DATA 624 - Project 1: ATM Forecast

Richie Rivera

From the assignment description:

> 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.

Interpreting why this forecast is necessary, I will approach this problem as someone tasked with maintaining an acceptable the service levels for these 4 ATMs. As the target service level isn't provided, I will then define the objective will be to determine how much money is required per ATM to ensure that there is a sufficient amount at least 95% of the time.

1. Loading the data

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

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

# Specify the file path and read the Excel file
file_path <- "data/ATM624Data.xlsx"
atm <- read_excel(file_path)

# make all column names lowercase
atm <- atm |>
  rename_with(tolower)

glimpse(atm)
```

```
## 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, ~
```

2. Investigating the ATM Data

It looks like the date has come through as the number of days since 1900-01-01. Excel stores date information in this format very often, so we'll need to use that to convert the date into a date object. With the date converted, we can then convert the data object to a tsibble.

```
# Converting the date column to a date datatype
atm <- atm |>
  mutate(
    date = as.Date(date, origin = "1900-01-01")
  )

# Converting the dataset into a tsibble
atm_ts <- atm |>
  as_tsibble(
    index = date,
    key = atm
  )

head(atm_ts)
```

```
## # A tsibble: 6 x 3 [1D]
## # Key:       atm [1]
##   date       atm    cash
##   <date>     <chr> <dbl>
## 1 2009-05-03 ATM1     96
## 2 2009-05-04 ATM1     82
## 3 2009-05-05 ATM1     85
## 4 2009-05-06 ATM1     90
## 5 2009-05-07 ATM1     99
## 6 2009-05-08 ATM1     88
```

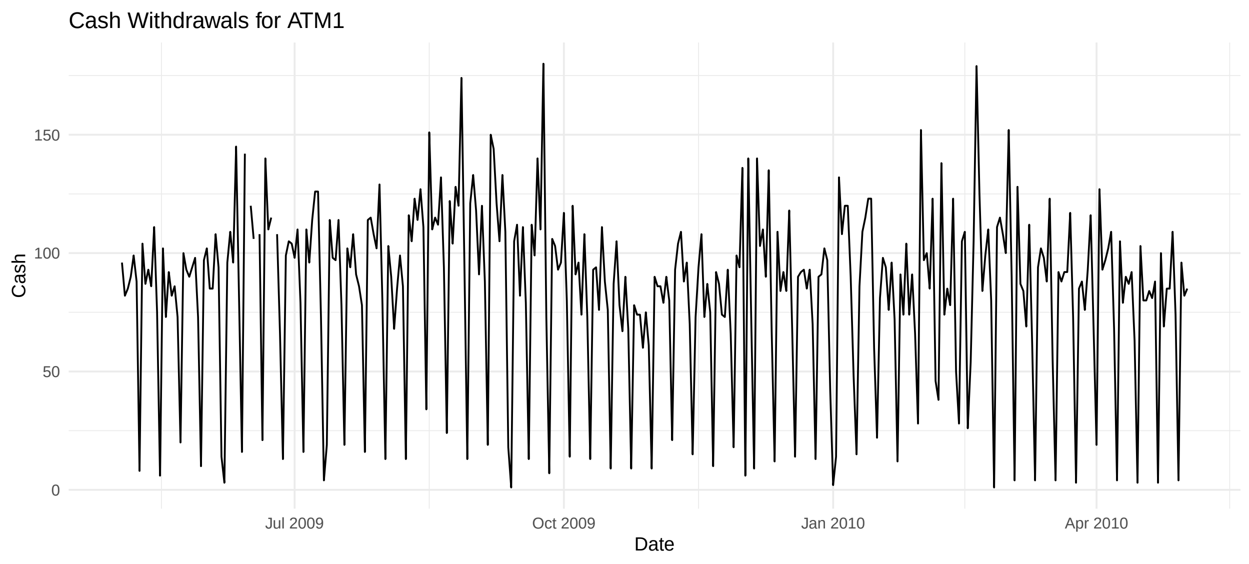
```
# See the number of records by group
atm |>
  count(atm)
```

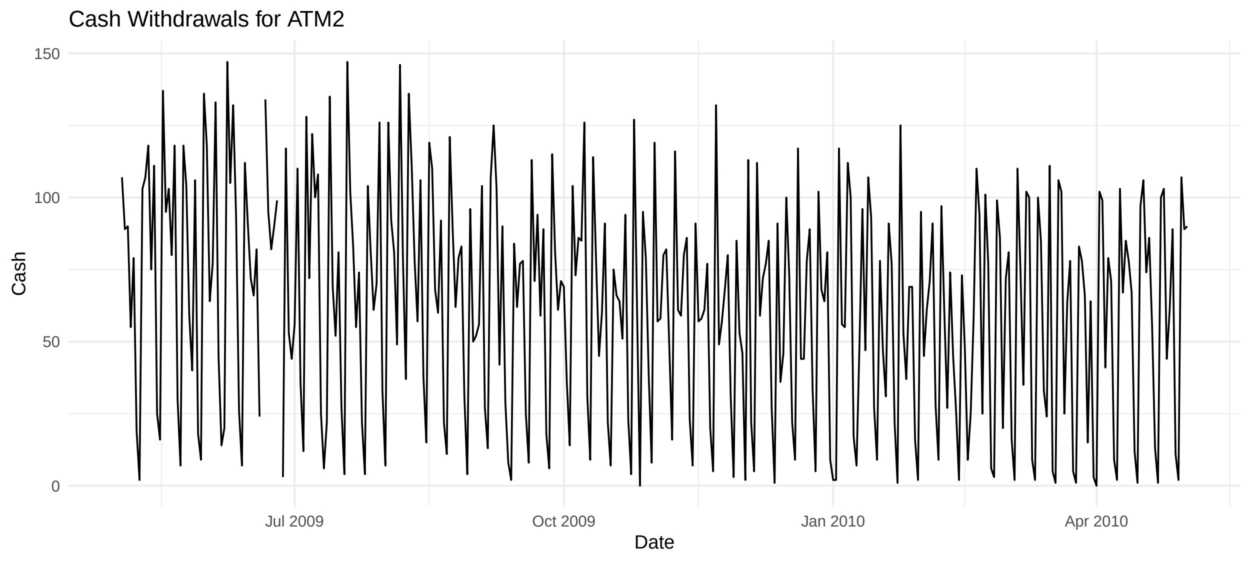
```
## # A tibble: 5 x 2
##   atm       n
##   <chr> <int>
## 1 ATM1    365
## 2 ATM2    365
## 3 ATM3    365
## 4 ATM4    365
## 5 <NA>     14
```

Additionally, there are 14 `NA` values for ATM. Given that our dataset has 365 observations of each other ATM and that we don't have any evidence of the NA values to be attributed to any other ATM, I'm leaning towards omitting these from our forecasts.

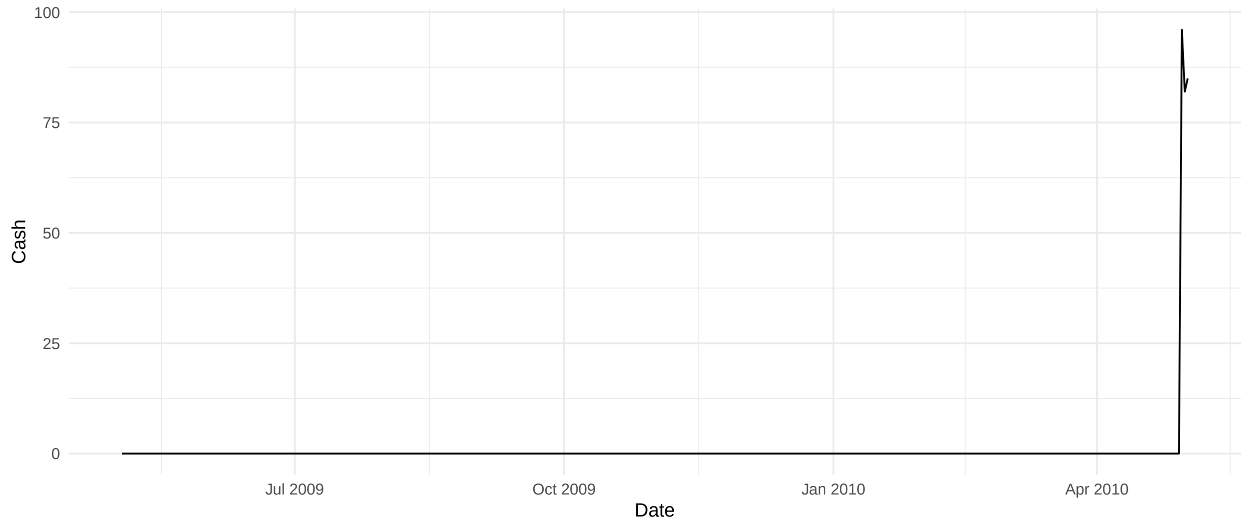
```r
for (plot_atm in unique(atm_ts$atm)) {
  plot <- atm_ts |>
    filter(atm == plot_atm) |>  # Filter data for the specific ATM
    autoplot(cash) +  # Replace 'cash' with the actual variable you want to plot
    labs(title = paste("Cash Withdrawals for", plot_atm),
         x = "Date",
         y = "Cash") +
    theme_minimal()

  print(plot)  # Print each plot
}
```

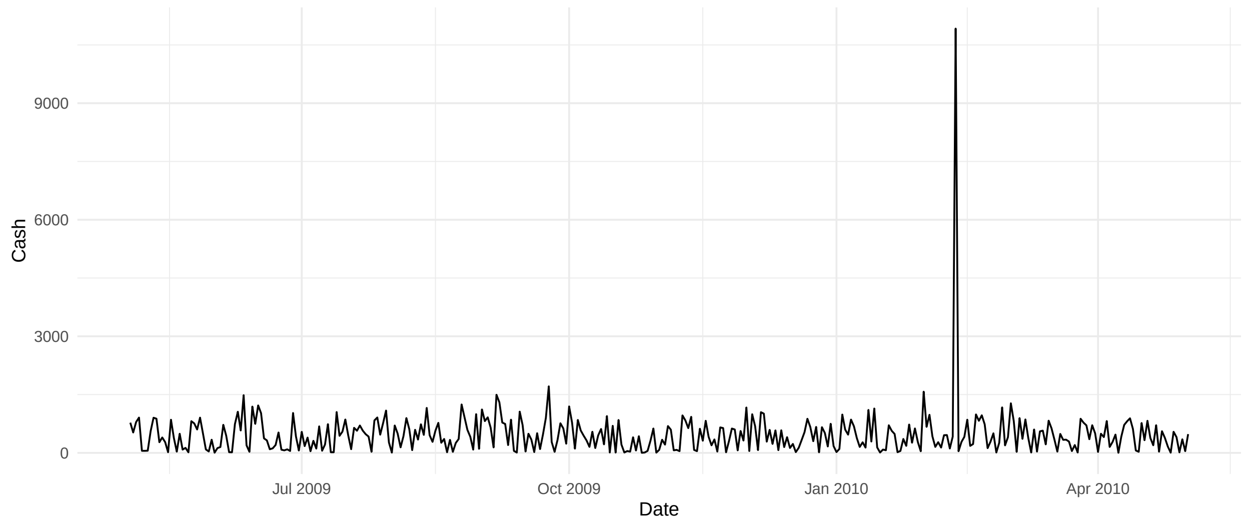




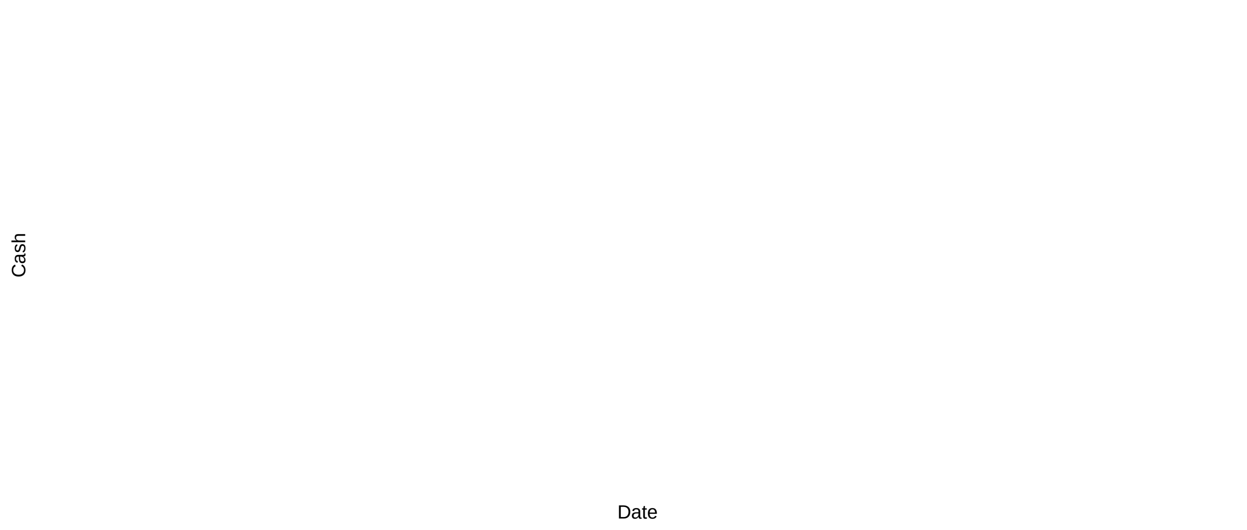
Cash Withdrawals for ATM3



Cash Withdrawals for ATM4



Cash Withdrawals for NA



There are a few observations we can make by looking at this data:

1. The `NA` group has no values, suggesting that the decision to remove the `NA` atm values will have no impact on our other timeseries.
2. **ATM 1**:
   - The cash withdrawn seems to be highly seasonal with a short seasonal cycle
3. **ATM 2**:
   - The cash withdrawn seems to be highly seasonal with a short seasonal cycle
4. **ATM 3**:
   - The cash withdrawn in ATM 3 has no activity until very recently
   - We may not be able to progress very far with ATM 3
5. **ATM 4**:
   - The cash withdrawn seems to be highly seasonal with a short seasonal cycle
   - There is an outlier with a greatly increased value than the rest of the timeseries

```
# Check for null entries
atm |>
  group_by(atm) |>
  summarise(
    null_count = sum(is.na(cash))
  )
```

Null checks

```
## # A tibble: 5 x 2
##   atm   null_count
##   <chr>      <int>
## 1 ATM1           3
## 2 ATM2           2
## 3 ATM3           0
## 4 ATM4           0
## 5 <NA>          14
```

From the above table, we can see that there are 3 null entries for `ATM1` and 2 null entries for `ATM2`. Due to the nature of our models, we will need this gap filled and to do so our only option seems to be imputation.

```
atm_ts |>
  filter(
    is.na(cash),
    !is.na(atm)
  )
```

```
## # A tsibble: 5 x 3 [1D]
## # Key:       atm [2]
##   date       atm    cash
##   <date>     <chr> <dbl>
## 1 2009-06-15 ATM1     NA
## 2 2009-06-18 ATM1     NA
## 3 2009-06-24 ATM1     NA
## 4 2009-06-20 ATM2     NA
## 5 2009-06-26 ATM2     NA
```

To handle this `NA` there are a few methods for imputation:

1. Mean - Use the mean value in the timeseries
2. Linear Interpolation
3. Forward Fill
4. Backward Fill

After reviewing these options, linear interpolation seems best, as it provides a value between the two points, which is likely to be realistic in most scenarios.

Data Pre-Processing

1. Filter out the `NA` Values
2. `ATM1` - Use linear interpolation to fill the `NA` values
3. `ATM2` - Use linear interpolation to fill the `NA` values
4. `ATM4` - Remove the one obvious outlier and use linear interpolation to fill the gap.

```r
# Filter out the NA atm values
atm_ts <- atm_ts |>
  filter(
    !is.na(atm)
  )

# creating a timeseries of just ATM1 and filling the gaps
atm1 <- atm_ts |>
  filter(
    atm == "ATM1"
  ) |>
  mutate(
    cash = na.approx(cash, na.rm = FALSE)
  )

# creating a timeseries of just ATM2 and filling the gaps
atm2 <- atm_ts |>
  filter(
    atm == "ATM2"
  ) |>
  mutate(
    cash = na.approx(cash, na.rm = FALSE)
  )
```

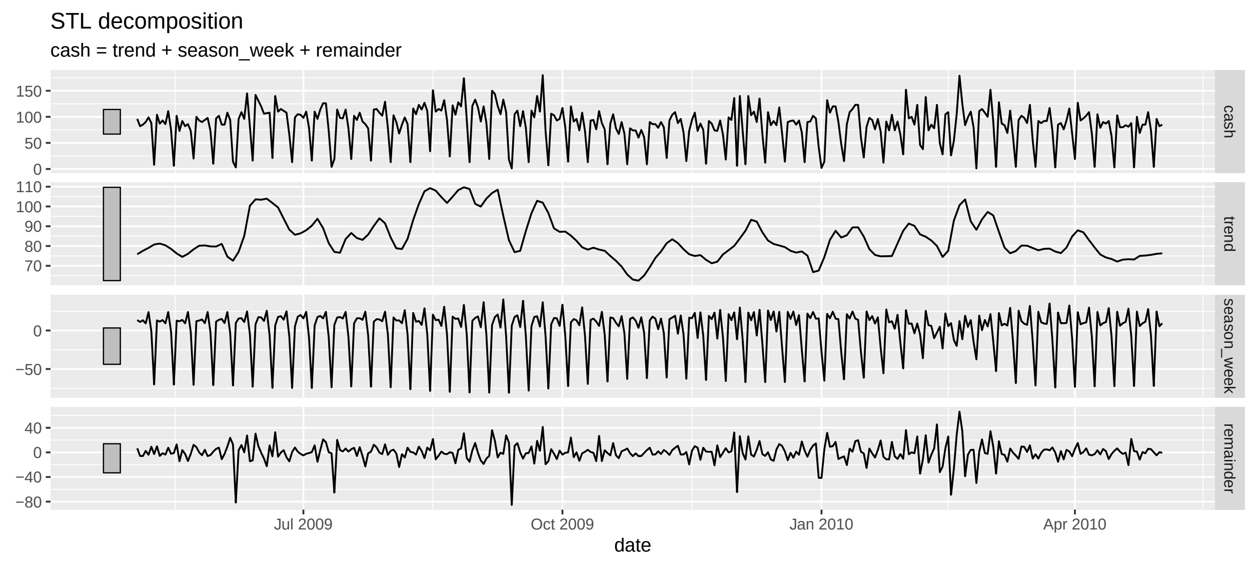
```r
# creating a timeseries of just ATM3
atm3 <- atm_ts |>
  filter(
    atm == "ATM3"
  )

# creating a timeseries of just ATM4
atm4 <- atm_ts |>
  filter(
    atm == "ATM4"
  ) |>
  mutate(
    cash = if_else(cash > 9000, NA_real_, cash)
  ) |>
  mutate(
    cash = na.approx(cash, na.rm = FALSE)
  )
```

ATM1 Forecast

For ATM1, we have a full timeseries. We'll start by looking at the `STL()` decomposition of ATM1:

```r
# Decomposing ATM1
atm1 |>
  model(stl = STL(cash)) |>
  components() |>
  autoplot()
```



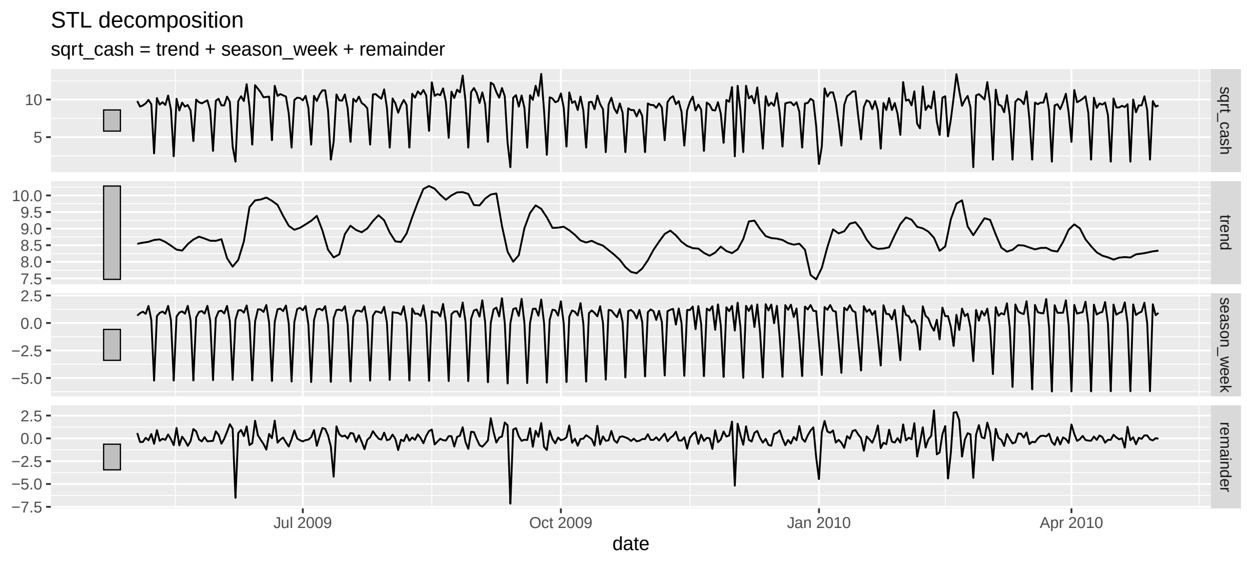
From the above `STL()` decomposition, we can see that the seasonal component has a window of a week with a significant peak and low value. We can also see that the seasonal component varies over time. Additionally, there doesn't seem to be a trend to the data. With this, we will normalize the data by obtaining the guerrero lambda value:

```
atm1_lambda <- atm1 |>
  features(cash, features = guerrero) |>
  pull(lambda_guerrero)
```

With a λ of 0.26, we can refer to the chart here and see that this transform is most similar to taking the square root.

```
atm1 <- atm1 |>
  mutate(sqrt_cash = sqrt(cash))

atm1 |>
  model(stl = STL(sqrt_cash)) |>
  components() |>
  autoplot()
```



With the data transformed, a few models make sense here to try:

1. `SNAIVE()` - Because there is not really a trend the seasonal NAIVE model may work here.
2. `ETS()` (Holt-Winters Additive Method) - For the same reason as the `SNAIVE()`. The seasonal variations are roughly constant, suggesting that the multiplicative method wouldn't be a good choice.
3. `ARIMA()` - With the built in differencing using the KPSS unit root test, we can apply an `ARIMA()` model.

To train our models, we will create a holdout group to test the accuracy of our model. The holdout window will be April 1st, 2010 onward.

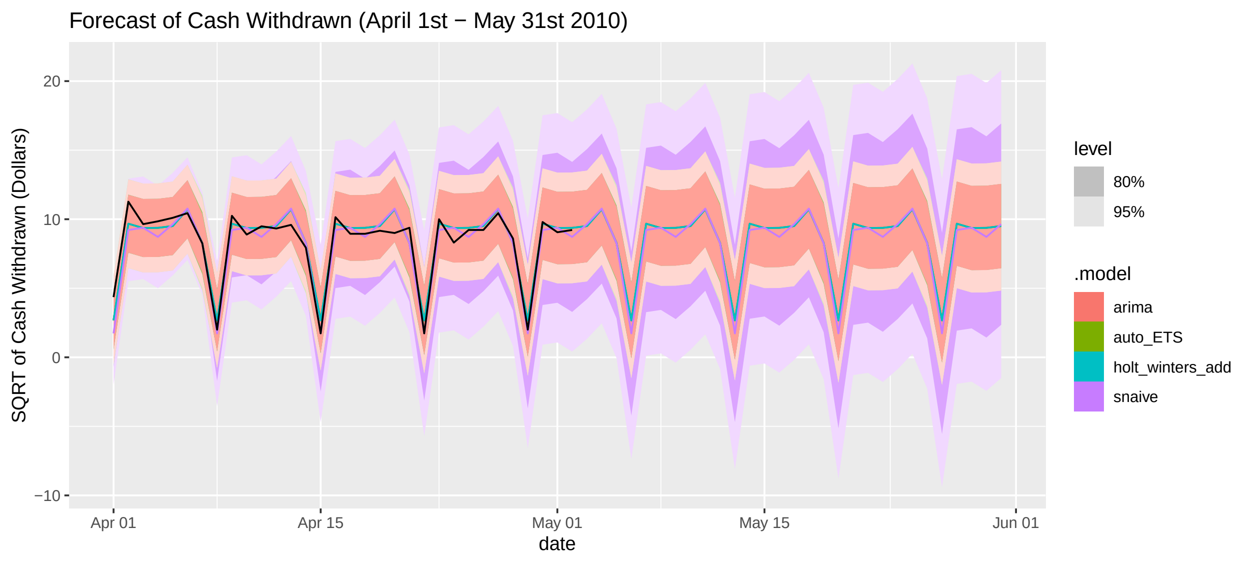
```
atm1_train <- atm1 |>
  filter(date < "2010-04-01")

atm1_test <- atm1 |>
  filter(date >= "2010-04-01")

atm1_fits <- atm1_train |>
  model(
    snaive = SNAIVE(sqrt_cash),
    auto_ETS = ETS(sqrt_cash),
    holt_winters_add = ETS(sqrt_cash ~ error("A") + trend("N") + season("A")),
    arima = ARIMA(sqrt_cash)
  )

atm1_fcs <- atm1_fits |>
  forecast(h = nrow(atm1_test) + 29)

atm1_fcs |>
  autoplot(
    atm1_test
  ) +
  labs(
    y = "SQRT of Cash Withdrawn (Dollars)",
    title = "Forecast of Cash Withdrawn (April 1st - May 31st 2010)"
  )
```



```
atm1_fits |>
  report()
```

```
## Warning in report.mdl_df(atm1_fits): Model reporting is only supported for
## individual models, so a glance will be shown. To see the report for a specific
## model, use ‘select()‘ and ‘filter()‘ to identify a single model.

## # A tibble: 4 x 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        3.60     NA    NA    NA    NA  NA    NA    NA     <NULL>
## 2 ATM1  auto_ETS      2.69  -1127. 2275. 2275. 2313.  2.62  2.63  0.961 <NULL>
## 3 ATM1  holt_winte~   2.69  -1127. 2275. 2275. 2313.  2.62  2.63  0.961 <NULL>
## 4 ATM1  arima         2.61   -619. 1247. 1247. 1262. NA    NA    NA     <cpl>
## # i 1 more variable: ma_roots <list>
```

```
atm1_fcs |>
  accuracy(atm1_test)
```

```
## Warning: The future dataset is incomplete, incomplete out-of-sample data will be treated as missing.
## 29 observations are missing between 2010-05-03 and 2010-05-31
```

```
## # A tibble: 4 x 11
##   .model         atm   .type      ME  RMSE   MAE   MPE  MAPE  MASE RMSSE    ACF1
##   <chr>          <chr> <chr>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl>
## 1 arima          ATM1  Test  -0.0974 0.736 0.577 -5.04 11.2    NaN   NaN -0.0208
## 2 auto_ETS       ATM1  Test  -0.0971 0.736 0.577 -5.04 11.2    NaN   NaN -0.0197
## 3 holt_winters_~ ATM1  Test  -0.0971 0.736 0.577 -5.04 11.2    NaN   NaN -0.0197
## 4 snaive         ATM1  Test   0.200  0.889 0.667  3.51  8.65   NaN   NaN  0.0972
```

From the graph we can see that each forecast picked up on the seasonality of the data well. Using the testing dataset as a way to gauge the performance, we can see that the `SNAIVE()` has the lowest MAPE although the `ARIMA()` model has the lowest AIC and AICc. It also appears that the auto-selected `ETS()` model has the same results as our Holt-Winters Additive Model.

Despite the `ARIMA()` model having a worse MAPE than the `SNAIVE()` I believe it's the best model available because it's AICc and AIC are much lower than that of the other models.

```
atm1_fits |>
  select(.model = "arima") |>
  report()
```

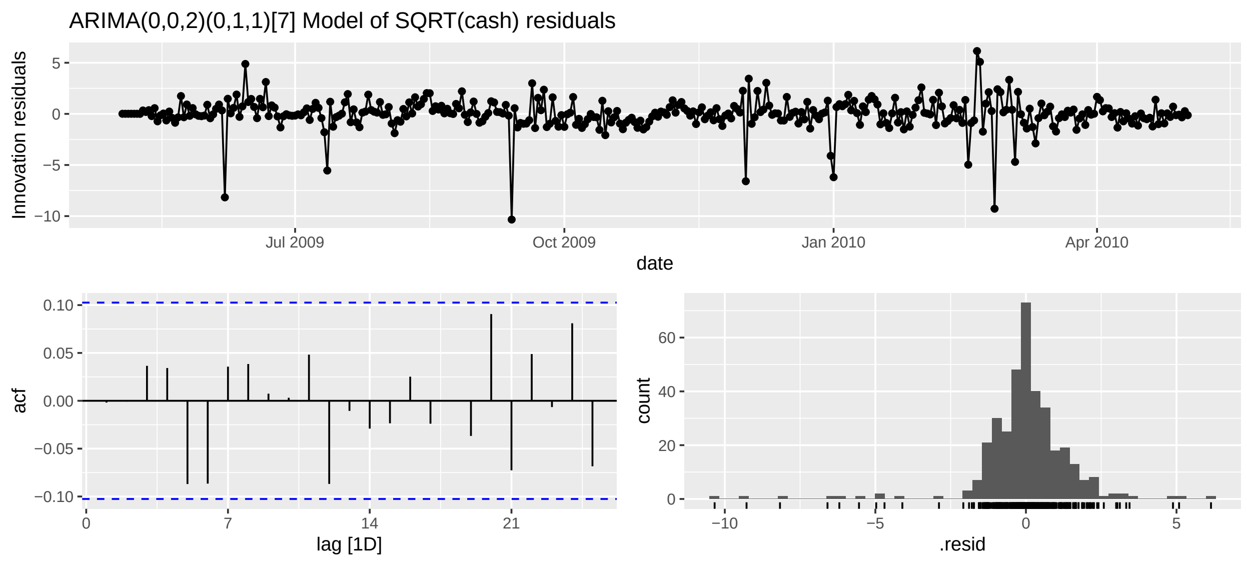
```
## Series: sqrt_cash
## Model: ARIMA(0,0,2)(0,1,1)[7]
##
## Coefficients:
##          ma1      ma2     sma1
##       0.1457  -0.1021  -0.6279
## s.e.  0.0547   0.0535   0.0503
##
## sigma^2 estimated as 2.614:  log likelihood=-619.46
## AIC=1246.93   AICc=1247.05   BIC=1262.08
```

We'll now retrain the model with the full dataset:

```
atm1_final_fit <- atm1 |>
  model(
    arima = ARIMA(sqrt_cash ~ pdq(0, 0, 2) + PDQ(0, 1, 1, period = 7))
  )

atm1_final_fc <- atm1_final_fit |>
  forecast(h = 29)

atm1_final_fit |>
  select(.model = "arima") |>
  gg_tsresiduals() +
  labs(
    title = "ARIMA(0,0,2)(0,1,1)[7] Model of SQRT(cash) residuals"
  )
```



Our residuals seem to be white noise and normally distributed with no autocorrelations above the critical point. A few business rules will need to be taken into consideration. An example of one would be that we must round up to match the smallest denomination of dollars that we can dispense and that we must know when our stock days are so that we can account for supply constraints.

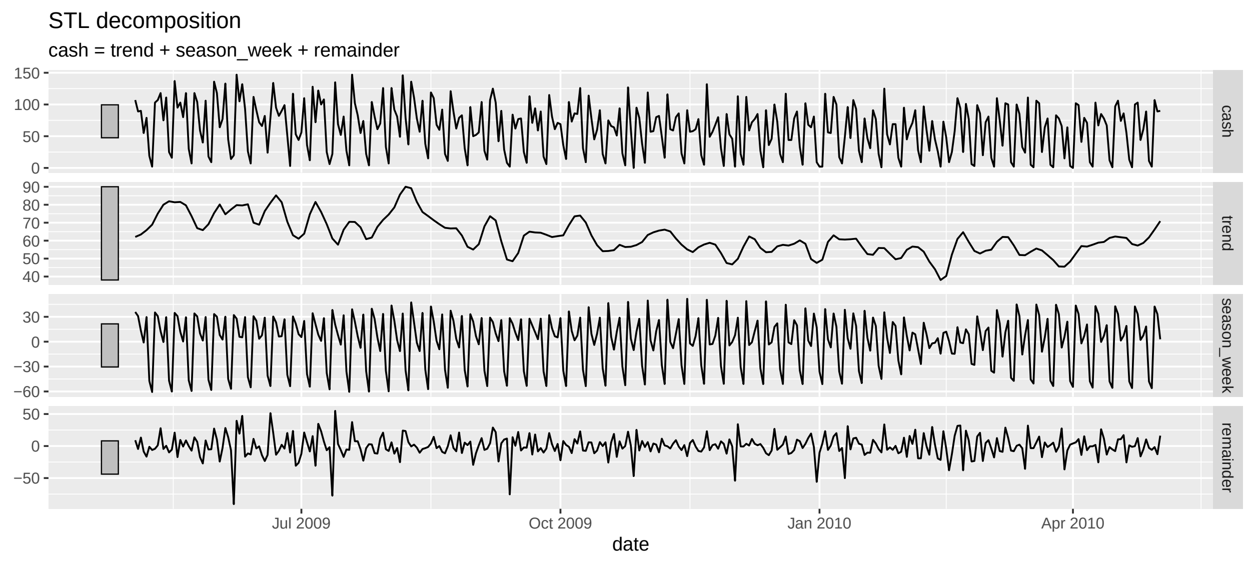
The last thing we need to do is to square the resulting prediction in order to have the models forecast be in dollars.

```
# We must square our results to bring the dimensions back to Cash.
atm1_final_fc |>
  as_tibble() |>
  filter(.model == "arima") |>
  mutate(
    cash_lower_ci_95 = hilo(sqrt_cash)$lower ^ 2,
    cash_prediction = mean(sqrt_cash) ^ 2,
    cash_upper_ci_95 = hilo(sqrt_cash)$upper ^ 2
  ) |>
  select(.model, date, cash_prediction, cash_lower_ci_95, cash_upper_ci_95) |>
  write_csv("forecasts/atm1_forecast_ci_ARIMA.csv")
```

ATM 2 Forecast

Just as we had with ATM1, we have a full history for ATM2 and we will start with an `STL()` to see the components:

```
# Decomposing ATM2
atm2 |>
  model(stl = STL(cash)) |>
  components() |>
  autoplot()
```



Here we can see that this data doesn't really seem to have much trend and is highly seasonal with a seasonal window of one week, just as we saw with ATM1. With that, we can follow a similar process as we did with ATM1 here:

```
atm2_lambda <- atm2 |>
  features(cash, features = guerrero) |>
  pull(lambda_guerrero)
```

With a λ of 0.72, we can refer to the chart here and see that this transform is pretty close to doing nothing, so we'll do nothing.

With that, there are a few models that make sense to try:

1. `SNAIVE()` - Because there is not really a trend the seasonal NAIVE model may work here.
2. `ETS()` (Holt-Winters Additive Method) - For the same reason as the `SNAIVE()`. The seasonal variations are roughly constant, suggesting that the multiplicative method wouldn't be a good choice.
3. `ARIMA()` - With the built in differencing using the KPSS unit root test, we can apply an `ARIMA()` model.

To train our models, we will create a holdout group to test the accuracy of our model. The holdout window will be April 1st, 2010 onward.

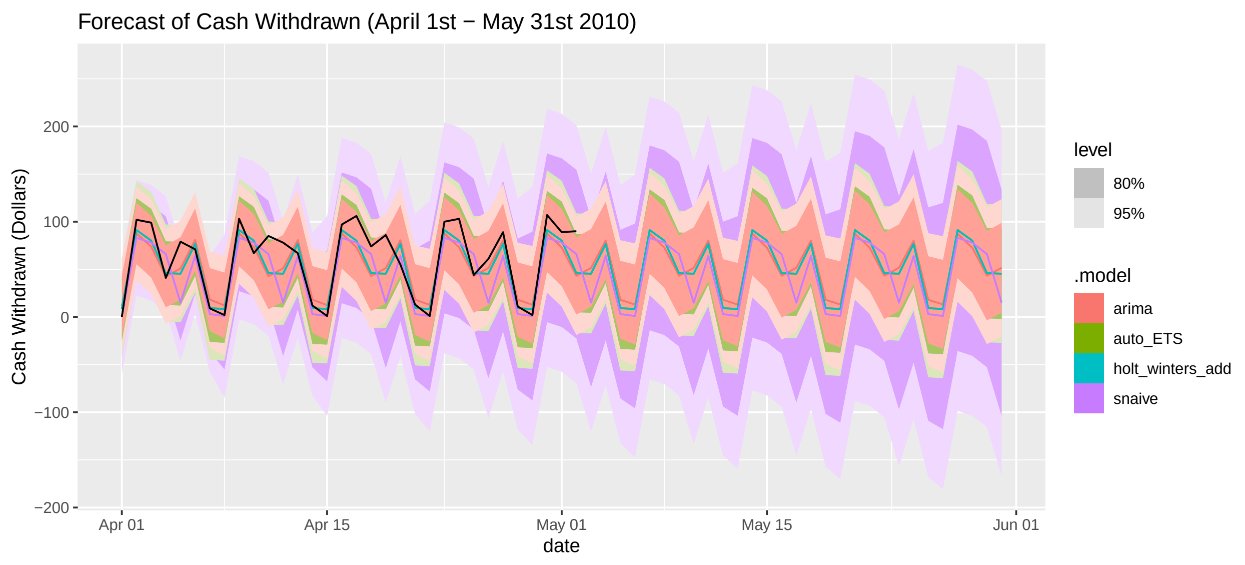
```
atm2_train <- atm2 |>
  filter(date < "2010-04-01")

atm2_test <- atm2 |>
  filter(date >= "2010-04-01")

atm2_fits <- atm2_train |>
  model(
    snaive = SNAIVE(cash),
    auto_ETS = ETS(cash),
    holt_winters_add = ETS(cash ~ error("A") + trend("N") + season("A")),
    arima = ARIMA(cash)
  )
```

```
atm2_fcs <- atm2_fits |>
  forecast(h = nrow(atm2_test) + 29)

atm2_fcs |>
  autoplot(
    atm2_test
  ) +
  labs(
    y = "Cash Withdrawn (Dollars)",
    title = "Forecast of Cash Withdrawn (April 1st - May 31st 2010)"
  )
```



```
atm2_fits |>
  report()
```

```
## Warning in report.mdl_df(atm2_fits): Model reporting is only supported for
## individual models, so a glance will be shown. To see the report for a specific
## model, use `select()` and `filter()` to identify a single model.

## # A tibble: 4 x 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  snaive          956.      NA    NA    NA    NA    NA    NA    NA  <NULL>
## 2 ATM2  auto_ETS        678.   -2048. 4116. 4117. 4154.  660.  662.  18.2 <NULL>
## 3 ATM2  holt_winter~    678.   -2048. 4116. 4117. 4154.  660.  662.  18.2 <NULL>
## 4 ATM2  arima           629.   -1513. 3038. 3038. 3061.   NA    NA    NA  <cpl>
## # i 1 more variable: ma_roots <list>
```

```
atm2_fcs |>
  accuracy(atm2_test)
```

```
## Warning: The future dataset is incomplete, incomplete out-of-sample data will be treated as missing.
## 29 observations are missing between 2010-05-03 and 2010-05-31
## # A tibble: 4 x 11
##   .model           atm   .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE    ACF1
##   <chr>            <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl>
## 1 arima            ATM2  Test   8.24  20.6  17.0  -Inf   Inf   NaN   NaN  0.112
## 2 auto_ETS         ATM2  Test   9.08  19.2  14.9  -Inf   Inf   NaN   NaN -0.0969
## 3 holt_winters_add ATM2  Test   9.08  19.2  14.9  -Inf   Inf   NaN   NaN -0.0969
## 4 snaive           ATM2  Test  14.9   26.4  19.1  -Inf   Inf   NaN   NaN -0.393
```

Similar to ATM1, the methods outlined here seemed to pick up the seasonality of ATM2 well. Additionally, we can see that the AIC and AICc are lowest for the `ARIMA()` model and that the `ARIMA()` model has a pretty comparable RMSE to the Holts-Winters Additive Model.

The MAPE for this model is infinity and as a result we aren't able to use it to compare the results of the model. But that being said, it seems that the `ARIMA()` has a much better AIC with a very comparable RMSE to the `ETS()` models, so we will select that model for this ATM's forecast. Because we did not perform any transformations, we won't need to undo any:

```r
atm2_fits |>
  select(.model = "arima") |>
  report()
```

```
## Series: cash
## Model: ARIMA(2,0,2)(0,1,1)[7]
##
## Coefficients:
##           ar1      ar2     ma1     ma2     sma1
##       -0.4339  -0.9207  0.4854  0.8021  -0.7812
## s.e.   0.0542   0.0409  0.0902  0.0553   0.0422
##
## sigma^2 estimated as 628.9:  log likelihood=-1513.08
## AIC=3038.16   AICc=3038.43   BIC=3060.89
```

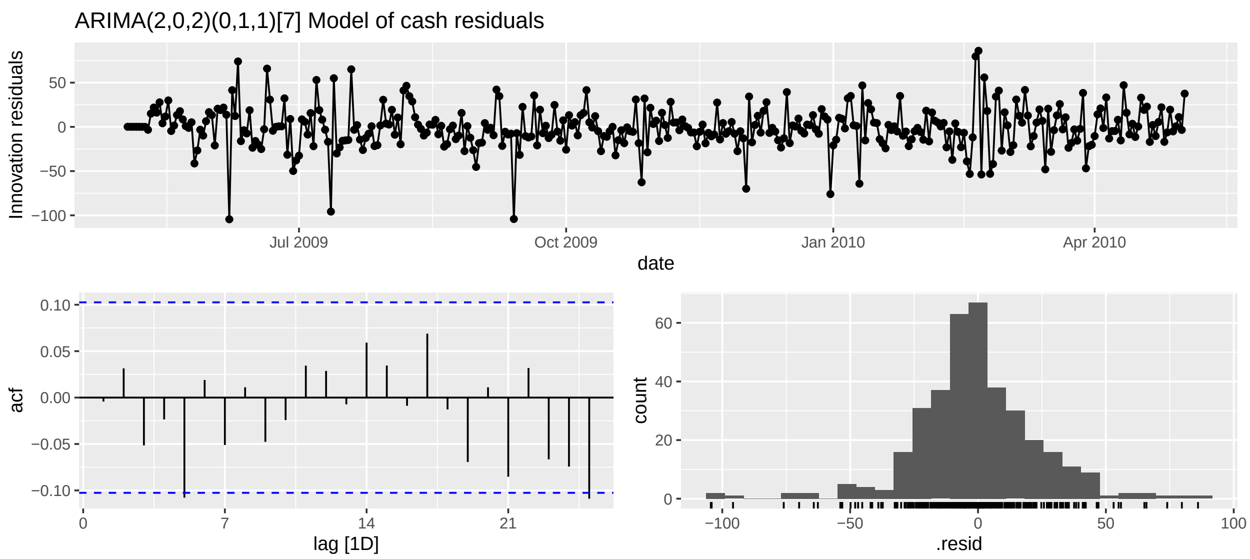
```r
atm2_final_fit <- atm2 |>
  model(
    arima = ARIMA(cash ~ pdq(2, 0, 2) + PDQ(0, 1, 1, period = 7))
  )

atm2_final_fc <- atm2_final_fit |>
  forecast(h = 29)

atm2_final_fit |>
  select(.model = "arima") |>
  report()
```

```
## Series: cash
## Model: ARIMA(2,0,2)(0,1,1)[7]
##
## Coefficients:
##           ar1      ar2     ma1     ma2     sma1
##       -0.4320  -0.9130  0.4773  0.8048  -0.7547
## s.e.   0.0553   0.0407  0.0861  0.0556   0.0381
##
## sigma^2 estimated as 602.5:  log likelihood=-1653.67
## AIC=3319.33   AICc=3319.57   BIC=3342.61
```

```r
atm2_final_fit |>
  select(.model = "arima") |>
  gg_tsresiduals() +
  labs(
    title = "ARIMA(2,0,2)(0,1,1)[7] Model of cash residuals"
  )
```



We can also see the residuals above where we see that there is one lag which is barely over the critical value and that the residuals are normally distributed.

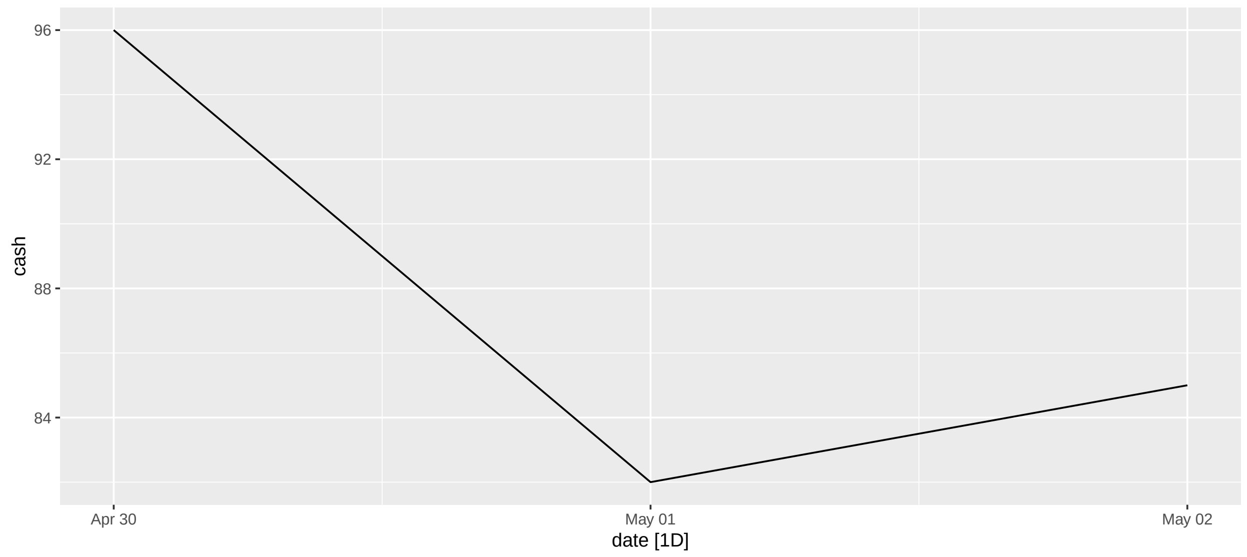
```r
atm2_final_fc |>
  as_tibble() |>
  filter(.model == "arima") |>
  mutate(
    cash_lower_ci_95 = hilo(cash)$lower,
    cash_prediction = mean(cash),
    cash_upper_ci_95 = hilo(cash)$upper
  ) |>
  select(.model, date, cash_prediction, cash_lower_ci_95, cash_upper_ci_95) |>
  write_csv("forecasts/atm2_forecast_ci_ARIMA.csv")
```

ATM3 Forecast

As we saw from our initial data exploration above, ATM3 only has data for the most recent few weeks. As such, it may be difficult to do something more sophisticated than a random walk model.

```r
atm3 |>
  filter(cash > 0) |>
  autoplot()
```

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

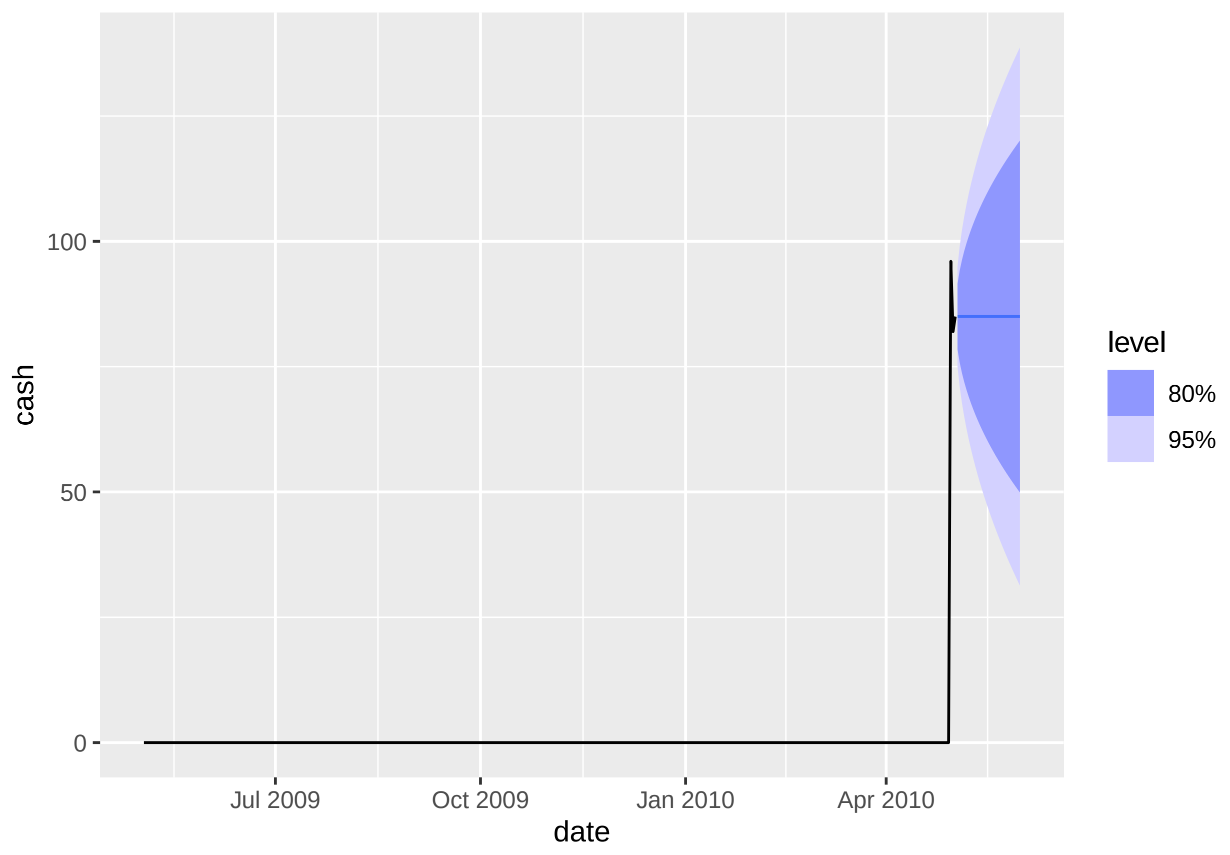


As we can see from the plot above, there is only 3 days of data available for ATM3. With that I would take a `NAIVE()` model and I don't believe that much else can be taken here:

```r
atm3_fit <- atm3 |>
  model(NAIVE(cash))

atm3_final_fc <- atm3_fit |>
  forecast(h = 29)

atm3_final_fc |>
  autoplot(atm3)
```



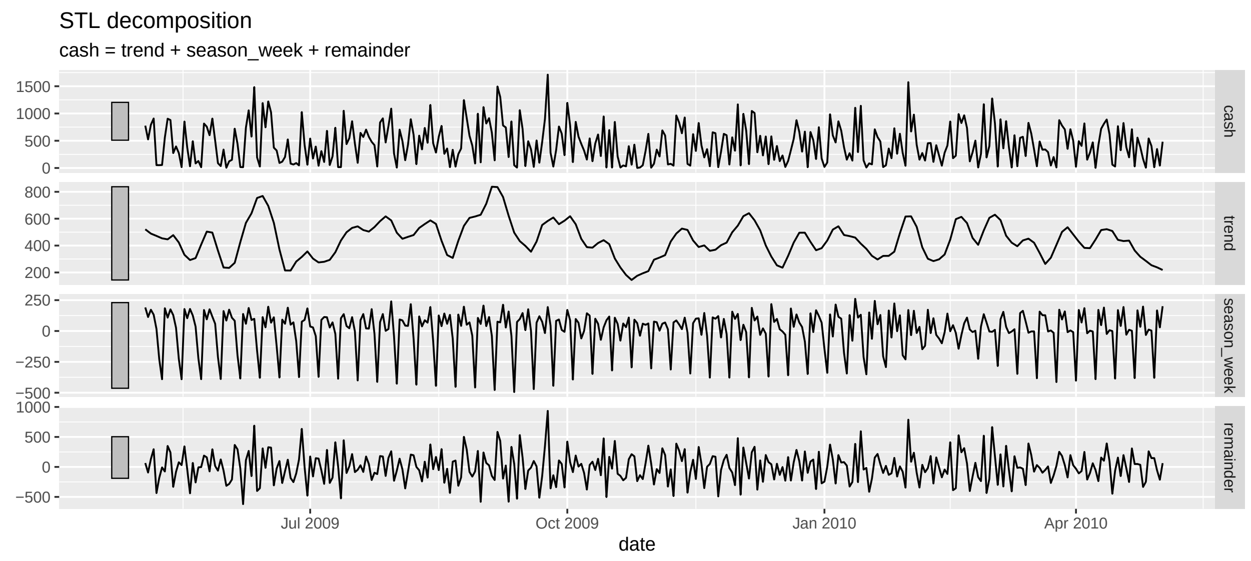
Again, because there is not much more data available on this dataset, I would recommend maintaining at least the upper 95% confidence interval given by a `NAIVE()` model and then revisiting this ATM once more data is available.

```
atm3_final_fc |>
  as_tibble() |>
  mutate(
    cash_lower_ci_95 = hilo(cash)$lower,
    cash_prediction = mean(cash),
    cash_upper_ci_95 = hilo(cash)$upper
  ) |>
  select(.model, date, cash_prediction, cash_lower_ci_95, cash_upper_ci_95) |>
  write_csv("forecasts/atm3_forecast_ci_NAIVE.csv")
```

ATM4 Forecast

Just as we did with ATM 1 and 2, we will see the `STL()` decomposition of this model to see the components:

```
# Decomposing ATM4
atm4 |>
  model(stl = STL(cash)) |>
  components() |>
  autoplot()
```



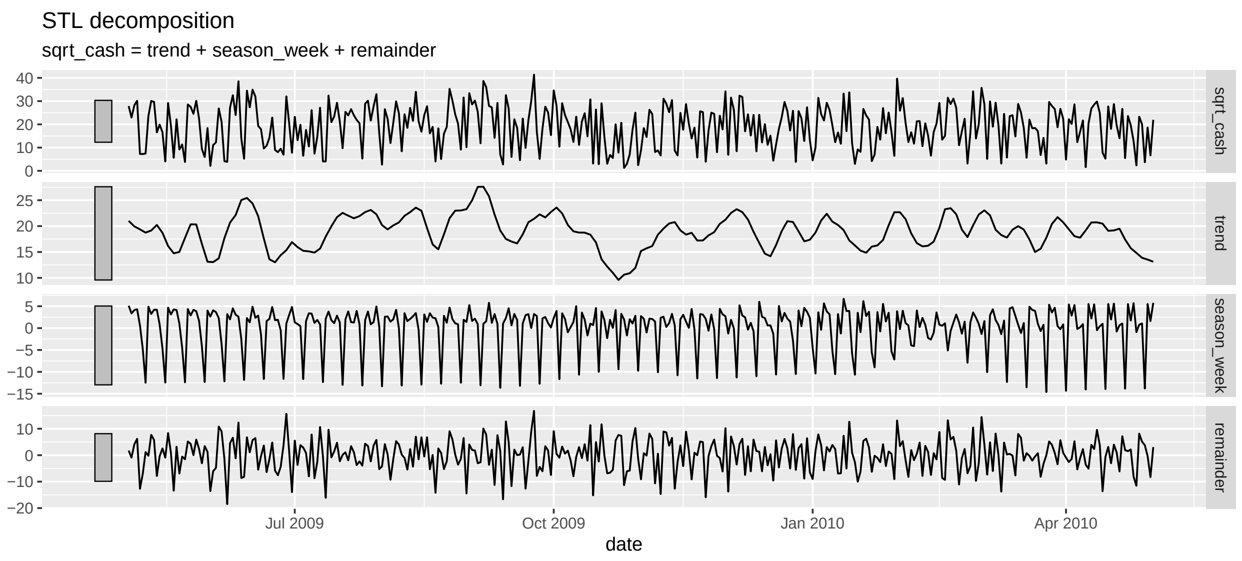
Here we can see that this data doesn't really seem to have much trend and is highly seasonal with a seasonal window of one week, just as we saw with ATM1. With that, we can follow a similar process as we did with ATM1 here:

```
atm4_lambda <- atm4 |>
  features(cash, features = guerrero) |>
  pull(lambda_guerrero)
```

With a λ of 0.45, we can refer to the chart here and see that this transform is most similar to taking the square root which we will do for the purpose of the model.

```
atm4 <- atm4 |>
  mutate(sqrt_cash = sqrt(cash))

atm4 |>
  model(stl = STL(sqrt_cash)) |>
  components() |>
  autoplot()
```



With that transform complete, there are a few models that make sense to try:

1. `SNAIVE()` - Because there is not really a trend the seasonal NAIVE model may work here.
2. `ETS()` (Holt-Winters Additive Method) - For the same reason as the `SNAIVE()`. The seasonal variations are roughly constant, suggesting that the multiplicative method wouldn't be a good choice.
3. `ARIMA()` - With the built in differencing using the KPSS unit root test, we can apply an `ARIMA()` model.

To train our models, we will create a holdout group to test the accuracy of our model. The holdout window will be April 1st, 2010 onward.

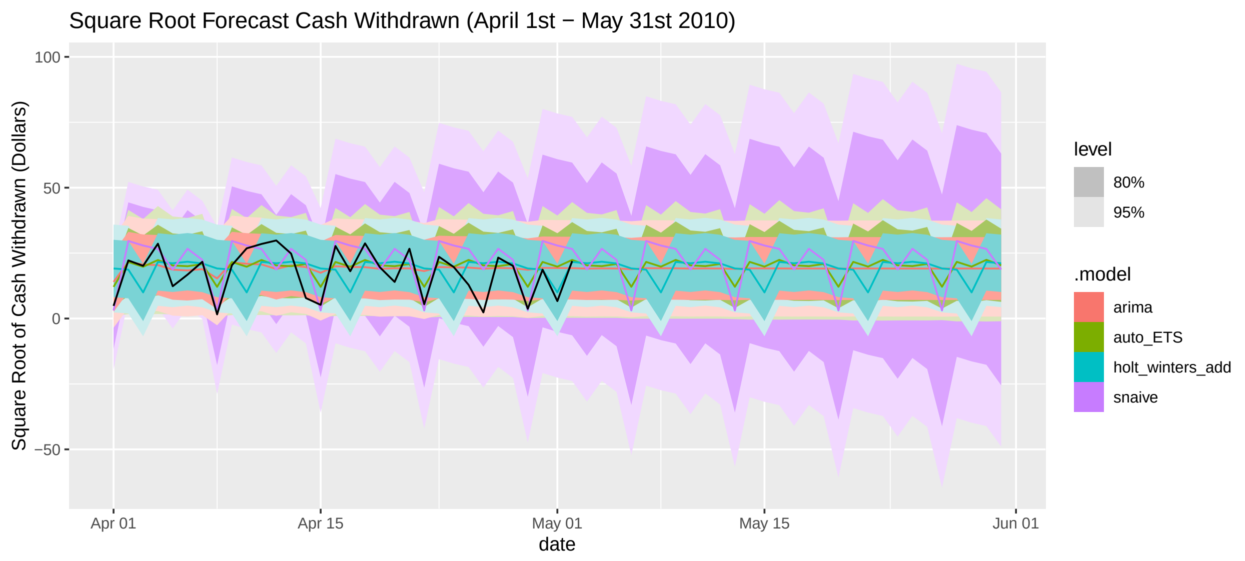
```r
atm4_train <- atm4 |>
  filter(date < "2010-04-01")

atm4_test <- atm4 |>
  filter(date >= "2010-04-01")

atm4_fits <- atm4_train |>
  model(
    snaive = SNAIVE(sqrt_cash),
    auto_ETS = ETS(sqrt_cash),
    holt_winters_add = ETS(sqrt_cash ~ error("A") + trend("N") + season("A")),
    arima = ARIMA(sqrt_cash)
  )

atm4_fcs <- atm4_fits |>
  forecast(h = nrow(atm4_test) + 29)

atm4_fcs |>
  autoplot(
    atm4_test
  ) +
  labs(
    y = "Square Root of Cash Withdrawn (Dollars)",
    title = "Square Root Forecast Cash Withdrawn (April 1st - May 31st 2010)"
  )
```



The models that seem to visually follow the actual line the most is the `SNAIVE()` and the `ETS()` models. That being said, it's pretty easy to see that the confidence intervals for the `SNAIVE()` model are the greatest among the models.

We will need to look at the model reports to know though:

```r
atm4_fits |>
  report()
```

```
## Warning in report.mdl_df(atm4_fits): Model reporting is only supported for
## individual models, so a glance will be shown. To see the report for a specific
## model, use ‘select()‘ and ‘filter()‘ to identify a single model.

## # A tibble: 4 x 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  snaive      133.         NA    NA    NA    NA  NA    NA   NA     <NULL>
## 2 ATM4  auto_ETS      0.221  -1675. 3371. 3371. 3409.  75.3  75.6  0.367 <NULL>
## 3 ATM4  holt_wint~   73.1    -1677. 3374. 3375. 3412.  71.2  71.3  6.80  <NULL>
## 4 ATM4  arima        79.9    -1200. 2411. 2411. 2430.  NA    NA   NA     <cpl>
## # i 1 more variable: ma_roots <list>
```

```r
atm4_fcs |>
  accuracy(atm4_test)
```

```
## Warning: The future dataset is incomplete, incomplete out-of-sample data will be treated as missing.
## 29 observations are missing between 2010-05-03 and 2010-05-31

## # A tibble: 4 x 11
##   .model           atm   .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE    ACF1
##   <chr>            <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl>
## 1 arima            ATM4  Test  -1.57  8.01  6.40 -89.5 108.    NaN   NaN -0.0867
## 2 auto_ETS         ATM4  Test  -1.86  7.01  5.55 -76.7  90.3   NaN   NaN  0.0524
## 3 holt_winters_add ATM4  Test  -1.19  9.42  7.74 -98.9 126.    NaN   NaN  0.0953
## 4 snaive           ATM4  Test  -4.47  8.22  6.28 -55.1  68.6   NaN   NaN  0.107
```

From the reports above we can see that, again, the `ARIMA()` model has the lowest AIC and AICc scores. When looking at how well the trained model performed on the test data, we can see that the MAPE of the `SNAIVE()` model was the best followed by the automatically selected `ETS()` model. In this case, it may be best to use the `ARIMA()` model despite the fact that it has a worse MAPE and RMSE than the `ETS()` model. This is because the AIC is much better than all of the other models and it gives us more confidence that the model isn't being overfit, allowing us to generalize the trend into the future.

```
atm4_fits |>
  select(.model = "arima") |>
  report()
```

```
## Series: sqrt_cash
## Model: ARIMA(0,0,1)(2,0,0)[7] w/ mean
##
## Coefficients:
##          ma1    sar1    sar2  constant
##       0.0822  0.1906  0.1749   12.1139
## s.e.  0.0545  0.0542  0.0549    0.5175
##
## sigma^2 estimated as 79.87:  log likelihood=-1200.25
## AIC=2410.5   AICc=2410.68   BIC=2429.54
```

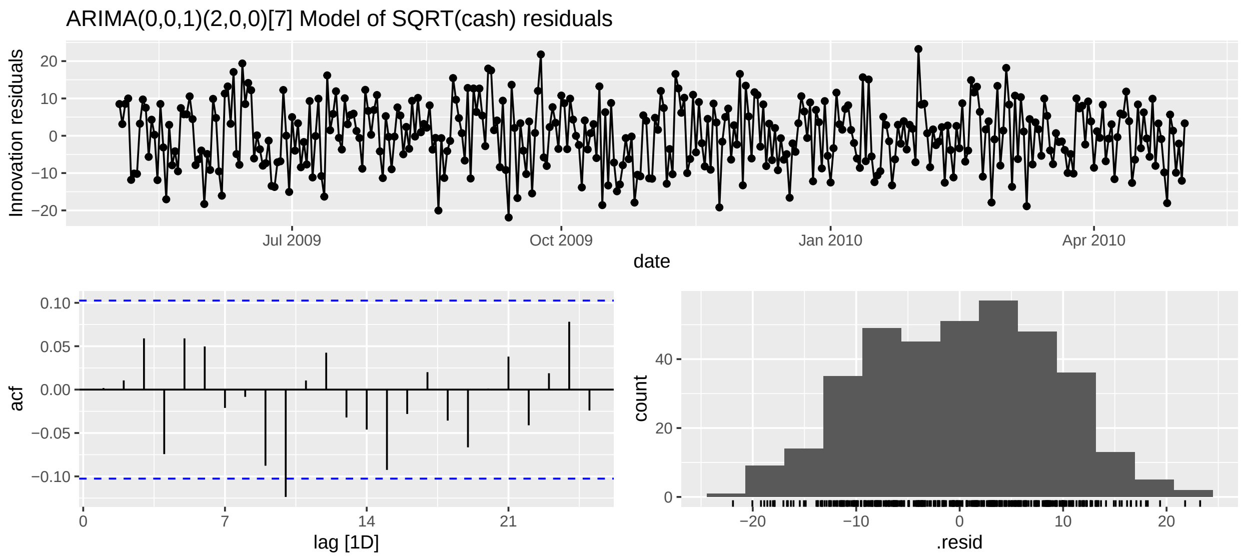
```
atm4_final_fits <- atm4 |>
  mutate(
    sqrt_cash = sqrt(cash)
  ) |>
  model(
    arima = ARIMA(sqrt_cash ~ pdq(0, 0, 1) + PDQ(2, 0, 0, period = 7))
  )

atm4_final_fc <- atm4_final_fits |>
  forecast(h = 29)

atm4_final_fits |>
  select(.model = "arima") |>
  report()
```

```
## Series: sqrt_cash
## Model: ARIMA(0,0,1)(2,0,0)[7] w/ mean
##
## Coefficients:
##          ma1    sar1    sar2  constant
##       0.0796  0.2021  0.1957   11.3740
## s.e.  0.0527  0.0517  0.0525    0.4866
##
## sigma^2 estimated as 77.81:  log likelihood=-1311.07
## AIC=2632.13   AICc=2632.3   BIC=2651.63
```

```
atm4_final_fits |>
  select(.model = "arima") |>
  gg_tsresiduals() +
  labs(
    title = "ARIMA(0,0,1)(2,0,0)[7] Model of SQRT(cash) residuals"
  )
```



In the case of this `ARIMA()` model also seems to have the residuals normally distributed and most of the ACF values are within the critical values.

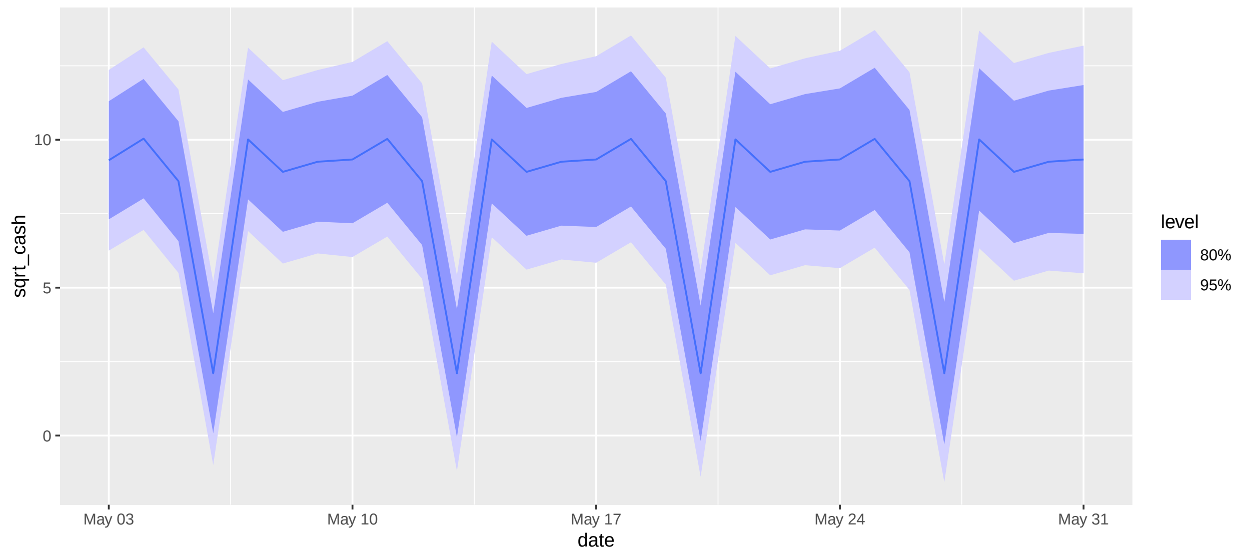
In order to export this forecast in a meaningful way, we will need to square the result before sharing them.

```
atm4_final_fc |>
  as_tibble() |>
  filter(.model == "arima") |>
  mutate(
    cash_lower_ci_95 = hilo(sqrt_cash)$lower ^ 2,
    cash_prediction = mean(sqrt_cash) ^ 2,
    cash_upper_ci_95 = hilo(sqrt_cash)$upper ^ 2
  ) |>
  select(.model, date, cash_prediction, cash_lower_ci_95, cash_upper_ci_95) |>
  write_csv("forecasts/atm4_forecast_ci_ARIMA.csv")
```

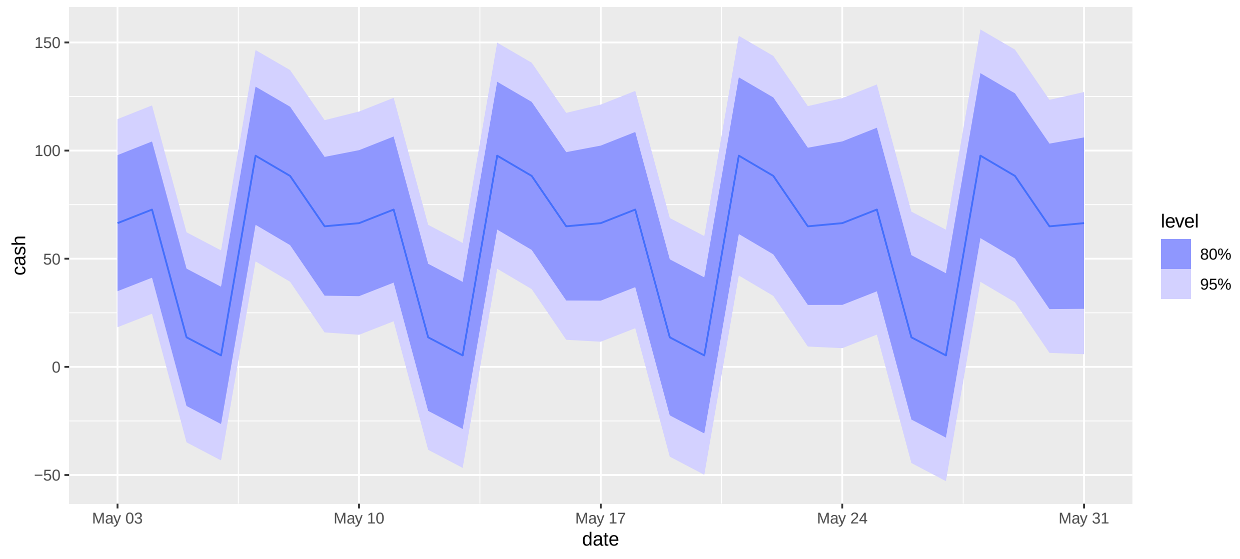
Results

With the models developed in this report, we were able to develop forecasts for the ATM's expected activity across the remainder of May. These are plotted below:

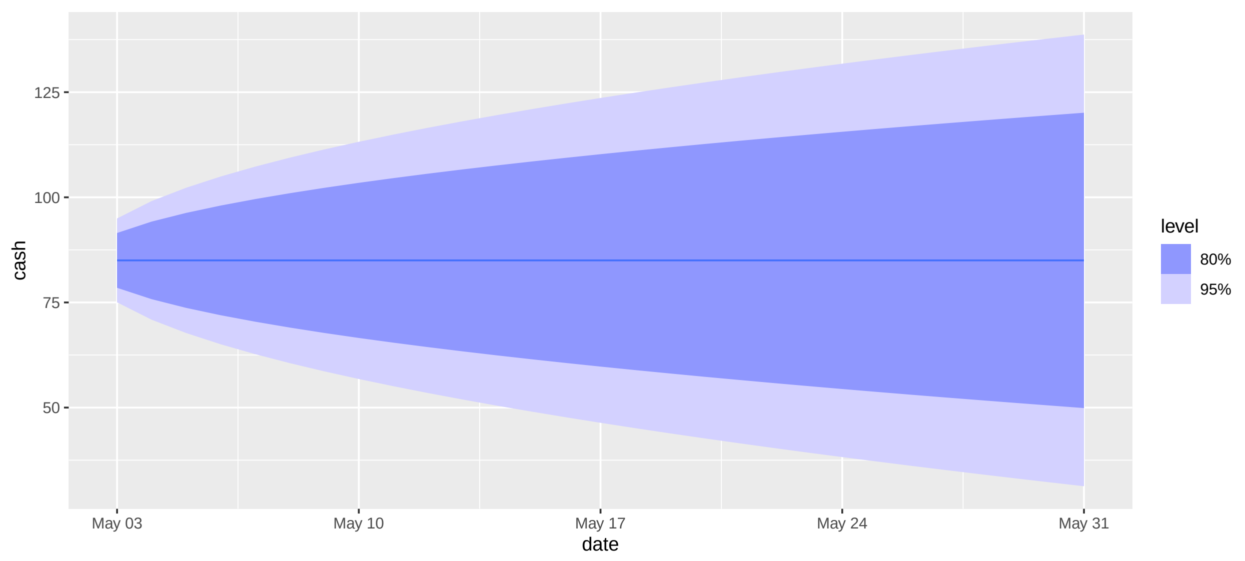
```
atm1_final_fc |>
  autoplot()
```



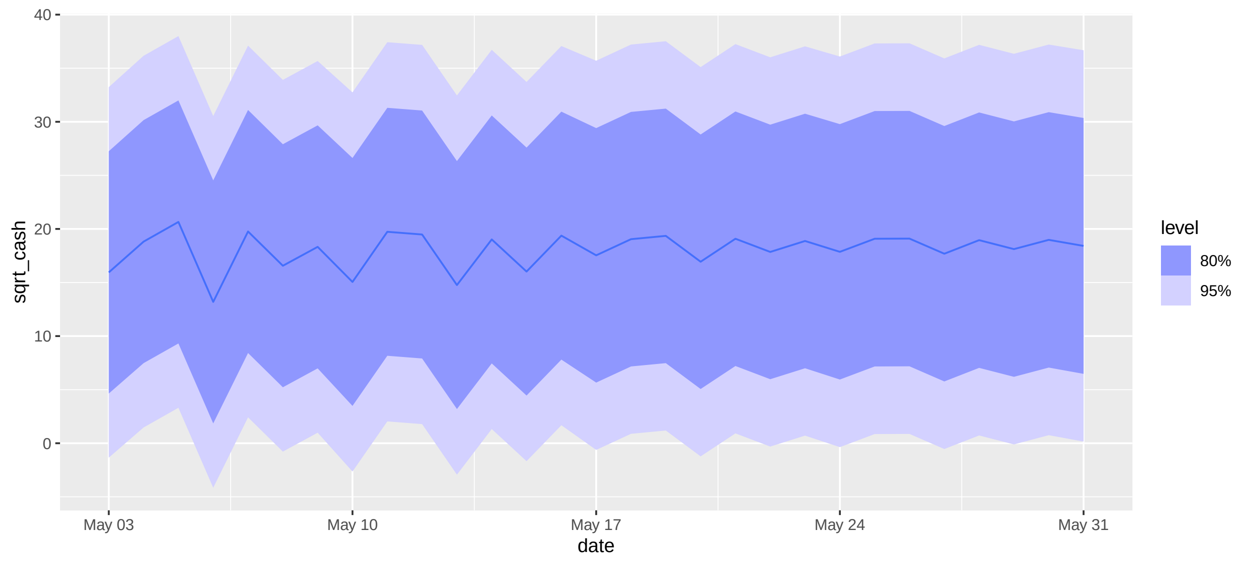
```r
atm2_final_fc |>
  autoplot()
```



```r
atm3_final_fc |>
  autoplot()
```



```r
atm4_final_fc |>
  autoplot()
```



It must be noted that the forecasts for `ATM1` and `ATM4` are shown in terms of the square root of cash. This transformation was done to make the data more normally distributed for the purposes of modeling.

A few notes on the forecasts:

- `ATM1` - The large seasonal impact remains after developing the model. Although the difference between the peaks and valleys will explode once squared, we can see here that it's highly seasonal and our forecast expects that to continue.
- `ATM2` - This ATM doesn't have a transform applied so we can see that the 95% service level amount for this ATM is between $50 and $150. Although it sees a dip, similar to `ATM1`, it isn't as drastic.
- `ATM3` - As we discussed, we don't have very much data for `ATM3`. As a result, a random walk model would be best to employ until more data is available.
- `ATM4` - This model seems to converge to a steadier value at around the mean. Although the values in this chart must be squared to represent dollars, we can see that the daily variation seems to drop off towards the end of the month.