Visionary Final

Justin DePue, John Rooney, and Tony Jiang

2023-04-26

library packages

```
library(tseries)
library(lubridate)
library(here)
library(tidyverse)
library(outliers)
library(ggplot2)
library(forecast)
library(kableExtra)
library(readxl)
```

load the data

```
# automatically set the working directory as personal local path to R project
# so the file path in read.csv can work for everyone
setwd(here())

# load the data
electricity_prices.df <- read.csv("./Data/Average_retail_price_of_electricity.csv", header = TRUE, skip=

nc_electricity.df<-electricity_prices.df[228,4:(ncol(electricity_prices.df))] %>%
  pivot_longer(cols=everything(), names_to = 'my_date', values_to = 'price_per_kWh')

# examine whether there is missing value or not
summary(nc_electricity.df)
```

```
##    my_date          price_per_kWh
##  Length:265         Length:265
##  Class :character   Class :character
##  Mode  :character   Mode  :character
```

```
# there is no missing value

nc_electricity.df$price_per_kWh<-as.numeric(nc_electricity.df$price_per_kWh)
```

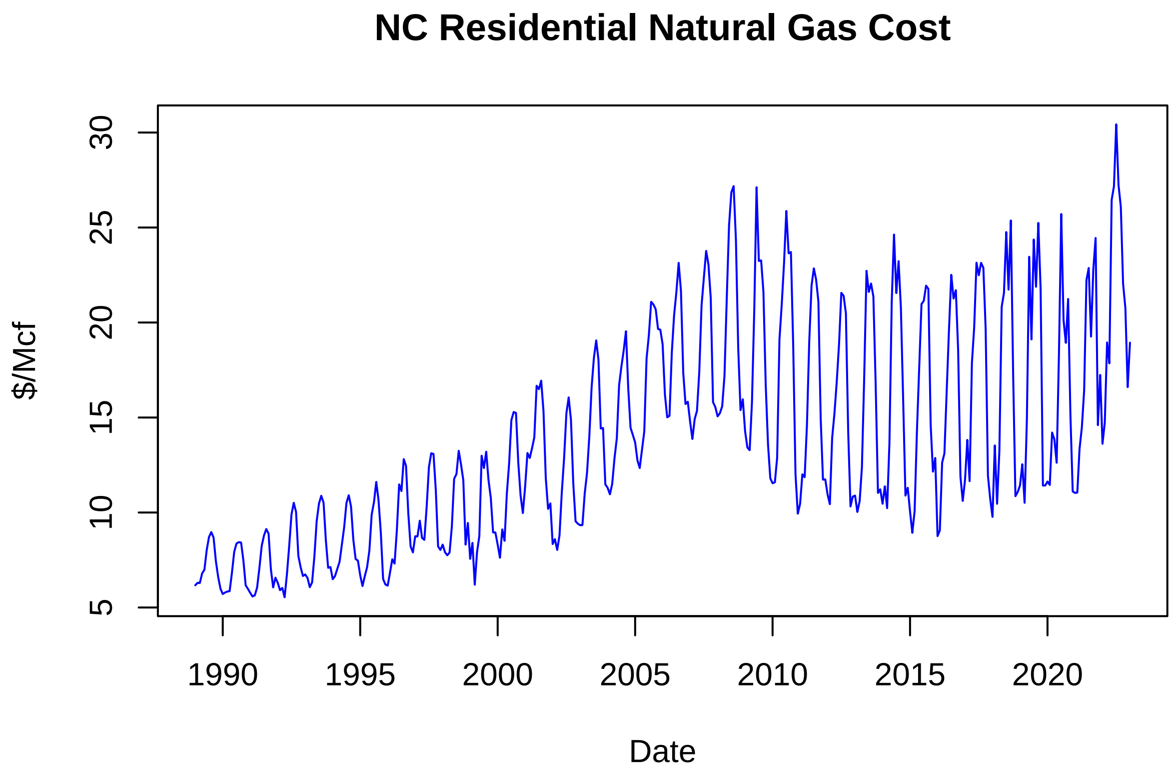
```r
# Import Natural Gas data

natural_gas.df <- read.csv("./Data/NC_NaturalGas.csv", header = TRUE, skip=2,col.names = c("year", "pri

#Check for missing data. There is an extra row with an NA at the end of the data, so I removed it with

summary(na.omit(natural_gas.df))
```

```
##      year               price
##  Length:409         Min.   : 5.54
##  Class :character   1st Qu.: 9.07
##  Mode  :character   Median :12.54
##                     Mean   :13.78
##                     3rd Qu.:18.11
##                     Max.   :30.43
```

```r
str(natural_gas.df)
```

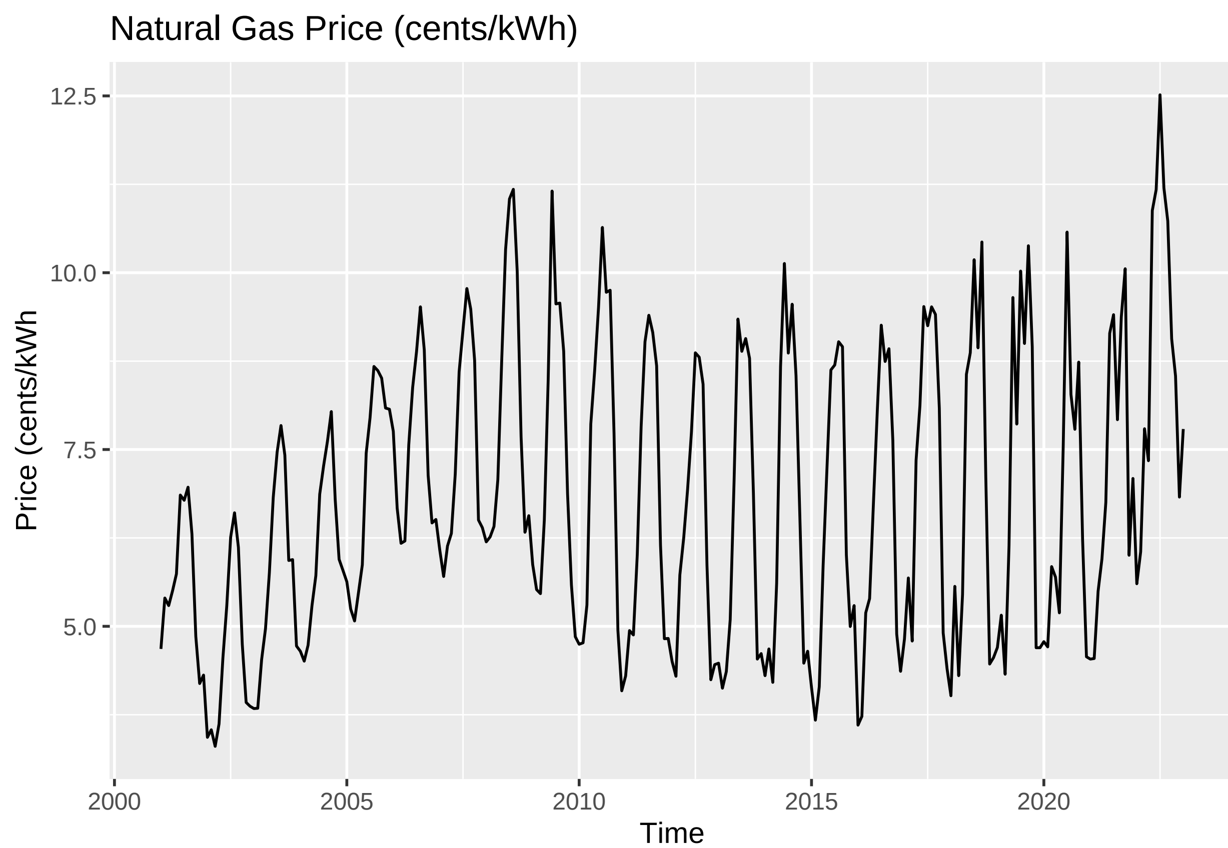
```
## 'data.frame':    410 obs. of  2 variables:
##  $ year : chr  "Jan-1989" "Feb-1989" "Mar-1989" "Apr-1989" ...
##  $ price: num  6.17 6.3 6.29 6.8 6.99 8.02 8.71 8.97 8.68 7.44 ...
```

```r
#create timeseries for natural gas

ts_NG<-ts(na.omit(natural_gas.df[,2]), start=c(1989,1), frequency=12)

# plot raw time series
plot(ts_NG, col="blue", ylab="$/Mcf", xlab="Date", main="NC Residential Natural Gas Cost")
```

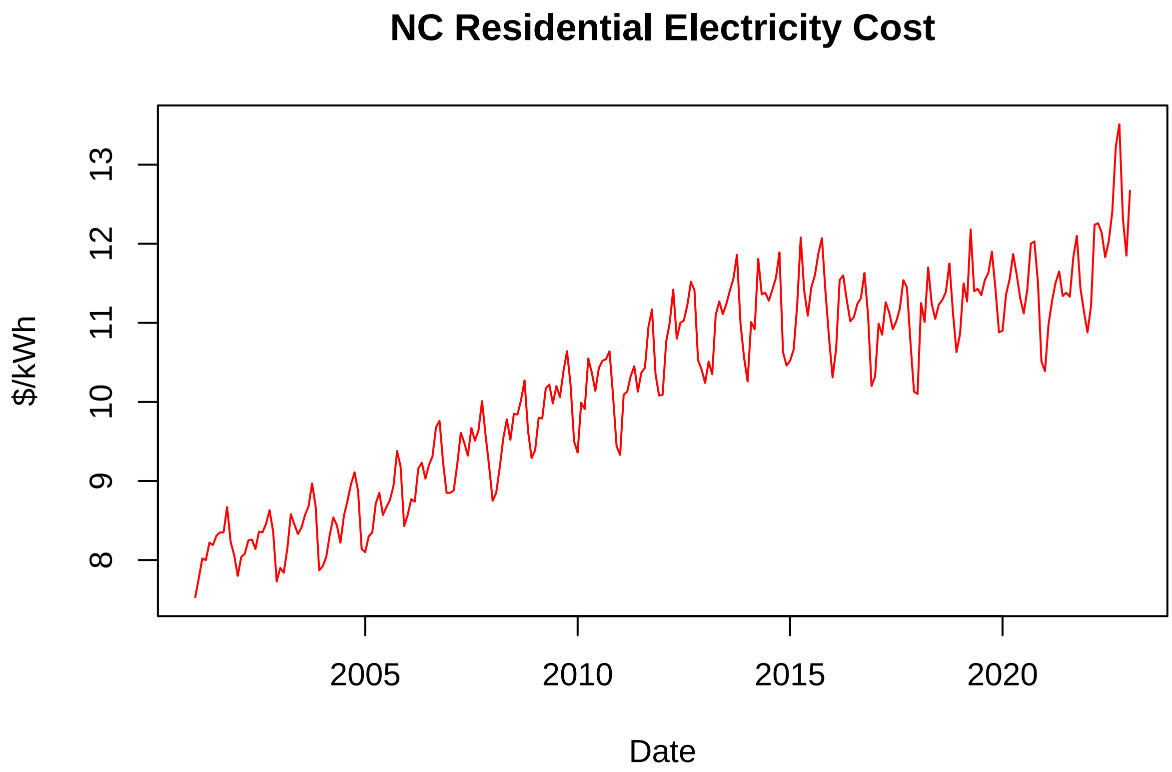


standardize the unit of natural gas data to make it comparable with electricity

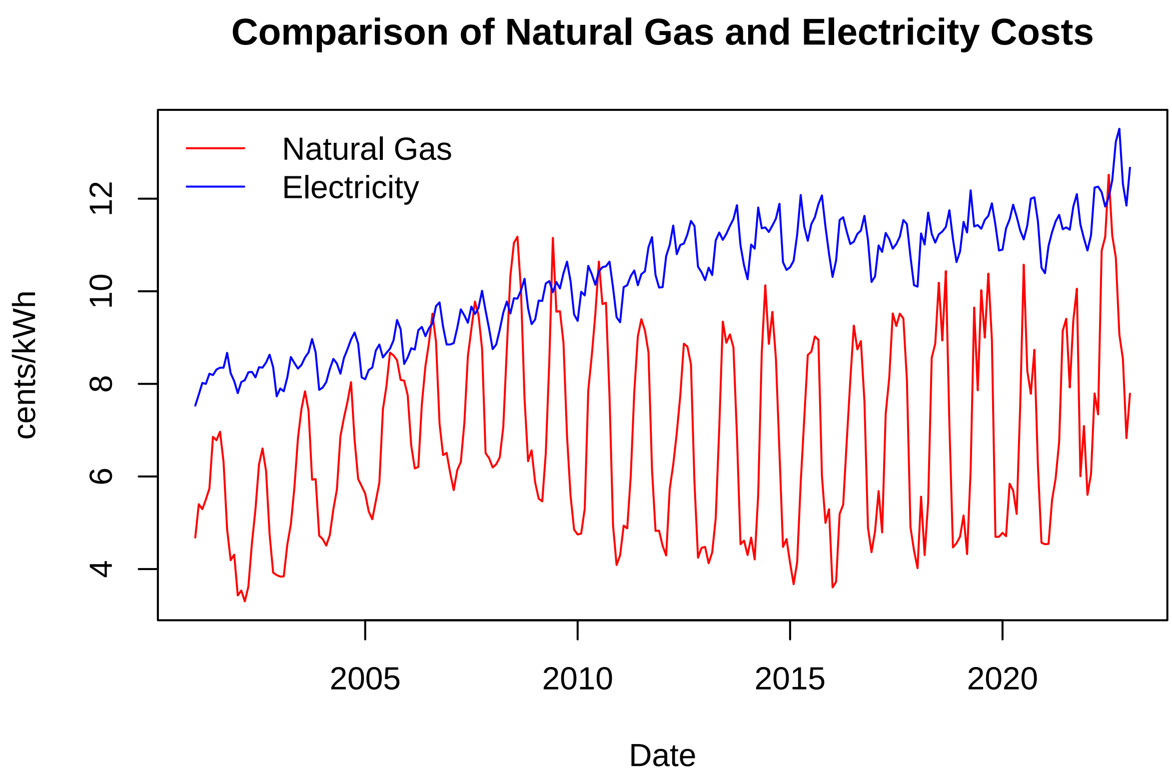
```
# convert natural gas data from $/Mcf to $/kWh based on 80% heating efficiency
# $/kWh = [($/mcf/1.037)/293.07107]/0.9 = $/mcf/273.52


conversion <- 0.8*293.07107*1.037/100


natural_gas.df$kwh_equiv<-((natural_gas.df$price)/conversion)


natural_gas.df<-na.omit(natural_gas.df)

ts_gas_equiv<-ts(natural_gas.df[,3], start=c(1989,1), frequency=12)

ts_gas_equiv<-window(ts_gas_equiv, start=c(2001, 1))

#plot gas TS in kw/hr equiv
autoplot(ts_gas_equiv) +
  ylab("Price (cents/kWh") +
  ggtitle("Natural Gas Price (cents/kWh)")
```




```
# coerce electricity to a time series object

ts_electricity<-ts(rev(nc_electricity.df[,2]), start=c(2001,1), frequency=12)

plot(ts_electricity, col="red", ylab="$/kWh", xlab="Date", main="NC Residential Electricity Cost")
```




```
#plot both electric & gas TS together
ts.plot(ts_gas_equiv, ts_electricity, gpars = list(col = c("red", "blue")),
        xlab="Date", ylab="cents/kWh",
        main="Comparison of Natural Gas and Electricity Costs")
legend("topleft", bty="n", lty=c(1,1), col=c("red","blue"),
       legend=c(" Natural Gas ", " Electricity "))
```




```r
#Create Dataframe with both Gas & Electricity prices...
#create new date column
new_date<-seq(as.Date("2001/1/1"), by = "month", length.out = nrow(ts_electricity))

#bind date, gas, & electricity
nc_energy.df<-cbind.data.frame("Month_Date"=new_date,
                               "electricity"=nc_electricity.df$price_per_kWh,
                               "gas_equiv"=ts_gas_equiv)

#Subtract Gas price from Electricity price
nc_energy.df$cost_diff<-(nc_energy.df$electricity - nc_energy.df$gas_equiv)
```

add covariates

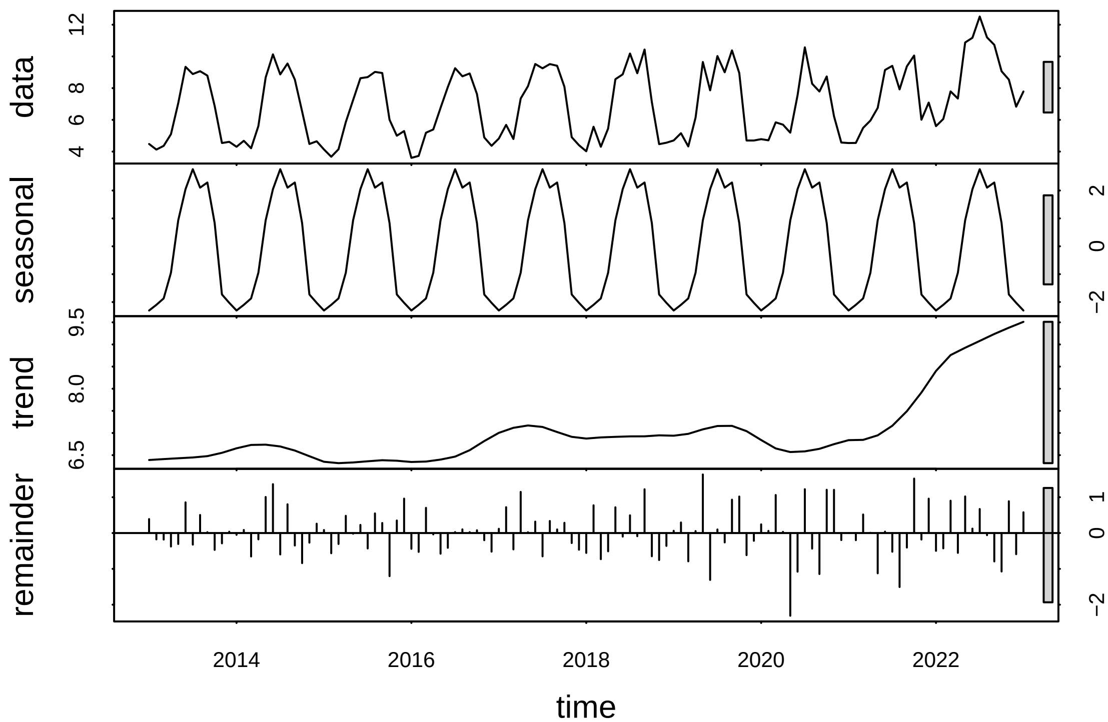
```r
# create an indicator variable for Ukraine War for gas

# create a column with row index
natural_gas.df <- rownames_to_column(natural_gas.df, var = "index")

natural_gas.df$index = as.numeric(natural_gas.df$index)

natural_gas.df$UKRWAR <- ifelse(natural_gas.df$index >= 399, 1, 0)

natural_gas.df <- natural_gas.df[, c(-1, -6)]
```

```r
ts_gas_equiv<-ts(natural_gas.df[,c(3,4)], start=c(1989,1), frequency=12) %>%
  window(start = c(2001, 1))

# create an indicator variable for Ukraine War for electricity
nc_electricity.df <- rownames_to_column(nc_electricity.df, var = "index")

nc_electricity.df$index <- as.numeric(nc_electricity.df$index)

nc_electricity.df$UKRWAR <- ifelse(nc_electricity.df$index >= 255, 1, 0)

nc_electricity.df <- nc_electricity.df[, -1]

# coerce electricity to a time series object

ts_electricity<-ts(rev(nc_electricity.df[,2:3]), start=c(2001,1), frequency=12)


# import temperature data
temperautre <- read_xlsx("./Data/Raleigh Temperature.xlsx") %>%
  gather(key = "Month", value = "temperature", Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, D

temperautre <- temperautre[order(temperautre$Year), ]

ts_temperature <- temperautre[,3] %>%
  na.omit() %>%
  ts(start = c(1995, 1), frequency = 12) %>%
  window(start = c(2001, 1), end = c(2023, 1))

# create a data frame for covariates for electricity (2001-2022)
# we need to do this because electricity and natural gas
# have different fourier terms
fourier_train_e <- fourier(window(ts_electricity[, 2],
                                  end = c(2022, 1)),
                           K = 6)

covariates_train_e <- ts_electricity[, 1] %>%
  window(end = c(2022, 1)) %>%
  cbind(window(ts_temperature, end = c(2022,1)),
        fourier_train_e)


# create covariates data frame for electricity (2001-2023)
fourier_full_e <- fourier(window(ts_electricity[, 2],
                                  end = c(2023, 1)),
                           K = 6)

covariates_full_e <- cbind(ts_electricity[, 1], ts_temperature, fourier_full_e)

# create covariates data frame for natural gas (2001-2022)
fourier_train_gas <- fourier(window(ts_gas_equiv[, 1],
                                  end = c(2022, 1)),
                           K = 6)
```

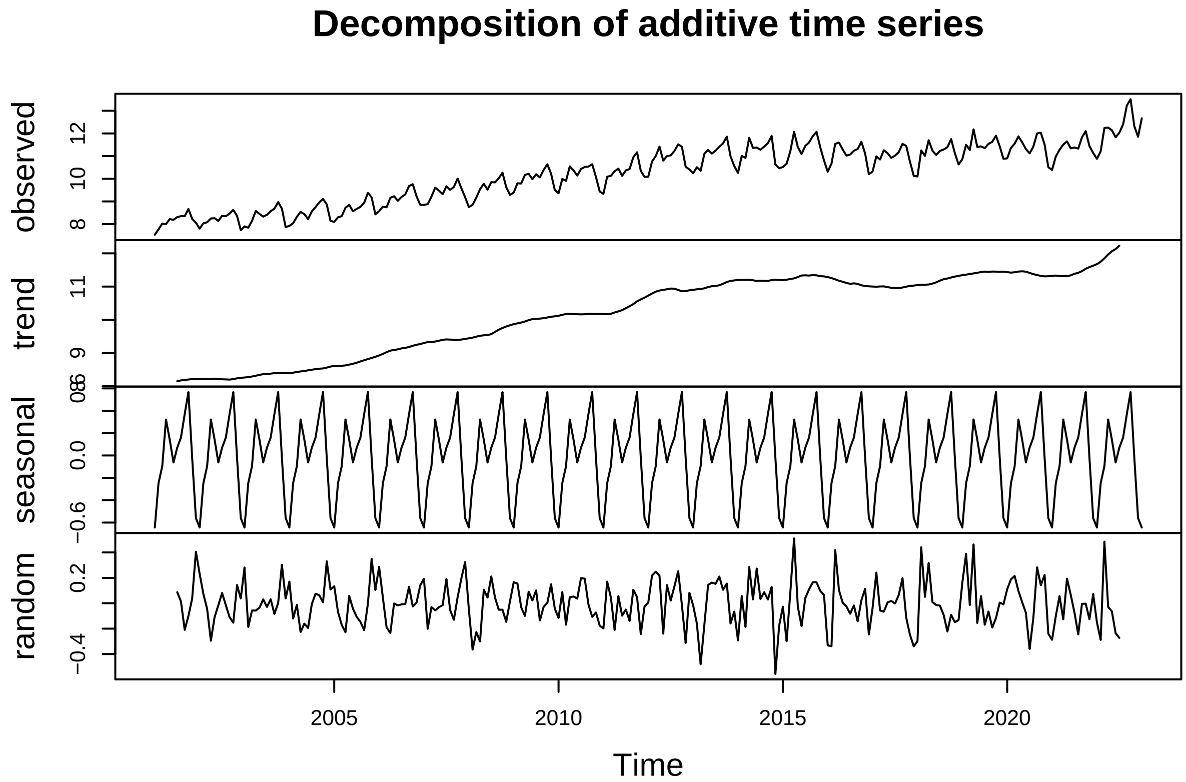
```r
covariates_train_gas <- ts_gas_equiv[, 2] %>%
  window(end = c(2022, 1)) %>%
  cbind(window(ts_temperature, end = c(2022,1)
               ),
        fourier_train_gas)


# create covariates data frame for electricity (2001-2023)
fourier_full_gas <- fourier(window(ts_gas_equiv[, 1],
                                   end = c(2023, 1)),
                            K = 6)

covariates_full_gas <- cbind(ts_gas_equiv[, 2], ts_temperature, fourier_full_gas)
```

Decompose TS

```r
#Decompose TS

decomp_elec<-decompose(ts_electricity[,2], type="additive")
decomp_gas<-decompose(ts_gas_equiv[,1], type="multiplicative")

ts_gas_equiv[,1] %>%
  window(start = c(2013,1), end = c(2023,1)) %>%
  stl(s.window = "periodic") %>%
  plot()
```



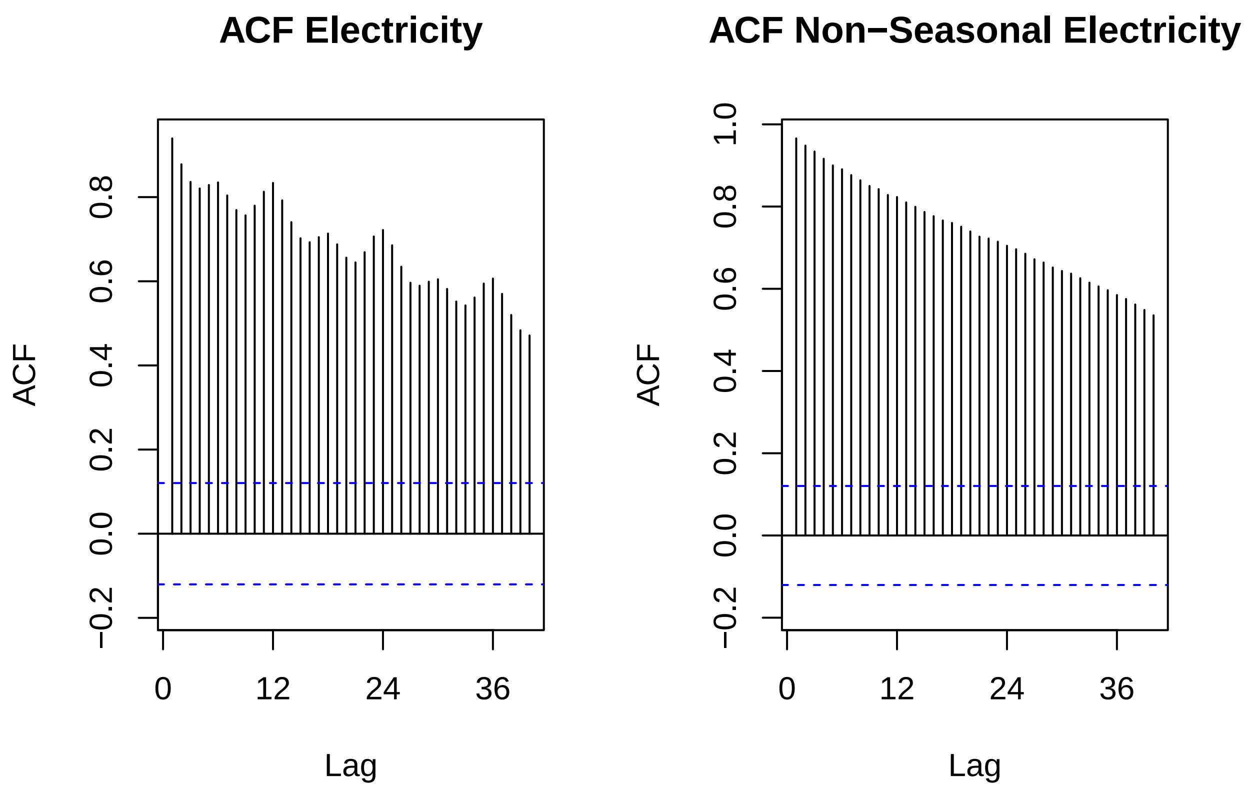

```
plot(decomp_elec)
```



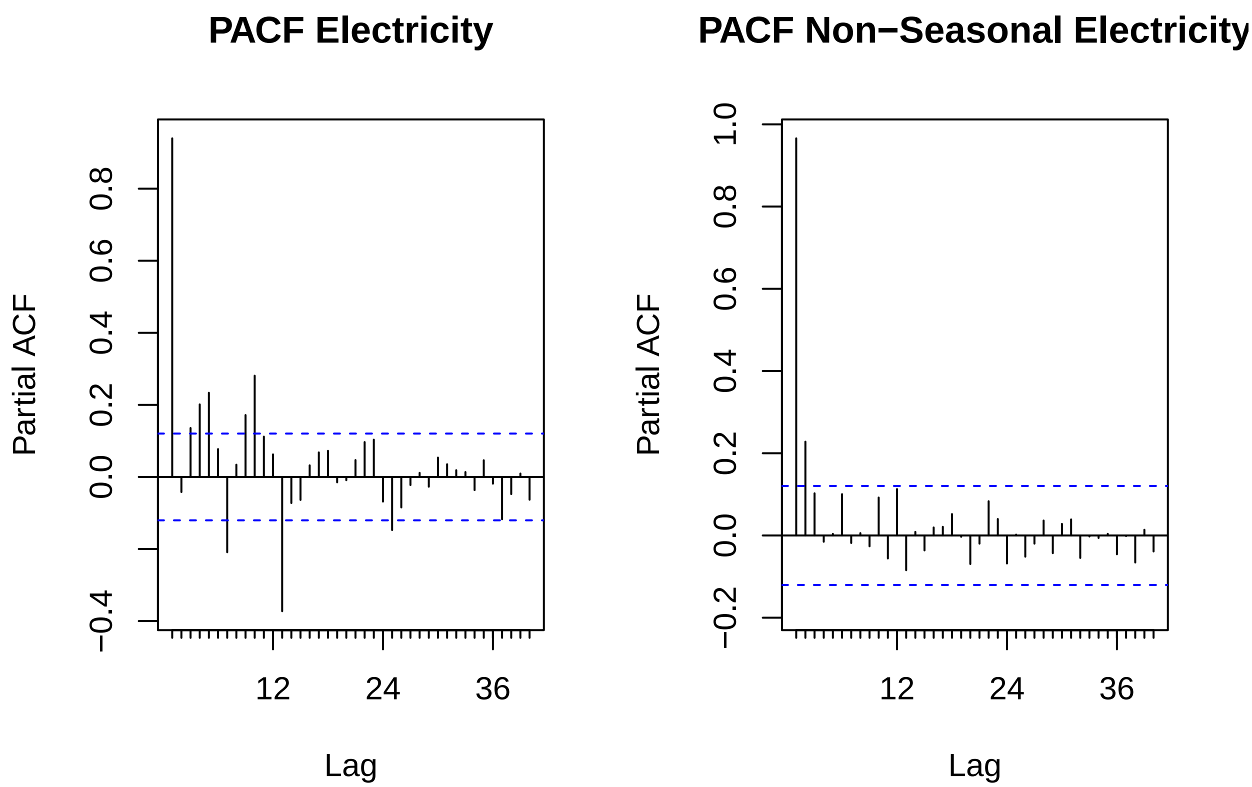

```r
#remove seasonality
ts_elec_ns<-(ts_electricity[,2] - decomp_elec$seasonal)
ts_gas_ns<-(ts_gas_equiv[,1] - decomp_gas$seasonal)
```

plot ACF and PACF to see the general pattern of two series

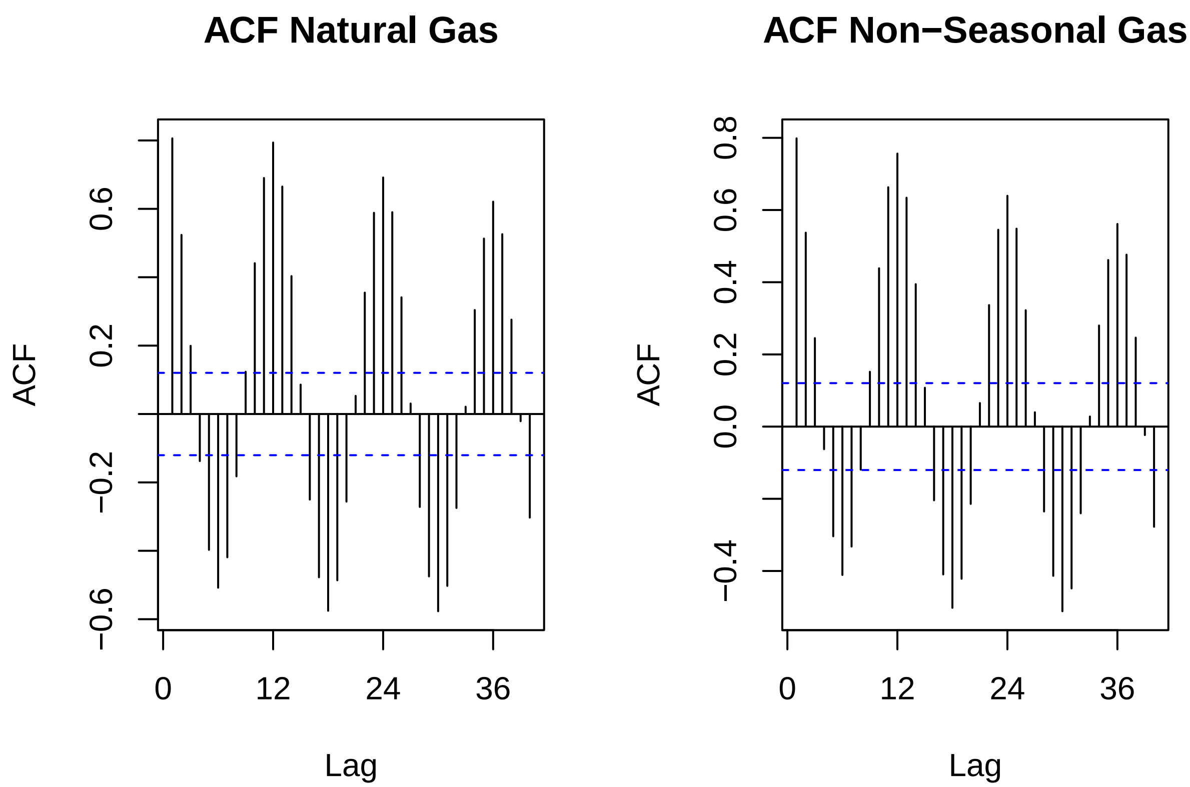
```r
# ACF and PACF of electricity data
par(mfrow=c(1,2))
Acf(ts_electricity[,2], lag.max=40, main="ACF Electricity")
Acf(ts_elec_ns, lag.max=40, main="ACF Non-Seasonal Electricity")
```



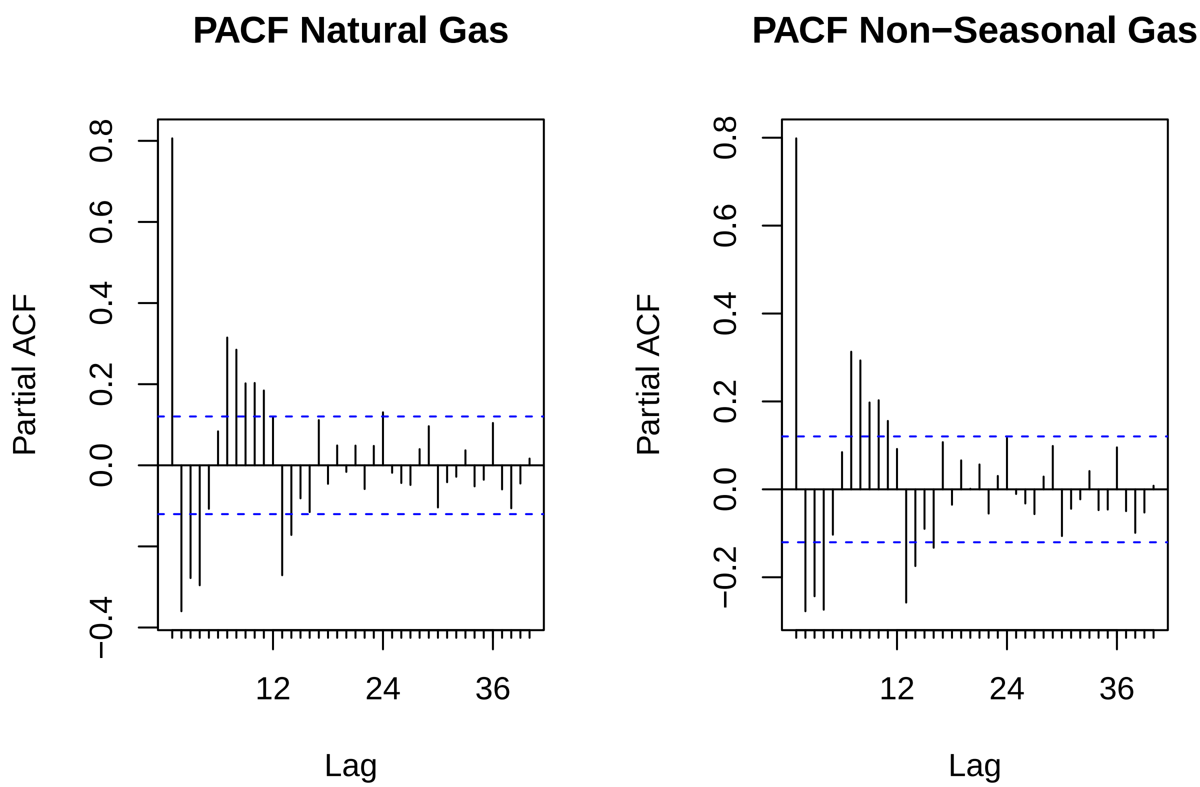

```r
par(mfrow=c(1,2))
Pacf(ts_electricity[,2], lag.max=40, main="PACF Electricity")
Pacf(ts_elec_ns, lag.max=40, main="PACF Non-Seasonal Electricity")
```




```r
# ACF and PACF of natural gas data
par(mfrow=c(1,2))
Acf(ts_gas_equiv[,1], lag.max=40, main="ACF Natural Gas")
Acf(ts_gas_ns, lag.max=40, main="ACF Non-Seasonal Gas")
```

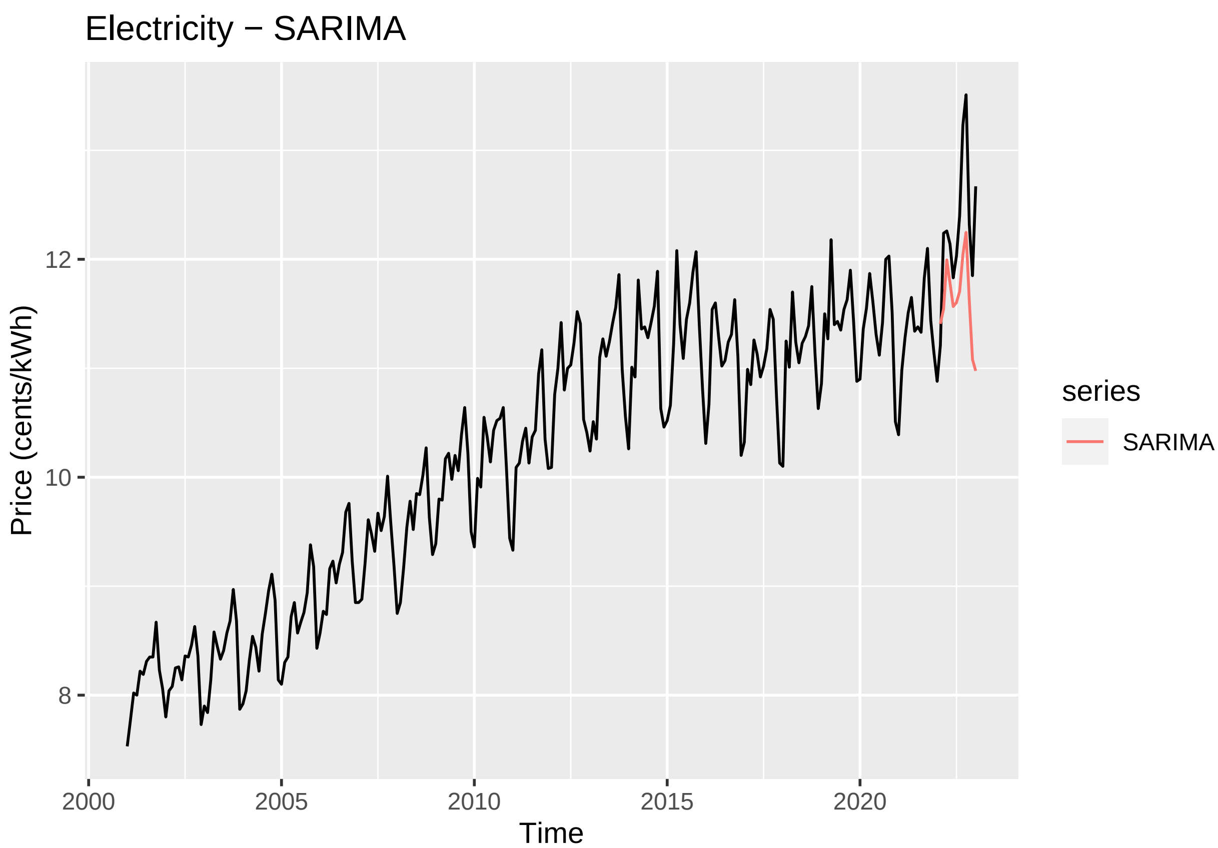



```
par(mfrow=c(1,2))
Pacf(ts_gas_equiv[,1], lag.max=40, main="PACF Natural Gas")
Pacf(ts_gas_ns, lag.max=40, main="PACF Non-Seasonal Gas")
```

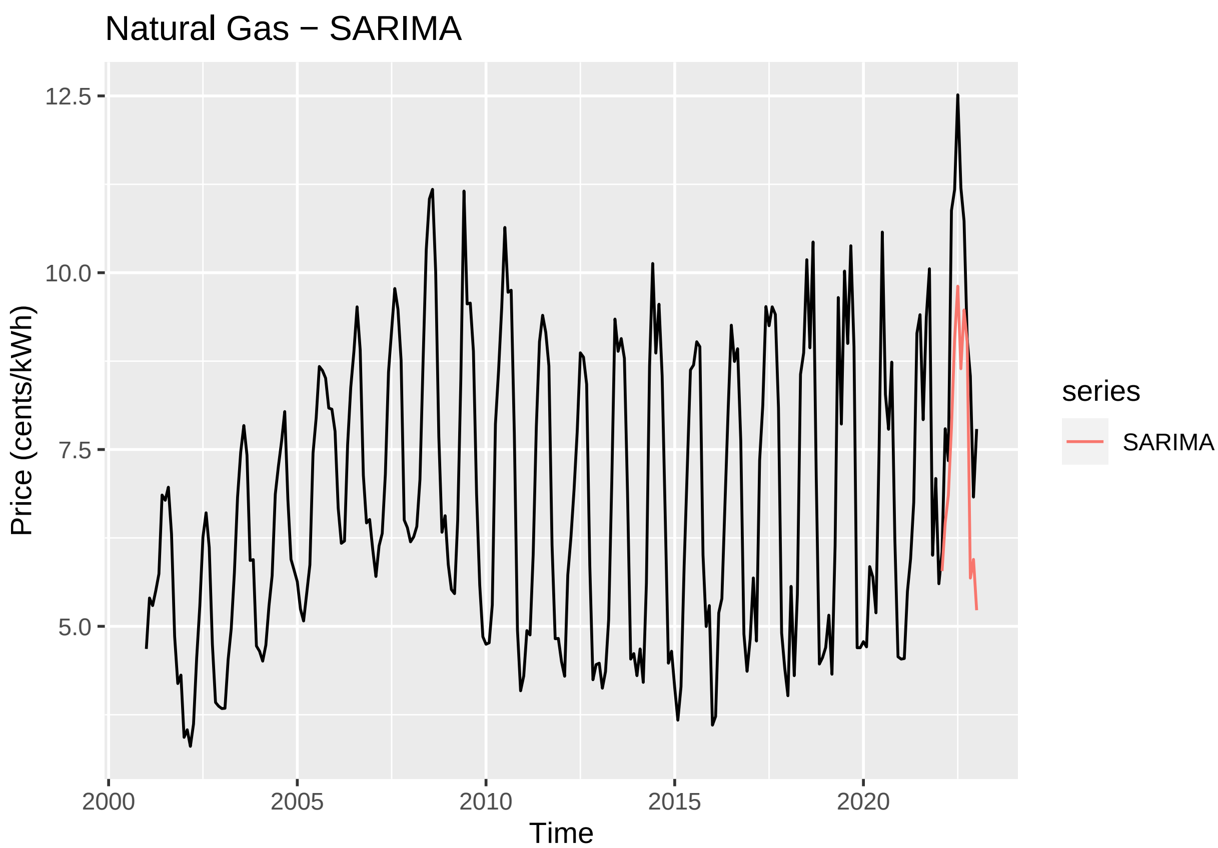


Use seasonal Arima to model eletricity and natural gas

```
#forecast Electricity SARIMA

arima.e.model<-auto.arima(window(ts_electricity[, 2], end=c(2022,1)))
arima.e.forecast<-forecast(arima.e.model, h=12)

autoplot(ts_electricity[, 2]) +
  autolayer(arima.e.forecast$mean, series = "SARIMA") +
  ylab("Price (cents/kWh)") +
  ggtitle("Electricity - SARIMA")
```




```
  # performance is not that good

#Gas SARIMA forecast
arima.gas.model<-auto.arima(window(ts_gas_equiv[, 1], end=c(2022,1)))
arima.gas.forecast<-forecast(arima.gas.model, h=12)

autoplot(ts_gas_equiv[, 1])+
  autolayer(arima.gas.forecast$mean, series = "SARIMA") +
  ylab("Price (cents/kWh)") +
  ggtitle("Natural Gas - SARIMA")
```




```
# performance is not that good
```

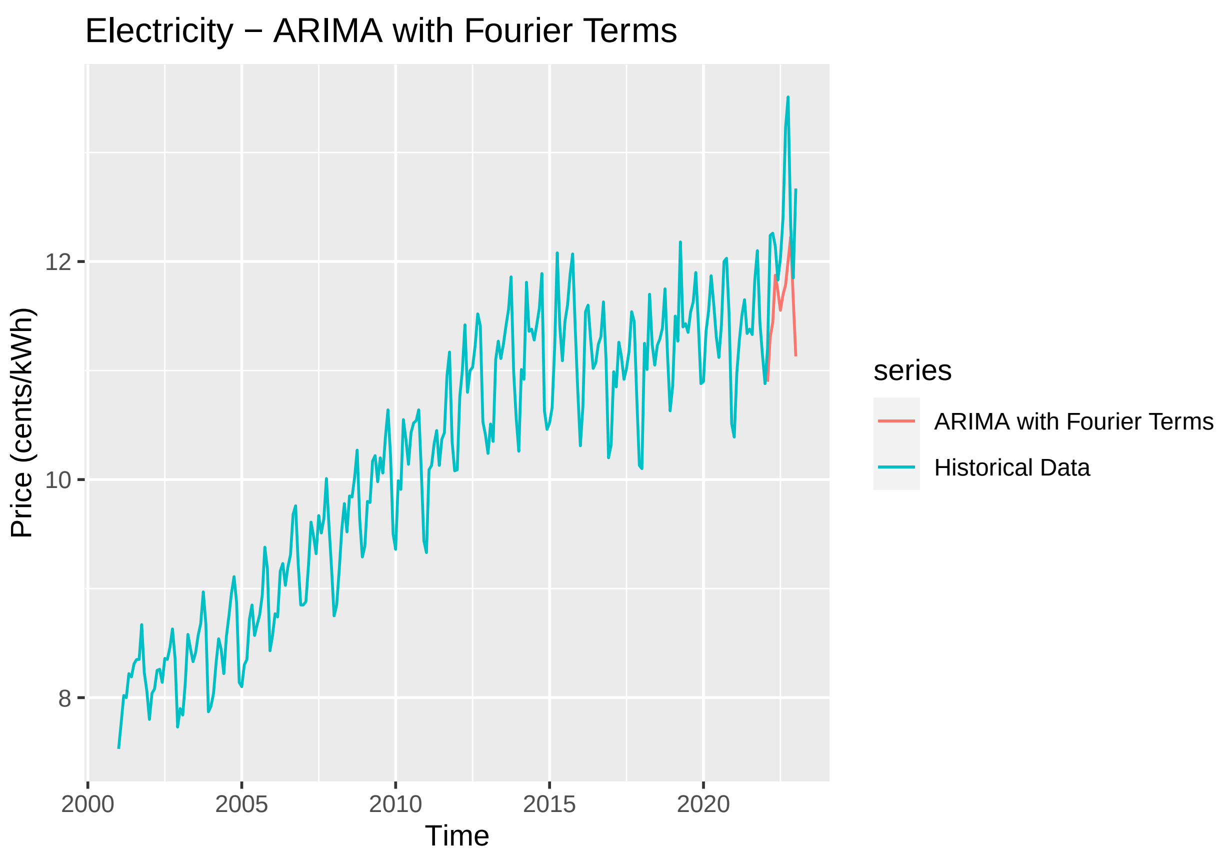
Examine seasonal Arima model's performance on electricity and NG data

```
# model performance for electricity data
sarima_e_perf <- ts_electricity[, 2] %>%
  window(start = c(2022, 2)) %>%
  accuracy(arima.e.forecast$mean)


# model performance for gas data
sarima_gas_perf <- ts_gas_equiv[, 1] %>%
  window(start = c(2022, 2)) %>%
  accuracy(arima.gas.forecast$mean)
```

Use Arima with Fourier terms to model

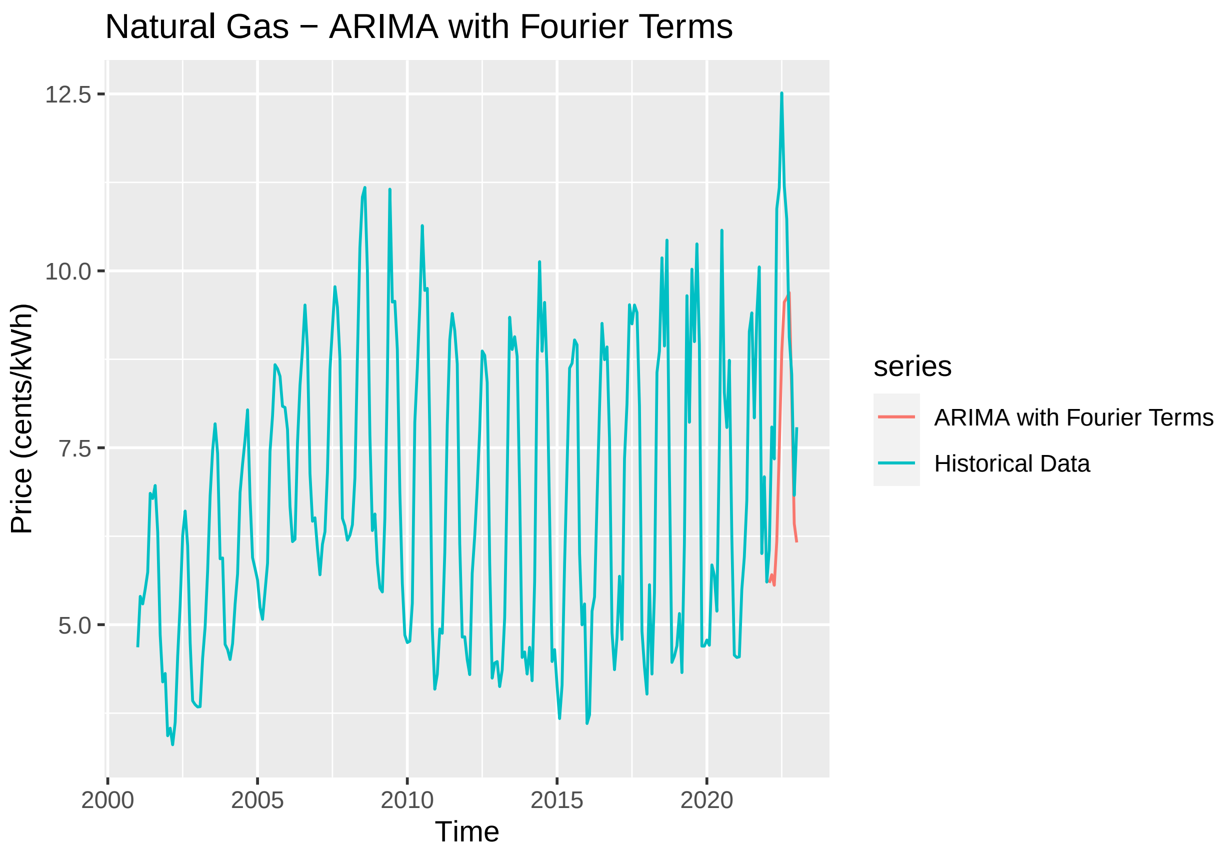
```
# arima with fourier terms for eletricity
arima.e.four.forecast <- ts_electricity[, 2] %>%
  window(end = c(2022, 1)) %>%
  auto.arima(seasonal = FALSE,
             xreg = fourier(window(ts_electricity[, 2], end = c(2022, 1)),
```

```
                           K = 6)) %>%
  forecast(xreg = fourier(window(ts_electricity[, 2],
                                start = c(2022,2)
                                ),
                           K = 6),
           h = 12)

autoplot(arima.e.four.forecast$mean, series = "ARIMA with Fourier Terms") +
  autolayer(ts_electricity[, 2], series = "Historical Data") +
  ggtitle("Electricity - ARIMA with Fourier Terms") +
  ylab("Price (cents/kWh)")
```




```
# arima with fourier terms for gas
arima.gas.four.forecast <- ts_gas_equiv[, 1] %>%
  window(end = c(2022, 1)) %>%
  auto.arima(seasonal = FALSE,
             xreg = fourier(window(ts_gas_equiv[, 1], end = c(2022, 1)),
                           K = 6)
             ) %>%
  forecast(xreg = fourier(window(ts_gas_equiv[, 1],
                                start = c(2022, 2)
                                ),
                          K = 6),
           h = 12)
```

```
autoplot(arima.gas.four.forecast$mean, series = "ARIMA with Fourier Terms") +
  autolayer(ts_gas_equiv[, 1], series = "Historical Data") +
  ggtitle("Natural Gas - ARIMA with Fourier Terms") +
  ylab("Price (cents/kWh)")
```



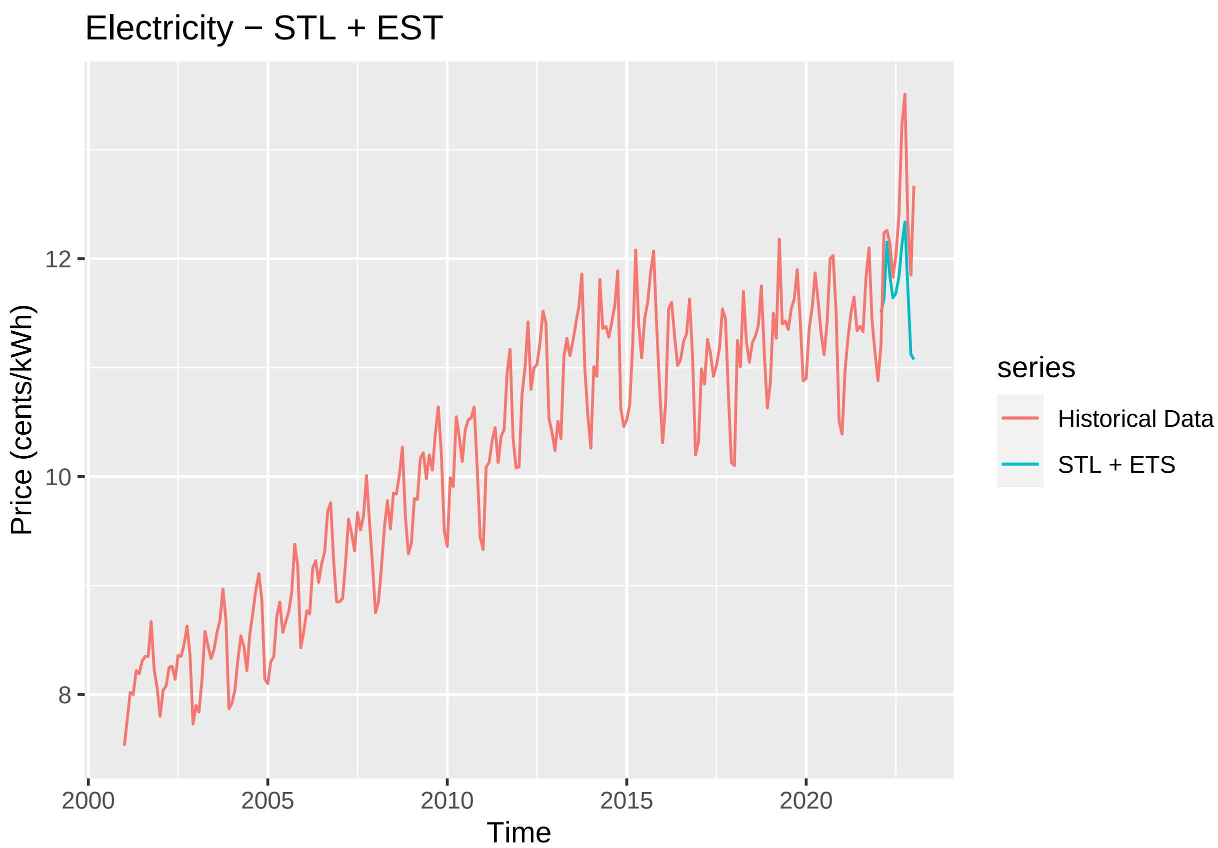
Examine Arima with fourier's performance on electricity and NG data

```
# model performance for electricity data
arima_four_e_perf <- ts_electricity[, 2] %>%
  window(start = c(2022, 2)) %>%
  accuracy(arima.e.four.forecast$mean)

# model performance for gas data
arima_four_gas_perf <- ts_gas_equiv[, 1] %>%
  window(start = c(2022, 2)) %>%
  accuracy(arima.gas.four.forecast$mean)
```

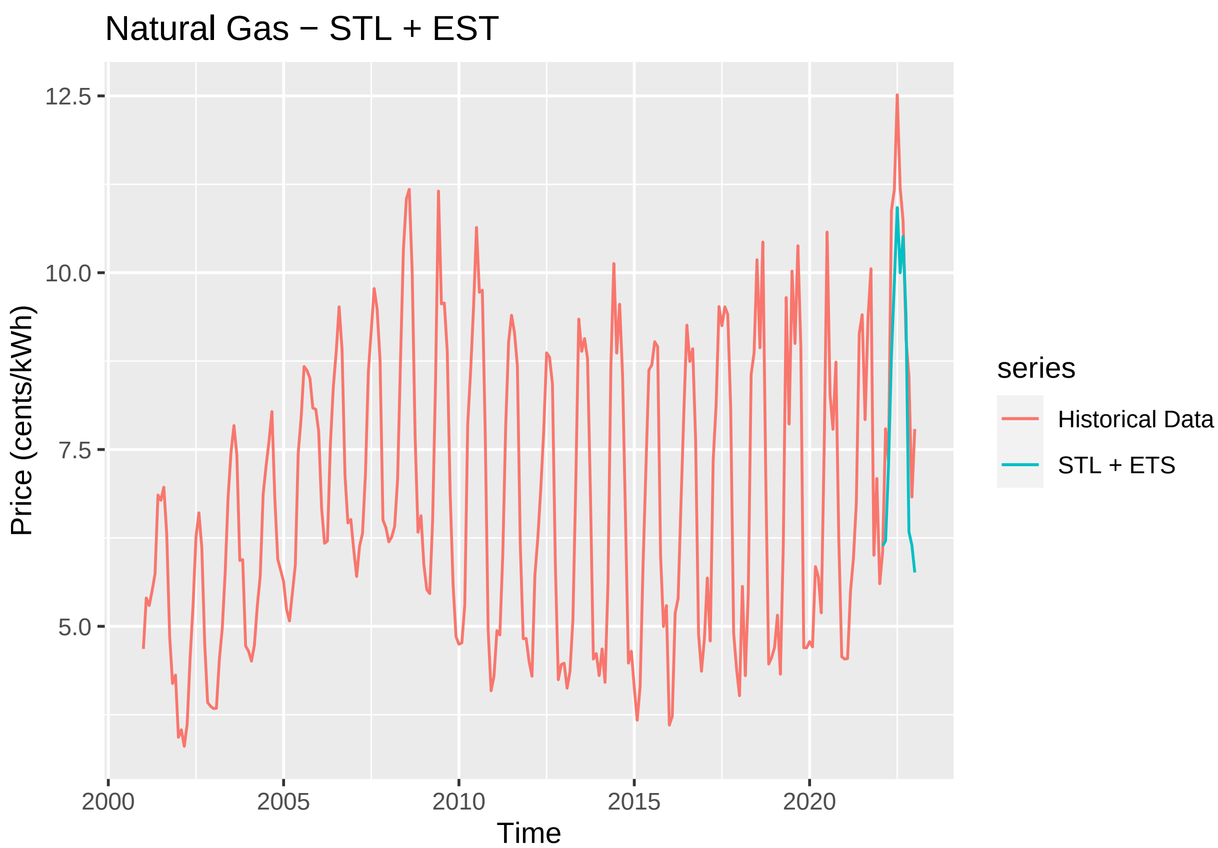
Use STL to model

```
# STL for electricity
stl.e.forecast <- ts_electricity[, 2] %>%
  window(end = c(2022, 1)) %>%
  stlf(h = 12)
```

```
autoplot(stl.e.forecast$mean, series = "STL + ETS") +
  autolayer(ts_electricity[, 2], series = "Historical Data") +
  ggtitle("Electricity - STL + EST") +
  ylab("Price (cents/kWh)")
```




```
# STL for gas
stl.gas.forecast <- ts_gas_equiv[, 1] %>%
  window(end = c(2022, 1)) %>%
  stlf(h = 12)

autoplot(ts_gas_equiv[, 1], series = "Historical Data") +
  autolayer(stl.gas.forecast$mean, series = "STL + ETS") +
  ggtitle("Natural Gas - STL + EST") +
  ylab("Price (cents/kWh)")
```



Examine STL's performance on electricity and NG data

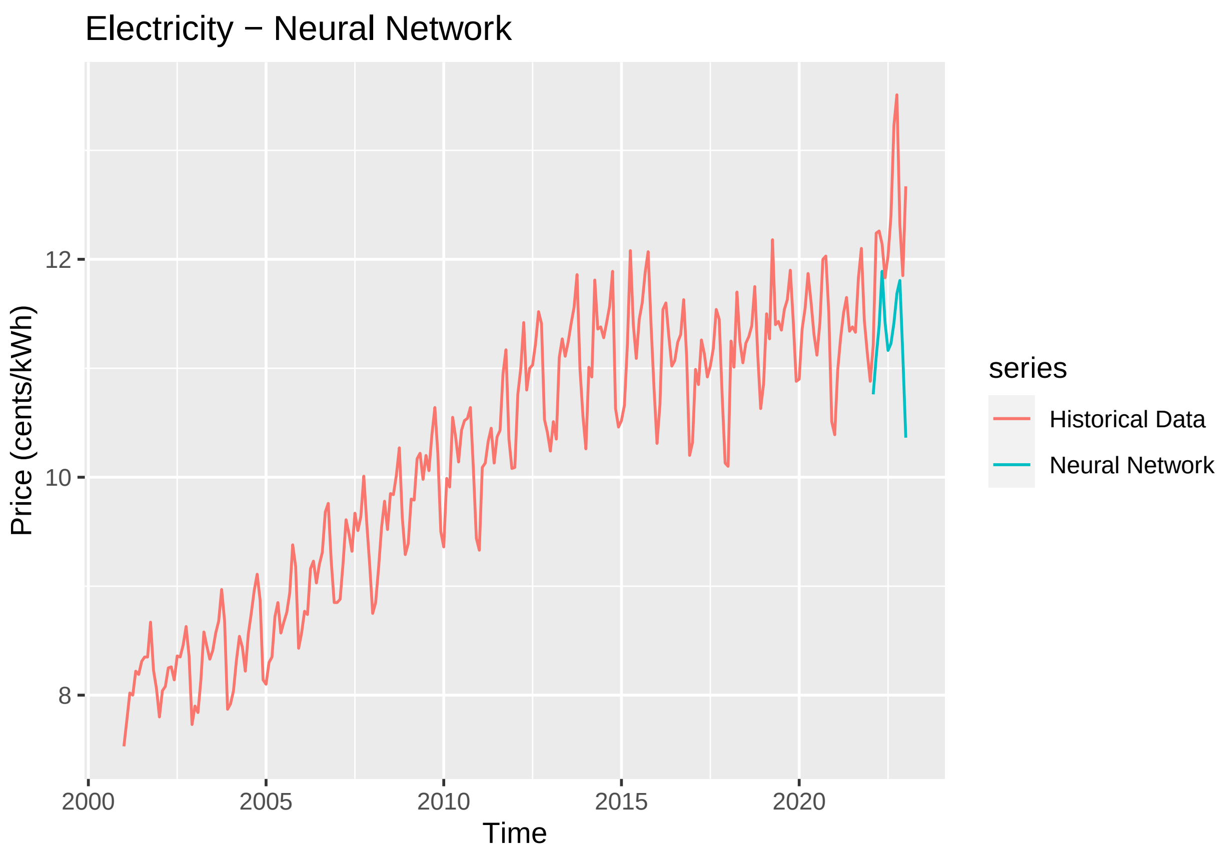
```
# model performance for electricity data
stl_e_perf <- ts_electricity[, 2] %>%
  window(start = c(2022, 2)) %>%
  accuracy(stl.e.forecast$mean)

# model performance for gas data
stl_gas_perf <- ts_gas_equiv[, 1] %>%
  window(start = c(2022, 2)) %>%
  accuracy(stl.gas.forecast$mean)
```

Use ETS to model – not finished – I don't know how to do this

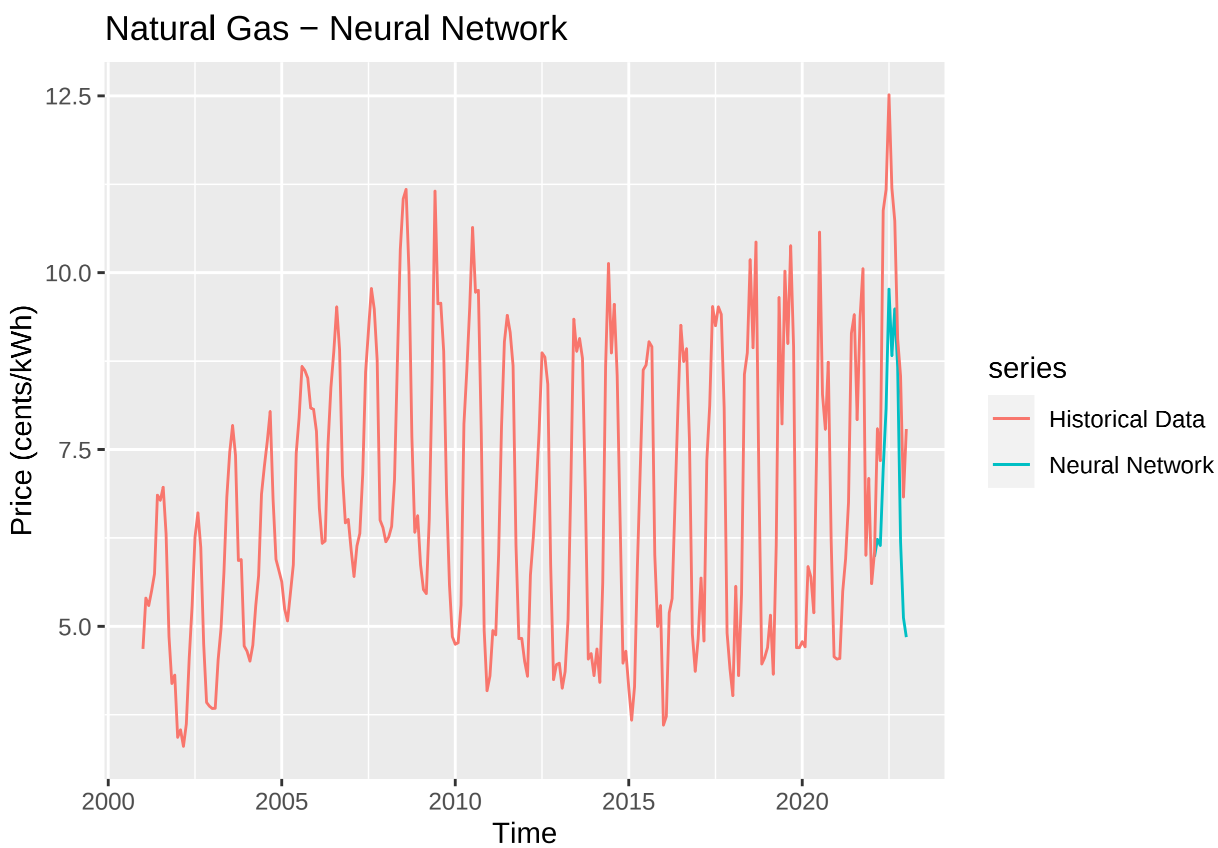
```
# I think STL doesn't need fourier terms. So, I will get covariates dataframes without fourier terms
# because UKRWAR and temperature are the same across two data frames, we only need one
cov_train_nofour <- covariates_train_e[, 1:2]
colnames(cov_train_nofour) <- c("UKRWAR", "temperature")

cov_test_nofour <- covariates_full_e[, 1:2] %>%
  window(start = c(2022, 2))
colnames(cov_test_nofour) <- c("UKRWAR", "temperature")
```

Use Neural Network and fourier to model

```
# neural network forecast for electricity data
nn.e.forecast <- ts_electricity[, 2] %>%
  window(end = c(2022 ,1)) %>%
  nnetar(p = 2, P = 1, # 2 and 1 are randomly decided. To find the optimal one,
         # we need to run a couple of combinations to find the best one
         xreg = fourier(window(ts_electricity[, 2],
                               end = c(2022, 1)
                               ),
                        K = 6)
         ) %>%
  forecast(xreg = fourier(window(ts_electricity[, 2],
                                 start = c(2022, 2)
                                 ),
                          K = 6),
           h = 12)

# neural network forecast for gas data
nn.gas.forecast <- ts_gas_equiv[, 1] %>%
  window(end = c(2022 ,1)) %>%
  nnetar(p = 2, P = 1,
         xreg = fourier(window(ts_gas_equiv[, 1],
                               end = c(2022, 1)
                               ),
                        K = 6)
         ) %>%
  forecast(xreg = fourier(window(ts_gas_equiv[, 1],
                                 start = c(2022, 2)
                                 ),
                          K = 6),
           h = 12)


autoplot(nn.e.forecast$mean, series = "Neural Network") +
  autolayer(ts_electricity[, 2], series = "Historical Data") +
  ggtitle("Electricity - Neural Network") +
  ylab("Price (cents/kWh)")
```




```
autoplot(nn.gas.forecast$mean, series = "Neural Network") +
  autolayer(ts_gas_equiv[, 1], series = "Historical Data") +
  ggtitle("Natural Gas - Neural Network") +
  ylab("Price (cents/kWh)")
```



neutral network model performance

```
# neural network model performance for electricity data
nn_e_perf <- ts_electricity[, 2] %>%
  window(start = c(2022, 2)) %>%
  accuracy(nn.e.forecast$mean)

# neural network model performance for gas data
nn_gas_perf <- ts_gas_equiv[, 1] %>%
  window(start = c(2022, 2)) %>%
  accuracy(nn.gas.forecast$mean)
```

find the optimal p and P values for neural network without fourier

```
## p = 1, P = 0
nn.e.forecast.10 <- ts_electricity[, 2] %>%
  window(end = c(2022 ,1)) %>%
  nnetar(p = 1, P = 0, # 2 and 1 are randomly decided. To find the optimal one,
         # we need to run a couple of combinations to find the best one
         ) %>%
  forecast(h = 12)
```

```r
nn.e.10.score <- ts_electricity[, 2] %>%
  window(start = c(2022, 2)) %>%
  accuracy(nn.e.forecast.10$mean)

## p = 1, P = 1
nn.e.forecast.11 <- ts_electricity[, 2] %>%
  window(end = c(2022 ,1)) %>%
  nnetar(p = 1, P = 1, # 2 and 1 are randomly decided. To find the optimal one,
         # we need to run a couple of combinations to find the best one
         ) %>%
  forecast(h = 12)

nn.e.11.score <- ts_electricity[, 2] %>%
  window(start = c(2022, 2)) %>%
  accuracy(nn.e.forecast.11$mean)

## p = 2, P = 0
nn.e.forecast.20 <- ts_electricity[, 2] %>%
  window(end = c(2022 ,1)) %>%
  nnetar(p = 2, P = 0, # 2 and 1 are randomly decided. To find the optimal one,
         # we need to run a couple of combinations to find the best one
         ) %>%
  forecast(h = 12)

nn.e.20.score <- ts_electricity[, 2] %>%
  window(start = c(2022, 2)) %>%
  accuracy(nn.e.forecast.20$mean)

## p = 2, P = 1
nn.e.forecast.21 <- ts_electricity[, 2] %>%
  window(end = c(2022 ,1)) %>%
  nnetar(p = 2, P = 1, # 2 and 1 are randomly decided. To find the optimal one,
         # we need to run a couple of combinations to find the best one
         ) %>%
  forecast(h = 12)

nn.e.21.score <- ts_electricity[, 2] %>%
  window(start = c(2022, 2)) %>%
  accuracy(nn.e.forecast.21$mean)

## p = 2, P = 2
nn.e.forecast.22 <- ts_electricity[, 2] %>%
  window(end = c(2022 ,1)) %>%
  nnetar(p = 2, P = 2, # 2 and 1 are randomly decided. To find the optimal one,
         # we need to run a couple of combinations to find the best one
         ) %>%
  forecast(h = 12)

nn.e.22.score <- ts_electricity[, 2] %>%
  window(start = c(2022, 2)) %>%
  accuracy(nn.e.forecast.22$mean)

## p = 1, P = 2
```

```r
nn.e.forecast.12 <- ts_electricity[, 2] %>%
  window(end = c(2022 ,1)) %>%
  nnetar(p = 1, P = 2, # 2 and 1 are randomly decided. To find the optimal one,
         # we need to run a couple of combinations to find the best one
         ) %>%
  forecast(h = 12)

nn.e.12.score <- ts_electricity[, 2] %>%
  window(start = c(2022, 2)) %>%
  accuracy(nn.e.forecast.12$mean)

## p = 3, P = 1
nn.e.forecast.31 <- ts_electricity[, 2] %>%
  window(end = c(2022 ,1)) %>%
  nnetar(p = 3, P = 1, # 2 and 1 are randomly decided. To find the optimal one,
         # we need to run a couple of combinations to find the best one
         ) %>%
  forecast(h = 12)

nn.e.31.score <- ts_electricity[, 2] %>%
  window(start = c(2022, 2)) %>%
  accuracy(nn.e.forecast.31$mean)

nn.scores <- rbind(nn.e.10.score, nn.e.11.score, nn.e.20.score, nn.e.21.score,
                   nn.e.22.score, nn.e.12.score, nn.e.31.score)

row.names(nn.scores) <- c("10", "11", "20", "21", "22", "12", "31")

nn.scores ## 11 has the lowest RMSE and 12 has the lowest MAPE
```

```
##             ME      RMSE       MAE        MPE      MAPE      ACF1 Theil's U
## 10 -1.1711449 1.2950457 1.1711449 -10.501094 10.501094 0.2845564 45.321970
## 11 -0.8741572 0.9866865 0.8741572  -7.618237  7.618237 0.1874326  4.726117
## 20 -1.0281403 1.1590371 1.0281403  -9.096220  9.096220 0.2979656 17.979926
## 21 -0.8828677 0.9963855 0.8828677  -7.702064  7.702064 0.2032835  4.741193
## 22 -0.8770749 1.0019129 0.8770749  -7.659080  7.659080 0.2181894  4.555208
## 12 -0.8706740 0.9905462 0.8706740  -7.593170  7.593170 0.1962725  4.550421
## 31 -0.8860134 1.0037685 0.8860134  -7.735259  7.735259 0.2230761  4.774109
```

```r
# we can do the same thing to natural gas. Do we want to??
```

Use Neural Network, temperature, UKRWAR, and fourier to model

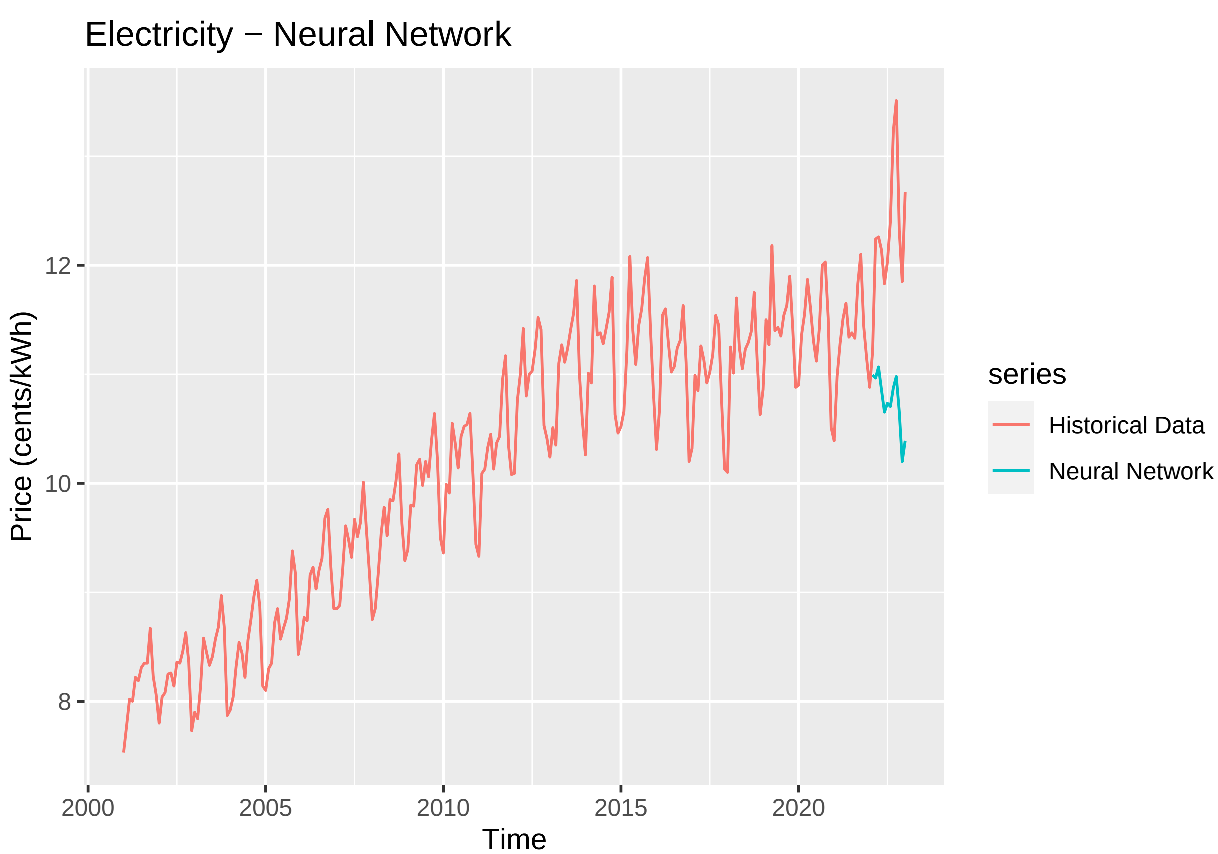
```r
# neural network forecast for electricity data
nn.cov.e.forecast <- ts_electricity[, 2] %>%
  window(end = c(2022 ,1)) %>%
  nnetar(p = 2, P = 1,
         xreg = covariates_train_e) %>%
  forecast(xreg = window(covariates_full_e, start = c(2022, 2)),
           h = 12)
```

```
## Warning in nnetar(., p = 2, P = 1, xreg = covariates_train_e): Constant xreg
## column, setting scale.inputs=FALSE

## Warning in forecast.nnetar(., xreg = window(covariates_full_e, start = c(2022, :
## xreg contains different column names from the xreg used in training. Please
## check that the regressors are in the same order.
```

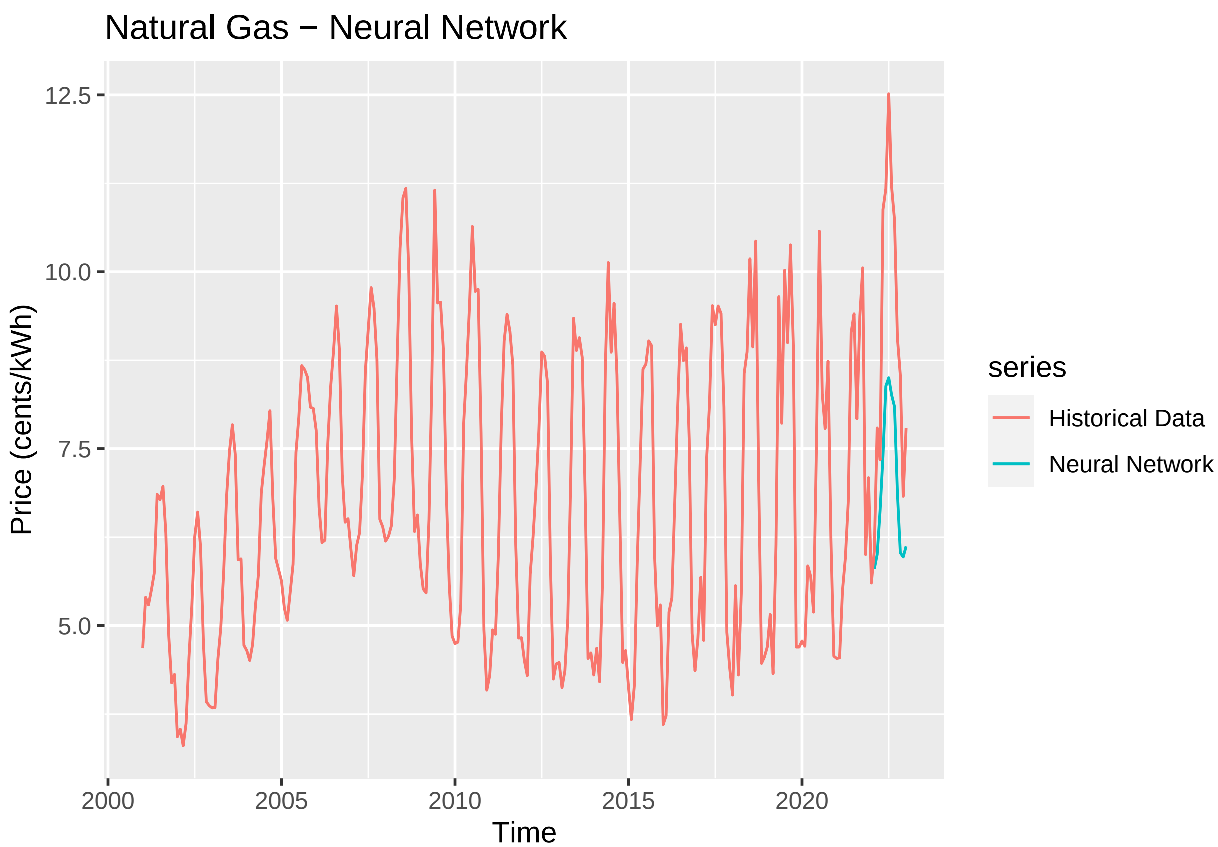
```
# neural network forecast for gas data
nn.cov.gas.forecast <- ts_gas_equiv[, 1] %>%
  window(end = c(2022 ,1)) %>%
  nnetar(p = 2, P = 1,
         xreg = covariates_train_gas) %>%
  forecast(xreg = window(covariates_full_gas, start = c(2022, 2)),
           h = 12)
```

```
## Warning in nnetar(., p = 2, P = 1, xreg = covariates_train_gas): Constant xreg
## column, setting scale.inputs=FALSE

## Warning in forecast.nnetar(., xreg = window(covariates_full_gas, start =
## c(2022, : xreg contains different column names from the xreg used in training.
## Please check that the regressors are in the same order.
```

```
autoplot(nn.cov.e.forecast$mean, series = "Neural Network") +
  autolayer(ts_electricity[, 2], series = "Historical Data") +
  ggtitle("Electricity - Neural Network") +
  ylab("Price (cents/kWh)")
```



```
autoplot(nn.cov.gas.forecast$mean, series = "Neural Network") +
  autolayer(ts_gas_equiv[, 1], series = "Historical Data") +
  ggtitle("Natural Gas - Neural Network") +
  ylab("Price (cents/kWh)")
```



neutral network model, temperature, fourier, UKRWAR performance

```
# neural network model performance for electricity data
nn_cov_e_perf <- ts_electricity[, 2] %>%
  window(start = c(2022, 2)) %>%
  accuracy(nn.cov.e.forecast$mean)

# neural network model performance for gas data
nn_cov_gas_perf <- ts_gas_equiv[, 1] %>%
  window(start = c(2022, 2)) %>%
  accuracy(nn.cov.gas.forecast$mean)
```

use TBATS model

```
# TBATS for electricity
tbats.e.forecast <- ts_electricity[, 2] %>%
```

```
  window(end = c(2022, 1)) %>%
  tbats() %>%
  forecast(h = 12)

# TBATS for natural gas
tbats.gas.forecast <- ts_gas_equiv[, 1] %>%
  window(end = c(2022, 1)) %>%
  tbats() %>%
  forecast(h = 12)
```

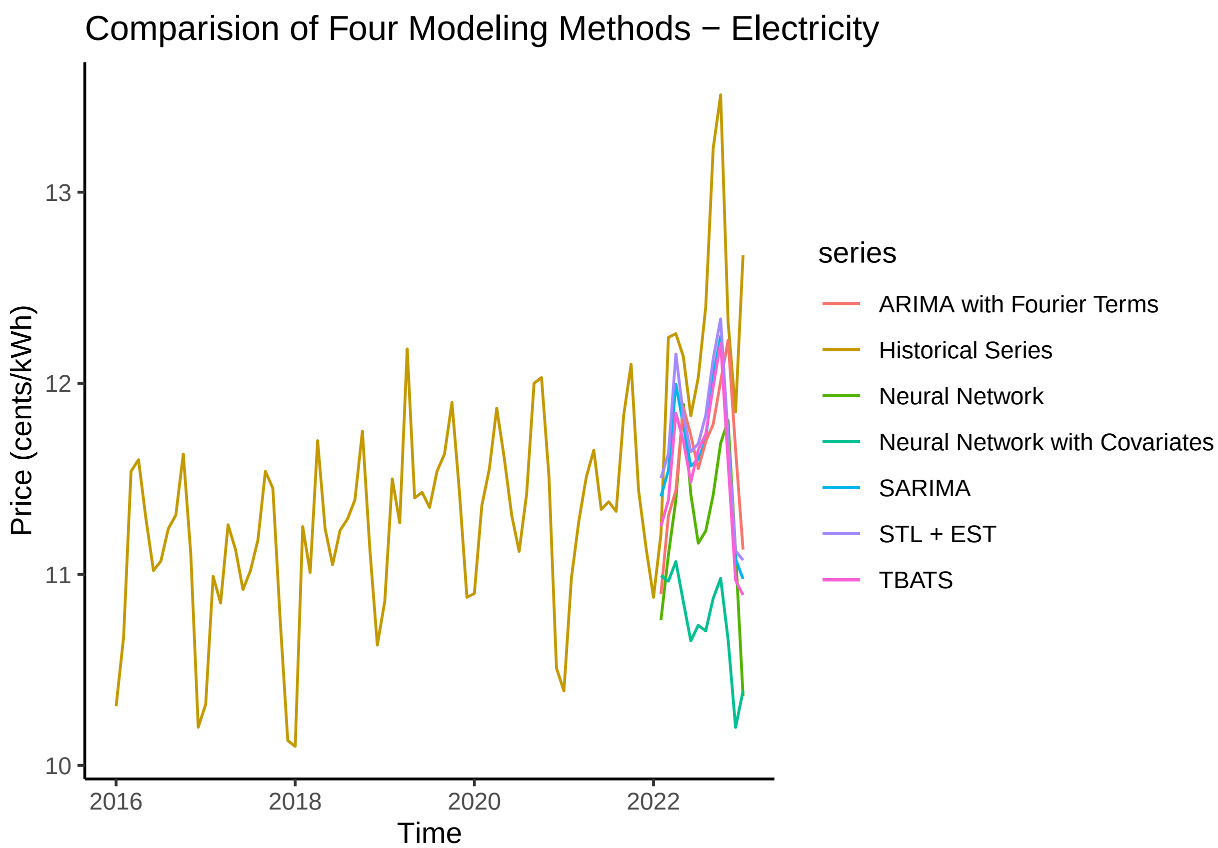
```
## Warning in bats(as.numeric(y), use.box.cox = use.box.cox, use.trend =
## use.trend, : optim() did not converge.
```

TBATS model performance

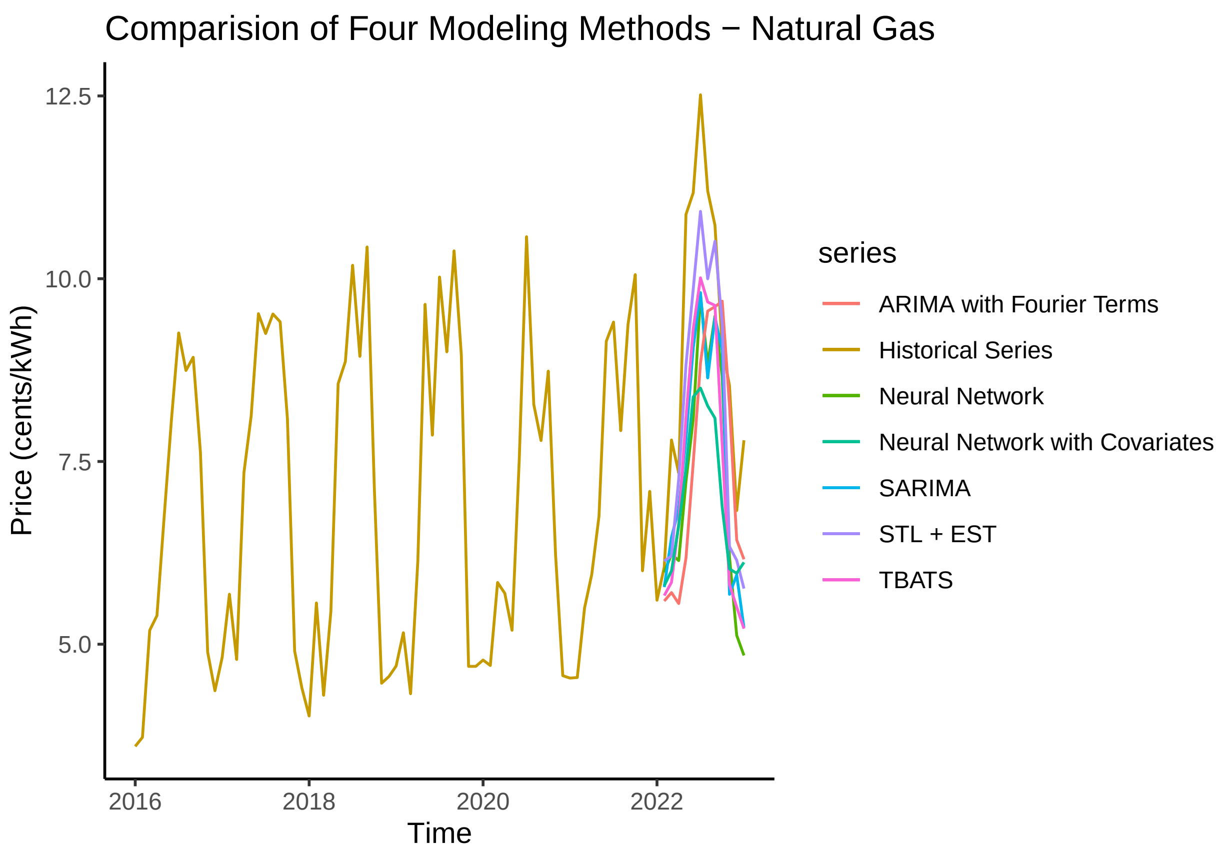
```
# neural network model performance for electricity data
tbats_e_perf <- ts_electricity[, 2] %>%
  window(start = c(2022, 2)) %>%
  accuracy(tbats.e.forecast$mean)

# neural network model performance for gas data
tbats_gas_perf <- ts_gas_equiv[, 1] %>%
  window(start = c(2022, 2)) %>%
  accuracy(tbats.gas.forecast$mean)
```

compare performance scores and generate tables for use

```
# plot these together
ts_electricity[, 2] %>%
  window(start = c(2016, 1)) %>%
  autoplot(series = "Historical Series") +
  autolayer(nn.e.forecast$mean, series = "Neural Network") +
  autolayer(arima.e.forecast$mean, series = "SARIMA") +
  autolayer(arima.e.four.forecast$mean, series = "ARIMA with Fourier Terms") +
  autolayer(stl.e.forecast$mean, series = "STL + EST") +
  autolayer(tbats.e.forecast$mean, series = "TBATS") +
  autolayer(nn.cov.e.forecast$mean, series = "Neural Network with Covariates") +
  ylab("Price (cents/kWh)") +
  ggtitle("Comparision of Four Modeling Methods - Electricity") +
  theme_classic()
```




```
ts_gas_equiv[, 1] %>%
  window(start = c(2016, 1)) %>%
  autoplot(series = "Historical Series") +
  autolayer(nn.gas.forecast$mean, series = "Neural Network") +
  autolayer(arima.gas.forecast$mean, series = "SARIMA") +
  autolayer(arima.gas.four.forecast$mean, series = "ARIMA with Fourier Terms") +
  autolayer(stl.gas.forecast$mean, series = "STL + EST") +
  autolayer(tbats.gas.forecast$mean, series = "TBATS") +
  autolayer(nn.cov.gas.forecast$mean,series = "Neural Network with Covariates") +
  ylab("Price (cents/kWh)") +
  ggtitle("Comparision of Four Modeling Methods - Natural Gas") +
  theme_classic()
```




```r
# scores for electricity
scores.e <- rbind(sarima_e_perf,
                  arima_four_e_perf,
                  stl_e_perf,
                  nn_e_perf,
                  tbats_e_perf,
                  nn_cov_e_perf
                  ) %>%
  as.data.frame()

  # rename rows
row.names(scores.e) <- c("SARIMA",
                         "ARIMA with Fourier",
                         "STL",
                         "Neural Network",
                         "TBATS",
                         "Neural Network with Covariates")

  # find the row index of the lowest RMSE
best.e.model <- scores.e$RMSE %>%
  which.min()

cat("The best model for electricity by RMSE is: ",
    row.names(scores.e[best.e.model, ]))
```

```
## The best model for electricity by RMSE is:  STL
```

```r
# generate a visualized table to use in the report
kbl(scores.e,
    caption = "Forecast Accuracy for NC Residential Electricity Price",
    digits = array(5, ncol(scores.e))) %>%
  kable_styling(full_width = FALSE, position = "center",
                latex_options = "hold_position") %>%
  kable_styling(latex_options = "striped",
                stripe_index = best.e.model,
                stripe_color = "red")
```

Table 1: Forecast Accuracy for NC Residential Electricity Price

	ME	RMSE	MAE	MPE	MAPE	ACF1	Theil's U
SARIMA	-0.67702	0.83764	0.70998	-5.83726	6.12612	0.25988	2.73983
ARIMA with Fourier	-0.69901	0.87542	0.69901	-6.04750	6.04750	0.08222	2.82856
STL	-0.58551	0.76708	0.63447	-5.01390	5.43949	0.25835	2.36056
Neural Network	-1.02893	1.20238	1.02893	-9.22150	9.22150	0.15129	3.04762
TBATS	-0.74971	0.89063	0.75658	-6.50522	6.56626	0.23462	2.86546
Neural Network with Covariates	-1.55135	1.66639	1.55135	-14.46686	14.46686	0.40902	8.22301

```r
# scores for gas
scores.gas <- rbind(sarima_gas_perf,
                  arima_four_gas_perf,
                  stl_gas_perf,
                  nn_gas_perf,
                  tbats_gas_perf,
                  nn_cov_gas_perf) %>%
  as.data.frame()

  # rename rows
row.names(scores.gas) <- c("SARIMA",
                         "ARIMA with Fourier",
                         "STL",
                         "Neural Network",
                         "TBATS",
                         "Neural Network with Covariates")

  # find the row index of the lowest RMSE
best.gas.model <- scores.gas$RMSE %>%
  which.min()

cat("The best model for natural gas by RMSE is: ",
    row.names(scores.gas[best.gas.model, ]))
```

```
## The best model for natural gas by RMSE is:  STL
```

```r
# generate a visualized table to use in the report
kbl(scores.gas,
    caption = "Forecast Accuracy for NC Residential Natural Gas Price",
    digits = array(5, ncol(scores.gas))) %>%
  kable_styling(full_width = FALSE, position = "center",
```

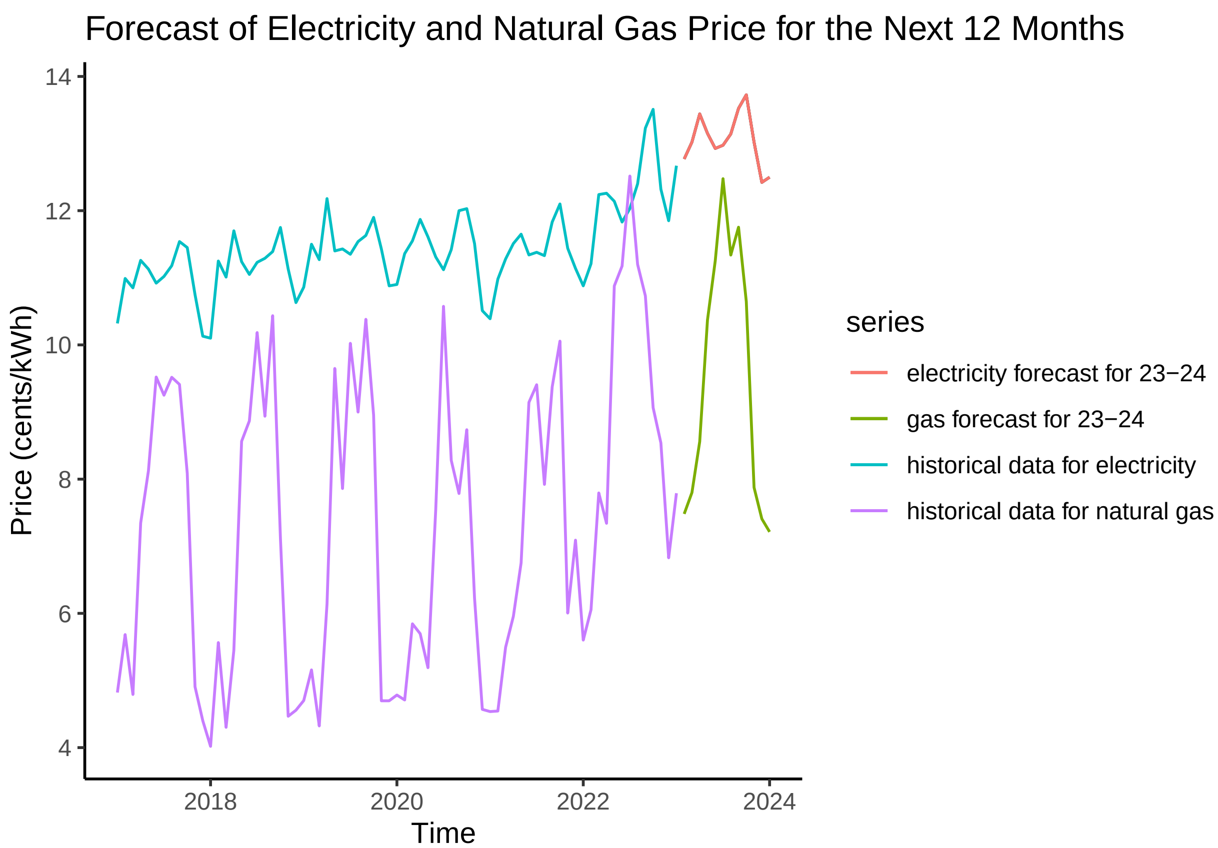
```
                latex_options = "hold_position") %>%
  kable_styling(latex_options = "striped",
                stripe_index = best.gas.model,
                stripe_color = "red")
```

Table 2: Forecast Accuracy for NC Residential Natural Gas Price

	ME	RMSE	MAE	MPE	MAPE	ACF1	Theil's U
SARIMA	-1.67902	1.97784	1.67902	-23.35174	23.35174	-0.15535	1.81675
ARIMA with Fourier	-1.73517	2.31199	1.83980	-25.17030	26.24979	0.57850	3.12540
STL	-1.04788	1.35599	1.11314	-14.03428	14.80904	-0.33348	1.28849
Neural Network	-1.94478	2.21270	1.94478	-28.26876	28.26876	0.20893	2.40962
TBATS	-1.72858	1.89245	1.72858	-24.53539	24.53539	-0.32231	2.08610
Neural Network with Covariates	-2.15401	2.41525	2.15401	-29.65107	29.65107	0.23588	3.92740

Use STL to model electricity and natural gas for the next 12 month

```
e.forecast.2324 <- ts_electricity[, 2] %>%
  stlf(h = 12)

gas.forecast.2324 <- ts_gas_equiv[, 1] %>%
  stlf(h = 12)

e.forecast.2324$mean %>%
  autoplot() +
  autolayer(e.forecast.2324$mean, series = "electricity forecast for 23-24") +
  autolayer(gas.forecast.2324$mean, series = "gas forecast for 23-24") +
  autolayer(window(ts_electricity[, 2], start = c(2017, 1)),
          series = "historical data for electricity") +
   autolayer(window(ts_gas_equiv[, 1], start = c(2017, 1)),
          series = "historical data for natural gas") +
  theme_classic() +
  ggtitle("Forecast of Electricity and Natural Gas Price for the Next 12 Months") +
  ylab("Price (cents/kWh)")
```


Forecast of Electricity and Natural Gas Price for the Next 12 Months
Price (cents/kWh)
14
12
10
8
6
4
2018
2020
2022
2024
Time
series
electricity forecast for 23−24
gas forecast for 23−24
historical data for electricity
historical data for natural gas