Usage of the ttprocessing package

J. P. Kabala

2022-11-17

ttprocessing R package

Version 0.2.0 (Pantelleria edition)

This pdf file is meant to be a tutorial and to provide documentation about the usage of the ttprocessing package, in its version released after the Pantelleria meeting, where new guidelines for handling the data have been provided.

Installation

The package is available on GitHub, and can be installed by running the following code:

```
library(devtools)
install_github("https://github.com/jpkabala96/ttprocessing")
```

Load the package for using it

To load the package in R, for using it, use the following command:

```
library(ttprocessing)
```

Load the data

For loading the data from one server use the *readTTData* function.

```
dati <- readTTData("link to the server")
```

Data from different servers can be merged together with the *rbind* command.

```
data1 <- readTTData("link to server 1")
data2 <- readTTData("link to server 2")
all_data <- rbind(data1, data2)
```

Data can also be loaded from local csv file, by supplying as parameter to the readTTData function, the path to the local file

```
all_data <- readTTData("path to the data file.csv")
```

Clean and convert the data

4D data are processed by the *clean4DData* function. This function accepts several parameters, most of them are thresholds for removing invalid values: for example *lower.TTree* is the minimum value of Tree Temperature regarded as reasonable by the user. There are similar thresholds that can be specified for Relative humidity, air temperature, sap flux etc. There is a *tz* parameter for specifying the time zone of the

TreeTalkers site, and a *species* parameter, that is needed to calculate the Volumetric Water Content of the wood with the proper, species specific equation following Asgharinia et al. (2022).

```
clean_data <- clean4DData(all_data)
```

Variables computed

id = Device id Theat0, 1 e Tref0 e 1 = temperatures of the sap flux probes Theat10: temperature of the heating probe for time 10 without heating, estimated by linear interpolation from Theat0 and Theat0 of the following hour. DT = deltaT: Theat1 - Theat10 DTmax_24h = deltaTmax calculated over the 24 hours K = K index for calculating sap flux

do_sap_flow = Sap flux calculated with the Do et al. (2011) equation. asgharinia_sap_flow = Sap flux calculated according to Asgharinia et al. (2022) Sap flux is expressed in the current version as g/(m^2*s) Tair = air temperature in degrees Celsius Rh = relative humidity % vpd = vpd in KiloPascal date = date voltage = Battery voltage level. For working good the device needs at least 3500 mV growth_DN = Raw digital number provided by the growth sensor. date_hour = timestamp converted as date hour, and rounded to hour. month = month as numeric variable f_month = month as character (same exist for year, hour etc.)

H24 cleaning

After the removal of invalid values, another step of data cleaning is that of removing for each device, the data of dates when there are invalid or missing data. This because the presence of invalid values might be an indication of the fact that the sensor is not reliable on that day, and so data of that day shall not be considered. The function that does this is *h24Clean* that accepts the following args: i) clean 4D data, output by the **clean4DData** function ii) the names of the columns to be cleaned, as strings: e.g. "do_sap_flow", or: c("do_sap_flow", "vpd", "RH"); iii) The number of valid observations in the day (24 hours) required to keep the data of that device.

To retrieve all the column names of a data.frame the **colnames** function is very helpful.

```
colnames(dati_clean)
dati_clean <- h24Clean(dati_clean,
                       cols_to_clean = c("do_sap_flow","vpd"),
                       complete_required = 24)
```

Filtering the data

In the package, several filtering (relational database operation) functions are provided. Those are mainly wrappers around the *dplyr* filter function, which simplyfie the operations that are usually needed for the TT+ data. The database can be filtered by date, by ID, or by hour.

```
dati2021 <- dati_clean %>%
  filterByDate(start = "2021-01-01", end = "2021-12-31") %>%
  filterIDs(ids = c("seriale1id", "seriale2id", "seriale3id"))
```

Exploratory plots

There are several plotting functions that allow the user to make exploratory plots.

Time series plot

The function **plotTS** makes timeseries plots of the data. A summary statistic can be plotted, by setting the statistic parameter of the function to "mean" or "median" or all the devices can be plotted (each in a different color) by setting the statistic param to "none". The **plotWithShade** function is very similar, but adds a shade on the night hours, helping the user in interpreting the values plotted. It is optimal for short timeseries (e.g. one week). The down and dusk times are calculated with the **suncalc** library, so site latitude and longitude must be provided.

```
plotTS(dati2021, variable = "do_sap_flow", statistic = "none")
plotWithShade(dati2021 %>%
                filterByDate(start = "2021-05-01",
                             end = "2021-05-07"),
              variable = "do_sap_flow", statistic = "median",
              lat = 40, lon = 14)
```

The plots are ggplot objects, so further ggplot layers can be added to them to customize them, and they can be saved with the *ggsave* function.

```
library(ggplot2)
plotTS(dati2021 %>%
         filterByDate(start = "2021-05-01", end = "2021-05-30"),
       variable = "do_sap_flow",
       statistic = "median")+
  ggtitle("Serie temporale maggio")+
  ylab("do sap flow maggio (g/(m^2*s))")+
  theme_bw()
```

For example, by plotting the *"voltage"* variable, the batteries status can be easily inspected. In that case one shall set: statistic = *"none"*.

```
library(ggplot2)
plotTS(dati2021 %>%
         filterByDate(start = "2022-10-01", end = "2021-11-30"),
       variable = "voltage",
       statistic = "none")+
  ggtitle("Voltaggio batterie")+
  ylab("battery voltage [mV]")+
  theme_bw()+
  geom_hline(yintercept = 3500) #aggiungo una linea rossa orizzonate a 3500
#per aiutarmi a capire quando sono scariche le batterie
```

Plots with seasonal summaries

The functions for plots with summaries by hour and by month are **plotByMH**, and for summaries by quarter and hour **plotByQuarter**. In both the variable and the summary statistic have to be set.

```
plotByMH(dati2021, variable = "do_sap_flow", statistic = "median")
plotByQuarter(dati2021, variable = "Tair", statistic = "mode")
```

Save data for external use

Data can be saved with the **readr** *write_csv* function or the **base** *write.csv* or alternatively the *write_csv2* for separating them with semicolons: ";", with comma as decimal separator.

```
write.csv(dati2021, file = "Dati_2021.csv", row.names = F)
#oppure
library(readr)
write_csv(dati2021, file = "Dati_2021.csv")
```

Growth processing

For processing the growth there exist the dedicated function: *growthElaboration* that takes as input the 4D data returned by *clean4DData* that contain the growth Digital Numbers. This function, works by aggregating the data according to the parameters *nweeks,* specified by the user. Data aggregated for that time window are summarized with the modal value, to have a representative estimate of the growth at that time. The dataframe in output contains several columns: time indication (central_date); the raw modal value, the converted value (as distance from the trunk) and the **growth__s0**. As input data of only one vegetative season shall be provided. This because winter data contain mainly noise and disturbance. The **growth__s0** is the absolute growth: the decrement in distance of the sensor from the trunk since the first measure obtained during the time period to which the data supplied to the function belong. For easily plotting growth there is the *plotGrowth* function. The function works well when integrated with the before described functions.

One can change the date of start and end of the vegetative season.

```r
stagione_vegetativa_2021 <- filterByDate(dati2021,
                                         start = "2021-04-01",
                                         end = "2021-11-01")
#aggrego per 1 settimana, ma se voglio fare bisettimanale metto 2 al posto di 1
growth2021 <- elaborateGrowth(stagione_vegetativa_2021, nweeks = 1)
plotGrowth(growth2021)
```

Changing the *plotGrowth* as for example the *variable* one, it is possible to plot also the raw data or the absolute growth: **growth__s0**. The data of the growth table can be furtherly handled with tidyverse and similar.

Per esempio:

```r
media_di_tutti <- growth2021 %>%
  group_by(ftp) %>%
  summarise(crescita_media = mean(growth_s0, na.rm = T),
            central_date = median(central_date, na.rm = T))
```

For example I can add a layer to the before mentioned plot.

```r
grafico_modificato <- plotGrowth(stagione_vegetativa_2021, nweeks = 1)+
  geom_line(aes(x = central_date,  y = crescita_media), color ="black")
print(grafico_modificato)
```

The growth data can be saved the same way as the others.

Device functioning assessment

This function enables the user to check if a device records a minimum number of valid records with the different sensors. As input the 4D data have to be provided. The *min__valid__obs* parameter allows the user to set the minimum of valid values he requires.

```r
deviceFunctioningAssessment(dati_clean,
                            id = "numeroserialedispositivo",
                            min_valid_obs = 24)
```

A big ensemble plot can be made with plotDeviveFunctioning, and can be printed on a pdf file.

```r
grafico_funzionamento <- plotDeviceFunctioning(dati_clean %>%
                                                 filterByDate(start = "2022-09-01", end =  "2022-11-10"
                                               ids = c("seriale1", "seriale2","seriale3"), min_valid_ob
                                               )
ggsave(plot = grafico_funzionamento,
       filename = "Grafico_funzionamento2022.png",
       units = "in",
       width = 8, height = 20)
#posso settare la larghezza e l'altezza del documento
#in base a quanti dispositivi ho per farli entrare bene e farli uscire delle giuste dimensioni.
```

In the working directory of the R session the pdf file with the plots shall have appeared.

Launch the GUI

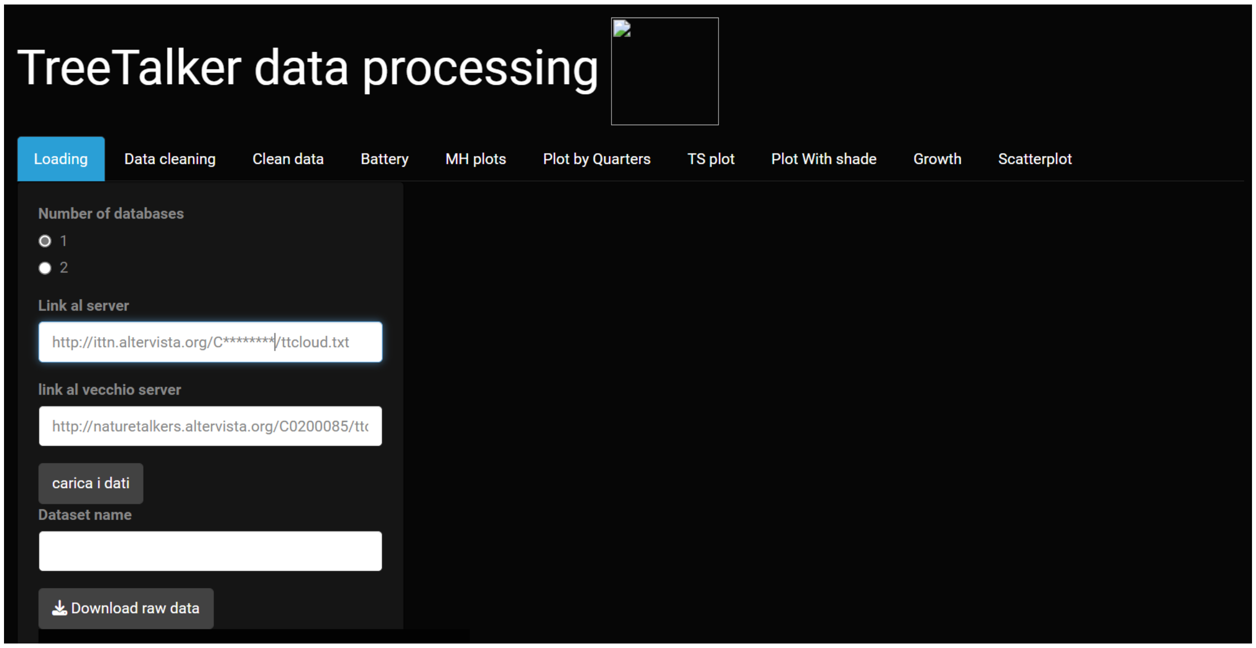
The GUI has most of the functionalities described above. The arguments of the different functions can be specified in the various tabs.

The GUI can be launched with the **launchGUI** function of the package.

```
launchGUI()
```

Tabs and features of the GUI

The first tab enables the user to load the data from the server, or from some local file, by providing the link or the file path and pressing “Carica i dati”. When the data are loaded, they appear on the right of the sidebar.



Number of databases
● 1
○ 2

Link al server
http://ittn.altervista.org/C********/ttcloud.txt

link al vecchio server
http://naturetalkers.altervista.org/C0200085/ttc

Dataset name

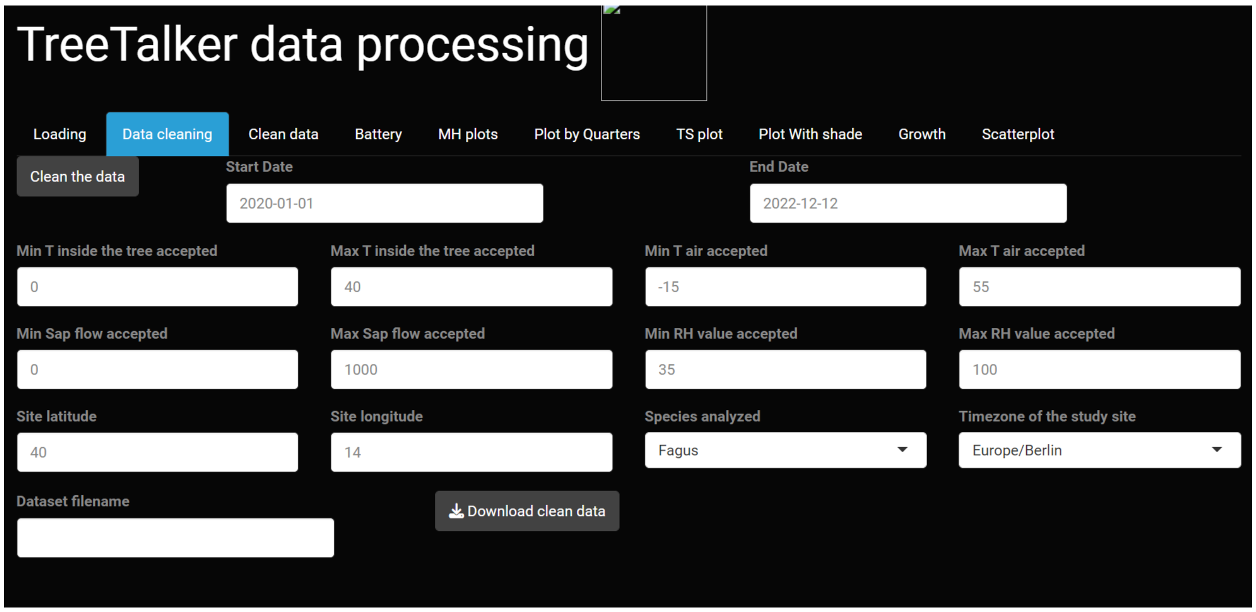
Figure 1: Data loading tab

The next tab is the Data cleaning one. The user can supply all the parameters that can be specified for the **clean4DData** function, and then press the “Clean the data” button.

When the procedure is finished, the data are displayed in the “Clean data” tab. The processed data can be downloaded by entering a filename in the dataset filename field, and pressing the “Download the clean data” button.

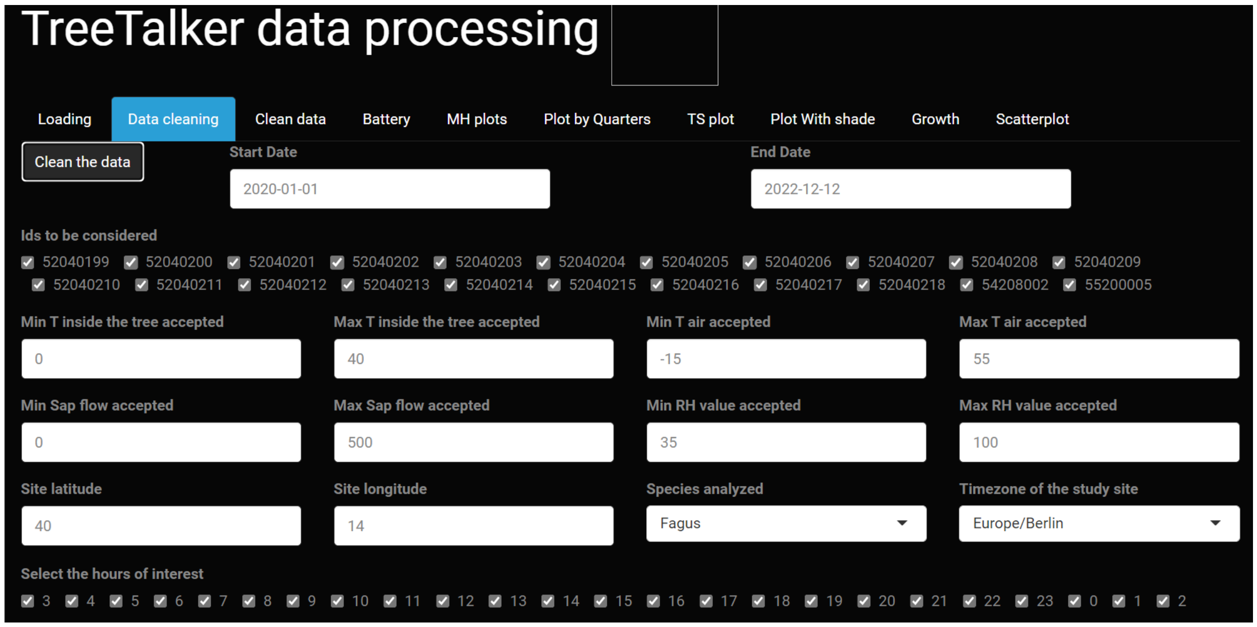
After cleaning the data, they can be displayed with the different plottting tabs. The list of ids and hours of the day appears in the Data cleaning tab, so the user can select/unselect ids he is interested in and eventually specific hours he is interested in.

The plotting tabs allow to select the variable to be plotted, the summary statistic desired, titles, labels and similar. To display the plot according to the settings the user has to press plot. The plot can be saved with



Start Date: 2020-01-01
End Date: 2022-12-12
Min T inside the tree accepted: 0
Max T inside the tree accepted: 40
Min T air accepted: -15
Max T air accepted: 55
Min Sap flow accepted: 0
Max Sap flow accepted: 1000
Min RH value accepted: 35
Max RH value accepted: 100
Site latitude: 40
Site longitude: 14
Species analyzed: Fagus
Timezone of the study site: Europe/Berlin
Dataset filename:

Figure 2: Data cleaning parameters tab



Start Date: 2020-01-01
End Date: 2022-12-12

Ids to be considered

✓ 52040199 ✓ 52040200 ✓ 52040201 ✓ 52040202 ✓ 52040203 ✓ 52040204 ✓ 52040205 ✓ 52040206 ✓ 52040207 ✓ 52040208 ✓ 52040209
✓ 52040210 ✓ 52040211 ✓ 52040212 ✓ 52040213 ✓ 52040214 ✓ 52040215 ✓ 52040216 ✓ 52040217 ✓ 52040218 ✓ 54208002 ✓ 55200005

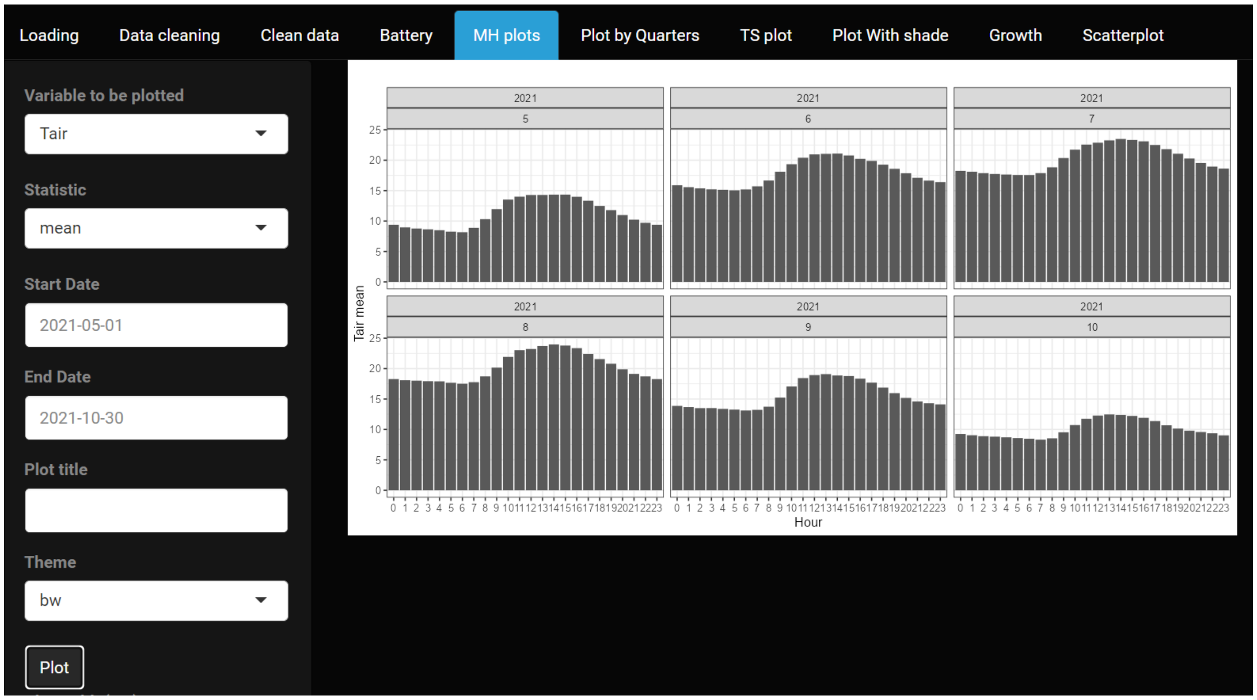
Min T inside the tree accepted: 0
Max T inside the tree accepted: 40
Min T air accepted: -15
Max T air accepted: 55
Min Sap flow accepted: 0
Max Sap flow accepted: 500
Min RH value accepted: 35
Max RH value accepted: 100
Site latitude: 40
Site longitude: 14
Species analyzed: Fagus
Timezone of the study site: Europe/Berlin

Select the hours of interest

✓ 3 ✓ 4 ✓ 5 ✓ 6 ✓ 7 ✓ 8 ✓ 9 ✓ 10 ✓ 11 ✓ 12 ✓ 13 ✓ 14 ✓ 15 ✓ 16 ✓ 17 ✓ 18 ✓ 19 ✓ 20 ✓ 21 ✓ 22 ✓ 23 ✓ 0 ✓ 1 ✓ 2

Figure 3: Ids and hours selection

the download button below. The plot width and height can be specified in cm (the resolution is 300 dpi).



Variable to be plotted: Tair

Statistic: mean

Start Date: 2021-05-01

End Date: 2021-10-30

Plot title:

Theme: bw

Figure 4: Mothly summary plot

There also exist a dedicated tab for processing the growth data, that enable also the download of the processed data as csv file.

Night time delta T max

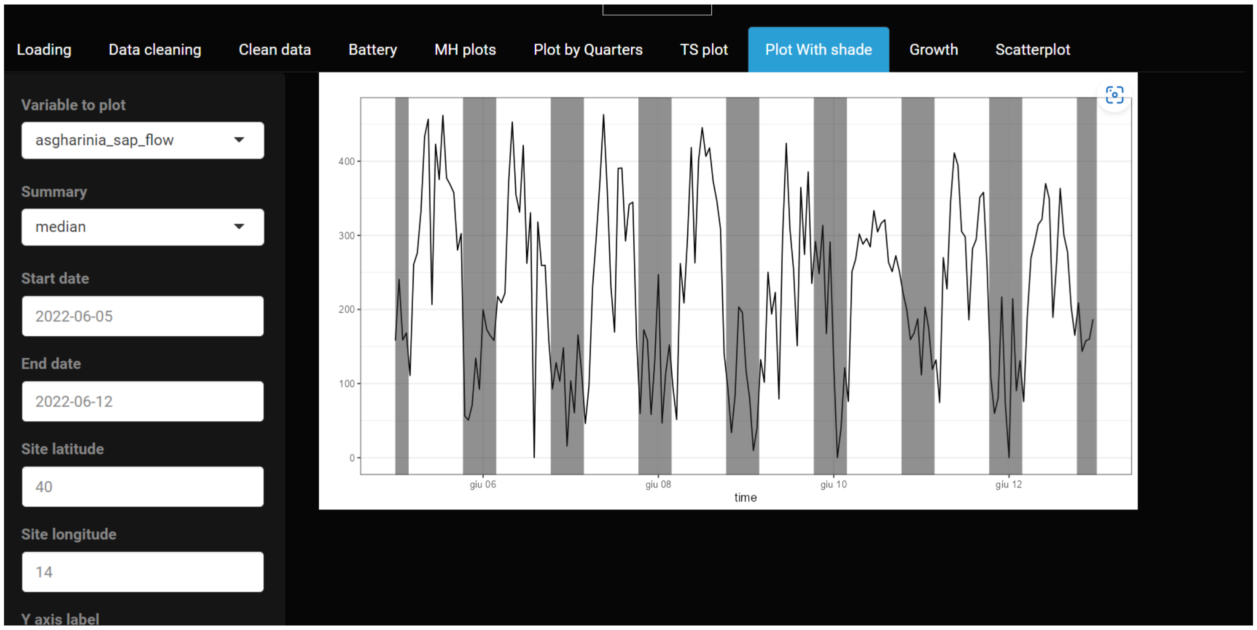
As emerged from the ITTN meeting in Pantelleria, in November 2022, a good option might be to force delta T max estimation during nighttime. For this aim, the **clean4DDataNight** and the **launchGUINight** functions have been developped. Those functions are clones of the original ones above mentioned, but they choose the delta T max from the delta T values recorded during the night. For selecting night time, they rely on the **suncalc** library, which calculates the down and dusk time, given the site coordinates and the date. So the **clean4DDataNight** function accepts 2 more params: **lat** and **lon**: the latitude and longitude of the site. The function replaces negative sap flux values, estimated when delta T is higher than delta T max, (when the real delta T max of the day does not occour during the night) with 0. For the other features of those functions, they work exactly the same as the ones described above.

Make customized plots

The TT+ data can be easily plotted with ggplot, as any other data, here there is an example.

```
ggplot(dati2021)+
  geom_point(aes(x = Tref0, y = Tref1))
```

The same way, any kind of plot can be done.



Variable to plot: asgharinia_sap_flow

Summary: median

Start date: 2022-06-05

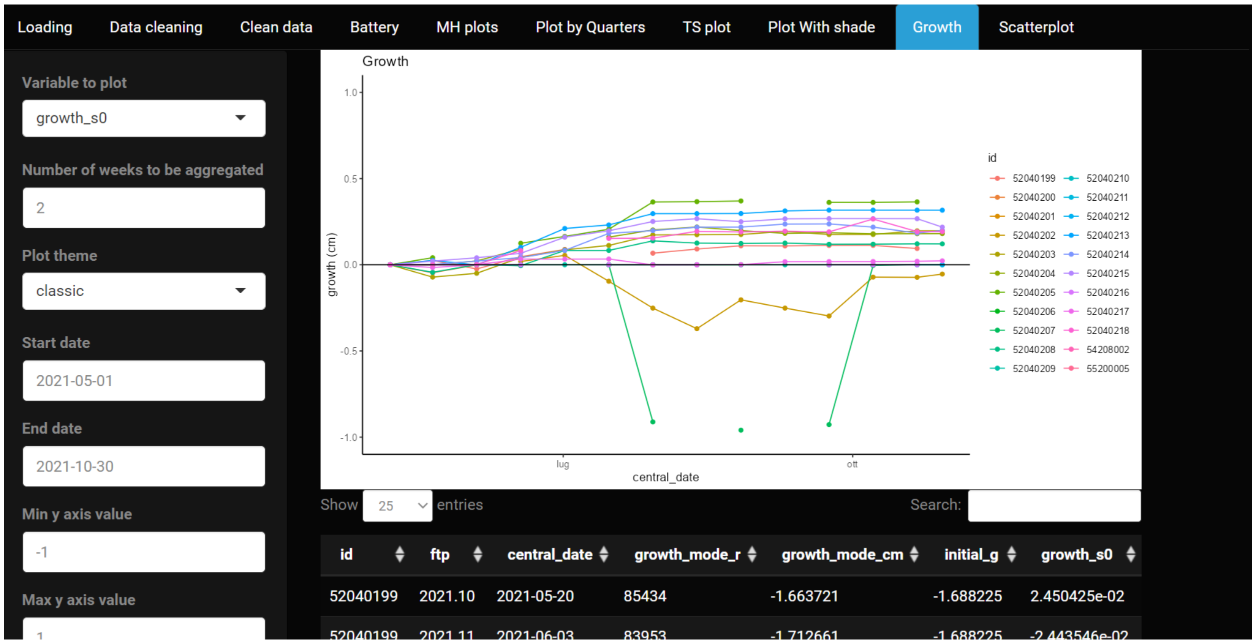
End date: 2022-06-12

Site latitude: 40

Site longitude: 14

Y axis label

Figure 5: Plot with shade on nighttime



Variable to plot: growth_s0

Number of weeks to be aggregated: 2

Plot theme: classic

Start date: 2021-05-01

End date: 2021-10-30

Min y axis value: -1

Max y axis value: 1

id	ftp	central_date	growth_mode_r	growth_mode_cm	initial_g	growth_s0
52040199	2021.10	2021-05-20	85434	-1.663721	-1.688225	2.450425e-02
52040199	2021.11	2021-06-03	83953	-1.712661	-1.688225	-2.443546e-02

Figure 6: Growth processing tab