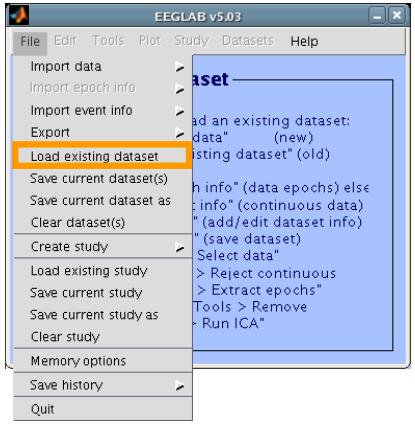
# Command line tools

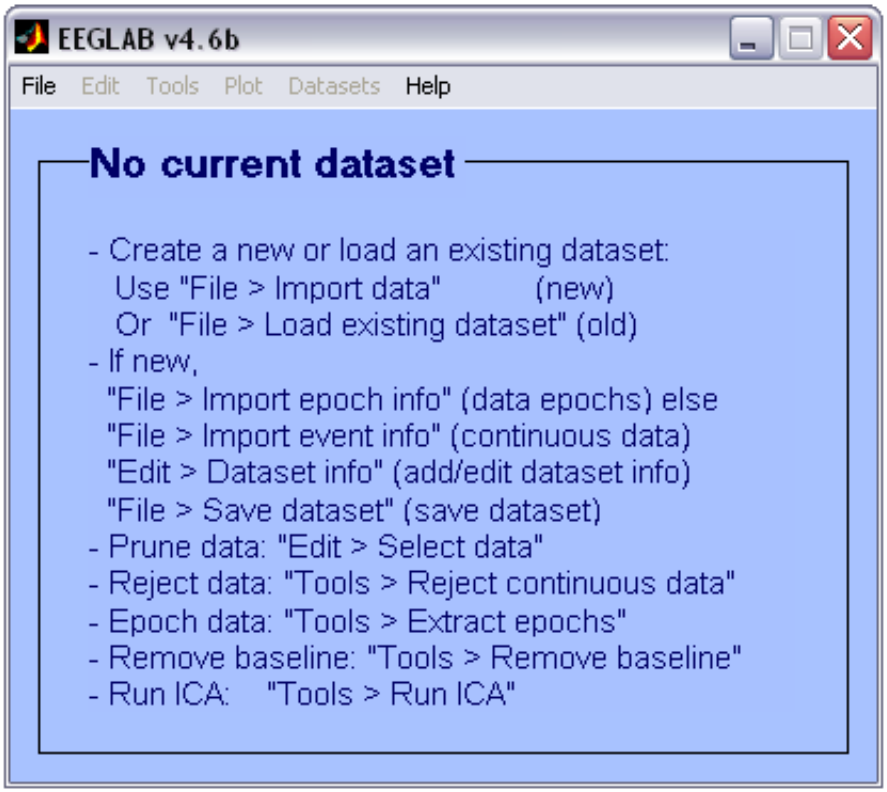



(Menus write both dataset and global history)

- Automated processing on groups of subjects (possibly on several processors).
- Richer options for plotting and processing functions (time-frequency decompositions, …)
- Selecting data/epoch based on event context
- Custom processing…

# Starting EEGLAB

```
>> eeglab
eeglab: options file is /Volumes/donnees/data/STUDYste
Adding path to all EEGLAB functions
Adding path to eeglab/external/bioelectromagnetism_ligt
Adding path to eeglab/external/biosig-partial
Adding path to eeglab/external/fieldtrip-partial
Adding path to eeglab/external/fieldtrip-partial subfolders
EEGLAB: adding plugin function "eegplugin_VisEd"
EEGLAB: adding "eepimport1.02" plugin (see >> help ee
EEGLAB: adding "bdfimport" plugin (see >> help eegplu
EEGLAB: adding "brainmovie0.1b" plugin (see >> help e
EEGLAB: adding "ctfimport1.03" plugin (see >> help eeg
EEGLAB: adding "dipfit2.2" plugin (see >> help eegplugi
EEGLAB: adding "EEG toolbox ERP plotting" plugin (see >> help eegplugin_eeg_toolbox)
EEGLAB: adding "erpssimport1.00" plugin (see >> help eegplugin_erpssimport)
EEGLAB: adding "fmrib1.21" plugin (see >> help eegplugin_fmrib)
EEGLAB: adding "iirfilt1.01" plugin (see >> help eegplugin_iirfilt)
EEGLAB: adding "eepimport1.02" plugin (see >> help eegplugin_ascinstep)
EEGLAB: adding "loreta1.0" plugin (see >> help eegplugin_loreta)
EEGLAB: adding "Butter1.0" plugin (see >> help eegplugin_ERPLAB_filters)
EEGLAB: adding "Measure_Product1.0" plugin (see >> help eegplugin_mp_clustering)
EEGLAB: adding plugin function "eegplugin_miclust"
EEGLAB: adding "4dneuroimaging1.00" plugin (see >> help eegplugin_4dneuroimaging)
>>
```



# Proper EEGLAB plugins

| | |
|---|---|
| **eepimport1.02** | Data importing for EEprobe data (Oostenved & ANT company) |
| **bva_io1.30** | Brain vision analyzer import/export plugin (Widmann & Delorme) |
| **ctfimport1.01** | MEG CTF import plugin (Carver, Weber & Delorme) |
| | |
| **dipfit2.0** | 4-shell and BEM (Oostenveld & Delorme) |
| **fmrib1.2b** | Removal of artifact from simultaneously EEG/fMRI recording (Niazi) |
| **iirfilt1.0** | Non-linear IIR filtering (Pozdin) |
| **loreta2.0** | Interface to LORETA-KEY (Delorme) |

# Matlab toolboxes interfaced as plugins

| | |
|---|---|
| **BIOSIG** | Data importing for rare data binary format (Schloegl) |
| **File-IO** | Data importing (Oostenveld) |
| **Fieldtrip** | Source localization and time-freq. decompositions (Oostenveld) |
| **LIMO-EEG** | General linear model and EEG |
| **SIFT** | Source information flow toolbox |

home
- SCCN web site
- EEGLAB Wiki
- MoBI Lab Wiki
- SCCN Wiki Home

eeglab wiki pages
- EEGLAB web page
- EEGLAB Wiki
- EEGLAB Tutorial
- Online EEGLAB Workshop
- Download EEGLAB
- Revision history
- Help EEGLAB

sccn toolboxes
- EEGLAB
- NFT
- BCILAB
- SIFT
- MoBILAB
- MPT

wiki tools
- Sandbox
- Basic Wiki Syntax
- Wiki Help
- New Users

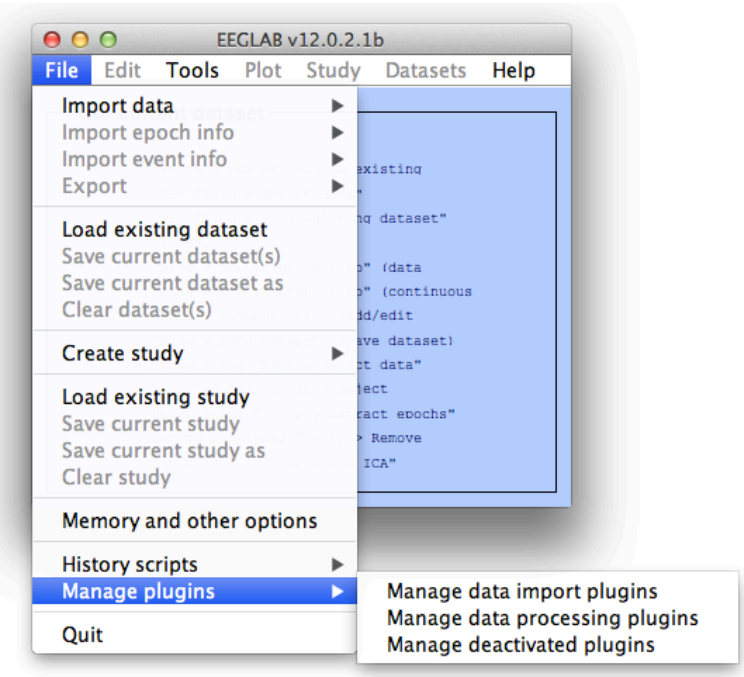
# Plugin list process

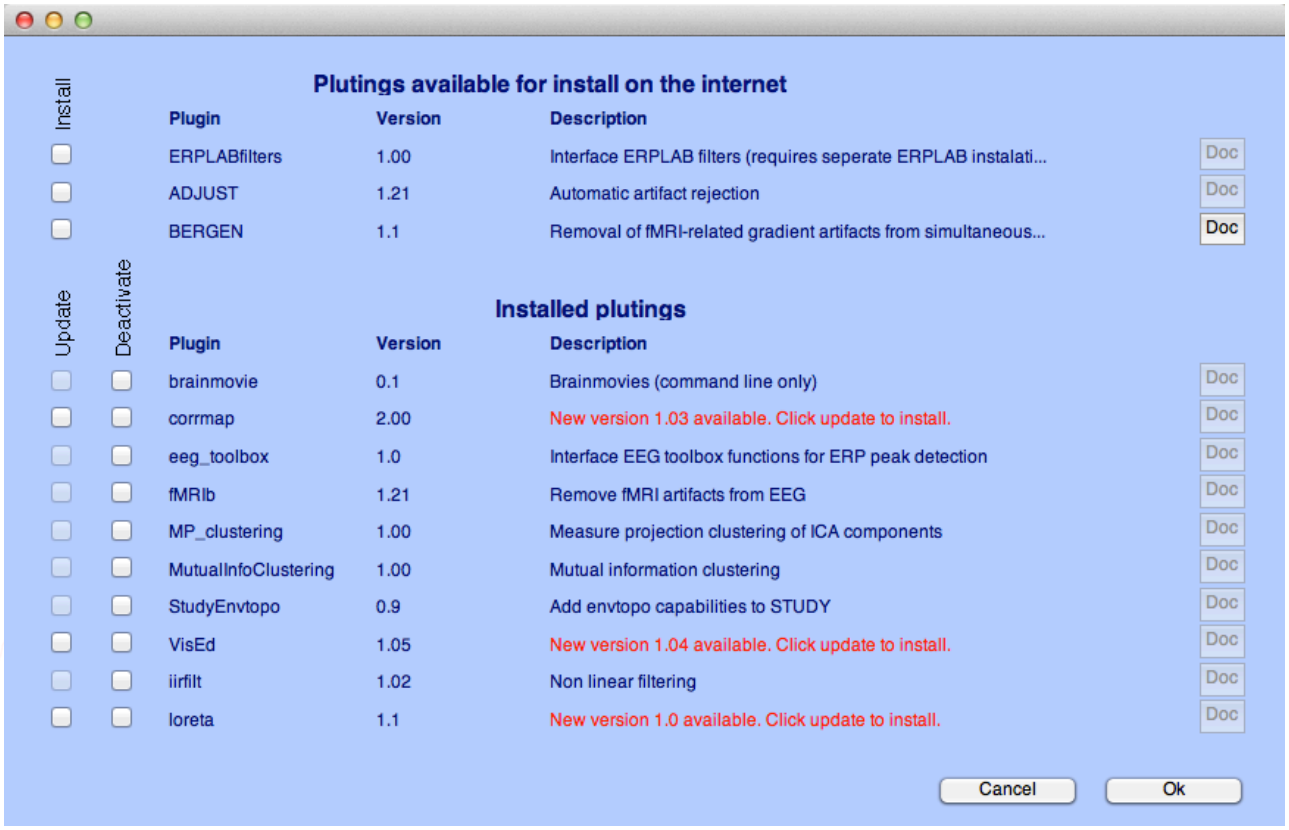
| Plugin name | Version | Short plugin description | Link | Comments |
|---|---|---|---|---|
| brainmovie | 0.1 | Brainmovies (command line only) | Download | User comments |
| corrmap | 1.03 | Import BIOPAC data files | Download | User comments |
| eeg_toolbox | 1.0 | Interface EEG toolbox functions for ERP peak detection | Download | User comments |
| ERPLABfilters | 1.00 | Interface ERPLAB filters (requires seperate ERPLAB instalation) | Download | User comments |
| fMRIb | 1.21 | Remove fMRI artifacts from EEG | Download | User comments |
| MP_clustering | 1.00 | Measure projection clustering of ICA components | Download | User comments |
| MutualInfoClustering | 1.00 | Mutual information clustering | Download | User comments |
| StudyEnvtopo | 0.9 | Add envtopo capabilities to STUDY | Download | User comments |
| VisEd | 1.04 | Add/Edit dataset events | Download | User comments |
| ADJUST | 1.21 | Automatic artifact rejection | Download | User comments |
| iirfilt | 1.02 | Non linear filtering | Download | User comments |
| loreta | 1.0 | Export and import data to/from LORETA software | Download | User comments |
| BERGEN | 1.1 | Removal of fMRI-related gradient artifacts from simultaneous EEG-fMRI data | Download | User comments |

## Add your plugin to the list

You may add your plugin to the list so users can download it automatically from within EEGLAB. There are 5 tabs:

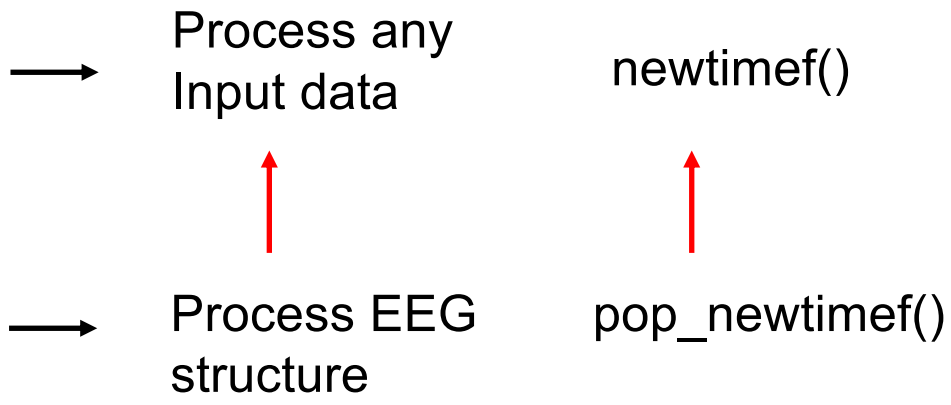
- **Plugin name**: this tab should contain the abbreviated name of your plugin and if necessary a link to the plugin documentation. The plugin documentation may be stored on this wiki.
- **Version**: this tab should contain the version of your plugin. The version listed on this page and the one made available in the eegplugin_xxx.m file must be consistent. This allows EEGLAB to automatically check for newer versions of your plugin.
- **Short plugin description**: this tab should contain a short plugin description (no more than one line). Additional documentation may be provided as a link in tab 1.


EEGLAB v12.0.2.1b
File
Edit
Tools
Plot
Study
Datasets
Help
Import data
Import epoch info
Import event info
Export
Load existing dataset
Save current dataset(s)
Save current dataset as
Clear dataset(s)
Create study
Load existing study
Save current study
Save current study as
Clear study
Memory and other options
History scripts
Manage plugins
Quit
Manage data import plugins
Manage data processing plugins
Manage deactivated plugins


Plutings available for install on the internet
Install
Plugin
Version
Description
ERPLABfilters
1.00
Interface ERPLAB filters (requires seperate ERPLAB instalati...
ADJUST
1.21
Automatic artifact rejection
BERGEN
1.1
Removal of fMRI-related gradient artifacts from simultaneous...
Doc
Installed plutings
Update
Deactivate
brainmovie
0.1
Brainmovies (command line only)
corrmap
2.00
New version 1.03 available. Click update to install.
eeg_toolbox
1.0
Interface EEG toolbox functions for ERP peak detection
fMRIb
1.21
Remove fMRI artifacts from EEG
MP_clustering
1.00
Measure projection clustering of ICA components
MutualInfoClustering
1.00
Mutual information clustering
StudyEnvtopo
0.9
Add envtopo capabilities to STUDY
VisEd
1.05
New version 1.04 available. Click update to install.
iirfilt
1.02
Non linear filtering
loreta
1.1
New version 1.0 available. Click update to install.
Cancel
Ok

# Writing EEGLAB plugins

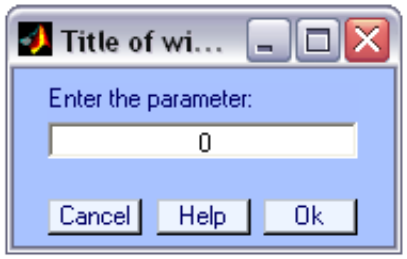
- Assuming that you have a signal processing function called xxxxx
- a pop_xxxxx function will interface your signal processing function
- a eegplugin_xxxxx function will add the menu to the main interface (and history etc…)



# Pop functions

- Called with the EEG structure only pop_xxxxx(EEG), they pop-up a GUI asking for more arguments
- Called with enough arguments, the simply call the signal processing function

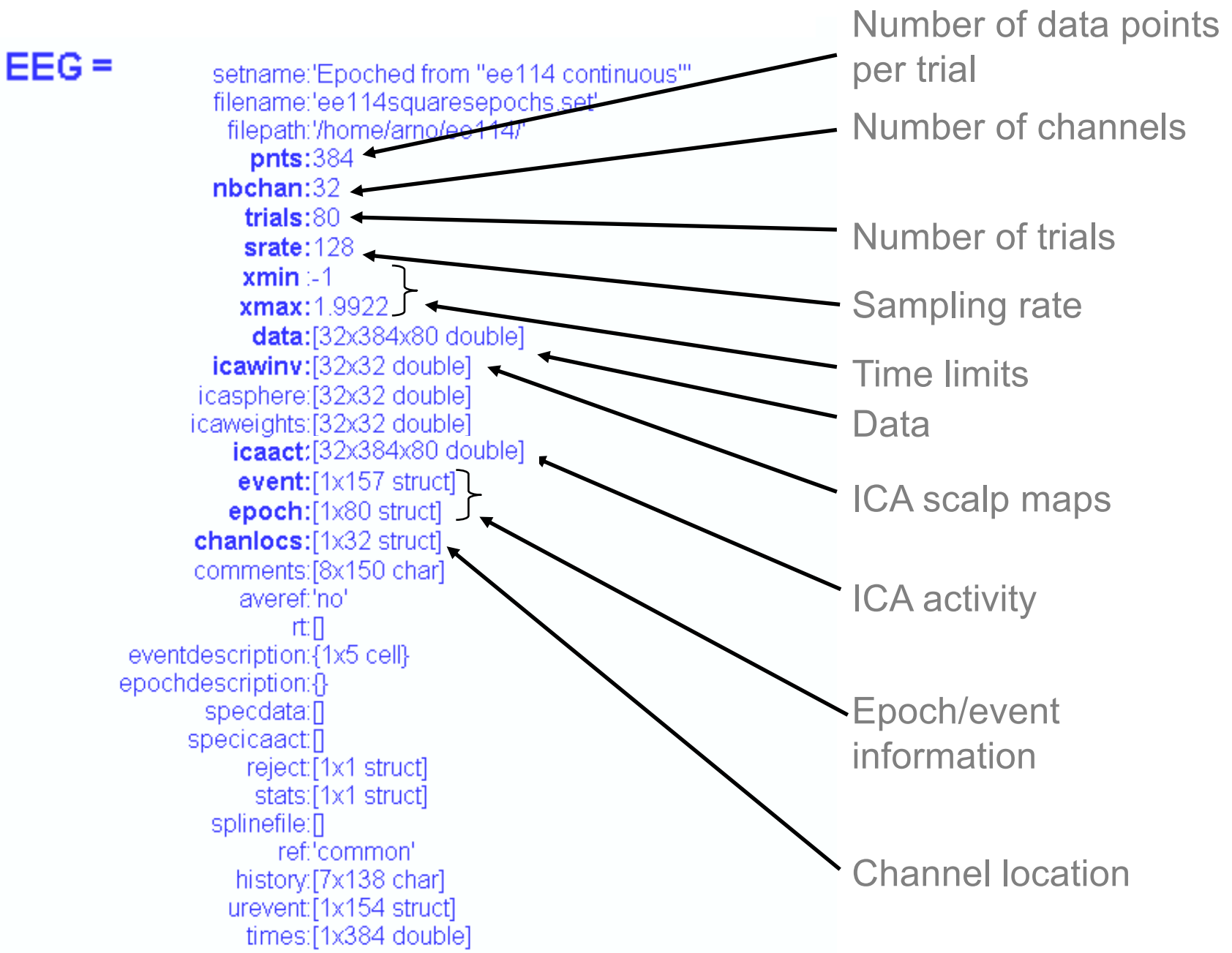
```
function [EEG, com] = pop_sample( EEG,  param1 );

com = ''; % empty history
if nargin < 2
      % pop up window if less than 2 arguments
      result       = inputdlg({ 'Enter the parameter:' }, 'Title of window', 1, { '0' })
      if length( result ) == 0 return; end;

      param1  = eval( [ '[' result{1} ']' ] ); % the brackets allow to process matlab arrays
end;

sample( EEG.data, param1); % run sample function

com = sprintf('pop_sample(EEG, %d );', param1); % return history
```



# EEGLAB Data Structures

1. EEG - root 'dataset' structure
   - .data - the dataset data (2-D, 3-D matrix)
   - .chanlocs - channel locations substructure
   - .event - data events substructure
   - .epoch - data epochs substructure
2. ALLEEG - vector of loaded EEG datasets
3. CURRENTSET - index in ALLEEG of current EEG dataset
4. STUDY - root 'studyset' structure
   - .cluster - component clustering substructure

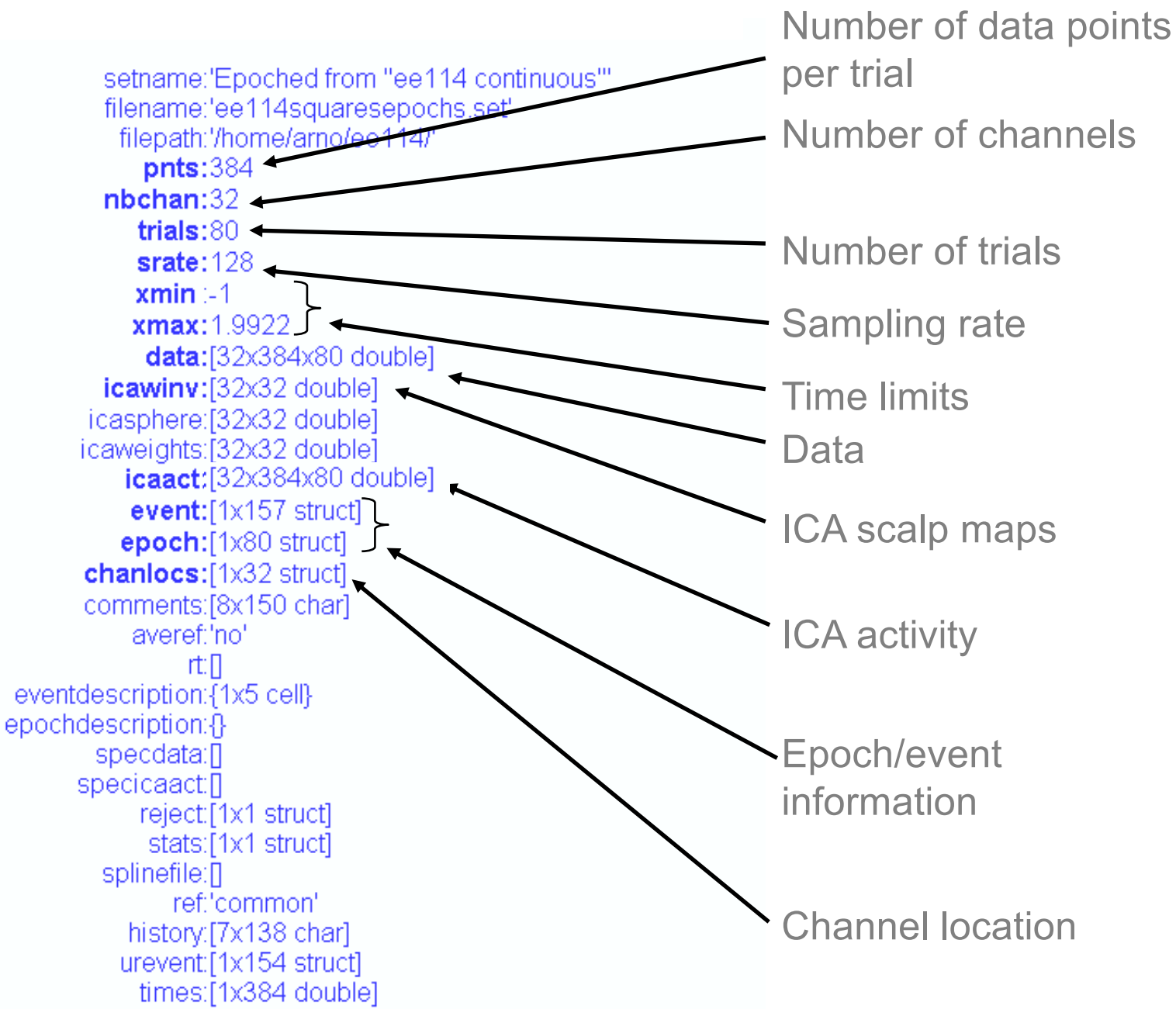
# EEG structure



# EEG structure

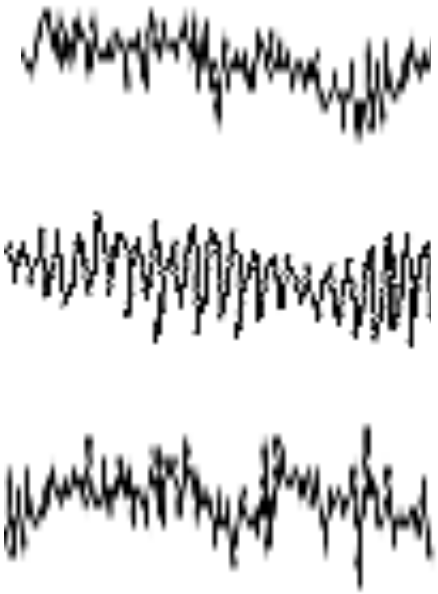
The EEG structure can be extended to include new fields

store information for future access



# Continuous data

$$\textbf{EEG.data} = \begin{bmatrix} 2.1 & 3.8 & 4.9 & 5.1 & 4.8 & 3.9 & \\ -1.3 & -2.4 & -0.5 & -0.3 & 1.4 & 2.5 & ... \\ 5.2 & 4.7 & 3.3 & 1.2 & 0.7 & 1.3 & \end{bmatrix}$$



# Data epochs

$$\textbf{EEG.data} = \begin{bmatrix} 2.1 & 3.8 & 4.9 & 5.1 & 4.8 & 3.9 & \\ -1.3 & -2.4 & -0.5 & -0.3 & 1.4 & 2.5 & ... \\ 5.2 & 4.7 & 3.3 & 1.2 & 0.7 & 1.3 & \end{bmatrix}$$ Trials 1: EEG.data(:,:,1)

$$\begin{bmatrix} 2.1 & 3.8 & 4.9 & 5.1 & 4.8 & 3.9 & \\ -1.3 & -2.4 & -0.5 & -0.3 & 1.4 & 2.5 & ... \\ 5.2 & 4.7 & 3.3 & 1.2 & 0.7 & 1.3 & \end{bmatrix}$$ Trials 2: EEG.data(:,:,2)

$$\begin{bmatrix} 2.1 & 3.8 & 4.9 & 5.1 & 4.8 & 3.9 & \\ -1.3 & -2.4 & -0.5 & -0.3 & 1.4 & 2.5 & ... \\ 5.2 & 4.7 & 3.3 & 1.2 & 0.7 & 1.3 & \end{bmatrix}$$ Trials 3: EEG.data(:,:,3)

Plot ERP for your data

```
>> figure; plot(mean(EEG.data,3)');
>> figure; plot(EEG.times, mean(EEG.data,3)');
```

# eegplugin functions

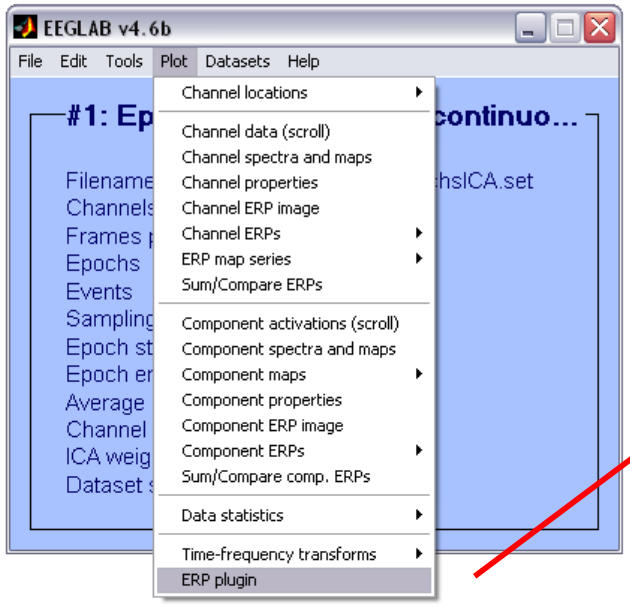
- eegplugin_erp.m function in “plugins” folder of EEGLAB

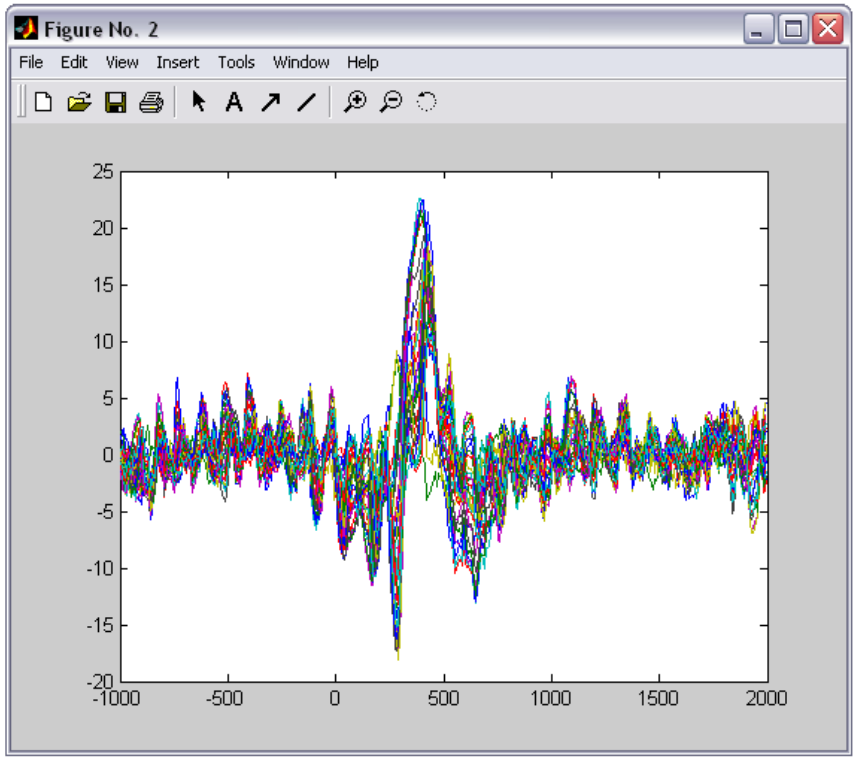
```
% eegplugin_erp() - plot ERP plugin

function eegplugin_erp( fig, try_strings, catch_strings)

% create menu
plotmenu = findobj(fig, 'tag', 'plot'); % find plot menu

% create submenu
uimenu( plotmenu, 'label', 'ERP plugin', ...
     'callback', 'figure; plot(EEG.times, mean(EEG.data,3));');
```

To test, load file in sample_data folder eeglab_data_epochs_ica.set

# eegplugin functions

```
>> eeglab
eeglab: adding "BIOSIGv0.86" plugin
eeglab: adding "eepimport1.02" plugin (see >> help eegplugin_eepimport)
eeglab: adding "bva_io1.30" plugin (see >> help eegplugin_bva_io)
eeglab: adding "ctfimport1.01" plugin (see >> help eegplugin_ctfimport)
eeglab: adding "dipfit2.0" plugin (see >> help eegplugin_dipfit2_0)
eeglab: adding plugin function "eegplugin_erp"
eeglab: adding "fmrib1.2b" plugin (see >> help eegplugin_fmrib)
eeglab: adding "icaclust1.00" plugin (see >> help eegplugin_icaclust)
eeglab: adding "iirfilt1.0" plugin (see >> help eegplugin_iirfilt)
eeglab: adding "loreta1.0" plugin (see >> help eegplugin_loreta)
eeglab: adding "newtimefreq1.00" plugin (see >> help eegplugin_ne
>>
```



# PCA plugin

```
function vers = eegplugin_pca(fig, trystrs, catchstrs)

   vers = 'pca1.00';

   % find tools menu
   menu = findobj(fig, 'tag', 'tools');

   % PCA command
    cmd = [ '[~ EEG.icawinv] = runpca(EEG.data(:,:));' ];
    cmd = [ cmd 'EEG.icaweights = pinv(EEG.icawinv);' ];
    cmd = [ cmd 'EEG.icasphere = eye(EEG.nbchan);' ];

   % create menu
   uimenu( menu, 'Label', 'Run PCA', 'CallBack', cmd, 'separator', 'on');
```

*'import data'  -> File > import data menu*
*'import epoch' -> File > import epoch menu*
*'import event' -> File > import event menu*
*'export'  -> File > export*
*'tools'  -> tools menu*
*'plot'  -> plot menu*

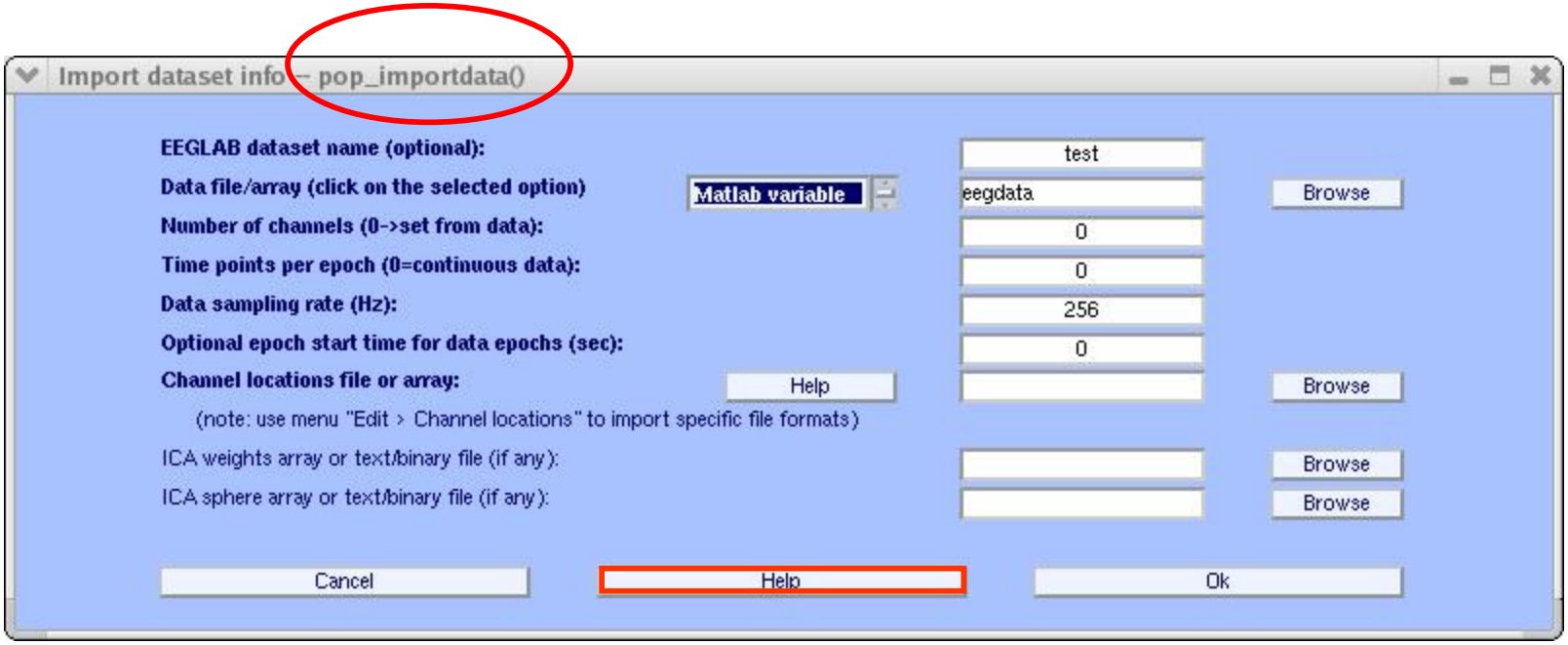
# EEGLAB documentation

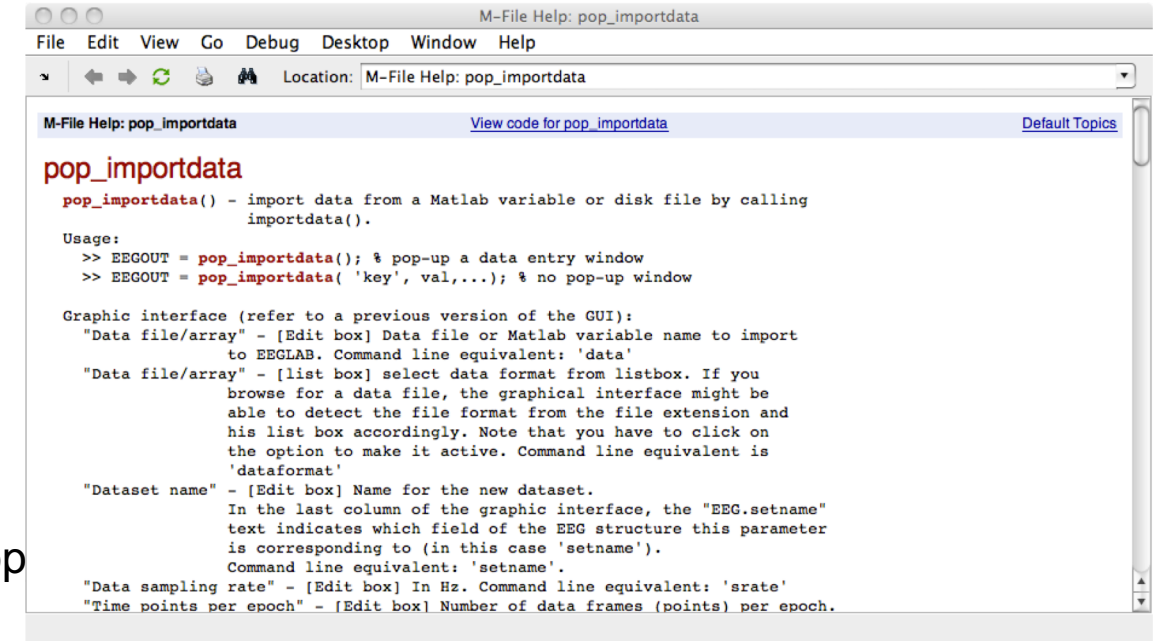
**EEGLAB Home Page** sccn.ucsd.edu/eeglab/

**EEGLAB Tutorial Index** sccn.ucsd.edu/wiki/EEGLAB

- 200 pages of tutorial (including “how to” for plugins) WEB or PDF
- Function documentation (next slide)
- Send questions to the mailing list eeglablist@sccn.ucsd.edu
  (or search mailing list archive using google)
- Bug submission http://sccn.ucsd.edu/eeglab/bugzilla
- Email us (suggestions) eeglab@sccn.ucsd.edu
- Workshop with practicum every year

# Help message




>> help pop

# Exercice

Write a plugin to plot ERPs

1. Save eegplugin_erp.m in the *plugins* folder of EEGLAB
2. Restart EEGLAB
3. Load epoched EEGLAB dataset
4. Use plugin menu

# Using EEGLAB history for basic scripting

**Task 1**

Create a script from ‘eegh’ output

**Task 2**

Adapt your script with variables

**Task 3**

Create a Matlab function

**Task 4**

**Exercise...**

# Using EEGLAB history for basic scripting

**Task 1**

Create a script from ‘eegh’ output

**Task 2**

Adapt your script with variables

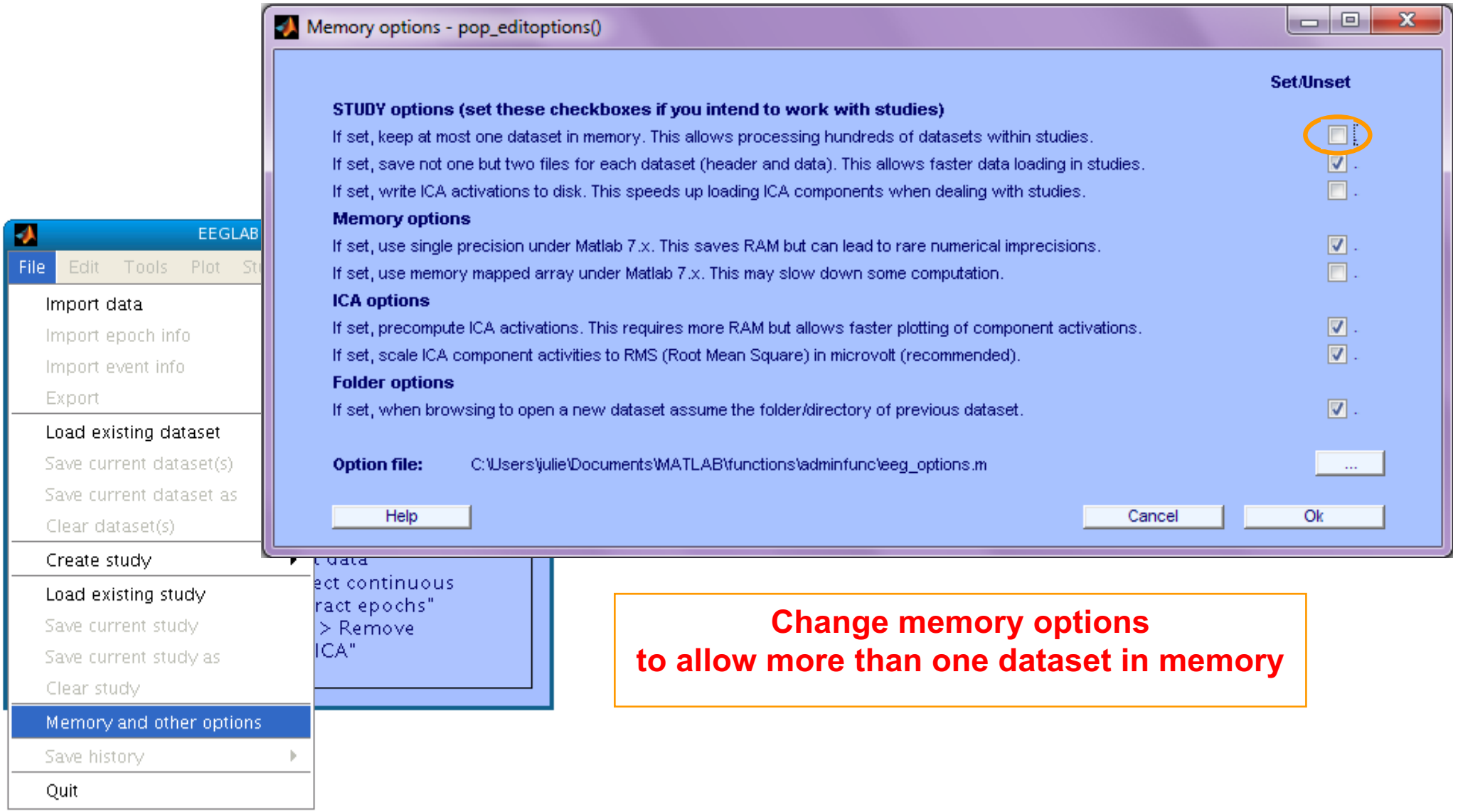
**Task 3**

Create a Matlab function

**Task 4**

**Exercise…**

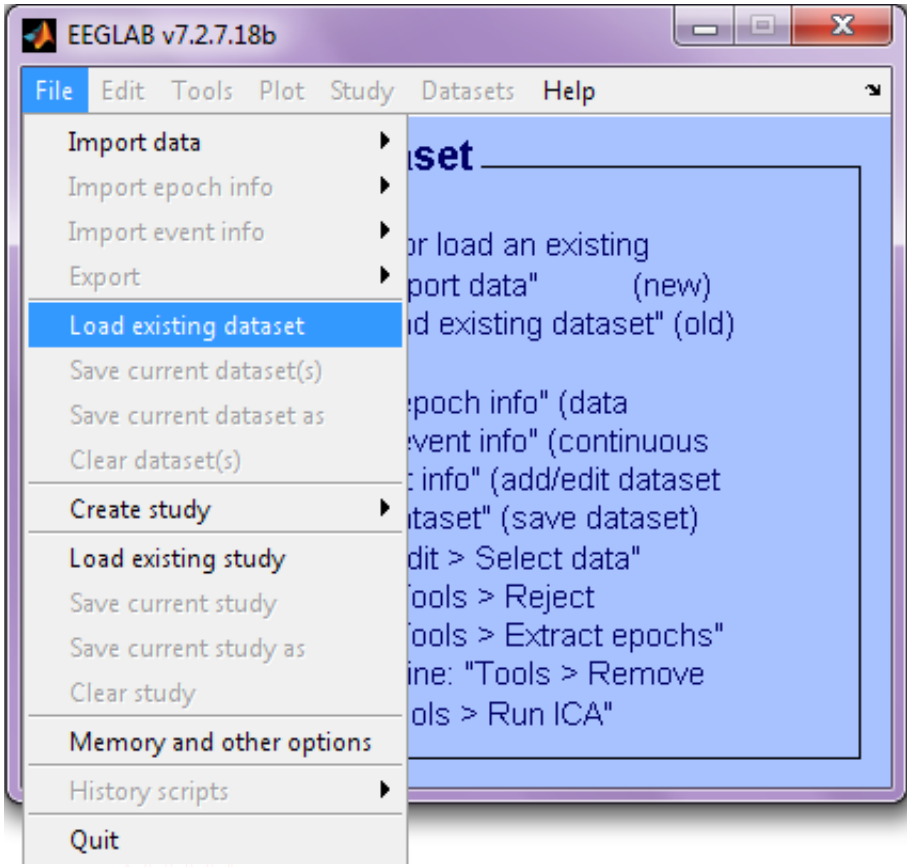
# Memory options

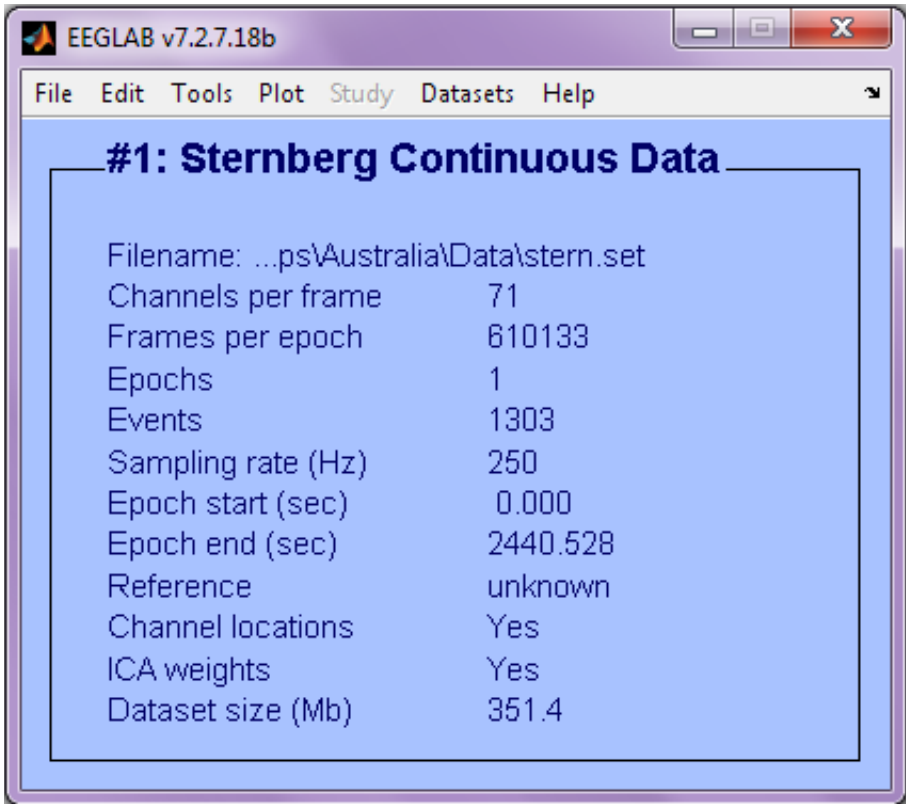



**Change memory options**
**to allow more than one dataset in memory**

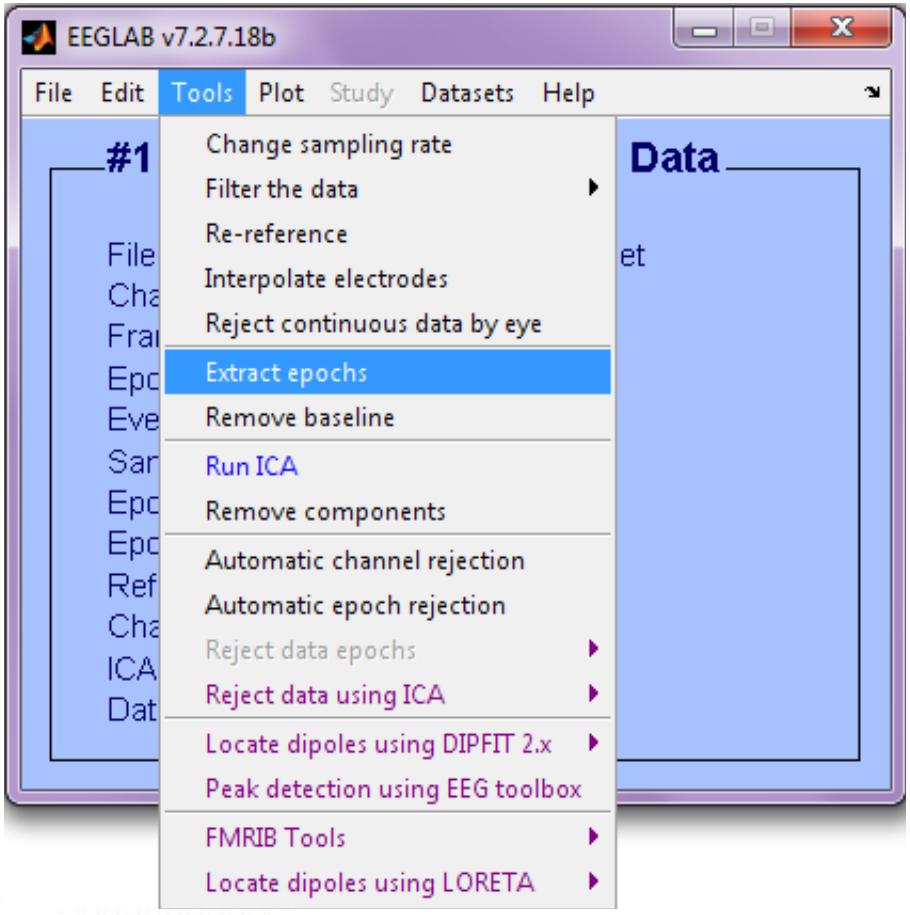
# Create a script from ‘eegh’ output

## Start by loading a continuous dataset

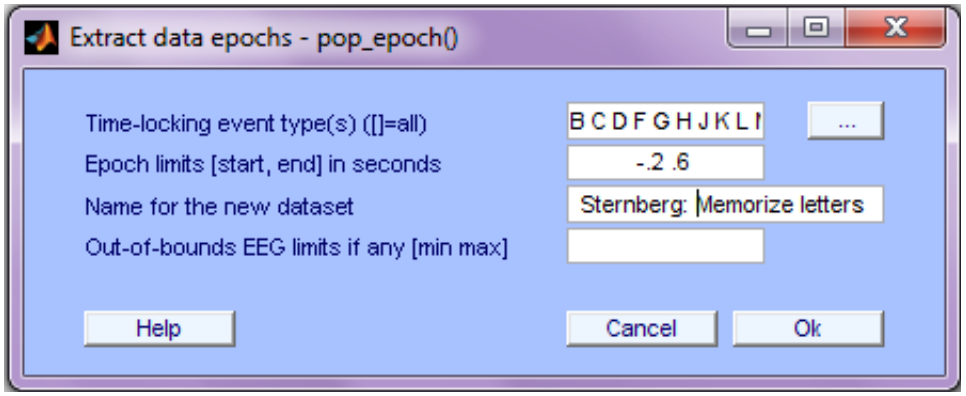


# Create a script from ‘eegh’ output

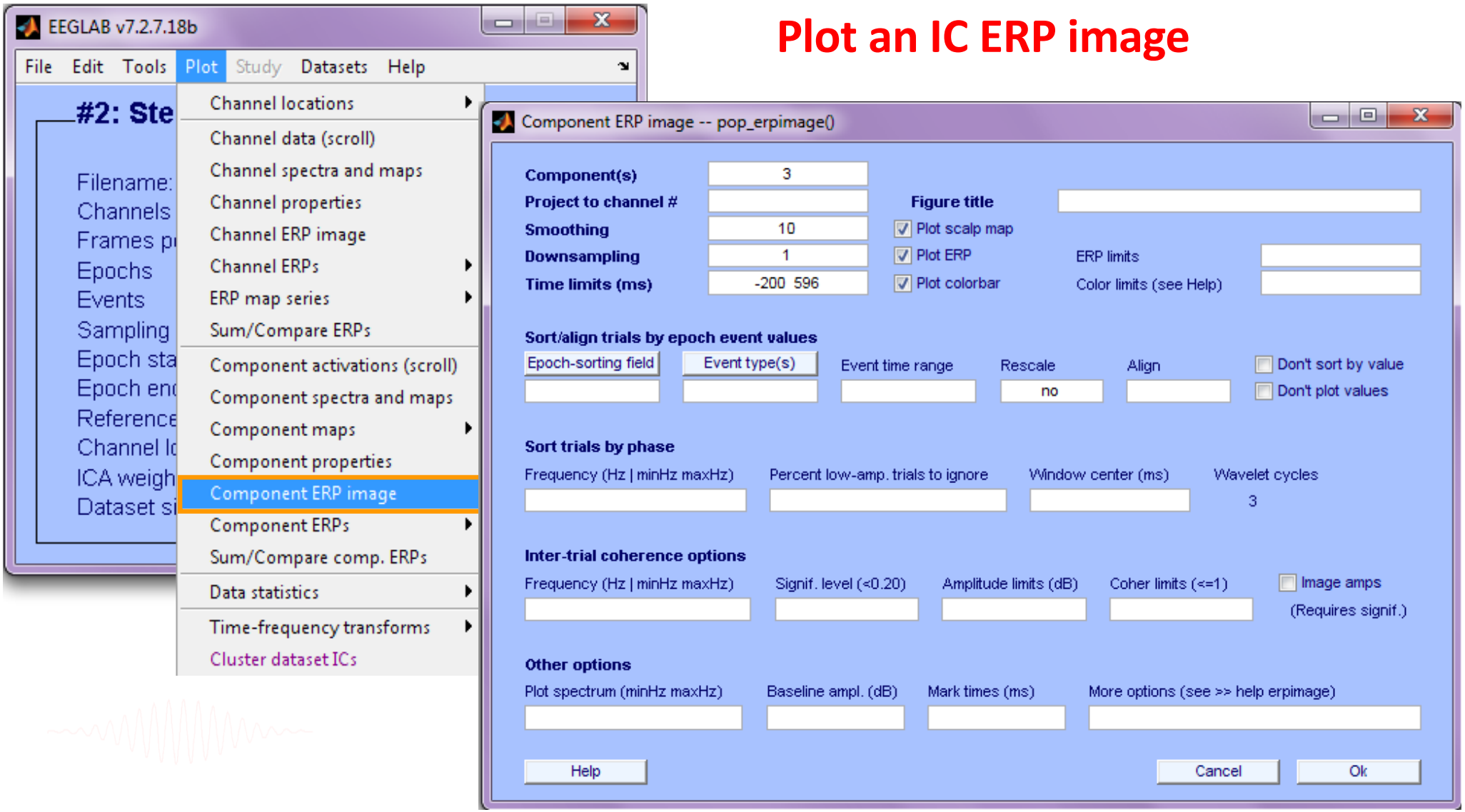



## Epoch on Memorize letters

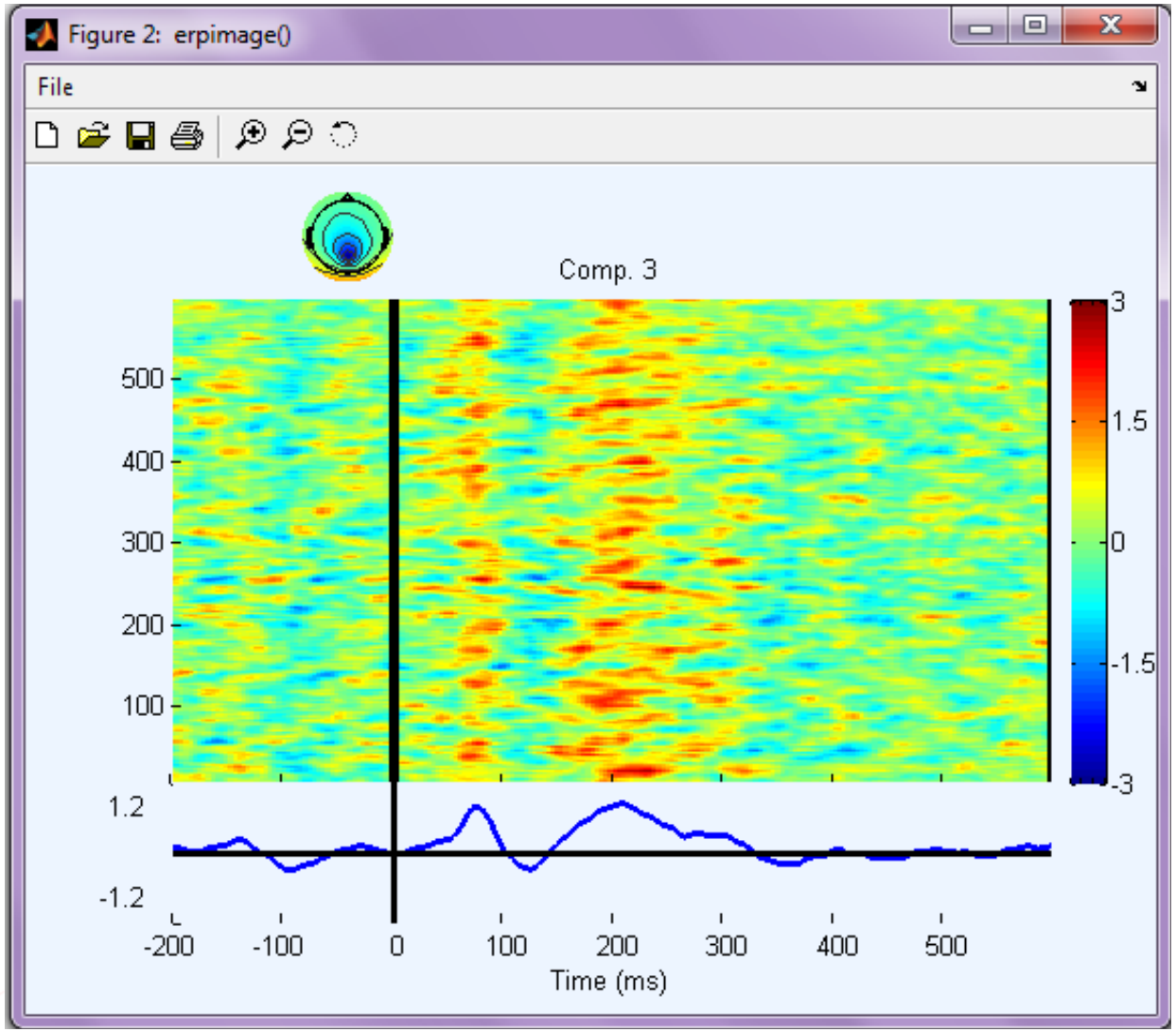


# Create a script from ‘eegh’ output

## Plot an IC ERP image



# Create a script from ‘eegh’ output



# Retrieve commands from eegh

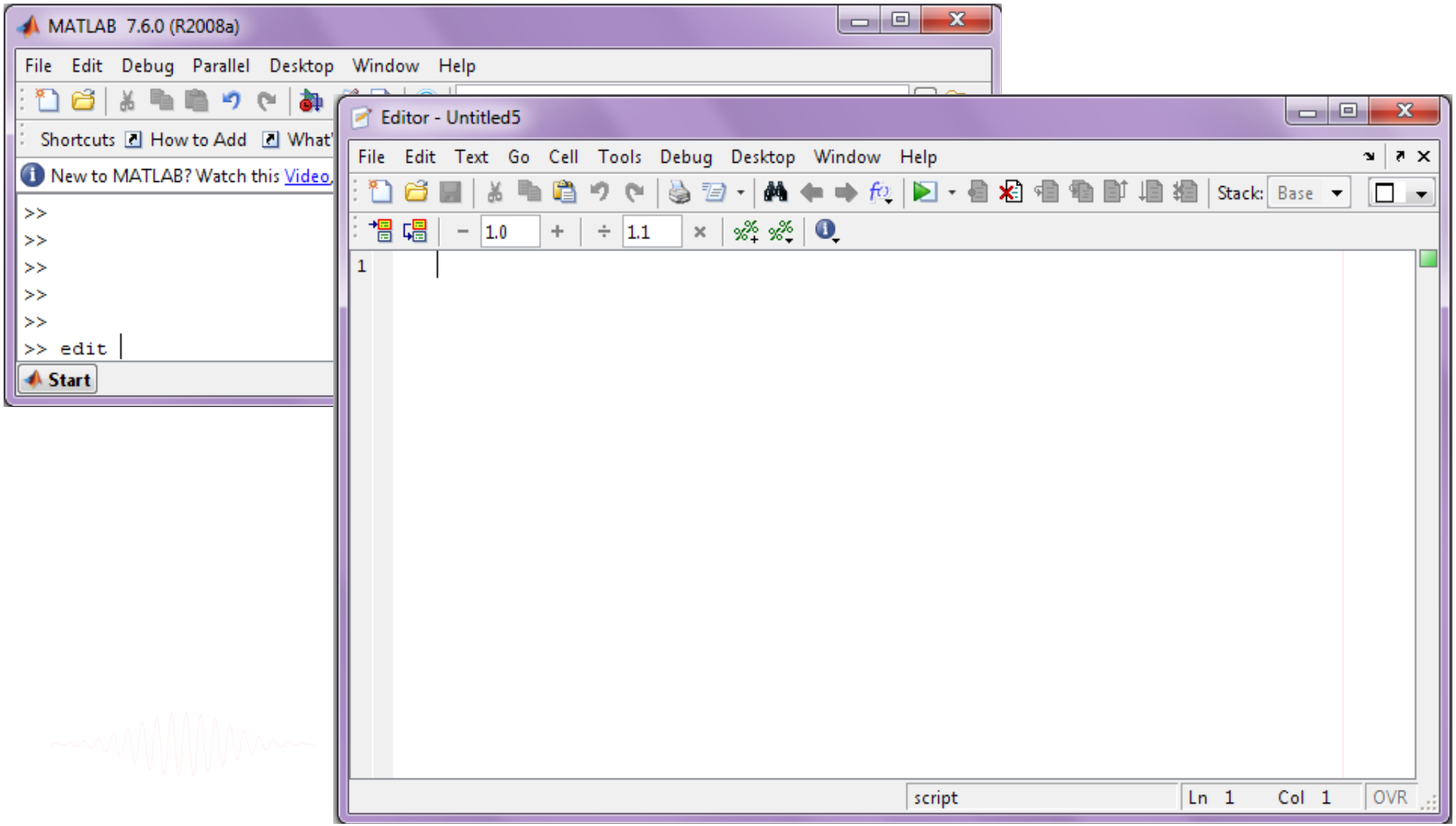
**Write a script to do this:**

```
>> eegh
```

# Retrieve commands from eegh

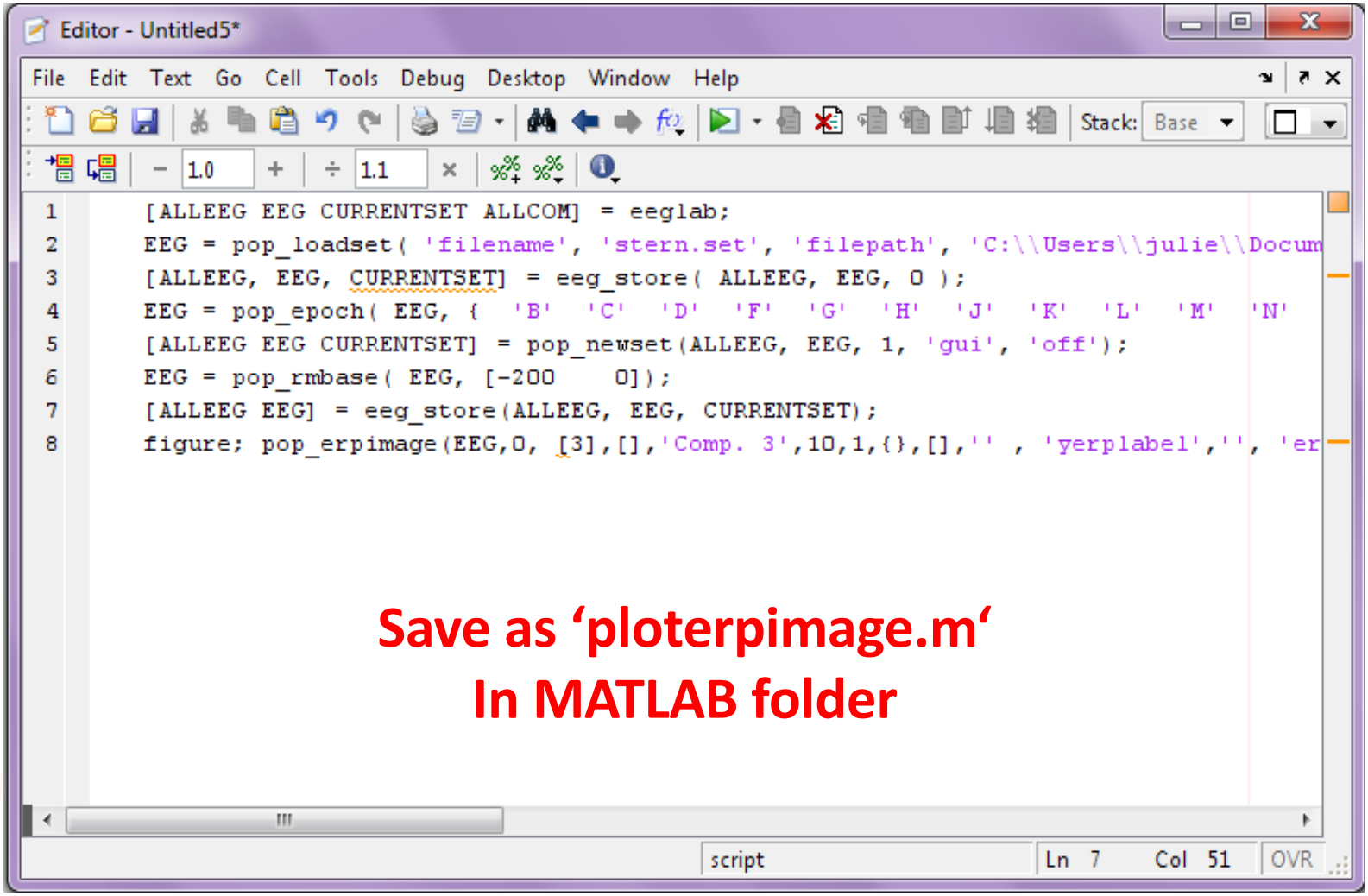
```
>> eegh
[ALLEEG EEG CURRENTSET ALLCOM] = eeglab;

EEG = pop_loadset('filename', 'stern_125Hz.set');
[ALLEEG EEG CURRENTSET] = eeg_store(ALLEEG, EEG, 0);

EEG = pop_epoch( EEG, {'B'  'C'  'D' ... }, [-0.2 0.6], 'newname',
'Memorize epochs', 'epochinfo', 'yes');
[ALLEEG EEG CURRENTSET] = eeg_store(ALLEEG, EEG, 1);
EEG = pop_rmbase( EEG, [-200 0]);
[ALLEEG EEG] = eeg_store(ALLEEG, EEG, CURRENTSET);

figure; pop_erpimage(EEG,0, [3],[],'Comp.
3',10,1,{},[],'','yerplabel', '', 'erp', 'on', 'cbar',
'on','topo',{mean(EEG.icawinv(:,[3]),2) EEG.chanlocs EEG.chaninfo
});
```

# Create a Matlab script



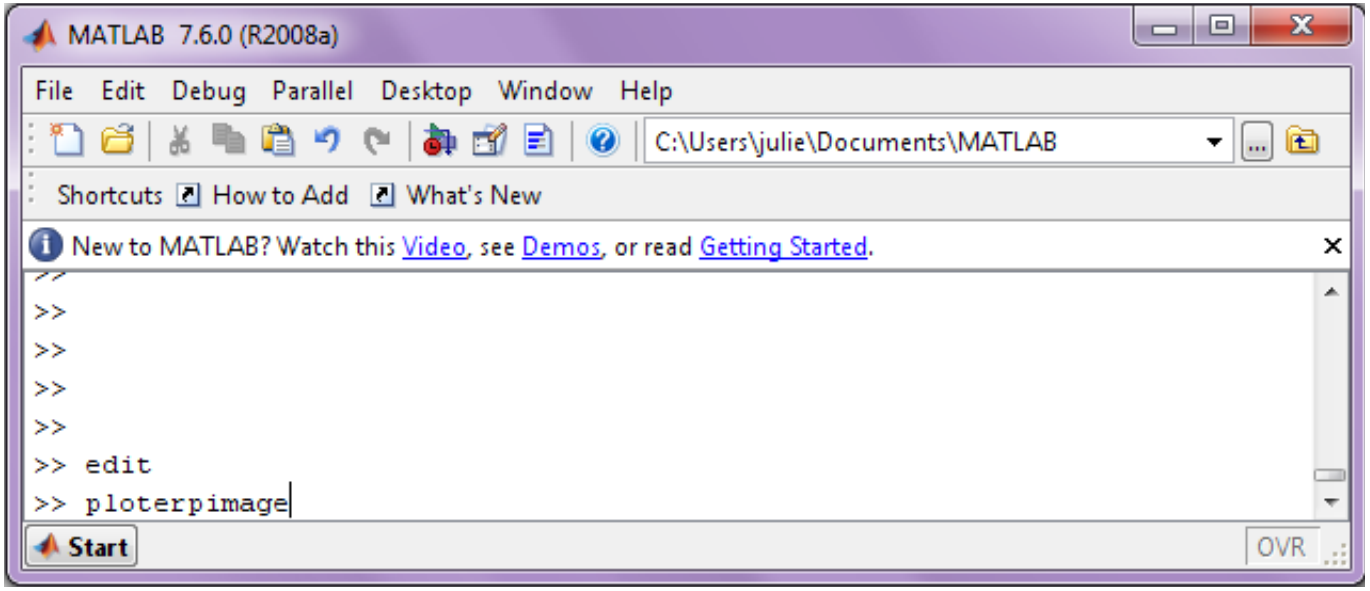
# Create a Matlab script

## Copy and paste from Matlab window:




## Save as ‘ploterpimage.m‘
## In MATLAB folder

# Run your new script

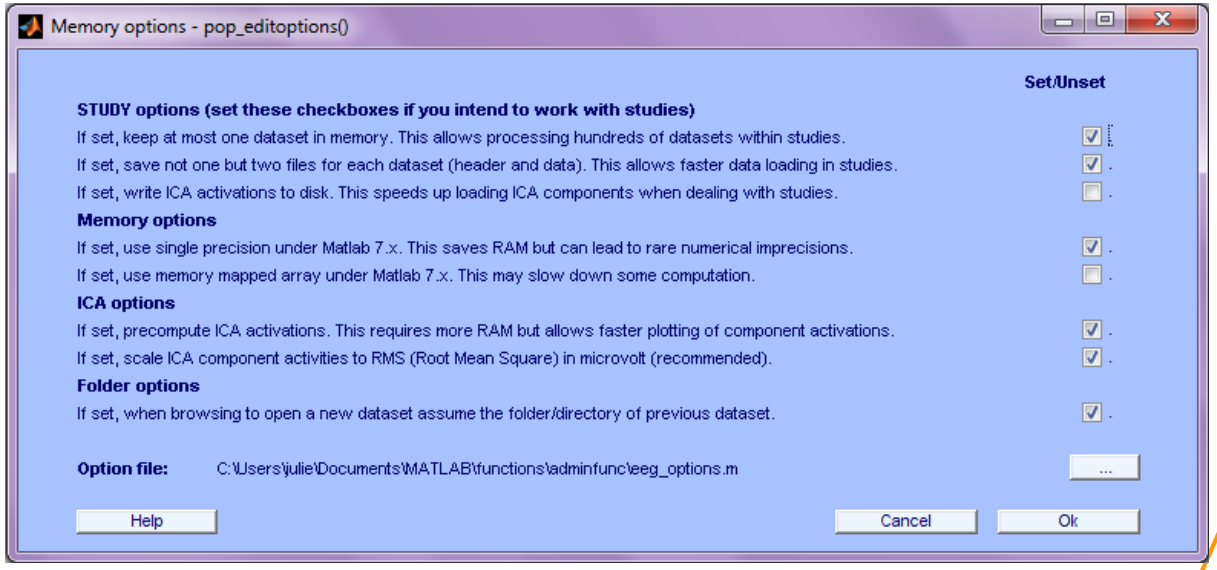


# Exercise page 1

```
>> eeglab

% load dataset stern_125hz.set in data folder
% epoch on 'memorize letter' B, C, etc...
% plot erpimage for component 3

>> eegh

% open Matlab editor

>> edit

% copy & paste eegh results into a new
% file and save it (ploterpimage.m)

>> clear
>> close all
>> ploterpimage
>> eeglab redraw
```

# Advanced Scripting in EEGLAB

# STUDY scripts

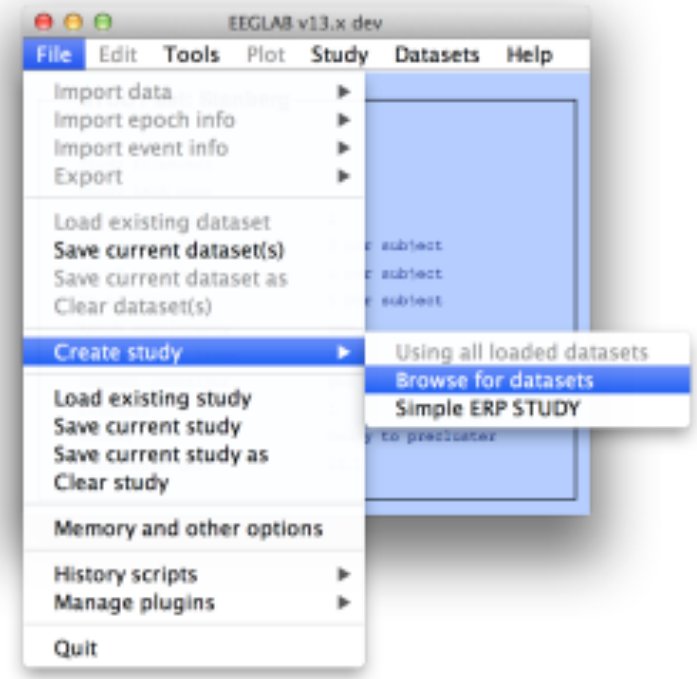


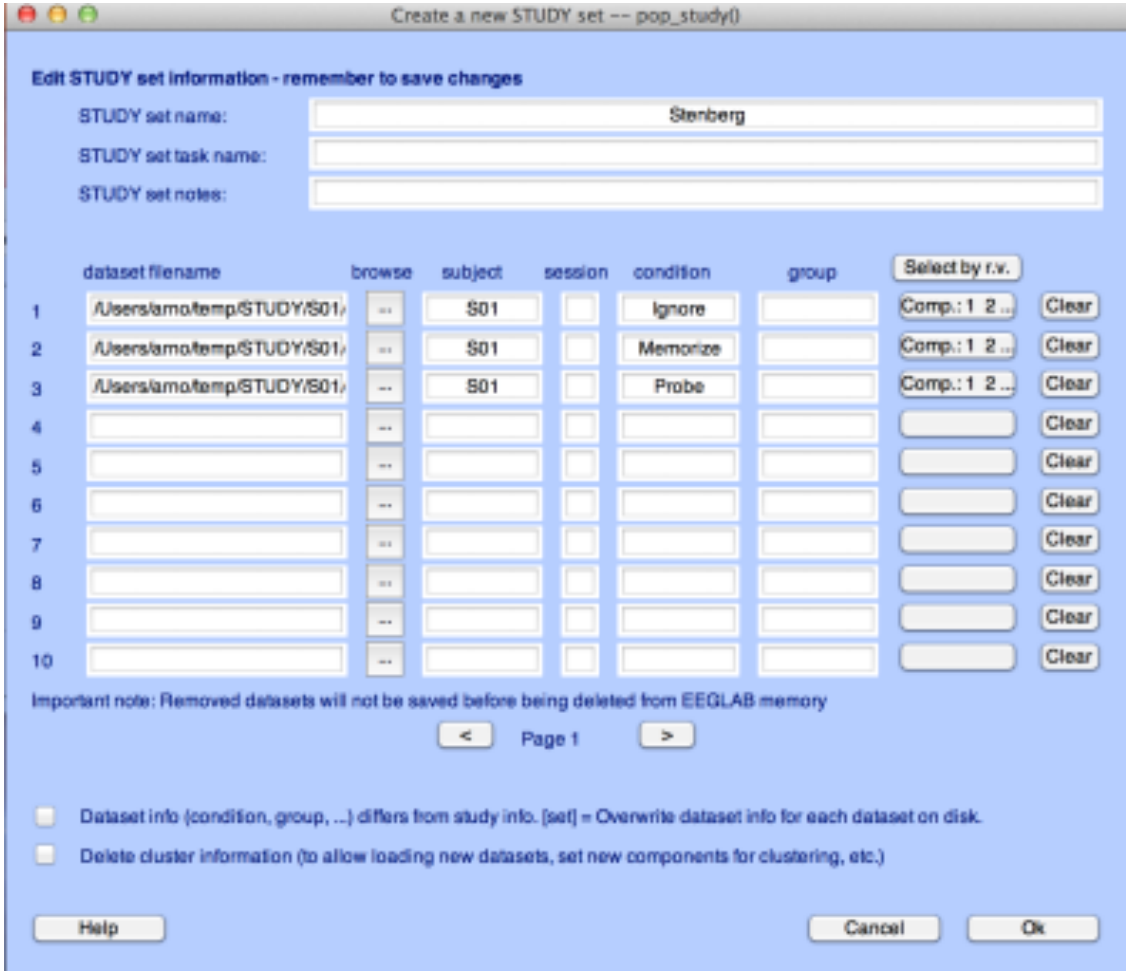

Most important option:

- Allows only one dataset to be loaded at once.
- Most STUDYs are too big to have all data loaded at once.

```
% Set memory options:
pop_editoptions( 'option_storedisk', 1);
```

# Edit dataset info




```
[STUDY ALLEEG] = std_editset( STUDY, ALLEEG, 'name','Stenberg','commands',{{'index' 1
'load' '/data/STUDY/S01/Ignore.set'} {'index' 2 'load' '/data/S01/Memorize.set'} {'index'
3 'load' '/data/S01/Probe.set'} {'index' 1 'subject' 'S01'} {'index' 2 'subject' 'S01'}
{'index' 3 'subject' 'S01'} {'index' 1 'condition' 'Ignore'} {'index' 2 'condition'
'Memorize'} {'index' 3 'condition' 'Probe'}},'updatedat','off' );
```

# Looking a the function that create STUDY

```
[STUDY ALLEEG] = std_editset( STUDY, ALLEEG, 'name','Stenberg','commands',
     {{'index' 1 'load' '/data/STUDY/S01/Ignore.set'} …
     {'index' 2 'load' '/data/S01/Memorize.set'} …
     {'index' 3 'load' '/data/S01/Probe.set'} …
     {'index' 1 'subject' 'S01'} …
     {'index' 2 'subject' 'S01'} …
     {'index' 3 'subject' 'S01'} …
     {'index' 1 'condition' 'Ignore'} …
     {'index' 2 'condition' 'Memorize'} …
     {'index' 3 'condition' 'Probe'}},'updatedat','off' );
```

```
[STUDY ALLEEG] = std_editset( STUDY, ALLEEG, 'name','Stenberg','commands',
     {{'index' 1 'load' '/data/STUDY/S01/Ignore.set' 'subject' 'S01' 'condition' 'Ignore'} …
     {'index' 2 'load' '/data/S01/Memorize.set' 'subject' 'S01' 'condition' 'Memorize'} …
     {'index' 3 'load' '/data/S01/Probe.set' 'subject' 'S01' 'condition' 'Probe'} …
     },'updatedat','off' );
```

# Exercice

```
[STUDY ALLEEG] = std_editset( STUDY, ALLEEG, 'name','Stenberg','commands',
     {{'index' 1 'load' '/data/STUDY/S01/Ignore.set' 'subject' 'S01' 'condition' 'Ignore'} …
     {'index' 2 'load' '/data/S01/Memorize.set' 'subject' 'S01' 'condition' 'Memorize'} …
     {'index' 3 'load' '/data/S01/Probe.set' 'subject' 'S01' 'condition' 'Probe'} …
     },'updatedat','off' );
```

**If not present, add it by hand because some dataset might not have it**

1- Start EEGLAB and import the 3 datasets for Subject 1 (Ignore.set, Memorize.set and Probe.set) in a STUDY (menu Tools > Create STUDY > Browse for datasets)

2- Look in the history (type eegh)

3- Copy to a script, add “eeglab redraw” at the end of your script

4- Restart Matlab, execute the script, look at your STUDY info and design (menu *STUDY* > *Edit STUDY* info and *STUDY* > *Select/Edit STUDY design*)

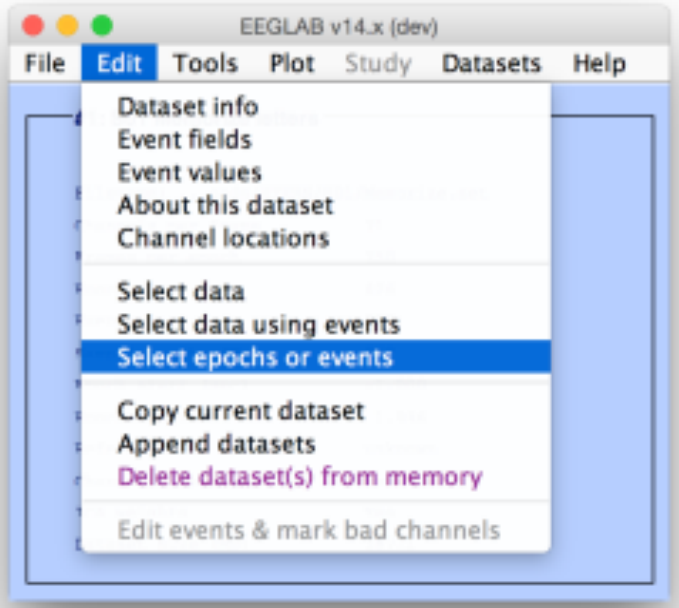
5- Modify the script to import subject 1 to 4

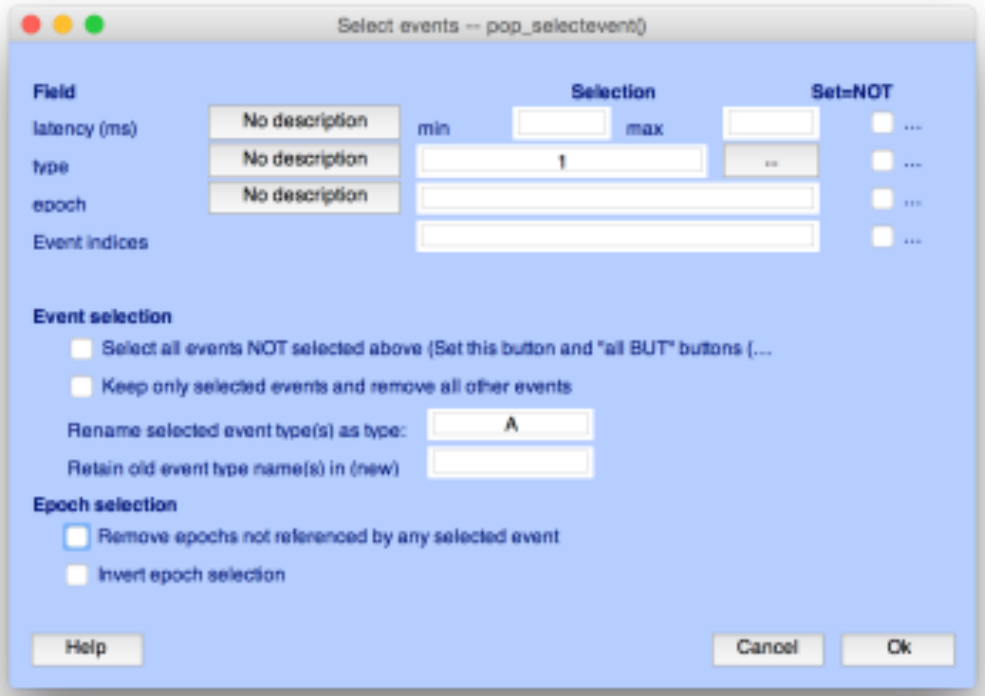
6- Restart Matlab, execute the script, look at your STUDY info and design

# Redefining events

## Adjusting latencies

```
for iEvent = 1:length(EEG.event)
    % shift by 16 samples (or 53.3ms at 200Hz) due to filter delay
    EEG.event(iEvent).latency = EEG.event(iEvent).latency + 16;
end;
EEG.saved = 'no';
[ALLEEG EEG CURRENTSET] = eeg_store(ALLEEG, EEG);
eeglab redraw
```




# Renaming events

```
for iDat = 1:length(ALLEEG)
    EEG = ALLEEG(iDat);
    % rename events
    EEG = pop_selectevent( EEG, 'type',1,'renametype','A','deleteevents','off');
    EEG = pop_selectevent( EEG, 'type',2,'renametype','B','deleteevents','off');
    EEG = pop_selectevent( EEG, 'type',3,'renametype','C','deleteevents','off');
    EEG = pop_selectevent( EEG, 'type',4,'renametype','D','deleteevents','off');
    EEG = pop_selectevent( EEG, 'type',8,'renametype','rt','deleteevents','off');
    EEG.saved = 'no';
    [ALLEEG EEG CURRENTSET] = eeg_store(ALLEEG, EEG, iDat);
end;
eeglab redraw
```


```
EEG.event(4)
        type: 'square'
    position: 2
     latency: 424
     urevent: 1
       epoch: 2

EEG.event(1)
        type: 'square'
    position: 2
     latency: 129
     urevent: 1
       epoch: 1

EEG.urevent(1)
        type: 'square'
    position: 2
     latency: 129
```

# Redefining events

```
for iDat = 1:length(ALLEEG)
    TMPEEG = eeg_checkset(ALLEEG(iDat), 'loaddata'); % load data

    % 'B'  'C'  'D'  'F'  'G'  'H'  'J'  'K'  'L' ... -> Memorize
    % 'gB' 'gC' 'gD' 'gF' 'gG' 'gH' 'gJ' 'gK' 'gL' ... -> Ignore
    for iEvent = 1:length(TMPEEG.event)
          prevEvent = TMPEEG.event(iEvent).urevent-2;
          if prevEvent > 2 && TMPEEG.urevent(prevEvent).type(1) == 'g'
             TMPEEG.event(iEvent).prevEvent = 'ignore';
          else TMPEEG.event(iEvent).prevEvent = 'memorize';
          end;
    end;

    TMPEEG.saved = 'no'; % tag as not saved
    ALLEEG(iDat) = pop_saveset(TMPEEG, 'savemode', 'resave'); % resave data
end;

STUDY = std_maketrialinfo(STUDY, ALLEEG); % update STUDY
STUDY.saved = 'no';
[STUDY EEG] = pop_savestudy( STUDY, EEG, 'savemode', 'resave'); % resave STUDY
```

Precomputed files need to be recomputed after changing events.

# Load dataset info from commandline

```
% Create Stern STUDY
[ALLEEG EEG CURRENTSET ALLCOM] = eeglab;
pop_editoptions( 'option_storedisk', 1);
subjects = {'S01' 'S02' 'S03' 'S04' 'S05' 'S06' 'S07' 'S08' 'S09' 'S10' 'S11' 'S12'};
filepath = '/Users/arno/temp/STUDY'; % XXXXX Change path here XXXXX
if ~exist(filepath), error('You need to change the path to the STUDY'); end;
commands = {}; % initialize STUDY dataset list

% Loop through all of the subjects in the study to create the dataset
for loopnum = 1:length(subjects) %for each subject
    IgnoreFile   = fullfile(filepath, subjects{loopnum}, 'Ignore.set');
    MemorizeFile = fullfile(filepath, subjects{loopnum}, 'Memorize.set');
    ProbeFile    = fullfile(filepath, subjects{loopnum}, 'Probe.set');
    commands = {commands{:} ...
        {'index' 3*loopnum-2 'load' IgnoreFile   'subject' subjects{loopnum} 'condition' 'Ignore'} ...
        {'index' 3*loopnum-1 'load' MemorizeFile 'subject' subjects{loopnum} 'condition' 'Memorize'} ...
        {'index' 3*loopnum   'load' ProbeFile    'subject' subjects{loopnum} 'condition' 'Probe'}};
end;
% Uncomment the line below to select ICA components with less than 15% residual variance
% commands = {commands{:} {'dipselect', 0.15}};
[STUDY, ALLEEG] = std_editset(STUDY, ALLEEG, 'name','Sternberg','commands',commands,'updatedat','on');

% Update workspace variables and redraw EEGLAB
CURRENTSTUDY = 1; EEG = ALLEEG; CURRENTSET = [1:length(EEG)];
[STUDY, ALLEEG] = std_checkset(STUDY, ALLEEG);
eeglab redraw
```

# STUDY structure

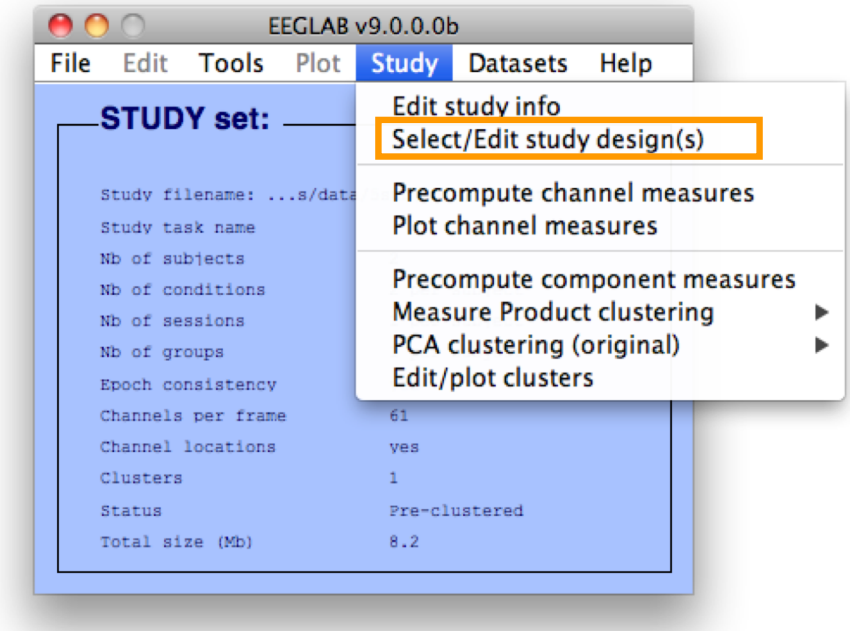
```
STUDY =
            name: 'Sternberg'
            task: 'Sternberg'
     datasetinfo: [1x39 struct]
           notes: ''
        filename: 'stern.study'
        filepath: 'C:\Users\julie\Documents\Workshops\Finland\STUDY'
         history: [1x7332 char]
         subject: {1x13 cell}
           group: {''}
         session: []
       condition: {'ignore'  'memorize'  'probe'}
          design: [1x1 struct]
             etc: [1x1 struct]
        preclust: [1x1 struct]
         cluster: [1x1 struct]
         changrp: [1x71 struct]
           saved: 'yes'
```

# Understanding STUDY structure

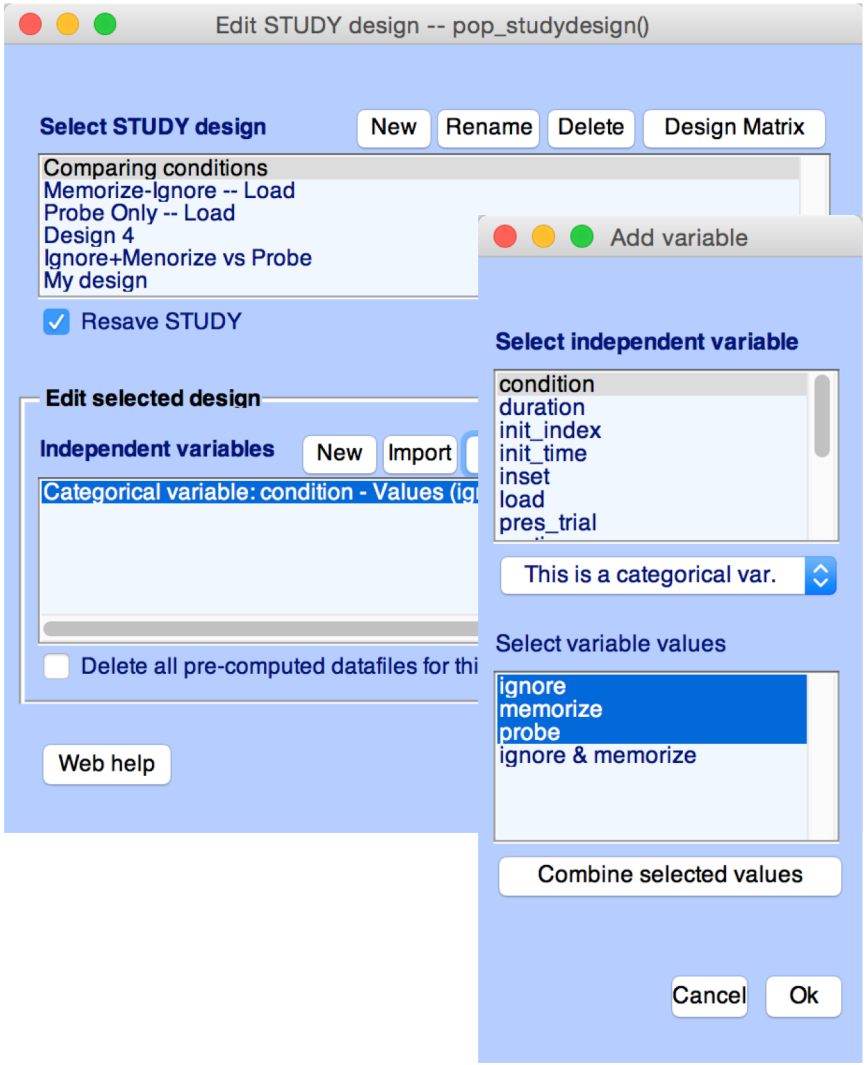
```
>> STUDY.datasetinfo(11) % access dataset 11
ans =
       filepath: [1x61 char]
       filename: 'S04.set'
        subject: 'S04'
        session: []
      condition: ''
          group: ''
      trialinfo: 1x350 struct
          index: 11
          comps: [1x24 double]
```

Subject 4!

```
>> trialinfo(163) % access trial 163
ans =
        stimtype: 'Memorize'
         latency: 13201
        duration: 0
        ...
```




# Select subjects




```
STUDY = std_makedesign(STUDY, ALLEEG, 3,
'variable1','condition',
'variable2','',
'name','Design 3',
'values1',{'ignore' 'memorize' 'probe'},
'subjselect',{'S02' 'S03'},
'dataselect',{'condition' {'probe'}});
```

# STUDY design structure

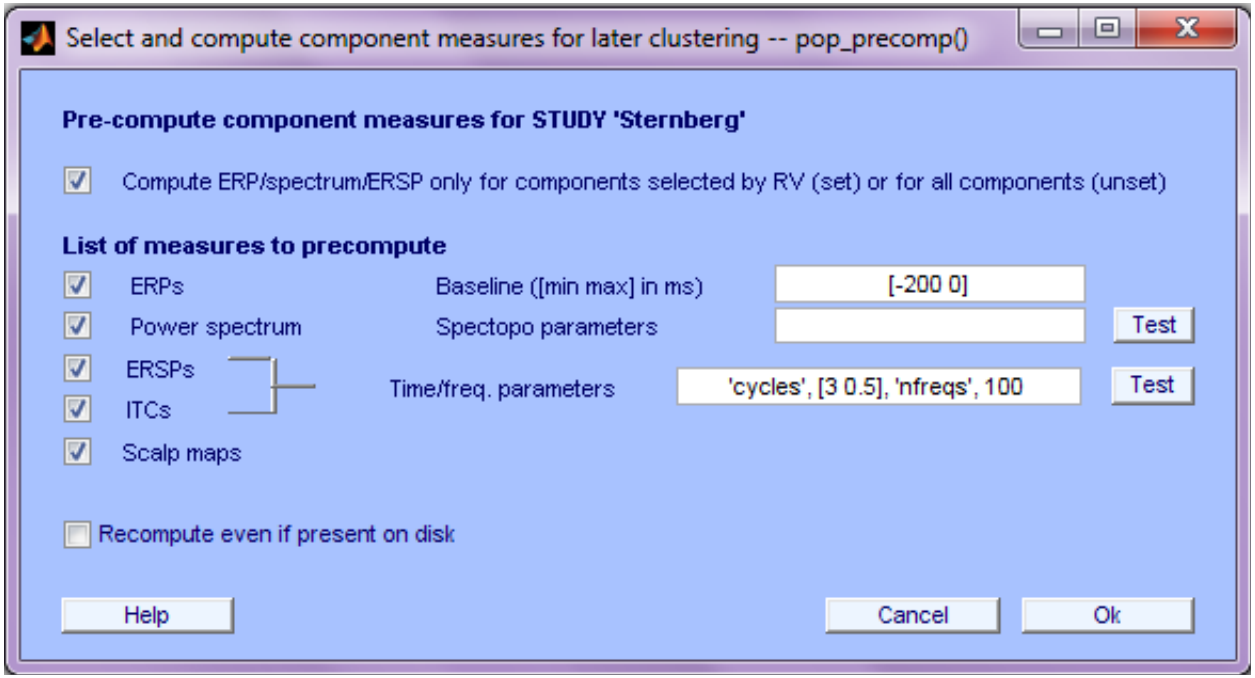
```
STUDY.design(1)

ans =

           name: 'Design 1 - compare letter types'
       variable: [1x2 struct]
          cases: [1x1 struct]
        include: {}
          cells: [1x39 struct]
```

Exploding the contents of each of these sub-structures, we obtain

```
           name: 'Design 1 - light and audio all subjects'

       variable: [1x2 struct]
            (1).label  : 'condition'
            (1).pairing: 'on'
            (1).value  : {'ignore'  'memorize'  'probe'}
            (2).label  : ''
            (2).pairing: 'off'
            (2).value  : {}

         cases: [1x1 struct]
            label: 'subject'
            value: {'S01'  'S02'  'S03'  'S04'  'S05'  'S06' }
```

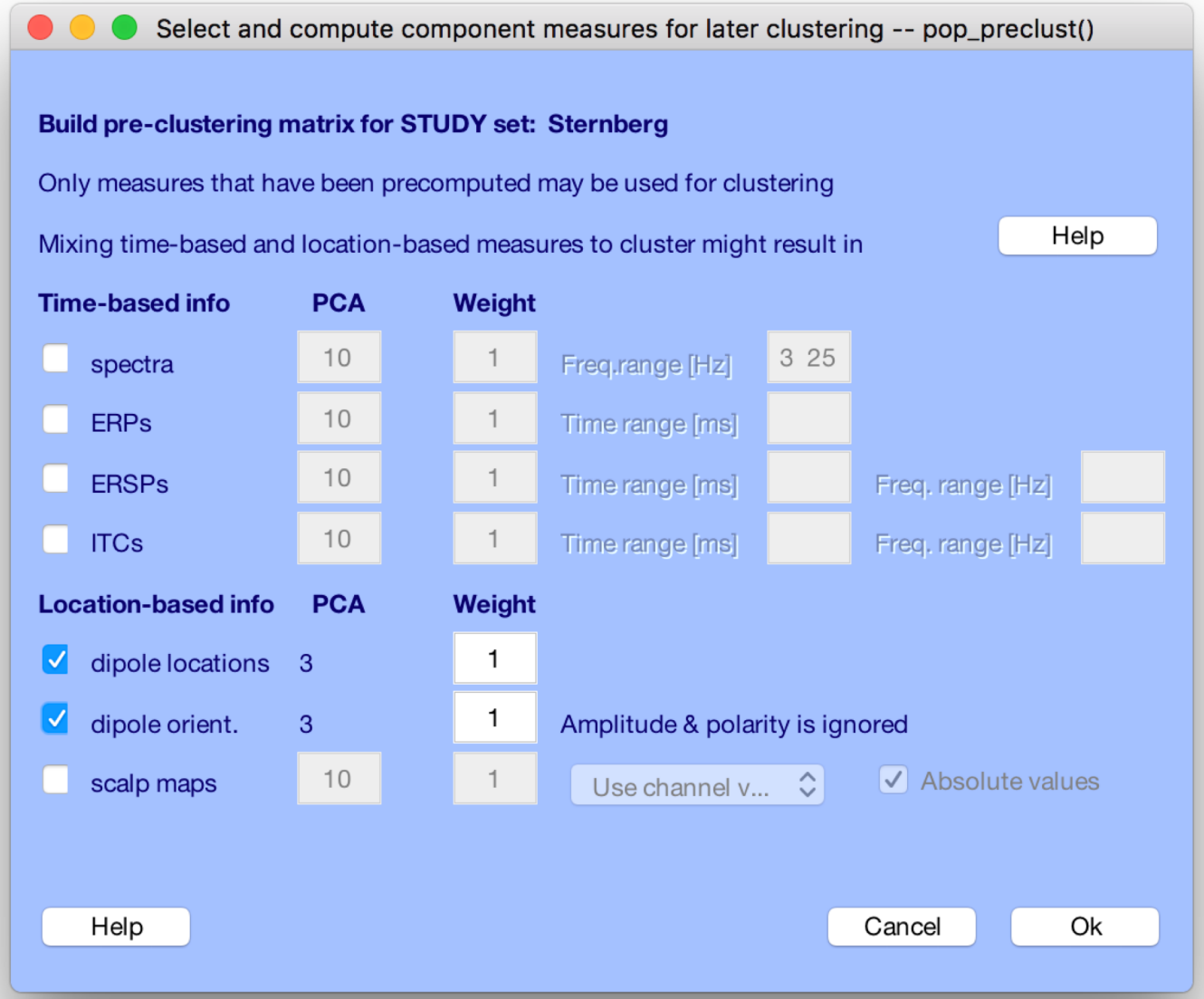
# Precompute data measures

**TIP: Compute all measures so you can test different combinations for clustering**



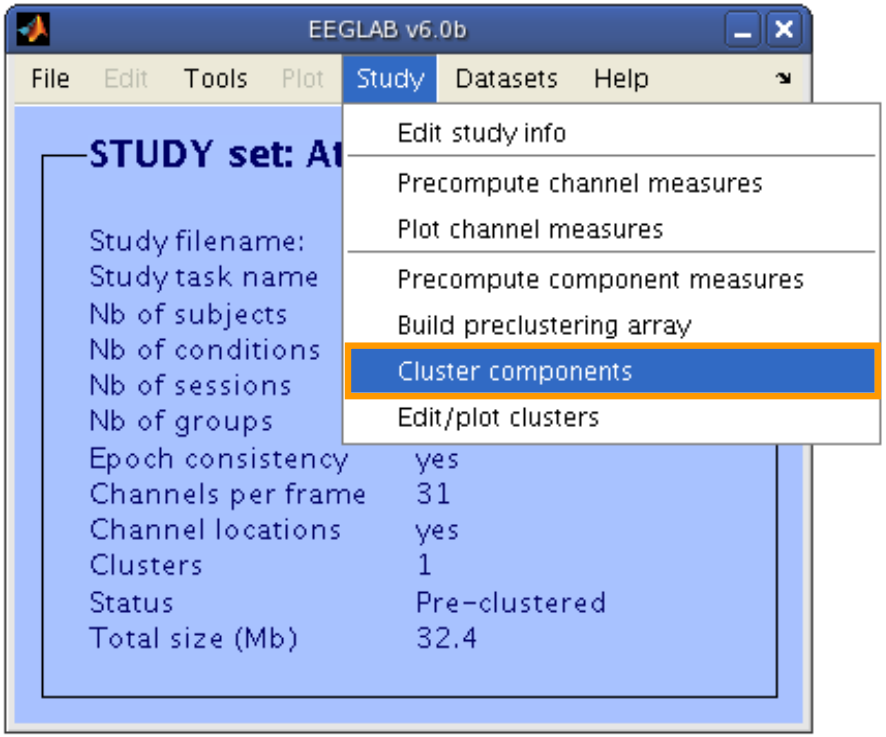

```
[STUDY ALLEEG] = std_precomp(STUDY, ALLEEG, 'components', 'erp', ...
'on', 'rmbase',[-200 0] , 'scalp', 'on', 'spec', 'on', ...
'specparams',{}, 'ersp', 'on', 'erspparams', {'cycles' [3 0.5] ...
'nfreqs',100, 'freqs',[3 70] }, 'itc', 'on');
```

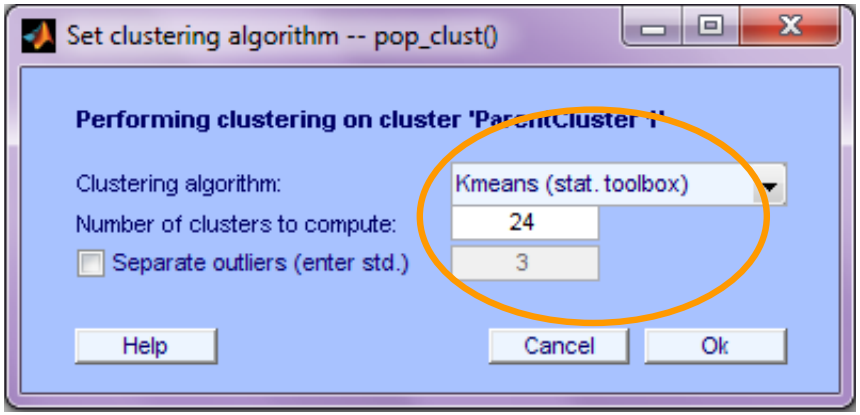
# Precluster the data




```
[STUDY ALLEEG] = std_preclust(STUDY, ALLEEG, 1, {'dipoles','weight',10},
{'moments','weight',10});
```

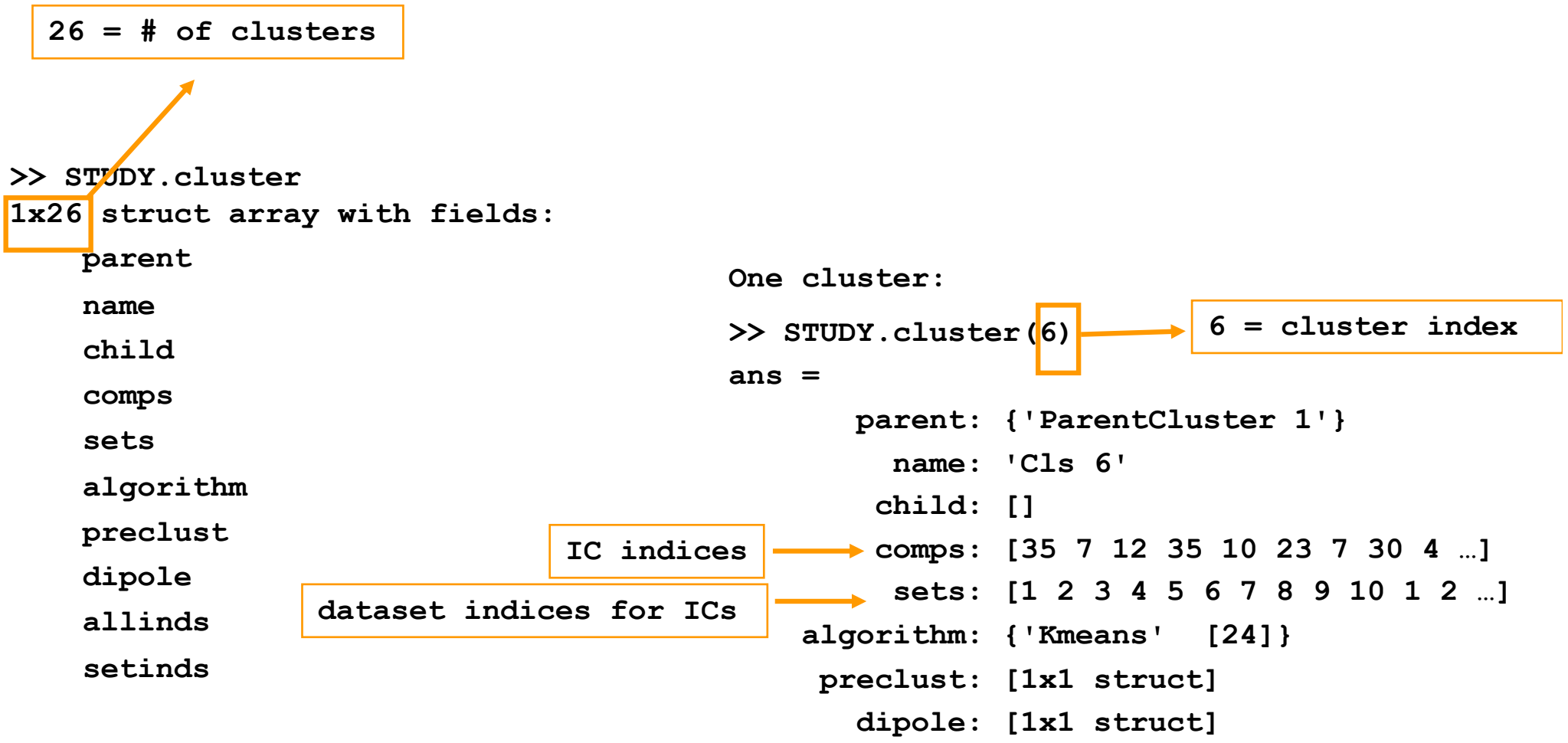
# Cluster components

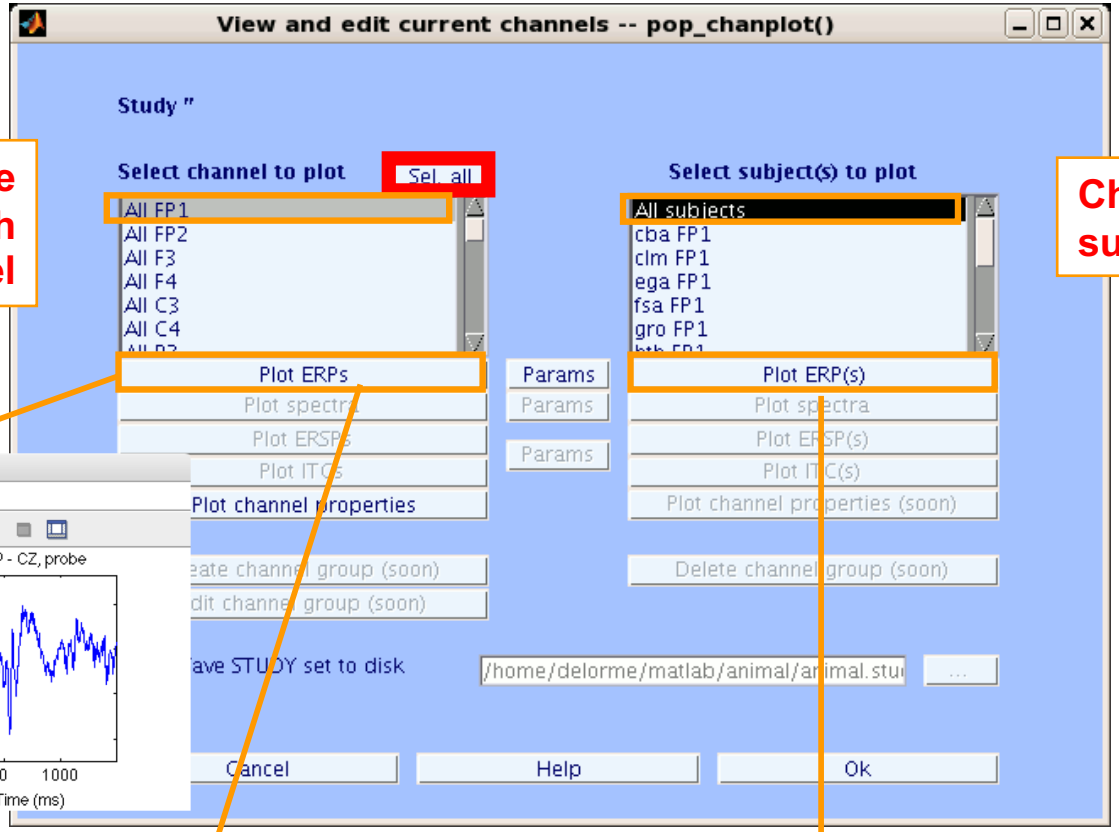



```
[STUDY] = pop_clust(STUDY, ALLEEG,'algorithm','kmeans','clus_num',24);
```

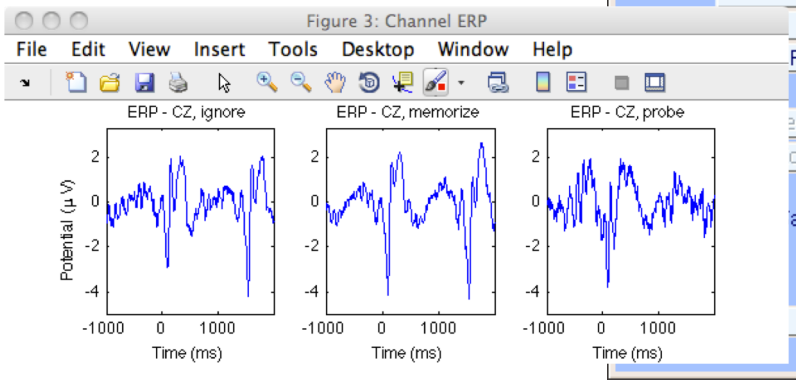
# Understanding STUDY structure



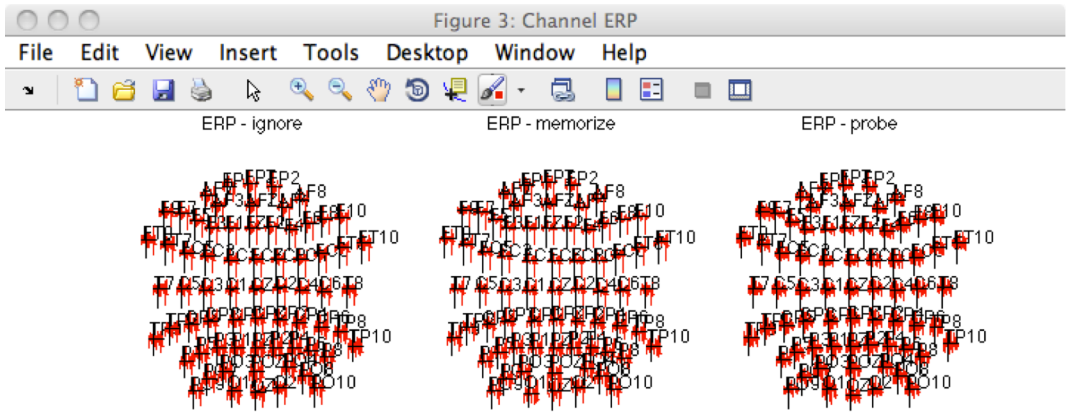


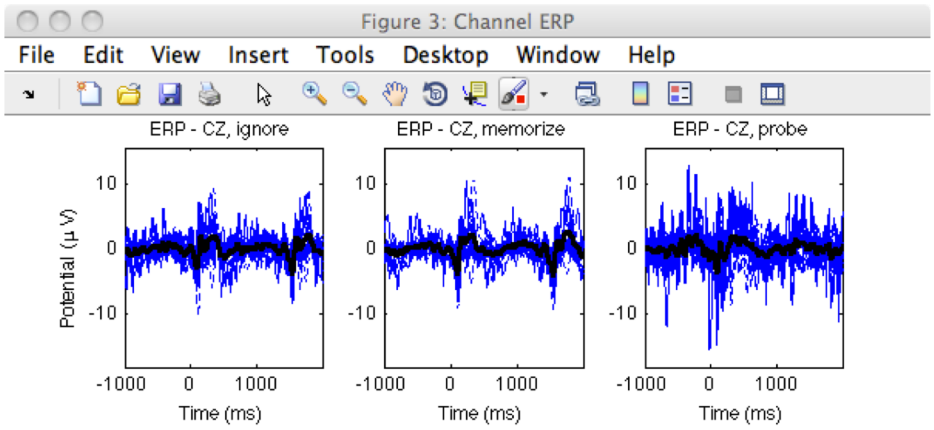

Choose which channel

Choose which subject



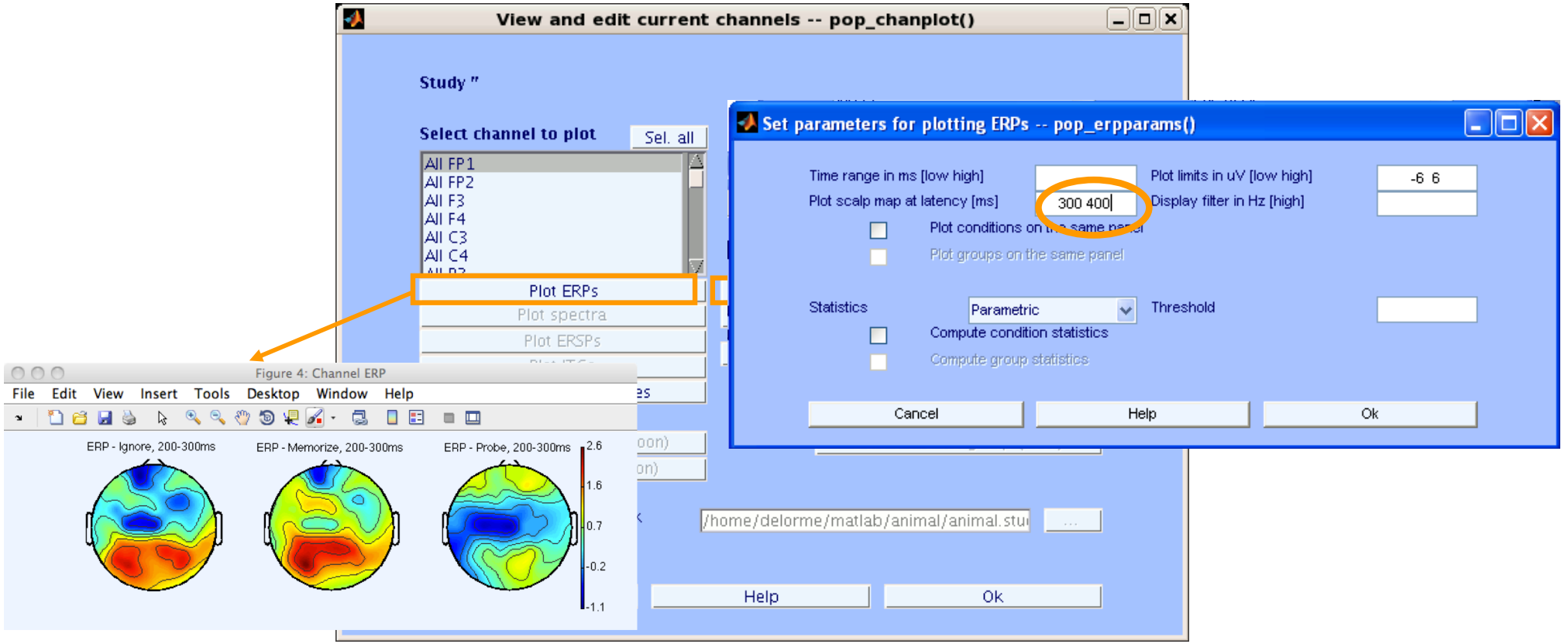

```
STUDY = std_erpplot(STUDY,ALLEEG,'channels',{'FP1'});
```




```
STUDY = std_erpplot(STUDY,ALLEEG,'channels',{'FP1' … });
```

```
STUDY = std_erpplot(STUDY,ALLEEG,'channels',{'FP1'}, 'plotsubjects', 'on' );
```



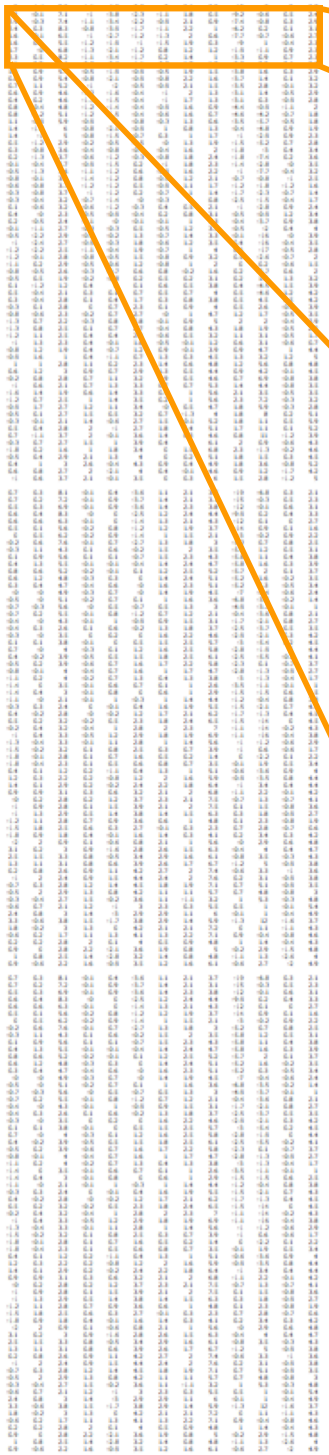

```
STUDY = std_erpplot(STUDY,ALLEEG, 'topotime',[200 300] , 'channels',{'OZ' 'O2' 'FP1' 'FPZ' 'FP2'});
[STUDY erpdata ] = std_erpplot(STUDY,ALLEEG, , 'topotime',[200 300] , 'channels',{'OZ' 'O2'});
```

## Exporting to excell file

```
[1x67x13 double]
[1x67x13 double]
[1x67x13 double]
```

```
xlswrite('myxlsfile',squeeze(erpdata{1}),1);
xlswrite('myxlsfile',squeeze(erpdata{2}),2);
xlswrite('myxlsfile',squeeze(erpdata{3}),3);
```

# Exporting text file



| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| -0.13 | -0.4 | 3.7 | -0.9 | -1.5 | 0.23 | -0.98 | 1.8 | 2.3 | -1.4 | -2.8 | -0.03 | 3.5 |
| -0.54 | -1.3 | 3.6 | -1.1 | -1.2 | 0.62 | -0.91 | 1.6 | 2.2 | -0.98 | -7.7 | -0.42 | 3.2 |
| -0.77 | -0.06 | 3.6 | -1.4 | -1.2 | 0.78 | -0.91 | 1.2 | 2.1 | -0.66 | -0.76 | -1 | 2.5 |
| -0.61 | -0.83 | 3.7 | -1.2 | -1.2 | 0.53 | -0.88 | 1.1 | 1.7 | -1.2 | -1.8 | -1.2 | 1.6 |
| -0.34 | -0.79 | 3.7 | -0.98 | -1.2 | 0.17 | -0.72 | 1 | 1.4 | -1.7 | -2.3 | -0.72 | 1.4 |
| -0.27 | -0.42 | 3.2 | -0.69 | -1.4 | -0.04 | -0.29 | 0.97 | 0.81 | -2.5 | -1.5 | -0.38 | 1.7 |
| 0.097 | -0.58 | 3.2 | -0.61 | -1.2 | -0.32 | 0.36 | 0.47 | 2.1 | -0.96 | -2.8 | 0.89 | 2.4 |
| 0.43 | -0.04 | 2.3 | -0.47 | -0.87 | -0.37 | 0.21 | 0.83 | 3.1 | -0.53 | -0.85 | 1.2 | 3.4 |
| 0.21 | -0.54 | 2.4 | -0.07 | -0.05 | -0.08 | -0.08 | 1 | 3.3 | -0.42 | -3.7 | 0.92 | 3.8 |
| -0.1 | -1.1 | 2.7 | -0.33 | -0.28 | 0.48 | -0.5 | 1.2 | 3.3 | -0.53 | -2 | 0.36 | 4 |
| -0.51 | -2.2 | 2.9 | -0.59 | -0.23 | 1.3 | -0.72 | 1.4 | 3.3 | -0.14 | -16 | -0.05 | 3.9 |

```
dlmwrite('erpfile.txt',squeeze(erpdata{1}),'delimiter', '\t', 'precision', 2);

dlmwrite('erpfile.txt',squeeze(erpdata{2}),'-append', 'roffset', 1,
         'delimiter', '\t', 'precision', 2);

dlmwrite('erpfile.txt',squeeze(erpdata{2}),'-append', 'roffset', 1,
         'delimiter', '\t', 'precision', 2);
```

# STUDY Script

```
% Create Stern STUDY
[ALLEEG EEG CURRENTSET ALLCOM] = eeglab;
pop_editoptions( 'option_storedisk', 1);
subjects = {'S01' 'S02' 'S03' 'S04' 'S05' 'S06' 'S07' 'S08' 'S09' 'S10' 'S11' 'S12'};
filepath = '/Users/arno/temp/STUDY'; % XXXXX Change path here XXXXX
if ~exist(filepath), error('You need to change the path to the STUDY'); end;
commands = {}; % initialize STUDY dataset list

% Loop through all of the subjects in the study to create the dataset
for loopnum = 1:length(subjects) %for each subject
    IgnoreFile   = fullfile(filepath, subjects{loopnum}, 'Ignore.set');
    MemorizeFile = fullfile(filepath, subjects{loopnum}, 'Memorize.set');
    ProbeFile    = fullfile(filepath, subjects{loopnum}, 'Probe.set');
    commands = {commands{:} ...
        {'index' 3*loopnum-2 'load' IgnoreFile   'subject' subjects{loopnum} 'condition' 'Ignore'} ...
        {'index' 3*loopnum-1 'load' MemorizeFile 'subject' subjects{loopnum} 'condition' 'Memorize'} ...
        {'index' 3*loopnum   'load' ProbeFile    'subject' subjects{loopnum} 'condition' 'Probe'}};
end;
% Uncomment the line below to select ICA components with less than 15% residual variance
% commands = {commands{:} {'dipselect', 0.15}};
[STUDY, ALLEEG] = std_editset(STUDY, ALLEEG, 'name','Sternberg','commands',commands,'updatedat','on');

% Update workspace variables and redraw EEGLAB
CURRENTSTUDY = 1; EEG = ALLEEG; CURRENTSET = [1:length(EEG)];
[STUDY, ALLEEG] = std_checkset(STUDY, ALLEEG);
eeglab redraw

[STUDY ALLEEG] = std_precomp(STUDY, ALLEEG, {},'rmicacomps','on','interp','on','recompute','on','erp','on');
STUDY = pop_erpparams(STUDY, 'topotime',[200 300] );
[STUDY erpdata] = std_erpplot(STUDY,ALLEEG,'channels',{'LEYE' 'REYE' 'OZ' 'O2' 'FP1' 'FPZ' 'FP2' 'AF7' ...
            'AF3' 'AFZ' 'AF4' 'AF8' 'F9' 'F7' 'F5' 'F3' 'F1' 'FZ' 'F2' 'F4' 'F6' 'F8' 'F10' 'FT9' ...
            'FT7' 'FC5' 'FC3' 'FC1' 'FCZ' 'FC2' 'FC4' 'FC6' 'FT8' 'FT10' 'T7' 'C5' 'C3' 'C1' 'CZ' ...
            'C2' 'C4' 'C6' 'T8' 'TP9' 'TP7' 'CP5' 'CP3' 'CP1' 'CPZ' 'CP2' 'CP4' 'CP6' 'TP8' 'TP10' ...
            'P7' 'P5' 'P3' 'P1' 'PZ' 'P2' 'P4' 'P6' 'P8' 'PO9' 'PO7' 'PO3' 'POZ' 'PO4' 'PO8' 'PO10' 'O1'});

dlmwrite('erpfile.txt',squeeze(erpdata{1}),'delimiter', '\t', 'precision', 2);
dlmwrite('erpfile.txt',squeeze(erpdata{2}),'-append', 'roffset', 1, 'delimiter', '\t', 'precision', 2);
dlmwrite('erpfile.txt',squeeze(erpdata{2}),'-append', 'roffset', 1, 'delimiter', '\t', 'precision', 2);
```

# Advanced scripting: EEG pipeline

```
sInfo = [];
sInfo(end+1).file = 'rawdata/S01.raw';
sInfo(end).name = 'S01';
sInfo(end).bad_channels = { 'E1' };
```

Raw data file

Subject name

Subject name

# Advanced scripting: EEG pipeline

```
sInfo = [];
sInfo(end+1).file = 'rawdata/S01.raw';
sInfo(end).name = 'S01';
sInfo(end).bad_channels = { 'E1' };
sInfo(end).bad_data  = [726 1495;6098 6831;13245 14057;15715 16399;22756 24457
```

Raw data file

Subject name

Subject name

Copy the output from the eeg_eegrej function in the history

# Advanced scripting: EEG pipeline

```
sInfo = [];
sInfo(end+1).file = 'rawdata/S01.raw';
sInfo(end).name = 'S01';
sInfo(end).bad_channels = { 'E1' };
sInfo(end).bad_data  = [726 1495;6098 6831;13245 14057;15715 16399;22756];
sInfo(end).bad_comps = [1.6681  1.9870 0.3979 0.4444 -0.2274 -0.1433 -0.2626 -
                        1.1917 -1.4838 0.7469 -1.1599 0.4773 -0.3257  0.3074 -
```

Raw data file

Subject name

Subject name

Copy transposed columns of the inverse weight matrix
EEG.icawinv for your selected artifact components

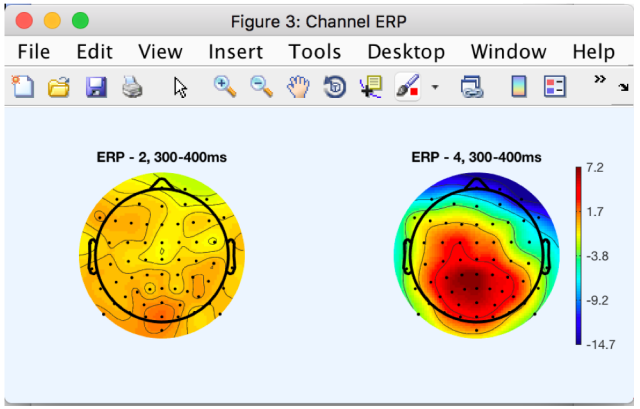
# Advanced scripting: EEG pipeline

datainfo.m file

```
sInfo = [];
sInfo(end+1).file = 'S01.raw';
sInfo(end).name = 'S01';
sInfo(end).bad_channels = { 'E1' };
sInfo(end).bad_data  = [726 1495;6098 6831;13245 14057;15715 16399;22756 24457;3074
sInfo(end).bad_comps = [1.6681  1.9870 0.3979 0.4444 -0.2274 -0.1433 -0.2626 -0.108
                        1.1917 -1.4838 0.7469 -1.1599 0.4773 -0.3257  0.3074 -0.163

sInfo(end+1).file = 'S02.raw';
sInfo(end).name = 'S02';
sInfo(end).bad_channels = { };
sInfo(end).bad_data  = [41661 43713;24000 24833;44878 46501;48706 49210;51190 52353
sInfo(end).bad_comps = [0.6960 -0.8637  0.9087 -0.8028  0.4873 -0.2142  0.2737 -0.2
                       -0.0875 -0.4056 -0.0287 -0.3870  0.0600 -0.3716  0.3425 -0.4
                        2.1928  1.5712  0.8622  0.3215 -0.0357 -0.3125 -0.2268 -0.3

sInfo(end+1).file = 'S03.raw';
sInfo(end).name = 'S03';
sInfo(end).bad_channels = { 'E10' 'E19' 'E20' 'E29' };
sInfo(end).bad_data  = [1 10449;19808 21815;25678 27254;29257 30010;34023 36016;367
sInfo(end).bad_comps = [ 1.8583  2.0468 -0.0516  0.3159 -0.4256 -0.2770 -0.3643 -0.
                         1.2189 -0.7385  1.2464 -0.8913  0.5475 -0.3971  0.2987 -0.
                        -0.1248 -0.1358 -0.1954 -0.2533 -0.1555 -0.2313 -0.0351 -0.
```

```
datainfo;
pop_editoptions( 'option_storedisk', 1);
outputEEGFolder = 'preprocessed_data';
if ~exist(outputEEGFolder), mkdir(outputEEGFolder); end;

for iSubj = 1:length(sInfo)

    % load dataset
    EEG = pop_biosig(sInfo(iSubj).file);
    EEG.setname = sInfo(iSubj).name;

    % proprocess data
    chanFile= 'plugins/dipfit2.3/standard_BEM/elec/standard_1005.elc';
    EEG = pop_chanedit(EEG, 'lookup', fullfile(fileparts(which('eeglab.m')), chanFile));
    EEG = pop_iirfilt( EEG, 0.5, 0, [], 0, 0); % high pass filtering
    EEG = pop_iirfilt( EEG, 0, 55, [], 0, 0); % low pass filtering
    EEG = pop_select(EEG, 'nochannel', sInfo(iSubj).bad_channels); % remove bad channels
    EEG = pop_reref( EEG, []); % average reference (optional)
    EEG = eeg_eegrej( EEG, sInfo(iSubj).bad_data); % remove bad portions of data

    % run ICA
    EEG = pop_runica(EEG, 'icatype', 'sobi');

    % tag bad components
    EEG = pop_findmatchingrejcomps(EEG,'matchcomps',sInfo(iSubj).bad_comps,'corrthresh',0.92);

    % extract data epochs
    EEG = pop_epoch(EEG, { 2 4 } , [-1 2]);

    % save dataset
    EEG.saved = 'no';
    EEG = pop_saveset( EEG, 'filepath', outputEEGFolder, 'filename', [ sInfo(iSubj).name '.set' ]
end
```

# Create STUDY




```
datainfo;
pop_editoptions( 'option_storedisk', 1);
outputEEGFolder = 'preprocessed_data';
studyCommand    = {};

% generate STUDY commands
for iSubject = 1:length(sInfo)
    fileName  = fullfile(outputEEGFolder, [ sInfo(iSubject).name '.set' ]);
    studyCommand = [ studyCommand { 'index' iSubject 'load' fileName 'subject' ...
           sInfo(iSubject).name } ];
end;

% create data
[STUDY ALLEEG] = std_editset( [], [], 'name', 'test', 'commands', studyCommand, ...
        'updatedat','off', 'filename', 'test.study', 'resave', 'on');
STUDY = std_makedesign(STUDY, ALLEEG, 1, 'name','STUDY.design 1','delfiles','off', ...
'defaultdesign','off','variable1','type','values1',{'2' '4' });


% update workspace variables and redraw EEGLAB
CURRENTSTUDY = 1; EEG = ALLEEG; CURRENTSET = [1:length(EEG)];
[STUDY, ALLEEG] = std_checkset(STUDY, ALLEEG);
eeglab redraw

% precompute and plot data
allchanlocs = eeg_mergelocs(ALLEEG.chanlocs);
[STUDY ALLEEG] = std_precomp(STUDY, ALLEEG, {},'interp','on','recompute','on','erp', 'on');
STUDY = pop_statparams(STUDY, 'condstats','on','singletrials','on','mode','fieldtrip', ...
'fieldtripmethod','montecarlo','fieldtripmcorrect','cluster');
[STUDY erp] = std_erpplot(STUDY,ALLEEG, 'channels',{allchanlocs.labels}, 'topotime',[300 400]);
print results.eps -depsc
```

# Exporting figures

## Transparency and complex figures

To export figures for publication, use .eps format (postscript) and edit for instance with adobe illustrator. Use "*set(gcf, 'renderer', 'painter')*" before exporting complex figures. Note that these cannot handle transparency and 3-D graphics.

Transparency: Use the "plot2svg" matlab toolbox to export figure for transparency.

# Exercise: build your own pipeline

**Suggestion for exercise**

1. Load oddball_file.set dataset (in Data folder or on the wiki)
2. High pass filter at 0.5Hz (menu Tools > Filter)
3. Re-reference to average ref. (optional) (menu Tools > Re-reference)
4. Reject bad channels using clean_rawdata
5. Re-reference to average ref. again (optional)
6. Run ICA
7. Run IClabel plugin
8. Tag artifactual components

   ```
   [~, ind] = max(EEG.etc.ic_classification.ICLabel.classifications, [], 2);
   EEG.reject.gcompreject([find(ind == 2);find(ind == 3)]) = 1;
   ```
9. Epoch data on Oddball (type 4) and Standard (type 2) – save dataset
10. Create a STUDY with this single file
11. Compare the ERP for Oddball (type 4) and Standard (type 2) and use single-trial statistics with cluster correction for multiple comparisons
12. Build a script that creates the STUDY and perform the same analysis
13. Save the figure at the end of the script in eps or jpg format (“print –depsc file” command or “print –djpg file” command).
14. Run the full pipeline (dataset processing and STUDY processing)
15. Change the filtering in the pipeline (step 2) and observe effects

# Evaluation of Artifact Subspace Reconstruction for Automatic EEG Artifact Removal

Chi-Yuan Chang, *Student Member, IEEE,* Sheng-Hsiou Hsu, *Student Member, IEEE,* Luca Pion-Tonachini, *Student Member, IEEE,* and Tzyy-Ping Jung, *Fellow, IEEE*

***Abstract*— One of the greatest challenges that hinder the decoding and application of electroencephalography (EEG) is that EEG recordings almost always contain artifacts – non-brain signals. Among existing automatic artifact-removal methods, artifact subspace reconstruction (ASR) is an online and real-time capable, component-based method that can effectively remove transient or large-amplitude artifacts. However, the effectiveness of ASR and the optimal choice of its parameter have not been evaluated and reported, especially on real EEG data. This study systematically validates ASR on ten EEG recordings in a simulated driving experiment. Independent component analysis (ICA) is applied to separate artifacts from brain signals to allow a quantitative assessment of ASR's effectiveness in removing various types of artifacts and preserving brain activities. Empirical results show that the optimal ASR parameter is between 10 and 100, which is small enough to remove activities from artifacts and eye-related components and large enough to retain signals from brain-related components. With the appropriate choice of the parameter, ASR can be a powerful and automatic artifact removal approach for offline data analysis or online real-time EEG applications such as clinical monitoring and brain-computer interfaces.**

identify and reject the artifact-related independent components (ICs) [7]. However, the ICA-based methods were less effective in removing transient, non-biological artifacts such as abrupt impedance changes due to headset motions and were computationally expensive and generally for offline analyses.

To address the challenges, Kothe and Jung [8] proposed the artifact subspace reconstruction (ASR) approach, which is an automatic, online-capable, component-based artifact removal method that could be useful in removing transient or large-amplitude artifacts. ASR is similar to principal component analysis (PCA)-based method in which large-variance components are rejected and channel data are reconstructed from remaining components. The main difference is that ASR automatically identifies and utilizes clean portions of data to determine thresholds for rejecting components. Although recent studies [1] [9] have indicated the potential use of ASR as a powerful data-cleaning method, the effectiveness of ASR and the guidelines for choosing its parameter have

# Automated pipeline

```
% Required plugins
% - BIOSIG plugin
% - IClabel plugin
% - clean_rawdata plugin
% - firfilt plugin

clear
filepath = '/Users/arno/Desktop/EEGLAB-workshop/EEG_data/';

% preprocessing
[ALLEEG EEG CURRENTSET ALLCOM] = eeglab;
EEG = pop_biosig(fullfile(filepath,'oddball_file.bdf'));
p = fileparts(which('eeglab'));
EEG = pop_chanedit(EEG, 'lookup',fullfile(p, 'plugins/dipfit2.4/standard_BESA/standard-10-5-cap385.elp'));

% filter
EEG = pop_eegfiltnew(EEG, [],0.5,1690,1,[],0);
EEG = pop_reref( EEG, []);

% clean using ASR
EEG = clean_rawdata(EEG, 5, [0.25 0.75], 0.8, 4, 5, 0.5);
EEG = pop_reref( EEG, []);

% ICA and labeling
EEG = pop_runica(EEG, 'icatype', 'runica');
EEG = eeg_checkset(EEG);
EEG = pop_iclabel(EEG);
[~, ind] = max(EEG.etc.ic_classification.ICLabel.classifications, [], 2);
EEG.reject.gcompreject([find(ind == 2);find(ind == 3)]) = 1;

% extract epoch and save
EEG = pop_epoch( EEG, {  '1'  '2'  }, [-1  2], 'newname', 'Simple Oddball epochs', 'epochinfo', 'yes');
EEG = pop_rmbase( EEG, [-1000 0] ,[],[]);
EEG = pop_saveset( EEG, 'filename','oddball_epochs.set','filepath',filepath);

% create study and plot
[STUDY ALLEEG] = std_editset( STUDY, [], 'commands',{{'index' 1 'load' fullfile(filepath,'oddball_epochs.set') ...
                            'subject' 'S01'}},'updatedat','on','rmclust','on' );
[STUDY ALLEEG] = std_precomp(STUDY, ALLEEG, {},'savetrials','on','rmicacomps','on','interp','on','recompute','on','erp','on');
STUDY = pop_erpparams(STUDY, 'topotime',100);
STUDY = std_erpplot(STUDY,ALLEEG,'channels',{'Fp1' 'AF7' 'AF3' 'F1' 'F3' 'F5' 'F7' 'FT7' 'FC5' 'FC3' 'FC1' 'C1' 'C3' 'C5' ...
     'T7' 'TP7' 'CP5' 'CP3' 'CP1' 'P1' 'P3' 'P5' 'P7' 'PO7' 'PO3' 'O1' 'Iz' 'Oz' 'POz' 'Pz' 'CPz' 'Fpz' 'Fp2' 'AF8' 'AF4' ...
     'AFz' 'Fz' 'F2' 'F4' 'F6' 'F8' 'FT8' 'FC6' 'FC4' 'FC2' 'FCz' 'Cz' 'C4' 'C6' 'TP8' 'CP6' 'CP4' 'CP2' 'P2' 'P4' 'P6' 'P8' ...
     'P10' 'PO8' 'PO4' 'O2'}, 'design', 1);
```