ELSEVIER

Journal of Neuroscience Methods 134 (2004) 9–21

JOURNAL OF NEUROSCIENCE METHODS

www.elsevier.com/locate/jneumeth

EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis

Arnaud Delorme*, Scott Makeig

Swartz Center for Computational Neuroscience, Institute for Neural Computation, University of California San Diego, La Jolla, CA 92093-0961, USA

Received 17 June 2003; received in revised form 22 September 2003; accepted 16 October 2003

Abstract

We have developed a toolbox and graphic user interface, EEGLAB, running under the crossplatform MATLAB environment (The Mathworks, Inc.) for processing collections of single-trial and/or averaged EEG data of any number of channels. Available functions include EEG data, channel and event information importing, data visualization (scrolling, scalp map and dipole model plotting, plus multi-trial ERP-image plots), preprocessing (including artifact rejection, filtering, epoch selection, and averaging), independent component analysis (ICA) and time/frequency decompositions including channel and component cross-coherence supported by bootstrap statistical methods based on data resampling. EEGLAB functions are organized into three layers. Top-layer functions allow users to interact with the data through the graphic interface without needing to use MATLAB syntax. Menu options allow users to tune the behavior of EEGLAB to available memory. Middle-layer functions allow users to customize data processing using command history and interactive 'pop' functions. Experienced MATLAB users can use EEGLAB data structures and stand-alone signal processing functions to write custom and/or batch analysis scripts. Extensive function help and tutorial information are included. A 'plug-in' facility allows easy incorporation of new EEG modules into the main menu. EEGLAB is freely available (http://www.sccn.ucsd.edu/eeglab/) under the GNU public license for noncommercial use and open source development, together with sample data, user tutorial and extensive documentation.
© 2003 Elsevier B.V. All rights reserved.

Keywords: EEG; ICA; ERP; Spectral decomposition; Single-trial; Matlab; Software

1. Introduction

Though computing capabilities of nearly every electrophysiology laboratory are now sufficient to allow advanced signal processing of biophysical signals including high-density electroencephalographic (EEG) recordings, many researchers continue to rely on amplitude and latency measures of peaks in EEG trial averages, termed event related potentials (ERPs). Historically, the response averaging method was developed under technical constraints imposed by hardware initially available for psychophysiological experiments in 1950s and 1960s. Before digital computers were available, researchers had to find a way to summarize event-related activity across several EEG trials representing brain responses to sensory stimulations. For this purpose, they first used analog registers to sum activity across EEG data trials. The first computerized response averaging computer, the computer of average transients (CAT, ca. 1962) helped promote the use of response averaging, called at first sensory 'evoked potentials' (EPs) and later the sensory/cognitive 'event-related potentials' (ERPs).

Using the fast and low-cost digital computers now available, technical limitations that constrained researchers to confine their EEG data analysis to simple ERP measures and parametric statistics are no longer relevant. The rationale used to justify response averaging is that the single-trial EEG data time locked to some class of experimental events consists of an average ERP, whose time course and polarity is fixed across the trials, plus other EEG processes whose time courses are completely unaffected by the experiment events. The cortical sources of ERP features may be assumed to be spatially distinct from sources of spontaneous EEG activities. However, as we have demonstrated recently, focusing data analysis on response averages alone ignores, first, event-related dynamics that do not appear in, or are poorly

* Corresponding author. Tel.: +1-858-458-1927; fax: +1-858-458-1847.
E-mail address: arno@sccn.ucsd.edu (A. Delorme).
URL: http://sccn.ucsd.edu/.

0165-0270/$ – see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/j.jneumeth.2003.10.009

represented in response averages, and second, ignores ongoing EEG processes that may be partially time and phase locked by experimental events, thereby contributing portions of response averages (Delorme et al., 2002; Makeig et al., 2002).

In the past decades, pioneer researchers have tried to apply to EEG data analysis techniques developed in electrical engineering and information theory, including time/frequency analysis (Bressler and Freeman, 1980; Makeig, 1993; Neuenschwander and Varela, 1993; Pfurtscheller and Aranibar, 1979; Tallon-Baudry et al., 1996; Weiss and Rappelsberger, 1996) and Independent Component Analysis (ICA) (Jung et al., 2001; Makeig et al., 1996, 1997, 1999). These techniques have revealed EEG processes whose dynamic characteristics are also correlated with behavioral changes, though they cannot be seen in the averaged ERP. For example, short-term changes in spectral properties of the ongoing EEG in specific frequency bands may be correlated with cognitive processes, e.g. expectancy of a target stimulus (Makeig et al., 1999) and with visual awareness (Rodriguez et al., 1999). The sufficiency of studying average ERPs has also been questioned by Makeig et al. (2002), who showed that some average ERP peaks may result from partial synchronization of oscillatory EEG processes to time locking events in single data trials.

Currently, most EEG researchers still interpret their data by measuring peaks in event-locked ERP averages. Free availability of more general and easy-to-use signal processing software for EEG data may encourage the wider adoption of more inclusive approaches. Our EEGLAB software toolbox for Matlab (freely available from http://www.sccn.ucsd.edu/eeglab/) allows processing of collections of single EEG data epochs using ICA and spectral analysis as well as data averaging techniques. Using this toolbox, we have demonstrated the advantages of combining ICA, time-frequency analysis, and multi-trial visualization in several publications (e.g., Delorme and Makeig, 2003; Delorme et al., 2002; Makeig et al., 1999, 2002). In EEGLAB, all these functions are available under a common graphic interface under Matlab, a widely used multi-platform computing environment. EEGLAB extends the collection of publicly available Matlab packages for brain imaging including SPM (Friston, 1995) and FRMLAB (Duann et al., 2002a) for functional MRI studies and Brainstorm (Baillet et al., 1999) for EEG/MEG source analysis.

2. Methods and results

2.1. Basic functions

The ICA/EEG toolbox of Makeig et al. (1997) included a collection of Matlab functions for signal processing and visualization of EEG data including *runica()*, a function for automated infomax ICA decomposition (Makeig et al., 1996, 1997), ERP-image plotting (Jung et al., 1999; Makeig et al., 1999), a method of visualizing time-locked potential variations across sets of single trials, and time-frequency decomposition (Makeig, 1993). By 2002, over 5000 researchers from over 50 countries had downloaded the ICA/EEG toolbox. However the provided tools could only be used for EEG analysis by knowledgeable users who were prepared to write custom data analysis scripts. EEGLAB, by contrast, includes a comprehensive graphic user interface for interactively calling and viewing results of enhanced and extended ICA/EEG toolbox functions while further facilitating the development of custom analysis scripts by prepared users. Fig. 1 shows a screen capture of an EEGLAB user session running under Linux.

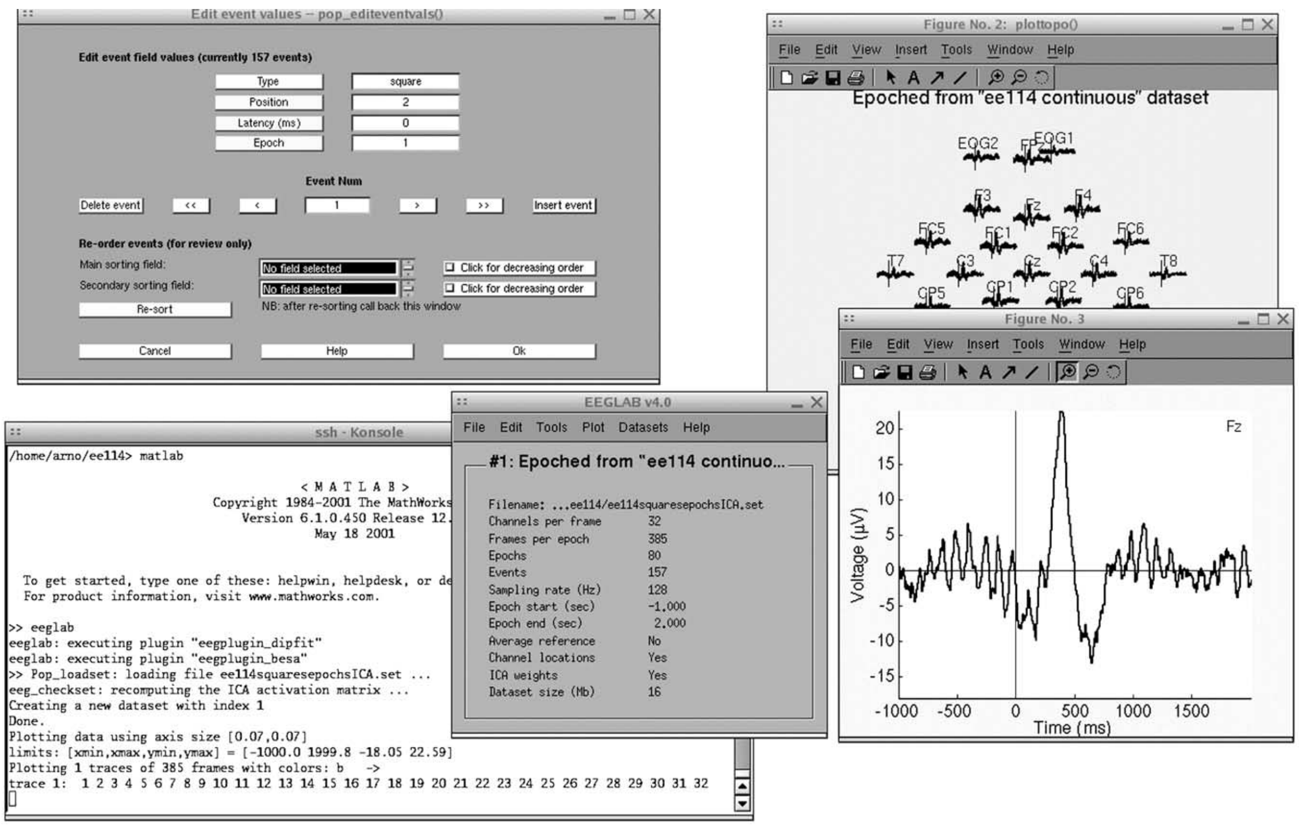
2.1.1. Data preprocessing

EEGLAB allows reading of data, event information, and channel location files in several different formats including binary, Matlab, ASCII, Neuroscan, EGI, Snapmaster, European standard BDF, and Biosemi EDF. Standard data analysis functions available in EEGLAB include data filtering, data epoch extraction, baseline removal, average reference conversion, data resampling and extraction of data epochs time locked to specified experimental events from continuous or epoched data. EEGLAB also includes methods allowing users to remove data channels, epochs, and/or components dominated by non-neural artifacts, by accepting or rejecting visually-cued EEGLAB recommendations derived from signal processing and information measures. EEG scalp maps and channel locations can be converted between several widely-used Cartesian, polar and spherical coordinate systems and then visualized in two or three dimensions. Continuous data and data epochs of any number of channels can also be scrolled (both vertically and horizontally).

2.1.2. Data structures and events

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. 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.

To illustrate the utility of EEGLAB, below we employ a small set of EEG data trials (also available from http://www.sccn.ucsd.edu/eeglab/) drawn from an experiment



Edit event field values (currently 157 events)

Type	square
Position	2
Latency (ms)	0
Epoch	1

Event Num: 1

Re-order events (for review only)

Main sorting field: No field selected ☐ Click for decreasing order
Secondary sorting field: No field selected ☐ Click for decreasing order
NB: after re-sorting call back this window

#1: Epoched from "ee114 continuo...

```
Filename: ...ee114/ee114squaresepochsICA.set
Channels per frame        32
Frames per epoch          385
Epochs                    80
Events                    157
Sampling rate (Hz)        128
Epoch start (sec)         -1.000
Epoch end (sec)            2.000
Average reference         No
Channel locations         Yes
ICA weights               Yes
Dataset size (Mb)         16
```

```
/home/arno/ee114> matlab

                                   < M A T L A B >
                       Copyright 1984-2001 The MathWorks
                          Version 6.1.0.450 Release 12.
                                    May 18 2001


  To get started, type one of these: helpwin, helpdesk, or de
  For product information, visit www.mathworks.com.

>> eeglab
eeglab: executing plugin "eegplugin_dipfit"
eeglab: executing plugin "eegplugin_besa"
>> Pop_loadset: loading file ee114squaresepochsICA.set ...
eeg_checkset: recomputing the ICA activation matrix ...
Creating a new dataset with index 1
Done.
Plotting data using axis size [0.07,0.07]
limits: [xmin,xmax,ymin,ymax] = [-1000.0 1999.8 -18.05 22.59]
Plotting 1 traces of 385 frames with colors: b   ->
trace 1:  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
```

Fig. 1. Sample EEGLAB session. Screen capture of an EEGLAB user session running under Linux. Users call EEGLAB functions from the main window (center) via 'pop-up' parameter selection windows (upper left). Warnings and data processing messages are shown in the Matlab command line window (lower left), which can also be used to call EEGLAB or other data processing functions directly.

in which the subject covertly attended a cued location on the computer screen, responding quickly with a thumb button press each time a target (filled square) was briefly presented at this location (Makeig et al., 1999). In different trial blocks, the attended location was any one of five positions arranged horizontally on the computer screen above a fixation cross. The sample dataset consists of 80 3-s EEG epochs time-locked to targets presented in the left visual field between 3° and 1.5° of visual angle. Data from thirty-one scalp electrodes (referred to the right mastoid) were sampled at 500 Hz (later reduced for compactness to 125 Hz). Fig. 2 shows five sample data epochs and illustrates the capabilities of *eegplot()*, the EEGLAB data scrolling function.

2.2. Multi-trial visualization

2.2.1. ERP-image plotting

The field of electrophysiological data analysis has been dominated by analysis of one-dimensional averaged event-related potential (ERP) time series (single channel values across latencies). The ERP-image is a more general two-dimensional representation of the data (single channel values within epochs across latencies) sorted in order of some relevant measure (e.g., collection time, subject response, amplitude or phase, etc.). Fig. 3(A) illustrates the process of constructing ERP-image plots. An ERP image is a colored rectangular image in which each horizontal line represents a potential time series during a single experimental trial. Instead of plotting activity in single trials as left-to-right traces in which potential is encoded by the ordinate of a data trace, trials are represented as horizontal lines whose changing color values indicate the potential at each time point in the trial. Trials may be plotted in any sorting order of interest, and a moving average across adjacent single trials may be used to highlight trial-to-trial consistency. Fig. 3(B) illustrates the process of sorting the data trials by the subject reaction time.

Some features of the visual ERP may be produced by partial phase-resetting of ongoing EEG activities following stimulus presentation (Makeig et al., 2002). Fig. 3(C) illustrates a phase-sorted ERP-image plot, a visualization tool used to assess whether partial phase synchronization may account for ERP features. Sorting by value or spectral amplitude in a given time window, or by an auxiliary variable are also supported. The *erpimage()* function can also plot the response average ERP, changes in signal power and inter-trial coherence (as defined below) at a selected frequency, the mean signal spectrum, and a representative scalp topography.

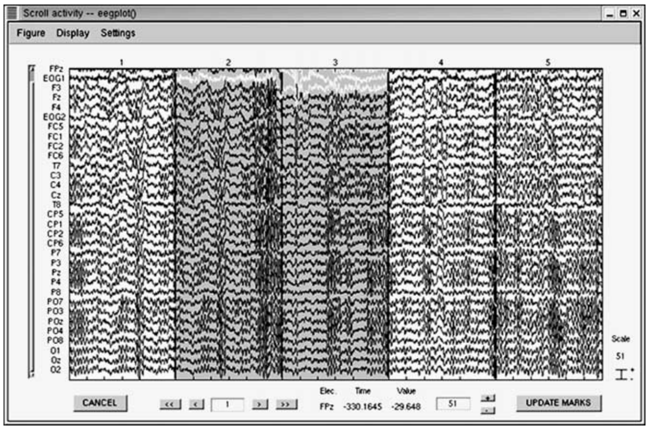


Fig. 2. Data scrolling. The EEGLAB scrolling data review function, *eegplot()*, allows the user to review and reject data by visual inspection. Here, five data epochs (separated by dashed lines) are plotted at 31 electrode sites (channel names on the left). Other channels in the dataset can be accessed using the vertical slider on the left. The arrow buttons (lower left) scroll horizontally through the data. The user may zoom in on a selected time range and/or electrode group and may change the plotting parameters using menu options (upper left). Values of the data point closest to the cursor are continuously displayed at the bottom of the display. In this example, two (central) data epochs have been automatically marked for rejection for out-of-bounds values set by the user in the EEGLAB data-rejection pop-up window (not shown). The rejection routine here highlights (in white) the channels containing the outlier values. The user can further mark (or unmark) data epochs for rejection by clicking on them. Pressing 'Update Marks' (lower right) saves the accumulated rejection markings.

Although a set of event-locked data trials has just one ERP average, the number of possible ERP images of a set of trials is very large since the trials can be sorted, optionally smoothed, and imaged along any path (linear or nonlinear), through the possibly high dimensional space of trial attributes and/or event values. However, not all trial sorting orders give equal insights into the brain dynamics expressed in the data. It is therefore up to the user to decide which ERP images to study.

ERP images can also be misinterpreted. For example, using phase-sorting at one frequency (see Fig. 3(C)) can obscure the presence of oscillatory phenomena at other frequencies. It is important not to lose sight of the fact that nearly all activity recorded from scalp electrodes is the volume conducted sum of activities originating within a number of cortical domains. EEGLAB uses ICA (see below) to separate out these activities under the assumption that their activities are temporally independent or at least more temporally independent than any linear combinations of their signals.

2.3. Independent component analysis (ICA)

A primary tool of EEGLAB is to facilitate the process of applying and evaluating the results of independent component analysis of EEG data. ICA algorithms have proven capable of isolating both artifactual and neurally generated EEG sources (Jung et al., 2000; Makeig et al., 1999) whose EEG contributions, across the training data, are maximally independent of one another. ICA was first applied to EEG by Makeig et al. (1996) and is now widely used in the EEG research community, most often to detect and remove stereotyped eye, muscle, and line noise artifacts (Jung et al., 1999, 2000). The temporal independence assumption of ICA is readily understood as a basis for separating artifact sources, since their activities will ordinarily not be reliably

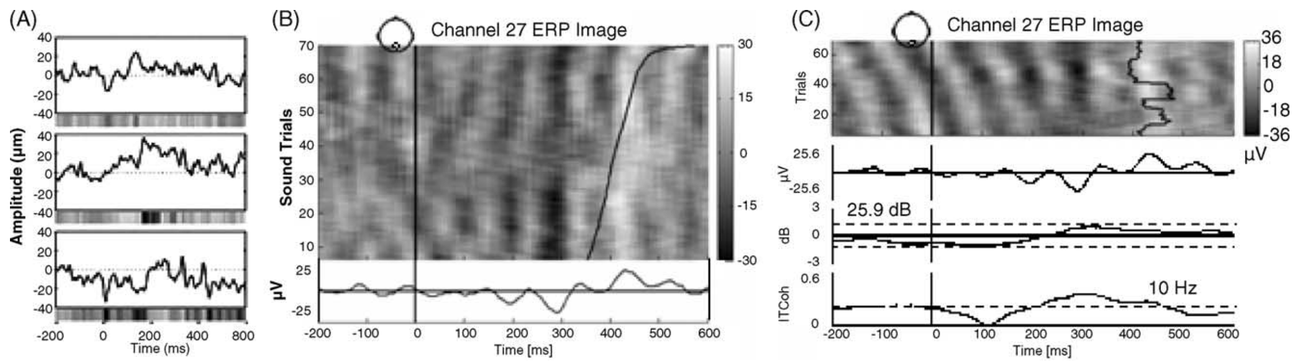


Fig. 3. ERP image construction. (A) ERP-image plots are constructed by color-coding (grey bars) potential variations occurring in single-trial epochs (black traces). (B) Vertically stacking thin color-coded horizontal bars, each representing a single trial in an event-related dataset, produces an ERP image. Here, trials were sorted vertically according to the subject reaction-time (right curving black trace) before applying a 10-epoch vertical moving average. The trace below the ERP image shows the ERP average of the imaged data epochs. The dot on the scalp map (top) indicates the scalp position of the channel whose data are imaged. (C) The *erpimage()* function automates several methods of sorting trials. Here, EEG phase in a given time/frequency window was used as the sorting variable. For each trial, a 10-Hz wavelet was applied to measure oscillatory activity in a 3-cycle window centered at time 0. Trials were then sorted (top to bottom) in order of their alpha band frequency phase values ($-\pi$ to π) relative to stimulus onset and were displayed as an ERP image, again smoothed by a 10-trial moving average. The data were not otherwise filtered. The partial inter-trial phase coherence of the data following the stimulus onset is then visible as a change in the slope of the imaged activity wave fronts to near-vertical after 200 ms. Inter-trial phase coherence (bottom trace) shows that the distribution of alpha activity phase across trials is non-random (i.e., is partially phase-reset) between 200 and 450 ms (dotted line in lower trace shows $P = 0.01$), resulting in same alpha activity appearing in the ERP average trace (top panel). The middle trace shows that mean changes in alpha power (in 'dB') did not change significantly (dotted lines) during the epochs. The baseline power level at the analysis frequency (25.9 dB, relative units) is indicated for possible comparison with other conditions.

phase-locked to one another, given enough training data. In practice, however, ICA also has proved capable of separating biologically plausible brain sources whose activity patterns are distinctly linked to behavioral phenomena. In fact, many of the biologically plausible sources ICA identifies in EEG data have scalp maps nearly fitting the projection of a single equivalent current dipole (Jung et al., 2001; Makeig et al., 2002), and are therefore quite compatible with the projection to the scalp electrodes of synchronous local field activity within a connected patch of cortex.

EEGLAB contains an automated version, *runica()* (Makeig, 1997), of the infomax ICA algorithm (Bell and Sejnowski, 1995) with several enhancements (Amari et al., 1996; Lee et al., 1999) both as a Matlab function and as a stand-alone binary C program that allows faster and less memory-intensive computation. The toolbox also allows the user to select any of over 20 available ICA algorithms including JADE (Cardoso and Souloumiac, 1993) and fixed-point ICA (Hyvärinen and Oja, 2000).

Though it is not our goal here to describe ICA in detail, we will try to give some insight about its nature. In short, ICA finds a coordinate frame in which the data projections have minimal temporal overlap. The core mathematical concept of ICA is to minimize the mutual information among the data projections or maximize their joint entropy. ICA can be viewed as an alternative linear decomposition to principal component analysis (PCA). PCA applied in the temporal domain would specifically make each successive component account for as much as possible of the activity uncorrelated with previously determined components—whereas ICA seeks maximally independent sources.

This difference in goals leads to dramatic differences in their results. PCA components are both temporally and spatially orthogonal, a constraint unrealistic for actual EEG sources, which arise in domains (spatial regions) of partially synchronous activity in electrically oriented cortical neurons (and possibly glia). Because the density of cortical connections is weighted towards local connections ($\ll$1 cm), particularly in the network of inhibitory cells that sustain cortical oscillations (Pauluis et al., 1999), the partially synchronous domains giving rise to EEG activity recorded on the scalp should be mainly compact—though the extent and density of these partially synchronous activities are not known. Through simple volume conduction, the projection of synchronous activity within nearly any patch of cortex will be widespread on the scalp. Any electrode will therefore sum contributions of EEG sources in a large portion of cortex. EEG source contributions to scalp electrode potentials depend on source strengths and orientations as much as source locations. The scalp projections of actual brain EEG sources, therefore, are nearly always overlapping and non-orthogonal, contrary to the assumption of PCA. Indeed, because of the spatial orthogonality constraint, projections of smaller principal components to the scalp typically resemble checkerboard maps that could not represent coherent activity within a connected patch of cortex.

Therefore, to find biologically plausible sources, PCA must be followed by an axis rotation procedure. Previously proposed procedures, such as Promax and Varimax, were drawn from the factor analysis literature. ICA can be viewed as a more powerful rotation method, though in practice ICA is usually applied to the original data without PCA pre-processing (for details, see Makeig et al., 1999). ICA seeks to find component time courses that are mutually independent, meaning that component cross-correlations as well as all the higher order moments of the signals are zero. ICA is free to adapt to the actual projection patterns of EEG generators if their activity time courses are (near) independent of one another. ICA is now being applied to many biomedical signal processing problems including decomposing fMRI data (Duann et al., 2002b) and speech and noise separation (Park et al., 1999). Performing ICA decomposition is most appropriate when sources are linearly mixed in the recorded signals, without differential time delays. These assumptions are precisely met for brain (and non-brain) generator processes summed by volume conduction in scalp EEG data. Because ICA does not attempt to maximize the variance of each component, ICA components may account for more equal portions of the total signals than PCA components. For example, in 32-channel decompositions ICA component activities typically account for near 0% to about 5% of the total signals. ICA may usefully be applied to data with 128 or 256 channels, though meaningful results can also be achieved using 32 or fewer channels (Makeig et al., 2002).

Some earlier studies applied ICA to collections of ERP data averages (Makeig et al., 1997, 1999). However, this approach requires care and caution in interpretation of results. To separate two or more processes, ICA requires that their independence be expressed in the data. A small set of data averages may not include enough conditions in the training set to demonstrate the independence of the underlying processes. If, for example, several processes are partially phase reset in similar ways, the resulting event-locked response averages may not express their underlying functional and temporal independence. Data averages, by their nature, contain sums of activities occurring at similar latencies relative to some class of events. When two or more sources invariably contribute to a set of response averages at the same latency, ICA, trained on these averages, may assign their summed activities to a single component. Trained on the unaveraged data however, ICA may use their relative variability in single trials to separate them. A second problem with applying ICA to data averages is that the averaging process nearly cancels out the activity of many of the EEG sources. Thus applying ICA to the unaveraged EEG data also allows ICA to separate ongoing activity of EEG sources even if they are only partially phase-locked for brief time periods. This is most useful when there are a sufficient number of channels to fit the most active EEG and artifact processes.

Theoretical assumptions underlying the use of ICA to decompose EEG data include: (1) the data must contain enough data points for the temporal independence of the underlying sources to be expressed (see Section 3). (2) No electrode activity should be a linear mixture of other electrode activities (as may occur for, e.g., average-reference data). If so, before running ICA training, *runica()* automatically performs PCA pre-processing to reduce the number of data dimensions to the rank of the input data. (3) ICA assumes that each data source is spatially stationary throughout the training data. This restriction may be partially relaxed in more recent ICA methods (Anemüller et al., 2003). (4) ICA assumes that the distributions of activation values for each EEG source are not precisely Gaussian. When a source distribution is sub-Gaussian (e.g., as with line noise), the extended option of infomax ICA must be used to separate it. The current distribution of EEGLAB, therefore, focuses on applying ICA directly to continuous EEG data or, typically, to concatenated collections of single EEG data trials. Fig. 4 illustrates the use of infomax ICA applied to the 80 EEG epochs of the EEGLAB sample dataset. The lower the component index returned from *runica()*, the more EEG data (neural and/or artifactual) it accounts for.

To determine which components are behaviorally relevant and should be selected for further investigation, EEGLAB allows the user to plot component contributions to the raw data spectrum and/or to the trial-average ERP at all (or specified) channels. Fig. 5(A) shows component contributions at an alpha frequency to channel POz during the sample epochs. The function returns the amount contributed by each component as a percentage of total data power. Another EEGLAB function for estimating component contributions to the data, depicted in Fig. 5(B), shows component contributions to the trial-average ERP in the −500 to 1000 ms latency range. These and other visualization functions help users to select which components to process further using ERP-image plotting (as described above) or using a variety of spectral decomposition techniques (discussed below).

2.4. Time/frequency analysis

To assess event-related spectral amplitude, phase and coherence perturbations in data recorded from single electrodes and/or in ICA components, EEGLAB employs custom spectral decomposition techniques. Our primary measures are the baseline or epoch-mean power spectrum and three event-related time/frequency measures: (1) the event-related spectral perturbation (ERSP), measuring mean event-related changes in the power spectrum at a data channel or component (Makeig, 1993), (2) inter-trial coherence (ITC magnitude and phase, also called phase-locking factor) at single channels or components, and (3) event-related cross-coherence (ERCOH, magnitude and phase) between two data channels or components.

2.4.1. ERSP

Plots of the baseline-normalized spectrogram or the event-related spectral perturbation (ERSP) are increasingly used in the EEG literature to visualize mean event-related changes in spectral power over time in a broad frequency range. They generalize the narrow-band event-related desynchronization (ERD) and synchronization (ERS) measures introduced by Pfurtscheller and colleagues (Pfurtscheller and Aranibar, 1979).

Calculating an ERSP requires computing the power spectrum over a sliding latency window then averaging across data trials. The color at each image pixel then indicates power (in dB) at a given frequency and latency relative to the time locking event. Typically, for n trials, if, $F_k(f, t)$ is the spectral estimate of trial k at frequency f and time t

$$\mathrm{ERSP}(f,t) = \frac{1}{n}\sum_{k=1}^{n} |F_k(f,t)|^2 \tag{1}$$

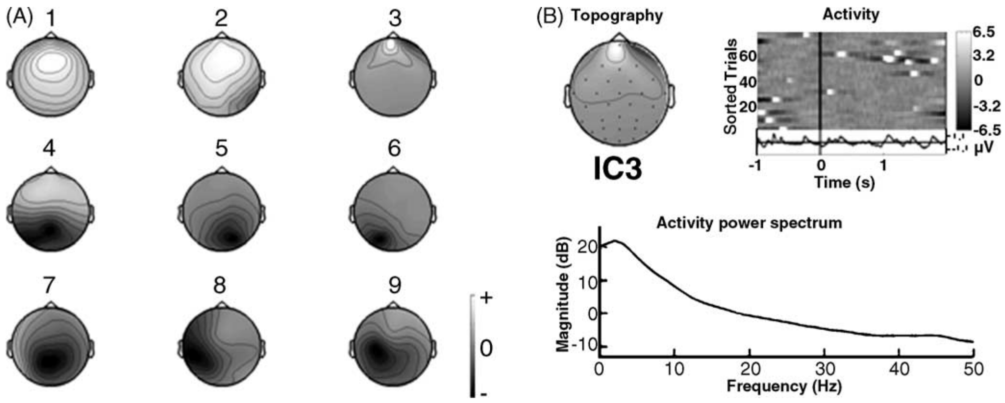


Fig. 4. Visualizing independent components. (A) Topographical 2-D scalp maps of the nine independent components (ICs) accounting for the most EEG variance of the 32 components returned by the ICA algorithm for the sample dataset. The component scalp map values returned by ICA are proportional to μV (scaling is distributed between the component maps and activity time courses). From its far-frontal scalp map, IC3 appears to account for eye movement artifacts. (B) The 'Component Properties' display for IC3 verifies that it accounts for eye artifacts since its activity spectrum is smoothly decreasing (bottom panel), and prominent eye movement artifacts appear in its activity ERP image (top right panel). By removing this and other eye movement components (not shown) from the dataset, the user can remove most evidence of eye movements from the data without removing other activity of interest (Jung et al., 2000).

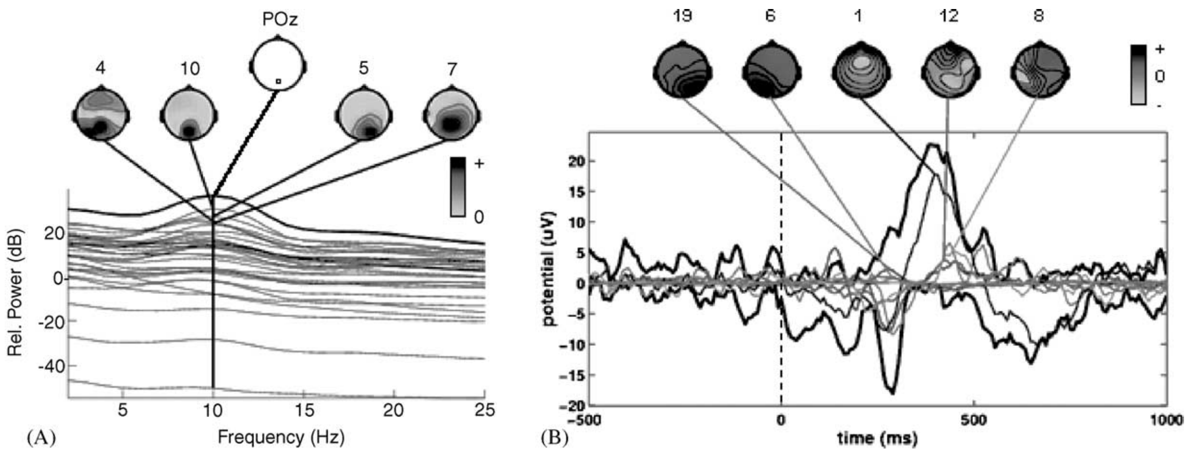


Fig. 5. Evaluating independent component contributions. (A) An EEGLAB *spectopo()* plot showing the components accounting for the largest portions of 10 Hz activity at electrode POz (middle scalp map). The figure shows the power spectrum of the selected channel (top black trace), the activity spectra of the projection to that channel of each of the 32 components (lower traces), and the scalp power maps of the four largest-contributing components (4, 5, 7, 10). (B) An *envtopo()* plot showing the envelopes (i.e., the min and max values, over all channels, at each time point) of the five independent components making the largest potential contributions to the ERP. The black thick traces show the envelope (all channels) of the ERP data and the thin traces, the envelopes of the depicted component contributions to the ERP.

To compute $F_k(f, t)$, EEGLAB uses either the short-time Fourier transform, a sinusoidal wavelet (short-time DFT) transform, or a Slepian multitaper decomposition (Thompson, 1982) that provides a specified time and frequency resolution. In our experience, there are no dramatic differences between these decompositions (though the number of cycles in each data window can be critical). Most often we use a version of sinusoidal wavelets in which the number of cycles is increased slowly with frequency (Fig. 6). This feature allows us to obtain better frequency resolution at higher frequencies than a conventional wavelet approach that uses constant cycle length. This method is also better matched to the linear scale we use to visualize frequencies. To visualize power changes across the frequency range, we subtract the mean baseline log power spectrum from each spectral estimate, producing the baseline-normalized ERSP.

Significance of deviations from baseline power is assessed using a bootstrap method. A surrogate data distribution is constructed by selecting spectral estimates for each trial from randomly selected latency windows in the specified epoch

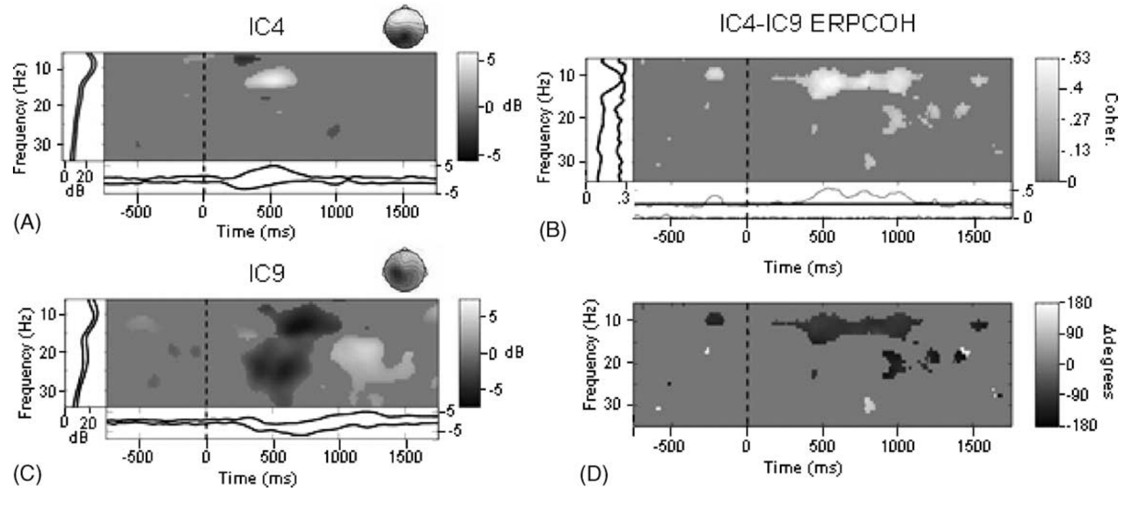


Fig. 6. Time/frequency decompositions of independent component activities. Time/frequency decomposition was applied to the activities of two independent EEG components using sinusoidal wavelet transforms, 3 cycles in length at the lowest frequency (6 Hz), increasing linearly with frequency up to nine cycles at the highest plotted frequency (35 Hz). Using this approach, it is possible to obtain reasonable time and frequency stability at all frequencies. (A-B) Event-related spectral perturbation (ERSP) plots showing mean changes in spectral power during the epoch, relative to a 1-s pre-stimulus baseline (plotted vertically on the left). Component IC4 shows a transient increase near 12 Hz centered at 500 ms, while component IC9 shows a power decrease in this range following 500 ms. (C-D) Phase cross-coherence (ERPCOH) magnitude and phase delay between the two components shown in panels A-B, zero-masked in regions in which cross-coherence magnitude was not significant ($P > 0.01$). The components appear to become partially synchronized above 10 Hz (coherence ≤ 0.53) during the period 400–1000 ms with a phase offset near $-120°$. Under the minimum phase assumption, this implies that high-alpha activity of IC9 tends to lead that of IC4 during this period by about 30 ms.

baseline (e.g., prior to stimulus onset), and then averaging these. Applying this process several hundred times (default: $N = 200$) produces a surrogate 'baseline' amplitude distribution whose specified percentiles are then taken as significance thresholds. If sufficient pre-stimulus data are not available, the surrogate data may be drawn from any other part or from the whole epoch. Fig. 6(A) and (B) show significant ERPS phenomena for two independent EEG components.

2.4.2. Inter-trial coherence (ITC)

ITC is a frequency-domain measure of the partial or exact synchronization of activity at a particular latency and frequency to a set of experimental events to which EEG data trials are time locked. The measure was introduced by Tallon-Baudry et al. (1996) and termed a 'phase locking factor.' The term 'inter-trial coherence' refers to its interpretation as the event-related phase coherence (ITPC) or event-related linear coherence (ITLC) between recorded EEG activity and an event-phase indicator function (e.g. a Dirac or cosine function centered on the time locking event). Using the same notation as above inter-trial phase coherence is defined by

$$\mathrm{ITPC}(f,t) = \frac{1}{n}\sum_{k=1}^{n}\frac{F_k(f,t)}{|F_k(f,t)|} \tag{2}$$

and inter-trial linear coherence by

$$\mathrm{ITLC}(f,t) = \frac{\sum_{i=1}^{n} F_k(f,t)}{\sqrt{n\sum_{i=1}^{n}|F_k(f,t)|^2}} \tag{3}$$

where || represents the complex norm. The most common (and default) version is inter-trial phase coherence (called 'phase-locking factor' by Tallon-Baudry et al., 1996). The ITC measure takes values between 0 and 1. A value of 0 (not expected in practice based on a finite number of epochs) represents absence of synchronization between EEG data and the timelocking events; a value near 1 indicates their perfect synchronization (i.e., near perfect EEG phase reproducibility across trials at a given latency). In the complex 2-D Cartesian coordinate frame, spectral estimates at given frequencies and times are returned as complex vectors in the 2-D phase space. The norm and phase angle of each vector are represented by the magnitude and phase of the spectral estimate. To compute inter-trail phase coherence (ITPC), we first normalize the lengths of each of the trial activity vectors to 1 and then compute their complex average. Thus, only the information about the phase of the spectral estimate of each trial is taken into account.

For linear inter-trial coherence (ITLC), the initial normalization step is omitted: the vector sum is computed and then normalized by RMS power in the single-trial estimates. EEGLAB function *erpimage()* computes ITPC at a single frequency for display beneath an ERP image (Fig. 3(C)); function *timef()* computes color-coded ITPC or ITLC images across frequencies (not shown). As for the ERSP, ITC significance levels are assessed using surrogate data by randomly shuffling the single-trial spectral estimates from different latency windows during the baseline period.

2.4.3. ERCOH

EEGLAB function *crossf()* computes event-related coherence (ERCOH) between two channel or component activities in sets of trials to determine the degree of synchronization between the two activity measures. As for ITC, both phase coherence (ERPCOH) and linear coherence (ERLCOH) measures are supported. Other phase coherence measures have not (yet) been included in EEGLAB (e.g., Lachaux et al., 1999). In EEGLAB, for two signals, *a* and *b*, and using the same notation as above phase cross-coherence is defined by

$$\mathrm{ERPCOH}^{a,b}(f,t) = \frac{1}{n}\sum_{k=1}^{n}\frac{F_k^a(f,t)F_k^b(f,t)^*}{|F_k^a(f,t)F_k^b(f,t)|} \tag{4}$$

and linear cross-coherence by

$$\mathrm{ERLCOH}^{a,b}(f,t) = \frac{\sum_{k=1}^{n} F_k^a(f,t)F_k^b(f,t)^*}{\sqrt{\sum_{k=1}^{n}|F_k^a(f,t)|^2}\sqrt{\sum_{k=1}^{n}|F_k^b(f,t)|^2}} \tag{5}$$

where $F_k^b(f,t)^*$ is the complex conjugate of $F_k^b(f,t)$. The magnitude of cross-coherence varies between 0 and 1, a value of 0 again indicating a complete absence of synchronization at the given frequency *f* in the time window centered on *t*, and 1 indicating perfect synchronization. As for ITPC, the normalizing factor in the ERPCOH denominator ensures that only the relative phase of the two spectral estimates at each trial is taken into account. Linear ERCOH (ERLCOH), by contrast, estimates the extent of complex linear relationship between the two signals (proportional amplitudes at a fixed delay).

When ERCOH magnitude (i.e., norm of the complex-valued ERCOH vector) is significantly above its expected baseline value, the phase of the ERCOH vector may indicate, under the minimum phase assumption, which of the two component activities tends to lead the other at the analysis frequency. The minimum phase assumption means that the actual phase lag is less than $\pm 180°$. Fig. 6(C) illustrates significant ERPCOH synchronization between two components. Even though independent components were identified by ICA as being (maximally) independent over the whole time range, they may exhibit partial but statistically significant synchronization, within specific event-related time/frequency windows (Delorme et al., 2002). Here again, *crossf()* can assess significance of the observed ERCOH using the method of surrogate data by computing the expected ERCOH distribution using randomly selected data windows from the 'baseline' portion of each epoch. Different surrogate data selection methods are used to estimate ERCOH for the two processes, either including or excluding any common spectral amplitude changes and/or partial phase-locking related to the time-locking experimental events. These four methods are referred to in EEGLAB as linear or phase coherence, with or without removal of common ITC. The preferable method may depend on several factors that we do not detail here.

2.5. Menu calls and script writing

The EEGLAB graphic user interface (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. Table 1 provides examples of EEGLAB scripts of different levels of complexity. EEGLAB functions may be roughly divided into three layers designed to increase ease-of-use for different types of users:

2.5.1. GUI-based use

Naive 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 five headings: ‘File’ menu functions read/save data file and data information files. ‘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. Finally, the user can use ‘Help’ menu functions to call up documentation on EEGLAB functions and data structures.

Table 1
Sample EEGLAB processing scripts

1	» pop_erpimage(EEG);
2	» figure; pop_erpimage(EEG, 1, [1], [], ‘Channel 1 erpimage’, 10, 1);
3	» erpimage(EEG.data(1,:), ones(1, EEG.trials)*EEG.xmax*1000, linspace(EEG.xmin*1000, EEG.xmax*1000, EEG.pnts), ‘Channel 1 ERP image’, 10, 1, ‘topo’, {1 EEG.chanlocs }, ‘erp’, ‘cbar’);

All scripts assume that the Matlab data structure ‘EEG’ contains the sample EEGLAB dataset (described in the EEGLAB tutorial and available for download). Script 1 calls the EEGLAB (‘pop’) interface function that in turn calls the *erpimage()* processing function to compute and draw an ERP image plot (Jung et al., 1999; Makeig et al., 1999) of a selected single-channel time record for each trial. Additional plotting parameters can then be entered manually by the user in the resulting pop-up window. Script 2 performs the same action, but now the ERP-image ‘pop’ function is called with specific arguments. The ERP-image plot then appears directly, with no intervening ‘pop’ window. Each time the user selects an operation from the EEGLAB menu, the resulting Matlab function call (including all input parameters) is appended to the EEGLAB session command history. Subsequently, the user can simply copy and paste commands from the command history to repeat the same actions. Thus, in Script 3, the ‘pop’ ERP-image function is bypassed and the eponymous EEGLAB data processing function, *erpimage()*, is called directly by the user script referencing parameters stored in the EEG data structure. The *erpimage()* function requires no knowledge of the EEG data structure used by EEGLAB, and can be applied to any user-defined data array. If the user selects the supplied default parameters in the *pop_erpimage()* pop-up data entry window, the three scripts will all have the same effect. See the Matlab help messages for the meaning of the *pop_erpimage()* and *erpimage()* function arguments (also available as HTML pages linked to the main EEGLAB website).

2.5.2. EEGLAB command history

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. For example, function *pop_erpimage()* calls *erpimage()*. 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 EEG data.

2.5.3. Custom EEGLAB scripting

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. 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. The GUI interface uses a single Matlab variable, a structure named ‘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.

2.6. Distribution, documentation and support

The EEGLAB toolbox is distributed under the GNU General Public License (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/eeglab/. As the toolbox currently includes approximately 300 Matlab functions comprising 50,000 lines of Matlab code, it is not possible to describe all of its functionality in a journal-length paper. An extensive user tutorial explains in detail how to import and process data using EEGLAB, including the derivation and evaluation of its independent components. We also provide ‘Frequently Asked Questions (FAQ)’ and ‘Known Bugs’ web pages, a support email (eeglab@sccn.ucsd.edu), a mailing list for software updates (eelagbnews@sccn.ucsd.edu), and a discussion mailing list (eeglablist@sccn.ucsd.edu) that currently reaches over a 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. The EEGLAB history feature also makes it easy to generate test scripts that we now launch nightly to maintain EEGLAB stability.

3. Discussion

We have developed EEGLAB, a complete interactive environment for processing EEG (or MEG) data under Matlab, to provide both standard and advanced EEG processing functions developed in our own and other laboratories. EEGLAB is strongly oriented towards single-trial visualization techniques, ICA, and event-related time/frequency analysis. Because the software was developed by and for ERP/EEG researchers, we have taken care to make the data processing as transparent as possible and to allow users to tune their parameters as easily as possible. We will now briefly review a few limitations of EEGLAB and, because the methods incorporated into EEGLAB are not yet widely practiced, some limitations of ICA applied to high-density EEG data.

3.1. Limitations of time/frequency decomposition

Filtering methods implemented in EEGLAB take advantage of linear filtering implemented in the Matlab Signal Processing toolbox. One of the drawbacks of using linear filters is that the signal roll-off at the cut-off frequency is weaker than what it would be using nonlinear filters. However, with linear filtering, data phase information is preserved across frequencies. Time-frequency decomposition in EEGLAB is limited to FFTs, multi-taper analysis, and a single type of sinusoidal wavelet, as is standard for EEG analysis. Other methods, for example the Hilbert method, are not currently implemented. However quantitative comparisons show that results on EEG data using Hilbert transforms do not differ dramatically from applying sinusoidal wavelets (Le Van Quyen et al., 2001). Also, bi-coherence between frequencies cannot yet be assessed within EEGLAB (e.g., Lachaux et al., 2003; von Stein and Sarnthein, 2000). We intend in the future to include functions to assess synchronization (1) of phase at one frequency with amplitude at another frequency, (2) of phase synchronization between frequencies, and (3) of amplitude correlation between frequencies. We welcome further open source contributions implementing other time-frequency approaches, and have added an EEGLAB plug-in facility to promote and ease development of such contributions.

3.1.1. Significance and statistical comparisons across subjects or conditions

To assess significance of within-subject measures, EEGLAB uses non-parametrical methods that do not assume a known activity distribution. A null hypothesis distribution, used to determine significance thresholds, is estimated by accumulating surrogate data, shuffling the data across latencies alone, latencies and trials, or trials alone. To compensate for multiple comparisons, significance thresholds may need to be decreased (e.g. Bonferroni, 1950; Holm, 1979). Since it is not reasonable to compute an unlimited amount of surrogate data to estimate very low probability thresholds heuristically, we have implemented a method to fit the observed surrogate data distribution using a fourth order distribution fit (Ramberg et al., 1979). This feature will be available in a near-term release of EEGLAB.

To test significance across conditions or subjects, we either use parametrical tests or accumulated significance results from each subject. Our ERP function *pop_comperp()* currently uses a t-test to compare two conditions for several subjects. When processing spectral decompositions of one channel (or component class) from different subjects (already been masked for significance), our *tftopo()* function applies a threshold derived by simple statistics on the number of subjects for which the spectral decomposition is significant at a give time/frequency point. If not enough subjects show a significant change at the specified point, this point is considered non-significant in the group average. This is a statistically conservative approach. For further statistical assessment, raw data, ERP, or independent component weights and activity can be exported as ASCII to statistical packages such as Statview (SAS Institute Inc.), SPSS (SPSS Inc.), or the Matlab Statistics Toolbox (The Mathworks, Inc.).

3.1.2. ICA stability

Because the infomax ICA algorithm begins with a random unmixing matrix and then randomly shuffles the order of the data time points before each training step, the results of successive ICA decompositions may be slightly different even when ICA is performed on the same data. In particular, the scalp maps and activity time courses of the independent components (and their order), may differ slightly across runs. Therefore, we advise that features of the decomposition that do not remain stable across decompositions of the same data should not be interpreted except as irresolvable ICA 'uncertainty.' Differences between decompositions trained on somewhat different data subsets may have additional causes. We are currently investigating the stability of ICA methods applied to typical datasets (Delorme et al., in preparation).

3.1.3. Difference between ICA algorithms

Which is the best ICA algorithm to use for EEG decomposition? From a theoretical point of view, all ICA algorithms maximize independence in an approximate sense (Lee et al., 2000), while the degree to which EEG data actually fit ICA assumptions is unknown. Applied to simulated, relatively low dimensional data sets for which the ICA assumptions are exactly fulfilled, leading ICA algorithms (including infomax, JADE, and FastICA) return near-equivalent components. However, the physiological significance of any differences in the results of the same or different ICA algorithms (or of different parameter choices for the various algorithms) has not been systematically tested and reported—neither by us nor, as far as we know, by others. Therefore, different ICA decompositions may give slightly different results, as has been shown for neural ensemble data (Laubach et al., 1999) and fMRI data (Duann et al., 2001; Esposito et al., 2002). Each ICA algorithm has its own particularities. The infomax algorithm in its native form can only separate sources with super-Gaussian (i.e., peaky, thick-tailed) activity distributions. If there are strong electrical artifacts in data, it is preferable to use the 'extended' ICA option of *runica()* (Lee et al., 1999), to allow the algorithm to detect sources with sub-Gaussian activity distribution, such as line current artifacts and/or slow activity.

Whereas infomax implicitly uses a combination of higher-order moments of the data to find independent components, the JADE algorithm (Cardoso and Souloumiac, 1993) diagonalizes all the fourth-order moments explicitly. Although for low numbers of data channels the JADE algorithm is fast and stable, the memory required to manipulate all the fourth-order moments becomes quite impractical with high numbers of channels. Whereas both infomax and JADE algorithms find and return all the independent components at once, the default setting of the fixed-point ICA algorithm of Hyvärinen and Oja (2000) computes and returns components one by one. The order of the components it returns, however cannot be known in advance, and performing a complete decomposition is not faster than with infomax. Also, in our experience (see also Esposito et al., 2002) the fixed-point ICA algorithm may be less robust than infomax ICA when applied high-dimensional real data. To decompose EEG data, therefore, we most often use infomax or extended infomax ICA. The infomax algorithm reliably finds independent components that are physiologically plausible, functionally distinct, and often have spatial and functional similarities across data sets, sessions, and subjects (Delorme et al., 2002; Makeig et al., 2002).

3.1.4. Insufficient data for running ICA

A chief case in which ICA algorithms may not return reliable results is when too few data are provided to them. ICA being a statistical method, if the independence of the functionally distinct EEG processes is not adequately exhibited in the data, ICA cannot separate them.

The size of the weight matrix being the square of the number of channels, a number of time points at least a few times the square of the number of channels is usually needed to obtain reliable decompositions. These data points may be drawn from continuous data or from several data epochs. Of course, additional data points can only improve the decomposition—when and if relative stationarity of the spatial structure of the EEG sources set can be assumed. In our experience, using short baseline-zeroed data epochs that include task-related behavior may give qualitatively more consistent results than using longer data epochs. Using short epochs constrains ICA to focus on the task-relevant portion of the data.

3.1.5. Nonlinearities

Another case in which ICA will fail to extract all the involved sources occurs when the data are not a linear sum of the underlying source projections—this chiefly occurs when the amplifiers become 'railed' at high signal levels, leading to signal 'clipping', or when high signal levels exceed the input range of the A/D converter, leading to signal 'wrap-around.' In either case, the severe nonlinearity involved will cause linear ICA algorithms to give spurious results, so such data epochs must be carefully rejected from the data before running ICA.

3.1.6. Noise

Finally, when the data contains many more strong spatial sources than the number of recording channels, the additional sources must be mixed into the output components. In particular, this may occur during 'paroxysmal' noise which may for instance be introduced into EEG data during strong head movements. Else, a loose electrode may introduce a large noise signal not linearly related to any of the other electrode signals. In this case, ICA may dedicate a single component to the electrode noise, thus unnecessarily reducing the number of components available to separate other neural and artifact sources. Therefore, we find it best to train ICA on carefully pruned 'clean' data epochs, which can, however, retain spatially stereotyped artifact activity such as eye blinks and eye movements, repeated muscle activity, etc.

3.1.7. Processing speed

Matlab offers a powerful environment for processing biophysical data because of (1) the simplicity of its command line language, (2) the many Matlab functions made available by The Mathworks, Inc., and by independent researchers, and (3) its high-level visualization capabilities. However, there are two possible problems in using Matlab for processing EEG data. First, though ever-increasing speed of current workstations continues to make processing time of less limiting importance for data analysis, interpreted computer languages are inherently slower than compiled languages. Matlab has facilities for compiling and running binary versions of scripted functions, but their speed may still be suboptimal. For this reason, we converted to C the most time consuming EEGLAB function, *runica()*. Both the Matlab ICA function–*runica()*—and the binary C-language ICA function–*binca()*—can be called from the EEGLAB GUI. Under Matlab v4, we observed a speed-up of about 10 for *binica()* compared to *runica()*. However, under Matlab v6 the speed-up factor seems to be much smaller (<100%). Most other EEGLAB functions are less compute intensive.

3.1.8. Memory requirements

Another relative disadvantage of using Matlab to process high-density EEG data is that Matlab currently converts all floating-point numbers to 64-byte double-precision, thus requiring large amounts of main memory to process large data sets. Though hopefully future Matlab versions may allow the option of processing data in 32-bit floating-point format, we have taken care to address this issue in EEGLAB by including various options to minimize memory usage, such as constraining EEGLAB to work on a single dataset, or computing the 'activation' time courses of independent components only as needed. However, this issue remains a serious problem for large datasets: parts of the toolbox may have to be updated to allow very large (e.g., long 256-channel) datasets to be analyzed within the current Linux 2GB/process limit. One possibility is to use the Matlab MEX language, an interface between C and Matlab that allows a wider variety of data types including single precision. Another possibility is to have EEGLAB load into main memory only a part of the dataset at a time. However, as 64-bit processors become more available, the current data space limits of operating systems and Matlab should increase, in which case the remaining problem would only be the burden of purchasing the necessary RAM.

Current development of EEGLAB focuses on processing of large datasets (>1 Gb), semi-automatically grouping independent component across subjects, and component source localization. EEGLAB will also be linked to our FMRLAB toolbox (http://www.sccn.ucsd.edu/fmrlab) to process simultaneously recording EEG and fMRI data (Duann et al., 2002a). We also have begun working with co-developers to increase the range of EEGLAB functions using the 'plug-in' facility, whereby contributors may easily contribute optional EEGLAB code that is readily incorporated into the EEGLAB menu. The plug-in facility is designed so that plug-in functions can be used and distributed both within EEGLAB and independently. By this mechanism we hope to encourage the open source development of comprehensive EEG (and MEG) signal processing tools under EEGLAB.

Acknowledgements

The authors acknowledge the contributions to EEGLAB and its hundreds of functions by many contributors. Principal among these were Colin Humphries, who wrote the topographic plotting functions and the first version of the data scrolling display function, Sigurd Enghoff, who wrote the first versions of the time/frequency functions and translated the MATLAB-coded *runica()* infomax ICA function to binary, and Tzyy-Ping Jung, who contributed the first version of the *erpimage()* function. The *runica()* function itself was written by one of us (SM) building on core ICA code contributed by Tony Bell and Te-won Lee. Several other members of the Computational Neurobiology Laboratory at The Salk Institute contributed other functions. We gratefully acknowledge the support, collaboration and encouragement of Terrence Sejnowski at Salk Institute throughout this research process. We thank Stefan Debener and anonymous reviewers for suggestions on this report, which was supported by the National Institutes of Health USA and by The Swartz Foundation.

References

Anemüller J, Sejnowski TJ, Makeig S. Complex independent component analysis of frequency-domain electroencephalographic data. Neural Networks 2003;16:1311–23.

Amari S-I, Cichocki A, Yang HH. A new learning algorithm for blind source separation. Adv Neural Inf Process Syst 1996;8:757–63.

Baillet S, Mosher JC, Leahy RM, Shattuck DW. BrainStorm: a matlab toolbox for the processing of MEG and EEG signals. In: Proceedings of the 5th International Conference on Human Brain Map., Neuroimage, 1999;9:246.

Bell AJ, Sejnowski TJ. An information-maximization approach to blind separation and blind deconvolution. Neural Comput 1995;7:1129–59.

Bressler SL, Freeman WJ. Frequency analysis of olfactory system EEG in cat, rabbit, and rat. Electroencephalogr Clin Neurophysiol 1980;50:19–24.

Bonferroni CE. Sulle medie multiple di potenze. Bollettino dell'Unione Matematica Italiana, 5 third series, 1950. p. 267–70.

Cardoso J-F, Souloumiac A. Blind beamforming for non gaussian signals. IEE-Proc-F 1993;140:362–70.

Delorme A, Makeig S, Fabre-Thorpe M, Sejnowski T. From single-trials EEG to brain area dynamics. Neurocomputing 2002;44/46:1057–64.

Delorme A, Makeig S. EEG changes accompanying learning regulation of the 12-Hz EEG activity. IEEE Trans Rehabilit Eng 2003;11:133–6.

Duann JR, Jung TP, Kuo WJ, Yeh S, Makeig S, Hsieh JC, Sejnowski TJ, Measuring the variability of event-related BOLD signals. In: Third International Workshop on Independent Component Analysis and Signal Separation, San Diego, USA, 2001.

Duann JR, Jung TP, Makeig S, Sejnowski TJ. fMRLAB: an ICA Toolbox for fMRI Data Analysis, In: Human Brain Mapping, 2002a; Sendai, Japan.

Duann JR, Jung TP, Kuo WJ, Yeh TC, Makeig S, Hsieh JC, Sejnowski TJ. Single-trial variability in event-related BOLD signals. Neuroimage 2002b;15:823–35.

Esposito F, Formisano E, Seifritz E, Goebel R, Morrone R, Tedeschi G, Di Salle F. Spatial independent component analysis of functional MRI time-series: to what extent do results depend on the algorithm used? Hum Brain Mapp 2002;16:146–57.

Friston KJ. Statistical parametric mapping: Ontology and current issues. J Cereb Blood Flow Metab 1995;15:361–70.

Holm S. A simple sequentially rejective multiple test procedure. Scand J Stat 1979;6:65–70.

Hyvärinen A, Oja E. Independent component analysis: algorithms and applications. Neural Netw 2000;13:411–30.

Jung TP, Makeig S, Westerfield M, Townsend J, Courchesne E, Sejnowski TJ. Analyzing and visualizing single-trial event-related potentials. Adv Neural Inf Process Syst 1999;11:118–24.

Jung TP, Makeig S, Westerfield M, Townsend J, Courchesne E, Sejnowski TJ. Analysis and visualization of single-trial event-related potentials. Hum Brain Mapp 2001;14:166–85.

Jung TP, Makeig S, Humphries C, Lee TW, McKeown MJ, Iragui V, Sejnowski TJ. Removing electroencephalographic artifacts by blind source separation. Psychophysiology 2000;37:163–78.

Lachaux JP, Rodriguez E, Martinerie J, Varela FJ. Measuring phase synchrony in brain signals. Hum Brain Mapp 1999;8:194–208.

Lachaux JP, Chavez M, Lutz A. A simple measure of correlation across time. J Neurosci Methods 2003;123:175–88.

Laubach M, Shuler M, Nicolelis MA. Independent component analyses for quantifying neuronal ensemble interactions. J Neurosci Methods 1999;94:141–54.

Le Van Quyen M, Foucher J, Lachaux J, Rodriguez E, Lutz A, Martinerie J, Varela FJ. Comparison of Hilbert transform and wavelet methods for the analysis of neuronal synchrony. J Neurosci Methods 2001;111:83–98.

Lee TW, Girolami M, Sejnowski TJ. Independent component analysis using an extended infomax algorithm for mixed sub-Gaussian and super-Gaussian sources. Neural Comput 1999;11:417–41.

Lee TW, Girolami M, Bell AJ, Sejnowski TJ. A unifying information-theoretic framework for independent component analysis. Comput Math Appl 2000;31:1–21.

Makeig S. Auditory event-related dynamics of the EEG spectrum and effects of exposure to tones. Electroencephalogr Clin Neurophysiol 1993;86:283–93.

Makeig S, Bell AJ, Jung TP, Sejnowski TJ, Independent component analysis of electroencephalographic data. In: Touretzky D, Mozer M, Hasselmo M, editors. Adv Neural Inf Process Syst 1996;8:145–51.

Makeig S, Jung TP, Bell AJ, Ghahremani D, Sejnowski TJ. Blind separation of auditory eventrelated brain responses into independent components. Proc Natl Acad Sci USA 1997;94:10979–84.

Makeig S, Westerfield M, Jung TP, Covington J, Townsend J, Sejnowski TJ, Courchesne E. Functionally independent components of the late positive event-related potential during visual spatial attention. J Neurosci 1999;19:2665–80.

Makeig S, Westerfield M, Jung TP, Enghoff S, Townsend J, Courchesne E, Sejnowski TJ. Dynamic brain sources of visual evoked responses. Science 2002;295:690–4.

Makeig S, et al. Matlab Toolbox for analysis of electrophysiological data. http://www.cnl.salk.edu/∼scott/ica.html, 1997.

Neuenschwander S, Varela FJ. Visually triggered neuronal oscillations in the pigeon: an autocorrelation study of tectal activity. Eur J Neurosci 1993;5:870–81.

Park H-M, Jung H-Y, Lee T-W, Lee S-Y. On subband-based blind signal separation for noisy speech recognition. Electron Lett 1999;35:2011–2.

Pauluis Q, Baker SN, Olivier E. Emergent oscillations in a realistic network: the role of inhibition and the effect of the spatiotemporal distribution of the input. J Comput Neurosci 1999;6:27–48.

Pfurtscheller G, Aranibar A. Evaluation of event-related desynchronization (ERD) preceding and following voluntary self-paced movement. PG-138-46. Electroencephalogr Clin Neurophysiol 1979;46.

Ramberg JS, Dudewicz EJ, PTadikalama PR, Mykytka EF. A probability distribution and its uses in fitting data. Technometrics 1979;21.

Rodriguez E, George N, Lachaux JP, Martinerie J, Renault B, Varela FJ. Perception's shadow: long-distance synchronization of human brain activity. Nature 1999;397:430–3.

Tallon-Baudry C, Bertrand O, Delpuech C, Pernier J. Stimulus specificity of phase-locked and non-phase-locked 40 Hz visual responses in human. J Neurosci 1996;16:4240–9.

Thompson DJ. Spectrum estimation and harmonic analysis. IEEE Proc 1982;70:1055–96.

von Stein A, Sarnthein J. Different frequencies for different scales of cortical integration: from local gamma to long range alpha/theta synchronization. Int J Psychophysiol 2000;38:301–13.

Weiss S, Rappelsberger P. EEG coherence within the 13–18 Hz band as a correlate of a distinct lexical organisation of concrete and abstract nouns in humans. Neurosci Lett 1996;209:17–20.