

Data624 - Project1

Amit Kapoor

3/28/2021

Contents

Overview	1
Part A - ATM Forecast	1
Exploratory Analysis	1
Data Cleaning	5
Time Series	6
Part B - Forecasting Power	29
Part C - Waterflow Pipe	30

Overview

This project includes 3 time series dataset and requires to select best forecasting model for all 3 datasets.

- Part A - ATM Forecast
- Part B - Forecasting Power
- Part C - Waterflow Pipe

Part A - ATM Forecast

The dataset contains cash withdrawals from 4 different ATM machines from May 2009 to Apr 2010. The variable 'Cash' is provided in hundreds of dollars and data is in a single file. Before starting our analysis we will first download the excel from github and then read it through read_excel.

Exploratory Analysis

```
temp.file <- tempfile(fileext = ".xlsx")
download.file(url="https://github.com/amit-kapoor/data624/blob/main/Project1/ATM624Data.xlsx?raw=true",
             destfile = temp.file,
             mode = "wb",
             quiet = TRUE)
atm.data <- read_excel(temp.file, skip=0, col_types = c("date","text","numeric"))

glimpse(atm.data)

## Rows: 1,474
## Columns: 3
## $ DATE <dtm> 2009-05-01, 2009-05-01, 2009-05-02, 2009-05-02, 2009-05-03, 2009-
## $ ATM <chr> "ATM1", "ATM2", "ATM1", "ATM2", "ATM1", "ATM2", "ATM1", "ATM2", "~
```

```
## $ Cash <dbl> 96, 107, 82, 89, 85, 90, 90, 55, 99, 79, 88, 19, 8, 2, 104, 103, ~
```

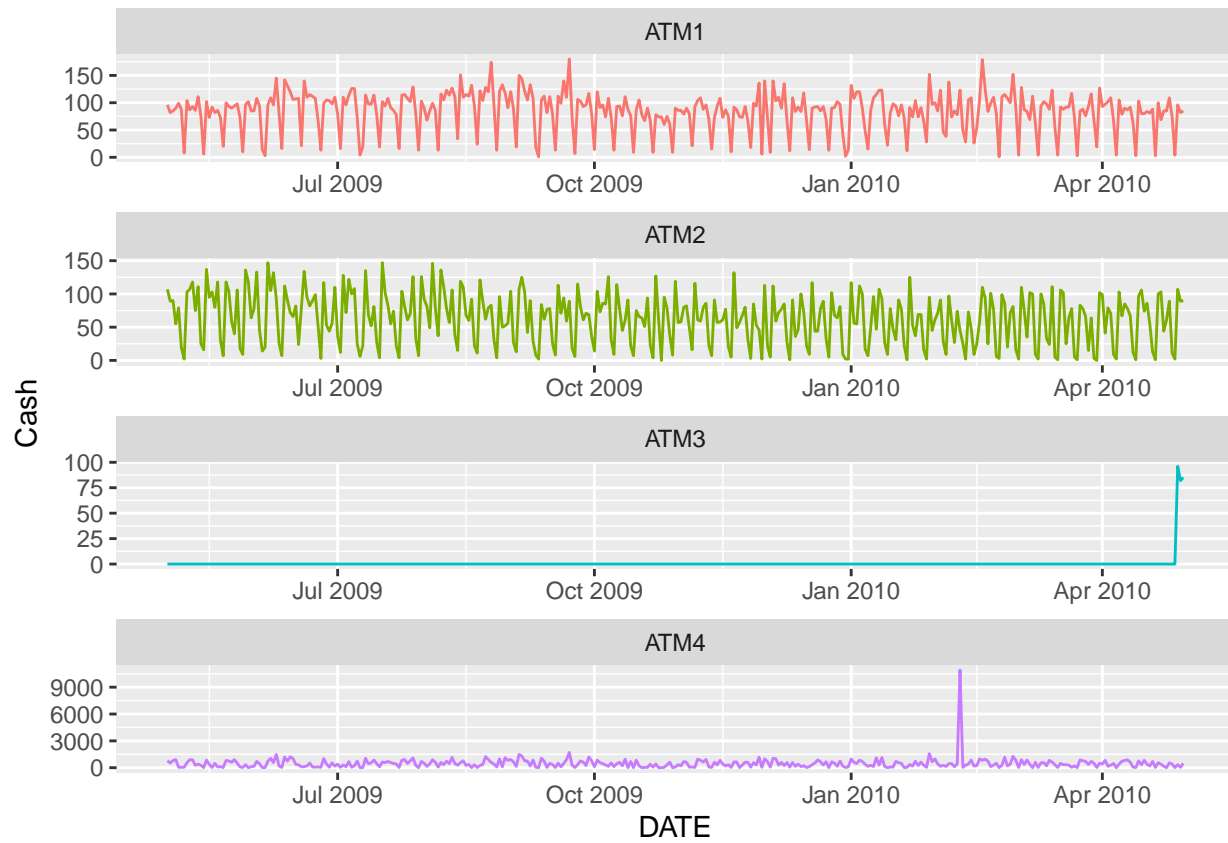
```
# rows missing values
```

```
atm.data[!complete.cases(atm.data),]
```

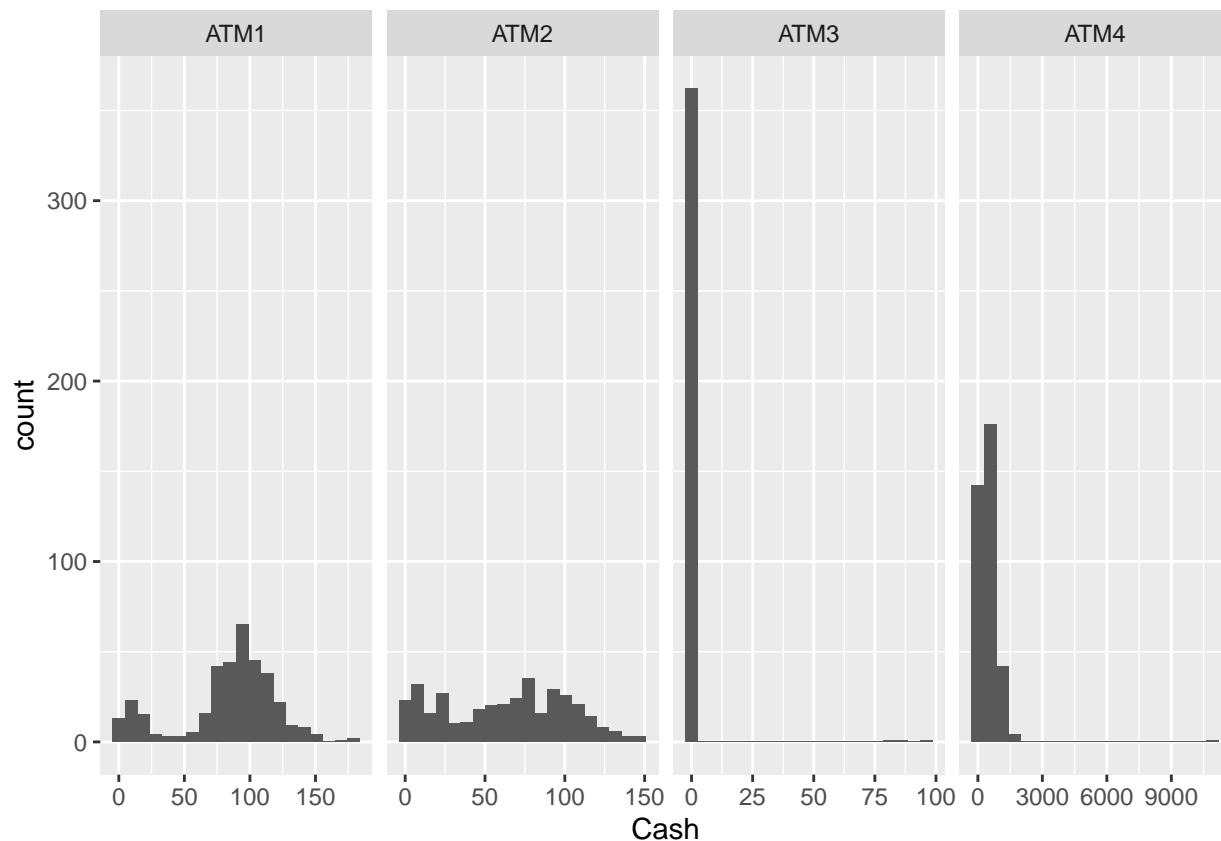
```
## # A tibble: 19 x 3
```

```
##   DATE           ATM    Cash
##   <dtm>         <chr> <dbl>
## 1 2009-06-13 00:00:00 ATM1     NA
## 2 2009-06-16 00:00:00 ATM1     NA
## 3 2009-06-18 00:00:00 ATM2     NA
## 4 2009-06-22 00:00:00 ATM1     NA
## 5 2009-06-24 00:00:00 ATM2     NA
## 6 2010-05-01 00:00:00 <NA>    NA
## 7 2010-05-02 00:00:00 <NA>    NA
## 8 2010-05-03 00:00:00 <NA>    NA
## 9 2010-05-04 00:00:00 <NA>    NA
## 10 2010-05-05 00:00:00 <NA>    NA
## 11 2010-05-06 00:00:00 <NA>    NA
## 12 2010-05-07 00:00:00 <NA>    NA
## 13 2010-05-08 00:00:00 <NA>    NA
## 14 2010-05-09 00:00:00 <NA>    NA
## 15 2010-05-10 00:00:00 <NA>    NA
## 16 2010-05-11 00:00:00 <NA>    NA
## 17 2010-05-12 00:00:00 <NA>    NA
## 18 2010-05-13 00:00:00 <NA>    NA
## 19 2010-05-14 00:00:00 <NA>    NA
```

```
ggplot(atm.data[complete.cases(atm.data),] , aes(x=DATE, y=Cash, col=ATM )) +  
  geom_line(show.legend = FALSE) +  
  facet_wrap(~ATM, ncol=1, scales = "free")
```



```
ggplot(atm.data[complete.cases(atm.data),] , aes(x=Cash )) +
  geom_histogram(bins=20) +
  facet_grid(cols=vars(ATM), scales = "free")
```



```
# consider complete cases
atm.comp <- atm.data[complete.cases(atm.data),]
# pivot wider with cols from 4 ATMs and their values as Cash
atm.comp <- atm.comp %>% pivot_wider(names_from = ATM, values_from = Cash)
head(atm.comp)
```

```
## # A tibble: 6 x 5
##   DATE                ATM1  ATM2  ATM3  ATM4
##   <dtm>              <dbl> <dbl> <dbl> <dbl>
## 1 2009-05-01 00:00:00    96   107    0 777.
## 2 2009-05-02 00:00:00    82    89    0 524.
## 3 2009-05-03 00:00:00    85    90    0 793.
## 4 2009-05-04 00:00:00    90    55    0 908.
## 5 2009-05-05 00:00:00    99    79    0 52.8
## 6 2009-05-06 00:00:00    88    19    0 52.2
```

```
# summary
atm.comp %>% select(-DATE) %>% summary()
```

```
##           ATM1           ATM2           ATM3           ATM4
##  Min.   : 1.00   Min.   : 0.00   Min.   : 0.0000   Min.   : 1.563
## 1st Qu.: 73.00   1st Qu.: 25.50   1st Qu.: 0.0000   1st Qu.: 124.334
## Median : 91.00   Median : 67.00   Median : 0.0000   Median : 403.839
## Mean   : 83.89   Mean   : 62.58   Mean   : 0.7206   Mean   : 474.043
## 3rd Qu.:108.00   3rd Qu.: 93.00   3rd Qu.: 0.0000   3rd Qu.: 704.507
## Max.   :180.00   Max.   :147.00   Max.   :96.0000   Max.   :10919.762
## NA's   :3       NA's   :2
```

Per above exploratory analysis, all ATMs show different patterns. We would perform forecasting for each

ATM separately.

- ATM1 and ATM2 shows similar pattern (approx.) throughout the time. ATM1 and ATM2 have 3 and 2 missing entries respectively.
- ATM3 appears to become online in last 3 days only and rest of days appears inactive. So the data available for this ATM is very limited.
- ATM4 requires replacement for outlier and we can assume that one day spike of cash withdrawal is unique. It has an outlier showing withdrawal amount 10920.

Data Cleaning

For this part we will first apply `ts()` function to get required time series. Next step is to apply `tsclean` function that will handle missing data along with outliers. To estimate missing values and outlier replacements, this function uses linear interpolation on the (possibly seasonally adjusted) series. Once we get the clean data we will use `pivot_longer` to get the dataframe in its original form.

```
atm.ts <- ts(atm.comp %>% select(-DATE))
head(atm.ts)
```

```
## Time Series:
## Start = 1
## End = 6
## Frequency = 1
##      ATM1 ATM2 ATM3      ATM4
## 1    96  107    0 776.99342
## 2    82   89    0 524.41796
## 3    85   90    0 792.81136
## 4    90   55    0 908.23846
## 5    99   79    0 52.83210
## 6    88   19    0 52.20845
```

```
# apply tsclean
atm.ts.cln <- sapply(X=atm.ts, tsclean)
atm.ts.cln %>% summary()
```

```
##      ATM1      ATM2      ATM3      ATM4
## Min.   : 1.00   Min.   : 0.00   Min.   : 0.0000   Min.   : 1.563
## 1st Qu.: 73.00  1st Qu.: 26.00  1st Qu.: 0.0000  1st Qu.: 124.334
## Median : 91.00  Median : 67.00  Median : 0.0000  Median : 402.770
## Mean   : 84.15  Mean   : 62.59  Mean   : 0.7206  Mean   : 444.757
## 3rd Qu.:108.00  3rd Qu.: 93.00  3rd Qu.: 0.0000  3rd Qu.: 704.192
## Max.   :180.00  Max.   :147.00  Max.   :96.0000  Max.   :1712.075
```

If we compare this summary with previous one of original data, ATM1 and ATM2 has no more NAs and ATM4 outlier value (10919.762) is handled and now the max value is 1712.075.

```
# convert into data frame, pivot longer , arrange by ATM and bind with dates
atm.new <- as.data.frame(atm.ts.cln) %>%
  pivot_longer(everything(), names_to = "ATM", values_to = "Cash") %>%
  arrange(ATM)

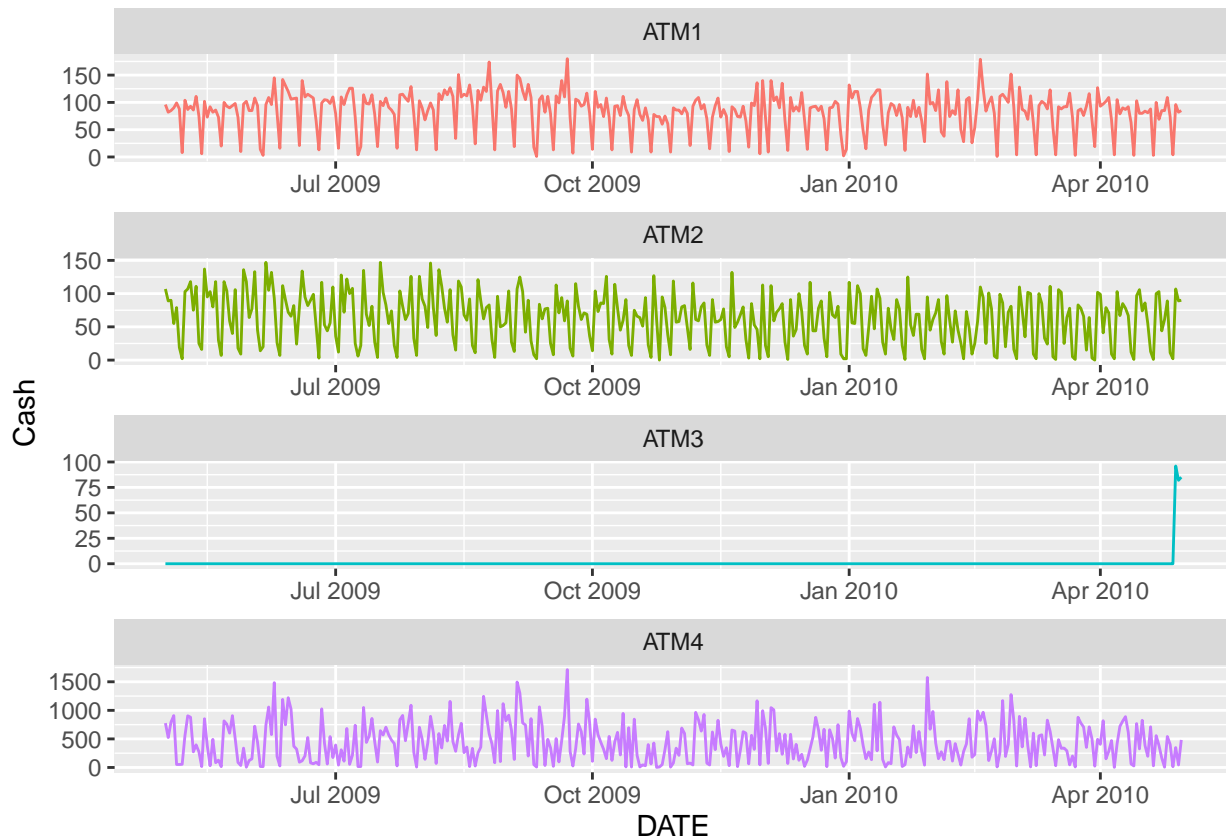
atm.new <- cbind(
  DATE = seq(as.Date("2009-05-1"), as.Date("2010-04-30"), length.out=365),
  atm.new)

head(atm.new)
```

```
##      DATE  ATM Cash
## 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
```

```
ggplot(atm.new , aes(x=DATE, y=Cash, col=ATM )) +
  geom_line(show.legend = FALSE) +
  facet_wrap(~ATM, ncol=1, scales = "free")
```



Though above plot doesn't show much differences for ATM1,2,3 but tsclean handled the ATM4 data very well after replacing the outlier.

Time Series

Function to plot forecast for various models.

```
# function to plot forecast(s)
atm.forecast <- function(timeseries) {
  # lambda value
  lambda <- BoxCox.lambda(timeseries)
  # models for forecast
  hw.model <- timeseries %>% hw(h=31, seasonal = "additive", lambda = lambda, damped = TRUE)
  ets.model <- timeseries %>% ets(lambda = lambda)
  arima.model <- timeseries %>% auto.arima(lambda = lambda)
  # forecast
  atm.hw.fcst <- forecast(hw.model, h=31)
  atm.ets.fcst <- forecast(ets.model, h=31)
  atm.arima.fcst <- forecast(arima.model, h=31)
```

```

# plot forecasts
p1 <- autoplot(timeseries) +
  autolayer(atm.hw.fcst, PI=FALSE, series="Holt-Winters") +
  autolayer(atm.ets.fcst, PI=FALSE, series="ETS") +
  autolayer(atm.arima.fcst, PI=FALSE, series="ARIMA") +
  theme(legend.position = "top") +
  ylab("Cash Withdrawl")
# zoom in plot
p2 <- p1 +
  labs(title = "Zoom in ") +
  xlim(c(51,56))

grid.arrange(p1,p2,ncol=1)
}

```

Function to calculate RMSEs for various models.

```

model_accuracy <- function(timeseries, atm_num) {
  # lambda value
  lambda <- BoxCox.lambda(timeseries)

  # split the data to train and test
  train <- window(timeseries, end=c(40, 3))
  test <- window(timeseries, start=c(40, 4))

  # models for forecast
  hw.model <- train %>% hw(h=length(train), seasonal = "additive", lambda = lambda, damped = TRUE)
  ets.model <- train %>% ets(model='ANA', lambda = lambda)

  # Arima model
  if (atm_num == 1) {
    # for ATM1
    arima.model <- train %>% Arima(order=c(0,0,2),
                                   seasonal = c(0,1,1),
                                   lambda = lambda)
  } else if (atm_num == 2) {
    # for ATM2
    arima.model <- train %>% Arima(order=c(3,0,3),
                                   seasonal = c(0,1,1),
                                   include.drift = TRUE,
                                   lambda = lambda,
                                   biasadj = TRUE)
  } else {
    # for ATM4
    arima.model <- train %>% Arima(order=c(0,0,1),
                                   seasonal = c(2,0,0),
                                   lambda = lambda)
  }

  # forecast
  hw.frct = forecast(hw.model, h = length(test))$mean
  ets.frct = forecast(ets.model, h = length(test))$mean
  arima.frct = forecast(arima.model, h = length(test))$mean
}

```

```

# dataframe having rmse
rmse = data.frame(RMSE=cbind(accuracy(hw.frct, test)[,2],
                             accuracy(ets.frct, test)[,2],
                             accuracy(arima.frct, test)[,2]))

names(rmse) = c("Holt-Winters", "ETS", "ARIMA")
# display rmse
rmse
}

```

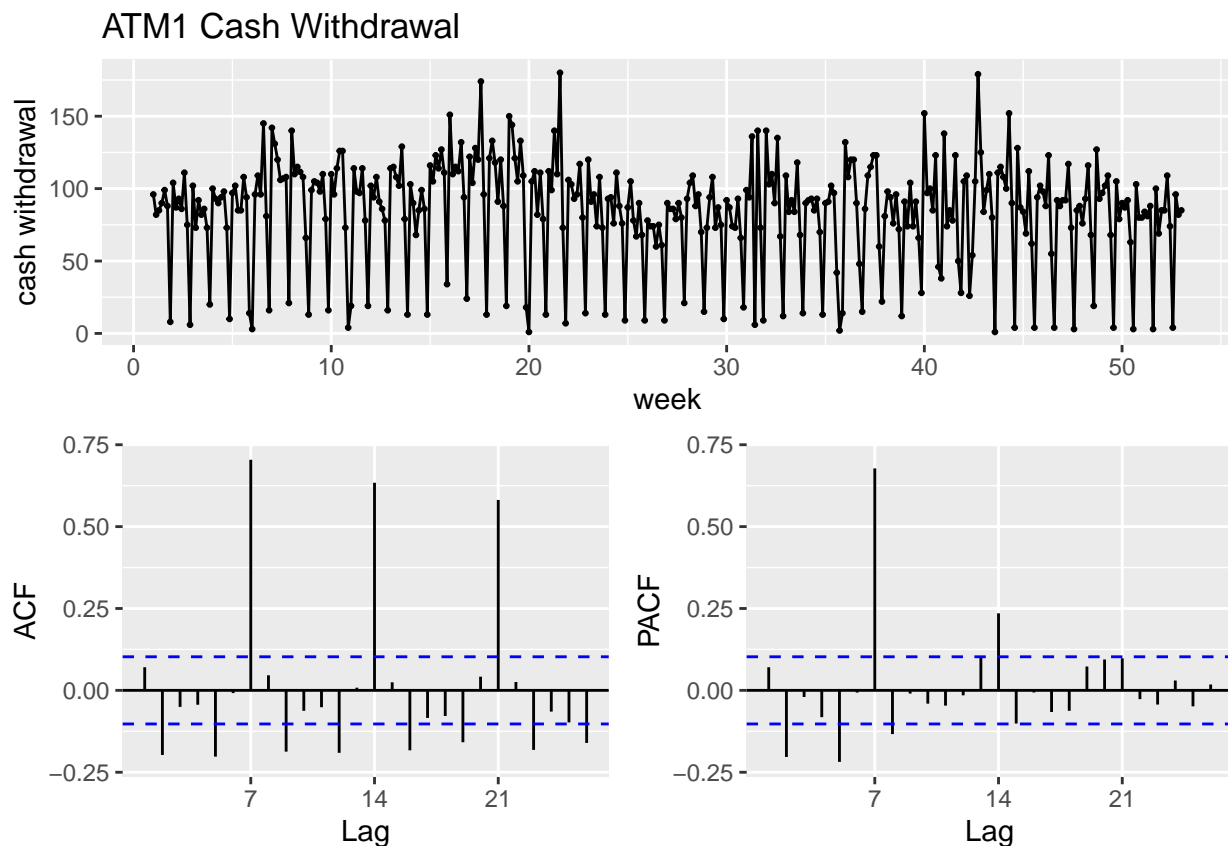
ATM1

Seeing the time series plot, it is clear that there is a seasonality in the data. We can see increasing and decreasing activities over the weeks in below plot. From the ACF plot, we can see a slight decrease in every 7th lag due to trend. PACF plot shows some significant lags at the beginning.

```

atm1.ts <- atm.new %>% filter(ATM=="ATM1") %>% select(Cash) %>% ts(frequency = 7)
ggtsdisplay(atm1.ts, main="ATM1 Cash Withdrawal", ylab="cash withdrawal", xlab="week")

```



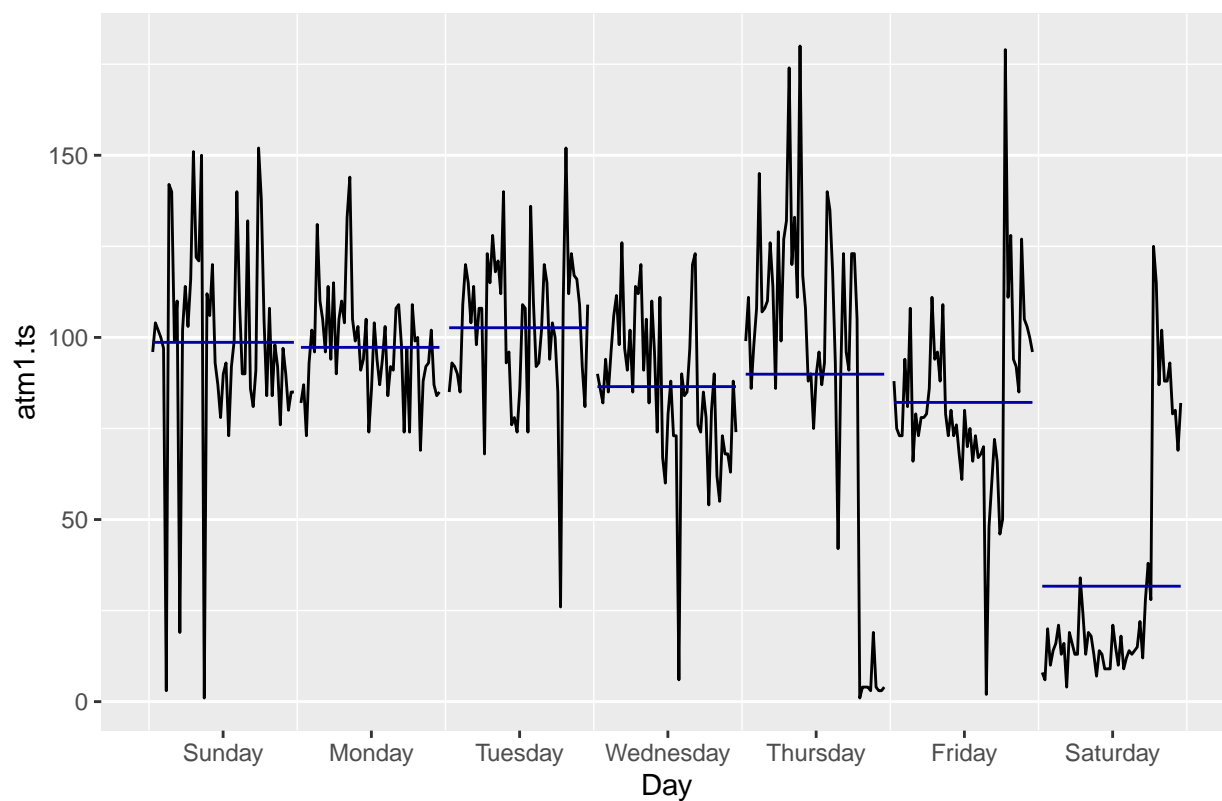
From the above plots it is evident that the time series is non stationary, showing seasonality and will require differencing to make it stationary.

```

ggsubseriesplot(atm1.ts, main="ATM1 Cash Withdrawal")

```


ATM1 Cash Withdrawal

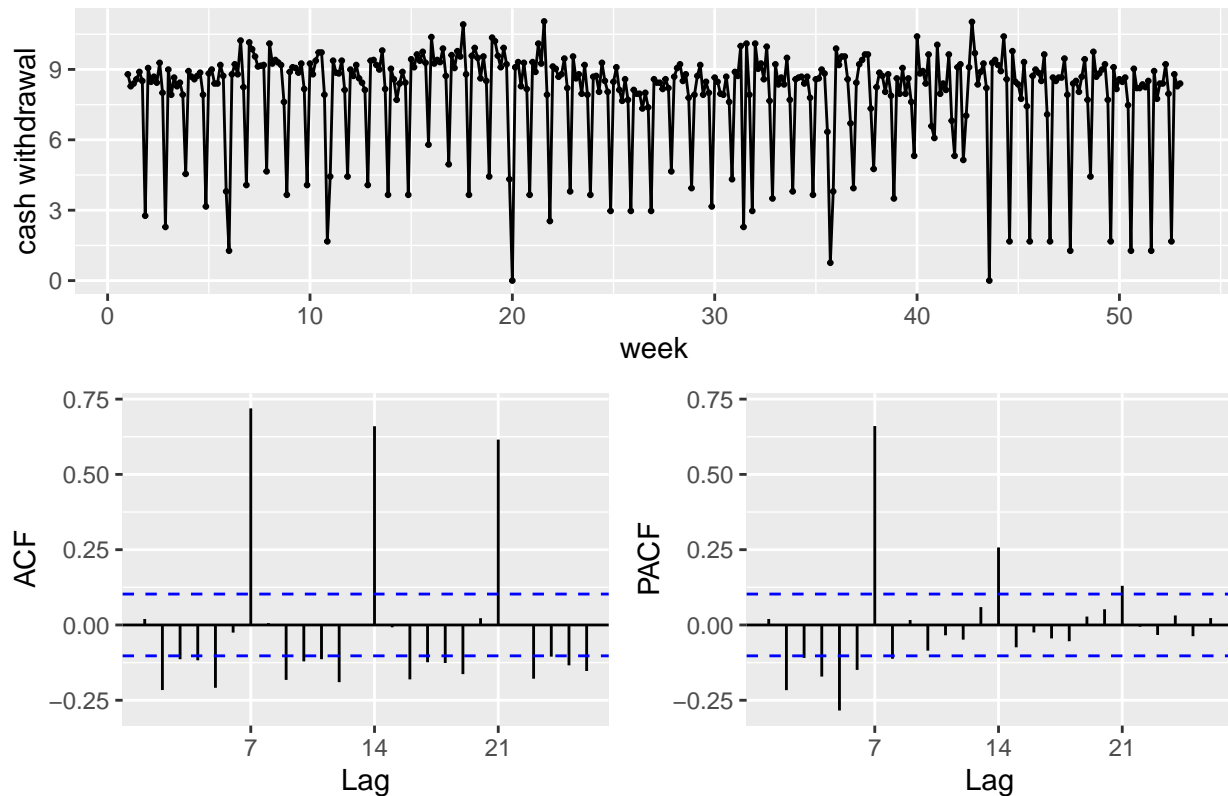


From the subseries plot, it is apparent that Tuesdays having highest mean of ash withdrawal while Saturdays being the lowest.

Next step is to apply BoxCox transformation. With λ being 0.26, the resulting transformation does handle the variability in time series as shown in below transformed plot.

```
atm1.lambda <- BoxCox.lambda(atm1.ts)
atm1.ts.bc <- BoxCox(atm1.ts, atm1.lambda )
ggtsdisplay(atm1.ts.bc, main=paste("ATM1 Cash Withdrawal",round(atm1.lambda, 3)), ylab="cash withdrawal")
```

ATM1 Cash Withdrawal 0.262



Next we will see the number of differences required for a stationary series and the number of differences required for a seasonally stationary series.

```
# Number of differences required for a stationary series
ndiffs(atm1.ts.bc)
```

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

```
# Number of differences required for a seasonally stationary series
nsdiffs(atm1.ts.bc)
```

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

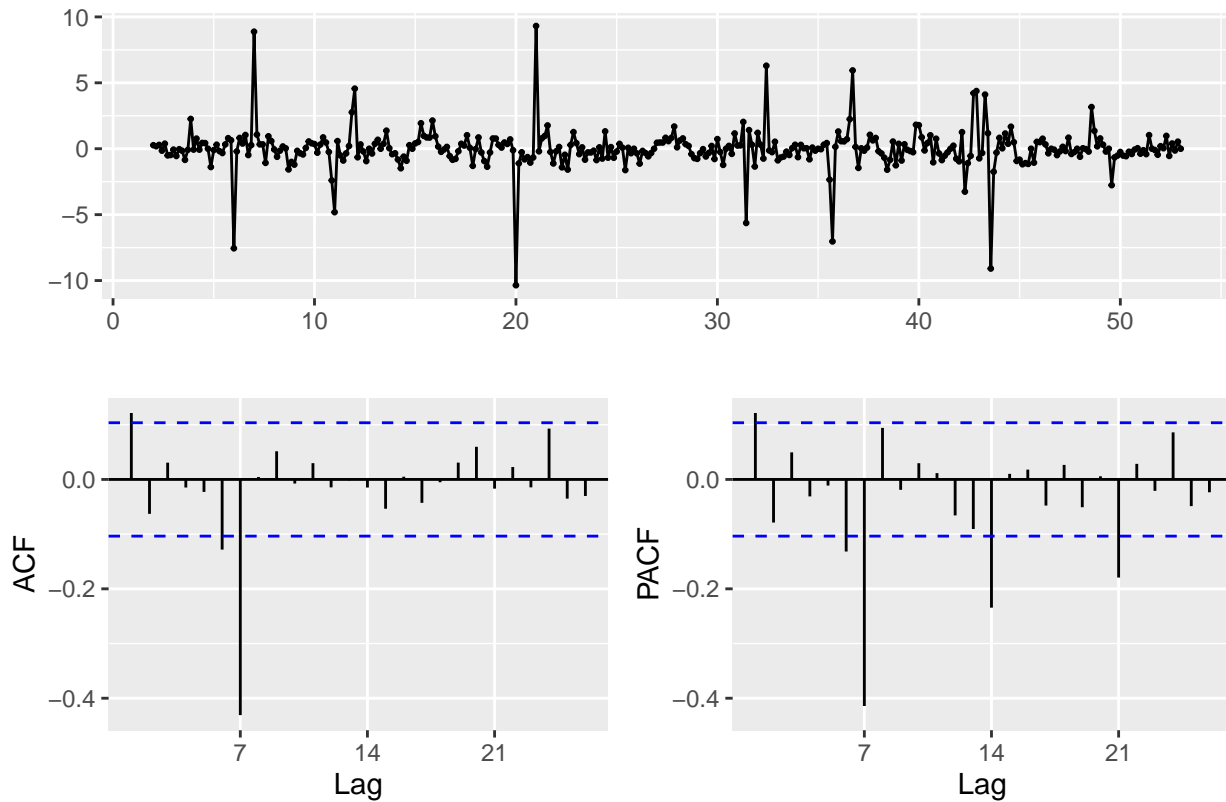
It shows number of differences required for a seasonality stationary series is 1. Next step is to check kpss summary.

```
atm1.ts.bc %>% diff(lag=7) %>% ur.kpss() %>% summary()
```

```
##
## #####
## # KPSS Unit Root Test #
## #####
##
## Test is of type: mu with 5 lags.
##
## Value of test-statistic is: 0.0153
##
## Critical value for a significance level of:
##          10pct  5pct  2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```

We can see the test statistic small and well within the range we would expect for stationary data. So we can conclude that the data are stationary.

```
atm1.ts.bc %>% diff(lag=7) %>% ggtsdisplay()
```



The data is non-stationary with seasonality so there will be a seasonal difference of 1. Finally, the differencing of the data has now made it stationary. From the ACF plot, it is apparent now that there is a significant spike at lag 7 but none beyond lag 7.

Lets start with Holt-Winter's additive model with damped trend since the seasonal variations are roughly constant through out the series.

```
# Holt Winters with damped True
atm1.ts %>% hw(h=31, seasonal = "additive", lambda = atm1.lambda, damped = TRUE)
```

##	Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95
## 53.14286	86.726308	48.2873323	144.09156	34.0075240	183.86219
## 53.28571	99.656005	56.7502143	162.78119	40.5780934	206.17461
## 53.42857	74.268913	40.2785592	125.84028	27.8645027	161.94499
## 53.57143	3.946722	0.9101988	11.36403	0.3067520	18.00566
## 53.71429	99.554782	56.6834213	162.63577	40.5259535	206.00148
## 53.85714	78.851329	43.2063605	132.58498	30.1007501	170.06058
## 54.00000	85.114307	47.2424187	141.74438	33.2015587	181.05113
## 54.14286	86.658670	45.6127105	150.10813	30.9111908	195.01621
## 54.28571	99.582554	53.7351794	169.36386	37.0454815	218.30796
## 54.42857	74.210981	37.9429783	131.29091	25.1978202	172.11308
## 54.57143	3.940224	0.7732156	12.30036	0.2189239	20.04980
## 54.71429	99.485702	53.6737241	169.22060	36.9987686	218.13522
## 54.85714	78.794338	40.7477446	138.25412	27.2771480	180.60622
## 55.00000	85.055212	44.6156330	147.70043	30.1637147	192.09415

```
## 55.14286      86.599982 43.2340302 155.85781 28.2323452 205.80698
## 55.28571      99.518822 51.0490613 175.64880 33.9793617 230.03192
## 55.42857      74.160715 35.8698562 136.50504 22.8992462 181.96358
## 55.57143       3.934588  0.6604831  13.21942  0.1555673  22.09545
## 55.71429      99.425760 50.9921256 175.50745 33.9371713 229.85958
## 55.85714      78.744887 38.5634686 143.67495 24.8394878 190.81691
## 56.00000      85.003935 42.2795812 153.39265 27.5361909 202.77913
## 56.14286      86.549058 41.0923732 161.39633 25.8838357 216.31745
## 56.28571      99.463519 48.6265521 181.69785 31.2829118 241.43875
## 56.42857      74.117099 34.0067798 141.53228 20.8914243 191.57013
## 56.57143       3.929701  0.5664429  14.12646  0.1096150  24.14933
## 56.71429      99.373745 48.5734860 181.55814 31.2445441 241.26666
## 56.85714      78.701976 36.5988213 148.89936 22.7069013 200.76969
## 57.00000      84.959439 40.1763734 158.87600 25.2333014 213.18774
## 57.14286      86.504867 39.1457044 166.76293 23.8038208 226.60572
## 57.28571      99.415528 46.4210490 187.55457 28.8873888 252.59311
## 57.42857      74.079250 32.3163685 146.40758 19.1195026 200.98424
```

Next is to apply exponential smoothing method on this time series. It shows that the ETS(A, N, A) model best fits for the transformed ATM4, i.e. exponential smoothing with additive error, no trend component and additive seasonality.

```
atm1.ts %>% ets(lambda = atm1.lambda )

## ETS(A,N,A)
##
## Call:
## ets(y = ., lambda = atm1.lambda)
##
## Box-Cox transformation: lambda= 0.2616
##
## Smoothing parameters:
##   alpha = 1e-04
##   gamma = 0.3513
##
## Initial states:
##   l = 7.9717
##   s = -4.5094 0.5635 1.0854 0.5711 0.9551 0.5582
##       0.7761
##
## sigma: 1.343
##
##      AIC      AICc      BIC
## 2379.653 2380.275 2418.652
```

Next we will find out the appropriate ARIMA model for this time series. The suggested model seems ARIMA(0,0,2)(0,1,1)[7].

```
atm1.fit3 <- atm1.ts %>% auto.arima(lambda = atm1.lambda )
atm1.fit3

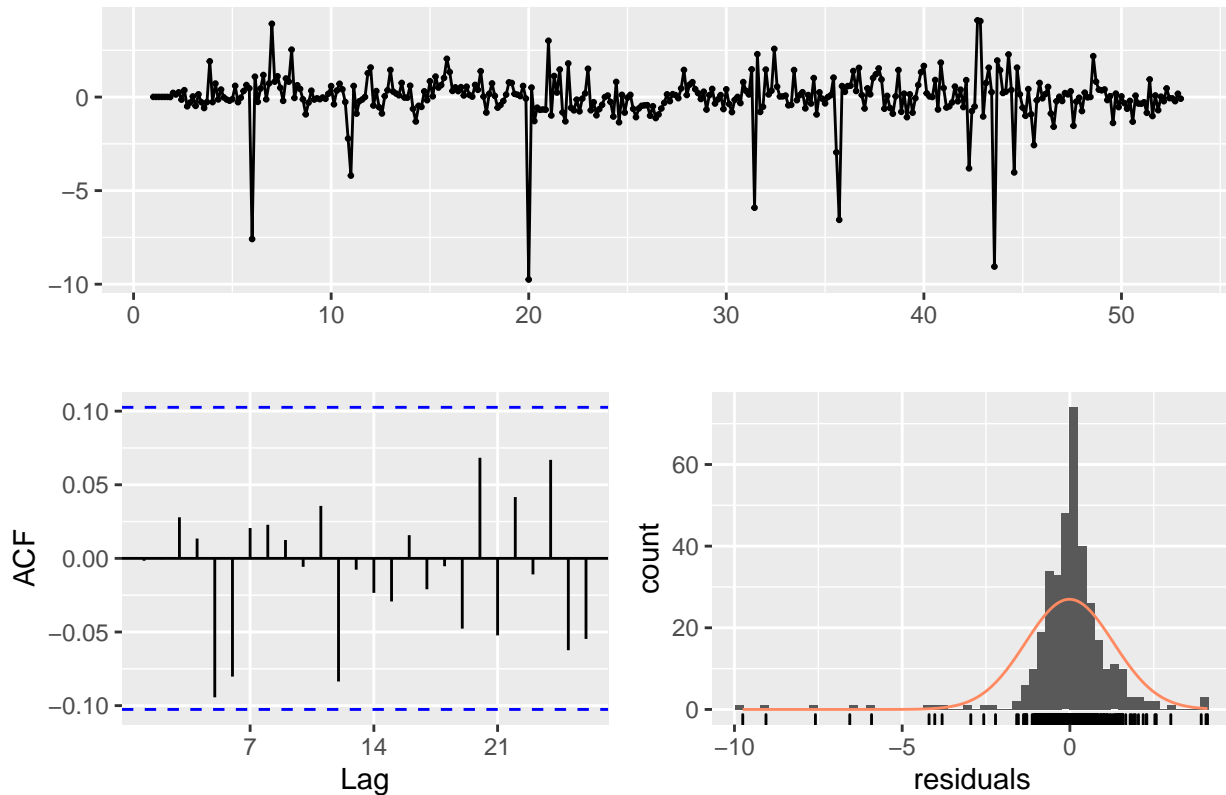
## Series: .
## ARIMA(0,0,2)(0,1,1)[7]
## Box Cox transformation: lambda= 0.2615708
##
## Coefficients:
##      ma1      ma2      sma1
```

```
##      0.1126  -0.1094  -0.6418
## s.e.  0.0524   0.0520   0.0432
##
## sigma^2 estimated as 1.764:  log likelihood=-609.99
## AIC=1227.98  AICc=1228.09  BIC=1243.5
```

Next is to see residuals time series plot which shows residuals are being near normal with mean of the residuals being near to zero. Also there is no significant autocorrelation that confirms that forecasts are good.

```
checkresiduals(atm1.fit3)
```

Residuals from ARIMA(0,0,2)(0,1,1)[7]

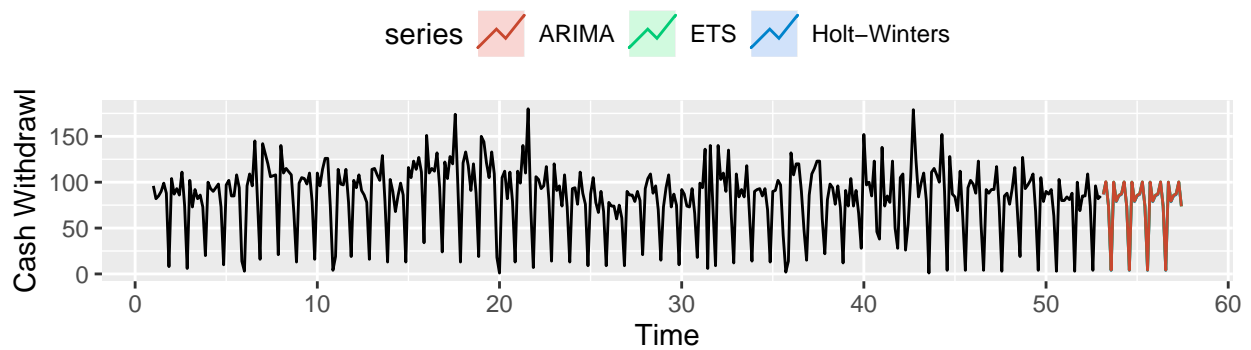


```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,0,2)(0,1,1)[7]
## Q* = 9.8626, df = 11, p-value = 0.5428
##
## Model df: 3.    Total lags used: 14
```

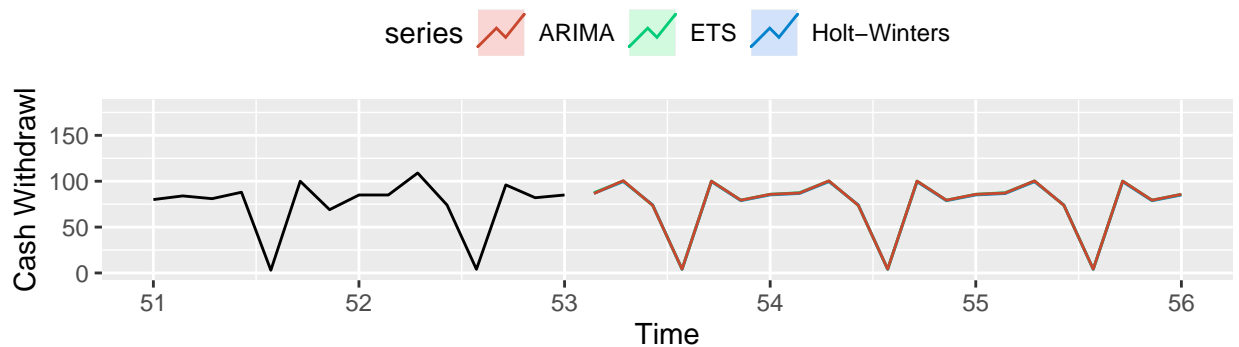
Let's plot the forecast for all the considered models above which will shows a nice visual comparison. it will also show a zoomed in plot to have a clearer view.

```
atm.forecast(atm1.ts)
```

```
## Scale for 'x' is already present. Adding another scale for 'x', which will
## replace the existing scale.
```



Zoom in



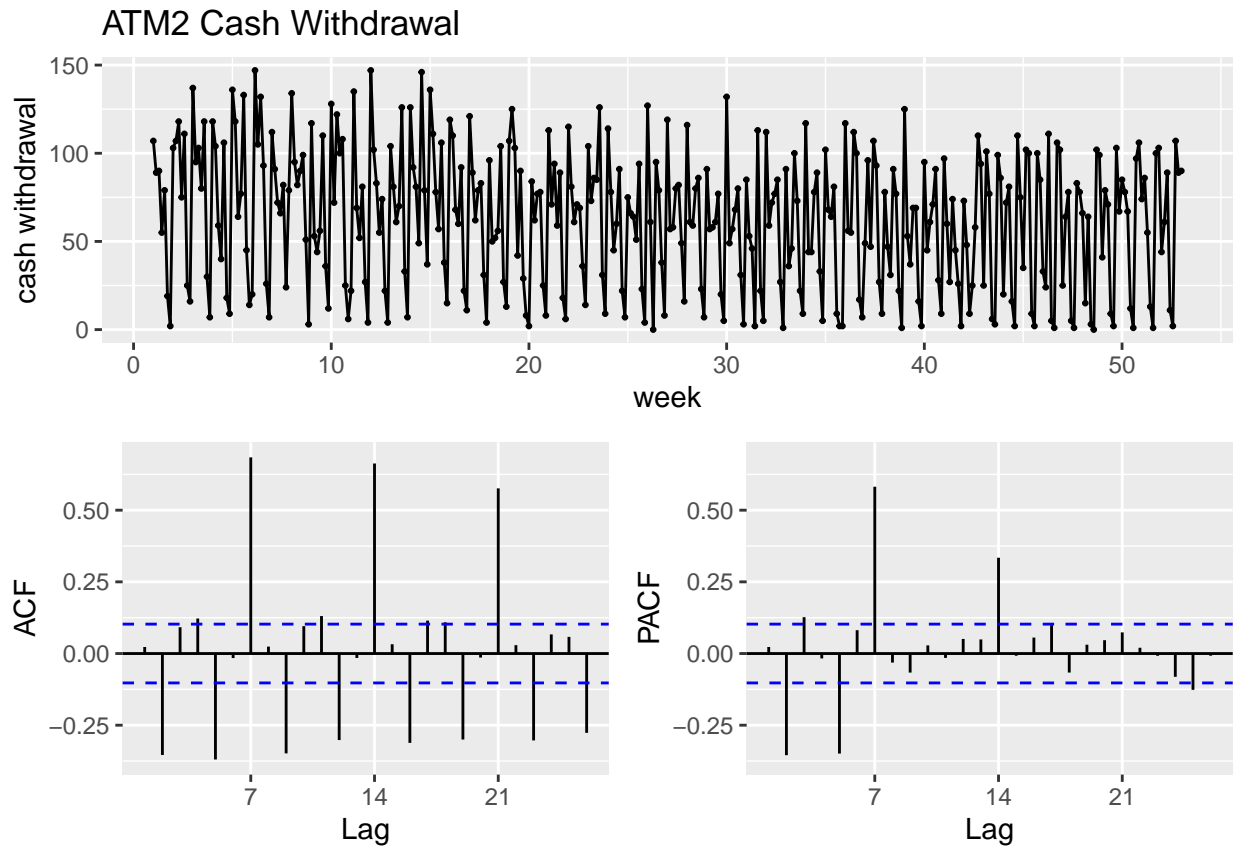
```
model_accuracy(atm1.ts,1)
```

```
##      Holt-Winters      ETS      ARIMA
## 1         49.35115 49.22521 49.18074
```

ATM2

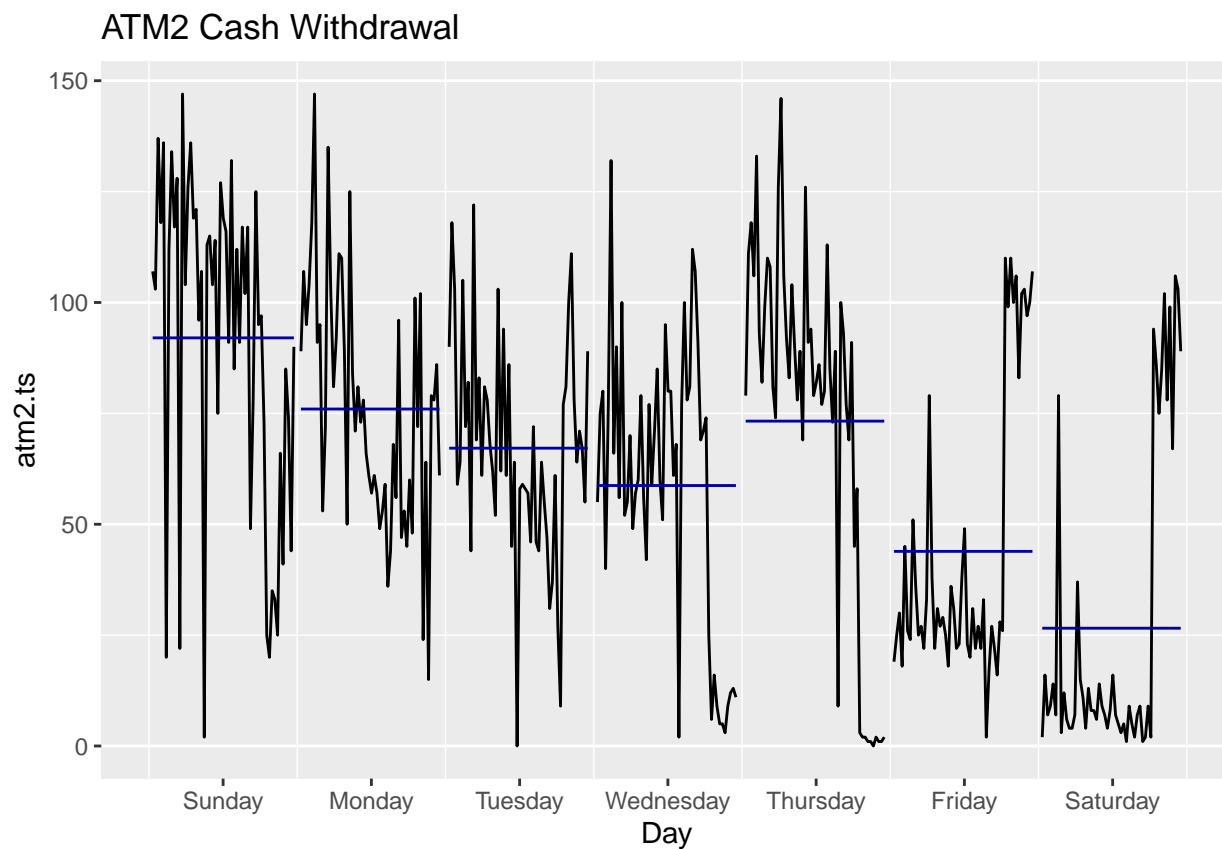
From the time series plot, it is apparent that there is a seasonality in the data but don't see a trend over the period. ACF shows significant lags at 7, 14, and 21 confirming seasonality. From the PACF, there are few significant lags at the beginning but others within critical limit. Overall, it is non-stationary, having seasonality and would require differencing for it to become stationary.

```
atm2.ts <- atm.new %>% filter(ATM=="ATM2") %>% select(Cash) %>% ts(frequency = 7)
ggtsdisplay(atm2.ts, main="ATM2 Cash Withdrawal", ylab="cash withdrawal", xlab="week")
```



From the subseries plot, it is clear that Sunday is having highest mean for cash withdrawal while Saturday has the lowest.

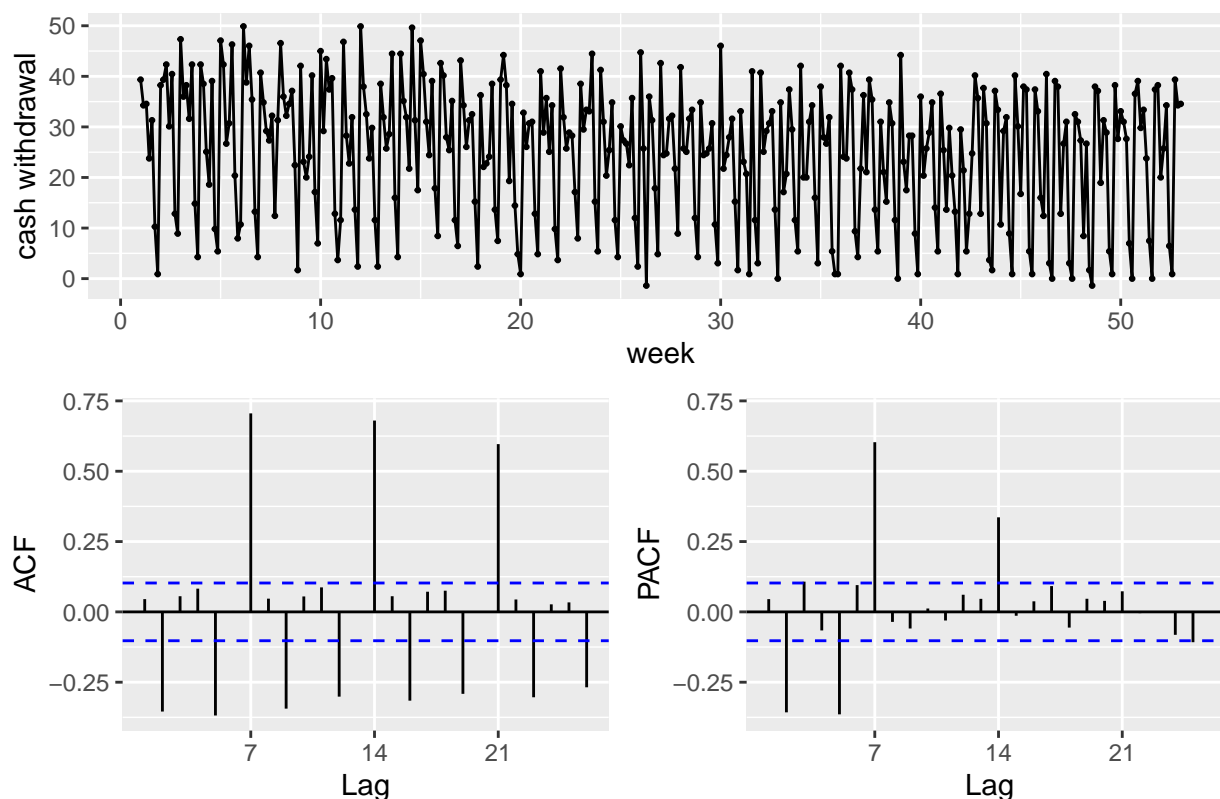
```
ggsubseriesplot(atm2.ts, main="ATM2 Cash Withdrawal")
```



Next step is to apply BoxCox transformation. With λ being 0.72, the resulting transformation does handle the variability in time series as shown in below transformed plot.

```
atm2.lambda <- BoxCox.lambda(atm2.ts)
atm2.ts.bc <- BoxCox(atm2.ts, atm2.lambda )
ggtsdisplay(atm2.ts.bc, main=paste("ATM2 Cash Withdrawal",round(atm2.lambda, 3)), ylab="cash withdrawal")
```


ATM2 Cash Withdrawal 0.724



Number of differences required for a stationary series

```
ndiffs(atm2.ts.bc)
```

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

Number of differences required for a seasonally stationary series

```
nsdiffs(atm2.ts.bc)
```

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

It shows number of differences required is 1 for boxcox transformed data.

```
atm2.ts.bc %>% diff(lag=7) %>% ur.kpss() %>% summary()
```

```
##
```

```
## #####
```

```
## # KPSS Unit Root Test #
```

```
## #####
```

```
##
```

```
## Test is of type: mu with 5 lags.
```

```
##
```

```
## Value of test-statistic is: 0.0162
```

```
##
```

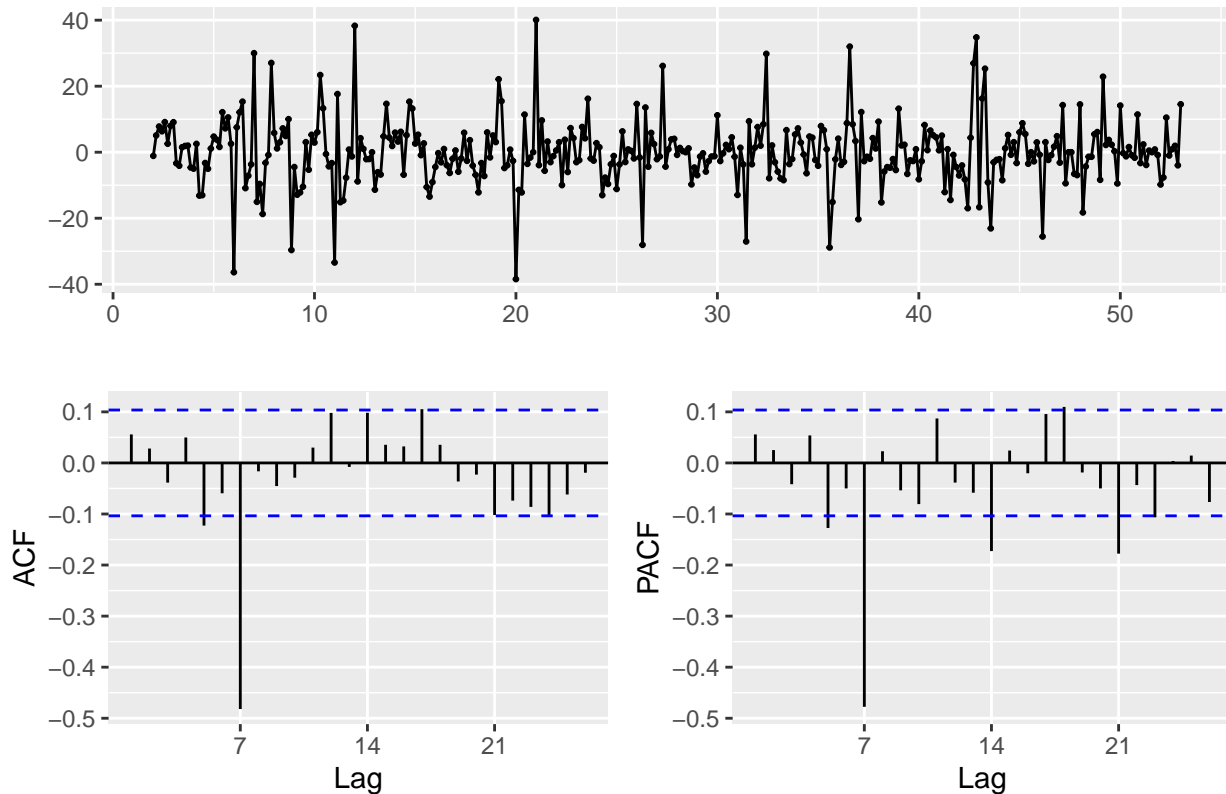
```
## Critical value for a significance level of:
```

```
##          10pct  5pct 2.5pct  1pct
```

```
## critical values 0.347 0.463 0.574 0.739
```

We can see the test statistic small and well within the range we would expect for stationary data. So we can conclude that the data are stationary

```
atm2.ts.bc %>% diff(lag=7) %>% ggtsdisplay()
```



First we will start with Holt-Winters damped method. Damping is possible with both additive and multiplicative Holt-Winters' methods. This method often provides accurate and robust forecasts for seasonal data is the Holt-Winters method with a damped trend.

```
# Holt Winters
```

```
atm2.ts %>% hw(h=31, seasonal = "additive", lambda = atm2.lambda, damped = TRUE)
```

##	Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95
## 53.14286	67.727881	35.291267	105.26894	20.74561	126.87010
## 53.28571	74.012766	40.580383	112.34286	25.35441	134.30920
## 53.42857	10.844773	-3.254323	36.70434	-13.45333	53.31462
## 53.57143	1.648706	-13.353074	22.08418	-26.56518	36.83677
## 53.71429	101.948220	64.792300	143.32926	47.14368	166.72907
## 53.85714	92.500300	56.498440	132.92025	39.58380	155.86508
## 54.00000	68.866332	36.243721	106.55382	21.56968	128.22256
## 54.14286	67.775348	33.216555	108.21505	18.01659	131.58961
## 54.28571	74.059485	38.420870	115.33960	22.46113	139.10021
## 54.42857	10.871202	-4.387783	38.91404	-15.98503	57.04338
## 54.57143	1.663821	-14.982817	24.01057	-29.59257	40.21221
## 54.71429	101.993433	62.324951	146.52593	43.68529	171.80421
## 54.85714	92.542577	54.122362	136.04975	36.29148	160.84582
## 55.00000	68.903765	34.144880	109.49813	18.80244	132.94355
## 55.14286	67.811143	31.293904	110.99079	15.54866	136.05209
## 55.28571	74.094716	36.416629	118.16249	19.82956	143.62898
## 55.42857	10.891142	-5.535746	41.01325	-18.47284	60.59806
## 55.57143	1.675242	-16.563883	25.85263	-32.52085	43.44674
## 55.71429	102.027528	60.025873	149.53496	40.49965	176.59647

```
## 55.85714      92.574457  51.911131 138.99687  33.26820 165.55113
## 56.00000      68.931993  32.200914 112.27407  16.29845 137.40928
## 56.14286      67.838136  29.500528 113.62437  13.30674 140.29932
## 56.28571      74.121282  34.544212 120.84032  17.42301 147.93815
## 56.42857      10.906182  -6.688629  43.01959 -20.91749  64.00567
## 56.57143       1.683868 -18.101954  27.62307 -35.36252  46.56110
## 56.71429     102.053237  57.869192 152.38739  37.54566 181.15191
## 56.85714      92.598496  49.839436 141.79169  30.47392 170.02576
## 57.00000      68.953279  30.388346 114.90931  14.02175 141.66104
## 57.14286      67.858490  27.819168 116.13717  11.26493 144.36295
## 57.28571      74.141315  32.785857 123.39486  15.21412 152.06003
## 57.42857      10.917527  -7.840726  44.94651 -23.32042  67.28683
```

Next is to apply exponential smoothing method on this time series. It shows that the ETS(A, N, A) model best fits for the transformed ATM4, i.e. exponential smoothing with additive error, no trend component and additive seasonality.

```
# ETS
atm2.ts %>% ets(lambda = atm2.lambda)

## ETS(A,N,A)
##
## Call:
## ets(y = ., lambda = atm2.lambda)
##
## Box-Cox transformation: lambda= 0.7243
##
## Smoothing parameters:
##   alpha = 1e-04
##   gamma = 0.3852
##
## Initial states:
##   l = 26.7912
##   s = -17.8422 -13.3191 10.8227 1.8426 4.2781 5.7994
##       8.4185
##
## sigma: 8.5054
##
##      AIC      AICc      BIC
## 3727.060 3727.682 3766.059
```

We will now find out the appropriate ARIMA model for this time series. The suggested model seems ARIMA(3,0,3)(0,1,1)[7] with drift.

```
atm2.fit3 <- atm2.ts %>% auto.arima(lambda = atm2.lambda )
atm2.fit3

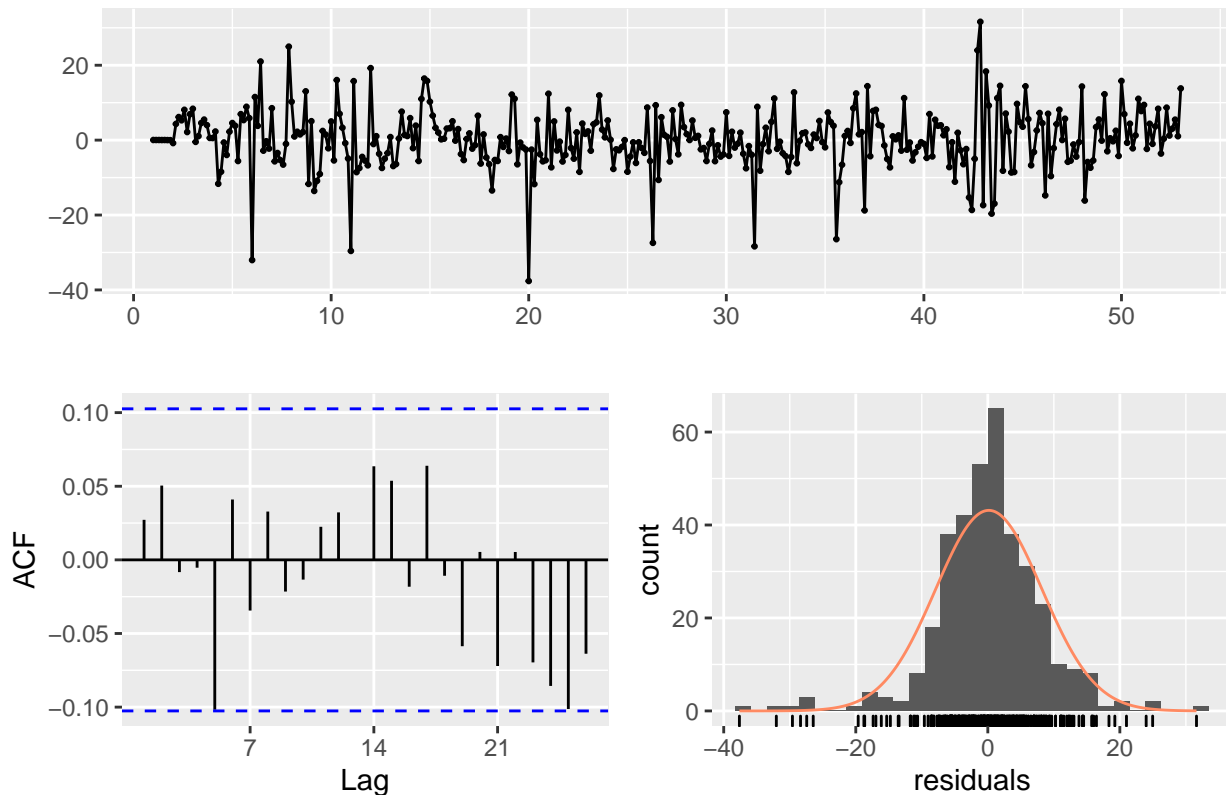
## Series: .
## ARIMA(3,0,3)(0,1,1)[7] with drift
## Box Cox transformation: lambda= 0.7242585
##
## Coefficients:
##      ar1      ar2      ar3      ma1      ma2      ma3      sma1      drift
##      0.4902 -0.4948  0.8326 -0.4823  0.3203 -0.7837 -0.7153 -0.0203
## s.e.  0.0863  0.0743  0.0614  0.1060  0.0941  0.0621  0.0453  0.0072
##
## sigma^2 estimated as 67.52: log likelihood=-1260.59
```

```
## AIC=2539.18   AICc=2539.69   BIC=2574.1
```

Next is to see residuals time series plot which shows residuals are being near normal with mean of the residuals being near to zero. Also there is no significant autocorrelation that confirms that forecasts are good.

```
checkresiduals(atm2.fit3)
```

Residuals from ARIMA(3,0,3)(0,1,1)[7] with drift

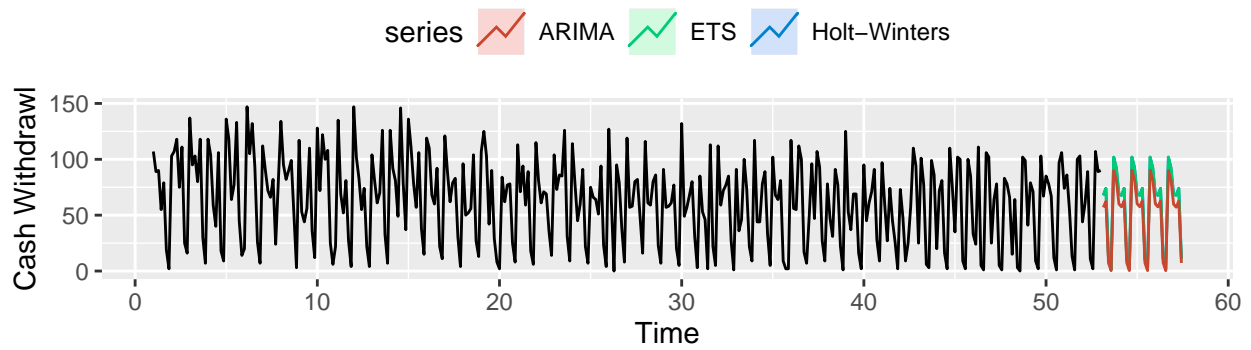


```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(3,0,3)(0,1,1)[7] with drift
## Q* = 8.944, df = 6, p-value = 0.1768
##
## Model df: 8.   Total lags used: 14
```

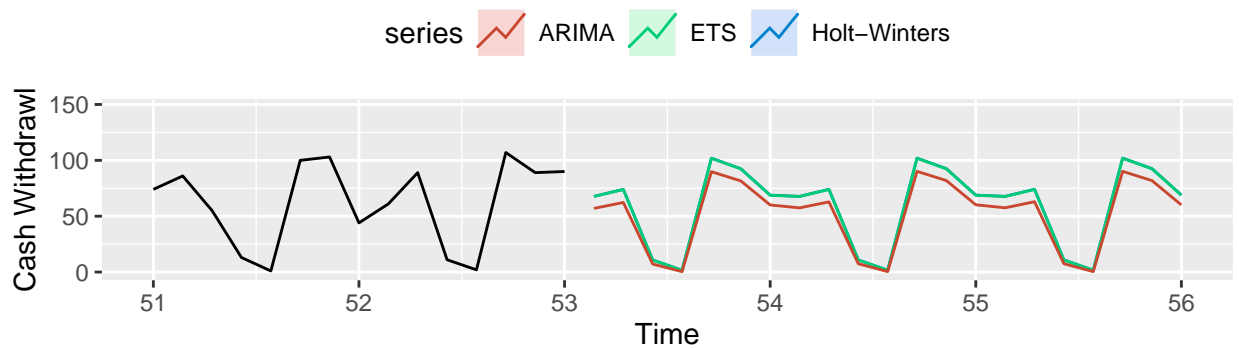
Next step is to plot the forecast for all the considered models above which will shows a nice visual comparison. it will also show a zoomed in plot to have a clearer view.

```
atm.forecast(atm2.ts)
```

```
## Scale for 'x' is already present. Adding another scale for 'x', which will
## replace the existing scale.
```



Zoom in

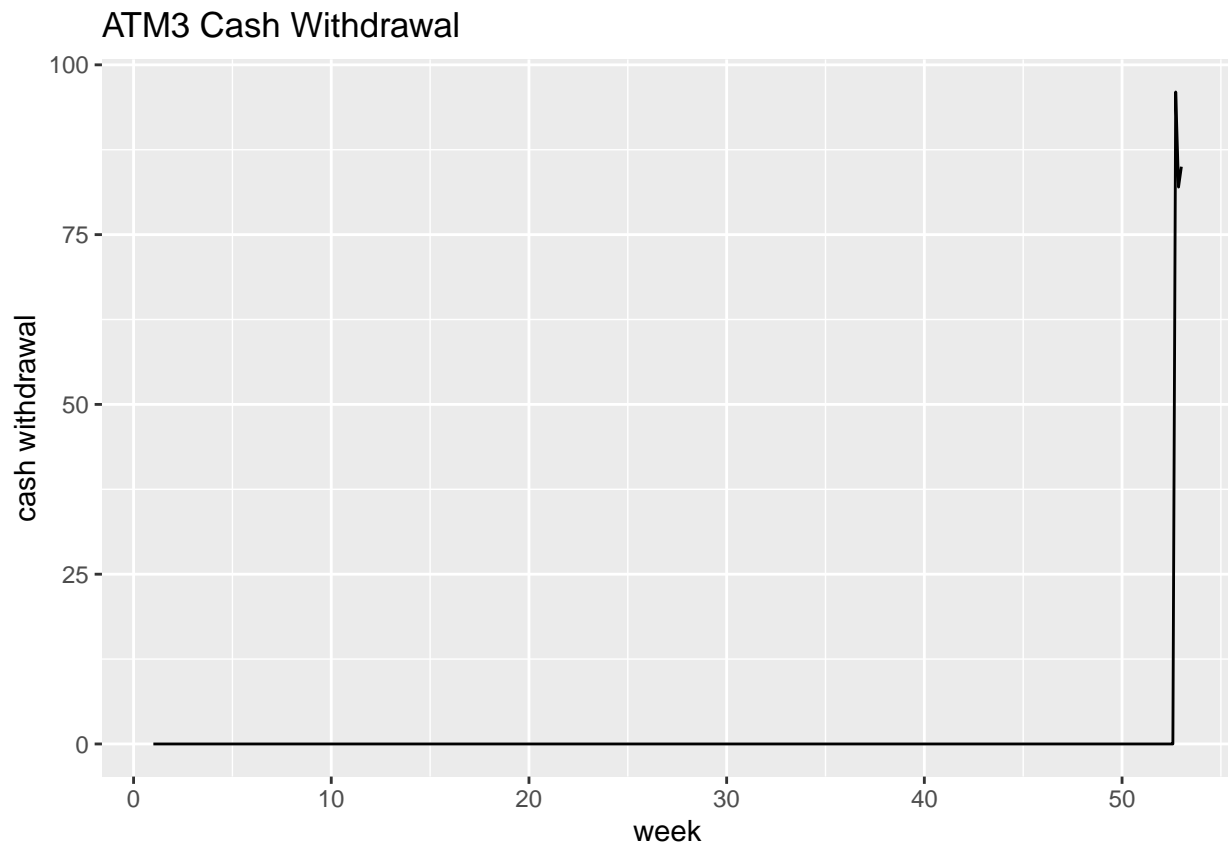


```
model_accuracy(atm2.ts,2)
```

```
##      Holt-Winters      ETS      ARIMA
## 1         57.20467 57.58101 56.58658
```

ATM3

```
atm3.ts <- atm.new %>% filter(ATM=="ATM3") %>% select(Cash) %>% ts(frequency = 7)
autoplot(atm3.ts, main="ATM3 Cash Withdrawal", ylab="cash withdrawal", xlab="week")
```

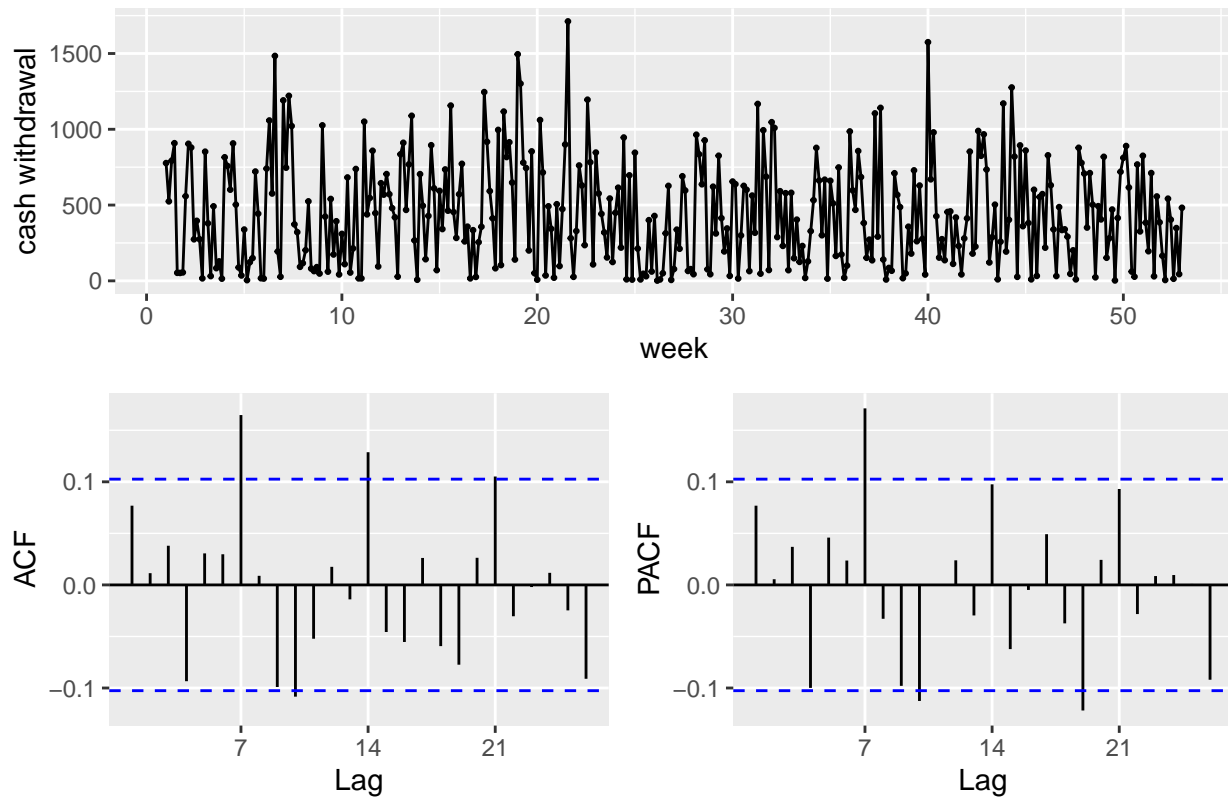


ATM4

Seeing the time series plot, it is apparent that there is seasonality in this series. ACF shows a decrease in every 7th lag. From the PACF, there are few significant lags at the beginning but others within critical limit. Overall, it is non stationary, having seasonality and might require differencing for it to become stationary.

```
atm4.ts <- atm.new %>% filter(ATM=="ATM4") %>% select(Cash) %>% ts(frequency = 7)
ggtsdisplay(atm4.ts, main="ATM4 Cash Withdrawal", ylab="cash withdrawal", xlab="week")
```

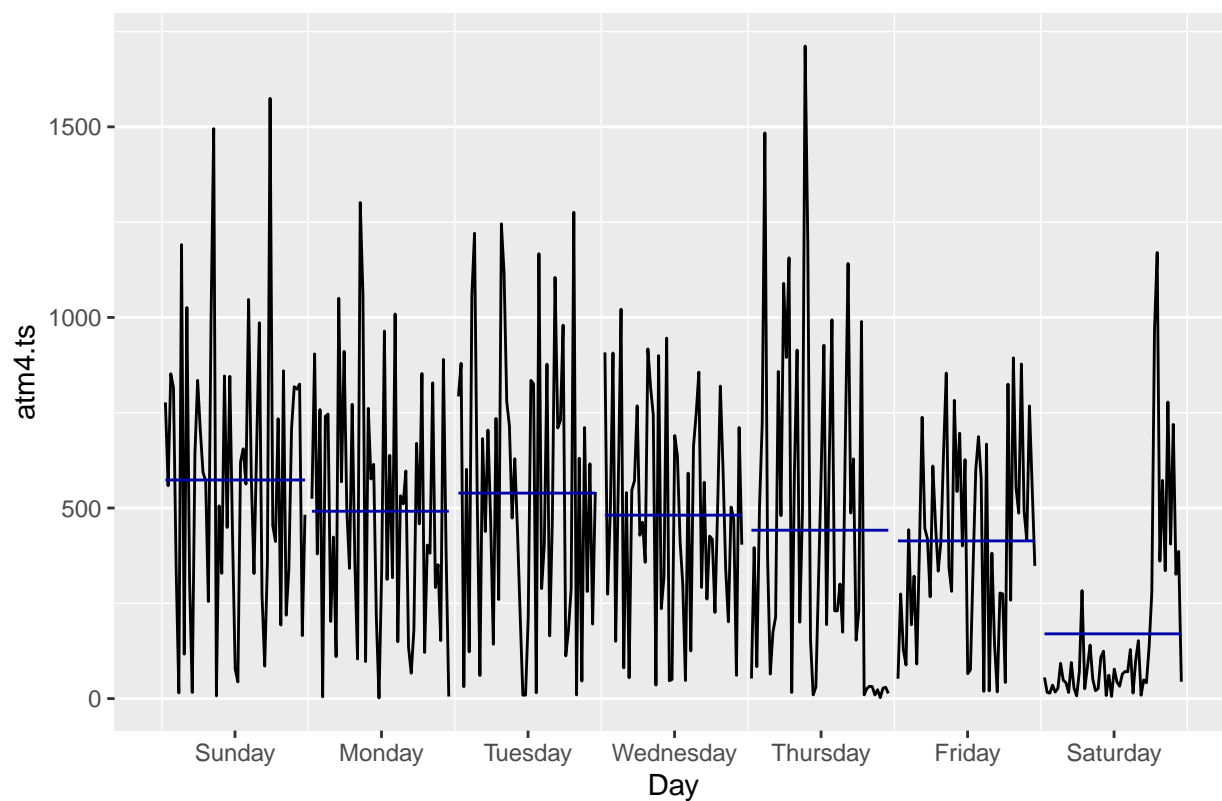
ATM4 Cash Withdrawal



From the subseries plot, it is clear that Sunday is having highest mean for cash withdrawal while Saturday has the lowest.

```
ggsubseriesplot(atm4.ts, main="ATM4 Cash Withdrawal")
```

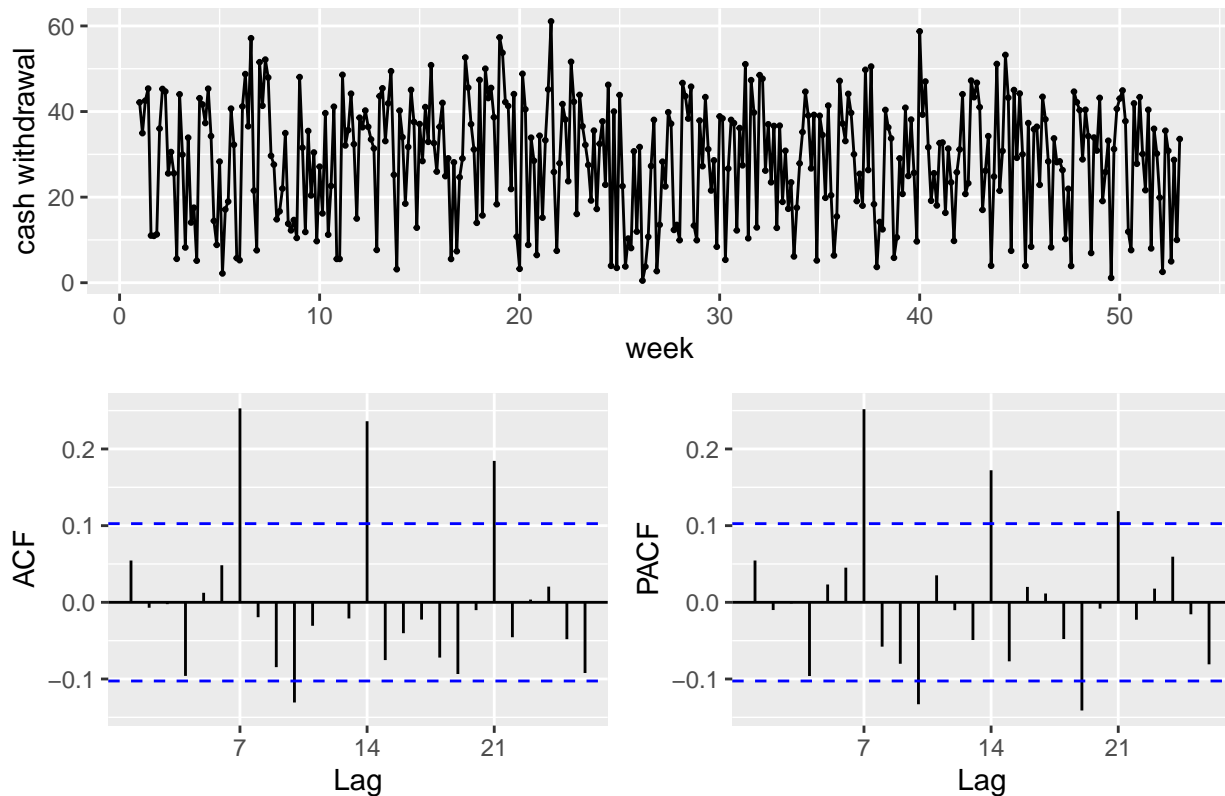
ATM4 Cash Withdrawal



Next step is to apply BoxCox transformation. With λ being 0.45, the resulting transformation does handle the variability in time series as shown in below transformed plot.

```
atm4.lambda <- BoxCox.lambda(atm4.ts)
atm4.ts.bc <- BoxCox(atm4.ts, atm4.lambda )
ggtsdisplay(atm4.ts.bc, main=paste("ATM4 Cash Withdrawal",round(atm4.lambda, 3)), ylab="cash withdrawal")
```


ATM4 Cash Withdrawal 0.45



```
# Number of differences required for a stationary series
ndiffs(atm4.ts.bc)
```

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

```
# Number of differences required for a seasonally stationary series
nsdiffs(atm4.ts.bc)
```

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

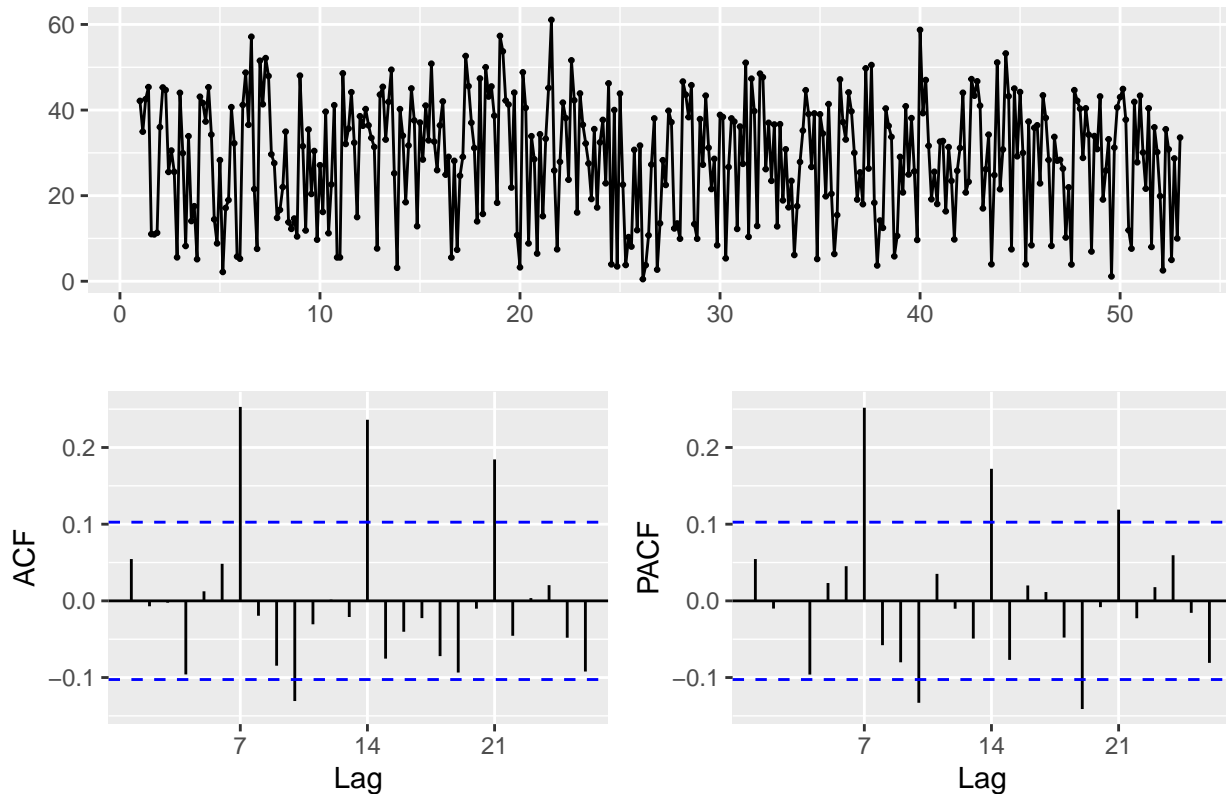
It shows number of differences required is 0 for boxcox transformed data.

```
atm4.ts.bc %>% ur.kpss() %>% summary()
```

```
##
## #####
## # KPSS Unit Root Test #
## #####
##
## Test is of type: mu with 5 lags.
##
## Value of test-statistic is: 0.0792
##
## Critical value for a significance level of:
##          10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463 0.574 0.739
```

We can see the test statistic small and well within the range we would expect for stationary data. So we can conclude that the data are stationary.

```
atm4.ts.bc %>% ggtsdisplay()
```



First we will start with Holt-Winters damped method. Damping is possible with both additive and multiplicative Holt-Winters' methods. This method often provides accurate and robust forecasts for seasonal data is the Holt-Winters method with a damped trend.

```
# Holt Winters
```

```
atm4.ts %>% hw(h=31, seasonal = "additive", lambda = atm4.lambda, damped = TRUE)
```

##	Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95
## 53.14286	326.46664	5.361266e+01	872.7889	4.7560920	1283.0394
## 53.28571	390.55947	7.881312e+01	980.9502	12.8286778	1416.0583
## 53.42857	397.88339	8.186526e+01	993.0862	13.9675943	1430.9036
## 53.57143	88.16707	-1.188133e-04	412.7690	-21.7513686	696.1136
## 53.71429	437.83425	9.906165e+01	1058.5849	20.8852913	1510.7692
## 53.85714	284.50971	3.881453e+01	799.7425	1.5164332	1192.4004
## 54.00000	507.20922	1.308726e+02	1169.8559	35.4549744	1645.5454
## 54.14286	324.77262	5.208909e+01	874.0891	4.2406561	1287.4075
## 54.28571	388.90207	7.701404e+01	982.6924	11.9597845	1421.1069
## 54.42857	396.39921	8.010639e+01	995.1580	13.0852412	1436.3713
## 54.57143	87.59346	-4.150601e-03	414.2213	-22.8793652	700.0263
## 54.71429	436.60517	9.725297e+01	1061.2815	19.8415430	1517.0757
## 54.85714	283.65049	3.777331e+01	802.2506	1.2832703	1198.1842
## 55.00000	506.16225	1.288966e+02	1173.1625	34.1181908	1652.7103
## 55.14286	324.04660	5.092781e+01	877.1018	3.8375566	1293.9333
## 55.28571	388.19148	7.560926e+01	986.0591	11.2521458	1428.1862
## 55.42857	395.76275	7.870397e+01	998.6853	12.3531475	1443.6612
## 55.57143	87.34775	-1.273091e-02	416.4752	-23.8878631	705.0385
## 55.71429	436.07791	9.575384e+01	1065.1735	18.9437425	1524.8790

```
## 55.85714      283.28192  3.689963e+01  805.6726   1.0925953 1205.1449
## 56.00000      505.71298  1.272021e+02 1177.4724  32.9319224 1661.1294
## 56.14286      323.73508  4.992740e+01  880.8442   3.4901477 1301.3790
## 56.28571      387.88653  7.438035e+01  990.1166  10.6232674 1436.1287
## 56.42857      395.48959  7.746167e+01 1002.8304  11.6956591 1451.7237
## 56.57143       87.24235 -2.513721e-02  419.0707 -24.8511354  710.5204
## 56.71429      435.85159  9.439585e+01 1069.5705  18.1202396 1533.3133
## 56.85714      283.12372  3.610421e+01  809.4805   0.9275239 1212.6021
## 57.00000      505.52010  1.256379e+02 1182.2034  31.8238709 1670.0733
## 57.14286      323.60135  4.900310e+01  884.8928   3.1755185 1309.2095
## 57.28571      387.75561  7.323505e+01  994.4625  10.0390607 1444.4301
## 57.42857      395.37231  7.629643e+01 1007.2323  11.0813715 1460.1059
```

Next is to apply exponential smoothing method on this time series. It shows that the ETS(A, N, A) model best fits for the transformed ATM4, i.e. exponential smoothing with additive error, no trend component and additive seasonality.

```
# ETS
atm4.ts %>% ets(lambda = atm4.lambda)

## ETS(A,N,A)
##
## Call:
## ets(y = ., lambda = atm4.lambda)
##
## Box-Cox transformation: lambda= 0.4498
##
## Smoothing parameters:
##   alpha = 1e-04
##   gamma = 0.1035
##
## Initial states:
##   l = 28.6369
##   s = -18.6503 -3.3529 1.6831 4.7437 5.4471 4.9022
##       5.2271
##
## sigma: 12.9202
##
##      AIC      AICc      BIC
## 4032.268 4032.890 4071.267
```

Next we will find out the appropriate ARIMA model for this time series. The suggested model seems ARIMA(0,0,1)(2,0,0)[7] with non-zero mean.

```
# Arima
atm4.fit3 <- atm4.ts %>% auto.arima(lambda = atm4.lambda)
atm4.fit3

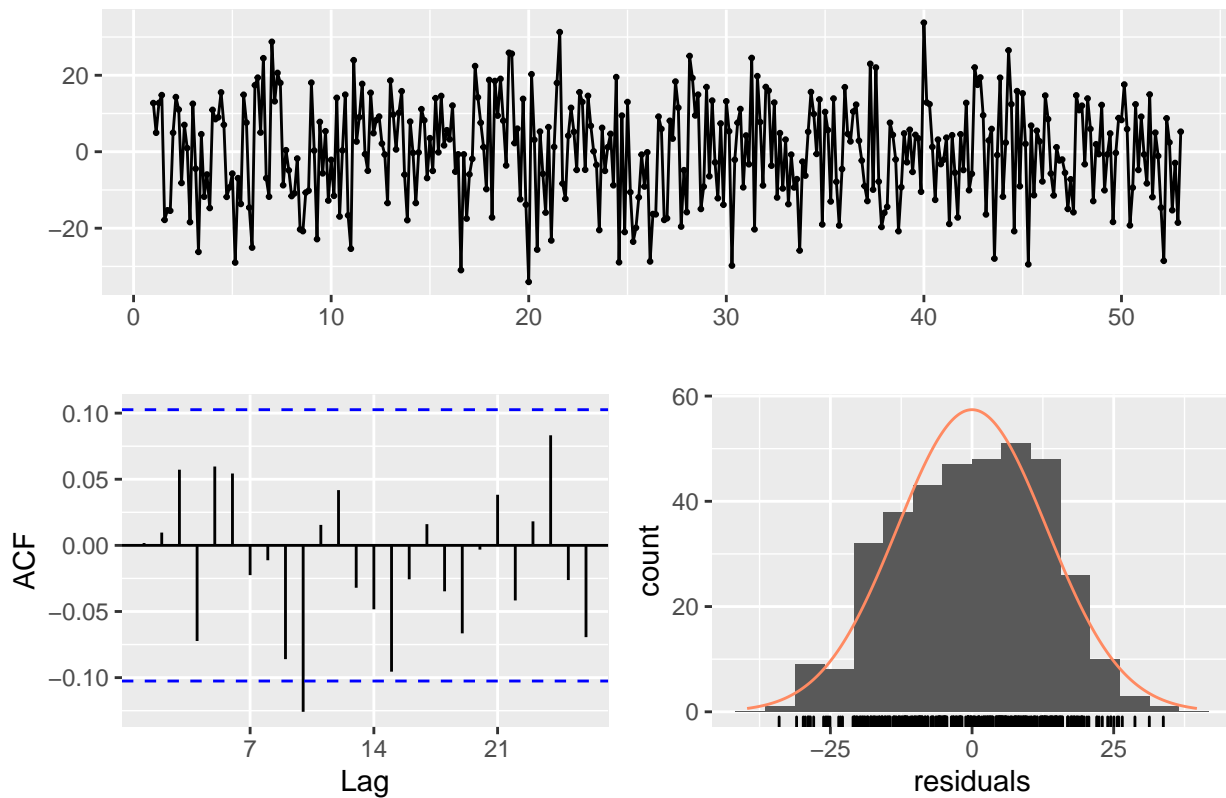
## Series: .
## ARIMA(0,0,1)(2,0,0)[7] with non-zero mean
## Box Cox transformation: lambda= 0.449771
##
## Coefficients:
##      ma1      sar1      sar2      mean
##    0.0790  0.2078  0.2023  28.6364
## s.e.  0.0527  0.0516  0.0525  1.2405
##
```

```
## sigma^2 estimated as 176.5: log likelihood=-1460.57
## AIC=2931.14 AICc=2931.3 BIC=2950.64
```

Next is to see residuals time series plot which shows residuals are being near normal with mean of the residuals being near to zero. Also there is no significant autocorrelation that confirms that forecasts are good.

```
checkresiduals(atm4.fit3)
```

Residuals from ARIMA(0,0,1)(2,0,0)[7] with non-zero mean

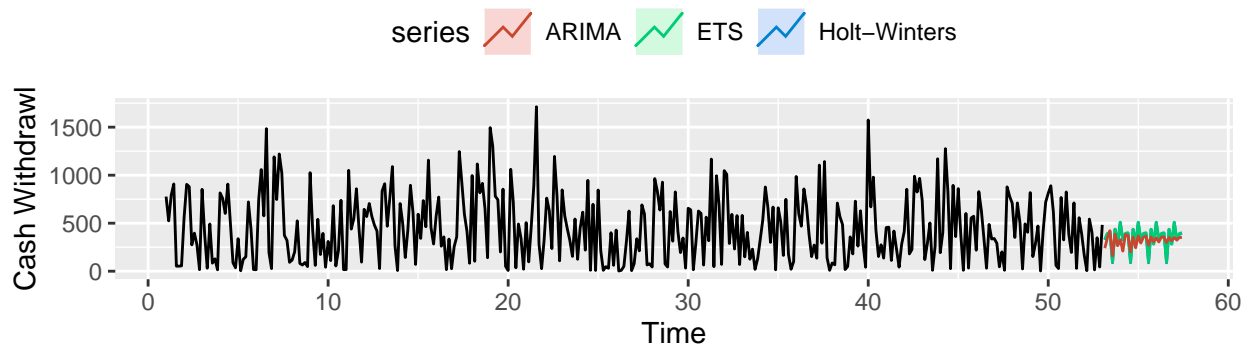


```
##
## Ljung-Box test
##
## data: Residuals from ARIMA(0,0,1)(2,0,0)[7] with non-zero mean
## Q* = 16.645, df = 10, p-value = 0.0826
##
## Model df: 4. Total lags used: 14
```

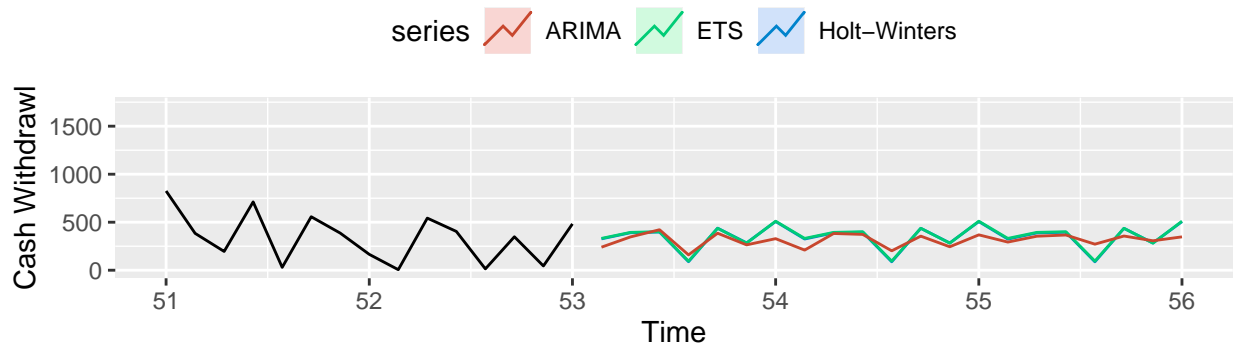
Next is to plot the forecast for all the considered models above which will shows a nice visual comparison. it will also show a zoomed in plot to have a clearer view.

```
atm.forecast(atm4.ts)
```

```
## Scale for 'x' is already present. Adding another scale for 'x', which will
## replace the existing scale.
```



Zoom in



```
model_accuracy(atm4.ts,4)
```

```
## Holt-Winters      ETS      ARIMA
## 1      360.3953 360.7951 315.7226
```

Part B - Forecasting Power

```
download.file(
  url="https://github.com/amit-kapoor/data624/blob/main/Project1/ResidentialCustomerForecastLoad-624.xlsx",
  destfile = temp.file,
  mode = "wb",
  quiet = TRUE)
power.data <- read_excel(temp.file, skip=0, col_types = c("numeric","text","numeric"))

head(power.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
```

Part C - Waterflow Pipe

```
download.file(url="https://github.com/amit-kapoor/data624/blob/main/Project1/Waterflow_Pipe1.xlsx?raw=t",
              destfile = temp.file,
              mode = "wb",
              quiet = TRUE)
pipe1.data <- read_excel(temp.file, skip=0, col_types = c("date","numeric"))

download.file(url="https://github.com/amit-kapoor/data624/blob/main/Project1/Waterflow_Pipe2.xlsx?raw=t",
              destfile = temp.file,
              mode = "wb",
              quiet = TRUE)

pipe2.data <- read_excel(temp.file, skip=0, col_types = c("date","numeric"))
head(pipe1.data)
```

```
## # A tibble: 6 x 2
##   `Date Time`      WaterFlow
##   <dtm>          <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
```