ANEXO V Detecci?n Din?mica de Comunidades

September 2, 2018

ANEXO V Detección Dinámica de Comunidades
José Pedro Manzano Patrón
Lenguaje: Matlab

Este anexo muestra el código necesario para, a partir de las series temporales de cada región cerebral (ROI_timeseries):

- Generar el enventanado temporal, de tamaño *w* y solapamiento temporal *overlapping*.
- Generar las matrices de conectividad asociada a cada ventana temporal, con la métrica solicitada (Pearson, correlación parcial o *distance correlation*).
- Generar 3 modelos nulos difrentes: un temporal que aleatoriza las series temporales, un modelo nulo nodal que aleatoriza los enlaces entre nodos y un modelo temporal que aleatoriza el orden de las ventanas temporales de las matrices de conectividad.
- Detección de comunidades para cada ventana temporal.
- Aplicación de métodos de consenso sobre las comunidades.
- Principales métricas de la comunidad y del grafo para cada ventana temporal.

El código hace llamadas a subrutinas de forma continua. Por favor, asegúrese que ha descargado e incluido en la ruta todo el código que aparece en el repositorio de Github.

Si tuviese algún problema, no dude en contactar a josepman@ucm.es

In [ ]:
```
clear all
close all

% Selecciona la ruta correcta de su ordenador
ROI_timeseries = csvread('/Users/hose/Desktop/TFM_TECI/Code/ts_con_CSF.csv',1,1);

nNodes = size(ROI_timeseries,2);
w = 20;                         % tamaño ventana
overlapping = 0.8;
l = size(ROI_timeseries,1);    % número de nodos

% A partir de las series temporales de cada nodo, genera las ventanas
% temporales de tamaño w y solapamiento temporal = overlapping
sliding_windows = gen_sliding_window(ROI_timeseries, w, overlapping);

% Create the connectivity matrix
metric_rule = 'corr';  %'corr', 'dist_corr', 'partial_corr', 'cross_corr', 'MI'
connectivity_matrix = gen_connectivity_matrix(sliding_windows, metric_rule);
```

```
% NULL-MODELS
for i=1:size(ROI_timeseries,2);
    tseries_null_model(:,i) = tseries_rand(ROI_timeseries(:,i));
end
tseries_null_model_sw = gen_sliding_window(tseries_null_model, w, overlapping);
tseries_null_model_conn = gen_connectivity_matrix(tseries_null_model_sw, metric_rule);

for i=1:size(connectivity_matrix,3)
     nodal_null_model(:,:,i) = randmio_und_connected(connectivity_matrix(:,:,i), 50);
end

temporal_null_model_conn = temporal_rand(connectivity_matrix);                    % Rando

% Para GN, descomente las siguientes líneas:
% gamma = 2;
% omega = 1;
% conn_mat = cell(1,size(connectivity_matrix,3));
% for i=1:size(connectivity_matrix,3)
%     conn_mat{1,i} = connectivity_matrix(:,:,i);
% end
% [B,twom] = multicat(conn_mat,gamma,omega);                                       % M
%
models = {connectivity_matrix,tseries_null_model_conn,temporal_null_model_conn };

for j=1:length(models);
    connectivity_matrix = models{3};
    % Get the communities
    n = 100;
    final_partition = zeros(nNodes,size(connectivity_matrix,3));
    final_Q = zeros(1,size(connectivity_matrix,3));
    thr_nodal = cell(1,size(connectivity_matrix,3));
    quality = zeros(1,size(connectivity_matrix,3));
    consensus = zeros(nNodes,size(connectivity_matrix,3));
    consensus_simm = zeros(1,size(connectivity_matrix,3));
    agreement_nmi = zeros(1,size(connectivity_matrix,3));
    agreement_z = zeros(1,size(connectivity_matrix,3));
    SWP = zeros(1,size(connectivity_matrix,3));
    delta_c = zeros(1,size(connectivity_matrix,3));
    delta_L = zeros(1,size(connectivity_matrix,3));
    persistence_LMC = cell(1,size(connectivity_matrix,3));

    for i=1:size(connectivity_matrix,3)
        % final partition is created from consensus clustering. See
        % community_detection script
        [final_partition(:,i), Q, thr_nodal{1,i}, quality(i), consensus(:,i), consensus_

        %Similarity between both methods
```

```
        agreement_nmi(i) = normalized_mutual_information(consensus(:,i), final_partition
        agreement_z(i) = zrand(consensus(:,i), final_partition(:,i));

        [SWP(i), delta_C(i), delta_L(i)] = small_world_propensity(connectivity_matrix(:,

        persistence_LMC{1,i} = sig_lmc(final_partition(:,1), connectivity_matrix(:,:,1))
    end

    %%%%
    % Improvement of multilayer partitions of categorical networks
    final_partition = postprocess_categorical_multilayer(final_partition,size(final_part

    % reorders nodes and layers to emphasize persistent structure in an unordered multil
    [final_partition_sorted,s1,s2]=sort_categorical(final_partition);
    pers=categorical_persistence(final_partition_sorted)
    %%%%


    % agreement between windows:
    agreement_w_nmi = zeros(size(connectivity_matrix,3),size(connectivity_matrix,3));
    agreement_w_z = agreement_w_nmi;
    for i=1:size(connectivity_matrix,3)
        for j=1:size(connectivity_matrix,3)
            agreement_w_nmi(i,j) = normalized_mutual_information(final_partition(:,i), f
            agreement_w_z(i,j) = zrand(final_partition(:,i), final_partition(:,j));
        end
    end

    % Coeff of changes of comm. per node
    flexib = flexibility(final_partition', 'cat');

    % Cohesion (if a node change, other also change):
    [Cij,node_cohesion,node_disjoint,node_flexibility,strength_cohesion,commChanges,comm
    % Promiscuity (fraction of all communities in which a node participates at
    % least once)
    P = promiscuity(final_partition');

    % Persistance
    persist = persistence(final_partition');

    % El resto de métricas se pueden extraer con la función bct_measures:
    thr = 0.5;    % para option='binary'
    option = 'weighted';

    G{i} = bct_measures(connectivity_matrix, option, thr);
end
```