Project 1

Joey Bochnik

2025-03-20

First do some EDA on the data. Read in the file and get a sense for the data.

```
library(fpp3)
```

```
## Warning: package 'fpp3' was built under R version 4.3.3

## Registered S3 method overwritten by 'tsibble':
##   method               from
##   as_tibble.grouped_df dplyr

## -- Attaching packages --------------------------------------- fpp3 1.0.1 --

## v tibble      3.2.1     v tsibble     1.1.5
## v dplyr       1.1.4     v tsibbledata 0.4.1
## v tidyr       1.3.1     v feasts      0.4.1
## v lubridate   1.9.4     v fable       0.4.1
## v ggplot2     3.5.1

## Warning: package 'lubridate' was built under R version 4.3.3

## Warning: package 'tsibble' was built under R version 4.3.3

## Warning: package 'feasts' was built under R version 4.3.3

## Warning: package 'fabletools' was built under R version 4.3.3

## Warning: package 'fable' was built under R version 4.3.3

## -- Conflicts ------------------------------------------------ fpp3_conflicts --
## x lubridate::date()    masks base::date()
## x dplyr::filter()      masks stats::filter()
## x tsibble::intersect() masks base::intersect()
## x tsibble::interval()  masks lubridate::interval()
## x dplyr::lag()         masks stats::lag()
## x tsibble::setdiff()   masks base::setdiff()
## x tsibble::union()     masks base::union()
```

```
library(readxl)
library(openxlsx)
```

```
## Warning: package 'openxlsx' was built under R version 4.3.3
```

```
library(imputeTS)
```

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

First read in the data using the read_excel function.

```
atm_data <- read_excel("ATM624Data.xlsx")
head(atm_data)
```

```
## # A tibble: 6 x 3
##    DATE ATM    Cash
##   <dbl> <chr> <dbl>
## 1 39934 ATM1     96
## 2 39934 ATM2    107
## 3 39935 ATM1     82
## 4 39935 ATM2     89
## 5 39936 ATM1     85
## 6 39936 ATM2     90
```

Alter the date to get it in the correct YYYY-MM-DD format. It is currently in excel serial format.

```
atm_data <- atm_data %>%
  mutate(DATE = as.Date(DATE,origin = "1899-12-30"))
head(atm_data)
```

```
## # A tibble: 6 x 3
##   DATE       ATM    Cash
##   <date>     <chr> <dbl>
## 1 2009-05-01 ATM1     96
## 2 2009-05-01 ATM2    107
## 3 2009-05-02 ATM1     82
## 4 2009-05-02 ATM2     89
## 5 2009-05-03 ATM1     85
## 6 2009-05-03 ATM2     90
```

Convert the data to a tsibble with DATE as the index and ATM as the Key.

```
atm_data <- atm_data %>%
  as_tsibble(
    index = DATE,
    key = ATM
  )
head(atm_data)
```

```
## # A tsibble: 6 x 3 [1D]
## # Key:       ATM [1]
##   DATE       ATM    Cash
##   <date>     <chr> <dbl>
## 1 2009-05-01 ATM1     96
## 2 2009-05-02 ATM1     82
## 3 2009-05-03 ATM1     85
## 4 2009-05-04 ATM1     90
## 5 2009-05-05 ATM1     99
## 6 2009-05-06 ATM1     88
```

Check how many entries we have for each ATM.

```
atm_data %>%
  distinct(ATM)
```

```
## # A tibble: 5 x 1
##   ATM
##   <chr>
## 1 ATM1
## 2 ATM2
## 3 ATM3
## 4 ATM4
## 5 <NA>
```

```
atm_data %>%
  count(ATM)
```

```
## # A tibble: 5 x 2
##   ATM       n
##   <chr> <int>
## 1 ATM1    365
## 2 ATM2    365
## 3 ATM3    365
## 4 ATM4    365
## 5 <NA>     14
```

Check the rows where ATM is NA. We 14 instances where ATM is NA

```
atm_data %>%
  filter(is.na(ATM))
```

```
## # A tsibble: 14 x 3 [1D]
## # Key:       ATM [1]
##    DATE       ATM    Cash
##    <date>     <chr> <dbl>
##  1 2010-05-01 <NA>     NA
##  2 2010-05-02 <NA>     NA
##  3 2010-05-03 <NA>     NA
##  4 2010-05-04 <NA>     NA
##  5 2010-05-05 <NA>     NA
##  6 2010-05-06 <NA>     NA
```

```
##  7 2010-05-07 <NA>     NA
##  8 2010-05-08 <NA>     NA
##  9 2010-05-09 <NA>     NA
## 10 2010-05-10 <NA>     NA
## 11 2010-05-11 <NA>     NA
## 12 2010-05-12 <NA>     NA
## 13 2010-05-13 <NA>     NA
## 14 2010-05-14 <NA>     NA
```

Looking at the NA rows they do not seem to be linked to any of the 4 ATMs in any way there was no Cash and the date sequence is just two weeks to start May so I will remove these rows since I do not think they are important.

```
atm_data <- atm_data %>%
  filter(!is.na(ATM))
atm_data %>%
  is.na() %>%
  colSums()
```

```
## DATE  ATM Cash
##    0    0    5
```

Now all the NA ATM rows are removed.

Now check to see if there are any more NA values in the data.

```
atm_data %>%
  filter(if_any(everything(), is.na))
```

```
## # A tsibble: 5 x 3 [1D]
## # Key:       ATM [2]
##   DATE       ATM    Cash
##   <date>     <chr> <dbl>
## 1 2009-06-13 ATM1     NA
## 2 2009-06-16 ATM1     NA
## 3 2009-06-22 ATM1     NA
## 4 2009-06-18 ATM2     NA
## 5 2009-06-24 ATM2     NA
```

We still have 5 missing values will have to impute. We can see that there are 3 dates for ATM1 and 2 for ATM2 that we will need to impute lets graph the time plot and then decide how to impute for ATM1 and ATM2 Now we can split up the ATM data for each ATM.

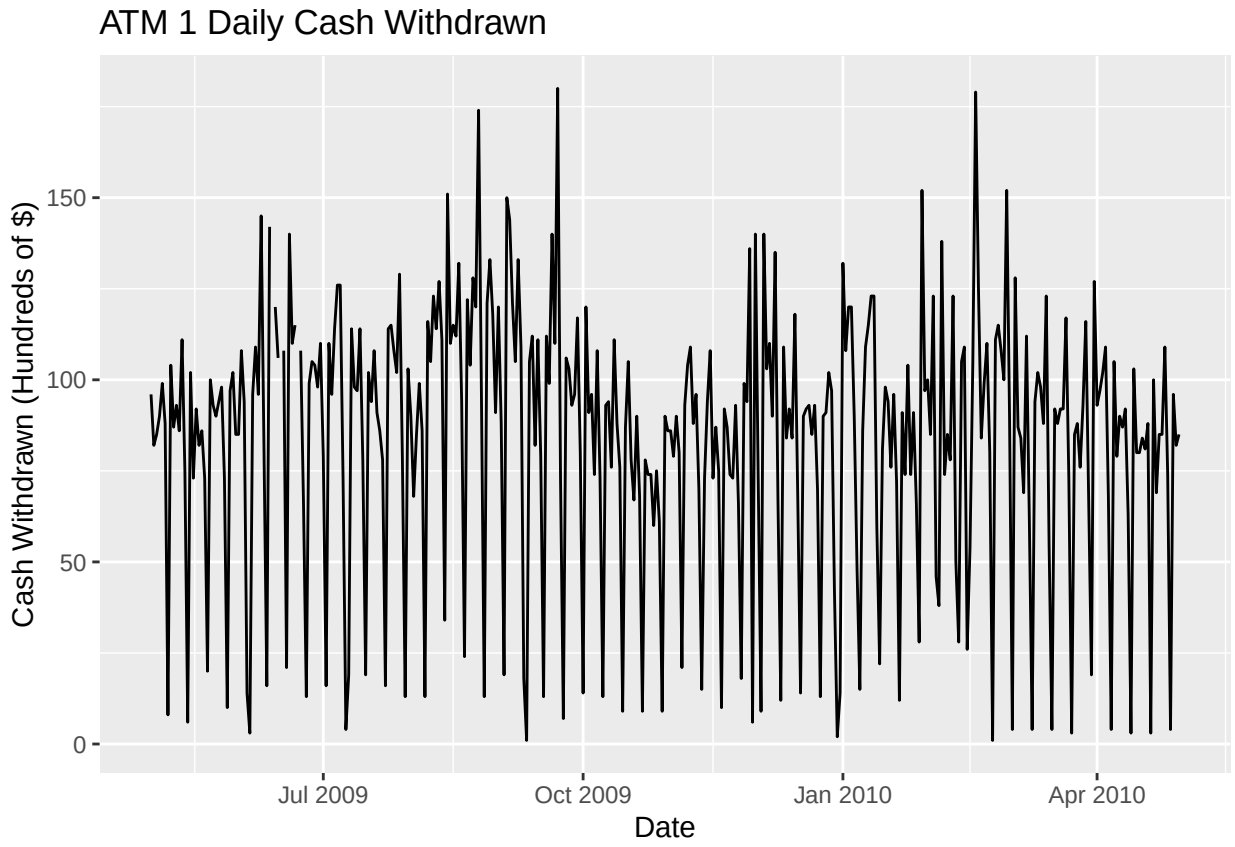
```
atm1 <- atm_data %>%
  filter(ATM == "ATM1")

atm2 <- atm_data %>%
  filter(ATM == "ATM2")

atm3 <- atm_data %>%
  filter(ATM == "ATM3")

atm4 <- atm_data %>%
  filter(ATM == "ATM4")
```

ATM 1

For ATM1 there are 3 missing values so we will need to impute those. First, lets look at a time plot of the data to get a feel for it.

```
atm1 %>%
  autoplot() +
  labs(title = "ATM 1 Daily Cash Withdrawn", x= "Date", y="Cash Withdrawn (Hundreds of $)")
```

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



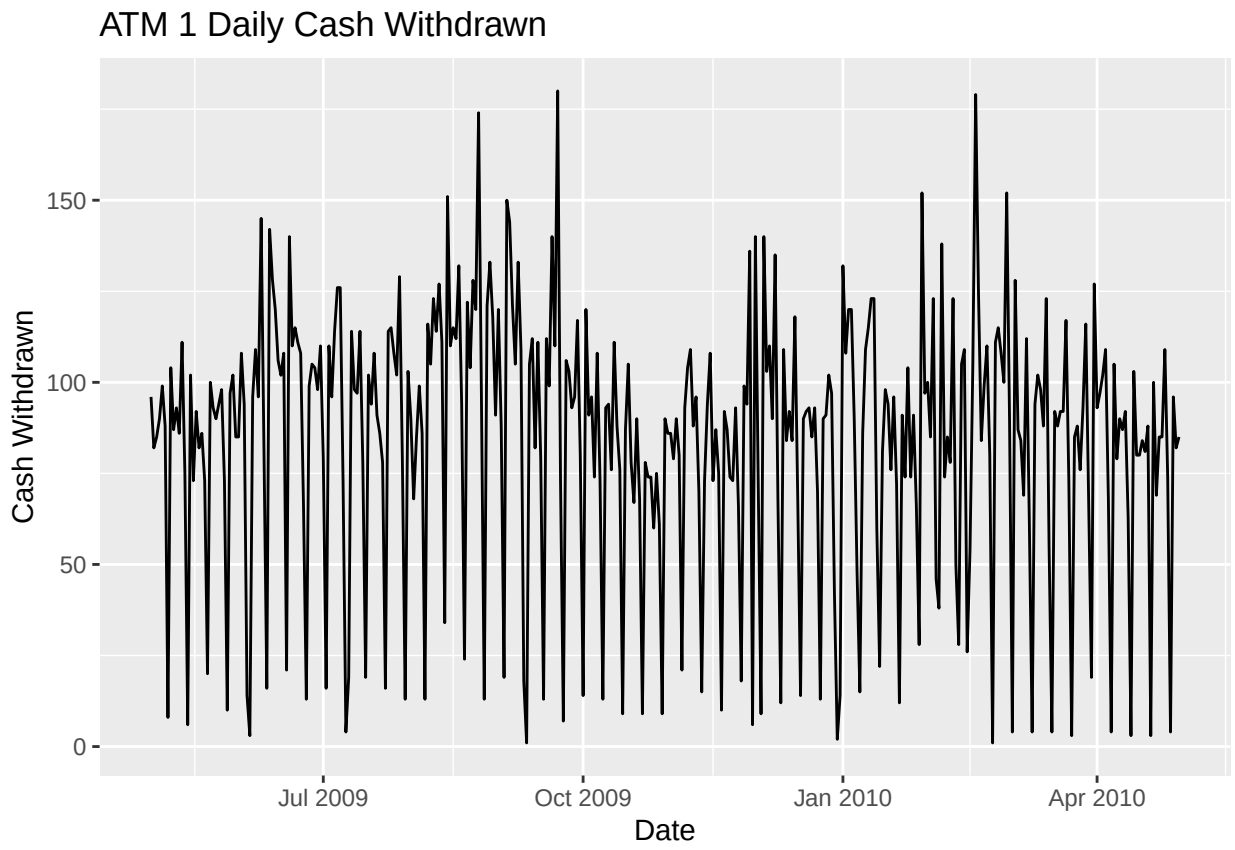
There appears to be some seasonality so lets use this function that I found that works well to interpolate seasonal data. It is from the imputeJS library its called na_seadec() by setting find_frequency to true it will automatically find any seasonality in the data.

```
library(imputeTS)

atm1 <- atm1 %>%
  mutate(Cash = na_seadec(Cash, algorithm = "interpolation", find_frequency= TRUE))
```

```
atm1 %>%
  autoplot() +
  labs(title = "ATM 1 Daily Cash Withdrawn", x= "Date", y="Cash Withdrawn")
```

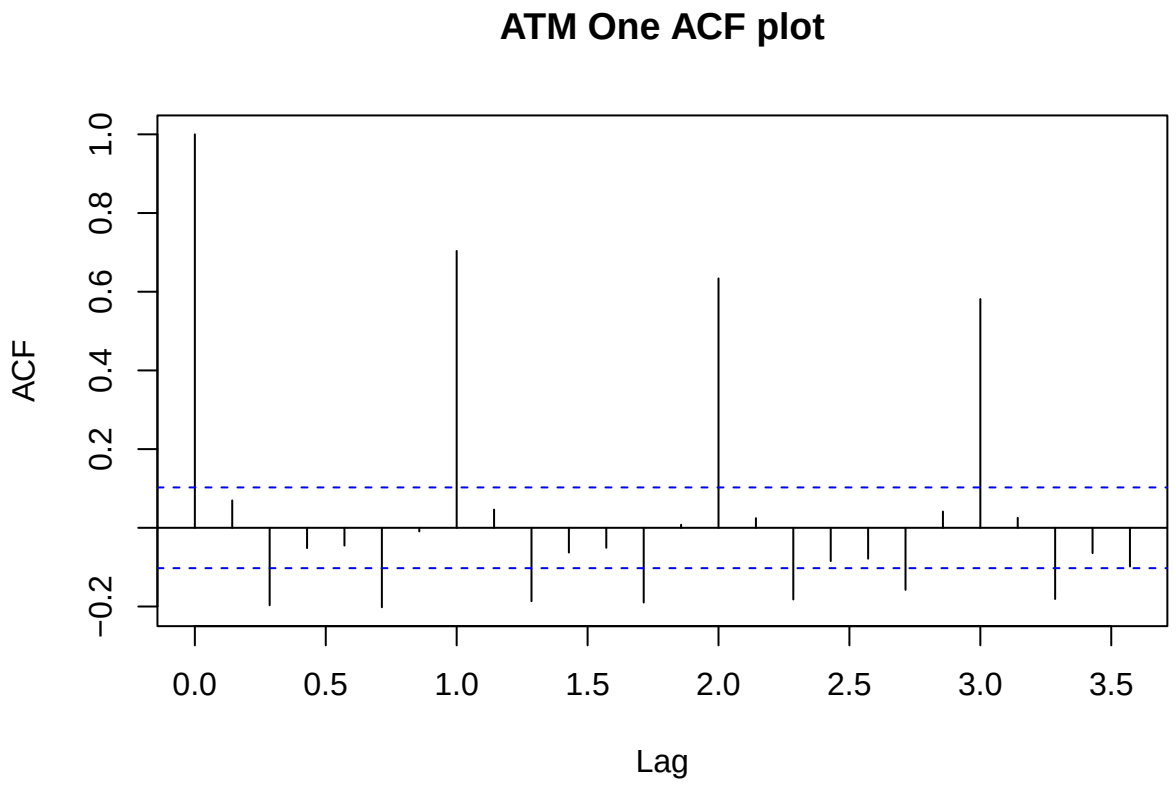
```
## Plot variable not specified, automatically selected `.vars = Cash`
```

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



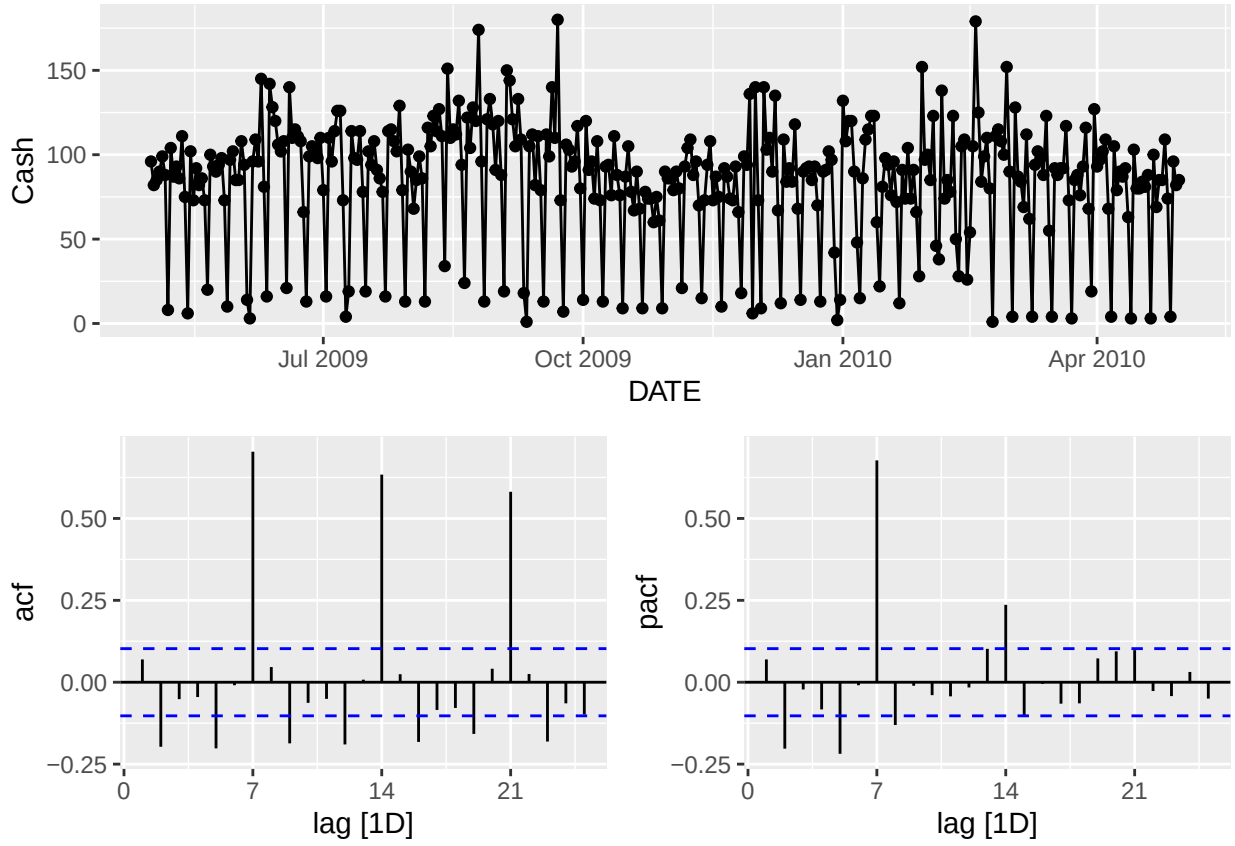
Now lets look at the PCAF and ACF plots to try and determine the seasonality for any models that will be created it appears to be weekly but lets confirm.

```
atm1 %>%
  acf(main = "ATM One ACF plot")
```



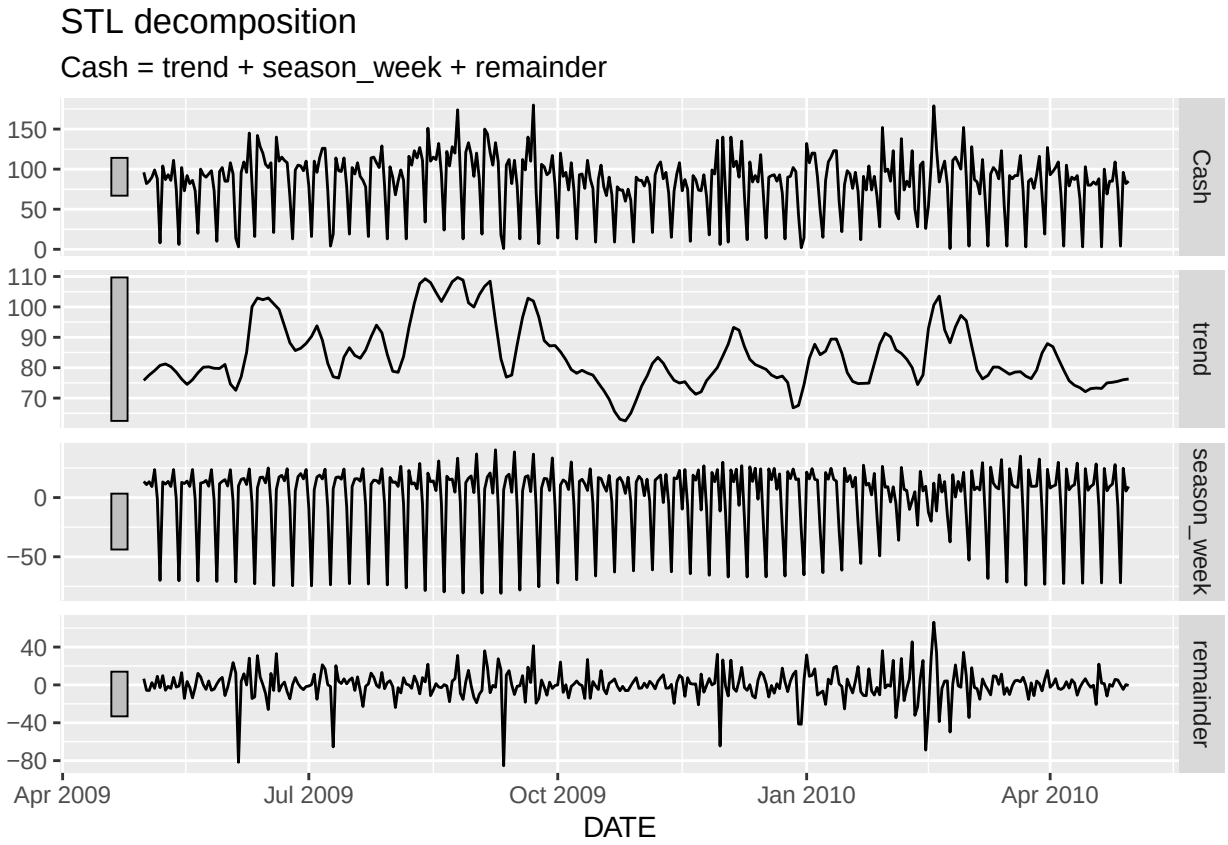

```
atm1 %>%
  gg_tsdisplay(Cash, plot_type = "partial")
```

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



Looks like weekly seasonality lets do STL decomposition to look at it further. We can see the ACF and PACF lags clearly spike every 7 days showing weekly seasonality. This data does not appear stationary and could possibly benefit from first order differencing for an ARIMA model.

```
atm1_stl <- atm1 %>%
  model(STL(Cash))

atm1_stl %>%
  components() %>%
  autoplot()
```



From the STL decomposition above we can see that there is no clear positive or negative trend and there is strong weekly seasonality that does have some variance.

Lets compute the optimal lambda for the ATM1 data.

```
atm1_lambda <- atm1 %>%
  features(Cash, features = guerrero) %>%
  pull(lambda_guerrero)

atm1_lambda
```

```
## [1] 0.2599006
```

The lambda above hints that a transformation may be useful. So, I will also try models on the transformed data.

Now lets try some models. Lets try a seasonal naive model, a ETS model, and an ARIMA model we will then compare the models and choose the best one for the final forecasts.

```
atm1_models <- atm1 %>%
  model(
    sNaive = SNAIVE(Cash),
    ets = ETS(Cash),
    arima = ARIMA(Cash),
    sNaive_trans = SNAIVE(box_cox(Cash, atm1_lambda)),
    ets_trans = ETS(box_cox(Cash, atm1_lambda)),
```

```
    arima_trans = ARIMA(box_cox(Cash, atm1_lambda))
  )

report(atm1_models)
```

```
## Warning in report.mdl_df(atm1_models): 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: 6 x 12
##   ATM   .model    sigma2 log_lik   AIC  AICc   BIC    MSE   AMSE    MAE ar_roots
##   <chr> <chr>      <dbl>   <dbl> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl> <list>
## 1 ATM1  sNaive    772.       NA    NA    NA    NA   NA     NA    NA     <NULL>
## 2 ATM1  ets       579.    -2233. 4486. 4487. 4525. 564.   568.   15.0   <NULL>
## 3 ATM1  arima     557.    -1640. 3288. 3288. 3304.  NA     NA    NA     <cpl>
## 4 ATM1  sNaive_t~   2.46     NA    NA    NA    NA   NA     NA    NA     <NULL>
## 5 ATM1  ets_trans   1.79  -1178. 2377. 2377. 2416.   1.74   1.75  0.723 <NULL>
## 6 ATM1  arima_tr~   1.74   -608. 1224. 1224. 1239.  NA     NA    NA     <cpl>
## # i 1 more variable: ma_roots <list>
```

From the output above the box-cox transformed ARIMA model seems to be performing the best with the lowest AIC sigma^2 AICC and BIC so lets look at that model to see its parameters.

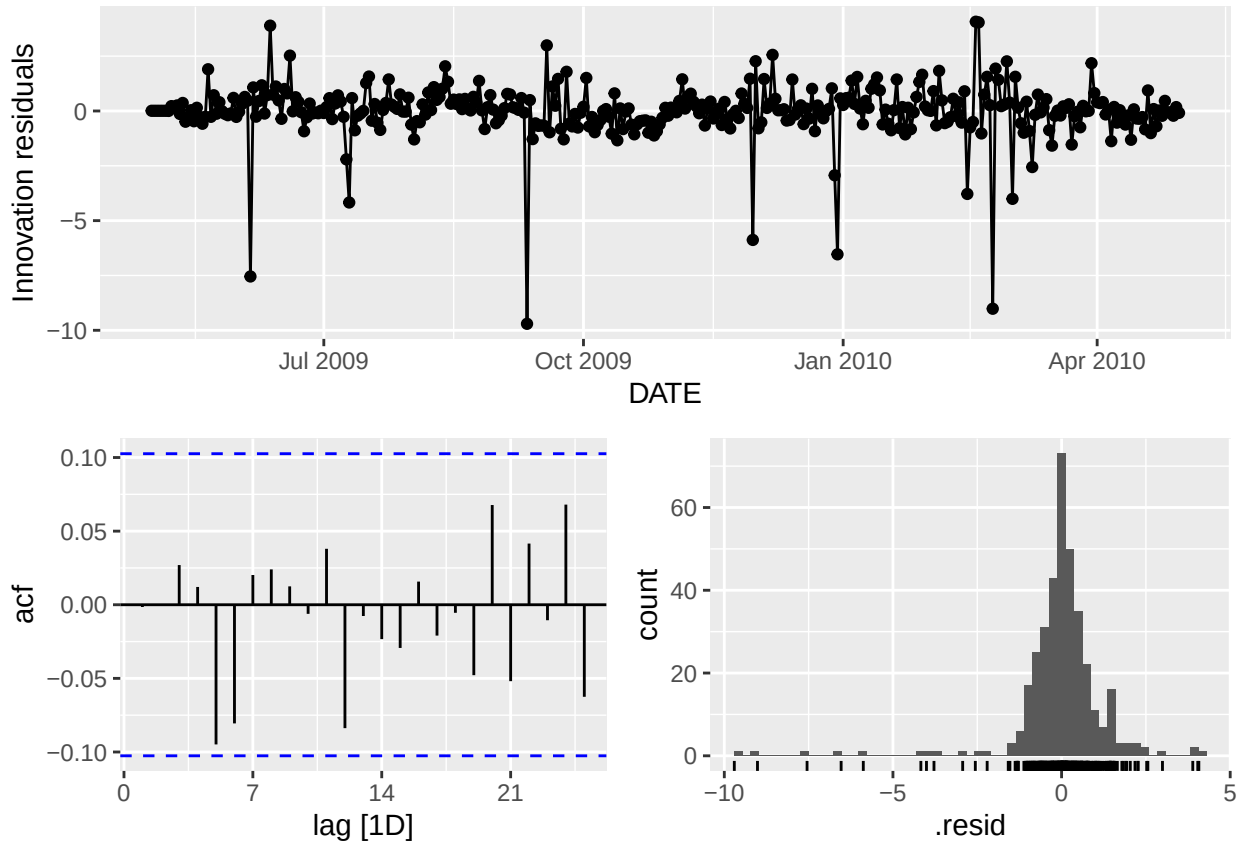
```
atm1_models %>%
  select(arima_trans) %>%
  report()
```

```
## Series: Cash
## Model: ARIMA(0,0,2)(0,1,1)[7]
## Transformation: box_cox(Cash, atm1_lambda)
##
## Coefficients:
##          ma1      ma2     sma1
##       0.1112  -0.1101  -0.6422
## s.e.  0.0524   0.0521   0.0431
##
## sigma^2 estimated as 1.745:  log likelihood=-607.97
## AIC=1223.95   AICc=1224.06   BIC=1239.47
```

We can see that the best model was an ARIMA(0,0,2)(0,1,1)[7] model and it has weekly seasonality. The auto fitted model has 2 components the first part has no auto regressive term, no differencing and 2 Moving averages then the seasonality part has no auto regressive term, first order difference and 1 moving average.

Lets take a look at the residuals

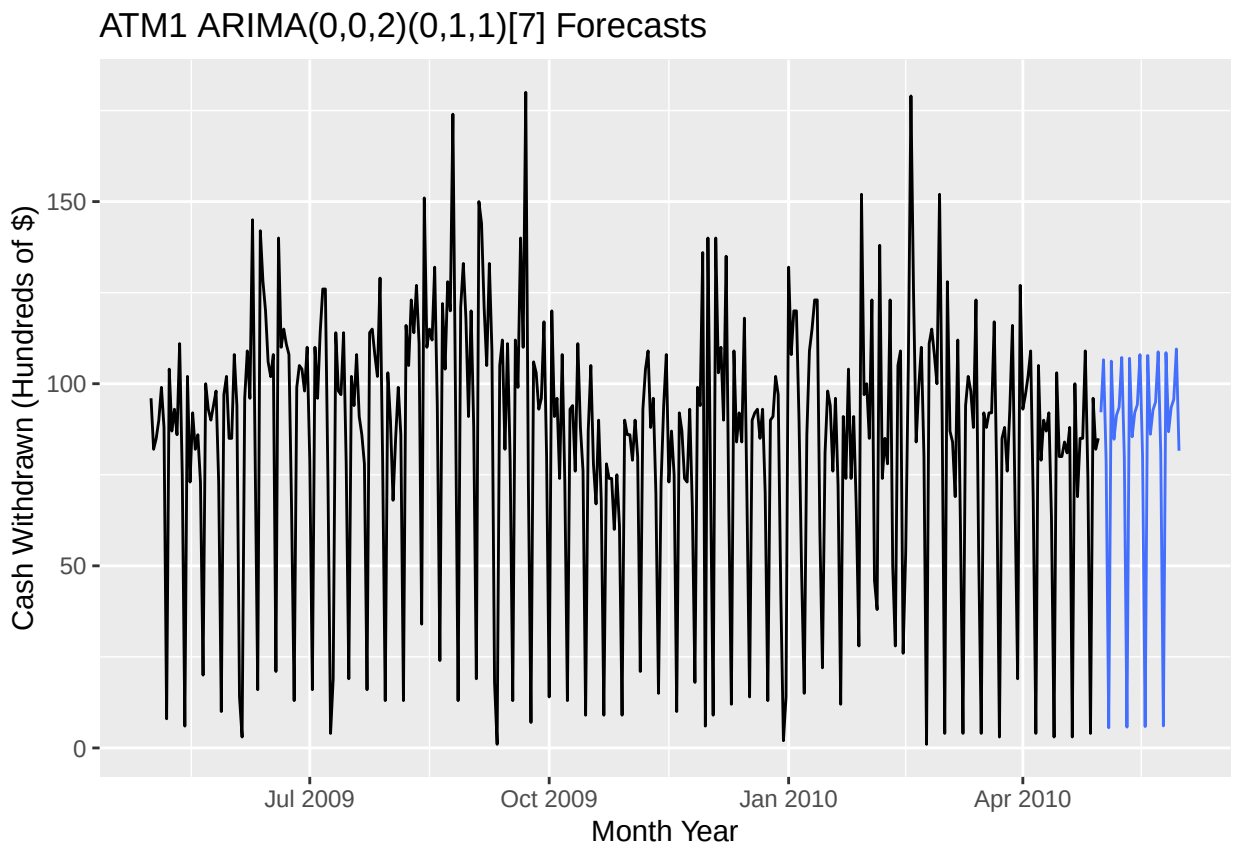
```
atm1 %>%
  model(ARIMA(box_cox(Cash,atm1_lambda))) %>%
  gg_tsresiduals()
```



We can see that the residuals appear mostly normally distributed with a mean centered around 0. The ACF plot also appears to be white noise which is good.

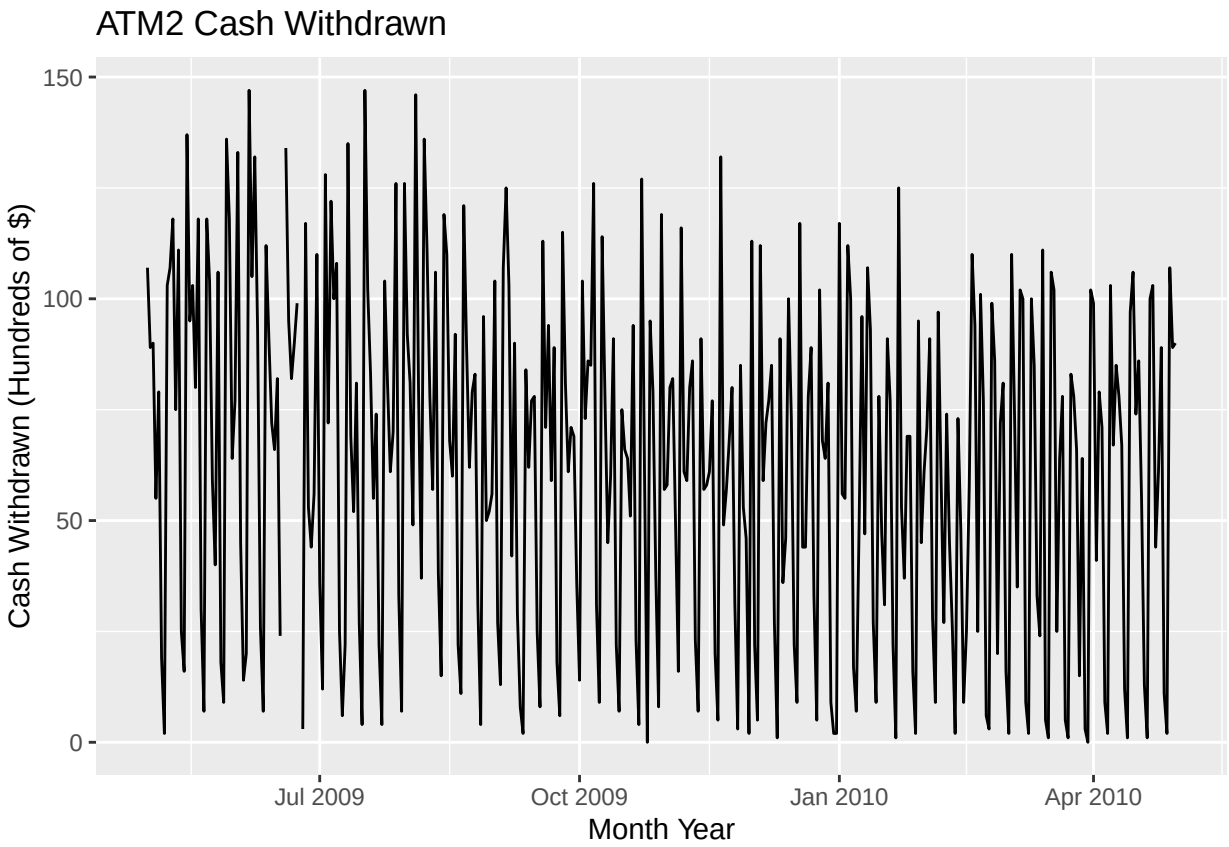
```
library(writexl)
atm1_forecast <- atm1 %>%
  model(ARIMA(box_cox(Cash, atm1_lambda))) %>%
  forecast(h=31)

atm1_forecast %>%
  autoplot(atm1,level = NA) +
  labs(title = "ATM1 ARIMA(0,0,2)(0,1,1)[7] Forecasts", x="Month Year", y="Cash Withdrawn (Hundreds of $
```

```
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
```




```
# Convert forecast to a data frame
atm1_forecast_df <- as_tibble(atm1_forecast)

# Write the forecast data to an Excel file
write_xlsx(atm1_forecast_df, "atm1_forecasts.xlsx")
```

```
atm2 %>%
  autoplot(Cash) +
  labs(title = "ATM2 Cash Withdrawn", x = "Month Year", y="Cash Withdrawn (Hundreds of $)")
```



ATM 2

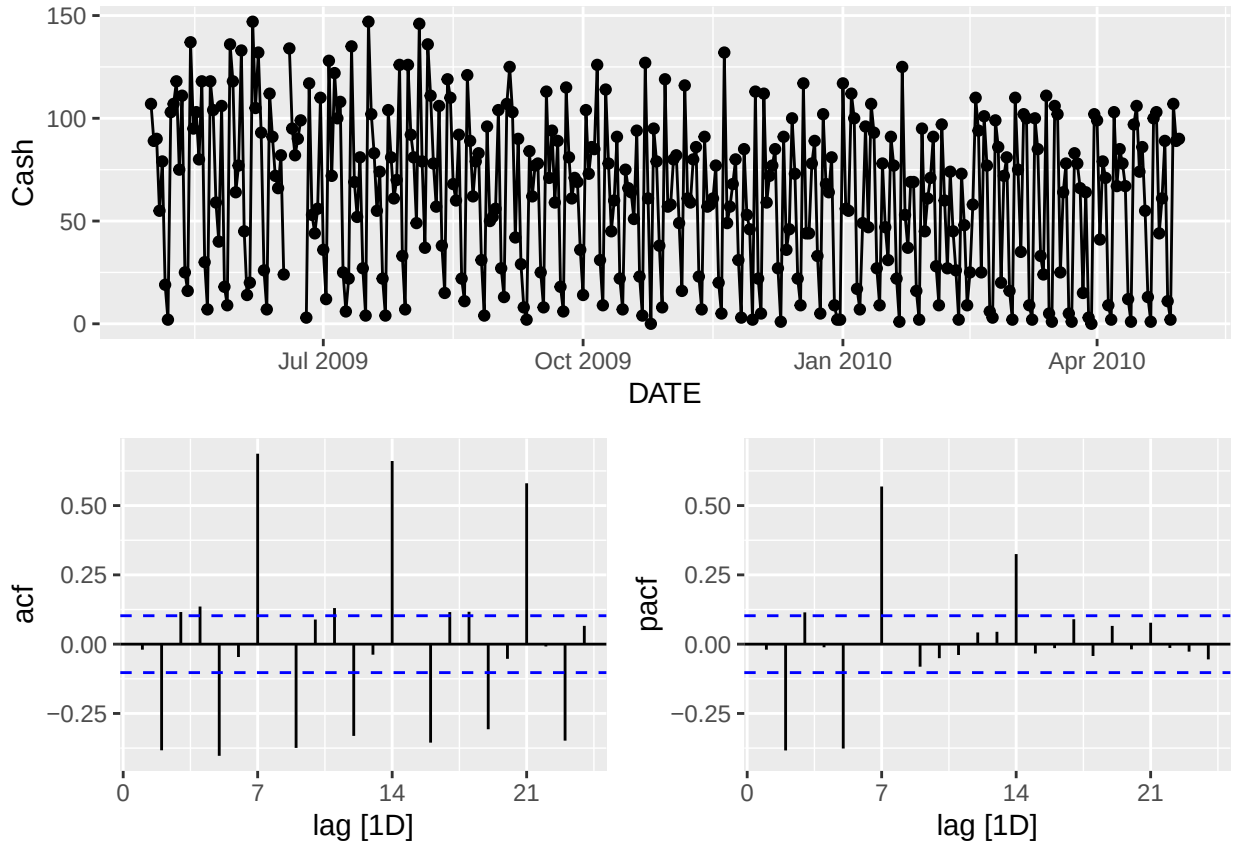
If we remember we have a couple of missing values for ATM2 so lets look at these and then decide on how to impute them it seems like the same method as before may be useful.

```
atm2 %>%
  filter(if_any(everything(), is.na))
```

```
## # A tsibble: 2 x 3 [1D]
## # Key:       ATM [1]
##   DATE       ATM    Cash
##   <date>     <chr> <dbl>
## 1 2009-06-18 ATM2     NA
## 2 2009-06-24 ATM2     NA
```

```
atm2 %>%
  gg_tsdisplay(Cash, plot_type = "partial")
```

```
## Warning: Removed 2 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



The plot above show that the data appears to have strong weekly seasonality just like atm1 so I will use the same method to impute the missing values.

```
atm2 <- atm2 %>%
  mutate(Cash = na_seadec(Cash, algorithm = "interpolation", find_frequency= TRUE))
atm2 <- atm2 %>%
  filter(DATE >= "2009-06-13")
atm2 %>%
  filter(if_any(everything(), is.na))
```

```
## # A tsibble: 0 x 3 [?]
## # Key:       ATM [0]
## # i 3 variables: DATE <date>, ATM <chr>, Cash <dbl>
```

Now we can see there are no missing values for atm2

Lets look to see if a box-cox transformation is needed, lets compute lambda and check its value

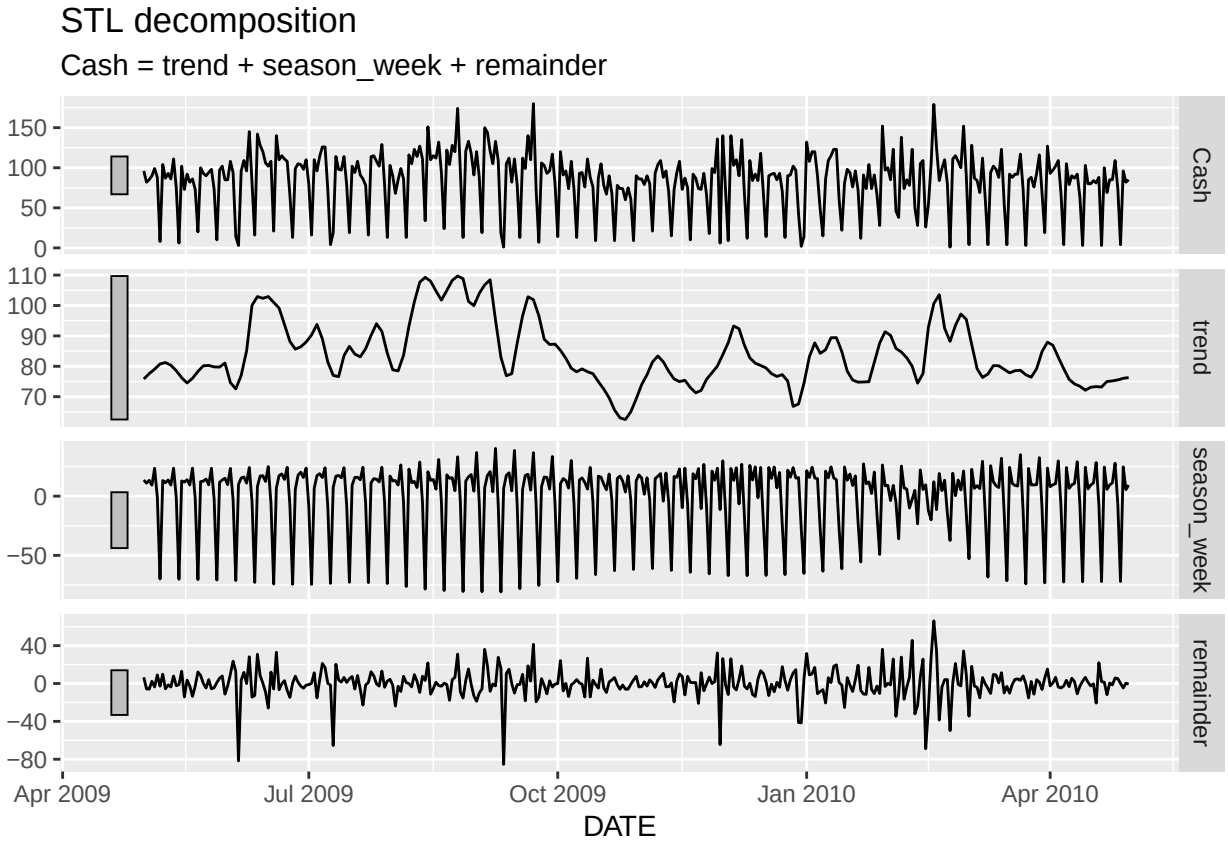
```
atm2_lambda <- atm2 %>%
  features(Cash, features = guerrero) %>%
  pull(lambda_guerrero)

atm2_lambda
```

```
## [1] 0.8849234
```

The lambda given above of 0.88 is close to one and a transformation might not make a hugw difference but lets try it anyway just to see if the results vary or improve at all.

lets try some STL decomposition on the data.

```
atm1_stl <- atm1 %>%
  model(STL(Cash))

atm1_stl %>%
  components() %>%
  autoplot()
```



From the STL decomposition above we can see the trend fluctuates. There is string weekly seasonality but the variance decreases and then increases over time for the seasonality.

Now lets try some models lets try a seasonal naive model, an ETS model a ARIMA model and a decomposition model

```
atm2_models <- atm2 %>%
  model(
    seasonl_naive = SNAIVE(Cash),
    ets = ETS(Cash),
    arima = ARIMA(Cash),
    seasonal_n_trans = SNAIVE(box_cox(Cash,atm2_lambda)),
    ets_trans = ETS(box_cox(Cash,atm2_lambda)),
    arima_trans = ARIMA(box_cox(Cash,atm2_lambda))
  )
```

```
report(atm2_models)
```

```
## Warning in report.mdl_df(atm2_models): 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: 6 x 12
##   ATM   .model       sigma2 log_lik   AIC  AICc   BIC   MSE  AMSE   MAE ar_roots
##   <chr> <chr>         <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <list>
## 1 ATM2  seasonl_nai~   870.      NA    NA    NA    NA    NA    NA    NA   <NULL>
## 2 ATM2  ets            634.  -1964. 3948. 3949. 3986.  616.  618.  17.5 <NULL>
## 3 ATM2  arima          625.  -1460. 2931. 2931. 2950.    NA    NA    NA   <cpl>
## 4 ATM2  seasonal_n_~   344.      NA    NA    NA    NA    NA    NA    NA   <NULL>
## 5 ATM2  ets_trans      253.  -1816. 3652. 3652. 3689.  246.  246.  11.0 <NULL>
## 6 ATM2  arima_trans    250.  -1316. 2643. 2643. 2661.    NA    NA    NA   <cpl>
## # i 1 more variable: ma_roots <list>
```

ARIMA model with the box-cox transformed data seems to give the best results. So, lets produce forecasts for this model.

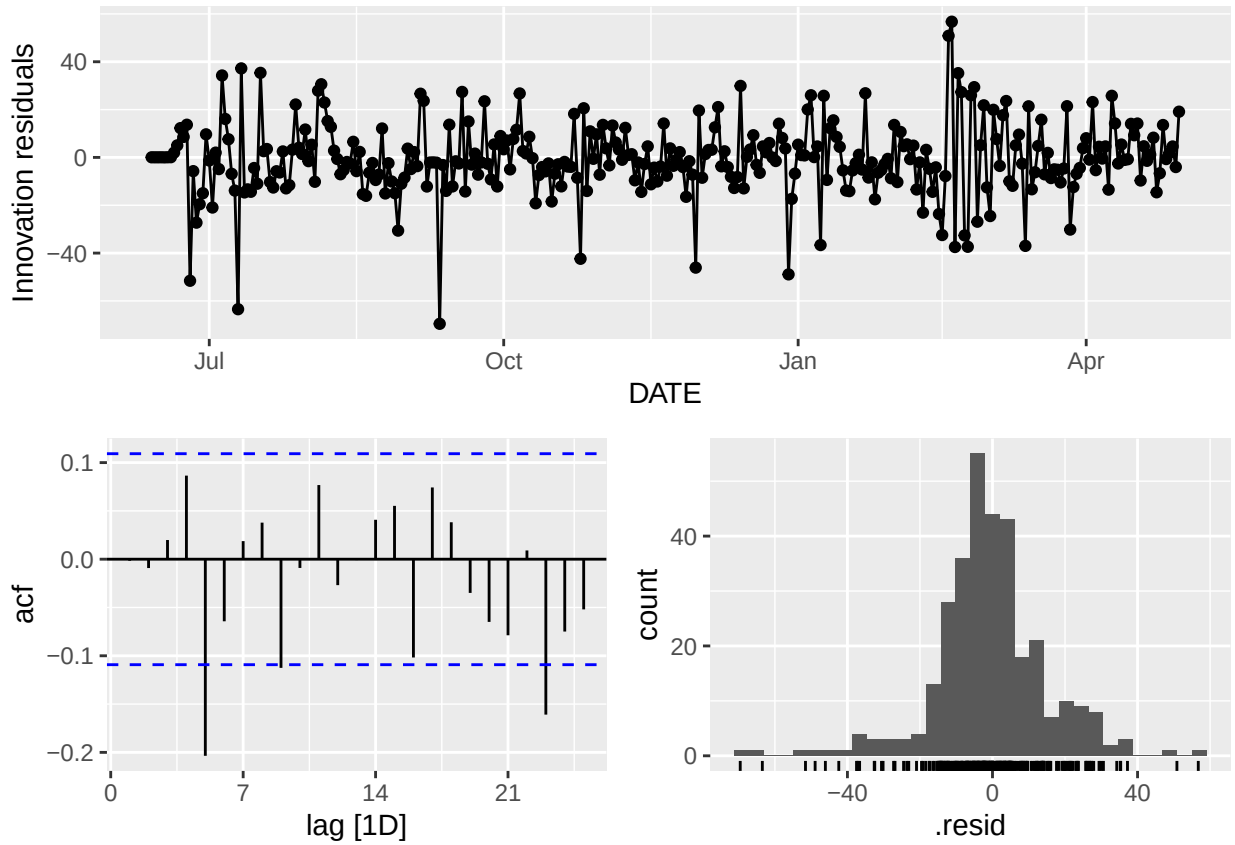
```
atm2_models %>%
  select(arima_trans) %>%
  report()
```

```
## Series: Cash
## Model: ARIMA(0,0,2)(1,1,1)[7]
## Transformation: box_cox(Cash, atm2_lambda)
##
## Coefficients:
##          ma1      ma2    sar1     sma1
##       0.0273  -0.0864  -0.117  -0.5265
## s.e.  0.0559   0.0549   0.105   0.0974
##
## sigma^2 estimated as 250.1:  log likelihood=-1316.27
## AIC=2642.53   AICc=2642.73   BIC=2661.3
```

We can see the optimal ARIMA model given above is an ARIMA(0,0,2)(1,1,1)[7] model. The nonseasonal part has no auto regressive term, no differencing and 2 moving averages. Then the seasonal part has 1 auto regressive term first order differencing and one moving average with weekly seasonality.

Lets check the models residuals

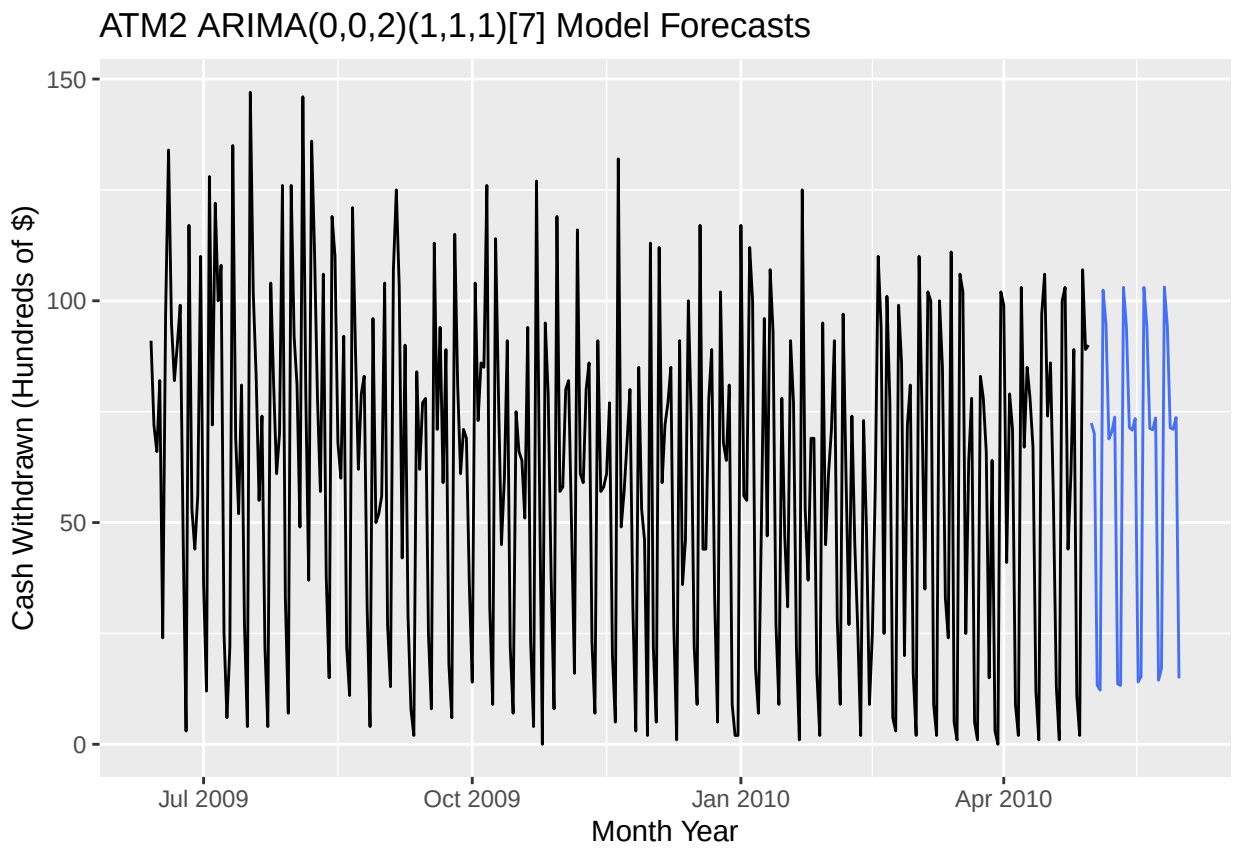
```
atm2 %>%
  model(ARIMA(box_cox(Cash, atm2_lambda))) %>%
  gg_tsresiduals()
```



The residuals look mostly normal with some skew and the ACF plot shows some significant spikes lets see how the forecasts look.

```
atm2_forecast <- atm2 %>%
  model(ARIMA(box_cox(Cash, atm2_lambda))) %>%
  forecast(h=31)

atm2_forecast %>%
  autoplot(atm2, level=NA) +
  labs(title = "ATM2 ARIMA(0,0,2)(1,1,1)[7] Model Forecasts", x= "Month Year", y = "Cash Withdrawn (Hun
```

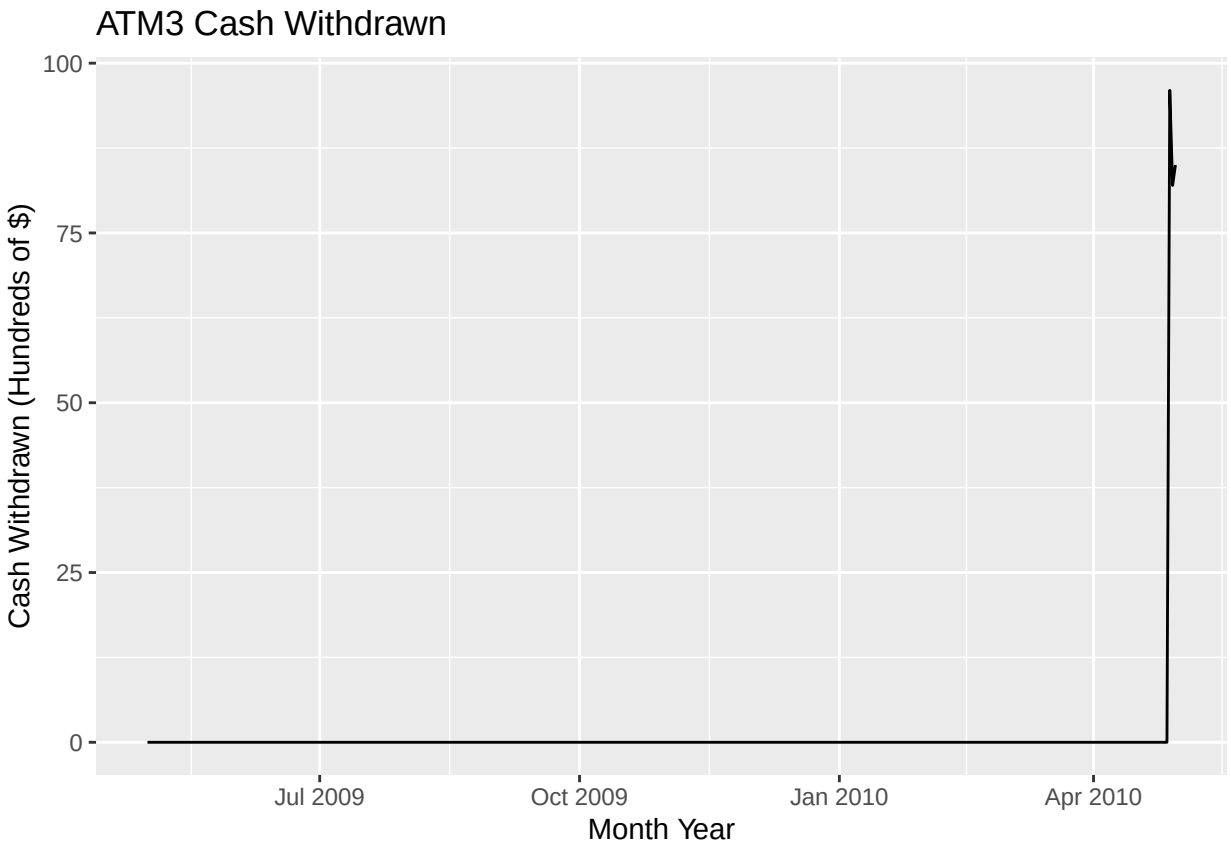
```
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
```




```
# Convert forecast to a data frame
atm2_forecast_df <- as_tibble(atm2_forecast)

# Write the forecast data to an Excel file
write_xlsx(atm2_forecast_df, "atm2_forecasts.xlsx")
```

The forecasts look pretty good they seem to just be a little bit off as the peaks and spikes are not quite as high as the previous data but still pretty good forecasts.

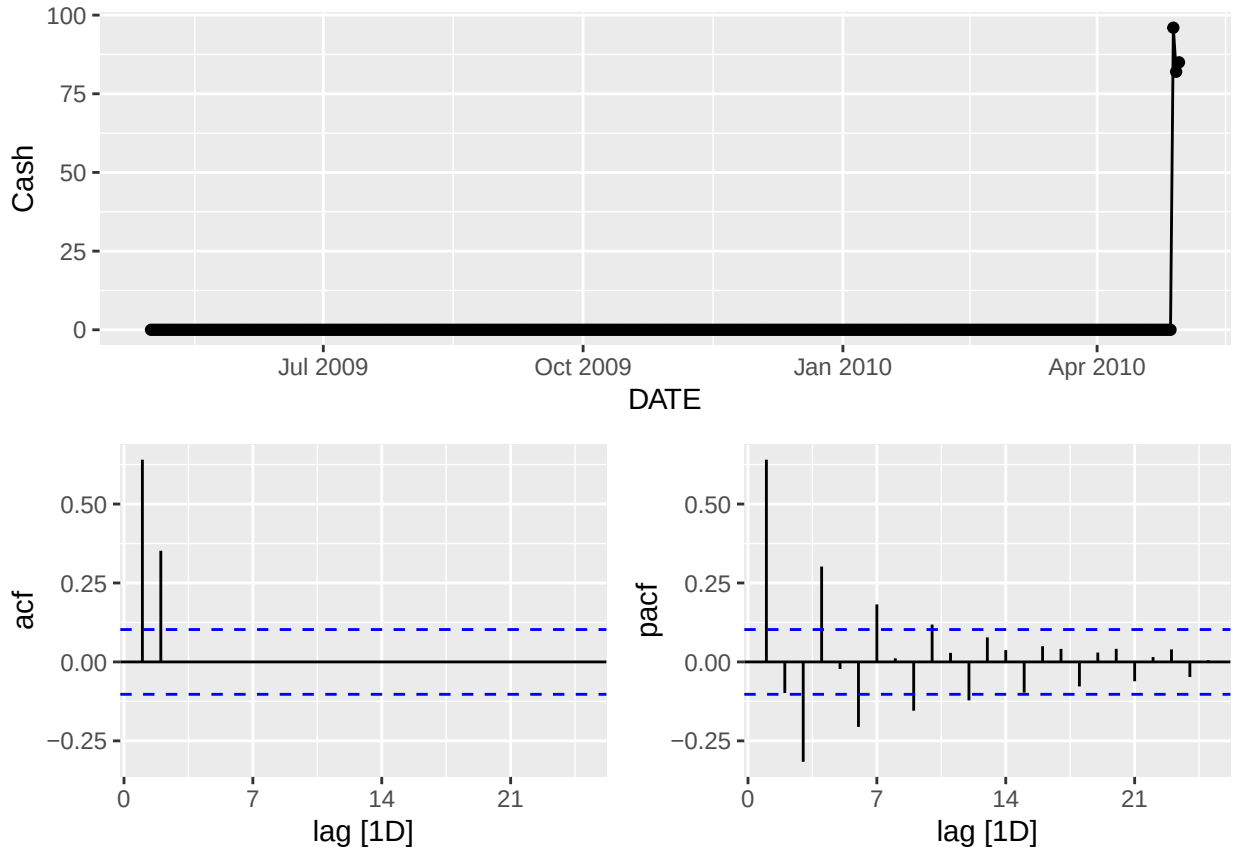
```
atm3 %>%
  autoplot(Cash) +
    labs(title = "ATM3 Cash Withdrawn", x= "Month Year", y = "Cash Withdrawn (Hundreds of $)")
```



ATM 3

This ATM has very limited data it seems most days have 0 withdrawals with only a few recent withdrawals.

```
atm3 %>%
  gg_tsdisplay(Cash, plot_type = "partial")
```



The ACF plot shows 2 significant spikes and then goes to 0. We can also see that there's 0 Cash withdrawn until the most 3 recent data points which have withdraws. This ATM could have been just installed and that's why the values are mostly 0.

Since this data is so sparse it does not seem like a transformation would have much impact. There is also no apparent seasonality in the data. Which does not mean that there is not any, this could be due to the limited amount of non-zero values.

So, lets try a simpler model like a Naive model along with some of the more complex models like ETS and ARIMA

```
atm3_models <- atm3 %>%
  model(
    naive = NAIVE(Cash),
    ets = ETS(Cash),
    arima = ARIMA(Cash),

  )

atm3_models %>%
  select(arima) %>%
  report()
```

```
## Series: Cash
## Model: ARIMA(0,0,2)
##
## Coefficients:
```

```
##          ma1     ma2
##       0.8392  0.8557
## s.e.  0.0496  0.0611
##
## sigma^2 estimated as 25.4:  log likelihood=-1108.69
## AIC=2223.39   AICc=2223.46   BIC=2235.09
```

```
atm3_models %>%
  select(ets) %>%
  report()
```

```
## Series: Cash
## Model: ETS(A,N,N)
##   Smoothing parameters:
##     alpha = 0.8585628
##
##   Initial states:
##        l[0]
##  -0.1269535
##
##   sigma^2:  25.4128
##
##      AIC     AICc      BIC
## 3338.324 3338.391 3350.024
```

```
atm3_models %>%
  select(naive) %>%
  report()
```

```
## Series: Cash
## Model: NAIVE
##
## sigma^2: 25.8985
```

```
atm3_models %>%
  report()
```

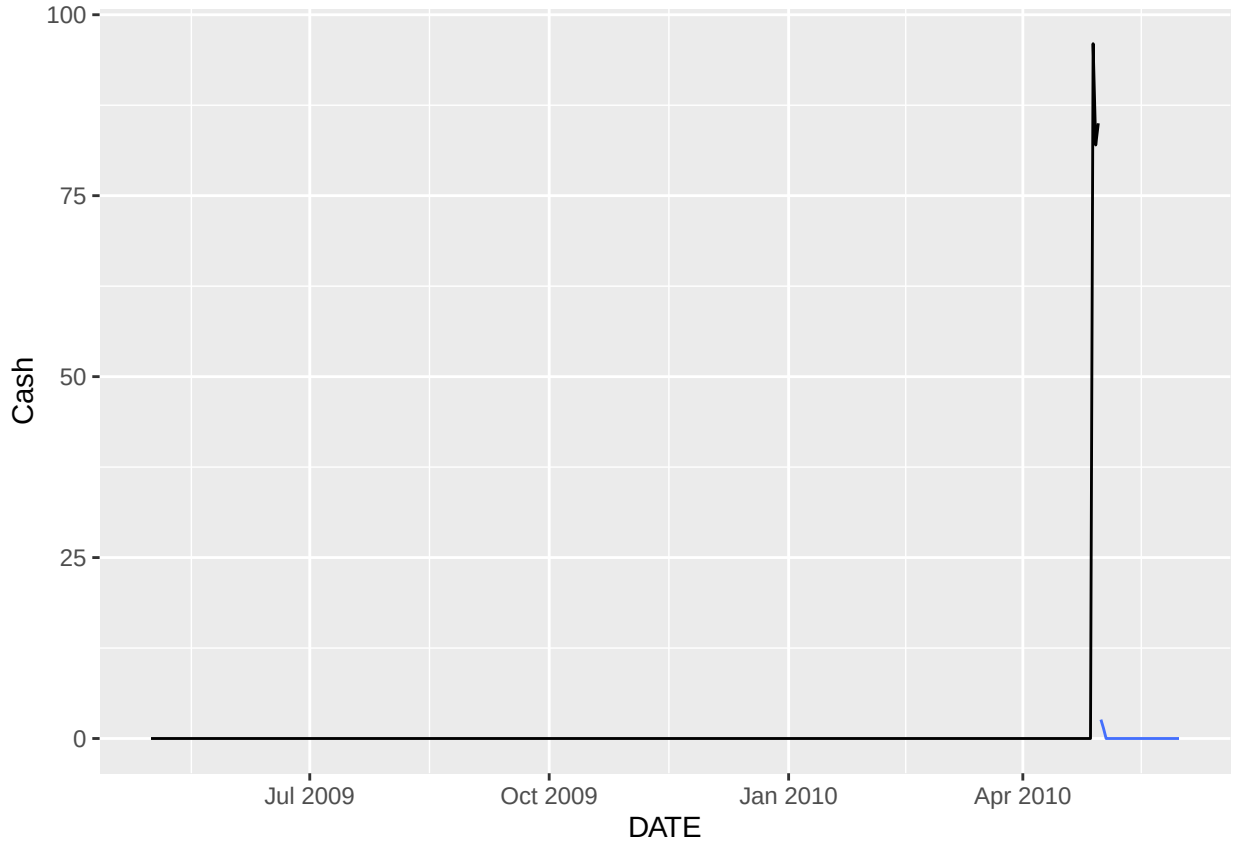
```
## Warning in report.mdl_df(.): Model reporting is only supported for individual
## models, so a glance will be shown. To see the report for a specific model, use
## `select()` and `filter()` to identify a single model.
```

```
## # A tibble: 3 x 12
##   ATM   .model sigma2 log_lik   AIC  AICc   BIC   MSE  AMSE    MAE ar_roots
##   <chr> <chr>   <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl> <list>
## 1 ATM3  naive    25.9     NA    NA    NA    NA   NA    NA   NA     <NULL>
## 2 ATM3  ets      25.4  -1666. 3338. 3338. 3350.  25.3  44.3  0.273 <NULL>
## 3 ATM3  arima    25.4  -1109. 2223. 2223. 2235.  NA    NA   NA     <cpl [0]>
## # i 1 more variable: ma_roots <list>
```

looking at the results the ARIMA model with (0,0,2) as the parameters seems to give the best results as we can see it gives the lowest sigma^2 and it also gives the lowest AIC, AICC and BIC of all the models. Lets try forecasting with the ARIMA(0,0,2) model.

```
atm3_forecast <- atm3 %>%
  model(ARIMA(Cash)) %>%
  forecast(h=31)

atm3_forecast %>%
  autoplot(atm3, level=NA)
```

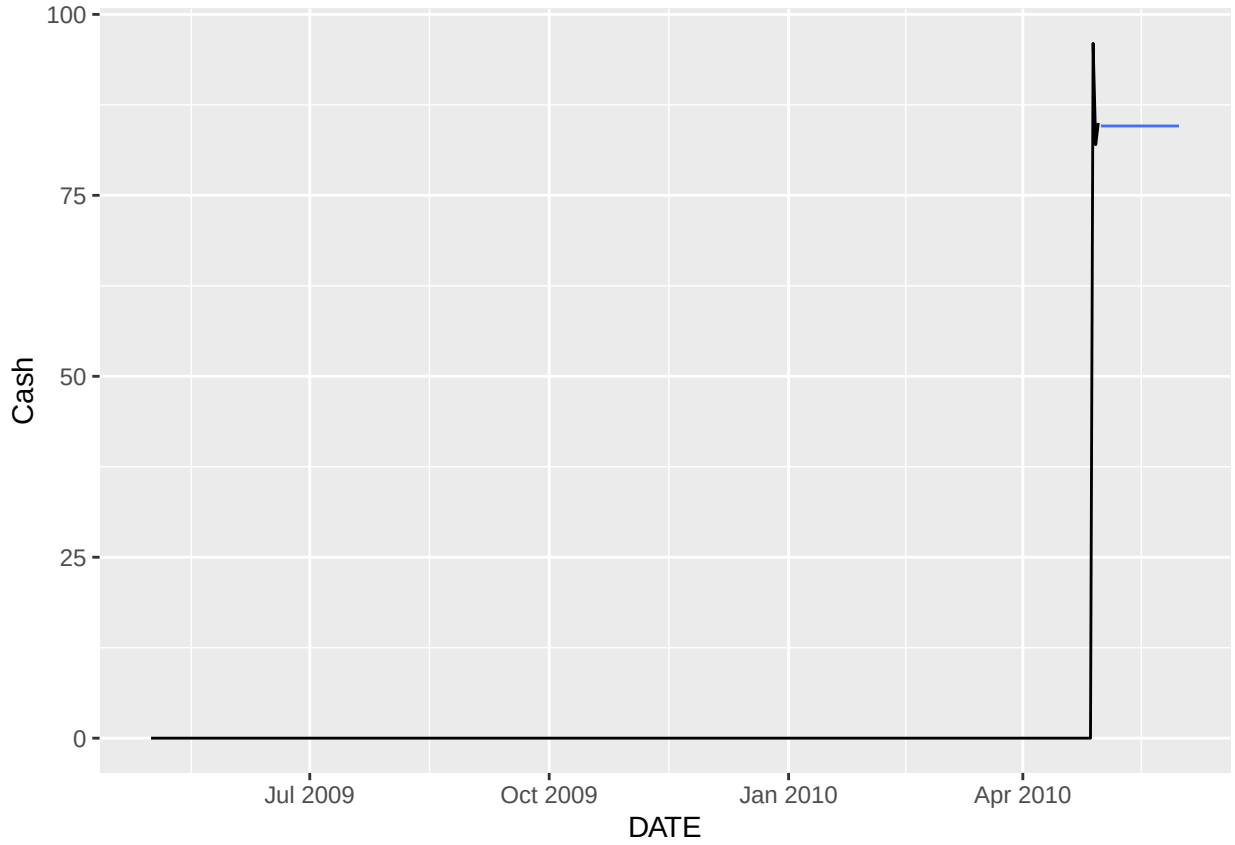
```
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
```



This model just seems to be predicting 0 everytime so lets try the ETS model forecasts.

```
atm3_forecast <- atm3 %>%
  model(ETS(Cash)) %>%
  forecast(h=31)

atm3_forecast %>%
  autoplot(atm3, level=NA)
```

```
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
```



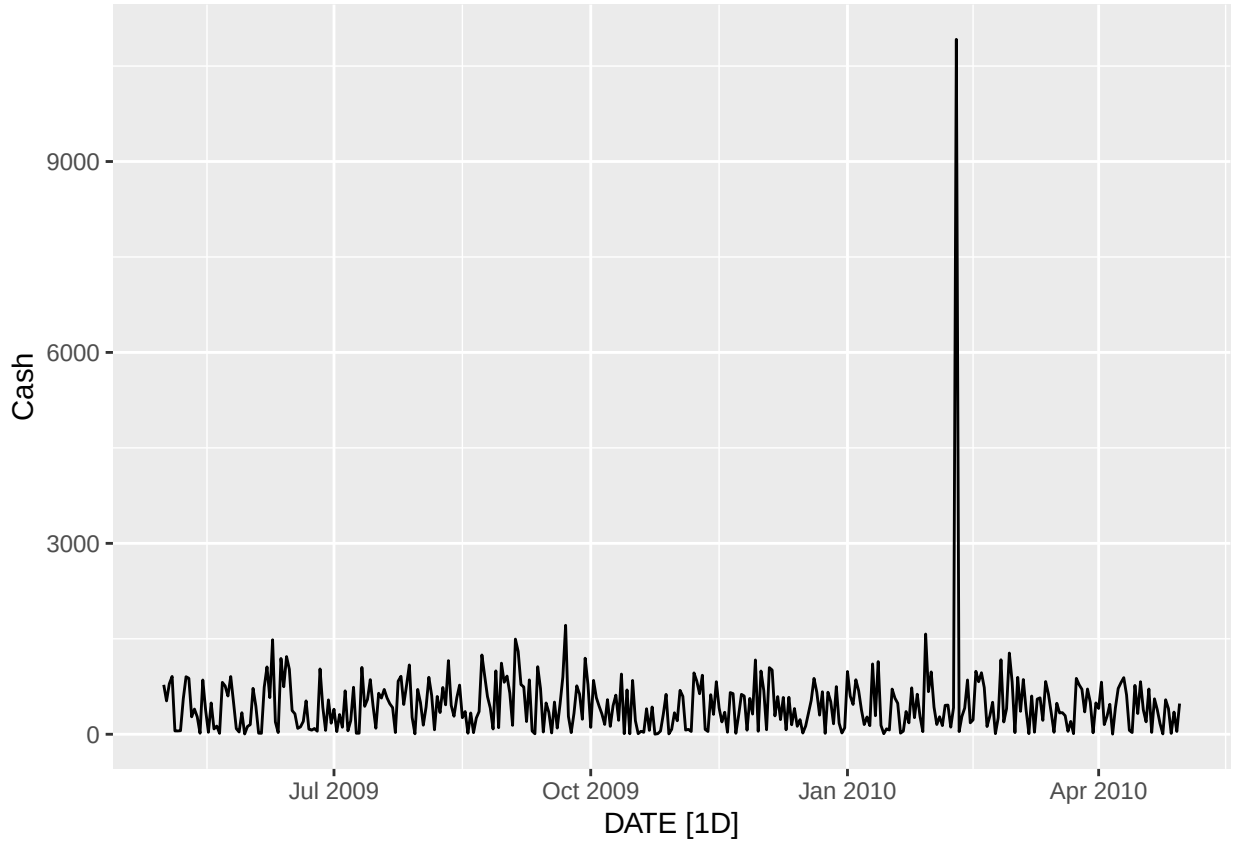

```
# Convert forecast to a data frame
atm3_forecast_df <- as_tibble(atm3_forecast)

# Write the forecast data to an Excel file
write_xlsx(atm3_forecast_df, "atm3_forecasts.xlsx")
```

These forecasts seem better so lets use these.

ATM 4 Now lets look at the last ATM to produce forecasts for.

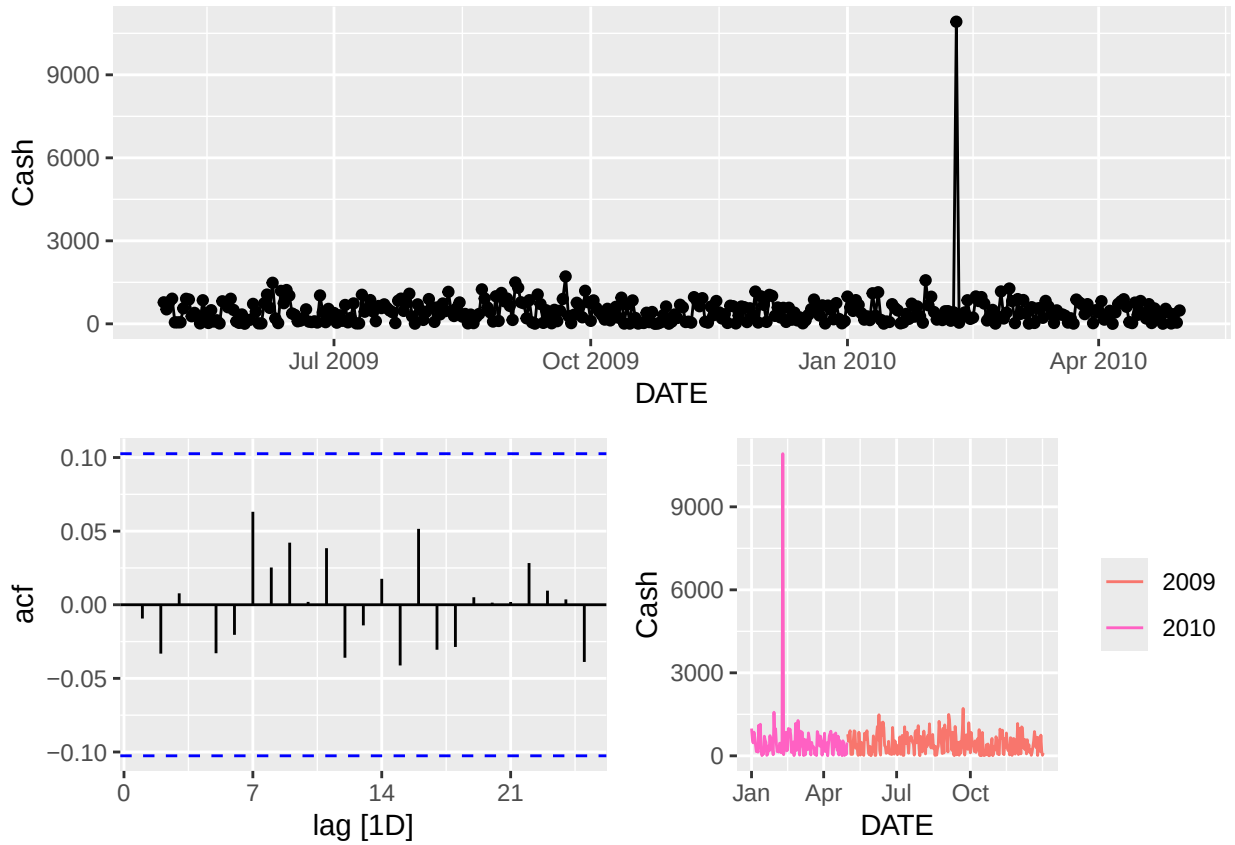
```
atm4 %>%
  autoplot()
```

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



Looking at the plot above we can see that there is one obvious outlier in the data with a huge spike. This will have to be dealt with as it could highly impact modeling. The outlier may have to be removed and then imputed with interpolation so that the data point is closer to the other data points.

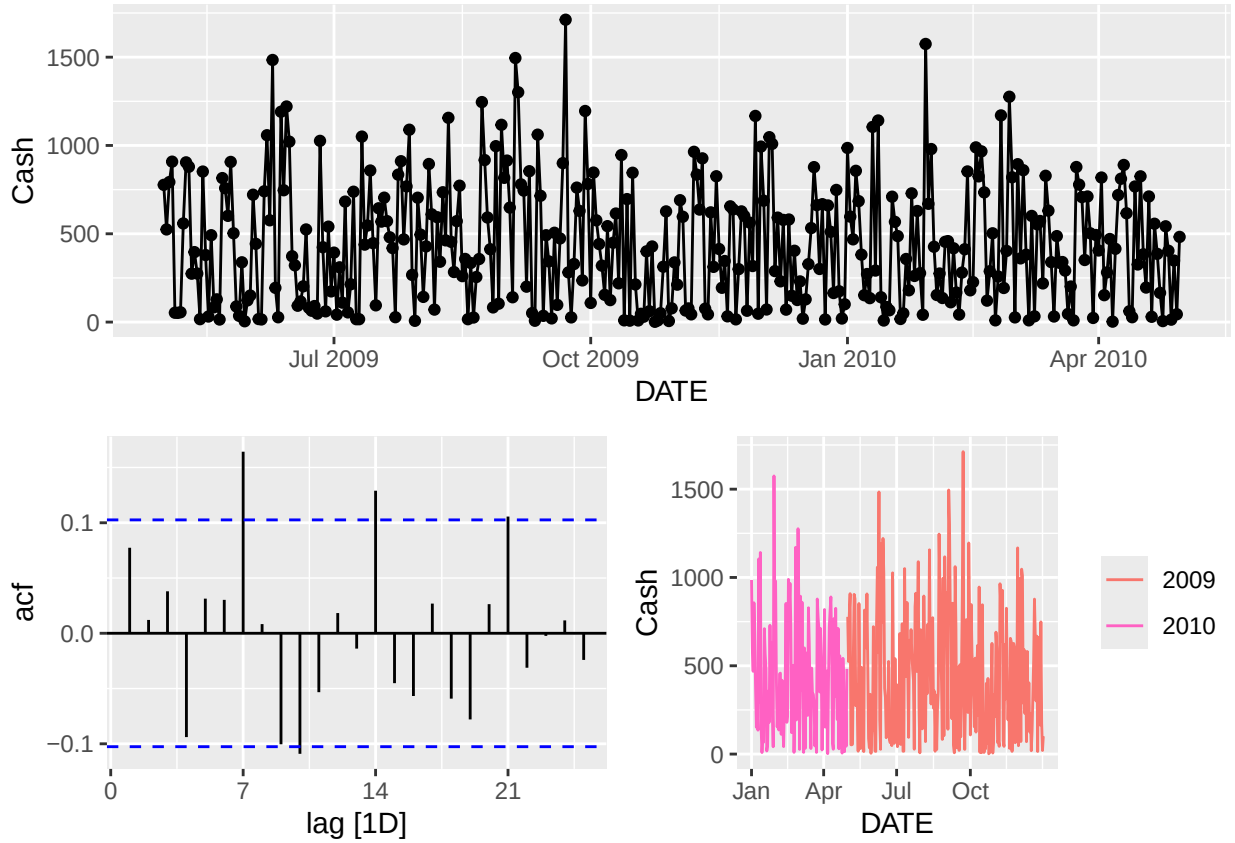
```
atm4 %>%
  gg_tsdisplay(Cash)
```



We can see from the residuals plot above that the outlier seems to be having a big effect on the data. There appears to be no seasonality but looking at the time plot and from domain knowledge there should be some seasonality. So lets find, remove and impute the outlier and then lets plot the residuals again.

```
atm4_clean <- atm4 %>%
  mutate(Cash = ifelse(Cash > 9000, NA, Cash))

atm4_clean <- atm4_clean %>%
  mutate(Cash = na_seadec(Cash, algorithm = "interpolation", find_frequency= TRUE))

atm4_clean %>%
  gg_tsdisplay(Cash)
```

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



Looking at the plot now we can see that there is clear weekly seasonality in the ACF plot.

Lets check if a transformation of the data would be beneficial.

```
atm4_lambda <- atm4_clean %>%
  features(Cash, features = guerrero) %>%
  pull(lambda_guerrero)

atm4_lambda
```

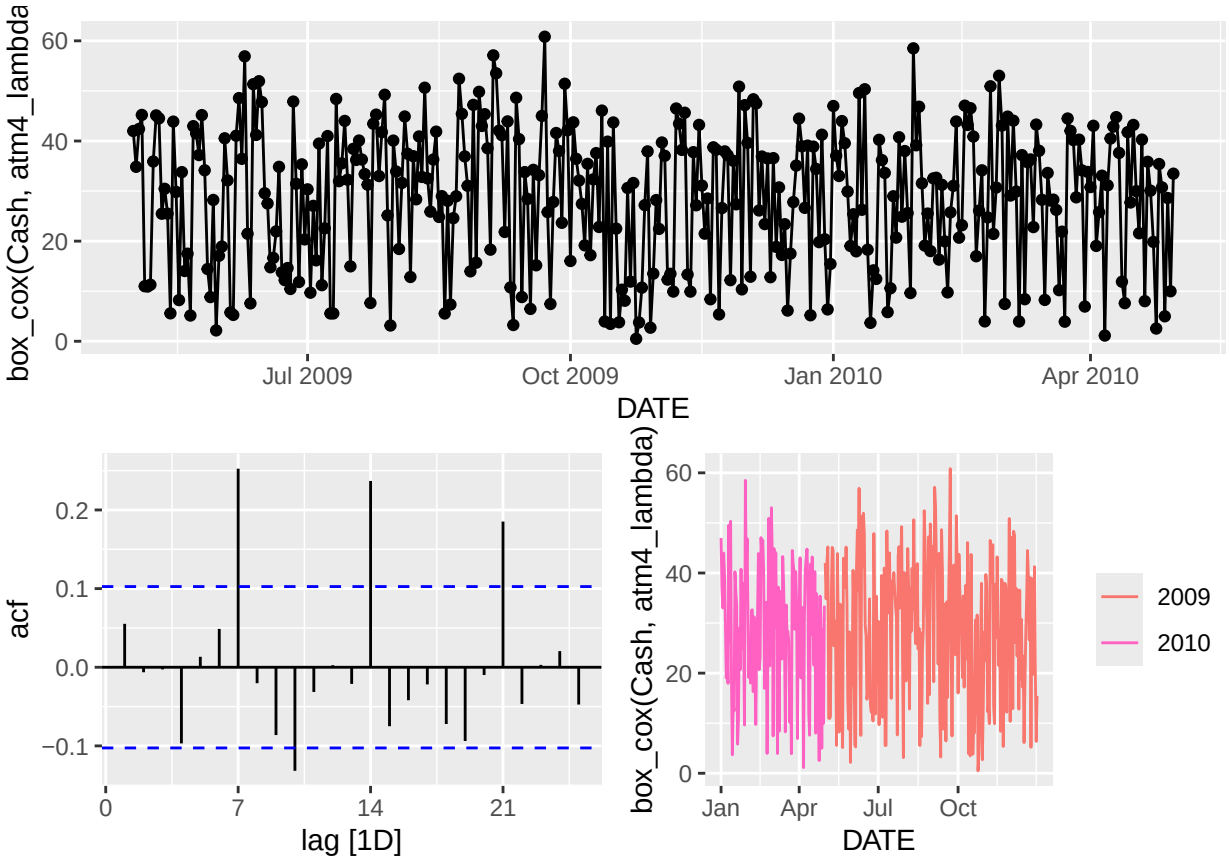
```
## [1] 0.4490413
```

This lambda hints that a transformation may be beneficial to stabilize the variance.

Lets look at the transformed data to see if its stationary or not.

```
atm4_clean %>%
  gg_tsdisplay(box_cox(Cash, atm4_lambda))
```

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



There appears to be weekly seasonality so the data is still not stationary. Differencing may be needed.

```
atm4_models <- atm4_clean %>%
  model(
    seasonal_naive = SNAIVE(Cash),
    ets = ETS(Cash),
    arima = ARIMA(Cash),
    seasonal_naive_trans = SNAIVE(box_cox(Cash, atm4_lambda)),
    ets_trans = ETS(box_cox(Cash, atm4_lambda)),
    arima_trans = ARIMA(box_cox(Cash, atm4_lambda))
  )

atm4_models %>%
  report()
```

```
## Warning in report.mdl_df(.): 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: 6 x 12
##   ATM   .model  sigma2 log_lik   AIC  AICc   BIC     MSE    AMSE    MAE ar_roots
##   <chr> <chr>    <dbl>   <dbl> <dbl> <dbl> <dbl>   <dbl>   <dbl>  <dbl> <list>
## 1 ATM4  seaso~ 2.06e+5     NA    NA    NA    NA      NA      NA  NA     <NULL>
## 2 ATM4  ets    7.29e-1  -3192. 6404. 6405. 6443. 125341. 125998.  0.636 <NULL>
## 3 ATM4  arima  1.18e+5  -2645. 5307. 5307. 5338.      NA      NA  NA     <cpl>
## 4 ATM4  seaso~ 2.88e+2     NA    NA    NA    NA      NA      NA  NA     <NULL>
```

```
## 5 ATM4  ets_t~ 2.21e-1  -1999. 4019. 4019. 4058.    164.    164.  0.369 <NULL>
## 6 ATM4  arima~ 1.75e+2  -1459. 2928. 2928. 2948.     NA      NA  NA     <cpl>
## # i 1 more variable: ma_roots <list>
```

From the results above the ARIMA model seems to be performing the best lets see what kind of ARIMA model it is and produce the forecasts.

```
atm4_model <- atm4_clean %>%
  model(ARIMA(box_cox(Cash, atm4_lambda)))

atm4_model %>%
  report()
```

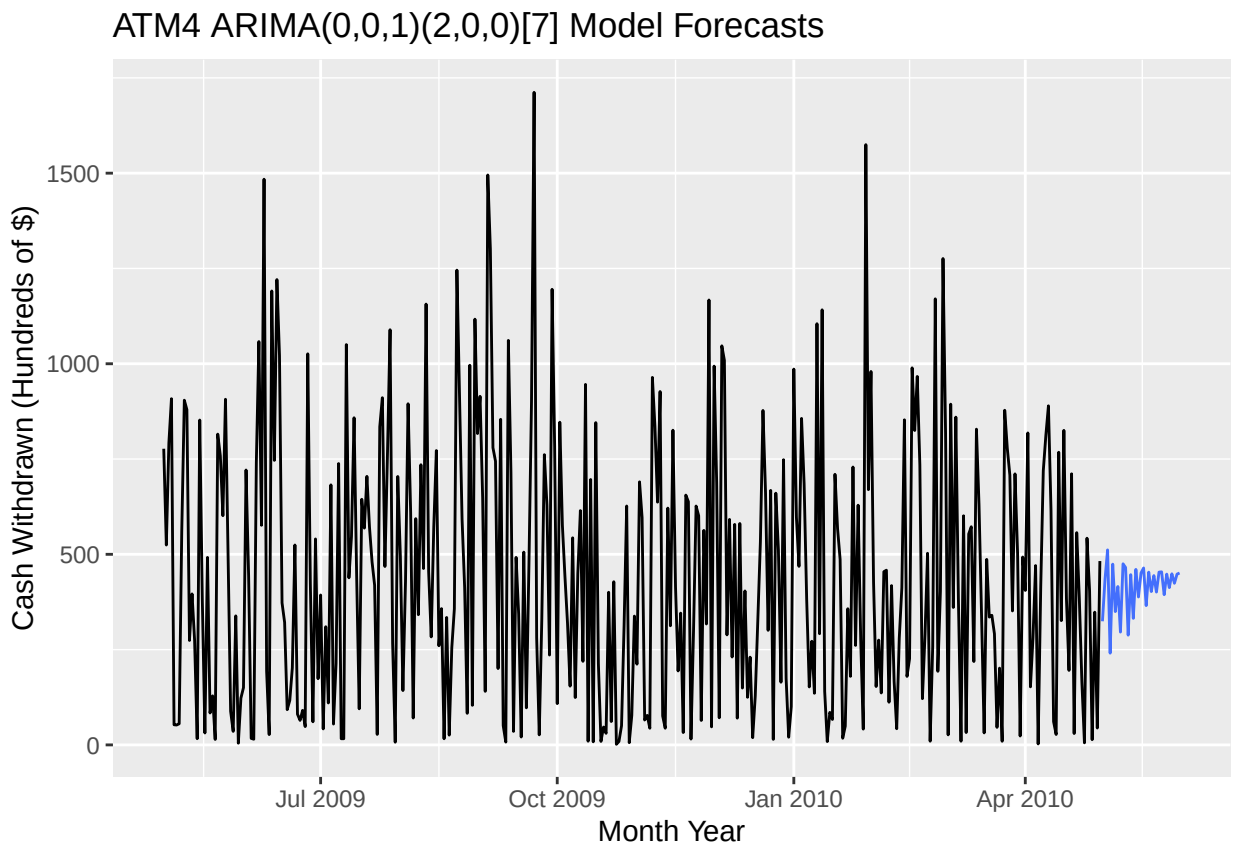
```
## Series: Cash
## Model: ARIMA(0,0,1)(2,0,0)[7] w/ mean
## Transformation: box_cox(Cash, atm4_lambda)
##
## Coefficients:
##          ma1    sar1    sar2  constant
##       0.0799  0.2071  0.2034   16.8205
## s.e.  0.0527  0.0516  0.0524    0.7295
##
## sigma^2 estimated as 175.1:  log likelihood=-1459.17
## AIC=2928.33   AICc=2928.5   BIC=2947.83
```

We can see it produced a ARIMA(0,0,1)(2,0,0)[7] model with weekly seasonality. The non seasonal part has no auto regressive term no differencing and 1 moving average. Then the seasonal part has 2 auto regressive terms and no differencing and no moving averages.

Now lets produce and plot forecasts.

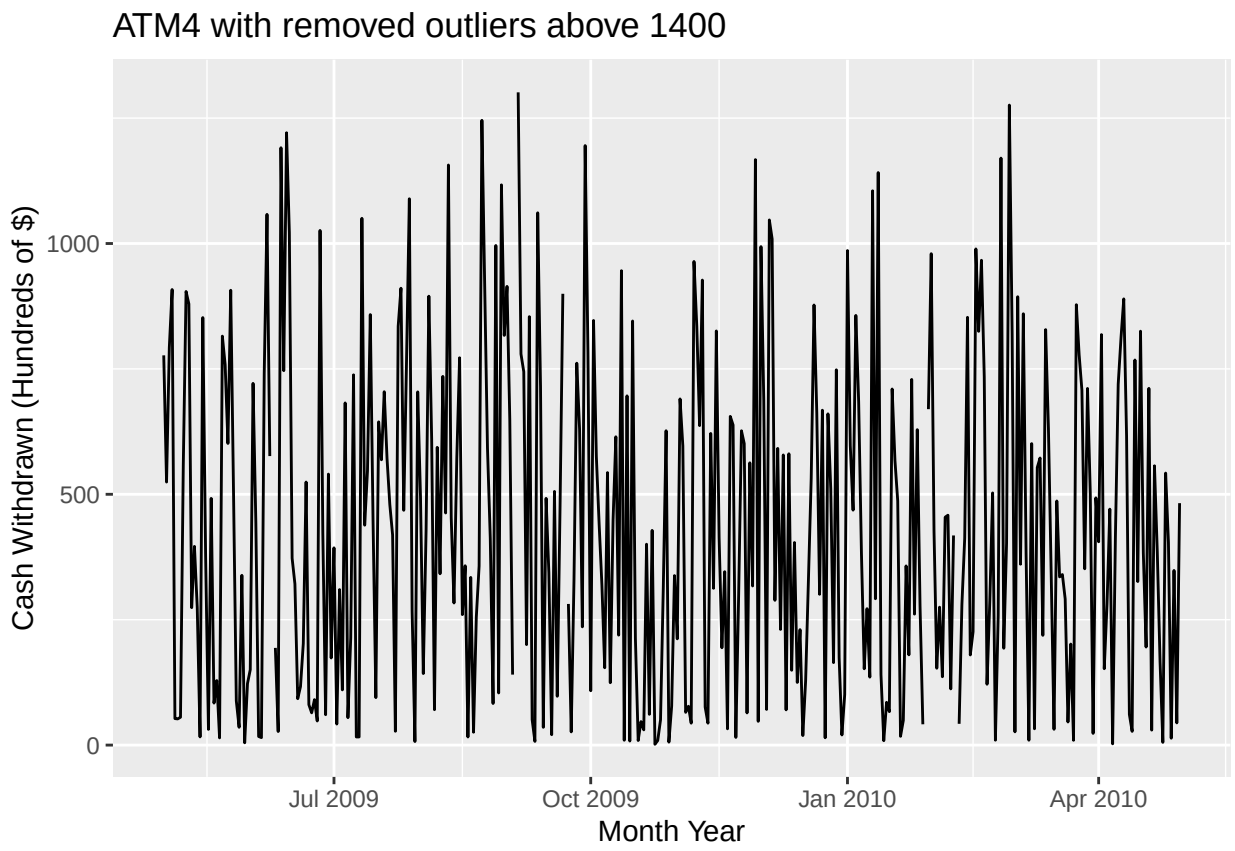
```
atm4_forecast <- atm4_model %>%
  forecast(h=31)

atm4_forecast %>%
  autoplot(atm4_clean,level=NA) +
  labs(title = "ATM4 ARIMA(0,0,1)(2,0,0)[7] Model Forecasts", x="Month Year", y="Cash Withdrawn (Hundre
```

```
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
```



Lets try removing all outliers greater than 1400 to see if we can imporve the forecasts. I will. impute them using the same technique as ATMs 1 and 2.

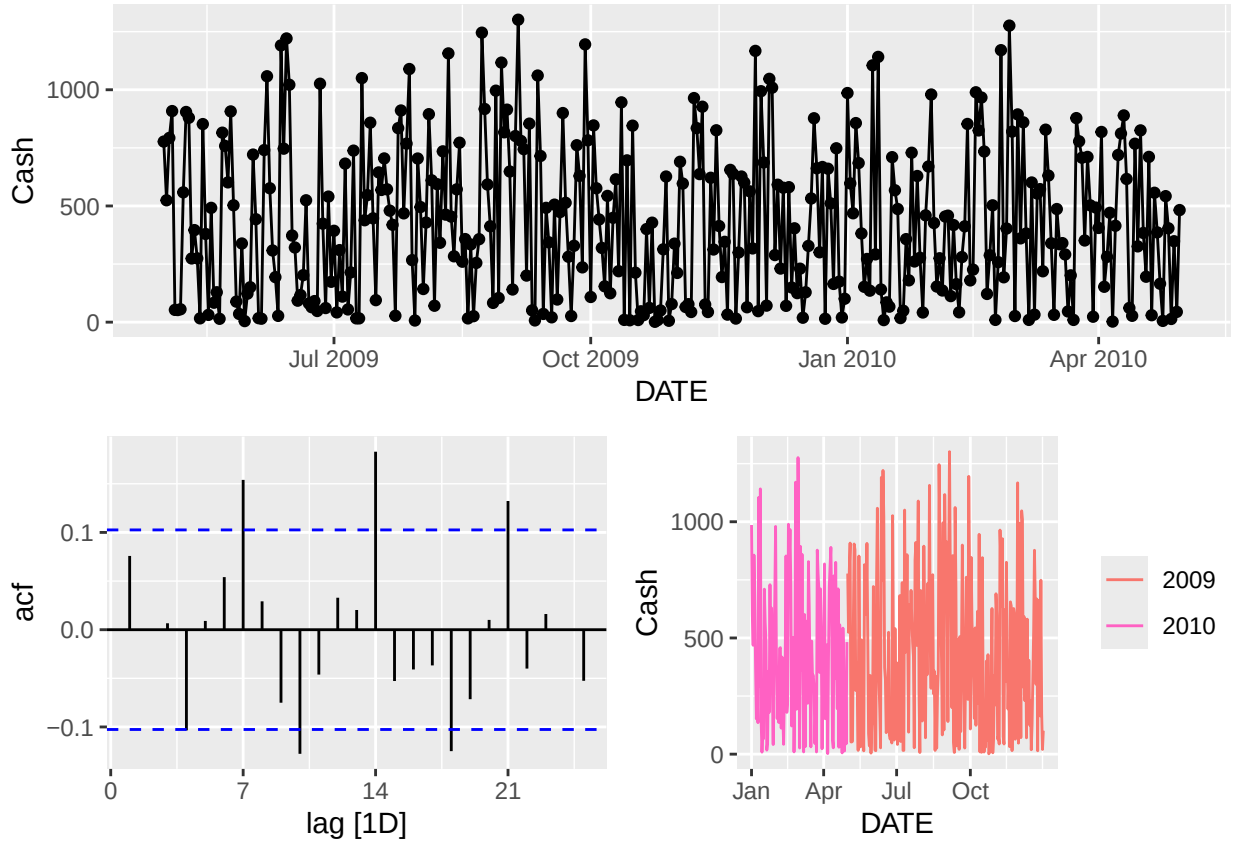
```
atm4_cleaned <- atm4 %>%
  mutate(Cash = ifelse(Cash > 1400, NA, Cash))

atm4_cleaned %>%
  autoplot(Cash) +
  labs(title = "ATM4 with removed outliers above 1400", x="Month Year", y="Cash Withdrawn (Hundreds of $
```



Now impute the data.

```
atm4_cleaned <- atm4_cleaned %>%
  mutate(Cash = na_seadec(Cash, algorithm = "interpolation", find_frequency= TRUE))

atm4_cleaned %>%
  gg_tsdisplay(Cash)
```

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




```
atm4_cleaned_lambda <- atm4_cleaned %>%
  features(Cash, features = guerrero) %>%
  pull(lambda_guerrero)

atm4_cleaned_lambda
```

```
## [1] 0.5277959
```

Now lets try some models.

```
atm4_cleaned_models <- atm4_cleaned %>%
  model(
    arima = ARIMA(Cash),
    arima_trans = ARIMA(box_cox(Cash, atm4_cleaned_lambda)),
    season_naive = SNAIVE(Cash),
    season_naive_trans = SNAIVE(box_cox(Cash, atm4_cleaned_lambda)),
    ets = ETS(Cash),
    ets_trans = ETS(box_cox(Cash, atm4_cleaned_lambda))
  )

atm4_cleaned_models %>%
  report()
```

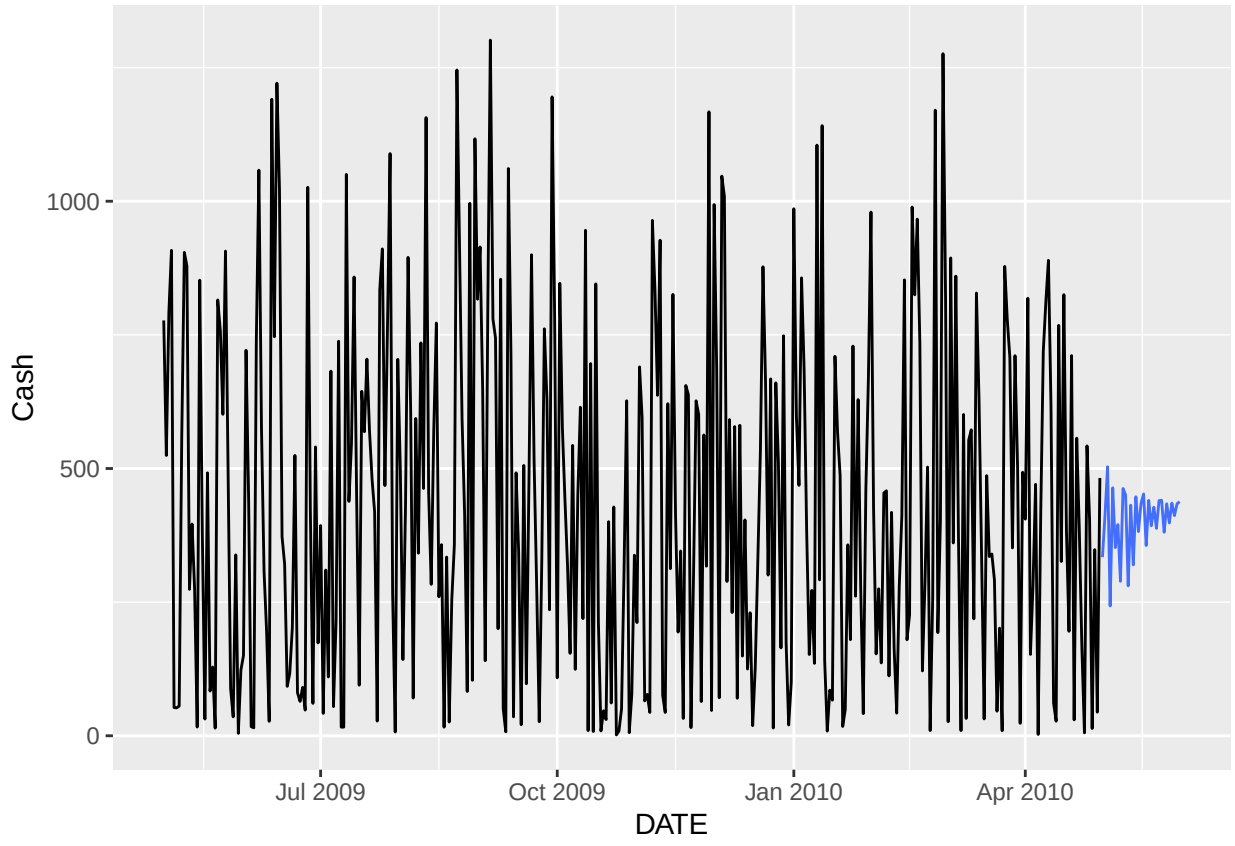
```
## Warning in report.mdl_df(.): Model reporting is only supported for individual
## models, so a glance will be shown. To see the report for a specific model, use
## `select()` and `filter()` to identify a single model.
```

```
## # A tibble: 6 x 12
##   ATM   .model  sigma2 log_lik   AIC  AICc   BIC ar_roots ma_roots    MSE   AMSE
##   <chr> <chr>    <dbl>   <dbl> <dbl> <dbl> <dbl> <list>   <list>    <dbl>  <dbl>
## 1 ATM4  arima  1.05e+5  -2626. 5260. 5260. 5275. <cpl>    <cpl>       NA     NA
## 2 ATM4  arima~ 4.00e+2  -1610. 3229. 3230. 3249. <cpl>    <cpl>       NA     NA
## 3 ATM4  seaso~ 1.85e+5     NA    NA    NA    NA  <NULL>   <NULL>      NA     NA
## 4 ATM4  seaso~ 6.73e+2     NA    NA    NA    NA  <NULL>   <NULL>      NA     NA
## 5 ATM4  ets    9.78e+4  -3169. 6359. 6359. 6398. <NULL>   <NULL>   95410. 95564.
## 6 ATM4  ets_t~ 2.68e-1  -2153. 4326. 4327. 4365. <NULL>   <NULL>     378.   380.
## # i 1 more variable: MAE <dbl>
```

lets try arima transformed and ets transformed predictions.

```
atm4_cleaned_arima_trans_forecasts <- atm4_cleaned %>%
  model(ARIMA(box_cox(Cash, atm4_cleaned_lambda))) %>%
  forecast(h=31)

atm4_cleaned_arima_trans_forecasts %>%
  autoplot(atm4_cleaned, level=NA)
```

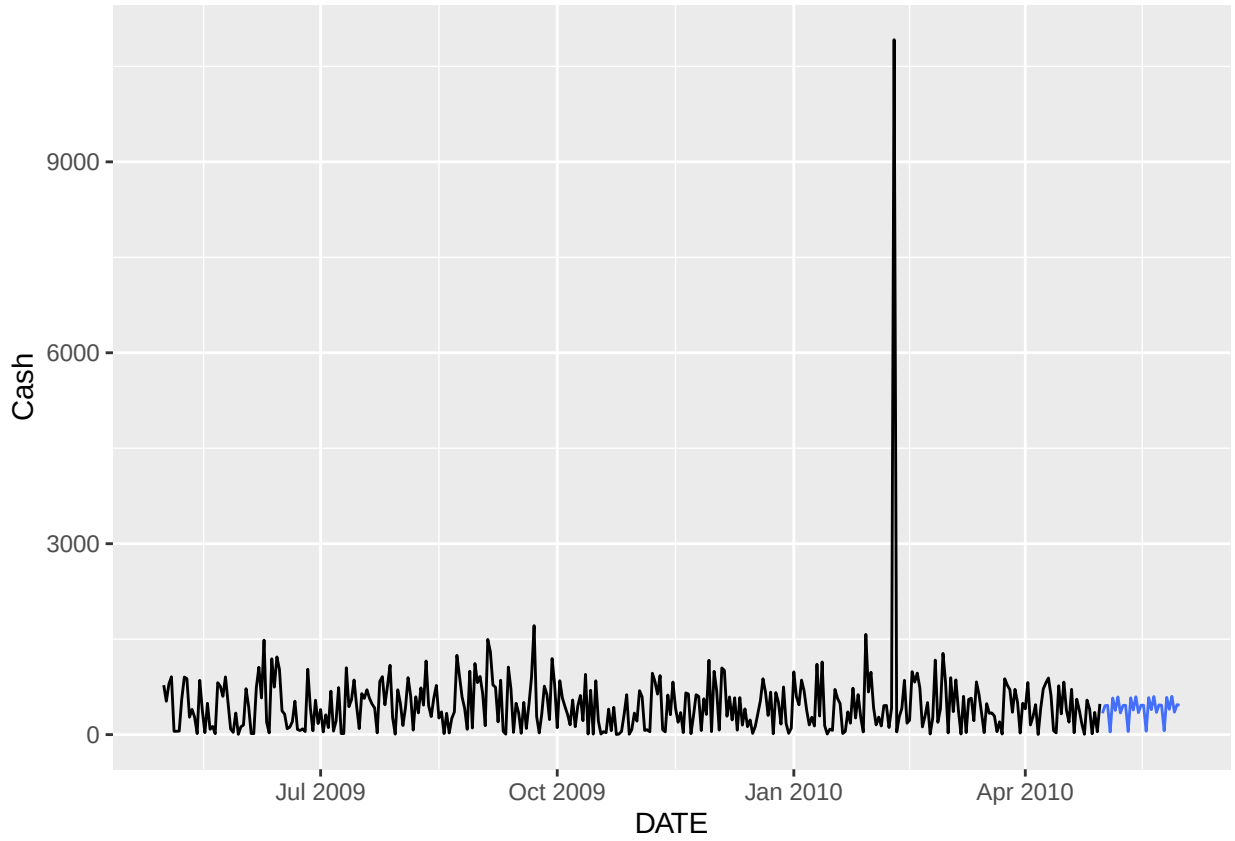
```
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
```



Still not amazing forecasts.

```
atm4_cleaned_ets_trans_forecasts <- atm4_cleaned %>%
  model(ETS(box_cox(Cash, atm4_cleaned_lambda))) %>%
  forecast(h=31)

atm4_cleaned_ets_trans_forecasts %>%
  autoplot(atm4, level=NA)
```

```
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
```




```
# Convert forecast to a data frame
atm4_forecast_df <- as_tibble(atm4_cleaned_ets_trans_forecasts)

# Write the forecast data to an Excel file
write_xlsx(atm4_forecast_df, "atm4_forecasts.xlsx")
```

These look much better lets use these as our final forecasts. ## Part 2

First load in the data

```
power_data <- read_excel("ResidentialCustomerForecastLoad-624.xlsx")
head(power_data)
```

```
## # A tibble: 6 x 3
```

```
##   CaseSequence `YYYY-MMM`     KWH
##          <dbl> <chr>        <dbl>
## 1          733 1998-Jan   6862583
## 2          734 1998-Feb   5838198
## 3          735 1998-Mar   5420658
## 4          736 1998-Apr   5010364
## 5          737 1998-May   4665377
## 6          738 1998-Jun   6467147
```

Clean the data and set the date column as a date with YearMonth

```
power_data <- power_data %>%
  mutate(Date = yearmonth(`YYYY-MMM`)) %>%
  select(Date, KWH)
```

```
head(power_data)
```

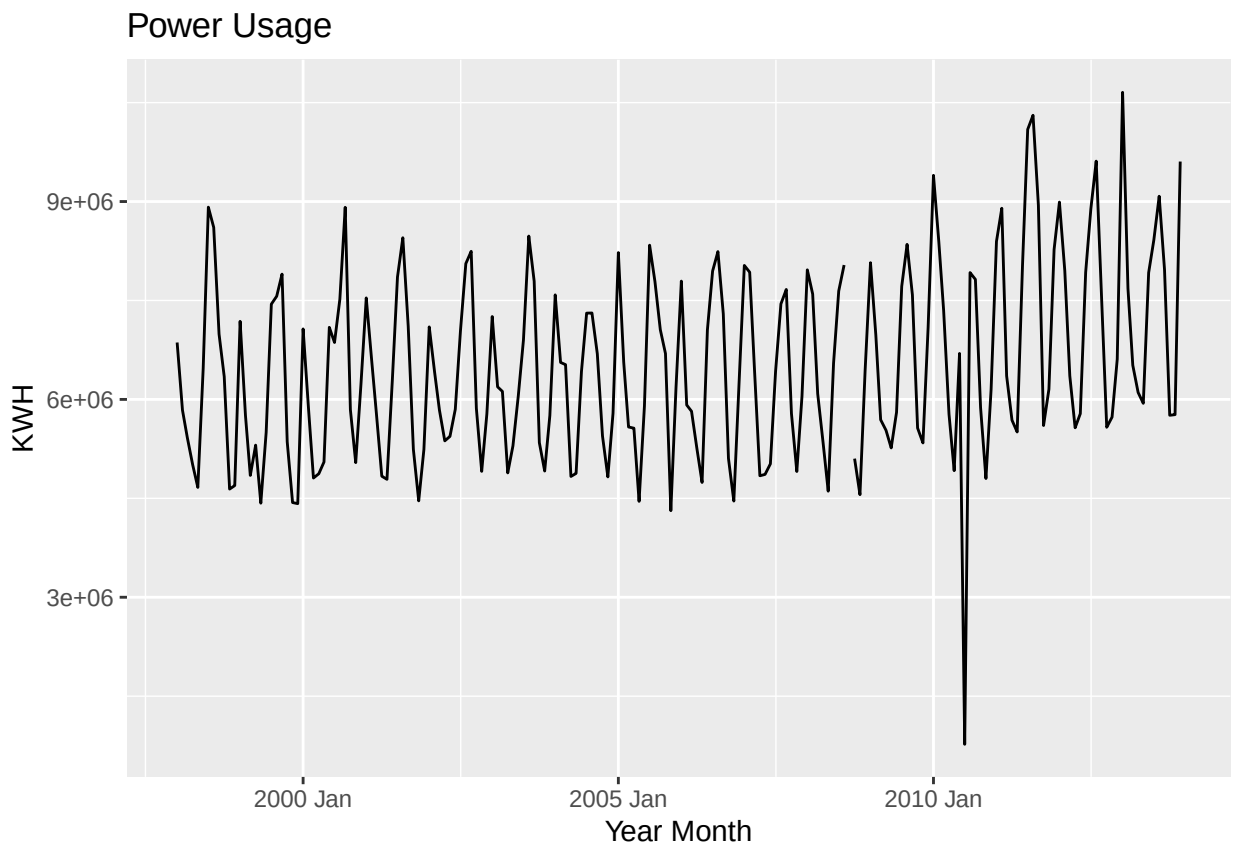
```
## # A tibble: 6 x 2
##       Date     KWH
##      <mth>   <dbl>
## 1 1998 Jan 6862583
## 2 1998 Feb 5838198
## 3 1998 Mar 5420658
## 4 1998 Apr 5010364
## 5 1998 May 4665377
## 6 1998 Jun 6467147
```

Convert the power data into a tstibble

```
power_data <- power_data %>%
  as_tsibble(
    index = Date
  )
head(power_data)
```

```
## # A tsibble: 6 x 2 [1M]
##       Date     KWH
##      <mth>   <dbl>
## 1 1998 Jan 6862583
## 2 1998 Feb 5838198
## 3 1998 Mar 5420658
## 4 1998 Apr 5010364
## 5 1998 May 4665377
## 6 1998 Jun 6467147
```

```
power_data %>%
  autoplot(KWH) +
  labs(title = "Power Usage", x="Year Month", y="KWH")
```



There appears to be some missing values lets see how many values are missing. We may need to impute these vaules. There also appears to be an outlier around 2010 that may need to be dealt with.

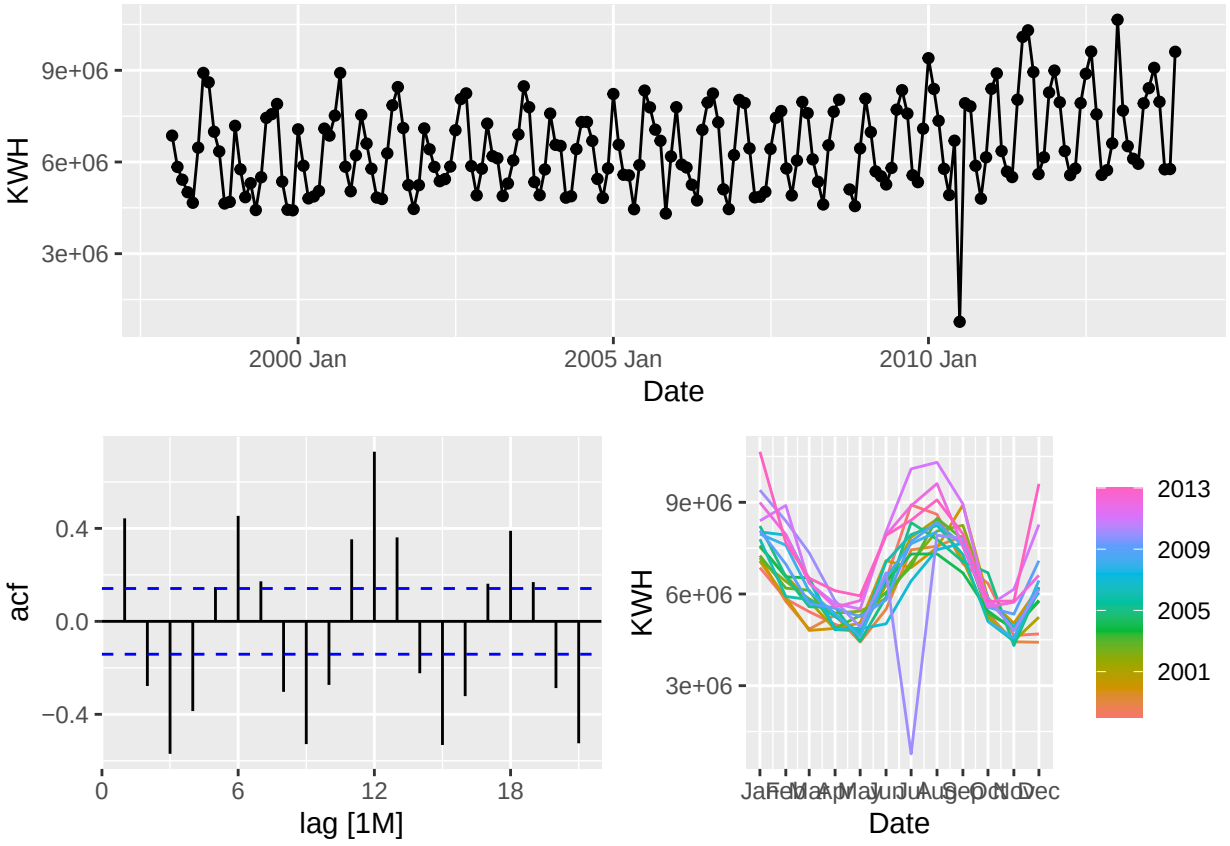
```
power_data %>%
  filter(if_any(everything(), is.na))
```

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

From the output above we can see that there is just one NA value in the time series in September 2008. We will impute this value lets check the residuals because there appears to be some seasonality in the data and if there is we can use the same method to impute the value as we did for the atm data.

```
power_data %>%
  gg_tsdisplay(KWH)
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_point()`).
```



From the output above we can see that there is clear seasonality so lets use the same approach as the ATM data to impute the missing value which will take the seasonality of the data into account.

```
power_data <- power_data %>%
  mutate(KWH = na_seadec(KWH, algorithm = "interpolation", find_frequency= TRUE))
```

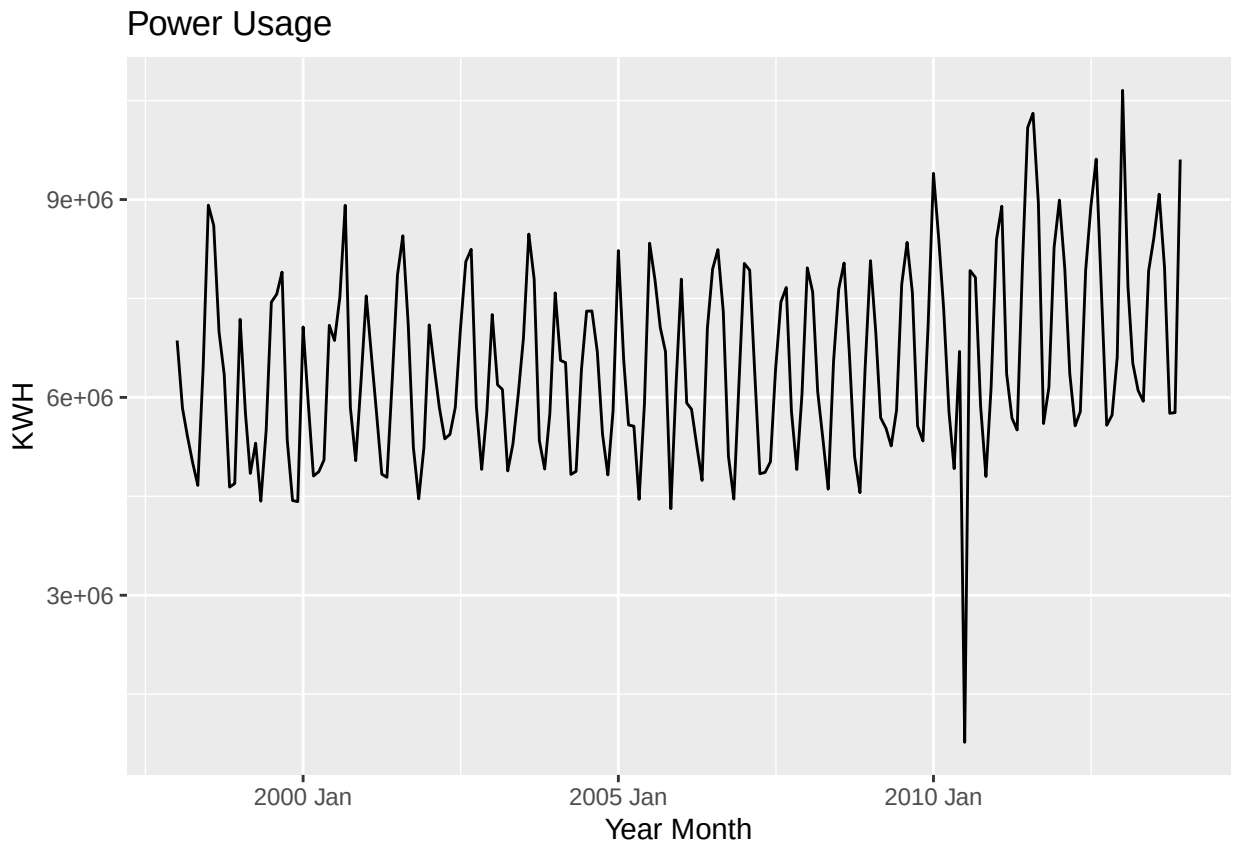
```
power_data %>%
  filter(if_any(everything(), is.na))
```

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

We can see now there's no more missing values lets look at the time plot to make sure the imputed data point is realistic.

```
power_data %>%
  autoplot(KWH)+
  labs(title = "Power Usage", x="Year Month", y="KWH")
```

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

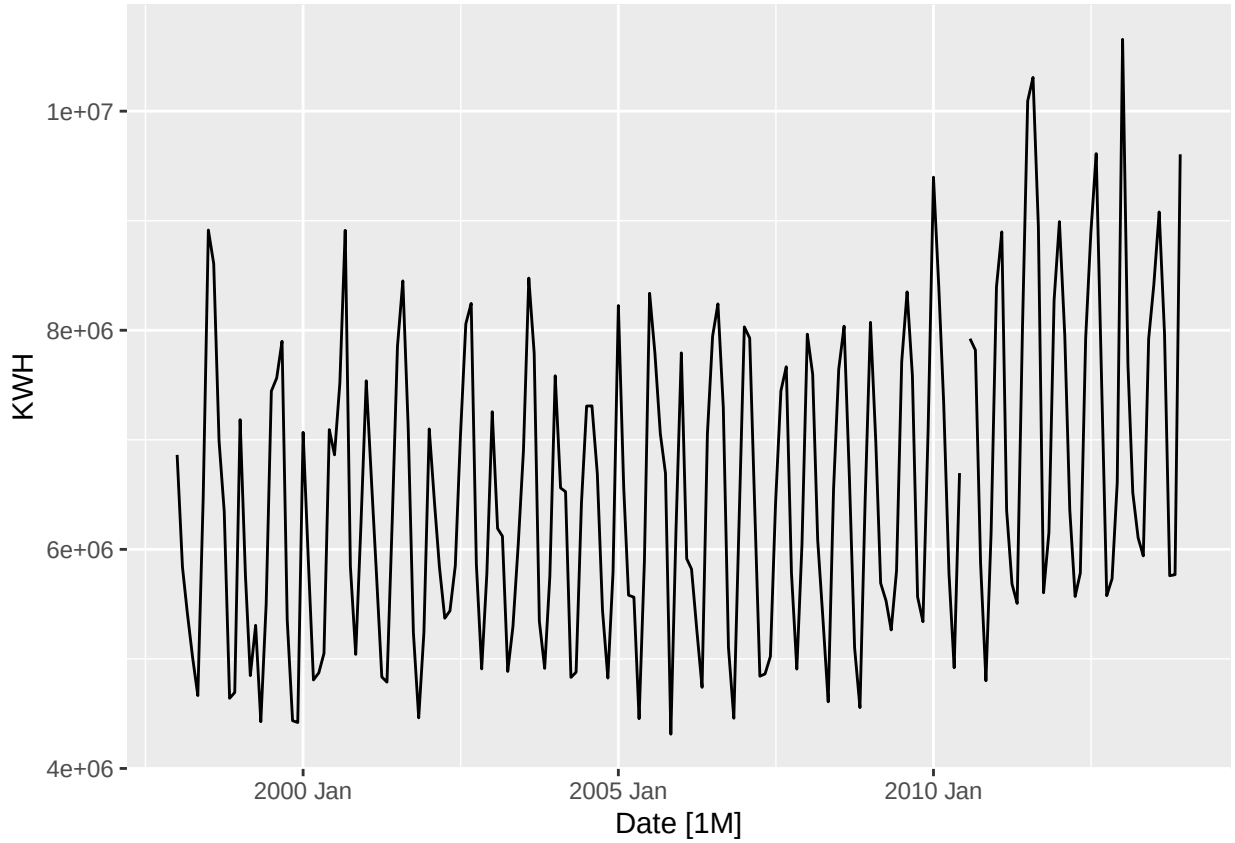


The imputed point looks to be realistic so now lets handle the outlier. I will remove and then impute the outlier using the same function as the missing data point.

```
power_data <- power_data %>%
  mutate(KWH = ifelse(KWH < 3e06, NA, KWH))
```

```
power_data %>%
  autoplot(KWH)
```

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

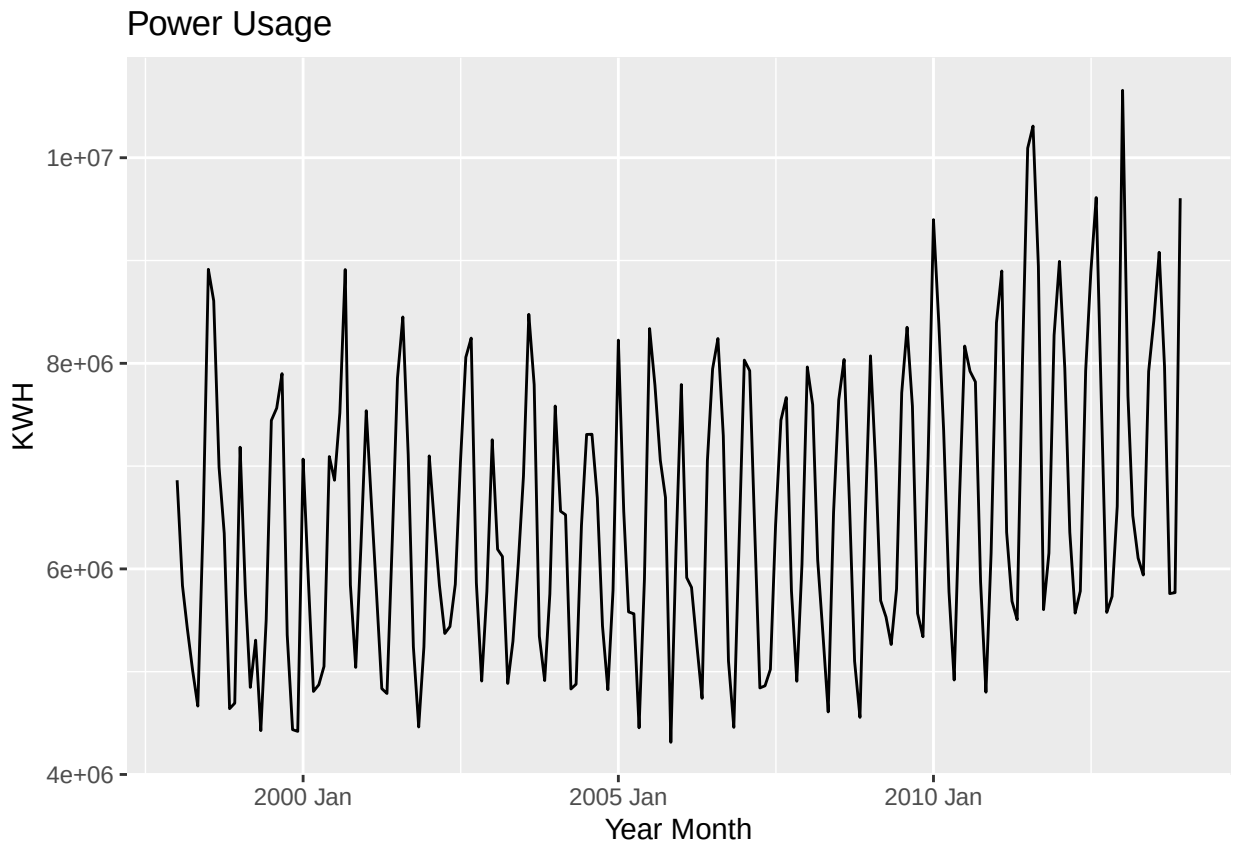


We can see now the outlier has been removed so lets impute it back in.

```
power_data <- power_data %>%
  mutate(KWH = na_seadec(KWH, algorithm = "interpolation", find_frequency= TRUE))
```

```
power_data %>%
  autoplot(KWH)+
  labs(title = "Power Usage", x="Year Month", y="KWH")
```

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



The outlier has now been removed and imputed to be more realistic now we have data that we can build models on.

Now lets check what the optimal lambda value is for a box-cox transformation and decide if a transformation would be useful or not.

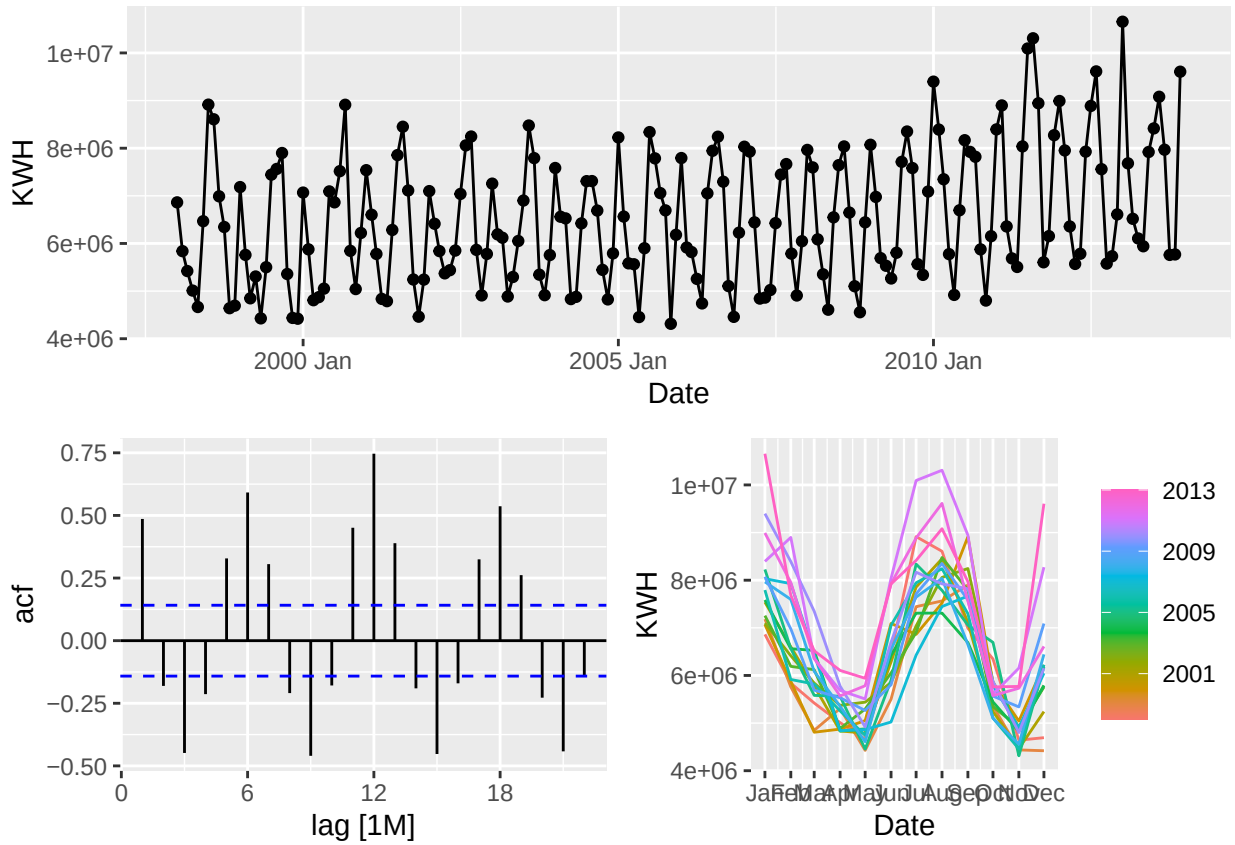
```
power_lambda <- power_data %>%
  features(KWH, features = guerrero) %>%
  pull(lambda_guerrero)
power_lambda
```

```
## [1] -0.217642
```

We get a lambda of -0.22 and looking at the time plot of the data the variance does seem to increase over time so a transformation may be useful. I will create models on both the non-transformed data and on box-cox transformed data.

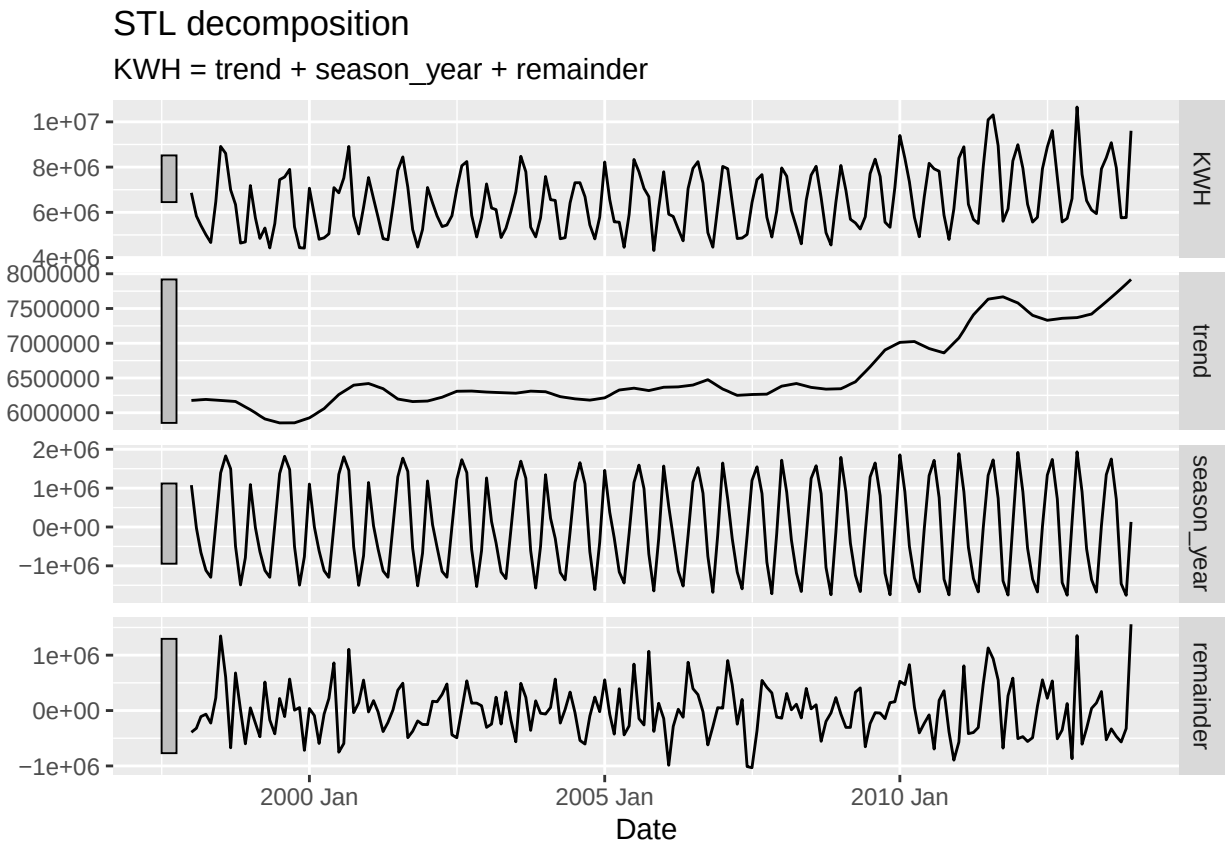
```
power_data %>%
  gg_tsdisplay(KWH)
```

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



The time plot there appears to be some seasonality and a very slight upward trend lets perform STL decomposition to get more info.

```
power_stl <- power_data %>%
  model(STL(KWH))

power_stl %>%
  components() %>%
  autoplot()
```



The STL decomposition shows that there is a much more positive trend than I initially thought. There is also clear seasonality in the data. This shows that the data is not stationary since there is a clear trend differencing will be needed for any ARIMA models.

Now lets try some models and evaluate them all.

```
power_models <- power_data %>%
  model(
    seasonal_naive = SNAIVE(KWH),
    ets = ETS(KWH),
    arima = ARIMA(KWH),
    seasonal_naive_trans = SNAIVE(box_cox(KWH, power_lambda)),
    ets_trans = ETS(box_cox(KWH, power_lambda)),
    arima_trans = ARIMA(box_cox(KWH, power_lambda))
  )

power_models %>%
  report()
```

```
## Warning in report.mdl_df(.): 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: 6 x 11
##   .model        sigma2 log_lik    AIC   AICc    BIC       MSE      AMSE      MAE
##   <chr>          <dbl>   <dbl>  <dbl>  <dbl>  <dbl>     <dbl>     <dbl>    <dbl>
## 1 seasonal_n~ 6.52e+11      NA     NA     NA     NA  NA         NA         NA
```

```
## 2 ets         9.25e- 3  -3056.  6142.  6145.  6191.  3.91e+11  4.13e+11  0.0735
## 3 arima       3.87e+11  -2657.  5331.  5332.  5356. NA        NA        NA
## 4 seasonal_n~ 1.52e- 5     NA     NA     NA     NA  NA        NA        NA
## 5 ets_trans   9.41e- 6    615. -1195. -1192. -1140.  8.63e- 6  9.05e- 6  0.00227
## 6 arima_trans 1.06e- 5    786. -1563. -1562. -1547. NA        NA        NA
## # i 2 more variables: ar_roots <list>, ma_roots <list>
```

```
power_ets <- power_data %>%
  model(ETS(box_cox(KWH, power_lambda)))

power_ets %>%
  report()
```

```
## Series: KWH
## Model: ETS(A,A,A)
## Transformation: box_cox(KWH, power_lambda)
##   Smoothing parameters:
##     alpha = 0.1102162
##     beta  = 0.000101183
##     gamma = 0.0002072429
##
##   Initial states:
##      l[0]         b[0]         s[0]        s[-1]        s[-2]       s[-3]
##  4.440435 4.135641e-05 -0.001605541 -0.009283762 -0.004254742 0.005238956
##        s[-4]      s[-5]        s[-6]        s[-7]       s[-8]        s[-9]
##  0.008655836 0.00702144 0.0007978052 -0.007997842 -0.00622126 -0.002385641
##     s[-10]      s[-11]
##  0.0027067 0.007328053
##
##   sigma^2:  0
##
##       AIC      AICc       BIC
## -1195.376 -1191.859 -1139.998
```

```
power_arima <- power_data %>%
  model(ARIMA(box_cox(KWH, power_lambda)))

power_arima %>%
  report()
```

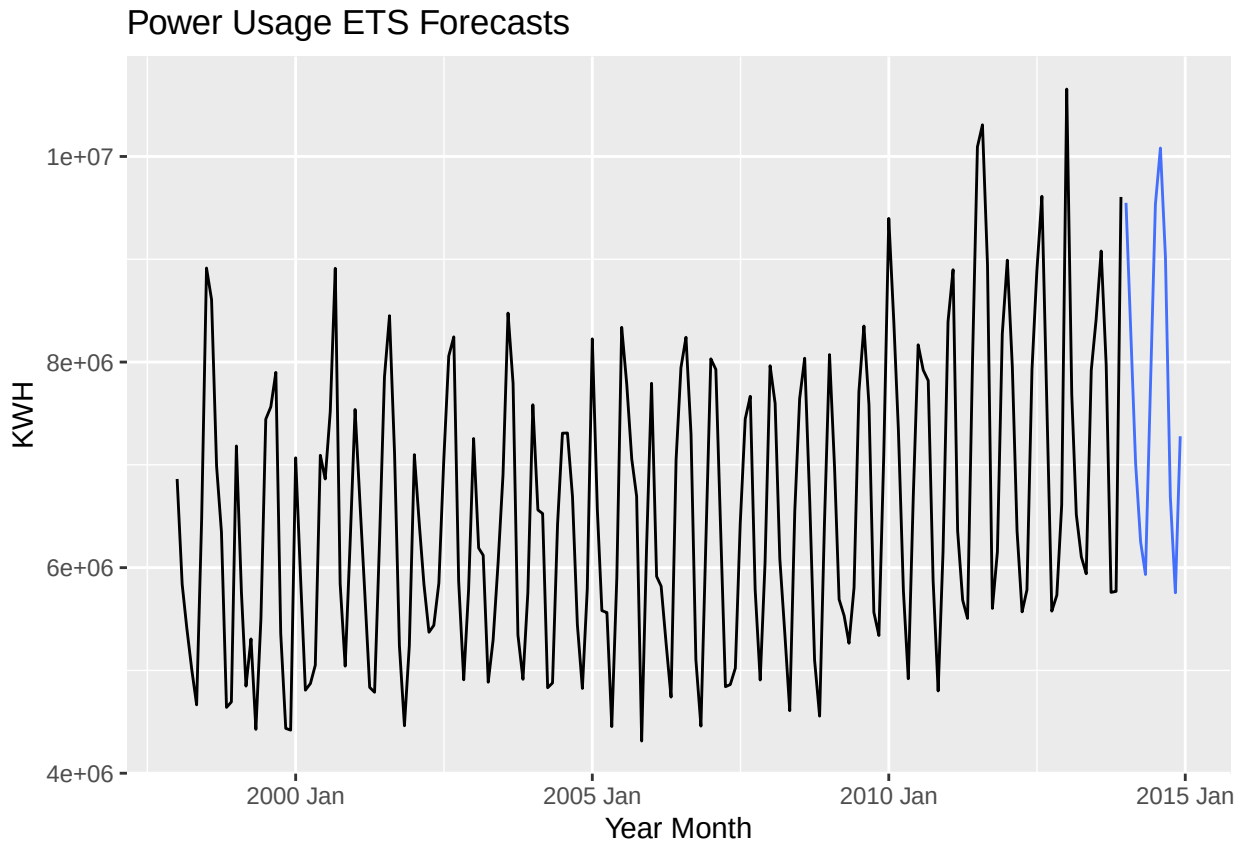
```
## Series: KWH
## Model: ARIMA(0,0,1)(2,1,0)[12] w/ drift
## Transformation: box_cox(KWH, power_lambda)
##
## Coefficients:
##          ma1     sar1     sar2  constant
##       0.2557  -0.7103  -0.3461    0.0011
## s.e.  0.0825   0.0751   0.0773    0.0003
##
## sigma^2 estimated as 1.059e-05:  log likelihood=786.33
## AIC=-1562.65   AICc=-1562.31   BIC=-1546.69
```

We can see that it auto estimated the ARIMA model to a ARIMA(0,0,1) model with yearly seasonality. This makes perfect sense as we can see the yearly seasonality in the time plots.

Now lets produce forecasts for the ETS model as it produced the best results.

```
power_ets_forecasts <- power_ets %>%
  forecast(h=12)
power_ets_forecasts %>%
  autoplot(power_data,level=NA) +
  labs(title = "Power Usage ETS Forecasts", x="Year Month", y="KWH")
```

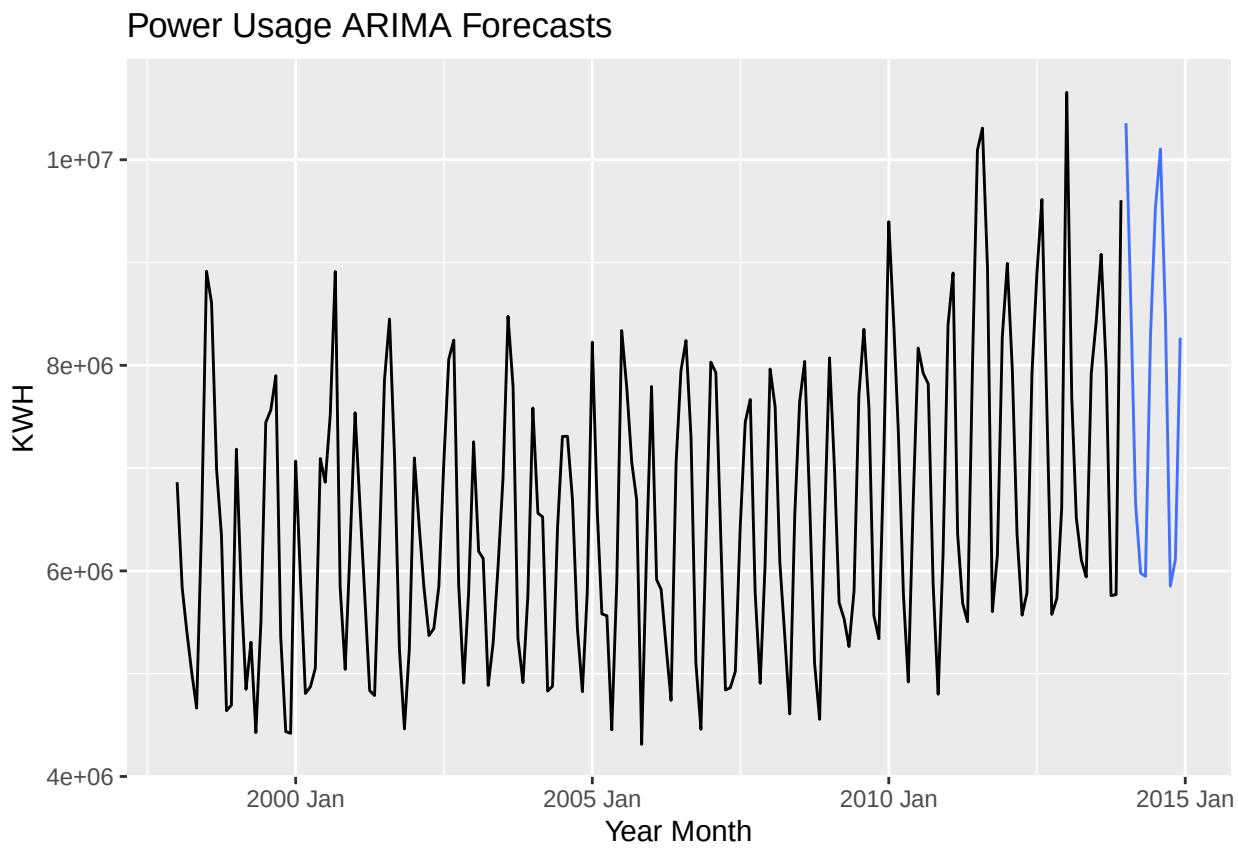
```
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
```



Lets also look at the ARIMA forecasts

```
power_arima_forecasts <- power_arima %>%
  forecast(h=12)
power_arima_forecasts %>%
  autoplot(power_data,level=NA) +
  labs(title = "Power Usage ARIMA Forecasts", x="Year Month", y="KWH")
```

```
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
```




```
# Convert forecast to a data frame
power_forecast_df <- as_tibble(power_arima_forecasts)

# Write the forecast data to an Excel file
write_xlsx(power_forecast_df, "power_forecasts.xlsx")
```

The ARIMA model seems to capture the peaks and valleys a little better so lets use that model for the forecasts.

Part C BONUS

Load in data

Load in the data

```
pipe_data1 <- read_excel("Waterflow_Pipe1.xlsx")
pipe_data2 <- read_excel("Waterflow_Pipe2.xlsx")
```

Use the convertToDateTime function to convert the dates to readable dates.

```
pipe_data1 <- pipe_data1 %>%
  mutate(timestamp = convertToDateTime(`Date Time`)) %>%
  select(timestamp, WaterFlow)
```

```
pipe_data2<- pipe_data2 %>%
  mutate(timestamp = convertToDateTime(`Date Time`))%>%
  select(timestamp, WaterFlow)
```

```
head(pipe_data1)
```

```
## # A tibble: 6 x 2
##   timestamp           WaterFlow
##   <dttm>                  <dbl>
## 1 2015-10-23 00:24:06     23.4
## 2 2015-10-23 00:40:02     28.0
## 3 2015-10-23 00:53:51     23.1
## 4 2015-10-23 00:55:40     30.0
## 5 2015-10-23 01:19:17      6.00
## 6 2015-10-23 01:23:58     15.9
```

```
head(pipe_data2)
```

```
## # A tibble: 6 x 2
##   timestamp           WaterFlow
##   <dttm>                  <dbl>
## 1 2015-10-23 01:00:00      18.8
## 2 2015-10-23 02:00:00      43.1
## 3 2015-10-23 03:00:00      38.0
## 4 2015-10-23 04:00:00      36.1
## 5 2015-10-23 05:00:00      31.9
## 6 2015-10-23 06:00:00      28.2
```

Now we have valid timestamps.

Next, time sequence the data combining rows that occur in the same hour taking the mean of all the flows.

```
pipe1_hourly <- pipe_data1 %>%
  mutate(timestamp = floor_date(timestamp, unit = "hour")) %>%
  group_by(timestamp) %>%
  summarise(mean_flow = mean(WaterFlow, na.rm = TRUE)) %>%
  ungroup()

pipe2_hourly <- pipe_data2 %>%
  mutate(timestamp = floor_date(timestamp, unit = "hour")) %>%
  group_by(timestamp) %>%
  summarise(mean_flow = mean(WaterFlow, na.rm = TRUE)) %>%
  ungroup()
```

```
head(pipe1_hourly)
```

```
## # A tibble: 6 x 2
##   timestamp           mean_flow
##   <dttm>                  <dbl>
## 1 2015-10-23 00:00:00      26.1
## 2 2015-10-23 01:00:00      18.9
```

```
## 3 2015-10-23 02:00:00      15.2
## 4 2015-10-23 03:00:00      23.1
## 5 2015-10-23 04:00:00      15.5
## 6 2015-10-23 05:00:00      22.7
```

```
head(pipe2_hourly)
```

```
## # A tibble: 6 x 2
##   timestamp           mean_flow
##   <dttm>                  <dbl>
## 1 2015-10-23 01:00:00      18.8
## 2 2015-10-23 02:00:00      43.1
## 3 2015-10-23 03:00:00      38.0
## 4 2015-10-23 04:00:00      36.1
## 5 2015-10-23 05:00:00      31.9
## 6 2015-10-23 06:00:00      28.2
```

Now we can combine the two data frames into one time series.

```
combined_hourly <- full_join(pipe1_hourly, pipe2_hourly, by = "timestamp", suffix = c("_pipe1", "_pipe2
  mutate(WaterFlow = rowMeans(select(., starts_with("mean_flow")), na.rm = TRUE)) %>%
  select(timestamp, WaterFlow)
```

```
head(combined_hourly)
```

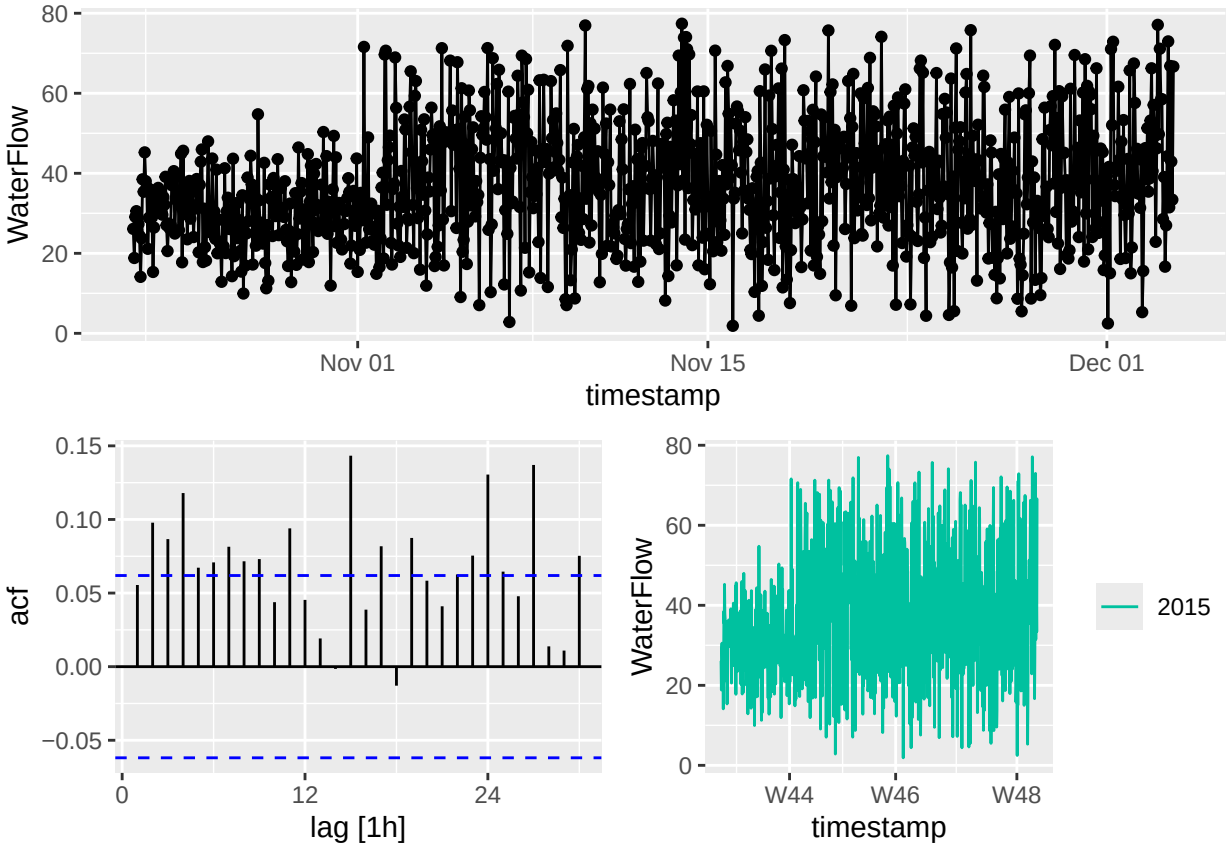
```
## # A tibble: 6 x 2
##   timestamp           WaterFlow
##   <dttm>                  <dbl>
## 1 2015-10-23 00:00:00      26.1
## 2 2015-10-23 01:00:00      18.8
## 3 2015-10-23 02:00:00      29.1
## 4 2015-10-23 03:00:00      30.5
## 5 2015-10-23 04:00:00      25.8
## 6 2015-10-23 05:00:00      27.3
```

Convert the data into a tsibble with the timestamp as the index

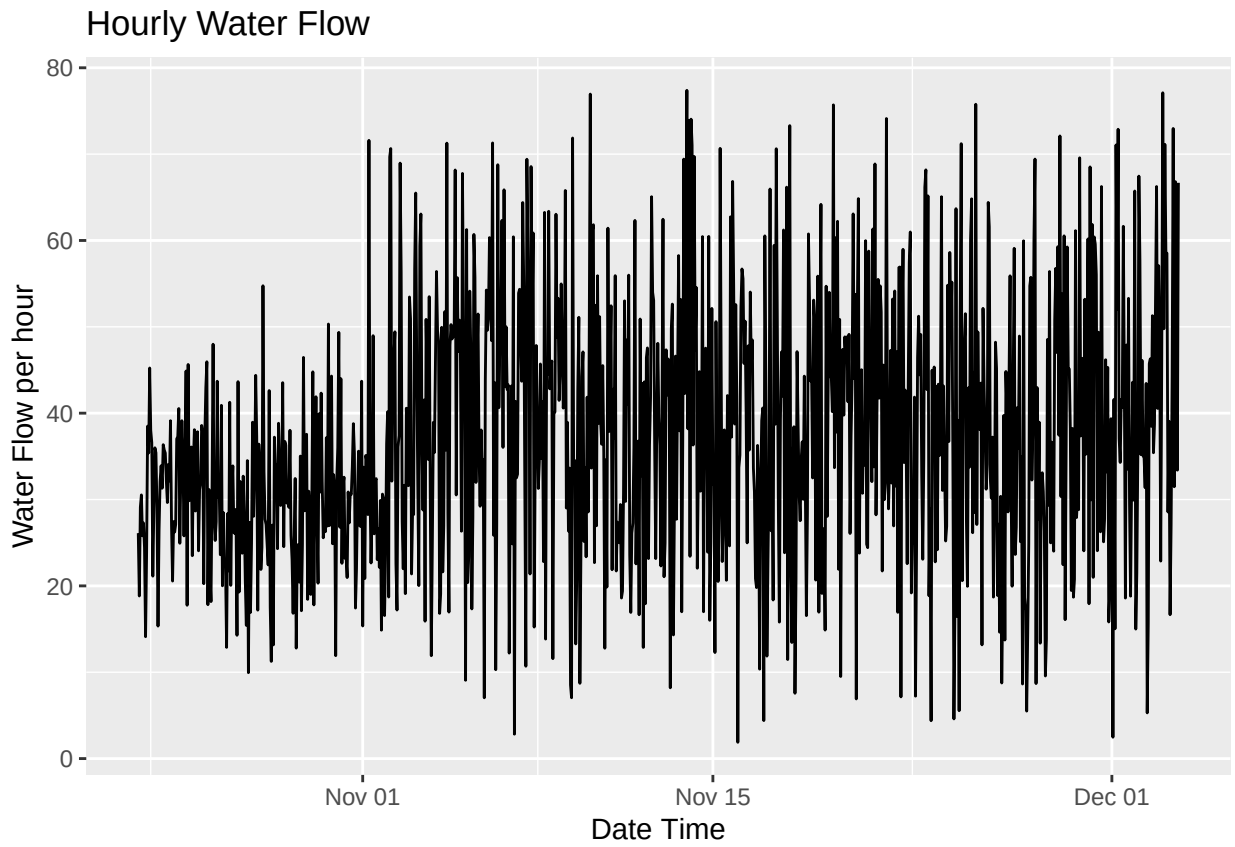
```
water_data <- combined_hourly %>%
  as_tsibble(
    index = timestamp
  )
```

Plot the data

```
water_data %>%
  gg_tsdisplay(WaterFlow)
```




```
water_data %>%
  autoplot(WaterFlow) +
  labs(title = "Hourly Water Flow", x="Date Time", y="Water Flow per hour")
```



Based on the plots above the water data does not appear to be stationary. There is an increase in variance as time goes on and the ACF plot shows that the lags do not quickly go to zero.

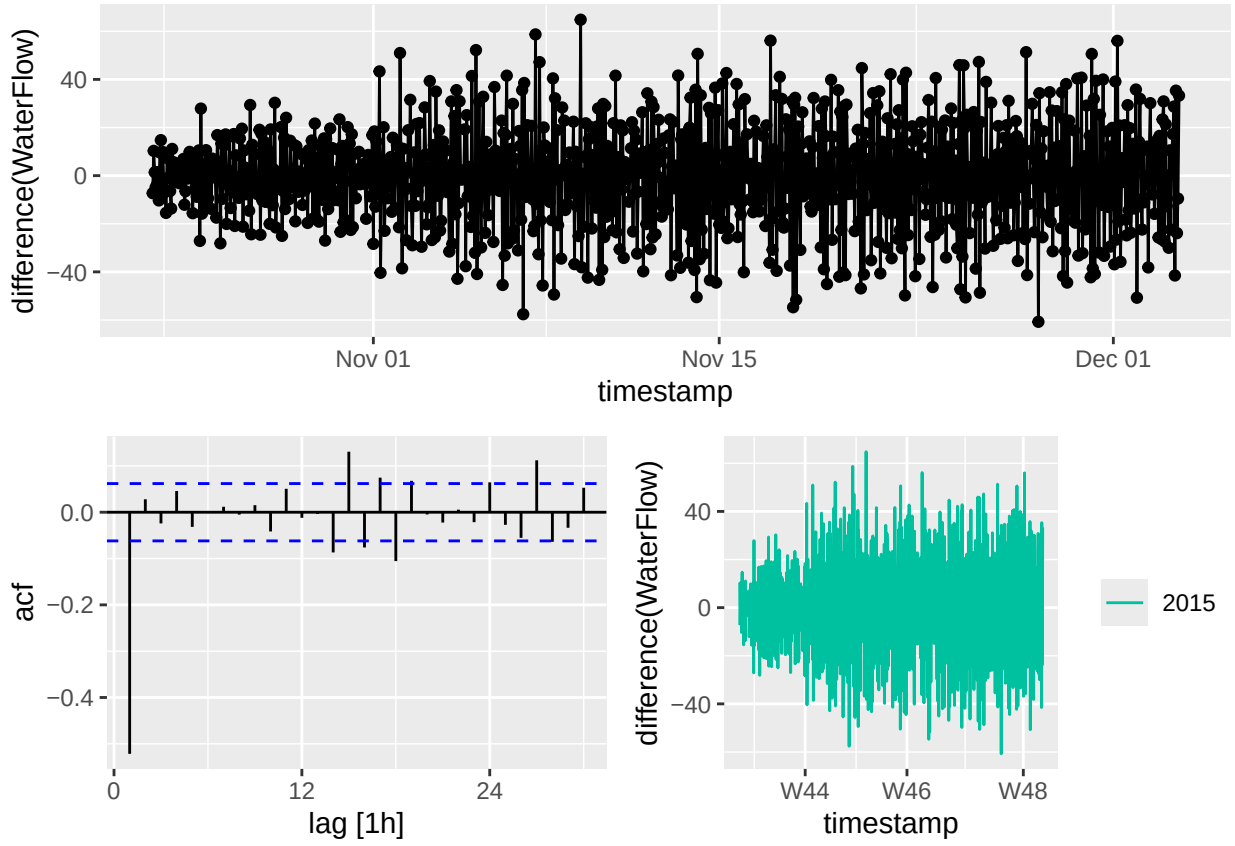
Lets difference the data and check if its stationary.

```
water_data %>%
  gg_tsdisplay(difference(WaterFlow))
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_point()`).
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
```



After the data is differenced it becomes stationary so an ARIMA model with first order differencing should be able to forecast. We can see there is mostly constant variance no trend and the lags in the ACF plot go to zero.

Lets check what the optimal lambda value would be for the data.

```
water_lambda <- water_data %>%
  features(WaterFlow, features = guerrero) %>%
  pull(lambda_guerrero)
water_lambda
```

```
## [1] -0.2164054
```

This lambda suggests that a transformation may be beneficial so lets try some models with the transformed data too.

```
water_arima <- water_data %>%
  model(
    ARIMA(WaterFlow)
  )
water_arima_trans <- water_data %>%
  model(ARIMA(box_cox(WaterFlow, water_lambda)))

report(water_arima)
```

```
## Series: WaterFlow
```

```
## Model: ARIMA(0,1,1)(0,0,1)[24]
##
## Coefficients:
##           ma1    sma1
##       -0.9735  0.0753
## s.e.   0.0078  0.0316
##
## sigma^2 estimated as 217.2:  log likelihood=-4113.85
## AIC=8233.71   AICc=8233.73   BIC=8248.44
```

```
report(water_arima_trans)
```

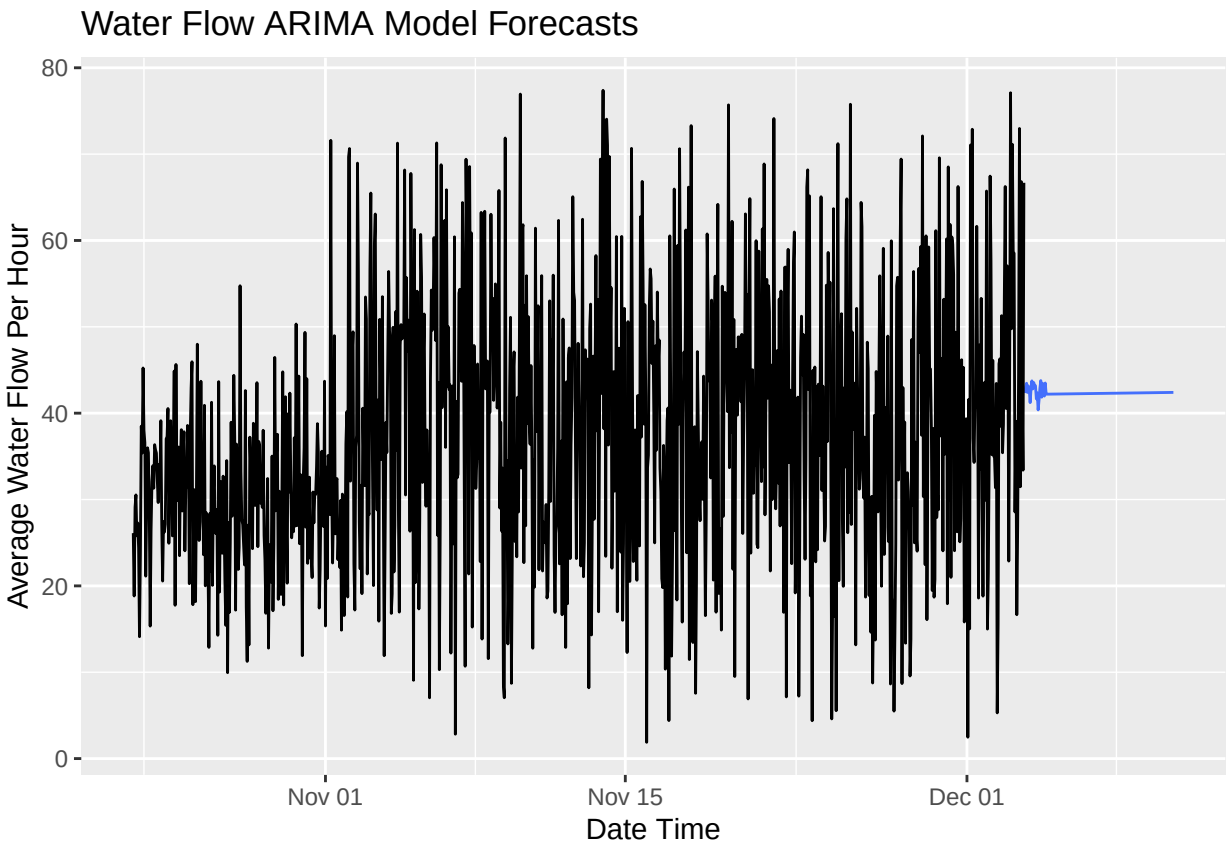
```
## Series: WaterFlow
## Model: ARIMA(0,1,1)(0,0,1)[24]
## Transformation: box_cox(WaterFlow, water_lambda)
##
## Coefficients:
##           ma1    sma1
##       -0.9857  0.0495
## s.e.   0.0067  0.0314
##
## sigma^2 estimated as 0.06157:  log likelihood=-25.97
## AIC=57.93   AICc=57.96   BIC=72.66
```

Looking at the results above we can see that the transformed ARIMA model that auto fitted to a ARIMA(0,1,1)(0,0,1)[24] model gave the best results. With an sigma^2 of 0.06157, AIC of 57.93, AICc of 57.96 and a BIC of 72.66

Lets produce forecasts for the model.

```
water_forecasts <- water_arima_trans %>%
  forecast(h=168)

water_forecasts %>%
  autoplot(water_data, level= NA) +
  labs(title="Water Flow ARIMA Model Forecasts", x="Date Time", y="Average Water Flow Per Hour")
```

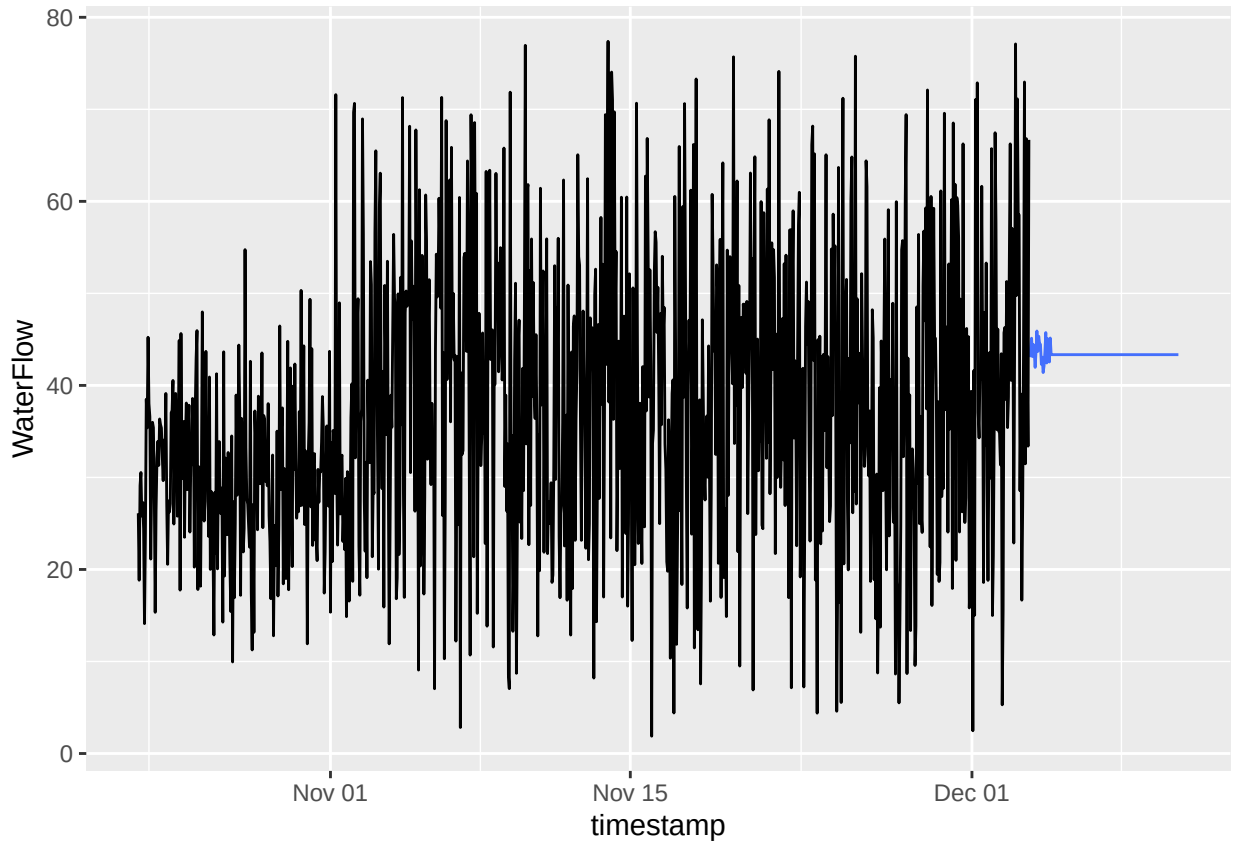
```
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
```



We make forecasts for 168 hours or one week forward the don't look too great. Lets try another model to see if we can improve.

```
water_arima_forecasts <- water_arima %>%
  forecast(h=168)

water_arima_forecasts %>%
  autoplot(water_data, level=NA)
```

```
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
```



These seem to have the same issues lets try an ETS model.

```
water_ets_models <- water_data %>%
  model(
    ets = ETS(WaterFlow),
    ets_trans = ETS(box_cox(WaterFlow,water_lambda))
  )

water_ets_models %>%
  report()
```

```
## Warning in report.mdl_df(.): 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.
```

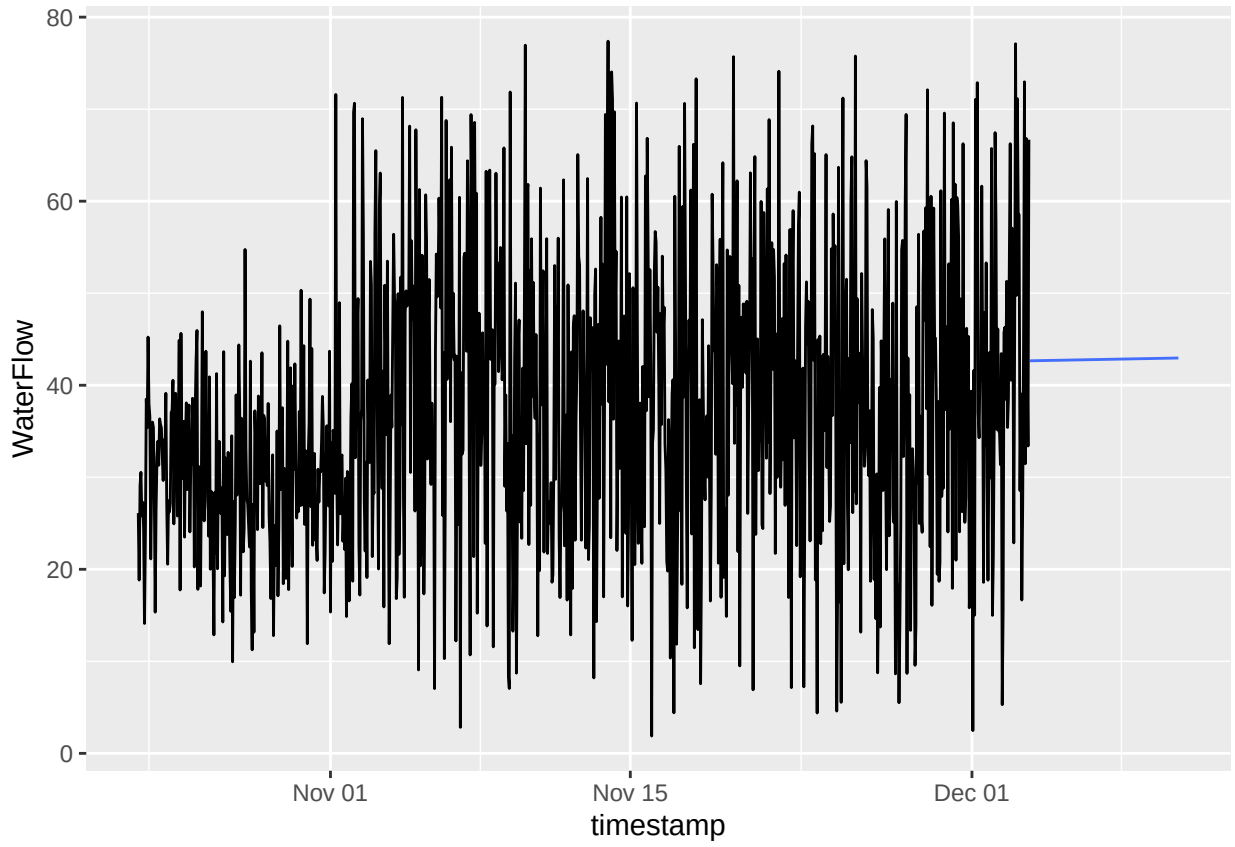
```
## # A tibble: 2 x 9
##   .model    sigma2 log_lik    AIC   AICc    BIC      MSE     AMSE    MAE
##   <chr>      <dbl>   <dbl>  <dbl>  <dbl>  <dbl>    <dbl>    <dbl>  <dbl>
## 1 ets       0.157   -6130. 12265. 12265. 12280. 218.     218.     0.316
## 2 ets_trans 0.0102  -2059.  4123.  4123.  4138.   0.0616   0.0616 0.0715
```

Lets check the forecasts for the transformed ETS model.

```
water_ets_forecasts <- water_data %>%
  model(ETS(box_cox(WaterFlow, water_lambda))) %>%
```

```
  forecast(h=168)

water_ets_forecasts %>%
  autoplot(water_data, level=NA)
```

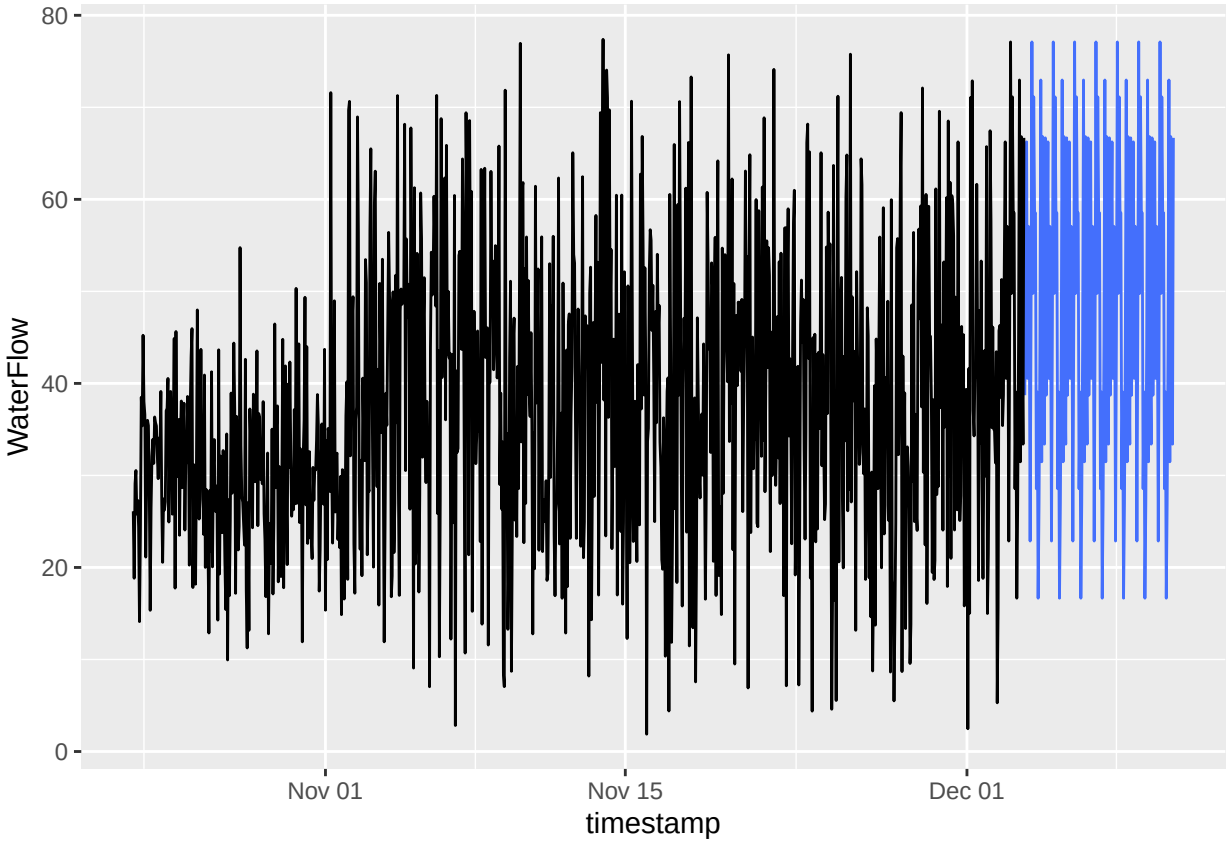
```
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
```



These forecasts do not appear to be much better. Lets try a seasonal naive model

```
water_seasonal_naive_forecast <- water_data %>%
  model(SNAIVE(WaterFlow)) %>%
  forecast(h=168)

water_seasonal_naive_forecast %>%
  autoplot(water_data, level=NA)
```

```
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
```




```
# Convert forecast to a data frame
water_forecast_df <- as_tibble(water_seasonal_naive_forecast)

# Write the forecast data to an Excel file
write_xlsx(water_forecast_df, "water_forecasts.xlsx")
```

These forecasts appear much better I will use these as my final forecasts.