A SIMPLE TUTORIAL TO START WITH FUNCTIONAL NETWORKS.

Table of Contents

Here we are! The main idea is to who how we can build up a functional brain network from a EEG recording database, then visualize some network features in the brain network. So let's star!

Author

Johann H. Martínez https://johemart.wixsite.com/neurocomplexity

Brief Data Description

20 healthy individual who were given the sedative propofol.

```
clearvars;  clc;    close all; % command for clear screen, command
 windows and clossing all open figures
load ALLEEG.mat;
% % 91 channels , 2500 timepoints , 20 subject x 4 conditions = 80
 essays
load('-mat','02-2010-anest 20100210 135.003.set')
% % extra info of dataset, e.g., time in ms, sampling rate in Hz and
 bands.
```

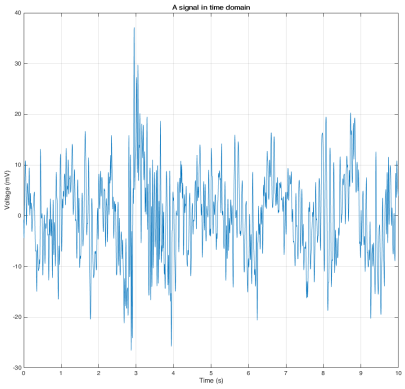
An EEG and its power.

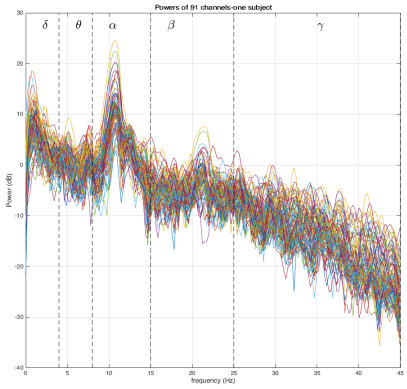
Here, it is single time series, then we plot a power of 91 signals of the same subject to see diferent power peaks. Note how the second subplot highlight the frequency bands.

```
ch = 1;             % Taking channel one
cd = 2;             % taking second condition
```

```
x = data(ch,:,cd);  % this is the resulting time series

figure;
subplot(121)
plot(EEG.times*1e-3, x)
grid on
xlabel('Time (s)')
ylabel('Voltage (mV)')
title('A signal in time domain')

subplot(122)
plot(EEG.freqs, EEG.spectra(:, :, 1))
xlim([0 45])
xlabel('frequency (Hz)')
ylabel('Power (dB)')
title('Powers of 91 channels-one subject')
ylim([-40 30])
grid on
hold
plot([EEG.freqwin(2) EEG.freqwin(2)],[-40 30], '--k')
plot([EEG.freqwin(3) EEG.freqwin(3)],[-40 30], '--k')
plot([EEG.freqwin(4) EEG.freqwin(4)],[-40 30], '--k')
plot([EEG.freqwin(5) EEG.freqwin(5)],[-40 30], '--k')
plot([EEG.freqwin(6) EEG.freqwin(6)],[-40 30], '--k')
text(2,27.5, '\delta', 'FontSize', 20)
text(6,27.5, '\theta', 'FontSize', 20)
text(10,27.5, '\alpha', 'FontSize', 20)
text(17,27.5, '\beta', 'FontSize', 20)
text(35,27.5, '\gamma', 'FontSize', 20)
set(gcf, 'units','normalized','outerposition',[0 0 1 1]) % EXPANDING
 FIGURE ON SCREEN
```

```
Current plot held
```

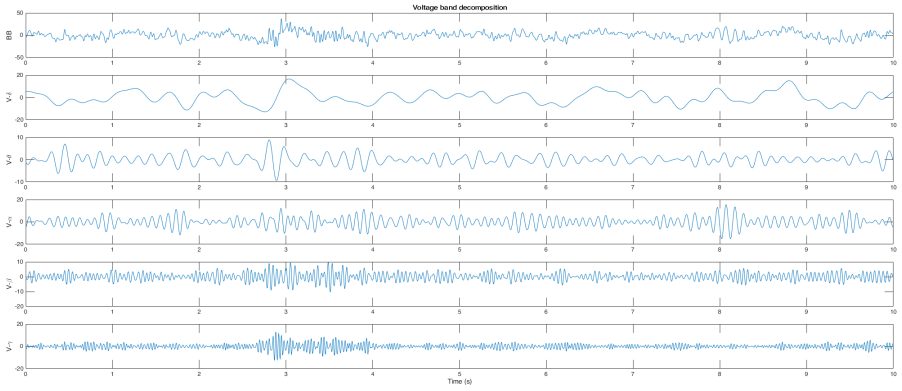




Band decompotition of one signal

The signal is now filtered under different frequency bands. Here we can observe the broad band signal and its recontructions from 5 frecuency bands. The function " `bandpass2.m` " takes a time series " X ", the sampling rate, here as " `EEG.srate` ", and the limits of the frequency band.

```
xb1 = bandpass2(0, 4, EEG.srate, x);    % signal filtered in delta
 band
xb2 = bandpass2(4, 8, EEG.srate, x);    % signal filtered in theta
 band
xb3 = bandpass2(8, 15, EEG.srate, x);   % signal filtered in alfa band
xb4 = bandpass2(15, 25, EEG.srate, x);  % signal filtered in beta band
xb5 = bandpass2(25, 45, EEG.srate, x);  % signal filtered in gamma
 band

figure;
subplot(611)
plot(EEG.times*1e-3, x)
ylabel('BB')
title('Voltage band decomposition')
subplot(612)
plot(EEG.times*1e-3, xb1)
ylabel('V-\delta')
subplot(613)
plot(EEG.times*1e-3, xb2)
ylabel('V-\theta')
subplot(614)
plot(EEG.times*1e-3, xb3)
ylabel('V-\alpha')
subplot(615)
plot(EEG.times*1e-3, xb4)
ylabel('V-\beta')
subplot(616)
plot(EEG.times*1e-3, xb5)
xlabel('Time (s)')
ylabel('V-\gamma')
set(gcf, 'units','normalized','outerposition',[0 0 1 1]) %EXPANDING
 FIGURE ON SCREEN
```



Gahtering dataset of 20 subject in condition 2 (mild sedation)

We state " *cd = 2* " as the variable to choose the second condition. Observing the file " `datainfo` ", we detect that second column declares the level of sedation (condition) and the data organization (same condition each 4 files). Dataset fieds are: filename | sedation level (1=baseline, 2=mild, 3=moderate, 4=recovery) % propofol concentration in plasma (microgram/litre) | correct responses in a task

```
load datainfo.mat;              % file to reveal the order at which
 date has been saved
cd = 2;
subper_cd = cd:4:80;            % by observing "datainfo.mat" we know
 the same condition is saved each four files
subs_cd2 = data(:,:,subper_cd); % 91 time series of 2500 points for 20
 subjects under mild sedation
```

Filtering alpha band the previous dataset

As we observed alpha band as the one of higher power, we filter previous dataset of mild sedation in alpha frequency band. Variable " *subs_cd2_al* " contains the 91 times series for 20 subjects in mild sedation for alpha band.

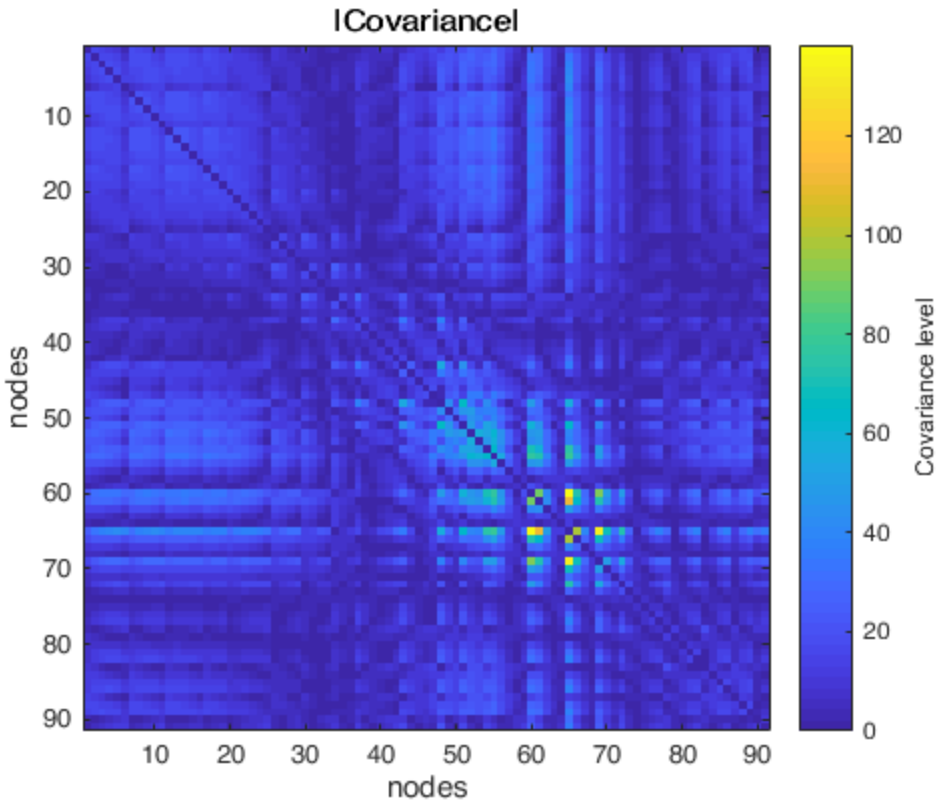
```
subs_cd2_al = zeros(size(subs_cd2));
for su = 1 : 20
    for ch = 1 : 91
        subs_cd2_al(ch,:,su) = bandpass2(8, 15, EEG.srate,
 subs_cd2(ch, :, su));
    end
end
```

Covariance

Covariance gives the level of variability of random variables respect their means.

$$cov(X,Y) = E[(X - E[X])(Y - E[Y])]$$

```
cov_cd2 = cov(subs_cd2_al(:,:,1)');
cov_cd2 = abs(cov_cd2 - diag(diag(cov_cd2)));

figure;
imagesc(cov_cd2);
axis square;
cb = colorbar;
cb.Label.String = 'Covariance level';
title('|Covariance|')
ylabel('nodes');    xlabel('nodes')
```

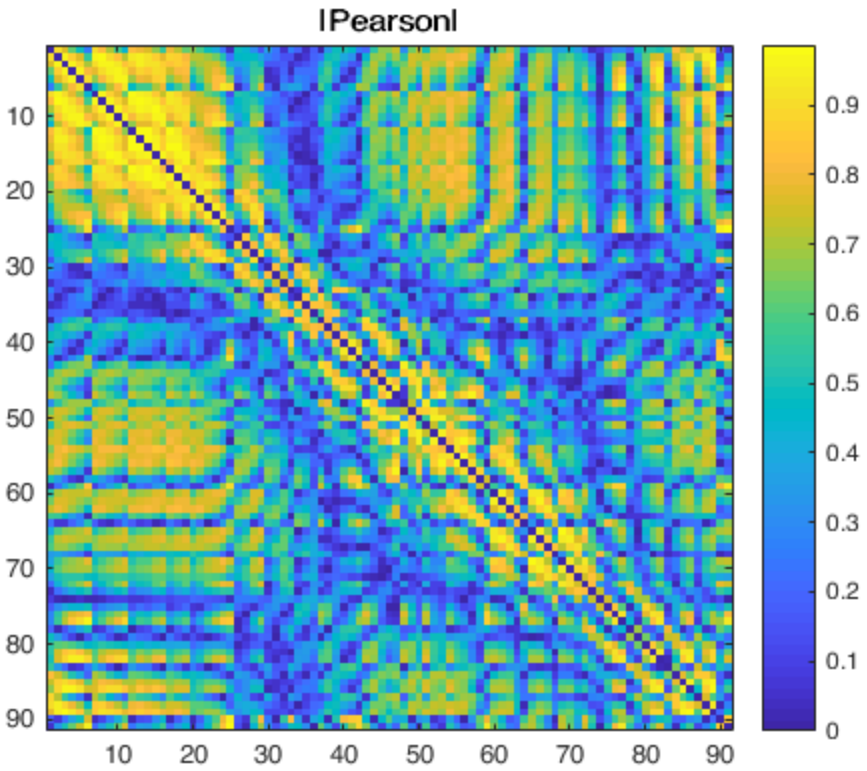


Pearson

Linear correlaton between two random variables. In other words, how close the interdependency is near to a linear relation. It's also known to be the Covariance normalized as well as the $\tau = 0$ of the Cross-Correlation function.

$$Pearson(X,Y) = \frac{Cov(X,Y)}{\sigma_X \sigma_Y}$$

```
pea_cd2 = corrcoef(subs_cd2_al(:,:,1)');
pea_cd2 = abs(pea_cd2 - diag(diag(pea_cd2)));

figure;
imagesc(pea_cd2);
axis square;    colorbar;
title('|Pearson|')
```



Phase Locking Value (PLV)

How well waves' phases are locked in a phase synchrony. The main idea is to take the phases of waveforms by taking into account the real part of the Hilbert Transform of signals. Then applying the equaton.

$$PLV(\theta) = \frac{1}{N}|\sum_{n=1}^{N} e^{i\Delta\theta}|$$

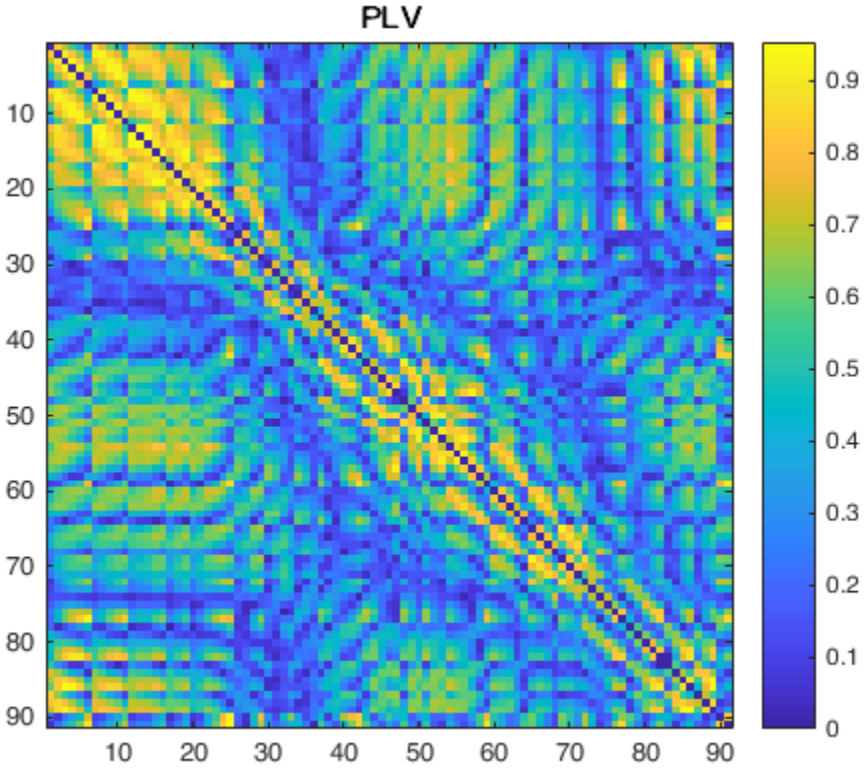
Here we take phases for subject one. Variable " *phis* " is then the phases of 91 signals of subject one.

```
sujeto = 1;
phis = angle(hilbert(subs_cd2_al(:,:,sujeto)'));
```

Now computing PLV over phases "phis" of subject one.

```
pairs = nchoosek(1:91,2);           % all combinations of 91 channels
 in groups of two
plv = zeros(91,91);
for pr = 1 : length(pairs)
    plv(pairs(pr,1), pairs(pr,2)) = abs(sum(exp(1i*
 ( phis(:,pairs(pr,1)) - phis(:,pairs(pr,2)) ))))/length(phis);
end
plv = plv + plv';                   % we take its transpose to create
 the connectivity matrix

figure
```

```
imagesc(plv);colorbar;
axis square;    colorbar
title('PLV')
```



Magnitude-Squared Coherence

Linear correlation between power espectrum in a frequency f. It is sensitive to both phase and amplitude. We take the average over the segments of the FFT.

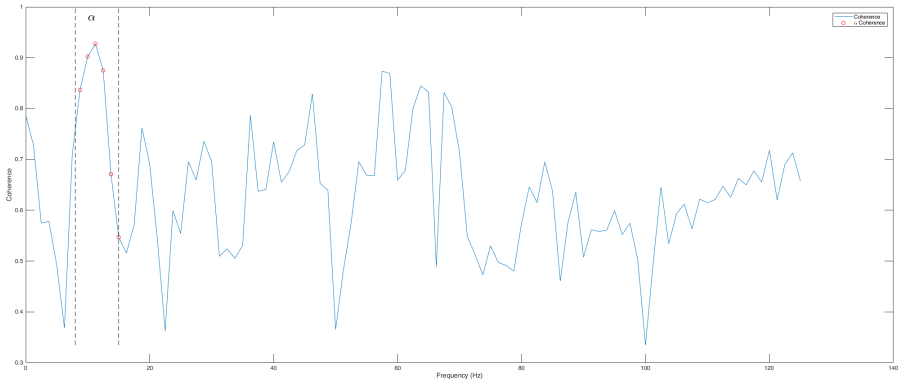
$$Coherence(Y,Y) = \frac{|S_{XY(f)}|^2}{|S_{XX(f)}S_{YY(f)}|^2}$$

We take two signals of subject one and visualize its coherence. When computing coherence, e.g., in alpha band, we will average the coherence points in the limits of such band (Here they are red dots), and that mean value is called as the coherence for two signals in alpha band.

```
sujeto = 1;
[coh, f] = mscohere(subs_cd2(1, :, sujeto), subs_cd2(2 ,:, sujeto),
 hanning(200), 25, 200, EEG.srate);

figure
plot(f,coh, '-')
hold
plot(f(f>=EEG.freqwin(3) & f<=EEG.freqwin(4)), coh(f>=EEG.freqwin(3) &
 f<=EEG.freqwin(4)), 'ro')
```

```
plot([EEG.freqwin(3) EEG.freqwin(3)], [min(coh) 1], '--k')
plot([EEG.freqwin(4) EEG.freqwin(4)], [min(coh) 1], '--k')
legend('Coherence', '\alpha Coherence')
xlabel('Frequency (Hz)')
ylabel('Coherence')
text(10,0.98, '\alpha', 'FontSize', 20)
set(gcf, 'units','normalized','outerposition',[0 0 1 1]) %EXPANDING
 FIGURE ON SCREEN
```

Current plot held



Now we will take all signals of subject one in order to compute the coherence of all pair of signals in alpha band.

```
% % % sujeto = 1;
% % % pairs = nchoosek(1:91,2);
% % % cxy = zeros(91,91);
% % % for pr = 1 : length(pairs)
% % %     [coh, f] = mscohere(subs_cd2(pairs(pr,1), :, sujeto),
 subs_cd2(pairs(pr,2) ,:, sujeto), hanning(200), 25, 200, EEG.srate);
% % %     cxy(pairs(pr,1), pairs(pr,2)) = mean(coh(f>=EEG.freqwin(3) &
 f<=EEG.freqwin(4)));
% % % end
% % % cxy = cxy + cxy';
% % %
% % % figure
% % % imagesc(cxy);
% % % axis square;    colorbar;  title('Coherence')
```

Ordinal Synchronization (A new information-based correlation)

Fast, robust-to noise tool to assess synchronization at higher orders D. It was recently published in Physica A https://www.sciencedirect.com/science/article/abs/pii/S0960077918309081. When computing it, we take the average over M segments of ordinal vectors of dimension D.

$$OS(X,Y) = \langle 2(\frac{IOS_i^r aw - min}{1 - min} - 0.5)\rangle$$

```
% % % sujeto = 1;
% % % pairs = nchoosek(1:91,2);
% % % osn = zeros(91,91);
% % % for pr = 1 : length(pairs)
% % %     osn(pairs(pr,1), pairs(pr,2)) =
 f_op_syn(subs_cd2_al(pairs(pr,1), :, sujeto),
 subs_cd2_al(pairs(pr,2), :, sujeto), 7, 'cons');
% % % end
% % % osn = abs(osn + osn');
% % %
% % % figure;
% % % imagesc(osn);
% % % axis square; colorbar
% % % title('Ordinal Synchronization')
```

Fully and not fully connected graph

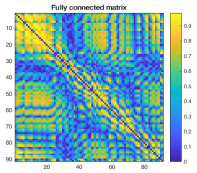
By taking a fully connected graph, one takes into account spurious correlations and we don't want that, so we need to threshold our matrix. Bad news. There is not an exact threshold. Good news! There are many ways to thresholding up a matrix.

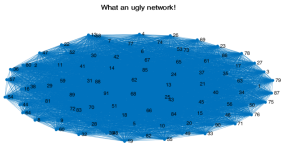
Here we keep the 20% of the weighted links of the Pearson correation matrix, always keeping the network conectedness. Here we make use of functions "`threshold_proportional.m`" and "`isconnected`" to threshold the matrix and testing its connectedness, respectively.

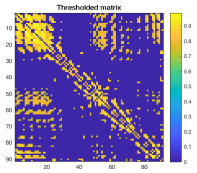
W = threshold_proportional(pea_cd2, 0.2); % function for threshold a matrix

isconnected(W); % function for testing its connectedness

```
figure
subplot(221)
imagesc(pea_cd2); colorbar;  axis square
title('Fully connected matrix')

subplot(222)
plot(graph(pea_cd2))
axis off
title('What an ugly network!')

subplot(223)
W = threshold_proportional(pea_cd2, 0.2);   % look at this! "W" is now
 our matrix with threshold
isconnected(W);                             % if this value is one,
 then "W" is connected
imagesc(W); colorbar;   axis square
title('Thresholded matrix')

subplot(224)
plot(graph(W))
axis off
title('A much better network to work...')
set(gcf, 'units','normalized','outerposition',[0 0 1 1]) %EXPANDING
 FIGURE ON SCREEN
```







Some Networks Features

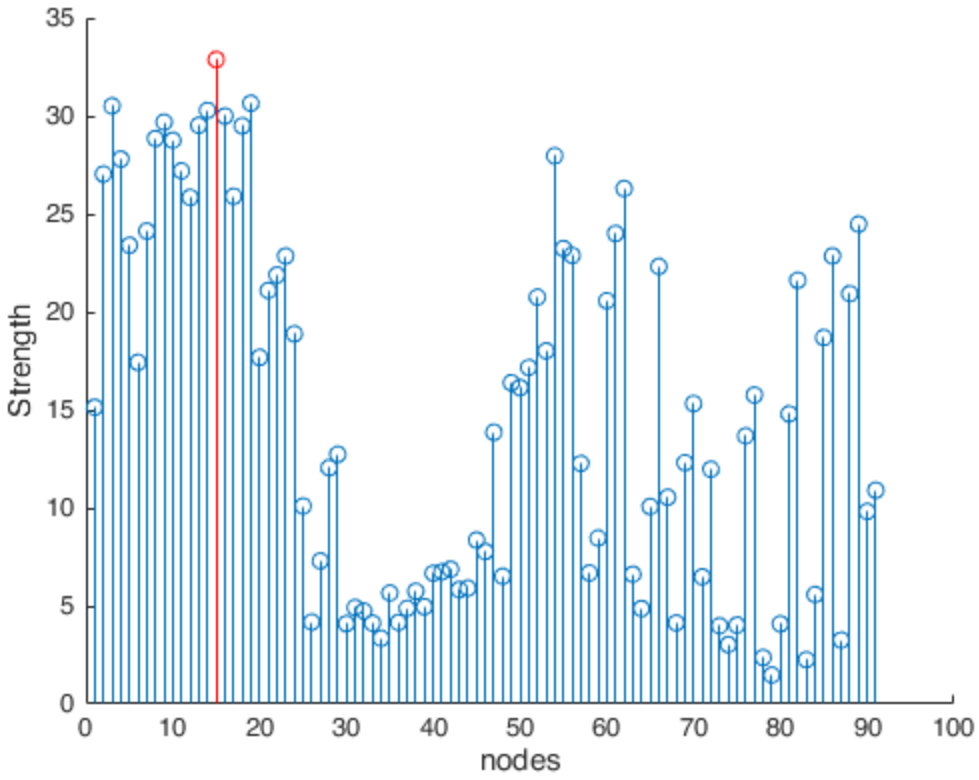
Network features are calculated on the matrix representation of the graph. Here let's consider the previous pearson correlation matrix "_W_" to play with.

Strength is one of the first centrality measures. It compute the " *strenght* " of a node by summing up its weights. Considering a conectivity matrix W , its strenght reads for:

$$s_i = \sum_j W_{ij}$$

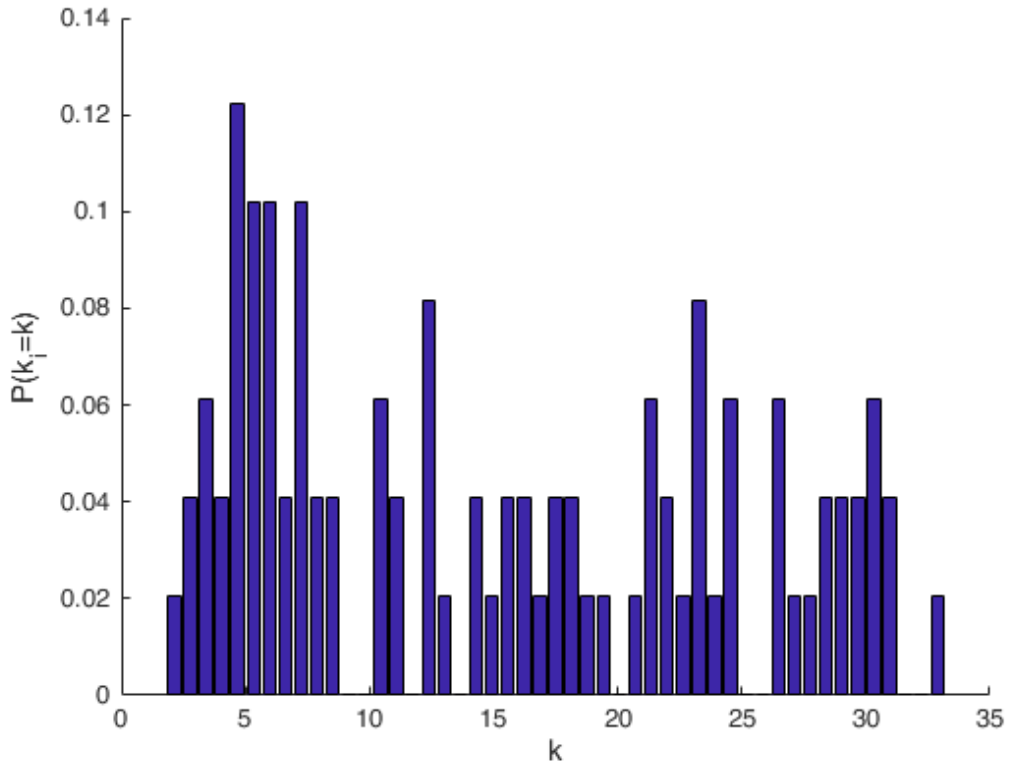
```
st = sum(W,2);
figure;
hold
stem(st)
[st_max, node_st_max] = max(st);
stem(node_st_max, st_max, 'r')
ylabel('Strength'); xlabel('nodes')
```

```
Current plot held
```



We also may obtain another information about network by computing its Probability Density Function (PDF) of finding a specific strenght s_i

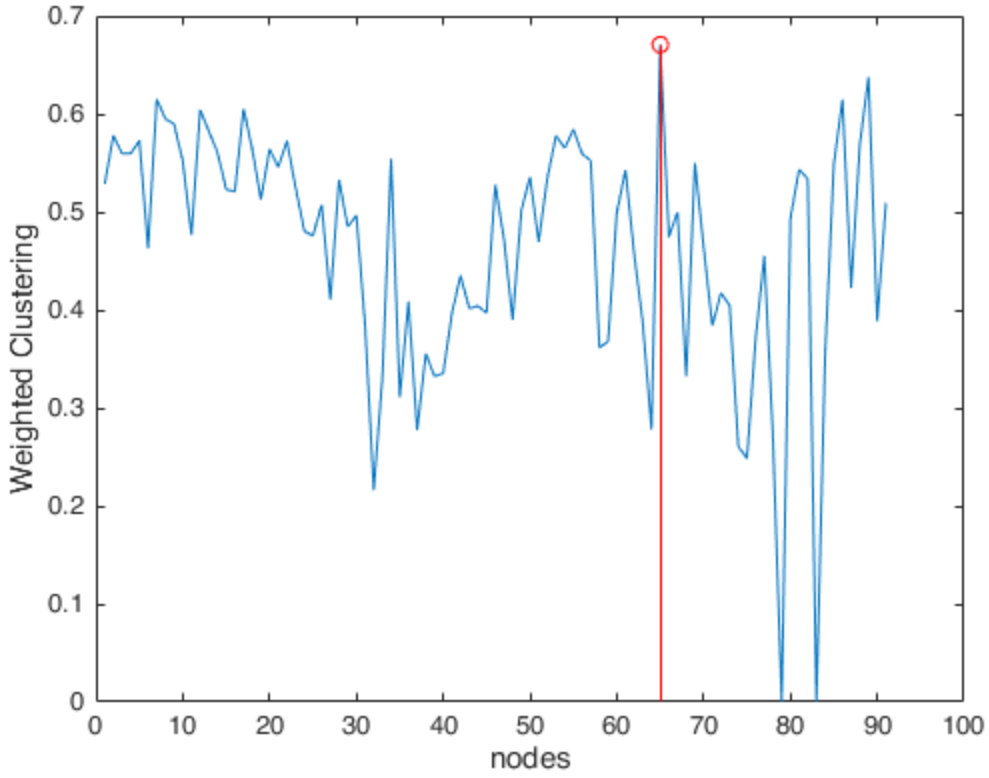
```
[N,binlimit] = histcounts(st, linspace(min(st),max(st),50));   % N is
 the frecuency of nodes along several bin limits
figure
hold;
bar(binlimit(1,2:end), N/numel(N))
ylabel('P(k_i=k)'); xlabel('k')

Current plot held
```



Clustering is another basic measure of node centrality. It takes into account the proportion of nodes's triangles formed for each node respect to the total number of triangles. In other words:

$$c_w = \frac{3 * \#triangles}{\#all\ triplets}$$

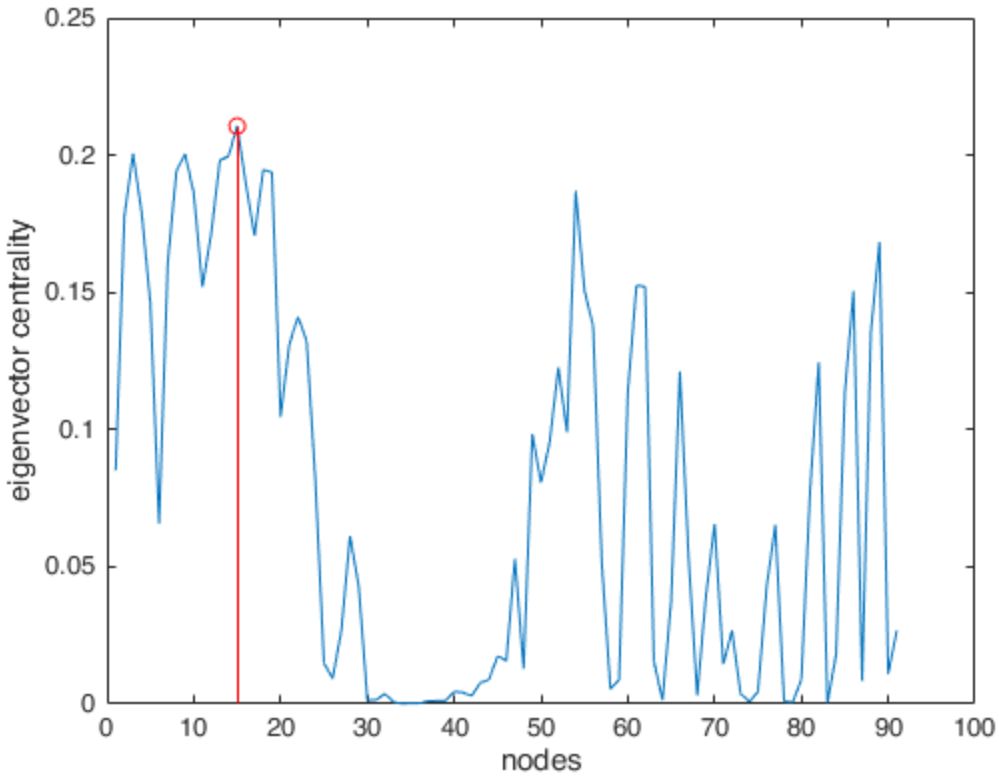
```
cw = clustering_coef_wu(W);

figure
plot(cw)
hold;
[cw_max, node_cw_max] = max(cw);
stem(node_cw_max, cw_max, 'or')
ylabel('Weighted Clustering'); xlabel('nodes')

Current plot held
```



Eigenvector Centrality is a centrality measure that give information about whether a node is well connected or not, understanding "well connected" by that node that is connected with hubs. Mathematically speaking, this vector centrality is the eigenvector associated associated to the first eigenvalue when solving:

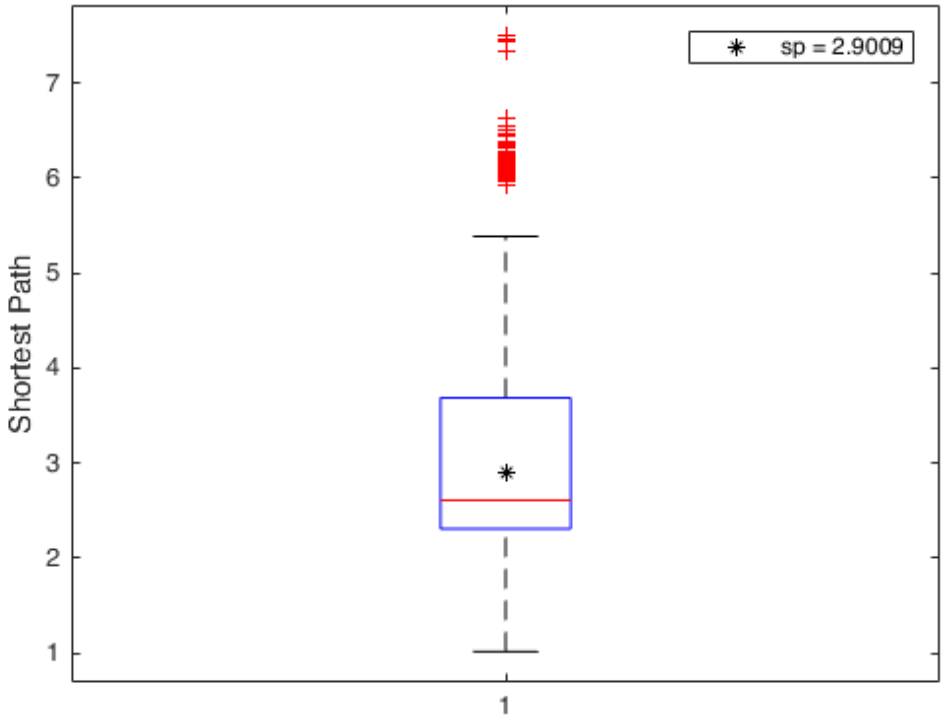
$$Wx = \lambda_1 x$$

```
ec = eigenvector_centrality_und(W);

figure
plot(ec)
hold;
[ec_max, node_ec_max] = max(ec);
stem(node_ec_max, ec_max, 'or')
ylabel('eigenvector centrality');    xlabel('nodes')

Current plot held
```



Average Shortest Path as its name indicates, it's the average shortest path to connect two nodes randomly choosen in the network. In weighted graphs, the weighted the link, the closer two node are. We take the inverse of the weights of matrix *W* to build up the connection-length matrix (*L*). We apply the Dijkstra's algorithm (the most accepted) to find al shortest paths among nodes in a matrix called (*D*). The average shortest path is then the mean value of this resulting matrix.

```
L = weight_conversion(W, 'lengths');    % taking the inverse of matrix
 elements
D = distance_wei(L);                    % applying Dijkstra's
 algorithm
sp = charpath(D);                       % averaging shortest paths'
 matrix

figure
plot(1,sp, '*k')
hold;
boxplot(D(D~=0));
legend(horzcat('sp = ', num2str(sp)))
ylabel('Shortest Path')

Current plot held
```



Rescaling network features for plotting networks

First, you have to define what network feature you want highlight by plotting a functional network (a node parameter, a link feature?). Then you have to rescale that feature respect to numerical figure's parameters of the language you are using to. In this example we will plot the **Strenght** and **link weights**.

Although there are several ways for scaling a collection of numbers, for simplicity here we adopt the classical linear scaling:

$$x' = a + \frac{(x - min_x)(b - a)}{max_x - min_x}$$

Where *a* and *b* are the rescaling limits.

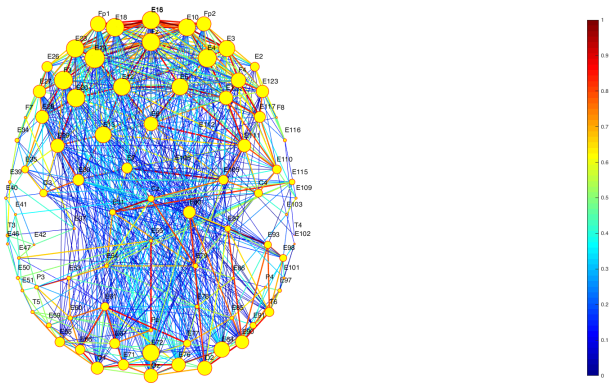
```
grafo = graph(W);                                          %
 converting matrix to graph list of edges
links = table2array(grafo.Edges);                          %
 converting the graph object into a table
wij = links(:,3);                                          % taking
 only link weigths

a = 1;            b = 3;                                   % range
 for rescaling limits
```

```
wij = ( (wij - min(wij)).*(b-a) )/(max(wij) - min(wij));    %
 rescaling formula
wij = wij+0.5;
rgb = squeeze(double2rgb(wij, colormap(jet))); close        % mapping
 values to RGB colors with pallete JET

a = 1;            b = 3;                                    % range
 for rescaling limits
wij = ( (wij - min(wij)).*(b-a) )/(max(wij) - min(wij));    %
 rescaling formula
wij = wij+0.5;
```

Plotting a Functional Network

The main idea is to superimpose different figures in a same canvas. Here we plot links, and then we superimpose nodes' plot.

```
figure
hold on;
for ll = 1 : length(links)  %% plotting links
    line([EEG.chanlocs(links(ll,1)).X,
 EEG.chanlocs(links(ll,2)).X], [EEG.chanlocs(links(ll,1)).Y,
 EEG.chanlocs(links(ll,2)).Y], 'LineWidth', wij(ll), 'Color',
 rgb(ll,:) );
end
colormap(jet);
colorbar
for nn = 1 : 91             %% plotting nodes
    plot(EEG.chanlocs(nn).X, EEG.chanlocs(nn).Y, 'ro', 'MarkerSize',
 st(nn), 'MarkerFaceColor', 'y')
    text(EEG.chanlocs(nn).X+0.5, EEG.chanlocs(nn).Y,
 EEG.chanlocs(nn).labels, 'FontSize', 11);
end
view([270 90]);             %% rotating to vertical view
axis equal; axis off;
set(gcf, 'units','normalized','outerposition',[0 0 1 1]) %EXPANDING
 FIGURE ON SCREEN
```



Acknowledgemetns

I would like to thank to my students and colleagues for comments in the preparation of this tutorial. We used some functions from the following tookits:

Brin Connectivity Toolbox by Rubinov M and Sporns O. https://sites.google.com/site/bctnet/

Matlab Tools for Network Analysis by MIT http://strategic.mit.edu/downloads.php?page=matlab_networks

function "double2rgb.m" by David Legland https://github.com/mattools/matImage/blob/master/matImage/imFilters/double2rgb.m

Dataset were extracted from the paper by Chennu et all PlosOne (2016) https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1004669

Published with MATLAB® R2017a