Seasonal and Trend decomposition using Loess (STL) for Spatio-temporal Data

Rizka Amelia Dwi Safira

2025-02-03

Library (installation and) preparation

```
# Library
## Uncomment install.packages() to install
# install.packages("ggfortify")
# install.packages("xts")
# install.packages("stats")
# install.packages("tidyr")
# install.packages("ggplot2")
# install.packages("reshape2")
# install.packages("car")
# install.packages("dplyr")
# install.packages("lubridate")
# install.packages("purrr")
# install.packages("sf")
# install.packages("zoo")
# install.packages("mblm")
library(ggfortify)
```

```
## Loading required package: ggplot2
```

```
library(xts)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

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

```
library(stats)
library(tidyr)
library(ggplot2)
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
```

```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```r
library(car)
```

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

```r
library(dplyr)
```

```
##
## ######################### 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: 'dplyr'
```

```
## The following object is masked from 'package:car':
##
##     recode
```

```
## The following objects are masked from 'package:xts':
##
##     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(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

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

```r
library(purrr)
```

```
##
## Attaching package: 'purrr'
```

```
## The following object is masked from 'package:car':
##
##     some
```

```r
library(sf)
```

```
## Linking to GEOS 3.9.3, GDAL 3.5.2, PROJ 8.2.1; sf_use_s2() is TRUE
```

```r
library(zoo)
library(mblm)
```

Calling the CSV data

```r
# Set directory
setwd("D:\\01. RIZKA\\Repo__GitHub\\1_STL")

# Call hydrology data
## Data is in CSV
grace_gsfc <- read.csv(file = "GRACE-GSFC_2002.04_2017.06.csv", sep=",")

## View data structure
str(grace_gsfc)
```

```
## 'data.frame':    32025 obs. of  4 variables:
##  $ lon   : num  109 109 109 109 109 ...
##  $ lat   : num  -0.75 -0.75 -0.75 -0.75 -0.75 -0.75 -0.75 -0.75 -0.75 -0.75 ...
##  $ time  : chr  "2002-04-01" "2002-05-01" "2002-06-01" "2002-07-01" ...
##  $ lwe_cm: num  -6.26 -5.22 -7.8 -10.37 -12.95 ...
```

```r
## Convert 'time' column into Date format
grace_gsfc$time <- as.Date(grace_gsfc$time)

## View first 5 rows and last 5 rows of data
head(grace_gsfc)
```

```
##      lon   lat       time     lwe_cm
## 1 109.25 -0.75 2002-04-01  -6.255509
## 2 109.25 -0.75 2002-05-01  -5.221497
## 3 109.25 -0.75 2002-06-01  -7.797912
## 4 109.25 -0.75 2002-07-01 -10.374328
## 5 109.25 -0.75 2002-08-01 -12.950744
## 6 109.25 -0.75 2002-09-01 -12.522039
```

```r
tail(grace_gsfc)
```

```
##          lon  lat       time    lwe_cm
## 32020 118.25 1.25 2017-01-01  6.039609
## 32021 118.25 1.25 2017-02-01  7.957205
## 32022 118.25 1.25 2017-03-01  9.874801
## 32023 118.25 1.25 2017-04-01 13.064045
## 32024 118.25 1.25 2017-05-01 16.922214
## 32025 118.25 1.25 2017-06-01 10.555866
```

Creating a time series object for each grid

```r
## Find unique grids (longitude-latitude pairs)
unique_coords <- unique(coredata(grace_gsfc)[,c("lon","lat")])
nrow(unique_coords)
```

```
## [1] 175
```

```r
## Create an empty list to store the results of time-series object of each grid
ts_list <- list()

## Create time-series object for each grid
for (i in 1:nrow(unique_coords)) {

  # Read unique longitude-latitude pairs
  lon_val <- unique_coords[i, "lon"]
  lat_val <- unique_coords[i, "lat"]

  # Create grid data from DataFrame
  grid_data <- grace_gsfc[grace_gsfc$lon == lon_val & grace_gsfc$lat == lat_val, ]

  # Create time-series object for each grid
  StartYear <- 2002 # Adjustable
  StartMonth <- 4
  ts_data <- ts(data = coredata(grid_data[, 4]),
                # 4 is the column of variable that will be decomposed
                start = c(2002, 4),
                frequency = 12)

  # Store time-series object
  ts_list[[paste(lon_val, lat_val, sep = "_")]] <- ts_data
}
```

```r
## Example of time-series object format of the first longitude-latitude pair from the ts_list
ts_list[1]
```

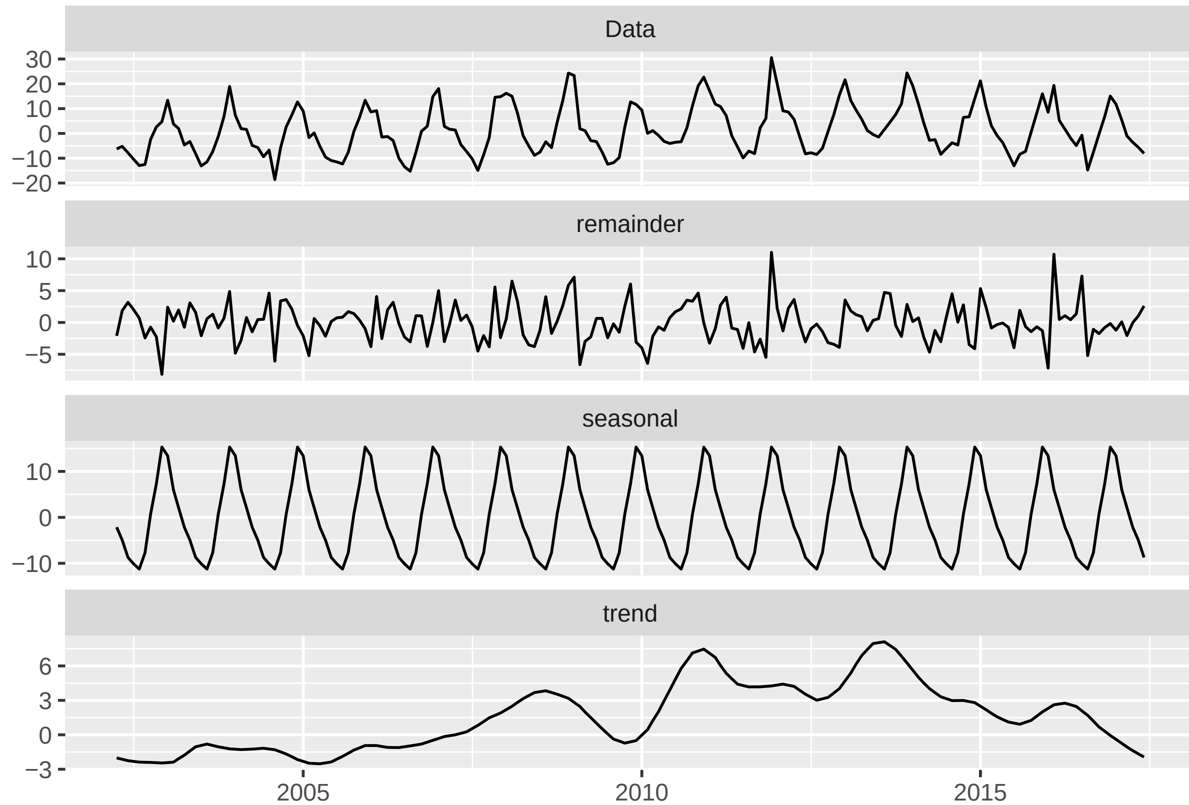
```
## $`109.25_-0.75`
##             Jan        Feb        Mar        Apr        May        Jun
## 2002                                  -6.2555095 -5.2214969 -7.7979125
## 2003 13.3884722  3.8306150  1.9756148 -4.6717127 -3.2659409 -8.1897882
## 2004  7.3242279  1.9378823  1.5981160 -4.8561237 -5.6875559 -9.3986277
## 2005  9.0008650 -1.6789828  0.2159528 -5.1762996 -9.5535129 -10.9361427
## 2006  8.6618104  9.1675129 -1.4702910 -1.2530214 -2.8682789 -10.0246926
## 2007 18.0892017  2.8547649  1.6966446  1.3788447 -4.4918047 -7.2983129
## 2008 16.1840863 15.0032386  8.2652265 -0.9455854 -5.0365515 -8.8263074
## 2009 23.3617223  1.8651460  1.1088357 -2.9209778 -3.2917802 -7.5418335
## 2010  9.3973543  0.0391210  1.1511743 -0.8136825 -3.2082397 -4.0513186
## 2011 17.2475791 11.8012426 10.8505806  7.1863530 -0.9414415 -5.4076521
## 2012 19.9683355  9.1157721  8.6372657  5.6985573 -1.2700189 -8.2385952
## 2013 21.6479347 13.2250106  9.4561963  5.6873820  1.1792149 -0.4356732
## 2014 19.1832969 11.7528106  4.3223244 -2.7802432 -2.5099728 -8.4279444
## 2015 21.2279719 10.5159814  3.0900978 -0.9053875 -3.6867972 -8.3693046
## 2016  8.5395003 19.3657103  5.2475029  1.6787239 -1.8900550 -4.8900478
## 2017 11.7996135  5.3834411 -1.0327313 -3.5556382 -5.5774205 -8.0596334
##             Jul        Aug        Sep        Oct        Nov        Dec
## 2002 -10.3743280 -12.9507435 -12.5220390 -2.4550851  2.6265523  4.6956733
## 2003 -13.1136356 -11.4831778 -7.3267330 -1.2356417  6.8567196 18.9455220
## 2004  -6.7188525 -18.6310504 -5.7987423  2.6131181  7.4747299 12.6918309
## 2005 -11.4840341 -12.3119439 -7.6023134  0.7447071  6.5320287 13.3803298
## 2006 -13.4108660 -15.2618636 -7.5157948  0.9116655  2.9082178 14.8114756
## 2007 -10.2196858 -14.9314414 -8.6003424 -1.6701088 14.5692952 14.8121003
## 2008  -7.5528178  -3.3705877 -5.7352550  4.3779299 13.3213992 24.3239128
```

```
## 2009 -12.4228155 -11.8342826  -9.7802245   2.4934931  12.7281230  11.7012701
## 2010  -3.5648329  -3.3439241   2.2634415  11.1507316  19.2147633  22.6939157
## 2011  -9.8738628  -7.1134441  -8.1521278   2.2401633   6.0434894  30.5790607
## 2012  -7.8117416  -8.4873302  -6.0044738   0.7562274   7.5169287  15.4118864
## 2013  -1.4604121   1.5775491   4.6155103   7.6534714  11.9515553  24.3993822
## 2014  -6.0973872  -3.7668301  -4.6758347   6.4445958   6.7249679  13.9764699
## 2015 -13.0518120  -8.4151754  -7.2582859   0.4938462   8.2459783  15.9981103
## 2016  -0.7179027 -14.7675582  -7.5733504  -0.3791426   6.8150652  15.0708873
## 2017
```

Performing STL for each grid

```r
## Create an empty list to store the results of STL decomposition of each grid
stl_list <- list()

# STL decomposition
for (key in names(ts_list)) {

  # Extract ts object from each grid (key)
  ts_data <- ts_list[[key]]

  # Decomposition
  fit <- stl(ts_data, s.window = "periodic")

  # Store the results of decomposition
  stl_list[[key]] <- fit
}
```

```r
## Example of a visualization of decomposed signals from the first lon-lat pair in the stl_list
autoplot(object = stl_list[1],ncol = 1)
```



Converting the decomposed signal into a DataFrame

```
## Extract decomposition results
for (key in names(stl_list)) {

  # Call each grid's decomposed signal
  fit <- stl_list[[key]]

  # Extract each component
  trend <- fit$time.series[, "trend"]
  seasonal <- fit$time.series[, "seasonal"]
  remainder <- fit$time.series[, "remainder"]
}

## Create an empty list to store the DataFrame
df_list <- list()

## Convert to DataFrame
for (key in names(stl_list)) {

  # Set key column from longitude-latitude
  lon_lat <- unlist(strsplit(key, "_"))
  lon <- as.numeric(lon_lat[1])
  lat <- as.numeric(lon_lat[2])
```

```r
  # Call each grid's decomposed signal
  fit <- stl_list[[key]]

  # Create time index
  time_index <- as.Date(as.yearmon(time(fit$time.series)))

  # Create a DataFrame
  temp_df <- data.frame(
    time = time_index,
    lon = lon,
    lat = lat,
    trend = as.numeric(fit$time.series[, "trend"]),
    seasonal = as.numeric(fit$time.series[, "seasonal"]),
    remainder = as.numeric(fit$time.series[, "remainder"])
  )

  # Append the dataframe to the list
  df_list[[key]] <- temp_df
}

# Combine decomposed signals with the original hydrology signal ----
## Merge all lists in df_list into a DataFrame
final_df <- do.call(rbind, df_list)

## Combine
grace_gsfc_decomposed <- merge(final_df, grace_gsfc, by=c("lon","lat","time"))
```

```r
## View first 5 rows and last 5 rows of grace_gsfc_decomposed
head(grace_gsfc_decomposed)
```

```
##      lon   lat       time    trend   seasonal  remainder    lwe_cm
## 1 109.25 -0.25 2002-04-01 9.412696  0.9653845 -0.9659498  9.412131
## 2 109.25 -0.25 2002-05-01 9.331558  0.7222551 10.6568410 20.710654
## 3 109.25 -0.25 2002-06-01 9.250420 -1.5084155  5.5282621 13.270267
## 4 109.25 -0.25 2002-07-01 9.281949 -1.6961597 -1.7559103  5.829879
## 5 109.25 -0.25 2002-08-01 9.313478 -5.4949519 -5.4290347 -1.610508
## 6 109.25 -0.25 2002-09-01 9.365214 -4.0556143 -9.4908229 -4.181223
```

```r
tail(grace_gsfc_decomposed)
```

```
##          lon  lat       time     trend   seasonal  remainder    lwe_cm
## 32020 118.25 1.25 2017-01-01  9.724605 -0.2609145 -3.4240811  6.039609
## 32021 118.25 1.25 2017-02-01 10.606387 -2.2503656 -0.3988164  7.957205
## 32022 118.25 1.25 2017-03-01 11.416182 -3.1482629  1.6068819  9.874801
## 32023 118.25 1.25 2017-04-01 12.225977  0.1848939  0.6531735 13.064045
## 32024 118.25 1.25 2017-05-01 12.966687  2.8268903  1.1286370 16.922214
## 32025 118.25 1.25 2017-06-01 13.707397  2.2770409 -5.4285718 10.555866
```