Data 624_Project 1

Jiaxin Zheng

2025-04-05

Part A

```
library(fpp3)
```

Download the library and load the data

```
## Warning: package 'fpp3' was built under R version 4.4.2

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

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

## v tibble      3.2.1     v tsibble     1.1.6
## v dplyr       1.1.4     v tsibbledata 0.4.1
## v tidyr       1.3.1     v feasts      0.4.1
## v lubridate   1.9.4     v fable       0.4.1
## v ggplot2     3.5.1

## Warning: package 'dplyr' was built under R version 4.4.3

## Warning: package 'ggplot2' was built under R version 4.4.2

## Warning: package 'tsibbledata' was built under R version 4.4.2

## Warning: package 'feasts' was built under R version 4.4.2

## Warning: package 'fabletools' was built under R version 4.4.2

## Warning: package 'fable' was built under R version 4.4.2

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

```r
library(dplyr)
library(readxl)
library(ggplot2)
library(lubridate)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(tsibble)
library(readr)
```

```
## Warning: package 'readr' was built under R version 4.4.3
```

```r
library(openxlsx)
```

```
## Warning: package 'openxlsx' was built under R version 4.4.3
```

```r
atm_data <- read.csv ('https://raw.githubusercontent.com/Jennyjjxxzz/Data-624_Project1/refs/heads/main/

head(atm_data)
```

```
##                   DATE  ATM Cash
## 1 5/1/2009 12:00:00 AM ATM1   96
## 2 5/1/2009 12:00:00 AM ATM2  107
## 3 5/2/2009 12:00:00 AM ATM1   82
## 4 5/2/2009 12:00:00 AM ATM2   89
## 5 5/3/2009 12:00:00 AM ATM1   85
## 6 5/3/2009 12:00:00 AM ATM2   90
```

```r
atm_data <- atm_data %>%
  mutate(DATE = mdy_hms(DATE),
         DATE = as.Date(DATE)) %>%
  as_tsibble(index = DATE, key = ATM) %>%
  arrange(DATE)

head(atm_data)
```

Transfer into tsibble dataframe

```
## # A tsibble: 6 x 3 [1D]
## # Key:       ATM [4]
##   DATE       ATM    Cash
```

```
##   <date>     <chr> <int>
## 1 2009-05-01 ATM1     96
## 2 2009-05-01 ATM2    107
## 3 2009-05-01 ATM3      0
## 4 2009-05-01 ATM4    777
## 5 2009-05-02 ATM1     82
## 6 2009-05-02 ATM2     89
```

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

Find the missing value from the data.

```
## # A tsibble: 19 x 3 [1D]
## # Key:       ATM [3]
##    DATE       ATM     Cash
##    <date>     <chr>  <int>
##  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
##  7 2010-05-02 ""        NA
##  8 2010-05-03 ""        NA
##  9 2010-05-04 ""        NA
## 10 2010-05-05 ""        NA
## 11 2010-05-06 ""        NA
## 12 2010-05-07 ""        NA
## 13 2010-05-08 ""        NA
## 14 2010-05-09 ""        NA
## 15 2010-05-10 ""        NA
## 16 2010-05-11 ""        NA
## 17 2010-05-12 ""        NA
## 18 2010-05-13 ""        NA
## 19 2010-05-14 ""        NA
```

```
atm_data %>%
  filter_index(~'2010-4-30') %>%
  filter(is.na(Cash))
```

Fillter and find out how many missing data for ATMs

```
## # A tsibble: 5 x 3 [1D]
## # Key:       ATM [2]
##   DATE       ATM    Cash
##   <date>     <chr> <int>
## 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
```

```
atm_df <- atm_data %>%
  filter_index(~'2010-04-30') %>%     # Only use data up to April 30, 2010
  fill_gaps() %>%
  filter(!is.na(ATM)) %>%
  group_by(ATM) %>%
  mutate(Cash = if_else(
    is.na(Cash),
    round(mean(Cash, na.rm = TRUE), 0),
    Cash
  )) %>%
  ungroup()
```

```
head(atm_df)
```

Impute the missing values with the means.

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

- Double check for missing values

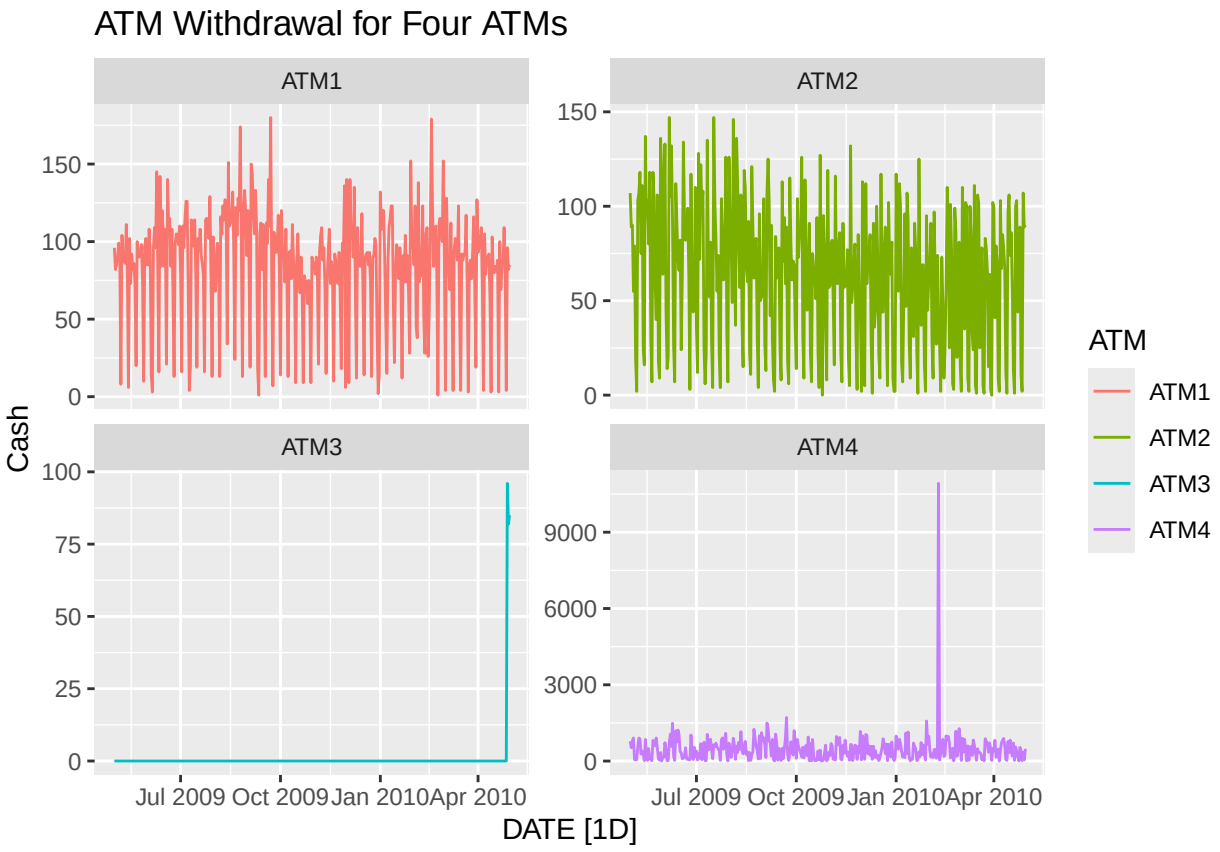
```
sum(is.na(atm_df$Cash))
```

```
## [1] 0
```

Autoplot the atm_data, for better understing for the dataset

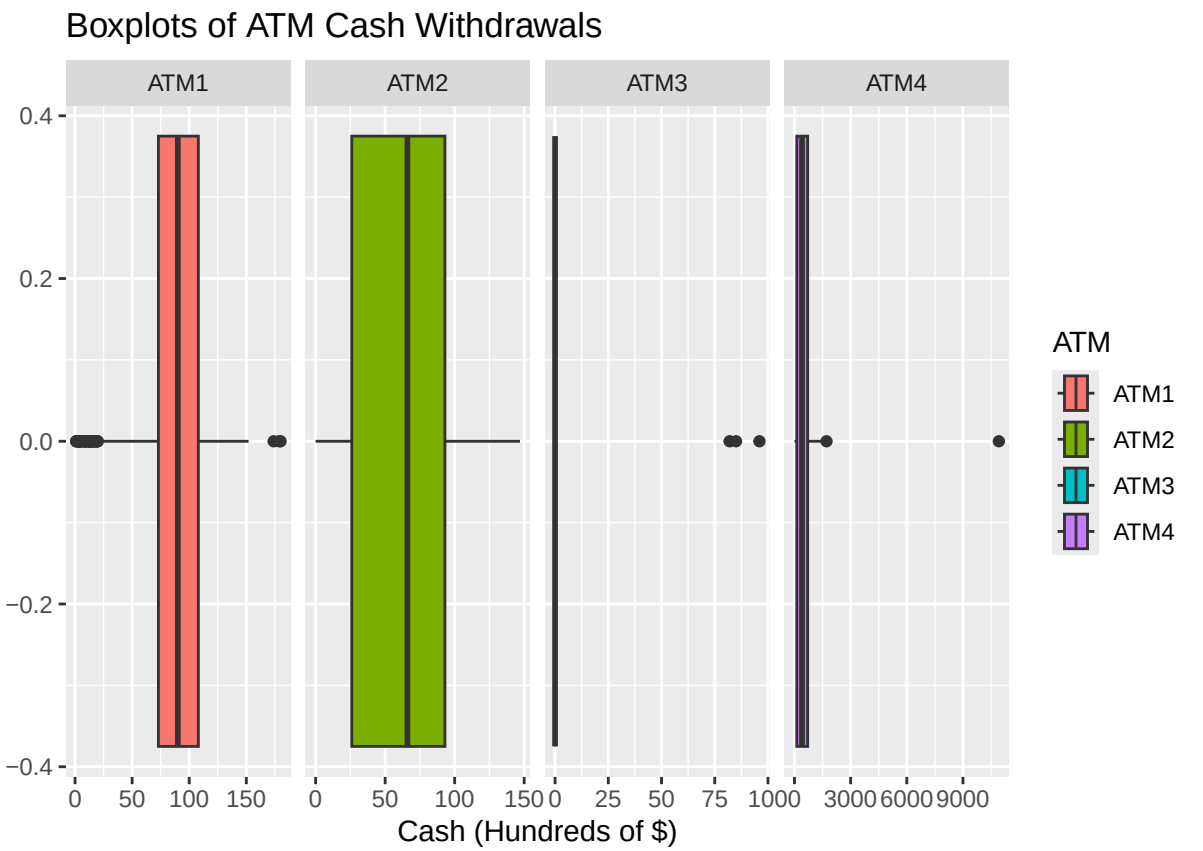
- There are outliers for ATM3 and ATM4

```
atm_df %>%
  autoplot(Cash) +
  facet_wrap(~ATM, scale='free_y') +
  ggtitle('ATM Withdrawal for Four ATMs')
```



- In the Boxplot, we can see there are some outlier in ATM1 too.

```
plot <- atm_df %>%
  ggplot(aes(x = Cash)) +
  geom_boxplot(aes(fill = ATM)) +
  facet_wrap(~ATM, nrow = 1, scales = "free_x") +
  labs(
    title = "Boxplots of ATM Cash Withdrawals",
    y = "",
    x = "Cash (Hundreds of $)"
  )
plot
```



Fillting the ATMs into the models(ETS, ARIMA, Naive) to see the RMSE

- As a result, we can see that ARIMA is the best model to use for forecasting for all the ATMs because ARIMA has the lowest RMSE.

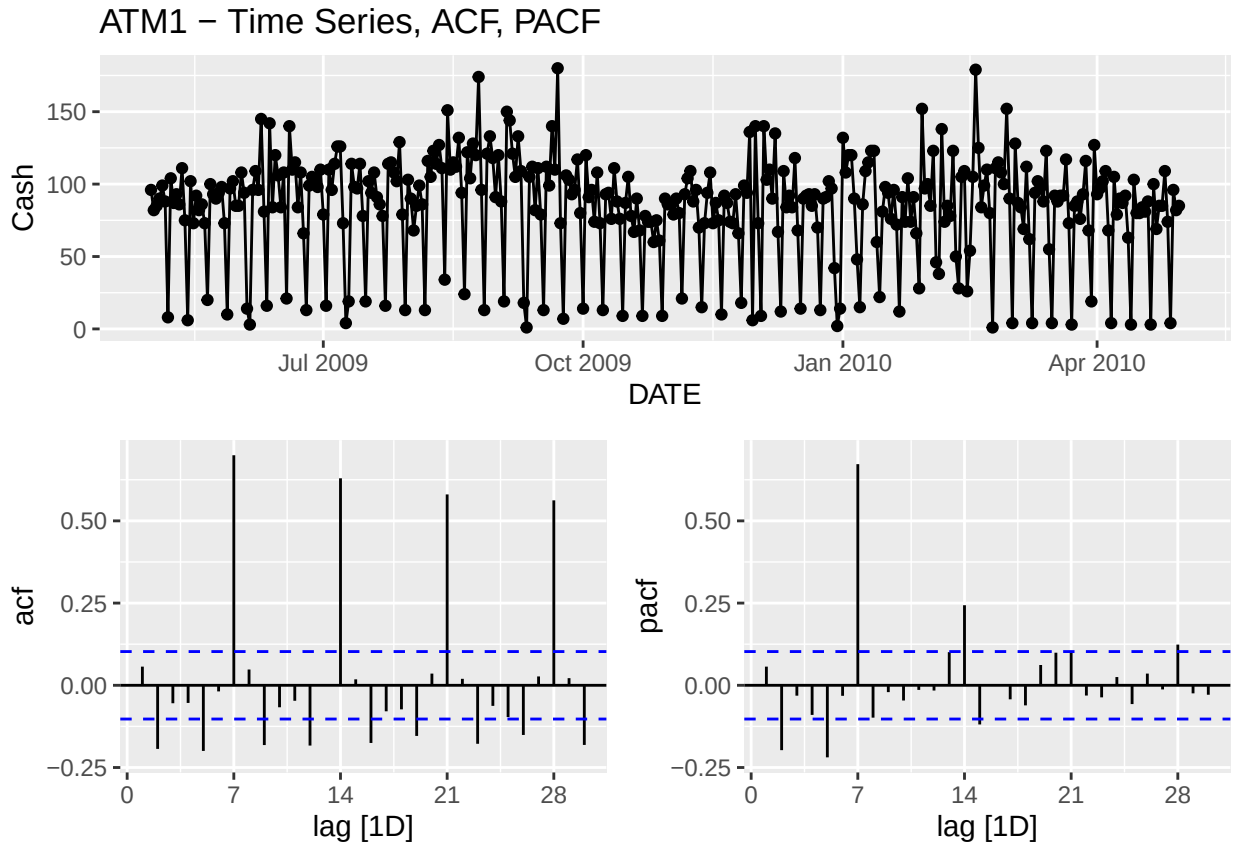
```
atm_fit_all <- atm_df %>%
  model(
    ETS = ETS(Cash),
    ARIMA = ARIMA(Cash, stepwise = FALSE),
    Naive = NAIVE(Cash),
    snaive = SNAIVE(Cash)
  )

accuracy_results <- atm_fit_all %>%
  accuracy() %>%
  arrange(ATM, RMSE)

accuracy_results
```

```
## # A tibble: 16 x 11
##    ATM   .model .type        ME   RMSE     MAE    MPE  MAPE  MASE RMSSE     ACF1
##    <chr> <chr>  <chr>     <dbl>  <dbl>   <dbl>  <dbl> <dbl> <dbl> <dbl>    <dbl>
##  1 ATM1  ARIMA  Trai~ -7.36e- 2  23.4   14.7   -103.  118.  0.825 0.840 -0.00795
##  2 ATM1  ETS    Trai~ -1.80e- 1  23.8   15.1   -107.  121.  0.846 0.854  0.125
##  3 ATM1  snaive Trai~ -3.63e- 2  27.9   17.8    -96.6 116.  1     1      0.151
```

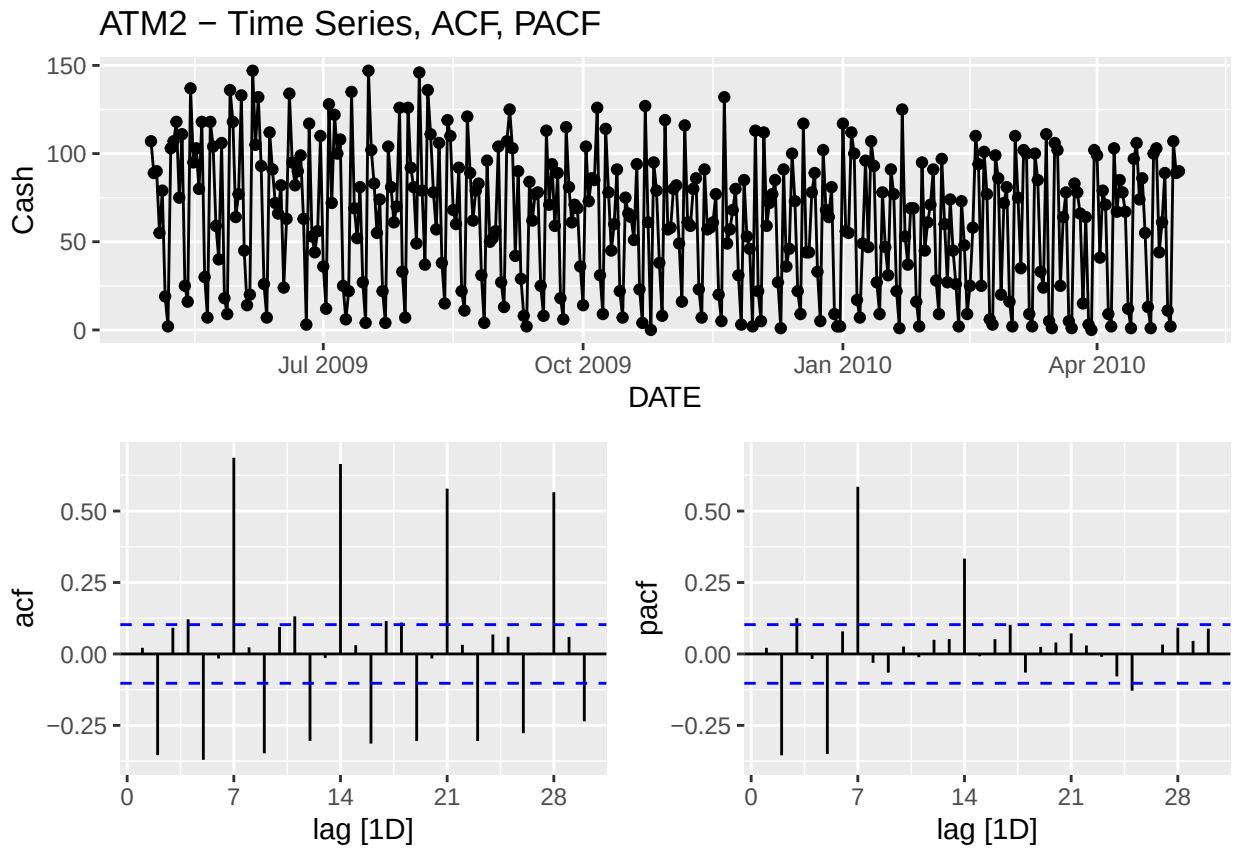
```
##  4 ATM1  Naive  Trai~ -3.02e- 2  50.1   37.7   -132.  167.  2.11  1.80  -0.367
##  5 ATM2  ARIMA  Trai~ -8.06e- 1  24.1   17.3   -Inf   Inf   0.835 0.803 -0.00893
##  6 ATM2  ETS    Trai~ -5.85e- 1  25.0   17.7   -Inf   Inf   0.851 0.830  0.0163
##  7 ATM2  snaive Trai~  2.23e- 2  30.1   20.8   -Inf   Inf   1     1      0.0458
##  8 ATM2  Naive  Trai~ -4.67e- 2  54.2   42.5   -Inf   Inf   2.05  1.80  -0.308
##  9 ATM3  ARIMA  Trai~  2.71e- 1   5.03   0.271   34.6  34.6 0.370 0.625  0.0124
## 10 ATM3  ETS    Trai~  2.70e- 1   5.03   0.273  Inf   Inf   0.371 0.625 -0.00736
## 11 ATM3  Naive  Trai~  2.34e- 1   5.09   0.310   28.8  40.2 0.423 0.632 -0.149
## 12 ATM3  snaive Trai~  7.35e- 1   8.04   0.735  100   100   1     1      0.640
## 13 ATM4  ETS    Trai~ -7.89e- 1 647.   310.    -556.  588.  0.770 0.722 -0.00587
## 14 ATM4  ARIMA  Trai~ -1.21e-10 650.   324.    -590.  620.  0.805 0.725 -0.00936
## 15 ATM4  snaive Trai~ -3.70e+ 0 896.   402.    -386.  441.  1     1     -0.0150
## 16 ATM4  Naive  Trai~ -8.10e- 1 925.   444.    -529.  589.  1.10  1.03  -0.488
```

- Plot to see the ACF and PACF plots

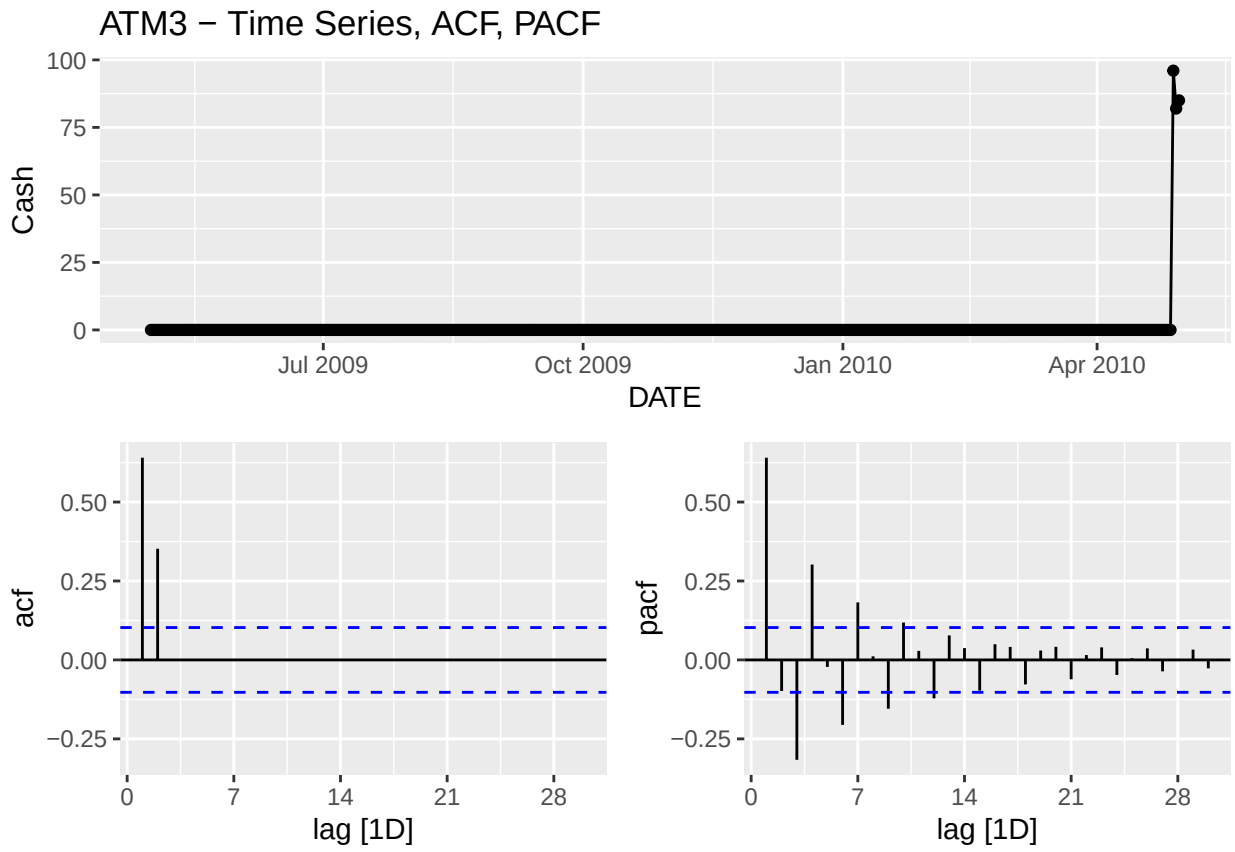
```
atm_df %>%
  filter(ATM == "ATM1") %>%
  gg_tsdisplay(Cash, plot_type = 'partial', lag = 30) +
  labs(title = "ATM1 - Time Series, ACF, PACF")
```



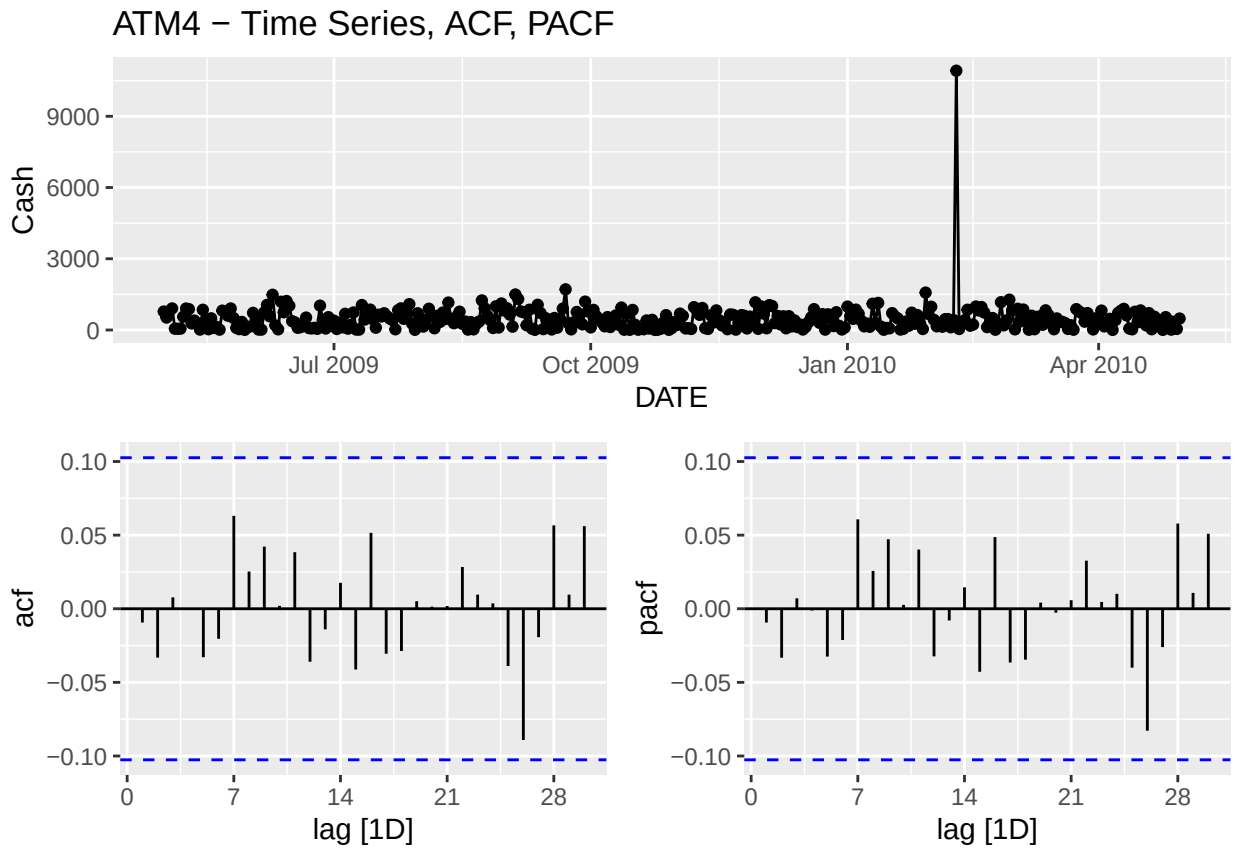

```
atm_df %>%
  filter(ATM == "ATM2") %>%
  gg_tsdisplay(Cash, plot_type = 'partial', lag = 30) +
  labs(title = "ATM2 - Time Series, ACF, PACF")
```




```
atm_df %>%
  filter(ATM == "ATM3") %>%
  gg_tsdisplay(Cash, plot_type = 'partial', lag = 30) +
  labs(title = "ATM3 - Time Series, ACF, PACF")
```

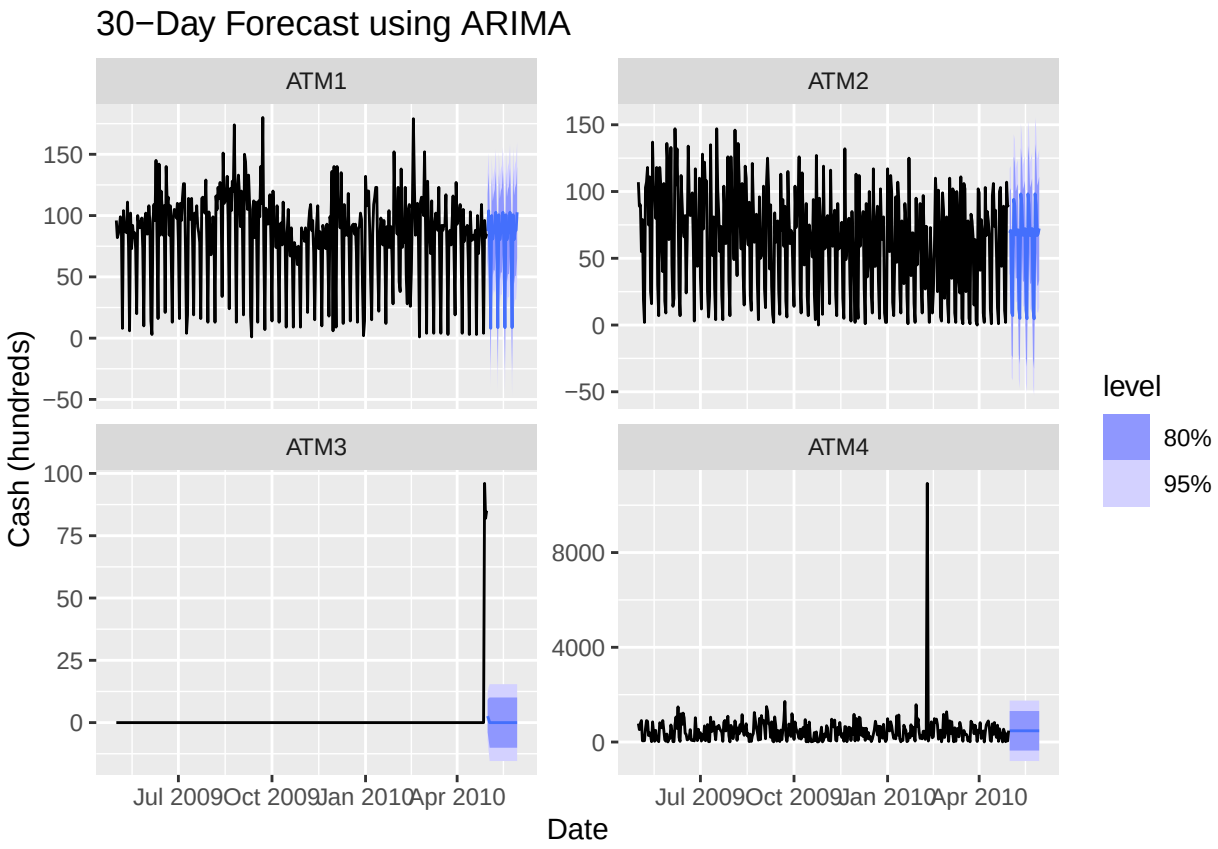



```
atm_df %>%
  filter(ATM == "ATM4") %>%
  gg_tsdisplay(Cash, plot_type = 'partial', lag = 30) +
  labs(title = "ATM4 - Time Series, ACF, PACF")
```



Plotting the forecasted data

```
atm_fit <- atm_df %>%
  model(auto_arima = ARIMA(Cash, stepwise = FALSE))

atm_fc <- atm_fit %>%
  forecast(h = "30 days")

atm_fc %>%
  filter(.model == "auto_arima") %>%
  autoplot(atm_df) +
  labs(title = "30-Day Forecast using ARIMA",
       y = "Cash (hundreds)", x = "Date") +
  facet_wrap(~ATM, scales = "free_y")
```



Analysis- Ljung-Box Test:

- The p- value is > 0.05, that means ARIMA model works well for forecasting.

```
ljung_box_results <- atm_fit %>%
  select(auto_arima) %>%
  augment() %>%
  features(.innov, ljung_box, lag = 10, dof = 3)

ljung_box_results
```

```
## # A tibble: 4 x 4
##   ATM   .model     lb_stat lb_pvalue
##   <chr> <chr>        <dbl>     <dbl>
## 1 ATM1  auto_arima  12.0       0.100
## 2 ATM2  auto_arima   3.78      0.805
## 3 ATM3  auto_arima   0.132     1.00
## 4 ATM4  auto_arima   3.42      0.843
```

```
# Filter forecasts from May 1, 2010
atm_export <- atm_fc %>%
  filter(.model == "auto_arima", DATE >= as.Date("2010-05-01")) %>%
  as_tibble() %>%
  select(ATM, DATE, Forecast = .mean)
```

```
# Create workbook and worksheet
wb <- createWorkbook()
addWorksheet(wb, "ATM Forecasts")
writeData(wb, sheet = "ATM Forecasts", atm_export)

# Save the Excel file
saveWorkbook(wb, "ATM_PartA_30Day_ARIMA_Forecast.xlsx", overwrite = TRUE)
```

Conclusion for part A:

- In part A's analysis, I explored time series forecasting for four different ATMs using cash withdrawal data up to April 30, 2010.
- In the beginning, I prepossessing the data with identifying and imputing five missing values.
- To identify the best forecasting model, I fitted the four models(ETS, Auto ARIME, Naive, and Seasonal Naive) for each ATM. Then, I selected the ARIMA model for all the ATMs for forecast, because ARIMA has lower RMSE than other models.
- Finally, I use residual diagnostics and ljung- box for testing. And the model was satisfied.

Part B

```
power_data <- read.csv ('https://raw.githubusercontent.com/Jennyjjxxzz/Data-624_Project1/refs/heads/mai

head(power_data)
```

Load the data

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

- Convert dataframe to tsibble

```
power_data <- power_data %>%
  mutate(Date = yearmonth(YYYY.MMM)) %>%
  as_tsibble(index = Date)
```

```
# There is 1 missing value
sum(is.na(power_data))
```

Check for the missing values.

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

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

```
## # A tsibble: 1 x 4 [1M]
##   CaseSequence YYYY.MMM   KWH     Date
##          <int> <chr>    <int>    <mth>
## 1          861 2008-Sep    NA 2008 Sep
```

```r
# Impute the missing value with the mean KWH
power_data <- power_data %>%
  mutate(KWH = if_else(
    is.na(KWH),
    round(mean(KWH, na.rm = TRUE)),
    KWH
  ))
```

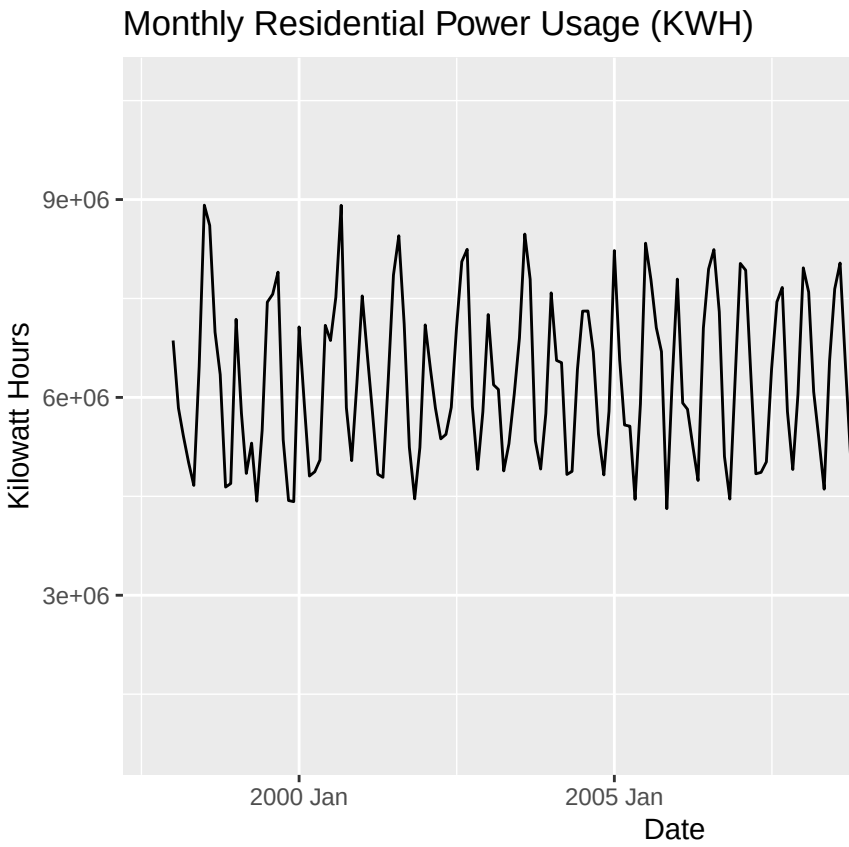
```r
# Check the missing value again
sum(is.na(power_data))
```

```
## [1] 0
```

```r
# double check the missing value in the row of 861
power_data %>%
  filter(CaseSequence==861)
```

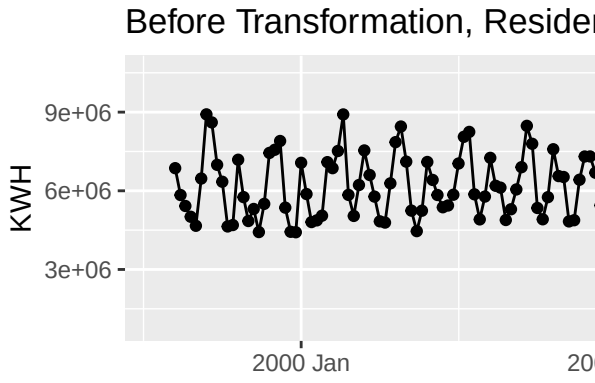
```
## # A tsibble: 1 x 4 [1M]
##   CaseSequence YYYY.MMM     KWH     Date
##          <int> <chr>      <dbl>    <mth>
## 1          861 2008-Sep 6502475 2008 Sep
```

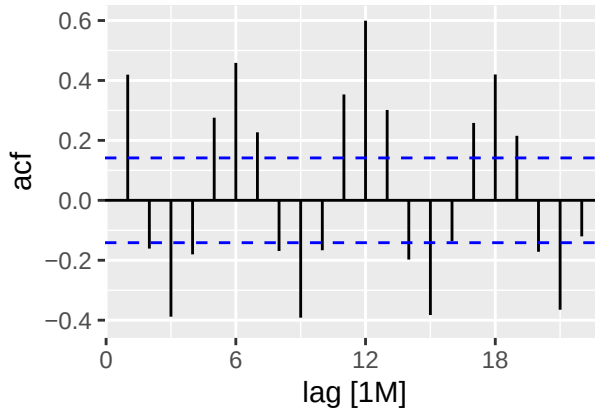
```r
power_data %>%
  autoplot(KWH) +
  labs(
    title = "Monthly Residential Power Usage (KWH)",
    y = "Kilowatt Hours", x = "Date"
  )
```



Plot the Monthly Residential Power Usage

```
power_data %>%
  gg_tsdisplay(KWH, plot_type='partial') +
  labs(title = "Before Transformation, Residential Power Usage")
```





Plot the Time series, ACF, and PACF(use difference() if need)
Fit and compare each model

- ARIMA model is the best model compare with others, because it has less RMSE.

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

power_fit_original <- power_data %>%
  model(
    ARIMA = ARIMA(KWH),
    ETS = ETS(KWH),
    SNaive = SNAIVE(KWH)
  )

accuracy_results_original <- power_fit_original %>%
  accuracy() %>%
  arrange(RMSE)

accuracy_results_original
```

```
## # A tibble: 3 x 10
##   .model .type         ME     RMSE     MAE   MPE  MAPE  MASE RMSSE   ACF1
##   <chr>  <chr>      <dbl>    <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>
## 1 ARIMA  Training -25069.  827744. 493541. -5.51  11.7 0.708 0.702 0.0128
## 2 ETS    Training  60461.  834738. 505667. -4.37  12.1 0.725 0.708 0.174
```

```
## 3 SNaive Training  94245. 1178792. 697453. -3.94  14.6 1     1     0.231
```

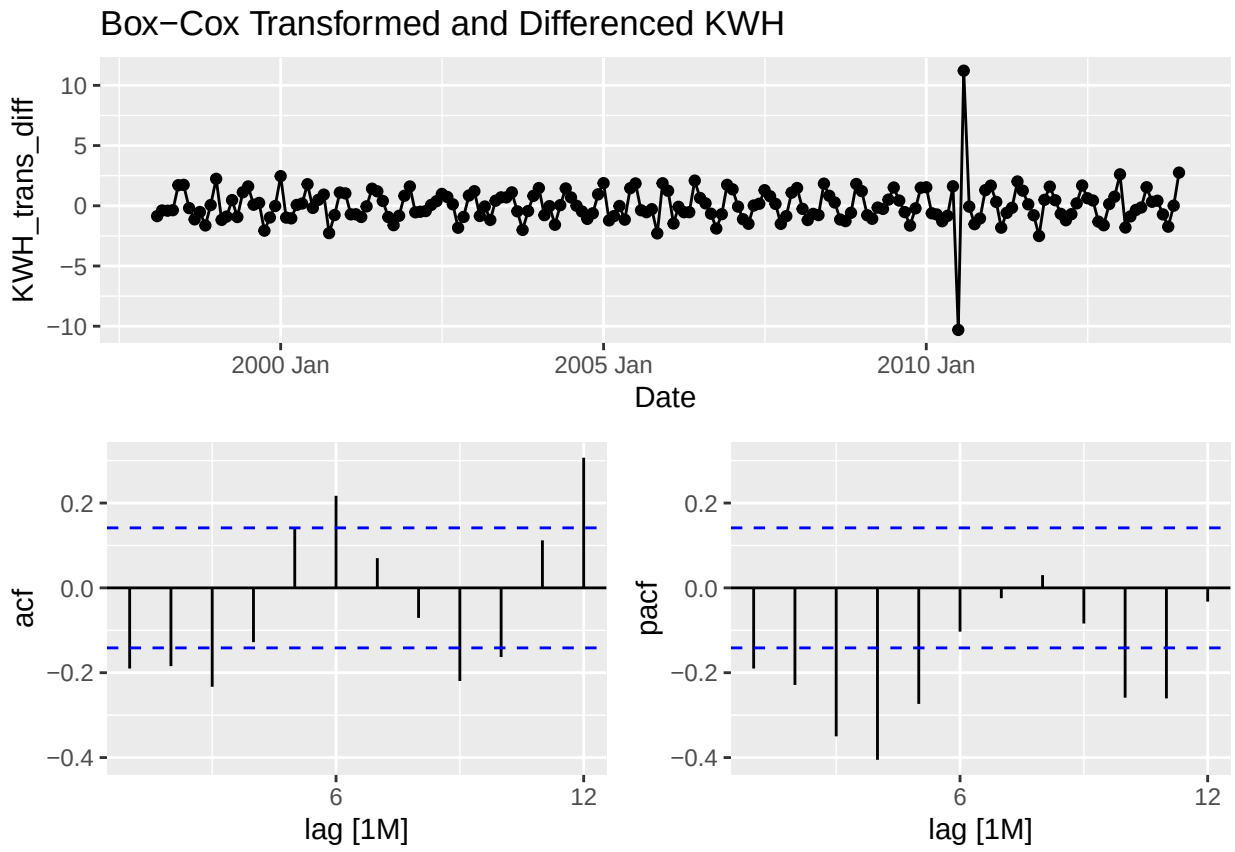
```
power_fit_summary <- power_fit_original %>%
  pivot_longer(everything(),
               names_to = "model",
               values_to = "order")

print(power_fit_summary)
```

```
## # A mable: 3 x 2
## # Key:     model [3]
##   model                                order
##   <chr>                               <model>
## 1 ARIMA  <ARIMA(0,0,1)(1,1,1)[12] w/ drift>
## 2 ETS                           <ETS(M,N,M)>
## 3 SNaive                            <SNAIVE>
```

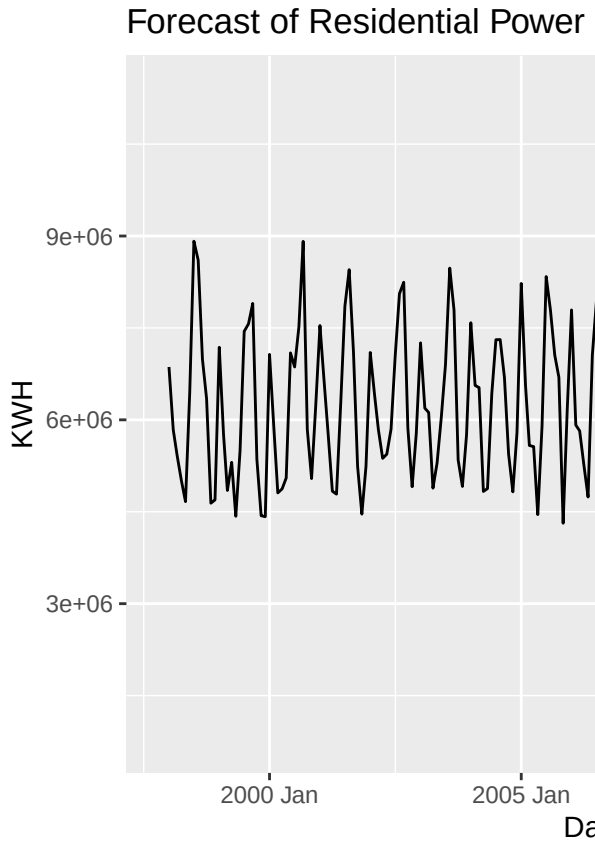
```
# Using Box-cox transfer the data
lambda <- power_data %>%
  features(KWH, features = guerrero) %>%
  pull(lambda_guerrero)

power_data <- power_data %>%
  mutate(KWH_trans_diff = difference(box_cox(KWH, lambda)))

power_data %>%
  gg_tsdisplay(KWH_trans_diff, plot_type = "partial", lag = 12) +
  ggtitle("Box-Cox Transformed and Differenced KWH")
```

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

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




```
power_arima_fit <- power_data %>%
  model(ARIMA = ARIMA(KWH))

power_arima_fc <- power_arima_fit %>%
  forecast(h = "12 months")

power_arima_fc %>%
  autoplot(power_data) +
  labs(
    title = "Forecast of Residential Power Usage",
    y = "KWH",
    x = "Date"
  )
```



Forecast of power usage by using ARIMA model for 12 months

```
report(power_arima_fit)
```

```
## Series: KWH
## Model: ARIMA(0,0,1)(1,1,1)[12] w/ drift
##
## Coefficients:
##          ma1     sar1     sma1   constant
##       0.2443  -0.1520  -0.7309  105249.87
## s.e.  0.0723   0.0963   0.0821   27038.86
##
## sigma^2 estimated as 7.474e+11:  log likelihood=-2720
## AIC=5450   AICc=5450.34   BIC=5465.96
```

- The model works well, the P-values are > 0.05

```
power_arima_fit %>%
  augment() %>%
  features(.innov, ljung_box, lag = 10, dof = 2)
```

```
## # A tibble: 1 x 3
##   .model lb_stat lb_pvalue
##   <chr>    <dbl>     <dbl>
## 1 ARIMA     7.28     0.507
```

```r
# convert to data frame
power_export <- power_arima_fc %>%
  filter(year(Date) == 2014) %>%
  as_tibble() %>%
  select(Date, Forecast = .mean)

# Create a new Excel workbook and add sheet
wb <- createWorkbook()
addWorksheet(wb, "Power Forecast 2014")
writeData(wb, sheet = "Power Forecast 2014", power_export)

# Save Excel file
saveWorkbook(wb, "PartB_Power_Forecast_2014.xlsx", overwrite = TRUE)
```

Conclusion for part B:

- In part B's analysis, I analyzed the monthly residential power data from January 1988 to December 2013, and forecasted the usage for the year of 2014. The original data show the data is non-stationary. To stabilize, I applied the Box-cox transformation. I evaluated the models, and ARIMA is the best model for this data. At the end, I used ljung- box for the test,and the model was satisfied.

Part C – BONUS

```r
pipe1 <- read.csv ('https://raw.githubusercontent.com/Jennyjjxxzz/Data-624_Project1/refs/heads/main/Wat

pipe2 <- read.csv ('https://raw.githubusercontent.com/Jennyjjxxzz/Data-624_Project1/refs/heads/main/Wat
```

```r
str(pipe1)
```

Load the datas

```
## 'data.frame':    1000 obs. of  2 variables:
##  $ Date.Time: chr  "10/23/15 12:24 AM" "10/23/15 12:40 AM" "10/23/15 12:53 AM" "10/23/15 12:55 AM" .
##  $ WaterFlow: num  23.4 28 23.1 30 6 ...
```

```r
str(pipe2)
```

```
## 'data.frame':    1000 obs. of  2 variables:
##  $ Date.Time: chr  "10/23/15 1:00 AM" "10/23/15 2:00 AM" "10/23/15 3:00 AM" "10/23/15 4:00 AM" ...
##  $ WaterFlow: num  18.8 43.1 38 36.1 31.9 ...
```

Aggregate Data

- Transform the Date.Time columns to actual date types.

```
pipe1 <- pipe1 %>%
  mutate(Date.Time = as.POSIXct(Date.Time, format = "%m/%d/%y %I:%M %p"))

pipe2 <- pipe2 %>%
  mutate(Date.Time = as.POSIXct(Date.Time, format = "%m/%d/%y %I:%M %p"))
```

```
pipe_df<- rbind(pipe1, pipe2)

head(pipe_df)
```

To combine the two data into one:

```
##             Date.Time WaterFlow
## 1 2015-10-23 00:24:00 23.369599
## 2 2015-10-23 00:40:00 28.002881
## 3 2015-10-23 00:53:00 23.065895
## 4 2015-10-23 00:55:00 29.972809
## 5 2015-10-23 01:19:00  5.997953
## 6 2015-10-23 01:23:00 15.935223
```

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

Check for missing values in data

```
## [1] Date.Time WaterFlow
## <0 rows> (or 0-length row.names)
```

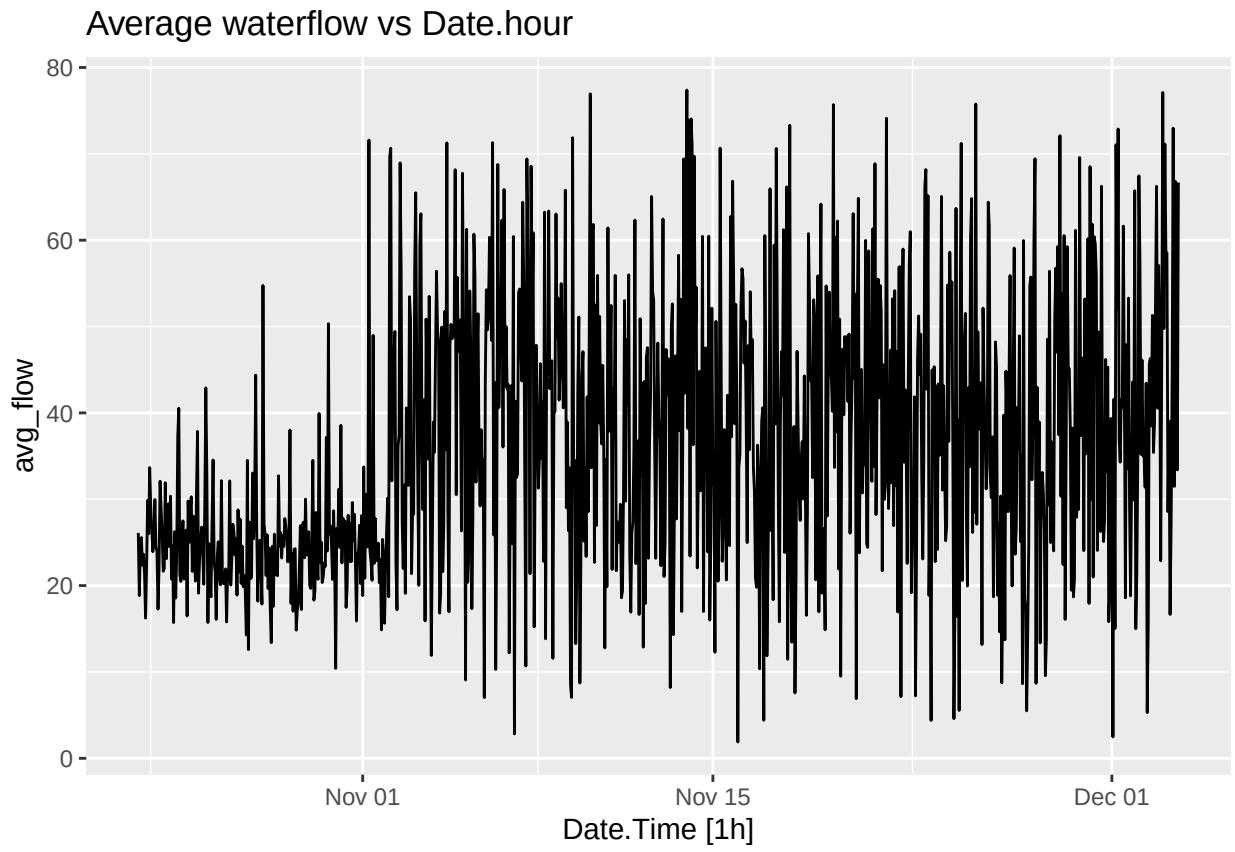
- Now take an average for each hour.

```
pipe_df <- pipe_df %>%
  mutate(Date.Time = floor_date(Date.Time, "hour")) %>%
  group_by(Date.Time) %>%
  summarize(avg_flow = mean(WaterFlow, na.rm=T)) %>%
  as_tsibble(index=Date.Time)

head(pipe_df)
```

```
## # A tsibble: 6 x 2 [1h] <?>
##   Date.Time           avg_flow
##   <dttm>                 <dbl>
## 1 2015-10-23 00:00:00     26.1
## 2 2015-10-23 01:00:00     18.8
## 3 2015-10-23 02:00:00     24.5
## 4 2015-10-23 03:00:00     25.6
## 5 2015-10-23 04:00:00     22.4
## 6 2015-10-23 05:00:00     23.6
```

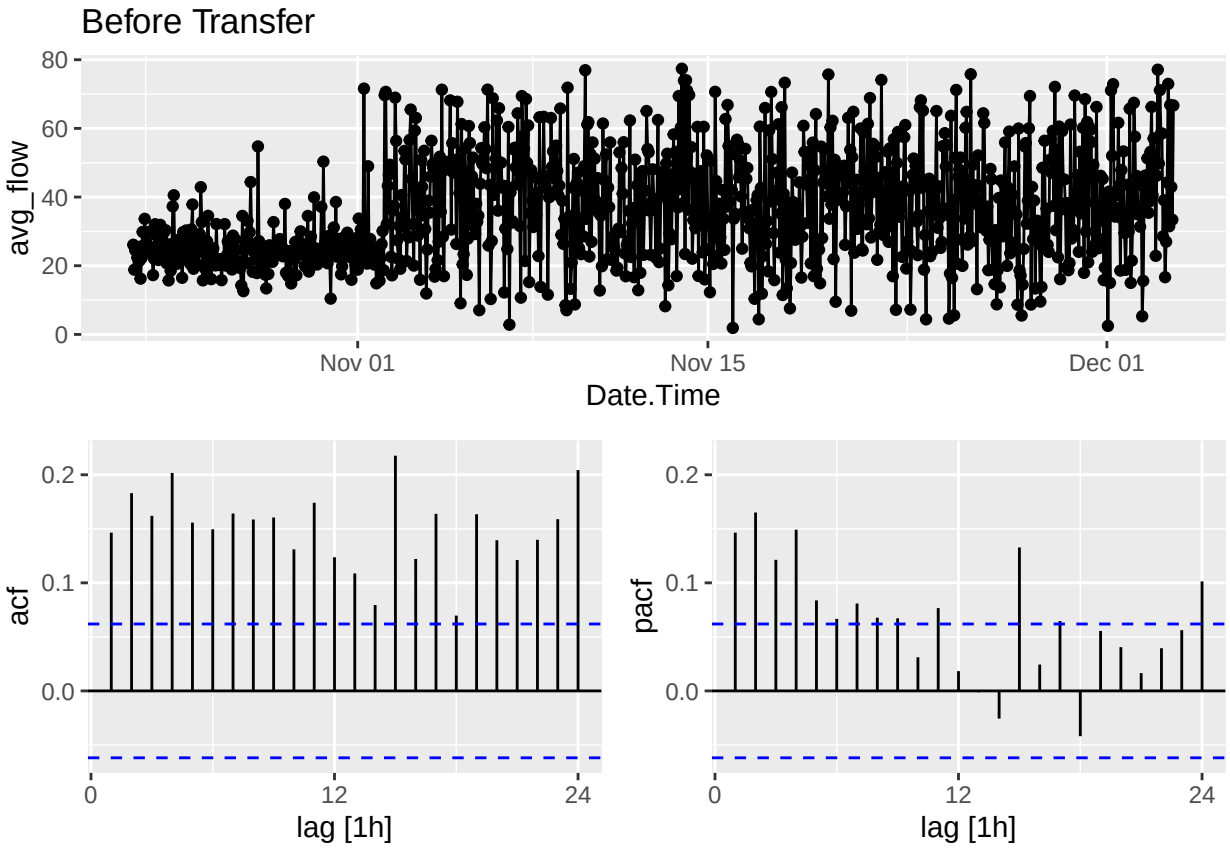
- Plot the data

```
pipe_df %>%
  autoplot(avg_flow) +
  ggtitle('Average waterflow vs Date.hour')
```



- We can see in ACF plots, the data is non-stationary, so we have to transfer

```
pipe_df %>%
  gg_tsdisplay(avg_flow, plot_type = 'partial', lag = 24)+ggtitle('Before Transfer')
```




```
#find lambad
lambad_pipe <- pipe_df %>%
  features(avg_flow, features = guerrero) %>%
  pull (lambda_guerrero)

#find ndiffs
pipe_df  %>%
  mutate(avg_flow = box_cox(avg_flow, lambad_pipe)) %>%
  features(avg_flow, unitroot_ndiffs)
```

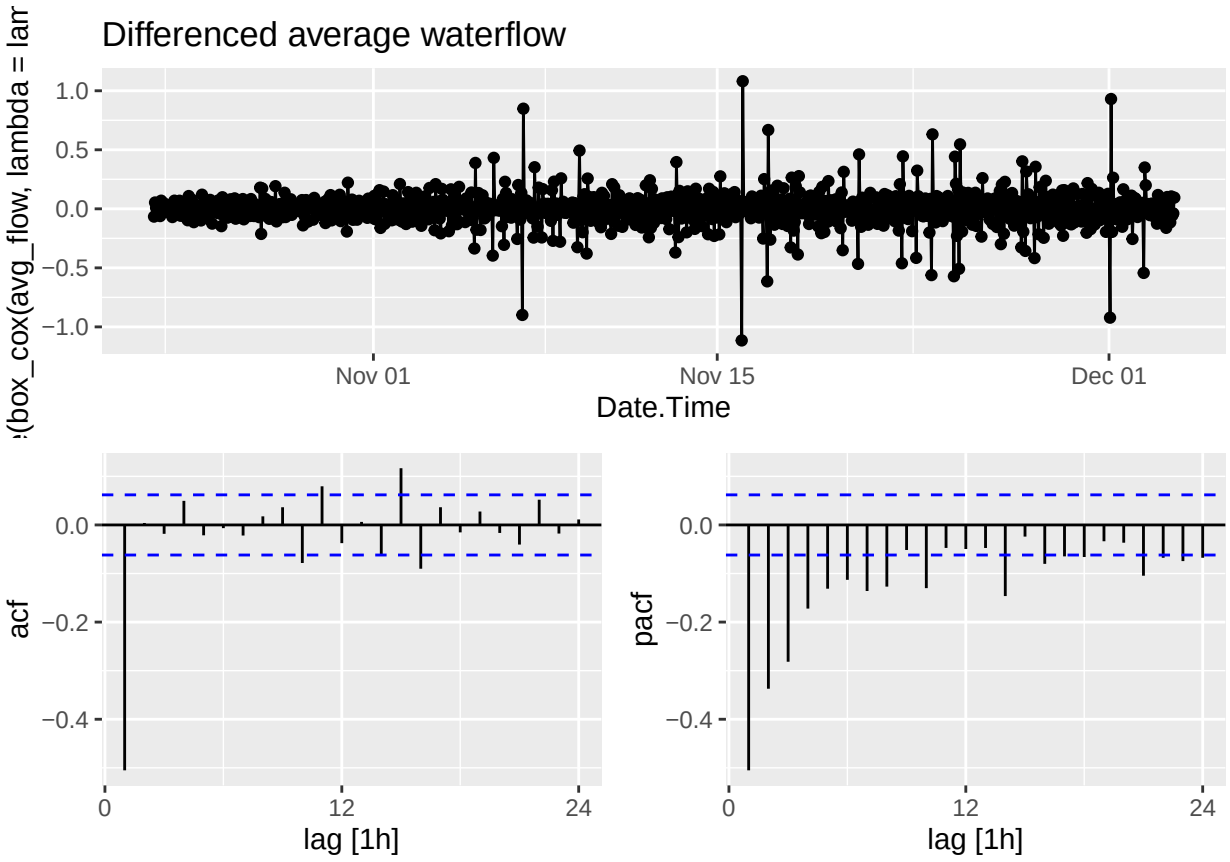
Transfer the data

```
## # A tibble: 1 x 1
##   ndiffs
##    <int>
## 1      1
```

```
pipe_df %>%
  gg_tsdisplay(difference(box_cox(avg_flow,lambda = lambad_pipe)), plot_type='partial', lag = 24) +
  labs(title = paste("Differenced average waterflow"))
```

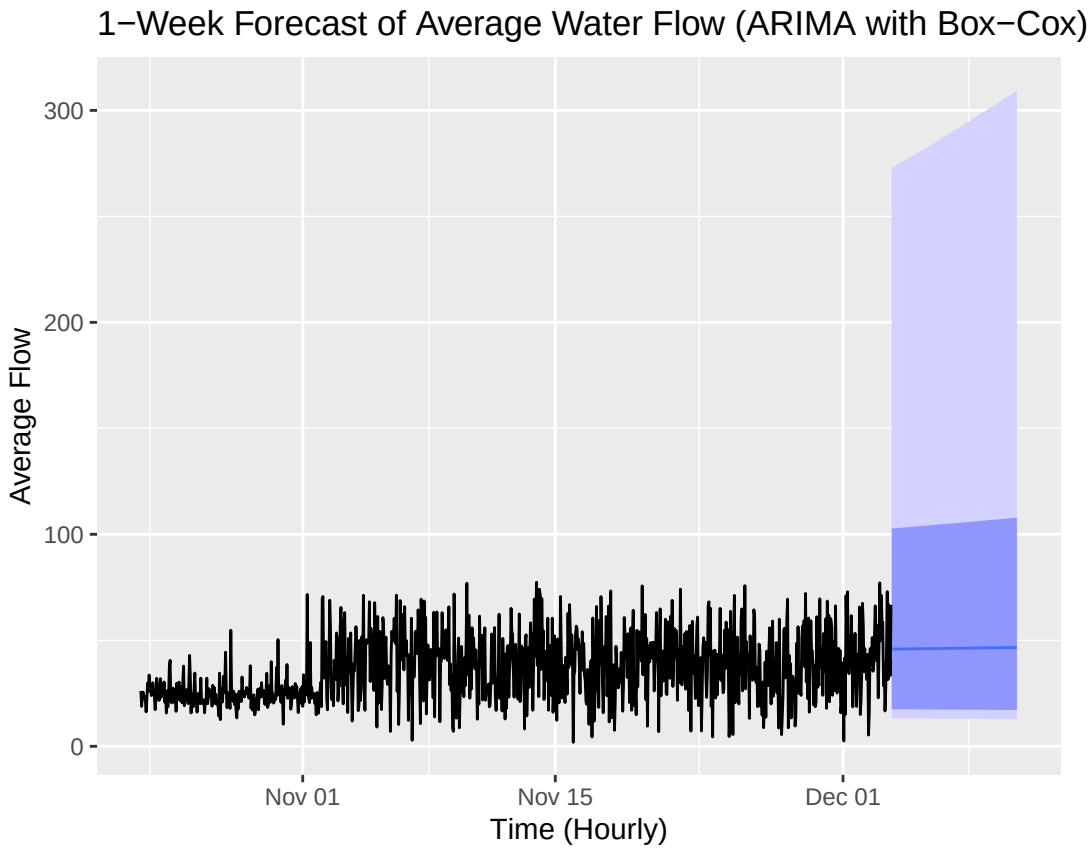
```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
```

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




```
pipe_fit_arima <- pipe_df %>%
  model(ARIMA = ARIMA(box_cox(avg_flow, lambda = lambad_pipe)))
```

```
pipe_forecast_arima <- pipe_fit_arima %>%
  forecast(h = "168 hours") # a week

pipe_forecast_arima %>%
  autoplot(pipe_df) +
  labs(
    title = "1-Week Forecast of Average Water Flow (ARIMA with Box-Cox)",
    x = "Time (Hourly)",
    y = "Average Flow"
  )
```



Forecast the data

- The p- value > 0.05

```
ljung_box_pipe <- pipe_fit_arima %>%
  augment() %>%
  features(.innov, ljung_box, lag = 24, dof = 2)

ljung_box_pipe
```

```
## # A tibble: 1 x 3
##   .model lb_stat lb_pvalue
##   <chr>    <dbl>     <dbl>
## 1 ARIMA     29.0     0.146
```

```
pipe_export <- pipe_forecast_arima %>%
  as_tibble() %>%
  select(DateTime = Date.Time, Forecast = .mean)

# Create a new workbook and write to Excel
wb <- createWorkbook()
addWorksheet(wb, "Pipe Forecast")
writeData(wb, sheet = "Pipe Forecast", pipe_export)

# Save the Excel file for Part C
saveWorkbook(wb, "PartC_Pipe1_1Week_Forecast.xlsx", overwrite = TRUE)
```

Conclusion for part C:

- In the part C, I worked with two data sets, and combined as one. They representing water flow rates at different timestaps for two different pipelines. The goal is to forecast waterflow one week ahead. In this data, the data is non- stationary, and I use box-cox to transfer the data. I selected the ARIMA model for final forecasting. Finally, I use Ljung- Box for test, and the result show the model ARIMA works well, the residual resembled white noise.