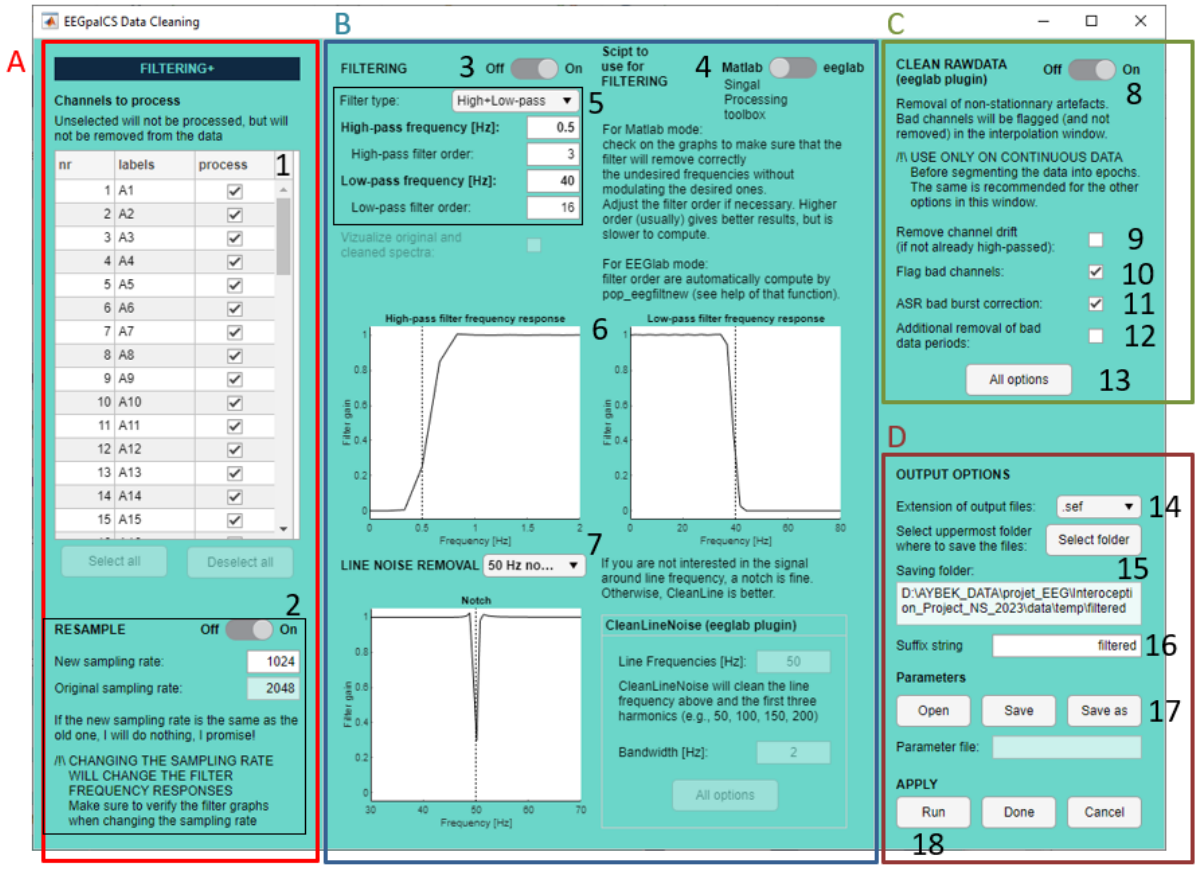
EEGpal: **Filtering+ module**

Version 1.3, 27.10.2024

The 'Filtering+' module can be used to resample the data and filter the unwanted frequency in the EEG signal. The user can choose between two different algorithms to perform this filtering. By default, it uses the Signal Processing Toolbox from Matlab, but it can also call the filtering function used in EEGLAB. In addition, the module can call several EEGLAB tools to clean the data (such as CleanLine, ASR or automatic bad channel detection).



Pannel A

1. You can remove channel from the processing.
2. You can resample your data to another sampling frequency (new frequency must be specified in Hz). This is typically used when you want to downsample your data.

Pannel B

3. Activate of deactivate the filtering (to clean and remove frequency of no interest)
4. Choose between the two modes of filtering:
    a. Matlab mode: It will use the *filtfilt* function include in the Signal Processing toolbox (addone to Matlab)

b. eeglab mode: It will use the *pop_eegfiltnew* function include in the eeglab toolbox (include in EEGpal)

5. The parameters of the filter depend on the mode:
    a. In Matlab mode, the user can choice between **High+Low pass** which correspond to a passband filter (apply successively a high pass then a low pass), **Low pass** or **High Pass**. You can also specify the order of the filters. This determines how sharp the cut-off frequency will be. The higher the value, the sharper the cut-off frequency and the longer the processing time. You can see the effect of the order on the graph in position **6**. WARNING: If you choose a filter order that is too high, you will have more problems with phase shift (temporal distortion we don't want in the EEG) or Gibb's ringing artefact (creates unwanted oscillations in the EEG signal).

    b. In eeglab mode, the high+low pass is replaced by a **bandpath** (more detail in FAQ). You still have the option to perform a **Low pass** or **High Pass** independently**.** In this mode, the filter order will be determined automatically (more details in FAQ). This order will depend on cutting frequency and the sampling rate. If these two values are identical through your dataset, the order will remind the same across participants. This value is displayed on the Matlab consol windows during the processing.

6. View the cut-off behaviour according to the threshold and order of the filter (only available in Matlab mode).
7. Remove line noise: The EEG signal is contaminated by noise from the power line. This noise is usually at 50Hz everywhere except in the USA where it is at 60Hz. However, to be sure, you can check the frequency of the line power for your country at https://www.generatorsource.com/Voltages_and_Hz_by_Country.aspx.
   To remove this noise, you can use a notch filter (default option in Matlab mode) which required the Signal Processing toolbox. Alternatively, you can use the Cleanline module of EEGLAB which remove also all the harmonic frequencies but take longer to process. For more details, please visit the following webpage: https://eeglab.org/plugins/cleanline/.
   The notch filter is disabled for EEGLAB mode because it required Signal Processing toolbox Matlab addone (better to use Cleanline in this case).

Pannel C

8. You can activate the pluging **clean rawdata** from EEGLAB. The next points (9-13) will details the possible option offer by this pluging. For more details, please consult this following webpage: https://eeglab.org/plugins/clean_rawdata/
9. **Remove channels drift:** This correction is useless if you have performed a low-pass filter in point 5 because it does the same job. So, in most of the case, don't use this correction. Also, it required the Signal Processing Toolbox of Matlab.
10. **Flag Bad channels**: It will automatically detect bad channels WITHOUT removing them. It will only mark them to help you during the interpolation step. The results will be visible in the Interpolation module (via the CleanRawData slider). In this way, you can always decide whether you agree with the suggestion or not.
    Note: The authors are not convinced by this detection and do not agree with most of the suggestions.
11. **ASR bad burst detection**: This applies the very popular Artifact Subspace Reconstruction (ASR) correction. It permits to clean signal from artefact burst like eye-blink, participant

movement, etc… It is a non-stationary method (i.e., it uses sliding window PCA). It is important to note that it correct the signal and does not remove any time point.

12. **Additional removal bad data periods**: This performs the final window rejection. If a sliding window (default 1.0 s. overlap 66%) finds more than given percentage of bad channels even after ASR, the window is rejected. By disabling this function, you can keep the data length to be the same between before and after the processing, if necessary.
    The authors advice to not use this option because a similar correction would be performed by the Epoching module.
13. You can access the parameters for the various corrections (9-12). However, unless you know what you are doing, we strongly advise you to leave the default value.

Pannel D

14. Select the format for the output files.
15. Select the destination folder where the results files will be saved (note: it reproduces the input structure. For example, a folder per participants if the input files where in subfolder).
16. The suffix added to the input filename to obtain the output filename
17. You can save a parameters file which will recode all the chosen options for a later processing (**save** and **save as**). You can use the button **open** to call a previous saved parameters file.
18. Click on **Run** to carry out the processing parameterized in the Filerting module. The button **Done** will close the Filtering module without perform the processing but keep in memory your parameters if you open again the Filerting module. The button **Cancel** closes the module without processing and without keep the entered parameters in memory.

FAQ

Should I use Signal Processing Toolbox (*filtfilt* function) or EEGLAB (*pop_eegfiltnew* function) to perform my filtering?

Difficult questions. The results are similar for these two options. The authors found that the quality of the filtfilt Matlab filter was slightly better. However, the EEGLAB filters can be used without the Signal Processing Toolbox license and are more widely cited in the scientific literature.

Why I can choose the filter order in Matlab mode and not in eeglab mode?

The two filtering functions use filter order with different scales. To avoid confusion, the authors decided to impose an automatic choice of these values by EEGLAB (default option).

Why is the high+low pass has been replaced by a bandpass option in eeglab mode?

The authors observed a weird result when applying a high pass then a low pass filter with the function *pop_eegfiltnew*. It is why they have decided to propose a bandpass filter as in the original filtering GUI in EEGlab.

In Matlab mode, why the option passband has been replaced by High-pass + Low-pass?

This is a choice of Michael De Pretto after reading of this reference: XXXX. The result is better by applying successively a high pass filter then the low pass filter.

Why doesn't the toolbox offer the option of DC removal of the baseline shift, which is common in other EEG processing software?

In standard EEG pre-processing, we recommend always applying a high pass filter to remove low frequency noise such as signal drift and others. The baseline shift due to DC is included in this type of noise (very low frequency). So, DC removal is useless if you apply a high pass filter with a cut-off between 0.3 and 0.5 Hz.

Is it a good idea to perform ICA after an ASR correction?

Again, this is a difficult question. Some people don't like to use both, for fear of losing too many signals of interest. However, the creator of the ASR correction claims that these two corrections are complementary (ASR == good at removing occasional large-amplitude noise/artifacts, ICA == good at decomposing constant fixed-source noise/artifacts/signals). So there is no consensus and both views are accepted in the literature.

Why I can not use cleanraw data to automatically remove electrode like the default option in EEGLAB ?

The authors decide to remove this option because they do not trust the bad channel detection performed by cleanraw data plugging. Instead, they use it only to help to take the decision about which electrode must be interpolated when there is a doubt.

Should I use ASR or not?

The ASR correction is difficult to understand (like a black box). It uses a sliding window (default 0.5 s, overlap 50%) to PCA-decompose all channels to identify a 'bad' PC (defined by a comparison with the data's own cleanest part in frequency-enhanced RMS) to reject and reconstruct the rejected PC activity from the remaining components.

After much experience and testing, the authors are confident of the result. It can save a bad recording (which is important in the case of a clinical trial). This algo is very popular in the literature. However, due to its "magic", if you only want to correct the blink artefact, we will go more to an ICA decomposition, which offers you more control.