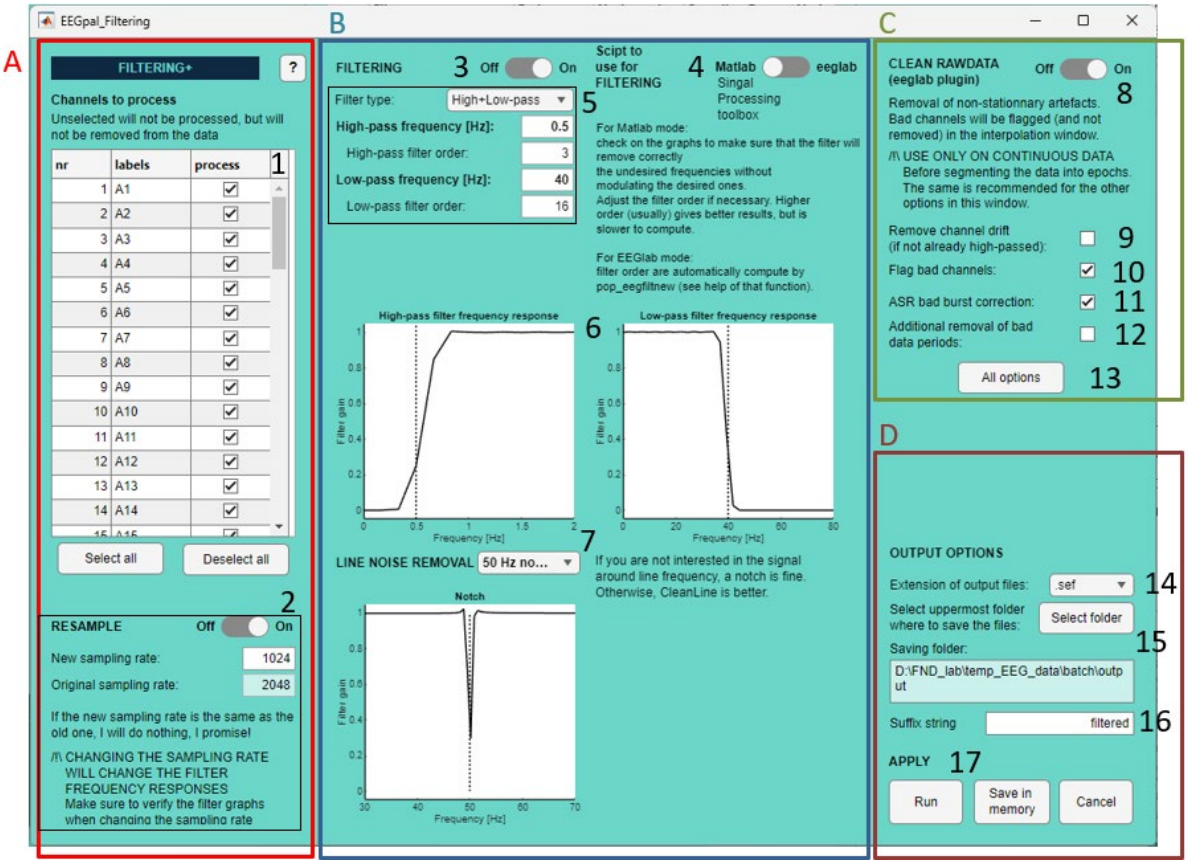
EEGpal: **Filtering+ module**

Version 2.0, 28.08.2025

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 select an electrode to be excluded from processing in the Filtering + module. The electrode will not be removed from the output file, which retains the original signal.
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 included in the Signal Processing toolbox (addone to Matlab)

   b. eeglab mode: It will use the *pop_eegfiltnew* function included in the eeglab toolbox (included in EEGpal)
5. The parameters of the filter depend on the mode:
   a. In Matlab mode, the user can choose between **High+Low pass** which corresponds to a passband filter (successively applying a high pass and then a low pass), **Low pass** or **High Pass**. The order of the filters can also be specified. 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 filter order on the graph at step **6**. Warning: if you choose a filter order that is too high, there will be more problems with phase shift (the temporal distortion that we do not want in the EEG) or Gibbs ringing artefacts (which create unwanted oscillations in the EEG signal).

   b. In eeglab mode, the high+low pass is replaced by **Band-pass** (more detail in FAQ). You can still perform a **Low pass** or **High Pass** independently. In this mode, the filter order is determined automatically (see FAQ for more details). This order depends on the cut-off frequency and the sampling rate. If these two values are identical throughout your dataset, the order will be the same for all participants. This value is displayed in the Matlab console window during processing.

6. View the cut-off behaviour according to the filter's threshold and order (this feature is 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, which is the default option in Matlab mode and requires the Signal Processing toolbox. Alternatively, you can use the Cleanline module of eeglab, which removes all harmonic frequencies but takes longer to process. For more details, please visit the following webpage: https://eeglab.org/plugins/cleanline/.
   The notch filter is disabled in eeglab mode because it requires the Signal Processing Toolbox Matlab add-on. It is better to use Cleanline in this case.

Pannel C

8. You can activate the pluging **clean rawdata** from eeglab. Steps 9–13 will detail the options offered by this plugin. For more details, please refer to the following webpage: https://eeglab.org/plugins/clean_rawdata/
9. **Remove channels drift:** This correction is unnecessary if you have applied a low-pass filter at step 5, as this achieves the same result. Therefore, in most cases, this correction should not be used. Also, it requires the Signal Processing Toolbox of Matlab.
10. **Flag Bad channels**: It will automatically detect bad channels without removing them. Instead, it will mark them to help you during the Interpolation step. You can view the results in the Interpolation module via the CleanRawData slider. This allows you to decide whether to accept the suggestion or not.
    Note: The authors are not convinced by this detection and do not agree with its suggestions (see FAQ for more details).
11. **ASR bad burst detection**: This applies the highly popular Artifact Subspace Reconstruction (ASR) correction method. It cleans the signal of artefact bursts such as eye blinks and

participant movement. It is a non-stationary method (i.e. it uses sliding window PCA). It is important to note that it corrects the signal without removing any time points.

12. **Additional removal bad data periods**: This performs the final window rejection. If a sliding window (default: 1.0 s with 66% overlap) identifies more than the specified percentage of poor channels, even after ASR, the window is rejected. Disabling this function keeps the data length the same before and after processing, if necessary.
    However, the authors advise to not use this option, as a similar correction would be performed by the Epoching module.
13. You can access the parameters for the various corrections (9-12). You can find a help file in this options box for more information about how these parameters work.
    However, unless you know what you're doing, we advise leaving the default value.

Pannel D

14. Select the format of the output files.
15. Select the destination folder where the results files will be saved. Note that this reproduces the input structure. For example, if the input files were in subfolders, there would be a folder per participant.
16. The suffix that is added to the input filename in order to obtain the output filename.
17. There are three validation buttons:
    a. The **Run** button will carry out the processing parameterized in the Filtering module.
    b. The **Save in memory** button will store all the parameters in memory and close the Filtering module without performing the processing.
    c. The button **Cancel** closes the module without processing and without keeping the entered parameters in memory. The same effect will be achieved by closing the Filtering+ module window.

FAQ

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

This is a difficult question. The results are similar for these two options. The authors found that the quality of the Matlab 'filtfilt' filter was slightly better. However, eeglab filters can be used without a Signal Processing Toolbox license and are cited more frequently in scientific literature.

Why can I choose the filter order in Matlab mode but not in EEGLab mode?

The two filtering functions use filter orders with different scales. To avoid confusion, the authors decided to set these values automatically when using eeglab (the default option).

Why has the high-low pass been replaced by a bandpass option in EEGLAB mode?

The authors obtained an unusual result when applying a high-pass filter followed by a low-pass filter using the *pop_eegfiltnew* function. This is why they decided to propose a bandpass filter, similar to the original filtering GUI in eeglab.

Why has the 'Passband' option been replaced by 'High-pass + Low-pass' in Matlab mode?

The result is improved by successively applying a high-pass filter, followed by a low-pass filter, because this offers flexibility in the order of the filters.

Why doesn't the toolbox offer the option to remove the baseline shift using DC, which is a common feature of other EEG processing software?

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

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

This is a difficult question. Some people don't like using both for fear of losing too many signals of interest. However, the creator of the ASR correction claims that the two types of correction complement each other: ASR is good at removing occasional large-amplitude noise/artefacts, while ICA is good at decomposing constant fixed-source noise/artefacts/signals. Therefore, there is no consensus, and both views are accepted in the literature.

Why can't I use the clean raw data to automatically remove the electrodes, as is the case with the default eeglab option?

The authors decided to remove this option because they did not trust the poor channel detection performed by CleanRawDataPlugging. Instead, they only use it to help decide which electrode to interpolate when in doubt. However, in batch mode, the have the possibility to automatically reject the suggested bad channel by choosing the option *Automatic Bad Electrodes Rejection* for interpolation (see the manual of the batch system for more information).

Should I use ASR or not?

The ASR correction is difficult to understand (it's like a black box). It uses a sliding window (default: 0.5 seconds, with a 50% overlap) to perform principal component analysis (PCA) on all channels, in

order to identify a 'bad' PC. This is defined by comparing the data with its cleanest part in frequency-enhanced root mean square (RMS). The 'bad' PC is then rejected, and its activity is reconstructed from the remaining components.

After extensive experience and testing, the authors are confident in the results. This can save poor-quality recording, which is important in the case of clinical study. This algorithm is very popular in literature. However, due to its 'magic', if you only want to correct the blink artefact, we recommend an ICA decomposition, which offers more control.