time series

Ayoub Asri

2024-03-26

Contents

R packages

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.0     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

data

```
a <- getwd()
file <- paste(str_remove(a,"/time series ideas"),"/data/data_article/hourly_5y_data.csv",sep ="")
data <- read_csv(file)
```

```
## Rows: 41607 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl  (1): measure
## dttm (1): time
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

time series packages

```
library(tseries)
```

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

```
library(timetk)
library(readxl)
library(tidyquant)
```

```
## Loading required package: PerformanceAnalytics

## Loading required package: xts

## Loading required package: zoo

##
## Attaching package: 'zoo'

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

##
## ############################ Warning from 'xts' package ##########################
## #                                                                                #
## # The dplyr lag() function breaks how base R's lag() function is supposed to  #
## # work, which breaks lag(my_xts). Calls to lag(my_xts) that you type or       #
## # source() into this session won't work correctly.                               #
## #                                                                                #
## # Use stats::lag() to make sure you're not using dplyr::lag(), or you can add #
```

```
## # conflictRules('dplyr', exclude = 'lag') to your .Rprofile to stop           #
## # dplyr from breaking base R's lag() function.                                #
## #                                                                             #
## # Code in packages is not affected. It's protected by R's namespace mechanism #
## # Set `options(xts.warn_dplyr_breaks_lag = FALSE)` to suppress this warning.  #
## #                                                                             #
## ###############################################################################

##
## Attaching package: 'xts'

## The following objects are masked from 'package:dplyr':
##
##     first, last

##
## Attaching package: 'PerformanceAnalytics'

## The following object is masked from 'package:graphics':
##
##     legend

## Loading required package: quantmod

## Loading required package: TTR
```

```
library(scales)
```

```
##
## Attaching package: 'scales'

## The following object is masked from 'package:purrr':
##
##     discard

## The following object is masked from 'package:readr':
##
##     col_factor
```

```
library(forecast)   #  forecasting pkg
library(sweep)      #  Broom tidiers for forecast pkg
library(timekit)
```

```
##
## Attaching package: 'timekit'

## The following objects are masked from 'package:timetk':
##
##     tk_augment_timeseries_signature, tk_get_timeseries_signature,
##     tk_get_timeseries_summary, tk_get_timeseries_unit_frequency,
##     tk_get_timeseries_variables, tk_index, tk_make_future_timeseries,
##     tk_tbl, tk_ts, tk_ts_, tk_ts_.data.frame, tk_ts_.default, tk_xts,
##     tk_xts_, tk_zoo, tk_zoo_, tk_zooreg, tk_zooreg_,
##     tk_zooreg_.data.frame, tk_zooreg_.default
```

```
library(broom)
library(ggthemes)
```

exploration

```
data %>% glimpse()
```

```
## Rows: 41,607
## Columns: 2
## $ time    <dttm> 2019-01-01 00:00:00, 2019-01-01 01:00:00, 2019-01-01 02:00:00~
## $ measure <dbl> 13.996154, 13.470886, 10.388608, 8.771795, 7.373077, 7.323377,~
```

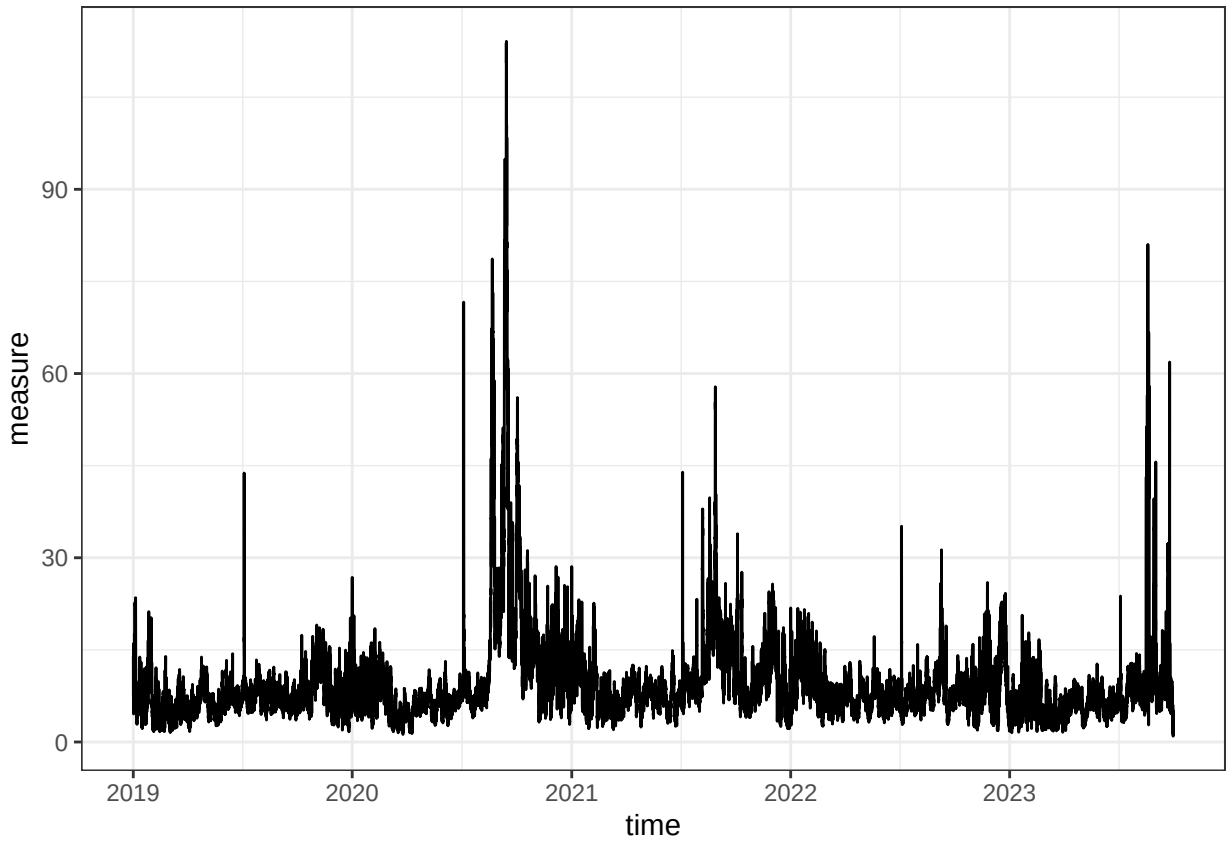
```
data %>%
  summarize_all(.funs = ~sum(is.na(.)))
```

```
## # A tibble: 1 x 2
##    time measure
##   <int>   <int>
## 1     0       0
```

```
data %>%
  summarize(mean = mean(measure),
            max = max(measure),
            min = mean(measure),
            IQR = IQR(measure))
```

```
## # A tibble: 1 x 4
##    mean   max   min   IQR
##   <dbl> <dbl> <dbl> <dbl>
## 1  9.19  114.  9.19  4.47
```

```
data %>%
  ggplot(aes(x = time, y = measure)) +
  geom_line() +
  theme_bw()
```



train/test split

ts object

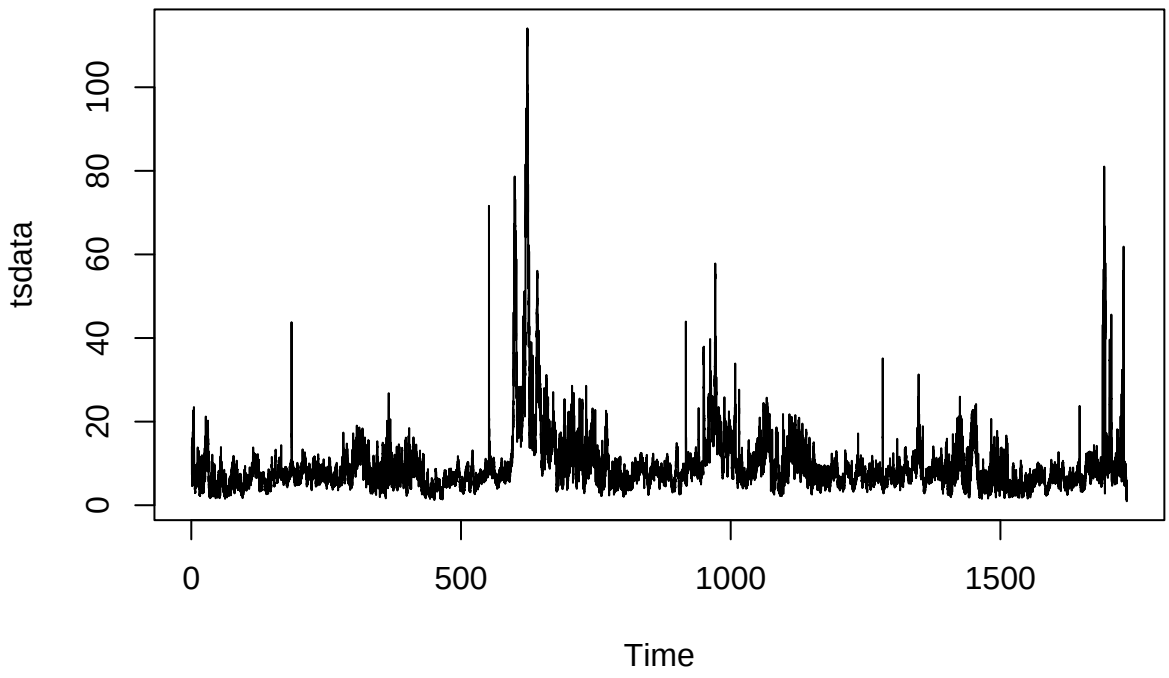
frequency is used to precise the frequency of data (here 24 is used to precise that is hourly data)

1. create the ts for all the data

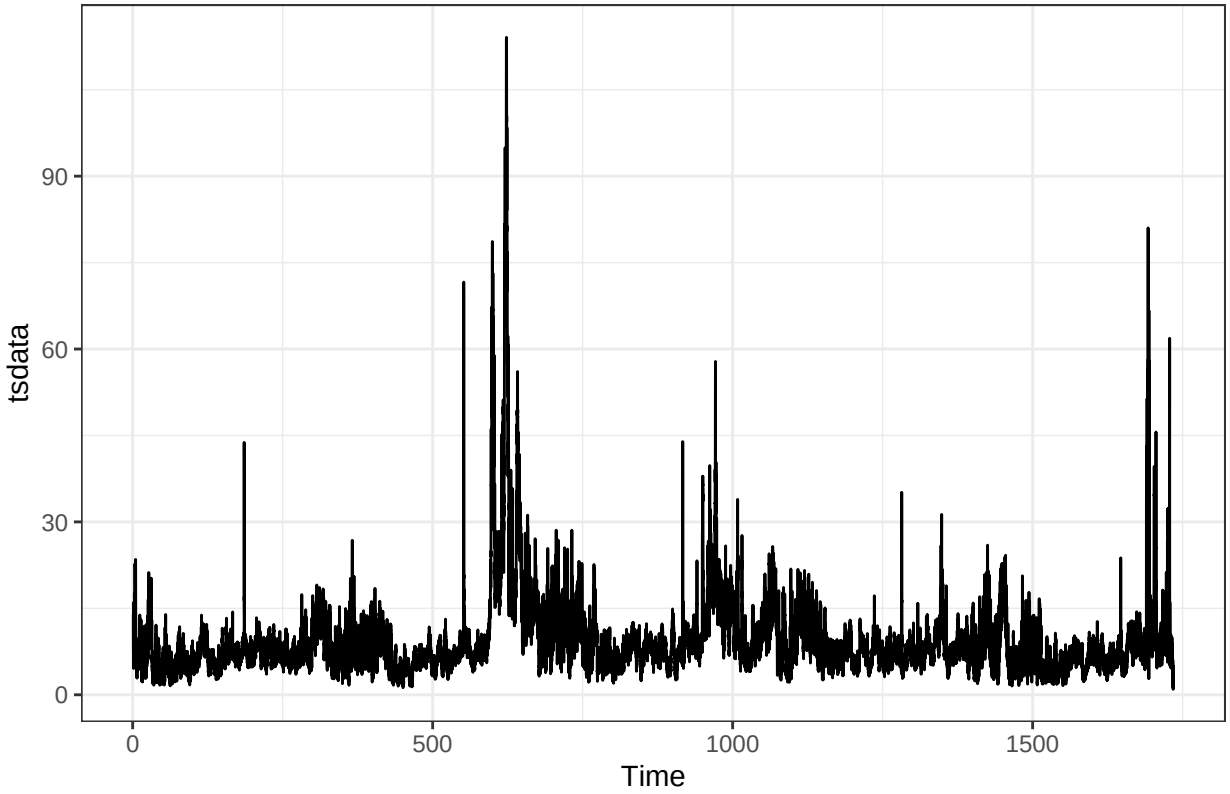
```
tsdata <- ts(data$measure,frequency = 24)
tsdata %>% head()
```

```
## Time Series:
## Start = c(1, 1)
## End = c(1, 6)
## Frequency = 24
## [1] 13.996154 13.470886 10.388608  8.771795  7.373077  7.323377
```

```
plot.ts(tsdata)
```



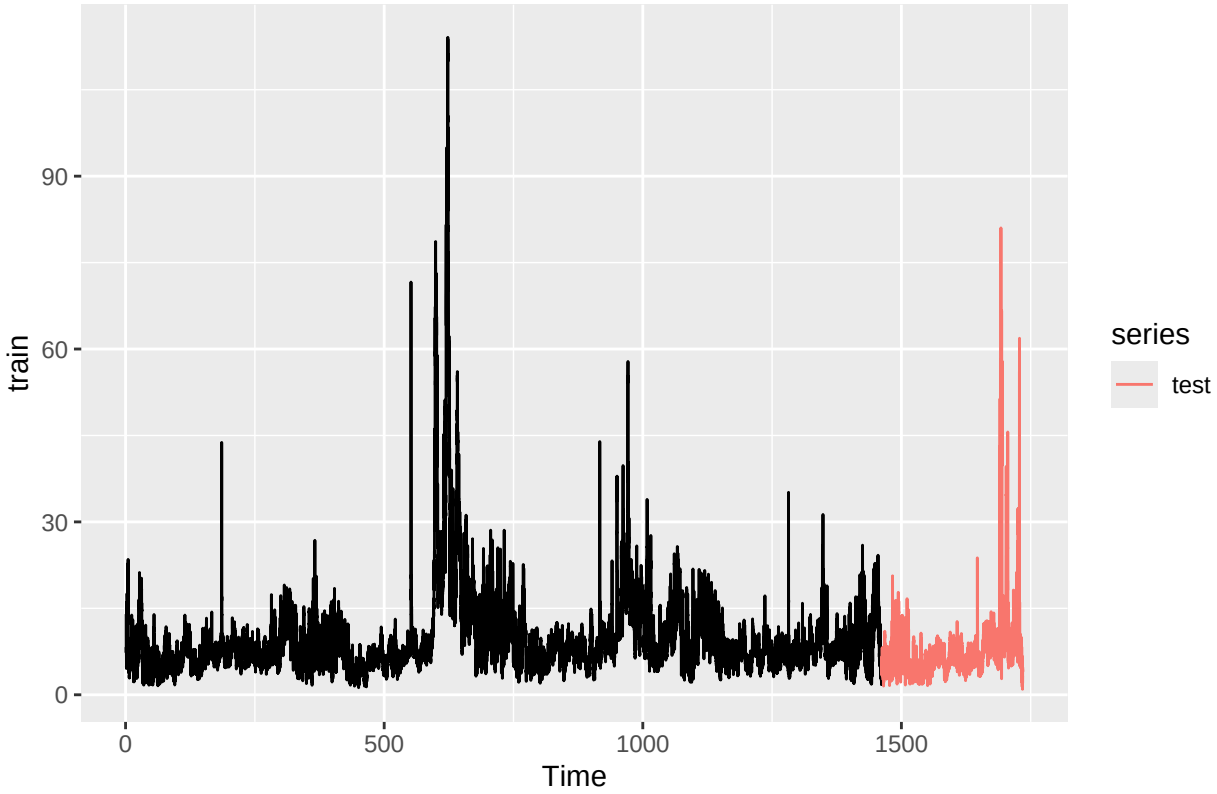

```
autoplot(tsdata) + theme_bw()
```



2. train 4 years : test = 2023

```
train <- tsdata %>% window(end = c(1461,24))
test <- tsdata %>% window(start = c(1462,1))
```

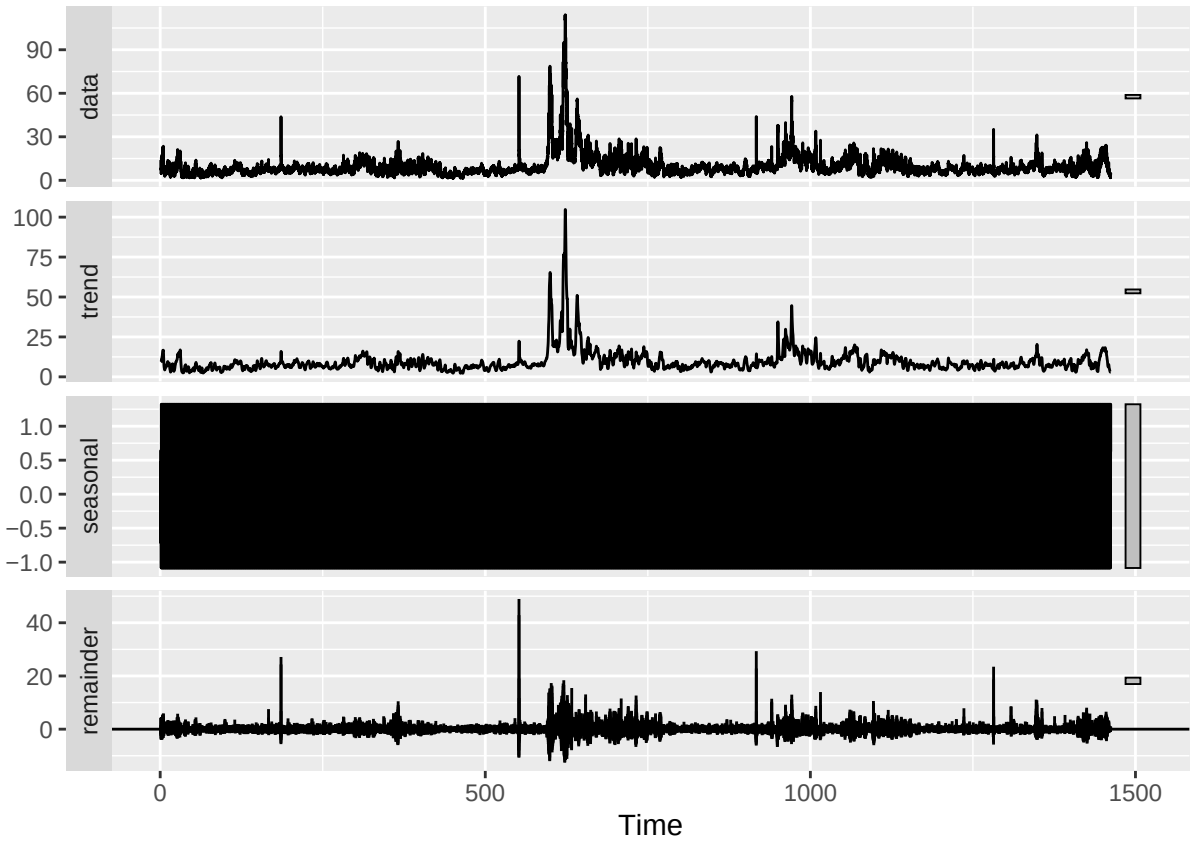
```
autoplot(train) +
  autolayer(test,series = "test")
```



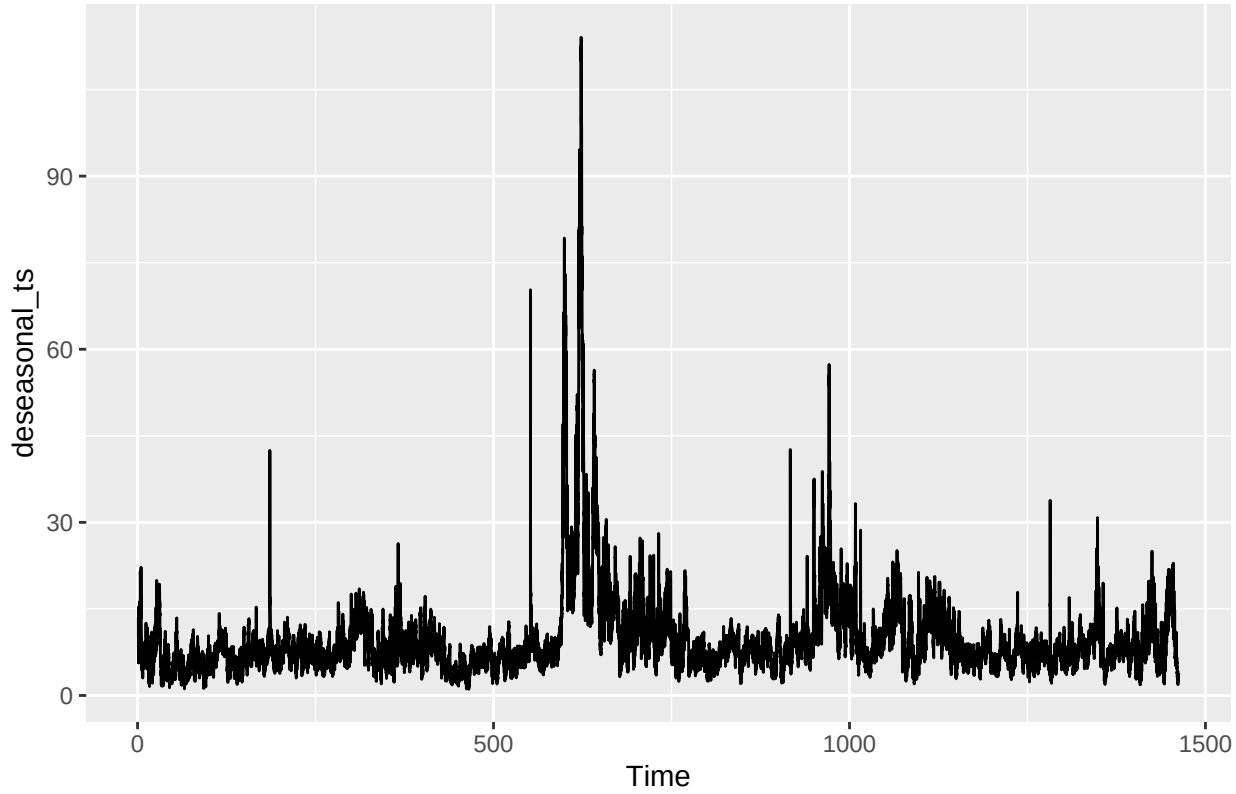
ARIMA model

stl is better than decompose() because it use non-linear approximation

```
decomp = stl(train,s.window = "periodic")

deseasonal_ts <- seasadj(decomp)

autoplot(decomp)
```




```
autoplot(deseasonal_ts)
```

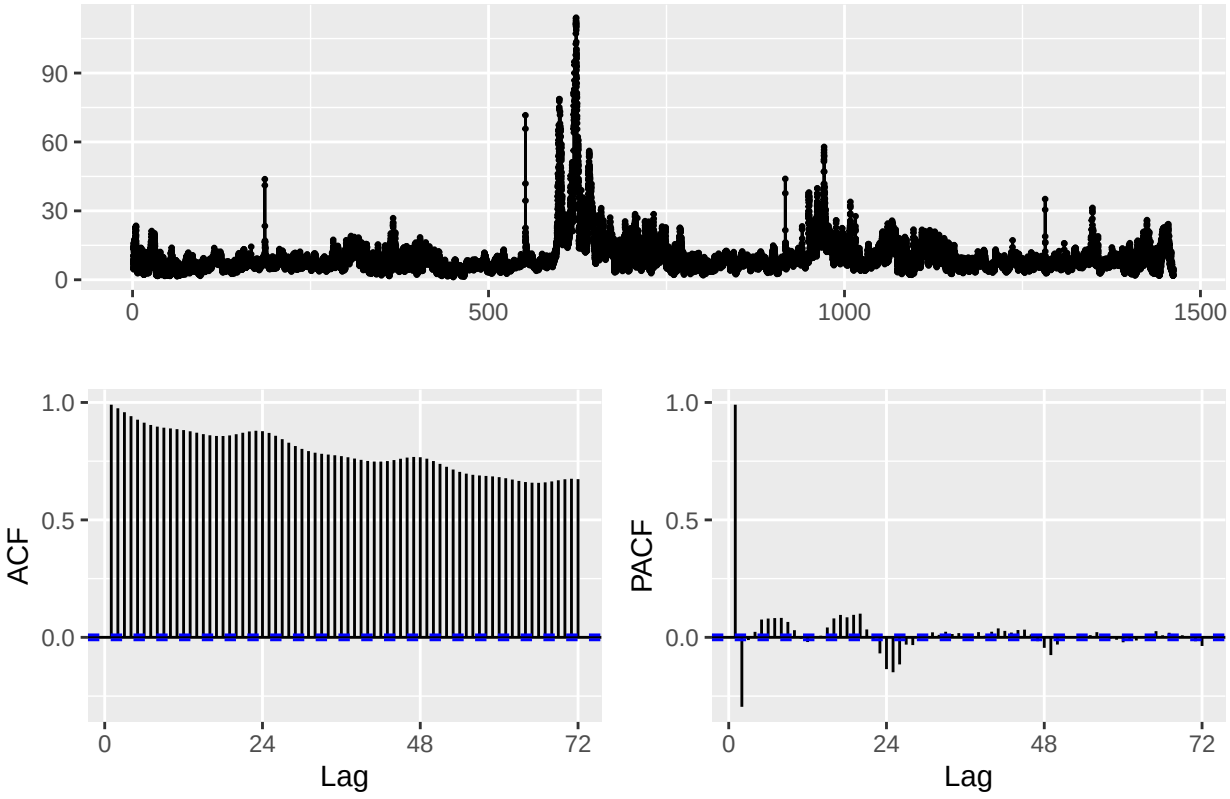



```
sweep::sw_tidy_decomp(decomp) %>% head()
```

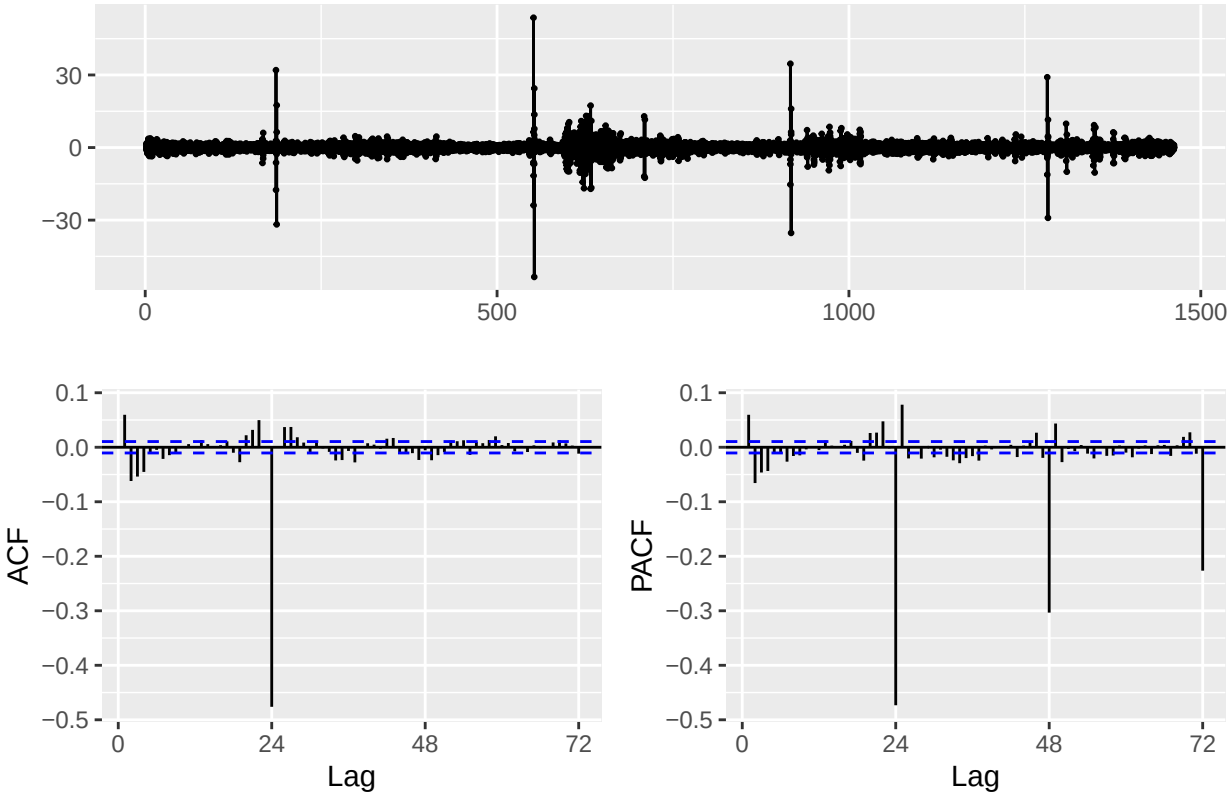
```
## # A tibble: 6 x 6
##   index observed  season trend remainder seasadj
##   <dbl>    <dbl>   <dbl> <dbl>     <dbl>   <dbl>
## 1  1       14.0   0.486   9.19     4.32    13.5
## 2  1.04    13.5   0.0420  9.21     4.22    13.4
## 3  1.08    10.4  -0.360   9.23     1.52    10.7
## 4  1.12     8.77 -0.627   9.25     0.146    9.40
## 5  1.17     7.37 -0.718   9.27    -1.18     8.09
## 6  1.21     7.32 -0.586   9.30    -1.39     7.91
```

some ideas

```
ggtsdisplay(train)
```




```
ggtsdisplay(diff(diff(train),lag = 24))
```

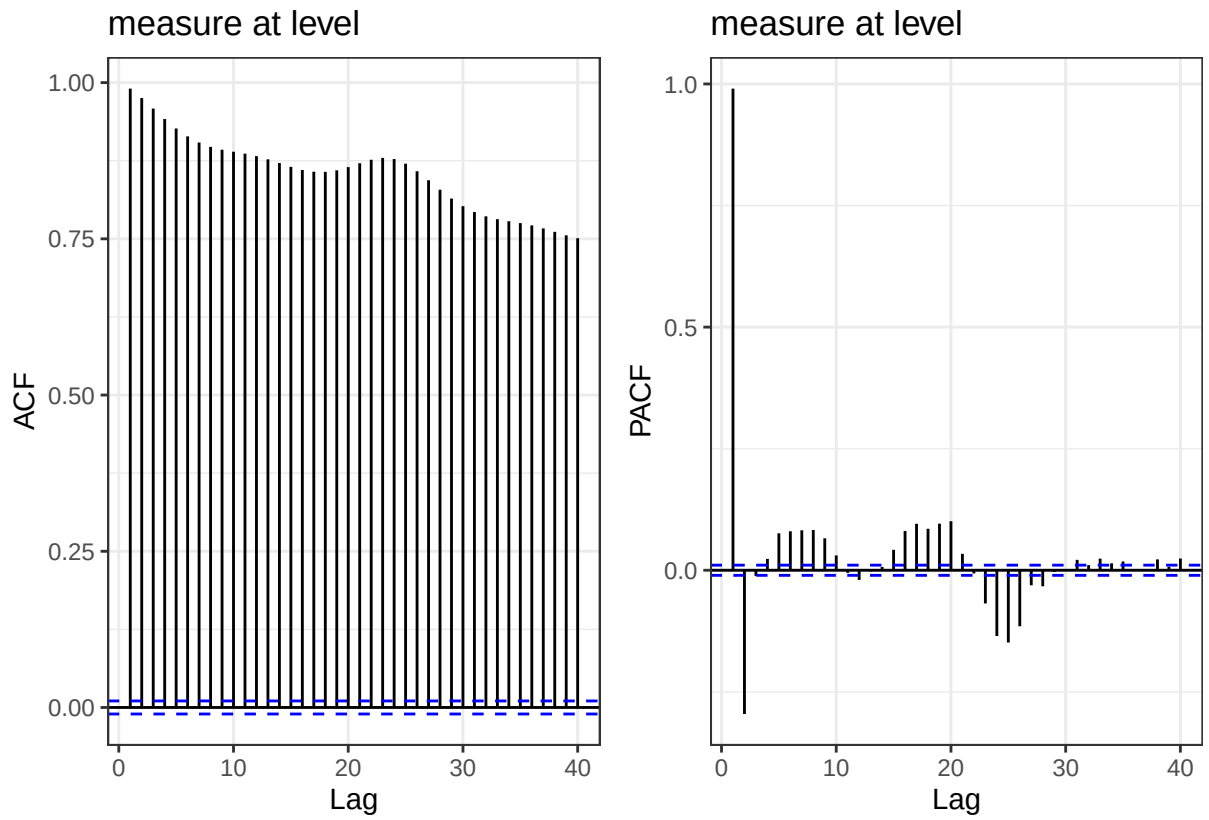


Autocorrelation plots

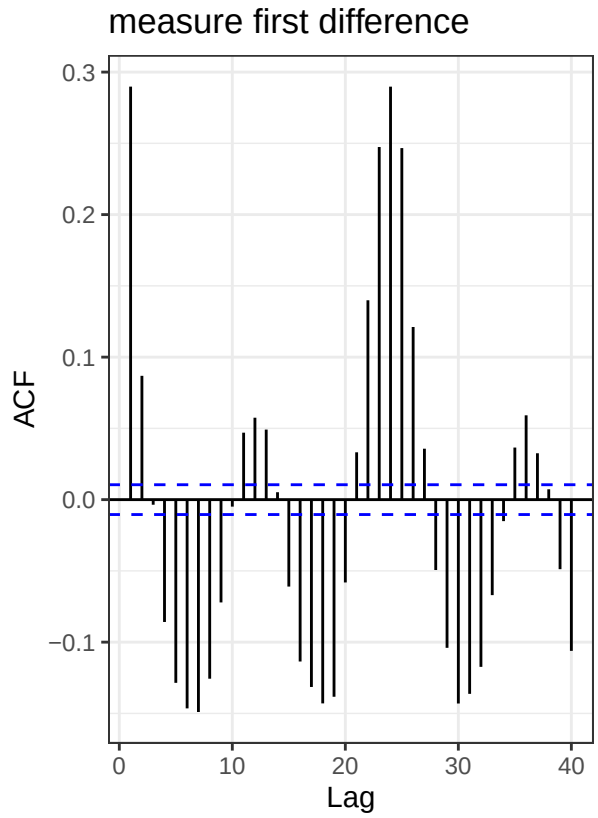
at level

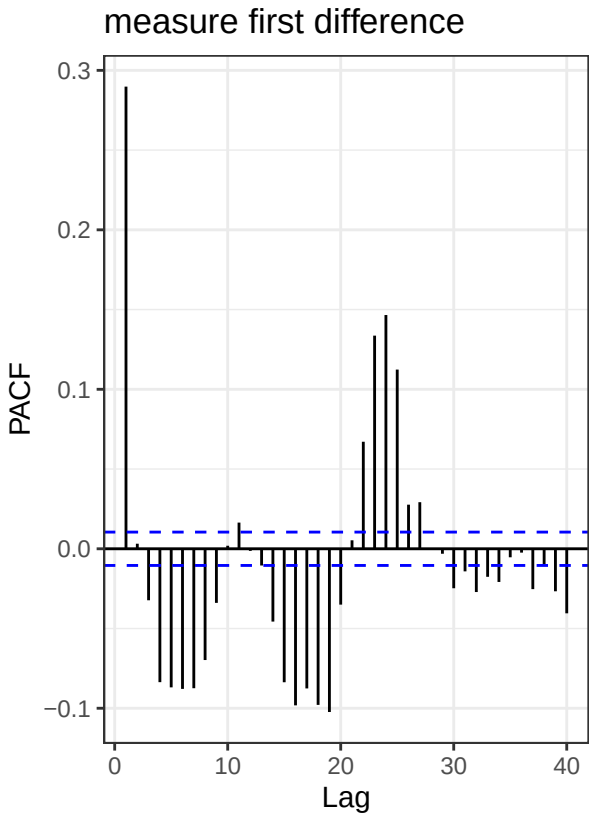
```
library(patchwork)
```

```
p1 <- ggAcf(train,lag=40) + labs(title="measure at level") + theme_bw()
p2 <- ggPacf(train,lag=40) + labs(title="measure at level") + theme_bw()

p1 + p2
```

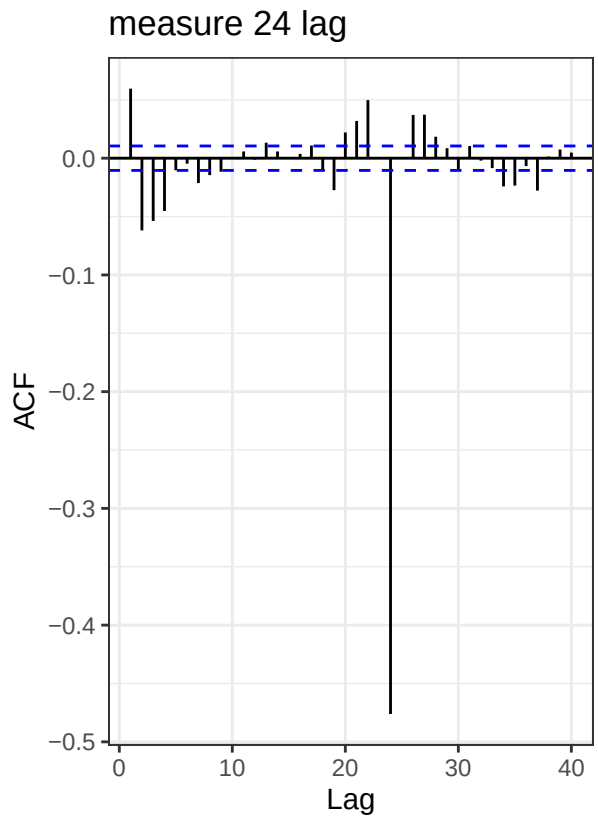


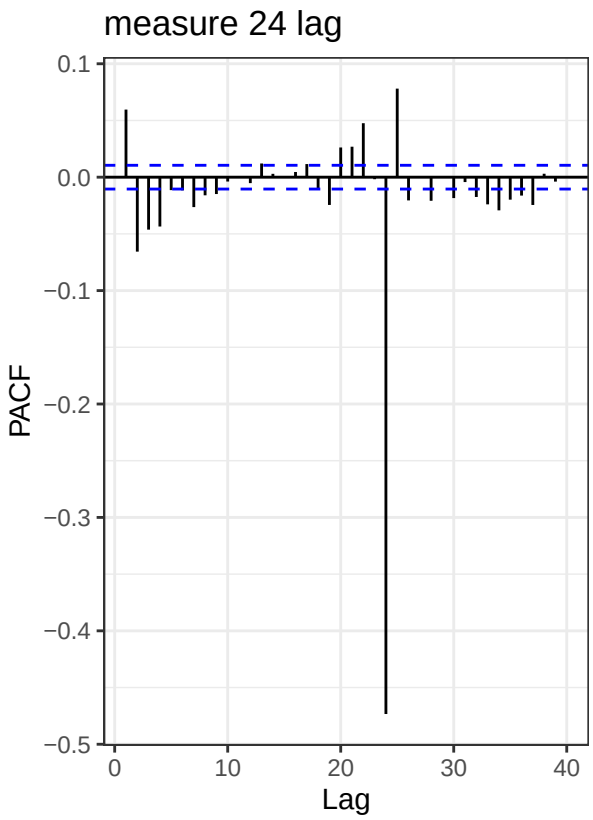

```
p1 <- ggAcf(diff(train),lag=40) + labs(title="measure first difference ") + theme_bw()
p2 <- ggPacf(diff(train),lag=40) + labs(title="measure first difference") + theme_bw()

p1 + p2
```



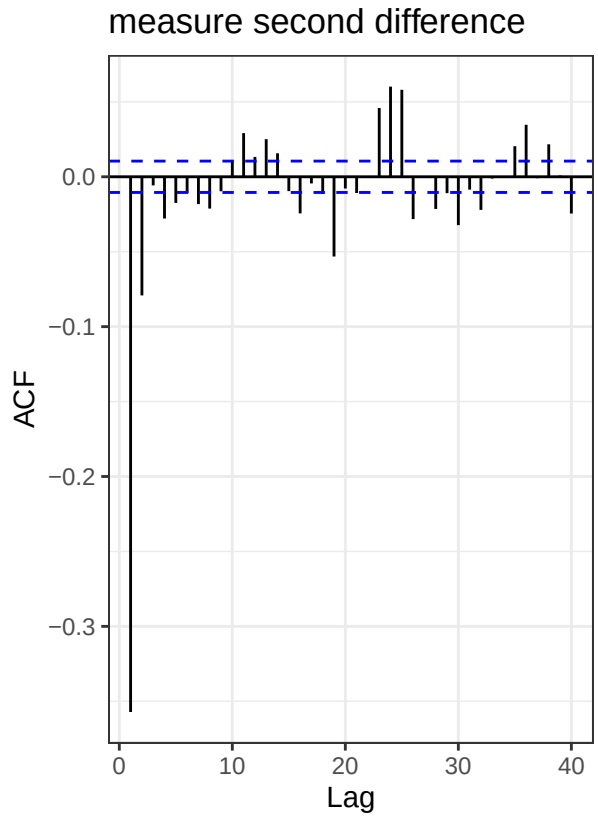


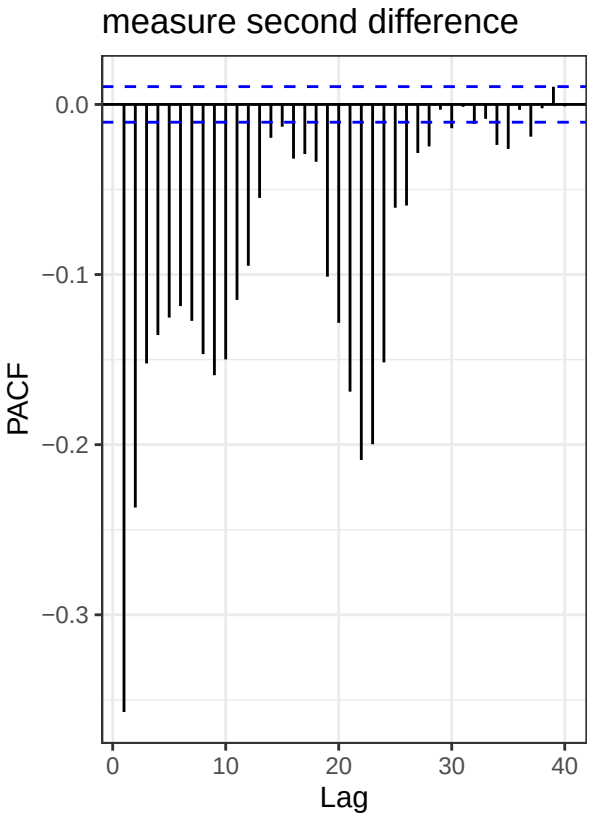

```
p1 <- ggAcf(diff(diff(train), lag = 24),lag=40) + labs(title="measure 24 lag") + theme_bw()
p2 <- ggPacf(diff(diff(train),lag = 24),lag=40) + labs(title="measure 24 lag") + theme_bw()

p1 + p2
```






```
p1 <- ggAcf(diff(diff(train)),lag=40) + labs(title="measure second difference ") + theme_bw()
p2 <- ggPacf(diff(diff(train)),lag=40) + labs(title="measure second difference") + theme_bw()

p1 + p2
```





ADF test

level

```
sw_glance(adf.test(train))
```

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

```
## # A tibble: 1 x 5
##   statistic p.value parameter method                      alternative
##       <dbl>   <dbl>     <dbl> <chr>                       <chr>
## 1     -10.5    0.01        32 Augmented Dickey-Fuller Test stationary
```

```
sw_glance(adf.test(diff(train,differences = 1)))
```

```
## Warning in adf.test(diff(train, differences = 1)): p-value smaller than printed
## p-value
```

```
## # A tibble: 1 x 5
##   statistic p.value parameter method                      alternative
##       <dbl>   <dbl>     <dbl> <chr>                       <chr>
## 1     -32.2    0.01        32 Augmented Dickey-Fuller Test stationary
```

```
sw_glance(adf.test(diff(train,differences = 2)))
```

```
## Warning in adf.test(diff(train, differences = 2)): p-value smaller than printed
## p-value
```

```
## # A tibble: 1 x 5
##   statistic p.value parameter method                      alternative
##       <dbl>   <dbl>     <dbl> <chr>                       <chr>
## 1     -59.4    0.01        32 Augmented Dickey-Fuller Test stationary
```

```
sw_glance(adf.test(diff(train,lag = 24)))
```

```
## Warning in adf.test(diff(train, lag = 24)): p-value smaller than printed
## p-value
```

```
## # A tibble: 1 x 5
##   statistic p.value parameter method                      alternative
##       <dbl>   <dbl>     <dbl> <chr>                       <chr>
## 1     -27.1    0.01        32 Augmented Dickey-Fuller Test stationary
```

```
sw_glance(adf.test(diff(train,differences = 1,lag = 24)))
```

```
## Warning in adf.test(diff(train, differences = 1, lag = 24)): p-value smaller
## than printed p-value
```

```
## # A tibble: 1 x 5
##   statistic p.value parameter method                      alternative
##       <dbl>   <dbl>     <dbl> <chr>                       <chr>
## 1     -27.1    0.01        32 Augmented Dickey-Fuller Test stationary
```

```
sw_glance(adf.test(diff(train,differences = 2,lag = 24)))
```

```
## Warning in adf.test(diff(train, differences = 2, lag = 24)): p-value smaller
## than printed p-value
```

```
## # A tibble: 1 x 5
##   statistic p.value parameter method                      alternative
##       <dbl>   <dbl>     <dbl> <chr>                       <chr>
## 1     -35.5    0.01        32 Augmented Dickey-Fuller Test stationary
```

Auto Arima

```
aar1 <- auto.arima(train, seasonal=TRUE,stepwise = T,trace = T,D = 1,max.p = 3,max.q = 3)
```

```
##
##  Fitting models using approximations to speed things up...
##
##  ARIMA(2,0,2)(1,1,1)[24] with drift         : Inf
##  ARIMA(0,0,0)(0,1,0)[24] with drift         : 190972.9
##  ARIMA(1,0,0)(1,1,0)[24] with drift         : 104479.8
##  ARIMA(0,0,1)(0,1,1)[24] with drift         : 155621.2
##  ARIMA(0,0,0)(0,1,0)[24]                    : 190970.9
##  ARIMA(1,0,0)(0,1,0)[24] with drift         : 113135.3
##  ARIMA(1,0,0)(2,1,0)[24] with drift         : 101303.6
##  ARIMA(1,0,0)(2,1,1)[24] with drift         : Inf
##  ARIMA(1,0,0)(1,1,1)[24] with drift         : Inf
##  ARIMA(0,0,0)(2,1,0)[24] with drift         : 190723.7
##  ARIMA(2,0,0)(2,1,0)[24] with drift         : 100790.8
##  ARIMA(2,0,0)(1,1,0)[24] with drift         : 104018.5
##  ARIMA(2,0,0)(2,1,1)[24] with drift         : Inf
##  ARIMA(2,0,0)(1,1,1)[24] with drift         : Inf
##  ARIMA(3,0,0)(2,1,0)[24] with drift         : 100750.7
##  ARIMA(3,0,0)(1,1,0)[24] with drift         : 103984.2
##  ARIMA(3,0,0)(2,1,1)[24] with drift         : Inf
##  ARIMA(3,0,0)(1,1,1)[24] with drift         : Inf
##  ARIMA(3,0,1)(2,1,0)[24] with drift         : 100706.6
##  ARIMA(3,0,1)(1,1,0)[24] with drift         : 103964.8
##  ARIMA(3,0,1)(2,1,1)[24] with drift         : Inf
##  ARIMA(3,0,1)(1,1,1)[24] with drift         : Inf
##  ARIMA(2,0,1)(2,1,0)[24] with drift         : 100757.1
##  ARIMA(3,0,2)(2,1,0)[24] with drift         : 100762
##  ARIMA(2,0,2)(2,1,0)[24] with drift         : Inf
##  ARIMA(3,0,1)(2,1,0)[24]                    : 100704.6
##  ARIMA(3,0,1)(1,1,0)[24]                    : 103962.8
##  ARIMA(3,0,1)(2,1,1)[24]                    : Inf
##  ARIMA(3,0,1)(1,1,1)[24]                    : Inf
##  ARIMA(2,0,1)(2,1,0)[24]                    : 100755.1
##  ARIMA(3,0,0)(2,1,0)[24]                    : 100748.7
##  ARIMA(3,0,2)(2,1,0)[24]                    : 100760
##  ARIMA(2,0,0)(2,1,0)[24]                    : 100788.8
##  ARIMA(2,0,2)(2,1,0)[24]                    : Inf
##
##  Now re-fitting the best model(s) without approximations...
##
##  ARIMA(3,0,1)(2,1,0)[24]                    : Inf
##  ARIMA(3,0,1)(2,1,0)[24] with drift         : Inf
##  ARIMA(3,0,0)(2,1,0)[24]                    : Inf
##  ARIMA(3,0,0)(2,1,0)[24] with drift         : Inf
##  ARIMA(2,0,1)(2,1,0)[24]                    : Inf
##  ARIMA(2,0,1)(2,1,0)[24] with drift         : Inf
##  ARIMA(3,0,2)(2,1,0)[24]                    : Inf
##  ARIMA(3,0,2)(2,1,0)[24] with drift         : Inf
##  ARIMA(2,0,0)(2,1,0)[24]                    : Inf
##  ARIMA(2,0,0)(2,1,0)[24] with drift         : Inf
##  ARIMA(1,0,0)(2,1,0)[24] with drift         : 101358.4
##
##  Best model: ARIMA(1,0,0)(2,1,0)[24] with drift
```

```
sw_tidy(aar1)
```

```
## # A tibble: 4 x 2
##   term   estimate
##   <chr>     <dbl>
## 1 ar1    0.971
## 2 sar1  -0.613
## 3 sar2  -0.296
## 4 drift -0.000182
```

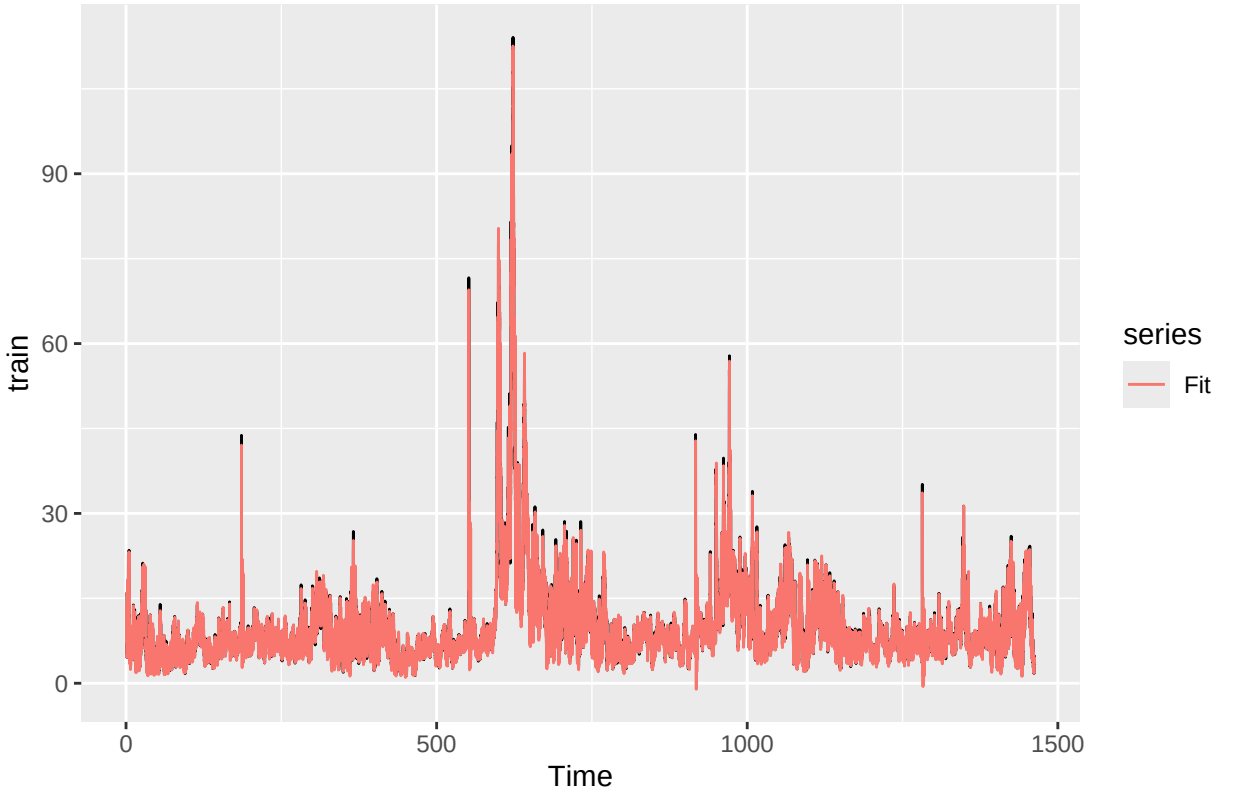
```
sw_glance(aar1)
```

```
## # A tibble: 1 x 12
##   model.desc sigma  logLik    AIC    BIC       ME  RMSE   MAE    MPE  MAPE  MASE
##   <chr>      <dbl>   <dbl>  <dbl>  <dbl>    <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1 ARIMA(1,0~  1.03 -50674. 1.01e5 1.01e5 -3.56e-5  1.03 0.570 -0.358  6.61 0.277
## # i 1 more variable: ACF1 <dbl>
```

```
sw_augment(aar1) %>% head()
```

```
## # A tibble: 6 x 4
##   index .actual .fitted  .resid
##   <dbl>   <dbl>   <dbl>   <dbl>
## 1  1      14.0    14.0  0.0140
## 2  1.04   13.5    13.5  0.0135
## 3  1.08   10.4    10.4  0.0104
## 4  1.12    8.77    8.76 0.00877
## 5  1.17    7.37    7.37 0.00737
## 6  1.21    7.32    7.32 0.00732
```

```
autoplot(train) +
  autolayer(aar1$fitted,series="Fit")
```



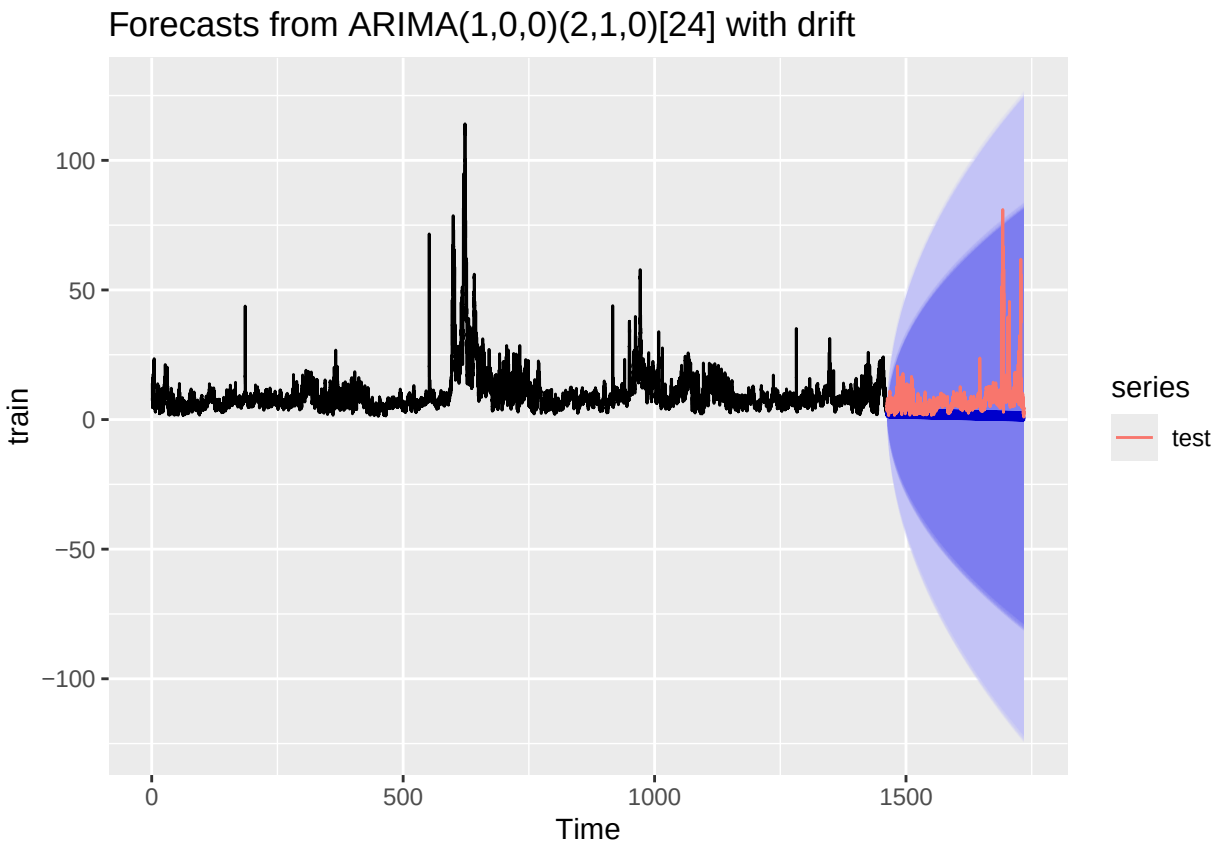
forecast on test

```
f1 <- forecast(aar1,h = 6543)
#f1 %>% head()
```

```
summary(aar1)
```

```
## Series: train
## ARIMA(1,0,0)(2,1,0)[24] with drift
##
## Coefficients:
##          ar1     sar1     sar2    drift
##       0.9709  -0.6127  -0.2958  -0.0002
## s.e.  0.0013   0.0051   0.0051   0.0041
##
## sigma^2 = 1.056:  log likelihood = -50674.19
## AIC=101358.4   AICc=101358.4   BIC=101400.7
##
## Training set error measures:
##                         ME     RMSE       MAE        MPE     MAPE      MASE
## Training set -3.557011e-05 1.027079 0.5700497 -0.3576875 6.614659 0.2771442
##                    ACF1
## Training set 0.1166776
```

```r
aar1 %>%
  forecast(h=6543) %>%
  autoplot() + autolayer(test)
```




```r
accuracy(f1 , test)
```

```
##                       ME     RMSE       MAE        MPE      MAPE      MASE
## Training set -3.557011e-05 1.027079 0.5700497 -0.3576875  6.614659 0.2771442
## Test set      6.083488e+00 8.373799 6.0897210 73.5912045 73.857791 2.9606734
##                   ACF1 Theil's U
## Training set 0.1166776        NA
## Test set     0.9426066  3.062574
```

comments :

no the best of results needs some reflexion !!