DATA624 - Project1

Glen Dale Davis

2023-10-14

Packages:

```
library(fpp3)
library(RColorBrewer)
library(knitr)
library(pracma)
library(cowplot)
library(readxl)
library(httr)
library(writexl)
```

Part A:

Data Preparation:

We load transaction data for four ATMS from May 2009 to April 2010.

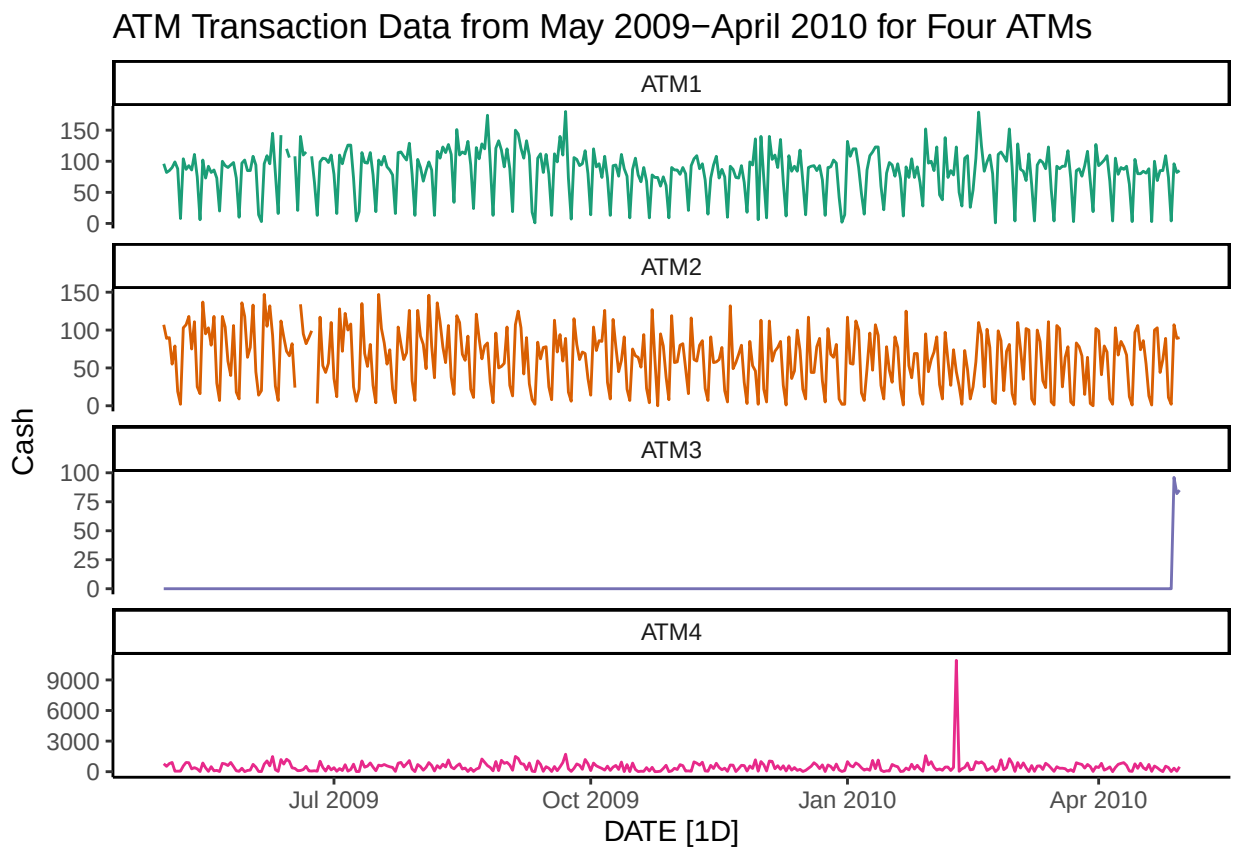
```
my_url <- "https://github.com/geedoubledee/data624_project1/raw/main/ATM624Data.xlsx"
col_types <- c("date", "text", "numeric")
temp <- tempfile(fileext = ".xlsx")
req <- GET(my_url, authenticate(Sys.getenv("GITHUB_PAT"), ""),
           write_disk(path = temp))
atm <- readxl::read_excel(temp, col_types = col_types)
```

We coerce the `DATE` variable to class *Date* and the `ATM` variable to class *Factor*. We remove observations that contain neither `ATM` nor `CASH` data. (These empty observations all occur during the period we will be forecasting for: May 2010.)

```
levels = c("ATM1", "ATM2", "ATM3", "ATM4")
atm <- atm |>
    mutate(DATE = as.Date(DATE),
           ATM = factor(ATM, levels = levels, ordered = TRUE)) |>
    filter(!is.na(ATM))
```

We create and plot the time series with subplots for each ATM, and we print a summary of 0 or `NA` values for each ATM.

```
theme_set(theme_classic())
palette <- brewer.pal(n = 8, name = "Dark2")
atm_colors <- palette[1:4]
names(atm_colors) <- levels
atm_ts <- atm |>
    as_tsibble(index = DATE, key = ATM)
p1 <- atm_ts |>
    autoplot(Cash) +
    facet_wrap(~ ATM, scales = "free_y" , ncol = 1) +
    scale_color_brewer(palette = "Dark2") +
    theme(legend.position = "none") +
    labs(title = "ATM Transaction Data from May 2009-April 2010 for Four ATMs")
p1
```




```
na_summary <- atm_ts |>
    as_tibble() |>
    filter(is.na(Cash) | Cash == 0) |>
    group_by(ATM, Cash) |>
    summarize(Count = n())
knitr::kable(na_summary)
```

ATM	Cash	Count
ATM1	NA	3

ATM	Cash	Count
ATM2	0	2
ATM2	NA	2
ATM3	0	362

Missing Data and Outliers:

We can immediately see missing value and outlier issues with the data that we will need to address. First, we will handle ATM3's data issues, as those need to be addressed differently from the other ATMs' data issues.

The distribution of `Cash` is degenerate for ATM3, as 99% of values are 0. The most likely interpretation of these values is that the machine was not operating during the period prior to the first non-zero recorded value. Since we will be forecasting for each ATM separately, and we therefore cannot exclude ATM3 from our analysis, we will treat these 0 values as missing and use Next Observation Carried Backward (NOCB) to fill them. (This technique is the same as Last Observation Carried Forward (LOCF), except it is performed in the opposite direction.)

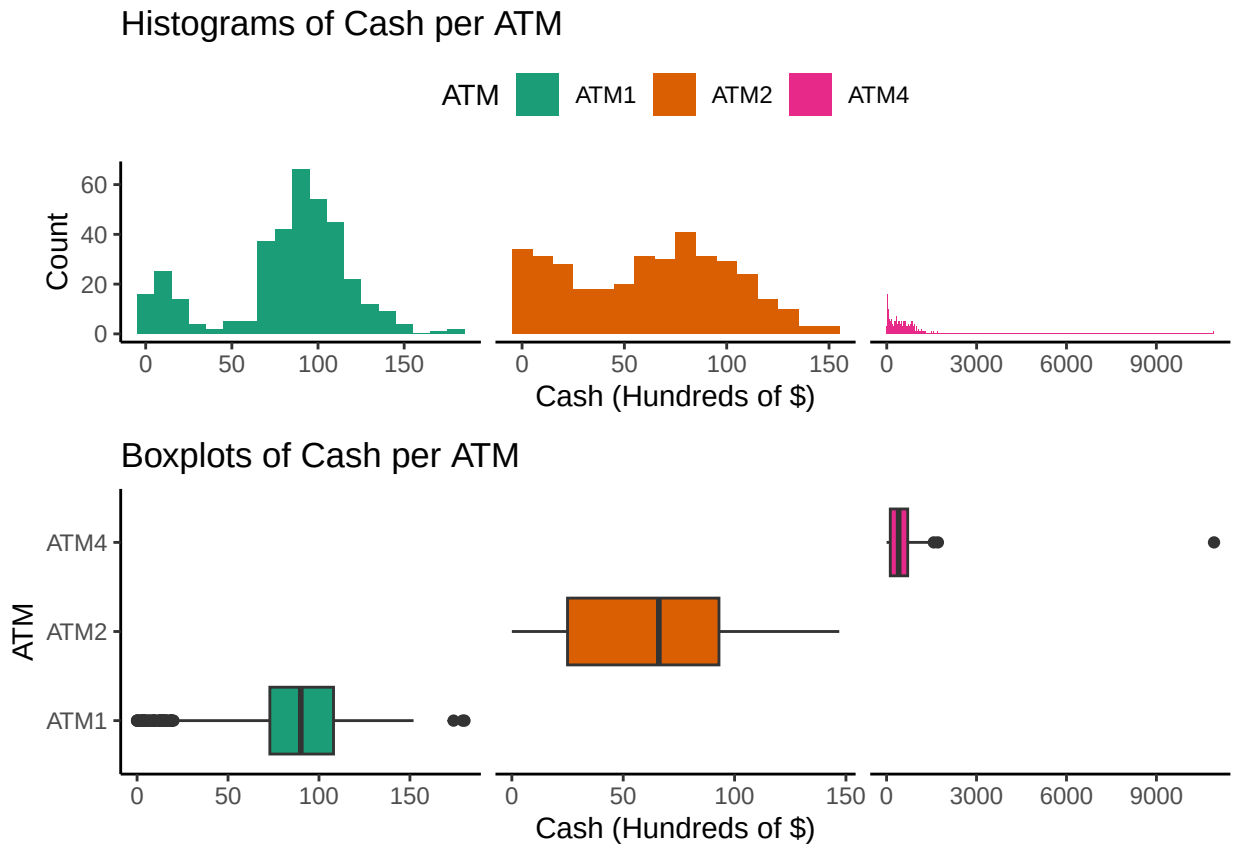
```
atm3 <- atm_ts |>
    filter(ATM == "ATM3")
atm3$Cash[atm3$Cash == 0] <- NA
atm3 <- atm3 |>
    fill(Cash, .direction = "up")
atm_ts <- atm_ts|>
    filter(ATM != "ATM3") |>
    bind_rows(atm3)
```

Next we take a closer look at ATMs 1, 2, and 4.

Since ATMs 1 and 2 have both 0 and `NA` values, the `NA` values will first be converted to 0 values. It is reasonable that these observations correctly indicate that no transactions occurred at these ATMs on those days.

```
atm_ts_na <- atm_ts |>
    filter(is.na(Cash)) |>
    mutate(Cash = 0)
atm_ts <- atm_ts|>
    filter(!is.na(Cash)) |>
    bind_rows(atm_ts_na)
p2a <- atm_ts |>
    filter(ATM != "ATM3") |>
    ggplot(aes(x = Cash, fill = ATM)) +
    geom_histogram(binwidth = 10) +
    facet_wrap(~ ATM, scales = "free_x") +
    scale_fill_manual(values = atm_colors) +
    theme(legend.position = "top",
          strip.background = element_blank(),
          strip.text.x = element_blank()) +
    labs(title = "Histograms of Cash per ATM", x = "Cash (Hundreds of $)",
         y = "Count")
p2b <- atm_ts |>
    filter(ATM != "ATM3") |>
    ggplot(aes(x=Cash, y=ATM, fill=ATM)) +
```

```
    geom_boxplot() +
    facet_grid(~ ATM, scales = "free_x") +
    labs(title="Boxplots of Cash per ATM",x="Cash (Hundreds of $)", y = "ATM") +
    scale_fill_manual(values = atm_colors) +
    theme(strip.background = element_blank(),
          strip.text.x = element_blank(),
          legend.position = "none")
p2 <- plot_grid(p2a, p2b, ncol = 1, align = "v", axis = "l")
p2
```



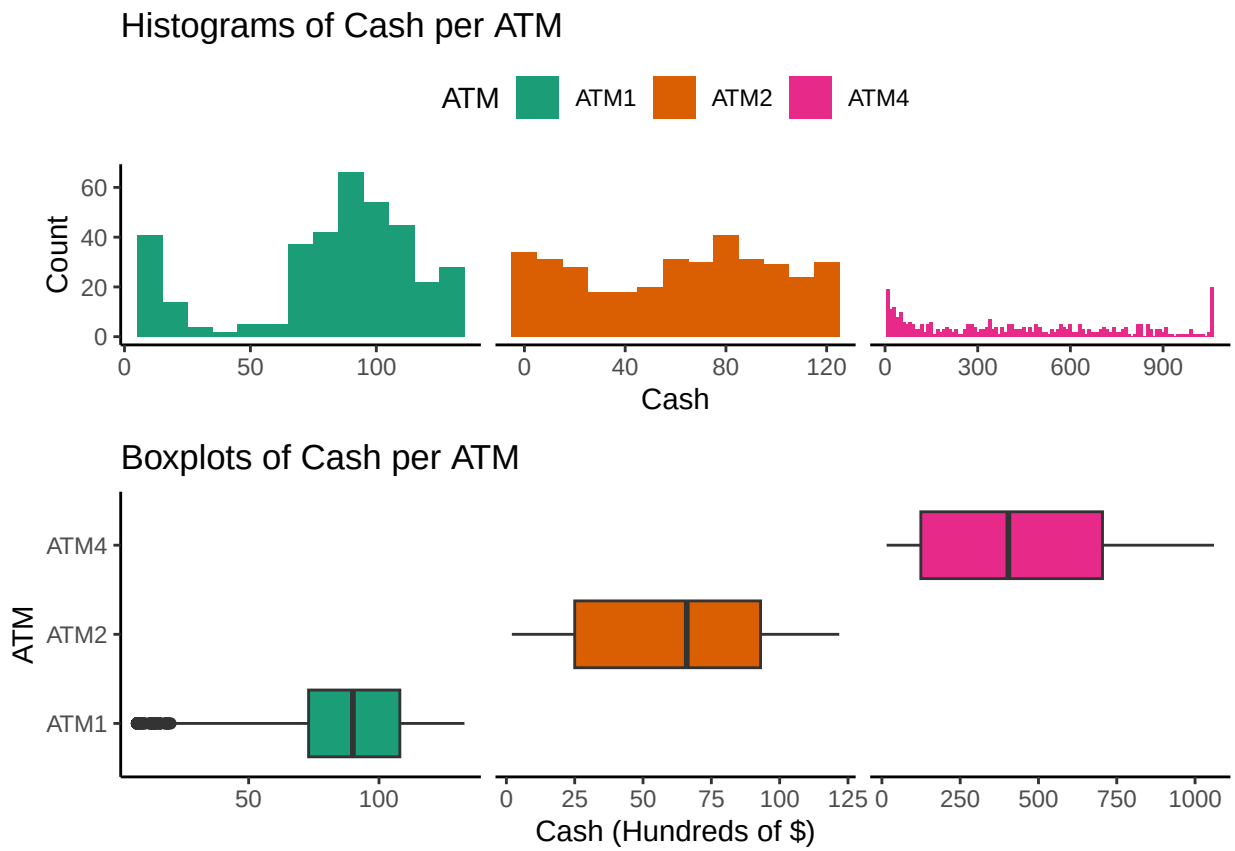
One `Cash` value is over 6 times larger than any other value for ATM4 or any other ATM in this dataset. Our understanding is that ATMs operated in retail locations (i.e. gas stations or hotels) typically hold a max of $\$20,000$, and ATMs operated in banks can hold up to $\$200,000$. We will replace this extreme value and other outliers for ATMs 1, 2, and 4 in the dataset by winsorizing them. By winsorizing, we replace any value of a variable above or below percentile k with the value of the k^{th} percentile itself. A k of 5 is standard, so we will replace outliers below the 5th percentile with the value of the 5th percentile, and we will replace outliers above the 95th percentile with the value of the 95th percentile.

```
atm3 <- atm_ts |>
    filter(ATM == "ATM3")
atm_ts <- atm_ts |>
    filter(ATM != "ATM3") |>
    group_by(ATM) |>
    mutate(Cash = case_when(
        Cash > quantile(Cash, probs = 0.95) ~ quantile(Cash, probs = 0.95),
        Cash < quantile(Cash, probs = 0.05) ~ quantile(Cash, probs = 0.05),
```

```
        TRUE ~ Cash)) |>
    bind_rows(atm3)
```

Now we can confirm whether the outliers in the distributions for ATMS 1, 2, and 4 have been removed.

```
p3a <- atm_ts |>
    filter(ATM != "ATM3") |>
    ggplot(aes(x = Cash, fill = ATM)) +
    geom_histogram(binwidth = 10) +
    facet_wrap(~ ATM, scales = "free_x") +
    scale_fill_manual(values = atm_colors) +
    theme(legend.position = "top",
          strip.background = element_blank(),
          strip.text.x = element_blank()) +
    labs(title = "Histograms of Cash per ATM",
         y = "Count")
p3b <- atm_ts |>
    filter(ATM != "ATM3") |>
    ggplot(aes(x=Cash, y=ATM, fill=ATM)) +
    geom_boxplot() +
    facet_grid(~ ATM, scales = "free_x") +
    labs(title="Boxplots of Cash per ATM",
         x="Cash (Hundreds of $)",
         y = "ATM") +
    scale_fill_manual(values = atm_colors) +
    theme(strip.background = element_blank(),
          strip.text.x = element_blank(),
          legend.position = "none")
p3 <- plot_grid(p3a, p3b, ncol = 1, align = "v", axis = "l")
p3
```

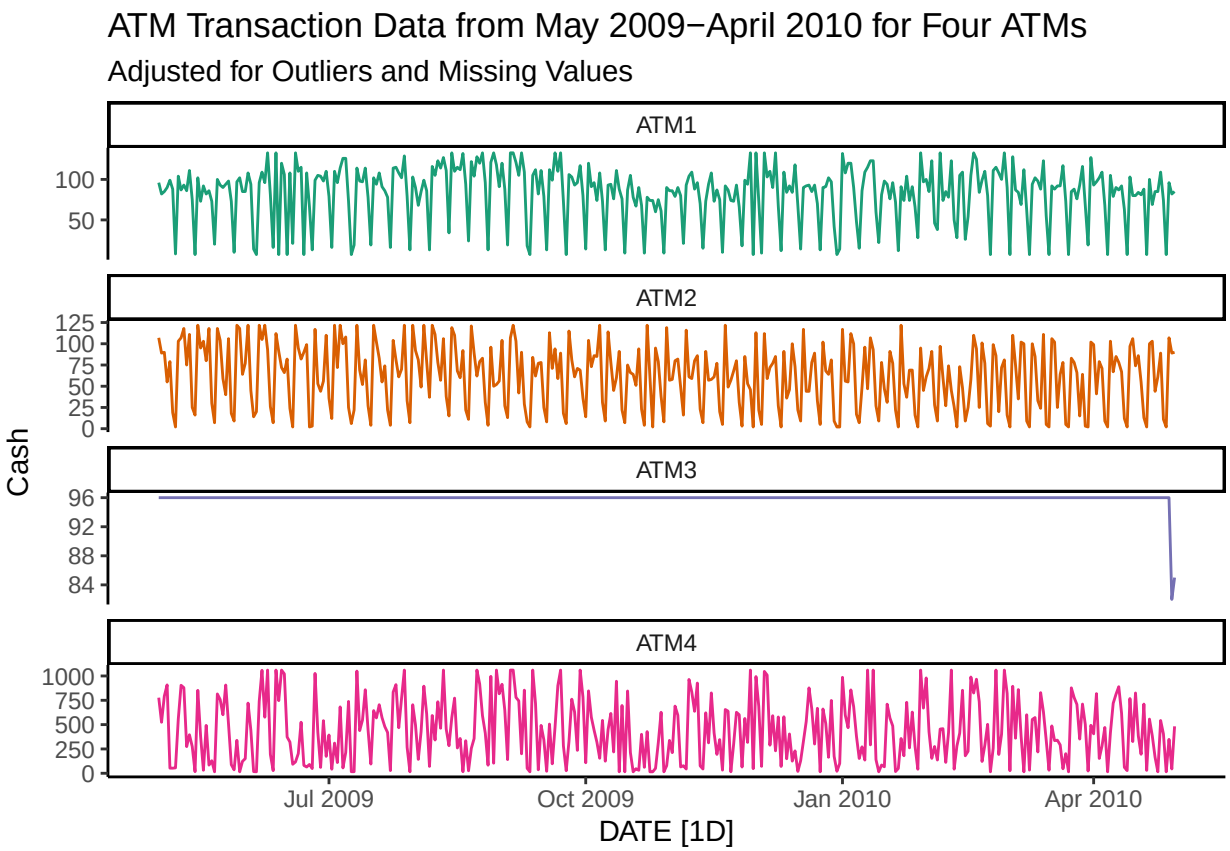


There are no longer any data points considered outliers for ATMs 2 and 4 after replacement. ATM1 still contains data points considered outliers on the low end after replacement. This can happen in bimodal distributions, but we have still reduced the weight of all outliers without eliminating them. We will proceed.

Looking for Trend and Seasonality:

First, we plot the adjusted time series.

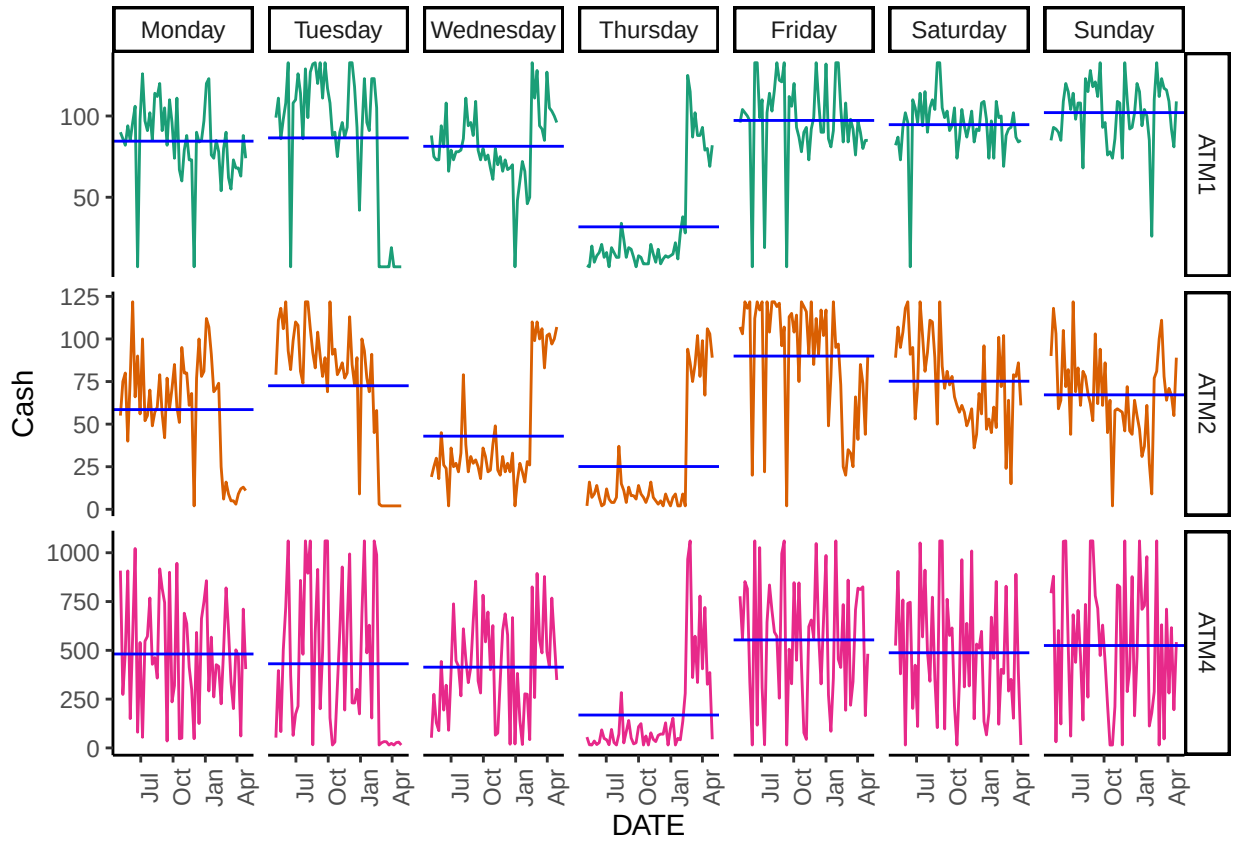
```
p5 <- atm_ts |>
    autoplot(Cash) +
    facet_wrap(~ ATM, scales = "free_y" , ncol = 1) +
    scale_color_brewer(palette = "Dark2") +
    theme(legend.position = "none") +
    labs(title = "ATM Transaction Data from May 2009-April 2010 for Four ATMs",
         subtitle = "Adjusted for Outliers and Missing Values")
p5
```



We can see there are no general trends visible in the time series plots. ATM transactions are neither increasing nor decreasing over time.

Next we look for weekly seasonality, which we expect for ATM transactions. (We don't need to check ATM3 for this.)

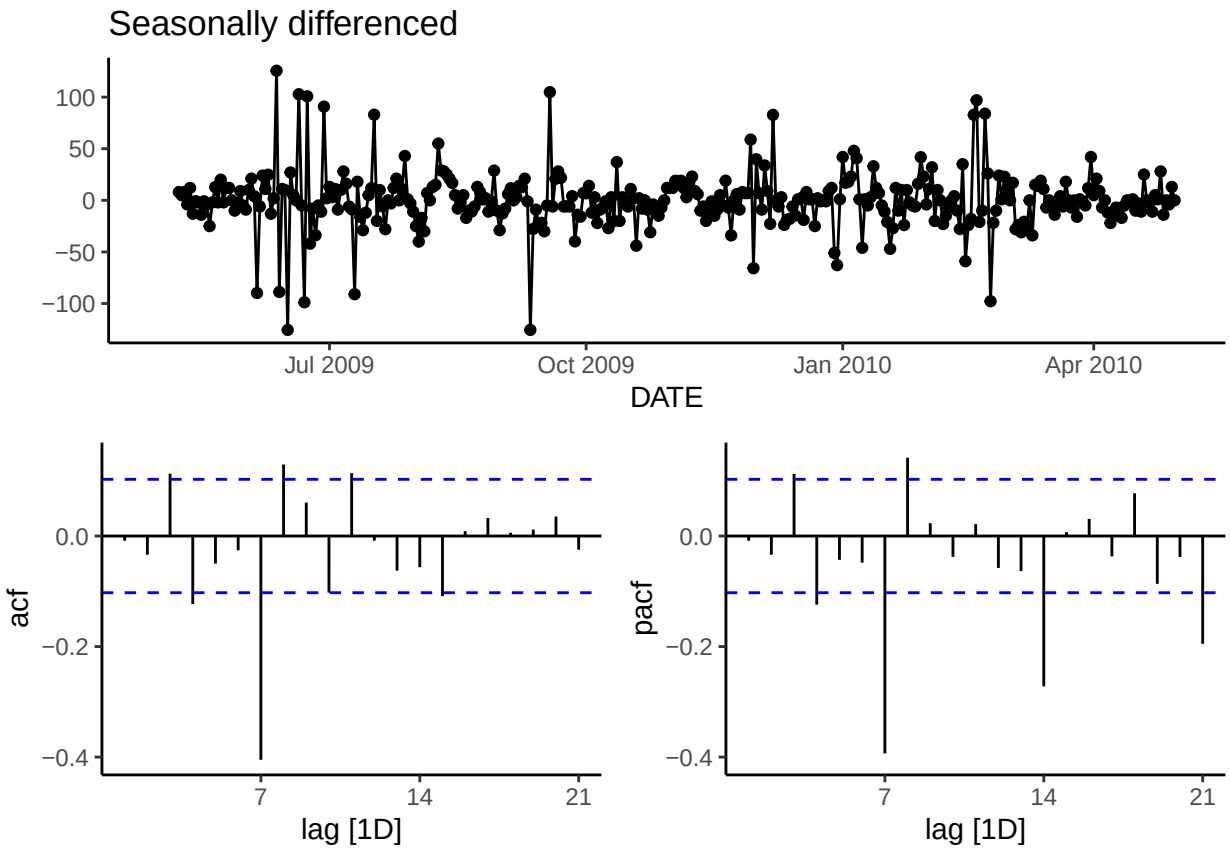
```
p6 <- atm_ts |>
    filter(ATM != "ATM3") |>
    gg_subseries(Cash, period = "week", aes(color = ATM)) +
    scale_color_manual(values = atm_colors) +
    theme(legend.position = "none")
p6
```



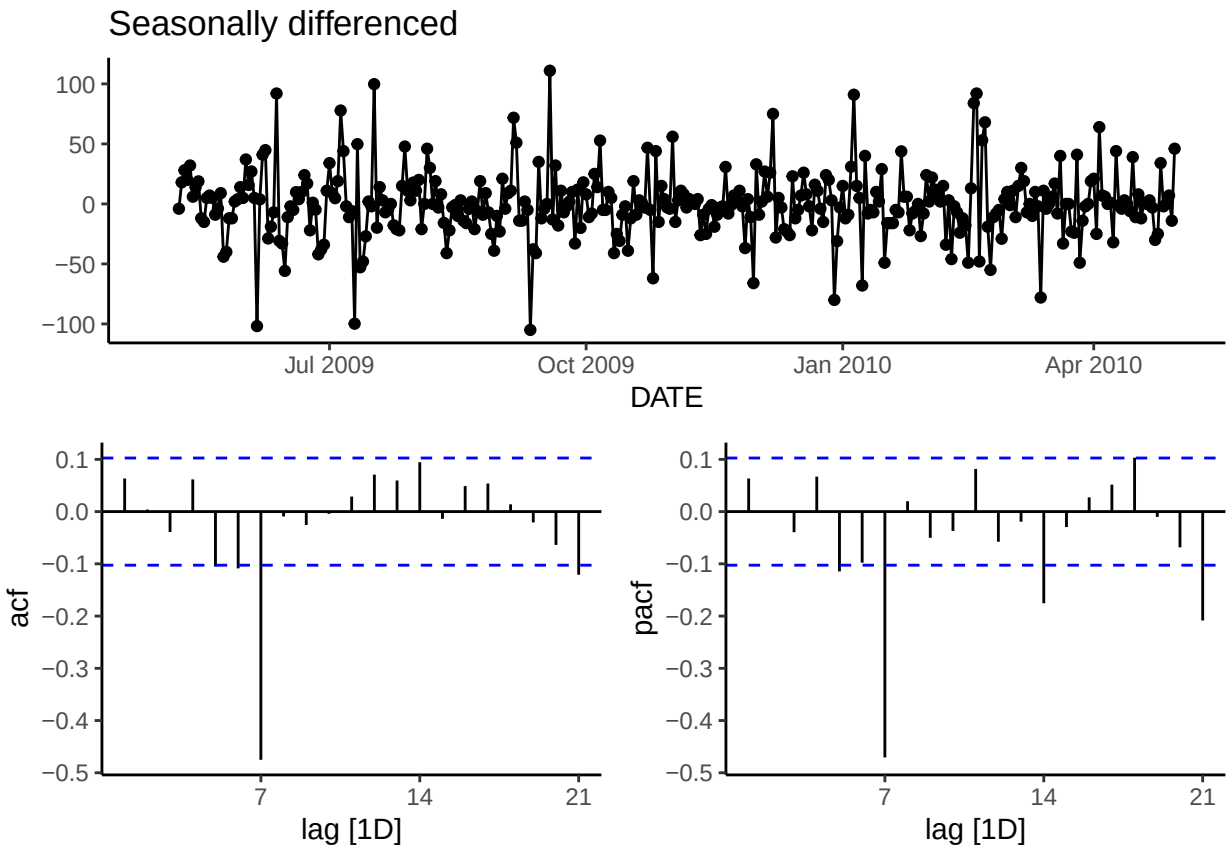
There is weekly seasonality. The middle days of the week generally experience the smallest transaction values, with Thursday being the lowest value transaction day for all ATMs and Wednesday being the second lowest. Friday is the largest value transaction day for ATMs 2 and 4, but Sunday is the largest value transaction day for ATM 1, with Friday being a close second.

Since the data are non-stationary, we will take a seasonal difference.

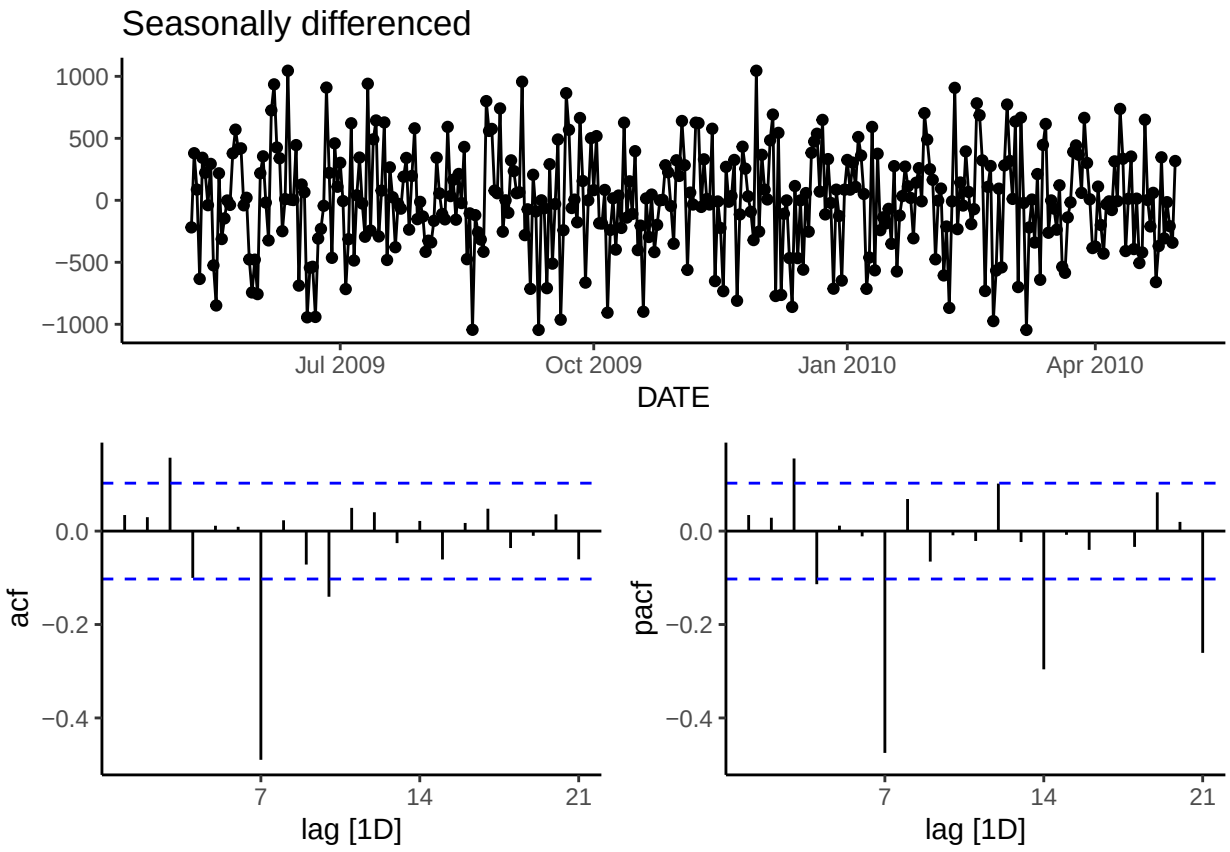
```
p7 <- atm_ts |>
    filter(ATM == "ATM1") |>
    gg_tsdisplay(difference(Cash, lag = 7),
                 plot_type = "partial", lag = 21) +
    labs(title = "Seasonally differenced", y="")
p7
```




```
p8 <- atm_ts |>
    filter(ATM == "ATM2") |>
    gg_tsdisplay(difference(Cash, lag = 7),
                 plot_type = "partial", lag = 21) +
    labs(title = "Seasonally differenced", y="")
p8
```




```
p9 <- atm_ts |>
    filter(ATM == "ATM4") |>
    gg_tsdisplay(difference(Cash, lag = 7),
                 plot_type = "partial", lag = 21) +
    labs(title = "Seasonally differenced", y="")
p9
```



Modeling/Forecasting:

For the seasonally differenced data from ATMS 1, 2, and 4, we see significant spikes at lag 7 in the ACF, and no other particularly significant spikes. We also see exponential decay in the seasonal lags of the PACF, suggesting an $ARIMA(0,0,0)(0,1,1)_7$ model.

We will fit this model as well as an automatically selected $ARIMA$ model.

```
atm1_fit <- atm_ts |>
    filter(ATM == "ATM1" & DATE <= as.Date("2010-03-31")) |>
    model(arima000011 = ARIMA(Cash ~ pdq(0,0,0) + PDQ(0,1,1, period = 7)),
          auto = ARIMA(Cash, stepwise = FALSE, approx = FALSE))
cols <- c("arima000011", "auto")
pivot_atm1_fit <- atm1_fit |>
    pivot_longer(cols = all_of(cols), names_to = "Model Name",
                 values_to = "Orders")
knitr::kable(pivot_atm1_fit, format = "simple")
```

ATM	Model Name	Orders
ATM1	arima000011	<ARIMA(0,0,0)(0,1,1)[7]>
ATM1	auto	<ARIMA(2,0,2)(0,1,2)[7]>

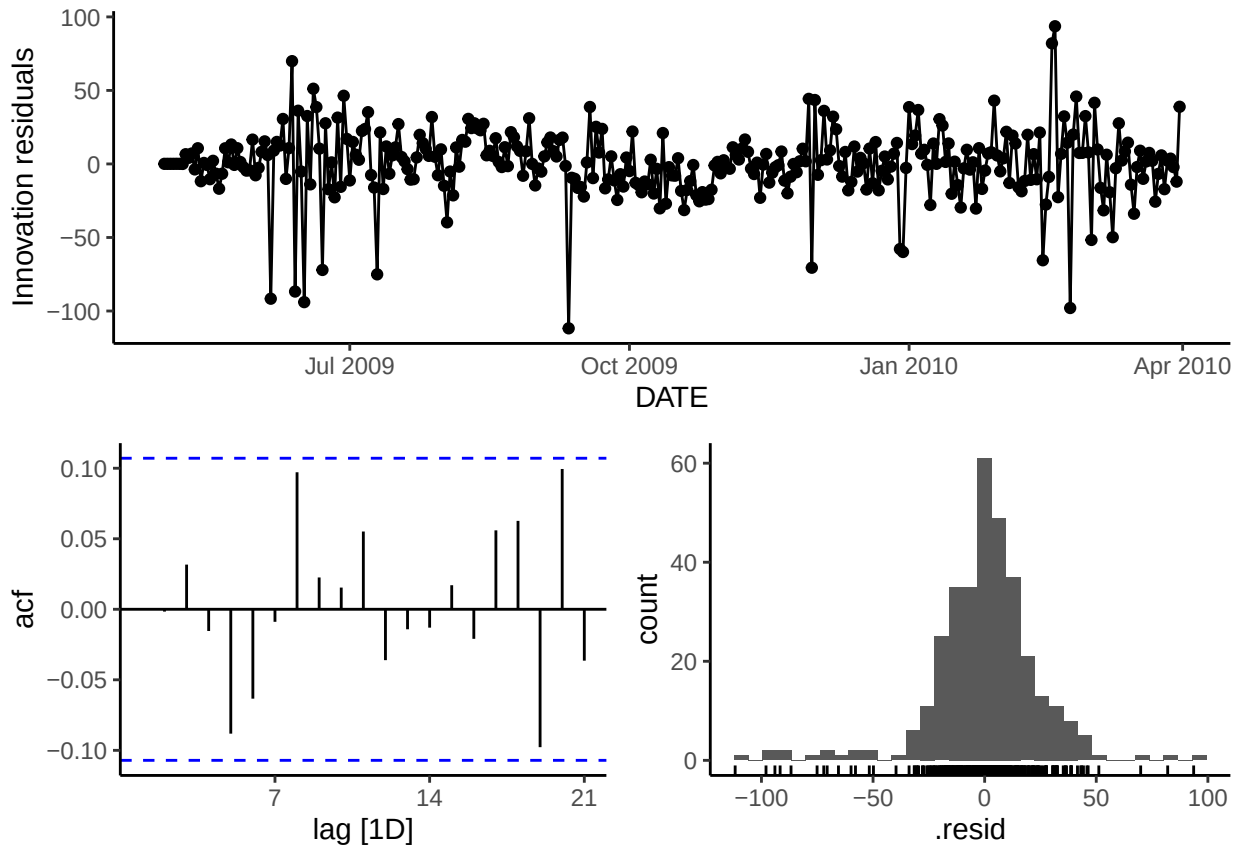
The automatically selected model is an $ARIMA(2,0,2)(0,1,2)_7$ model.

```
glance_atm1_fit <- glance(atm1_fit) |>
    arrange(AICc) |>
    select(.model:BIC)
knitr::kable(glance_atm1_fit)
```

.model	sigma2	log_lik	AIC	AICc	BIC
auto	587.1798	-1510.664	3035.329	3035.679	3061.880
arima000011	612.3163	-1519.385	3042.771	3042.808	3050.357

The automatically selected model has a lower AIC_c than the alternative model we proposed, so we will check the residuals for the automatically selected model.

```
p10 <- atm1_fit |>
    select(auto) |>
    gg_tsresiduals(lag=21)
p10
```



The residuals from the automatically selected model are consistent with white noise. We will confirm using a Ljung-Box test.

```
dof <- atm1_fit |>
    select(auto) |>
    tidy() |>
```

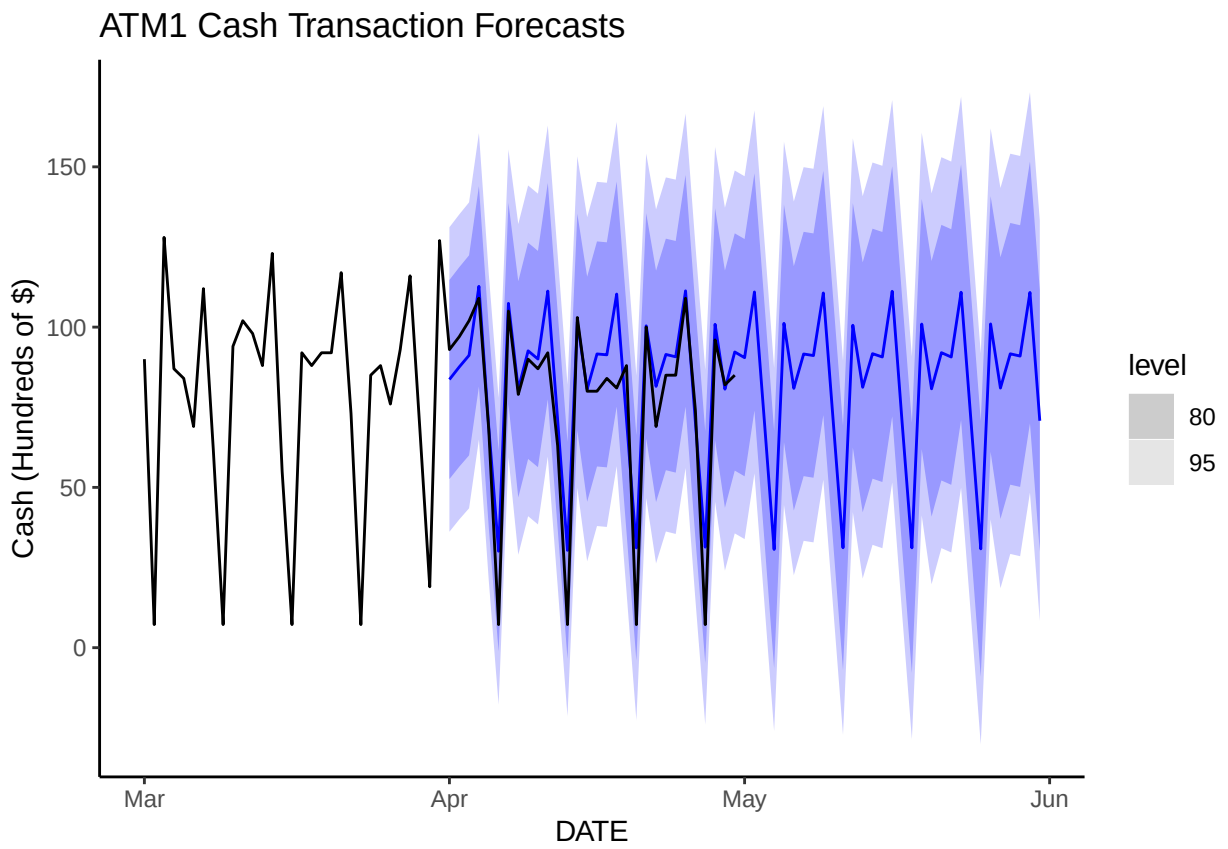
```
    filter(term != "constant") |>
    NROW()
m = 7 #period of seasonality
l = 2*m #for seasonal data, otherwise l = 10
augment(atm1_fit) |>
    filter(.model == "auto") |>
    features(.innov, ljung_box, lag=l, dof=dof)
```

```
## # A tibble: 1 x 4
##   ATM   .model lb_stat lb_pvalue
##   <ord> <chr>    <dbl>     <dbl>
## 1 ATM1  auto      9.65     0.291
```

The large p-value from the test confirms that the residuals are not distinguishable from white noise.

Now we can compare forecasts for April 2010 against the actual values from April 2010, as well as look at forecasts for May 2010. We will not display data prior to March 2010 so that we can zoom in on the accuracy of these predictions.

```
atm_ts_since_march_2010 <- atm_ts |>
    filter(DATE >= as.Date("2010-03-01"))
p11 <- atm1_fit |>
    forecast(h=61) |>
    filter(.model == "auto") |>
    autoplot(atm_ts_since_march_2010 |> filter(ATM == "ATM1")) +
    labs(title = "ATM1 Cash Transaction Forecasts",
         y="Cash (Hundreds of $)")
p11
```



The forecasts from the automatically selected model capture the typical midweek dips and end of week peaks very well. So we can anticipate our May 2010 forecasts being reliable estimates of ATM transactions for ATM1. We save the forecasts to an Excel file.

```
parta_excel_forecasts_atm1 <- atm1_fit |>
    forecast(h=61) |>
    filter(.model == "auto" & month(DATE) == 5) |>
    as_tibble() |>
    mutate(DATE = as.character(DATE),
           Cash = as.character(Cash))
write_xlsx(parta_excel_forecasts_atm1, "parta_excel_forecasts_atm1.xlsx")
```

We repeat the process for ATMs 2 and 4.

```
atm2_fit <- atm_ts |>
    filter(ATM == "ATM2" & DATE <= as.Date("2010-03-31")) |>
    model(arima000011 = ARIMA(Cash ~ pdq(0,0,0) + PDQ(0,1,1, period = 7)),
          auto = ARIMA(Cash, stepwise = FALSE, approx = FALSE))
pivot_atm2_fit <- atm2_fit |>
    pivot_longer(cols = all_of(cols), names_to = "Model Name",
                 values_to = "Orders")
knitr::kable(pivot_atm2_fit, format = "simple")
```

ATM	Model Name	Orders
ATM2	arima000011	<ARIMA(0,0,0)(0,1,1)[7]>
ATM2	auto	<ARIMA(2,0,2)(0,1,1)[7] w/ drift>

The automatically selected model for ATM2 is an $ARIMA(2,0,2)(0,1,1)_7$ model with drift.

```
drift <- atm2_fit |>
    select(auto) |>
    tidy() |>
    filter(term == "constant") |>
    select(estimate) |>
    as.numeric()
drift <- round(drift, 2)
```

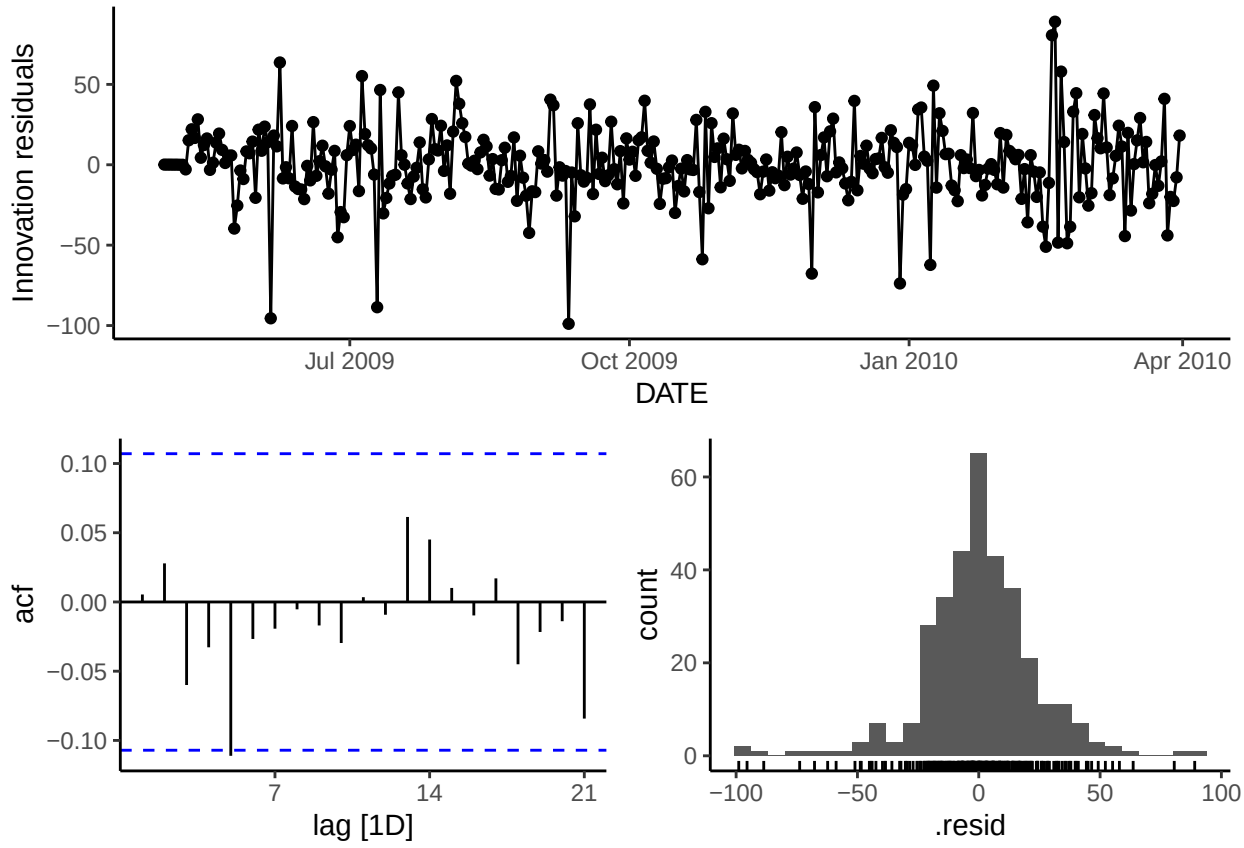
The drift term is -1.09, indicating a decreasing trend over time that we did not pick up on initially. Looking back at the time series plots, it is clear that ATM2 has smaller peaks at the end of its time series plot than at the beginning.

```
glance_atm2_fit <- glance(atm2_fit) |>
    arrange(AICc) |>
    select(.model:BIC)
knitr::kable(glance_atm2_fit)
```

.model	sigma2	log_lik	AIC	AICc	BIC
auto	555.1921	-1501.378	3016.756	3017.106	3043.307
arima000011	599.9369	-1515.574	3035.148	3035.185	3042.734

The automatically selected model has a lower AIC_c than the alternative model we proposed, so we will check the residuals for the automatically selected model.

```
p12 <- atm2_fit |>
    select(auto) |>
    gg_tsresiduals(lag=21)
p12
```



One small spike at lag 5 is still consistent with white noise. We will confirm using a Ljung-Box test.

```
dof <- atm2_fit |>
    select(auto) |>
    tidy() |>
    filter(term != "constant") |>
    NROW()
m = 7 #period of seasonality
l = 2*m #for seasonal data, otherwise l = 10
augment(atm2_fit) |>
    filter(.model == "auto") |>
    features(.innov, ljung_box, lag=l, dof=dof)
```

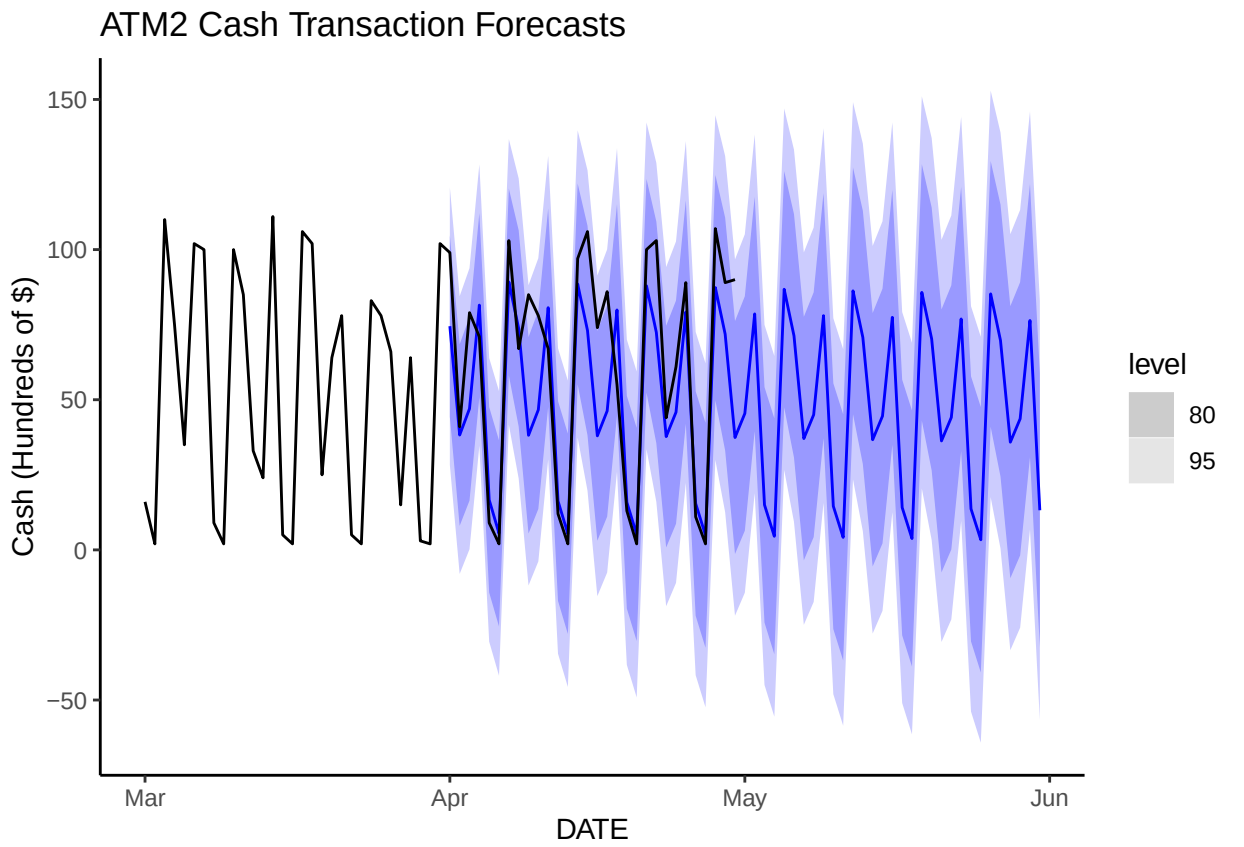
```
## # A tibble: 1 x 4
##   ATM   .model lb_stat lb_pvalue
##   <ord> <chr>    <dbl>     <dbl>
## 1 ATM2  auto      8.94     0.442
```

The large p-value from the test confirms that the residuals are not distinguishable from white noise.

Now we can forecast.

```
p13 <- atm2_fit |>
    forecast(h=61) |>
    filter(.model == "auto") |>
    autoplot(atm_ts_since_march_2010 |> filter(ATM == "ATM2")) +
```

```
    labs(title = "ATM2 Cash Transaction Forecasts",
         y="Cash (Hundreds of $)")
p13
```



The forecasts from the automatically selected model look pretty reasonable for ATM2. We save them to an Excel file.

```
parta_excel_forecasts_atm2 <- atm2_fit |>
    forecast(h=61) |>
    filter(.model == "auto" & month(DATE) == 5) |>
    as_tibble() |>
    mutate(DATE = as.character(DATE),
           Cash = as.character(Cash))
write_xlsx(parta_excel_forecasts_atm2, "parta_excel_forecasts_atm2.xlsx")
```

We proceed with ATM4, noting that in order for the $ARIMA(0,0,0)(0,1,1)_7$ model we've been proposing for all ATMs to be stable for ATM4, a drift constant has to be added.

```
atm4_fit <- atm_ts |>
    filter(ATM == "ATM4" & DATE <= as.Date("2010-03-31")) |>
    model(arima000011drift = ARIMA(Cash ~ 1 + pdq(0,0,0) + PDQ(0,1,1,
                                                               period = 7)),
          auto = ARIMA(Cash, stepwise = FALSE, approx = FALSE))
cols <- c("arima000011drift", "auto")
pivot_atm4_fit <- atm4_fit |>
```

```
    pivot_longer(cols = all_of(cols), names_to = "Model Name",
                 values_to = "Orders")
knitr::kable(pivot_atm4_fit, format = "simple")
```

ATM	Model Name	Orders
ATM4	arima000011drift	<ARIMA(0,0,0)(0,1,1)[7] w/ drift>
ATM4	auto	<ARIMA(0,0,0)(2,0,0)[7] w/ mean>

The automatically selected model for ATM4 is an $ARIMA(0,0,0)(2,0,0)_7$ model with mean (i.e. intercept).

```
intercept <- atm4_fit |>
    select(auto) |>
    tidy() |>
    filter(term == "constant") |>
    select(estimate) |>
    as.numeric()
intercept <- round(intercept, 2)
```

The intercept term for the automatically selected model is 320.51.

```
glance_atm4_fit <- glance(atm4_fit) |>
    arrange(AICc) |>
    select(.model:BIC)
knitr::kable(glance_atm4_fit)
```

.model	sigma2	log_lik	AIC	AICc	BIC
arima000011drift	99928.85	-2359.774	4725.547	4725.622	4736.927
auto	106292.62	-2412.687	4833.375	4833.496	4848.631

It's not possible to compare the AIC_c values of these two models, as they have different orders of differencing d. ATM4 is the first ATM for which the automatically selected model did not include one level of seasonal differencing. We know from exploring the data previously that this level of differencing is needed, so we will reject the automatically selected model on that basis.

Next, we will see if the automatically selected models for the previous two ATMs perform better for ATM4 than the $ARIMA(0,0,0)(0,1,1)_7$ model we proposed for all of them. Then we can choose the model among those with the lowest AIC_c value. All models require a drift term.

```
atm4_fit <- atm_ts |>
    filter(ATM == "ATM4" & DATE <= as.Date("2010-03-31")) |>
    model(arima000011drift = ARIMA(Cash ~ 1 + pdq(0,0,0) + PDQ(0,1,1,
                                                               period = 7)),
          arima202012drift = ARIMA(Cash ~ 1 + pdq(2,0,2) + PDQ(0,1,2, period = 7)),
          arima202011drift = ARIMA(Cash ~ 1 + pdq(2,0,2) + PDQ(0,1,1,
                                                               period = 7)))
cols <- c("arima000011drift", "arima202012drift", "arima202011drift")
pivot_atm4_fit <- atm4_fit |>
    pivot_longer(cols = all_of(cols), names_to = "Model Name",
                 values_to = "Orders")
knitr::kable(pivot_atm4_fit, format = "simple")
```

ATM	Model Name	Orders
ATM4	arima000011drift	<ARIMA(0,0,0)(0,1,1)[7] w/ drift>
ATM4	arima202012drift	<ARIMA(2,0,2)(0,1,2)[7] w/ drift>
ATM4	arima202011drift	<ARIMA(2,0,2)(0,1,1)[7] w/ drift>

```
glance_atm4_fit <- glance(atm4_fit) |>
    arrange(AICc) |>
    select(.model:BIC)
knitr::kable(glance_atm4_fit)
```

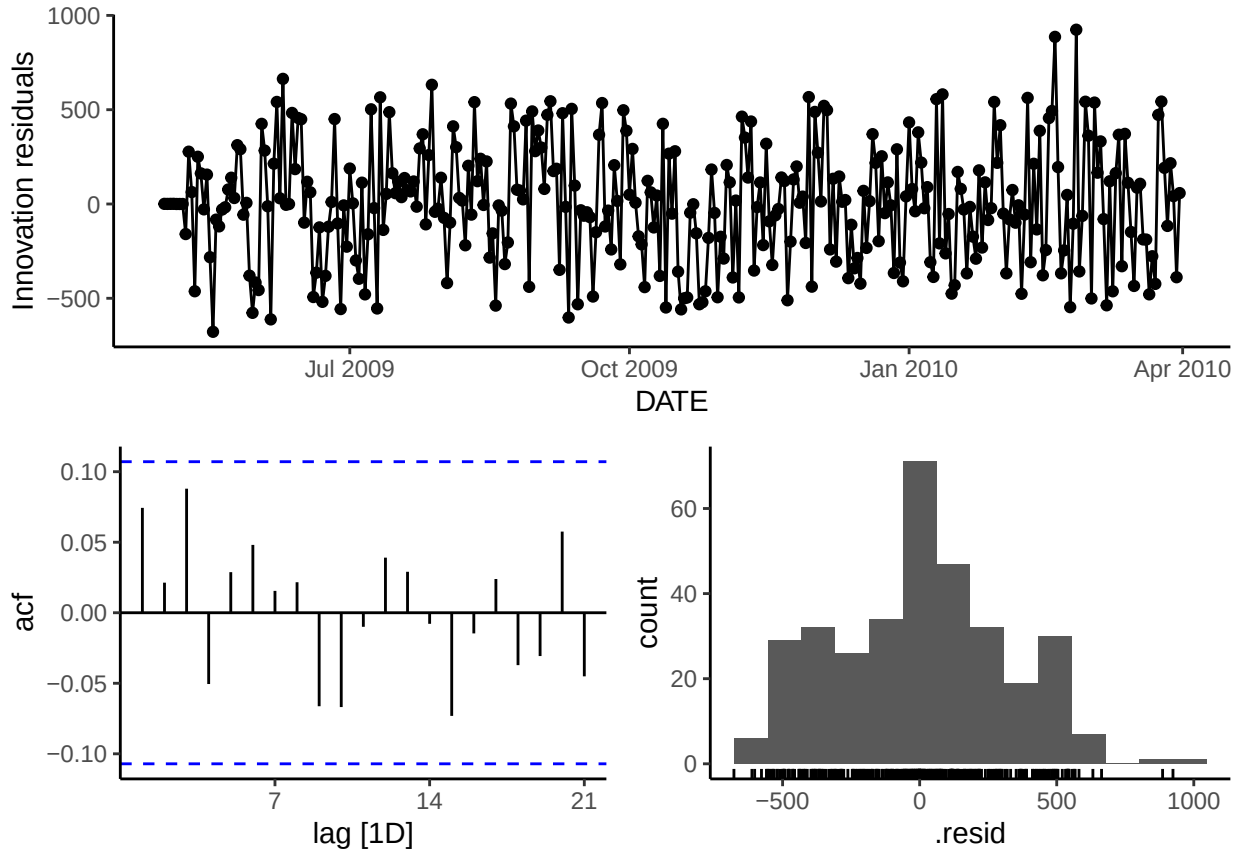
.model	sigma2	log_lik	AIC	AICc	BIC
arima000011drift	99928.85	-2359.774	4725.547	4725.622	4736.927
arima202012drift	98743.95	-2357.075	4730.150	4730.601	4760.494
arima202011drift	101175.22	-2359.768	4733.536	4733.886	4760.087

The $ARIMA(0,0,0)(0,1,1)_7$ model we proposed has the lowest AIC_c value.

```
drift <- atm4_fit |>
    select(arima000011drift) |>
    tidy() |>
    filter(term == "constant") |>
    select(estimate) |>
    as.numeric()
drift <- round(drift, 2)
```

Its drift term is -0.12.

```
p14 <- atm4_fit |>
    select(arima000011drift) |>
    gg_tsresiduals(lag=21)
p14
```



The residuals look like white noise. We will confirm with a Ljung-Box test.

```
dof <- atm4_fit |>
    select(arima000011drift) |>
    tidy() |>
    filter(term != "constant") |>
    NROW()
m = 7 #period of seasonality
l = 2*m #for seasonal data, otherwise l = 10
augment(atm4_fit) |>
    filter(.model == "arima000011drift") |>
    features(.innov, ljung_box, lag=l, dof=dof)
```

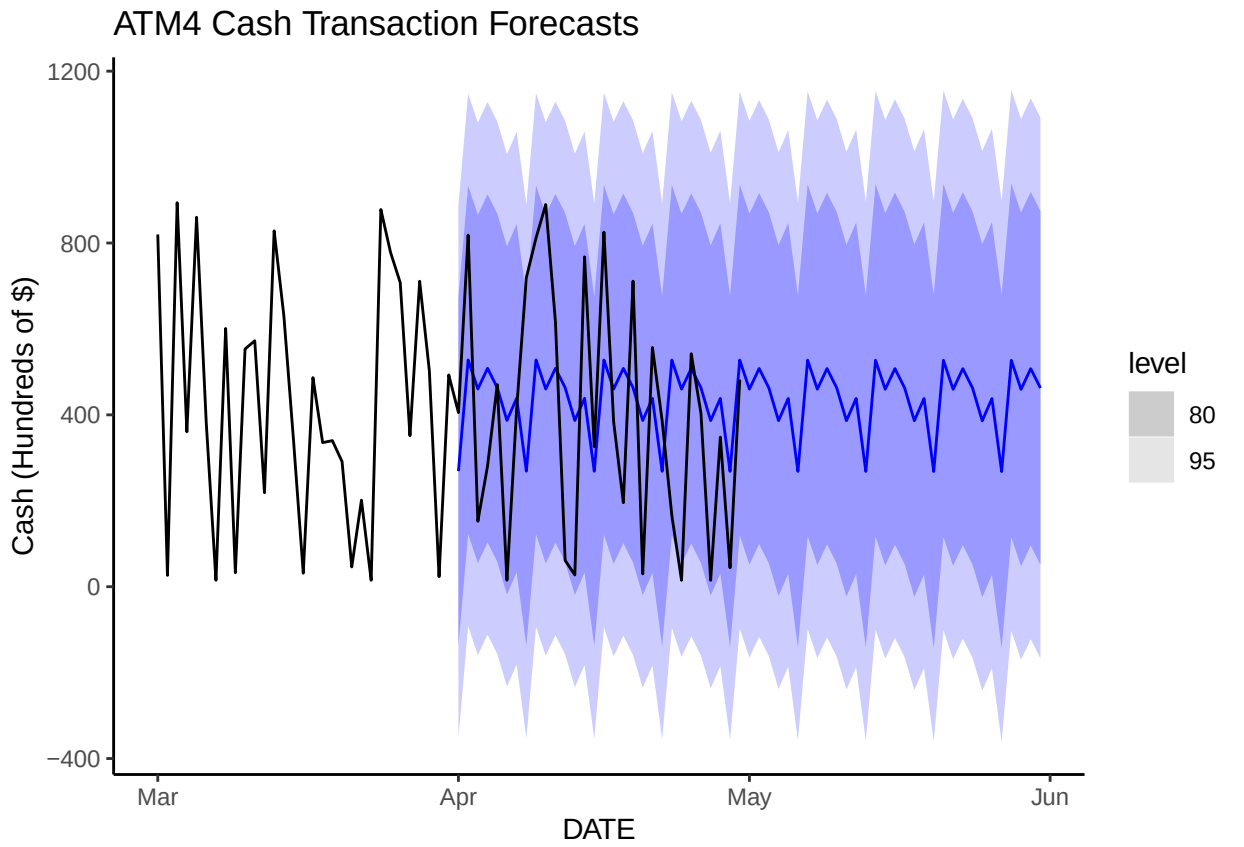
```
## # A tibble: 1 x 4
##   ATM   .model           lb_stat lb_pvalue
##   <ord> <chr>              <dbl>     <dbl>
## 1 ATM4  arima000011drift    10.8     0.627
```

The large p-value from the test confirms that the residuals are not distinguishable from white noise.

Now we can forecast.

```
p15 <- atm4_fit |>
    forecast(h=61) |>
    filter(.model == "arima000011drift") |>
    autoplot(atm_ts_since_march_2010 |> filter(ATM == "ATM4")) +
```

```
    labs(title = "ATM4 Cash Transaction Forecasts",
         y="Cash (Hundreds of $)")
p15
```



These April 2010 forecasts vs. actual values aren't as close as those for ATMs 1 or 2. So it's possible the May 2010 forecasts for ATM4 are the least reliable of the three. It's worth noting that the 80% prediction intervals do a good job of capturing what the point forecasts do not though.

We consider whether transforming the data for ATM4 would improve our forecasting ability.

```
lambda <- atm_ts |>
    filter(ATM == "ATM4") |>
    features(Cash, features = guerrero) |>
    pull(lambda_guerrero)
lambda <- round(lambda, 2)
```

With a proposed lambda of 0.58, a square root transformation of the Cash data for ATM4 is reasonable. We refit the model we originally proposed on the transformed data, and we also fit a new automatically selected model.

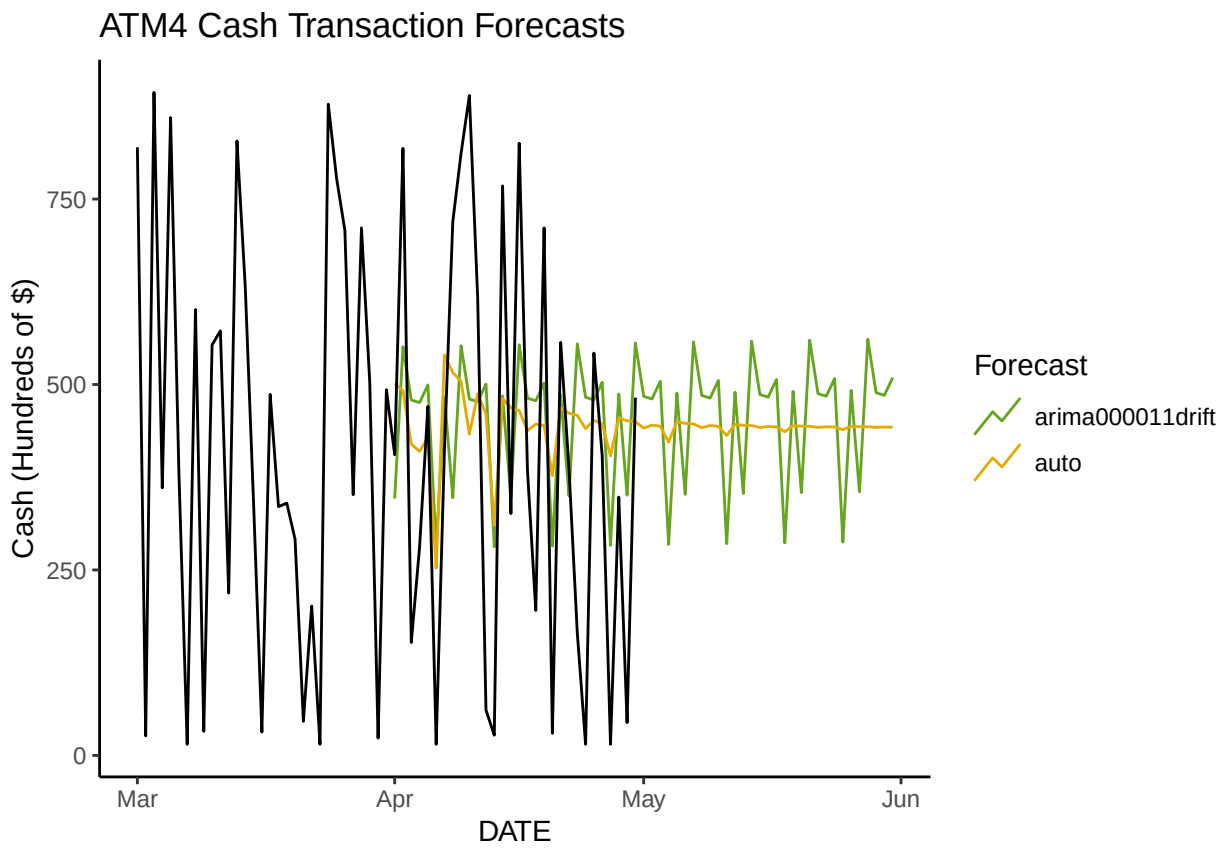
```
atm4_fit <- atm_ts |>
    filter(ATM == "ATM4" & DATE <= as.Date("2010-03-31")) |>
    model(arima000011drift = ARIMA(sqrt(Cash) ~ 1 + pdq(0,0,0) + PDQ(0,1,1,
                                                      period = 7)),
          auto = ARIMA(sqrt(Cash), stepwise = FALSE, approx = FALSE))
cols <- c("arima000011drift", "auto")
```

```
pivot_atm4_fit <- atm4_fit |>
    pivot_longer(cols = all_of(cols), names_to = "Model Name",
                 values_to = "Orders")
knitr::kable(pivot_atm4_fit, format = "simple")
```

ATM	Model Name	Orders
ATM4	arima000011drift	<ARIMA(0,0,0)(0,1,1)[7] w/ drift>
ATM4	auto	<ARIMA(0,0,0)(2,0,0)[7] w/ mean>

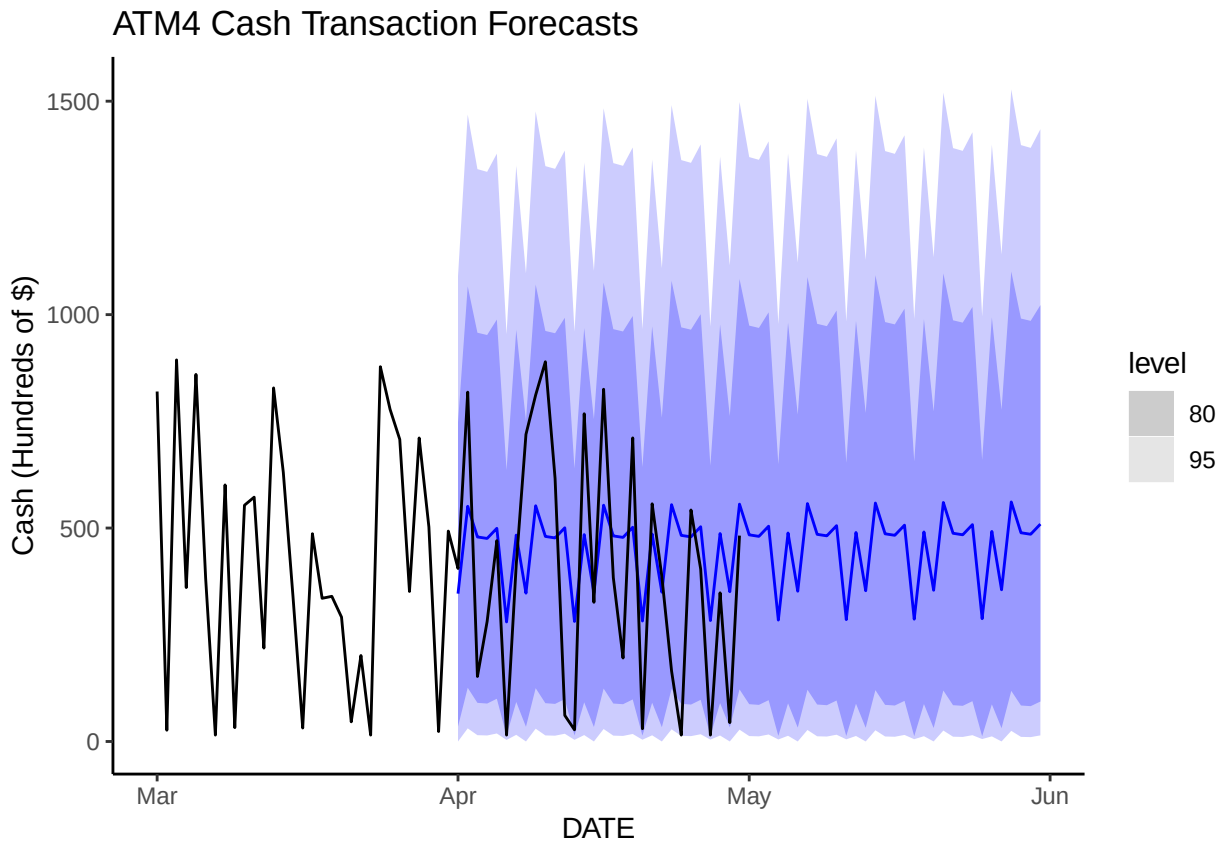
We can't compare the AIC_c values for the automatically selected model and the original model, as they have different orders of differencing, so we will skip to forecast comparisons.

```
palette <- brewer.pal(n = 8, name = "Dark2")
model_colors <- palette[5:6]
names(model_colors) <- c("arima000011drift", "auto")
p16 <- atm4_fit |>
    forecast(h=61) |>
    autoplot(atm_ts_since_march_2010 |> filter(ATM == "ATM4"), level = NULL) +
    labs(title = "ATM4 Cash Transaction Forecasts",
         y="Cash (Hundreds of $)") +
    guides(color = guide_legend(title = "Forecast")) +
    scale_color_manual(values = model_colors)
p16
```



The automatically selected model does not perform better than the originally proposed model on the transformed data. So we choose the originally proposed model as the superior forecasting model here, replot the forecasts for only this model with prediction intervals, and save the forecasts to an Excel file.

```
p17 <- atm4_fit |>
    forecast(h=61) |>
    filter(.model == "arima000011drift") |>
    autoplot(atm_ts_since_march_2010 |> filter(ATM == "ATM4")) +
    labs(title = "ATM4 Cash Transaction Forecasts",
         y="Cash (Hundreds of $)")
p17
```

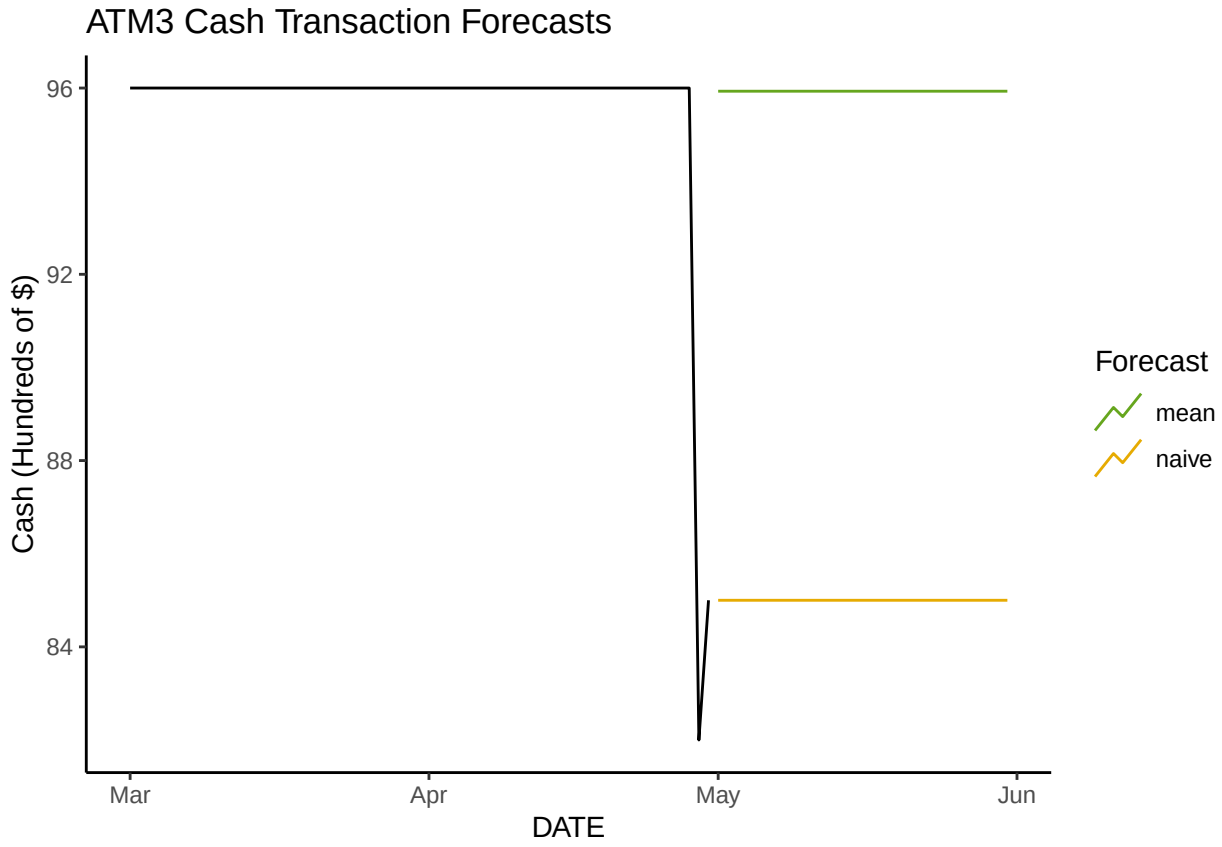



```
parta_excel_forecasts_atm4 <- atm4_fit |>
    forecast(h=61) |>
    filter(.model == "arima000011drift" & month(DATE) == 5) |>
    as_tibble() |>
    mutate(DATE = as.character(DATE),
           Cash = as.character(Cash))
write_xlsx(parta_excel_forecasts_atm4, "parta_excel_forecasts_atm4.xlsx")
```

We return to ATM3, for which we will simply compare forecasts from the MEAN and NAIVE methods.

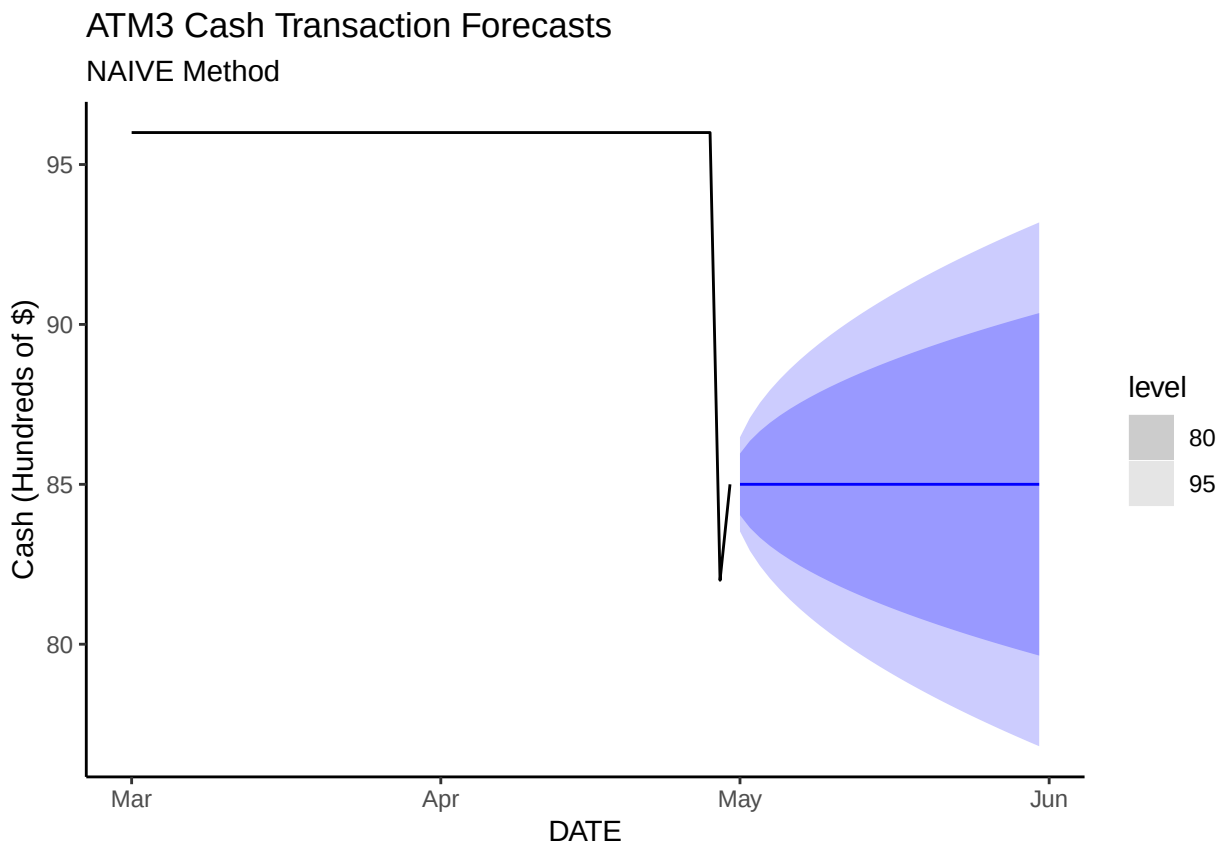
```
atm3_fit <- atm_ts |>
    filter(ATM == "ATM3") |>
    model(mean = MEAN(Cash),
```

```
          naive = NAIVE(Cash))
names(model_colors) <- c("mean", "naive")
p18 <- atm3_fit |>
    forecast(h=31) |>
    autoplot(atm_ts_since_march_2010 |> filter(ATM == "ATM3"), level = NULL) +
    labs(title = "ATM3 Cash Transaction Forecasts",
         y="Cash (Hundreds of $)") +
    guides(color = guide_legend(title = "Forecast")) +
    scale_color_manual(values = model_colors)
p18
```



We favor the NAIVE method here because we imputed most of the data for ATM3, and its mean is therefore quite fabricated. Using the most recent observation as the predicted value for all future observations is reasonable until more data can be collected for this ATM. We replot only the forecasts from the NAIVE method with prediction intervals and save the forecasts to an Excel file.

```
p19 <- atm3_fit |>
    forecast(h=31) |>
    filter(.model == "naive") |>
    autoplot(atm_ts_since_march_2010 |> filter(ATM == "ATM3")) +
    labs(title = "ATM3 Cash Transaction Forecasts",
         subtitle = "NAIVE Method",
         y="Cash (Hundreds of $)")
p19
```




```
parta_excel_forecasts_atm3 <- atm3_fit |>
    forecast(h=31) |>
    filter(.model == "naive") |>
    as_tibble() |>
    mutate(DATE = as.character(DATE),
           Cash = as.character(Cash))
write_xlsx(parta_excel_forecasts_atm3, "parta_excel_forecasts_atm3.xlsx")
```

Part B:

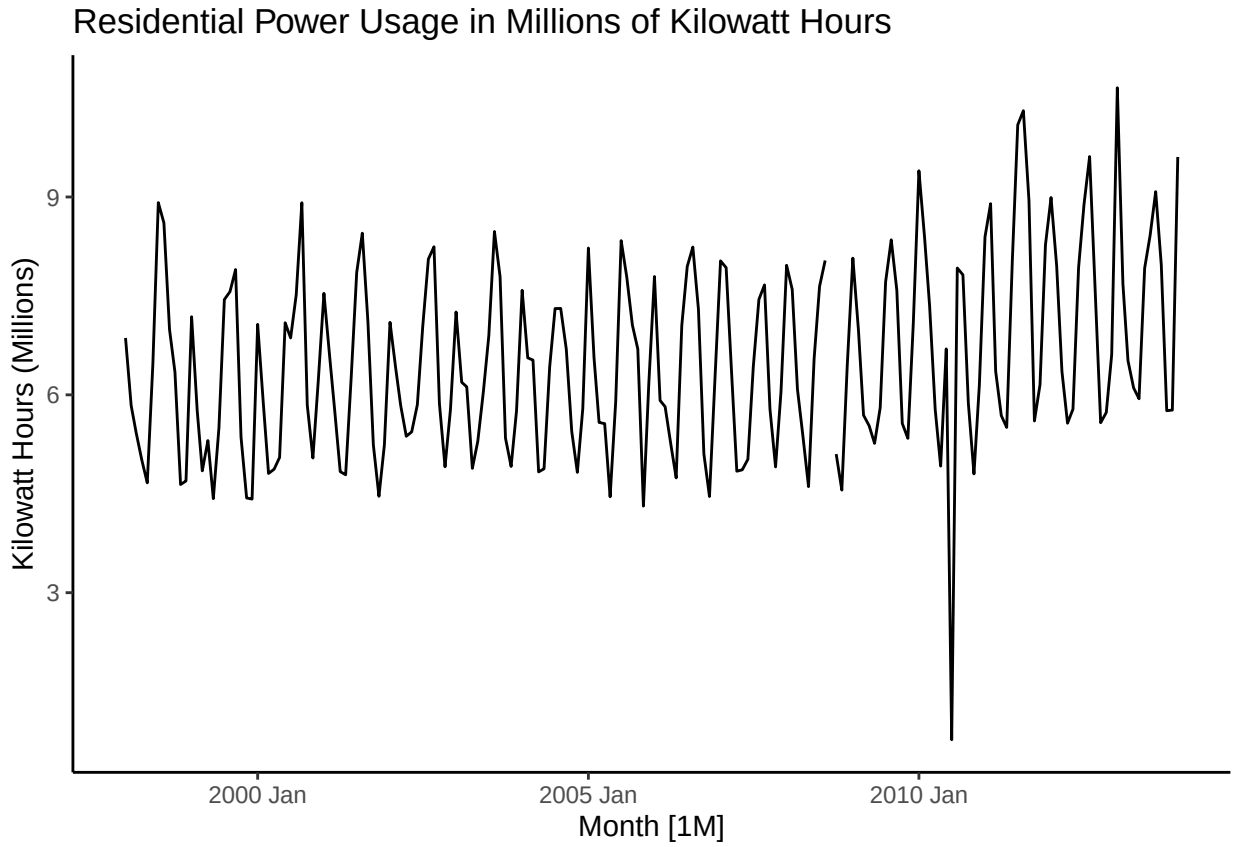
Data Preparation:

We load data on residential power usage from January 1998 until December 2013.

```
my_url2 <- "https://github.com/geedoubledee/data624_project1/raw/main/ResidentialCustomerForecastLoad-6
col_types <- c("numeric", "text", "numeric")
col_names <- c("CaseSeq", "Month", "KWH")
temp <- tempfile(fileext = ".xlsx")
req <- GET(my_url2, authenticate(Sys.getenv("GITHUB_PAT"), ""),
           write_disk(path = temp))
res_power <- readxl::read_excel(temp, col_types = col_types)
colnames(res_power) <- col_names
```

We remove the unnecessary `CaseSequence` variable, coerce the `Month` variable to class *yearmonth*, and rescale Kilowatt hours as millions of Kilowatt hours. We create and plot the time series.

```
res_power <- res_power |>
    select(-CaseSeq) |>
    mutate(Month = yearmonth(Month),
           KWH_Mil = KWH / 1000000)
res_power_ts <- res_power |>
    as_tsibble(index = Month)
res_power_ts |>
    autoplot(KWH_Mil) +
    labs(title = "Residential Power Usage in Millions of Kilowatt Hours",
         y = "Kilowatt Hours (Millions)")
```



Missing Data and Outliers:

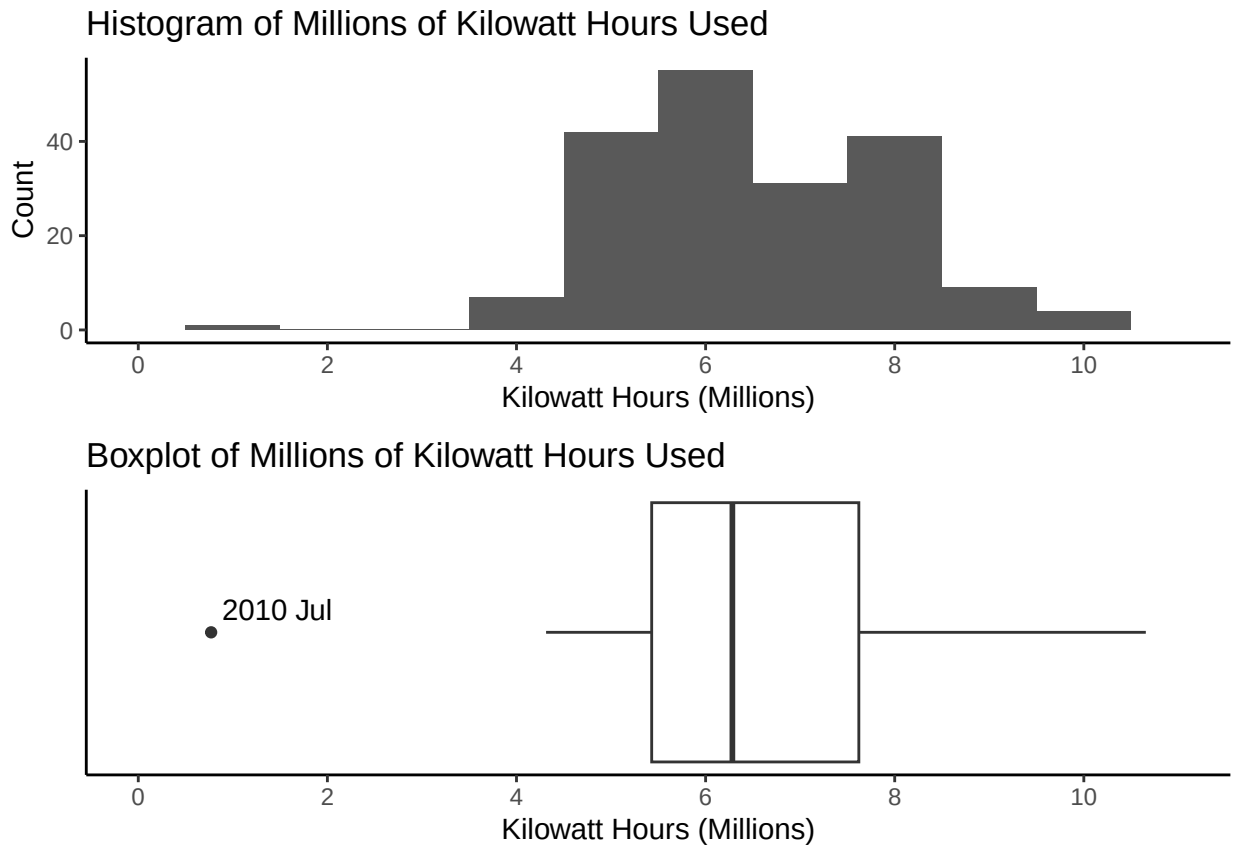
We can already see an outlier, and we print a summary of 0 or `NA` values.

```
na_summary <- res_power_ts |>
    select(-KWH) |>
    as_tibble() |>
    filter(is.na(KWH_Mil) | KWH_Mil == 0)
knitr::kable(na_summary)
```

Month	KWH_Mil
2008 Sep	NA

There is one missing value for September 2008 that we will have to deal with. We check further for outliers.

```
p20a <- res_power_ts |>
    filter(!is.na(KWH_Mil)) |>
    ggplot(aes(x = KWH_Mil)) +
    geom_histogram(binwidth = 1) +
    scale_x_continuous(limits = c(0, 11), breaks = seq(0, 12, 2)) +
    labs(title = "Histogram of Millions of Kilowatt Hours Used",
         x = "Kilowatt Hours (Millions)",
         y = "Count")
p20b <- res_power_ts |>
    filter(!is.na(KWH_Mil)) |>
    ggplot(aes(x = KWH_Mil)) +
    geom_boxplot() +
    scale_x_continuous(limits = c(0, 11), breaks = seq(0, 12, 2)) +
    theme(axis.text.y = element_blank(),
          axis.ticks.y = element_blank()) +
    labs(title = "Boxplot of Millions of Kilowatt Hours Used",
         x = "Kilowatt Hours (Millions)")
#Restructure required if more than 1 outlier or if tie values in KWH_Mil
out <- ggplot_build(p20b)[["data"]][[1]][["outliers"]]
row <- res_power_ts[which(res_power_ts$KWH_Mil == out[1]), ]
month <- as.character(row$Month)
names(out) <- month
tidyout <- purrr::map_df(out, tibble::as_tibble, .id = "month")
p20b <- p20b +
    geom_text(data = tidyout,
              aes(x = value, y = 0, label = month),
              hjust = -0.1,
              vjust = -0.6) +
    theme(axis.title.y = element_blank())
p20 <- plot_grid(p20a, p20b, ncol = 1, align = "v", axis = "l")
p20
```



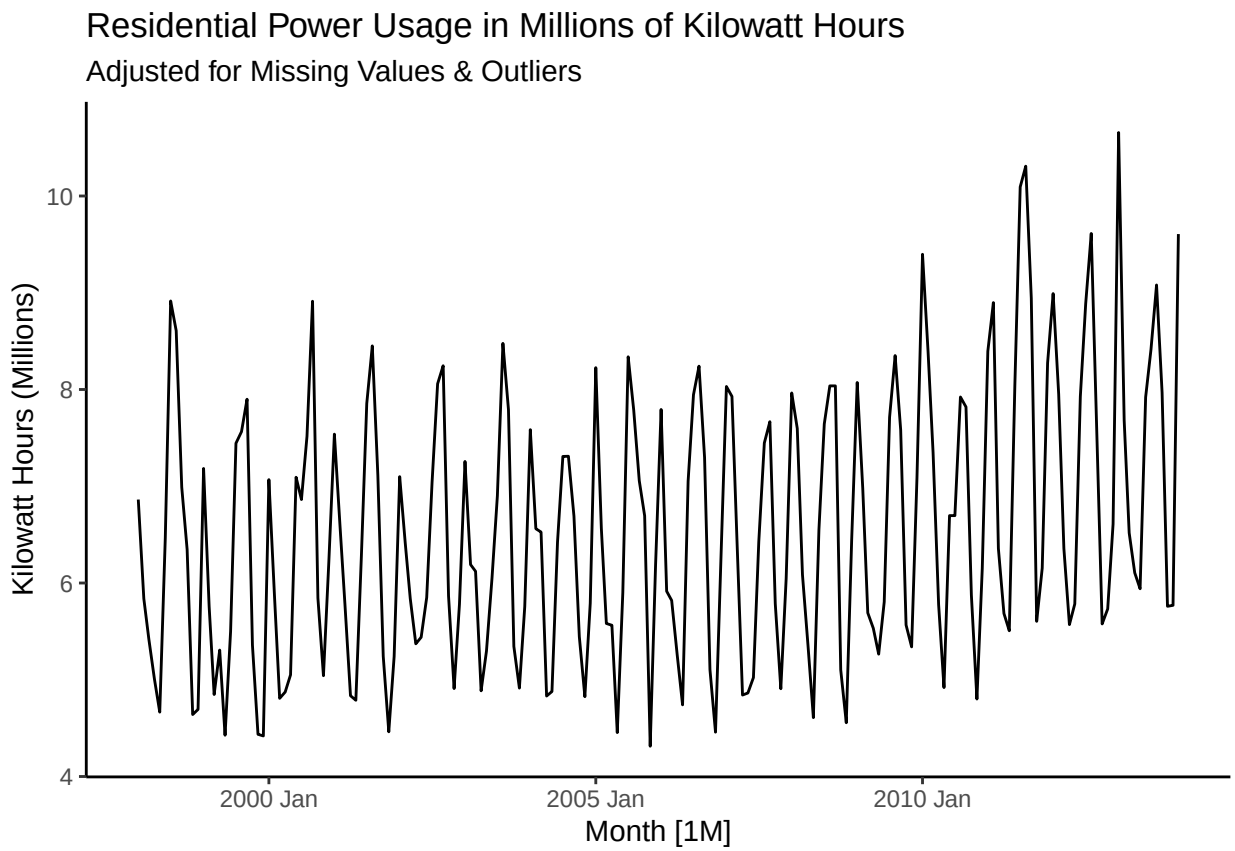
The one extreme outlier we will have to deal with is for July 2010.

We will set the outlier value for July 2010 to `NA` and replace this and the `NA` value for September 2008 via Last Observation Carried Forward (LOCF) imputation. This will reduce the weight these observations would have had on the models we create. (We considered winsorizing the data, but using that method with either a k of 1 or 5 would not be preferable here. A k of 5 would consider too many reasonable values outliers, and a k of 1 would still result in what are probably underestimates for September 2008 and July 2010.)

```
res_power_ts_na <- res_power_ts |>
    filter(KWH_Mil == out[1]) |>
    mutate(KWH = NA,
           KWH_Mil = NA)
res_power_ts <- res_power_ts|>
    filter(is.na(KWH_Mil) | KWH_Mil != out[1]) |>
    bind_rows(res_power_ts_na)
res_power_ts <- res_power_ts |>
    fill(KWH_Mil, .direction = "down")
```

We now plot the adjusted time series.

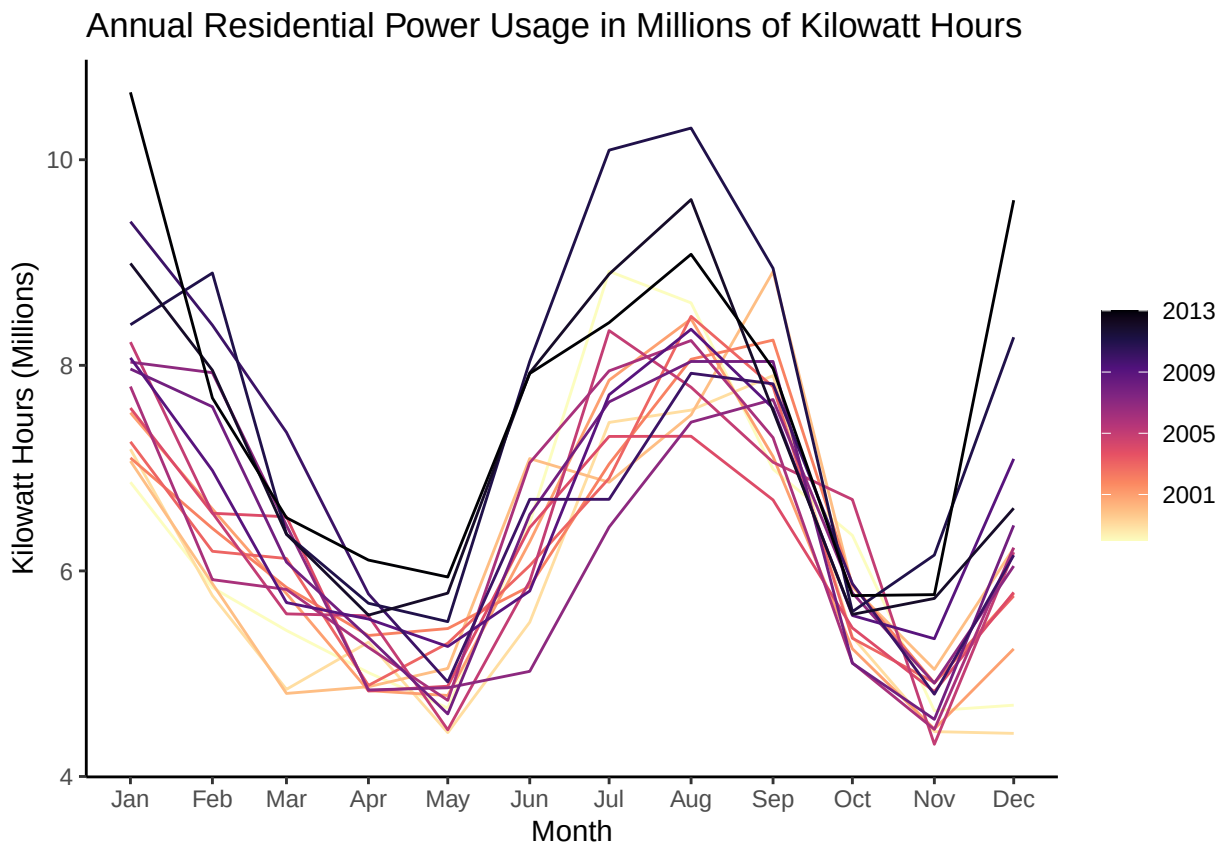
```
p21 <- res_power_ts |>
    autoplot(KWH_Mil) +
    labs(title = "Residential Power Usage in Millions of Kilowatt Hours",
         subtitle = "Adjusted for Missing Values & Outliers",
         y = "Kilowatt Hours (Millions)")
p21
```



Looking for Trend and Seasonality:

There is an upward trend apparent in the data as well as seasonality. We take a closer look at the seasonal patterns.

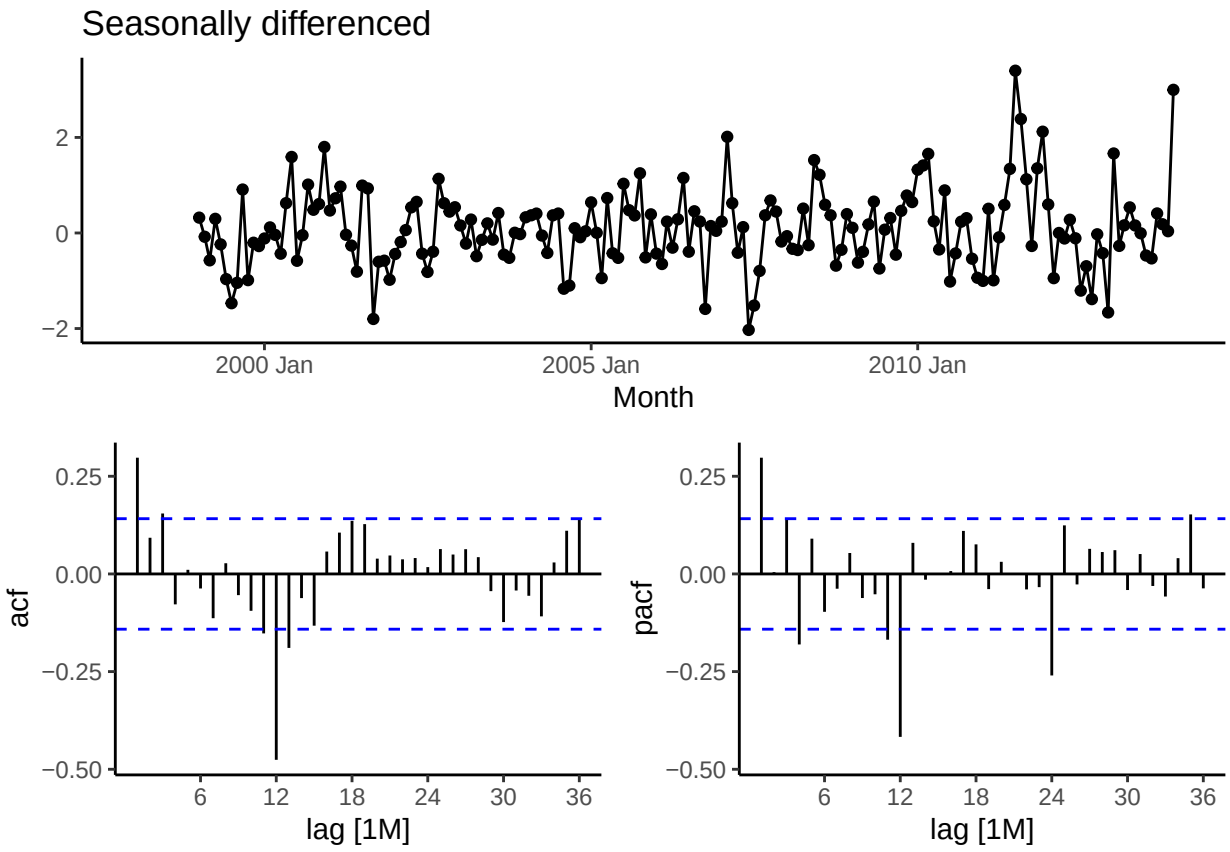
```
palette <- scales::viridis_pal(option = "magma", direction = -1)(9)
p22 <- res_power_ts |>
    gg_season(KWH_Mil, period = "year", pal = palette) +
    labs(title = "Annual Residential Power Usage in Millions of Kilowatt Hours",
         y = "Kilowatt Hours (Millions)")
p22
```



There is yearly seasonality. Residential power usage usually peaks in August and January, while usage is generally lowest in May and November.

Since the data are non-stationary, we will take a seasonal difference.

```
p23 <- res_power_ts |>
    gg_tsdisplay(difference(KWH_Mil, lag = 12),
                 plot_type = "partial", lag = 36) +
    labs(title = "Seasonally differenced", y="")
p23
```



Modeling/Forecasting:

The significant spike at lag 1 in the ACF suggests a non-seasonal MA(1) component. The significant spike at lag 12 in the ACF suggests a seasonal MA(1) component. We will propose an $ARIMA(0,0,1)(0,1,1)_{12}$ model with drift and compare it to a stepwise selected $ARIMA$ model, as well as two Holt-Winters' seasonality models: one additive and one multiplicative.

```
res_power_fit <- res_power_ts |>
    filter(year(Month) <= 2012) |>
    model(arima001011drift = ARIMA(KWH_Mil ~ 1 + pdq(0,0,1) + PDQ(0,1,1,
                                                                  period = 12)),
          stepwise = ARIMA(KWH_Mil),
          holt_wint_add = ETS(KWH_Mil ~ error("A") + trend("A") + season("A")),
          holt_wint_mult = ETS(KWH_Mil ~ error("M") + trend("A") + season("M")))
cols <- c("arima001011drift", "stepwise", "holt_wint_add", "holt_wint_mult")
pivot_res_power_fit <- res_power_fit |>
    pivot_longer(cols = all_of(cols), names_to = "Model Name",
                 values_to = "Orders")
knitr::kable(pivot_res_power_fit, format = "simple")
```

Model Name	Orders
arima001011drift	<ARIMA(0,0,1)(0,1,1)[12] w/ drift>
stepwise	<ARIMA(0,0,1)(2,1,0)[12] w/ drift>

Model Name	Orders
holt_wint_add	<ETS(A,A,A)>
holt_wint_mult	<ETS(M,A,M)>

```
drift1 <- res_power_fit |>
    select(arima001011drift) |>
    tidy() |>
    filter(term == "constant") |>
    select(estimate) |>
    as.numeric()
drift1 <- round(drift1, 2)
drift2 <- res_power_fit |>
    select(stepwise) |>
    tidy() |>
    filter(term == "constant") |>
    select(estimate) |>
    as.numeric()
drift2 <- round(drift2, 2)
```

The drift term for the $ARIMA$ model we proposed is 0.08, while the drift term for the stepwise selected $ARIMA$ model is larger at 0.2. Both reflect the general upward trend we noted earlier.

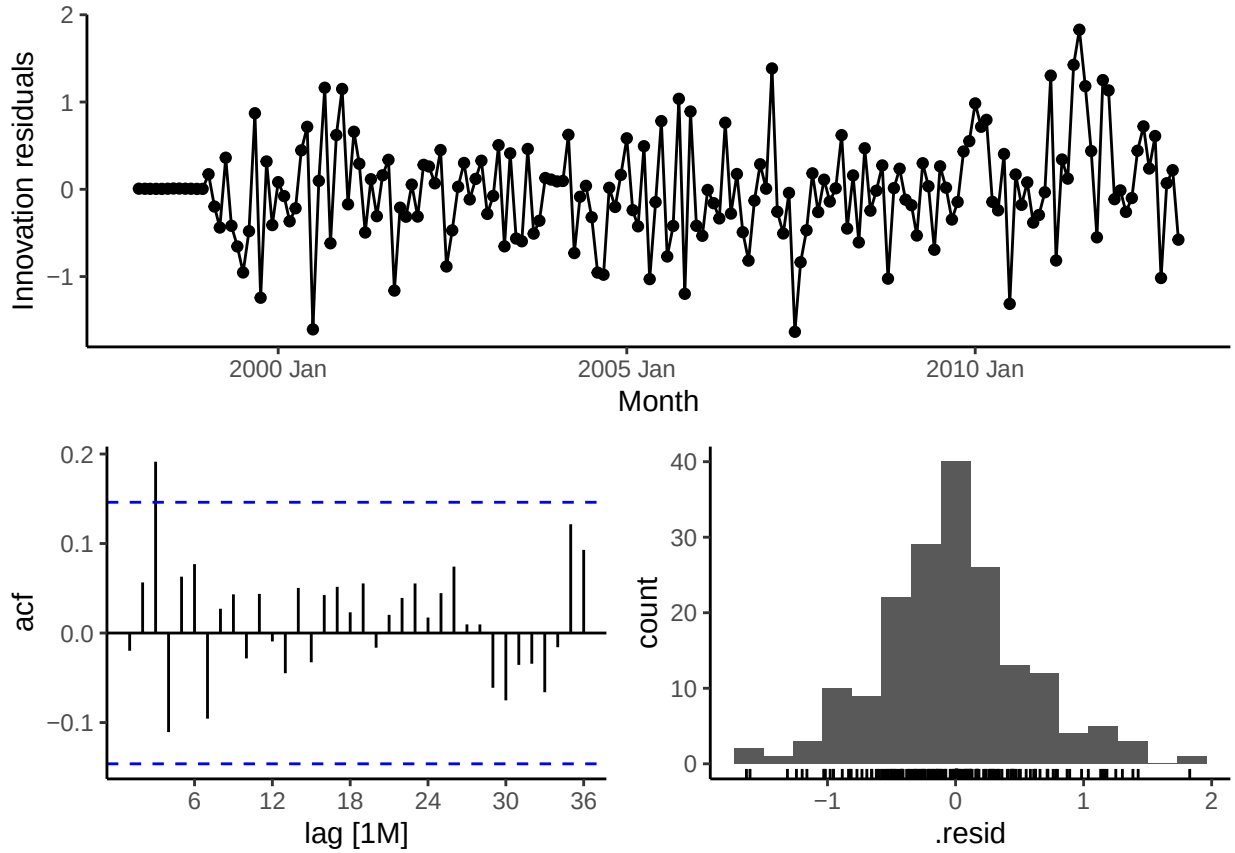
```
exclude <- c("holt_wint_add", "holt_wint_mult")
glance_res_power_fit <- glance(res_power_fit) |>
    arrange(AICc) |>
    select(.model:BIC) |>
    filter(!.model %in% exclude)
knitr::kable(glance_res_power_fit)
```

.model	sigma2	log_lik	AIC	AICc	BIC
arima001011drift	0.3614993	-156.4981	320.9962	321.2416	333.4920
stepwise	0.3640105	-155.7435	321.4871	321.8574	337.1069

Between the $ARIMA$ model we proposed and the stepwise selected $ARIMA$ model, the model we proposed has slightly lower AIC_c. We have excluded the Holt-Winters' additive/multiplicative seasonality models from an AIC_c comparison because $ARIMA$ and ETS models cannot be directly compared that way. We will instead compare the residual diagnostic plots and the forecast performance of the $ARIMA$ model we proposed and the Holt-Winters' models.

First we look at residual diagnostic plots for the $ARIMA$ model.

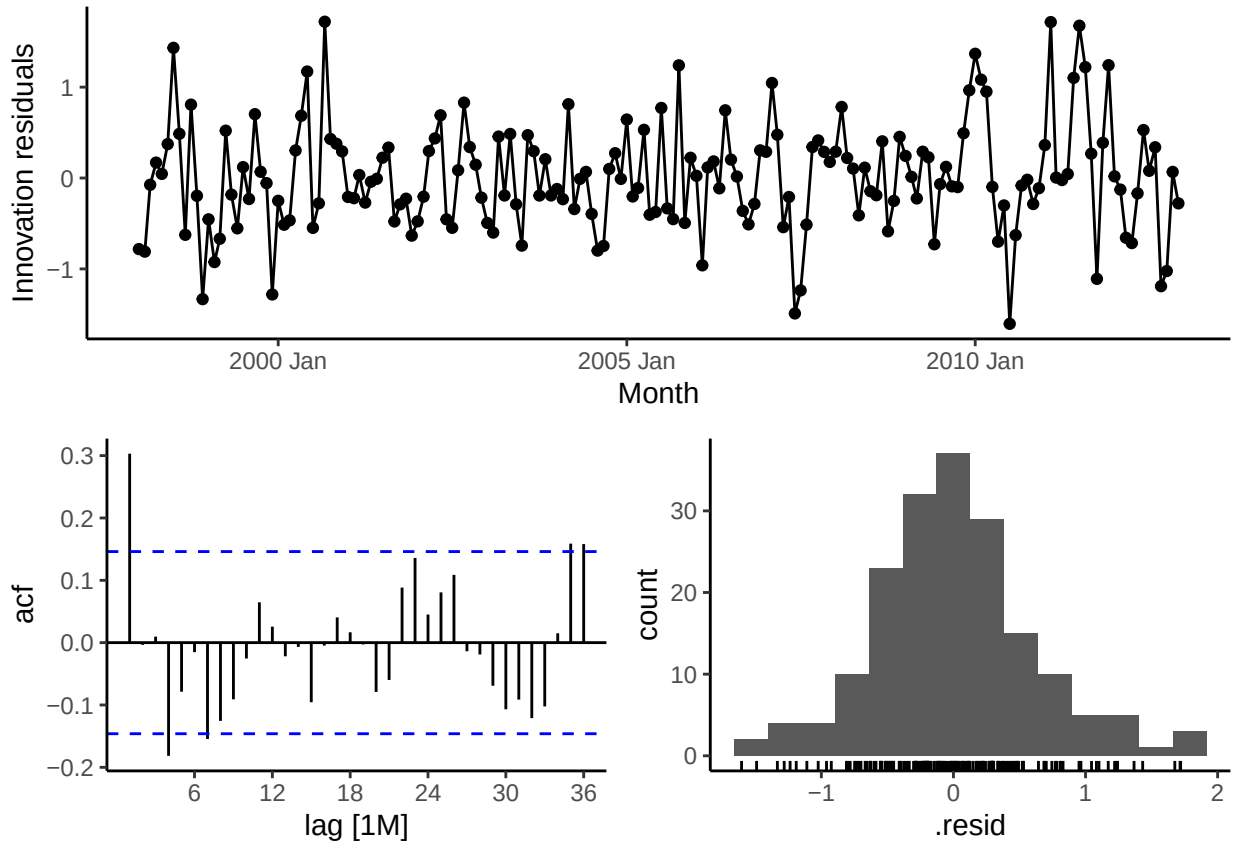
```
p24 <- res_power_fit |>
    select(arima001011drift) |>
    gg_tsresiduals(lag=36)
p24
```



One small spike outside the bounds of the ACF plot is still consistent with white noise, so the $ARIMA$ model appears to capture all the dynamics in the data pretty well.

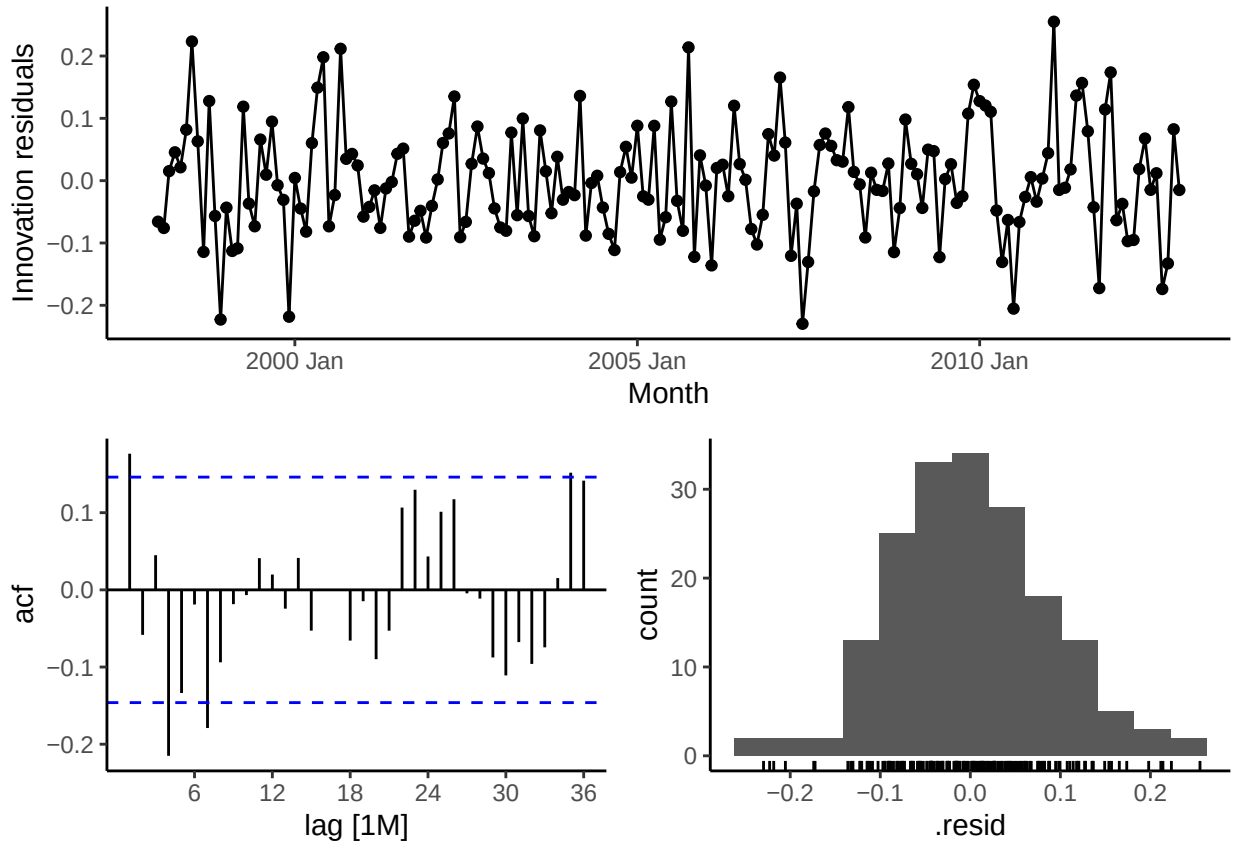
Next we look at residual diagnostic plots for the ETS models.

```
p25 <- res_power_fit |>
    select(holt_wint_add) |>
    gg_tsresiduals(lag=36)
p25
```



With one large spike outside the bounds of the ACF plot as well as some additional small spikes, the additive Holt-Winters' model doesn't appear to capture the data dynamics as well as the $ARIMA$ model.

```
p26 <- res_power_fit |>
    select(holt_wint_mult) |>
    gg_tsresiduals(lag=36)
p26
```



The multiplicative Holt-Winters' model doesn't appear to capture the data dynamics any better.

Let's confirm whether the Ljung-Box tests indicate the residuals for the $ARIMA$ model are indistinguishable from white noise, but the residuals for both ETS models are not.

```
dof <- res_power_fit |>
    select(arima001011drift) |>
    tidy() |>
    filter(term != "constant") |>
    NROW()
m = 12 #period of seasonality
l = 2*m #for seasonal data, otherwise l = 10
augment(res_power_fit) |>
    filter(.model == "arima001011drift") |>
    features(.innov, ljung_box, lag=l, dof=dof)
```

```
## # A tibble: 1 x 3
##   .model           lb_stat lb_pvalue
##   <chr>              <dbl>     <dbl>
## 1 arima001011drift    18.2     0.691
```

The large p-value for the $ARIMA$ model's tests indicates the residuals from the $ARIMA$ model are in fact indistinguishable from white noise.

```
augment(res_power_fit) |>
    filter(.model %in% exclude) |>
    features(.innov, ljung_box, lag=1)
```

```
## # A tibble: 2 x 3
##   .model         lb_stat lb_pvalue
##   <chr>            <dbl>     <dbl>
## 1 holt_wint_add     44.6   0.00657
## 2 holt_wint_mult    37.4   0.0403
```

The small p-values for the *ETS* models' tests indicate there is information in the data not being captured by these models, as we suspected.

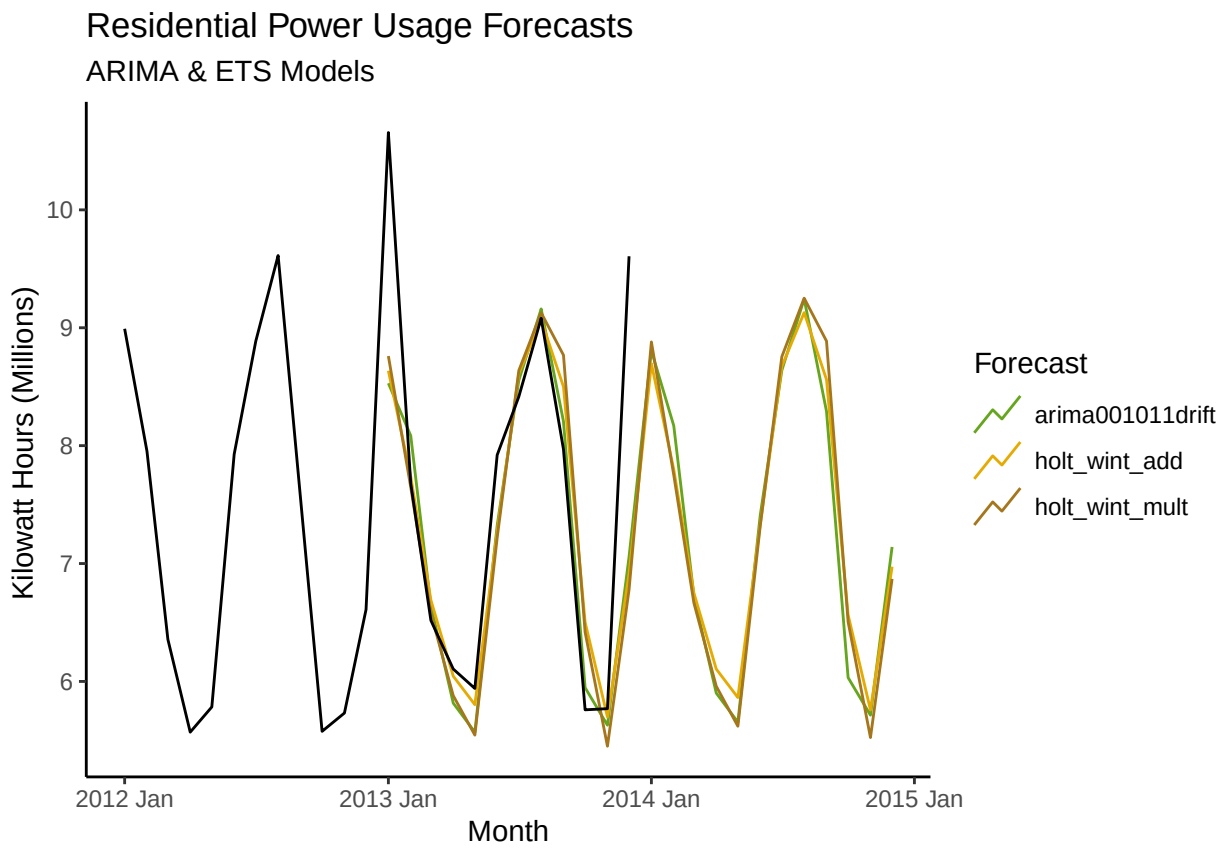
Now we can compare forecasts.

```
accuracy <- res_power_fit |>
    select(-stepwise)
accuracy <- bind_rows(accuracy |> accuracy(),
                      accuracy |> forecast(h = 12) |> accuracy(res_power_ts)) |>
    select(-ME, -MPE, -ACF1)
knitr::kable(accuracy)
```

.model	.type	RMSE	MAE	MAPE	MASE	RMSSE
arima001011drift	Training	0.5756513	0.4284426	6.693421	0.6865167	0.7033699
holt_wint_add	Training	0.5966002	0.4549045	7.051341	0.7289181	0.7289667
holt_wint_mult	Training	0.5873320	0.4516355	7.018337	0.7236800	0.7176422
arima001011drift	Test	0.9966323	0.6035823	6.945364	0.9671526	1.2177532
holt_wint_add	Test	1.0284351	0.6121385	7.097938	0.9808627	1.2566121
holt_wint_mult	Test	1.0614635	0.6819801	8.159838	1.0927736	1.2969683

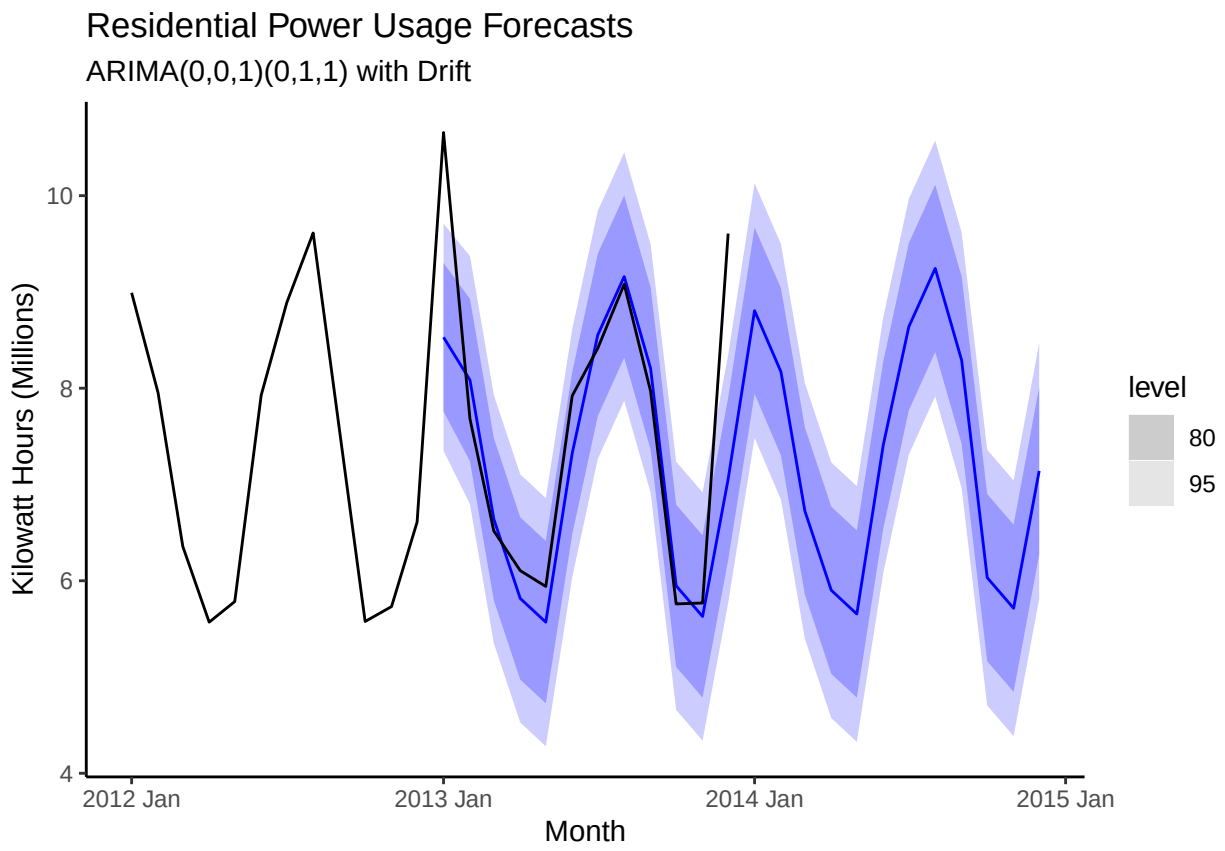
The *ARIMA* model is slightly more accurate based on the test set RMSE, MAPE and MASE. We will visually compare the forecasts for all three models even though we already have good evidence to choose the *ARIMA* model for forecasting.

```
palette <- brewer.pal(n = 8, name = "Dark2")
model_colors <- palette[5:7]
names(model_colors) <- c("arima001011drift", "holt_wint_add", "holt_wint_mult")
res_power_ts_since_2012 <- res_power_ts |>
    filter(year(Month) >= 2012)
p27 <- res_power_fit |>
    forecast(h=24) |>
    filter(.model != "stepwise") |>
    autoplot(res_power_ts_since_2012, level = NULL) +
    labs(title = "Residential Power Usage Forecasts",
         subtitle = "ARIMA & ETS Models",
         y="Kilowatt Hours (Millions) ") +
    guides(color = guide_legend(title = "Forecast")) +
    scale_color_manual(values = model_colors)
p27
```



Looking at 2013 forecasts from the three different models vs. the actual values recorded for 2013, the models actually perform pretty similarly. They all get close to the data, and different models make closer predictions at different points. However, we know that the $ARIMA$ model makes the closest predictions overall and captures the dynamics in the data the best, so we choose it as the superior forecasting model here, replot the forecasts for only this model with prediction intervals, and save the forecasts to an Excel file.

```
p28 <- res_power_fit |>
    forecast(h=24) |>
    filter(.model == "arima001011drift") |>
    autoplot(res_power_ts_since_2012) +
    labs(title = "Residential Power Usage Forecasts",
         subtitle = "ARIMA(0,0,1)(0,1,1) with Drift",
         y="Kilowatt Hours (Millions)")
p28
```




```
partb_excel_forecasts <- res_power_fit |>
    forecast(h=24) |>
    filter(.model == "arima001011drift" & year(Month) == 2014) |>
    as_tibble() |>
    mutate(Month = as.character(Month),
           KWH_Mil = as.character(KWH_Mil))
write_xlsx(partb_excel_forecasts, "partb_excel_forecasts.xlsx")
```

Part C:

Data Preparation:

We load data on waterflow for two pipes.

```
my_url3 <- "https://github.com/geedoubledee/data624_project1/raw/main/Waterflow_Pipe1.xlsx"
my_url4 <- "https://github.com/geedoubledee/data624_project1/raw/main/Waterflow_Pipe2.xlsx"
col_types <- c("date", "numeric")
col_names <- c("DateTime", "WaterFlow")
temp <- tempfile(fileext = ".xlsx")
req <- GET(my_url3, authenticate(Sys.getenv("GITHUB_PAT"), ""),
           write_disk(path = temp))
pipe1 <- readxl::read_excel(temp, col_types = col_types)
temp <- tempfile(fileext = ".xlsx")
req <- GET(my_url4, authenticate(Sys.getenv("GITHUB_PAT"), ""),
```

```
          write_disk(path = temp))
pipe2 <- readxl::read_excel(temp, col_types = col_types)
colnames(pipe1) <- col_names
colnames(pipe2) <- col_names
```

The data for Pipe1 need to be reduced to the hour level of precision, then aggregated and summarized to get the mean value per hour.

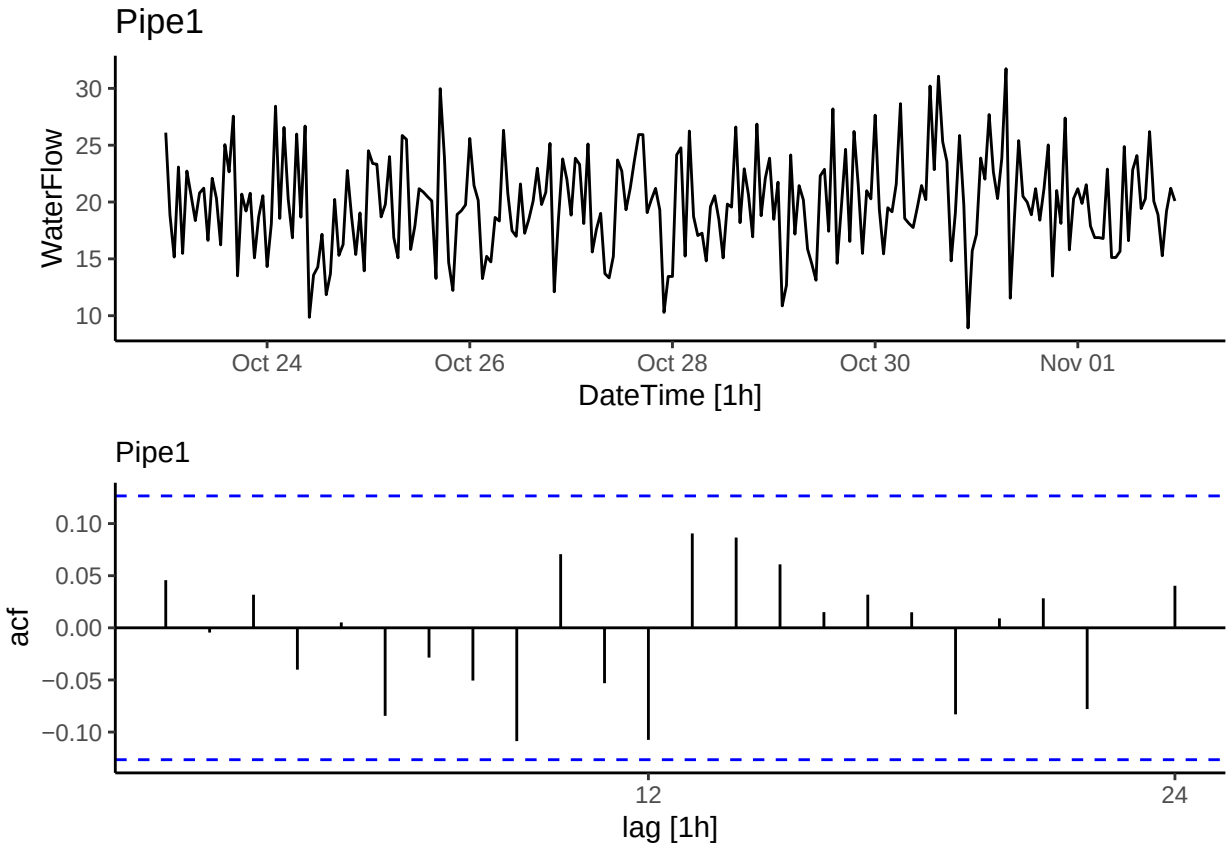
```
pipe1 <- pipe1 |>
    mutate(DateTime = floor_date(DateTime, "hour")) |>
    group_by(DateTime) |>
    summarize(WaterFlow = mean(WaterFlow))
```

Now we can create and plot time series for Pipes 1 and 2 alongside their ACF plots to determine if the data are stationary.

Missing Data and Outliers:

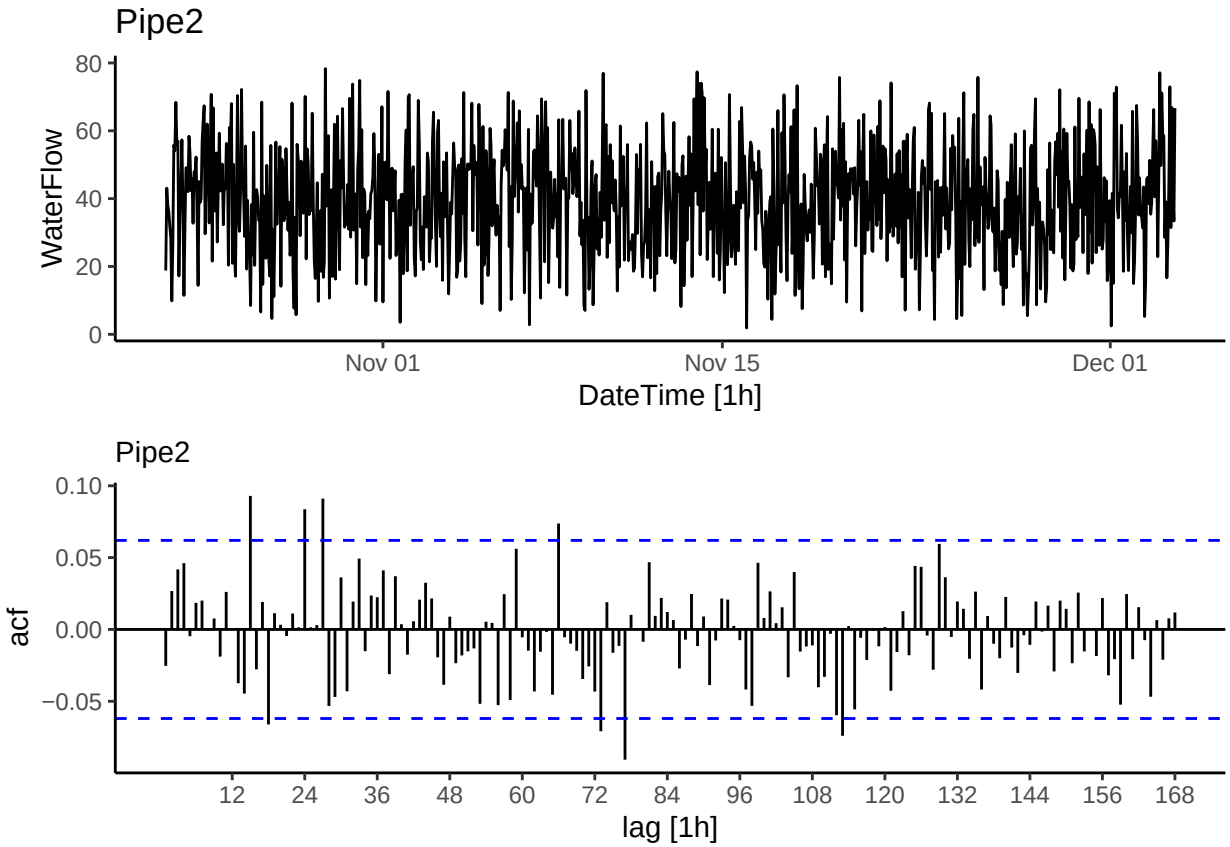
There are some gaps in the time series for Pipe1, so we will create NA observations for those gaps and fill them using Last Observation Carried Forward (LOCF).

```
pipe1_ts <- pipe1 |>
    as_tsibble(index = DateTime) |>
    fill_gaps() |>
    fill(WaterFlow, .direction = "down")
pipe2_ts <- pipe2 |>
    as_tsibble(index = DateTime)
p29a <- pipe1_ts |>
    autoplot(WaterFlow) +
    labs(title = "Pipe1")
p30a <- pipe2_ts |>
    autoplot(WaterFlow) +
    labs(title = "Pipe2")
p29b <- pipe1_ts |>
    ACF(WaterFlow, lag_max = 24) |>
    autoplot() +
    labs(subtitle = "Pipe1")
p30b <- pipe2_ts |>
    ACF(WaterFlow, lag_max = 168) |>
    autoplot() +
    labs(subtitle = "Pipe2")
p29 <- plot_grid(p29a, p29b, ncol = 1, align = "v", axis = "l")
p30 <- plot_grid(p30a, p30b, ncol = 1, align = "v", axis = "l")
p29
```



The hourly data for Pipe1 do appear stationary. There's no daily seasonality. We don't have enough data for Pipe1 to see if there are weekly rather than daily patterns, but we do have enough data to see if there are weekly patterns for Pipe2.

```
p30
```



There don't appear to be weekly patterns in the data for Pipe2. There is some autocorrelation at r24, but it is not the largest spike in the ACF. So there may be a seasonal MA(1) component for Pipe 2, or there may not be.

Modeling/Forecasting:

We will fit a white noise $ARIMA(0,0,0)$ model with mean and a stepwise selected $ARIMA$ model for comparison for Pipe1.

```
pipe1_fit <- pipe1_ts |>
    filter(month(DateTime) <= 10) |>
    model(whitenoise = ARIMA(WaterFlow ~ 1 + pdq(0,0,0)),
          stepwise = ARIMA(WaterFlow))
cols <- c("whitenoise", "stepwise")
pivot_pipe1_fit <- pipe1_fit |>
    pivot_longer(cols = all_of(cols), names_to = "Model Name",
                 values_to = "Orders")
knitr::kable(pivot_pipe1_fit, format = "simple")
```

Model Name	Orders
whitenoise	<ARIMA(0,0,0) w/ mean>
stepwise	<ARIMA(0,0,0) w/ mean>

The stepwise model selected for Pipe1 matches our white noise model with mean.

We will fit a white noise $ARIMA(0,0,0)(0,0,0)_{24}$ model with mean and a stepwise selected $ARIMA$ model for comparison for Pipe2.

```
pipe2_fit <- pipe2_ts |>
    filter(month(DateTime) <= 11) |>
    model(whitenoise = ARIMA(WaterFlow ~ 1 + pdq(0,0,0) + PDQ(0, 0, 0,
                                                              period = 24)),
          stepwise = ARIMA(WaterFlow))
cols <- c("whitenoise", "stepwise")
pivot_pipe2_fit <- pipe2_fit |>
    pivot_longer(cols = all_of(cols), names_to = "Model Name",
                 values_to = "Orders")
knitr::kable(pivot_pipe2_fit, format = "simple")
```

Model Name	Orders
whitenoise	<ARIMA(0,0,0) w/ mean>
stepwise	<ARIMA(1,0,0)(0,0,1)[24] w/ mean>

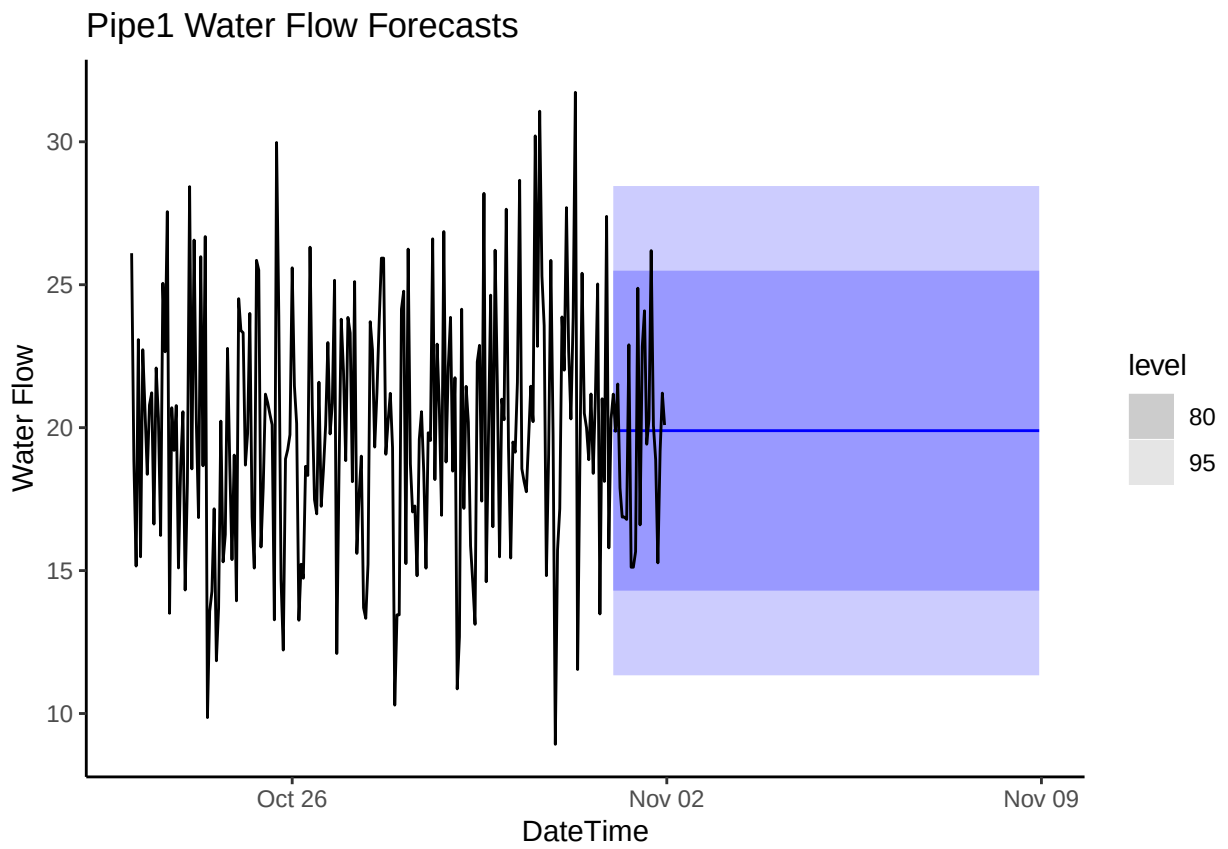
The stepwise selected $ARIMA$ model for Pipe2 picks up on the seasonal MA(1) component we weren't certain about and adds a nonseasonal AR(1) component. We compare AIC_c scores.

```
glance_pipe2_fit <- glance(pipe2_fit) |>
    arrange(AICc) |>
    select(.model:BIC)
knitr::kable(glance_pipe2_fit)
```

.model	sigma2	log_lik	AIC	AICc	BIC
stepwise	251.9361	-3910.227	7828.455	7828.498	7847.817
whitenoise	254.5841	-3915.985	7835.970	7835.983	7845.651

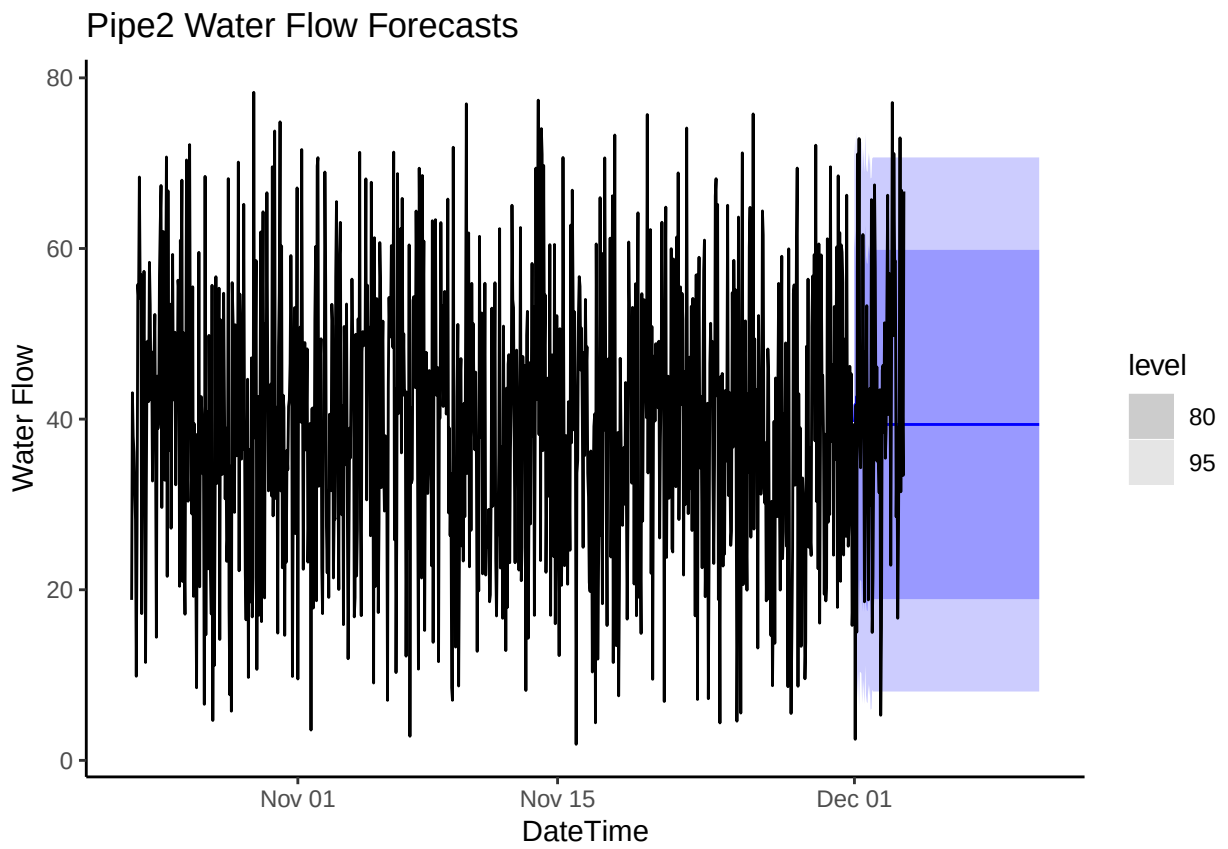
The stepwise selected model has a slightly better AIC_c score, so it's better than the white noise model that assumes no autocorrelation. We forecast for Pipe1 using the white noise model and for Pipe2 using the stepwise model.

```
p31 <- pipe1_fit |>
    forecast(h=192) |>
    filter(.model == "whitenoise") |>
    autoplot(pipe1_ts) +
    labs(title = "Pipe1 Water Flow Forecasts",
         y="Water Flow")
p31
```



The forecasts for Pipe1 are equivalent to using a MEAN model.

```
p32 <- pipe2_fit |>
    forecast(h=240) |>
    filter(.model == "stepwise") |>
    autoplot(pipe2_ts) +
    labs(title = "Pipe2 Water Flow Forecasts",
         y="Water Flow")
p32
```



The forecasts for Pipe2 approach the mean very quickly despite being more complex for earlier forecasts. It even happens so quickly that the forecasts have reached the mean before the end of the training data.

Both forecasts are reasonable. We save them to separate Excel files.

```
partc_excel_forecasts_pipe1 <- pipe1_fit |>
    forecast(h=192) |>
    filter(.model == "whitenoise" & as.Date(DateTime) > as.Date("2015-11-01")) |>
    as_tibble() |>
    mutate(DateTime = as.character(DateTime),
           WaterFlow = as.character(WaterFlow))
write_xlsx(partc_excel_forecasts_pipe1, "partc_excel_forecasts_pipe1.xlsx")

partc_excel_forecasts_pipe2 <- pipe2_fit |>
    forecast(h=240) |>
    filter(.model == "stepwise" & as.Date(DateTime) > as.Date("2015-12-03")) |>
    as_tibble() |>
    mutate(DateTime = as.character(DateTime),
           WaterFlow = as.character(WaterFlow))
write_xlsx(partc_excel_forecasts_pipe2, "partc_excel_forecasts_pipe2.xlsx")
```