Code Sample for World Bank STC Position

Eshan Prashar

2024-03-31

Goal: Forecast monthly residential electricity consumption in CA. This file has 4 broad sections. First, we explore patterns in consumption data over time Second, we explore patterns in independent variables such as temperature, natural gas consumption, prices etc. In the third and fourth sections, we build time series models: simple models first to establish a baseline and then more complex models to capture relationships between independent variables and electricity consumption

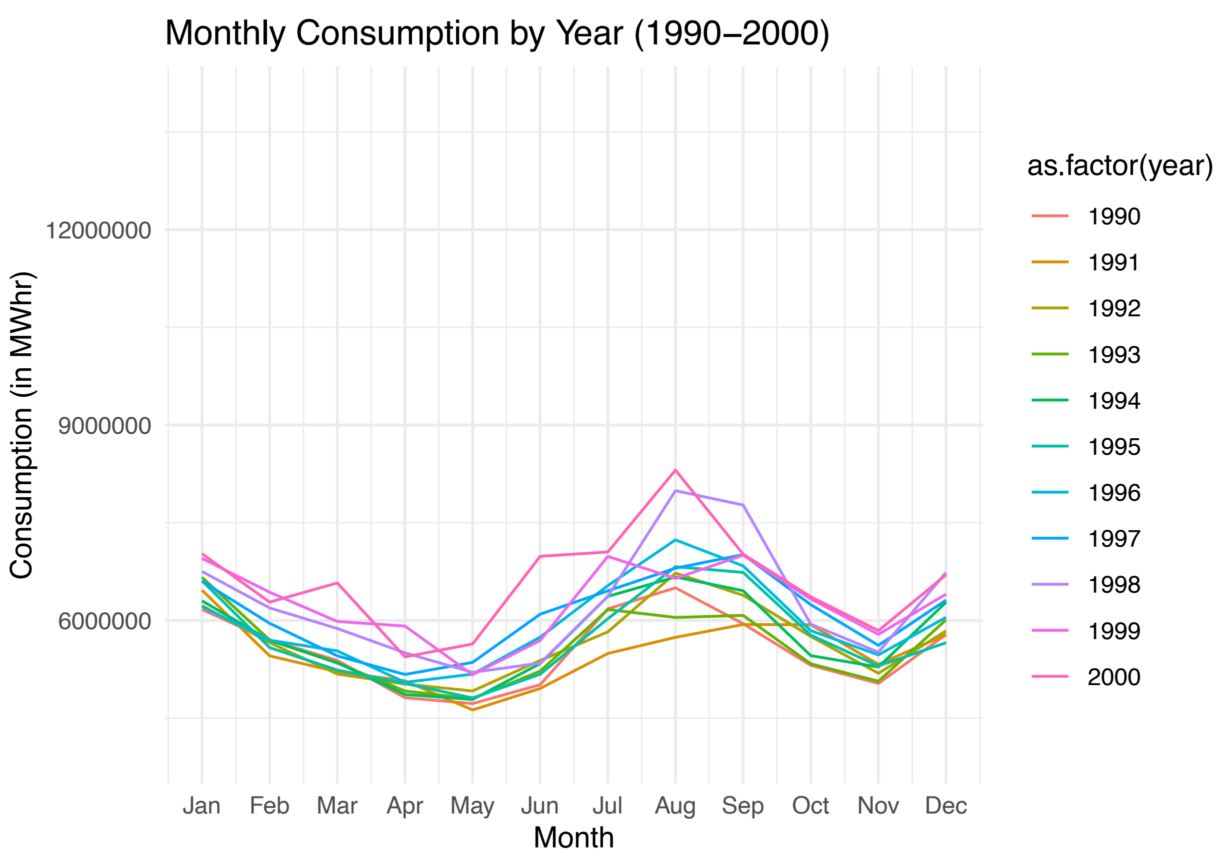
```
suppressMessages({
  library(readxl)
  library(dplyr)
  library(forecast)
  library(tidyverse)
  library(tsibble)
  library(fpp)
  library(tseries)
  library(tidyr)
  library(TSA)
  options(scipen = 999) #this is to switch off the scientific notation
})
```

```
## Warning: package 'ggplot2' was built under R version 4.3.1
```

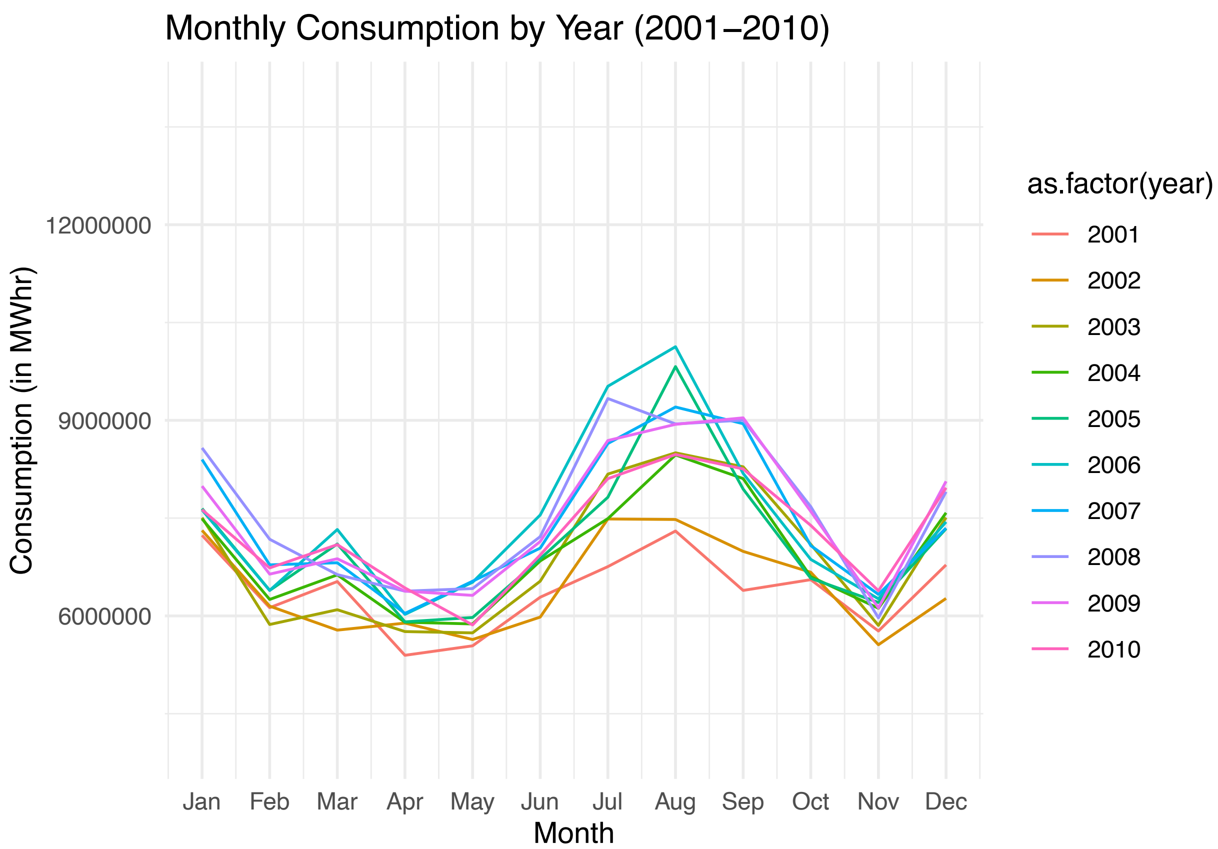
```
# Adding residential electricity data source
file_path <- 'data/sorted_electricity_consumption_data.xlsx'
sorted_data <- read_excel(file_path)
residential <- ts(as.numeric(sorted_data$RESIDENTIAL.Sales.Megawatthours), start = c(1990, 1), frequency
```

```
# Plotting basic trends

library(ggplot2)
library(dplyr)
```

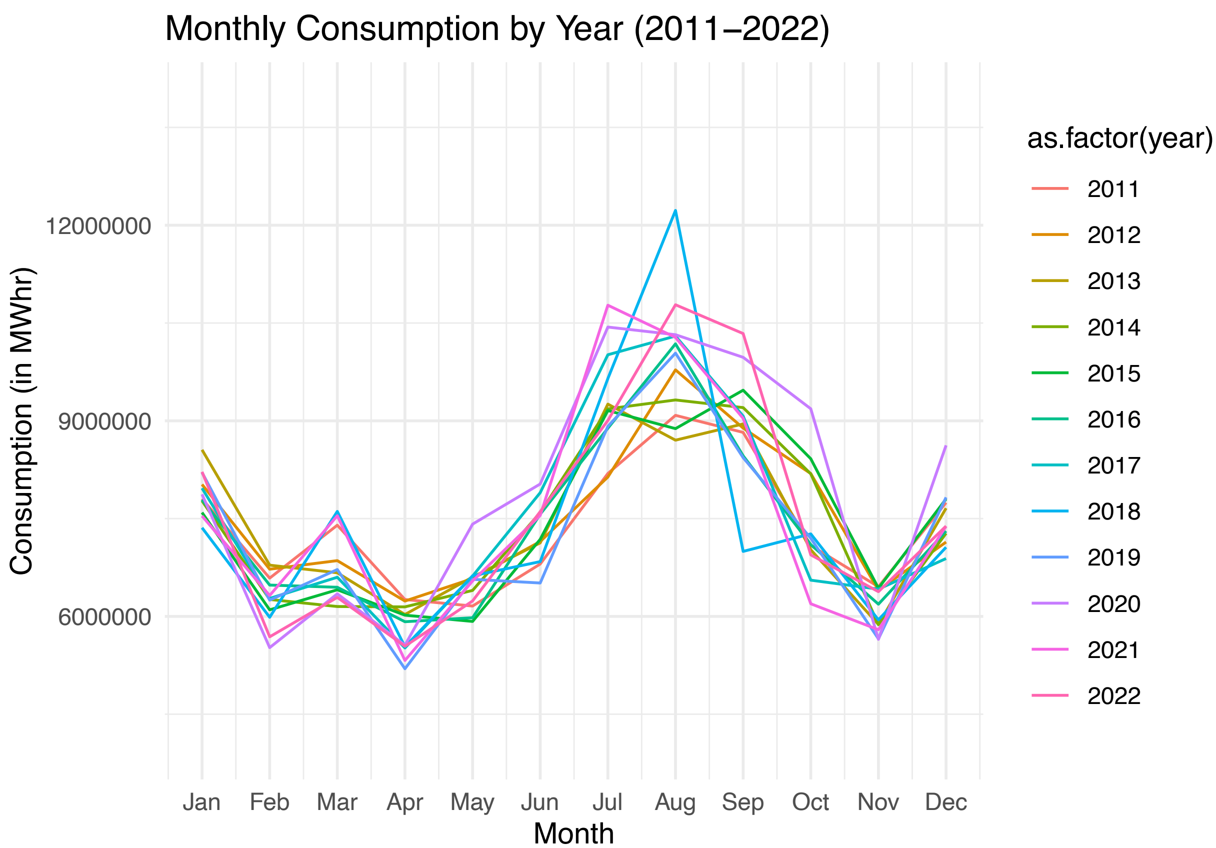
```r
library(lubridate)

# Setting up start and end dates and defining dataframe
dates <- seq(as.Date("1990-01-01"), as.Date("2023-07-01"), by = "month")
df_residential <- data.frame(date = dates, residential)


# Extractinh year and month
df_residential$year <- year(df_residential$date)
df_residential$month <- month(df_residential$date)

# Filter data till 2022
df_residential_1990_2000 <- df_residential %>% filter(year < 2001)
df_residential_2001_2010 <- df_residential %>% filter(year > 2000 & year < 2011)
df_residential_2011_2022 <- df_residential %>% filter(year > 2010 & year < 2023)

# Plot
ggplot(df_residential_1990_2000, aes(x = month, y = residential, group = year, color = as.factor(year))
  geom_line() +
  scale_x_continuous(breaks = 1:12, labels = month.abb) +
  labs(title = "Monthly Consumption by Year (1990-2000)", x = "Month", y = "Consumption (in MWhr)") +
  scale_y_continuous(limits = c(4e6, 14e6)) +
  theme_minimal()
```



```
ggplot(df_residential_2001_2010, aes(x = month, y = residential, group = year, color = as.factor(year))
  geom_line() +
  scale_x_continuous(breaks = 1:12, labels = month.abb) +
  labs(title = "Monthly Consumption by Year (2001-2010)", x = "Month", y = "Consumption (in MWhr)") +
  scale_y_continuous(limits = c(4e6, 14e6)) +
  theme_minimal()
```

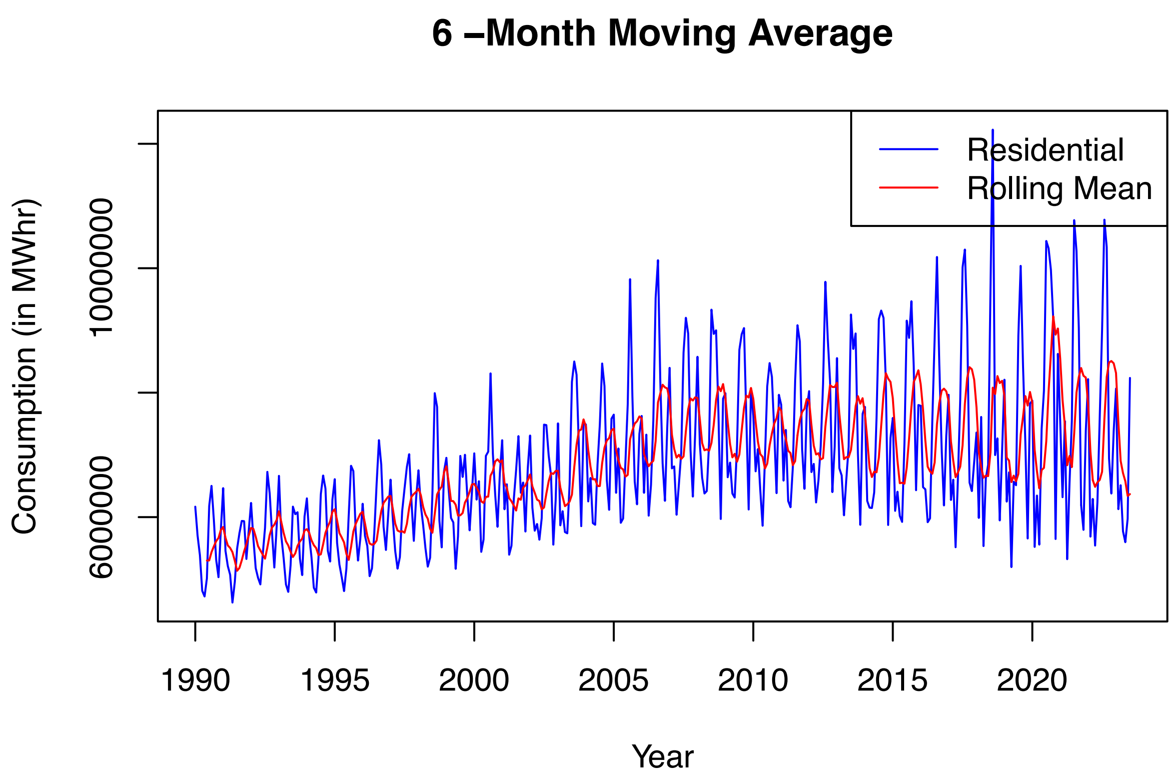


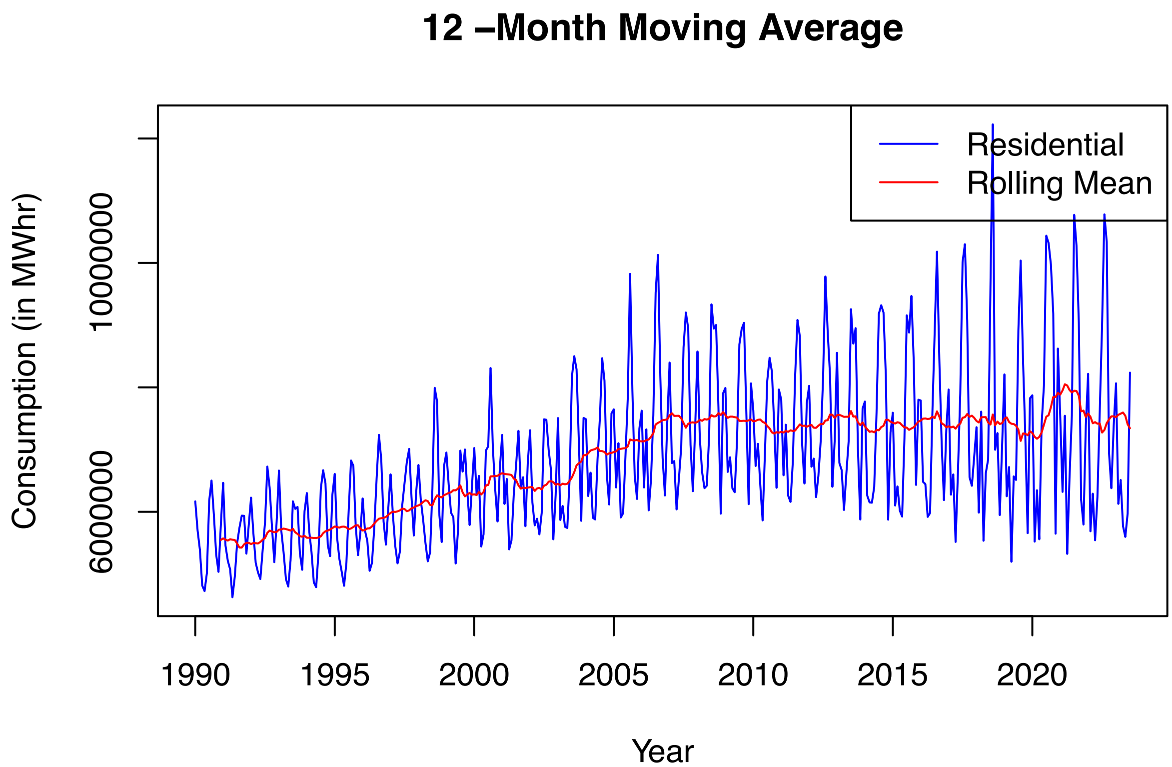

```
ggplot(df_residential_2011_2022, aes(x = month, y = residential, group = year, color = as.factor(year))
  geom_line() +
  scale_x_continuous(breaks = 1:12, labels = month.abb) +
  labs(title = "Monthly Consumption by Year (2011-2022)", x = "Month", y = "Consumption (in MWhr)") +
  scale_y_continuous(limits = c(4e6, 14e6)) +
  theme_minimal()
```

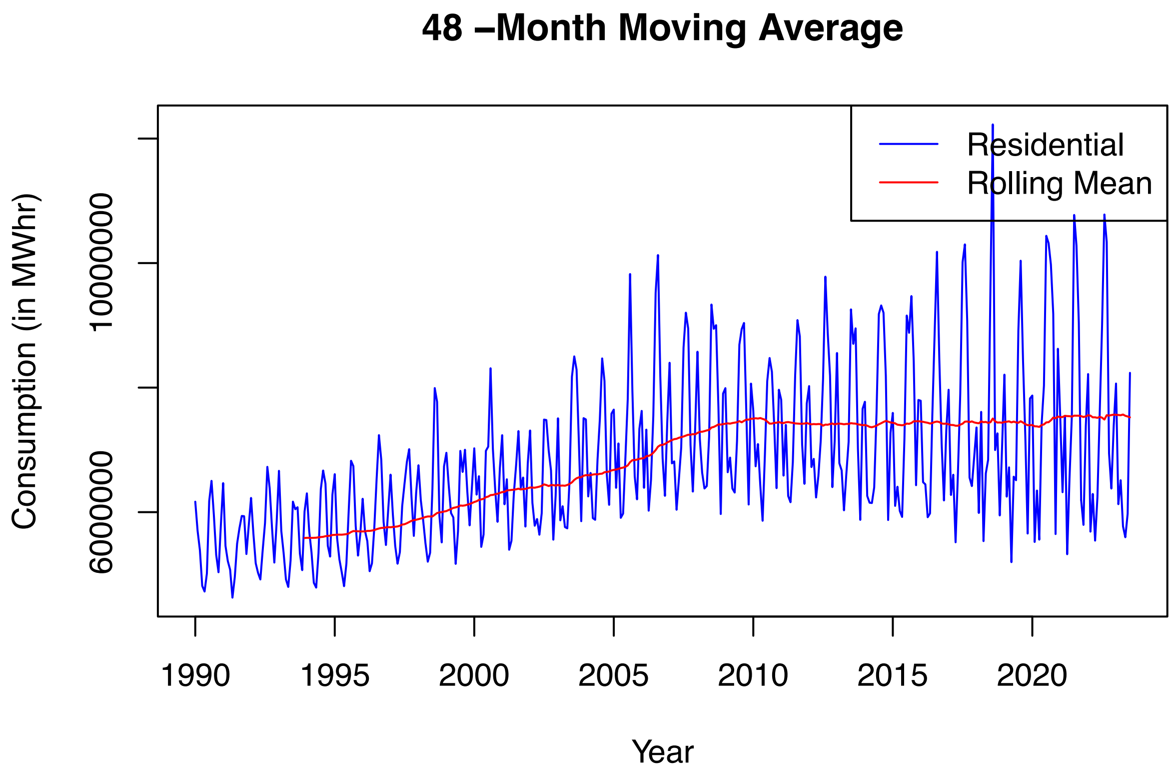



```r
# Plotting moving mean and standard deviation
# To explore variation of data with time, it is standard practice to check how
# averages and variance/standard deviation varies over fixed periods of time (6
# months, 12 months etc.)

# Define window sizes
window_sizes <- c(6, 12, 18, 24, 36, 48, 56)

# Create rolling averages for each window size
rolling_avgs <- lapply(window_sizes, function(WinSize) {
  df_residential %>%
    mutate(
      rolling_mean = zoo::rollmean(residential, WinSize, fill = NA, align = "right"),
      rolling_sd = zoo::rollapply(residential, WinSize, sd, fill = NA, align = "right")
      )
    })
plot_rolling_avgs <- function(rolling_avgs, window_sizes) {
  for (i in 1:6) {
    # Create the initial plot with the first series
    plot(rolling_avgs[[i]]$residential, type = "l", col = "blue", xlab = "Year",
         ylab = "Consumption (in MWhr)", main = paste(window_sizes[i], "-Month Moving Average"))

    # Add the other two series to the existing plot using lines()
    lines(rolling_avgs[[i]]$rolling_mean, col = "red")

    # Add a legend
```

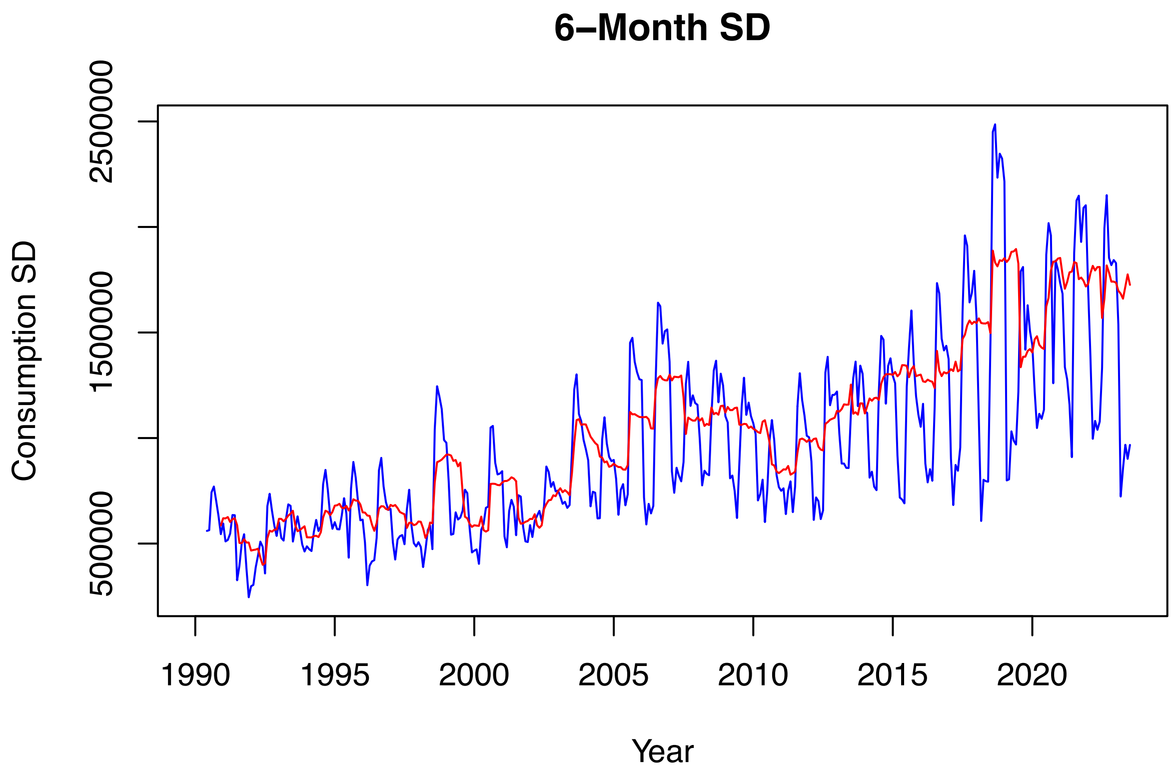
```
    legend("topright", legend = c("Residential", "Rolling Mean"),
           col = c("blue", "red"), lty = 1)

    # Pause before moving to the next plot (optional)
    Sys.sleep(1)
    }
  }
# Call the function with your rolling_avgs list
plot_rolling_avgs(rolling_avgs, window_sizes)
```



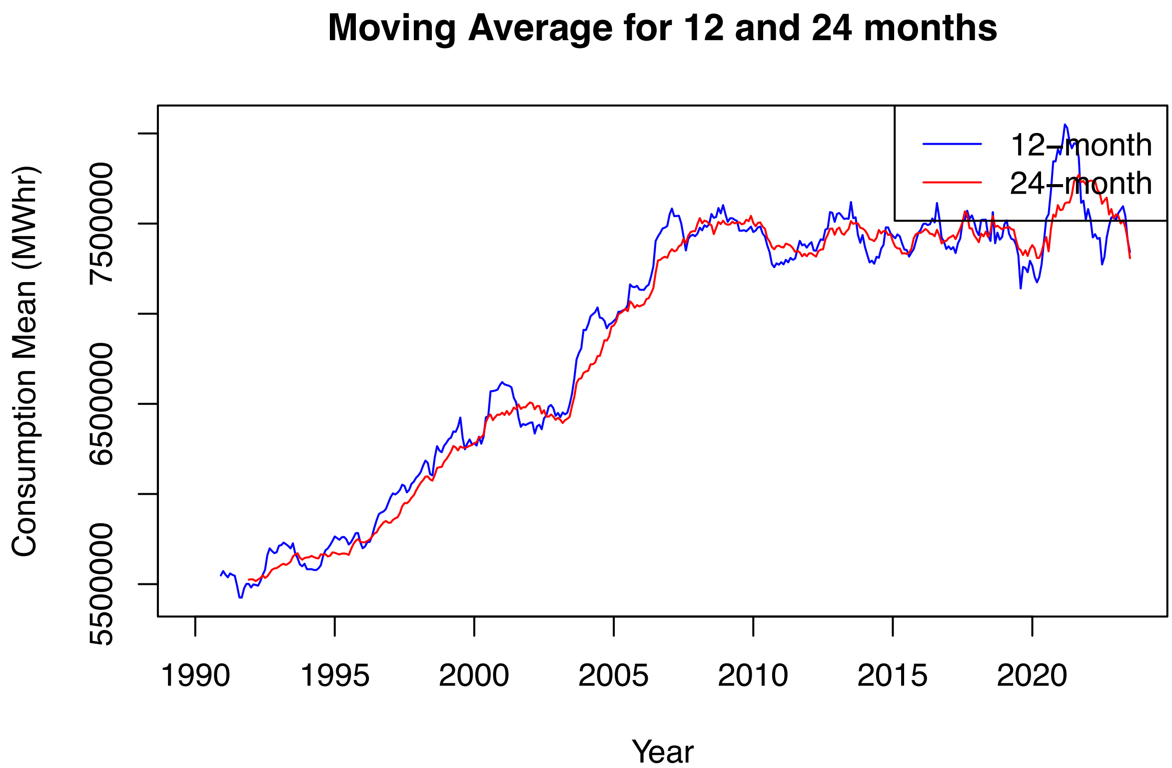

12 –Month Moving Average
Residential
Rolling Mean
Consumption (in MWhr)
6000000
10000000
1990
1995
2000
2005
2010
2015
2020
Year




```
# Also drawing the SD plots
# 6 month SD, 12-month SD
# i = 1, i = 2
plot(rolling_avgs[[1]]$rolling_sd, type = "l", col = "blue", xlab = "Year",ylab = "Consumption SD",
     main = "6-Month SD")

lines(rolling_avgs[[2]]$rolling_sd, type = "l", col = "red", xlab = "Year",ylab = "Consumption SD",
     main = "12-Month SD")
```

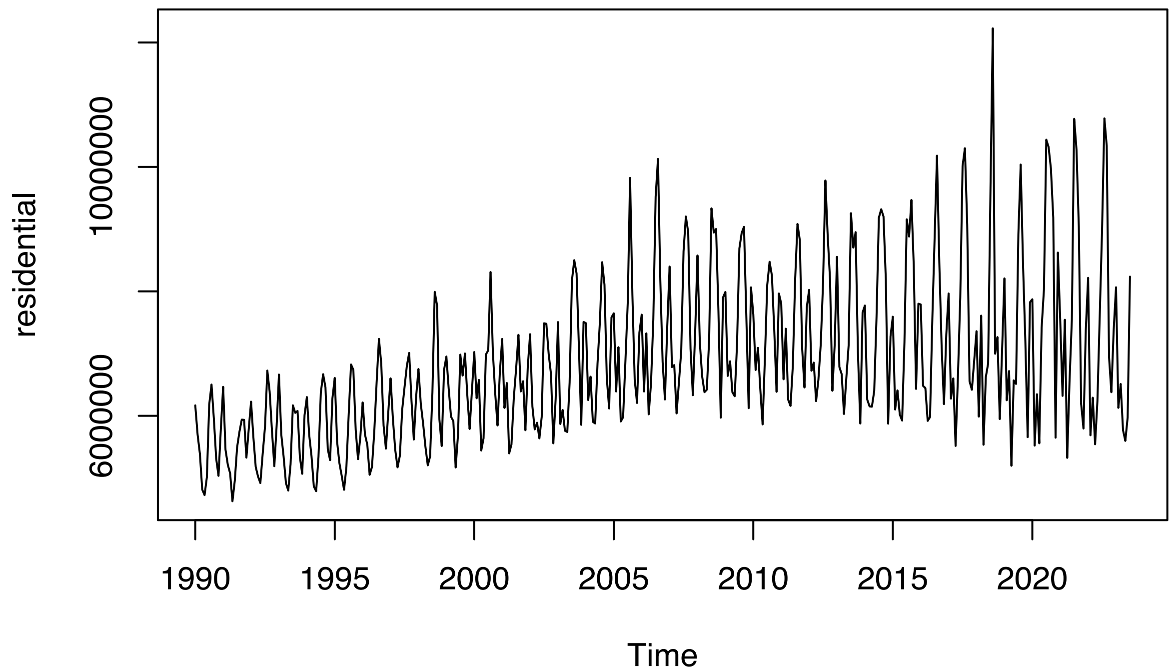



```r
## Checking presence of different data generating processes

# 12, 24 month MA
# i = 2,4

plot(rolling_avgs[[2]]$rolling_mean, type = "l", col = "blue", xlab = "Year",
     ylab = "Consumption Mean (MWhr)",
     main = "Moving Average for 12 and 24 months")

lines(rolling_avgs[[4]]$rolling_mean, type = "l", col = "red")
legend("topright", legend = c("12-month", "24-month"), col = c("blue", "red"),
       lty = 1)
```

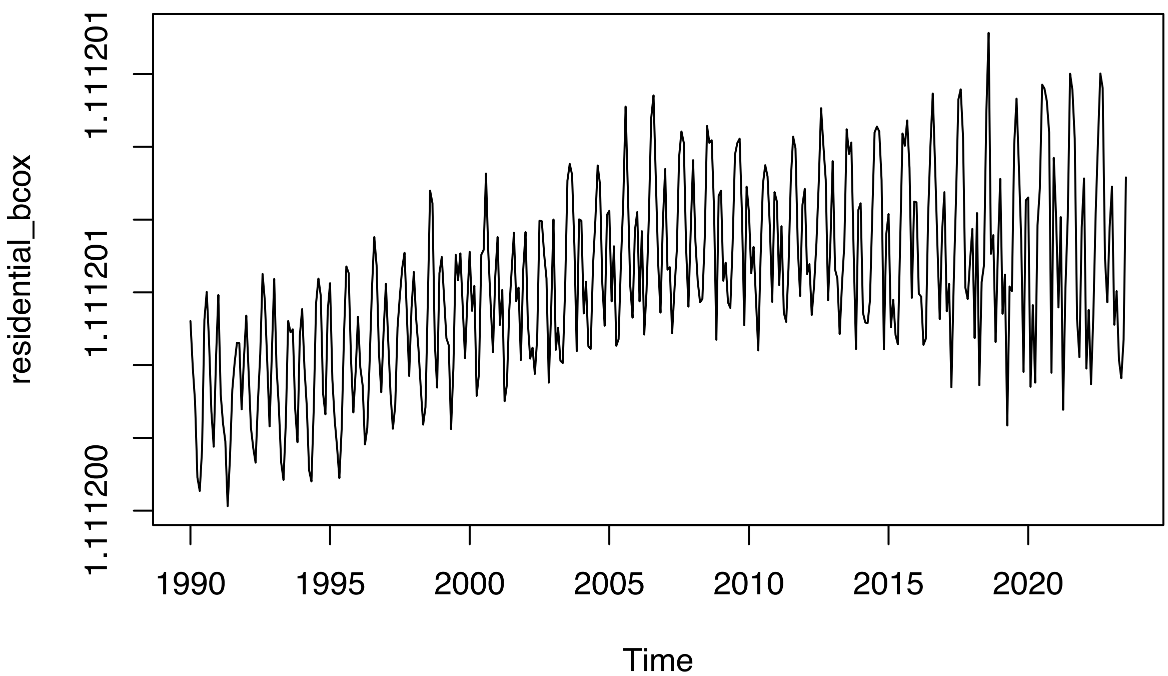


Shift between 2000 and 2005 indicates change due to a new data generation ### process (could be impact of regulation/policy).

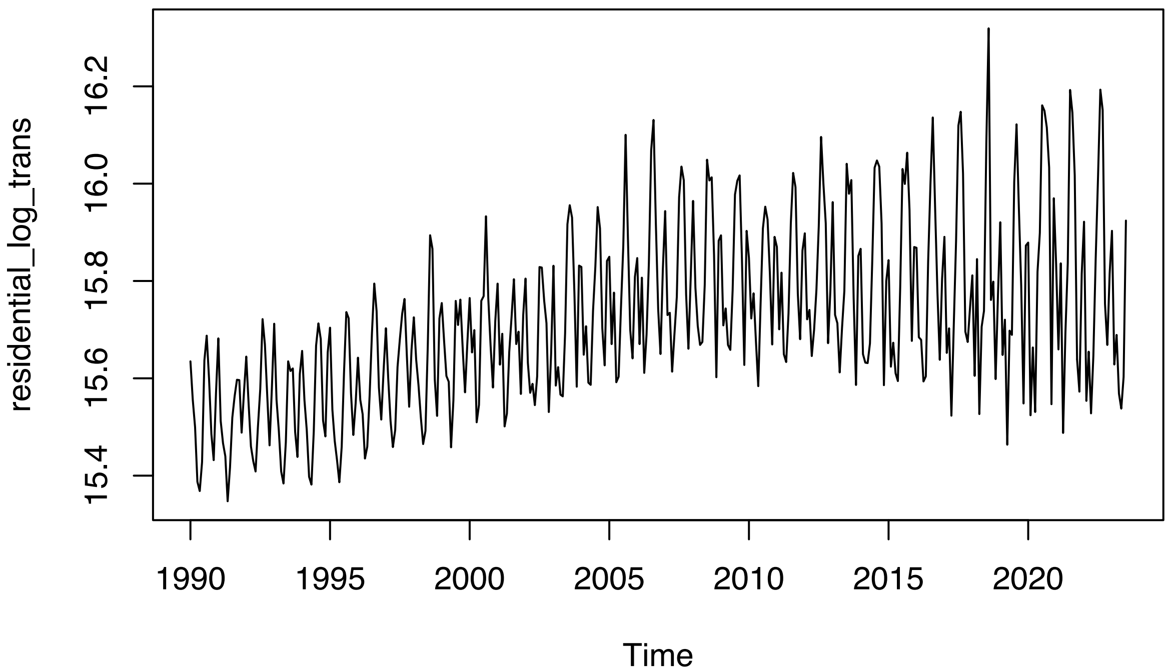
```
# Checking how log consumption looks like before/after applying Box-cox
# transformations
residential_log_trans <- BoxCox(residential, lambda = 0)
residential_bcox <- BoxCox(residential, lambda = 'auto')
plot(residential)
```




```
plot(residential_bcox)
```



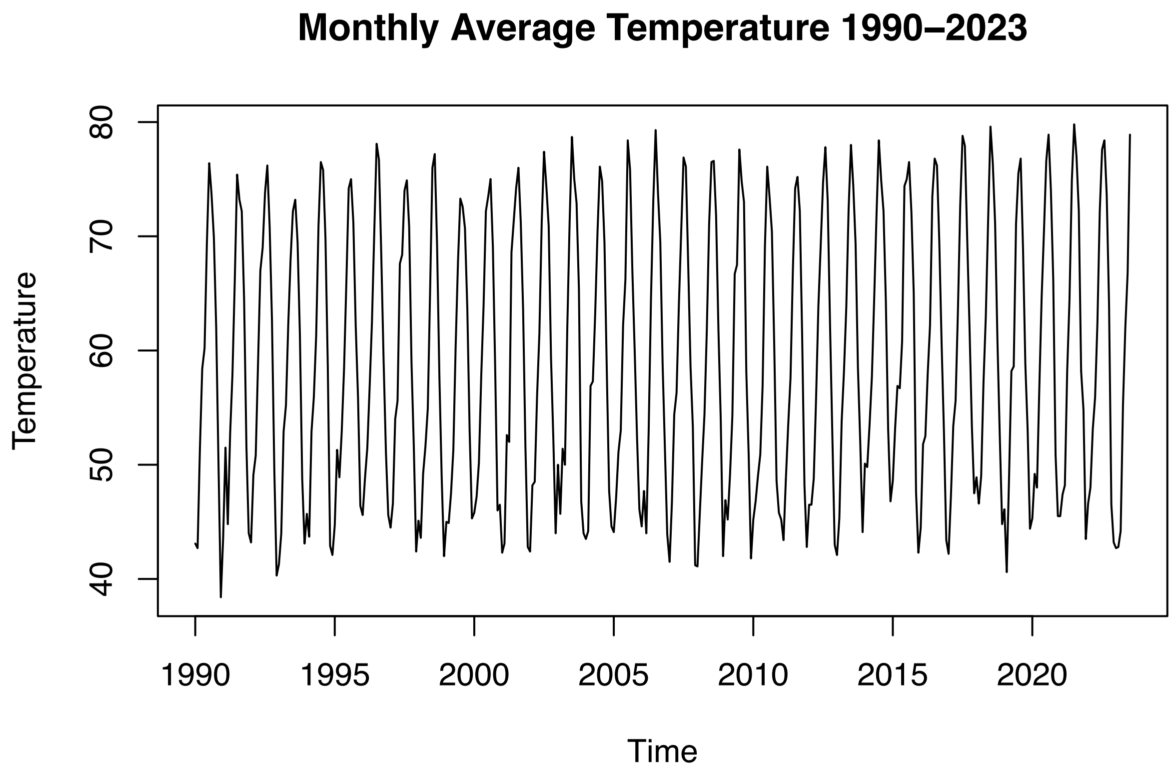

```
plot(residential_log_trans)
```



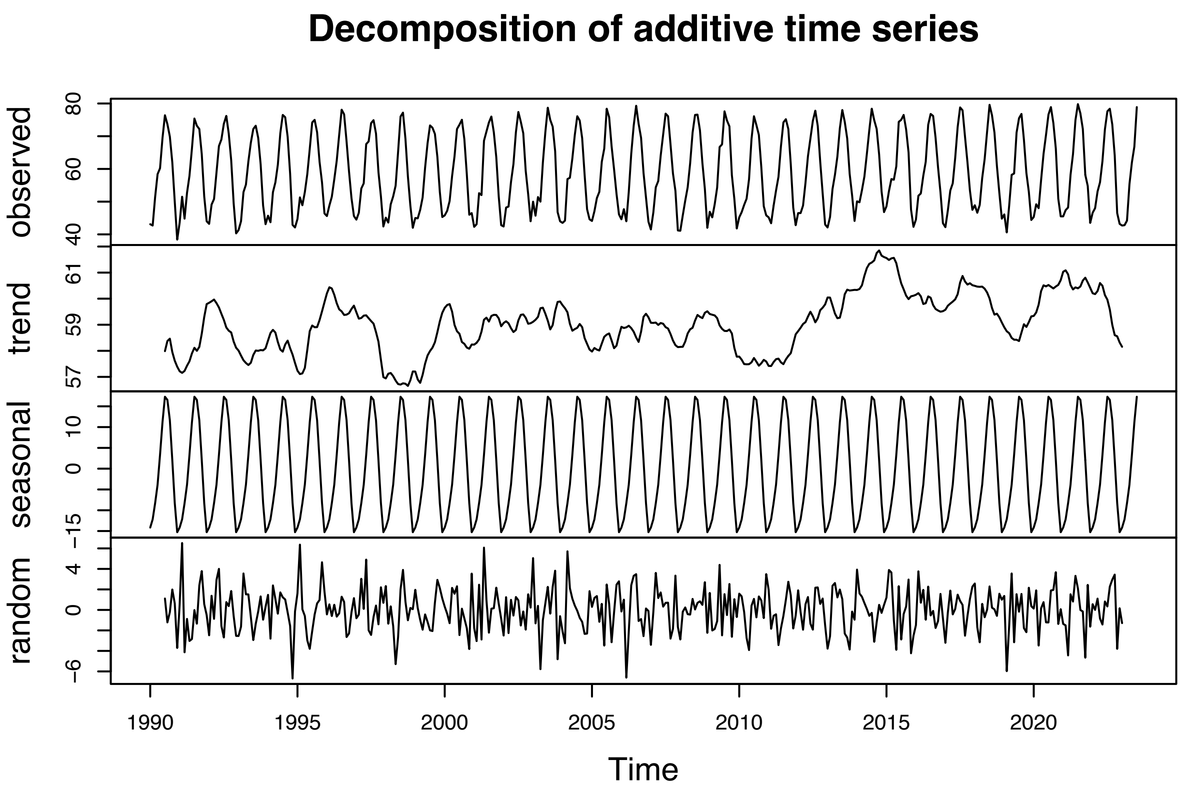
Supplementary Data: we explore patterns in potential independent variables, then split all data sources into test and train

Monthly Temperature Data for all of California

```r
# Load the Monthly Temperature Data

temp_data <- read_excel('./data/ca_monthly_temp_data.xlsx')
temp_ts <- ts(temp_data$Value, frequency = 12, start=c(1990,1), end=c(2023, 7))
plot(temp_ts, ylab="Temperature", main="Monthly Average Temperature 1990-2023")
```




```
plot(decompose(temp_ts))
```



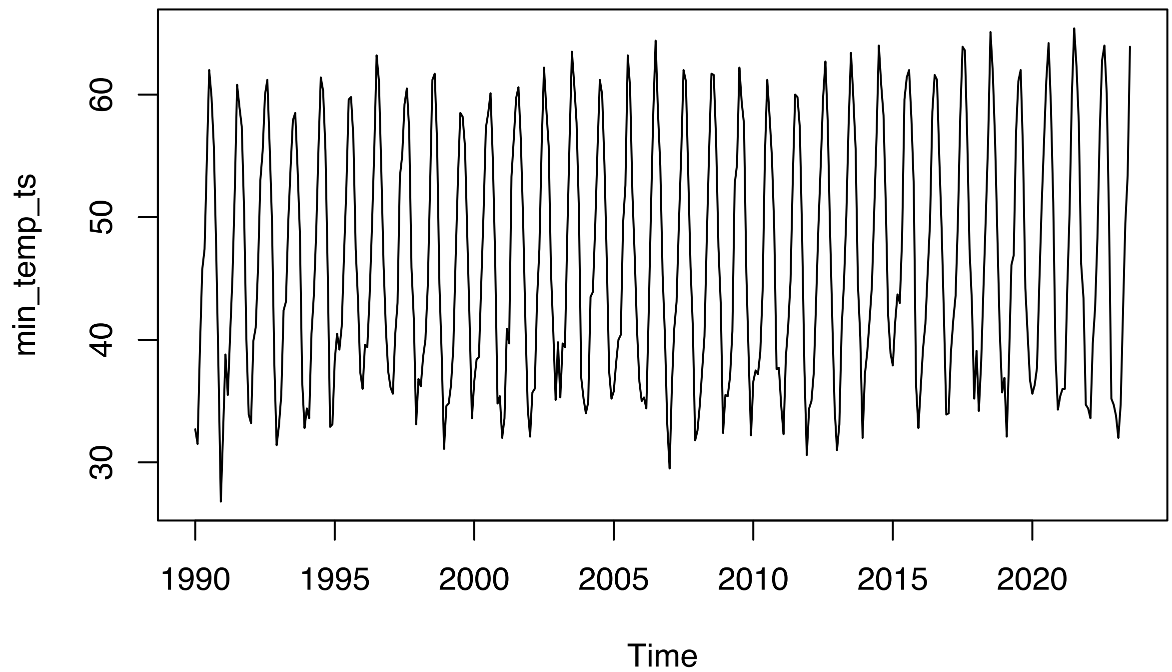
With the temperature data we do not seem to have a notable upward or downward trend, though from the decomposition it might be experiencing a slight positive trend. We see a consistent 12-month seasonal trend that seems to be additive in nature given that the amplitude of the variance is not growing from the looks of it.

Read in minimum temperatures.

```
min_temp <- read_csv(url("https://www.ncei.noaa.gov/access/monitoring/climate-at-a-glance/statewide/tim
```

```
## Rows: 408 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (3): Date, Value, Anomaly
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
min_temp_ts <- ts(min_temp$`Value`, frequency = 12, start =c(1990,1), end=c(2023, 7))

plot(min_temp_ts)
```

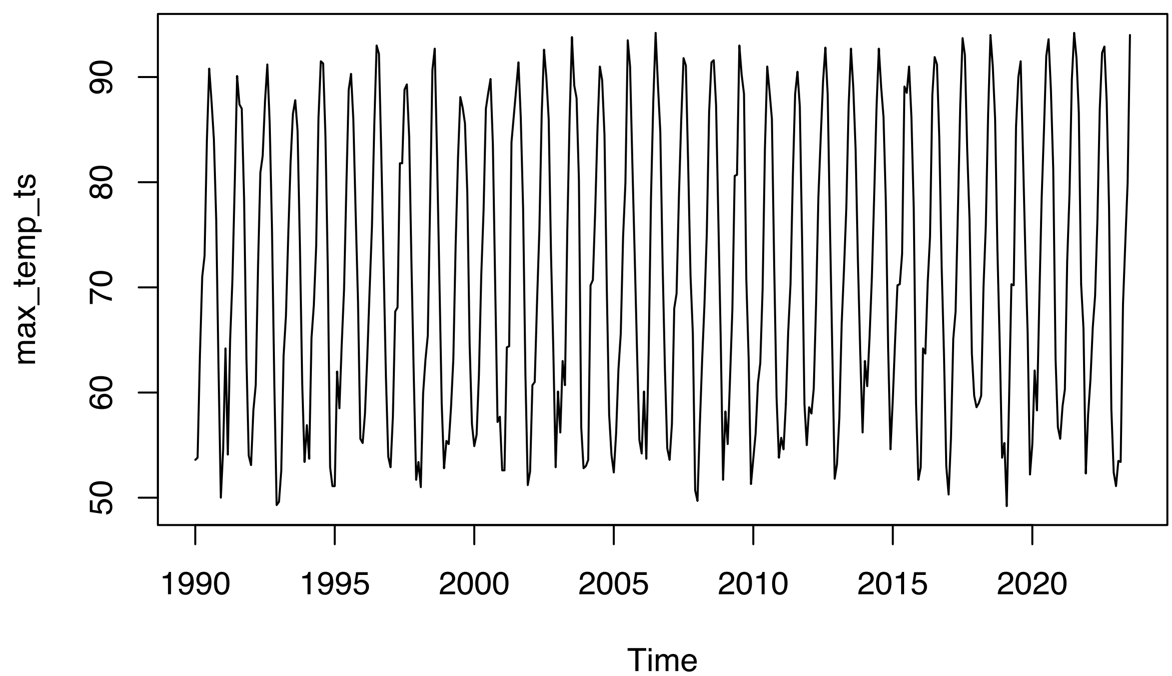


Read in maximum temperatures.

```
max_temp <- read_csv(url("https://www.ncei.noaa.gov/access/monitoring/climate-at-a-glance/statewide/tim
```

```
## Rows: 408 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (3): Date, Value, Anomaly
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
max_temp_ts <- ts(max_temp$`Value`, frequency = 12, start =c(1990,1), end=c(2023, 7)) # ending in July
plot(max_temp_ts)
```

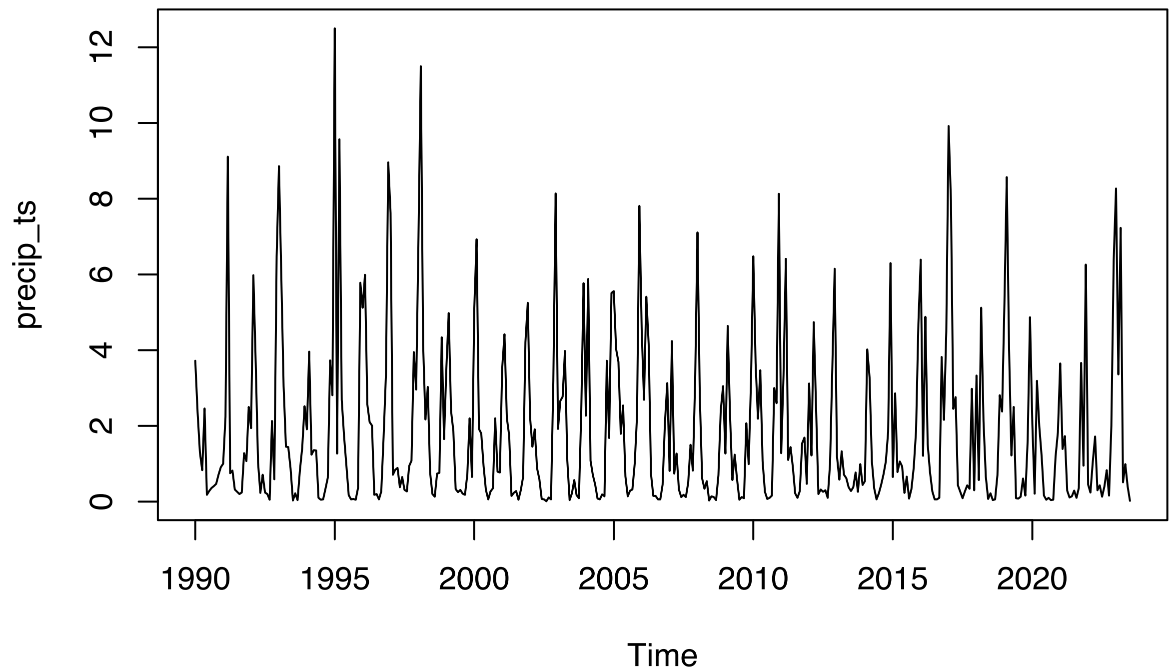


Precipitation Data

```
cali_precip <- read_csv('./data/weather_exp/data_preci.csv')
```

```
## Rows: 406 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): month
## dbl (4): Date, month_num, year, Value
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
precip_ts <- ts(cali_precip$`Value`, frequency = 12, start =c(1990,1), end=c(2023, 7)) # ending in July

plot(precip_ts)
```

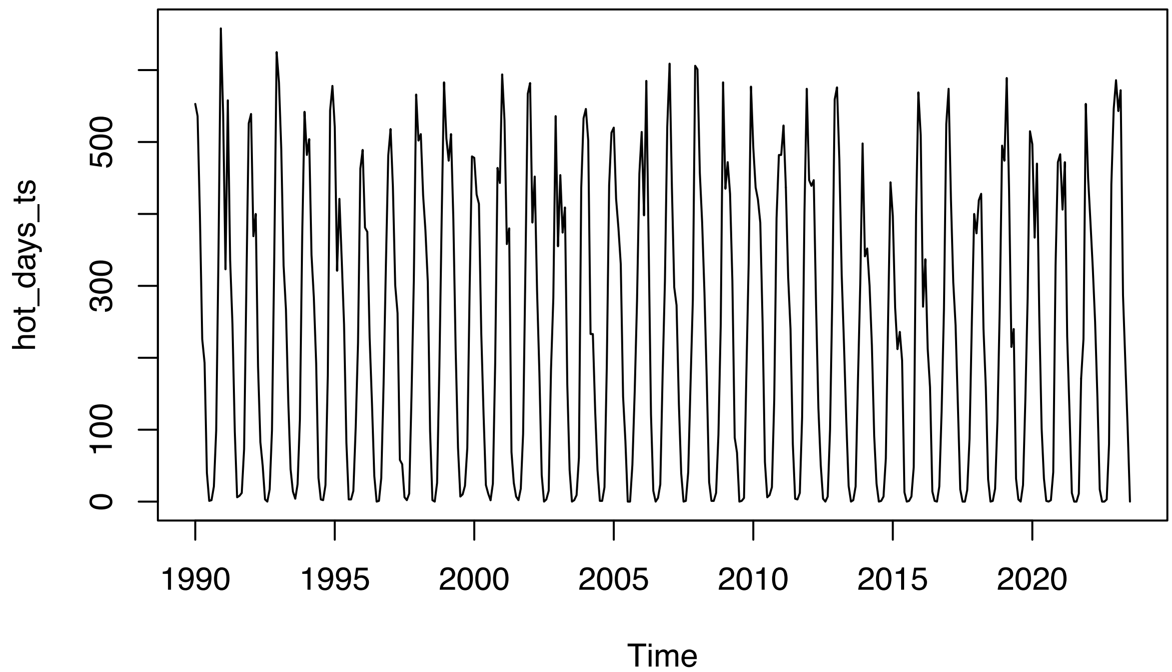


Hot Days

```
hot_days <- read_csv('./data/weather_exp/data_heat_days.csv')
```

```
## Rows: 406 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): month
## dbl (4): Date, mont_num, year, Value
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
hot_days_ts <- ts(hot_days$`Value`, frequency = 12, start =c(1990,1), end=c(2023, 7)) # ending in July 

plot(hot_days_ts)
```

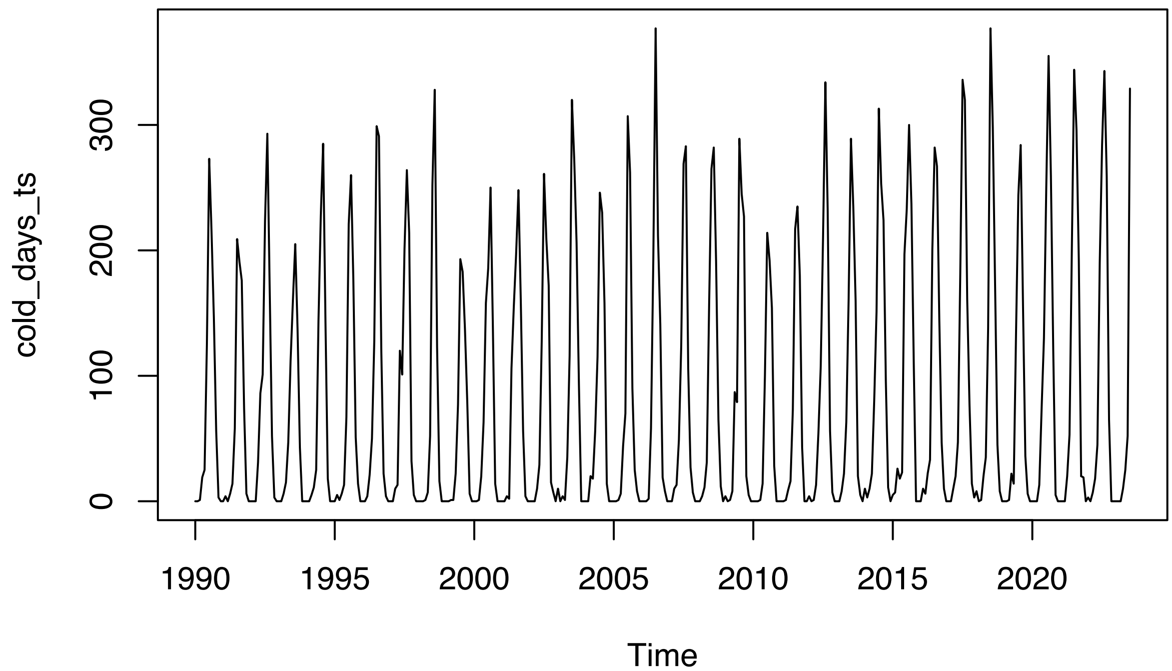


Cold Days

```r
cold_days <- read_csv('./data/weather_exp/data_cool_days.csv')
```

```
## Rows: 406 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): month
## dbl (4): Date, month_num, year, Value
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
cold_days_ts <- ts(cold_days$`Value`, frequency = 12, start =c(1990,1), end=c(2023, 7)) # ending in July

plot(cold_days_ts)
```

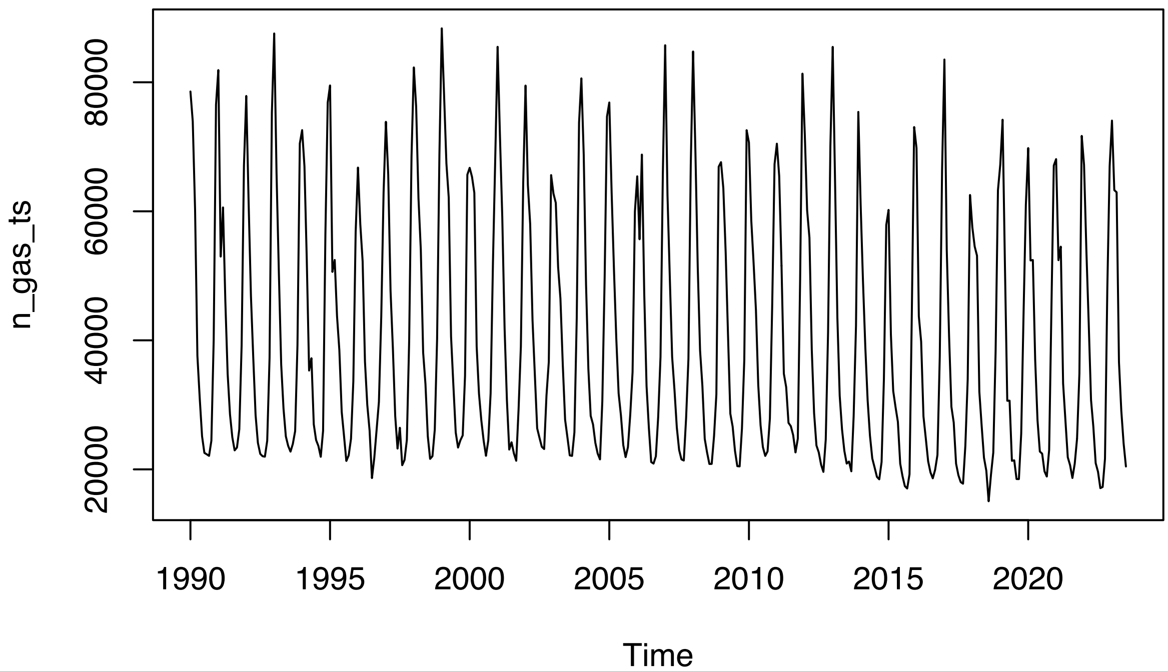


Natural gas consumption

```r
n_gas <- read_csv('./data/natural_gas/California_Natural_Gas_Residential_Consumption.csv')
```

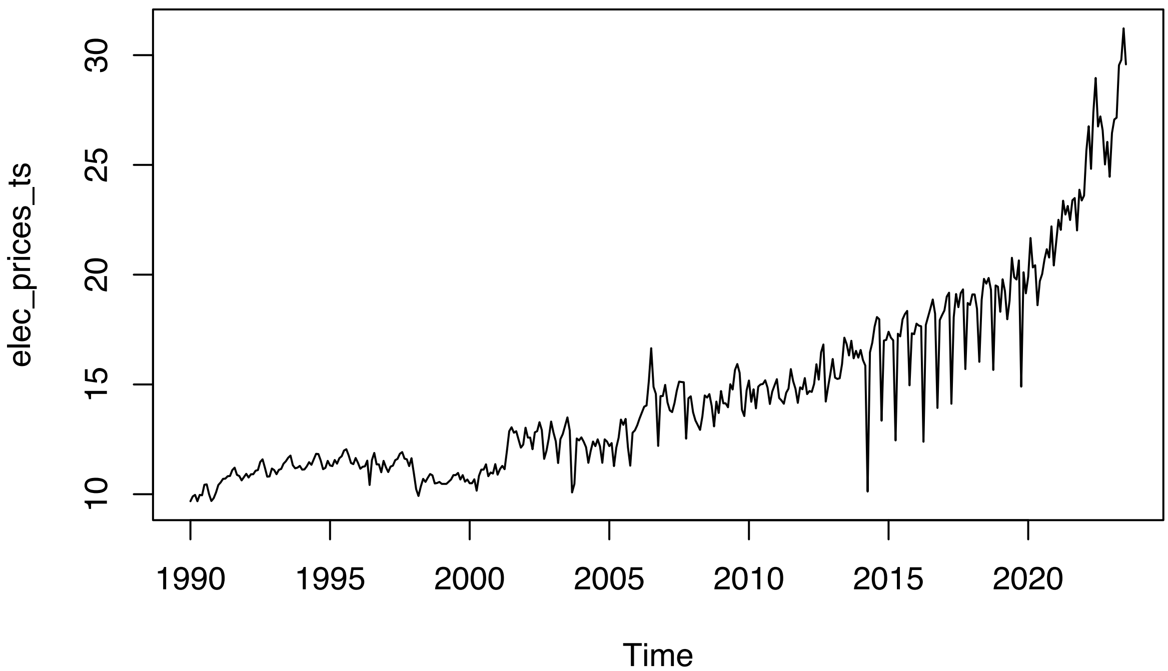
```
## Rows: 404 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): stamp, month
## dbl (3): num_month, year, Value
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
n_gas_ts <- ts(n_gas$`Value`, frequency = 12, start =c(1990,1), end=c(2023, 7))
# ending in July 2023 when consumption data ends
plot(n_gas_ts)
```



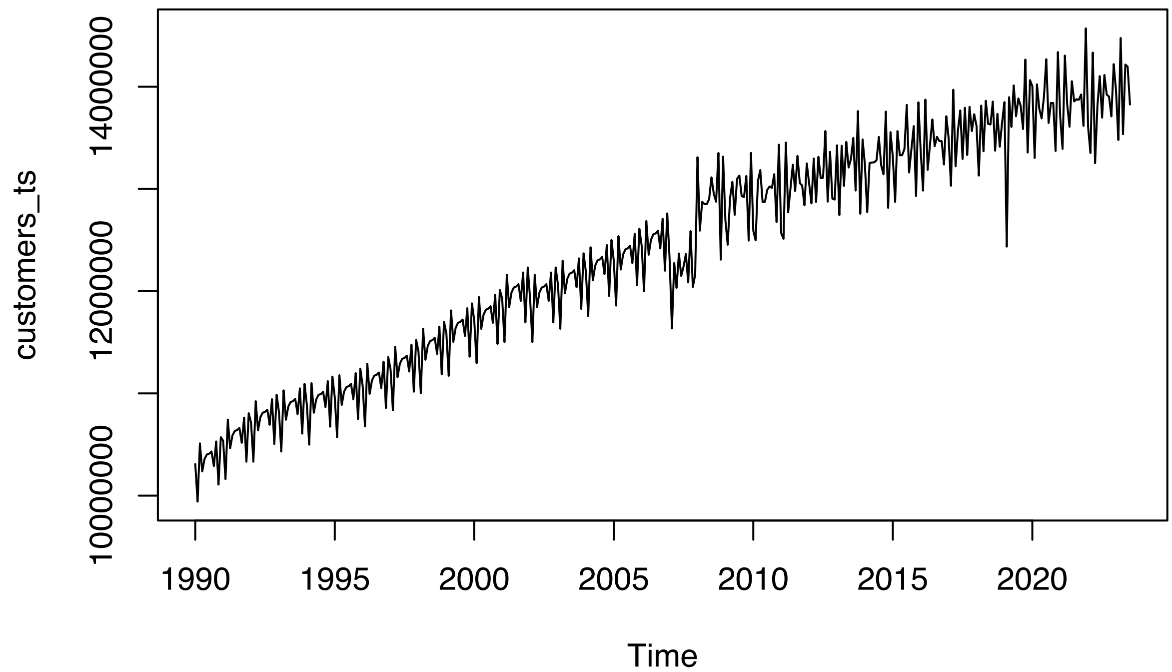
Pricing

```
elec_prices_ts <- ts(as.numeric(sorted_data$RESIDENTIAL.Price.Cents.kWh), start = c(1990, 1),
                     end=c(2023, 7),frequency = 12)
plot(elec_prices_ts)
```



Number of residential customers

```
customers <- read_excel('./data/customers_imputed.xlsx', sheet = "Sheet2")
customers_ts <- ts(customers$`customers`, frequency = 12, start =c(1990,1),
                   end=c(2023, 7)) # ending in July 2023 when consumption data ends

plot(customers_ts)
```



TRAIN TEST SPLITS

```
#train/test split at 1990
train_res_1990 <- window(residential, start=1990, end=c(2022, 6))
train_res_logt_1990 <- window(residential_log_trans, start = 1990,
                              end = c(2022, 6))
train_res_boxc_1990 <- window(residential_bcox, start = 1990, end = c(2022,6))
test_res_1990 <- window(residential, start=c(2022, 7), end=c(2023, 7))
train_test_res_1990 <- window(residential, start=1990, end=c(2023, 7))
```

Train-Test Splits for Independent Variables

```
split_time_series <- function(time_series, year) {
  start_train <- c(year, 1)
  end_train <- c(2022, 6)
  start_test <- c(2022, 7)
  end_test <- c(2023, 7)
  start_train_test <- c(year, 1)
  end_train_test <- c(2023, 7)

  train_set <- window(time_series, start = start_train, end = end_train)
  test_set <- window(time_series, start = start_test, end = end_test)
```

```r
  train_test_set <- window(time_series, start = start_train_test, end = end_train_test)

  return(list(train = train_set, test = test_set, train_test = train_test_set))
}
```

```r
# train test split for temp data
min_1990 <- split_time_series(min_temp_ts, 1990)
train_min_temp_1990 <- min_1990$train
test_min_temp_1990 <- min_1990$test
train_test_min_temp_1990 <- min_1990$train_test

max_1990 <- split_time_series(max_temp_ts, 1990)
train_max_temp_1990 <- max_1990$train
test_max_temp_1990 <- max_1990$test
train_test_max_temp_1990 <- max_1990$train_test

## New additions

#temperature
avgtemp_1990 <- split_time_series(temp_ts, 1990)
train_avgtemp_1990 <- avgtemp_1990$train
test_avgtemp_1990 <- avgtemp_1990$test
train_test_avgtemp_1990 <- avgtemp_1990$train_test

# precipitation
precip_1990 <- split_time_series(precip_ts, 1990)
train_precip_1990 <- precip_1990$train
test_precip_1990 <- precip_1990$test
train_test_precip_1990 <- precip_1990$train_test

# hot days
hdays_1990 <- split_time_series(hot_days_ts, 1990)
train_hdays_1990 <- hdays_1990$train
test_hdays_1990 <- hdays_1990$test
train_test_hdays_1990 <- hdays_1990$train_test

# cold days
cdays_1990 <- split_time_series(cold_days_ts, 1990)
train_cdays_1990 <- cdays_1990$train
test_cdays_1990 <- cdays_1990$test
train_test_cdays_1990 <- cdays_1990$train_test

# natural gas
ngas_1990 <- split_time_series(n_gas_ts, 1990)
train_ngas_1990 <- ngas_1990$train
test_ngas_1990 <- ngas_1990$test
train_test_ngas_1990 <- ngas_1990$train_test

#pricing
elec_prices_1990 <- split_time_series(elec_prices_ts, 1990)
train_elec_prices_1990 <- elec_prices_1990$train
test_elec_prices_1990 <- elec_prices_1990$test
train_test_elec_prices_1990 <- elec_prices_1990$train_test
```

```
# customers
customers_1990 <- split_time_series(customers_ts, 1990)
train_customers_1990 <- customers_1990$train
test_customers_1990 <- customers_1990$test
train_test_customers_1990 <- customers_1990$train_test
```

Now that we have split all data sources (dependent and independent into

train and test sets, we can can start building forecastign models)

BASELINE MODELS

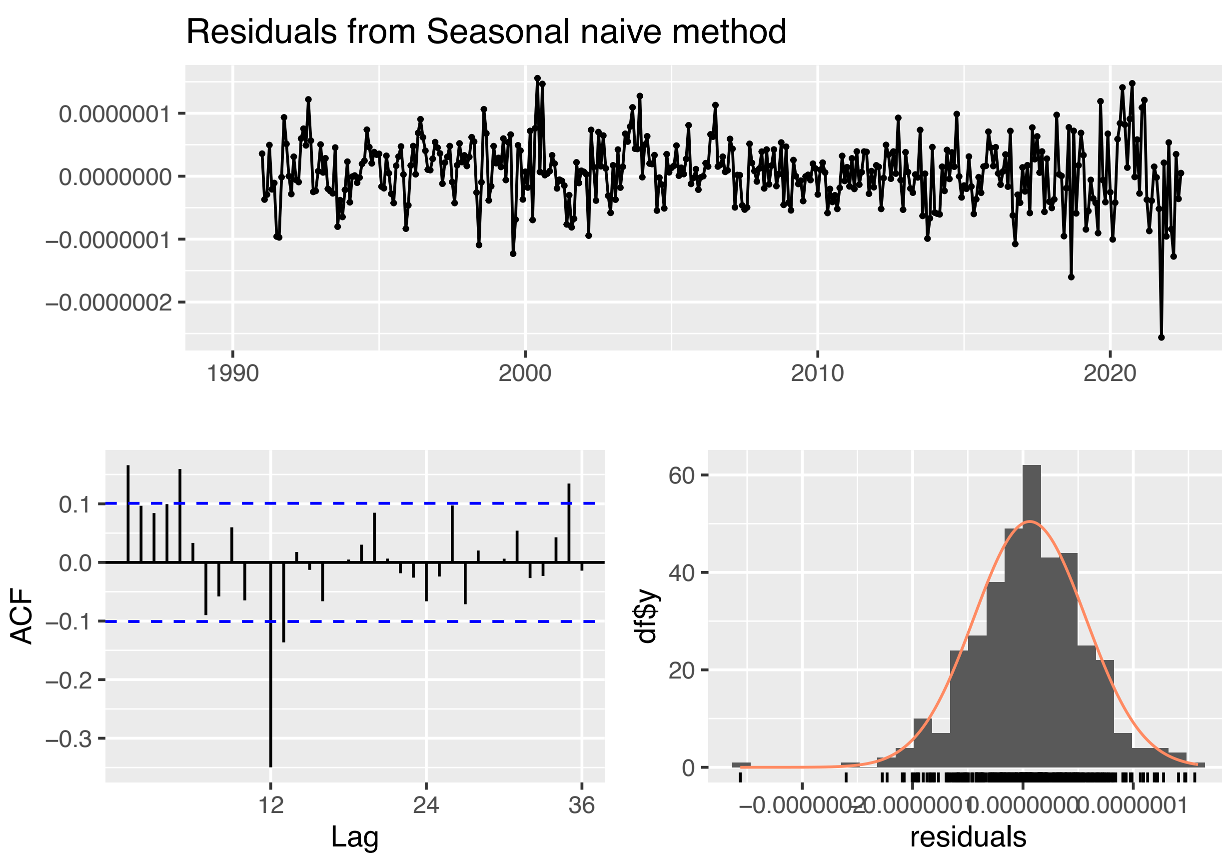
Seasonal Naive baseline model

The point forecasts are the same for each of these models, but the information criteria estimates are different (and the confidence interval bands are different for each)

```
snaive_model_1990 = snaive(train_res_1990, lambda='auto', h=12)
forecast::accuracy(snaive_model_1990)
```

```
##                    ME     RMSE      MAE       MPE     MAPE MASE      ACF1
## Training set 59447.13 562980.8 394083.5 0.5763874 5.521388    1 0.1227548
```

```
checkresiduals(snaive_model_1990)
```



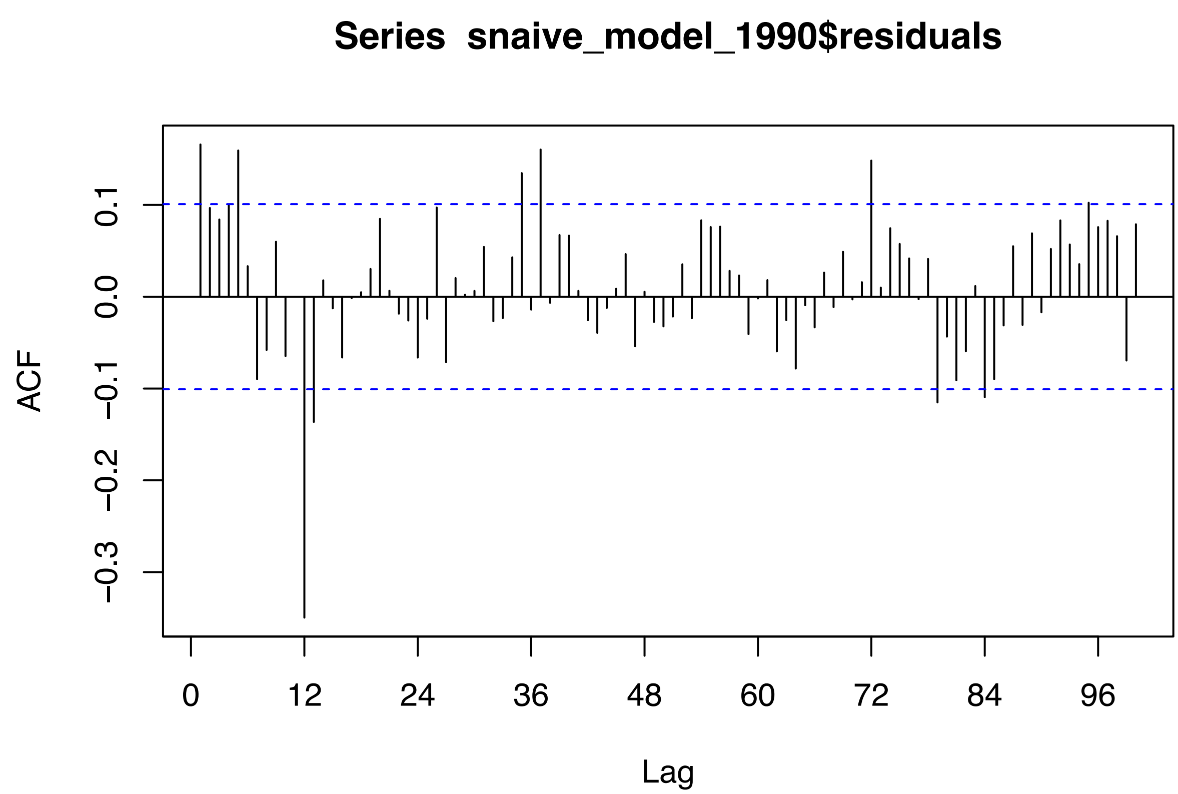
```
##
##  Ljung-Box test
##
## data:  Residuals from Seasonal naive method
## Q* = 100.9, df = 24, p-value = 0.00000000002109
##
## Model df: 0.   Total lags used: 24
```

```r
kpss.test(snaive_model_1990$residuals)
```

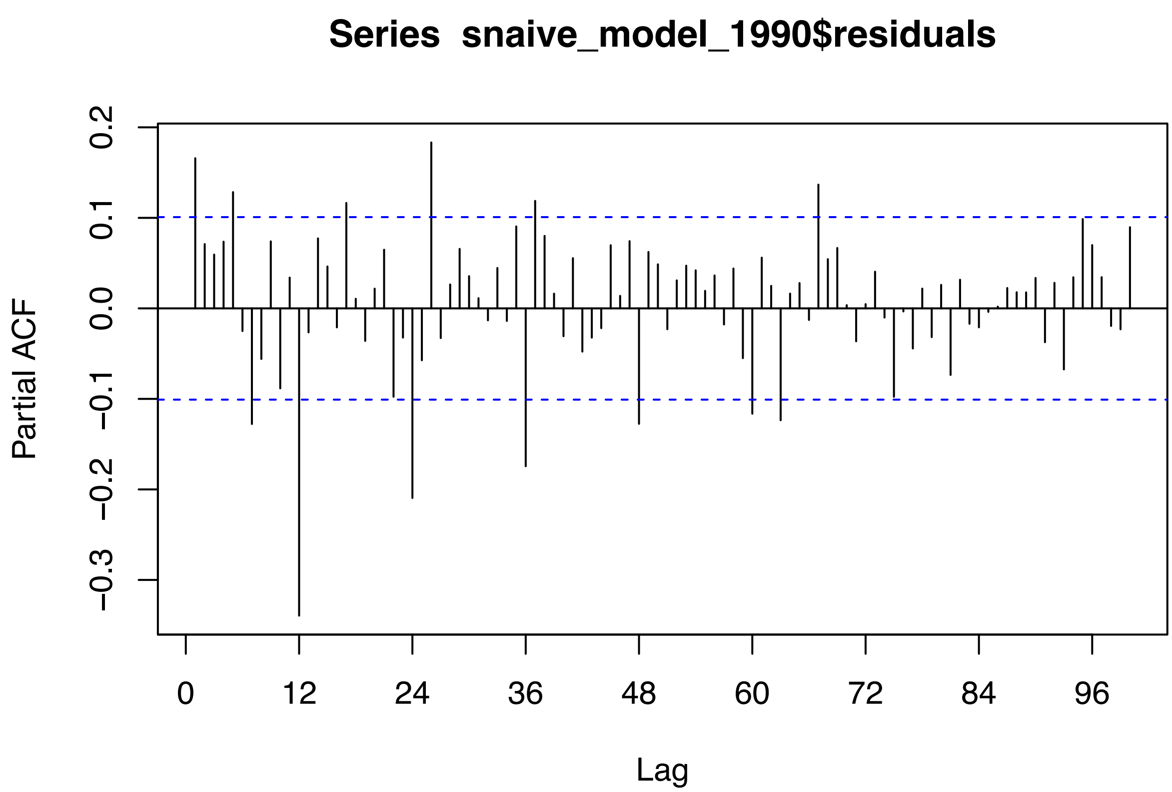
```
##
##  KPSS Test for Level Stationarity
##
## data:  snaive_model_1990$residuals
## KPSS Level = 0.3693, Truncation lag parameter = 5, p-value = 0.09039
```

```r
#snaive_model_2005 = snaive(train_res_2005, lambda='auto', h=12)
#accuracy(snaive_model_2005)

#snaive_model_2013 = snaive(train_res_2013, lambda='auto', h=12)
#accuracy(snaive_model_2013)

#plot(snaive_model_2013)
```

```r
# Examining autocorrelation and partial autocorrelation plots
Acf(snaive_model_1990$residuals, lag=100)
```



```r
Pacf(snaive_model_1990$residuals, lag=100)
```



Exponential Smoothing

model type: ETS(M,N,M) MAPE for 1990: 3.300601 RMSE for 1990: 370256.2 AICc value = -11137.42

```r
ets_model_1990 = ets(train_res_1990, lambda = 'auto')
forecast::accuracy(ets_model_1990)
```

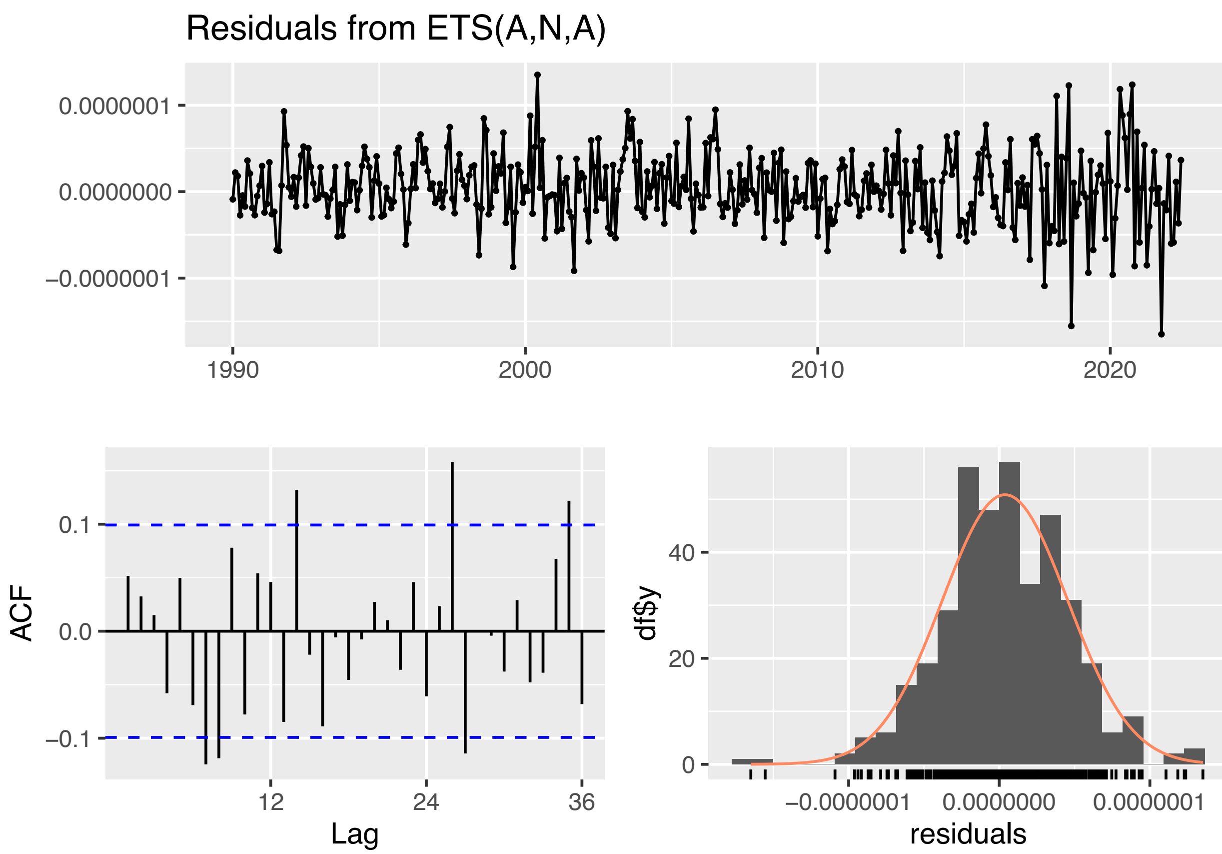
```
##                    ME     RMSE      MAE       MPE    MAPE      MASE       ACF1
## Training set 67127.16 466622.9 328583.2 0.5335448 4.58164 0.8337908 0.03955669
```

```r
summary(ets_model_1990)
```

```
## ETS(A,N,A)
##
## Call:
##  ets(y = train_res_1990, lambda = "auto")
##
##   Box-Cox transformation: lambda= -0.8999
##
##   Smoothing parameters:
##     alpha = 0.112
```

```
##     gamma = 0.2446
##
##   Initial states:
##     l = 1.1112
##     s = 0 0 0 0 0 0
##            0 0 0 0 0 0
##
##   sigma:  0
##
##        AIC      AICc       BIC
## -10894.27 -10892.98 -10834.78
##
## Training set error measures:
##                    ME     RMSE      MAE       MPE    MAPE      MASE       ACF1
## Training set 67127.16 466622.9 328583.2 0.5335448 4.58164 0.8337908 0.03955669
```

```r
checkresiduals(ets_model_1990)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,N,A)
## Q* = 42.136, df = 24, p-value = 0.01246
##
## Model df: 0.   Total lags used: 24
```

```r
kpss.test(ets_model_1990$residuals)
```

```
## Warning in kpss.test(ets_model_1990$residuals): p-value greater than printed
## p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  ets_model_1990$residuals
## KPSS Level = 0.27573, Truncation lag parameter = 5, p-value = 0.1
```

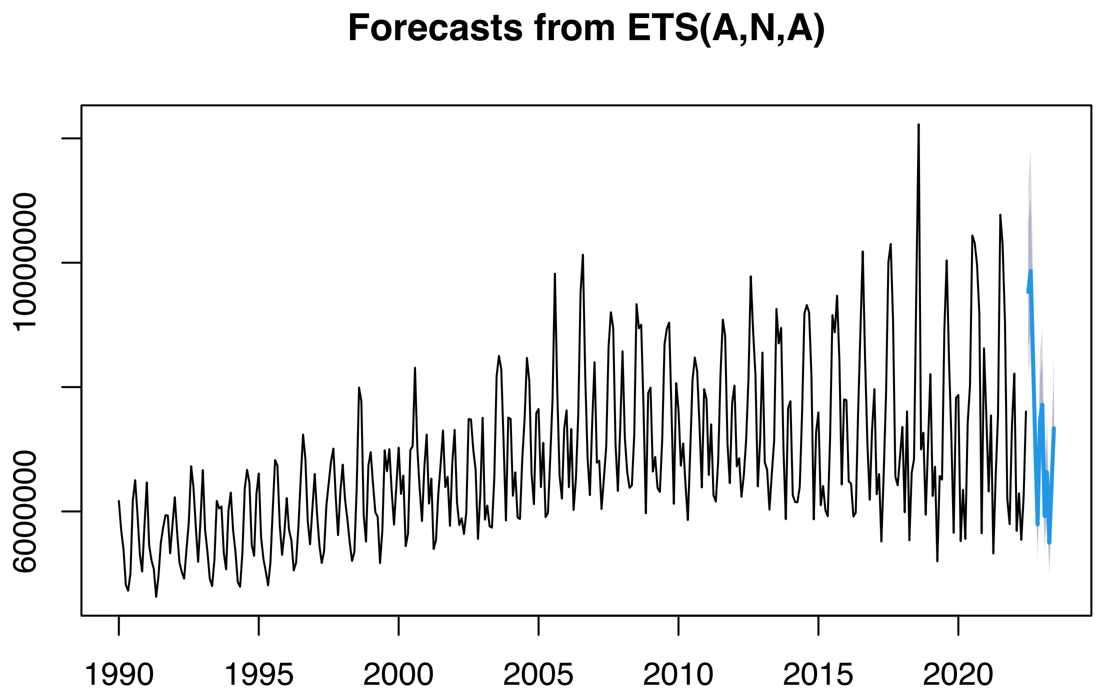
```r
Box.test(ets_model_1990$residuals, lag = 36, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  ets_model_1990$residuals
## X-squared = 71.314, df = 36, p-value = 0.0004106
```

```r
shapiro.test(ets_model_1990$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  ets_model_1990$residuals
## W = 0.99034, p-value = 0.01159
```

```r
plot(forecast(ets_model_1990, h=12))
```



Linear Regression

```
tslm_model_1990 = tslm(train_res_1990 ~ trend + season, lambda = 'auto')
summary(tslm_model_1990)
```

```
##
## Call:
## tslm(formula = train_res_1990 ~ trend + season, lambda = "auto")
##
## Residuals:
##              Min               1Q           Median               3Q              Max
## -0.000000192476 -0.000000034992  0.000000008174  0.000000038508  0.000000120037
##
## Coefficients:
##                     Estimate        Std. Error      t value
## (Intercept)  1.11120066472753  0.00000001122092 99029380.228
## trend        0.00000000062182  0.00000000002595       23.965
## season2     -0.00000012707583  0.00000001420073       -8.949
## season3     -0.00000011417578  0.00000001420081       -8.040
## season4     -0.00000020374290  0.00000001420092      -14.347
## season5     -0.00000017943958  0.00000001420109      -12.636
## season6     -0.00000009246076  0.00000001420130       -6.511
## season7      0.00000003068361  0.00000001431122        2.144
## season8      0.00000007213066  0.00000001431125        5.040
```

```
## season9      0.00000003021425  0.00000001431132       2.111
## season10    -0.00000006818556  0.00000001431144      -4.764
## season11    -0.00000017480338  0.00000001431160     -12.214
## season12    -0.00000004144284  0.00000001431181      -2.896
##                         Pr(>|t|)
## (Intercept) < 0.0000000000000002 ***
## trend       < 0.0000000000000002 ***
## season2     < 0.0000000000000002 ***
## season3       0.0000000000000117 ***
## season4     < 0.0000000000000002 ***
## season5     < 0.0000000000000002 ***
## season6       0.0000000002390218 ***
## season7                   0.0327 *
## season8       0.0000007235618681 ***
## season9                   0.0354 *
## season10      0.0000027075274752 ***
## season11    < 0.0000000000000002 ***
## season12                  0.0040 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.00000005768 on 377 degrees of freedom
## Multiple R-squared:  0.7953, Adjusted R-squared:  0.7888
## F-statistic: 122.1 on 12 and 377 DF,  p-value: < 0.00000000000000022
```

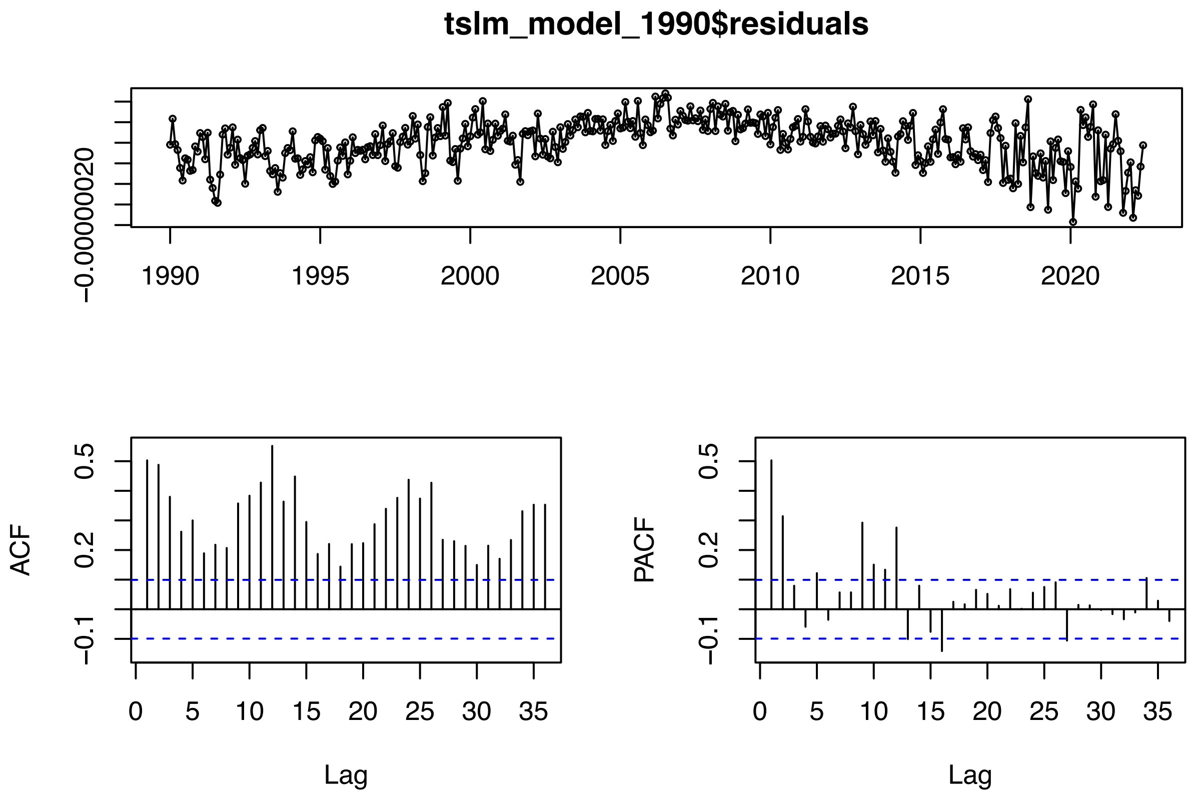
```
forecast::accuracy(tslm_model_1990)
```

```
##                    ME     RMSE      MAE         MPE     MAPE     MASE      ACF1
## Training set 46578.25 574445.5 443042.8 -0.01926594 6.402133 1.124236 0.4616438
```

```
Box.test(tslm_model_1990$residuals, lag = 36, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  tslm_model_1990$residuals
## X-squared = 1565.6, df = 36, p-value < 0.00000000000000022
```

```
tsdisplay(tslm_model_1990$residuals)
```

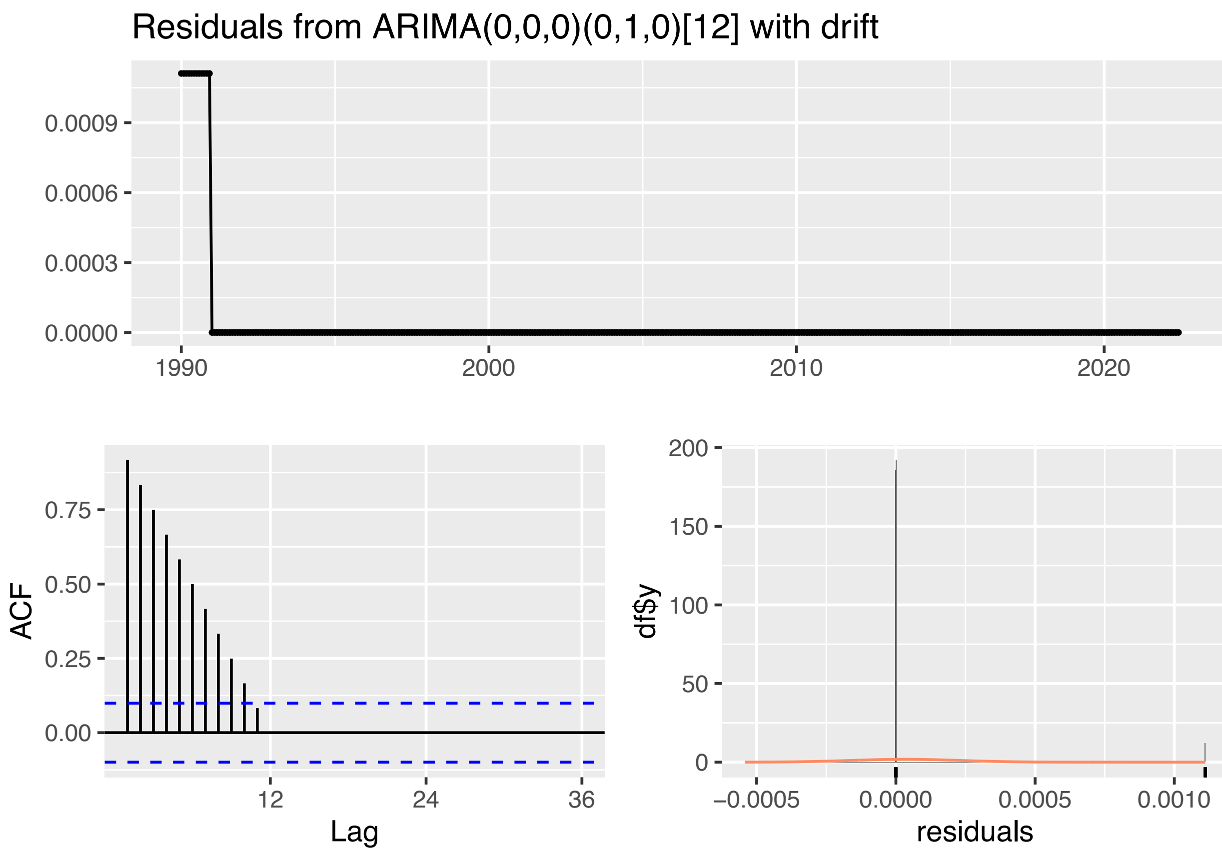


Auto Arima

```r
residential_auto_arima_1990 <- auto.arima(train_res_1990,
                                          seasonal = TRUE,
                                          allowdrift = TRUE,
                                          lambda = 'auto')

summary(residential_auto_arima_1990)
```

```
## Series: train_res_1990
## ARIMA(0,0,0)(0,1,0)[12] with drift
## Box Cox transformation: lambda= -0.8999268
##
## Coefficients:
##        drift
##       0.0000
## s.e.  0.0001
##
## sigma^2 = 0.0000000393:  log likelihood = 5812.67
## AIC=-11621.35   AICc=-11621.32   BIC=-11613.48
##
## Training set error measures:
##                    ME    RMSE      MAE     MPE     MAPE     MASE      ACF1
## Training set 168611.8 1124845 551172.4 2.78575 8.429675 1.398618 0.7081196
```

```r
checkresiduals(residential_auto_arima_1990)
```

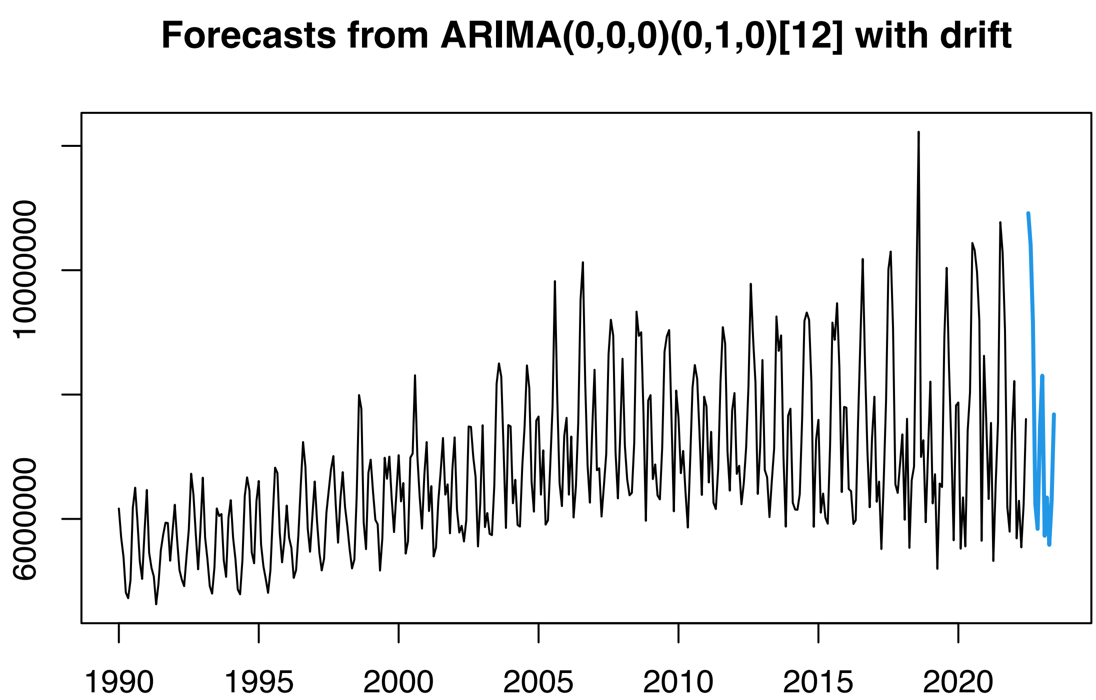



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,0,0)(0,1,0)[12] with drift
## Q* = 1388, df = 24, p-value < 0.0000000000000022
##
## Model df: 0.   Total lags used: 24
```

```r
forecast::accuracy(residential_auto_arima_1990)
```

```
##                    ME    RMSE      MAE     MPE     MAPE     MASE      ACF1
## Training set 168611.8 1124845 551172.4 2.78575 8.429675 1.398618 0.7081196
```

```r
plot(forecast(residential_auto_arima_1990, h=12))
```




```r
Box.test(residential_auto_arima_1990$residuals, lag = 24, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  residential_auto_arima_1990$residuals
## X-squared = 1388, df = 24, p-value < 0.00000000000000022
```

- Residuals are not white noise - can reject Ljung-Box null

With this, we are done establishing some baseline models

Now, we look at correlations of indepenedent variables with consumption

```r
# Correlation plots (if time)

corr_res_avgtemp <- cor(train_res_1990, train_avgtemp_1990)
corr_res_avgtemp
```

```
## [1] 0.398072
```

```r
corr_res_maxtemp <- cor(train_res_1990,train_max_temp_1990)
corr_res_maxtemp
```

```
## [1] 0.3841379
```

```r
corr_res_mintemp <- cor(train_res_1990,train_min_temp_1990)
corr_res_mintemp
```

```
## [1] 0.414023
```

```r
corr_res_cdays <- cor(train_res_1990,train_cdays_1990) # Selected
corr_res_cdays
```

```
## [1] 0.6063661
```

```r
corr_res_hdays <- cor(train_res_1990,train_hdays_1990) # Selected bcs of highest negative correlation
corr_res_hdays
```

```
## [1] -0.3010071
```

```r
corr_res_ngas <- cor(train_res_1990,train_ngas_1990)
corr_res_ngas
```

```
## [1] -0.1975193
```

```r
corr_res_precip <- cor(train_res_1990,train_precip_1990)
corr_res_precip
```

```
## [1] -0.1759033
```

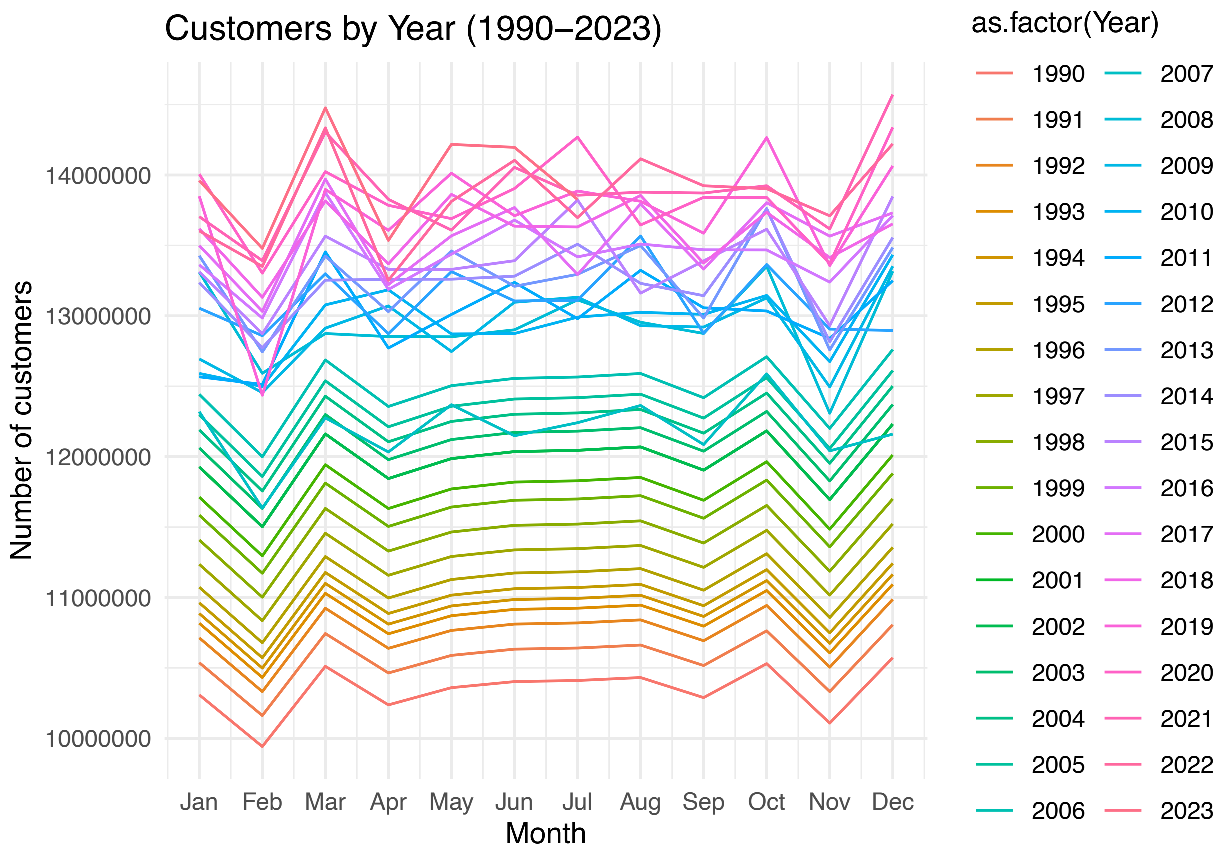
```r
corr_res_customers <- cor(train_res_1990,train_customers_1990) # Selected
corr_res_customers
```

```
## [1] 0.5842694
```

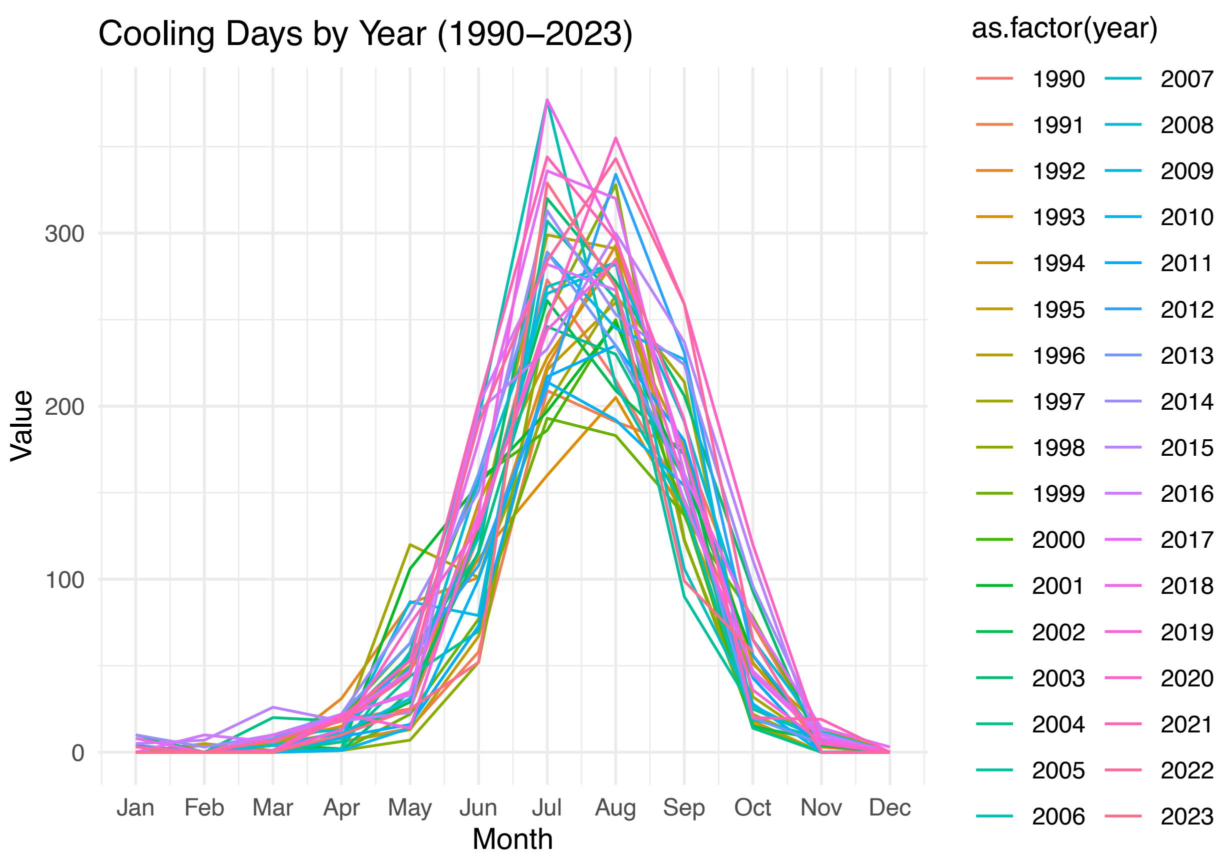
```r
corr_res_elecprices <- cor(train_res_1990,train_elec_prices_1990) # Selected
corr_res_elecprices
```

```
## [1] 0.4547766
```

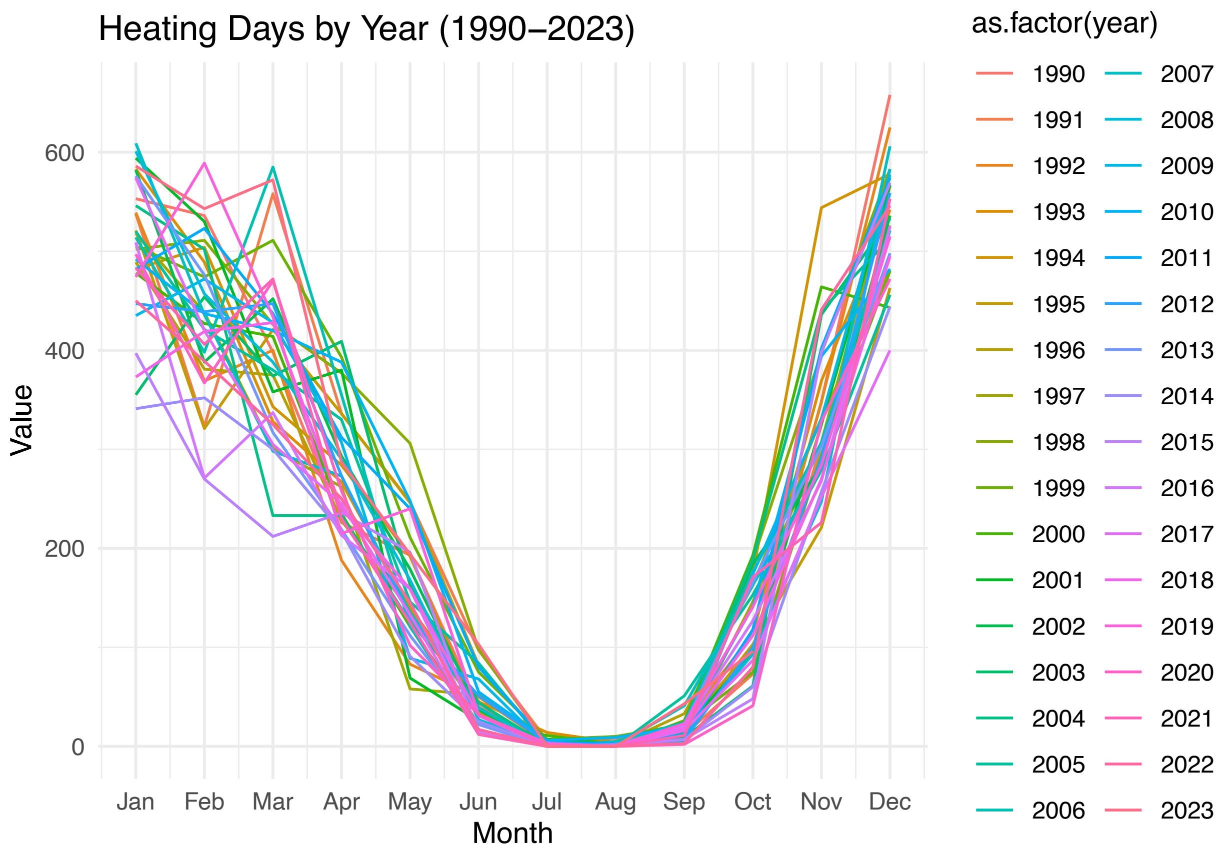
```r
## Plotting independent variables

# Plotting customers
ggplot(customers, aes(x = Month, y = customers, group = Year, color = as.factor(Year))) +
  geom_line() +
  scale_x_continuous(breaks = 1:12, labels = month.abb) +
  labs(title = "Customers by Year (1990-2023)", x = "Month", y = "Number of customers") +
  #scale_y_continuous(limits = c(4e6, 14e6)) +
  theme_minimal()
```




```r
# Plotting cold days
ggplot(cold_days, aes(x = month_num, y = Value, group = year, color = as.factor(year))) +
  geom_line() +
  scale_x_continuous(breaks = 1:12, labels = month.abb) +
  labs(title = "Cooling Days by Year (1990-2023)", x = "Month", y = "Value") +
  #scale_y_continuous(limits = c(4e6, 14e6)) +
  theme_minimal()
```

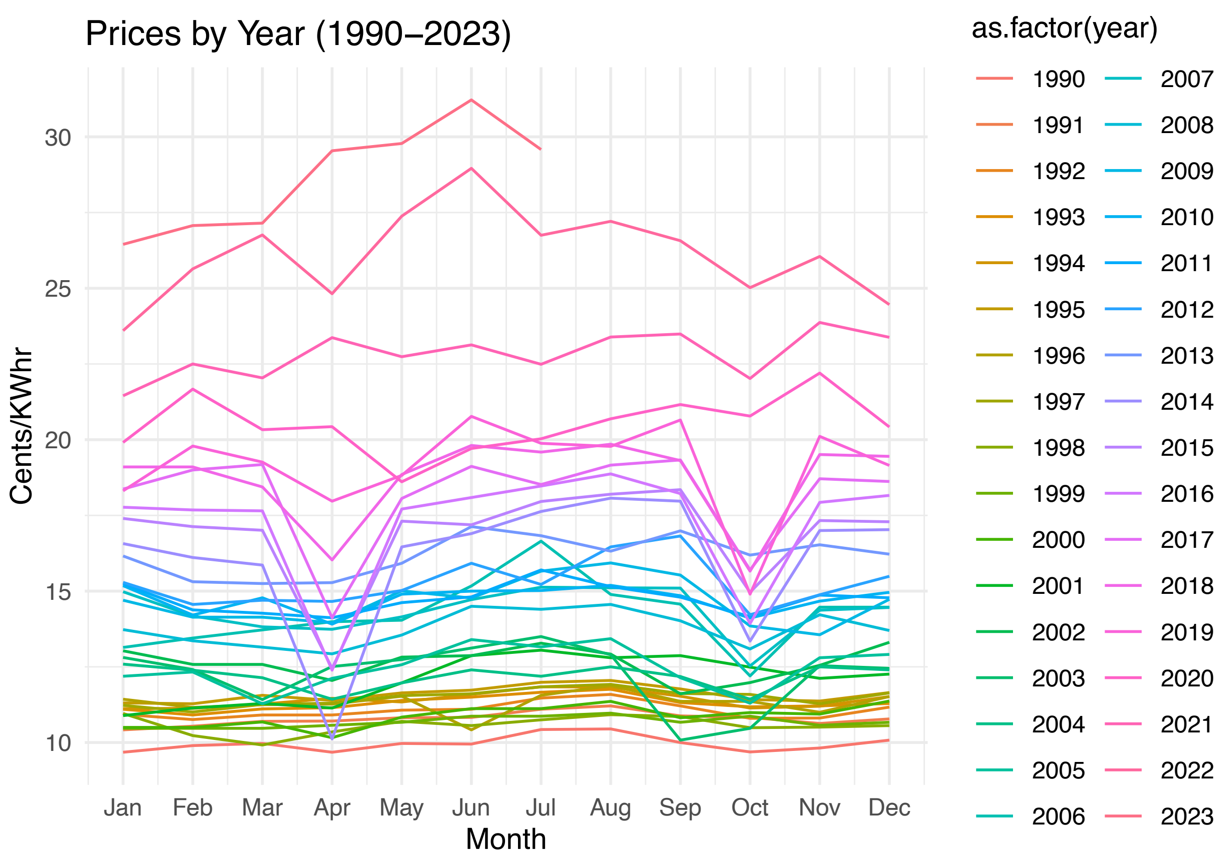



```r
# Plotting hot days
ggplot(hot_days, aes(x = mont_num, y = Value, group = year, color = as.factor(year))) + #spelling error
  geom_line() +
  scale_x_continuous(breaks = 1:12, labels = month.abb) +
  labs(title = "Heating Days by Year (1990-2023)", x = "Month", y = "Value") +
  #scale_y_continuous(limits = c(4e6, 14e6)) +
  theme_minimal()
```

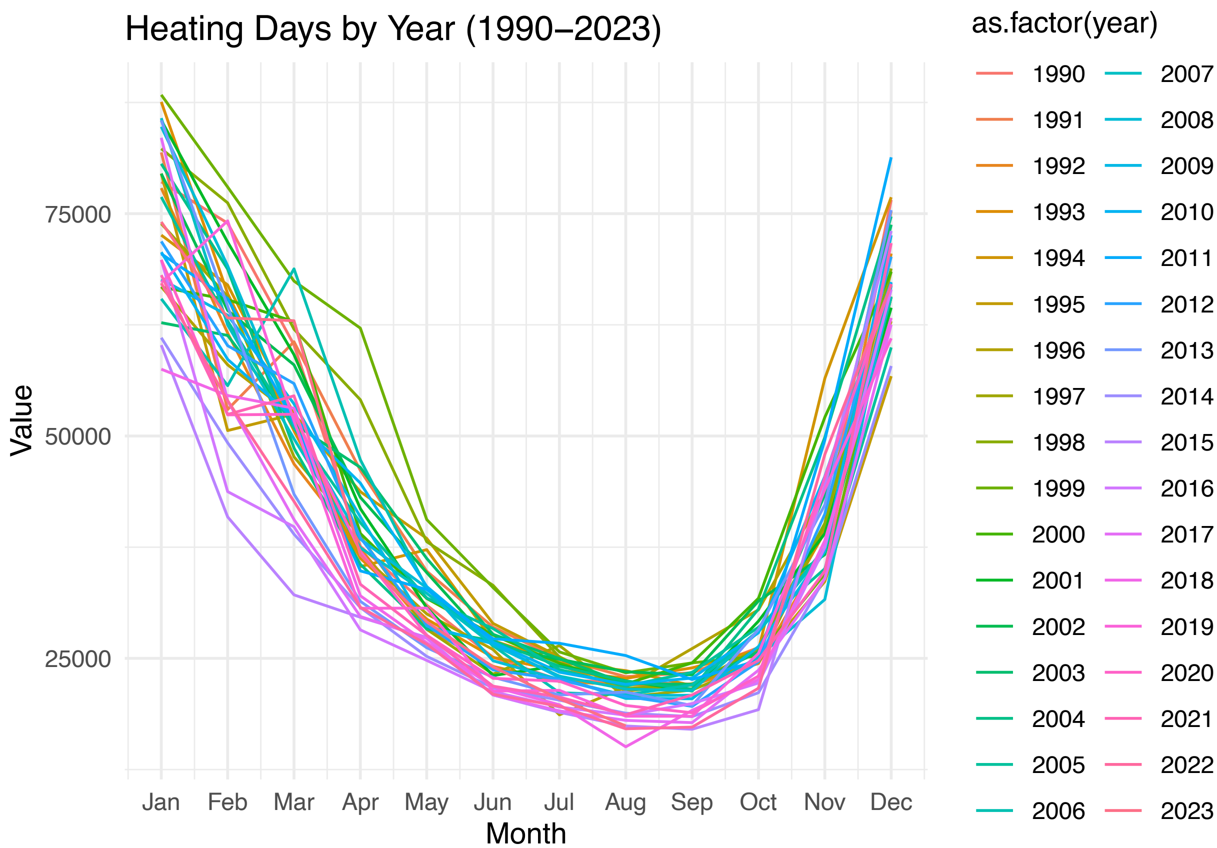



```
# Commenting this out because of knitting error

# For prices
dates <- seq(as.Date("1990-01-01"), as.Date("2023-07-01"), by = "month")
df_prices <- data.frame(date = dates, elec_prices_ts)

# Extract year and month
df_prices$year <- year(df_prices$date)
df_prices$month <- month(df_prices$date)

# Plotting hot days
ggplot(df_prices, aes(x = month, y = elec_prices_ts, group = year, color = as.factor(year))) + #spellin
  geom_line() +
  scale_x_continuous(breaks = 1:12, labels = month.abb) +
  labs(title = "Prices by Year (1990-2023)", x = "Month", y = "Cents/KWhr") +
  #scale_y_continuous(limits = c(4e6, 14e6)) +
  theme_minimal()
```

```
## Don't know how to automatically pick scale for object of type <ts>. Defaulting
## to continuous.
```




```
# Plotting natural gas
ggplot(n_gas, aes(x = num_month, y = Value, group = year, color = as.factor(year))) +
  geom_line() +
  scale_x_continuous(breaks = 1:12, labels = month.abb) +
  labs(title = "Heating Days by Year (1990-2023)", x = "Month", y = "Value") +
  #scale_y_continuous(limits = c(4e6, 14e6)) +
  theme_minimal()
```



Linear Regression with independent variables

```
residential_multiv_reg_1990 <- tslm(train_res_1990 ~ train_customers_1990 + train_elec_prices_1990 +
                                    train_cdays_1990 + train_hdays_1990, lambda = 0)
# Reducing the variables makes me fail the Shapiro-Wilk test for normality
# Removing lambda = auto contributes to Shapiro-Wilk failure
# Removed avg_temp and h_days bcs of negative coefficients - doesn't make intuitive sense

# Store the residuals
residuals_residential_multiv_reg_1990 <- residuals(residential_multiv_reg_1990)

# Store the fitted values
fitted_residential_multiv_reg_1990 <- residential_multiv_reg_1990$fitted.values

summary(residential_multiv_reg_1990)
```

```
##
## Call:
## tslm(formula = train_res_1990 ~ train_customers_1990 + train_elec_prices_1990 +
##     train_cdays_1990 + train_hdays_1990, lambda = 0)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.26004 -0.06186 -0.01084  0.06089  0.37757
```

```
##
## Coefficients:
##                             Estimate      Std. Error t value
## (Intercept)           14.077526851188  0.068125720076 206.640
## train_customers_1990    0.000000129945  0.000000007246  17.934
## train_elec_prices_1990 -0.014845090812  0.002189174990  -6.781
## train_cdays_1990        0.001747236181  0.000075161511  23.246
## train_hdays_1990        0.000494335881  0.000037646053  13.131
##                                  Pr(>|t|)
## (Intercept)           < 0.0000000000000002 ***
## train_customers_1990   < 0.0000000000000002 ***
## train_elec_prices_1990       0.000000000045 ***
## train_cdays_1990       < 0.0000000000000002 ***
## train_hdays_1990       < 0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08762 on 385 degrees of freedom
## Multiple R-squared:  0.7615, Adjusted R-squared:  0.759
## F-statistic: 307.3 on 4 and 385 DF,  p-value: < 0.00000000000000022
```

```
forecast::accuracy(residential_multiv_reg_1990)
```

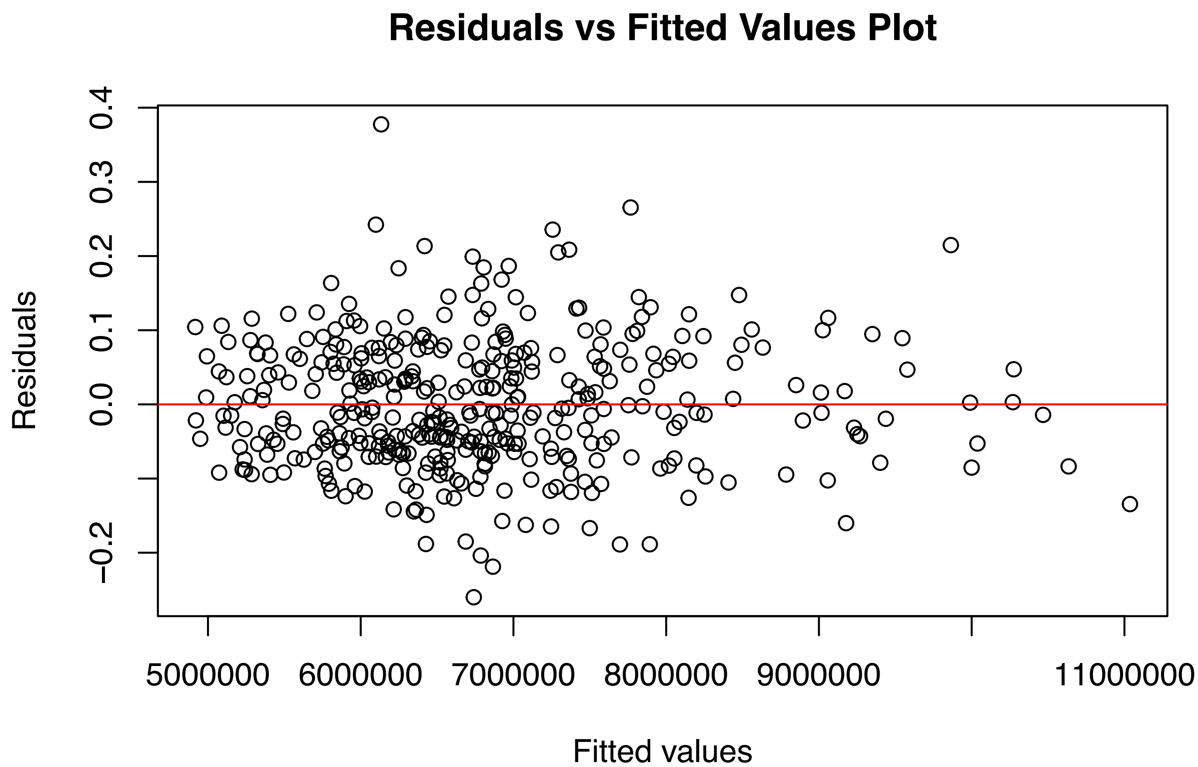
```
##                    ME     RMSE    MAE        MPE     MAPE     MASE      ACF1
## Training set 26510.41 621693.9 483199 -0.3756417 6.994061 1.226134 0.1960032
```

```
Box.test(residential_multiv_reg_1990$residuals, type = "Ljung-Box", lag = 36)
```

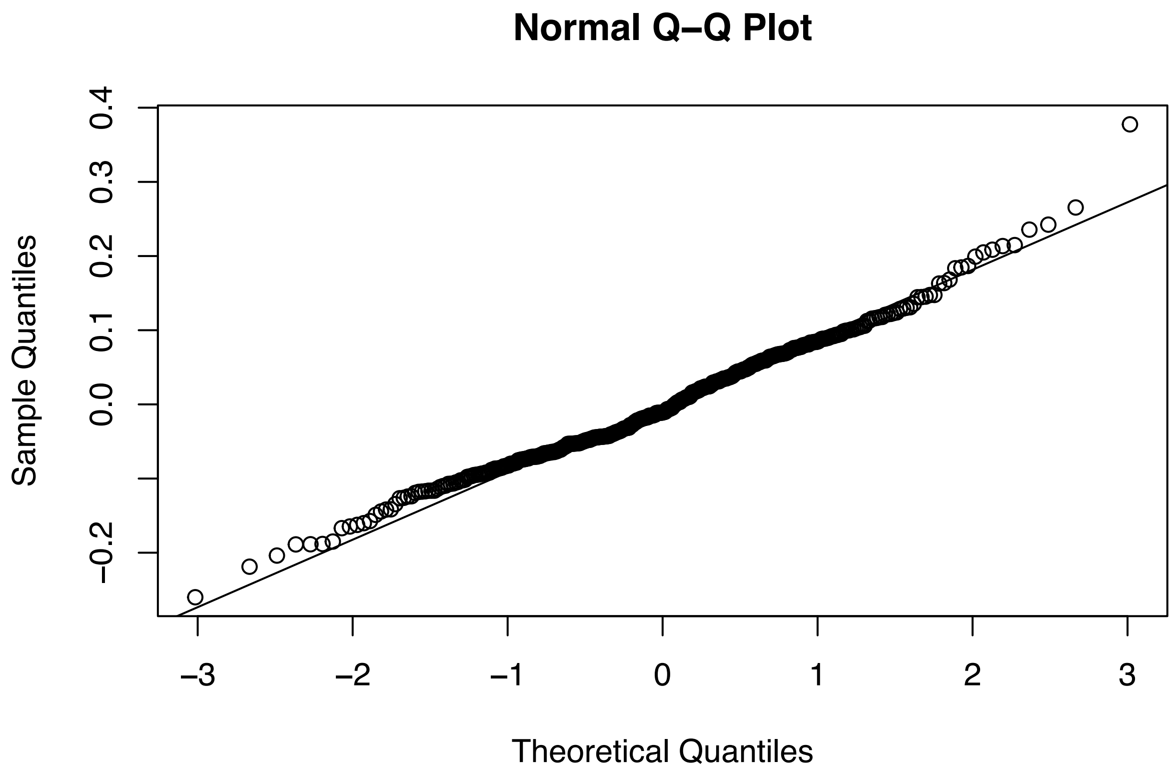
```
##
##  Box-Ljung test
##
## data:  residential_multiv_reg_1990$residuals
## X-squared = 703.22, df = 36, p-value < 0.00000000000000022
```

Baseline regression model analysis

```
## Regression analysis: potential to create a function that does all of this for any given regression

# First, let's plot residuals vs. fitted values
plot(fitted_residential_multiv_reg_1990, residuals_residential_multiv_reg_1990,
     xlab = "Fitted values", ylab = "Residuals",
     main = "Residuals vs Fitted Values Plot")
abline(h = 0, col = "red")  # Add a horizontal line at y = 0 (for reference)
```



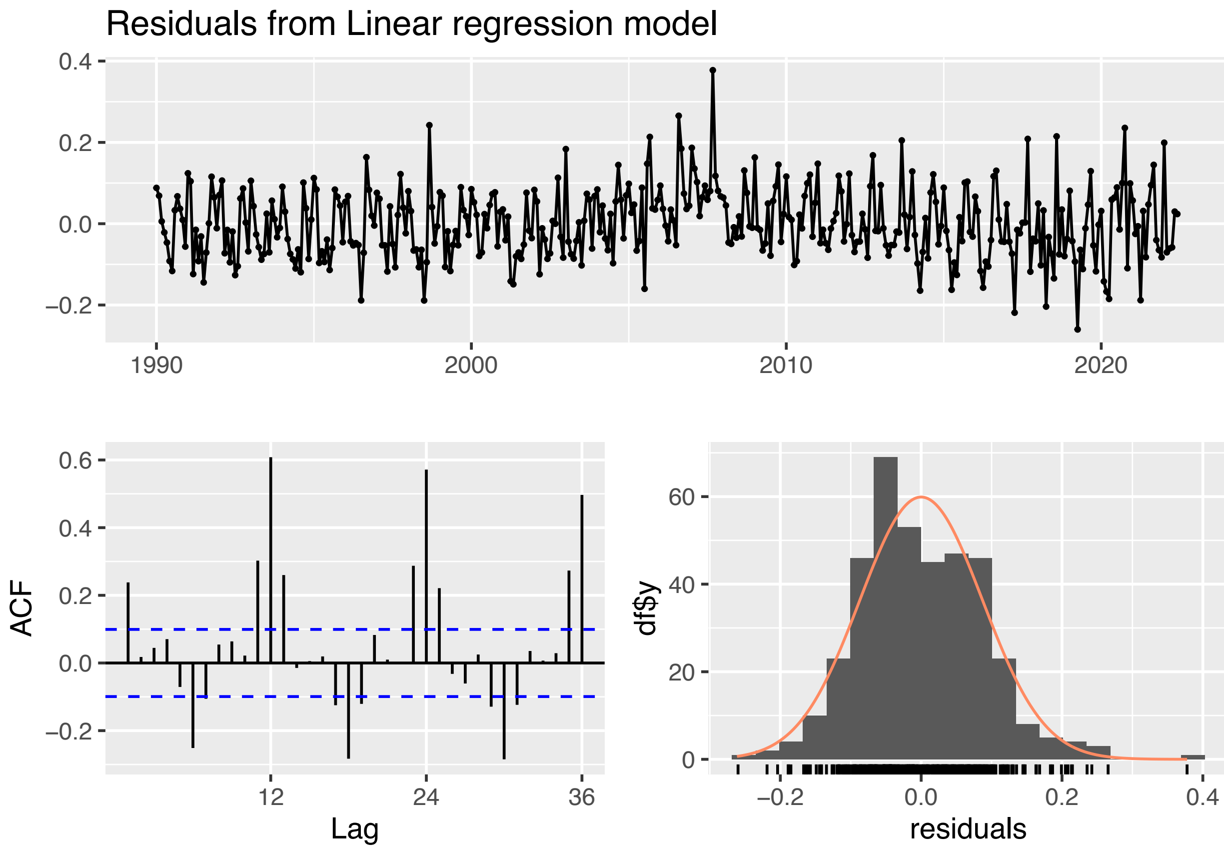

```r
# What conclusion can be drawn from this? -> Residual values are pretty low, which is good I guess?

# Now, we can create a Q-Q plot for residuals
qqnorm(residuals_residential_multiv_reg_1990)
qqline(residuals_residential_multiv_reg_1990)
```




```r
# Shapiro-Wilk normality test
shapiro.test(residuals_residential_multiv_reg_1990)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals_residential_multiv_reg_1990
## W = 0.98799, p-value = 0.002646
```

```r
# We can also check the ACF plot for residuals
checkresiduals(residential_multiv_reg_1990)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 24
##
## data:  Residuals from Linear regression model
## LM test = 209.88, df = 24, p-value < 0.00000000000000022
```

Overall takeaways from baseline model:

- Spurious correlation alert for h_days and prices (different signs for
- correlation and regression coefficients)
- Adjusted R^2? - Almost half the variation can be explained by these variables
- Plot of residuals vs. fitted values does not convey anything clearly
- QQ plot indicates significant deviation from normality; confirmed by Shapiro-Wilk test
- Breusch-Godfrey test and ACF plots also show clear auto-correlations in residuals

```
# Spurious correlation alert for h_days and prices (different signs for
# correlation and regression coefficients)

# One way to deal with this is add a square term; let's try that

sq_train_hdays_1990 <- (train_hdays_1990)^2
sq_train_elec_prices_1990 <- (train_elec_prices_1990)^2


# Re-running with these terms
```

```r
residential_multiv_reg_sq_1990 <- tslm(train_res_1990 ~ train_customers_1990 +
                                          sq_train_elec_prices_1990 +
                                       train_elec_prices_1990 + train_cdays_1990
                                       + sq_train_hdays_1990 +
                                       train_hdays_1990, lambda = 0)

# Reducing the variables makes me fail the Shapiro-Wilk test for normality
# Removing lambda = auto contributes to Shapiro-Wilk failure
# Removed avg_temp and h_days bcs of negative coefficients - doesn't make intuitive sense

# Store the residuals
residuals_residential_multiv_reg_sq_1990 <- residuals(residential_multiv_reg_sq_1990)

summary(residential_multiv_reg_sq_1990)
```

```
##
## Call:
## tslm(formula = train_res_1990 ~ train_customers_1990 + sq_train_elec_prices_1990 +
##     train_elec_prices_1990 + train_cdays_1990 + sq_train_hdays_1990 +
##     train_hdays_1990, lambda = 0)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.25177 -0.05451 -0.00931  0.05654  0.32394
##
## Coefficients:
##                               Estimate     Std. Error t value
## (Intercept)               14.15657237172  0.07473302415 189.429
## train_customers_1990       0.00000011979  0.00000000861  13.913
## sq_train_elec_prices_1990 -0.00052534445  0.00031046358  -1.692
## train_elec_prices_1990     0.00592508846  0.01163309166   0.509
## train_cdays_1990           0.00118143323  0.00011499236  10.274
## sq_train_hdays_1990        0.00000137714  0.00000022348   6.162
## train_hdays_1990          -0.00045432812  0.00015734894  -2.887
##                                       Pr(>|t|)
## (Intercept)               < 0.0000000000000002 ***
## train_customers_1990      < 0.0000000000000002 ***
## sq_train_elec_prices_1990               0.0914 .
## train_elec_prices_1990                  0.6108
## train_cdays_1990          < 0.0000000000000002 ***
## sq_train_hdays_1990              0.00000000182 ***
## train_hdays_1990                        0.0041 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08359 on 383 degrees of freedom
## Multiple R-squared:  0.7841, Adjusted R-squared:  0.7807
## F-statistic: 231.8 on 6 and 383 DF,  p-value: < 0.00000000000000022
```

```r
forecast::accuracy(residential_multiv_reg_sq_1990)
```

```
##                    ME     RMSE      MAE        MPE     MAPE     MASE      ACF1
## Training set 25145.32 589745.2 458675.3 -0.3411913 6.648508 1.163904 0.2141602
```

```r
Box.test(residential_multiv_reg_sq_1990$residuals, type = "Ljung-Box", lag = 36)
```

```
##
##  Box-Ljung test
##
## data:  residential_multiv_reg_sq_1990$residuals
## X-squared = 538.95, df = 36, p-value < 0.00000000000000022
```

Idea of adding additional variables doesn't work. Decision - remove h_days and prices

But we do add these in regression with ARIMA errors

Adding trend and seasonality along with other variables

```r
residential_ts_multiv_reg_1990 <- tslm(train_res_1990 ~ trend + season +
                                         train_customers_1990 +
                                          + train_cdays_1990 + train_hdays_1990,
                                       lambda = 0)


# Store the residuals
residuals_residential_ts_multiv_reg_1990 <- residuals(residential_ts_multiv_reg_1990)

# Store the fitted values
fitted_residential_ts_multiv_reg_1990 <- residential_ts_multiv_reg_1990$fitted.values

summary(residential_ts_multiv_reg_1990)
```

```
##
## Call:
## tslm(formula = train_res_1990 ~ trend + season + train_customers_1990 +
##     +train_cdays_1990 + train_hdays_1990, lambda = 0)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.19715 -0.03907  0.00026  0.03830  0.25561
##
## Coefficients:
##                          Estimate     Std. Error t value             Pr(>|t|)
## (Intercept)          13.53617743286  0.19597705185  69.070 < 0.0000000000000002
## trend                -0.00092296496  0.00017831502  -5.176       0.000000370752
## season2              -0.07032058867  0.01901149644  -3.699             0.000249
## season3              -0.16502544331  0.01871764146  -8.817 < 0.0000000000000002
## season4              -0.18990612582  0.02193596339  -8.657 < 0.0000000000000002
## season5              -0.18114014715  0.02770462472  -6.538       0.000000000204
## season6              -0.13860049072  0.03495308910  -3.965       0.000087831403
## season7              -0.10741194580  0.04375680798  -2.455             0.014553
## season8              -0.04411074428  0.04397801619  -1.003             0.316501
## season9               0.02714741574  0.03753448486   0.723             0.469969
```

```
## season10            -0.05713364096  0.02990206599  -1.911             0.056809
## season11            -0.12340076320  0.02087787088  -5.911       0.000000007681
## season12            -0.10696618346  0.01723515840  -6.206       0.000000001441
## train_customers_1990  0.00000018512  0.00000001858   9.965 < 0.0000000000000002
## train_cdays_1990      0.00125498686  0.00012876979   9.746 < 0.0000000000000002
## train_hdays_1990      0.00034553514  0.00006698733   5.158       0.000000405237
##
## (Intercept)          ***
## trend                ***
## season2              ***
## season3              ***
## season4              ***
## season5              ***
## season6              ***
## season7              *
## season8
## season9
## season10             .
## season11             ***
## season12             ***
## train_customers_1990 ***
## train_cdays_1990     ***
## train_hdays_1990     ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06727 on 374 degrees of freedom
## Multiple R-squared:  0.8635, Adjusted R-squared:  0.858
## F-statistic: 157.7 on 15 and 374 DF,  p-value: < 0.00000000000000022
```

```
forecast::accuracy(residential_ts_multiv_reg_1990)
```

```
##                    ME     RMSE      MAE        MPE    MAPE      MASE      ACF1
## Training set 16455.82 474536.4 344108.1 -0.2169492 4.97718 0.8731858 0.2050652
```

```
Box.test(residential_ts_multiv_reg_1990$residuals, type = "Ljung-Box", lag = 36)
```

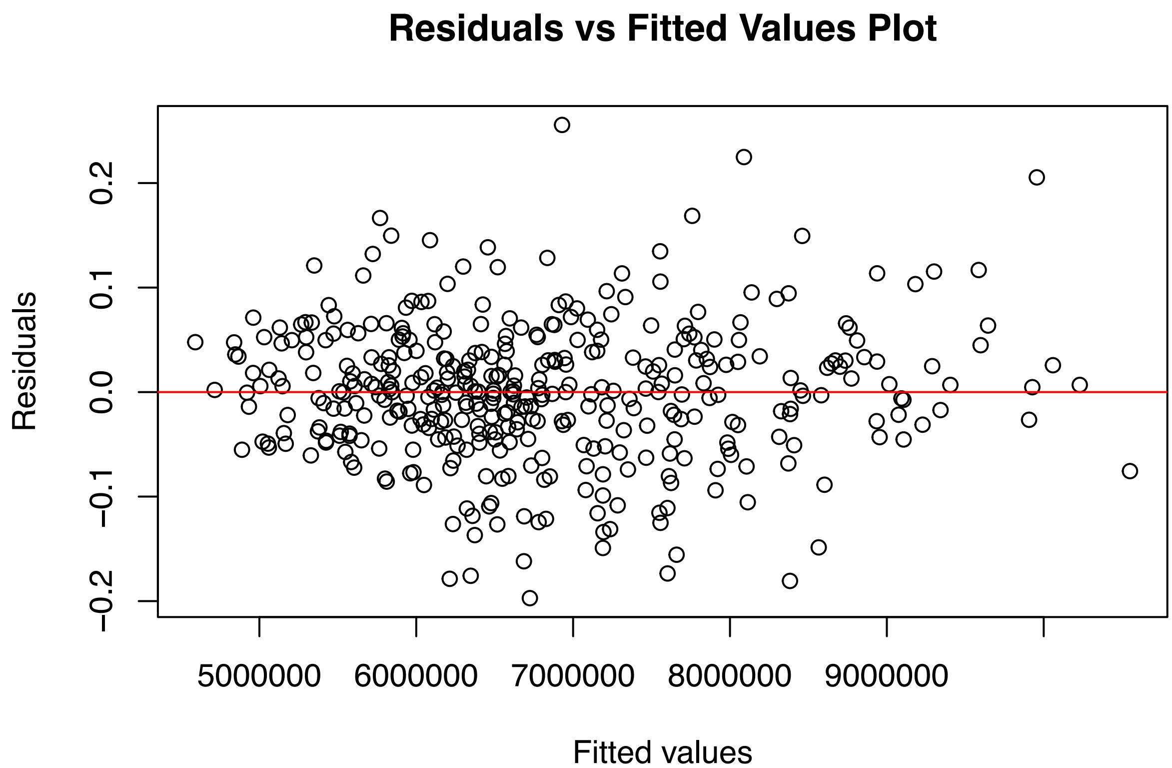
```
##
##  Box-Ljung test
##
## data:  residential_ts_multiv_reg_1990$residuals
## X-squared = 481.89, df = 36, p-value < 0.00000000000000022
```

```
kpss.test(residential_ts_multiv_reg_1990$residuals, "Level")
```

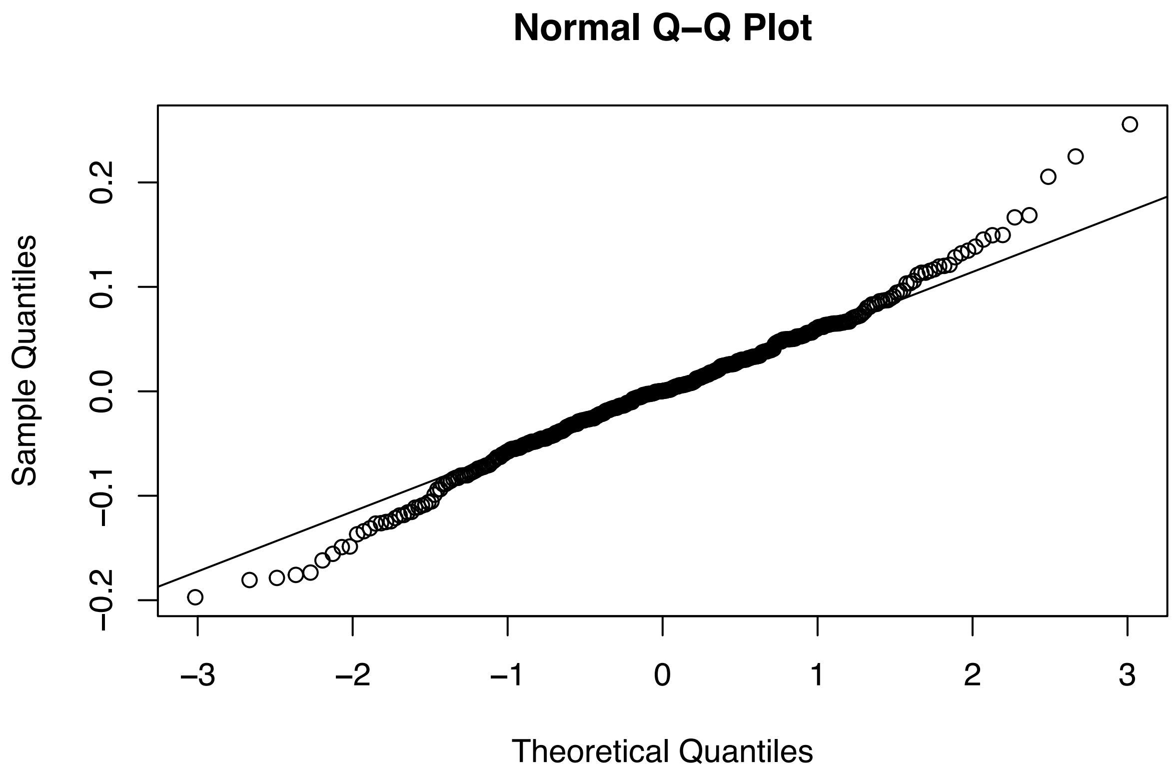
```
##
##  KPSS Test for Level Stationarity
##
## data:  residential_ts_multiv_reg_1990$residuals
## KPSS Level = 0.72394, Truncation lag parameter = 5, p-value = 0.01137
```

Baseline regression with trend and seasonality model analysis

```
# First, let's plot residuals vs. fitted values
plot(fitted_residential_ts_multiv_reg_1990, residuals_residential_ts_multiv_reg_1990,
     xlab = "Fitted values", ylab = "Residuals",
     main = "Residuals vs Fitted Values Plot")
abline(h = 0, col = "red")  # Add a horizontal line at y = 0 (for reference)
```



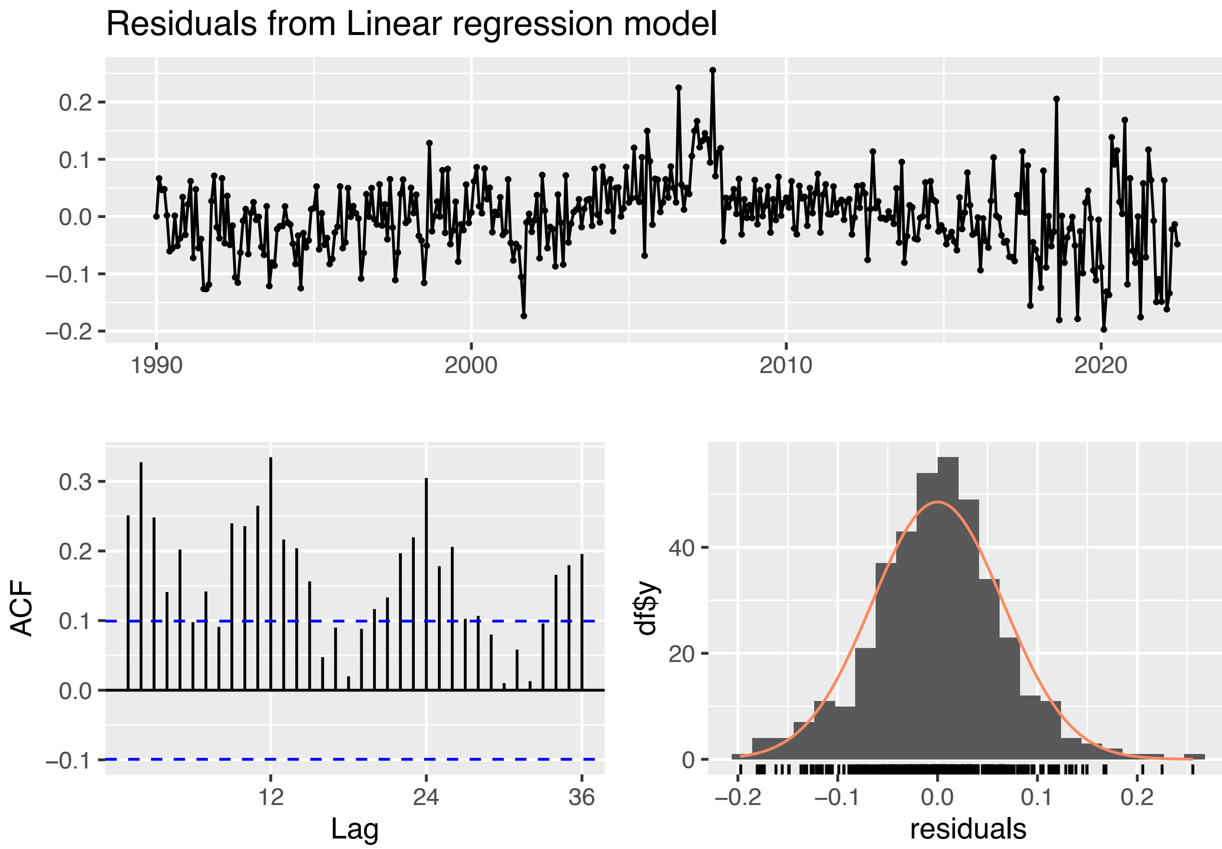

```
# Now, we can create a Q-Q plot for residuals
qqnorm(residuals_residential_ts_multiv_reg_1990)
qqline(residuals_residential_ts_multiv_reg_1990)
```




```r
# Shapiro-Wilk normality test
shapiro.test(residuals_residential_ts_multiv_reg_1990)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals_residential_ts_multiv_reg_1990
## W = 0.98835, p-value = 0.003301
```

```r
# We can also check the ACF plot for residuals
checkresiduals(residential_ts_multiv_reg_1990)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 24
##
## data:  Residuals from Linear regression model
## LM test = 128.74, df = 24, p-value = 0.0000000000000002617
```

What changed?

- Residuals seem to be normally distributed - Not anymore
- However, they still have serial autocorrelation
- From this analysis, we can however narrow down to the important variables:
  - trend, season, cold_days, prices and customers

Potential next steps for model building:

- To forecast independent variables, use seasonal naive
- Megan already has Regression with ARIMA error, just make all variables stationary before running the models
- Check if residuals still show auto-correlation
- Check accuracy measures on train and test
- Plot forecasts with confidence intervals
- With this, our regression analysis should be complete

REGRESSION WITH ARIMA ERRORS

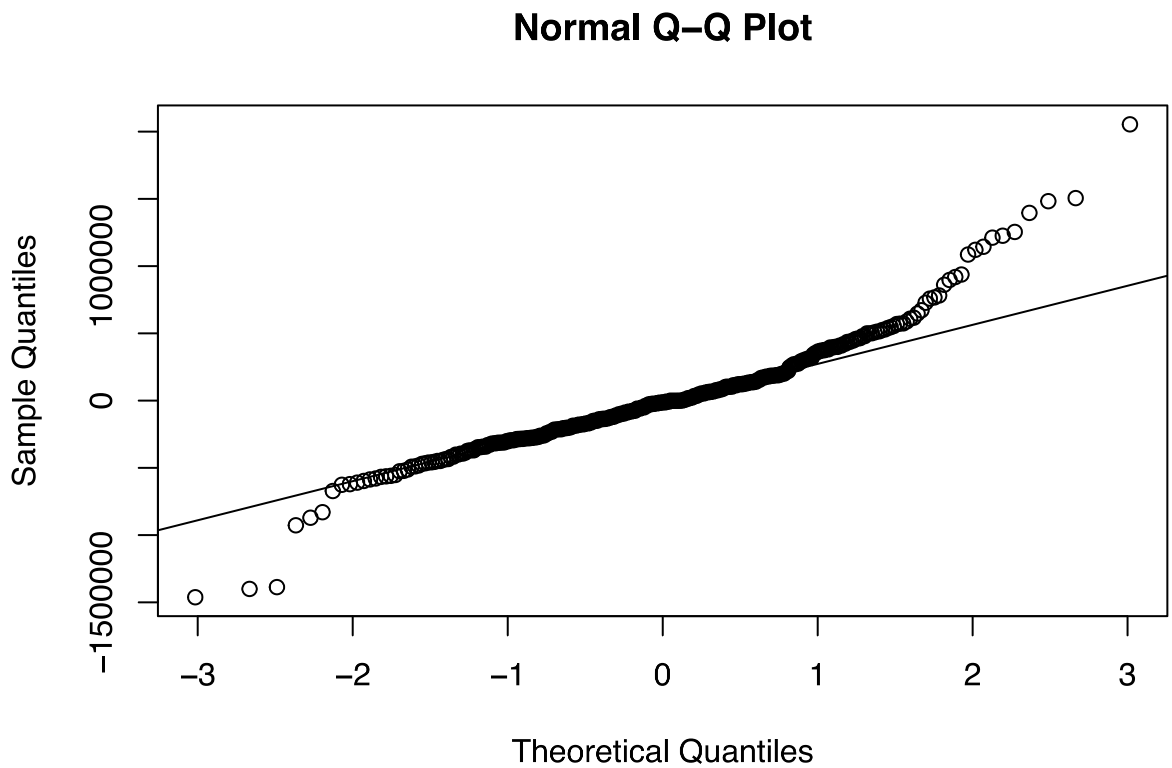
```r
# Regression with ARIMA on the errors
residential_reg_w_arima_err_1990 <- auto.arima(train_res_1990,stationary = FALSE,
                                              seasonal = TRUE,
                                              stepwise = TRUE,trace = FALSE,
                                              xreg = cbind(train_customers_1990,
                                                           train_elec_prices_1990^2,
                                                           train_elec_prices_1990,
                                       train_cdays_1990,train_cdays_1990^2,
                                       train_hdays_1990, train_hdays_1990^2))




# Store the residuals
residuals_residential_reg_w_arima_1990 <- residuals(residential_reg_w_arima_err_1990)
forecast::accuracy(residential_reg_w_arima_err_1990)
```

```
##                    ME   RMSE    MAE        MPE     MAPE      MASE        ACF1
## Training set 12944.95 396788 278234 -0.1452337 3.949165 0.7060282 0.006119146
```

```r
summary(residential_reg_w_arima_err_1990)
```

```
## Series: train_res_1990
## Regression with ARIMA(3,0,2)(0,1,1)[12] errors
##
## Coefficients:
##           ar1     ar2     ar3     ma1      ma2     sma1  train_customers_1990
##       -0.2009  0.8312  0.2236  0.1580  -0.7724  -0.6466                0.5413
## s.e.   0.0945  0.0609  0.0605  0.0904   0.0849   0.0435                0.1166
##       train_elec_prices_1990^2  train_elec_prices_1990  train_cdays_1990
##                      -1878.676                32187.37          4420.020
## s.e.                  2120.791                78284.27          3230.636
##       train_cdays_1990^2  train_hdays_1990  train_hdays_1990^2
##                   6.2961         -1886.311              4.0898
## s.e.              6.5695          2082.337              2.3521
##
## sigma^2 = 168224326500:  log likelihood = -5418.82
## AIC=10865.63   AICc=10866.79   BIC=10920.72
##
## Training set error measures:
##                    ME   RMSE    MAE        MPE     MAPE      MASE        ACF1
## Training set 12944.95 396788 278234 -0.1452337 3.949165 0.7060282 0.006119146
```

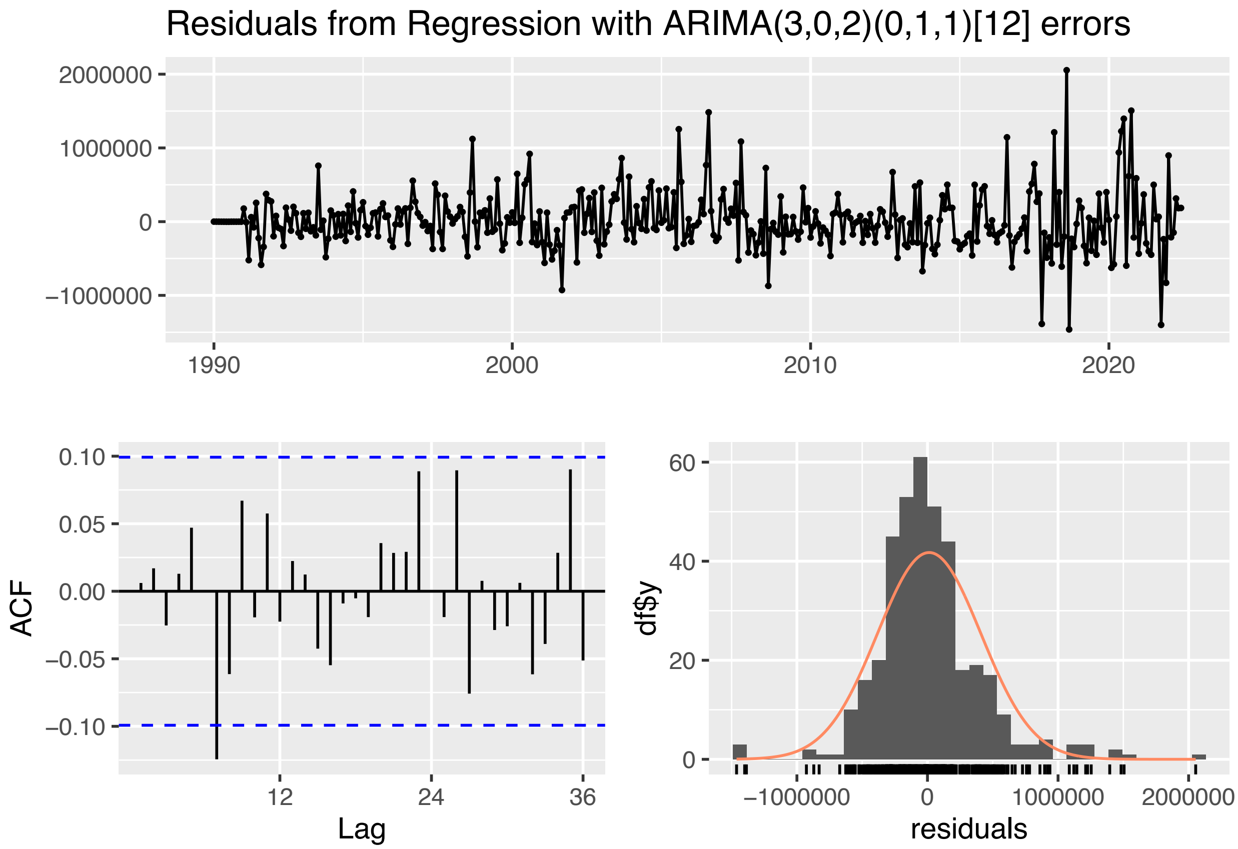
```r
# Model Analysis

# Q-Q plot for residuals
qqnorm(residuals_residential_reg_w_arima_1990)
qqline(residuals_residential_reg_w_arima_1990)
```




```r
# Shapiro-Wilk normality test
shapiro.test(residuals_residential_reg_w_arima_1990)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals_residential_reg_w_arima_1990
## W = 0.93237, p-value = 0.000000000002675
```

```r
# We can also check the ACF plot for residuals
checkresiduals(residential_reg_w_arima_err_1990)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(3,0,2)(0,1,1)[12] errors
## Q* = 19.428, df = 18, p-value = 0.3659
##
## Model df: 6.   Total lags used: 24
```

```
Box.test(residential_reg_w_arima_err_1990$residuals, lag = 36, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  residential_reg_w_arima_err_1990$residuals
## X-squared = 33.298, df = 36, p-value = 0.5978
```

Passed Ljung-Box but failed Shapiro-Wilk test

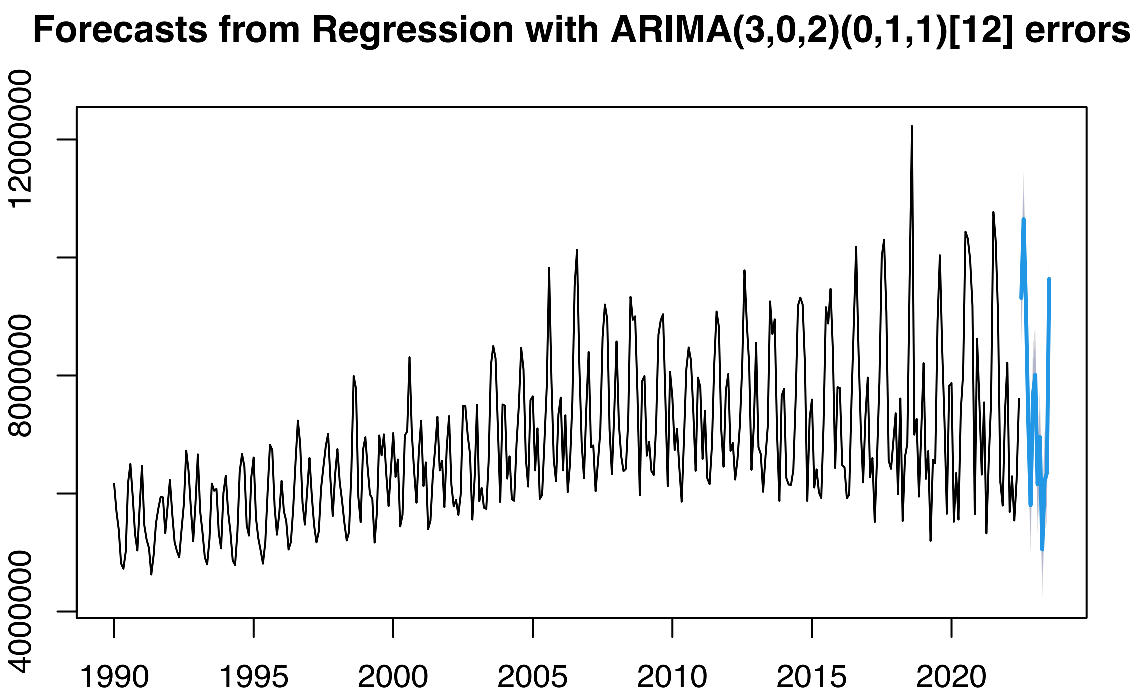
i.e. residuals do not have auto-correlation but are not normally distributed

```
## Using train values of independent variables to forecast consumption

fcast_residential_reg_w_arima_err_1990 <- forecast(residential_reg_w_arima_err_1990,
```

```r
                                                    xreg = cbind(test_customers_1990, test_elec_prices_199
                                                                ,test_elec_prices_1990,
                                            test_cdays_1990,test_cdays_1990^2, test_hdays_1990, test_hdays_199
```

```
## Warning in forecast.forecast_ARIMA(residential_reg_w_arima_err_1990, xreg =
## cbind(test_customers_1990, : xreg contains different column names from the xreg
## used in training. Please check that the regressors are in the same order.
```

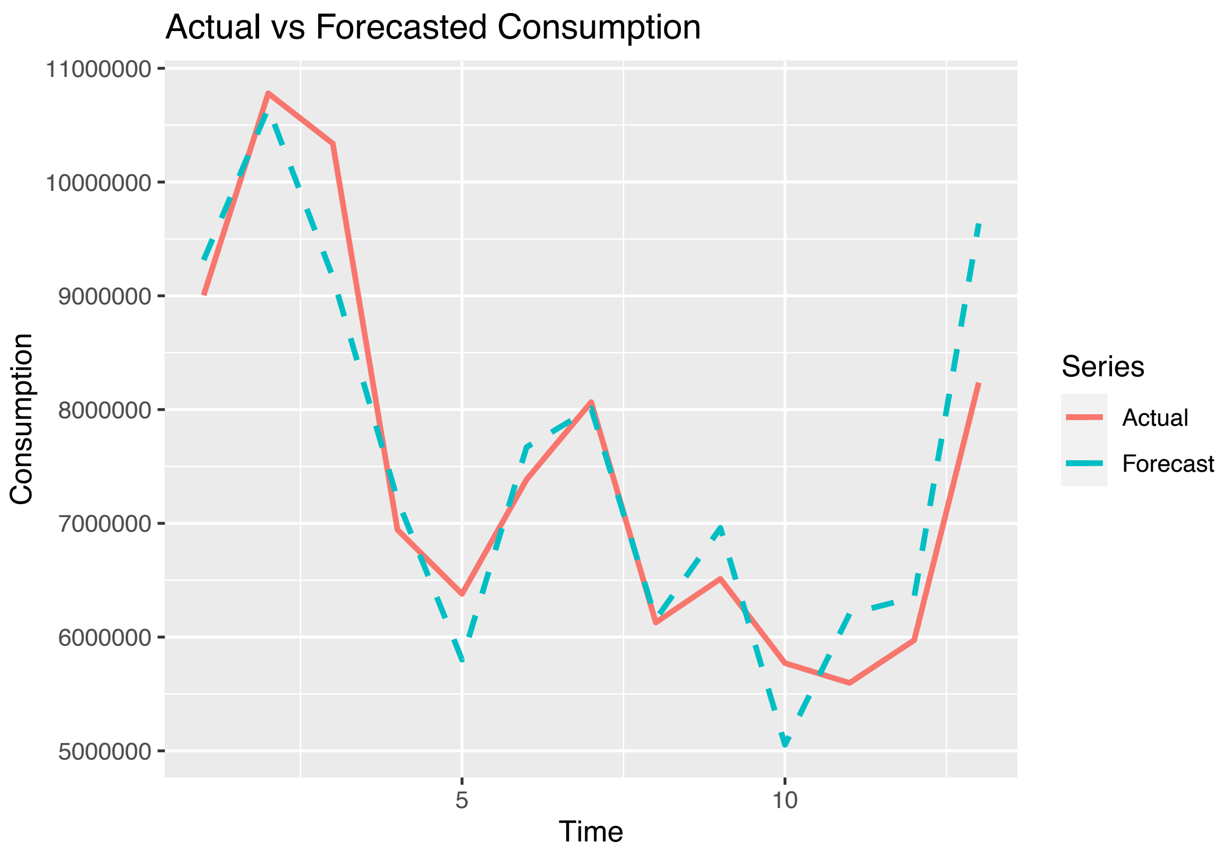
```r
plot(fcast_residential_reg_w_arima_err_1990)
```




```r
mean_fcast_residential_reg_w_arima_err_1990 <- fcast_residential_reg_w_arima_err_1990$mean

df_comparison <- data.frame(
  Time = seq_along(test_res_1990),
  Actual = test_res_1990,
  Forecast = mean_fcast_residential_reg_w_arima_err_1990
)

ggplot(df_comparison, aes(x = Time)) +
  geom_line(aes(y = Actual, color = "Actual"), size = 1) +
  geom_line(aes(y = Forecast, color = "Forecast"), size = 1, linetype = "dashed") +
  labs(x = "Time", y = "Consumption", color = "Series") +
  ggtitle("Actual vs Forecasted Consumption")
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## Don't know how to automatically pick scale for object of type <ts>. Defaulting
## to continuous.
```

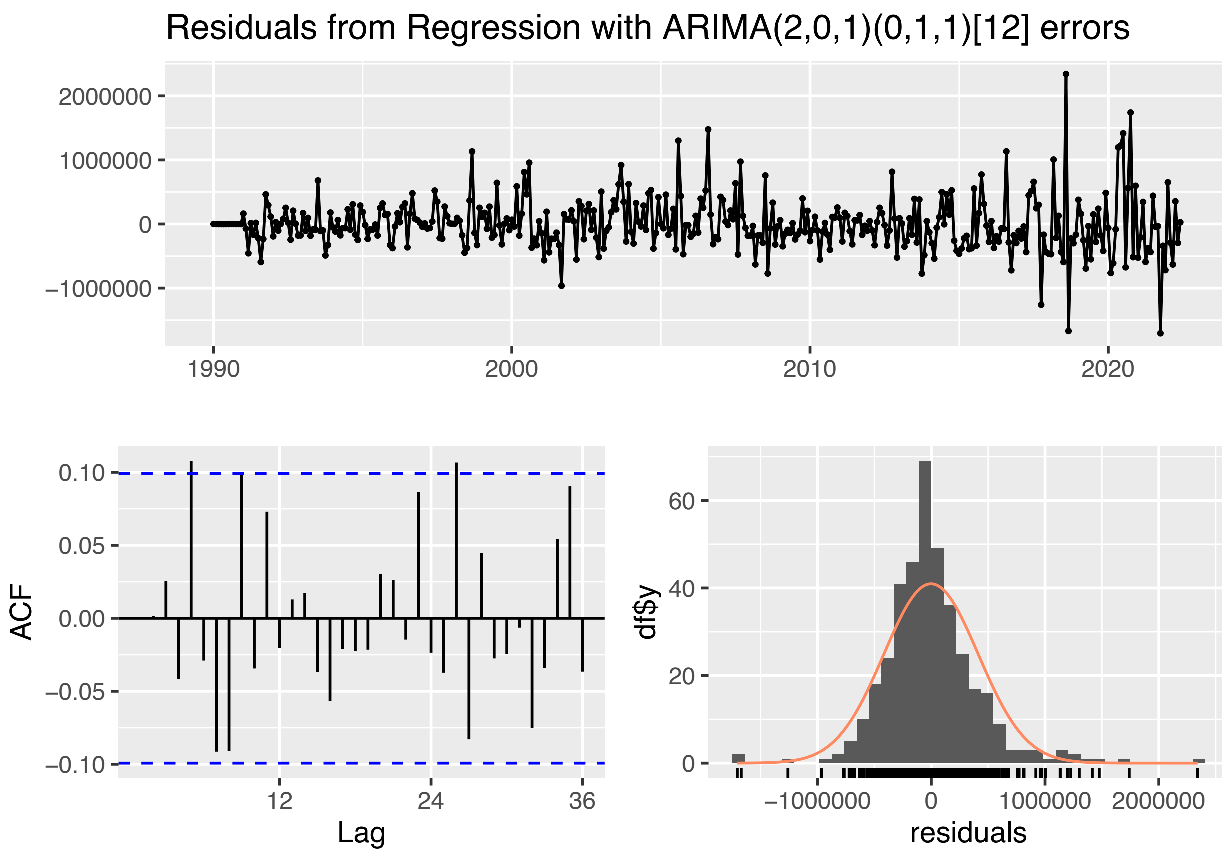



```r
forecast::accuracy(fcast_residential_reg_w_arima_err_1990, test_res_1990)
```

```
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set  12944.95 396788.0 278234.0 -0.1452337 3.949165 0.7060282
## Test set     -81824.22 630133.5 490227.6 -1.3722630 6.718602 1.2439688
##                     ACF1 Theil's U
## Training set  0.006119146        NA
## Test set     -0.093951086 0.5159823
```

```r
## Another regression with ARIMA errors: decision based on standard errors:
# Regression with ARIMA on the errors

residential_reg_w_arima_err_v2_1990 <- auto.arima(train_res_1990,stationary = FALSE,
                                          seasonal = TRUE, # FALSE restricts to non-seasonal model
                                          stepwise = TRUE,trace = FALSE,
```

```r
                                                  xreg = cbind(train_customers_1990,
                                                               train_cdays_1990^2,
                                                               train_hdays_1990^2)
                                                  )


checkresiduals(residential_reg_w_arima_err_v2_1990)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(2,0,1)(0,1,1)[12] errors
## Q* = 26.01, df = 20, p-value = 0.1655
##
## Model df: 4.   Total lags used: 24
```

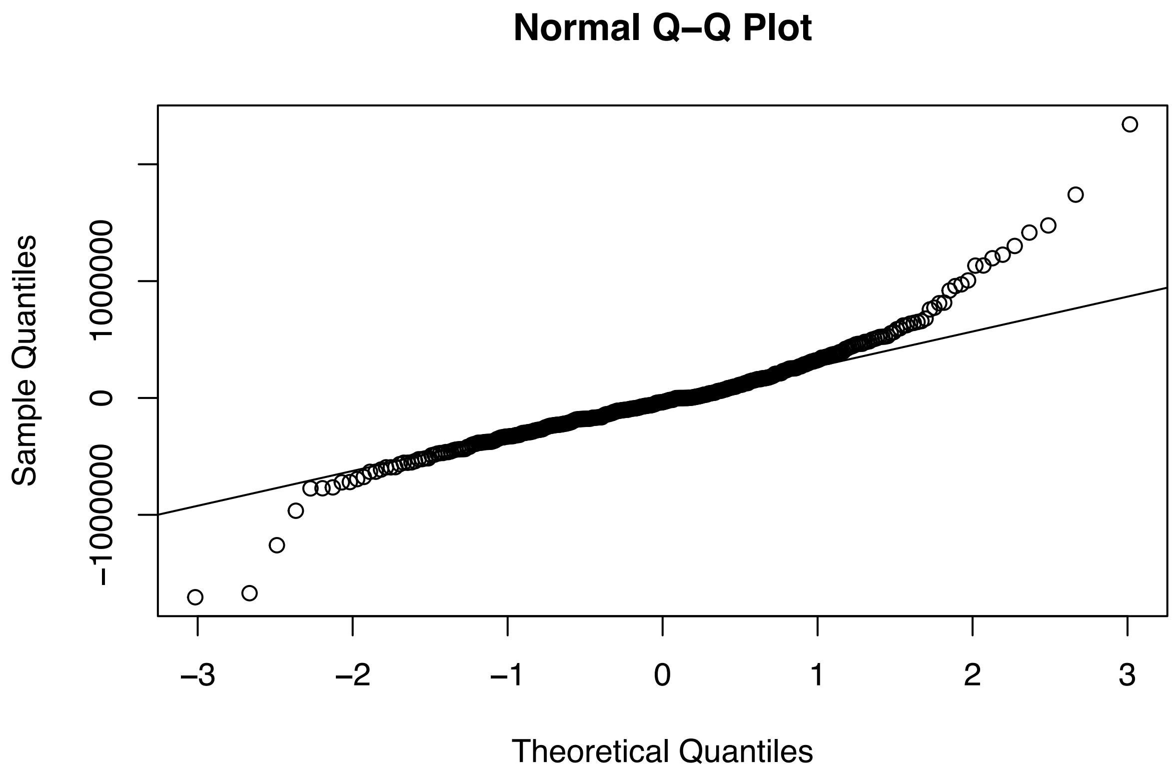
```r
# Performance
forecast::accuracy(residential_reg_w_arima_err_v2_1990)
```

```
##                     ME     RMSE    MAE        MPE     MAPE      MASE
## Training set -1926.627 414933.3 288317 -0.3965911 4.094071 0.7316139
##                       ACF1
## Training set -0.0001951116
```

```r
summary(residential_reg_w_arima_err_v2_1990)
```

```
## Series: train_res_1990
## Regression with ARIMA(2,0,1)(0,1,1)[12] errors
##
## Coefficients:
##          ar1     ar2      ma1     sma1  train_customers_1990
##       0.7925  0.1476  -0.8286  -0.6866                0.5509
## s.e.  0.0879  0.0632   0.0736   0.0399                0.1107
##       train_cdays_1990^2  train_hdays_1990^2
##                  16.1744              1.4689
## s.e.              1.9219              0.4687
##
## sigma^2 = 180986989621:  log likelihood = -5435.66
## AIC=10887.31   AICc=10887.7   BIC=10918.79
##
## Training set error measures:
##                     ME     RMSE    MAE        MPE     MAPE      MASE
## Training set -1926.627 414933.3 288317 -0.3965911 4.094071 0.7316139
##                       ACF1
## Training set -0.0001951116
```

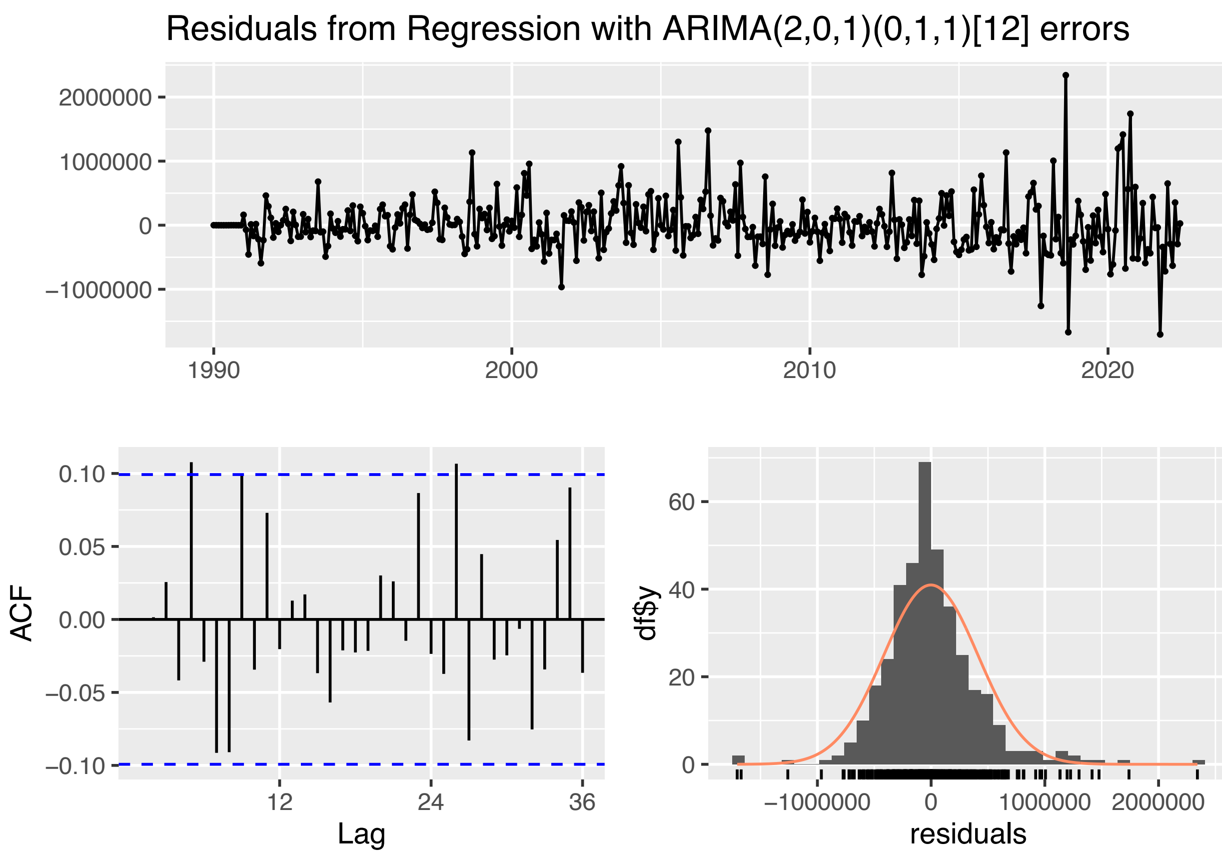
```r
# Model Analysis
# Q-Q plot for residuals
qqnorm(residential_reg_w_arima_err_v2_1990$residuals)
qqline(residential_reg_w_arima_err_v2_1990$residuals)
```




```r
# Shapiro-Wilk normality test
shapiro.test(residential_reg_w_arima_err_v2_1990$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residential_reg_w_arima_err_v2_1990$residuals
## W = 0.92471, p-value = 0.0000000000004314
```

```r
# We can also check the ACF plot for residuals
checkresiduals(residential_reg_w_arima_err_v2_1990)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(2,0,1)(0,1,1)[12] errors
## Q* = 26.01, df = 20, p-value = 0.1655
##
## Model df: 4.   Total lags used: 24
```

```
Box.test(residential_reg_w_arima_err_v2_1990$residuals, lag = 36, type = "Ljung-Box")
```

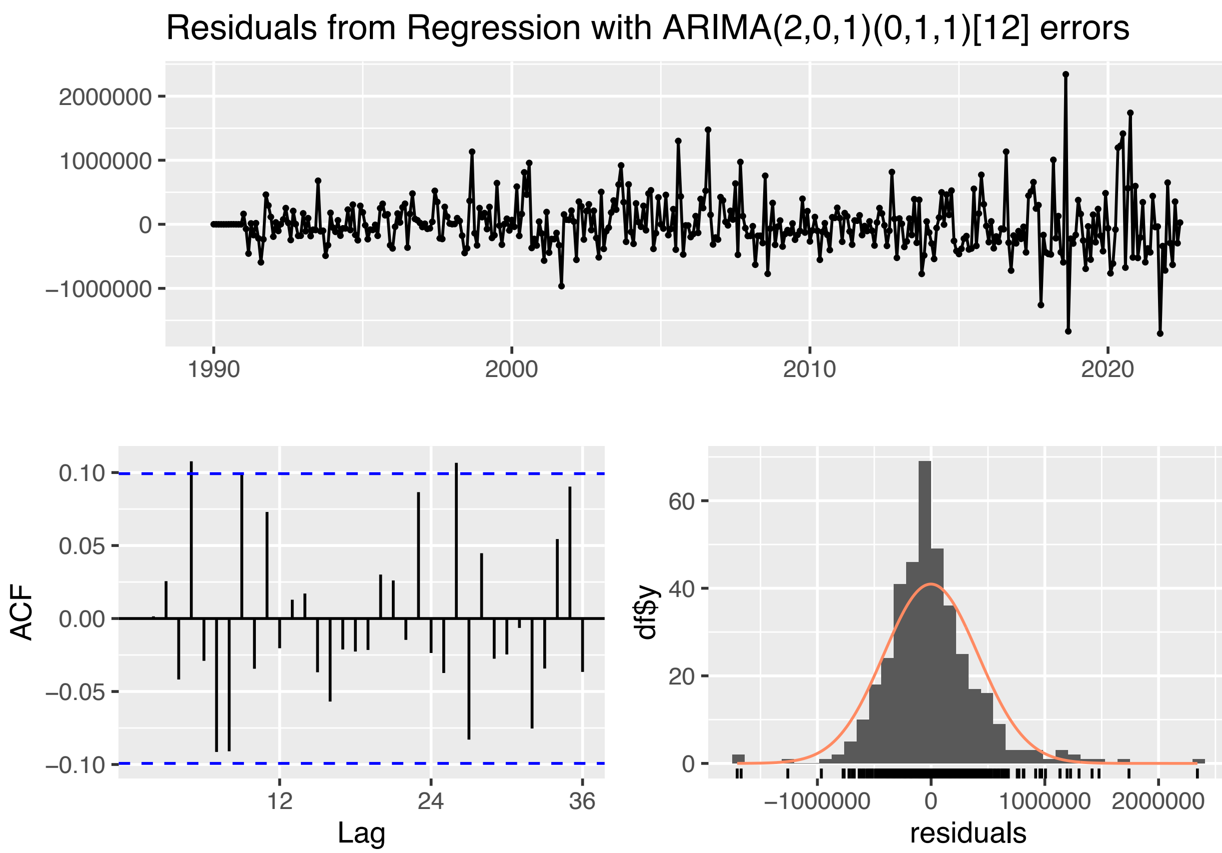
```
##
##  Box-Ljung test
##
## data:  residential_reg_w_arima_err_v2_1990$residuals
## X-squared = 44.004, df = 36, p-value = 0.1689
```

```
## Forecast for ARIMA + reg model (fewer variables)

## Using train values of independent variables to forecast consumption

fcast_residential_reg_w_arima_err_v2_1990 <- forecast(residential_reg_w_arima_err_v2_1990,
                                                      xreg = cbind(test_customers_1990,test_cdays_1990^2, t
                                                      h =13)
```

```
## Warning in forecast.forecast_ARIMA(residential_reg_w_arima_err_v2_1990, : xreg
```

```
## contains different column names from the xreg used in training. Please check
## that the regressors are in the same order.
```

```r
checkresiduals(fcast_residential_reg_w_arima_err_v2_1990)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(2,0,1)(0,1,1)[12] errors
## Q* = 26.01, df = 20, p-value = 0.1655
##
## Model df: 4.   Total lags used: 24
```

```r
summary(fcast_residential_reg_w_arima_err_v2_1990)
```

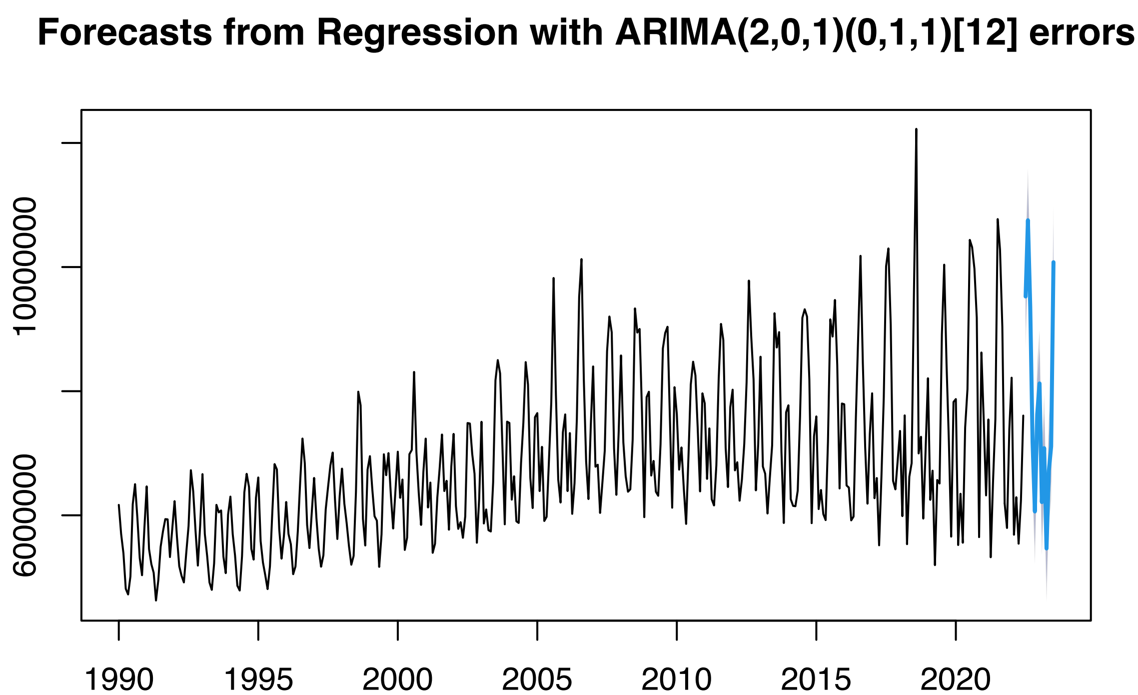
```
##
## Forecast method: Regression with ARIMA(2,0,1)(0,1,1)[12] errors
##
## Model Information:
## Series: train_res_1990
## Regression with ARIMA(2,0,1)(0,1,1)[12] errors
##
## Coefficients:
##           ar1     ar2      ma1     sma1  train_customers_1990
```

```
##       0.7925  0.1476  -0.8286  -0.6866                0.5509
## s.e.  0.0879  0.0632   0.0736   0.0399                0.1107
##       train_cdays_1990^2  train_hdays_1990^2
##                  16.1744              1.4689
## s.e.              1.9219              0.4687
##
## sigma^2 = 180986989621:  log likelihood = -5435.66
## AIC=10887.31   AICc=10887.7   BIC=10918.79
##
## Error measures:
##                     ME     RMSE    MAE        MPE     MAPE      MASE
## Training set -1926.627 414933.3 288317 -0.3965911 4.094071 0.7316139
##                       ACF1
## Training set -0.0001951116
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80   Lo 95    Hi 95
## Jul 2022        9525785  8980580 10070990 8691966 10359604
## Aug 2022       10754681 10209120 11300243 9920317 11589045
## Sep 2022        9362544  8813141  9911948 8522305 10202784
## Oct 2022        7187749  6636211  7739288 6344244  8031255
## Nov 2022        6061506  5507882  6615129 5214812  6908200
## Dec 2022        7630906  7075441  8186372 6781396  8480417
## Jan 2023        8124765  7567646  8681885 7272724  8976806
## Feb 2023        6212726  5654124  6771328 5358418  7067034
## Mar 2023        7082304  6522373  7642236 6225963  7938646
## Apr 2023        5465464  4904340  6026589 4607299  6323630
## May 2023        6717504  6155310  7279698 5857702  7577306
## Jun 2023        7111325  6548170  7674479 6250055  7972595
## Jul 2023       10078461  9480156 10676766 9163433 10993490
```

```r
forecast::accuracy(fcast_residential_reg_w_arima_err_v2_1990, test_res_1990)
```

```
##                       ME     RMSE      MAE        MPE     MAPE      MASE
## Training set   -1926.627 414933.3 288317.0 -0.3965911 4.094071 0.7316139
## Test set     -323140.554 775461.3 573067.7 -4.9989820 8.068319 1.4541785
##                       ACF1 Theil's U
## Training set -0.0001951116        NA
## Test set      0.2853544411   0.69236
```

```r
plot(fcast_residential_reg_w_arima_err_v2_1990)
```




```r
mean_fcast_residential_reg_w_arima_err_v2_1990 <- fcast_residential_reg_w_arima_err_v2_1990$mean
mean_fcast_residential_reg_w_arima_err_v2_1990
```

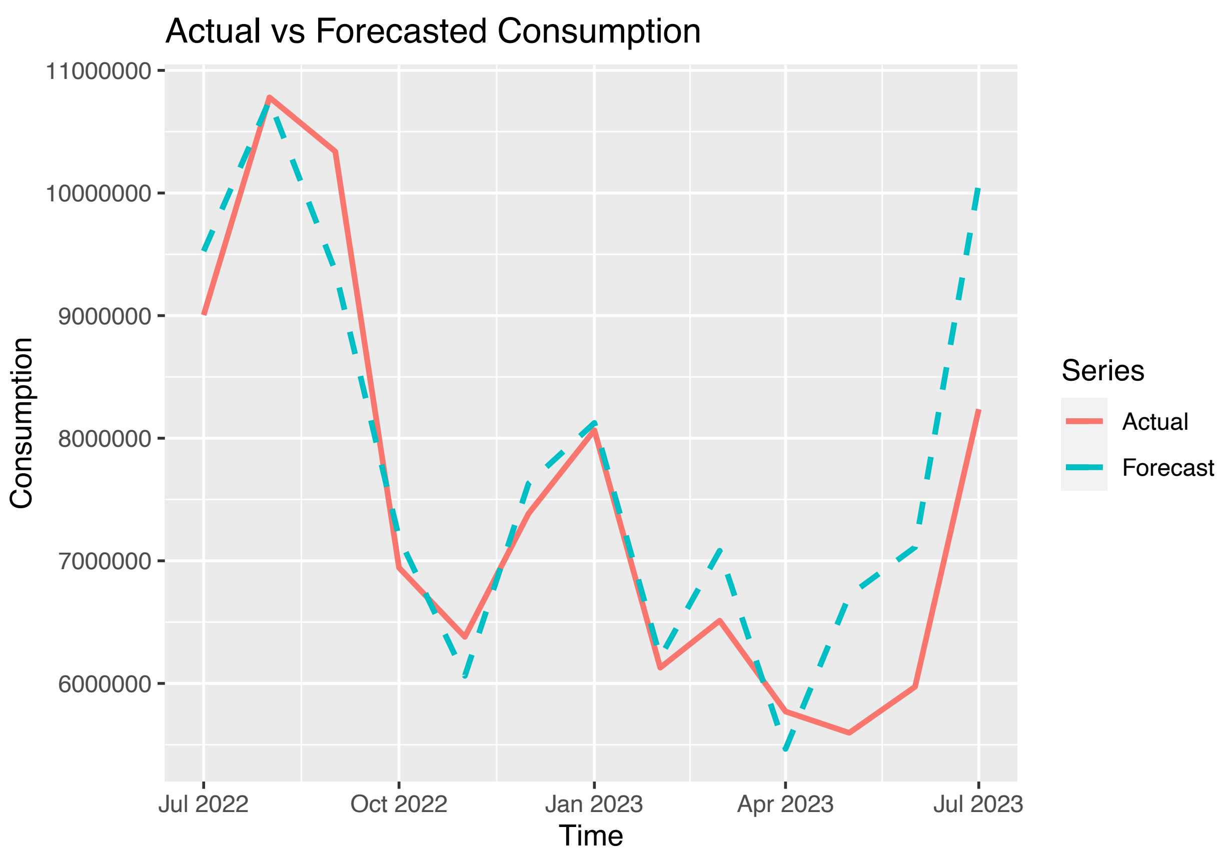
```
##          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
## 2022                                                          9525785 10754681
## 2023  8124765  6212726  7082304  5465464  6717504  7111325 10078461
##          Sep      Oct      Nov      Dec
## 2022  9362544  7187749  6061506  7630906
## 2023
```

```r
library(scales)
```

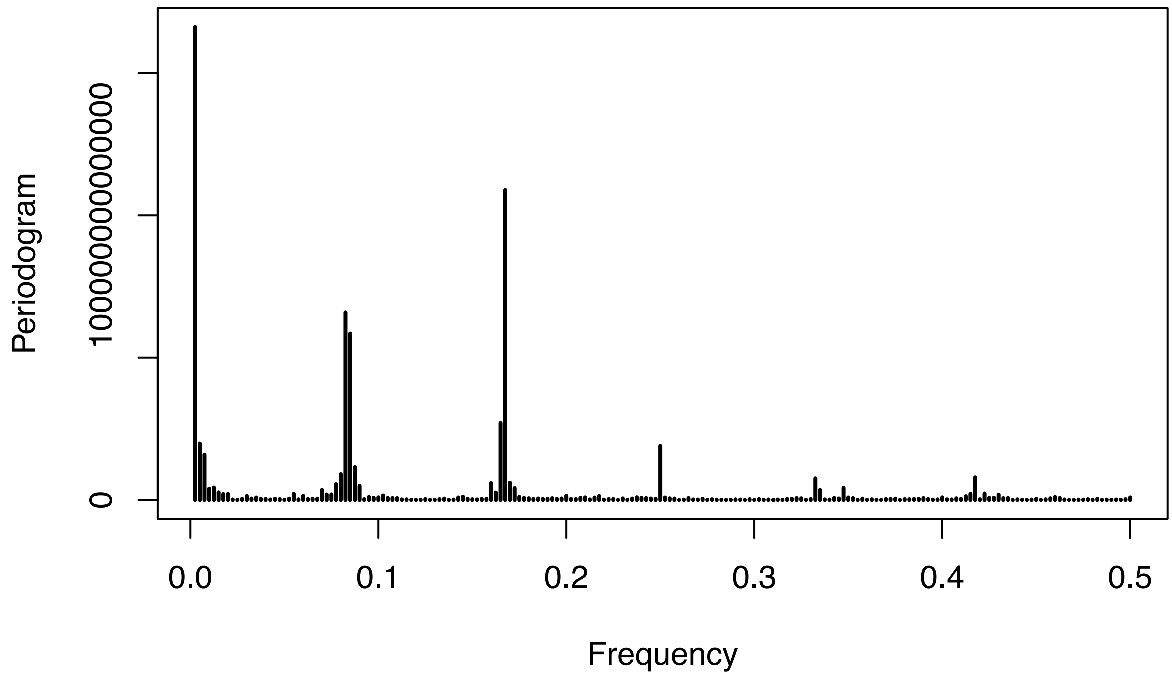
```
##
## Attaching package: 'scales'

## The following object is masked from 'package:purrr':
##
##     discard

## The following object is masked from 'package:readr':
##
##     col_factor
```

```r
# Assuming your Time variable represents months (numeric sequence)
# Creating a sequence of months starting from July and ending in July
months_sequence <- seq(as.Date("2022-07-01"), by = "months", length.out = length(test_res_1990))

# Generating abbreviated month names for the sequence of months
month_labels <- format(months_sequence, "%b")

# Creating the comparison data frame with the month labels
df_comparison_v2 <- data.frame(
  Time = months_sequence,
  Actual = test_res_1990,
  Forecast = mean_fcast_residential_reg_w_arima_err_v2_1990
)

# Plotting the graph with modified x-axis labels
ggplot(df_comparison_v2, aes(x = Time)) +
  geom_line(aes(y = Actual, color = "Actual"), size = 1) +
  geom_line(aes(y = Forecast, color = "Forecast"), size = 1, linetype = "dashed") +
  labs(x = "Time", y = "Consumption", color = "Series") +
  ggtitle("Actual vs Forecasted Consumption")
```

```
## Don't know how to automatically pick scale for object of type <ts>. Defaulting
## to continuous.
```



```r
# Checking periodogram for Fourier
temp <- periodogram(train_res_1990)
```




```r
max_freq <- temp$freq[which.max(temp$spec)]
seasonality <- 1/max_freq
seasonality
```

```
## [1] 400
```

```r
# Now trying Fourier
residential_fourier_1990 <- auto.arima(train_res_1990, xreg = cbind(fourier(train_res_1990,5)),
                                       seasonal = TRUE, lambda = 0)

summary(residential_fourier_1990)
```

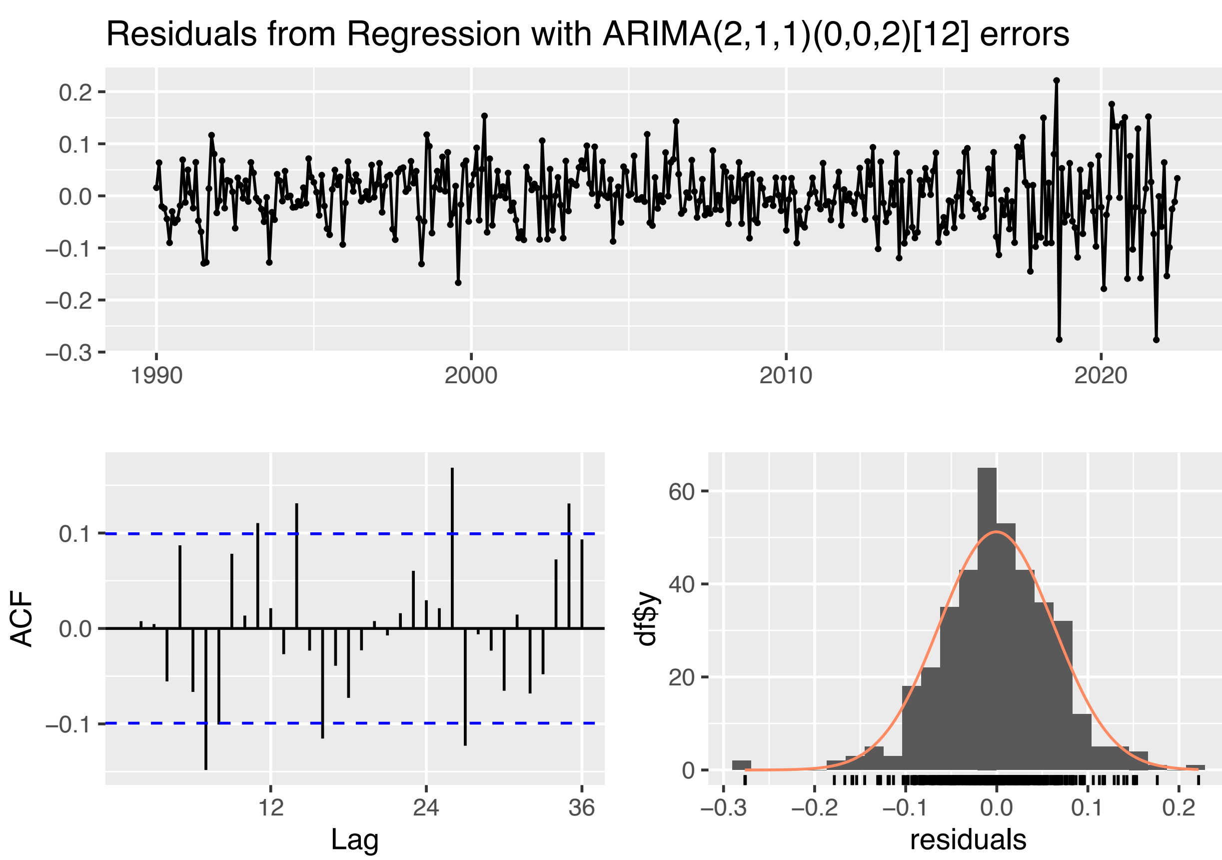
```
## Series: train_res_1990
## Regression with ARIMA(2,1,1)(0,0,2)[12] errors
## Box Cox transformation: lambda= 0
##
## Coefficients:
##          ar1     ar2      ma1    sma1    sma2   drift    S1-12    C1-12   S2-12
##       0.2111  0.1215  -0.9614  0.3494  0.2080  0.0008  -0.1091  -0.0084  0.1242
## s.e.  0.0523  0.0525   0.0130  0.0522  0.0542  0.0003   0.0087   0.0087  0.0068
##         C2-12   S3-12   C3-12   S4-12   C4-12   S5-12   C5-12
```

```
##       -0.0126  0.025  0.0329  0.0322  0.0215  0.0292  0.0088
## s.e.   0.0068  0.006  0.0060  0.0058  0.0058  0.0061  0.0061
##
## sigma^2 = 0.004149:  log likelihood = 521.3
## AIC=-1008.6   AICc=-1006.95   BIC=-941.22
##
## Training set error measures:
##                    ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 10081.3 462546.8 332098.4 -0.2403052 4.766758 0.8427108
##                      ACF1
## Training set -0.01507275
```

```
forecast::accuracy(residential_fourier_1990)
```

```
##                    ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 10081.3 462546.8 332098.4 -0.2403052 4.766758 0.8427108
##                      ACF1
## Training set -0.01507275
```

```
checkresiduals(residential_fourier_1990)
```




```
##
##  Ljung-Box test
##
```

```
## data:  Residuals from Regression with ARIMA(2,1,1)(0,0,2)[12] errors
## Q* = 44.478, df = 19, p-value = 0.0008106
##
## Model df: 5.   Total lags used: 24
```

FILE ENDS

MODEL TESTING

CODE WRITTEN BELOW WAS NOT USED

Models to Test

```
#models <- list(
#  list("Residential Auto Arima - 1990", residential_auto_arima_1990, train_res_1990, test_res_1990, 12
#  list("Residential Auto Arima - 2005", residential_auto_arima_2005, train_res_2005, test_res_2005, 12
#  list("Residential Auto Arima - 2013", residential_auto_arima_2013, train_res_2013, test_res_2013, 12
#  list("Residential Seasonal Naive - 1990", snaive_model_1990, train_res_1990, test_res_1990, 12),
#  list("Residential Seasonal Naive - 2005", snaive_model_2005, train_res_2005, test_res_2005, 12),
#  list("Residential Seasonal Naive - 2013", snaive_model_2013, train_res_2013, test_res_2013, 12),
#  list("ETS - 1990", ets_model_1990, train_res_1990, test_res_1990, 12),
#  list("ETS - 2005", ets_model_2005, train_res_2005, test_res_2005, 12),
#  list("ETS - 2013", ets_model_2013, train_res_2013, test_res_2013, 12),
#  list("Linear Regression - 1990", tslm_model_1990, train_res_1990, test_res_1990, 12),
#  list("Linear Regression - 2005", tslm_model_2005, train_res_2005, test_res_2005, 12),
#  list("Linear Regression - 2013", tslm_model_2013, train_res_2013, test_res_2013, 12),
#  list("Regression with ARIMA Errors - 1990", residential_reg_w_arima_err_1990, train_res_1990, test_r
#  list("Regression with ARIMA Errors - 2005", residential_reg_w_arima_err_2005, train_res_2005, test_r
#  list("Regression with ARIMA Errors - 2013", residential_reg_w_arima_err_2013, train_res_2013, test_r
#)
```

Training Metrics

```
#evaluate_models_train <- function(models) {
#  results <- list()

#  for (model_info in models) {

#    model_name <- model_info[[1]]
#    model <- model_info[[2]]
#    train_data <- model_info[[4]]
#    test_data <- model_info[[4]]
#    horizon <- model_info[[5]]

#    errors <- accuracy(model)
#    MAPE <- errors["Training set", "MAPE"]
#    RMSE <- errors["Training set", "RMSE"]
#    AICc <- ifelse(exists("aicc", where = model), model$aicc, NA)

#    results[[model_name]] <- list(MAPE = MAPE, RMSE = RMSE, AICc = AICc)

#    cat(paste0(model_name, " Model Performance on Training Data: \n",
```

```
#               "MAPE: ", MAPE, "\n",
#               "RMSE: ", RMSE, "\n",
#               "AICc: ", AICc, "\n\n"))
#  }

#  return(results)
#}

#results_training <- evaluate_models_train(models)
```

Testing Metrics

```
#evaluate_models_test <- function(models) {
#  results <- list()

#  for (model_info in models) {

#    model_name <- model_info[[1]]
#    model <- model_info[[2]]
#    train_data <- model_info[[4]]
#    test_data <- model_info[[4]]
#    horizon <- model_info[[5]]
#    if (length(model_info) >= 6){
#      xreg_val <- model_info[[6]]
#      model_forecast <- forecast(model, xreg=xreg_val, h=horizon)
#    } else {
#      model_forecast <- forecast(model, h=horizon)
#    }

#    errors <- accuracy(model_forecast, test_data)
#    MAPE <- errors["Test set", "MAPE"]
#    RMSE <- errors["Test set", "RMSE"]
#    AICc <- ifelse(exists("aicc", where = model), model$aicc, NA)

#    results[[model_name]] <- list(MAPE = MAPE, RMSE = RMSE, AICc = AICc, Forecast = model_forecast)

#    cat(paste0(model_name, " Model Performance on Test Data: \n",
#               "MAPE: ", MAPE, "\n",
#               "RMSE: ", RMSE, "\n",
#               "AICc: ", AICc, "\n\n"))
#  }

#  return(results)
#}

#results <- evaluate_models_test(models)
```