ShovanBiswas-DATA24_Project_01

Shovan Biswas

10/21/2020

Project 1 Description

This project consists of 3 parts - two required and one bonus and is worth 15% of your grade. The project is due at 11:59 PM on Sunday Oct 25. I will accept late submissions with a penalty until the meetup after that when we review some projects.

Libraries

```
library(tidyverse)
library(dplyr)
library(readxl)
library(forecast)
library(lubridate)
library(tseries)
library(kableExtra)
library(corrplot)
library(caret)
library(Amelia)
library(dlookr)
library(plotly)
library(gridExtra)
library(ggplot2)
library(urca)
library(tseries)
library(xlsx)
library(DT)
```

Part A – ATM Forecast, ATM624Data.xlsx

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

Reading data from ATM624Data2.csv.

Please note that first I tried to work with file ATM624Data.xlsx, but after reading I had problems manipulating the data. So, I exported ATM624Data.xlsx into ATM624Data2.csv and worked on it.

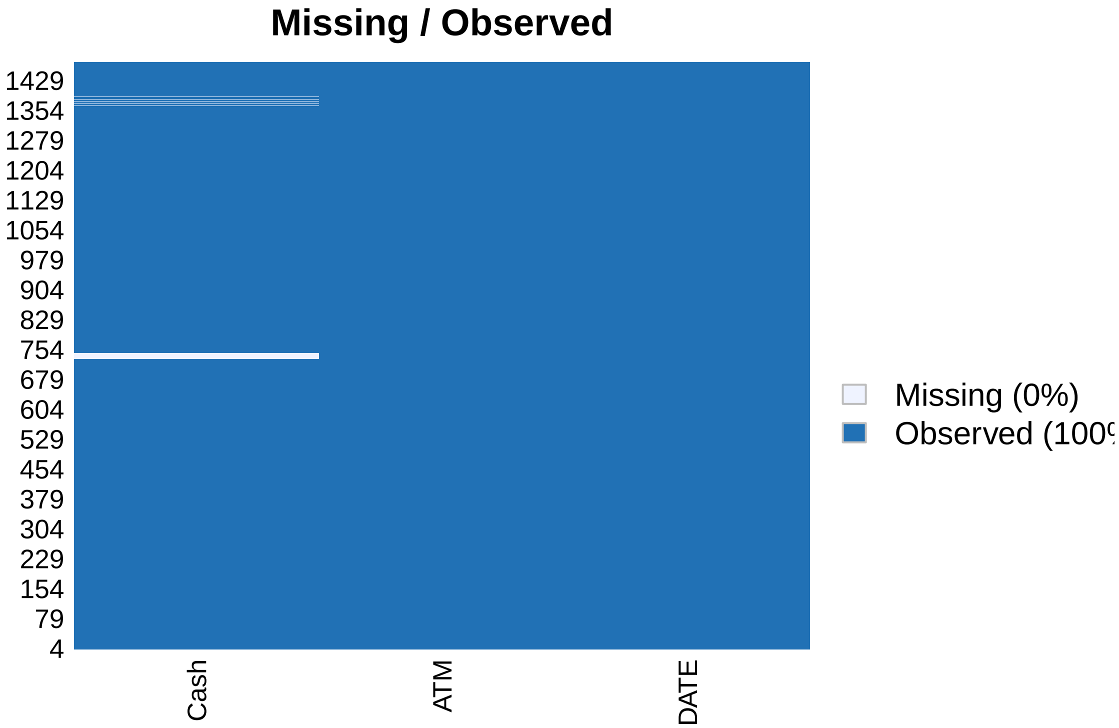
```
# atm_data <- read_excel('./ATM624Data.xlsx') %>% mutate(DATE = as.Date(DATE, origin = "1899-12-30"))
atm_data <- read.csv('./ATM624Data2.csv')
print(paste0("Number of rows = ", nrow(atm_data)))
```

```
## [1] "Number of rows = 1474"
```

Data exploration.

The missmap() function tells me that there are missing values in data, but it's not clear and detailed.

```
atm_data %>% missmap(main = "Missing / Observed")
```



By taking a summary, I observe that there are 19 cases missing cash values.

On the ATM column, there are 14 missing values. SO, out of the 19 records with missing cash values, 14 would not have ATMs (assuming that there is no record with ATM, but no Cash, which would be absurd).

So, I'll remove those 14 records and impute the remaining 5 missing cash values.

```
atm_data %>% summary()
```

```
##                      DATE        ATM          Cash
##  1/1/2010 12:00:00 AM :   4       : 14   Min.    :    0.0
##  1/10/2010 12:00:00 AM:   4   ATM1:365   1st Qu.:    0.5
##  1/11/2010 12:00:00 AM:   4   ATM2:365   Median :   73.0
##  1/12/2010 12:00:00 AM:   4   ATM3:365   Mean   :  155.6
##  1/13/2010 12:00:00 AM:   4   ATM4:365   3rd Qu.:  114.0
##  1/14/2010 12:00:00 AM:   4              Max.   :10920.0
##  (Other)              :1450              NA's   :19
```

Data sanitization.

I'll *drop* 14 rows with missing ATMs, first.

```
atm_data <- atm_data %>% filter( !(atm_data$ATM == '') )
print(paste0("Number of rows = ", nrow(atm_data)))
```

```
## [1] "Number of rows = 1460"
```

Here, I'll *impute* the 5 cash values with mean of the cash values in their respective ATM-groups. The ATM means are given below.

```
atm_mean <- atm_data %>% filter(!is.na(Cash)) %>% group_by(ATM) %>% summarise(m = mean(Cash))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
atm_mean  # display the means
```

```
## # A tibble: 4 x 2
##   ATM         m
##   <fct>   <dbl>
## 1 ATM1   83.9
## 2 ATM2   62.6
## 3 ATM3    0.721
## 4 ATM4  474.
```

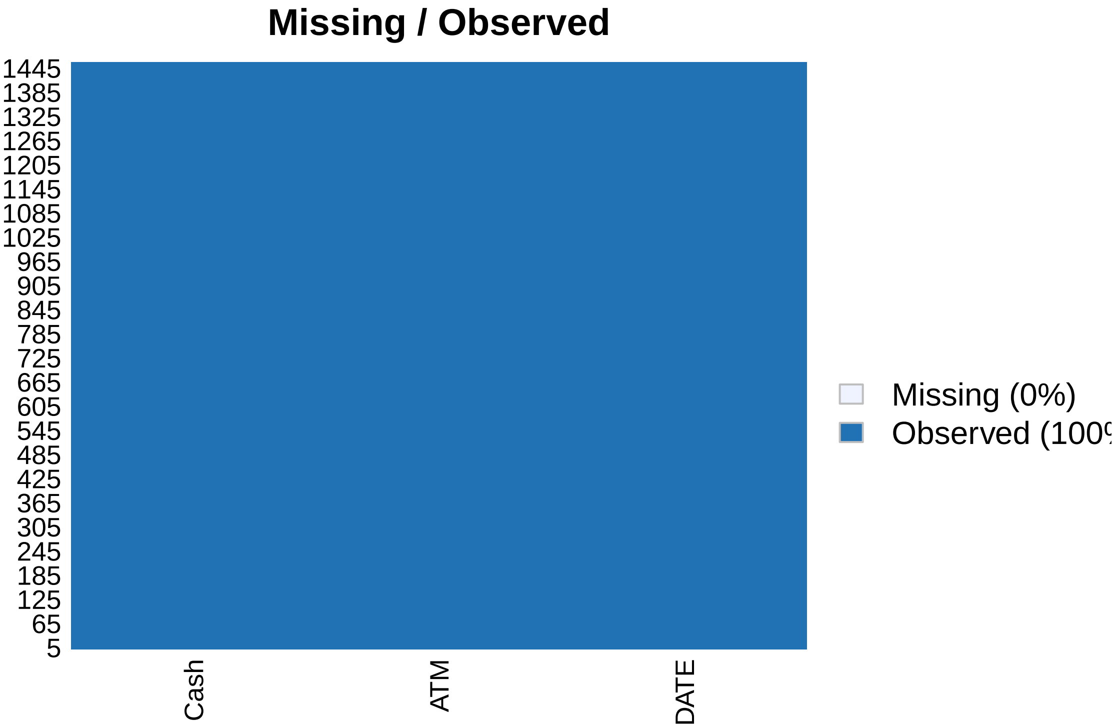
```
#
if( length(atm_data[is.na(atm_data$Cash) & atm_data$ATM == 'ATM1', ]$Cash) )  {atm_data[is.na(atm_data$C
if( length(atm_data[is.na(atm_data$Cash) & atm_data$ATM == 'ATM2', ]$Cash) )  {atm_data[is.na(atm_data$C
if( length(atm_data[is.na(atm_data$Cash) & atm_data$ATM == 'ATM3', ]$Cash) )  {atm_data[is.na(atm_data$C
if( length(atm_data[is.na(atm_data$Cash) & atm_data$ATM == 'ATM4', ]$Cash) )  {atm_data[is.na(atm_data$C
```

I created below code chunk for dumping the imputed the dataset, to verify whether it was indeed imputed in the appropriate places. But I am keeping it commented.

```
# write.csv(atm_data, './ATM624Data2_out.csv')
```

The missmap() function confirms that there are missing values in data.

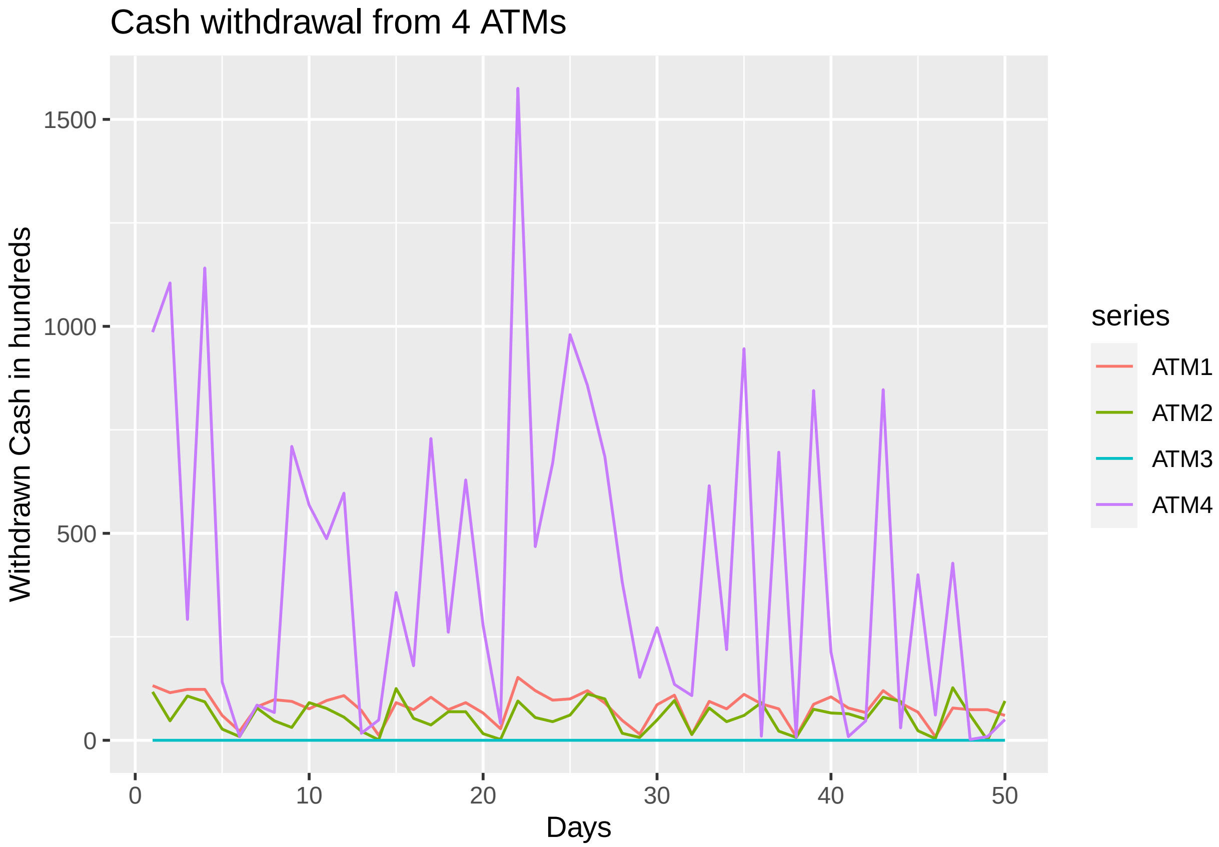
```
atm_data %>% missmap(main = "Missing / Observed")
```



So, at this point, the data is sanitized, and we can form time series, visualize the data and proceed to make forecasts.

A first look at withdrawals from the ATMs.

```
autoplot(ts(ts(atm_data %>% spread(ATM, Cash) %>% select(-DATE))[1:50, ])) + ggtitle("Cash withdrawal fr
```



Having taken a bird's eye view of the activities at 4 ATMs, we'll take a detailed look at each ATM level.

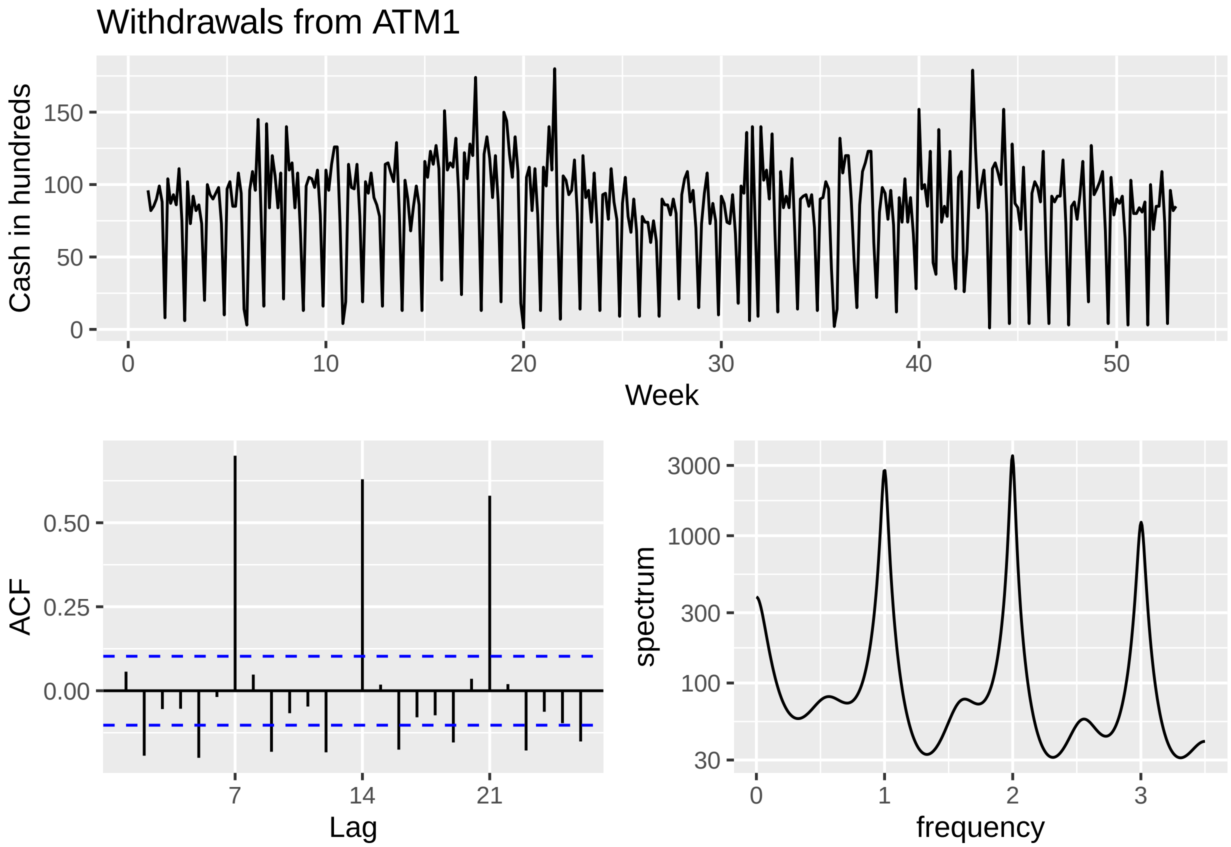
In each of the the four code-chunks, we'll separate Cash for each ATM and then visualize them with function ggtsdisplay().

ATM1 withdrawals:

In the ATM1, there was considerable amount of withdrawal activity. So, ATM1 is a candidate for forecasting. In ACF, the spikes peak at 7th lag and in spectrum plot, at frequencies 1, 2, 3.

So, ACF and spectrum plots suggest a seasonal Arima model.

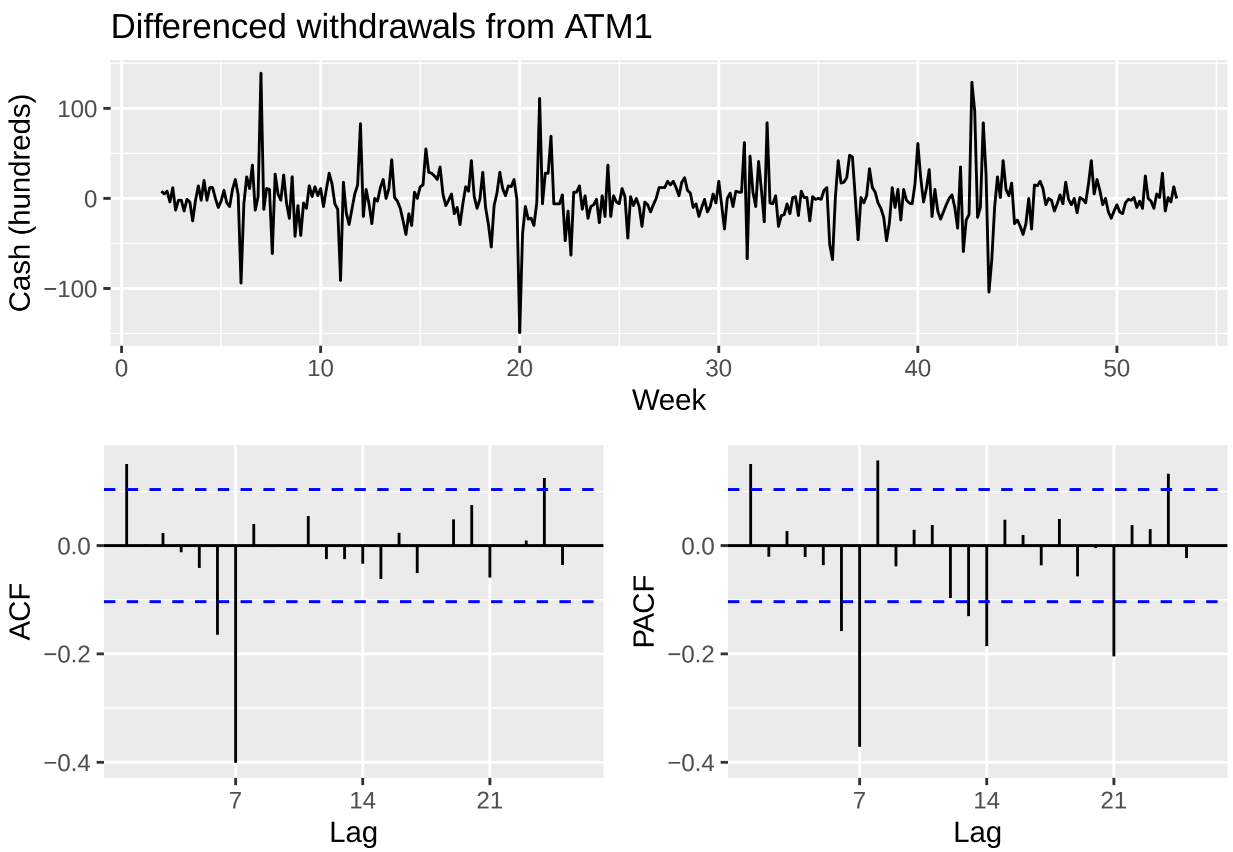
```
ATM1 <- filter(atm_data, ATM == "ATM1")
ATM1_ts <- ts(ATM1$Cash, frequency = 7)
ATM1_ts %>% ggtsdisplay(points = FALSE, plot.type = "spectrum", main = "Withdrawals from ATM1", xlab =
```



The ACF graph shows weekly seasonality, with large spikes in lags 7, 14 and 21 (i.e. at k = 1, 2, 3), and the Spectrum plot shows the frequencies 1, 2, 3. This is suggestive of seasonal ARIMA model.

There is a lag of 7, so it's being differenced right below.

```r
diff(ATM1_ts, 7) %>% ggtsdisplay(points = FALSE, main = "Differenced withdrawals from ATM1", xlab = "We
```



```r
ndiffs(ATM1_ts)
```

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

Observations after differencing: The time series looks stationary. There is a large spike in ACF at lag k = 1. There're moderate spikes in PACF at lags 7, 14 and 21.

At this point, our aim is to find the appropriate ARIMA model, based on ACF and PCF. The significant spike at first lag in ACF and PACF suggests a non-seasonal AR(1) component, and the significant spike at lag 7 in ACF suggests a seasonal AR(1) component (please refer page 257 of printed book). So, there are 16 - 1 = 15 possible models: AIRMA(p, d, q)(P, d, Q).

And why so (i.e. 15 possibilities)? We observed that ndiffs(ATM1_ts) = 0. Therefore d = 0. So, in AIRMA(p, d, q)(P, D, Q), d is fixed as 0 and D is fixed as 1. It's a case of Ar(1). So, p, q, P, Q are the only free variables, which can only choose between 0 and 1, without all of them being 0. Therefore, by plain combinatorics, we get $2^4 - 1 = 16 - 1 = 15$ combinations.

Explanation of (p, d, q) and (P, d, Q) (refer page 255 of printed book):
(p, d, q) is for the non-seasonal part of the model.
(P, D, Q) is for the seasonal part of the model.

So, we'll apply AIC function (Akaike's An Information Criterion. Refer: https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/AIC), on each of 15 combinations, mentioned above and select minimum AIC.

```
ATM1_eval_aic <- function(p, d, q, P, D, Q) {
  AIC(Arima(ATM1_ts, order = c(p, d, q), seasonal = c(P, D, Q), lambda = BoxCox.lambda(ATM1_ts)))
}
```

```
expand.grid(p = c(0, 1), q = c(0, 1), P = c(0, 1), Q = c(0, 1)) %>% filter(p > 0 | q > 0 | P > 0 | Q >
```

p	q	P	Q	aic
1	1	0	1	1170.495

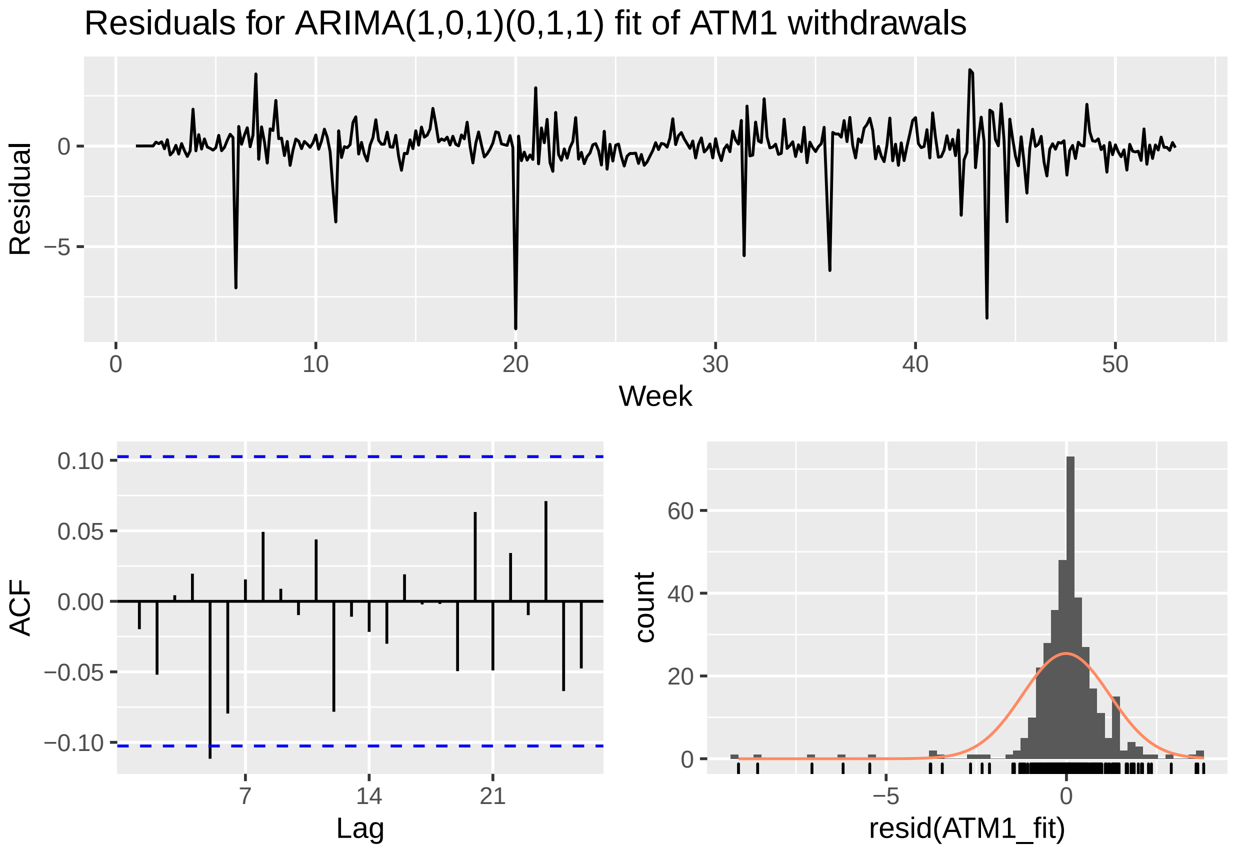
This is the minimum AIC value for non-seasonal AR(1) and seasonal AR(0). So, the model is ARIMA(1, 0, 1)(0, 1, 1).

This is a crucial step in the forecasting.

```
ATM1_fit <- Arima(ATM1_ts, order = c(1, 0, 1), seasonal = c(0, 1, 1), lambda = BoxCox.lambda(ATM1_ts))
Box.test(resid(ATM1_fit), type = "L", lag = 7)
```

```
##
##  Box-Ljung test
##
## data:  resid(ATM1_fit)
## X-squared = 8.3791, df = 7, p-value = 0.3004
```

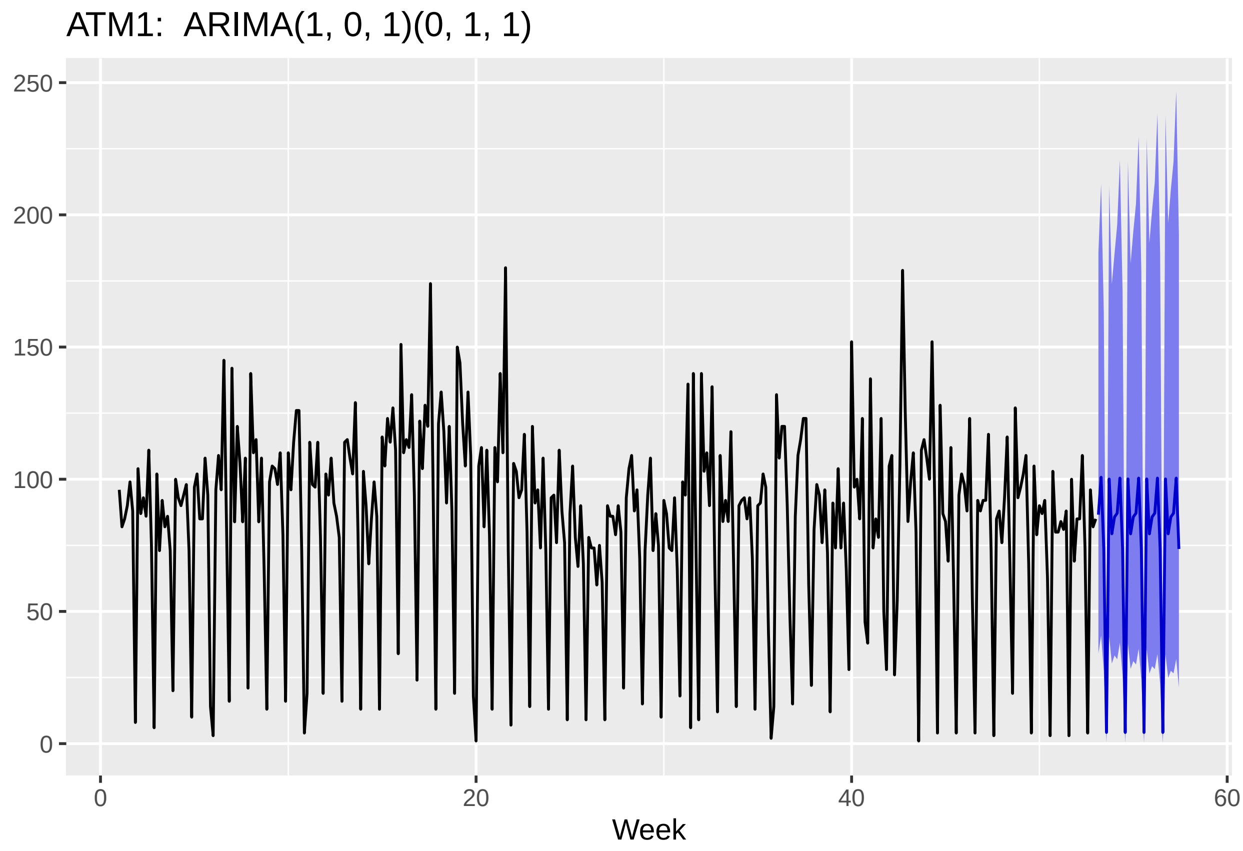
```
ggtsdisplay(resid(ATM1_fit), points = FALSE, plot.type = "histogram", main = "Residuals for ARIMA(1,0,1
```



Final observations on ATM1 withdrawals:
- Based on Box-Ljung test, p-value = 0.3004 > 0.05, which suggest that residuals might be White Noise.
- The histogram shows that the residuals are normally distributed aroud mean of 0.
- Can't comment about autocorrelation.

The forecast for ATM1 is as follows:

```r
ATM1_forecast <- forecast(ATM1_fit, 31, level = 95)
#
autoplot(ATM1_forecast) + labs(title = "ATM1:  ARIMA(1, 0, 1)(0, 1, 1)", x = "Week", y = NULL) + theme(
```

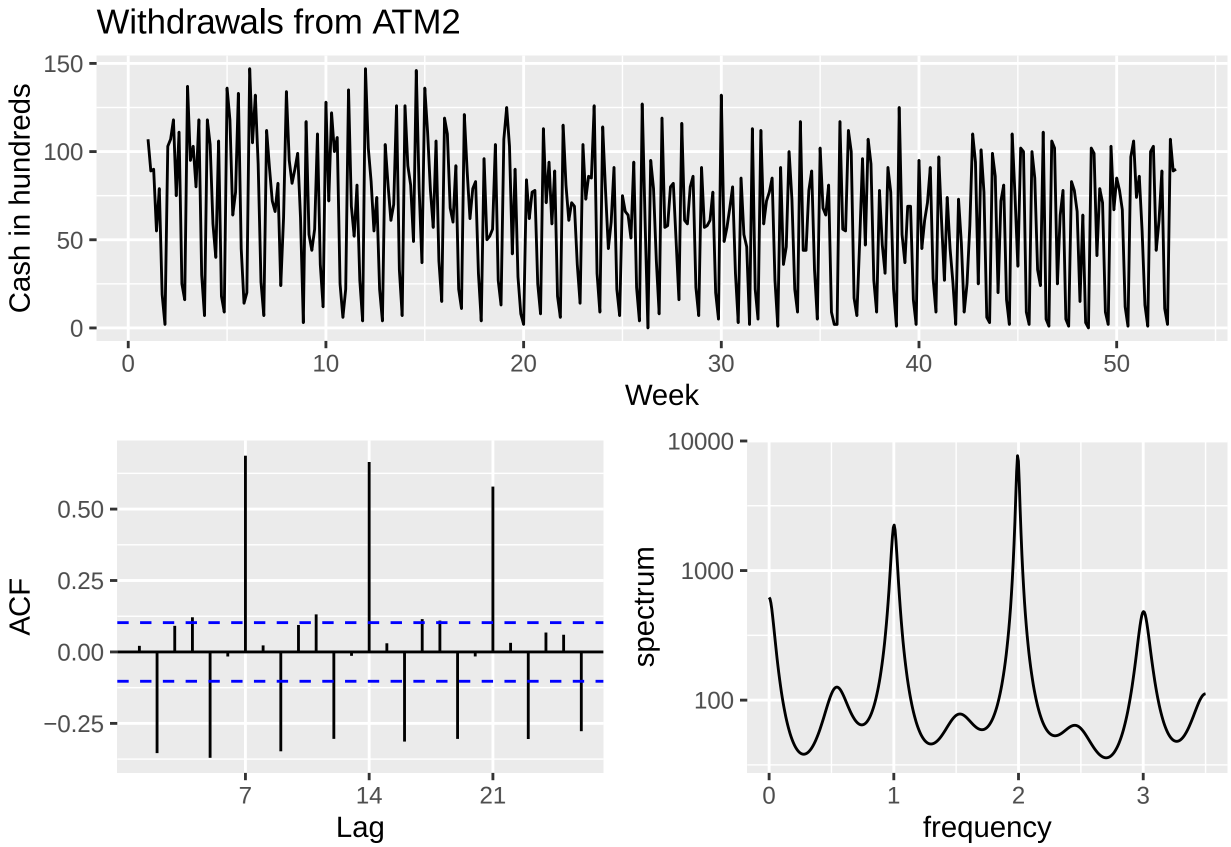


ATM2 withdrawals:

In the ATM2 too, there was considerable amount of withdrawal activity. So, ATM2 is also a candidate for forecasting.
The below graphs (undifferenced) strongly resemble ATM1 graphs. In ACF, the spikes peak at 7th lag and in spectrum plot, at frequencies 1, 2, 3.

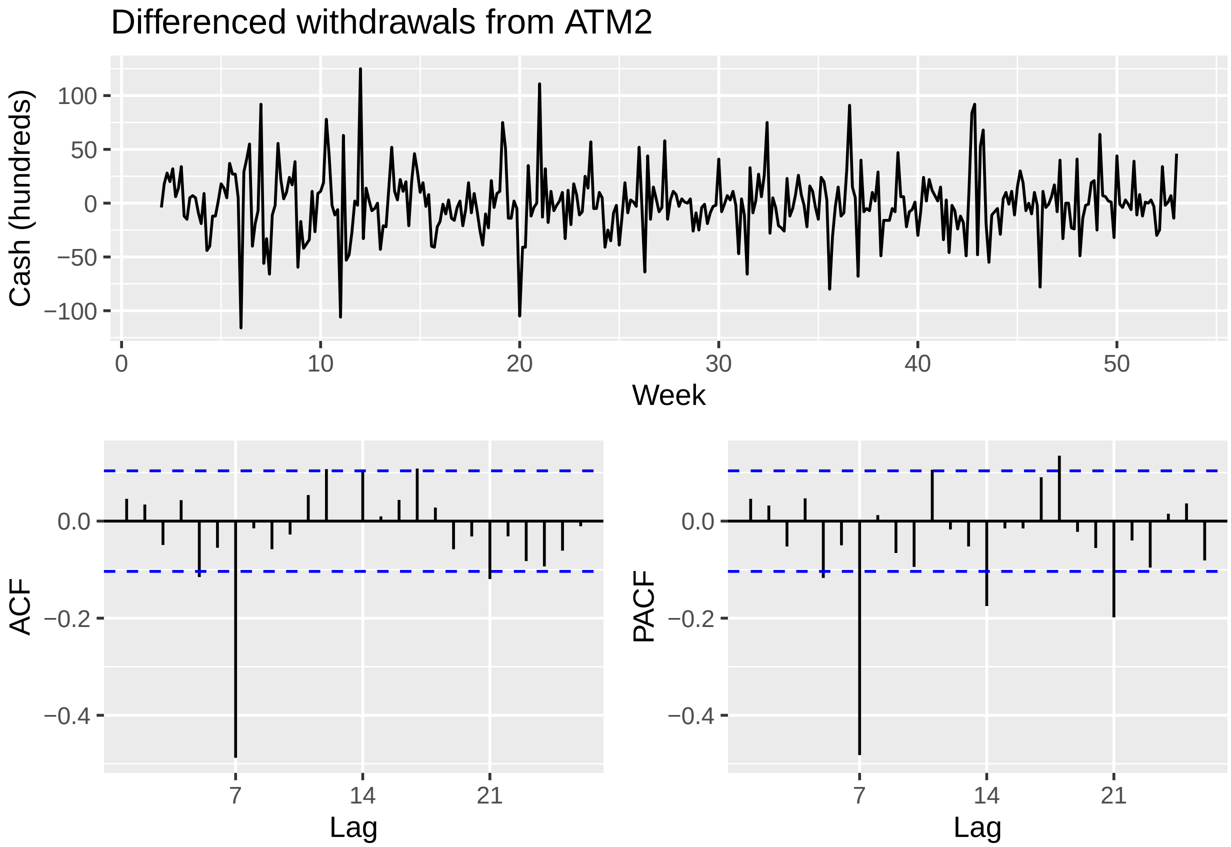
So, ACF and spectrum plots suggest a seasonal Arima model.

```r
ATM2 <- filter(atm_data, ATM == "ATM2")
ATM2_ts <- ts(ATM2$Cash, frequency = 7)
ATM2_ts %>% ggtsdisplay(points = FALSE, plot.type = "spectrum", main = "Withdrawals from ATM2", xlab =
```



Since the undifferenced ACF and Spectrum ATM2 look like Like those of ATM1, there is a weekly seasonality, so it's being differenced right below.

```
diff(ATM2_ts, 7) %>% ggtsdisplay(points = FALSE, main = "Differenced withdrawals from ATM2", xlab = "We
```



```
print(paste0("FIrst differencing of ATM2_ts: ", ndiffs(ATM2_ts)))
```

```
## [1] "FIrst differencing of ATM2_ts: 1"
```

```
print(paste0("Second differencing of ATM2_ts: ", ndiffs(diff(ATM2_ts))))
```

```
## [1] "Second differencing of ATM2_ts: 0"
```

Observations: After differencing looks stationary (same as in ATM1). There's a large spike in ACF at lag k = 1. There're moderate spikes in PACF at lags 7, 14 and 21 (no different from ATM1).

So, the conditions look similar to ATM1, except that the degree of differencing is d = 1. So, we'll use the same technique to minimize AIC.

```
ATM2_eval_aic <- function(p, d, q, P, D, Q) {
  AIC(Arima(ATM2_ts, order = c(p, d, q), seasonal = c(P, D, Q), lambda = BoxCox.lambda(ATM2_ts)))
}
```

```
expand.grid(p = c(0, 1), q = c(0, 1), P = c(0, 1), Q = c(0, 1)) %>% filter(p > 0 | q > 0 | P > 0 | Q >
```

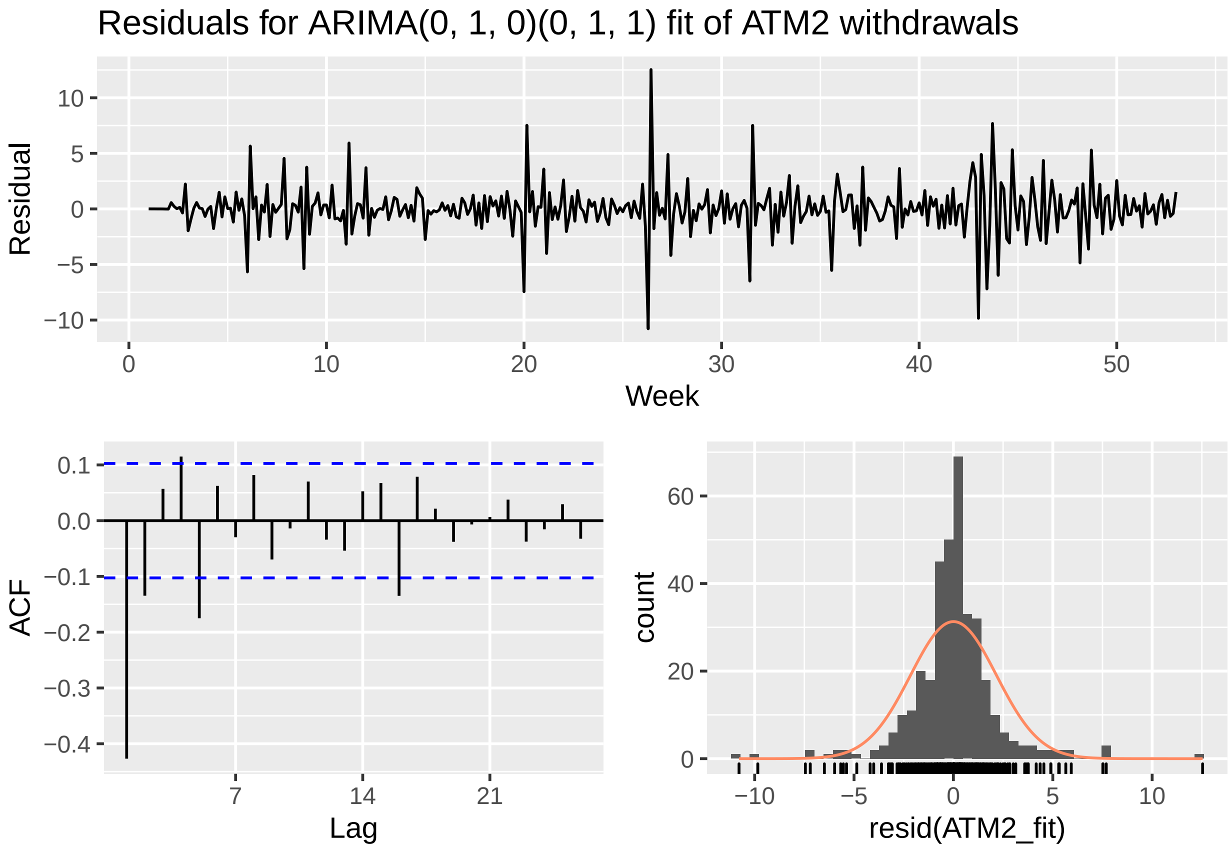
p	q	P	Q	aic
0	1	0	1	2551.867

This is the minimum AIC value for non-seasonal AR(1) and seasonal AR(0). So, the model is ARIMA(0, 1, 0)(0, 1, 1).

```
ATM2_fit <- Arima(ATM2_ts, order = c(0, 1, 0), seasonal = c(0, 1, 1), lambda = BoxCox.lambda(ATM1_ts))
Box.test(resid(ATM2_fit), type = "L", lag = 7)
```

```
##
##  Box-Ljung test
##
## data:  resid(ATM2_fit)
## X-squared = 92.98, df = 7, p-value < 2.2e-16
```

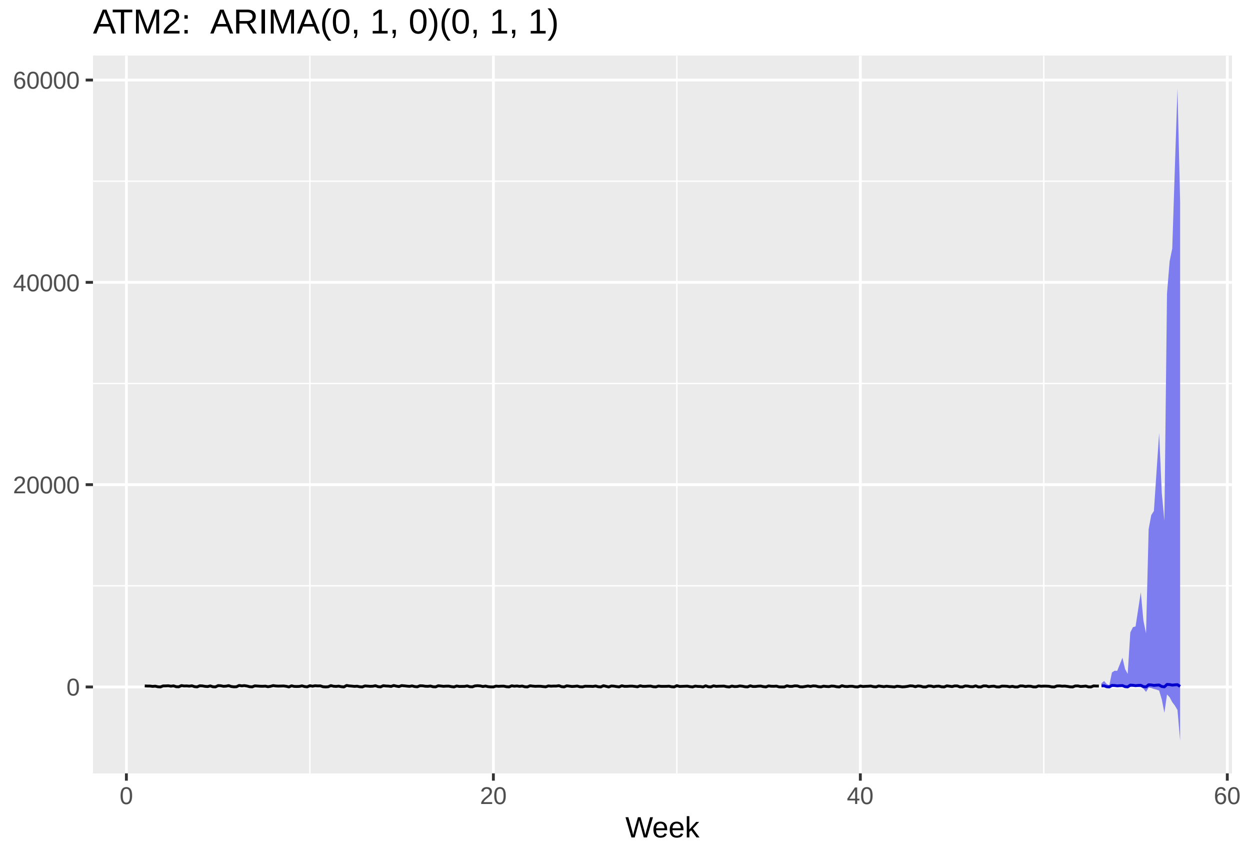
```
ggtsdisplay(resid(ATM2_fit), points = FALSE, plot.type = "histogram", main = "Residuals for ARIMA(0, 1,
```



Final observations on ATM2 withdrawals:
- Based on Box-Ljung test, p-value = 2.2e-16 < 0.05, which suggest that residuals might not be White Noise.
- The histogram shows that the residuals are normally distributed aroud mean of 0.
- Can't comment about autocorrelation.

The forecast for ATM2 is as follows:

```
ATM2_forecast <- forecast(ATM2_fit, 31, level = 95)
#
autoplot(ATM2_forecast) + labs(title = "ATM2:  ARIMA(0, 1, 0)(0, 1, 1)", x = "Week", y = NULL) + theme(
```

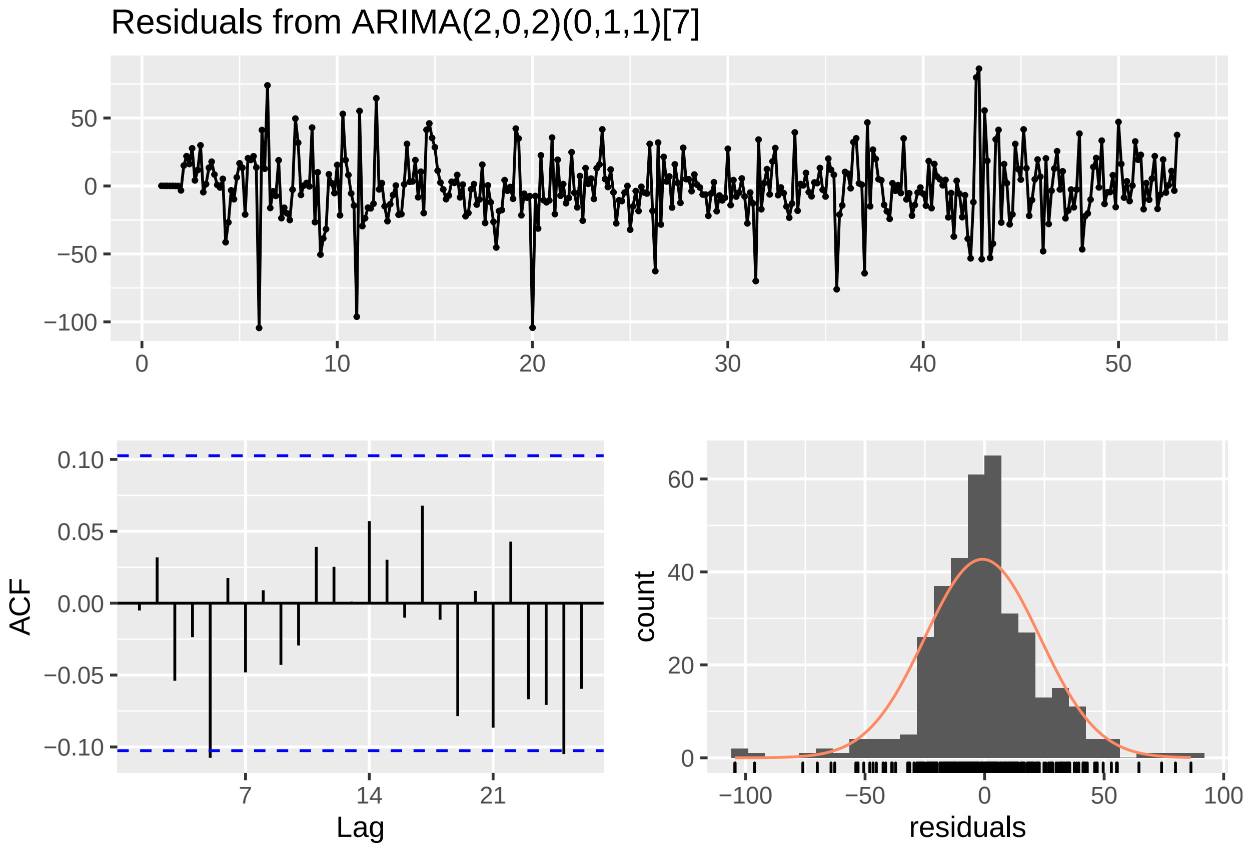


The condition is similar to ATM1, but the forecasting is way different, so it doesn't look reliable. So, I'll try Arima's auto.arima() function.

In above forecasting for ATM2, the indicators for the values p, d, q, P, D, Q for ARIMA(p, d, q)(P, D, Q) might not have been correct. So, we'll use function auto.arima().

auto.arima(ATM2_ts) gives us the values for ARIMA(p, d, q)(P, D, Q).

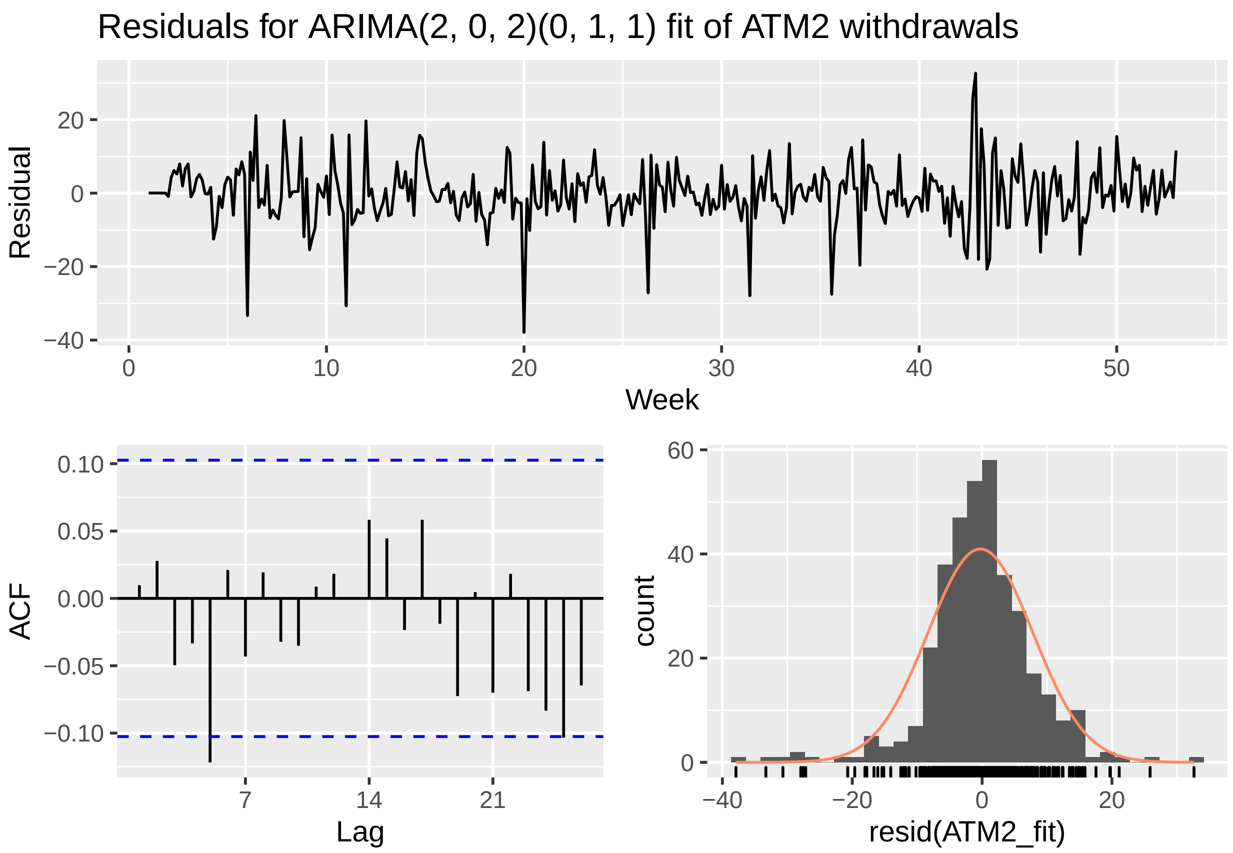
```
ATM2_fit <- auto.arima(ATM2_ts)
checkresiduals(ATM2_fit)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,0,2)(0,1,1)[7]
## Q* = 10.073, df = 9, p-value = 0.3446
##
## Model df: 5.   Total lags used: 14
```

Now, we'll run Arima, with order and seasonal.

```
ATM2_fit <- Arima(ATM2_ts, order = c(2, 0, 2), seasonal = c(0, 1, 1), lambda = BoxCox.lambda(ATM2_ts))
ggtsdisplay(resid(ATM2_fit), points = FALSE, plot.type = "histogram", main = "Residuals for ARIMA(2, 0,
```



Residuals for ARIMA(2, 0, 2)(0, 1, 1) fit of ATM2 withdrawals

```
# checkresiduals(ATM2_fit)
```

```
ATM2_fit
```

```
## Series: ATM2_ts
## ARIMA(2,0,2)(0,1,1)[7]
## Box Cox transformation: lambda= 0.7240342
##
## Coefficients:
##           ar1      ar2     ma1     ma2     sma1
##       -0.4245  -0.9067  0.4682  0.7993  -0.7251
## s.e.   0.0587   0.0440  0.0889  0.0586   0.0409
##
## sigma^2 estimated as 67.58:  log likelihood=-1261.79
## AIC=2535.57   AICc=2535.81   BIC=2558.86
```

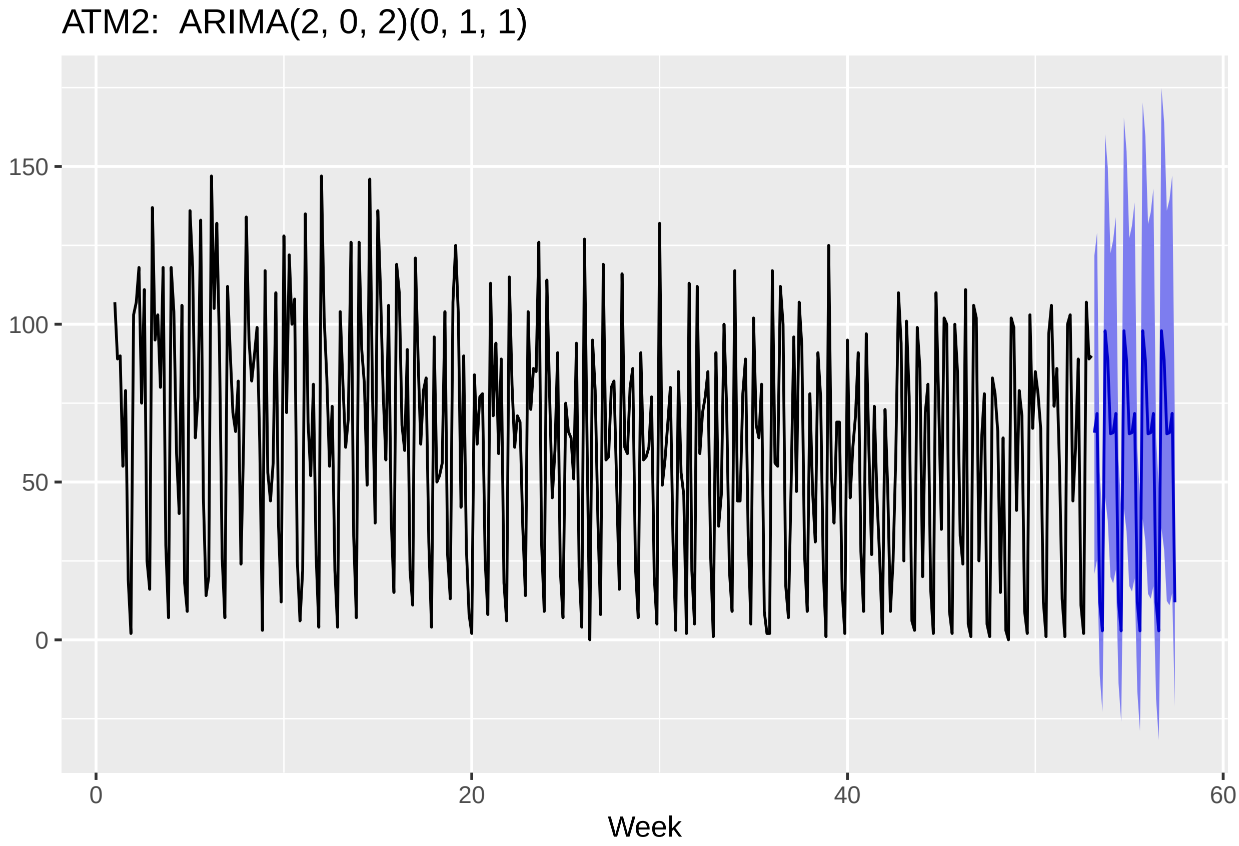
```
Box.test(resid(ATM2_fit), type = "L", lag = 7)
```

```
##
##  Box-Ljung test
##
## data:  resid(ATM2_fit)
## X-squared = 8.0186, df = 7, p-value = 0.331
```

Final observations on ATM2 withdrawals:
- Based on Box-Ljung test, p-value = 0.331 > 0.05, which suggest that residuals might be White Noise.
- The histogram shows that the residuals are more or less normally distributed aroud mean of 0.
- The AIC value is 2535.57 - Cant's comment about autocorrelation.

The forecast for ATM2 is as follows:

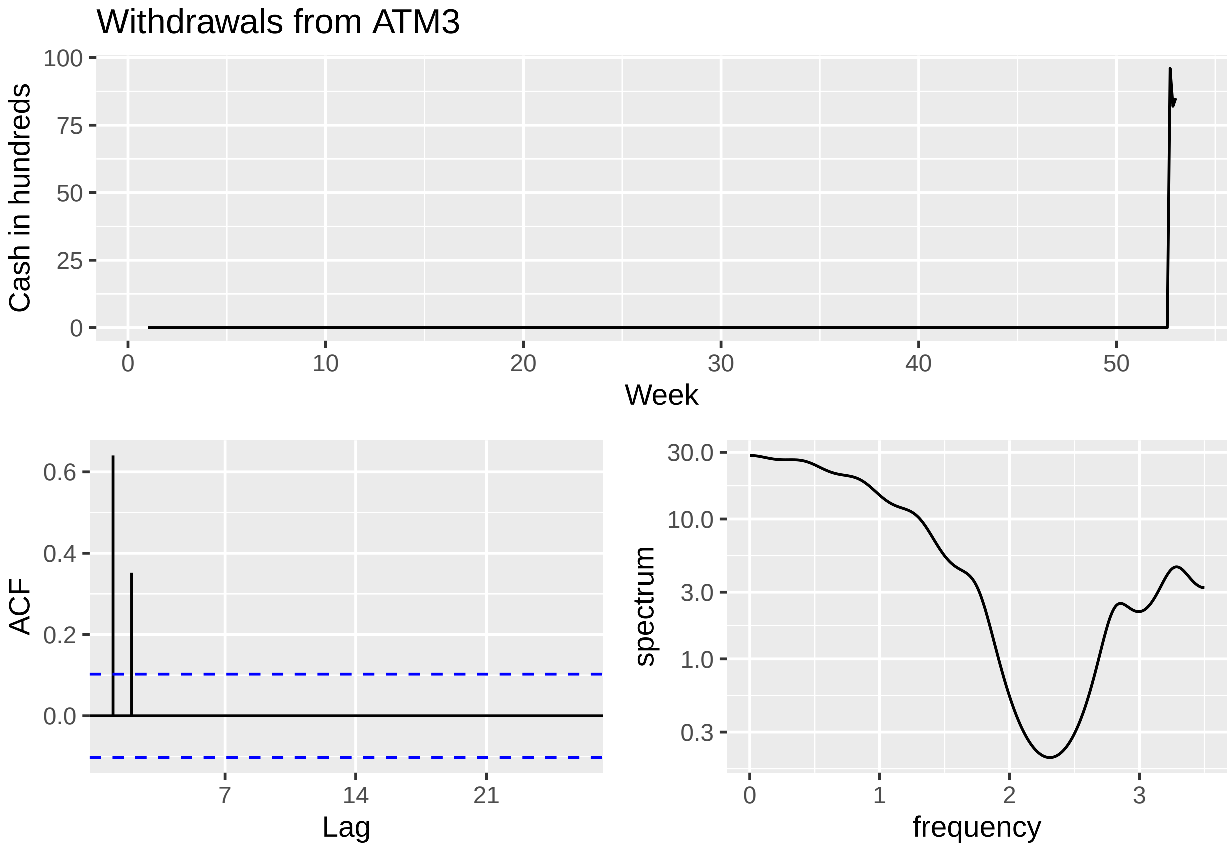
```
ATM2_forecast <- forecast(ATM2_fit, 31, level = 95)
autoplot(ATM2_forecast) + labs(title = "ATM2:  ARIMA(2, 0, 2)(0, 1, 1)", x = "Week", y = NULL) + theme(
```



ATM3 withdrawals:

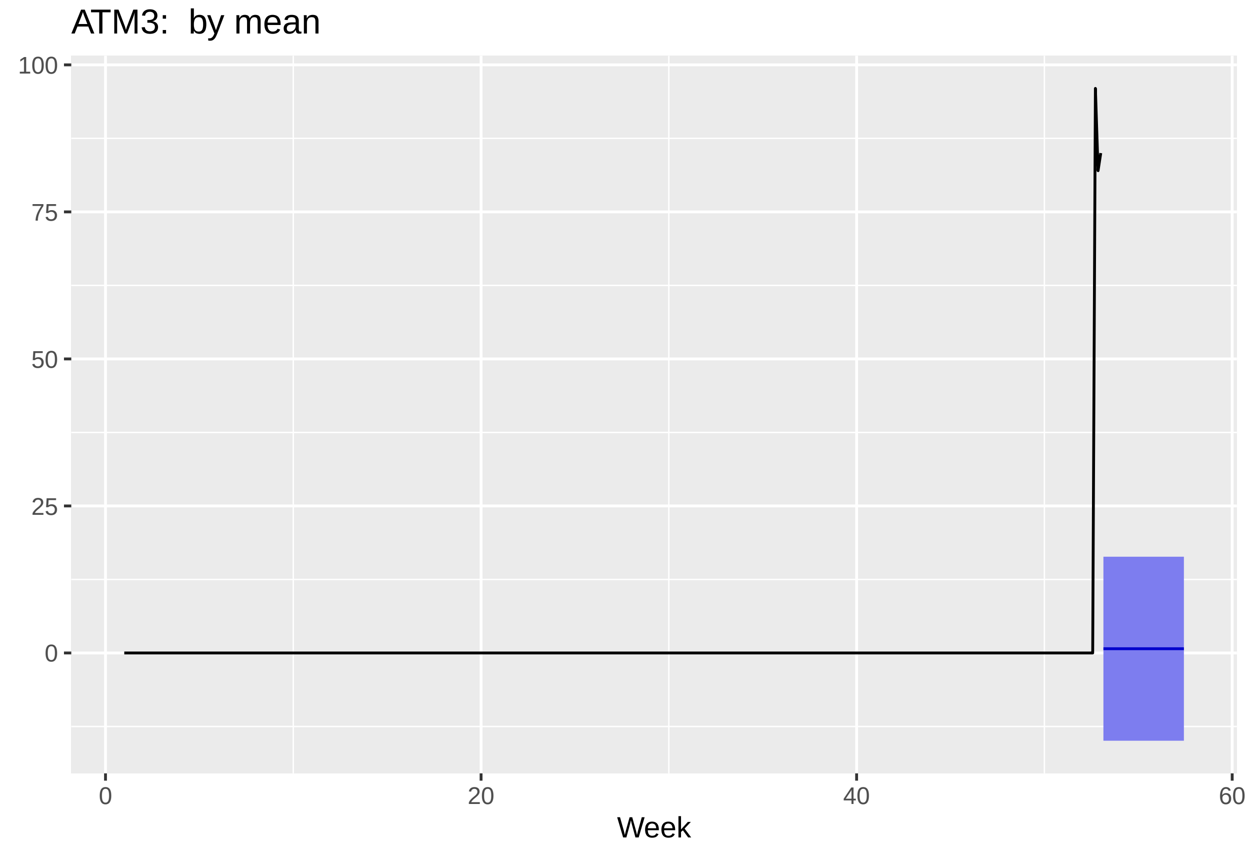
We'll observe below that there is hardly any data in ATM3. So, it might be a good idea to find the mean of the data.

```
ATM3 <- filter(atm_data, ATM == "ATM3")
ATM3_ts <- ts(ATM3$Cash, frequency = 7)
ATM3_ts %>% ggtsdisplay(points = FALSE, plot.type = "spectrum", main = "Withdrawals from ATM3", xlab =
```



The forecast for ATM3 is as follows:

```
ATM3_forecast <- meanf(ATM3_ts, 31, level = 95)
autoplot(ATM3_forecast) + labs(title = "ATM3:  by mean", x = "Week", y = NULL) + theme(legend.position
```



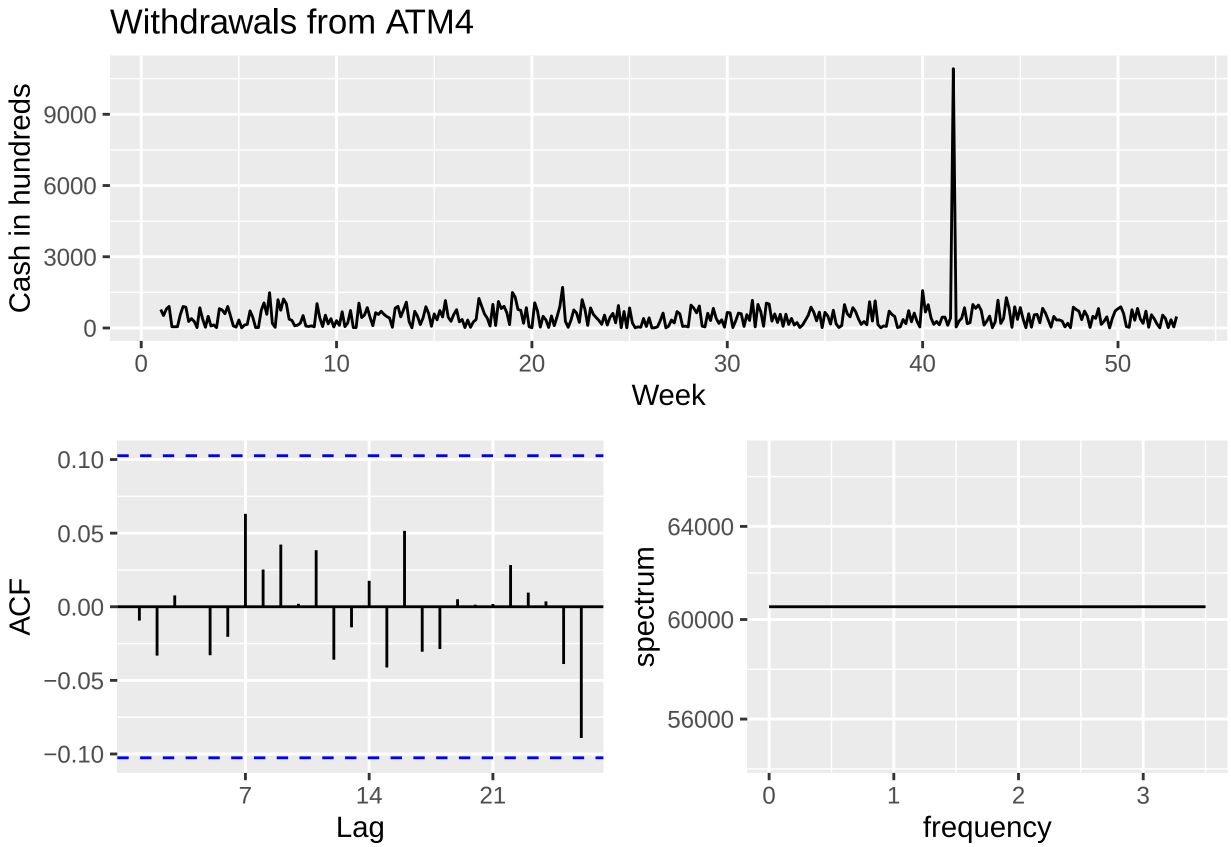
ATM4 withdrawals:

We'll do similar exercise for ATM4.

In the ATM4 too, there was considerable amount of withdrawal activity. So, ATM4 is also a candidate for forecasting.
There is one spike at lag 7. That's abou it. There's no conspicuous seasonality, but could be weekly seasonal.

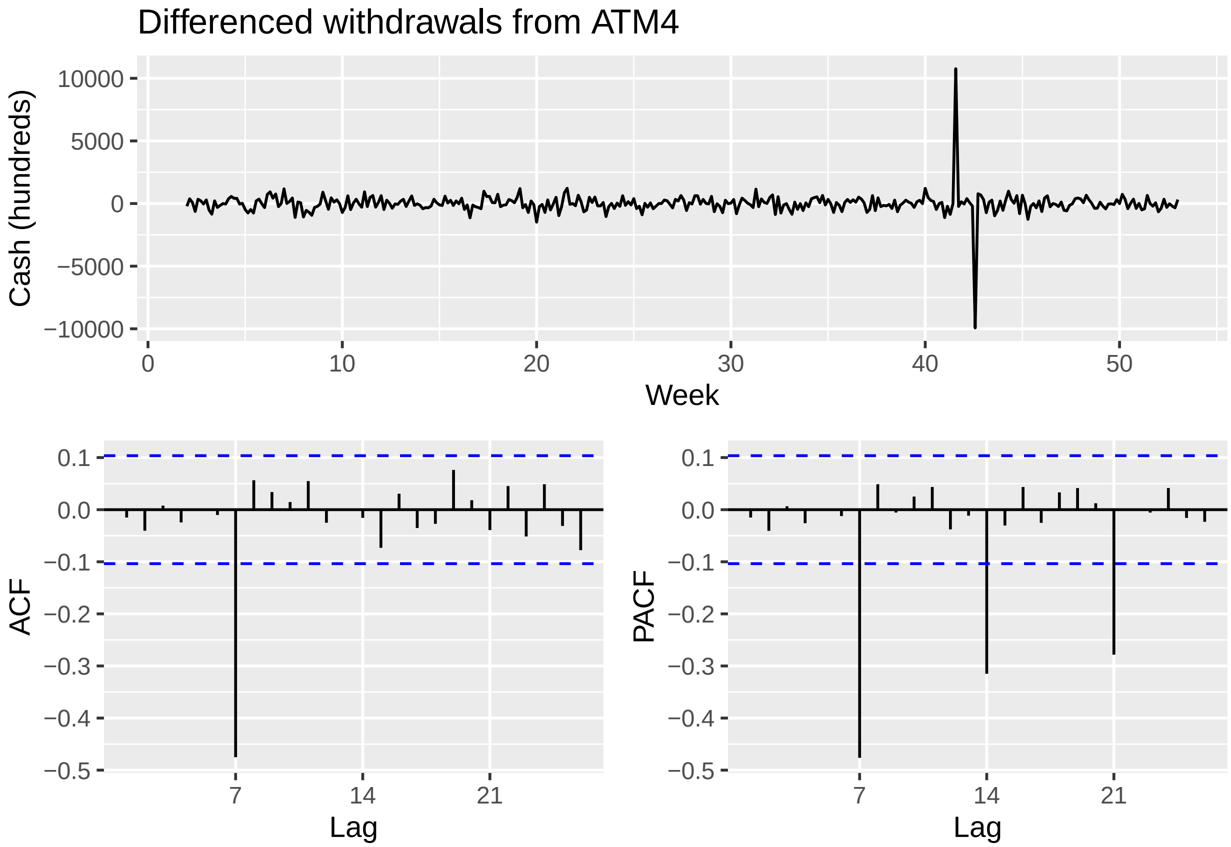
So, ACF and spectrum plots suggest a seasonal Arima model.

```
ATM4 <- filter(atm_data, ATM == "ATM4")
ATM4_ts <- ts(ATM4$Cash, frequency = 7)
ATM4_ts %>% ggtsdisplay(points = FALSE, plot.type = "spectrum", main = "Withdrawals from ATM4", xlab =
```



The graph is not clear, but there could be weekly seasonality in ATM4, so, doesn't hurt to difference, it's being differenced right below.

```
diff(ATM4_ts, 7) %>% ggtsdisplay(points = FALSE, main = "Differenced withdrawals from ATM4", xlab = "We
```



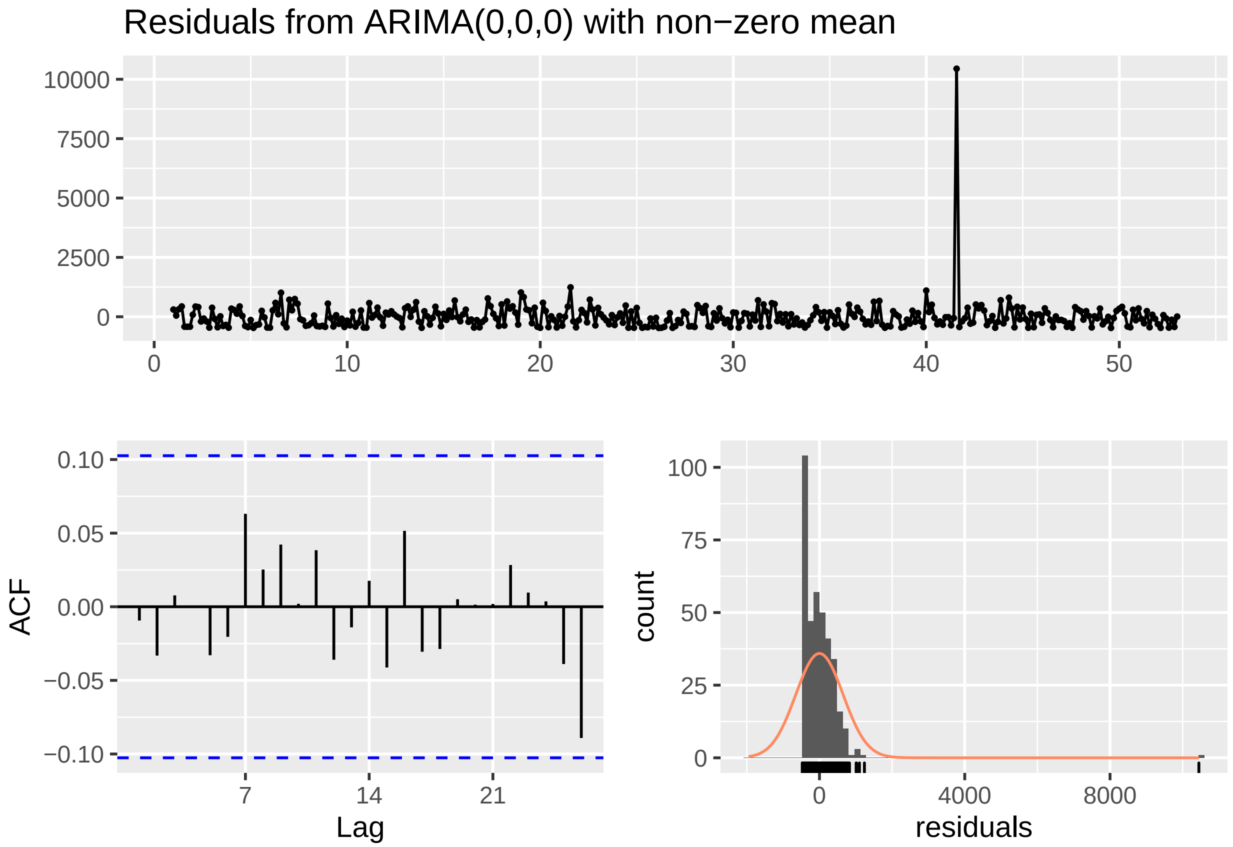
```
print(paste0("First differencing of ATM4_ts: ", ndiffs(ATM4_ts)))
```

```
## [1] "First differencing of ATM4_ts: 0"
```

In ATM4, we don't have the indicators for the values p, d, q, P, D, Q for ARIMA(p, d, q)(P, D, Q). So, we'll use function auto.arima().

auto.arima(ATM2_ts) gives us the values for ARIMA(p, d, q)(P, D, Q).

```
ATM4_fit <- auto.arima(ATM4_ts)
checkresiduals(ATM4_fit)
```

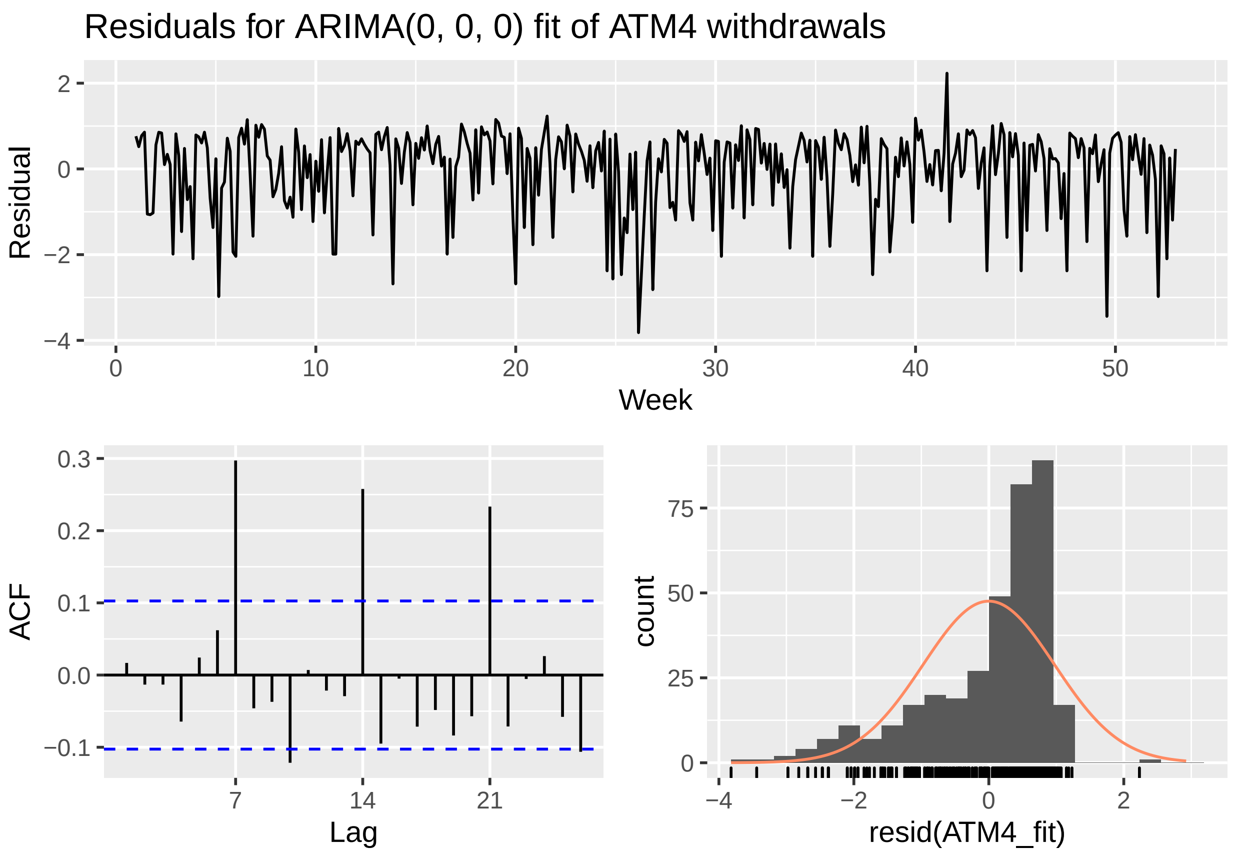


```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,0,0) with non-zero mean
## Q* = 4.6649, df = 13, p-value = 0.9818
##
## Model df: 1.   Total lags used: 14
```

auto.arima() determined ARIMA(0, 0, 0), which means White Noise, which means no autocorrelation (page 42 of printed book).

Now, we'll run Arima, only with parameter order (not use seasonal parameter).

```
ATM4_fit <- Arima(ATM4_ts, order = c(0, 0, 0), lambda = BoxCox.lambda(ATM4_ts))
ggtsdisplay(resid(ATM4_fit), points = FALSE, plot.type = "histogram", main = "Residuals for ARIMA(0, 0,
```



```r
# checkresiduals(ATM4_fit)
```

```r
ATM4_fit
```

```
## Series: ATM4_ts
## ARIMA(0,0,0) with non-zero mean
## Box Cox transformation: lambda= -0.07382441
##
## Coefficients:
##         mean
##       4.4952
## s.e.  0.0510
##
## sigma^2 estimated as 0.9502:  log likelihood=-508.1
## AIC=1020.2   AICc=1020.23   BIC=1028
```

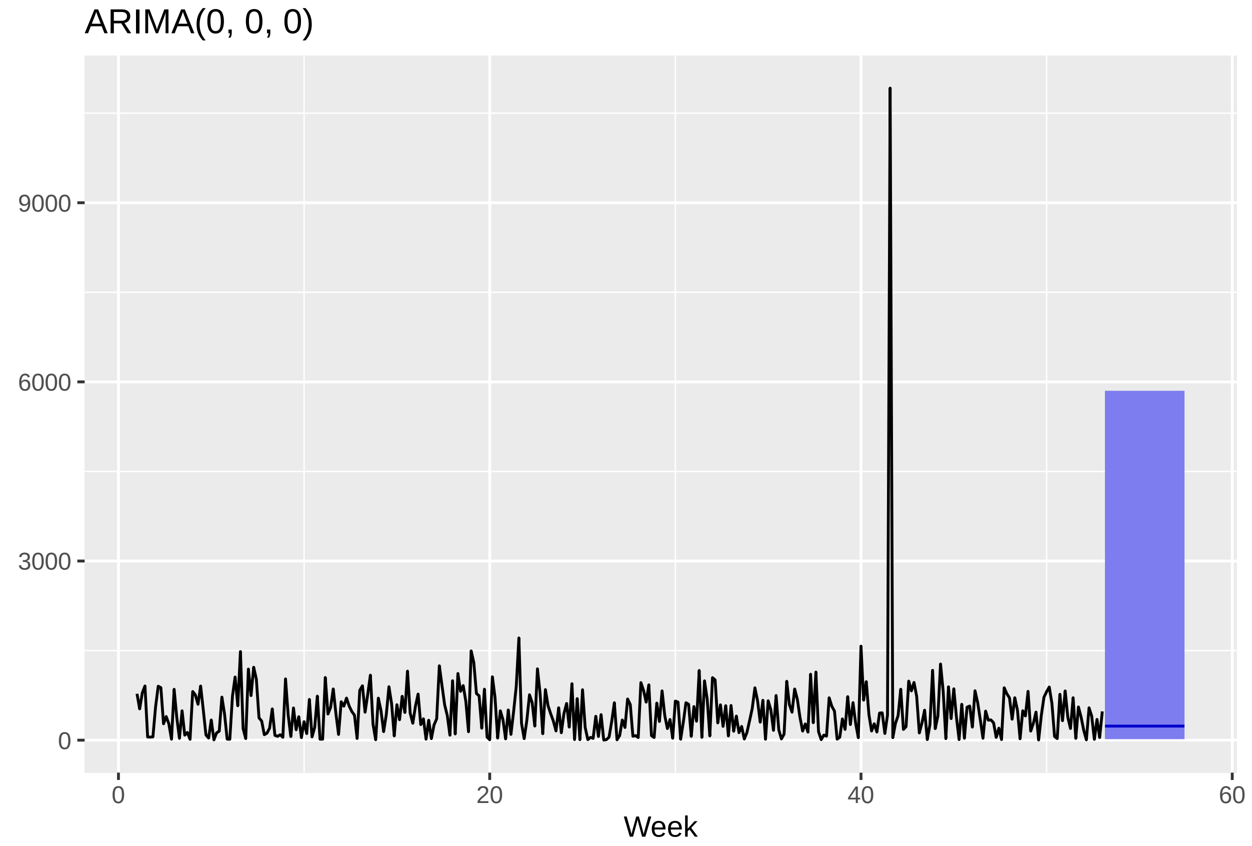
```r
Box.test(resid(ATM4_fit), type = "L", lag = 7)
```

```
##
##  Box-Ljung test
##
## data:  resid(ATM4_fit)
## X-squared = 36.486, df = 7, p-value = 5.869e-06
```

Final observations on ATM2 withdrawals:
- Based on Box-Ljung test, p-value = 0.331 > 0.05, which suggest that residuals might be White Noise.
- The histogram shows that the residuals are more or less normally distributed aroud mean of 0.
- The AIC value is 949.04. - Cant's comment about autocorrelation.

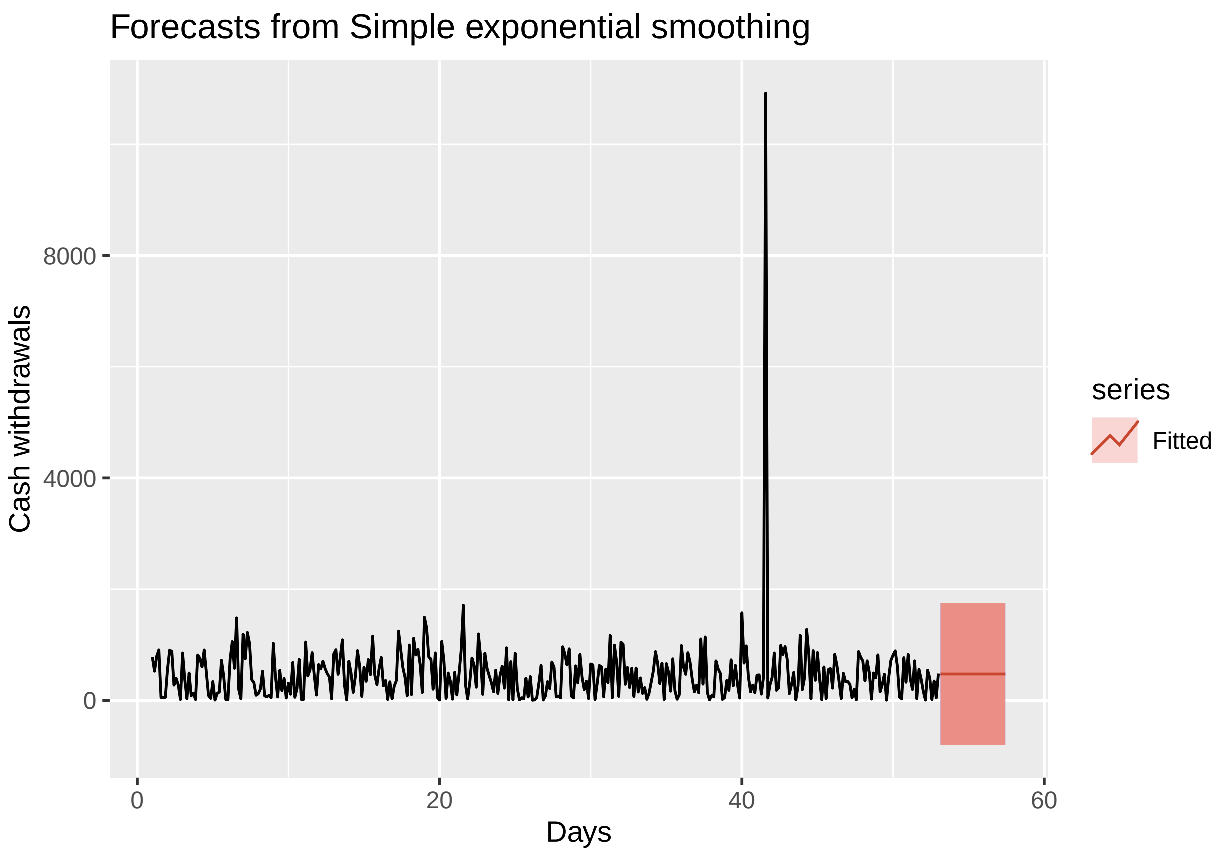
The forecast for ATM4 is as follows:

```
ATM4_forecast <- forecast(ATM4_fit, 31, level = 95)
autoplot(ATM4_forecast) + labs(title = "ARIMA(0, 0, 0)", x = "Week", y = NULL) + theme(legend.position =
```



I don't feel sure about the forecasting, so it might be good idea to try ses forecasitng.

```
# ATM4_forecast_ses <- ses(ATM4_ts, 31, level = 95)
# autoplot(ATM4_forecast_ses) + autolayer( (ATM4_forecast_ses), series = "Fitted") + xlab("Days") + yla
autoplot(ses(ATM4_ts, 31, level = 95)) + autolayer( (ses(ATM4_ts, 31, level = 95)), series = "Fitted") +
```



Ses forecasting doesn't make a perceptible diffrence.

So, I'll go by the ARIMA forecasting.

Consilodation of forecasts into output spread sheet.

Collecting dates for the month of May 2010, followed by binding of ATM forecasts.

```
may_2010_dates <- seq(ymd("2010-05-01"), ymd("2010-05-31"), by = "1 day")
#
ATM1_df <- data.frame("DATE" = may_2010_dates, "ATM" = c("ATM1"), "Cash" = c(ATM1_forecast$mean))
ATM2_df <- data.frame("DATE" = may_2010_dates, "ATM" = c("ATM2"), "Cash" = c(ATM2_forecast$mean))
ATM3_df <- data.frame("DATE" = may_2010_dates, "ATM" = c("ATM3"), "Cash" = c(ATM3_forecast$mean))
ATM4_df <- data.frame("DATE" = may_2010_dates, "ATM" = c("ATM4"), "Cash" = c(ATM4_forecast$mean))
all_ATM_df <- rbind(ATM1_df, ATM2_df, ATM3_df, ATM4_df)
write.csv(all_ATM_df, "./ATM624Data2-Forecasts.csv")
```

Part B – Forecasting Power, ResidentialCustomerForecastLoad-624.xlsx

Part B consists of a simple dataset of residential power usage for January 1998 until December 2013. Your assignment is to model these data and a monthly forecast for 2014. The data is given in a single file. The variable 'KWH' is power consumption in Kilowatt hours, the rest is straight forward. Add this to your existing files above.

Reading data from ATM624Data2.csv.

```
customer_data <- readxl::read_excel('./ResidentialCustomerForecastLoad-624.xlsx')
```

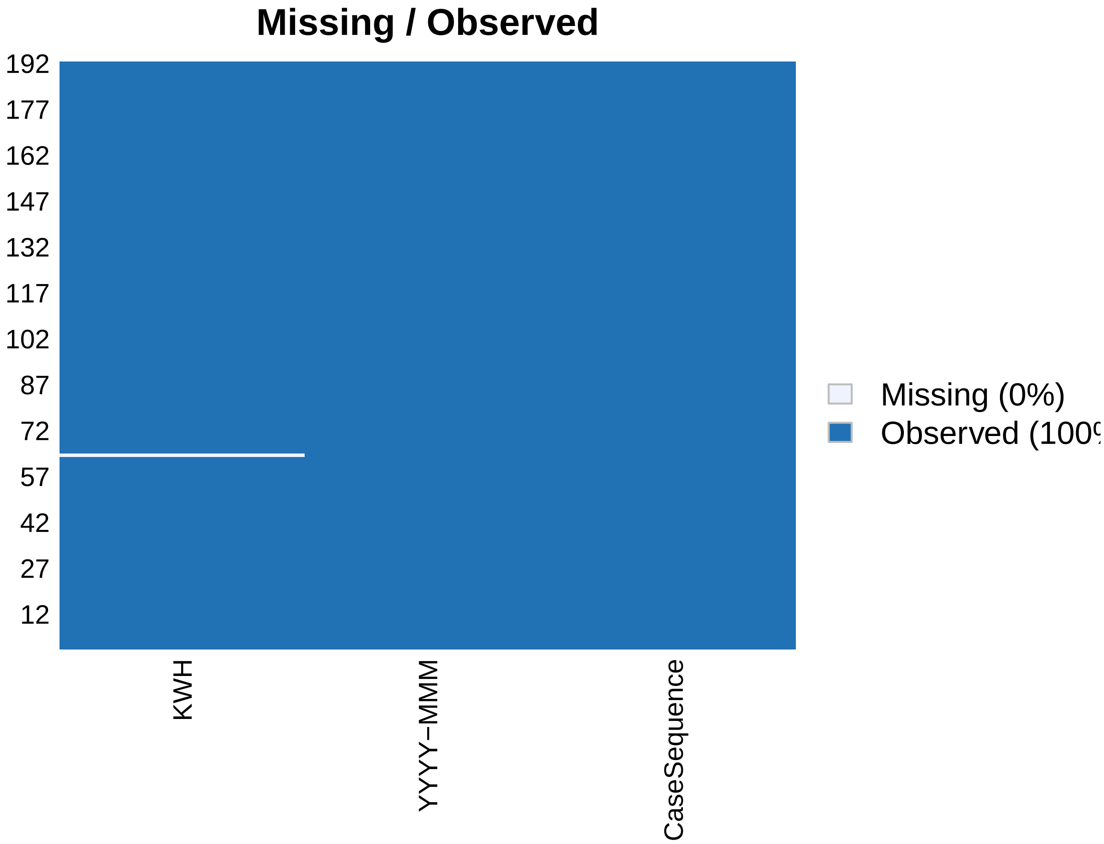
```
print(paste0("Number of rows = ", nrow(customer_data)))
```

```
## [1] "Number of rows = 192"
```

The missmap() function tells me that there are missing values in data, but it's not clear and detailed.

```
customer_data %>% missmap(main = "Missing / Observed")
```

```
## Warning: Unknown or uninitialised column: `arguments`.

## Warning: Unknown or uninitialised column: `arguments`.

## Warning: Unknown or uninitialised column: `imputations`.
```



By taking a summary, I observe that there is one 1 missing KWH value.

```
customer_data %>% summary()
```

```
##   CaseSequence     YYYY-MMM              KWH
##  Min.   :733.0   Length:192         Min.   :  770523
##  1st Qu.:780.8   Class :character   1st Qu.: 5429912
##  Median :828.5   Mode  :character   Median : 6283324
##  Mean   :828.5                      Mean   : 6502475
##  3rd Qu.:876.2                      3rd Qu.: 7620524
##  Max.   :924.0                      Max.   :10655730
##                                     NA's   :1
```

Data sanitization.

Since it's only 1 case, it's not worth the effort to impute. So, I'll remove the row.

After dropping the NA-row, the number of rows reduce to 191.

```
customer_data <- customer_data %>% filter(!is.na(KWH))
dim(customer_data)
```

```
## [1] 191   3
```

At this point, the data is sanitized, and we can form time series, visualize and try to make forecasts.

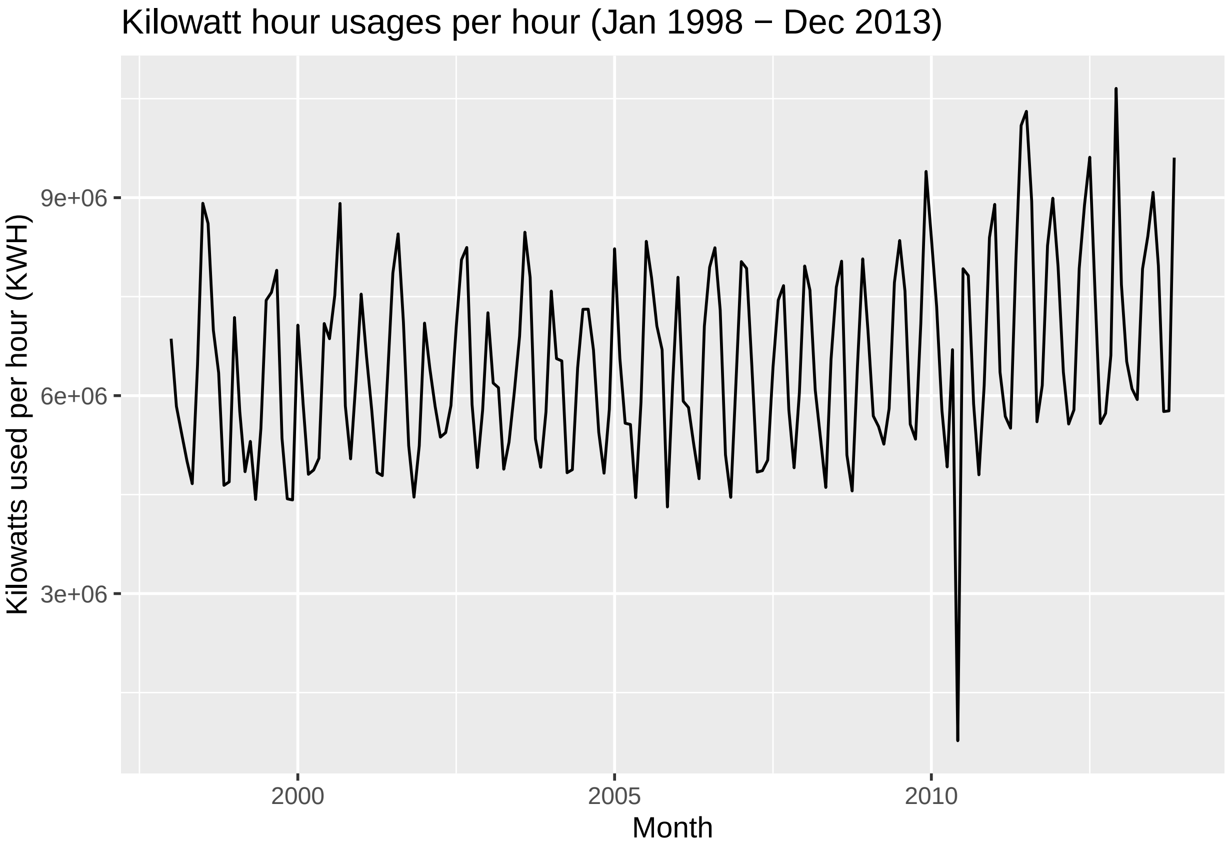
A first look at the data with head(), which also returns the column names.

```
head(customer_data)
```

```
## # A tibble: 6 x 3
##   CaseSequence `YYYY-MMM`     KWH
##          <dbl> <chr>        <dbl>
## 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
```

A first look at the time series, showing KWH usages, by month.

```
customer_ts <- ts(customer_data[, "KWH"], start=c(1998,1), frequency = 12)
autoplot(customer_ts, main = "Kilowatt hour usages per hour (Jan 1998 - Dec 2013)") + xlab("Month") + yl
```

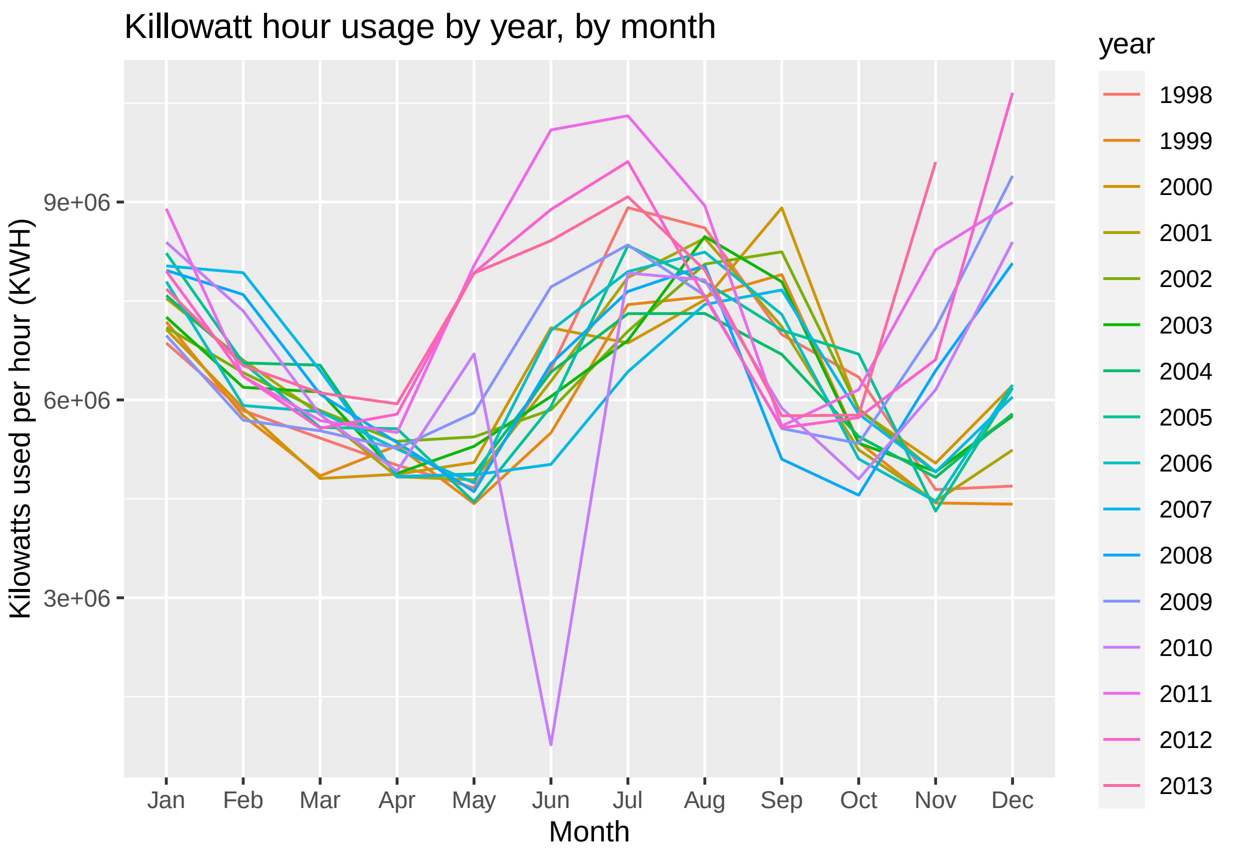


Observations:
- KWH usage is annually seasonal, if not semi-annually.
- An obvious outlier in 2010.
- Slight upward trend after 2010.

At this point, I like to check the graph of year, to explore any hidden details. I'll use ggseasonal, which we visited in chapter 2.

By breaking down by year, it shows an approximately sinusoidal graphs, with a period of one year, which gives a slight clue of the pattern, in following year.

```
ggseasonplot(customer_ts) + ggtitle("Killowatt hour usage by year, by month") + xlab("Month") + ylab("Ki
```



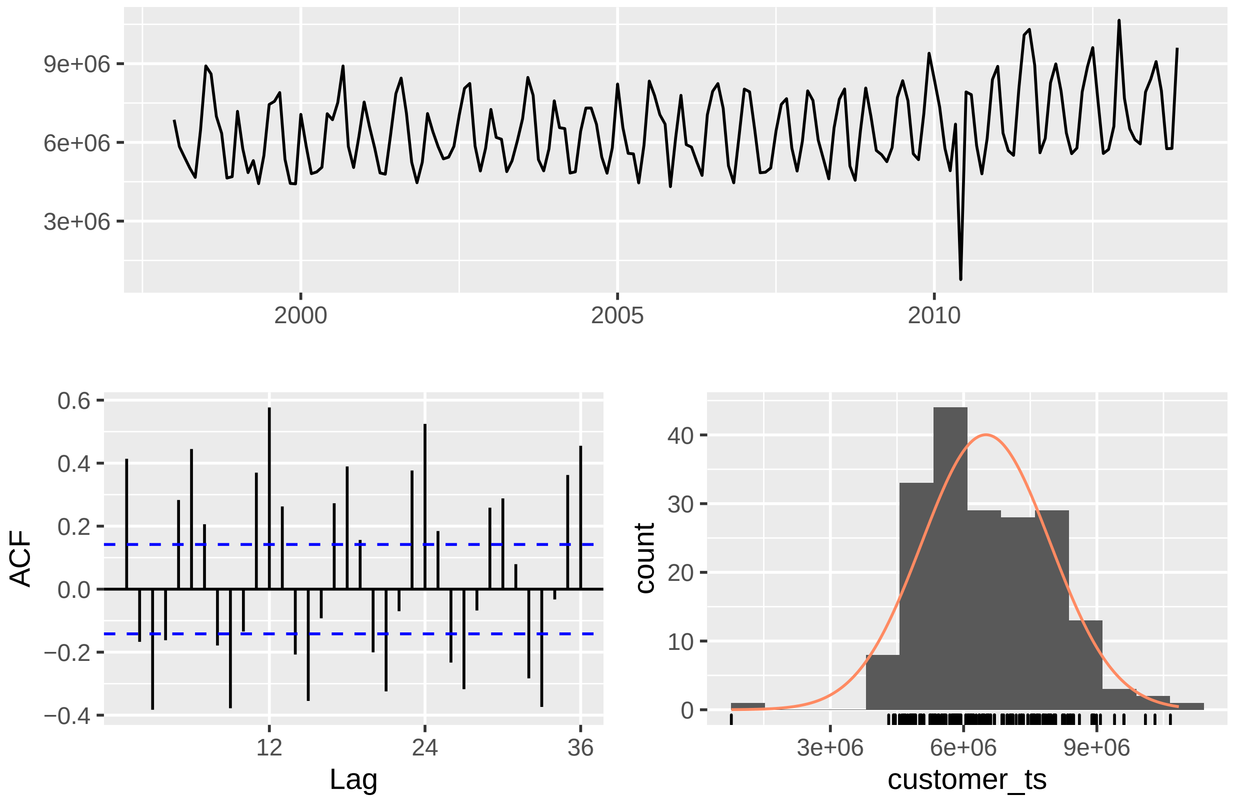
```r
ndiffs(customer_ts)
```

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

We'll display the details with ggtsdisplay().

ACF shows seasonality and the histogram is not quite normal. So, BoxCox transformation may be required.

```r
ggtsdisplay(customer_ts, points = FALSE, plot.type = "histogram")
```



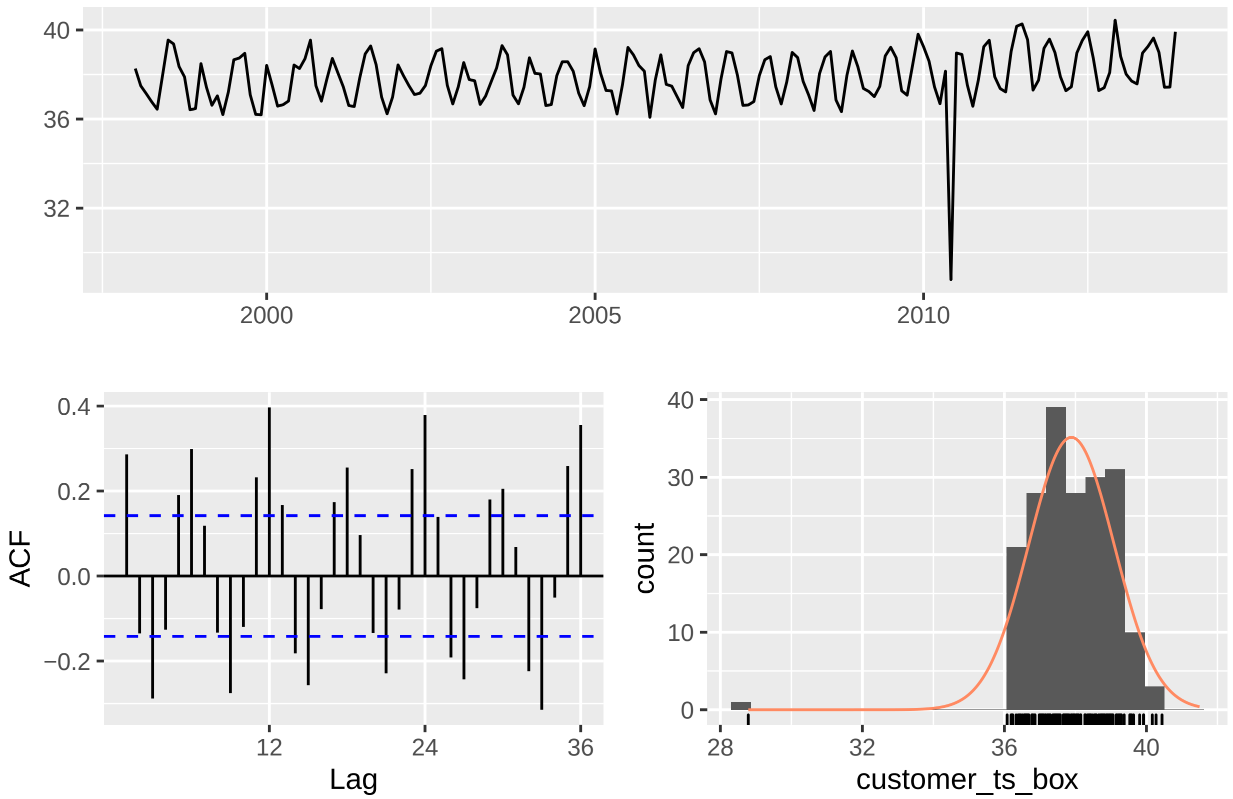
The tentative BoxCox transformation in the following.

```
customer_ts_lambda <- BoxCox.lambda(customer_ts)
customer_ts_lambda
```

```
## [1] 0.06361748
```

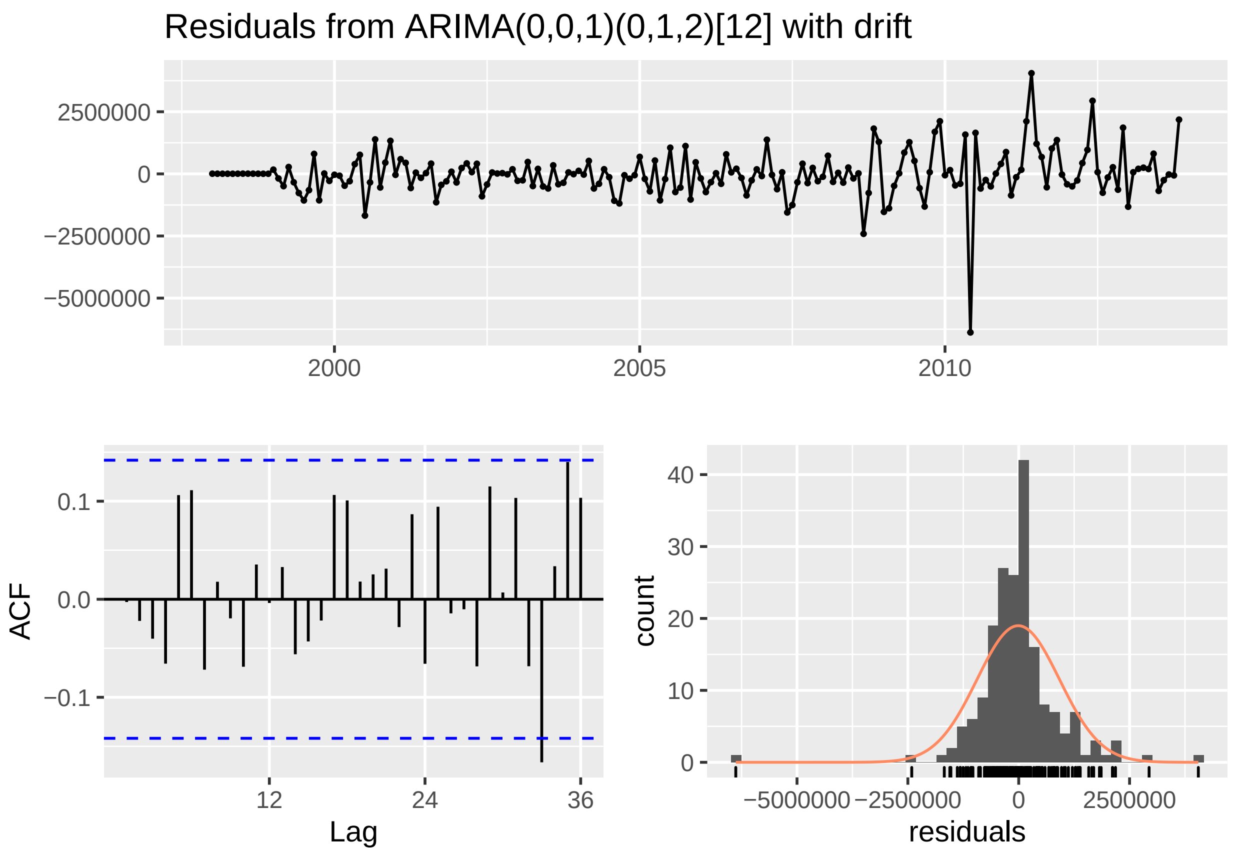
Applying BoxCox transformation to normalize the histogram. This is a test. If histogram is significantly normalized, then I'll use BoxCox transformation.

```
customer_ts_box <- customer_ts %>% BoxCox(lambda = 0.1)  # rounded 0.06361748 to 0.1
ggtsdisplay(customer_ts_box, points = FALSE, plot.type = "histogram")
```



The histogram is slightly normalized, but not quite significant. So, BoxCox transformation may not be appropriate here. But I may revisit, depending on other investigations.

```
customer_ts_fit <- auto.arima(customer_ts)
checkresiduals(customer_ts_fit)
```

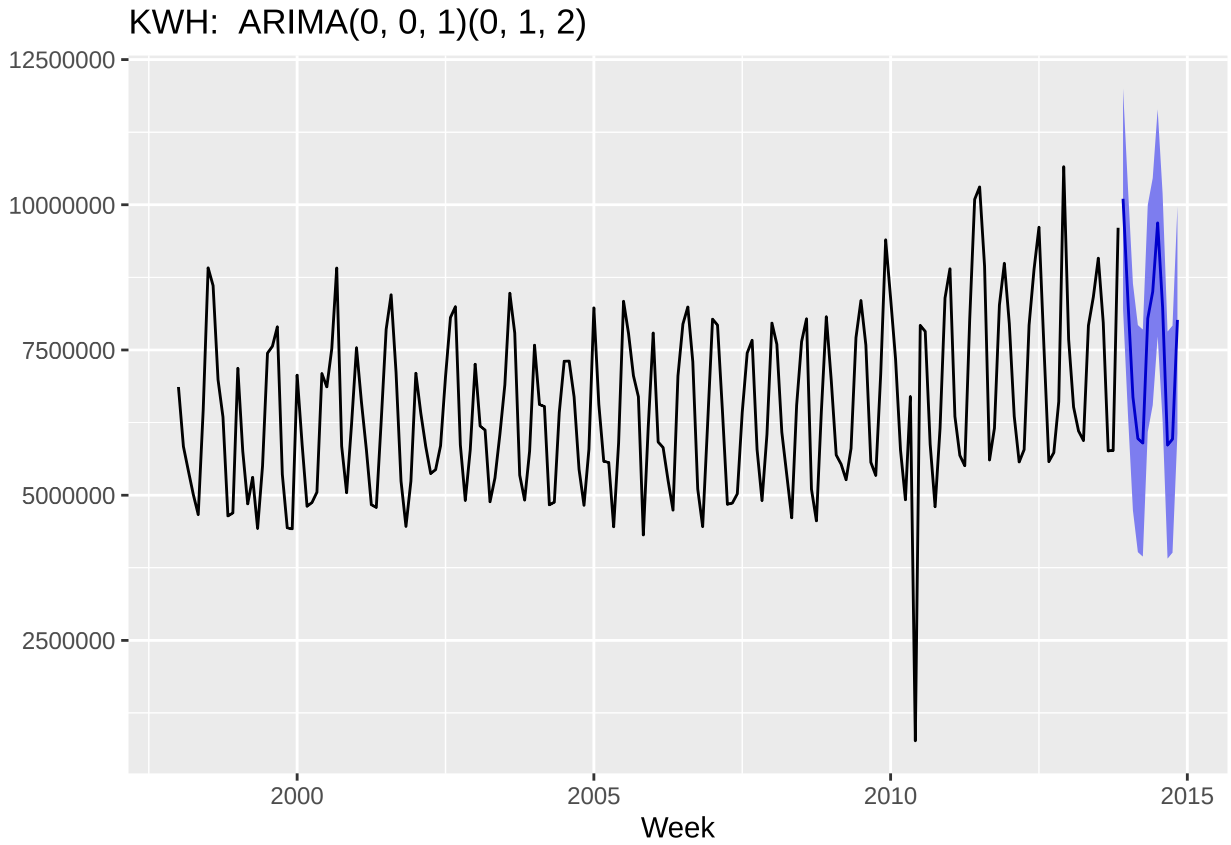


```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,0,1)(0,1,2)[12] with drift
## Q* = 17.494, df = 20, p-value = 0.6207
##
## Model df: 4.   Total lags used: 24
```

Residuals histogram of non-BoxCoxed series is more normal. So, I'll discard BoxCox.

I'll use auto.arima() for prediction.

```
customer_ts_forecast <- forecast(customer_ts_fit, 12, level = 95)
autoplot(customer_ts_forecast) + labs(title = "KWH:  ARIMA(0, 0, 1)(0, 1, 2)", x = "Week", y = NULL) +
```



Consilodation of forecasts into output spread sheet.

Writing monthly forecasts of 2014 in Excel.

```
max_seq <- max(customer_data$CaseSequence)

customer_dates <- format(seq(ymd("2014-01-01"), ymd("2014-12-31"), by = "1 month"), "%Y-%b")

customer_ts_forecast_df <- data.frame("CaseSequence" = c(max_seq + 1, max_seq + 2, max_seq + 3, max_seq

write.xlsx(customer_ts_forecast_df, "ResidentialCustomerForecastLoad-624-Forecasts.xlsx")
```

Part C – BONUS, optional (part or all), Waterflow_Pipe1.xlsx and Waterflow_Pipe2.xlsx

Part C consists of two data sets. These are simple 2 columns set, however they have different time stamps. Your optional assignment is to time-base sequence the data and aggregate based on hour (example of what this looks like, follows). Note for multiple recordings within an hour, take the mean. Then to determine if the data is stationary and can it be forecast. If so, provide a week forward forecast and present results via Rpubs and. rmd and the forecast in an Excel readable file.

Reading data from Waterflow__Pipe1__mod.xlsx and Waterflow__Pipe2__mod.xlsx.

In order to make the data formats compatible wiht R, I modified the date formats in the given files to 'yyyy-mm-dd hh:mm'. Original date format was 'mm/dd/yy hh:mm AM' or 'mm/dd/yy hh:mm PM'.
I retained data in original files Waterflow__Pipe1 and Waterflow__Pipe2 intact, but created new files Waterflow__Pipe1__mod and Waterflow__Pipe2__mod with new date formats.

```
wp1 <- readxl::read_excel('./Waterflow_Pipe1_mod.xlsx')
wp2 <- readxl::read_excel('./Waterflow_Pipe2_mod.xlsx')
```

Housekeeping of the files: Managing spaces in column names, taking houly means in wp1, join wp1 and wp2.

Since the given column names have spaces, I am changing them to names without spaces.

```
names(wp1) <- c("DateTime", "WaterFlow")
names(wp2) <- c("DateTime", "WaterFlow")
```

```
head(wp1)
```

```
## # A tibble: 6 x 2
##   DateTime            WaterFlow
##   <dttm>                  <dbl>
## 1 2015-10-23 00:24:06     23.4
## 2 2015-10-23 00:40:02     28.0
## 3 2015-10-23 00:53:51     23.1
## 4 2015-10-23 00:55:40     30.0
## 5 2015-10-23 01:19:17      6.00
## 6 2015-10-23 01:23:58     15.9
```

```
head(wp2)
```

```
## # A tibble: 6 x 2
##   DateTime            WaterFlow
##   <dttm>                  <dbl>
## 1 2015-10-23 01:00:00      18.8
## 2 2015-10-23 02:00:00      43.1
## 3 2015-10-23 03:00:00      38.0
## 4 2015-10-23 04:00:00      36.1
## 5 2015-10-23 05:00:00      31.9
## 6 2015-10-23 06:00:00      28.2
```

Since the periodicity of the readings of wp1 and wp2 are different, I'll normalize them. While seconds pipe reads every hour, the first one reads more that once within the hour. So, for first pipe, I'll for takes means of each hour.

```
wp1 <- wp1 %>% mutate(Date = date(DateTime), Hour = hour(DateTime) + 1)
wp1 <- wp1 %>% group_by(Date, Hour) %>% summarise_at(vars(WaterFlow), list(name = mean)) %>% mutate(Date

wp1 <- data.frame(wp1) %>% select(DateTime, WaterFlow = name)
wp2 <- data.frame(wp2)
```

head(wp1) shows the hourly averages.

```
nrow(wp1)
```

```
## [1] 236
```

```
head(wp1)
```

```
##              DateTime WaterFlow
## 1 2015-10-23 01:00:00  26.10280
## 2 2015-10-23 02:00:00  18.85202
## 3 2015-10-23 03:00:00  15.15857
## 4 2015-10-23 04:00:00  23.07886
## 5 2015-10-23 05:00:00  15.48219
## 6 2015-10-23 06:00:00  22.72539
```

```
nrow(wp2)
```

```
## [1] 1000
```

```
head(wp2)
```

```
##              DateTime WaterFlow
## 1 2015-10-23 01:00:00  18.81079
## 2 2015-10-23 02:00:00  43.08703
## 3 2015-10-23 03:00:00  37.98770
## 4 2015-10-23 04:00:00  36.12038
## 5 2015-10-23 05:00:00  31.85126
## 6 2015-10-23 06:00:00  28.23809
```

Join wp1 and wp2. In the join, I am adding waterflows through both of the pipes. Adding makes sense because after normalizing, each data in each pipe is an hour of waterflow.

```
water_data <- full_join(wp1, wp2, by = "DateTime", suffix = c("_1", "_2")) %>% mutate(WaterFlow_1 = ifel
  mutate(WaterFlow = WaterFlow_1 + WaterFlow_2) %>% select(DateTime, WaterFlow)

nrow(water_data)
```

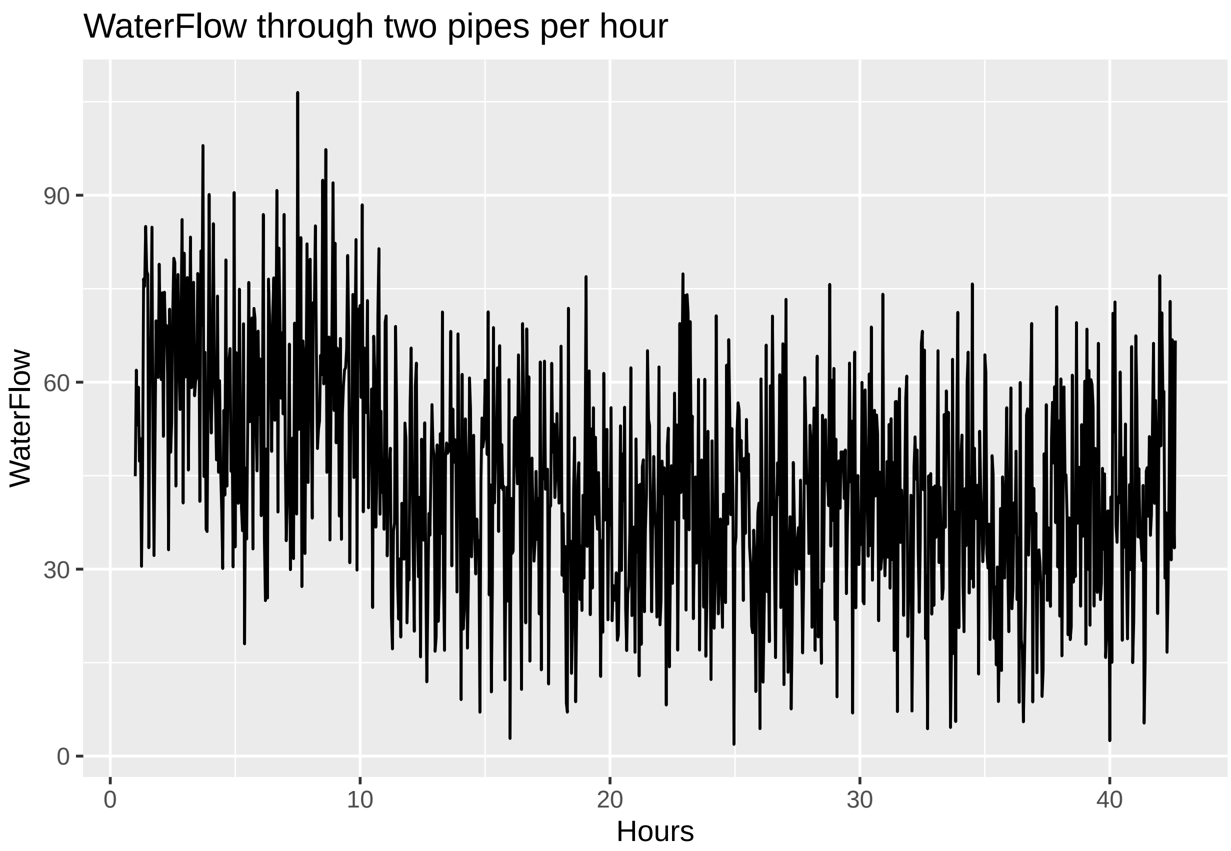
```
## [1] 1000
```

```
head(water_data)
```

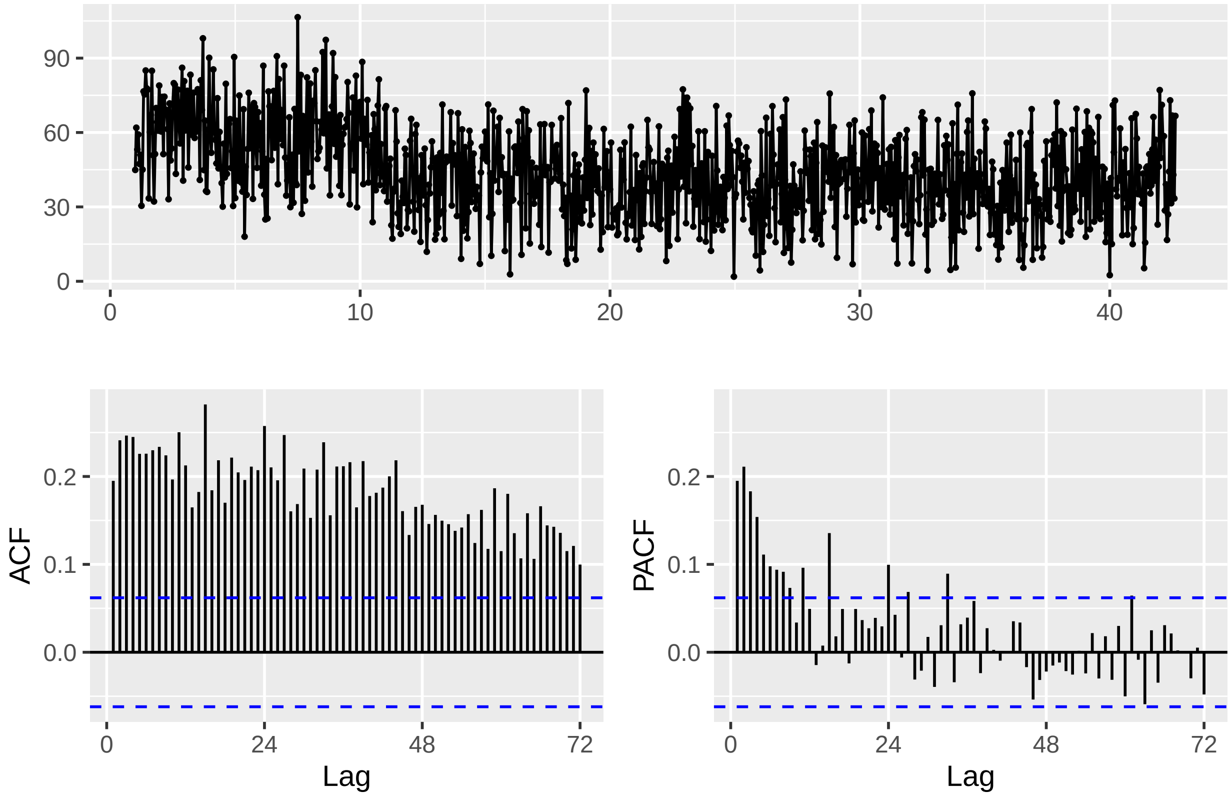
```
##              DateTime WaterFlow
## 1 2015-10-23 01:00:00  44.91359
## 2 2015-10-23 02:00:00  61.93904
## 3 2015-10-23 03:00:00  53.14627
## 4 2015-10-23 04:00:00  59.19923
## 5 2015-10-23 05:00:00  47.33345
## 6 2015-10-23 06:00:00  50.96348
```

Now, we are done with house keeping of data, and in a position to convert into time series.

```
water_data_ts <-ts(water_data$WaterFlow, frequency = 24)
autoplot(water_data_ts) + ggtitle("WaterFlow through two pipes per hour") + xlab("Hours") + ylab("Water
```

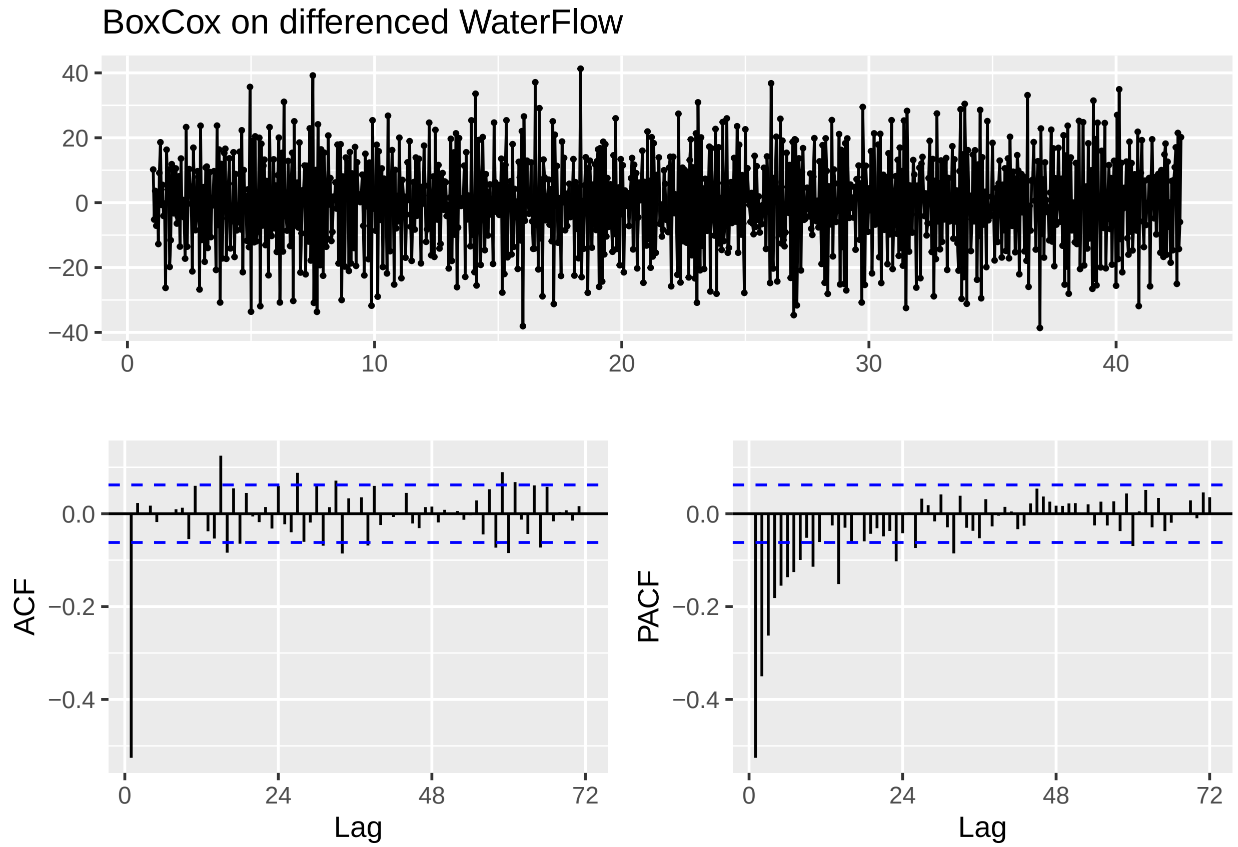


```
ggtsdisplay(water_data_ts)
```



Observation: The data is not stationary. So, we'll perform BoxCox with one lag of differencing.

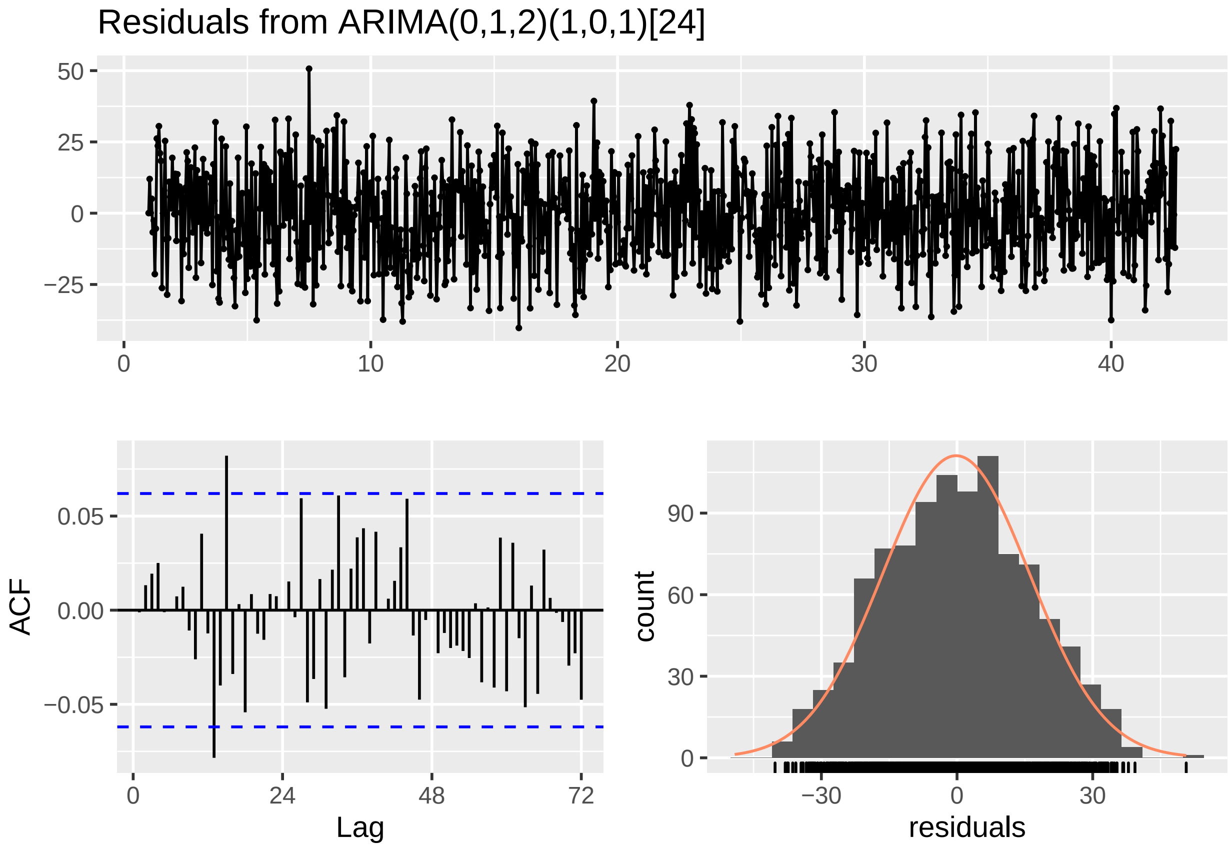
```
water_data_ts_box <- BoxCox(water_data_ts, BoxCox.lambda(water_data_ts))

ggtsdisplay(diff(water_data_ts_box), main = "BoxCox on differenced WaterFlow")
```



Observations:
- The time series is stationary.
- ACF has a big spike.
- No seasonality.

Now, I'll determine the ARIM coordinates, with auto.arima().

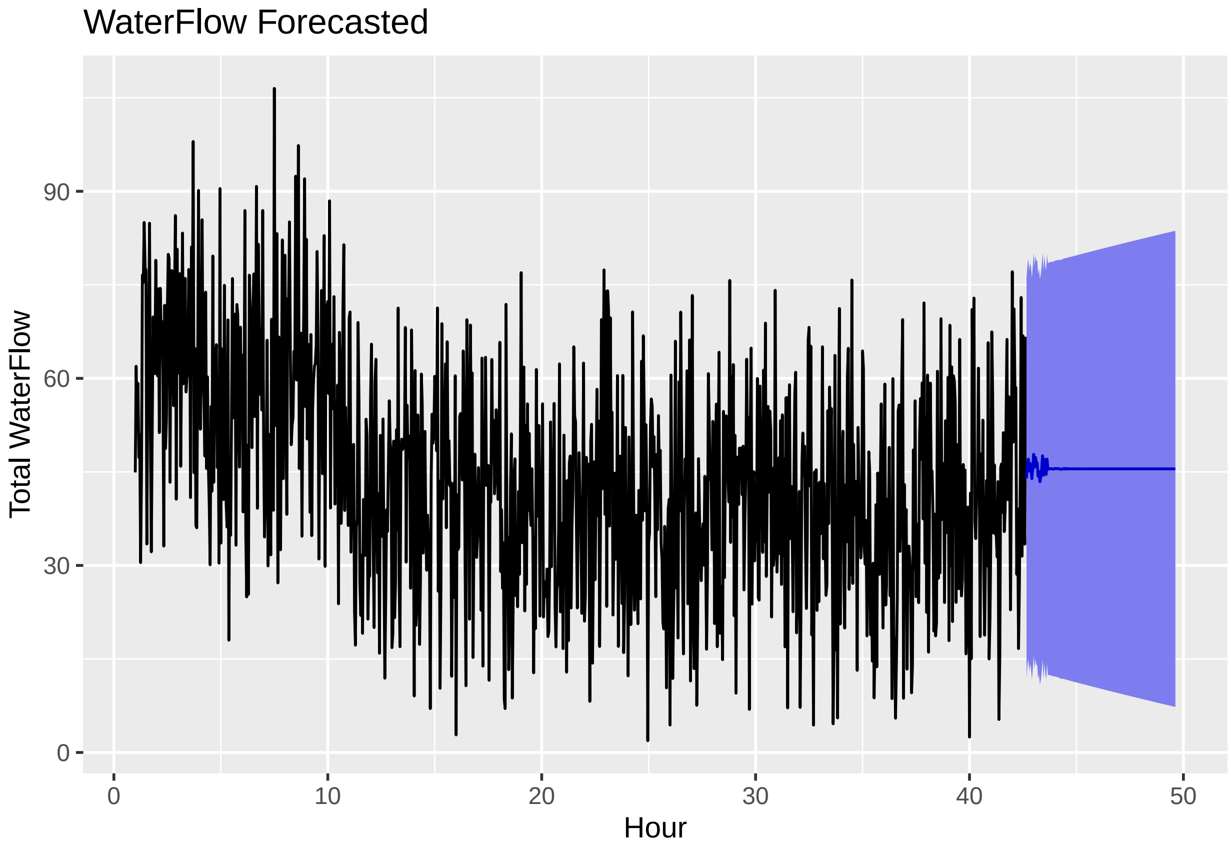
```
water_data_ts_autoarima_fit <- auto.arima(water_data_ts)
checkresiduals(water_data_ts_autoarima_fit)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,2)(1,0,1)[24]
## Q* = 53.956, df = 44, p-value = 0.1445
##
## Model df: 4.   Total lags used: 48
```

The forecast.

```
water_data_ts_autoarima_forecast <- forecast(water_data_ts_autoarima_fit, 24 * 7, level = 95)

autoplot(water_data_ts_autoarima_forecast) + ggtitle("WaterFlow Forecasted") + xlab("Hour") + ylab("Tot
```



# Consilodation of forecasts into output spread sheet.

Writing WaterFlow forecasts in Excel.

```
water_data_ts_forecast_df <- data.frame(DateTime = max(water_data$DateTime) + hours(1:168), WaterFlow =
write.xlsx(water_data_ts_forecast_df, "Waterflow_Pipe-Forecasts.xlsx")
```