DATA 624 - Project 1: Power Forecast

Richie Rivera

From the assignment description:

> Part B consists of a simple dataset of residential power usage for January 1998 until December 2013. Your assignment is to model these data and a monthly forecast for 2014. The data is given in a single file. The variable 'KWH' is power consumption in Kilowatt hours, the rest is straight forward.

1. Loading the data

```
# Loading packages
library(readxl)
library(fpp3)
library(dplyr)
library(tsibble)
library(zoo)
library(readr)
```

```
# Specify the working directory
setwd("F:/git/cuny-msds/data624-predictive-analytics/projects/project-1")

power <- read_excel("data/ResidentialCustomerForecastLoad-624.xlsx")

# rename columns
colnames(power) <- c(
  "casesequence",
  "month",
  "kwh"
)

#rename columns
glimpse(power)
```

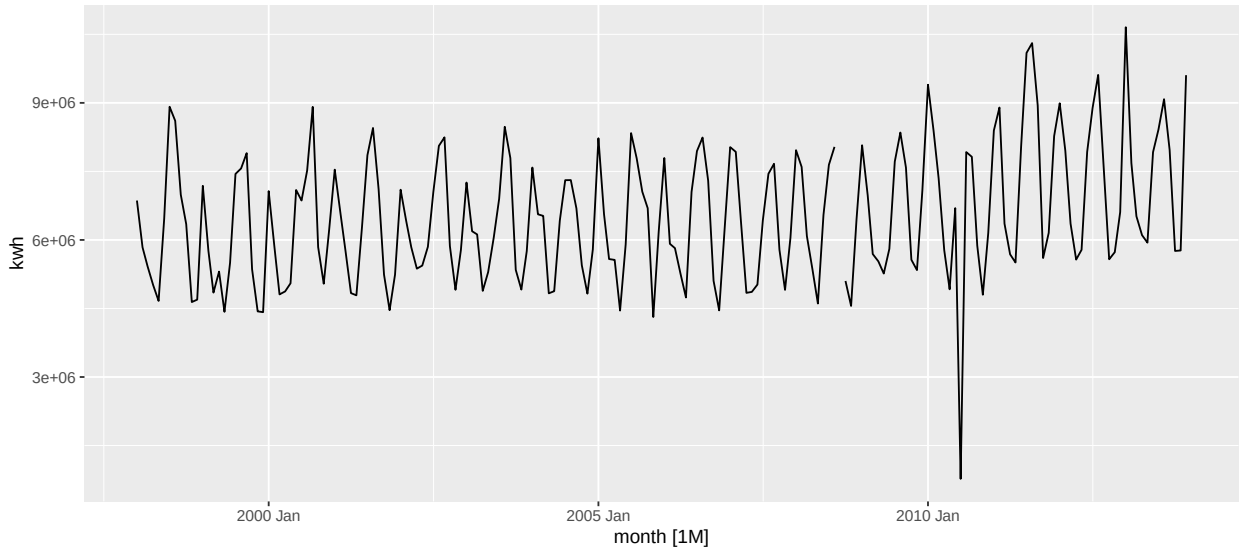
```
## Rows: 192
## Columns: 3
## $ casesequence <dbl> 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 74~
## $ month        <chr> "1998-Jan", "1998-Feb", "1998-Mar", "1998-Apr", "1998-May~
## $ kwh          <dbl> 6862583, 5838198, 5420658, 5010364, 4665377, 6467147, 891~
```

2. Investigating the Power Data

Looking at the values in the data, it looks like the column `month` represents the month but it is currently represented as a string and will need to represent a date.

```
power_ts <- power |>
  mutate(month = yearmonth(month)) |>
  as_tsibble(
    index = month
  ) |>
  select(-casesequence)

power_ts |>
  autoplot()
```

```
## Plot variable not specified, automatically selected `.vars = kwh`
```




```
power_ts |>
  filter(is.na(kwh))
```

```
## # A tsibble: 1 x 2 [1M]
##      month   kwh
##      <mth> <dbl>
## 1 2008 Sep    NA
```

```
power_ts |>
  arrange(kwh)
```

```
## Warning: Current temporal ordering may yield unexpected results.
## i Suggest to sort by ``, `month` first.
```

```
## # A tsibble: 192 x 2 [1M]
```

```
##       month     kwh
##       <mth>   <dbl>
##  1 2010 Jul  770523
##  2 2005 Nov 4313019
##  3 1999 Dec 4419229
##  4 1999 May 4426794
##  5 1999 Nov 4436269
##  6 2005 May 4453983
##  7 2006 Nov 4458429
##  8 2001 Nov 4461979
##  9 2008 Nov 4555602
## 10 2008 May 4608528
## # i 182 more rows
```

Fixing the date allowed us to plot this timeseries and we can see that it is highly seasonal with not much of a trend. There also seems to be one outlier in July 2010 where the power has dropped off significantly. Additionally, there's a gap in September 2010, which we will need to fill.

3. Data Pre-Processing

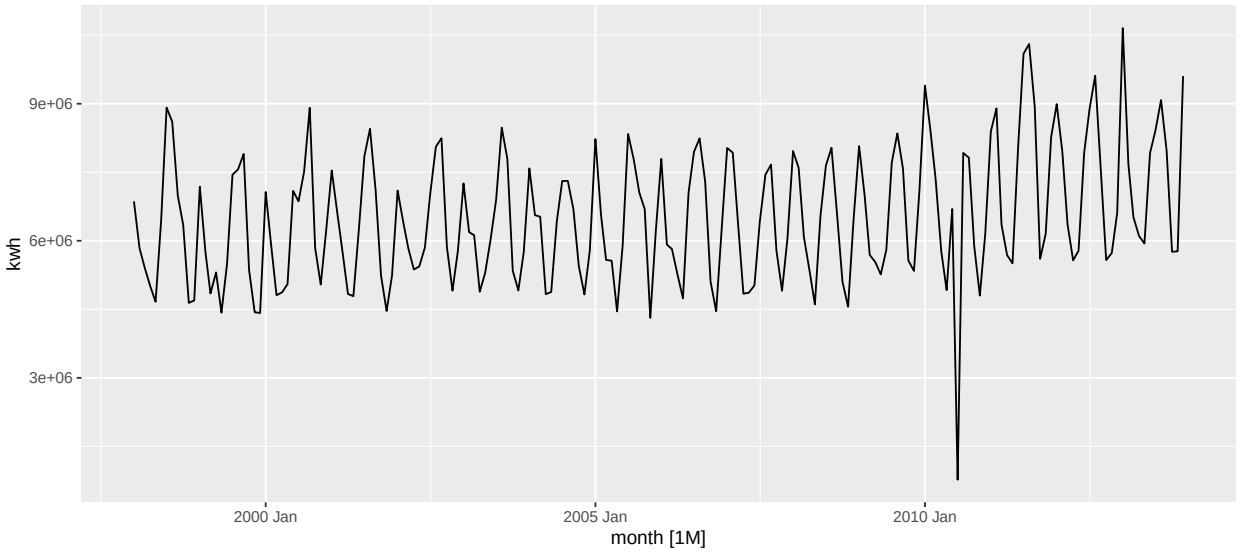
We found two items that will need to address to fix this time series.

1. We must fill the gap that exists in September 2008
2. We must decide how we wish to handle the outlier in July 2010

For the gap, we can perform a simple linear interpolation. This seems valid as it seems that the power was going to begin the descending part of the cycle.

```
power_ts <- power_ts |>
  mutate(
    kwh = na.approx(kwh, na.rm = FALSE)
  )

power_ts |>
  autoplot()
```

```
## Plot variable not specified, automatically selected `.vars = kwh`
```




```
power_ts |>
  filter(is.na(kwh))
```

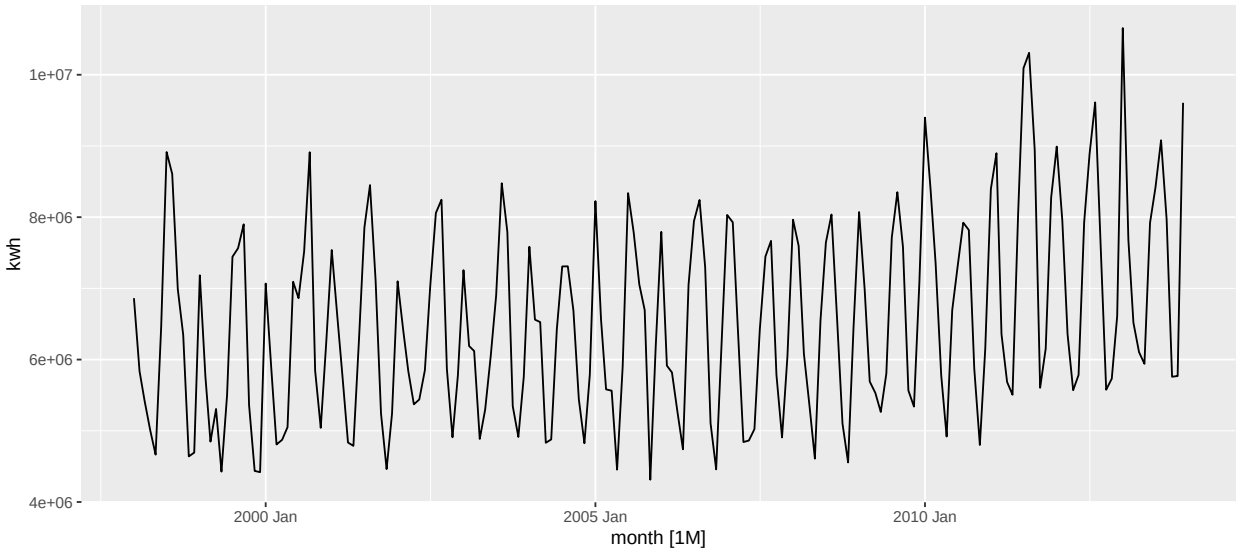
```
## # A tsibble: 0 x 2 [?]
## # i 2 variables: month <mth>, kwh <dbl>
```

We can now see that we've filled the gap with a value between August 2008 and October 2008.

Judging from the trends, it looks like the outlier is potentially a data collection issue. To fix this, I think it'd be worthwhile to also perform a linear interpolation as we can interpret this as a black swan event and it is not very helpful for our model:

```
power_ts <- power_ts |>
  mutate(
    kwh = if_else(kwh == 770523, NA_real_, kwh)
  ) |>
  mutate(
    kwh = na.approx(kwh, na.rm = FALSE)
  )

power_ts |>
  autoplot()
```

```
## Plot variable not specified, automatically selected `.vars = kwh`
```



This timeseries seems much better. It also seems that there could be a slight upward trend now which was obfuscated by the outlier.

Although the data seems pretty normally distributed right now, we will pull the guerrero lambda to see if there is a transformation we can do which will make this data more normally distributed:

```
kwh_lambda <- power_ts |>
  features(kwh, features = guerrero) |>
  pull(lambda_guerrero)
```

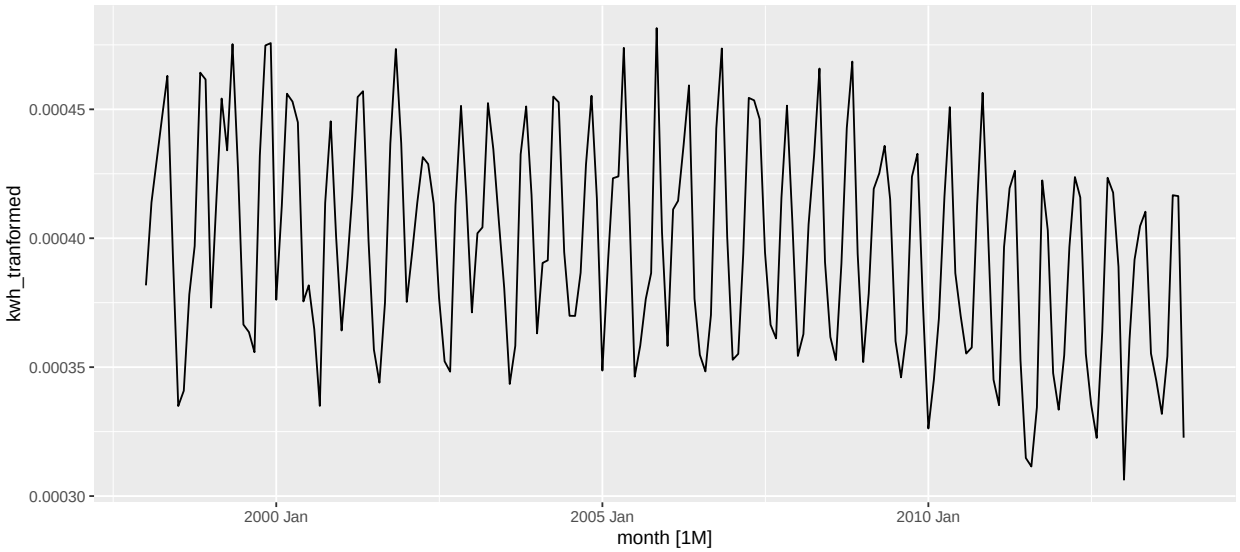
With a λ of -0.21, we can refer to the chart here and see that this transform is most similar to taking the negative half root or:

$$\frac{1}{\sqrt{Y}}$$

This operation will need to be undone at the end so our model outputs are real. This can be undone by using the below equation:

$$\frac{1}{y^2}$$

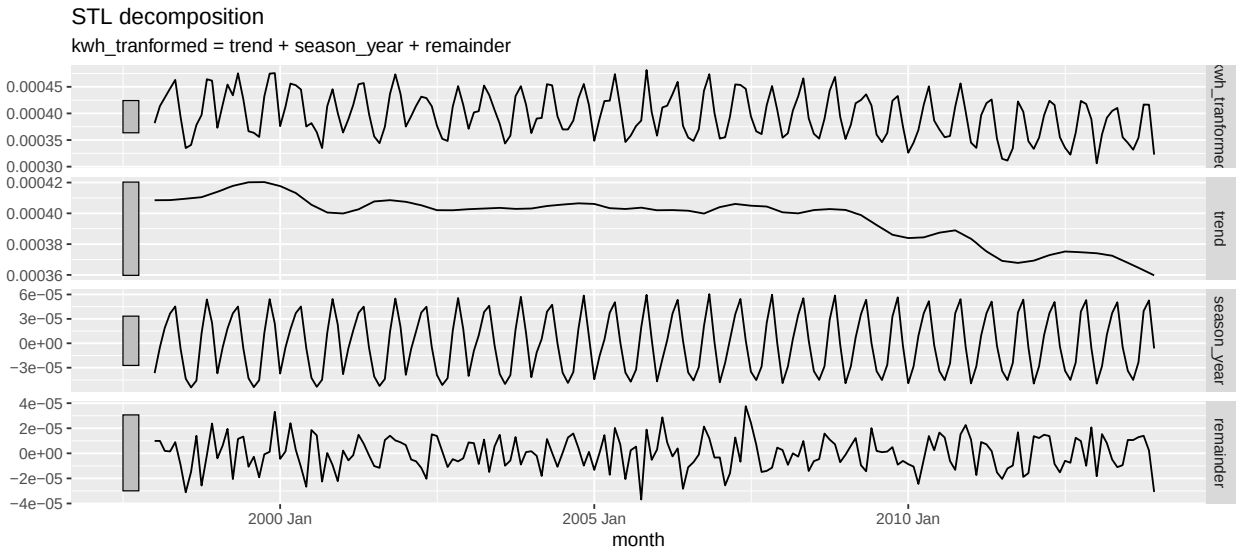
```
power_ts <- power_ts |>
  mutate(
    kwh_tranformed = kwh ^ -0.5
  )

power_ts |>
  autoplot(kwh_tranformed)
```



4 Developing the model

With our data transformed, we can look at a `STL()` decomposition of our data:

```
power_ts |>
  model(
    stl = STL(kwh_tranformed)
  ) |>
  components() |>
  autoplot()
```



From this decomposition we can see that the data is relatively stationary but the most impactful component is the seasonal component.

we have a few applicable univariate models we can apply here:

1. `SNAIVE()` - Because there is a strong seasonal component and not much trend. I doubt this will be the best model as this model will apply the seasonal component to the last observed value.
2. `ETS()` (Holt-Winters Additive Method) - For the same reason as `SNAIVE()`. This model may perform well as the seasonal variation is roughly constant. The multiplicative method is better for seasonal effects which increase over time which we can see from the `STL()` decomposition isn't the case here.
3. `ARIMA()` - With the built in differencing using the KPSS unit root test, an `ARIMA()` model is also applicable here.

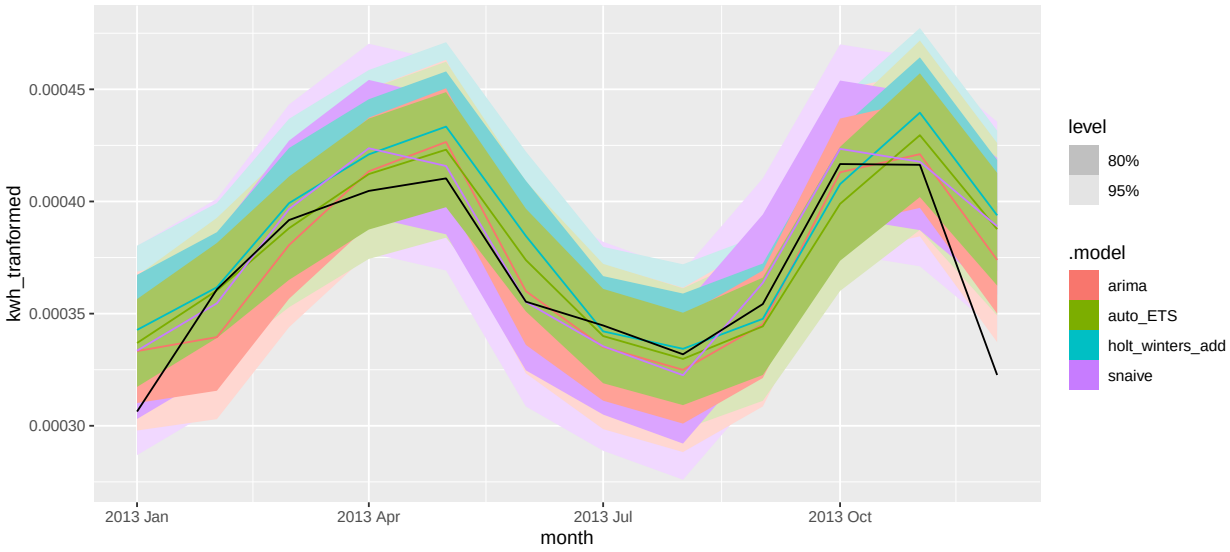
Now we can create our training and testing splits, we will have our testing split be a holdout period of 1 year:

```
# Training subset
power_train <- power_ts |>
  filter(
    month < yearmonth("2013-01-01")
  )

# Testing subset
power_test <- power_ts |>
  filter(
    month >=  yearmonth("2013-01-01")
  )

# Fitting the models
power_fits <- power_train |>
  model(
    snaive = SNAIVE(kwh_tranformed),
    auto_ETS = ETS(kwh_tranformed),
    holt_winters_add = ETS(
      kwh_tranformed ~ error("A") + trend("N") + season("A")
    ),
    arima = ARIMA(kwh_tranformed)
  )
```

```
## Warning in sqrt(diag(best$var.coef)): NaNs produced
```

```
# creating the forecast over the training period
power_fcs <- power_fits |>
  forecast(h = nrow(power_test))

# Charting the forecast
power_fcs |>
  autoplot(power_test) |>
  labs(
    y = "Transformed KWH Usage",
    title = "Forecasts of Tranformed KWH Usage (Jan 2013)"
  )
```

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




```
##
## $y
## [1] "Transformed KWH Usage"
##
## $title
## [1] "Forecasts of Tranformed KWH Usage (Jan 2013)"
##
## attr(,"class")
## [1] "labels"
```

As the chart above shows, we are seeing that each of the models have their predicted values within the confidence band, suggesting that the models fit fairly well to the data. For some more rigorous tests:

```
power_fits |>
  report()
```

```
## Warning in report.mdl_df(power_fits): Model reporting is only supported for
## individual models, so a glance will be shown. To see the report for a specific
## model, use `select()` and `filter()` to identify a single model.
```

```
## # A tibble: 4 x 11
##   .model        sigma2 log_lik    AIC   AICc    BIC       MSE      AMSE      MAE
##   <chr>          <dbl>   <dbl>  <dbl>  <dbl>  <dbl>     <dbl>     <dbl>    <dbl>
## 1 snaive      5.63e-10      NA     NA     NA     NA  NA        NA        NA
## 2 auto_ETS    2.05e- 3   1506. -2981. -2978. -2933.  3.03e-10  3.25e-10  3.40e-2
## 3 holt_winte~ 3.67e-10   1495. -2960. -2958. -2913.  3.38e-10  3.46e-10  1.42e-5
## 4 arima       3.26e-10   1595. -3179. -3179. -3164. NA        NA        NA
## # i 2 more variables: ar_roots <list>, ma_roots <list>
```

```
power_fcs |>
  accuracy(power_test)
```

```
## # A tibble: 4 x 10
```

```
##   .model          .type       ME    RMSE     MAE   MPE  MAPE  MASE RMSSE    ACF1
##   <chr>           <chr>    <dbl>   <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl>
## 1 arima           Test  -4.25e-6 1.95e-5 1.45e-5 -1.35  4.19   NaN   NaN  0.0520
## 2 auto_ETS        Test  -9.09e-6 2.30e-5 1.55e-5 -2.78  4.47   NaN   NaN  0.155
## 3 holt_winters_a~ Test  -1.60e-5 2.70e-5 1.90e-5 -4.63  5.43   NaN   NaN  0.177
## 4 snaive          Test  -9.63e-6 2.22e-5 1.38e-5 -2.84  4.05   NaN   NaN -0.0301
```

With the two tables above, we can note the following:

- The ARIMA model has the lowest AIC, suggesting that it may be the best fit of the models tested
- The `SNAIVE()` model has the lowest MAPE, although the `ARIMA()` model is very close

With those two observations, I believe it'd be best to use the `ARIMA()` model as it has evidence of being the best fit and performs almost as well as all the other models.

```
power_fits |>
  select(.model = "arima") |>
  report()
```

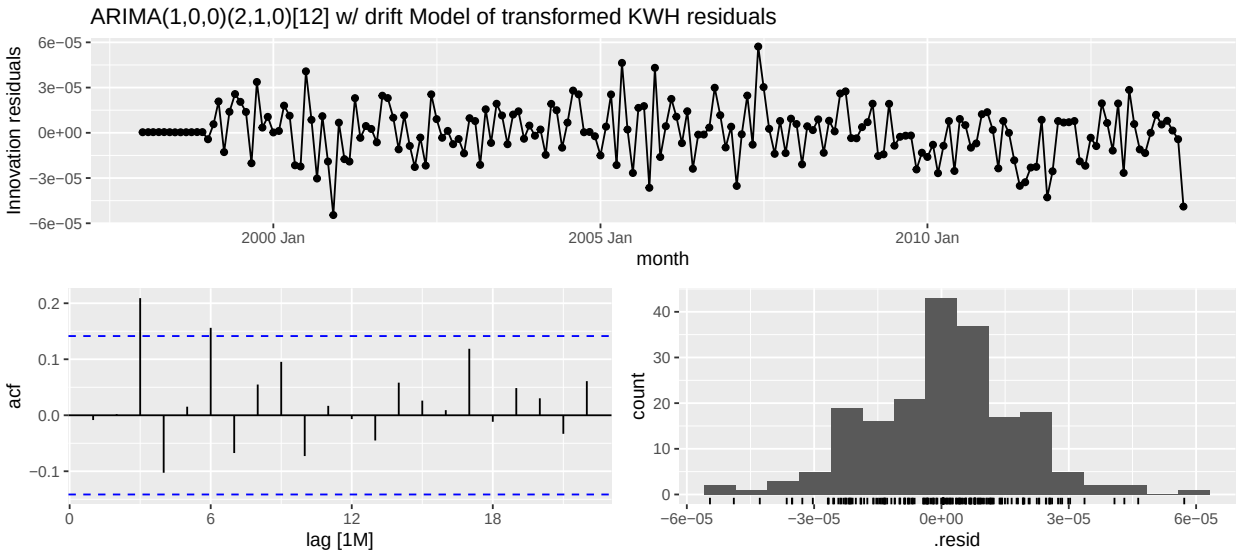
```
## Series: kwh_tranformed
## Model: ARIMA(1,0,0)(2,1,0)[12] w/ drift
##
## Coefficients:
##          ar1     sar1     sar2  constant
##       0.2560  -0.7208  -0.3740         0
## s.e.  0.0719   0.0375   0.0794       NaN
##
## sigma^2 estimated as 3.265e-10:  log likelihood=1594.66
## AIC=-3179.32   AICc=-3178.95   BIC=-3163.7
```

The `ARIMA()` model selected is an $ARIMA(1,0,0)(2,1,0)[12]w/drift$. Now that we have our model selected, we will retrain our model with the entire dataset:

```
power_final_fit <- power_ts |>
  model(
    arima = ARIMA(kwh_tranformed ~ pdq(1, 0, 0) + PDQ(2, 1, 0, period = 12))
  )
```

```
## Warning in sqrt(diag(best$var.coef)): NaNs produced
```

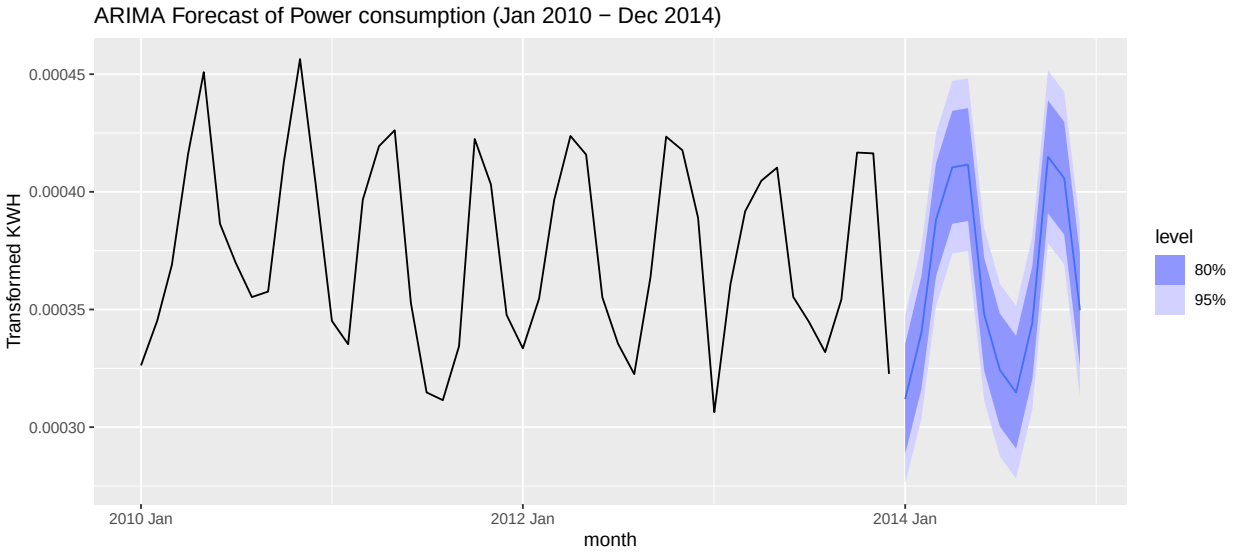
```
power_final_fc <- power_final_fit |>
  forecast(h = 12)

power_final_fit |>
  select(.model = "arima") |>
  gg_tsresiduals() +
  labs(
    title = "ARIMA(1,0,0)(2,1,0)[12] w/ drift Model of transformed KWH residuals"
  )
```



From the residuals plot above, we can see that the residuals seem normally distributed and two of the autocorrelations exceed the critical value. With this, we can proceed with this model and save the results.

The final forecast is shown below:

```
power_final_fc |>
  autoplot(
    power_ts |>
      filter(
        month >= yearmonth("2010-01-01")
      )
  ) +
  labs(
    title = "ARIMA Forecast of Power consumption (Jan 2010 - Dec 2014)",
    y = "Transformed KWH"
  )
```



The above plot shows the transformed KWH. In the supplied excel document, the dependent has been

un-transformed to match the same measurement that was provided.

```
power_final_fc |>
  as_tibble() |>
  filter(.model == "arima") |>
  mutate(
    power_lower_ci_95 = 1 / (hilo(kwh_tranformed)$lower ^ 2),
    power_prediction = 1 / (mean(kwh_tranformed) ^ 2),
    power_upper_ci_95 = 1 / (hilo(kwh_tranformed)$upper ^ 2)
  ) |>
  select(month, power_prediction, power_lower_ci_95, power_upper_ci_95) |>
  write_csv("forecasts/power_forecast_ci_ARIMA.csv")
```

With our output, I would expect that the power consumption to be between the 95% confidence interval provided.