Energy Price Forecasting: Model Validation

J Alexandra

2025-07-17

Libraries

```
suppressPackageStartupMessages({
library(dplyr)
library(tidyr)
library(tseries)
library(nortest)
library(purrr)
library(FinTS)
library(rugarch)
library(ggplot2)
library(reshape2)
library(forecast)
  })
```

Ensuring reproducibility of results:

```
set.seed(48)
```

Loading models and data

```
ARIMA_model1 <- readRDS("models/Initial models from training/arima_model.rds")
SARIMAX_model_fx <- readRDS("models/Initial models from training/arima_with_xreg.rds")
SARIMAX_model1 <- readRDS("models/Initial models from training/SARIMAX_model.rds")
Energy_generation_model <- readRDS("models/Initial models from training/Energy_generation_model.rds")
GARCH_model <- readRDS("models/Initial models from training/garch_model.rds")
SARIMA_model1 <- readRDS("models/Initial models from training/sarima_model.rds")
STL_ARIMA_model1 <- readRDS("models/Initial models from training/stl_arima_model.rds")
TBATS_model1 <- readRDS("models/Initial models from training/Tbats_model.rds")
```

```
time_series_data <- read.csv("Saved_Datasets/timeseries.csv")
Training_set <- read.csv("Saved_Datasets/Training_set.csv")

Val_time_series <- read.csv("Saved_Datasets/Val_time_series.csv")
Val_set <- read.csv("Saved_Datasets/Val_set.csv")
xreg <- read.csv("Saved_Datasets/xreg.csv")
```

```r
xreg_future <- read.csv("Saved_Datasets/xreg_future.csv")

xreg <- as.matrix(xreg)
xreg_future <- as.matrix(xreg_future)
```

```r
time_series_data <- ts(time_series_data$Value, start = c(2017, 1), frequency = 48)
Val_time_series <- ts(Val_time_series$Value, start = c(2018, 1), frequency = 48)
First_order_diff <- diff(time_series_data)
```

Structural checks and Statistical diagnostics: Evaluating residuals

ARIMA

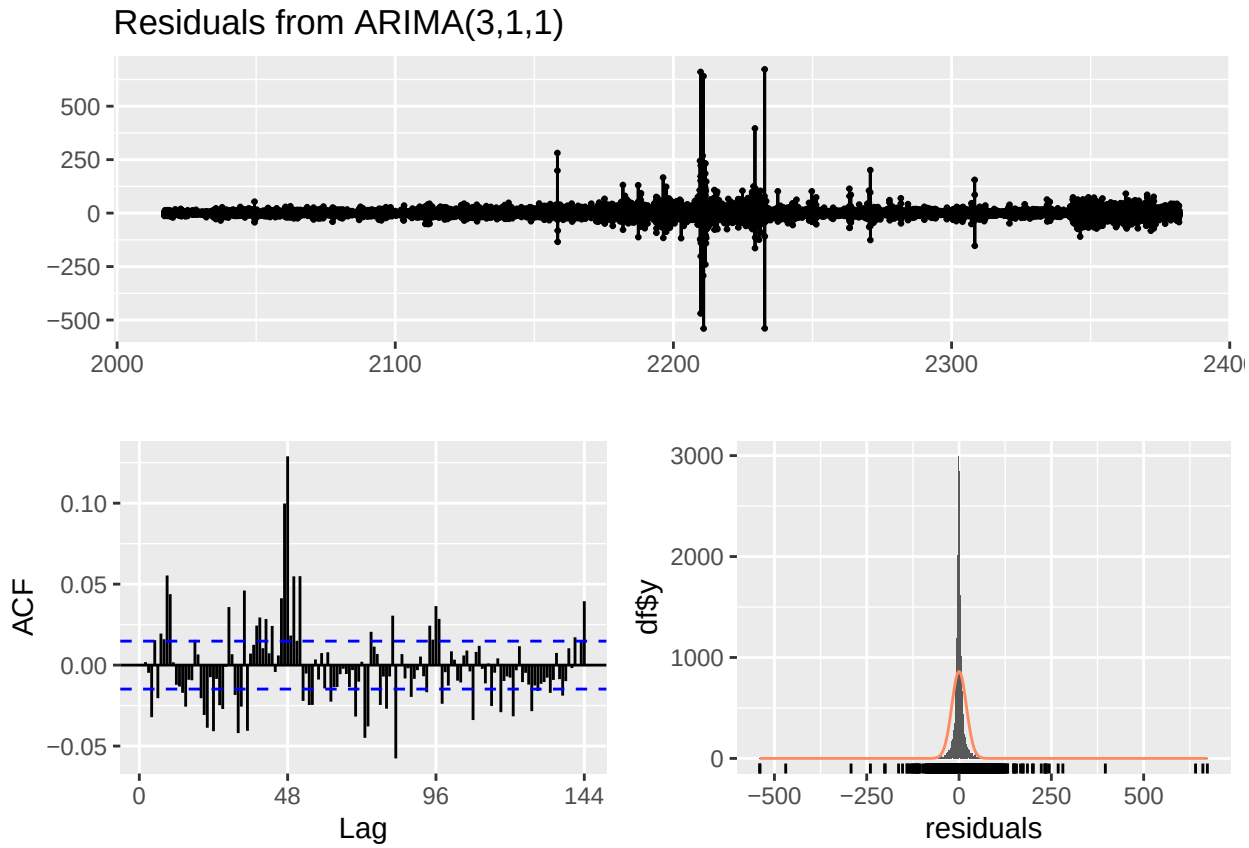
Ljung-Box test:

Null hypothesis: There is no autocorrelation in the residuals.

Alternative hypothesis: There is autocorrelation in the residuals.

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

```r
checkresiduals(ARIMA_model1)
```

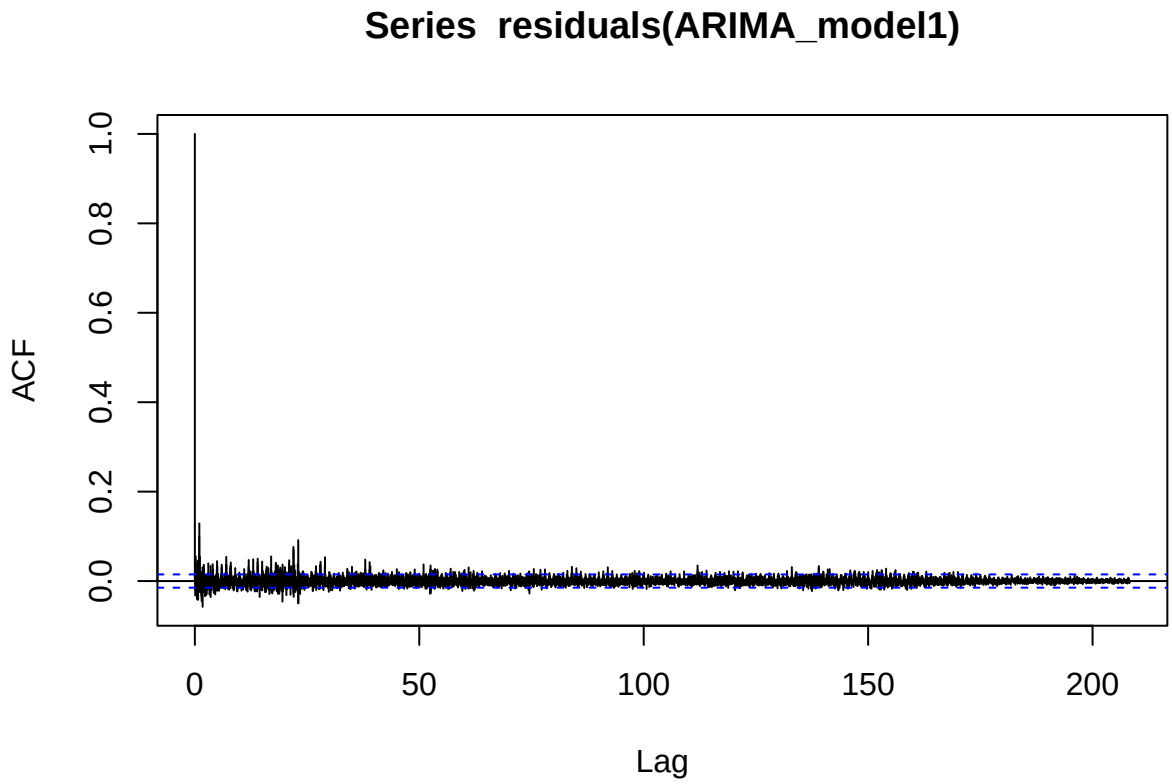


##

```
##  Ljung-Box test
##
## data:  Residuals from ARIMA(3,1,1)
## Q* = 1375.7, df = 92, p-value < 2.2e-16
##
## Model df: 4.   Total lags used: 96
```

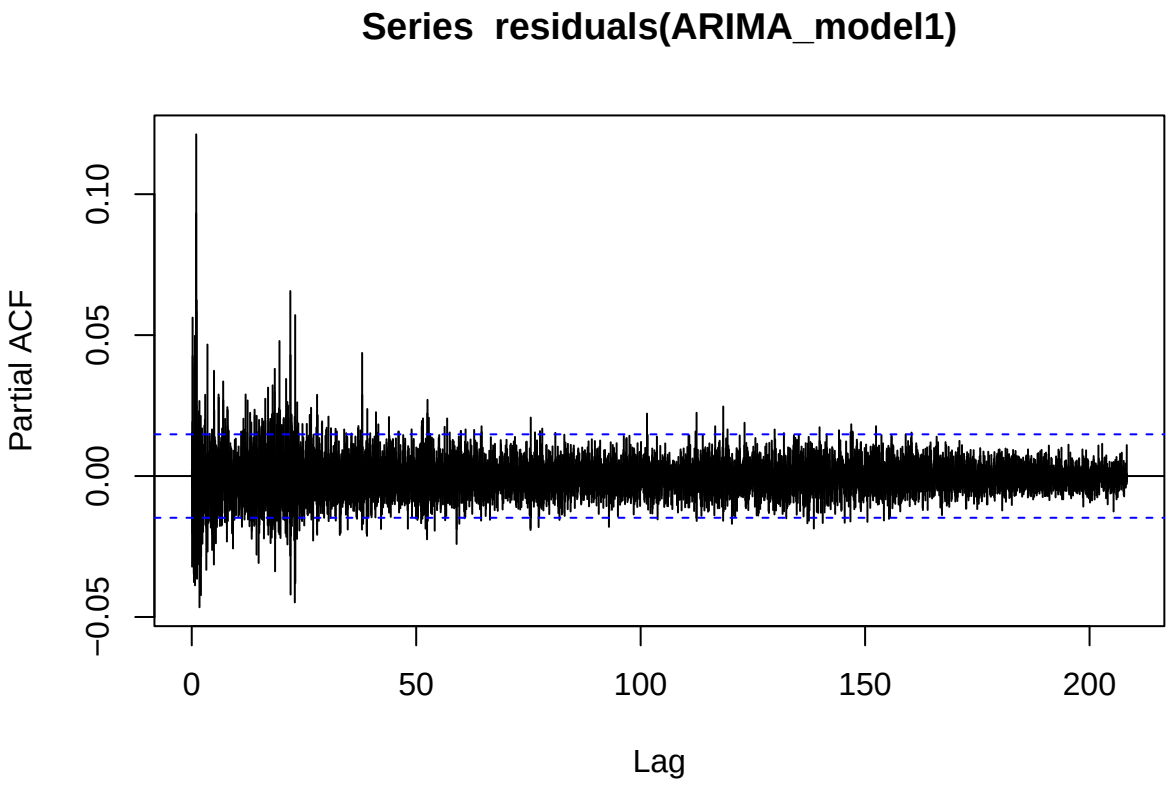
The p-value is much smaller than the significance level of 0.05, I reject the null hypothesis and conclude that there is autocorrelation in the residuals.

```
acf(residuals(ARIMA_model1), lag.max = 10000)
```



There are a lot of spikes outside the confidence bounds between lags 0 and 25 particularly, this gradually tapers off as the lags increase. Due to this I will be increasing the MA terms.

```
pacf(residuals(ARIMA_model1), lag.max = 10000)
```



In the partial autocorrelation plot, there are many spikes that are outside the confidence bounds, particularly at around lag 1, lag 25 and lag 48. This then gradually tapers off. This means that there is still some autocorrelation in the residuals, this is supported by the ACF1 value of 0.65 from the previous evaluation of the model on the validation set. Due to this I will be increasing the AR term.

Anderson-Darling normality test:

Null hypothesis: The residuals follow a normal distribution.

Alternative hypothesis: The residuals deviate significantly from a normal distribution.

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

```
##
##  Anderson-Darling normality test
##
## data:  residuals(ARIMA_model1)
## A = 1609.7, p-value < 2.2e-16
```

The p-value is much smaller than the significance level of 0.05, I reject the null hypothesis and conclude that the residuals deviate significantly from a normal distribution.

Since the residuals deviate significantly from normality, the ARIMA models forecast uncertainty estimates could be inaccurate (too narrow or too wide).

However ARIMA can produce decent forecasts without perfectly normal residuals, so this isn't extremely concerning.

ARCH test on residuals:

Null Hypothesis: No ARCH effects in the residuals (the variance is constant over time (homoscedastic)).

Alternative Hypothesis: ARCH effects are present (the residual variance changes over time and depends on past squared residuals (heteroskedasticity)).

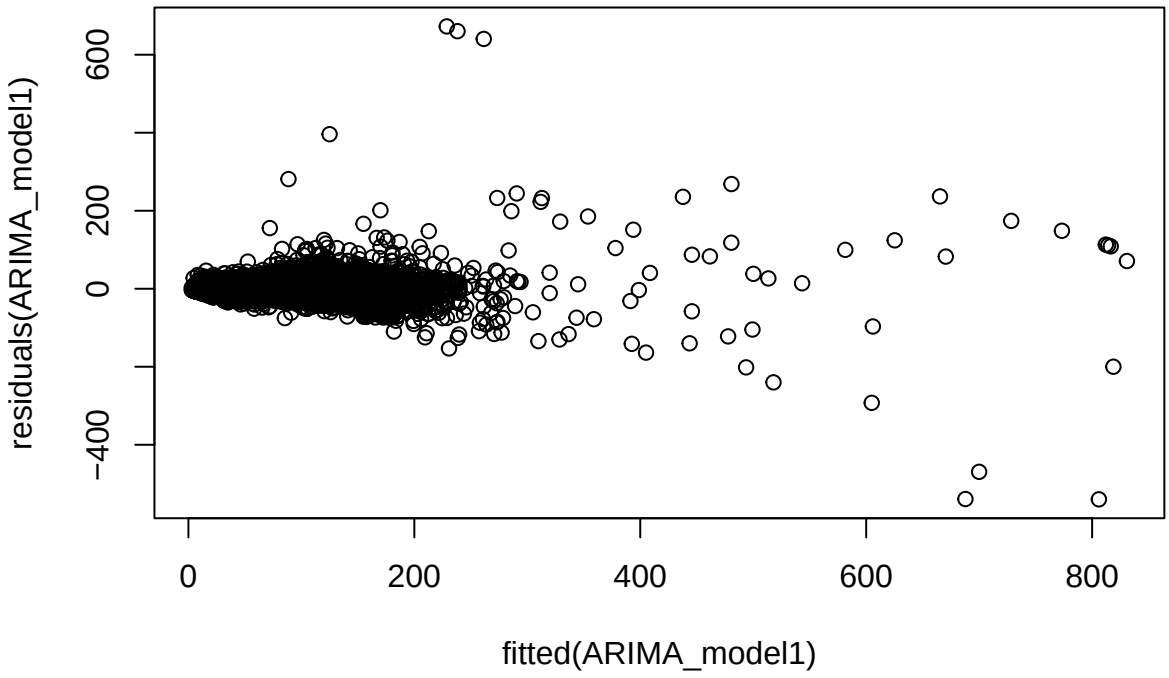
```
ArchTest(residuals(ARIMA_model1), lags = 48)
```

```
##
##  ARCH LM-test; Null hypothesis: no ARCH effects
##
## data:  residuals(ARIMA_model1)
## Chi-squared = 3242.2, df = 48, p-value < 2.2e-16
```

The p-value is much smaller than the significance level of 0.05, I reject the null hypothesis and conclude that there are ARCH effects present in the residuals (the residual variance changes over time and depends on past squared residuals (heteroskedasticity). The assumption of constant variance for the ARIMA model is violated, meaning that the models forecast confidence intervals may be inaccurate.

Checking for constant variance in plot:

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



As the fitted values increase, the residuals fan out, showing increasing variance. This plot supports the results from the ARCH test.

Checking the mean of the residuals is reasonably close to zero with a one sample t-test:

```r
t.test(residuals(ARIMA_model1), mu = 0)
```

```
##
##  One Sample t-test
##
## data:  residuals(ARIMA_model1)
## t = 0.18465, df = 17519, p-value = 0.8535
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.2638915  0.3187828
## sample estimates:
##  mean of x
## 0.02744566
```

The p-value is much larger than the significance level of 0.05 which means that I fail to reject the null hypothesis and conclude that the mean of the residuals is reasonably close to zero.

Checking that the time series is invertible (eg. if its errors can be represented as a weighted sum of past observations)

```r
ARIMA_model1$coef
```

```
##         ar1         ar2         ar3         ma1
##  0.68309798  0.14722539 -0.03951882 -0.97654980
```

The model has one MA term ma1.

```r
ma_coefs <- c(1, -ARIMA_model1$coef["ma1"])  # Inverting sign for root checking
Mod(polyroot(ma_coefs))  # Gives modulus of each root (polyroot gives the complex roots)
```

```
## [1] 1.024013
```

The root of the MA polynomial is larger than 1, which means that the root is outside of the unit circle, this means that the model is invertible. The ARIMA model satisfies the invertibility condition for both the MA component.

Based on the results of these tests I've decided to adjust the ARIMA model in the following ways:

I am going to apply a Box-Cox transformation to my time series prior to fitting ARIMA to stabilize the variance.

I'm going to use robust standard errors as they ar less sensitive to violations of constant variance and non-normal residuals.

I am going to increase the orders of the AR and MA components because the Ljung box test shows that there is still autocorrelation in the residuals, which means the ARIMA model is not capturing all the underlying structure/patterns in the data.

SARIMA

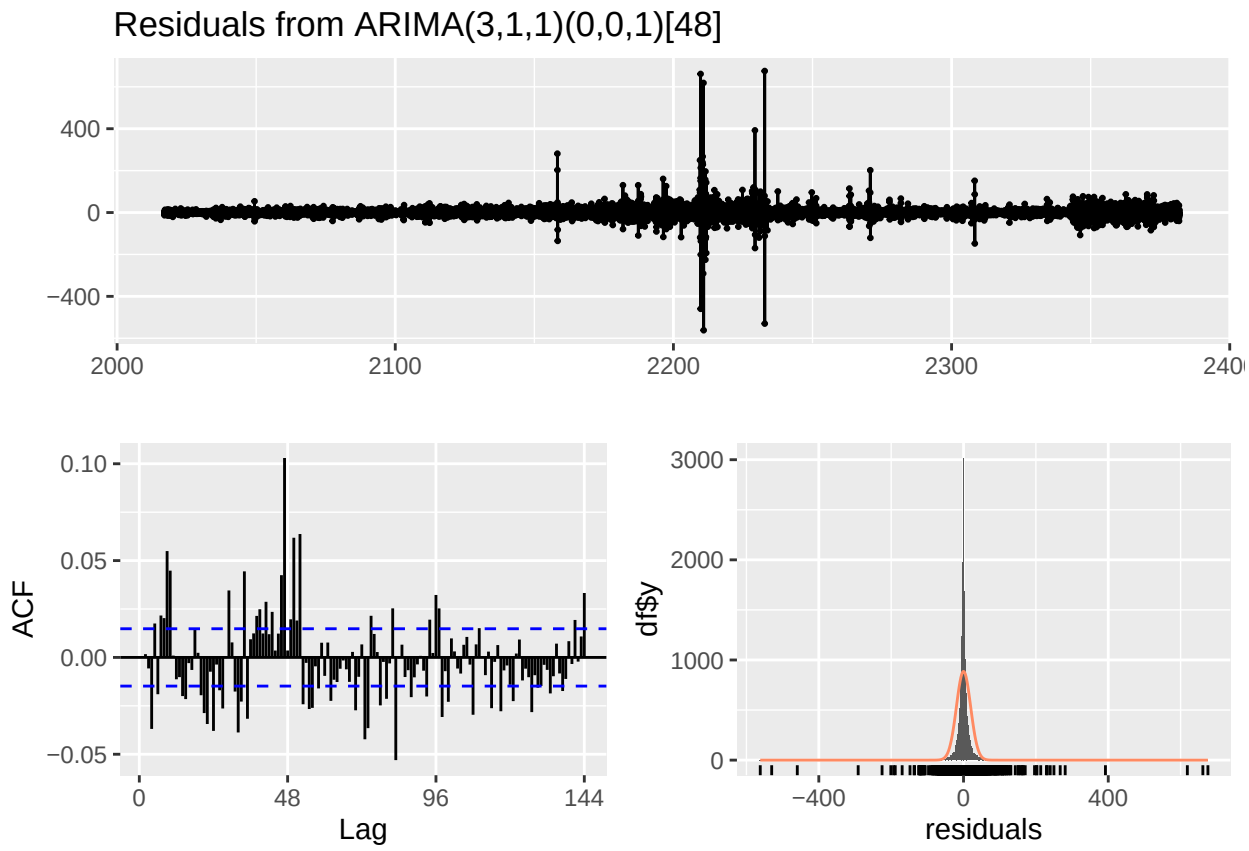
Ljung-Box test:

Null hypothesis: There is no autocorrelation in the residuals.

Alternative hypothesis: There is autocorrelation in the residuals.

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

```
checkresiduals(SARIMA_model1)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(3,1,1)(0,0,1)[48]
## Q* = 1058.7, df = 91, p-value < 2.2e-16
##
## Model df: 5.   Total lags used: 96
```

The p-value is extremely small, I reject the null hypothesis and conclude that there is autocorrelation in the residuals. This means that the SARIMA model hasn't fully captured the time dependent structure in the series.

Anderson-Darling normality test:

```
ad.test(residuals(SARIMA_model1))
```

```
##
##  Anderson-Darling normality test
##
## data:  residuals(SARIMA_model1)
## A = 1598.1, 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.

ARCH test on residuals:

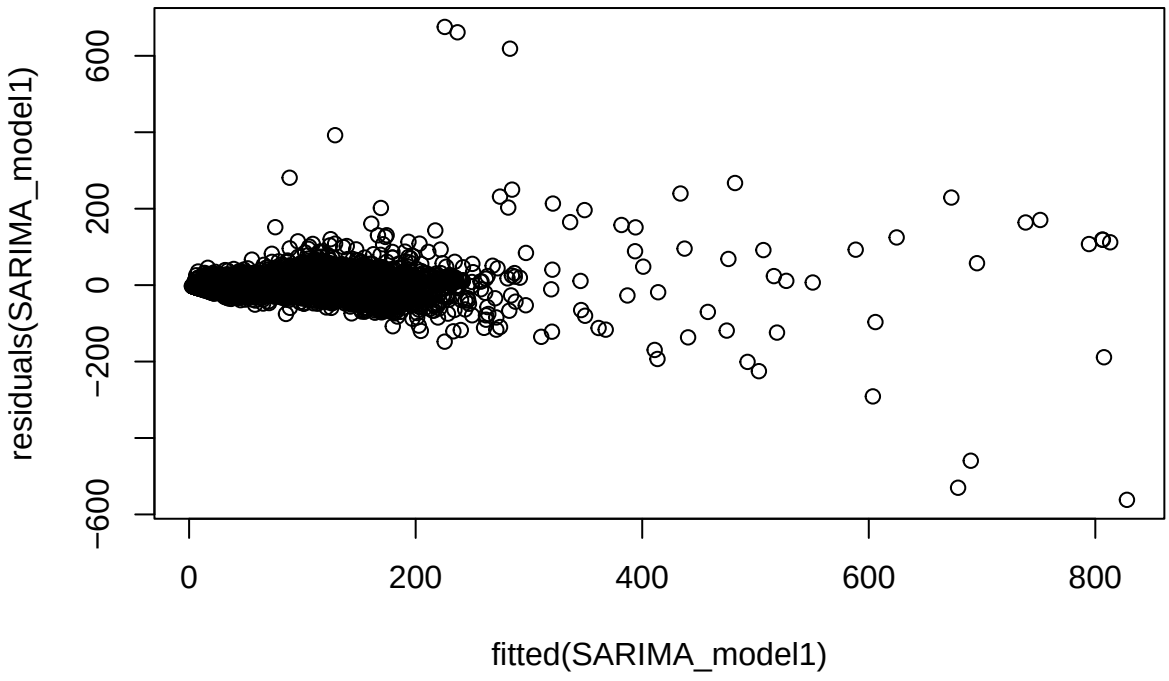
```
ArchTest(residuals(SARIMA_model1), lags = 48)
```

```
##
##  ARCH LM-test; Null hypothesis: no ARCH effects
##
## data:  residuals(SARIMA_model1)
## Chi-squared = 3166.1, df = 48, p-value < 2.2e-16
```

The p-value is much smaller than 0.05 which means I reject H0 (that there is no ARCH effect (residuals are homoskedastic)) and conclude that there is evidence of heteroskedasticity

Checking for constant variance in plot:

```
plot(residuals(SARIMA_model1) ~ fitted(SARIMA_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.

This plot supports the results from the ARCH test.

Checking the mean of the residuals is reasonably close to zero with a one sample t-test:

```r
t.test(residuals(SARIMA_model1), mu = 0)
```

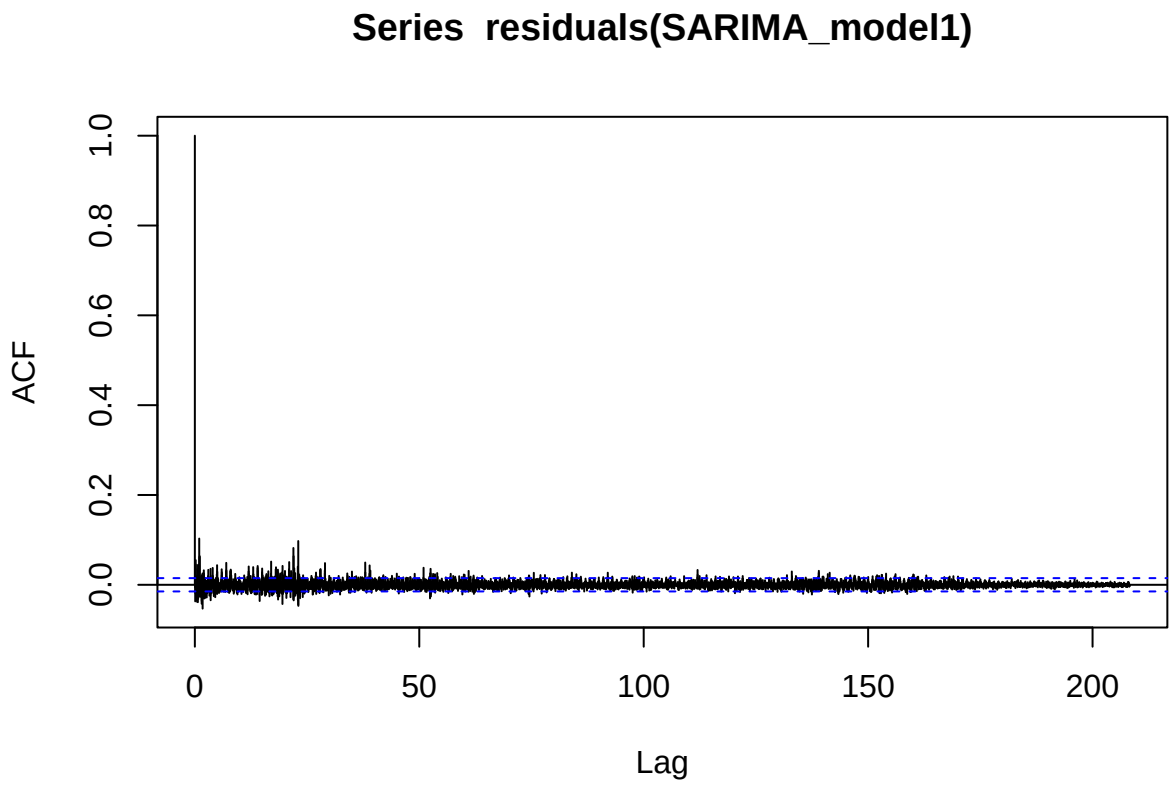
```
##
##  One Sample t-test
##
## data:  residuals(SARIMA_model1)
## t = 0.16805, df = 17519, p-value = 0.8665
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.2641790  0.3137259
## sample estimates:
##  mean of x
## 0.02477343
```

The mean of the residuals is 0.025

The p-value of the one sample t-test is 0.8665, there is not enough evidence to reject the null hypothesis, I conclude that the mean of the residuals is equal to zero.

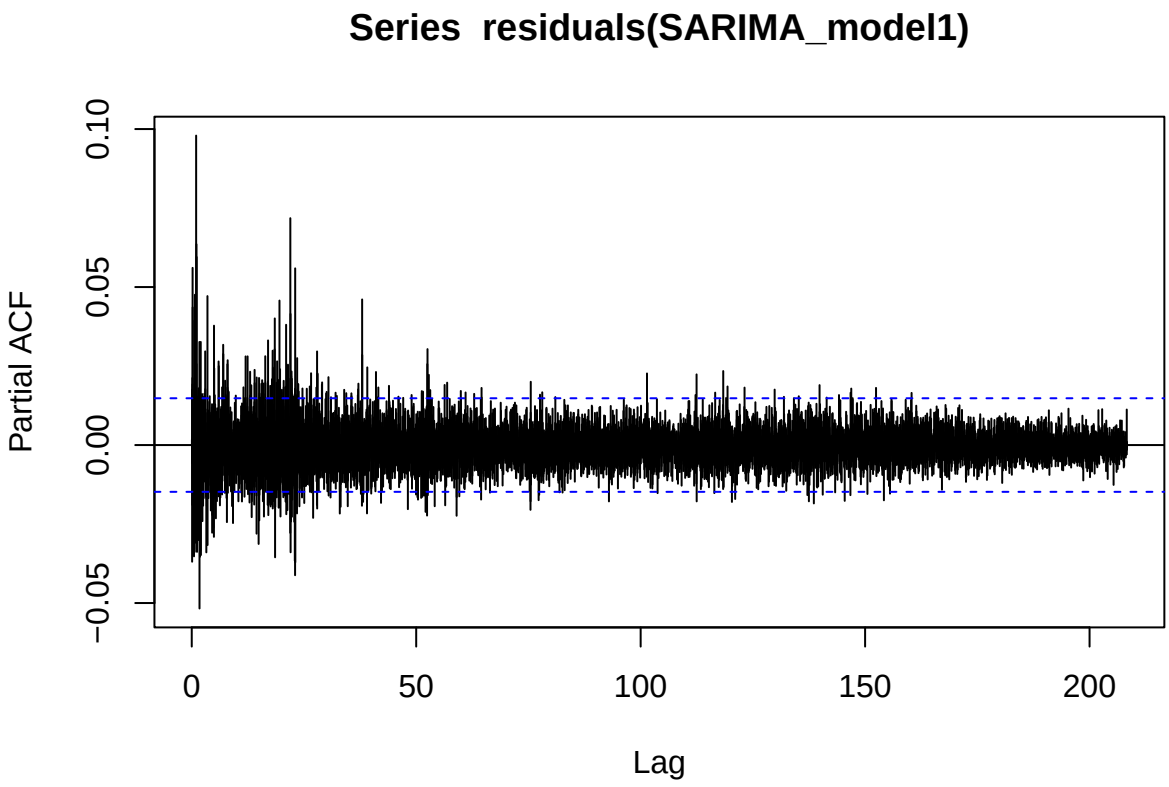
This tells me the model is unbiased, with residuals (errors) centered around 0. It also tells me that the models forecasts aren't systematically over or under predicting.

```r
acf(residuals(SARIMA_model1), lag.max = 10000)
```



Most autocorrelation bars fall within the dashed blue confidence bands, suggesting they're not statistically significant. However there are some lags outside of the confidence bounds at around lag 25, this suggests the MA term is under specified.

```
pacf(residuals(SARIMA_model1), lag.max = 10000)
```



There are a few spikes outside of the confidence bounds, particularly between lag 1 and lag 25 suggesting there is still significant autocorrelation in the residuals. This implies that the AR term (p) is under specified. I am going to try increasing the AR component to capture the remaining autocorrelation.

Checking that the time series is invertible (eg. if its errors can be represented as a weighted sum of past observations)

```
SARIMA_model1$coef
```

```
##         ar1         ar2         ar3         ma1        sma1
##  0.66836466  0.14446019 -0.03123472 -0.97687565  0.12701331
```

The model has MA terms ma1, and sma1.

I'm going to check the roots of the MA polynomial (All roots should be outside the unit circle (modulus > 1) for the model to be invertible).

```
ma_coefs <- c(1, -SARIMA_model1$coef["ma1"])  # Inverting sign for root checking
Mod(polyroot(ma_coefs))  # Gives modulus of each root (polyroot gives the complex roots)
```

```
## [1] 1.023672
```

I'm checking the seasonal MA separately from the non seasonal MA because the invertibility condition must hold for both polynomials (non seasonal and seasonal) independently.

```
sma_coefs <- c(1, -SARIMA_model1$coef["sma1"])
Mod(polyroot(sma_coefs))
```

```
## [1] 7.87319
```

For invertibility, the modulus (absolute value) of all roots must be larger than 1.

All of these are well outside the unit circle (i.e. modulus > 1), which means the ARIMA model satisfies the invertibility condition for both the non-seasonal and seasonal MA components.

STL-ARIMA

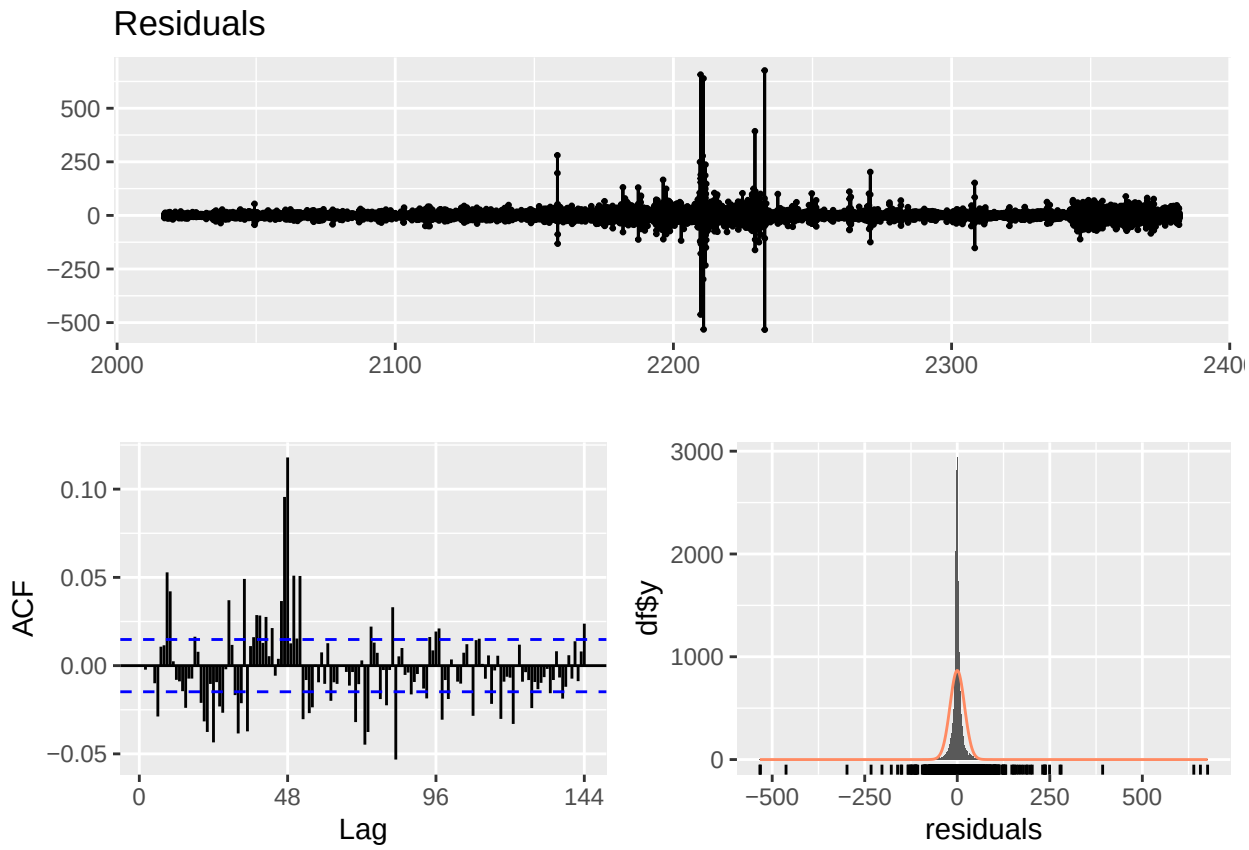
Ljung-Box test:

Null hypothesis: There is no autocorrelation in the residuals.

Alternative hypothesis: There is autocorrelation in the residuals.

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

```
checkresiduals(STL_ARIMA_model1)
```




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

```
## data:  Residuals
## Q* = 1221.5, 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 and conclude that there is autocorrelation in the residuals. This means that the STL-ARIMA 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 = 1634.9, 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.

ARCH test on residuals:

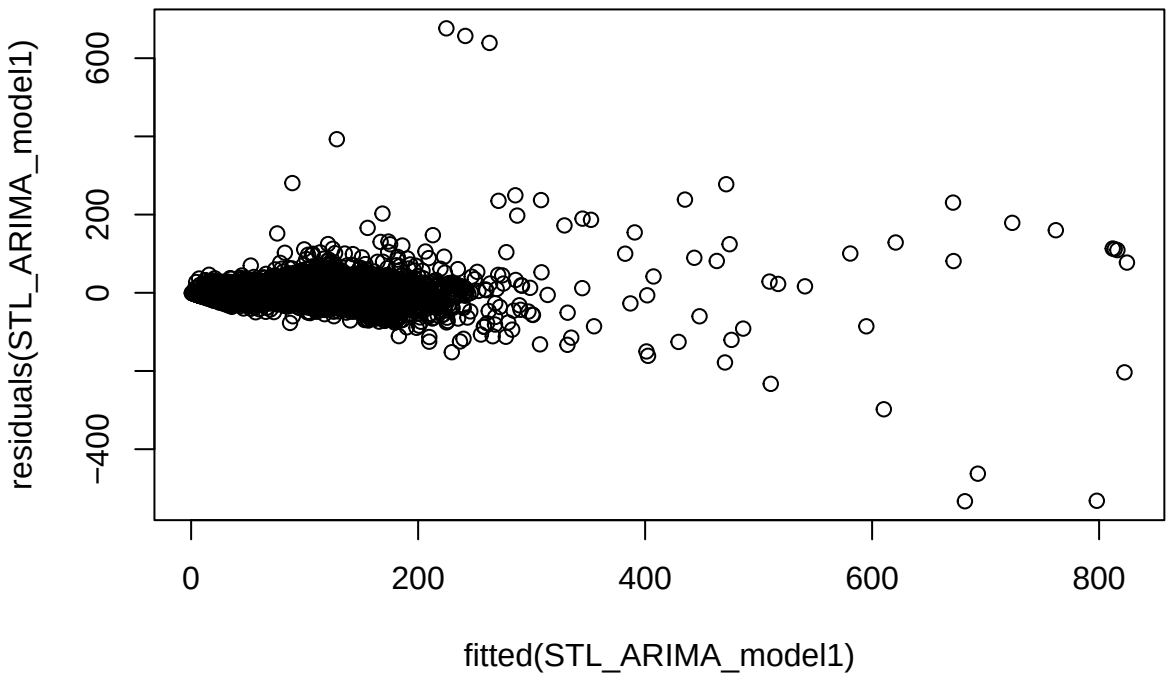
```
ArchTest(residuals(STL_ARIMA_model1), lags = 48)
```

```
##
##  ARCH LM-test; Null hypothesis: no ARCH effects
##
## data:  residuals(STL_ARIMA_model1)
## Chi-squared = 3165.9, df = 48, p-value < 2.2e-16
```

The p-value is much smaller than 0.05 which means I reject H0 (that there is no ARCH effect (residuals are homoskedastic)) and conclude that there is evidence of heteroskedasticity.

Checking for constant variance and mean zero in plot:

```
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. This plot supports the results of the ARCH test.

Checking mean is reasonably close to zero with a one sample t-test:

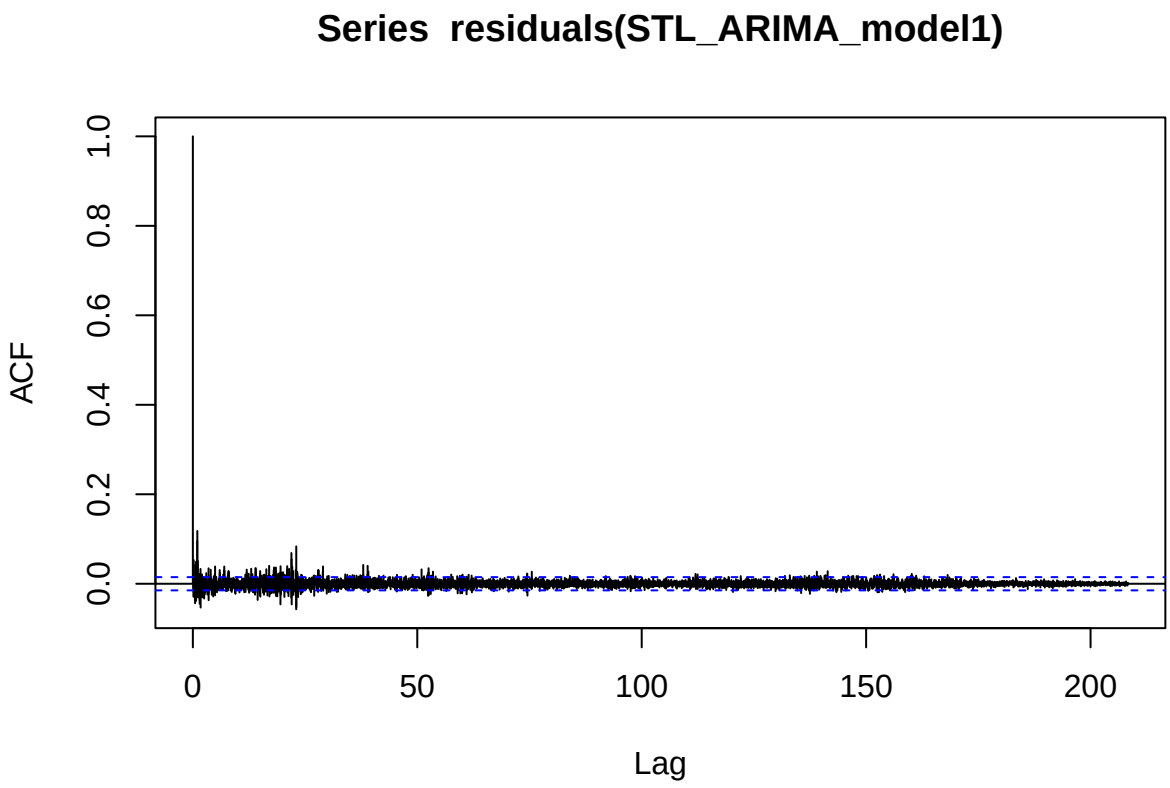
```
t.test(residuals(STL_ARIMA_model1), mu = 0)
```

```
##
##  One Sample t-test
##
## data:  residuals(STL_ARIMA_model1)
## t = 0.17712, df = 17519, p-value = 0.8594
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.2624609  0.3146053
## sample estimates:
##  mean of x
## 0.02607221
```

The mean of the residuals is 0.026

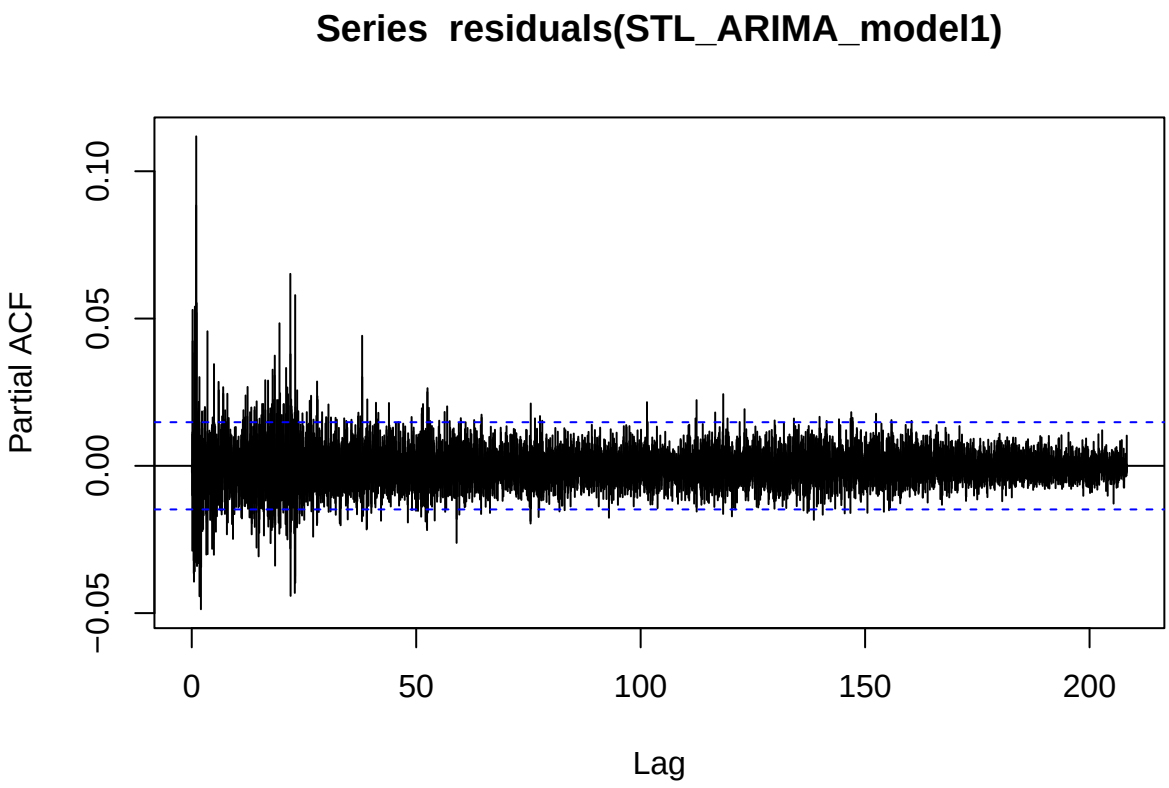
The p-value of the one sample t-test is 0.8594, there is not enough evidence to reject the null hypothesis, I conclude that the mean of the residuals is equal to zero.

```
acf(residuals(STL_ARIMA_model1), lag.max = 10000)
```



There is a significant spike at around lags 1 and 25, this suggests that the MA term is under specified.

```
pacf(residuals(STL_ARIMA_model1), lag.max = 10000)
```



There are a significant spikes outside of the confidence bounds, particularly between lag 1 and lag 25 suggesting there is still significant autocorrelation in the residuals. This implies that the AR term (p) is under specified. I am going to try increasing the AR component to capture the remaining autocorrelation.

Checking that the time series is invertible (eg. if its errors can be represented as a weighted sum of past observations)

```
arima_part <- STL_ARIMA_model1$model
arima_part$coef
```

```
##         ar1         ar2         ar3         ar4         ar5         ma1
##  0.67547312  0.14643818 -0.04283513 -0.03254515  0.04974068 -0.97720058
```

The model has one MA term, ma1.

I'm going to check the roots of the MA polynomial (All roots should be outside the unit circle (modulus > 1) for the model to be invertible).

```
ma_coefs <- c(1, -arima_part$coef["ma1"])  # Inverting sign for root checking
Mod(polyroot(ma_coefs))  # Gives modulus of each root (polyroot gives the complex roots)
```

```
## [1] 1.023331
```

For invertibility, the modulus (absolute value) of all roots must be larger than 1.

Since the absolute value of the root is outside the unit circle (i.e. modulus > 1), this means the ARIMA part of the STL-ARIMA model satisfies the invertibility condition for both the MA components.

SARIMAX

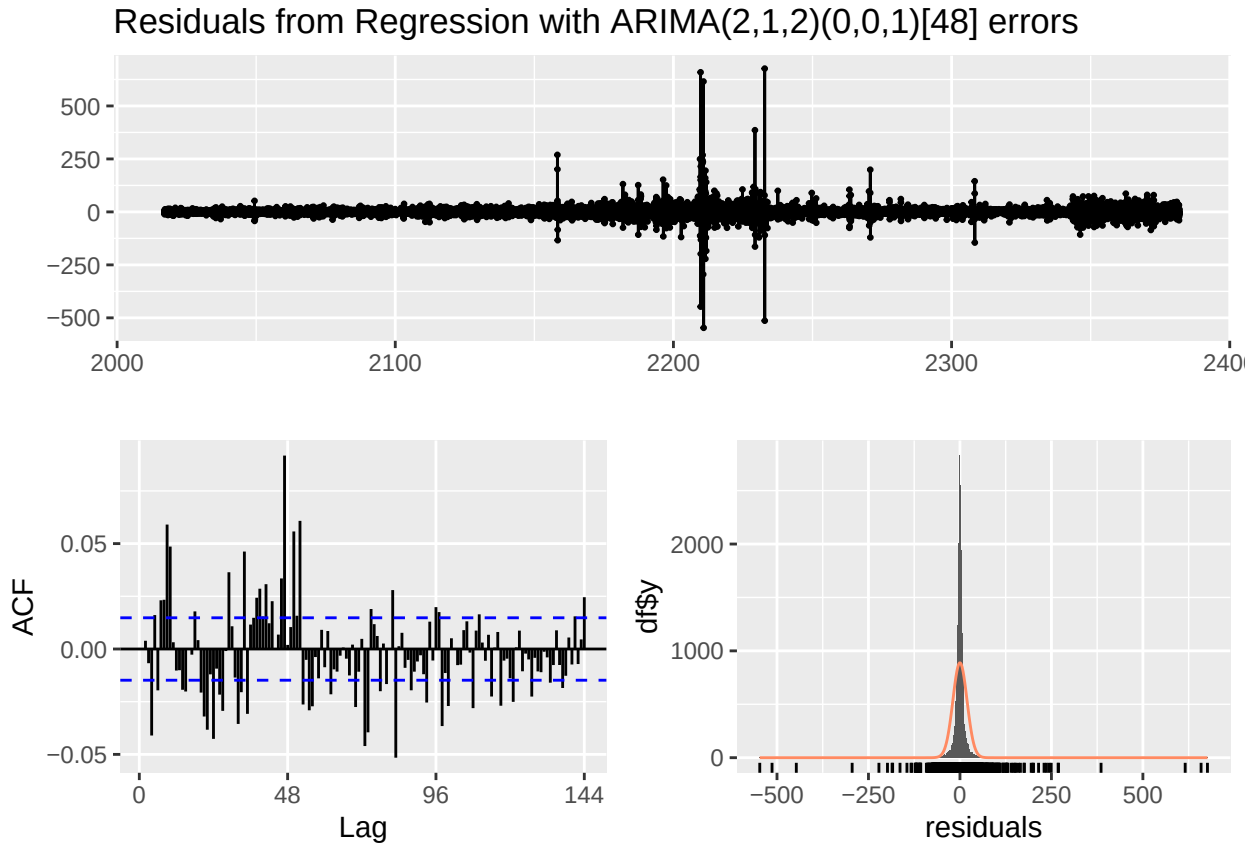
Ljung-Box test:

Null hypothesis: There is no autocorrelation in the residuals.

Alternative hypothesis: There is autocorrelation in the residuals.

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

```
checkresiduals(SARIMAX_model1)
```

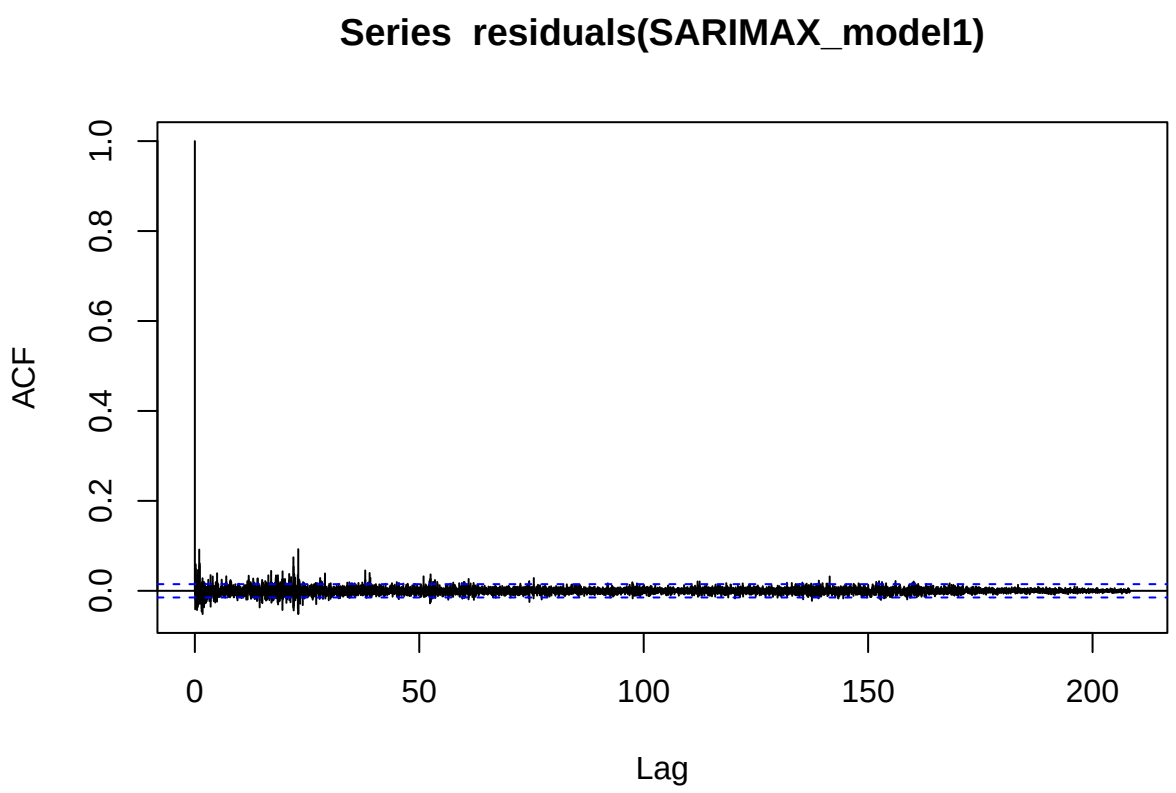



```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(2,1,2)(0,0,1)[48] errors
## Q* = 1030.2, df = 91, p-value < 2.2e-16
##
## Model df: 5.   Total lags used: 96
```

The p-value is much smaller than the significance level of 0.05, I reject the null hypothesis and conclude that there is autocorrelation in the residuals.

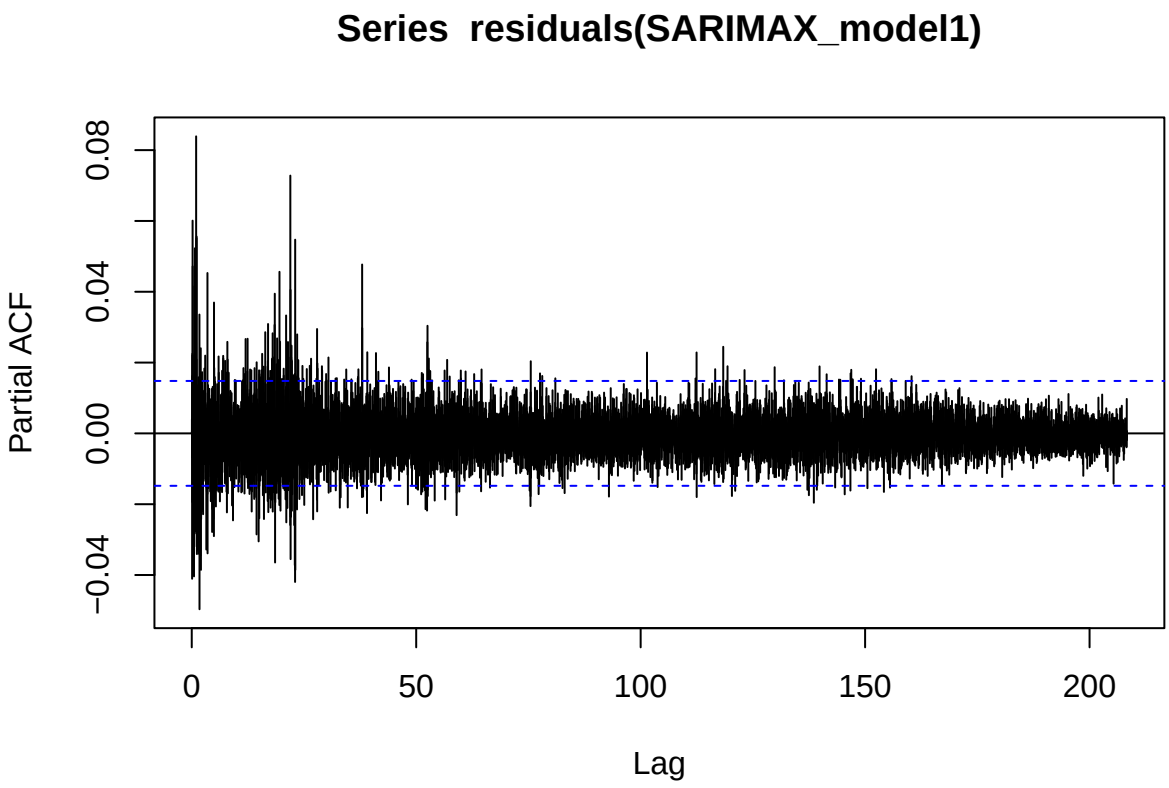
The ACF plot shows lags outside the confidence bounds, particularly at lag 48 and around lag 96

```r
acf(residuals(SARIMAX_model1), lag.max = 10000)
```



Most lags are within the confidence bounds but there is a spike around lag 25 that is outside the confidence bounds. I'm going to increase the MA term because of this.

```r
pacf(residuals(SARIMAX_model1), lag.max = 10000)
```



There are a significant spikes outside of the confidence bounds, particularly between lag 1 and lag 25 suggesting there is still significant autocorrelation in the residuals. This implies that the AR term (p) is under specified. I am going to try increasing the AR component to capture the remaining autocorrelation.

Anderson-Darling normality test:

```
ad.test(residuals(SARIMAX_model1))
```

```
##
##  Anderson-Darling normality test
##
## data:  residuals(SARIMAX_model1)
## A = 1565.4, p-value < 2.2e-16
```

The p-value is much smaller than the significance level of 0.05, I reject the null hypothesis and conclude that the residuals deviate significantly from a normal distribution.

Since the residuals deviate significantly from normality, the SARIMAX model's forecast uncertainty estimates could be inaccurate (too narrow or too wide).

ARCH test on residuals:

```
ArchTest(residuals(SARIMAX_model1), lags = 48)
```

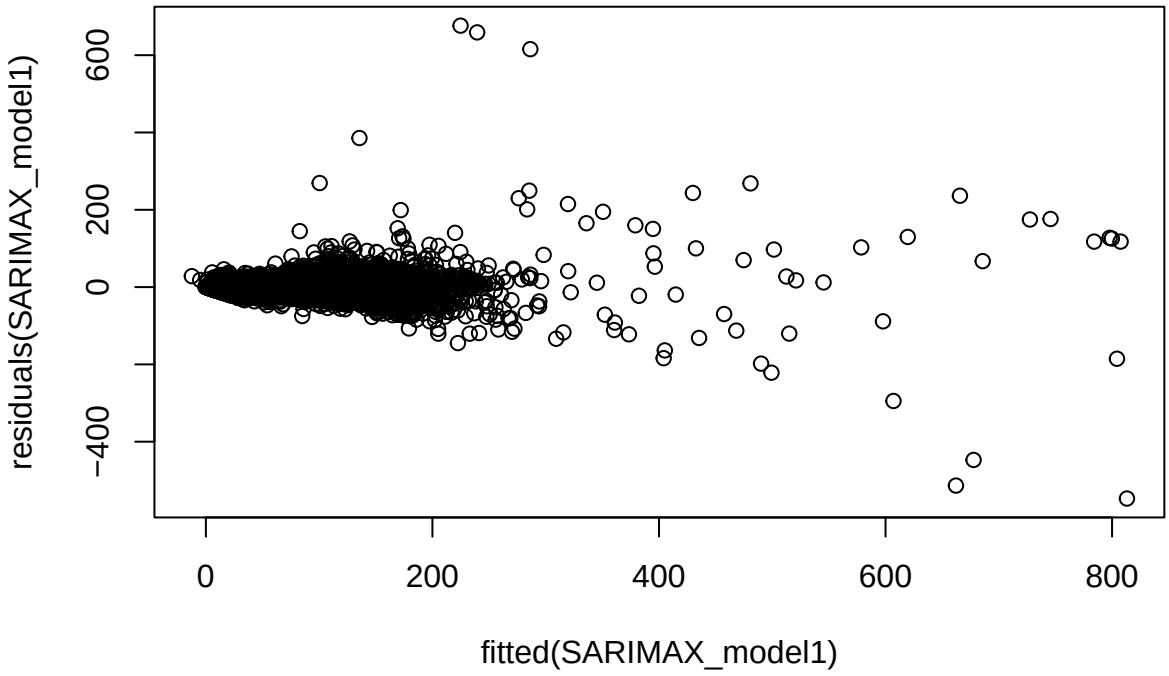
```
##
##  ARCH LM-test; Null hypothesis: no ARCH effects
```

```
##
## data:  residuals(SARIMAX_model1)
## Chi-squared = 3072.6, df = 48, p-value < 2.2e-16
```

The p-value is much smaller than the significance level of 0.05, I reject the null hypothesis and conclude that there are ARCH effects present in the residuals (the residual variance changes over time and depends on past squared residuals (heteroskedasticity). The assumption of constant variance for the SARIMAX model is violated, meaning that the models forecast confidence intervals may be inaccurate.

Checking for constant variance in plot:

```
plot(residuals(SARIMAX_model1) ~ fitted(SARIMAX_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.

This plot supports the results from the ARCH test.

Checking the mean of the residuals is reasonably close to zero with a one sample t-test:

```
t.test(residuals(SARIMAX_model1), mu = 0)
```

```
##
##  One Sample t-test
##
## data:  residuals(SARIMAX_model1)
## t = 0.15241, df = 17519, p-value = 0.8789
```

```
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.2625996  0.3068800
## sample estimates:
##  mean of x
## 0.02214019
```

The mean of the residuals is 0.02

The p-value of the one sample t-test is 0.8789, there is not enough evidence to reject the null hypothesis, I conclude that the mean of the residuals is equal to zero.

Checking that the time series is invertible (eg. if its errors can be represented as a weighted sum of past observations)

```
SARIMAX_model1$coef
```

```
##               ar1               ar2               ma1               ma2
##         0.4546581         0.2634779        -0.7851263        -0.1860038
##              sma1 Energy_generation      SeasonSpring      SeasonSummer
##         0.1151639        32.5625298        17.1364364         5.0376527
##      SeasonWinter        Is_Weekend
##         8.9367055         0.9156400
```

The model has MA terms ma1, ma2, and sma1.

```
ma_coefs <- c(1, -SARIMAX_model1$coef["ma1"], -SARIMAX_model1$coef["ma2"])  # Inverting sign for root c
Mod(polyroot(ma_coefs))  # Gives modulus of each root (polyroot gives the complex roots)
```

```
## [1] 2.318671 2.318671
```

```
sma_coefs <- c(1, -SARIMAX_model1$coef["sma1"])
Mod(polyroot(sma_coefs))
```

```
## [1] 8.683273
```

For invertibility, the modulus (absolute value) of all roots must be larger than 1.

All of these are well outside the unit circle (i.e. modulus > 1), which means the SARIMAX model satisfies the invertibility condition for both the non-seasonal and seasonal MA components.

Assessing variable significance using a t-test for regression coefficients.

```
summary(SARIMAX_model1)
```

```
## Series: time_series_data
## Regression with ARIMA(2,1,2)(0,0,1)[48] errors
##
## Coefficients:
##          ar1     ar2      ma1      ma2    sma1  Energy_generation  SeasonSpring
```

```
##       0.4547  0.2635  -0.7851  -0.1860  0.1152            32.5625       17.1364
## s.e.  0.0470  0.0335   0.0481   0.0463  0.0075             1.4108       13.8609
##       SeasonSummer  SeasonWinter  Is_Weekend
##             5.0377        8.9367      0.9156
## s.e.       12.0068       12.0052      1.4094
##
## sigma^2 = 369.9:  log likelihood = -76651.97
## AIC=153326   AICc=153326   BIC=153411.4
##
## Training set error measures:
##                       ME    RMSE      MAE       MPE    MAPE      MASE
## Training set 0.02214019 19.2276 8.470801 -70.76959 80.0227 0.4343825
##                       ACF1
## Training set 0.0003994259
```

Calculating t-values (coefficient / standard error) and approximate p-values:

```
t_values <- data.frame(variables = c("Energy_generation", "SeasonSpring", "SeasonSummer", "SeasonWinter
                       coefficient = c(32.5625, 17.1364, 5.0377, 8.9367, 0.9156),
                       standard_error = c(1.4108, 13.8609, 12.0068, 12.0052, 1.4094)
                       )
t_values$t_values <- t_values$coefficient / t_values$standard_error

n = 17520 # observations
k = 10 # estimated parameters: 2 AR terms + 2 MA terms + 1 seasonal MA term + 5 regressors

t_values$p_values <- 2 * (1 - pt(abs(t_values$t_values), df = n - k))
t_values
```

```
##           variables coefficient standard_error   t_values  p_values
## 1 Energy_generation     32.5625         1.4108 23.0808761 0.0000000
## 2      SeasonSpring     17.1364        13.8609  1.2363122 0.2163591
## 3      SeasonSummer      5.0377        12.0068  0.4195706 0.6748043
## 4      SeasonWinter      8.9367        12.0052  0.7444024 0.4566430
## 5        Is_Weekend      0.9156         1.4094  0.6496381 0.5159345
```

The Energy_generation variable at the 5% significance level is statistically significant as its p-value is much smaller than 0.05

This tells me that the Energy_generation variable is a strong predictor in my model.

The variables SeasonSpring, SeasonSummer, SeasonWinter, and Is_Weekend are not statistically significant as their p-values are larger than the significance level of 0.05.

SARIMA-fiGARCH

Ljung-Box test:

Null hypothesis: There is no autocorrelation in the residuals.

Alternative hypothesis: There is autocorrelation in the residuals.

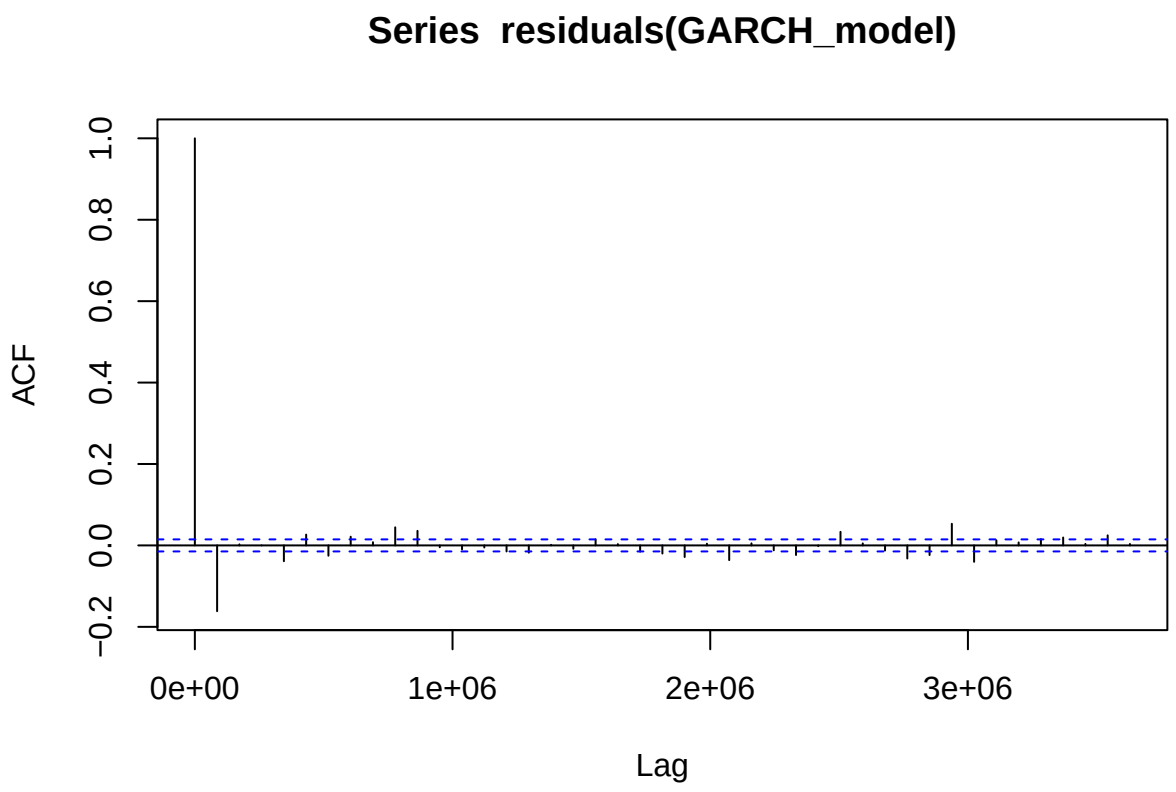
Autocorrelation check with Ljung Box test, acf and pacf plots:

```
Box.test(residuals(GARCH_model), lag = 20, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  residuals(GARCH_model)
## X-squared = 595.46, df = 20, p-value < 2.2e-16
```

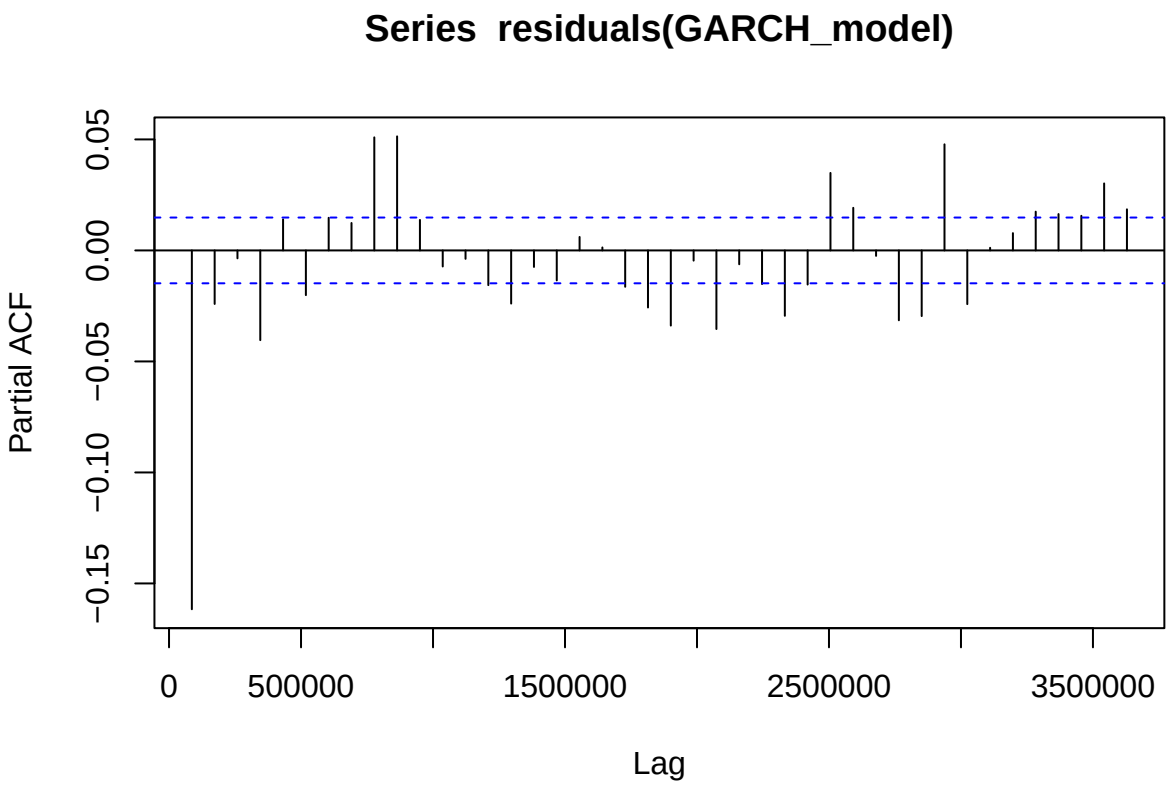
The p-value is much smaller than the significance level of 0.05, I reject the null hypothesis and conclude that there is autocorrelation in the residuals.

```
acf(residuals(GARCH_model))
```



There are a lot of spikes outside the confidence bounds between lags 0 and 100,000 particularly, this gradually tapers off as the lags increase. Due to this I will be increasing the MA terms.

```
pacf(residuals(GARCH_model))
```



In the partial autocorrelation plot, there are many spikes that are outside the confidence bounds, particularly at around lag 1, lag 400,000 and lag 800,000. This then gradually tapers off. This means that there is still some autocorrelation in the residuals, this is supported by the ACF1 value of 0.91 from the previous evaluation of the model on the validation set. Due to this I will be increasing the AR term.

Anderson-Darling normality test:

Null hypothesis: The residuals follow a normal distribution.

Alternative hypothesis: The residuals deviate significantly from a normal distribution.

```
ad.test(as.numeric(residuals(GARCH_model)))
```

```
##
##  Anderson-Darling normality test
##
## data:  as.numeric(residuals(GARCH_model))
## A = 1742, p-value < 2.2e-16
```

The p-value is much smaller than the significance level of 0.05, I reject the null hypothesis and conclude that the residuals deviate significantly from a normal distribution.

ARCH test on residuals:

Null Hypothesis: No ARCH effects in the residuals (the variance is constant over time (homoscedastic)).

Alternative Hypothesis: ARCH effects are present (the residual variance changes over time and depends on past squared residuals (heteroskedasticity)).

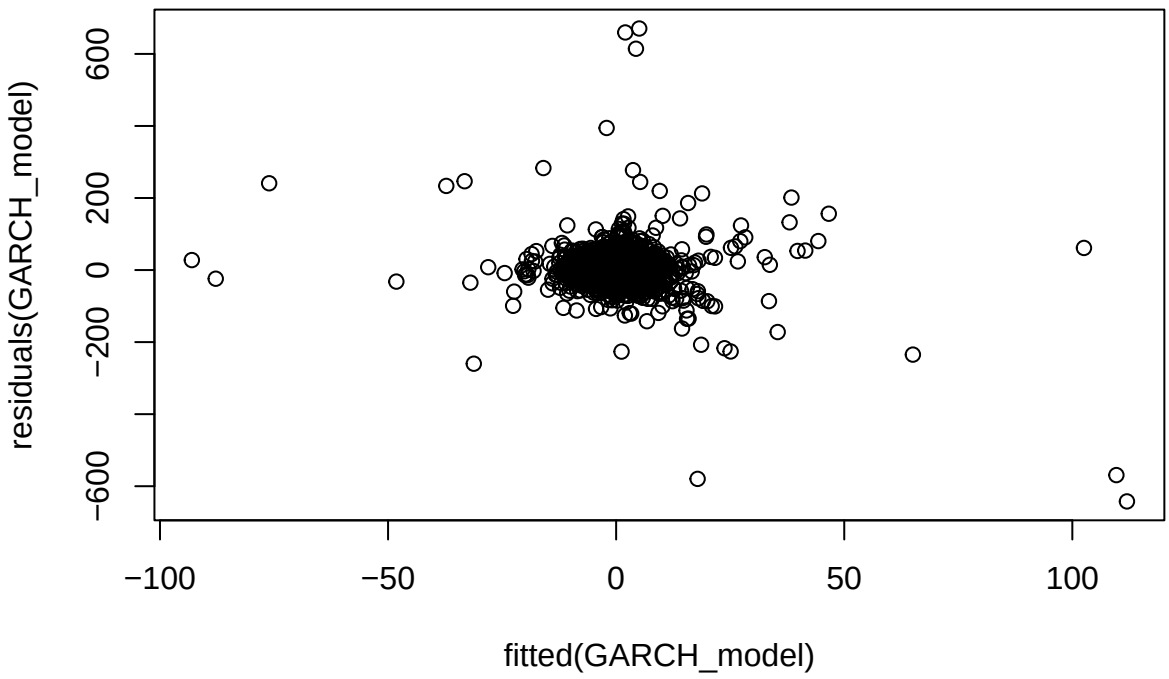
```
ArchTest(residuals(GARCH_model), lags = 48)
```

```
##
##  ARCH LM-test; Null hypothesis: no ARCH effects
##
## data:  residuals(GARCH_model)
## Chi-squared = 4315.8, df = 48, p-value < 2.2e-16
```

The p-value is much smaller than the significance level of 0.05, I reject the null hypothesis and conclude that there are ARCH effects present in the residuals (the residual variance changes over time and depends on past squared residuals (heteroskedasticity).

Checking for constant variance in plot:

```
plot(residuals(GARCH_model) ~ fitted(GARCH_model))
```



Checking the mean of the residuals is reasonably close to zero with a one sample t-test:

```
t.test(as.numeric(residuals(GARCH_model)), mu = 0)
```

```
##
##  One Sample t-test
##
## data:  as.numeric(residuals(GARCH_model))
## t = -0.25138, df = 17519, p-value = 0.8015
```

```
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.3304595  0.2553327
## sample estimates:
##   mean of x
## -0.03756341
```

The p-value is much larger than the significance level of 0.05 which means that I fail to reject the null hypothesis and conclude that the mean of the residuals is reasonably close to zero.

Checking that the time series is invertible. I've already checked that if SARIMA model is invertible, so I am only going to check if the fiGARCH part of the model is invertible. To do this I am checking if d the fractional differencing parameter is bigger than 0 and smaller than 1

```
coef(GARCH_model)
```

```
##         mu        ar1      omega     alpha1      beta1      delta       skew
## 0.06972062 0.16566284 7.37571965 0.13630621 0.38050833 0.60582936 1.06959282
##      shape
## 3.08821239
```

d = 0.6 which means that the fiGARCH part is invertible.

TBATS

```
summary(TBATS_model1)
```

```
##                   Length Class  Mode
## lambda                 1 -none- numeric
## alpha                  1 -none- numeric
## beta                   1 -none- numeric
## damping.parameter      1 -none- numeric
## gamma.one.values       1 -none- numeric
## gamma.two.values       1 -none- numeric
## ar.coefficients        0 -none- NULL
## ma.coefficients        0 -none- NULL
## likelihood             1 -none- numeric
## optim.return.code      1 -none- numeric
## variance               1 -none- numeric
## AIC                    1 -none- numeric
## parameters             2 -none- list
## seed.states           18 -none- numeric
## fitted.values      17520 ts     numeric
## errors             17520 ts     numeric
## x                 315360 -none- numeric
## seasonal.periods       1 -none- numeric
## k.vector               1 -none- numeric
## y                  17520 ts     numeric
## p                      1 -none- numeric
## q                      1 -none- numeric
## call                   2 -none- call
## series                 1 -none- character
## method                 1 -none- character
```

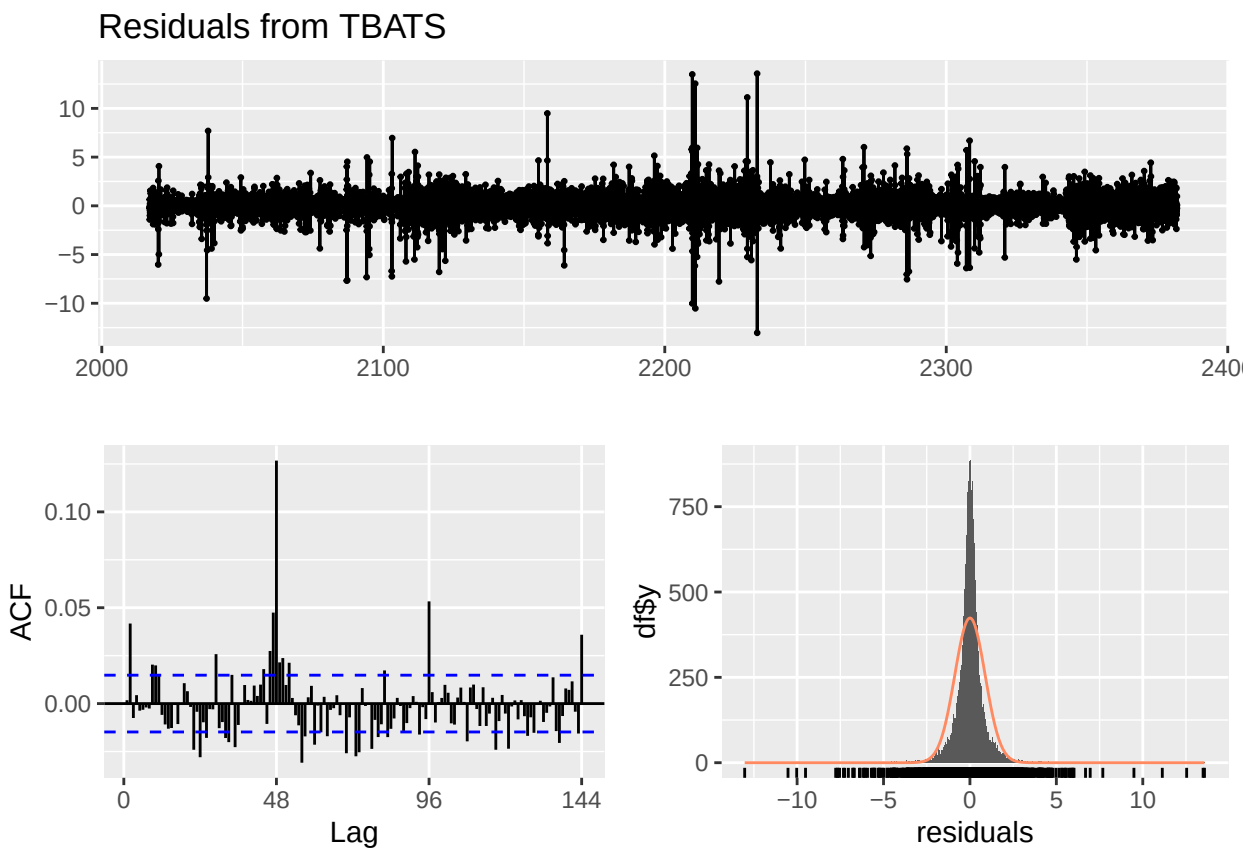
Ljung-Box test:

Null hypothesis: There is no autocorrelation in the residuals.

Alternative hypothesis: There is autocorrelation in the residuals.

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

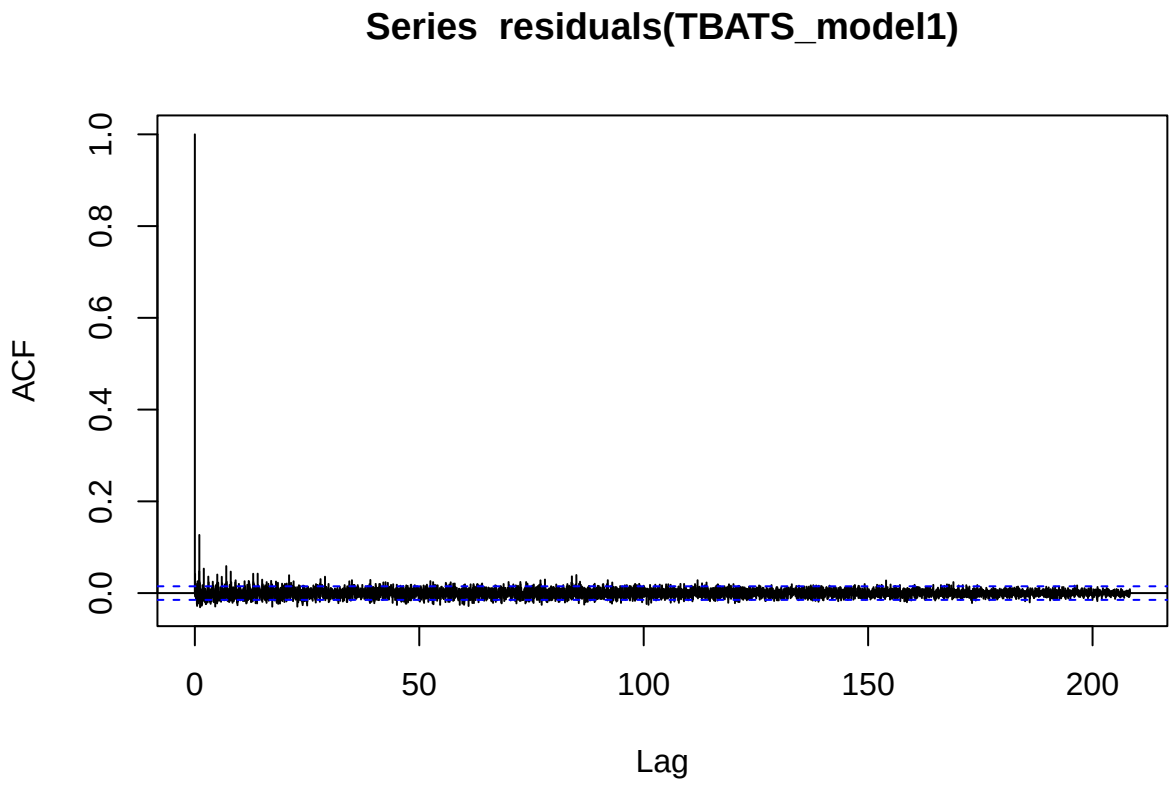
```
checkresiduals(TBATS_model1)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from TBATS
## Q* = 694.52, df = 96, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 96
```

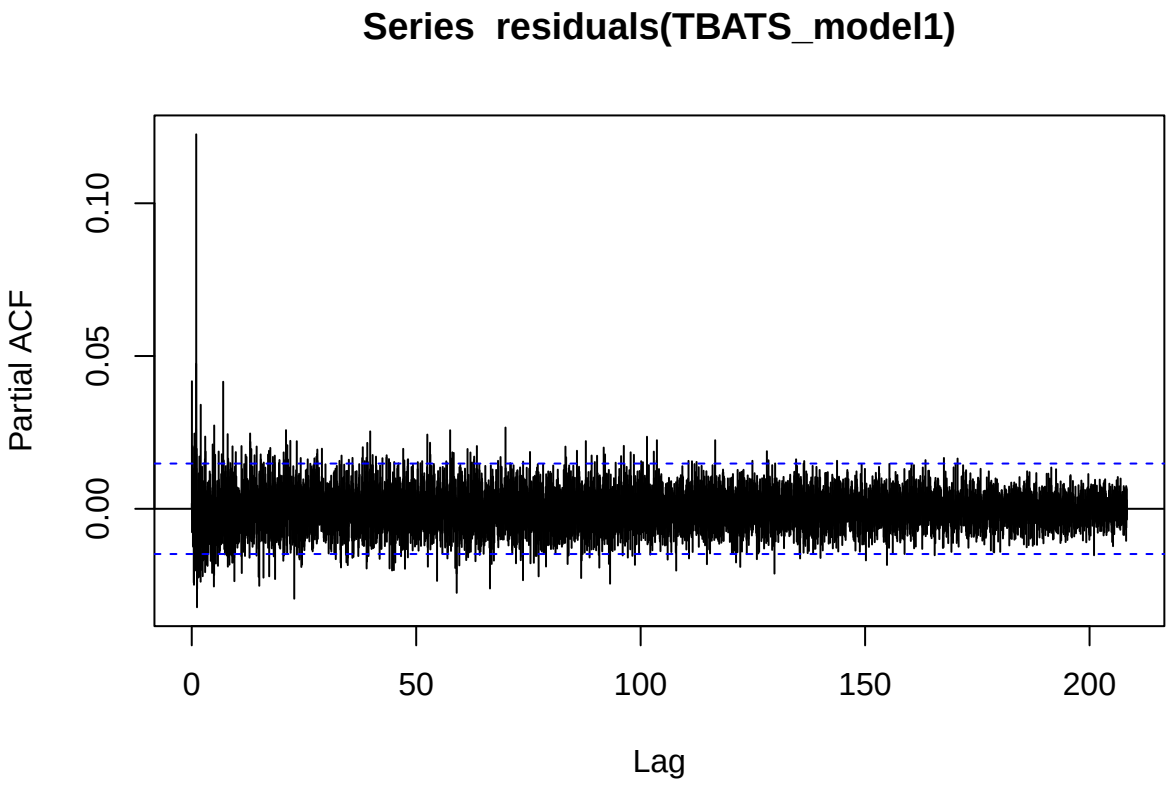
The p-value is much smaller than the significance level of 0.05, I reject the null hypothesis and conclude that there is autocorrelation in the residuals.

```
acf(residuals(TBATS_model1), lag.max = 10000)
```



There are a lot of spikes outside the confidence bounds between lags 0 and 25 particularly, this gradually tapers off as the lags increase.

```
pacf(residuals(TBATS_model1), lag.max = 10000)
```



In the partial autocorrelation plot, there are many spikes that are outside the confidence bounds, particularly at around lag 1, and lag 20. This then gradually tapers off. This means that there is still some autocorrelation in the residuals, this is supported by the ACF1 value of 0.70 from the previous evaluation of the model on the validation set. Unfortunately, even though there is clearly autocorrelation in the residuals, I can't manually adjust the AR and MA terms in my TBATS model. To resolve this I am going to consider hybrid models.

Anderson-Darling normality test:

Null hypothesis: The residuals follow a normal distribution.

Alternative hypothesis: The residuals deviate significantly from a normal distribution.

```
ad.test(residuals(TBATS_model1))
```

```
##
##  Anderson-Darling normality test
##
## data:  residuals(TBATS_model1)
## A = 578.78, p-value < 2.2e-16
```

The p-value is much smaller than the significance level of 0.05, I reject the null hypothesis and conclude that the residuals deviate significantly from a normal distribution.

ARCH test on residuals:

Null Hypothesis: No ARCH effects in the residuals (the variance is constant over time (homoscedastic)).

Alternative Hypothesis: ARCH effects are present (the residual variance changes over time and depends on past squared residuals (heteroskedasticity)).

```
ArchTest(residuals(TBATS_model1), lags = 48)
```

```
##
##  ARCH LM-test; Null hypothesis: no ARCH effects
##
## data:  residuals(TBATS_model1)
## Chi-squared = 2063.3, df = 48, p-value < 2.2e-16
```

The p-value is much smaller than the significance level of 0.05, I reject the null hypothesis and conclude that there are ARCH effects present in the residuals (the residual variance changes over time and depends on past squared residuals (heteroskedasticity).

Checking the mean of the residuals is reasonably close to zero with a one sample t-test:

```
t.test(residuals(TBATS_model1), mu = 0)
```

```
##
##  One Sample t-test
##
## data:  residuals(TBATS_model1)
## t = 0.23576, df = 17519, p-value = 0.8136
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.01142870  0.01455384
## sample estimates:
##   mean of x
## 0.001562573
```

The p-value is much larger than the significance level of 0.05 which means that I fail to reject the null hypothesis and conclude that the mean of the residuals is reasonably close to zero.

Model refinement

For every single model the residuals showed autocorrelation, non-normality and had ARCH effects (non constant variance).

Based on this I've decided to adjust the models in the following ways:

I am going to apply a Box-Cox transformation to my time series prior to fitting the models to stabilize the variance.

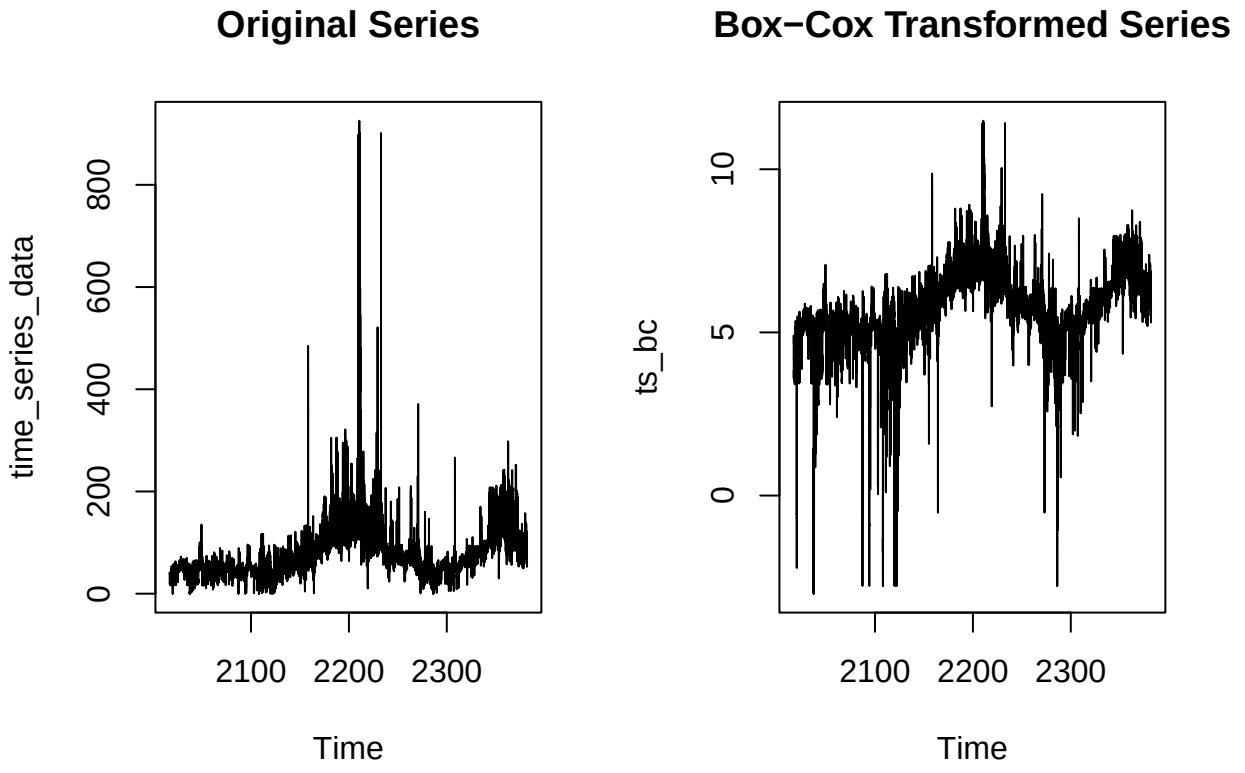
I am going to increase the orders of the AR and MA components because the Ljung box test's show that there is still autocorrelation in the residuals of every single model, which means that every single model is not capturing all the underlying structure/patterns in the data. The ACF and PACF plots support this.

I will be adding additional variables to SARIMAX to see if that improves the models forecasting accuracy.

ARIMA

Applying Box-Cox to time series:

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

par(mfrow = c(1, 2))
plot(time_series_data, main = "Original Series")
plot(ts_bc, main = "Box-Cox Transformed Series")
```



Box-Cox time series model with no change in order of AR and MA components:

```
ARIMA_model2 <- Arima(ts_bc, order = c(3, 1, 1))
```

Box-Cox time series model with increased order of AR component:

```
ARIMA_model3 <- Arima(ts_bc, order = c(4, 1, 1))
```

Box-Cox time series model with increased order of MA component:

```
ARIMA_model4 <- Arima(ts_bc, order = c(3, 1, 2))
```

Box-Cox time series model with increased order of AR and MA components:

```
ARIMA_model5 <- Arima(ts_bc, order = c(4, 1, 2))
```

Time series model (no transformation) with increased AR component:

```
ARIMA_model6 <- Arima(time_series_data, order = c(4,1,1))
```

Time series model (no transformation) with increased MA component:

```
ARIMA_model7 <- Arima(time_series_data, order = c(3,1,2))
```

Time series model (no transformation) with increased AR and MA components:

```
ARIMA_model8 <- Arima(time_series_data, order = c(4,1,2))
```

SARIMA

Box-Cox time series model:

```
SARIMA_model2 <- Arima(ts_bc, order = c(3,1,1),
                       seasonal = list(order = c(0, 0, 1), period = 48))
```

Box-Cox time series model with increased AR:

```
SARIMA_model3 <- Arima(ts_bc, order = c(4,1,1),
                       seasonal = list(order = c(0, 0, 1), period = 48))
```

Box-Cox time series model with increased MA:

```
SARIMA_model4 <- Arima(ts_bc, order = c(3,1,2),
                       seasonal = list(order = c(0, 0, 1), period = 48))
```

Box-Cox time series model with increased AR and MA:

```
SARIMA_model5 <- Arima(ts_bc, order = c(4,1,2),
                       seasonal = list(order = c(0, 0, 1), period = 48))
```

Box-Cox time series model with increased seasonal MA:

```
SARIMA_model6 <- Arima(ts_bc, order = c(3,1,1),
                       seasonal = list(order = c(0, 0, 2), period = 48))
```

Time series without transformation, and increased AR:

```
SARIMA_model7 <- Arima(ts_bc, order = c(4,1,1),
                       seasonal = list(order = c(0, 0, 1), period = 48))
```

Time series without transformation, and increased MA:

```
SARIMA_model8 <- Arima(ts_bc, order = c(3,1,2),
                       seasonal = list(order = c(0, 0, 1), period = 48))
```

Time series without transformation, and increased AR and MA terms:

```
SARIMA_model9 <- Arima(ts_bc, order = c(4,1,2),
                       seasonal = list(order = c(0, 0, 1), period = 48))
```

Time series without transformation, and increased seasonal MA terms:

```
SARIMA_model10 <- Arima(ts_bc, order = c(3,1,1),
                       seasonal = list(order = c(0, 0, 2), period = 48))
```

STL-ARIMA

```
STL_ARIMA_model2 <- stlm(ts_bc, s.window = "periodic", robust = TRUE,
                         modelfunction = function(x) Arima(x, order = c(5, 1, 1)))
```

```
STL_ARIMA_model3 <- stlm(ts_bc, s.window = "periodic", robust = TRUE,
                         modelfunction = function(x) Arima(x, order = c(6, 1, 1)))
```

```
STL_ARIMA_model4 <- stlm(ts_bc, s.window = "periodic", robust = TRUE,
                         modelfunction = function(x) Arima(x, order = c(5, 1, 2)))
```

```
STL_ARIMA_model5 <- stlm(ts_bc, s.window = "periodic", robust = TRUE,
                         modelfunction = function(x) Arima(x, order = c(6, 1, 2)))
```

```
STL_ARIMA_model6 <- stlm(time_series_data, s.window = "periodic", robust = TRUE,
                         modelfunction = function(x) Arima(x, order = c(6, 1, 1)))
```

```
STL_ARIMA_model7 <- stlm(time_series_data, s.window = "periodic", robust = TRUE,
                         modelfunction = function(x) Arima(x, order = c(5, 1, 2)))
```

```
STL_ARIMA_model8 <- stlm(time_series_data, s.window = "periodic", robust = TRUE,
                         modelfunction = function(x) Arima(x, order = c(6, 1, 2)))
```

SARIMAX

SARIMAX with the original xreg (season, energy generation and Is_weekend):

```
SARIMAX_model2 <- Arima(ts_bc, order = c(2,1,2), seasonal = list(order = c(0, 0, 1), period = 48),
                        xreg = xreg)
```

```
SARIMAX_model3 <- Arima(ts_bc, order = c(3,1,2), seasonal = list(order = c(0, 0, 1), period = 48),
                        xreg = xreg)
```

```
SARIMAX_model4 <- Arima(ts_bc, order = c(2,1,3), seasonal = list(order = c(0, 0, 1), period = 48),
                        xreg = xreg)
```

```
SARIMAX_model5 <- Arima(ts_bc, order = c(3,1,3), seasonal = list(order = c(0, 0, 1), period = 48),
                        xreg = xreg)
```

```
SARIMAX_model6 <- Arima(time_series_data, order = c(3,1,2), seasonal = list(order = c(0, 0, 1), period =
                      xreg = xreg)
```

```
SARIMAX_model7 <- Arima(time_series_data, order = c(2,1,3), seasonal = list(order = c(0, 0, 1), period =
                      xreg = xreg)
```

```
SARIMAX_model8 <- Arima(time_series_data, order = c(3,1,3), seasonal = list(order = c(0, 0, 1), period =
                      xreg = xreg)
```

Adding new variables to the xreg for SARIMAX

Bids I obtained this data from the Electricity Authority on their Wholesale datasets: Bids page.

Loading data:

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

Energy_Bids_2017 <- csv_files %>% lapply(read.csv) %>% bind_rows()
head(Energy_Bids_2017)
```

```
##   TradingDate TradingPeriod ParticipantCode PointOfConnection Unit
## 1  2017-01-01             1            CTCT           ASB0661
## 2  2017-01-01             1            CTCT           ASB0661
## 3  2017-01-01             1            CTCT           ASB0661
## 4  2017-01-01             1            CTCT           ASB0661
## 5  2017-01-01             1            CTCT           ASB0661
## 6  2017-01-01             1            CTCT           ASB0661
##   UTCSubmissionDate UTCSubmissionTime SubmissionOrder IsLatestYesNo
## 1        2016-12-28      21:14:45.000               1             N
## 2        2016-12-28      21:14:45.000               1             Y
## 3        2016-12-28      21:14:45.000               1             Y
## 4        2016-12-28      21:14:45.000               1             Y
## 5        2016-12-28      21:14:45.000               1             Y
## 6        2016-12-28      21:14:45.000               1             Y
##   DispatchableYesNo Tranche Megawatts DollarsPerMegawattHour
## 1                 1     1.756                   2000
## 2                 2     0.000                      0
## 3                 3     0.000                      0
## 4                 4     0.000                      0
## 5                 5     0.000                      0
## 6                 6     0.000                      0
```

```
Energy_Bids_2017$TradingDate <- as.Date(Energy_Bids_2017$TradingDate)
```

Summarizing the data and removing place holder bids (where no megawatts of energy were bid on)

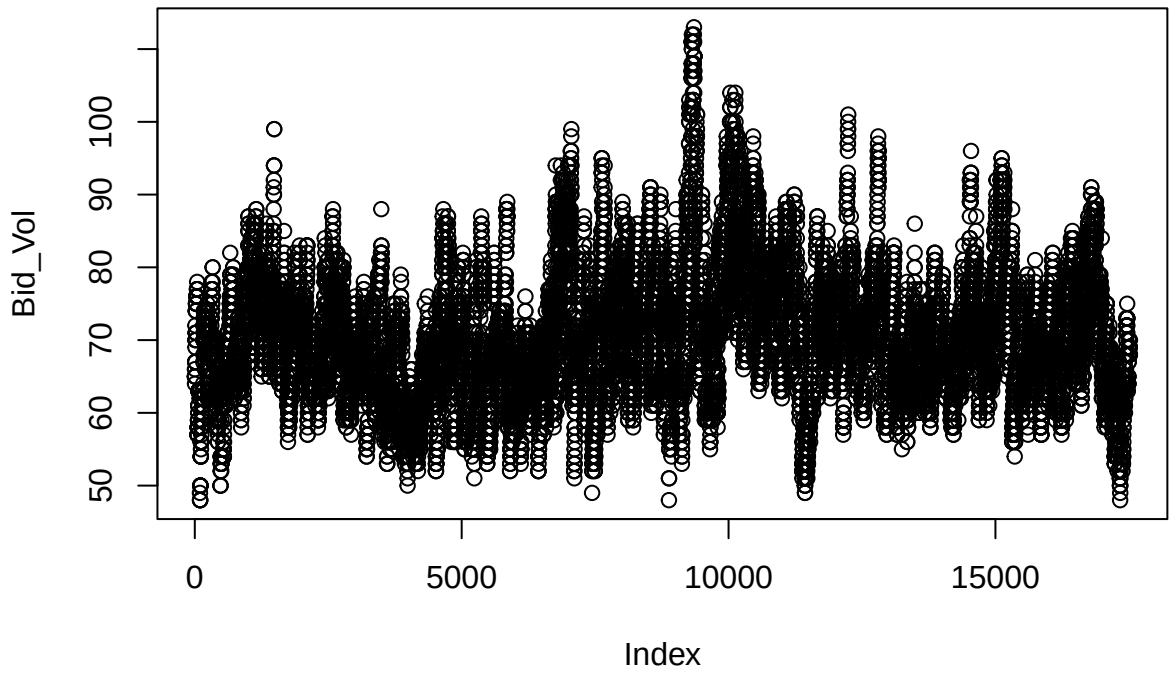
```
summary_counts <- Energy_Bids_2017 %>%
  filter(Megawatts > 0) %>% # Removing place holder bids (where no megawatts of energy were bid on)
  group_by(TradingDate, TradingPeriod) %>%
  summarise(Bid_Volume = n(), .groups = 'drop')
```

```r
print(summary_counts)
```

```
## # A tibble: 17,520 x 3
##    TradingDate TradingPeriod Bid_Volume
##    <date>              <int>      <int>
##  1 2017-01-01              1         65
##  2 2017-01-01              2         65
##  3 2017-01-01              3         66
##  4 2017-01-01              4         64
##  5 2017-01-01              5         65
##  6 2017-01-01              6         65
##  7 2017-01-01              7         67
##  8 2017-01-01              8         67
##  9 2017-01-01              9         67
## 10 2017-01-01             10         67
## # i 17,510 more rows
```

```r
Bid_Vol <- summary_counts$Bid_Volume
```

```r
plot(Bid_Vol)
```



Temperature I obtained this data from the Ministry for the Environment.

```r
temperature <- read.csv("Temperature/daily-temperature-1909-2019.csv")
head(temperature)
```

```
##   location_name       date statistic temperature station_name_niwa agent_number
## 1      Auckland 1966-01-01   Maximum        22.2     Auckland Aero         1962
## 2      Auckland 1966-01-01   Minimum        13.9     Auckland Aero         1962
## 3      Auckland 1966-01-01   Average        18.1     Auckland Aero         1962
## 4      Auckland 1966-01-02   Maximum        23.9     Auckland Aero         1962
## 5      Auckland 1966-01-02   Minimum        17.1     Auckland Aero         1962
## 6      Auckland 1966-01-02   Average        20.5     Auckland Aero         1962
```

I am going to restrict the dataset to temperature values in 2017.

```r
temperature$date <- as.Date(temperature$date)
temperature <- temperature[format(temperature$date, "%Y") == 2017, ]
```

```r
unique(temperature$station_name_niwa)
```

```
##  [1] "Auckland Aero"                "Blenheim Aero Aws"
##  [3] "Christchurch Aero"            "Dannevirke"
##  [5] "Dunedin, Musselburgh Ews"     "Gisborne Aws"
##  [7] "Gore Aws"                     "Hamilton Aws"
##  [9] "Hokitika Aero"                "Invercargill Aero"
## [11] "Kerikeri Ews"                 "Lake Tekapo, Air Safaris"
## [13] "Masterton, Te Ore Ore Cws"    "Milford Sound"
## [15] "Napier?Nelson?Pk"             "Nelson Aero"
## [17] "New Plymouth Aws"             "Queenstown Aero Aws"
## [19] "Reefton Ews"                  "Rotorua Aero Aws"
## [21] "Tara Hills Aws"               "Taumarunui"
## [23] "Taupo Aws"                    "Tauranga Aero Aws"
## [25] "Timaru Aero Aws"              "Waiouru?Treatment?Plant"
## [27] "Wellington, Kelburn Aws"      "Wanganui, Spriggens Park Ews"
## [29] "Whangaparaoa Aws"             "Whangarei Aero Aws"
```

This project is only using the ABY0111 point of connection which is in the Albury area. The closest automatic weather station to Albury is Timaru Aero Aws and this is therefor the closest temperature reading for the Albury area. I will be restricting the dataset to this AWS.

```r
temperature <- temperature[temperature$station_name_niwa == "Timaru Aero Aws", ]
head(temperature)
```

```
##         location_name       date statistic temperature station_name_niwa
## 1730158        Timaru 2017-01-01   Maximum        28.3   Timaru Aero Aws
## 1730159        Timaru 2017-01-01   Minimum        12.4   Timaru Aero Aws
## 1730160        Timaru 2017-01-01   Average        20.4   Timaru Aero Aws
## 1730161        Timaru 2017-01-02   Maximum        12.9   Timaru Aero Aws
## 1730162        Timaru 2017-01-02   Minimum        10.9   Timaru Aero Aws
## 1730163        Timaru 2017-01-02   Average        11.9   Timaru Aero Aws
##         agent_number
## 1730158         5086
## 1730159         5086
```

```
## 1730160          5086
## 1730161          5086
## 1730162          5086
## 1730163          5086
```

I am going to use this temperature data to make the HDD (heating degree days) and CDD (cooling degree days) variables to reflect winter heating demand and summer cooling demand. This should improve the models accuracy by accounting for temperature based energy demand. The base temperature will be 18 Celsius as that is the recommended base temperature for calculating HDD according to Consumer NZ (Consumer NZ, 31 Jan 2019).

Because I only have 3 observations per day (minimum, maximum and average) then I am going to interpolate the rest of temperature values using a two phase sine curve.

The minimum daily temperature is typically just before sunrise, sunrise is typically around 7am. About an hour before that at 6am will be the assumed minimum temperature time. For the curve the index for the minimum temperature will be 12.

The maximum daily temperature is typically in the afternoon when the sun is at its peak, I think this would be around 3 pm. For the curve the index for the maximum temperature will be 30.

This function will use the daily minimum, maximum and average temperatures to generate the temperatures for 48 half hour periods for a day.

```
generate_halfhour <- function(Tmin, Tmax, Tmean, tau_min = 12, tau_max = 30) {
  h <- 0:47

  # Rising from Tmin to Tmax
  rising <- (h >= tau_min & h <= tau_max)
  y <- numeric(48)
  y[rising] <- Tmin + (Tmax - Tmin) * sin(pi * (h[rising] - tau_min) / (tau_max - tau_min))
  # Falling from Tmax back to next day's Tmin
  falling <- !rising
  span2   <- 48 - (tau_max - tau_min)
  y[falling] <- Tmin + (Tmax - Tmin) * cos(pi * (h[falling] - tau_max) / span2)

  # Shift so that the daily mean matches Tmean
  y + (Tmean - mean(y))
}
```

Making min, max and average temperature into vectors:

```
tmean <- temperature[temperature$statistic == "Average", 4]
tmin <- temperature[temperature$statistic == "Minimum", 4]
tmax <- temperature[temperature$statistic == "Maximum", 4]
```

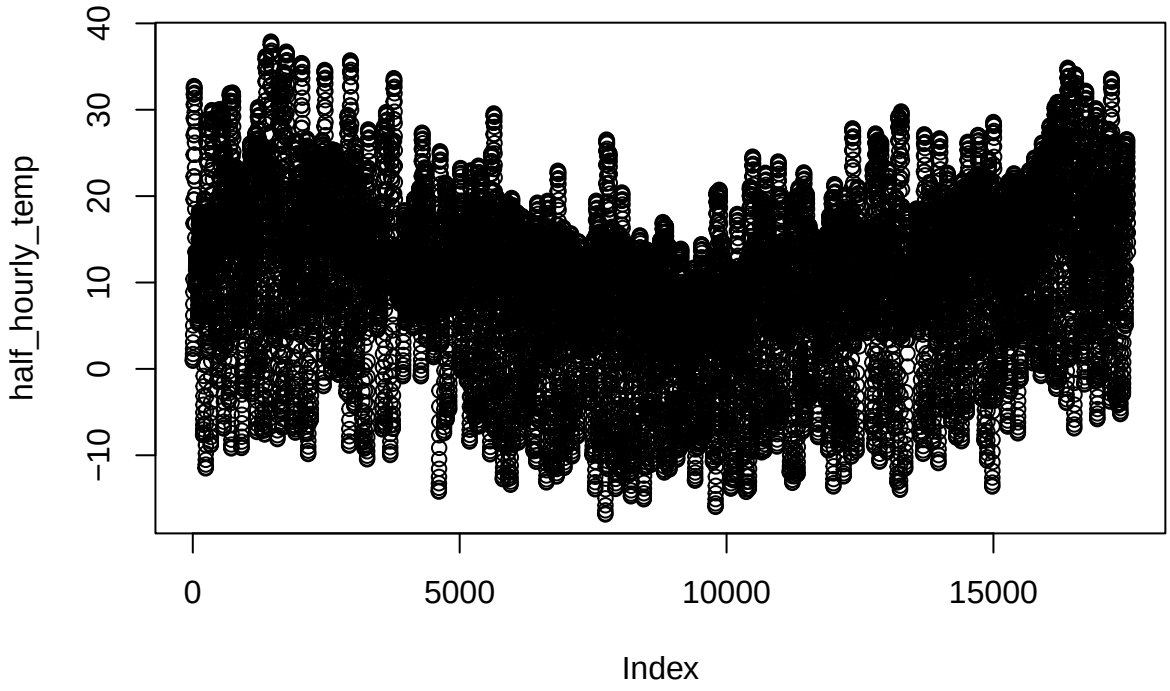
This function will generature 48 half hourly temeperatures for multiple days, given a min, max and average temperature.

```
# tmin, tmax, tmean are numeric vectors of length N_days
all_days <- mapply(
  generate_halfhour,
  Tmin  = tmin,
  Tmax  = tmax,
  Tmean = tmean,
```

```
  MoreArgs = list(tau_min = 12, tau_max = 30),
  SIMPLIFY = FALSE
)

# Binding the temperature values into a single long vector:
half_hourly_temp <- unlist(all_days)
length(half_hourly_temp)  # should be N_days * 48
```

```
## [1] 17520
```

```
plot(half_hourly_temp)
```



Getting CDD and HDD from temperature:

```
HDD <- pmax(0, 18 - half_hourly_temp)
CDD <- pmax(0, half_hourly_temp - 18)
```

Fuel prices Weekly fuel price monitoring data was obtained from the Ministry of Business, Innovation and Employment.

```
fuel_price <- read.csv("Temperature/weekly-table.csv")
head(fuel_price)
```

```
##      Week       Date   Fuel            Variable  Value    Unit Status
## 1 2004w17 2004-04-23   <NA>   Dubai crude price 32.100 USD/bbl  Final
## 2 2004w17 2004-04-23   <NA>   Dubai crude price 50.740 NZD/bbl  Final
## 3 2004w17 2004-04-23   <NA>       Exchange rate  0.633 USD/NZD  Final
## 4 2004w17 2004-04-23 Diesel       Importer cost 43.400 NZD c/L  Final
## 5 2004w17 2004-04-23 Diesel Price excluding tax 59.600 NZD c/L  Final
## 6 2004w17 2004-04-23 Diesel                ETS  0.000 NZD c/L  Final
```

I'm restricting the data to final diesel prices for 2017 in NZD. I will be using the ETS (emissions trading scheme) price as an exogenous variables for my SARIMAX.

ETS prices affect energy generation costs because electricity generators have to pay for carbon emissions. This will have a knock on effect on wholesale energy prices.

```
fuel_price$Date <- as.Date(fuel_price$Date)
fuel_price <- fuel_price[format(fuel_price$Date, "%Y") == 2017, ]
fuel_price <- fuel_price[fuel_price$Variable == "ETS", ]
fuel_price <- fuel_price[fuel_price$Fuel == "Diesel", ]
```

```
head(fuel_price)
```

```
##          Week       Date   Fuel Variable  Value    Unit Status
## 19896 2017w01 2017-01-06 Diesel      ETS 2.4030 NZD c/L  Final
## 19926 2017w02 2017-01-13 Diesel      ETS 2.4030 NZD c/L  Final
## 19956 2017w03 2017-01-20 Diesel      ETS 2.4030 NZD c/L  Final
## 19986 2017w04 2017-01-27 Diesel      ETS 2.4030 NZD c/L  Final
## 20016 2017w05 2017-02-03 Diesel      ETS 3.0278 NZD c/L  Final
## 20046 2017w06 2017-02-10 Diesel      ETS 3.0278 NZD c/L  Final
```

I have the ETS price in NZD for each week in 2017, since my data is 48 trading periods for each day I will need to adjust the variable.

```
ETS <- fuel_price$Value
ETS <- lapply(ETS, function(x) rep(x, 48*7))
ETS <- unlist(ETS)
ETS <- c(ETS, rep(tail(ETS, 48)))
```

```
csv_files <- list.files(path = "Energy Generation", pattern = "*.csv", full.names = TRUE)

Generation <- csv_files %>% lapply(read.csv) %>% bind_rows()

Generation_Overall_Output <- Generation[, -c(56, 57)] %>%
  pivot_longer(
    cols = matches("^TP\\d+$"), # gets all the trading period columns (TP1 to TP50)
    names_to = "TradingPeriod", # makes a new column containing the old period column names
    names_prefix = "TP", # removes the "TP" prefix from all rows in the TradingPeriod column
    values_to = "Energy_output" # makes a new column for site energy output
  ) %>%
  mutate(
    TradingPeriod = as.integer(TradingPeriod),  # converting the TradingPeriod values to integers
```

```
    Trading_date = as.Date(Trading_date)  # converting Trading_date into a date object
  ) %>%
  group_by(Trading_date, TradingPeriod, Fuel_Code) %>%
  # Treats each unique combination of trading date, trading period and fuel code as its own group
  summarise(
    Energy_output = sum(Energy_output, na.r9m = TRUE),
  # calculates the total energy output for that trading date, period and fuel type)
  # each group by summing the energy output values across all sites
    .groups = "drop"
  )

head(Generation_Overall_Output, 7)
```

Energy generation based on fuel type

```
## # A tibble: 7 x 4
##   Trading_date TradingPeriod Fuel_Code Energy_output
##   <date>               <int> <chr>             <dbl>
## 1 2017-01-01               1 Coal                1
## 2 2017-01-01               1 Diesel              1
## 3 2017-01-01               1 Gas            183799.
## 4 2017-01-01               1 Geo            431504.
## 5 2017-01-01               1 Hydro         1037449.
## 6 2017-01-01               1 Wind           118887.
## 7 2017-01-01               1 Wood            16424
```

There isn't any coal or diesel based energy generation for 2017, this could be because the EMI only reports grid connected energy generation.

Because diesel was not used for grid connected electricity generation in 2017, then diesel ETS prices are not useful for the model.

I was not able to find ETS pricing for other fuels such as gas, geothermal and wood, because of this I have excluded ETS pricing from the model even though ETS prices are important for estimating Energy prices.

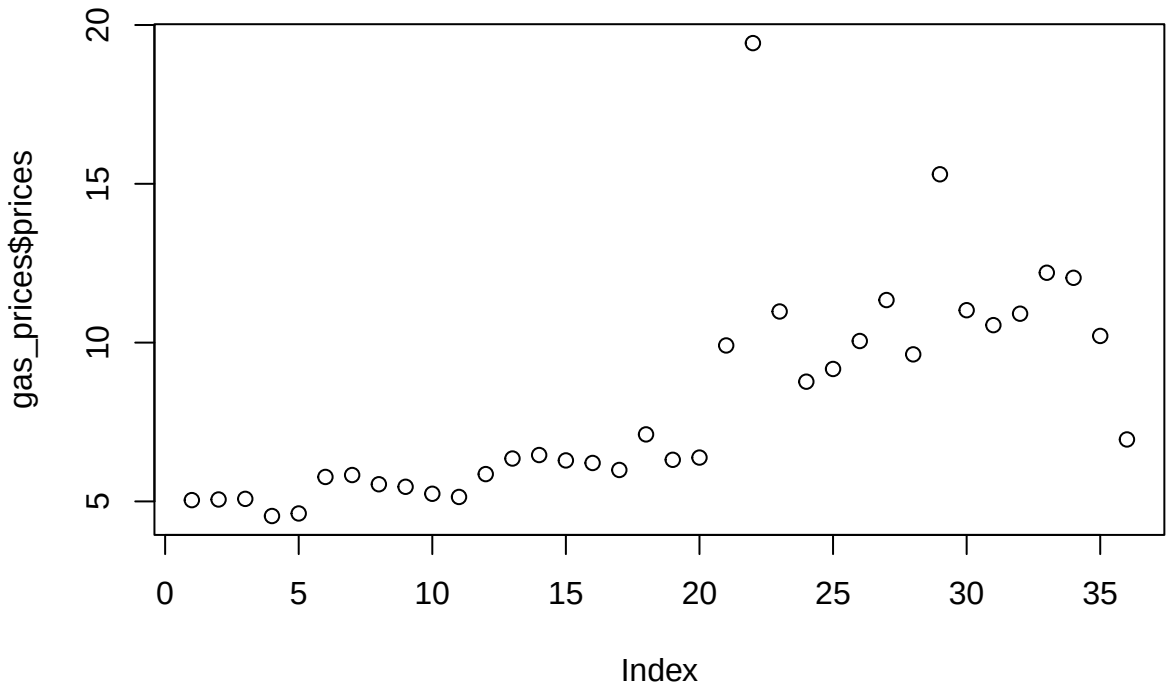
Gas Prices Since around 95% of natural gas in NZ is sold under long term contracts, then most natural gas prices aren't publicly available. However about 5% of natural gas in NZ is traded on the spot market (via emsTradepoint). The spot market prices aren't representative of the whole market, but the spot market prices will still help my model forecast energy prices because natural gas sets the marginal price, as natural gas is the most expensive fuel type for energy generation in NZ. Including natural gas prices in my model will help to capture fuel supply shocks and price spikes that drive marginal generation costs.

Data was obtained from emsTradepoint.

```
gas_prices <- read.csv("Gas Prices/gas_prices.csv")
head(gas_prices)
```

```
##   year    month prices
## 1 2017  January   5.04
## 2 2018 February   5.06
## 3 2019    March   5.08
## 4 2017    April   4.54
## 5 2018      May   4.62
## 6 2019     June   5.77
```

```r
plot(x = gas_prices$prices)
```



I can see from this plot that natural gas prices generally increase over time.

Since I only have the natural gas price for each month of 2017 and my data is 48 trading periods for each day I am adjusting the variable by repeating each gas price about 30 times (1 for each day) and then another 48 times (1 for each trading period).

```r
Gas_Price <- gas_prices[gas_prices$year == 2017, "prices"]
Gas_Price <- lapply(Gas_Price, function(x) rep(x, 30*48))
Gas_Price <- unlist(Gas_Price)
Gas_Price <- c(Gas_Price, rep(tail(Gas_Price, 240)))
```

Combining gas price with the amount of energy generated using natural gas:

```r
Gas_Gen <- Generation_Overall_Output[Generation_Overall_Output$Fuel_Code == "Gas", ]
# Checking for NA's
which(is.na(Gas_Gen), arr.ind = TRUE)
```

```
##        row col
## [1,] 12815   4
## [2,] 12816   4
```

```r
Gas_Gen[12814:12817, ]
```

```
## # A tibble: 4 x 4
##   Trading_date TradingPeriod Fuel_Code Energy_output
##   <date>               <int> <chr>             <dbl>
## 1 2017-09-24              46 Gas              179346.
## 2 2017-09-24              47 Gas                  NA
## 3 2017-09-24              48 Gas                  NA
## 4 2017-09-25               1 Gas              187084.
```

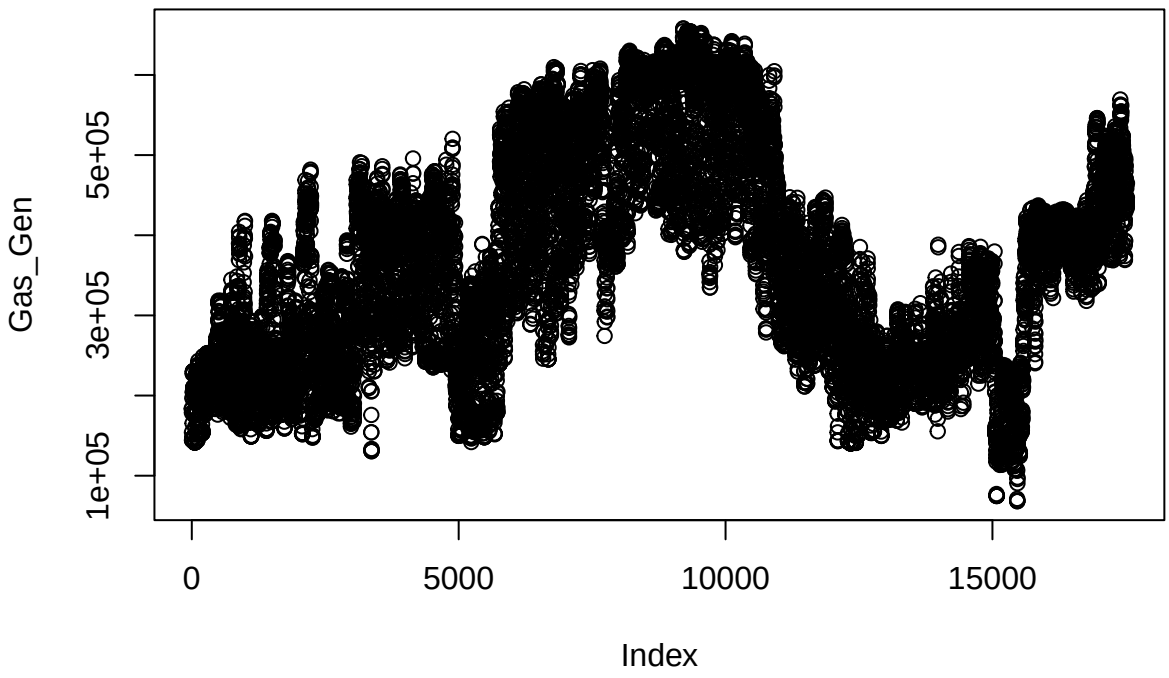
Using linear interpolation to fill gap:

```
# finding the row just before period 47 on 2017-09-24
i_prev <- which(Gas_Gen$Trading_date == as.Date("2017-09-24") &
                Gas_Gen$TradingPeriod == 46)

# Computing the two interpolated values:
# Extracting the two known prices
price_prev <- Gas_Gen$Energy_output[i_prev]
price_next <- Gas_Gen$Energy_output[i_prev + 3]

# runing approx() over the X = {46,49} → Y = {prev,next}, get Y at Xout = {47,48}
interp <- approx(
  x    = c(46, 49),
  y    = c(price_prev, price_next),
  xout = c(47, 48),
  method = "linear"
)

# interp$x == c(47,48); interp$y == interpolated prices
interp
```

```
## $x
## [1] 47 48
##
## $y
## [1] 181925.1 184504.5
```

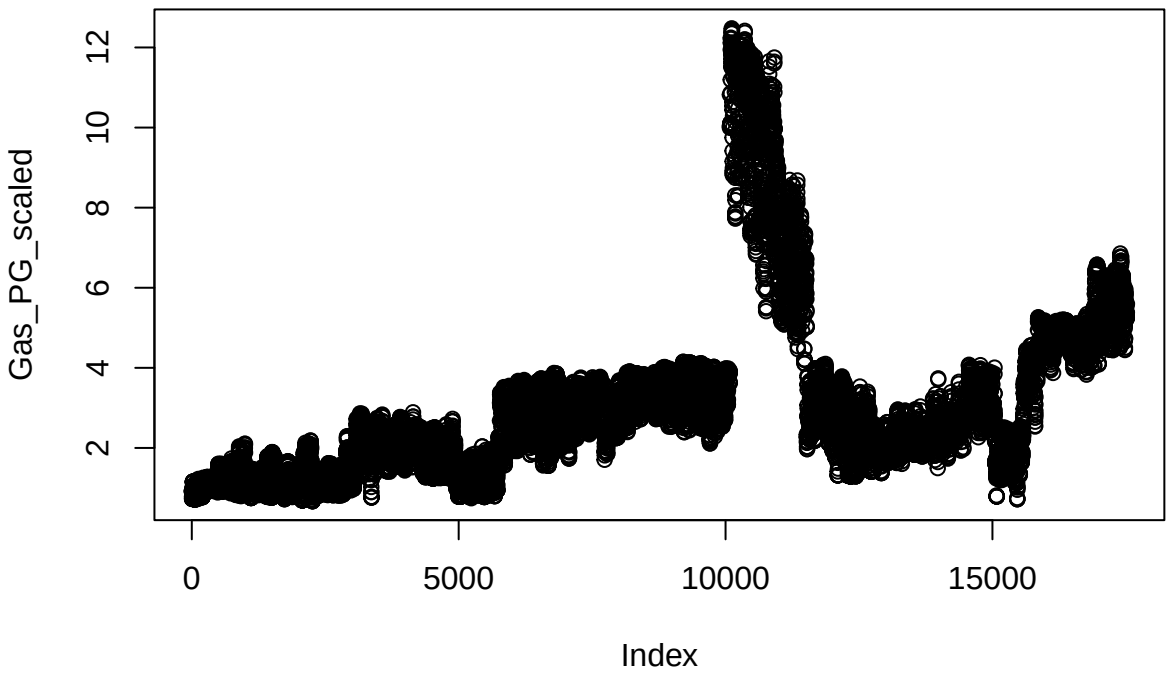
```
Gas_Gen[12815, 4] <- 181925.1
Gas_Gen[12816, 4] <- 184504.5
```

```
Gas_Gen <- Gas_Gen$Energy_output
plot(Gas_Gen)
```




```
Gas_PG <- Gas_Gen * Gas_Price
Gas_PG_scaled <- Gas_PG / 1e6
```

```
plot(Gas_PG_scaled)
```



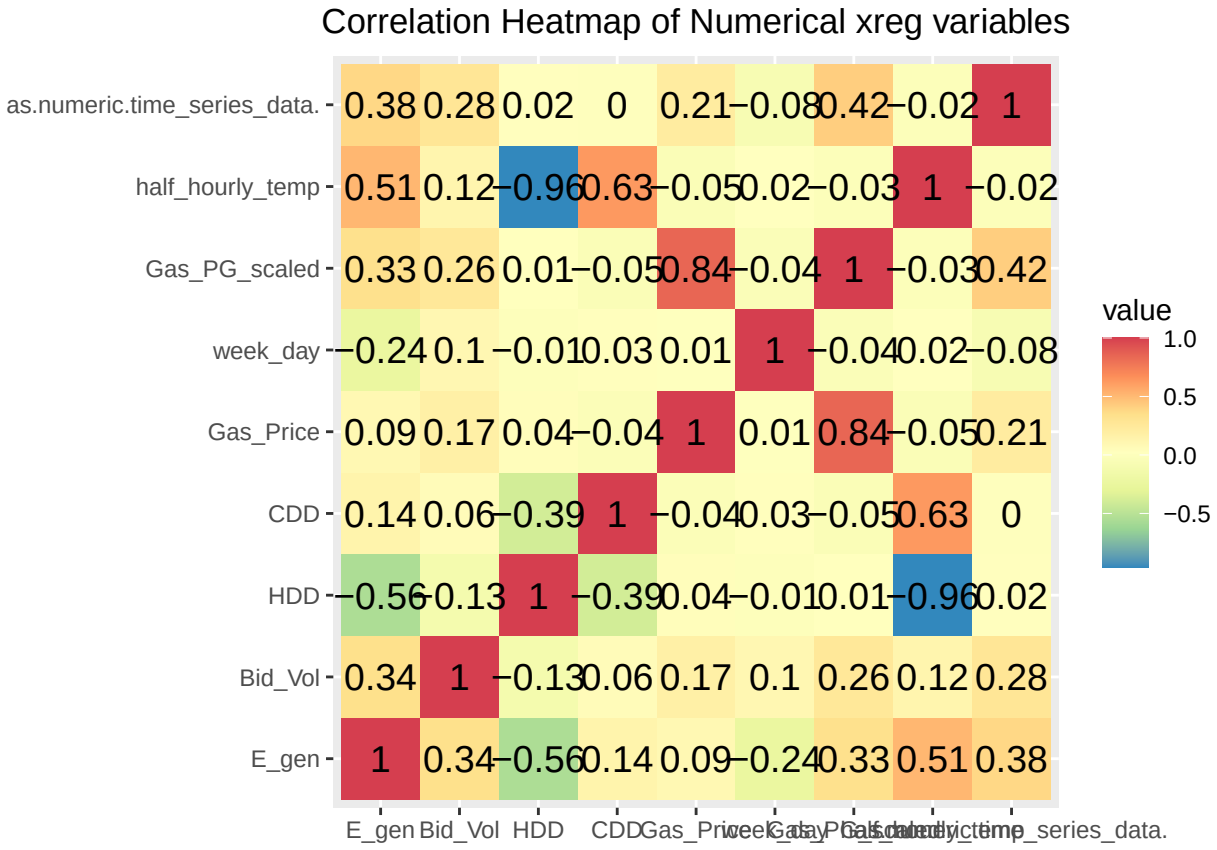
Fitting models with different xreg's

Fitting SARIMAX with different xreg's

```
E_gen <- xreg[, 1]
week_day <- xreg[, 5]

all_variables <- cbind(E_gen, Bid_Vol, HDD, CDD, Gas_Price, week_day, Gas_PG_scaled, half_hourly_temp)
```

Checking feature correlation with response:

```
numeric_vars <- data.frame(all_variables, as.numeric(time_series_data))
cor_matrix1 <- cor(numeric_vars, use = "complete.obs")
cor_matrix <- melt(cor_matrix1)

ggplot(data = cor_matrix, aes(x=Var1, y=Var2, fill=value)) +
  geom_tile() +
  geom_text(aes(label = round(value, 2)), color = "black", size = 5) +
  scale_fill_distiller(palette = "Spectral") +
  labs(title = "Correlation Heatmap of Numerical xreg variables", x = NULL, y = NULL) +
  theme(plot.title = element_text(hjust = 0.5))
```



The Gas Price variable and the Gas price * Energy Generation variable are strongly correlated with a correlation of 0.84

The half_hourly_temp variable is strongly correlated with HDD, CDD and moderately correlated with E_gen.

The reason was half_hourly_temp is moderately correlated with E_gen is presumably because some of the electricity generated will be from solar power. More solar power will be generated in the day when temperatures higher so there will be some correlation between these two variables.

HDD is moderately correlated with E_gen, presumably for similar reasons to those already stated above.

Gas_Price is strongly correlated with Gas_PG_Scaled. Since Gas_PG_scaled is simply Gas_Price * Gas based Energy generation then this isn't surprising.

None of the other variables are strongly correlated with each other.

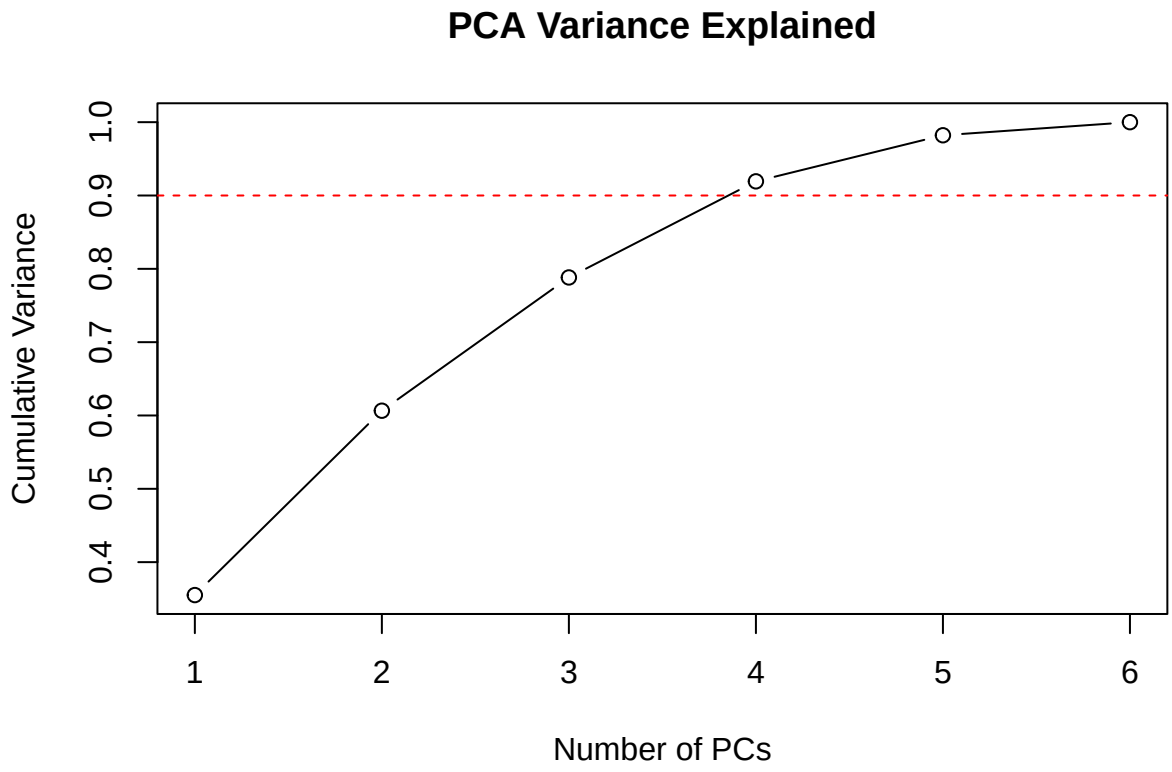
Variables that are strongly correlated with each other will not be placed in the same model to avoid multi-collinearity.

```
xreg_9 <- cbind(E_gen, Bid_Vol, HDD, CDD, Gas_Price)

SARIMAX_model9 <- auto.arima(time_series_data, stepwise = TRUE, approximation = TRUE, xreg = xreg_9,
                             d = 1, max.p = 3, max.q = 3, max.P = 2, max.Q = 2, seasonal = TRUE)
```

```
xreg_10 <- cbind(Bid_Vol, HDD, CDD, Gas_PG_scaled)

SARIMAX_model10 <- auto.arima(time_series_data, stepwise = TRUE, approximation = TRUE, xreg = xreg_10,
                             d = 1, max.p = 3, max.q = 3, max.P = 2, max.Q = 2, seasonal = TRUE)
```

Using PCA components:

```
all_variables <- cbind(E_gen, Bid_Vol, Gas_Price, week_day, Gas_PG_scaled, half_hourly_temp)

pca_mod <- prcomp(scale(all_variables), center = FALSE, scale. = FALSE)

var_explained <- pca_mod$sdev^2 / sum(pca_mod$sdev^2)

plot(cumsum(var_explained), type = "b",
     xlab = "Number of PCs", ylab = "Cumulative Variance",
     main = "PCA Variance Explained")
abline(h = 0.9, lty = 2, col = "red")
```



90% of the variance is captured by around 4 principal components.

```
PC_scores <- as.data.frame(pca_mod$x[, 1:4])
colnames(PC_scores) <- paste0("PC", 1:4)

# Converting to a time series object
PC_ts <- ts(PC_scores, start = c(2017, 1), frequency = frequency(48))
```

```
SARIMAX_model11 <- auto.arima(time_series_data, stepwise = TRUE, approximation = TRUE, xreg = PC_ts,
                              d = 1, max.p = 3, max.q = 3, max.P = 2, max.Q = 2, seasonal = TRUE)
```

```r
xreg_12 <- cbind(E_gen, HDD, Gas_PG_scaled)

SARIMAX_model12 <- auto.arima(time_series_data, stepwise = TRUE, approximation = TRUE, xreg = xreg_12,
                              d = 1, max.p = 3, max.q = 3, max.P = 2, max.Q = 2, seasonal = TRUE)
```

```r
xreg_13 <- cbind(E_gen, Bid_Vol, HDD, Gas_PG_scaled)

SARIMAX_model13 <- auto.arima(time_series_data, stepwise = TRUE, approximation = TRUE, xreg = xreg_13,
                              d = 1, max.p = 3, max.q = 3, max.P = 2, max.Q = 2, seasonal = TRUE)
```

```r
xreg_14 <- cbind(E_gen, Bid_Vol, half_hourly_temp, Gas_PG_scaled)

SARIMAX_model14 <- auto.arima(time_series_data, stepwise = TRUE, approximation = TRUE, xreg = xreg_14,
                              d = 1, max.p = 3, max.q = 3, max.P = 2, max.Q = 2, seasonal = TRUE)
```

TBATS

Training a TBATS model on cleaned data (removing outliers)

```r
outliers  <- tsoutliers(time_series_data)
cleaned_ts <- replace(time_series_data, outliers$index, outliers$replacements)
TBATS_model_2 <- tbats(cleaned_ts)
```

```r
show(TBATS_model_2)
```

```
## TBATS(0.542, {0,0}, 0.809, {<48,7>})
##
## Call: tbats(y = cleaned_ts)
##
## Parameters
##   Lambda: 0.542258
##   Alpha: 0.8655062
##   Beta: -0.1742691
##   Damping Parameter: 0.808606
##   Gamma-1 Values: 0.003619715
##   Gamma-2 Values: -0.0003076263
##
## Seed States:
##                [,1]
##  [1,] 12.1544577469
##  [2,] -0.4587706041
##  [3,] -0.6130534746
##  [4,] -0.1009844007
##  [5,]  0.2211227783
##  [6,]  0.0873445261
##  [7,] -0.0824472184
##  [8,] -0.0213831123
##  [9,] -0.0182741425
## [10,] -0.4349542114
## [11,] -0.5947210576
## [12,] -0.0086095156
```

```
## [13,]  0.1075248286
## [14,]  0.0006883303
## [15,]  0.0837554107
## [16,]  0.0531706823
## attr(,"lambda")
## [1] 0.5422578
##
## Sigma: 1.515064
## AIC: 252474.5
```

SARIMA-fiGARCH

Extracting the residuals from my sarima model:

```r
residuals_sarima <- residuals(SARIMA_model1)

spec <- ugarchspec(
  variance.model = list(model = "fiGARCH", garchOrder = c(1, 1)),
  mean.model     = list(armaOrder = c(1, 0)),
  distribution.model = "ged")

GARCH_model_Sarima <- ugarchfit(spec = spec, data = residuals_sarima, solver = "hybrid")
show(GARCH_model_Sarima)
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : fiGARCH(1,1)
## Mean Model   : ARFIMA(1,0,0)
## Distribution : ged
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      -0.21324    0.018662 -11.4263 0.000000
## ar1      0.25524    0.006144  41.5404 0.000000
## omega    6.33423    0.813900   7.7826 0.000000
## alpha1   0.16260    0.055142   2.9487 0.003191
## beta1    0.38196    0.059649   6.4035 0.000000
## delta    0.57513    0.022696  25.3408 0.000000
## shape    0.80513    0.009673  83.2363 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      -0.21324    0.013207 -16.1450 0.000000
## ar1      0.25524    0.007994  31.9268 0.000000
## omega    6.33423    1.067272   5.9350 0.000000
## alpha1   0.16260    0.055209   2.9452 0.003228
## beta1    0.38196    0.063499   6.0152 0.000000
```

```
## delta    0.57513    0.027926  20.5949 0.000000
## shape    0.80513    0.014016  57.4454 0.000000
##
## LogLikelihood : -62432.19
##
## Information Criteria
## ------------------------------------
##
## Akaike       7.1278
## Bayes        7.1309
## Shibata      7.1278
## Hannan-Quinn 7.1288
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      79.20       0
## Lag[2*(p+q)+(p+q)-1][2]     79.75       0
## Lag[4*(p+q)+(p+q)-1][5]     84.00       0
## d.o.f=1
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                    0.09165  0.7621
## Lag[2*(p+q)+(p+q)-1][5]   0.42921  0.9680
## Lag[4*(p+q)+(p+q)-1][9]   1.10163  0.9817
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]    0.4389 0.500 2.000  0.5076
## ARCH Lag[5]    0.5001 1.440 1.667  0.8836
## ARCH Lag[7]    0.9453 2.315 1.543  0.9223
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  5.2665
## Individual Statistics:
## mu     0.9308
## ar1    0.5879
## omega  2.1557
## alpha1 0.5318
## beta1  0.9600
## delta  0.9935
## shape  0.3579
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.69 1.9 2.35
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
```

```
## -------------------------------------
##                     t-value    prob sig
## Sign Bias             2.030 0.04242  **
## Negative Sign Bias    1.014 0.31069
## Positive Sign Bias    0.367 0.71365
## Joint Effect          8.278 0.04060  **
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## -------------------------------------
##   group statistic p-value(g-1)
## 1    20     102.9    1.596e-13
## 2    30     113.9    5.220e-12
## 3    40     135.7    1.265e-12
## 4    50     139.7    1.180e-10
##
##
## Elapsed time : 9.310382
```

Extracting the residuals from my SARIMAX model:

```
residuals_sarimax <- residuals(SARIMAX_model1)

spec <- ugarchspec(
  variance.model = list(model = "fiGARCH", garchOrder = c(1, 1)),
  mean.model     = list(armaOrder = c(1, 0)),
  distribution.model = "ged")

GARCH_model_Sarimax <- ugarchfit(spec = spec, data = residuals_sarimax, solver = "hybrid")
show(GARCH_model_Sarimax)
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : fiGARCH(1,1)
## Mean Model   : ARFIMA(1,0,0)
## Distribution : ged
##
## Optimal Parameters
## -----------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      -0.18261    0.020658  -8.8397 0.000000
## ar1      0.28939    0.007274  39.7813 0.000000
## omega    8.63910    1.014344   8.5169 0.000000
## alpha1   0.11775    0.057266   2.0563 0.039758
## beta1    0.34266    0.061986   5.5281 0.000000
## delta    0.60681    0.023514  25.8063 0.000000
## shape    0.84627    0.010267  82.4228 0.000000
##
```

```
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      -0.18261    0.014093 -12.9572 0.000000
## ar1      0.28939    0.010990  26.3331 0.000000
## omega    8.63910    1.344901   6.4236 0.000000
## alpha1   0.11775    0.061105   1.9271 0.053968
## beta1    0.34266    0.070441   4.8646 0.000001
## delta    0.60681    0.032567  18.6326 0.000000
## shape    0.84627    0.016697  50.6846 0.000000
##
## LogLikelihood : -62656.44
##
## Information Criteria
## ------------------------------------
##
## Akaike       7.1534
## Bayes        7.1565
## Shibata      7.1534
## Hannan-Quinn 7.1544
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      104.3       0
## Lag[2*(p+q)+(p+q)-1][2]     104.8       0
## Lag[4*(p+q)+(p+q)-1][5]     111.0       0
## d.o.f=1
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                    0.03343  0.8549
## Lag[2*(p+q)+(p+q)-1][5]   0.33004  0.9805
## Lag[4*(p+q)+(p+q)-1][9]   0.82298  0.9924
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]    0.4246 0.500 2.000  0.5146
## ARCH Lag[5]    0.4308 1.440 1.667  0.9038
## ARCH Lag[7]    0.7486 2.315 1.543  0.9507
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  6.075
## Individual Statistics:
## mu     0.09752
## ar1    0.22483
## omega  4.05186
## alpha1 0.88730
## beta1  1.83103
## delta  1.35693
```

```
## shape  0.27335
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.69 1.9 2.35
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                     t-value    prob sig
## Sign Bias          1.955208 0.05057   *
## Negative Sign Bias 0.521383 0.60211
## Positive Sign Bias 0.006064 0.99516
## Joint Effect       5.631683 0.13097
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     92.01    1.460e-11
## 2    30    105.95    1.070e-10
## 3    40    109.00    1.536e-08
## 4    50    132.23    1.411e-09
##
##
## Elapsed time : 10.09306
```

Extracting the residuals from my TBATS model:

```r
residuals_TBATS <- residuals(TBATS_model_2)

spec <- ugarchspec(
  variance.model = list(model = "fiGARCH", garchOrder = c(1, 1)),
  mean.model     = list(armaOrder = c(1, 0)),
  distribution.model = "ged")

GARCH_model_TBATS <- ugarchfit(spec = spec, data = residuals_TBATS, solver = "hybrid")
show(GARCH_model_TBATS)
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## ------------------------------------
## GARCH Model  : fiGARCH(1,1)
## Mean Model   : ARFIMA(1,0,0)
## Distribution : ged
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.022656    0.003144  7.20564  0.00000
```

```
## ar1     0.159470    0.005751 27.72818  0.00000
## omega   0.190085    0.037641  5.04992  0.00000
## alpha1  0.085601    0.128492  0.66619  0.50529
## beta1   0.181334    0.137108  1.32257  0.18598
## delta   0.389667    0.028345 13.74755  0.00000
## shape   0.910209    0.011596 78.49387  0.00000
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.022656    0.002133 10.62340 0.000000
## ar1     0.159470    0.005912 26.97364 0.000000
## omega   0.190085    0.039917  4.76202 0.000002
## alpha1  0.085601    0.112144  0.76331 0.445278
## beta1   0.181334    0.121793  1.48887 0.136522
## delta   0.389667    0.027358 14.24334 0.000000
## shape   0.910209    0.016215 56.13289 0.000000
##
## LogLikelihood : -28167.74
##
## Information Criteria
## ------------------------------------
##
## Akaike       3.2163
## Bayes        3.2194
## Shibata      3.2163
## Hannan-Quinn 3.2173
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      109.7       0
## Lag[2*(p+q)+(p+q)-1][2]     115.2       0
## Lag[4*(p+q)+(p+q)-1][5]     119.7       0
## d.o.f=1
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      1.847  0.1741
## Lag[2*(p+q)+(p+q)-1][5]     3.068  0.3948
## Lag[4*(p+q)+(p+q)-1][9]     4.504  0.5057
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3] 0.0000793 0.500 2.000  0.9929
## ARCH Lag[5] 1.3759644 1.440 1.667  0.6255
## ARCH Lag[7] 2.4753458 2.315 1.543  0.6175
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  3.8545
```

```
## Individual Statistics:
## mu     0.2423
## ar1    0.0664
## omega  2.1319
## alpha1 0.3676
## beta1  0.7897
## delta  0.7379
## shape  1.3826
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.69 1.9 2.35
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value     prob sig
## Sign Bias            2.157 0.030988  **
## Negative Sign Bias   2.325 0.020083  **
## Positive Sign Bias   1.911 0.056009   *
## Joint Effect        15.783 0.001256 ***
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     73.47    2.418e-08
## 2    30     88.67    5.829e-08
## 3    40     92.68    2.907e-06
## 4    50    101.35    1.634e-05
##
##
## Elapsed time : 13.87629
```

Increasing the AR and MA terms for SARIMA-fiGARCH

```r
residuals_sarima2 <- residuals(SARIMA_model1)

spec <- ugarchspec(
  variance.model = list(model = "fiGARCH", garchOrder = c(1, 2)),
  mean.model     = list(armaOrder = c(2, 1)),
  distribution.model = "ged")

GARCH_model_Sarima2 <- ugarchfit(spec = spec, data = residuals_sarima2, solver = "hybrid")
show(GARCH_model_Sarima2)
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## ------------------------------------
## GARCH Model  : fiGARCH(1,2)
```

```
## Mean Model   : ARFIMA(2,0,1)
## Distribution : ged
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error   t value Pr(>|t|)
## mu      0.024770    1.0e-06  29858.27        0
## ar1    -0.000012    0.0e+00   -362.05        0
## ar2     0.001712    0.0e+00  29855.53        0
## ma1     0.000018    0.0e+00  29855.59        0
## omega   0.380742    1.3e-05  29846.52        0
## alpha1  0.051748    2.0e-06  29846.52        0
## beta1   0.238462    1.0e-06 168235.81        0
## beta2   0.450000    2.0e-06 193784.79        0
## delta   0.511888    3.0e-06 192900.39        0
## shape   2.000000    7.0e-06 275208.72        0
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.024770         NaN      NaN      NaN
## ar1    -0.000012         NaN      NaN      NaN
## ar2     0.001712         NaN      NaN      NaN
## ma1     0.000018         NaN      NaN      NaN
## omega   0.380742         NaN      NaN      NaN
## alpha1  0.051748         NaN      NaN      NaN
## beta1   0.238462         NaN      NaN      NaN
## beta2   0.450000         NaN      NaN      NaN
## delta   0.511888         NaN      NaN      NaN
## shape   2.000000         NaN      NaN      NaN
##
## LogLikelihood : -83173.19
##
## Information Criteria
## ------------------------------------
##
## Akaike       9.4958
## Bayes        9.5002
## Shibata      9.4958
## Hannan-Quinn 9.4973
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                       0.836  0.3606
## Lag[2*(p+q)+(p+q)-1][8]      2.476  1.0000
## Lag[4*(p+q)+(p+q)-1][14]     7.382  0.4856
## d.o.f=3
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                   8.401e-05  0.9927
## Lag[2*(p+q)+(p+q)-1][8]  2.611e-04  1.0000
```

```
## Lag[4*(p+q)+(p+q)-1][14] 1.165e-02  1.0000
## d.o.f=3
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[4] 2.309e-05 0.500 2.000  0.9962
## ARCH Lag[6] 1.193e-04 1.461 1.711  1.0000
## ARCH Lag[8] 2.160e-04 2.368 1.583  1.0000
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  NaN
## Individual Statistics:
## mu     NaN
## ar1    NaN
## ar2    NaN
## ma1    NaN
## omega  NaN
## alpha1 NaN
## beta1  NaN
## beta2  NaN
## delta  NaN
## shape  NaN
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          2.29 2.54 3.05
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                     t-value   prob sig
## Sign Bias          1.062790 0.2879
## Negative Sign Bias 0.009211 0.9927
## Positive Sign Bias 0.362979 0.7166
## Joint Effect       1.259951 0.7387
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20      2605            0
## 2    30      2843            0
## 3    40      2992            0
## 4    50      3158            0
##
##
## Elapsed time : 1.549957
```

```r
residuals_sarima3 <- residuals(SARIMA_model1)

spec <- ugarchspec(
  variance.model = list(model = "fiGARCH", garchOrder = c(1, 1)),
  mean.model     = list(armaOrder = c(2, 1)),
```

```
  distribution.model = "ged")

GARCH_model_Sarima3 <- ugarchfit(spec = spec, data = residuals_sarima3, solver = "hybrid")
show(GARCH_model_Sarima3)
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : fiGARCH(1,1)
## Mean Model   : ARFIMA(2,0,1)
## Distribution : ged
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      -0.54518    0.020763 -26.2575  0.00000
## ar1      1.09323    0.013416  81.4899  0.00000
## ar2     -0.13261    0.004671 -28.3920  0.00000
## ma1     -0.82821    0.015299 -54.1337  0.00000
## omega    8.99438    1.137470   7.9074  0.00000
## alpha1   0.13916    0.059550   2.3368  0.01945
## beta1    0.35067    0.064444   5.4415  0.00000
## delta    0.59081    0.025007  23.6256  0.00000
## shape    0.68767    0.008382  82.0422  0.00000
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      -0.54518    0.036201 -15.0597 0.000000
## ar1      1.09323    0.062560  17.4749 0.000000
## ar2     -0.13261    0.022741  -5.8312 0.000000
## ma1     -0.82821    0.071422 -11.5960 0.000000
## omega    8.99438    1.432685   6.2780 0.000000
## alpha1   0.13916    0.056454   2.4649 0.013703
## beta1    0.35067    0.065753   5.3332 0.000000
## delta    0.59081    0.031520  18.7440 0.000000
## shape    0.68767    0.015885  43.2896 0.000000
##
## LogLikelihood : -61856.46
##
## Information Criteria
## ------------------------------------
##
## Akaike       7.0623
## Bayes        7.0663
## Shibata      7.0623
## Hannan-Quinn 7.0636
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
```

```
##                           statistic   p-value
## Lag[1]                        58.93 1.632e-14
## Lag[2*(p+q)+(p+q)-1][8]      132.01 0.000e+00
## Lag[4*(p+q)+(p+q)-1][14]     153.07 0.000e+00
## d.o.f=3
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                    0.001567  0.9684
## Lag[2*(p+q)+(p+q)-1][5]   0.281928  0.9855
## Lag[4*(p+q)+(p+q)-1][9]   0.826196  0.9923
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]    0.2548 0.500 2.000  0.6137
## ARCH Lag[5]    0.4156 1.440 1.667  0.9082
## ARCH Lag[7]    0.7682 2.315 1.543  0.9481
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  5.9041
## Individual Statistics:
## mu     0.2161
## ar1    0.1992
## ar2    0.2358
## ma1    0.2102
## omega  3.7523
## alpha1 0.5251
## beta1  1.3893
## delta  1.0515
## shape  0.4848
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          2.1 2.32 2.82
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value   prob sig
## Sign Bias          1.53365 0.1251
## Negative Sign Bias 1.21594 0.2240
## Positive Sign Bias 0.04113 0.9672
## Joint Effect       5.58269 0.1338
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     127.6    4.069e-18
## 2    30     148.6    5.079e-18
```

```
## 3    40     156.6    4.996e-16
## 4    50     161.4    6.384e-14
##
##
## Elapsed time : 22.49655
```

```
residuals_sarima4 <- residuals(SARIMA_model1)

spec <- ugarchspec(
  variance.model = list(model = "fiGARCH", garchOrder = c(2, 1)),
  mean.model     = list(armaOrder = c(2, 0)),
  distribution.model = "ged")

GARCH_model_Sarima4 <- ugarchfit(spec = spec, data = residuals_sarima4, solver = "hybrid")
show(GARCH_model_Sarima4)
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : fiGARCH(2,1)
## Mean Model   : ARFIMA(2,0,0)
## Distribution : ged
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error    t value Pr(>|t|)
## mu     -0.226698    0.022236 -10.194940 0.000000
## ar1     0.251589    0.005903  42.619748 0.000000
## ar2     0.059244    0.002872  20.628790 0.000000
## omega   6.666919    1.200992   5.551176 0.000000
## alpha1  0.144200    0.095099   1.516308 0.129441
## alpha2  0.000000    0.025981   0.000001 0.999999
## beta1   0.365956    0.101006   3.623133 0.000291
## delta   0.577863    0.022831  25.310938 0.000000
## shape   0.794630    0.009547  83.231120 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error    t value Pr(>|t|)
## mu     -0.226698    0.017005 -13.331416 0.000000
## ar1     0.251589    0.007112  35.373478 0.000000
## ar2     0.059244    0.001907  31.073608 0.000000
## omega   6.666919    1.171197   5.692396 0.000000
## alpha1  0.144200    0.065606   2.197976 0.027951
## alpha2  0.000000    0.022428   0.000001 0.999999
## beta1   0.365956    0.073088   5.007083 0.000001
## delta   0.577863    0.027878  20.728132 0.000000
## shape   0.794630    0.013815  57.520495 0.000000
##
## LogLikelihood : -62383.52
##
```

```
## Information Criteria
## ------------------------------------
##
## Akaike       7.1224
## Bayes        7.1264
## Shibata      7.1224
## Hannan-Quinn 7.1237
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic  p-value
## Lag[1]                      69.20 1.11e-16
## Lag[2*(p+q)+(p+q)-1][5]     99.51 0.00e+00
## Lag[4*(p+q)+(p+q)-1][9]    104.04 0.00e+00
## d.o.f=2
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                     0.06392  0.8004
## Lag[2*(p+q)+(p+q)-1][8]    0.84380  0.9836
## Lag[4*(p+q)+(p+q)-1][14]   2.72571  0.9690
## d.o.f=3
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[4]   0.08358 0.500 2.000  0.7725
## ARCH Lag[6]   0.24768 1.461 1.711  0.9579
## ARCH Lag[8]   1.07950 2.368 1.583  0.9123
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  6.5464
## Individual Statistics:
## mu     0.8328
## ar1    0.6748
## ar2    0.9980
## omega  2.2340
## alpha1 0.5117
## alpha2 0.2700
## beta1  0.9403
## delta  0.9779
## shape  0.3870
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          2.1 2.32 2.82
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value    prob sig
## Sign Bias           2.2563 0.02407  **
```

```
## Negative Sign Bias  0.9549 0.33964
## Positive Sign Bias  0.2295 0.81849
## Joint Effect        9.0867 0.02816  **
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     106.8    3.042e-14
## 2    30     136.2    7.898e-16
## 3    40     137.3    7.046e-13
## 4    50     157.3    2.741e-13
##
##
## Elapsed time : 22.64651
```

Reevaluation on Validation set

Forecasting

ARIMA

I made a short custom function for fitting the model, inverting the Box-Cox and converting it to a time series because I felt that doing this for every model was taking up too much space.

```
Forecasting_model <- function(model, h, is_box = FALSE) {
  forecast_model <- forecast(model, h)
  if(is_box) {
    forecast_model <- InvBoxCox(forecast_model$mean, lambda)
    print("inverted boxcox")
  }
  forecast_model <- ts(forecast_model, start = c(2018, 1), frequency = 48)
}
```

```
forecast_ARIMA2 <- Forecasting_model(ARIMA_model2, 17520, is_box = TRUE)
```

```
## [1] "inverted boxcox"
```

```
forecast_ARIMA3 <- Forecasting_model(ARIMA_model3, 17520, is_box = TRUE)
```

```
## [1] "inverted boxcox"
```

```
forecast_ARIMA4 <- Forecasting_model(ARIMA_model4, 17520, is_box = TRUE)
```

```
## [1] "inverted boxcox"
```

```
forecast_ARIMA5 <- Forecasting_model(ARIMA_model5, 17520, is_box = TRUE)
```

```
## [1] "inverted boxcox"
```

```
forecast_ARIMA6 <- Forecasting_model(ARIMA_model6, 17520, is_box = FALSE)
```

```
forecast_ARIMA7 <- Forecasting_model(ARIMA_model7, 17520, is_box = FALSE)
```

```
forecast_ARIMA8 <- Forecasting_model(ARIMA_model8, 17520, is_box = FALSE)
```

SARIMA

```
forecast_SARIMA2 <- Forecasting_model(SARIMA_model2, 17520, is_box = TRUE)
```

```
## [1] "inverted boxcox"
```

```
forecast_SARIMA3 <- Forecasting_model(SARIMA_model3, 17520, is_box = TRUE)
```

```
## [1] "inverted boxcox"
```

```
forecast_SARIMA4 <- Forecasting_model(SARIMA_model4, 17520, is_box = TRUE)
```

```
## [1] "inverted boxcox"
```

```
forecast_SARIMA5 <- Forecasting_model(SARIMA_model5, 17520, is_box = TRUE)
```

```
## [1] "inverted boxcox"
```

```
forecast_SARIMA6 <- Forecasting_model(SARIMA_model6, 17520, is_box = TRUE)
```

```
## [1] "inverted boxcox"
```

```
forecast_SARIMA7 <- Forecasting_model(SARIMA_model7, 17520, is_box = FALSE)
```

```
forecast_SARIMA8 <- Forecasting_model(SARIMA_model8, 17520, is_box = FALSE)
```

```
forecast_SARIMA9 <- Forecasting_model(SARIMA_model9, 17520, is_box = FALSE)
```

```
forecast_SARIMA10 <- Forecasting_model(SARIMA_model10, 17520, is_box = FALSE)
```

STL-ARIMA

```
forecast_STL2 <- Forecasting_model(STL_ARIMA_model2, 17520, is_box = TRUE)
```

```
## [1] "inverted boxcox"
```

```
forecast_STL3 <- Forecasting_model(STL_ARIMA_model3, 17520, is_box = TRUE)
```

```
## [1] "inverted boxcox"
```

```
forecast_STL4 <- Forecasting_model(STL_ARIMA_model4, 17520, is_box = TRUE)
```

```
## [1] "inverted boxcox"
```

```
forecast_STL5 <- Forecasting_model(STL_ARIMA_model5, 17520, is_box = TRUE)
```

```
## [1] "inverted boxcox"
```

```
forecast_STL6 <- Forecasting_model(STL_ARIMA_model6, 17520, is_box = FALSE)
```

```
forecast_STL7 <- Forecasting_model(STL_ARIMA_model7, 17520, is_box = FALSE)
```

```
forecast_STL8 <- Forecasting_model(STL_ARIMA_model8, 17520, is_box = FALSE)
```

SARIMAX

```
Forecasting_model_X <- function(model, h, is_box = FALSE, xreg_future) {
  forecast_model <- forecast(model, h, xreg = xreg_future)
  if(is_box) {
    forecast_model <- InvBoxCox(forecast_model$mean, lambda)
    print("inverted boxcox")
  }
  forecast_model <- ts(forecast_model, start = c(2018, 1), frequency = 48)
}
```

```
forecast_SARIMAX2 <- Forecasting_model_X(SARIMAX_model2, 17520, is_box = TRUE, xreg_future)
```

```
## [1] "inverted boxcox"
```

```
forecast_SARIMAX3 <- Forecasting_model_X(SARIMAX_model3, 17520, is_box = TRUE, xreg_future)
```

```
## [1] "inverted boxcox"
```

```
forecast_SARIMAX4 <- Forecasting_model_X(SARIMAX_model4, 17520, is_box = TRUE, xreg_future)
```

```
## [1] "inverted boxcox"
```

```
forecast_SARIMAX5 <- Forecasting_model_X(SARIMAX_model5, 17520, is_box = TRUE, xreg_future)
```

```
## [1] "inverted boxcox"
```

```
forecast_SARIMAX6 <- Forecasting_model_X(SARIMAX_model6, 17520, is_box = FALSE, xreg_future)
```

```
forecast_SARIMAX7 <- Forecasting_model_X(SARIMAX_model7, 17520, is_box = FALSE, xreg_future)
```

```
forecast_SARIMAX8 <- Forecasting_model_X(SARIMAX_model8, 17520, is_box = FALSE, xreg_future)
```

SARIMAX with different xreg Making future xreg for forecasting:

I'm re-using the 2018 forecasts I made for energy generation and weekday from the model training file.

I'm making forecasts for Bid Volume, HDD, CDD, Gas prices, and Gas prices * Energy generation using natural gas.

```
Bid_Vol_est <- auto.arima(Bid_Vol, approximation = TRUE)
Gas_Price_est <- auto.arima(Gas_Price, approximation = TRUE)
Gas_PG_scaled_est <- auto.arima(Gas_PG_scaled, approximation = TRUE)
# I'm making a temperature model and forecast, then obtaining HDD and CDD from this.
temperature_est <- auto.arima(half_hourly_temp, approximation = TRUE)
```

```
BID_forecast <- forecast(Bid_Vol_est, h = 17520)
temperature_forecast <- forecast(temperature_est, h = 17520)
HDD_forecast <- pmax(0, 18 - temperature_forecast$mean)
CDD_forecast <- pmax(0, temperature_forecast$mean - 18)
Gas_Price_forecast <- forecast(Gas_Price_est, h = 17520)
Gas_PG_scaled_forecast <- forecast(Gas_PG_scaled_est, h = 17520)
```

PCA components:

```
EGen_est <- xreg_future[,1]
Week_day_est <- xreg_future[,5]

all_variables_est <- cbind(as.numeric(EGen_est),
                           as.numeric(BID_forecast$mean),
                           as.numeric(Gas_Price_forecast$mean),
                           as.numeric(Week_day_est),
                           as.numeric(Gas_PG_scaled_forecast$mean),
                           as.numeric(temperature_forecast$mean))

all_variables_scaled <- scale(all_variables_est)

PC_new <- as.matrix(all_variables_scaled) %*% pca_mod$rotation
PC_new <- PC_new[, 1:4]
PC_new_ts <- ts(PC_new, start = c(2018, 1) + c(0,1), frequency = frequency(48))
```

```
xreg_future_9 <- cbind(as.numeric(EGen_est),
                       as.numeric(BID_forecast$mean),
                       as.numeric(HDD_forecast),
                       as.numeric(CDD_forecast),
                       as.numeric(Gas_Price_forecast$mean)
                       )
```

```
xreg_future_10 <- cbind(as.numeric(BID_forecast$mean),
                        as.numeric(HDD_forecast),
                        as.numeric(CDD_forecast),
                        as.numeric(Gas_PG_scaled_forecast$mean)
                         )

xreg_future_12 <- cbind(as.numeric(EGen_est),
                        as.numeric(HDD_forecast),
                        as.numeric(Gas_PG_scaled_forecast$mean)
                         )

xreg_future_13 <- cbind(as.numeric(EGen_est),
                        as.numeric(BID_forecast$mean),
                        as.numeric(HDD_forecast),
                        as.numeric(Gas_PG_scaled_forecast$mean)
                         )

xreg_future_14 <- cbind(as.numeric(EGen_est),
                        as.numeric(BID_forecast$mean),
                        as.numeric(temperature_forecast$mean),
                        as.numeric(Gas_PG_scaled_forecast$mean)
                         )
```

Ensuring the future xreg contains the same names as the xreg used for training.

```
colnames(xreg_9)
```

```
## [1] "E_gen"     "Bid_Vol"   "HDD"       "CDD"       "Gas_Price"
```

```
colnames(xreg_future_9)
```

```
## NULL
```

```
colnames(xreg_10)
```

```
## [1] "Bid_Vol"       "HDD"           "CDD"           "Gas_PG_scaled"
```

```
colnames(xreg_future_10)
```

```
## NULL
```

```
colnames(xreg_12)
```

```
## [1] "E_gen"         "HDD"           "Gas_PG_scaled"
```

```
colnames(xreg_future_12)
```

```
## NULL
```

```r
colnames(xreg_13)
```

```
## [1] "E_gen"         "Bid_Vol"       "HDD"           "Gas_PG_scaled"
```

```r
colnames(xreg_future_13)
```

```
## NULL
```

```r
colnames(xreg_14)
```

```
## [1] "E_gen"            "Bid_Vol"          "half_hourly_temp" "Gas_PG_scaled"
```

```r
colnames(xreg_future_14)
```

```
## NULL
```

```r
colnames(xreg_future_9) <- c("E_gen", "Bid_Vol", "HDD", "CDD", "Gas_Price")
colnames(xreg_future_10) <- c("Bid_Vol", "HDD", "CDD", "Gas_PG_scaled")
colnames(xreg_future_12) <- c("E_gen", "HDD", "Gas_PG_scaled")
colnames(xreg_future_13) <- c("E_gen", "Bid_Vol", "HDD", "Gas_PG_scaled")
colnames(xreg_future_14) <- c("E_gen", "Bid_Vol", "half_hourly_temp", "Gas_PG_scaled")
```

```r
SARIM_forecast <- forecast(SARIMAX_model9, h = 17520, xreg = xreg_future_9)
Forecast_SARIMAX_9 <- ts(SARIM_forecast$mean, start = c(2018, 1), frequency = 48)

SARIM_forecast2 <- forecast(SARIMAX_model10, h = 17520, xreg = xreg_future_10)
Forecast_SARIMAX_10 <- ts(SARIM_forecast2$mean, start = c(2018, 1), frequency = 48)

SARIM_forecast3 <- forecast(SARIMAX_model11, h = 17520, xreg = PC_new_ts)
Forecast_SARIMAX_11 <- ts(SARIM_forecast3$mean, start = c(2018, 1), frequency = 48)

SARIM_forecast4 <- forecast(SARIMAX_model12, h = 17520, xreg = xreg_future_12)
Forecast_SARIMAX_12 <- ts(SARIM_forecast4$mean, start = c(2018, 1), frequency = 48)

SARIM_forecast5 <- forecast(SARIMAX_model13, h = 17520, xreg = xreg_future_13)
Forecast_SARIMAX_13 <- ts(SARIM_forecast5$mean, start = c(2018, 1), frequency = 48)

SARIM_forecast6 <- forecast(SARIMAX_model14, h = 17520, xreg = xreg_future_14)
Forecast_SARIMAX_14 <- ts(SARIM_forecast6$mean, start = c(2018, 1), frequency = 48)
```

SARIMA-fiGARCH

```r
forecast_raw_Garch <- ugarchforecast(GARCH_model_Sarima, n.ahead = 17520)
forecast_GARCH_SARIMA <- ts(fitted(forecast_raw_Garch), start = c(2018, 1), frequency = 48)
```

```r
forecast_raw_Garch <- ugarchforecast(GARCH_model_Sarima2, n.ahead = 17520)
```

```
## Warning in sqrt(ans$h): NaNs produced
```

```
forecast_GARCH_SARIMA2 <- ts(fitted(forecast_raw_Garch), start = c(2018, 1), frequency = 48)
```

```
forecast_raw_Garch <- ugarchforecast(GARCH_model_Sarima3, n.ahead = 17520)
forecast_GARCH_SARIMA3 <- ts(fitted(forecast_raw_Garch), start = c(2018, 1), frequency = 48)
```

```
forecast_raw_Garch <- ugarchforecast(GARCH_model_Sarima4, n.ahead = 17520)
forecast_GARCH_SARIMA4 <- ts(fitted(forecast_raw_Garch), start = c(2018, 1), frequency = 48)
```

```
forecast_raw_Garch <- ugarchforecast(GARCH_model_TBATS, n.ahead = 17520)
forecast_GARCH_TBATS <- ts(fitted(forecast_raw_Garch), start = c(2018, 1), frequency = 48)
```

```
forecast_raw_Garch <- ugarchforecast(GARCH_model_Sarimax, n.ahead = 17520)
forecast_GARCH_SARIMAX <- ts(fitted(forecast_raw_Garch), start = c(2018, 1), frequency = 48)
```

TBATS

```
forecast_TBATS2 <- forecast(TBATS_model_2, h = 17520)
```

Loading Original Models Forecasts

```
forecast_ARIMA <- readRDS("Saved_Forecasts/forecast_ARIMA.rds")
forecast_SARIMA <- readRDS("Saved_Forecasts/forecast_SARIMA.rds")
forecast_Stl_ARIMA <- readRDS("Saved_Forecasts/forecast_STL.rds")
forecast_SARIMAX <- readRDS("Saved_Forecasts/forecast_SARIMAX.rds")
forecast_GARCH <- readRDS("Saved_Forecasts/forecast_GARCH.rds")
forecast_TBATS <- readRDS("Saved_Forecasts/forecast_TBATS.rds")
```

Model accuracy:

```
ARIMA_acc <- accuracy(forecast_ARIMA, Val_time_series)
ARIMA2_acc <- accuracy(forecast_ARIMA2, Val_time_series)
ARIMA3_acc <- accuracy(forecast_ARIMA3, Val_time_series)
ARIMA4_acc <- accuracy(forecast_ARIMA4, Val_time_series)
ARIMA5_acc <- accuracy(forecast_ARIMA5, Val_time_series)
ARIMA6_acc <- accuracy(forecast_ARIMA6, Val_time_series)
ARIMA7_acc <- accuracy(forecast_ARIMA7, Val_time_series)
ARIMA8_acc <- accuracy(forecast_ARIMA8, Val_time_series)
```

```
SARIMA_acc <- accuracy(forecast_SARIMA, Val_time_series)
SARIMA2_acc <- accuracy(forecast_SARIMA2, Val_time_series)
SARIMA3_acc <- accuracy(forecast_SARIMA3, Val_time_series)
SARIMA4_acc <- accuracy(forecast_SARIMA4, Val_time_series)
SARIMA5_acc <- accuracy(forecast_SARIMA5, Val_time_series)
SARIMA6_acc <- accuracy(forecast_SARIMA6, Val_time_series)
SARIMA7_acc <- accuracy(forecast_SARIMA7, Val_time_series)
```

```
SARIMA8_acc <- accuracy(forecast_SARIMA8, Val_time_series)
SARIMA9_acc <- accuracy(forecast_SARIMA9, Val_time_series)
SARIMA10_acc <- accuracy(forecast_SARIMA10, Val_time_series)
```

```
STL_ARIMA_acc <- accuracy(forecast_Stl_ARIMA, Val_time_series)
STL_ARIMA2_acc <- accuracy(forecast_STL2, Val_time_series)
STL_ARIMA3_acc <- accuracy(forecast_STL3, Val_time_series)
STL_ARIMA4_acc <- accuracy(forecast_STL4, Val_time_series)
STL_ARIMA5_acc <- accuracy(forecast_STL5, Val_time_series)
STL_ARIMA6_acc <- accuracy(forecast_STL6, Val_time_series)
STL_ARIMA7_acc <- accuracy(forecast_STL7, Val_time_series)
STL_ARIMA8_acc <- accuracy(forecast_STL8, Val_time_series)
```

```
SARIMAX_acc <- accuracy(forecast_SARIMAX, Val_time_series)
SARIMAX2_acc <- accuracy(forecast_SARIMAX2, Val_time_series)
SARIMAX3_acc <- accuracy(forecast_SARIMAX3, Val_time_series)
SARIMAX4_acc <- accuracy(forecast_SARIMAX4, Val_time_series)
SARIMAX5_acc <- accuracy(forecast_SARIMAX5, Val_time_series)
SARIMAX6_acc <- accuracy(forecast_SARIMAX6, Val_time_series)
SARIMAX7_acc <- accuracy(forecast_SARIMAX7, Val_time_series)
SARIMAX8_acc <- accuracy(forecast_SARIMAX8, Val_time_series)
SARIMAX9_acc <- accuracy(Forecast_SARIMAX_9, Val_time_series)
SARIMAX10_acc <- accuracy(Forecast_SARIMAX_10, Val_time_series)
SARIMAX11_acc <- accuracy(Forecast_SARIMAX_11, Val_time_series)
SARIMAX12_acc <- accuracy(Forecast_SARIMAX_12, Val_time_series)
SARIMAX13_acc <- accuracy(Forecast_SARIMAX_13, Val_time_series)
SARIMAX14_acc <- accuracy(Forecast_SARIMAX_14, Val_time_series)
```

```
Garch_acc <- accuracy(forecast_GARCH, Val_time_series)
Garch_acc_sarima <- accuracy(forecast_GARCH_SARIMA, Val_time_series)
Garch_acc_sarima2 <- accuracy(forecast_GARCH_SARIMA2, Val_time_series)
Garch_acc_sarima3 <- accuracy(forecast_GARCH_SARIMA3, Val_time_series)
Garch_acc_sarima4 <- accuracy(forecast_GARCH_SARIMA4, Val_time_series)
Garch_acc_TBATS <- accuracy(forecast_GARCH_TBATS, Val_time_series)
Garch_acc_sarimax <- accuracy(forecast_GARCH_SARIMAX, Val_time_series)
```

```
TBATS_acc <- accuracy(forecast_TBATS, Val_time_series)
TBATS_acc2 <- accuracy(forecast_TBATS2, Val_time_series)
```

```
acc_list_A <- list(ARIMA  = ARIMA_acc, ARIMA2 = ARIMA2_acc, ARIMA3 = ARIMA3_acc, ARIMA4 = ARIMA4_acc,
                   ARIMA5 = ARIMA5_acc, ARIMA6 = ARIMA6_acc, ARIMA7 = ARIMA7_acc, ARIMA8 = ARIMA8_acc)

acc_list_S <- list(SARIMA  = SARIMA_acc, SARIMA2 = SARIMA2_acc, SARIMA3 = SARIMA3_acc, SARIMA4 = SARIMA4
                   SARIMA5 = SARIMA5_acc, SARIMA6 = SARIMA6_acc, SARIMA7 = SARIMA7_acc, SARIMA8 = SARIMA8
                   SARIMA9 = SARIMA9_acc, SARIMA10 = SARIMA10_acc)

acc_list_STL <- list(STL_ARIMA  = STL_ARIMA_acc, STL_ARIMA2 = STL_ARIMA2_acc, STL_ARIMA3 = STL_ARIMA3_a
                     STL_ARIMA4 = STL_ARIMA4_acc, STL_ARIMA5 = STL_ARIMA5_acc, STL_ARIMA6 = STL_ARIMA6_ac
                     STL_ARIMA7 = STL_ARIMA7_acc, STL_ARIMA8 = STL_ARIMA8_acc)

acc_list_X <- list(SARIMAX  = SARIMAX_acc, SARIMAX2 = SARIMAX2_acc, SARIMAX3 = SARIMAX3_acc, SARIMAX4 =
                   SARIMAX5 = SARIMAX5_acc, SARIMAX6 = SARIMAX6_acc, SARIMAX7 = SARIMAX7_acc, SARIMAX8 =
```

```
                SARIMAX9 = SARIMAX9_acc, SARIMAX10 = SARIMAX10_acc, SARIMAX11 = SARIMAX11_acc,
                SARIMAX12 = SARIMAX12_acc, SARIMAX13 = SARIMAX13_acc)

acc_list_G <- list("SARIMA-fiGARCH" = Garch_acc, "SARIMA-fiGARCH 1" = Garch_acc_sarima,
                   "SARIMA-fiGARCH 2" = Garch_acc_sarima2, "SARIMA-fiGARCH 3" = Garch_acc_sarima3,
                   "SARIMA-fiGARCH 4" = Garch_acc_sarima4, "TBATS-fiGARCH" = Garch_acc_TBATS,
                   "SARIMAX-fiGARCH" = Garch_acc_sarimax
                   )

acc_list_T <- list(TBATS = TBATS_acc, TBATS2 = TBATS_acc2)
```

ARIMA

```
acc_df <- map_df(acc_list_A, ~ as.data.frame(.x)["Test set", ], .id = "Model")
acc_df %>% arrange(MASE) %>% mutate(across(where(is.numeric), ~ round(.x, digits = 2)))
```

```
##               Model    ME   RMSE   MAE      MPE    MAPE MASE ACF1 Theil's U
## Test set...1 ARIMA6  4.25  18.39 13.12     0.61   15.29 0.67 0.65      1.38
## Test set...2 ARIMA7  4.24  18.39 13.12     0.60   15.29 0.67 0.65      1.38
## Test set...3  ARIMA  4.25  18.39 13.12     0.61   15.29 0.67 0.65      1.38
## Test set...4 ARIMA8  4.33  18.41 13.13     0.70   15.28 0.67 0.65      1.37
## Test set...5 ARIMA2 16.20 101.89 53.38 -2136.53 2163.73   NA 0.91     12.57
## Test set...6 ARIMA3 16.12 101.87 53.40 -2138.74 2165.90   NA 0.91     12.58
## Test set...7 ARIMA4 16.12 101.87 53.40 -2138.52 2165.69   NA 0.91     12.58
## Test set...8 ARIMA5 16.13 101.87 53.40 -2138.27 2165.44   NA 0.91     12.58
```

The original model was outperformed by ARIMA(411) and ARIMA(312).

SARIMA

```
acc_df_S <- map_df(acc_list_S, ~ as.data.frame(.x)["Test set", ], .id = "Model")
acc_df_S %>% arrange(MASE) %>% mutate(across(where(is.numeric), ~ round(.x, digits = 2)))
```

```
##                  Model    ME   RMSE   MAE      MPE    MAPE   MASE ACF1
## Test set...1    SARIMA  6.67  19.18 13.62     3.44   15.51   0.70 0.65
## Test set...2  SARIMA10 82.00  83.93 82.00    92.76   92.76 164.82 0.65
## Test set...3   SARIMA8 82.01  83.94 82.01    92.77   92.77 164.84 0.65
## Test set...4   SARIMA9 82.01  83.94 82.01    92.77   92.77 164.84 0.65
## Test set...5   SARIMA7 82.01  83.94 82.01    92.77   92.77 164.85 0.65
## Test set...6   SARIMA2 17.04 102.02 53.15 -2114.40 2142.05     NA 0.91
## Test set...7   SARIMA3 16.89 102.00 53.19 -2118.25 2145.82     NA 0.91
## Test set...8   SARIMA4 16.95 102.01 53.17 -2116.61 2144.21     NA 0.91
## Test set...9   SARIMA5 16.94 102.01 53.17 -2117.04 2144.64     NA 0.91
## Test set...10  SARIMA6 17.29 102.06 53.08 -2107.70 2135.49     NA 0.91
##               Theil's U
## Test set...1       1.38
## Test set...2       5.99
## Test set...3       5.99
```

```
## Test set...4       5.99
## Test set...5       5.99
## Test set...6      12.44
## Test set...7      12.46
## Test set...8      12.45
## Test set...9      12.46
## Test set...10     12.40
```

The original SARIMA model performed best across all accuracy metrics.

STL-ARIMA

```
acc_df_STL <- map_df(acc_list_STL, ~ as.data.frame(.x)["Test set", ], .id = "Model")
acc_df_STL %>% arrange(MASE) %>% mutate(across(where(is.numeric), ~ round(.x, digits = 2)))
```

```
##                   Model    ME   RMSE   MAE      MPE    MAPE MASE ACF1 Theil's U
## Test set...1 STL_ARIMA7  4.77  19.75 14.85     0.96   17.52 0.76 0.69      1.52
## Test set...2 STL_ARIMA8  4.68  19.76 14.89     0.86   17.59 0.76 0.69      1.52
## Test set...3  STL_ARIMA  4.65  19.77 14.90     0.82   17.62 0.76 0.69      1.53
## Test set...4 STL_ARIMA6  4.64  19.78 14.94     0.81   17.67 0.77 0.69      1.53
## Test set...5 STL_ARIMA2 17.00 101.43 52.74 -2063.15 2090.68   NA 0.91     12.07
## Test set...6 STL_ARIMA3 16.82 101.40 52.79 -2068.01 2095.45   NA 0.91     12.10
## Test set...7 STL_ARIMA4 17.06 101.44 52.73 -2061.63 2089.20   NA 0.91     12.06
## Test set...8 STL_ARIMA5 16.84 101.40 52.79 -2067.35 2094.81   NA 0.91     12.09
```

The original model was outperformed by the STL-ARIMA(5,1,2) model (which had its MA term increased by one).

SARIMAX

```
acc_df_X <- map_df(acc_list_X, ~ as.data.frame(.x)["Test set", ], .id = "Model")
acc_df_X %>% arrange(RMSE) %>% mutate(across(where(is.numeric), ~ round(.x, digits = 2)))
```

```
##                   Model    ME   RMSE   MAE      MPE    MAPE MASE ACF1 Theil's U
## Test set...1    SARIMAX  7.96  20.78 15.85     4.93   18.09 0.81 0.68      1.50
## Test set...2   SARIMAX6  8.11  20.83 15.87     5.11   18.09 0.81 0.68      1.50
## Test set...3   SARIMAX8  8.24  20.86 15.88     5.26   18.07 0.81 0.67      1.50
## Test set...4   SARIMAX7  8.26  20.87 15.88     5.28   18.07 0.81 0.67      1.50
## Test set...5  SARIMAX12 -3.42  97.12 59.13 -2884.63 2903.77   NA 0.90     16.95
## Test set...6   SARIMAX5 14.46  98.63 51.19 -2251.19 2276.35   NA 0.90     13.21
## Test set...7   SARIMAX2 13.99  98.68 51.55 -2260.97 2286.05   NA 0.90     13.26
## Test set...8   SARIMAX4 14.23  98.93 51.65 -2237.04 2262.31   NA 0.90     13.11
## Test set...9   SARIMAX3 15.13  99.07 51.25 -2213.46 2239.04   NA 0.90     12.98
## Test set...10 SARIMAX11 14.38  99.73 52.01 -2209.02 2234.52   NA 0.90     12.96
## Test set...11 SARIMAX13 19.04 101.65 52.16 -2006.11 2034.96   NA 0.91     11.75
## Test set...12  SARIMAX9 19.19 101.68 52.15 -1999.42 2028.40   NA 0.91     11.71
## Test set...13 SARIMAX10 17.31 102.04 53.04 -2108.94 2136.72   NA 0.91     12.41
```

The original model performed best in regards to its ME, RMSE, MAE and MPE. I had slightly worse performance on its MAPE, and ACF1 (which means it captured 0.01 less of the autocorrelation in the residuals compared to SARIMAX 8 and 7). There was no best model for the Theil's U or MASE accuracy metrics.

I can see that across all models performances on the validation set, the models that were trained on BoxCox transformed time series performed worse than models that were trained on the original time series.

Unfortunately the models with the new exogenous variables performed much worse overall than the other SARIMAX models.

SARIMA-fiGARCH

```
acc_df_G <- map_df(acc_list_G, ~ as.data.frame(.x)["Test set", ], .id = "Model")
acc_df_G %>% arrange(MAPE) %>% mutate(across(where(is.numeric), ~ round(.x, digits = 2)))
```

```
##                         Model     ME   RMSE    MAE    MPE   MAPE ACF1 Theil's U
## Test set...1 SARIMA-fiGARCH 2 100.51 142.20 100.51  99.34  99.39 0.91      1.00
## Test set...2    TBATS-fiGARCH 100.51 142.20 100.51  99.40  99.43 0.91      1.00
## Test set...3   SARIMA-fiGARCH 100.45 142.24 100.45  98.15 100.20 0.91      1.00
## Test set...4  SARIMAX-fiGARCH 100.72 142.35 100.72 104.84 104.84 0.91      1.01
## Test set...5 SARIMA-fiGARCH 1 100.75 142.37 100.75 105.65 105.65 0.91      1.01
## Test set...6 SARIMA-fiGARCH 4 100.76 142.38 100.76 106.01 106.01 0.91      1.01
## Test set...7 SARIMA-fiGARCH 3 101.09 142.61 101.09 114.46 114.46 0.91      1.02
```

The original model outperformed the other models in terms of its ME, MAE, and MPE.

The fiGARCH model trained on the TBATS residuals had a lower RMSE and MAPE compared to the original model.

All of the models had the same ACF1

The original model and the fiGARCH model trained on the TBATS residuals both had a Theil's U of 1, which means they had the same performance as a naive forecast. Since both models are no better than a naive forecast I have decided not to move forward with this type of model.

TBATS

```
acc_df_T <- map_df(acc_list_T, ~ as.data.frame(.x)["Test set", ], .id = "Model")
acc_df_T %>% arrange(RMSE) %>% mutate(across(where(is.numeric), ~ round(.x, digits = 2)))
```

```
##               Model   ME  RMSE   MAE  MPE  MAPE MASE ACF1 Theil's U
## Test set...1 TBATS2 7.08 21.09 16.27 3.74 18.98 0.97  0.7      1.61
## Test set...2  TBATS 7.92 21.35 16.58 4.75 19.14 0.85  0.7      1.60
```

The second TBATS model (TBATS2) trained on the 2017 time series with outliers removed has a better ME, RMSE, MAE, MPE and MAPE than the original TBATS model. It has lower absolute and percentage errors when compared to the original TBATS model, which indicates that model TBATS2 has stronger predictive power.

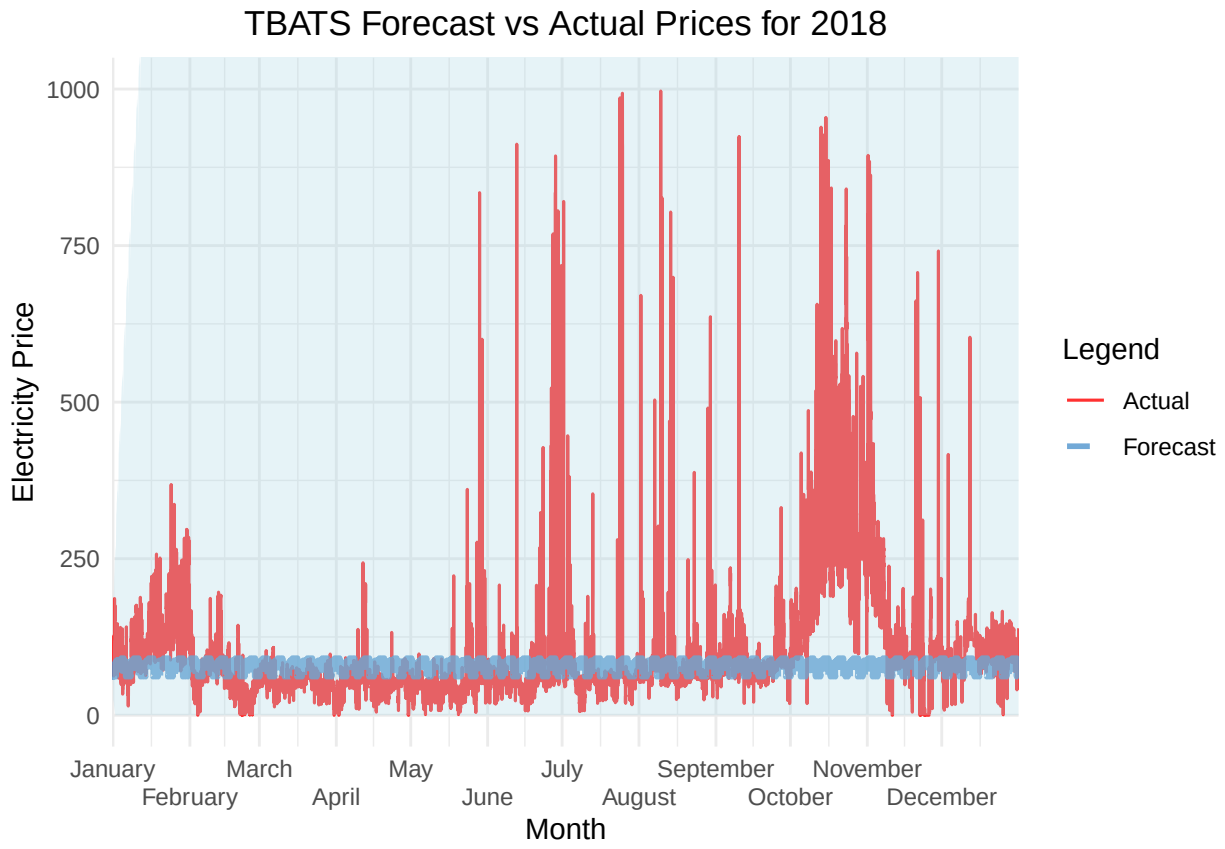
The MASE however is higher for the second TBATS than the first TBATS. This could be because the models use different seasonal periods {<48,8>} vs {<48,7>}, and therefore the naive benchmark that each model is being compared to with MASE is slightly different. Because the models use different seasonal periods, and there for different naive benchmarks, the MASE for the models cannot be directly compared.

Plots

```
plot_actual_forecast <- function(start_time, actual_vals, start_year, forecasts, model_name) {
start_time     <- as.POSIXct(start_time, tz = "UTC")
times_forecast <- seq(start_time, by = "30 min", length.out = 17520)

Test_time_series2 <- ts(actual_vals$DollarsPerMegawattHour, frequency = 48, start = c(start_year, 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(forecasts$mean),
  Lower = as.numeric(forecasts$lower[,2]),  # 95% lower bound
  Upper = as.numeric(forecasts$upper[,2]),  # 95% upper bound
  Type  = "Forecast"
)

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, linetype = "dashed") +
  geom_ribbon(data = df_forecast, aes(x = Date, ymin = Lower, ymax = Upper),
              fill = "lightblue", alpha = 0.3) +
  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(n.dodge = 2)) +
  labs(title = paste(model_name, " Forecast vs Actual Prices for ", start_year, sep = ""),
       x = "Month", y = "Electricity Price", color = "Legend") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5)) +
  coord_cartesian(ylim = c(0, 1000))
}
```

```
plot_actual_forecast("2018-01-01 00:00:00", Val_set, 2018, forecast_TBATS2, "TBATS")
```

```
## 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.
```



References and Citations

Electricity Authority. (n.d.). Final energy prices by month [Dataset]. EMI - Electricity Market Information. Retrieved between July 11 and July 15, 2025, from https://www.emi.ea.govt.nz/Wholesale/Datasets/DispatchAndPricing/FinalEnergyPrices/ByMonth

Electricity Authority. (n.d.). Generation output by plant [Dataset]. EMI - Electricity Market Information. Retrieved between July 18 and July 20, 2025, from https://www.emi.ea.govt.nz/Wholesale/Datasets/Generation/Generation_MD

Electricity Authority. (n.d.). Bids [Datasets]. EMI - Electricity Market Information. Retrieved between July 24 and July 27, 2025, from www.emi.ea.govt.nz/Wholesale/Datasets/BidsAndOffers/Bids

Ministry for the Environment. (15 Oct 2020). Daily temperature, 1909 - 2019 [Dataset]. Ministry for the Environment. Retrieved on August 16th, from https://data.mfe.govt.nz/table/105056-daily-temperature-1909-2019/

Ministry of Business, Inovation and Employment. (n.d.). weekly-table [Dataset]. Ministry for the Environment. Retrieved on August 16th, from https://www.mbie.govt.nz/building-and-energy/energy-and-natural-resources/energy-statistics-and-modelling/energy-statistics/weekly-fuel-price-monitoring

Consumer NZ. (31 Jan 2019). Fixing the New Zealand Building Code. Consumer NZ. https://www.consumer.org.nz/articles/fixing-the-new-zealand-building-code

...