```
% pop_timef() - Returns estimates and plots of event-related (log) spectral
%            perturbation (ERSP) and inter-trial coherence (ITC) changes
%            timelocked to a set of input events in one data channel.
%
% Usage:
%   >> pop_timef(EEG, typeplot); % pop_up window
%   >> pop_timef(EEG, typeplot, lastcom); % pop_up window
%   >> pop_timef(EEG, typeplot, channel); % do not pop-up
%   >> pop_timef(EEG, typeproc, num, tlimits,cycles,
%                        'key1',value1,'key2',value2, ... );
%
% Inputs:
%   INEEG    - input EEG dataset
%   typeproc - type of processing. 1 process the raw
%              data and 0 the ICA components
%   num      - component or channel number
%   tlimits  - [mintime maxtime] (ms) sub-epoch time limits
%   cycles   -  >0 -> Number of cycles in each analysis wavelet
%                0 -> Use FFTs (with constant window length)
%
% Optional inputs:
%    See the timef() function.
%
% Outputs: same as timef(), no outputs are returned when a
%          window pops-up to ask for additional arguments
%
% Author: Arnaud Delorme, CNL / Salk Institute, 2001
%
% See also: timef(), eeglab()

%123456789012345678901234567890123456789012345678901234567890123456789012

% Copyright (C) 2002 arno@salk.edu, Arnaud Delorme, CNL / Salk Institute
%
% This program is free software; you can redistribute it and/or modify
% it under the terms of the GNU General Public License as published by
% the Free Software Foundation; either version 2 of the License, or
% (at your option) any later version.
%
% This program is distributed in the hope that it will be useful,
% but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
% GNU General Public License for more details.
%
% You should have received a copy of the GNU General Public License
% along with this program; if not, write to the Free Software
% Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA

% $Log: pop_timef.m,v $
% Revision 1.31  2004/01/22 21:54:28  scott
% rm same
%
% Revision 1.30  2004/01/22 21:49:32  scott
% same
%
% Revision 1.29  2004/01/22 21:48:15  scott
% add plotphase button
```

```
%
% Revision 1.28  2003/08/08 22:16:35  arno
% comment is_sccn
%
% Revision 1.27  2003/08/05 23:22:53  arno
% debug time limit
%
% Revision 1.26  2003/04/22 21:33:51  arno
% run newtimef if at SCCN
%
% Revision 1.25  2003/02/13 00:03:52  arno
% debugging last
%
% Revision 1.24  2003/02/13 00:01:04  arno
% background color
%
% Revision 1.23  2003/02/01 00:22:09  arno
% adding title with electrode name
%
% Revision 1.22  2002/08/12 01:48:37  arno
% same
%
% Revision 1.21  2002/08/12 01:45:00  arno
% color
%
% Revision 1.20  2002/08/11 22:21:05  arno
% color
%
% Revision 1.19  2002/08/09 22:31:39  arno
% updating
% text
%
% Revision 1.18  2002/08/09 22:31:04  arno
% updating history
%
% Revision 1.17  2002/07/30 18:23:55  arno
% changind default
%
% Revision 1.16  2002/07/30 00:21:30  arno
% implementing history
%
% Revision 1.15  2002/07/29 23:20:56  arno
% updating text
%
% Revision 1.14  2002/04/29 21:18:24  arno
% default time limit
%
% Revision 1.13  2002/04/23 23:13:37  arno
% removing channel bug
% ,
%
% Revision 1.12  2002/04/23 17:42:56  scott
% editing legends -sm
%
% Revision 1.11  2002/04/23 17:38:08  scott
% [same] -sm
%
```

```
% Revision 1.10  2002/04/23 17:36:58  scott
% editing coher button legend -sm
%
% Revision 1.9  2002/04/23 17:35:50  scott
% points optional parameter help to timef() help -sm
%
% Revision 1.8  2002/04/23 17:25:34  arno
% adding additional help button
%
% Revision 1.7  2002/04/23 02:49:38  arno
% changed graphic interface
%
% Revision 1.6  2002/04/08 20:27:29  arno
% debuging command line calls
%
% Revision 1.5  2002/04/07 19:18:51  scott
% rm'd int2str() attempt for topovec -sm
%
% Revision 1.4  2002/04/07 19:14:20  scott
% worked on menu text, int2str(num) for topovec call -sm
%
% Revision 1.3  2002/04/06 03:43:36  arno
% adding topoplot options
%
% Revision 1.2  2002/04/05 23:59:06  arno
% correcting figure title
%
% Revision 1.1  2002/04/05 17:32:13  jorn
% Initial revision
%

% 01-25-02 reformated help & license -ad
% 03-08-02 add eeglab option & optimize variable sizes -ad
% 03-10-02 change timef call -ad
% 03-18-02 added title -ad & sm
% 04-04-02 added outputs -ad & sm

function varargout = gvpop_timef(EEG, typeproc, num, tlimits, cycles, varargin
);

varargout{1} = '';
% display help if not enough arguments
% ------------------------------------
if nargin < 2
      help gvpop_timef;
      return;
end;
lastcom = [];
if nargin < 3
      popup = 1;
else
      popup = isstr(num) | isempty(num);
      if isstr(num)
            lastcom = num;
      end;
end;
```

```
% pop up window
% -------------
if popup
      [txt vars] = gethelpvar('gvtimef.m');

      geometry = { [1 0.5 0.5] [1 0.5 0.5] [1 0.5 0.5] [0.92 0.1 0.78] [1 0.5
0.5] [1 0.8 0.2] [1] [1 1]};
    uilist = { ...
                           { 'Style', 'text', 'string', fastif(typeproc,
'Channel number', 'Component number'), 'fontweight', 'bold'  } ...
                     { 'Style', 'edit', 'string', getkeyval(lastcom,3,[],'1') }
{} ...
                     { 'Style', 'text', 'string', 'Epoch time range [min max]
(msec)', 'fontweight', 'bold', ...
                         'tooltipstring', 'Sub epoch time limits' } ...
                     { 'Style', 'edit', 'string', getkeyval(lastcom,4,[],[
int2str(EEG.xmin*1000) ' ' int2str(EEG.xmax*1000) ]) } {} ...
                     { 'Style', 'text', 'string', 'Wavelet cycles (0->FFT, see
>> help gvtimef)', 'fontweight', 'bold', ...
                         'tooltipstring', context('cycles',vars,txt) } ...
                     { 'Style', 'edit', 'string', getkeyval(lastcom,5,[],'3
0.5') } {} ...
                     { 'Style', 'text', 'string',  '[set]->Linear coher /
[unset]->Phase coher', 'fontweight', 'bold', ...
                         'tooltipstring', ['Compute linear inter-trial coherence
(coher)' 10 ...
                              'OR Amplitude-normalized inter-trial phase
coherence (phasecoher)'] } ...
                     { 'Style', 'checkbox', 'value',
~getkeyval(lastcom,'phasecoher','present',1) } { } ...
                     { 'Style', 'text', 'string', 'Bootstrap significance level
(Ex: 0.01 -> 1%)', 'fontweight', 'bold', ...
                         'tooltipstring', context('alpha',vars,txt) } ...
                     { 'Style', 'edit', 'string', getkeyval(lastcom,'alpha') }
{} ...
                     { 'Style', 'text', 'string', 'Optional gvtimef() arguments
(see Help)', 'fontweight', 'bold', ...
                         'tooltipstring', 'See gvtimef() help via the Help
button on the right...' } ...
                     { 'Style', 'edit', 'string', '''padratio'', 4,
''plotphase'',''off'',''erspmax'',5' } ...
                     { 'Style', 'pushbutton', 'string', 'Help', 'callback',
'pophelp(''gvtimef'');' } ...
                     {} ...
                     { 'Style', 'checkbox', 'value',
~getkeyval(lastcom,'plotersp','present',0), 'string', ...
                         'Plot Event Related Spectral Power', 'tooltipstring',
...
                         'Plot log spectral perturbation image in the upper
panel' } ...
                     { 'Style', 'checkbox', 'value',
~getkeyval(lastcom,'plotitc','present',0), 'string', ...
                         'Plot Inter Trial Coherence', 'tooltipstring', ...
                         'Plot the inter-trial coherence image in the lower
panel' } ...
                   };
      % { 'Style', 'edit', 'string', '''padratio'', 4, ''plotphase'', ''off''' }
...
                     %{ 'Style', 'text', 'string',  '[set] -> Plot ITC phase
sign', 'fontweight', 'bold', ...
                  %      'tooltipstring', ['Plot the sign (+/-) of inter-trial
coherence phase' 10 ...
                  %           'as red (+) or blue (-)'] } ...
                  %   { 'Style', 'checkbox', 'value',
~getkeyval(lastcom,'plotphase','present',1) } { } ...

      result = inputgui( geometry, uilist, 'pophelp(''gvpop_timef'');', ...
                                 fastif(typeproc, 'Plot channel time frequency -
- gvpop_timef()', ...
                                            'Plot component time frequency --
gvpop_timef()'));
      if length( result ) == 0 return; end;

      num        = eval( [ '[' result{1} ']' ] );
      tlimits      = eval( [ '[' result{2} ']' ] );
      cycles       = eval( [ '[' result{3} ']' ] );
    if result{4}
      options = [',''type'', ''coher''' ];
    else
            options = [',''type'', ''phasecoher''' ];
    end;

    % add topoplot
    % ------------
    if ~isempty(EEG.chanlocs)
      if typeproc == 1
              options = [options ', ''topovec'', ' int2str(num) ', ''elocs'',
EEG.chanlocs' ];
      else
              options = [options ', ''topovec'', EEG.icawinv(:,' int2str(num)
...
               '), ''elocs'', EEG.chanlocs' ];
      end;
    end;

    % add title
    % ---------
      if isempty( findstr(  '''title''', result{6}))
        if ~isempty(EEG.chanlocs) & typeproc
            chanlabel = EEG.chanlocs(num).labels;
        else
            chanlabel = int2str(num);
        end;
          switch lower(result{4})
             case 'coher', options = [options ', ''title'',' fastif(typeproc,
'''Channel ', '''Component ') chanlabel ...
             ' power and inter-trial coherence' fastif(~ isempty(EEG.setname),
[' (' EEG.setname ')''' ], '''') ];
             otherwise, options = [options ', ''title'',' fastif(typeproc,
'''Channel ', '''Component ') chanlabel ...
             ' power and inter-trial phase coherence' fastif(~
isempty(EEG.setname), [' (' EEG.setname ')''' ], '''') ];
          end;
      end;
      if ~isempty( result{5} )
            options      = [ options ', ''alpha'',' result{5} ];
      end;
      if ~isempty( result{6} )
              options = [ options ',' result{6} ];
      end;
      if ~result{7}
            options = [ options ', ''plotersp'', ''off''' ];
      end;
      if ~result{8}
            options = [ options ', ''plotitc'', ''off''' ];
      end;
      figure; try, icadefs; set(gcf, 'color', BACKCOLOR); catch, end;
else
      options = [];
      for i=1:length( varargin )
            if isstr( varargin{ i } )
                  options = [ options ', ''' varargin{i} '''' ];
            else
              if isstruct( varargin{ i } )
                options = [ options ', EEG.chanlocs' ];
              else
                options = [ options ', [' num2str(varargin{i}(:)') ']' ];
              end;
            end;
      end;
end;

% compute epoch limits
% --------------------
if isempty(tlimits)
      tlimits = [EEG.xmin, EEG.xmax]*1000;
end;
pointrange1 = round(max((tlimits(1)/1000-EEG.xmin)*EEG.srate, 1));
pointrange2 = round(min((tlimits(2)/1000-EEG.xmin)*EEG.srate, EEG.pnts));
pointrange = [pointrange1:pointrange2];

% call function sample either on raw data or ICA data
% ---------------------------------------------------
if typeproc == 1
      tmpsig = EEG.data(num,pointrange,:);
else
      if ~isempty( EEG.icasphere )
        eeg_options; % changed from eeglaboptions 3/30/02 -sm
          if option_computeica
            tmpsig = EEG.icaact(num,pointrange,:);
          else
            tmpsig =
(EEG.icaweights(num,:)*EEG.icasphere)*reshape(EEG.data(:,pointrange,:),
EEG.nbchan, EEG.trials*length(pointrange));
        end;
      else
            error('You must run ICA first');
      end;
end;
tmpsig = reshape( tmpsig, length(num), size(tmpsig,2)*size(tmpsig,3));
```

```
% outputs
% -------
outstr = '';
if ~popup
    for io = 1:nargout, outstr = [outstr 'varargout{' int2str(io) '},' ]; end;
    if ~isempty(outstr), outstr = [ '[' outstr(1:end-1) '] =' ]; end;
end;

% plot the datas and generate output command
% --------------------------------------------
if length( options ) < 2
    options = '';
end;
varargout{1} = sprintf('figure; gvpop_timef( %s, %d, %d, [%s], [%s] %s);',
inputname(1), typeproc, num, ...
                  int2str(tlimits), num2str(cycles), options);
%if is_sccn
%    com = sprintf('%s newtimef( tmpsig(:, :), length(pointrange), [tlimits(1)
tlimits(2)], EEG.srate, cycles %s);', outstr, options);
%else
    com = sprintf('%s gvtimef( tmpsig(:, :), length(pointrange), [tlimits(1)
tlimits(2)], EEG.srate, cycles %s);', outstr, options);
%end;
eval(com)

return;

% get contextual help
% -------------------
function txt = context(var, allvars, alltext);
      loc = strmatch( var, allvars);
      if ~isempty(loc)
            txt= alltext{loc(1)};
      else
            disp([ 'warning: variable ''' var ''' not found']);
            txt = '';
      end;
```