Data Visualization: Pre-Covid and Post-Covid

```
library(readxl)
```

```
## Warning: package 'readxl' was built under R version 4.0.5
```

```
Tng_Ctr_Hour <- read_excel("C:/Users/prach/Desktop/Rutgers/BF/Project/Tng_Ctr_Hour.xlsx")
View(Tng_Ctr_Hour)
```

```
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 4.0.5
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.0.5
```

```
library(TTR)
```

```
## Warning: package 'TTR' was built under R version 4.0.5
```

```
library(fpp)
```

```
## Warning: package 'fpp' was built under R version 4.0.5
```

```
## Loading required package: forecast
```

```
## Warning: package 'forecast' was built under R version 4.0.5
```

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

```
## Loading required package: fma
```

```
## Warning: package 'fma' was built under R version 4.0.5
```

```
## Loading required package: expsmooth
```

```
## Warning: package 'expsmooth' was built under R version 4.0.5
```

```
## Loading required package: lmtest
```

```
## Warning: package 'lmtest' was built under R version 4.0.5

## Loading required package: zoo

## Warning: package 'zoo' was built under R version 4.0.5

##
## Attaching package: 'zoo'

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

## Loading required package: tseries

## Warning: package 'tseries' was built under R version 4.0.5
```

```
library(fpp2)
```

```
## Warning: package 'fpp2' was built under R version 4.0.5

##
## Attaching package: 'fpp2'

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

```
library(fpp3)
```

```
## Warning: package 'fpp3' was built under R version 4.0.5

## -- Attaching packages --------------------------------------------- fpp3 0.4.0 --

## v tibble      3.1.4      v tsibble     1.0.1
## v dplyr       1.0.7      v tsibbledata 0.3.0
## v tidyr       1.1.4      v feasts      0.2.2
## v lubridate   1.7.10     v fable       0.3.1

## Warning: package 'tibble' was built under R version 4.0.5

## Warning: package 'dplyr' was built under R version 4.0.5

## Warning: package 'tidyr' was built under R version 4.0.5

## Warning: package 'lubridate' was built under R version 4.0.5

## Warning: package 'tsibble' was built under R version 4.0.5
```

```
## Warning: package 'tsibbledata' was built under R version 4.0.5

## Warning: package 'feasts' was built under R version 4.0.5

## Warning: package 'fabletools' was built under R version 4.0.5

## Warning: package 'fable' was built under R version 4.0.5

## -- Conflicts ------------------------------------------------ fpp3_conflicts --
## x dplyr::between()       masks data.table::between()
## x lubridate::date()      masks base::date()
## x dplyr::filter()        masks stats::filter()
## x dplyr::first()         masks data.table::first()
## x fabletools::forecast() masks forecast::forecast()
## x lubridate::hour()      masks data.table::hour()
## x tsibble::index()       masks zoo::index()
## x tsibble::intersect()   masks base::intersect()
## x tsibble::interval()    masks lubridate::interval()
## x lubridate::isoweek()   masks data.table::isoweek()
## x tsibble::key()         masks data.table::key()
## x dplyr::lag()           masks stats::lag()
## x dplyr::last()          masks data.table::last()
## x lubridate::mday()      masks data.table::mday()
## x lubridate::minute()    masks data.table::minute()
## x lubridate::month()     masks data.table::month()
## x lubridate::quarter()   masks data.table::quarter()
## x lubridate::second()    masks data.table::second()
## x tsibble::setdiff()     masks base::setdiff()
## x tsibble::union()       masks base::union()
## x lubridate::wday()      masks data.table::wday()
## x lubridate::week()      masks data.table::week()
## x lubridate::yday()      masks data.table::yday()
## x lubridate::year()      masks data.table::year()

##
## Attaching package: 'fpp3'

## The following object is masked from 'package:fpp2':
##
##     insurance

## The following object is masked from 'package:fpp':
##
##     insurance
```

```
library(stats)
library(dplyr)
library(ggfortify)
```

```
## Warning: package 'ggfortify' was built under R version 4.0.5
```

```
## 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
```

```
library(graphics)
```

```
setDT(Tng_Ctr_Hour)
#Tng_Ctr_Hour[,Year:=factor(Year)]
#TH <- select(Tng_Ctr_Hour, Year, Device_Hrs)
TH = Tng_Ctr_Hour[,c(1,4)]
summary(TH)
```

```
##      Year             Device_Hrs
##  Length:81          Min.   : 222.8
##  Class :character   1st Qu.: 899.0
##  Mode  :character   Median :1008.0
##                     Mean   : 990.1
##                     3rd Qu.:1101.7
##                     Max.   :1519.9
```

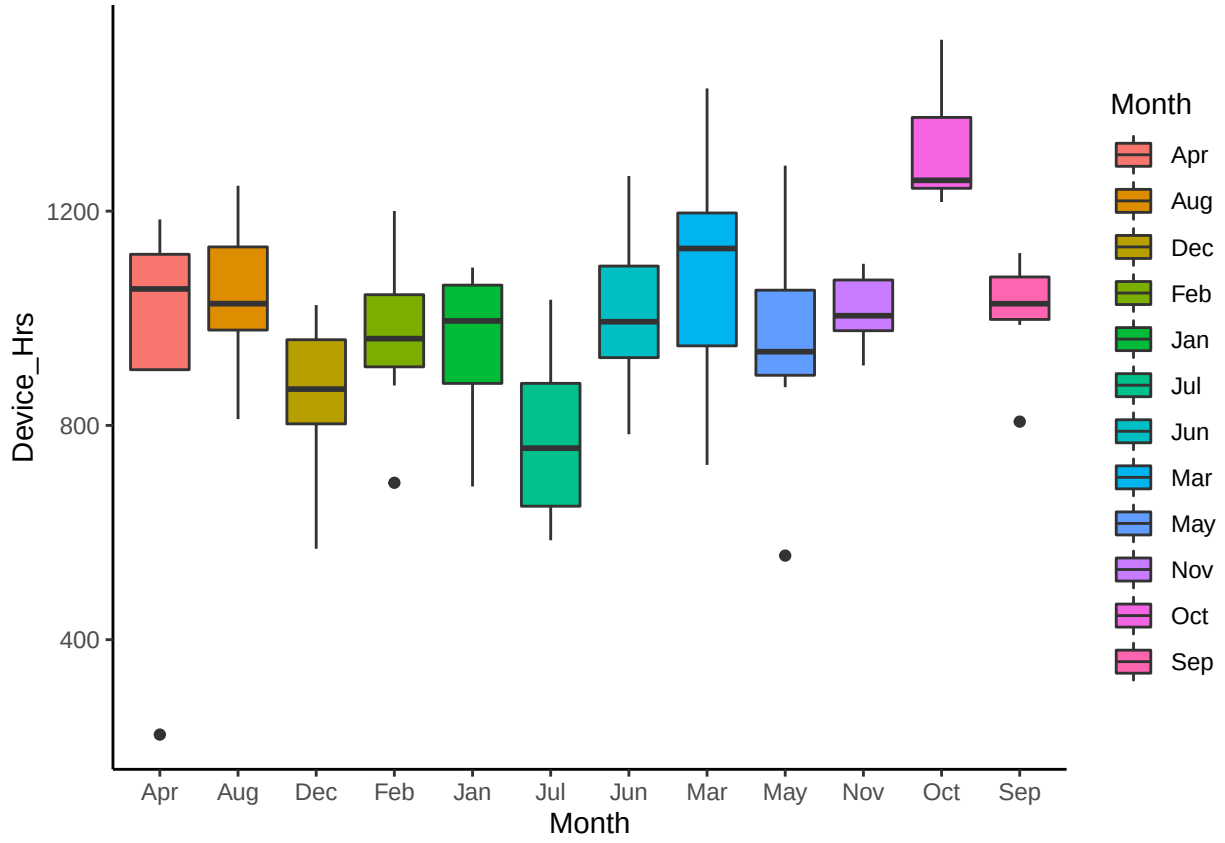
```
TH[, Count:=c(1:81)]
TH
```

```
##        Year Device_Hrs Count
##  1: 2015-01     960.42     1
##  2: 2015-02     944.08     2
##  3: 2015-03    1429.12     3
##  4: 2015-04    1097.00     4
##  5: 2015-05     915.85     5
##  6: 2015-06     783.45     6
##  7: 2015-07    1034.52     7
##  8: 2015-08    1169.50     8
##  9: 2015-09    1027.08     9
## 10: 2015-10    1262.32    10
## 11: 2015-11     999.25    11
## 12: 2015-12     929.42    12
## 13: 2016-01     796.42    13
## 14: 2016-02     874.55    14
## 15: 2016-03    1091.55    15
## 16: 2016-04    1141.84    16
## 17: 2016-05     871.36    17
```

```
## 18: 2016-06    1181.21    18
## 19: 2016-07     757.59    19
## 20: 2016-08     972.73    20
## 21: 2016-09     807.02    21
## 22: 2016-10    1519.92    22
## 23: 2016-11    1101.67    23
## 24: 2016-12     801.83    24
## 25: 2017-01     995.09    25
## 26: 2017-02     962.00    26
## 27: 2017-03    1130.24    27
## 28: 2017-04    1054.71    28
## 29: 2017-05    1044.95    29
## 30: 2017-06    1013.73    30
## 31: 2017-07     693.33    31
## 32: 2017-08     983.25    32
## 33: 2017-09     987.64    33
## 34: 2017-10    1252.69    34
## 35: 2017-11     969.31    35
## 36: 2017-12     806.10    36
## 37: 2018-01    1060.57    37
## 38: 2018-02    1200.25    38
## 39: 2018-03    1262.25    39
## 40: 2018-04    1184.45    40
## 41: 2018-05    1059.92    41
## 42: 2018-06     993.55    42
## 43: 2018-07     908.37    43
## 44: 2018-08    1096.93    44
## 45: 2018-09    1121.75    45
## 46: 2018-10    1412.47    46
## 47: 2018-11    1010.25    47
## 48: 2018-12     970.12    48
## 49: 2019-01    1063.13    49
## 50: 2019-02    1036.95    50
## 51: 2019-03    1130.87    51
## 52: 2019-04     903.97    52
## 53: 2019-05    1284.95    53
## 54: 2019-06    1265.56    54
## 55: 2019-07     848.64    55
## 56: 2019-08    1247.40    56
## 57: 2019-09    1106.84    57
## 58: 2019-10    1217.08    58
## 59: 2019-11    1091.84    59
## 60: 2019-12    1024.67    60
## 61: 2020-01    1094.62    61
## 62: 2020-02    1050.98    62
## 63: 2020-03     726.19    63
## 64: 2020-04     222.80    64
## 65: 2020-05     556.92    65
## 66: 2020-06     899.00    66
## 67: 2020-07     585.58    67
## 68: 2020-08     811.74    68
## 69: 2020-09    1047.41    69
## 70: 2020-10    1239.26    70
## 71: 2020-11     911.93    71
```

```
## 72: 2020-12     569.75    72
## 73: 2021-01     685.91    73
## 74: 2021-02     692.88    74
## 75: 2021-03     805.42    75
## 76: 2021-04     904.00    76
## 77: 2021-05     937.62    77
## 78: 2021-06     954.00    78
## 79: 2021-07     605.00    79
## 80: 2021-08    1027.23    80
## 81: 2021-09    1008.00    81
##        Year Device_Hrs Count
```

```
ggplot(Tng_Ctr_Hour, aes(x=Month, y=Device_Hrs)) + geom_boxplot(aes(fill=Month)) + theme_classic()
```



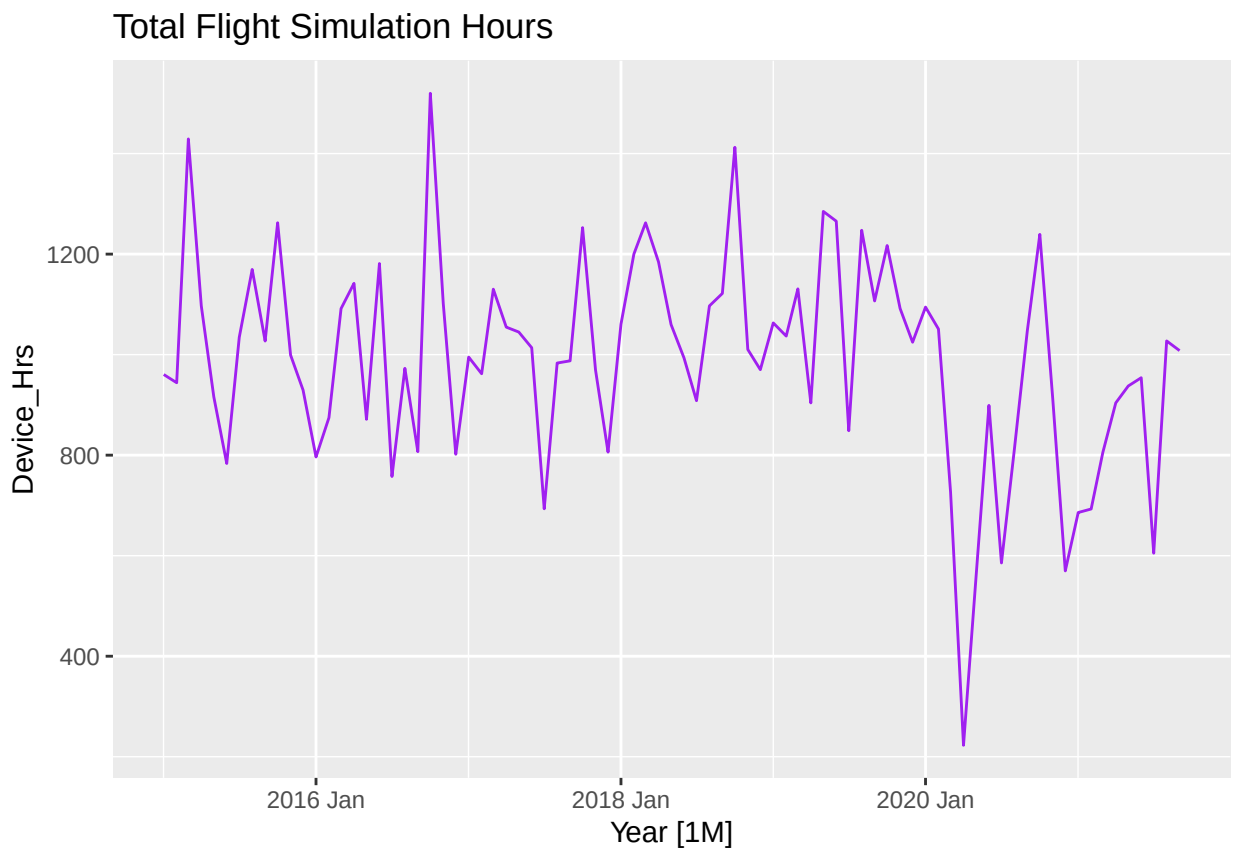

```
TC_ts <- TH %>%
  mutate(Year = yearmonth(Year)) %>%
  as_tsibble(index =  Year)
TC_ts
```

```
## # A tsibble: 81 x 3 [1M]
##        Year Device_Hrs Count
##       <mth>      <dbl> <int>
##  1 2015 Jan       960.     1
##  2 2015 Feb       944.     2
##  3 2015 Mar      1429.     3
```

```
##  4 2015 Apr      1097      4
##  5 2015 May       916.     5
##  6 2015 Jun       783.     6
##  7 2015 Jul      1035.     7
##  8 2015 Aug      1170.     8
##  9 2015 Sep      1027.     9
## 10 2015 Oct      1262.    10
## # ... with 71 more rows
```

Time Series Decomposition

```
TC_ts %>%
  autoplot(Device_Hrs, color = 'purple') +
  labs(title = "Total Flight Simulation Hours")
```



Quantiles of the time series:

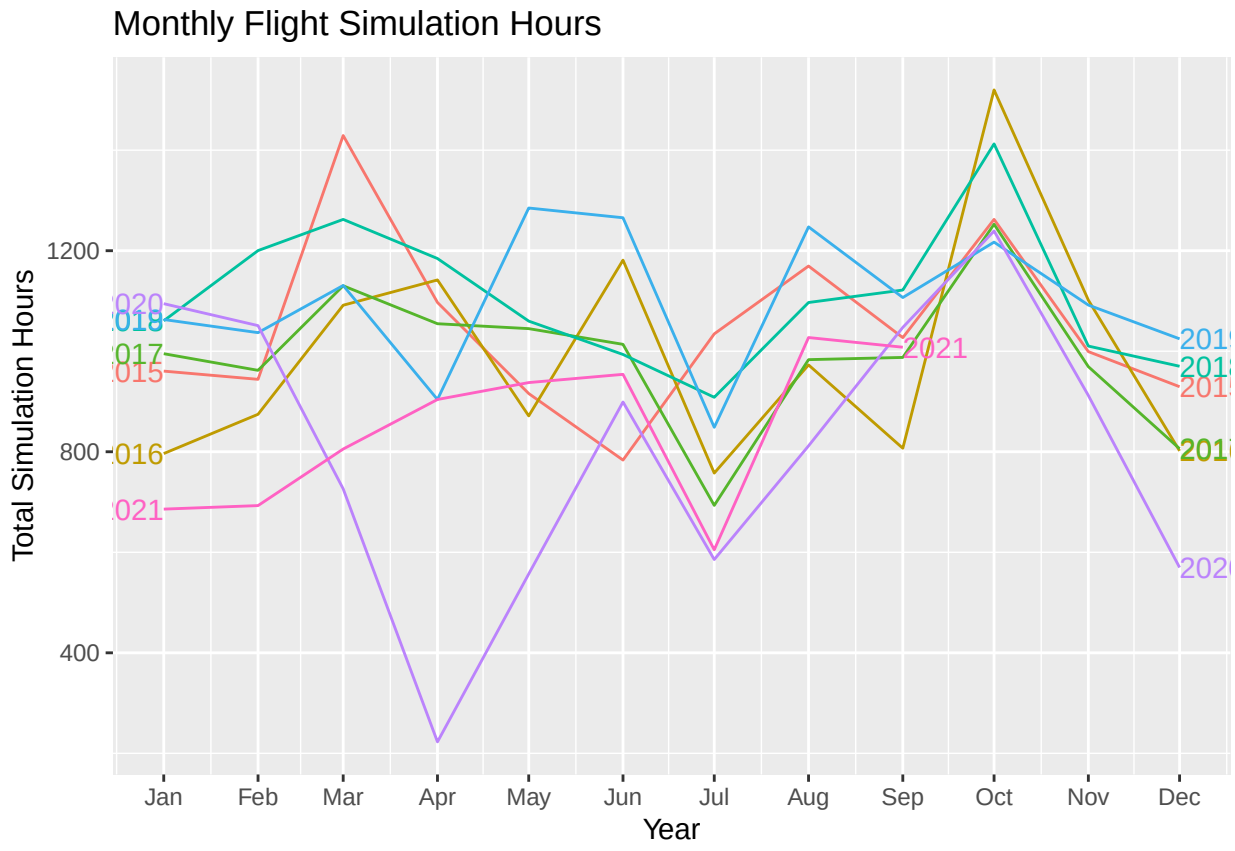
```
TC_ts %>% features(Device_Hrs, quantile)
```

```
## # A tibble: 1 x 5
##    '0%' '25%' '50%' '75%' '100%'
```

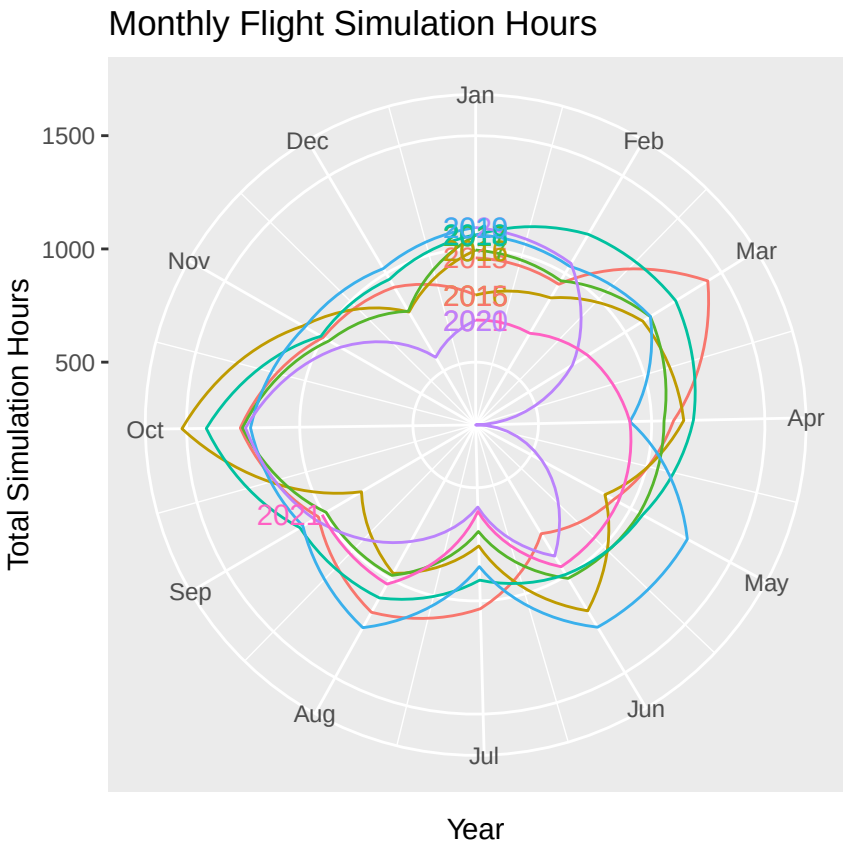
```
##   <dbl> <dbl> <dbl> <dbl>  <dbl>
## 1  223.   899  1008 1102.  1520.
```

Seasonal Plot:

```
TC_ts %>%
  gg_season(Device_Hrs, labels = "both") +
  labs(y = "Total Simulation Hours",
       title = "Monthly Flight Simulation Hours")
```

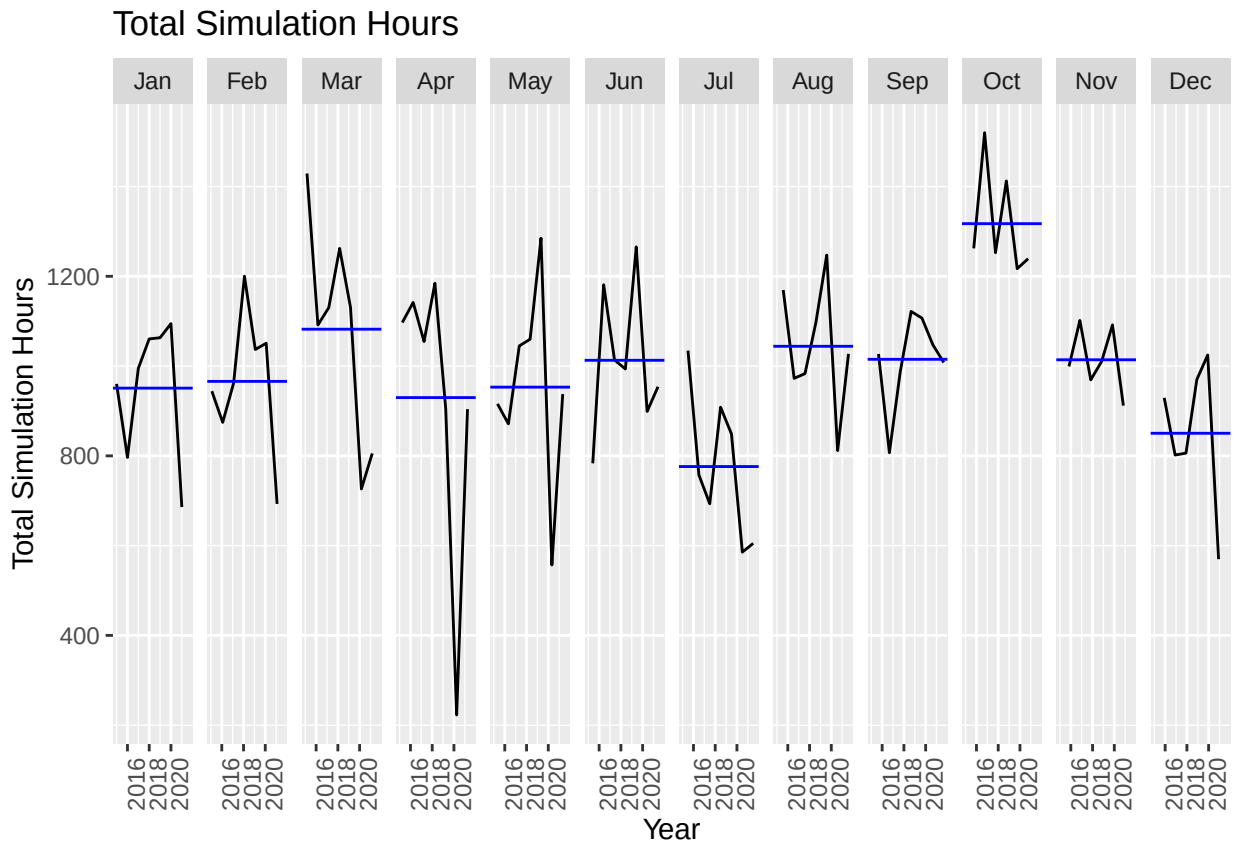



```
TC_ts %>%
  gg_season(Device_Hrs,polar = TRUE, labels = "both") +
  labs(y = "Total Simulation Hours",
       title = "Monthly Flight Simulation Hours")
```

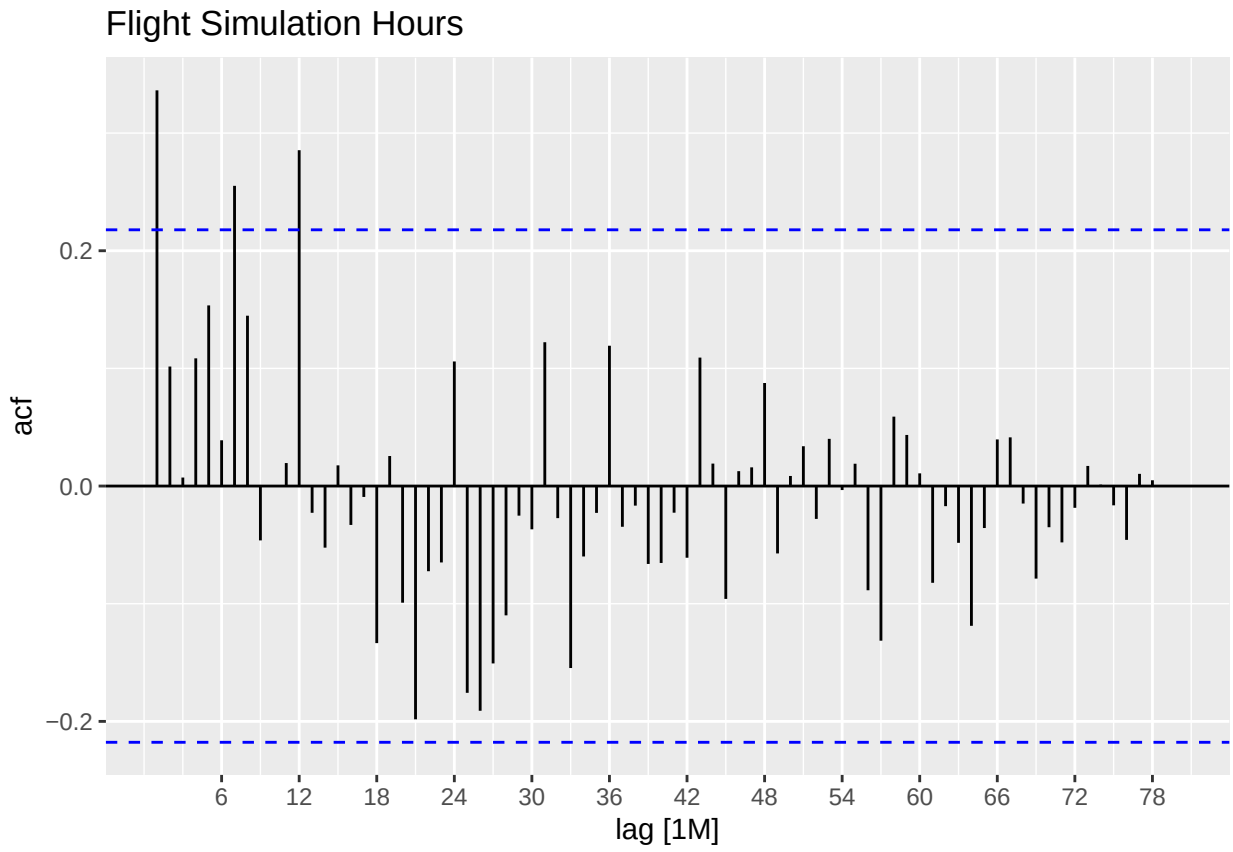


Seasonal Subseries Plot:

```
TC_ts %>%
  gg_subseries(Device_Hrs) +
  labs(
    y = "Total Simulation Hours",
    title = "Total Simulation Hours"
  )
```

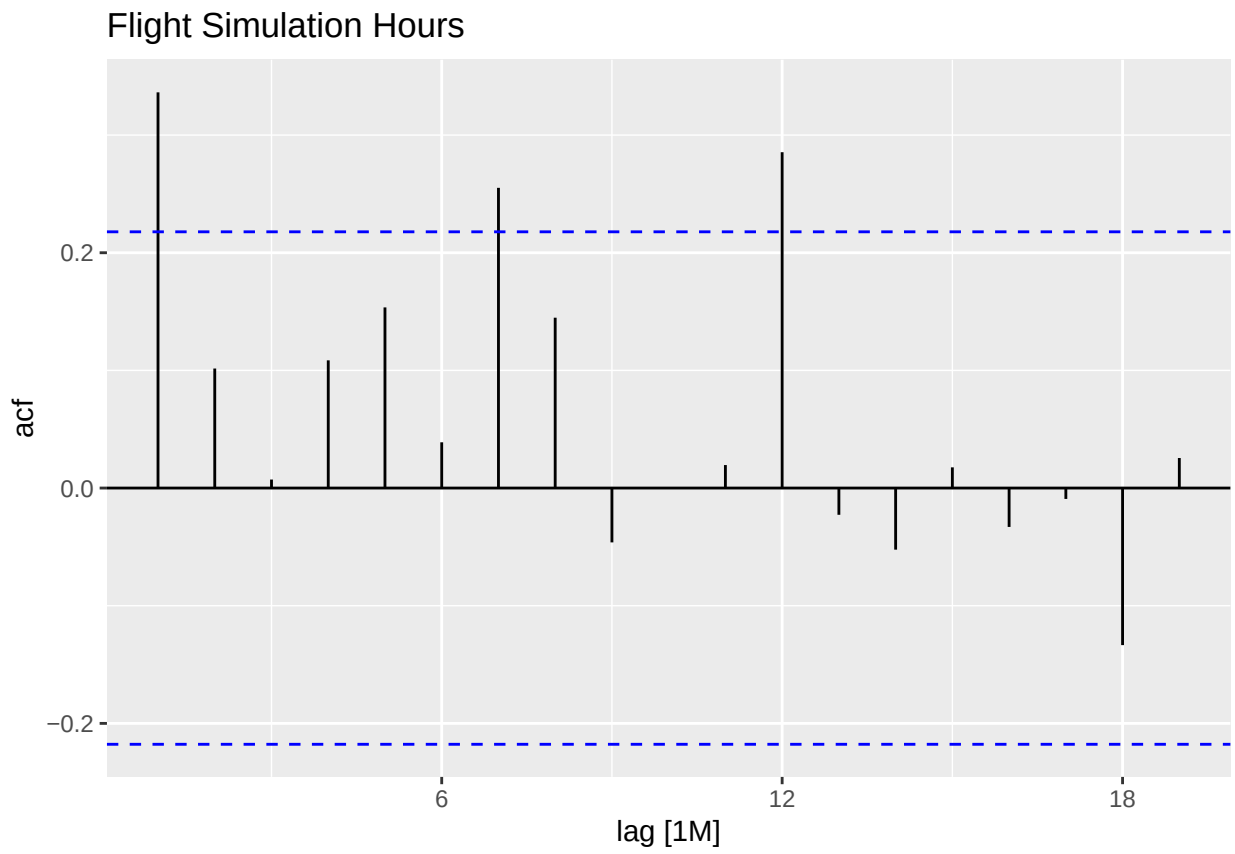



```
TC_ts %>%
  ACF(Device_Hrs, lag_max = 82) %>%
  autoplot() + labs(title = "Flight Simulation Hours")
```



ACF Plots:

```
TC_ts %>%
  ACF(Device_Hrs) %>%
  autoplot() + labs(title = "Flight Simulation Hours")
```




```
TC_tstrain <- TC_ts %>%
  filter(Count == c(1:60))
```

Training and Test set

```
## Warning in Count == c(1:60): longer object length is not a multiple of shorter
## object length
```

```
TC_tstrain
```

```
## # A tsibble: 60 x 3 [1M]
##        Year Device_Hrs Count
##       <mth>      <dbl> <int>
##  1 2015 Jan       960.     1
##  2 2015 Feb       944.     2
##  3 2015 Mar      1429.     3
##  4 2015 Apr      1097      4
##  5 2015 May       916.     5
##  6 2015 Jun       783.     6
##  7 2015 Jul      1035.     7
##  8 2015 Aug      1170.     8
##  9 2015 Sep      1027.     9
```

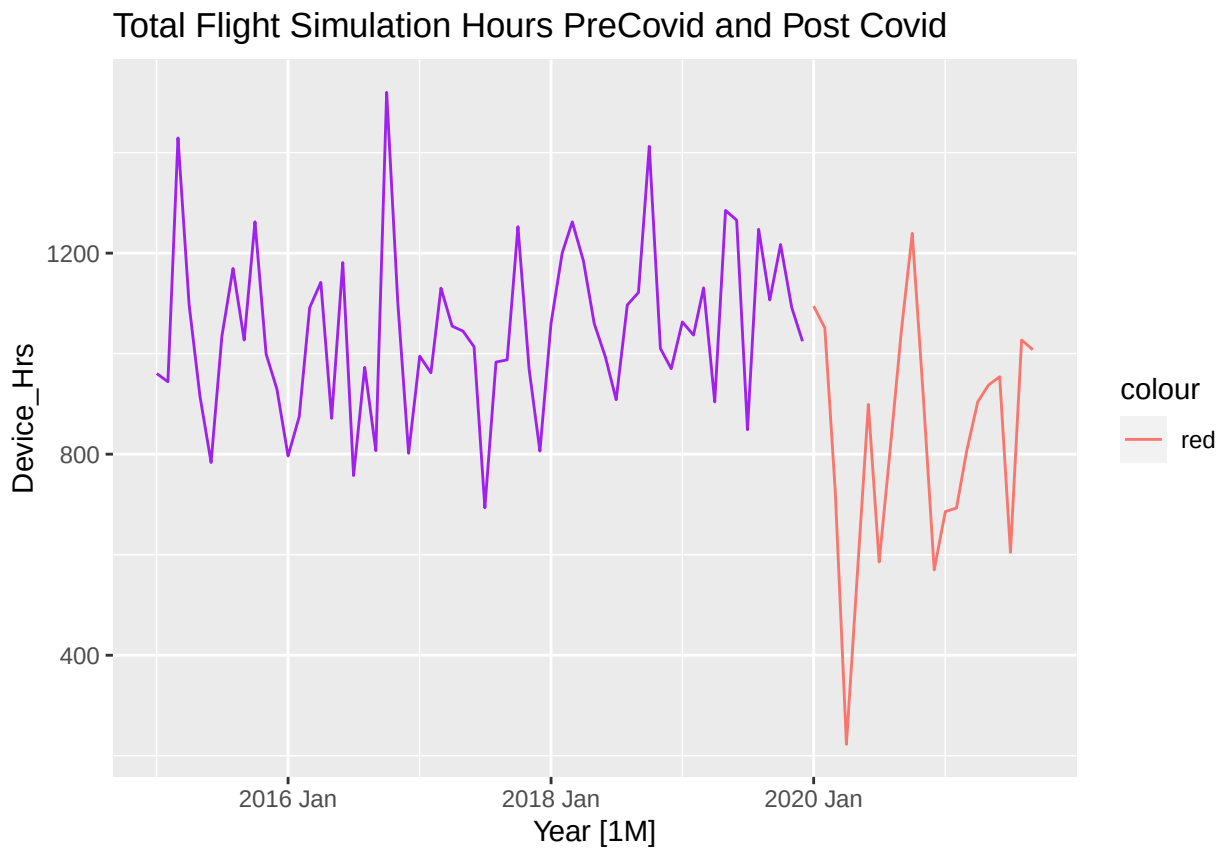
```
## 10 2015 Oct      1262.    10
## # ... with 50 more rows
```

```
TC_tstest <- TC_ts %>%
  filter(Count > 60)
TC_tstest
```

```
## # A tsibble: 21 x 3 [1M]
##        Year Device_Hrs Count
##       <mth>      <dbl> <int>
##  1 2020 Jan      1095.    61
##  2 2020 Feb      1051.    62
##  3 2020 Mar       726.    63
##  4 2020 Apr       223.    64
##  5 2020 May       557.    65
##  6 2020 Jun       899     66
##  7 2020 Jul       586.    67
##  8 2020 Aug       812.    68
##  9 2020 Sep      1047.    69
## 10 2020 Oct      1239.    70
## # ... with 11 more rows
```

```
TD <- TC_tstest$Device_Hrs
```

```
TC_tstrain %>%
  autoplot(Device_Hrs, color = 'purple') +
  geom_line(data = TC_tstest, aes(x = Year, y=Device_Hrs, color = 'red')) +
  labs(title = "Total Flight Simulation Hours PreCovid and Post Covid")
```



Forecast with ETS:

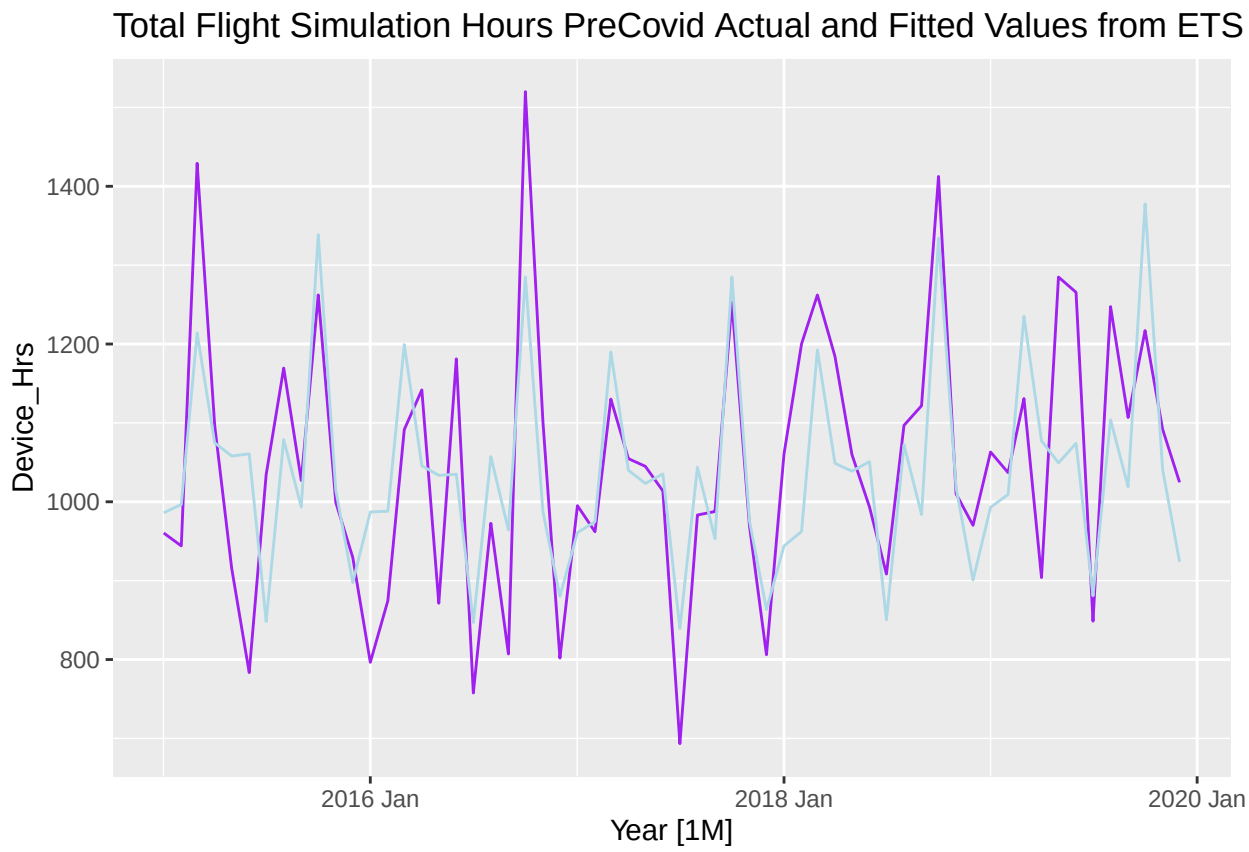
```
fitETS <- TC_tstrain %>%
  model(ETS(Device_Hrs))
report(fitETS)
```

```
## Series: Device_Hrs
## Model: ETS(M,N,M)
##   Smoothing parameters:
##     alpha = 0.05755771
##     gamma = 0.000101081
##
##   Initial states:
##      l[0]      s[0]    s[-1]    s[-2]     s[-3]    s[-4]    s[-5]   s[-6]
##  1047.432 0.8577446 0.968935 1.273859 0.9470808 1.033602 0.822901 1.01371
##     s[-7]    s[-8]    s[-9]   s[-10]    s[-11]
##  1.003268 1.020398 1.164308 0.952857 0.9413351
##
##   sigma^2:  0.0162
##
##      AIC     AICc      BIC
## 845.0223 855.9314 876.4374
```

Check the fit:

```
autoplot(TC_tstrain, color = 'purple') +
    geom_line(data = augment(fitETS), aes(x = Year,  y = .fitted), color = 'lightblue') +
  labs(title = "Total Flight Simulation Hours PreCovid Actual and Fitted Values from ETS")
```

```
## Plot variable not specified, automatically selected `.vars = Device_Hrs`
```

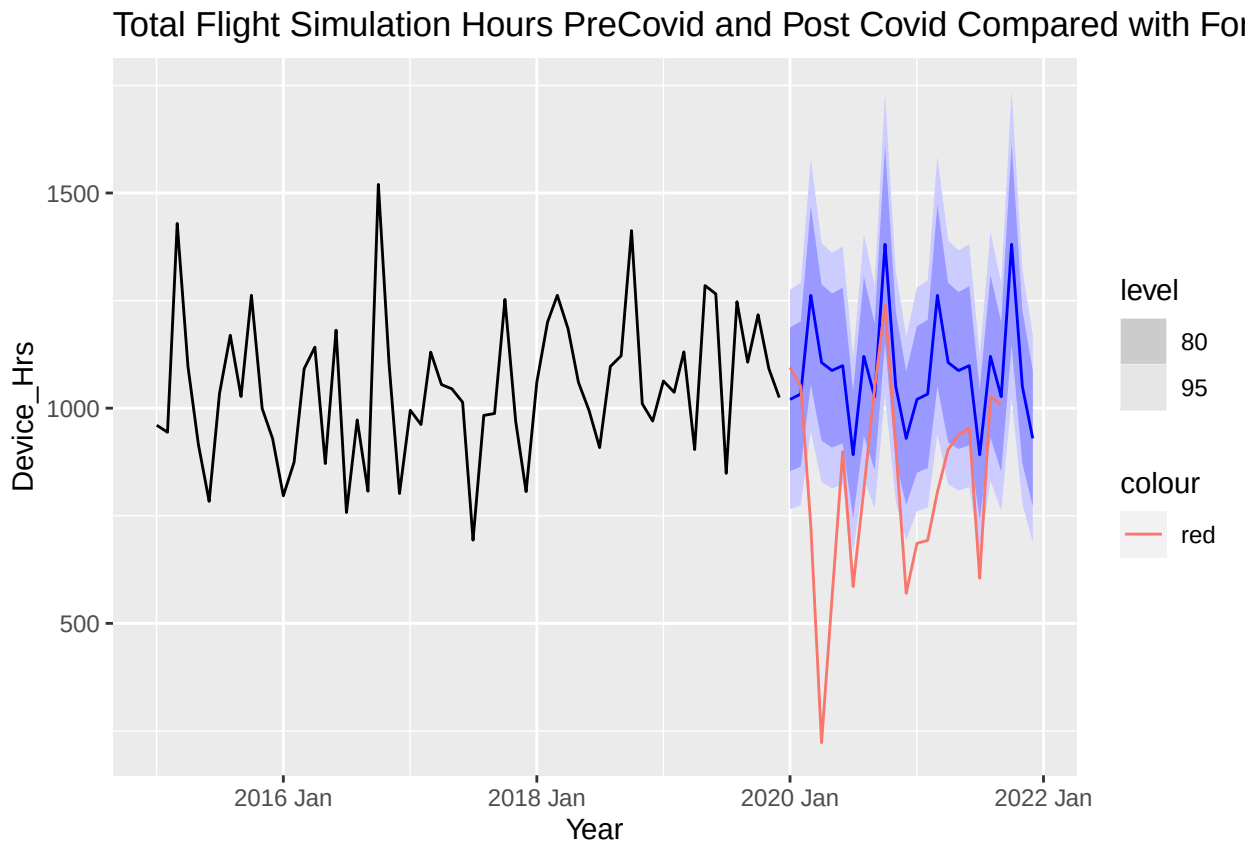



```
ETS_fcast <- forecast(fitETS, h = 24, level = c(80,95))
ETS_fcast
```

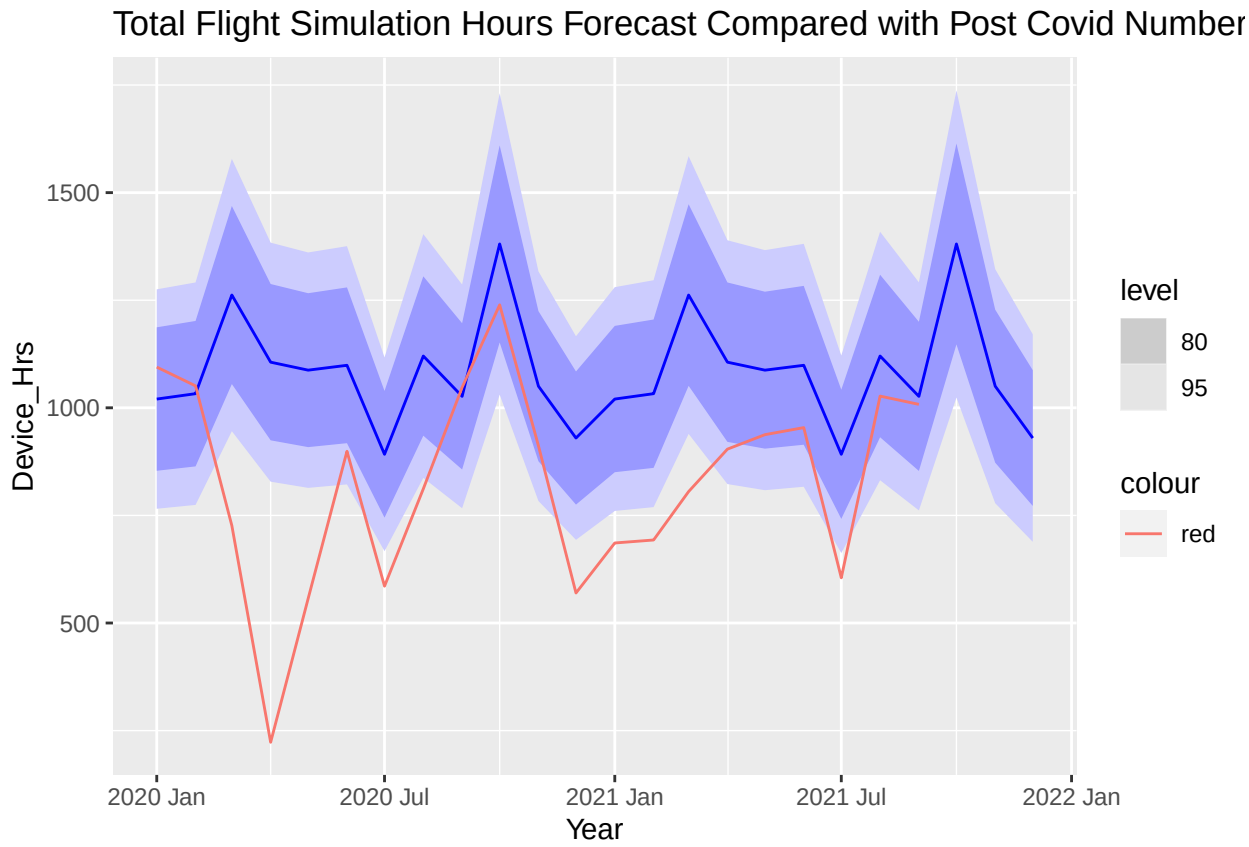
```
## # A fable: 24 x 4 [1M]
## # Key:     .model [1]
##    .model              Year     Device_Hrs .mean
##    <chr>              <mth>         <dist> <dbl>
##  1 ETS(Device_Hrs) 2020 Jan N(1020, 16910) 1020.
##  2 ETS(Device_Hrs) 2020 Feb N(1033, 17385) 1033.
##  3 ETS(Device_Hrs) 2020 Mar N(1262, 26044) 1262.
##  4 ETS(Device_Hrs) 2020 Apr N(1106, 20071) 1106.
##  5 ETS(Device_Hrs) 2020 May N(1087, 19467) 1087.
##  6 ETS(Device_Hrs) 2020 Jun N(1099, 19940) 1099.
##  7 ETS(Device_Hrs) 2020 Jul  N(892, 13184)  892.
##  8 ETS(Device_Hrs) 2020 Aug N(1120, 20868) 1120.
```

```
##  9 ETS(Device_Hrs) 2020 Sep N(1027, 17579) 1027.
## 10 ETS(Device_Hrs) 2020 Oct N(1381, 31906) 1381.
## # ... with 14 more rows
```

```
ETS_fcast %>%
  autoplot(TC_tstrain) +
  geom_line(data = TC_tstest, aes(x = Year, y=Device_Hrs, color = 'red')) +
  labs(title = "Total Flight Simulation Hours PreCovid and Post Covid Compared with Forecast from PreCo
```



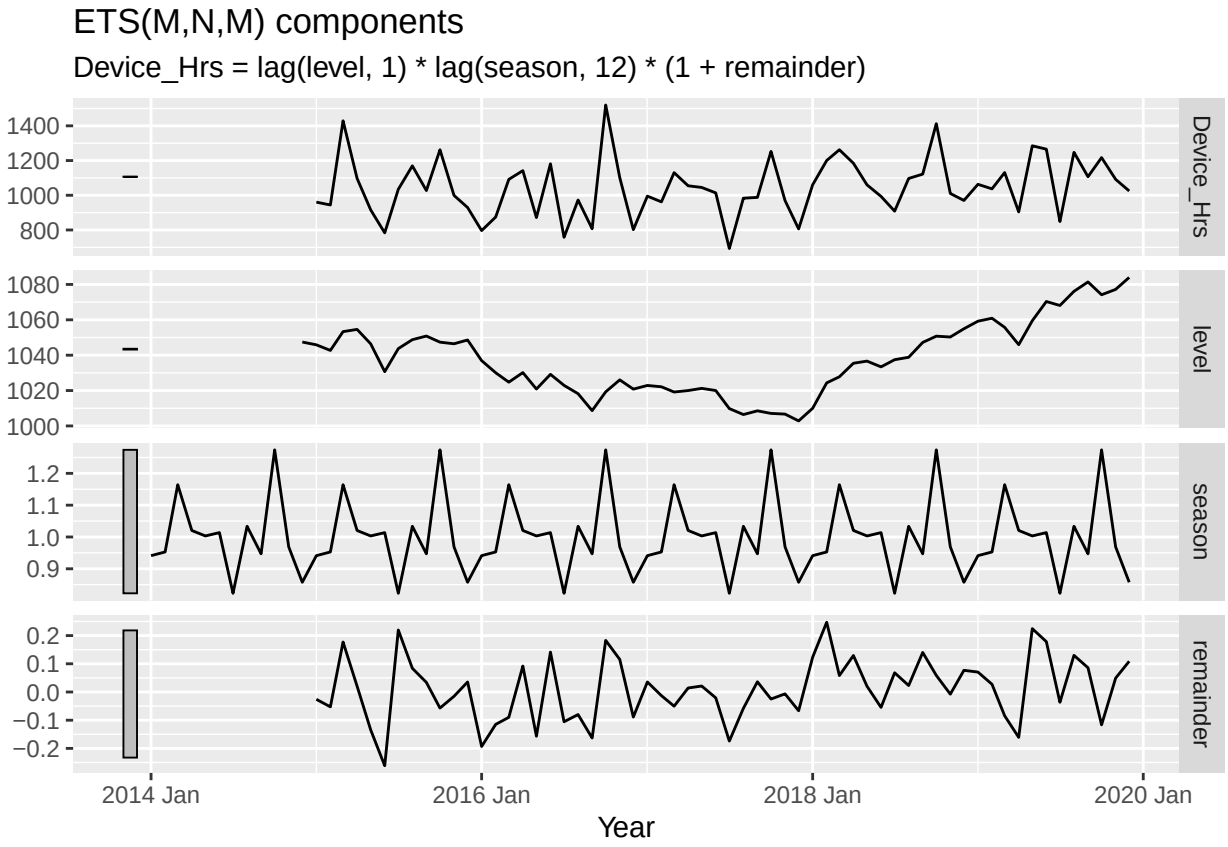

```
autoplot(ETS_fcast) +
  geom_line(data = TC_tstest, aes(x = Year, y=Device_Hrs, color = 'red')) +
  labs(title = "Total Flight Simulation Hours Forecast Compared with Post Covid Numbers")
```



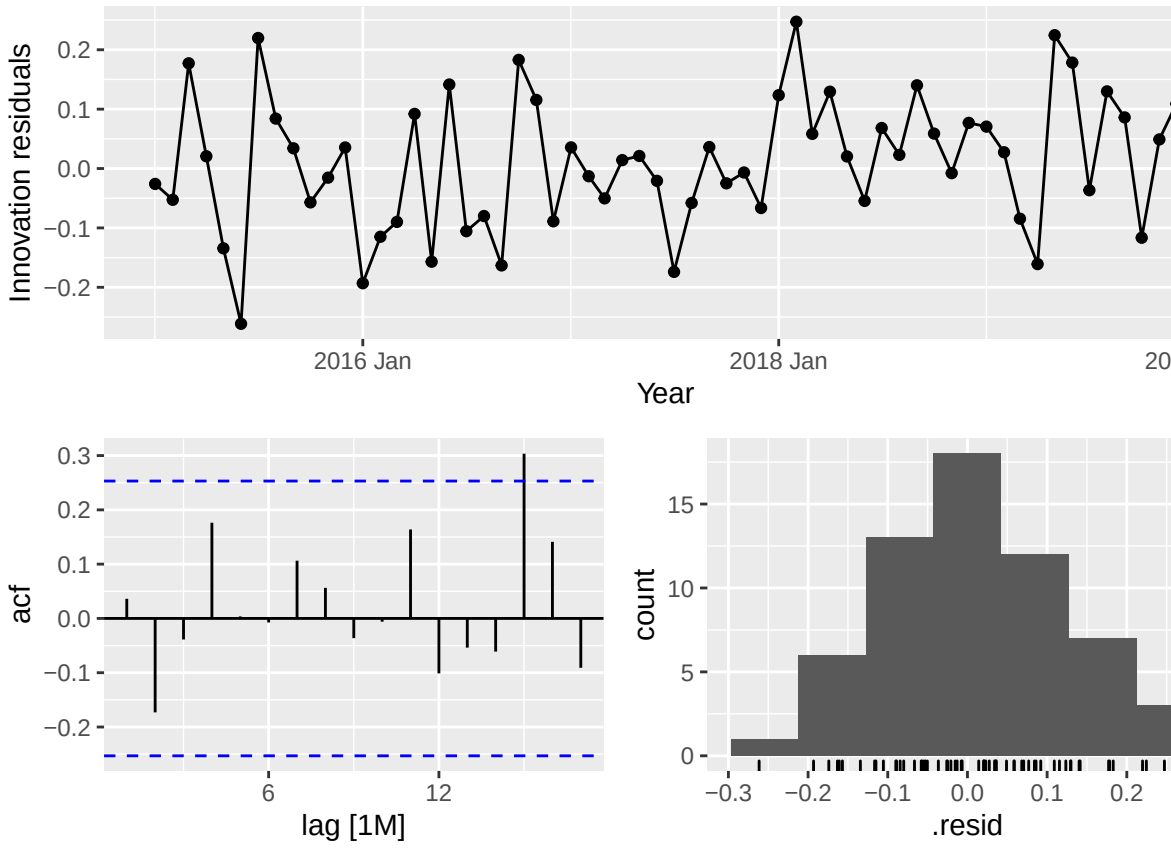
A look at the components of ETS:

```
components(fitETS) %>%
  autoplot() +
  labs(title = "ETS(M,N,M) components")
```

```
## Warning: Removed 12 row(s) containing missing values (geom_path).
```

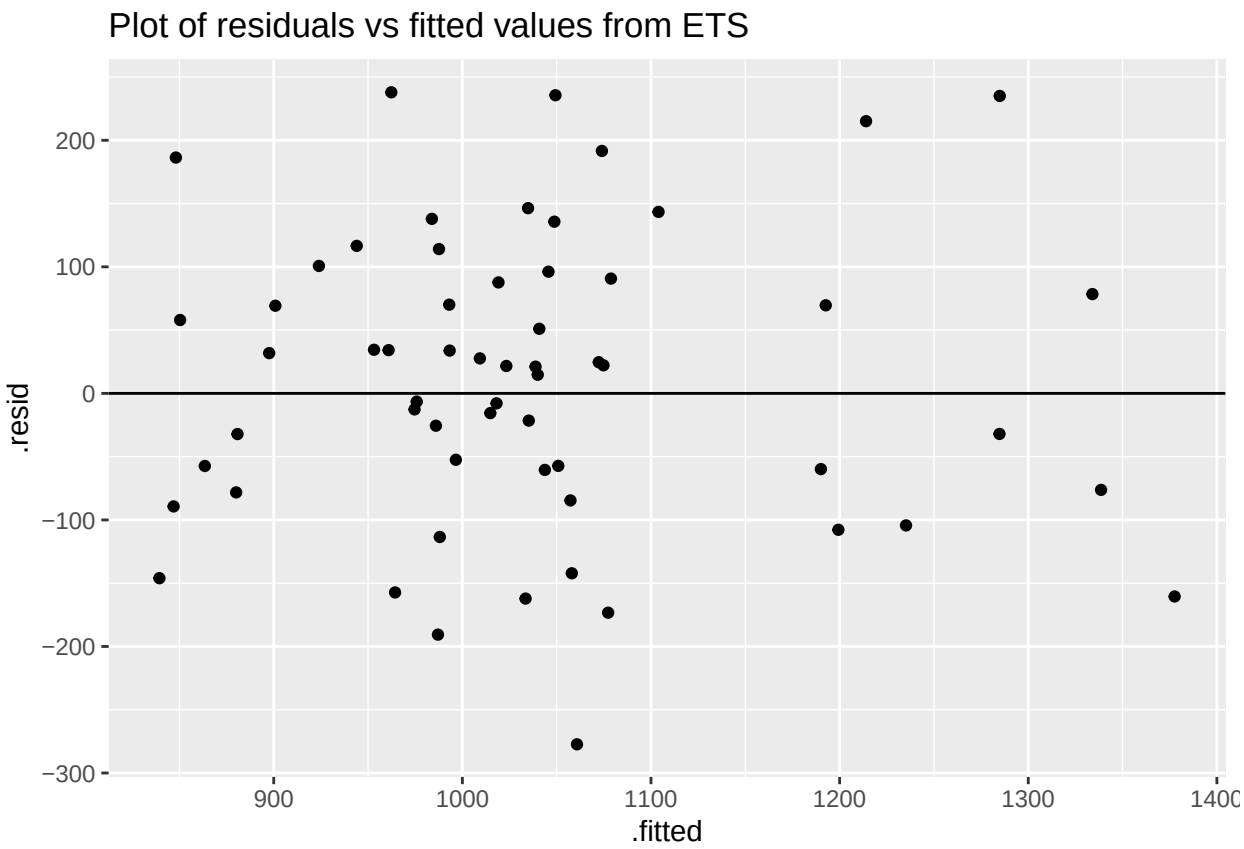



```
fitETS %>% gg_tsresiduals()
```

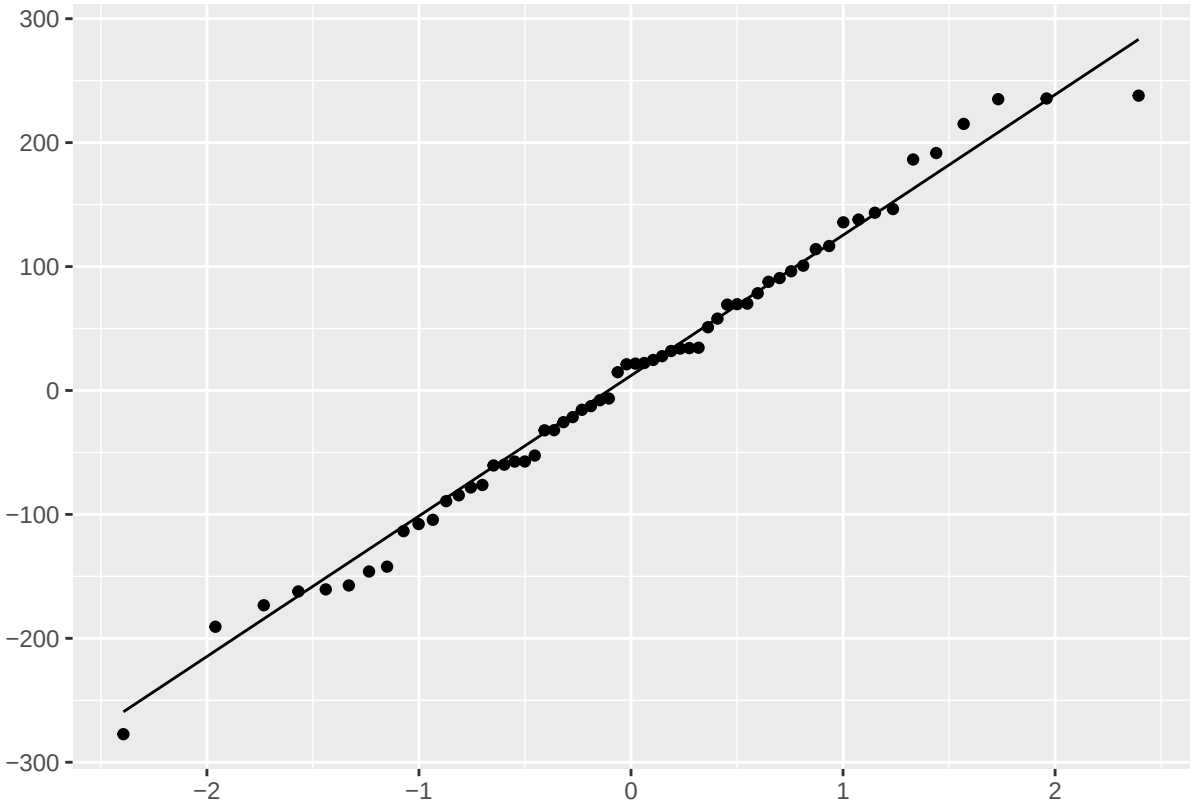


Check of residuals

```
ggplot(data = augment(fitETS), aes(x = .fitted, y = .resid)) + geom_point() + geom_hline(yintercept = 0)
  labs(title = 'Plot of residuals vs fitted values from ETS')
```




```
afitETS <- augment(fitETS)
qplot(sample = .resid, data = afitETS, geom = 'qq') + geom_qq_line()
```




```
accuracy(fitETS)
```

```
## # A tibble: 1 x 10
##   .model          .type       ME  RMSE   MAE    MPE  MAPE  MASE RMSSE   ACF1
##   <chr>           <chr>    <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl>  <dbl>
## 1 ETS(Device_Hrs) Training  10.5  116.   94.0 -0.225  9.11 0.672 0.690 0.0441
```

Forecast with decompositon:

```
TC_tstrain1 <- ts(TH$Device_Hrs, frequency = 12, start = c(2015,1), end = c(2019,12))
```

```
summary(TH)
```

```
##       Year             Device_Hrs         Count
##  Length:81          Min.   : 222.8   Min.   : 1
##  Class :character   1st Qu.: 899.0   1st Qu.:21
##  Mode  :character   Median :1008.0   Median :41
##                     Mean   : 990.1   Mean   :41
##                     3rd Qu.:1101.7   3rd Qu.:61
##                     Max.   :1519.9   Max.   :81
```

```
TH1 <- TH[Count > 60, ]
TH1
```

```
##        Year Device_Hrs Count
##  1: 2020-01    1094.62    61
##  2: 2020-02    1050.98    62
##  3: 2020-03     726.19    63
##  4: 2020-04     222.80    64
##  5: 2020-05     556.92    65
##  6: 2020-06     899.00    66
##  7: 2020-07     585.58    67
##  8: 2020-08     811.74    68
##  9: 2020-09    1047.41    69
## 10: 2020-10    1239.26    70
## 11: 2020-11     911.93    71
## 12: 2020-12     569.75    72
## 13: 2021-01     685.91    73
## 14: 2021-02     692.88    74
## 15: 2021-03     805.42    75
## 16: 2021-04     904.00    76
## 17: 2021-05     937.62    77
## 18: 2021-06     954.00    78
## 19: 2021-07     605.00    79
## 20: 2021-08    1027.23    80
## 21: 2021-09    1008.00    81
##        Year Device_Hrs Count
```

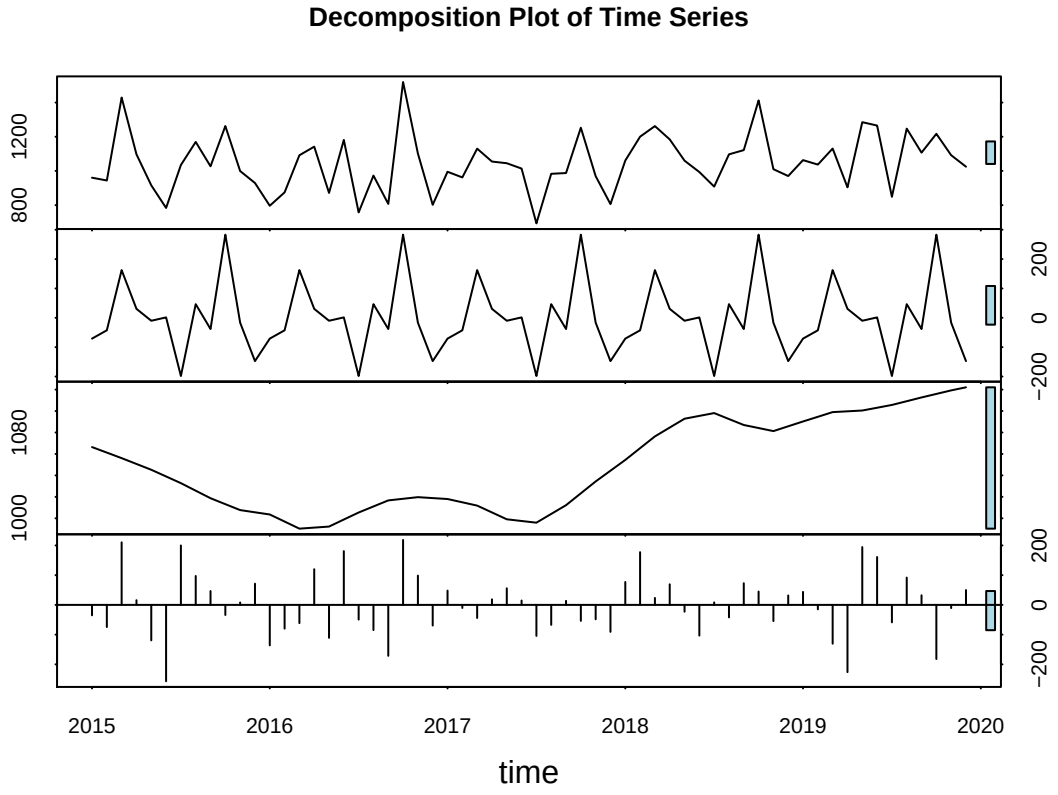
```
TC_tstest1 <- ts(TH1$Device_Hrs, frequency = 12, start = c(2020,1))
```

```
stl_decomp <- stl(TC_tstrain1,s.window = 'periodic')
stl_decomp
```

```
##  Call:
##  stl(x = TC_tstrain1, s.window = "periodic")
##
## Components
##             seasonal     trend   remainder
## Jan 2015  -70.915777 1066.4205  -35.084739
## Feb 2015  -42.611463 1061.2528  -74.561376
## Mar 2015  162.492749 1056.0852  210.542088
## Apr 2015   30.504883 1050.6743   15.820826
## May 2015  -10.059213 1045.2634 -119.354208
## Jun 2015    1.393934 1039.0208 -256.964697
## Jul 2015 -198.257090 1032.7781  199.998984
## Aug 2015   46.454532 1025.7504   97.295063
## Sep 2015  -38.201779 1018.7227   46.559073
## Oct 2015  283.244350 1013.1913  -34.115634
## Nov 2015  -16.571712 1007.6599    8.161849
## Dec 2015 -147.473465 1005.6070   71.286507
## Jan 2016  -70.915777 1003.5541 -136.218277
## Feb 2016  -42.611463  996.9635  -79.802043
## Mar 2016  162.492749  990.3730  -61.315707
```

```
## Apr 2016   30.504883  991.3423  119.992841
## May 2016  -10.059213  992.3116 -110.892381
## Jun 2016    1.393934  998.8536  180.962509
## Jul 2016 -198.257090 1005.3955  -49.548430
## Aug 2016   46.454532 1011.0417  -84.766246
## Sep 2016  -38.201779 1016.6879 -171.466130
## Oct 2016  283.244350 1018.2038  218.471859
## Nov 2016  -16.571712 1019.7197   98.522038
## Dec 2016 -147.473465 1018.8521  -69.548663
## Jan 2017  -70.915777 1017.9846   48.021194
## Feb 2017  -42.611463 1014.9185  -10.307084
## Mar 2017  162.492749 1011.8525  -44.105260
## Apr 2017   30.504883 1005.4660   18.739121
## May 2017  -10.059213  999.0795   55.929732
## Jun 2017    1.393934  997.5038   14.832273
## Jul 2017 -198.257090  995.9281 -104.341015
## Aug 2017   46.454532 1004.0896  -67.294128
## Sep 2017  -38.201779 1012.2511   13.590691
## Oct 2017  283.244350 1023.3028  -53.857162
## Nov 2017  -16.571712 1034.3545  -48.472825
## Dec 2017 -147.473465 1044.3906  -90.817098
## Jan 2018  -70.915777 1054.4266   77.059186
## Feb 2018  -42.611463 1065.3955  177.465993
## Mar 2018  162.492749 1076.3643   23.392902
## Apr 2018   30.504883 1084.6045   69.340655
## May 2018  -10.059213 1092.8446  -22.865363
## Jun 2018    1.393934 1095.4737 -103.317629
## Jul 2018 -198.257090 1098.1028    8.524276
## Aug 2018   46.454532 1092.5451  -42.069635
## Sep 2018  -38.201779 1086.9874   72.964385
## Oct 2018  283.244350 1084.1472   45.078464
## Nov 2018  -16.571712 1081.3070  -54.485268
## Dec 2018 -147.473465 1085.7953   31.798174
## Jan 2019  -70.915777 1090.2836   43.762173
## Feb 2019  -42.611463 1094.6359  -15.074463
## Mar 2019  162.492749 1098.9882 -130.610997
## Apr 2019   30.504883 1099.7370 -226.271912
## May 2019  -10.059213 1100.4858  194.523402
## Jun 2019    1.393934 1103.0818  161.084246
## Jul 2019 -198.257090 1105.6778  -58.780740
## Aug 2019   46.454532 1109.1747   91.770742
## Sep 2019  -38.201779 1112.6716   32.370157
## Oct 2019  283.244350 1115.9599 -182.124262
## Nov 2019  -16.571712 1119.2482  -10.836491
## Dec 2019 -147.473465 1122.1292   50.014278
```

```
plot(stl_decomp, labels = colnames(stl_decomp),
     main = 'Decomposition Plot of Time Series', range.bars = TRUE,
     col.range = "light blue")
```



Seasonally adjusted numbers

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

```
##            Jan       Feb       Mar       Apr       May       Jun       Jul
## 2015 1031.3358  986.6915 1266.6273 1066.4951  925.9092  782.0561 1232.7771
## 2016  867.3358  917.1615  929.0573 1111.3351  881.4192 1179.8161  955.8471
## 2017 1066.0058 1004.6115  967.7473 1024.2051 1055.0092 1012.3361  891.5871
## 2018 1131.4858 1242.8615 1099.7573 1153.9451 1069.9792  992.1561 1106.6271
## 2019 1134.0458 1079.5615  968.3773  873.4651 1295.0092 1264.1661 1046.8971
##            Aug       Sep       Oct       Nov       Dec
## 2015 1123.0455 1065.2818  979.0757 1015.8217 1076.8935
## 2016  926.2755  845.2218 1236.6757 1118.2417  949.3035
## 2017  936.7955 1025.8418  969.4457  985.8817  953.5735
## 2018 1050.4755 1159.9518 1129.2257 1026.8217 1117.5935
## 2019 1200.9455 1145.0418  933.8357 1108.4117 1172.1435
```

```
f_stl <- forecast(stl_decomp, h=24)
f_stl
```

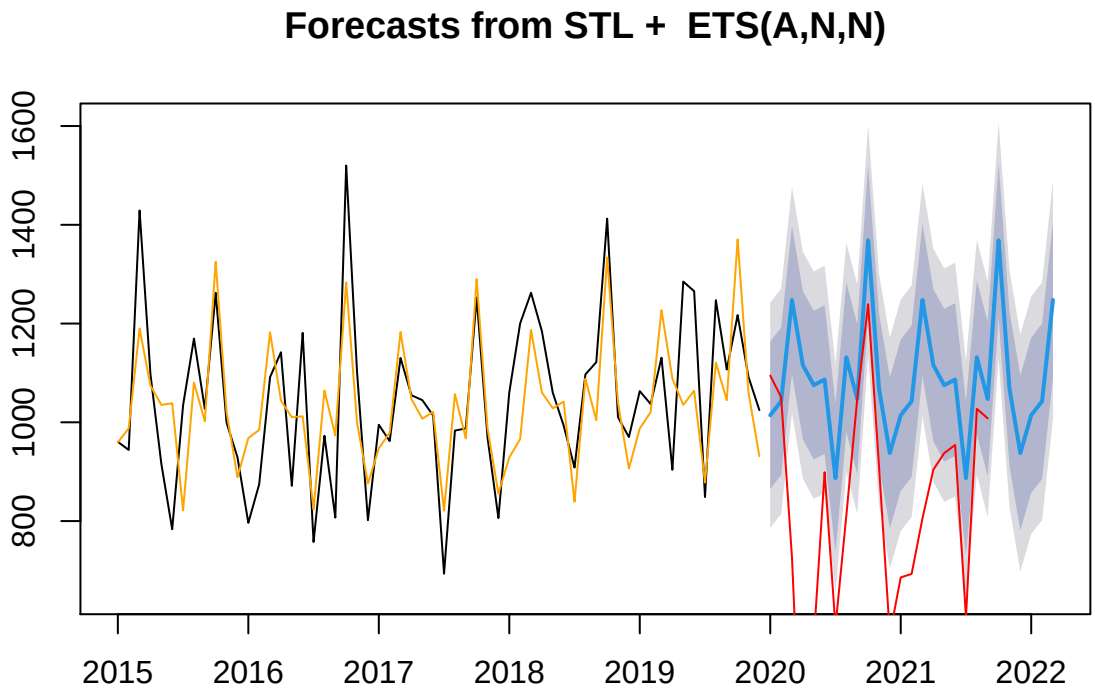
```
##          Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## Jan 2020      1014.3270  865.2967 1163.357  786.4049 1242.249
## Feb 2020      1042.6313  893.2525 1192.010  814.1762 1271.086
## Mar 2020      1247.7355 1098.0091 1397.462 1018.7487 1476.722
## Apr 2020      1115.7477  965.6743 1265.821  886.2303 1345.265
```

```
## May 2020      1075.1836  924.7641 1225.603  845.1369 1305.230
## Jun 2020      1086.6367  935.8720 1237.401  856.0619 1317.211
## Jul 2020       886.9857  735.8764 1038.095  655.8840 1118.087
## Aug 2020      1131.6973  980.2443 1283.150  900.0699 1363.325
## Sep 2020      1047.0410  895.2451 1198.837  814.8891 1279.193
## Oct 2020      1368.4871 1216.3490 1520.625 1135.8120 1601.162
## Nov 2020      1068.6711  916.1915 1221.151  835.4738 1301.868
## Dec 2020       937.7693  784.9491 1090.589  704.0510 1171.488
## Jan 2021      1014.3270  861.1669 1167.487  780.0889 1248.565
## Feb 2021      1042.6313  889.1321 1196.130  807.8746 1277.388
## Mar 2021      1247.7355 1093.8979 1401.573 1012.4613 1483.010
## Apr 2021      1115.7477  961.5724 1269.923  879.9570 1351.538
## May 2021      1075.1836  920.6714 1229.696  838.8777 1311.489
## Jun 2021      1086.6367  931.7884 1241.485  849.8167 1323.457
## Jul 2021       886.9857  731.8019 1042.169  649.6526 1124.319
## Aug 2021      1131.6973  976.1788 1287.216  893.8523 1369.542
## Sep 2021      1047.0410  891.1885 1202.893  808.6852 1285.397
## Oct 2021      1368.4871 1212.3014 1524.673 1129.6216 1607.353
## Nov 2021      1068.6711  912.1527 1225.189  829.2969 1308.045
## Dec 2021       937.7693  780.9191 1094.620  697.8876 1177.651
```

```
f_stl <- forecast(stl_decomp, h=27)
f_stl
```

```
##          Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## Jan 2020      1014.3270  865.2967 1163.357  786.4049 1242.249
## Feb 2020      1042.6313  893.2525 1192.010  814.1762 1271.086
## Mar 2020      1247.7355 1098.0091 1397.462 1018.7487 1476.722
## Apr 2020      1115.7477  965.6743 1265.821  886.2303 1345.265
## May 2020      1075.1836  924.7641 1225.603  845.1369 1305.230
## Jun 2020      1086.6367  935.8720 1237.401  856.0619 1317.211
## Jul 2020       886.9857  735.8764 1038.095  655.8840 1118.087
## Aug 2020      1131.6973  980.2443 1283.150  900.0699 1363.325
## Sep 2020      1047.0410  895.2451 1198.837  814.8891 1279.193
## Oct 2020      1368.4871 1216.3490 1520.625 1135.8120 1601.162
## Nov 2020      1068.6711  916.1915 1221.151  835.4738 1301.868
## Dec 2020       937.7693  784.9491 1090.589  704.0510 1171.488
## Jan 2021      1014.3270  861.1669 1167.487  780.0889 1248.565
## Feb 2021      1042.6313  889.1321 1196.130  807.8746 1277.388
## Mar 2021      1247.7355 1093.8979 1401.573 1012.4613 1483.010
## Apr 2021      1115.7477  961.5724 1269.923  879.9570 1351.538
## May 2021      1075.1836  920.6714 1229.696  838.8777 1311.489
## Jun 2021      1086.6367  931.7884 1241.485  849.8167 1323.457
## Jul 2021       886.9857  731.8019 1042.169  649.6526 1124.319
## Aug 2021      1131.6973  976.1788 1287.216  893.8523 1369.542
## Sep 2021      1047.0410  891.1885 1202.893  808.6852 1285.397
## Oct 2021      1368.4871 1212.3014 1524.673 1129.6216 1607.353
## Nov 2021      1068.6711  912.1527 1225.189  829.2969 1308.045
## Dec 2021       937.7693  780.9191 1094.620  697.8876 1177.651
## Jan 2022      1014.3270  857.1456 1171.508  773.9388 1254.715
## Feb 2022      1042.6313  885.1195 1200.143  801.7377 1283.525
## Mar 2022      1247.7355 1089.8939 1405.577 1006.3376 1489.133
```

```
plot(f_stl)
lines(f_stl$fitted, col = "orange")
lines(TC_tstest1, col = 'red')
```

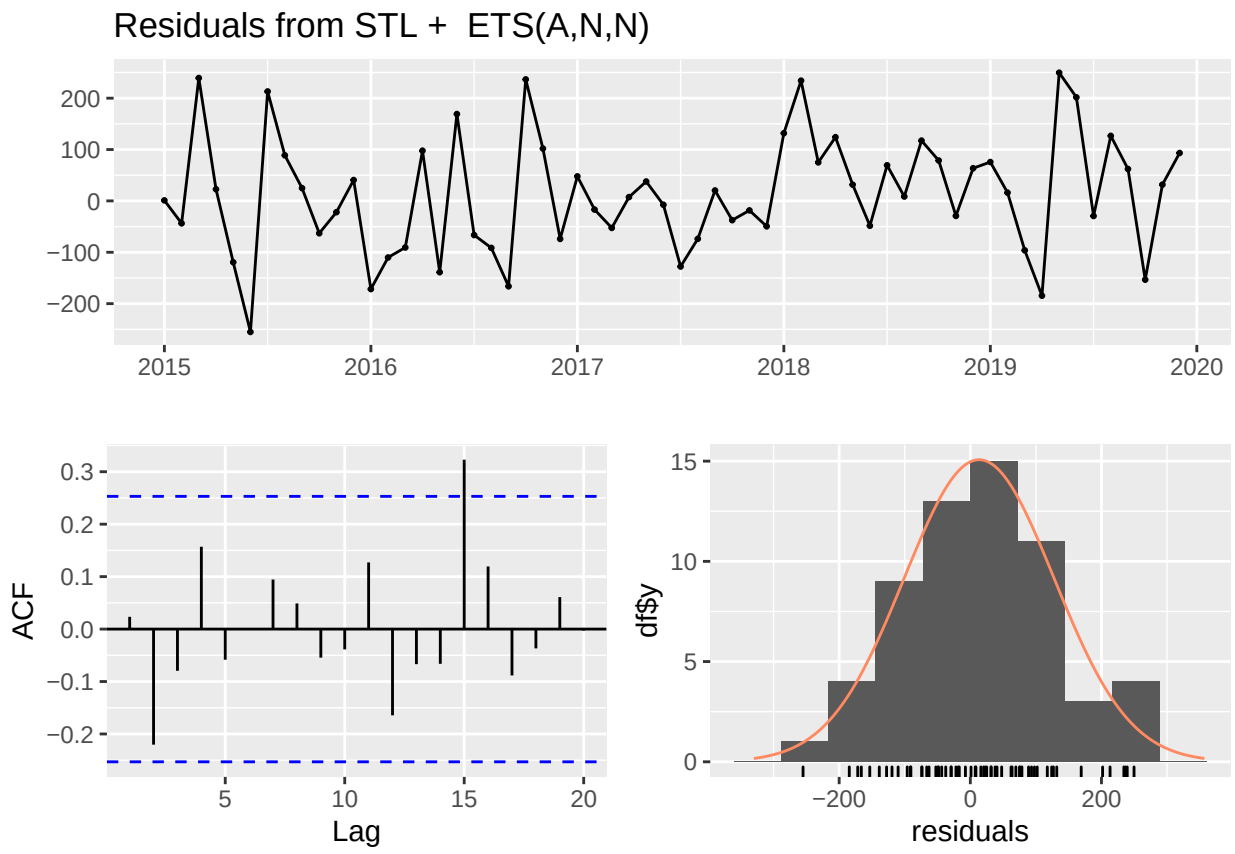



```
accuracy(f_stl)
```

```
##                     ME     RMSE      MAE       MPE     MAPE      MASE        ACF1
## Training set 13.38141 114.3344 91.29621 0.1050379 8.793197 0.6529485 0.02342607
```

```
checkresiduals(f_stl)
```

```
## Warning in checkresiduals(f_stl): 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* = 9.8731, df = 10, p-value = 0.4517
##
## Model df: 2.   Total lags used: 12
```

```
attributes(f_stl)
```

```
## $names
##  [1] "model"     "mean"      "level"     "x"         "upper"     "lower"
##  [7] "fitted"    "method"    "series"    "residuals"
##
## $class
## [1] "forecast"
```

```
resid <- f_stl$residuals
fitted <- f_stl$fitted
stldf <- data.table(resid, fitted)
stldf
```

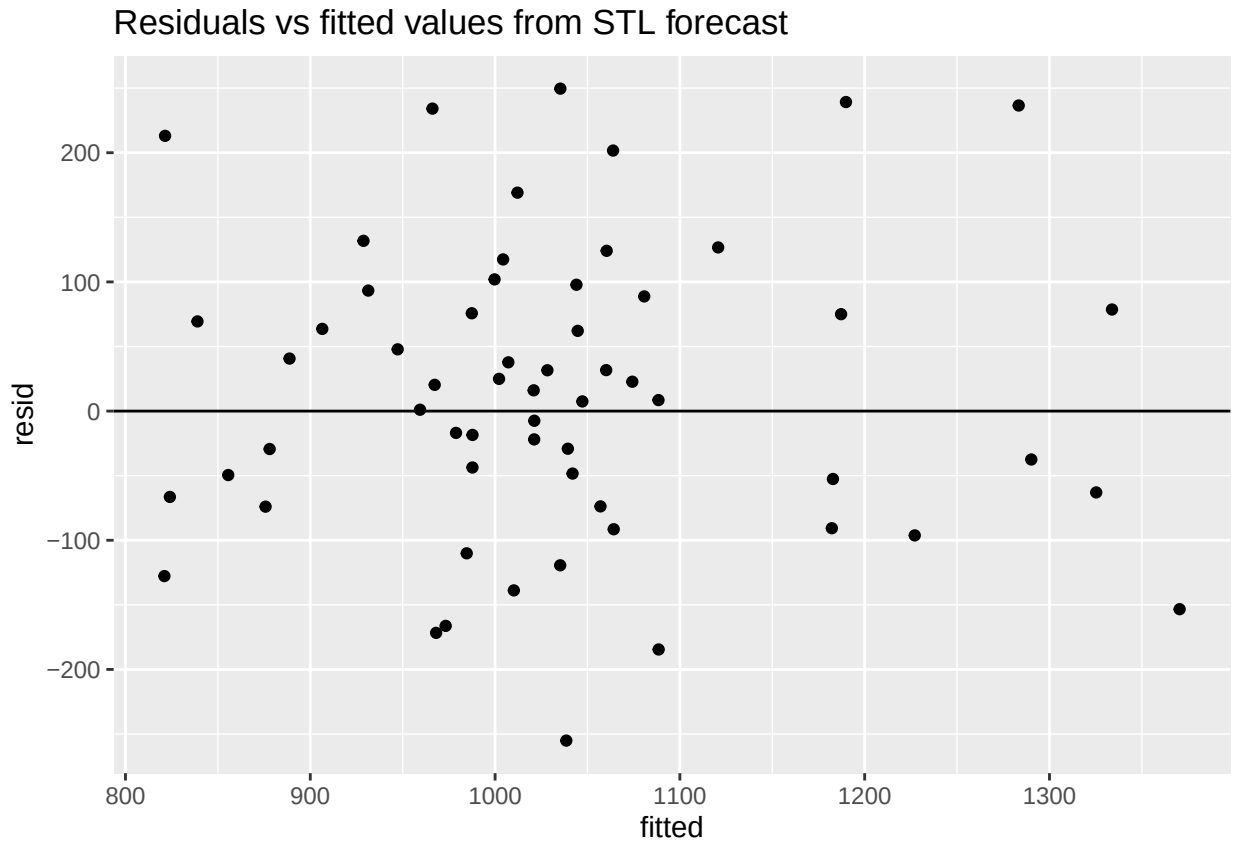
```
##           resid    fitted
##  1:    1.033490  959.3865
```

```
##  2:  -43.681545  987.7615
##  3:  239.243323 1189.8767
##  4:   22.740034 1074.2600
##  5: -119.401946 1035.2519
##  6: -255.084550 1038.5345
##  7:  213.091626  821.4284
##  8:   88.778381 1080.7216
##  9:   24.939687 1002.1403
## 10:  -62.973037 1325.2930
## 11:  -21.917801 1021.1678
## 12:   40.653764  888.7662
## 13: -171.685817  968.1058
## 14: -110.111861  984.6619
## 15:  -90.681241 1182.2312
## 16:   97.801842 1044.0382
## 17: -138.806534 1010.1665
## 18:  169.088696 1012.1213
## 19:  -66.450832  824.0408
## 20:  -91.475297 1064.2053
## 21: -166.269433  973.2894
## 22:  236.562070 1283.3579
## 23:  101.940452  999.7295
## 24:  -73.973465  875.8035
## 25:   47.790768  947.2992
## 26:  -16.873815  978.8738
## 27:  -52.583371 1182.8234
## 28:    7.472717 1047.2373
## 29:   37.765463 1007.1845
## 30:   -7.491933 1021.2219
## 31: -127.728244  821.0582
## 32:  -73.779564 1057.0296
## 33:   20.315400  967.3246
## 34:  -37.470889 1290.1609
## 35:  -18.470736  987.7807
## 36:  -49.515050  855.6151
## 37:  131.785521  928.7845
## 38:  234.143271  966.1067
## 39:   75.016895 1187.2331
## 40:  124.071438 1060.3786
## 41:   31.615463 1028.3045
## 42:  -48.371094 1041.9211
## 43:   69.409909  838.9601
## 44:    8.508645 1088.4214
## 45:  117.402719 1004.3473
## 46:   78.642853 1333.8271
## 47:  -29.142529 1039.3925
## 48:   63.623416  906.4966
## 49:   75.722048  987.4080
## 50:   16.056158 1020.8938
## 51:  -96.226759 1227.0968
## 52: -184.554203 1088.5242
## 53:  249.618732 1035.3313
## 54:  201.694453 1063.8655
## 55:  -29.376251  878.0163
```

```
## 56:  126.682312 1120.7177
## 57:   62.109893 1044.7301
## 58: -153.346346 1370.4263
## 59:   31.723035 1060.1170
## 60:   93.284017  931.3860
##           resid    fitted
```

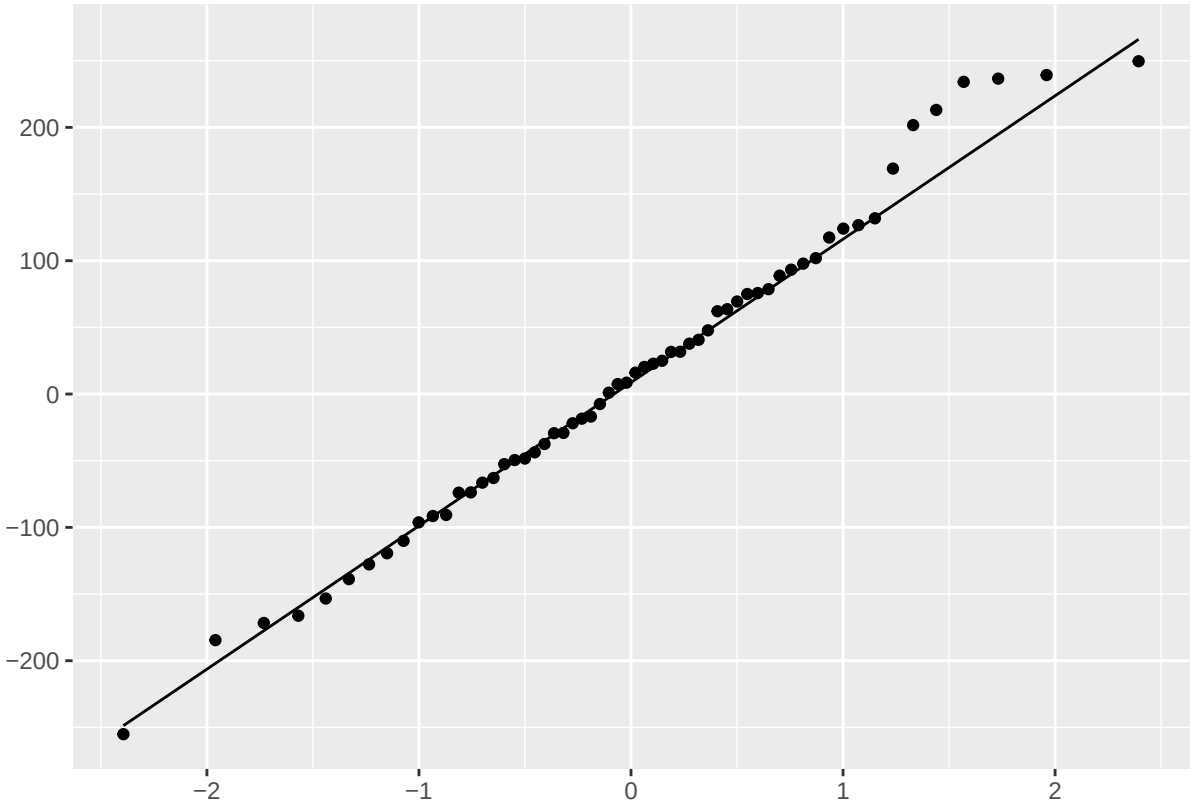
```
ggplot(data = stldf, aes(x=fitted, y = resid)) + geom_point() + geom_hline(yintercept = 0)+
  labs(title = 'Residuals vs fitted values from STL forecast')
```

```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
```




```
qplot(sample = resid, data = stldf, geom = 'qq') + geom_qq_line()
```

```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
```


200
100
0
−100
−200
−2
−1
0
1
2