Divvy Bike Weekly Usage Forecast

load the package:

```r
library(fpp)
```

```
## Loading required package: forecast

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo

## Loading required package: fma

## Loading required package: expsmooth

## Loading required package: lmtest

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: tseries
```

```r
library(TSA)
```

```
## Registered S3 methods overwritten by 'TSA':
##   method       from
##   fitted.Arima forecast
##   plot.Arima   forecast

##
## Attaching package: 'TSA'

## The following objects are masked from 'package:stats':
##
##     acf, arima

## The following object is masked from 'package:utils':
##
##     tar
```

```
library(tseries)
library(ggplot2)
library(forecast)
library(car)
```

```
## Loading required package: carData
```

```
library(MLmetrics)
```

```
##
## Attaching package: 'MLmetrics'
```

```
## The following object is masked from 'package:base':
##
##     Recall
```

```
library(vars)
```

```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
```

```
## The following objects are masked from 'package:fma':
##
##     cement, housing, petrol
```

```
## Loading required package: strucchange
```

```
## Loading required package: sandwich
```

```
## Loading required package: urca
```

Part 1: Data Processing

Import Divvy Bike data:

```
divvy_data <- read.csv('Divy_daily_duration.csv')
divvy<-divvy_data$duration
```

Start aggregate the data by weekly basis

```
divvy_agg <- c()
l <- (length(divvy)-6)/7


for (i in 1:l){
  a <- (i-1)*7+1+3 # since the start day is Thursday and end day is Tuesday, we do not use the first an
```

```
  b <- i*7+3
  c <- sum(divvy[a:b])
  #divvy_agg <- c(divvy_agg,c)
  divvy_agg[i] = c
}
```

From 2013/06/30 Sunday to 2019/12/28 Saturday; In total 339 weeks

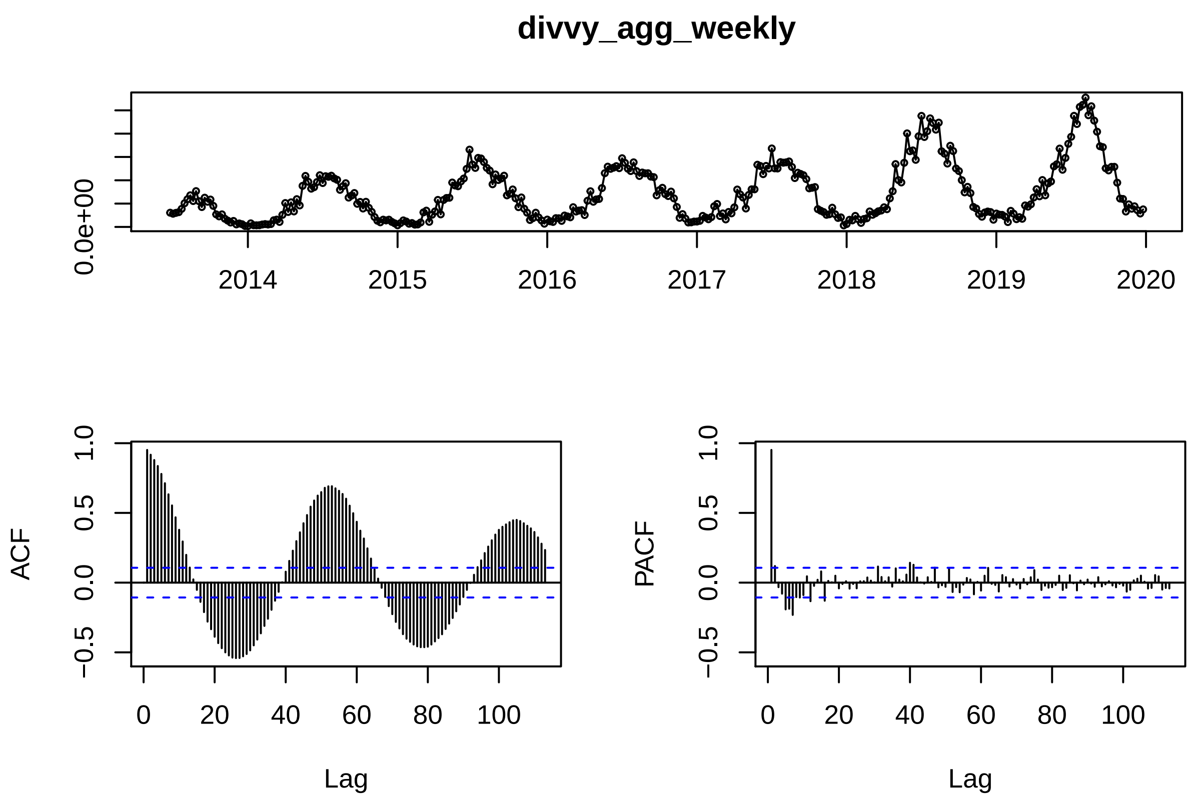
Plot the aggreated data

```
divvy_agg_weekly <- ts(divvy_agg, start=c(2013,26),frequency=52)
divvy_agg_weekly
```

```
## Time Series:
## Start = c(2013, 26)
## End = c(2019, 52)
## Frequency = 52
##   [1]  30439829  28311019  30120450  31644021  38723962  50727234  59251259
##   [8]  67795105  55483059  76746751  55198113  42622049  62657947  53968821
##  [15]  58631980  45072507  26891531  22628060  26898843  17519566  13520373
##  [22]   9467009  12806685   5424109   7423663   5735693   2037577   1769624
##  [29]   7643736   3637395   3476235   3672188   5144109   6055938   5196131
##  [36]   6182792  14298979  15627834  10649585  25880720  51193992  32085948
##  [43]  52705680  33233906  59509035  46052670  88036847 109346699  96710047
##  [50]  82026798  85128221  96131570 110856704  94086275 108726776 107493505
##  [57] 109742674 104067226 101018086  79459035  87298888  93675136  62871412
##  [64]  67141213  72684614  49552691  53533373  39515335  53858265  40954383
##  [71]  32302099  21314255  13249115   9956103  15532575  13961422  15485312
##  [78]  10792629   7722454   3453641   8110685  13956181  11566295   6828039
##  [85]   8256944   4880240   5387374   9492880  31178979  34835921  10897601
##  [92]  26153030  33127794  57879123  26904303  57871569  62041228  62518825
##  [99]  95116524  88762174  87217583  97541328 103761228 124468016 165695652
## [106] 133869031 126501805 148029068 146772515 139132873 126602131 120893481
## [113]  91455600 112505923 100596514 103877142 109850897  67697792  71058907
## [120]  80322494  61370067  42409054  63183747  39034160  30538988  14847081
## [127]  18715883  30466798  20294634  13871228   6973792  15730623  12031672
## [134]  10667481  18796603  18553732  12950387  24408728  22508149  20630594
## [141]  42279014  32997876  35035773  35961257  25510119  56178451  76485787
## [148]  53982158  58946479  59973361  83307230 115197430 129145909 124578313
## [155] 126995995 130293956 125991656 147258977 137348172 124790301 119846463
## [162] 138495595 119772878 109284368 116562030 114895204 115113488 107912819
## [169] 106750137  67693242  78914234  83883029  70330831  66215186  75762207
## [176]  61257889  42793753  19507173  27408561  17851450   9505671   9845882
## [183]  11467814  11510543  13054657  23623974  20201302  16473600  20951457
## [190]  44258321  49210576  23372376  28899895  16108092  32133816  29063933
## [197]  41950159  80150389  69688401  63069970  39469918  68581655  80616026
## [204]  80633362 133322300 130653552 113127668 130802419 125380470 168238982
## [211] 125608749 125315016 138944037 137703073 138476849 140375669 128606895
## [218] 104787498 116200652 113581048 110891360 102451527  82743228  83942465
## [225]  85431815  38093795  34675408  31837263  25955102  27128811  41199915
## [232]  27202364  19263277  20423234   3313079   6185129  15316609  14075970
## [239]  23673027  16523306   8615457  16952340  18543893  33171946  25980799
## [246]  29328362  33414790  35152681  42083791  38019741  61218335  76525071
```

```
## [253] 134676910 100397490  95368995 137531594 200561134 162318031 164028059
## [260] 143841202 194417432 238100235 192821773 205257489 232747003 222801891
## [267] 208563949 223589296 162149504 157095665 135893008 174030599 162663703
## [274] 125259055 119806304 100289079  74087564  85948336  72491516  42853728
## [281]  39509845  27763304  21873709  30928069  32993725  31855264  15498967
## [288]  28600842  26271614  25799946  22180847  10426745  34445520  28418032
## [295]  16602542  20919843  17463181  46039518  43521017  48485360  63131408
## [302]  80919978  65638033 100525547  67743270  93395401  97437584 129847172
## [309] 134325875 167957018 122566812 147974818 178285109 193242332 238218202
## [316] 220713985 257302325 261689847 277309784 239483908 258729796 227934962
## [323] 204146310 172990844 171079453 125723606 121237335 128755658 128838632
## [330]  94832753  60821586  59900414  32995787  48439660  39050558  44207862
## [337]  35896430  28824347  37586842
```

```
tsdisplay(divvy_agg_weekly)
```



Split Train and Test (287 Weeks for Train; 52 Weeks for Test)

```
train_divvy <- window(divvy_agg_weekly,start=c(2013,26),end=c(2018,52))
test_divvy <- window(divvy_agg_weekly,start=c(2019,1),end=c(2019,52))
```

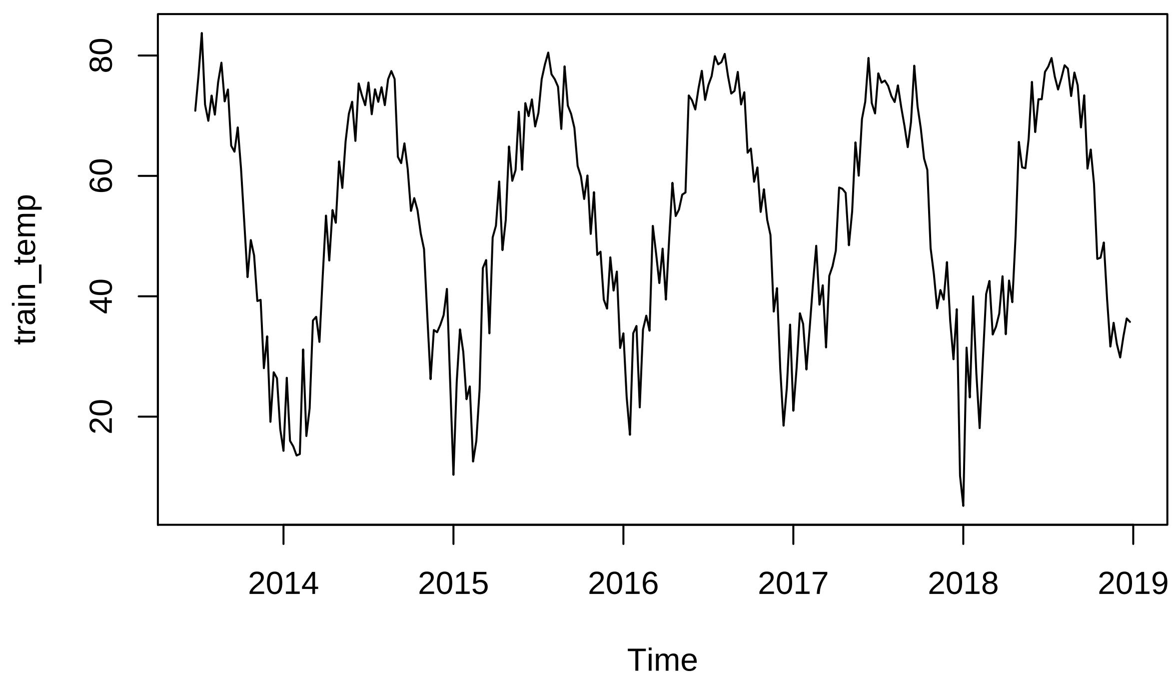
Import Weather Data and Split it the same way as Divvy Bike data

```
weather <- read.csv('weather_data.csv')
```

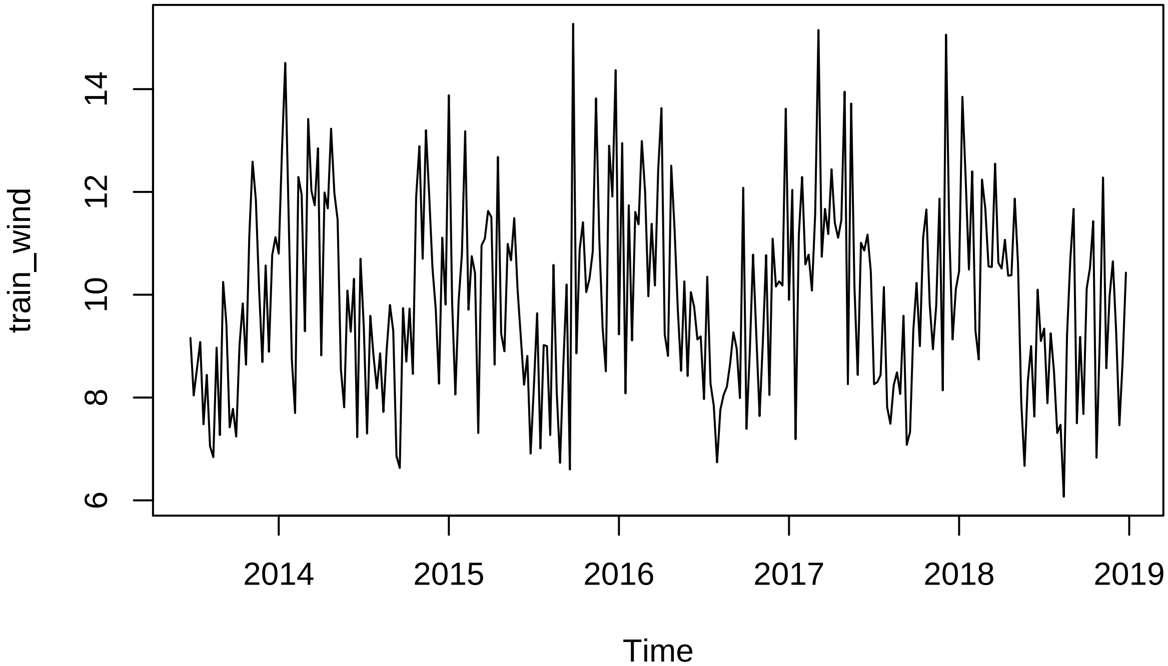
```r
temperature <- ts(weather$avg_temp,start=c(2013,26),frequency=52)
precipitation <- ts(weather$sum_pre,start=c(2013,26),frequency=52)
snowdepth <- ts(weather$sum_snow,start=c(2013,26),frequency=52)
windspeed <- ts(weather$avg_wind,start=c(2013,26),frequency=52)

train_temp <- window(temperature,start=c(2013,26),end=c(2018,52))
train_pre <- window(precipitation,start=c(2013,26),end=c(2018,52))
train_snow <- window(snowdepth,start=c(2013,26),end=c(2018,52))
train_wind <- window(windspeed,start=c(2013,26),end=c(2018,52))

test_temp <- window(temperature,start=c(2019,1),end=c(2019,52))
```

Plot each of the 4 variables in weather dataset

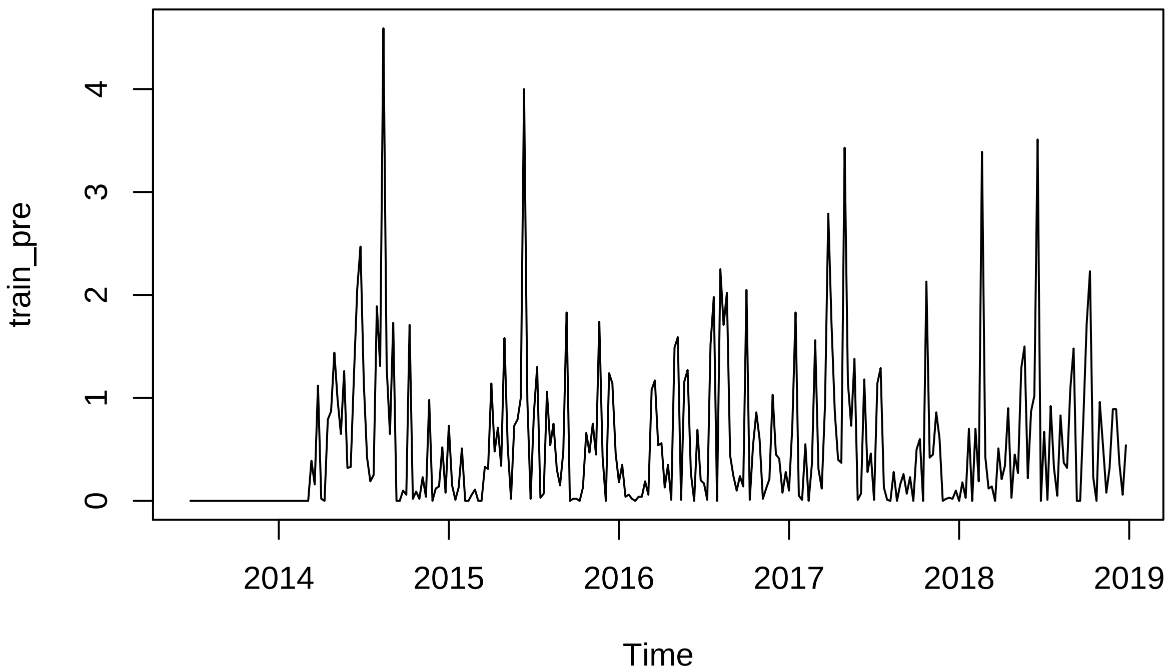
```r
plot(train_temp)
```



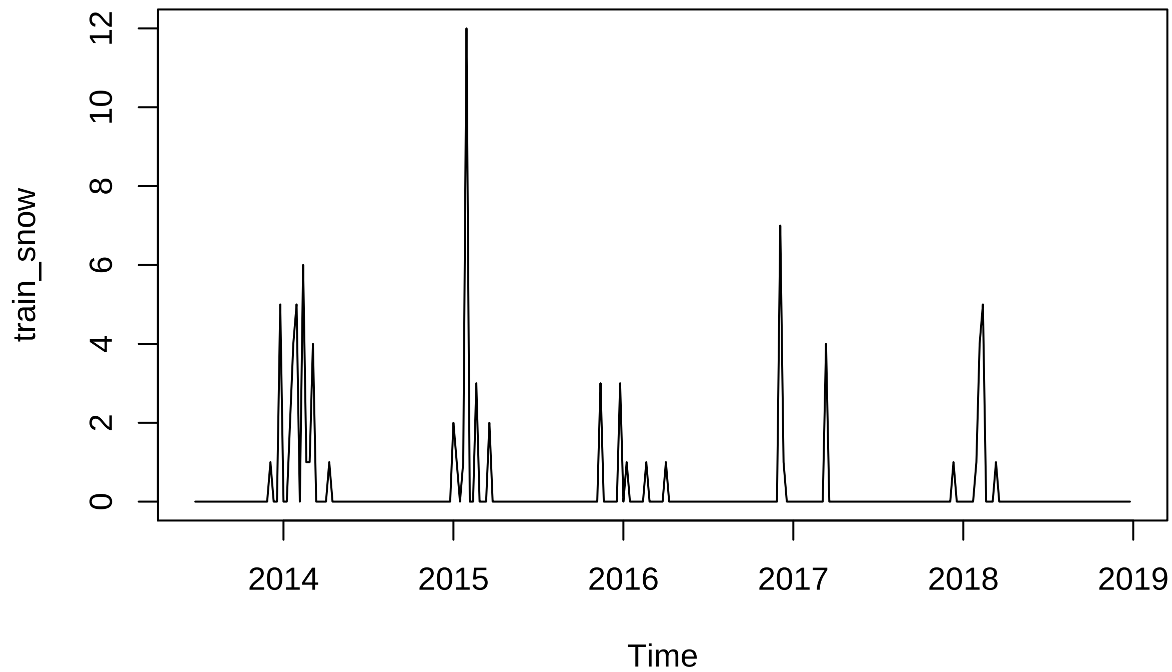

```r
plot(train_wind)
```




```
plot(train_pre)
```

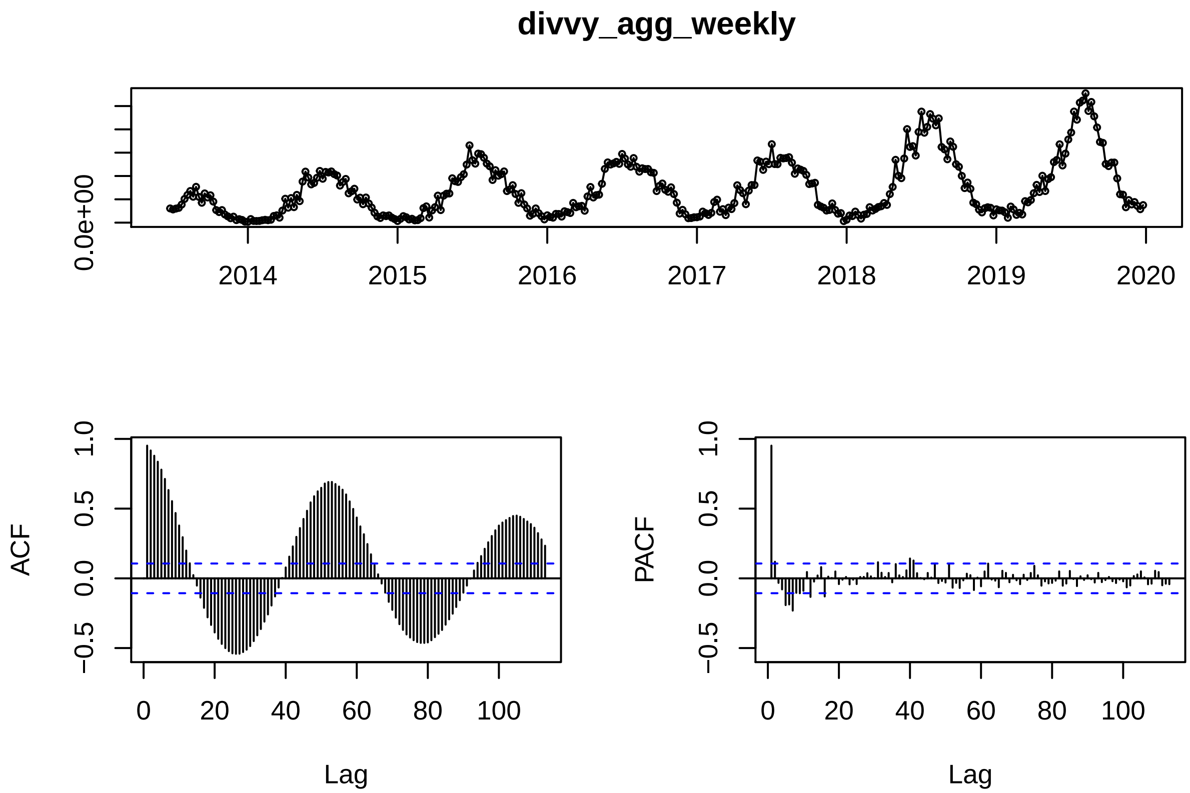



```
plot(train_snow)
```



Part 2: EDA

```
divvy_agg_weekly <- ts(divvy_agg, start=c(2013,26),frequency=52)
tsdisplay(divvy_agg_weekly)
```




```
length(divvy_agg_weekly)
```

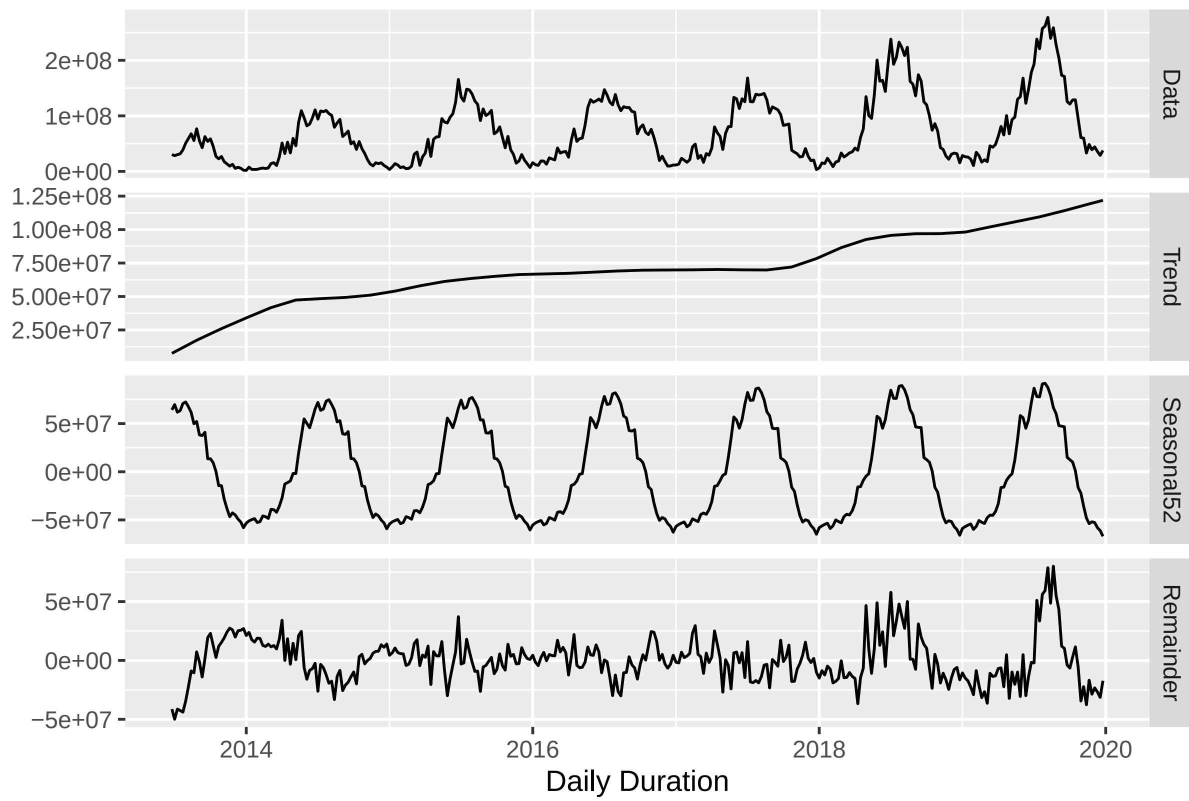
```
## [1] 339
```

```
str(divvy_agg_weekly)
```

```
##  Time-Series [1:339] from 2013 to 2020: 30439829 28311019 30120450 31644021 38723962 50727234 592512
```

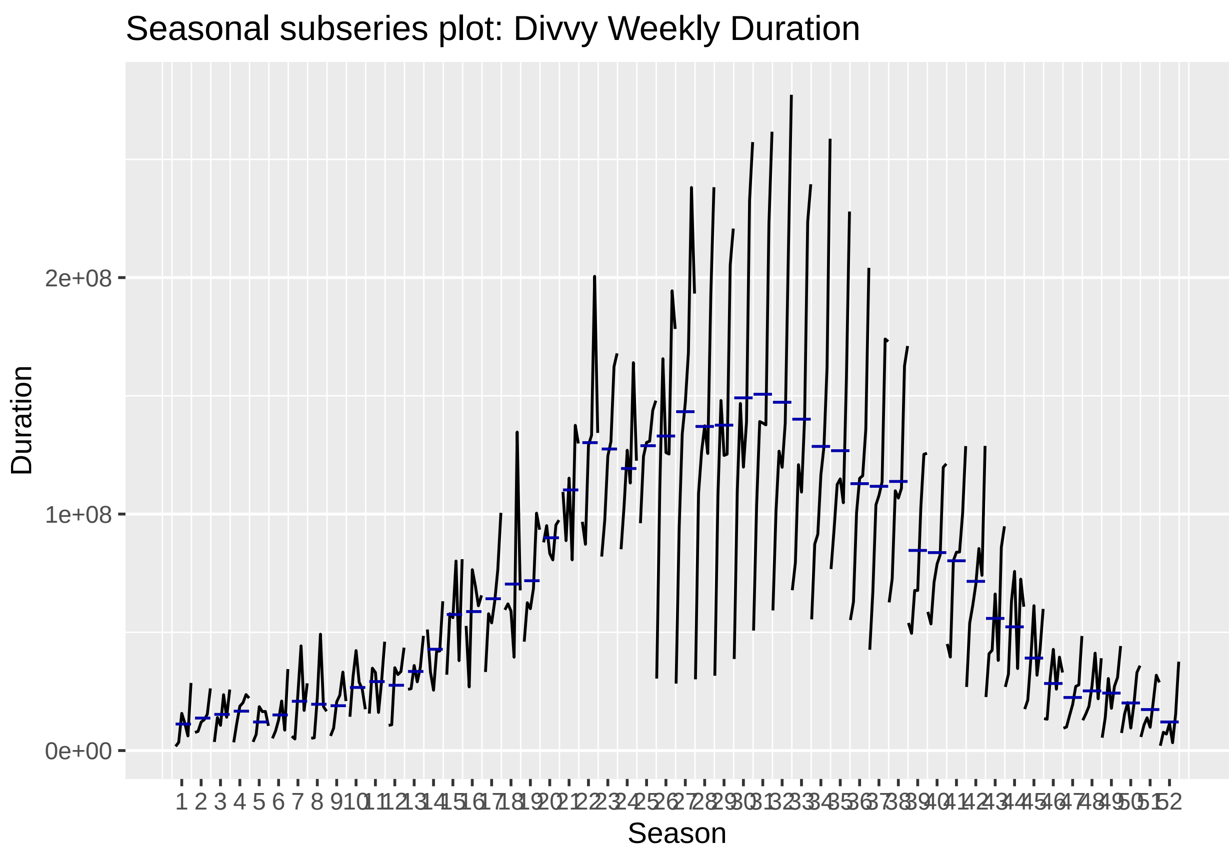
Time Series Decomposition Plots

```
divvy_agg_weekly%>% mstl() %>%
  autoplot() + xlab("Daily Duration")
```

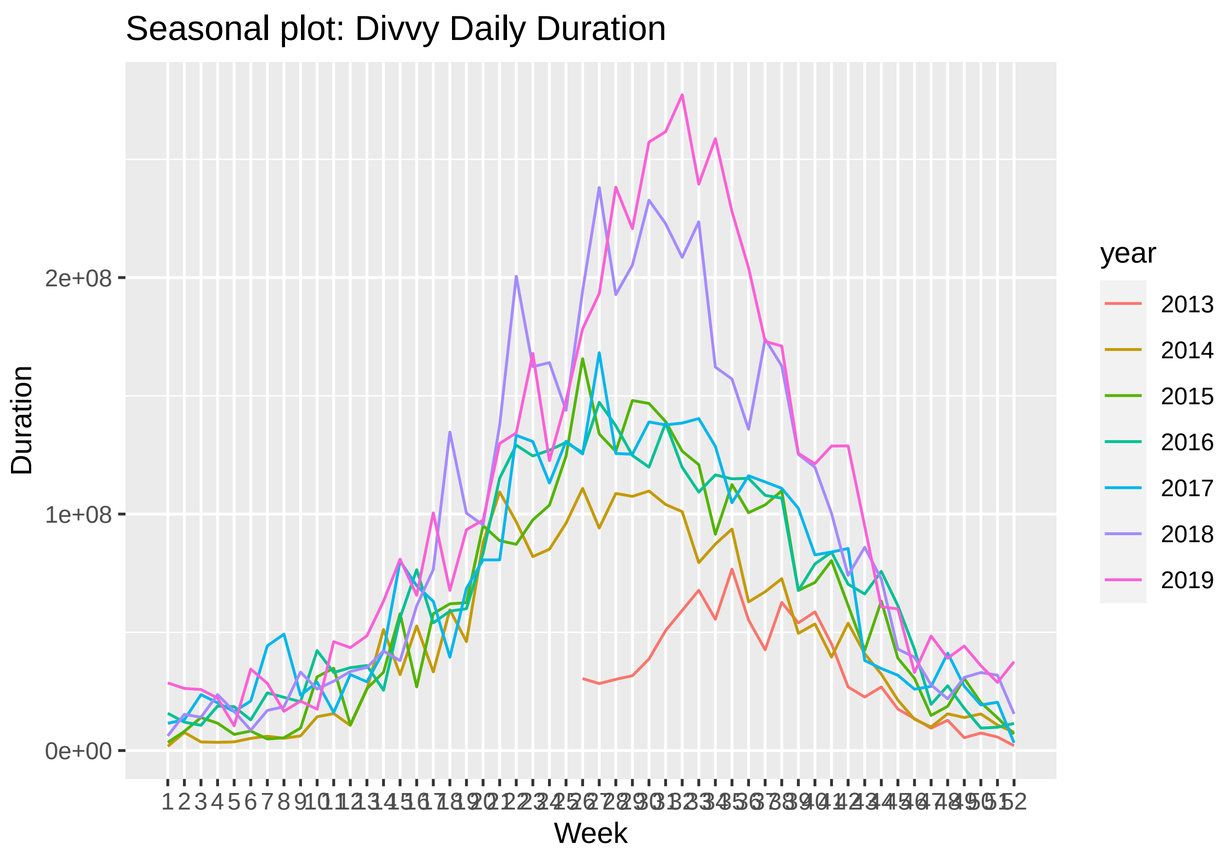


Seasonal Subseries Plots

```
ggsubseriesplot(divvy_agg_weekly) +
  ylab("Duration") +
  ggtitle("Seasonal subseries plot: Divvy Weekly Duration")
```




```r
ggseasonplot(divvy_agg_weekly) +
  ylab("Duration") +
  ggtitle("Seasonal plot: Divvy Daily Duration")
```

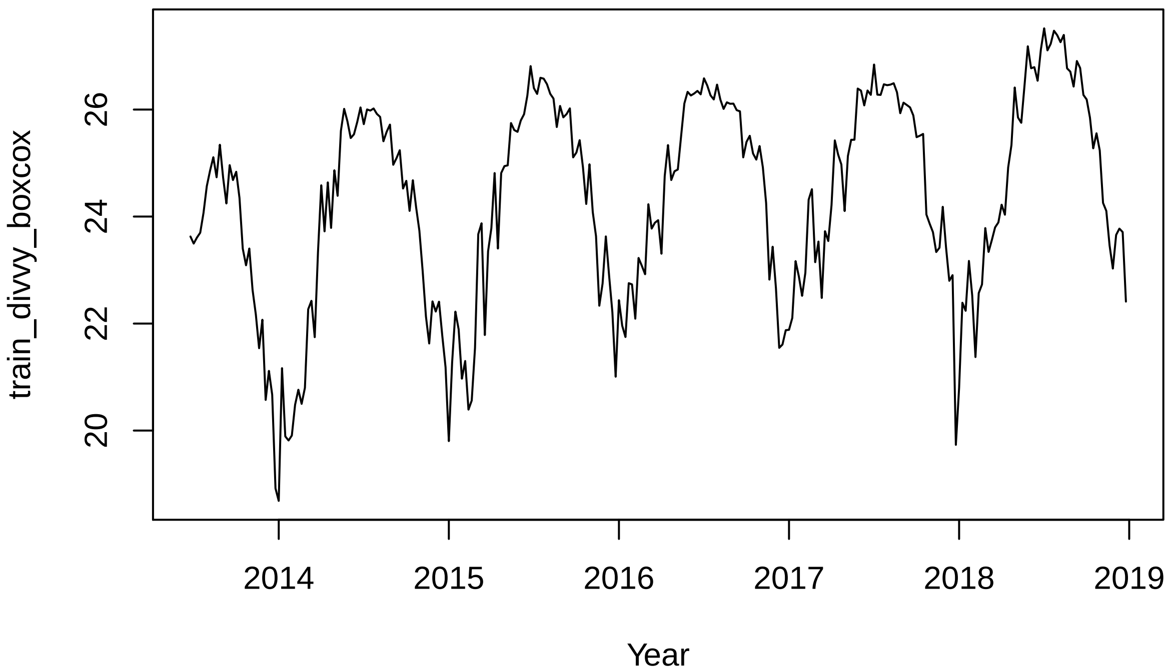


Box-Cox transformation

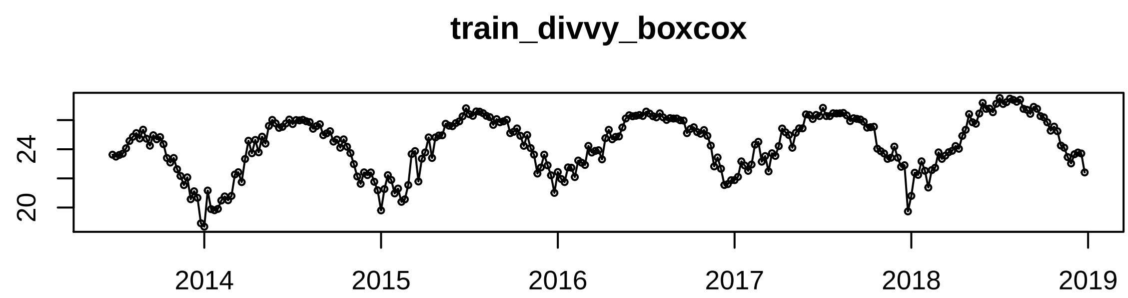
```
BoxCox.lambda(train_divvy)
```

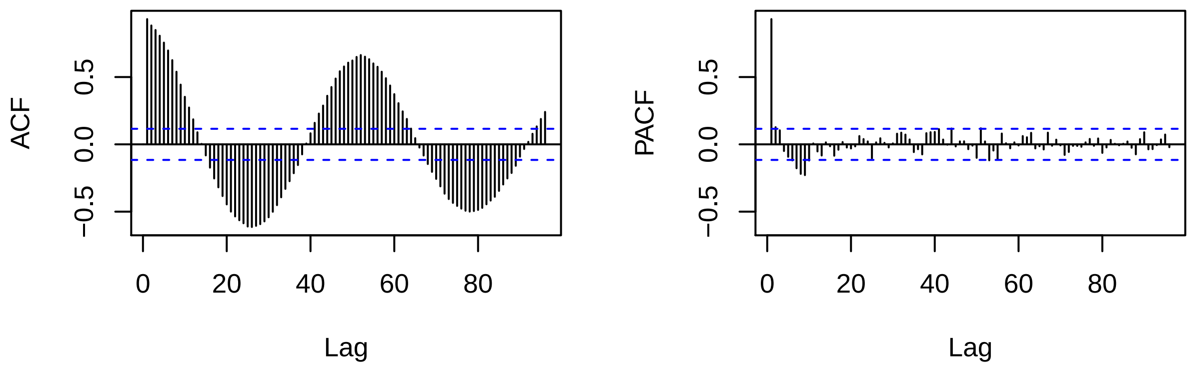
```
## [1] 0.03489689
```

```
train_divvy_boxcox <- BoxCox(train_divvy,lambda = 0.03489689)
plot(train_divvy_boxcox,plot.type="single", col=1:2, xlab="Year")
```




```
tsdisplay(train_divvy_boxcox)
```





Check if trend stationary

```
kpss.test(train_divvy_boxcox) ## small p; not trend stationary
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  train_divvy_boxcox
## KPSS Level = 0.59309, Truncation lag parameter = 5, p-value = 0.02326
```

p-value < 0.05, the null hypothesis of trend being stationary is rejected. Therefore, we apply seasonal differencing.

```
train_divvy_boxcox_seadiff <- diff(train_divvy_boxcox,lag=52)
```

Check again if the transformed series is stationary.

```
kpss.test(train_divvy_boxcox_seadiff)
```

```
## Warning in kpss.test(train_divvy_boxcox_seadiff): p-value smaller than printed
## p-value
```

```
##
##  KPSS Test for Level Stationarity
```

```
##
## data:  train_divvy_boxcox_seadiff
## KPSS Level = 0.88875, Truncation lag parameter = 4, p-value = 0.01
```

small p; trend not stationary after D=1, then we try first order differencing.

```
train_divvy_boxcox_seadiff_diff <- diff(train_divvy_boxcox_seadiff)
```

Check again if the transformed series is stationary.

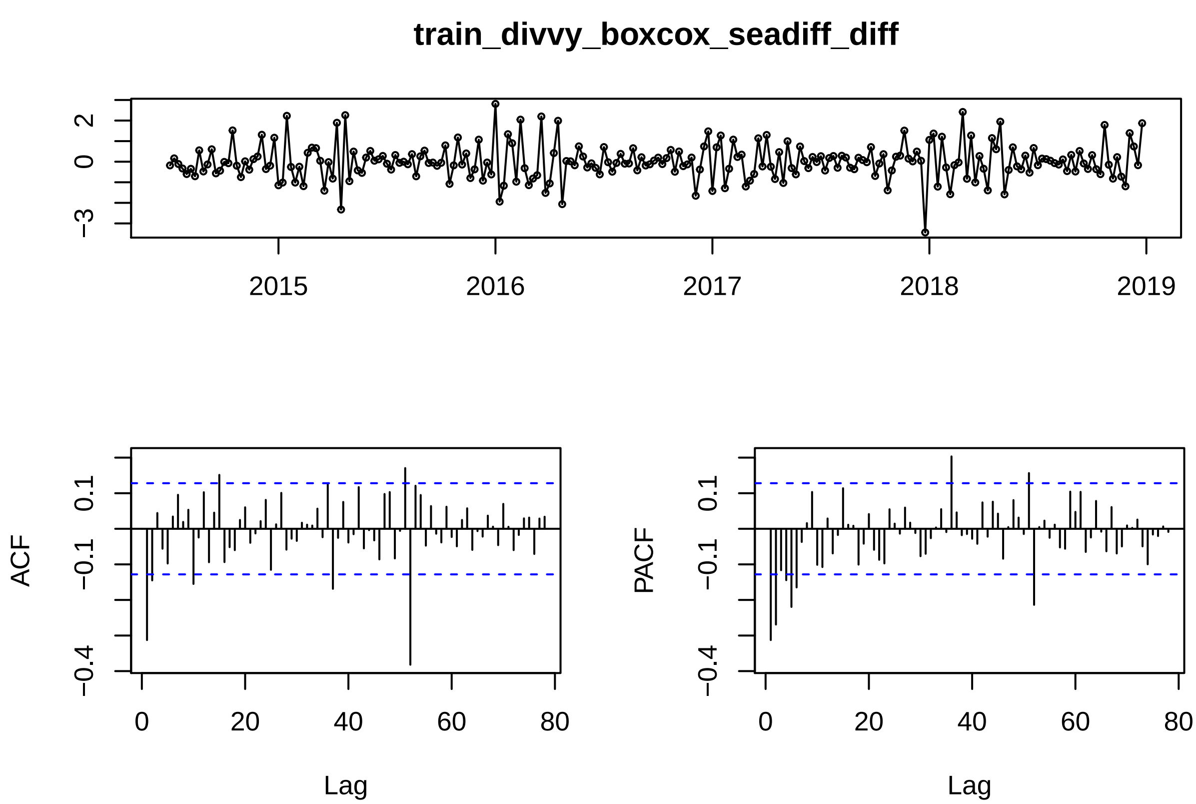
```
kpss.test(train_divvy_boxcox_seadiff_diff)
```

```
## Warning in kpss.test(train_divvy_boxcox_seadiff_diff): p-value greater than
## printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  train_divvy_boxcox_seadiff_diff
## KPSS Level = 0.08886, Truncation lag parameter = 4, p-value = 0.1
```

large p; trend stationary; d=1

```
tsdisplay(train_divvy_boxcox_seadiff_diff)
```



The divvy bike sereis needs seasonal differencing of lag = 52 and the first order differencing.

Check correlation between weather variables and divvy useage

```
cor(train_temp,train_divvy)
```

```
## [1] 0.8330095
```

```
cor(train_pre,train_divvy)
```

```
## [1] 0.1683076
```

```
cor(train_wind,train_divvy)
```

```
## [1] -0.4791209
```

```
cor(train_snow,train_divvy)
```

```
## [1] -0.2632397
```

The "divvy vs avg_temp" has a high positive correlation. The second highest correlation is "divvy vs avg_wind", which has a negative correlation. However, its Pearson r is only 0.48, indicating that the correlation is not significant. Therefore, we would only use temperature as additional variable going forward

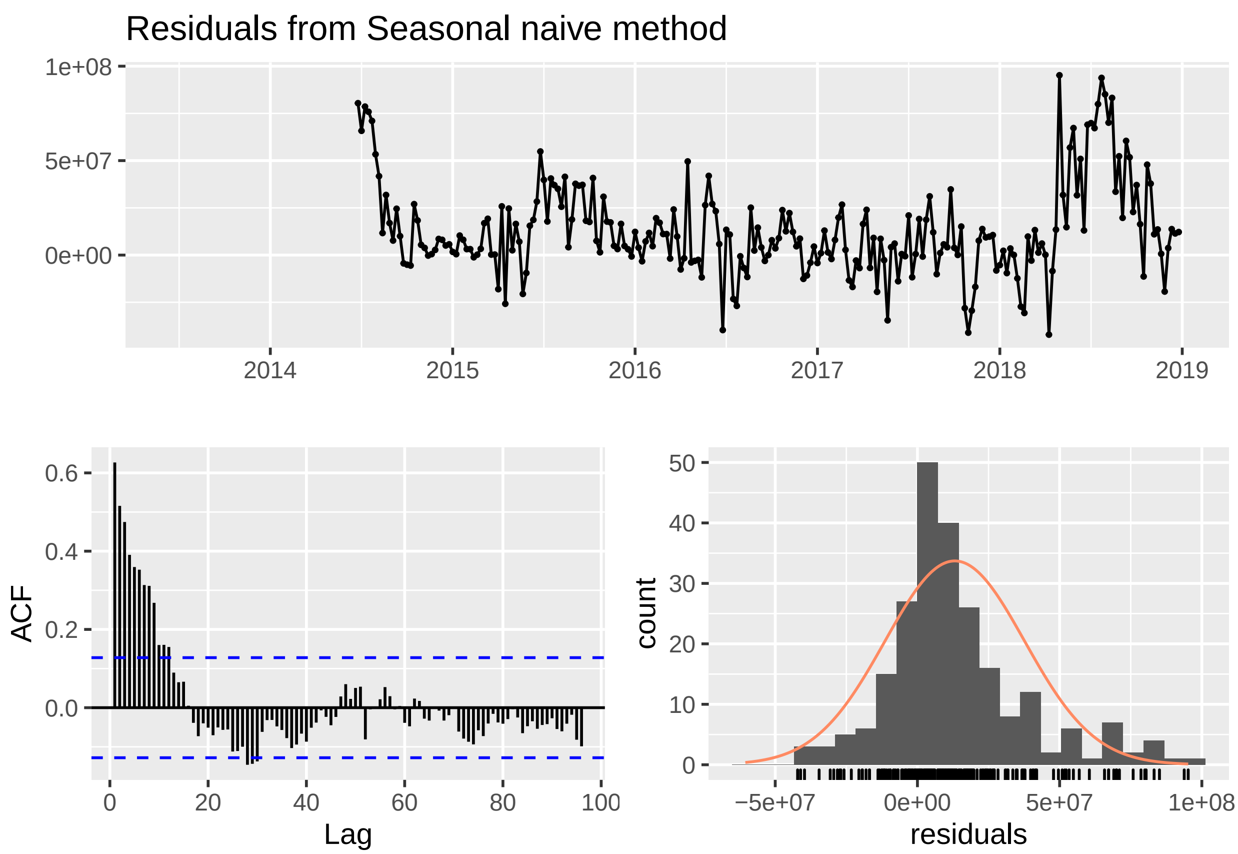
Part 3: Seasonal Naive

Build Model and get prediction

```
model_snaive <- snaive(train_divvy,h=52)

pred_snaive <- model_snaive$mean
```

Check residuals

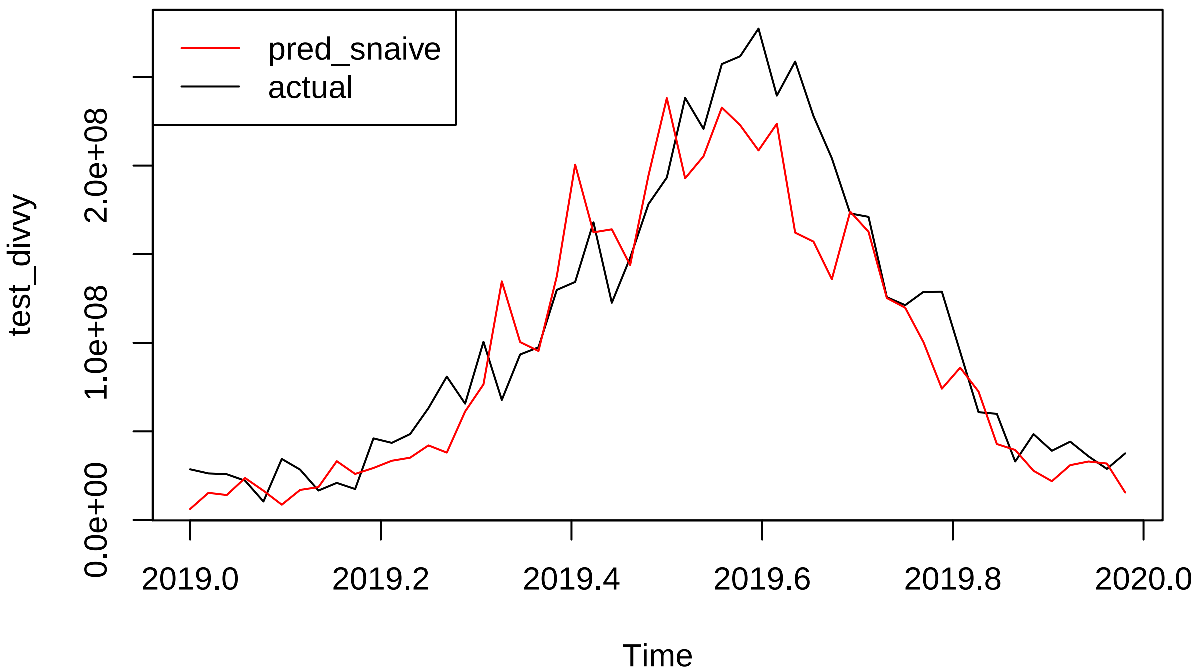
```
checkresiduals(model_snaive)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Seasonal naive method
## Q* = 452.06, df = 57, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 57
```

Make forecast and compare it with the actual test data

```
plot(test_divvy)
lines(pred_snaive,col='red')
legend('topleft',legend =c('pred_snaive','actual'),col=c('red','black'),lty=1)
```



Calculate MAPE

```
MAPE(pred_snaive,test_divvy)
```

```
## [1] 0.2752298
```

Part 4: STL + ETS

Build model

```
ets_fit <- stlf(train_divvy)
summary(ets_fit)
```

```
##
## Forecast method: STL +  ETS(A,Ad,N)
##
## Model Information:
## ETS(A,Ad,N)
##
## Call:
##  ets(y = na.interp(x), model = etsmodel, allow.multiplicative.trend = allow.multiplicative.trend)
##
##   Smoothing parameters:
##     alpha = 0.4015
```

```
##     beta  = 1e-04
##     phi   = 0.9346
##
##   Initial states:
##     l = -45225995.0126
##     b = 7244827.9108
##
##   sigma:  12469859
##
##      AIC     AICc      BIC
## 11009.71 11010.01 11031.67
##
## Error measures:
##                    ME     RMSE     MAE       MPE     MAPE      MASE       ACF1
## Training set 213037.2 12360759 9001462 -8.124831 25.64128 0.4702201 0.09706505
##
## Forecasts:
##          Point Forecast     Lo 80     Hi 80         Lo 95     Hi 95
## 2019.000       27998085  12017318  43978852   3557610.836  52438559
## 2019.019       30955263  13734177  48176349   4617884.953  57292641
## 2019.038       32842445  14464071  51220819   4735147.484  60949743
## 2019.058       35062086  15594671  54529501   5289243.615  64834928
## 2019.077       31285270  10786208  51784332    -65340.053  62635879
## 2019.096       29390426   7908864  50871987  -3462787.311  62243639
## 2019.115       37964849  15543495  60386204   3674347.071  72255352
## 2019.135       38570510  15246925  61894094   2900165.730  74240854
## 2019.154       36209308  12016871  60401745   -789831.658  73208447
## 2019.173       45461226  20429854  70492599   7179045.641  83743407
## 2019.192       41975182  16131891  67818473   2451279.372  81499084
## 2019.212       41287767  14657117  67918417    559702.540  82015831
## 2019.231       46088209  18692653  73483765   4190322.148  87986097
## 2019.250       53109055  24969226  81248885  10072899.878  96145210
## 2019.269       68791484  39926428  97656541  24646189.982 112936779
## 2019.288       72891389  43318759 102464019  27663954.763 118118824
## 2019.308       72723330  42459550 102987109  26438872.756 119007787
## 2019.327       85991637  55052038 116931236  38673603.171 133309671
## 2019.346       82817938  51216873 114419004  34488279.744 131147597
## 2019.365      101640170  69391115 133889225  52319497.012 150960844
## 2019.385      119667169  86782809 152551528  69374881.073 169959456
## 2019.404      146102022 112594327 179609716  94856425.272 197347618
## 2019.423      135550758 101431046 169670470  83369161.215 187732354
## 2019.442      134009656  99288650 168730662  80908459.619 187110852
## 2019.462      139528969 104216849 174841089  85523742.036 193534196
## 2019.481      153424477 117530924 189318031  98530025.148 208318930
## 2019.500      166917716 130451950 203383481 111148140.229 222687291
## 2019.519      147715070 110685888 184744251  91083823.941 204346315
## 2019.538      151186023 113601830 188770216  93705959.891 208666086
## 2019.558      159049909 120918744 197181074 100733324.276 217366494
## 2019.577      159102085 120431648 197772521  99960755.301 218243414
## 2019.596      150615587 111413264 189817910  90660808.413 210570366
## 2019.615      148596270 108869152 188323388  87838886.556 209353654
## 2019.635      129721545  89476447 169966642  68171978.872 191271110
## 2019.654      129636678  88880157 170393198  67304958.458 191968397
## 2019.673      119205301  77943672 160466929  56101085.939 182309515
```

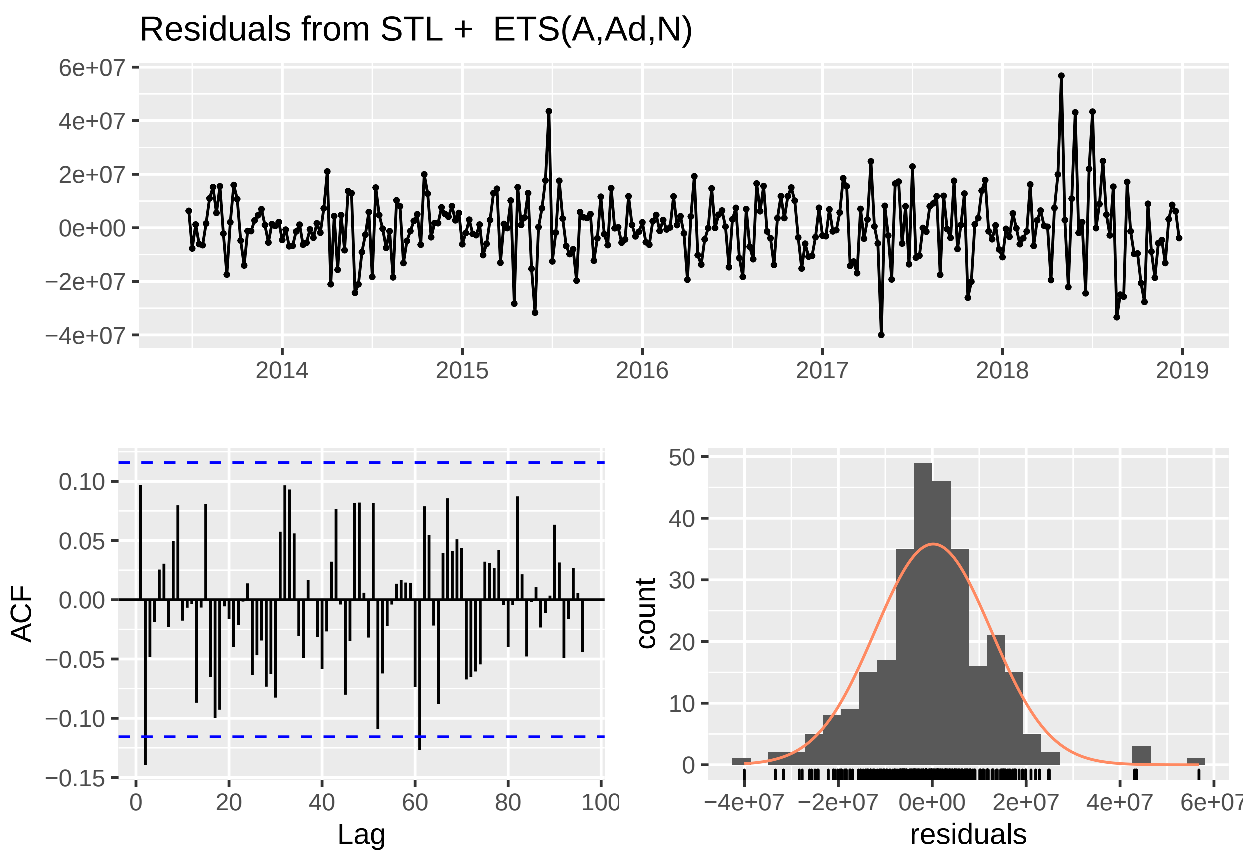
```
## 2019.692       124816827  83056178 166577475  60949426.355 188684227
## 2019.712       124991557  82737761 167245352  60369952.695 189613160
## 2019.731        96507326  53766054 139248597  31140191.819 161874460
## 2019.750        94831719  51608452 138054986  28727436.075 160936001
## 2019.769        90267655  46567693 133967618  23434330.144 157100981
## 2019.788        79375354  35203826 123546883  11820831.420 146929877
## 2019.808        65681138  21043012 110319264  -2586984.912 133949260
## 2019.827        66520519  21420610 111620428  -2453839.851 135494878
## 2019.846        50181839   4624816  95738863 -19491615.399 119855294
## 2019.865        41424581  -4585025  87434188 -28941040.053 111790203
## 2019.885        30821250 -15636542  77279041 -40229810.643 101872310
## 2019.904        35359201 -11542501  82260904 -36370762.646 107089165
## 2019.923        33313116 -14028344  80654575 -39089398.766 105715630
## 2019.942        28752696 -19024481  76529873 -44316191.263 101821583
## 2019.962        26549023 -21659941  74757987 -47180225.425 100278271
## 2019.981        17744512 -30892412  66381437 -56639244.858  92128269
## 2020.000        27948736 -21112422  77009895 -47083830.313 102981303
## 2020.019        30909144 -18572617  80390905 -44766678.489 106584967
## 2020.038        32799344 -17099479  82698168 -43514320.563 109113009
## 2020.058        35021805 -15290629  85334239 -41924422.274 111968033
## 2020.077        31247625 -19475051  81970302 -46326013.951 108821264
## 2020.096        29355245 -21774387  80484876 -48840778.504 107551268
## 2020.115        37931971 -13601406  89465348 -40881528.268 116745470
## 2020.135        38539783 -13394205  90473770 -40886397.340 117965962
## 2020.154        36180591 -16150943  88512126 -43853584.481 116214767
## 2020.173        45434389  -7291699  98160477 -35203203.375 126071982
## 2020.192        41950101 -11167612  95067814 -39286430.902 123186633
## 2020.212        41264327 -12242147  94770802 -40566764.206 123095419
## 2020.231        46066304  -7826131  99958738 -36355062.520 128487670
## 2020.250        53088583  -1187069 107364235 -29918864.181 136096030
## 2020.269        68772352  14116167 123428537 -14817070.174 152361774
## 2020.288        72873509  17839420 127907598 -11293867.423 157040885
## 2020.308        72706619  17297202 128116037 -12034772.266 157448011
## 2020.327        85976020  30193798 141758243    664472.601 171287568
## 2020.346        82803344  26650790 138955898  -3074576.831 168681264
## 2020.365       101626531  45106070 158146991  15185945.802 188067116
## 2020.385       119654421  62768432 176540410  32654808.920 206654034
## 2020.404       146090109  88840924 203339294  58535036.011 233645182
## 2020.423       135539624  77929532 193149717  47432591.135 223646658
## 2020.442       133999251  76030497 191968005  45343691.951 222654810
## 2020.462       139519245  81194034 197844456  50318531.753 228719958
## 2020.481       153415390  94735886 212094894  63672832.540 243157947
## 2020.500       166909223 107877551 225940895  76628071.993 257190374
## 2020.519       147707132  88325381 207088884  56890580.736 238523684
## 2020.538       151178605  91448825 210908385  59829789.484 242527421
## 2020.558       159042977  98967184 219118770  67164979.538 250920974
## 2020.577       159095606  98675780 219515432  66691456.441 251499755
## 2020.596       150609532  89847622 211371443  57682208.914 243536856
## 2020.615       148590612  87488530 209692693  55143042.370 242038181
## 2020.635       129716256  68275887 191156626  35751320.487 223681192
## 2020.654       129631736  67854931 191408541  35152265.503 224111206
## 2020.673       119200682  57089263 181312101  24209464.051 214191900
## 2020.692       124812510  62368270 187256751  29312286.045 220312735
## 2020.712       124987523  62212225 187762820  28980990.016 220994055
```

```
## 2020.731       96503556  33398938 159608174     -6629.369 193013741
## 2020.750       94828195  31395966 158260425  -2183028.245 191839419
## 2020.769       90264363  26506205 154022521  -7245325.410 187774051
## 2020.788       79372277  15289848 143454706 -18633340.778 177377895
## 2020.808       65678262   1273194 130083330 -32820789.242 164177313
## 2020.827       66517832   1791733 131243930 -32472193.695 165507857
## 2020.846       50179328 -14866217 115224872 -49299248.983 149657904
## 2020.865       41422234 -23941197 106785664 -58542506.600 141386974
## 2020.885       30819056 -34860722  96498834 -69629495.818 131267608
## 2020.904       35357151 -30637458 101351760 -65572893.336 136287195
## 2020.923       33311200 -32996745  99619145 -68098051.265 134720451
## 2020.942       28750905 -37868903  95370713 -73135298.906 130637109
## 2020.962       26547349 -40382869  93477567 -75813585.715 128908284
## 2020.981       17742948 -49496246  84982143 -85090526.138 120576423
```

Check residuals

```
checkresiduals(ets_fit)
```

```
## Warning in checkresiduals(ets_fit): The fitted degrees of freedom is based on
## the model used for the seasonally adjusted data.
```




```
##
##  Ljung-Box test
```

```
##
## data:  Residuals from STL +  ETS(A,Ad,N)
## Q* = 61.321, df = 52, p-value = 0.1764
##
## Model df: 5.   Total lags used: 57
```

Calculate MAPE

```
MAPE(ets_fit$mean,test_divvy)
```

```
## [1] 0.3259718
```

Part 5: Seasonal ARIMA

Build Model with auto.arima

```
arima <- auto.arima(train_divvy,seasonal = 'TRUE', lambda = 'auto',trace=TRUE,approximation = FALSE)
```

```
##
##  ARIMA(2,1,2)(1,1,1)[52]                    : Inf
##  ARIMA(0,1,0)(0,1,0)[52]                    : 597.6318
##  ARIMA(1,1,0)(1,1,0)[52]                    : 531.3113
##  ARIMA(0,1,1)(0,1,1)[52]                    : Inf
##  ARIMA(1,1,0)(0,1,0)[52]                    : 575.1696
##  ARIMA(1,1,0)(1,1,1)[52]                    : Inf
##  ARIMA(1,1,0)(0,1,1)[52]                    : Inf
##  ARIMA(0,1,0)(1,1,0)[52]                    : 547.5909
##  ARIMA(2,1,0)(1,1,0)[52]                    : 514.1516
##  ARIMA(2,1,0)(0,1,0)[52]                    : 559.0914
##  ARIMA(2,1,0)(1,1,1)[52]                    : Inf
##  ARIMA(2,1,0)(0,1,1)[52]                    : Inf
##  ARIMA(3,1,0)(1,1,0)[52]                    : 508.9005
##  ARIMA(3,1,0)(0,1,0)[52]                    : 557.4164
##  ARIMA(3,1,0)(1,1,1)[52]                    : Inf
##  ARIMA(3,1,0)(0,1,1)[52]                    : Inf
##  ARIMA(4,1,0)(1,1,0)[52]                    : 507.9322
##  ARIMA(4,1,0)(0,1,0)[52]                    : 552.9106
##  ARIMA(4,1,0)(1,1,1)[52]                    : Inf
##  ARIMA(4,1,0)(0,1,1)[52]                    : Inf
##  ARIMA(5,1,0)(1,1,0)[52]                    : 500.3405
##  ARIMA(5,1,0)(0,1,0)[52]                    : 540.5245
##  ARIMA(5,1,0)(1,1,1)[52]                    : Inf
##  ARIMA(5,1,0)(0,1,1)[52]                    : Inf
##  ARIMA(5,1,1)(1,1,0)[52]                    : 495.9344
##  ARIMA(5,1,1)(0,1,0)[52]                    : 532.9529
##  ARIMA(5,1,1)(1,1,1)[52]                    : Inf
##  ARIMA(5,1,1)(0,1,1)[52]                    : Inf
##  ARIMA(4,1,1)(1,1,0)[52]                    : Inf
##  ARIMA(5,1,2)(1,1,0)[52]                    : 493.8081
##  ARIMA(5,1,2)(0,1,0)[52]                    : 535.5424
##  ARIMA(5,1,2)(1,1,1)[52]                    : Inf
##  ARIMA(5,1,2)(0,1,1)[52]                    : Inf
```

```
##  ARIMA(4,1,2)(1,1,0)[52]                    : Inf
##  ARIMA(5,1,3)(1,1,0)[52]                    : 492.5909
##  ARIMA(5,1,3)(0,1,0)[52]                    : Inf
##  ARIMA(5,1,3)(1,1,1)[52]                    : Inf
##  ARIMA(5,1,3)(0,1,1)[52]                    : Inf
##  ARIMA(4,1,3)(1,1,0)[52]                    : 490.1978
##  ARIMA(4,1,3)(0,1,0)[52]                    : 536.9072
##  ARIMA(4,1,3)(1,1,1)[52]                    : Inf
##  ARIMA(4,1,3)(0,1,1)[52]                    : Inf
##  ARIMA(3,1,3)(1,1,0)[52]                    : Inf
##  ARIMA(4,1,4)(1,1,0)[52]                    : 493.2381
##  ARIMA(3,1,2)(1,1,0)[52]                    : Inf
##  ARIMA(3,1,4)(1,1,0)[52]                    : Inf
##  ARIMA(5,1,4)(1,1,0)[52]                    : Inf
##
##  Best model: ARIMA(4,1,3)(1,1,0)[52]
```

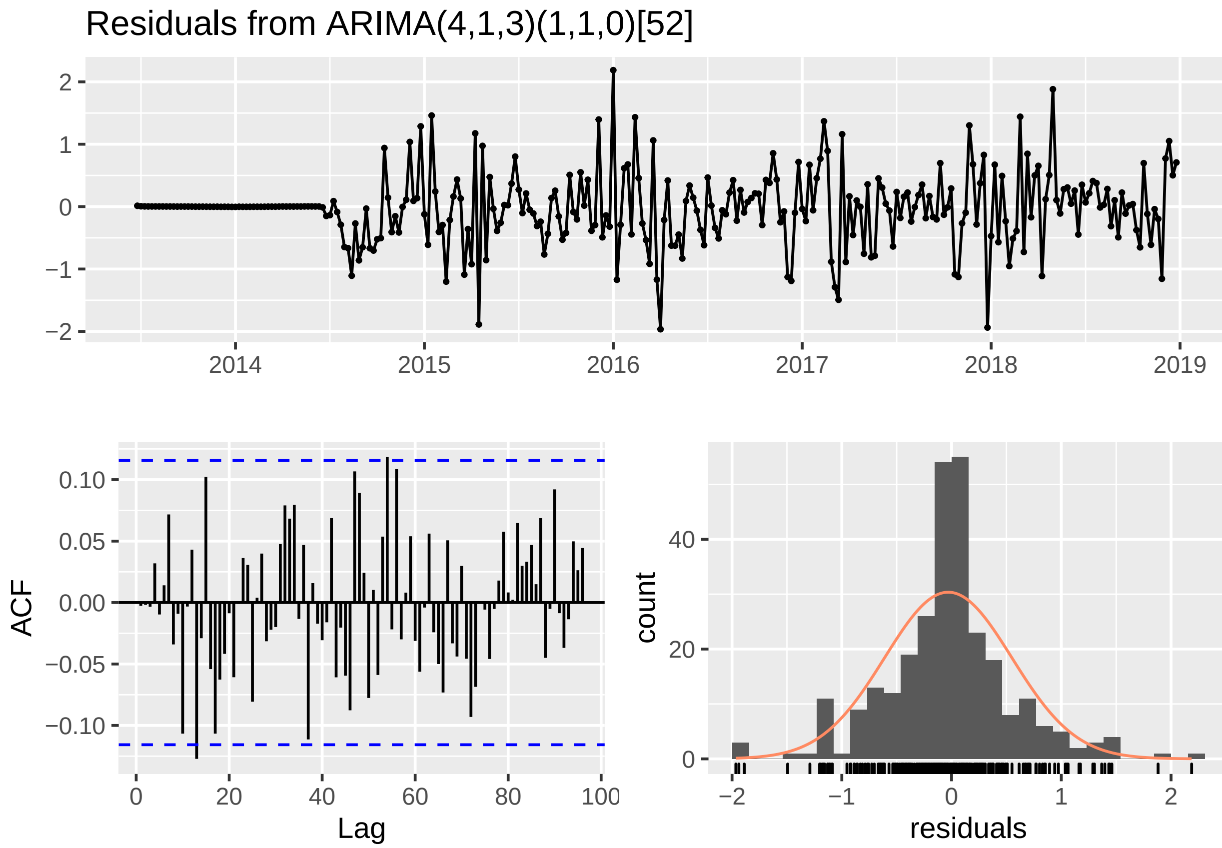
```
arima
```

```
## Series: train_divvy
## ARIMA(4,1,3)(1,1,0)[52]
## Box Cox transformation: lambda= 0.03488702
##
## Coefficients:
##          ar1      ar2     ar3      ar4      ma1     ma2      ma3     sar1
##       1.1765  -1.2798  0.3741  -0.1505  -1.6976  1.6812  -0.7769  -0.4566
## s.e.  0.1293   0.1099  0.1202   0.0965   0.1315  0.1210   0.1363   0.0733
##
## sigma^2 estimated as 0.427:  log likelihood=-235.7
## AIC=489.39   AICc=490.2   BIC=520.49
```

(4,1,3) (1,1,0) period=52

Check residuals

```
checkresiduals(arima)
```

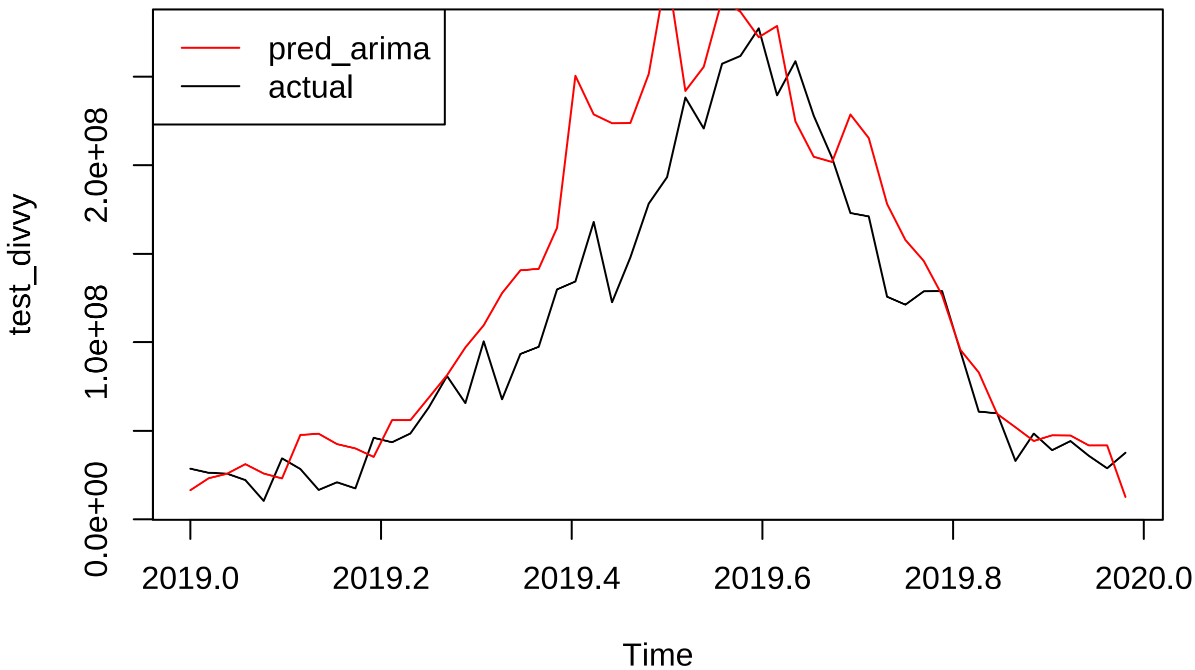



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(4,1,3)(1,1,0)[52]
## Q* = 65.523, df = 49, p-value = 0.05738
##
## Model df: 8.   Total lags used: 57
```

P=0.0574; Barely not reject the null hypothesis

Make Predictions and Compare with the actual test data

```
pred_arima <- forecast(arima,h=52)
plot(test_divvy)
lines(pred_arima$mean,col='red')
legend('topleft',legend =c('pred_arima','actual'),col=c('red','black'),lty=1)
```



Calculate MAPE

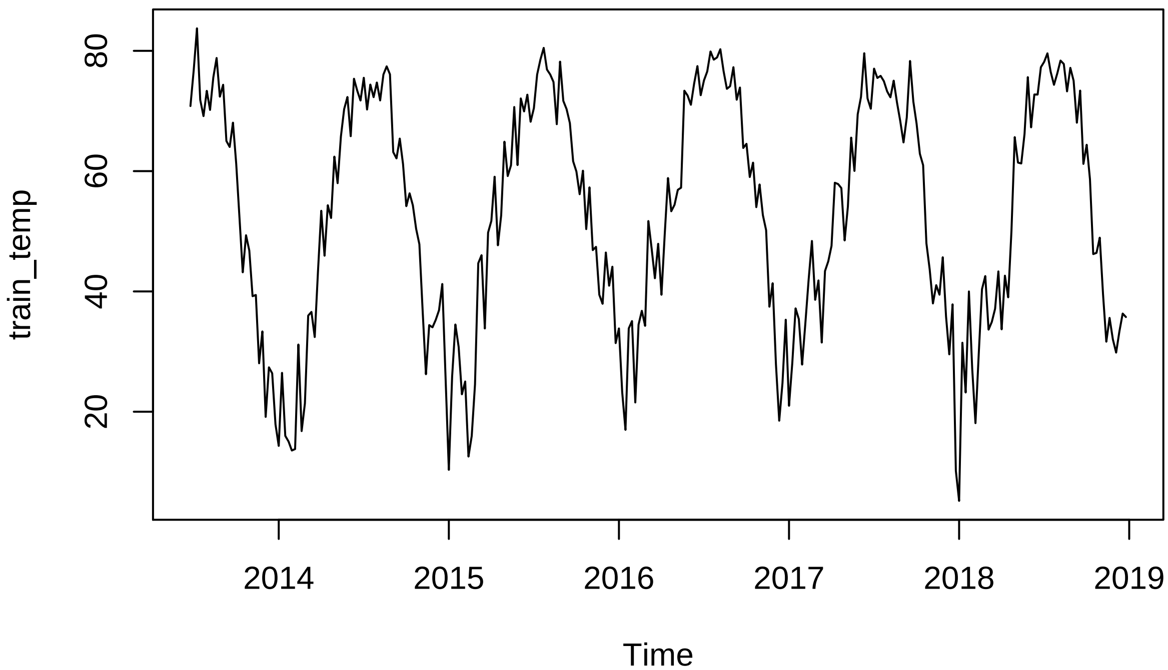
```
MAPE(pred_arima$mean,test_divvy)
```

```
## [1] 0.376772
```

Part 6: Regression with ARIMA error (with Temp)

Plot temp data and see if Boxcox is needed

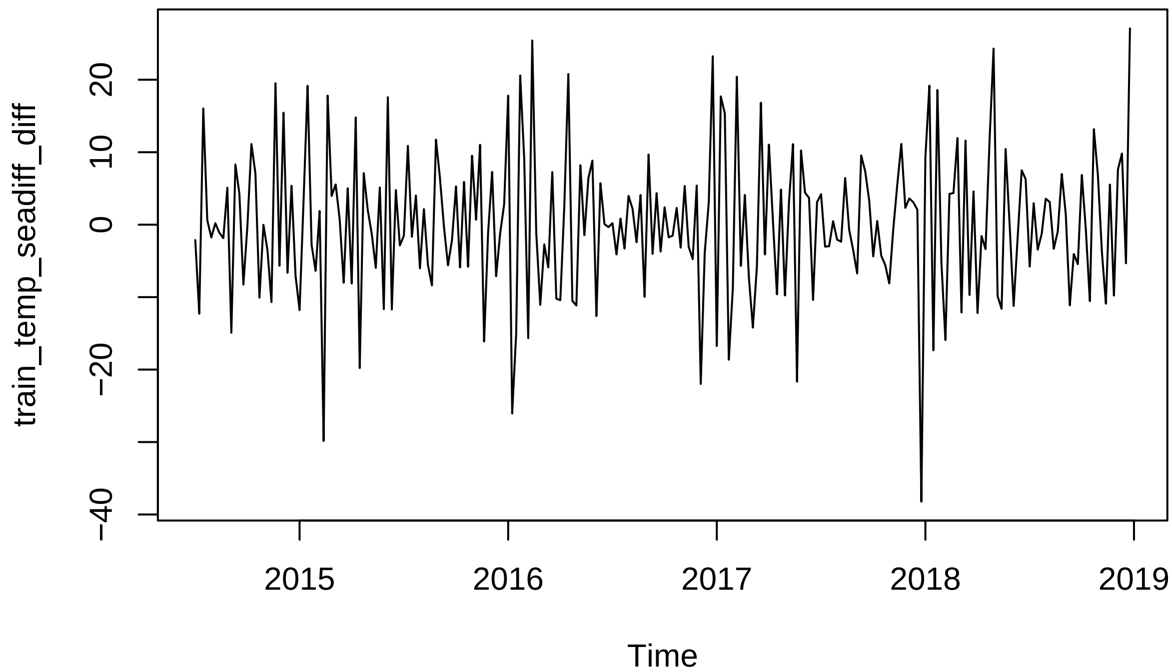
```
plot(train_temp)
```



No need for BoxCox

Use Seasonal Diff and First Order Diff to remove and Seasonality and make it trend stationary

```
train_temp_seadiff_diff <- diff(diff(train_temp,lag=52))

plot(train_temp_seadiff_diff)
```




```
kpss.test(train_temp_seadiff_diff)
```

```
## Warning in kpss.test(train_temp_seadiff_diff): p-value greater than printed p-
## value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  train_temp_seadiff_diff
## KPSS Level = 0.039703, Truncation lag parameter = 4, p-value = 0.1
```

larger p indicating trend is stationary

Prepare new training data for model fitting

```
train_divvy_new <- train_divvy[54:287]

train_divvy_new <- ts(train_divvy_new,frequency = 52,start=c(2014,27))
```

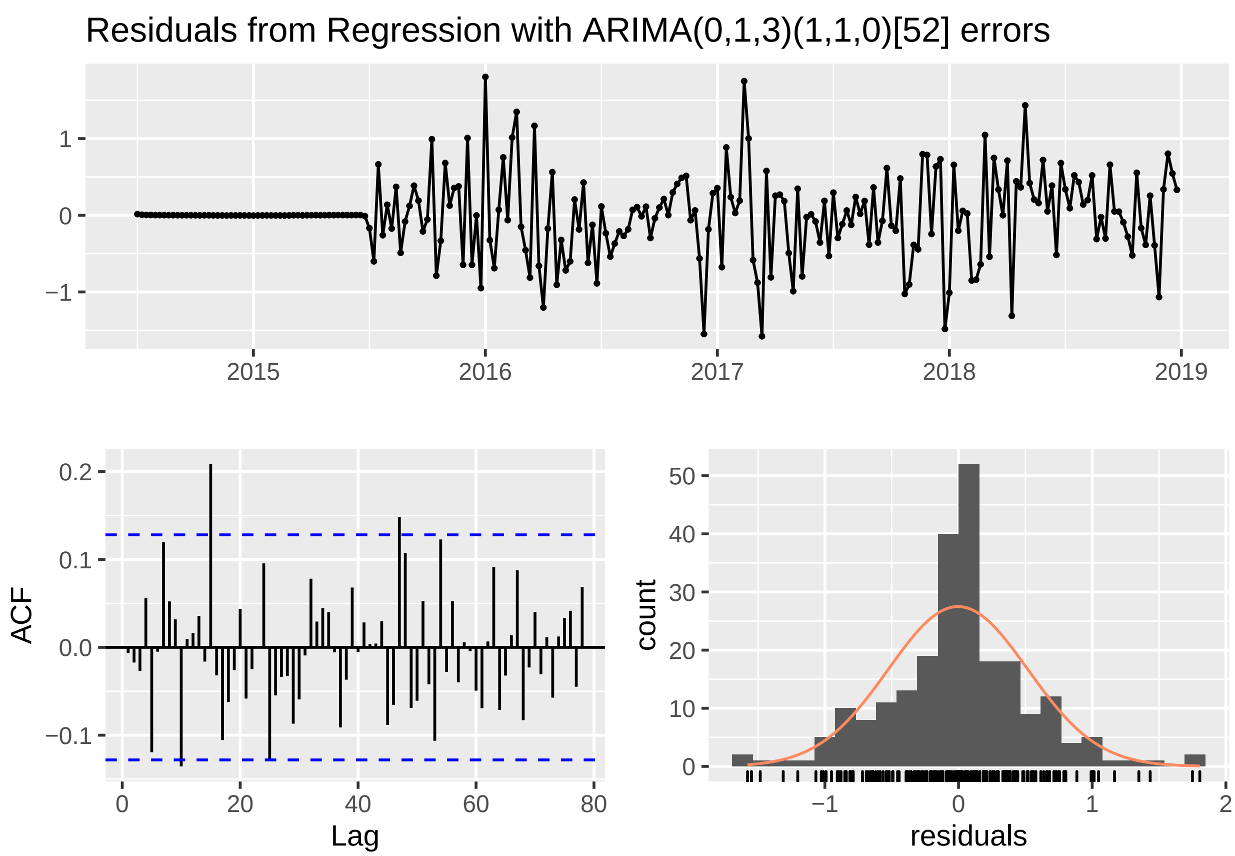
Fit model using xreg = train_temp_seadiff_diff (stationary)

```
model_arimae <- auto.arima(train_divvy_new,xreg=train_temp_seadiff_diff,lambda = 0.0349,d=1,D=1)
model_arimae
```

```
## Series: train_divvy_new
## Regression with ARIMA(0,1,3)(1,1,0)[52] errors
## Box Cox transformation: lambda= 0.0349
##
## Coefficients:
##           ma1      ma2      ma3     sar1    xreg
##       -0.4921  -0.3048  -0.0695  -0.3671  0.0161
## s.e.   0.0775   0.0738   0.0732   0.0853  0.0025
##
## sigma^2 estimated as 0.3611:  log likelihood=-166.44
## AIC=344.87   AICc=345.35   BIC=364.06
```

Check Residuals

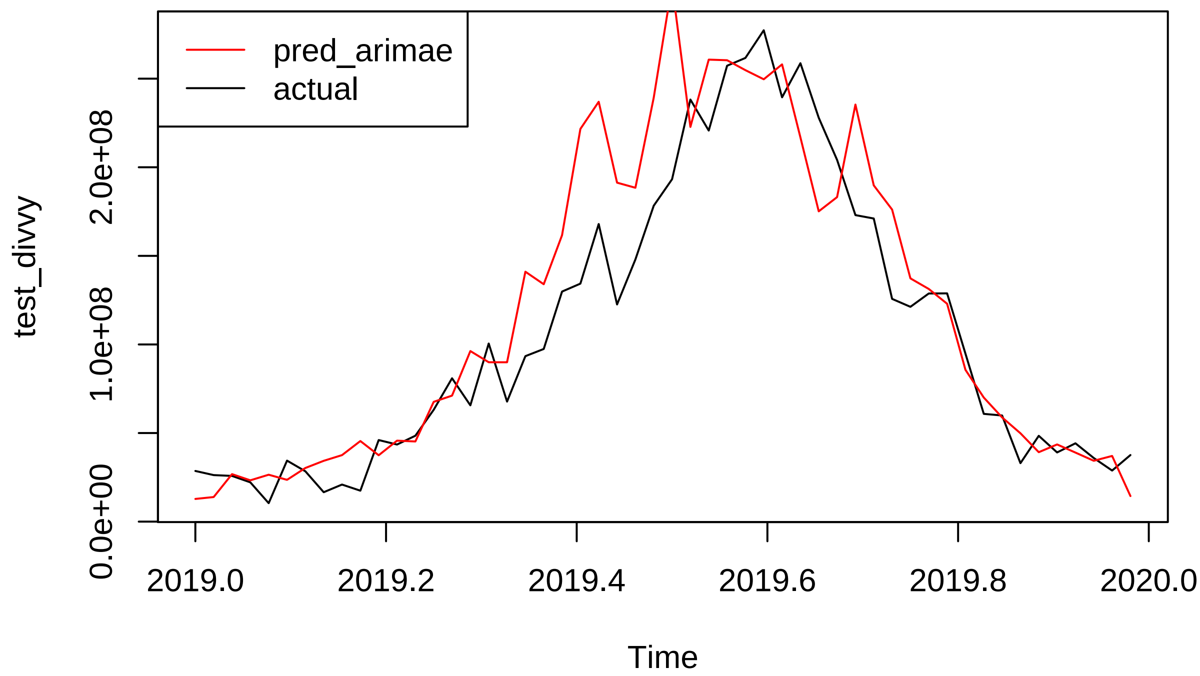
```
checkresiduals(model_arimae)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(0,1,3)(1,1,0)[52] errors
## Q* = 59.302, df = 42, p-value = 0.04025
##
## Model df: 5.   Total lags used: 47
```

Make Predictions and Compare with the actual test data

```
test_temp_new <- c(train_temp[235:287],test_temp)
pred_arimae <- forecast(model_arimae,xreg=diff(diff(test_temp_new,lag=52)),h=52)
plot(test_divvy)
lines(pred_arimae$mean,col='red')
legend('topleft',legend =c('pred_arimae','actual'),col=c('red','black'),lty=1)
```



Calculate MAPE

```
MAPE(pred_arimae$mean,test_divvy)
```

```
## [1] 0.3077718
```

Part 7: Vector AutoRegression (VAR)

```
library(vars)
data = cbind(divvy=train_divvy,temp=train_temp)
```

Check what lag order is appropriate for VAR model

```
VARselect(data,lag.max=5,type='both')$selection
```

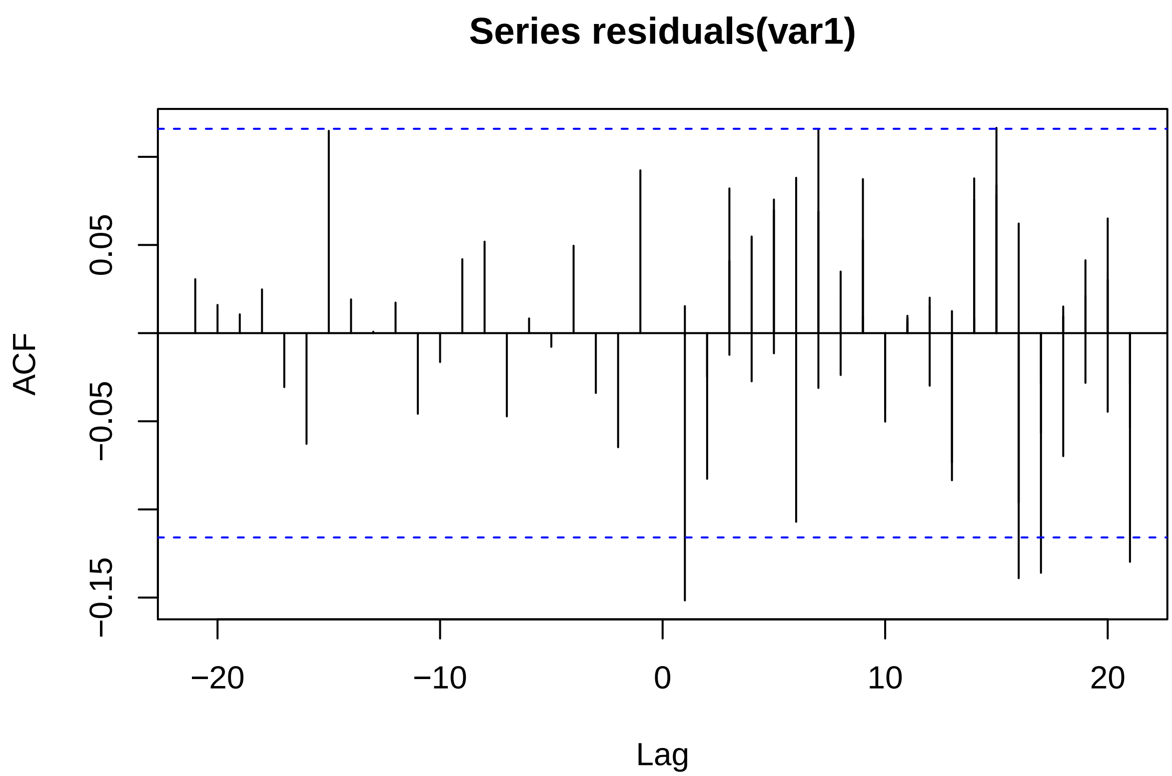
```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      3      3      1      3
```

Fit model with lag = 1 and check residual independence

```
var1 <- VAR(data,p=1,type='both',season = 52)

serial.test(var1,lags.pt=10,type='PT.asymptotic')
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object var1
## Chi-squared = 52.23, df = 36, p-value = 0.03927
```

```
acf(residuals(var1))
```

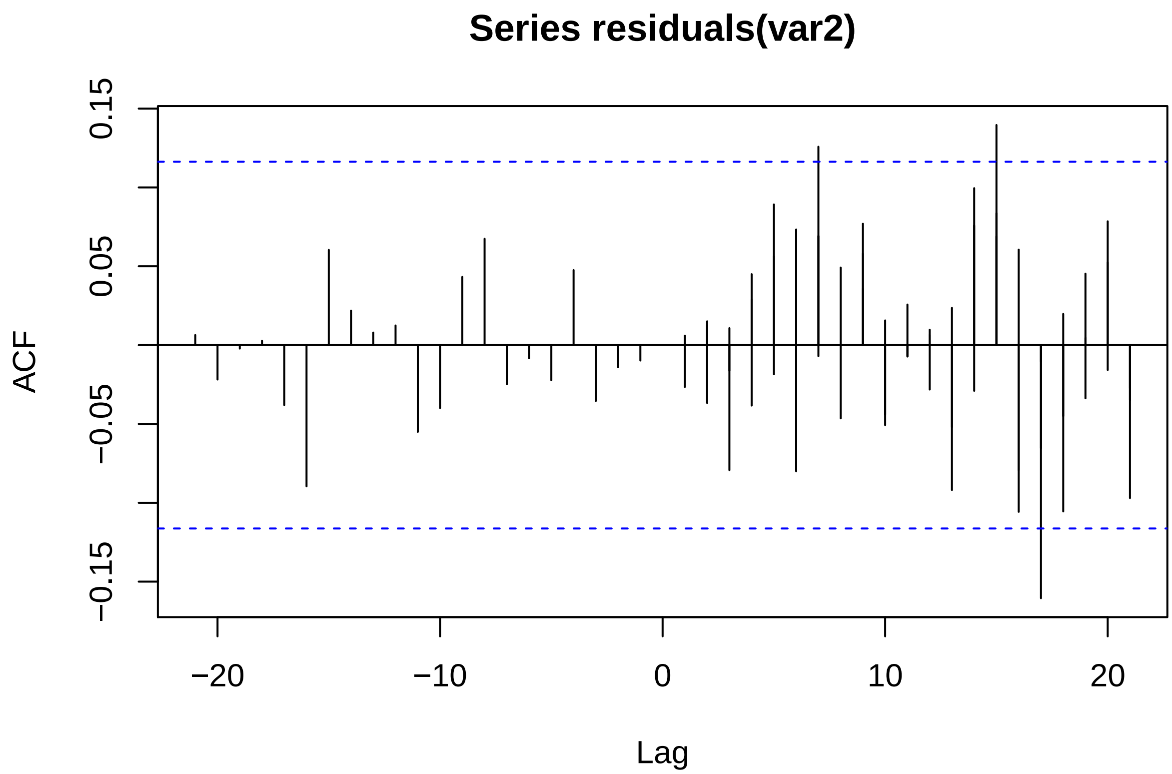


Fit model with lag = 3 and check residual independence

```
var2 <- VAR(data,p=3,type='both',season = 52)

serial.test(var2,lags.pt=10,type='PT.asymptotic')
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object var2
## Chi-squared = 30.372, df = 28, p-value = 0.3456
```

```
acf(residuals(var2))
```

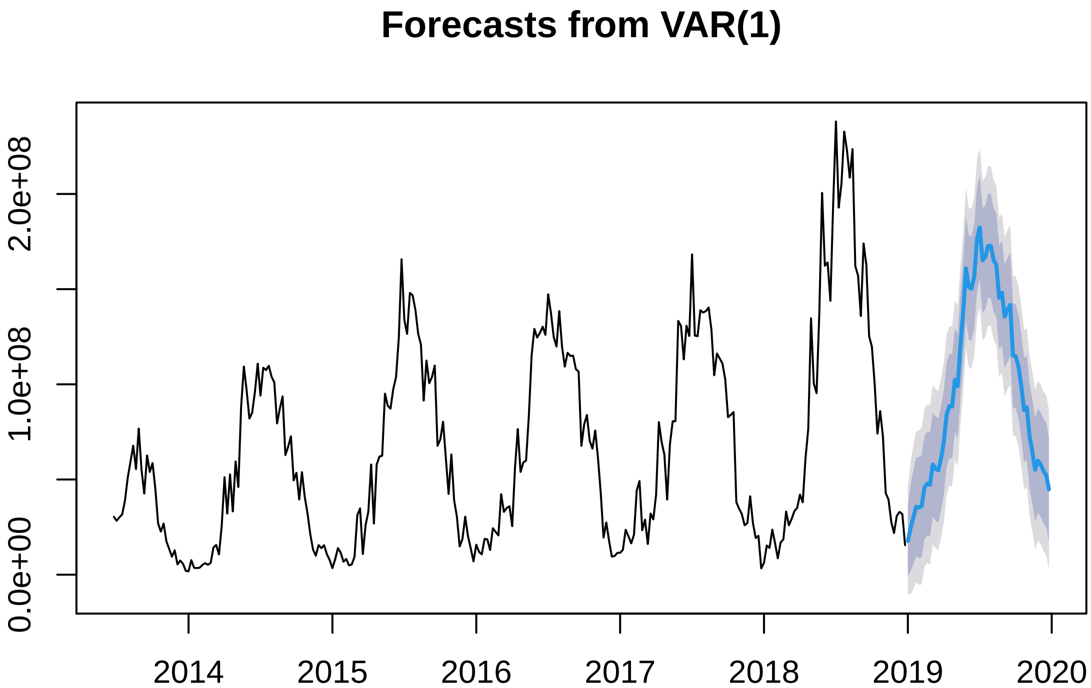


In both cases, residuals are not white noise due to small p values from serial test and spike in acf.

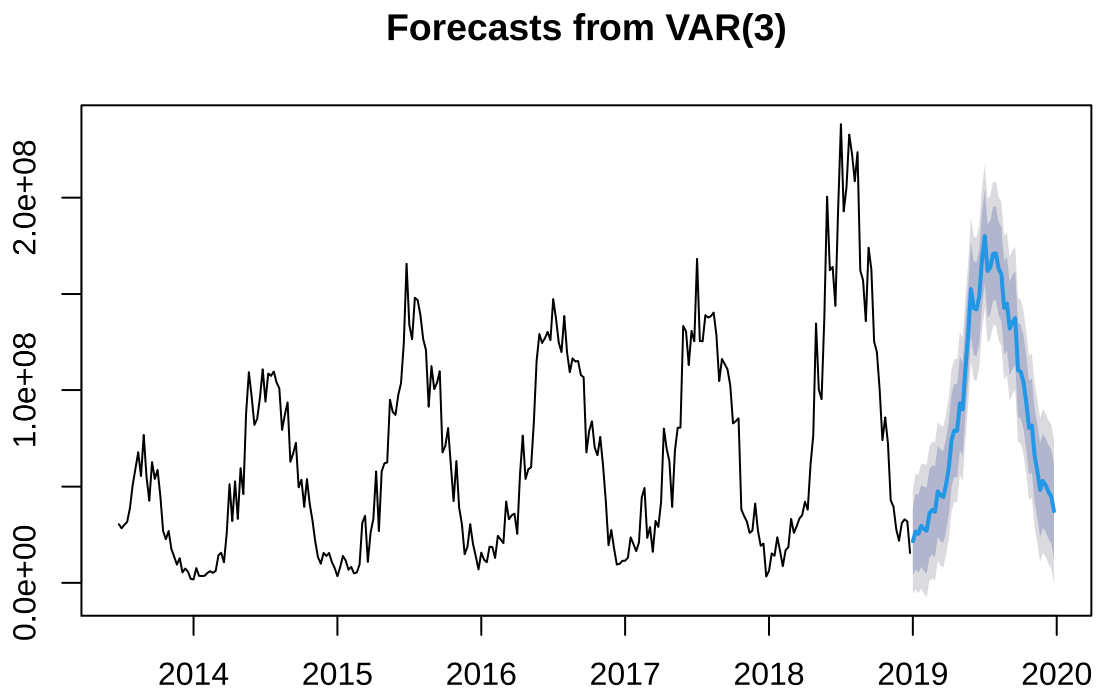
Make predictions and Compare with the actual test data for both models

```
pred_var1 <- forecast(var1,h=52)
pred_var2 <- forecast(var2,h=52)
```

```
plot(pred_var1$forecast$divvy)
```




```
plot(pred_var2$forecast$divvy)
```




```
MAPE(pred_var1$forecast$divvy$mean,test_divvy)
```

```
## [1] 0.3989797
```

```
MAPE(pred_var2$forecast$divvy$mean,test_divvy)
```

```
## [1] 0.2948619
```

Part 8: Fourier Transform

Let's first check the periodogram of train_divvy dataset.

```
kpss.test(train_divvy_boxcox)
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  train_divvy_boxcox
## KPSS Level = 0.59309, Truncation lag parameter = 5, p-value = 0.02326
```

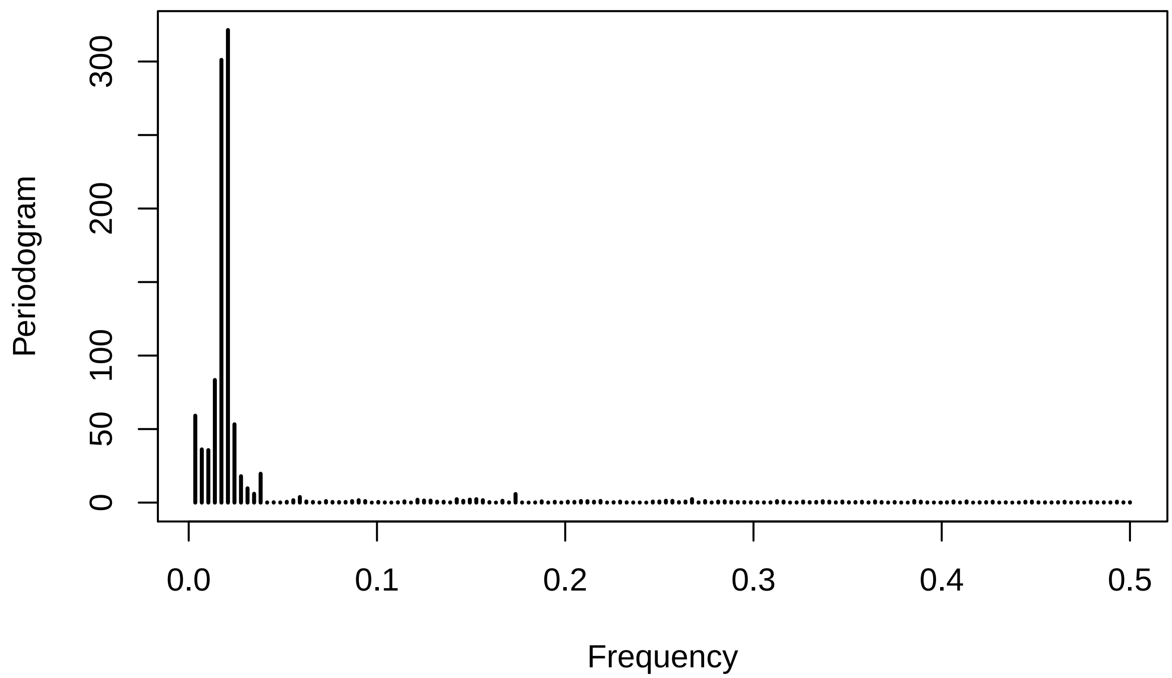
```
adf.test(train_divvy_boxcox)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  train_divvy_boxcox
## Dickey-Fuller = -3.7341, Lag order = 6, p-value = 0.02273
## alternative hypothesis: stationary
```

The Kpss test shows that the series is trend stationary ($p>0.05$).

The adf test shows that the data is stationary ($p<0.05$).

```
periodogram(train_divvy_boxcox)
temp <- periodogram(train_divvy_boxcox)
```




```
temp$freq
```

```
##   [1] 0.003472222 0.006944444 0.010416667 0.013888889 0.017361111 0.020833333
##   [7] 0.024305556 0.027777778 0.031250000 0.034722222 0.038194444 0.041666667
##  [13] 0.045138889 0.048611111 0.052083333 0.055555556 0.059027778 0.062500000
##  [19] 0.065972222 0.069444444 0.072916667 0.076388889 0.079861111 0.083333333
##  [25] 0.086805556 0.090277778 0.093750000 0.097222222 0.100694444 0.104166667
##  [31] 0.107638889 0.111111111 0.114583333 0.118055556 0.121527778 0.125000000
##  [37] 0.128472222 0.131944444 0.135416667 0.138888889 0.142361111 0.145833333
##  [43] 0.149305556 0.152777778 0.156250000 0.159722222 0.163194444 0.166666667
##  [49] 0.170138889 0.173611111 0.177083333 0.180555556 0.184027778 0.187500000
```

```
##  [55] 0.190972222 0.194444444 0.197916667 0.201388889 0.204861111 0.208333333
##  [61] 0.211805556 0.215277778 0.218750000 0.222222222 0.225694444 0.229166667
##  [67] 0.232638889 0.236111111 0.239583333 0.243055556 0.246527778 0.250000000
##  [73] 0.253472222 0.256944444 0.260416667 0.263888889 0.267361111 0.270833333
##  [79] 0.274305556 0.277777778 0.281250000 0.284722222 0.288194444 0.291666667
##  [85] 0.295138889 0.298611111 0.302083333 0.305555556 0.309027778 0.312500000
##  [91] 0.315972222 0.319444444 0.322916667 0.326388889 0.329861111 0.333333333
##  [97] 0.336805556 0.340277778 0.343750000 0.347222222 0.350694444 0.354166667
## [103] 0.357638889 0.361111111 0.364583333 0.368055556 0.371527778 0.375000000
## [109] 0.378472222 0.381944444 0.385416667 0.388888889 0.392361111 0.395833333
## [115] 0.399305556 0.402777778 0.406250000 0.409722222 0.413194444 0.416666667
## [121] 0.420138889 0.423611111 0.427083333 0.430555556 0.434027778 0.437500000
## [127] 0.440972222 0.444444444 0.447916667 0.451388889 0.454861111 0.458333333
## [133] 0.461805556 0.465277778 0.468750000 0.472222222 0.475694444 0.479166667
## [139] 0.482638889 0.486111111 0.489583333 0.493055556 0.496527778 0.500000000
```

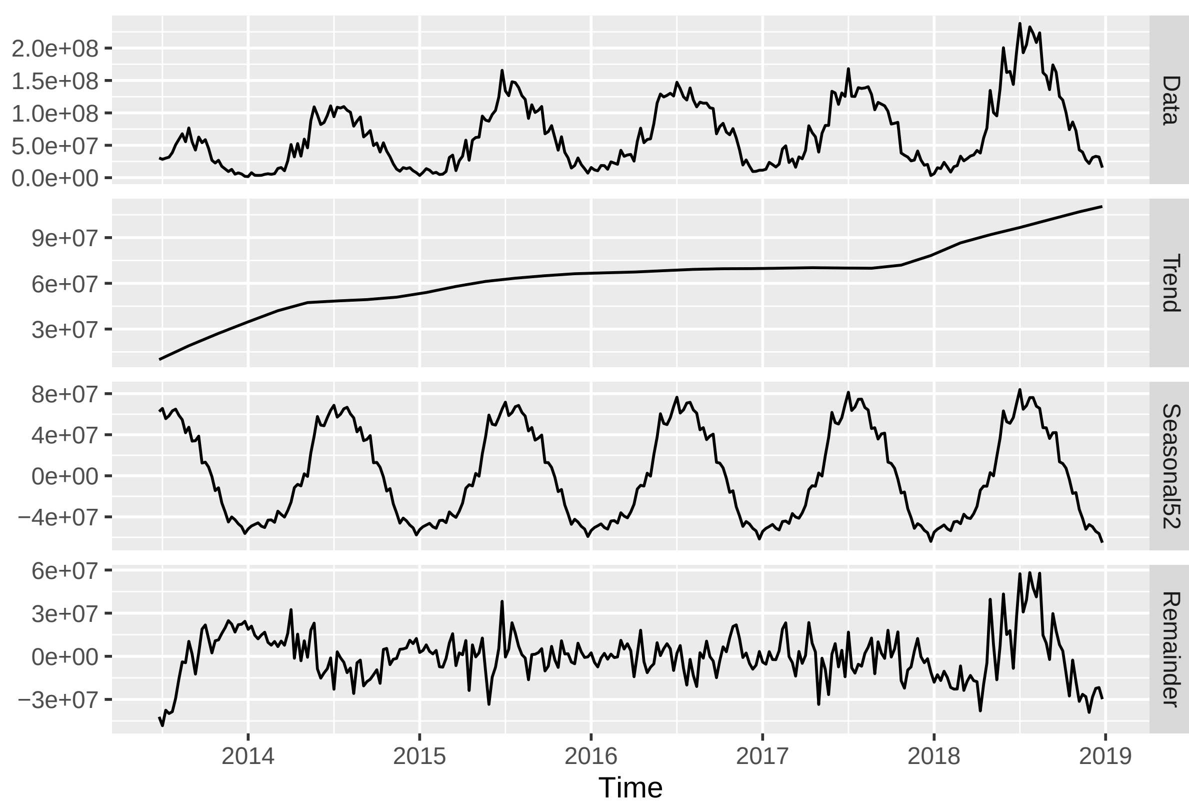
```
Max_freq <- temp$freq[which.max(temp$spec)]
1/Max_freq
```

```
## [1] 48
```

The highest two periodogram fall on frequency 0.017361111 and 0.020833333, which is 57.6 weeks and 48 weeks.

Plot STL to show the important seasonality in this data.

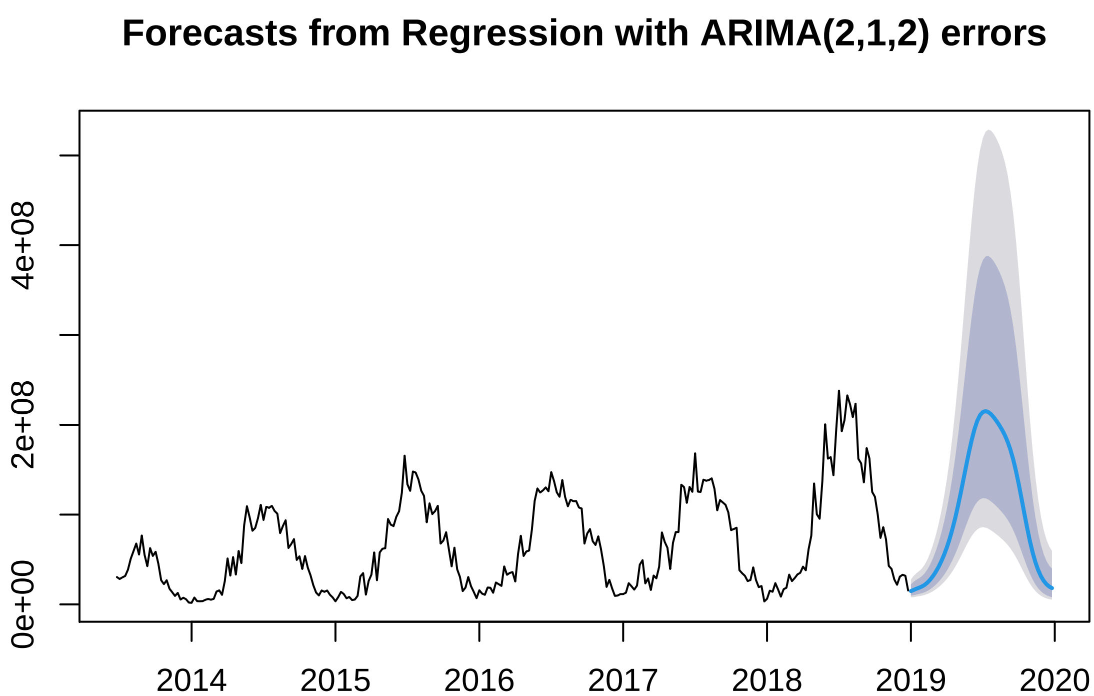
```
autoplot(mstl(train_divvy))
```



The 52 persiod is the most significant seasonality in this dataset.

Combine Fourier terms with ARIMA errors

```
arima_fourier_3 <- auto.arima(train_divvy, xreg=fourier(train_divvy,3), seasonal=FALSE,lambda = "auto")
arima_fourier_forecast_3 <- forecast(arima_fourier_3, xreg=fourier(train_divvy, 3, 52))
plot(arima_fourier_forecast_3)
```



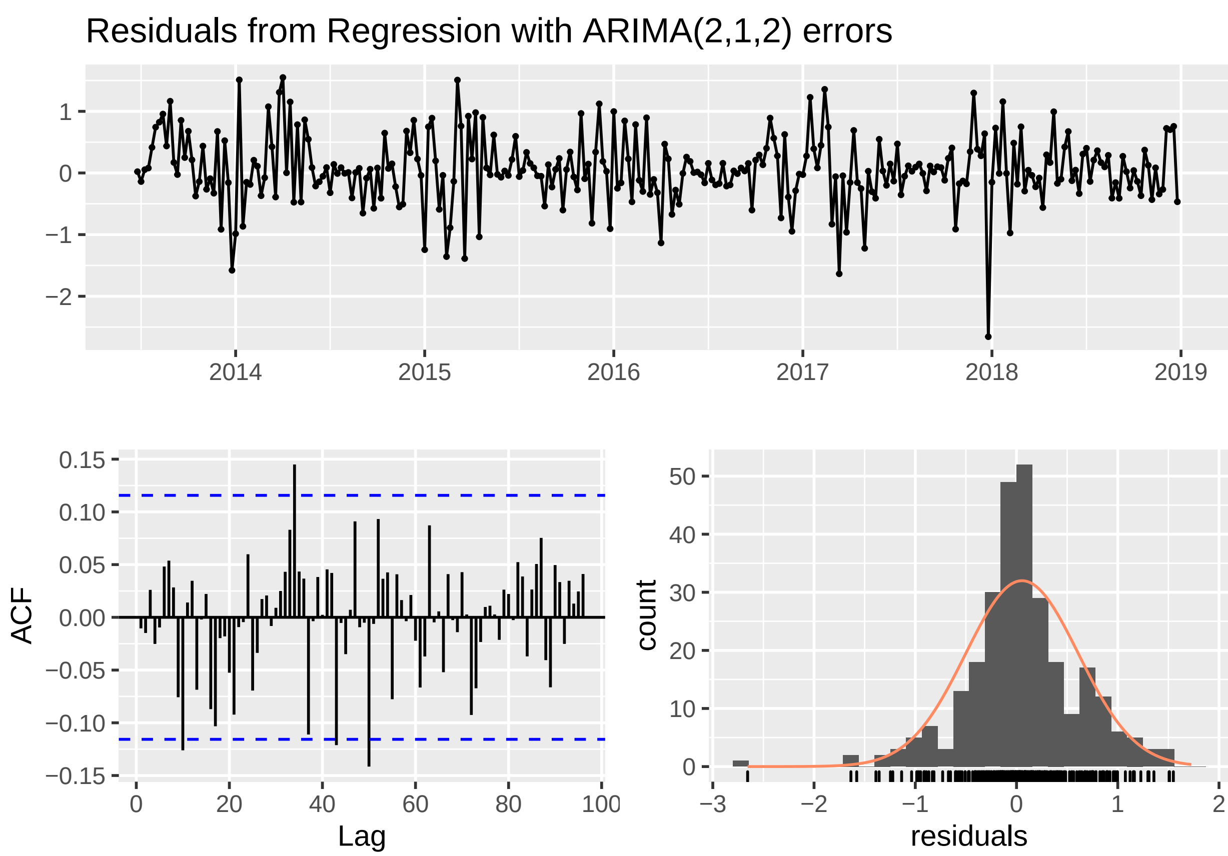
K=3

```
summary(arima_fourier_3)
```

```
## Series: train_divvy
## Regression with ARIMA(2,1,2) errors
## Box Cox transformation: lambda= 0.03488702
##
## Coefficients:
##          ar1      ar2      ma1     ma2    S1-52   C1-52    S2-52    C2-52
##       0.6597  -0.2782  -1.1942  0.3367  1.1340  2.0534  -0.2436  -0.2874
## s.e.  0.3661   0.1655   0.3649  0.3575  0.1018  0.0979   0.0635   0.0610
##         S3-52   C3-52
##       -0.0113  0.1291
## s.e.   0.0539  0.0532
##
## sigma^2 estimated as 0.325:  log likelihood=-240.48
```

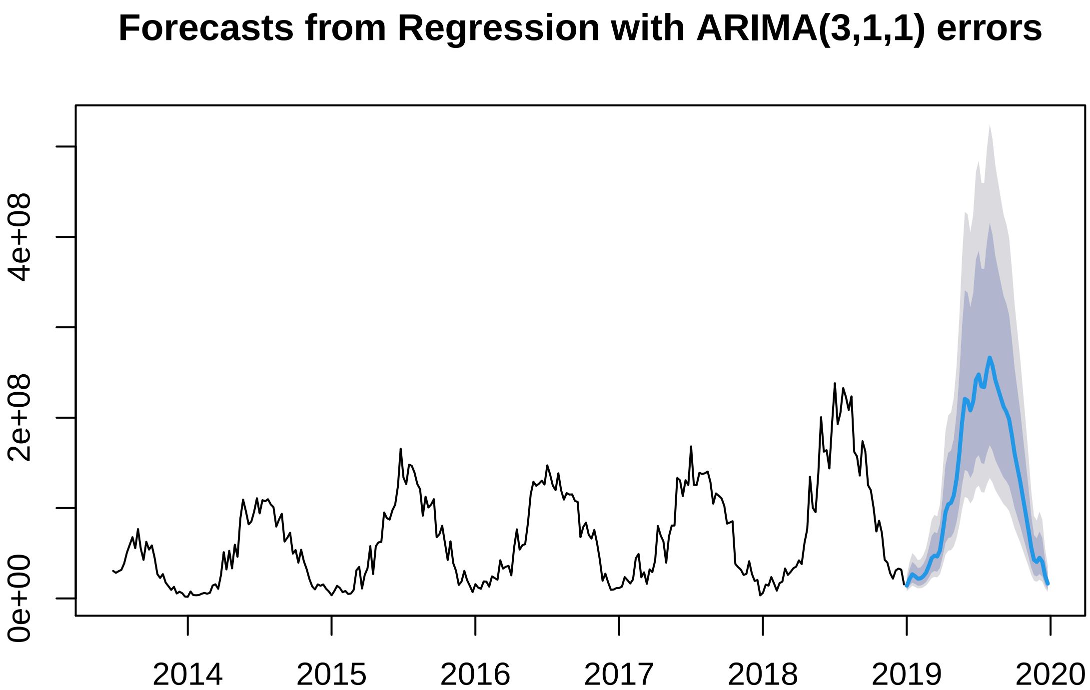
```
## AIC=502.96   AICc=503.92   BIC=543.18
##
## Training set error measures:
##                    ME     RMSE     MAE       MPE     MAPE      MASE        ACF1
## Training set 2208815 14009139 9723598 -2.305292 22.59074 0.5079432 0.003965753
```

```
checkresiduals(arima_fourier_3)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(2,1,2) errors
## Q* = 62.722, df = 47, p-value = 0.06219
##
## Model df: 10.   Total lags used: 57
```

```
arima_fourier_13 <- auto.arima(train_divvy, xreg=fourier(train_divvy,13), seasonal=FALSE,lambda = "auto
arima_fourier_forecast_13 <- forecast(arima_fourier_13, xreg=fourier(train_divvy, 13, 52))
plot(arima_fourier_forecast_13)
```



K=13

The forecast now looks more reasonable.

```
summary(arima_fourier_13)
```

```
## Series: train_divvy
## Regression with ARIMA(3,1,1) errors
## Box Cox transformation: lambda= 0.03488702
##
## Coefficients:
##          ar1     ar2     ar3      ma1   drift   S1-52   C1-52    S2-52    C2-52
##       0.3285  0.0057  0.1483  -0.9470  0.0097  1.1023  2.0470  -0.2467  -0.2912
## s.e.  0.0668  0.0654  0.0646   0.0306  0.0034  0.0874  0.0852   0.0732   0.0721
##         S3-52   C3-52    S4-52   C4-52    S5-52   C5-52    S6-52    C6-52
##       -0.0261  0.1224  -0.0162  -0.049  -0.0047  0.0166  -0.0900  -0.0254
## s.e.   0.0636  0.0632   0.0563   0.056   0.0506  0.0505   0.0465   0.0465
##        S7-52   C7-52    S8-52    C8-52   S9-52    C9-52   S10-52  C10-52
##       0.0490  0.0239  -0.1431  -0.0648  0.2051  -0.0368  -0.0480  0.0248
## s.e.  0.0436  0.0436   0.0417   0.0417  0.0406   0.0406   0.0401  0.0402
##       S11-52  C11-52   S12-52   C12-52  S13-52   C13-52
##       0.0866  0.0074  -0.0595  -0.0135  0.0617  -0.0348
## s.e.  0.0401  0.0402   0.0405   0.0406  0.0409   0.0410
##
## sigma^2 estimated as 0.2963:  log likelihood=-216.1
## AIC=496.21   AICc=504.56   BIC=613.2
##
## Training set error measures:
```

```
##                    ME     RMSE     MAE       MPE     MAPE      MASE      ACF1
## Training set 945492 13595850 9472912 -3.537875 21.39672 0.4948478 0.1057392
```

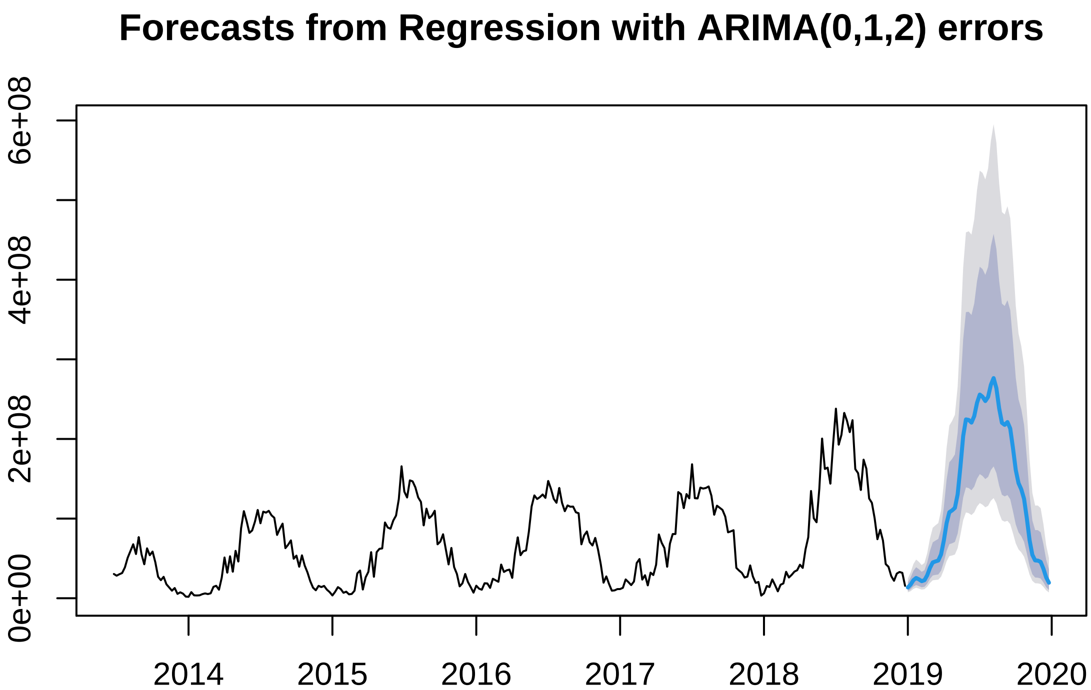
Let's check the residual of the model

```
checkresiduals(arima_fourier_13)
```


```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(3,1,1) errors
## Q* = 68.077, df = 26, p-value = 1.259e-05
##
## Model df: 31.   Total lags used: 57
```

Let's tune the parameter K=9.

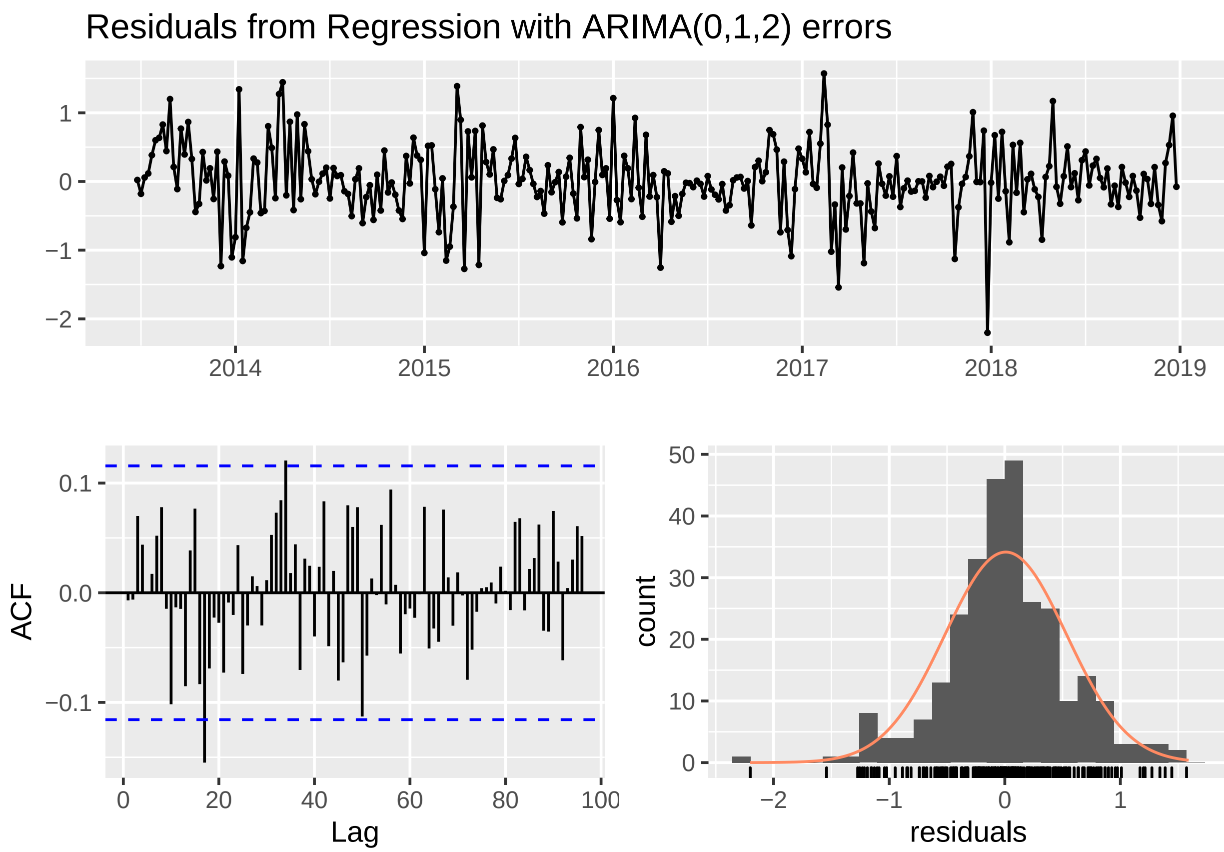
```
arima_fourier_9 <- auto.arima(train_divvy, xreg=fourier(train_divvy,9), seasonal=FALSE,lambda = "auto")
arima_fourier_forecast_9 <- forecast(arima_fourier_9, xreg=fourier(train_divvy, 9, 52))
plot(arima_fourier_forecast_9)
```




```r
summary(arima_fourier_9)
```

```
## Series: train_divvy
## Regression with ARIMA(0,1,2) errors
## Box Cox transformation: lambda= 0.03488702
##
## Coefficients:
##           ma1      ma2   drift   S1-52   C1-52    S2-52    S3-52    S3-52
##       -0.5985  -0.2310  0.0108  1.1181  2.0605  -0.2459  -0.2880  -0.0208
## s.e.   0.0599   0.0694  0.0055  0.0838  0.0812   0.0616   0.0604   0.0559
##        C3-52    S4-52    C4-52    S5-52  C5-52    S6-52    C6-52   S7-52
##       0.1287  -0.0204  -0.0454  -0.0021  0.018  -0.0939  -0.0214  0.0501
## s.e.  0.0555   0.0537   0.0533   0.0521  0.052   0.0510   0.0509  0.0498
##        C7-52    S8-52    C8-52   S9-52    C9-52
##       0.0226  -0.1460  -0.0602  0.2042  -0.0413
## s.e.  0.0499   0.0488   0.0488  0.0476   0.0476
##
## sigma^2 estimated as 0.3002:  log likelihood=-223.36
## AIC=490.73   AICc=494.57   BIC=571.16
##
## Training set error measures:
##                    ME     RMSE     MAE       MPE     MAPE     MASE       ACF1
## Training set 640186.7 13605937 9468819 -4.139442 21.95883 0.494634 0.06714366
```

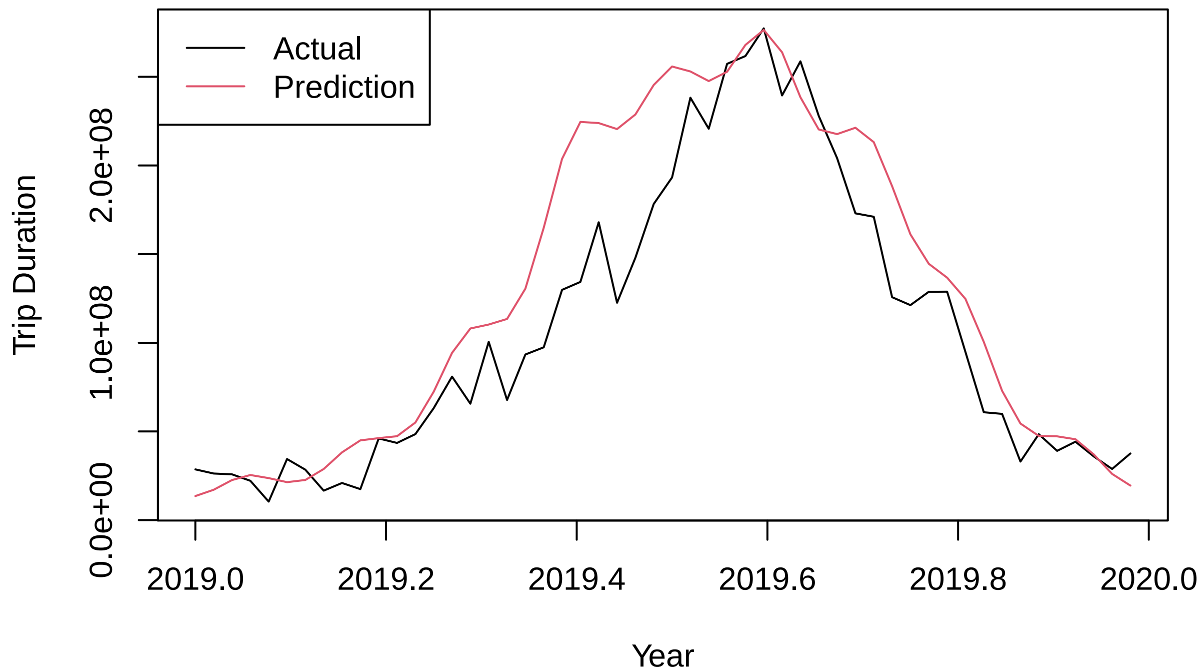
```
checkresiduals(arima_fourier_9)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(0,1,2) errors
## Q* = 64.7, df = 36, p-value = 0.002325
##
## Model df: 21.   Total lags used: 57
```

We found that when K=9, the AICc is the smallest. (AICc=494.57). However, as the K gets larger, the auto-correlation in the residual becomes more significant. When K=9, the Ljun Box test indicates that there is auto-correlation in its residual. We should try some other parameter.

```
plot(test_divvy,col=1,xlab="Year", ylab="Trip Duration")
lines(arima_fourier_forecast_9$mean,col=2)
legend('topleft',legend =c('Actual','Prediction'),col=1:2,lty=1)
```

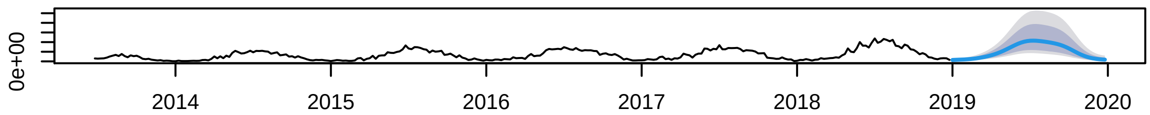



```r
MAPE(y_pred=arima_fourier_forecast_9$mean,y_true=test_divvy)
```

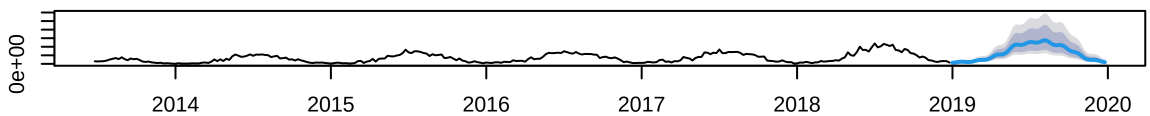
```
## [1] 0.3375399
```

```r
par(mfrow=c(3,1))
plot(arima_fourier_forecast_3, main='K = 3')
plot(arima_fourier_forecast_9, main='K = 9')
plot(arima_fourier_forecast_13, main='K = 13')
```

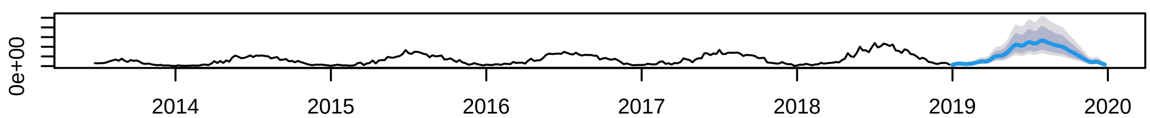
K = 3



K = 9



K = 13



Part 9: TBATS

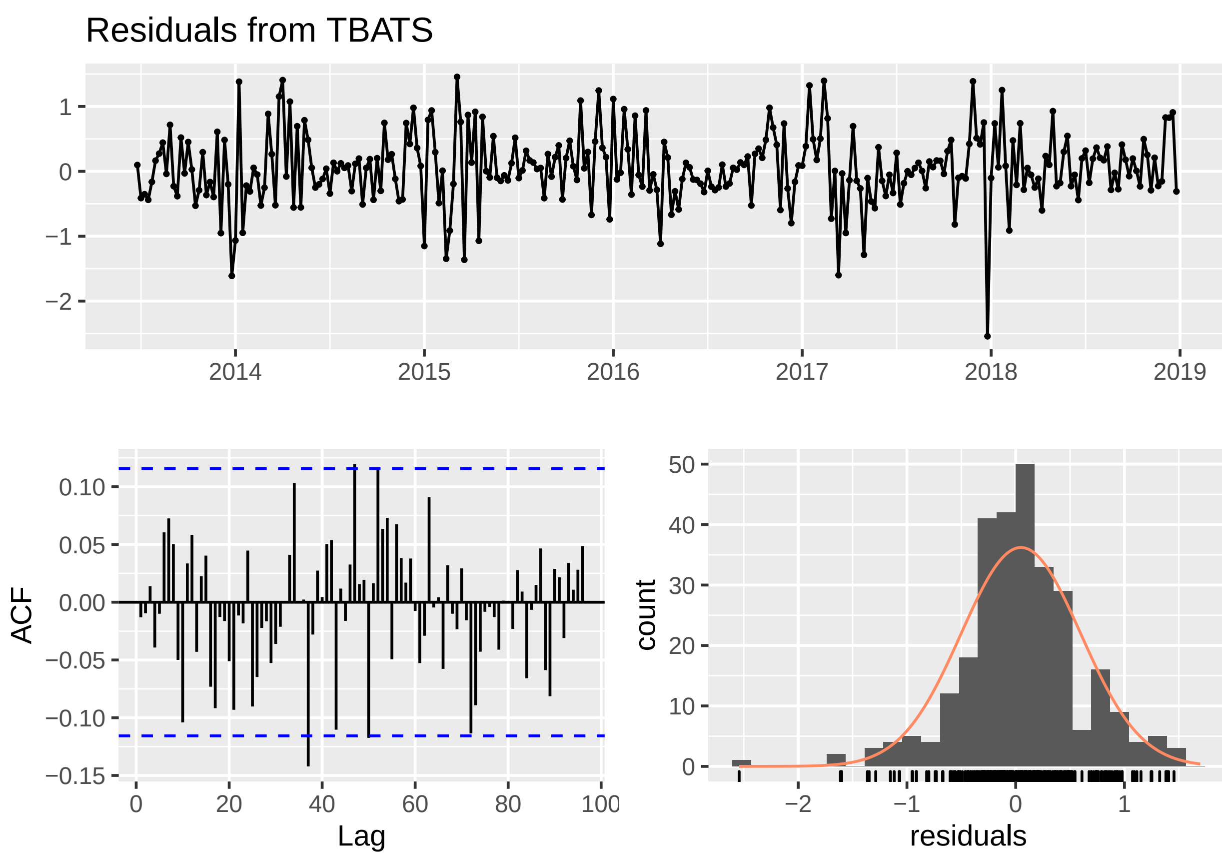
```
tbats_model <- tbats(train_divvy_boxcox,seasonal.period=52)
tbats_model
```

```
## TBATS(1, {2,1}, 0.873, {<52,3>})
##
## Call: tbats(y = train_divvy_boxcox, seasonal.periods = 52)
##
## Parameters
##   Alpha: 0.4153293
##   Beta: -0.03432027
##   Damping Parameter: 0.873104
##   Gamma-1 Values: 0.000541631
##   Gamma-2 Values: -0.0006066662
##   AR coefficients: 0.596125 -0.288484
##   MA coefficients: -0.543414
##
## Seed States:
##               [,1]
##  [1,] 20.93749400
##  [2,]  0.39032850
##  [3,]  2.41049684
```

```
##   [4,] -0.30060597
##   [5,]  0.14151109
##   [6,]  0.91360756
##   [7,] -0.15710719
##   [8,] -0.04656257
##   [9,]  0.00000000
##  [10,]  0.00000000
##  [11,]  0.00000000
##
## Sigma: 0.5509299
## AIC: 1320.083
```

Check residual

```
checkresiduals(tbats_model)
```

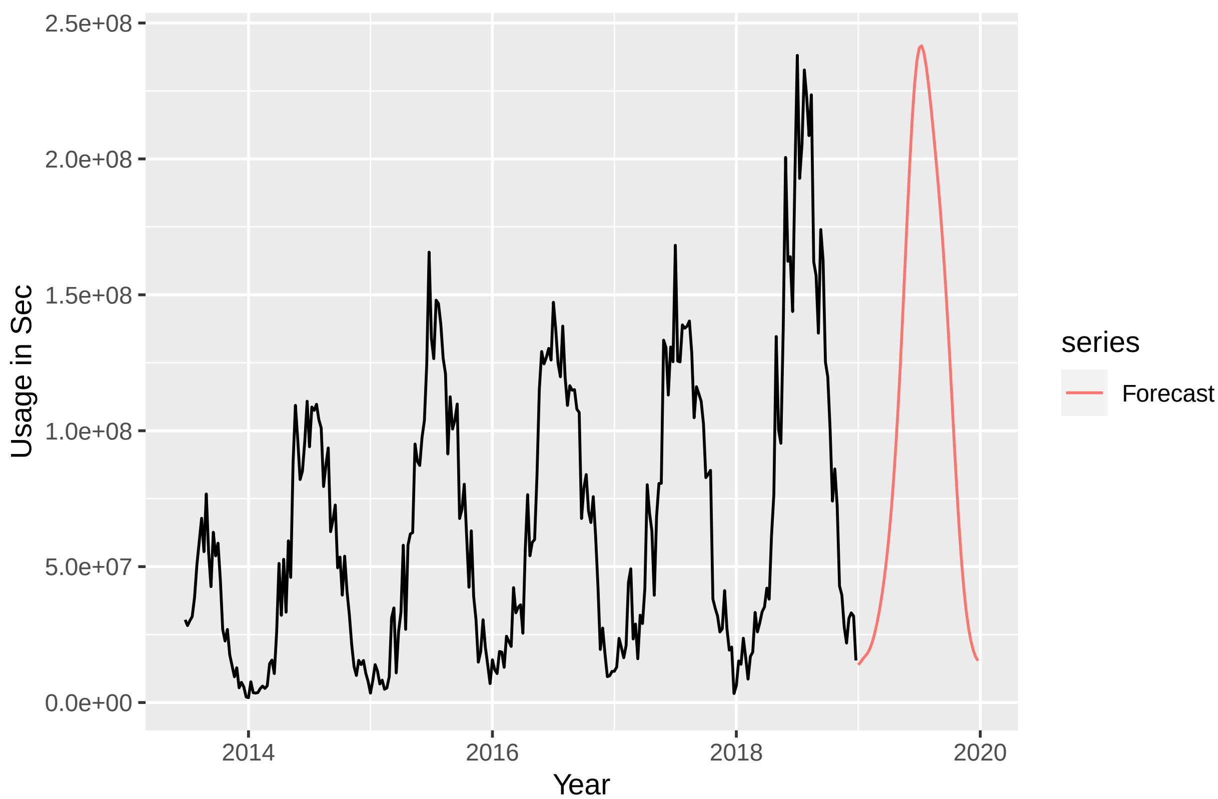



```
##
##  Ljung-Box test
##
## data:  Residuals from TBATS
## Q* = 63.714, df = 38, p-value = 0.005579
##
## Model df: 19.   Total lags used: 57
```

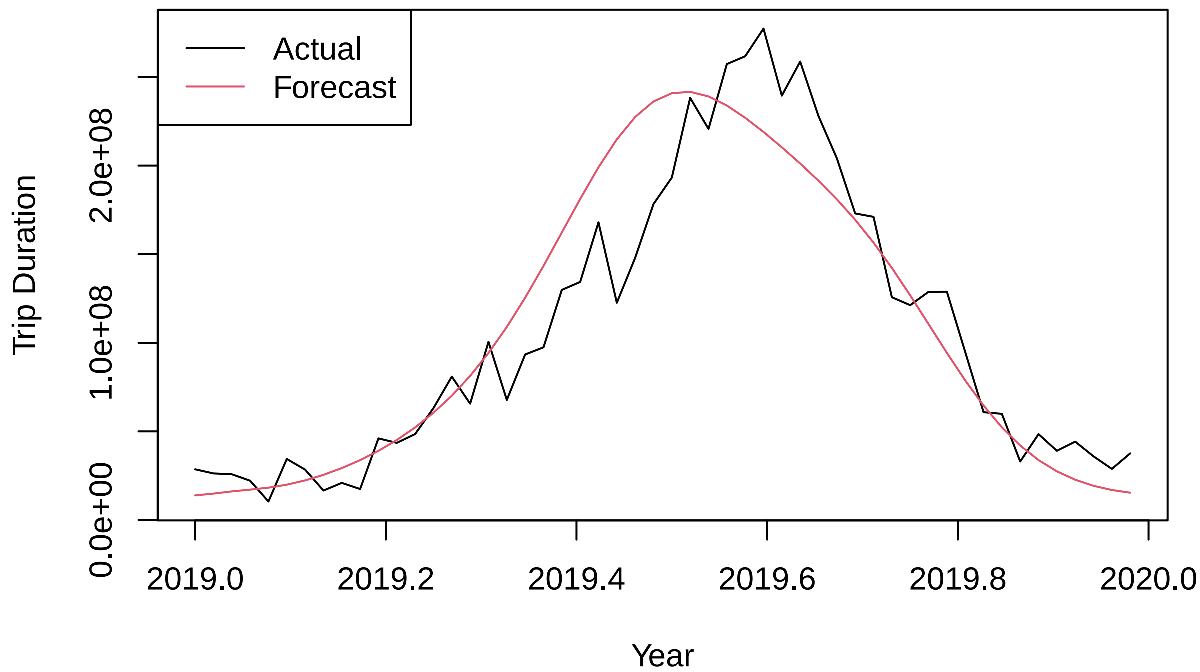
Reverse the BoxCox transformation

```
tbats_model_forecasts <- forecast(tbats_model,h=52)
forecasts_invboxcox <- InvBoxCox(tbats_model_forecasts$mean,lambda=0.03489689)
```

```
autoplot(train_divvy) +
  autolayer((forecasts_invboxcox), series="Forecast") +
  xlab("Year") + ylab("Usage in Sec")
```




```
plot(test_divvy,col=1,xlab="Year", ylab="Trip Duration")
lines(forecasts_invboxcox,col=2)
legend('topleft',legend =c('Actual','Forecast'),col=1:2,lty=1)
```




```
MAPE(y_pred=forecasts_invboxcox,y_true=test_divvy)
```

```
## [1] 0.2815186
```

Part 10: Neural Network Autoregression

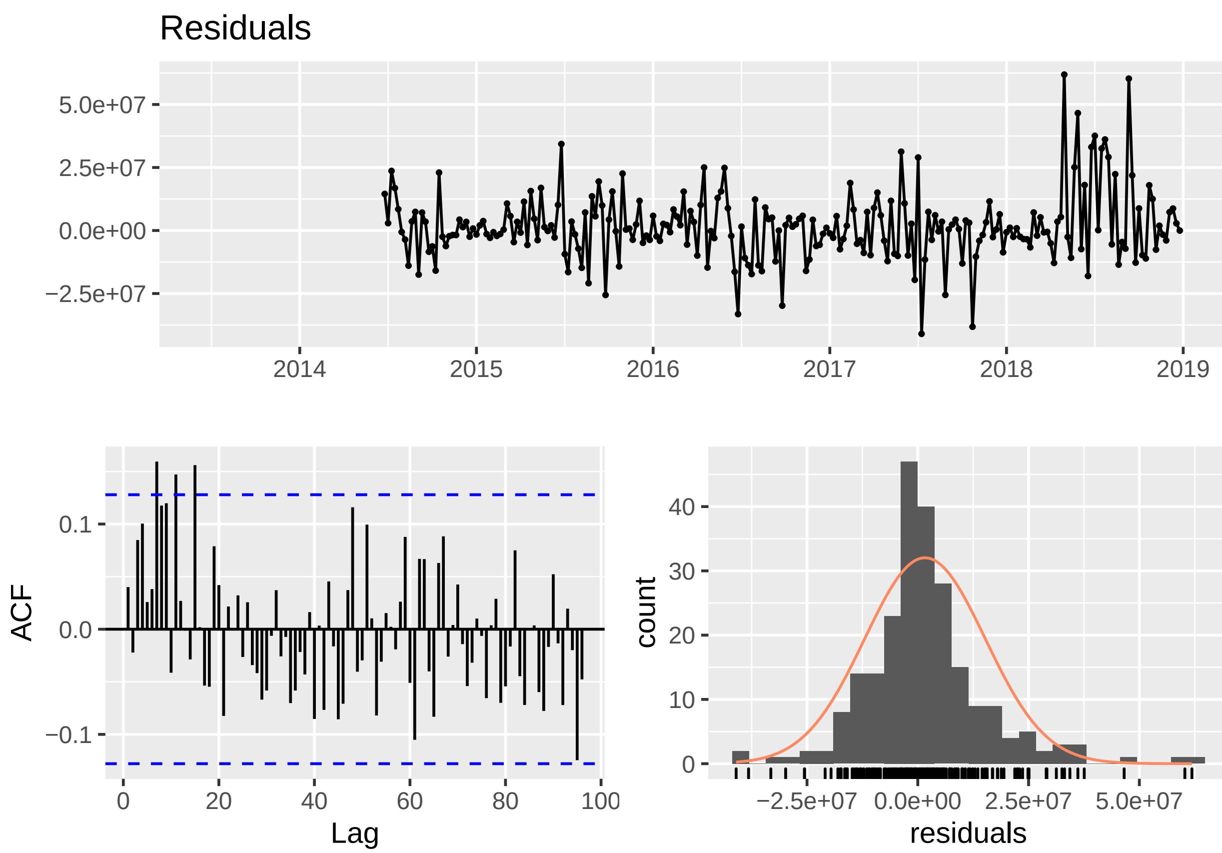
```
nnetar_model <- nnetar(train_divvy,lambda=0)
nnetar_model
```

```
## Series: train_divvy
## Model:  NNAR(7,1,4)[52]
## Call:   nnetar(y = train_divvy, lambda = 0)
##
## Average of 20 networks, each of which is
## a 8-4-1 network with 41 weights
## options were - linear output units
##
## sigma^2 estimated as 0.06357
```

Check residual

```
checkresiduals(nnetar_model$residuals)
```

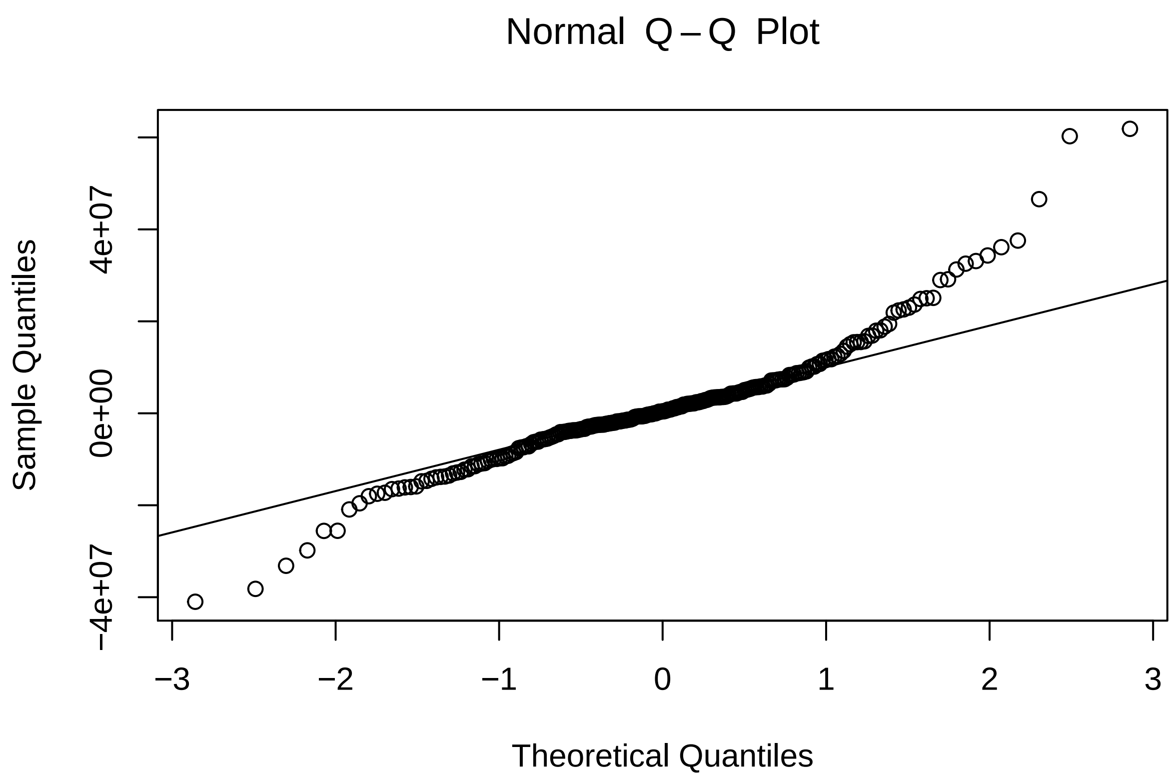
```
## Warning in modeldf.default(object): Could not find appropriate degrees of
## freedom for this model.
```




```
shapiro.test(nnetar_model$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  nnetar_model$residuals
## W = 0.93274, p-value = 6.912e-09
```

```
qqnorm(nnetar_model$residuals,main=expression(Normal~~Q-Q~~Plot))
qqline(nnetar_model$residuals)
```




```r
Box.test(nnetar_model$residuals,lag=52,type = c("Ljung-Box"),fitdf=41)
```

```
##
##  Box-Ljung test
##
## data:  nnetar_model$residuals
## X-squared = 59.576, df = 11, p-value = 1.111e-08
```