Mach 1 Training Center

```
library(readxl)
Tng_Ctr_Hour2 <- read_excel("C:/RBS/Business Forecasting/Group Project/Final Project/Tng_Ctr_Hour2.xlsx
View(Tng_Ctr_Hour2)
```

```
library(data.table)
library(ggplot2)
library(TTR)
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
```

```
library(fpp2)
```

```
##
## Attaching package: 'fpp2'

## The following objects are masked from 'package:fpp':
##
##     ausair, ausbeer, austa, austourists, debitcards, departures,
##     elecequip, euretail, guinearice, oil, sunspotarea, usmelec
```

```r
library(ggplot2)
library(stats)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:data.table':
##
##     between, first, last

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(graphics)
library(ggfortify)
```

```
## Registered S3 methods overwritten by 'ggfortify':
##   method                 from
##   autoplot.Arima         forecast
##   autoplot.acf           forecast
##   autoplot.ar            forecast
##   autoplot.bats          forecast
##   autoplot.decomposed.ts forecast
##   autoplot.ets           forecast
##   autoplot.forecast      forecast
##   autoplot.stl           forecast
##   autoplot.ts            forecast
##   fitted.ar              forecast
##   fortify.ts             forecast
##   residuals.ar           forecast
```

We are evaluating the pilot training hours logged at the Mach 1 Training Center. We will create a time series forecast of the data set.

In addition to a time series forecast, we will perform multiple regression analysis using economic metrics to see if these factors may be influencing training center hours. The economic factors selected include: "Consumer Sentiment (University of Michigan)", "NJ Unemployment Rate", "Revenue Passenger Miles for U.S. Air Carrier Domestic and International, Scheduled Passenger Flights", " Consumer Price Index for All Urban Consumers: All Items in U.S. City Average", and"Median Consumer Price Index". These economic factors were selected because they could influence the demand for flights, either directly or indirectly. This would cause increased demand for pilots and the training of pilots.

```r
summary(Tng_Ctr_Hour2)
```

```
##      Year             Quarter             Month             Device_Hrs
##  Length:81          Length:81          Length:81          Min.   : 222.8
##  Class :character   Class :character   Class :character   1st Qu.: 899.0
##  Mode  :character   Mode  :character   Mode  :character   Median :1008.0
##                                                            Mean   : 990.1
##                                                            3rd Qu.:1101.7
##                                                            Max.   :1519.9
##
##  DH_Prev_Year       DH_YoY_Change      DH_YoY_Ch_Per      Total_Inst_Hrs
##  Length:81          Length:81          Length:81          Min.   : 504.6
##  Class :character   Class :character   Class :character   1st Qu.:1937.3
##  Mode  :character   Mode  :character   Mode  :character   Median :2203.2
##                                                            Mean   :2165.7
##                                                            3rd Qu.:2446.8
##                                                            Max.   :3084.1
##
##  Total_Inst_Hrs_Prev_Year Inst_Hrs_YoY_Change Total_Inst_Hrs_YoY_Change_Per2
##  Length:81                Length:81           Length:81
##  Class :character         Class :character    Class :character
##  Mode  :character         Mode  :character    Mode  :character
##
##
##
##
##    Cons_Sent         NJURN             RPM                CPIUrban
##  Min.   : 70.30   Min.   : 2.900   Min.   :  2908236   Min.   :234.7
##  1st Qu.: 89.00   1st Qu.: 4.100   1st Qu.: 68459347   1st Qu.:241.2
##  Median : 93.80   Median : 4.900   Median : 77115921   Median :250.8
##  Mean   : 91.49   Mean   : 5.615   Mean   : 70822495   Mean   :250.2
##  3rd Qu.: 97.90   3rd Qu.: 6.200   3rd Qu.: 85326186   3rd Qu.:257.4
##  Max.   :101.40   Max.   :16.600   Max.   :101794185   Max.   :274.1
##                                    NA's   :1
##    CPIMedian
##  Min.   :0.9755
##  1st Qu.:2.1551
##  Median :2.5922
##  Mean   :2.5862
##  3rd Qu.:2.9557
##  Max.   :5.5690
##
```

Create a Factor for the Dataset

```r
setDT(Tng_Ctr_Hour2)
#changing the character values into factors
Tng_Ctr_Hour2[,Quarter:=factor(Quarter)]
Tng_Ctr_Hour2[,Month:=factor(Month)]
Tng_Ctr_Hour2[,Year:=factor(Year)]
```

Create a subset for the tested data

```
FD = select(Tng_Ctr_Hour2, Year, Quarter, Month, Device_Hrs, Total_Inst_Hrs, Cons_Sent, NJURN, RPM, CPIU
```

```
FD
```

```
##        Year Quarter Month Device_Hrs Total_Inst_Hrs Cons_Sent NJURN      RPM
##  1: 2015-01      Q1   Jan     960.42        1700.67      98.1   6.8 65975447
##  2: 2015-02      Q1   Feb     944.08        1614.00      95.4   6.7 59784666
##  3: 2015-03      Q1   Mar    1429.12        2532.90      93.0   6.3 75751609
##  4: 2015-04      Q2   Apr    1097.00        2152.25      95.9   5.8 73090871
##  5: 2015-05      Q2   May     915.85        1695.43      90.7   6.0 78002935
##  6: 2015-06      Q2   Jun     783.45        1675.91      96.1   5.9 82695216
##  7: 2015-07      Q3   Jul    1034.52        2095.00      93.1   6.2 88263768
##  8: 2015-08      Q3   Aug    1169.50        2459.83      91.9   5.5 85234949
##  9: 2015-09      Q3   Sep    1027.08        2219.00      87.2   5.1 72483190
## 10: 2015-10      Q4   Oct    1262.32        2765.47      90.0   4.9 76094668
## 11: 2015-11      Q4   Nov     999.25        2239.33      91.3   4.8 70030247
## 12: 2015-12      Q4   Dec     929.42        2054.59      92.6   4.6 74827186
## 13: 2016-01      Q1   Jan     796.42        1935.51      92.0   5.3 69265865
## 14: 2016-02      Q1   Feb     874.55        2017.40      91.7   5.4 64559135
## 15: 2016-03      Q1   Mar    1091.55        2235.33      91.0   5.3 78684812
## 16: 2016-04      Q2   Apr    1141.84        2409.30      89.0   4.9 75127770
## 17: 2016-05      Q2   May     871.36        1937.34      94.7   4.8 80784690
## 18: 2016-06      Q2   Jun    1181.21        2606.56      93.5   5.0 86518457
## 19: 2016-07      Q3   Jul     757.59        1791.01      90.0   5.4 90628483
## 20: 2016-08      Q3   Aug     972.73        2216.60      89.8   5.1 86136355
## 21: 2016-09      Q3   Sep     807.02        1934.39      91.2   4.9 75323792
## 22: 2016-10      Q4   Oct    1519.92        3084.09      87.2   4.6 77274227
## 23: 2016-11      Q4   Nov    1101.67        2361.81      93.8   4.3 72215277
## 24: 2016-12      Q4   Dec     801.83        1853.99      98.2   4.3 76957615
## 25: 2017-01      Q1   Jan     995.09        2446.80      98.5   5.1 71433297
## 26: 2017-02      Q1   Feb     962.00        2169.17      96.3   5.1 64261254
## 27: 2017-03      Q1   Mar    1130.24        2768.35      96.9   4.6 80838984
## 28: 2017-04      Q2   Apr    1054.71        2291.76      97.0   4.2 79494360
## 29: 2017-05      Q2   May    1044.95        2172.54      97.1   4.3 83542041
## 30: 2017-06      Q2   Jun    1013.73        2366.74      95.0   4.5 89667877
## 31: 2017-07      Q3   Jul     693.33        1739.90      93.4   4.9 94106754
## 32: 2017-08      Q3   Aug     983.25        2304.53      96.8   4.7 90444528
## 33: 2017-09      Q3   Sep     987.64        2302.29      95.1   4.4 74645832
## 34: 2017-10      Q4   Oct    1252.69        2810.70     100.7   4.2 80629312
## 35: 2017-11      Q4   Nov     969.31        2249.47      98.5   4.2 75655182
## 36: 2017-12      Q4   Dec     806.10        1800.08      95.9   4.0 79629124
## 37: 2018-01      Q1   Jan    1060.57        2466.01      95.7   4.9 73352756
## 38: 2018-02      Q1   Feb    1200.25        2414.06      99.7   4.9 68213441
## 39: 2018-03      Q1   Mar    1262.25        2666.14     101.4   4.5 85599897
## 40: 2018-04      Q2   Apr    1184.45        2625.94      98.8   3.9 82536712
## 41: 2018-05      Q2   May    1059.92        2455.24      98.0   3.8 88019738
## 42: 2018-06      Q2   Jun     993.55        2098.89      98.2   4.2 94375329
## 43: 2018-07      Q3   Jul     908.37        1973.29      97.9   4.4 98883723
## 44: 2018-08      Q3   Aug    1096.93        2403.06      96.2   3.9 94835592
## 45: 2018-09      Q3   Sep    1121.75        2368.10     100.1   3.5 79102243
```

```
## 46: 2018-10      Q4   Oct    1412.47        2955.81      98.6   3.3  84374600
## 47: 2018-11      Q4   Nov    1010.25        2203.17      97.5   3.1  79461417
## 48: 2018-12      Q4   Dec     970.12        1991.45      98.3   3.3  82764474
## 49: 2019-01      Q1   Jan    1063.13        2542.16      91.2   4.2  76935981
## 50: 2019-02      Q1   Feb    1036.95        2441.90      93.8   4.0  70762598
## 51: 2019-03      Q1   Mar    1130.87        2456.02      98.4   3.6  90028517
## 52: 2019-04      Q2   Apr     903.97        2286.02      97.2   2.9  86209795
## 53: 2019-05      Q2   May    1284.95        2734.56     100.0   3.0  92550535
## 54: 2019-06      Q2   Jun    1265.56        2571.35      98.2   3.2  97811522
## 55: 2019-07      Q3   Jul     848.64        2075.30      98.4   3.8 101794185
## 56: 2019-08      Q3   Aug    1247.40        2767.26      89.8   3.5  98025331
## 57: 2019-09      Q3   Sep    1106.84        2441.50      93.2   3.2  83345979
## 58: 2019-10      Q4   Oct    1217.08        2626.36      95.5   3.2  87646386
## 59: 2019-11      Q4   Nov    1091.84        2377.05      96.8   3.2  80521747
## 60: 2019-12      Q4   Dec    1024.67        2085.33      99.3   3.2  89959899
## 61: 2020-01      Q1   Jan    1094.62        2523.89      99.8   4.1  81046955
## 62: 2020-02      Q1   Feb    1050.98        2137.86     101.0   3.9  73892509
## 63: 2020-03      Q1   Mar     726.19        1556.44      89.1   4.2  42770967
## 64: 2020-04      Q2   Apr     222.80         504.57      71.8  16.4   2908236
## 65: 2020-05      Q2   May     556.92        1181.00      72.3  16.6   7137356
## 66: 2020-06      Q2   Jun     899.00        1831.79      78.1  16.1  14806854
## 67: 2020-07      Q3   Jul     585.58        1427.42      72.5  13.6  22062675
## 68: 2020-08      Q3   Aug     811.74        1982.89      74.1  10.9  23569064
## 69: 2020-09      Q3   Sep    1047.41        2283.34      80.4   7.7  23073723
## 70: 2020-10      Q4   Oct    1239.26        2568.26      81.8   7.1  27870756
## 71: 2020-11      Q4   Nov     911.93        1968.93      76.9   9.6  27957384
## 72: 2020-12      Q4   Dec     569.75        1303.50      80.7   7.2  30832402
## 73: 2021-01      Q1   Jan     685.91        1685.08      79.0   8.0  27800390
## 74: 2021-02      Q1   Feb     692.88        1605.12      76.8   8.2  26323164
## 75: 2021-03      Q1   Mar     805.42        1810.00      84.9   7.8  42683735
## 76: 2021-04      Q2   Apr     904.00        2178.17      88.3   7.1  47644846
## 77: 2021-05      Q2   May     937.62        1977.58      82.9   7.0  57822304
## 78: 2021-06      Q2   Jun     954.00        2056.29      85.5   7.9  68541316
## 79: 2021-07      Q3   Jul     605.00        1457.42      81.2   7.5  78168642
## 80: 2021-08      Q3   Aug    1027.23        2175.39      70.3   6.9  71714174
## 81: 2021-09      Q3   Sep    1008.00        2173.00      72.8   6.2        NA
##        Year Quarter Month Device_Hrs Total_Inst_Hrs Cons_Sent NJURN       RPM
##     CPIUrban CPIMedian
##  1:  234.747 1.9475296
##  2:  235.342 1.9544945
##  3:  235.976 2.4333359
##  4:  236.222 2.9626505
##  5:  237.001 2.5012911
##  6:  237.657 2.9723996
##  7:  238.034 1.9220253
##  8:  238.033 1.6357906
##  9:  237.498 2.9984157
## 10:  237.733 2.2831645
## 11:  238.017 2.2654806
## 12:  237.761 1.8347197
## 13:  237.652 2.7084166
## 14:  237.336 2.5377144
## 15:  238.080 2.2220046
## 16:  238.992 3.8254779
```

```
## 17:  239.557 3.2805192
## 18:  240.222 2.1033905
## 19:  240.101 2.4901647
## 20:  240.545 2.7425709
## 21:  241.176 2.4983293
## 22:  241.741 2.0180256
## 23:  242.026 2.4588315
## 24:  242.637 2.0383851
## 25:  243.620 3.3492298
## 26:  243.872 2.6177801
## 27:  243.766 1.4650202
## 28:  244.274 2.0820924
## 29:  244.069 2.1551132
## 30:  244.218 1.8236004
## 31:  244.280 2.1967409
## 32:  245.205 2.7462102
## 33:  246.551 2.8264339
## 34:  246.657 2.3877066
## 35:  247.378 2.2123387
## 36:  247.736 2.9557472
## 37:  248.721 3.6292773
## 38:  249.300 2.0689953
## 39:  249.517 2.2785928
## 40:  250.275 2.9320091
## 41:  250.786 3.1113162
## 42:  251.152 3.0118933
## 43:  251.345 2.2764981
## 44:  251.735 2.2967113
## 45:  252.183 2.6782871
## 46:  252.899 2.3248663
## 47:  252.822 3.4834371
## 48:  252.493 2.9268951
## 49:  252.441 2.9421529
## 50:  252.969 2.8563444
## 51:  254.147 2.9894583
## 52:  255.326 3.1675885
## 53:  255.371 2.5585351
## 54:  255.423 3.9747278
## 55:  255.925 2.6178534
## 56:  256.118 2.6257566
## 57:  256.532 3.1506552
## 58:  257.387 2.7022755
## 59:  257.989 2.9606861
## 60:  258.203 1.9992224
## 61:  258.687 2.5922152
## 62:  258.824 2.7887863
## 63:  257.989 2.5767300
## 64:  256.192 1.7631315
## 65:  255.942 3.0919936
## 66:  257.282 1.7091233
## 67:  258.604 2.8424264
## 68:  259.511 2.4333628
## 69:  260.149 1.1658029
## 70:  260.462 2.7957567
```

```
## 71:  260.927 0.9755080
## 72:  261.560 1.7310060
## 73:  262.231 0.9761601
## 74:  263.161 2.8363413
## 75:  264.793 1.8331691
## 76:  266.832 2.8953414
## 77:  268.551 3.2082219
## 78:  270.981 2.9056238
## 79:  272.265 3.6798008
## 80:  273.012 4.0950754
## 81:  274.138 5.5690464
##     CPIUrban CPIMedian
```

```
summary(FD)
```

```
##       Year      Quarter      Month       Device_Hrs      Total_Inst_Hrs
##  2015-01: 1   Q1:21    Apr     : 7   Min.   : 222.8   Min.   : 504.6
##  2015-02: 1   Q2:21    Aug     : 7   1st Qu.: 899.0   1st Qu.:1937.3
##  2015-03: 1   Q3:21    Feb     : 7   Median :1008.0   Median :2203.2
##  2015-04: 1   Q4:18    Jan     : 7   Mean   : 990.1   Mean   :2165.7
##  2015-05: 1            Jul     : 7   3rd Qu.:1101.7   3rd Qu.:2446.8
##  2015-06: 1            Jun     : 7   Max.   :1519.9   Max.   :3084.1
##  (Other):75            (Other):39
##    Cons_Sent          NJURN              RPM               CPIUrban
##  Min.   : 70.30   Min.   : 2.900   Min.   :  2908236   Min.   :234.7
##  1st Qu.: 89.00   1st Qu.: 4.100   1st Qu.: 68459347   1st Qu.:241.2
##  Median : 93.80   Median : 4.900   Median : 77115921   Median :250.8
##  Mean   : 91.49   Mean   : 5.615   Mean   : 70822495   Mean   :250.2
##  3rd Qu.: 97.90   3rd Qu.: 6.200   3rd Qu.: 85326186   3rd Qu.:257.4
##  Max.   :101.40   Max.   :16.600   Max.   :101794185   Max.   :274.1
##                                    NA's   :1
##    CPIMedian
##  Min.   :0.9755
##  1st Qu.:2.1551
##  Median :2.5922
##  Mean   :2.5862
##  3rd Qu.:2.9557
##  Max.   :5.5690
##
```

Convert to Time Series Data

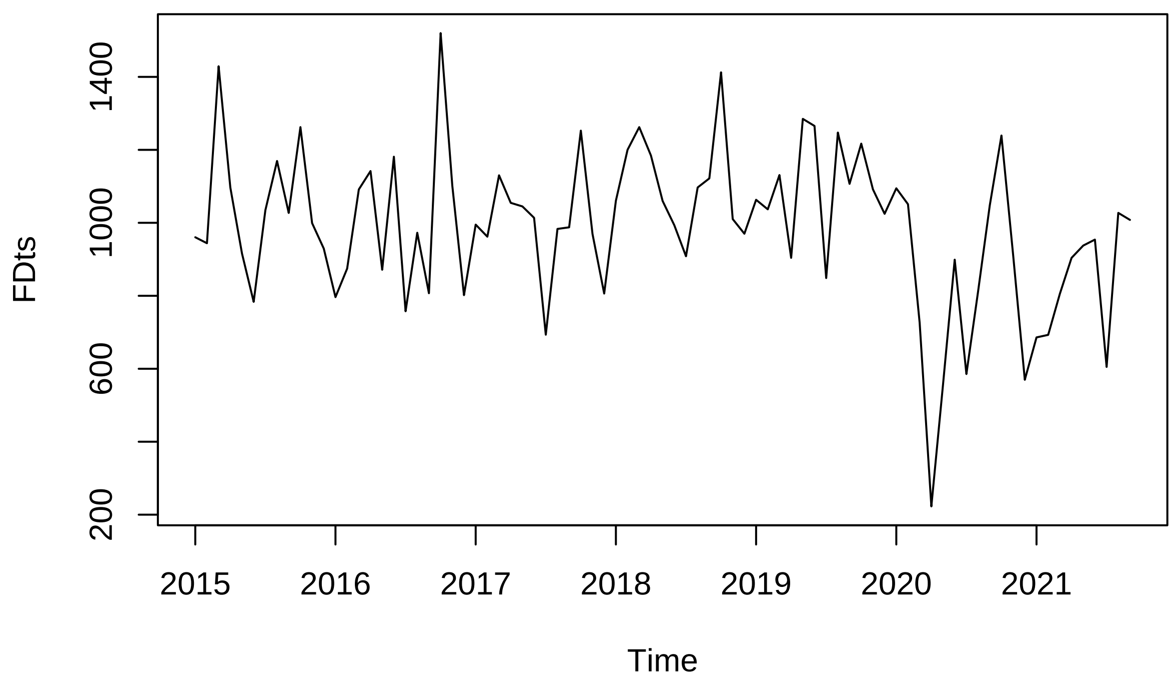
```
FDts = ts(FD$Device_Hrs, frequency = 12, start = c(2015,1))
FDts
```

```
##          Jan     Feb     Mar     Apr     May     Jun     Jul     Aug     Sep
## 2015  960.42  944.08 1429.12 1097.00  915.85  783.45 1034.52 1169.50 1027.08
## 2016  796.42  874.55 1091.55 1141.84  871.36 1181.21  757.59  972.73  807.02
## 2017  995.09  962.00 1130.24 1054.71 1044.95 1013.73  693.33  983.25  987.64
## 2018 1060.57 1200.25 1262.25 1184.45 1059.92  993.55  908.37 1096.93 1121.75
```

```
## 2019 1063.13 1036.95 1130.87  903.97 1284.95 1265.56  848.64 1247.40 1106.84
## 2020 1094.62 1050.98  726.19  222.80  556.92  899.00  585.58  811.74 1047.41
## 2021  685.91  692.88  805.42  904.00  937.62  954.00  605.00 1027.23 1008.00
##          Oct     Nov     Dec
## 2015 1262.32  999.25  929.42
## 2016 1519.92 1101.67  801.83
## 2017 1252.69  969.31  806.10
## 2018 1412.47 1010.25  970.12
## 2019 1217.08 1091.84 1024.67
## 2020 1239.26  911.93  569.75
## 2021
```

Create a plot of the time series

```
plot(FDts)
```



The plot shows a series of peaks and valleys, which suggests seasonality. Hours seemed to flow between ~700 and ~1500 consistently until 2020 when COVID struck. This caused a drop to 222 hrs in April. October is the most busy month, never falling below 1200 hrs. July and December appear to be the lightest months, only surpassing 1000 hours once each during the observed history.

Because of the pandemic, we will remove the data from 2020 onward as it was influenced by the COVID pandemic, rendering the observations afterwards as inconsistent with the forecast

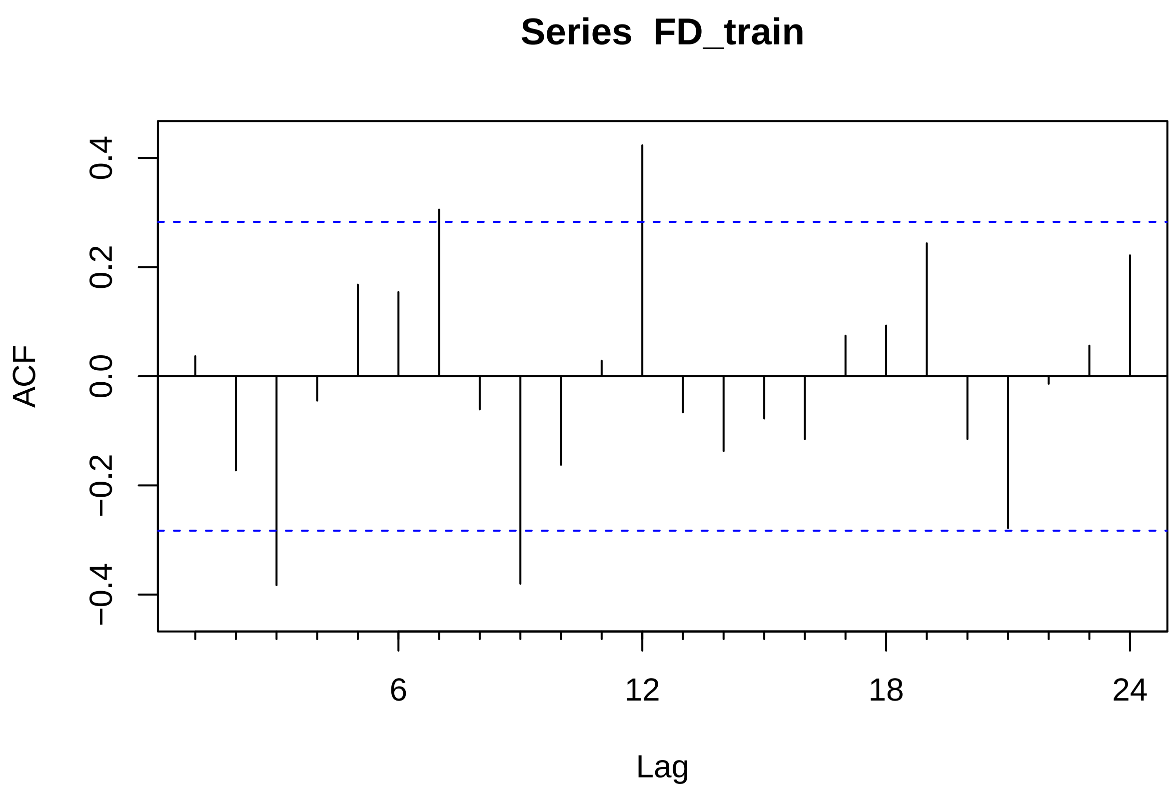
Training the Model

```
FD_train = ts(FD$Device_Hrs, frequency = 12, start = c(2015,1), end = c(2018, 12))
```

```
FD_test = ts(FD$Device_Hrs, frequency = 12, start = c(2019,1), end = c(2019, 12))
```

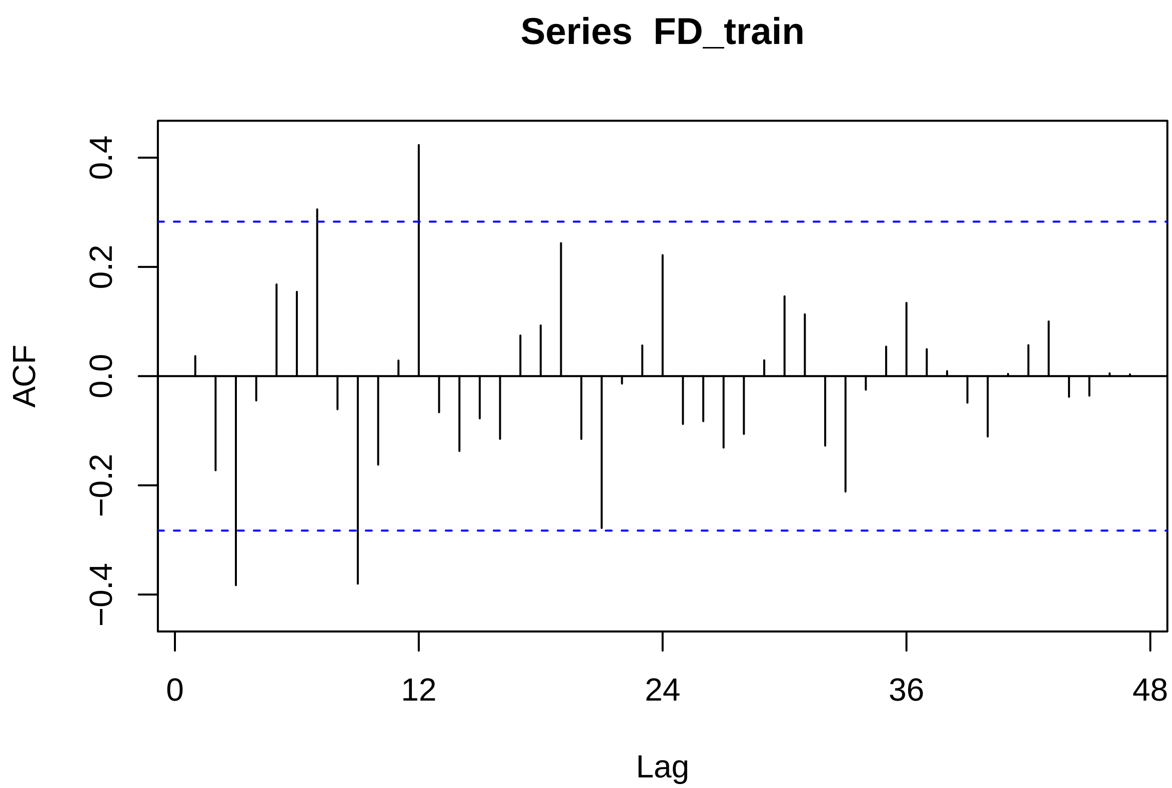
Autocorrlation of the data

```
Acf(FD_train)
```



We observe strong positive and negative autocorrlation, which furthers ours suspicions that there is seasonality

```r
Acf(FD_train, lag.max = 48)
```



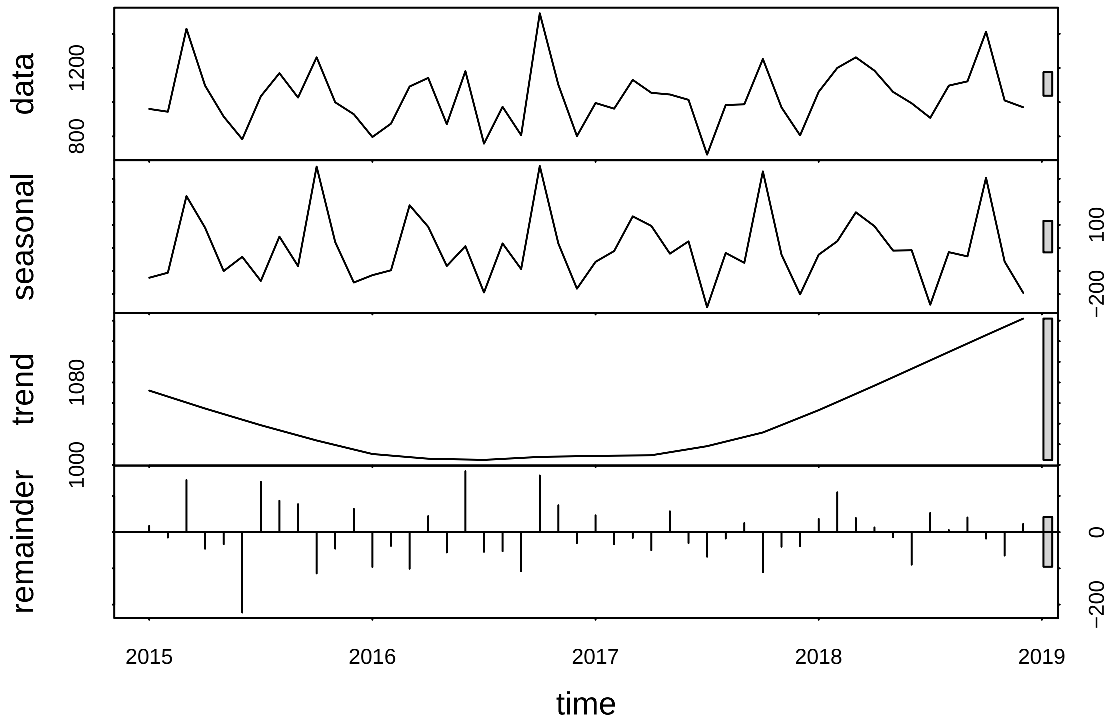
Checking For Seasonality and Trends

```r
fit= stl(FD_train, s.window = 5)
fit
```

```
##  Call:
##  stl(x = FD_train, s.window = 5)
##
## Components
##                seasonal    trend   remainder
## Jan 2015 -129.045449 1072.110   17.355668
## Feb 2015 -107.262914 1066.307  -14.963741
## Mar 2015  224.832060 1060.504  143.784413
## Apr 2015   88.044656 1054.700  -45.745057
## May 2015 -100.010890 1049.299  -33.438306
## Jun 2015  -38.452890 1043.898 -221.995102
## Jul 2015 -142.974820 1038.497  138.998032
## Aug 2015   49.132423 1033.557   86.811014
## Sep 2015  -78.637347 1028.616   77.101009
## Oct 2015  352.636565 1023.676 -113.992678
## Nov 2015   25.472369 1019.294  -45.516306
## Dec 2015 -149.739347 1014.912   64.247586
## Jan 2016 -117.920199 1010.530  -96.189386
## Feb 2016  -96.644317 1009.011  -37.816415
```

```
## Mar 2016  185.174945 1007.492 -101.116824
## Apr 2016   91.762657 1005.973   44.104317
## May 2016  -78.120110 1005.592  -56.111737
## Jun 2016    7.703243 1005.211  168.296088
## Jul 2016 -192.946834 1004.829  -54.292656
## Aug 2016   19.761423 1005.800  -52.831165
## Sep 2016  -91.470378 1006.770 -108.279615
## Oct 2016  355.701025 1007.740  156.478730
## Nov 2016   19.348695 1008.064   74.257790
## Dec 2016 -176.476142 1008.387  -30.080643
## Jan 2017  -60.060469 1008.710   46.440415
## Feb 2017  -13.295386 1008.911  -33.615230
## Mar 2017  137.033245 1009.111  -15.904423
## Apr 2017   95.565533 1009.312  -50.167271
## May 2017  -24.715705 1012.262   57.403293
## Jun 2017   28.641174 1015.213  -30.124259
## Jul 2017 -256.977073 1018.164  -67.856686
## Aug 2017  -21.654380 1022.593  -17.688292
## Sep 2017  -64.283047 1027.022   24.901463
## Oct 2017  332.006351 1031.450 -110.766847
## Nov 2017  -28.975050 1038.667  -40.382206
## Dec 2017 -201.159951 1045.884  -38.624064
## Jan 2018  -29.025710 1053.101   36.494935
## Feb 2018   29.202447 1061.037  110.010983
## Mar 2018  154.571226 1068.972   38.706409
## Apr 2018   94.558383 1076.908   12.983457
## May 2018  -11.493527 1085.086  -13.672874
## Jun 2018   -9.889678 1093.265  -89.824965
## Jul 2018 -245.915270 1101.443   52.842385
## Aug 2018  -18.182053 1109.607    5.505051
## Sep 2018  -36.462252 1117.771   40.441132
## Oct 2018  304.327121 1125.935  -17.792358
## Nov 2018  -59.028409 1133.988  -64.710038
## Dec 2018 -194.556831 1142.042   22.635174
```

```
plot(fit)
```

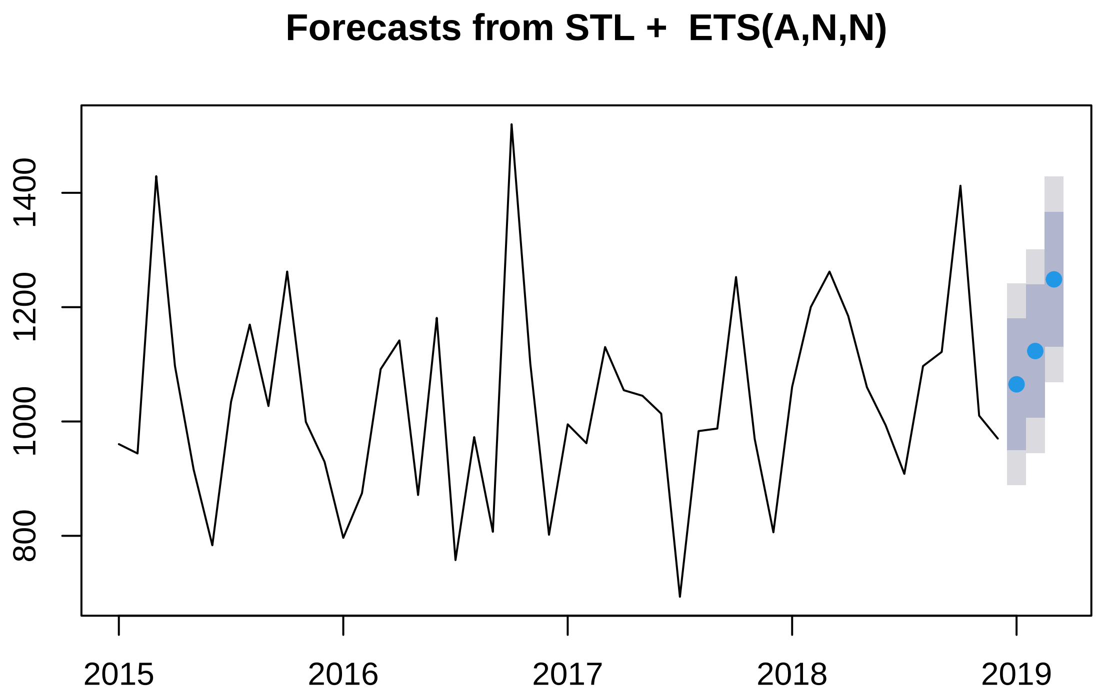


We can see some seasonality though it is not perfect. The trend is interesting as it appears that training hours were trending down, only to rebound.

Forecasting Data

Simple forecast of three periods

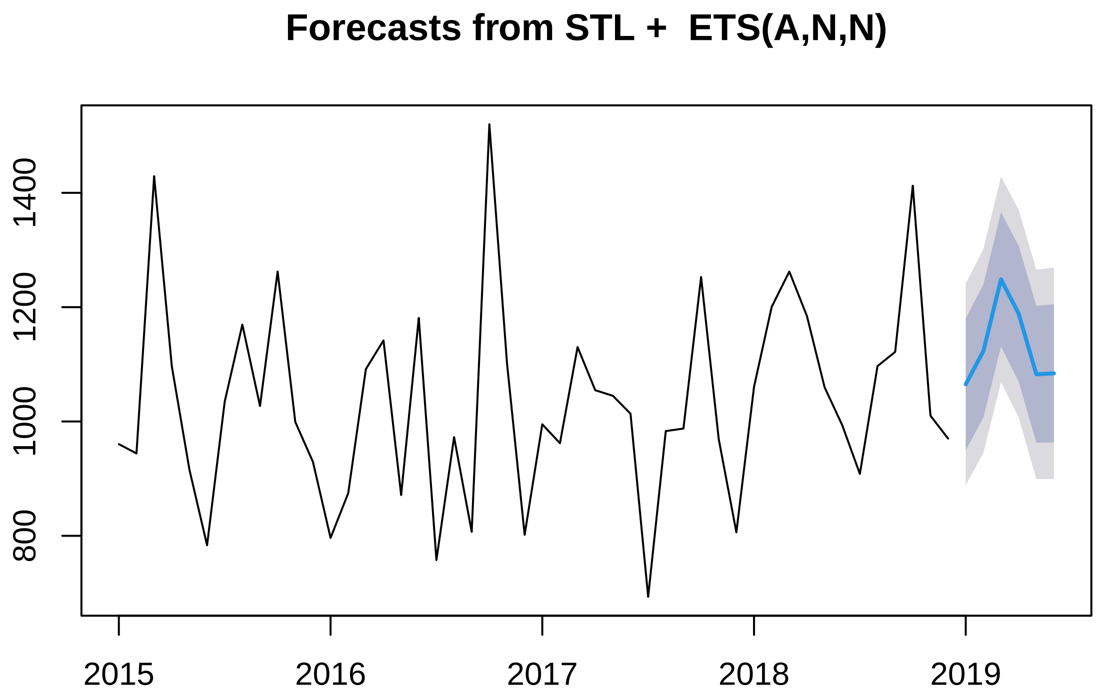
```
FitFore3 = forecast(fit, h=3)
plot(FitFore3)
```



Over the next three periods, we expect the number of hours to increase

What about 6 periods?

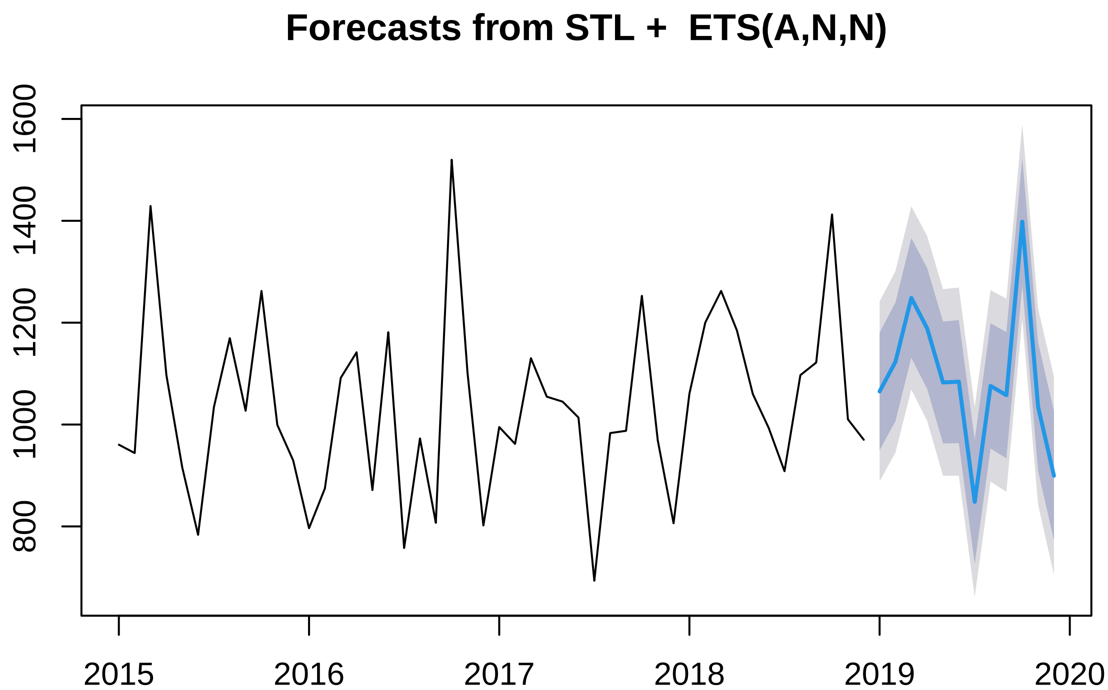
```
FitFore6 = forecast(fit, h=6)
plot(FitFore6)
```



We expect the data to increase, then decrease after three periods, consistent with our history

What about 12 Periods?!?!

```
FitFore12 = forecast(fit, h=12)
plot(FitFore12)
```



This is interesting and shows the seasonality of the data. It doesn't appear to be an exact duplication of the previous 12 months.

```
accuracy(FitFore3)
```

Accuracy Test of Forecast

```
##                    ME     RMSE      MAE         MPE     MAPE      MASE
## Training set 7.149583 88.13674 72.24431 -0.02977668 7.049975 0.5119645
##                     ACF1
## Training set -0.04022578
```

```
accuracy(FitFore6)
```

```
##                    ME     RMSE      MAE         MPE     MAPE      MASE
## Training set 7.149583 88.13674 72.24431 -0.02977668 7.049975 0.5119645
##                     ACF1
## Training set -0.04022578
```

```
accuracy(FitFore12)
```

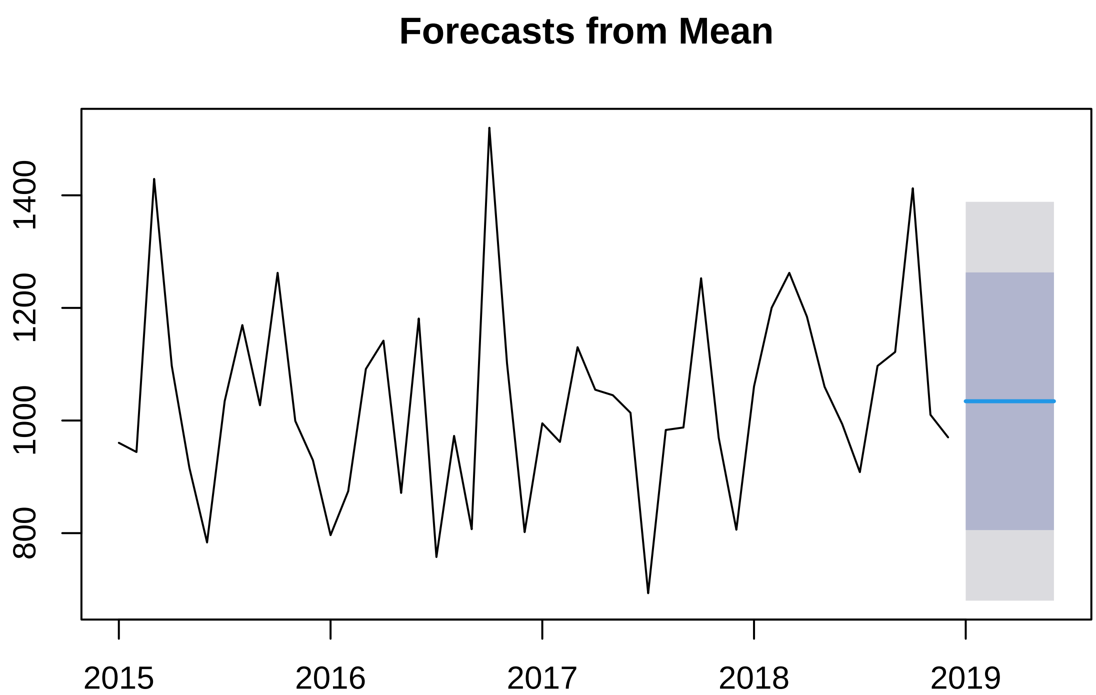
```
##                    ME     RMSE      MAE         MPE     MAPE      MASE
```

```
## Training set 7.149583 88.13674 72.24431 -0.02977668 7.049975 0.5119645
##                     ACF1
## Training set -0.04022578
```

All forecasts have the same accuracy measures. Length of the forecast does not impact the accuracy. Interesting.

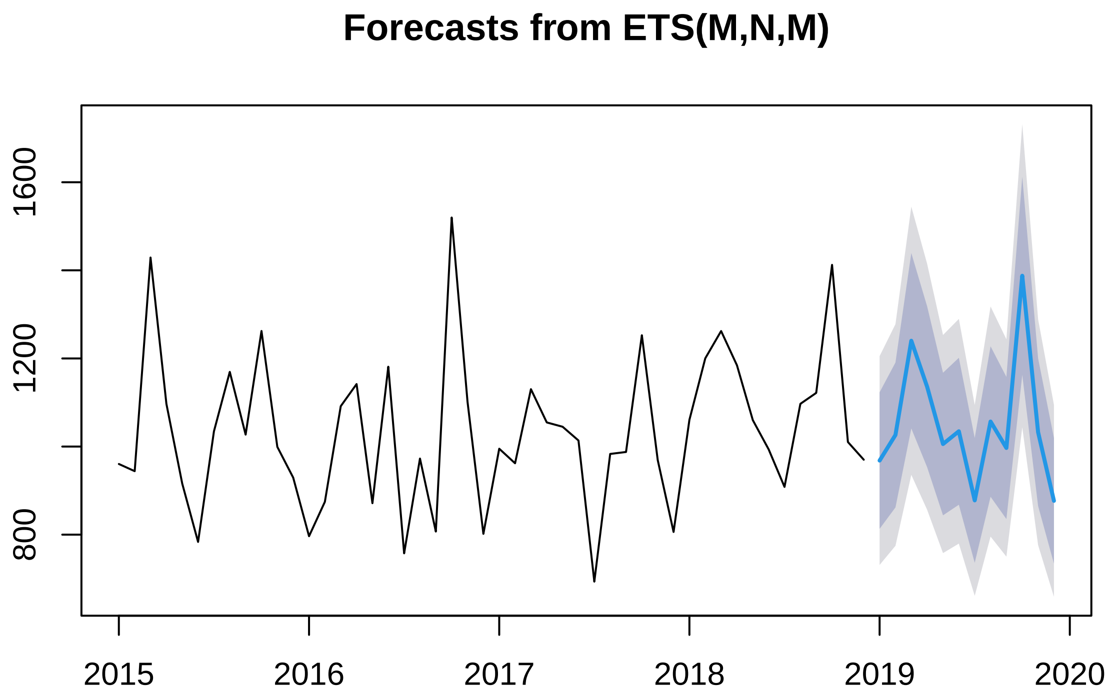
Mean Forecast Method

```
mean_FDT <- meanf(FD_train,6) # 6 is the forecasting period (6 quarters out)
plot(mean_FDT)
```

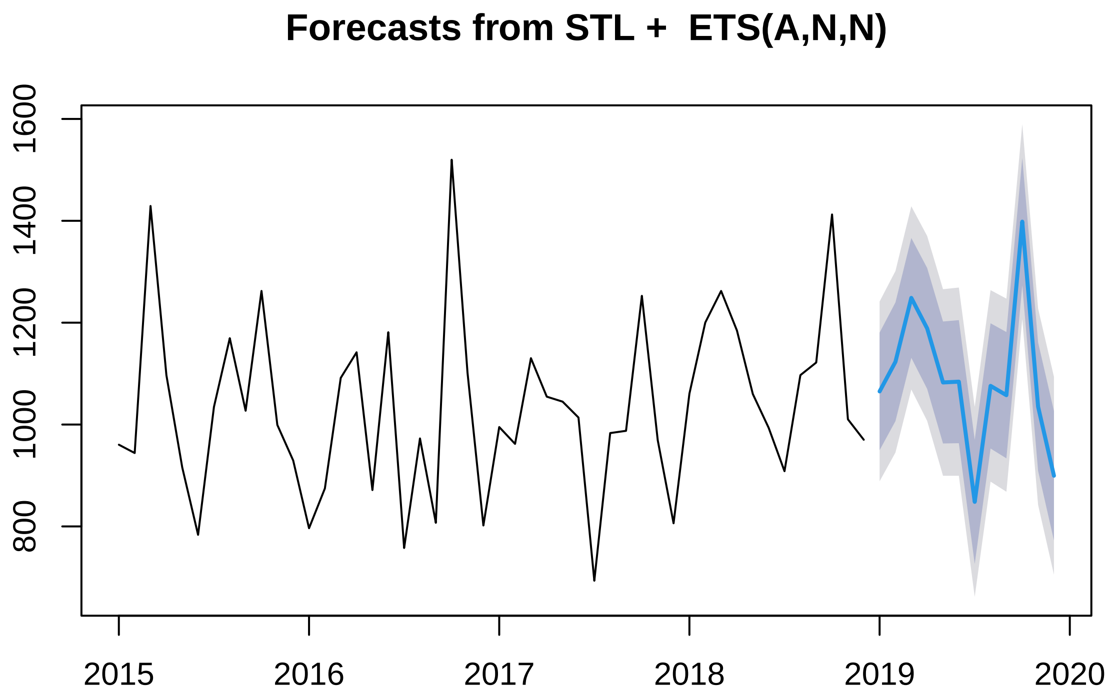


12 Month Forecast of Trained data (raw) vs. FIT data

```
Forecast_Train12 = forecast(FD_train, h=12)
plot(Forecast_Train12)
```




```
FitFore12 = forecast(fit, h=12)
plot(FitFore12)
```



Fit forecast is steeper. Is it capturing more of the seasonal swings?

```
accuracy(FitFore12)
```

Let's compare the accuracy of the forecasts

```
##                    ME     RMSE      MAE         MPE     MAPE      MASE
## Training set 7.149583 88.13674 72.24431 -0.02977668 7.049975 0.5119645
##                     ACF1
## Training set -0.04022578
```

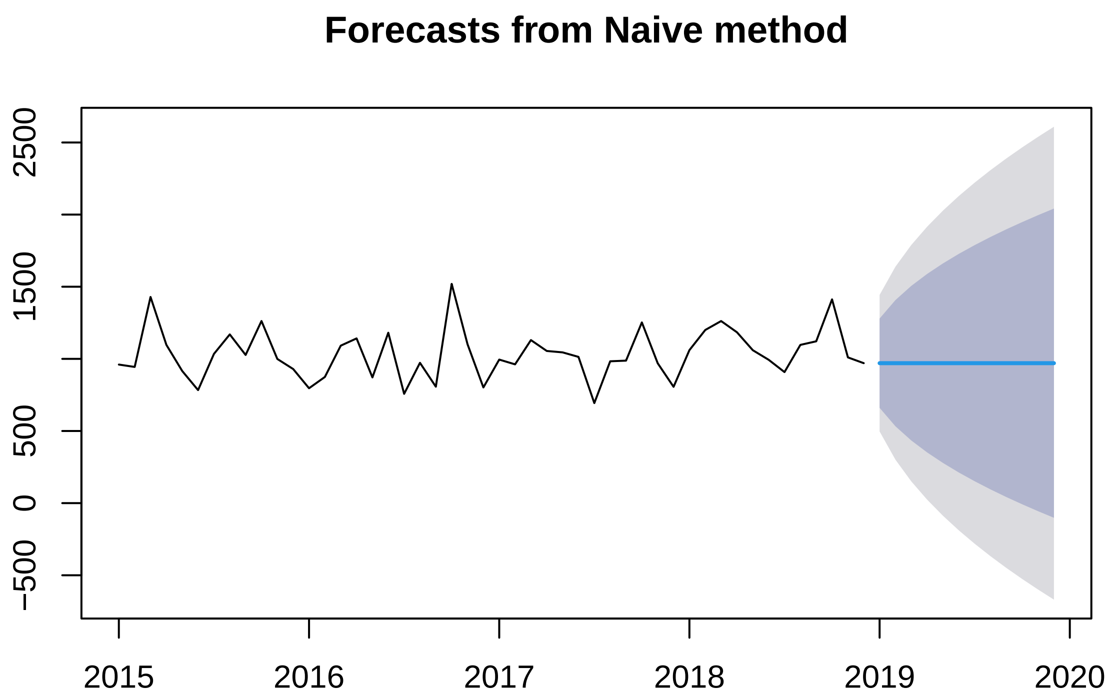
```
accuracy(Forecast_Train12)
```

```
##                    ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 8.442611 105.9033 87.19015 -0.2842957 8.633811 0.6178793
##                    ACF1
## Training set 0.02967374
```

Fit forecast appears more accurate. Has lower RMSE and MAPE.

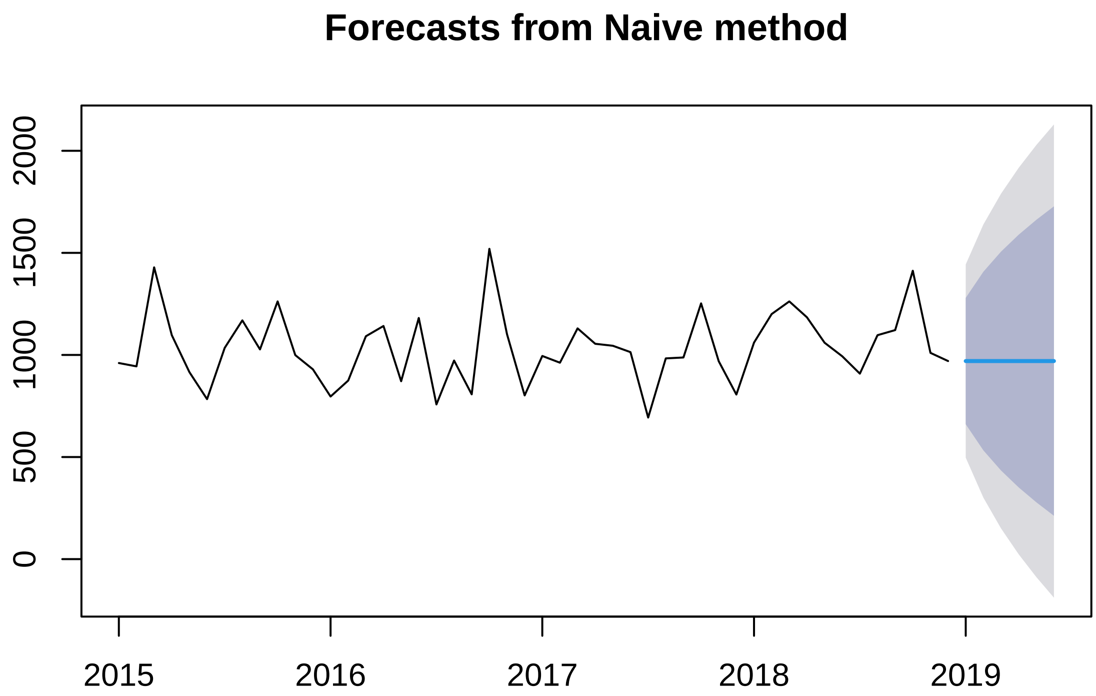
Naive Forecast

```
naive_forecast <- naive(FD_train,12)
plot(naive_forecast)
```



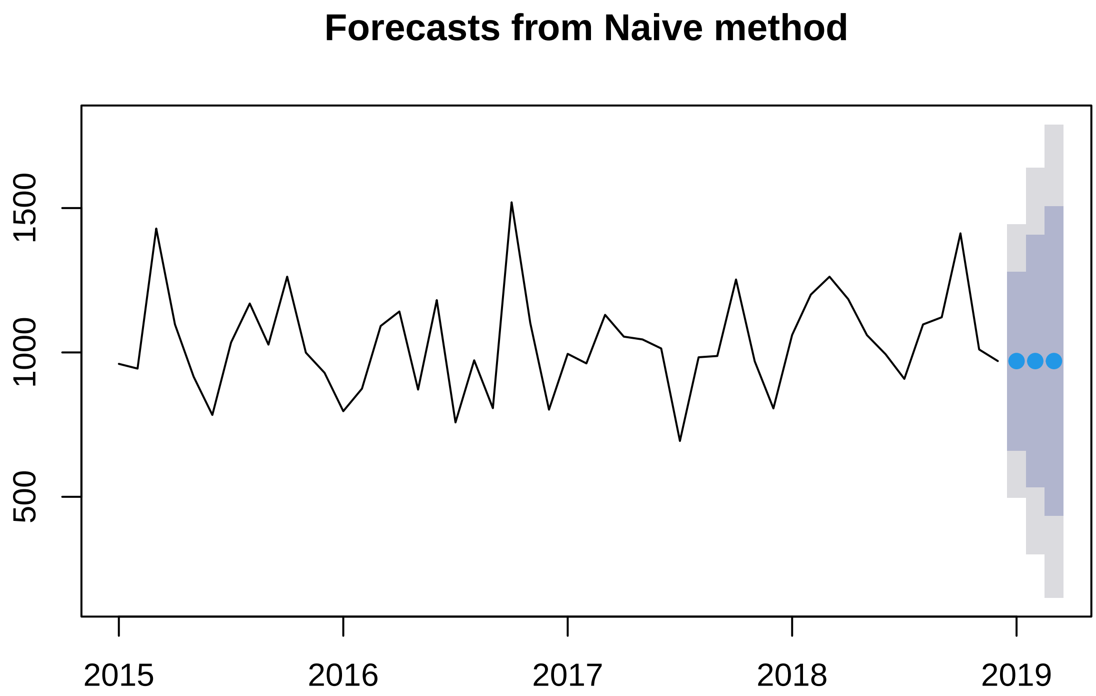
Not particularly useful because the the scope stretches into negative territory. Lets trim down the forecast to something shorter

```
naive_forecast6 <- naive(FD_train,6)
plot(naive_forecast6)
```



Still not good. 3 Months?

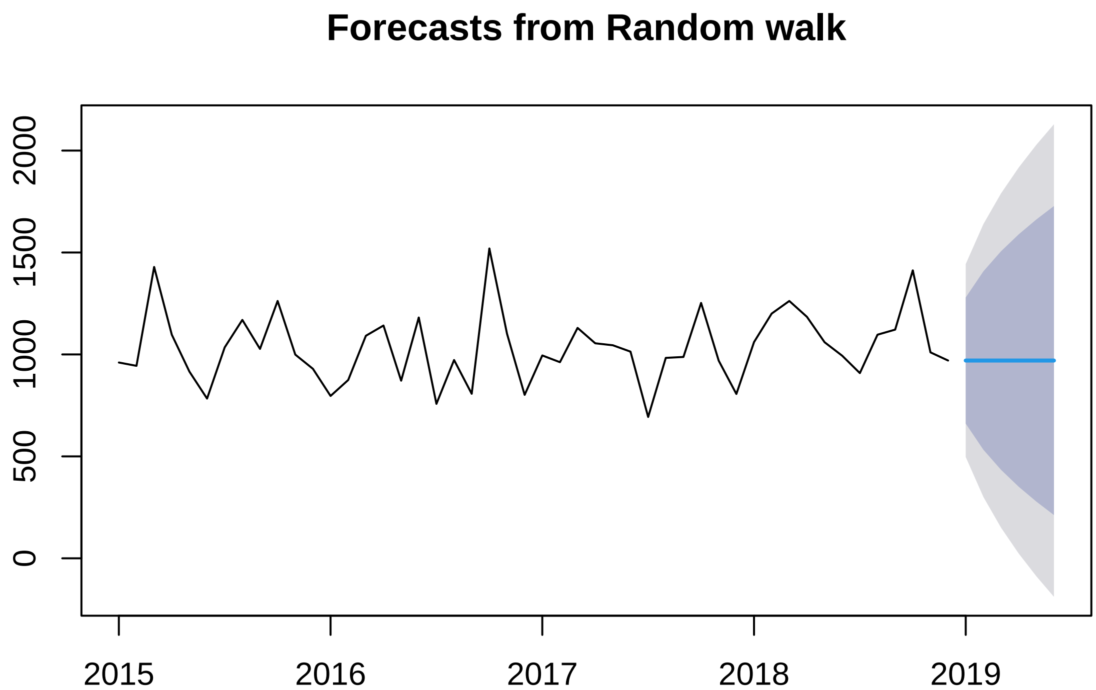
```
naive_forecast3 <- naive(FD_train,3)
plot(naive_forecast3)
```



The range of outcomes still appears too large. Naive doesn't look like the best model.

Random Walk Forecast

```
rwf_forecast <- rwf(FD_train,6)
plot(rwf_forecast)
```



This graph also shows little by way of seasonality.

Lets compare the accuracy of all the forecasts:

```
accuracy(Forecast_Train12)
```

```
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 8.442611 105.9033 87.19015 -0.2842957 8.633811 0.6178793
##                     ACF1
## Training set 0.02967374
```

```
accuracy(FitFore12)
```

```
##                     ME     RMSE      MAE         MPE     MAPE      MASE
## Training set 7.149583 88.13674 72.24431 -0.02977668 7.049975 0.5119645
##                      ACF1
## Training set -0.04022578
```

```
accuracy(rwf_forecast)
```

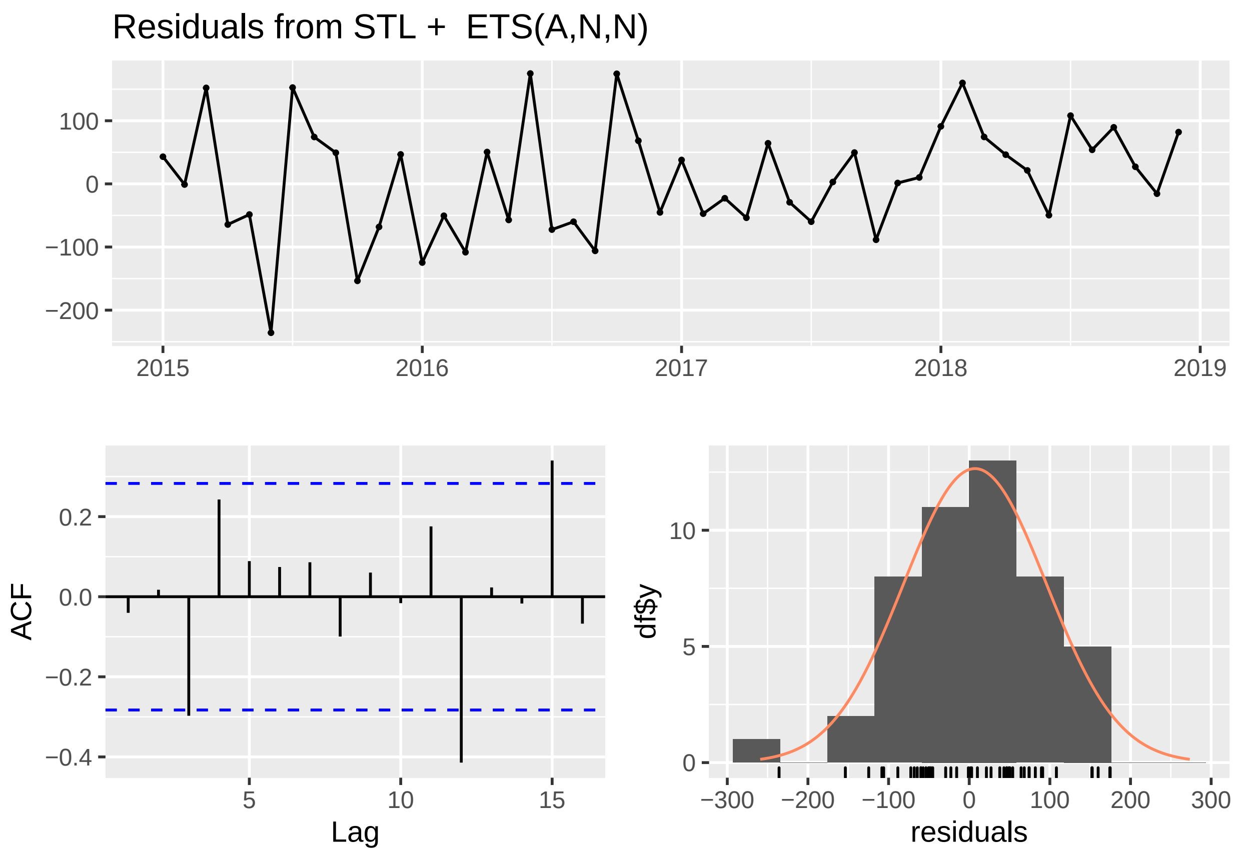
```
##                     ME     RMSE      MAE       MPE     MAPE     MASE       ACF1
## Training set 0.206383 241.3982 194.2936 -2.591405 18.87992 1.376876 -0.3923039
```

Fit forecast appears the strongest based on these metrics. It also captures the seasonality well.

Check Residuals for Fit Forecast

```
checkresiduals(FitFore12)
```

```
## Warning in checkresiduals(FitFore12): The fitted degrees of freedom is based on
## the model used for the seasonally adjusted data.
```



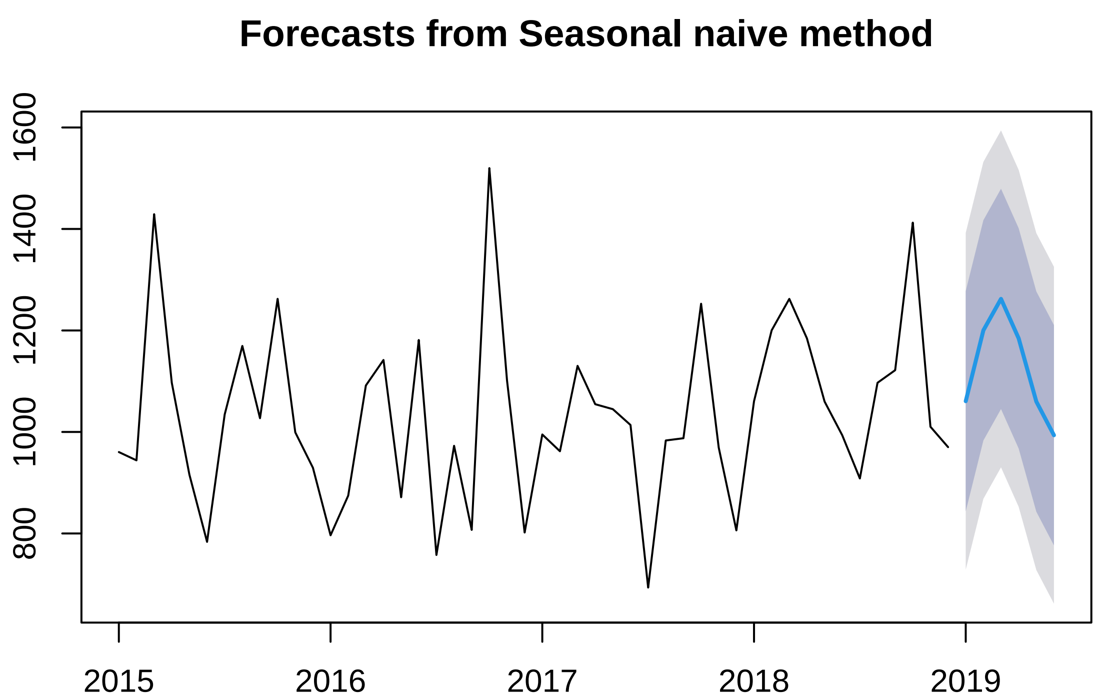

```
##
##  Ljung-Box test
##
## data:  Residuals from STL +  ETS(A,N,N)
## Q* = 10.05, df = 8, p-value = 0.2615
##
## Model df: 2.   Total lags used: 10
```

Residuals appear to be normally distributed with some slight skewing to the left. P-values are above .05 meaning

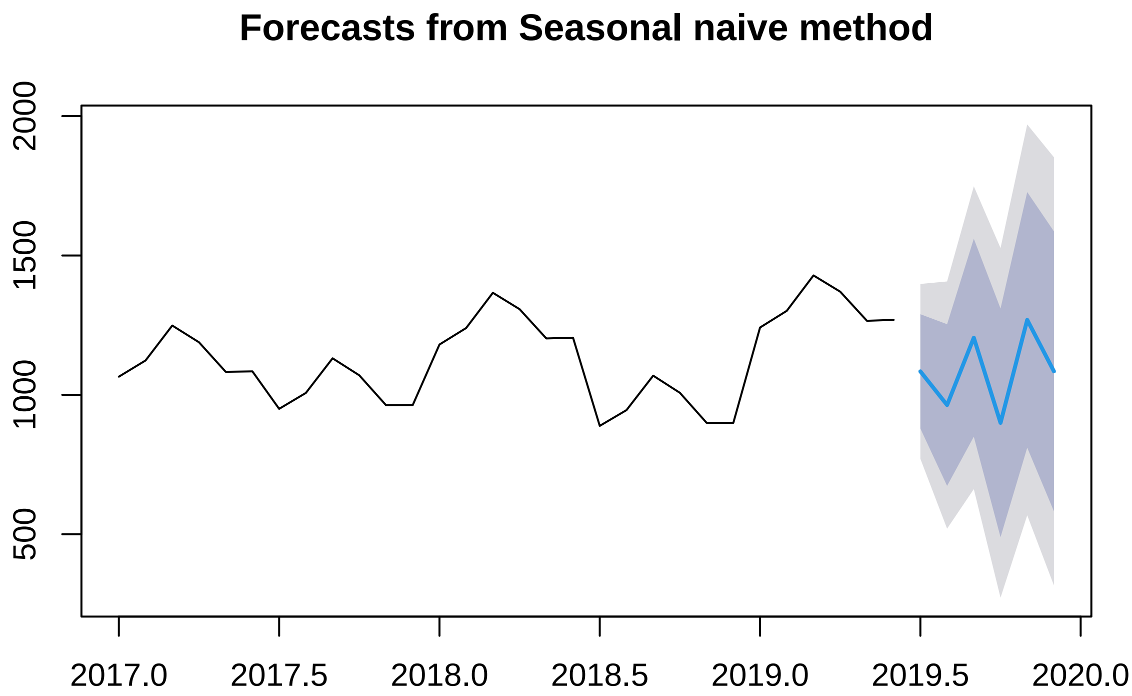
what? ^^^

Seasonal Naive

```
snaive_forecast <- snaive(FD_train, 6)
plot(snaive_forecast)
```




```
snaive_Fit <- snaive(FitFore6, 6)
plot(snaive_Fit)
```

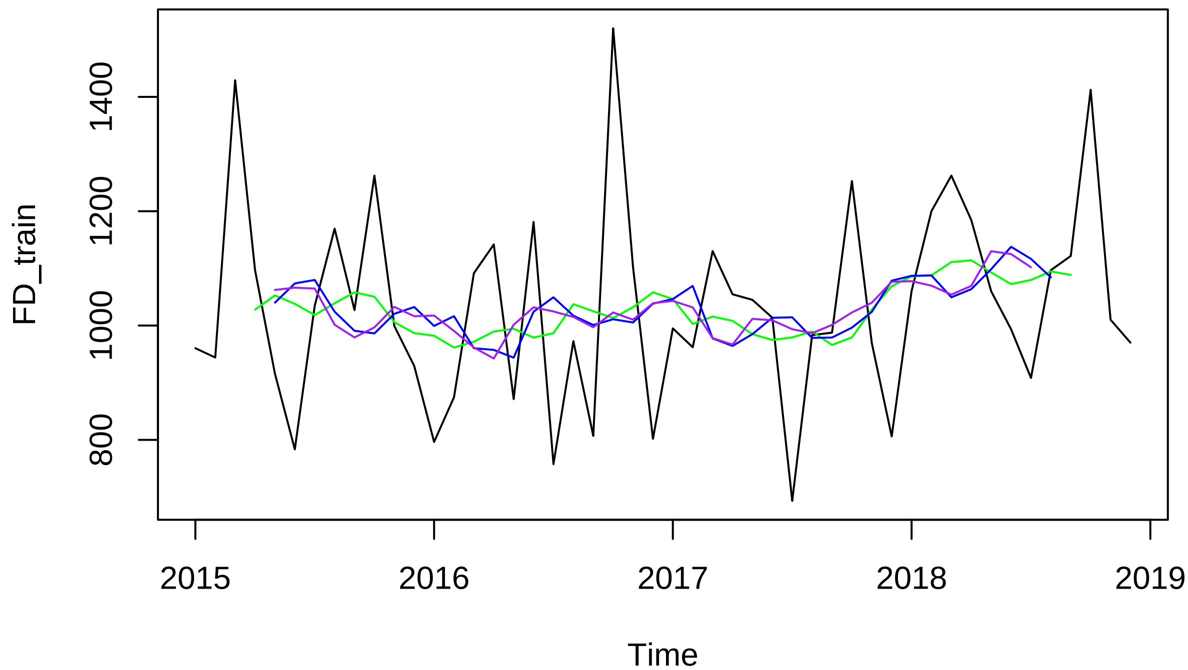


These results are interesting and very different.

```
# moving averages
MA5_forecast <- ma(FD_train ,order=6)
plot(FD_train) +
lines(MA5_forecast,col="Green")
```

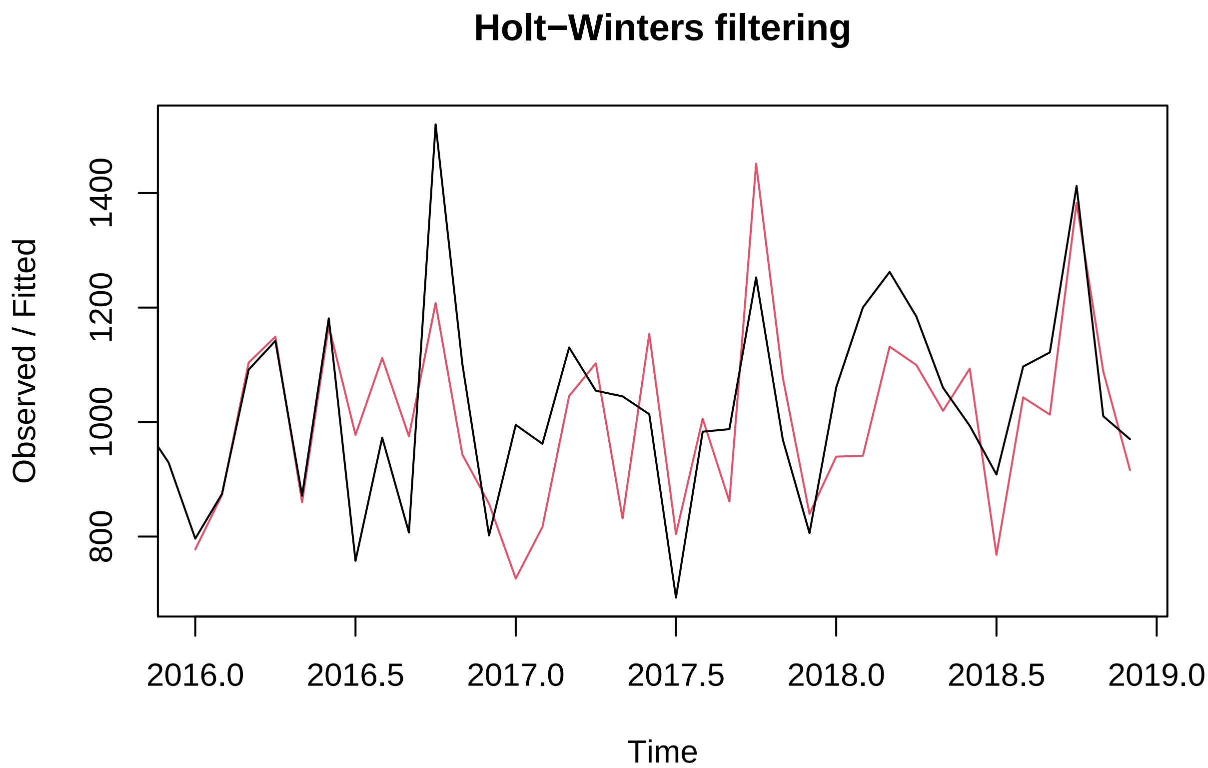
```
## integer(0)
```

```
MA9_forecast <- ma(FD_train,order=9)
lines(MA9_forecast,col="Blue")
MA50_forecast <- ma(FD_train,order=10, centre = FALSE)
lines(MA50_forecast,col="Purple")
```



Holt Winters

```
HW_FDTrain = HoltWinters(FD_train)
plot(HW_FDTrain)
```




```
HW_FDTrain
```

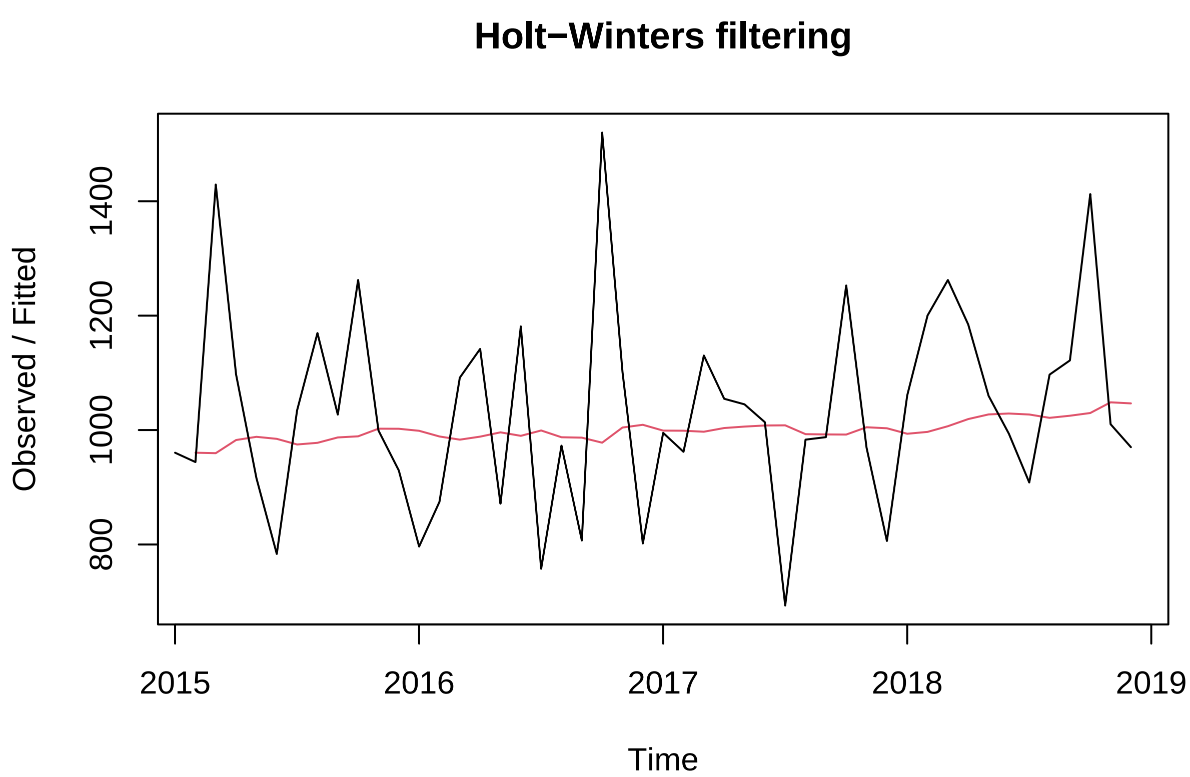
```
## Holt-Winters exponential smoothing with trend and additive seasonal component.
##
## Call:
## HoltWinters(x = FD_train)
##
## Smoothing parameters:
##  alpha: 0.01670918
##  beta : 1
##  gamma: 0.7138949
##
## Coefficients:
##             [,1]
## a   1090.033935
## b     12.085411
## s1    38.103050
## s2   134.598709
## s3   226.239016
## s4   153.356954
## s5    32.759172
## s6    -1.165754
## s7  -163.195873
## s8    39.529001
## s9    36.894803
## s10  337.423704
```

```
## s11  -45.657127
## s12 -135.072455
```

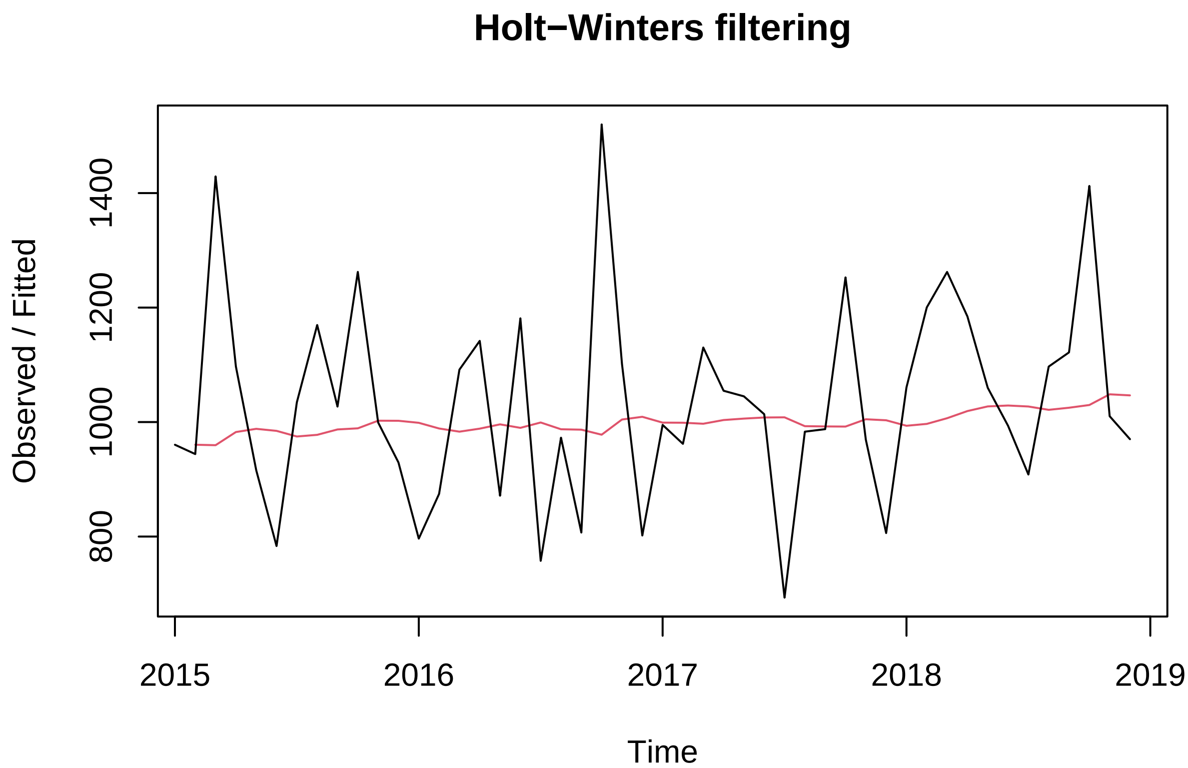
```r
HW_FDTrain2 = HoltWinters(x = FD_train, beta = FALSE, gamma = FALSE)
HW_FDTrain2
```

```
## Holt-Winters exponential smoothing without trend and without seasonal component.
##
## Call:
## HoltWinters(x = FD_train, beta = FALSE, gamma = FALSE)
##
## Smoothing parameters:
##  alpha: 0.04889481
##  beta : FALSE
##  gamma: FALSE
##
## Coefficients:
##       [,1]
## a 1042.894
```

```r
plot(HW_FDTrain2)
```

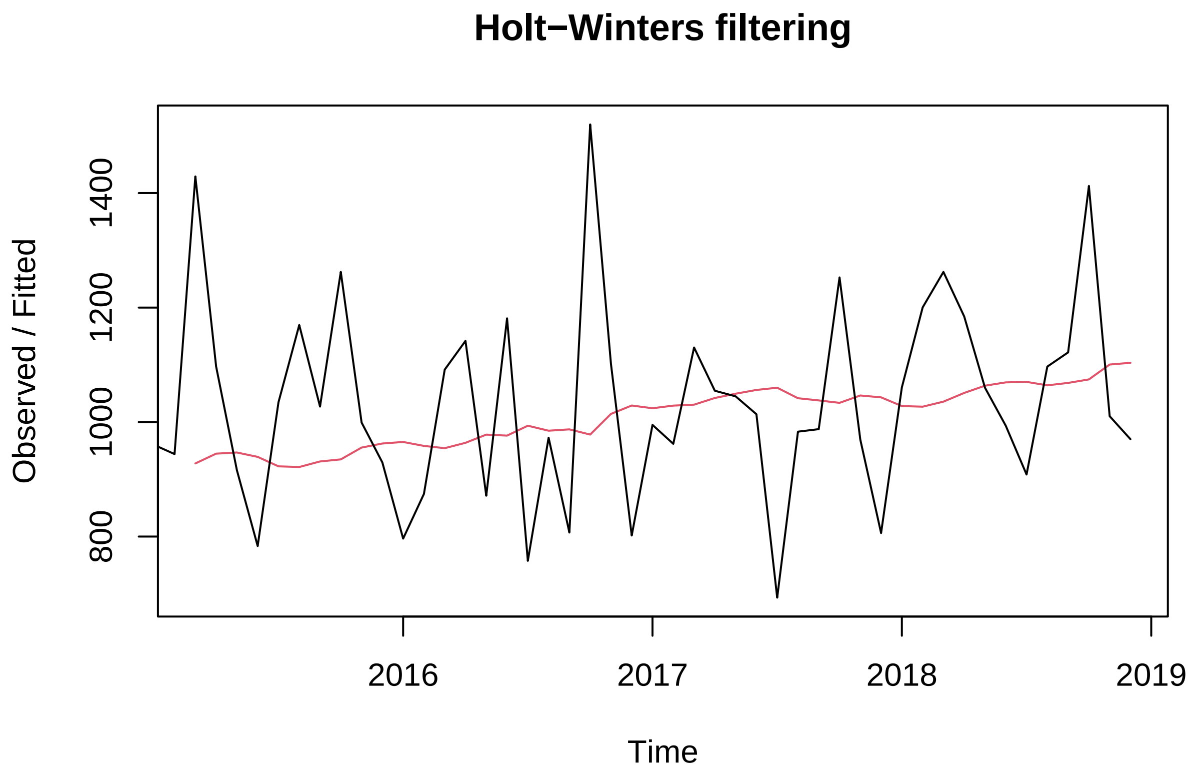



```r
SSE_Simple <- HoltWinters(FD_train,beta=FALSE,gamma=FALSE)
plot(SSE_Simple)
```



SSE with Trend but no seasonality so gamma = FALSE

```
SSE_Trend <- HoltWinters(FD_train,gamma=FALSE)
plot(SSE_Trend)
```




```
SSE_Simple$SSE
```

```
## [1] 1534136
```

```
SSE_Trend$SSE
```

```
## [1] 1617916
```

ETS

```
ets_forecast <- ets(FD_train)
attributes(ets_forecast)
```

```
## $names
##  [1] "loglik"     "aic"        "bic"        "aicc"       "mse"
##  [6] "amse"       "fit"        "residuals"  "fitted"     "states"
## [11] "par"        "m"          "method"     "series"     "components"
## [16] "call"       "initstate"  "sigma2"     "x"
##
## $class
## [1] "ets"
```

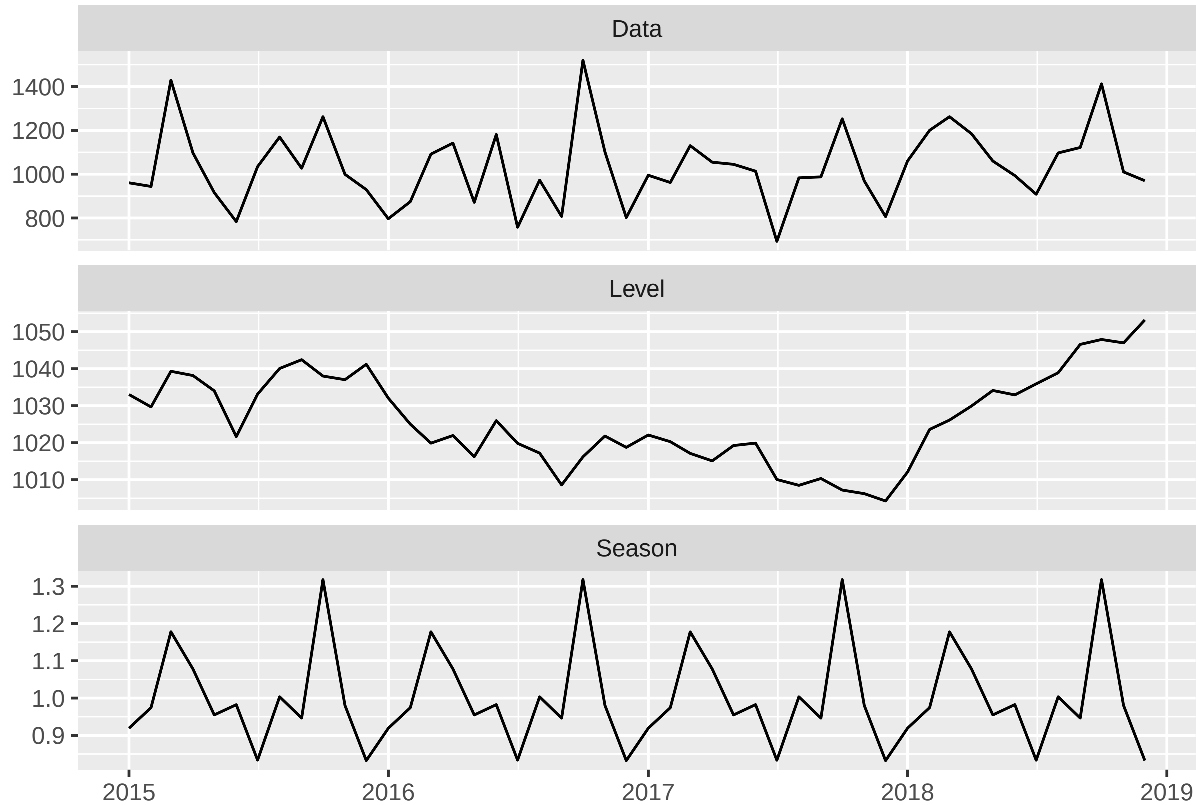
```r
ets_forecast$mse
```

```
## [1] 11215.5
```

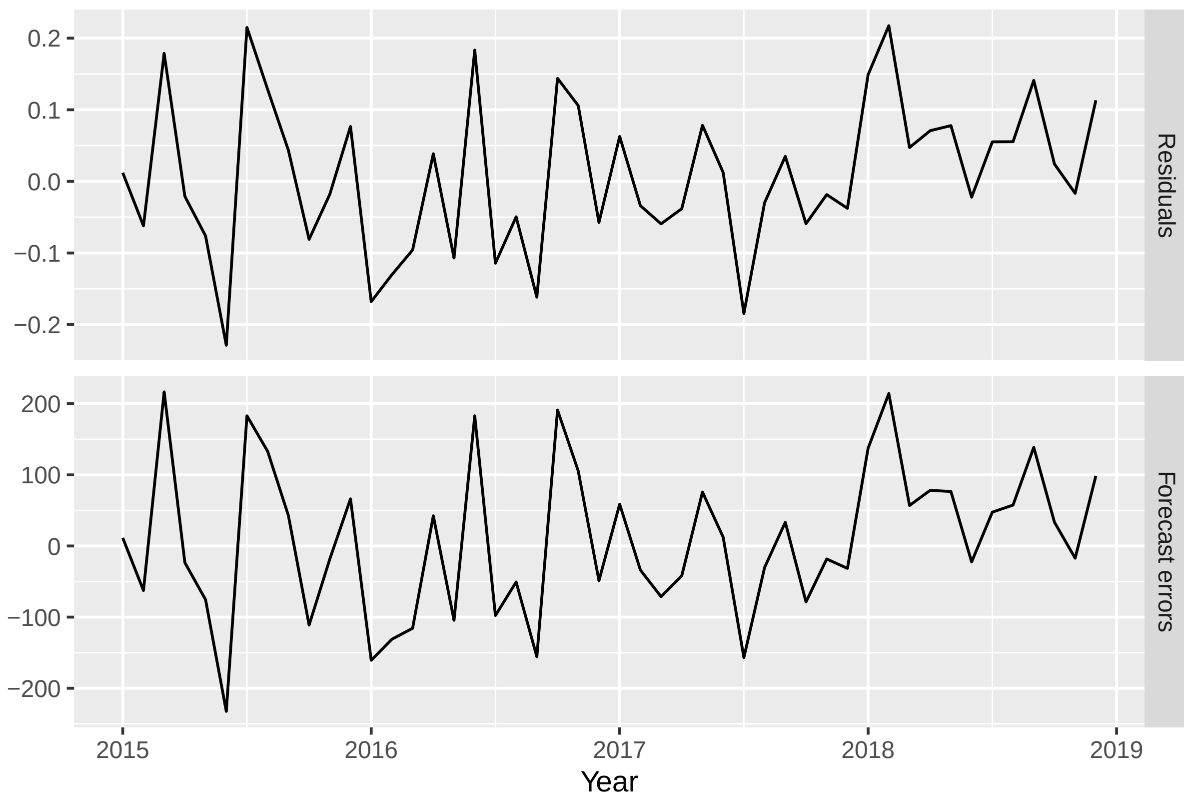
```r
ets_forecast
```

```
## ETS(M,N,M)
##
## Call:
##  ets(y = FD_train)
##
##   Smoothing parameters:
##     alpha = 0.0523
##     gamma = 0.0012
##
##   Initial states:
##     l = 1032.3831
##     s = 0.8324 0.9804 1.3176 0.9464 1.0033 0.8335
##            0.9825 0.9549 1.078 1.1775 0.9744 0.9192
##
##   sigma:  0.1249
##
##      AIC     AICc      BIC
## 664.3594 679.3594 692.4274
```

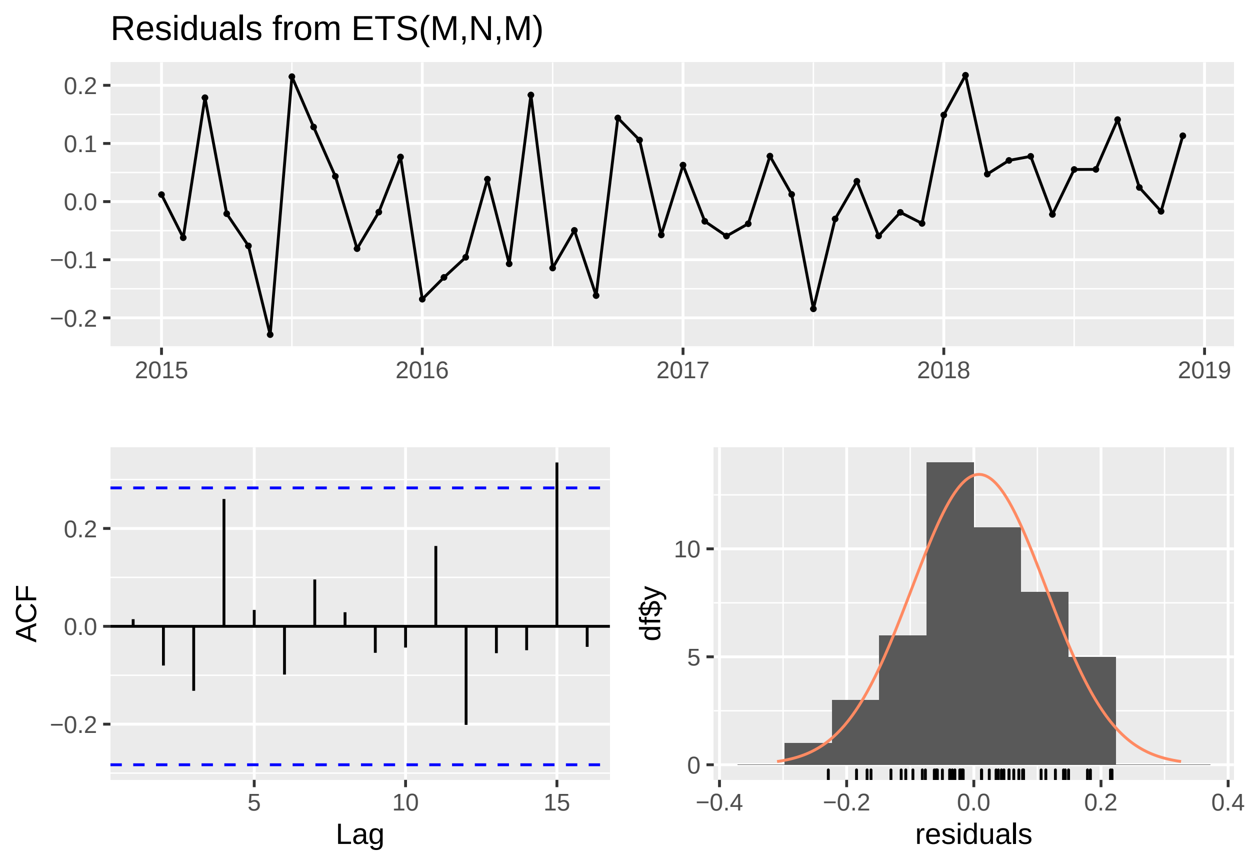
```r
autoplot(ets_forecast)
```




```
cbind('Residuals' = residuals(ets_forecast),
 'Forecast errors' = residuals(ets_forecast,type='response')) %>%
 autoplot(facet=TRUE) + xlab("Year") + ylab("")
```




```
checkresiduals(ets_forecast)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,N,M)
## Q* = 20.465, df = 3, p-value = 0.000136
##
## Model df: 14.   Total lags used: 17
```

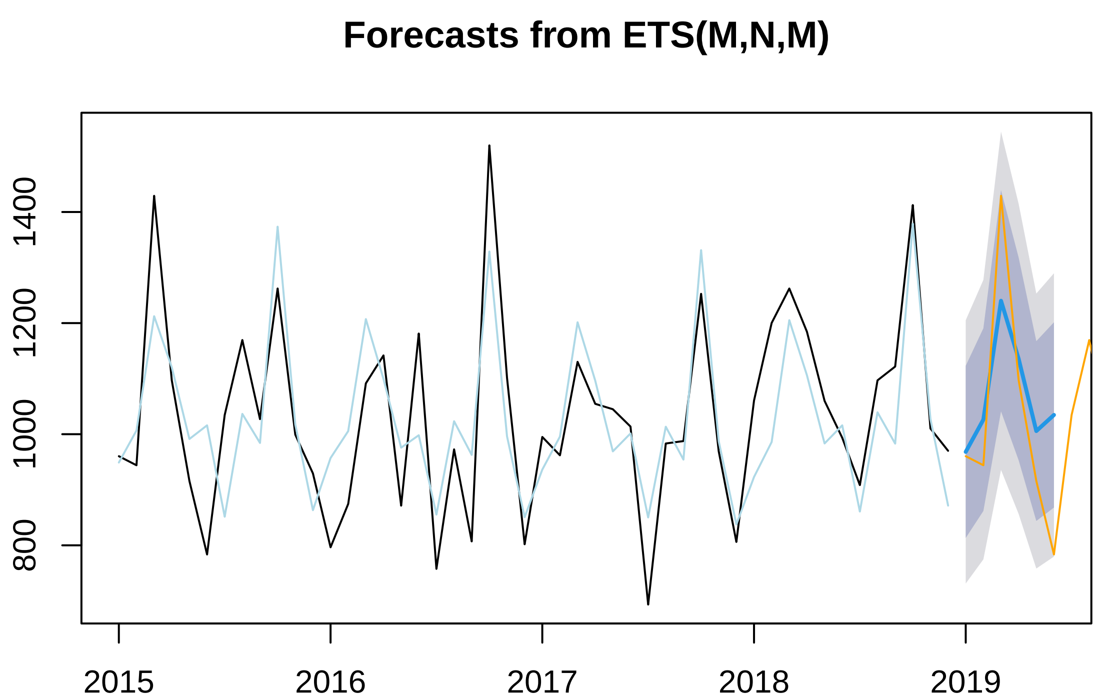
Residuals are skewed right, not the most normal we've seen so far.

Forecast with ETS

```r
# forecast with ets
forecast.ets(ets_forecast, h=6)
```

```
##          Point Forecast     Lo 80    Hi 80    Lo 95    Hi 95
## Jan 2019        968.184  813.2283 1123.140 731.1997 1205.168
## Feb 2019       1026.268  861.7881 1190.748 774.7176 1277.818
## Mar 2019       1240.202 1041.1600 1439.245 935.7933 1544.611
## Apr 2019       1135.404  952.9295 1317.879 856.3332 1414.475
## May 2019       1005.668  843.8214 1167.515 758.1447 1253.192
## Jun 2019       1034.692  867.9455 1201.439 779.6752 1289.709
```

```
forecast_ets <- forecast.ets(ets_forecast, h=6)
plot(forecast_ets)
lines(forecast_ets$fitted, col = "lightblue")
lines(FD_test, col = 'orange')
```



The forecast shows the forecasted points fall within the cofidence intervals

```
accuracy(forecast_ets)
```

```
##                    ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 8.442611 105.9033 87.19015 -0.2842957 8.633811 0.6178793
##                    ACF1
## Training set 0.02967374
```

```
accuracy(Forecast_Train12)
```

```
##                    ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 8.442611 105.9033 87.19015 -0.2842957 8.633811 0.6178793
##                    ACF1
## Training set 0.02967374
```

```
accuracy(FitFore12)
```

```
##                    ME     RMSE      MAE         MPE      MAPE       MASE
```

```
## Training set 7.149583 88.13674 72.24431 -0.02977668 7.049975 0.5119645
##                      ACF1
## Training set -0.04022578
```

```
accuracy(rwf_forecast)
```

```
##                    ME     RMSE      MAE       MPE     MAPE     MASE       ACF1
## Training set 0.206383 241.3982 194.2936 -2.591405 18.87992 1.376876 -0.3923039
```

ETS Forecast is the same as the fit forecast?

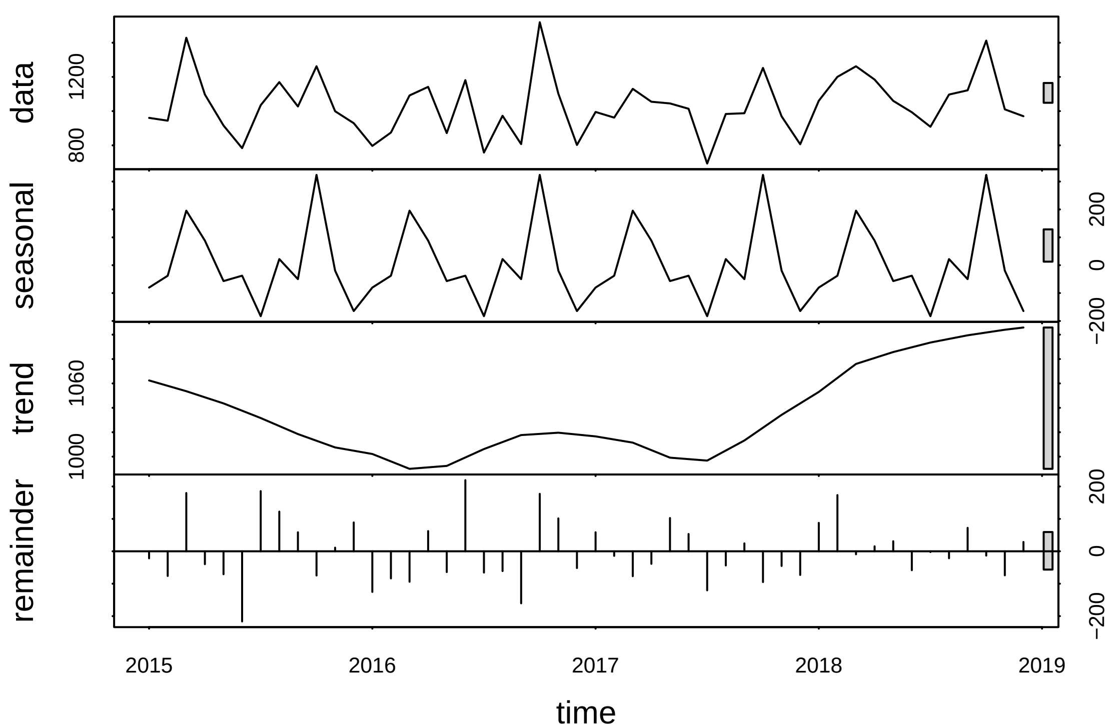
Decompositiion

```
stl_decomp <- stl(FD_train,s.window = 'periodic')
stl_decomp
```

```
##  Call:
##  stl(x = FD_train, s.window = "periodic")
##
## Components
##           seasonal     trend   remainder
## Jan 2015  -80.24561 1062.4550  -21.789419
## Feb 2015  -37.81585 1058.0382  -76.142381
## Mar 2015  195.58890 1053.6214  179.909677
## Apr 2015   88.16603 1048.6299  -39.795893
## May 2015  -56.94693 1043.6383  -70.841376
## Jun 2015  -37.66431 1037.6384 -216.524096
## Jul 2015 -182.87953 1031.6385  185.761019
## Aug 2015   21.95281 1025.0648  122.482424
## Sep 2015  -50.09491 1018.4910   58.683883
## Oct 2015  324.02207 1013.0154  -74.717425
## Nov 2015  -19.56858 1007.5397   11.278884
## Dec 2015 -164.51421 1004.8528   89.081460
## Jan 2016  -80.24561 1002.1658 -125.500201
## Feb 2016  -37.81585  996.0810  -83.715141
## Mar 2016  195.58890  989.9962  -94.035062
## Apr 2016   88.16603  991.1921   62.481865
## May 2016  -56.94693  992.3881  -64.081122
## Jun 2016  -37.66431  999.3013  219.573013
## Jul 2016 -182.87953 1006.2145  -65.745018
## Aug 2016   21.95281 1011.9359  -61.158694
## Sep 2016  -50.09491 1017.6572 -160.542316
## Oct 2016  324.02207 1018.6284  177.269570
## Nov 2016  -19.56858 1019.5995  101.639074
## Dec 2016 -164.51421 1018.0979  -51.753707
## Jan 2017  -80.24561 1016.5963   58.739275
## Feb 2017  -37.81585 1014.0360  -14.220179
## Mar 2017  195.58890 1011.4757  -76.824613
## Apr 2017   88.16603 1005.3158  -38.771854
## May 2017  -56.94693  999.1559  102.740991
## Jun 2017  -37.66431  997.9515   53.442775
```

```
## Jul 2017 -182.87953  996.7471 -120.537606
## Aug 2017   21.95281 1004.9838  -43.686579
## Sep 2017  -50.09491 1013.2204   24.514502
## Oct 2017  324.02207 1023.7274  -95.059453
## Nov 2017  -19.56858 1034.2344  -45.355789
## Dec 2017 -164.51421 1043.6364  -73.022142
## Jan 2018  -80.24561 1053.0383   87.777269
## Feb 2018  -37.81585 1064.5129  173.552900
## Mar 2018  195.58890 1075.9876   -9.326450
## Apr 2018   88.16603 1080.8642   15.419734
## May 2018  -56.94693 1085.7409   31.126005
## Jun 2018  -37.66431 1089.6332  -58.418844
## Jul 2018 -182.87953 1093.5254   -2.275859
## Aug 2018   21.95281 1096.4897  -21.512523
## Sep 2018  -50.09491 1099.4540   72.390867
## Oct 2018  324.02207 1101.7365  -13.288588
## Nov 2018  -19.56858 1104.0190  -74.200425
## Dec 2018 -164.51421 1105.8608   28.773428
```

```
plot(stl_decomp)
```



Again the data shows seasonality

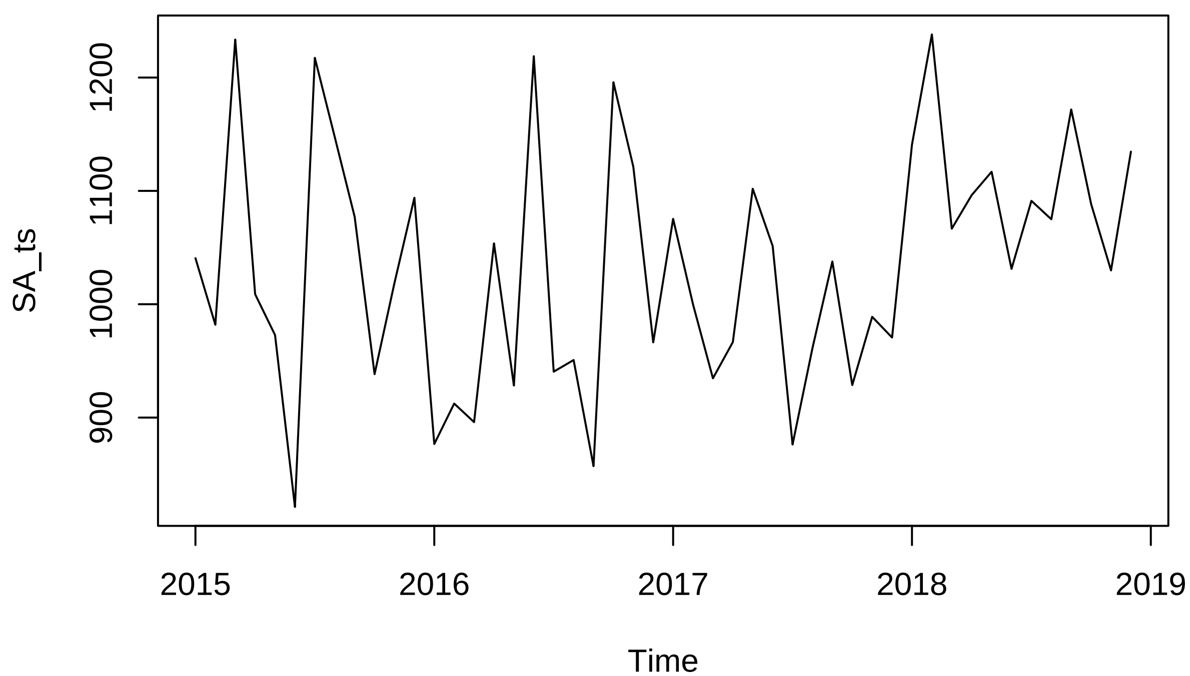
Table of seasonal adjustment

```
SA_ts <- seasadj(stl_decomp)
SA_ts
```

```
##            Jan       Feb       Mar       Apr       May       Jun       Jul
## 2015 1040.6656  981.8958 1233.5311 1008.8340  972.7969  821.1143 1217.3995
## 2016  876.6656  912.3658  895.9611 1053.6740  928.3069 1218.8743  940.4695
## 2017 1075.3356  999.8158  934.6511  966.5440 1101.8969 1051.3943  876.2095
## 2018 1140.8156 1238.0658 1066.6611 1096.2840 1116.8669 1031.2143 1091.2495
##            Aug       Sep       Oct       Nov       Dec
## 2015 1147.5472 1077.1749  938.2979 1018.8186 1093.9342
## 2016  950.7772  857.1149 1195.8979 1121.2386  966.3442
## 2017  961.2972 1037.7349  928.6679  988.8786  970.6142
## 2018 1074.9772 1171.8449 1088.4479 1029.8186 1134.6342
```

It's hard to draw conclusions with the data in this format.

```
plot(SA_ts)
```



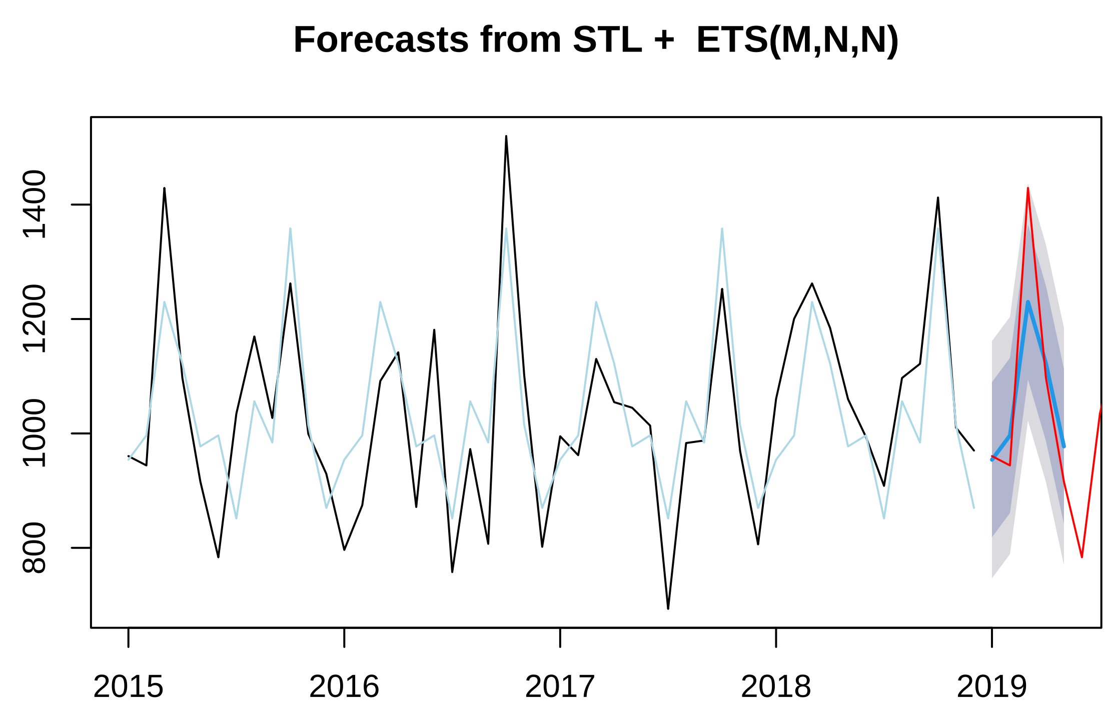
The Data looks even more volatile when you adjust for seasonality

Forecasting the decomposition is typically not very useful, but we will try it.

```
# forecast
f_stl <- forecast(stl_decomp, h=5)
f_stl
```

```
##          Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## Jan 2019       954.0297  818.4401 1089.619  746.6632 1161.396
## Feb 2019       996.4595  860.8698 1132.049  789.0930 1203.826
## Mar 2019      1229.8642 1094.2746 1365.454 1022.4977 1437.231
## Apr 2019      1122.4414  986.8517 1258.031  915.0749 1329.808
## May 2019       977.3284  841.7387 1112.918  769.9619 1184.695
```

```
plot(f_stl)
lines(f_stl$fitted, col = "lightblue")
lines(FD_test, col = 'red')
```




```
accuracy(f_stl)
```

```
##                       ME     RMSE      MAE       MPE     MAPE     MASE
## Training set -0.00529333 103.5708 86.01888 -1.071521 8.579715 0.609579
##                    ACF1
## Training set 0.05904374
```

```r
accuracy(ets_forecast)
```

```
##                    ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 8.442611 105.9033 87.19015 -0.2842957 8.633811 0.6178793
##                    ACF1
## Training set 0.02967374
```

This appears to be the strongest forecast model we have found. Lower MAPE and RMSE

Multiple Regression Analysis

```r
setDT(Tng_Ctr_Hour2)
class(Tng_Ctr_Hour2)
```

```
## [1] "data.table" "data.frame"
```

```r
Tng_Ctr_Hour2[, Quarter := factor(Quarter, ordered = T)]
Tng_Ctr_Hour2[, Month := factor(Month, ordered = T)]
summary(Tng_Ctr_Hour2)
```

```
##       Year     Quarter     Month      Device_Hrs     DH_Prev_Year
##  2015-01: 1   Q1:21   Apr    : 7   Min.   : 222.8   Length:81
##  2015-02: 1   Q2:21   Aug    : 7   1st Qu.: 899.0   Class :character
##  2015-03: 1   Q3:21   Feb    : 7   Median :1008.0   Mode  :character
##  2015-04: 1   Q4:18   Jan    : 7   Mean   : 990.1
##  2015-05: 1           Jul    : 7   3rd Qu.:1101.7
##  2015-06: 1           Jun    : 7   Max.   :1519.9
##  (Other):75           (Other):39
##  DH_YoY_Change      DH_YoY_Ch_Per      Total_Inst_Hrs
##  Length:81          Length:81          Min.   : 504.6
##  Class :character   Class :character   1st Qu.:1937.3
##  Mode  :character   Mode  :character   Median :2203.2
##                                        Mean   :2165.7
##                                        3rd Qu.:2446.8
##                                        Max.   :3084.1
##
##  Total_Inst_Hrs_Prev_Year Inst_Hrs_YoY_Change Total_Inst_Hrs_YoY_Change_Per2
##  Length:81                Length:81           Length:81
##  Class :character         Class :character    Class :character
##  Mode  :character         Mode  :character    Mode  :character
##
##
##
##
##    Cons_Sent          NJURN             RPM                CPIUrban
##  Min.   : 70.30   Min.   : 2.900   Min.   :  2908236   Min.   :234.7
##  1st Qu.: 89.00   1st Qu.: 4.100   1st Qu.: 68459347   1st Qu.:241.2
##  Median : 93.80   Median : 4.900   Median : 77115921   Median :250.8
##  Mean   : 91.49   Mean   : 5.615   Mean   : 70822495   Mean   :250.2
```

```
##  3rd Qu.: 97.90   3rd Qu.: 6.200   3rd Qu.: 85326186   3rd Qu.:257.4
##  Max.   :101.40   Max.   :16.600   Max.   :101794185   Max.   :274.1
##                                    NA's   :1
##    CPIMedian
##  Min.   :0.9755
##  1st Qu.:2.1551
##  Median :2.5922
##  Mean   :2.5862
##  3rd Qu.:2.9557
##  Max.   :5.5690
##
```

```
str(Tng_Ctr_Hour2)
```

```
## Classes 'data.table' and 'data.frame':   81 obs. of  16 variables:
##  $ Year                          : Factor w/ 81 levels "2015-01","2015-02",..: 1 2 3 4 5 6 7 8 9 10
##  $ Quarter                       : Ord.factor w/ 4 levels "Q1"<"Q2"<"Q3"<..: 1 1 1 2 2 2 3 3 3 4 ...
##  $ Month                         : Ord.factor w/ 12 levels "Apr"<"Aug"<"Dec"<..: 5 4 8 1 9 7 6 2 12
##  $ Device_Hrs                    : num  960 944 1429 1097 916 ...
##  $ DH_Prev_Year                  : chr  "NA" "NA" "NA" "NA" ...
##  $ DH_YoY_Change                 : chr  "NA" "NA" "NA" "NA" ...
##  $ DH_YoY_Ch_Per                 : chr  "NA" "NA" "NA" "NA" ...
##  $ Total_Inst_Hrs                : num  1701 1614 2533 2152 1695 ...
##  $ Total_Inst_Hrs_Prev_Year      : chr  "NA" "NA" "NA" "NA" ...
##  $ Inst_Hrs_YoY_Change           : chr  "NA" "NA" "NA" "NA" ...
##  $ Total_Inst_Hrs_YoY_Change_Per2: chr  "NA" "NA" "NA" "NA" ...
##  $ Cons_Sent                     : num  98.1 95.4 93 95.9 90.7 96.1 93.1 91.9 87.2 90 ...
##  $ NJURN                         : num  6.8 6.7 6.3 5.8 6 5.9 6.2 5.5 5.1 4.9 ...
##  $ RPM                           : num  65975447 59784666 75751609 73090871 78002935 ...
##  $ CPIUrban                      : num  235 235 236 236 237 ...
##  $ CPIMedian                     : num  1.95 1.95 2.43 2.96 2.5 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
Mach1 = subset(Tng_Ctr_Hour2, select = c(Month, Device_Hrs, Cons_Sent, NJURN, RPM, CPIUrban, CPIMedian)
```

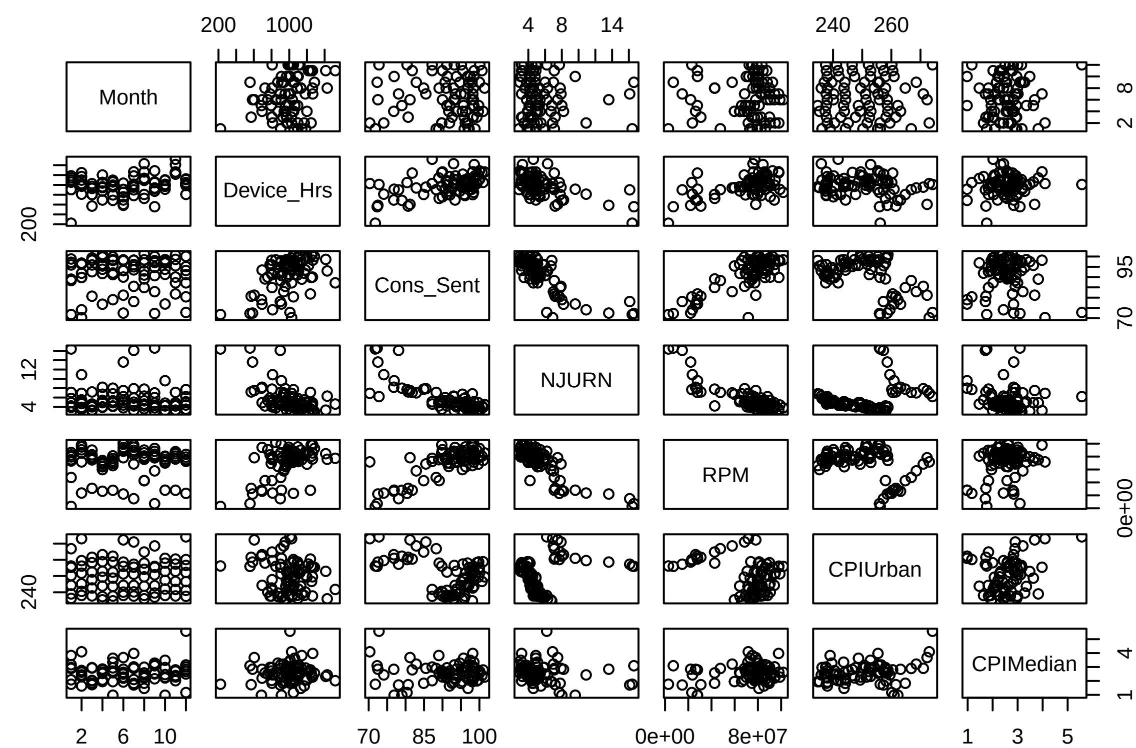
```
LM_Mach = lm(Device_Hrs ~ ., Mach1)
summary(LM_Mach )
```

```
##
## Call:
## lm(formula = Device_Hrs ~ ., data = Mach1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -377.13  -74.78    2.32   71.19  347.28
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.403e+02  7.492e+02   0.454 0.651282
## Month.L      2.047e+02  5.664e+01   3.613 0.000602 ***
## Month.Q      1.537e+02  5.915e+01   2.599 0.011634 *
## Month.C     -8.334e+01  5.726e+01  -1.455 0.150511
```

```
## Month^4     -1.169e+02  5.960e+01  -1.962 0.054223 .
## Month^5     -7.904e+01  6.044e+01  -1.308 0.195718
## Month^6     -9.531e+01  7.399e+01  -1.288 0.202428
## Month^7     -1.183e+02  5.881e+01  -2.011 0.048622 *
## Month^8     -2.306e+02  5.606e+01  -4.113 0.000115 ***
## Month^9      1.546e+02  5.682e+01   2.722 0.008389 **
## Month^10     4.619e+01  6.022e+01   0.767 0.445946
## Month^11    -8.832e+01  5.409e+01  -1.633 0.107486
## Cons_Sent    1.811e+00  4.672e+00   0.388 0.699692
## NJURN       -4.201e+00  1.609e+01  -0.261 0.794845
## RPM          4.709e-06  2.417e-06   1.949 0.055773 .
## CPIUrban     9.757e-01  1.984e+00   0.492 0.624510
## CPIMedian   -2.655e+01  2.998e+01  -0.886 0.379123
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 138.6 on 63 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.6611, Adjusted R-squared:  0.575
## F-statistic:  7.68 on 16 and 63 DF,  p-value: 1.361e-09
```

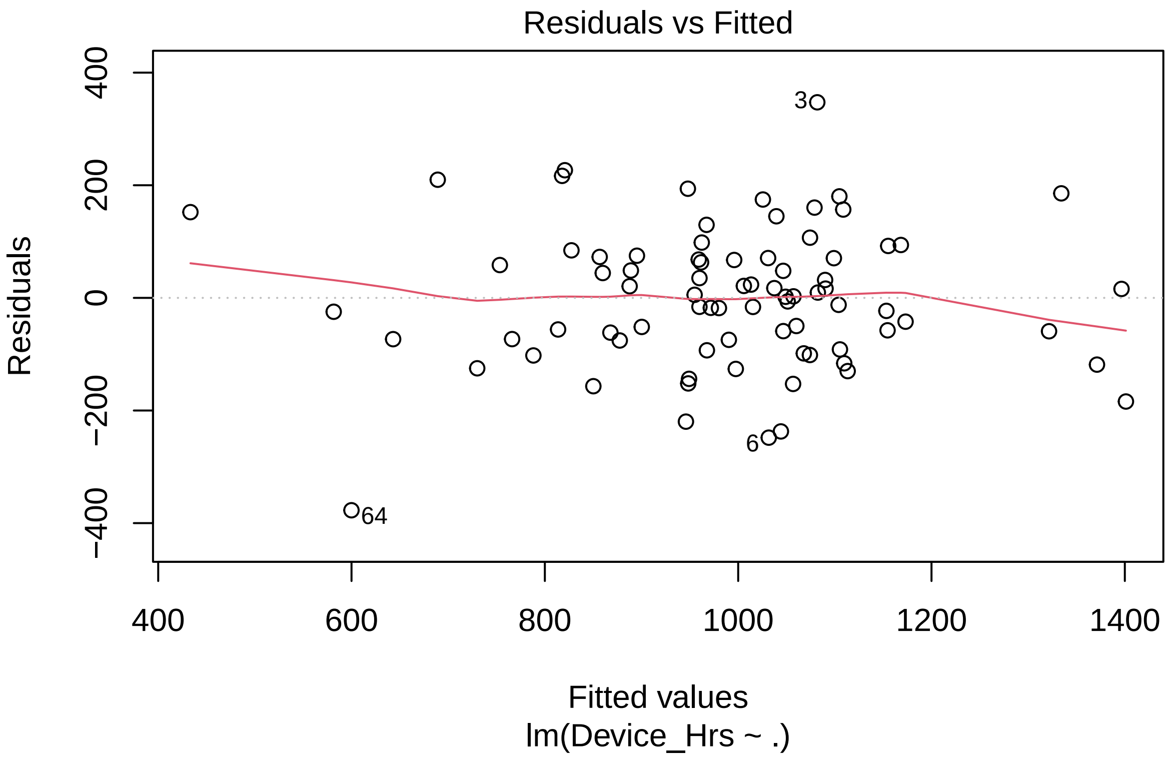
We show a relatively strong correlation between the model and Device Hours

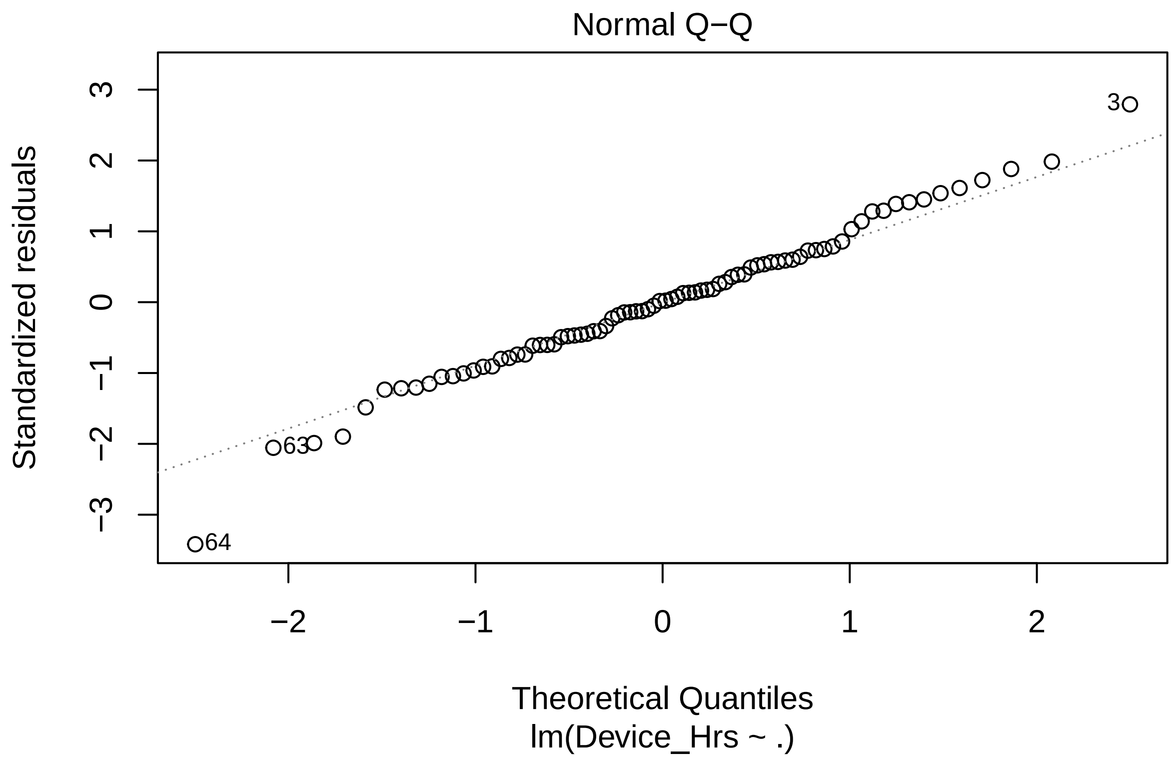
```
pairs(Mach1)
```

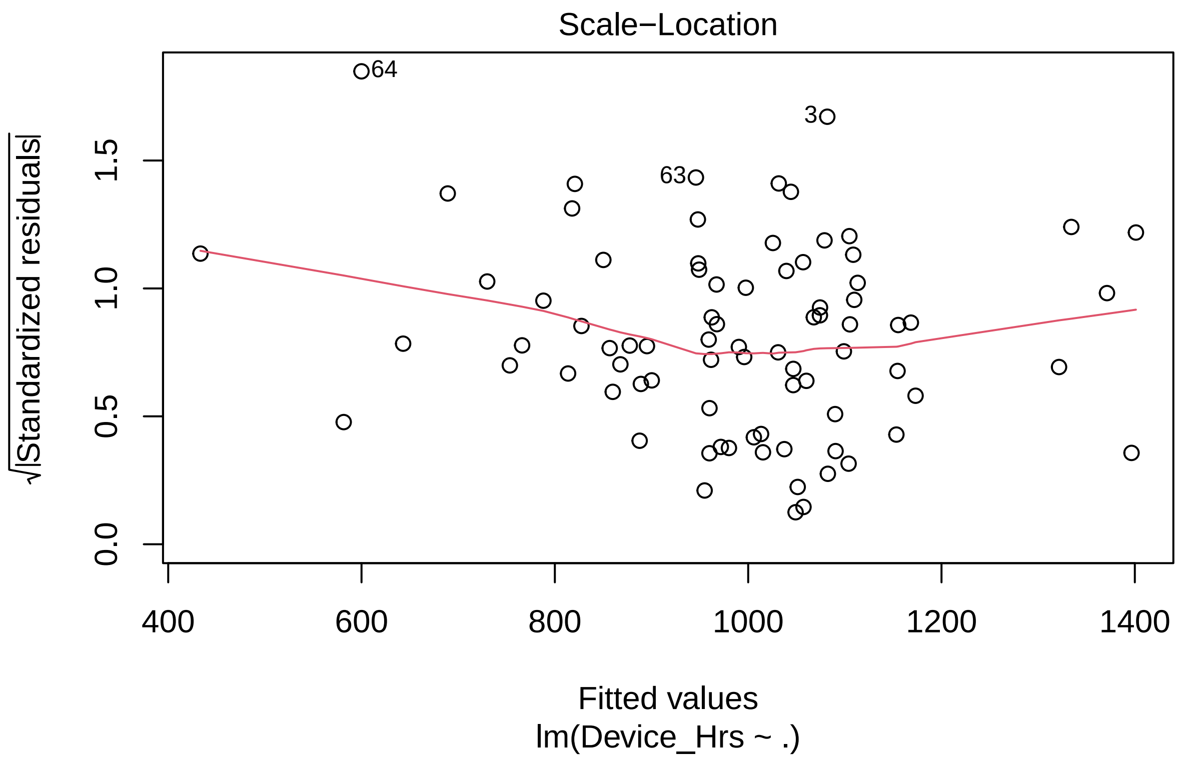


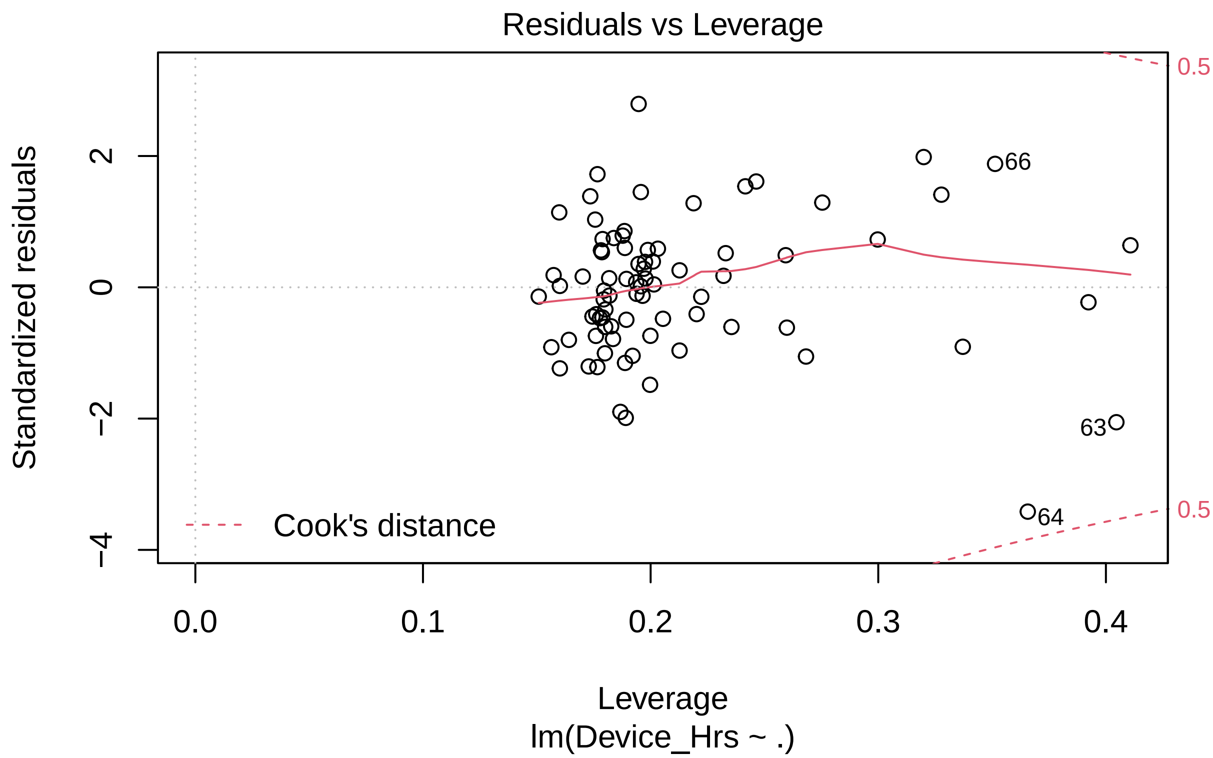
We see some correlation between Consumer Sentiment and NJ Unemployment, and Consumer Sentiment and Revenue Passenger Miles. Its possible there is some redundancy. The CPI Urban and CPI Median don't show a very strong correlation because CPI Urban is the CPI level, while CPI Median is the percentage change. That is why we're including both.

```
plot(LM_Mach)
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(Device_Hrs ~ .)
3
63
64
−3
−2
−1
0
1
2
3
−2
−1
0
1
2

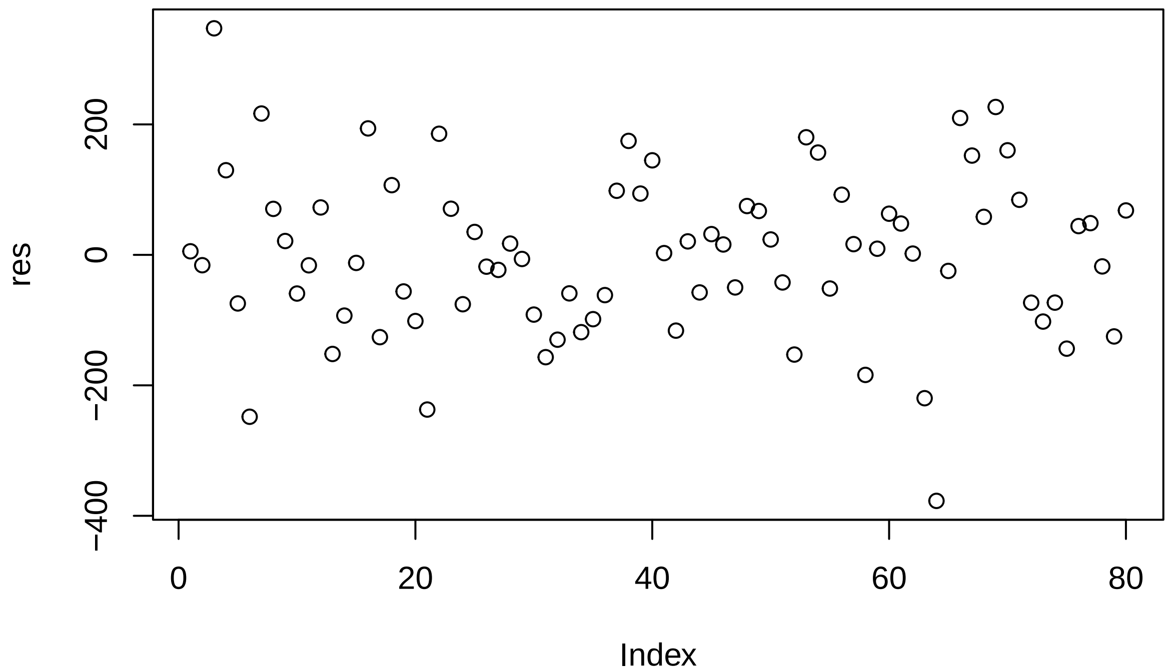

Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
400
600
800
1000
1200
1400
64
3
63
Fitted values
lm(Device_Hrs ~ .)



Our model looks like it is generally normal

Let's look at the residuals

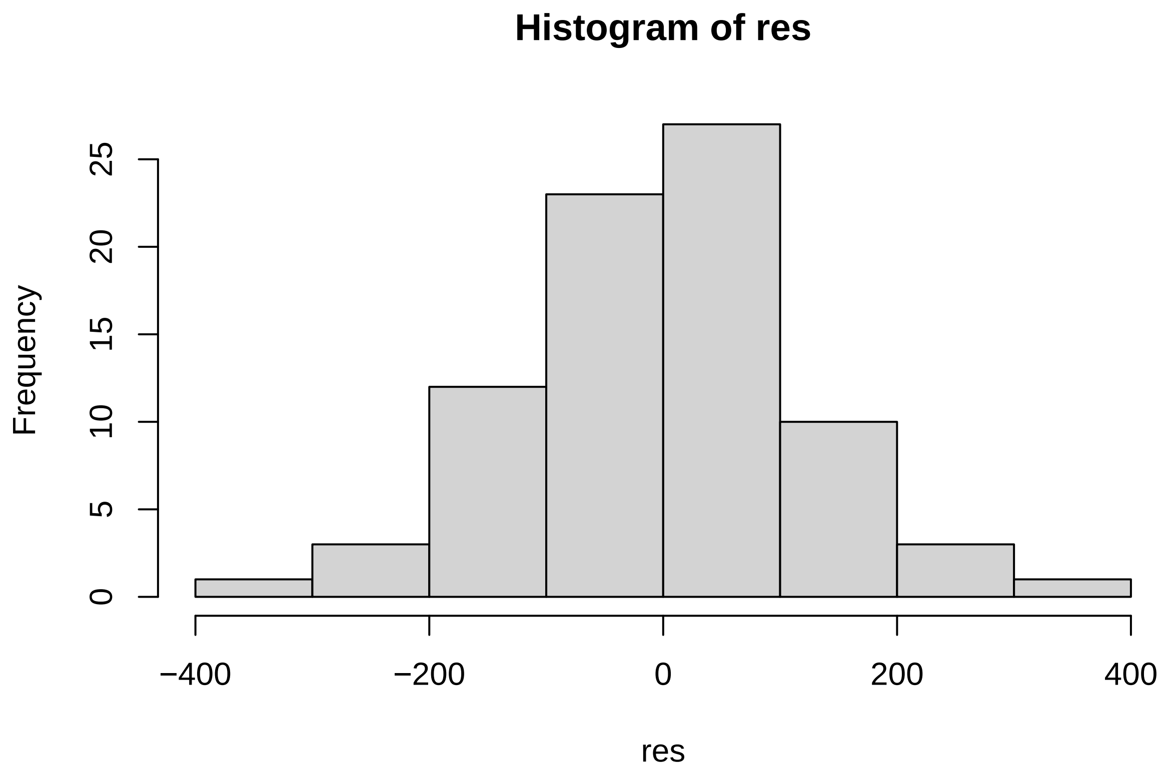
```
res= residuals(LM_Mach)
plot(res)
```



There doesn't appear to be any pattern, which leads us to believe the model is not missing any factors.

Lets take other looks at the residuals

```
hist(res)
```



This looks skewed left, so not completely normal. Maybe we can get a stronger model by removing some independent variables and simplifying the model.

Stepwise

Because we have multiple independent variables, it is important to run a stepwise to see if we can get a stronger model.

```
Mach.Step = step(LM_Mach)
```

```
## Start:  AIC=803.91
## Device_Hrs ~ Month + Cons_Sent + NJURN + RPM + CPIUrban + CPIMedian
##
##             Df Sum of Sq     RSS    AIC
## - NJURN      1      1309 1210999 801.99
## - Cons_Sent  1      2883 1212572 802.10
## - CPIUrban   1      4646 1214335 802.21
## - CPIMedian  1     15064 1224754 802.90
## <none>                   1209689 803.91
## - RPM        1     72925 1282614 806.59
## - Month     11   1127124 2336813 834.58
##
## Step:  AIC=801.99
## Device_Hrs ~ Month + Cons_Sent + RPM + CPIUrban + CPIMedian
```

```
##
##             Df Sum of Sq     RSS    AIC
## - Cons_Sent  1      3536 1214534 800.23
## - CPIUrban   1      7381 1218380 800.48
## - CPIMedian  1     15320 1226319 801.00
## <none>                   1210999 801.99
## - RPM        1    181252 1392251 811.15
## - Month     11   1280870 2491868 837.72
##
## Step:  AIC=800.23
## Device_Hrs ~ Month + RPM + CPIUrban + CPIMedian
##
##             Df Sum of Sq     RSS    AIC
## - CPIUrban   1      6674 1221208 798.67
## - CPIMedian  1     20309 1234843 799.55
## <none>                   1214534 800.23
## - Month     11   1353176 2567710 838.12
## - RPM        1    897519 2112053 842.49
##
## Step:  AIC=798.67
## Device_Hrs ~ Month + RPM + CPIMedian
##
##             Df Sum of Sq     RSS    AIC
## - CPIMedian  1     14905 1236113 797.64
## <none>                   1221208 798.67
## - Month     11   1346804 2568013 836.13
## - RPM        1   1084290 2305499 847.50
##
## Step:  AIC=797.64
## Device_Hrs ~ Month + RPM
##
##         Df Sum of Sq     RSS    AIC
## <none>               1236113 797.64
## - Month 11   1355720 2591834 834.87
## - RPM    1   1112291 2348404 846.98
```

```
Mach.Step
```

```
##
## Call:
## lm(formula = Device_Hrs ~ Month + RPM, data = Mach1)
##
## Coefficients:
## (Intercept)      Month.L      Month.Q      Month.C      Month^4      Month^5
##   6.146e+02    2.129e+02    1.539e+02   -7.292e+01   -1.308e+02   -8.675e+01
##     Month^6      Month^7      Month^8      Month^9     Month^10     Month^11
##  -8.133e+01   -1.351e+02   -2.265e+02    1.634e+02    6.577e+01   -8.436e+01
##         RPM
##   5.342e-06
```

```
summary(Mach.Step)
```

```
##
```

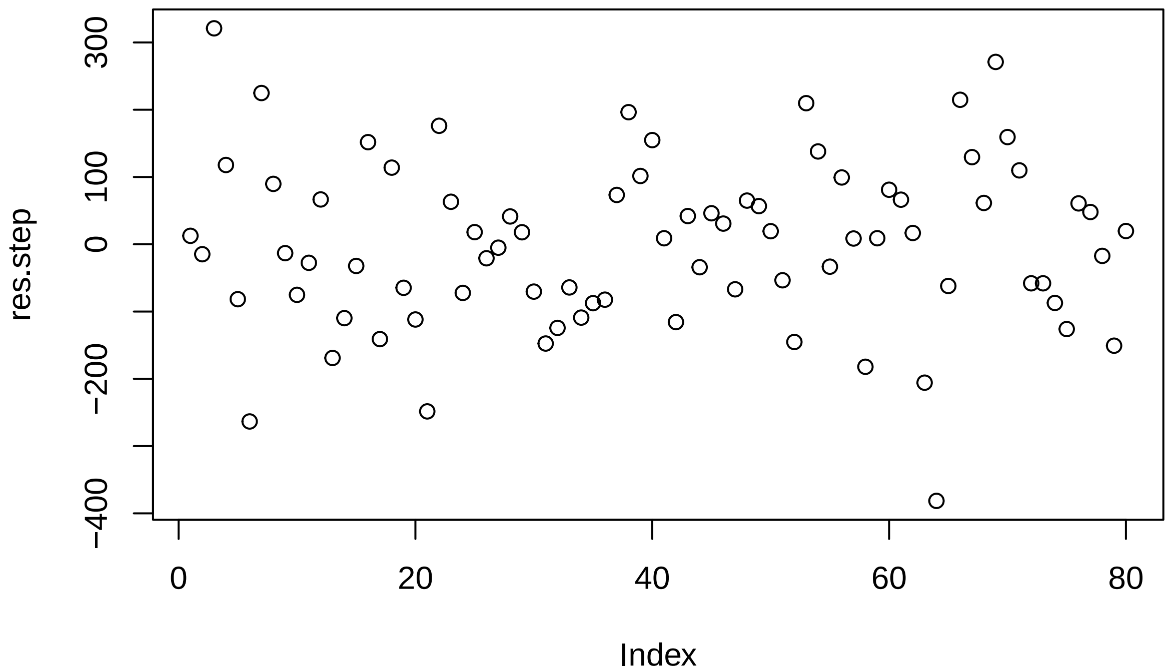
```
## Call:
## lm(formula = Device_Hrs ~ Month + RPM, data = Mach1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -381.44  -76.77    8.75   68.32  321.01
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.146e+02  5.103e+01  12.045  < 2e-16 ***
## Month.L      2.129e+02  5.354e+01   3.977 0.000174 ***
## Month.Q      1.539e+02  5.275e+01   2.918 0.004789 **
## Month.C     -7.292e+01  5.296e+01  -1.377 0.173118
## Month^4     -1.308e+02  5.343e+01  -2.448 0.017001 *
## Month^5     -8.675e+01  5.296e+01  -1.638 0.106103
## Month^6     -8.133e+01  5.364e+01  -1.516 0.134196
## Month^7     -1.351e+02  5.339e+01  -2.530 0.013759 *
## Month^8     -2.265e+02  5.353e+01  -4.232  7.2e-05 ***
## Month^9      1.634e+02  5.264e+01   3.104 0.002800 **
## Month^10     6.577e+01  5.168e+01   1.272 0.207602
## Month^11    -8.436e+01  5.141e+01  -1.641 0.105541
## RPM          5.342e-06  6.880e-07   7.765  6.4e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 135.8 on 67 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.6537, Adjusted R-squared:  0.5917
## F-statistic: 10.54 on 12 and 67 DF,  p-value: 2.594e-11
```

Stepwise reveals that the best way to predict training hours is to use the month and the Revenue Passenger Miles. The other variables were not strong predictors.

This model has a decently strong Adjusted R2, a high F-statistic, and a very low p-value.

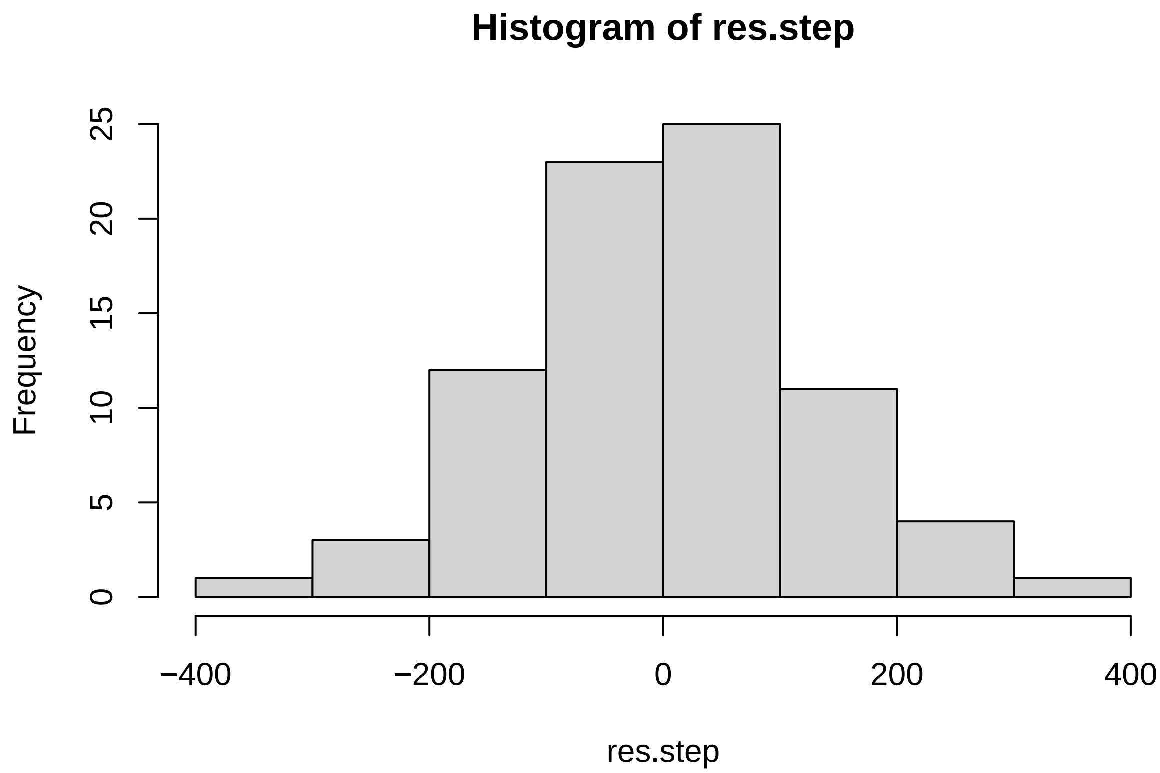
Lets look at the residuals of the new model.

```
res.step= residuals(Mach.Step)
plot(res.step)
```



This also looks normally distributed.

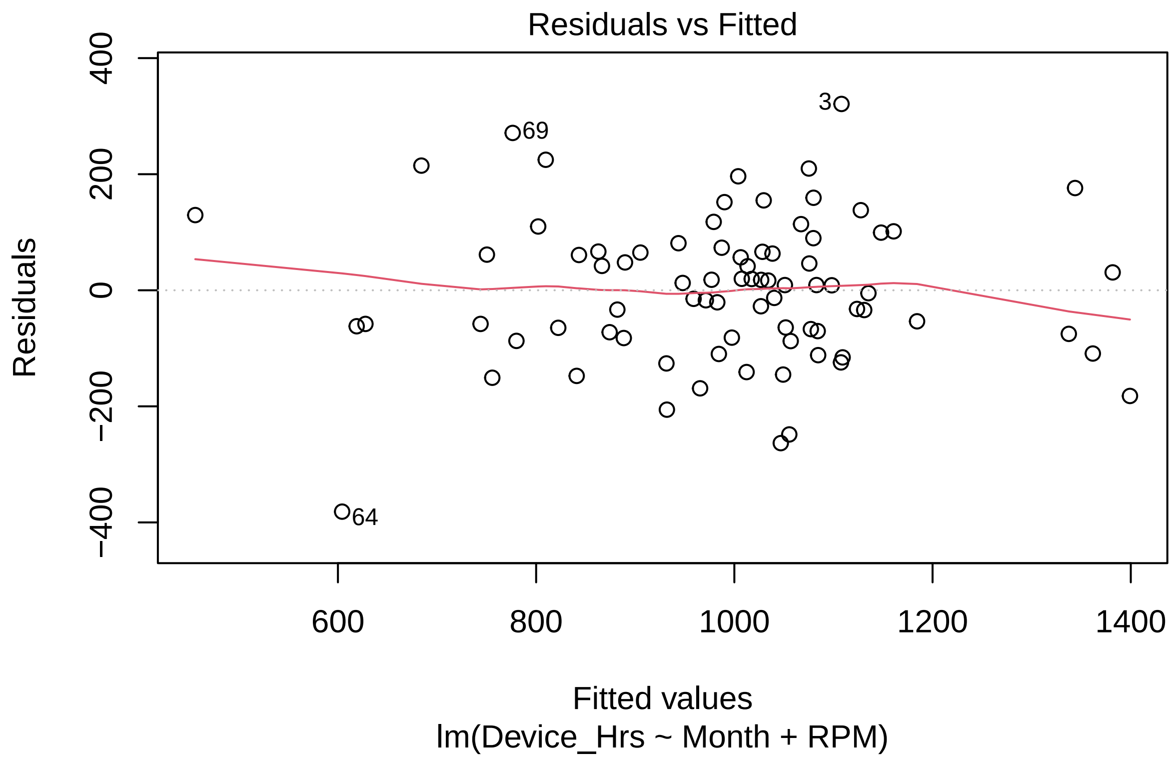
```
hist(res.step)
```

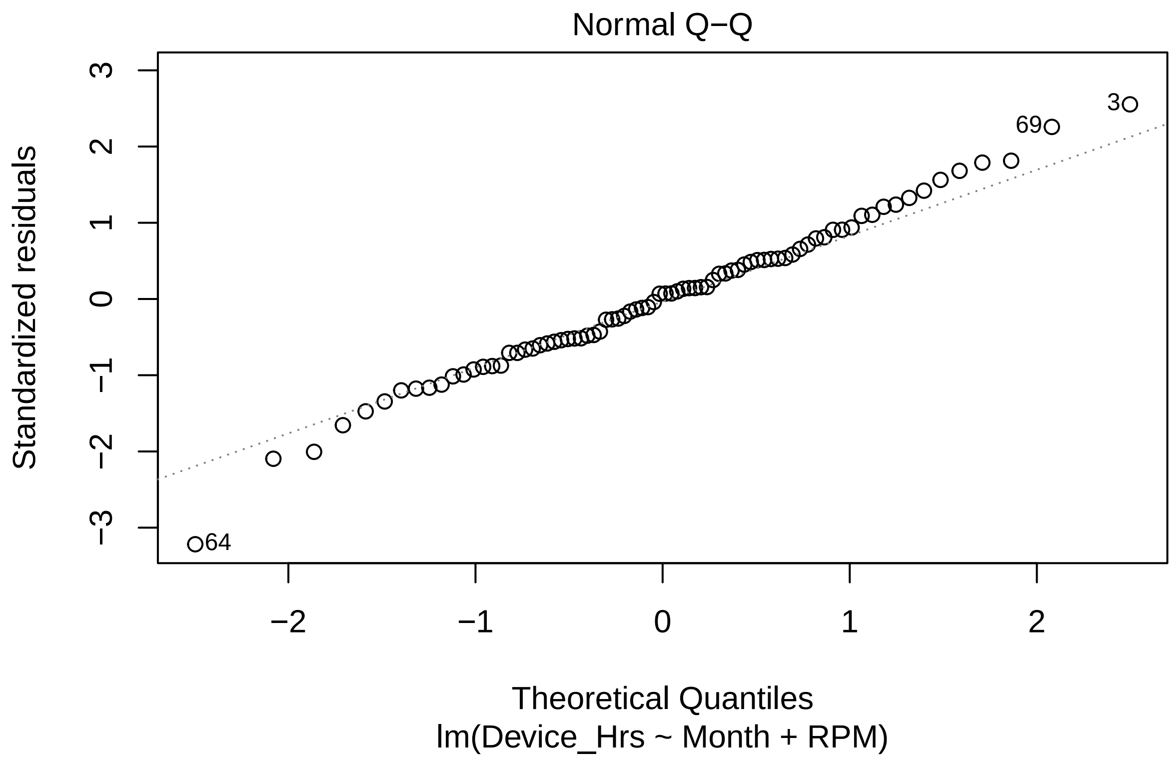


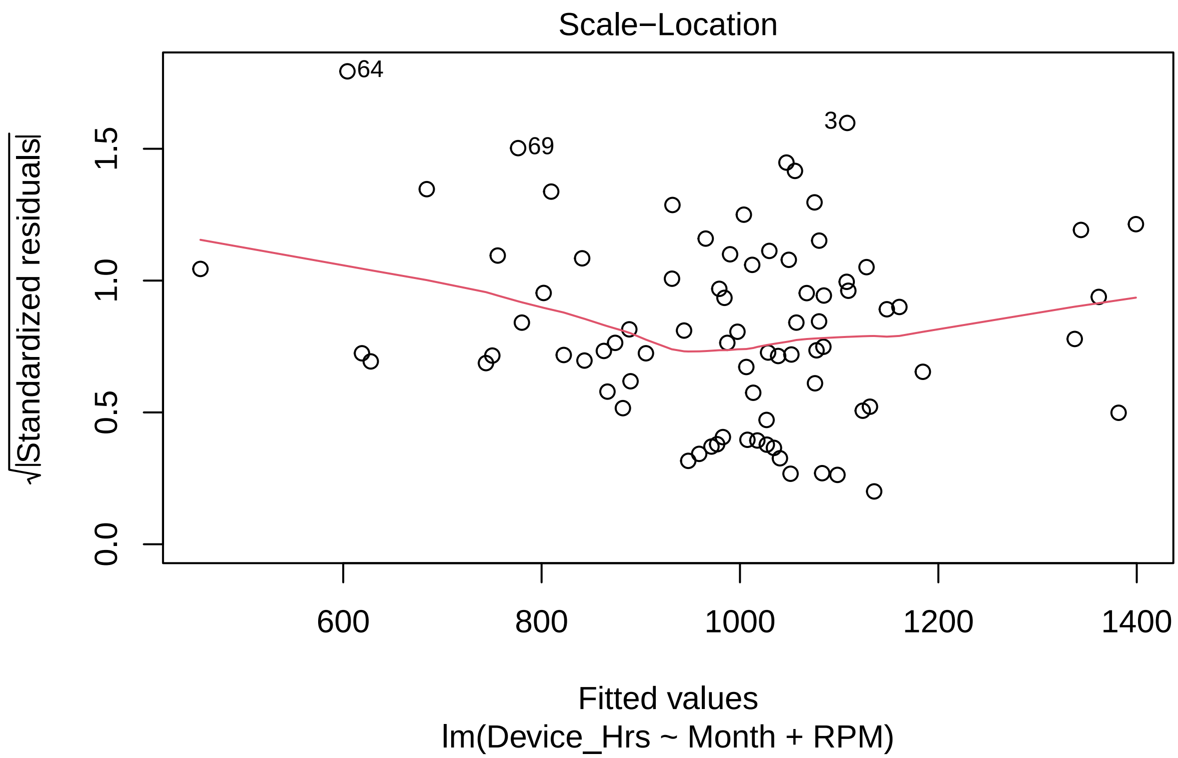
this looks to be more normally distributed, though it still slightly skewed to the right.

Final residual check.

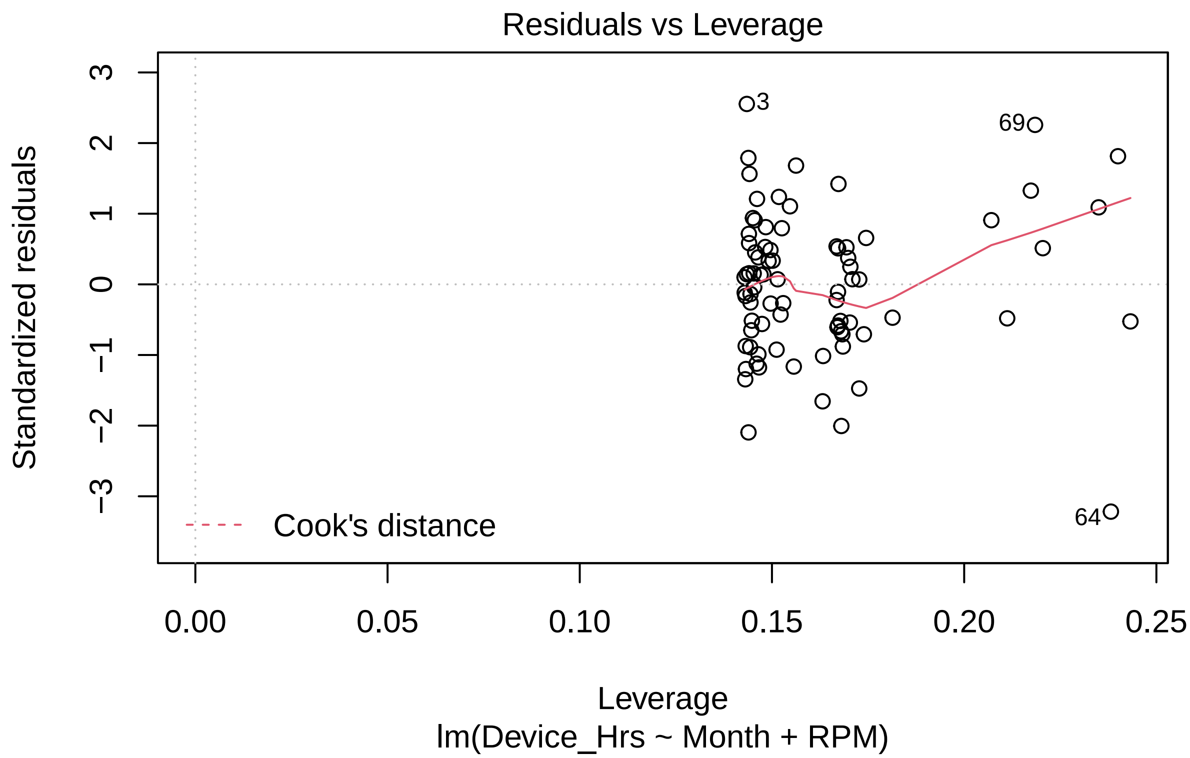
```
plot(Mach.Step)
```


Residuals vs Fitted
Residuals
400
200
0
−200
−400
3
69
64
600
800
1000
1200
1400
Fitted values
lm(Device_Hrs ~ Month + RPM)


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(Device_Hrs ~ Month + RPM)
3
69
64
−3
−2
−1
0
1
2
3



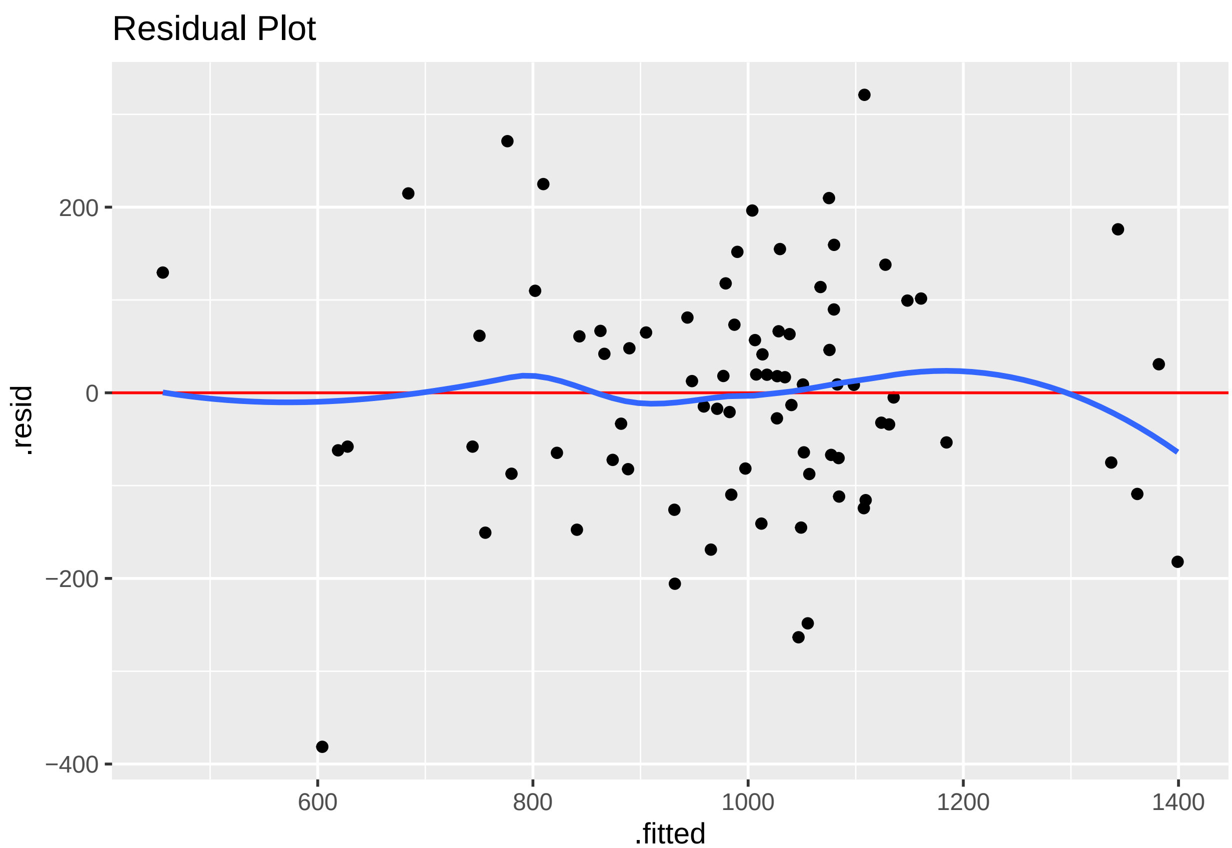
Scale−Location
64
69
3
√|Standardized residuals|
0.0
0.5
1.0
1.5
600
800
1000
1200
1400
Fitted values
lm(Device_Hrs ~ Month + RPM)



Normal QQ plot shows that the model is relatively normal.

```
p1= ggplot(Mach.Step) +
  aes(x=.fitted, y=.resid)+
  geom_point() + geom_abline(intercept = 0, slope = 0, color="red") + geom_smooth(method = "loess", se
  labs(title= "Residual Plot")
p1
```

```
## `geom_smooth()` using formula 'y ~ x'
```



We show some skewedness at the right side of the graph, but generally it appears strong..

Lets look at a comparison of the big model vs the small model.

```
anova(LM_Mach,Mach.Step)
```

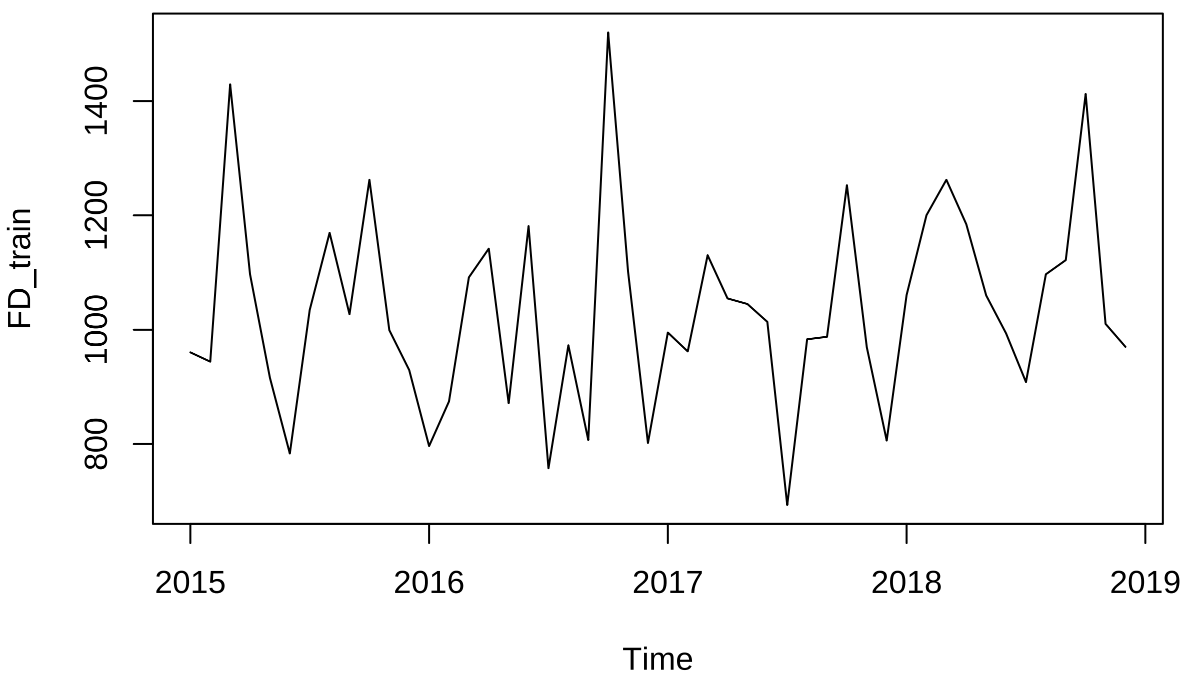
```
## Analysis of Variance Table
##
## Model 1: Device_Hrs ~ Month + Cons_Sent + NJURN + RPM + CPIUrban + CPIMedian
## Model 2: Device_Hrs ~ Month + RPM
##   Res.Df     RSS Df Sum of Sq     F Pr(>F)
## 1     63 1209689
## 2     67 1236113 -4    -26424 0.344 0.8472
```

```
anova(Mach.Step,LM_Mach)
```

```
## Analysis of Variance Table
##
## Model 1: Device_Hrs ~ Month + RPM
## Model 2: Device_Hrs ~ Month + Cons_Sent + NJURN + RPM + CPIUrban + CPIMedian
##   Res.Df     RSS Df Sum of Sq     F Pr(>F)
## 1     67 1236113
## 2     63 1209689  4     26424 0.344 0.8472
```

ARIMA Testing

```
plot(FD_train)
```




```
adf.test(FD_train)
```

```
## Warning in adf.test(FD_train): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  FD_train
## Dickey-Fuller = -5.3294, Lag order = 3, p-value = 0.01
## alternative hypothesis: stationary
```

P-Value is less than 0.01

Kipps test says differences is required if p-value is < 0.05

```r
kpss.test(FD_train)
```

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

```
##
##  KPSS Test for Level Stationarity
##
## data:  FD_train
## KPSS Level = 0.16197, Truncation lag parameter = 3, p-value = 0.1
```

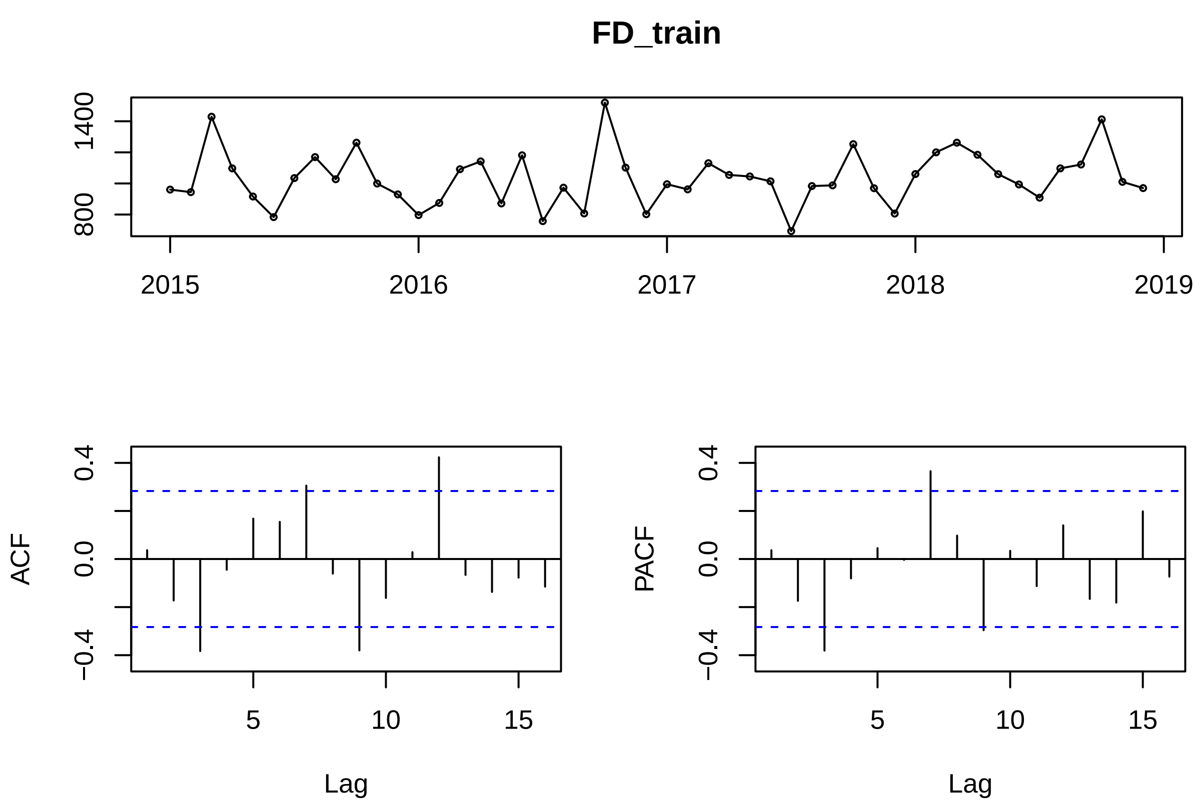
P-Value less than 0.05 so no difference needed

```r
nsdiffs(FD_train)
```

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

```r
MachDiff = ndiffs(FD_train)
```

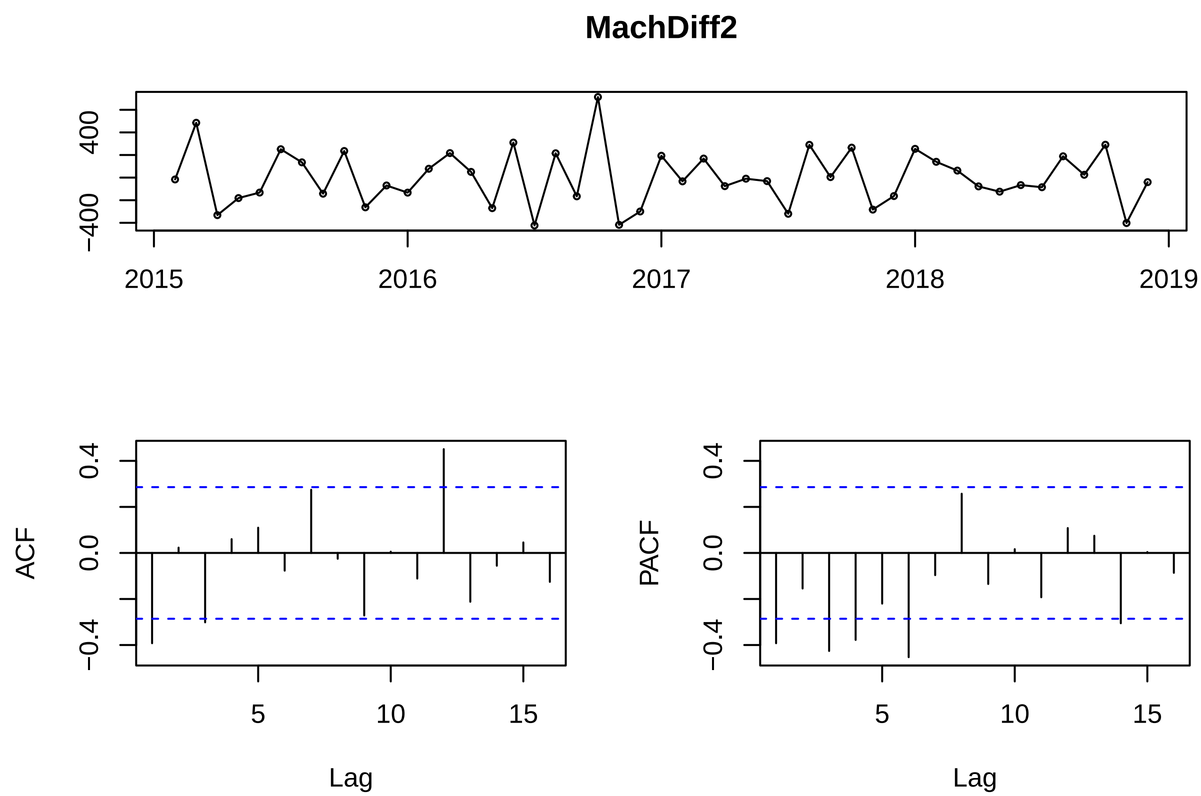
```r
#tsdisplay plots ACF,PACF and timeseries plot together. How cool!
tsdisplay(FD_train)
```



We see some significance in the ACF for months 3, 7, 9, and 12. This leads us to believe that there is some dependence. There is significance in the PACF for months 3, 7, and 9.

```
MachDiff2 <- diff(FD_train, differences=1)
```

```
tsdisplay(MachDiff2)
```



There is even ACF significance for months 1, 3, 7, 9, and 12. PACF negative in 1, 3,4,6, 8, and 14

```
ndiffs(MachDiff2)
```

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

There is no longer significance

```
ndiffs(MachDiff)
```

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

No significance

```
adf.test(MachDiff2)
```

```
## Warning in adf.test(MachDiff2): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  MachDiff2
## Dickey-Fuller = -7.117, Lag order = 3, p-value = 0.01
## alternative hypothesis: stationary
```

get the partial autocorrelation values

```
Pacf(FD_train, lag.max=20,plot=FALSE)
```

```
##
## Partial autocorrelations of series 'FD_train', by lag
##
##      1      2      3      4      5      6      7      8      9     10     11
##  0.037 -0.174 -0.381 -0.081  0.045 -0.003  0.365  0.097 -0.296  0.034 -0.113
##     12     13     14     15     16     17     18     19     20
##  0.140 -0.166 -0.181  0.198 -0.074  0.001  0.143  0.027 -0.046
```

```
auto.arima(FD_train)
```

```
## Series: FD_train
## ARIMA(0,1,1)(1,1,0)[12]
##
## Coefficients:
##           ma1     sar1
##       -0.8729  -0.3746
## s.e.   0.0903   0.1942
##
## sigma^2 estimated as 27192:  log likelihood=-229.02
## AIC=464.03   AICc=464.8   BIC=468.7
```

```
auto.arima(MachDiff)
```

```
## Series: MachDiff
## ARIMA(0,0,0) with non-zero mean
##
## Coefficients:
## intercept
##         0
##
## sigma^2 estimated as 0:  log likelihood=Inf
## AIC=-Inf   AICc=-Inf   BIC=-Inf
```

```
auto.arima(MachDiff2)
```

```
## Series: MachDiff2
## ARIMA(0,0,1)(1,1,0)[12]
##
```

```
## Coefficients:
##           ma1     sar1
##       -0.8729  -0.3746
## s.e.   0.0903   0.1942
##
## sigma^2 estimated as 27192:  log likelihood=-229.02
## AIC=464.03   AICc=464.8   BIC=468.7
```

```r
# or save the model. BIC and AIC is also given as values.
auto_fit <- auto.arima(FD_train, trace=TRUE, stepwise=FALSE)
```

```
##
##  ARIMA(0,1,0)(0,1,0)[12]                    : 484.0533
##  ARIMA(0,1,0)(0,1,1)[12]                    : Inf
##  ARIMA(0,1,0)(1,1,0)[12]                    : 481.0162
##  ARIMA(0,1,0)(1,1,1)[12]                    : Inf
##  ARIMA(0,1,1)(0,1,0)[12]                    : 465.4676
##  ARIMA(0,1,1)(0,1,1)[12]                    : Inf
##  ARIMA(0,1,1)(1,1,0)[12]                    : 464.8046
##  ARIMA(0,1,1)(1,1,1)[12]                    : Inf
##  ARIMA(0,1,2)(0,1,0)[12]                    : 467.7442
##  ARIMA(0,1,2)(0,1,1)[12]                    : Inf
##  ARIMA(0,1,2)(1,1,0)[12]                    : 467.3637
##  ARIMA(0,1,2)(1,1,1)[12]                    : Inf
##  ARIMA(0,1,3)(0,1,0)[12]                    : 470.1542
##  ARIMA(0,1,3)(0,1,1)[12]                    : Inf
##  ARIMA(0,1,3)(1,1,0)[12]                    : 469.8397
##  ARIMA(0,1,3)(1,1,1)[12]                    : Inf
##  ARIMA(0,1,4)(0,1,0)[12]                    : 466.5286
##  ARIMA(0,1,4)(0,1,1)[12]                    : Inf
##  ARIMA(0,1,4)(1,1,0)[12]                    : 466.677
##  ARIMA(0,1,5)(0,1,0)[12]                    : Inf
##  ARIMA(1,1,0)(0,1,0)[12]                    : 475.7162
##  ARIMA(1,1,0)(0,1,1)[12]                    : Inf
##  ARIMA(1,1,0)(1,1,0)[12]                    : 475.5616
##  ARIMA(1,1,0)(1,1,1)[12]                    : Inf
##  ARIMA(1,1,1)(0,1,0)[12]                    : 467.7553
##  ARIMA(1,1,1)(0,1,1)[12]                    : Inf
##  ARIMA(1,1,1)(1,1,0)[12]                    : Inf
##  ARIMA(1,1,1)(1,1,1)[12]                    : Inf
##  ARIMA(1,1,2)(0,1,0)[12]                    : 469.5689
##  ARIMA(1,1,2)(0,1,1)[12]                    : Inf
##  ARIMA(1,1,2)(1,1,0)[12]                    : 469.8107
##  ARIMA(1,1,2)(1,1,1)[12]                    : Inf
##  ARIMA(1,1,3)(0,1,0)[12]                    : 470.9884
##  ARIMA(1,1,3)(0,1,1)[12]                    : Inf
##  ARIMA(1,1,3)(1,1,0)[12]                    : 472.2383
##  ARIMA(1,1,4)(0,1,0)[12]                    : Inf
##  ARIMA(2,1,0)(0,1,0)[12]                    : 476.926
##  ARIMA(2,1,0)(0,1,1)[12]                    : Inf
##  ARIMA(2,1,0)(1,1,0)[12]                    : 476.5811
##  ARIMA(2,1,0)(1,1,1)[12]                    : Inf
##  ARIMA(2,1,1)(0,1,0)[12]                    : 470.298
##  ARIMA(2,1,1)(0,1,1)[12]                    : Inf
```

```
##  ARIMA(2,1,1)(1,1,0)[12]                    : 469.8879
##  ARIMA(2,1,1)(1,1,1)[12]                    : Inf
##  ARIMA(2,1,2)(0,1,0)[12]                    : 472.0187
##  ARIMA(2,1,2)(0,1,1)[12]                    : Inf
##  ARIMA(2,1,2)(1,1,0)[12]                    : 471.8468
##  ARIMA(2,1,3)(0,1,0)[12]                    : Inf
##  ARIMA(3,1,0)(0,1,0)[12]                    : 468.4805
##  ARIMA(3,1,0)(0,1,1)[12]                    : Inf
##  ARIMA(3,1,0)(1,1,0)[12]                    : 467.4776
##  ARIMA(3,1,0)(1,1,1)[12]                    : Inf
##  ARIMA(3,1,1)(0,1,0)[12]                    : 468.2049
##  ARIMA(3,1,1)(0,1,1)[12]                    : Inf
##  ARIMA(3,1,1)(1,1,0)[12]                    : 468.4294
##  ARIMA(3,1,2)(0,1,0)[12]                    : 470.7111
##  ARIMA(4,1,0)(0,1,0)[12]                    : 470.3022
##  ARIMA(4,1,0)(0,1,1)[12]                    : Inf
##  ARIMA(4,1,0)(1,1,0)[12]                    : 469.8215
##  ARIMA(4,1,1)(0,1,0)[12]                    : 470.7823
##  ARIMA(5,1,0)(0,1,0)[12]                    : 472.0753
##
##
##
##  Best model: ARIMA(0,1,1)(1,1,0)[12]
```

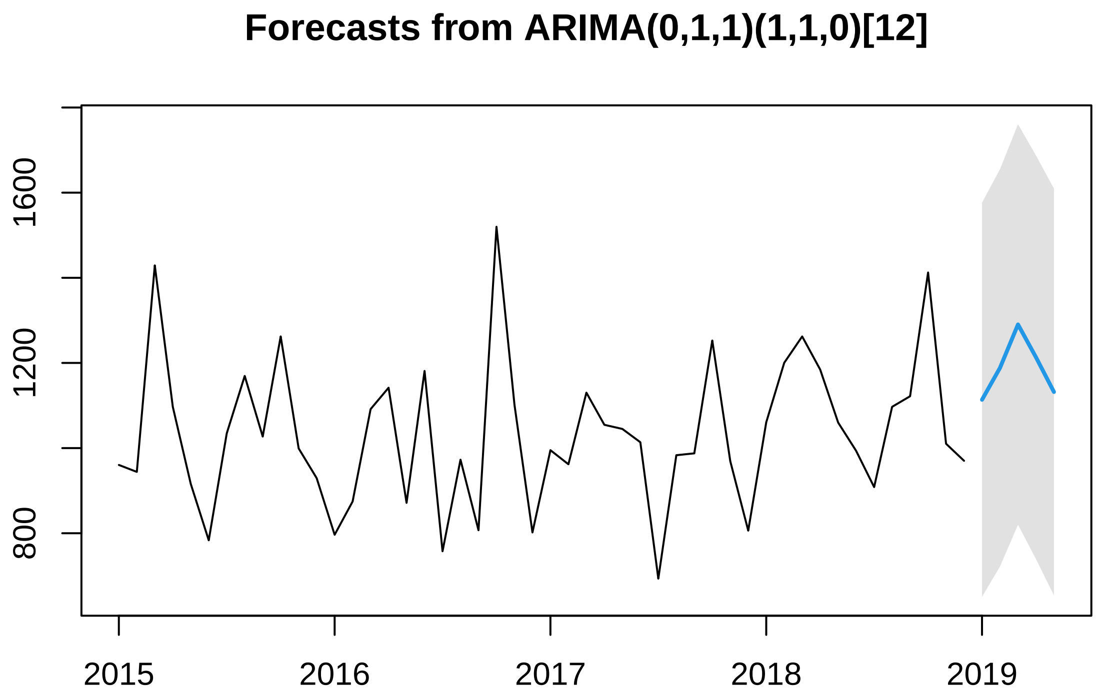
```
auto_fit
```

```
## Series: FD_train
## ARIMA(0,1,1)(1,1,0)[12]
##
## Coefficients:
##           ma1     sar1
##       -0.8729  -0.3746
## s.e.   0.0903   0.1942
##
## sigma^2 estimated as 27192:  log likelihood=-229.02
## AIC=464.03   AICc=464.8   BIC=468.7
```

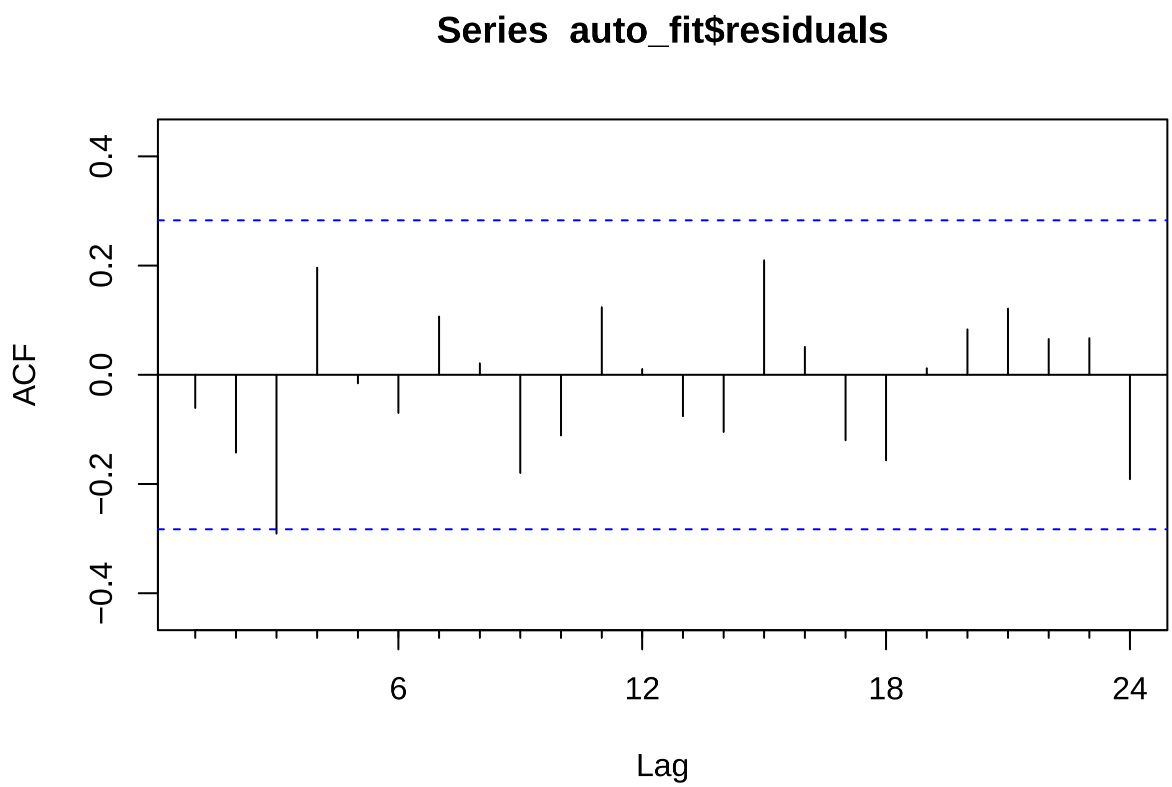
```
forecast(auto_fit,h=5,level=c(99.5))
```

```
##          Point Forecast  Lo 99.5  Hi 99.5
## Jan 2019       1113.566 650.6612 1576.470
## Feb 2019       1188.517 721.8870 1655.148
## Mar 2019       1290.320 819.9933 1760.647
## Apr 2019       1213.371 739.3761 1687.365
## May 2019       1131.839 654.2051 1609.473
```

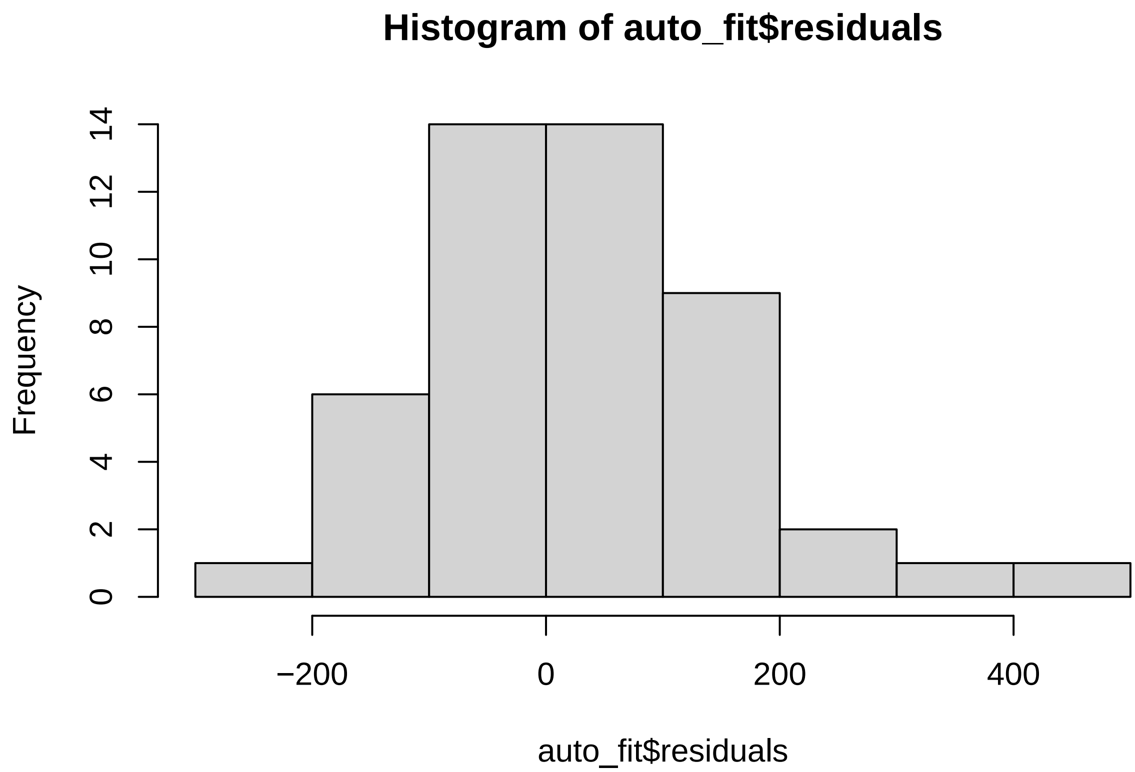
```
plot(forecast(auto_fit,h=5,level=c(99.5)))
```




```
Acf(auto_fit$residuals)
```

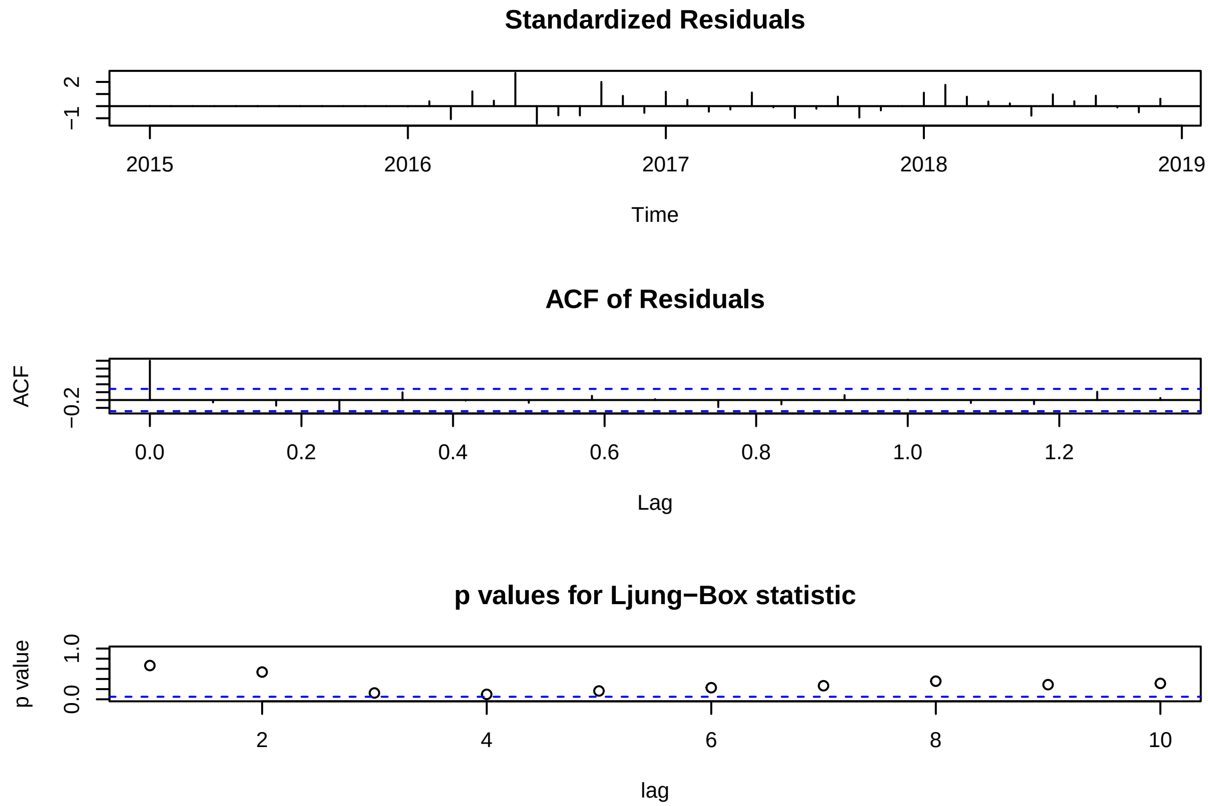



```
hist(auto_fit$residuals)
```



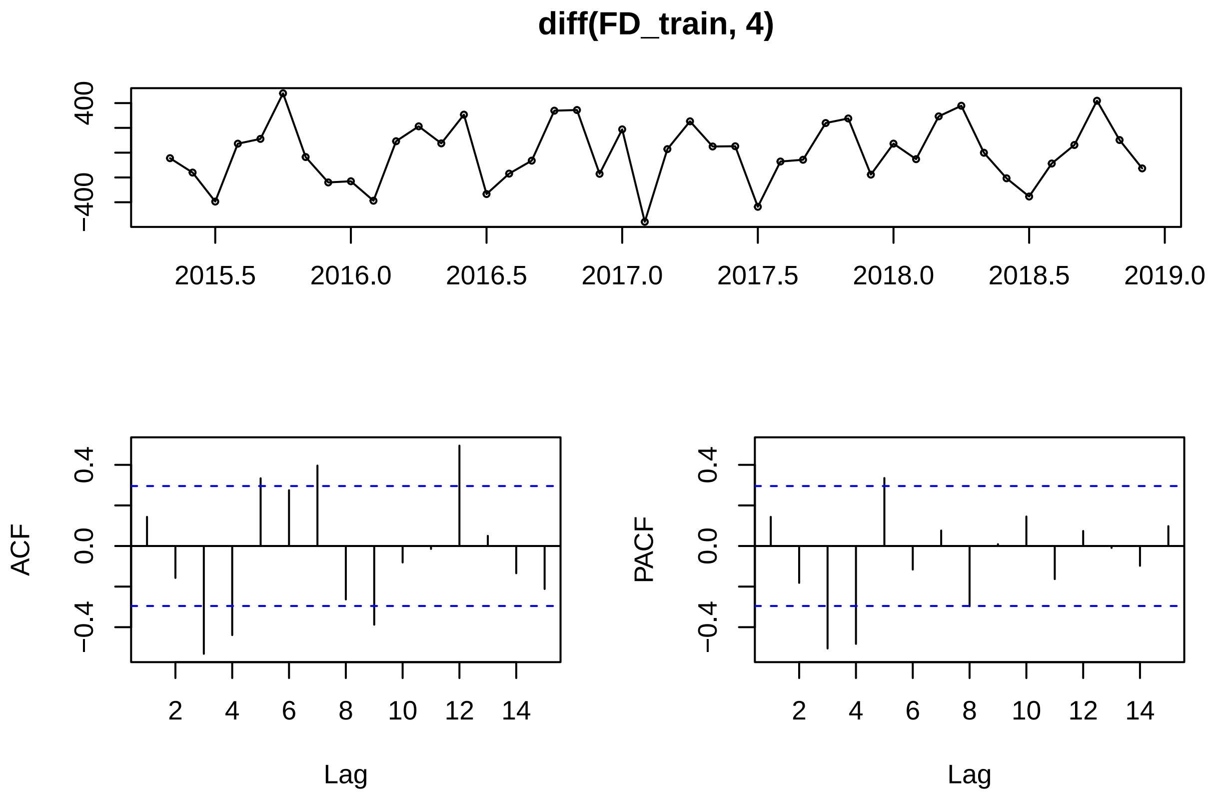
Appears to be skewed left

```
tsdiag(auto_fit)
```

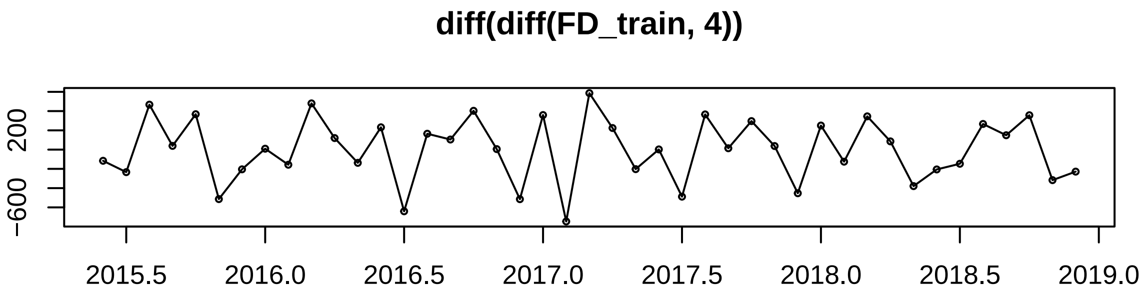


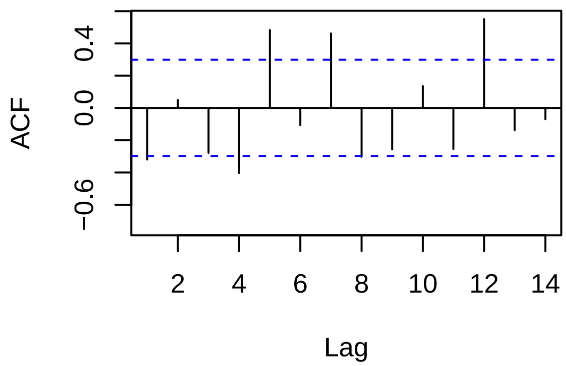
Seasonal ARIMA

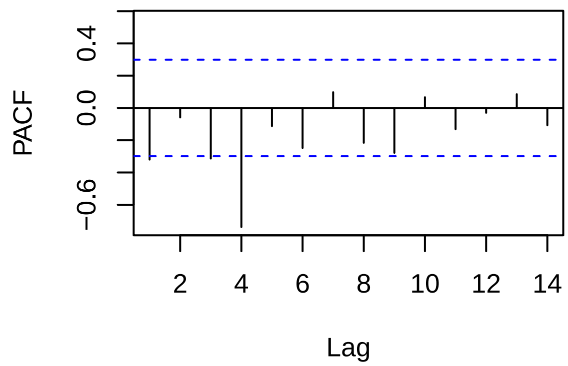
```
tsdisplay(diff(FD_train,4))
```




```
# It did not make it stationary. So lets take lag 1 diffence
tsdisplay(diff(diff(FD_train,4)))
```







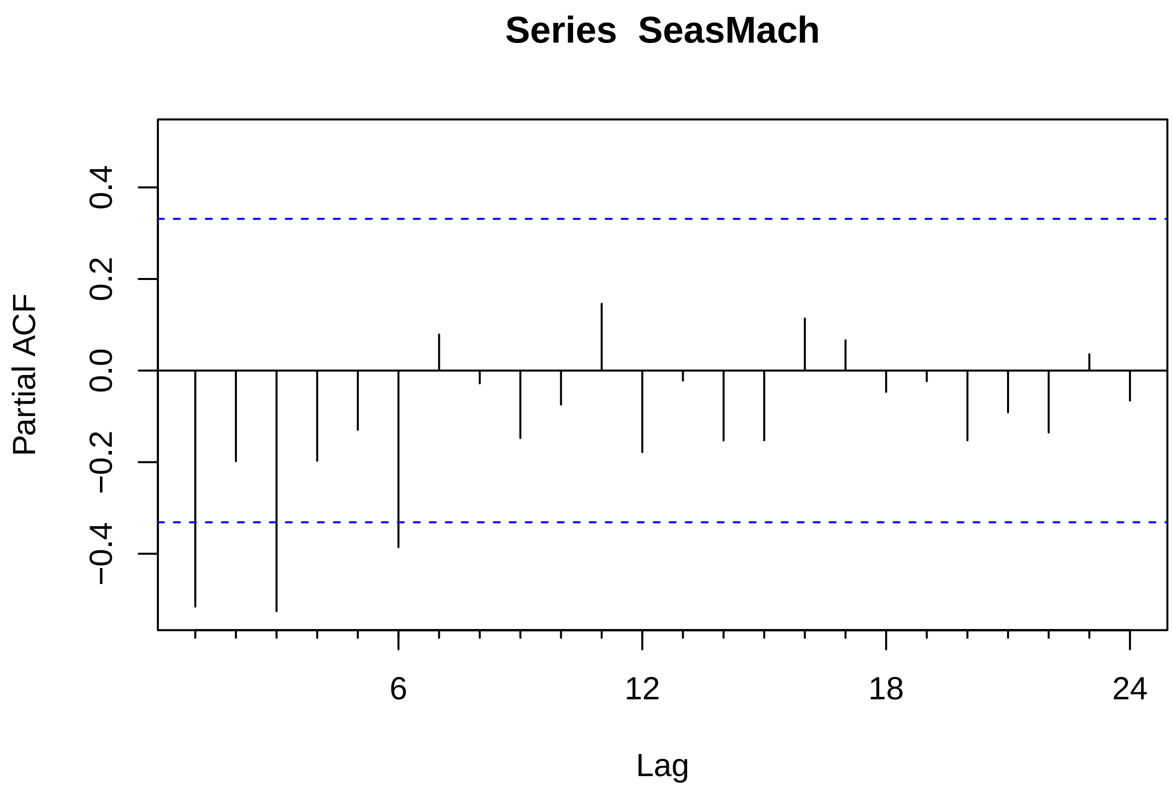

```
ndiffs(diff(FD_train,12))
```

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

Checking the model

```
SeasMach = diff(diff(FD_train,12))
```

```
Pacf(SeasMach)
```


Series SeasMach
Partial ACF
0.4
0.2
0.0
−0.2
−0.4
6
12
18
24
Lag