ForecastCompetition

Ben and Yiyan

4/13/2022

Data Cleaning

At this step, we cleaned the load, temperature, and humidity datasets. First, we imported our datasets. Then we averaged the hourly temperature and humidity across all weather stations. Then we cleaned these datasets and hour hourly load data to return daily average temperature and humidity across all weather stations. This process resulted in the following three dataframes.

```
library(readxl)
load <- read_excel("Data/load.xlsx")
humidity <- read_excel("Data/relative_humidity.xlsx")
empty_col2<-("humidity_average")
humidity[,empty_col2]<-NA
humidity$humidity_average<-rowMeans(humidity[,(3:30)])
temp <- read_excel("Data/temperature.xlsx")
empty_col3<-("temp_average")
temp[,empty_col3]<-NA
temp$temp_average<-rowMeans(temp[,(3:30)])
#Separate the year, month, date using a pipe function
humidity_2 <- humidity %>%
  mutate( Date = ymd(date)) %>%
  mutate( Year = year(date),
          Month = month(date),
          Day = day(date)) %>%
  select( Date, Year, Month, Day, humidity_average)
temp_2 <- temp %>%
  mutate( Date = ymd(date)) %>%
  mutate( Year = year(date),
          Month = month(date),
          Day = day(date)) %>%
  select( Date, Year, Month, Day, temp_average)
#Create a data frame with daily observations
humidity_daily <- humidity_2 %>%
  filter( !is.na(humidity_average)) %>%
  group_by(Date,Year,Month,Day) %>% # here we left column with hour out to calculate daily mean
  summarise( daily_humidity_average = mean(humidity_average))
```

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

```
temp_daily <- temp_2 %>%
  filter( !is.na(temp_average)) %>%
```

```
  group_by(Date,Year,Month,Day) %>% # here we left column with hour out to calculate daily mean
  summarise(daily_temp_average = mean(temp_average))
```

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

```
#Create an additional column for daily averages
empty_col <- ("Daily Average")
load[,empty_col] <- NA
load$`Daily Average` <- rowMeans(load[,(3:26)])
load <- load[-c(1,3:26)]
temp_daily<-temp_daily[,-c(2:4)]
humidity_daily<-humidity_daily[,-c(2:4)]
#Display output
head(load, 10)
```

```
## # A tibble: 10 x 2
##    date                `Daily Average`
##    <dttm>                        <dbl>
##  1 2005-01-01 00:00:00           2889.
##  2 2005-01-02 00:00:00           2789.
##  3 2005-01-03 00:00:00           2708.
##  4 2005-01-04 00:00:00           2212.
##  5 2005-01-05 00:00:00           2035.
##  6 2005-01-06 00:00:00           2110.
##  7 2005-01-07 00:00:00           2313.
##  8 2005-01-08 00:00:00           2480.
##  9 2005-01-09 00:00:00           3116.
## 10 2005-01-10 00:00:00           3222.
```

```
head(temp_daily, 10)
```

```
## # A tibble: 10 x 2
## # Groups:   Date [10]
##    Date       daily_temp_average
##    <date>                  <dbl>
##  1 2005-01-01               53.6
##  2 2005-01-02               53.8
##  3 2005-01-03               55.9
##  4 2005-01-04               61.7
##  5 2005-01-05               60.4
##  6 2005-01-06               62.0
##  7 2005-01-07               55.9
##  8 2005-01-08               57.7
##  9 2005-01-09               47.6
## 10 2005-01-10               49.3
```

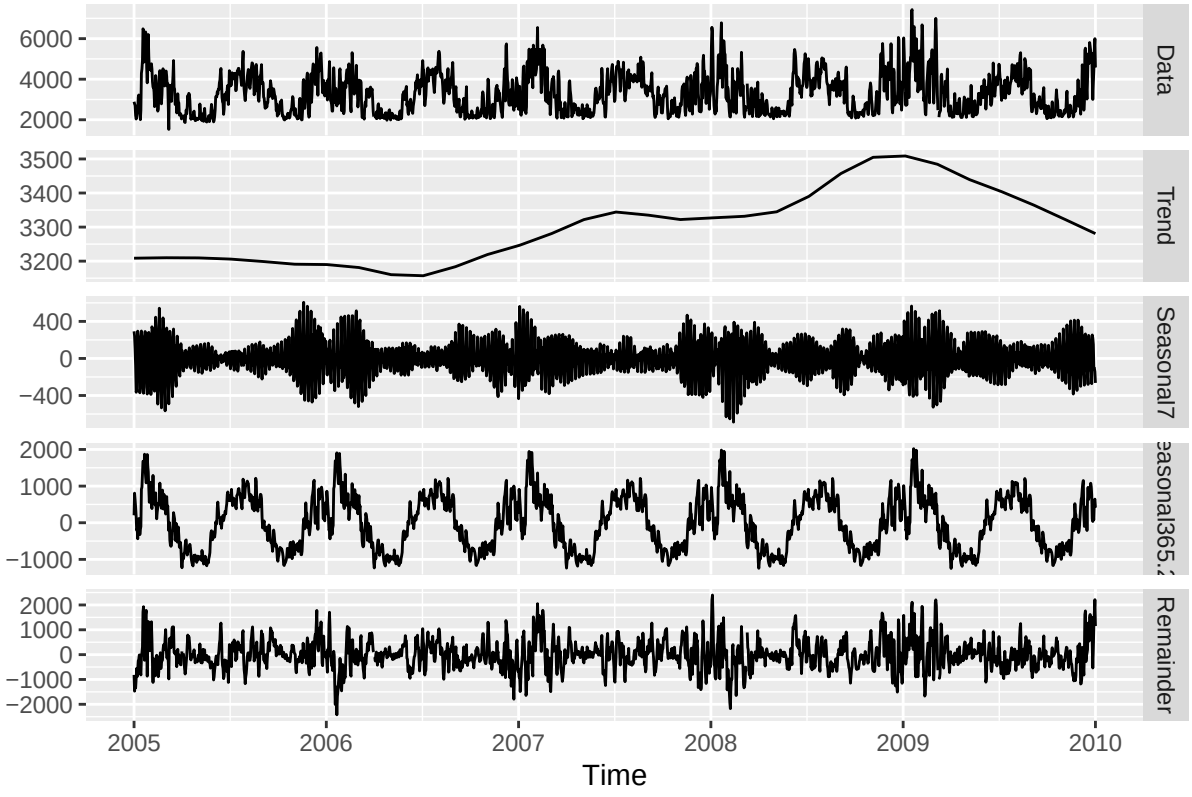
```
head(humidity_daily, 10)
```

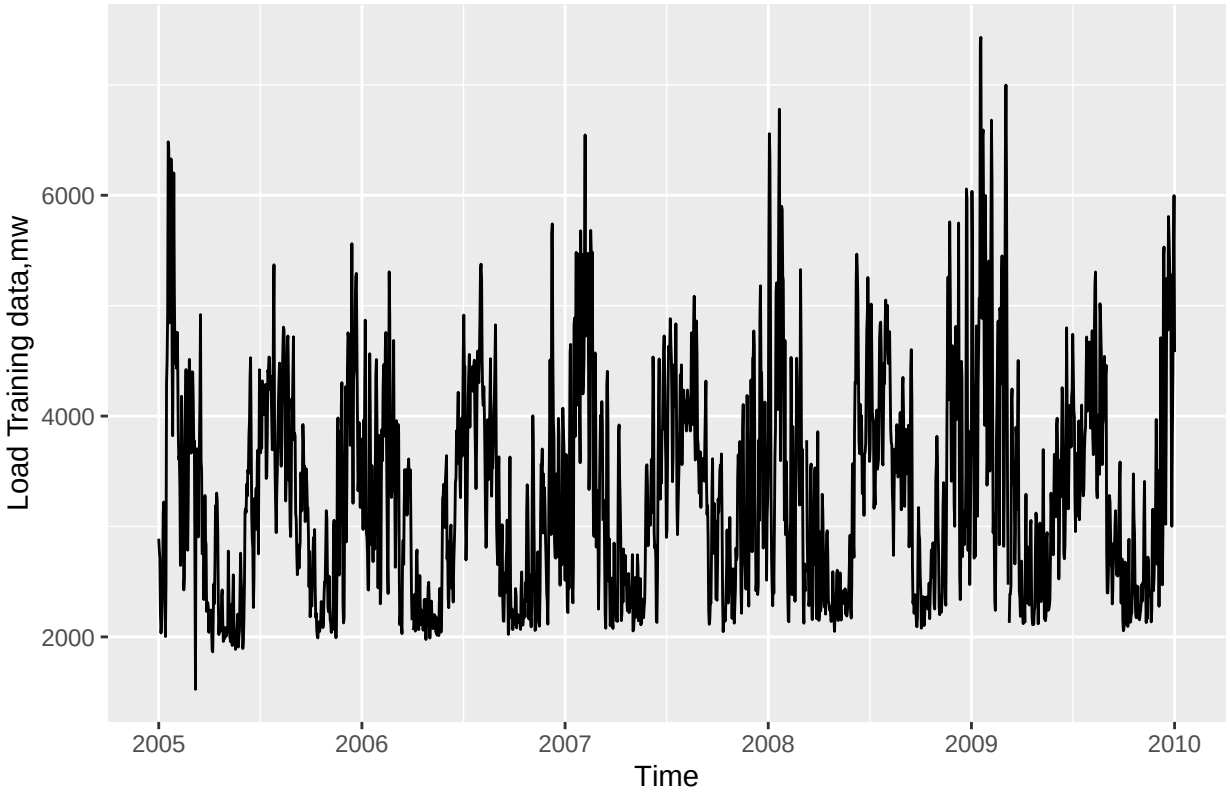
```
## # A tibble: 10 x 2
## # Groups:   Date [10]
##    Date       daily_humidity_average
```

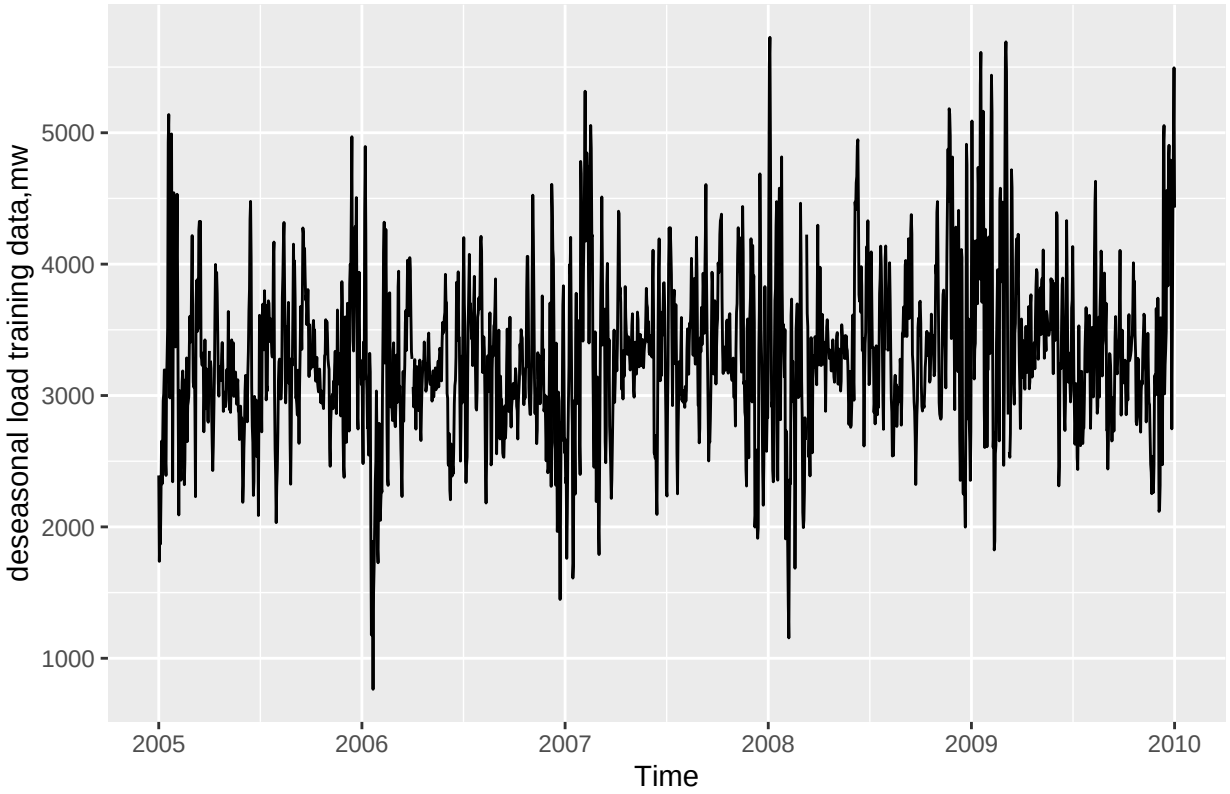
```
##    <date>                     <dbl>
##  1 2005-01-01                  76.7
##  2 2005-01-02                  80.5
##  3 2005-01-03                  81.2
##  4 2005-01-04                  74.8
##  5 2005-01-05                  76.1
##  6 2005-01-06                  78.0
##  7 2005-01-07                  71.7
##  8 2005-01-08                  81.8
##  9 2005-01-09                  72.7
## 10 2005-01-10                  81.1
```

Create ts object and create test and training data

At this step, we create time-series projects for load, temp, and humidity, with weekly and yearly seasonality. The original data are from 2005 to 2010. Then, we created a training (2005-2009) and a testing (Jan, 2010) dataset for each of the three variables. We also decomposed the load data and created the deseasonal load data.

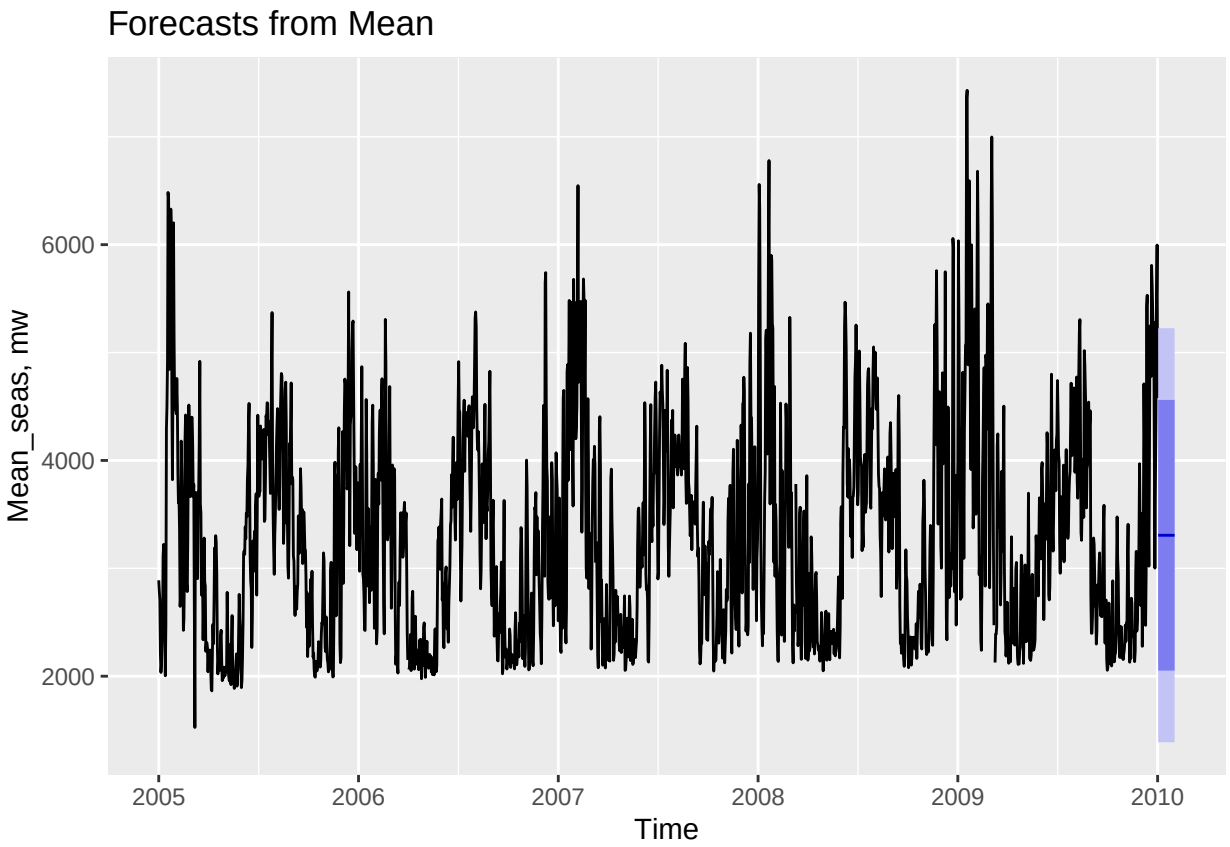



Load Training data,mw
6000
4000
2000
2005
2006
2007
2008
2009
2010
Time

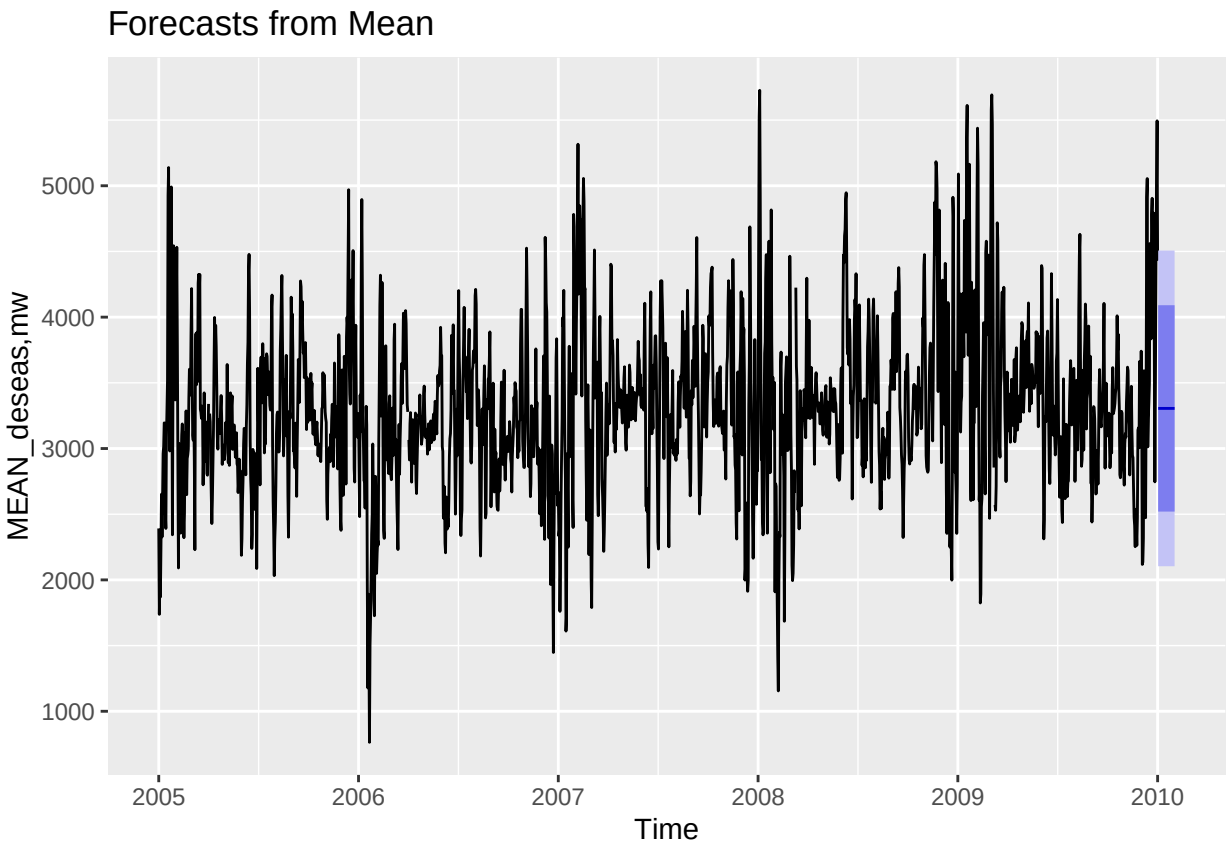


##4 Simple models At this step, we applied 4 simple models to the load training data to generate forecasts for Jan 2010. The 4 models are arithmetic mean on original data, arithmetic mean on deseasonal data, seasonal naïve on original data, and naïve on deseasonal data. Because these forecasting methods are very basic, they do not have very high accuracy.

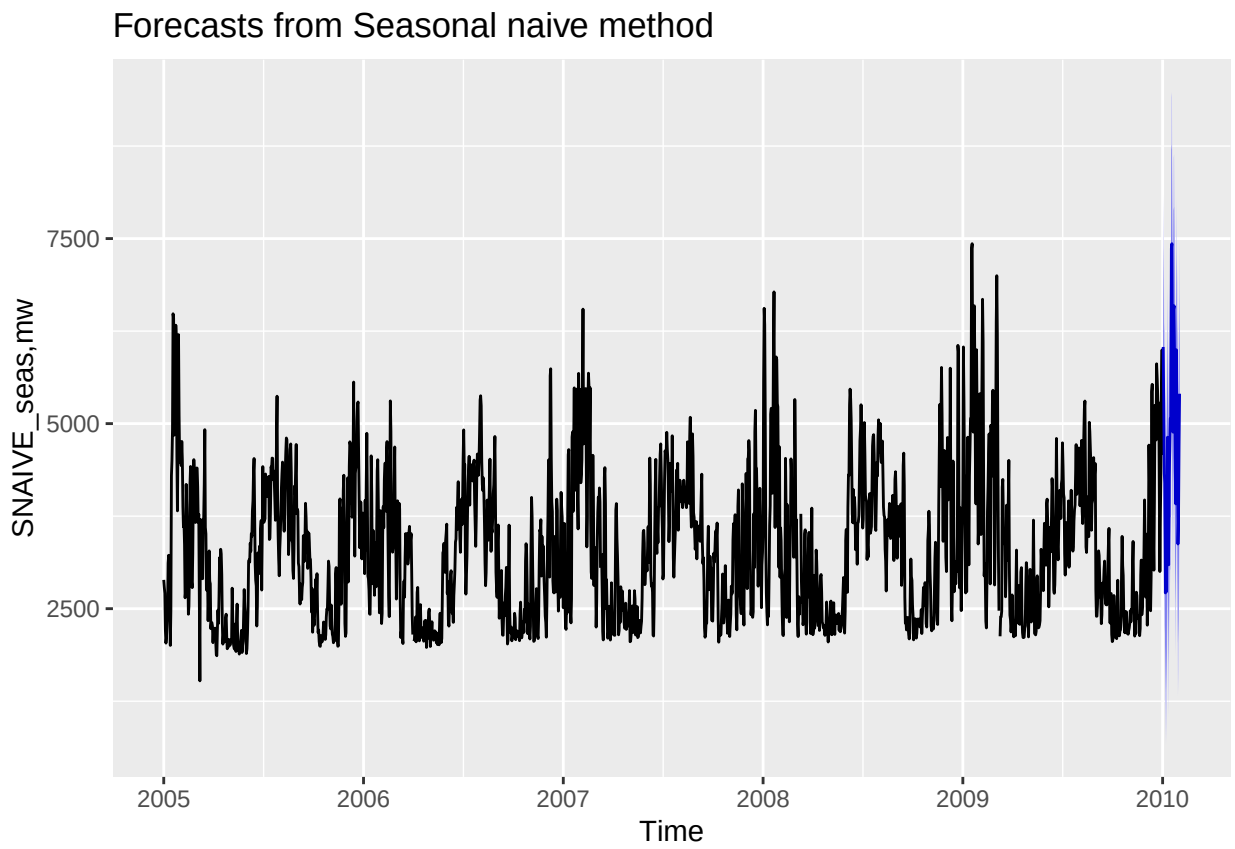
```
#Arithmetic mean on original data
MEAN_seas <- meanf(y = ts_load_09, h = 31)
autoplot(MEAN_seas)+ylab("Mean_seas, mw")
```



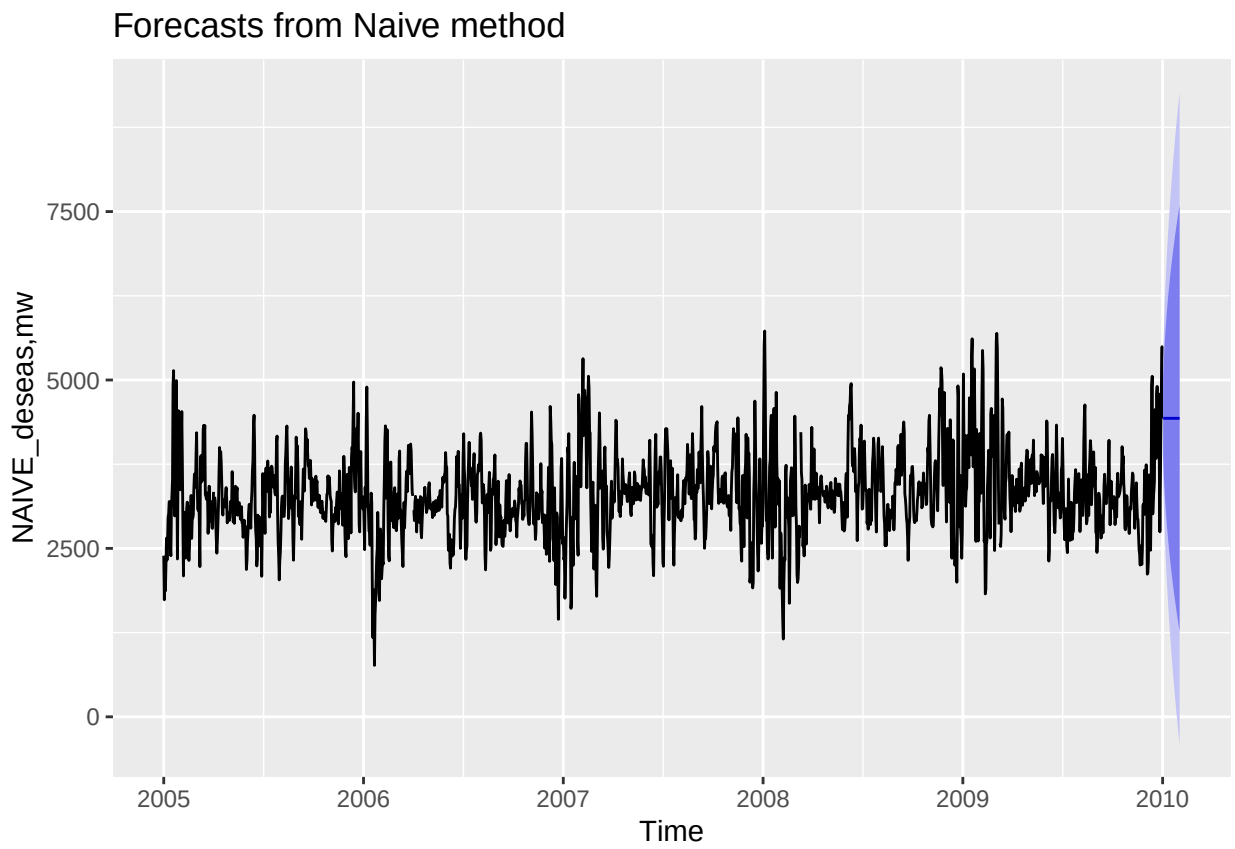

```
#Arithmetic mean on deseas data
MEAN_deseas <- meanf(deseasonal_load_09, h=31)
autoplot(MEAN_deseas)+ylab("MEAN_deseas,mw")
```




```
#Seasonal naive on original data
SNAIVE_seas <- snaive(ts_load_09, h=31)
autoplot(SNAIVE_seas)+ylab("SNAIVE_seas,mw")
```




```
#Naive on deseas data
NAIVE_deseas <- naive(deseasonal_load_09, h=31)
autoplot(NAIVE_deseas)+ylab("NAIVE_deseas,mw")
```



##Manually explore ARIMA At this step, we first ran the ADF test, the p-value is less than 0.01, implying that there is no unit root. Then, we ran the seasonal Mann-Kendall test, 2-sided p-value =8.6608e-12, implying that there is a trend in the series. To manually explore ARIMA model, we first ran the n_diff function, differenced the series accordingly, and created ACF and PACF for the differenced data. When looking at the first 12 lags for ACF and PACF, we see the ACF shows a slow decay and PACF shows a cutoff at lag 4. Indicating p=4,q=0, and we know that d=1. There are no significant lags at 12, 24... in ACF and PACF, thus it has no seasonal components in the seasonal ARIMA model. The model we used is (4,1,0)(0,0,0). The output is a forecast using this model.

```
##adf test
deseasonal_load_09 <- na.fill(deseasonal_load_09,"extend")
adf.test(deseasonal_load_09,alternative="stationary")
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  deseasonal_load_09
## Dickey-Fuller = -11.298, Lag order = 12, p-value = 0.01
## alternative hypothesis: stationary
```

```
##seasonal Mann-Kendall test
SeasonalMannKendall(deseasonal_load_09)
```

```
## tau = 0.146, 2-sided pvalue =8.6608e-12
```

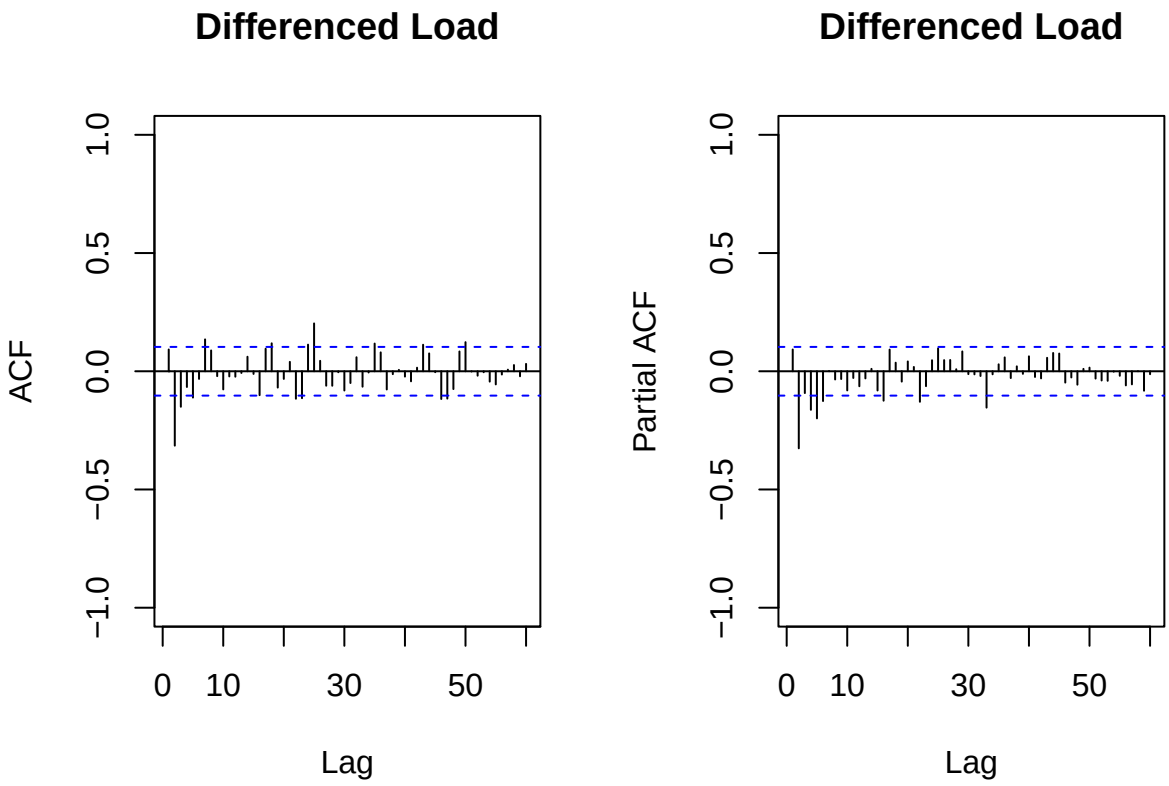
```r
#Use n_diff to find out the number of differencing needed
n_diff_trend <- ndiffs(deseasonal_load_09)
cat("Number of differencing needed: ",n_diff_trend)
```

```
## Number of differencing needed:  1
```

```r
##d=1
ns_diff_seasonal <- nsdiffs(ts_load_09)
cat("Number of seasonal differencing needed: ",ns_diff_seasonal)
```

```
## Number of seasonal differencing needed:  0
```

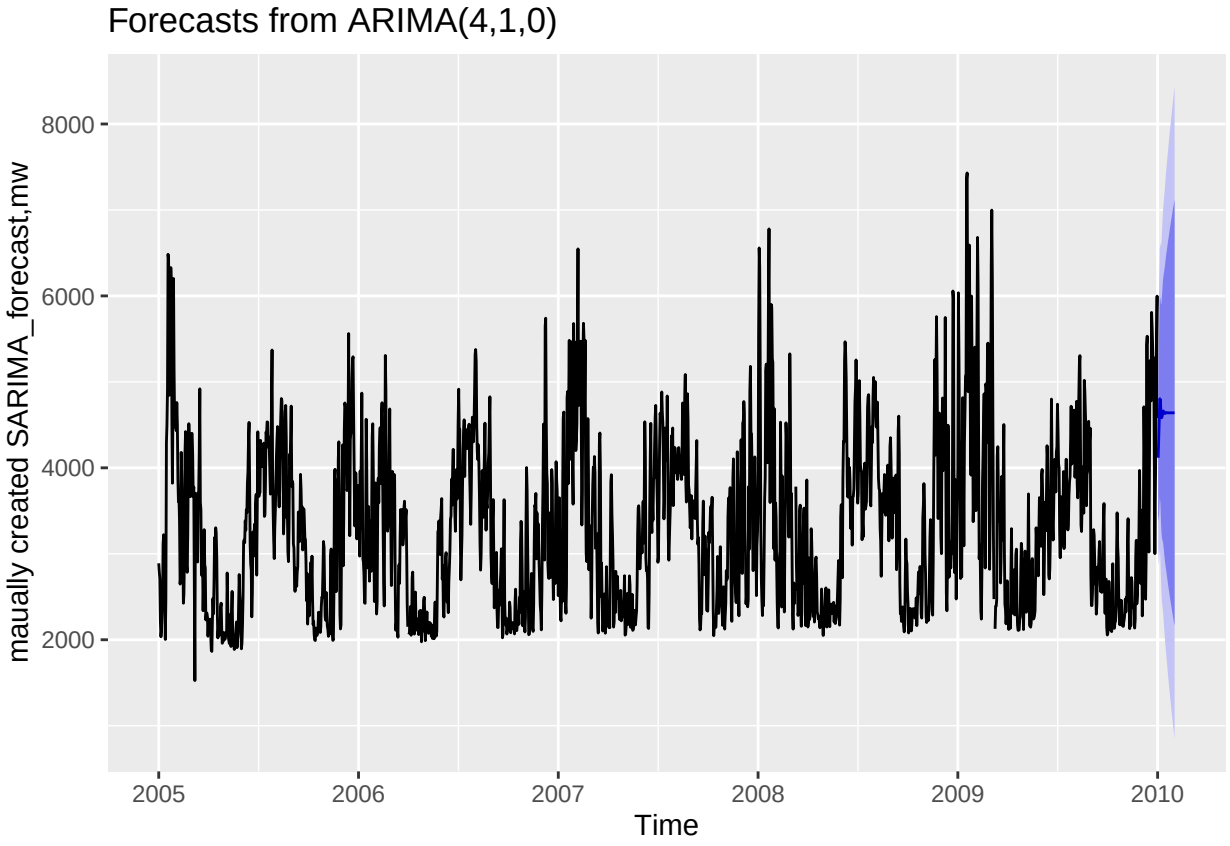
```r
##D=0
ts_load_09_diff <- diff(ts_load_09,lag =1, differences=1)
##Create ACF and PACF
par(mfrow=c(1,2))
Acf(ts_load_09_diff,lag.max=60,main="Differenced Load",ylim=c(-1,1))
Pacf(ts_load_09_diff,lag.max=60,main="Differenced Load",ylim=c(-1,1))
```




```r
##Forecast using manual sarima model
SARIMA_Model <- Arima(ts_load_09,order=c(4,1,0),seasonal=c(0,0,0),include.drift=FALSE)
print(SARIMA_Model)
```

```
## Series: ts_load_09
## ARIMA(4,1,0)
##
## Coefficients:
##          ar1      ar2      ar3      ar4
##       0.0503  -0.3465  -0.1037  -0.1741
## s.e.  0.0232   0.0230   0.0230   0.0231
##
## sigma^2 = 273250:  log likelihood = -13974.3
## AIC=27958.6   AICc=27958.64   BIC=27986.15
```

```
SARIMA_forecast<-forecast::forecast(object=SARIMA_Model,h=31)
autoplot(SARIMA_forecast)+ylab("maually created SARIMA_forecast,mw")
```



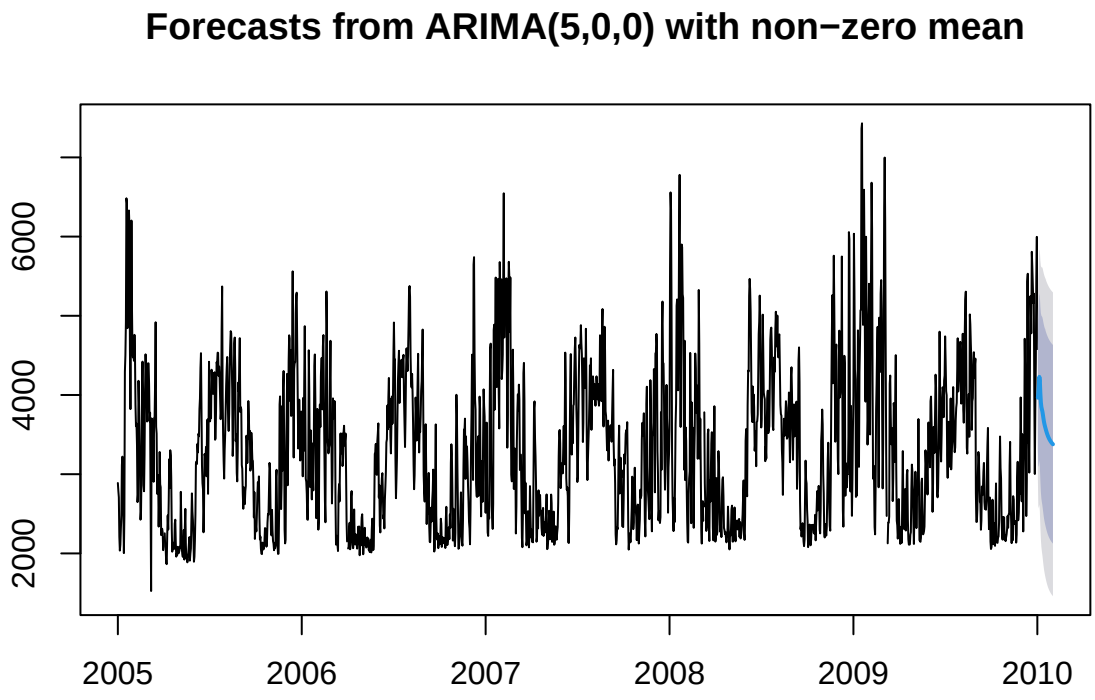
##Auto ARIMA We ran the auto ARIMA function on our time series load data. We ran A Seasonal ARIMA model on the original load data, obtaining a model of ARIMA (5,0,0) and no seasonal component. The AIC score is 27905.49. we also ran an ARIMA model on the deseasonal data, obtaining a model of ARIMA (1,1,1) and an AIC of 27334.23. Surprisingly, the order of the non-seasonal components between the two models is quite different. Because these two models ran on two different series, their AICs are not directly comparable.

```
# Model:  SARIMA on original data
SARIMA_autofit <- auto.arima(ts_load_09)
print(SARIMA_autofit)
```

```
## Series: ts_load_09
```

```
## ARIMA(5,0,0) with non-zero mean
##
## Coefficients:
##          ar1      ar2     ar3      ar4     ar5       mean
##       1.0045  -0.3993  0.2156  -0.0727  0.1272  3306.5091
## s.e.  0.0233   0.0331  0.0340   0.0331  0.0233    95.5246
##
## sigma^2 = 262752:  log likelihood = -13945.75
## AIC=27905.49   AICc=27905.55   BIC=27944.06
```

```
SARIMA_forecast <- forecast::forecast(object = SARIMA_autofit, h = 31)
plot(SARIMA_forecast)
```

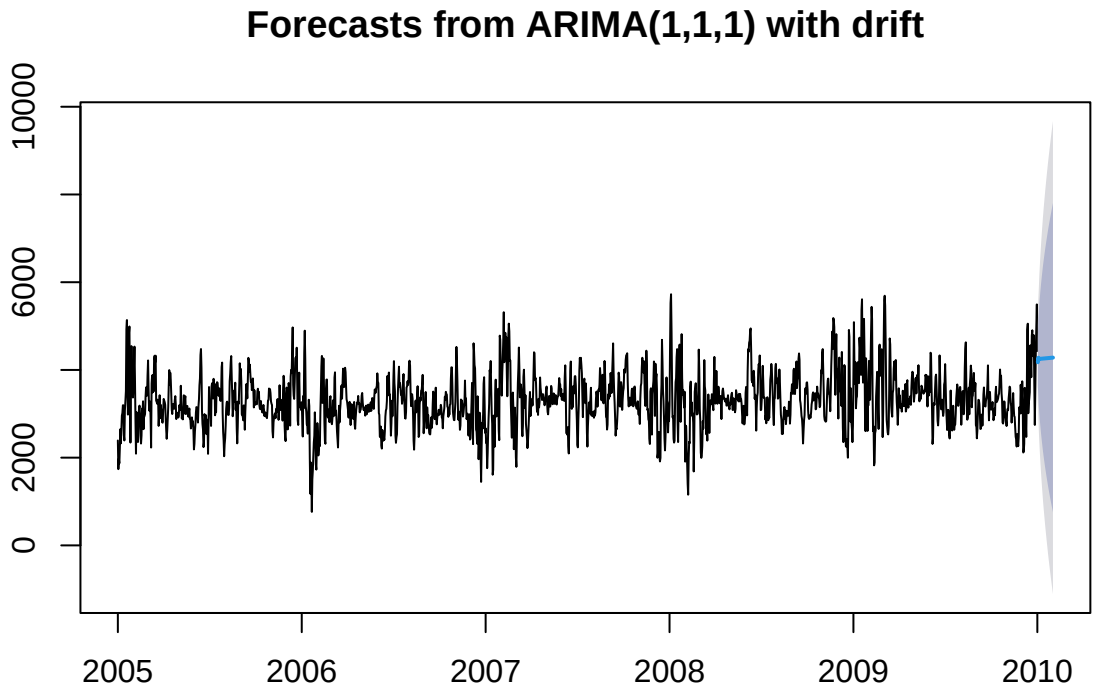



```
# Model:  ARIMA on deseasonal data
ARIMA_autofit <- auto.arima(deseasonal_load_09, max.D = 0, max.P = 0, max.Q = 0)
print(ARIMA_autofit)
```

```
## Series: deseasonal_load_09
## ARIMA(1,1,1) with drift
##
## Coefficients:
##          ar1     ma1    drift
##       -0.3881  0.5913   1.0916
## s.e.   0.0541  0.0449  11.5834
##
```

```
## sigma^2 = 186653:  log likelihood = -13663.11
## AIC=27334.23   AICc=27334.25   BIC=27356.27
```

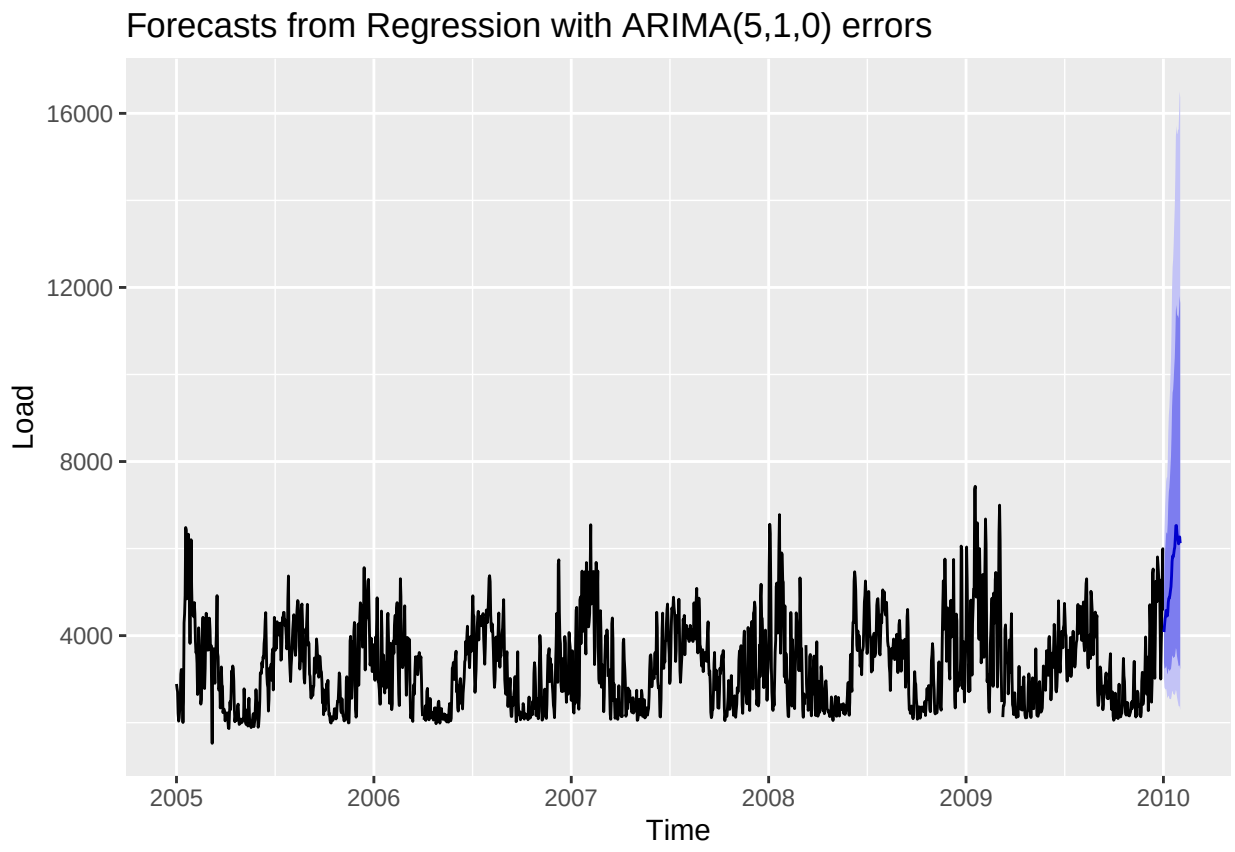
```
ARIMA_forecast <- forecast::forecast(object = ARIMA_autofit, h = 31)
plot(ARIMA_forecast)
```



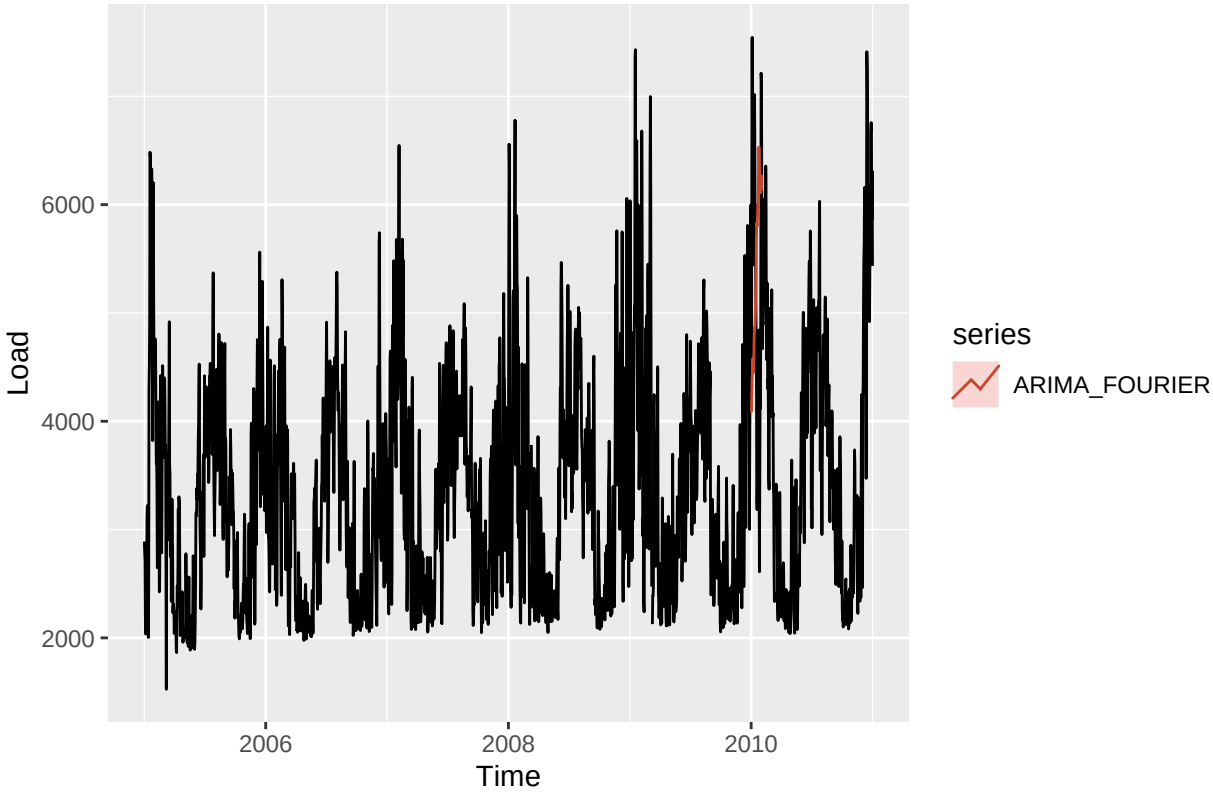
##ARIMA + FOURIER terms In this section, we ran an ARIMA model with fourier terms and an ARIMA model with fourier terms and the exogenous regressors. We first created a matrix with the fourier terms and a matrix with fouier terms and the exogenous variables. The training period is used to fit the ARIMA model and the testing period is used to forecast the ARIMA model. Lastly, we plotted the forecast results and compared them with the observed data.

```
##Create a matrix of just the fourier terms and a matrix with the fourier terms and the exogenous regre
fourier_mat<-fourier(ts_load_09,K=c(2,12),h=nrow(ts_load_09))
fourier_mat_for<-fourier(ts_load_09,K=c(2,12),h=31)
xregs_fourier<-cbind(fourier_mat,xregs_train)
xregs_for<-xregs_daily[(nrow(xregs_fourier)+1):(nrow(xregs_fourier)+31),]
xregs_fourier_for<-cbind(fourier_mat_for,xregs_for)
##without exogenous regressors
##Fit ARIMA with fourier matrix
ARIMA_Four_fit <- auto.arima(ts_load_09,
                             seasonal=FALSE,
                             lambda=0,
                             xreg=fourier_mat
                             )
#Forecast with ARIMA fit
```

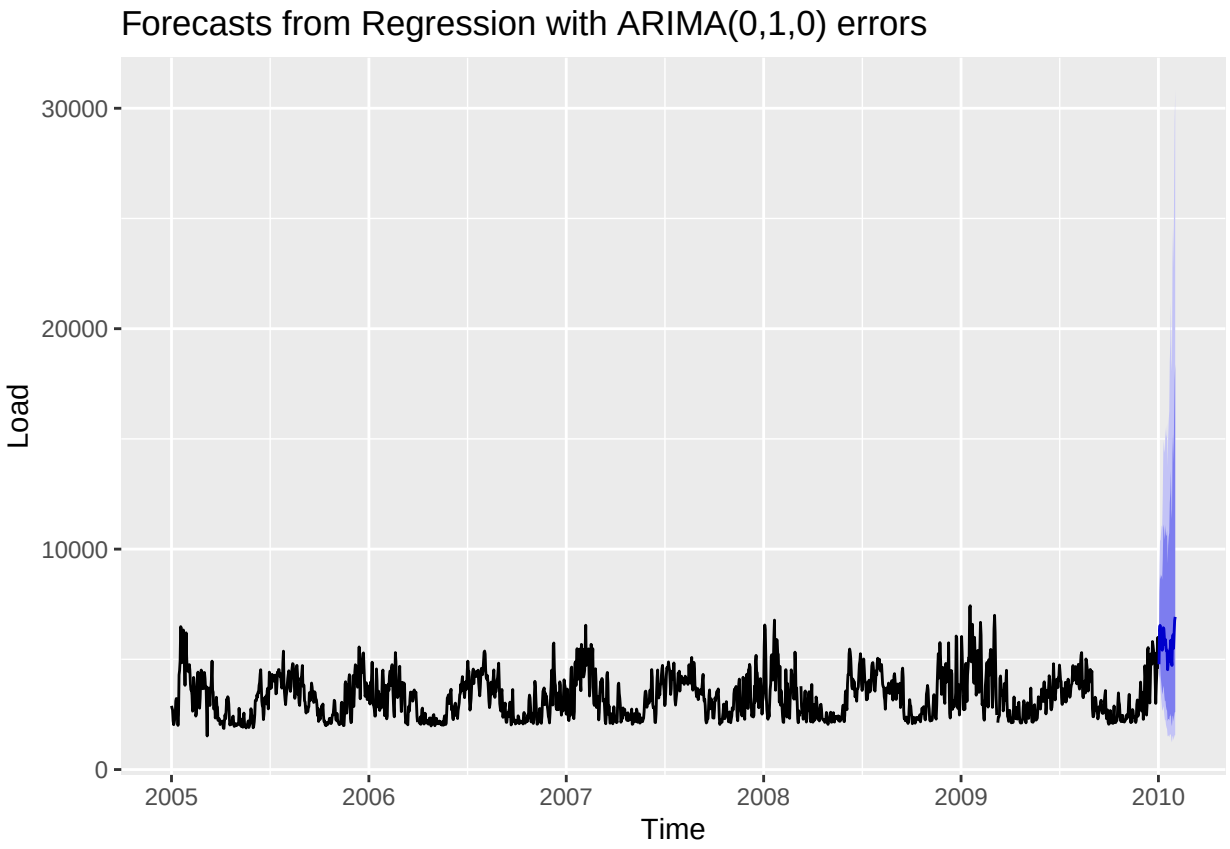
```
ARIMA_Four_for <- forecast::forecast(ARIMA_Four_fit,
                           xreg=fourier_mat_for,
                           h=31
                           )
#Plot foresting results
autoplot(ARIMA_Four_for) + ylab("Load")
```



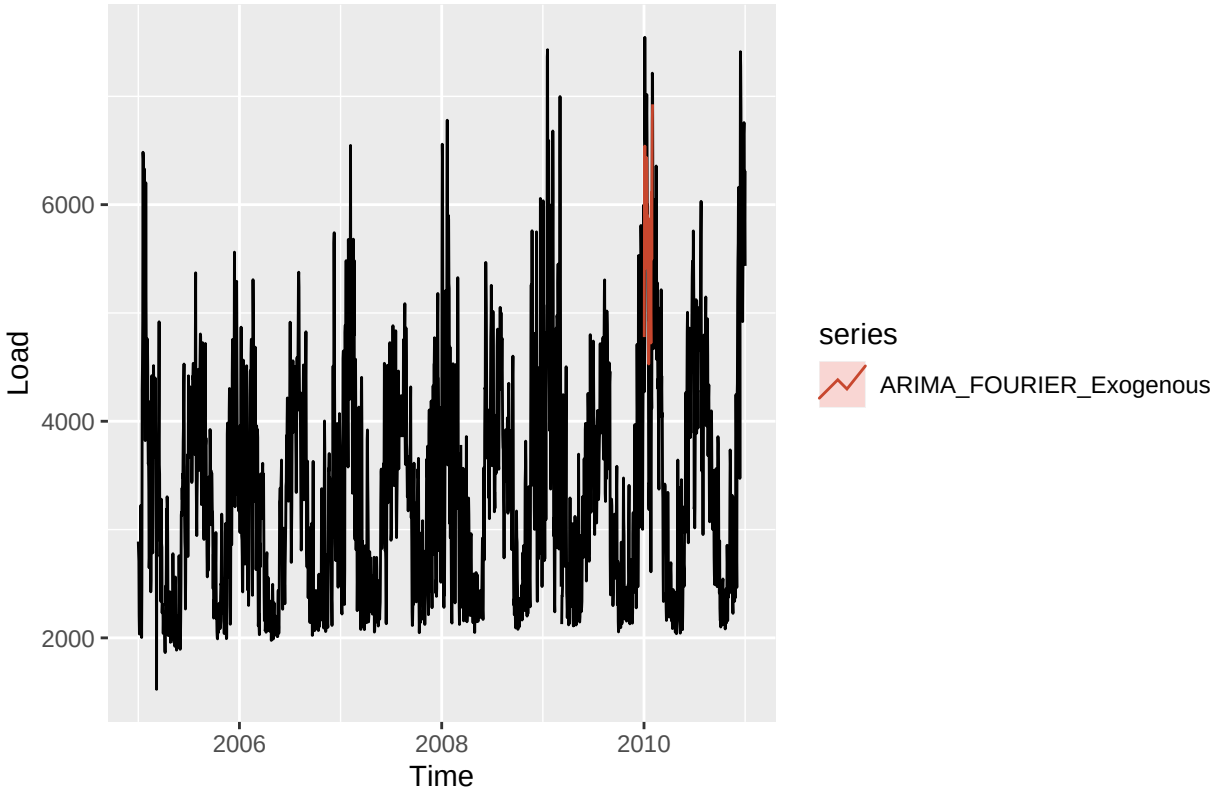

```
#Plot model + observed data
autoplot(ts_load) +
  autolayer(ARIMA_Four_for, series="ARIMA_FOURIER",PI=FALSE) +
  ylab("Load")
```




```
##with exogenous regressors
ARIMA_Four_fit2 <- auto.arima(ts_load_09,
                              seasonal=FALSE,
                              lambda=0,
                              xreg=xregs_fourier
                              )
#Forecast with ARIMA fit
ARIMA_Four_for2 <- forecast::forecast(ARIMA_Four_fit2,
                                      xreg=xregs_fourier_for,
                             h=31
                             )
#Plot foresting results
autoplot(ARIMA_Four_for2) + ylab("Load")
```

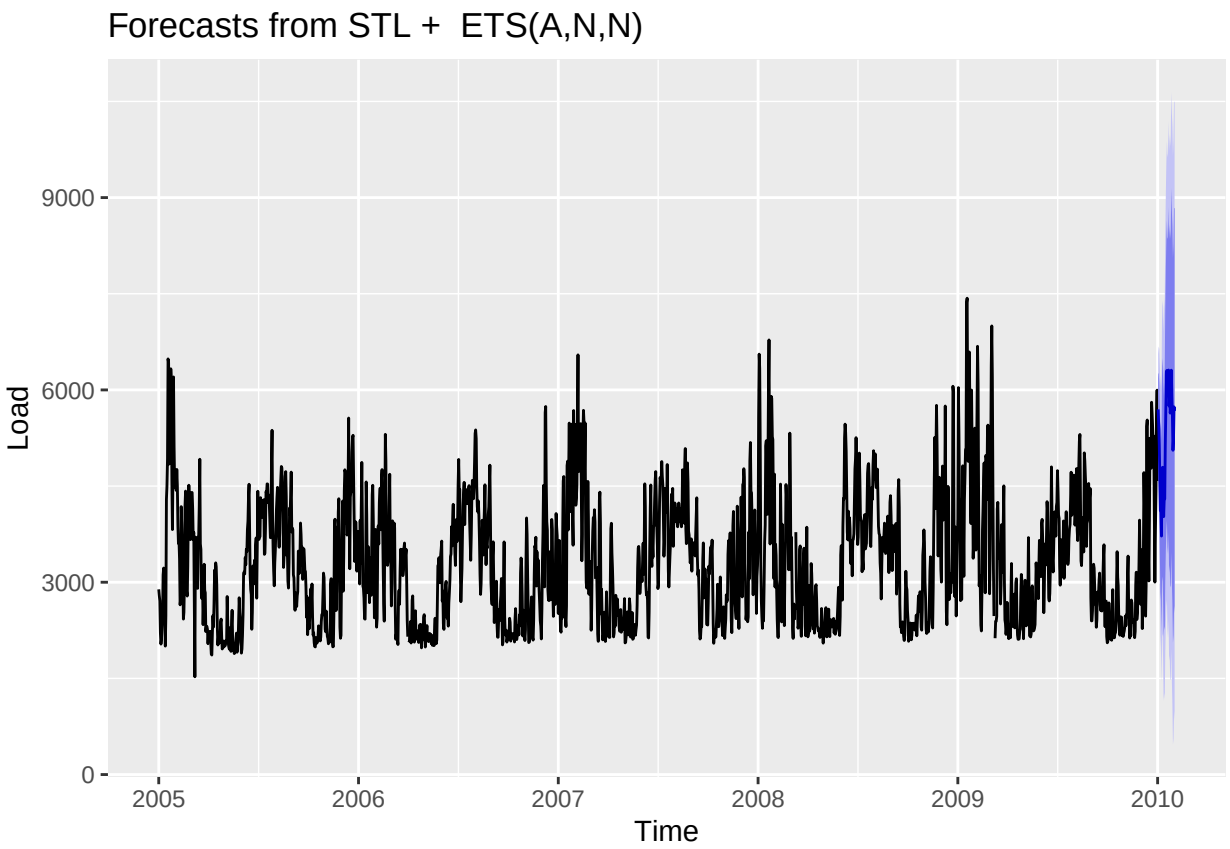



```
#Plot model + observed data
autoplot(ts_load) +
  autolayer(ARIMA_Four_for2, series="ARIMA_FOURIER_Exogenous",PI=FALSE) +
  ylab("Load")
```

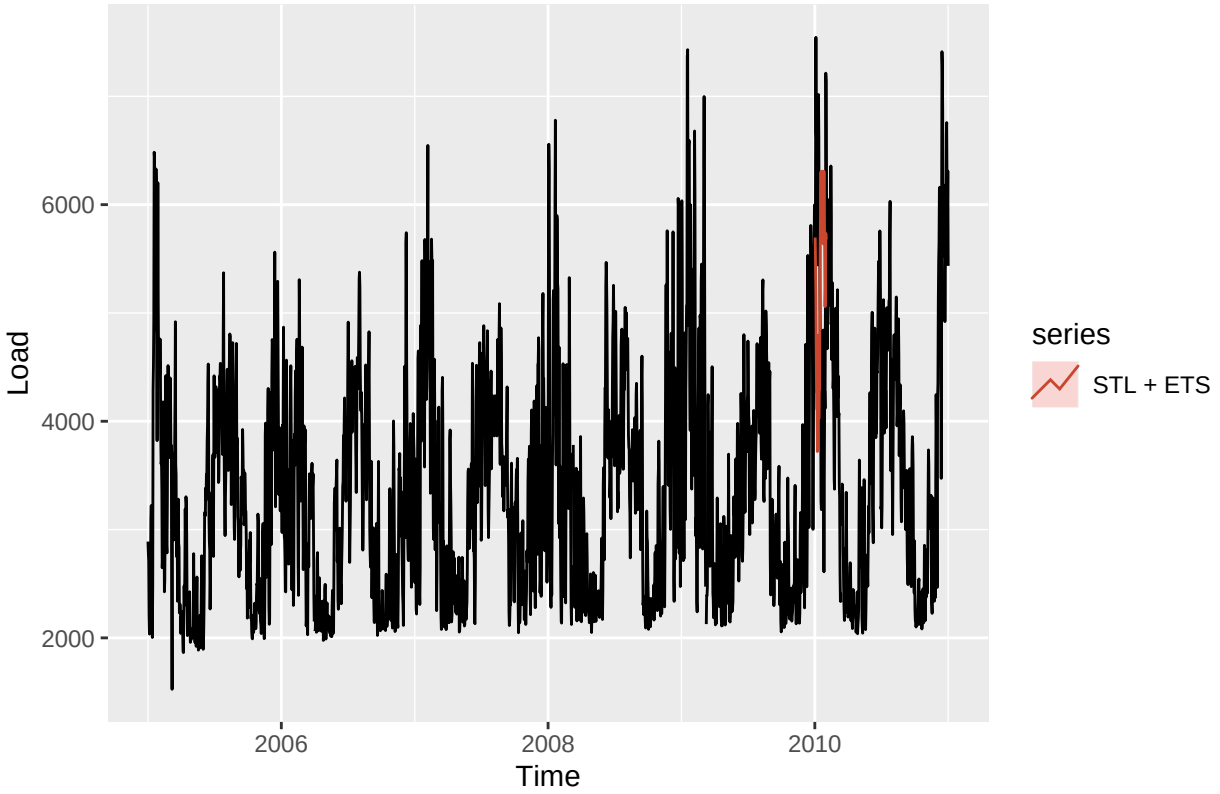


##ETS Model In this section, we fit and forecast the load data using the STL+ETS model, plotted the forecast results, and compared it with the observed data

```
#Fit and forecast STL + ETS model to data
ETS_fit <-  stlf(ts_load_09,h=31)
#Plot foresting results
autoplot(ETS_fit) + ylab("Load")
```




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

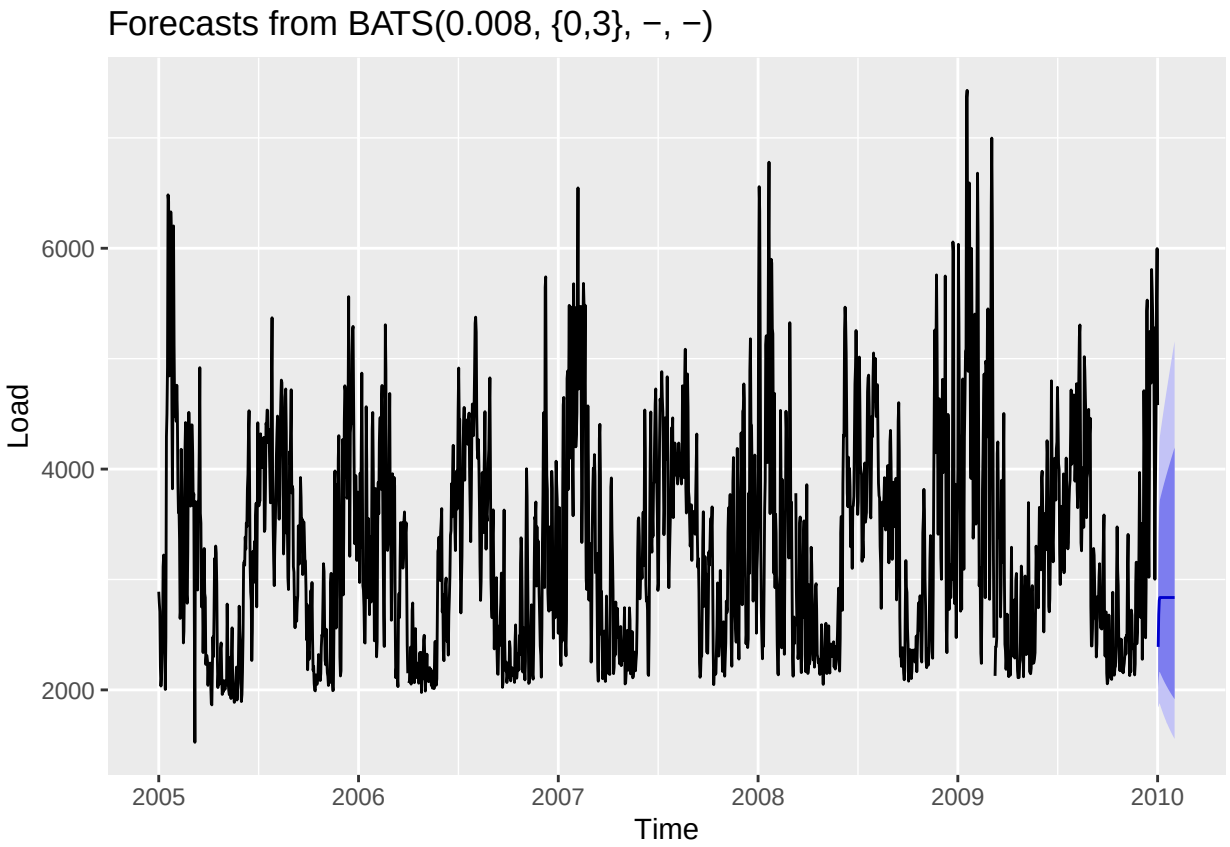


##TBATS Model In this section, we fit and forecast the load data using the TBATS model, plotted the forecast results, and compared it with the observed data

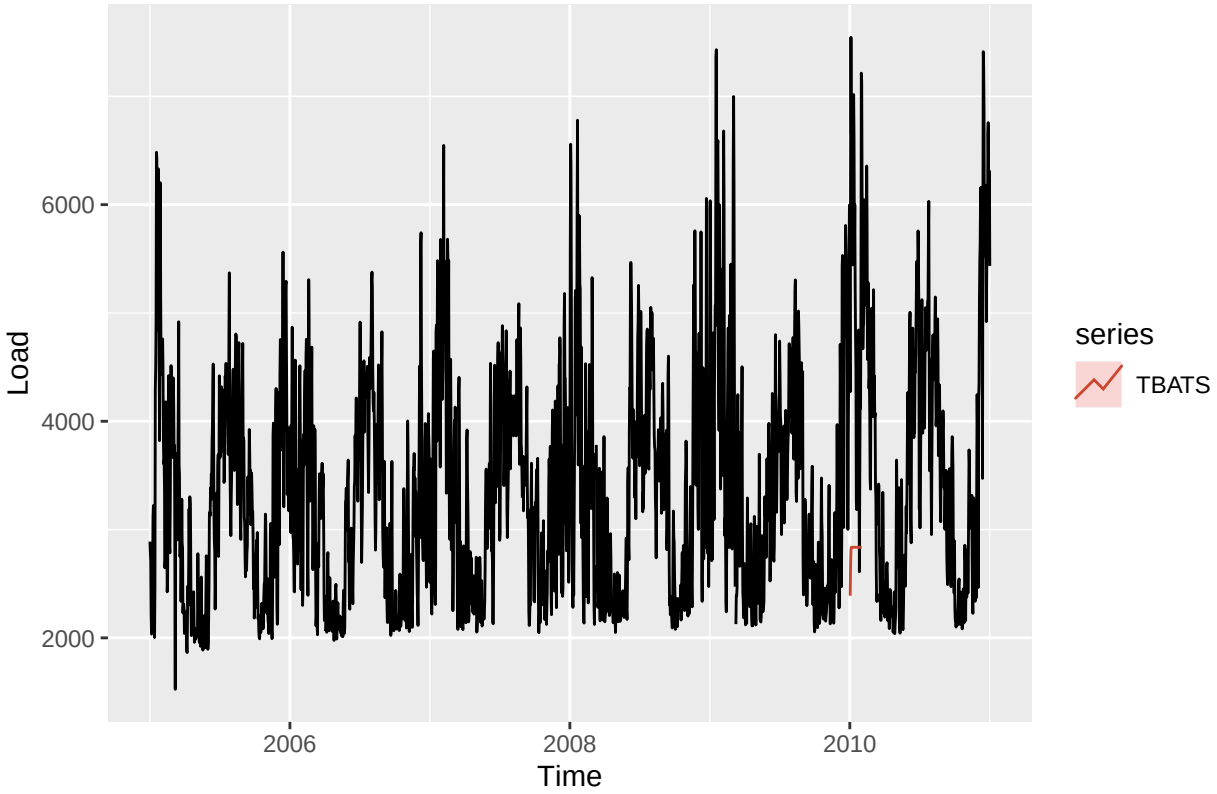
```
# TBATS can take time to fit
TBATS_fit <- tbats(ts_load_09)
```

```
## Warning in tbats(ts_load_09): Missing values encountered. Using longest
## contiguous portion of time series
```

```
TBATS_for <- forecast::forecast(TBATS_fit, h=31)
#Plot foresting results
autoplot(TBATS_for) +
  ylab("Load")
```

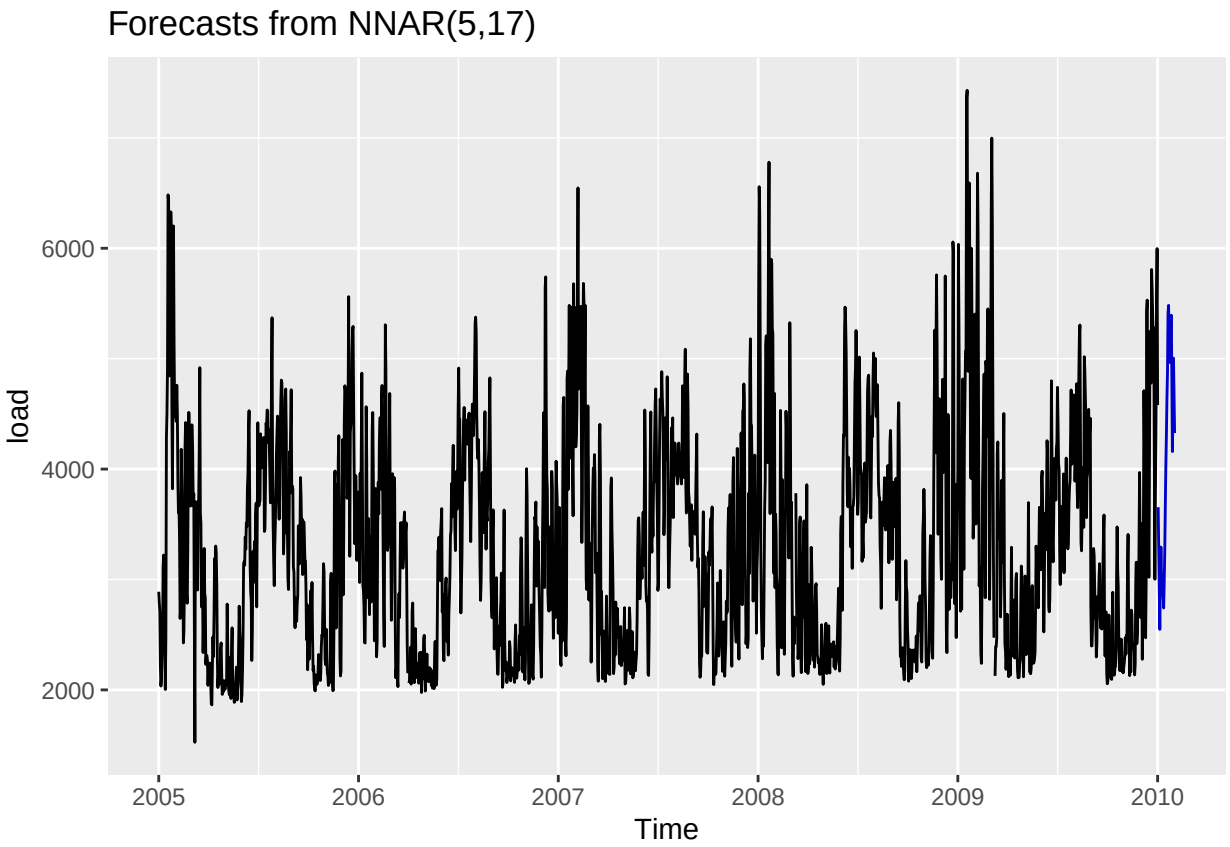



```
#Plot model + observed data
autoplot(ts_load) +
  autolayer(TBATS_for, series="TBATS",PI=FALSE)+
  ylab("Load")
```

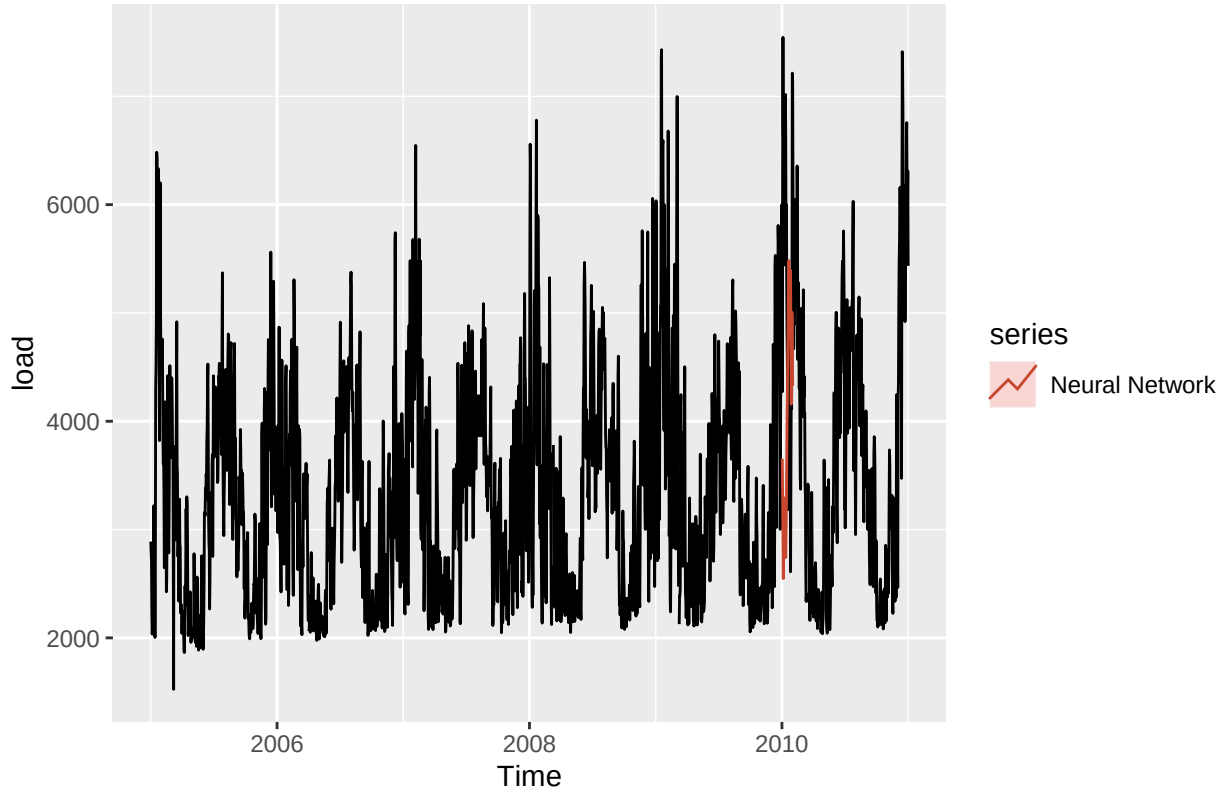


##Neural Network Similar analysis was conducted using the Neural Network model. We used P=5 from the auto ARIMA model. We also ran a model of a Neural Network with exogenous regressors and fourier terms.

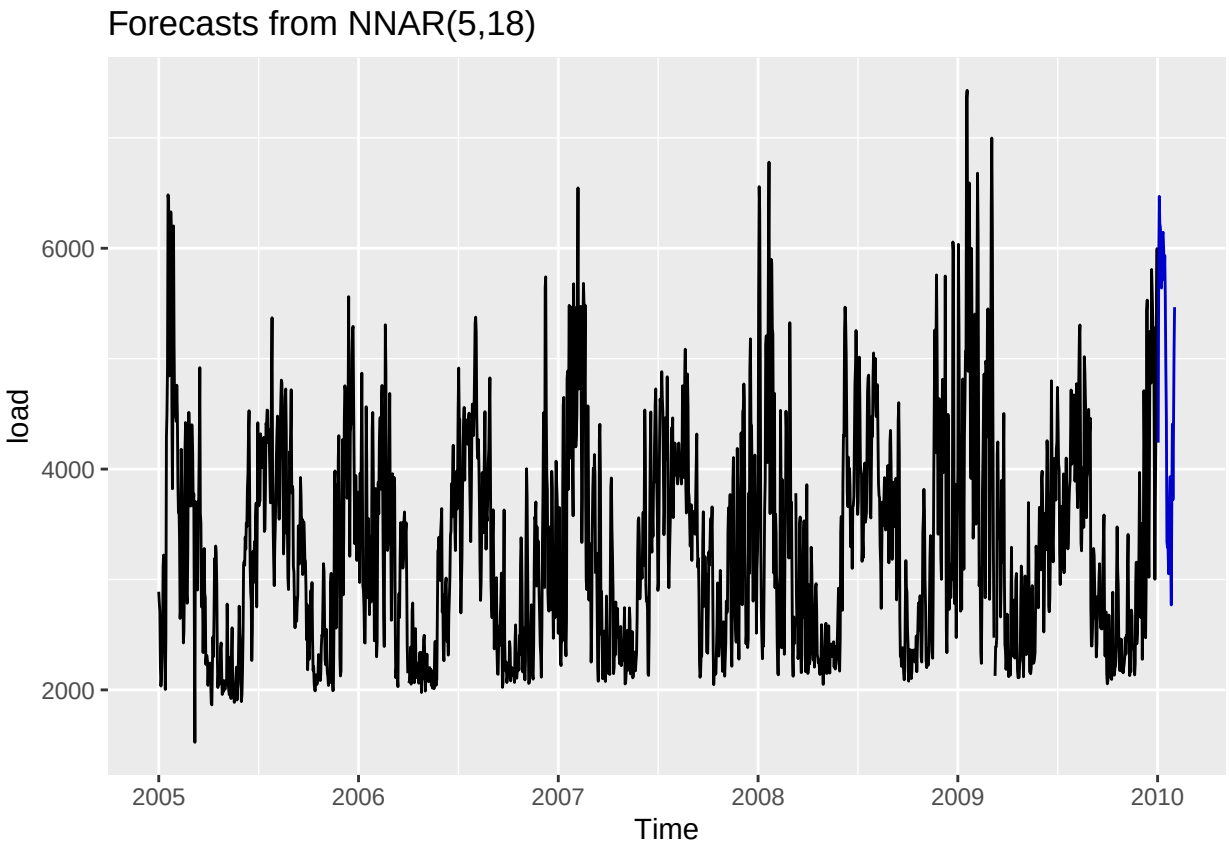
```
#NN_fit
NN_fit <- nnetar(ts_load_09,p=5,P=0,xreg=fourier(ts_load_09, K=c(2,12)))
#NN_for
NN_for <- forecast::forecast(NN_fit, h=31,xreg=fourier(ts_load_09, K=c(2,12),h=31))
#Plot foresting results
autoplot(NN_for) +
  ylab("load")
```



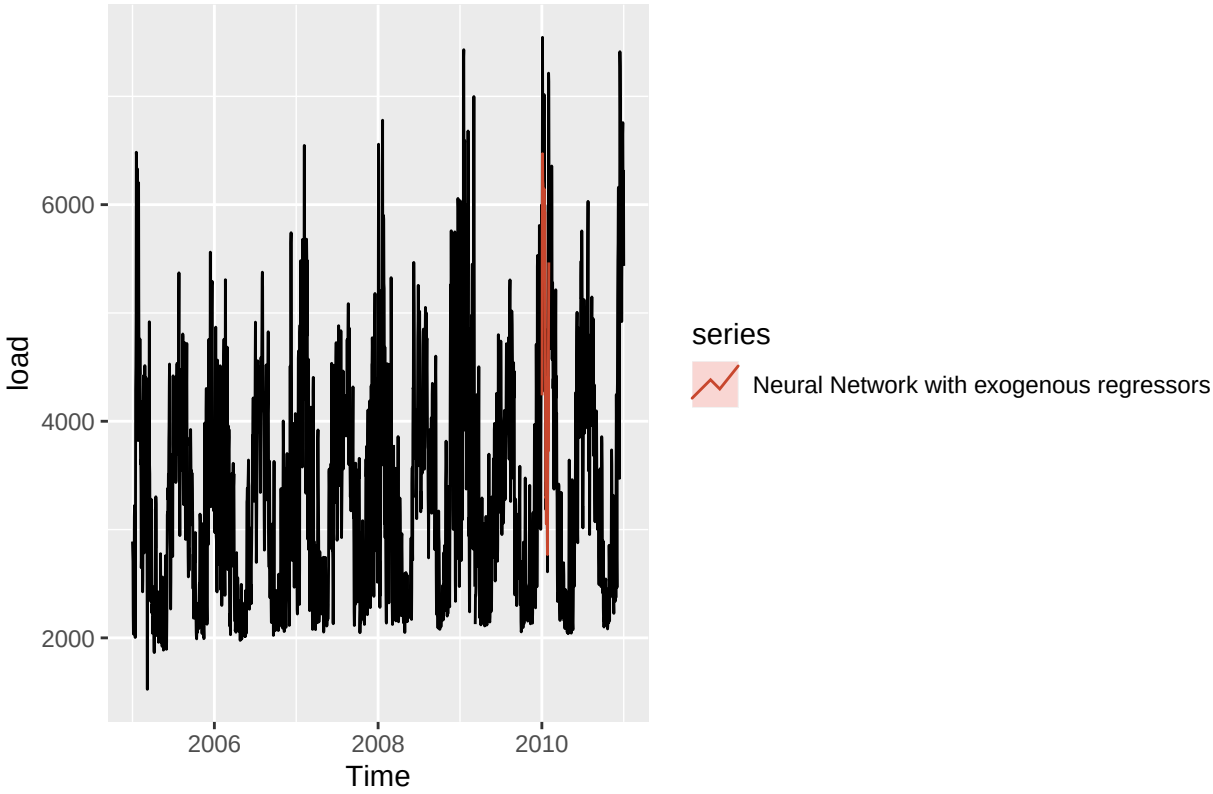

```
#Plot model + observed data
autoplot(ts_load) +
  autolayer(NN_for, series="Neural Network",PI=FALSE)+
  ylab("load")
```




```
##with exogenous regressors and fourier
#NN_fit
NN_fit2 <- nnetar(ts_load_09,p=5,P=0,xreg=xregs_fourier)
#NN_for
NN_for2 <- forecast::forecast(NN_fit2, h=31,xreg=xregs_fourier_for)
#Plot foresting results
autoplot(NN_for2) +
  ylab("load")
```




```
#Plot model + observed data
autoplot(ts_load) +
  autolayer(NN_for2, series="Neural Network with exogenous regressors",PI=FALSE)+
  ylab("load")
```



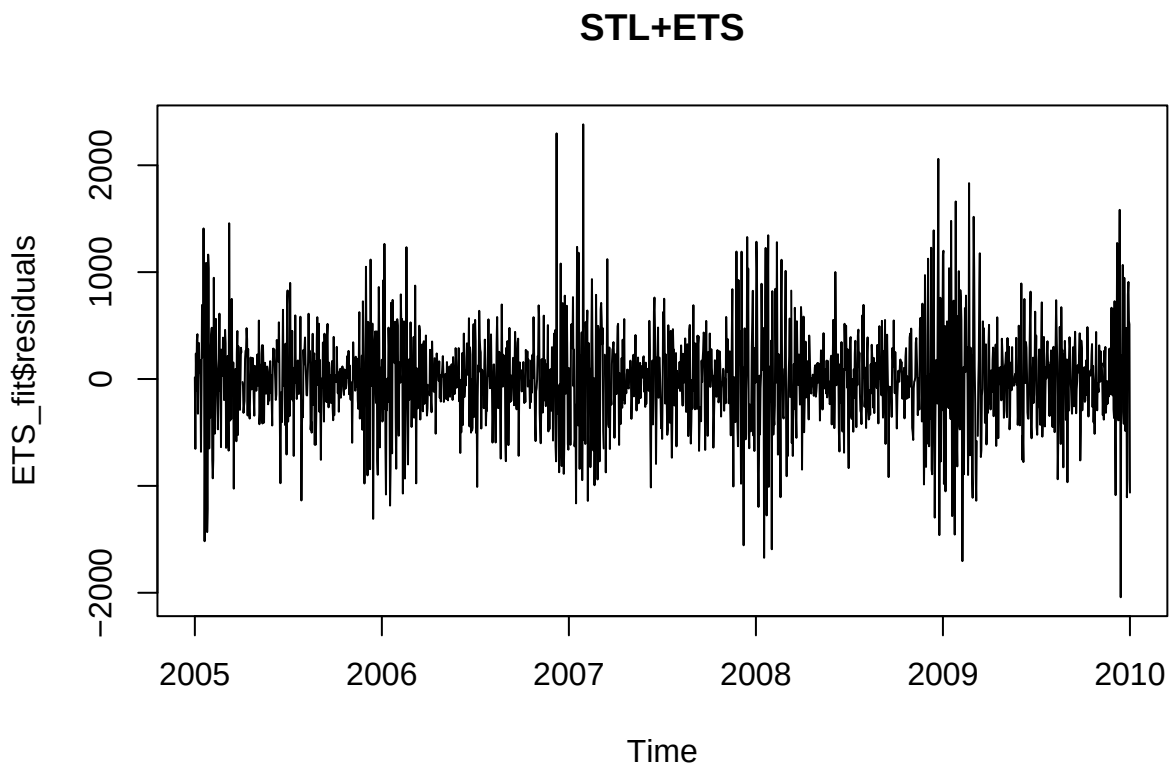
##Check Accuracy In this section, we checked the accuracy scores for our models to compare the performance of our models. We ran 6 models in total.

```
#Model 1: STL + ETS
ETS_scores <- accuracy(ETS_fit$mean,ts_load_test)
#Model 2: ARIMA + Fourier
ARIMA_scores <- accuracy(ARIMA_Four_for$mean,ts_load_test)
ARIMA_scores2<- accuracy(ARIMA_Four_for2$mean,ts_load_test)
# Model 3:  TBATS
TBATS_scores <- accuracy(TBATS_for$mean,ts_load_test)
# Model 3:  Neural Network
NN_scores <- accuracy(NN_for$mean,ts_load_test)
NN_scores2 <- accuracy(NN_for2$mean,ts_load_test)
```

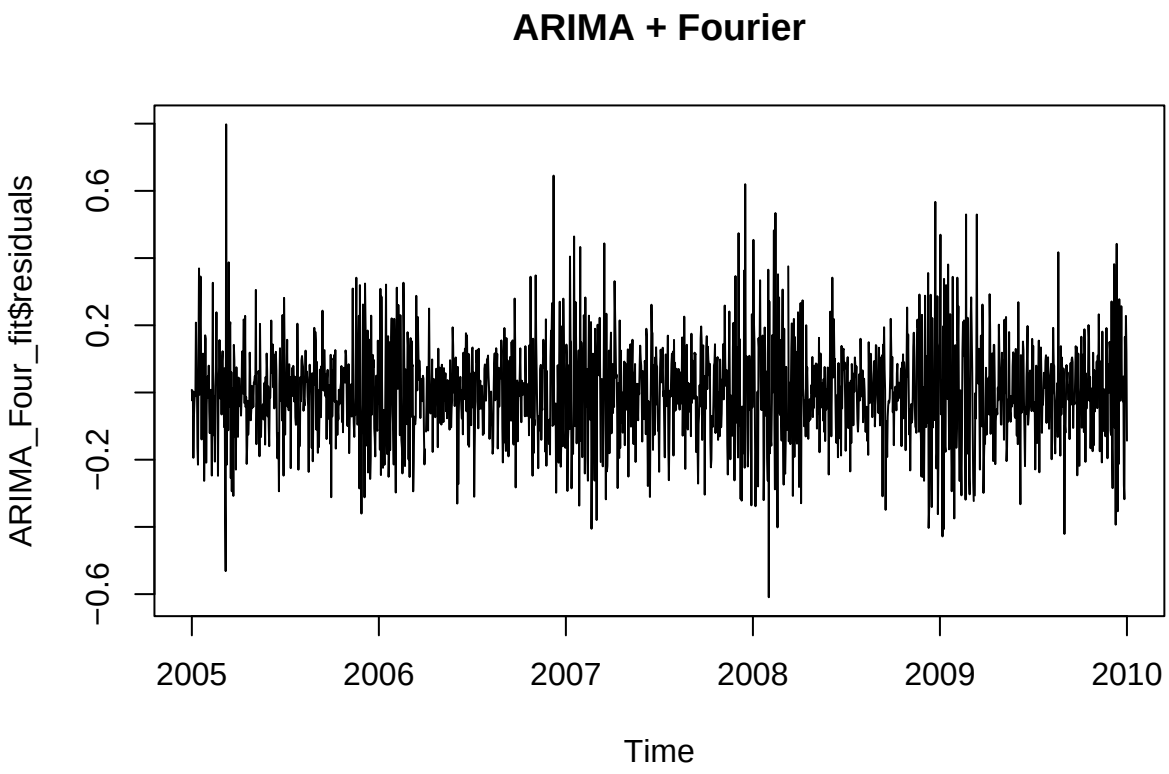
Check Residuals

We then plotted the residuals of our models to check for any sort of bias. A model that accurately captures trend and seasonality should look like uniform white noise. We see evidence of seasonality in STL+ETS, ARIMA+Fourier, and Neural Networks as error bars in these residual plots increase during winter months and decrease during summer months. The addition of Xregs in the latter two models seems to eliminate the seasonality and leaves residuals that are closer to true white noise.

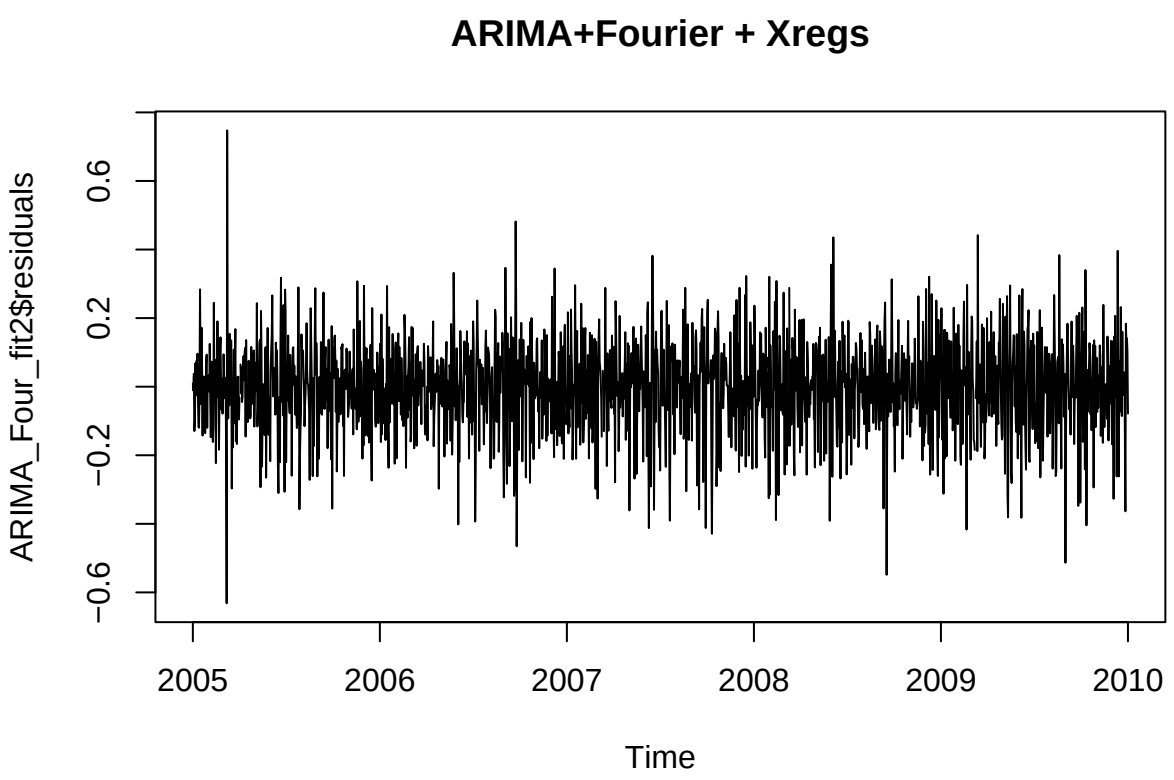
```
ts.plot(ETS_fit$residuals, main="STL+ETS")
```



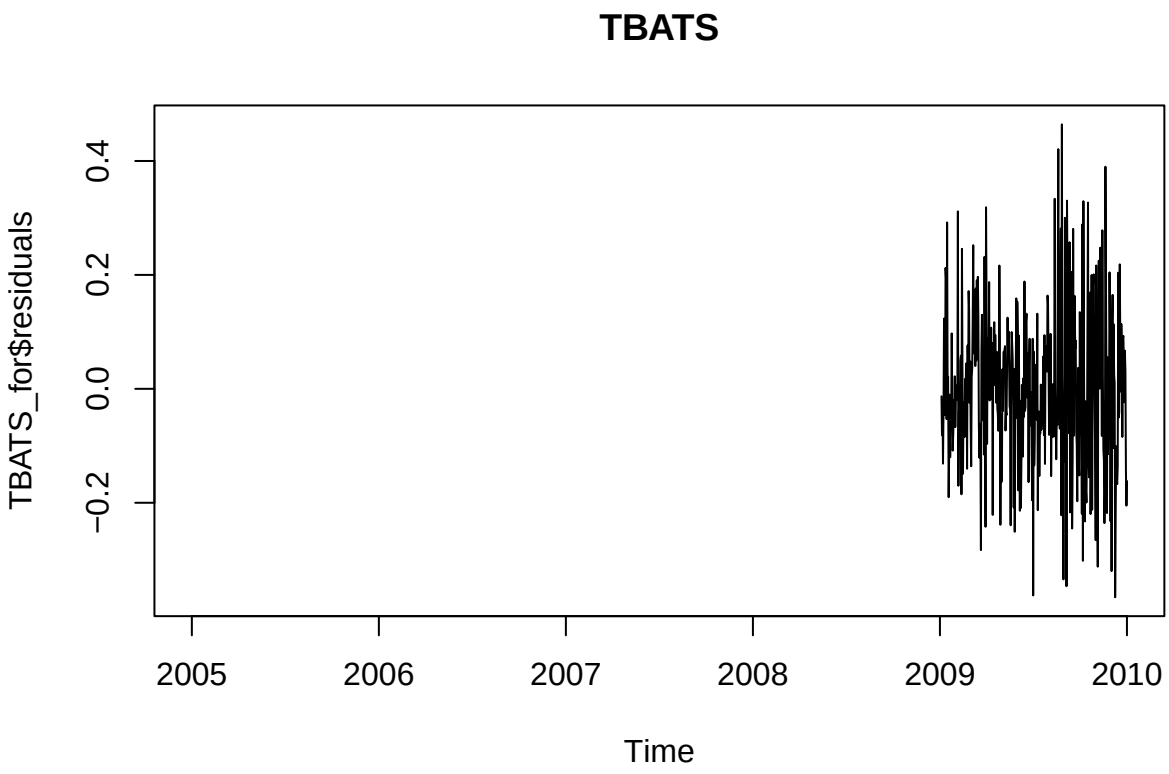

```
ts.plot(ARIMA_Four_fit$residuals, main="ARIMA + Fourier")
```



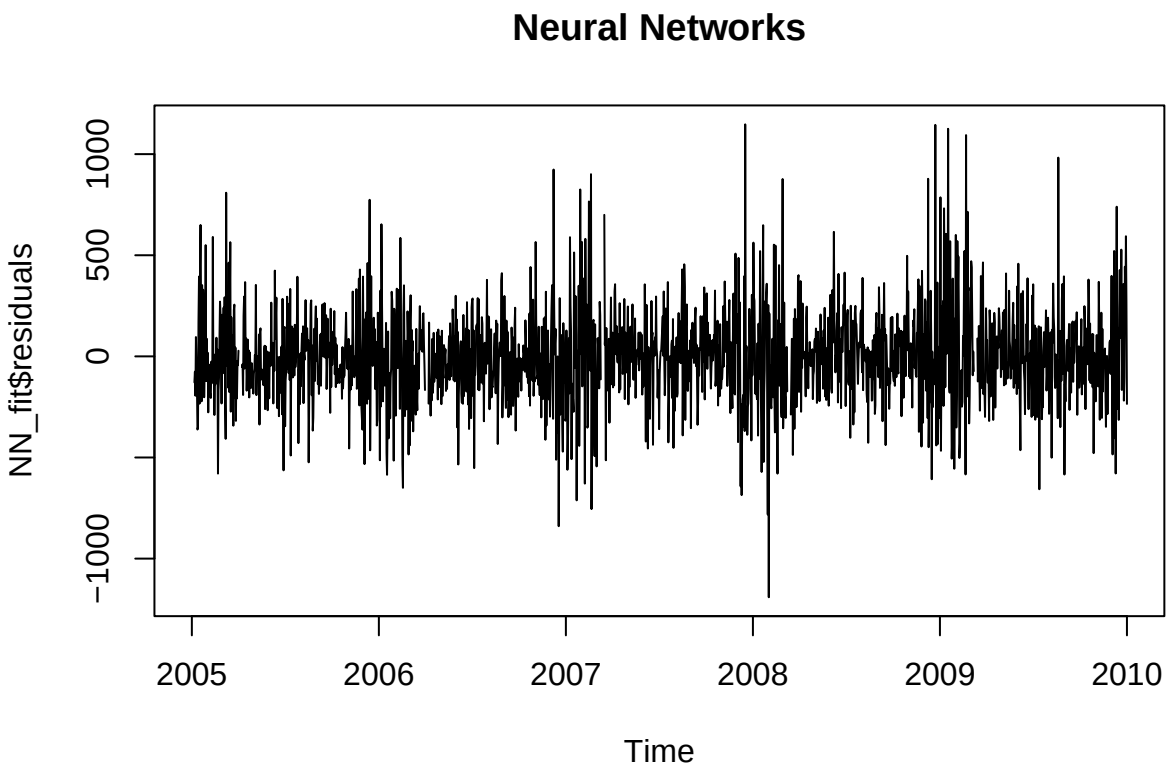

```
ts.plot(ARIMA_Four_fit2$residuals, main="ARIMA+Fourier + Xregs")
```



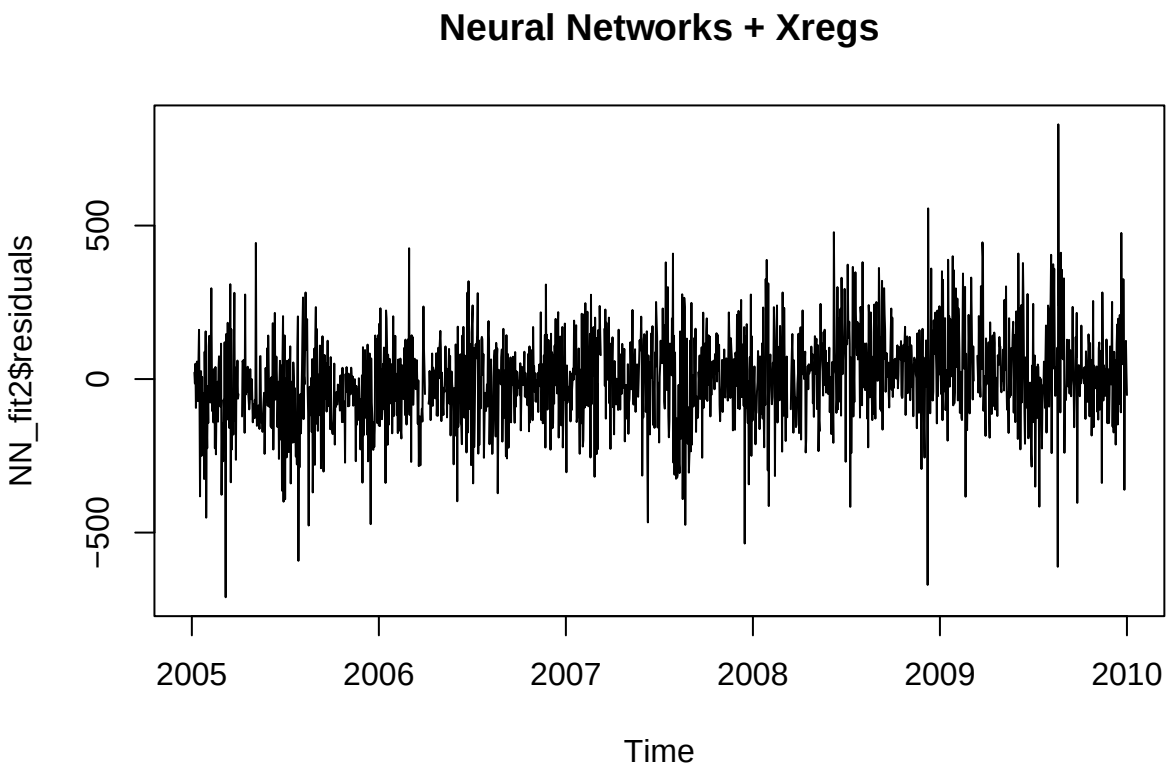

```
ts.plot(TBATS_for$residuals, main="TBATS")
```




```
ts.plot(NN_fit$residuals, main="Neural Networks")
```




```
ts.plot(NN_fit2$residuals, main="Neural Networks + Xregs")
```



##Comparing Performance Metrics

We compared performance metrics by creating a dataframe to hold all performance metrics by model type. We also returned the model type with the best RMSE score which was the neural network with exogenous regressors. This was unsurprising as we saw that it produced a white noise residuals plot.

```
#create a dataframe of accuracy scores
scores <- as.data.frame(
  rbind(ETS_scores, ARIMA_scores,ARIMA_scores2, TBATS_scores, NN_scores, NN_scores2)
  )
row.names(scores) <- c("STL+ETS", "ARIMA+Fourier", "ARIMA+Fourier+XRegs" ,"TBATS","NN", "NN+Xregs")

#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+Xregs
```

```
print(scores)
```

```
##                            ME      RMSE       MAE        MPE      MAPE
## STL+ETS              -133.5084 1982.2491 1800.8504 -14.714124 40.300320
## ARIMA+Fourier        -290.5115 1981.8450 1782.7333 -17.632188 40.409491
## ARIMA+Fourier+XRegs  -551.4196 1091.9865  904.6900 -17.042174 22.267645
## TBATS                2320.3354 2710.5455 2334.9495  40.469374 41.029273
## NN                   1099.9949 2467.0747 2081.1433  10.069110 39.878996
```

```
## NN+Xregs              349.7656  661.6772  462.0104   5.609243  8.298668
##                           ACF1 Theil's U
## STL+ETS              0.8423850 2.3892629
## ARIMA+Fourier        0.8958580 2.4962501
## ARIMA+Fourier+XRegs 0.8385537 1.5674183
## TBATS                0.7805849 2.4426533
## NN                   0.8707546 2.3532796
## NN+Xregs             0.5251329 0.6569259
```

##Finding Best Weather Stations

Upon closer investigation of the accuracy scores, we noticed that adding the exogenous variables improved the accuracy scores of both our ARIMA and Neural Networks models. Our approach to utilizing exogenous variables by taking averages across all weather stations was fairly rudimentary. Our next step was to try to improve upon our models by taking a more descriminant approach to including exogenous variables.

In this chunk, we re-clean humidity and temperature data by creating data frames holding averaging daily values for each weather station rather than averaging the averages and getting only one variable for humidity and one for temperature. Then we convert the dataframes to msts and cbind with our load data to create a larger time series.

Then we find the correlation between each variable (load and humidity and temperature for each of our weather stations) and print the names of the weather stations that have the strongest correlations with load. These are the weather stations that we will use to improve upon our previous models.

The top 5 most correlated weather stations were all collecting temperature, not humidity. This is unsurprising as temperature, not humidity, is a more logical driver of electrical load than humidity because people's behavior responds more to temperature.

```
#Create new humidity dataframe with daily averages for every weather station
humidity_3<- humidity %>%
  mutate( Date = ymd(date)) %>%
  mutate( Year = year(date),
          Month = month(date),
          Day = day(date)) %>%
  select( Date, Year, Month, Day, 3:33)
humidity_daily_2 <- humidity_3 %>%
  group_by(Date,Year,Month,Day) %>%
  summarise_all(mean)
humidity_daily_2<- humidity_daily_2[,(5:33)]

#Create new temp dataframe with daily averages for every weather station
temp_3<- temp %>%
  mutate( Date = ymd(date)) %>%
  mutate( Year = year(date),
          Month = month(date),
          Day = day(date)) %>%
  select( Date, Year, Month, Day, 3:33)
temp_daily_2 <- temp_3 %>%
  group_by(Date,Year,Month,Day) %>%
  summarise_all(mean)
temp_daily_2<- temp_daily_2[,(5:33)]

#Make new xreg dataframes into msts and add to load msts
ts_full_humidity <- msts(humidity_daily_2,
                         seasonal.periods =c(7,365.25),
```

```
                            start=c(2005,1,1))
ts_full_temp <- msts(temp_daily_2,
                            seasonal.periods =c(7,365.25),
                            start=c(2005,1,1))
full_data <- cbind(ts_load,ts_full_humidity,ts_full_temp)

# Calculate the correlation between load and each weather station
corrmat <- cor(full_data[,1],full_data[,-1], use = "complete.obs")
corrmat_abs <- abs(corrmat)
corrmat_sorted <- corrmat_abs[,order(corrmat_abs[1,])]

#Sort matrix and print the weather stations with highest absolute value correlation coefficients
top_5_corr <- row.names(as.data.frame(tail(corrmat_sorted,n=5)))
print(top_5_corr)
```

```
## [1] "ts_full_temp.t_ws25" "ts_full_temp.t_ws5"  "ts_full_temp.t_ws18"
## [4] "ts_full_temp.t_ws7"  "ts_full_temp.t_ws23"
```

Create New Fourier+Xregs Matrices

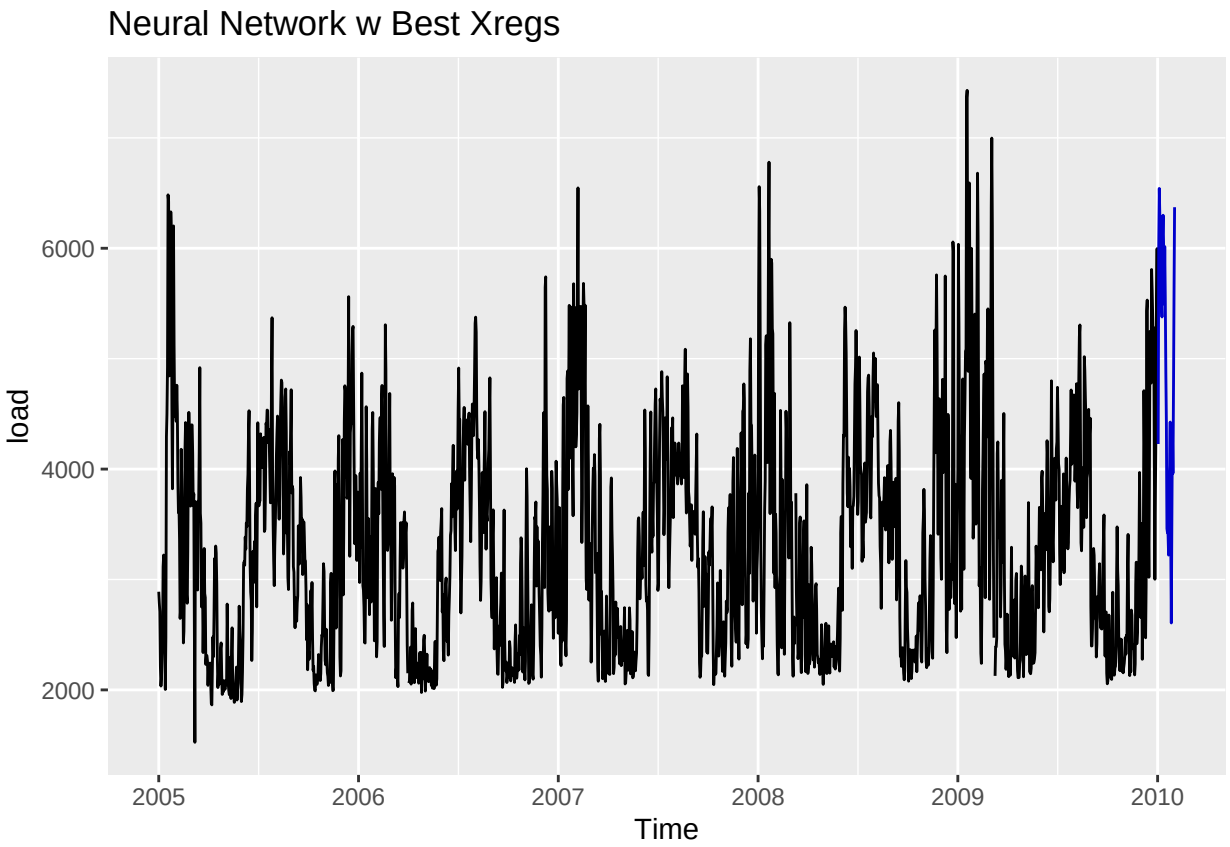
In this chunk, we create matrices that combine our fourier matrices and the vectors of data from the 5 weather stations with highest absolute value found in the previous chunk to be used to train new models and forecast.

```
#create matrix w fourier and 5 best correlations
xregs_5_best <- temp_daily_2[,c(25,5,18,7,23)]
xregs_5_best_train <- xregs_5_best[(1:nrow(fourier_mat)),]
xregs_best_fourier<-cbind(fourier_mat,xregs_5_best_train)
xregs_best_for<-xregs_5_best[(nrow(xregs_best_fourier)+1):(nrow(xregs_best_fourier)+31),]
xregs_best_fourier_for<-cbind(fourier_mat_for,xregs_best_for)
```

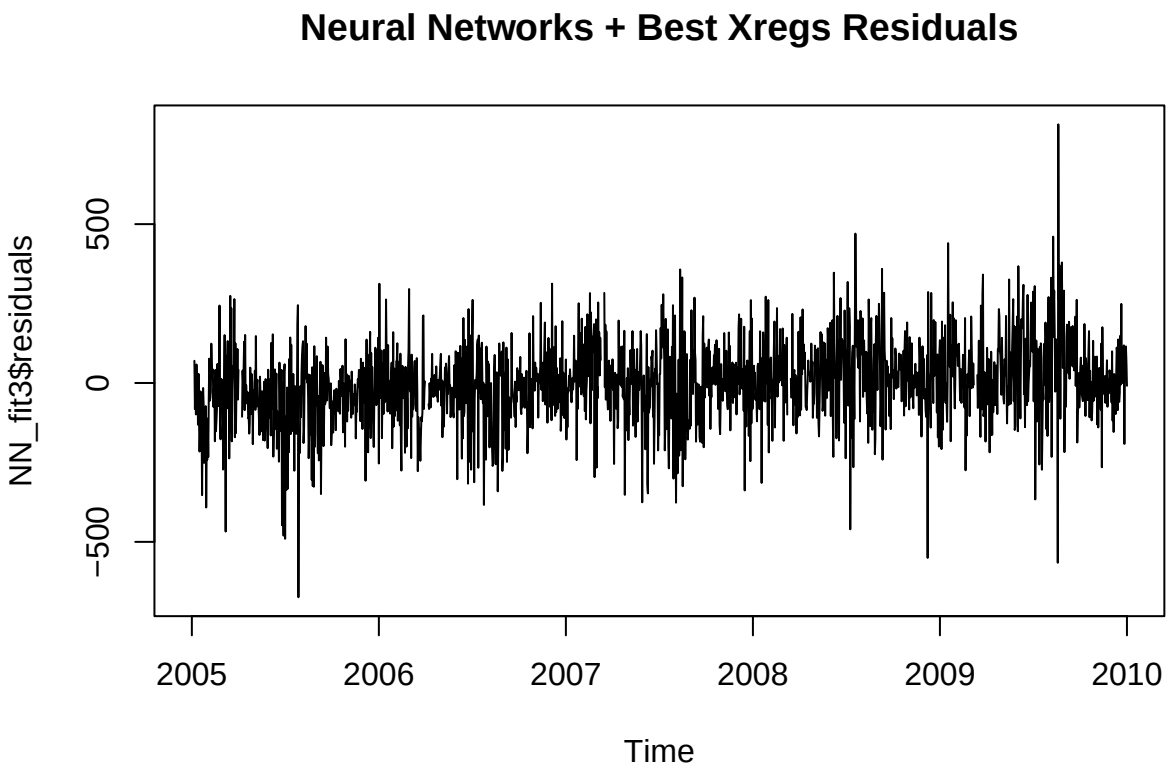
##Rerunning Neural Networks with best weather stations

Our next is to utilize our fourier terms and our 5 best exogenous regressors to rerun the neural network model and forecast. Our residuals look similarly unbiased as before. Our RMSE has improved as we had hoped.

```
##Run neural network with top 5 weather stations and fourier
NN_fit3 <- nnetar(ts_load_09,p=5,P=0,xreg=xregs_best_fourier)
NN_for3 <- forecast::forecast(NN_fit3, h=31,xreg=xregs_best_fourier_for)

#Plot
autoplot(NN_for3, main="Neural Network w Best Xregs") + ylab("load")
```




```
#Plot residuals
ts.plot(NN_fit3$residuals, main="Neural Networks + Best Xregs Residuals")
```




```
#Accuracy Test
NN_scores3 <- accuracy(NN_for3$mean,ts_load_test)
print(NN_scores3)
```

```
##                 ME     RMSE     MAE      MPE     MAPE      ACF1 Theil's U
## Test set 252.2299 466.5146 315.335 3.846067 5.439594 0.4498753 0.4513651
```

Define Fourier and Xregs Matrices for Jan '11 Train and Forecast

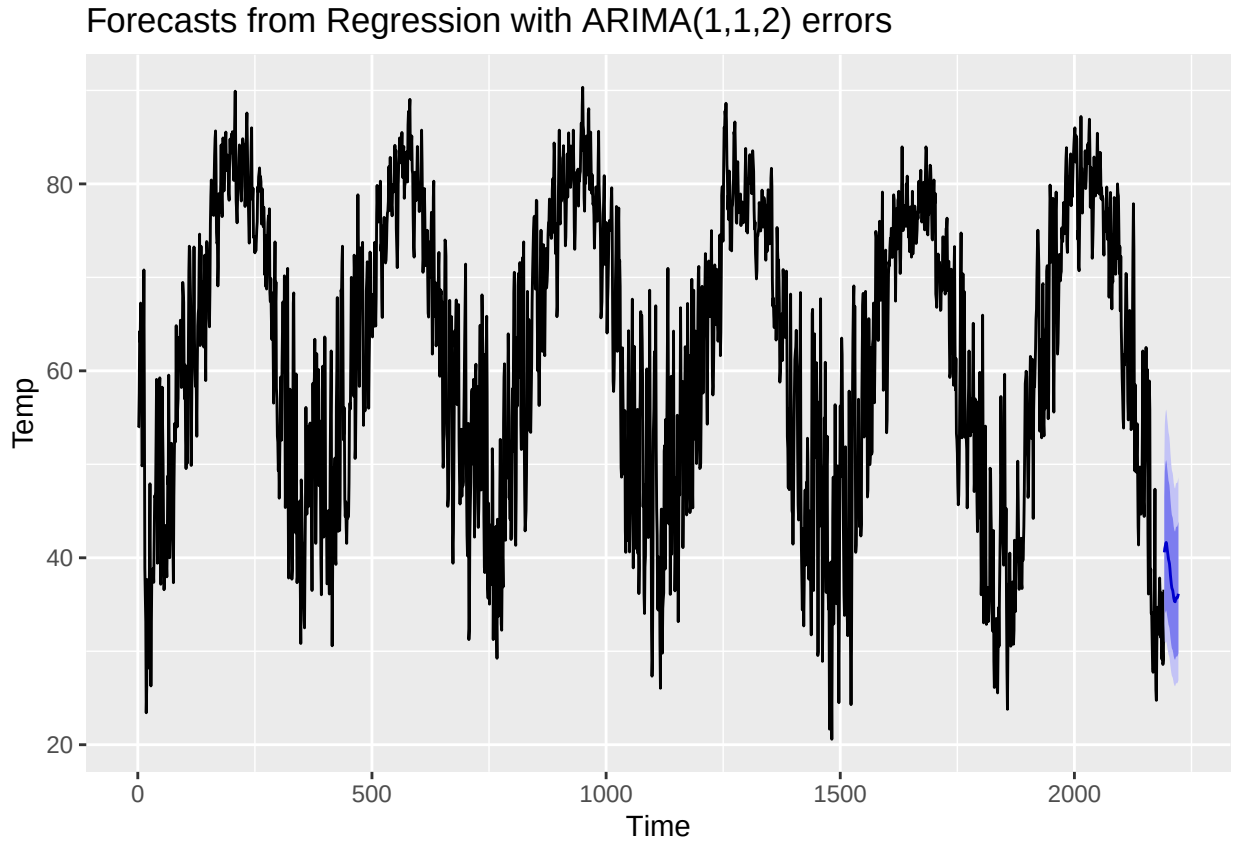
Then we generated training and forecasting matrices of our fourier terms and exogenous regressors.

```
fourier_mat_10_train <- fourier(ts_load,K=c(2,12),h=nrow(ts_load))
fourier_mat_10_for<-fourier(ts_load,K=c(2,12),h=31)
xregs_best_four_mat_10 <- cbind(fourier_mat_10_train,xregs_5_best)
```

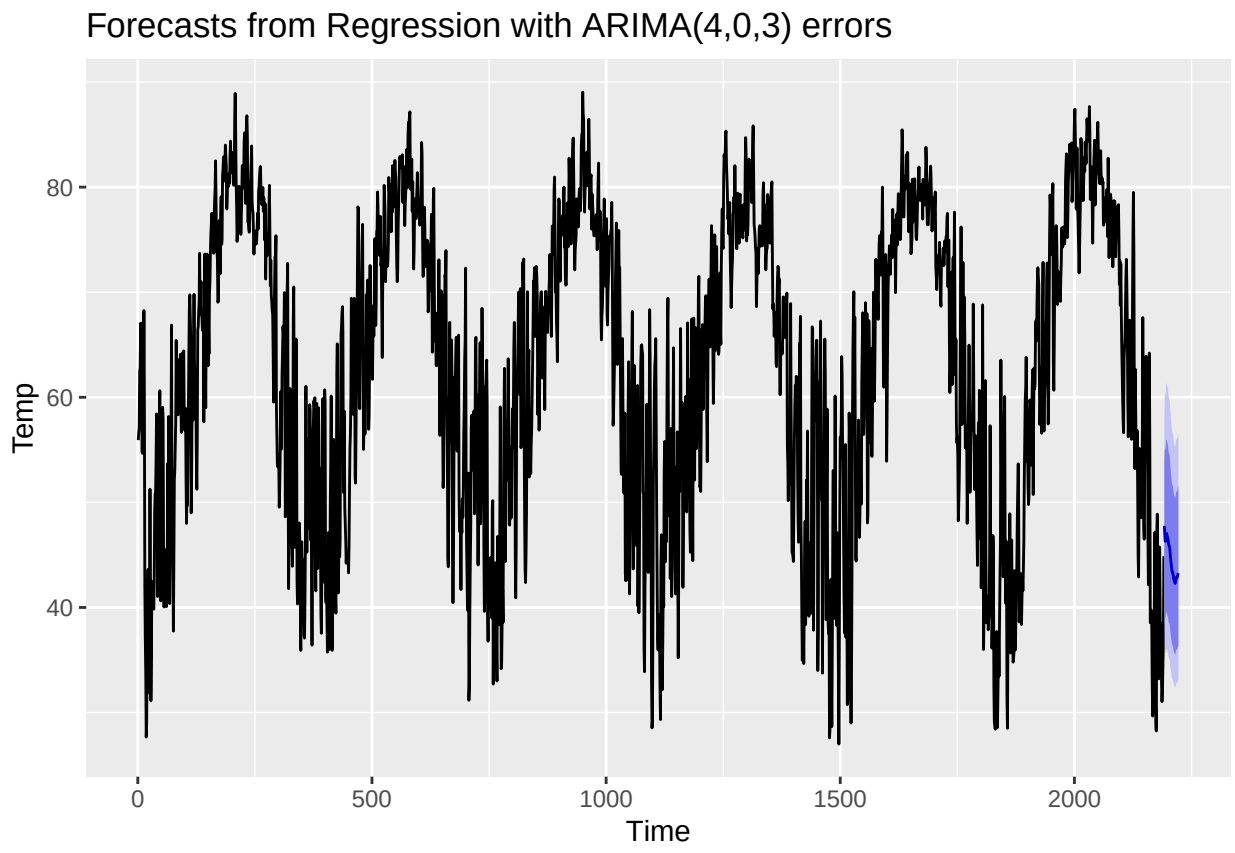
Forecast Xregs for Use in Load Forecast

We then forecasted our exogenous variables so we could use them to forecast the load for January 2011. We first utilized auto.arima models to forecast each of our 5 most correlated temperature variables with load and then our average temperature and humidity across alll weather stations. The plots of these forecasts are shown below.

```r
# forecast xregs
ARIMA_Four_fit_xreg1 <- auto.arima(xregs_5_best$t_ws25, seasonal=FALSE, lambda=0,xreg=fourier_mat_10_tra
ARIMA_Four_fit_xreg2 <- auto.arima(xregs_5_best$t_ws5, seasonal=FALSE, lambda=0,xreg=fourier_mat_10_trai
ARIMA_Four_fit_xreg3 <- auto.arima(xregs_5_best$t_ws18, seasonal=FALSE, lambda=0,xreg=fourier_mat_10_tra
ARIMA_Four_fit_xreg4 <- auto.arima(xregs_5_best$t_ws7, seasonal=FALSE, lambda=0,xreg=fourier_mat_10_trai
ARIMA_Four_fit_xreg5 <- auto.arima(xregs_5_best$t_ws23, seasonal=FALSE, lambda=0,xreg=fourier_mat_10_tra
ARIMA_Four_fit_xreg_Ave_temp <- auto.arima(xregs_daily[,1], seasonal=FALSE, lambda=0,xreg=fourier_mat_1
ARIMA_Four_fit_xreg_Ave_humidity <- auto.arima(xregs_daily[,2], seasonal=FALSE, lambda=0,xreg=fourier_ma

#Forecast with ARIMA fit
#also need to specify h for fourier terms
ARIMA_Four_for_xreg1 <- forecast::forecast(ARIMA_Four_fit_xreg1, xreg=fourier_mat_10_for, h=31)
ARIMA_Four_for_xreg2 <- forecast::forecast(ARIMA_Four_fit_xreg2, xreg=fourier_mat_10_for, h=31)
ARIMA_Four_for_xreg3 <- forecast::forecast(ARIMA_Four_fit_xreg3, xreg=fourier_mat_10_for, h=31)
ARIMA_Four_for_xreg4 <- forecast::forecast(ARIMA_Four_fit_xreg4, xreg=fourier_mat_10_for, h=31)
ARIMA_Four_for_xreg5 <- forecast::forecast(ARIMA_Four_fit_xreg5, xreg=fourier_mat_10_for, h=31)
ARIMA_Four_for_xreg_Ave_temp <- forecast::forecast(ARIMA_Four_fit_xreg_Ave_temp, xreg=fourier_mat_10_fo
ARIMA_Four_for_xreg_Ave_humidity <- forecast::forecast(ARIMA_Four_fit_xreg_Ave_humidity, xreg=fourier_ma

autoplot(ARIMA_Four_for_xreg1) + ylab("Temp")
```



```
autoplot(ARIMA_Four_for_xreg2) + ylab("Temp")
```

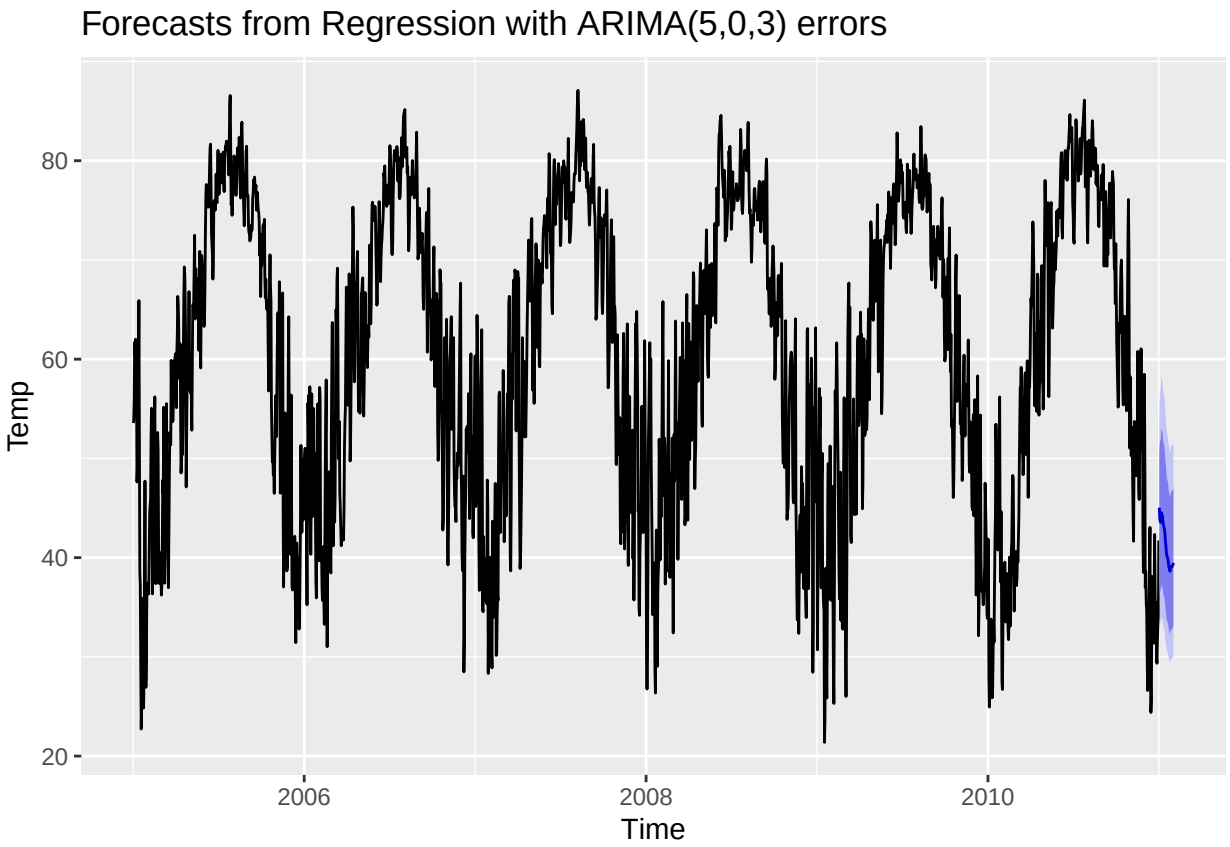



```
autoplot(ARIMA_Four_for_xreg3) + ylab("Temp")
```

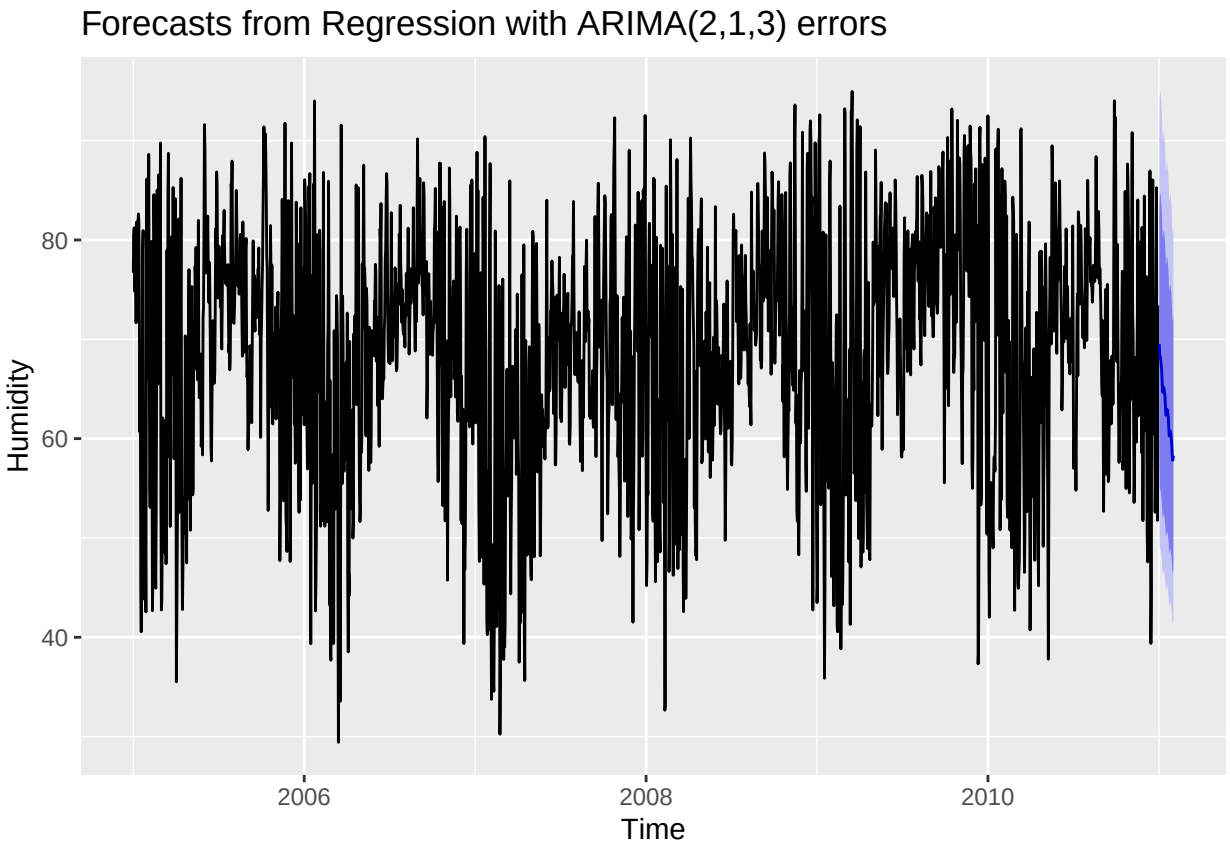

```
autoplot(ARIMA_Four_for_xreg4) + ylab("Temp")
```


```
autoplot(ARIMA_Four_for_xreg5) + ylab("Temp")
```


```
autoplot(ARIMA_Four_for_xreg_Ave_temp) + ylab("Temp")
```



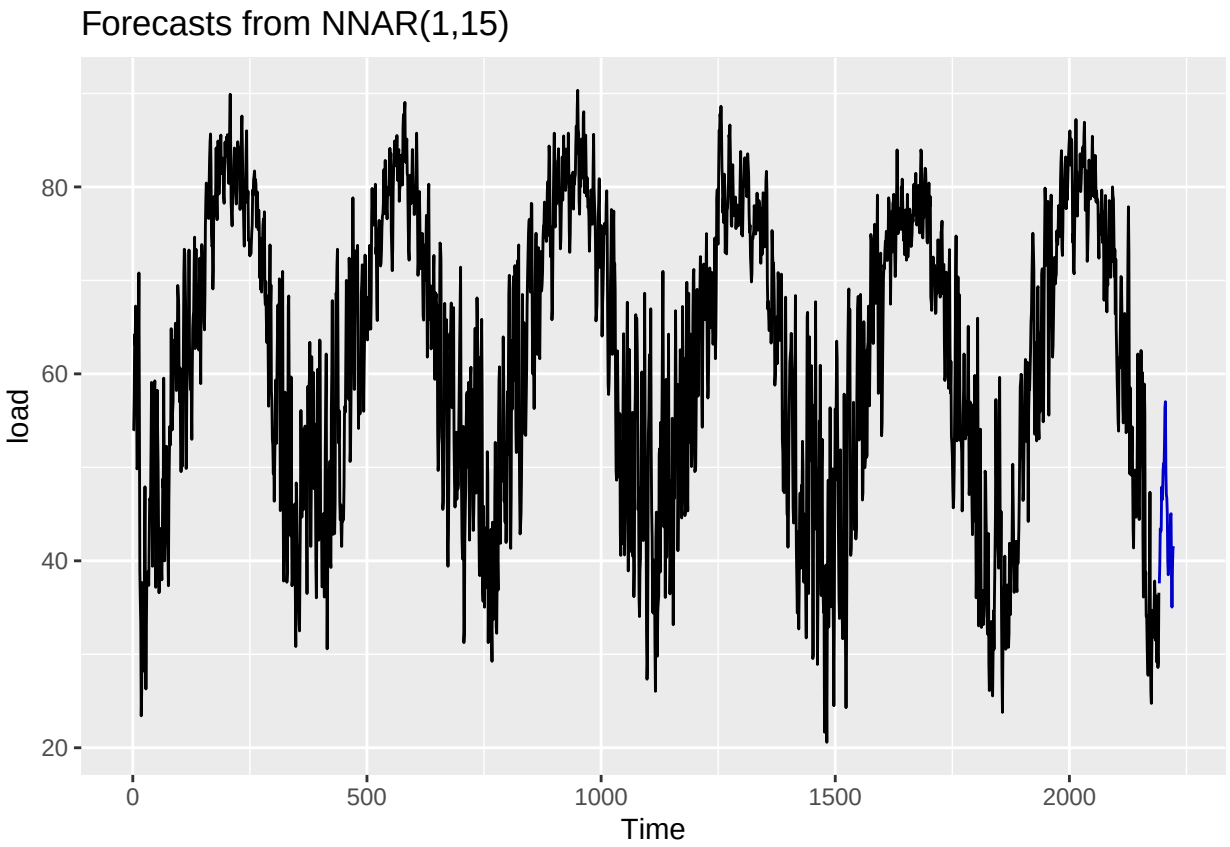

```
autoplot(ARIMA_Four_for_xreg_Ave_humidity) + ylab("Humidity")
```



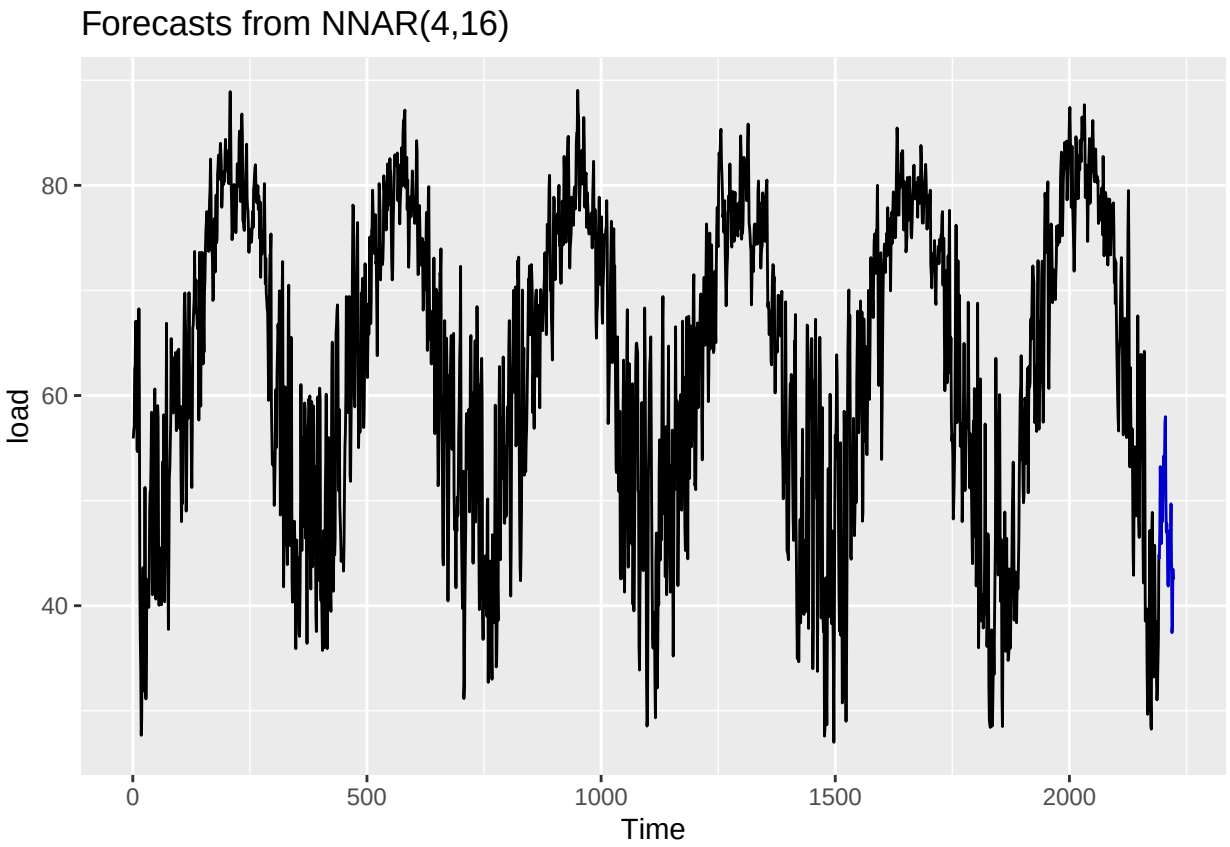
NN Forecast Xregs for Use in Load Forecast

Then we used the p order generated from the auto.arimas to run neural network forecasts of each of the exogenous regressors. Plots shown below.

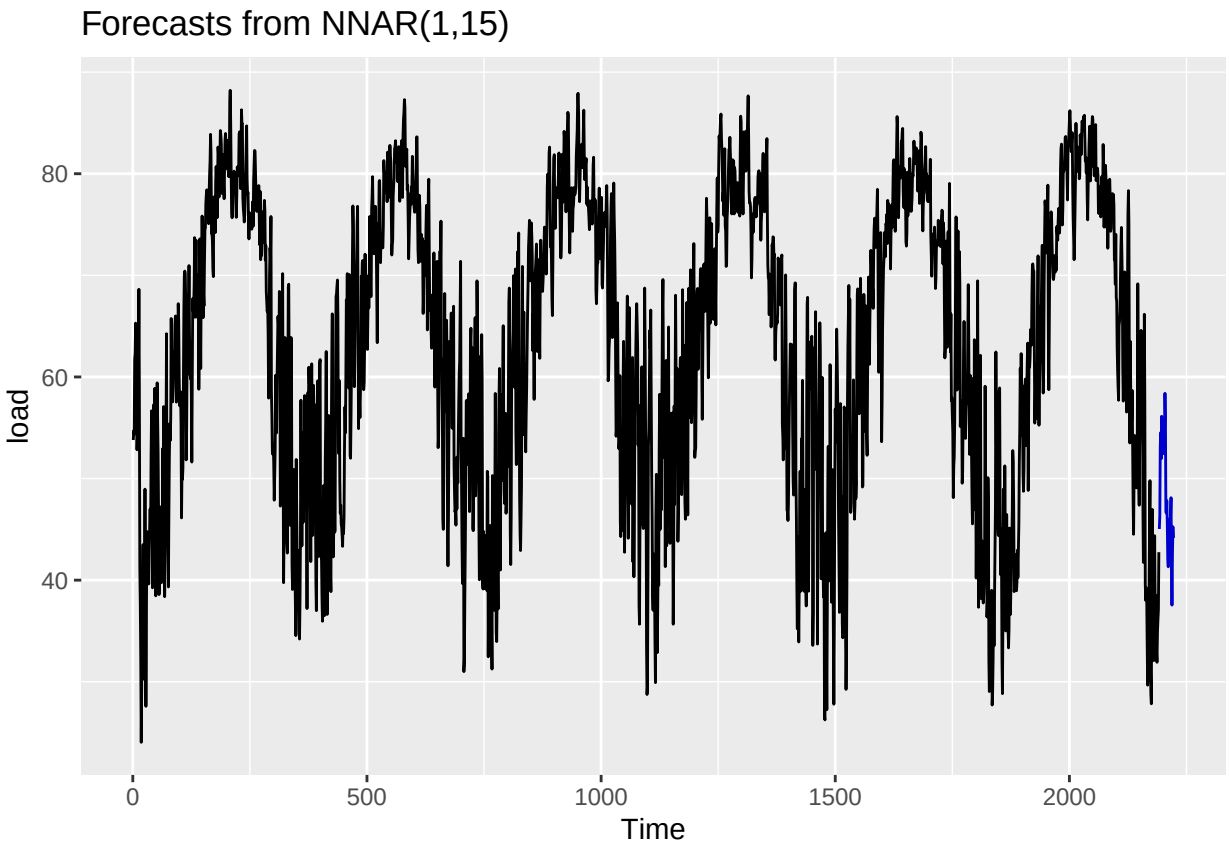
```
#Running NN for xregs using the p produced from Auto.ARIMAs
NN_fit_xreg1 <- nnetar(xregs_5_best$t_ws25,p=1,P=0,xreg=fourier_mat_10_train)
NN_fit_xreg2 <- nnetar(xregs_5_best$t_ws5,p=4,P=0,xreg=fourier_mat_10_train)
NN_fit_xreg3 <- nnetar(xregs_5_best$t_ws18,p=1,P=0,xreg=fourier_mat_10_train)
NN_fit_xreg4 <- nnetar(xregs_5_best$t_ws7,p=3,P=0,xreg=fourier_mat_10_train)
NN_fit_xreg5 <- nnetar(xregs_5_best$t_ws23,p=1,P=0,xreg=fourier_mat_10_train)
NN_fit_xreg_ave_temp <- nnetar(xregs_daily[,1],p=5,P=0,xreg=fourier_mat_10_train)
NN_fit_xreg_ave_humidity <- nnetar(xregs_daily[,2],p=2,P=0,xreg=fourier_mat_10_train)

#Forecasts
NN_for_xreg1 <- forecast::forecast(NN_fit_xreg1, h=31,xreg=fourier_mat_10_for)
NN_for_xreg2 <- forecast::forecast(NN_fit_xreg2, h=31,xreg=fourier_mat_10_for)
NN_for_xreg3 <- forecast::forecast(NN_fit_xreg3, h=31,xreg=fourier_mat_10_for)
NN_for_xreg4 <- forecast::forecast(NN_fit_xreg4, h=31,xreg=fourier_mat_10_for)
NN_for_xreg5 <- forecast::forecast(NN_fit_xreg5, h=31,xreg=fourier_mat_10_for)
NN_for_xreg_ave_temp <- forecast::forecast(NN_fit_xreg_ave_temp, h=31,xreg=fourier_mat_10_for)
NN_for_xreg_ave_humidity <- forecast::forecast(NN_fit_xreg_ave_humidity, h=31,xreg=fourier_mat_10_for)

#Plots
autoplot(NN_for_xreg1) + ylab("load")
```



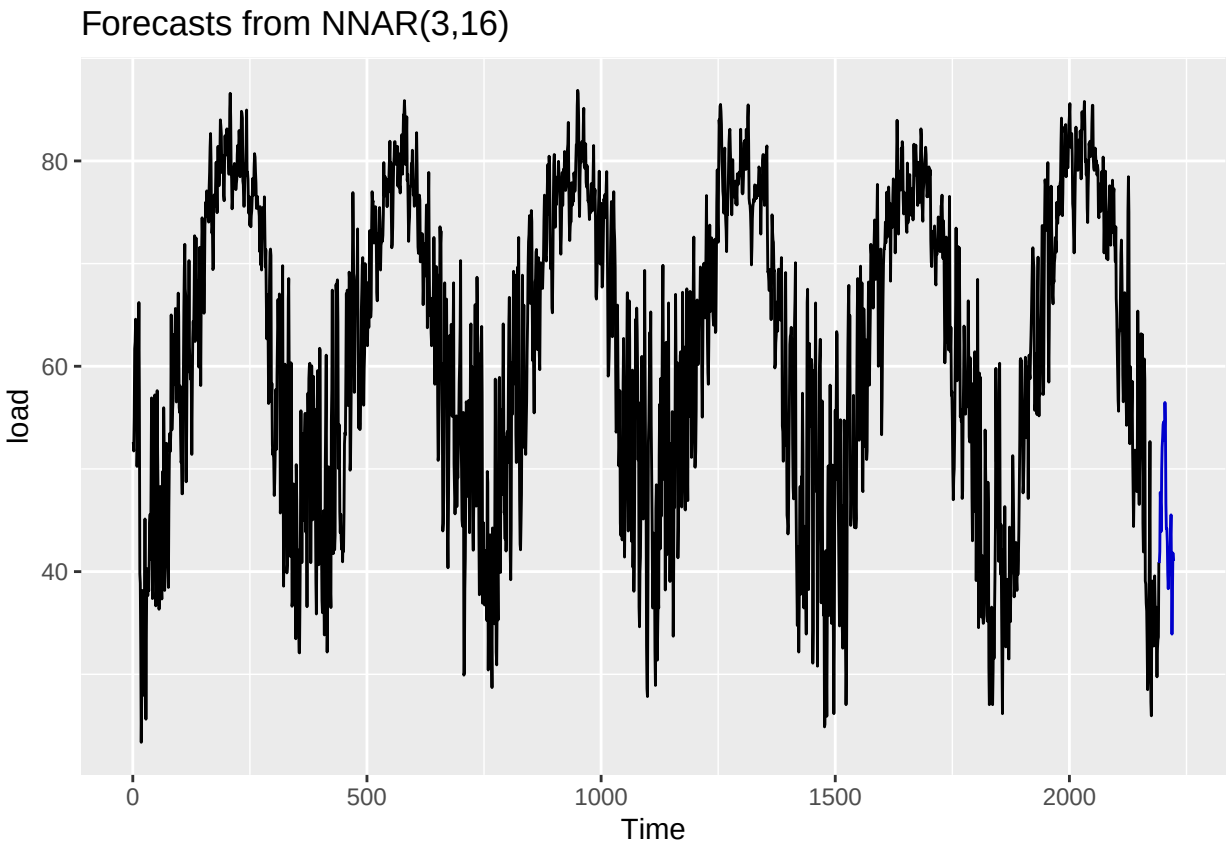

```
autoplot(NN_for_xreg2) + ylab("load")
```



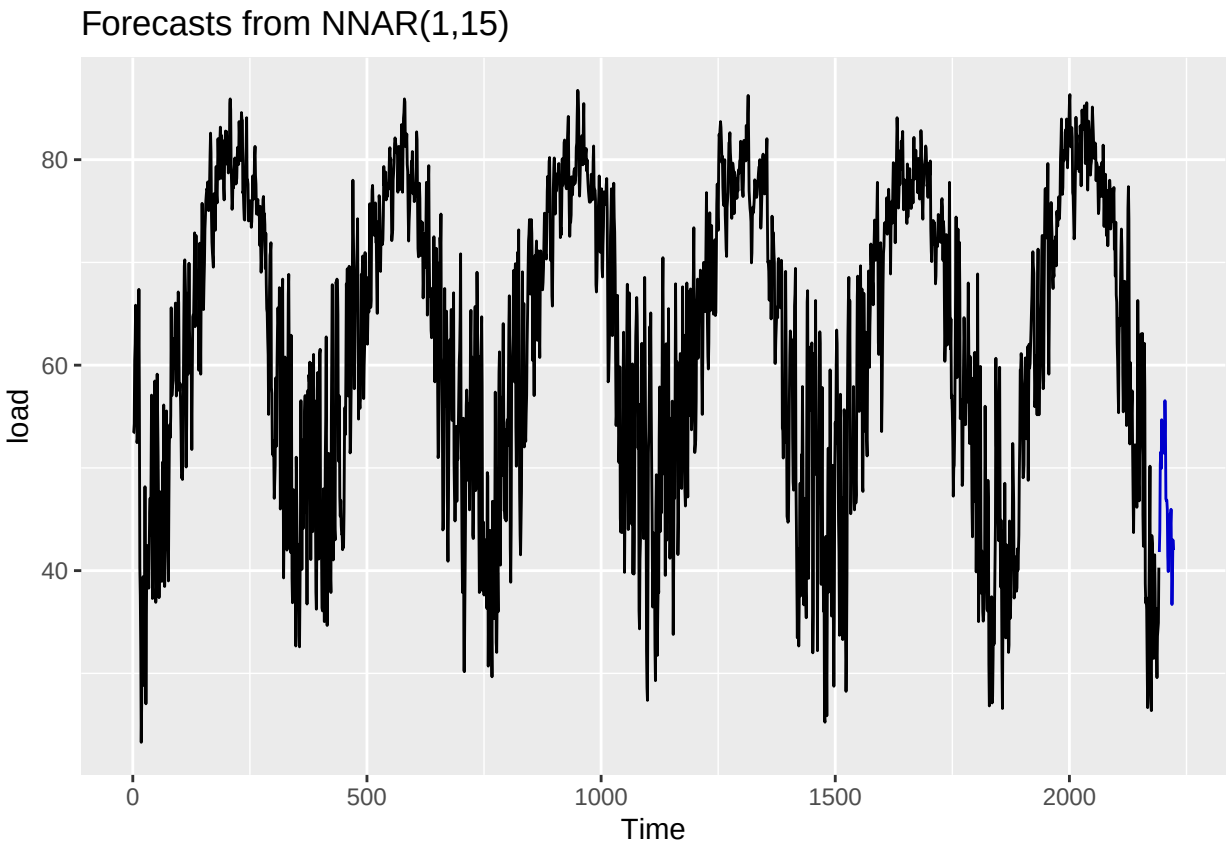

```
autoplot(NN_for_xreg3) + ylab("load")
```



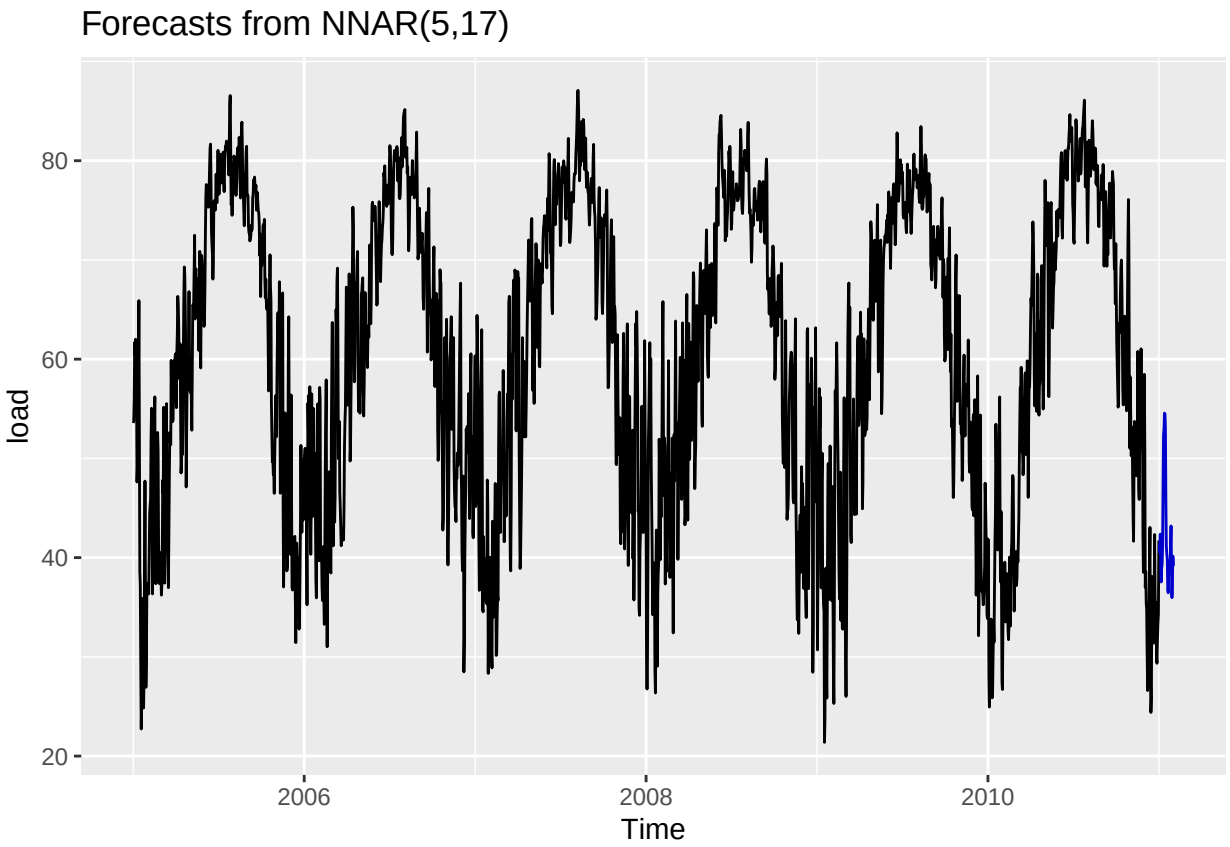

```
autoplot(NN_for_xreg4) + ylab("load")
```



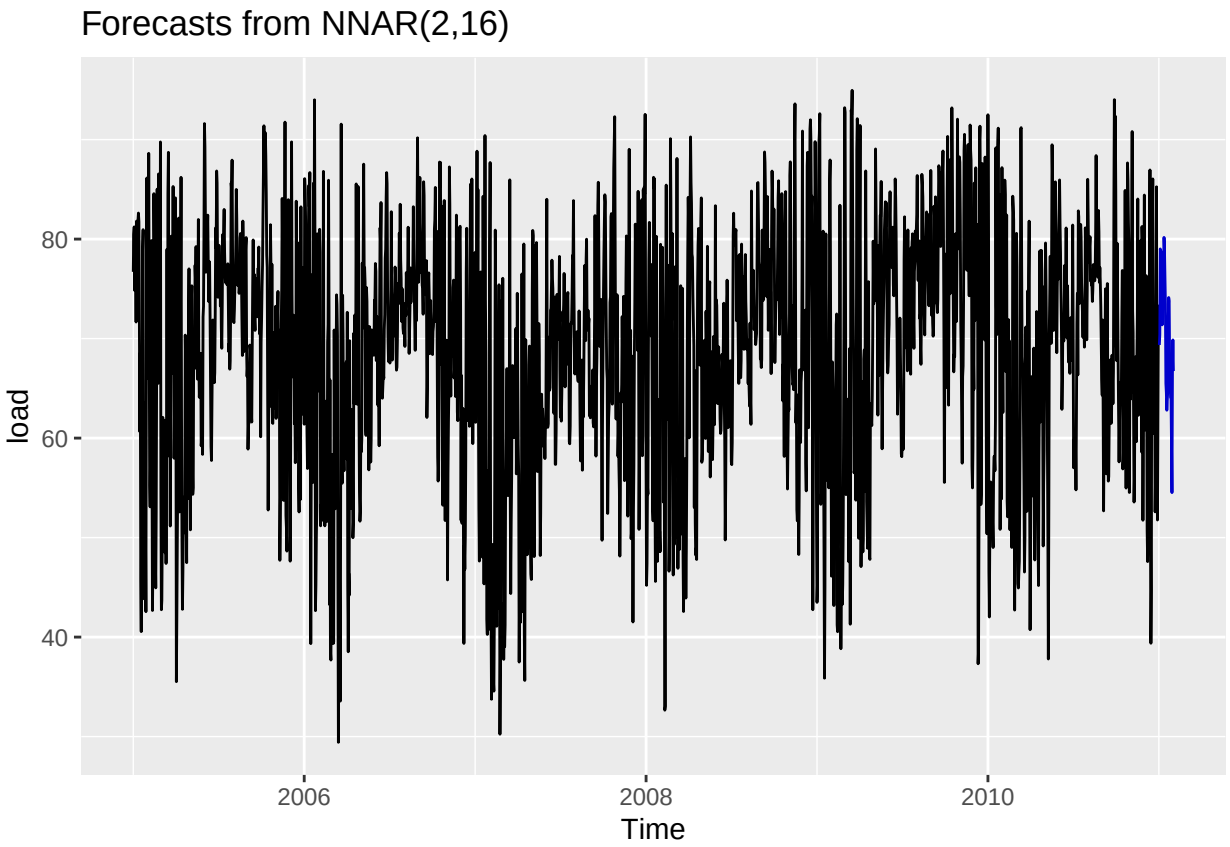

```
autoplot(NN_for_xreg5) + ylab("load")
```




```
autoplot(NN_for_xreg_ave_temp) + ylab("load")
```




```
autoplot(NN_for_xreg_ave_humidity) + ylab("load")
```



Define Fourier and Xregs Matrices for Jan '11 Forecast with Xreg Forecasts

Then we generated training and forecasting matrices of our fourier terms and exogenous regressor forecasts.

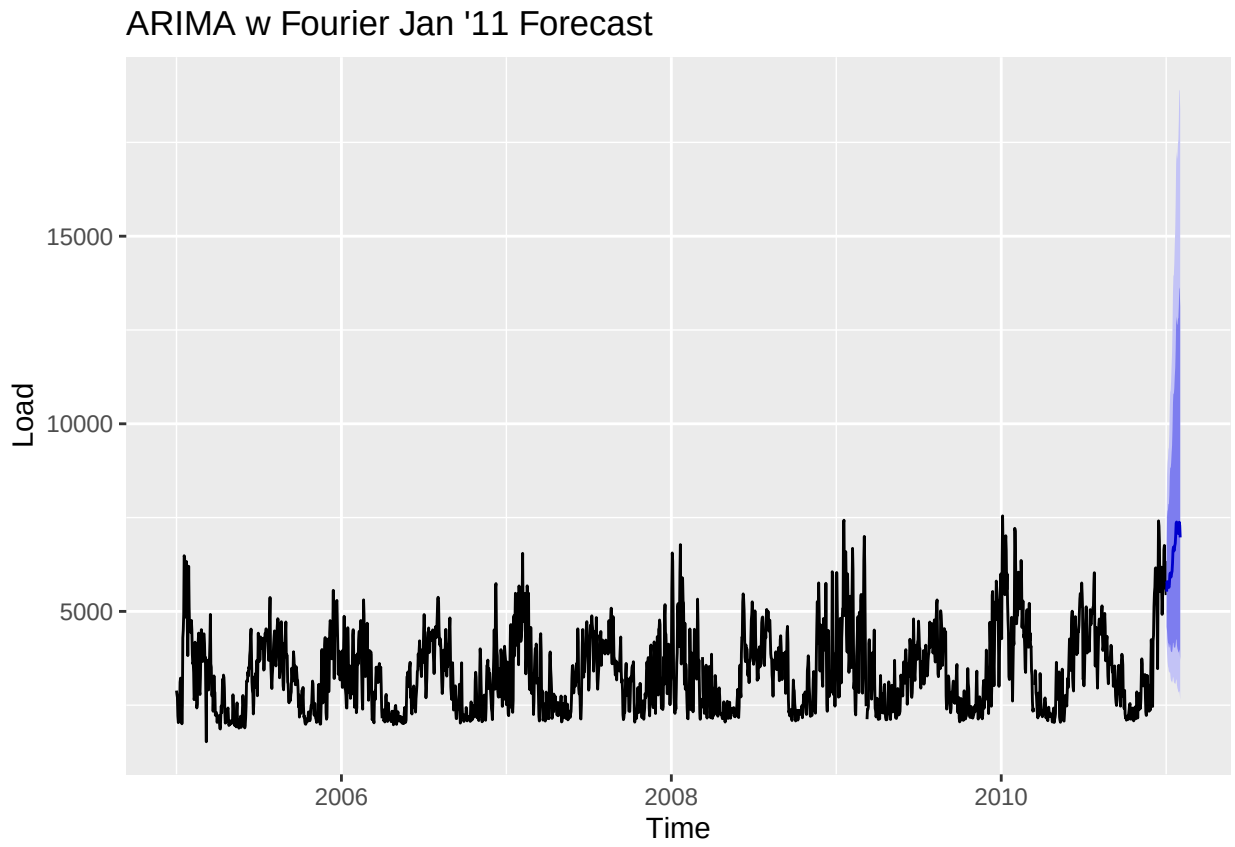
```
xreg_best_for <- cbind(NN_for_xreg1$mean,NN_for_xreg2$mean,NN_for_xreg3$mean,NN_for_xreg4$mean,NN_for_x
xreg_ave_four_10  <- cbind(fourier_mat_10_train, xregs_daily)
xreg_ave_fourier_for <- cbind(fourier_mat_10_for, NN_for_xreg_ave_temp$mean,NN_for_xreg_ave_humidity$mea
xregs_best_fourier_for <- cbind(fourier_mat_10_for, xreg_best_for)
```

Run Jan '11 ARIMA Forecasts and Export

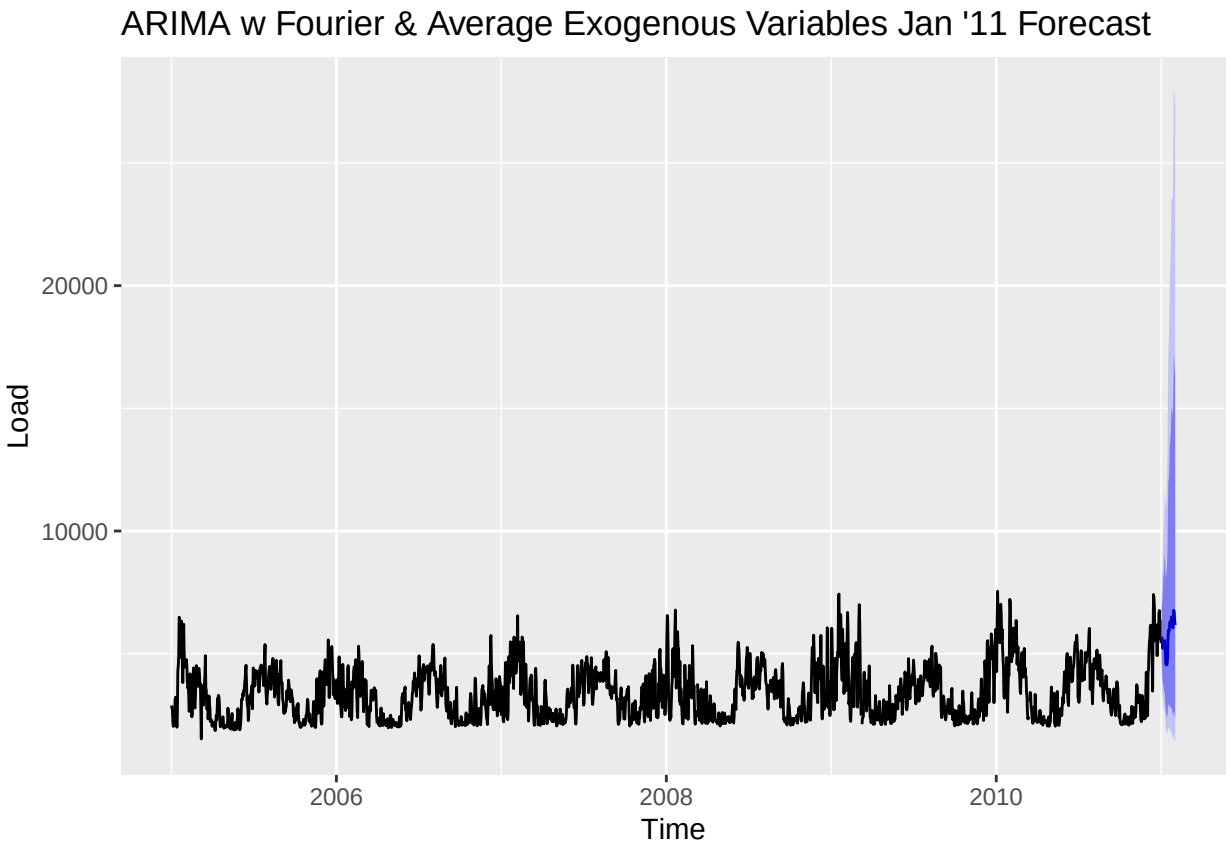
We ran 3 different forecasts of January '11 load data using ARIMA models. The first did not include exogenous variables. The second used our average temperature and humidity values across all weather stations as exogenous variables with fourier terms. The third used our top 5 most correlated weather station temperature variables as exogenous variables with fourier terms. We exported these forecasts and uploaded to kaggle. See plots below.

```
##ARIMA with fourier
ARIMA_Four_fit_Jan11 <- auto.arima(ts_load,seasonal=FALSE,lambda=0,xreg=fourier_mat_10_train)
ARIMA_Four_for_Jan11 <- forecast::forecast(ARIMA_Four_fit_Jan11,xreg=fourier_mat_10_for,h=31)

##ARIMA with average xregs and fourier
ARIMA_Four_AveXreg_fit_Jan11 <- auto.arima(ts_load,seasonal=FALSE,lambda=0,xreg=xreg_ave_four_10)
ARIMA_Four_AveXreg_for_Jan11 <- forecast::forecast(ARIMA_Four_AveXreg_fit_Jan11,xreg=xreg_ave_fourier_f
```

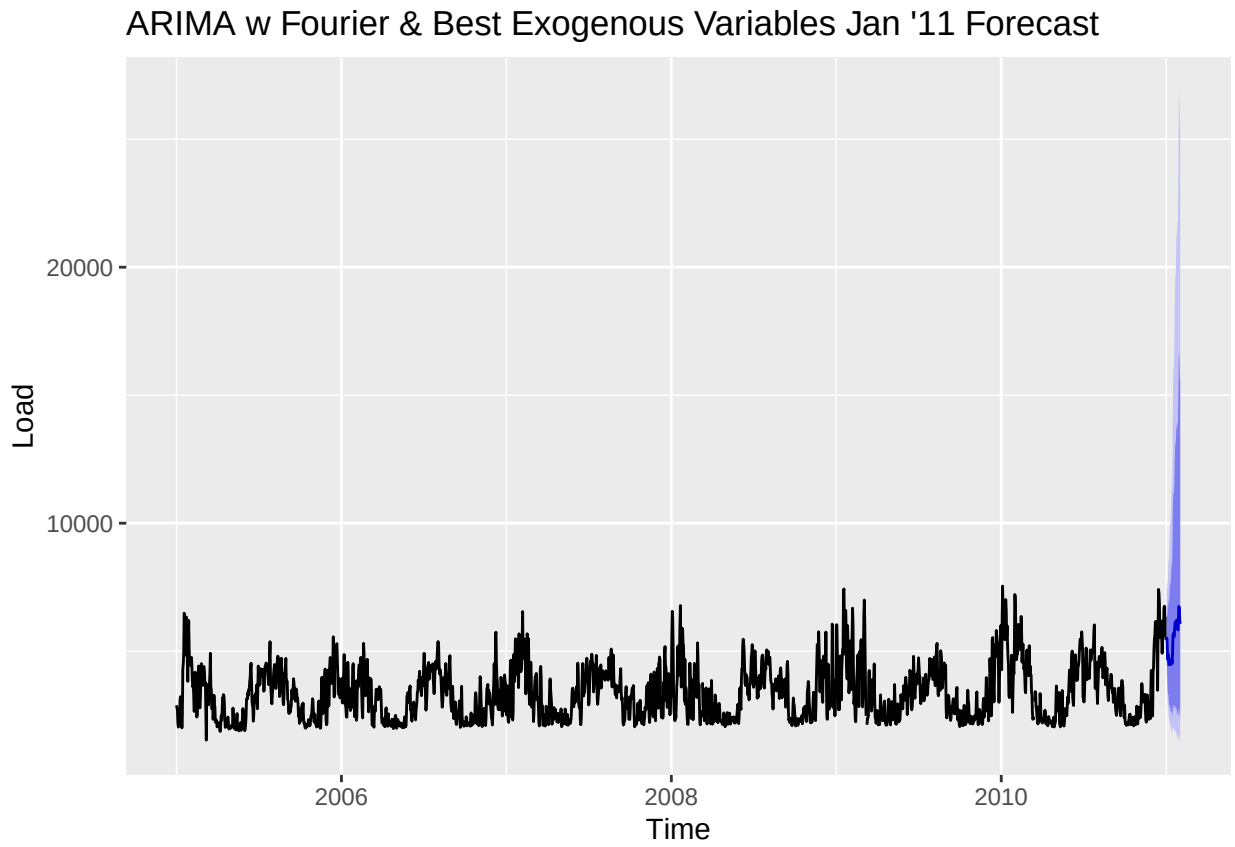
```
##ARIMA with best xregs and fourier
ARIMA_Four_BestXreg_fit_Jan11 <- auto.arima(ts_load,seasonal=FALSE,lambda=0,xreg=as.matrix(xregs_best_f
ARIMA_Four_BestXreg_for_Jan11 <- forecast::forecast(ARIMA_Four_BestXreg_fit_Jan11,xreg=xregs_best_fouri

#Plots
autoplot(ARIMA_Four_for_Jan11, main="ARIMA w Fourier Jan '11 Forecast") + ylab("Load")
```




```
autoplot(ARIMA_Four_AveXreg_for_Jan11, main="ARIMA w Fourier & Average Exogenous Variables Jan '11 Fore
```




```
autoplot(ARIMA_Four_BestXreg_for_Jan11, main="ARIMA w Fourier & Best Exogenous Variables Jan '11 Foreca
```



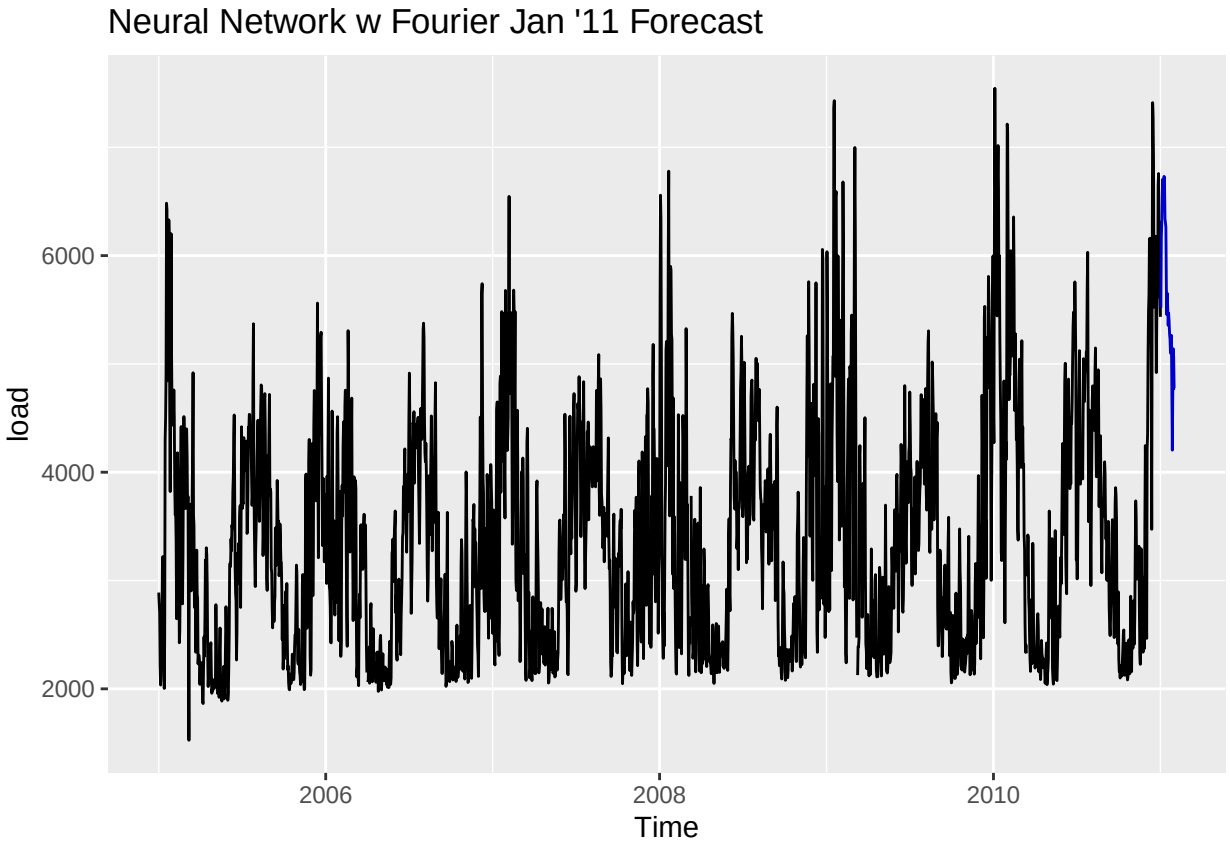

```
#Export
write.csv(ARIMA_Four_for_Jan11$mean,"./ARIMA_Four_for_Jan11.csv", row.names = FALSE)
write.csv(ARIMA_Four_AveXreg_for_Jan11$mean,"./ARIMA_Four_AveXreg_for_Jan11.csv", row.names = FALSE)
write.csv(ARIMA_Four_BestXreg_for_Jan11$mean,"./ARIMA_Four_BestXreg_for_Jan11.csv", row.names = FALSE)
```

##Run Jan '11 Neural Network Forecasts and Export

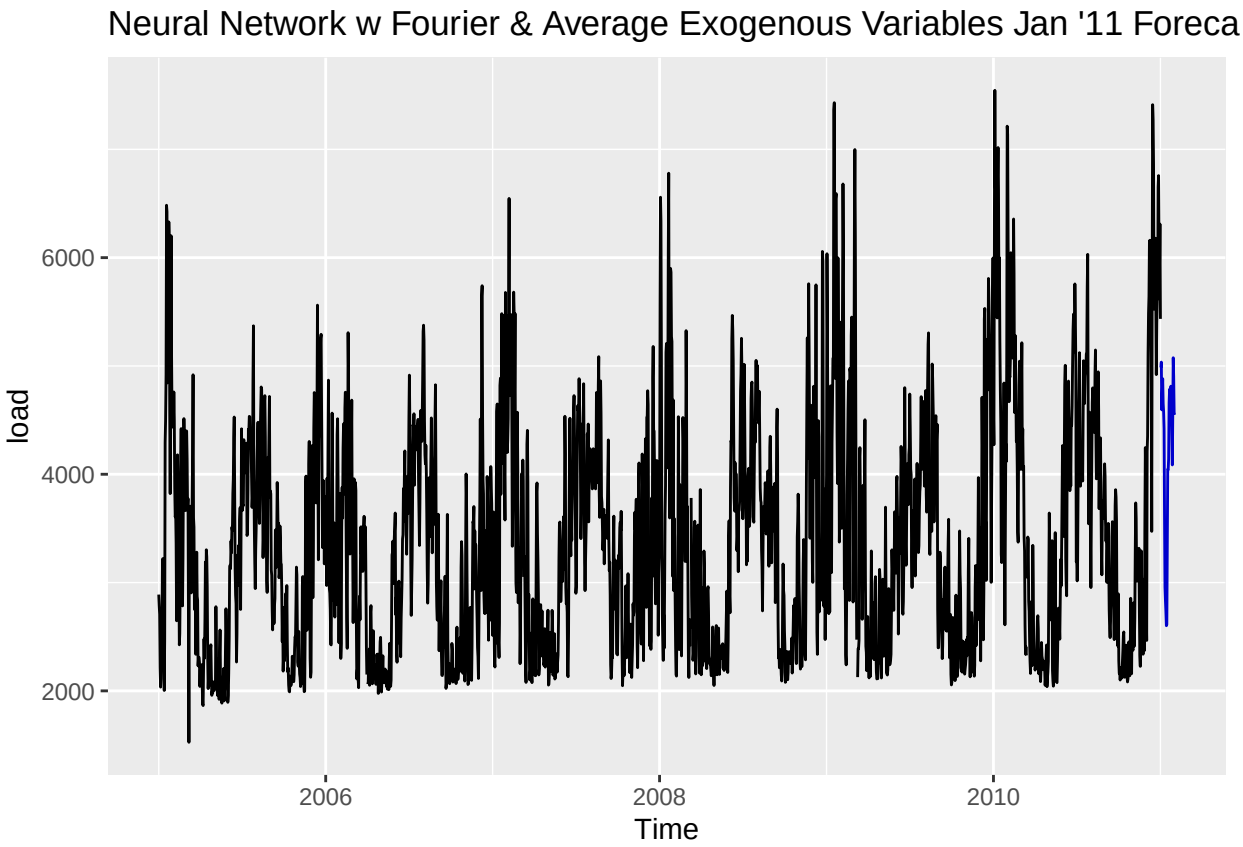
We ran 3 different forecasts of January '11 load data using Neural Network models. The first did not include exogenous variables. The second used our average temperature and humidity values across all weather stations as exogenous variables with fourier terms. The third used our top 5 most correlated weather station temperature variables as exogenous variables with fourier terms. We exported these forecasts and uploaded to kaggle. See plots below. These forecasts look much more volatile than the ARIMA forecasts. This is neither a bad nor a good thing but a significant change nonetheless.

```
##Neural Network with Fourier
NN_Four_fit_Jan11 <- nnetar(ts_load,p=5,P=0,xreg=fourier_mat_10_train)
NN_Four_for_Jan11 <- forecast::forecast(NN_Four_fit_Jan11, h=31,xreg=fourier_mat_10_for)

##Neural Network with Fourier and Average Xregs
NN_Four_AveXreg_fit_Jan11 <- nnetar(ts_load,p=5,P=0,xreg=xreg_ave_four_10)
NN_Four_AveXreg_for_Jan11 <- forecast::forecast(NN_Four_AveXreg_fit_Jan11, h=31,xreg=xreg_ave_fourier_f

##Neural Network with Fourier and Best Xregs
NN_Four_BestXreg_fit_Jan11 <- nnetar(ts_load,p=5,P=0,xreg=xregs_best_four_mat_10)
NN_Four_BestXreg_for_Jan11 <- forecast::forecast(NN_Four_BestXreg_fit_Jan11, h=31,xreg=xregs_best_fourie
```

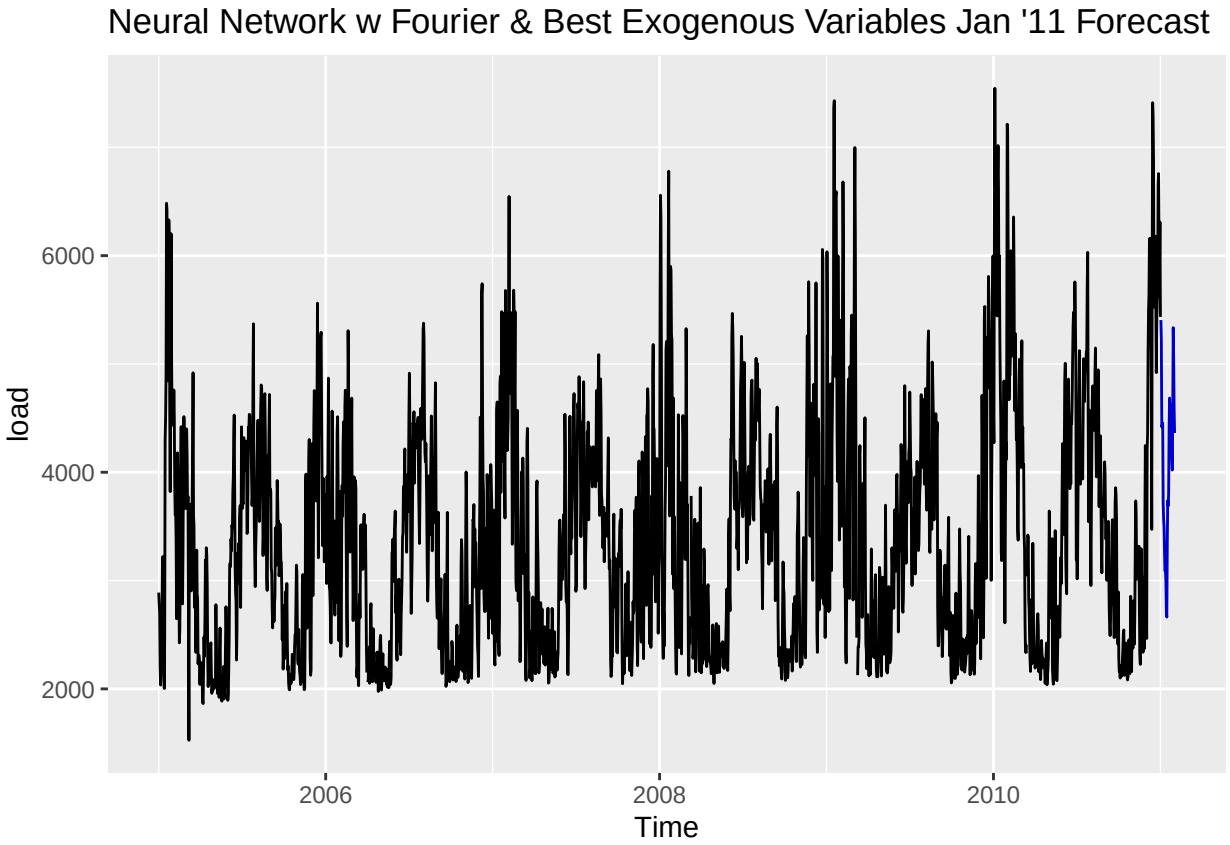
```
#Plots
autoplot(NN_Four_for_Jan11, main="Neural Network w Fourier Jan '11 Forecast") + ylab("load")
```




```
autoplot(NN_Four_AveXreg_for_Jan11, main="Neural Network w Fourier & Average Exogenous Variables Jan '1
```




```
autoplot(NN_Four_BestXreg_for_Jan11, main="Neural Network w Fourier & Best Exogenous Variables Jan '11 F
```




```
##Export
write.csv(NN_Four_for_Jan11$mean,"./NN_Four_for_Jan11.csv", row.names = FALSE)
write.csv(NN_Four_AveXreg_for_Jan11$mean,"./NN_Four_AveXreg_for_Jan11.csv", row.names = FALSE)
write.csv(NN_Four_BestXreg_for_Jan11$mean,"./NN_Four_BestXreg_for_Jan11.csv", row.names = FALSE)
```

Averaged Models

Our last forecasting attempt was to take the averages across all forecasts, export it, and upload it as another kaggle forecast.

```
#Create dataframe with all 6 previous Jan '11 load forecasts
All_ARIMA_NN <- as.data.frame(cbind(ARIMA_Four_for_Jan11$mean, ARIMA_Four_AveXreg_for_Jan11$mean, ARIMA_

#Average all the daily forecasts
empty_col<-("All ARIMA NN Averages")
All_ARIMA_NN[,empty_col]<-NA
All_ARIMA_NN$`All ARIMA NN Averages`<-rowMeans(All_ARIMA_NN[,(1:5)])
averaged_All_ARIMA_NN <- All_ARIMA_NN[,6]

#Export as csv
write.csv(averaged_All_ARIMA_NN,"./averaged_All_ARIMA_NN.csv", row.names = FALSE)
```