Chapter 19
EEG/ERP Data Analysis Toolboxes

Gan Huang

Abstract The development and application of new methods allow us to perform more advanced analyses on EEG signals. However, the understanding of the mathematical and methodological details about these new methods would be no easy for the researchers without engineering and mathematics background. As a collection of tools for EEG signal processing and data visualization, EEG/ERP analysis toolboxes make the researchers be able to perform the complex analysis by simply clicking buttons or running some lines of MATLAB script. In this chapter, we firstly make a brief overview about the currently popular toolboxes in EEG/ERP analysis, such as EEGLab, FieldTrip, BrainVision Analyzer, etc., and then focus on the introduction of Letswave, which is an intuitive and streamlined tool to process and visualize EEG data, with a shallow learning curve. Examples are provided for a better understanding of Letswave7 in EEG/ERP data analysis.

Keywords Toolbox · Letswave · Signal processing · Data visualization

19.1 Brief Overview

In the recent decades, a variety of new methods have been developed and applied in the area of EEG analysis. By integrating these methods, the development of EEG/ERP toolboxes provides the researcher with powerful and convenient tools for EEG signal visualization and processing. In the following, we make a brief overview of these most frequently used toolboxes in EEG/ERP analysis.

EEGLAB (Delorme and Makeig 2004): Currently the most widely used open-source toolbox for EEG/ERP analysis, which provides the basic operations from data importing, segmentation, ICA, bandpass filtering, baseline correction to time-frequency analysis, etc. EEGLAB has been developed by Arnaud Delorme and

G. Huang (✉)
School of Biomedical Engineering, Health Science Center, Shenzhen University, Shenzhen, China

Scott Makeig at the Salk Institute (La Jolla CA, USA) since 1997. Based on different levels of programming sophistication, EEGLAB provides both menu-based graphic user interface (GUI) and script-based command-line interface (CLI) on the platform of MATLAB. Rich plugins make it easier for users to complete more diverse tasks with EEGLAB.

FieldTrip (Oostenveld et al. 2011): Another famous MATLAB-based open-source EEG analysis toolbox. The development team is from the Donders Institute for Brain, Cognition and Behaviour (Radboud University, the Netherlands), led by Robert Oostenveld and Jan-Mathijs Schoffelen. FieldTrip provides a set of functions with some advanced operations for EEG analysis, such as source analysis and cluster-based permutation test. However, no graphic interface makes it difficult for beginners to use, especially those without any programming foundation.

Letswave: As another comprehensive EEG/ERP analysis toolbox, Letswave was developed by the team of André Mouraux (Institute of Neuroscience, Université catholique de Louvain, Belgium). Compared to other EEG signal processing toolboxes, Letswave emphasizes on intuitive and streamlined tools to process and visualize EEG data, with a shallow learning curve. More detailed introduction about Letswave is arranged in the following section.

Some other toolboxes may be not developed just for EEG signal processing but also provide excellent tools on EEG/ERP analysis. For example, BrainStorm (Tadel et al. 2011) shares a comprehensive set of user-friendly tools with the scientific community using MEG/EEG as an experimental technique. Statistical parametric mapping (SPM) (Litvak et al. 2011), designed for the analysis of brain imaging data sequences, is compatible for the analysis of fMRI, PET, SPECT, EEG, and MEG. Different from most of the open-source toolboxes, MNE-Python (Gramfort et al. 2013) provides a collection of functions for MEG/EEG processing and visualization on the Python platform.

In addition to these free academic toolboxes, there are some excellent EEG analysis commercial programs that are often used with some specialized amplifiers, such as BrainVision Analyzer for Brain Product and Scan for Neuroscan. In most cases, the commercial tools for EEG analysis are developed with programming languages, which should be called program since they normally exactly generate an executable program. While the free tools should be called as toolboxes, because it is a set of scripts on certain platform, not a program in the strict sense. Compared with free toolboxes, commercial program normally provides a more efficient and friendlier user interface, but the higher prices limit their use in academia community. To the best of the authors' knowledge, we have seen any research group packaged their newly developed methods into a plugin for these programs for the others to use.

Some free toolboxes and commercial programs may not provide full functionality for EEG analysis but only provide analysis functions in some specialized areas, such as SIFT (Delorme et al. 2011), MVGC (Barnett and Seth 2014), Hermes (Niso et al. 2013), TRENTOOL (Lindner et al. 2011), EEGNET (Lawhern et al. 2016), and Chronux (Bokil et al. 2010) for connectivity estimation and LORETA, BESA, and Curry for source analysis.

The web links of all toolboxes and programs are listed as follows:

- EEGLAB: https://sccn.ucsd.edu/eeglab/
- FieldTrip: http://www.fieldtriptoolbox.org/
- Letswave: https://letswave.cn/
- BrainStorm: https://neuroimage.usc.edu/brainstorm
- SPM: https://www.fil.ion.ucl.ac.uk/spm/
- MNE-Python: https://mne-tools.github.io/
- Analyzer: https://www.brainproducts.com/promo_analyzer2.php
- Scan: https://compumedicsneuroscan.com/tag/scan/
- SIFT: https://sccn.ucsd.edu/wiki/SIFT
- MVGC: http://www.sussex.ac.uk/sackler/mvgc/
- Hermes: http://hermes.ctb.upm.es/
- TRENTOOL: http://www.trentool.de/
- EEGNET: https://sites.google.com/site/eegnetworks
- Chronux: http://chronux.org/
- LORETA: http://www.uzh.ch/keyinst/loretaOldy.htm
- BESA: http://www.besa.de/
- Curry: https://compumedicsneuroscan.com/curry-8-released/

19.2 The Introduction of Letswave

19.2.1 General Overview

Letswave is a free, open-source MATLAB toolbox to analyze EEG and other neurophysiological signals, covered under the terms of the GNU General Public License. The project is managed by André Mouraux (Institute of Neuroscience, Université catholique de Louvain, Belgium), in collaboration with Gan Huang (Shenzhen University, China), Bruno Rossion (Institute of Neuroscience, Université catholique de Louvain), Li Hu (Institute of Psychology, Chinese Academy of Sciences, China), and Giandomenico Iannetti (University College London, UK). The development of Letswave started in 2006. The first four versions are developed in Borland Delphi. It is switched to MATLAB platform since Letswave5. Up to today, the newest version of the Letswave is Letswave7, and it continues to be developed.

Letswave can be used on Windows, Mac, or Linux system with MATLAB 2010 or later version. Considering the major updating of graphics system in MATLAB 2014b, MATLAB with the version later than 2014b is highly recommended for running Letswave. The basic processing requirement is 4GB RAM. For larger dataset analysis, 8GB RAM is recommended, and 16GB or more of RAM would be helpful in time-frequency analysis.

Letswave includes the following four modules, which is shown in Fig. 19.1:

- **Manage module** is the main interface of Letswave. It can be launched by typing “letswave” in the command window of MATLAB. The role of the manage

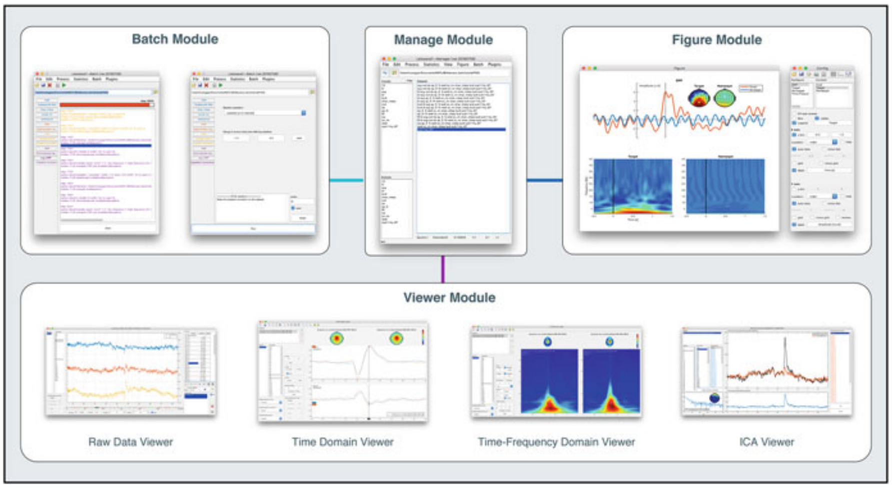


Fig. 19.1 The framework of Letswave. There are four modules in the GUI design, which are manage module, batch module, viewer module, and figure module

module is mainly data management, such as dataset import/export/delete/rename, epochs/channels/events management, and also multiple datasets selection for further operation. In the manage module, all the datasets are managed by the affixes in their filenames, which are separated by space. Multiple datasets can be easily and quickly selected from tens of thousands of datasets by the affixes filtering in the left side. After the selection, the interface to the other modules is provided in the menu of the manage module and right-clicking menu.

- **Batch module** contains multiple functions for EEG signal analysis, including data preprocessing, segmentation, time/frequency domain analysis, and statistical analysis. Batch module allows the GUI-based users to do the same processing on a single dataset or multiple datasets at the same time. It is also easier for the users to do multiple processing steps on the same dataset(s). This processing history can also be viewed in the batch module. Furthermore, users can define their own processing flow, which may involve multiple datasets and multiple processing steps. The frequently used processing flow can also be saved in the menu of the manage module. For another similar dataset(s), the users can call out the processing flow and just modify the input datasets to run the analysis. More interestingly, once the processing flow has been made, the script has also been generated automatically. Just by clicking the button of "script", users can get the corresponding script without writing any code.
- **Viewer module** is used for the observation of one or multiple dataset(s). Viewer module includes a set of viewers, which are used for the observation of the continuous data and the result in time domain and time-frequency domain, respectively. For example, there is a batch of time domain datasets with multiple channels and several epochs. The viewer allows the users to display the result in

their own way, that is, show the datasets in separated windows in different rows or columns for different datasets/channels/epochs or superimpose them in the same windows. Spatial topography information can be jointly displayed for a single time/frequency point or the mean value from certain interval. Simple statistic of the mean, maximum, and minimum value with their location on the selected interval would make the work of peak detection with their latency information become simple and intuitive.

- **Figure module** is used for figure generation, that is, to customize the figure to show the waveform, time-frequency response, and the topography which could be used in the publication or presentation or importing to the other graphics software for further editing. Different from the other existence toolboxes, the figure module is very open. The GUI-based user can make their graphics from certain templates or directly from a blank canvas. It allows user to flexibly customize the layout, free to the style of font, line, and map. Further, the graphics can also be saved as a template. The user can generate a new graphic in the same style with different data sources by using this template.

19.2.2 To GUI-Based Users

With the graphic interface, the users can easily to run the whole progress for EEG analysis, from data importing, preprocessing, time-frequency analysis to statistical analysis. Compared with the existence toolboxes, there are two major features in the design of Letswave7.

Firstly, the function of data management is strengthened in Letswave. With number of subjects increasing, the lack of data management and batch processing functions makes the existence toolbox be difficult to handle the large number of datasets by GUI. Take an experiment with a two-way ANOVA design for example. If there are 3 levels for each factor and 120 subjects were included in the experiment and 1 dataset for 1 condition from 1 subject, then there would be more than 1000 raw datasets involved in the study. Including intermediate files and the final results, the number of the datasets involved in this study would be three or four times larger than 1000. Even selecting the target datasets for the corresponding processing would be difficult for GUI-based user. Hence, the datasets management and data batch processing are necessary in the design of the EEG signal processing toolbox. Hence, in Letswave, the datasets are managed by affixes; two list boxes in the left side of the manage module make it easy to dataset selection.

Secondly, the function for data visualization is separated into two parts, which are data observation during data analysis in viewer module and figure generation for publication in figure module. Data observation does not require beauty of the image, but is highly demanding in terms of convenience and observability during the whole process of the data processing, from data import, artifact rejection in the preprocessing to the joint result observation in time-frequency-spatial domain.

19.2.3 To Script-Based Users

In addition to the GUI, script may be the most commonly used feature in Letswave. The following unique designs make the script writing super easy in Letswave:

19.3 Full Transparency Between Script and GUI

In batch module, Letswave runs a processing flow by running the corresponding MATLAB script. Once the processing flow has been made, the corresponding MATLAB script has already been generated automatically. Users can click the "script" button in the batch module to get the full script. Hence, the users, who have learned the use of GUI, almost already understand the writing of script.

The script for the single-step operation can also be gotten with the "script" button in each step, which can tangibly reduce the users' dependence on the tutorial and the help document. This feature is essential, since it can effectively help the script-based user to speed up the script writing and improve the script quality. More importantly, it can really reduce the script-based user's entry threshold. Even the users without a solid background for MATLAB programming are able to write high-quality script for batch EEG signal processing.

Not only batch module, the operations in other modules can also become scripts automatically. Hence, from data import, to preprocessing, time/frequency analysis, and statistical analysis, to the final figure generation, the whole process can be easily and quickly written as a script for the EEG signal analysis.

19.4 Easy Access to the Data

Each dataset can be easily accessed by clicking "send to workspace" in the right menu in the manage module, which would be appeared as a variable with the name of "lwdata" in the workspace of MATLAB. Then the script-based user can make their own operation on the dataset in MATLAB. After the operation, the user can also save the dataset in the format of Letswave by clicking "read from workspace" in the right menu in the manage module.

Similarly, the other information is also easily to be accessed in the other modules, such as the event information, the channel name, and statistical result about the mean, maximum, and minimum value.

19.5 The Simplest Grammar

In Letswave, we provide a rich library of functions for the script-based users, which are starting with `FLW_`. The grammar of all the functions in this library is uniform. For one step of operation, we just need to define the option and then call the corresponding FLW function.

```
option=struct('XXX',xxx,'XXX',xxx);
lwdata= FLW_XXX.get_lwdata(lwdata,option);
```

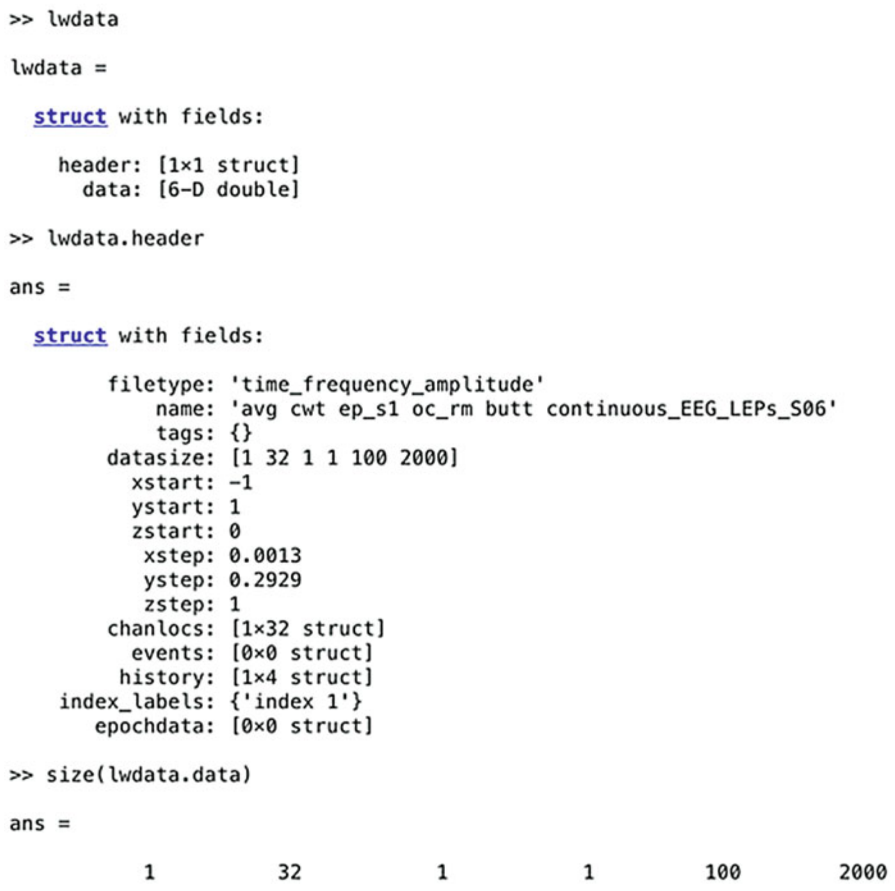
The parameters setting in the option is identical to the parameter's selection in the GUI. It just needs to be noticed whether the input and output of the function is a single dataset or multiple datasets.

19.5.1 To Developers

Developers can also make their own methods integrated into Letswave by developing the FLW file. For these advanced users, the requirements on the programming will be much higher.

Before the developing, the understanding of the data structure of Letswave is important (Fig. 19.2). In Letswave, each dataset is saved in the hard driver by two files, which are "lw6" and "mat" files. "Lw6" file is a basic description about the dataset, including some simple information such as filename, data size, the starting point, and step size for each dimension. Some information, like the channel, event, and history, are kept in the "lw6" file as structure variables. "Mat" file is using a six-dimension matrix to save the data with respect to epoch, channel, and index, z, y, x correspondingly. In MATLAB, the dataset "lwdata" is composed of header and data, which are form lw6 and mat file, respectively.

The developing of FLW function mainly includes four parts of work. Firstly, design the UI which would be integrated into batch module. Secondly, define the parameter setting for Letswave generating the script automatically. Thirdly, implement their methods. Finally, put the FLW file into the corresponding place for Letswave to call it. A guide about how to write a FLW file is also included in the toolbox.



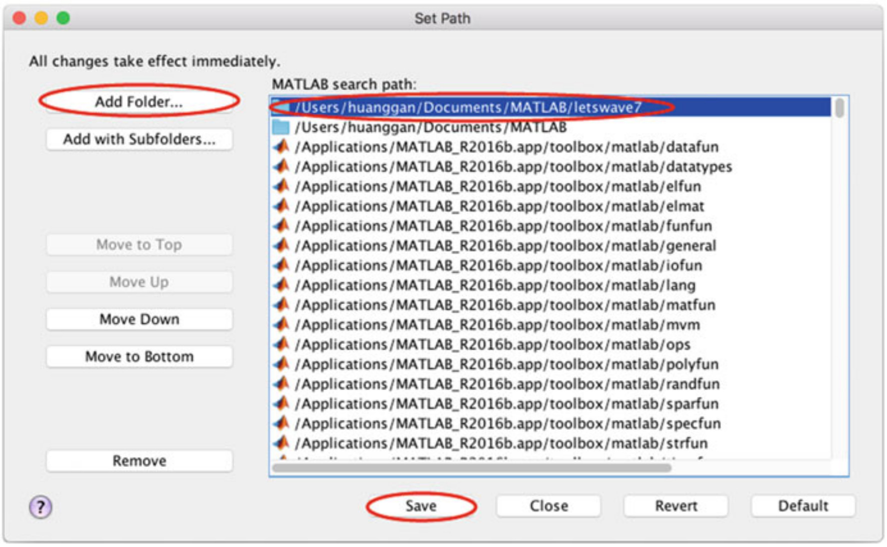
```
>> lwdata

lwdata =

  struct with fields:

    header: [1×1 struct]
      data: [6-D double]

>> lwdata.header

ans =

  struct with fields:

        filetype: 'time_frequency_amplitude'
            name: 'avg cwt ep_s1 oc_rm butt continuous_EEG_LEPs_S06'
            tags: {}
        datasize: [1 32 1 1 100 2000]
          xstart: -1
          ystart: 1
          zstart: 0
           xstep: 0.0013
           ystep: 0.2929
           zstep: 1
        chanlocs: [1×32 struct]
          events: [0×0 struct]
         history: [1×4 struct]
    index_labels: {'index 1'}
       epochdata: [0×0 struct]

>> size(lwdata.data)

ans =

           1          32           1           1         100        2000
```

Fig. 19.2 The data structure for a dataset in Letswave

19.6 Download and Setup

19.6.1 Download

The latest version can be downloaded as an archive from the NOCIONS Github repository (https://github.com/NOCIONS/letswave7/archive/master.zip). The users can also download the zip file by visiting the NOCIONS Github repository (https://github.com/NOCIONS/letswave7) and manually download it.



All changes take effect immediately.

MATLAB search path:

```
/Users/huanggan/Documents/MATLAB/letswave7
/Users/huanggan/Documents/MATLAB
/Applications/MATLAB_R2016b.app/toolbox/matlab/datafun
/Applications/MATLAB_R2016b.app/toolbox/matlab/datatypes
/Applications/MATLAB_R2016b.app/toolbox/matlab/elfun
/Applications/MATLAB_R2016b.app/toolbox/matlab/elmat
/Applications/MATLAB_R2016b.app/toolbox/matlab/funfun
/Applications/MATLAB_R2016b.app/toolbox/matlab/general
/Applications/MATLAB_R2016b.app/toolbox/matlab/iofun
/Applications/MATLAB_R2016b.app/toolbox/matlab/lang
/Applications/MATLAB_R2016b.app/toolbox/matlab/matfun
/Applications/MATLAB_R2016b.app/toolbox/matlab/mvm
/Applications/MATLAB_R2016b.app/toolbox/matlab/ops
/Applications/MATLAB_R2016b.app/toolbox/matlab/polyfun
/Applications/MATLAB_R2016b.app/toolbox/matlab/randfun
/Applications/MATLAB_R2016b.app/toolbox/matlab/sparfun
/Applications/MATLAB_R2016b.app/toolbox/matlab/specfun
/Applications/MATLAB_R2016b.app/toolbox/matlab/strfun
```

Fig. 19.3 In the Set Path window, click the "Add Folder" button

19.6.2 Installation

- Unzip the zip archive of Letswave7 in the folder of your choice.
- Launch MATLAB.
- In the MATLAB menu ribbon, click on "Set Path".
- In the Set Path window, click button "Add Folder" (Fig. 19.3).
- Select the location of the Letswave7 folder.
- Save the modified path and close.

To check whether the installation is successful or not, input "letswave7" in the command windows of MATLAB. The pop-up of manage module, which is the main interface in Letswave7, indicates that the installation is successful.

19.7 Example for Single-Subject Analysis

In this part, a dataset with P300 experiment will be used to illustrate the data processing for single subject. After the preprocessing with frequency filtering, bad electrodes interpolation, and ICA decomposing, both time domain analysis and time-frequency domain analysis will be performed on the single-subject level.

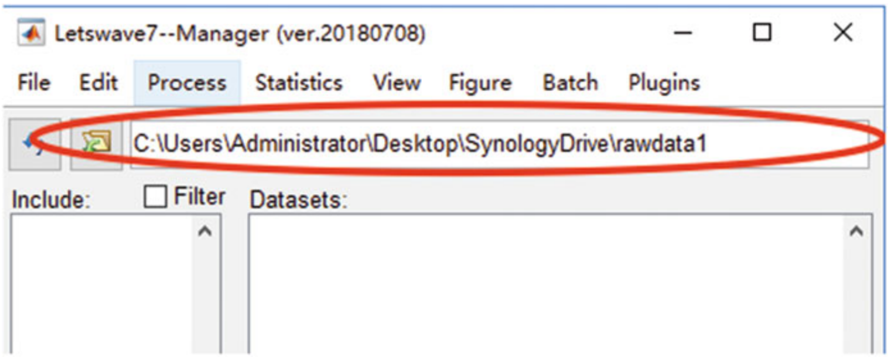
19.7.1 Dataset Importing

The dataset is a P300 experiment for one subject, recorded by BrainAmp (Brain Products GmbH, Germany) with 64 channels, 1000Hz sampling rate, and referenced to FCz. Visual oddball experiment with the red as the target stimuli (marked as “S 9”) and white as the nontarget stimuli (marked as “S 10”) on the screen. Each square lasts 80ms with the ISI 200ms. Six hundred trials of the stimuli in all are arranged in a 2-min session, in which target stimuli come with the possibility of 5%. The participant is asked to count the number of red square and report after the session to make the participant keep attention on the screen. Download the raw dataset (https://github.com/NOCIONS/letswave7_tutorial/raw/master/rawdata1.zip), and unzip the rawdata1.zip file. There are three files, which are sub093.eeg, sub093.vhdr, and sub093.vmrk.

Open MATLAB, and input “letswave7” in the command windows of MATLAB to open Letswave7. Set the path of Letswave to the folder of the dataset, like “C:\Users\Adminstrator\Desktop\SynologyDrive\rawdata1” here (Fig. 19.4).

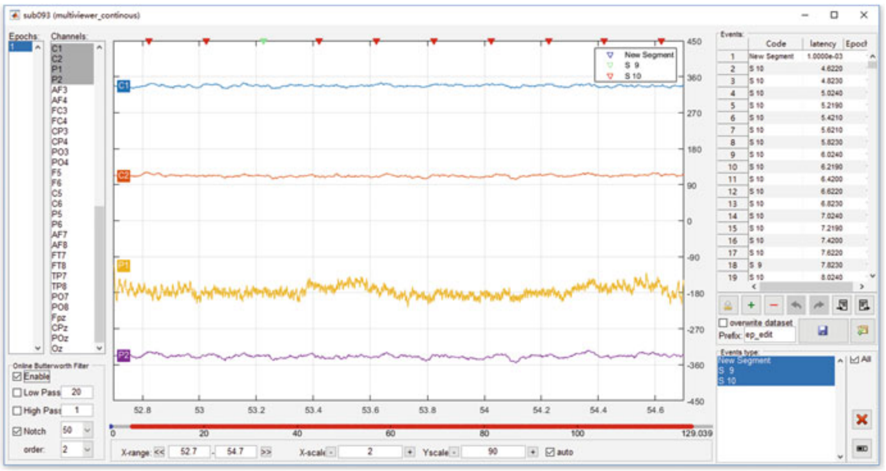
Select **File->Import->Import EEG/MEG data files** in the menu of the manage module, and then the dialog of import data will pop up. Press the button **add files** to add the file *sub093.eeg*. Press the button **import files** to import the dataset. During the importing, it will display “processing.” Once the importing is finished, the corresponding dataset will turn red and display “done.” Close the dialog of import data; the dataset *sub093* will be appeared in the manage module.

Select dataset *sub093*, and click **View->Continuous Data Viewer** in the menu to check the data quality of the imported dataset. In the viewer of the continuous data, it can be seen the channel P1 is obviously abnormal (Fig. 19.5). A strong 50Hz power frequency interference covers the signal. Even after a notch filter, there is still a strong noise. Hence in the following processing, channel **P1** is treated as a bad electrode, which would be interpolated by the surround channels.



C:\Users\Administrator\Desktop\SynologyDrive\rawdata1

Fig. 19.4 Set the path of Letswave to the folder of the dataset



Online Butterworth Filter
☑ Enable
☐ Low Pass 20
☐ High Pass 1
☑ Notch 50
order: 2

X-range: 52.7 - 54.7 X-scale 2 Yscale 90 ☑ auto

☐ overwrite dataset
Prefix: ep_edit

	Code	latency	Epoch
1	New Segment	1.0000e-03	
2	S 10	4.6220	
3	S 10	4.8230	
4	S 10	5.0240	
5	S 10	5.2190	
6	S 10	5.4210	
7	S 10	5.6210	
8	S 10	5.8230	
9	S 10	6.0240	
10	S 10	6.2190	
11	S 10	6.4200	
12	S 10	6.6220	
13	S 10	6.8230	
14	S 10	7.0240	
15	S 10	7.2190	
16	S 10	7.4200	
17	S 10	7.6220	
18	S 9	7.8230	
19	S 10	8.0240	

Fig. 19.5 Check the quality of the imported dataset

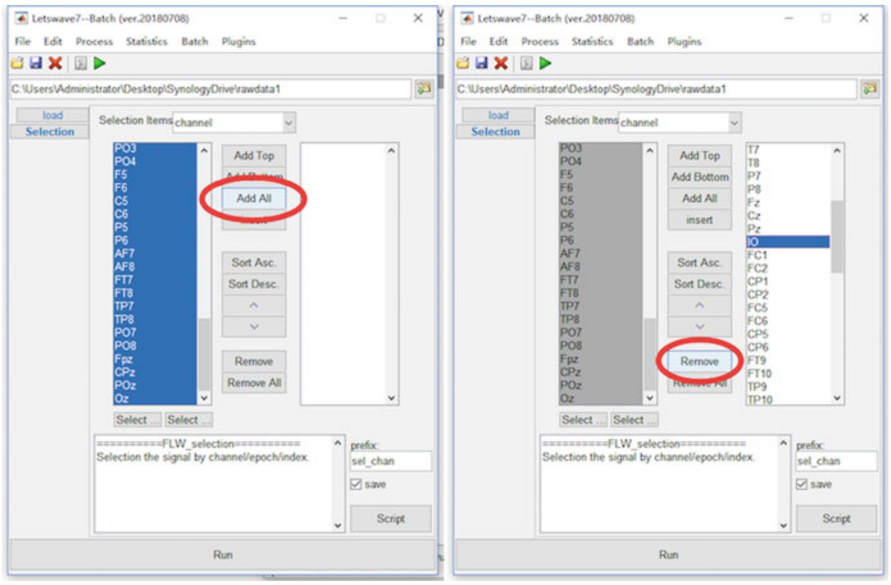
19.7.2 Useless Channels Removing

Before removing useless channels, usually it is necessary to assign the coordinate information for each channel in the dataset firstly. But Letswave7 can automatically assign the channel location with commonly used 10–20 system. If the other EEG coordinate system is used, the users can still manually assign the correct channel location by clicking **Edit->Electrodes->Edit electrode coordinates** in the menu.

To make an efficient analysis and save the storage space, removing the useless channels is necessary. In this example, channel **IO** will be removed, which records the electrooculogram (EOG) signals. To remove channel IO, select the dataset *sub093* in the manage module, and click **Edit->Arrange signals->Rearrange or delete epochs, channels, indexes** in the menu. In the batch module (Fig. 19.6), press the button **Add all** to add all the channels into the left list box. Then select channel IO, and press button **Remove** to delete channel **IO.** Click the button **Run** in the bottom of batch module, and then a new dataset with the name *sel_chan sub093* will be appeared in the data list of the manage module.

19.7.3 Frequency Filtering

Frequency filtering is an effective way to filter out the high-frequency artifact, low-frequency drift, and the 50/60Hz power-line interference. In this case, the bandpass filter is set as 0.05–30Hz. Select the dataset *sel_chan sub093* in the manage module, and click **Process->Frequency analysis and filters->Butterworth filters** in the menu. In the batch module, set the low cutoff frequency (Hz) as **0.05Hz**, and


Letswave7--Batch (ver.20180708)
File Edit Process Statistics Batch Plugins
C:\Users\Administrator\Desktop\SynologyDrive\rawdata1
load
Selection
Selection Items channel
PO3 PO4 F5 F6 C5 C6 P5 P6 AF7 AF8 FT7 FT8 TP7 TP8 PO7 PO8 Fpz CPz POz Oz
Add Top
Add Bottom
Add All
insert
Sort Asc.
Sort Desc.
Remove
Remove All
T7 T8 P7 P8 Fz Cz Pz IO FC1 FC2 CP1 CP2 FC5 FC6 CP5 CP6 FT9 FT10 TP9 TP10
Select ...
Select ...
prefix:
sel_chan
save
Script
Run

==========FLW_selection==========
Selection the signal by channel/epoch/index.

Fig. 19.6 Useless channels removing

click the button **Run** to run the bandpass filtering. A new dataset with the name *butt sel_chan sub093* will be appeared in the data list of the manage module.

19.7.4 Bad Electrodes Interpolation

In this step, channel P1 would be interpolated with the surround channels. Select the dataset *butt sel_chan sub093* in the manage module, and click **Edit->Electrodes->Interpolate channel using neighboring electrodes** in the menu. In the batch module, select channel **P1** in the **Channel to Interpolate** list box, and then click the button **Find closest electrodes**. Since the **number of channels used for interpolation** is **3** for the default setting, Letswave7 will find the closest electrodes **P3**, **Pz**, and **CP1** for interpolation automatically according to the location of the channels. Click the button **Run** for the bad electrodes interpolation. A new dataset with the name *chan_interp butt sel_chan sub093* will be appeared in the data list of the manage module.

19.7.5 ICA Decomposing

Independent component analysis (ICA) is a kind of blind signal separation (BSS) method. Linearly, ICA can be modeled as in Fig. 19.7. Consider **X** is the recorded

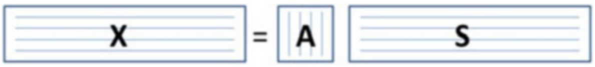


Fig. 19.7 Linear model for ICA

EEG signal with the dimension *channel* × *time*, **S** is the source signal with the dimension *component* × *time*, and **A** is the mixing matrix with the dimension *channel* × *component*. The aim of ICA is to find the mixing matrix **A** to make each component (each row) to be independent of each other. According to the linear model, ICA can be used for artifact removing in EEG signal processing by the following steps:

1. Run ICA algorithm to get the mixing matrix **A**.
2. Automatically, we can have the source signal **S** = **pinv(A)** × **X**.
3. Identify the artifact component manually in **S**. By setting the corresponding row as **0**, we have **S_bar**.
4. By **S_bar** with the artifact removed, we can automatically get **X_bar** = **A** × **S_bar**.

The signal **X_bar** is the result by ICA artifact removal. In Letswave7, steps 2 and 4 can be completed automatically, in which **pinv(A)**, also called unmixing matrix, is the pseudo-inverse matrix for matrix **A**. Step 1 (computer ICA matrix) and step 3 (identify artifact component) need to be performed manually. Hence, in Letswave7, we need two steps to finish the work of artifact removal by ICA.

Select the dataset *chan_interp butt sel_chan sub093* in the manage module, and click **Process->Spatial filters (ICA/PCA)->Compute ICA matrix** in the menu. In the batch module, select number of components as **decide by user**, and set the components numbers as **40**. Click the button **Run** for compute ICA matrix. A new dataset with the name *ica chan_interp butt sel_chan sub093* will be appeared in the data list of the manage module.

After computing the ICA matrix, we need to manually identify the components with artifact. Select the dataset *ica chan_interp butt sel_chan sub093* in the manage module, and click **Process->Spatial filters(ICA/PCA)->Apply ICA/PCA spatial filter** in the menu. The interface is popped up for manually removing component for spatial filter (Fig. 19.8). Different types of information is marked with different colors. The black color represents the original signal **X**, color blue for source **S**, and color orange for the filtered signal **X_bar**. In the left panel (black), we can select the dataset, epoch, and channel to check the original signal **X** as the black curve in the middle panel. Next, we can select the component in the left panel (blue) to check the time, frequency, and spatial feature of each component of source **S** in the bottom panel (blue). After identifying the component as the artifact in the right panel in orange, the orange curve in the middle panel would show the corresponding filtered signal **X_bar**. We can check the result of ICA filtering immediately. In this study, **component 1** is identified as the eye blink artifact. The scalp topography suggests the "equivalent current dipole" (ECD)s close to the eyes. The waveform in the time domain looks like a spike. The power in the frequency domain concentrates at low-frequency band (<5Hz). All these suggest **comp 1** as the eye blink artifact. After removing this component by selecting **comp 1** in the right panel, the blink

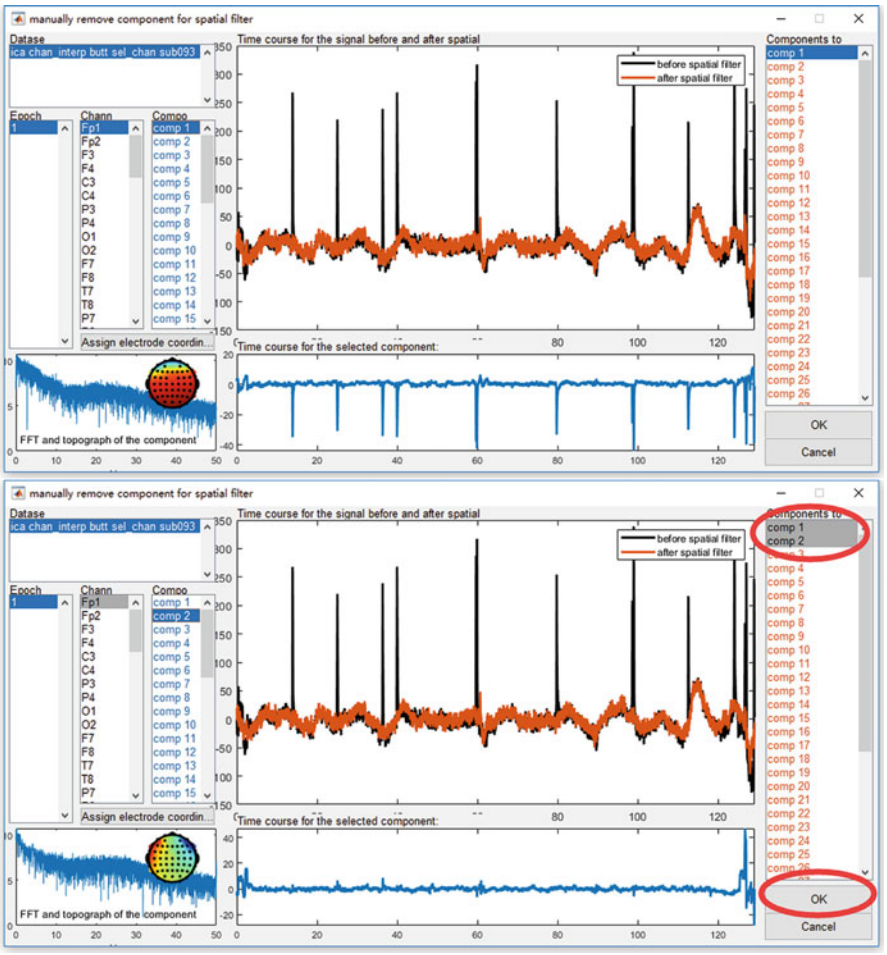

manually remove component for spatial filter
Datase
ica chan_interp butt sel_chan sub093
Epoch
Chann
Compo
Assign electrode coordin...
FFT and topograph of the component
Time course for the signal before and after spatial
before spatial filter
after spatial filter
Time course for the selected component:
Components to
OK
Cancel

Fig. 19.8 Artifact component identification

artifact has been effectively removed as compared the orange curve **X_bar** with the black curve **X** in the middle panel. Similarly, **component 2** as the artifact with lateral eye movement. The clear evidence can be observed from the scalp topography. Select **comp 1** and **comp 2** in the right panel, and click the button **OK**. A new dataset with the name *sp_filter ica chan_interp butt sel_chan sub093* will be appeared in the data list of the manage module, which is a result of artifact removing by ICA.

The order for ICA and segmentation should also be discussed in preprocessing. Since ICA is a data-driven method for artifact removal, we need enough data to run ICA. While, excessive amount of data can greatly increase computing time, but the improvement of the accuracy in the result is limited. Normally, we run the ICA after segmentation, since segmentation could shorten the data length and remove irrelevant noises. However, in this case of the P300 study, the overlap between trials is very serious; running ICA after segmentation would unnecessarily increase calculation time. Hence, ICA is applied before segmentation.

In this case, we run the ICA by selecting the number of the components as **40**. Normally, the maximum number of independent components that can be separated is equal to the number of channels in the original signal **X**. If **X** has been referenced, the maximum number should minus 1. For each channel has been interpolated, the maximum number should minus 1 again. In addition, decreasing the number of independent components to separate is a way to reduce the computing time for the ICA matrix. In this case, since the channel number 64 is large enough, we set the number of independent components as 40 is OK.

19.7.6 Segmentation

Select the dataset *sp_filter ica chan_interp butt sel_chan sub093* in the manage module, and click **Process->Epoch segmentation->Segment relative to events (one file per event code)** in the menu. In the batch module, select event codes **S 9** and **S 10**, and set the epoch starting time and duration as **-1** and **3**. Click the button **Run** in the bottom of batch module, and then two new datasets with the name *ep_S 9 sp_filter ica chan_interp butt sel_chan sub093* and *ep_S 10 sp_filter ica chan_interp butt sel_chan sub093* will be appeared in the manage module.

In the menu, there are two items for segmentation, which are **Segment relative to events** and **Segment relative to events (one file per event code)**. They have similar function, but the output would be different. If multiple event codes have been selected, **Segment relative to events** will segment all the epochs with different event codes into one dataset. However, with **Segment relative to events (one file per event code)**, separated dataset will be generated according to different event codes. For example, in this case of P300 dataset, the target and nontarget events are marked as **S 9** and **S 10**. Hence with **Segment relative to events (one file per event code)**, two datasets have been generated.

Since no obviously artifact has been observed for both the two datasets, it is not necessary to perform artifact rejection on these datasets. Hence, this step is omitted in the preprocessing of this P300 dataset. If necessary, the operation of artifact rejection can be done by clicking **Edit->Arrange signals->Rearrange or delete epochs, channels, indexes** in the menu.

19.7.7 Rereference

In the study of P300, the average of bilateral mastoids regions normally is selected as reference. Hence, we will rereference to the mean value of TP9 and TP10. Select the datasets *ep_S 9 sp_filter ica chan_interp butt sel_chan sub093* and *ep_S 10 sp_filter ica chan_interp butt sel_chan sub093*, and click **Process->Rereference signals-**

>Rereference in the menu. In the batch module, select the **TP9** and **TP10** as the **new reference** in the left list box, and select all channels in the right list box for **apply reference to**. Click the button **Run** in the bottom of batch module to finish the work of artifact rejection. Two new datasets with the name *reref ep_S 9 sp_filter ica chan_interp butt sel_chan sub093* and *reref ep_S 10 sp_filter ica chan_interp butt sel_chan sub093* will be appeared in the manage module.

Here, TP9 and TP10 are selected for using the mean value of the two channels as the new reference. For the common average reference, we can select all channels in the left list box, and then the average of all channels will be used as the new reference.

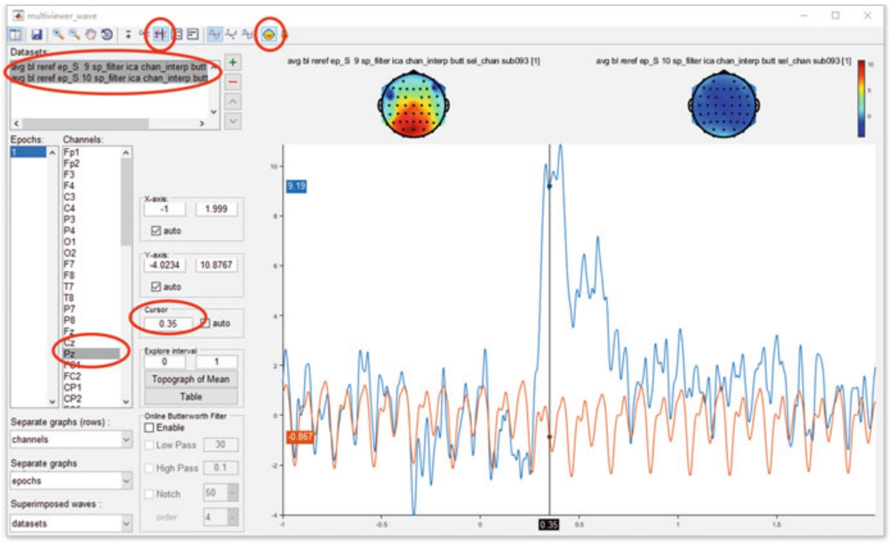
19.7.8 Baseline Correction

In the previous processing, the epochs are segmented from **-1s** to **2s**. Hence, the baseline is set from **-1s** to **0s** for baseline correction. Select the datasets *reref ep_S 9 sp_filter ica chan_interp butt sel_chan sub093* and *reref ep_S 10 sp_filter ica chan_interp butt sel_chan sub093*, and click **Process->Baseline operation-> Baseline correction** in the menu. Keep the default setting in the batch module, and click the button **Run** in the bottom of batch module to finish the work of artifact rejection. Two new datasets with the name *bl reref ep_S 9 sp_filter ica chan_interp butt sel_chan sub093* and *bl reref ep_S 10 sp_filter ica chan_interp butt sel_chan sub093* will be appeared in the manage module.

19.7.9 Averaging

After the ten steps of preprocessing, we can simply average the epochs to do time domain analysis. Select the datasets *bl reref ep_S 9 sp_filter ica chan_interp butt sel_chan sub093* and *bl reref ep_S 10 sp_filter ica chan_interp butt sel_chan sub093*, and click **Process->Average->Compute averag, std, median across epochs** in the menu. Keep the default setting in the batch module, and click the button **Run** in the bottom of batch module for averaging. Two new datasets with the name *avg bl reref ep_S 9 sp_filter ica chan_interp butt sel_chan sub093* and *avg bl reref ep_S 10 sp_filter ica chan_interp butt sel_chan sub093* will be appeared in the data list of the manage module.

To observe the waveform of time domain analysis result in the multiviewer, select datasets *avg bl reref ep_S 9 sp_filter ica chan_interp butt sel_chan sub093* and *avg bl reref ep_S 10 sp_filter ica chan_interp butt sel_chan sub093*, and click the **view** in the right-click menu. Selecting both the two datasets, selecting channel **Pz**, opening the topography in the toolbar, and setting the cursor to **0.35**, the P300 result from single subject is shown as it is in Fig. 19.9.


multiviewer_wave
Datasets:
avg bl reref ep_S 9 sp_filter ica chan_interp butt
avg bl reref ep_S 10 sp_filter ica chan_interp butt
Epochs:
Channels:
Fp1
Fp2
F3
F4
C3
C4
P3
P4
O1
O2
F7
F8
T7
T8
P7
P8
Fz
Cz
Pz
FC1
FC2
CP1
CP2
avg bl reref ep_S 9 sp_filter ica chan_interp butt sel_chan sub093 [1]
avg bl reref ep_S 10 sp_filter ica chan_interp butt sel_chan sub093 [1]
9.19
-0.867
0.35
Topograph of Mean
Table

X-axis: -1 1.999 ☑ auto
Y-axis: -4.0234 10.8767 ☑ auto
Cursor: 0.35 ☐ auto
Explore interval: 0 1
Online Butterworth Filter: ☐ Enable
☐ Low Pass 30
☐ High Pass 0.1
☐ Notch 50
order 4
Separate graphs (rows): channels
Separate graphs: epochs
Superimposed waves: datasets

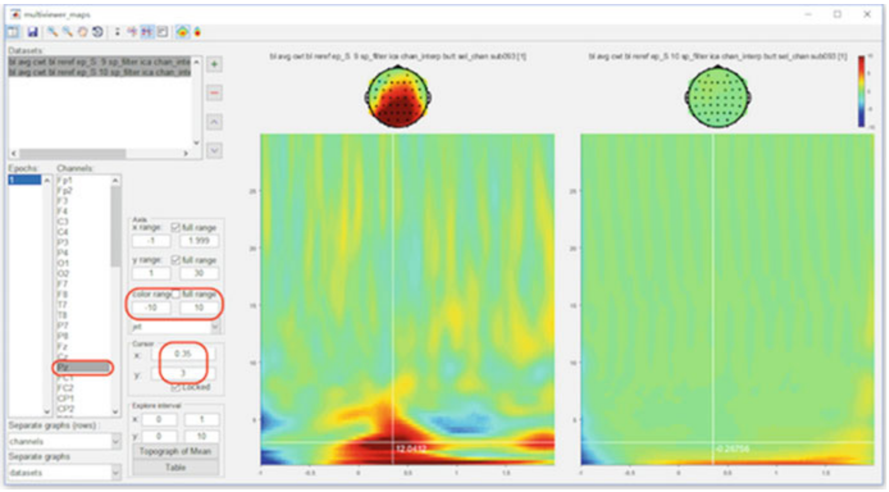
Fig. 19.9 The result of time domain analysis for P300 on single-subject level

19.7.10 Continuous Wavelet Transform

To perform CWT for time-frequency analysis before the time domain averaging, select the datasets *bl reref ep_S 9 sp_filter ica chan_interp butt sel_chan sub093* and *bl reref ep_S 10 sp_filter ica chan_interp butt sel_chan sub09*3, and click **Plugins->my_tfa->Averaged CWT** in the menu. Keep the default parameter setting in the batch module, and click the button **Run** in the bottom of batch module. Two new datasets with the name *avg cwt bl reref ep_S 9 sp_filter ica chan_interp butt sel_chan sub093* and *avg cwt bl reref ep_S 10 sp_filter ica chan_interp butt sel_chan sub093* will be appeared in the manage module.

The operation of **Plugins->my_tfa->Averaged CWT** combines the steps of CWT and averaging together, which can be performed by **Process->Frequency analysis and filters->CWT (Continuous Wavelet Transform)** and **Process->Average->Compute average, std, median across epochs**, respectively, in the menu. Since the time-frequency analysis is time-consuming and also need a larger space for storage, the computer with small memory would easily go to the error of "out of memory". Hence, we combine the steps of time-frequency analysis and averaging together as a plugin for saving the storage space.

After the time-frequency analysis, baseline correction in time-frequency domain is necessary as it is in time domain. Select the datasets *avg cwt bl reref ep_S 9 sp_filter ica chan_interp butt sel_chan sub093* and *avg cwt bl reref ep_S 10 sp_filter ica chan_interp butt sel_chan sub093*, and click **Process->Baseline operation-> Baseline correction** in the menu. It is suggested to shrink the interval for baseline correction in time-frequency domain. Hence, we set the **-0.75** to **-0.25**



Axis
x range: ☑ full range -1 1.999
y range: ☑ full range 1 30
color range ☐ full range -10 10
jet
Cursor
x: 0.35
y: 3
☑ Locked
Explore interval
x: 0 1
y: 0 10
Separate graphs (rows): channels
Separate graphs: datasets

Fig. 19.10 The result of time-frequency analysis for P300 on single-subject level

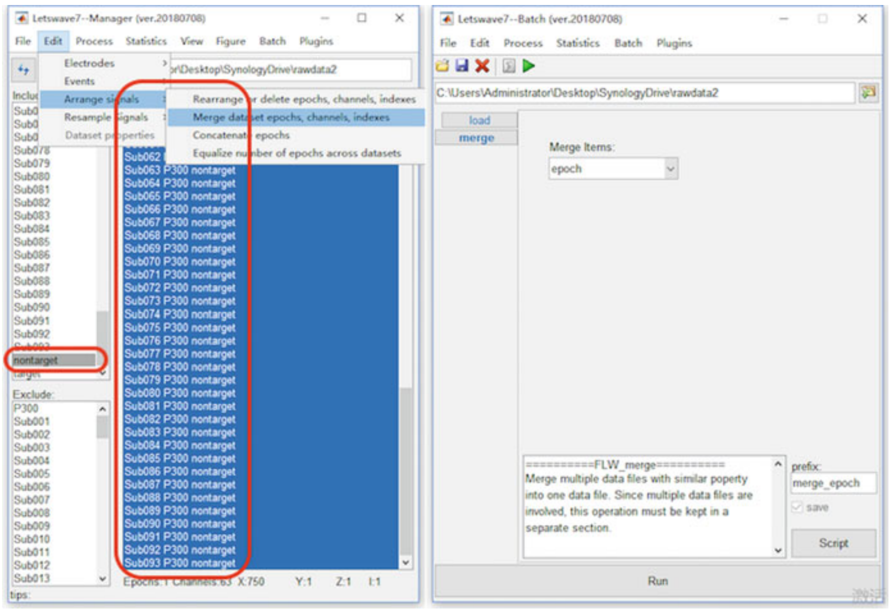
seconds as the baseline in the batch module. Click the button **Run** in the bottom of batch module for baseline correction. Two new datasets with the name *bl avg cwt bl reref ep_S 9 sp_filter ica chan_interp butt sel_chan sub093* and *bl avg cwt bl reref ep_S 10 sp_filter ica chan_interp butt sel_chan sub093* will be appeared in the manage module.

Select datasets *bl avg cwt bl reref ep_S 9 sp_filter ica chan_interp butt sel_chan sub093* and *bl avg cwt bl reref ep_S 10 sp_filter ica chan_interp butt sel_chan sub093*. Click the **view** in the right-click menu, set the separate graphs (columns) as **datasets**, select both the two datasets, select channel **Pz**, and set color range from **-10** to **10**; the result of time-frequency analysis can be observed in the multiviewer (Fig. 19.10). Open the topography, and set the cursor to **x=0.35** and **y=3**; we can observe the phase-locked P300 activity on the topography.

19.8 Example for Multiple-Subject Analysis

19.8.1 Datasets Merge

Based on the single-subject analysis, we will focus on the multiple-subject analysis in this part. Before this, we need to arrange the datasets. For ease of the group analysis, we need firstly to downsample the datasets from 1000Hz to 250Hz by clicking **Edit->Resample Signals->Downsample signals (integer ratio)** and setting the down sampling ratio to 4. And then we need to rename these datasets, since the filename is too long for displaying. Select these dataset *ds avg bl reref ep_S 9 sp_filter ica chan_interp butt sel_chan sub093*, and press the **rename** in the right-


Letswave7--Manager (ver.20180708)
File Edit Process Statistics View Figure Batch Plugins
Electrodes
Events
Arrange signals
Resample signals
Dataset properties
Rearrange or delete epochs, channels, indexes
Merge dataset epochs, channels, indexes
Concatenate epochs
Equalize number of epochs across datasets
Include
nontarget
target
Exclude:
tips:
Letswave7--Batch (ver.20180708)
File Edit Process Statistics Batch Plugins
load
merge
Script
Run

or\Desktop\SynologyDrive\rawdata2

Sub078
Sub079
Sub080
Sub081
Sub082
Sub083
Sub084
Sub085
Sub086
Sub087
Sub088
Sub089
Sub090
Sub091
Sub092

P300
Sub001
Sub002
Sub003
Sub004
Sub005
Sub006
Sub007
Sub008
Sub009
Sub010
Sub011
Sub012
Sub013

Sub062 P300 nontarget
Sub063 P300 nontarget
Sub064 P300 nontarget
Sub065 P300 nontarget
Sub066 P300 nontarget
Sub067 P300 nontarget
Sub068 P300 nontarget
Sub069 P300 nontarget
Sub070 P300 nontarget
Sub071 P300 nontarget
Sub072 P300 nontarget
Sub073 P300 nontarget
Sub074 P300 nontarget
Sub075 P300 nontarget
Sub076 P300 nontarget
Sub077 P300 nontarget
Sub078 P300 nontarget
Sub079 P300 nontarget
Sub080 P300 nontarget
Sub081 P300 nontarget
Sub082 P300 nontarget
Sub083 P300 nontarget
Sub084 P300 nontarget
Sub085 P300 nontarget
Sub086 P300 nontarget
Sub087 P300 nontarget
Sub088 P300 nontarget
Sub089 P300 nontarget
Sub090 P300 nontarget
Sub091 P300 nontarget
Sub092 P300 nontarget
Sub093 P300 nontarget

Epochs:1 Channels:63 X:750 Y:1 Z:1 I:1

C:\Users\Administrator\Desktop\SynologyDrive\rawdata2

Merge Items: epoch

==========FLW_merge==========
Merge multiple data files with similar poperty into one data file. Since multiple data files are involved, this operation must be kept in a separate section.

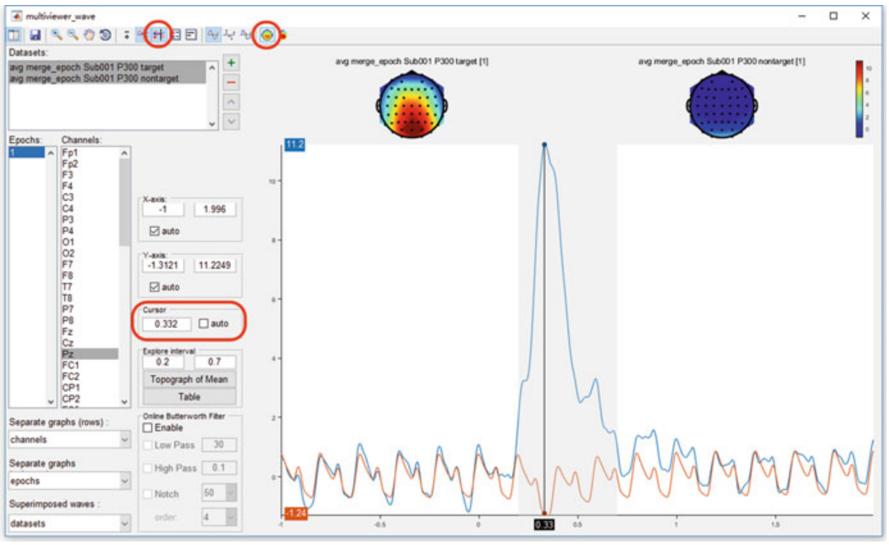
prefix: merge_epoch
save

Fig. 19.11 Datasets merge

clicked menu, and rename the selected dataset to *Sub093 P300 target*. Similarly, rename the dataset *ds avg bl reref ep_S 10 sp_filter ica chan_interp butt sel_chan sub093* to *Sub093 P300 nontarget*.

Copy all the results on single-subject level to a new folder such as **rawdata2** here. In this example, all 93∗2 files, which are the average of all the epochs in the two conditions of target and nontarget from 93 subjects, can be downloaded here (https://github.com/NOCIONS/letswave7_tutorial/raw/master/rawdata2.zip). For multiple-subject analysis, we need to merge the averaged P300s from 93 subjects into one dataset, in which each subject in the new dataset is treated as an epoch. Select the tag **nontarget** in the **include** list box from the left part of the manage module (Fig. 19.11), and select all the datasets in the dataset list box in the right part of the manage module. Select **Edit->Arrange signals->Merge dataset epochs, channels, indexes**, keep the default setting in the batch module, and click the button **Run** to merge all these selected datasets in the condition of nontarget into one dataset *merge_epoch Sub001 P300 nontarget*. In the same way, we can merge all the datasets in the condition of target by selecting the tag **target** in the **include** list box and get the dataset *merge_epoch Sub001 P300 target* with 93 epochs.

Select the tag **merge_epoch** in the **include** list box, and select all the datasets in the manage module. Similar with the averaging operation on the single-subject level, select **Process->Average->Compute average, std, median across epochs**. Keep the default setting in the batch module, and click the button **Run** to average all the epochs in the two datasets, which is actually run the grand average for all the subjects



avg merge_epoch Sub001 P300 target
avg merge_epoch Sub001 P300 nontarget
Epochs: 1
X-axis: -1 1.996 ☑ auto
Y-axis: -1.3121 11.2249 ☑ auto
Cursor: 0.332 ☐ auto
Explore interval: 0.2 0.7
Online Butterworth Filter: ☐ Enable
Low Pass 30
High Pass 0.1
Notch 50
order 4
Separate graphs (rows) : channels
Separate graphs: epochs
Superimposed waves : datasets

Fig. 19.12 The time domain result of the P300 for multiple-subject analysis

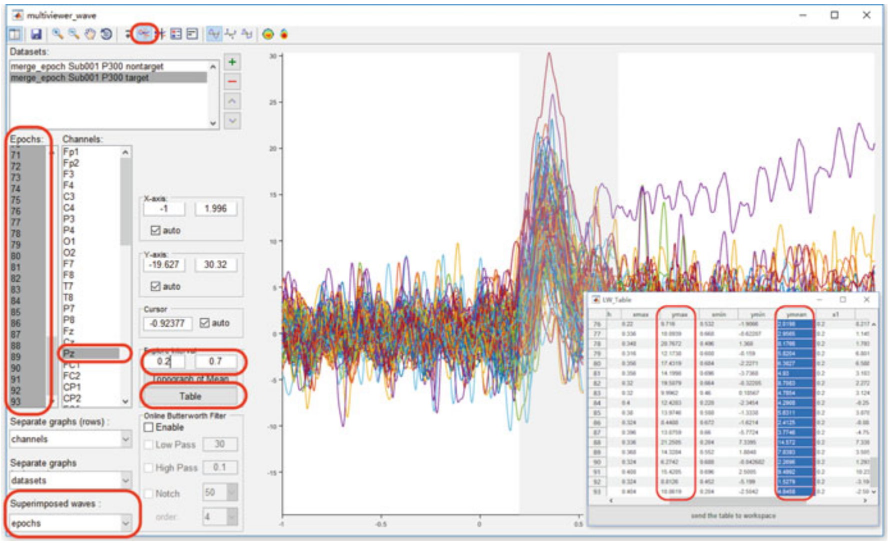
in the conditions of target and nontarget. Two new datasets with the name *avg merge_epoch Sub001 P300 nontarget* and *avg merge_epoch Sub001 P300 target* will be appeared in the manage module.

To view the result of grand average, select datasets *avg merge_epoch Sub001 P300 nontarget* and *avg merge_epoch Sub001 P300 target*, click the **view** in the right-click menu, and select channel **Pz**. To observe the results with the same order as it is in the single-subject level, select the dataset **avg merge_epoch Sub001 P300 target**, and click the button **dataset up**. Enabling the **cursor** and the **2D topography** in the toolbar and setting the location of the cursor to be **0.332**, the grand average topography of the P300 experiment can be observed in both the conditions of target and nontarget at the peak time point of the P300 component (Fig. 19.12).

19.8.2 Statistical Analysis on Predefined Interval

With the hypothesis that **the maximum or mean value of ERP from the predefined interval 0.2 to 0.7s at channel Pz would be different between target and nontarget conditions**, paired-sample t-test will be performed on the 93 subjects.

Firstly, we need to pick up the maximum and mean value from the interval 0.2 to 0.7s at channel Pz for both target and nontarget conditions. Select the datasets *merge_epoch Sub001 P300 nontarget* and *merge_epoch Sub001 P300 target*, and click the **view** in the right-click menu. Select dataset **merge_epoch Sub001 P300 target** and channel **Pz**, set the **Superimposed waves** as **epochs**, enable the **interval selection** in the toolbar, and set the explore interval as **0.2–0.7**s. Press the button



```
Datasets:
merge_epoch Sub001 P300 nontarget
merge_epoch Sub001 P300 target
X-axis: -1  1.996  [✓ auto]
Y-axis: -19.627  30.32  [✓ auto]
Cursor: -0.92377  [✓ auto]
Explore interval: 0.2  0.7
Topograph of Mean
Table
Online Butterworth Filter
Enable
Low Pass 30
High Pass 0.1
Notch 50
order 4
Separate graphs (rows): channels
Separate graphs: datasets
Superimposed waves: epochs
```

	h	xmax	ymax	xmin	ymin	ymean	x1
76		0.22	9.716	0.532	-1.9066	2.0198	0.2	8.217
77		0.336	10.0939	0.668	-0.62207	2.9565	0.2	1.140
78		0.348	20.7672	0.496	1.368	8.1766	0.2	1.793
79		0.316	12.1730	0.600	-6.159	5.8204	0.2	6.801
80		0.356	17.4319	0.604	-2.2271	6.3627	0.2	6.588
81		0.356	14.1998	0.696	-3.7368	4.93	0.2	3.163
82		0.32	19.5079	0.664	-0.32295	8.7063	0.2	2.272
83		0.32	9.9962	0.46	0.10567	4.7654	0.2	3.124
84		0.4	12.4283	0.228	-2.3454	4.2908	0.2	-6.25
85		0.38	13.9746	0.568	-1.3338	5.6311	0.2	3.670
86		0.324	8.4400	0.672	-1.6214	2.4125	0.2	-8.88
87		0.396	13.0759	0.68	-5.7724	3.7746	0.2	-4.75
88		0.336	21.2585	0.264	7.3395	14.572	0.2	7.338
89		0.368	14.3284	0.552	1.8848	7.8383	0.2	3.505
90		0.324	6.2742	0.688	-0.042682	2.2698	0.2	1.293
91		0.408	15.4205	0.696	2.5005	9.4892	0.2	10.23
92		0.324	8.8126	0.452	-5.199	1.5279	0.2	-3.19
93		0.404	10.8619	0.204	-2.5042	4.8458	0.2	-2.50

send the table to workspace

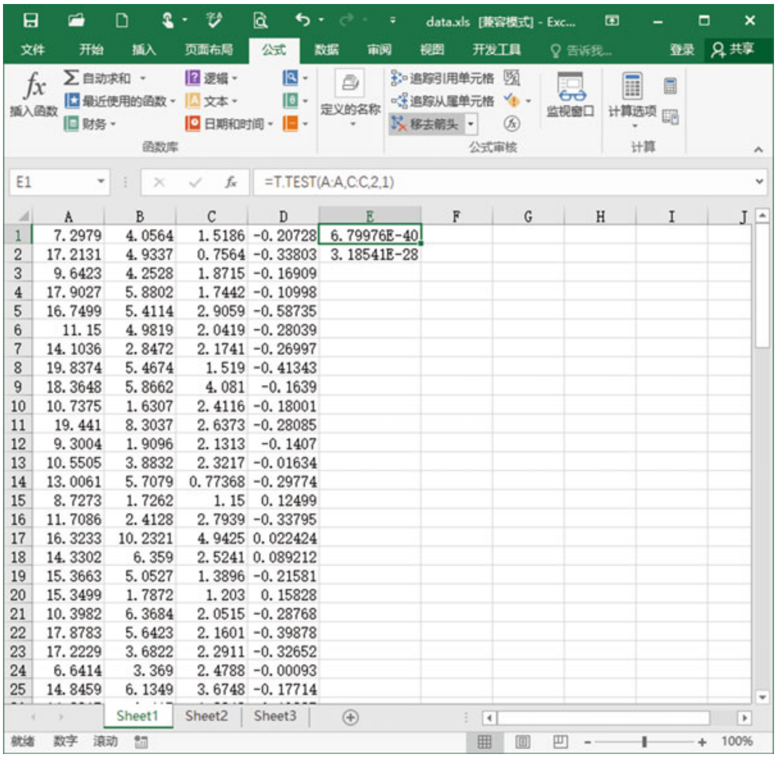
Fig. 19.13 Detect the peak and mean value of the P300 for all subjects in the interval of 0.2–0.7s at channel Pz

Table; a table is popped up with the statistic of the maximum and mean value; copy these data to the external software, such as Excel or SPSS, for the further statistical analysis. Similarly, select the dataset *merge_epoch Sub001 P300 nontarget*, and press the button **Table**; pick up the same value on another dataset, and copy it to the external software (Fig. 19.13).

Here the statistical inferences are done in Excel. Previously, we copy the maximum and mean value in the condition of **target** to columns A and B and copy the maximum and mean value in the condition of **nontarget** to columns C and D. In item **E1**, input "**=T.TEST(A:A,C:C,2,1)**" for the paired-sample t-test on the maximum value. The result **p** = **6.8** ∗ **10^-40** indicates that for the maximum value of the EEG signal in the interval 0.2–0.7s on channel Pz, there is a significant difference between the conditions of target and nontarget. Similarly, input "**=T.TEST(A:A,C:C,2,1)**" in item **E2** for the paired-sample t-test on the mean value. The result **p** = **3.2** ∗ **10^-28** indicates that for the mean value of the EEG signal in the interval 0.2–0.7s on channel Pz, there is a significant difference between the conditions of target and nontarget (Fig. 19.14).

19.8.3 Point-Wised Statistical Analysis

Different from the hypothesis-driven approach as is shown above, we can also do the point-wised statistical analysis, which is a data-driven method without any previous experience for interval selection. Select the datasets *merge_epoch Sub001 P300*



E1 =T.TEST(A:A,C:C,2,1)

	A	B	C	D	E
1	7. 2979	4. 0564	1. 5186	-0. 20728	6. 79976E-40
2	17. 2131	4. 9337	0. 7564	-0. 33803	3. 18541E-28
3	9. 6423	4. 2528	1. 8715	-0. 16909	
4	17. 9027	5. 8802	1. 7442	-0. 10998	
5	16. 7499	5. 4114	2. 9059	-0. 58735	
6	11. 15	4. 9819	2. 0419	-0. 28039	
7	14. 1036	2. 8472	2. 1741	-0. 26997	
8	19. 8374	5. 4674	1. 519	-0. 41343	
9	18. 3648	5. 8662	4. 081	-0. 1639	
10	10. 7375	1. 6307	2. 4116	-0. 18001	
11	19. 441	8. 3037	2. 6373	-0. 28085	
12	9. 3004	1. 9096	2. 1313	-0. 1407	
13	10. 5505	3. 8832	2. 3217	-0. 01634	
14	13. 0061	5. 7079	0. 77368	-0. 29774	
15	8. 7273	1. 7262	1. 15	0. 12499	
16	11. 7086	2. 4128	2. 7939	-0. 33795	
17	16. 3233	10. 2321	4. 9425	0. 022424	
18	14. 3302	6. 359	2. 5241	0. 089212	
19	15. 3663	5. 0527	1. 3896	-0. 21581	
20	15. 3499	1. 7872	1. 203	0. 15828	
21	10. 3982	6. 3684	2. 0515	-0. 28768	
22	17. 8783	5. 6423	2. 1601	-0. 39878	
23	17. 2229	3. 6822	2. 2911	-0. 32652	
24	6. 6414	3. 369	2. 4788	-0. 00093	
25	14. 8459	6. 1349	3. 6748	-0. 17714	

Fig. 19.14 Paired-sample t-test in Excel

nontarget and *merge_epoch Sub001 P300 target*, and click **Statistics->Compare two datasets (paired sample/two sample t-test)**. Keep the default setting in the batch module, and click the button **Run** to get the point-wise t-test result *ttest merge_epoch Sub001 P300 target* in the manage module.

Select dataset *ttest merge_epoch Sub001 P300 target*, and click the **view** in the right-click menu; select the channel as **Pz**, and keep the index as **p-value**. Since we set the significant level $\alpha = 0.05$, we can set the Y-axis as **0–0.05** to watch the interval with the p-value lower than 0.05. It could be found except the main cluster around 0.2–0.7 seconds; there are still several clusters in the other time intervals, even before the stimulus, which indicates a high family-wise error rate (Fig. 19.15). Bonferroni correction can be done by simply setting the y-axis as **0–0.000001**. Since there are 750 time points and 63 channels in the test, the corrected α value by Bonferroni method is **0.05/750/63 = 10^-6.**

Cluster-based permutation test is an efficient way to reduce family-wise error rate. In the same way in the point-wise t-test, select the datasets *merge_epoch Sub001 P300 nontarget* and *merge_epoch Sub001 P300 target*, and click **Statistics->Compare two datasets (paired sample/two sample t-test)**. Enable the bottom panel for the **cluster-based permutation test**. To get a more precise result, set the number of

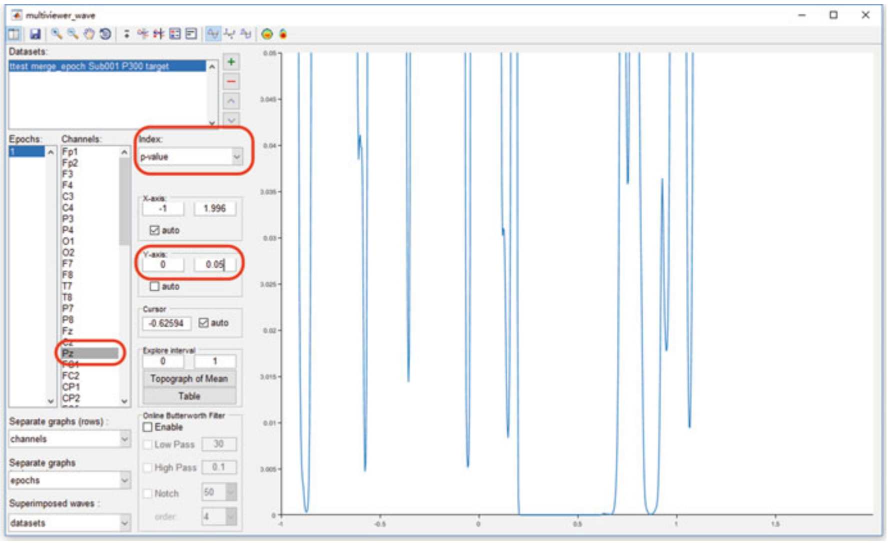


Fig. 19.15 Point-wised paired-sample t-test result on channel Pz

permutations as **20000**, which would take more time for computing. Click the button **Run**; the result would be kept in the dataset with the same name *ttest merge_epoch Sub001 P300 target*.

Open the dataset *ttest merge_epoch Sub001 P300 target* by clicking the **view** in the right-click menu. The uncorrected result is exactly the same as it is in the point-wise t-test. Set the index as "**cluster p-value**", and set the cursor as **0.332**, which is the peak of the P300 component. It can be found that only the main cluster on channel Pz is reserved after cluster-based permutation test; the other clusters are excluded as false positive (Fig. 19.16).

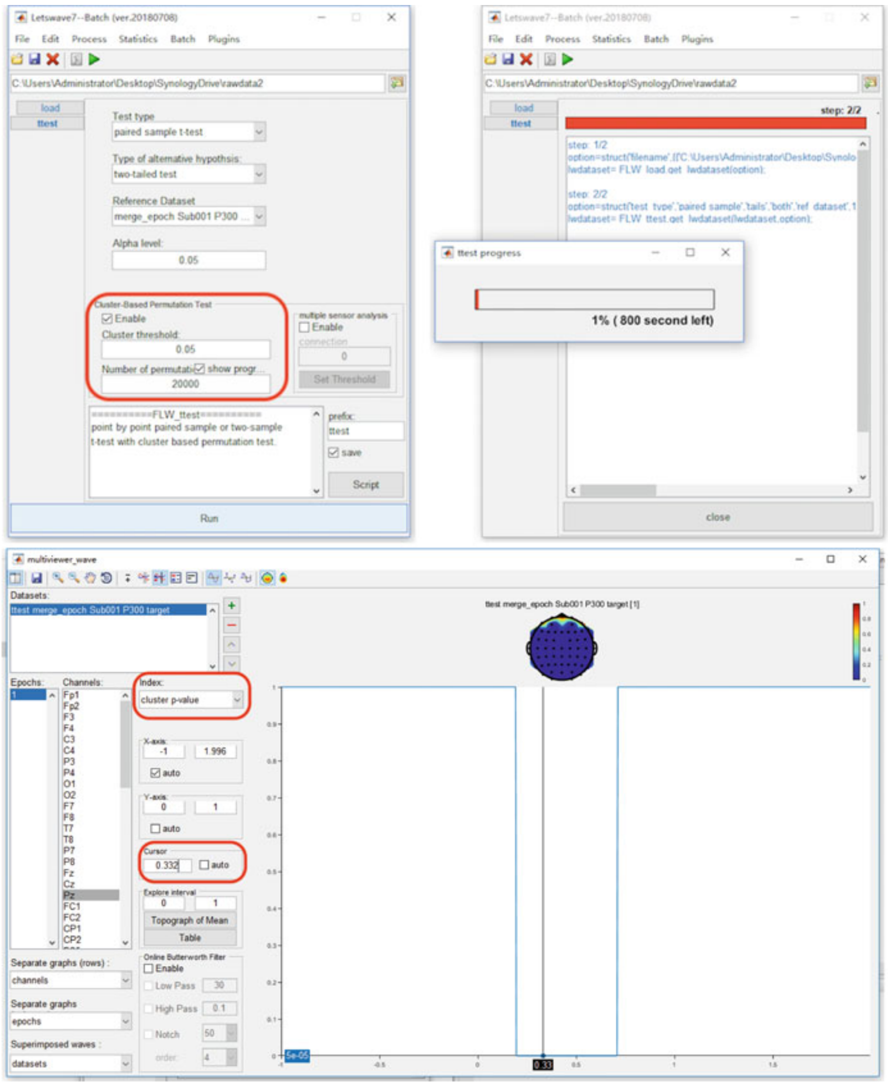
19.9 Figure Generation and Batch Processing

19.9.1 Figure Generation

Figure generation is an important feature in Letswave7. Based on the result of multiple-subject analysis and statistical analysis, we can generate Fig. 19.17 by the following steps:

Step1: *Open the figure module*

- Select the datasets *avg Sub001 P300 nontarget* and *avg Sub001 P300 target* in the folder of rawdata1.
- Click **Figure->General Figure creator** to open the figure module with a blank canvas.


Letswave7--Batch (ver.20180708)

C:\Users\Administrator\Desktop\SynologyDrive\rawdata2

Test type: paired sample t-test

Type of alternative hypothesis: two-tailed test

Reference Dataset: merge_epoch Sub001 P300 ...

Alpha level: 0.05

Cluster-Based Permutation Test: [✓ Enable]

Cluster threshold: 0.05

Number of permutations: 20000 [✓ show progress]

multiple sensor analysis: Enable

connection: 0

==========FLW_ttest==========
point by point paired sample or two-sample t-test with cluster based permutation test.

prefix: ttest [✓ save]

```
step: 1/2
option=struct('filename',{{'C:\Users\Administrator\Desktop\Synolo
lwdataset= FLW_load.get_lwdataset(option);

step: 2/2
option=struct('test_type','paired sample','tails','both','ref_dataset',1
lwdataset= FLW_ttest.get_lwdataset(lwdataset,option);
```

step: 2/2

ttest progress: 1% (800 second left)

multiviewer_wave

Datasets: ttest merge_epoch Sub001 P300 target

Epochs: 1

Channels: Fp1, Fp2, F3, F4, C3, C4, P3, P4, O1, O2, F7, F8, T7, T8, P7, P8, Fz, Cz, Pz, FC1, FC2, CP1, CP2

Index: cluster p-value

X-axis: -1 1.996 [✓ auto]

Y-axis: 0 1 auto

Cursor: 0.332 auto

Explore interval: 0 1

Online Butterworth Filter: Enable; Low Pass 30; High Pass 0.1; Notch 50; order 4

Separate graphs (rows): channels

Separate graphs: epochs

Superimposed waves: datasets

ttest merge_epoch Sub001 P300 target [1]

Fig. 19.16 The result from cluster-based permutation test

<u>Step2</u>: <u>*Create the subfigures*</u>

- Set the width and height of the figure as **1000** and **400**.
- Add a **curve** with the title **P300**, font size **12**, and position **x=80**, **y= 70**, **w=870**, and **h=300**.
- Add a **topography** with the title **0 ms**, font size **12**, and position **x= 100**, **y=180**, **w=120**, and **h=120**.

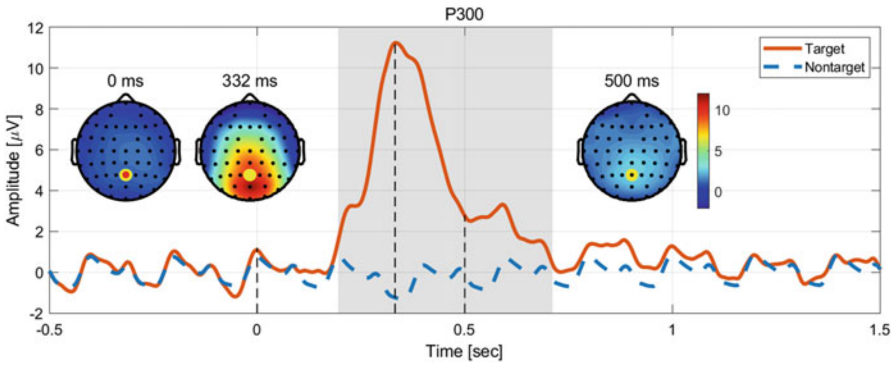


Fig. 19.17 The result from cluster-based permutation test

- Add a **topography** with the title **332 ms**, font size **12**, and position **x= 230**, **y=180**, **w=120**, and **h=120**.
- Add a **topography** with the title **500 ms**, font size **12**; enable the color bar in the **content** panel and position **x=100**, **y=180**, **w=120**, and **h=120**.

Step3: *Add the content*

- Click the **content** button in the toolbar, and select **P300** in the subfigure. Add a **curve** with width as **3**, the data source as **avg merge_epoch Sub001 P300 target**, and channel as **Pz**.
- Add a **curve** with the color as blue **[0,0.45,0.74]**, width as **3**, line style as **dashed**, data source as **Sub093 P300 nontarget**, and channel as **Pz**.
- Add a **rect** with face opacity **0.25**, edge opacity **0**, position **x = 0.196**, **y= -2**, **w = 0.516**, and **h = 14**. And sort the rect above the **curve1** and **curve2**.
- Add a **line** with width as **1**, line style as **dashed**, and the position as **x1=0**, **y1=-2**, **x2=0**, and **y2=1.2**.
- Add a **line** with width as **1**, line style as **dashed**, and the position as **x1=0.332**, **y1=-2**, **x2=0.332**, and **y2=11.3**.
- Add a **line** with width as **1**, line style as **dashed**, and the position as **x1=0.5**, **y1=-2**, **x2=0.5**, and **y2=2.8**.
- Select **0 ms** in the subfigure. Set the data source as "**avg merge_epoch Sub001 P300 Target**", x from **0** to **0**, head radius as **0.5**, shrink as **0.95**, range from **-2** to **12** to keep it the same as the range of P300 curve. For the electrodes, set the size as **8** and marker the channel **Pz**.
- Select **332 ms** in the subfigure. Set the data source as "**avg merge_epoch Sub001 P300 Target**", x from **0.332** to **0.332**, head radius as **0.5**, shrink as **0.95**, range from **-2** to **12** to keep it the same as the range of P300 curve. For the electrodes, set the size as **8** and marker the channel **Pz**.
- Select **500 ms** in the subfigure. Set the data source as "**avg merge_epoch Sub001 P300 Target**", x from **0.5** to **0.5**, head radius as **0.5**, shrink as **0.95**,

range from **-2** to **12** to keep it the same as the range of P300 curve. For the electrodes, set the size as **8** and marker the channel **Pz**.

Step4: *Set the axis parameters*

- Click the **axis** button in the toolbar, select **P300** in the subfigure, and enable the **Box** and **Legend**.
- Select **curve1** in the list box of the content; change its name to **Target** in the legend. Select **curve2** in the list box of the content; change its name to **NonTarget** in the legend.
- For x-axis, set the x-lim from **-0.5** to **1.5** second. Enable the **grid** and the **label**; set label as "**Time [sec]**".
- For y-axis, enable the **grid** and the **label**; set the label as "**Amp [\muV]**".

19.9.2 Batch Processing

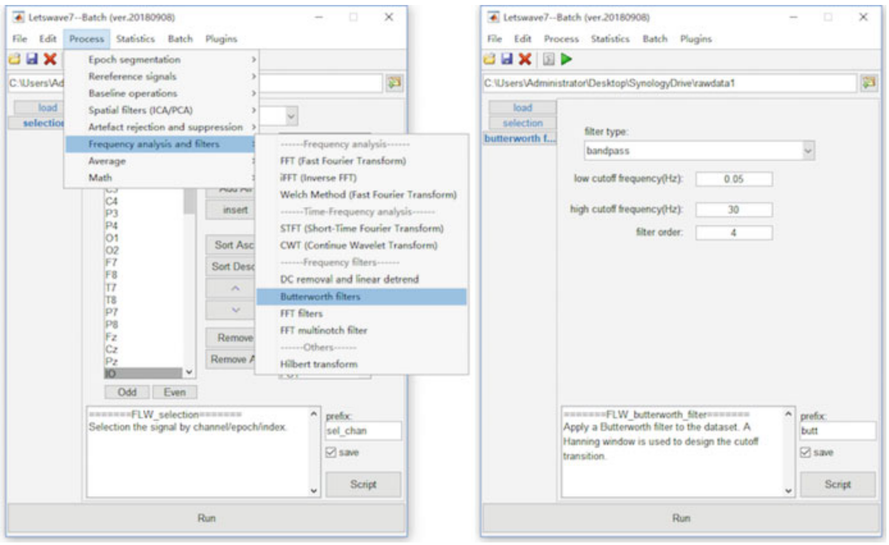
In this part, all the time domain analysis on single-subject level will be done again but using batch processing. After data importing, remove channel IO, which is exactly the same operation as what we do in Sect. 19.3. After that, click the tab **selection** in the left, and select **Process -> Frequency analysis and filters -> Butterworth filters** in the menu of the **batch module**, and set the low cutoff frequency (Hz) as **0.05Hz** (Fig. 19.18). From this step, the operation in the batch processing would be different. We call out the Butterworth filter from the batch module, **NOT** the manage module.

Step by step, add all the steps in the batch module to get the final processing flow as it is shown in Fig. 19.19. By pressing the button "**Run**", we can run the whole batch processing. In the step of **Identify Artifact Component**, we still manually select **comp 1** and **comp 2** in the right panel in orange color and click the button **OK**. It needs more than 2 minutes for all the processing on the testing, in which the step of computing ICA matrix is time-consuming.

To reuse the batch, we can save the entire process. Click the **save** button in the toolbar of the batch module, and save the process as "P300.lw_script", for example. For the next time, we can click the **open** button to load the process. It should be noticed that since the processing object has been changed, the users need to change the input datasets in all the steps of **load**.

If the batch processing is frequently used, we can put the saved **.lw_script** file, like "P300.lw_script" in this case, into the path of the Letswave installed "**../letswave7/plugins/**". After that, restart Letswave7, and the batch operation will come out in the menu of the batch in both the manage module and batch module.

Once the processing flow has been made, the corresponding MATLAB script has already been generated automatically. Users can click the "script" button in the batch module to get the full script. Once the users have learned the use of GUI, they almost already understand the writing of script.


Letswave7--Batch (ver.20180908)
File Edit Process Statistics Batch Plugins
Epoch segmentation
Rereference signals
Baseline operations
Spatial filters (ICA/PCA)
Artefact rejection and suppression
Frequency analysis and filters
Average
Math
------Frequency analysis------
FFT (Fast Fourier Transform)
iFFT (Inverse FFT)
Welch Method (Fast Fourier Transform)
------Time-Frequency analysis------
STFT (Short-Time Fourier Transform)
CWT (Continue Wavelet Transform)
------Frequency filters------
DC removal and linear detrend
Butterworth filters
FFT filters
FFT multinotch filter
------Others------
Hilbert transform
load
selection
butterworth f...
Odd
Even
insert
Sort Asc
Sort Desc
Remove
prefix:
save
Script
Run

C:\Users\Administrator\Desktop\SynologyDrive\rawdata1

filter type: bandpass

low cutoff frequency(Hz): 0.05

high cutoff frequency(Hz): 30

filter order: 4

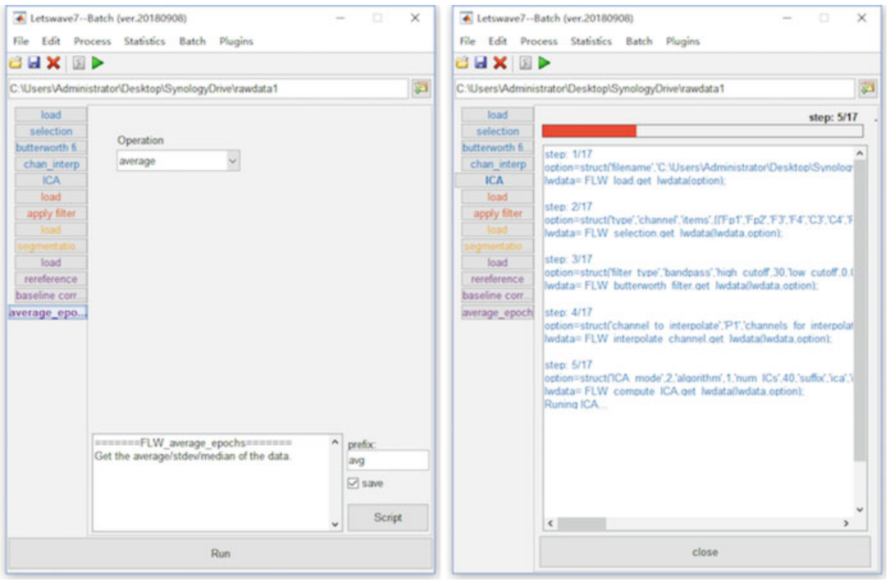
=======FLW_selection=======
Selection the signal by channel/epoch/index.

prefix: sel_chan

=======FLW_butterworth_filter=======
Apply a Butterworth filter to the dataset. A Hanning window is used to design the cutoff transition.

prefix: butt

Fig. 19.18 Add the step of frequency filtering in batch module



C:\Users\Administrator\Desktop\SynologyDrive\rawdata1

Operation: average

=======FLW_average_epochs=======
Get the average/stdev/median of the data.

prefix: avg

step: 5/17

```
step: 1/17
option=struct('filename','C:\Users\Administrator\Desktop\Synolog
lwdata= FLW_load.get_lwdata(option);

step: 2/17
option=struct('type','channel','items',{{'Fp1','Fp2','F3','F4','C3','C4','F
lwdata= FLW_selection.get_lwdata(lwdata,option);

step: 3/17
option=struct('filter_type','bandpass','high_cutoff',30,'low_cutoff',0.0
lwdata= FLW_butterworth_filter.get_lwdata(lwdata,option);

step: 4/17
option=struct('channel_to_interpolate','P1','channels_for_interpolat
lwdata= FLW_interpolate_channel.get_lwdata(lwdata,option);

step: 5/17
option=struct('ICA_mode',2,'algorithm',1,'num_ICs',40,'suffix','ica','
lwdata= FLW_compute_ICA.get_lwdata(lwdata,option);
Runing ICA...
```

Fig. 19.19 Add the step of frequency filtering in batch module

References

Barnett L, Seth AK. The MVGC multivariate Granger causality toolbox: a new approach to Granger-causal inference. J Neurosci Methods. 2014;223:50–68.

Bokil H, Andrews P, Kulkarni JE, Mehta S, Mitra PP. Chronux: a platform for analyzing neural signals. J Neurosci Methods. 2010;192(1):146–51.

Delorme A, Makeig S. EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis. J Neurosci Methods. 2004;134(1):9–21.

Delorme A, Mullen T, Kothe C, Acar ZA, Bigdely-Shamlo N, Vankov A, Makeig S. EEGLAB, SIFT, NFT, BCILAB, and ERICA: new tools for advanced EEG processing. Comp Intell Neurosci. 2011;2011:10.

Gramfort A, Luessi M, Larson E, Engemann DA, Strohmeier D, Brodbeck C, Goj R, Jas M, Brooks T, Parkkonen L, Hämäläinen M. MEG and EEG data analysis with MNE-Python. Front Neurosci. 2013;7:267.

Lawhern VJ, Solon AJ, Waytowich NR, Gordon SM, Hung CP, Lance BJ. Eegnet: a compact convolutional network for eeg-based brain-computer interfaces. arXiv preprint arXiv. 2016:1611.08024.

Lindner M, Vicente R, Priesemann V, Wibral M. TRENTOOL: a Matlab open source toolbox to analyse information flow in time series data with transfer entropy. BMC Neurosci. 2011;12 (1):119.

Litvak V, Mattout J, Kiebel S, Phillips C, Henson R, Kilner J, Barnes G, Oostenveld R, Daunizeau J, Flandin G, Penny W. EEG and MEG data analysis in SPM8. Comp Intell Neurosci. 2011;2011

Niso G, Bruña R, Pereda E, Gutiérrez R, Bajo R, Maestú F, del-Pozo F. HERMES: towards an integrated toolbox to characterize functional and effective brain connectivity. Neuroinformatics. 2013;11(4):405–34.

Oostenveld R, Fries P, Maris E, Schoffelen JM. FieldTrip: open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data. Comput Intell Neurosci. 2011;2011:1.

Tadel F, Baillet S, Mosher JC, Pantazis D, Leahy RM. Brainstorm: a user-friendly application for MEG/EEG analysis. Comput Intell Neurosci. 2011;2011:8.