Energy Price Forecasting

Juliet Alexandra

2025-07-11

Libraries

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

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

```
library(ggplot2)
library(ggpmisc)
```

```
## Loading required package: ggpp

## Registered S3 methods overwritten by 'ggpp':
##   method                  from
##   heightDetails.titleGrob ggplot2
##   widthDetails.titleGrob  ggplot2

##
## Attaching package: 'ggpp'

## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
library(tseries)
```

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

```
library(nortest)
library(zoo)
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(car)
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
library(lubridate)
```

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

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

```
library(purrr)
```

```
##
## Attaching package: 'purrr'
```

```
## The following object is masked from 'package:car':
##
##     some
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(forecast)
```

Loading Data

```r
csv_folder <- "2017"
csv_files <- list.files(path = csv_folder, pattern = "*.csv", full.names = TRUE)

training_set <- csv_files %>% lapply(read.csv) %>% bind_rows()
head(training_set)
```

```
##   TradingDate TradingPeriod PointOfConnection DollarsPerMegawattHour
## 1  2017-01-01             1           ABY0111                  35.95
## 2  2017-01-01             1           ALB0331                  38.11
## 3  2017-01-01             1           ALB1101                  38.04
## 4  2017-01-01             1           APS0111                  35.37
## 5  2017-01-01             1           ARA2201                  34.82
## 6  2017-01-01             1           ARG1101                  37.44
```

This forecasting is going to be performed on one Point of Connection (ABY0111) for simplicity:

```r
Training_set <- training_set[training_set$PointOfConnection == "ABY0111", ]
Training_set <- Training_set[order(Training_set$TradingDate, Training_set$TradingPeriod), ]
head(Training_set)
```

```
##      TradingDate TradingPeriod PointOfConnection DollarsPerMegawattHour
## 1     2017-01-01             1           ABY0111                  35.95
## 244   2017-01-01             2           ABY0111                  41.65
## 487   2017-01-01             3           ABY0111                  26.58
## 730   2017-01-01             4           ABY0111                  26.78
## 973   2017-01-01             5           ABY0111                  26.66
## 1216  2017-01-01             6           ABY0111                  19.67
```

Data Cleaning and Wrangling

```r
summary(Training_set)
```

```
##  TradingDate        TradingPeriod   PointOfConnection  DollarsPerMegawattHour
##  Length:17520       Min.   : 1.00   Length:17520       Min.   :  0.02
##  Class :character   1st Qu.:12.75   Class :character   1st Qu.: 47.17
##  Mode  :character   Median :24.50   Mode  :character   Median : 65.47
##                     Mean   :24.50                      Mean   : 78.56
##                     3rd Qu.:36.25                      3rd Qu.: 96.36
##                     Max.   :50.00                      Max.   :925.29
```

TradingDate has being incorrectly classed as a categorical variable instead of a date.

```
Training_set$TradingDate <- as.Date(Training_set$TradingDate)
```

```
summary(Training_set)
```

```
##   TradingDate         TradingPeriod   PointOfConnection  DollarsPerMegawattHour
##  Min.   :2017-01-01   Min.   : 1.00   Length:17520       Min.   :  0.02
##  1st Qu.:2017-04-02   1st Qu.:12.75   Class :character   1st Qu.: 47.17
##  Median :2017-07-02   Median :24.50   Mode  :character   Median : 65.47
##  Mean   :2017-07-01   Mean   :24.50                      Mean   : 78.56
##  3rd Qu.:2017-10-01   3rd Qu.:36.25                      3rd Qu.: 96.36
##  Max.   :2017-12-31   Max.   :50.00                      Max.   :925.29
```

There are four variables TradingDate, TradingPeriod, PointOfConnection, DollarsPerMegawattHour.

There are 17520 observations (rows).

The TradingDate variable is a date object ranging from 2017-01-01 to 2017-12-31.

The TradingPeriod variable gives the 30 minute interval where electricity was bought and sold. It is numerical.

The PointOfConnection variable is an ordinal categorical variable, it gives the grid location where electricty is entering (or exiting) the network. I'm only working with the ABY0111 point of connection.

The DollarsPerMegawattHour variable is numerical, it gives the wholesale price of electricity. It's the price at which electricity is bought and sold in the wholesale market at a specific date and time, and point of connection.

I can see that TradingPeriod has a max value of 50 which shouldn't be possible as there are only 48 trading periods in New Zealand's electricity market.

```
Training_set[Training_set$TradingPeriod > 48, ]
```

```
##         TradingDate TradingPeriod PointOfConnection DollarsPerMegawattHour
## 1073089  2017-04-02            49           ABY0111                   5.81
## 1073332  2017-04-02            50           ABY0111                   2.05
```

There are 2 observations where the trading period is larger than 48, since NZ's electricity market has 48 trading periods a day (each representing a 30 minute interval) these observations could be errors. Both are on 2017-04-02.

On 2017-04-02 daylight savings ended and clocks were turned back one hour, meaning there was an extra hour for that day, resulting in two more 30 minute trading periods

I am going to remove these observations from the dataset as I want to maintain a consistent daily structure:

```
ErrorIndices <- which(Training_set$TradingPeriod > 48)
Training_set <- Training_set[-ErrorIndices, ]
```

```
sapply(Training_set, anyNA)
```

```
##            TradingDate          TradingPeriod      PointOfConnection
##                  FALSE                  FALSE                  FALSE
## DollarsPerMegawattHour
##                  FALSE
```

There are no NA values in the data set for any of the variables.

Checking that the data contains all 365 days of the year:

```
NROW(unique(Training_set$TradingDate))
```

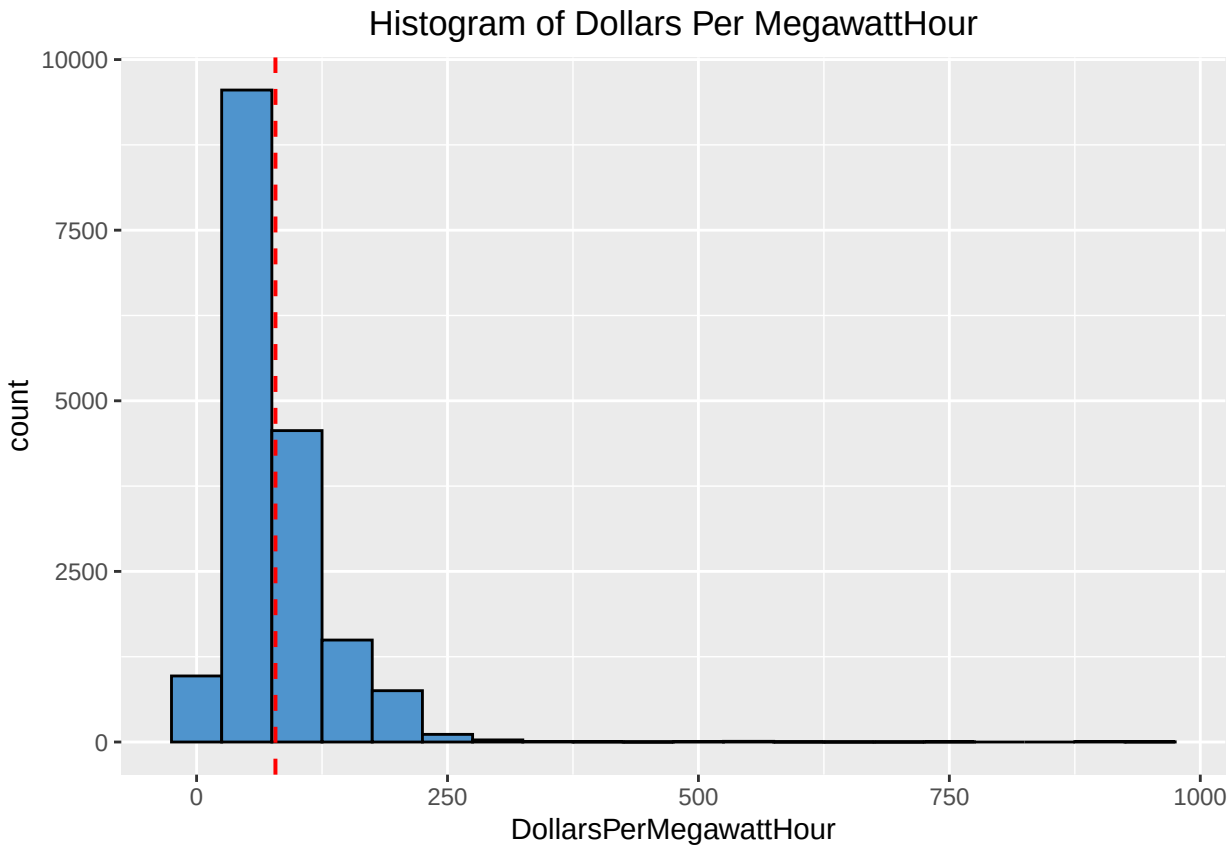
```
## [1] 365
```

There are 365 days like I expected

If there hadn't been (eg due to leap years), I would have dropped the extra day as ARIMA model which needs uniform seasonality

EDA

```
ggplot(Training_set, aes(x = DollarsPerMegawattHour)) +
  geom_histogram(binwidth = 50, color = 'black', fill = 'steelblue3') +
  geom_vline(aes(xintercept = mean(DollarsPerMegawattHour)), color = "red", linetype = "dashed", linewi
  ggtitle("Histogram of Dollars Per MegawattHour") +
  theme(plot.title = element_text(hjust = 0.5))
```



There are very few values above 400 for DollarsPerMegawattHour.

The mode for DollarsPerMegawattHour is around 70 dollars.

There are no negative values for DollarsPerMegawattHour.

```
Training_set[Training_set$DollarsPerMegawattHour > 400,]
```

```
##         TradingDate TradingPeriod PointOfConnection DollarsPerMegawattHour
## 1648756  2017-05-22            16           ABY0111                  484.73
## 1648999  2017-05-22            17           ABY0111                  449.12
## 2246536  2017-07-12            28           ABY0111                  534.98
## 2246779  2017-07-12            29           ABY0111                  532.88
## 2248966  2017-07-12            38           ABY0111                  898.49
## 2249452  2017-07-12            40           ABY0111                  501.21
## 2249695  2017-07-12            41           ABY0111                  545.22
## 2250181  2017-07-12            43           ABY0111                  534.55
## 2256742  2017-07-13            22           ABY0111                  506.06
## 2256985  2017-07-13            23           ABY0111                  673.43
## 2257228  2017-07-13            24           ABY0111                  681.21
## 2257471  2017-07-13            25           ABY0111                  509.21
## 2257714  2017-07-13            26           ABY0111                  749.12
## 2257957  2017-07-13            27           ABY0111                  749.40
## 2258200  2017-07-13            28           ABY0111                  752.90
## 2258443  2017-07-13            29           ABY0111                  902.01
## 2258686  2017-07-13            30           ABY0111                  921.85
## 2258929  2017-07-13            31           ABY0111                  925.29
## 2259172  2017-07-13            32           ABY0111                  925.29
## 2259415  2017-07-13            33           ABY0111                  925.21
## 2259658  2017-07-13            34           ABY0111                  618.93
## 2261116  2017-07-13            40           ABY0111                  901.95
## 2261359  2017-07-13            41           ABY0111                  902.35
## 2261602  2017-07-13            42           ABY0111                  901.84
## 2268406  2017-07-14            22           ABY0111                  538.99
## 2268649  2017-07-14            23           ABY0111                  598.66
## 2268892  2017-07-14            24           ABY0111                  557.35
## 2269378  2017-07-14            26           ABY0111                  545.40
## 2269621  2017-07-14            27           ABY0111                  544.52
## 2269864  2017-07-14            28           ABY0111                  539.85
## 2270107  2017-07-14            29           ABY0111                  538.32
## 2270836  2017-07-14            32           ABY0111                  482.39
## 2476900  2017-08-01            16           ABY0111                  521.17
## 2517724  2017-08-04            40           ABY0111                  901.24
```

There are 34 observations where the DollarsPerMegawattHour value was larger than the 400, which in a data set of 17158 observations is very few.

```
mean(Training_set$DollarsPerMegawattHour)
```

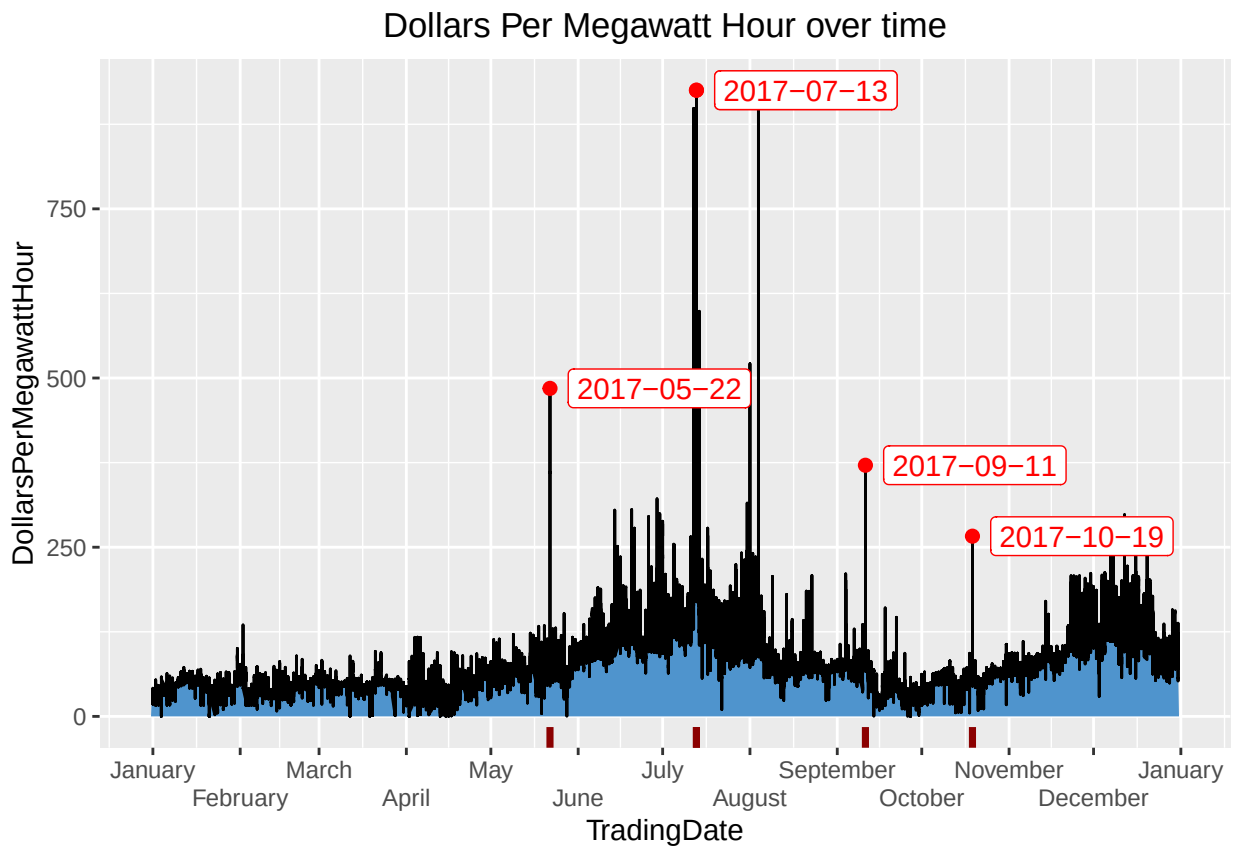
```
## [1] 78.57018
```

The mean DollarsPerMegawattHour for year 2017 at point connection ABY0111 is 78.57 (round to 2.dp)

Time series plot:

```
ggplot(Training_set, aes(x = TradingDate, y = DollarsPerMegawattHour)) +
  geom_area(fill = "steelblue3")+
  geom_line() +
```

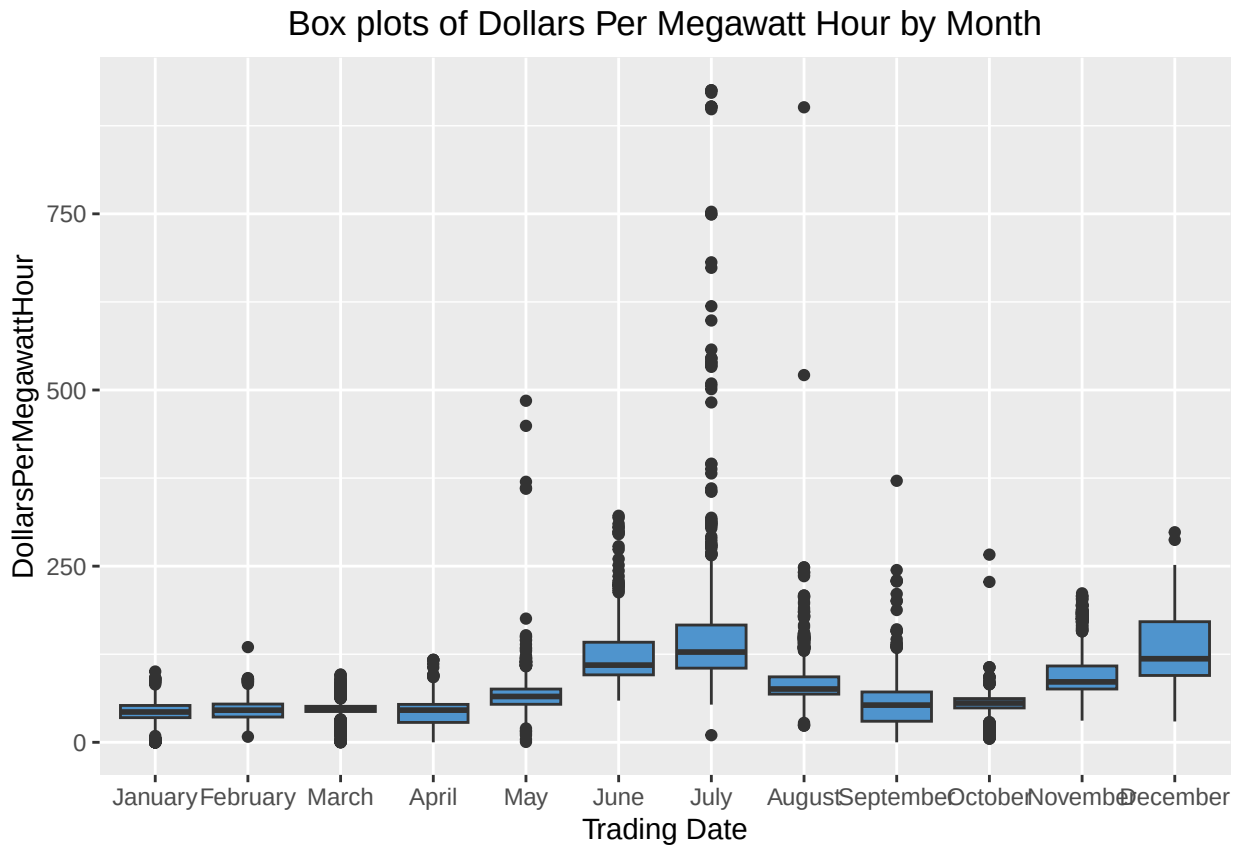
```
  scale_x_date(date_breaks = "1 month", date_labels = "%B", guide = guide_axis(n.dodge = 2)) +
  stat_peaks(geom = "point", span = 3225, color = "red", size = 2) +
  stat_peaks(geom = "label", span = 3225, color = "red", angle = 0, hjust = -0.1, x.label.fmt = "%Y-%m-
  stat_peaks(geom = "rug", span = 3225, color = "darkred", sides = "b", size = 1.25) +
  ggtitle("Dollars Per Megawatt Hour over time") +
  theme(plot.title = element_text(hjust = 0.5))
```



I can see that there are specific dates where the whole sale price of electricity spiked to unusually high amounts.

There were spikes in electricity price on 2027-05-22, 2017-07-13, 2017-09-11, and 2017-10-19.

```
ggplot(Training_set, aes(x = format(TradingDate, "%m"), y = DollarsPerMegawattHour)) +
  geom_boxplot(fill = "steelblue3") +
  xlab("Trading Date") + scale_x_discrete(labels = month.name) +
  ggtitle("Box plots of Dollars Per Megawatt Hour by Month") +
  theme(plot.title = element_text(hjust = 0.5))
```



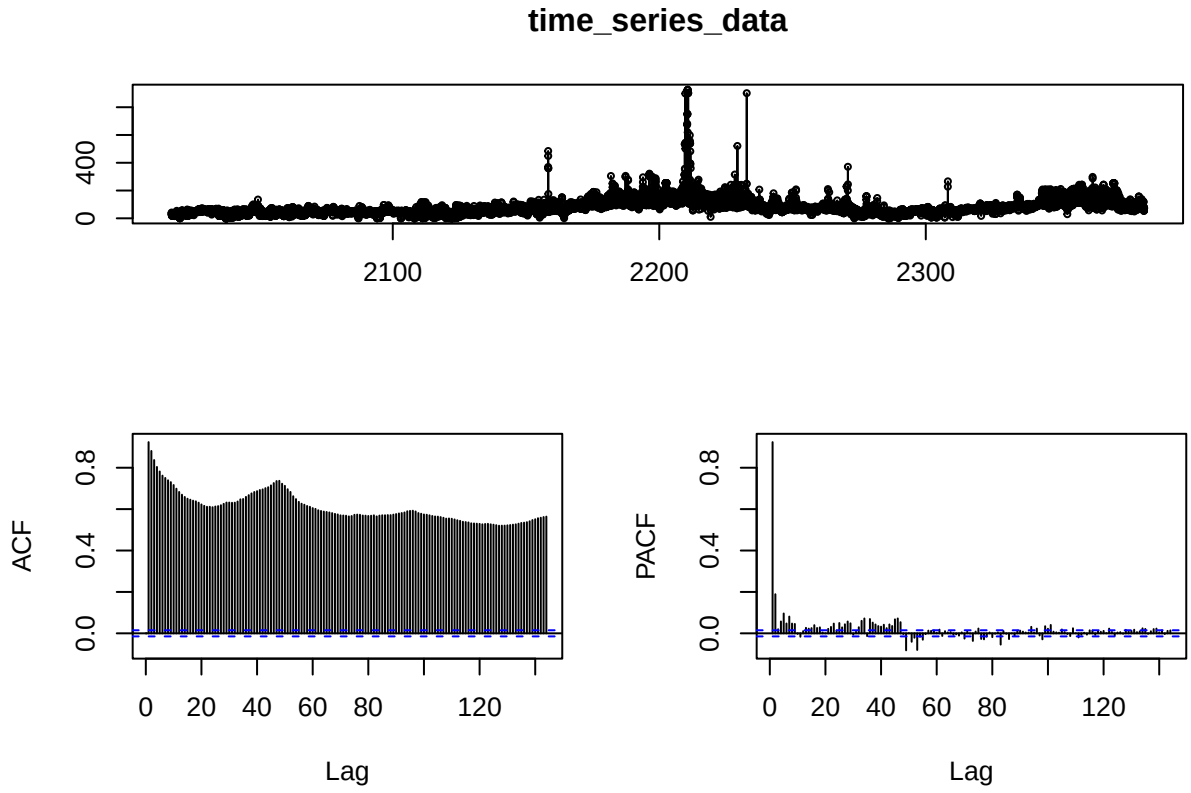
The price of electricity is much higher for June and July. The lower quartile of June and July doesn't even overlap with the upper quartiles of the previous months.

The prices dip again for August, September and October but then start increasing again in November and December.

Autocorrelation & Partial Autocorrelation Analysis

```
price_vector <- as.numeric(Training_set$DollarsPerMegawattHour)
# frequency is 48 because I have half hourly data
time_series_data <- ts(price_vector, start = c(2017, 1), frequency = 48)
```

```
tsdisplay(time_series_data)
```



ACF:

I can see that all of the spikes in the ACF plot are outside of the blue dashed significance bound this means that the time series data is highly autocorrelated and non-stationary. The autocorrelation at all of the lags is statistically meaningful and not just white noise. Past values have a strong influence on future ones across many lags.

There is particularly high autocorrelation at the early lags. This suggests that recent values strongly influence near future behavior, implying short term memory or an autoregressive structure.

There is a spike at around every 48 lags which suggests a seasonal pattern.

The bars gradually decay suggesting a persistent trend or that it's non-stationary. This suggests the mean and variance may be changing over time, and the structure could be due to an autoregressive process or an underlying seasonal component.

PACF:

There is a large spike at lag 1 in the PACF plot suggesting that the current price is heavily influenced by the immediate past period. This spike at lag 1 suggests an AR(1) structure. This supports the idea that the series has short-term autoregressive structure.

Past lag 1 there is gradual tapering, rather than a sharp cutoff like a pure AR(p) process.

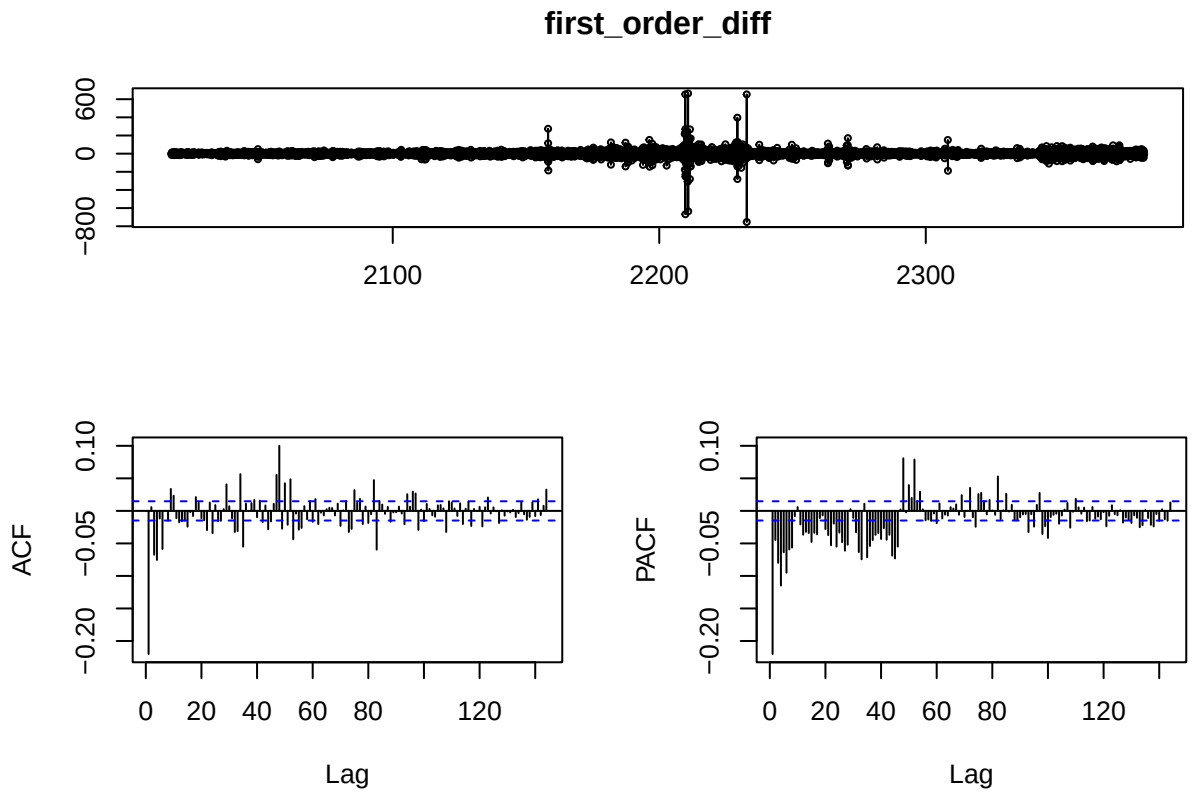
Conclusion:

All of the above information from the ACF and PACF plots suggests that the data is non-stationary and that there could be a seasonal component to the data.

I am going to apply first order differencing to the data to make the data stationary, as stationary data is a requirement of ARIMA.

```r
first_order_diff <- diff(time_series_data)
```

```r
tsdisplay(first_order_diff)
```



ACF:

After applying first order differencing Most lags in the ACF plot are within the confidence bounds, meaning there is no strong autocorrelation.

There is no sharp cutoff or pattern in the plot. There are no clear periodic spikes (e.g. at lag 48), which indicates that there is no seasonality.

The ACF plot no longer shows a slow decay, instead there are isolated significant spikes in the plot. This pattern indicates that the trend has been removed and that the series is now stationary, with stable mean and variance over time.

PACF:

There is a moderate spike at lag 1, outside the confidence bounds. Most subsequent lags are within the confidence bounds. This suggests a short term autoregressive pattern. This means each electricity price is influenced by its immediate predecessor.

The decay after the first lag also supports that the first order differencing has stabilized the series, leaving behind no long range autocorrelation.

Since this resembles an AR(1) structure still, when I fit the ARIMA model I will include an AR(1) component to model the autoregressive behavior.

Stationarity Diagnostics: ADF, KPSS & Phillips–Perron

Augmented Dickey-Fuller test:

Null hypothesis: has a unit root (non stationary)

Alternative hypothesis: doesn't have a unit root (stationary)

```
adf_result <- adf.test(first_order_diff)
```

```
## Warning in adf.test(first_order_diff): p-value smaller than printed p-value
```

```
print(adf_result)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  first_order_diff
## Dickey-Fuller = -39.215, Lag order = 25, p-value = 0.01
## alternative hypothesis: stationary
```

The p-value is less than the significance level of 0.5, I reject the null hypothesis and conclude that the differenced series is stationary.

KPSS test:

Null hypothesis: is stationary

Alternative hypothesis: isn't stationary

```
kpss_result <- kpss.test(first_order_diff, null = "Trend")
```

```
## Warning in kpss.test(first_order_diff, null = "Trend"): p-value greater than
## printed p-value
```

```
print(kpss_result)
```

```
##
##  KPSS Test for Trend Stationarity
##
## data:  first_order_diff
## KPSS Trend = 0.0010824, Truncation lag parameter = 14, p-value = 0.1
```

The p-value is larger than the significance level of 0.5 which means I fail to reject the null hypothesis and conclude that the differenced series is stationary.

Phillips–Perron test:

Null hypothesis: has a unit root (non stationary)

Alternative hypothesis: doesn't have a unit root (stationary)

```r
Phillips <- pp.test(first_order_diff)
```

```
## Warning in pp.test(first_order_diff): p-value smaller than printed p-value
```

```r
print(Phillips)
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  first_order_diff
## Dickey-Fuller Z(alpha) = -16744, Truncation lag parameter = 14, p-value
## = 0.01
## alternative hypothesis: stationary
```

The p-value is less than the significance level of 0.5, I reject the null hypothesis and conclude that the differenced series is stationary.

The ADF, KPSS and Phillips–Perron tests all concluded that the first order differenced series is stationary.

Fitting models

ARIMA

I'm using auto.arima with d = 1 (first order differencing) to find a suitable starting ARIMA model that I can manually adjust later.

```r
ARIMA_model1 <- auto.arima(time_series_data, stepwise = TRUE, approximation = TRUE, d = 1, max.p = 3, ma
```

```r
summary(ARIMA_model1)
```

```
## Series: time_series_data
## ARIMA(3,1,1)(0,0,2)[48]
##
## Coefficients:
##          ar1     ar2      ar3      ma1    sma1    sma2
##       0.6674  0.1452  -0.0306  -0.9768  0.1288  0.0273
## s.e.  0.0080  0.0091   0.0078   0.0026  0.0076  0.0074
##
## sigma^2 = 380.6:  log likelihood = -76894.86
## AIC=153803.7   AICc=153803.7   BIC=153858.1
##
## Training set error measures:
##                       ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 0.02386625 19.50573 8.505806 -79.89163 88.51167 0.4361455
##                       ACF1
## Training set -0.0001269205
```

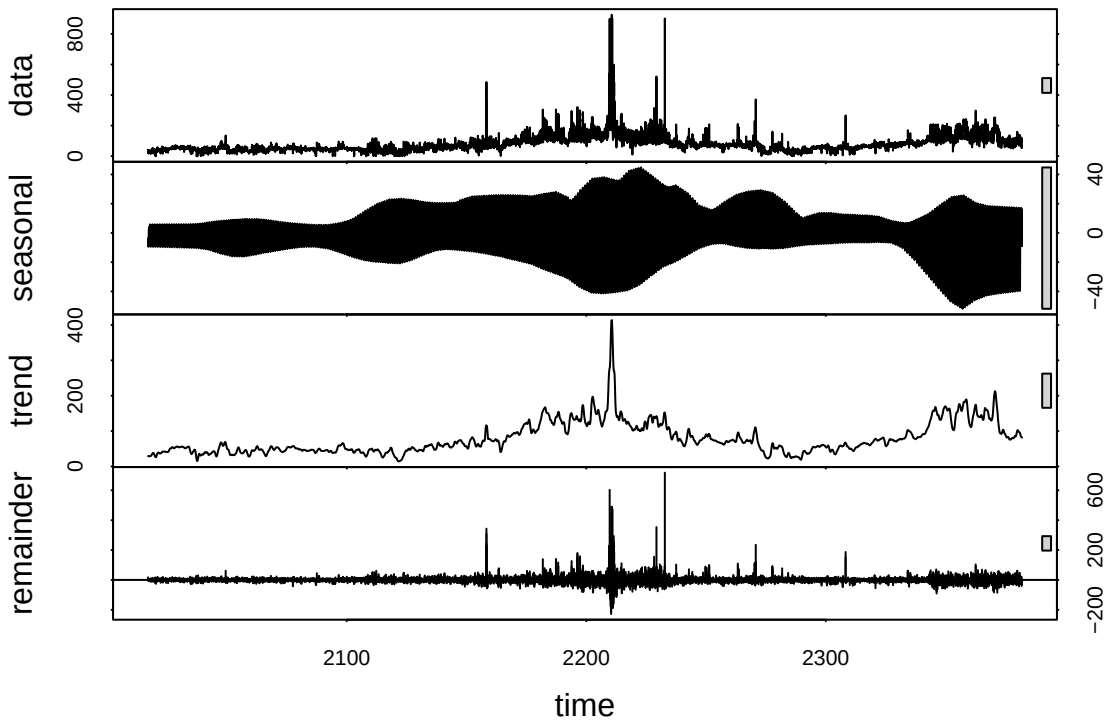
My current model is ARIMA(3,1,1)(0,0,2)[48]

I have a non seasonal AR(3), first order differencing, MA(1) and no seasonal AR or differencing and seasonal MA, there is a seasonal period of 48

STL-ARIMA

Decomposing the series and visualizing the decomposition:

```
decomp <- stl(time_series_data, s.window = 48)
plot(decomp)
```



The second panel shows a repeating pattern cycling between 40 and -40, meaning there's definitely a seasonal component. It consistently oscillates around zero, meaning that the default seasonal detection works well.

The third panel shows a smooth and mild upward movement suggests a weak long-term drift. The overall trend shows short-term fluctuations.

The fourth panel shows that the remainder is spiky and uneven, it contains outliers and volatility that is not explained by seasonality or trend.

Fitting an STL - ARIMA to seasonally adjusted series:

```
STL_ARIMA_model1 <- stlm(time_series_data, method = "arima")
```

My current model is STL-ARIMA(5,1,1)

ARIMAX

Feature engineering:

```r
# Day of Week
Day_of_Week <- as.numeric(format(Training_set$TradingDate, "%w"))
# Weekend / Weekday flag
Is_Weekend <- as.numeric(format(Training_set$TradingDate, "%w") %in% c("0", "6"))
# Month
Month <- as.numeric(format(Training_set$TradingDate, "%m"))
# Hour of day
Hour_of_Day <- floor((Training_set$TradingPeriod - 1) / 2)
# Peak period flag
Is_peak <- ifelse(Training_set$TradingPeriod %in% c(15:40), 1, 0)

xreg <- cbind(Day_of_Week, Is_Weekend, Month, Hour_of_Day, Is_peak)
```

These engineered variables are my xreg matrix for ARIMAX

fitting model:

```r
ARIMAX_model1 <- auto.arima(time_series_data, xreg = xreg, stepwise = TRUE, approximation = TRUE, d = 1
```

Since the series is being differenced (d = 1) I've decided that drift is not necessary.

TBATS

There is some residual autocorrelation in the series as seen from the acf and pacf plots. So I have set use.arma.errors = TRUE to include ARMA errors to model this residual autocorrelation.

Fitting model:

```r
TBATS_model1 <- tbats(time_series_data, use.arma.errors = TRUE, seasonal.periods = c(48, 336)) # daily
```

Forecasting

Loading Test Data

Loading test data:

```r
csv_folder <- "2018"
csv_files <- list.files(path = csv_folder, pattern = "*.csv", full.names = TRUE)

test_data <- csv_files %>% lapply(read.csv) %>% bind_rows()
test_data <- test_data[test_data$PointOfConnection == "ABY0111", ]
test_data <- test_data[order(test_data$TradingDate, test_data$TradingPeriod), ]
test_data$TradingDate <- as.Date(test_data$TradingDate)
```

Making it a time series object:

```r
price_vector2 <- as.numeric(test_data$DollarsPerMegawattHour)
Test_time_series <- ts(price_vector2, start = c(2018, 1), frequency = 48)
```

ARIMA

```
forecast_raw_ARIMA_1 <- forecast(ARIMA_model1, h = 17520)
```

```
forecast_raw_ARIMA <- ts(forecast_raw_ARIMA_1, start = c(2018, 1), frequency = 48)
```

STL-ARIMA

I'm going to forecast 48 trading periods for 12 months

```
forecast12months <- forecast(STL_ARIMA_model1, h = 17520)
```

```
forecast_raw_STL <- ts(forecast12months, start = c(2018, 1), frequency = 48)
```

ARIMAX forecasting

```
# Day of Week
Day_of_Week <- as.numeric(format(test_data$TradingDate, "%w"))
# Weekend / Weekday flag
Is_Weekend <- as.numeric(format(test_data$TradingDate, "%w") %in% c("0", "6"))
# Month
Month <- as.numeric(format(test_data$TradingDate, "%m"))
# Hour of day
Hour_of_Day <- floor((test_data$TradingPeriod - 1) / 2)
# Peak period flag
Is_peak <- ifelse(test_data$TradingPeriod %in% c(15:40), 1, 0)

xreg_future <- cbind(Day_of_Week, Is_Weekend, Month, Hour_of_Day, Is_peak)
```

forecasting:

```
forecast12months2 <- forecast(ARIMAX_model1, h = 17520, xreg = xreg_future)
```

```
forecast_raw_ARIMAX <- ts(forecast12months2, start = c(2018, 1), frequency = 48)
```

TBATS

Forecasting:

```
forecast_tbats <- forecast(TBATS_model1, h = 17520)
forecast_raw_TBATS <- ts(forecast_tbats, start = c(2018, 1), frequency = 48)
```

Accuracy metrics

Model accuracy:

```
ARIMA_acc <- accuracy(forecast_raw_ARIMA, Test_time_series)
STL_ARIMA_acc <- accuracy(forecast_raw_STL, Test_time_series)
ARIMAX_acc <- accuracy(forecast_raw_ARIMAX, Test_time_series)
TBATS_acc <- accuracy(forecast_raw_TBATS, Test_time_series)
```

```
acc_list <- list(
  ARIMA     = ARIMA_acc,
  STL_ARIMA = STL_ARIMA_acc,
  ARIMAX    = ARIMAX_acc,
  TBATS     = TBATS_acc)

acc_df <- map_df(acc_list, ~ as.data.frame(.x)["Test set", ], .id = "Model")

acc_df
```

```
##                   Model        ME     RMSE      MAE       MPE     MAPE      MASE
## Test set...1     ARIMA  6.130026 18.29114 12.67222  3.125512 14.33552 0.6497836
## Test set...2 STL_ARIMA 11.084312 21.15245 16.55240  8.990807 18.40260 0.8487441
## Test set...3    ARIMAX 53.027272 55.86795 53.02727 58.906641 58.90664 2.7190373
## Test set...4     TBATS 12.628650 23.61572 19.15047 10.283711 21.43884 0.9819636
##                    ACF1 Theil's U
## Test set...1 0.6247587  1.324960
## Test set...2 0.6316401  1.486087
## Test set...3 0.6298740  3.958930
## Test set...4 0.7159969  1.683164
```

The ARIMA model has the best ME, RMSE, MAE, MPE, MAPE and MASE. ARIMA was the most accurate model.

The worst model was ARIMAX with the worst ME, RMSE, MAE, MPE, MAPE and MASE.

The second worst model was TBATS.

All models have similar ACF1 values which suggests that the residuals for all models still have short range correlation.

The MASE for ARIMA and STL-ARIMA is less than 1 which means they beat a naive "yesterday's price" benchmark on average absolute error.

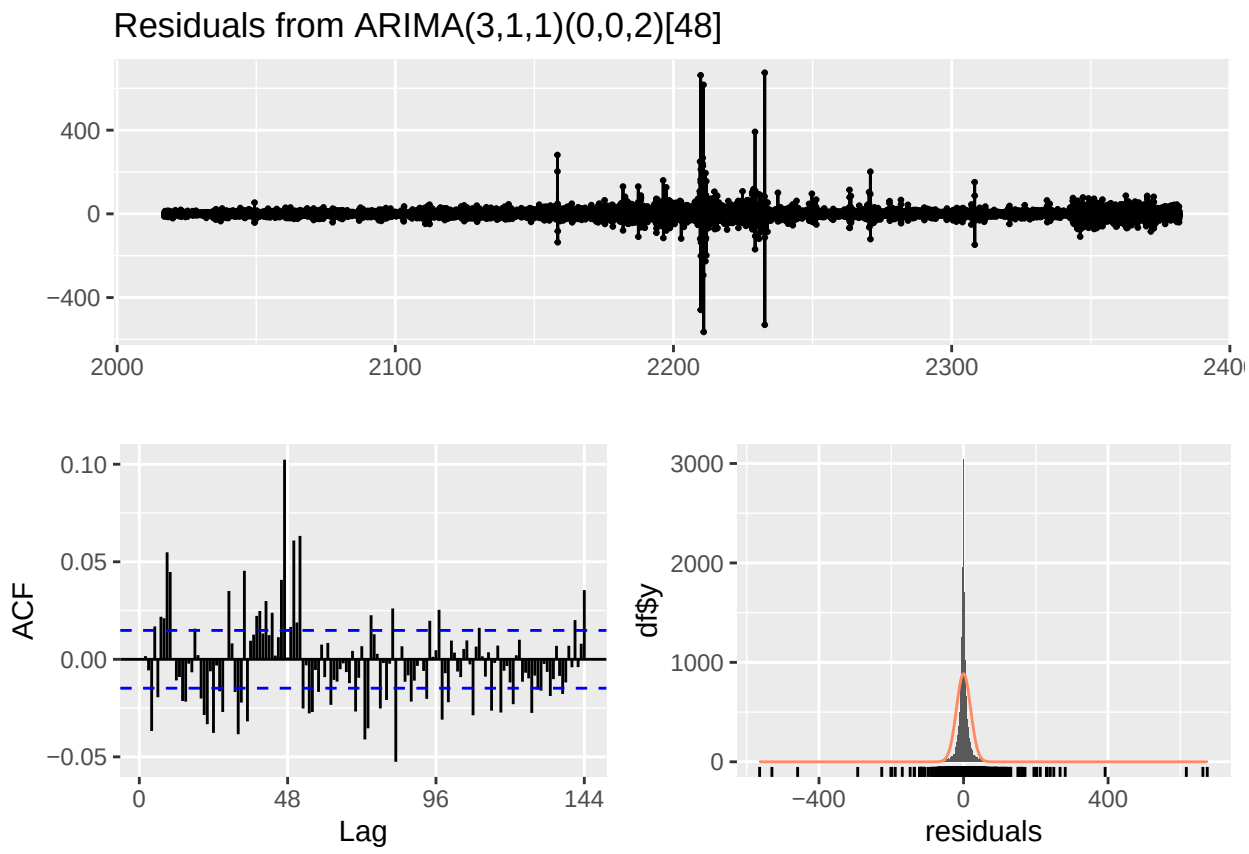
I'm going to check the residuals of the ARIMA and STL-ARIMA models and do some model refinement. I will not be continuing with the worst performing models ARIMAX and TBATS.

Structural checks and Statistical diagnostics

ARIMA

Autocorrelation check with Ljung Box test, acf and pacf plots:

```
checkresiduals(ARIMA_model1)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(3,1,1)(0,0,2)[48]
## Q* = 1035.3, df = 90, p-value < 2.2e-16
##
## Model df: 6.   Total lags used: 96
```

The p-value is extremely small I reject the null hypothesis that there is no autocorrelation in the residuals and conclude that there is autocorrelation in the residuals. This means that the TBATS model hasn't fully captured the time-dependent structure in the series.

Anderson-Darling normality test:

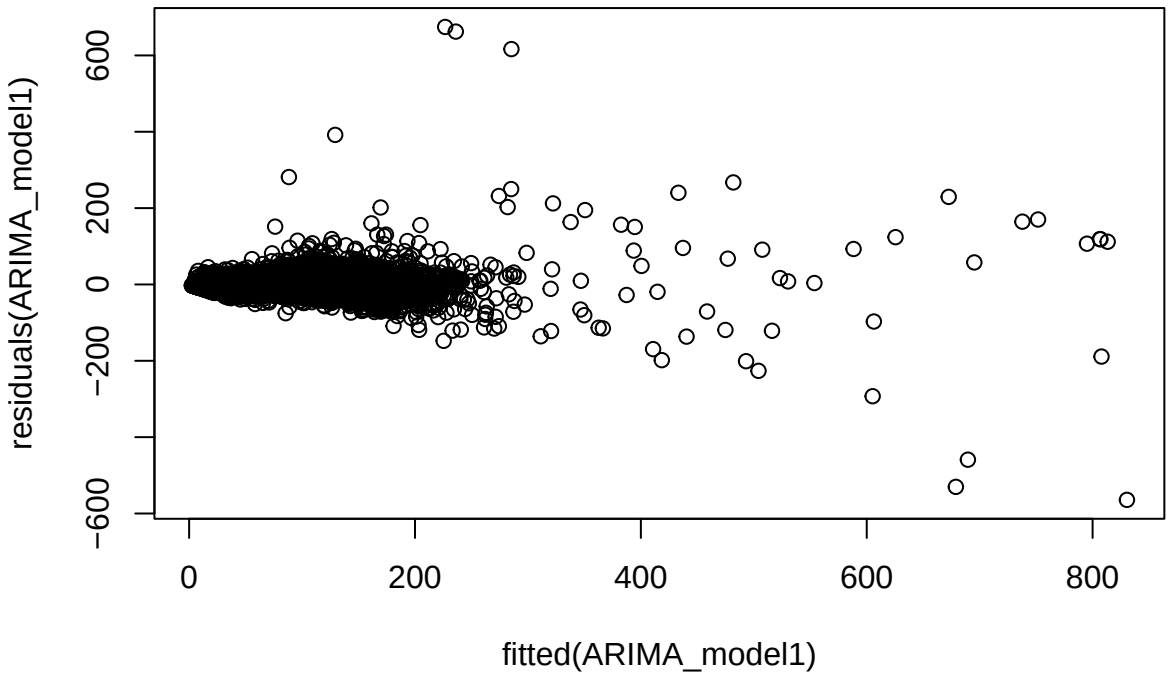
```
ad.test(residuals(ARIMA_model1))
```

```
##
##  Anderson-Darling normality test
##
## data:  residuals(ARIMA_model1)
## A = 1605.6, p-value < 2.2e-16
```

The p-value is very small, I reject the null hypothesis that the residuals are normally distributed and conclude that the residuals are non-normal.

Checking for constant variance and mean zero in plot:

```r
plot(residuals(ARIMA_model1) ~ fitted(ARIMA_model1))
```



As the fitted values increase, the residuals fan out, showing increasing variance.

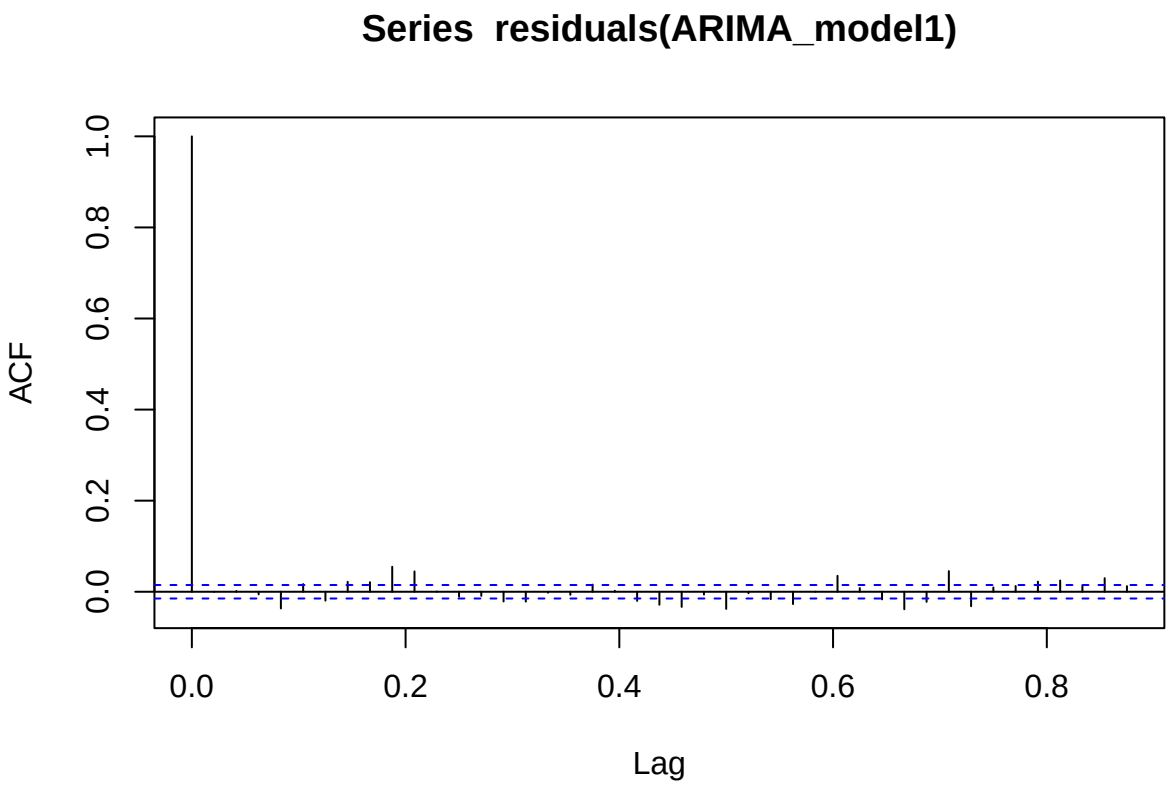
There's a tight cluster around zero residuals for lower fitted values.

Checking mean is reasonably close to zero:

```r
mean(residuals(ARIMA_model1))
```

```
## [1] 0.02386625
```

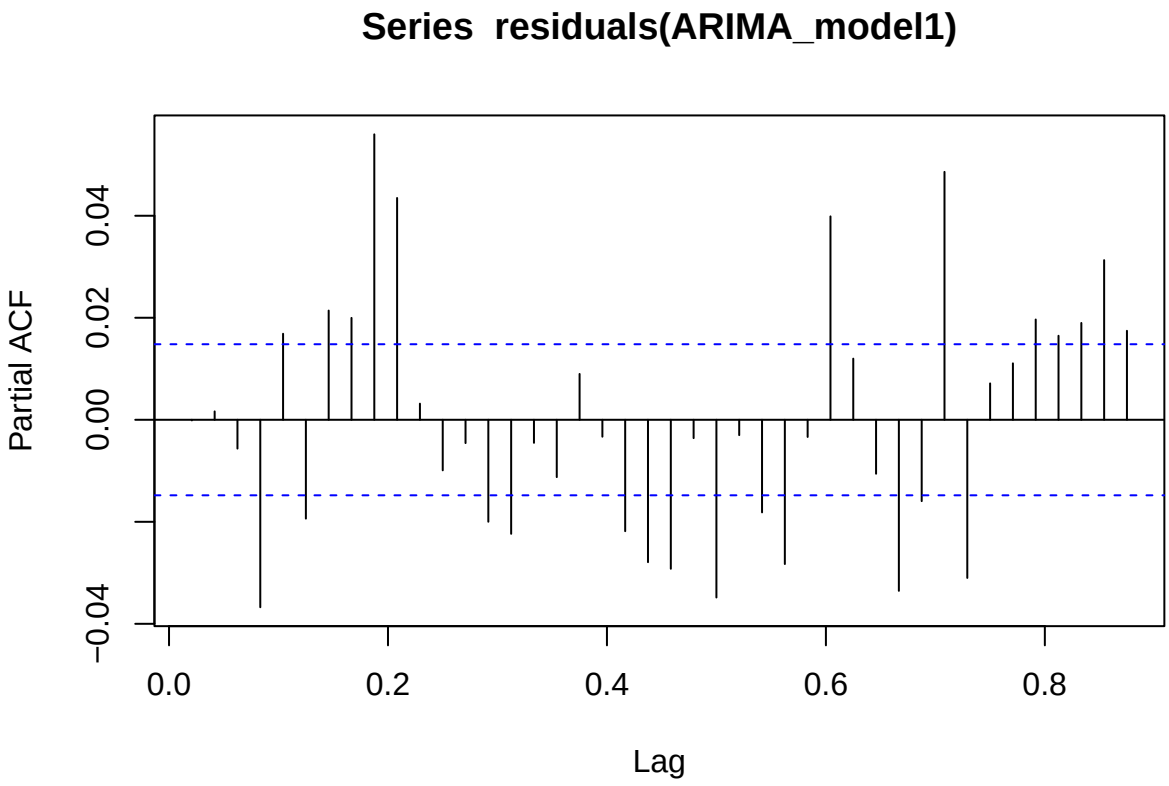
```r
acf(residuals(ARIMA_model1))
```



Most autocorrelation bars fall within the dashed blue confidence bands, suggesting they're not statistically significant.

There's no strong pattern of lingering autocorrelation. The residuals are behaving like white noise.

```
pacf(residuals(ARIMA_model1))
```

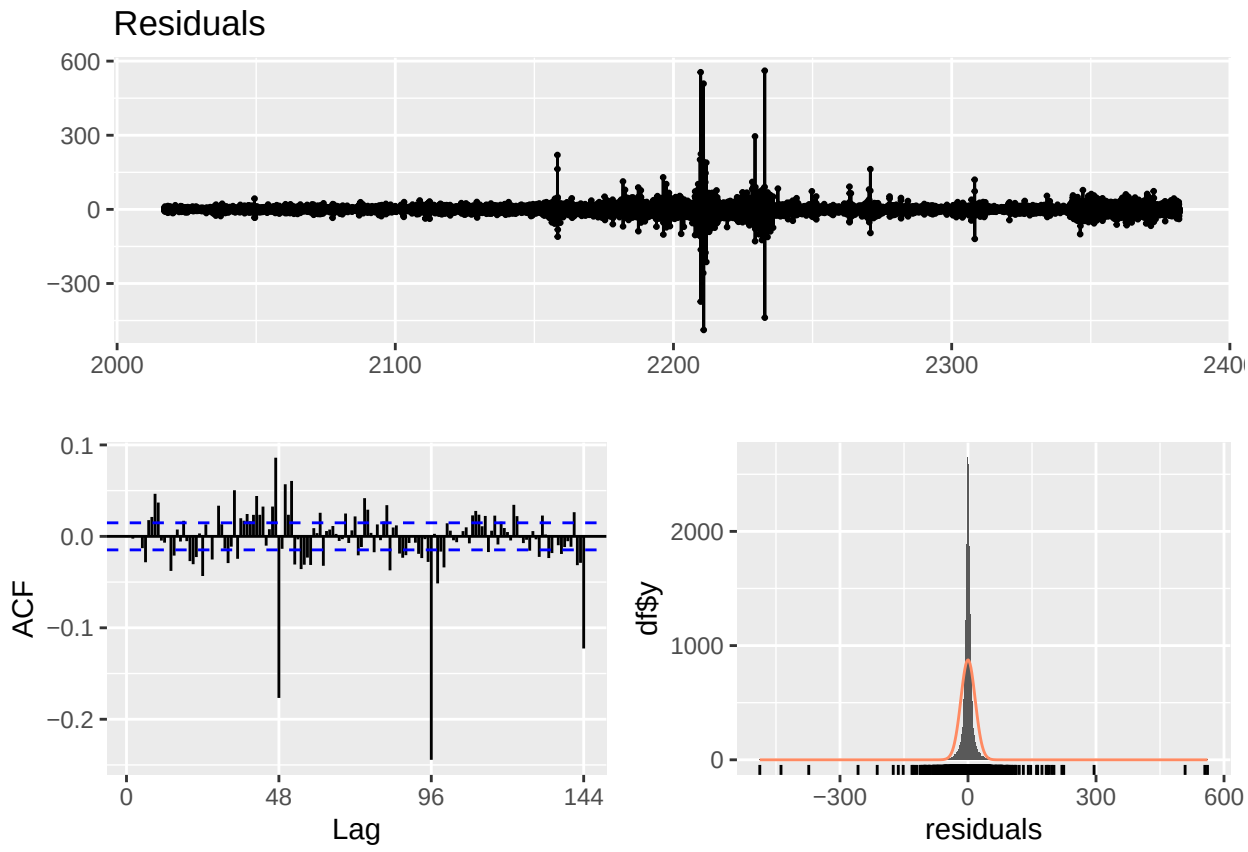


Most lags fall within the blue dashed confidence bands, meaning their partial autocorrelations aren't statistically significant.

There are a few spikes outside of the confidence bounds, particularly at lag 1 suggesting hort-term autocorrelation. This implies that the AR term (p) is slightly under-specified. I am going to try increasing the AR component to try and capture the remaining autocorrelation.

STL-ARIMA

```
checkresiduals(STL_ARIMA_model1)
```




```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 2617.1, df = 96, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 96
```

The p-value is extremely small I reject the null hypothesis that there is no autocorrelation in the residuals and conclude that there is autocorrelation in the residuals. This means that the TBATS model hasn't fully captured the time-dependent structure in the series.

Anderson-Darling normality test:

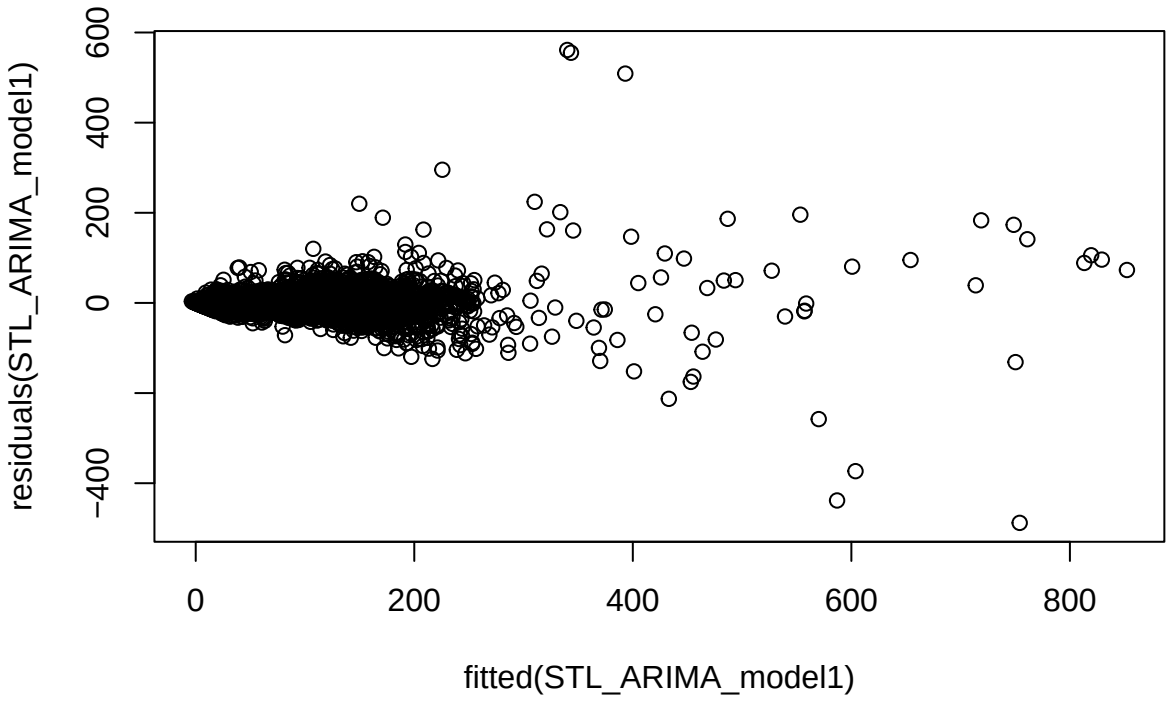
```
ad.test(residuals(STL_ARIMA_model1))
```

```
##
##  Anderson-Darling normality test
##
## data:  residuals(STL_ARIMA_model1)
## A = 1478.8, p-value < 2.2e-16
```

The p-value is very small, I reject the null hypothesis that the residuals are normally distributed and conclude that the residuals are non-normal.

Checking for constant variance and mean zero in plot:

```r
plot(residuals(STL_ARIMA_model1) ~ fitted(STL_ARIMA_model1))
```



As the fitted values increase, the residuals fan out, showing increasing variance.

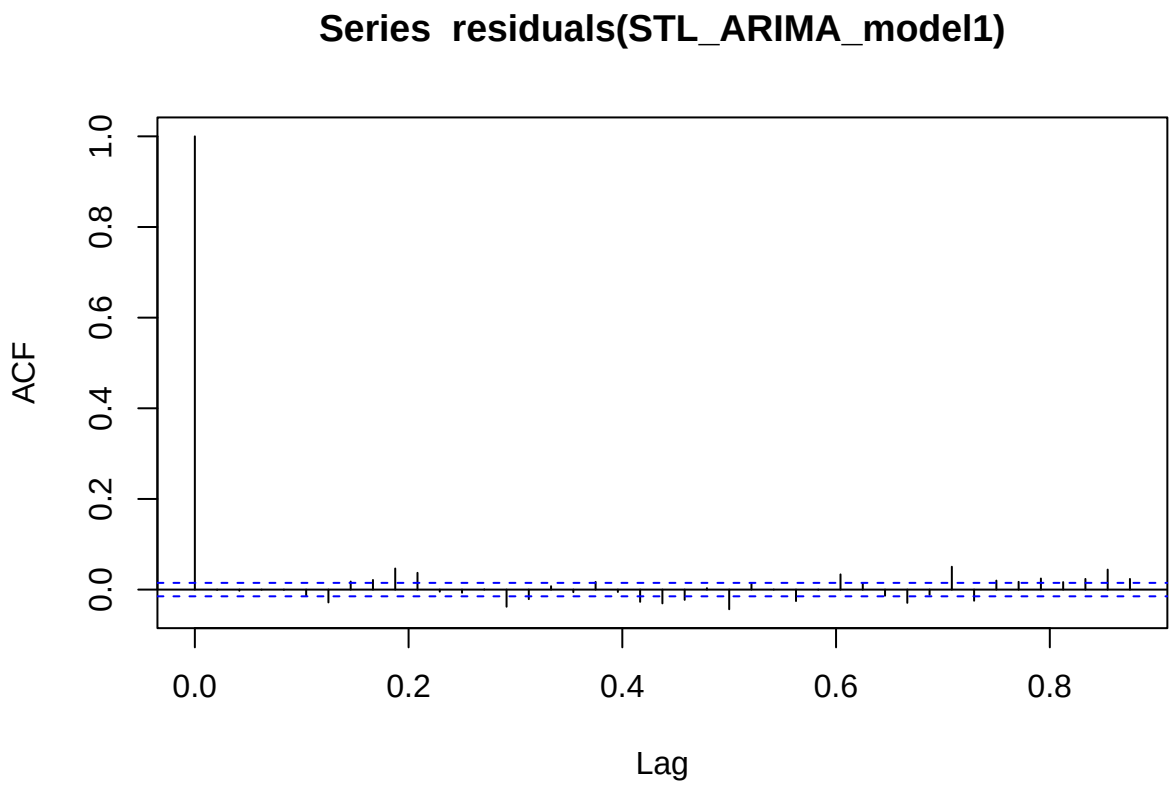
There's a tight cluster around zero residuals for lower fitted values.

Checking mean is reasonably close to zero:

```r
mean(residuals(STL_ARIMA_model1))
```

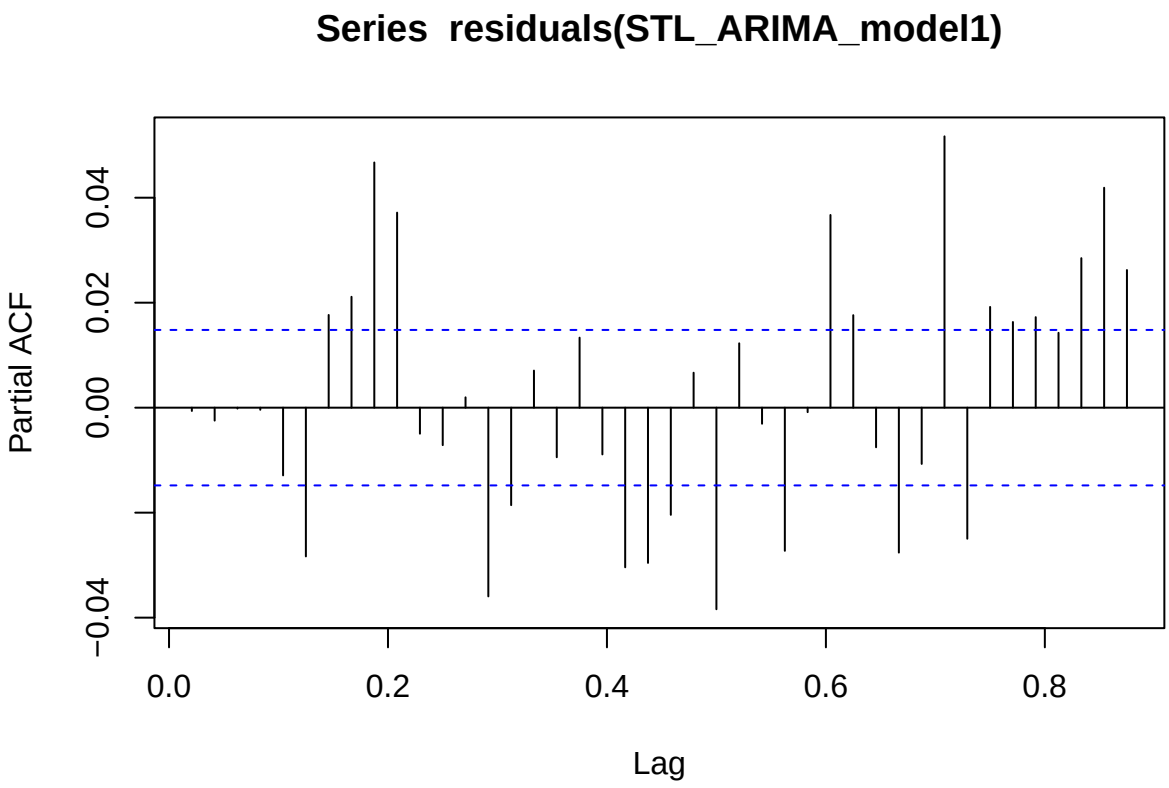
```
## [1] 0.01559731
```

```r
acf(residuals(STL_ARIMA_model1))
```



The models residuals are like white noise, most of the bars are within the confidence bounds.

```
pacf(residuals(STL_ARIMA_model1))
```



There are significant spikes outside the confidence bands, this suggests that there are lags that aren't captured by my current AR structure.

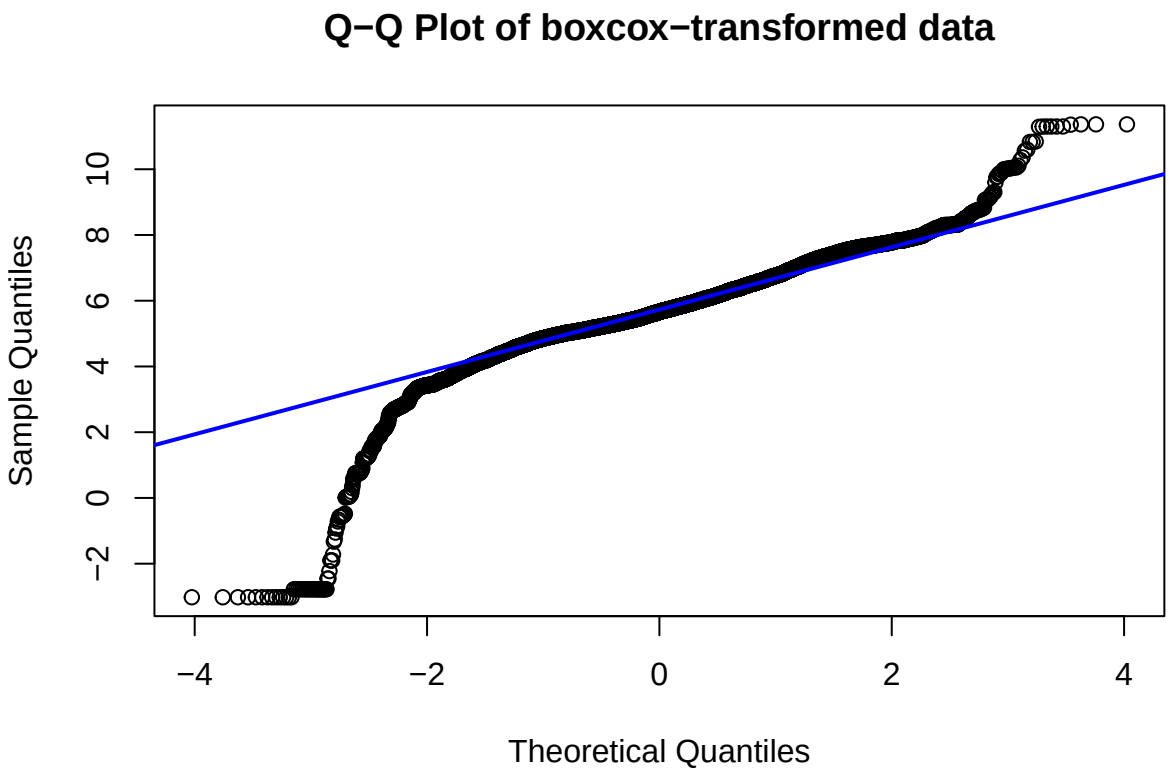
I am going to increase the AR component for the model.

Model refinement

Applying a box cox transformation to the data to deal with non-normality:

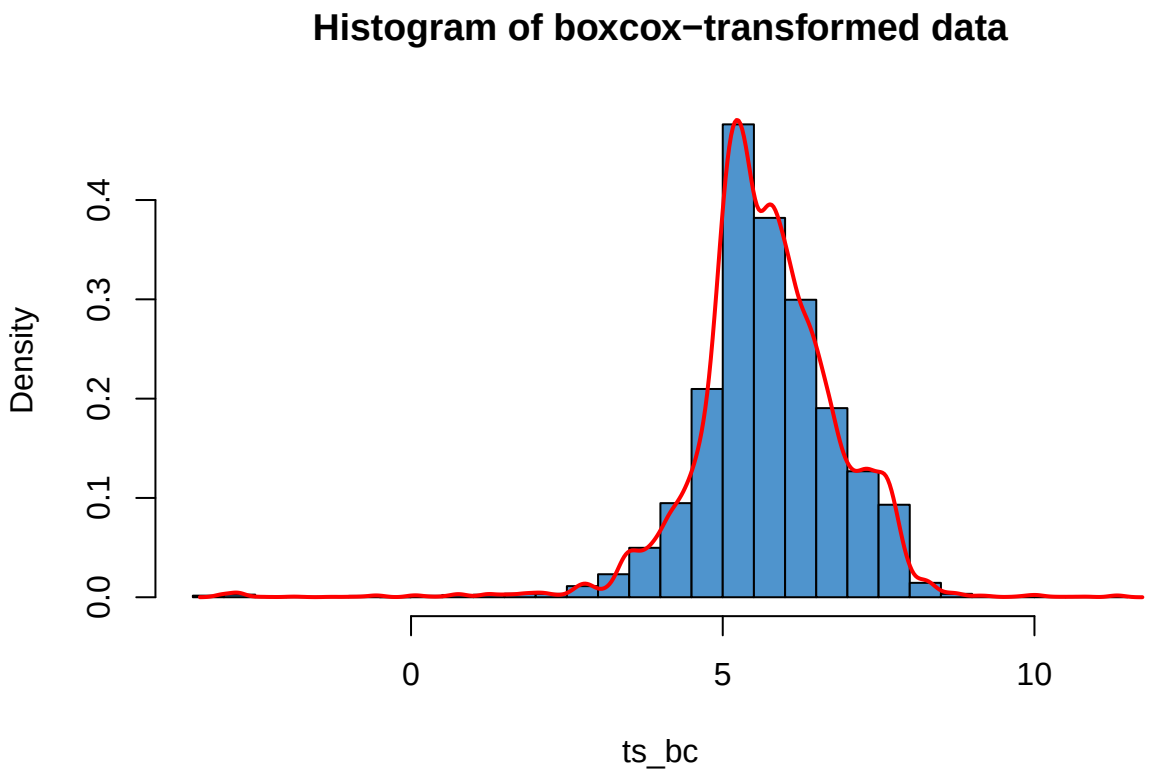
```
lambda <- BoxCox.lambda(time_series_data)
ts_bc <- BoxCox(time_series_data, lambda)
```

```
qqnorm(ts_bc, main = "Q-Q Plot of boxcox-transformed data")
qqline(ts_bc, col = "blue", lwd = 2)
```



There is a curved pattern in the data which indicates that the data is skewed or is not normally distributed.

```
hist(ts_bc, breaks = 50, probability = TRUE,
     main = "Histogram of boxcox-transformed data", col = "steelblue3")
lines(density(ts_bc), col = "red", lwd = 2)
```



The data is not centered around zero, so its not a standard normal distribution.

The data appears to be right skewed.

The data is not normally distributed, however the box-cox transformed data is closer to a normal distribution than the untransformed data, so I will be using it for further modelling.

Anderson-Darling normality test:

```
ad.test(ts_bc)
```

```
##
##  Anderson-Darling normality test
##
## data:  ts_bc
## A = 147.46, p-value < 2.2e-16
```

The small p-value indicates that the data is still not normal even after a BoxCox transformation.

Checking for outliers with Tukey's Fences:

```
y <- as.numeric(time_series_data)

# Computing the basic boxplot stats
stats <- boxplot.stats(y)$stats  # Q1, median, Q3, ...
iqr   <- stats[3] - stats[1]     # Q3 - Q1
```

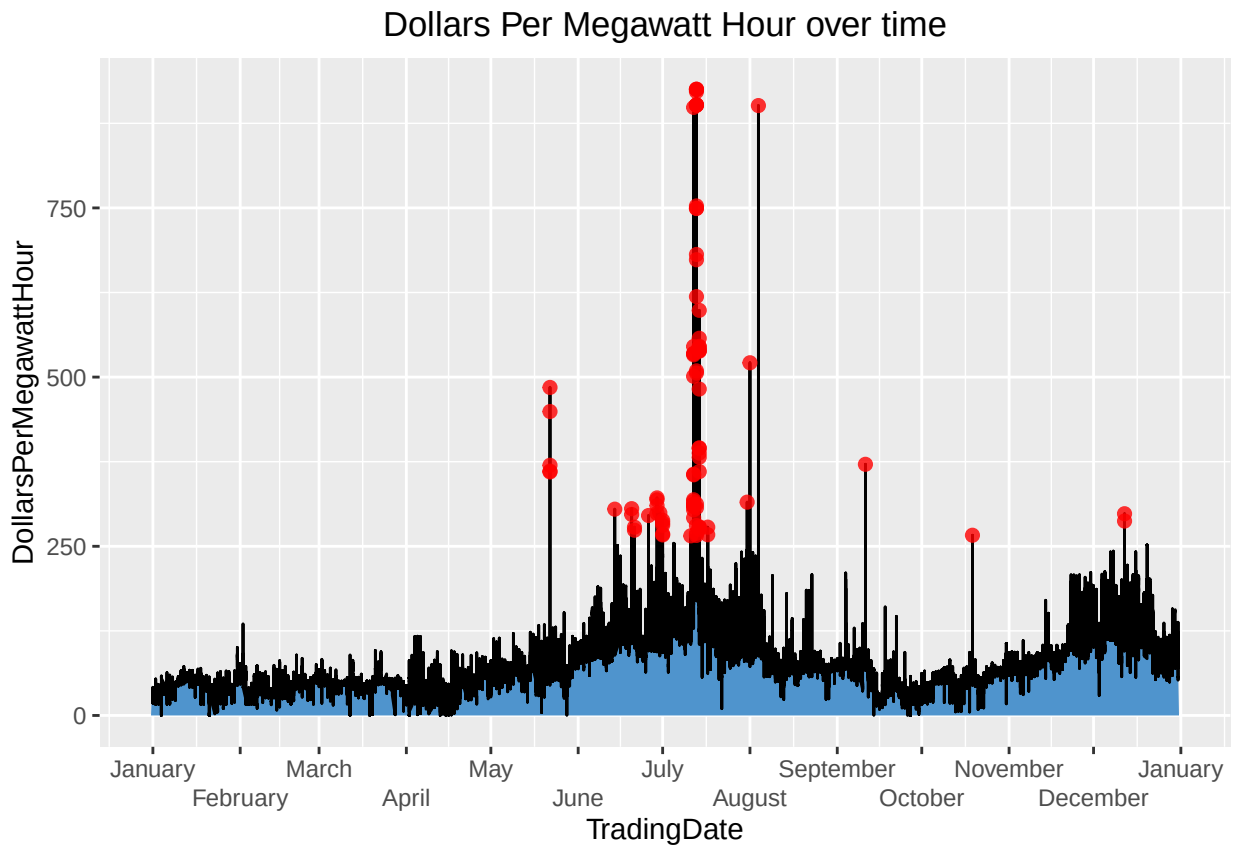
```r
# Defining the outer fences
lower <- stats[1] - 3*iqr
upper <- stats[3] + 3*iqr

# Finding the outliers
outliers_global <- which(y < lower | y > upper)

length(outliers_global)
```

```
## [1] 84
```

There are 84 outliers according to Tukey's Fences.

Visualizing the outliers on a time series plot:

```r
ggplot(Training_set, aes(x = TradingDate, y = DollarsPerMegawattHour)) +
  geom_area(fill = "steelblue3")+
  geom_line() +
  scale_x_date(date_breaks = "1 month", date_labels = "%B", guide = guide_axis(n.dodge = 2)) +
  ggtitle("Dollars Per Megawatt Hour over time") +
  # Adding red points for outliers
  geom_point(data = Training_set[outliers_global, ], aes(x = TradingDate, y = DollarsPerMegawattHour),
             color = "red", size = 2, alpha = 0.8) +
  theme(plot.title = element_text(hjust = 0.5))
```



There are outliers in the series but since energy prices are so volatile, I don't want to remove them, I am choosing not to discard the outliers in the series.

Refitting models:

For both models I have increased AR component by 1, and applied a box-cox transformation to the time series data.

Fitting ARIMA:

```
ARIMA_model2 <- Arima(ts_bc, order = c(4,1,1), # Non-seasonal ARIMA(p,d,q)
             seasonal = list(order = c(0, 0, 2), period = 48),  # Seasonal ARIMA(P,D,Q)[S]
             include.constant = FALSE)  # Excluding constant since first order differencing is used
```

Fitting an STL - ARIMA to seasonally adjusted series:

```
STL_ARIMA_model2 <- stlm(ts_bc, s.window = 48, robust = TRUE,
                         modelfunction = function(x) Arima(x, order = c(6, 1, 1)))
```

Final Forecasting and reporting

I'm going to forecast 48 trading periods for six months

ARIMA:

```
forecast12months_A <- forecast(ARIMA_model2, h = 17520)
```

Returning to the original scale (boxcox scale was used for training the ARIMA model) after forecasting to obtain interpretable electricity price predictions.

```
forecast_raw_A <- InvBoxCox(forecast12months_A$mean, lambda) - 1
```

```
forecast_raw_ARIMA2 <- ts(forecast_raw_A, start = c(2018, 1), frequency = 48)
```

STL-ARIMA:

```
forecast12months_S <- forecast(STL_ARIMA_model2, h = 17520)
```

Returning to the original scale (boxcox scale was used for training the STL-ARIMA model) after forecasting to obtain interpretable electricity price predictions.

```
forecast_raw_S <- InvBoxCox(forecast12months_S$mean, lambda) - 1
```

```
forecast_raw_STL2 <- ts(forecast_raw_S, start = c(2018, 1), frequency = 48)
```

Overall Model Comparison

```
ARIMA_acc2 <- accuracy(forecast_raw_ARIMA2, Test_time_series)
STL_ARIMA_acc2 <- accuracy(forecast_raw_STL2, Test_time_series)
```

```r
acc_list2 <- list(ARIMA = ARIMA_acc, ARIMA_2 = ARIMA_acc2, STL_ARIMA = STL_ARIMA_acc, STL_ARIMA_2 = STL_

acc_df2 <- map_df(acc_list2, ~ as.data.frame(.x)["Test set", ], .id = "Model")

acc_df2
```

```
##                    Model        ME      RMSE      MAE          MPE       MAPE
## Test set...1       ARIMA  6.130026  18.29114 12.67222     3.125512   14.33552
## Test set...2     ARIMA_2 18.066604 102.20341 52.87934 -2087.006715 2115.22873
## Test set...3   STL_ARIMA 11.084312  21.15245 16.55240     8.990807   18.40260
## Test set...4 STL_ARIMA_2 18.535612 101.72906 52.66573 -1987.800118 2016.86837
##                    MASE      ACF1 Theil's U
## Test set...1 0.6497836 0.6247587  1.324960
## Test set...2        NA 0.9063950 12.286920
## Test set...3 0.8487441 0.6316401  1.486087
## Test set...4        NA 0.9055993 11.649619
```

The MASE could not be calculated for my two most recent models presumably due to the box-cox transformation I applied to the time series for those models, so I will be focusing on the other available metrics.

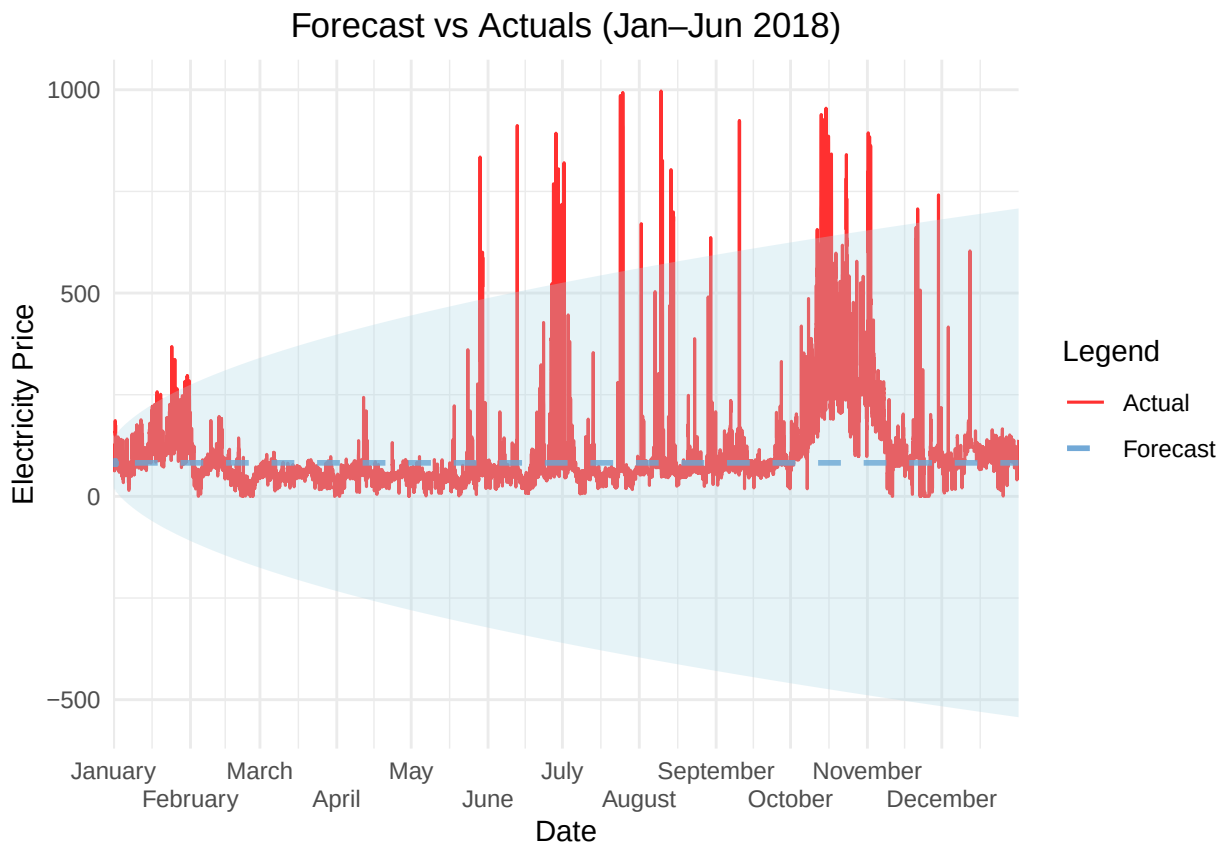
The model that performed best was my original ARIMA model, my second best model was my original STL_ARIMA model.

It seems my refinement of the models made their performance worse.

The best model is ARIMA and the worst model is STL-ARIMA 2

```r
start_time     <- as.POSIXct("2018-01-01 00:00:00", tz = "UTC")
times_forecast <- seq(start_time, by = "30 min", length.out = 17520)

Test_time_series2 <- ts(test_data$DollarsPerMegawattHour, frequency = 48, start = c(2018, 1))
df_actual <- data.frame(Date = times_forecast, Value = as.numeric(Test_time_series2), Type = "Actual")

df_forecast <- data.frame(
  Date  = times_forecast,
  Mean  = as.numeric(forecast_raw_ARIMA$mean),
  Lower = as.numeric(forecast_raw_ARIMA$lower[,2]),  # 95% lower bound
  Upper = as.numeric(forecast_raw_ARIMA$upper[,2]),  # 95% upper bound
  Type  = "Forecast"
)
```

```r
ggplot() +
  geom_line(data = df_actual, aes(x = Date, y = Value, color = "Actual"), size = 0.55) +
  geom_line(data = df_forecast, aes(x = Date, y = Mean, color = "Forecast"), size = 1, alpha = 0.8, lin
  geom_ribbon(data = df_forecast, aes(x = Date, ymin = Lower, ymax = Upper), fill = "lightblue", alpha =
  scale_color_manual(values = c("Actual" = "firebrick1", "Forecast" = "steelblue3")) +
  scale_x_datetime(date_breaks = "1 month",  date_labels = "%B",  expand = c(0, 0), guide = guide_axis(
  labs(title = "Forecast vs Actuals (Jan-Jun 2018)",
       x = "Date", y = "Electricity Price", color = "Legend") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```



This plot shows my ARIMA model forecast for the final half hourly electricity prices for the year 2018 compared to the actual values for that year.

There are large fluctuations in the actual electricity prices, particularly in July and November.

The forecast as shown by the blue dashed lines appears to follow the overall trend of electricity prices. Its 95% confidence interval (the light blue ribbon) covers the majority of the electricity prices for 2018 but misses many large spikes between June and December which means the model is underestimating the real volatility.

However it does model the initial five months January to May quite well.

Due to the models issues with volatility if I were to continue forecasting the 2018 energy price data with another model I would consider a GARCH model as it is good for modelling volatility clustering.

Conclusion

The best model was the ARIMA model: ARIMA(3,1,1)(0,0,2)[48].

Its non-seasonal component had AR(3) (which captured short term autocorrelation up to lag 3), I(1) (a first order differencing to make the data stationary), MA(1) (which corrected for noise and shocks using lag 1 residuals). Its seasonal component with a periodicity of 48 (the seasonality repeats every full day as there are 48 trading periods per day) and it did not include a seasonal AR or I (no seasonal autocorrelation or differencing) and had a seasonal MA(2) (uses forecast errors from 1 and 2 days ago (lags at 48 and 96 intervals) to adjust the current days forecast).

The models mean error was 6.13 which means on average its forecasts overestimate actual prices by about 6.13 dollars per megawatt hour.

The models root mean square error was 18.29 which indicates large forecast errors, since squared errors penalize larger errors more heavily this high RMSE value is probably due to how volatile the data is.

The models mean absolute error was 12.67 which means that forecasts were off by 12.67 units on average. This is 12.6% of the annual mean, the mean energy price for 2018 was 100.53, so the mean absolute error isn't that high considering how volatile the data is.

The mean percentage data was 3.13% which means that the model tends to forecast higher electricity prices than reality.

The mean absolute percentage error was 14.34% which means that the models forecasts were off by 14.34% on average.

The models MASE was 0.65 which is a scaled comparison to a naive model which means the ARIMA model outperforms a naive forecast.

The models ACF1 was 0.65 which indicates that the residuals are moderately autocorrelated so there is some structure or pattern in the data that the model did not capture.

The ARIMA model did better than the STL-ARIMA, ARIMAX and TBATS models which just shows that a more complex model does not equal a better performance, those other models were overfitting on the training data and not generalizing well enough to forecast the following year.

References and Citations

Electricity Authority. (n.d.). Final energy prices by month [Dataset]. EMI – Electricity Market Information. Retrieved between May 2 and July 13, 2025, from https://www.emi.ea.govt.nz/Wholesale/Datasets/DispatchAndPricing/FinalEnergyPrices/ByMonth