# Table of Contents

1 A12: Quick Tutorial on Rejection
  1.1 .................................................................................................................. 1

2 A13: Compiled EEGLAB
  2.1 .................................................................................................................... 3
  2.2 .................................................................................................................... 3
  2.3 .................................................................................................................... 3
  2.4 .................................................................................................................... 3
  2.5 .................................................................................................................... 3
  2.6 .................................................................................................................... 4
  2.7 .................................................................................................................... 4
  2.8 .................................................................................................................... 4

3 Binica
  3.1 .................................................................................................................... 5
  3.2 .................................................................................................................... 5

4 Channel Location Files
  4.1 .................................................................................................................... 7

5 Downloading and contributing to EEGLAB from GitHub
  5.1 .................................................................................................................... 9
  5.2 .................................................................................................................... 9
  5.3 .................................................................................................................... 9

6 Edit EEGLAB code ......................................................................................... 10

7 EEGLAB and high performance computing
  7.1 .................................................................................................................... 11

8 ...................................................................................................................... 12
  8.1 .................................................................................................................... 12

9 ...................................................................................................................... 15
  9.1 .................................................................................................................... 15
  9.2 .................................................................................................................... 15

10 EEGLAB Bugs
  10.1 .................................................................................................................. 17

11 EEGLAB Extensions details
  11.1 .................................................................................................................. 18

12 EEGLAB Menu Functions
  12.1 .................................................................................................................. 22

13 EEGLAB revision history
  13.1 .................................................................................................................. 25
  13.2 .................................................................................................................. 25
  13.3 .................................................................................................................. 25
  13.4 .................................................................................................................. 25
  13.5 .................................................................................................................. 25

14 EEGLAB revision history version 10
  14.1 .................................................................................................................. 27
  14.2 .................................................................................................................. 27
  14.3 .................................................................................................................. 27
  14.4 .................................................................................................................. 28
  14.5 .................................................................................................................. 28
  14.6 .................................................................................................................. 29
  14.7 .................................................................................................................. 29
  14.8 .................................................................................................................. 29
# Table of Contents

31 Independent Component Clustering Example ............................................................................................................. 62
  31.1 ........................................................................................................................................................................... 62

32 Category: IV. Appendix .................................................................................................................................................. 65

33 IV. Appendix .......................................................................................................................................................... 66

34 Mex files in EEGLAB .................................................................................................................................................. 67

35 TIPS and FAQ .......................................................................................................................................................... 68

36 ........................................................................................................................................................................................ 69
  36.1 ........................................................................................................................................................................... 69
  36.2 ........................................................................................................................................................................... 70
  36.3 ........................................................................................................................................................................... 71

37 ........................................................................................................................................................................................ 73
  37.1 ........................................................................................................................................................................... 73
  37.2 ........................................................................................................................................................................... 73
  37.3 ........................................................................................................................................................................... 74
  37.4 ........................................................................................................................................................................... 76
  37.5 ........................................................................................................................................................................... 78

38 ........................................................................................................................................................................................ 79
  38.1 ........................................................................................................................................................................... 79
  38.2 ........................................................................................................................................................................... 79
  38.3 ........................................................................................................................................................................... 80
1 A12: Quick Tutorial on Rejection

1.1

Independent Component Analysis is a powerful tool for eliminating several important types of non-brain artifacts from EEG data. EEGLAB allows the user to reject many such artifacts in an efficient and user-friendly manner. This short tutorial is designed to guide impatient users who want to try using EEGLAB to remove artifacts from their data. The many other capabilities of EEGLAB are explained in detail in the main EEGLAB tutorial. To perform artifact rejection:

1.1.1

Type >> eeglab to start EEGLAB under Matlab. Select menu item File > Import data to import your data file in any of a variety of file formats including EGI and Neuroscan binary. See the Import data appendix for more details.

Scroll and check data using menu item Plot > Channel data (scroll).

1.1.1.1

Importing a channel location file is critical for visualizing the independent components of your data.

Select menu item Edit > Channel locations and press the button Read locations in the bottom right corner of the channel edit window. The program recognizes channel location files in most known formats (spherical BESA, polar Neuroscan, 3-D cartesian EGI, 3-D cartesian Polhemus, ...). Press OK after selecting the file and then press OK to have EEGLAB recognize the file format automatically from the file extension (Note: files with extension ".elp" are considered Polhemus files by default and not BESA files). Press OK in the channel edit window to import the channel locations into EEGLAB.

To check that your channel locations have been imported correctly, use menu item Plot > Channel locations > By name

1.1.1.1.1

The quality of the data is critical for obtaining a good ICA decomposition. ICA can separate out certain types of artifacts -- only those associated with fixed scalp-amp projections. These include eye movements and eye blinks, temporal muscle activity and line noise. ICA may not be used to efficiently reject other types of artifacts -- those associated with a series of one-of-a-kind scalp maps. For example, if the subject were to scratch their EEG cap for several seconds, the result would be a long series of slightly different scalp maps associated with the channel and wire movements, etc. Therefore, such types of "non-stereotyped" or "paroxysmal" noise need to be removed by the user before performing ICA decomposition.

To reject noisy channels of either continuous or epoched data, select menu item Edit > select data.

To reject noisy portions of continuous data, select menu item Tools > Reject continuous data, then mark noisy portions of continuous data for reject by dragging the mouse horizontally with the left button held down. Press Reject when done. A new window pops up to ask for a name for the new dataset.

To reject noisy data epochs, select menu item Tools > Reject data epochs > Reject by inspection. Check the second checkbox to reject data at once (instead of simply marking epochs for rejection) and press OK. Then, in the scrolling window, click on data epochs you wish to reject. If you change your mind about a data epoch marked for rejection, click it again to un-select it. Press Reject when done. A new window pops up to ask for a name for the new dataset.

EEGLAB also has facilities to automatically suggest data channels, portions and/or epochs to reject. See menu item Tools > Reject data epochs > Reject data (all methods). See the Data rejection tutorial for more details.

1.1.1.1.2

Use menu Tools > Run ICA to run the ICA algorithm. To accept the default options, press OK.

Use menu Tools > Reject data using ICA > reject component by maps to select artifactual components. See the Data analysis (running ICA) tutorial for more details.
Select menu item Tools > Remove components to actually remove the selected component from the data.

See the Data analysis (running ICA) tutorial for more details and some hints on how to select artifactual components.

Your data has now hopefully been pruned of its major artifactual components. You may now proceed with further EEGLAB processing of the remaining non-artifactual independent components (see Data analysis (working with ICA components)).

You may also export your data by selecting menu item File > Export > Data and ICA activity to text file (EEGLAB v4.1). (Note: We believe Neuroscan versions 4.1 and higher can import data files in text format.)
2 A13: Compiled EEGLAB

EEGLAB now exist as a compiled binary for Mac, Windows and Ubuntu and may be downloaded on the EEGLAB download page.

2.1

2.2

- Download the ZIP file for the EEGLAB compiled version on the download page and uncompress it
- Each zip file contains 2 folders "for_redistribution" and "for_redistribution_files_only." Start by starting the installer in "for_redistribution" folder. If all goes well, this is the only thing you have to do to install EEGLAB.
- If for some reason the installation process fails (permission issue for example), you may use the files in the folder "for_redistribution_files_only." Note that these files requires that the Matlab runtime engine be installed on your system - this step should have been done automatically above but we will assume here that this step fails. These are available as separate download here (the version needed should appear when you start EEGLAB).
- Notes for MacOS and Linux users: Apparently the compiled version of EEGLAB need to be installed using root user credentials. If you experience problems during the installation process on Mac, after downloading the zip file, open a terminal, go to the folder for_redistribution/MyAppInstaller_web.app/Contents/MacOS and type in sudo ./setup. This will start the installation as root. Once installed, you can run EEGLAB under your non-root account.

2.3

- Click "EEGLAB" to start EEGLAB (however see below)
- Create shortcuts if necessary on the Desktop and in the "Start" menu
- It is often preferable to have the command line output of EEGLAB as it contains relevant information. On windows, start EEGLAB using the file "eeglab_run_this_one_on_windows.bat." On OSx and Ubuntu, use the file run_EEGLAB.sh (might require to have it permission changed to be executable otherwise the content of the script will be shown).

2.4

- Both version graphical interface are identical
- There is nothing in the graphic interface that you can do under Matlab that is not possible to do in the compiled version. This includes using all the plugins and all the external modules attached to EEGLAB, saving scripts and running Matlab scripts.

2.5

- It is not possible to add new plugins or download and install third party plugins. To do so EEGLAB needs to be recompiled with the additional plugins.
- When using scripts, is not possible to use external custom Matlab functions or to define new functions
- When using scripts, the range of possible Matlab command usable is limited to the Matlab core commands, all EEGLAB commands and a few commands from the signal processing, the statistics and the optimization toolbox (commands which are included in the compiled code).
• It is not possible to modify EEGLAB source code.

2.6

Compiling EEGLAB usually consist in opening the "eeglab.prj" file (in the root EEGLAB folder) using the Matlab Compiler Graphical App, then pressing the package button. However, there might be some path issue that require fixing on your system. This page contains notes we use to compile EEGLAB in our lab (it is relatively complex because we have automated nightly build for 3 operating systems).

2.7

Differences between the Matlab and the compiled EEGLAB version mostly arise because of the way external data files are handled. This is how we check the integrity of the compiled version (since it is not possible to automate this process yet).

1. Start EEGLAB
2. Edit options and change them; go back to option and make sure they have changed
3. Load continuous dataset. Look up electrodes, run clean raw data. Run ICA, run IClabel
4. Perform source localization of ICA components using Dipfit
5. Plot 3-D ERP after coregistering
7. Check that the help menus are functional

If all of the above checks, then the compiled EEGLAB version is considered checked. We check the version separately on Windows and Unix/MacOS (Unix and MacOS are not checked independently since MacOS is a version of Unix).

2.8

• Can I run some of my own Matlab scripts using the EEGLAB compiled version? Yes as long as they use standard Matlab functions or EEGLAB functions.
• Is the compiled code faster than the non compiled one? No, it is the same speed as it is still interpreted by the Matlab runtime engine.
• What version of Matlab was used to compile the code? As of 2019, we provide a compiled version for every released EEGLAB version.
• Is it legal since I do not own Matlab? Yes, it is perfectly legal. Although it is illegal to run cracked version of Matlab, it is perfectly legal to distribute the Matlab Runtime Engine and compiled Matlab code.
• Can I use this program for commercial applications such as my clinical private practice? EEGLAB is a research software not a clinical software. The program is provided as is with no warrantee of any kind. Although the EEGLAB core code is released under a BSD 2.0 license that allows for commercial use, some of the plugins included in the compiled version are not (this includes plugins handling source reconstruction for example).
• Trouble shooting! Do not hesitate to submit a bug or comment on the EEGLAB Gihub issue tracker if you encounter a problem running a compiled version of EEGLAB.
3 Binica

3.1

This binary version of the runica() function of Makeig et al. contained in the EEG/ICA Toolbox runs ~12x faster than the Matlab version and may be ~4x more compact. It uses the logistic infomax ICA algorithm of Bell and Sejnowski, with natural gradient and extended ICA extensions. It was programmed for unsupervised operation by Scott Makeig at CNL, Salk Institute, La Jolla CA. Sigurd Enghoff translated it into C++ code and compiled it for multiple platforms. J-R Duann has improved the PCA dimension-reduction and has compiled the linux and free_bsd versions.

Use: To use the function, call with a ".sc" file argument. For individual applications, copy and modify the sample script "ica.sc".m. Under UNIX or DOS command line

% ica < myversion.sc

Be sure that the directory that you store the binary file in is in your search path. e.g., In Unix/Linux add this directory to your root .cshrc file ”setenv path” line.

Outputs: Ica creates two files, "xxx.wts" and "xxx.sph" containing weights and sphere matrices such that (under Matlab)

>> ICA_activations = wts * sph * data;

The "xxx" stem in the output files may be specified within the input .sc parameter file. See the sample .sc file for arguments, and the EEG/ICA toolbox tutorial for more details.

◊ Linux version (works under Red Hat 7 or 8). See also OSX compiled version with GCC below.
◊ FreeBSD 4.0
◊ Mac OSX PPC (Recopiled by William Beaudot in 2004) or Recent compile (April 2016)
◊ Mac OSX Intel (Recopiled by Grega Repovš in 2009)
◊ Win x64 Intel, Windows: (Recompiled by Ernest Pedapati and Ellen Russo) the source code should compile on windows architectures (i.e. 32 bit/ AMD).

To use one of these programs from within Matlab (and EEGLAB)

1. download the file and place them in the functions/support_files of the eeglab directory (you may create a subfolder for them or uncompress them in the function subfolder).
2. edit the icadefs.m file to specify the file name of the executable you intend to use (search for variable ICABINARY).
3. From the command line, you may use the binica.m function that will call the binary executable. From the EEGLAB graphical interface, run ICA using the 'binica' option of the Tools > Run ICA graphic interface (see the tutorial for how to compute ICA components). As a test, try for example “binica(rand(10,1000))”. This is how to use the algorithm on an EEGLAB dataset.

EEG = pop_runicaEEG,, ,,,;

Older version of the ICA binary are available below (these versions are not compatible with the Matlab binica.m function and cannot be used directly in Matlab or from EEGLAB (although see bug 1604): The ica.sc text configuration file (sample here) in the archive must be edited manually). SGI Unix (older version) Sun Unix (older version) Windows PC (95, 98, NT, 2000, XP?. older version)

3.2

The Github repository (13Mb) contains the source code, and binica_full.zip (~180Mb) source code plus many binaries. This code is distributed under the GNU GPL license and may not be used for commercial applications. It is copyrighted by the Salk Institute for biological studies and the University of San Diego California. This code can usually compile under most Unix machines. The binary above for Mac OSX also contains a make file for Mac OSX. Some recommendation below:

1. if you uncompress using winzip, deactivate the "tar smart CR/LF“ option in winzip in the menu Option > configuration tab Miscellaneous
2. recompile BLAS (folder CLABPACK\BLAS)
3. recompile LABPACK (CLABPACK folder)
   For 2 and 3 it is actually better if you find on the Internet the latest versions of these libraries
4. make the ICA binary file by using the makefile in the main directory
5. Modify the icadefs.m Matlab file under EEGLAB so that it points to your binary (in case you want to call it from Matlab).

The README file embedded in the compressed archive also contains additional details. We would advise to use the GNU C compiler since the Makefiles should be compatible with it. Please when you have succeeded compiling it, send us a copy of the exe file (and a small report of how you did it) so we can put the new file on the Internet.

Important: Infomax ICA is under a patent by the Salk Institute and any commercial product using this type of algorithm (or the recompiled binary files distributed here) should contact the Salk Institute patent office.

Code provided for convenience: This code is not supported and this page is provided for convenience only. If you want to add more documentation (readme file etc...) and guidelines for users or want to set up a project under sourceforge to support development and maintenance of BINICA, please feel free to do so. We will not answer questions regarding the compilation of this code. - Arnaud Delorme, November 21, 2007.
4 Channel Location Files

4.1

• Access Sample files sent to us in the following formats:

EEGLAB ".loc" and ".txt",
Polhemus ".elp",
EGI ".spl" and ".xyz",
Besa ".elp",
Neuroscan ".map"

• EEGLAB can read all these file formats (except for Neuroscan ".map"). Share your location files by adding files directly to the wiki HERE. Please do not forget to add the link to your uploaded files below or in the Talk:Channel Location Files page.

• Neuroscan company web page with 32- and 64-channel headmodel files (see also dave.zip).

• EGI/Philips Neuro channel location files with BESA-compatible electrode location.

• Standard electrode locations

• Easy-cap electrode locations in 3-D spherical, 3-D cartesian, and 3-D spherical equidistant coordinates.

We have created software (in EEGLAB) to transform between spherical, 3-D cartesian and polar coordinates in a variety of file formats (see readlocs() function help). We will post more information here soon...

4.1.1

• Upload Files by clicking 'Upload file' on the sidebar to the left and then add the link below. These files should be zipped as wikimedia only allow downloading a limited number of file types.

• Add a link to the files you have uploaded in the list above.
Image at top of page: Independent component of 252-channel EEG data with bilateral occipital scalp projection. Derived by independent component analysis of approximately 700,000 points of data using runica() in the EEGLAB Toolbox, implementing extended infomax ICA. Here, the 252-channel data were reduced to 160 independent components by PCA prior to ICA training. Channels located below the head center are shown in this topoplot() as extending out from the model head borders. Note that ICA finds the gradient of the projection from the source to the skin surface not only at the scalp electrodes but also for channels on the neck, forehead and temples. Residual variance of the bet-fiting symmetric dual-dipole model in a model sphere head was 2.0%. Dipole calculations used the DIPFIT plug in.
5 Downloading and contributing to EEGLAB from GitHub

1. REDIRECT How_to_download_EEGLAB

This page describes how to download, access and contribute to the development version of EEGLAB from GitHub.

5.1

5.1.1

The GIT repository of EEGLAB in GitHub is comprised by several branches that contain almost ALL the previous EEGLAB releases as well as the current and future release versions. Accessing or exploring the branches is possible from the main page of EEGLAB GitHub repository. This version in the develop branch contains the latest developments in the code and is recommended for EEG advanced users.

5.1.2

EEGLAB download in ZIP format is available at EEGLAB download link and from the This includes the latest release as well as old versions.

5.1.3

5.1.4

5.2

5.2.1

5.2.2

5.3

EEGLAB was first under RCS (2002-2005), then under CVS (2005-2010), and finally under SVN (2010-2014) before migrating to GIT (2014-). All the revision message have been preserved in the migration process. It is however not possible to access the RCS, CVS, and SVN repositories since they refer to obsolete versions of EEGLAB.
EEGLAB is now under CVS (Concurrent Version System).

If you want to edit files at SCCN, you first need to set your environment variable CVSROOT (in your .cshrc file for instance)

```bash
setenv CVSROOT /data/common/cvs/
```

Then the simplest way to edit files is to use the Matlab editor and set the Matlab preference to CVS for revision control. As with RCS, you may check out any EEGLAB file, and then check it back in and enter your revision comment. It is that simple. The difference with our prior revision scheme however is that **IF YOU DO NOT CHECK BACK IN YOUR CHANGES, THEY WILL BE ERASED OVERNIGHT**. Your changes will be saved in temporary files though.

Files are erased overnight because the EEGLAB files at SCCN are just a checked out version of the real EEGLAB repository (which contains all history of changes and some fragmented Matlab code but not the actual Matlab files). There can be as many checked out version as we want (one in France, one in India, etc...) and all of them can be edited simultaneously. If two people modify the same file, and check them back in, then there is a conflict and it has to be resolved manually using special tools that allow to compare the files side by side with color highlighting of the changes.

So, if you want to work on a couple of functions for several days, the best is to check out the whole tree structure of the plugin in your home folder (using dedicated software or Linux command) and then check back in your changes when you are done.

Arno Delorme - Jan 30th 2009
7 EEGLAB and high performance computing

7.1

The documentation for NSG has been moved to https://github.com/sccn/nsgportal/wiki.
8

8.1

GPU-based processing is promising in MATLAB. There are three options on the market: GPUmat, Jacket, and the Parallel Computing Toolbox (see the Conclusions below for more information).

As of mid-2010 -- we have only tested GPUmat. We tried testing the MATLAB Parallel Computing Toolbox but it did not support Matrix indexing so we could not really test it.

8.1.1

The recent enthusiasm for using GPU (Graphical Processing Unit) computational capabilities led us to try the freely available GPUmat (Please correct this link as it does not point to the correct website) Matlab toolbox that runs data processing under GPUs (a GPU must be present on the machine you are using to test it). Computing using the GPUmat toolbox usually only involves recasting variables and requires minor changes to Matlab scripts or functions.

GPU processing can lead to speed-up of more than 100x for optimized applications. We were interested in testing the use of our Nvidia GPUs for running time-frequency decompositions and non-parametric surrogate statistics. Below we report first tests of using GPUmat for these two EEGLAB signal processing functionalities.

One of our servers has two Intel Xeon W5580 CPUs (total of 16 cores), 72 GB RAM, and one NVidia Tesla C2060 GPU boards [1]. When using the main 16 cores and running Matlab on one of them, it is likely that Matlab will automatically parallelize its computations on some of the other cores. We attempted to use the function maxNumCompThreads to set the maximum number of threads to 1 but it did not alter computation time. It is unclear if Matlab 7.10 supports the maxNumCompThreads function as a warning message indicates that it will be removed in future releases. Still, we thought we should be able to see a difference between using the main processor and the GPU board (with 112 cores).

This is our system configuration as returned by the GPUmat toolbox:

- Running on -> "glnxa64"
- Matlab ver. -> "7.10.0.499 (R2010a)"
- GPUmat version -> 0.251
- GPUmat build -> 03-May-2010
- GPUmat architecture -> "glnxa64"

GPUmat used a GPU board with 112 cores. When typing "GPUstart" on the Matlab command line, the following message appears

Starting GPU
There  device supporting CUDA
CUDA Driver :
CUDA Runtime :
Device : "Tesla C1060"
CUDA Capability Major revision number:
CUDA Capability Minor revision number:
Total amount of memory:  bytes
Number of multiprocessors:
Number of cores:
  - CUDA compute capability
  - Loading module EXAMPLES_CODEOPT
  - Loading module EXAMPLES_NUMERICS
    -> numerics13.
  - Loading module NUMERICS
    -> numerics13.

8.1.2

GPUstart;
EEG = pop_loadset;
data = GPUsingleEEG::;
data = data data data data data data ;
data = data data data data data ;
data2 = data;
Raising each value in the EEG data matrix to a fractional power (1.3) using the GPU rather than the central processor produced a 20x speed increase.

### 8.1.3

We modified the repeated-measures ANOVA function to be GPUmat compatible. (All the Matlab GPU functions we used are made available at the bottom of this page).

```matlab
; tmp = data.^; GPUsync;
Elapsed time  seconds.
; tmp = data2.^;
Elapsed time  seconds.
```

The `anova2_cell()` function is highly optimized (no loops!), and the GPU computation appeared to be about 66% faster than when using the main CPUs. This relatively minor speed up seems to be because the GPU functions are slow at accessing sub-indices in very large matrices. For smaller matrices, the difference between the GPU code reached about 100% speed up.

### 8.1.4

```matlab
EEG = pop_loadset;
data2 = EEG.;
; timefreqdata2, data,, EEG., data,/EEG., EEG., , ;
Elapsed time  seconds.
= GPUsingle,, GPUsingle,,;
; FC FR FI dfc dfr dfi = anova2_cellc; GPUsync;
Elapsed time  seconds.
```

Here we did observe a (2.6x) speed-up from performing the time-frequency wavelet decompositions on the GPU rather than the CPU.

### 8.1.5

*Arnaud Delorme - August 28, 2010*

There are currently 3 options on the market: GPUmat (free), Jacket (commercial), and the Parallel Computing Toolbox of MATLAB (commercial). Overall, we were relatively disappointed with current GPU solutions. Even if Jacket proves more efficient than other options, we can only expect an additional speed-up of about 5-20% compared to GPUmat. This is far from the 100x speed-up that we were hearing about when GPU cards came out. However, a 3x speed up is still welcome. It would also be nice to have a GPU profiler to be able to see which commands are slow on the GPU so as to try to avoid them.

**GPUmat:** There is still some ways to go before the GPUmat toolbox can take full advantage of GPU processing capabilities. Based on feedback from other users, it seems that the way a function is programmed for GPUs dramatically influences its processing speed. Different CUDA language implementations may give speed-up differences of up to 1000x. It seems that the GPUmat version we tried leaves some room for improvement, in particular when accessing sub-indices of large matrices. Another bit of bad news for us was that for large data matrices (larger than 100MB) our GPU functions crashed. The real advantage of using GPU processing with EEGLAB would be to be able to process very large matrices (up to several Gb), to compute statistics and time-frequency decompositions across multiple data component or channel signals, and to bring other compute-intensive processing within a user's 'compute horizon' (the time the user is willing to wait for results...). Therefore currently, performing GPU-based EEGLAB processing via GPUmat remains of limited but possible interest.

**PCT:** Also, Matlab 2010b supports GPU (if you have a Matlab Parallel Toolbox license). It only offers primitive functionality, for instance you can't even do any indexing into matrices with the PCT solution so we were not able to test our functions. [January, 2019 - This question should be revisited!]
Jacket: The following is a quote from a Jacket representative. "Jacket has the fastest performance (see versus GPUmat here and see versus R2010B here) and broadest function support. It has also been used by many neuroscientists and is currently being leveraged by the SPM crowd. Jacket may be downloaded a free 15-day trial from the AccelerEyes website. Jacket costs $350 for academics. Jacket provides a GPU profiler, with GPROFVIEW". We have not tested jacket yet but we are planning to.

The EEGLAB-compatible GPUmat-based functions we tested are available here. Note that these functions are not totally functional (they only work under a limited set of conditions as tested above) and thus are only made available for exploratory testing purposes.
We have a funded project to run EEGLAB on the Neuroscience Gateway to run EEGLAB jobs on the San Diego supercomputer. This is a free service and anybody in the world can use it. See this page for more information.

In the short term, Octave is the shortest way to using EEGLAB functions and actually obtain useful results. This page describes how to use EEGLAB on Octave.

9.1

When it comes to using supercomputers, Matlab, although quite efficient, may become incredibly expensive. A single Matlab license may cost $2,100 ($1,050 for academia), and with all its commercial toolboxes might come to $145,000 or more. If you have a supercomputer with about 100 processors (as of 2011, this amounts to about $30,000 or 20,000 euros), you might need to pay the Mathworks about $30,000 to $500,000 to be able to run Matlab on it (the exact price depends on the number of users on the cluster, the number of nodes, and the extra toolboxes). This may be much more than the price of the supercomputer itself! Given that the Matlab core has not evolved dramatically over the past 10 years, and still has flaws (lack of consistency of the graphic interface between platforms; numerical inconsistencies in early version of Matlab 7.0), free alternatives to Matlab are needed in the Open Source community to run computation on supercomputers.

We have attempted to tackle this problem and as of June 2018 (EEGLAB 15+), we are currently supporting Octave (v4.4.0) for supercomputing applications (command line calls only, no graphic support). In our tests, Octave is about 50% slower than Matlab but this can easily be compensated by increasing the number of processors assigned to a specific processing task. Note that EEGLAB functions have not been parallelized (except a few rare exceptions). Therefore, you are required to open a Octave/Matlab session on each node and run custom scripts you write to take advantage of your parallel processing capability. Again, this page describes how to use EEGLAB on Octave.

9.2

Hadoop Mapreduce is a framework for performing computation on large clusters of computers. There are two steps in Mapreduce job: a mapping task where a large number of workers (computers) work on a large number of data lines, and a reduce step, where (usually) a single worker pools all the mapping results.

Below we provide guidelines for using Elastic Mapreduce on the Amazon cloud. Note that Elastic Mapreduce is tailored to processing large quantities of log text files and not binary data. The gain in terms of processing speed compared to the cost of running such solution remains unclear if you have a local cluster of computers. In short, you might spend more time programming the solution and it might cost you more in terms of bandwidth and storage that if you are running it locally. These are the steps you should follow. These are new technologies so expertise in computer science is highly recommended.

- Installing Hadoop command line interface. First install the Command Line Interface to Elastic Mapreduce. This will allow you to configure and run jobs on the Amazon cloud. You will also need to create an AWS account. Hadoop will need to run in streaming mode, where the data is simply streamed to any executable. It might also be possible to run Hadoop in native Java mode and compile Matlab code using the Java builder (this is probably much more complex than using the streaming mode though).

- Transfer your data to Amazon storage cloud (the Amazon storage cloud is named S3). A useful tool to do this is the s3cp tools. Note that your data should be formatted in strings of characters. If you want to process raw EEG data, you will have to serialize it in text, with each channel for example representing one line. There is no limit to the length of a line of text. However, one must remember the overhead in terms of both signal processing and bandwidth associated with processing text. If you have 128 channels and 100 data files, this corresponds to 12800 processing hadoop steps. If you can allocate 1000 workers to the task, this means that each worker will process about 13 channels, a potential speedup of about 1000 on your task. To minimize bandwidth overhead, you might want to transfer the compressed binary data to S3, then have a local amazon EC2 amazon node uncompress it and put it back to S3 (this is because EC2 nodes bandwidth with S3 is free). If you are dealing with Terabytes of data, this task can take a long time (as S3 is configured to have a very slow reading latency and very high writing latency). There are tools to copy data in parallel to S3.

- Solution 1 (easiest to implement) using Octave. EEGLAB command line code is compatible with Octave. Octave may be installed relatively easy on each of the nodes using the bootstraping method (a method to automatically install software on each of the nodes). The command to automatically install Octave on EC2 Amazon nodes is:

```
sudo yum ?y install octave --enablerepo=epel
```
Then, for your main Matlab script, you might want to add the following at the beginning of the main script. This will make it executable and will allow it to process data on STDIN.

```
#!/usr/bin/octave -qf
Q = stdin;
```

Hadoop communicate with workers through STDIN and STDOUT pipes. You may write the output of your data processing using the printf or disp Matlab commands.

- **Solution 2**, compiling Matlab code. Compiling Matlab code is the most efficient solution as Matlab compiled code is often 2 to 4 times faster than Octave code and compiled code does not require a Matlab licence. If you compile Matlab code on your local Unix workstation, you will need to make sure to use an Amazon AMI (virtual machine image) with the same set of libraries so that your code can run on that machine. You will need to pick an AMI that is compatible with Hadoop as well. Also, Matlab does not have a simple mechanism allowing it to read from STDIN. The easiest solution is to use third party compiled Mex files to do so (see for example `popen`). Another solution is to have a shell command write STDIN on disk, then call the Matlab executable (although this might impair performance).

- **Reduce step**: once all the worker have computed what they had to compute (spectral power for example), the reduce step may write it back on S3 Amazon storage (and also do further processing if necessary such as grouping back channels belonging to the same subject).

- **Running Hadoop**: using the AWS command line interface, type something like the following.

```
elastic-mapreduce --create --stream -- s3n://Arno/myEEGserializedtextfiles/ \
--mapper s3://Arno/process_octave \
--reducer s3://Arno/reducer. \
--output s3n://Arno/output --debug --verbose \
--log-uri s3n://Arno/logs --enable-debugging \
--bootstrap-action s3n://Arno/install_octave
```

Note the reduce step can be written in any programming language that takes data from STDIN and writes to STDOUT. The reduce step will usually not require to run EEGLAB commands. It is simply about pooling data from the workers and summarizing it. In this case, we used Python custom program (reducer.py) but it could have also been Octave/Matlab since Octave is installed on each of the workers. The exact content of your code will depend on what task you are interested in doing.

The solution outlined above should only be tried when dealing with gigantic amount of data that no local processor or cluster can handle. It is costly (mostly in terms of Amazon storage as storing 10 Terabytes of data will cost you about $800 per month as of 2013). It is therefore best suited when bootstrapping data is required (lots of computation on little data). Send us your comments at eeglab@sccn.ucsd.edu.

---

Return to EEGLAB Wiki Home
10 EEGLAB Bugs

Some information here may be outdated or fixed with newer versions of EEGLAB. Please examine your current version documentation to check.

10.1

EEGLAB bugs are managed under Github EEGLAB Issues. The old EEGLAB Bugzilla interface has been deprecated although it is still being used to track potential improvements.

10.1.1

- Since EEGLAB has been developed under Matlab, there is little risk that using EEGLAB will crash your machine or erase files inadvertently, unless Matlab itself crashes. This is one advantage of using Matlab.

- Most of the core EEGLAB functions have been or are being used in our published manuscripts, which are available for download from the SCCN website. Consult these papers, plus the extensive EEGLAB tutorial and help facilities, for instructions and examples of their use.

- About 1000 test cases run daily on EEGLAB code to test its integrity and that its functions are stable. See EEGLAB test cases

- EEGLAB is an open source project. To understand in more detail how any signal processing is performed, you may simply study the function source file. To adjust its performance, you may simply edit it yourself. Note: If you do this successfully, please consider issuing a pull request on Github. See more information on contributing to EEGLAB.

- If you encounter a bug, please first read carefully the Matlab commandline and any error window text, to determine whether you may be able to avoid the problem directly. Next, test whether the error occurs using current release of EEGLAB. If so, check the Github EEGLAB Issues to see if your issue has already been reported.
  - ♦ If your issue has been reported, you may comment on the bug.
  - ♦ If your issue has not been reported, you may submit a new bug. Once you press the "New issue" button, you will be guided on how on the type of information needed to report your bug.

10.1.2

- **Epoch selection using pop_eegplot():** Epochs selected for rejection using eegplot() data scrolling are not saved in EEGLAB history until the epochs are actually rejected. This means that they will not be reproduced automatically in a new EEGLAB session. However, the labeled epochs are identified in the field EEG.reject.manualrej that is saved along with the dataset. Also, and more importantly, when the labeled epochs are actually rejected, this operation is saved in EEGLAB history.

- **Zooming using pop_eegplot():** When zooming and selecting epochs, only the data may be zoomed.

- **Spectral analysis (with no Matlab Signal Processing Toolbox):** The spec() function emulates the function psd() but not the function pwelch() (psd() was replaced by pwelch(), beginning with EEGLAB 4.3, for technical reasons). As a result, the scaling of the spectrum (by the spectopo() function only) is slightly different than when the Matlab Signal Processing Toolbox is present. Also, for unknown reasons, the spec() function cannot handle frequencies that have been filtered out and may return inaccurately high power over these frequency regions.

- **Matlab versions and OS:** Matlab versions have different bugs under different OS and these bugs - usually graphical bugs - may affect EEGLAB. The latest one we know of is the fact that Matlab version 2018a (all OS) requires a patch for EEGLAB to work.
11 EEGLAB Extensions details

This page is deprecated. The main page for EEGLAB extension is this one.

EEGLAB extensions (called ‘plug-ins’ until v13) allow users to build and publish new data processing and/or visualization functions using EEGLAB data structures and conventions. Extension functions can be easily used and tested by selecting the new menu items they introduce into the EEGLAB menus of users who download them. As of EEGLAB v13, extensions are installed by the EEGLAB Extension Manager (see EEGLAB menu tab File > Manage EEGLAB Extensions).

11.1

These extensions allow to import various type of data. Although EEGLAB contains native function to import some data formats, these functions support other formats.

- **BIOSIG data import**: Import/export data in a wide variety of data formats, developed by Alois Schloegl, the creator of the EDF+ data format. For more information about BIOSIG toolbox from [this page](http://biosig.sourceforge.net).
  - FileIO: toolbox allowing data import in multiple data formats.
- **CTF data import**: Import CTF MEG data. Available from Darren Weber's [EEG sourceforge](http://sourceforge.net) project, this extension imports MEG data (plus concurrent EEG, if any) plus sensor locations and data events from data in the CTF (Vancouver, CA) data format.
- **ANT data import (v1.03)**: Import data files in the EEP format. Contributed by [ANT Software](http://www.ant-software.nl) (Netherlands) to import data in their format. Email contact: info@ant-software.nl.
- **BVA data import/export**: Import/export files from/to the Brain Vision Software Analyser suite. Contributed by Andreas Widmann of the University of Leipzig (Germany) with Arnaud Delorme.
- **Neuroimaging 4D**: Christian Wienbruch of the University of Konstanz (Germany) has an extension available for loading Neuroimaging 4-D data into EEGLAB.
- **TDT data import**: Adam Wilson at the NITRO Lab at the University of Wisconsin Madison (USA) offers an extension available for loading Tucker-Davis Technology format data into EEGLAB.
- **NeurOne data import**: EEGLAB extension for reading the file format of NeurOne system.

11.1.1

Many other EEGLAB extensions are available for EEGLAB. The list below is not complete, and the methods they make available may have not been assessed by the EEGLAB developers. (We recommend that EEG researchers thoroughly study and consider the basis of any methods they apply to experimental data). To allow us to add new extensions or information to the list below, send us an email at eeglab@sccn.ucsd.edu:

- **DIPFIT2**: Dipole modeling of independent data components using a spherical or boundary element head model. Uses functions from the FIELDTRIP toolbox of Robert Oostenveld at the Donders Center, University of Nijmegen. A [DIPFIT2 tutorial](http://fieldtrip.nim达到) is available.
- **IIRfilt**: Infinite impulse response filtering: Apply short non-linear filters to EEGLAB data. Contributed by Maksyn Pozdin.
- **FMRIIB**: Remove FMRI-environment artifacts from EEGLAB data. This extension, by Rami Niazy of Cardiff University (Wales, UK), allows removal of scanner-related artifacts from EEG data collected during fMRI scanning. These tools provide a gui for removing FMRI gradient artifacts, detecting QRS complexes from an ECG channel, and removing pulse-related ballistocardiographic (BCG) artifacts from the EEG data. All of the tools can also be used from the Matlab command line, allowing expert users to use them in custom scripts.
- **LORETA**: Import/export command line bridge function between EEGLAB and this well-known ‘low-resolution’ EEG source imaging approach by R.D. Pascual-Marqui. Contributed by Arnaud Delorme.
- **CLUSTSET**: Cluster ICs of a single dataset by their residual mutual information. See tutorial [here](http://www.sccn.ucsd.edu). Contributed by Nima Bigdely Shamlo of SCCN (UCSD, La Jolla)
• AAR (Automatic Artifact Removal toolbox): This toolbox (web page here), implemented as an EEGLAB extension, aims to integrate several state-of-the-art methods for automatic removal of ocular and muscular artifacts in the electroencephalogram (EEG). Contact is German Gomez Herrero (Tampere, Finland) for details.

• ADJUST: A completely automatic algorithm that identifies artifact-related Independent Components by combining stereotyped artifact-specific spatial and temporal features. Features are optimized to capture blinks, eye movements and generic discontinuities. Once artifacte-related ICs are identified, they can be simply removed from the data while leaving the activity due to neural sources almost unaffected. Download the extension and tutorial here. Contact mail: ADJUST staff. Contributed by Andrea Mognon and Marco Buiatti.

• batch_context: The batch_context extension provides an interface for generating data processing pipelines and executing them on multiple EEGLAB data files either locally or on remote compute clusters. The main development source is located at [1]. Email James here.

• BCILAB: An extensive toolbox by Christian Kothe for building and running online brain-computer interface (BCI) models for a wide variety of purposes (volitional control, cognitive monitoring, neurofeedback, etc.). Extensive documentation and code are available here, and a series of over 60 short video lectures here.

• BERGEN: Removal of fMRI-related gradient artifacts from simultaneous EEG-fMRI data. The BERGEN extension for EEGLAB provides a GUI with different methods for gradient artifact correction. Contributed by Matthias Moosmann and Emanuel Neto.

• CIAC (cochlear implant artifact correction): is a semi-automatic ICA-based tool for the correction of electrical artifacts originating from cochlear implants. A validation paper describing CIAC in detail has been published in Hearing Research. More info and download.

• CORRMAP: Semi-automatic identification of common EEG artifacts based in a template. The CORRMAP extension consists of a set of Matlab functions allowing the identification and clustering of independent components representing common EEG artifacts (eye blinks, other ocular artifacts and heartbeat artifacts) in a large number of datasets (requires STUDY structure). Contributed by Filipa Campos Viola. Download extension and tutorial available here.

• ERPLAB: The ERPLAB Toolbox is a set of open source Matlab routines for analyzing ERP data that operate as a set of extensions to EEGLAB. The development of ERPLAB Toolbox is being coordinated by Steve Luck and Javier Lopez-Calderon at UC Davis.

• EYE-EEG: The EYE-EEG Toolbox is an extension for EEGLAB developed by Olaf Dimigen & Ulrich Reinacher in Werner Sommer’s Biological Psychology lab at Humboldt University Berlin with the goal of facilitating integrated analyses of electrophysiological and oculomotor data. The extension parses, imports, and synchronizes simultaneously recorded eye tracking data and adds it as extra channels to the EEG. Saccades and fixations can be imported from the eye tracking raw data or detected with a velocity-based algorithm. Eye movements are added as new time-locking events to the existing EEGLAB event structure, allowing easy saccade- and fixation-related EEG analyses in the time and frequency domains (e.g., fixation-related potentials, FRPs). Alternatively, EEG data can be aligned to stimulus onsets and analyzed according to oculomotor behavior (e.g. pupil size, microsaccades) in a given trial. Saccade-related ICA components can be objectively identified based on their covariance with the electrically independent eye tracker.

• FASTER: implements a fully automated, unsupervised method for processing of high density EEG data. FASTER can be used to process EEGLAB datasets, .set and .bdf files. Includes common features such as data importing, epoching, re-referencing, and grand average creation, as well as automated channel, epoch and artifact rejection based on ICA. FASTER has been peer-reviewed, it is free and the software is open source. If you use FASTER, please reference: Nolan, H., Whelan, R., & Reilly, R.B. Journal of Neuroscience Methods, 192, 152-162, which can be obtained here. Download FASTER here. Contributed by Hugh Nolan and Robert Whelan.

• FIRfilt: Apply a variety of linear filters to EEGLAB data. Contributed by Andreas Widmann (Leipzig, Germany). Latest version updates are available here. For more information about this extension, check firfilt FAQ.

• Grandaverage: Perform grand averaging across specified EEGLAB datasets. Contributed by Andreas Widmann of the University of Leipzig (Germany). Download here.

• LIINC extensions: Cogniscan data import, Linear Discrimination, Generalized Eigenvalue decomposition, Common Spatial Patterns, Peak Fitting, Eye Movement Removal: Paul Sajda and colleagues at the LIINC Lab at Columbia University (New York City) distribute several extensions for use in single-trial response detection.
A reference article has been published here. The download link is here.

• **MARA**: Automatic identification of artifactual independent components contributed by Irene Winkler and colleagues. MARA is a linear classifier that learns from expert ratings by extracting six features from the spatial, the spectral and the temporal domain. Features were optimized to solve the binary classification problem "reject vs. accept", and should be able to handle eye artifacts, muscular artifacts and loose electrodes equally well. Download the extension and tutorial here.

• **Mass Univariate ERP Toolbox**: is a freely available set of MATLAB functions by David Groppe and colleagues for performing mass univariate analyses of event-related brain potentials (ERPs), a noninvasive measure of neural activity popular in cognitive neuroscience. A mass univariate analysis is the analysis of a massive number of simultaneously measured dependent variables via the performance of univariate hypothesis tests (e.g., t-tests). Savvy corrections for multiple comparisons are applied to make spurious findings unlikely while still retaining a useful degree of statistical power. This approach is popular in the fMRI community but has not been commonly used by ERP researchers. Compatible with EEGLAB and ERPLAB. Documentation and downloads here. See also David's lecture on multiple comparisons in the Online EEGLAB Workshop.

• **MPT**: A toolbox for Measure Projection Analysis developed by Nima Bigdely-Shamlo at SCCN/UCSD for projecting EEG measures tagged by source location into a common template brain space, testing local spatial measure consistency, and parsing measure-consistent brain areas into measure-separable domains. Attractive 3-D graphics and some support for condition and group statistics are provided. A paper is available.

• **NFT**: The Neuroelectromagnetic Forward Head Modeling Toolbox, an elaborate extension by Zeynep Akalin Acar, builds custom Boundary Element Method (BEM) and Finite Element Model (FEM) forward head models from subject MR head images and/or from an MNI template brain model warps to measured electrode positions. Web documentation and a reference paper are available here.

• **PACT**: is an EEGLAB extension for computing cross-frequency phase-amplitude coupling developed by Makoto Miyakoshi at SCCN/UCSD, with with documentation here

• **REGICA**: An extension by Manousos A. Klados of Aristotle University of Thessaloniki, Greece to remove EOG artifacts by regression performed on ICA components. A semi-simulated dataset that might be used in any artifact rejection study is also available. A paper on the method is here. Email Manousos Klados here.

• **SIFT**: The Source Information Flow Toolbox by Tim Mullen computes a wide variety of multivariate effective causal models of source-resolved EEG data. Interactive visualizations and animations of event-related 'information flow' networks are included. Extensive documentation is available here.

• **bioelectromag**: The bioelectromagnetism Matlab toolbox is interfaced in this extension to plot average ERPs and to find their minima and maxima. Only a few files from this toolbox are included in this extension.

• **Fieldtrip**: The Fieldtrip toolbox may be used an extension to EEGLAB. Some Fieldtrip functions are used within EEGLAB for source localization (DIPFIT) and for computing STUDY statistics.

### 11.1.2

The tools below may not create new EEGLAB menus. Nevertheless they may be used with EEGLAB.

• **Svarog data format**: This web site allows importing Svarog data format. Though this is not an EEGLAB extension, once data and its parameters have been imported into Matlab, they can be imported into EEGLAB link.

• **LOC**: Performs approximate localization of electrocorticographic electrode positions from x-ray images, as documented by Kai Miller (University of Washington, Seattle) in this J. Neurosci. Methods paper. The download link is here (27.8 MB)]

• **LIINC extensions**: Bilinear Discriminant Component Analysis (BDCA) by Paul Sajda and colleagues at the LIINC Lab at Columbia University (New York City). The download link is here.

• **BESAfit**: dipole modeling using BESA3: Computes equivalent dipole locations for independent data components using BESA (old) version 3.0 (Megis Software, Germany) run external to Matlab. Download extension version 1.0 here.

• **Micromed data import**: Micromed (Italy) has an extension available for loading their data format into EEGLAB. Contact Cristiano Rizzo for details.
11.1.3

Installing and removing an extension is easy. Simply use menu item **File > Manage EEGLAB Extensions** in EEGLAB v13 and above.

Alternatively, you may download the zip file of the extension and place it in the eeglab/plugins folder. For example, to begin using an extension called 'myextension'. Uncompress the downloaded extension file in the main EEGLAB "plugins" sub-directory/sub-folder. Remove the old version of the extension if it is present in this directory. Then restart EEGLAB. During start-up, EEGLAB should print the following on the Matlab command line:

```
eeglab: adding extension "eegplugin_myextension"   % (see >> help eegplugin_myextension)
```

The extension will typically have added one or more new items to the EEGLAB menu (often under the Tools heading). To make EEGLAB ignore a downloaded extension, simply move or remove its folder from the EEGLAB 'plugins' (or main) directory and restart EEGLAB.

11.1.4

See the simple instructions under the EEGLAB wiki appendix A07: **Contributing to EEGLAB**.
12 EEGLAB Menu Functions

In EEGLAB, all menu items call stand-alone functions. The correspondence is indicated below. Note that this part of the documentation is not totally up to date. Some recently added menus might be missing.

12.1

Import data

   From ASCII/float file or Matlab array -- pop_importdata()
   From continuous or seg. EGI .RAW file -- pop_readegi()
   From Multiple seg. EGI .RAW files -- pop_readsegegi()
   From BC1200 ASCII file -- pop_loadbci()
   From Snapmaster .SMA file -- pop_snapread()
   From Neuroscan .CNT file -- pop_loadcnt()
   From Neuroscan .EEG file -- pop_loadeeg()
   From Biosemi .BDF file -- pop_biosig()
   From .EDF file -- pop_biosig()
   From ANT EEProbe .CNT file -- pop_loadeep()
   From ANT EEProbe .AVR file -- pop_loadeep_avg()
   From .BDF file (backup function) -- pop_readbdf()
   From Brain Vis. Rec. .VHDR file -- pop_loadbv.m
   From Brain Vis. Anal. Matlab file -- pop_loadbva()
   From CTF Folder (MEG) -- 'ctf_folder'
   From ERPSS .RAW or .RDF file -- pop_read_erpss()
   From INStep .ASC file -- pop_loadascinstep()
   From 4D .M4D pdf file -- pop_read4d()
   From other formats using FILEIO -- pop_fileio()
   From other formats using BIOSIG -- pop_biosig()
   From Matlab array or ASCII file -- pop_importepoch()
   From Neuroscan .DAT file -- pop_loaddat()

Import event info

   From Matlab array or ASCII file -- pop_importevent()
   From data channel -- pop_chanevent()
   From Presentation .LOG file -- pop_importpres()
   From Neuroscan .EV2 file -- pop_importev2()

Export

   Data and ICA to text file -- pop_export()
   Weight matrix to text file -- pop_expica()
   Inverse weight matrix to text file -- pop_expica()
   Data to EDF/BDF/GDF file -- pop_writeeeg()
   Write Brain Vis. exchange format file -- --

Load existing dataset -- pop_loadset()
Save current dataset -- pop_saveset()
Save datasets -- pop_saveset()
Clear dataset(s) -- pop_delset()
Create Study

   Using all loaded datasets -- pop_study()
   Browse for datasets -- pop_study()

Load existing study -- pop_loadstudy()
Save current study -- pop_savesstudy()
Save current study as -- pop_savesstudy()
Clear study -- pop_delset()
Maximize memory -- pop_editoption()

History Scripts

   Save dataset history script -- pop_saveh()
   Save session history script -- pop_saveh()
   Run Script -- -- --
   Quit
12.1.1

Dataset info -- pop_editset()
Event fields -- pop_editeventfield()
Event values -- pop_editeventvals()
About this dataset -- pop_comments()
Channel locations -- pop_chanedit()
Select data -- pop_select()
Select data using events -- pop_rmdat()
Select epochs or events -- pop_selectevent()
Copy current dataset -- pop_copyset()
Append datasets -- pop_mergeset()
Delete dataset(s) -- pop_delset()

12.1.1.1

Change sampling rate -- pop_resample()

Filter the data
  Basic FIR filter -- pop_eegfilt()
  Short IIR filter -- pop_iirfilt()
Re-reference -- pop_reref()
Interpolate electrodes -- --
Reject continuous data by eye -- pop_eegplot()
Extract epochs -- pop_epoch()
Remove baseline -- pop_rmmbase()
Run ICA -- pop_runica()
Remove components -- pop_subcomp()
Automatic channel rejection -- pop_rejchan()
Automatic epoch rejection -- pop_autorej()

Reject data epochs
  Reject data (all methods) -- pop_rejmenu()
  Reject by inspection -- pop_eegplot()
  Reject extreme values -- pop_eegthresh()
  Reject by linear trend/variance -- pop_rejtrend()
  Reject by probability -- pop_jointprob()
  Reject by kurtosis -- pop_rejkurt()
  Reject by spectra -- pop_rejspec()
  Reject marked epochs -- pop_rejepoch()

Reject using ICA
  Reject components by map -- pop_selectcomps()
  Reject data (all methods) -- pop_rejmenu()
  Reject by inspection -- pop_eegplot()
  Reject extreme values -- pop_eegthresh()
  Reject by linear trend/variance -- pop_rejtrend()
  Reject by probability -- pop_jointprob()
  Reject by kurtosis -- pop_rejkurt()
  Reject by spectra -- pop_rejspec()
  Reject marked epochs -- pop_rejepoch()

Run ICA

Locate dipoles using DIPFIT 2.x
  Head model and settings -- pop_dipfit_settings()
  Coarse fit (grid scan) -- --
  Fine fit (iterative) -- pop_dipfit_nonlinear()
  Autofit (coarse fit, fine fit & plot) -- --
  Plot component dipoles -- --

12.1.1.2

Channel locations
  By name -- topoplot()
  By number -- topoplot()
Channel data (scroll) -- pop_eegplot()
Channel spectra and maps -- pop_spectopo()
Channel properties -- pop_prop()
Channel ERP image -- pop_erpimage()
Channel ERPs
With scalp maps -- pop_timtopo()
In scalp array -- pop_plottopo()
In rect. array -- pop_plotdata()

ERP maps
As 2-D scalp maps -- pop_topoplot()
As 3-D head plots -- pop_headplot()

Sum/Compare ERPs -- pop_comperp()
Component activations (scroll) -- pop_eegplot()
Component spectra and maps -- pop_spectopo()

Component maps
As 2-D scalp maps -- pop_topoplot()
As 3-D head plots -- pop_headplot()
Component properties -- pop_prop()
Component ERP image -- pop_erimage()

Component ERPs
With component maps -- pop_envtopo()
With comp. maps (compare) -- pop_envtopo()
In rectangular array -- pop_plotdata()

Sum/Compare comp. ERPs -- pop_comperp()

Data statistics
Channel statistics -- pop_signalstat()
Component statistics -- pop_signalstat()
Event statistics -- pop_eventstat()

Time-frequency transforms
Channel time-frequency -- pop_timef()
Channel cross-coherence -- pop_crossf()
Component time-frequency -- pop_timef()
Component cross-coherence -- pop_crossf()

12.1.1.3
Edit study info -- pop_study()
Precompute channel measures -- pop_precomp()
Plot channel measures -- pop_channplot()
Precompute component measures -- pop_precomp()
Build preclustering array -- pop_preclust()
Cluster components -- pop_clust()
Edit/plot clusters -- pop_clustedit()

12.1.1.4
Current/Active Datasets (listed as selectable items) -- -- --
Select multiple datasets -- -- --
Select the study set -- -- --

12.1.1.5
About EEGLAB -- eeglab()
About EEGLAB Help -- eeg_helphelp()
EEGLAB menus -- eeg_helpmenu()

EEGLAB functions
Toolbox functions -- -- --
Signal processing functions -- eeg_helpsigproc()
Interactive pop functions -- eeg_helppop()

EEGLAB advanced
Dataset structure -- eeg_checkset()
Admin functions -- eeg_helpadmin()
Web tutorial -- -- --
Email EEGLAB -- -- --
13 EEGLAB revision history

Below are EEGLAB version as they are tagged in the SVN repository. You may check out each version by selecting the appropriate tag in the repository. The current version (HEAD) should not be different from the latest tag version. Critical updates are indicated with bug fixes in red. This correspond to updates where the processing performed was inaccurate. Updates fixing function crashes under some particular calling conditions are not indicated in red.

13.1

The HEAD revision of EEGLAB is usually not available as a .ZIP file. It is available only using the SVN revision software. About every month, a revision number is issued and a zip file released.

13.2

The HEAD EEGLAB revision (currently version 13) is the revision we are working on. It contains more bug fixes but is also at times more unstable (especially new features). We are thus also supporting version 12. In version 12, only important bugs are corrected. All the development takes place in version 13.

13.3

The first number is the current revision (12 or 13). Following numbers indicate revision of different level of importance. For instance, version 10.2.3.5 means that we are dealing with EEGLAB 10 and 2 major revisions were issued, 3 semi-important revisions, and 5 minor revisions. The second, third and fourth number vary independently. For instance, if a major revision is made to version 10.2.3.5, the revision number is going to be 10.3.3.5. However, when we change of EEGLAB version (11 to 12 or 12 to 13), we reset all subsequent sub-revision numbers (the first version 13 will be 13.0.0.0). The Mathworks uses a similar scheme for Matlab (for instance 7.11.0.584).

Starting in 2019, we are using the year of the release as the main revision number. Minor revision are indicated using a second number 2019.0 is version 2019 and first release; 2019.1 is version 2019 and second release.

13.4

EEGLAB version 2019 revision history (supported)

EEGLAB version 14 revision history (supported)

13.5

EEGLAB version 13 revision history (no longer supported) - minor bug 1971 when rejecting data, use EEGLAB 14

EEGLAB version 12 revision history (no longer supported)

EEGLAB version 11 revision history (no longer supported) - STUDY implementation backward compatible but different from EEGLAB 9 and 10

EEGLAB version 10 revision history (no longer supported)

EEGLAB version 9 revision history (no longer supported) - STUDY implementation backward compatible but different from EEGLAB 6, 7 and 8

EEGLAB version 8 revision history (no longer supported)

EEGLAB version 7 revision history (no longer supported)

EEGLAB version 6 revision history (no longer supported)

EEGLAB version 5 revision history (no longer supported)

EEGLAB version 4 revision history (no longer supported)

EEGLAB early versions prior to 2002 - see the first decade of EEGLAB
14.1

**Issue date:** September 10th, 2011
**SVN revision:** 9392
**External modules:** FILEIO SVN version 3587, BIOSIG (partial) SVN version 2694
- `std_readerp.m`, fix reading ICA component - bug introduced in version 10.2.5.4 (SVN 9391 - Arno)
- `Contents.m`, change revision number (SVN 9392 - Arno)

14.2

**Issue date:** September 8th, 2011
**SVN revision:** 9390
**External modules:** FILEIO SVN version 3587, BIOSIG (partial) SVN version 2694
- `pop_studydesign.m`, remove debug message (SVN 9385 - Arno)
- `std_readerp.m`, fix common baseline subtraction in STUDY (SVN 9383-9384 - Arno)
- `std_readerp.m`, fix reading multiple channels and components in STUDY (SVN 9386 - Arno)
- `statcond.m`, allow comparing identical conditions (SVN 9387 - Arno)
- `std_plotcurve.m`, `std_erpplot.m`, fix when several threshold are set and curve are being plotted (SVN 9388 - Arno)
- `std_readerp.m`, warning message in case of ERSP baseline incompatibility (SVN 9389 - Arno)
- `Contents.m`, change revision number (SVN 9390 - Arno)

14.3

**Issue date:** August 29th, 2011
**SVN revision:** 9382
**External modules:** FILEIO SVN version 3587, BIOSIG (partial) SVN version 2694
- `std_makedesign.m`, `std_checkset.m`, fixing duplicate base file in STUDY design 1 (SVN 9349 - Arno)
- `pop_erpparams.m`, header update (SVN 9350 - Arno)
- `std_maketrialinfo.m`, fix bug when dealing with EEG objects (SVN 9351 - Arno)
- `std_plottf.m`, fix rare plotting bug when the number of frequency and time point is equal (SVN 9352 - Arno)
- `std_readfile.m`, now allow to write custom information in 'datafile' field (SVN 9353 - Arno)
- `std_chaninds.m`, issue warning instead of error if a channel is missing (SVN 9354 - Arno)
- `std_ersp.m`, fix important bug for computing ERSP of components in STUDY (SVN 9355 - Arno)
- `std_readerp.m`, fix bug when retrieving non-interpolated channels in STUDY (SVN 9356 - Arno)
- `std_savedat.m`, now allow to write custom information in 'datafile' field (SVN 9357 - Arno)
- `std_preclust.m`, fix time limit assessment for warning message (SVN 9358 - Arno)
- `dipplot.m, pop_dipfit_nonlinear.m, pop_dipplot.m`, fix normalizing symmetrical dipoles and change default plotting options (SVN 9359 - Arno)
- `pop_studydesign.m`, fix combining values (SVN 9360 - Arno)
- `logimagesc.m`, adding output for axis and tick labels (SVN 9361 - Tim checked in by Arno)
- `pop_erpimage.m`, fix problem with projection channel (SVN 9362 - Arno) bug 1124
- `pop_writeeeg.m`, convert epoched data to continuous data (SVN 9363 - Arno)
- `pop_saveset.m`, fix saving ICA data files when running ICA in STUDY (SVN 9364 - Arno) bug 1123
pop_runica.m, add pop up window to query user when rank is not full (SVN 9365 - Arno)
statcond.m, add the option to process input bootstrap array (SVN 9366 - Arno)
biosig2eeglab.m, change default threshold to detect events in the last data channel (SVN 9367 - Arno)
pop_loadstudy.m, show warning for corrupted ERSP files in STUDY (SVN 9377 - Arno)
std_makedesign.m, fix the code that removed duplicate entry in design 1 (SVN 9378 - Arno)
update revision number (SVN 9379 - Arno)
std_readspec.m, std_specplot.m, removing old code that was not executed (SVN 9380 - Arno)
pop_studydesign.m, fixing menu for combined subject groups (SVN 9381 - Arno)
std_readerp.m, fix subtracting individual subject spectrum baseline in the presence of empty cells (SVN 9382 - Arno)

14.4

- Issue date: July 27th 2011
- SVN revision: 9348
- External modules: FILEIO SVN version 3587, BIOSIG (partial) SVN version 2694
  - eeglab.m, now automatically remove the path to fmrlab to avoid conflict with its function (SVN 9327 - Arno)
  - topoplot.m, fix setting EMARKERSIZE variable (SVN 9328 - Arno)
  - pop_runica.m, fix rare crash when concatenating subjects in STUDY (SVN 9329 - Arno)
  - eeg_checkset.m, when checking boundary event, allow for mixed string and numerical event types (SVN 9330 - Arno)
  - eeg_checkset.m, move section checking event boundaries to avoid having to deal with mixed numerical and string event types (SVN 9331 - Arno)
  - std_makedesign.m, fix modifying default design for old file format - the filebase field was changed while it should not have been (SVN 9332 to 9333 - Arno)
  - std_getindvar.m, allow using independent variables even when they are missing from some datasets (SVN 9334 - Arno)
  - inputgui.m, supergui.m, allows GUI positioning and centering (SVN 9335 - Arno)
  - pop_clustedit.m, allows optional GUI parameters (SVN 9336 to 9337 - Arno)
  - std_readfile.m, fix reading several channels in several files (SVN 9338 - Arno)
  - std_chaninds.m, allow processing channel location structures (SVN 9339 - Arno)
  - std_readersp.m, std_readerp.m, fixing reading channels when they are not saved in the same order as the channel location structure in STUDY. For more information see Interpolation_bug_July_2011 (SVN 9340 - Arno)
  - std_precomp.m, fixing channel order when performing interpolation (SVN 9341 - Arno)
  - vararg2str.m, allowing empty entry (SVN 9342 - Arno)
  - eeg_getdatact.m, allowing to enter projection channel(s) for ICA components (SVN 9343 - Arno)
  - pop_erpimage.m, allowing to enter projection channel(s) for ICA components (SVN 9344 - Arno)
  - Contents.m, update revision number
  - eegplot.m, do not show values and electrodes in stacked mode (SVN 9348 - Ozgur checked in by Arno)

14.5

- Issue date: July 11th, 2011
- SVN revision 9326
- External modules: FILEIO SVN version 3587, BIOSIG (partial) SVN version 2694
  - std_erp.m, processing empty dataset input (SVN 9300 - Arno)
  - eeg_getdatact.m, fix selecting empty set of trials (SVN 9301 - Arno)
  - std_getindvar.m, allow processing events either as string or scalar from unhomogenous datasets (SVN 9302 - Arno)
  - eeg_optionsbackup.m, message update (SVN 9303 - Arno)
  - eeglab_options.m, pop_editoptions.m, optional reading from folder defined in icadefs.m (SVN 9304 - Arno)
  - icadefs.m, adding folder settings for reading eeglab options (SVN 9305 - Arno)
  - eegplot.m, fix plotting topography (SVN 9306 - Arno)
  - biosig2eeglab.m, update channel location import which was not function with updated BIOSIG function (SVN 9307 - Arno)
  - readeigiloc.m, update message to user (SVN 9308 - Arno)
  - readneurolocs.m, handle new carthesian coordinate 3-D file from Neuroscan (SVN 9309 - Arno)
  - eeg_checkset.m, added new section to check boundary event consistency (SVN 9310 - Arno)
  - pop_importegimat.m, import new Matlab from EGI Matlab files (SVN 9311 - Arno)
  - pop_resample.m, fix boundary event issue - when two events are too close to each other - and fix the option for not using the signal processing toolbox (SVN 9312 - Arno). Note that the 3 functions
pop_resample.m, pop_select.m and pop_rejspec.m were submitted under the same SVN number by mistake.

pop_select.m, fix removing fields that do not exist for EEG object (SVN 9312 - Arno)

pop_rejspec.m, fix calling the function from the general rejection menu (SVN 9312 - Arno)

statcond.m, make the function use inhomogenous variance by default (SVN 9213 - Arno)

deleting functions evalc.m and uimenu.m in octavefunc folder for octave that were creating warning messages for some users (SVN 9214 - Arno)

@eegobj/subassign.m, fix assigning indexed structure that would create empty elements in between indices (SVN 9315 - Arno)

std_readfile.m, test for empty array in data files (SVN 9316 - Arno)

std_precomp.m, do not compute ERP or ERSP if datasets of the STUDY do not have the same number of data points (SVN 9317 - Arno)

std_ersp.m, allow computing ERSP of empty array - this is useful in case of specific rare designs (SVN 9318 - Arno)

std_erp.m, allow computing ERP of empty array - this is useful in case of specific rare designs (SVN 9319 - Arno)

std_plotcurve.m, reprogrammed part of the function to enforce the data (in all cases) to be of the format "points x channels x subject x conditions". This simplifies processing and fixes some rare problems (SVN 9320 - Arno)

std_figtitle.m, fix for very short title (1 char) that were creating an error (SVN 9321 - Arno)

pop_precomp.m, check if precomputed file is present and issue a warning (SVN 9322 - Arno)

std_readersp.m, allow processing empty input array (SVN 9323 - Arno)

pop_studyerp.m, check if precomputed file is present and issue a warning (SVN 9324 - Arno)

eeglab.m, remove any reference to Octave direct path and fix reading icadefs.m by eeglab_options by moving it down in the code (SVN 9325 - Arno)

Contents.m, update revision number (SVN 9326 - Arno)

14.6

- Issue date: June 22th, 2011
- SVN revision 9299
- External modules: FILEIO SVN version 3587, BIOSIG (partial) SVN version 2694
  - std_erp.m, processing empty dataset input (SVN 9300 - Arno)
  - eeg_getdatact.m, fix selecting empty set of trials (SVN 9301 - Arno)

14.7

- Issue date: June 21th, 2011
- SVN revision 9297
- External modules: FILEIO SVN version 3587, BIOSIG (partial) SVN version 2694
  - dipplot.m, fix zooming issue (SVN 9296 - Arno)
  - Contents.m, update revision number (SVN 9297 - Arno)

14.8

- Issue date: June 17th, 2011
- SVN revision 9294
- External modules: FILEIO SVN version 3587, BIOSIG (partial) SVN version 2694
  - pop_corrmap.m, removing code accessing channel location file name that was generating crashes in rare cases (SVN 9294 - Arno)
  - Contents.m, update revision number (SVN 9295 - Arno)

14.9

- Issue date: June 16th, 2011
- SVN revision 9293
- External modules: FILEIO SVN version 3587, BIOSIG (partial) SVN version 2694
  - eeglab.m, activate chat and Octave compatibility (SVN 9239 - Arno)
  - eeglabchat.m, remove eeglabchat.m temporary function (SVN 9240 - Arno)
  - supergui.m, shift text slightly for alignment with buttons and text boxes (SVN 9241 - Arno)
  - ismatlab.m, new function that determine if the user is running Matlab or Octave (SVN 9242 - Arno)
  - octavefunc, new folder containing functions for Octave only (SVN 9243 - Arno)
  - eegplot.m, making marked region half transparent so it is possible to visualized superposed regions (SVN 9244 - Arno)
@eegobj/subsasgn.m, better creation of empty dataset (SVN 9245 - Arno)

A few EEGLAB functions were modified to be made compatible with Octave. Note that, at this point, we do not intend to make functions with graphical outputs compatible with Octave (SVN 9246 - Arno)

corrmap plugin: Octave compatibility, remove statistics toolbox dependency, added option 'plot', improved help message (SVN 9247 - Arno)

bdplugin, Octave compatibility (SVN 9248 - Arno)

bva-io, Octave compatibility (SVN 9249 - Arno)

bva-io, remove dependency on the ismatlab function (SVN 9250 - Arno)

pop_dipfit_settings.m, Octave compatibility (SVN 9251 - Arno)

std_rebuilddesign.m, new function for rebuilding STUDY design after the datasets in the STUDY have been modified (SVN 9252 - Arno)

pop_studyerp.m, new function with a simplified interface for grand average ERP analysis (SVN 9253 - Arno)

std_checkset.m, integration of std_rebuilddesign (SVN 9254 - Arno)

pop_study.m -m, automatically select field to delete cluster information when the user select components by dipoles (SVN 9255 - Arno)

pop_preclust.m, improve error message and fix history output (SVN 9256 - Arno)

pop_precomp.m, fix dependency on statistic toolbox (SVN 9260 - Arno)

corrcmap_cell.m, speedup function by 2 by using bsxfun function (SVN 9261 - Arno)

spectopo.m, when the statistics toolbox is present, now using pwelch function from Octave instead of the spec.m function (SVN 9262 - Arno)

eegfilt.m, remove error when the signal processing toolbox is absent (SVN 9263 - Arno)

chancenter.m, add call to fminsearch function from Octave (SVN 9264 - Arno)

fminsearch, pwelch, filtfilt, etc... adding signal processing and optimization toolbox replacement functions for Octave (SVN 9265 - Arno)

std_readfile.m, fixing function for processing file names containing "." characters (SVN 9266 - Arno)

bug 1051

pop_studydesign.m, fixing combining conditions which had already been combined (SVN 9267 - Arno)

std_preclust.m, better sensitivity for triggering warning message for datasets with different time limits (SVN 9268 - Arno)

pop_studyerp.m, adding Scott's GUI edit and fixing crash when pressing cancel button (SVN 9269 - Arno)

std_readtopoclust.m -m "minor backward compatible code edition (SVN 9270 - Arno)

importevent.m, adding fminsearch of Octave in case optimization toolbox is missing (SVN 9271 - Arno)

pop_editset.m, adding dataset check at the end of the function (SVN 9272 - Arno)

pop_spectopo.m, fix figure pop up when calling function for pure calculation (SVN 9273 - Arno)

eeglab.m, is_sccn.m, eeglab2fieldtrip.m, chat warning if not at UCSD, adding paths to replacement function if Matlab signal processing and optimization toolboxes are absent, menu edition.

eeglab2fieldtrip.m fix for processing channel data (SVN 9274 - Arno)

newtimef.m, revert accidentsial submission in 9274 (SVN 9275 - Arno)

eeglab.m, fix EEGLAB chat Java import (SVN 9276 - Jeff submitted by Arno)

pop_mergeset.m, fix modifying event structure with events containing empty urevent indices (SVN 9277 - Christian submitted by Arno)

eeglab.m, fix path and chat for UCSD users (SVN 9278-9279 - Arno)

erpimage.m, try-catch clause to call the proctile function (SVN 9280 - Arno)

topoplot.m, octave compatibility (SVN 9281 - Arno)

pop_rejspec.m, fix output indices (SVN 9282 - Arno)

pop_fileio.m, fix option forwarding to the ft_read_data function (SVN 9283 - Arno)

pop_selectevent.m, header fix (SVN 9284 - Arno)

eeglab.m, fix call to pop_rejspec function (SVN 9285 - Arno)

std_readtopo.m, avoid entering infinite loop (SVN 9286 - Joaquin checked in by Arno)

eeg_checkset.m and eeg_checkchanlocs.m, speed optimization using the deal function (SVN 9287 - Christian checked in by Arno)

Contents.m, change revision number (SVN 9288 - Arno)

eeglab.m, fixing adding the fminsearch folder for Octave (SVN 9289 - Arno)

dipplot.m, fix camera zoom (SVN 9290 - Arno)

egglab_options.m, fix finding the correct eeg_options file (SVN 9291 - Arno)

icadefs.m, font size option (SVN 9292 - Arno)

eeg_checkset.m, removing channel location structure if it does not fit with the number of channels (SVN 9293 - Arno)
14.10

- Issue date: May 9th, 2011
- SVN revision 9238
- External modules: FILEIO SVN version 3382, BIOSIG (partial) SVN version 2667
  - loadeep.m, add warning for importing events when selecting a data (SVN 9190 - Arno)
  - eegplugin_dipfit.m, change "h" to "eegh" (SVN 9191 - Arno)
  - readlocs.m, added support to convert Fieldtrip channel location structure (SVN 9192&9193 - Arno)
  - eegthresh.m, check lower boundary to avoid crash (SVN 9194 - Arno)
  - eeg_checkset.m, fix for EEG object support (SVN 9195 - Arno)
  - pop_rejspec.m, update the function to use 'key', val input arguments (SVN 9196 - Arno)
  - eeg_mergelocs.m, update warning message (SVN 9197 - Arno)
  - pop_rejchan.m, added option to use precomputed measure as input (SVN 9198 - Arno)
  - pop_rejchanspec.m, added option to use precomputed channel spectrum as input (SVN 9199&9200 - Arno)
  - pop_rejcont.m, added option to visualize rejected portions of data and reuse precomputed data (SVN 9201&9202 - Arno)
  - superfui.m, add try, catch clauses and set warning off (SVN 9203 - Christian - checked in by Arno)
  - eegrej.m, fix when events may have latencies < 0.5 (e.g. after resampling) (SVN 9204 - Christian - checked in by Arno)
  - runica.m, test for empty values (SVN 9205 - Christian - checked in by Arno)
  - eeglab_options.m, pop_editoptions.m, now clearing function in pop_editoptions instead of eeglab_options to speed up calculations (SVN 9206 - Christian - checked in by Arno)
  - iseeglabdeployed.m, add try, catch clause (SVN 9207 - Christian - checked in by Arno)
  - eeg_checkset.m, check that the data field is not empty when checking chanlocs structure (SVN 9208 - Christian - checked in by Arno)
  - pop_editeventvals.m, minor code optimization (SVN 9209 - Christian - checked in by Arno)
  - eeg_decodechan.m, minor code optimization (SVN 9210 - Christian - checked in by Arno)
  - pop_resample.m, minor code optimization (SVN 9211 - Christian - checked in by Arno)
  - pop_resample.m, minor code optimization (SVN 9212 - Christian - checked in by Arno)
  - pop_mergeset.m, heavy code optimization, not object compatible yet (SVN 9212 - Christian - checked in by Arno)
  - pop_select.m, minor code optimization (SVN 9213 - Christian - checked in by Arno)
  - pop_epoch.m, fix selected epoch indices (output) when epoch are selected (SVN 9213:9214 - Christian - checked in by Arno)
  - coregister.m, fix crash when canceling electrode selection (SVN 9215 - Arno)
  - laplac2d.m, edit header and add warning (SVN 9216 - Arno)
  - pop_resample.m, pad data to avoid border effects (SVN 9217 - Andreas Widmann - checked in by Arno)
  - pop_rejcont.m, remove all non bounadry events to avoid problem when event are present at epoch limits (SVN 9218 - Arno)
  - @eegobj/rmfield.m, better compatibility with structures (SVN 9219 - Arno)
  - eeglab_options.m, reprogram function to use eeg_options.m in home folder (SVN 9220 - Arno)
  - pop_editoptions.m, now uses eeg_options.m in home folder (SVN 9221 - Arno)
  - eeg_optionsbackup.m, added the chat option (SVN 9222 - Arno)
  - javachatfunc.m, adding java chat functions (SVN 9223 - Arno)
  - laplac2d.m, header edit (SVN 9224 - Arno)
  - eeg_laplac.m, new function to compute laplacian on EEGLAB structure (SVN 9225 - Arno)
  - eeglab_options.m, temporary main EEGLAB window to interface EEGLAB chat (SVN 9226 - Arno)
  - pop_editoptions.m, eeglab_options.m, fix reading default eeg_options.m file (SVN 9227 - Arno)
  - decompressserps.mexmac64, add option to decompress under 64-bit OS X (SVN 9228 - Arno)
  - pop_editoptions.m, fix error message (SVN 9229 - Arno)
  - pop_mergeset.m, fix object compatibility (SVN 9230 - Arno)
  - newtimef.m, fix FFT normalization and disable 'lowmem' option (SVN 9231 - Arno)
  - eegplotold.m, adding back the eegplotold.m function so the "icademo" function can run (SVN 9232 - Arno)
  - eeglab.m, speeding up plugin function call (SVN 9233 - Christian checked in by Arno)
  - eeglabchat.m, fix when the java jar file is not in the current folder (SVN 9234 - Arno)
  - eeglab_options.m, fix home directory for windows user (SVN 9235 - Arno)
  - newtimef.m, octave parsing compatibility (SVN 9236 - Arno)
  - renaming chat_with_pane.jar to Chat_with_pane.jar (SVN 9237 - Arno)
  - changing revision to 10.1.2.0 in the Contents.m file (SVN 9238 - Arno)
14.11

- Issue date: April 11th, 2011
- SVN revision 9189
- External modules: FILEIO SVN version 3174, BIOSIG (partial) SVN version 2667 (March 18th, 2011)
  - std_plotcurve.m, fix plotting error for standard error shaded area (SVN 9180 - Arno)
  - Contents.m, changed revision number for 10.0.1.0 and reissue a release (SVN 9181 - Arno)
  - decompresserpss.m, fixed compiled function for Windows 32-bit (SVN 9182 - Arno)
  - eeg_check.m, update by C. Kothe for efficiency (SVN 9183 - Christian, submitted by Arno)
  - @eegobj, Adding the @eegobj EEG object (SVN 9184 - Arno)
  - updating about 70 functions for EEG object compatibility - THIS IS A MAJOR SUBMISSION (SVN 9185 - Arno)
  - eeg_checkset.m, fixed typo when calling ALLEEGNEW (SVN 9186 - Arno)
  - eeg_eegrej.m, pop_epoch.m, pop_select.m, added eeg_checkamica (SVN 9187 - Ozgur, submitted by Arno)
  - pop_mergeset.m, fix urevent with no latency information (SVN 9188 - Arno)
  - Contents.m, change revision index (SVN 9189 - Arno)
  - fix script copying ANT plugin to release (no SVN tag)

14.12

- Issue date: March 19th, 2011
- SVN revision 9179
- External modules: FILEIO SVN version 3174, BIOSIG (partial) SVN version 2667 (March 18th, 2011)
  - Because of too many compatibility issues, Fieldtrip external link was removed and necessary functions for source localization were added back to the external folders (partial Fieldtrip at SVN 2942). These functions are now within the EEGLAB SVN (SVN 9145-9148 - Arno)
  - DIPFIT plugin, adding functions for electrode warping. These functions were previously in Fieldtrip but some functions were not backward compatible and we could not find the problem. This uses functions from the 2007 version of Fieldtrip (SVN 9150 - Arno)
  - coregister.m, fix for compatibility with old warping function added back above (SVN 9151 - Arno)
  - pop_fileio.m, fix for new version of FILEIO (SVN 9152 - Arno)
  - std_readic.m, for command line users, fix header and function which was not working any more - but it is not called by any other EEGLAB function (SVN 9153 - Arno)
  - std_readsp.m, fix header (SVN 9154 - Arno)
  - FILEIO was added as an external module to the external folder. This is a SVN direct link to the FILEIO repository (SVN 9155 - Arno)
  - in the private folder of Fieldtrip, change read_vol call to ft_read_vol for prepare_headmodel.m (SVN 9156 - Arno)
  - readlocs.m, add 'CMS' and 'DRL' to the list of non data channel (SVN 9157 - Arno)
  - eeg_checkset.m, remove the portion of the code for channel checking - now performed in eeg_checkchanlocs.m (SVN 9158 - Arno)
  - eeg_checkchanlocs.m, new function to check channel location structure consistency (SVN 9159 - Arno)
  - std_figtitle.m, fix typo that was generating rare crash (SVN 9160 - Arno)
  - pop_channedit.m, now uses the function eeg_checkchanlocs.m to check channel locations - major edits (SVN 9161 - Arno)
  - pop_autorej.m, fix history call for GUI popup (SVN 9162 - Arno)
  - ANT plugin update (SVN 9163 - Arno)
  - ANT plugin rename (SVN 9164 - Arno)
  - pop_writebva.m, allow marker number discontinuity in BVA plugin (SVN 9165 - Andreas)
  - pop_writebva.m, fixed new segment bug and added boundary event conversion in BVA plugin (SVN 9166 - Andreas)
  - eegplot.m, new button to collapse all EEG traces by Ozgur Balkan (SVN 9167 - Ozgur & Arno)
  - pop_select.m, pop_epoch.m, eeg_eegrej.m, update by Ozgur Balkan for the AMICA plugin (SVN 9168 - Ozgur & Arno)
  - eeglab.m, fix call to pop_rejchan.m to avoid double pop-up of new data window (SVN 9169 - Arno)
  - reref.m, fix referencing when an external channel had low index - which is in theory not possible if the interface is being used bug 1001; fixed retaining the reference channels when multiple channels are used (SVN 9171 - Arno)
  - pop_reref.m, fix when multiple references are being used (SVN 9172 - Arno)
  - eeg_latencyur.m, new function to compute the reverse of eeg_uurlatency.m (SVN 9173 - Arno)
  - eeg_uurlatency.m, fix header and allow computing latencies on input arrays in addition to single values (SVN 9174 - Arno)

32
pop_select.m, pop_epoch.m, put back recent fixes not included in Ozgur version (SVN 9175 - Arno)
pop_rejcont.m, fix header, add new option 'eegplot' (SVN 9176 - Arno)
eeg_regepochs.m, merge datasets when several datasets are given as input; this fixes a rare bug in
STUDY when computing spectrum on continuous dataset (SVN 9177 - Arno)
std_erpplot.m and std_specplot.m, add option plotstderr for plotting standard error of the mean
across datasets - experimental (SVN 9178 - Arno)
showmiclusts.m, only loop over displacement if coordinates are present (SVN 9179 - Nima & Arno)

14.13

- Issue date: Feb 11th, 2011
- SVN revision 9044
- External modules: Fieldtrip (partial) SVN version 2578, BIOSIG (partial) SVN version 2578
  
  - pophelp.m, backward compatibility with Matlab 7.4 (SNV 9115 - Arno)
  - eeg_regepochs.m, add 'extractepoch' entry and allow to use 'key', 'val' format for inputs (SVN 9116 -
Arno)
  - std_indvarmatch.m (new function), std_setcomps2cell.m, std_changroup.m, std_selectdataset.m,
    std_checkset.m, pop_studydesign.m, std_fiptitle.m, std_getindvar.m, std_makedesign.m, fixed
    handling of numerical values for independent variables (was not working before), had to change the
    why the values are stored when multiple values are selected both for string and numerical values.
    When multiple strings are selected, they are stored in a cell array (before they were stored as a string).
    (SVN 9117 - Arno)
  - std_plotcurve.m, fix minor bug plotting curve together for groups (SVN 9120 - Arno - merge of SVN
    9119 from EEGLAB 9)
  - Fieldtrip functions fieldtripdefs.m ft_electroderealign.m ft_channelsselection.m ft_dipolefitting.m
    updated in EEGLAB repository to Fieldtrip SVN revision 2578. (SVN 9122 - Arno)
  - std_erpplot.m, std_fiptitle.m, and std_plotcurve.m, generate legends for all figures and attempt to fix
    all title inconsistencies. (SVN 9123 - Arno)
  - adding corrmap plugin to central repository (SVN 9124 - Arno)
  - pop_saveset.m, rthrow last error instead of generating a template error message (SVN 9125 - Arno)
  - fixing corrmap plugin to generate clusters and ignore identical ICA decompositions (SVN 9126 -
Arno)
  - std_topoplot.m, fix titles so they are not interpreted in Latex (SVN 9127 - Arno)
  - pop_studydesign.m, fix crash when user create two identical combined sets of independent values
    (SVN 9128 - Arno)
  - pop_erpparams.m and pop_specparams.m, remove constraint of not being able to plot both
    independent variables on the same panel (SVN 9129 - Arno)
  - pop_loadstudy.m, reselect current design when loading a STUDY. This fixes some issues with
    corrupted .setinds and .allinds fields for channels and clusters (SVN 9130 - Arno)
  - std_createclust.m, add option to ignore cluster index 0 and fix crash when STUDY.cluster.preclust is
    not set (SVN 9131 - Arno)
  - std_plotcurve.m, fix plotting all subjects in some cases, plotting both independent variables on the
    same panel, and replace all & and | by && and || (SVN 9132 - Arno)
  - std_erpplot.m, fix plotting several ERPs between 2 and 5 channels, plotting panel title and legend
    (SVN 9133 - Arno)
  - pop_runica.m, fixing selecting channel type and indices when running the function from the STUDY
    (SVN 9134 - Arno)
  - plotcurve.m, do not interpret titles in latex (SVN 9135 - Arno)
  - std_readerp.m, fix rare crash when some conditions/groups are missing (SVN 9136 - Arno)
  - std_plotcurve.m, fix color issue when conditions/groups are missing (SVN 9137 - Arno)
  - std_createclust.m, fix creating cluster with empty or absent STUDY.etc.preclust structure (SVN 9138
    - Arno)
  - pop_preclust.m, new warning message if some subjects are missing (SVN 9139 - Arno)
  - std_clustmaxelec.m, new function to find the electrode with the maximum weight for each
    independent component of a cluster (SVN 9140 - Arno)
  - Contents.m, update revision number (SVN 9141 - Arno)
  - loadcnt.m, pop_epoch.m, update error messages (SVN 9142 & 9143 - Arno)
  - eeg_regepochs.m, fix optional parameters in legacy call (SVN 9144)
15 EEGLAB TUTORIAL OUTLINE

15.1

[Download the current Wiki Tutorial as a PDFbook](#)
(Note: The PDF is generated dynamically. Please do not refresh the page before it begins downloading.)

-- Getting Started
-- Quick Rejection Tutorial

15.1.1

<ncl>Category:I.Single subject data processing tutorial</ncl>

15.1.1.1

<ncl>Category:II.Multiple subject processing tutorial</ncl>

15.1.1.2

<ncl>Category:III. Advanced Topics</ncl>

15.1.1.3

<ncl>Category:IV. Appendix</ncl>

15.1.1.4

This tutorial was written by: Arnaud Delorme, and Scott Makeig. The tutorial on compiling EEGLAB for developers was written by Ramon Martinez-Cancino, Arnaud Delorme, and Scott Makeig

Many thanks to Hilit Serby, Nima Bigdely, and Toby Fernsler for additions and or edits. Thanks also to Payton Lin for capturing some images in earlier versions and to Micah Richert, Yannick Marchand, Elizabeth Milne, and Stefan Debener for their detailed comments. In addition, thanks to all those who have contributed code and suggestions to EEGLAB, and to Devapratim Sarma for converting and updating the EEGLAB documentation to a WIKI.

Return to SCCN WIKI HOME
### EEGLAB vs. Commercial EEG Software

"Can we trust the results of a new paper if they depend on calculations carried out by proprietary software with non-public source code?" *M. Buchanan, Digital science, Nature Physics, June 2016*

This page compares the feature of EEGLAB some of the most common features in what is currently best in the industry.

<table>
<thead>
<tr>
<th>Feature</th>
<th>EEGLAB</th>
<th>Leading EEG commercial softwares</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary file import</td>
<td>EEGLAB offers a comprehensive library of file import functions. Most data formats can be imported using three different methods.</td>
<td>File import and export are usually limited to a few formats.</td>
</tr>
<tr>
<td>Memory requirements</td>
<td>EEGLAB has a considerable memory requirement for processing single data sets. When processing multiple data sets, they may stay on disk but EEGLAB must be able to hold any of the datasets in memory. This has been moderated by the new EEGLAB memory-mapping feature which allows the data to stay on disk. This EEGLAB feature is, however, still under development and only halves EEGLAB memory requirements.</td>
<td>Most commercial software has been designed to allowing processing of large datasets using relatively little memory</td>
</tr>
<tr>
<td>Features</td>
<td>EEGLAB has more features than any current commercial software. In general EEGLAB provides the user with a wider range of processing choices.</td>
<td>Some leading (though frequently expensive) commercial software currently offers more methods for source localization than are available EEGLAB tools. However, see the newly available Neuroelectromagnetic Forward head modeling Toolbox (NFT) that operates on EEGLAB data, and several other freely available Matlab packages for this purpose.</td>
</tr>
<tr>
<td>Availability of new features</td>
<td>EEGLAB benefits from tools implemented by the scientific community. To our knowledge, already at least three new toolboxes use EEGLAB dataset format and EEGLAB function as their base: ERPWAVELAB, ERPLAB, BCILAB.</td>
<td>A large part of commercial software development supports software compiled under Windows OS. Development of both EEGLAB and dedicated commercial software are driven by the open source and research community; as it adopts a new methods, commercial software will implement them eventually. EEGLAB allows testing relatively new tools, many of which may not yet be available in any commercial software</td>
</tr>
<tr>
<td>Graphic interface</td>
<td>EEGLAB graphic interface may not be as refined as in commercial software. A lot of the processing is performed on the command line; the EEGLAB graphic user interface (gui) is only a convenient way to automate such processing. In future, we intend to separate the graphic interface functions from the rest of the EEGLAB distribution so experts in graphic interface design can develop their own &quot;skins&quot; for EEGLAB.</td>
<td>Commercial software developers usually perfect their graphic interface, sometimes at the expense of the breadth of tools implemented.</td>
</tr>
<tr>
<td>Documentation</td>
<td>EEGLAB has more than 300 pages of tutorial documentation. Users may also access the EEGLAB code to check exactly what processing is performed. In addition, each function has its own documentation</td>
<td>Documentation is usually not a top priority of commercial software, and its source code is not available.</td>
</tr>
<tr>
<td>Code stability</td>
<td>Because EEGLAB is in constant development, the code stability of the development version may not be as optimal as for commercial software. The EEGLAB code is also not rigorously validated against given standard or certification by industry standards, though we intend to address these issues in future revisions. Note also that EEGLAB is not approved for routine clinical use and should not be used for such purposes.</td>
<td>The most reliable commercial code may be more stable; though this is not necessarily true for all commercial code.</td>
</tr>
<tr>
<td>Scripting capabilities</td>
<td>The EEGLAB scripting language is Matlab itself. Commercial code cannot compete with Matlab for range, flexibility, and the amount of code available</td>
<td>Scripting capabilities usually rely on a proprietary language. Some commercial software allows running Matlab code within their scripts -</td>
</tr>
<tr>
<td>Preparation of figures for publication</td>
<td>EEGLAB and Matlab allow creation of complex figures with panels. Most of EEGLAB functions are compatible with panels so users may use EEGLAB function to generate their own paneled results. Formatting details of figures may be edited directly under Matlab from the command line or from the Matlab GUI. Even complex figures containing bitmaps may be saved as postscript files for further detailed editing. Matlab also allow saving figures and movies in about 10 different formats.</td>
<td>Figures may only be saved using a few formats. Capabilities to build complex figures from within the software is absent.</td>
</tr>
<tr>
<td>User support</td>
<td>EEGLAB Bugzilla database helps track EEGLAB bugs. In the best case scenario, bugs may be fixed within 24 hours and a new release issued automatically overnight. However, the EEGLAB team does not have dedicated personnel for supporting users, and the help provided may depend on availability and advancement on other research projects.</td>
<td>Support personnel are usually more readily available to help users, and they can have high level of expertise. Note that this is not true of all commercial software.</td>
</tr>
<tr>
<td>Price</td>
<td>A minimal Matlab installation may be within most research budgets or be already available for other purposes in research and university settings. The compiled version of EEGLAB does not require Matlab although scripting capabilities are more limited.</td>
<td>Prices range from $5000 to $30000</td>
</tr>
</tbody>
</table>
17 EEGLAB Wiki

17.1

17.1.1

EEGLAB hardware and software recommendations
Download EEGLAB as zip file
Download EEGLAB from GIT
Download a compiled version of EEGLAB
EEGLAB extensions/plug-ins
EEGLAB revision history
Bugs and Suggestions

17.1.1.1

The EEGLAB tutorial is available in a subsequent section on this page. Other type of documentation are listed below.

EEGLAB News and Discussion email lists
EEGLAB reference articles - Please cite
Using EEGLAB vs. Commercial EEG Software
Working with EEGLAB and Fieldtrip
Running EEGLAB on open source Octave
EEGLAB on Matlab versus Python
NEW Running EEGLAB on high performance computing resources - The Open EEGLAB Portal
NEW Reading 3D electrode locations from an inexpensive 3D photo scan - get_chanlocs
Using EEGLAB to process MEG data
List of functions called by the EEGLAB menu
EEGLAB History: The first decade of 2001-2011
Run EEGLAB on NSF Supercomputers!

17.1.1.2

EEGLAB and MEX functions to recompile
EEGLAB filtering FAQ
Use Google - add "eeglablist" or "EEGLAB" to your query
Ask eeglablist@sccn.ucsd.edu (requires subscription here)
Post a bug on Bugzilla for EEGLAB
Other TIPS and FAQ
17.1.1.3
EEGLAB extensions and plug-ins
get_chanlocs - record 3-D electrode positions quickly using a 3-D camera
HeadIT - Human Electrophysiology, Anatomic Data & Integrated Tools
Publicly available datasets for download
ERICA - Experimental Real-time Interactive Control and Analysis framework
Channel Location Files download
Binary version of the runica() infomax ICA decomposition function
Download EEGLAB test scripts

17.2

Mugs from the 17th EEGLAB workshop

The Online EEGLAB Workshop - Includes online videos, slides, and tutorial materials!

Future EEGLAB workshops
Thirty-first EEGLAB Workshop - Lublin, Poland (2020)
Thirtieth EEGLAB Workshop - San Diego, USA (2020)
Twenty-ninth EEGLAB Workshop - Aspet, France (2019)
Twenty-eighth EEGLAB Workshop - San Diego, USA (2018)
Twenty-seventh EEGLAB Workshop - Pittsburgh, USA (2018)
Twenty-sixth EEGLAB Workshop - Beer-Sheva, Israel (2017)
Twenty-fifth EEGLAB Workshop - Tokyo, Japan (2017)
Twenty-fourth EEGLAB Workshop - Aspet, France (2017)
Twenty-third EEGLAB Workshop - Mysore, India (2017)
Twenty-second EEGLAB Workshop - San Diego, USA (2016)
Twenty-first EEGLAB Workshop - Santa Margherita Ligure, Italy (2016)
Twentieth EEGLAB Workshop - Sheffield, UK (2015)
Nineteenth EEGLAB Workshop - Aspet, France (2015)
Eighteenth EEGLAB Workshop - Rio de Janeiro, Brazil (2014)
Seventeenth EEGLAB Workshop - San Diego, CA, USA (2013)
Sixteenth EEGLAB Workshop - Aspet, France (2013)
Fifteenth EEGLAB Workshop - Beijing, China (2012)
Fourteenth EEGLAB Workshop - La Palma, Mallorca (2011)
Thirteenth EEGLAB Workshop - Aspet, France (2011)
Twelfth EEGLAB Workshop - San Diego, CA, USA (2010)
Eleventh EEGLAB Workshop - Hsinchu, Taiwan (2010)
Tenth EEGLAB Workshop - Jyväskylä, Finland (2010)
Ninth EEGLAB Workshop - Sydney, Australia (2009)
Eighth EEGLAB Workshop - Aspet, France (2009)
Seventh EEGLAB Workshop - Bloomington, Indiana (2009)
Sixth EEGLAB Workshop - Santiago, Chile (2007)
Fifth EEGLAB Workshop - San Diego, CA, USA (2007)
Fourth EEGLAB Workshop - Aspet, France (2007)
Third EEGLAB Workshop - Singapore (2006)
Second EEGLAB Workshop - Porto, Portugal (2005)
First EEGLAB Workshop - San Diego, CA, USA (2004)
17.3.1

Online EEGLAB Workshop - Includes online videos, slides, and tutorial materials!

Download the 2011 Wiki Tutorial as a PDFbook (later versions of the tutorial in PDF format are not available for technical reasons)

Quick ICA Component Rejection Tutorial

17.3.2

Getting Started

17.3.2.1

- Chapter 01: Loading Data in EEGLAB
- Chapter 02: Channel Locations
- Chapter 03: Plotting Channel Spectra and Maps
- Chapter 04: Preprocessing Tools
- Chapter 05: Extracting Data Epochs
- Chapter 06: Data Averaging
- Chapter 07: Selecting Data Epochs and Comparing
- Chapter 08: Plotting ERP images
- Chapter 09: Decomposing Data Using ICA
- Chapter 10: Working with ICA components
- Chapter 11: Time/Frequency decomposition
- Chapter 12: Multiple Datasets

17.3.2.2

- Chapter 01: Multiple Subject Processing Overview
- Chapter 02: STUDY Creation
- Chapter 03: Working with STUDY designs
- Chapter 04: STUDY Data Visualization Tools
- Chapter 05: Component Clustering Tools
- Chapter 06: Study Statistics and Visualization Options
- Chapter 07: EEGLAB Study Data Structures
- Chapter 08: Command line STUDY functions

17.3.2.3

- Chapter 01: Rejecting Artifacts
- Chapter 02: Writing EEGLAB Scripts
- Chapter 03: Event Processing

17.3.2.4

- A01: Importing Continuous and Epoched Data
- A02: Importing Event Epoch Info
- A03: Importing Channel Locations
- A04: Exporting Data
- A05: Data Structures
- A06: EEGLAB option menu
- A07: Contributing to EEGLAB
- A08: DIPFIT
- A09: Using custom MRI from individual subjects
- A10: MI-clust
- A11: BESA (outdated)
- A12: Quick Tutorial on Rejection
- A13: Compiled EEGLAB
18 EEGLAB, big data, and supercomputing applications

We have a funded project to run EEGLAB on the Neuroscience Gateway to run EEGLAB jobs on the San Diego supercomputer. This is a free service and anybody in the world can use it. See this page for more information.

In the short term, Octave is the shortest way to using EEGLAB functions and actually obtain useful results. This page describes how to use EEGLAB on Octave.

18.1

When it comes to using supercomputers, Matlab, although quite efficient, may become incredibly expensive. A single Matlab license may cost $2,100 ($1,050 for academia), and with all its commercial toolboxes might come to $145,000 or more. If you have a supercomputer with about 100 processors (as of 2011, this amounts to about $30,000 or 20,000 euros), you might need to pay the Mathworks about $30,000 to $500,000 to be able to run Matlab on it (the exact price depends on the number of users on the cluster, the number of nodes, and the extra toolboxes). This may be much more than the price of the supercomputer itself! Given that the Matlab core has not evolved dramatically over the past 10 years, and still has flaws (lack of consistency of the graphic interface between platforms; numerical inconsistencies in early version of Matlab 7.0), free alternatives to Matlab are needed in the Open Source community to run computation on supercomputers.

We have attempted to tackle this problem and as of June 2018 (EEGLAB 15+), we are currently supporting Octave (v4.4.0) for supercomputing applications (command line calls only, no graphic support). In our tests, Octave is about 50% slower than Matlab but this can easily be compensated by increasing the number of processors assigned to a specific processing task. Note that EEGLAB functions have not been parallelized (except a few rare exceptions). Therefore, you are required to open a Octave/Matlab session on each node and run custom scripts you write to take advantage of your parallel processing capability. Again, this page describes how to use EEGLAB on Octave.

18.2

Hadoop Mapreduce is a framework for performing computation on large clusters of computers. There are two steps in Mapreduce job: a mapping task where a large number of workers (computers) work on a large number of data lines, and a reduce step, where (usually) a single worker pools all the mapping results.

Below we provide guidelines for using Elastic Mapreduce on the Amazon cloud. Note that Elastic Mapreduce is tailored to processing large quantities of log text files and not binary data. The gain in terms of processing speed compared to the cost of running such solution remains unclear if you have a local cluster of computers. In short, you might spend more time programming the solution and it might cost you more in terms of bandwidth and storage that if you are running it locally. These are the steps you should follow. These are new technologies so expertise in computer science is highly recommended.

- Installing Hadoop command line interface. First install the Command Line Interface to Elastic Mapreduce. This will allow you to configure and run jobs on the Amazon cloud. You will also need to create an AWS account. Hadoop will need to run in streaming mode, where the data is simply streamed to any executable. It might also be possible to run Hadoop in native Java mode and compile Matlab code using the Java builder (this is probably much more complex than using the streaming mode though).

- Transfer your data to Amazon storage cloud (the Amazon storage cloud is named S3). A useful tool to do this is the s3cp tools. Note that your data should be formatted in strings of characters. If you want to process raw EEG data, you will have to serialize it in text, with each channel for example representing one line. There is no limit to the length of a line of text. However, one must remember the overhead in terms of both signal processing and bandwidth associated with processing text. If you have 128 channels and 100 data files, this corresponds to 12800 processing hadoop steps. If you can allocate 1000 workers to the task, this means that each worker will process about 13 channels, a potential speedup of about 1000 on your task. To minimize bandwidth overhead, you might want to transfer the compressed binary data to S3, then have a local amazon EC2 amazon node uncompress it and put it back to S3 (this is because EC2 nodes bandwidth with S3 is free). If you are dealing with Terabytes of data, this task can take a long time (as S3 is configured to have a very slow reading latency and very high writing latency). There are tools to copy data in parallel to S3.

- Solution 1 (easiest to implement) using Octave. EEGLAB command line code is compatible with Octave. Octave may be installed relatively easy on each of the nodes using the bootstraping method (a method to automatically install software on each of the nodes). The command to automatically install Octave on EC2 Amazon nodes is:

  sudo yum ?y install octave --enablerepo=epel
Then, for your main Matlab script, you might want to add the following at the beginning of the main script. This will make it executable and will allow it to process data on STDIN.

```bash
#!/usr/bin/octave -qf
Q = stdin;
```

Hadoop communicate with workers through STDIN and STDOUT pipes. You may write the output of your data processing using the printf or disp Matlab commands.

- **Solution 2**, compiling Matlab code. Compiling Matlab code is the most efficient solution as Matlab compiled code is often 2 to 4 times faster than Octave code and compiled code does not require a Matlab licence. If you compile Matlab code on your local Unix workstation, you will need to make sure to use an Amazon AMI (virtual machine image) with the same set of libraries so that your code can run on that machine. You will need to pick an AMI that is compatible with Hadoop as well. Also, Matlab does not have a simple mechanism allowing it to read from STDIN. The easiest solution is to use third party compiled Mex files to do so (see for example `popen`). Another solution is to have a shell command write STDIN on disk, then call the Matlab executable (although this might impair performance).

- **Reduce step**: once all the worker have computed what they had to compute (spectral power for example), the reduce step may write it back on S3 Amazon storage (and also do further processing if necessary such as grouping back channels belonging to the same subject).

- **Running Hadoop**: using the AWS command line interface, type something like the following.

```
elastic-mapreduce --create --stream -- s3n://Arno/myEEGserializedtextfiles/ \
--mapper s3://Arno/process_octave \
--reducer s3://Arno/reducer. \
--output s3n://Arno/output --debug --verbose \
--log-uri s3n://Arno/logs --enable-debugging \
--bootstrap-action s3n://Arno/install_octave
```

Note the reduce step can be written in any programming language that takes data from STDIN and writes to STDOUT. The reduce step will usually not require to run EEGLAB commands. It is simply about pooling data from the workers and summarizing it. In this case, we used Python custom program (reducer.py) but it could have also been Octave/Matlab since Octave is installed on each of the workers. The exact content of your code will depend on what task you are interested in doing.

The solution outlined above should only be tried when dealing with gigantic amount of data that no local processor or cluster can handle. It is costly (mostly in terms of Amazon storage as storing 10 Terabytes of data will cost you about $800 per month as of 2013). It is therefore best suited when bootstrapping data is required (lots of computation on little data). Send us your comments at eeglab@sccn.ucsd.edu.

**Return to EEGLAB Wiki Home**
19 Firfilt FAQ

19.1
A. You can find explanations in this file(pdf) created by Andreas Widmann.

19.1.1
A. The heuristic for automatically determining the filter length in the legacy basic FIR filter (pop_eegfilt) was inappropriate (possibly causing suboptimal filtering or unexpected filter effects). The new basic FIR filter (pop_eegfiltnew) has a new heuristic for automatically determining the filter length and is based on the firfilt plugin. The legacy function should only be used for backward compatibility purposes.

19.1.2
A. Both are based on windowed sinc filters. The basic filter applies a hardcoded hamming window, has an automatic default for filter length, and is defined by the passband edges. The windowed sinc FIR filter allows manual selection of window type, estimation of filter length by transition band width (no default), and is defined by cutoff frequencies (-6dB, half amplitude).

19.1.3
A. The firfilt plugin does not use filtfilt to achieve zero-phase but shifts the signal by the filter's group delay (NB: requiring ODD filter length/even filter order). So, the data are only filtered once with multi-threading (filtfilt does not seem to be multi-threading capable).

19.1.4
A. Ripple, i.e. deviation from the requested frequency response (0 in stopband, 1 in passband) is equal in passband and stopband in windowed sinc FIR filters. Ripple/attenuation is defined by the window type. 0.002 to 0.001 (that is, 0.2 to 0.1%; Hamming or Kaiser window) are reasonable starting values. This equals a stopband attenuation of -53 to -60 dB which is ok.

19.1.5
A. With separate high- and lowpass filter transition band width can be defined independently. Highpass filter often require narrower transition bands than lowpass filters. Separate filtering is preferable in these cases.

19.1.6
A. Filter order is defined as filter length minus 1.

19.1.7
A. The transition band is the frequency band/range between the passband edge and the stopband edge. In windowed sinc FIR filters the -6dB cutoff is in center of the transition band.
19.1.8

A. Slope CANNOT be defined in dB/oct for windowed sinc FIR filters. Rather use transition band width. There is no straightforward conversion of slope in dB/oct to transition band width due to conceptual differences between FIR and IIR filters. IIR filters do not have a defined stop band.

19.1.9

A. There is no theoretical lower limit, however, the lower the cutoff the steeper is the roll-off and the higher is the filter order (length). Very low cutoff frequencies as low as 0.01 Hz as sometimes found in the literature require extremely long filters (FIR) or are prone to instability (IIR). To our experience a lower limit of about 0.1 Hz is recommendable for FIR filters. For lower cutoff frequencies consider IIR filters combined with a reduced sampling frequency of the signal.

19.1.10

A. Generally, the slope in the frequency domain should be as low/flat as possible, that is the transition band as wide as possible. Steeper slopes reduce the precision in the time domain; distortions and artifacts are additionally spread wider due to the longer filter length. A good starting value for a highpass filter: twice the cutoff frequency (-6 dB) for cutoff <= 1 Hz, 2 Hz for cutoff frequency < 1 and <= 8 Hz and 25% of cutoff frequency for cutoff > 8 Hz. This is also the heuristic implemented in the new basic FIR filter (generalized for all critical frequencies). However, please note, that it is recommended to (manually) adjust this heuristic to the application of interest. Do not go beyond twice of the cutoff frequency (i.e., transition band goes below DC/0 Hz). TIP: Strong attenuation (<< -60dB) can be important for DC/0 Hz (e.g., to get rid of the DC offset for Biosemi files or to avoid baseline correction). By using twice of the cutoff frequency for transition band width, a type 1 windowed sinc filter can be tuned for excellent DC attenuation.

19.1.11

A. The community default of Hamming window/-53dB is a good starting value. With Kaiser windows stopband attenuation can be precisely adjusted.

19.1.12

A. By testing and systematically comparing the effects of different filters on the signal in the time domain.

19.1.13

A. Check out Engineers guide to digital signal processing.

19.1.14

A. There are a few important things to confirm.

- Barnett and Seth (2011) showed that multivariate Granger causality is in theory invariant under zero-phase (a.k.a. phase-invariant) filter. They do recommend filtering to achieve stationarity (e.g., drift, line noise) See Seth, Barrett, Bernett (2015).
- However, in practice, filtering causes problems in calculating multivariate Granger causality. The main problem is the increase in model order. This is because filtering makes the power spectrum density of the signal complicated (low power in the stopband, steep roll-off, etc). See the following example: 33ch EEG, downsampled to 100 Hz, without (top) and with (bottom) applying 44.5Hz low-pass filter (FIR, Blackman, TBW 1 Hz). Notice that the estimated model orders is worsened from 10 (without LPF) to 14 (with LPF)--but apparently taking 16 (with LPF) is the right decision here from AIC, FPE and HQ results.
The second problem, which comes from the same reason mentioned above, is that empirical estimates of VAR parameters yields unstable models due to poor parameter estimate for increased model order.

The third problem is that filtering causes numerical instabilities in estimating causality.

How can we address these problems?

- When applying a high-pass filter to achieve stationarity, let the transition band end at DC (i.e. 0Hz). For example, when you use EEGLAB's 'Basic FIR filter (new, default)' to apply high-pass filter with 'passband edge' below 2-Hz, the transition band is automatically adjusted so that it always ends at DC. (We use 1-Hz highpass for the ICA purpose; empirical test is required to see whether 2-Hz highpass is beneficial for GCA compared with 1-Hz). If you want to set the high-pass filter passband edge above 2 Hz, we recommend you use 'Windowed sinc FIR filter' to design the filter so that it has the stopband at DC. (CAUTION: 'Windowed sinc FIR filter' uses cutoff frequency and not passband edge i.e. cutoff frequency of 1 Hz is equivalent to passband edge at 2 Hz; See Q3)

- When treating the line noise, use CleanLine() instead of notch filter because the former is phase-invariant.

- When downsampling data (which is useful for multivariate Granger causality analysis), use mild anti-aliasing filter and do not let the stopband below the Nyquist frequency. In practice, use the following example.

**EEG = pop_resample(EEG, 200, 0.8, 0.4);**

In this example, you are downsampling your data to 200Hz, with cutoff frequency being 160Hz (i.e. 200Hz*0.8) and the transient bandwidth 80Hz (i.e. 200Hz*0.4).

---

19.1.15


- Tanner D, Norton JJ, Morgan-Short K, Luck SJ. (2016). On high-pass filter artifacts (they're real) and baseline correction (it's a good idea) in ERP/ERMF analysis. J Neurosci Methods. [Epub ahead of print]


(This page was created by Makoto Miyakoshi and Andreas Widmann.)
20 Getting Started

EEGLAB is an interactive Matlab toolbox for processing continuous and event-related EEG, MEG and other electrophysiological data using independent component analysis (ICA), time/frequency analysis, and other methods including artifact rejection. First developed on Matlab 6.1 under Linux, EEGLAB runs on Matlab versions 6 (R13) and 7 (R14) under Linux or Unix, Windows, and Mac OS (Matlab v6 or current v7 recommended).

EEGLAB provides an interactive graphic user interface (gui) allowing users to flexibly and interactively process their high-density EEG and other dynamic brain data using independent component analysis (ICA) and/or time/frequency analysis (TFA), as well as standard averaging methods. EEGLAB also incorporates extensive tutorial and help windows, plus a command history function that eases users' transition from gui-based data exploration to building and running batch or custom data analysis scripts. EEGLAB offers a wealth of methods for visualizing and modeling event-related brain dynamics. For experienced Matlab users, EEGLAB offers a structured programming environment for storing, accessing, measuring, manipulating and visualizing event-related EEG, MEG, or other electrophysiological data. For creative research programmers and methods developers, EEGLAB offers an extensible, open-source platform through which they can share new methods with the world research community by contributing EEGLAB 'plug-in' functions that appear automatically in the EEGLAB menu. For example, EEGLAB is also being used for analysis of MEG data in several laboratories; EEGLAB plug-in functions might be created and released to perform specialized import/export, plotting and inverse source modeling for MEG data.

This tutorial will demonstrate (hands-on) how to use EEGLAB to interactively preprocess, analyze, and visualize the dynamics of event-related EEG, MEG, or other electrophysiological data by operating on the tutorial EEG dataset eeglab_data.set which you may download here (4Mb). With this dataset, you should be able to reproduce the sample actions discussed in the tutorial and get the same (or equivalent) results as shown in the many results figures.

EEGLAB functions may be roughly divided into three layers designed to increase ease-of-use for different types of users. These three layers are described below.

The EEGLAB gui is designed to allow non-experienced Matlab users to apply advanced signal processing techniques to their data. However, more experienced users can also use the gui to save time in writing custom and/or batch analysis scripts in Matlab by incorporating menu shortcuts and EEGLAB history functions.

Text files containing event and epoch information can be imported via the EEGLAB menu. The user can also use the menu to import event and epoch information in any of several file formats (Presentation, Neuroscan, ASCII text file), or can read event marker information from the binary EEG data file (as in, e.g., EGI, Neuroscan, and Snapmaster data formats). The menu then allows users to review, edit or transform the event and epoch information. Event information can be used to extract data epochs from continuous EEG data, select epochs from EEG data epochs, or to sort data trials to create ERP-image plots (Jung et al., 1999; Makeig et al., 1999). EEGLAB also provides functions to compute and visualize epoch and event statistics.

New Matlab users may choose to interact only with the main EEGLAB window menu, first to import data into EEGLAB (in any of several supported formats), and then to call any of a large number of available data processing and visualization functions by selecting main-window menu items organized under seven headings:

- **File** menu functions read/load and store/save datasets and studysets.
- **Edit** menu functions allow editing a dataset, changing its properties, reviewing and modifying its event and channel information structures.
- **Tools** menu functions extract epochs from continuous data (or sub-epochs from data epochs), perform frequency filtering, baseline removal, and ICA, and can assist the user in performing semi-automated artifact
data rejection based on a variety of statistical methods applied to activity in the raw electrode channels or their independent components.

- **Plot** menu functions allow users to visualize the data in a variety of formats, via (horizontally and vertically) scrolling displays or as trial (ERP), power spectrum, event-related time/frequency averages, etc. A large number of visualization functions are dedicated to the display and review of properties of scalp data channels and underlying independent data components. The user can make use of standard Matlab capabilities to edit, print, and/or save the resulting plots in a variety of formats.

- **Study** menu entries show the current studyset and which of its constituent datasets are currently loaded.

- **Datasets** menu entries list loaded datasets, and allows the user to switch back and forth among them.

- **Help** menu functions allow users to call up documentation on EEGLAB functions and data structures, including function lists and scrolling function help messages.

### 20.4.1.1

EEGLAB uses a single structure (EEG) to store data, acquisition parameters, events, channel locations, and epoch information as an EEGLAB dataset. This structure can also be accessed directly from the Matlab command line. The EEG structure contains two key sub-structures: EEG.chanlocs, holding channel locations, and EEG.event storing dataset event information. In EEGLAB v5.0 (March, 2006), a new superordinate structure, the **STUDY** has been introduced to allow automated processing of a group of EEG datasets. The first such facility introduced is a set of study functions to perform and evaluate clustering of similar independent data components across subjects, conditions, and sessions.

Intermediate level users may first use the menu to perform a series of data loading, processing and visualization functions, and then may take advantage of the EEGLAB command history functions to easily produce batch scripts for processing similar data sets. Every EEGLAB menu item calls a Matlab function that may also be called from the Matlab command line. These interactive functions, called "pop" functions, work in two modes. Called without (or in some cases with few) arguments, an interactive data-entry window pops up to allow input of additional parameters. Called with additional arguments, "pop" functions simply call the eponymous data processing function, without creating a pop-up window. When a "pop" function is called by the user by selecting a menu item in the main EEGLAB window, the function is called without additional parameters, bringing up its gui pop-up window to allow the user to enter computation parameters. When the processing function is called by EEGLAB, its function call is added as a command string to the EEGLAB session history variable. By copying history commands to the Matlab command line or embedding them in Matlab text scripts, users can easily apply actions taken during a gui-based EEGLAB session to a different data set. A comprehensive help message for each of the "pop" functions allows users to adapt the commands to new types of data.

### 20.4.1.1.1

More experienced Matlab users can take advantage of EEGLAB functions and dataset structures to perform computations directly on datasets using their own scripts that call EEGLAB and any other Matlab functions while referencing EEGLAB data structures. Most "pop_" functions describe above call signal processing functions. For example, the pop_erpimage() function calls signal processing and plotting function erpimage(). Since all the EEGLAB data processing functions are fully documented, they can be used directly. Experienced users should benefit from using all three modes of EEGLAB processing: gui-based, history-based, and autonomously scripted data analyses. Such users can take advantage of the data structure (EEG) in which EEGLAB datasets are stored. EEGLAB uses a single Matlab variable, a structure, EEG, that contains all dataset information and is always available at the Matlab command line. This variable can easily be used and/or modified to perform custom signal processing or visualizations. Finally, while EEGLAB "pop" functions (described above) assume that the data are stored in an EEG data structure, most EEGLAB signal processing functions accept standard Matlab array arguments. Thus, it is possible to bypass the EEGLAB interface and data structures entirely, and directly apply the signal processing functions to data matrices.

### 20.5

The EEGLAB toolbox is distributed under the GNU General Public License (for details see http://www.gnu.org/licenses/gpl.txt). The source code, together with web tutorials and function description help pages, is freely available for download from http://sccn.ucsd.edu/eeeglab/. The toolbox currently includes well over 300 Matlab functions comprising more than 50,000 lines of Matlab code. This user tutorial explains in detail how to import and process data using EEGLAB, including the derivation and evaluation of its independent components. SCCN also provides "Frequently Asked Questions (FAQ)" and "Known Bugs" web pages, a support email (eeglab@sccn.ucsd.edu), a dedicated mailing list for software updates (eeglabnews@sccn.ucsd.edu), and an email discussion mailing list (eeglablist@sccn.ucsd.edu), which currently reaches over two thousand EEG researchers.

Open-source EEGLAB functions are not precompiled; users can read and modify the source code of every function. Each EEGLAB function is also documented carefully using a standardized help-message format and each function argument is described in detail with links to related functions. We have attempted to follow recognized best practice in software design for developing EEGLAB. The source code of EEGLAB is extensively documented and is internally under the Linux revision control system (RCS), which allows us to easily collaborate with remote researchers on the
development of new functions. Matlab allows incremental design of functions, so adding new features to a function can be easily accomplished while preserving backward compatibility.

(Adapted from, A Delorme & S Makeig. EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics. *Journal of Neuroscience Methods* 134:9-21 (2004)).
21 Help EEGLAB

We need help to fix important issues for EEGLAB users.

21.1

Please post your scripts and data to our script library so other users may benefit from your experience.

Experienced users may also develop plugins. The whole process of creating a plugin is described in the plugin contribution page.

21.2

Below some projects posted by the EEGLAB developpers.

21.2.1

One of our main issue is currently the consistency of FFT and wavelet frequency decompositions. The log-spaced transform of the absolute power for FFT and Morlet wavelet do not match. If you are interested in trying to tackle this issue, please let us know. This issue is detailed in bug 661.

21.2.1.1

The FMR-IB plugin from Rami Niazi (for artifact removal) is not supported any more. If you want to take over maintaining this plugin, there are several issues listed in the Bugzilla database. You would maintain this plugin on your own server or on our servers through SVN.

21.3

This link will list all the improvements currently listed in Bugzilla. Feel free to take one on although you should post a comment on the improvement report so other developers are aware you are working on the problem.

Below any user may post a project that they would want to see implemented in EEGLAB. For small improvements and bug reports, use the EEGLAB bugzilla interface at http://sccn.ucsd.edu/eeglab/bugzilla.

21.3.1

This project was initially proposed in this email. Per Arno's suggestion we have created a page specifically for this project:

http://sccn.ucsd.edu/wiki/Interactive_Epoching_Project
22 How to download EEGLAB

This page describes how to download the development version of EEGLAB. This version contains the latest development. It is recommended for EEG advanced users.

22.1

EEGLAB download in ZIP format is available at EEGLAB download link. This includes the latest release as well as old versions.

22.2

Since the end of 2014, it is possible to use GIT to download the latest development version of EEGLAB from Github at any time - this version is more recent that the ZIP above. Because of our development scheme, the latest version of EEGLAB is usually the most stable. We recommend sourcetree to visualize branches. Simply clone the EEGLAB GitHub repository as you would do with any standard git package. When cloning make sure you use the the option --recurse-submodule (git clone --recurse-submodule git@github.com:eeglabdevelopers/eeglab.git) otherwise the firfilt plugin - which is the default filtering plugin - will not be downloaded and will not be functional.

22.3

The Master branch is a copy of the latest ZIP release. The develop branch is the latest stable code with updates and bug fixes. Most likely you will want to use this branch.

All other branches refer to previous version of EEGLAB or unstable code that is under intensive development. To update the EEGLAB code using the latest development sources, simply right click on the EEGLAB folder and select "git pull".

22.4

To contribute code to EEGLAB, fork the code and create a pull request as indicated on this page. This other page contains other information on how to contribute to EEGLAB.

22.5

EEGLAB was first under RCS (2002-2005), then under CVS (2005-2010), and finally under SVN (2010-2014) before migrating to GIT (2014-). All the revision message have been preserved in the migration process. It is however not possible to access the RCS, CVS, and SVN repositories since they refer to obsolete versions of EEGLAB.
23 How to edit EEGLAB code

EEGLAB is now under CVS (Concurrent Version System). Below are a couple of hints to help you check in and out eeglab functions. At this moment, only researchers with an account at SCCN may edit the EEGLAB code. If you want to contribute to the project and do not have a SCCN account, let us know at eeglab@sccn.ucsd.edu.

23.1

23.1.1

Then the simplest way to edit files is to use the Matlab editor and set the Matlab preference to CVS for revision control. As with RCS, you may check out any EEGLAB file, and then check it back in and enter your revision comment. It is that simple. The difference with our prior revision scheme however is that IF YOU DO NOT CHECK BACK IN YOUR CHANGES, THEY WILL BE ERASED OVERNIGHT (at 3:30 AM). Your changes will be saved in temporary files though. Be sure to change your settings so that other users can modify the file you save.

Files are erased overnight because the EEGLAB files at SCCN are just a checked out version of the real EEGLAB repository (which contains all history of changes and some fragmented Matlab code but not the actual Matlab files). There can be as many checked out version as we want (one in France, one in India, etc...) and all of them can be edited simultaneously. If two people modify the same file, and check them back in, then there is a conflict and it has to be resolved manually using special tools that allow to compare the files side by side with color highlighting of the changes.

23.1.1.1

Log onto SCCN and go to the folder you are interested in. You may just type

```
edit xxx.m
```

edit the file using xemacs. When you close the file, you may enter comment in a vi editing window. You must type "I" to insert comments and "ESC" then ":wq" to write and leave the comment editing window.

Alternatively you may edit the file directly

```
cd /data/common/matlab/eeglab/functions/sigprocfunc/
xemacs xxxx.m
```

```
save (but be careful that other can still edit your functions and have write permissions). Then type
```

```
setenv CVS_ROOT /data/common/cvs

cvs commit xxx.m
```

I think the first line is not even necessary as Bob said he would make that system wide. Then another vi pops up and you can enter comments. It's that simple.

23.1.1.2

If you want to work on a couple of functions for several days, the best is to check out the whole tree structure in your home folder (using dedicated software or Linux command) and then check back in your changes when you are done. Go to the folder where you want to check out the repository. Under the linux command line

```
cvs checkout eeglab
```

Bob set up CVS so the CVSROOT variable which tells CVS where the repository files are is set by default (equivalent to typing "setenv CVSROOT /data/common/cvs/"). To edit files with vi, do as recommended in the two previous sections.

To checkout files remotely (on your Linux laptop etc...), use the SSH protocol and any of the SCCN computers. Under Linux, you may type in:

```
setenv CVS_RSH "ssh"
cvs -d :ext:yourname@sccn.ucsd.edu:/data/common/cvs checkout eeglab
```

You may then edit and commit your changes as described in the previous section without the need to use different commands (in the version you checked out, there is a CVS folder that contains all the information about where the file are coming from etc...).
23.1.1.3

For instance, using tortoisecvs, you may simply check out the full EEGLAB tree, modify some files using Matlab (you may use or not the CVS feature of Matlab but in this case it is more of a burden as you would have to check in every single file). When you are done, right click on the eeglab folder and select Commit. This will commit all of your changes at once, and allow you to enter a comment that will apply to all of them.

Protocol: ssh
Protocol parameters:
Server: speaking.sccn.ucsd.edu
Port:
Repository folder: /data/common/cvs/
Username: arno

For username, you may also enter your username followed by a colon and your password. There are more secure ways to set that up of course. You may then press the Fetch list button and select EEGLAB. The checkout process will take a few minutes depending on the speed of your connection.

23.2

To create and add a new function. Simply move it to the folder where you want it to be in the EEGLAB SCCN folder (/data/common/matlab/eeglab/), open it with Matlab and check it in using the Matlab menu (File > check in). Local folders have CVS folders that refer to the repository so CVS knows at all time where you are in the EEGLAB repository tree structure. If you were to try that in your home folder, it would crash because your home folder is not associated with any checked out module of CVS.

If you are using tortoisecvs, copy or create a file within the EEGLAB structure. Then right click on it and select Add. Right click again and select Commit. Your file is now in the repository.

Arno Delorme - Jan 30th 2009
This is the 'Single subject data processing tutorial' Category Pages.

(Formerly known as the Main Tutorial)

The 'Single subject data processing tutorial' is a subcategory of the EEGLAB Tutorial
25 I. Single subject data processing tutorial

The EEGLAB Tutorial is split into three parts. The first part is the Main Tutorial. In the Main Tutorial, a user is led through how to set up EEGLAB on their computer and is introduced to basic EEGLAB functions. The user is instructed through a series of Key and Exploratory Steps to manipulate a sample data set (provided). This basic tutorial should enable the user to develop a good understanding of how to use EEGLAB for their own experiments. As of 02/4/2020, there are twelve topics in the Main Tutorial.

- Chapter 01: Loading Data in EEGLAB
- Chapter 02: Channel Locations
- Chapter 03: Plotting Channel Spectra and Maps
- Chapter 04: Preprocessing Tools
- Chapter 05: Extracting Data Epochs
- Chapter 06: Data Averaging
- Chapter 07: Selecting Data Epochs and Comparing
- Chapter 08: Plotting ERP images
- Chapter 09: Decomposing Data Using ICA
- Chapter 10: Working with ICA components
- Chapter 11: Time/Frequency decomposition
- Chapter 12: Multiple Datasets
26 Category: II. Multiple subject processing tutorial

These pages incorporate the Multiple Subject Data Processing Tutorial of the EEGLAB Tutorial. This is under the EEGLAB Category.
27 II. Multiple subject processing tutorial

The EEGLAB Tutorial is split into four parts. The second part is the Multiple Subject processing tutorial. In this tutorial, the user is introduced to more advanced elements of EEGLAB focused around the STUDY framework used for processing multiple subjects as once. These topics are recommended for the user to successfully analyse large scale experiments which go beyond single trials. As of 02/4/2020, there are eight topics in the Multi-subject processing tutorial.

- Chapter 01: Multiple Subject Processing Overview
- Chapter 02: STUDY Creation
- Chapter 03: Working with STUDY designs
- Chapter 04: STUDY Data Visualization Tools
- Chapter 05: Component Clustering Tools
- Chapter 06: Study Statistics and Visualization Options
- Chapter 07: EEGLAB Study Data Structures
- Chapter 08: Command line STUDY functions
28 Category:III. Advanced Topics

These pages incorporate the Advanced Topics of the EEGLAB Tutorial This is under the EEGLAB Category
29 III. Advanced Topics

The EEGLAB Tutorial is split into four parts. The third part is the Advanced Topics. In the Advanced Topics, the user is introduced to more advanced elements of EEGLAB such as script writing and artifact rejection. These topics are recommended for the user to successfully master EEGLAB. These sections are not as step by step as the single subject or multiple subject processing tutorials and should be used as a reference and not a manual. As of 02/4/2020, there are three topics in the Advanced Topics.

- Chapter 01: Rejecting Artifacts
- Chapter 02: Writing EEGLAB Scripts
- Chapter 03: Event Processing
31 Independent Component Clustering Example

31.1


Do different subjects performing the same task have equivalent independent EEG components (ICs)? In exchange for the benefits that ICA offers to EEG analysis in both spatial and temporal resolution of separable source-level activities, it also introduces a new level complexity into EEG analysis. In traditional scalp channel signal analysis, clustering of event-related EEG phenomena across subjects is straightforward, as each scalp electrode is assumed to be comparable with results from equivalently placed electrodes for the all subjects. Comparing ICA results across subjects, on the other hand, requires that, if possible, subject ICs from different subjects should likewise be grouped into clusters of ICs that are functionally equivalent despite differences in their scalp maps.

As we have seen, however, data recorded at a single scalp channel within each subject is heterogeneous, so the idea of grouping channel activity across subjects may actually be a risky proposition. In particular, clear physical differences between subjects in the locations and, particularly, the orientations of cortical gyri and sulci mean that even exactly equivalent cortical sources may project, across subjects, with varying relative strengths to any single scalp channel location, no matter how exactly reproduced across subjects. Thus, the basic assumption in nearly all EEG research, that activity at a given scalp location should be equivalent in every subject, is itself questionable. On the other hand, changing the basis of EEG evaluation from scalp channel recordings to IC activities necessitates an extra step compared to channel analysis -- that of combining and/or comparing results across subjects through identifying equivalent IC processes, if any, in their data.

Approaches to IC clustering. The process of identifying sets of equivalent ICs across subjects, or even across sessions from the same subject, can proceed in many ways depending on the measures and experimental questions of interest. An appealing approach to clustering ICs is by their scalp map characteristics. Such clustering can be attempted by eye, by correlation, or by an algorithm that searches for common features of IC scalp maps. The disadvantage of this method is that, as shown in Fig. 2, slight differences across subjects in the orientation of equivalent dipoles for a set of equivalent ICs can produce quite different IC scalp maps.

Clustering ICs based on the 3-D locations of their equivalent dipoles may avoid this problem. Using this method, it is possible to describe typical event-related or other activities in cortical areas of interest, or at least in cortical areas with sufficient density of IC equivalent dipoles across subjects or sessions. Common clustering algorithms such as K-means and other distance-based algorithms can be used to cluster ICs based on the 3-D locations of their equivalent dipoles quickly and easily. However, clustering on estimated cortical location alone may introduce similar confounds as clustering by scalp channel location, since subjects may have multiple types of IC processes in the same general cortical regions.

For one, comparing cortical locations across subjects raises the same spatial normalization questions as arise in functional magnetic resonance imaging (fMRI) analysis. Since brain shapes differ across subjects, true comparison of 3-D equivalent dipole locations should be performed only after spatially normalizing each set of subject IC locations to his or her normalized individual structural magnetic resonance (MR) brain image. This requires MR images be obtained for each EEG subject, a requirement that may greatly increase the resources required for EEG data acquisition.

A simpler method normalizes the 3-D equivalent dipole locations via normalizing the subject head shape, as learned from the recorded 3-D locations of the scalp electrodes, to a standard head model. When 3-D electrode location information is not available, the expected functional specificity of equivalent dipole clusters based on estimated equivalent dipole locations in a standard head model must be reduced. In this case, some IC processes estimated to be located in the same cortical area may not express the same functional activities. Despite this drawback, our results show that clustering component dipole locations in a standard spherical head model still allows meaningful conclusions about differences in regional EEG activities across one or more subject groups, if sufficient statistical testing is applied to the data, and if the limitations of the analysis are acknowledged.

Because homogeneity of an IC cluster is most accurately assessed and characterized by the activities of its constituent ICs, a more direct route to obtaining functionally consistent clusters may be to group ICs from experimental event-related studies according to their event-related activity patterns. For example, a recent EEG/fMRI study of Debener et al. (2005) clustered components contributing most strongly to the event-related negativity (ERN) feature of the average ERP time locked to incorrect button presses in a speeded choice manual response task. Remarkably, the authors showed that trial-to-trial variations in the strength of the activity underlying the ERN correlated with changes in the fMRI BOLD signal only in the immediate vicinity of the equivalent dipole source for the component cluster. In some cases, therefore, clustering ICs on similarities in their ERP contributions can be a simple but powerful approach to discovering sources of well-documented ERP peaks.
If the measure of primary interest is not the average ERP but, instead, event-related fluctuations in spectral power of the ongoing EEG across frequencies and latencies, as measured by average ERSPs, then component ERSP characteristics may similarly be used as a basis for IC clustering. Given a small number of subjects and a simple experimental design, it might be possible to group component ERSPs across subjects by eye, though this quickly becomes discouraging as the number of subjects and/or task conditions rise. In any case, an objective approach is more desirable.

**Figure:** Clustering ICs from 29 subjects by common properties of their mean event-related activity time courses can be an efficient method for finding homogeneous groups of independent processes across sessions or subjects. Here, ICs were clustered by similarities in 3-D dipole location as well as features of their mean ERSPs time-locked to auditory performance feedback signals within two task conditions (following Correct and Wrong button presses) and by the significant ERSP difference (when any) between them (this significance estimated by non-parametric binomial statistics, $p<1e-5$). Colored spheres show the locations of the equivalent dipoles for the clustered components. Colored lines connect these clusters to the respective cluster-mean ERSP and ERSP-difference images. Although IC equivalent dipole locations were only a portion of the data used in the clustering.
algorithm, the equivalent dipole models for four of the obtained clusters (A-D) are spatially distinct. Two other, spatially intermingled clusters (E-F) illustrate how activity-based clustering can differentiate spatially similar components that would not be separated in clustering based on location alone.

As an example, let us consider data from the same two-back task described earlier and illustrated for one subject. Assume there are 20 subjects, each with a mean of 15 dipolar cortical ICs. To prepare the data, each 2-D (latencies, frequencies) component ERSP image for one or more task conditions (correct, incorrect, etc.) must be concatenated and then reshaped into a 1-D (1, latencies × frequencies × conditions) vector. Thereafter, the vectorized component ERSPs from the 20 subjects can be concatenated to form a large 2-D matrix of size (#ICs, latencies × frequencies × conditions) or, in this case, (300, latencies × frequencies × conditions). A number of options are now available.

A simple approach is to use standard clustering algorithms such as K-means to cluster on Euclidean distances between the rows of the component matrix, whose dimensionality can be made manageable by preliminary PCA reduction. It is also possible to combine dissimilar IC activity and/or location measures in computing component “location” measures from which component-pair “distances” can be derived. The open source EEG analysis toolbox EEGLAB includes a clustering interface that implements this method. The ICACLUST facility enables component clustering across subjects or sessions using a variable set of IC features: ERPs, ERSPs, scalp maps, mean spectra, and/or equivalent dipole locations.

The figure above illustrates preliminary clustering results on 368 near-dipolar ICs from 29 subjects performing the two-back task described earlier. In the figure, equivalent dipoles of the same colors were clustered by computing a Euclidean ?distance? measure between the concatenated average component ERSPs time-locked to auditory feedback tones signaling ?correct? and ?wrong? responses, as well as the significant ERSP difference between the two, and also 3-D dipole location of each IC. The ERSPs plotted for each cluster represent the means over all the cluster components, after masking spectral perturbations not significant (p < 0.00001) by binomial probability across the set of clustered components.

Note that although IC equivalent dipole locations were here only a portion of the data used in the clustering, ICs with similar event-related activity patterns proved to be naturally associated with distinct cortical regions. From the ?Difference? activity the (A, light blue) central midline cluster, it is clear that this cortical area produced a different activity pattern following wrong responses, namely a 400-ms theta band burst that began before the auditory feedback during the period of the motor response. This result is in line with our previous findings (Luu et al., 2004) and neatly reproduces the recent result of Deben et al. (2005) who used time-domain analysis of simultaneous EEG and functional magnetic resonance imaging (fMRI) data to show that trial-to-trial variations in post-error activity of a very similarly located IC cluster were correlated with trial-to-trial variations in fMRI blood oxygen level-dependent (BOLD) signal only directly below the cortical projection of the component cluster, and highly coincident with the location of the equivalent dipole cluster in the figure above.

ERSP and equivalent dipole locations for three other activity-derived but spatially ?tight? component clusters are shown in the figure above. IC clusters located in/near left and right hand somatomotor cortex (B, yellow and C, magenta, respectively) exhibited significantly stronger alpha activity for a half-second after receiving Correct feedback than after receiving Wrong feedback, as confirmed by permutation-based statistical testing. Analysis of responses to matching versus non-matching letters (not shown) revealed the expected dominance of spectral perturbations contralateral to the actual response hand. Finally, event-related perturbations in spectral power in the bilateral occipital (D, green) IC cluster differed little following Correct versus Wrong feedback.

In contrast, the same clustering of ICs on dipole locations, condition ERSPs, and ERSP differences produced two more spatially diffuse and highly overlapping occipital dipole clusters (red and blue). Although the spatial distributions of these two clusters cannot be distinguished, the difference ERSPs between the two clusters do differ. Much like the left somatomotor (yellow) cluster, components in the blue cluster exhibit increased alpha activity following Correct auditory feedback. Of course, interpretation of the clustered component activities is a problem separate from clustering. However, once successful clustering of independent component activities has been accomplished, meaningful conclusions about brain function may be approached with more confidence.

Julie Onton & Scott Makeig,
SCCN/INC/UCSD, March 2006
32 Category: IV. Appendix

These are the Appendix Files of the EEGLAB Tutorial. The Appendix is a sub category of EEGLAB.
33 IV. Appendix

The EEGLAB Tutorial is split into four parts, the last of which is the Appendices. In the Appendices, the user is introduced to more advanced and technical elements of EEGLAB such as input data formats and Matlab data structures. These topics are presented to the user to fully describe the architecture, etc. of the EEGLAB system. These sections are a reference only; more information can be found by contacting the appropriate SCCN personnel. As of 02/4/2020, there are thirteen Topics in the Appendix.

- A01: Importing Continuous and Epoched Data
- A02: Importing Event Epoch Info
- A03: Importing Channel Locations
- A04: Exporting Data
- A05: Data Structures
- A06: EEGLAB option menu
- A07: Contributing to EEGLAB
- A08: DIPFIT
- A09: Using custom MRI from individual subjects
- A10: MI-clust
- A11: BESA (outdated)
- A12: Quick Tutorial on Rejection
- A13: Compiled EEGLAB
34 Mex files in EEGLAB

If you get an error "could not locate mex file" or any related mex file error, do not blame EEGLAB. EEGLAB itself does not include any precompiled functions (also called mex functions). However, some external modules use mex files for reading binary data file or performing source localization. Usually, an error indicates the precompiled file is not available for your platform (it would thus need to be recompiled, something you can sometimes do yourself - for more information see below). There are 4 modules in EEGLAB that use mex files.

- **Fieldtrip functions**: If you get an error that some functions in the fieldtrip folder cannot be found, refer to the [fieldtrip documentation](https://fieldtrip.fcdtu.ee/) for how to recompile such functions.

- **BIOSIG**: Some release of BIOSIG contains some updated mex files so you might want to check if you have the latest version of BIOSIG. Sometimes BIOSIG does not preserve backward compatibility so you may experience problem when reading data after updating BIOSIG.

- **ANT plugin**: The ANT plugin was made by the ANT company. Contact [Maarten](mailto:maarten@uu.nl) for an updated version of the compiled binaries.

- **ERPSS plugin**: Simply recompile decompresserpss.c (type "mex decompresserpss.c")
35 TIPS and FAQ

This FAQ page contains questions we receive and answers we give users, as well as general tips we think it is important for users to know! Many other tips are available on the eeglablist archive. To search the list archive simply use Google, enter relevant keyword and add the "eeglablist" keyword.
36

36.1

36.1.1

**Question:** Is EEGLAB of any interest for expert Matlab users?

**Answer:** We believe so. First, EEGLAB implements new algorithms for artifact rejection. Second, for the experienced Matlab user EEGLAB is a fast and accurate way to start processing EEG and ERP data and to directly manipulate signal arrays. If you intend to use the ICA toolbox functions underlying EEGLAB, EEGLAB itself is a good starting point and introduction. EEGLAB also provides a full EEG structure to describe your data (signal, trials, channel location, reaction time, type of the trials, time limits and sampling rate) and allows you use this structure either from the Matlab command line or in Matlab scripts.

**Question:** Is Matlab too slow and does it use too much memory ?

**Answer:** Yes, to an extent, but... Because Matlab is sometime too slow we implemented the most time consuming algorithms in C (for instance the runica() function). We also took great care of inserting options in EEGLAB and in several processing functions to handle low memory conditions. On the other hand the Matlab environment offers the advantage of stability and ease of use. Even the novice user under Matlab can scale a data array by multiplying it by a scalar for instance (and in our software the data array is directly accessible to the user). Matlab also offers the advantage of modularity. All of our function are stand-alone functions and most of them can be used independently of each other. Besides, Matlab has grown much faster and we hope may become more memory-efficient in future. For our purposes, this should include a 32-bit float data processing option (Is Matlab listening?)

**Question:** Support ?

**Answer:** We cannot guarantee that we will provide full support for this software but we will be glad to help out if someone encountered any problem and to correct bugs when reported to us: Write to eeglab@sccn.ucsd.edu.

36.1.2

**Question:** When I ask the Mathworks salesman to sell me "only" a Matlab license, he gave me prices for Matlab, Simulink and Symbolic Math. Do I need all these to run EEGLAB?

**Answer:** No, you do not need all that, you only need Matlab. If possible, ask for an educationnal or even student version, as they are cheaper. For some spectral decompositions, you may also need the Data Processing toolbox which has to be purchased separately. We are gradually attempting remove this dependency.

36.1.3

**Question:** under Unix, I often get the following message

```
Warning: One or more output arguments not assigned during call to 'XXX'.
??? Unable to find subsindex function for class char.
```

**Answer:** In most cases, this error indicates that Matlab on Unix may experience problems. Matlab might return this error when you or EEGLAB has defined a variable of the same name as any variable in your Matlab workspace or .m file in the Matlab path. To solve the problem, clear the variable or rename the function.

36.1.4

**Question:** Matlab/EEGLAB return an out of memory error
Answer: 3 solutions

- Buy more memory (RAM) for your computer
- Try the memory mapping scheme (in EEGLAB options) which will allow to keep the data on disk. Note that expect for Neuroscan files, it is still necessary to import the full data file in memory.
- Change the amount of memory Matlab or the memory management scheme.

36.1.5

Question: does Matlab/EEGLAB work on 64-bit platforms?

Answer: yes and there are 3 choices

- Mac OSX server (up to 32Gb of RAM). Preferred choice for instalation and ease of use.
- Linux 64-bit (Fedora Core or Redhat distribution). Preferred choice for high performance computing and clusters. Matlab instalation may be harder.

!!! However, it must be noted that there issues when using the 64-bit version of Ubuntu 8. It seems that there are issues with the Signal Processing Toolbox in Matlab (required for EEGLAB). Please refer to bug 768 for more information! [--Dev Sarma 21:48, 28 October 2009 (UTC)]

36.1.6

Question: does it benefit to have a multi-core machine?

Answer: yes, it benefits in two ways. First, you may start in parallel several Matlab session. Each of them is assigned one of the processor. Second, if you go to the Matlab options, you may have the option to enable multi-core computation (General > Multithreading). This option is usually set by default. This is a very efficient option that will speed up your code usually linearly with the number of core (2 -> twice faster etc...).

36.2

36.2.1

Question: is it possible with EEGLAB to save an EEG data sorted in 10 epochs (for example) in 10 ascii files?

Answer: you have to do this on the Matlab command line:

```matlab
>> epoch1 = EEG.data(:,:,1); >> save -ascii epoch1.txt epoch1 >> epoch2 = EEG.data(:,:,2); >> save -ascii epoch2.txt epoch2 >> epoch3 = EEG.data(:,:,3); >> save -ascii epoch3.txt epoch3
```

36.2.2

Question: I work with recordings from 30 channels + 2 EOG channels. Naturally while exploring the data for artifact components, I want to have a look at the EOG too. As soon as I want to plot maps, I liked to leave the EOG out. But if I want to load an electrode position file with less electrodes than channels in the dataset, EEGLAB doesn't accept that. Is there a more convenient method than creating two datasets (one without eog or without an electrode location file)?

Answer: Simply blank out all the position fields for these channels using the channel editor after you have imported the channel location file (so they do not have an assigned position). Note: You may still include these channels in the ICA decomposition, even if their reference is different from the other scalp channels, though you should not attempt to plot them with a mixed reference. (We are working on a solution to this for other purposes).
36.2.3

**Question:** EEGLAB does not work when I try select the type of data file to import under >File > Import data".

**Answer:** Next to the importing text box there is a list box to indicate which type of data you want to import. Note that, not only you need to scroll the the list box but also to CLICK on the selected importing options so that they become selected.

36.3

36.3.1

**Question:** We have been trying to input our polhemus 3-d file into EEGLAB. Displaying it in 2-d, the main problem is that "0 degrees" is towards the right ear in Polhemus and "90 degrees" is toward nasion, while in EEGLAB "0 degrees" is toward nasion and "90 degrees" is toward the right ear. In other words not only that everything is shifted 90 degrees, but Polhemus (Neuroscan) goes counter-clockwise and EEGLAB goes clockwise. We are trying to develop some conversion solution for this, but if

1. you already have some experience w/ a situation like this you may have some easy solution;
2. you could advice use to get our Polhemus coordinates into EEGLAB some other (easier) way, we would appreciate.

**Answer:** It actually depends on how you reccorded your Polhemus coordinates. To fix this problem, under EEGLAB, in the channel editing window, there is a button "tranform axes". Press this button and enter "theta = 90-theta;" and that will do the trick. Note: press the "auto shrink" button to visualize all your electrodes. You may also manipulate the XYZ coordinates and reconvert them to polar.

36.3.2

Remember that all EEGLAB graphic interface always process the current dataset. This means that if you use an interactive function (Data scrolling, Epoch rejection, Component rejection) and manually select or load another dataset from the EEGLAB interface, EEGLAB will apply all the changes or plot to the new dataset being scrolled. For instance, if you are scrolling data and select bad epochs, then load another data to look at it, then come back to your scrolling window to reject these epochs (and actually press the reject button), the epochs will be rejected in the New recently loaded dataset. We might implement a warning to prevent that.

36.3.3

**Question:** The maximum time does not correspond to the maximum time I specified. For instance I asked for epochs between 0 and 3 seconds at 125 Hz and end up with an interval of 0 to 2.992 s. There should be something wrong!

**Answer:** Nothing is wrong. In your example, we must draw (125Hz * 3seconds = 375 points) and not 376 otherwise we would loose time linearity i.e. 2 epochs of 3 seconds would be 752, whereas if we draw 6 seconds of the data we would get 751 points !), but if we assign time 0 to the first point, then we must assign time 2.992 to the last point. Actually, the first point time is undetermined. Since the recording is made at 125 Hz, there is no possibility to know when the first point was recorded in between 0 or 0.08 seconds (otherwise one has to sample faster). By convention we take the first point to be at latency 0 and the last one at 2.992 (but we could choose latency 0.04 for the first point and 2.996 for the last one or 0.08 to 3...).
36.3.4

**Question:** Could you help me to know why:
I have 7 events, 3 being = 2 and 4 being = 1 in a "mother" file
why, when I extracted epochs on event = 2, I only got 3 epochs (that's right) and 6 events (that's wrong, isn't it)?

**Answer:** Your 6 events are probably OK. It depends on the time window you used for epoching. Any event within the time window of each epoch will be taken into account. This means that some event outside of all epoch time window will be ignored and that some events within several time windows might be duplicated. You should look at the event latencies before and after epoching and you will see that it is consistent.

36.3.5

**Question:** I have a problem to filter data: we see in our EEG data a component with the frequency of the electronics here, 50 Hz. I would like to remove only this frequency... but I don't really know to do this with lowpass and high pass filtering. I think that this phenomenon of "pollution" by the electric frequency is something current in EEG?

**Answer:** Yes, it is very common.

**Question:** So I think I can do this filtering with EEGLAB?

**Answer:** Yes you can but you need the Matlab Signal Processing toolbox. What we usually do is that we run ICA with 'extended' option (in the option box for running ICA put "'extended', 1"). Then some components capture this noise which is independent of the brain activity.

**Question:** could you tell me witch other additional toolbox you used in which function?

**Answer:** You only need the Signal processing toolbox.

**Question:** Are there advanced filter functions for EEGLAB?

**Answer:** Check out firfilt plugin at EEGLAB Plugins#Other available plug-ins:
37

37.1

37.1.1

**Question:** When using channel ERP image ('Plot' - 'Channel ERP image'), the top part of the output is the window with the sorted trials. The number of the trials seems to be off and it appears that not all trials are displayed. For instance, using a file with 17 trials the 'Sorted Trials' numbering goes from 6 to 12 with apparently 5 trials being displayed and two half trials on the top and at the bottom of the display (see attached jpg). This was also observed with another file that had more trials.

**Answer:** The erpimage() output is due to the smoothing window you selected. If you use a 5-trial average moving window, the first output should be the middle of this window (e.g., trial 2.5). Use a 0-smoothing window (width 1) and all the trials will be shown.

37.1.2

**Question:** Is there any way to add lines above and below the average voltage line to indicate +/- 1 or 2 SD to show data dispersion across trials?

**Answer:** Use option 'erpstd' of erpimage() in the "More options" text box of the pop_erpimage() interactive window.

37.1.3

**Question:** Do you have any good way to across subject/patient analyses with respect to power and coherence, especially making statistical comparisons between subjects/patients? Could we somehow use the data output of EEGLAB?

**Answer:** For cross-subject comparison, tftopo() is a powerful function that can summarize this information (it uses stored timef() and crossfO() function outputs). It has to be called from the command line though.

37.2

37.2.1

**Question:** It is not very difficult to find components related to eyeblinks, etc. In my case, there are phases during the experiment, where people speak and/or move there eyes. I find it quite hard to determine which components are related to these artifacts and I already wonder if it is possible et all.

**Answer:** To determine which components are related to these artefacts, one approach is to isolate these trials (selecting them) and then use menu item "Plot > Component ERPs > With component maps" and select the time window where these event appears. This function will plot which components contribute to this type of artifact. However, you are correct in thinking that ICA cannot cleanly resolve ALL artifacts into one or a few components. For instance, "paroxysmal" artifacts (like the subject scratching their head during recording) would require a large number of ICA components to capture the variability of all the artifactual contributions in the data. Similar events and artifacts should be carefully pruned from the data before ICA decomposition.

**Question:** Would muscle components be seen at high frequencies?

**Answer:** Yes, that is typically what we observe. Muscle artifacts have a broadband high spectral amplitude at high frequencies.
frequencies (20-100 Hz or more). Also their spectrum does not look like the standard EEG (exponential decrease).

37.2.2

**Question:** I am currently using ICA to correct for artifacts. In the past I’ve visually inspected each single-trial epoch separately, identified those trials with artifact activity, and then trained ICA on each trial separately to identify and remove artifactual components. As, you can imagine this process is extremely time consuming. Is it effective to train ICA on multiple or all concatenated trials at once, remove artifactual components, and then go back to visually inspect the corrected data for outlying artifacts?

**Answer:** You may not have grasped the nature of ICA, which is to extract data components whose activities (activations) are independent of the activities of other data sources. For this independence to be detected, it must be fully *expressed in the data* -- and this normally requires many (>nchannels^2) data points. So separately decomposing individual epoch is just the wrong approach. We currently favor a seven-step approach:

1. Visually reject unsuitable (e.g., paroxysmal) portions of the continuous data.
2. Separate the data into suitable short data epochs.
3. Perform ICA to derive independent components.
4. Perform rejection on the derived components based on inspection of their properties.
5. Visually inspect the raw data epochs selecting some for further rejection.
6. Reject the marked components and data epochs.
7. Perform ICA a second time on the pruned data - this may improve the quality of the ICA decomposition, revealing more independent components accounting for neural, as opposed to mixed artifactual activity.

These steps are made easier and more efficient by EEGLAB, which also includes functions suggesting to the user epochs, channels and components suitable for rejection.

37.3

37.3.1

**Question:** While trying within EEGLAB to remove artifacts using ICA, I had trouble in recalculating an ICA decomposition after removing components. I tried to follow the guidelines in the tutorial, thinking that with fuzzy components it might work better to remove some clear artifact components first and run a new ICA. When I tried to do that, the second ICA always took much longer and I got also some error message in the end telling me, that there was something wrong with the result.

**Answer:** The standard procedure we advise is first to perform ICA on the data and to remove bad trials using the ICA component activities. If you remove ICA components, the rank of the data will decrease (to <nchans). If the data have n channels, the rank of the data is (most probably) n. If you remove one component it will become n-1, and ICA will not be able to find n components in the pruned data. Thus, as a first step, you should only remove bad trials. This procedure will not alter the dimensionality of the data. As a second step, recompute ICA and remove bad components (the second run of ICA should result in clearer artifact components (for instance muscle at high frequencies), not contaminated by strong outlier trials. If you remove ICA components and want to re-run ICA, you must decompose the data with the ‘pca’ option to reduce the dimensionality of the decomposition to match the data rank (see below).

37.3.2

**Question:** What is the rationale behind baseline zero'ing the data before running it thru ICA, and is that always recommended?

**Answer:** It is recommended because the EEG might have some electrical artifacts (slow trends) that you want to remove. If your data is perfectly flat (at very low frequencies), then you shouldn't need to do that. You should baseline-zero each
epoch, else use the continuous data and lowpass it if you are interested in focusing on e.g. 3-40 Hz activity. Also, you should prune the data of 'messy' patches (probably associated with a series of 1-of-a-kind, non-stationary maps). The data scrolling utility in EEGLAB makes this convenient, and if you perform this on the continuous data, records breakpoint events that guide subsequent epoching. Else, you can more severely prune the data to train the ICA model, then pass more of the data through the model (at a cost of somewhat higher SNR (signal to noise ratio) in the activation time series).

37.3.3

**Question:** I don't know how to make the independent component dimension reduced by PCA in EEGLab4.0. For example, I want to obtain 4 independent components from 32-channel EEG data.

**Answer:** To extract 4 components, in the second text box of the "Tools > Run ICA" interactive pop-up window, enter "'pca', 4 " or "'ncomps', 4" and that will do it.

Note we do not recommend using PCA ("'pca', 4") unless you have some good reason. Using first PCA components only will truncate the data (irrespective of components), and then ICA may not be able to find relevant components. The second possibility ("'ncomps', 4") is more acceptable theoretically since it is a true ICA decomposition (that uses a rectangular matrix). In general, we advise finding as many components as possible (e.g. if you have enough memory on your computer to run ICA over all the channels).

37.3.4

**Question:** you mentioned that we should use the extended ICA algorithm to extract subgaussian components (i.e., 60-Hz noise). For our MEG data (using the old binica() for Windows) we get several subgaussian components, but there is still leakage of 60 Hz onto many of the supergaussian components as observed with an FFT (multitaper). In the EEGLAB tutorial (1st_readme.txt) I noticed a question mark on whether binica for Windows (the old one) was stable. Is it unstable? Should we use the extended version? Can we use the old binica() for Windows?

**Answer:** Matlab seems to have speeded up running runica() tenfold from 5.3 to 6.x! So binica() just gives us a 30% improvement in speed these days (although also a 2-4-fold decrease in process size, important for large datasets under 32-bit memory addressing). In the future, we will compile the improved version of binica() for Windows. The improvements concerned mainly the 'pca' option, which you may not need to use.

37.3.5

**Question:** am I right in assuming that binica() only runs under unix? I get the following error message...

```
binica: using source file 'C:\MATLAB6p5\toolbox\eeglab4.08\binica.sc'
binica: using binary ica file
'C:\MATLAB6p5\toolbox\eeglab4.08\ica_linux2.4'
Running ica from script file binica4411.sc
'ica_linux2.4' is not recognized as an internal or external command,
operable program or batch file.
??? Error using ==> floatread
floatread() fopen() error.
Error in ==> C:\MATLAB6p5\toolbox\eeglab4.08\binica.m
On line 265 ==> wts = floatread(weightsfile,[ncomps Inf]);
```

**Answer:** Yes, binica() only runs under unix (though there is a C executable for windows). I think you have not declared the path that contain the ica_linux2.4 in your unix PATH variable (in .cshrc). This is the reason why the Matlab function cannot execute the binary file.
37.3.6

**Question:** I have noticed that the runica() does not include a field for epoch size, so how does ICA recognize the epochs? Doesn't this make a difference to the way ICA is handled?

**Answer:** Epochs are concatenated before running ICA. In ICA, all time points of all epochs are shuffled so that epoch information is irrelevant.

**Question:** I have some continuous 32-channel EEG data on which I would like to apply Infomax ICA. I am primarily interested in the 100 epochs from the data, which are 3000 frames each. There are only about 20-40 frames between the end of one epoch and the beginning of the next. Should I apply ICA to the continuous data, then epoch the ICs, or apply ICA to the concatenated epochs?

**Answer:** You can apply ICA to either of them. Usually, we prefer to apply ICA to the concatenated epochs so ICA component are more likely to represent activity related to the task, but continuous data are fine too, especially if you have few epochs or few data points, since most of the same EEG and artifact processes are likely to be active 'between' epochs.

37.3.7

**Question:** For multiple-epoch data, the scalp map obtained for the different epochs is the same for a particular component. Is this normal or is there some mistake that's being done in the analysis?

**Answer:** Because the ICA algorithm is applied in the electrode space domain, the same scalp maps are returned for all epochs. However the time course of one ICA component is different for each epoch (if its activation value is 0 at a given time, it means that this component is not expressed in the data at that particular time).

37.4

37.4.1

**Question:** Is it not true that the ERP for a condition can be completely reconstructed from the timef() results, incuding ITC? One could write a function that takes the outputs of timef ('times','freqs','ersp','itc') and gives as it's output the ERP. Could one then usefully manipulate the values in ERSP and ITC and see how the ERP would have looked without some aspect of the ERSP (like, take away the inter-trial coherence in the alpha band and leave all else the same). Finally, could one subtract the 'ersp' and 'itc' between two conditions and then reconstruct the difference ERP?

I've been playing with the parameters of timef() and am surprised by the large differences in results dependent on the values for 'cycles', 'winsize', etc. This even makes we insecure about previous timef() results ....

**Answer:** Yes, this is possible (with some fuzziness regarding overlap-adding the overlapping spectral estimates, undoing the effects of tapering (windowing), etc. I havent focused on strictly "invertible" time/frequency transforms - which tend to be restrictive, since I am interested in analysis rather than synthesis.

However, in general the answers will be like the following:

Removing alpha ITC would be like filtering out the alpha in single trials and replacing it with random-phase noise. The effect on the ERP would be more or less exactly like filtering out the 10 Hz from the ERP, plus adding some noisiness...

The ERSP is a different matter, as ERSP changes can only produce a (stat. signif) ERP in conjunction with signif ITC. Adding an ampl. increase at alpha, say, to the ERSP and then going back to the time domain would scale up alpha band power in the ERP -- if the ITC stays the same.

Changing both ERSP and ITC can be made to give any kind of mixed results summing those above.
Re timef() variability - think of it like the focus / f-stops on a camera - you cannot achieve focus on every plane at once. It is also like Heisenbug uncertainty, I hear - you cannot localize a 'particle' (wavelet) in time and frequency simultaneously. Or, as in cinema you, where one cannot record motion in a single instant, one cannot record an exact power estimate at an instant. Basically, a "good" time/frequency method can only give an estimate within a time/frequency area whose area is fixed, but not its shape. i.e. Do you want long-and-thin, or fat-and-short, or square?

So, one can make exact comparisons only across channels or components within the same t/f transform.

37.4.2

**Question:** I am afraid that the time scale is slightly off for the time-frequency plots. E.g. a component time-frequency plot from a dataset where the epochs are from -2560 ms to +2046 ms and according to the plot's time scale it appears that the epoch is from slightly before -2000 ms to slightly over +1500 ms. Do you know why?

**Answer:** It is normal that the time limits are different from the original dataset, since the FFT (or wavelet) is applied over time windows and we consider the center of these windows. As a result, you loose half the window size on each edge of the plot (some hundred milliseconds depending on the window size and the sampling rate).

37.4.3

You should most likely use the pwelch method (implemented in the EEGLAB spectopo function). It is a windowed FFT (several FFT averaged).

The Thomson method (usually known as multitaper) is good too. It is first projecting the data onto an orthogonal base, then performing FFT.

They should all return similar results (FFT, pwelch, multitaper). I guess the Thomson method is the less sensitive to noise but also the most complex to use. I guess it would also be possible to use the welch method on top of multitaper. It is all a matter of preference. I would advised using the pwelch method which is easy (you just give as an option the length of the windows and the overlap). Multitaper would require you to select the number of basis vector in your orthogonal base and this is much less intuitive (and also has consequences on the frequency resolution you can acheive).

37.4.4

**Question:** I have been using the new EELAB toolbox for the past couple of weeks, especially timef() and crossf(). The multitaper method with bootstrap statistics has been giving me very nice stable results. Timef() with wavelets gives slightly different results, but also interesting. I noticed though that all the analysis has been designed to study coherence, phase-coherence, ITC, etc, for data organized as epochs. (e.g. inter-trial effects).

Is there a function for computing time-varying coherence between 'independent' activation functions for continous spontaneous recordings? (i.e., spontaneous coherence for brief windows of time, 200-300ms). J.P. Lachaux (from Varela's Lab) has several papers investigating this issue.

**Answer:** We also programed multitaper methods for timef(), but removed them from the current more flexible versions (not available yet on the Internet (May 23, 2003)). Note that we usually use neither the FFT (0) or N-cycle wavelet methods, but a compromise (0.5) setting that trades off frequency resolution (at low freqs) with stability at high freqs.

We have not yet had cause to program flexible visualization and handling of continuous coherence measures, though you can run timef()/crossf() on a continuous dataset (which is internally considered by EEGLAB to be 1 epoch). We'd welcome suggestions and/or code.

In our experience, multitaper decomposition are not very useful because we are interested in low frequency activities (i.e. 5 Hz) at which we want the maximum time resolution. So we had to use 1 multitaper only (using several
multitapers reduces noise but also reduces the time/frequency resolution) which is equivalent to a standard FFT. Using several multitapers in the gamma band would make sense though because the time resolution is much higher and not critical. Multitaper has been removed from the newtimef() function but it is still possible to use it in the timef() function.

37.4.5

**Question:** We normally display our chrono-spectrograms with the lower frequencies at the bottom of the plot, just the opposite how Eagle does it. Is there an easy way to change this?

**Answer:** Right now the only way to change this is to click on the figure and type

```matlab
>> set(gca, 'ydir', 'normal');
```

I guess you could edit the timef() and crossf() Matlab functions to change all the plots to this format. We will introduce this as an option in the future.

37.5

37.5.1

**Question:** How can I sort single trial ERP in erpimage() based on their amplitude at a determined latency,

**Answer:** use the 'ampsort' option of erpimage() (not separately queried in GUI, you have to put it into the "More options" text box at the bottom right of the pop_erpimage() interactive window). Erpimage() option 'ampsort' sorts on spectral amplitude; if you want to sort on potential value at some epoch time point, then use option 'valsort'.
38

38.1

**Question:** For reasons too complex to describe, I am trying to epoch with respect to various latencies RELATIVE to the events in my dataset. For example, epoch onset = event latency + some value, so I can calculate the set of latencies and use them in a call to epoch.m (the second parameter). I couldn't specify the latencies directly using pop_epoch function.

**Answer:** There are 2 solutions for this problem:

1) modify event latencies or create new events. For instance

```matlab
index = :EEG.
    EEG.index. = EEG.index. + ;
;
ALLEEG EEG CURRENTSET = eeg_storeALLEEG, EEG, CURRENTSET;
```

2) use the epoch() function instead of the pop_epoch() function. The epoch() function processes latencies directly. You will then obtain a Matlab array that you may import in EEGLAB.

38.2

38.2.1

**Question:** The lines in erpimage() (e.g. 'X' axis on the average) are too thick. How can I control their thickness?

**Answer:** To change the figure aspect for publication, you can go in the figure menu and use the Matlab menu item "Tools > Edit". Then you can select any object in the figure. Second button will display a contextual menu where you will be able to change line thickness, color, font aspects,... or even draw additional lines or add text. We also export figures as Postcript files and open them with Adobe Illustrator in vector format to fine tune it. See also this web page.

38.2.2

**Question:** Could you tell me how to export figures from EEGLAB (to include in a "word.doc" for example, or to export to Excel)? This for the scroll channel data eegplot(), and for channel spectra and maps...and other plots....

**Answer:** For most EEGLAB figures, simply use menu item "File > Export".

For the scrolling channel data function (eegplot()), first use menu item "Figure > Edit figure" to restore the default Matlab menu. From the command line, you may also use the command

```bash
>>' print -djpeg scrollfigure.jpg  %to print in jpeg
>>' print -depsc scrollfigure.eps  %to print in postcript color
```

Note: use the command ">>' set(gcf, 'paperpositionmode', 'auto')" first to print the figure in the same aspect ratio as it is shown on screen.
38.2.3

**Question:** When I tried to plot the channel spectrum, the axis labels and tick labels did not appear clearly on the screen and I've got the following error messages from the MATLAB command line: "Plotting scalp distributions: Warning: Unrecognized OpenGL version, defaulting to 1.0"

**Answer:** You may solve the problem by changing the OpenGL version on the MATLAB command line by typing: "feature('UseGenericOpenGL',1)".

**Question:** Sometimes Matlab crashes when I try to print a figure. If I save the figure on disk, first, some parts are missing. Do you know how to fix this problem?

**Answer:** You may solve the problem by changing the OpenGL version on the MATLAB command line by typing: "feature('UseGenericOpenGL',1)". For the printing error, we also experience this; it is a Matlab problem which is not consistent between Windows and Linux. We always print or save to files (.jpg or .eps Postscript), then print the files. For instance, use the software FreeRawPrint to send the postscript file to the printer under windows. Even with this strategy, some parts of complex figures may disappear, but this is rare (then we use screen captures, or use a Windows machine, since printing seems to be more reliable under Windows OS). We hope Matlab will become better at this in the future (Is Matlab listening?).

38.3

- Archive of the eeglablist email discussion list
- FAQ collection from the old ICA toolbox release
- Search engine on the EEGLAB main page