Proyecto Final

Modelos Computacionales I

Marcela Aguirre Valdez

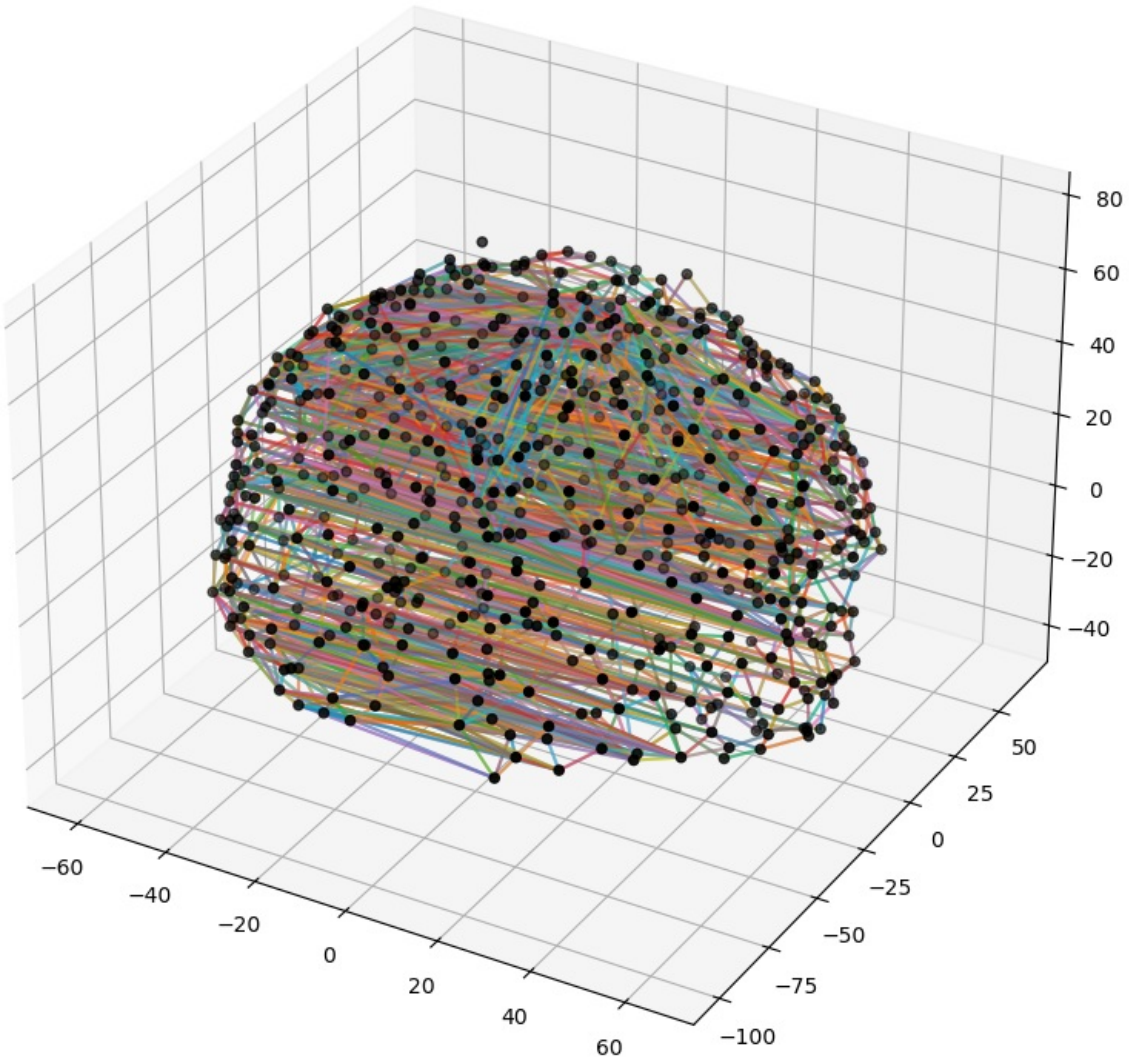
In [406…
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import networkx as nx
import scipy
import pandas as pd
from matplotlib import animation
from IPython.display import Image
from matplotlib.animation import FuncAnimation, PillowWriter

path = r"Coactivation_matrix.mat"
mat_json = scipy.io.loadmat(path)
x,y,z = mat_json['Coord'][:,0],mat_json['Coord'][:,1], mat_json['Coord'][:,2]
CM=mat_json['Coactivation_matrix']
```

Ejercicio 1

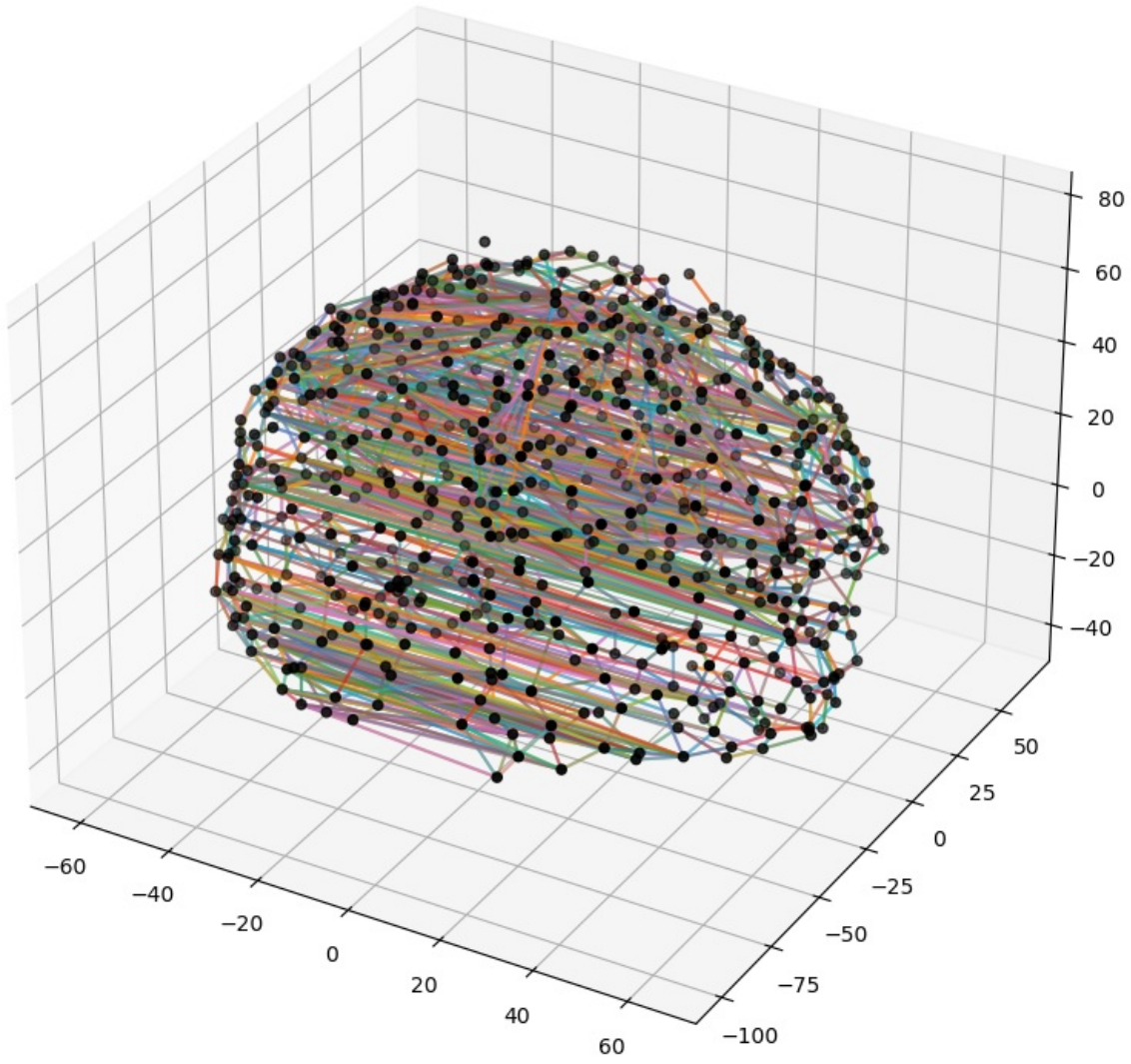
Definir grafos con la matriz estableciendo umbrales de coactivación de 0.8, 0.9 y 1 y graficar cada grafo. Añadir las coordenadas tridimensionales (incluidas en el archiv .mat).

In [408…
```
u = [0.08,0.09,0.1]
for i in u :
    print(i)
    cm= np.where(CM >= i, 1, 0)
    G = nx.from_numpy_array(cm, create_using=nx.DiGraph)
    # nx.draw(G, node_size=10)

    fig = plt.figure(figsize=(10, 10))
    ax = fig.add_subplot(projection='3d')

    ax.scatter3D(x, y, z, color='k')

    for edge in G.edges():
        n1, n2 = edge
        x_coords = [x[n1], x[n2]]
        y_coords = [y[n1], y[n2]]
        z_coords = [z[n1], z[n2]]
        ax.plot(x_coords, y_coords, z_coords, alpha=0.5)
    plt.show()
```

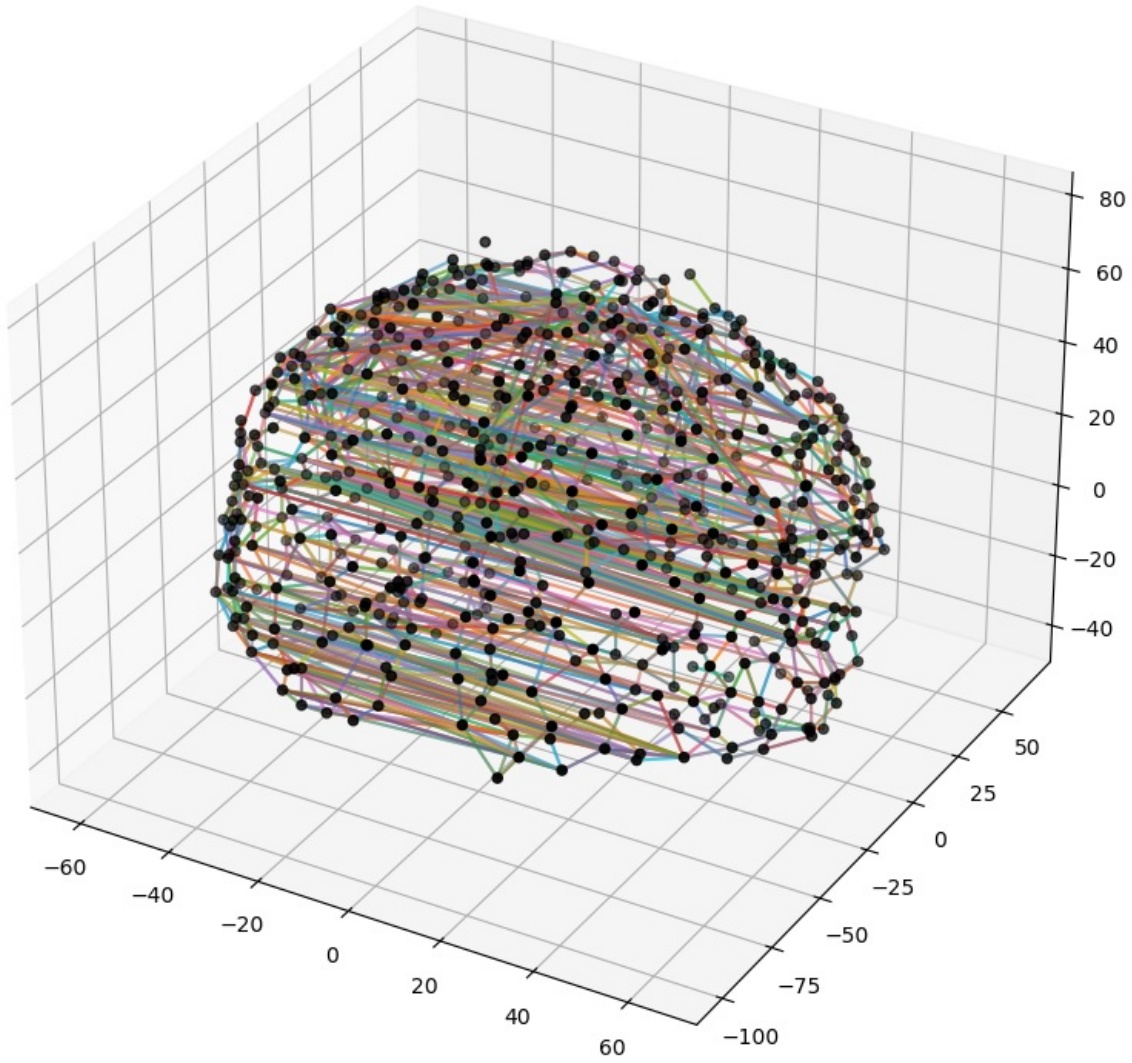
```
0.08
```



0.09



0.1



Modifiqué los umbrales a 0.08, 0.09 y 0.1 debido a que el rango de los valores de la matriz de coactivación no alcanza los umbrales establecidos en el ejercicio.

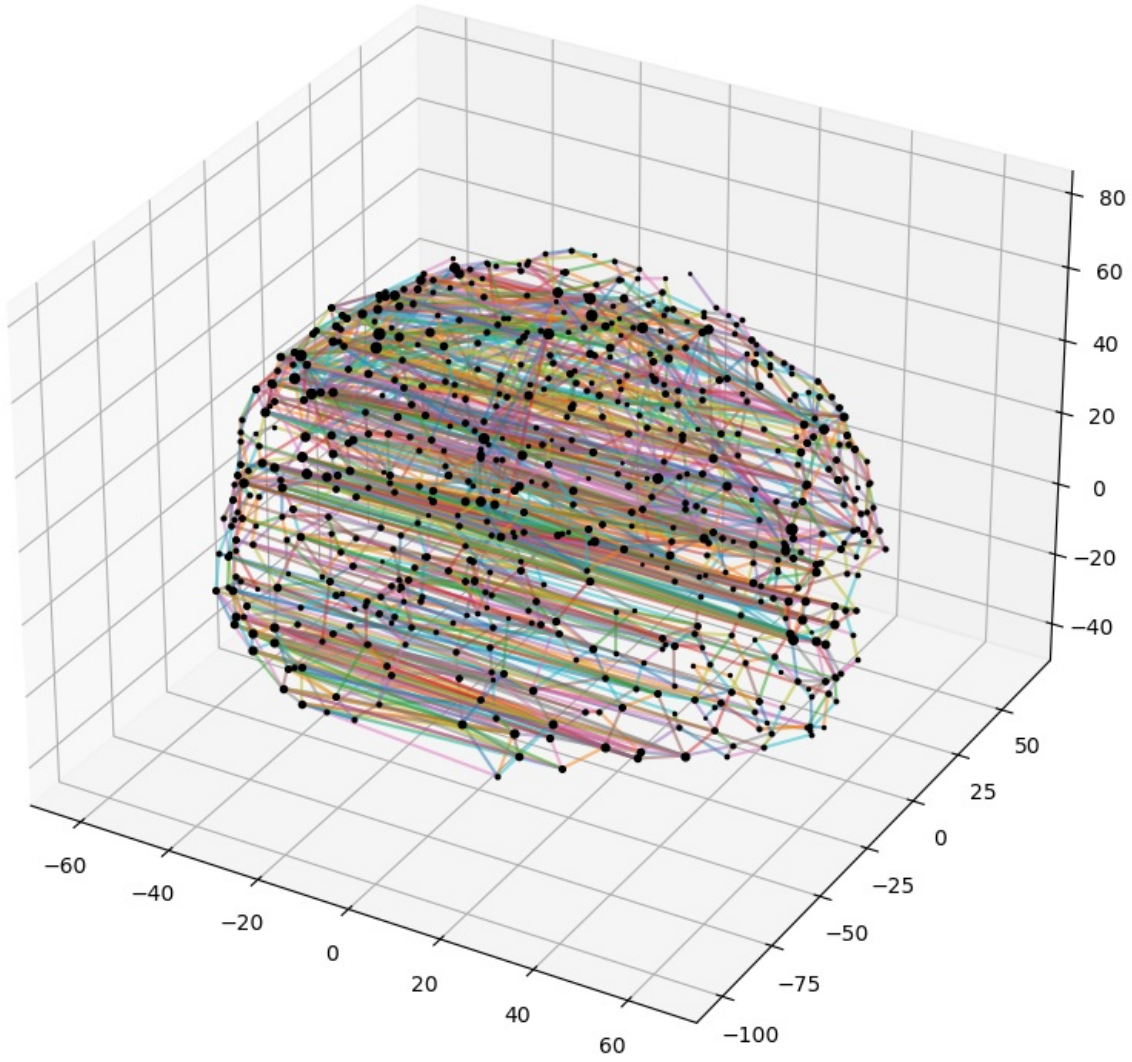
Ejercicio 2

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.

In [411…

```
CM[CM>0.09]=1
CM[CM<=0.09]=0
G = nx.from_numpy_array(CM, create_using=nx.Graph)

grados = dict(G.degree())
lista_grados = list(grados.values())

nodes = np.array([[i, j, k] for i, j, k in zip(x, y, z)])

fig = plt.figure(figsize=(10, 10))
ax = fig.add_subplot(projection='3d')

for i in range(len(nodes)):
    ax.scatter(x[i], y[i], z[i], color='k', s= lista_grados[i])

for edge in G.edges():
    node1, node2 = edge
    x_coords = [x[node1], x[node2]]
    y_coords = [y[node1], y[node2]]
    z_coords = [z[node1], z[node2]]
    ax.plot(x_coords, y_coords, z_coords, alpha=0.5)

def init():
    ax.view_init(elev=20, azim=0)
    return fig,

def animate(i):
    ax.view_init(elev=20, azim=i*4)
    return fig

ani = animation.FuncAnimation(fig, animate, init_func=init, frames=90, interval=200, blit=False)
plt.show()
#nx.draw_networkx(G, arrows=True, node_color='k') #, node_size=grados/3
```

```
C:\Users\marca\anaconda3\Lib\site-packages\matplotlib\animation.py:892: UserWarning: Animation was deleted witho
ut rendering anything. This is most likely not intended. To prevent deletion, assign the Animation to a variable
, e.g. `anim`, that exists until you output the Animation using `plt.show()` or `anim.save()`.
  warnings.warn(
```



Umbral 0.09

Ejercicio 3

Encontrar los hubs del grafo, y establecer el tamaño del nodo proporcional al valor del grad.o

In [414…
```
print(np.mean(lista_grados))
print(np.std(lista_grados))
umbral = np.mean(lista_grados)+np.std(lista_grados)
ix=np.where(lista_grados>umbral)
print('Los nodos hubs son', ix)
```

```
5.601880877742946
3.542592993040061
Los nodos hubs son (array([ 19,  38,  69,  70, 100, 121, 135, 186, 202, 213, 230, 235, 237,
       253, 261, 262, 266, 267, 271, 279, 285, 308, 309, 327, 328, 330,
       331, 334, 344, 345, 346, 350, 352, 353, 356, 361, 362, 373, 393,
       397, 400, 403, 407, 410, 416, 418, 452, 453, 454, 457, 465, 473,
       477, 481, 482, 485, 488, 491, 494, 496, 500, 532, 552, 553, 599,
       601, 603, 606, 607, 611, 612, 613, 615, 616, 617, 618, 619, 621,
       622, 629], dtype=int64),)
```

Ejercicio 4

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

In [416…
```
clusters = list(nx.connected_components(G))
num_clusters = len(clusters)
print("Número de clústeres:", num_clusters)
```

```
Número de clústeres: 2
```

La función nx.strongly_connected_components(G) en NetworkX se utiliza para identificar las componentes fuertemente conectadas de

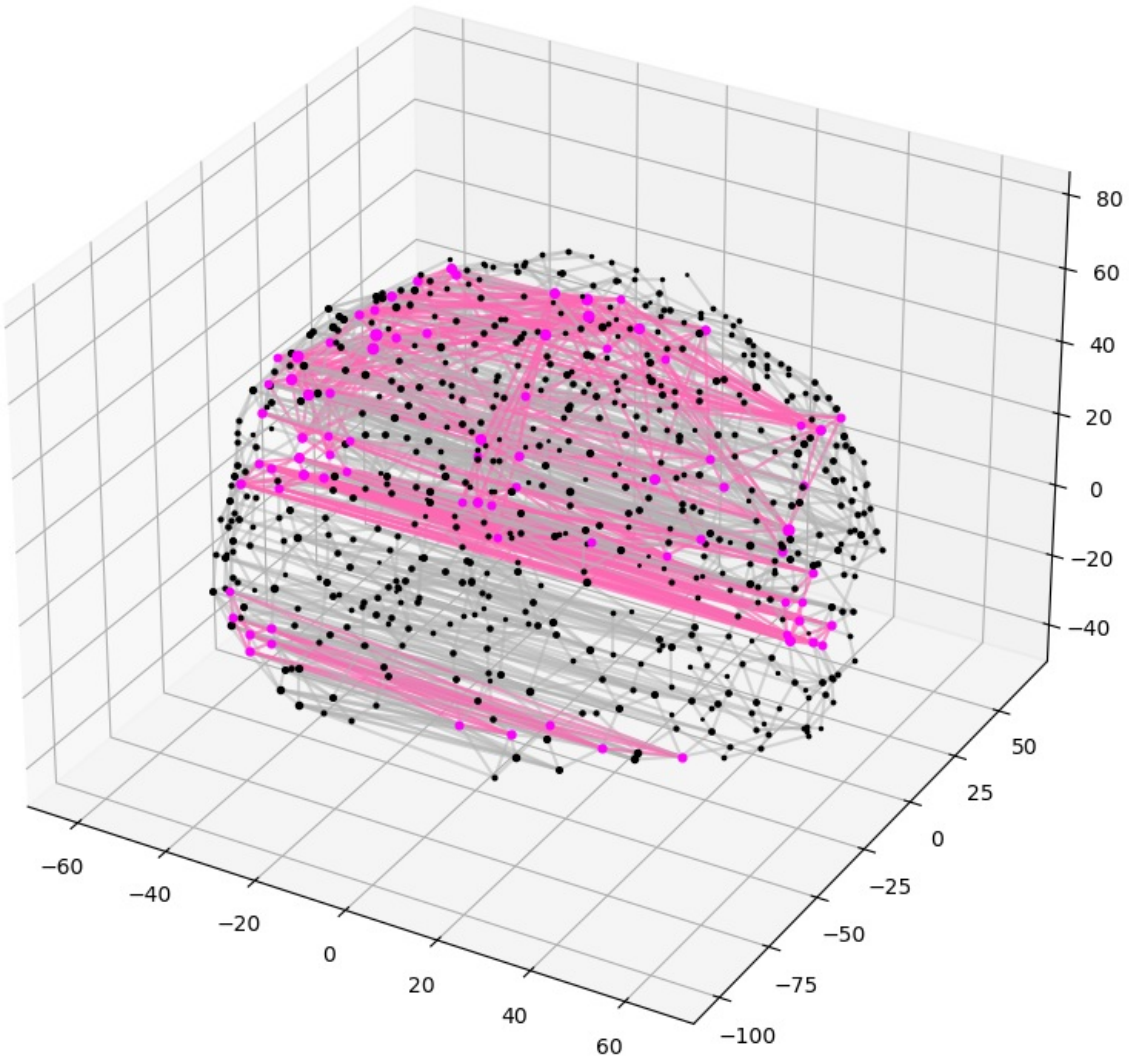
un grafo dirigido G. Estas son subredes donde cada nodo es accesible desde cualquier otro nodo dentro de la misma componente, y de esta manera nos permite separar en módulos al grafo según la tendencia de conectividad de los nodos..

Ejercicio 5

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

In [419…

```
rich_club = [nodo for nodo, grado in grados.items() if grado >= umbral]

figura = plt.figure(figsize=(10, 10))
ax = figura.add_subplot(projection='3d')

for i in range(len(nodes)):
    color = 'magenta' if i in rich_club else 'k'
    ax.scatter(x[i], y[i], z[i], color=color, s= lista_grados[i])

for edge in G.edges():
    node1, node2 = edge
    x_coords = [x[node1], x[node2]]
    y_coords = [y[node1], y[node2]]
    z_coords = [z[node1], z[node2]]
    if node1 in rich_club and node2 in rich_club:
        ax.plot(x_coords, y_coords, z_coords, c='hotpink', alpha=0.7, linewidth=1.5)
    else:
        ax.plot(x_coords, y_coords, z_coords, c=[.7, .7, .7], alpha=0.5)

Ax.set_title('Rich Club', fontsize=30)
plt.show()
```



Vemos que los nodos hubs tienen una alta conectividad interhemisféricas, donde cada área conecta con sus áreas homólogas contralaterales (en el otro hemisferio). Estás áreas hubs son cortezas occipitales y temporales de asociación y prefrontales. Las cortezas de asociación reciben inputs multisensoriales que integran, y por ello deben estar altamente conectadas con estas estrucutras

sensoriales. Vemos también la alta conectividad de la corteza prefrontal que ejerce un control top-down en el resto de estructuras cerebrales para regular la conducta.

Ejercicio 6

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

In [422…
```
Gi = G.copy()
Gi.remove_nodes_from(rich_club)

print('Número de nodos G:', G.number_of_nodes(), 'G modificada:', Gi.number_of_nodes())
print('Número de aristas G:', G.number_of_edges(), 'G modificada:', Gi.number_of_edges())
print('Grado promedio G:', np.mean([grado for nodo, grado in G.degree()]), 'G modificada:', np.mean([grado for
print('Coeficiente de clustering promedio G:', nx.average_clustering(G), 'G modificada:', nx.average_clustering
print('Centralidad de cercanía G:', np.mean(list(nx.closeness_centrality(G).values())), 'G modificada:', np.mea
print('Centralidad de intermediación G:', np.mean(list(nx.betweenness_centrality(G).values())), 'G modificada:'
```

```
Número de nodos G: 638 G modificada: 558
Número de aristas G: 1787 G modificada: 1076
Grado promedio G: 5.601880877742946 G modificada: 3.85663082437276
Coeficiente de clustering promedio G: 0.2782681200009071 G modificada: 0.23573419809978932
Centralidad de cercanía G: 0.143580181791967 G modificada: 0.10216627891884043
Centralidad de intermediación G: 0.00966904180492804 G modificada: 0.014886995829872266
```

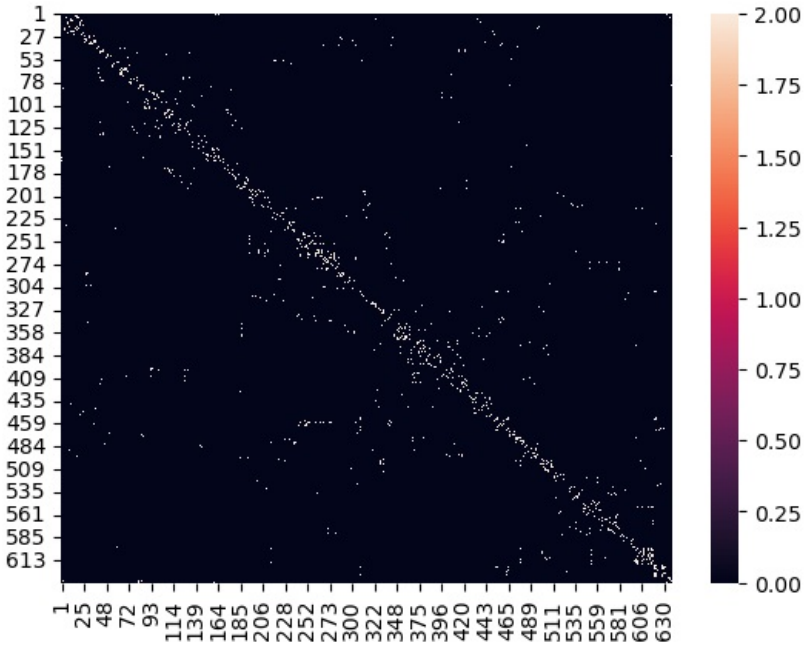
Se eliminaron 80 nodos y de ellos, 1468 aristas. esto es más de un tercio del número de aristas totales correspondientes a un octavo del total de nodos. El grado promedio por lo tanto disminuye, al igual que el coeficient de cluster y la centralidad de cercanía. Aumenta la centralidad de intermediación.

Ejercicio 7

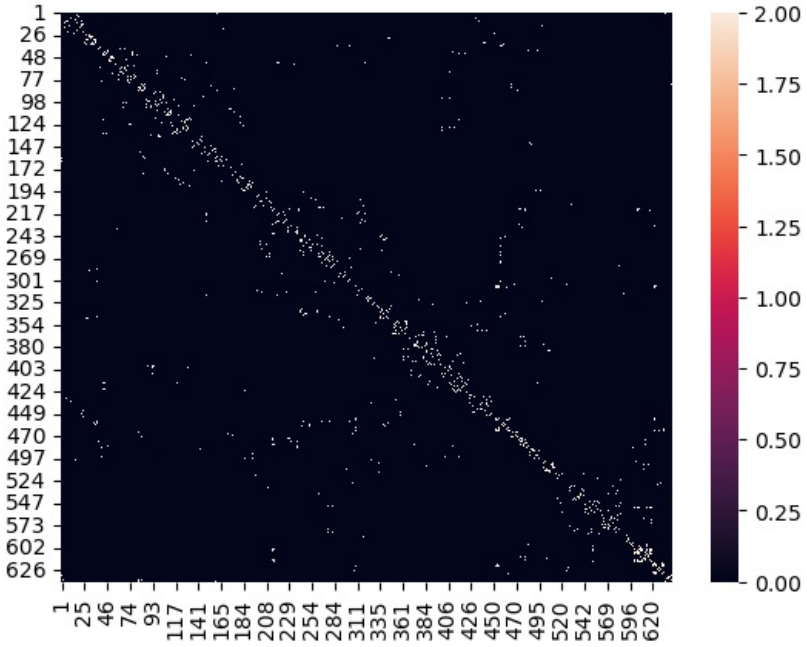
Quitar 10%-50% de los nodos con mayor medida de intermediación y describir cómo cambian las propiedades topológicas del grafo, hacer comparativas del grado coeficiente de cluster, coeficiente de mundo pequeño y las medidas de centralid d (cercanía, intermediación)

In [425…
```
centralidad_intermediacion = nx.betweenness_centrality(G)
nodos_ordenados = sorted(centralidad_intermediacion, key=centralidad_intermediacion.get, reverse=True)

porcentaje = [0.1, 0.2, 0.3, 0.4, 0.5]
for i in porcentaje :
    n_eliminar = int(i * G.number_of_nodes())

    Gi= nx.from_numpy_array(A)
    n_eliminar = nodos_ordenados[:n_eliminar]
    Gi.remove_nodes_from(n_eliminar)

    print('Nodos eliminados en grafo:',i , ': ', n_eliminar)

    M = nx.to_pandas_adjacency(Gi)
    sns.heatmap(M)
    plt.show()

    print('Comparativa porcentaje', i)
    print('Número de nodos G:', G.number_of_nodes(), 'G modificada:', Gi.number_of_nodes())
    print('Número de aristas G:', G.number_of_edges(), 'G modificada:', Gi.number_of_edges())
    print('Grado promedio G:', np.mean([grado for nodo, grado in G.degree()]), 'G modificada:', np.mean([grado
    print('Coeficiente de clustering promedio G:', nx.average_clustering(G), 'G modificada:', nx.average_cluste
    print('Centralidad de cercanía G:', np.mean(list(nx.closeness_centrality(G).values())), 'G modificada:', np
    print('Centralidad de intermediación G:', np.mean(list(nx.betweenness_centrality(G).values())), 'G modifica
```

```
Nodos eliminados en grafo: 0.1 :  [373, 235, 334, 285, 330, 481, 619, 352, 400, 220, 275, 277, 121, 280, 353, 13
4, 38, 50, 135, 482, 397, 16, 427, 589, 618, 431, 2, 591, 565, 488, 3, 0, 371, 65, 128, 154, 344, 286, 605, 532,
160, 517, 553, 237, 356, 611, 362, 159, 328, 410, 491, 546, 230, 513, 454, 37, 281, 346, 302, 57, 549, 590, 153]
```

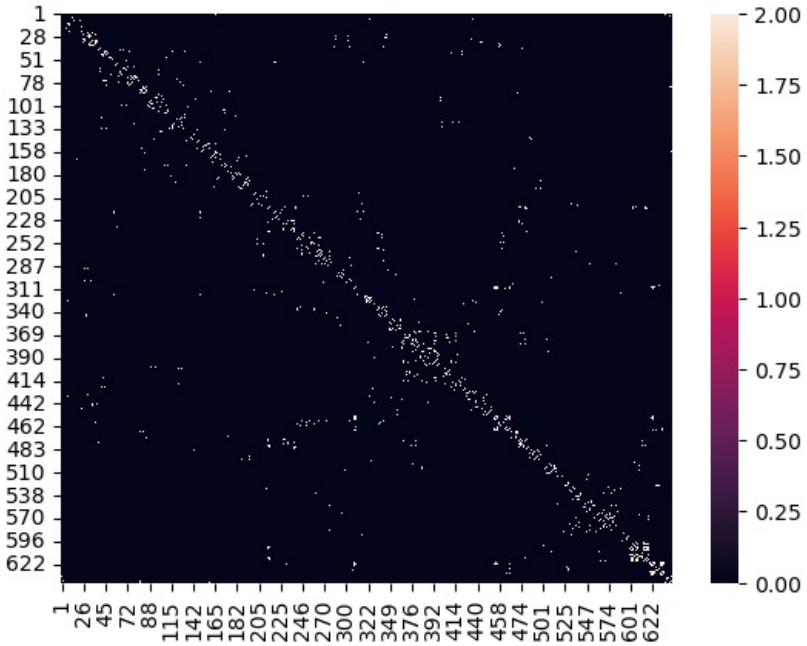


Comparativa porcentaje 0.1
Número de nodos G: 638 G modificada: 575
Número de aristas G: 1787 G modificada: 1250
Grado promedio G: 5.601880877742946 G modificada: 4.3478260869565215
Coeficiente de clustering promedio G: 0.2782681200009071 G modificada: 0.26121185095098115
Centralidad de cercanía G: 0.143580181791967 G modificada: 0.10434164118653266
Centralidad de intermediación G: 0.00966904180492804 G modificada: 0.01514029420155019
Nodos eliminados en grafo: 0.2 : [373, 235, 334, 285, 330, 481, 619, 352, 400, 220, 275, 277, 121, 280, 353, 134, 38, 50, 135, 482, 397, 16, 427, 589, 618, 431, 2, 591, 565, 488, 3, 0, 371, 65, 128, 154, 344, 286, 605, 532, 160, 517, 553, 237, 356, 611, 362, 159, 328, 410, 491, 546, 230, 513, 454, 37, 281, 346, 302, 57, 549, 590, 153, 496, 69, 593, 485, 592, 500, 6, 329, 55, 196, 487, 629, 223, 272, 267, 434, 186, 579, 276, 62, 97, 582, 559, 310, 292, 545, 232, 193, 253, 22, 195, 120, 521, 584, 108, 67, 560, 554, 100, 262, 523, 399, 345, 599, 359, 303, 279, 430, 202, 250, 499, 428, 61, 512, 52, 99, 268, 318, 149, 375, 259, 289, 291, 112]

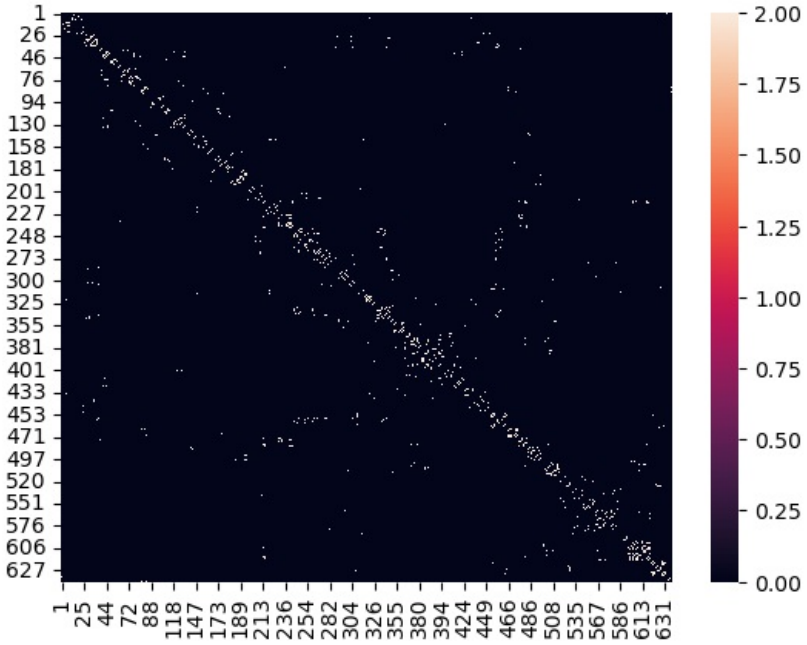


Comparativa porcentaje 0.2
Número de nodos G: 638 G modificada: 511

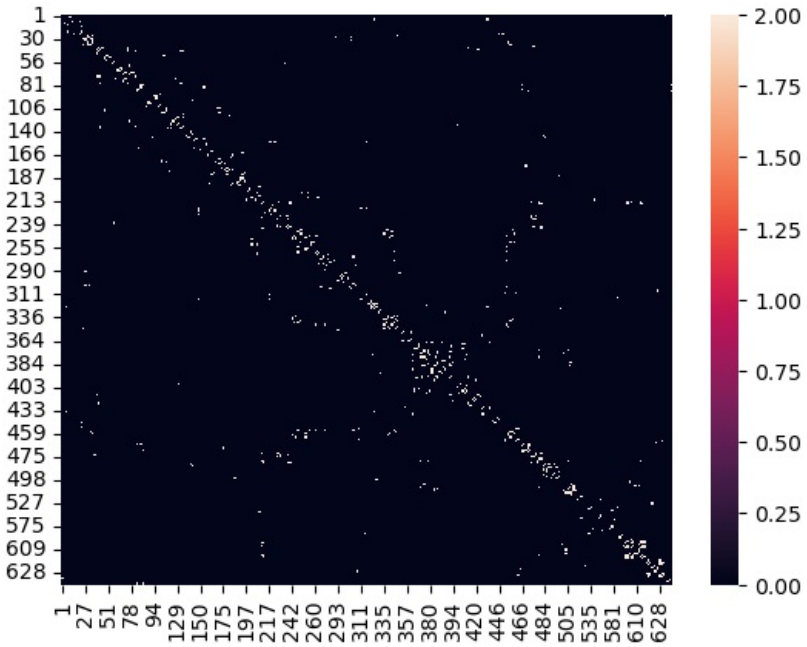
Número de aristas G: 1787 G modificada: 938
Grado promedio G: 5.601880877742946 G modificada: 3.671232876712329
Coeficiente de clustering promedio G: 0.2782681200009071 G modificada: 0.2523684947364005
Centralidad de cercanía G: 0.143580181791967 G modificada: 0.07348860683304327
Centralidad de intermediación G: 0.00966904180492804 G modificada: 0.017994279553735535
Nodos eliminados en grafo: 0.3 : [373, 235, 334, 285, 330, 481, 619, 352, 400, 220, 275, 277, 121, 280, 353, 13
4, 38, 50, 135, 482, 397, 16, 427, 589, 618, 431, 2, 591, 565, 488, 3, 0, 371, 65, 128, 154, 344, 286, 605, 532,
160, 517, 553, 237, 356, 611, 362, 159, 328, 410, 491, 546, 230, 513, 454, 37, 281, 346, 302, 57, 549, 590, 153,
496, 69, 593, 485, 592, 500, 6, 329, 55, 196, 487, 629, 223, 272, 267, 434, 186, 579, 276, 62, 97, 582, 559, 310
, 292, 545, 232, 193, 253, 22, 195, 120, 521, 584, 108, 67, 560, 554, 100, 262, 523, 399, 345, 599, 359, 303, 27
9, 430, 202, 250, 499, 428, 61, 512, 52, 99, 268, 318, 149, 375, 259, 289, 291, 112, 361, 111, 494, 48, 211, 18,
432, 222, 407, 331, 41, 104, 327, 144, 271, 598, 503, 479, 564, 516, 363, 350, 122, 531, 110, 116, 171, 257, 456
, 416, 333, 597, 226, 552, 194, 524, 204, 283, 136, 562, 15, 247, 93, 548, 542, 315, 21, 417, 339, 312, 563, 405
, 125, 607, 162, 438, 132, 103, 526, 17, 91, 423, 60, 117]



Comparativa porcentaje 0.3
Número de nodos G: 638 G modificada: 447
Número de aristas G: 1787 G modificada: 717
Grado promedio G: 5.601880877742946 G modificada: 3.208053691275168
Coeficiente de clustering promedio G: 0.2782681200009071 G modificada: 0.25893898544905236
Centralidad de cercanía G: 0.143580181791967 G modificada: 0.044999805034782465
Centralidad de intermediación G: 0.00966904180492804 G modificada: 0.010800137832945523
Nodos eliminados en grafo: 0.4 : [373, 235, 334, 285, 330, 481, 619, 352, 400, 220, 275, 277, 121, 280, 353, 13
4, 38, 50, 135, 482, 397, 16, 427, 589, 618, 431, 2, 591, 565, 488, 3, 0, 371, 65, 128, 154, 344, 286, 605, 532,
160, 517, 553, 237, 356, 611, 362, 159, 328, 410, 491, 546, 230, 513, 454, 37, 281, 346, 302, 57, 549, 590, 153,
496, 69, 593, 485, 592, 500, 6, 329, 55, 196, 487, 629, 223, 272, 267, 434, 186, 579, 276, 62, 97, 582, 559, 310
, 292, 545, 232, 193, 253, 22, 195, 120, 521, 584, 108, 67, 560, 554, 100, 262, 523, 399, 345, 599, 359, 303, 27
9, 430, 202, 250, 499, 428, 61, 512, 52, 99, 268, 318, 149, 375, 259, 289, 291, 112, 361, 111, 494, 48, 211, 18,
432, 222, 407, 331, 41, 104, 327, 144, 271, 598, 503, 479, 564, 516, 363, 350, 122, 531, 110, 116, 171, 257, 456
, 416, 333, 597, 226, 552, 194, 524, 204, 283, 136, 562, 15, 247, 93, 548, 542, 315, 21, 417, 339, 312, 563, 405
, 125, 607, 162, 438, 132, 103, 526, 17, 91, 423, 60, 117, 635, 209, 138, 167, 105, 585, 637, 414, 480, 142, 418
, 320, 231, 36, 124, 465, 98, 622, 367, 522, 539, 557, 538, 439, 612, 473, 368, 156, 306, 86, 422, 514, 71, 210,
529, 4, 395, 70, 425, 398, 95, 314, 73, 180, 123, 436, 577, 29, 544, 169, 448, 137, 54, 358, 406, 308, 151, 165,
206, 412, 261, 596, 219, 351]



Comparativa porcentaje 0.4
Número de nodos G: 638 G modificada: 383
Número de aristas G: 1787 G modificada: 529
Grado promedio G: 5.601880877742946 G modificada: 2.7624020887728458
Coeficiente de clustering promedio G: 0.2782681200009071 G modificada: 0.22608893862157564
Centralidad de cercanía G: 0.143580181791967 G modificada: 0.03190754354496044
Centralidad de intermediación G: 0.00966904180492804 G modificada: 0.009535330850994247
Nodos eliminados en grafo: 0.5 : [373, 235, 334, 285, 330, 481, 619, 352, 400, 220, 275, 277, 121, 280, 353, 134, 38, 50, 135, 482, 397, 16, 427, 589, 618, 431, 2, 591, 565, 488, 3, 0, 371, 65, 128, 154, 344, 286, 605, 532, 160, 517, 553, 237, 356, 611, 362, 159, 328, 410, 491, 546, 230, 513, 454, 37, 281, 346, 302, 57, 549, 590, 153, 496, 69, 593, 485, 592, 500, 6, 329, 55, 196, 487, 629, 223, 272, 267, 434, 186, 579, 276, 62, 97, 582, 559, 310, 292, 545, 232, 193, 253, 22, 195, 120, 521, 584, 108, 67, 560, 554, 100, 262, 523, 399, 345, 599, 359, 303, 279, 430, 202, 250, 499, 428, 61, 512, 52, 99, 268, 318, 149, 375, 259, 289, 291, 112, 361, 111, 494, 48, 211, 18, 432, 222, 407, 331, 41, 104, 327, 144, 271, 598, 503, 479, 564, 516, 363, 350, 122, 531, 110, 116, 171, 257, 456, 416, 333, 597, 226, 552, 194, 524, 204, 283, 136, 562, 15, 247, 93, 548, 542, 315, 21, 417, 339, 312, 563, 405, 125, 607, 162, 438, 132, 103, 526, 17, 91, 423, 60, 117, 635, 209, 138, 167, 105, 585, 637, 414, 480, 142, 418, 320, 231, 36, 124, 465, 98, 622, 367, 522, 539, 557, 538, 439, 612, 473, 368, 156, 306, 86, 422, 514, 71, 210, 529, 4, 395, 70, 425, 398, 95, 314, 73, 180, 123, 436, 577, 29, 544, 169, 448, 137, 54, 358, 406, 308, 151, 165, 206, 412, 261, 596, 219, 351, 574, 28, 551, 365, 534, 263, 184, 452, 568, 547, 519, 258, 158, 571, 580, 113, 387, 453, 126, 348, 468, 201, 550, 528, 509, 296, 32, 163, 11, 441, 570, 264, 381, 588, 613, 227, 7, 238, 10, 66, 541, 450, 587, 508, 578, 35, 447, 176, 199, 556, 25, 85, 274, 177, 555, 72, 566, 40, 444, 192, 221, 42, 567, 602]




```
Comparativa porcentaje 0.5
Número de nodos G: 638 G modificada: 319
Número de aristas G: 1787 G modificada: 381
Grado promedio G: 5.601880877742946 G modificada: 2.3887147335423196
Coeficiente de clustering promedio G: 0.2782681200009071 G modificada: 0.19838035527690698
Centralidad de cercanía G: 0.143580181791967 G modificada: 0.020270047054337905
Centralidad de intermediación G: 0.00966904180492804 G modificada: 0.002304560042297328
```

Conforme se aumenta el porcentaje de conexiones eliminadas, notamos que disminuye el grado promedio, al igual que el coeficient de cluster y la centralidad de cercanía. Mientras que la medida de centralidad de intermediación aumenta para los porcentajes eliminados más bajos, y de ahí comienza a descender de nuevo hasta ser menor que la medida de intermediación original.

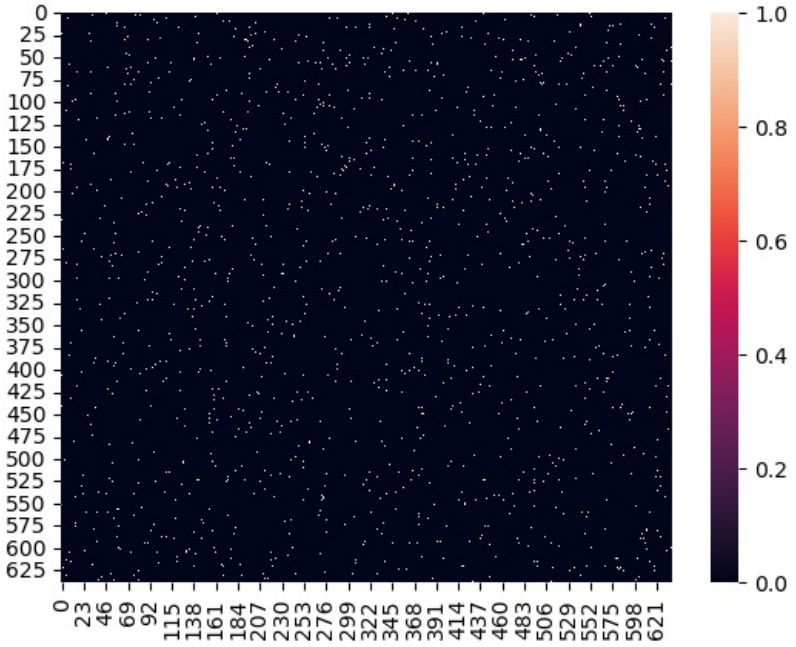
Ejercicio 8

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 origina del cerebro.

In [428…

```
num_nodos = G.number_of_nodes()
num_aristas = G.number_of_edges()

probabilidad_arista = num_aristas / (num_nodos * (num_nodos - 1) / 2)

Gi = nx.erdos_renyi_graph(num_nodos, probabilidad_arista)

M = nx.to_pandas_adjacency(Gi)
sns.heatmap(M)

print('Número de nodos G:', G.number_of_nodes(), 'G modificada:', Gi.number_of_nodes())
print('Número de aristas G:', G.number_of_edges(), 'G modificada:', Gi.number_of_edges())
print('Grado promedio G:', np.mean([grado for nodo, grado in G.degree()]), 'G modificada:', np.mean([grado for
print('Coeficiente de clustering promedio G:', nx.average_clustering(G), 'G modificada:', nx.average_clustering
print('Centralidad de cercanía G:', np.mean(list(nx.closeness_centrality(G).values())), 'G modificada:', np.mea
print('Centralidad de intermediación G:', np.mean(list(nx.betweenness_centrality(G).values())), 'G modificada:'
```

```
Número de nodos G: 638 G modificada: 638
Número de aristas G: 1787 G modificada: 1817
Grado promedio G: 5.601880877742946 G modificada: 5.695924764890282
Coeficiente de clustering promedio G: 0.2782681200009071 G modificada: 0.007936690616941394
Centralidad de cercanía G: 0.143580181791967 G modificada: 0.25590192473494444
Centralidad de intermediación G: 0.00966904180492804 G modificada: 0.004541226657594349
```



El total de conexiones se distribuye de manera aleatoria y uniforme en la matriz de coactivación, el grado promedio es comparable al grado promedio original. Disminuye el coeficiente de cluster, la medida de centralidad de intermediación y aumenta la de cercanía.

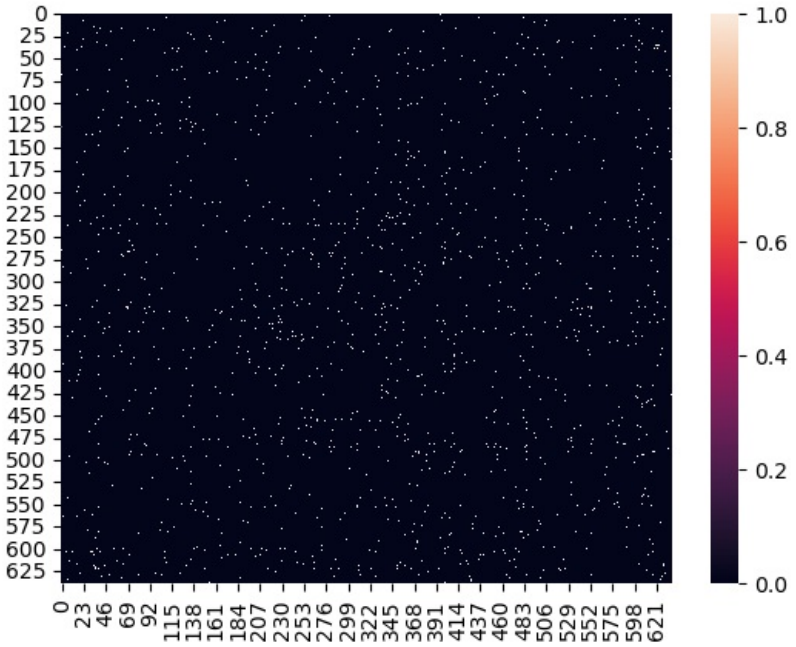
Ejercicio 9

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

In [431…

```
degree_sequence = [degree for node, degree in G.degree()]

Gi = nx.configuration_model(degree_sequence, create_using=nx.Graph)

Gi = nx.Graph(Gi)  #eliminar bucles
Gi.remove_edges_from(nx.selfloop_edges(Gi))

M = nx.to_pandas_adjacency(Gi)
sns.heatmap(M)

print('Número de nodos G:', G.number_of_nodes(), 'G modificada:', Gi.number_of_nodes())
print('Número de aristas G:', G.number_of_edges(), 'G modificada:', Gi.number_of_edges())
print('Grado promedio G:', np.mean([grado for nodo, grado in G.degree()]), 'G modificada:', np.mean([grado for
print('Coeficiente de clustering promedio G:', nx.average_clustering(G), 'G modificada:', nx.average_clustering
print('Centralidad de cercanía G:', np.mean(list(nx.closeness_centrality(G).values())), 'G modificada:', np.mea
print('Centralidad de intermediación G:', np.mean(list(nx.betweenness_centrality(G).values())), 'G modificada:'
```

```
Número de nodos G: 638 G modificada: 638
Número de aristas G: 1787 G modificada: 1777
Grado promedio G: 5.601880877742946 G modificada: 5.570532915360501
Coeficiente de clustering promedio G: 0.2782681200009071 G modificada: 0.014439939465610906
Centralidad de cercanía G: 0.143580181791967 G modificada: 0.26441604462366886
Centralidad de intermediación G: 0.00966904180492804 G modificada: 0.004392260154877499
```



La cantidad de conexiones es comparable y no es exactamente igual porque se eliminan los bucles. El grado promedio es comparable al grado promedio original. Aumenta el coefiente de cluster, al igual que la centralidad de cercanía, pero disminuye la centralidad de intermediación.

Ejercicio 10

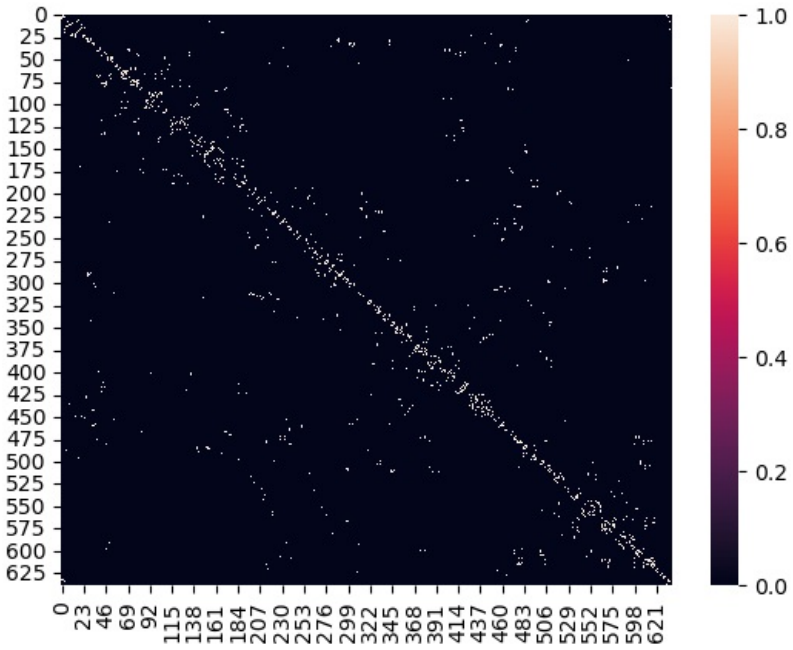
Generar un modelo nulo utilizando una probabilidad de conexión en función de la distancia geométrica, con el mismo número de nodos y conexiones y compara sus propiedades y discutir la importancia de las conexiones a larga distancia en el cerebro.

In [434…

```
posiciones = {i: (x[i], y[i], z[i]) for i in range(len(G.nodes))}

num_nodes = len(G.nodes)
distancias = np.zeros((num_nodes, num_nodes))
for i in range(num_nodes):
    for j in range(num_nodes):
        if i != j:
            distancias[i, j] = np.linalg.norm(np.array(posiciones[i]) - np.array(posiciones[j]))

def probabilidad_conexion(distancia, beta=1.0):
    return np.exp(-beta * distancia)

Gi = nx.Graph()
Gi.add_nodes_from(range(num_nodes))

probabilidades = np.exp(-distancias)
probabilidades[np.tril_indices(num_nodes)] = 0

posibles_aristas = np.column_stack(np.where(probabilidades > 0))
pesos = probabilidades[posibles_aristas[:, 0], posibles_aristas[:, 1]]

seleccionadas = np.random.choice(
```

```
    len(pesos),
    size=num_aristas,
    replace=False,
    p=pesos / pesos.sum(),
)

Gi.add_edges_from(posibles_aristas[seleccionadas])

M = nx.to_pandas_adjacency(Gi)
sns.heatmap(M)

print('Número de nodos G:', G.number_of_nodes(), 'G modificada:', Gi.number_of_nodes())
print('Número de aristas G:', G.number_of_edges(), 'G modificada:', Gi.number_of_edges())
print('Grado promedio G:', np.mean([grado for nodo, grado in G.degree()]), 'G modificada:', np.mean([grado for
print('Coeficiente de clustering promedio G:', nx.average_clustering(G), 'G modificada:', nx.average_clustering
print('Centralidad de cercanía G:', np.mean(list(nx.closeness_centrality(G).values())), 'G modificada:', np.mea
print('Centralidad de intermediación G:', np.mean(list(nx.betweenness_centrality(G).values())), 'G modificada:'
```

```
Número de nodos G: 638 G modificada: 638
Número de aristas G: 1787 G modificada: 1787
Grado promedio G: 5.601880877742946 G modificada: 5.601880877742946
Coeficiente de clustering promedio G: 0.2782681200009071 G modificada: 0.38852926940701316
Centralidad de cercanía G: 0.143580181791967 G modificada: 0.10218249210074462
Centralidad de intermediación G: 0.00966904180492804 G modificada: 0.013732317501255135
```



El grafo nulo contiene exactamente la misma cantidad de nodos y aristas, al igual que el grado promedio de los nodos. El coeficiente de cluster y centralidad de intermediación resultan mayores a los de la matriz original, y sin embargo disminuye ligeramente la medida de centralidad de cercanía.

Las conexiones a larga distancia en el cerebro son cruciales para diversas funciones cognitivas. Permiten la integración de información entre diferentes áreas,promueven la flexibilidad cognitiva en equilibrio con una robustez mantenida del sistema, facilitan la sincronización de actividades neuronales como la integración sensorial para la ejecución motora, aumentan la eficiencia temporal de la transmisión de señales y aumentan la resiliencia del cerebro. Estas conexiones son esenciales para la eficiencia, permitiendo respuestas rápidas y coordinadas a estímulos complejos.

Ejercicio 11

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

El cerebro adquiere funcionalidad gracias a las propiedades emergentes que surgen de la conectividad de los circuitos neuronales. Estos circuitos procesan la información de manera eficiente para garantizar su correcta transmisión. Estudiar la conectividad cerebral desde una perspectiva teórica y matemática nos permite entender los patrones de conexión que caracterizan a los cerebros de diferentes organismos, los cuales están adaptados a sus respectivos entornos y conductas. Además, esto nos ayuda a comprender las implicaciones funcionales y conductuales que conlleva su conectividad estructural y funcional.

Actualmente, las herramientas disponibles se enfocan en medidas como el coeficiente de agrupamiento y las distintas formas de centralidad, ya sea de cercanía o de intermediación, que analizan el papel de cada nodo dentro de los circuitos. Estas métricas nos permiten observar qué tan conectados están los nodos y con quiénes establecen conexiones. En el cerebro es esencial mantener un equilibrio entre la especialización local y la integración global. Este equilibrio se logra mediante la presencia de hubs, conexiones largas y enlaces entre diferentes clústeres o módulos funcionales.

Los hubs, o nodos altamente conectados, desempeñan un papel central al actuar como puntos clave de integración y distribución de información. Estas estructuras permiten que diferentes módulos cerebrales puedan comunicarse de manera eficiente, facilitando la coordinación de diversas funciones cognitivas y comportamentales. Por otro lado, las conexiones largas, que conectan regiones distantes del cerebro, son cruciales para la comunicación global. A pesar de ser metabólicamente costosas, estas conexiones son fundamentales para integrar información de diferentes áreas especializadas, lo que favorece procesos complejos como la memoria, la percepción y la toma de decisiones. Finalmente, las conexiones entre clústeres aseguran la cohesión funcional del cerebro. Este tipo de enlaces permiten que las áreas especializadas trabajen en conjunto y mantienen la organización modular del cerebro, característica de los sistemas de "mundo pequeño", los cuales combinan eficiencia local y global.

En resumen, la interacción de hubs, conexiones largas y enlaces entre clústeres no solo optimiza la transmisión de información, sino que también dota al cerebro de una notable resiliencia frente a fallos o lesiones. Estas características son esenciales para el funcionamiento del cerebro sano y ofrecen claves para entender las alteraciones que ocurren en condiciones neurodegenerativas o en trastornos psiquiátricos como la esquizofrenia.).