

# EJ1

September 25, 2018

## 1 EJERCICIO 1: INTERACCION DE PROTEINAS

Considere las tres redes de interacción de proteínas relevadas para levadura disponibles en la página de la materia. Se trata de: una red de interacciones binarias (yeast\_Y2H.txt), de copertenencia a complejos proteicos (yeast\_AP-MS.txt) y obtenida de literatura (yeast\_LIT.txt) obtenidas del Yeast Interactome Database.

```
In [29]: #paquetes
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt
%matplotlib inline
```

FUNCIÓN PARA ABRIR ARCHIVOS .TXT

```
In [30]: def ldata(archive):
    f=open(archive)
    data=[]
    for line in f:
        line=line.strip()
        col=line.split()
        data.append(col)
    return data
```

ABRIMOS LAS 3 REDES

```
In [31]: redInteraccionesBinarias = ldata('yeast_Y2H.txt')
redComplejosProteicos = ldata('yeast_AP-MS.txt')
redLiteratura = ldata('yeast_LIT.txt')
```

definimos una funcion que nos va a hacer los grafos

```
In [32]: def grafo(datosRed):
    G = nx.Graph()
    for i in range(np.shape(datosRed)[0]):
        G.add_edges_from([(datosRed[i][0], datosRed[i][1])])
    return G
```

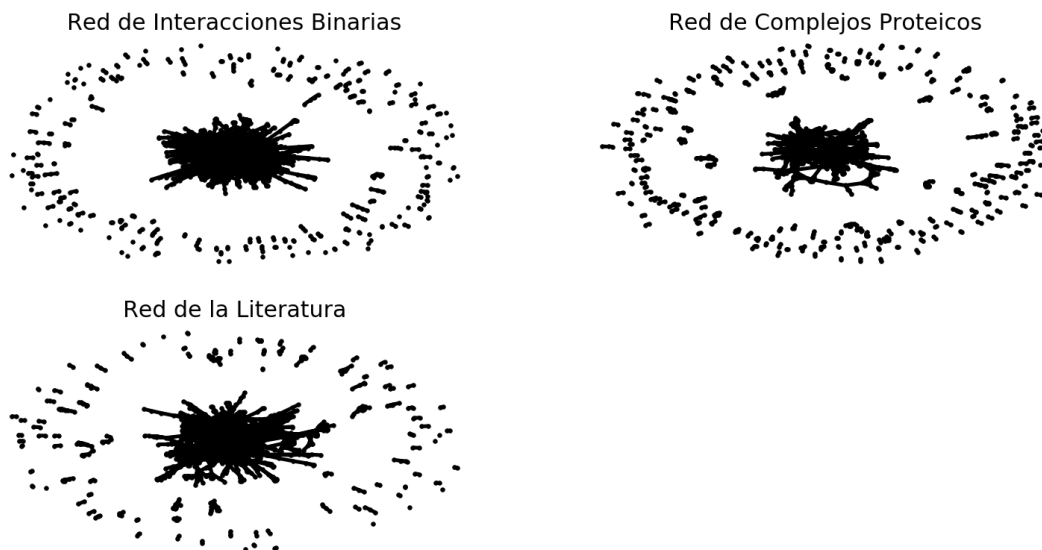
```
In [33]: grafoRedInteraccionesBinarias = grafo(redInteraccionesBinarias)
grafoRedComplejosProteicos = grafo(redComplejosProteicos)
grafoRedLiteratura = grafo(redLiteratura)
```

## 2 a. Presente una comparación gráfica de las 3 redes.

```
In [34]: #ponemos estas opciones para que grafique igual en todos los subplots
options = {
    'node_color': 'black',
    'node_size': 30,
    'width': 5,
}

f = plt.figure(figsize=(26,13))                                     #con esta linea le damos
f.suptitle('Tres redes',fontweight="bold", size=40)
sub1 = f.add_subplot(221)
nx.draw(grafoRedInteraccionesBinarias, **options)
sub2 = plt.subplot(222)
nx.draw(grafoRedComplejosProteicos, **options)
sub3 = plt.subplot(223)
nx.draw(grafoRedLiteratura, **options)
sub1.set_title('Red de Interacciones Binarias', size=30)
sub2.set_title('Red de Complejos Proteicos', size=30)
sub3.set_title('Red de la Literatura', size=30)
plt.show()
```

### Tres redes



## 3 NUMERO DE NODOS

```
In [35]: nodosInteraccionesBinarias = grafoRedInteraccionesBinarias.number_of_nodes()
nodosComplejosProteicos = grafoRedComplejosProteicos.number_of_nodes()
```

```

    nodosRedLiteratura = grafoRedLiteratura.number_of_nodes()

    print (nodosInteraccionesBinarias, nodosComplejosProteicos, nodosRedLiteratura)

2018 1622 1536

```

## 4 ENLACES

```

In [36]: enlacesInteraccionesBinarias = grafoRedInteraccionesBinarias.number_of_edges()
        enlacesComplejosProteicos = grafoRedComplejosProteicos.number_of_edges()
        enlacesRedLiteratura = grafoRedLiteratura.number_of_edges()

        print (enlacesInteraccionesBinarias, enlacesComplejosProteicos, enlacesRedLiteratura)

2930 9070 2925

```

DIRIGIDA O NO DIRIGIDA: ESTO SE INTERPRETA POR CONOCIMIENTO DEL PROCESO POR EJEMPLO: EN REDES SOCIALES COMO TWITTER O INSTAGRAM PUEDO SEGUIR A ALGUIEN Y QUE ESA PERSONA NO ME SIGA, ENTONCES ES DIRIGIDO. EN CASOS DE PROTEINAS ES NO DIRIGIDO PORQUE AMBAS INTERACTUAN ENTRE ELLAS.

## 5 ALGUNOS PARAMETROS DE LAS REDES

```

In [37]: G=grafoRedComplejosProteicos
        H=grafoRedInteraccionesBinarias
        I=grafoRedLiteratura

```

como hay dos comunidades no conectadas no puede definir una longitud entonces.. definimos la siguiente funcion para obtener la componente gigante y ahi calcularle el diametro y la densidad.

```

In [38]: def get_giant(G):
        Gcc=sorted(nx.connected_component_subgraphs(G), key = len, reverse=True)
        G0=Gcc[0]
        return(G0)

In [39]: def props(G):
        giant_G=get_giant(G)
        return("diameter: %d" % nx.diameter(giant_G), 'density', nx.density(G))

In [40]: props(G), props(H), props(I)

Out[40]: (('diameter: 15', 'density', 0.006899274397150227),
          ('diameter: 14', 'density', 0.0014396951973635397),
          ('diameter: 19', 'density', 0.002481168566775244))

```

## 6 CLUSTERING

### LOCAL

```
In [41]: nx.average_clustering(G),nx.average_clustering(H),nx.average_clustering(I)
```

```
Out[41]: (0.5546360657013013, 0.046194001297365166, 0.2924923005815711)
```

### GLOBAL

```
In [42]: nx.transitivity(G),nx.transitivity(H),nx.transitivity(I)
```

```
Out[42]: (0.6185901626483971, 0.02361415364051535, 0.3461926495315878)
```

## 7 MEAN DEGREE

```
In [43]: #forma rapida y linda
maxdg_G=np.array([j for (i,j) in G.degree]).max()
maxdg_H=np.array([j for (i,j) in H.degree]).max()
maxdg_I=np.array([j for (i,j) in I.degree]).max()

maxdg_G,maxdg_H,maxdg_I
```

```
Out[43]: (127, 91, 40)
```

```
In [44]: mindg_G=np.array([j for (i,j) in G.degree]).min()
mindg_H=np.array([j for (i,j) in H.degree]).min()
mindg_I=np.array([j for (i,j) in I.degree]).min()

mindg_G,mindg_H,mindg_I
```

```
Out[44]: (1, 1, 1)
```

```
In [45]: mv_G=np.mean(np.array([j for (i,j) in G.degree]))
mv_H=np.mean(np.array([j for (i,j) in H.degree]))
mv_I=np.mean(np.array([j for (i,j) in I.degree]))

mv_G,mv_H,mv_I
```

```
Out[45]: (11.183723797780518, 2.9038652130822595, 3.80859375)
```

## 8 RED DIRIGIDA O NO DIRIGIDA

```
In [46]: #PARA VER SI ERA RED DIRIGIDA
#la idea es que si hay interseccion entre set1 y set2 es que se da el enlace nodo1-nodo2
#NO se hace distincion y que el nodo1 interactua con el nodo2 y al reves entonces.. es
#es claramente dirigida. Y la no intersecc es si el array inters tiene length 0.

def direccion(G):
```

```

set1=set(G.edges)
set2 = {(nombre2,nombre1) for nombre1, nombre2 in set1}
inters = set1.intersection(set2)
if len(inters) == 0:
    print("Dirigida")
else:
    print("No dirigida")

```

```

In [47]: direccion(G)
         direccion(H)
         direccion(I)

```

```

Dirigida
No dirigida
No dirigida

```

## 8.1 COMENTARIOS para el grupo

```

In [48]: [(i,j) for (i,j) in H.degree] # este me crea una lista con las tuplas que le digo
         #como me quiero quedar solo con los degrees, me conviene pedirle los j's:
         [(j) for (i,j) in H.degree]
         # lo paso a array y despues le pido el maximo
         asd=np.array([(j) for (i,j) in H.degree])
         asd.max()

```

```

Out[48]: 91

```

## 9 TABLA CON RESULTADOS

```

In [2]: from IPython.display import HTML, display

data = [
    ["", "Red de compl proteicos", "Red de int binarias", "Red de literatura"],
    ["Nodos", 2018, 1622, 1536],
    ["Total enlaces", 2930, 9070, 2925],
    ["Dirigida", "SI", "NO", "NO"],
    ["Grado medio", 11.18, 2.90, 3.81],
    ["Grado max", 127, 91, 40],
    ["Grado min", 1, 1, 1],
    ["Densidad", 0.00690, 0.00144, 0.00248],
    ["Coef. clustering local", 0.555, 0.046, 0.292],
    ["Coef. clustering global", 0.619, 0.024, 0.346],
    ["Diametro", 15, 14, 19]
]

display(HTML(
    '<table><tr>{}</tr></table>'.format(
        '</tr><tr>'.join(

```

```
        '<td>{}</td>'.format('</td><td>'.join(str(_) for _ in row)) for row in data)
    )
))
```

<IPython.core.display.HTML object>