DATA 624 Project 1: Forecasting Techniques Review

Kevin Kirby

2024-10-22

```
library(readxl)
library(httr)
library(tidyverse)
library(ggplot2)
library(dplyr)
library(fpp3)
library(prophet)
library(fable)
```

Overview

This is project one of the Fall 2024 edition of DATA 624: Forecasting and Predictive Analytics. There are three assigned parts, all of which use data provided in Excel files. Parts A and B are required, part C is optional.

- A. Forecast how much cash is taken out of 4 different ATM machines in May 2010
- B. Produce monthly forecasts for power usage for calendar year 2014
- C. Create time-base sequences for provide data aggregate based on hour; decide if the data is stationary; provide week-forward forecast if possible.

I have opted against the optional component and will do A and B.

To meet the requirement that the data be made available to you in Excel-readable foremast, clicking the Google API links below for each file will begin an automatic download of the respective file.

```
atm_624_url <- "https://storage.googleapis.com/data_science_masters_files/2024_fall/data_624_prediction_

GET(atm_624_url, write_disk(tf <- tempfile(fileext = ".xlsx")))
```

```
atm_624_data <- read_excel(tf)
atm_624_hold <- read_excel(tf)

residential_customer_url <- "https://storage.googleapis.com/data_science_masters_files/2024_fall/data_6

GET(residential_customer_url, write_disk(tf <- tempfile(fileext = ".xlsx")))
```

```
res_cust_forecast_load <- read_excel(tf)
res_cust_forecast_hold <- read_excel(tf)

waterflow_p1_url <- "https://storage.googleapis.com/data_science_masters_files/2024_fall/data_624_predi

GET(waterflow_p1_url, write_disk(tf <- tempfile(fileext = ".xlsx")))
```

```
waterflow_pipe1 <- read_excel(tf)
```

```
waterflow_p2_url <- "https://storage.googleapis.com/data_science_masters_files/2024_fall/data_624_predi

GET(waterflow_p2_url, write_disk(tf <- tempfile(fileext = ".xlsx")))
```

```
waterflow_pipe2 <- read_excel(tf)
```

A. Forecast how much cash is taken out of 4 different ATM machines in May 2010

The assignment states: "In part A, I want you to forecast how much cash is taken out of 4 different ATM machines for May 2010. The data is given in a single file. The variable 'Cash' is provided in hundreds of dollars, other than that it is straight forward. I am being somewhat ambiguous on purpose to make this have a little more business feeling. Explain and demonstrate your process, techniques used and not used, and your actual forecast.I am giving you data via an excel file, please provide your written report on your findings, visuals, discussion and your R code via an RPubs link along with the actual.rmd file Also please submit the forecast which you will put in an Excel readable file."

Forecasts I plan to use:

ATM1 and ATM2: STL decomposition with ARIMA

ATM3: Random walk with drift

ATM4: STL decomposition with ARIMA

Answer: There are three fields in this dataset: * Date: this came in as a numeric character and the values are the numeric representation of the underlying date * ATM: string character with values like "ATM1", "ATM2". Since the assignment says four ATMs, I expect four different values. * Cash: numeric integer character with random values like "89" and 111

```
glimpse(atm_624_data)
```

```
## Rows: 1,474
## Columns: 3
## $ DATE <dbl> 39934, 39934, 39935, 39935, 39936, 39936, 39937, 39937, 39938, 39~
## $ ATM  <chr> "ATM1", "ATM2", "ATM1", "ATM2", "ATM1", "ATM2", "ATM1", "ATM2", "~
## $ Cash <dbl> 96, 107, 82, 89, 85, 90, 90, 55, 99, 79, 88, 19, 8, 2, 104, 103, ~
```

Tidying the data

There are a few housekeeping items I need to do before going further: * Convert the DATE column to be an actual date instead of the Excel representation of a date * Change all column names to lowercase because you generally don't need uppercase letters until it's time to show the date to an end-user * See if there's missing data and, if so, resolve

```
atm_624_data$DATE <- as.Date(atm_624_data$DATE, origin = "1899-12-30")
colnames(atm_624_data) <- tolower(colnames(atm_624_data))

glimpse(atm_624_data)
```

```
## Rows: 1,474
## Columns: 3
## $ date <date> 2009-05-01, 2009-05-01, 2009-05-02, 2009-05-02, 2009-05-03, 2009~
## $ atm  <chr> "ATM1", "ATM2", "ATM1", "ATM2", "ATM1", "ATM2", "ATM1", "ATM2", "~
## $ cash <dbl> 96, 107, 82, 89, 85, 90, 90, 55, 99, 79, 88, 19, 8, 2, 104, 103, ~
```

```
summary(atm_624_data)
```

```
##       date                atm                cash
##  Min.   :2009-05-01   Length:1474        Min.   :    0.0
##  1st Qu.:2009-08-01   Class :character   1st Qu.:    0.5
##  Median :2009-11-01   Mode  :character   Median :   73.0
##  Mean   :2009-10-31                      Mean   :  155.6
##  3rd Qu.:2010-02-01                      3rd Qu.:  114.0
##  Max.   :2010-05-14                      Max.   :10919.8
##                                          NA's   :19
```

```
atm_na_subset <- atm_624_data[rowSums(is.na(atm_624_data)) > 0, ]

atm_na_subset
```

Handling outliers and missing values

```
## # A tibble: 19 x 3
##    date       atm    cash
##    <date>     <chr> <dbl>
##  1 2009-06-13 ATM1     NA
##  2 2009-06-16 ATM1     NA
##  3 2009-06-18 ATM2     NA
##  4 2009-06-22 ATM1     NA
##  5 2009-06-24 ATM2     NA
##  6 2010-05-01 <NA>     NA
##  7 2010-05-02 <NA>     NA
##  8 2010-05-03 <NA>     NA
##  9 2010-05-04 <NA>     NA
## 10 2010-05-05 <NA>     NA
## 11 2010-05-06 <NA>     NA
## 12 2010-05-07 <NA>     NA
## 13 2010-05-08 <NA>     NA
## 14 2010-05-09 <NA>     NA
## 15 2010-05-10 <NA>     NA
## 16 2010-05-11 <NA>     NA
## 17 2010-05-12 <NA>     NA
## 18 2010-05-13 <NA>     NA
## 19 2010-05-14 <NA>     NA
```

I'm going to drop the rows where it's NA for both ATM and cash since I don't believe imputting values just based on date is a great approach.

```
atm_624_data <- atm_624_data[!(is.na(atm_624_data$cash) & is.na(atm_624_data$atm)), ]

cash_na_subset <- atm_624_data[rowSums(is.na(atm_624_data)) > 0, ]

cash_na_subset
```

```
## # A tibble: 5 x 3
##   date       atm    cash
##   <date>     <chr> <dbl>
## 1 2009-06-13 ATM1     NA
## 2 2009-06-16 ATM1     NA
## 3 2009-06-18 ATM2     NA
## 4 2009-06-22 ATM1     NA
## 5 2009-06-24 ATM2     NA
```

I'll return to this further down. I need to check the rest of the data before deciding on an NA strategy.

```
atm_624_presum <- atm_624_data %>%
  group_by(atm) %>%
  summarise(
    total_values = n(),
    min_value = min(cash, na.rm = TRUE),
    mean_value = mean(cash, na.rm = TRUE),
    median_value = median(cash, na.rm = TRUE),
    max_value = max(cash, na.rm = TRUE)
  )

atm_624_presum
```

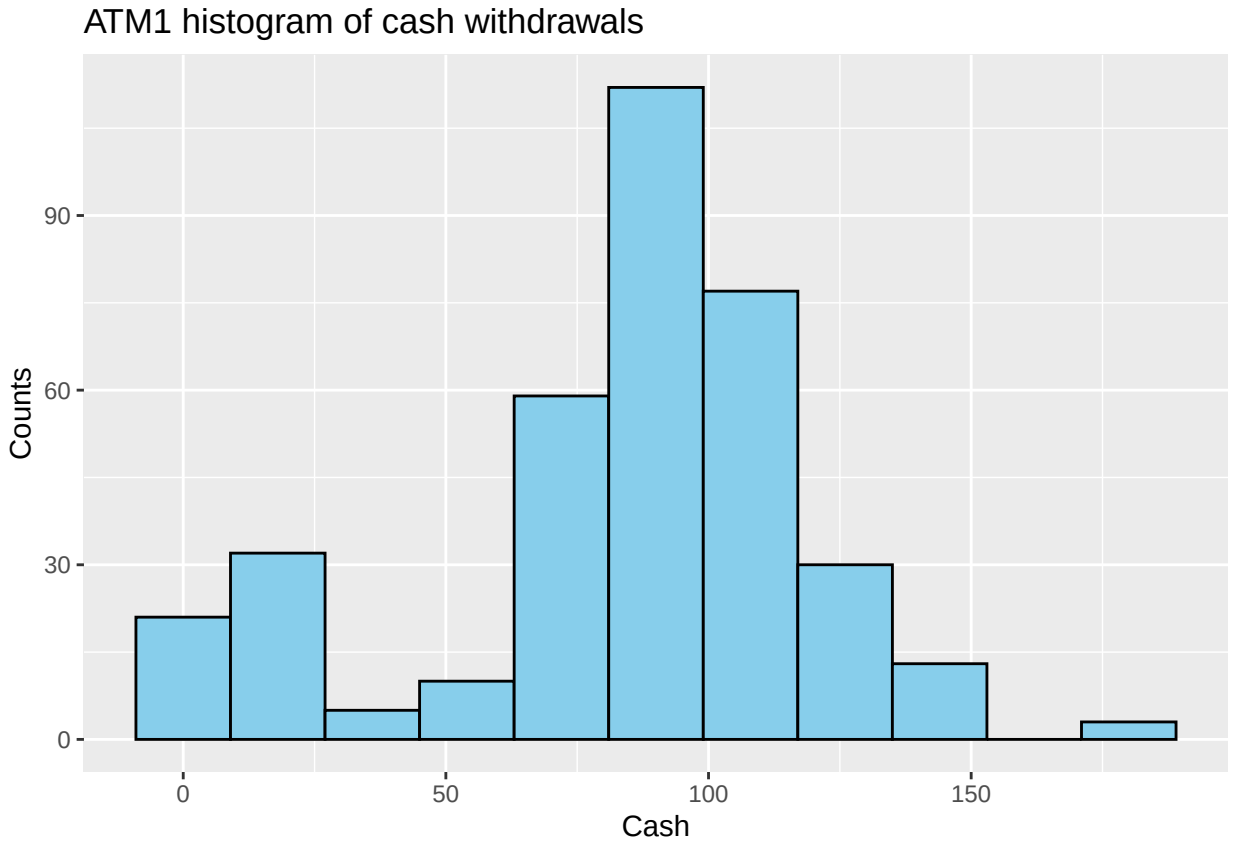
```
## # A tibble: 4 x 6
##   atm   total_values min_value mean_value median_value max_value
##   <chr>        <int>     <dbl>      <dbl>        <dbl>     <dbl>
## 1 ATM1           365      1        83.9           91       180
## 2 ATM2           365      0        62.6           67       147
## 3 ATM3           365      0         0.721          0        96
## 4 ATM4           365      1.56    474.           404.    10920.
```

```
sturges_bin_width <- function(x) {
  (max(x, na.rm = TRUE) - min(x, na.rm = TRUE)) / (1 + 3.3 * log10(length(x[!is.na(x)])))
}

atm_624_data <- atm_624_data %>%
  group_by(atm) %>%
  mutate(sturges_bw = sturges_bin_width(cash)) %>%
  ungroup()

atm_624_sturges <- atm_624_data %>%
  group_by(atm) %>%
  summarise(
    total_values = n(),
    min_value = min(cash, na.rm = TRUE),
    mean_value = mean(cash, na.rm = TRUE),
    median_value = median(cash, na.rm = TRUE),
    max_value = max(cash, na.rm = TRUE),
    sturges_bw = first(sturges_bw)
  )

atm_624_sturges
```

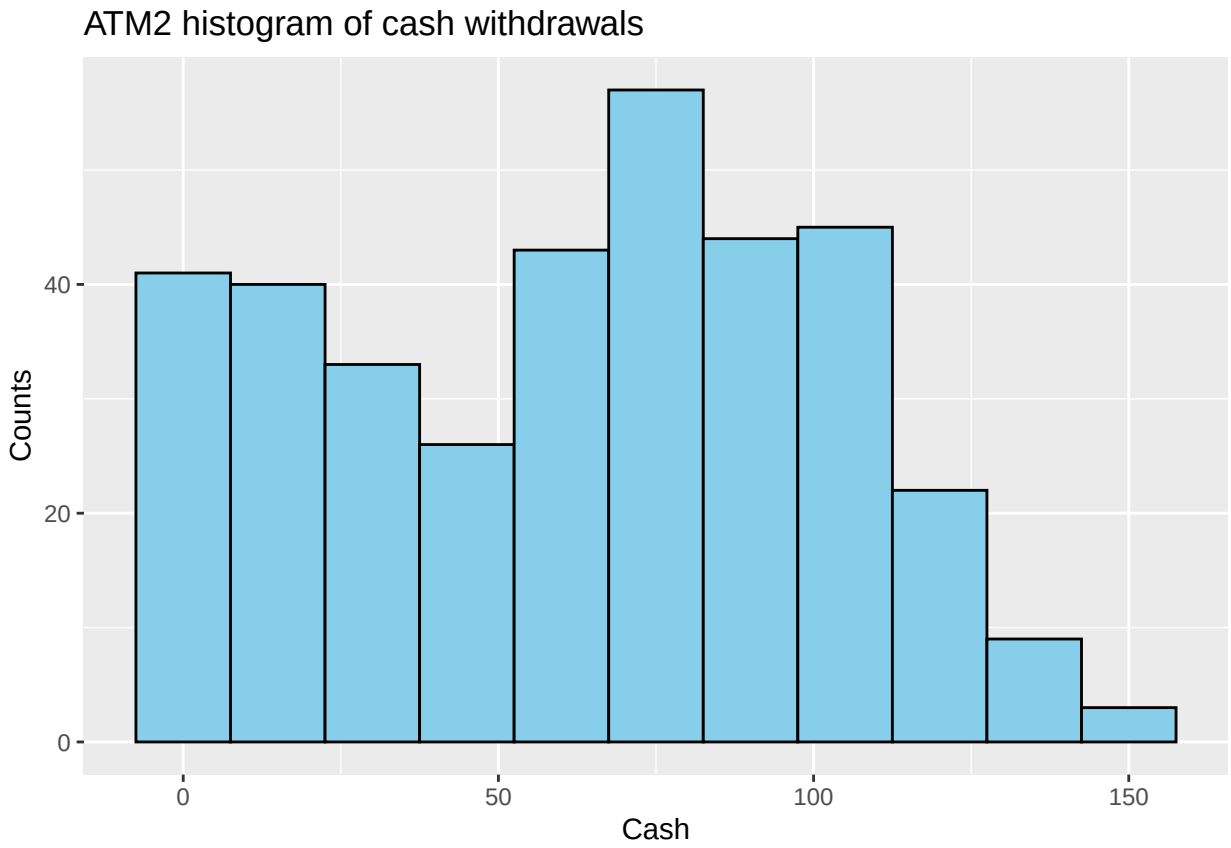
```
## # A tibble: 4 x 7
##   atm   total_values min_value mean_value median_value max_value sturges_bw
##   <chr>        <int>     <dbl>      <dbl>        <dbl>     <dbl>      <dbl>
## 1 ATM1           365      1        83.9           91       180        19.0
## 2 ATM2           365      0        62.6           67       147        15.6
## 3 ATM3           365      0         0.721          0        96        10.2
## 4 ATM4           365      1.56    474.           404.    10920.     1155.
```

```
sturge_histo <- function(data, atm, binwidth, title) {
  data %>%
    filter(!is.na(cash)) %>%
    filter(atm == !!atm) %>%
```

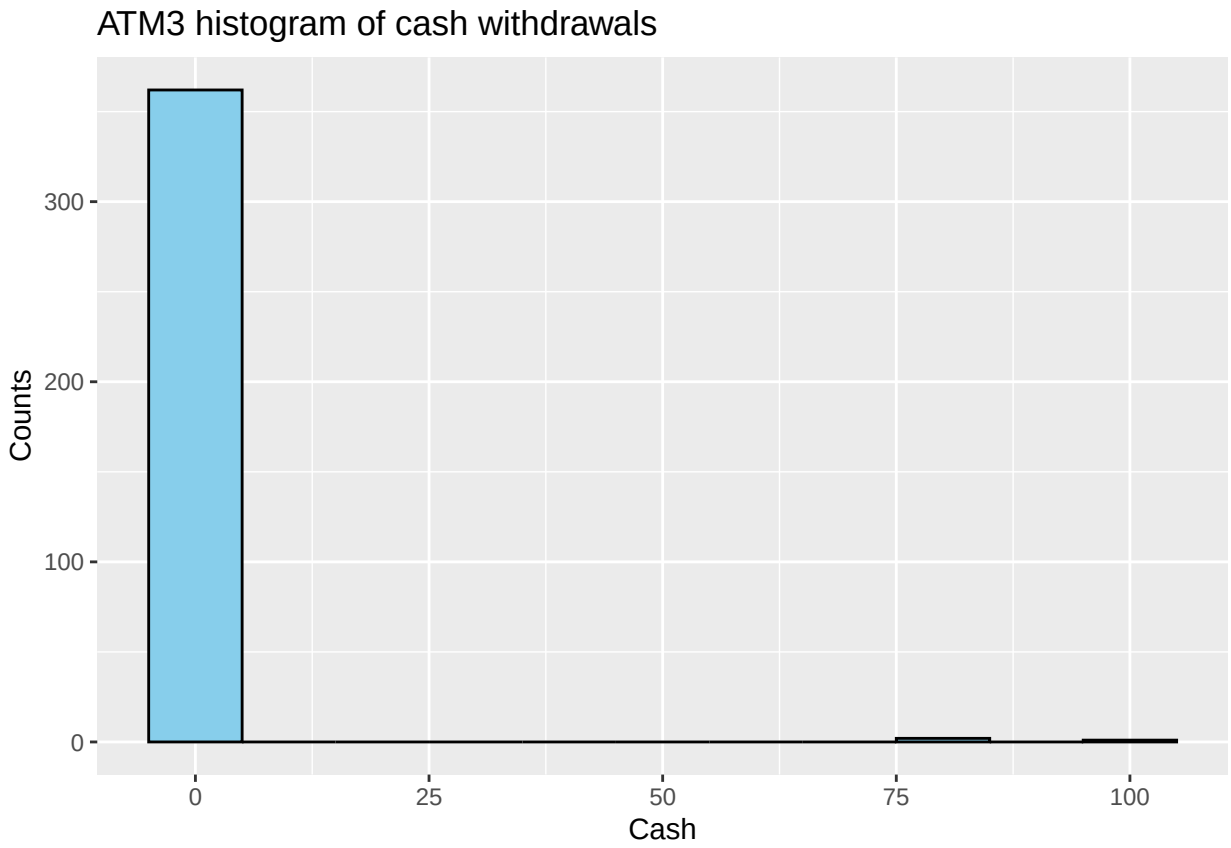
```
    ggplot(aes(x = cash)) +
    geom_histogram(binwidth = binwidth,
                   fill = "skyblue",
                   color = "black") +
    labs(title = title,
         x = "Cash",
         y = "Counts")
}

sturge_histo(atm_624_data, "ATM1", 18, "ATM1 histogram of cash withdrawals")
```



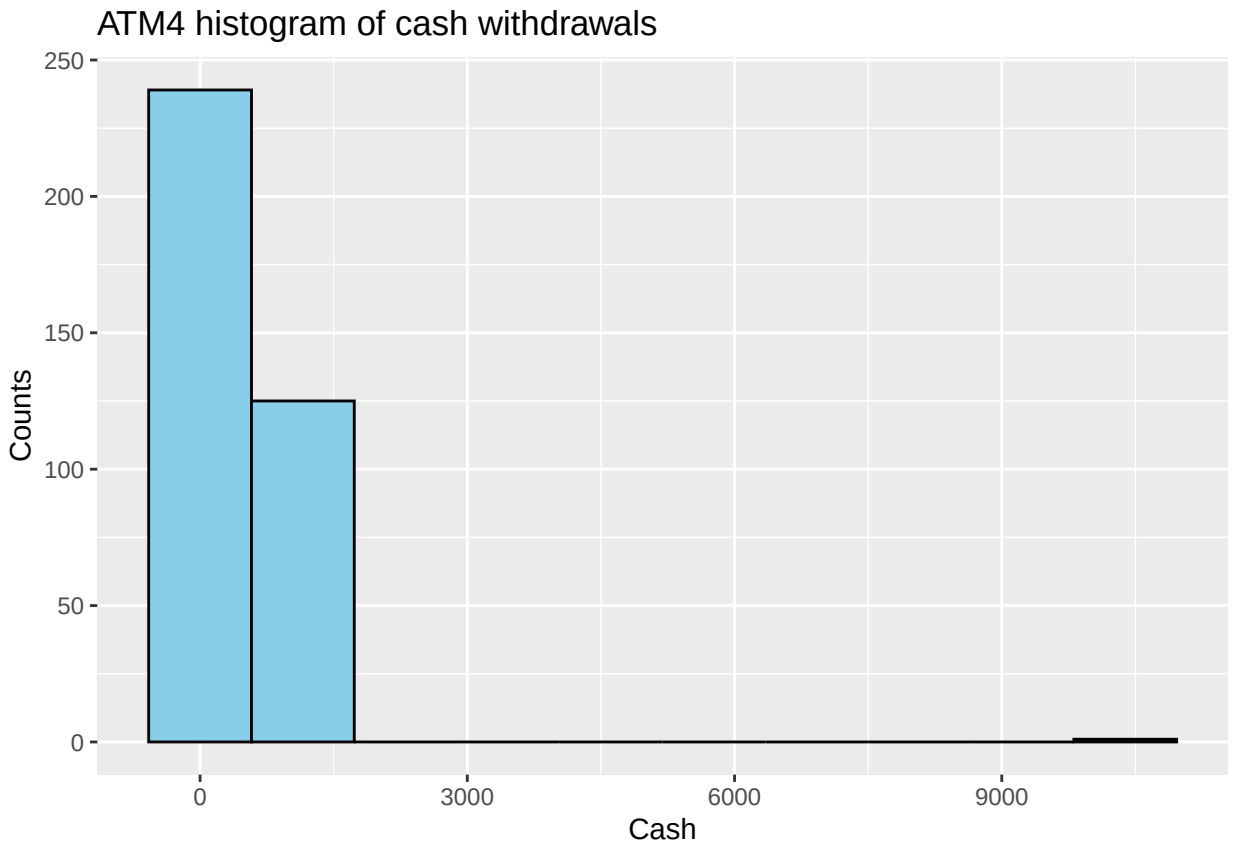

```
sturge_histo(atm_624_data, "ATM2", 15, "ATM2 histogram of cash withdrawals")
```



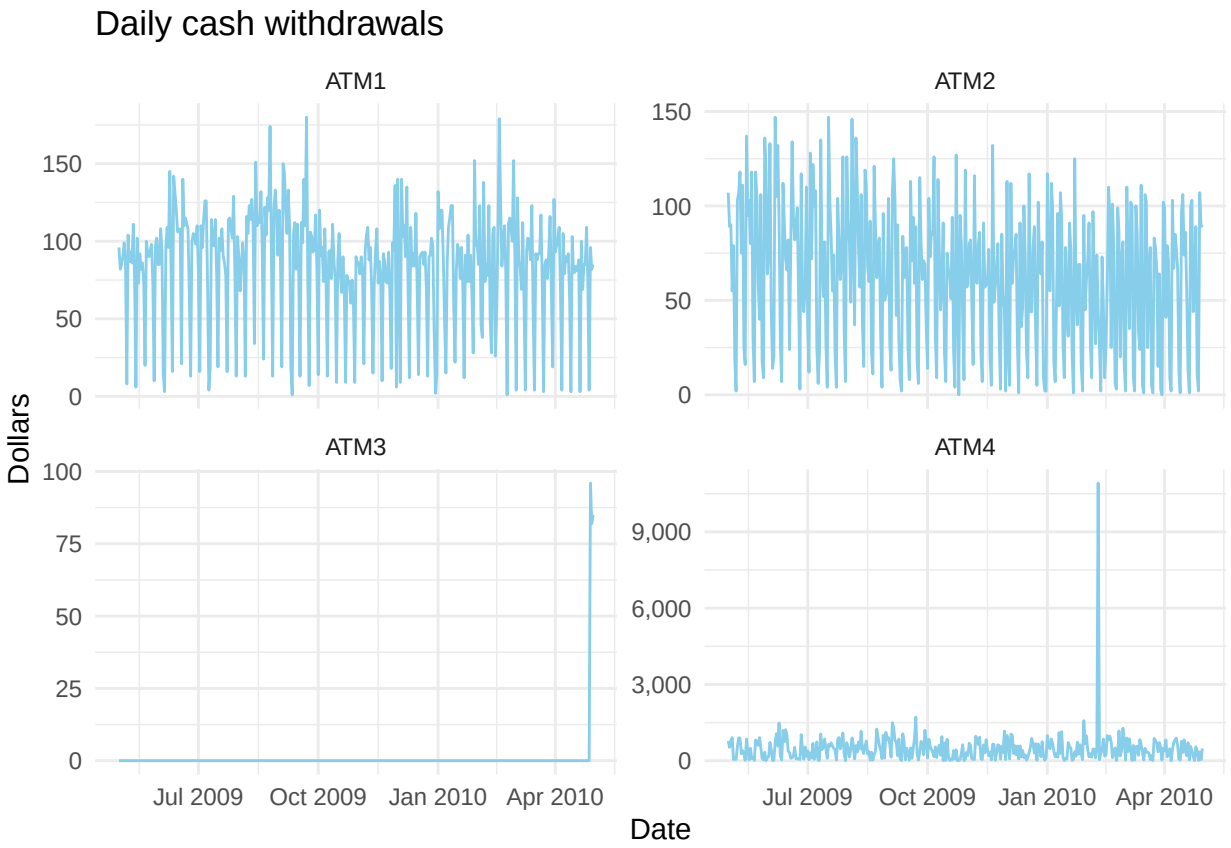

```
sturge_histo(atm_624_data, "ATM3", 10, "ATM3 histogram of cash withdrawals")
```




```
sturge_histo(atm_624_data, "ATM4", 1154, "ATM4 histogram of cash withdrawals")
```




```
atm_624_data %>%
  filter(!is.na(cash)) %>%
  ggplot(aes(x = date, y = cash)) +
    geom_line(color = "skyblue") +
    facet_wrap(~atm, ncol = 2, scales = "free_y") +
    labs(title = "Daily cash withdrawals",
         x = "Date",
         y = "Dollars") +
    scale_y_continuous(labels = scales::comma_format()) +
    theme_minimal()
```



There are a whole ton of zero values for ATM3, as you can see from the charts below. Looks like there are only actual valyues for the last three days and then zeroes going backwards for the rest. I can leave that data as is and do a forecast that takes that into account. I do want to handle that outlier in the ATM4 data because it's really just a crazy one off that doesn't represent anything. I'm going to drop it and otherwise leave this data as is until I handle NAs.

```
atm_624_data <- atm_624_data[!(atm_624_data$atm == "ATM4" &
                                atm_624_data$cash > 4 * sort(atm_624_data$cash[atm_624_data$atm == "ATM4

atm_624_outlier <- atm_624_data %>%
  group_by(atm) %>%
  summarise(
    total_values = n(),
    min_value = min(cash, na.rm = TRUE),
    mean_value = mean(cash, na.rm = TRUE),
    median_value = median(cash, na.rm = TRUE),
    max_value = max(cash, na.rm = TRUE),
  )

cat("Summary data after outlier removal: \n")
```

```
## Summary data after outlier removal:
```

```
atm_624_outlier
```

```
## # A tibble: 4 x 6
##   atm   total_values min_value mean_value median_value max_value
##   <chr>        <int>     <dbl>      <dbl>        <dbl>     <dbl>
## 1 ATM1           365         1       83.9           91       180
```

```
## 2 ATM2           365      0        62.6            67       147
## 3 ATM3           365      0         0.721           0        96
## 4 ATM4           364      1.56    445.            403.     1712.
```

```
cat("I'm now going to return to the NAs from above and implement a strategy. Here they are: \n")
```

```
## I'm now going to return to the NAs from above and implement a strategy. Here they are:
```

```
cash_na_subset2 <- atm_624_data[rowSums(is.na(atm_624_data)) > 0, ]
cat("The remaining missing values by ATM are: \n")
```

```
## The remaining missing values by ATM are:
```

```
cash_na_subset2
```

```
## # A tibble: 5 x 4
##   date       atm    cash sturges_bw
##   <date>     <chr> <dbl>      <dbl>
## 1 2009-06-13 ATM1     NA       19.0
## 2 2009-06-16 ATM1     NA       19.0
## 3 2009-06-18 ATM2     NA       15.6
## 4 2009-06-22 ATM1     NA       19.0
## 5 2009-06-24 ATM2     NA       15.6
```

Using the median value as the imputation for the missing values is the best approach here. The number of NAs are small and the rest of the data for each of the ATMs with missing values is robust enough. I'm also now going to add a column that takes the cash amount, which is in hundreds of dollars, and convert to full dollars by multiplying by 100.

```
atm_624_data <- atm_624_data %>%
  group_by(atm) %>%
  mutate(cash = ifelse(is.na(cash), median(cash, na.rm = TRUE), cash)) %>%
  ungroup()

atm_624_data <- atm_624_data %>%
  mutate(full_usd_amt = cash * 100)

cat("Here's a sample of the data after dollars represented as hundreds is coverted to full dollar value
```

```
## Here's a sample of the data after dollars represented as hundreds is coverted to full dollar value:
```

```
atm_624_data
```

```
## # A tibble: 1,459 x 5
##    date       atm    cash sturges_bw full_usd_amt
##    <date>     <chr> <dbl>      <dbl>        <dbl>
##  1 2009-05-01 ATM1     96       19.0         9600
##  2 2009-05-01 ATM2    107       15.6        10700
##  3 2009-05-02 ATM1     82       19.0         8200
##  4 2009-05-02 ATM2     89       15.6         8900
##  5 2009-05-03 ATM1     85       19.0         8500
##  6 2009-05-03 ATM2     90       15.6         9000
##  7 2009-05-04 ATM1     90       19.0         9000
##  8 2009-05-04 ATM2     55       15.6         5500
##  9 2009-05-05 ATM1     99       19.0         9900
## 10 2009-05-05 ATM2     79       15.6         7900
## # i 1,449 more rows
```

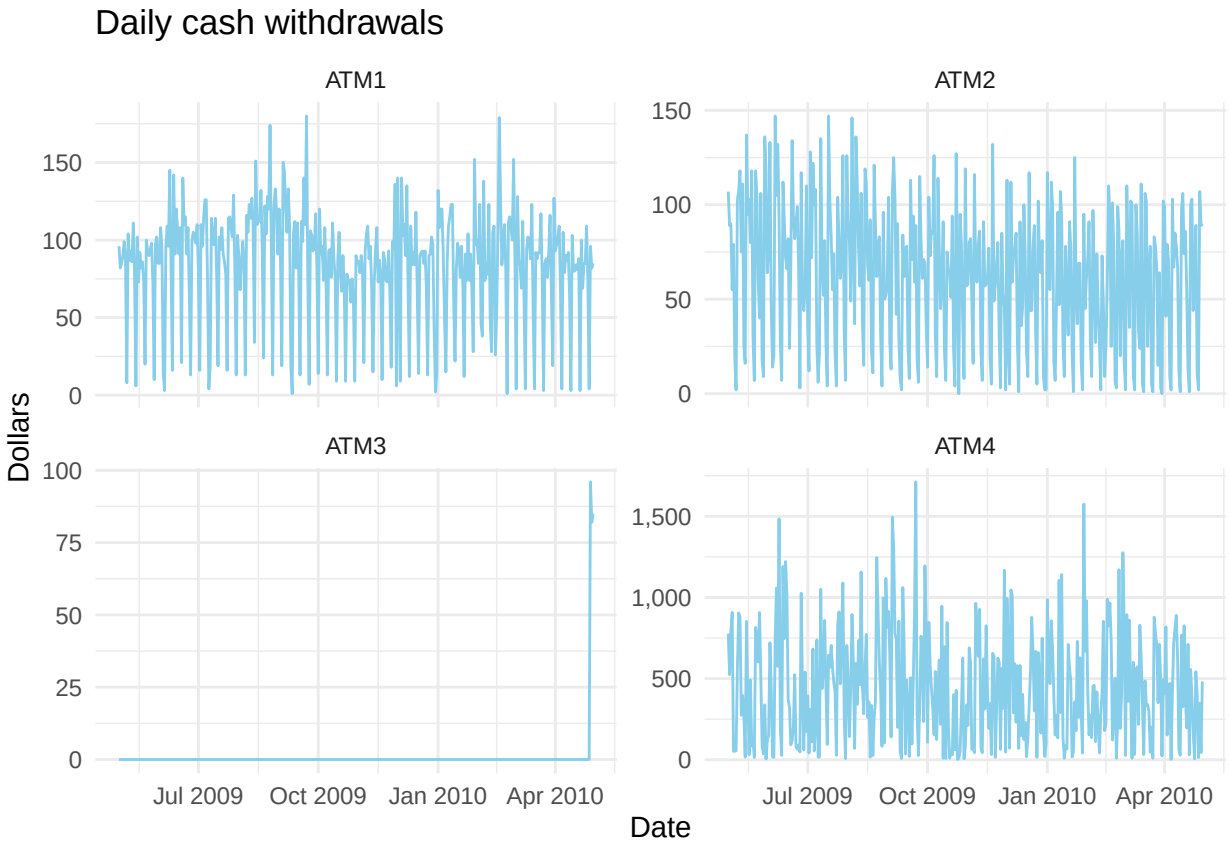
Forecasting May 2010 cash withdrawals

The dataset provided is daily level cash withdrawals from the four ATMs for 5/1/2009 to 4/30/2009. The task now is to produce a May 2010 forecast for each ATM.

From the assignment: "I am being somewhat ambiguous on purpose to make this have a little more business feeling. Explain and demonstrate your process, techniques used and not used, and your actual forecast."

Here are the charts by ATM, with ATM looking much better without the outlier.

```
atm_624_data %>%
  filter(!is.na(cash)) %>%
  ggplot(aes(x = date, y = cash)) +
    geom_line(color = "skyblue") +
    facet_wrap(~atm, ncol = 2, scales = "free_y") +
    labs(title = "Daily cash withdrawals",
         x = "Date",
         y = "Dollars") +
    scale_y_continuous(labels = scales::comma_format()) +
    theme_minimal()
```



I'm going to use the following forecasting techniques for the noted ATMs, with the characteristics of the data dictating the approach:

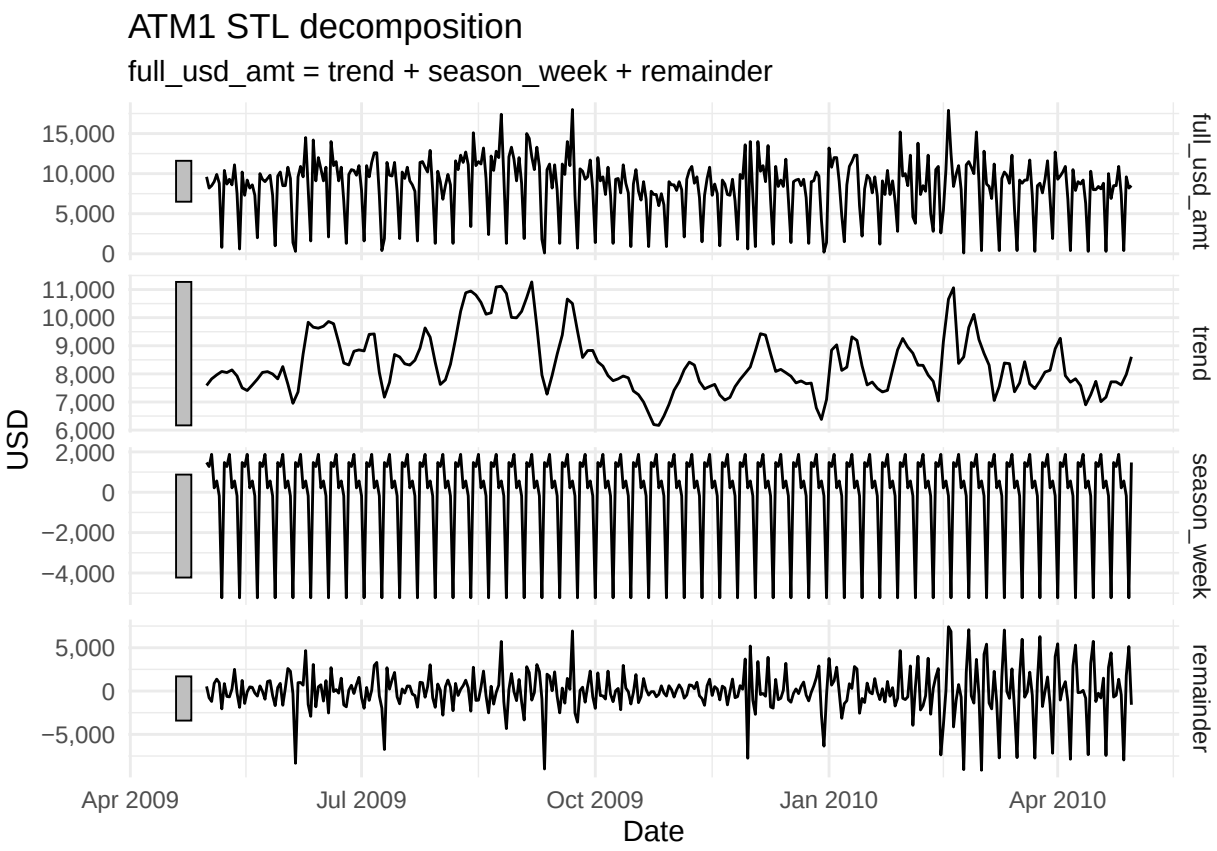
ATM1 and ATM2: Both have stable patterns with some seasonality. I'll use STL decomposition and ARIMA to capture both trend and seasonality effectively.

ATM3: Given that there are only three data points, I'll use a straight forward naive with drift forecast.

ATM4: Given the chaotic nature of the data, I'll use SARIMA after decomposing the trend and seasonality.

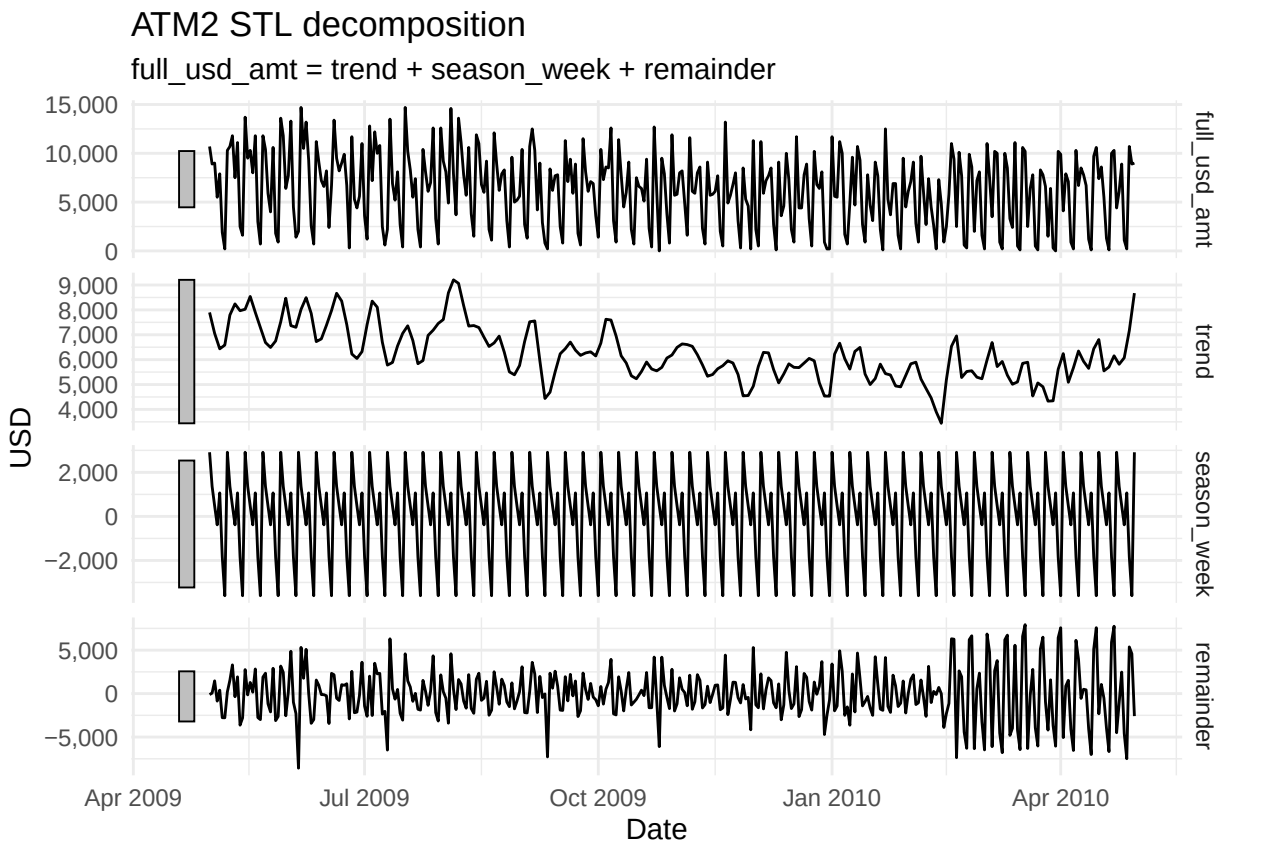
ATM 1 and 2 - decomposition and forecast

```
atm1_data <- atm_624_data %>%
  filter(atm == "ATM1") %>%
  select(date, full_usd_amt) %>%
  as_tsibble(index = date)

atm1_stl <- atm1_data %>%
  model(stl = STL(full_usd_amt ~ season(window = "periodic")))

atm1_stl %>%
  components() %>%
  autoplot() +
    labs(title = "ATM1 STL decomposition",
         x = "Date",
         y = "USD") +
    scale_y_continuous(labels = scales::comma_format()) +
    theme_minimal()
```




```
atm2_data <- atm_624_data %>%
  filter(atm == "ATM2") %>%
  select(date, full_usd_amt) %>%
  as_tsibble(index = date)

atm2_stl <- atm2_data %>%
  model(stl = STL(full_usd_amt ~ season(window = "periodic")))
```

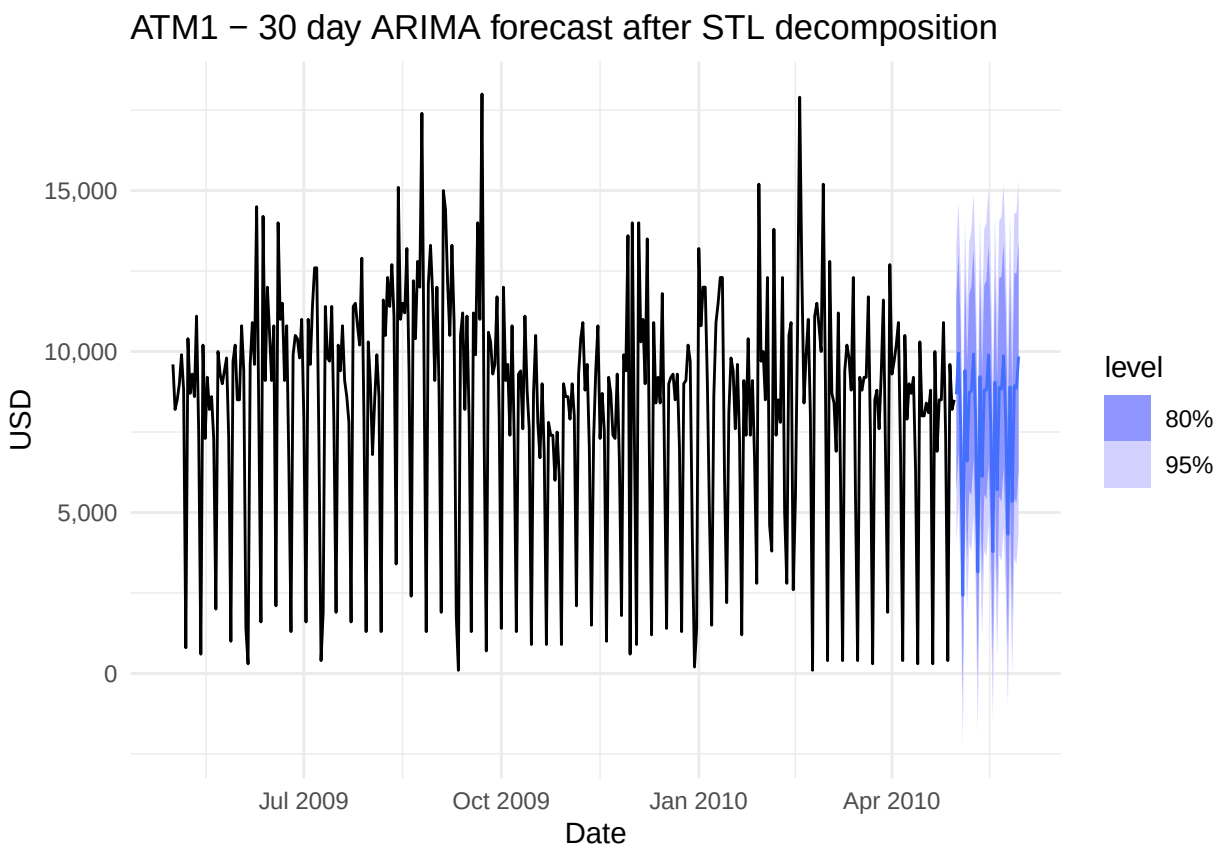
```
atm2_stl %>%
  components() %>%
  autoplot() +
    labs(title = "ATM2 STL decomposition",
         x = "Date",
         y = "USD") +
    scale_y_continuous(labels = scales::comma_format()) +
    theme_minimal()
```



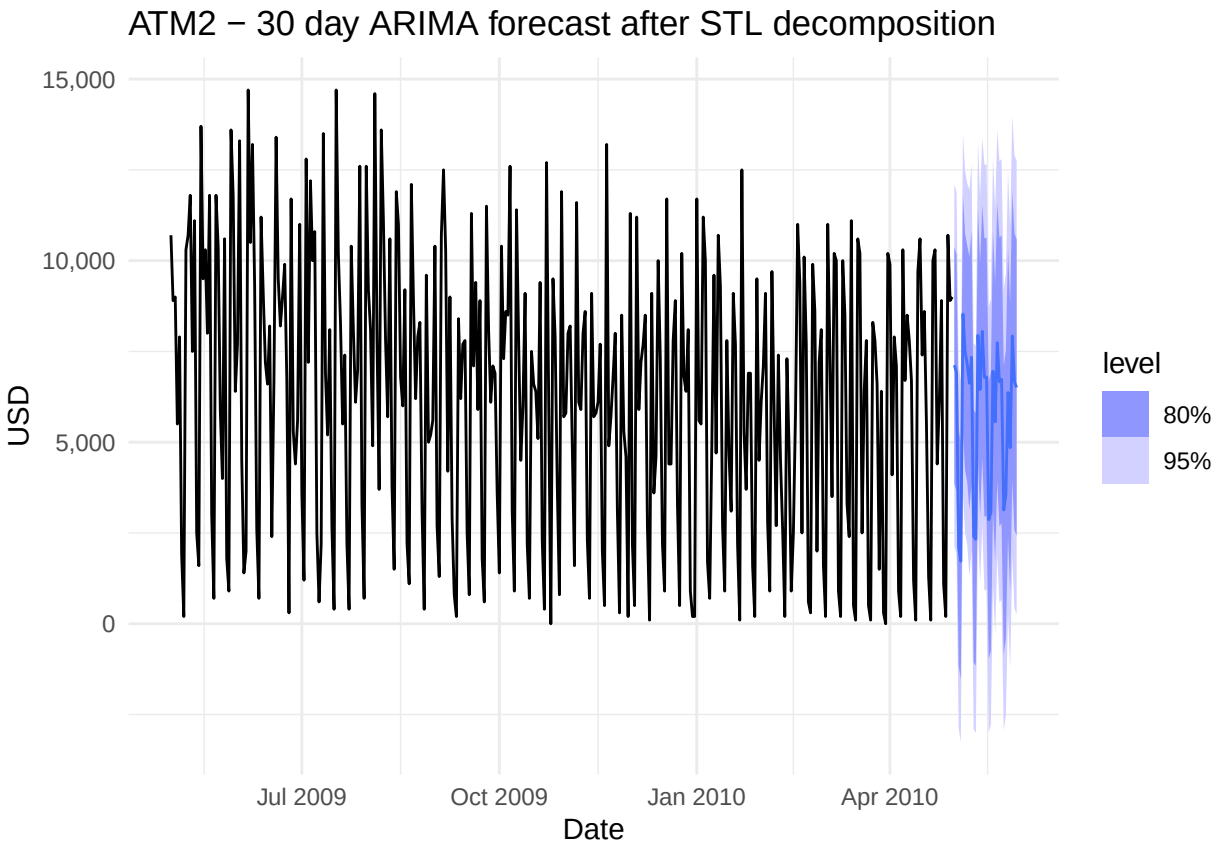
ATM1: the trend shows a slight upward movement overall, with consistent weekly seasonality visible in the decomposition. The remainder indicates some noise and spikes, suggesting occasional fluctuations outside the normal pattern.

ATM2: the trend appears more volatile, but there's still a clear seasonal weekly pattern. The remainder component shows increased variability, particularly toward the end, pointing to some unpredictability in the withdrawals.

```
atm1_fc <- atm1_data %>%
  model(ARIMA(full_usd_amt ~ trend() + season())) %>%
  forecast(h = "30 days")

atm1_fc %>%
  autoplot(atm1_data) +
    labs(title = "ATM1 - 30 day ARIMA forecast after STL decomposition",
       x = "Date",
       y = "USD") +
    scale_y_continuous(labels = scales::comma_format()) +
    theme_minimal()
```




```
atm2_fc <- atm2_data %>%
  model(ARIMA(full_usd_amt ~ trend() + season())) %>%
  forecast(h = "30 days")

atm2_fc %>%
  autoplot(atm2_data) +
    labs(title = "ATM2 - 30 day ARIMA forecast after STL decomposition",
       x = "Date",
       y = "USD") +
    scale_y_continuous(labels = scales::comma_format()) +
    theme_minimal()
```

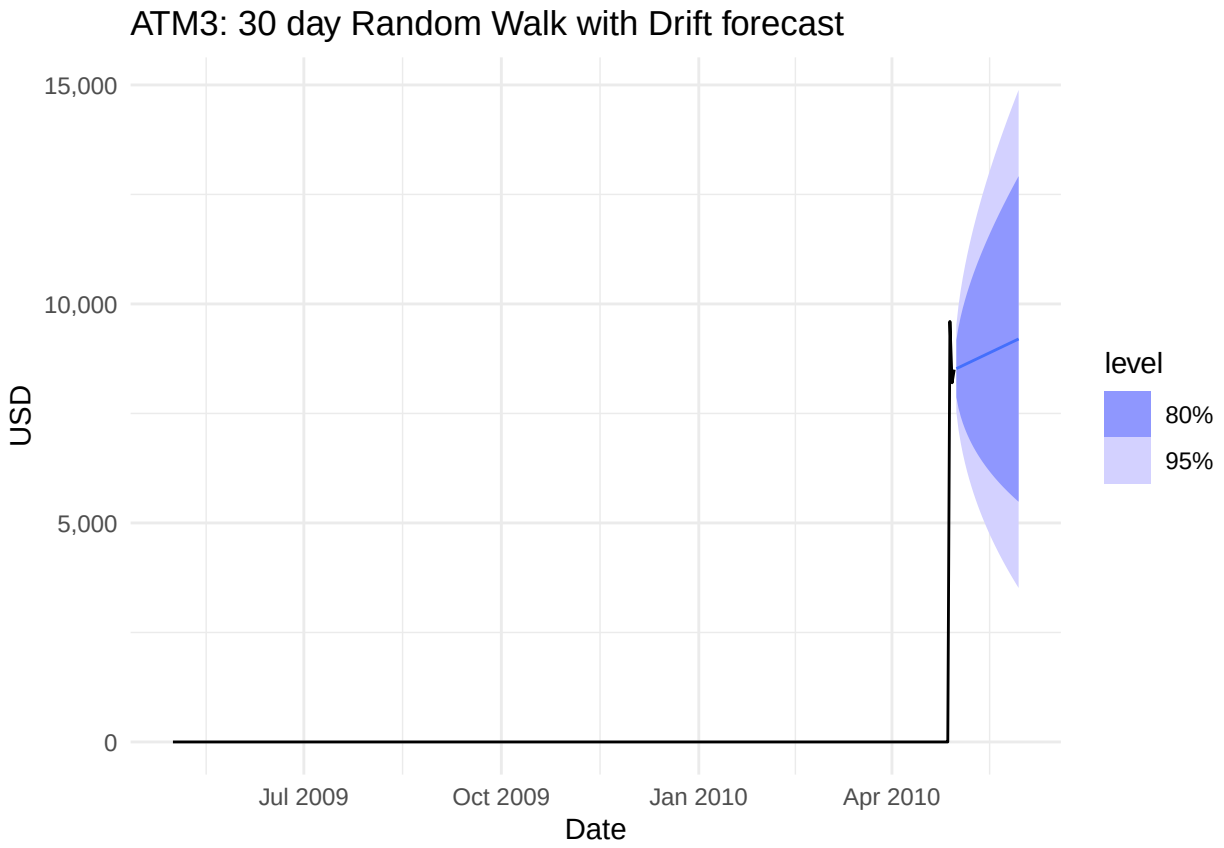


ATM1: the ARIMA forecast shows a continuation of the noisy withdrawal patterns, with a relatively stable level around $10,000 USD and tight confidence intervals. However, there are occasional spikes in the forecast, suggesting the possibility of short-term variability in withdrawals.

ATM2: the forecast looks similar, with consistent levels and fluctuations around $10,000 USD. The forecast intervals remain tight, but the volatility of the data means we could still see some significant spikes in daily activity.

ATM3 forecast with naive with drift

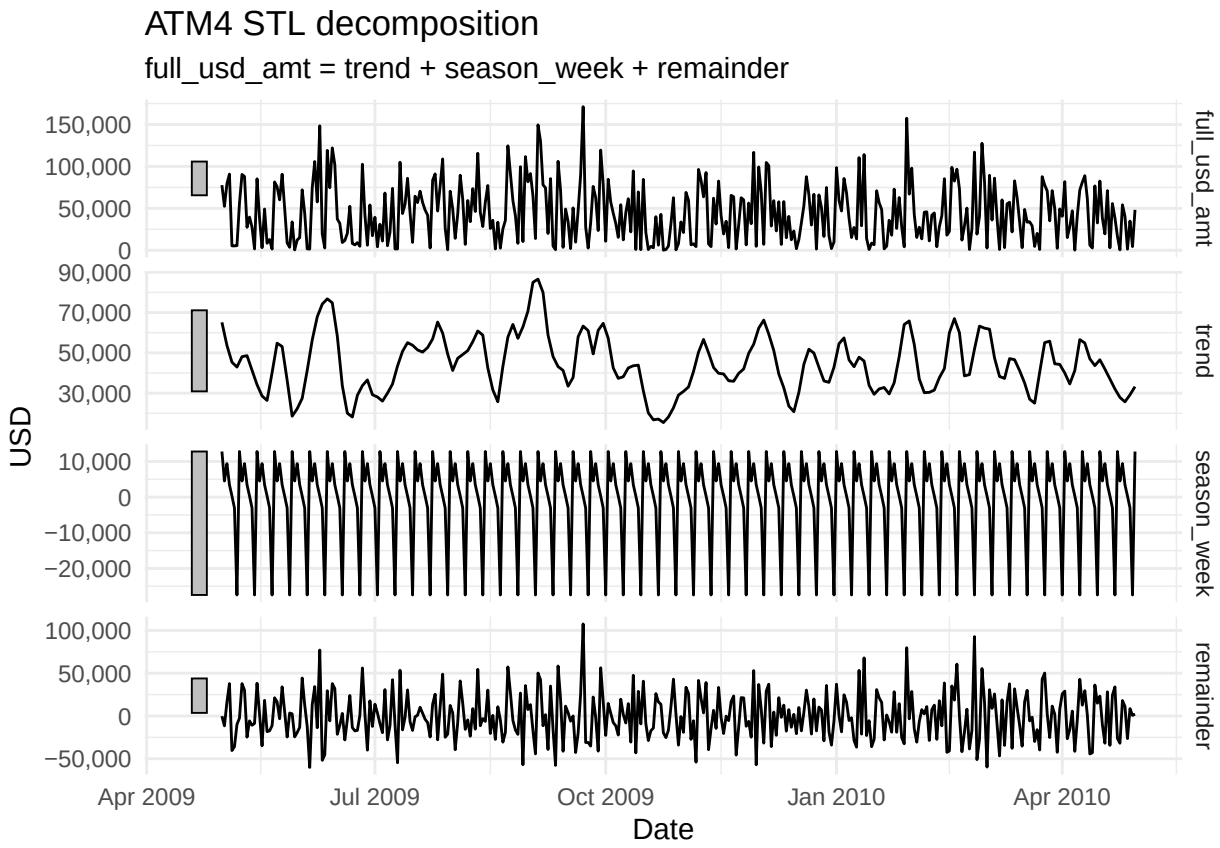
```
atm3_data <- atm_624_data %>%
  filter(atm == "ATM3") %>%
  select(date, full_usd_amt) %>%
  as_tsibble(index = date)

atm3_fc <- atm3_data %>%
  model(RW(full_usd_amt ~ drift())) %>%
  forecast(h = "30 days")

atm3_fc %>%
  autoplot(atm3_data) +
    labs(title = "ATM3: 30 day Random Walk with Drift forecast",
       x = "Date",
       y = "USD") +
    scale_y_continuous(labels = scales::comma_format()) +
    theme_minimal()
```



This forecast looks accurate and is what I would use in the real world, with a caveat to return to it every few weeks to see if the model can change based on new data.

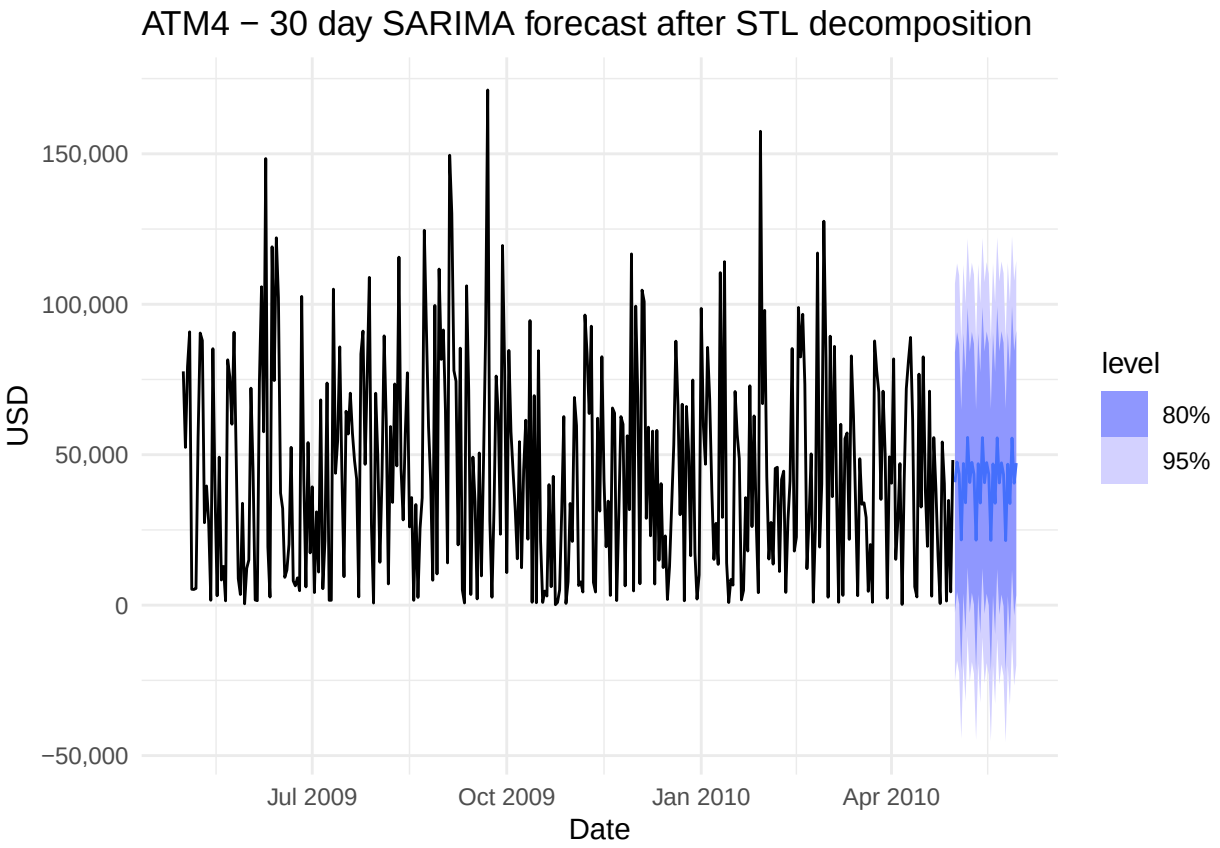
ATM4 SARIMA forecast with STL decomposition

```
atm4_data <- atm_624_data %>%
  filter(atm == "ATM4") %>%
  as_tsibble(index = date) %>%
  select(date, full_usd_amt) %>%
  fill_gaps() %>%
  mutate(full_usd_amt = ifelse(is.na(full_usd_amt), mean(full_usd_amt, na.rm = TRUE), full_usd_amt))

atm4_stl <- atm4_data %>%
  model(stl = STL(full_usd_amt ~ season(window = "periodic")))

atm4_stl %>%
  components() %>%
  autoplot() +
    labs(title = "ATM4 STL decomposition",
         x = "Date",
         y = "USD") +
    scale_y_continuous(labels = scales::comma_format()) +
    theme_minimal()
```



This shows a clear weekly seasonality pattern, with significant fluctuations in the remainder, suggesting irregular spikes in withdrawals. The trend component is more volatile than the other ATMs, indicating instability in the overall withdrawal amounts throughout the year.

```
atm4_fc <- atm4_data %>%
  model(ARIMA(full_usd_amt ~ pdq(0,1,1) + PDQ(0,1,1))) %>%
  forecast(h = "30 days")

atm4_fc %>%
  autoplot(atm4_data) +
    labs(title = "ATM4 - 30 day SARIMA forecast after STL decomposition",
       x = "Date",
       y = "USD") +
    scale_y_continuous(labels = scales::comma_format()) +
    theme_minimal()
```



This SARIMA forecast shows a continuation of the volatility seen in the historical data, with wide fluctuations in withdrawal amounts. The confidence intervals are relatively tight, but the forecast still suggests significant variability in the near future, mirroring the unpredictable nature of past withdrawals.

B. Residential power forecast

The assignment states: "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. Add this to your existing files above."

Overview:

There are three fields in this dataset: * CaseSequence: a double precision floating point number. It looks to be a column added by the dataset creator to help put the `YYYY-MMM` column in order. If you sort by CaseSequence, you get the date column in order. Sort by the date column, you just get an alphabetical sort

- YYYY-MMM: string character representation of a a date in YYYY-MMM format. IE 1998-Apr. Curious choice for a date format that clearly was causing issues of some sort since the CaseSequence column only exists to keep dates in order.
- KWH: a double precision floating point representation of power consumption in kilowatt hours

```
cat("Glimise of date:\n")
```

```
## Glimise of date:
```

```
glimpse(res_cust_forecast_load)
```

```
## Rows: 192
```

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

```
res_intials <- res_cust_forecast_load %>%
  summarise(
    total_values = n(),
    missing_values = sum(is.na(KWH)),
    min_value = min(KWH, na.rm = TRUE),
    mean_value = mean(KWH, na.rm = TRUE),
    median_value = median(KWH, na.rm = TRUE),
    max_value = max(KWH, na.rm = TRUE)
  )

cat("Initial summary stats before change: \n")
```

```
## Initial summary stats before change:
```

```
res_intials
```

```
## # A tibble: 1 x 6
##   total_values missing_values min_value mean_value median_value max_value
##          <int>          <int>     <dbl>      <dbl>        <dbl>     <dbl>
## 1          192              1    770523   6502475.      6283324  10655730
```

Tidying the data

I'm going to convert this `YYYY-MMM` column to a date value column and then drop `CaseSequence`. The task is a monthly forecast of KWH for 2014 and that doesn't require a link back to this `CaseSequence` column.

```
res_cust_forecast_load <- res_cust_forecast_load %>%
  select(-CaseSequence) %>%
  rename(
    year_month = `YYYY-MMM`,
    kwh = KWH
  )

res_cust_forecast_load$year_month <- as.Date(paste0(res_cust_forecast_load$year_month, "-01"), format =

cat("Here's a glimpse of the data after the above. Much better :) \n")
```

```
## Here's a glimpse of the data after the above. Much better :)
```

```
glimpse(res_cust_forecast_load)
```

```
## Rows: 192
## Columns: 2
## $ year_month <date> 1998-01-01, 1998-02-01, 1998-03-01, 1998-04-01, 1998-05-01~
## $ kwh        <dbl> 6862583, 5838198, 5420658, 5010364, 4665377, 6467147, 89147~
```

Let's see what the missing value is to determine what to do about it:

```
res_na_init <- res_cust_forecast_load[rowSums(is.na(res_cust_forecast_load)) > 0, ]

cat("The missing value is: /n")
```

```
## The missing value is: /n
```

```
res_na_init
```

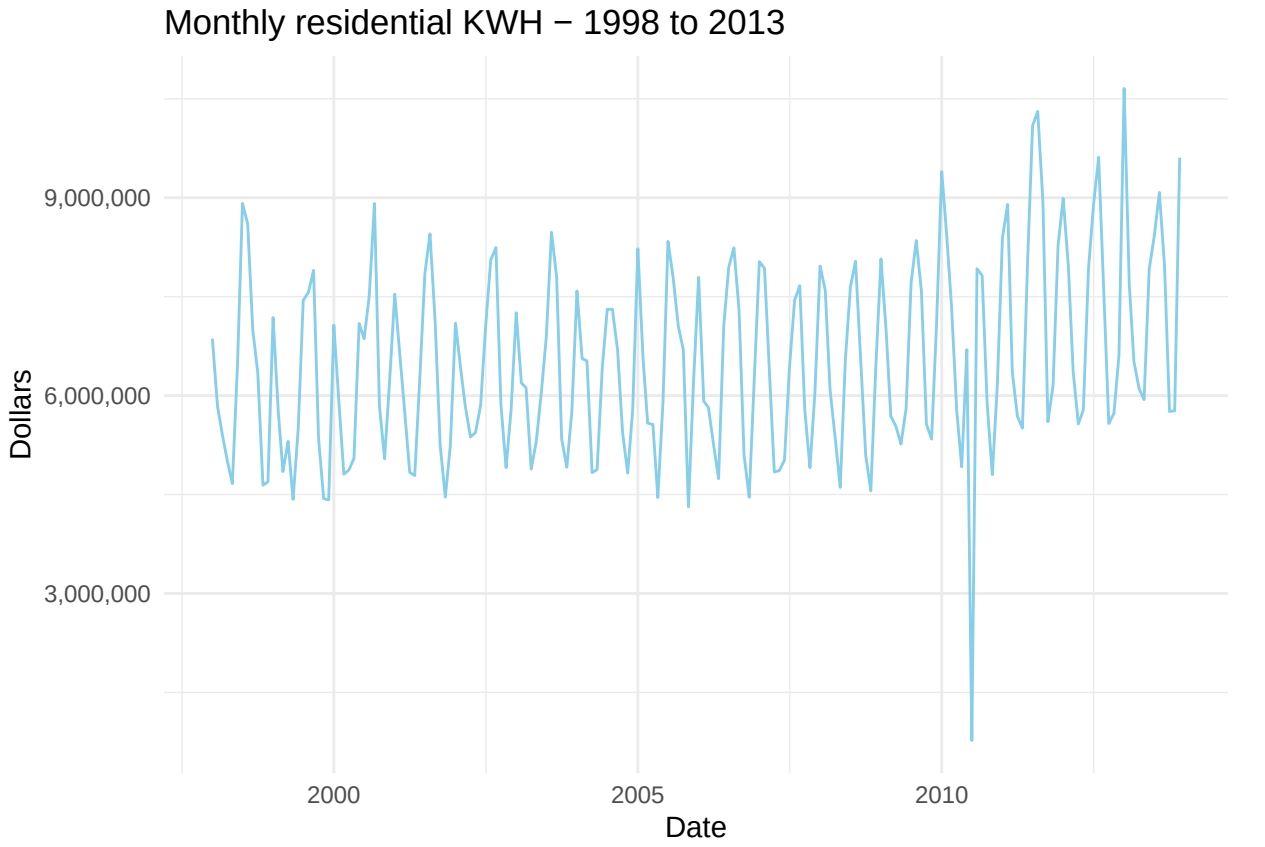
```
## # A tibble: 1 x 2
##   year_month   kwh
##   <date>     <dbl>
## 1 2008-09-01    NA
```

Looks to be KWH for September 2008.There are ten years of date, which means I have fourteen other values for September. I'm going to plot all the data to see what it looks like because I'm included to use a value that's grounded in just September monthly data.

First, I'll plot the data overall and then broken down by month.

Overall:

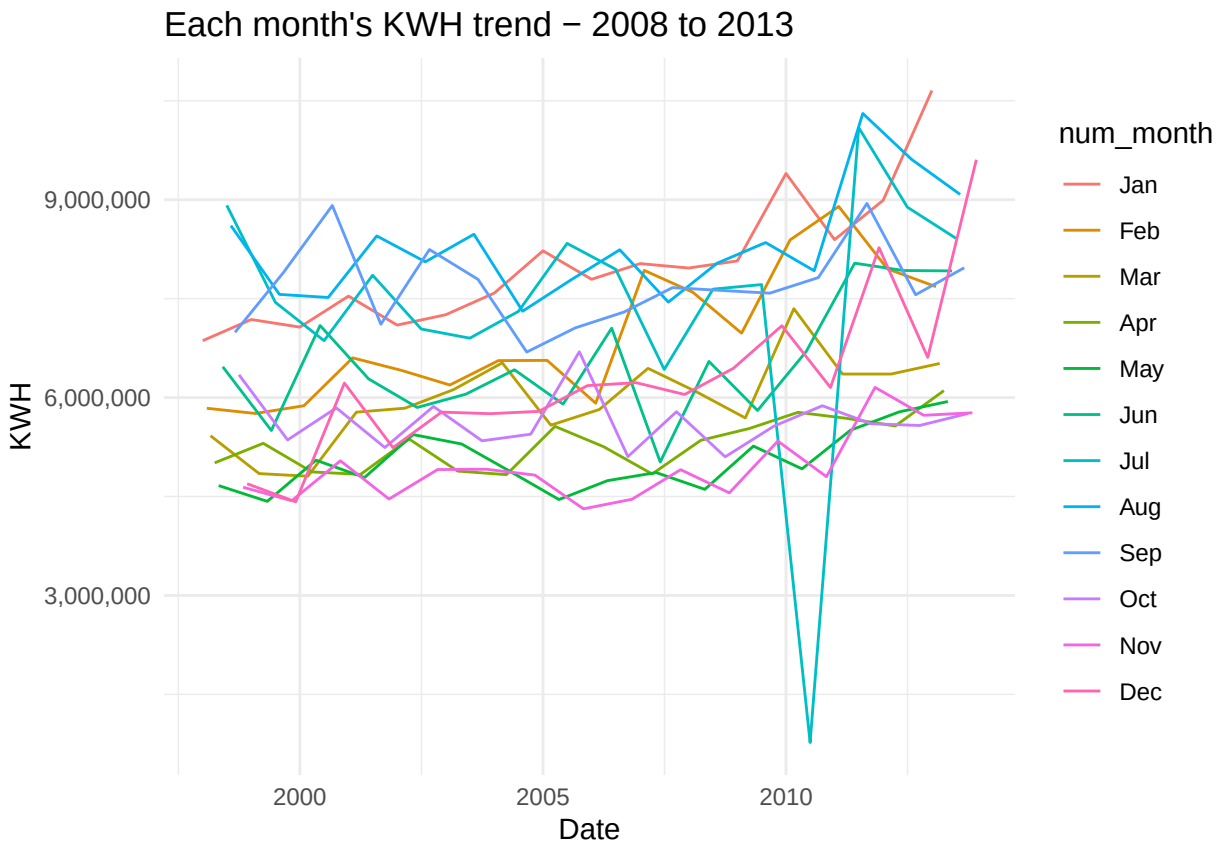
```
res_cust_forecast_load %>%
  filter(!is.na(kwh)) %>%
  ggplot(aes(x = year_month, y = kwh)) +
    geom_line(color = "skyblue") +
    labs(title = "Monthly residential KWH - 1998 to 2013",
         x = "Date",
         y = "Dollars") +
    scale_y_continuous(labels = scales::comma_format()) +
    theme_minimal()
```



I see the outlier there but I

By month:

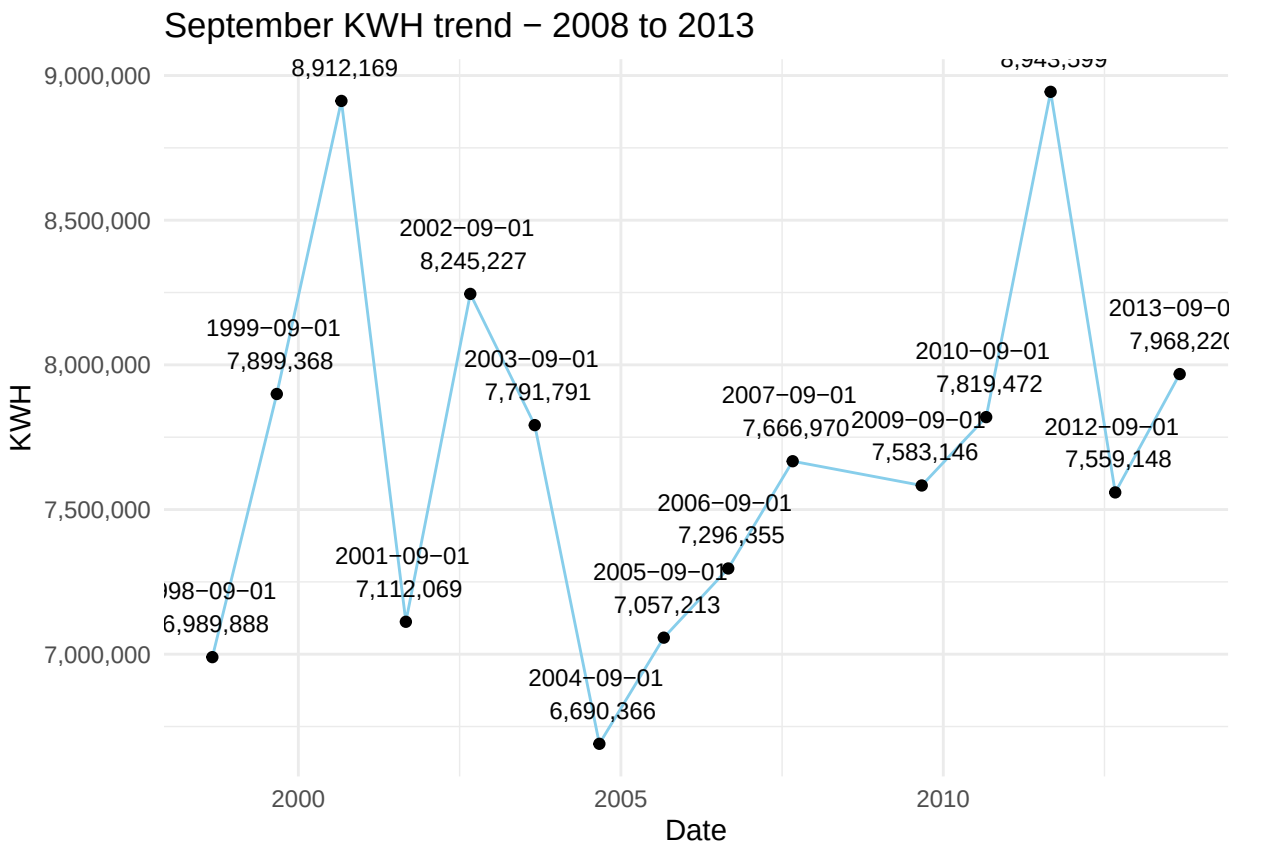
```
res_cust_forecast_load$num_month <- factor(month(res_cust_forecast_load$year_month, label = TRUE))

res_cust_forecast_load %>%
  filter(!is.na(kwh)) %>%
  ggplot(aes(x = year_month, y = kwh, color = num_month)) +
    geom_line() +
    labs(title = "Each month's KWH trend - 2008 to 2013",
         x = "Date",
         y = "KWH") +
    scale_y_continuous(labels = scales::comma_format()) +
    scale_color_manual(values = scales::hue_pal()(12)) +
    theme_minimal()
```



OK. While that chart is interesting, it doesn't really make September stand out so I'm going to make one just for the month where I need to figure out a value:

```
res_cust_forecast_load %>%
  filter(!is.na(kwh) & num_month == 'Sep') %>%
  ggplot(aes(x = year_month, y = kwh)) +
    geom_line(color = "skyblue") +
    geom_point() +
    geom_text(aes(label = paste(year_month, "\n", scales::comma(kwh))),
              vjust = -0.5, hjust = 0.5, size = 3, angle = 0) +
    labs(title = "September KWH trend - 2008 to 2013",
         x = "Date",
         y = "KWH") +
    scale_y_continuous(labels = scales::comma_format()) +
```

```
  theme_minimal()
```



With the missing month as September 2008, I can safely impose a value that's halfway between the 2007-09-01 and 2009-09-01 value. Since it's printed right there, I can just grab it, do some math, and then impute it.

```
sept_2008 <- 7666970 - ((7666970 - 7583146) / 2)

res_cust_forecast_load$kwh[res_cust_forecast_load$year_month == "2008-09-01"] <- sept_2008

res_na_removed <- res_cust_forecast_load[rowSums(is.na(res_cust_forecast_load)) > 0, ]

cat("Here's where a missing value would be if it existed: /n")
```

```
## Here's where a missing value would be if it existed: /n
```

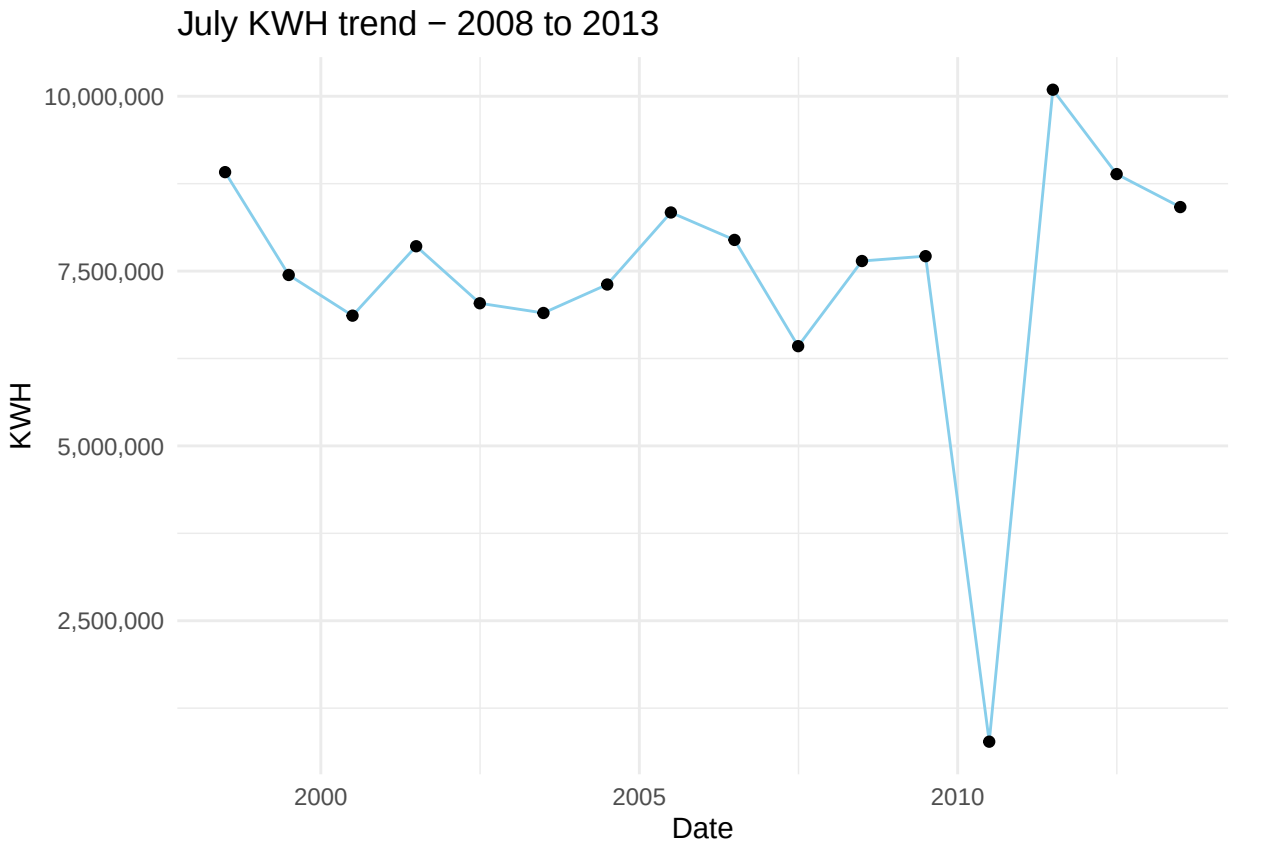
```
res_na_removed
```

```
## # A tibble: 0 x 3
## # i 3 variables: year_month <date>, kwh <dbl>, num_month <ord>
```

That outlier dip to the bottom is from July so let's take look at that more:

```
res_cust_forecast_load %>%
  filter(!is.na(kwh) & num_month == 'Jul') %>%
  ggplot(aes(x = year_month, y = kwh)) +
    geom_line(color = "skyblue") +
    geom_point() +
    labs(title = "July KWH trend - 2008 to 2013",
         x = "Date",
         y = "KWH") +
```

```
    scale_y_continuous(labels = scales::comma_format()) +
    theme_minimal()
```




```
july_min <- res_cust_forecast_load %>%
  filter(num_month == 'Jul') %>%
  summarise(min_kwh = min(kwh, na.rm = TRUE)) %>%
  pull(min_kwh)

cat("The minimum value across all of July values is: \n")
```

```
## The minimum value across all of July values is:
```

```
july_min
```

```
## [1] 770523
```

I'm going to handle that value during forecasting rather than in advance.

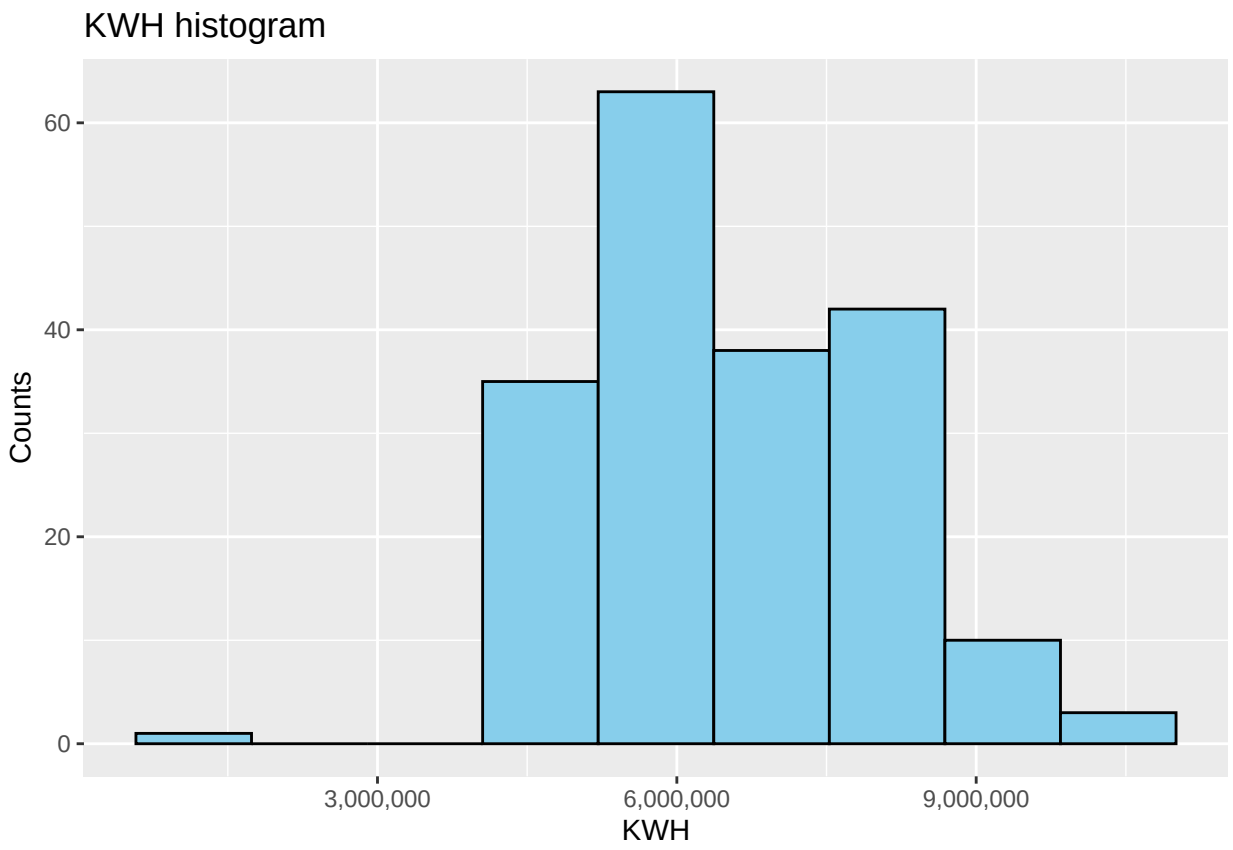
```
res_redux <- res_cust_forecast_load %>%
  summarise(
    total_values = n(),
    missing_values = sum(is.na(kwh)),
    min_value = min(kwh, na.rm = TRUE),
    mean_value = mean(kwh, na.rm = TRUE),
    median_value = median(kwh, na.rm = TRUE),
    max_value = max(kwh, na.rm = TRUE)
  )

cat("Summary stats after all of the above: \n")
```

```
## Summary stats after all of the above:
res_redux
```

```
## # A tibble: 1 x 6
##   total_values missing_values min_value mean_value median_value max_value
##          <int>          <int>     <dbl>      <dbl>        <dbl>     <dbl>
## 1          192              0    770523   6508321.      6314472  10655730
```

Forecasting using STL decomposition and Box Cox transformation

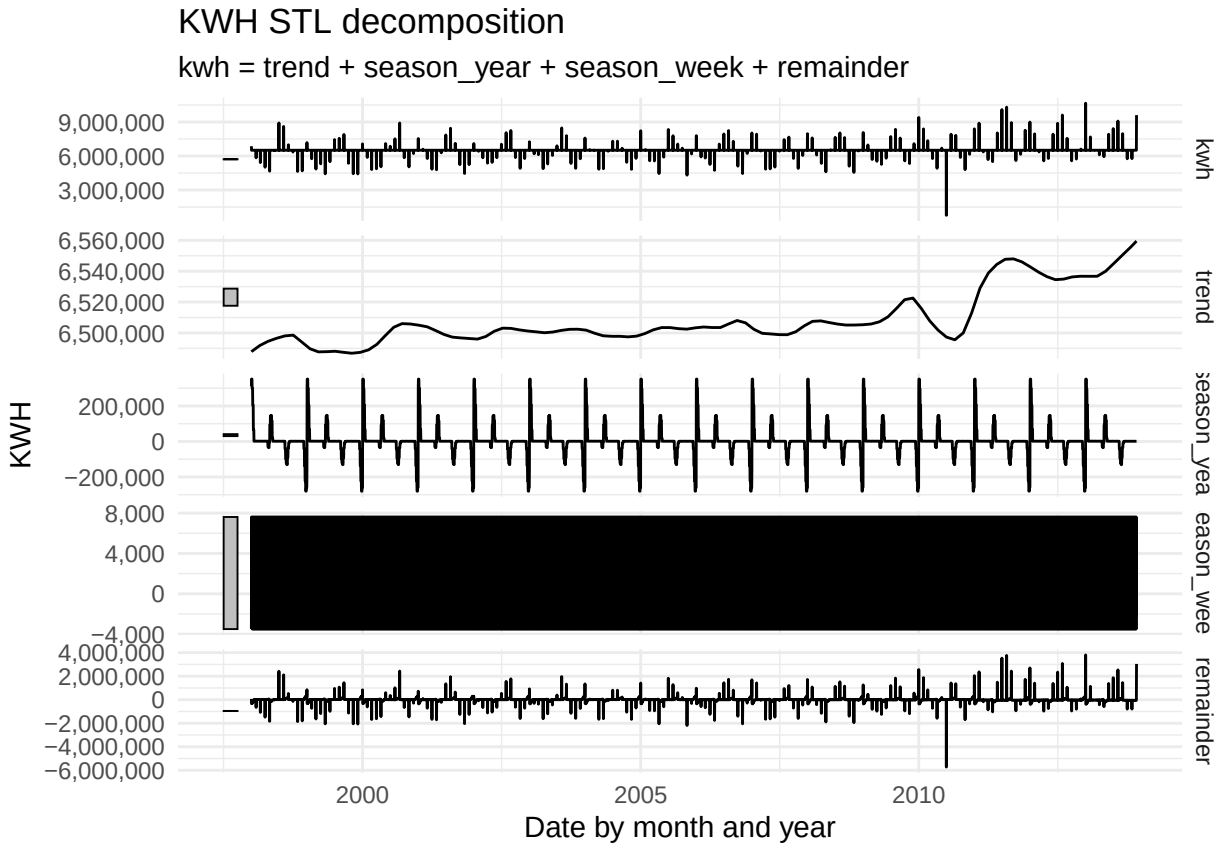
```
sturges_kwh <- round(sturges_bin_width(res_cust_forecast_load$kwh),0)

  res_cust_forecast_load %>%
    ggplot(aes(x = kwh)) +
    geom_histogram(binwidth = sturges_kwh,
                   fill = "skyblue",
                   color = "black") +
    labs(title = "KWH histogram",
         x = "KWH",
         y = "Counts") +
    scale_x_continuous(labels = scales::comma_format())
```



The histogram isn't

```
rest_cust_fd <- res_cust_forecast_load %>%
  as_tsibble(index = year_month) %>%
  fill_gaps() %>%
  mutate(kwh = ifelse(is.na(kwh), mean(kwh, na.rm = TRUE), kwh))
```

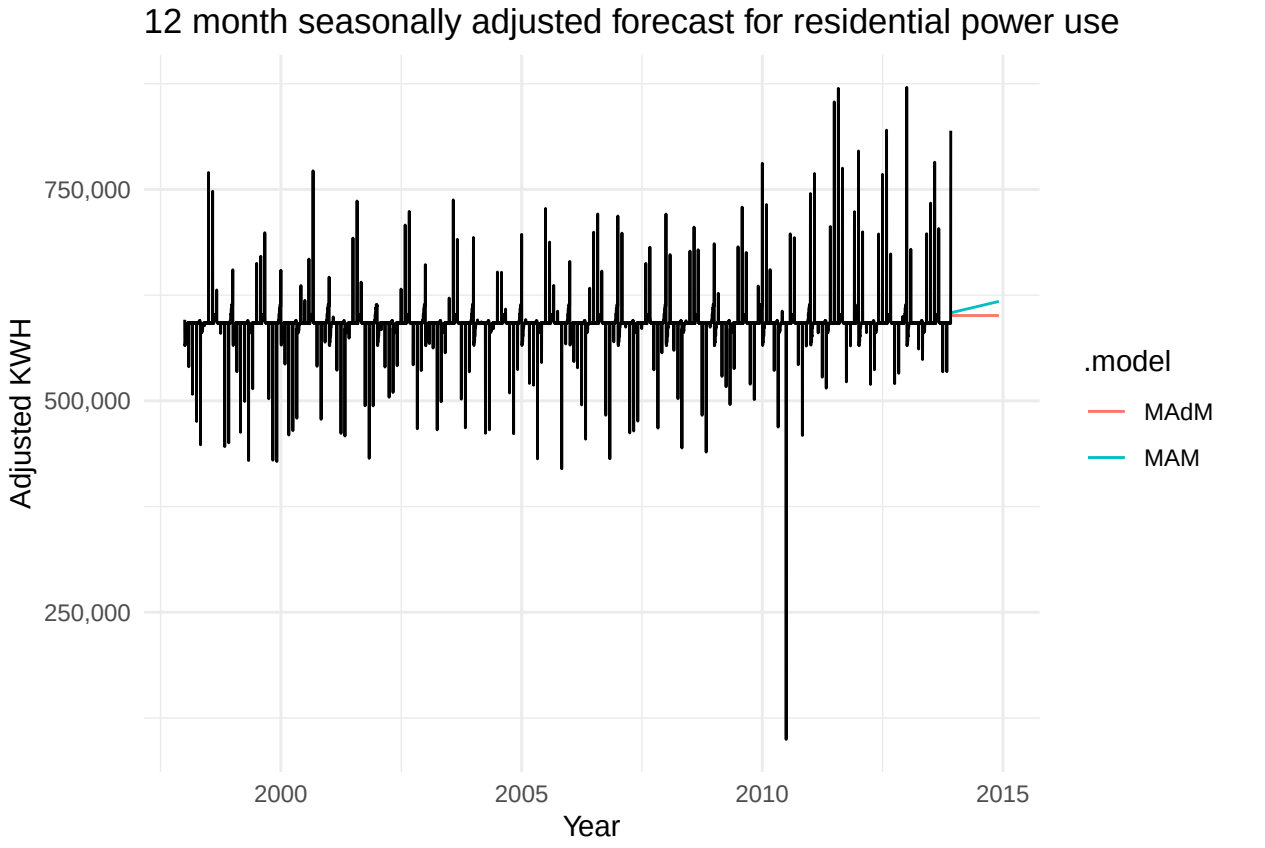
```
rest_cust_stl <- rest_cust_fd %>%
  model(stl = STL(kwh ~ season(window = "periodic")))

rest_cust_stl %>%
  components() %>%
  autoplot() +
    labs(title = "KWH STL decomposition",
         x = "Date by month and year",
         y = "KWH") +
    scale_y_continuous(labels = scales::comma_format()) +
    theme_minimal()
```



This decomposition shows a steady upward trend in KWH usage over time, indicating a gradual increase in power consumption. There is clear seasonality with regular fluctuations each year, but the remainder component highlights some erratic behavior and variability that isn't explained by the trend or seasonal components.

```
respower_la <- rest_cust_fd %>%
  features(kwh, features = guerrero) %>%
  pull(lambda_guerrero)

aus_box_respower <- rest_cust_fd %>%
  mutate(respower_bc = box_cox(kwh, lambda = respower_la))

respower_stl <- aus_box_respower %>%
  model(kwh_stl = STL(respower_bc ~ trend() + season(window = "periodic")))
```

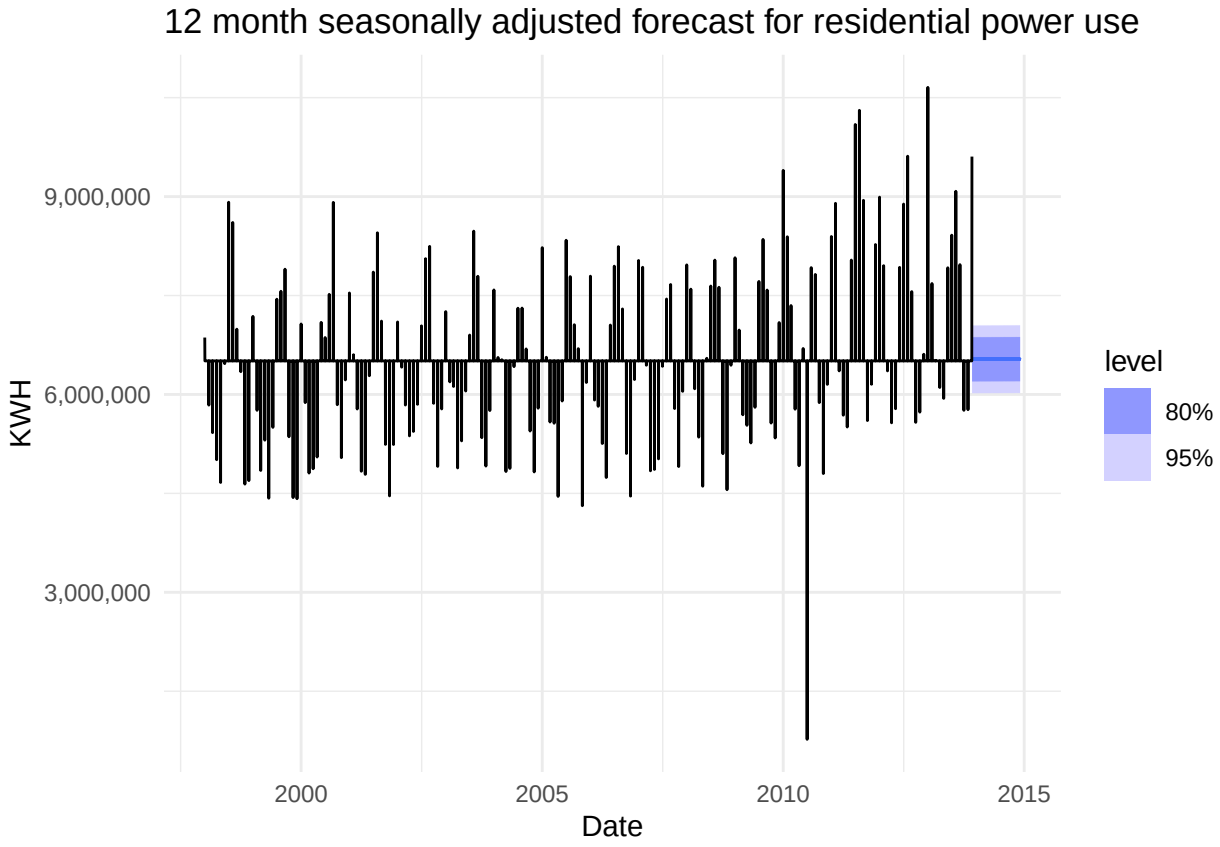
```
respower_season <- respower_stl %>%
  components() %>%
  select(year_month, season_adjust = season_adjust)

respower_ets <- respower_season %>%
  model(MAM = ETS(season_adjust ~ error("M") + trend("A") + season("N")),
        MAdM = ETS(season_adjust ~ error("M") + trend("Ad") + season("N")))


restpower_fc <- respower_ets %>%
  forecast(h = "12 months")

restpower_fc %>%
  autoplot(respower_season, level=NULL) +
  labs(title = "12 month seasonally adjusted forecast for residential power use",
       x = "Year",
       y = "Adjusted KWH") +
  scale_y_continuous(labels = scales::comma_format()) +
  theme_minimal()
```



Both models are closely aligned, suggesting similar predictions, though MAM appears to project a slightly higher trend in the short term. The historical volatility is evident, but the forecasts suggest a relatively stable and consistent growth moving forward.

However, I don't really like them because they're not capturing the seasonality the way I hope they would. I'm going to use the prophet model instead, which was designed to handle data with strong seasonality with complex patterns. Power usage is historically volatile so a more complex model is warranted.

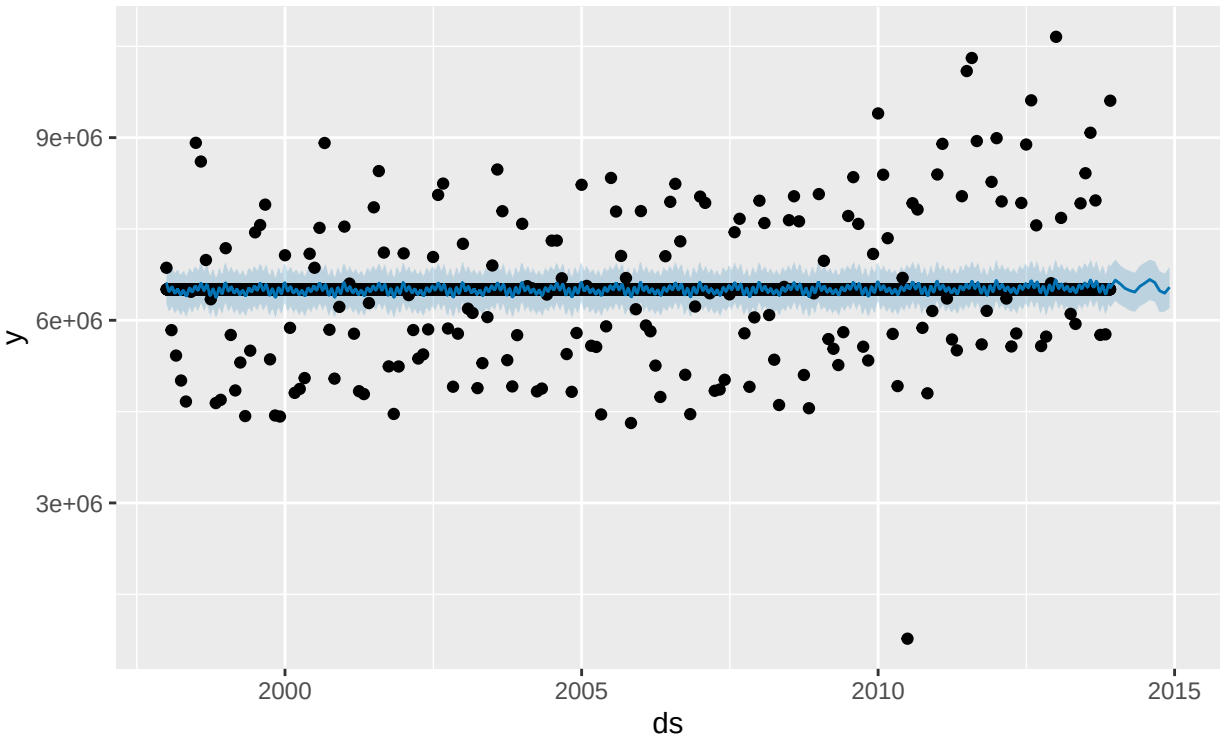
```
rest_cust_fcs <- rest_cust_fd %>%
  model(ARIMA(kwh ~ pdq(0,1,1) + PDQ(0,1,1))) %>%
  forecast(h = "12 months")

rest_cust_fcs %>%
  autoplot(rest_cust_fd) +
    labs(title = "12 month seasonally adjusted forecast for residential power use",
       x = "Date",
       y = "KWH") +
    scale_y_continuous(labels = scales::comma_format()) +
    theme_minimal()
```




```
rest_cust_prophet <- res_cust_forecast_load %>%
  as_tsibble(index = year_month) %>%
  fill_gaps() %>%
  as_tibble() %>%
  rename(ds = year_month, y = kwh) %>%
  mutate(y = ifelse(is.na(y), mean(y, na.rm = TRUE), y))

res_cust_pro <- prophet(rest_cust_prophet)
```

```
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
```

```
rc_pro_ft <- make_future_dataframe(res_cust_pro, periods = 12, freq = "month")
rc_prophet_fc <- predict(res_cust_pro, rc_pro_ft)

plot(res_cust_pro, rc_prophet_fc)
```




```
cat("\n This is a 12 month prophet forecast")
```

```
##
##  This is a 12 month prophet forecast
```

This shows a steady central trend with minimal expected growth but fails to fully capture the high level of volatility present in the historical data. The model's confidence intervals are narrow and tightly bound around the central forecast, indicating an overly optimistic and high degree of certainty in the model, given the scattered outliers. Overall, it performs fine for the general pattern but underestimates the intense fluctuations in the historical record.

End of assignment

I have opted against question C.