Supplementary Material

A systematic data-driven approach to analyze sensor-level EEG connectivity: Identifying robust phase-synchronized network components using surface Laplacian with spectral-spatial PCA

Ezra E. Smith [a], Tarik S. Bel-Bahar [a], & Jürgen Kayser [a,b,*]

[a] Division of Translational Epidemiology, New York State Psychiatric Institute, New York, NY, USA

[b] Department of Psychiatry, Vagelos College of Physicians & Surgeons, Columbia University, New York, NY, USA

***Psychophysiology*, 2022** (accepted 7 April 2022)

* Corresponding author: Jürgen Kayser, New York State Psychiatric Institute, Unit 50, 1051 Riverside Drive, New York, NY 10032, USA. Tel.: +1 646 774 5207; E-mail address: jurgen.kayser@nyspi.columbia.edu

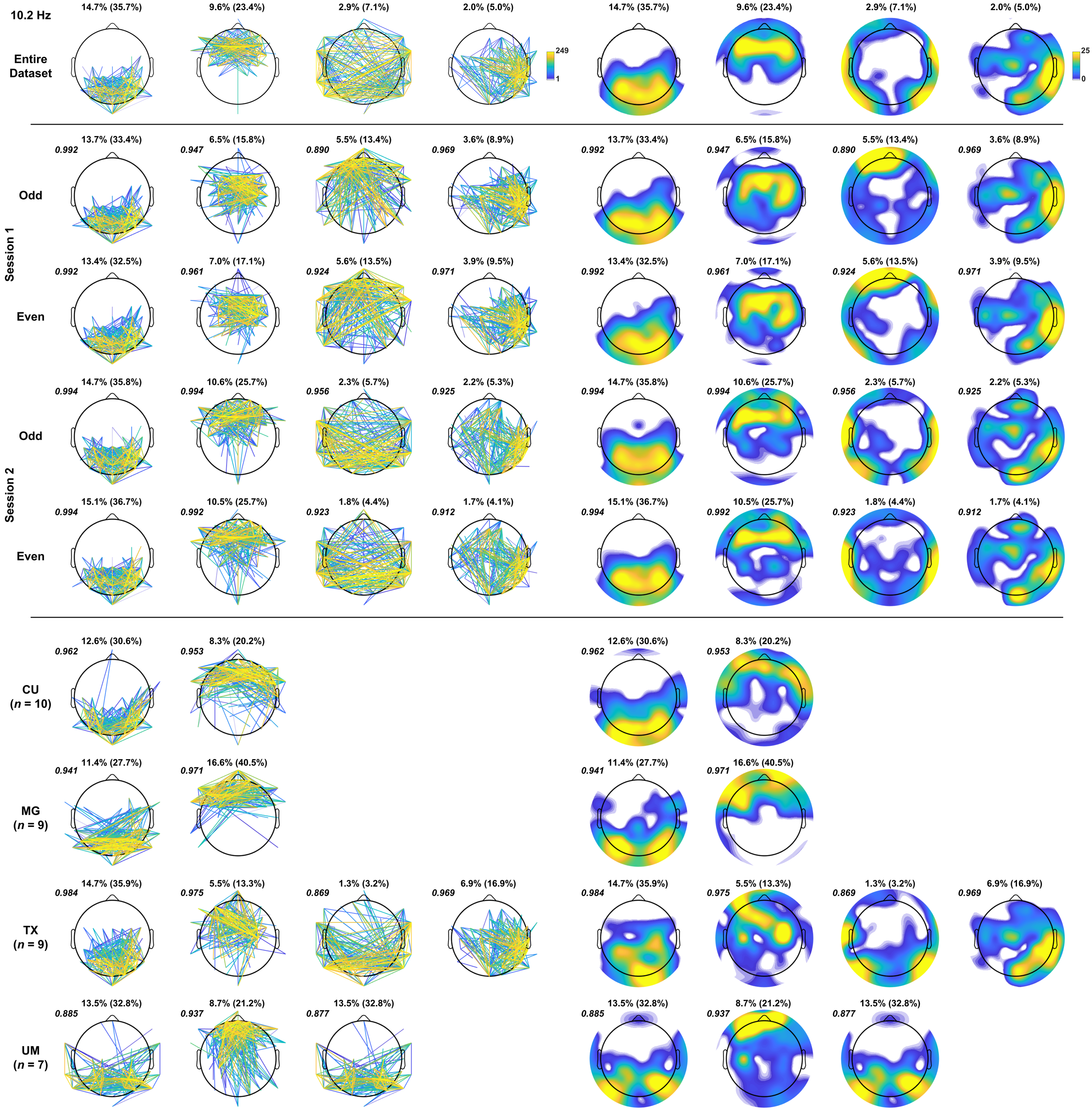


Figure S1. Edge and node strength of mid-alpha (10.2 Hz) spatial FC components identified in the step-two PCA applied to subsample data sets using step-one PCA components derived from the entire dataset (cf. Figures 2 and 3). For comparison, the top row shows edge and node strength topographies of the first 4 spatial FC components obtained for the original PCA solution. Rows depicts the best match for each subsample PCA solution, as determined by Tucker's congruence coefficients ϕ (plotted in italics at the top left corner of each map); only loading maps having at least "fair similarity" (ϕ ≥ .85) are shown. Rows 2-5 show results for subsamples (*N* = 35) created by crossing session (1/2) with epoch (odd/even); rows 6-9 show those for subgroups based on EEG acquisition site (*n* = 7-10). Each subplot title lists the total variance explained and the corresponding step-two variance (in parentheses). Note that for subgroup UM (row 9), the same spatial FC component provided the best match for two different spatial FC components of the entire dataset (row 9, columns 1/5 and 3/7, respectively).

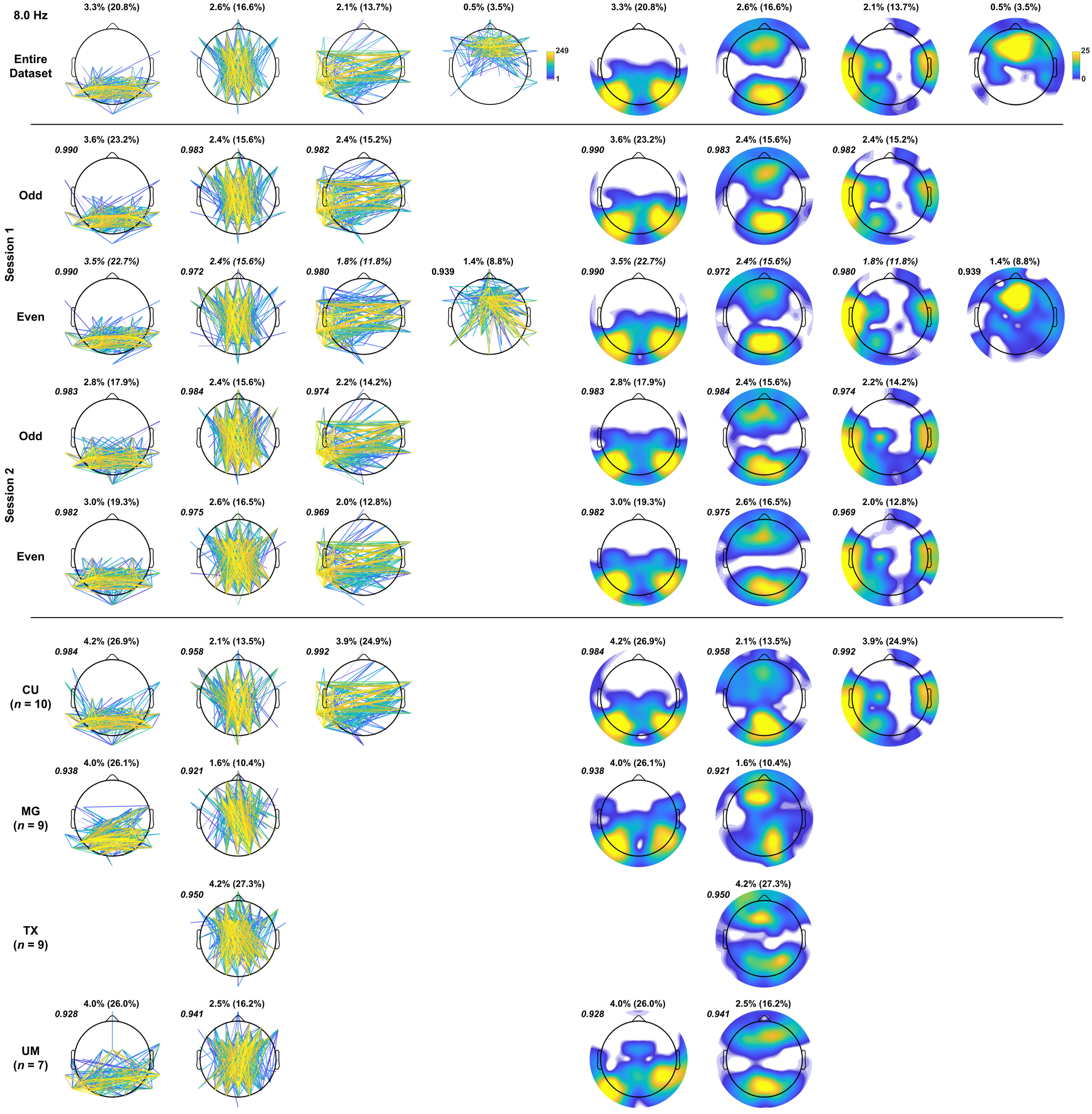


Figure S2. Edge and node strength of low-alpha (8.0 Hz) spatial FC components (all other details as in Figure S1).

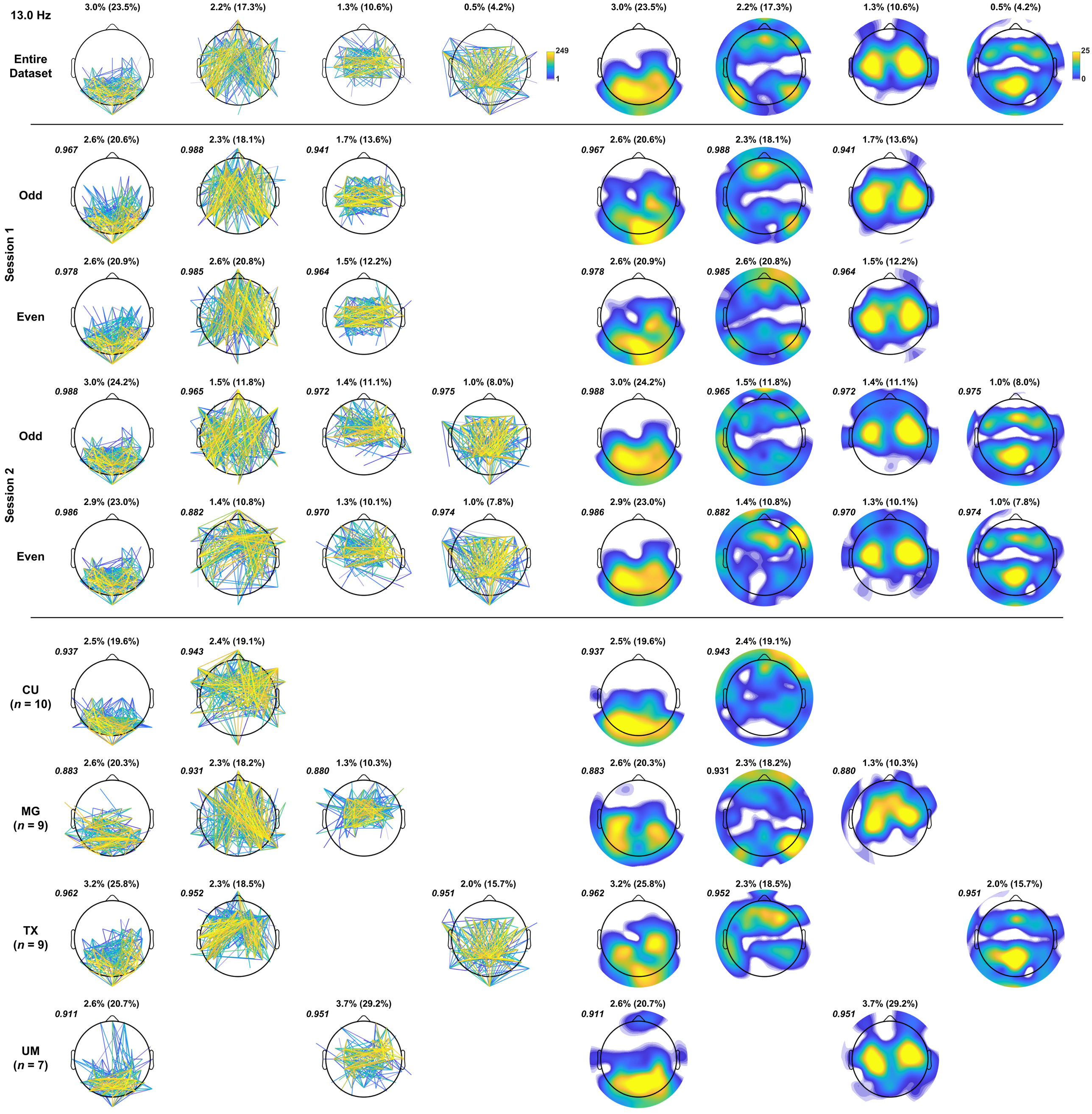


Figure S3. Edge and node strength of high-alpha (13.0 Hz) spatial FC components (all other details as in Figure S1).

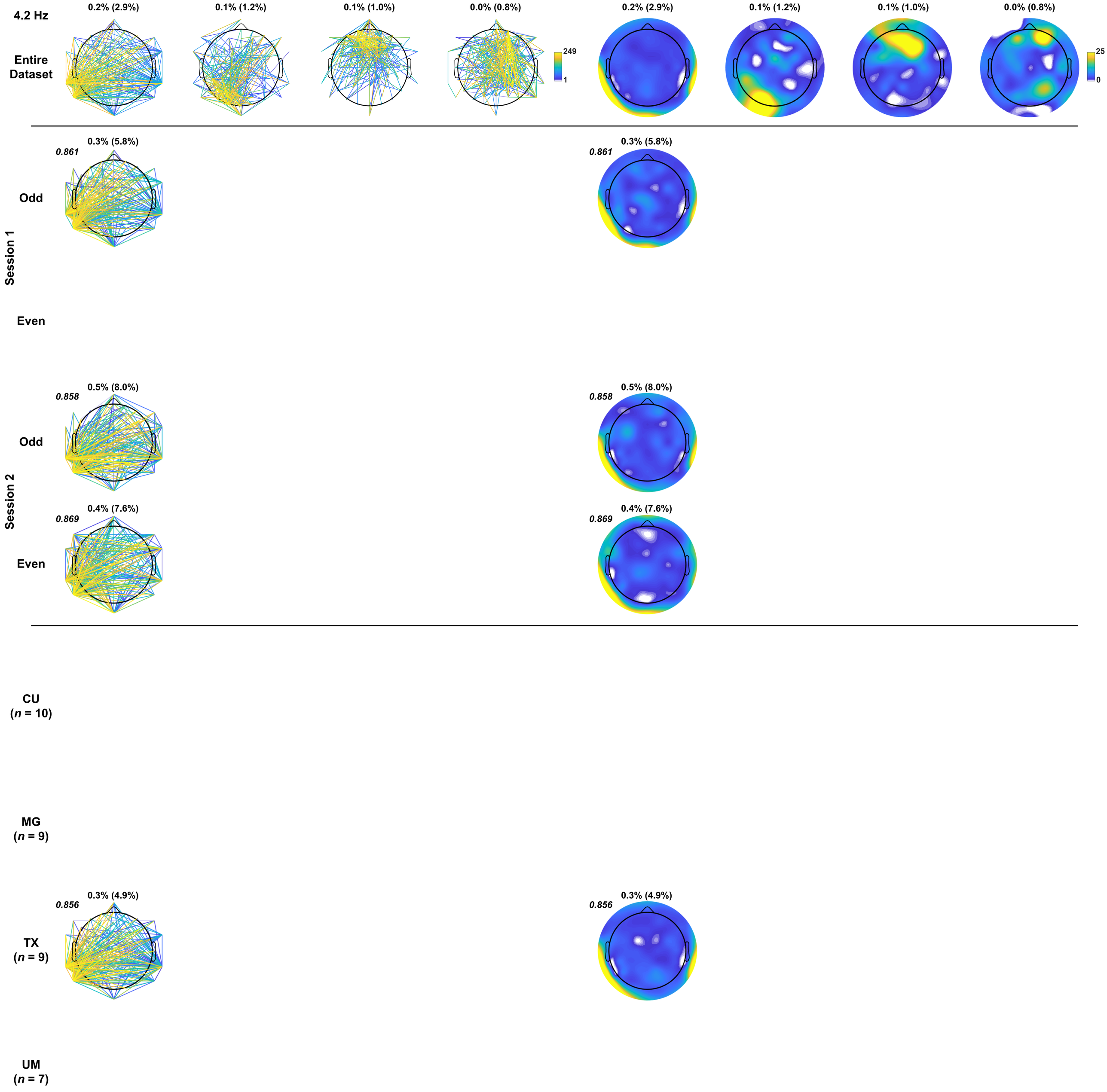


Figure S4. Edge and node strength of low-theta (4.2 Hz) spatial FC components (all other details as in Figure S1).

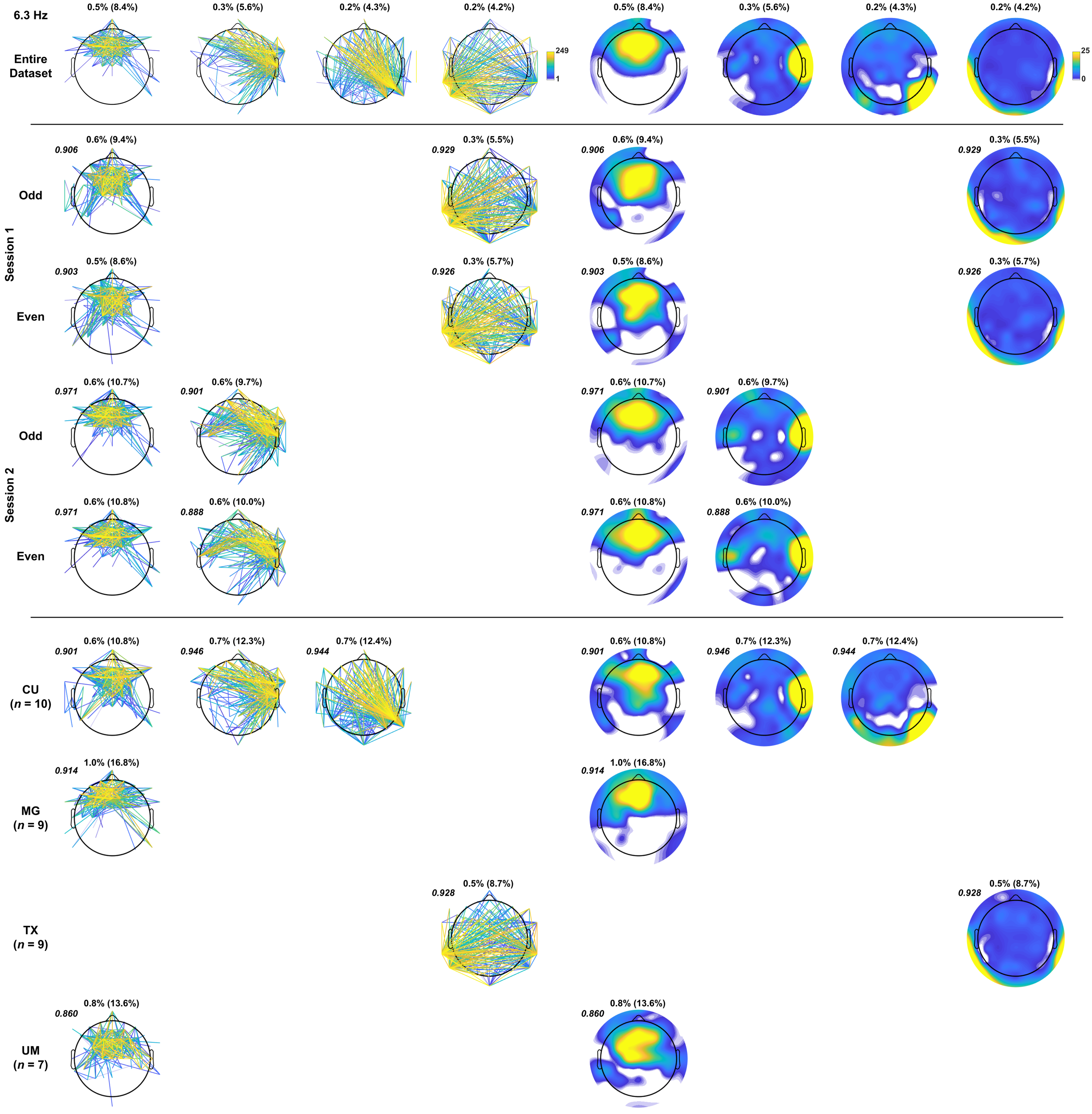


Figure S5. Edge and node strength of mid-theta (6.3 Hz) spatial FC components (all other details as in Figure S1).

5.1 Hz

Entire Dataset

0.1% (2.7%)
0.1% (1.6%)
0.1% (1.2%)
0.0% (0.9%)
0.1% (2.7%)
0.1% (1.6%)
0.1% (1.2%)
0.0% (0.9%)

249
1
25
0

Session 1

Odd

Even

0.5% (8.7%)
0.867
0.5% (8.7%)
0.867

Session 2

Odd

Even

CU
(*n* = 10)

MG
(*n* = 9)

TX
(*n* = 9)

UM
(*n* = 7)

Figure S6. Edge and node strength of low/mid-theta (5.1 Hz) spatial FC components (all other details as in Figure S1).

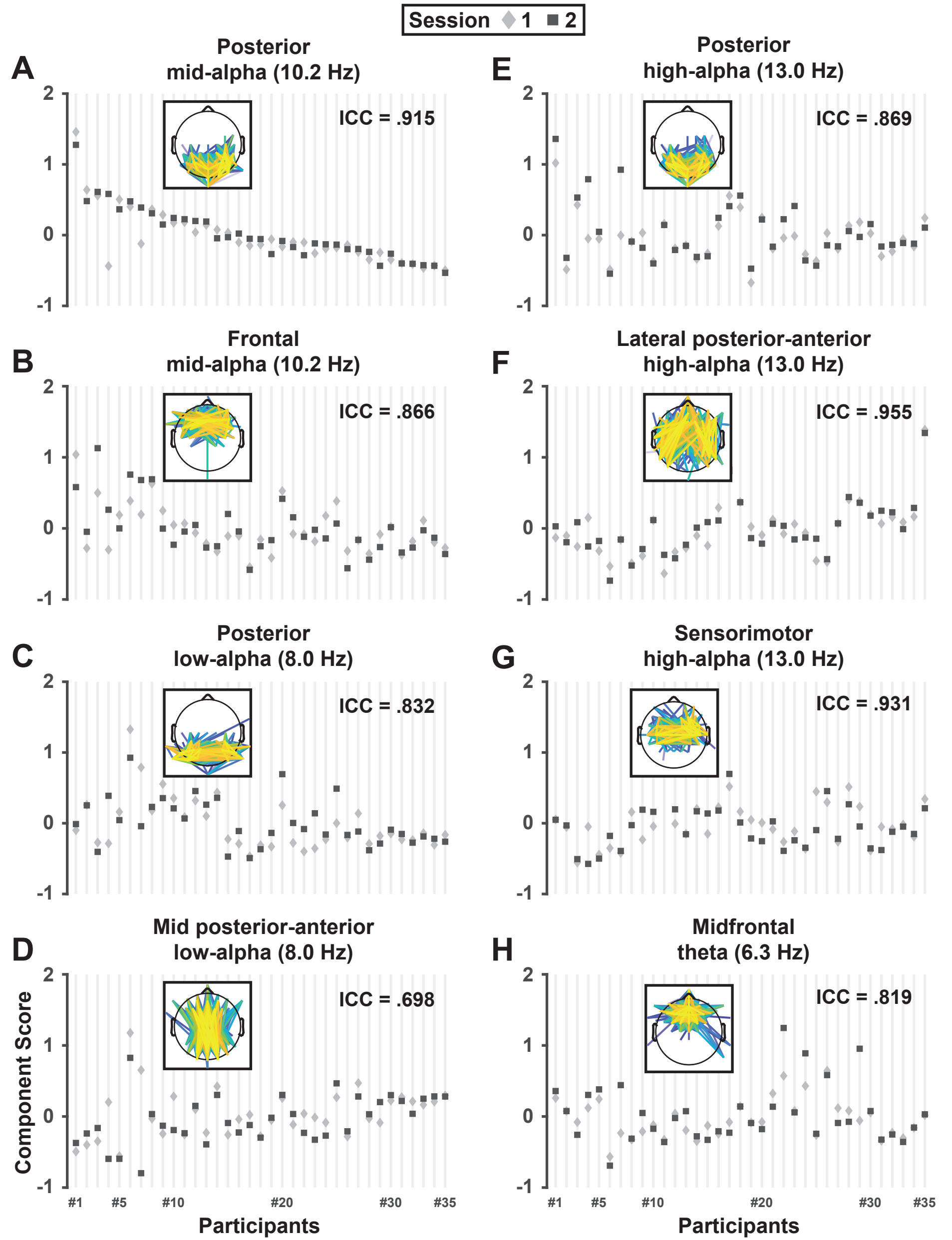


Figure S7. Replotting of Fig. 4 without resorting participants by their maximum score across sessions for each component. The order of participants obtained for panel A is maintained for all other panels (B-H). Across RSNs, different participants contributed the largest (and smallest) component scores.

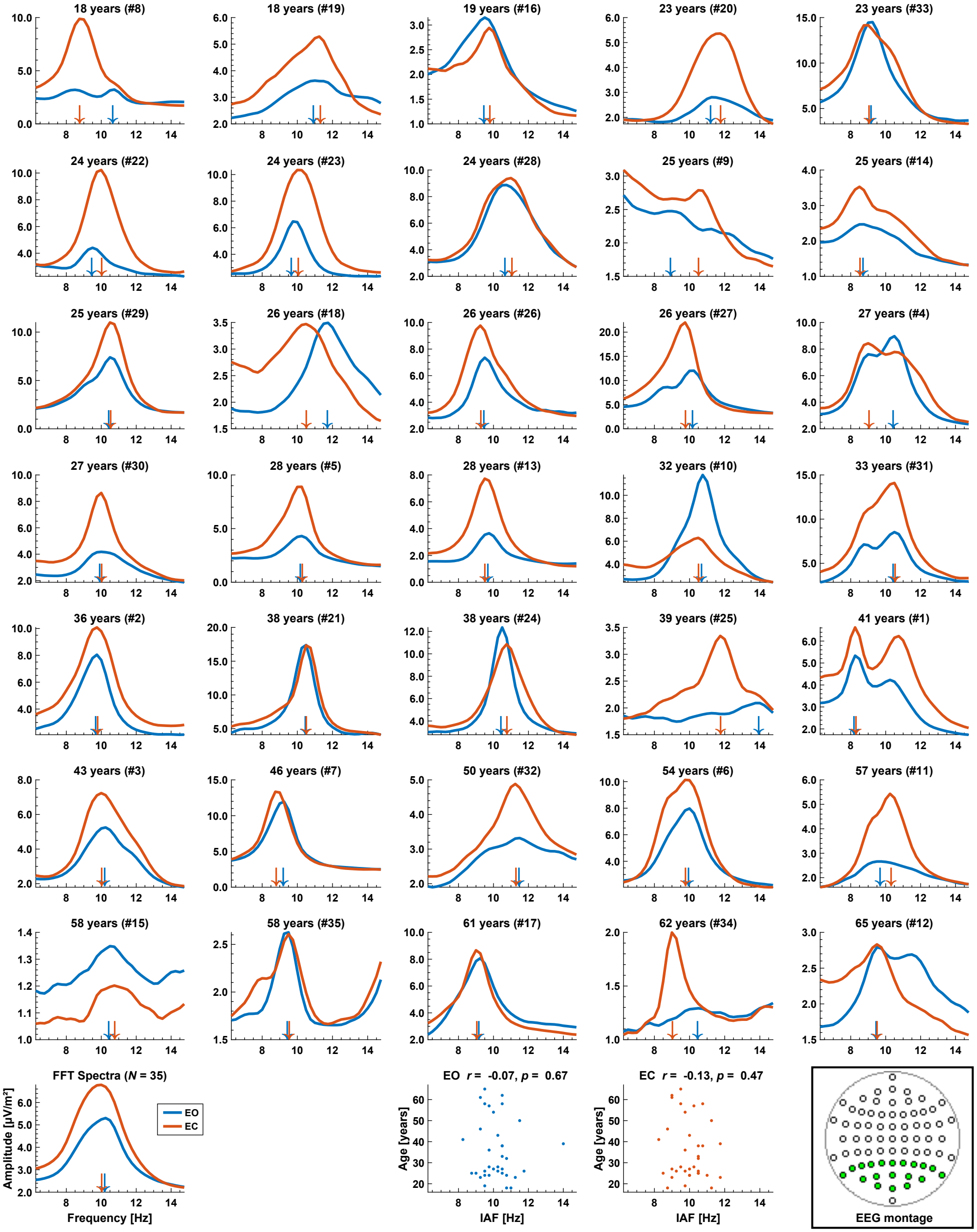


Figure S8. Individual current source density (CSD) amplitude spectra (µV/m²) for eyes open (EO) and eyes closed (EC) conditions pooled across 19 posterior scalp sites (marked green in bottom row inset). Plots for 35 healthy adults (rows 1-7) are sorted by participants' age as noted in subplot title (with subject number). Arrows above abscissa mark individual alpha frequency (IAF) peaks for each condition that were determined as local maxima between 6 and 15 Hz. Grand mean amplitude spectra (N = 35) are shown in bottom row (column 1) along with scatterplots between IAF and age for each condition (columns 3 and 4). Despite the sample's considerable age range (18 to 65 years), age was not systematically associated with IAF estimates (Pearson's r). Likewise, no age/IAF relationship was observed at any of these 19 posterior sites for either condition ($-.24 < r < .22$, all $p > .16$). Importantly, IAF estimates varied considerably across the scalp, manifesting as multiple local maxima in the pooled spectra (e.g., see subjects #4, #31 and #1 in rows 3-5, column 5), which is consistent with prior alpha generator findings (e.g., Smith et al., 2020).

Supplement A

The following excerpts of Matlab code (The MathWorks, Inc., 2022) document the data dimensions and critical computations corresponding to the steps depicted in Figure 1. All code snipsets and functions can be retrieved from http://psychophysiology.cpmc.columbia.edu/CSDfcPCA2022supplA.html.

For the purpose of this documentation, we assume that the preprocessed EEG epochs (artifact-corrected, only valid trials, referenced to any arbitrary scheme [e.g., Nose, linked mastoids, common average]) are stored for each participant in an EEGlab structure *EEG* (Delorme & Makeig, 2004). Resting EEG conditions (eyes open [O]/closed [C]) are appropriately coded (e.g., the block sequence O-C-C-O was coded as 110-20-30-140). For a single subject, the following data dimensions apply (not all fields are shown; cf. Figure 1A):

```
load('EEG1.mat'); % load EEGlab EEG structure for single subject (session 1)

EEG =
  struct with fields:
             setname: 'Neuroscan EEG data pruned with ICA'
            comments: 'Original file: EEG1Bart.eeg'
              nbchan: 71
              trials: 710
                pnts: 512
               srate: 256
                xmin: -0.9990
                xmax: 0.9971
                data: [71×512×710 double]
               epoch: [1×710 struct]

EEG.times = [EEG.xmin:1/EEG.rate:EEG.xmax]*1000; % timeline for each epoch [ms]
```

Epoch tabulating reveals that 428 eyes-closed and 282 eyes-open epochs are included:

```
T = tabulate(cell2mat({EEG.event.epochtype}));
table(T(T(:,2)>0,:))

ans =

  4×1 table

            Var1
    ____________________

     20       214    30.141
     30       214    30.141
    110       143    20.141
    140       139    19.577
```

The respective indices of eyes-closed (EC) and eyes-open (EO) epochs are:

```
ec_idx = find( cell2mat({EEG.event.epochtype})  ==  20 | ...
               cell2mat({EEG.event.epochtype})  ==  30);
eo_idx = find( cell2mat({EEG.event.epochtype})  == 110 | ...
               cell2mat({EEG.event.epochtype})  == 140);
```

Spherical spline interpolations (Perrin et al., 1989) as implemented in the CSD toolbox (Kayser, 2009; for a hands-on tutorial, see *https://psychophysiology.cpmc.columbia.edu/software/csdtoolbox/tutorial.html*) may be used to transform these EEG epochs to corresponding CSD epochs:

```
E = {EEG.chanlocs.labels}';                % get EEG montage and predefined ...
M = ExtractMontage('10-5-System_Mastoids_EGI129.csd',E);  % ... scalp locations
[G,H] = GetGH(M,4);   % create transformation matrices (spline flexibility = 4)
C = CSD(EEG.data,G,H);
```

Given the sample-by-sample character of the CSD transformation, all data dimensions (channels × samples × epochs), including EC/EO indices, are maintained (cf. Figure 1A):

```
size(EEG.data)    % matrix dimension (channels-by-samples-by-epochs)

ans =

    71   512   710

size(C)           % matrix dimension (channels-by-samples-by-epochs)

ans =

    71   512   710

EEG.data = C; % assign CSD-transformed epochs to EEG structure
```

Assuming that the same EEG montage is present for each data recording, the existing *G* and *H* transformation matrices can be used for all other EEG recordings (i.e., subjects × sessions).

Closely following the Matlab code published by Cohen (2014, chapter 26), dwPLI functional connectivity matrices were calculated from the CSD-transformed epochs for each subject and for all conditions (session × eyes open/closed × odd/even epochs) using wavelets as follows (cf. Figure 1C):

```
fx_range = [2 50]; % set frequency range …
num_frex = 40;     % … and number of frequencies
frex = logspace(log10(fx_range(1)),log10(fx_range(2)),num_frex); % wavelet ...
s_all = logspace(log10(3),log10(10),num_frex)./(2*pi*frex);      % ... family
dwPLI  = zeros(EEG.nbchan,EEG.nbchan,EEG.trials,num_frex); % initialize FC matrix
```

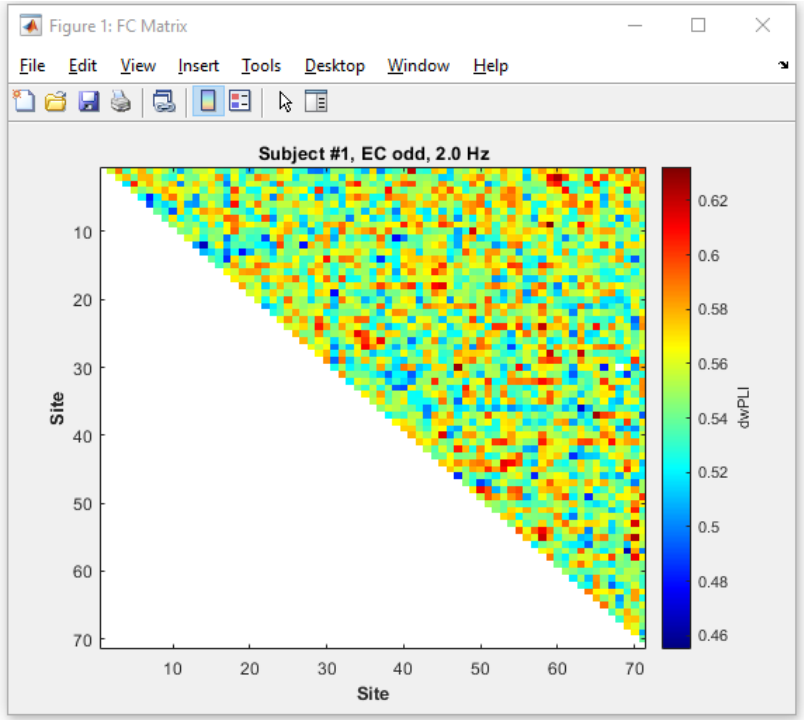
```
time2analyze = [-500 500]; % specify time-frequency parameters [ms]
tidx = dsearchn(EEG.times', time2analyze'); % time to analyze in indices
                                            % setup wavelets
time          = -1:1/EEG.srate:1;           % time [s]
half_wavelet  = (length(time)-1)/2;         % ½-wavelet time points
n_wavelet     = length(time);               % number of wavelet time points
n_data        = EEG.pnts*EEG.trials;        % total number of data points
n_convol      = n_wavelet+n_data-1;         % number of convolution points
WaveletFFT_ = @(nfx,time,n_convol) ...      % box function for wavelet FFT:
    fft(exp(2*1i*pi*frex(nfx).*time) .* ... % nfx = frequency index for ...
    exp(-time.^2./(2*(s_all(nfx)^2))), ...  %       ... center frequency
    n_convol);
sig_ = @(i,A) squeeze(A(i,tidx(1):tidx(2),:)); % box function for data extraction
xsd_ = @(sg1,sg2) sg1 .* conj(sg2);         % box function for complex conjugate

for nfx = 1:num_frex % loop on center frequencies
    tic
    disp(sprintf('%3d/%3d looping. Calculating FC for: %.4f Hz', ...
         nfx,num_frex,frex(nfx)));
    center_freq = frex(nfx); % center frequency [Hz]
    dwpli = zeros(EEG.nbchan,EEG.nbchan,EEG.trials); % initialize fc matrix
    wavelet_fft = WaveletFFT_(nfx,time,n_convol); % create wavelet for FFT
    ASig = zeros(EEG.nbchan,EEG.pnts,EEG.trials); % initialize signal matrix
    for chani = 1:EEG.nbchan % compute analytic signal for all channels
        data_fft = fft(reshape(EEG.data(chani,:,:),1,n_data), ... % FFT of data
                       n_convol);
        conv_result = ifft(wavelet_fft.*data_fft,n_convol); % convolution
        conv_result = conv_result(half_wavelet+1:end-half_wavelet);
        ASig(chani,:,:) = reshape(conv_result,EEG.pnts,EEG.trials);
    end
    SIG = zeros(half_wavelet+1,size(ASig,3),EEG.nbchan); % initialize SIG matrix
    for ii = 1:EEG.nbchan; SIG(:,:,ii) = sig_(ii,ASig); end; % assign signals
    for chani = 1:EEG.nbchan-1 % compute dwPLI for all unique channel pairs
        sig1 = SIG(:,:,chani); % extract data for first channel
        for chanj = chani+1:EEG.nbchan
            sig2 = SIG(:,:,chanj); % extract data for second channel
            xsd = xsd_(sig1,sig2); % complex conjugate
            cdi = imag(xsd);       % take imaginary part of signal only
            imagsum      = sum(cdi);
            imagsumW     = sum(abs(cdi));
            debiasfactor = sum(cdi.^2);
            dwpli(chani,chanj,:) = ... % fc matrix
                 (imagsum.^2 - debiasfactor)./(imagsumW.^2 - debiasfactor);
        end
    end
    dwPLI(:,:,:,nfx) = dwpli; % assign data for this center frequency to FC matrix
    toc
end
```

Means for each condition of interest can now be easily obtained from the dwPLI matrix:

```
                                          % compute means for each condition
EO = squeeze(mean(dwPLI(:,:,eo_idx,:),3)); % mean FC for eyes open ...
EC = squeeze(mean(dwPLI(:,:,ec_idx,:),3)); % ... and eyes closed ...
EO_odd = squeeze(mean(dwPLI(:,:,eo_idx([1:2:end]),:),3));  % ... EO odd epochs ...
EO_even = squeeze(mean(dwPLI(:,:,eo_idx([2:2:end]),:),3)); % ... EO even epochs ...
EC_odd = squeeze(mean(dwPLI(:,:,ec_idx([1:2:end]),:),3));  % ... EC odd epochs ...
EC_even = squeeze(mean(dwPLI(:,:,ec_idx([2:2:end]),:),3)); % ... EC even epochs
```

Example code for visualizing an FC matrix:

```
% visualize FC matrix for first frequency (eyes closed, odd)
for nfx = 21
    idx=triu(ones(EEG.nbchan),+1);
    ec_odd = EC_odd(:,:,nfx);
    ec_odd(~idx)=nan;
    figure('name','FC Matrix');
    ax = imagesc(ec_odd,[min(ec_odd,[],'all') max(ec_odd,[],'all')]);
    xlabel('Site','FontWeight','bold');
    ylabel('Site','FontWeight','bold');
    h = colorbar;
    ylabel(h,'dwPLI');
    colormap([1 1 1; jet]);
    title(sprintf('Subject #%d, %s, %.1f Hz',1,'EC odd',frex(nfx)));
end
```



After obtaining all FC matrices (subject × session × eyes open/closed × odd/even epochs), all dwPLI values are re-arranged into an edges-by-frequency bins matrix:

```
n_sites = EEG.nbchan; % number of electrode sites
n_edges = (n_sites * (n_sites-1))/2; % number of unique channel pairs
A = zeros(n_edges,num_frex); % initialize matrix
for nfx = 1:num_frex % loop through all frequencies
    ec_odd = EC_odd(:,:,nfx); % get FC matrix for one condition and mark ...
    ec_odd(~idx)=nan;         % ... lower triangle and diagonale as NaN
    d = reshape(ec_odd,n_sites^2,1); % reshape to column vector and ...
    A(:,nfx) = d(~isnan(d));         % ... remove all NaNs
end

B = zeros(n_edges,2*num_frex); % initialize matrix for frequency interpolation
InterpFreqs = logspace(log10(fx_range(1)),log10(fx_range(2)),2*num_frex);
```

```
for c = 1:n_edges                                % loop through all edges
    B(c,:) = interp1(frex,A(c,:),InterpFreqs); % interpolate frequency bins
end

fx_range2 = [3 16]; % set new frequency range [Hz]
R = InterpFreqs >= fx_range2(1) & InterpFreqs <= fx_range2(2);
frex2 = InterpFreqs(R); % interpolated frequency bin vector
B2 = B(:,R); % matrix for interpolated new frequency range

frex2([1:4 end-3:end]) % first and last four interpolated frequency bins

ans =

    3.0060    3.1310    3.2612    3.3968  ...  14.1388   14.7267   15.3392   15.9771

size(B2)               % matrix dimension (edges-by-frequency bins)

ans =

        2485          42
```

After re-arranging all FC matrices (subject × session × eyes open/closed × odd/even epochs), these are vertically concatenated to create a single 2-D data matrix, consisting of 695,800 observations or cases (rows = 35 subjects × 2 sessions × 2 eyes open/closed × 2 odd/even epochs × 2485 edges) and 42 variables (columns). With this combined data matrix *D1*, a solution for step 1 (spectral FC PCA) is obtained as follows (cf. Figure 1D):

```
size(D1)    % dimensions (edges-by-frequency bins) of dwPLI data matrix
disp(sprintf('dwPLI value range: %.3f .. %.3f',min(min(D1)),max(max(D1))));
Y = erpPCA_Varimax4M_progress(D1) % PCA solution for step 1
```

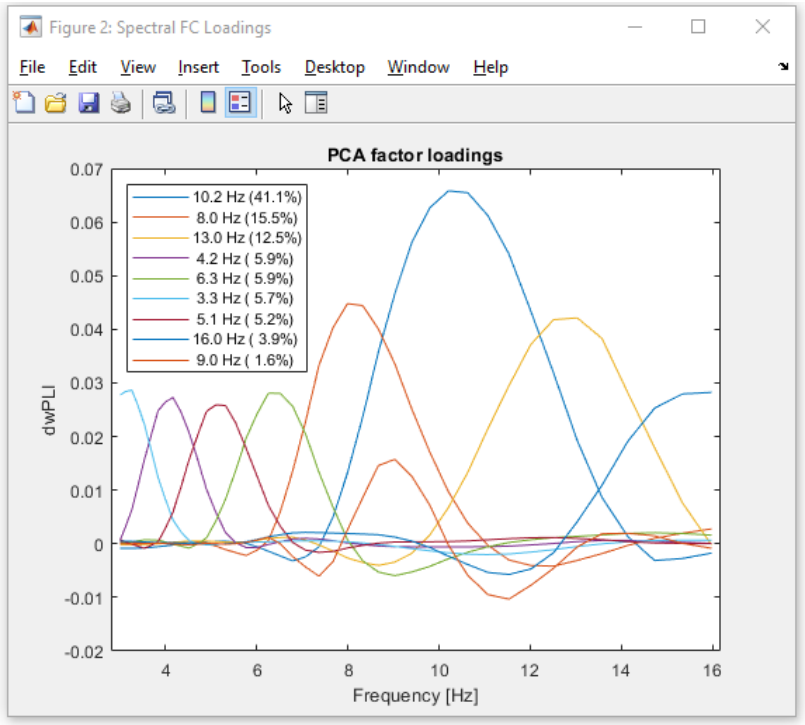
The core function `erpPCA_Varimax4M_progress.m` is an improved implementation of function `erpPCA.m` originally published in the appendix of Kayser & Tenke (2003). This original function, along with two required Matlab routines (`Varimax4M.m, SimplicityG.m`), can be retrieved from *https://psychophysiology.cpmc.columbia.edu/mmedia/Kayser2003a/Appendix.html*.

The PCA solution *Y* includes the rotated factor loadings *LR*, a variance table *VT*, and the grand mean *mu* with its standard deviation *sigma*:

```
Y =

  struct with fields:

       LU: [42×23 double]
       LR: [42×23 double]
      FSr: [695800×12 double]
       VT: [23×4 double]
        G: [23×4 double]
       mu: [1×42 double]
    sigma: [1×42 double]
    FSCFr: [42×23 double]
     rank: 23
```

The factor loadings can be labelled and visualized as follows:

```
figure('name','Spectral FC Loadings');
n_factors = size(Y.LR,2); % number of extracted PCA factors
[m,idx] = max(Y.LR); % get maxima of factor loadings with index
clear flab; % get labels: peak loadings frequency (variance explained)
for i = 1:n_factors
    fspec = ['%4.1f Hz (%4.1f%%)'];
    flab(i,:) = {sprintf(fspec,frex2(idx(i)),Y.VT(i,4))};
end;
R = Y.VT(:,4) >= 1;        % plot factor loadings >= 1% explained variance
plot(frex2,Y.LR(:,R'));
xlabel('Frequency [Hz]');
ylabel('dwPLI');
ax = gca;
ax.XLim = [frex2(1)-0.2 frex2(end)+0.2];
title('PCA factor loadings');
legend(flab(R),'location','northwest');
```



For the next step (cf. Figure 1E), each factor of interest is back-projected to dwPLI data space using the fact that a PCA solution represents a linear, multivariate decomposition of the original data matrix, which is now represented by factor loadings (across variables), associated factor scores (across cases), and the matrix grand mean. Using all extracted factors and the grand mean with an unrestricted PCA solution will recreate the original data matrix (see also the appendix of Kayser et al., 2019). Conversely, using only a subset of factors will create a virtual data matrix, effectively a factor-based

filter of the original data. In the current scenario, only a single factor was back-projected at a time without re-inserting the grand mean to the recreated data matrix:

```
theFact = 1;                      % spectral factor for back-projection ...
disp(sprintf('Factor %d: %s',theFact,flab{theFact}));
U = Y.FSr(:,theFact) * Y.LR(:,theFact)';         % ... to dwPLI unit space
size(U) % matrix dimension (all edges-by-frequency bins)
```

For the final step, the back-projected dwPLI values in the matrix *U* are transposed for each FC matrix (i.e., from 2,485 edges × 42 frequency bins to 42 frequency bins × 2,485 edges):

```
n_freqs = size(U,2); % number of frequency bins
n_all = size(U,1)/n_edges; % number of all FC matrices
D2 = zeros(n_all*n_freqs,n_edges); % initialize re-arranged matrix <U>
for i = 1:n_all
    D2((i-1)*n_freqs+1:i*n_freqs,:) = ... % transpose each FC matrix and ...
      U((i-1)*n_edges+1:i*n_edges,:)';    % ... assign to new matrix <D2>
end
```

With this new spectrally-filtered FC data matrix *D2*, a solution for step 2 (spatial FC PCA) is obtained as follows (cf. Figure 1F):

```
Y2 = erpPCA_Varimax4M_progress(D2) % PCA solution for step 2

Y2 =

  struct with fields:

       LU: [2485×280 double]
       LR: [2485×280 double]
      FSr: [11760×245 double]
       VT: [280×4 double]
        G: [101×4 double]
       mu: [1×2485 double]
    sigma: [1×2485 double]
    FSCFr: [2485×280 double]
     rank: 280
```

For additional step-2 PCAs involving data subsets, component extraction was restricted to 50 to reduce computation time using the modified syntax:

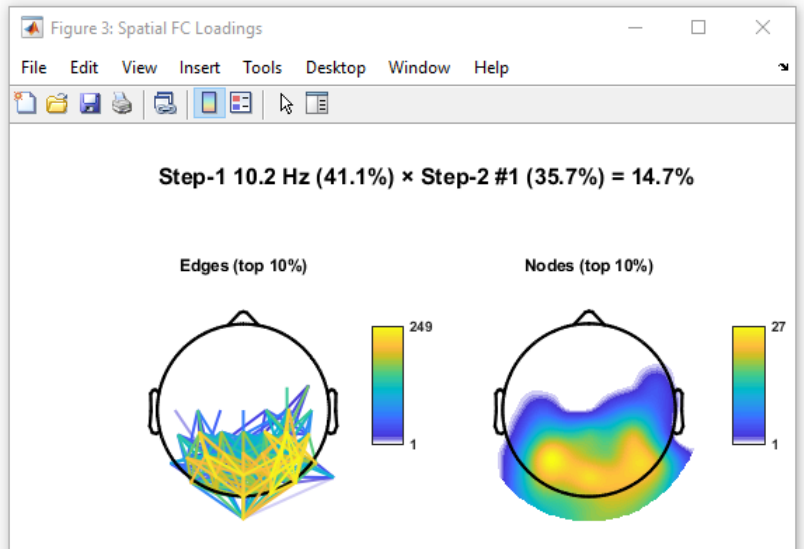
```
Y2 = erpPCA_Varimax4M_progress(D2,'rank',50) % syntax for data subsets
```

These step-2 factor loadings can be visualized as follows:

```
figure('name','Spatial FC Loadings');
theFact2 = 1; % spatial factor selected for visualization
subplot(1,2,1); % plot edges
axis square;
load('chanlocs.mat'); % load EEGlab structure with EEG montage locations
pct2display = 10; % display top 10% of loadings
load('mod_parula.mat');      % load and assign a color ...
cmap = colormap(mod_parula); % ... map for plotting
```

```
ConnPCA_Plot_Pct_Top_Links(Y2.LR,theFact,n_sites,chanlocs,pct2display,cmap);
cbe = colorbar;
topidx = 1:round([(pct2display/100)*size(Y2.LR,1)]);
set(cbe,'Position',[.46 .46 .04 .2], ... % include scale
        'Ticks',[0 1],'TickLabels',[1 length(topidx)],'FontWeight','bold');
Step1Var = Y.VT(theFact,4);    % variance of step-1 PCA solution
Step2Var = Y2.VT(theFact2,4); % variance of step-2 PCA solution
Step1x2Var = (Step1Var/100)*(Step2Var/100)*100; % total explained variance
title(sprintf('Edges (top %.0f%%)',pct2display));
```

The function `ConnPCA_Plot_Pct_Top_Links.m`, an adaption of routines developed by Johann (2022), depicts the strongest edges (specified here as the top 10%) via a head map showing linked pairs of sites (left subplot in figure below).




```
subplot(1,2,2); % plot node degree
[B,I] = sort(Y2.LR(:,theFact2),1,'descend'); % identify suprathreshold connections
top_loads = I(topidx);
axis square;
nLR = nan(n_edges,1);
nLR(top_loads,1) = Y2.LR(top_loads,theFact2);
binarizedLR = nLR(:,1);
binarizedLR(~isnan(binarizedLR)) = 1;
adjmat = nan(n_sites,n_sites);
c = 0;
for i = 1:n_sites-1
    for j = i+1:n_sites
        c = c + 1;
        adjmat(i,j) = binarizedLR(c,1);
    end
end
symmat = triu(adjmat) + triu(adjmat)';
node_degree = nansum(symmat);
topoplot_MG(node_degree,chanlocs, 'electrodes','off', 'style','map', ...
            'shading','interp', 'whitebk','on', ...
            'maplimits',[0 max(node_degree)],'colormap',cmap);
cbn = colorbar;
set(cbn,'Position',[.92 .46 .04 .2], ... % include scale
        'Ticks',[0 max(node_degree)],'TickLabels',[1 max(node_degree)], ...
        'FontWeight','bold');
```

```
title(sprintf('Nodes (top %.0f%%)',pct2display));
set(gcf,'color','w'); % set figure background to white
```

The EEGlab function `topoplot_MG.m` (modified from its original `topoplot.m` [© Colin Humphries & Scott Makeig] by Michael Goldstein) was used to depict the most-connected nodes (right subplot in figure above).

Finally, the following code inserts a main title giving the respective explained variance of the depicted component for the step-1 and step-2 PCA solutions as well as their combined overall total variance explained:

```
MainTitle = sprintf('Step-1 %s × Step-2 #%d (%.1f%%) = %.1f%%', ...
                    flab{theFact},theFact2,Step2Var,Step1x2Var);
annotation('textbox',[.13 .85 .8 .1], 'string',MainTitle, ...
           'HorizontalAlignment','center', 'FontWeight','bold', ...
           'LineSTyle','none', 'FontSize',12);
```

References

Cohen, M.X. (2014). Analyzing neural time series data: theory and practice. Cambridge, MA: MIT Press.

Delorme, A., Makeig, S. (2004). EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis. J. Neurosci. Methods, 134(1), 9-21. https://doi.org/10.1016/j.jneumeth.2003.10.009

Johann (2022). Easy Plot EEG Brain Network Matlab. MATLAB Central File Exchange. https://www.mathworks.com/matlabcentral/fileexchange/57372-easy-plot-eeg-brain-network-matlab

Kayser, J. (2009). Current source density (CSD) interpolation using spherical splines - CSD Toolbox (Version 1.1). http://psychophysiology.cpmc.columbia.edu/Software/CSDtoolbox

Kayser, J., & Tenke, C.E. (2003). Optimizing PCA methodology for ERP component identification and measurement: theoretical rationale and empirical evaluation. Clin. Neurophysiol., 114, 2307-2325. https://doi.org/10.1016/s1388-2457(03)00241-4

Kayser, J., Tenke, C.E., Svob, C., Gameroff, M.J., Miller, L., Skipper, J., Warner, V., Wickramaratne, P., Weissman, M. M. (2019). Family risk for depression and prioritization of religion or spirituality: early neurophysiological modulations of motivated attention. Front. Hum. Neurosci., 13, 436. https://doi.org/10.3389/fnhum.2019.00436

Perrin, F., Pernier, J., Bertrand, O., & Echallier, J.F. (1989). Spherical splines for scalp potential and current density mapping. EEG Clin. Neurophysiol., 72, 184-187. [Corrigenda EEG 02274, EEG Clin. Neurophysiol., 1990, 76, 565] https://doi.org/10.1016/0013-4694(89)90180-6

The MathWorks, Inc. (2022). MatLab. The language of technical computing. https://www.mathworks.com/products/matlab.html