1. Component ERP image

2. Cluster ERP image

3. Accessing STUDY raw data

4. Dipole density
Component ERP image
Component ERP Images

- Select fields:
  - Latency
  - Type
  - Epoch

- Figure title
  - Plot scalp map
  - Plot ERP
  - Plot colorbar

- Epoch-sorting field:
  - Latency

- Event type(s):
  - Bp1
  - Bp4

- Inter-trial coherence:
  - Frequency (Hz): min 0, max 0.20
  - Other options:
    - Plot spectrum (min Hz): L
    - Other options:
      - Rescale: no
      - Align: no

- Figure 5: erpimage()
Component ERP Images

Phase-sorted image
## Component ERP Images

**Figure title**

- Plot scalp map
- Plot ERP
- Plot colorbar

### Phase-sorted alpha power

**Figure 3: erpimage()**

- Epoch-sorting field
- Event type(s)
- Event time

### Sort trials by phase

- **Frequency (Hz | minHz maxHz)**
  - 10 12

### Inter-trial coherence options

- **Frequency (Hz | minHz maxHz)**
  - 10 12
- **Signif. level (<0.01)**
  - .01

### Other options

- **Plot spectrum (minHz maxHz)**
  - Baseline ampl. (dB)

---

EEGLAB Workshop, June 16-18, 2012, Beijing, China: Julie Onton – Trial-by-trial visualization and raw data
Component ERP

Component ERP image -- pop_erpimage

Component(s) 3
Project to channel #
Smoothing 10
Downsampling 1
Time limits (ms) -800 1000

Figure title
Plot scalp map
Plot ERP
Plot colorbar

Phase-sorted Trials

Same data: Sorted by alpha amplitude

EEGLAB Workshop, June 16-18, 2012, Beijing, China: Julie Onton

Trial by trial visualization and raw data

>> help erpimage

'ampsort' = [center_ms, prcnt, freq, maxfreq] Sort epochs by amplitude.
Component ERP Images

Same sorting order: Amplitude vs. activations
Sorting options in ERP image: RT

EEGLAB Workshop, June 16-18, 2012, Beijing, China: Julie Onton – Trial-by-trial visualization and raw data
Sorting options in ERP image: RT

Figure 2: erpimage()
Sorting options in ERP image: type

ie, “was the probe letter in memorized set or not?”
Sorting options in ERP image: type (img amps)
Trial-by-trial visualization and raw data

1. Component ERP image
2. Cluster ERP image
3. Accessing STUDY raw data
4. Dipole density
Cluster ERP image: match polarity

reversed polarities reflect mismatched scalp maps

reorienting maps and activations gives a more coherent picture
Movie of IC scalp map over time
EEGLAB STUDY
matches polarities for you

EEGLAB Workshop, June 16-18, 2012, Beijing, China: Julie Onton – Trial-by-trial visualization and raw data
Matching activation polarity

However, original IC maps/activations may be opposite within a cluster:

Reversed polarity

EEGLAB Workshop, June 16-18, 2012, Beijing, China: Julie Onton – Trial-by-trial visualization and raw data
Matching activation polarity

Reorient map AND activation of one IC to align
Cluster ERP image: RT sort

Consistent scalp maps

Sort cluster ERP image by response time

EEGLAB Workshop, June 16-18, 2012, Beijing, China: Julie Onton – Trial-by-trial visualization and raw data
STUDY ERP image (from the GUI)
% plot all mean maps to get topo polarity:
STUDY = std_topoplot(STUDY,ALLEEG,'clusters',[2:length(STUDY.cluster)]);
design = 1;
clust = 7;  % choose a cluster
cond = 3;  % choose a condition (from STUDY.condition),
            % Probe is the only condition with a button press (RT)

% requires memory options set to 'pre-calculate ica activations'
% collect activations (correctly oriented) for all cluster ICs:
CURRENTSTUDY = 1; EEG = ALLEEG; CURRENTSET = [1:length(EEG)];
rts = []; allacts = zeros(1,length(ALLEEG(1).times),0);
for ic = 1:length(STUDY.cluster(clust).allinds{cond})
    design_idx = STUDY.cluster(clust).setinds{cond}(1,ic);
    setidx = STUDY.design/design.cell(design_idx).dataset;
    comp = STUDY.cluster(clust).allinds{cond}(1,ic);
    [ALLEEG EEG CURRENTSET] = pop_newset(ALLEEG, EEG, CURRENTSET,...
        'retrieve',setidx,'study',CURRENTSTUDY);
    for ep = 1:length(EEG.epoch)
        zlat = find(cell2mat(EEG.epoch(ep).eventlatency)==0);
        sortvar = [sortvar,EEG.epoch(ep).eventrt{zlat}];
    end;
    rmsuv = sqrt(mean(ALLEEG(setidx).icawinv(:,comp).^2));% RMS at scalp
    acts = ALLEEG(setidx).icaact(comp,:, :) * rmsuv*STUDY.cluster(clust).topopol(ic);
    allacts(:, :, end+1:end+size(acts,3)) = acts;
end;
allacts = squeeze(allacts); % makes a (frames x trials) matrix

Only necessary for sorting by or plotting RT
% PLOT activations:-------------------
clim = 2.5;  % color limits
smoothby = 10;  % Vertical smoothing across trials

figure; [outdata, outvar, outtrials, limits, axhndls, erp, amps, coher, cohs ig, ampsig, outamps, phsangls, phsamp, sortidx, erpsig] = ...

erpimage( allacts, sortvar, linspace(EEG.xmin*1000,...
EEG.xmax*1000, EEG.pnts), ['Cluster ',int2str(clust)],...
smoothby, 1 ,'yerplabel','','erp','limits','cbar','...
'caxis',[−clim clim],'coher', [9 12 .01]);
1. **Component ERP image**

2. **Cluster ERP image**

3. **Accessing STUDY raw data**

4. **Dipole density**
** Where is the raw data stored?**
Data for each subject is stored in the file path of that subject: STUDY.datasetinfo(subj).filepath

** What is it called?**
File name format: 'design?_Subj_var1_var2.extension'
extension = '.ica*' or '.dat*' (for channel data)
for example:
design2_S04_memorize_3.icaerp % ERP data
design2_S04_memorize_3.icaersp % ERSP data
design2_S04_memorize_3.icaitec % ITC data
design2_S04_memorize_3.icaspec % Power spectrum data
design2_S04_memorize_3.icatopo % Scalp map data

% Example of channel data file name:
design2_S04_memorize_3.daterp % ERP data
Load individual ERSPs

% load ERSP data for all ICs in a whole cluster at once:
clust = 5; % choose a cluster
cond = 1; % choose experimental condition

[STUDY logersp timevals logfreqs pgroup pcond pinter] = ...

std_erspplot(STUDY, ALLEEGB, 'clusters', clust, 'plotsubjects', 'off');

% Check imported variables in workspace:

>> whos logersp
Name           Size              Bytes  Class Attributes
logersp        3x1    3 conditions 1983216  cell

>> logersp
logersp =

[72x153x15 single]
[72x153x15 single] frequency x times x ICs (different subjects)
[72x153x15 single]
Plot individual ERSPs

% load ERSP data for all ICs in a single cluster:
clust = 5; % choose a cluster
cond = 1; % choose experimental condition

(>> help std_erspplot for function call)
[STUDY logersp timevals logfreqs pgroup pcond pinter] = ...
std_erspplot(STUDY,ALLEEG,'clusters',clust,'plotsubjects','on');

% or plot them yourself from output:
figure; clim = 3; % standardize color limits
for ic = 1:size(logersp{cond},3) % all ICs in the cluster
    sbplot(row,col,ic);
    imagesclogy(timevals, logfreqs, logersp{cond}(:, :, ic));
    set(gca,'clim', [-clim clim]); % adjust the color limits
    set(gca,'ydir','norm'); % plot low freqs at the bottom
    title(['IC ',int2str(STUDY.cluster(clust).comps(ic))]);
end;

textsc(['Cluster ','int2str(clust)'],'title');
cbar; % include a colorbar
Plot individual ERSPs

ERSPs for one condition from all ICs in one cluster
Trial-by-trial visualization and raw data

1. **Component ERP image**

2. **Cluster ERP image**

3. **Accessing STUDY raw data**

4. **Dipole density**
Dipole density plotting

**PURPOSE:** to visualize distributions of dipoles in ‘MRI-esque’ way
'method' - ['alldistance'|'distance'|'entropy'|'releventropy'
'alldistance' - {default} takes into account the gaussian-weighted distances from each voxel to all the dipoles. See 'methodparam' (below) to specify a standard deviation (in mm) for the gaussian weight kernel.
'distance' - takes into account only the distances to the nearest dipole for each subject. See 'methodparam' (below).
cond = 1;  clust = 3;
dipsources = struct('posxyz',[],'momxyz',[],'rv',[]);  n = 1;
nowidx = 0;  % initialize
for ic = 1:length(STUDY.cluster(clust).comps)
    setidx = STUDY.cluster(clust).sets(cond,ic);
    comp = STUDY.cluster(clust).comps(ic);
    if setidx ~= nowidx  % don't call in if already active
        [ALLEEG EEG CURRENTSET] = pop_newset(ALLEEG, EEG, CURRENTSET, ...
            'retrieve',setidx, 'study',CURRENTSTUDY);  nowidx = setidx;
    end;
    model = EEG.dipfit.coordformat;
    dipsources(1,n).posxyz = EEG.dipfit.model(comp).posxyz;
    dipsources(1,n).momxyz = EEG.dipfit.model(comp).momxyz;
    dipsources(1,n).rv = EEG.dipfit.model(comp).rv;  n = n + 1;
end;
dipoledensity(dipsources , 'method','alldistance','methodparam',10,...
    'coordformat',model);
Exercise

• **Novice**
  - Explore the STUDY structure: Identify subject and dataset number(s) for a single IC from one cluster
  - Load ERSP or spectral data for a cluster and plot (std_erspplot)
    > script a loop plotting all (or some) clusters

• **Intermediate / Advanced**
  - Script a loop to build a STUDY from the command line
  - Load raw ERSP data all ICs/subjs in a cluster and extract raw and/or baseline-corrected power in a time-frequency window
    > (use >> load('…icaersp',-mat))
  - Plot an ERP image for a cluster. See what happens when you make STUDY.cluster(clust).topopol all positive (how many were -1 before?)
Supplementary lessons

Trial-by-trial visualization and raw data
Component ERP Images

EEGLAB Workshop, June 16-18, 2012, Beijing, China: Julie Onton – Trial-by-trial visualization and raw data
Frontal midline EEG dynamics during working memory.

Raw data files

% Load *raw* ERSP data

load_string = 'C:\workshop\STUDY\S01\Memorize.icaersp';

ERSPdata = load('-mat',load_string); % .mat format!
Raw data structure

>> ERSPdata

ERSP dB data

compl1_ersp: [100 x 200 single]
compl1_erspbase: [1 x 100 single]
compl1_erspboot: [100 x 2 single]
compl2_ersp: [100 x 200 single]
compl2_erspbase: [1 x 100 single]
compl2_erspboot: [100 x 2 single]

100 frequency bins

freqs: [1 x 100 double]
times: [1 x 200 double]
datatype: 'ERSP'
parameters: {1 x 26 cell}
datafile: [1 x 57 char]

200 time points

upper and lower bootstrap limits

100 frequency bins

bootstrap limits

200 time points

bootstrap limits

>> comp = 1;
>> oneic = ['ERSPdata.comp',int2str(comp),'_ersp'];
>> oneic = eval(oneic);
>> tms = find(RAWdata.times > 500 & RAWdata.times < 1000);
>> frs = find(RAWdata.freqs > 4 & RAWdata.freqs < 8);
>> dat = mean(mean(oneic(frs,tms))) % mean raw power in window
(compare 'dat' across ICs/clusters/groups, etc)
Load other cluster raw data measures

```
subjs = {'S01','S02','S03','S04','S05','S06','S07','S08','S09','S10','S11','S12','S13'};
conds = {'memorize','ignore','probe'};
des = 2;
subj = 1;

% raw ERSPs
load_string = [basedir,subjs{subj},'\Memorize.icaersp'];
load_string = [basedir,subjs{subj},'\design',int2str(des),'_',subjs{subj},conds{cond},'\icaersp'];

% OR, raw ITCs
load_string = [basedir,subjs{subj},'\Memorize.icaictc'];
load_string = [basedir,subjs{subj},'\design',int2str(des),'_',subjs{subj},conds{cond},'\icaictc'];

ERSPdata = load('-mat',load_string); % Run actual 'load' command
```
Load raw ERSP and subtract baseline

% Mask an ERSP using calculated bootstrap limits
ic = 7; % Choose and IC to plot (must be pre-calculated for this subject)
oneic = ['ERSPdata.comp',int2str(ic),'_ersp'];
oneic = eval(oneic);
onebase = ['ERSPdata.comp',int2str(ic),'_erspbase']; % to see the baseline spectrum
onebase = eval(onebase);
oneboot = ['RAWdata.comp',int2str(ic),'_erspboot'];
oneboot = eval(oneboot); % Bootstrap significance limits

maskERSP = oneic;
maskERSP(find(oneic > repmat(oneboot(:,1),[1 size(oneic,2)])) & ...
    oneic < repmat(oneboot(:,2),[1 size(oneic,2)]))) = 0;
clim = 6; % set +/- color limits
figure; imagesc(ERSPdata.times,ERSPdata.freqs,maskERSP,[-clim clim]);
set(gca,'ydir','norm');
title(['Subj ',int2str(subj),' IC ',int2str(ic)]); 
cbar;
Mask raw ERSP with bootstrap values

Solid green indicates non-sig values set to zero
% Now loop through subjects and add to the STUDY:
index = 1; % initialize STUDY index
for subj = 1:length(subjs) % for each subject
    for cond = 1:length(setnames) % for each condition
        datset = [basedir,subjs{subj},'\',setnames{cond}]; % concatenate strings
        [STUDY ALLEEG] = std_editset( STUDY, ALLEEG,'name',studiname,'task', taskname,...
            'commands',{['index',index,'load',datset], {'index',index,'subject',subjs{subj}}},...
            {'index',index,'session', 1}, {'index',index, 'group', 1},...
            {'index',index,'condition', conds{cond}}, {'index',index,'dipselect','.12}},{
            'inbrain','on', 'updatedat','off',...'
savedat', 'off', 'filename', [basedir,savename]});
        index = index + 1; % update set file index
        CURRENTSTUDY = 1; EEG = ALLEEG; CURRENTSET = [1:length(EEG)]; % reassign EEGLAB variables
        [STUDY, ALLEEG] = std_checkset(STUDY, ALLEEG); % check STUDY for consistency
    end;
end;
eeglab redraw % need to refresh GUI to see the STUDY you built
STUDY.cluster(clust).setinds and STUDY.cluster(clust).allinds are cell arrays that are linked with the current STUDY.design. They are adjusted each time a new STUDY.design is selected.

STUDY.cluster(clust).setinds are indexes into the STUDY.design structure and STUDY.cluster(clust).allinds are the corresponding component indices.

In contrast, STUDY.cluster(clust).sets and STUDY.cluster(clust).comps fields correspond to each other but do NOT change when a new STUDY.design is selected. STUDY.cluster(clust).sets is a [cond x ncomps] matrix using all original STUDY conditions and gives the index of the corresponding dataset saved in STUDY.datasetinfo. Each column corresponds to the components listed in STUDY.cluster(clust).comps.

You will find that STUDY.cluster(clust).allinds{1}, for example, will have the same values as in STUDY.cluster(clust).comps (which are component indices included in the cluster). However, the order of the components may be different. That is because STUDY.cluster(clust).allinds refers to a different structure, the ‘STUDY.design’, where the selected components might be different because of different subjects included in the current STUDY design. The actual STUDY.datasetinfo indices can be retrieved through STUDY.setinds and STUDY.design.
Both STUDY.cluster(clust).allinds and STUDY.cluster(clust).setinds are cell arrays with the number of rows equal to the number of primary independent variables (i.e., conditions) in your selected STUDY design, and the number of columns equal to the number of secondary independent variables.

The matrix within one of these cells contains the number of columns equal to the number of components in the cluster.

**STUDY.cluster(clust).setinds are indices into the structure STUDY.design.cell**

Take an example from 'design 1' and 'cluster 6':

```matlab
>> design_num = 1;

>> clust = 6;
```
Now, if you have a cluster such as:

```matlab
>> STUDY.cluster(clust)
ans =
    name: 'Cls 6'
    sets: [3x17 double]
    comps: [14 10 6 13 6 7 8 12 17 13 11 14 9 5 10 12 5]
    parent: {'Parentcluster 1'}
    child: []
    preclust: [1x1 struct]
    allinds: {3x1 cell}
    setinds: {3x1 cell}
    algorithm: {'Kmeans' [25]}
    topo: [67x67 double]
    topox: [67x1 double]
    topoy: [67x1 double]
    topoall: {1x17 cell}
    topopol: [1 1 1 1 1 1 1 1 1 -1 1 1 1 1 1 1]```

Here we have a STUDY design with 3 conditions since
STUDY.cluster(clust).setinds and STUDY.cluster(clust).allinds have 3 rows.

Assume you are interested in finding the STUDY.datasetinfo index for the first component in the STUDY.cluster(clust).allinds{1} field:
STUDY.allinds/STUDY.setinds explanation

>> STUDY.cluster(clust).allinds{1}

ans =

[14 10 6 13 6 7 8 12 17 13 11 14 ... ]

In this case, we are interested in component 14, but we do not yet know which subject or dataset this is referencing.

(Note that in this STUDY example, STUDY.cluster(clust).allinds{1} and STUDY.cluster(clust).allinds{2} will be identical because both conditions contain the same ICs).
To find the dataset index, we have to look in `STUDY.cluster(clust).setinds{1}`:

```matlab
>> STUDY.cluster(clust).setinds{1}
ans =
[1  2  3  4  5  6  7  7  8  9  10  10  11...]
```

Again, remember that these indices are NOT referring to `STUDY.datasetinfo`, but they will get you there eventually...

Take the first index, in this case 1, and plug that into `STUDY.design(design_num).cell`:

```matlab
>> STUDY.design(design_num).cell(1)
ans =

    dataset: 2
    trials: {[1x267 double]}
    value: {'ignore'  ''}
    case: 'S01'
    filebase: [1x60 char]
```
NOW, we can retrieve the actual STUDY.datasetinfo index, which in this case is dataset number 2.

```matlab
>> STUDY.datasetinfo(2)
ans =

    filepath: [1x53 char]
    filename: 'Ignore.set'
    subject: 'S01'
    session: 1
    condition: 'ignore'
      group: 1
      index: 2
    comps: [1x23 double]
  trialinfo: [1x267 struct]
```