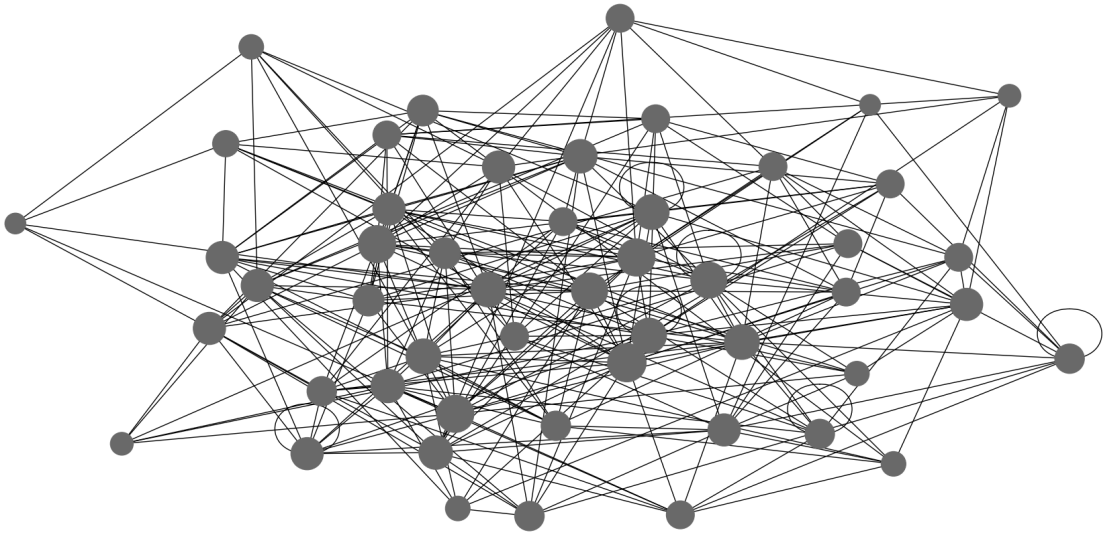
ANÁLISIS DE LAS CARACTERÍSTICAS DE UNA RED DE RELACIONES



Matemática 3

2° Cuatrimestre 2023

Autores: Santiago Bouza, Camila Piraneo

UNSAM

Universidad Nacional de San Martín

1. **Objetivo:**

   El objetivo de este proyecto es analizar ciertas características que surgen en una red de relaciones entre personas, a saber, quién es la persona más o menos popular, las importancias de ciertas amistades sobre otras y la posible diseminación de una enfermedad, por nombrar algunas.

   El proyecto está realizado en el lenguaje Python, a través de la IDE Google Colab, y se emplearon las librerías *random, matplotlib, pandas, networkx* y *re*.

2. **Implementación:**

   i. La red de relaciones se generó a partir de un dataset ficticio creado por nosotros y que toma como parámetro para cuántas personas simulará relaciones (**n**).

Índice	**Persona**	**Amigo**	**Importancia**
0	user1	user2	1
1	user1	user3	3
…	…	…	…
n-1	user”n”	user(1 hasta “n”)	1 hasta 10

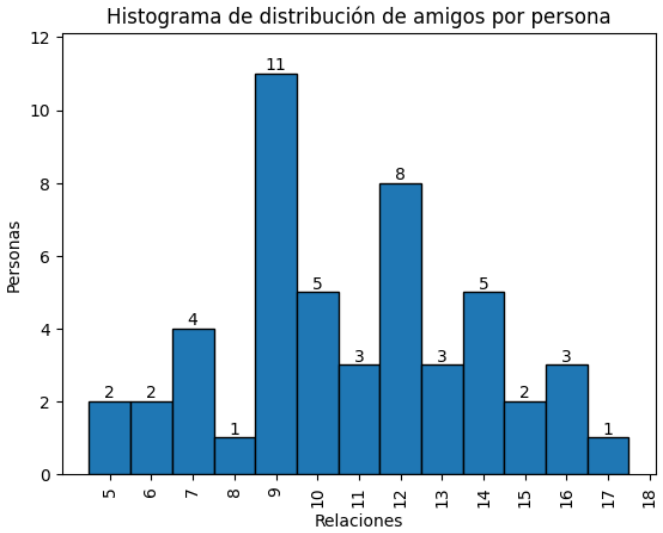
   Para crear este dataset la información de cada usuario se almacenó en un diccionario que contiene el nombre de la persona y una lista de amigos y otra de la importancia de cada uno.

   ii. Una vez creado el dataset se creó dicha red con el método *nx.from_pandas_edgelist* de la librería *networkx*, el cual toma cada fila y establece una relación entre los elementos en ella. El método brinda la opción de agregar el parámetro *edge_attribute*, en donde se especifica que el valor de la columna “importancia” debe ser tomado como atributo de la arista que une a ambas personas.

   Además, una vez diseñada la red se le proporcionó a cada persona una profesión, de una lista de profesiones, con el método *node_attr* y la librería *random*.

3. **Análisis:**

   i. **Histograma**: visualización de la distribución de amigos por persona.



El eje horizontal representa el número de amigos, y el eje vertical representa la cantidad de personas con ese número de amigos.

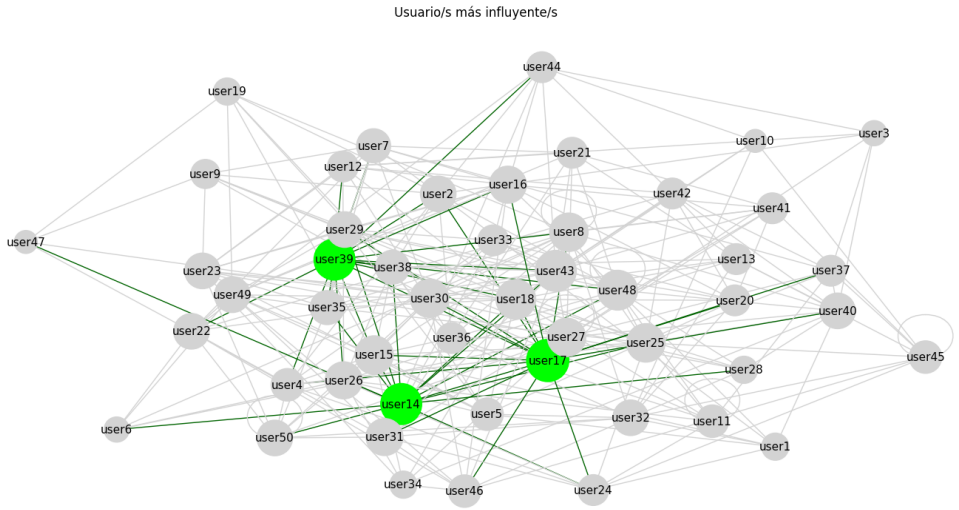
Del gráfico se desprende que:

- Muestra una distribución asimétrica. Esto quiere decir que no hay una distribución normal y las columnas suben y bajan. Esto se corresponde con la mayoría de las relaciones sociales reales.
- Las columnas en el medio muestran los valores más comunes. La mayoría de las personas tienen entre 8 y 11 amigos.
- En los extremos están los valores mínimos y máximos. Hay pocas personas que tienen solo 2 o hasta 17 amigos.

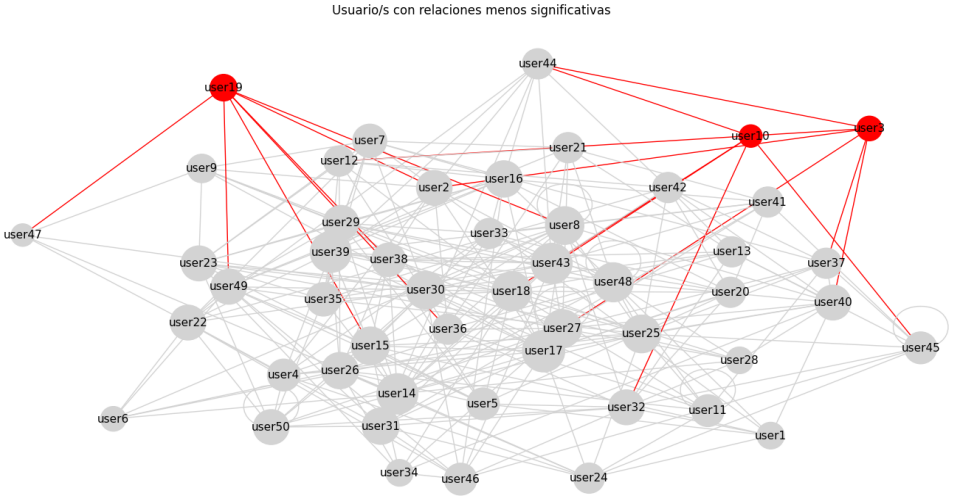
ii. **Persona más o menos influyente**: implementado con una función que toma como parámetro si se busca a la persona más o menos influyente, a cuántas personas que cumplan lo anterior y si se debe visualizar en un grafo de redes o no. Dicha visualización consiste en colorear a los nodos y sus aristas de un color distinto al resto.

```
most_less_influencias("most",         3,                  True)
#                      ^              ^                    ^
#       "most" -> más influyentes     Cuantos usuarios     Opción de visualizar
#       "less" -> menos ""            a mostrar
```

La anterior función arrojará este gráfico:



Si el primer parámetro fuera *less*, estaremos buscando a la persona menos influyente y el gráfico sería así:



iii. **Profesiones del nodo más o menos influyente**: implementado con una función que toma como parámetro si se busca a la persona más o menos influyente y a cuántas personas que cumplan lo anterior mostrar.

```
most_less_profesiones("most",          4)
#                      ^               ^
#        "most" -> más influyentes     Cuantos usuarios
#        "less" -> menos ""            a mostrar
```

La anterior función arrojará:

```
El usuario user17 tiene 17 aristas y su profesión es artista.

El usuario user39 tiene 16 aristas y su profesión es arquitecto.

El usuario user14 tiene 16 aristas y su profesión es científico.

El usuario user43 tiene 16 aristas y su profesión es escritor.
```

Si el primer parámetro fuera *less*, estaremos buscando la profesión de la persona menos influyente:

```
El usuario user10 tiene 5 aristas y su profesión es médico.

El usuario user47 tiene 5 aristas y su profesión es científico.

El usuario user3 tiene 6 aristas y su profesión es músico.

El usuario user6 tiene 6 aristas y su profesión es escritor.
```

iv. **Profesión más prevalente entre las personas más o menos influyentes**: implementado con una función que toma como parámetro si se busca a la persona más o menos influyente y a cuántas personas que cumplan lo anterior mostrar.

```
count_profesiones("less",           10)
#                   ^             ^
#       "most" -> más influyentes   Cuantos usuarios
#       "less" -> menos ""          a mostrar
```

La anterior función imprime en pantalla lo siguiente:

```
La profesión más prevalente entre los 10 nodos menos populares es escritor.
```

Si el primer argumento fuera *most*:

```
La profesión más prevalente entre los 10 nodos más populares es ingeniero.
```

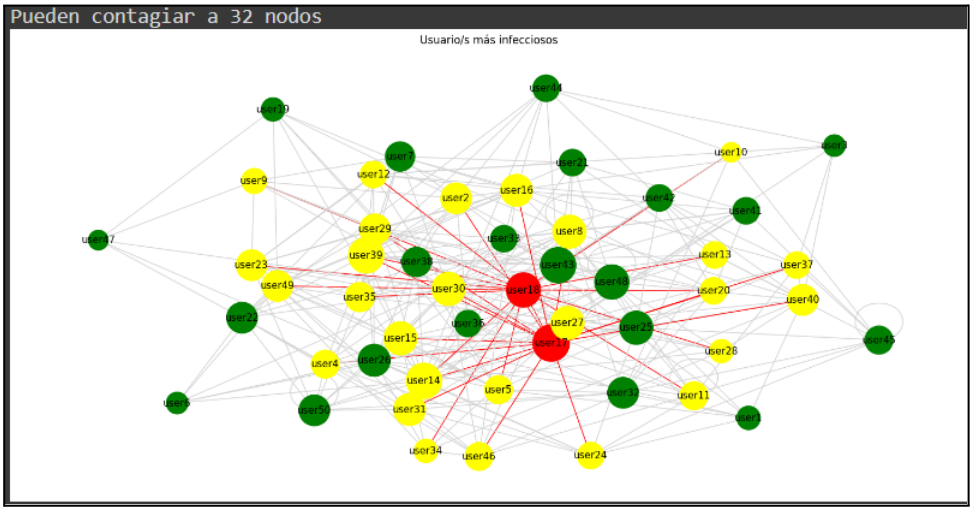
v. **Diseminación de una enfermedad**: implementado con una función que toma como parámetro si se busca a la persona más o menos infecciosa, a cuántas personas que cumplan lo anterior mostrar y qué

métrica usar. Los nodos y aristas que cumplan dicha condición se grafican de un color distinto al resto.

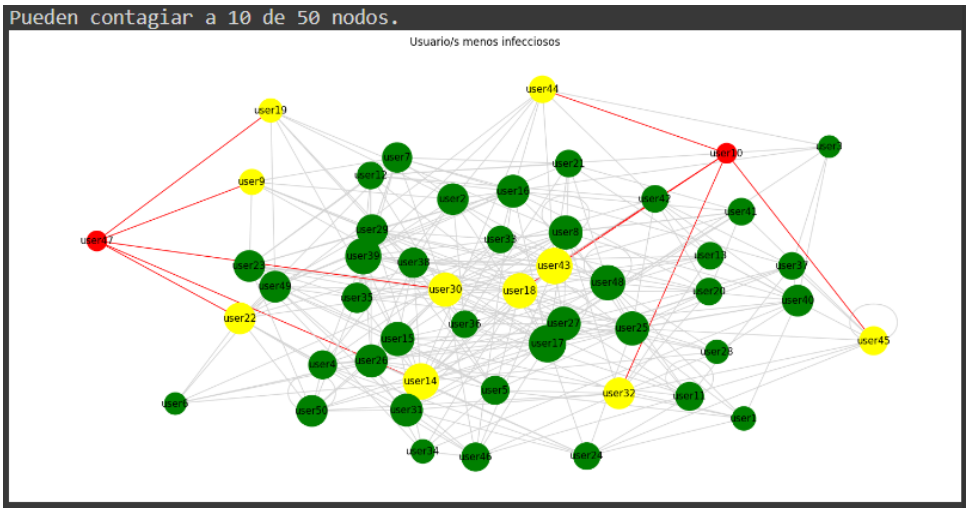
i. Métrica 1. Producto de las métricas, nativas de la librería *networkx*, *betweeneess_centrality* y *closeness_centrality*.

```
diseminacion_enfermedad("most",       2,                False)
```

La anterior función crea a este grafo e imprime en pantalla:



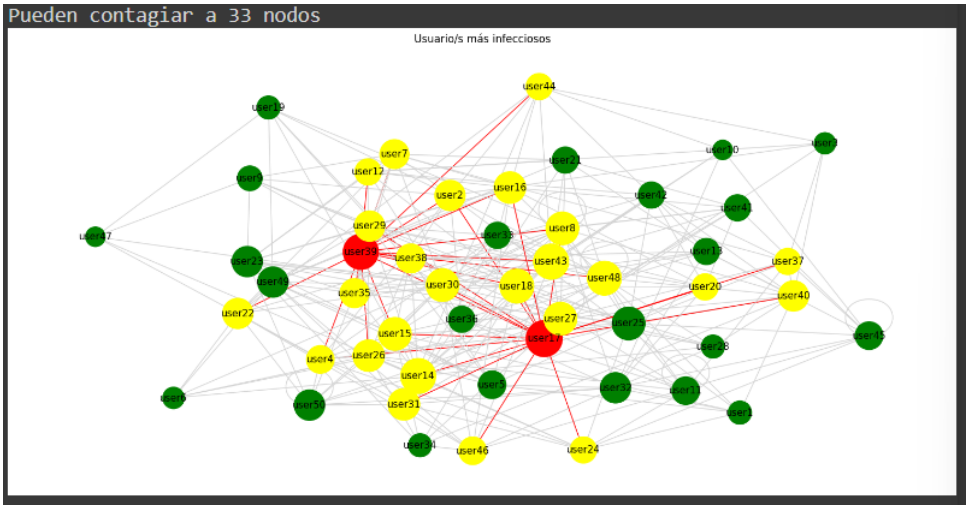
Si el primer argumento fuera *less*:



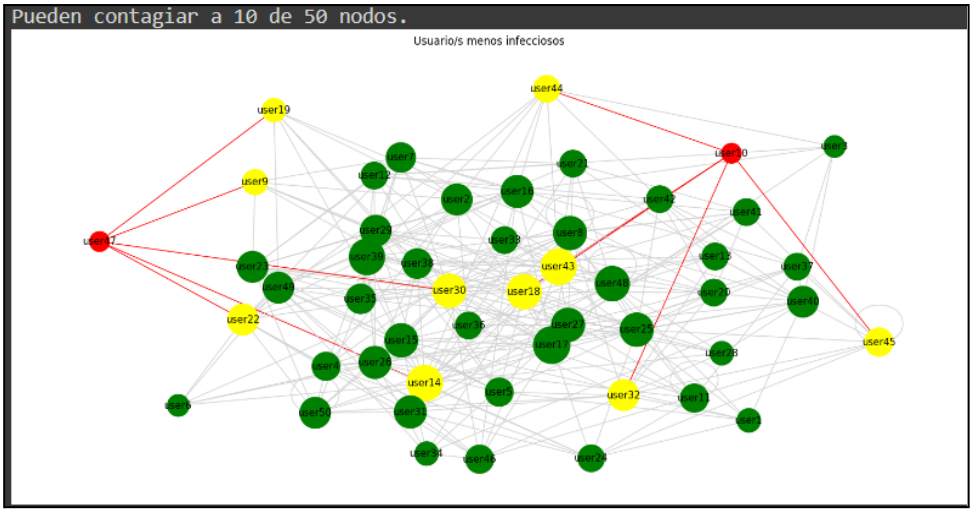
ii. Métrica 2. Métrica nativa de la librería *networkx*, *communicability_betweeneess_centrality.*

```
diseminacion_enfermedad("most",      2,                  True)
```

La anterior función crea a este grafo e imprime en pantalla:



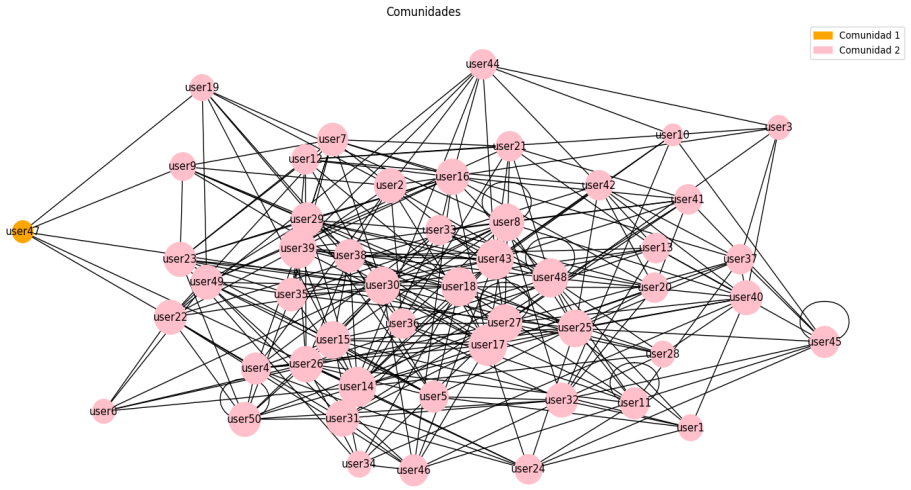
Si el primer argumento fuera *less*:



vi. **Comunidades**: implementado con una función que toma como parámetro qué métrica usar y cuántas comunidades visualizar.

```
algoritmo_comunidades(False,None)
```

El parámetro *False* le indica a la función que utilice el algoritmo nativo de *networkx* “Girvan-Newman”. Dicho algoritmo no permite determinar cuántas comunidades se desean encontrar, esto se indica con el parámetro *None*. En pantalla se imprime lo siguiente:



Observar que al intentar interpretar relaciones dentro de un dataframe ficticio, el algoritmo no puede encontrar delimitaciones claras para separar a las personas en comunidades. Toma por default el nodo más alejado.

Si el primer parámetro fuera *True* y el segundo *2*, esto le indica a la función que utilice al algoritmo de “comunidades fluidas” para hallar a dos comunidades.

Cada comunidad (dos en este caso) está representada como un tipo de líquido. Cada una comienza en un nodo aleatorio del grafo y en cada paso va sumando (o no) a su vecino a la comunidad. Por este motivo en los extremos del grafo predomina un color y en el centro de él no. Así se vería en pantalla: