624 Project1

Chi Hang(Philip) Cheung

2025-03-11

```r
library(fpp3)
library(lubridate)
library(dplyr)
library(writexl)
library(imputeTS)
```

Part 1

Load the ATM file

```r
atm_path<- 'https://raw.githubusercontent.com/stormwhale/data-mines/refs/heads/main/ATM624Data%20(6).cs

atm <- read.csv(atm_path)
head(atm, 2)
```

```
##                   DATE  ATM Cash
## 1 5/1/2009 12:00:00 AM ATM1   96
## 2 5/1/2009 12:00:00 AM ATM2  107
```

format the data into tsibble format:

```r
#Converting the date into datetime format:
atm<- atm %>%
  mutate(DATE = as.POSIXct(DATE, format = '%m/%d/%Y %I:%M:%S %p'))

#format the data into tsibble format:
atm_ts<- atm %>%
  mutate(DATE = as.Date(DATE)) %>%
  as_tsibble(key = ATM, index=DATE) %>%
  arrange(DATE)

#check the data set:
head(atm_ts)
```

```
## # 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
```

To check for missing values other than 2010 May and days:

5 missing values as shown:

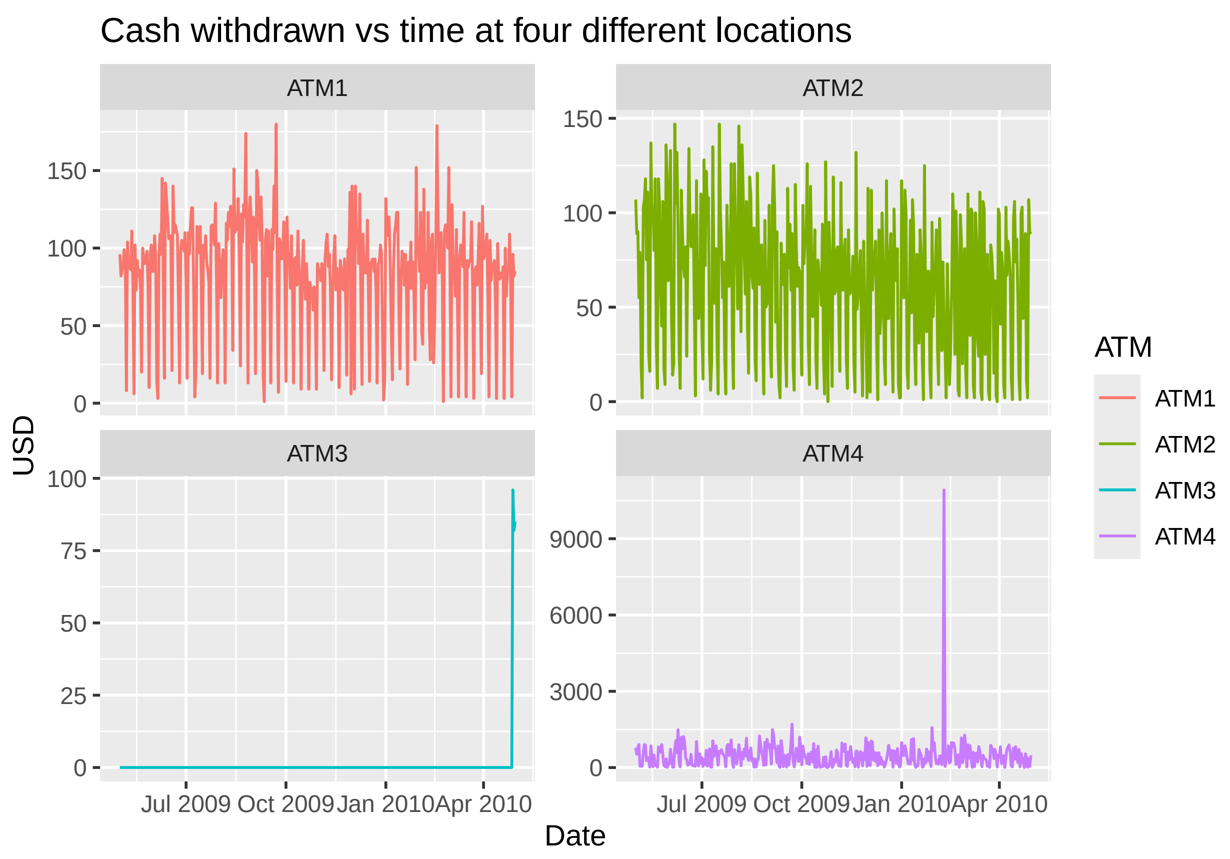
```
atm_ts %>%
  filter_index(~'2010-4-30') %>%
  filter(is.na(Cash))
```

```
## # 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
```

```
# 5 missing values in ATM1 and ATM2
```

To visualize the raw dataset

```
atm_ts %>%
  filter(!is.na(Cash)) %>%
  ggplot(aes(x=DATE, y = Cash, color = ATM))+
  geom_line()+
  facet_wrap(~ATM, scale='free_y')+
  labs(title = 'Cash withdrawn vs time at four different locations',
       x = 'Date', y = 'USD')
```



Need to fill the missing Cash values by imputing the mean value of that ATM location:

```
#Calculating the mean value for ATM1 and fill it into the missing values:
atm1_mean<- atm_ts %>%
  as_tibble() %>%
  filter(ATM=='ATM1') %>%
  summarize(mean = mean(Cash, na.rm=TRUE))

atm2_mean<- atm_ts %>%
  as_tibble() %>%
  filter(ATM=='ATM2') %>%
  summarize(mean = mean(Cash, na.rm=TRUE))

#To put the average Cash value into the missing spots using the imputeTS:
atm1<- atm_ts %>%
  filter(ATM=='ATM1') %>%
  na_mean()

atm2<- atm_ts %>%
  filter(ATM=='ATM2') %>%
  na_mean()

#defining ATM3 and ATM 4 for combining all data below:
atm3<- atm_ts %>%
  filter(ATM=='ATM3')

atm4<- atm_ts %>%
  filter(ATM=='ATM4')
```

Recombine all the data back into a tidy format:

```
atm_comb<- bind_rows(atm1,atm2,atm3,atm4)
#Ensuring the dataframe is correctly combined:
atm_comb %>%
  group_by(ATM) %>%
  count()
```

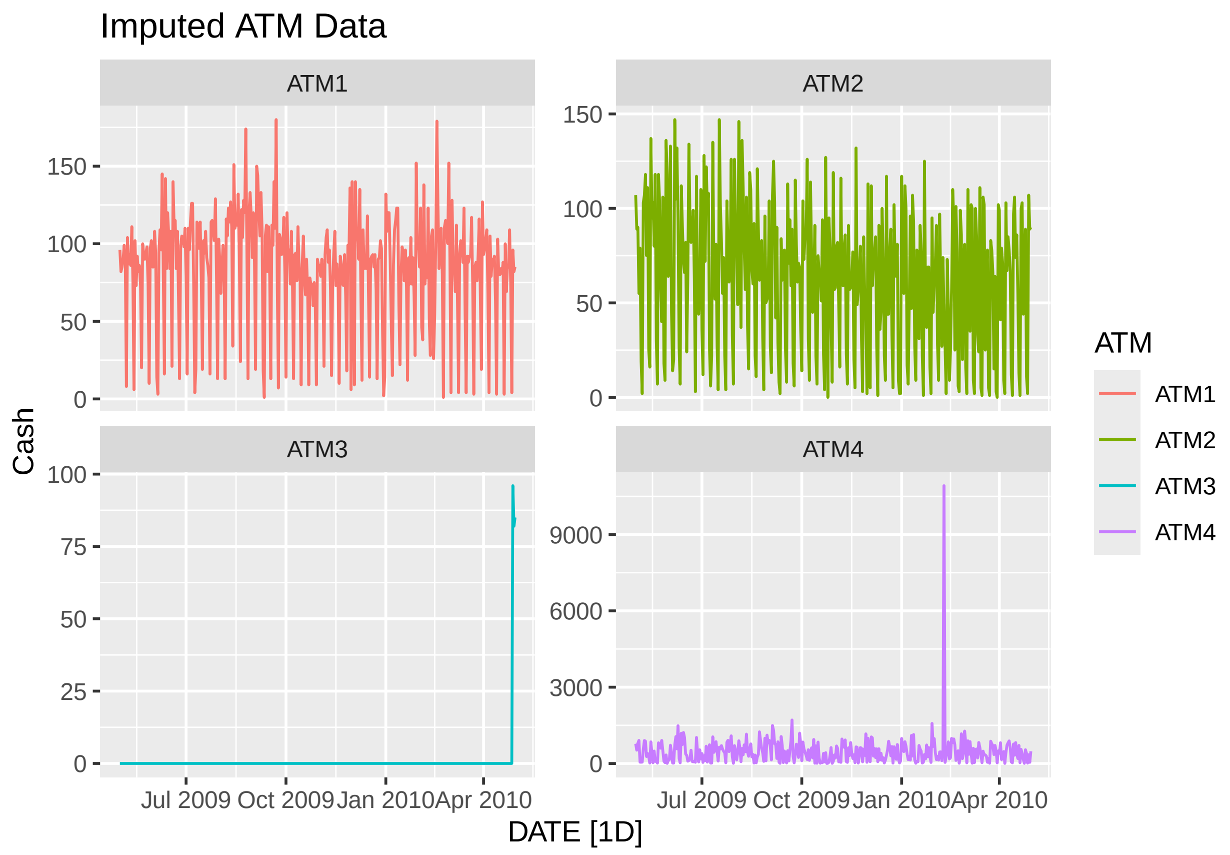
```
## # A tibble: 4 x 2
## # Groups:   ATM [4]
##   ATM       n
##   <chr> <int>
## 1 ATM1    365
## 2 ATM2    365
## 3 ATM3    365
## 4 ATM4    365
```

```
#Checking for any additional missing values:
any(sapply(atm_comb,is.na)) #False is returned and all values are in place
```

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

To visualize the imputed data:

```
atm_comb %>%
  autoplot(Cash) +
  facet_wrap(~ATM, scale='free_y') +
  ggtitle('Imputed ATM Data')
```



Fitting each location into models:

```
set.seed(123)
#ATM location 1,2,and 4 will have SNAIVE as benchmark as seasonality pattern is observed:
atm_fit124<- atm_comb %>%
  filter(!ATM=='ATM3') %>%
  model(ETS(Cash),
        auto_arima = ARIMA(Cash, stepwise=FALSE),
        Snaive_benchmark = SNAIVE(Cash))

#ATM location 3 will have NAIVE model as benchmark due to the simplicity of the data:
atm_fit3<- atm_comb %>%
  filter(ATM=='ATM3') %>%
  model(ETS(Cash),
        auto_arima = ARIMA(Cash, stepwise=FALSE),
        Naive_benchmark = NAIVE(Cash))

#create forecast for ATM location 1, 2, 4 for 30 days in May 2010
atm_fc124<- atm_fit124 %>%
  forecast(h = '30 day')
#create forecast for ATM location 3 for 30 days in May 2010
atm_fc3<- atm_fit3 %>%
  forecast(h = '30 day')
```

```
# recombine the data into one forecast dataframe:
atm_fc<- bind_rows(atm_fc124,atm_fc3)
```

Compare accuracy:

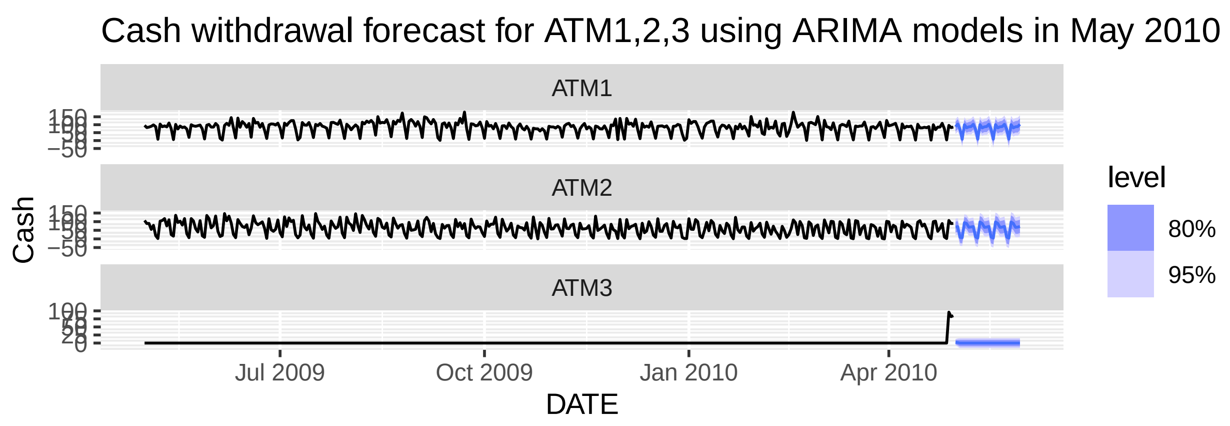
```


bind_rows(atm_fit124 %>% accuracy(),
          atm_fit3 %>% accuracy()) %>%
  select(-ME,-MPE, -ACF1) %>%
  arrange(ATM, desc=FALSE)
```

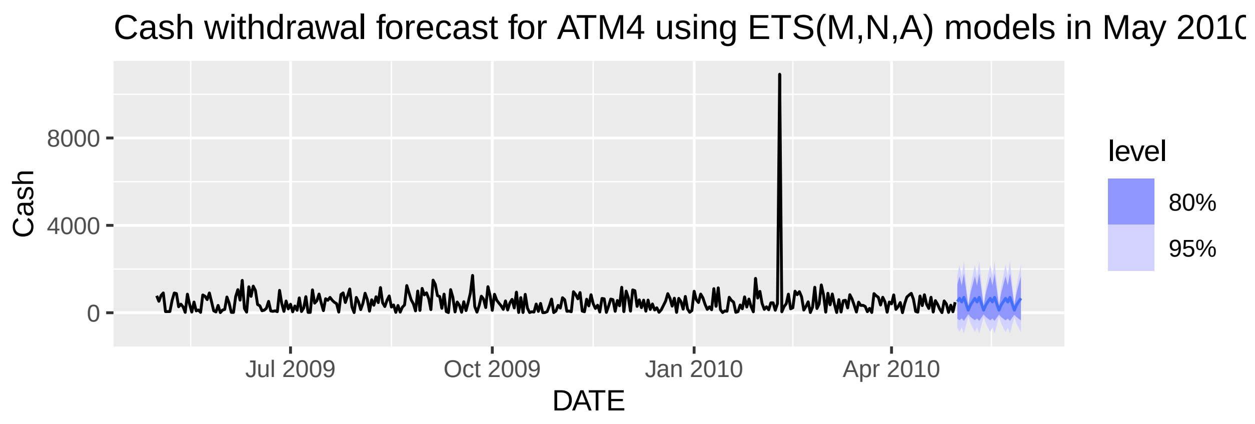
ARIMA model out performs both the benchmark and the ETS models in ATM1, ATM2, ATM3 EXCEPT in ATM4, where ETS model is better as shown by the RMSE below:

```
## # A tibble: 12 x 8
##    ATM   .model            .type     RMSE    MAE  MAPE  MASE RMSSE
##    <chr> <chr>             <chr>    <dbl>  <dbl> <dbl> <dbl> <dbl>
##  1 ATM1  ETS(Cash)         Training  23.8  15.1  121.  0.846 0.854
##  2 ATM1  auto_arima        Training  23.4  14.7  118.  0.825 0.840
##  3 ATM1  Snaive_benchmark  Training  27.9  17.8  116.  1     1
##  4 ATM2  ETS(Cash)         Training  25.0  17.8  Inf   0.856 0.832
##  5 ATM2  auto_arima        Training  24.1  17.3  Inf   0.835 0.803
##  6 ATM2  Snaive_benchmark  Training  30.1  20.8  Inf   1     1
##  7 ATM3  ETS(Cash)         Training   5.03  0.273 Inf  0.371 0.625
##  8 ATM3  auto_arima        Training   5.03  0.271  34.6 0.370 0.625
##  9 ATM3  Naive_benchmark   Training   5.09  0.310  40.2 0.423 0.632
## 10 ATM4  ETS(Cash)         Training 647.   310.    588.  0.770 0.722
## 11 ATM4  auto_arima        Training 650.   324.    620.  0.805 0.725
## 12 ATM4  Snaive_benchmark  Training 896.   402.    441.  1     1
```

Plotting the forecasted data:

```
atm123<- atm_fc %>%
  filter(ATM!= 'ATM4', .model=='auto_arima') %>%
  autoplot(atm_comb)+
  ggtitle('Cash withdrawal forecast for ATM1,2,3 using ARIMA models in May 2010')+
  guides(color= guide_legend(title='.model'))

atm4<- atm_fc %>%
  filter(ATM== 'ATM4', .model=='ETS(Cash)') %>%
  autoplot(atm_comb)+
  ggtitle('Cash withdrawal forecast for ATM4 using ETS(M,N,A) models in May 2010')+
  guides(color= guide_legend(title='.model'))

gridExtra::grid.arrange(atm123, atm4, nrow=2)
```





To check the ARIMA models selected:

```
atm_fit124_long<-atm_fit124 %>%
  pivot_longer(col=-ATM,
               values_to = 'order',
               names_to = 'model')
atm_fit3_long<- atm_fit3 %>%
  pivot_longer(col=-ATM,
               values_to = 'order',
               names_to = 'model')

#combine the two dataframes:
atm_fit_combo<- bind_rows(atm_fit124_long,atm_fit3_long)

#for ATM1,2,3 ARIMA model
atm_fit_combo%>%
  filter(model=='auto_arima', ATM!='ATM4') #ARIMA(001)(012); ARIMA(500)(011); ARIMA(002), respectively
```

```
## # A mable: 3 x 3
## # Key:     ATM, model [3]
##   ATM   model                           order
##   <chr> <chr>                         <model>
## 1 ATM1  auto_arima <ARIMA(0,0,1)(0,1,2)[7]>
## 2 ATM2  auto_arima <ARIMA(5,0,0)(0,1,1)[7]>
## 3 ATM3  auto_arima           <ARIMA(0,0,2)>
```

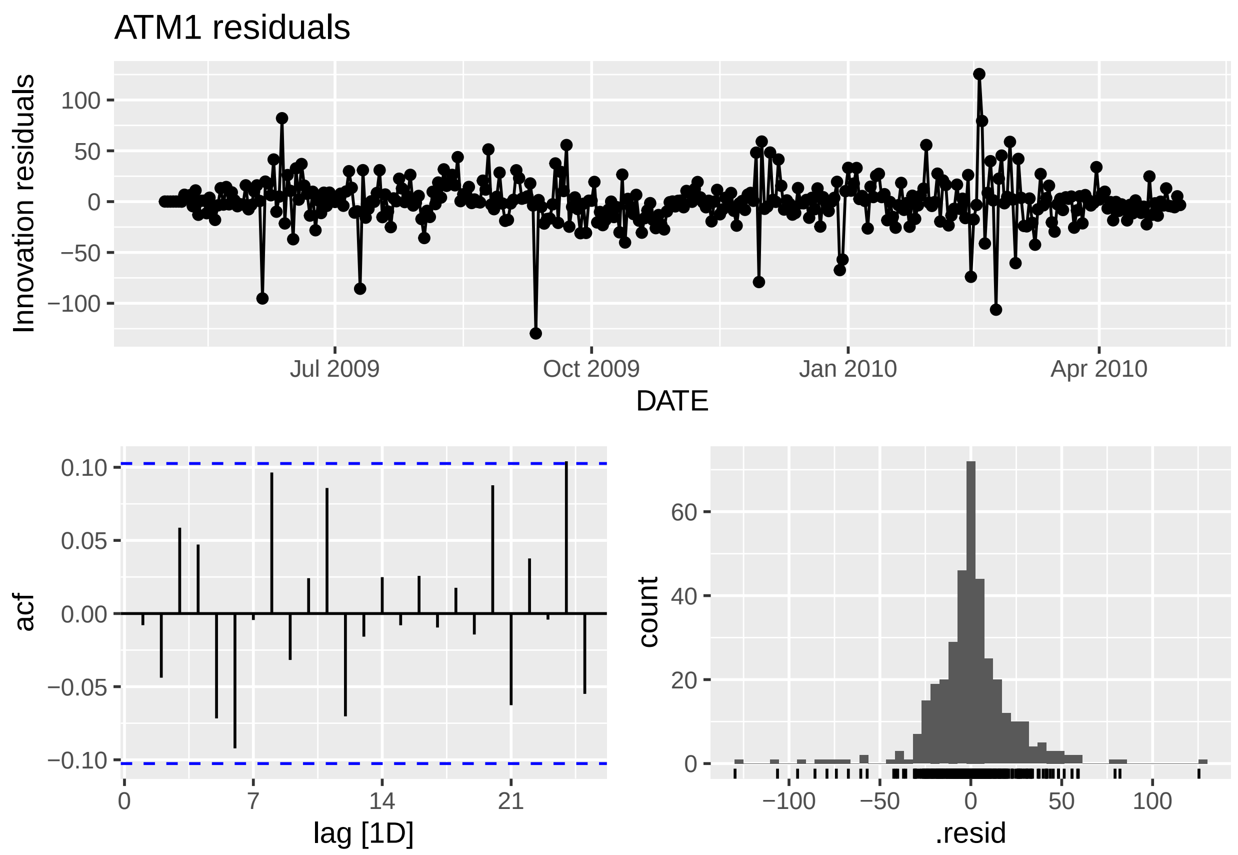
```
#for ATM4 ETS model:
atm_fit_combo%>%
  filter((ATM=='ATM4' & model =='ETS(Cash)')) #ETS(M,N,A)
```

```
## # A mable: 1 x 3
## # Key:     ATM, model [1]
##   ATM   model           order
##   <chr> <chr>         <model>
## 1 ATM4  ETS(Cash) <ETS(M,N,A)>
```

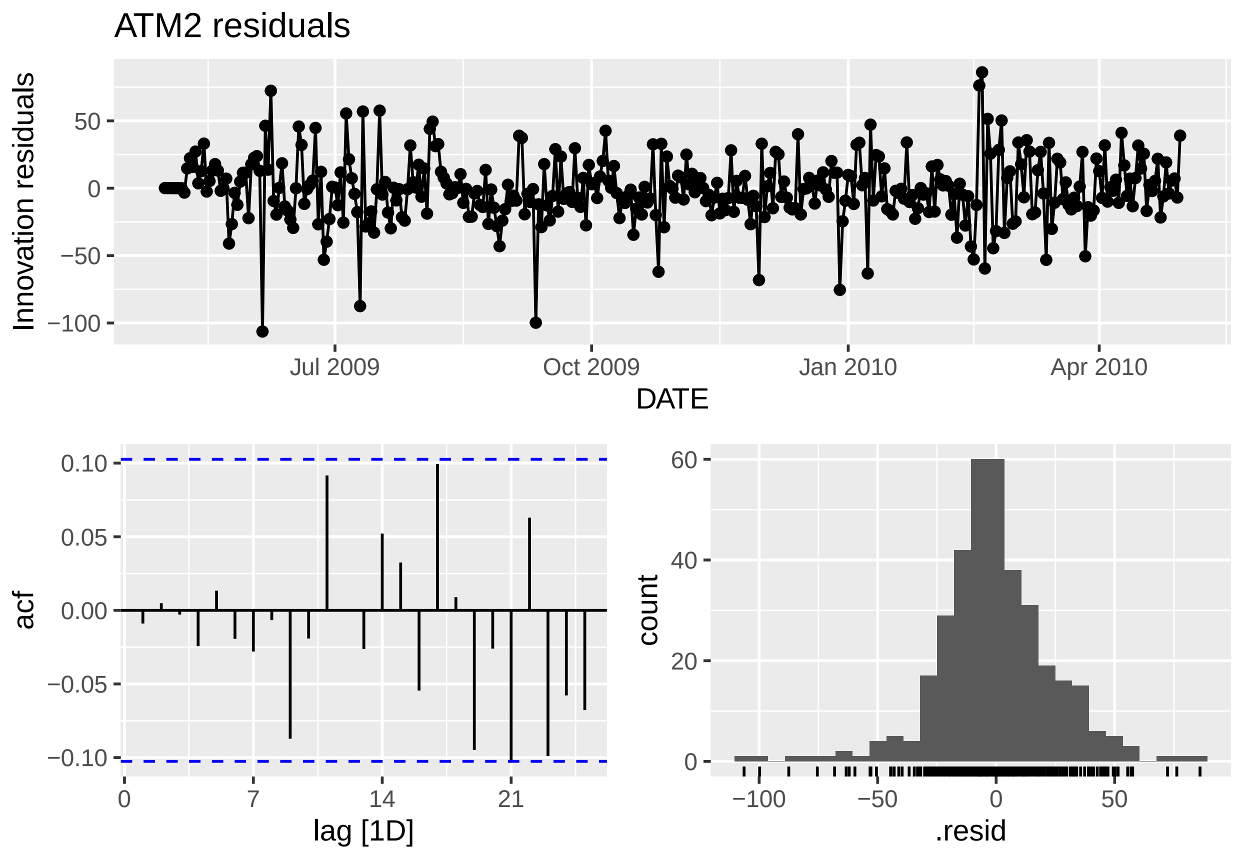
Diagnostic analysis-Residual Plots:

Residual plots

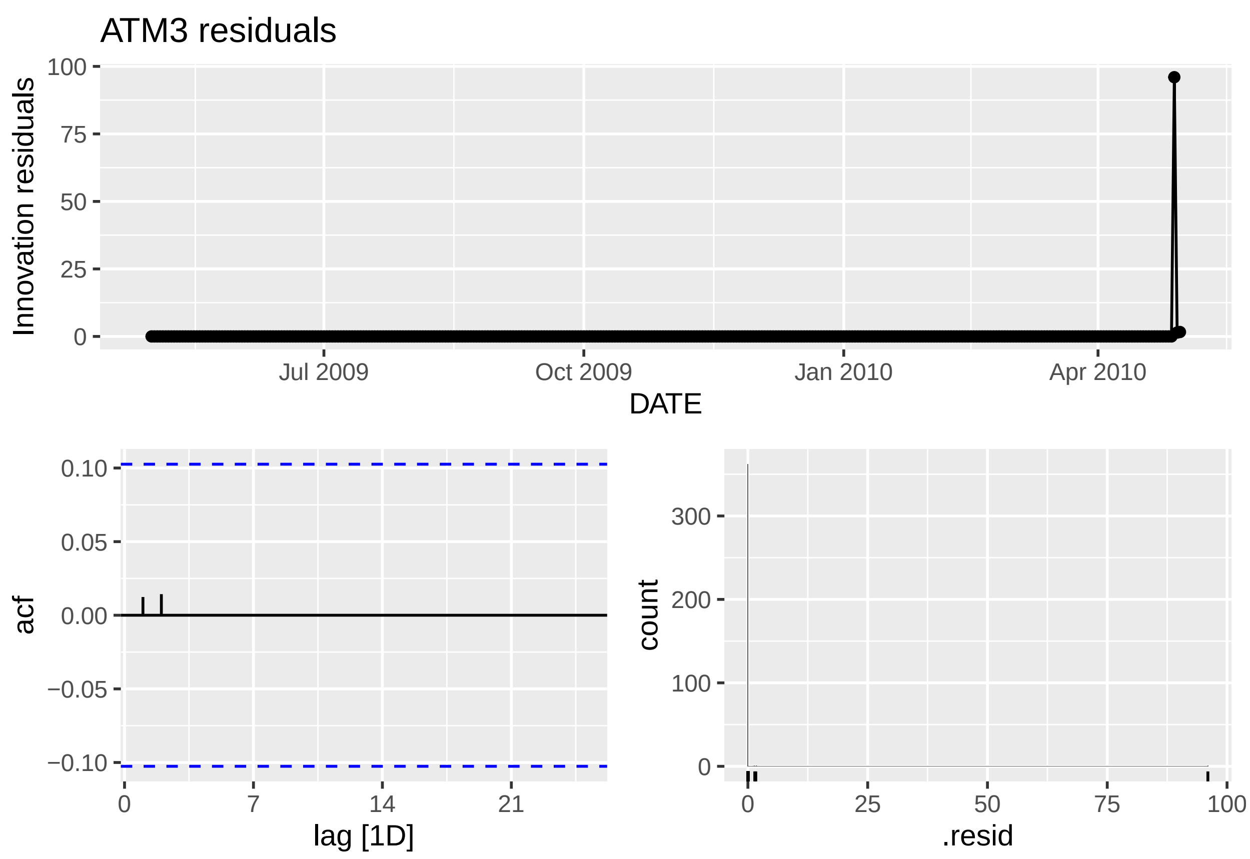
```
#checking for residual plots and ljung_box test for white-noise:
#ATM1
atm_fit124 %>%
  filter(ATM=='ATM1') %>%
  select(auto_arima) %>%
  gg_tsresiduals() + ggtitle('ATM1 residuals')
```



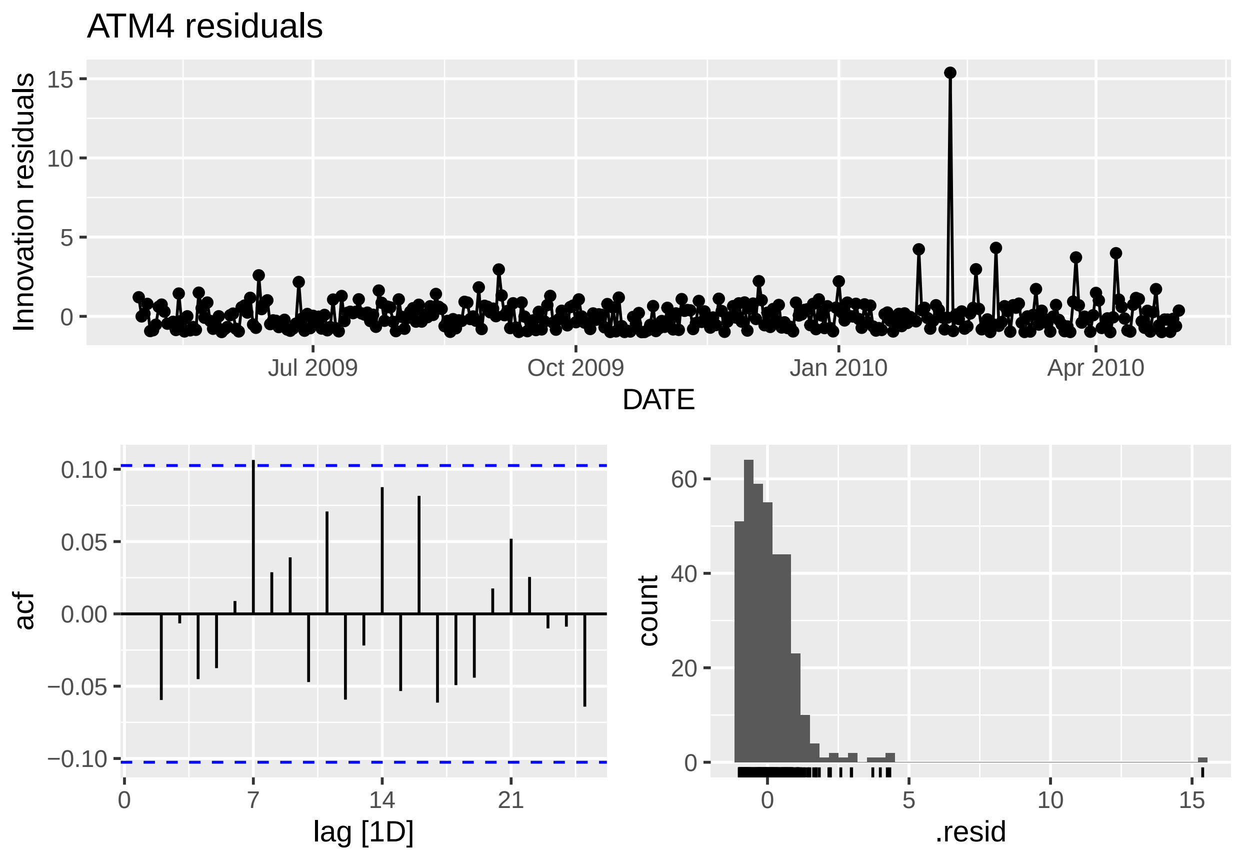
```r
#ATM2
atm_fit124 %>%
  filter(ATM=='ATM2') %>%
  select(auto_arima) %>%
  gg_tsresiduals() + ggtitle('ATM2 residuals')
```



```r
#ATM3
atm_fit3 %>%
  select(auto_arima) %>%
  gg_tsresiduals() + ggtitle('ATM3 residuals')
```



```
#ATM4
atm_fit124 %>%
  filter(ATM=='ATM4') %>%
  select(`ETS(Cash)`) %>%
  gg_tsresiduals() + ggtitle('ATM4 residuals')
```



Diagnostic analysis- Ljung-Box Test:

All the models passed with p-value >0.05

```
#ATM1 model:
atm_fit124 %>%
  filter(ATM=='ATM1') %>%
  select(auto_arima) %>%
  augment() %>%
  features(.innov, ljung_box, lag = 10, dof = 3)
```

```
## # A tibble: 1 x 3
##   .model     lb_stat lb_pvalue
##   <chr>        <dbl>     <dbl>
## 1 auto_arima    12.0     0.100
```

```
#ATM2 model:
atm_fit124 %>%
  filter(ATM=='ATM2') %>%
  select(auto_arima) %>%
  augment() %>%
  features(.innov, ljung_box, lag = 10, dof = 6)
```

```
## # A tibble: 1 x 3
##   .model     lb_stat lb_pvalue
##   <chr>        <dbl>     <dbl>
## 1 auto_arima    3.78     0.437
```

```r
#ATM3 model:
atm_fit3 %>%
  select(auto_arima) %>%
  augment() %>%
  features(.innov, ljung_box, lag = 10, dof = 2)
```

```
## # A tibble: 1 x 3
##   .model     lb_stat lb_pvalue
##   <chr>        <dbl>     <dbl>
## 1 auto_arima   0.132      1.00
```

```r
#ATM4 model:
atm_fit124 %>%
  filter(ATM=='ATM4') %>%
  select(`ETS(Cash)`) %>%
  augment() %>%
  features(.innov, ljung_box, lag = 10)
```

```
## # A tibble: 1 x 3
##   .model    lb_stat lb_pvalue
##   <chr>       <dbl>     <dbl>
## 1 ETS(Cash)    8.59     0.571
```

Result output to an excel sheet:

```r
#exporting ATM 1 - 3 and ATM 4 will be in a
atm_output123<- atm_fc %>%
  filter(.model == 'auto_arima', ATM!='ATM4')
atm_output4<- atm_fc %>%
  filter(.model == 'ETS(Cash)', ATM=='ATM4')

#combining the fables and convert them into dataframe for exporting:
atm_output_combo <- bind_rows(atm_output123, atm_output4) %>%
  as.data.frame() %>%
  select(ATM, DATE, .mean) %>%
  rename(Cash = .mean)
#To export:
write_xlsx(atm_output_combo, 'partA_atm_output_combo.xlsx')
```

Conclusion for part A:

Due to seasonality patterns, the SNAIVE method was selected as the benchmark model for ATM1, ATM2, ATM4 to compare whether ARIMA or ETS model can out-perform it in RMSE measurements. The auto ARIMA model selected for ATM 1, 2, and 3 were ARIMA(0,0,1)(0,1,2), ARIMA(5,0,0)(0,1,1), and ARIMA(0,0,2), respectively. An ETS model of (M,N,A) was used to model ATM4 since it had the lowest RSME value in the comparison. These models were selected automatically based on the AICc values. All the models except ATM3 has some degree of seasonality as reflected on the seasonal parameters in the selected models. The residuals of all four of the ATM models were checked and tested for white-noise. All four fitted models have a p_value > 0.05 from the Ljung-box test, indicating the residuals are white-noise and the models perform fairly well in capturing all the data.

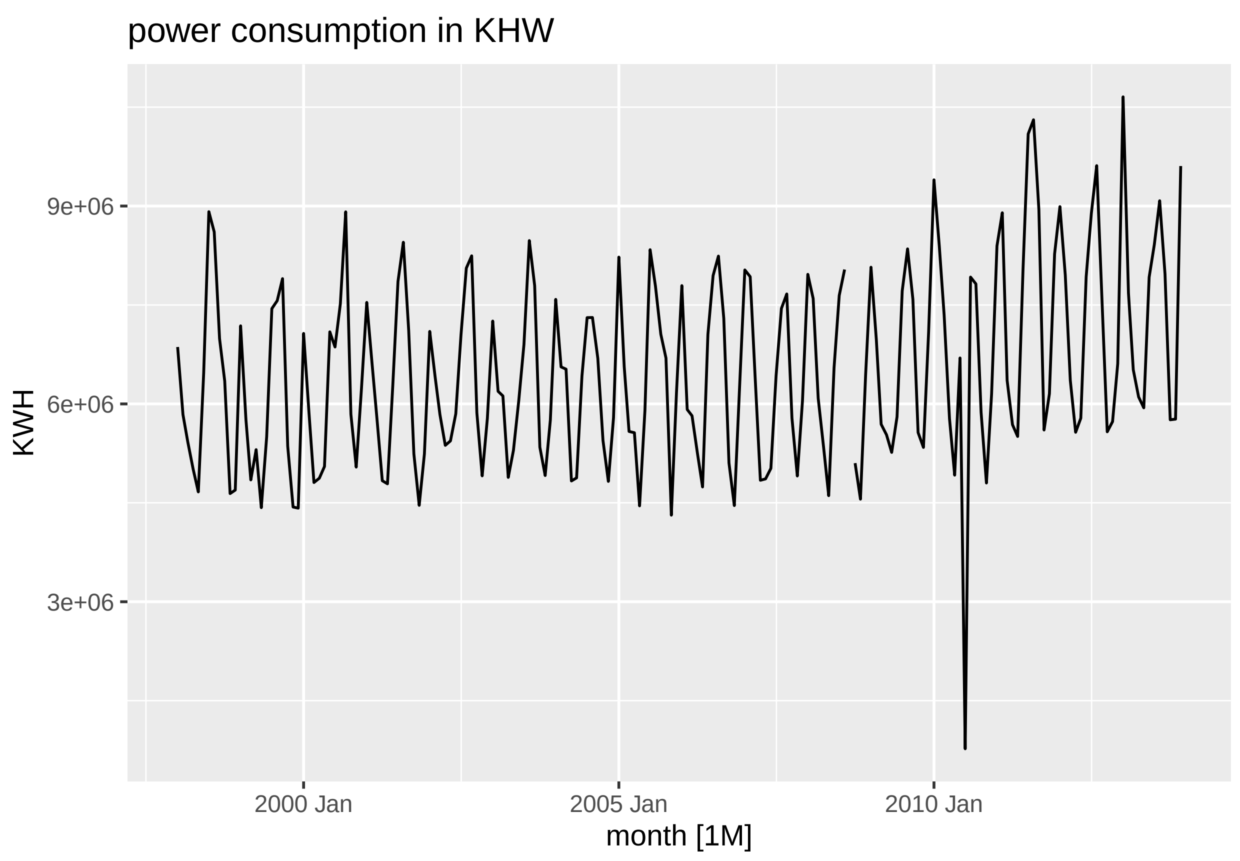
Part B

Monthly forecast of power usage in 2014: Loading data:

```
url2<- 'https://raw.githubusercontent.com/stormwhale/data-mines/refs/heads/main/ResidentialCustomerFore
power<- read.csv(url2)

#checking column types:
str(power)
```

```
## 'data.frame':    192 obs. of  3 variables:
##  $ CaseSequence: int  733 734 735 736 737 738 739 740 741 742 ...
##  $ YYYY.MMM    : chr  "1998-Jan" "1998-Feb" "1998-Mar" "1998-Apr" ...
##  $ KWH         : int  6862583 5838198 5420658 5010364 4665377 6467147 8914755 8607428 6989888 634562
```

```
#convert dataframe to tsibble:
power<- power %>%
  mutate(month = yearmonth(YYYY.MMM)) %>%
  select(-YYYY.MMM) %>%
  as_tsibble(index = month) %>%
  arrange(month, desc=TRUE)

#To visualize
power %>% autoplot(KWH)+ggtitle('power consumption in KHW')
```



To check for missing values:

```
power %>%
  filter(is.na(KWH)) # 1 missing value from CaseSequence 861
```

```
## # A tsibble: 1 x 3 [1M]
##   CaseSequence   KWH    month
##          <int> <int>    <mth>
## 1          861    NA 2008 Sep
```

```
#We will impute the missing value with the mean KWH from the dataset:
power<- power %>%
  mutate(KWH = if_else(is.na(KWH), mean(KWH, na.rm=TRUE), KWH))

#Double check the imputed value:
power %>%
  filter(CaseSequence==861)
```

```
## # A tsibble: 1 x 3 [1M]
##   CaseSequence      KWH    month
##          <int>    <dbl>    <mth>
## 1          861 6502475. 2008 Sep
```

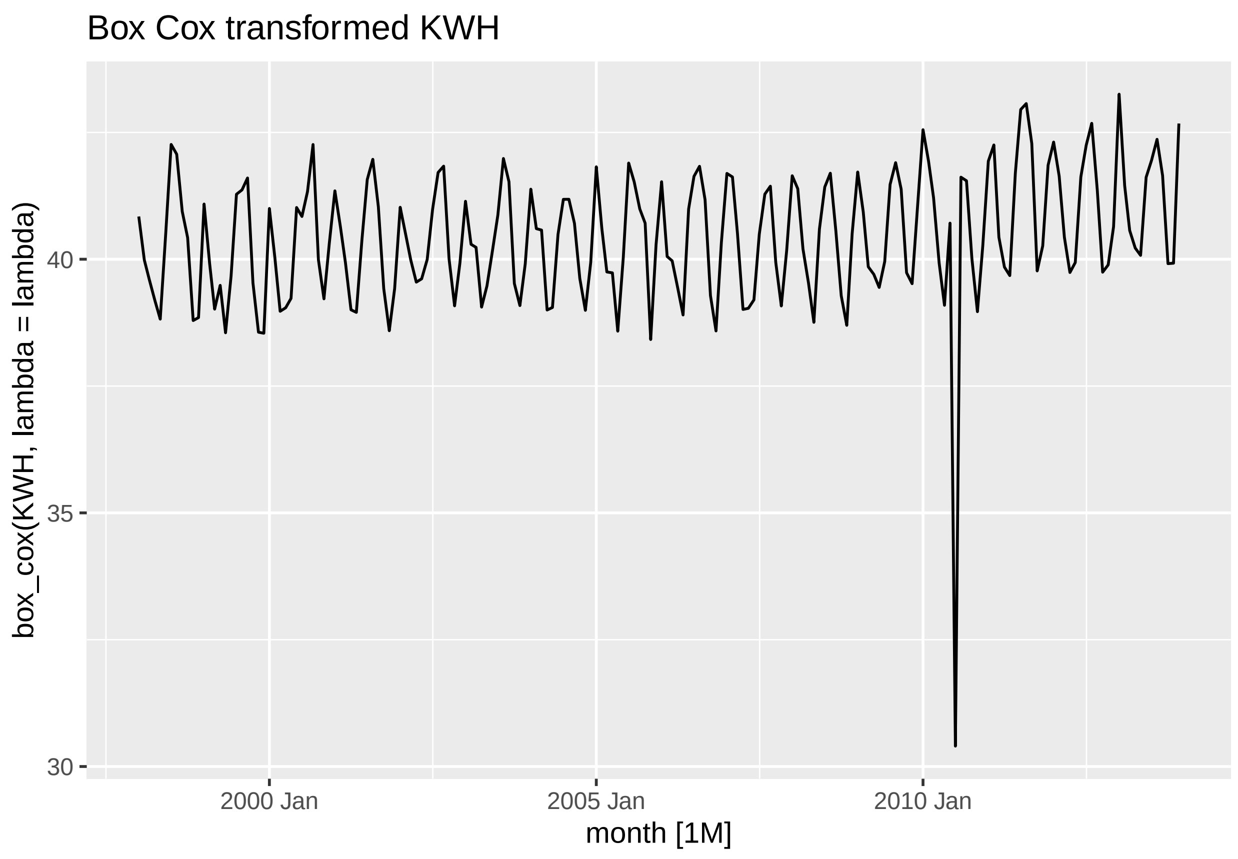
```r
#checking for missing values:
any(is.na(power)) #False
```

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

Possible transformations on the dataset:

The transformation might not help with the outlier as it amplified the outlier's effect.

```r
#Trying out box_cox transformation to stabilize the variance:
lambda<- power %>%
  features(KWH, box_cox, feature=guerrero) %>%
  pull(lambda_guerrero)

#plotting the transformed data:
power %>%
  autoplot(box_cox(KWH, lambda = lambda)) +
  ggtitle('Box Cox transformed KWH')
```



To fit the data without the transformation:

```r
#Fitting different models:
power_fit<- power %>%
  model(
    ETS = ETS(KWH),
    auto_arima = ARIMA(KWH, stepwise = FALSE),
    snaive = SNAIVE(KWH)
  )
```

To check the models selected automatically:

```r
power_fit_long<- power_fit %>%
  pivot_longer(everything(),
               values_to= 'order',
               names_to = 'model')
print(power_fit_long)
```

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

To compare the performance of each model against each other:

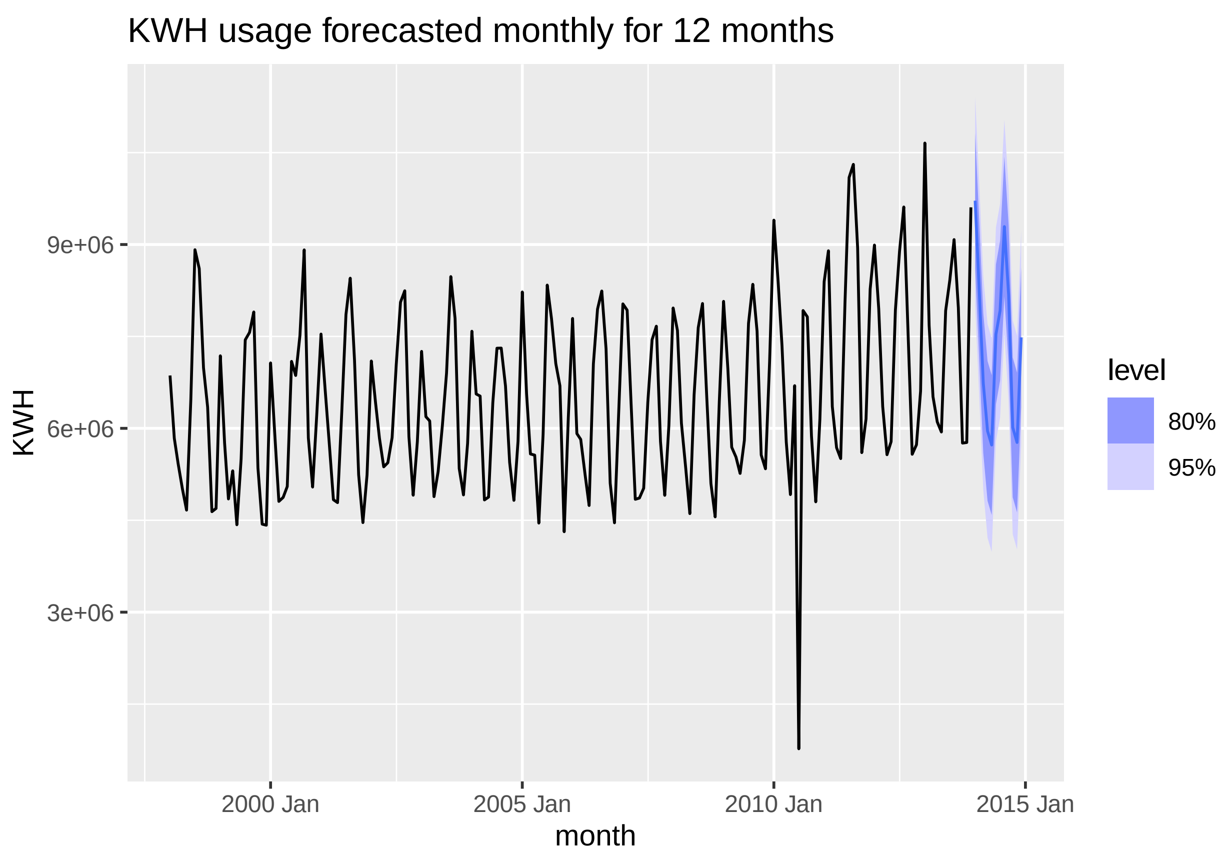
auto_arima out performs the snaive benchmark and the ETS(M,N,M) model

```r
accuracy(power_fit)
```

```
## # 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 ETS        Training  60461.  834738. 505667. -4.37  12.1 0.725 0.708 0.174
## 2 auto_arima Training -25069.  827744. 493541. -5.51  11.7 0.708 0.702 0.0128
## 3 snaive     Training  94245. 1178792. 697453. -3.94  14.6 1     1     0.231
```

The auto_arima model is selected for the forecast:

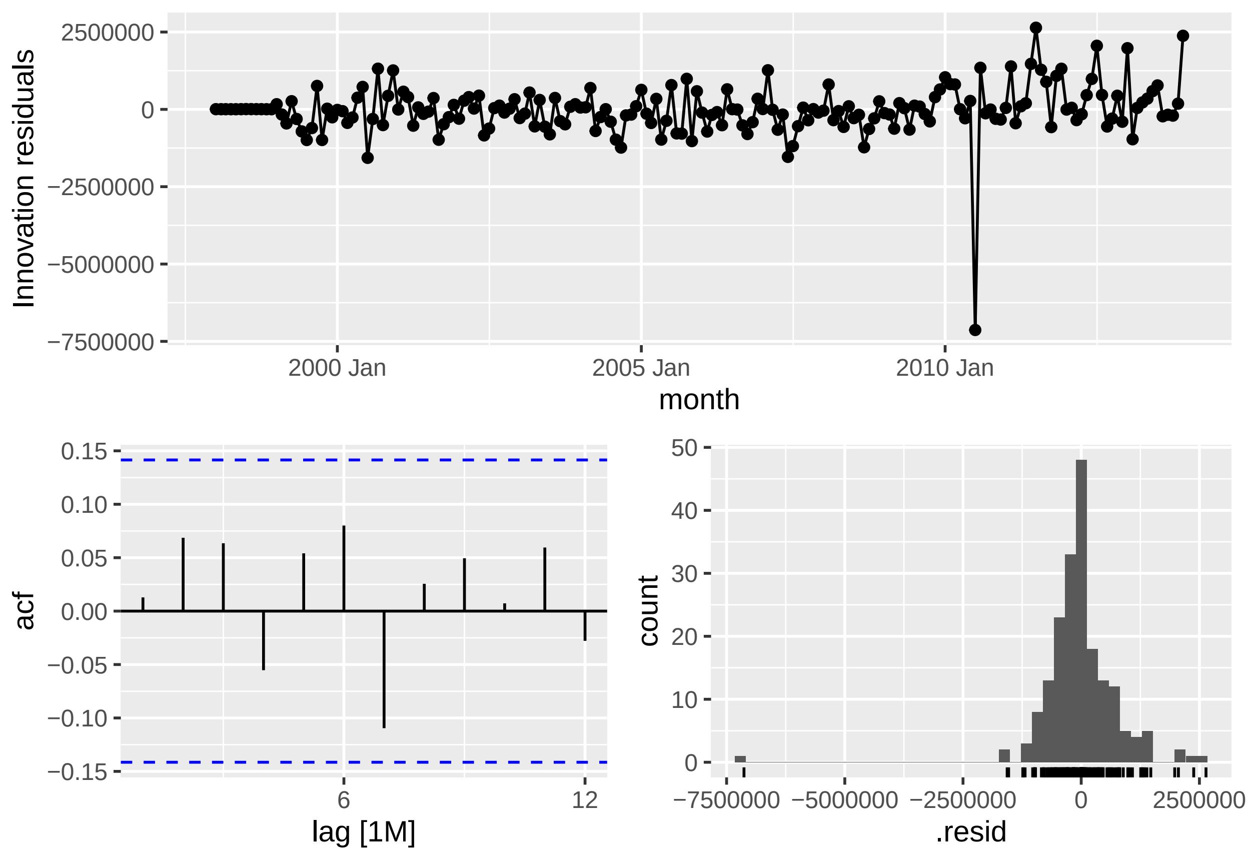
```r
power_fc<- power_fit  %>%
  select('auto_arima') %>%
  forecast(h = '12 month')

#To visualize:
power_fc %>% autoplot(power) +
  ggtitle('KWH usage forecasted monthly for 12 months')
```



Diagnostic analysis:

Ljung-box P-Value = 0.52, indicating that the residuals are white-noise and the model is fairly well fitted

```
#Except for one outlier, the residuals are all well within acceptable range for variation
power_fit %>%
  select(auto_arima) %>%
  gg_tsresiduals(lag = 12)
```



```r
#ljung-box test:
power_fit %>%
  select(auto_arima) %>%
  augment() %>%
  features(.innov, ljung_box, lag = 12, dof = 3)
```

```
## # A tibble: 1 x 3
##   .model     lb_stat lb_pvalue
##   <chr>        <dbl>     <dbl>
## 1 auto_arima    8.17     0.517
```

Exporting the results:

```r
expo<- power_fc %>%
  as.data.frame() %>%
  select(month, .mean) %>%
  rename(KWH = .mean) %>%
  mutate(month = as.character(month))

write_xlsx(expo, "Part_B_KWH.xlsx")
```

Part B Conclusion:

Box-cox transformation was first considered to stabilize the dataset's variations. However, after visualizing the transformed data, the outlier's effect was seen amplified and the raw without the transformation performed fairly well in the fitted model. Thus, transformation of the data was not considered as it may negatively impact the modeling of the dataset and decrease the interpretability. An ARIMA(0,0,1)(1,1,1) was autoselected with step-wise function turned off to increase the search parameters of the model. This ARIMA model out-performs the ETS and the benchmark SNAIVE model and was selected to forecast the data. The residual plots and the Ljung-box test (P-value > 0.05) showed that the residuals are considered to be white-noises and there is not patterns or auto-correlations between them. This model is well fitted to forecast this dataset.

Bonus part C:

loading the data:

```
url1<- 'https://raw.githubusercontent.com/stormwhale/data-mines/refs/heads/main/Waterflow_Pipe1.csv'
url2<- 'https://raw.githubusercontent.com/stormwhale/data-mines/refs/heads/main/Waterflow_Pipe2.csv'

water1<- read.csv(url1)
water2<- read.csv(url2)
```

Convert the data into date format:

```
#Convert the Date.Time column into date format.
water1<- water1 %>%
  rename(Date = 'Date.Time') %>%
  mutate(Date = as.POSIXct(Date, format='%m/%d/%y %I:%M %p'))

water2<-water2 %>%
  rename(Date = 'Date.Time') %>%
  mutate(Date = as.POSIXct(Date, format='%m/%d/%y %I:%M %p'))
```

To combine the two data into one:

```
#To combine the two data into one:
water_combo<- rbind(water1, water2)

#Checking for missing values:
any(is.na(water_combo)) #None
```

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

Taking the mean values for the Dates that are within an hour:

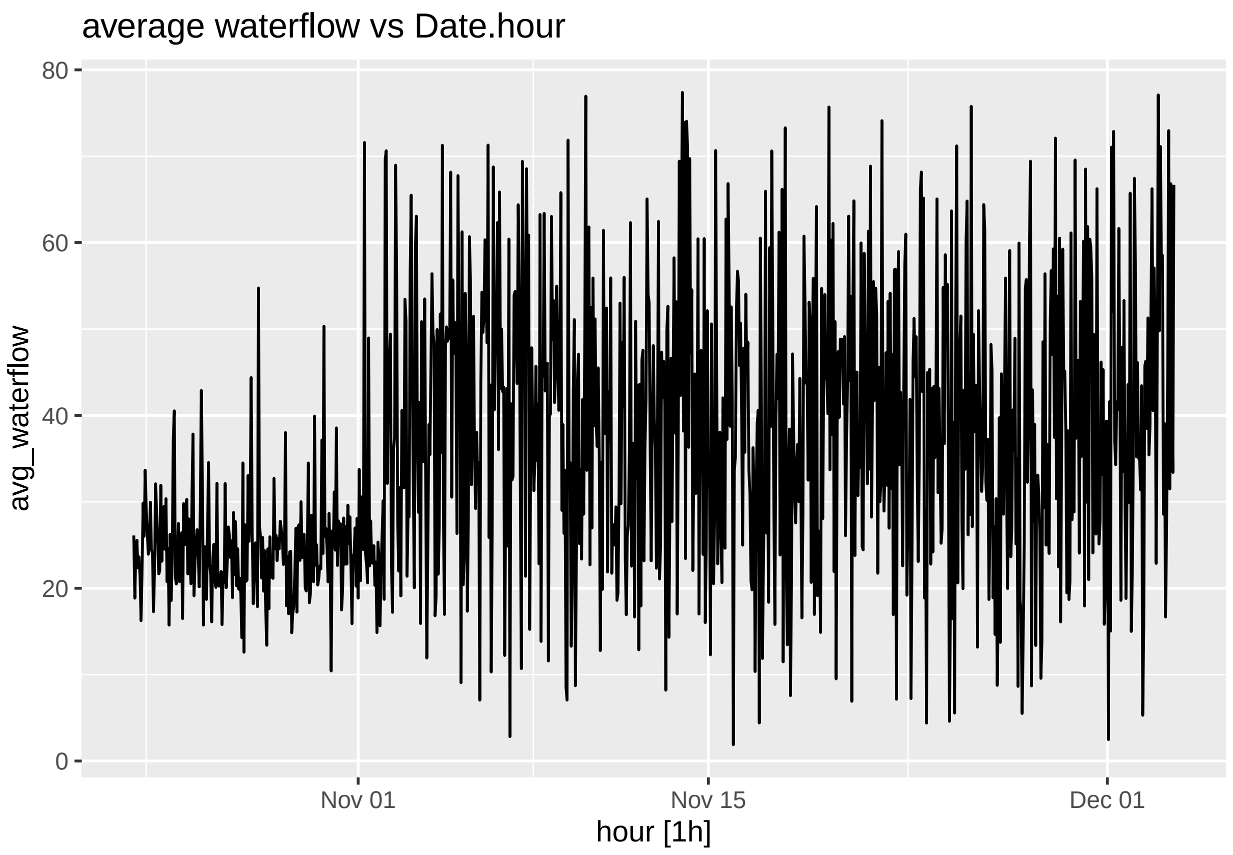
First we will round the time to the nearest hour from the Date column, then extract and group the hours together and get the average value. Finally, we will only consider one time period for the same day with different hours.

```r
avg_waterflow<- water_combo %>%
  mutate(hour = floor_date(Date, unit = 'hour')) %>% #To round and extract the hour
  group_by(hour) %>%
  mutate(avg_waterflow = mean(WaterFlow)) %>%
  ungroup() %>%
  distinct(hour, avg_waterflow) #Extract only the unique value

#To convert this new averaged data into tsibble:
ts_water<-avg_waterflow %>%
  as_tsibble(index=hour)

head(ts_water)
```

```
## # A tsibble: 6 x 2 [1h] <?>
##   hour                avg_waterflow
##   <dttm>                      <dbl>
## 1 2015-10-23 00:00:00          26.1
## 2 2015-10-23 01:00:00          18.8
## 3 2015-10-23 02:00:00          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
```

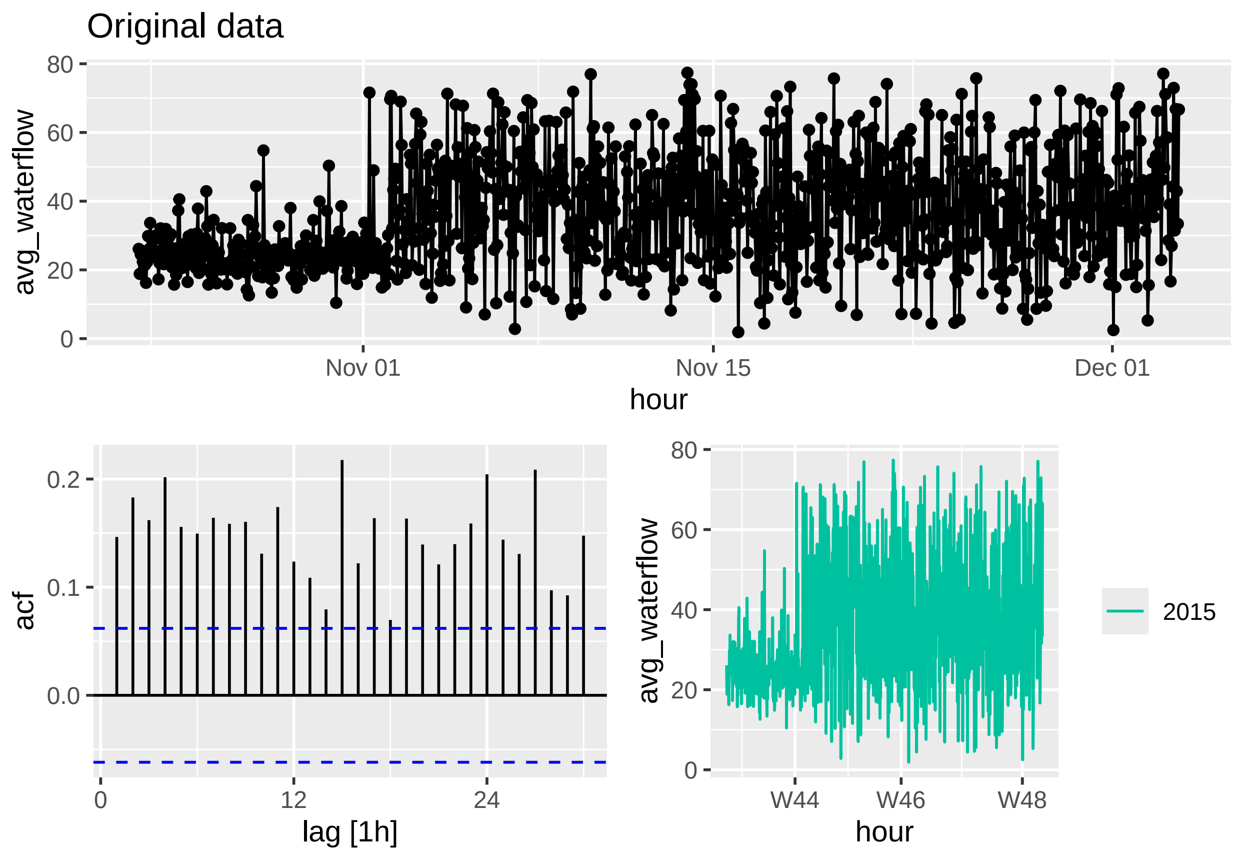
To visualize the data:

```r
ts_water %>%
  autoplot(avg_waterflow) +
  ggtitle('average waterflow vs Date.hour')
```



To check if the combined dataset is stationary and if not, how many differencing is needed to make it stationary:

```
#The data is not stationary and will need differencing to stabilize it.
ts_water %>%
  gg_tsdisplay(avg_waterflow)+ggtitle('Original data')
```



```r
#To check how many degrees of differencing is needed:
ts_water %>%
  features(avg_waterflow, unitroot_ndiffs) #needs 1 differencing
```

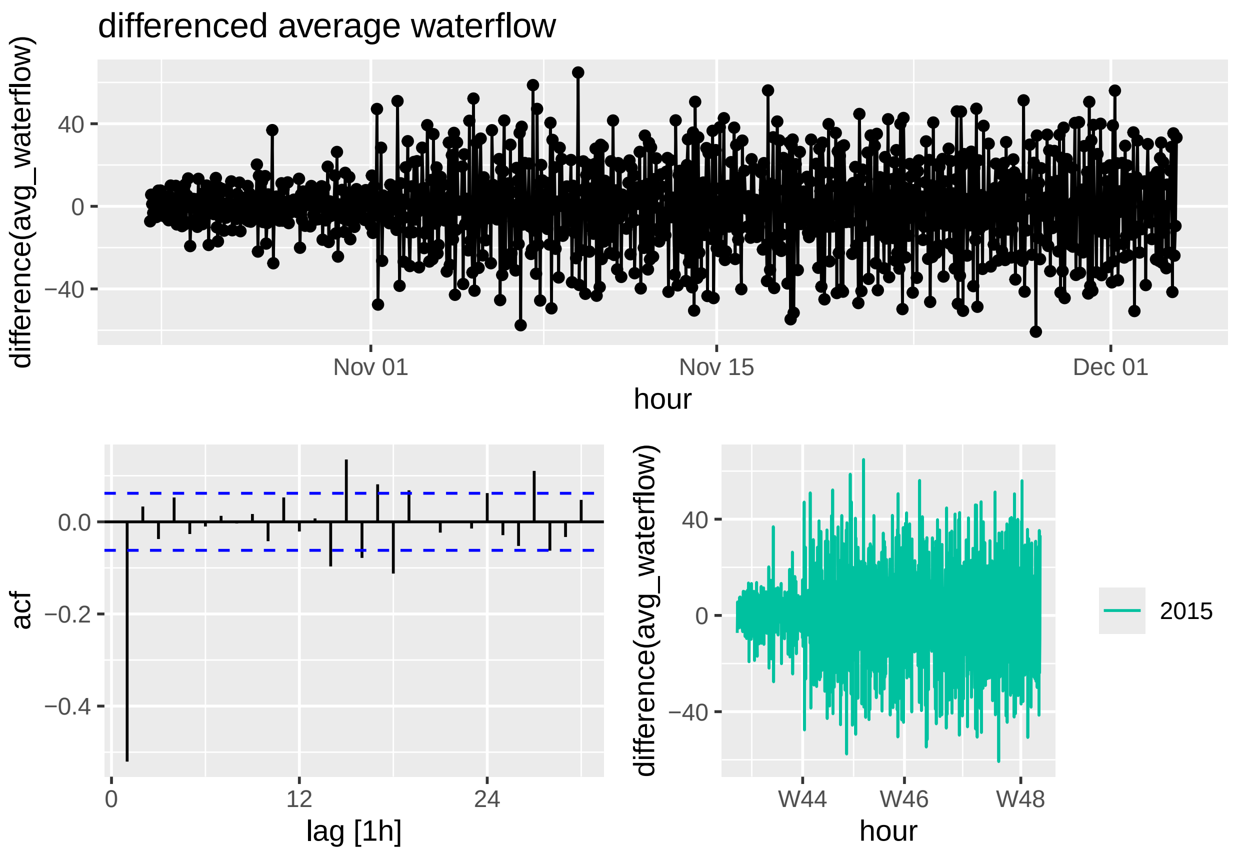
```
## # A tibble: 1 x 1
##   ndiffs
##    <int>
## 1      1
```

```r
#To check if seasonal differencing is needed:
ts_water %>%
  features(avg_waterflow, unitroot_nsdiffs) #No seasonal differencing is needed
```

```
## # A tibble: 1 x 1
##   nsdiffs
##     <int>
## 1       0
```

```r
#Re-check if the differenced data is stationary:
ts_water %>%
  gg_tsdisplay(difference(avg_waterflow))+
  ggtitle('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()‘).

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



```
#Then check with kpss test:
ts_water %>%
  features(difference(avg_waterflow), unitroot_kpss)
```

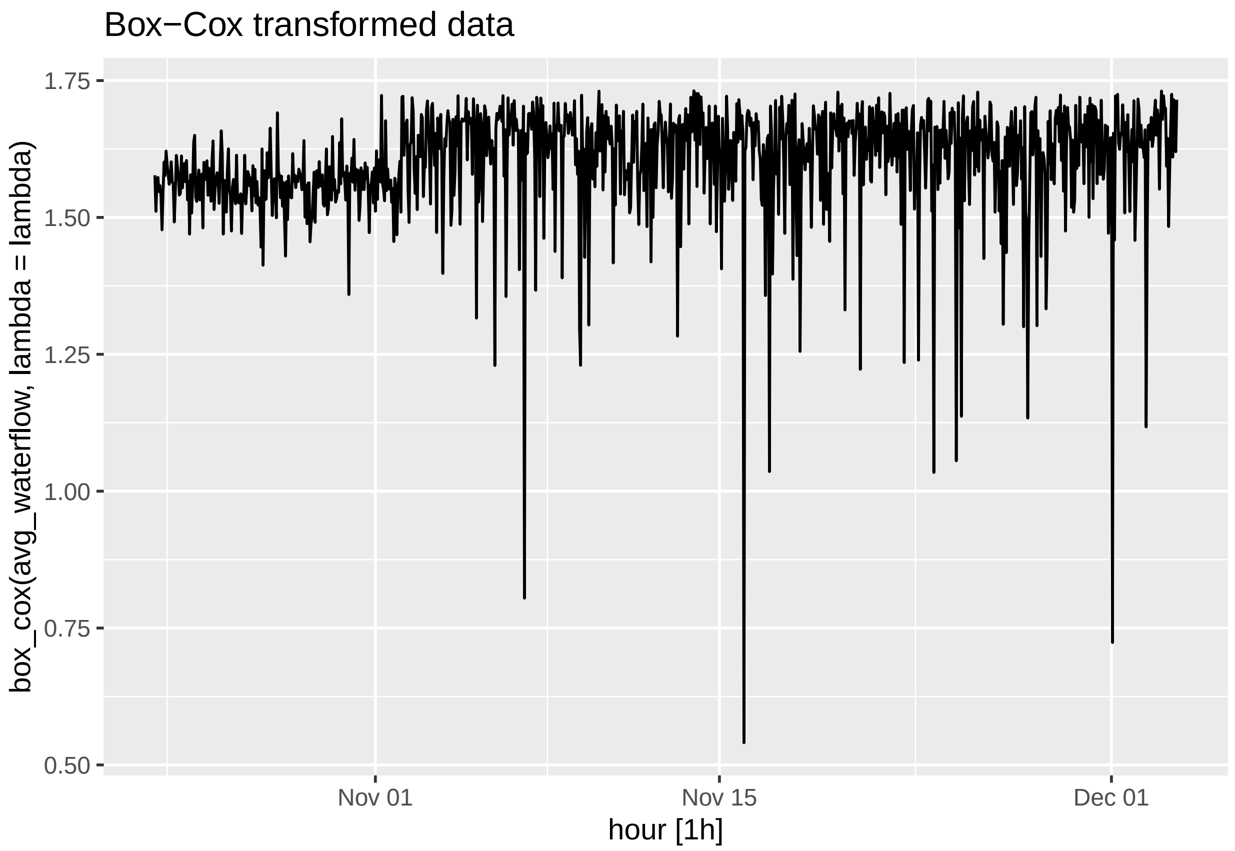
```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1   0.00750         0.1
```

```
#P-value = 0.1. The data appears to be stationary
```

Check to see if data needs transformation, using Box-Cox lambda value:

```
lambda<- ts_water %>%
  features(avg_waterflow, box_cox, feature = guerrero) %>%
  pull(lambda_guerrero )
```

```r
#visualize the transformed data:
ts_water %>%
  autoplot(box_cox(avg_waterflow, lambda= lambda)) + ggtitle('Box-Cox transformed data')
```



The box-cox transformation does not seem to improve the variance by much and also amplifies some of the spikes. We will keep it aside and fit models without it for now.

Fitting the waterflow data into model without the transformation:

```r
water_fit<- ts_water %>%
  model(
    auto_ETS = ETS(avg_waterflow),
    ETS_MNA = ETS(avg_waterflow ~ error('M') + trend('N') + season('A')),
    ETS_MNM = ETS(avg_waterflow ~ error('M') + trend('N') + season('M')),
    auto_arima = ARIMA(avg_waterflow, stepwise = FALSE),
    snaive = SNAIVE(avg_waterflow)
  )
```

The Snaive model is added as a benchmark for the model testing. Two manually selected ETS model (M,N,A) and ETS (M,N,M) are included to capture the seasonality pattern in the data. An auto_selected ARIMA model is also included.

To check the RMSE for each model and select the lowest one for best fit:

```
accuracy(water_fit) %>% select(.model, RMSE, MAE, RMSSE)
```

```
## # A tibble: 5 x 4
##   .model       RMSE   MAE RMSSE
##   <chr>       <dbl> <dbl> <dbl>
## 1 auto_ETS     14.6  11.2 0.742
## 2 ETS_MNA      14.4  11.1 0.734
## 3 ETS_MNM      14.4  11.1 0.734
## 4 auto_arima   14.5  11.1 0.739
## 5 snaive       19.7  15.1 1
```

ETS(MNM) slightly out-performs the ETS(MNA) model and has the lowest RMSE. The ETS(MNM) model will be used for forecasting.

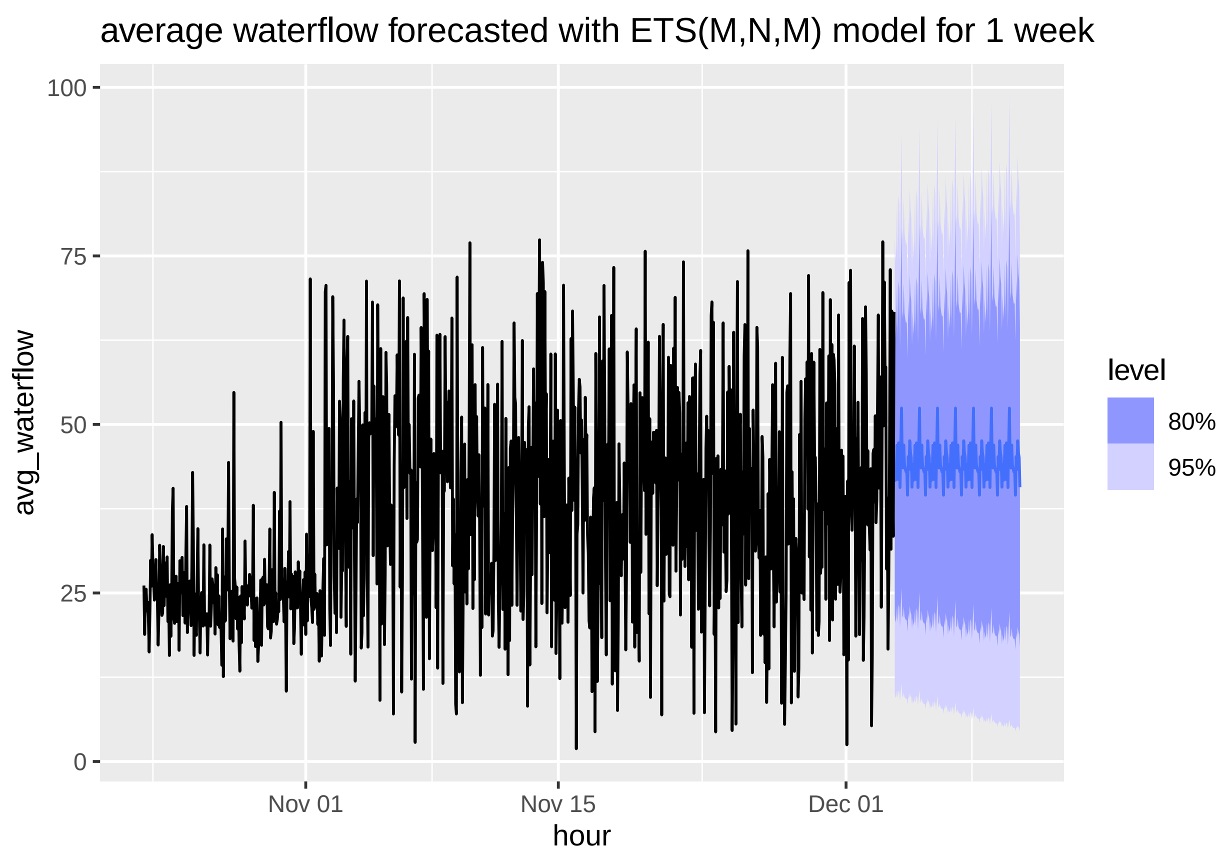
Check the models selected for the auto ETS and ARIMA:

```
water_fit_long<- water_fit %>%
  pivot_longer(everything(),
               values_to = 'order',
               names_to = 'model')

print(water_fit_long)
```

```
## # A mable: 5 x 2
## # Key:     model [5]
##   model                             order
##   <chr>                           <model>
## 1 auto_ETS                   <ETS(M,N,N)>
## 2 ETS_MNA                    <ETS(M,N,A)>
## 3 ETS_MNM                    <ETS(M,N,M)>
## 4 auto_arima <ARIMA(0,1,1)(0,0,1)[24]>
## 5 snaive                         <SNAIVE>
```

Visualizing the forecasted data with the ETS_MNM model:

```
water_fc<- water_fit %>%
  select(ETS_MNM) %>%
  forecast(h= '1 week')

water_fc %>%
  autoplot(ts_water) +
  ggtitle('average waterflow forecasted with ETS(M,N,M) model for 1 week')
```

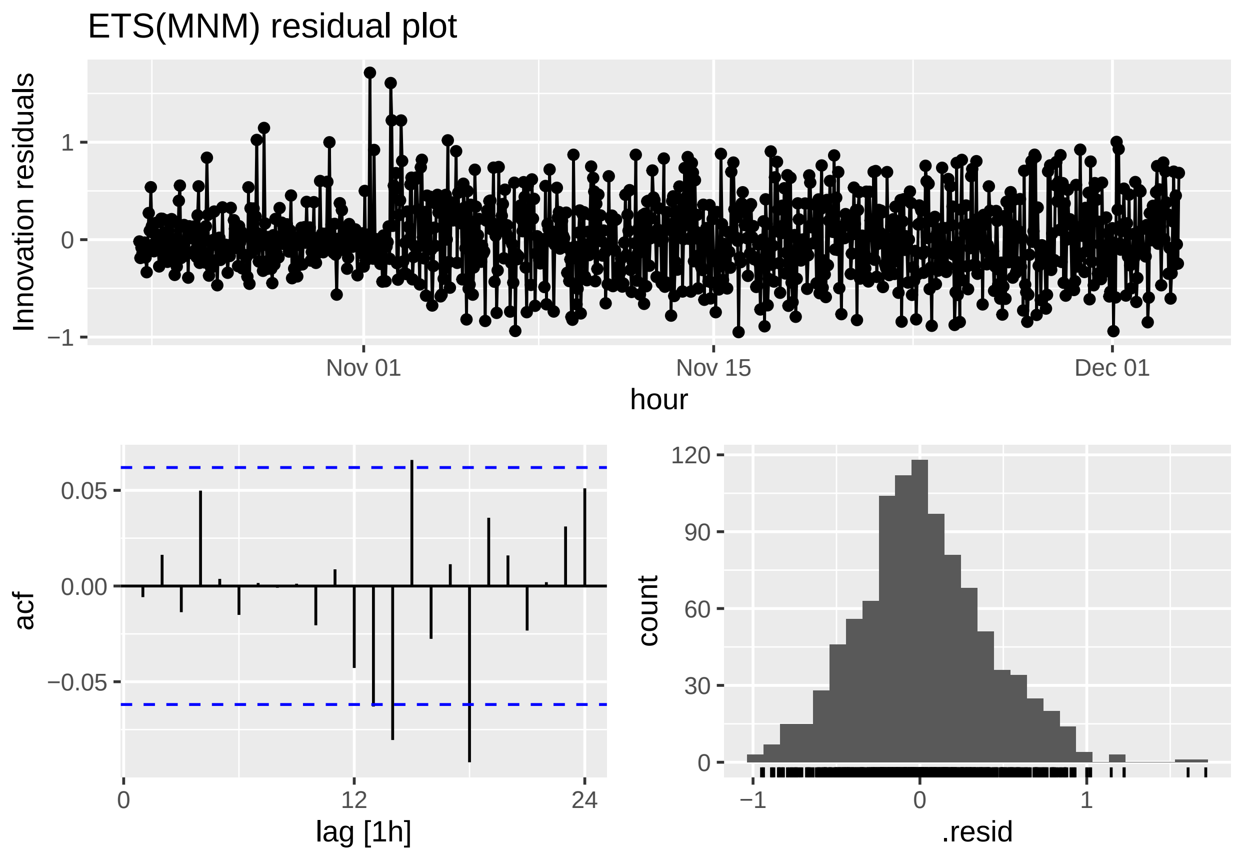


Diagnostic analysis:

To ensure the ETS_MNM fully captures all the data in the fitted model, we will analyzes its residuals to determine if they are white-noises.

Ljung-Box p-value = 0.055. We will accept the Null hypothesis that the residuals are white-noises.

```
#visualizing the residual plot:
water_fit %>%
  select(ETS_MNM) %>%
  gg_tsresiduals(lag=24) +
  ggtitle('ETS(MNM) residual plot')
```



```r
#To compute the ljung-box statistical test:
water_fit %>%
  select(ETS_MNM) %>%
  augment() %>%
  features(.innov, ljung_box, lag = 24)
```

```
## # A tibble: 1 x 3
##   .model  lb_stat lb_pvalue
##   <chr>     <dbl>     <dbl>
## 1 ETS_MNM    36.0    0.0545
```

```r
#p-value =0.055 Passed the ljung-box test.
```

To export the results:

```r
water_expo<- water_fc %>%
  as.data.frame() %>%
  select(hour, .mean) %>%
  rename(Date.hour = 'hour', avg_waterflow = '.mean')

write_xlsx(water_expo, 'average_waterflow.xlsx')
```

Bonus Part C Conclusion:

The two waterflow data were combined into one big data frame and the average waterflow values were calculated when the date and hours are within 1 hour period. The data was determined to be non-stationary and confirmed that 1 differencing is needed by the unit-root test. However, no seasonal differencing is needed. Box-Cox or other transformations were avoided due slim improvement seen and the transformed data could reduce interpretability of the result. The auto-arima model (0,1,1)(0,0,1) confirms that only one differencing was applied and no seasonal differencing was used. The auto_ETS model selected (M,N,N), which did not include seasonality, which results in higher RMSE. However, seasonal patterns are observed in the raw data and two manually selected ETS models ETS_MNA and ETS_MNM were introduced to model the data. A SNAIVE model is also introduced as a benchmark. The RMSE showed that the ETS_MNM has the lowest RMSE, out-performing other models. This is most likely due to how the data shifts over time and the multiplicity of the seasonality is better fitted for this data. The fitted model from the ETS_MNM was then analyzed for white-noise in the residual plots and the Ljung-Box statistical test. Although the ACF showed few spikes after lag 12, the P-value from Ljung-Box test showed the P-value = 0.055, which accepts for null hypothesis that the residuals are white-noises. The ETS_MNM will produce a fairly accurate prediction for the waterflow data.