

PROYECTO FINAL MODELOS COMPUTACIONALES

Beth Yaravi Perez Torres

```
load GroupAverage_rsfMRI_matrix.mat
Matriz_coactivacion = corr(GroupAverage_rsfMRI);

C(1, 638) = corr(GroupAverage_rsfMRI(:, 1), GroupAverage_rsfMRI(:, 638));
for c1 = 1:638
    for c2 = 1:638
        C(c1, c2) = corr(GroupAverage_rsfMRI(:, c1), GroupAverage_rsfMRI(:, c2));
    end
end

Matrix_coact= C.*~eye(size(C))
```

```
Matrix_coact = 638x638
    0    0.4357    0.5516    0.3348    0.5694    0.0028    0.0842    0.2491 ...
0.4357         0    0.2858    0.1277    0.3763   -0.1453   -0.1215    0.0640
0.5516    0.2858         0    0.6546    0.2361    0.0603    0.2105    0.6010
0.3348    0.1277    0.6546         0    0.0681    0.1638    0.4074    0.7847
0.5694    0.3763    0.2361    0.0681         0   -0.0534   -0.0615    0.0307
0.0028   -0.1453    0.0603    0.1638   -0.0534         0    0.5670    0.1911
0.0842   -0.1215    0.2105    0.4074   -0.0615    0.5670         0    0.4818
0.2491    0.0640    0.6010    0.7847    0.0307    0.1911    0.4818         0
0.1247    0.0525    0.2610    0.2952   -0.0452    0.2757    0.3414    0.3152
0.2505    0.6745    0.1463   -0.0231    0.2187   -0.1414   -0.1252   -0.0296
:
:
```

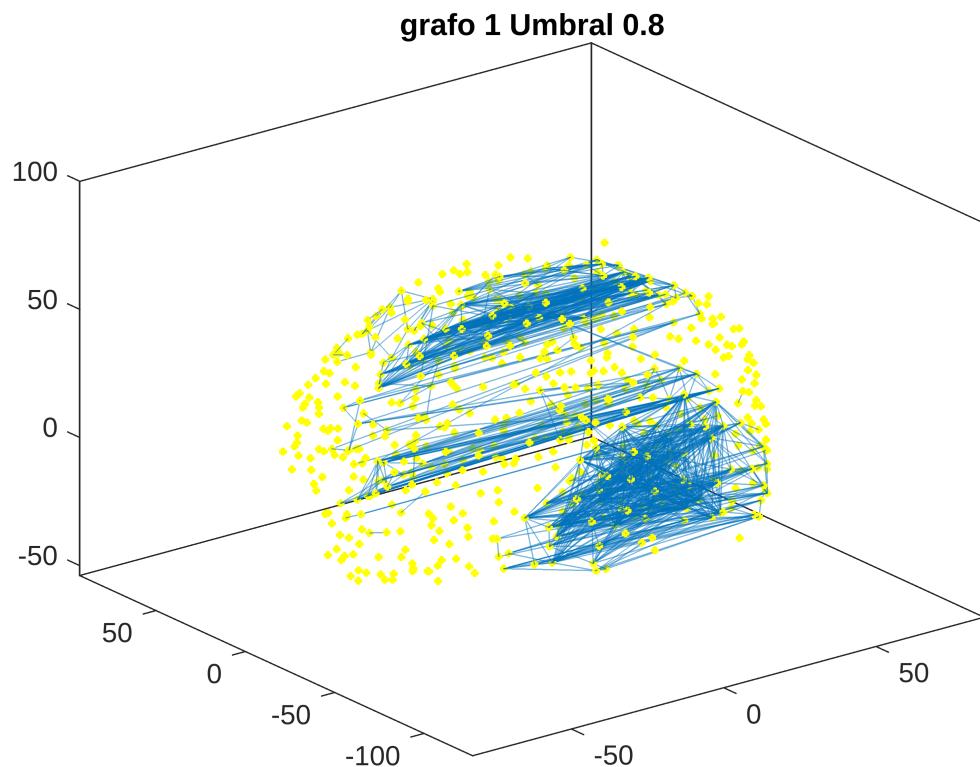
- 1) Definir grafos con la matriz estableciendo umbrales de coactivación de 0.8, 0.9 y 1 y graficar cada grafo. Añadir las coordenadas tridimensionales (incluidas en el archivo.mat).

```
% Definición de umbrales
umbral1 = 0.8;
umbral2 = 0.9;
umbral3 = 1;

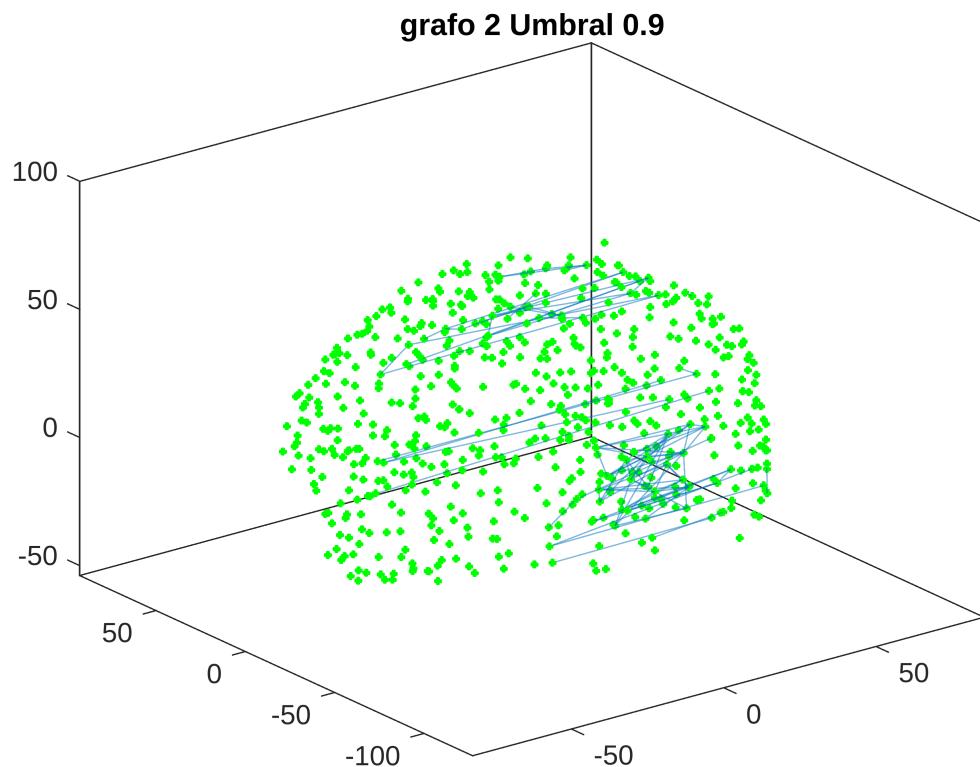
% Grafos con los umbrales respectivos
grafo1 = Matrix_coact >= umbral1;
grafo2 = Matrix_coact >= umbral2;
grafo3 = Matrix_coact >= umbral3;

% Obtener las coordenadas tridimensionales
x = Coord(:, 1);
y = Coord(:, 2);
z = Coord(:, 3);

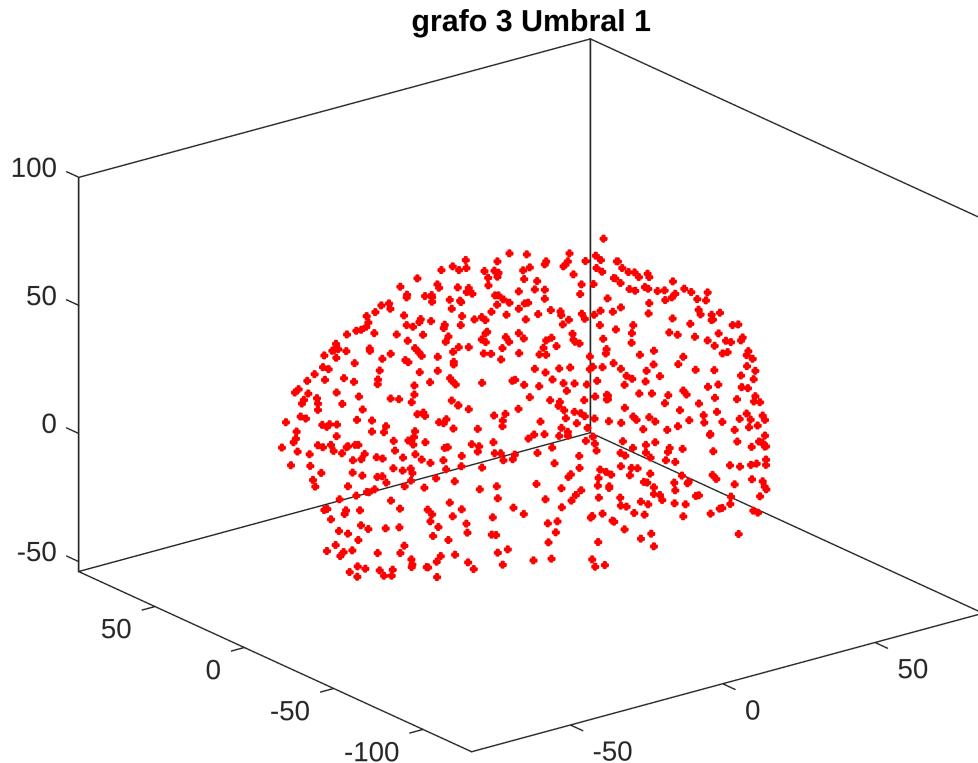
% Plot grafo 1
figure;
Graf1 = graph(grafo1,"omitselfloops");
plot(Graf1, 'XData', x, 'YData', y, 'ZData', z, 'NodeColor', 'yellow');
title('grafo 1 Umbral 0.8');
```



```
% Plot grafo 2
figure;
Grafo2 = graph(grafo2, "omitselfloops");
plot(Grafo2, 'XData', x, 'YData', y, 'ZData', z, 'NodeColor', 'green');
title('grafo 2 Umbral 0.9');
```



```
% Plot grafo 3
figure;
Grafo3 = graph(grafo3, "omitselfloops");
plot(Grafo3, 'XData', x, 'YData', y, 'ZData', z, 'NodeColor', 'red');
title('grafo 3 Umbral 1');
```



- 2) Con uno de los grafos en el punto uno con umbral 0.9, generar una animación donde se haga girar 360° el grafo del cerebro para visualizar las conexiones establecidas.

```
% COORDENADAS Y GRAFO DEL grafol
Grafol = graph(grafol, "omitselfloops");
plot(Grafol, 'XData', x, 'YData', y, 'ZData', z, 'NodeColor', 'yellow');
title('Grafol 3D Umbral 0.8');
num_frames = 100;

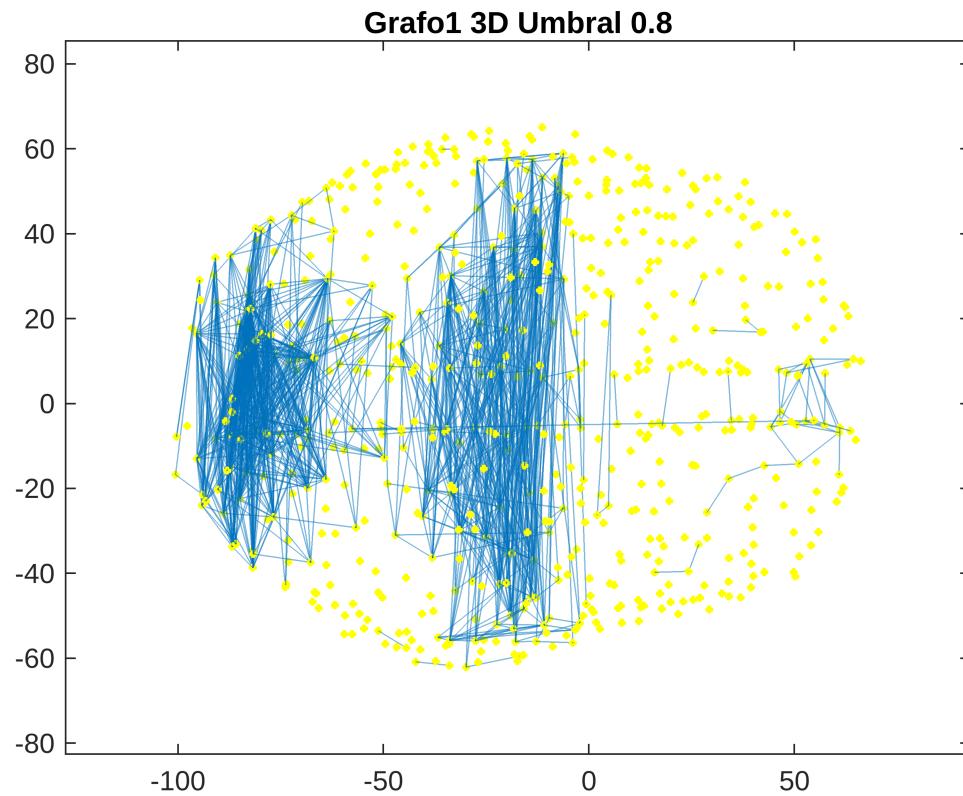
% Inicializar un vector de ángulos para girar 360°
theta = linspace(0, 2*pi, num_frames);
% Ángulo azimutal para la rotación 3D
phi = linspace(0, pi, num_frames);

% Configurar la animación
for frame = 1:num_frames
    % Calcular las nuevas coordenadas de vista para una rotación completa en
    % 3D
    % Utilizando un ángulo azimutal (phi) y un ángulo polar (theta)
    x_view = cos(theta(frame)) * sin(phi(frame));
    y_view = sin(theta(frame)) * sin(phi(frame));
    z_view = cos(phi(frame));
    % Establecer la nueva vista en 3D
    view([x_view, y_view, z_view]);
    % Pausa para que se vea la animación
    pause(0.05);
end
```

```

pause(0.08);
% Capturar el cuadro actual
% frame_data(frame) = getframe(gcf);
end

```

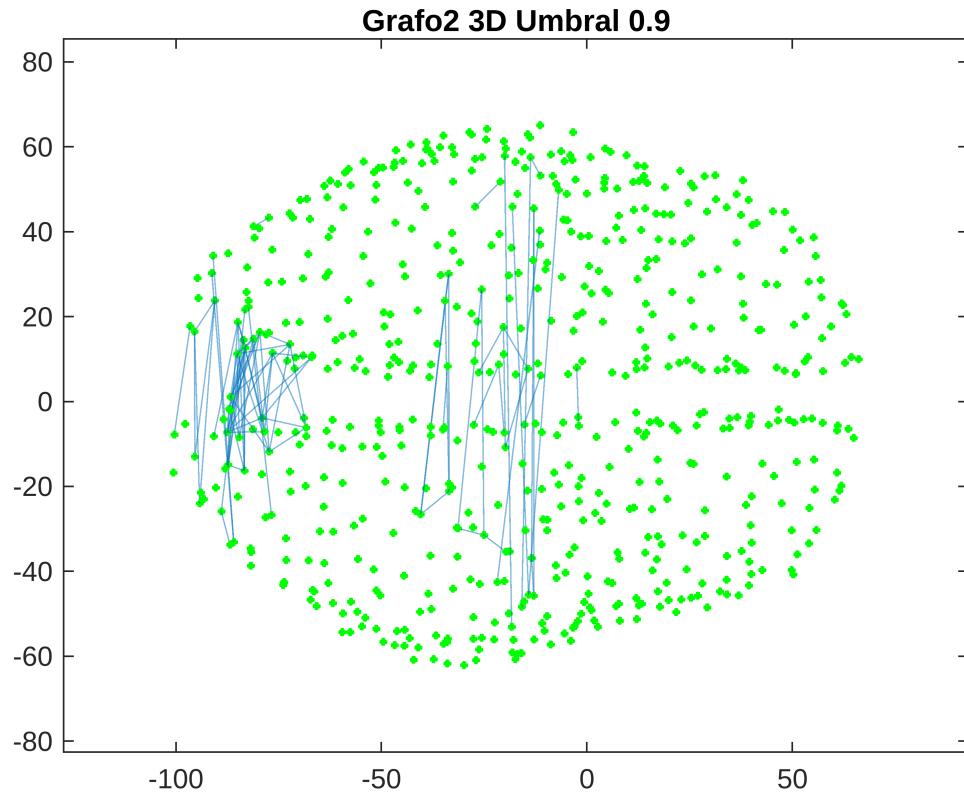


```

% COORDENADAS Y GRAFO DEL grafo2
Grafo2 = graph(grafo2, "omitselfloops");
plot(Grafo2, 'XData', x, 'YData', y, 'ZData', z, 'NodeColor', 'green');
title('Grafo2 3D Umbral 0.9');
num_frames = 100;

theta = linspace(0, 2*pi, num_frames);
phi = linspace(0, pi, num_frames);
for frame = 1:num_frames
    x_view = cos(theta(frame)) * sin(phi(frame));
    y_view = sin(theta(frame)) * sin(phi(frame));
    z_view = cos(phi(frame));
    view([x_view, y_view, z_view]);
    pause(0.08);
end

```



```
% COORDENADAS Y GRAFO DEL grafo
Grafo3 = graph(grafo3, "omitselfloops");
plot(Grafo3, 'XData', x, 'YData', y, 'ZData', z, 'NodeColor', 'red');
title('Grafo3 3D Umbral 1');
num_frames = 100;

theta = linspace(0, 2*pi, num_frames);
phi = linspace(0, pi, num_frames);
for frame = 1:num_frames
    x_view = cos(theta(frame)) * sin(phi(frame));
    y_view = sin(theta(frame)) * sin(phi(frame));
    z_view = cos(phi(frame));
    view([x_view, y_view, z_view]);
    pause(0.08);
end
```

3) Encontrar los hubs del grafo, y establecer el tamaño del nodo proporcional al valor del grado.

```
%Grado umbral .8
matriz_adyacencia_1=(Matrix_coact >=umbral1)
```

```
matriz_adyacencia_1 = 638x638 logical array
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ...
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ...
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ...
```

```

0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
.
:
```

```
Grado_1=sum(matriz_adyacencia_1)
```

```
Grado_1 = 1x638
0   2   1   0   0   1   1   1   0   4   0   0   1 ...
```

```
grafo_gradol=graph(matriz_adyacencia_1, 'upper', 'OmitSelfLoops');
%Graph, marker size=grado, no acepta 0. Grado+1.
```

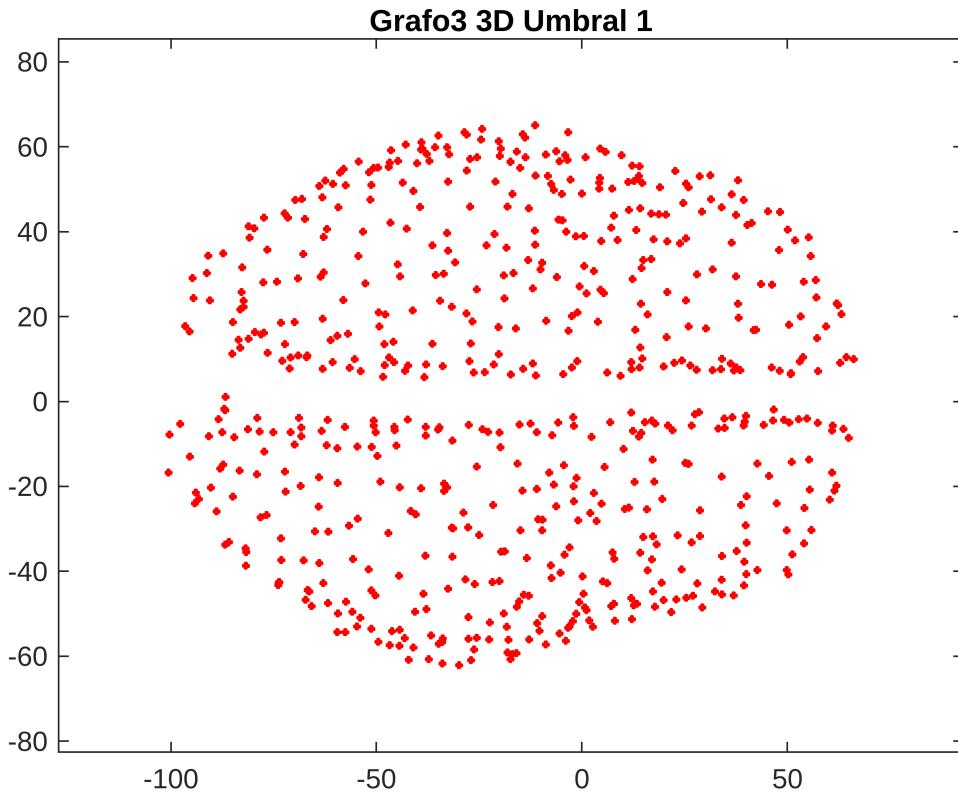
```
%Grado umbral .9
matriz_adyacencia_2=(Matrix_coact >= umbral2);
Grado_2=sum(matriz_adyacencia_2)
```

```
Grado_2 = 1x638
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 ...
```

```
%Grado umbral 1
matriz_adyacencia_3 = (Matrix_coact >=umbral3);
Grado_3=sum(matriz_adyacencia_3)
```

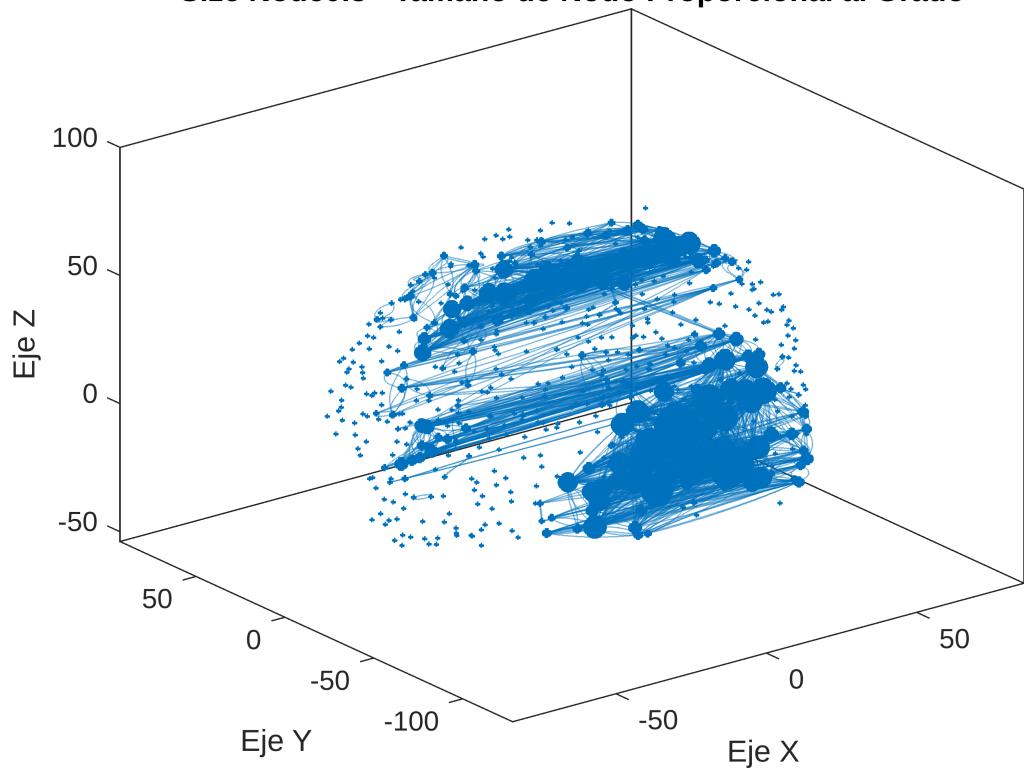
```
Grado_3 = 1x638
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 ...
```

```
%Hubs/tamaño/conexión del nodo .8
for i_1 = 1:length(umbral1)
    % Crear objeto de grafo dirigido desde la matriz de adyacencia
    grafo_1 = digraph(matriz_adyacencia_1,"omitselfloops");
    % Normalizar los grados para que se ajusten al tamaño del nodo
    tamano_nodo_1 = (10 * Grado_1 / max(Grado_1));
    % Crear subgráfica para el umbral actual
    subplot ( 1, length(umbral1), i_1) ;
    % Graficar el grafo con tamaños de nodo proporcionales a los grados
    plot(grafo_1, 'XData', Coord(:,1), 'YData', Coord(:,2), 'ZData',
    Coord(:,3), 'NodeLabel', {}, 'MarkerSize', tamano_nodo_1 +1 );
    % Configurar títulos y etiquetas
    title(['Size Node', num2str(umbral1(i_1)), ' - Tamano de Nodo Proporcional
    al Grado']);
    xlabel('Eje X');
    ylabel('Eje Y');
    zlabel('Eje Z');
end
```

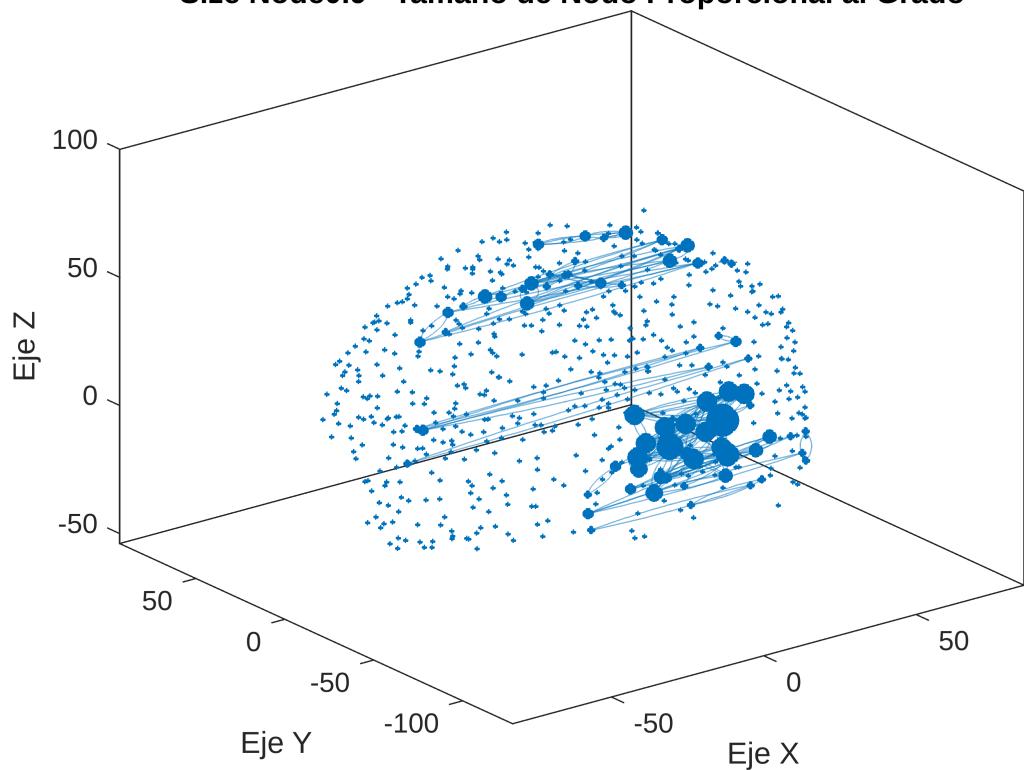


```
%Hubs/tamaño/conexión del nodo .9
for i_2 = 1:length(umbral2)
    grafo_2 = digraph(matriz_adyacencia_2,"omitselfloops");
    tamano_nodo_2 = (10 * Grado_2 / max(Grado_2));
    subplot ( 1, length(umbral2), i_2) ;
    plot(grafo_2, 'XData', Coord(:,1), 'YData', Coord(:,2), 'ZData',
    Coord(:,3), 'NodeLabel', {}, 'MarkerSize', tamano_nodo_2 +1 );
    title(['Size Node', num2str(umbral2(i_2)), ' - Tamaño de Nodo Proporcional
    al Grado']);
    xlabel('Eje X');
    ylabel('Eje Y');
    zlabel('Eje Z');
end
```

Size Node0.8 - Tamaño de Nodo Proporcional al Grado



Size Node0.9 - Tamaño de Nodo Proporcional al Grado



```
%Hubs/tamaño/conexión del nodo 1 (NO HAY CONEXIONES)
%for i_3 = 1:length(umbral3)
%  grafo_3 = digraph(matriz_adyacencia_3,"omitselfloops");
%  tamano_nodo_3 = (10 * Grado_3 / max(Grado_3));
%  subplot ( 1, length(umbral3), i_3) ;
%  plot(grafo_3, 'XData', Coord(:,1), 'YData', Coord(:,2), 'ZData',
Coord(:,3), 'NodeLabel', {}, 'MarkerSize', tamano_nodo_3 +1 );
%  title(['Size Node', num2str(umbral3(i_3)), ' - Tamaño de Nodo
Proporcional al Grado']);
%  xlabel('Eje X');
%  ylabel('Eje Y');
%  zlabel('Eje Z');
%end
```

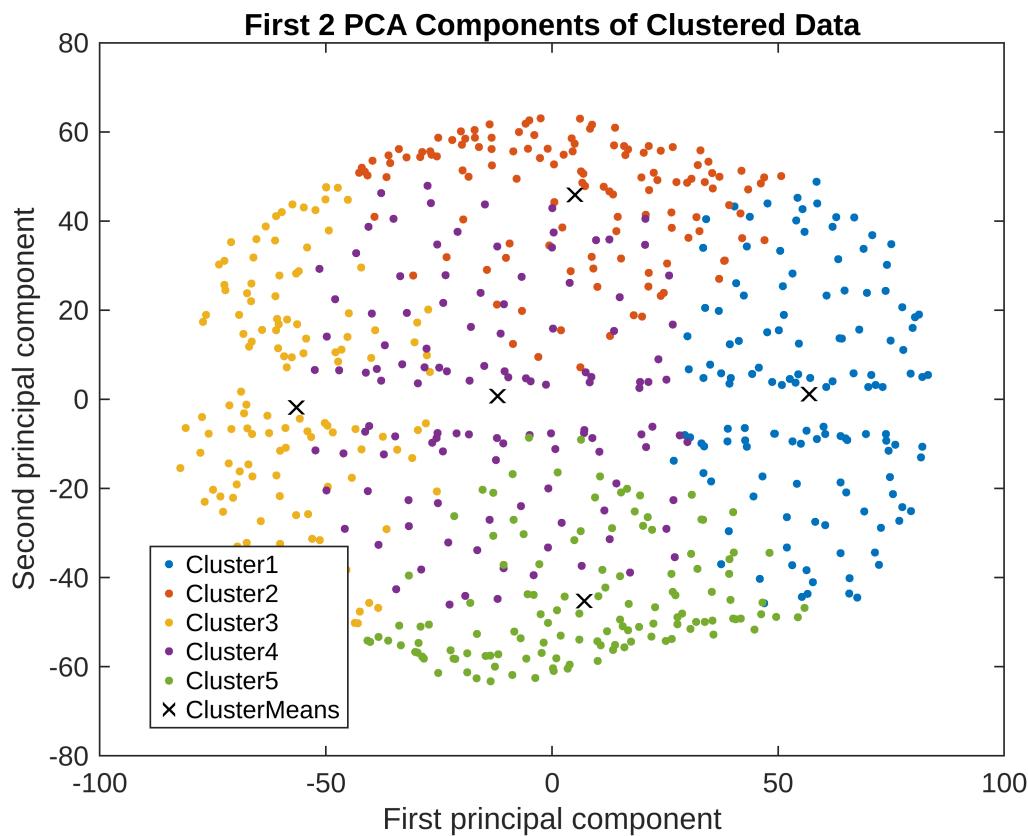
- 4) En función de la matriz de emparejamiento (correlación de la matriz de adyacencia), establecer una partición de los nodos en módulos. Escoger el número de módulos que creas conveniente y justificar por qué escogiste ese número.

```
% Modularidad con umbral 0.8, .9 y 1.
grafosizedir_1 = digraph(matriz_adyacencia_1);
grafosizedir_2 = digraph(matriz_adyacencia_2);
grafosizedir_3 = digraph(matriz_adyacencia_3);

K4 = 5;
[clusterIndices,centroids2] = kmeans(Coord,K4);
clear K4

% SCATTER PLOT PCA
figure
[~,score2] = pca(Coord);
clusterMeans2 = grpstats(score2,clusterIndices,"mean");
h = gscatter(score2(:,1),score2(:,2),clusterIndices);
for i = 1:numel(h)
    h(i).DisplayName = strcat("Cluster",h(i).DisplayName);
end

clear h i score2
hold on
h = scatter(clusterMeans2(:,1),clusterMeans2(:,2),50,"kx","LineWidth",2);
hold off
h.DisplayName = "ClusterMeans";
clear h clusterMeans2
legend;
title("First 2 PCA Components of Clustered Data");
xlabel("First principal component");
ylabel("Second principal component");
```



JUSTIFICACION PARA EL NUMERO DE NODOS

El número de módulos utilizados para dividir a los nodos de una matriz de adyacencia se hace con el objetivo de identificar los grupos que están más densamente conectados entre sí que con el resto de la red. Una base para la selección de clústers es segun los lóbulos del cerebro, parietal, temporal, occipital, frontal, ínsula y límbico. En el gráfico se muestran cinco clusters: amarillo para frontal, morado y azul para temporales, verde para occitales y naranja para parietal.

5) Determinar el conjunto del Rich Club y discutir las implicaciones anatómicas y funcionales de este grupo de nodos (mínimo 100 palabras).

```
%Nodos Umbral .8 bidireccionales utilizando una función del brain
%connectivity toolbox
Nodos_richclub= rich_club_bu (matriz_adyacencia_1);

% Encontrar los nodos del Rich Club
umbral_richclub = 1;
nodos_richclub = find(Nodos_richclub >= umbral_richclub);

% Visualizar el conjunto del Rich Club
disp('Nodos en el Rich Club:' );
```

Nodos en el Rich Club:

```
disp(nodos_richclub);
```

```
% Crear un grafo no dirigido desde la matriz de adyacencia
grafo = graph(matriz_adyacencia_1, 'upper', 'OmitSelfLoops');

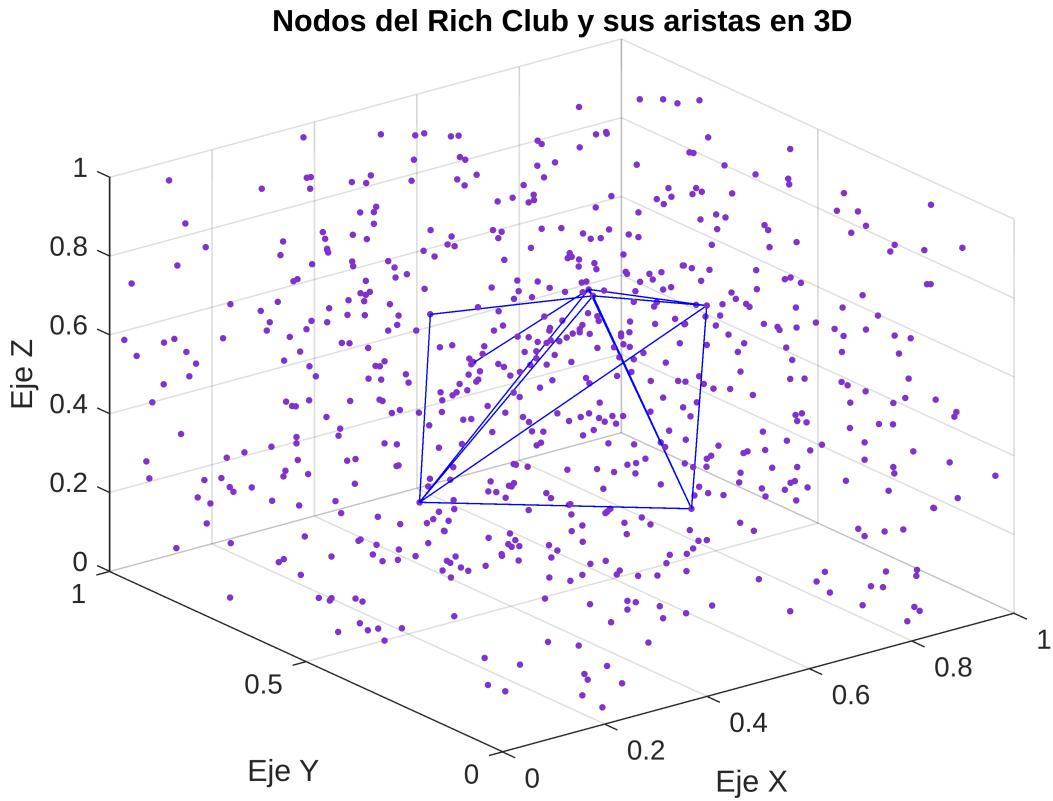
% Crear un subgrafo con los nodos del Rich Club y sus aristas
subgrafo_richclub = subgraph(grafo, nodos_richclub);

% Obtener las coordenadas tridimensionales
coordenadas_3d = rand(638, 3);

% Visualizar el subgrafo del Rich Club en un gráfico 3D
figure;
plot3(coordenadas_3d(:, 1), coordenadas_3d(:, 2), coordenadas_3d(:, 3), 'r.',
'MarkerSize', 8, 'Color', '[ .5 .2 .8]');
hold on;

% Visualizar las aristas del subgrafo del Rich Club
aristas_richclub = subgrafo_richclub.Edges.EndNodes;
for i = 1:size(aristas_richclub, 1)
    nodos_conexion = aristas_richclub(i, :);
    plot3(coordenadas_3d(nodos_conexion, 1), coordenadas_3d(nodos_conexion, 2),
    coordenadas_3d(nodos_conexion, 3), 'b-');
end

% Configurar la visualización
title('Nodos del Rich Club y sus aristas en 3D');
xlabel('Eje X');
ylabel('Eje Y');
zlabel('Eje Z');
grid on;
hold off;
```



IMPLICACIONES ANATOMICAS Y FUNCIONALES DE ESTE GRUPO DE NODOS

Tienen una mayor densidad de conexión, anatómica y funcionalmente hablando, por tanto, son áreas que procesan mucha información de diversas áreas y actúan como punto de distribución. Sin ellos, probablemente habría una mayor conexión entre todas las áreas, más materia blanca e implicaría un gasto de recursos y espacio considerable, haciendo menos eficiente al sistema.

7) Supongamos que eliminamos los nodos del RichClub, describir cómo cambian las propiedades topológicas del grafo, hacer comparativas del grado, coeficiente de cluster, coeficiente de mundo pequeño y las medidas de centralidad (cercanía, intermediación).

```
% Crear un grafo no dirigido desde la matriz de adyacencia
grafonrc_1 = graph(matriz_adyacencia_1, 'upper', 'OmitSelfLoops');
% Calcular la centralidad de grado
centralidad_grado = centrality(grafonrc_1, 'degree');

% Eliminar los nodos del Rich Club de la matriz de adyacencia
matrizadyacencia_sinrichclub = matriz_adyacencia_1;
matrizadyacencia_sinrichclub(nodos_richclub, :) = 0;
matrizadyacencia_sinrichclub(:, nodos_richclub) = 0;
% Visualizar la nueva matriz de adyacencia sin los nodos del Rich Club
disp('Matriz de adyacencia sin nodos del Rich Club:' );
```

Matriz de adyacencia sin nodos del Rich Club:

```

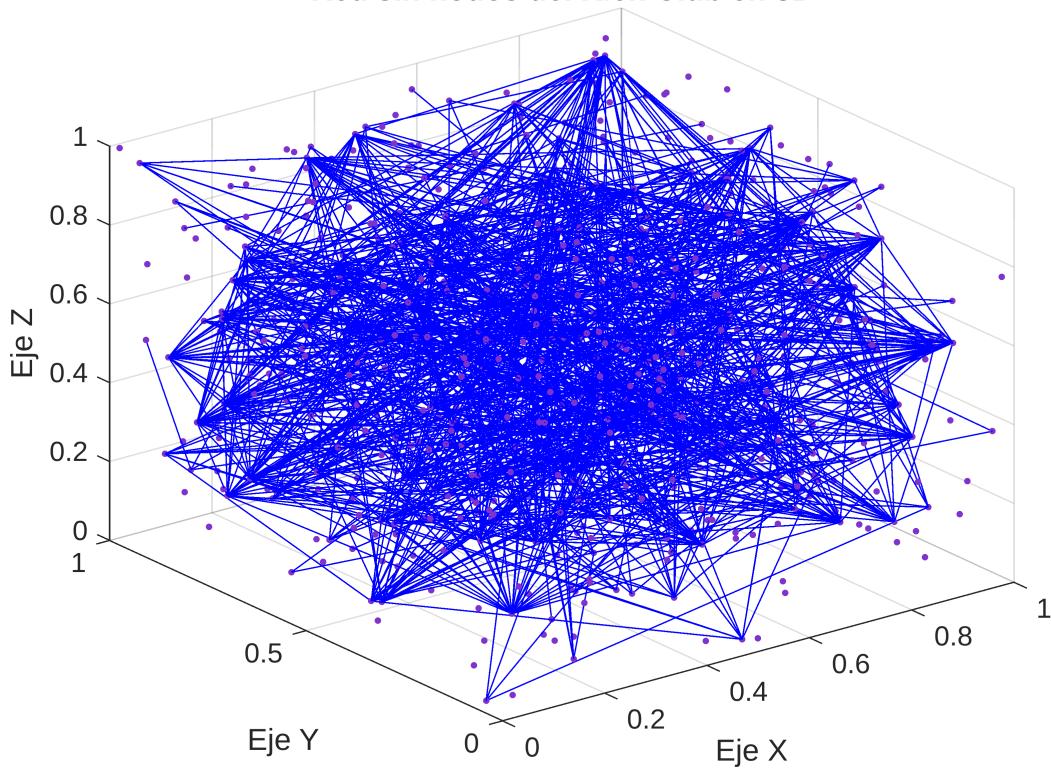
% Crear un grafo no dirigido desde la matriz sin Rich Club
grafo_sinrichclub = graph(matrizadyacencia_sinrichclub,
'upper','OmitSelfLoops');
coordenadas_3d = rand(638,3);
% Visualizar el grafo en un gráfico 3D
figure;
plot3(coordenadas_3d(:, 1), coordenadas_3d(:, 2), coordenadas_3d(:, 3), '.', ...
'MarkerSize', 8, 'Color', '[.5 .2 .8]');
hold on;

% Visualizar las aristas del grafo sin Rich Club
aristas_sinrichclub = grafo_sinrichclub.Edges.EndNodes;
for i = 1:size(aristas_sinrichclub, 1)
nodos_conexion = aristas_sinrichclub(i, :);
plot3(coordenadas_3d(nodos_conexion, 1), coordenadas_3d(nodos_conexion, 2),
coordenadas_3d(nodos_conexion, 3), 'b-');
end

% Configurar la visualización
title('Red sin nodos del Rich Club en 3D');
xlabel('Eje X');
ylabel('Eje Y');
zlabel('Eje Z');
grid on;
hold off;

```

Red sin nodos del Rich Club en 3D



8) Quitar 10%-50% de los nodos con mayor medida de intermediación y describir cómo cambian las propiedades topológicas del grafo, hacer comparativas del grado, coeficiente de cluster, coeficiente de mundo pequeño y las medidas de centralidad (cercanía, intermediación).

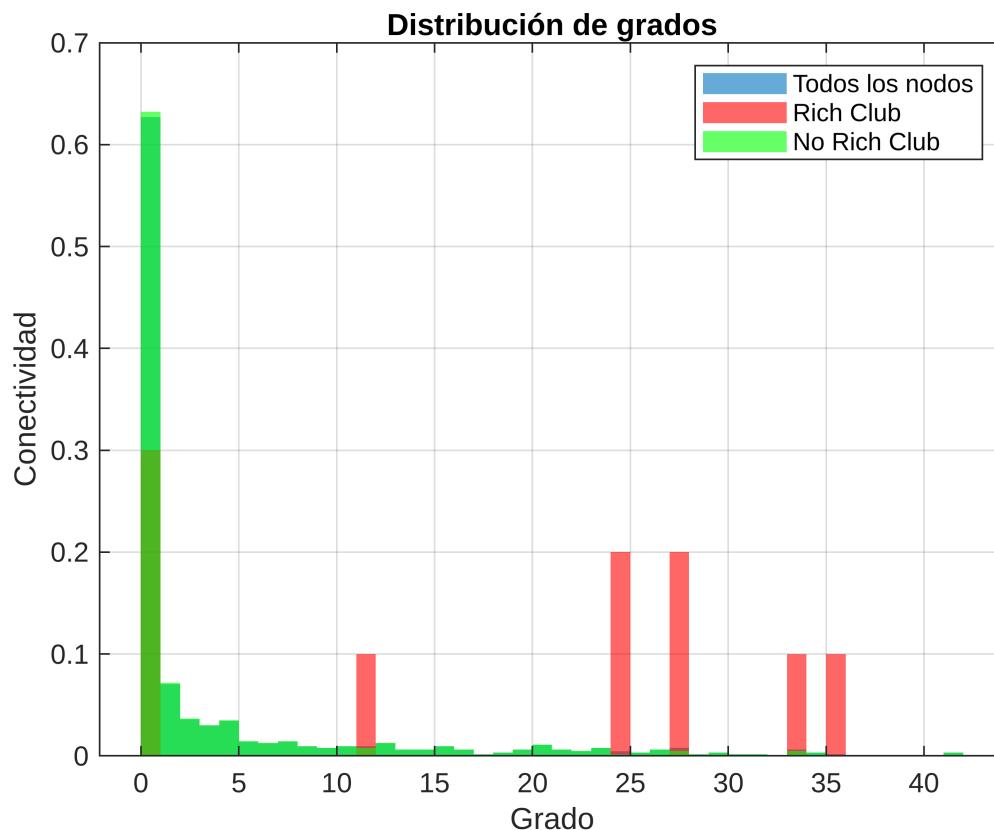
```
% Calcular los grados de todos los nodos
grados_totales = degree(grafo_grado1);

% Calcular los grados de los nodos del Rich Club
grados_richclub = degree(grafo, nodos_richclub);

% Obtener los nodos que no son del Rich Club
nodos_norichclub = setdiff(1:grafo.numnodes, nodos_richclub);
% Calcular los grados de los nodos que no son del Rich Club
grados_norichclub = degree(grafo, nodos_norichclub);

% Visualizar las distribuciones de grado
figure;
histogram(grados_totales, 'BinEdges', 0:1:max(grados_totales)+1,
'Normalization', 'probability', 'EdgeColor', 'none', 'FaceColor', 'auto',
'DisplayName', 'Todos los nodos');
hold on;
histogram(grados_richclub, 'BinEdges', 0:1:max(grados_totales)+1,
'Normalization', 'probability', 'EdgeColor', 'none', 'FaceColor', 'r',
'DisplayName', 'Rich Club');
histogram(grados_norichclub, 'BinEdges', 0:1:max(grados_totales)+1,
'Normalization', 'probability', 'EdgeColor', 'none', 'FaceColor', 'g',
'DisplayName', 'No Rich Club');
hold off;

% Configuración de la visualización
title('Distribución de grados');
xlabel('Grado');
ylabel('Conejividad');
legend('show');
grid on;
```

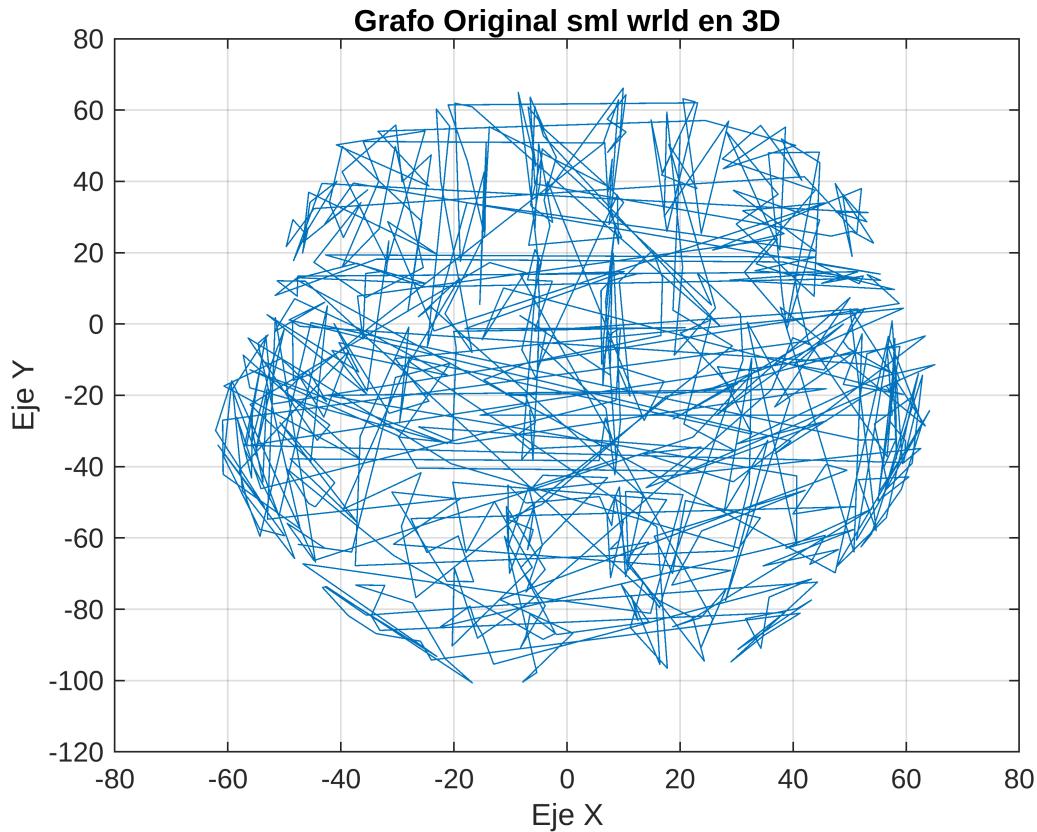


```
%coeficiente_mundo_pequeno_inicial = smallworld(grafosize_1);
CoefMunPe=clustering_coef_bu (Coord)
```

```
CoefMunPe = 638x1
31.3998
31.3998
31.3998
31.3998
31.3998
31.3998
31.3998
31.3998
31.3998
31.3998
31.3998
31.3998
31.3998
31.3998
31.3998
31.3998
31.3998
31.3998
31.3998
31.3998
:
:
```

```
%coeficiente de cluster --> cercanía e intermediación
figure;
plot(CoefMunPe, 'XData', x, 'YData', y, 'ZData', z);
title('Small World .8');
hold on;
title('Grafo Original sml wrld en 3D');
xlabel('Eje X');
ylabel('Eje Y');
zlabel('Eje Z');
```

```
grid on;
```



COMPARATIVA A PARTIR DEL UMBAL 0.8

Eliminar los nodos del RichClub, aquellos con un alto grado de conectividad, tiene un impacto significativo en las propiedades topológicas de la red del grafo.

- El grado es el número de conexiones que tiene un nodo con otros nodos, por tanto, si eliminamos los nodos del RichClub el grado promedio de la red disminuirá notablemente. Los nodos restantes (que tienen un grado más bajo) se conectarán de manera más dispersa, lo que puede resultar en una disminución de la conectividad general de la red o, incluso, en la generación de componentes aislados.
- El coeficiente de cluster mide la tendencia de los nodos a formar grupos de vecinos interconectados entre sí. Si eliminamos a los nodos del RichClub disminuimos el coeficiente de cluster, especialmente si los nodos de alto grado eran los responsables de interconectar a varios nodos en la red y, por tanto, hay una menor interconexión.
- El coeficiente de mundo pequeño se refiere a la propiedad de las redes donde, a pesar de ser grandes, la distancia entre nodos es pequeña, lo que significa que pueden alcanzarse con pocos pasos. Al eliminar los hubs de alto grado se interrumpirán las conexiones globales, aumentando así la distancia entre los nodos restantes y bajando su eficacia.

9) Generar un modelo nulo aleatorio donde se tenga el mismo número de nodos y el mismo número total de conexiones, y comparar sus propiedades con el grafo original del cerebro.

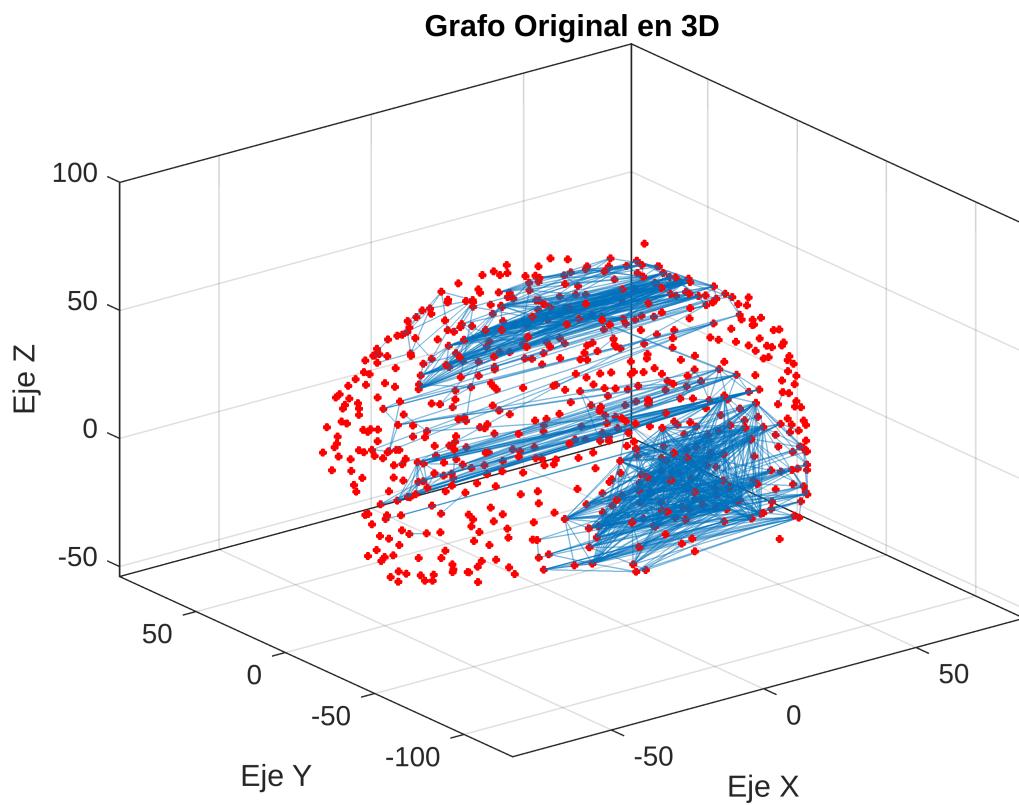
```
% Crear un modelo nulo mediante reconexión aleatoria  
num_reconexiones = 638;
```

```

grafo_modelonulo = graph(randmio_und(matriz_adyacencia_1, num_reconexiones));
coordenadas_3d = rand(638, 3);

% Visualizar el grafo original en un gráfico 3D
figure;
plot(Grafol, 'XData', x, 'YData', y, 'ZData', z, 'NodeColor', 'red');
title('Umbral 0.8');
hold on;
title('Grafo Original en 3D');
xlabel('Eje X');
ylabel('Eje Y');
zlabel('Eje Z');
grid on;

```



```

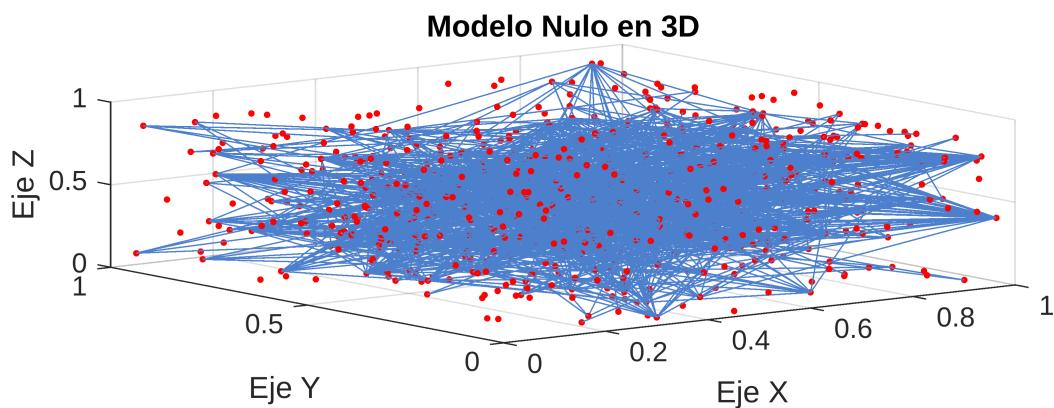
% Visualizar el modelo nulo en un gráfico 3D
subplot(2, 1, 2);
plot3(coordenadas_3d(:, 1), coordenadas_3d(:, 2), coordenadas_3d(:, 3), '.', ...
'MarkerSize', 8, 'Color', 'r');
hold on;
aristas_original = grafo_gradol.Edges.EndNodes;
aristas_modelonulo = (grafo_modelonulo.Edges.EndNodes);
for i = 1:size(aristas_modelonulo, 1)
nodos_conexion = aristas_modelonulo(i, :);
plot3(coordenadas_3d(nodos_conexion, 1), coordenadas_3d(nodos_conexion, 2), ...
coordenadas_3d(nodos_conexion, 3), 'Color', '[.3 .5 .8]');

```

```

end
title('Modelo Nulo en 3D');
xlabel('Eje X');
ylabel('Eje Y');
zlabel('Eje Z');
grid on;

```



COMPARACIÓN DE PROPIEDADES

Se puede apreciar una diferencia muy grande en la forma, para empezar, el modelo nulo es más pequeño, no está claramente conectado, y se vé mas caos en cuanto a las conexiones, mientras que en el cerebro se observa una línea de conexión mayor.

10) Generar un modelo nulo aleatorio donde se conserve la distribución de grado y comparar sus propiedades con el grafo original del cerebro.

```

% Definir el número de nodos y grafo original
num_nodos = size(matriz_adyacencia_1, 1);
grafo_modelonulo2 = graph(randmio_und(matriz_adyacencia_1, num_nodos));
coordenadas_3d = rand(638, 3);

% Visualizar el grafo original en 3D
figure;
subplot(2, 1, 1);
plot(Grafo1, 'XData', coordenadas_3d(:, 1), 'YData', coordenadas_3d(:, 2),
    'ZData', coordenadas_3d(:, 3), 'NodeColor', 'red');
title('Grafo Original en 3D');

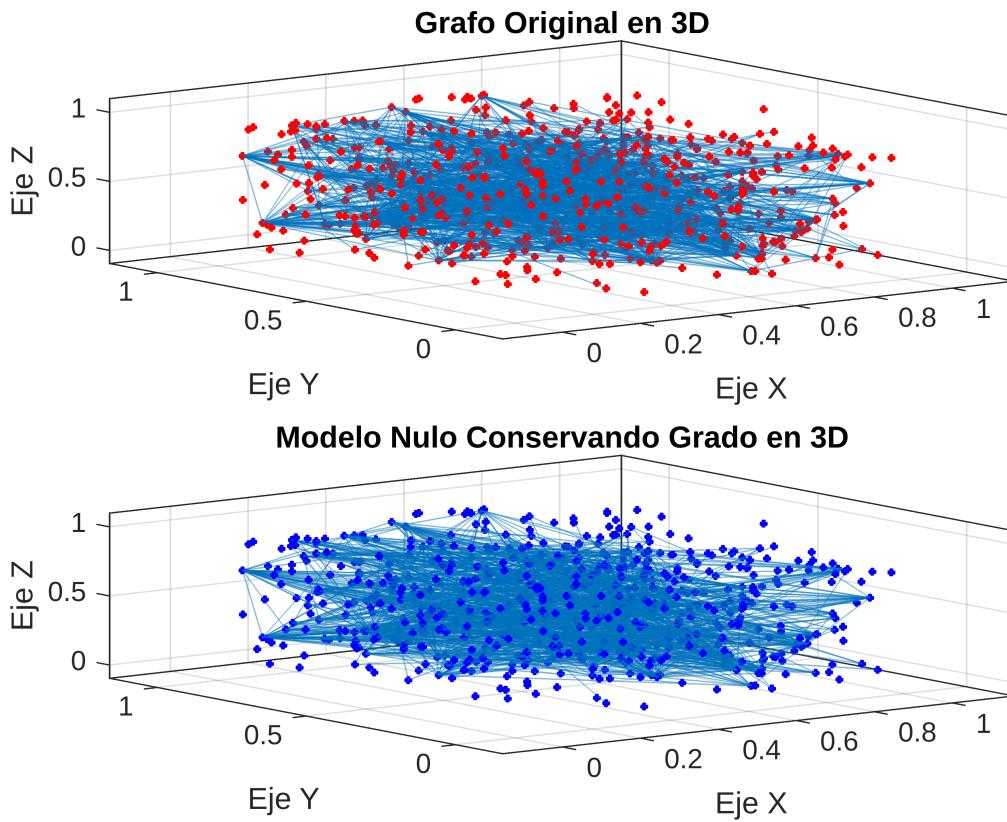
```

```

xlabel('Eje X');
ylabel('Eje Y');
zlabel('Eje Z');
grid on;

% Visualizar el modelo nulo en 3D
subplot(2, 1, 2);
plot(grafo_modelonulo2, 'XData', coordenadas_3d(:, 1), 'YData',
coordenadas_3d(:, 2), 'ZData', coordenadas_3d(:, 3), 'NodeColor', 'blue');
title('Modelo Nulo Conservando Grado en 3D');
xlabel('Eje X');
ylabel('Eje Y');
zlabel('Eje Z');
grid on;

```



```

% Comparación de propiedades: Distribución de grados
grados_original = degree(Grafol);
grados_nulo = degree(grafo_modelonulo2);

% Graficar la distribución de grados de ambos grafos
figure;
subplot(1, 2, 1);
histogram(grados_original, 'FaceColor', 'r');
title('Distribución de Grados - Grafo Original');
xlabel('Grado');

```

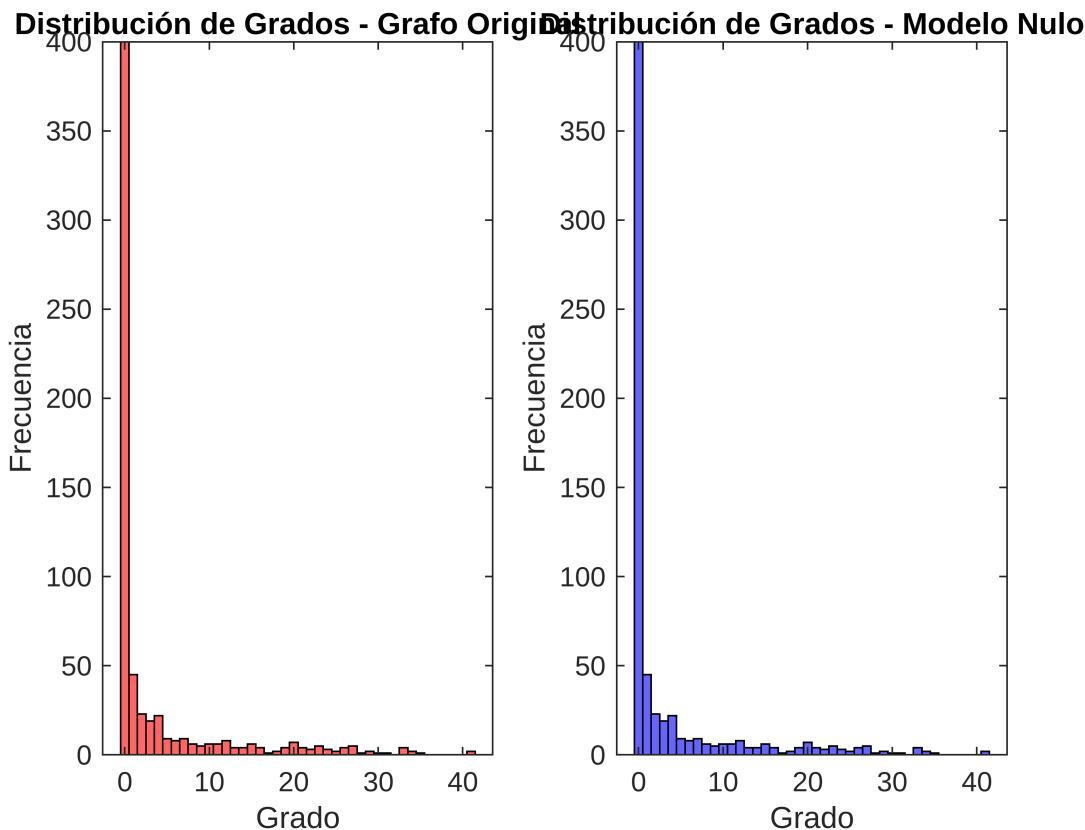
```

ylabel('Frecuencia');

subplot(1, 2, 1);
histogram(grados_original, 'FaceColor', 'r');
title('Distribución de Grados - Grafo Original');
xlabel('Grado');
ylabel('Frecuencia');

subplot(1, 2, 2);
histogram(grados_nulo, 'FaceColor', 'b');
title('Distribución de Grados - Modelo Nulo');
xlabel('Grado');
ylabel('Frecuencia');

```



```

% Calcular y comparar propiedades adicionales
disp(['Promedio de grados grafo original: ',
num2str(mean(grados_original))]);

```

Promedio de grados grafo original: 3.6489

```

disp(['Promedio de grados modelo nulo: ', num2str(mean(grados_nulo))]);

```

Promedio de grados modelo nulo: 3.6489

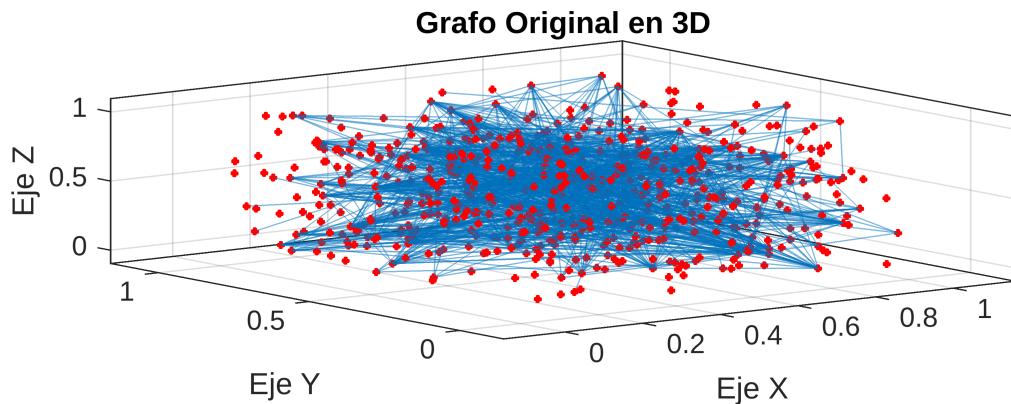
- 11) Generar un modelo nulo utilizando una probabilidad de conexión en función de la distancia geométrica, con el mismo número de nodos y conexiones y compara sus propiedades y discutir la importancia de las conexiones a larga distancia en el cerebro.

```

% Definir el número de nodos y grafo original
num_nodos = size(matriz_adyacencia_1, 1);
coordenadas_3d = rand(num_nodos, 3);
grafo_original = graph(matriz_adyacencia_1);

```

```
% Visualizar el grafo original en 3D
figure;
subplot(2, 1, 1);
plot(grafo_original, 'XData', coordenadas_3d(:, 1), 'YData',
coordenadas_3d(:, 2), 'ZData', coordenadas_3d(:, 3), 'NodeColor', 'red');
title('Grafo Original en 3D');
xlabel('Eje X');
ylabel('Eje Y');
zlabel('Eje Z');
grid on;
```



```
% Crear la matriz de distancias entre nodos (distancias euclidianas)
distancias = squareform(pdist(coordenadas_3d)); % Matriz de distancias
euclidianas entre nodos
umbral_distancia = 0.3; % Umbral de distancia para conexiones más probables

% Generar la matriz de probabilidades basada en la distancia inversa
probabilidades = exp(-distancias / umbral_distancia); % Probabilidad
inversa de la distancia

% Forzar la simetría de la matriz de probabilidades
probabilidades = (probabilidades + probabilidades') / 2; % Fuerza simetría
de la matriz
```

```

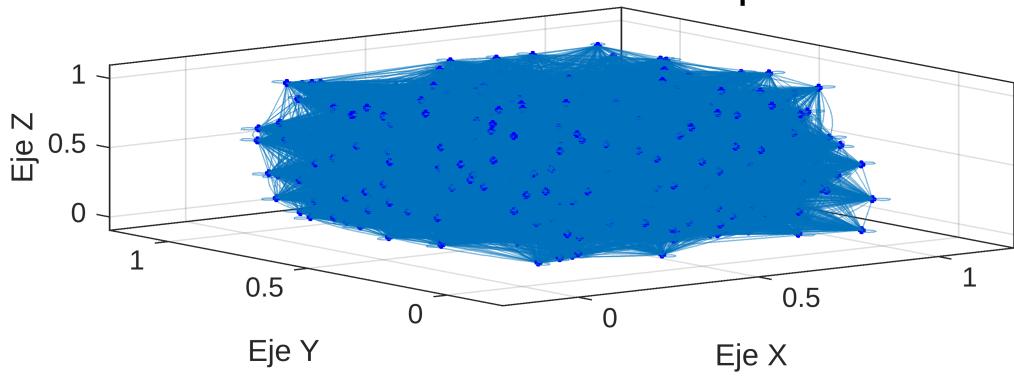
% Convertir la matriz de probabilidades en binaria (0 o 1)
probabilidades_binarias = (probabilidades > rand(num_nodos, num_nodos)); %
Establecer conexiones aleatorias

% Crear el grafo usando la matriz de adyacencia binaria y simétrica
grafo_nulo = digraph(probabilidades_binarias);

% Visualizar el modelo nulo en 3D
figure;
subplot(2, 1, 2);
plot(grafo_nulo, 'XData', coordenadas_3d(:, 1), 'YData', coordenadas_3d(:, 2), 'ZData', coordenadas_3d(:, 3), 'NodeColor', 'blue');
title('Modelo Nulo con Probabilidad de Conexión por Distancia en 3D');
xlabel('Eje X');
ylabel('Eje Y');
zlabel('Eje Z');
grid on;

```

Modelo Nulo con Probabilidad de Conexión por Distancia en 3D



12) Escribir una reseña de lo aprendido en el curso, incluyendo la importancia de conocer herramientas de teoría de grafos para comprender la conectividad del cerebro (mínimo 200 palabras).

El estudio de la conectividad cerebral, es decir, el cómo las distintas regiones del cerebro están interconectadas y cómo interactúan entre sí, es fundamental para la comprensión de diversos procesos cognitivos, emocionales y motores. En este sentido, conocer herramientas de teoría de grafos como MATLAB

Y Python, serán de suma importancia ya que estos nos permitirán modelar y analizar esta compleja red de interacciones.

Un ejemplo de esto sería su utilización en técnicas de neuroimagen como en la resonancia magnética, especialmente en resonancia magnética funcional (fMRI) y la resonancia magnética de difusión (dMRI), que nos proporciona información detallada sobre la estructura y la conectividad del cerebro y, utilizando la teoría de grafos, nos será posible representar las regiones cerebrales como nodos y las conexiones entre ellas como aristas, para así analizar las redes cerebrales en términos de topología, eficiencia y resiliencia. En resumen, las herramientas o lenguajes de programación como Matlab y Python nos permiten analizar grandes volúmenes de datos y así obtener una visión más clara de la organización y el funcionamiento.