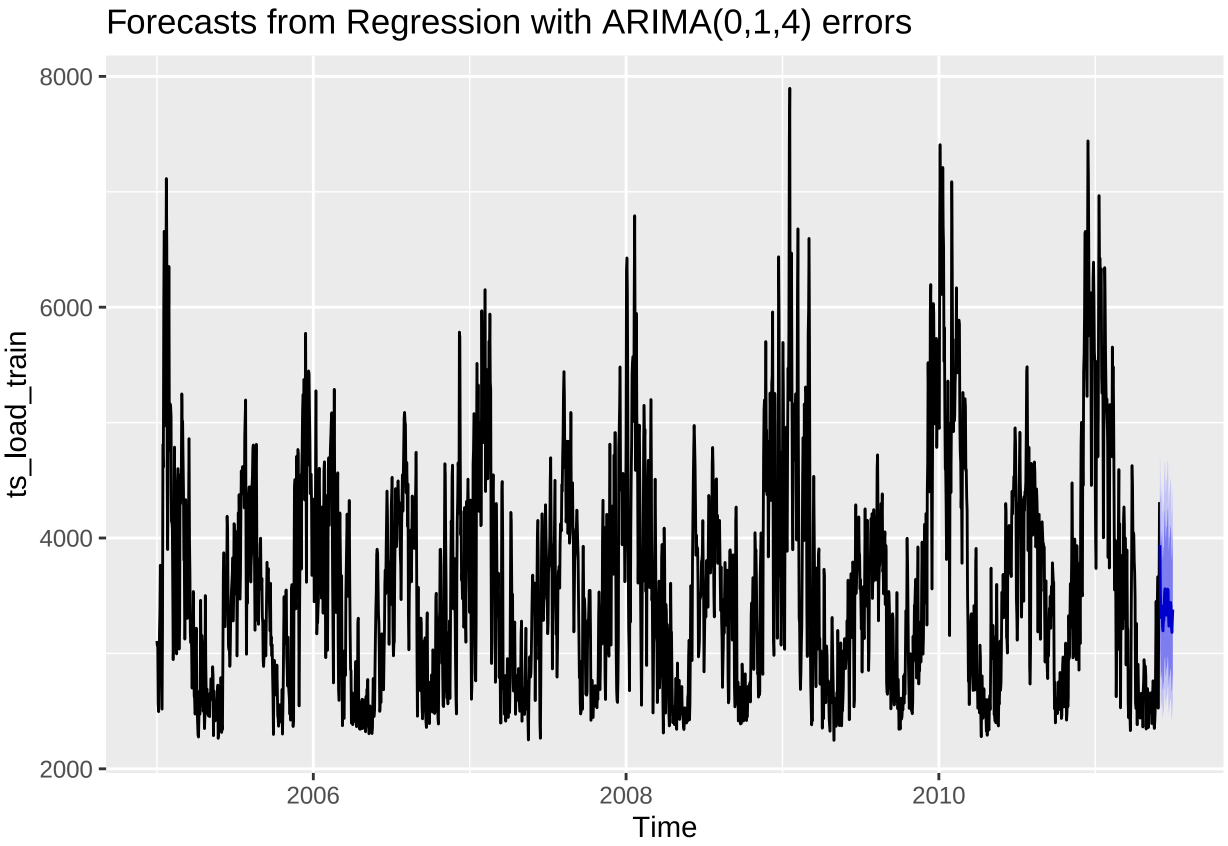
```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```r
suppressPackageStartupMessages(library(quantmod))

# Create a time series object using 'h_combined' column
#ts_data <- ts(load$d_mean, start = min(load$date), end = max(load$date), frequency = 365)

#auto_arima_model <- auto.arima(ts_data)

# Print the summary of the automatically selected ARIMA model
#summary(auto_arima_model)
```

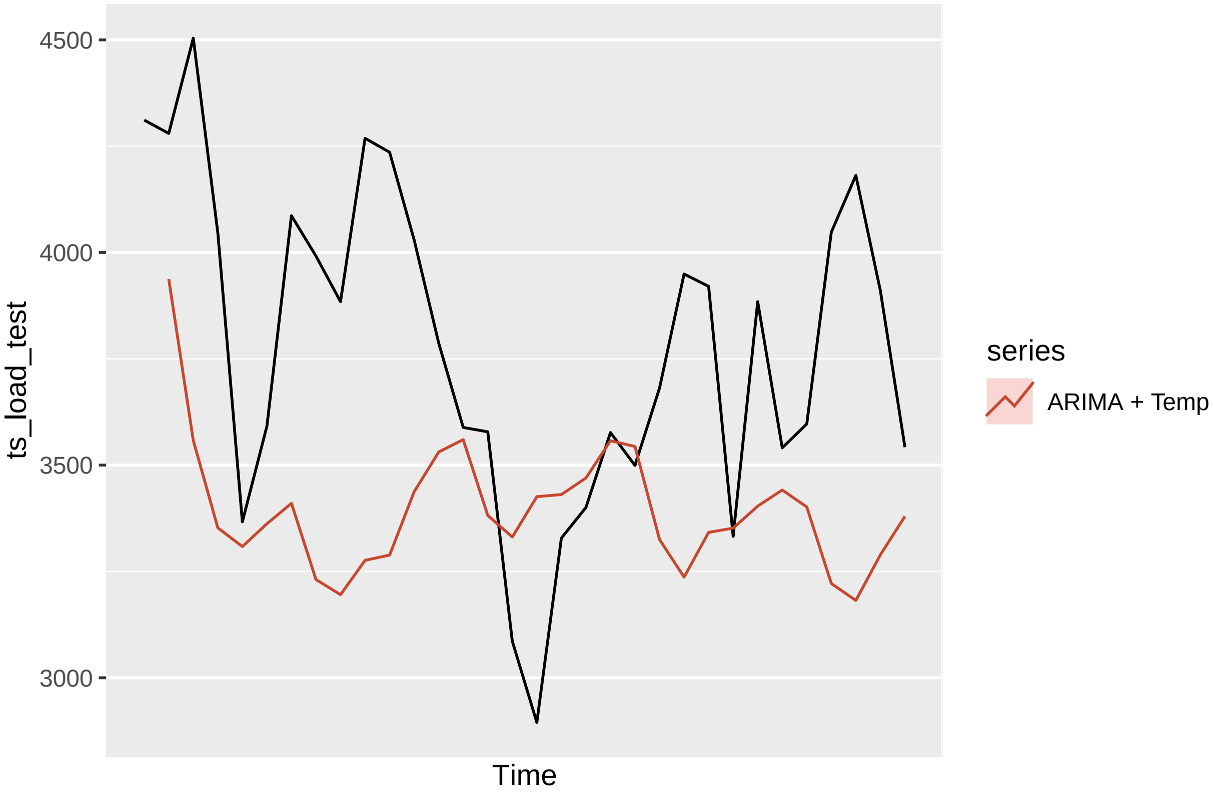
```r
#Creating time series
ts_load <- msts(load$d_mean,seasonal.periods =c(7,365.25), start=c(2005,01,01))
ts_load_train <- subset(ts_load,end =length(ts_load)-31)
ts_load_test <- subset(ts_load,start =length(ts_load)-31)

ts_temp <- msts(temp$d_mean,seasonal.periods=c(7,365.25), start=c(2005,01,01))
ts_temp_train <- subset(ts_temp,end=length(ts_load)-31)
ts_temp_test <- subset(ts_temp,start =length(ts_load)-31)

ts_hum <- msts(hum$d_mean,seasonal.periods=c(7,365.25),start=c(2005,01,01))
ts_hum_train <- subset(ts_hum,end =length(ts_load)-31)
ts_hum_test <- subset(ts_hum,start =length(ts_load)-31)

temp_regressor<- as.matrix(data.frame(fourier(ts_load_train,K=c(2,12)), "temp"= ts_temp_train))
temp_fc<-forecast(ts_temp_train,h=31)
temp_regressor_fc<-as.matrix(data.frame(fourier(ts_load_train,K=c(2,12),h=31),"temp"=temp_fc$mean))
```

```
hum_regressor<- as.matrix(data.frame(fourier(ts_load_train, K=c(2,12)), "hum"=ts_hum_train))
hum_fc<-forecast(ts_hum_train,h=31)
hum_regressor_fc<-as.matrix(data.frame(fourier(ts_load_train,K=c(2,12),h=31),"hum"= hum_fc$mean))


temp_hum_regressors<- as.matrix(data.frame(fourier(ts_load_train, K=c(2,12)), "temp"= ts_temp_train, "hu
temp_hum_regressors_fc<-as.matrix(data.frame(fourier(ts_load_train,K=c(2,12),h=31), "temp"=temp_fc$mean
```

```
#Arima+Temperature
ARIMA_fit_tp<-auto.arima(ts_load_train,seasonal= FALSE, lambda=0,xreg=temp_regressor)
ARIMA_fc_tp<-forecast(ARIMA_fit_tp,xreg=temp_regressor_fc,h=31)

autoplot(ARIMA_fc_tp)
```



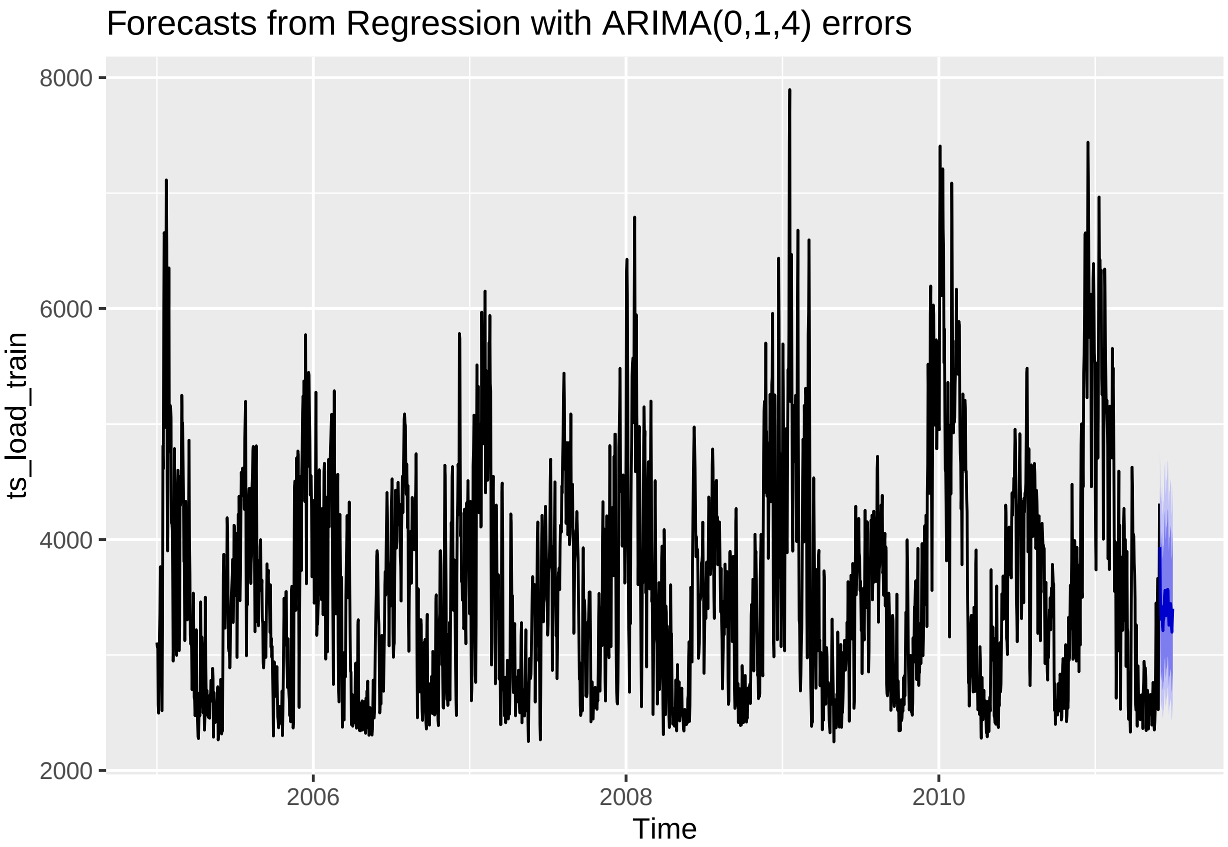
```
autoplot(ts_load_test) +
  autolayer(ARIMA_fc_tp, series="ARIMA + Temp",PI=FALSE)
```



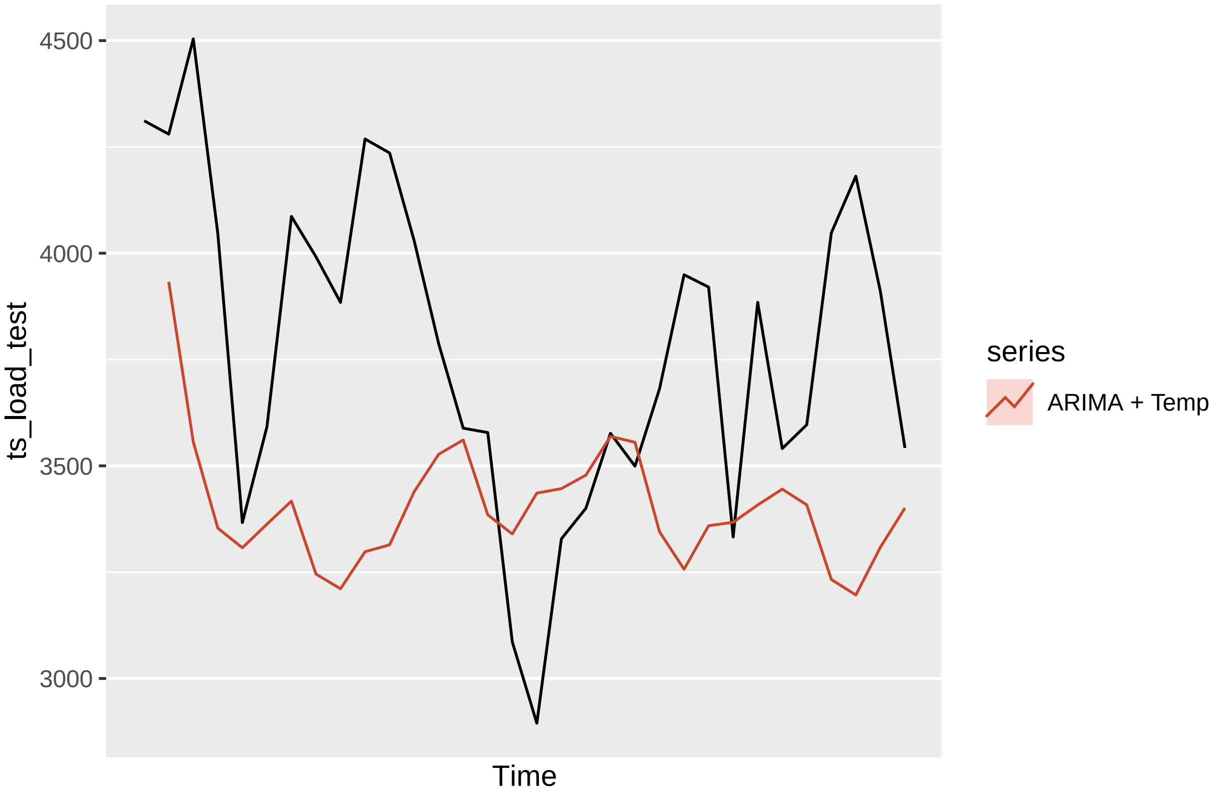
```
ARIMA_scores_tp <- accuracy(ARIMA_fc_tp$mean,ts_load_test)
print(ARIMA_scores_tp)
```

```
##                ME     RMSE     MAE      MPE    MAPE      ACF1 Theil's U
## Test set 369.2936 541.7948 434.547 8.907236 11.0521 0.6046054  1.598575
```

```
#Arima+ temp + hum
ARIMA_fit_tp_hum<-auto.arima(ts_load_train,seasonal= FALSE, lambda=0,xreg=temp_hum_regressors)
ARIMA_fc_tp_hum<-forecast(ARIMA_fit_tp_hum,xreg=temp_hum_regressors_fc,h=31)

autoplot(ARIMA_fc_tp_hum)
```



```
autoplot(ts_load_test) +
  autolayer(ARIMA_fc_tp_hum, series="ARIMA + Temp",PI=FALSE)
```



```
ARIMA_scores_tp_hum <- accuracy(ARIMA_fc_tp_hum$mean,ts_load_test)
print(ARIMA_scores_tp_hum)
```

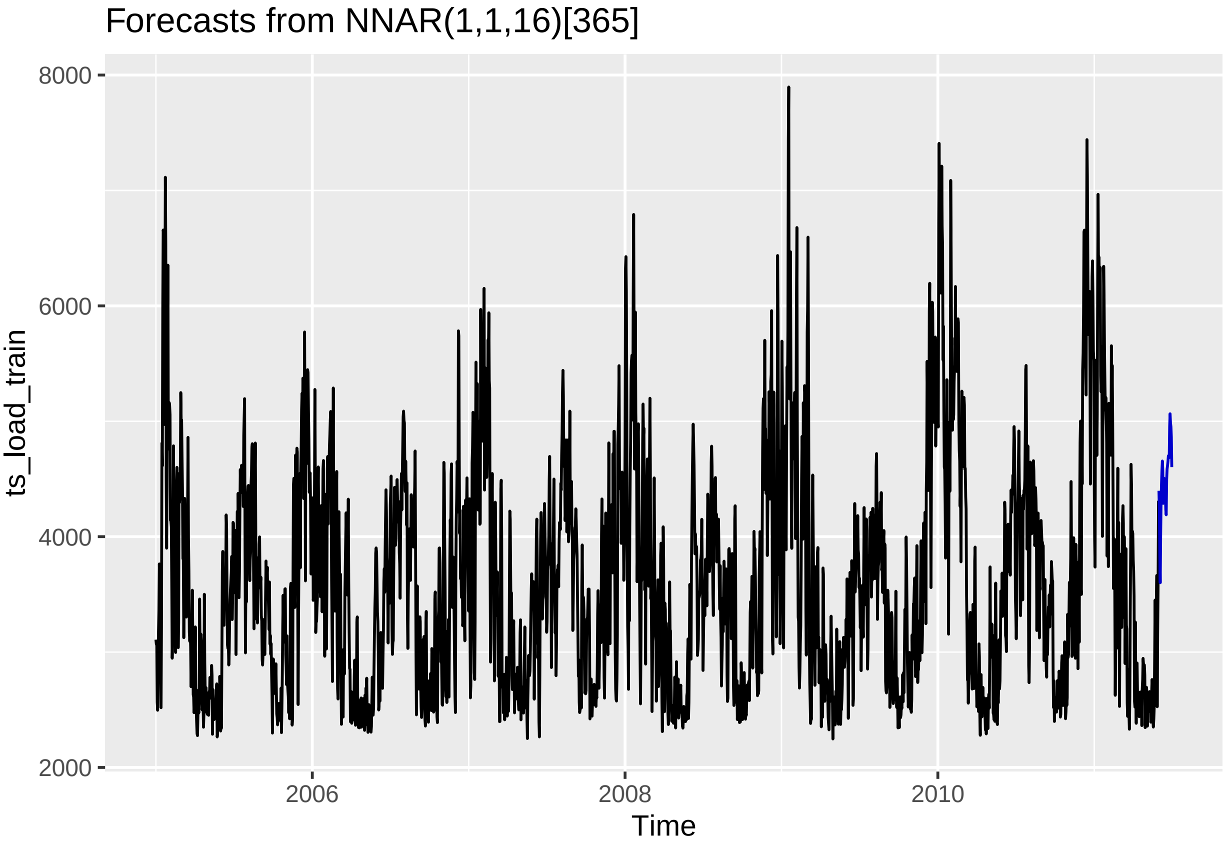
```
##                 ME     RMSE      MAE      MPE     MAPE      ACF1 Theil's U
## Test set 359.5194 533.8021 429.3158 8.645433 10.92958 0.6040277  1.575357
```

```
#dim(ARIMA_fit_tp_hum$xreg)
#dim(temp_hum_regressors)
#dim(temp_hum_regressors_fc)
#dim(ARIMA_fit_tp$xreg)
#dim(temp_regressor_fc)
#dim(ARIMA_fc_tp_hum$xreg)
```

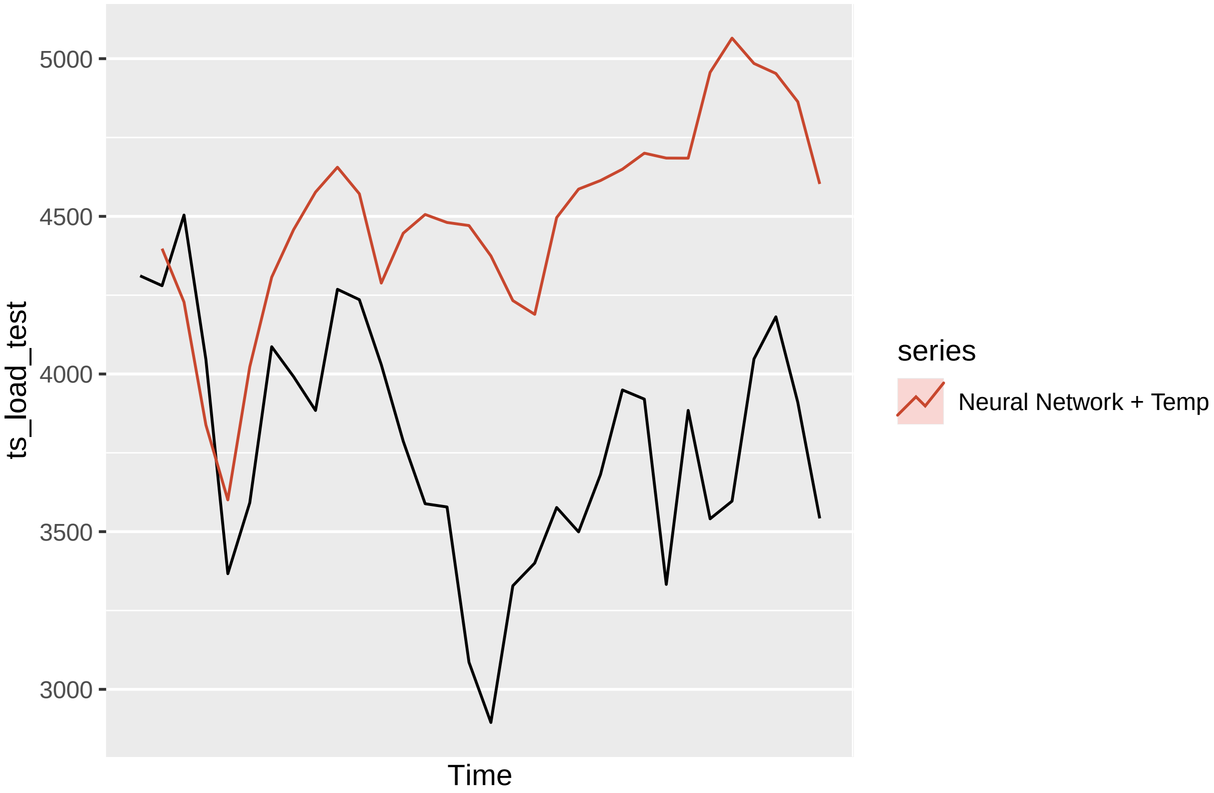
```
#temp_regressor_fc
```

```
#temp_hum_regressors_fc
```

```
## NN + temp
NN_fit_tp <- nnetar(ts_load_train,p=1,P=1,xreg=temp_regressor)
NN_fc_tp <- forecast(NN_fit_tp,h=31, xreg=temp_regressor_fc)
autoplot(NN_fc_tp)
```



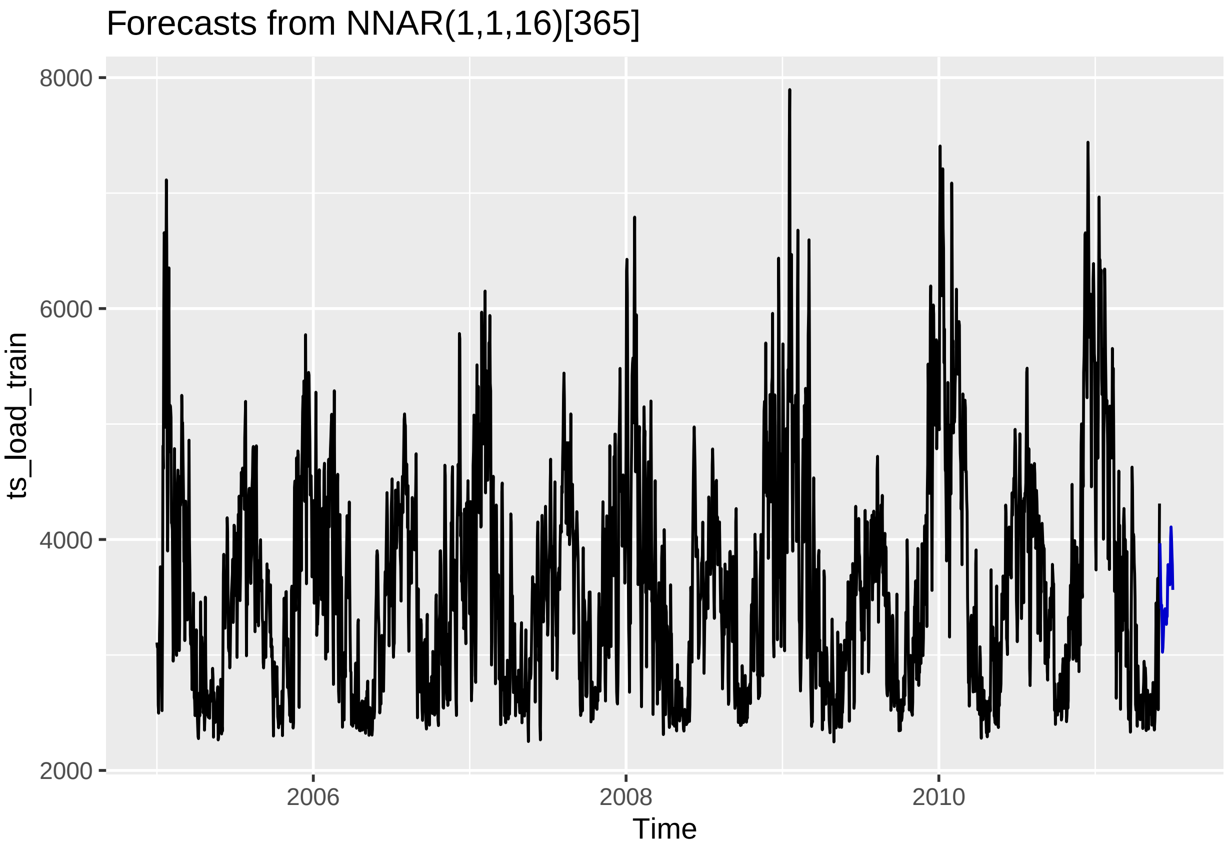
```
autoplot(ts_load_test) +
  autolayer(NN_fc_tp, series="Neural Network + Temp",PI=FALSE)
```



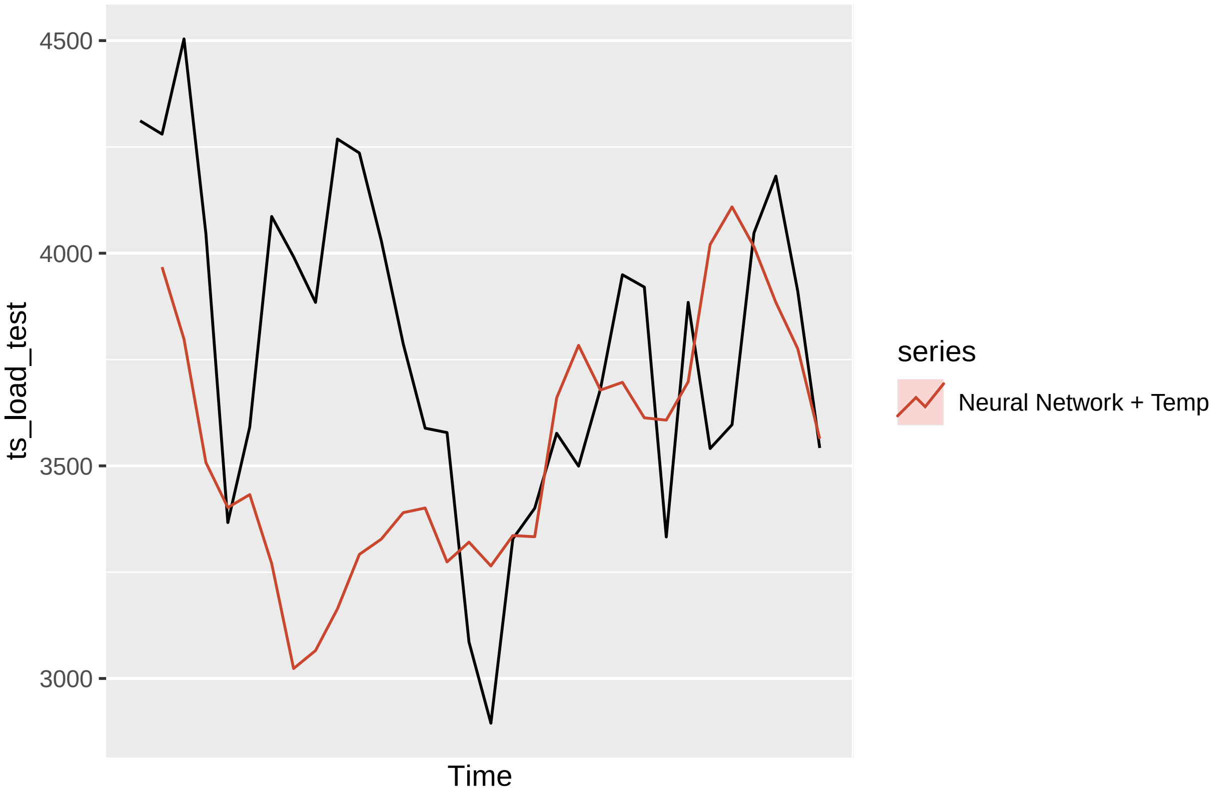
```
NN_scores_tp <- accuracy(NN_fc_tp$mean,ts_load_test)
print(NN_scores_tp)
```

```
##                  ME     RMSE      MAE       MPE     MAPE      ACF1 Theil's U
## Test set -737.8672 865.4101 769.0658 -20.67495 21.40129 0.7277396  2.799905
```

```
## NN + hum
NN_fit_hum <- nnetar(ts_load_train,p=1,P=1,xreg=hum_regressor)
NN_fc_hum <- forecast(NN_fit_hum,h=31, xreg=hum_regressor_fc)
autoplot(NN_fc_hum)
```



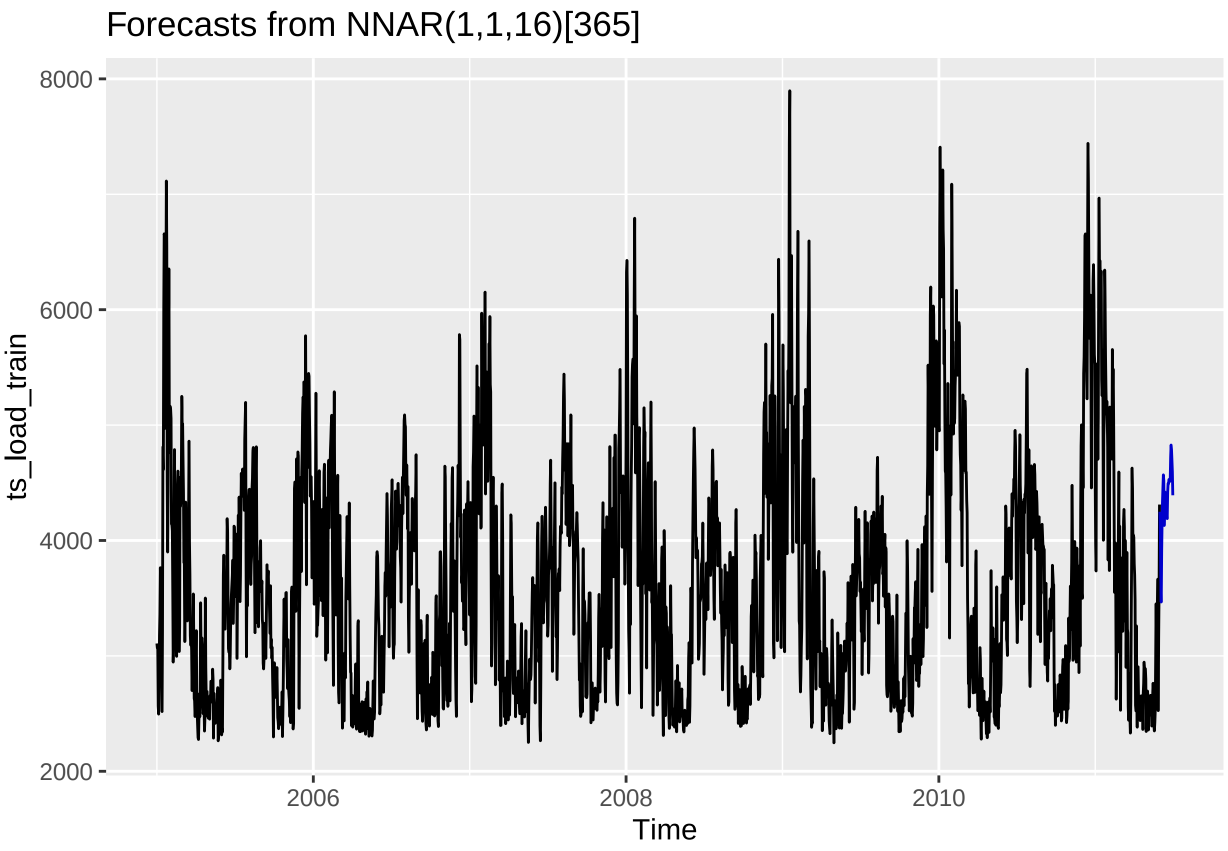
```r
autoplot(ts_load_test) +
  autolayer(NN_fc_hum, series="Neural Network + Temp",PI=FALSE)
```



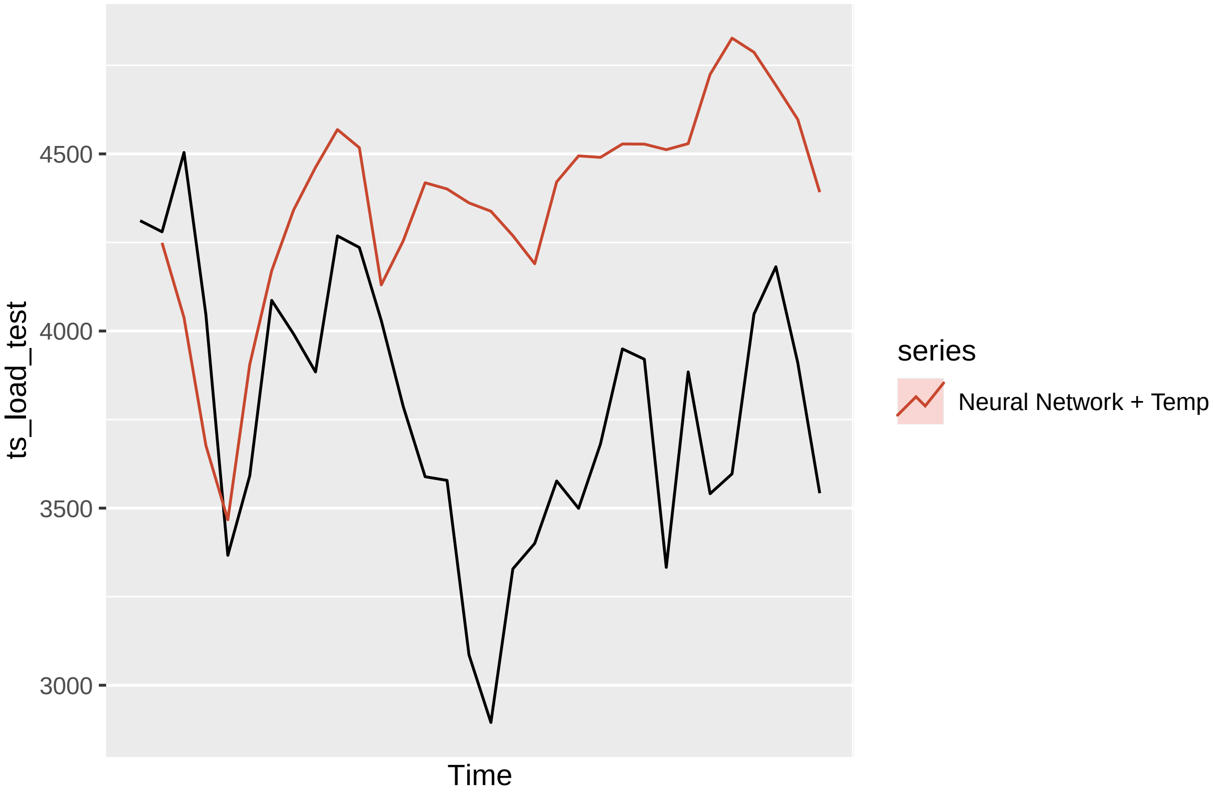
```r
NN_scores_hum <- accuracy(NN_fc_hum$mean,ts_load_test)
print(NN_scores_hum)
```

```
##                 ME     RMSE      MAE      MPE     MAPE      ACF1 Theil's U
## Test set 223.7758 482.9094 372.3726 5.146192 9.581892 0.6932434  1.412929
```

```r
## NN + temp + hum
NN_fit_tp_hum <- nnetar(ts_load_train,p=1,P=1,xreg=temp_hum_regressors)
NN_fc_tp_hum <- forecast(NN_fit_tp_hum,h=31, xreg=temp_hum_regressors_fc)
autoplot(NN_fc_tp_hum)
```



```r
autoplot(ts_load_test) +
  autolayer(NN_fc_tp_hum, series="Neural Network + Temp",PI=FALSE)
```



```
NN_scores_tp_hum <- accuracy(NN_fc_tp_hum$mean,ts_load_test)
print(NN_scores_tp_hum)
```

```
##                  ME     RMSE      MAE       MPE     MAPE      ACF1 Theil's U
## Test set -602.267 755.4622 658.1725 -17.10805 18.41148 0.7396114  2.474469
```

```
# print the scores in a table
scores <- rbind(ARIMA_scores_tp, ARIMA_scores_tp_hum, NN_scores_tp, NN_scores_hum, NN_scores_tp_hum)
rownames(scores) <- c("ARIMA with temperature", "ARIMA with temperature and humidity", "Neural Network w
print(scores)
```

```
##                                                   ME     RMSE      MAE        MPE
## ARIMA with temperature                      369.2936 541.7948 434.5470   8.907236
## ARIMA with temperature and humidity         359.5194 533.8021 429.3158   8.645433
## Neural Network with temperature            -737.8672 865.4101 769.0658 -20.674953
## Neural Network with humidity(best)          223.7758 482.9094 372.3726   5.146192
## Neural Network with temperature humidity -602.2670 755.4622 658.1725 -17.108050
##                                                 MAPE      ACF1 Theil's U
## ARIMA with temperature                     11.052101 0.6046054  1.598575
## ARIMA with temperature and humidity        10.929584 0.6040277  1.575357
## Neural Network with temperature            21.401292 0.7277396  2.799905
## Neural Network with humidity(best)          9.581892 0.6932434  1.412929
## Neural Network with temperature humidity 18.411478 0.7396114  2.474469
```

```
# Combine msts_oil and msts_oil_test into one multi-seasonal time series
ts_load_pd <- msts(c(ts_load_train, ts_load_test), seasonal.periods = c(7, 365.25))
# Combine msts_oil and msts_oil_test into one multi-seasonal time series
ts_temp_pd <- msts(c(ts_temp_train, ts_temp_test), seasonal.periods = c(7, 365.25))
ts_temp_pd <- subset(ts_temp_pd,end=length(ts_load_pd))
# Combine msts_bitcoin and msts_bitcoin_test into one multi-seasonal time series
```

```r
ts_hum_pd <- msts(c(ts_hum_train, ts_hum_test), seasonal.periods = c(7, 365.25))
ts_hum_pd <- subset(ts_hum_pd,end=length(ts_load_pd))
temp_regressor_pd<- as.matrix(data.frame(fourier(ts_load_pd,K=c(2,12)), "temp"= ts_temp_pd))
temp_fc_pd<-forecast(ts_temp_pd,h=31)
temp_regressor_fc_pd<-as.matrix(data.frame(fourier(ts_load_pd,K=c(2,12),h=31),"temp"=temp_fc_pd$mean))


hum_regressor<- as.matrix(data.frame(fourier(ts_load_pd, K=c(2,12)), "hum"=ts_hum_pd))
hum_fc_pd<-forecast(ts_hum_pd,h=31)
hum_regressor_fc_pd<-as.matrix(data.frame(fourier(ts_load_pd,K=c(2,12),h=31),"hum"= hum_fc_pd$mean))


temp_hum_regressors_pd<- as.matrix(data.frame(fourier(ts_load_pd, K=c(2,12)), "temp"= ts_temp_pd, "hum"=
temp_hum_regressors_fc_pd<-as.matrix(data.frame(fourier(ts_load_pd,K=c(2,12),h=31), "temp"=temp_fc_pd$me

forecast_result <- forecast(ARIMA_fit_tp,xreg = temp_regressor_fc_pd, h = 31)

# Print the forecasted values
#print(forecast_result)
```

```r
# Define the start date and end date
start_date <- as.Date("2011-07-01")
end_date <- as.Date("2011-07-31")

# Generate a sequence of dates from start_date to end_date
forecast_dates <- seq(start_date, end_date, by = "day")

forecast_load <- forecast_result$mean

# Combine dates and load values into a data frame
forecast_df <- data.frame(date = forecast_dates, load = forecast_load)

# Write the data frame to a CSV file
#write.csv(forecast_df, file = "forecast results arima tp.csv", row.names = FALSE)
```

```r
forecast_result <- forecast(ARIMA_fit_tp_hum,xreg = temp_hum_regressors_fc_pd, h = 31)
# Define the start date and end date
start_date <- as.Date("2011-07-01")
end_date <- as.Date("2011-07-31")

# Generate a sequence of dates from start_date to end_date
forecast_dates <- seq(start_date, end_date, by = "day")

forecast_load <- forecast_result$mean

# Combine dates and load values into a data frame
forecast_df <- data.frame(date = forecast_dates, load = forecast_load)

# Write the data frame to a CSV file
#write.csv(forecast_df, file = "forecast results arima tp&hum.csv", row.names = FALSE)
```

```r
forecast_result <- forecast(NN_fit_tp,xreg = temp_regressor_fc_pd, h = 31)
```

```r
# Define the start date and end date
start_date <- as.Date("2011-07-01")
end_date <- as.Date("2011-07-31")

# Generate a sequence of dates from start_date to end_date
forecast_dates <- seq(start_date, end_date, by = "day")

forecast_load <- forecast_result$mean

# Combine dates and load values into a data frame
forecast_df <- data.frame(date = forecast_dates, load = forecast_load)

# Write the data frame to a CSV file
#write.csv(forecast_df, file = "forecast results NN tp.csv", row.names = FALSE)
```

```r
##with best result
forecast_result <- forecast(NN_fit_hum,xreg = hum_regressor_fc_pd, h = 31)
# Define the start date and end date
start_date <- as.Date("2011-07-01")
end_date <- as.Date("2011-07-31")

# Generate a sequence of dates from start_date to end_date
forecast_dates <- seq(start_date, end_date, by = "day")

forecast_load <- forecast_result$mean

# Combine dates and load values into a data frame
forecast_df <- data.frame(date = forecast_dates, load = forecast_load)

# Write the data frame to a CSV file
#write.csv(forecast_df, file = "forecast results NN hum.csv", row.names = FALSE)
```

```r
forecast_result <- forecast(NN_fit_tp_hum,xreg = temp_hum_regressors_fc_pd, h = 31)
# Define the start date and end date
start_date <- as.Date("2011-07-01")
end_date <- as.Date("2011-07-31")

# Generate a sequence of dates from start_date to end_date
forecast_dates <- seq(start_date, end_date, by = "day")

forecast_load <- forecast_result$mean

# Combine dates and load values into a data frame
forecast_df <- data.frame(date = forecast_dates, load = forecast_load)

# Write the data frame to a CSV file
#write.csv(forecast_df, file = "forecast results NN tp&hum.csv", row.names = FALSE)
```