Manual tutorial for the cleaning of continuous biological parameters from the Flemish Hydrological Information Center (HIC) databases

Funded by De Vlaamse Waterweg (Flemish Waterways)

For the Hydrological Information Center (HIC) of Flanders Hydraulics Research

Written and developed by Pali Felice Gelsomini with support from Tom Maris
Ecosystem Management Research Group (ECOBE) University of Antwerp

Contact palifelice.gelsomini@uantwerpen.be

August 21, 2020

Contents

Data cleaning, validation and calibration methodology

The cleaning and validation methodology was adapted from the methodology utilized by the Hydrological Information Center (HIC) for processing continuous hydrological data. The one major difference is we used median and median absolute deviations (MAD), instead of mean and standard deviation, as per recommendation of the HIC. The chlorophyll-a florescence measurements additionally must be post-calibrated using lab-tested point samples and is done following the protocol given by the sensor manufacturer YSI.

There are two steps to the data cleaning and validation process: first an automated spike removal and second a manual check. Following these two steps, the data is post calibrated if need be (e.g. the chlorophyll-a florescence must be post calibrated).

The R package "HICbioclean" was developed to automate this process. See that section for a tutorial and more details.

Continuous monitoring stations, reference site and sensor maintenance data

For the validation and calibration, the closest available point-sampled water-quality monitoring stations from OMES were selected as reference sites (see figure 1 and table 1 to see which point sampled site is reference to with which continuous site, for the location of these sites and the distances between the reference and continuous sites). The OMES monitoring campaign is funded by the Flemish Waterways. The water-quality monitoring stations used as references are sampled biweekly during the growing season April to September and monthly the rest of the year. Sensor maintenance data on cleaning and sensor changes was provided by the HIC and was used during the manual validation procedure to asses miscalibrations and sensor drifts (see table 1 for the file names and to which continuous monitored sites they belong).

Figure 1: Map of continuous monitoring sites and their respective point sampled reference sites which were used for validation and calibration of the continuous data. The point sampled site Grens was evaluated as a possible reference site for Lillo, but was not chosen. The other available point sampled sites which were not used as reference sites are displayed but are not labeled.

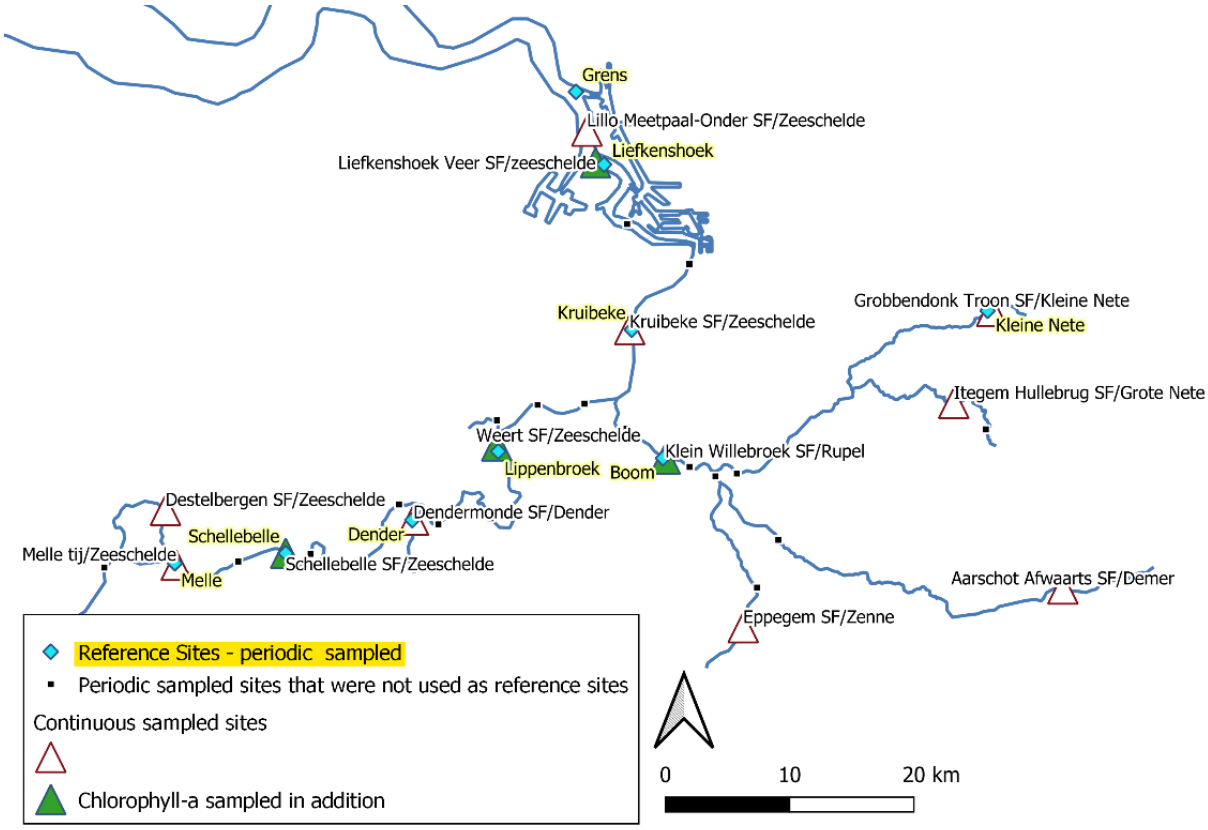


Table 1: Metadata for continuously measured biological data cleaning, calibration and validation. Continuous monitoring site names, location information, site number, the continuously measured biological parameters at each site that were cleaned and validated, the respective file for sensor maintenance information on cleanings and sensor changes, the point sampled site from the OMES campaign which was used for validating and calibrating the continuous data, the distance between the continuous measurement site and the point sampled reference site.

Continuous data site name	Site number	river	km from mouth	2018 data	2019 data	Sensor maintenance file	Point sampled reference site from OMES	Distance from reference site (m)
Aarschot Afwaarts SF/Demer	dem02a-SF-CM	Demer	142	DO, pH	DO, pH	WISKI_MPS_Aarschot_		
Dendermonde SF/Dender	den02a-SF-CM	Dender	125	DO, pH	DO, pH	WISKI_MPS_Appels_	Dender	0
Itegem Hullebrug SF/Grote Nete	gnt03a-SF-CM	Grote Nete	118	DO, pH	DO, pH	WISKI_MPS_Itegem Hullebrug_		
Grobbendonk Troon SF/Kleine Nete	knt03a-SF-CM	Kleine Nete	119	DO, pH	DO, pH	WISKI_MPS_Grobbendonk_	Kleine Nete	345
Klein Willebroek SF/Rupel	rup02e-SF-CM	Rupel	99	DO, pH, chl-a	DO, pH, chl-a, PPFD	WISKI_MPS_Boom_	Boom	193
Eppegem SF/Zenne	zen03a-SF-CM	Zenne	116	DO, pH	DO, pH	WISKI_MPS_Eppegem_		
Lillo Meetpaal-Onder SF/Zeeschelde	zes07g-SF-CMO	Zeeschelde	60	DO	DO	WISKI_MPS_Gavere_	Liefkenshoek	3032
Liefkenshoek Veer SF/zeeschelde	zes09x-SF-CM	Zeeschelde	63		DO, pH, chl-a, PPFD	WISKI_MPS_Liefkenshoek_	Liefkenshoek	698
Kruibeke SF/Zeeschelde	zes24a-SF-CM	Zeeschelde	85	DO, pH	DO, pH	WISKI_MPS_Kruibekeveer_	Kruibeke	163
Weert SF/Zeeschelde	zes39c-SF-CM	Zeeschelde	103	DO, pH, chl-a	DO, pH, chl-a, PPFD	WISKI_MPS_Weert_	Lippenbroek	462
Lippenbroek UIT SF/Zeeschelde	zes40a-SF-CM	Zeeschelde	104	DO, pH	DO, pH	WISKI_MPS_LippenbroekUIT_		
Schellebelle SF/Zeeschelde	zes54m-SF-CM	Zeeschelde	140	DO, pH, chl-a	DO, pH, chl-a, PPFD	WISKI_MPS_Schellebelle_	Schellebelle	0
Melle SF/Zeeschelde	zes57a-SF-CM	Zeeschelde	150	DO, pH	DO, pH	WISKI_MPS_Melle_	Melle	185
Destelbergen SF/Zeeschelde	zes57n-SF-CM	Zeeschelde	153	DO, pH	DO, pH	WISKI_MPS_Zulte_		

Automated spike removal

1. **Min/max filter:** Data was first passed through a min/max filter to remove unreasonably large and small values. 0 was the minimum value for all parameters. Dissolved oxygen and pH had a maximum of 15, chlorophyll a 1000 and PPFD 2000.

2. **Spike removal:** All sample points that were more than 3 MAD (the scale factor 1.4826 was used assuming normal distribution; this is the default in R) from the median of the 10 surrounding data points (5 before and 5 after) are automatically deleted. Median was used because it is a robust statistic, being more resilient to outliers. There needed to be a minimum of 5 surrounding data points, otherwise the point was not evaluated. Given the 5-minute sampling interval of the continuous data, 5 data points before and after was interpreted as 25 minutes before and 25 minutes after; this means that the algorithm would look no farther than 25 minutes, even if data was missing. All evaluated data points that passed the test were flagged as "automatic good".

3. **Gap linear interpolation**: All data less than or equal to 1 hour were interpolated linearly. All interpolated data points were flagged as "automatic good"

The PPFD data has a different auto validation workflow. PPFD data was from paired sensors at a fixed distance from each other for calculating the light attenuation coefficient kd. This means that the two sensors needed to be auto-validated in tandem with each other.

1. **Min/max filter on the PPFD data** for the upper and lower sensors.
2. **Spike removal on the PPFD data** for the upper and lower sensors.
3. Generally both the upper and lower sensors show the same trends with spikes occurring at the same times. If a spike was deleted and then interpolated from one sensor but not the other sensor, then that will lead to an artificially large or small difference between sensors, thus creating an artificially large or kd value. Therefor **if a point was deleted from one sensor, it must be deleted in the other sensor as well.**
4. Many spikes were registered both in the upper and lower sensors. These spikes were most likely not sensor errors, but may have been passing clouds or debris and this information is very important for understanding the total light climate**. All spikes that are registered in both the upper and lower sensors will not be deleted.**
5. **PPFD data is linear interpolated for data gaps of max one hour.**
6. **Light attenuation coefficient kd is calculated.** $kd = 1/\Delta z * \ln(E1/E2)$ where Δz is the distance between sensors in meters (0.4m) and E1 is the upper sensor PPFD and E2 is the lower sensor PPFD.
7. **Delete all PPFD values where kd is negative.** Light cannot be greater when deeper in the water column.
8. **Remove all kd values where PPFD is below the detection limits** (1 µmol/s/m² for the upper sensor and 0.25 µmol/s/m² for the lower sensor). When the light levels approach zero, it becomes too difficult to accurately measure the difference between the upper and lower sensors.
9. **Spike removal on kd. Remove those spikes also from the PPFD data.**
10. **Interpolate PPFD data again for data gaps of max one hour.**
11. **Recalculate kd and remove kd values that are outside the detection limit again.**

Manual validation and calibration

1. **The time series is plotted** along with the reference site values and the sensor maintenance data on cleaning and sensor changes.

2. **The data is then visually evaluated for the following issues:**
   Sudden jumps relating to sensor replacements: There can be sudden shifts in the data following a sensor change due to miscalibrations or sensor drift or jumps relating to sensor start up (particularly an issue with pH where the value right after the sensor is placed out in the field is extremely low and then slowly rises back up over the course of the next day). Sensor miscalibration can be recalibrated linearly either two sided or one sided. The data may be sifted (+ -) and/or scaled (* /), with an attempt to match both the values and the signal amplitude to the previous and following data sections. The reference site values were used for the recalibration. When no reference site is available, then trends seen in the other measured parameters and the surrounding sites can be used as a guide. Recalibrated data is quality flagged as “estimate”. If the recalibration is very untrustworthy, as in the data section still doesn’t match up with the previous and following data or the amplitude of the data section does not match the surrounding data, it is flagged as “suspect”. Issues related to sensor start up (e.g. pH) are simply deleted and quality flagged as “missing”.
   Data noise and sensor error: Some sections of data are simply signal noise and are clearly sensor error, they are deleted and quality flagged as “missing”. Some sections are sensor error (e.g. flatlines), they are deleted and flagged as “missing”. Some sections seem to have an error in the sensor, but useful information may still be derived from the data, they are quality flagged as “suspect”.
   Spikes in very irregular timeseries and prolonged spikes: The automated spike removal is not effective for timeseries with very high variance. Also, the automated spike removal will not remove spikes that are cause by prolonged disturbances such as debris trapped on the sensors. A visual evaluation must be made of the remaining data spikes. If the spikes are cyclical and follow the general data trends then they are left as is. If the spikes severally deviate from the general data trend or are associated with a sensor failure then they are deleted and flagged as “missing”. If it is unclear, then they are flagged as “suspect”.
3. **Gaps created by deleting data during the manual cleaning may be linear-interpolated.** Gaps of up to one hour large may be filled automatically and are quality flagged as “estimate”. If the data gap is in a very simple signal shape and a linear-interpolation of greater than hone hour will not alter the signal shape, then a linear interpolation of greater than one hour may be done, and the points are quality flagged as “estimate”

4. **The chlorophyll-a data is then post-calibrated** after the manual cleaning and validation. Only non-suspect data is used for calculating the calibration. The calibration is calculated using linear regression between the lab-tested reference-site-data and the nearest, continuous-measured florescence-data. A y-intercept of 0 is used for the linear-regression as recommended by YSI, the sensor manufacturer. After the calibration has been applied to the data, no special quality flags are given to the data.

Data quality flagging rule summary:

- Data that is clearly sensor error is deleted and flagged as “missing” and then gaps of up to one hour are interpolated and flagged as “estimate”. If the resulting data gap is in a very simple signal shape and a linear interpolation of greater than hone hour will not alter the signal shape, then a linear interpolation of greater than one hour may be done, and the points are flagged as “estimate”
- Data that looks to be sensor error but may still provide information is flagged as “suspect”
- Data that where the sensor is mis-calibrated and the data must be recalibrated is flagged as “estimate”
- Data that was recalibrated, but the recalibration is very untrustworthy, as in the data section still doesn’t match up with the previous and following data or the amplitude of the data section does not match the surrounding data, is flagged as “suspect”
- When data is post-calibrates such as (such as the chlorophyll-a) then no data flag is applied. Post-calibration is defined as calibration needed for the sensor sampling protocol and it is not used for fixing an incorrectly calibrated sensor.
- At the end of the manual validation and calibration, all data points that are not already flagged as either “estimate” or “suspect” will be flagged as “good”

Final file formatting

As per request of the HIC, the data will be given the state of value codes 11 good measurements, 21 good calculation, 31 estimate measurements, 41 estimate calculations, 61 suspect measurements, 71 suspect calculations. Generally only the values 11 and 61 will be used.

All data that was originally missing will receive their original NA value of -777 and the state of value flag of 255. All data that was deleted during this data cleaning process will receive the NA value of -88888 and the state of value flag 61. The file format will be a .zrx file with the below syntax. The highlight text is the sample location and parameter code.

```
#REXCHANGE1013plu15a-1066VAL|*|RINVAL-777.0|*|
#TZUTC+1|*| CUNITmm|*|
#LAYOUT(timestamp,value,primary_status)|*|
201708231515          0.8  <kwaliteitscode>
201708231520          0.7  <kwaliteitscode>
201708231525          -777.0     <kwaliteitscode>
201708231530          1.6  <kwaliteitscode>
201708231535          0.2  <kwaliteitscode>
201708231540          0.4  <kwaliteitscode>
201708231545          0.6  <kwaliteitscode>
```

R package HICbioclean

An R package specifically designed for automating the process of cleaning, calibrating and validating continuous biological water quality data from the Flemish HIC (Hydrological Information Centre) database.

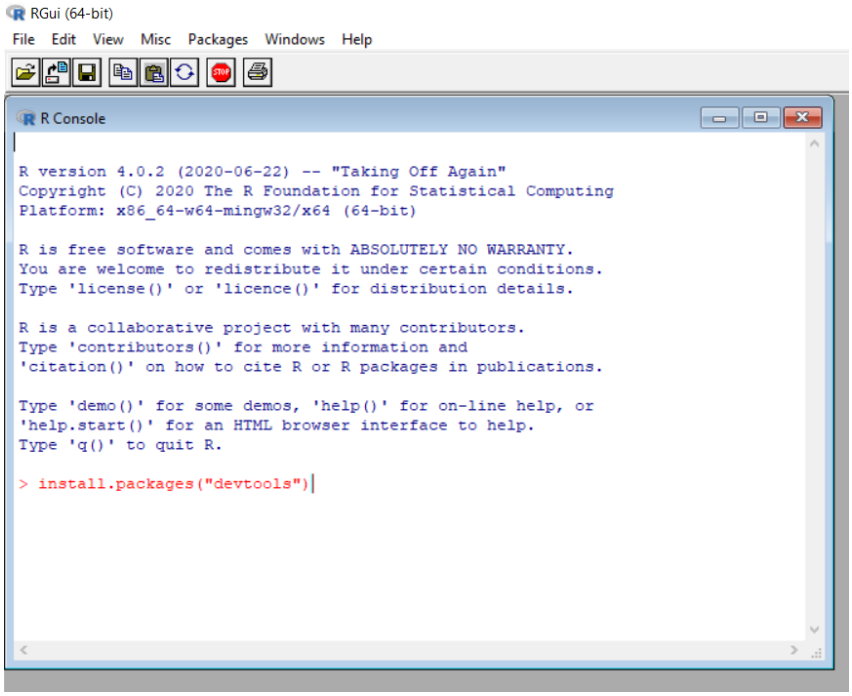
It provides both R functions for integration into R scripts and easy to use R Shiny graphical apps for intuitive data cleaning, validation and calibration without the need for coding.

Installation

Download the latest version of R from https://cran.r-project.org/ if not already installed.

Open the program R.

To download the HICbioclean package from github, you will first have to install the package devtools if you don't already have it. Copy the following code into the R console and press enter and follow the onscreen instructions:
install.packages("devtools")



Install the HICbioclean package into R. Copy the following code into the R console and press enter:
devtools::install_github("pgelsomini/HICbioclean", build_vignettes = TRUE)

Open the package library for HICbioclean. Copy the following code into the R console and press enter:
library(HICbioclean)

Tutorial

The work flow for cleaning the continuous biological water quality data is split into three main steps:

1. Data formatting
2. Auto-validation
3. Manual-validation and final export

Via coding in R

The first two steps can be done with the following code in R.

Place all your HIC database exported text files into one folder. Place all your WISKI maintenance files into one folder.

Open R and enter **library(HICbioclean)** into the R Console and press enter. This loads this package into R.

Enter the below code into the R console replacing the red text with links to your data and press enter.

```
HIC.maint("C:\\Users\\FolderWithYourWISKImaintenanceData",”FormattedMaintData”);
HIC.Continuous.Data.Import.Format(
InputDirectory = "C:\\Users\\FolderWithYourHICdatabaseData",
OutputDirectory = "FormattedHICdata");
dspk.DespikingWorkflow.CSVfileBatchProcess(
input.directory = 'FormattedHICdata',
sep = ',', dec = '.', header = T,
Value = 3, val.NAvalue = -777,
DateTime = "DateTimeUnix",
ConditionalMinMaxColumn = 'Parameter.Name',
ConditionalMinMaxValues = c('DO','pH','chfyla','PPFD1','PPFD'),
ConditionalMin = c(0,0,0,0,0), ConditionalMax = c(15,15,1000,2000,2000),
max.gap = 3600)
```

See the help files at the end of this document for details.

This can all be done using visual R Shiny apps if you’d rather not code.

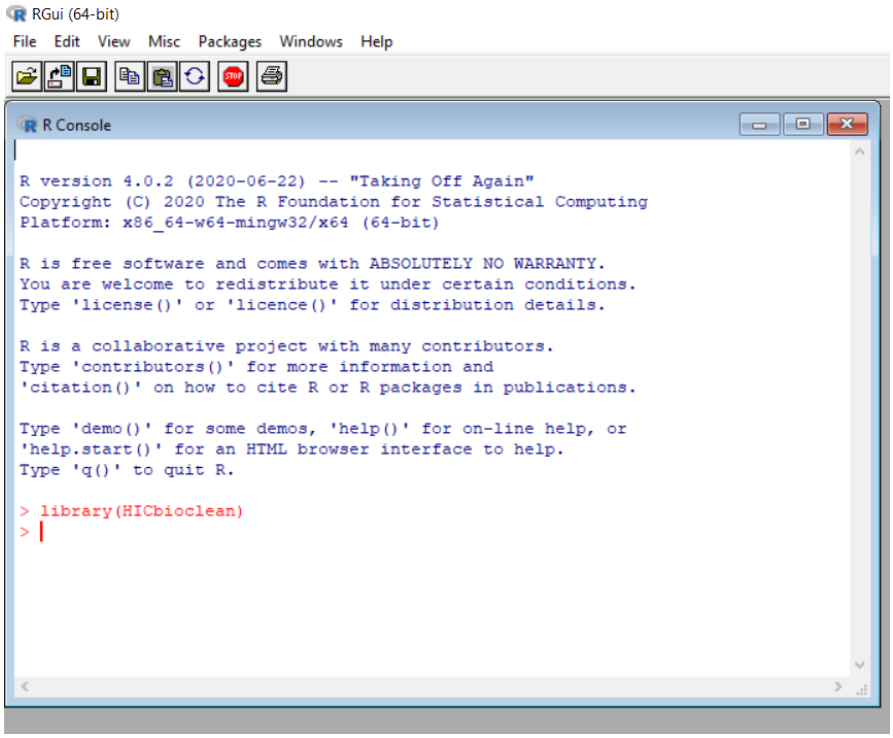
The last step, the manual validation must be done with the R Shiny app because it is a visual inspection.

Via Visual R Shiny Apps (no coding required)

Data formatting

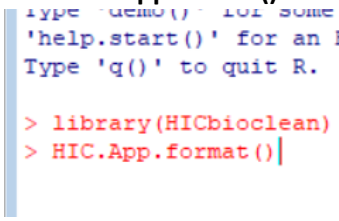
The data must first be formatted because the raw text files exported from the HIC database cannot be loaded into R and used as is. In this step the date and time is also formatted into a datetime column and a numeric datetime column for easier processing in R. The metadata in the header of the text files are placed in columns next to the data. The value column name is renamed “Value”.

1. Open R and enter **library(HICbioclean)** into the R Console and press enter. This loads this package into R.

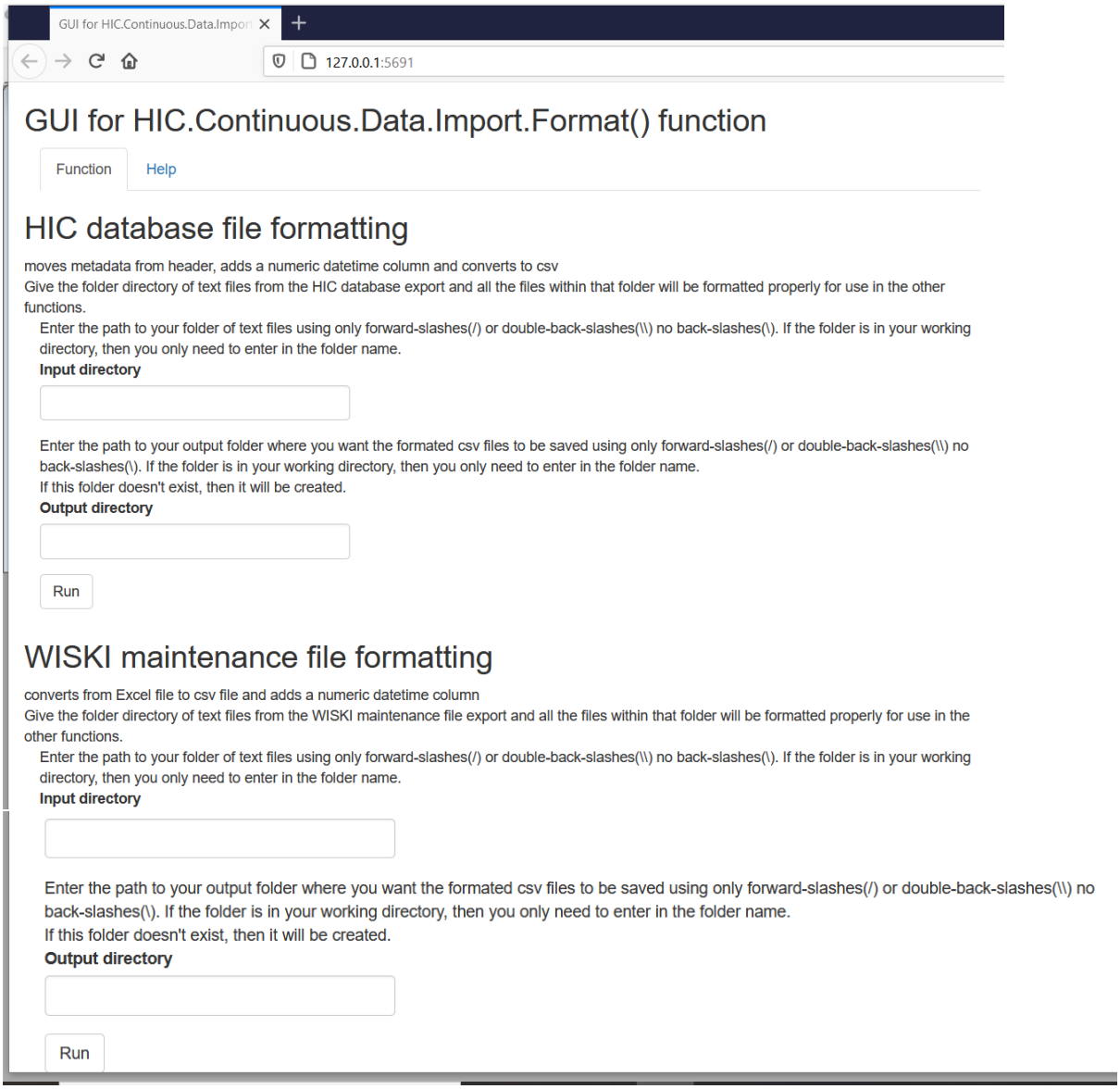


2. Data exported as csv files from the Hydrological Information Center database has the metadata in the header and the date and times are in separate columns. These files cannot be loaded into R because of this. We will use the HIC.App.format to format these files as a batch process so they are ready to work with.

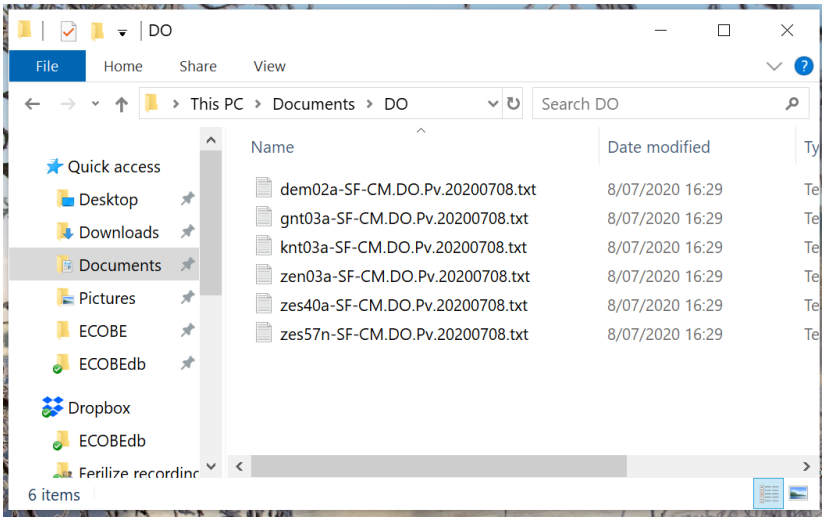
   Enter **HIC.App.format()** into the R Console and press enter



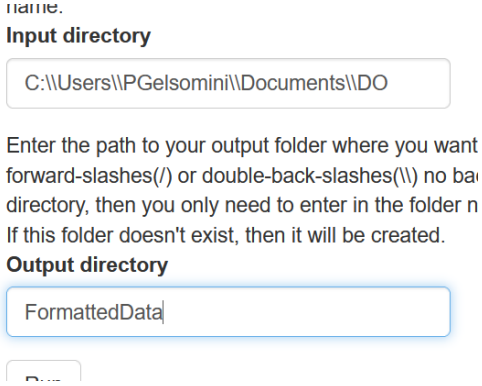
3. The app should have opened in your web browser.



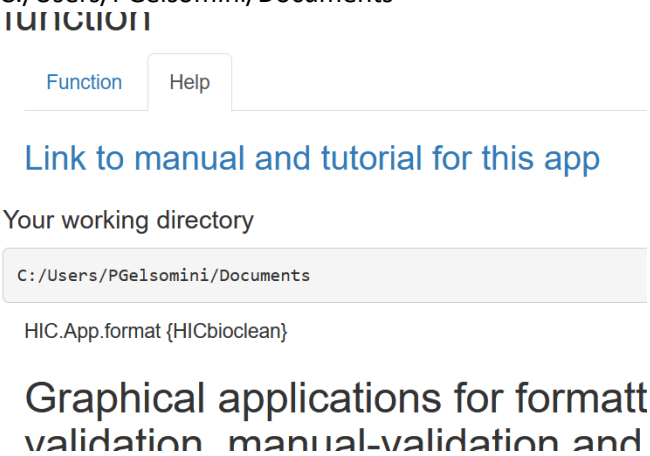
4. Place all the HIC database output text files that you wish to format into one folder.



5. In the field “Input directory” give the folder directory of text files from the HIC database export and all the files within that folder will be formatted properly for use in the other functions. In the field “Output directory” give the folder directory where you would like to save the formatted csv files. Enter the paths to your folders using only forward-slashes(/) or double-back-slashes(\\) no back-slashes(\). If the folder is in your working directory, then you only need to enter in the folder name. If you copy the link in windows it will have back slashes and these need to be fixed.

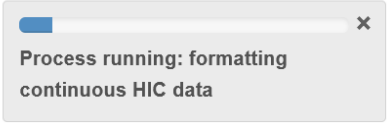


In this example the input directory was copied from the windows file browser and we fixed the back slash (\) issue by adding double back slashes (\\). I entered an incomplete file path for the output directory, so the folder “FormattedData” will be created in your working directory. You can see your working directory in the help tab. My working directory is C:/Users/PGelsomini/Documents

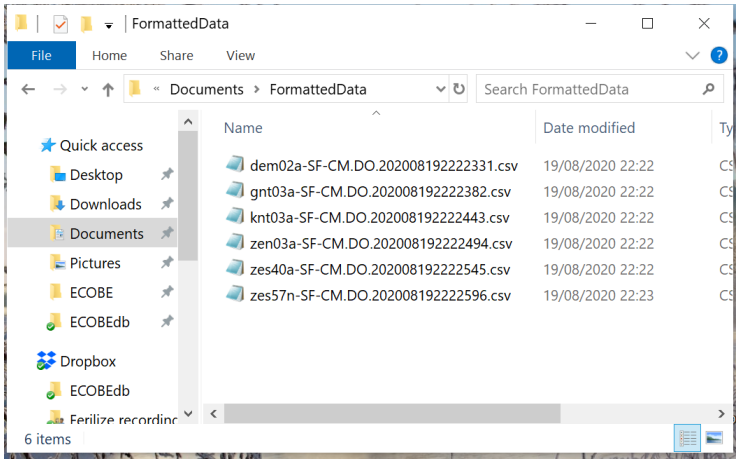


6. Press the run button 

to start the process. While the process is running a little popup will be present in the corner of the window.



7. Now when we go to our working directory C:/Users/PGelsomini/Documents to the folder “FormattedData” all the files are now here as csv documents. The naming is first the station number then the parameter (in this case DO for dissolved oxygen) and the time of export.



8. Now repeat that process but with your WISKI maintenance files. These files are given as Excels that have a header and poor column names for working in R. The header needs to be removed and the column names fixed. This will also convert the datetime into a numeric format which is easier to handle in R.
9. Once all your data is formatted you must close the app window in your web browser before moving onto the next step which is auto-validation.

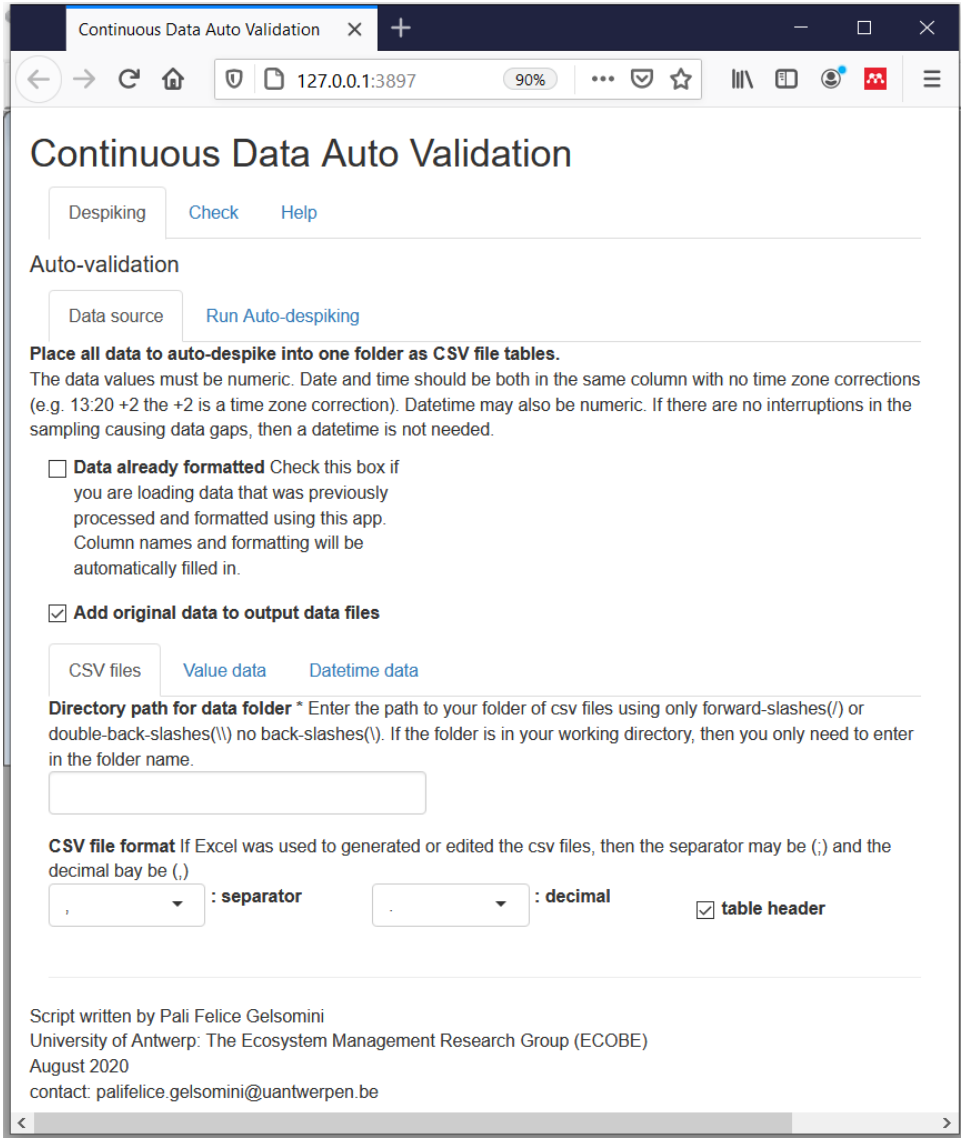
Auto-validation

In this step the data runs through a series of automated processes to clean and despike the data.

10. If this is a new R session (you've closed the program and restarted it) you will have to enter **library(HICbioclean)** into the R console and press enter again first. Otherwise move onto the next step.
11. Enter **HIC.App.auto()** into the R console and press enter to open the auto-validation app.

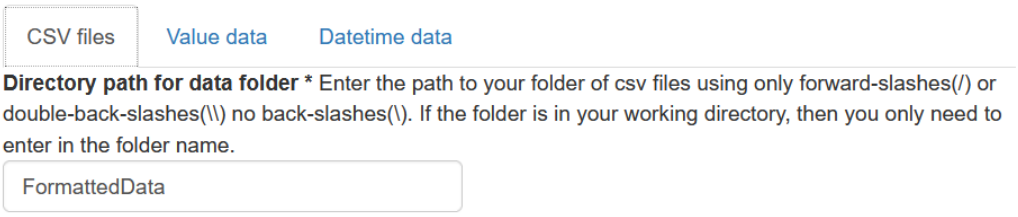
```
exporting data tabl
> HIC.App.auto()
```

12. The app should now be open in your web browser.



13. All the default settings in the app are selected so that you should be able to process the HIC data without making any changes to them. If need be you can always change the settings in the app. See the in app descriptions of each setting.

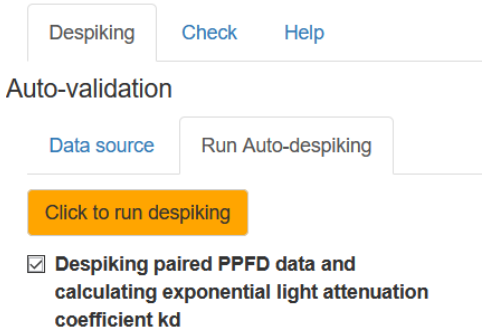
14. Enter folder where you saved your formatted data from the previous step into the field "Directory path for data folder". For me that is "FormattedData" and because I saved that folder in my working directory I don't need a full path.



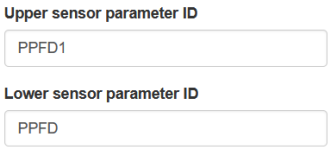
15. The other tabs "value data" and "datetime data" contain the column names and data format of the values and datetime. If you are working with HIC database text files then the settings are already filled in for you correctly by default. There is no need to make any changes to these tabs.
16. Open the "Run Auto-despiking" tab.

PPFD data specific instructions

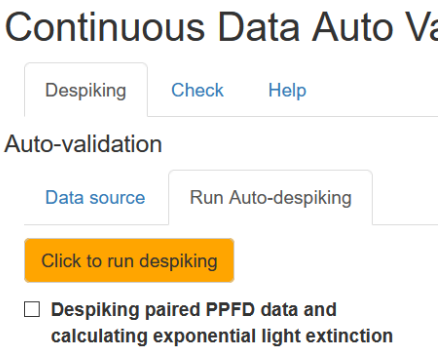
a. If you are auto validating paired PPFD data for calculating light attenuation coefficient kd, the steps are all generally identical. But you have to tell the app that you are auto validating PPFD data. Under the tap "Despiking">>"Run Auto-despiking" check the box "Despiking paired PPFD data…".



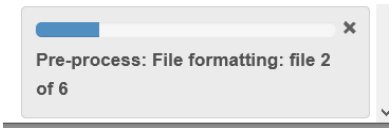
b. Make sure that in your data in your data the upper sensor is labeled PPFD1 and the lower sensor is labeled PPFD. You can change these options farther down on the page if need be.



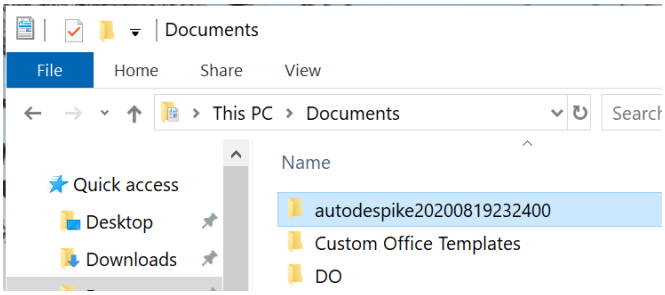
17. and click the orange "Click to run despiking" button. This will run the full work flow with the the default settings.



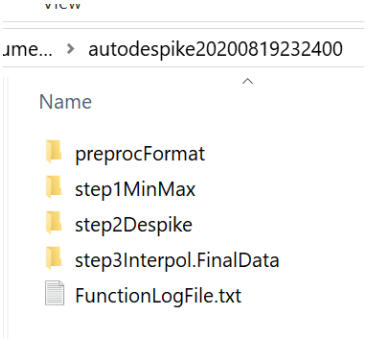
18. A progress bar will appear in the corner of the screen wile the process is running



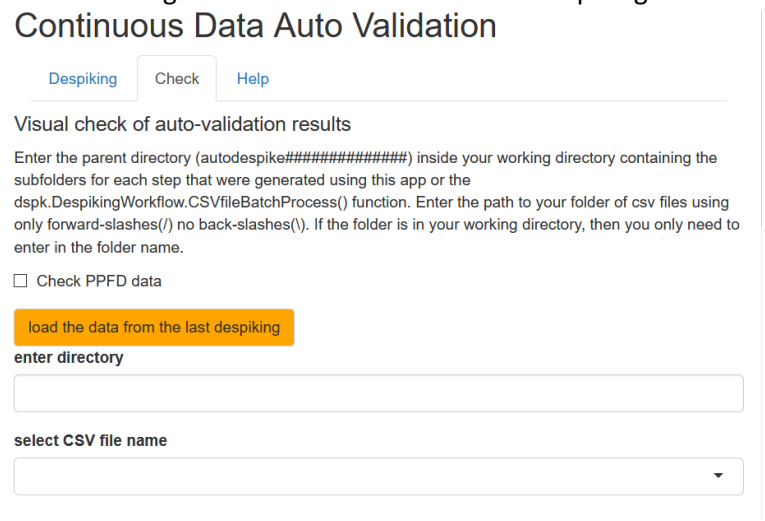
19. The auto validated data is saved in your working directory in the autogenerated folder "autodespikeYYYYMMDDHHMMSS". If it is PPFD data you are processing it will be saved in "autoPPFDdespikeYYYYMMDDHHMMSS".



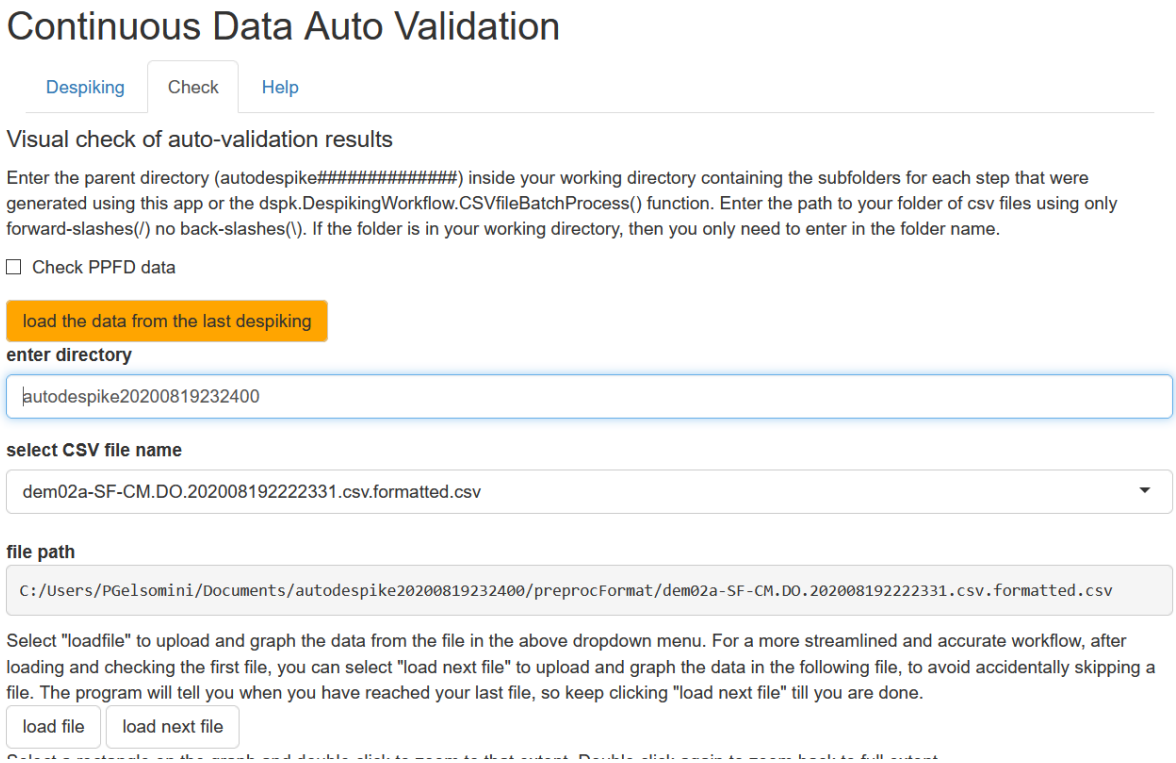
20. Inside that folder you have 4 subfolders with the data generated at each step. The last folder labeled "FinalData" that the fully auto-validated data. There is a text file "FunctionLogFile.txt" that stores the function arguments for later reference and reproducibility, any calculated function arguments such as sampling interval and any error messages for each file.



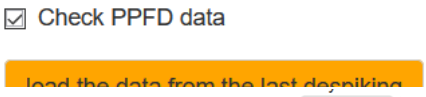
21. After the process is complete you can move from the "Despiking" tab to the "Check" tab and click the orange "load the data from the last despiking" button.

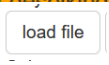


22. This will populate the “enter directory” field with the folder that was just auto generated during the last auto-validation process. In the drop down menu you can select the different files, but don’t do this just yet. You can enter a folder name into the “enter directory” field but this folder must be in your working directory.



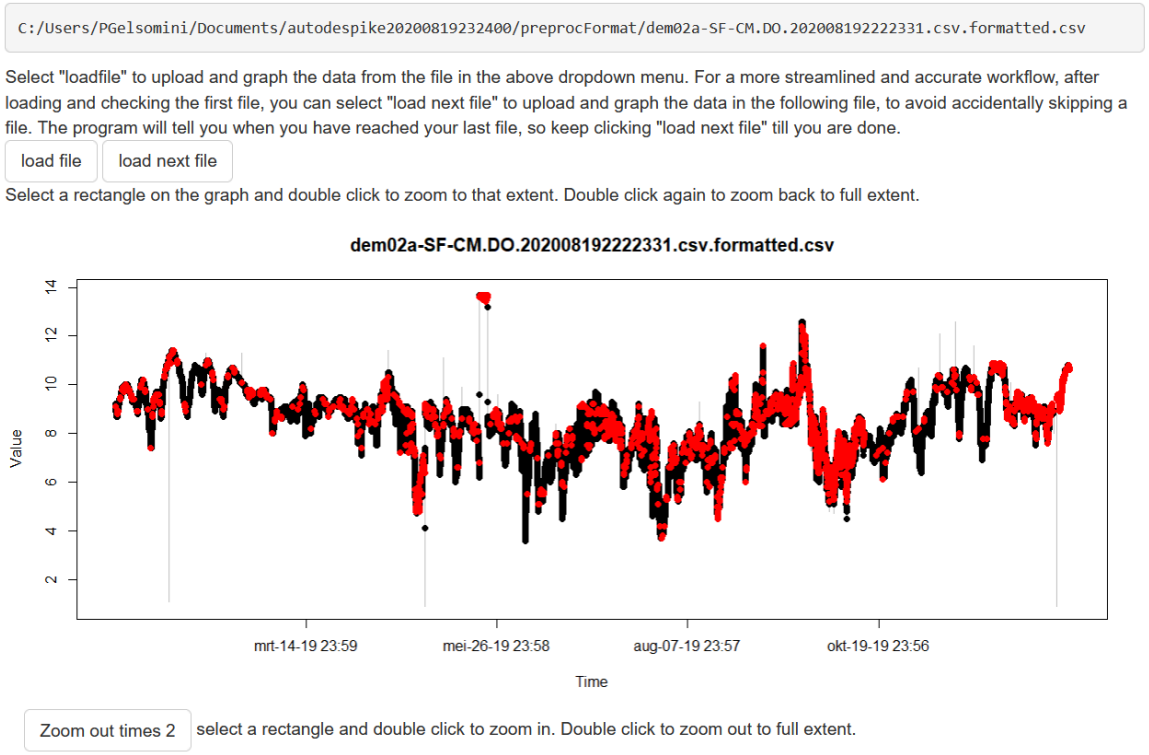
23. If you are checking PPFD data you must tell the app that it will be graphing paired PPFD data and light attenuation coefficient kd data. Check the box next to “Check PPFD data”.



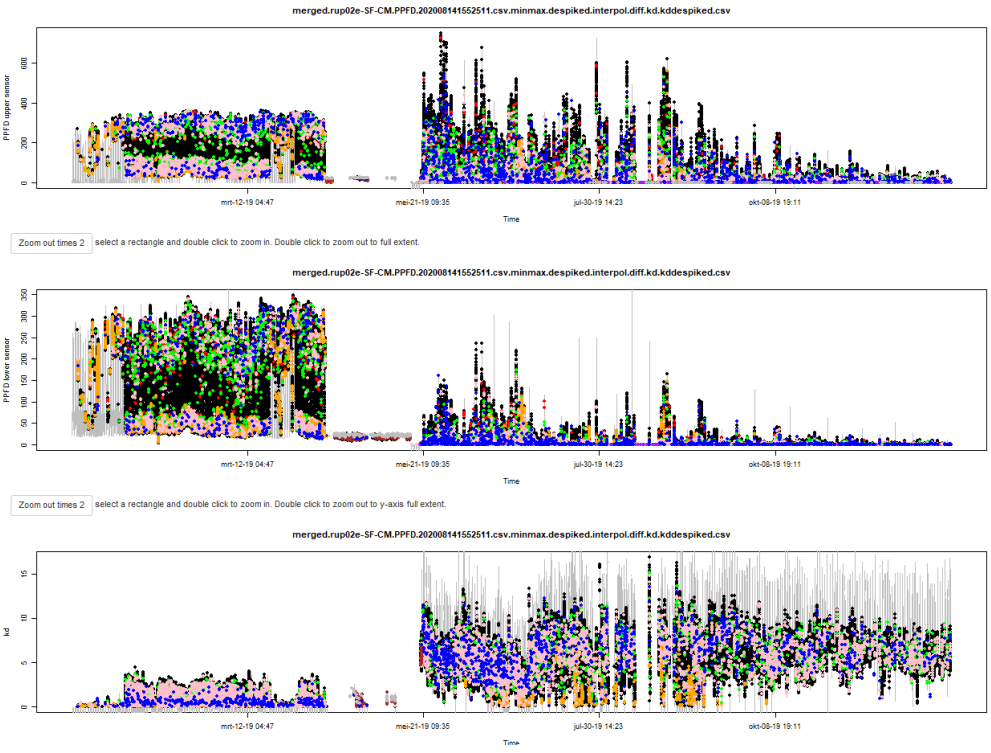
24. Click the load file button 

to load the data from the first file.

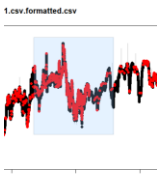
25. The graph of the first dataset will appear just below.



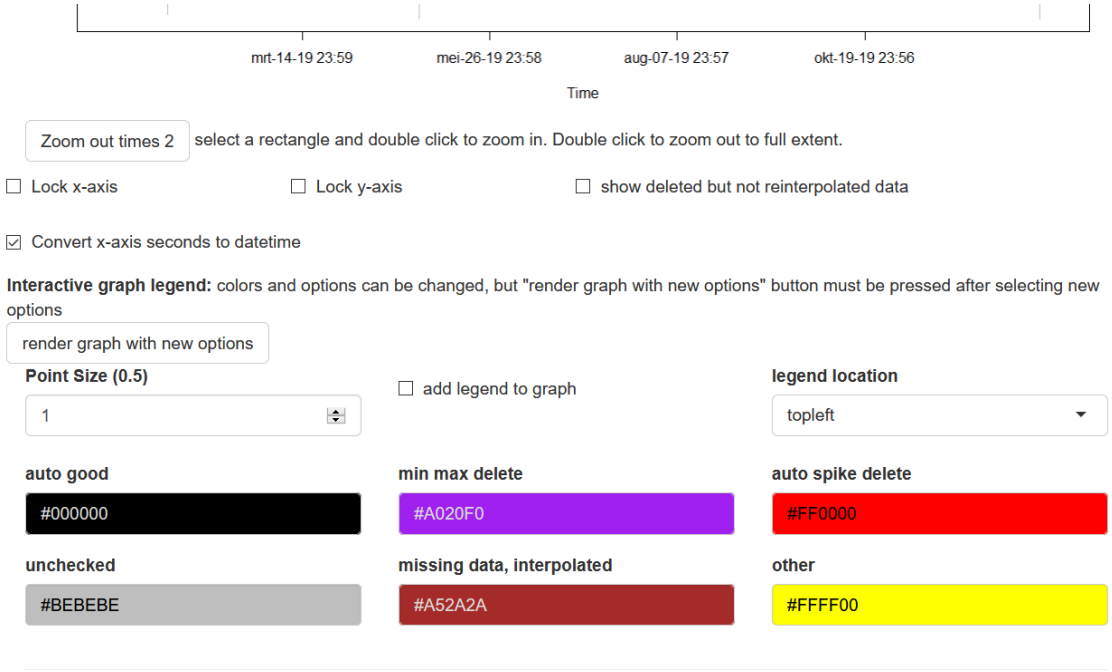
26. If you are graphing PPFD data there will be three graphs. (upper sensor PPFD, lower sensor PPFD, and kd) and some extra state of value codes specific to PPFD validation procedures. The x-axis of all three graphs are linked to each other so the data will always be displayed perfectly aligned.



27. The origin data is graphed in a thin gray line. The Validated and interpolated data is graphed as points. The points are color coded according to their state of value

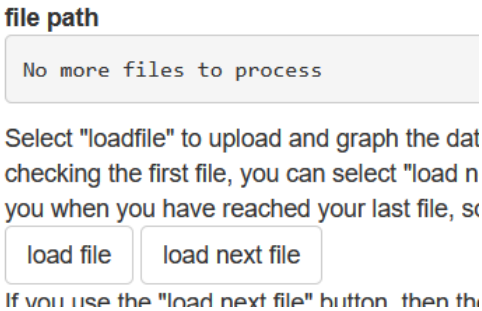


28. Pull a box on the graph and double click the graph to zoom to that area. Double click again on the graph to zoom back out the full extent. Click the zoom out times 2 bottom Zoom out times 2 to increase your extent by 2. Use these methods to move around the graph to explore the data and see if the auto-despiking worked sufficiently.
29. Below the graphs there are more options and tools for exploring the data. You can lock the x and y axises while zooming. Show data that was deleted but didn't get reinterpolated. Convert x-axis from seconds to datetime. Point size. Legend colors. And you can add the legend to the graph. After you make changes to the interactive graph legend, you must click the button "render graph with new options". With large yearly datasets graph rendering is quite slow so it is best if this is done once you are done making changes to the legend.



30. Click the "load next file" button load next file which is found next to the "load file" to move on to the next data file. This way you won't accidentally forget one. You may have to double click the graph to reset it to full extent to see all the data. Using the "load next file" button locks the dropdown menu, so you must first click the "reset file dropdown menu" button if you want to manually select a file, for example if you want to go back to a previous file.

31. Once you've reached your last file, the app will tell you "No more files to process".



32. If you are not happy with the results you can change the settings of the auto-validation under the tab "Despiking">>"Run Auto-despiking". See the in app instructions on their use and for additional help and details on the algorithms see the "Help" tab.
33. Now either return to step 22 and enter a new parent directory in the "enter directory" field, return to step 14 to auto-validate a new set of data or close your app browser window and move onto the next step which is the manual-validation and final export.

To find your working directory and for more details into the algorithms see the 'Help' tab.

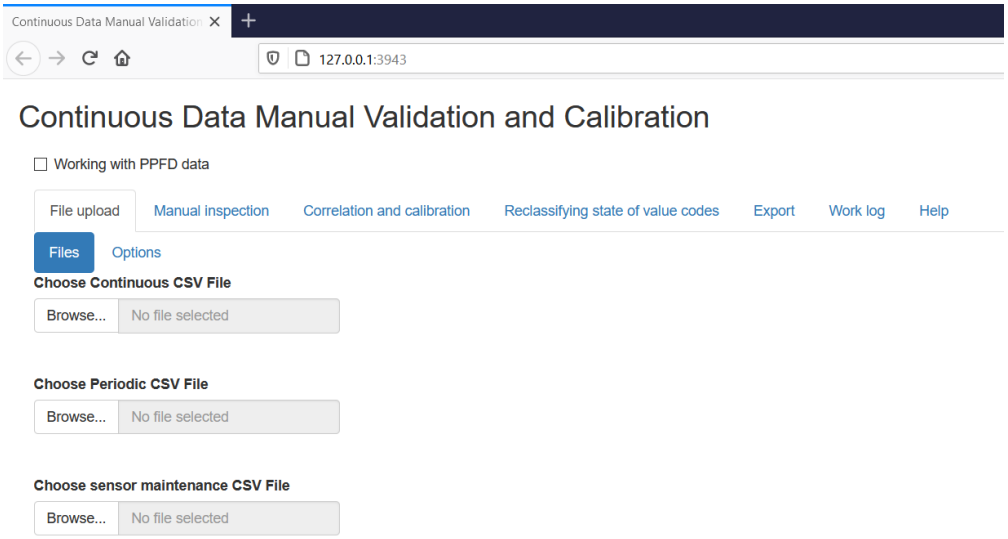
Manual-validation and final export

The last step after the formatting and the auto-validation. The other steps can be done in R code without the use of the Shiny apps, but this step must be done in the Shiny app because this is a manual check so it must be done visually.

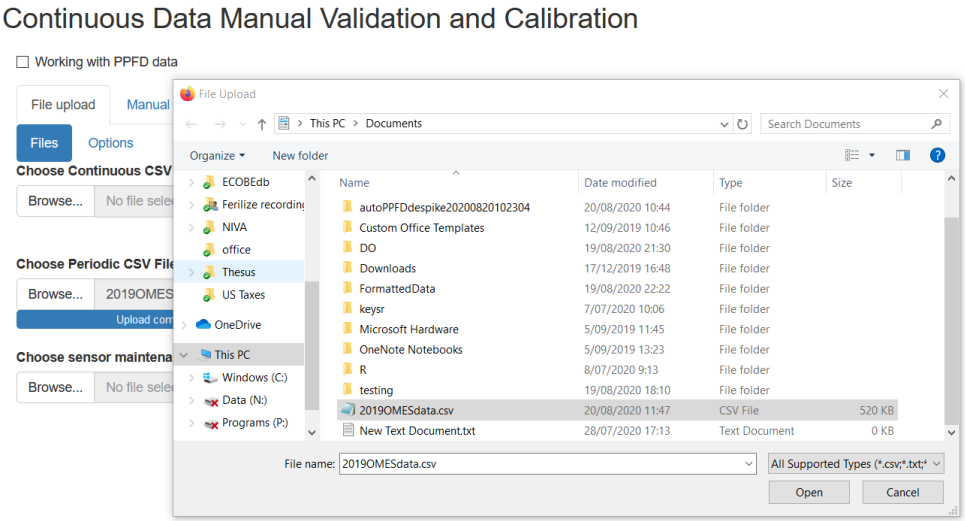
34. If this is a new R session (you've closed the program and restarted it) you will have to enter **library(HICbioclean)** into the R console and press enter again first. Otherwise move onto the next step.
35. Type **HIC.App.manual()** into the R console and press enter.

```
   [No stack trace av
> HIC.App.manual()
```

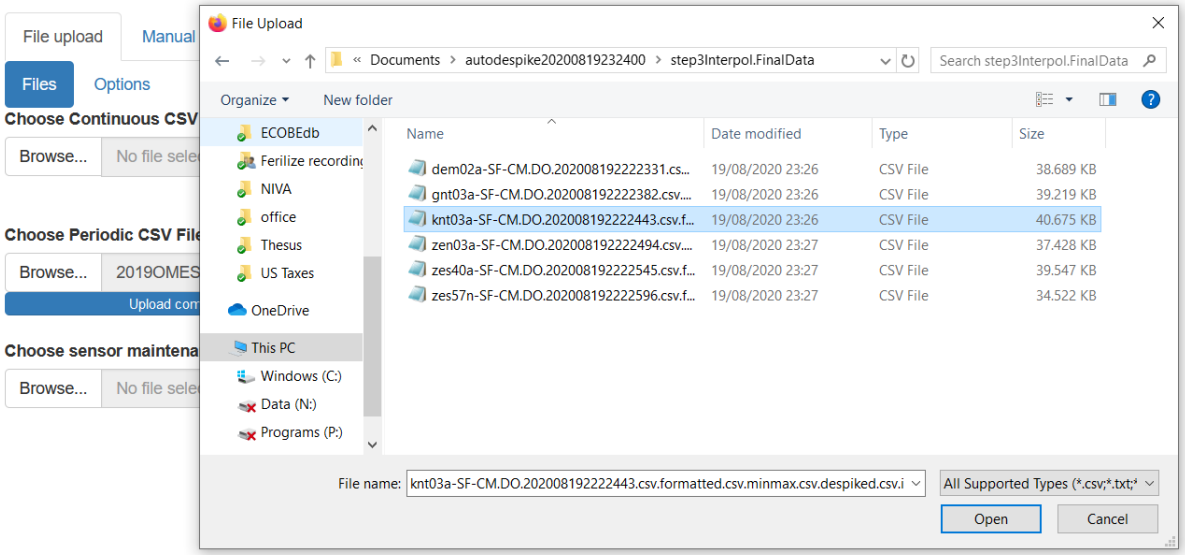
36. The app should open in your web browser.



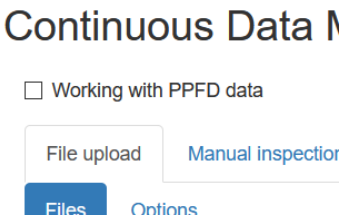
37. Click browse to choose the OMES data file as your periodic csv file for your reference data. The OMES data must be in a csv with semicolon ";" separation, the column names "ReadingDate", "StationName", "ParameterName", and "ReadingValue" and datetime format 15/01/2019 13:13. You can format this file in excel and save as csv to achieve this.



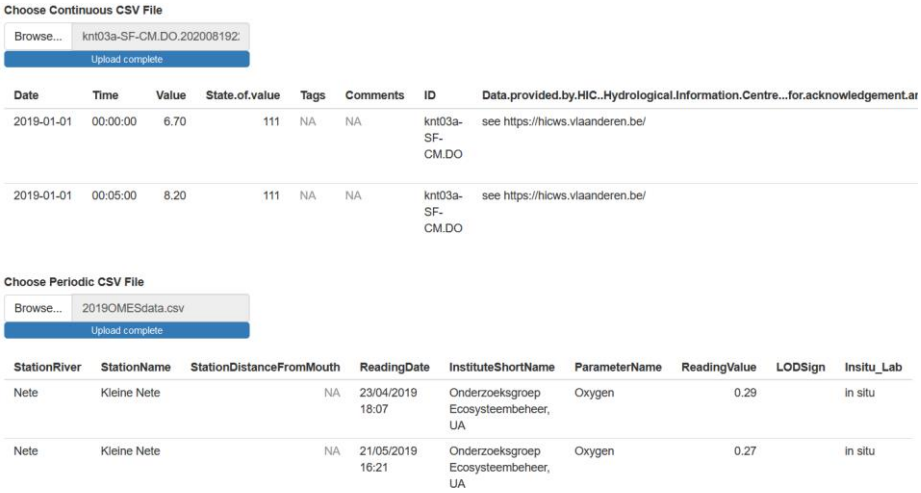
38. Click browse to choose your continuous data file that you have auto-validated. It will be in the folder “autodespikeYYYYMMDDHHMMSS” or “autoPPFDdespikeYYYYMMDDHHMMSS” in the subsolder “FinalData” in your working directory (see the help tab for your working directory).



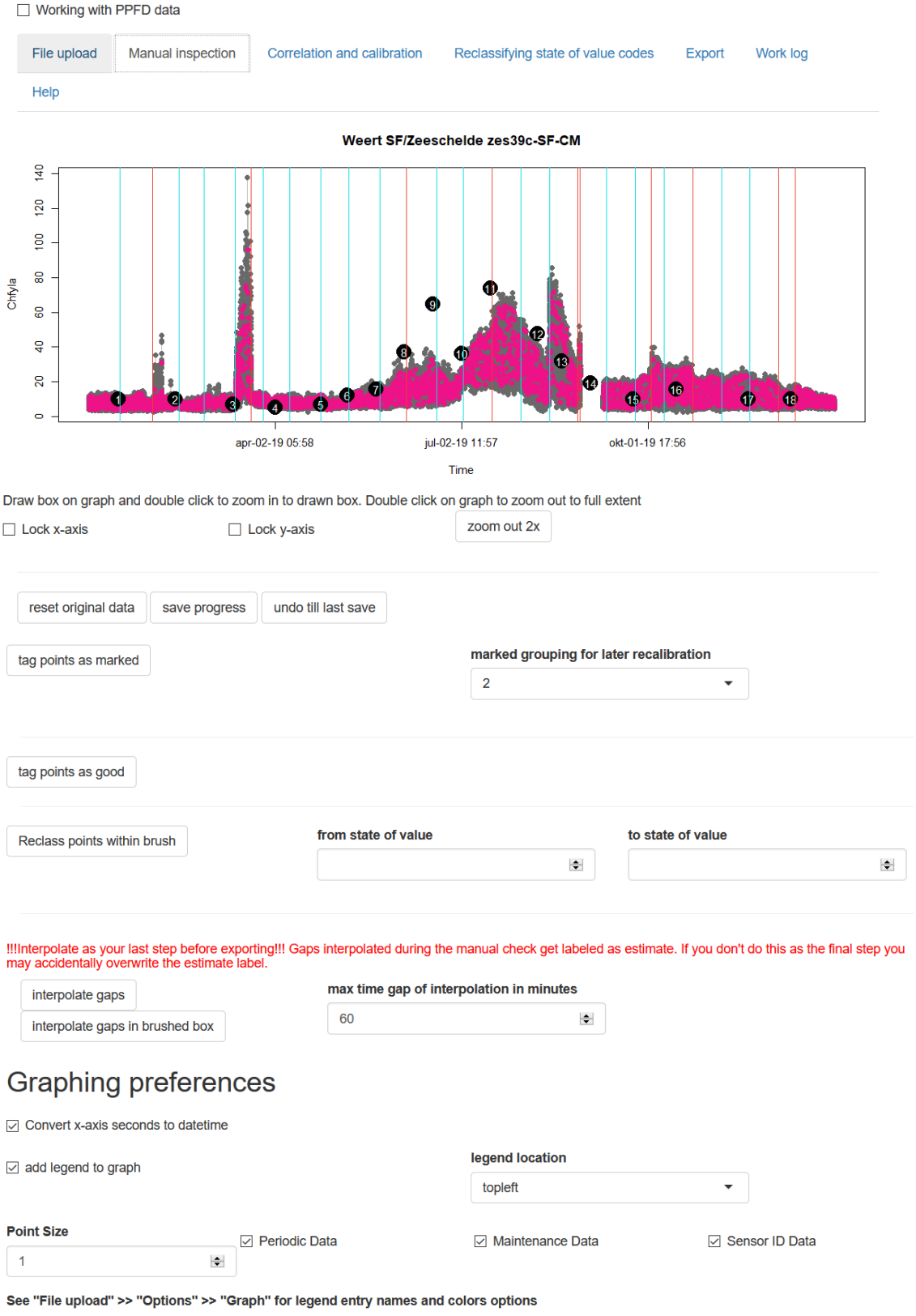
39. Just as before, here you must also say if you are working with PPFD data because it is graphed in three graphs instead of one and all three parameters (upper PPFD, lower PPFD and kd) are dealt with simultaneously.



40. If you have a sensor maintenance file for your site, load that as well. The excel sensor maintenance files from WISKI must first be converted to csv files with the function HIC.maint() or the R Shiny app HIC.App.format(). See the earlier section on formatting.
41. Once the data is uploaded into the app, you will be able to see the first few lines of the files. The periodic OMES file gets subsetted based on the parameter and the location, so if you don’t see the periodic OMES file, then there is either no reference site for the continuous data that you have open, or there is no data on that parameter.

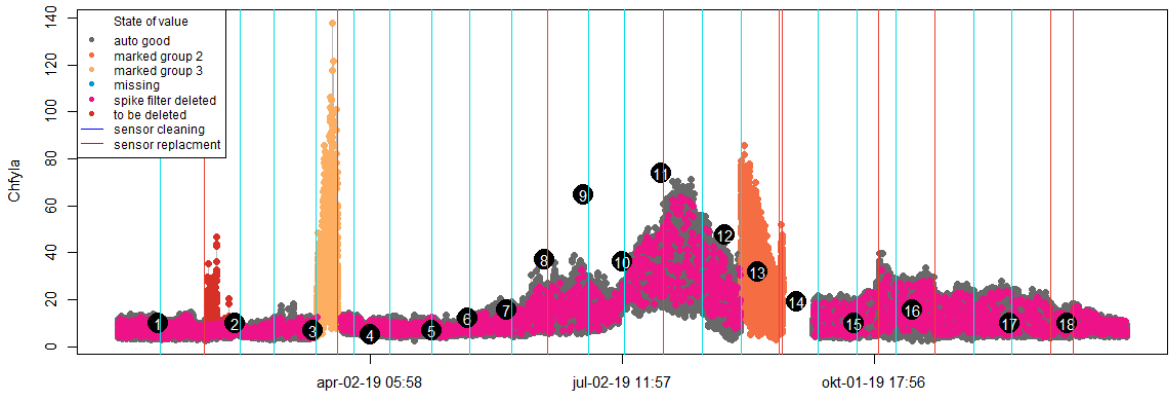


2. Once the data is uploaded you can move to the “Manual inspection” tab. Here the data will be graphed. The continuous data is graphed colored by state of value. The reference site data is graphed over it. The sensor maintenance times are graphed as horizontal lines. You have tools for formatting the graph, exploring the graph and marking sections of data.



a. The graph can be zoomed in on by drawing a box on the graph and double clicking. Double click on the graph again to zoom out to full extent. Use the “Zoom out 2x” button to increase your extent by 2 times. You can lock the x and y axis while zooming.
b. Your work can be saved, reverted back to the previous save or reset to the original data.
c. You can pull a box around a section of data on the graph and mark that section as a “marked group”. These marked groups can be mathematically transformed later, deleted, or have their state of value set to a non-work-class state of value such as “suspect”. Upon export, only non-work-class state of values will be kept, all others will be set to state of value “good”. Marked group 1 will be labeled in your graph as “to be deleted” but of course you don’t have to delete it. This was just set like this so that if you save an image of the graph, you know that those points are deleted.
d. You can pull a box around points and tag them as “good”.
e. You can reclassify state of values within just your box that you pulled around some data form one value to another. You must enter numeric values here, no words. To know the all numeric values of the state of values you can go to the tab “File upload”>>”Options”>>”Graph”. To know all the codes that are currently in your data go to the tab “Reclassifying state of value codes” and see the table at the bottom of the page. You are manipulating the actual number codes so be careful.
f. You can interpolate gaps. Either interpolate all you data or just the data within your brushed box on the map. The maximum data gap size to interpolate can be selected. It is best to do this as the absolute very last step because interpolated data upon export gets the state of value label “estimate” and there is a risk that you will overwrite the code with another state of value if you keep working on that data.
g. You can change some of the graphing preferences to make it easier for you to see the data. The colors and the legend labels can be changed in the tab “File upload”>>”Options”>>”Graph”.

43. The recommended workflow is to first mark in different groups all the sections of data that you want to delete, recalibrate and/or mark as suspect. You don’t have to mark anything as “good” since the data will be automatically marked as “good” upon export.



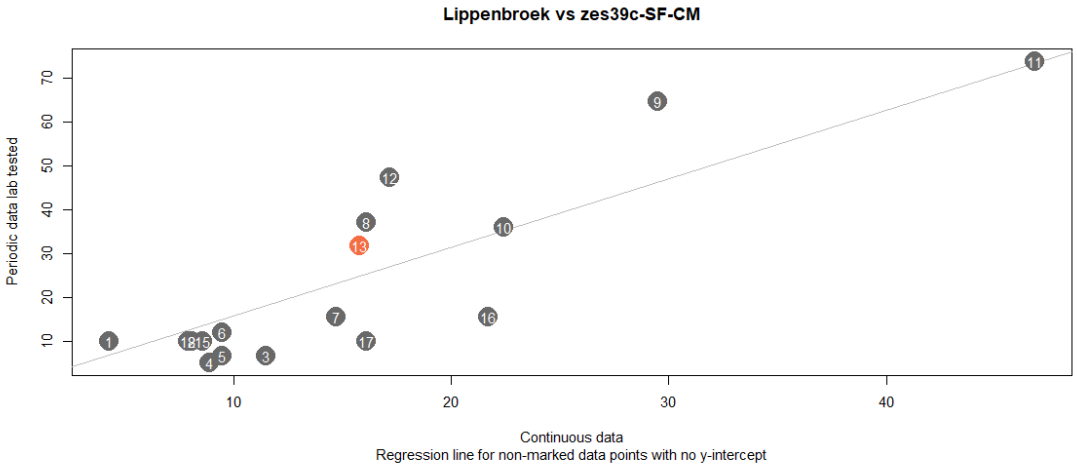
In the above example a marked a cluster of chlorophyll spikes in February to be deleted (marked group 1), I marked a period of very high chlorophyll in March as group 3 which will be labeled as “suspect” and I marked a section right after a sensor cleaning in August as group 2 which will be recalibrated and marked as “estimate” to fit with the trend of the surrounding data.

44. Now go to the “Correlation and calibration” tab. Here you can perform mathematical transformations on your data.

☐ Working with PPFD data

File upload | Manual inspection | Correlation and calibration | Reclassifying state of value codes | Export | Work log | Help

a



☐ add legend to graph

legend location

topleft

b

The marked group do you wish to calibrate (0 means not marked)

0

Calibration formulas based on the selected group. If no marked group is selected (you selected 0 above) then the calibration formulas will be based on all non-marked data that is not labeled as "suspect" or "suspect calc". However when you click the button "calibrate points" the points labeled "suspect" and "suspect calc" will still be calibrated.

a

```
y = -5.4743 + 1.8126*x    r.squared = 0.763
```

```
y = 1.5688*x    r.squared = 0.883
```

d

☑ use formula with no y-intercept

Auto calibrate points

Enter calibration formula here manually as a function of x with base R operators. If this is blank then the above automatic calibration formulas will be used.

e

example: (5 + 6*log(x)^3)/2

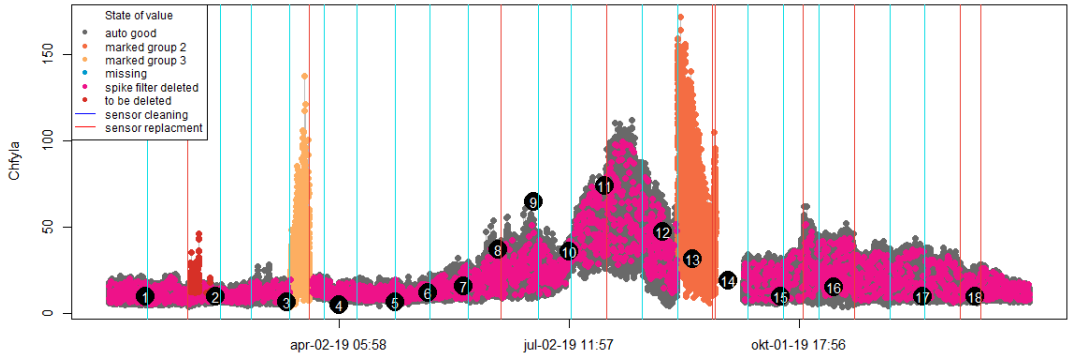
Manual calibrate points

f

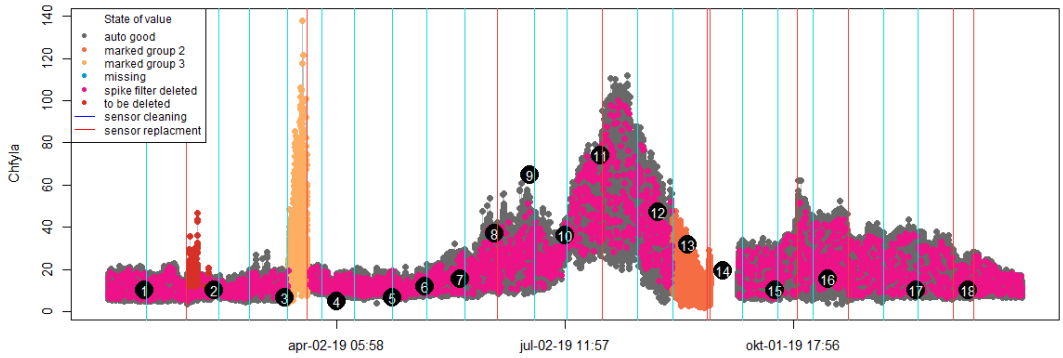
tPeri	valPeri	valCont	state	tCont	corID
1547549640.00	10.00	4.30	80.00	1547549700.00	1
1549970520.00	10.00	8.10	80.00	1549970700.00	2
1552388400.00	6.67	11.50	80.00	1552388700.00	3
1554199380.00	5.00	8.90	80.00	1554199500.00	4
1556101860.00	6.67	9.50	80.00	1556102100.00	5

a. If you have reference data, then a correlation graph and linear regression equations will be displayed. The linear regression equations and the trend line on the graph are based on the marked grouping that you have selected in the “The marked group do you wish to calibrate” dropdown menu.
b. The marked grouping that you have selected in the “The marked group do you wish to calibrate” dropdown menu is also the grouping that the mathematical transformations will be done on. So even if you have no reference data, it is still important to select the correct grouping. So for example if you have group 0 selected (which stands for not marked) then all the marked data points will not be transformed. You will have to do each group separately if you want to do the same transformation on all your data.
c. You may use the “Auto calibrate points” button or the “Manual calibrate points” button to transform your data.
d. The “Auto calibrate points” button will use the linear regression equation. With the check box “use formula with no y-intercept” you can chose which linear regression you want to use.
e. With the “Manual calibrate points” button you can type a formula into the filed above it and that formula will be used. Enter the formula with base R arguments and as a function of x. e.g. (5+6*log(x)^3)/2
f. If you have reference data, the reference-data-point next to the nearest in time continuous-data-point is shone in a table at the bottom of the page.

45. In my example I am working with florescence chlorophyll data and chlorophyll data must all be post calibrated with a linear regression with no y-intercept. So I will click the button “Auto calibrate points” with group 0 selected. Group 3 will be marked as suspect but I still want it calibrated with the rest of the data. I will select group 3, manually enter the formula 1.5688*x and press the “Manual calibrate points” button.
46. Point 13 was from group 2 which was the section of data that seemed to have a wrong calibration compared to the rest of the dataset. I will try to auto calibrate it but first I will save my work by going to the “Manual inspection” tab and clicking the “save progress” button. Now I will go back to “Correlation and calibration” tab, select group 2 as the marked group I wish to calibrate and I will click “Auto calibrate points” and then look at the data again in the ““Manual inspection”.

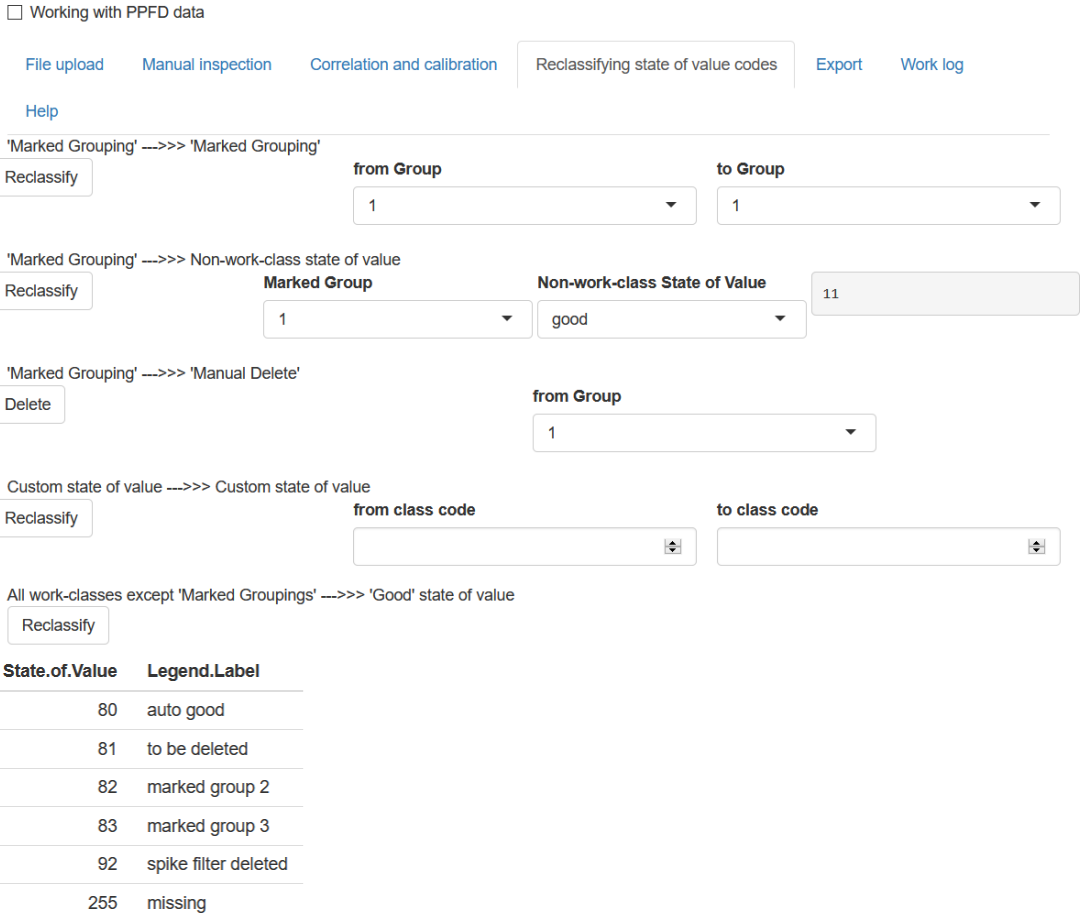


As you can see it didn’t work at all. Marked group 2 is now even greater compared to the rest of the data. So I click the “undo till last save” button. From visual inspection I estimate that the correction formula needs to be x/1.8. In the “Correlation and calibration” tab I will select group 2, type that formula in and press the “Manual calibrate points” button.



Now the data in group 2 looks much better.

47. Now go to the “Reclassifying state of value codes” tab. Here you set the final state of values of the different marked groups and you can delete points in marked groups.

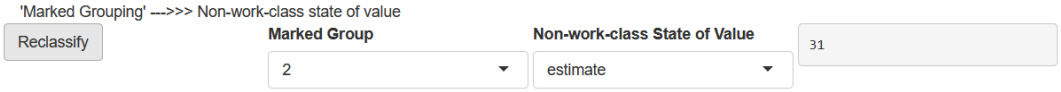


State.of.Value	Legend.Label
80	auto good
81	to be deleted
82	marked group 2
83	marked group 3
92	spike filter deleted
255	missing

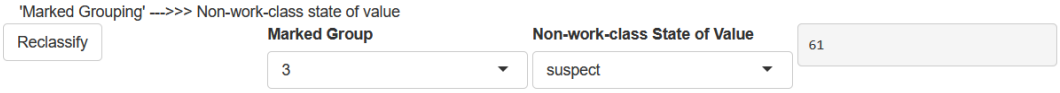
a. You can reclassify one marked group to another group.
b. You can reclassify a marked group to a non-work-class state of value. These are the final values that will be exported. Anything marked as one of these values will not be labeled as “good” upon export.
c. You can delete all points within a group.
d. You can reclassify any state of value numeric code to another state of value numeric code. Be careful since you are amending the actual codes.
e. You can change all working-class codes (which are the codes that are given during the data formatting and auto-validation and the marked groupings) to the “good” state of value. This is not required since these will be changed to “good” anyways upon export.
f. At the bottom of the page you can see all the current state of values and their numeric codes that you have in your dataset.

48. In my example I will delete group 1, so I will press the “delete” button with group one selected.

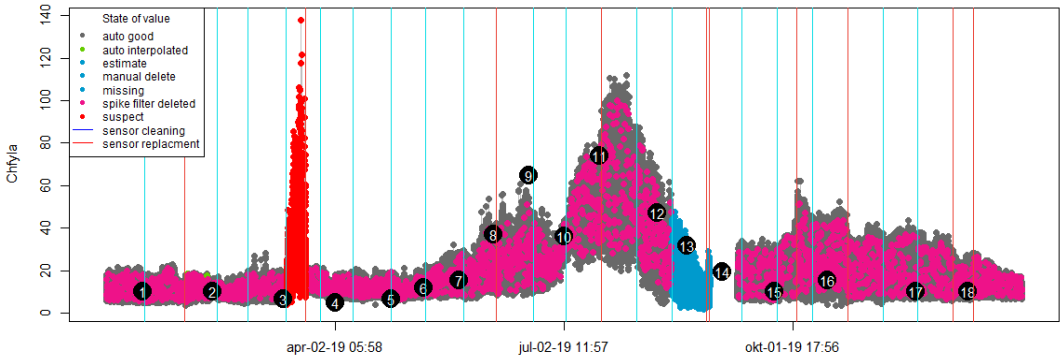
I will reclassify marked group 2 to “estimate” since I recalibrated that data visually to match the surrounding data.



I will reclassify marked group 3 to “suspect” since I don’t trust those extremely high chlorophyll values in March.

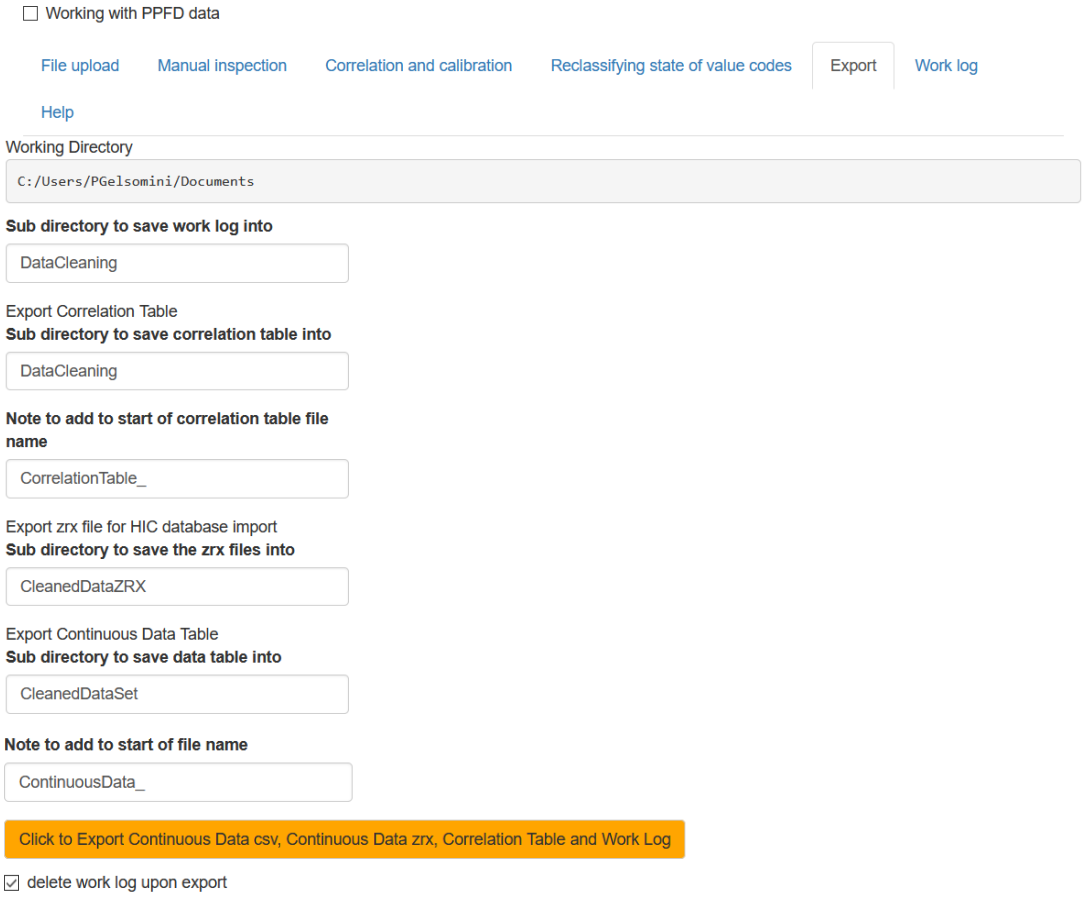


I will now go back to the “Manual inspection” tab to look at my data one last time and interpolate the gaps that I created with deleting group 1 with the “Interpolate gaps” button.



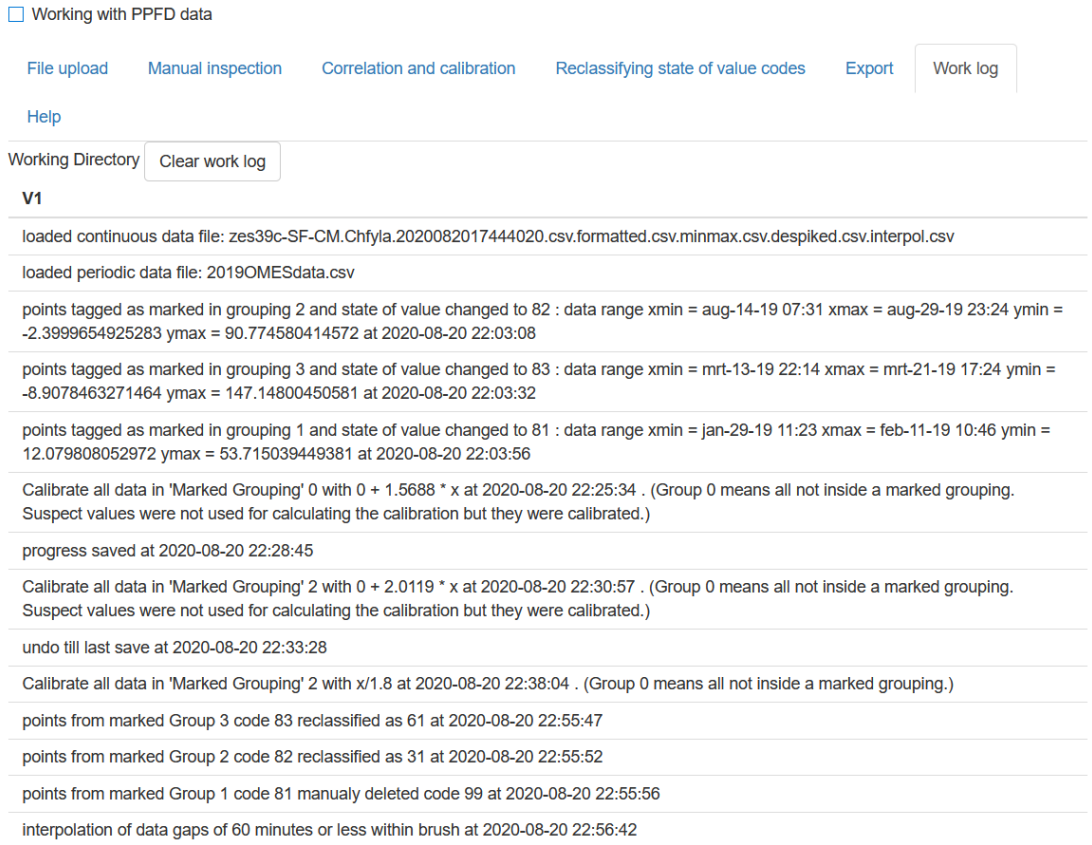
The final data looks good.

49. Now go to the “Export” tab. Here you select the names of the output directories and the prefixes you want put on the front of each file. When you click the orange button at the bottom of the page, the current version of the continuous data will be exported to a csv file, a zrx file will be exported for import back into the HIC database with the state of value codes all formatted to the requirements of the HIC, the correlation table from the “correlation and calibration” tab will be exported and the work log where you will find a detailed account of every change you made to the file for later reference and reproducibility will be exported as well. The final data will all be saved into your working directory in the subdirectory folders that you gave on this page. You can see you working directory at the top of this page.



*NOTE:The station parameter codes that are needed for import back into the HIC database which are used in the zrx files are stored inside this package. Type **zrxFileStationCodes** into the R console and press enter to see the list of codes. Please contact us if you need to make changes to this file.*

50. In the tab “Work log” you can see an account of everything you did to your data. This can be very useful especially when you need to know what calibration you used on your data. This work log will be deleted from the app once you export the data, but it will be saved as a text file.



51. Now go back to step 38 and select the next auto-validated continuous file and WISKI maintenance file. The periodic OMES dataset only needs to be uploaded once at the start of your session, because the file will automatically update if there is a reference site and available data.
52. When you are done, you must close the app window before continuing work in R or opening another R Shiny app.

R package help files

Batch process: Format data exported from the HIC database

Description

Bach process function for importing the continuous water quality data from the HIC Hydrological Information Center(HIC) database. This function takes the csv file that is exported from the HIC data base and converts it into a format that can be used easily in R. The header of the HIC csv file has horizontally oriented metadata. These meta data are taken from the header and put into columns in the dataset. The dates and times are converted into R friendly datetime format and into UNIX numeric format for easier handling in R.

Usage

```
HIC.Continuous.Data.Import.Format(
  InputDirectory,
  OutputDirectory,
  Data.sep = "\t",
  Meta.sep = ":",
  Data.header.line = NULL,
  Dec = ".",
  DateFormat = "%d/%m/%Y",
  TimeZone = "Etc/GMT-1",
  OneYearDataSet = F,
  ValueColumnNum = 3,
  ParamNameColmn = "Parameter.Name",
  StationNoColmn = "Station.Number",
  DateColmn = "Date",
  TimeColmn = "Time"
)
```

Arguments

`InputDirectory`
: Path to the folder directory containing all the csv tables that you wish to format placed in quotations. Must have no back slashes (\), they must be all forward slashes (/) or double back slashes (\\).

`OutputDirectory`
: Path to the directory where you wish to save the formatted data in quotations. Must have no back slashes (\), they must be all forward slashes (/) or double back slashes (\\).

`Data.sep`
: The field separator character for the value data. The columns are separated by this character. The default is tab separated "\t".

`Meta.sep`
: The field separator for the metadata values. The columns are separated by this character. The default is colon separated ":".

`Data.header.line`
: It is assumed that the data header is the first line with the most separations. But if not, then the line number of the data header can be specified.

`Dec`
: The decimal character. By default ".".

`DateFormat`
: Character string giving the date format. See the strptime() help file for additional help.

`TimeZone`
: The time zone is by default UTC+1 "Etc/GMT-1". Use OlsonNames() for a list of all time zone names.

`OneYearDataSet`
: If the dataset is only within one calender year, then you can change this to TRUE and the year will be added to the ID and the file name, but if there are more than one calender years in the dataset then a warning message will appear and the year will not be added to the ID or file name.

`ValueColumnNum`
: The column number of the data values. This column has inconsistent naming and thus must be refered to by column number.

`ParamNameColmn`
: The parameter name column name in the meta data. If you enter in new names, then replace all spaces and special characters with "."

`StationNoColmn`
: The station number column name in the meta data. If you enter in new names, then replace all spaces and special characters with "."

`DateColmn`
: Date column name in quotations.

`TimeColmn`
: Time column name in quotations.

Details

Place all HIC csv files into one directory. Specify this InputDirectory in the function in quotes and with forward-slashes(/) or double-back-slashes(\\) no back-slashes(\). Specify the OutputDirectory where you would like to have the data be exported to in quotes and with forward-slashes(/) or double-back-slashes(\\) no back-slashes(\). If you copy the directory path from windows, it will have back-slashes(\) and these need to be changed to forward-slashes(/) or double-back-slashes(\\). If you don't write the full path for the OutputDirectory, then it will create that directory in your working directory.

Assumed input data file structure

Data table format assumed to be a vertical list of the meta data on top of the horizontally oriented data table.
Example of the assumed data table structure of the input data:

Station.Number: RTZ25
Parameter.Name: temp
Parameter.Unit: C

Date	Value	State.of.Value
25/01/2018	5.6	110
25/01/2018	7.8	110
25/01/2018	4.2	110

Possible issues

This code can't deal with extremely inconsistent column names between files. It searches for key words to find the columns in the meta data, but if there are no common words between the different files, then it can't find them. Data is all saved with auto-names Station.ParameterName.Year.SystemTimeInSecondsFileNumber.csv so there is a risk of overwriting older data if you run this batch process in a loop and if multiple files are processed within less than a second of each other with the same station and parameter and happen to be the same file number in their folder. This is unlikely to occur but in theory is possible.

Value

This function returns each seperate csv file in the input directory as a separate comma separated csv file in the output directory with all the metadata placed into columns to the right of the data, date and time merged into one datetime column ("DateTime"), a numeric datetime column ("DateTimeUnix") in UNIX seconds and all parameter values in the column "Value".

Examples

```
HIC.Continuous.Data.Import.Format(
    InputDirectory = "C:/Rdata/originaldata/HICdata",
    OutputDirectory = "FormattedHICdata")
#the folder "FormattedHICdata" will be created in your working directory since it is not a full path.
```

Batch process folder for converting maintenance Excel files to csv files

Description

This function was specifically made to convert the excel sensor maintenance files into csv files that can be easily read into R.

Usage

```
HIC.maint(input.directory, output.directory)
```

Arguments

`input.directory`	folder directory of Excel files. No back slashes(\), only use forward slashes(/) or double back slashes(\\).
`output.directory`	folder directory where you would like to save the csv files. No back slashes(\), only use forward slashes(/) or double back slashes(\\).

Details

It batch processes all the files inside a given folder.

If you don't provide a full path to the output directory then it will be placed in your working directory.

Value

A folder of csv files with the same name as the input excel files.

Examples

```
HIC.maint("C:/Rdata/MaintenanceFiles", "MaintenanceFilesCSV")
```

Batch process: Despike and autovalidate continuous data

Description

A full work flow for auto validation of continuous data. It may be run as a batch process or on individual R objects. It performs min max filtering, despiking and linear gap interpolation. You may run all these steps or a select few. Files are all outputted as csv files to your working directory at each step.

Usage

```
dspk.DespikingWorkflow.CSVfileBatchProcess(
  steps = c(1, 2, 3),
  input.directory = NULL,
  sep = ",",
  dec = ".",
  header = T,
  Data = NULL,
  Value,
  val.NAvalue = NULL,
  unchecked.state.of.value.code = 110,
  NA.state.of.value.code = 255,
  add.original.data = T,
  DateTime = NULL,
  datetime.format = NULL,
  datetime.timezone = "GMT",
  ConditionalMinMaxColumn = NULL,
  ConditionalMinMaxValues = NULL,
  ConditionalMin = NULL,
  ConditionalMax = NULL,
  Min = (-Inf),
  Max = Inf,
  minmax.state.of.value.code = 91,
  sampling.interval = NULL,
  despiked.state.of.value.code = 92,
  good.state.of.value.code = 80,
  despike.threshold = 3,
  despike.Method = "median",
  precision = NULL,
  max.gap = Inf
)
```

Arguments

`steps`

Numeric vector containing the values 1, 2 and/or 3 corresponding to step 1 min max filter, step 2 despiking and step 3 gap interpolation. For example ‘steps=c(1,3)’ will run a min max filter and then interpolate the gaps. Will run all steps by default.

`input.directory`

Character string of the path to the folder containing all the csv files that you wish to batch process. This argument may be omitted if you are entering vectors directly into the ‘Value’ and ‘DateTime’ arguments.

`sep`

Arguments indicating the formatting of the input csv files. It is the field separator character. Values are separated by this character. By default it is comma ",".

`dec`

Arguments indicating the formatting of the input csv files. It the character used for decimal points. By default if is a period ".".

`header`

Arguments indicating the formatting of the input csv files. It is a logical value indicating if the first line is the column titles. By default it is TRUE.

`Data`

A dataframe object. If you only wish to process one data frame, then it can be entered directly from the R environment with this argument. If you enter in an input.directory then 'Data = ' will be ignored and the files from the input directory will be processed.

`Value`

If the data is from a csv file or a dataframe, it is a quoted character string indicating the column name or an integer indicating the column number of the column containing the data values that you wish to despike. Data may also be entered as a single vector object (unquoted) such as 'Value = mydata$values' or 'Value = values'

`val.NAvalue`

The value indicating an NA value in your input data. If this value is NA, then this argument can be omitted.

`unchecked.state.of.value.code`

Number indicating that a given value is unchecked. By default 110.

`NA.state.of.value.code`

State of value code given to missing data.

`add.original.data`

A logical value indicating if the original input data should be included in the output tables. Note that if you input a csv file, then every column in that file will be kept. TRUE by default.

`DateTime`

If the data is from a csv file, it is a quoted character string indicating the column name or an integer indicating the column number of the column containing the datetime values of the samples. Data may also be entered as a single vector object (unquoted) such as 'DateTime = mydata$time' or 'DateTime = time'

`datetime.format`

Character string giving the datetime format. See the strptime() help file for additional help.

`datetime.timezone`

Character string giving the time zone of the datetime. By default "GMT". Use OlsonNames() for a list of all time zones.

`ConditionalMinMaxColumn`

The column name in quotes or column number or vector object that contains the factor variable to base your conditional min max filter on

`ConditionalMinMaxValues`

A vector containing the factor values to base the conditional min max filter on

`ConditionalMin`

A vector containing the condition minimums that correspond to the respective values in ConditionalMinMaxValues

`ConditionalMax`

A vector containing the condition maximums that correspond to the respective values in ConditionalMinMaxValues

`Min`

Number giving the minimum reasonable value. All values below this will be deleted.

`Max`

Number giving the maximum reasonable value. All values above this will be deleted.

`minmax.state.of.value.code`

Number indicating that the value has been deleted during the min max filter. By default 91.

`sampling.interval`

As numeric, the time between samples. If you enter NULL then it will calculate it for you. By default NULL.

`despiked.state.of.value.code`

Number indicating that a given value was deleted during the despiking. By default 92.

`good.state.of.value.code`

Number indicating that a given value has been check and deemed not a spike during the despiking. By default 80.

`despike.threshold`
: Number indicating the threshold for defining a spike. By default it is 3, which corresponds to 3 median absolute deviations or 3 standard deviations.

`despike.Method`
: Character string "median" or “mean” indicating the method to use for the despiking. By default “median”.

`precision`
: A number indicating the precision of the input values. Interpolated values will be rounded to this precision. If left as NULL then the numbers will be rounded to the largest decimal length found in the data.

`max.gap`
: As numeric, the time span of the maximum data gap you wish to interpolate

Details

Each csv file will be process separately and saved into new csv files at each step in the despiking process (pre-process: formatting, step 1: min/max filter, step 2: despiking, step 3: gap interpolation). The final data will be found in the folder “autodespikeYYYYMMDDHHMMSS” within the subfolder “step3Interpol.FinalData”. Use the function getwd() to find your working directory. State of value codes are added to the data to keep track of how each value was handled during the auto-validation process (110 unchecked, 255 missing, 80 auto good value, 91 deleted during min/max filter, 92 deleted during despiking). The original data will still be in the newly generated csv files, with the processed data saved in new columns.

Please see the FunctionLogFile.txt that was generated to see any error messages and details about the selected preferences and calculated preferences.

Quick Start

Place all your data you wish to auto-despike into one folder as csv files. The data values must be numeric. Date and time should be both in the same column with no time zone corrections (e.g. 13:20 +2 The +2 is a time zone correction). Datetime may also be numeric. If there are no interruptions in the sampling causing data gaps, then a datetime is not needed.

The default state of values codes

110 Unchecked
255 Missing value
80 Auto good
91 Auto delete min max filter
92 Auto delete Spike

Algorithm overview and workflow

Pre-processing: Formatting and compatibility check: If an ‘input.directory’ containing multiple csv files was provided, then each file will be processed and saved separately. The ‘Value’ data is checked that it is numeric and are then saved into a new column "dspk.Values" to not overwrite old data. NA codes ‘val.NAvalue’ will be replaced with the value NA. The ‘DateTime’ data will be checked if it is numeric and saved into a new column "dspk.DateTimeNum" to not overwrite old data. If it is a character string, then it will be converted to numeric using the provided ‘datetime.format’ and ‘datetime.timezone’. If no datetime is provided then the samples will be numbered consecutively and saved as the datetime. A new column "dspk.StateOfValue" will be generated with all values equal to the ‘unchecked.state.of.value.code’ (default 110). NA values will be given the 'NA.state.of.value.code' (default 255). If the entered csv data table already has a "dspk.StateOfValue" column, then the original state of values from that column will be used. If ‘add.original.data’ is equal to TRUE (this is the default) then the original data will be included in the formatted data table. The formatted data table will be saved to a new csv file in the directory “autodespikeYYYYMMDDHHMMSS/preprocFormat” within your working directory.

Step 1: Min/Max filter: Each csv file generated from the previous step will be processed and saved separately. All data points that are above the entered ‘Max’ or below entered ‘Min’ will be deleted. The "dspk.StateOfValue" of the deleted values will be set to ‘minmax.state.of.value.code’ (default 91). The data will be saved in a new csv file in the directory “autodespikeYYYYMMDDHHMMSS/step1MinMax” within your working directory.

Step 2: Despiking: Each csv file generated from the previous step will be processed and saved separately. With the default “despike.Method” median and the default “despike.threshold” 3: all data points that are more than 3 median absolute deviations away from the median of the 10 surrounding data points (5 before and 5 after) will be deleted. At least 5 surrounding data points is required for the sample to be evaluated. The algorithm will not look farther than 5 sampling intervals before and after the data point, for handling data gaps. If a “sampling.interval” is not provided then it will be calculated as the mode of the interval between samples. The "dspk.StateOfValue" of the deleted values will be set to “despiked.state.of.value.code” (default 92). The "dspk.StateOfValue" of the values that passed the despike test will be set to “good.state.of.value.code” (default 80). The data will be saved in a new csv file in the directory “autodespikeYYYYMMDDHHMMSS/step2Despike” within your working directory.'

Step 3: Data gap interpolation: Each csv file generated from the previous step will be processed and saved separately. All data gaps will be linear interpolated unless a ‘max.gap’ length for interpolation is given. If a ‘precision’ is given, then the interpolated values will be rounded to that precision. The state of values will not be changed to know what the original state of the value was. If the value has a state of value deleted or missing but there is a value then it can be assumed that it was interpolated. The data will be saved in a new csv file in the directory “autodespikeYYYYMMDDHHMMSS/step3Interpol.FinalData” within your working directory.

Details

Make sure that when you enter the path name for the directory it has forward-slashes(/) or double-back-slashes(\\) and not back-slashes(\). If you copy the directory path from windows, it will have back-slashes(\) and these need to be changed to forward-slashes(/) or double-back-slashes(\\).

Interpolated the values will be rounded to the same decimal places as the original data. If you wish you may enter a custom precision. Examples: 0.566 would be ‘precision = 0.001’ 1200 would be ‘precision = 100’ Measurements with steps of 5 would be ‘precision = 5’ Measurements to the nearest half unit would be ‘precision = 0.5’

You are not required to supply ‘DateTime’ for this function. This is only necessary for handling data gaps while despiking and interpolating the data. If you omit this argument, then it will generate a datetime column which contains the samples numbered consecutively so you can still indicate with ‘max.gap’ the maximum data gap you wish to interpolate by filling in the number of missing samples into ‘max.gap’. The ‘DateTime’ data can be numeric values or as a datetime character strings (e.g. “2018-04-23 15:32:18”). If the datetime data are character strings, then a ‘datetime.format’ must be provided (e.g. datetime.format = " function documentation for help on syntax. If you enter a date time as a character string, then it will be converted into UNIX seconds. The default time zone is GMT but it can be changed to GMT+1 with 'datetime.timezone = "Etc/GMT-1"' Use the OlsonNames() function for a list of all time zones. If you enter a datetime, then the date and time must be in the same cell, as in they cannot be in separate columns. Often a time conversion to GMT is supplied with the time (e.g. 16:32 +2). This function uses the as.POSIXct function and cannot handle these conversions (like the "+2" in the example). They need to be removed and dealt with prior to analysis.

When the data is being formatted the columns "dspk.Values", "dspk.DateTimeNum" and "dspk.StateOfValue" will be generated. If these columns already exist in the data table, then they will be overwritten. This may or may not be desired. If the column "dspk.StateOfValue" is in the original data, then that data will be used for the 'state of values', otherwise the 'unchecked.state.of.value.code' (default 110) will be used for all data signifying 'unchecked' status.

The spike removal algorithm is by default (despike.Method = "median") all sample points that are more than the threshold 3 median absolute deviations (the scale factor 1.4826 is used assuming normal distribution) from the median of the 10 surrounding data points (5 before and 5 after) are automatically deleted. The threshold of 3 can be changed with "despike.threshold =" You can set despike.Method = "mean" to use standard deviations and mean for the algorithm instead of the default median absolute deviations and median. However median is a much more robust statistic for handling outliers.

If you have data gaps and you have the sampling times and you don”t want the despiking algorithm to look past the data gaps, then make sure to to supply “DateTime”. The time interval between samples "sampling.interval = " will be calculated as the mode of the difference between consecutive samples. If there are irregularities in the sampling interval that will prevent this calculation then you can enter in the sampling interval in "sampling.interval = " in the numeric-datetime unit. If you entered in a character datatime column then it is POSIX time converted to numeric so the unit is in UNIX seconds.

The default is that it will do linear interpolation of all data gaps. You can restrict the size of the data gap with "max.gap.interpolate". This needs to be in the unit of your numeric datetime. If you entered in a datetime column containing character strings then it is POSIX time converted to numeric so the unit is in UNIX seconds. If you did not entered a time column, then the unit is in samples, for example "max.gap.interpolate = 5" will only interpolate gaps of up to 5 samples long.

Value

Your final data can be found in “autodespikeYYYYMMDDHHMMSS/step3Interpol.FinalData” within your working directory as comma separated csv files. The cleaned values will be in column "dspk.Values", the state of values in column "dspk.StateOfValue", and the numeric datetime will be in column "dspk.DateTimeNum".

Examples

```
#HIC data cleaning and validation protocol
#Batch process HIC database files that were formatted
#with the HIC.Continuous.Data.Import.Format() function.
#With default despiking algorithm (threshold of 3 MAD from the median).
dspk.DespikingWorkflow.CSVfileBatchProcess(
   input.directory = 'data/HIC.data', #Load csv files from the folder 'HIC.data'
   sep = ',', dec = '.', header = T,  #The csv files are separated by commas with
   #point decimals and the first line is the column names.
   Value = 3, val.NAvalue = -777, #The values are found in the third column.
   #-777 stands for no-value
   DateTime = "DateTimeUnix",
   #The numeric datetime column name.
   #Because it is numeric no datetime formatting info is needed
   ConditionalMinMaxColumn = 'Parameter.Name',
   ConditionalMinMaxValues = c('DO','pH','chfyla','PPFD1','PPFD'),
```

```
   ConditionalMin = c(0,0,0,0,0), ConditionalMax = c(15,15,1000,2000,2000)
   #conditional min max filter based on parameter with minimum reasonable value
   #for oxygen, pH, chlorophyll a, and PPFD being 0 and the maximum
   #reasonable value for oxygen and pH being 15 and chlorophyll 1000 and PPFD being 2000
   max.gap = 3600)
   #The maximum data gap that should be interpolated is one hour or 3600 seconds.

#Example: Running full despiking work flow with batch process from a folder of
#csv files, with default despiking algorithm (threshold of 3 MAD from the median)
dspk.DespikingWorkflow.CSVfileBatchProcess(
   input.directory = 'data/PPFD.data', #Load csv files from the folder 'PPFD.data'
   sep = ',', dec = '.', header = T,  #The csv files are separated by commas with
   #point decimals and the first line is the column names.
   Value = 3, val.NAvalue = -777, #The values are found in the third column.
   #-777 stands for no-value
   DateTime = "DateTime",
   #The datetime column name is "DateTime".
   datetime.format = "%Y-%m-%d %H:%M:%S",
   #The datetimes are character strings with this format "2018-04-23 15:32:18".
   datetime.timezone = 'Etc/GMT-1', #The time zone is UTC+1.
   Min=(-50), Max=1700, #The minimum reasonable value is -50 and the maximum  is 1700
   sampling.interval = NULL,
   #Sampling interval is regular and it will be calculated from the provided data
   precision = NULL, #The data precision will be calculated from the data.
   max.gap = 3600)
   #The maximum data gap that should be interpolated is on hour or 3600 seconds.

#Example: Max filter on a vector with interpolation of resulting data gaps of up to 2 records
example.data = c(2,2,4,16,-4,2,0,96,8,12,26,66,2)
dspk.DespikingWorkflow.CSVfileBatchProcess(
   steps = c(1,3), #run step 1 min/max filter and step 3 data gap interpolation
   Value = example.data, #The values are found in the vector 'example.data'
   Max=10, #Filter out all values above 10
   sampling.interval = 60,
   #This is extraneous information since no time data was given, it will be ignored
   precision = 2, #The data are all even so precision is set to 2.
   max.gap = 2)  #The maximum data gap that should be interpolated is 2 missing values.
>Output CSV file:
>"Value","dspk.Values","dspk.DateTimeNum","dspk.StateOfValue"
>  2,         2,               1,               110
>  2,         2,               2,               110
>  4,         4,               3,               110
>  16,        0,               4,               91
>  -4,       -4,               5,               110
>  2,         2,               6,               110
>  0,         0,               7,               110
>  96,        4,               8,               91
>  8,         8,               9,               110
>  12,        NA,              10,              91
>  26,        NA,              11,              91
>  66,        NA,              12,              91
>  2,         2,               13,              110

#Example: Full despiking work flow on an R dataframe with no data gaps.
#999 is the NA value. Despiking is done with a threshold of 4 using the
#standard deviations from the mean.
dspk.DespikingWorkflow.CSVfileBatchProcess(
     Value = datatable$values, val.NAvalue = 999,
     #The values are found in datatable$values. 999 stands for no-value
     Min=(0), Max=200, #The minimum resonable value is 0 and the maximum  is 200
     despike.threshold = 4, despike.Method = "mean",
     #The despiking algorithm is all data points more than 4 standard deviations
     #from the mean of the surrounding 10 data points
     precision = 0.01, #The data has a precision to the hundredth decimal place.
     max.gap = 10) #the maximum data gap that should be interpolated is 10 samples.

#Example: The same example as above but now entering in the dataframe in
#the Data =' argument.
dspk.DespikingWorkflow.CSVfileBatchProcess(
     Data = datatable, Value = 'values', val.NAvalue = 999,
     #The values are found in datatable$values. 999 stands for no-value
     Min=(0), Max=200, #The minimum reasonable value is 0 and the maximum is 200
     despike.threshold = 4, despike.Method = "mean",
     #The despiking algorithm is all data points more than 4 standard deviations
     #from the mean of the surrounding 10 data points
     max.gap = 10)  #the maximum data gap that should be interpolated is 10 samples.

#Example: Full despiking work flow with a conditional min max filter. No Min or Max
#was given so if the conditions are not met, then the min and max will be set to
#infinity.
dspk.DespikingWorkflow.CSVfileBatchProcess(
     Data = datatable, Value = "values", #The values are found in datatable$values.
     ConditionalMinMaxColumn = "Parameter.Name",
     #The factors for basing the conditional min max are in column "Parameter.Name"
     ConditionalMinMaxValues = c('PercentO2','Temp'),
     ConditionalMin = c(0,-30), ConditionalMax = c(100,150)
     #conditional min max filter based on parameter with minimum reasonable value
```

```
     #for percent oxygen being 0% and for temperature being -30C and maximum
     #reasonable value for percent oxygen being 100% and for temperature being 150C
     max.gap = 10) #The maximum data gap that should be interpolated is 10 samples.

#Example: Full despiking work flow with a conditional min max filter. A Min and a
#Max was now given so if the conditions are not met, then the min and max will be
#set to 10 and 100. If you give a Min and a Max in addition to the conditional
#values, then if the conditions are not met the min and the max will be set to
#those values given.
dspk.DespikingWorkflow.CSVfileBatchProcess(
     Data = datatable, Value = "values", #The values are found in datatable$values.
     Min = 10, Max = 100, #if the bellow conditional min max values are not met,
     #then it will take these values as the min and max
     ConditionalMinMaxColumn = "Parameter.Name",
     #The factors for basing the conditional min max are in column "Parameter.Name"
     ConditionalMinMaxValues = c('PercentO2','Temp'),
     ConditionalMin = c(0,-30),
     ConditionalMax = c(100,150)
     #conditional min max filter based on parameter with minimum reasonable value
     #for percent oxygen being 0% and for temperature being -30C and maximum
     #reasonable value for percent oxygen being 100% and for temperature being 150C
     max.gap = 10) #The maximum data gap that should be interpolated is 10 samples.
```

Batch process: Despike and autovalidate paired PPFD data and calculate light attenuation coefficient kd

Description

This function was designed to auto-validate paired continuous PPFD data where the two sensors are placed at a fixed distance from each other to calculate the light attenuation coefficient kd. It takes a folder containing pairs of csv files of upper and lower PPFD data and calculates the light attenuation coefficient based on those PPFD values and performs an auto-validation process on those data.

Usage

```
HIC.PPFDAutoValidation.CSVfileBatchProcess(
  input.directory = NULL,
  sep = ",",
  dec = ".",
  header = T,
  DataUpper = NULL,
  DataLower = NULL,
  Value,
  val.NAvalue = NULL,
  unchecked.state.of.value.code = 110,
  add.original.data = T,
  DateTime = NULL,
  datetime.format = NULL,
  datetime.timezone = "GMT",
  ConditionalMinMaxColumn = NULL,
  ConditionalMinMaxValues = NULL,
  ConditionalMin = NULL,
  ConditionalMax = NULL,
  Min = 0,
  Max = 2000,
  minmax.state.of.value.code = 91,
  sampling.interval = NULL,
  despiked.state.of.value.code = 92,
  good.state.of.value.code = 80,
  despike.threshold = 3,
  despike.Method = "median",
  precision = NULL,
  max.gap = Inf,
  DeletedSpikeOtherSensor.state.of.value.code = 94,
  NotDeletedSpikeBothSensors.state.of.value.code = 95,
  kdDespiked.state.of.value.code = 96,
  DeletedNegativeKd.state.of.value.code = 97,
  NA.state.of.value.code = 255,
  kddlupper = 1,
  kddllower = 0.25,
  Dist.Sensors = 0.4,
  UpperSensorParameterName = "PPFD1",
  LowerSensorParameterName = "PPFD"
)
```

Arguments

`sep`	Arguments indicating the formatting of the input csv files. It is the field separator character. Values are separated by this character. By default it is comma ",".
`dec`	Arguments indicating the formatting of the input csv files. It the character used for decimal points. By default if is a period ".".
`header`	Arguments indicating the formatting of the input csv files. It is a logical value indicating if the first line is the column titles. By default it is TRUE.
`DataUpper`	A dataframe object for the upper sensor. If you only wish to process one data frame, then it can be entered directly from the R environment with this argument. If you enter in an input.directory then 'DataUpper = ' will be ignored and the files from the input directory will be processed.

`DataLower`

A dataframe object for the lower sensor. If you only wish to process one data frame, then it can be entered directly from the R environment with this argument. If you enter in an input.directory then 'DataLower = ' will be ignored and the files from the input directory will be processed.

`Value`

If the data is from a csv file or a dataframe, it is a quoted character string indicating the column name or an integer indicating the column number of the column containing the data values that you wish to despike. Data may also be entered as a single vector object (unquoted) such as ‘Value = mydata$values’ or ‘Value = values’

`val.NAvalue`

The value indicating an NA value in your input data. If this value is NA, then this argument can be omitted.

`unchecked.state.of.value.code`

Number indicating that a given value is unchecked. By default 110.

`add.original.data`

A logical value indicating if the original input data should be included in the output tables. Note that if you input a csv file, then every column in that file will be kept. TRUE by default.

`DateTime`

If the data is from a csv file, it is a quoted character string indicating the column name or an integer indicating the column number of the column containing the datetime values of the samples. Data may also be entered as a single vector object (unquoted) such as ‘DateTime = mydata$time’ or ‘DateTime = time’

`datetime.format`

Character string giving the datetime format. See the strptime() help file for additional help.

`datetime.timezone`

Character string giving the time zone of the datetime. By default “GMT”. Use OlsonNames() for a list of all time zones.

`ConditionalMinMaxColumn`

The column name in quotes or column number or vector object that contains the factor variable to base your conditional min max filter on

`ConditionalMinMaxValues`

A vector containing the factor values to base the conditional min max filter on

`ConditionalMin`

A vector containing the condition minimums that correspond to the respective values in ConditionalMinMaxValues

`ConditionalMax`

A vector containing the condition maximums that correspond to the respective values in ConditionalMinMaxValues

`Min`

Number giving the minimum reasonable value. All values below this will be deleted.

`Max`

Number giving the maximum reasonable value. All values above this will be deleted.

`minmax.state.of.value.code`

Number indicating that the value has been deleted during the min max filter. By default 91.

`sampling.interval`

As numeric, the time between samples. If you enter NULL then it will calculate it for you. By default NULL.

`despiked.state.of.value.code`

Number indicating that a given value was deleted during the despiking. By default 92.

`good.state.of.value.code`

Number indicating that a given value has been check and deemed not a spike during the despiking. By default 80.

`despike.threshold`

Number indicating the threshold for defining a spike. By default it is 3, which corresponds to 3 median absolute deviations or 3 standard deviations.

`despike.Method`

Character string "median" or “mean” indicating the method to use for the despiking. By default “median”.

`precision`

A number indicating the precision of the input values. Interpolated values will be rounded to this precision. If left as NULL then the numbers will be rounded to the largest decimal length found in the data.

`max.gap`

As numeric, the time span of the maximum data gap you wish to interpolate.

`DeletedSpikeOtherSensor.state.of.value.code`

State of value code given to values deleted because they were deleted in the other sensor.

`NotDeletedSpikeBothSensors.state.of.value.code`

State of value code given to values that were spikes in both sensors and thus not deleted.

`kdDespiked.state.of.value.code`

State of value code given to values that were deleted because they were spikes in the light attenuation coefficient kd.

`DeletedNegativeKd.state.of.value.code`

State of value code given to values that were deleted because the light attenuation coefficient kd was negative.

`NA.state.of.value.code`

State of value code given to missing data.

`kddlupper`

Minimum PPFD value of the upper sensor for calculating light attenuation coefficient kd.

`kddllower`

Maximum PPFD value of the upper sensor for calculating light attenuation coefficient kd.

`Dist.Sensors`

Distance between PPFD sensors for calculating light attenuation coefficient kd.

`UpperSensorParameterName`

Upper sensor parameter name in the input file name.

`LowerSensorParameterName`

Lower sensor parameter name in the input file name.

`input.directoryCharacter`

string of the path to the folder containing all the csv files that you wish to batch process. This argument may be omitted if you are entering vectors directly into the ‘Value’ and ‘DateTime’ arguments.

Details

All the algorithms in this function are the same as in the function dspk.DespikingWorkflow.CSVfileBatchProcess(). See the documentation for that function for more details.

This function can either batch process paired files in an input directory placed in the argument input.directory or it can process two R objects placed in the arguments DataUpper and Datalower. Specify the input directory in quotes and with forward-slashes(/) or double-back-slashes(\\) but no back-slashes(\). If you copy the directory path from windows, it will have back-slashes(\) and these need to be changed to forward-slashes(/) or double-back-slashes(\\).

Input directory must be a folder containing all the csv files of the PPFD data with the upper and lower sensor data files together with unique station names for each pair of files and consistent separate parameter IDs for the upper and lower sensors also in the file names. For example the files in your directory may be station1.PPFD1.csv, station1.PPFD.csv, station2.PPFD1.csv, station2.PPFD.csv where you have two stations (station1 and station2) and you have an upper sensor ID (PPFD1) and a lower sensor ID (PPFD). You must give the 'UpperSensorParameterName' and the 'LowerSensorParameterName' in order for the function to know which files belong to which sensor. This is not a generic function because it assumes that the components of the csv file names are separated by decimals(.) and that when the files are arranged alphabetically, the upper and lower sensor pairs will be next to each other.

Please see the FunctionLogFile.txt that was generated to see any error messages and details about the selected preferences and calculated preferences.

Default state of value codes

255 missing data
110 unchecked data
80 good data
91 deleted, min max filter
92 deleted, despiked
94 deleted, spike in other sensor
95 not deleted, spike in both sensors
96 deleted, spike in kd
97 deleted, negative kd

General work flow

Preprocess-formatting. Min max filter on PPFD. Despike PPFD data. All values deleted in one sensor dataset must be deleted in the other as well. Restore original values where spikes are in both sensor datasets. Data gap interpolation PPFD (default max gap to interpolate 1 hour). Calculate light attenuation coefficient kd. Delete PPFD values where kd is negative. Remove kd values where PPFD is below detection limit. Despike kd and delete those spikes from both the PPFD data and the kd data. Data gap interpolation PPFD again. Calculate light attenuation kd again. Delete kd values where PPFD is below detection limit.

Detailed work flow overview

All data is saved to a folder 'autoPPFDdespikeYYYYMMDDHHMMSS' in your working directory. Formatted data is saved to the subfolder 'preprocFormat'. The columns 'dspk.Values', 'dspk.DateTimeNum', and 'dspk.StateOfValue' are created to not overwrite the original data. The original data is saved into the new column 'orig.values' for ease of later reference. Unchecked data is given the state of value code 110 (default) and missing data is coded as 255 (default).

PPFD data is first min/max filtered removing unreasonably high and low values (defaults are min 0 corresponding to no light and max 2000 corresponding to full sun). Deleted values are coded as 91 (default).

PPFD data is then despiked. Checked values coded as 80 (default), deleted as 92 (default) and unchecked values are left with their original state of value code. See documentation for dspk.DespikingWorkflow.CSVfileBatchProcess() for details on the despiking algorithm.

Merge the two datasets joining on time. The upper sensors column have the suffix x and the lower sensors columns have the suffix y.

All values that were deleted in either the upper sensor's dataset or the lower sensors dataset must be deleted in both datasets and coded as 94 (default). This must be done because the two datasets are compared to each other to calculate light attenuation coefficient kd.

If a spike was detected in both the upper and lower datasets, restore their original values and code as 95 (default). Spikes were often found in both datasets simultaneously which means it wasn't sensor error. These may have been passing clouds or plumes of suspended matter which is important data for understanding the light climate.

Despiked PPFD data is saved to the subfolder 'step2Despike'.

PPFD data gaps of maximum one hour (default) are linear interpolated. The stat of value codes are not changed. If the state of value code says it was deleted or was missing but there is a value, then it can be assumed that it was interpolated.

Light attenuation coefficient kd is calculated as kd = 1/Δz*ln(E1/E2) where Δz is the distance between sensors in meters 0.4m (default) and E1 is the upper sensor PPFD and E2 is the lower sensor PPFD. Saved to column ‘kd’. Unite m^-1.

Data saved as csv files to the subdirectory ‘step3Interpol.kd’.

Delete all PPFD data values where kd is negative. State of value code 97 (default). Light cannot be greater lower in the water column.

Copy the column ‘kd’ to the new column 'dspk.kd' and remove all values from 'dspk.kd' outside detection limits 1 PPFD (default) for the upper sensor and 0.25 PPFD (default) for the lower sensor. When the light levels approach zero, it becomes too difficult to accurately measure the difference between the upper and lower sensors.

Despike ‘dspk.kd’ and delete those spikes also from the PPFD columns 'dspk.Values.x' and 'dspk.Values.y'. State of value code 96 (default).

Make a unified state of value for kd. Copy ‘dspk.StateOfValue.x’ into the new column ‘dspk.StateOfValue’ and make all 91 and 92 codes the code 94 (these are the default codes).

Interpolate PPFD data gaps of maximum one hour (default) again.

Calculate kd light attenuation coefficient again in column ‘dspk.kd’.

Delete all ‘dspk.kd’ values where the PPFD values are outside the detection limit: upper sensor PPFD is less than 1 (default) and lower sensor is less than 0.25 (default).

Save the final dataset in subfolder ‘step4Despikekd.FinalData’

Value

Each pair of input files gets outputted as one csv file. All data is saved to a folder ‘/autoPPFDdespikeYYYYMMDDHHMMSS’ in your working directory. The final data will be in the subfolder /step4Despikekd.FinalData. Upper sensor data has suffix .x and lower sensor data has suffix .y. The original PPFD data is in columns orig.values.x and orig.values.y The despiked PPFD data is in columns dspk.Values.x and dspk.Values.y. The PPFD state of value codes are in columns dspk.StateOfValue.x and dspk.StateOfValue.y The despiked light attenuation coefficient kd values are in column dspk.kd. The state of value codes for kd are in column dspk.StateOfValue. The UNIX seconds datetime is in column dspk.DateTimeNum.

Examples

```
#HIC data cleaning and validation protocol
#Batch process HIC database PPFD files that were formatted
#with the HIC.Continuous.Data.Import.Format() function.
HIC.PPFDAutoValidation.CSVfileBatchProcess(
    input.directory = "C:/Rdata/PPFDdata",
    Value = "Value", val.NAvalue = -777, #all values of -777 will be set to NA
    DateTime = "DateTimeUnix",
    max.gap = 3600) #3600 seconds or 1 hour maximum gap to interpolate

#Process the r object tables PPFDuppersensor and PPFDlowersensor on the column "Value".
HIC.PPFDAutoValidation.CSVfileBatchProcess(DataUpper = PPFDuppersensor,
    DataLower = PPFDlowersensor, Value = "Value")
```

Graphical applications for formatting, auto-validation, manual-validation and final export of HIC database data

Description

Three graphical user interfaces (GUI) for the processing of the Flemish Hydrological Information Center (HIC) data. These graphical apps will walk you through the entire work flow of data import, validation/calibration and data export, in an intuitive visual manner without the need for coding. This is the only method of manual validation and calibration and final export.

Usage

```
HIC.App.format()
```

Details

Either enter "format" to format HIC database data, "auto" to auto-validate formatted data, or "manual" to manually check and calibrate auto-validated data against reference site locations (if available) and export final zrx files for upload back into the HIC database.

The formatting step and the auto validation step can be done in R code using the functions HIC.Continuous.Data.Import.Format(), spk.DespikingWorkflow.CSVfileBatchProcess(), and HIC.PPFDAutoValidation.CSVfileBatchProcess(). The manual validation, calibration and final export must be done using this graphical app.

Recomended workflow

First: format the csv files that are exported from the HIC database - HIC.App.format()

Second: auto-validate the formatted files - HIC.App.auto()

Third: manual-validate/calibrate the files and export the data - HIC.App.manual()

Files can be batch processed at each step so there is no need to run through all the steps for each individual file.

Manual Tutorial

https://github.com/pgelsomini/HICbioclean/blob/master/MANUAL-TUTORIAL-HICbioclean.-Rpackage.pdf

Value

Three shiny apps

Examples

```
HIC.App.format() # to format the HIC database output csv files
HIC.App.auto() # to auto-validate the formatted data
HIC.App.manual() # to manually check the auto-validated data
```

Graphical applications for formatting, auto-validation, manual-validation and final export of HIC database data

Description

Three graphical user interfaces (GUI) for the processing of the Flemish Hydrological Information Center (HIC) data. These graphical apps will walk you through the entire work flow of data import, validation/calibration and data export, in an intuitive visual manner without the need for coding. This is the only method of manual validation and calibration and final export.

Usage

```
HIC.App.auto()
```

Details

Either enter "format" to format HIC database data, "auto" to auto-validate formatted data, or "manual" to manually check and calibrate auto-validated data against reference site locations (if available) and export final zrx files for upload back into the HIC database.

The formatting step and the auto validation step can be done in R code using the functions HIC.Continuous.Data.Import.Format(), spk.DespikingWorkflow.CSVfileBatchProcess(), and HIC.PPFDAutoValidation.CSVfileBatchProcess(). The manual validation, calibration and final export must be done using this graphical app.

Recomended workflow

First: format the csv files that are exported from the HIC database - HIC.App.format()

Second: auto-validate the formatted files - HIC.App.auto()

Third: manual-validate/calibrate the files and export the data - HIC.App.manual()

Files can be batch processed at each step so there is no need to run through all the steps for each individual file.

Manual Tutorial

https://github.com/pgelsomini/HICbioclean/blob/master/MANUAL-TUTORIAL-HICbioclean.-Rpackage.pdf

Value

Three shiny apps

Examples

```
HIC.App.format() # to format the HIC database output csv files
HIC.App.auto() # to auto-validate the formatted data
HIC.App.manual() # to manually check the auto-validated data
```

Graphical applications for formatting, auto-validation, manual-validation and final export of HIC database data

Description

Three graphical user interfaces (GUI) for the processing of the Flemish Hydrological Information Center (HIC) data. These graphical apps will walk you through the entire work flow of data import, validation/calibration and data export, in an intuitive visual manner without the need for coding. This is the only method of manual validation and calibration and final export.

Usage

```
HIC.App.manual()
```

Details

Either enter "format" to format HIC database data, "auto" to auto-validate formatted data, or "manual" to manually check and calibrate auto-validated data against reference site locations (if available) and export final zrx files for upload back into the HIC database.

The formatting step and the auto validation step can be done in R code using the functions HIC.Continuous.Data.Import.Format(), spk.DespikingWorkflow.CSVfileBatchProcess(), and HIC.PPFDAutoValidation.CSVfileBatchProcess(). The manual validation, calibration and final export must be done using this graphical app.

Recomended workflow

First: format the csv files that are exported from the HIC database - HIC.App.format()

Second: auto-validate the formatted files - HIC.App.auto()

Third: manual-validate/calibrate the files and export the data - HIC.App.manual()

Files can be batch processed at each step so there is no need to run through all the steps for each individual file.

Manual Tutorial

https://github.com/pgelsomini/HICbioclean/blob/master/MANUAL-TUTORIAL-HICbioclean.-Rpackage.pdf

Value

Three shiny apps

Examples

```
HIC.App.format() # to format the HIC database output csv files
HIC.App.auto() # to auto-validate the formatted data
HIC.App.manual() # to manually check the auto-validated data
```

Mode

Description

Mode

Usage

```
dspk.getmode(v)
```

Arguments

v	A vector. May be a string, numeric, ect.

Value

The mode of the input vector

Min max filter

Description

Mininum maximum filter.

Usage

```
dspk.MinMaxfilter(
  Data = NULL,
  Value,
  Min,
  Max,
  State.of.value.data = NULL,
  state.of.value.code = 91,
  default.state.of.value.code = 110,
  NAvalue = NULL,
  logoutput = F
)
```

Arguments

`Data`	A datatable or NULL
`Value`	A vector, datatable column name as a string or datatable column number as numeric
`Min`	Minimum value to keep
`Max`	Maximum value to keep
`State.of.value.data`	A vector, datatable column name as a string or datatable column number as numeric
`state.of.value.code`	number that deleted values will be taged with
`default.state.of.value.code`	number that values will be tagged with if no State.of.value.data is given
`NAvalue`	Value that should be read as NA
`logoutput`	TRUE if you want to have a logged record of what the function did

Value

returns a datatable with columns dspk.Values and dspk.StateOfValue containing the filtered data. If logoutput is TRUE, then $data contains the datatable and $logdata contains the info for the log file

Examples

```
SomeValues <- c(5,NA,NA,NA,5,3,2,2,3,NA,8,2,3,3)
SomeTimes <- c(1,2,3,4,5,6,7,8,9,22,23,24,25,26)
ADataframe <- data.frame(SomeValues,SomeTimes)

#entering in a vector into the function.
dspk.MinMaxfilter(Value = SomeValues, Min = 3,Max = 5)
#entering a dataframe and column name into the function
dspk.MinMaxfilter(Data = ADataframe, Value = "SomeValues", Min = 3,Max = 5)
#entering a dataframe and column number into the function
dspk.MinMaxfilter(Data = ADataframe, Value = 1, Min = 3,Max = 5)
```

Spike removal algorithm

Description

Outlier removal function using standard deviation or median absolute deviation thresholds from the 10 surrounding datapoints.

Usage

```
dspk.Spikefilter(
  Data = NULL,
  Value,
  NumDateTime = NULL,
  sampling.interval = NULL,
  State.of.value.data = NULL,
  state.of.value.code = 92,
  good.state.of.value.code = 80,
  default.state.of.value.code = 110,
  NAvalue = NULL,
  threshold = 3,
  Method = "median",
  logoutput = F
)
```

Arguments

`Data`	A dataframe. Leave as NULL if you are entering in vectors directly.
`Value`	A vector, datatable column name as a string or datatable column number as numeric
`NumDateTime`	Numeric datetime. A vector, datatable column name as a string or datatable column number as numeric
`sampling.interval`	Time between samples for use in handling data gaps. May be left as NULL and the function will calculate it.
`State.of.value.data`	A vector, datatable column name as a string or datatable column number as numeric
`state.of.value.code`	Number that deleted values are marked with
`good.state.of.value.code`	Number that checked values are marked with
`default.state.of.value.code`	Number that unchecked values are marked with if no State.of.value.data was provided.
`NAvalue`	Value that is read as NA
`threshold`	Number indicating the threshold for defining a spike. By default it is 3, which corresponds to 3 median absolute deviations or 3 standard deviations.
`Method`	Character string "median" or “mean” indicating the method to use for the despiking. By default “median”.
`logoutput`	TRUE if you want to have a logged record of what the function did

Details

With the default “Method” median and the default “threshold” 3: all data points that are more than 3 median absolute deviations away

from the median of the 10 surrounding data points (5 before and 5 after) will be deleted. At least 5 surrounding data points is required for the sample to be evaluated. The algorithm will not look farther than 5 sampling intervals before and after the data point, for handling data gaps. If a “sampling.interval” is not provided then it will be calculated as the mode of the interval between samples.

To keep track of what vales were evaluated and removed the output "dspk.StateOfValue" is generated. The state of value of the deleted values will be set to “state.of.value.code” (default 92). The state of value of the values that passed the despike test will be set to “good.state.of.value.code” (default 80). If the value was unchecked (due to too few point) then the state of value will be left as is. If the no "sate.of.value.data" was provided then the state of value for unckecked will be "default.state.of.value.code" (default 110).

Value

returns a datatable with columns dspk.Values and dspk.StateOfValue containing the filtered data. If logoutput is TRUE, then $data contains the datatable and $logdata contains the info for the log file

Examples

```
SomeValues <- c(5,6,2,3,5,66,2,2,3,69,8,2,3,3)
SomeTimes <- c(1,2,3,4,5,6,7,8,9,22,23,24,25,26)
ADataframe <- data.frame(SomeValues,SomeTimes)

#entering in a vector into the function.
dspk.Spikefilter(Value = SomeValues)
#entering a dataframe and column name into the function
dspk.Spikefilter(Data = ADataframe, Value = "SomeValues")
#entering a dataframe and column number into the function
dspk.Spikefilter(Data = ADataframe, Value = 1)

#If you have data gaps then provide sampling times so that the
#function won't compare data that isn't actually next to each
#other in time. The time must be provided as a numeric class.
dspk.Spikefilter(Value = SomeValues, NumDateTime = SomeTimes)


#-------------------------------------------------------------------------
# If you enter in no "Method" or "threshold" it will evaluate the
#despiking using the default method of a threshold of 3 median absolute
#deviations from the median.

#Running the despiking using the threshold of 5 median absolute deviations from the median.
dspk.Spikefilter(Value = SomeValues, threshold = 5)
#Running the despiking using the threshold of 5 standard deviations from the mean.
dspk.Spikefilter(Value = SomeValues, Method = "mean", threshold = 5)
```

Linear interpolation

Description

Linear interpolation between gaps. Excludes gaps greater than "max.gap". If no "NumDateTime" is provided then max.gap is in the unit samplings. If a "NumDateTime" is provided then max.gap must be in the same unit. The entered in "NumDateTime" must be numeric class.

Usage

```
dspk.DataGapInterpolation(
  Data = NULL,
  Value,
  precision = NULL,
  NumDateTime = NULL,
  max.gap = Inf,
  State.of.value.data = NULL,
  state.of.value.code = 93,
  default.state.of.value.code = 110,
  NAvalue = NULL,
  logoutput = F
)
```

Arguments

`Data`	A dataframe. Leave as NULL if you are entering in vectors.
`Value`	A vector, datatable column name as a string or datatable column number as numeric
`precision`	If you enter in a precision then it will round the interpolated values. Precision is the is the smallest measurable unit on the scale. e.g. 13000 would have precision = 1000 and 0.23 would have precision = 0.01
`NumDateTime`	A Numeric datetime. Vector, datatable column name as a string or datatable column number as numeric
`max.gap`	is the largest data gap that you want to perform linear interpolation on.
`State.of.value.data`	A vector, datatable column name as a string or datatable column number as numeric
`state.of.value.code`	A number that lables all values that were interpolated
`default.state.of.value.code`	Number that values are marked with if no State.of.value.data was provided.
`NAvalue`	This values is read as NA
`logoutput`	TRUE if you want to have a logged record of what the function did

Value

returns a datatable with columns dspk.Values and dspk.StateOfValue containing the interpolated data. If logoutput is TRUE, then $data contains the datatable and $logdata contains the info for the log file

Examples

```
SomeValues <- c(5,NA,NA,NA,5,3,2,2,3,NA,8,2,3,3)
SomeTimes <- c(1,2,3,4,5,6,7,8,9,22,23,24,25,26)
ADataframe <- data.frame(SomeValues,SomeTimes)
```

```
#entering in a vector into the function.
dspk.DataGapInterpolation(Value = SomeValues)
#entering a dataframe and column name into the function
dspk.DataGapInterpolation(Data = ADataframe, Value = "SomeValues")
#entering a dataframe and column number into the function
dspk.DataGapInterpolation(Data = ADataframe, Value = 1)

#If you don't want gaps larger than 2 samplings to be interpolated
dspk.DataGapInterpolation(Value = SomeValues, max.gap = 2)

#If you have data gaps then provide sampling times so that the
#function will take the sampling times into account when assesing max.gap.
#The time must be provided as a numeric class and max.gap must be in the
#same unit as your provided time.
dspk.DataGapInterpolation(Value = SomeValues, NumDateTime = SomeTimes, max.gap = 2)
```

Continuous measurement sites and their reference sites.

Description

Dataset with the site numbers of the continuously measured biological parameters and their nearest periodically measured reference site.

Usage

```
ReferenceSiteLinkage
```

Format

A data frame with 2 variables:

MonthlySites

: Site names of the nearest periodically measured site

ContinuousSites

: Site numbers of the continuously measured sites

...

Source

Flemish Hydrological Information Center, OMES Monitoring

Import codes for the Flemish Hydrological Information Center database .

Description

Codes for each site number and parameter for importation into the Flemish Hydrological Information Center database.

Usage

```
zrxFileStationCodes
```

Format

A data frame with 4 variables:

StationNo

: Continuous measurement station number

Par

: Parameter

Code

: Database code

Unit

: Unit

...

Source

Flemish Hydrological Information Center
