R Notebook

https://github.com/hxia5/XiaGupta_ENV797_TSA_Competition_S2024

Haochong Xia, Ayush Gupta

2024-04-26

1. Data preprocessing

```
library(readxl)
suppressPackageStartupMessages(library(readxl))

# Load the Excel file into a data frame
data <- read_excel("/Users/xiahaochong/Desktop/797 Time Series/XiaGupta_ENV797_TSA_Competition_S2024/da


# Read the Excel file
temperature_data <- read_excel('/Users/xiahaochong/Desktop/797 Time Series/XiaGupta_ENV797_TSA_Competit


relative_humidity_data <- read_excel("/Users/xiahaochong/Desktop/797 Time Series/XiaGupta_ENV797_TSA_Co
```

```
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
```

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

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

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

```
suppressPackageStartupMessages(library(lubridate))
suppressPackageStartupMessages(library(magrittr))
suppressPackageStartupMessages(library(dplyr))

load <- data %>%
  mutate(date = ymd(date)) %>% #converts date format
```

```r
  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
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)
```

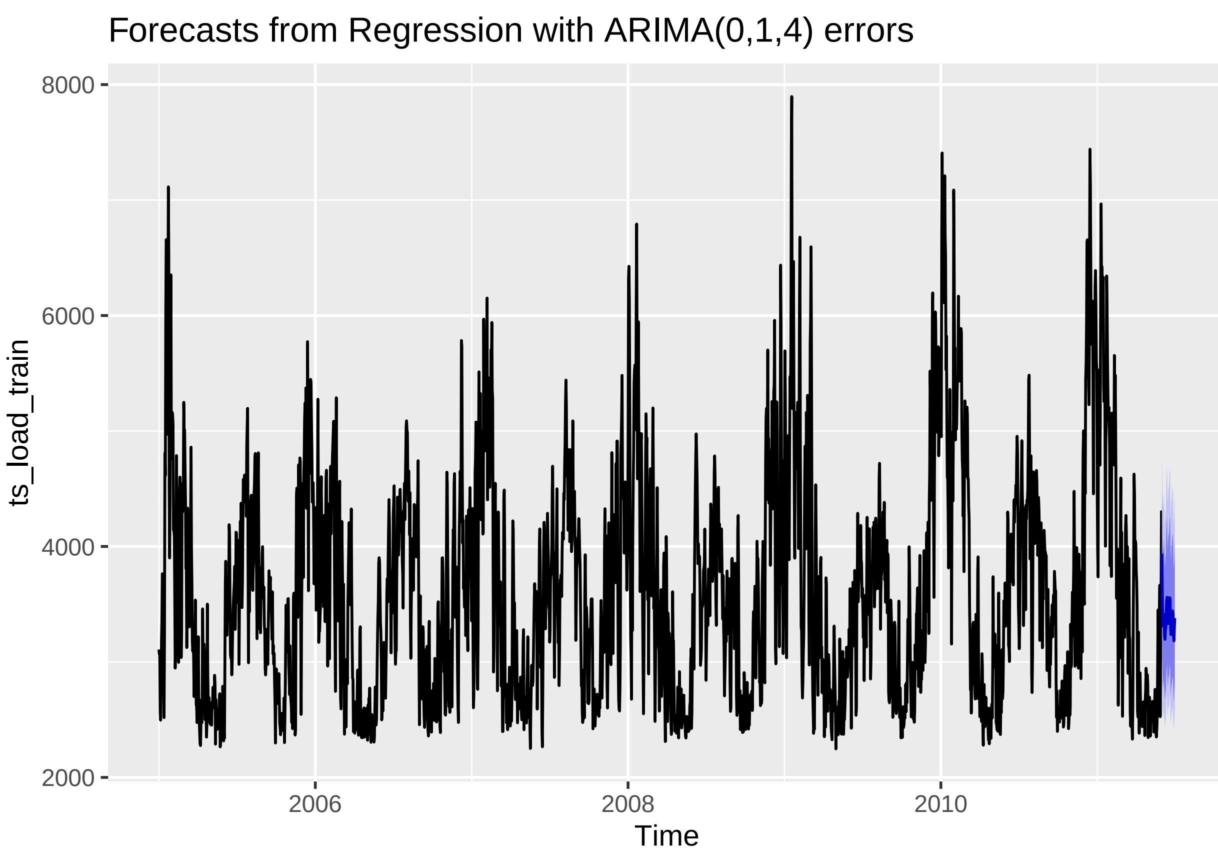
2. Creating time series and regressors

```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)
```

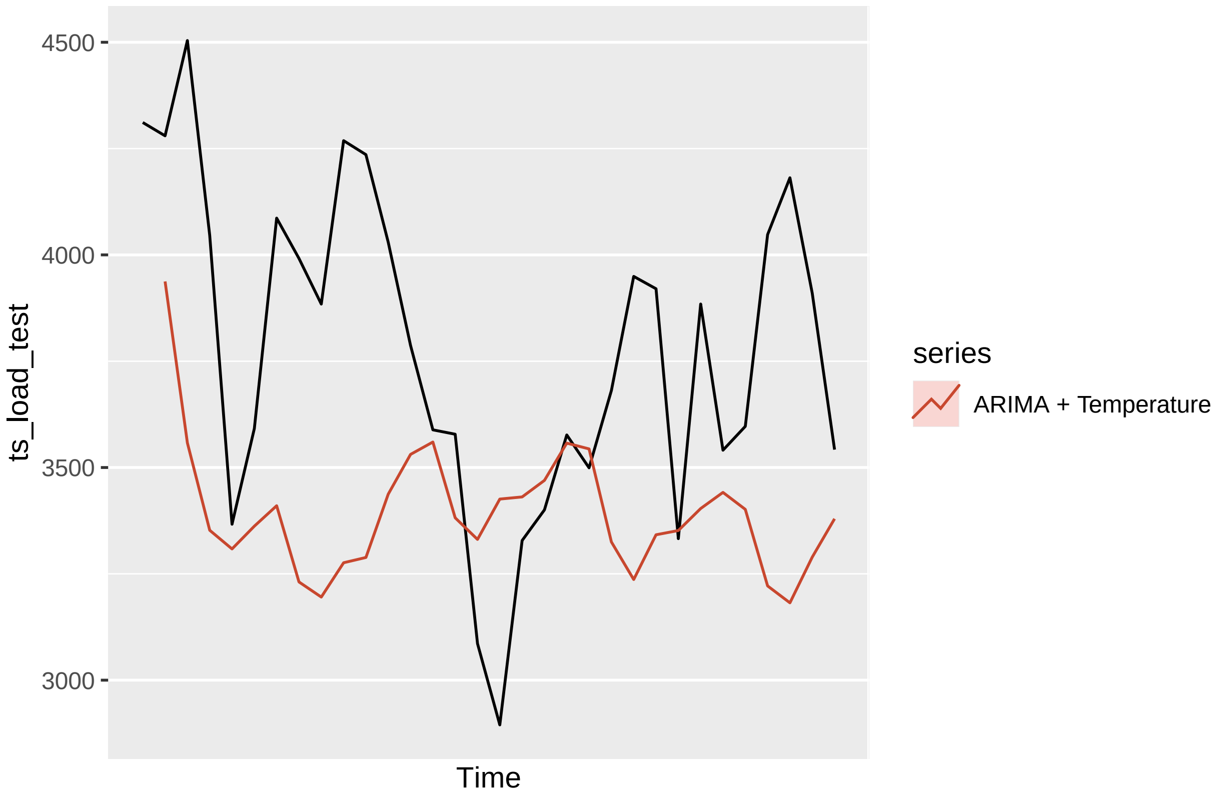
```
ts_hum <- msts(hum$d_mean,seasonal.periods=c(7,365.25),start=c(2005,01,01))
ts_hum_train <- subset(ts_hum,end =length(ts_load)-31)
ts_hum_test <- subset(ts_hum,start =length(ts_load)-31)

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
```

3. Model fitting

```
#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)
```



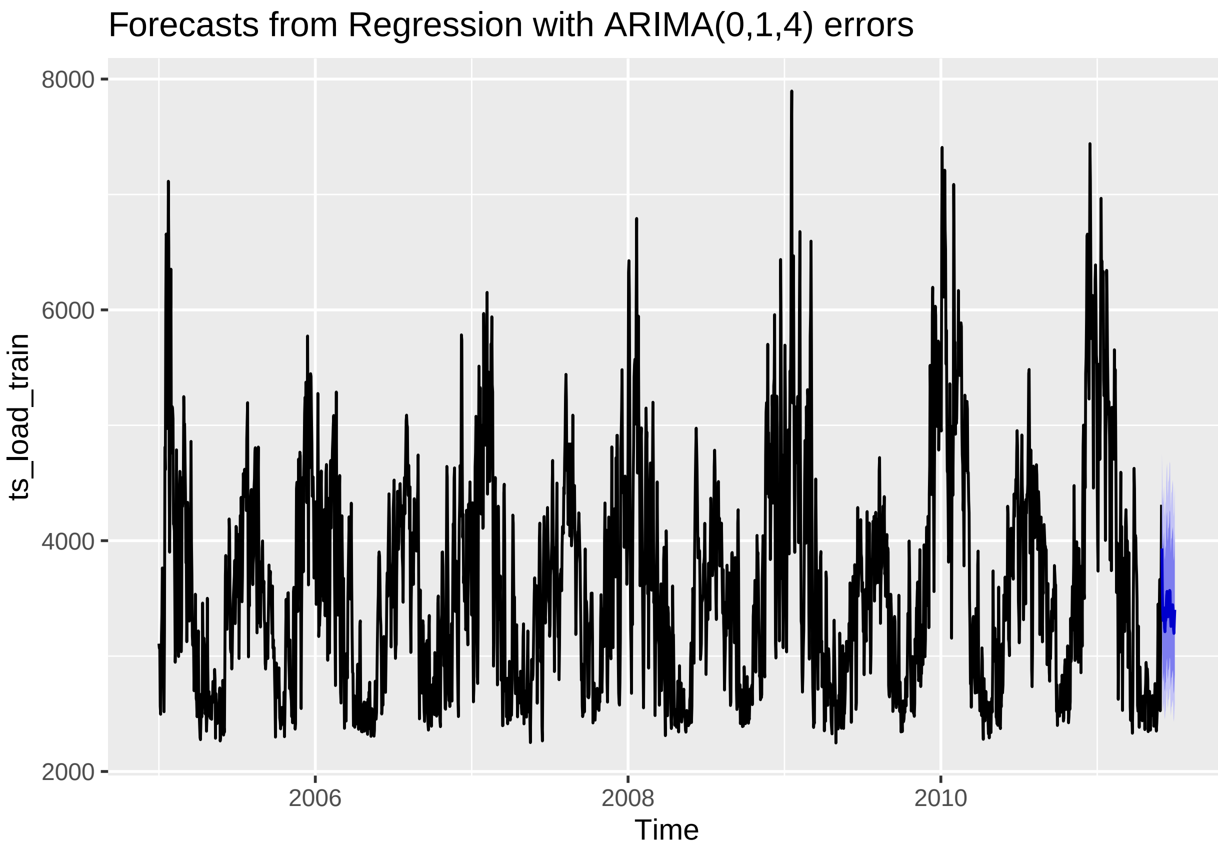
```r
autoplot(ts_load_test) +
  autolayer(ARIMA_fc_tp, series="ARIMA + Temperature",PI=FALSE)
```




```r
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
```

```r
#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)
```




```r
autoplot(ts_load_test) +
  autolayer(ARIMA_fc_tp_hum, series="ARIMA + Temperature + Humidity",PI=FALSE)
```

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

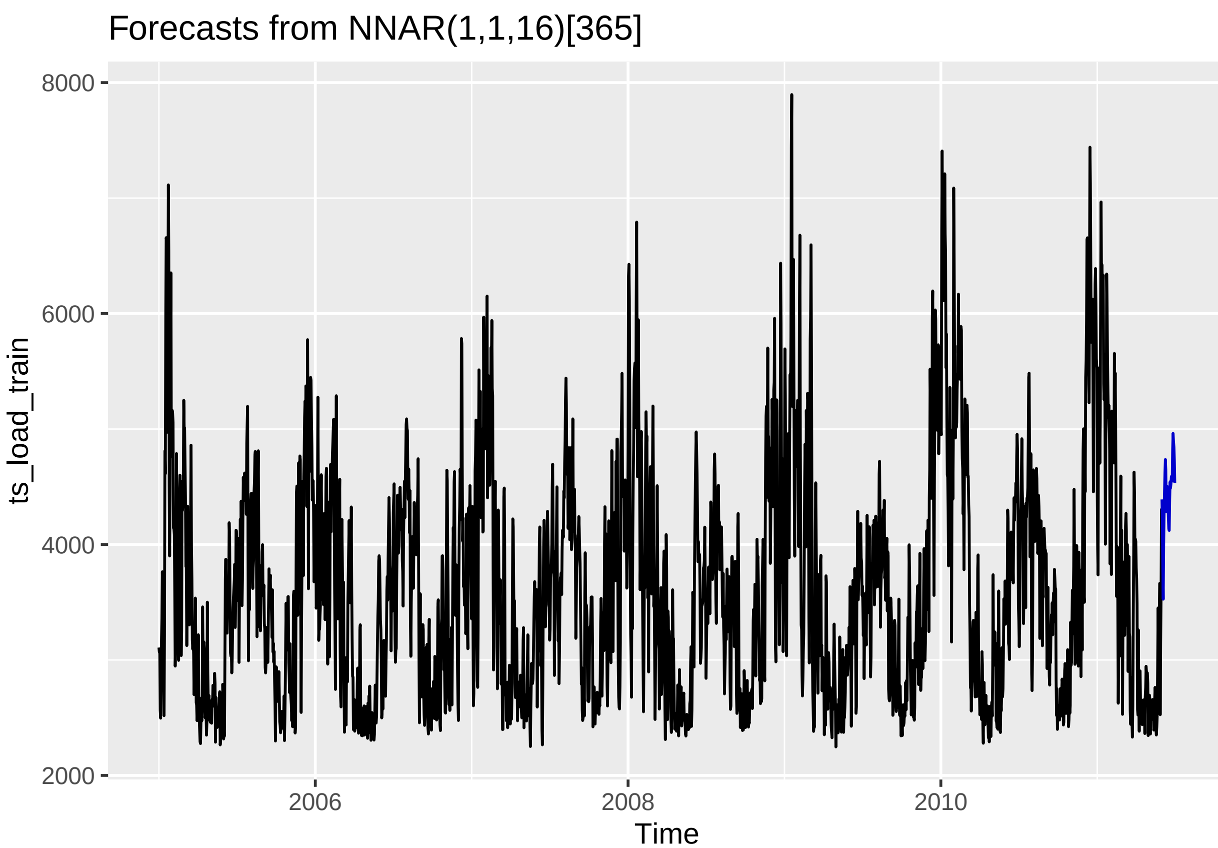
```
##                ME     RMSE      MAE      MPE     MAPE      ACF1 Theil's U
## Test set 359.5194 533.8021 429.3158 8.645433 10.92958 0.6040277  1.575357
```

```r
#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)
```

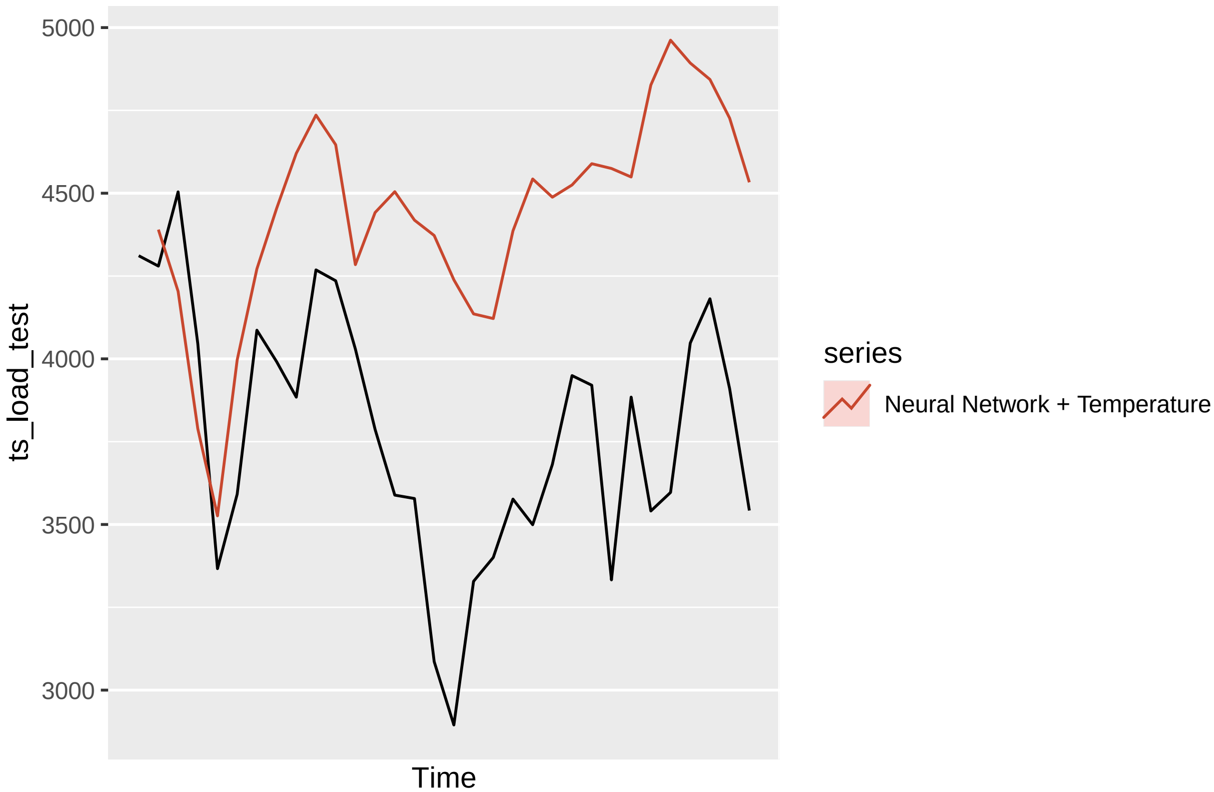
```r
#temp_regressor_fc
```

```r
#temp_hum_regressors_fc
```

```r
## 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)
```



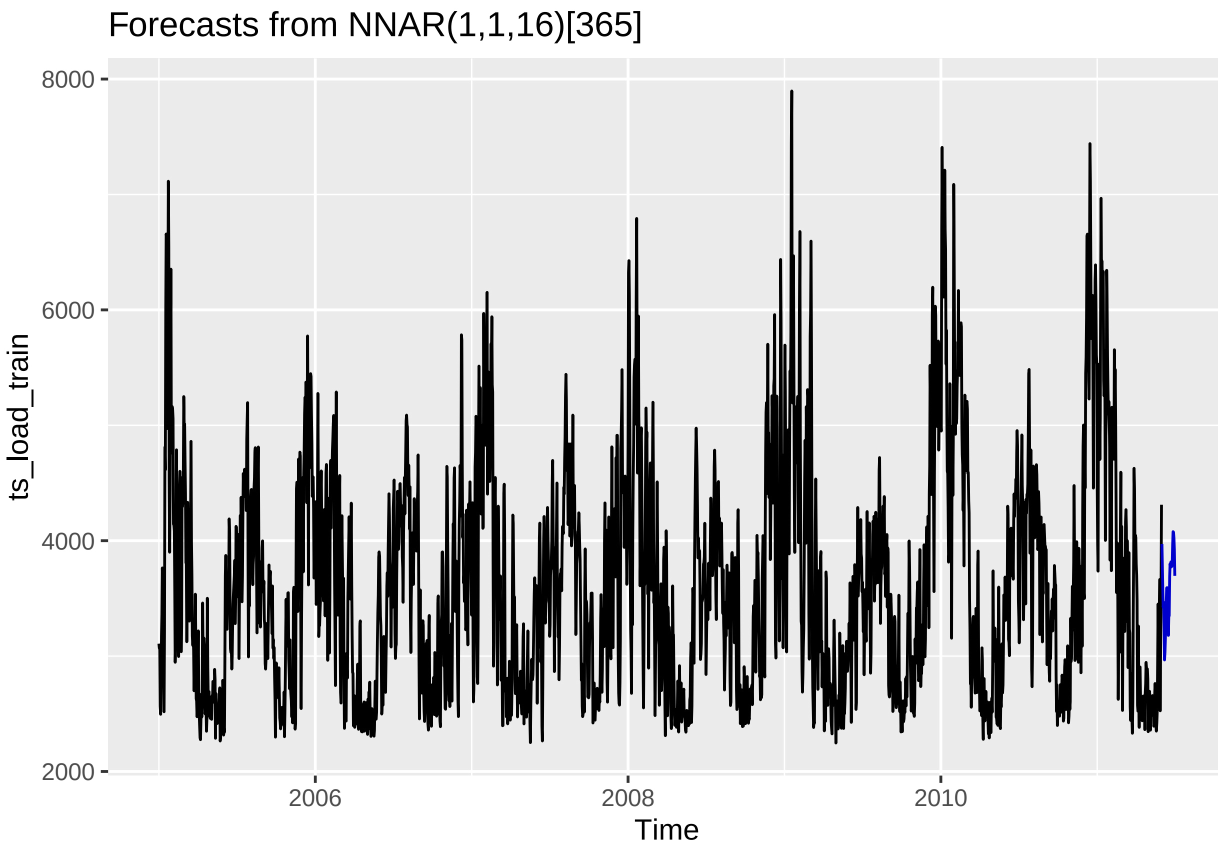

```r
autoplot(ts_load_test) +
  autolayer(NN_fc_tp, series="Neural Network + Temperature",PI=FALSE)
```



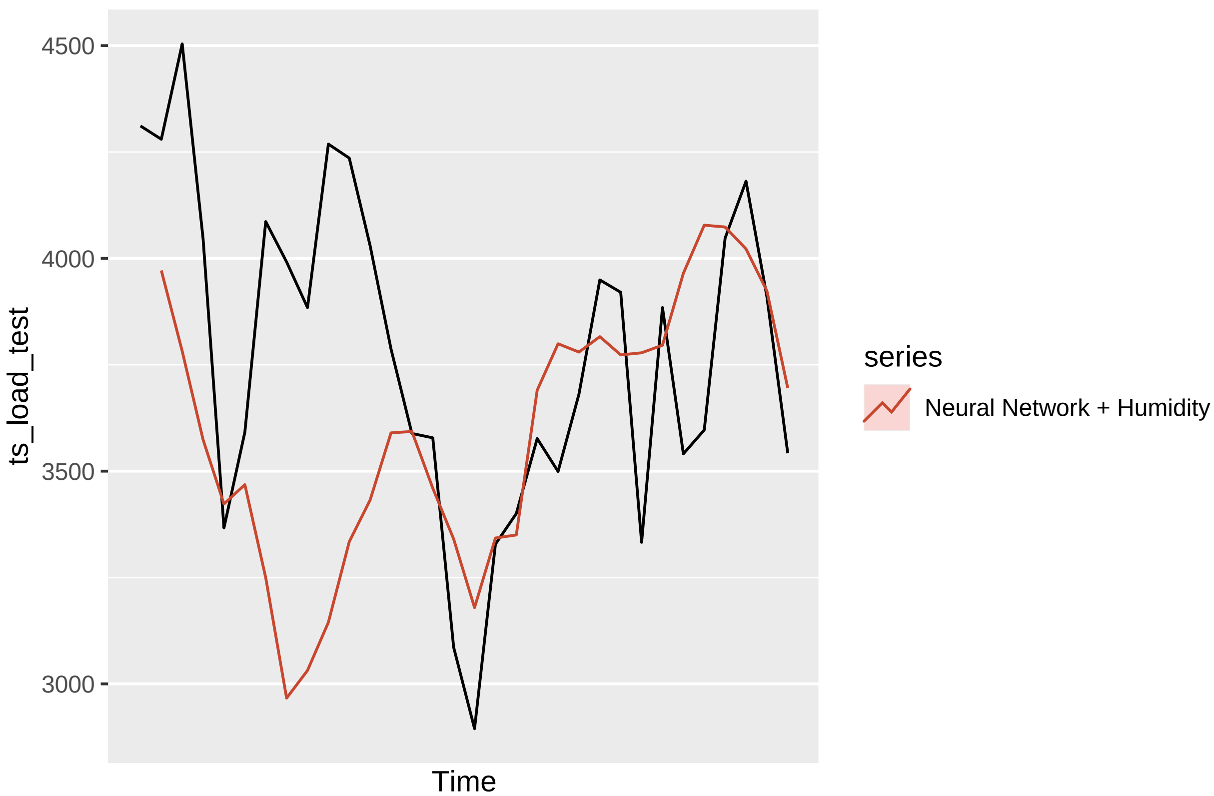

```
NN_scores_tp <- accuracy(NN_fc_tp$mean,ts_load_test)
print(NN_scores_tp)
```

```
##                 ME     RMSE      MAE       MPE     MAPE      ACF1 Theil's U
## Test set -676.7746 796.8262 712.7161 -18.95783 19.79746 0.6855336  2.570492
```

```
## 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)
```



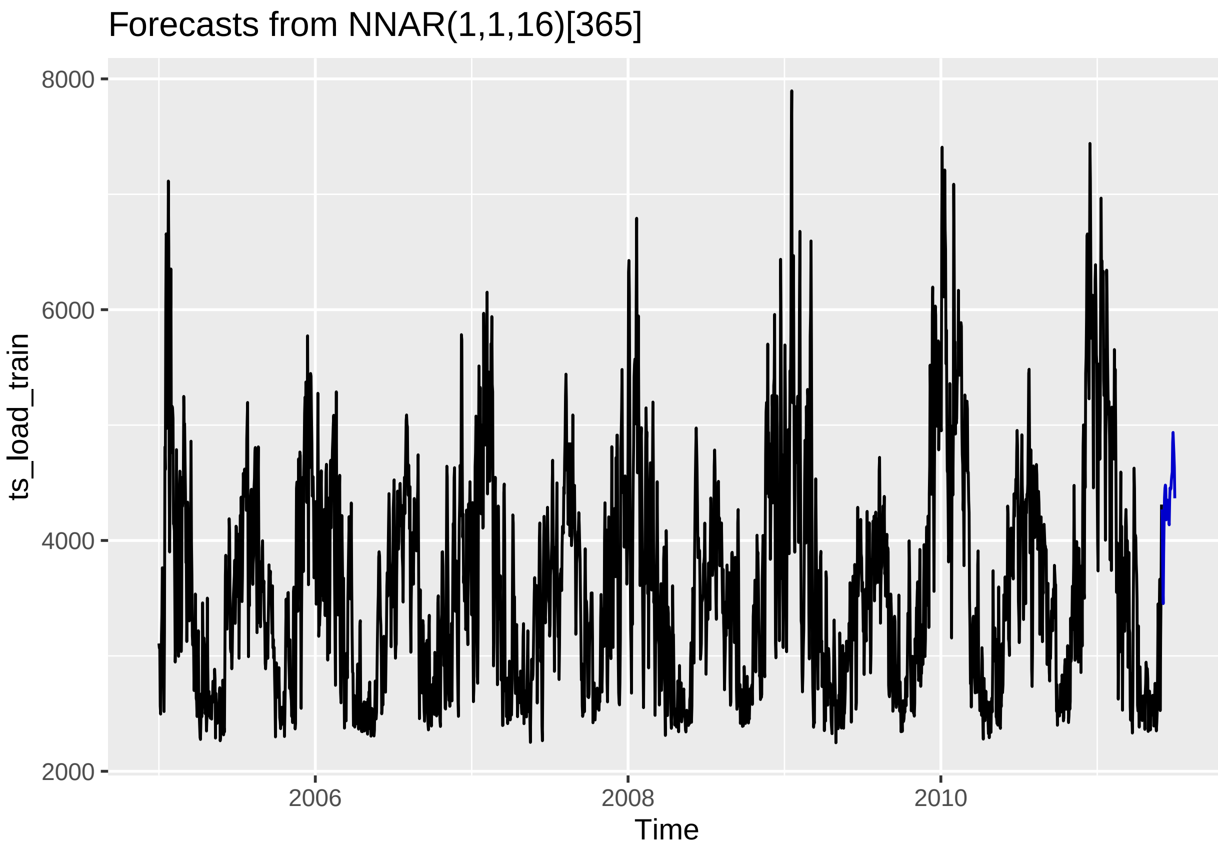

```r
autoplot(ts_load_test) +
  autolayer(NN_fc_hum, series="Neural Network + Humidity",PI=FALSE)
```



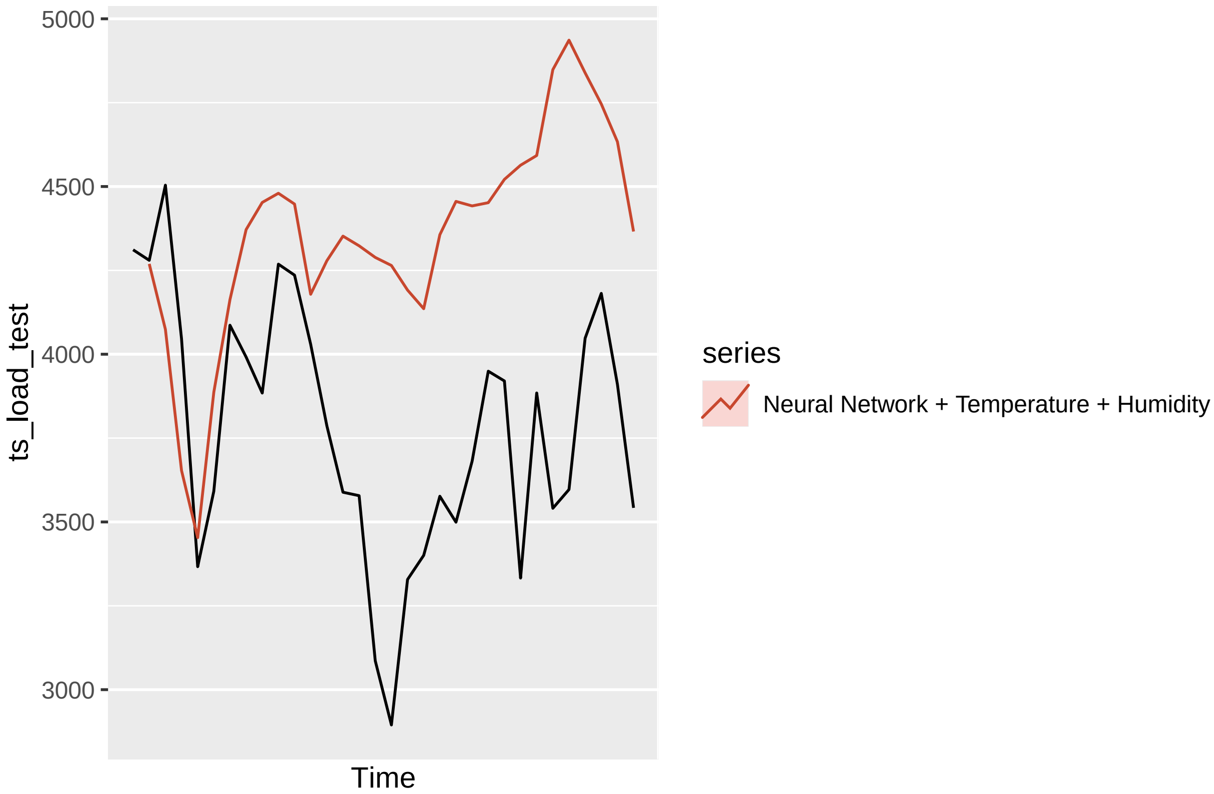
```r
NN_scores_hum <- accuracy(NN_fc_hum$mean,ts_load_test)
print(NN_scores_hum)
```

```
##                ME     RMSE      MAE      MPE     MAPE      ACF1 Theil's U
## Test set 167.2941 467.9781 339.5941 3.654611 8.737851 0.7264313  1.366386
```

```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 + Temperature + Humidity",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 -593.7073 748.9402 647.4636 -16.83396 18.09193 0.7335115   2.43851
```

4. Score comparison

```
library(knitr)
suppressPackageStartupMessages(library(knitr))
ARIMA_scores_tp_RMSE <- ARIMA_scores_tp[1, 2]
# Combine scores into a data frame
scores_df <- data.frame(
  Method = c("ARIMA with temperature",
             "ARIMA with temperature and humidity",
             "Neural Network with temperature",
             "Neural Network with humidity(best)",
             "Neural Network with temperature humidity"),
  ME = c(ARIMA_scores_tp[1, 1], ARIMA_scores_tp_hum[1, 1], NN_scores_tp[1, 1], NN_scores_hum[1, 1], NN_s
  RMSE = c(ARIMA_scores_tp[1, 2], ARIMA_scores_tp_hum[1, 2], NN_scores_tp[1, 2], NN_scores_hum[1, 2], NN
  MAE = c(ARIMA_scores_tp[1, 3], ARIMA_scores_tp_hum[1, 3], NN_scores_tp[1, 3], NN_scores_hum[1, 3], NN_
  MAPE = c(ARIMA_scores_tp[1, 4], ARIMA_scores_tp_hum[1, 4], NN_scores_tp[1, 4], NN_scores_hum[1, 4], NN
  ACF1 = c(ARIMA_scores_tp[1, 5], ARIMA_scores_tp_hum[1, 5], NN_scores_tp[1, 5], NN_scores_hum[1, 5], NN
  Theil = c(ARIMA_scores_tp[1, 6], ARIMA_scores_tp_hum[1, 6], NN_scores_tp[1, 6], NN_scores_hum[1, 6], N
)

# Print formatted table
kable(scores_df, format = "markdown")
```

Method	ME	RMSE	MAE	MAPE	ACF1	Theil
ARIMA with temperature	369.2936	541.7948	434.5470	8.907236	11.052100	0.6046054
ARIMA with temperature and humidity	359.5194	533.8021	429.3158	8.645433	10.929583	0.6040277
Neural Network with temperature	-676.7746	796.8262	712.7161	-18.957832	19.797456	0.6855336
Neural Network with humidity(best)	167.2941	467.9781	339.5941	3.654611	8.737851	0.7264313
Neural Network with temperature humidity	-593.7073	748.9402	647.4636	-16.833957	18.091928	0.7335115

5. Forecasting and creating submissions

```r
# 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
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_pd<- 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
```

```r
ARIMA_fit_tp_pd<-auto.arima(ts_load_pd,seasonal= FALSE, lambda=0,xreg=temp_regressor_pd)
forecast_result <- forecast(ARIMA_fit_tp_pd,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
ARIMA_fit_tp_hum_pd<-auto.arima(ts_load_pd,seasonal= FALSE, lambda=0,xreg=temp_hum_regressors_pd)
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
NN_fit_temp_pd <- nnetar(ts_load_pd,p=1,P=1,xreg=temp_regressor_pd)
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
NN_fit_hum_pd <- nnetar(ts_load_pd,p=1,P=1,xreg=hum_regressor_pd)
forecast_result <- forecast(NN_fit_hum_pd,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
NN_fit_temp_hum_pd <- nnetar(ts_load_pd,p=1,P=1,xreg=temp_hum_regressors_pd)
forecast_result <- forecast(NN_fit_temp_hum_pd,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)
```