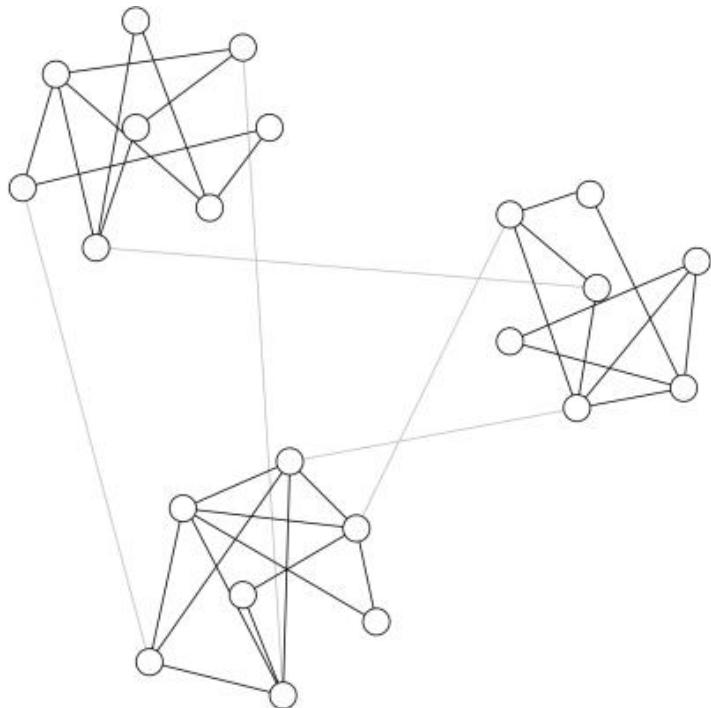


# Detección de comunidades y enriquecimiento funcional

# Comunidades

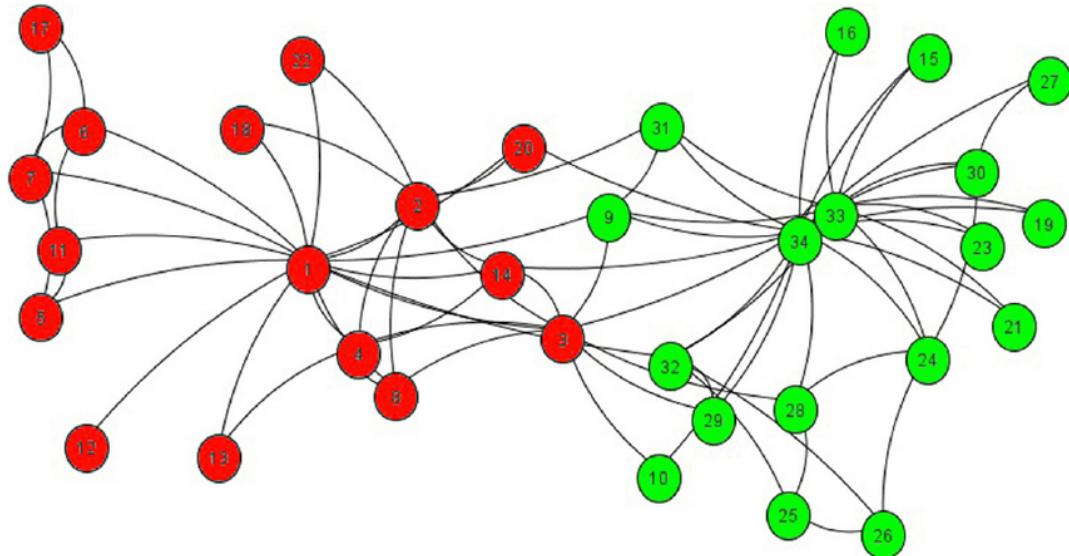
- Concepto
- Algoritmos de detección
- Medidas
- Enriquecimiento

# Comunidades o módulos



<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC122977/>

# Comunidades o módulos



## Partición de la red

Número de enlaces **DENTRO** de la comunidad

MAYOR que

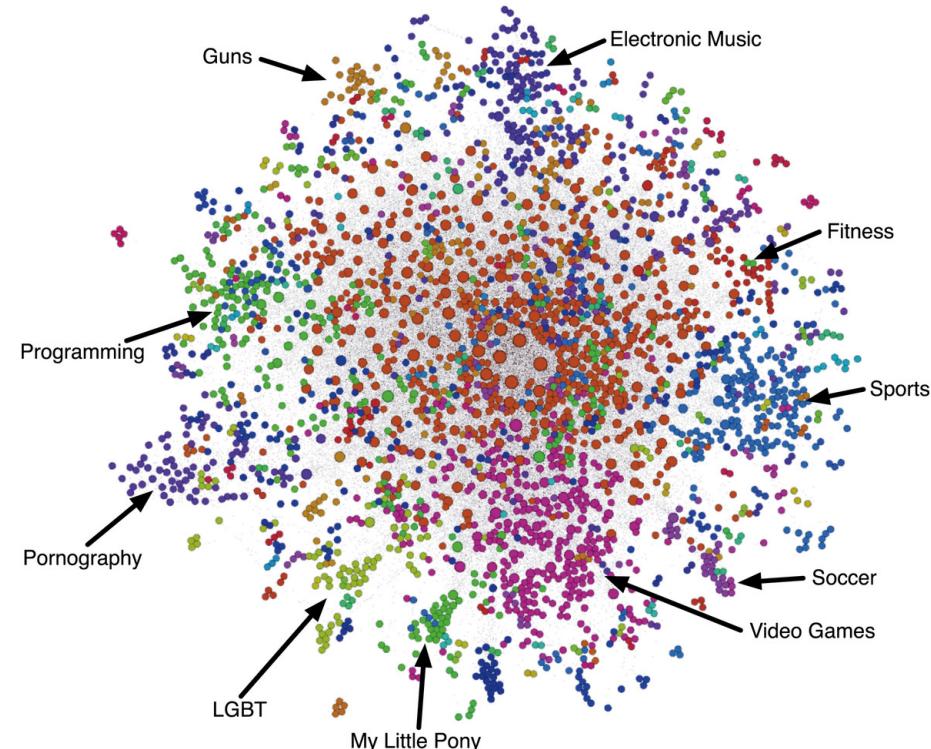
Número de enlaces **FUERA** de la comunidad

# Comunidades o módulos

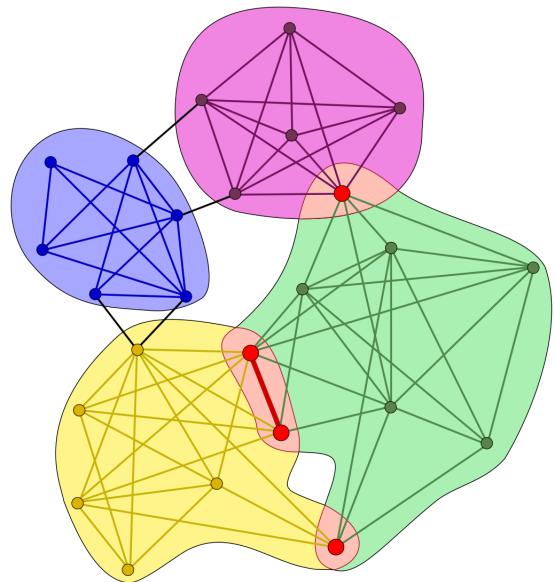
Cada comunidad o módulo de un grafo  $G$  es un subgrafo de  $G$ ;

Compuesto por

- un subconjunto de nodos del conjunto de nodos de  $G$
- un subconjunto de enlaces del conjunto de enlaces de  $G$



# Comunidades o módulos



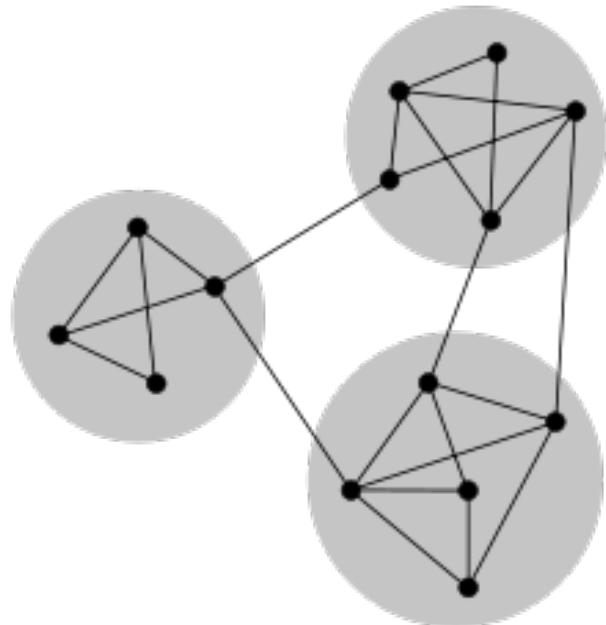
La estructura de comunidades  
puede tener comunidades

**CON**

**o**

traslape

**SIN**



# Preguntas

- 1) ¿Todas las redes tienen una estructura de comunidades?
- 2) ¿Existe una única partición de una red?
- 3) ¿Un componente es lo mismo que una comunidad?

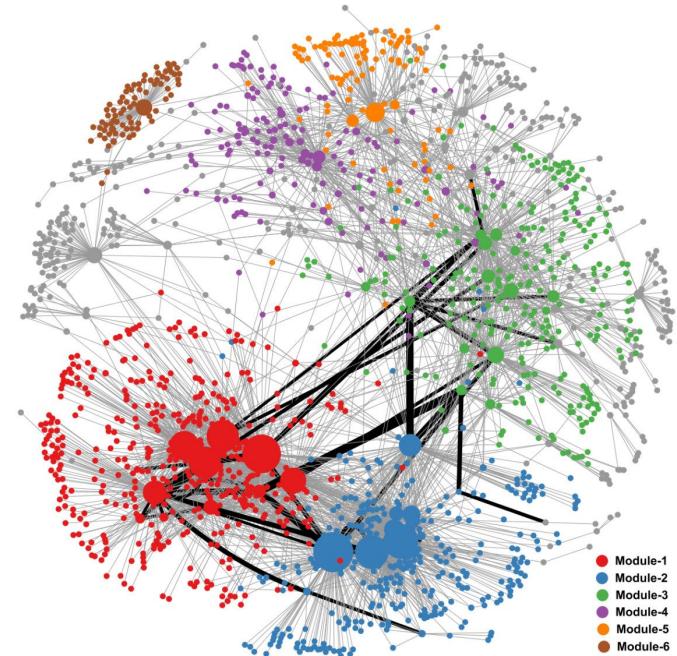
# Algoritmos de detección

# Algoritmos de detección de comunidades

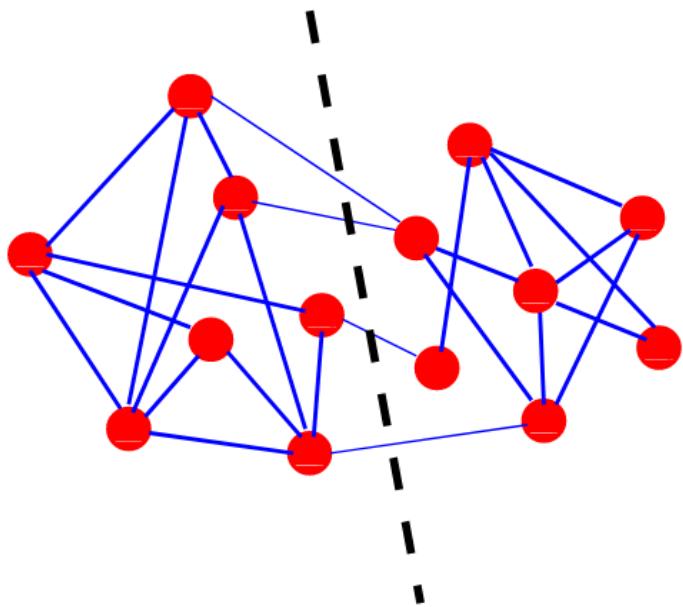


# Tipos de algoritmos

- Métodos tradicionales: particionar grafo
- Optimización de modularidad
- Propiedades espectrales de la matriz de adyacencia
- Dinámicos: Caminantes aleatorios
- Otros



# Particionar grafo minimizando número de cortes



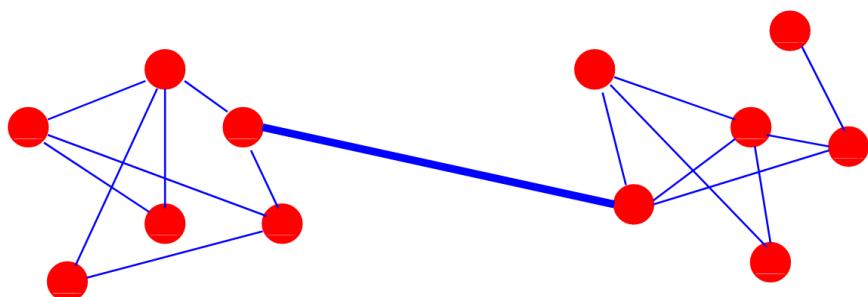
Problema: partir el grafo en  $k$  grupos de tamaño predeterminado.

Ejemplo: 2 grupos con  $n/2$  nodos

Parámetros: # grupos, tamaño grupos

Optimizar: Mínimo número de enlaces que retirar

# Algoritmos divisivos: Girvan y Newman

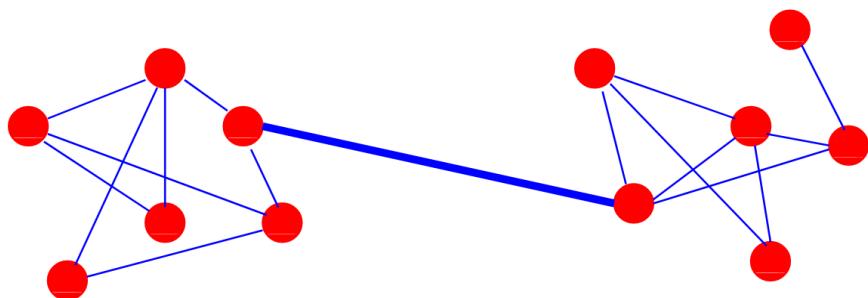


Iterativamente, removamos el enlace que contribuye más a la conexión de la red:

## **Mayor Edge betweenness**

- 1) Calcular edge betweenness para todos los enlaces
- 2) Retirar el enlace con mayor Edge betweenness
- 3) Agrupar (contar componentes conectados)

# Algoritmos divisivos: Girvan y Newman



Iterativamente, removamos el enlace que contribuye más a la conexión de la red:

## **Mayor Edge betweenness**

- 1) Calcular edge betweenness para todos los enlaces
- 2) Retirar el enlace con mayor Edge betweenness
- 3) Agrupar (contar componentes conectados)

# Modularidad

$$Q = \frac{1}{2m} \sum_{vw} \left[ A_{vw} - \frac{k_v k_w}{2m} \right] \delta(c_v, c_w)$$

# Modularidad

Valor de la matriz de adyacencia para los dos nodos

$$Q = \frac{1}{2m} \sum_{vw} \left[ A_{vw} - \frac{k_v k_w}{2m} \delta(c_v, c_w) \right]$$

# aristas

índices de nodos

Grado de los nodos

$$\frac{k_v k_w}{2m}$$

Esta es la probabilidad de que v y w estén conectados si las conexiones fueran al azar

Función delta:  
1 si  $C_v = C_w$   
0 de otra forma

Comunidad a la que pertenecen los nodos

# Modularidad

$$Q = \frac{1}{2m} \sum_{vw} \left[ A_{vw} - \frac{k_v k_w}{2m} \right] \delta(c_v, c_w)$$

Podemos usarla para optimizar.  
Dejamos de partir cuando nuestra Q sea  
máxima.

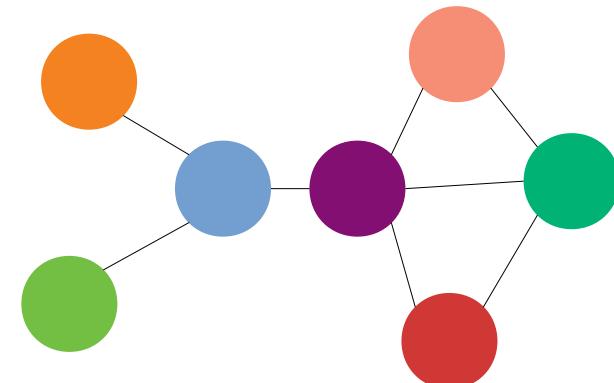
# Métodos basados en modularidad

Ejemplo: Fast greedy

- 1) Poner todos los nodos en su propia comunidad
- 2) Calcular  $\Delta Q_{i,j}$
- 3) Unir las comunidades entre las cuales se encuentra el máximo  $\Delta Q_{i,j}$
- 4) Repetir hasta que se tenga solo una comunidad
- 5) Escoger la partición con  $Q$  máxima

Si  $i,j$  están conectados

$$\Delta Q_{ij} = \begin{cases} 1/2m - k_i k_j / (2m)^2 & \text{Si } i,j \text{ están conectados} \\ 0 & \text{Otherwise} \end{cases}$$



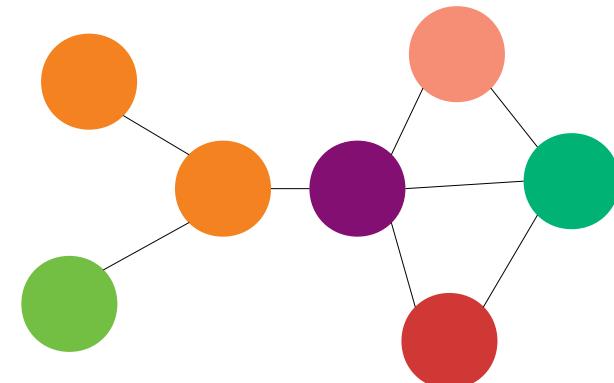
# Métodos basados en modularidad

Ejemplo: Fast greedy

- 1) Poner todos los nodos en su propia comunidad
- 2) Calcular  $\Delta Q_{i,j}$
- 3) Unir las comunidades entre las cuales se encuentra el máximo  $\Delta Q_{i,j}$
- 4) Repetir hasta que se tenga solo una comunidad
- 5) Escoger la partición con  $Q$  máxima

Si  $i,j$  están conectados

$$\Delta Q_{ij} = \begin{cases} 1/2m - k_i k_j / (2m)^2 & \text{Si } i,j \text{ están conectados} \\ 0 & \text{Otherwise} \end{cases}$$



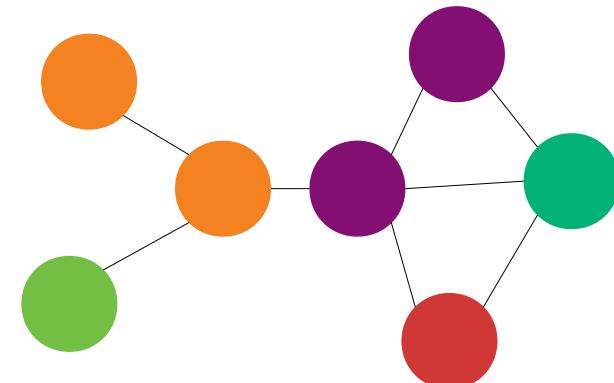
# Métodos basados en modularidad

Ejemplo: Fast greedy

- 1) Poner todos los nodos en su propia comunidad
- 2) Calcular  $\Delta Q_{i,j}$
- 3) Unir las comunidades entre las cuales se encuentra el máximo  $\Delta Q_{i,j}$
- 4) Repetir hasta que se tenga solo una comunidad
- 5) Escoger la partición con  $Q$  máxima

Si  $i,j$  están conectados

$$\Delta Q_{ij} = \begin{cases} 1/2m - k_i k_j / (2m)^2 & \text{Si } i,j \text{ están conectados} \\ 0 & \text{Otherwise} \end{cases}$$



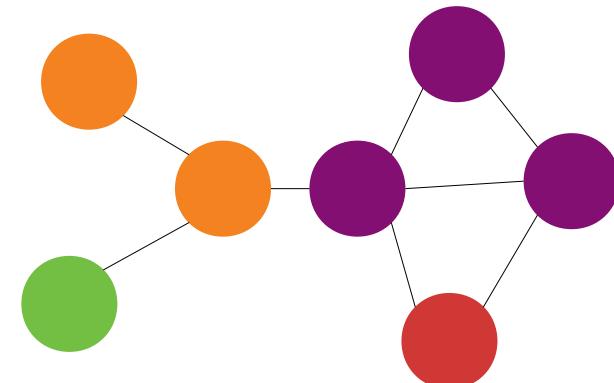
# Métodos basados en modularidad

Ejemplo: Fast greedy

- 1) Poner todos los nodos en su propia comunidad
- 2) Calcular  $\Delta Q_{i,j}$
- 3) Unir las comunidades entre las cuales se encuentra el máximo  $\Delta Q_{i,j}$
- 4) Repetir hasta que se tenga solo una comunidad
- 5) Escoger la partición con  $Q$  máxima

Si  $i,j$  están conectados

$$\Delta Q_{ij} = \begin{cases} 1/2m - k_i k_j / (2m)^2 & \text{Si } i,j \text{ están conectados} \\ 0 & \text{Otherwise} \end{cases}$$



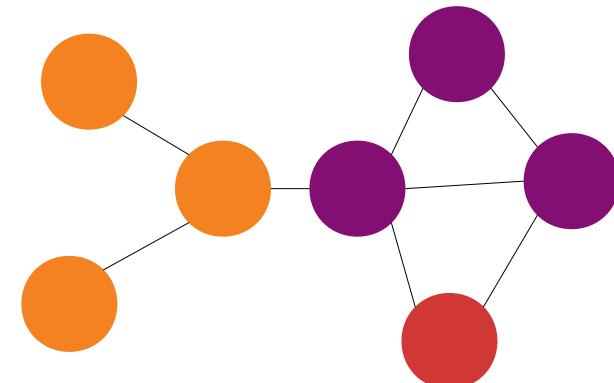
# Métodos basados en modularidad

Ejemplo: Fast greedy

- 1) Poner todos los nodos en su propia comunidad
- 2) Calcular  $\Delta Q_{i,j}$
- 3) Unir las comunidades entre las cuales se encuentra el máximo  $\Delta Q_{i,j}$
- 4) Repetir hasta que se tenga solo una comunidad
- 5) Escoger la partición con  $Q$  máxima

Si  $i,j$  están conectados

$$\Delta Q_{ij} = \begin{cases} 1/2m - k_i k_j / (2m)^2 & \text{Si } i,j \text{ están conectados} \\ 0 & \text{Otherwise} \end{cases}$$



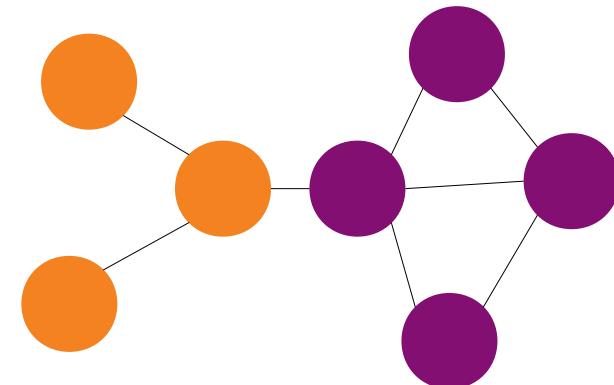
# Métodos basados en modularidad

Ejemplo: Fast greedy

- 1) Poner todos los nodos en su propia comunidad
- 2) Calcular  $\Delta Q_{i,j}$
- 3) Unir las comunidades entre las cuales se encuentra el máximo  $\Delta Q_{i,j}$
- 4) Repetir hasta que se tenga solo una comunidad
- 5) Escoger la partición con  $Q$  máxima

Si  $i,j$  están conectados

$$\Delta Q_{ij} = \begin{cases} 1/2m - k_i k_j / (2m)^2 & \text{Si } i,j \text{ están conectados} \\ 0 & \text{Otherwise} \end{cases}$$



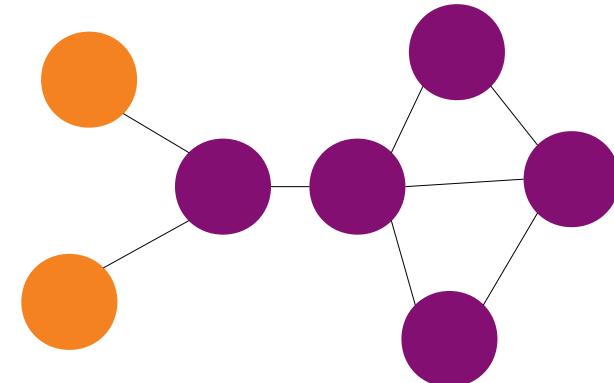
# Métodos basados en modularidad

Ejemplo: Fast greedy

- 1) Poner todos los nodos en su propia comunidad
- 2) Calcular  $\Delta Q_{i,j}$
- 3) Unir las comunidades entre las cuales se encuentra el máximo  $\Delta Q_{i,j}$
- 4) Repetir hasta que se tenga solo una comunidad
- 5) Escoger la partición con  $Q$  máxima

Si  $i,j$  están conectados

$$\Delta Q_{ij} = \begin{cases} 1/2m - k_i k_j / (2m)^2 & \text{Si } i,j \text{ están conectados} \\ 0 & \text{Otherwise} \end{cases}$$



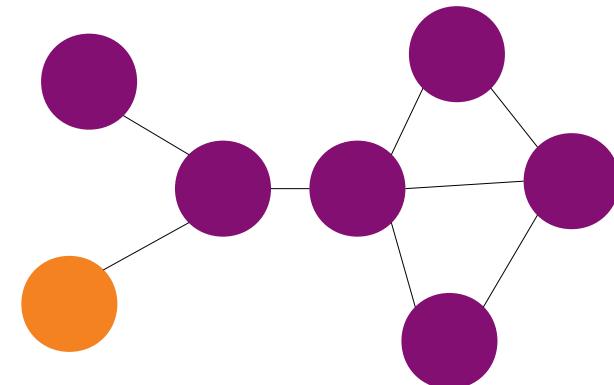
# Métodos basados en modularidad

Ejemplo: Fast greedy

- 1) Poner todos los nodos en su propia comunidad
- 2) Calcular  $\Delta Q_{i,j}$
- 3) Unir las comunidades entre las cuales se encuentra el máximo  $\Delta Q_{i,j}$
- 4) Repetir hasta que se tenga solo una comunidad
- 5) Escoger la partición con  $Q$  máxima

Si  $i,j$  están conectados

$$\Delta Q_{ij} = \begin{cases} 1/2m - k_i k_j / (2m)^2 & \text{Si } i,j \text{ están conectados} \\ 0 & \text{Otherwise} \end{cases}$$



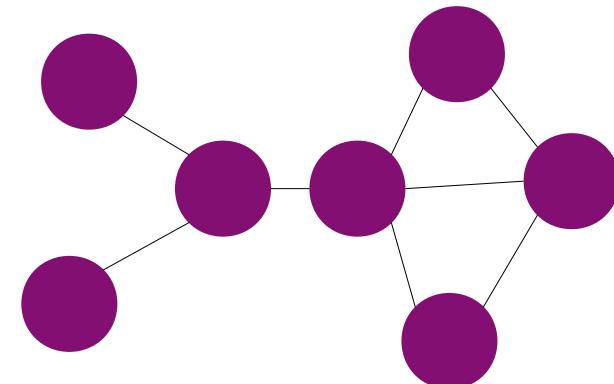
# Métodos basados en modularidad

Ejemplo: Fast greedy

- 1) Poner todos los nodos en su propia comunidad
- 2) Calcular  $\Delta Q_{i,j}$
- 3) Unir las comunidades entre las cuales se encuentra el máximo  $\Delta Q_{i,j}$
- 4) Repetir hasta que se tenga solo una comunidad
- 5) Escoger la partición con  $Q$  máxima

Si  $i,j$  están conectados

$$\Delta Q_{ij} = \begin{cases} 1/2m - k_i k_j / (2m)^2 & \text{Si } i,j \text{ están conectados} \\ 0 & \text{Otherwise} \end{cases}$$



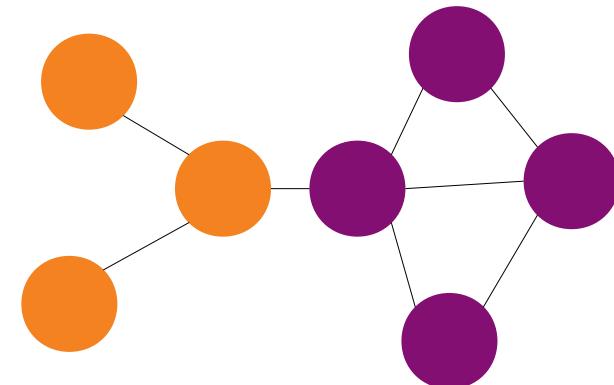
# Métodos basados en modularidad

Ejemplo: Fast greedy

- 1) Poner todos los nodos en su propia comunidad
- 2) Calcular  $\Delta Q_{i,j}$
- 3) Unir las comunidades entre las cuales se encuentra el máximo  $\Delta Q_{i,j}$
- 4) Repetir hasta que se tenga solo una comunidad
- 5) Escoger la partición con  $Q$  máxima

Si  $i,j$  están conectados

$$\Delta Q_{ij} = \begin{cases} 1/2m - k_i k_j / (2m)^2 & \text{Si } i,j \text{ están conectados} \\ 0 & \text{Otherwise} \end{cases}$$



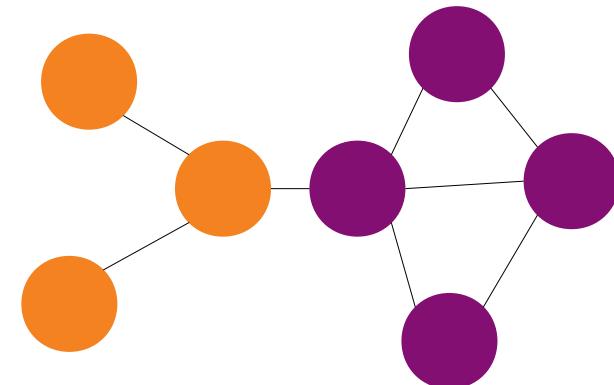
# Métodos basados en modularidad

Ejemplo: Fast greedy

- 1) Poner todos los nodos en su propia comunidad
- 2) Calcular  $\Delta Q_{i,j}$
- 3) Unir las comunidades entre las cuales se encuentra el máximo  $\Delta Q_{i,j}$
- 4) Repetir hasta que se tenga solo una comunidad
- 5) Escoger la partición con  $Q$  máxima

Si  $i,j$  están conectados

$$\Delta Q_{ij} = \begin{cases} 1/2m - k_i k_j / (2m)^2 & \text{Si } i,j \text{ están conectados} \\ 0 & \text{Otherwise} \end{cases}$$



# Métodos basados en modularidad

Fast greedy

Simulated Annealing

**Louvain**

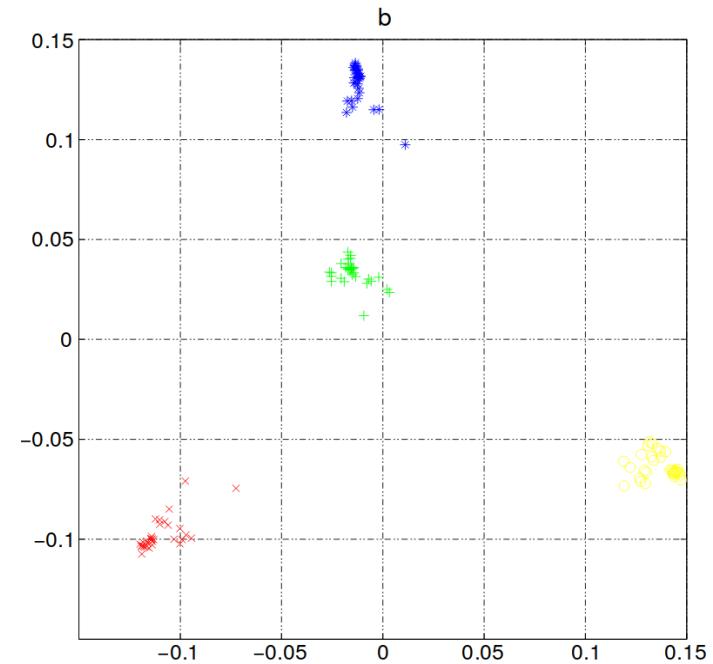
*Mención especial:*  
MCode (optimiza  
clustering coefficient  
de subgrafos)

# Métodos espectrales

**Usar propiedades espectrales de la matriz de adyacencia, o de matrices derivadas de esta (ej, matriz laplaciana)**

Ejemplo:

- 1) Obtener matriz laplaciana ( $L = D - A$ )
- 2) Calcular eigenvectores
- 3) posicionar cada nodo  $i$  en el espacio; las coordenadas serán el  $i$ -ésimo valor de los eigenvectores  $1:M$  (mientras más  $M$ , mejor separación)

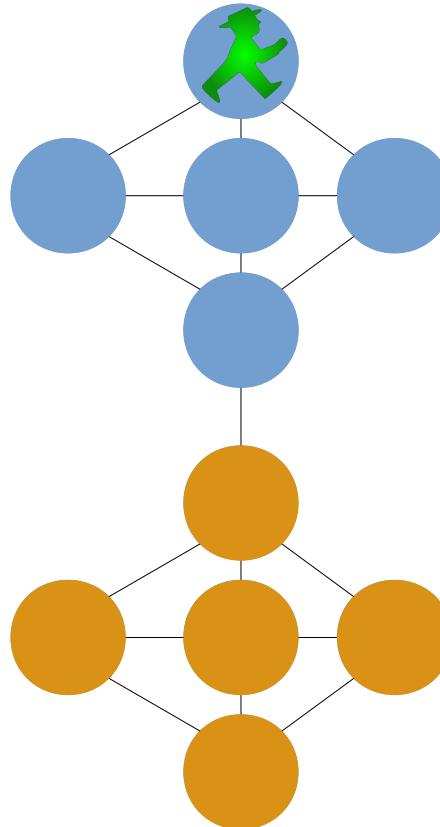


# Métodos con caminantes aleatorios

Caminante aleatorio: camina sobre la red... aleatoriamente

Fundamento: el caminante tarda en salir de las comunidades porque hay más caminos por dentro que por fuera

Optimizar: Descriptores de la caminata



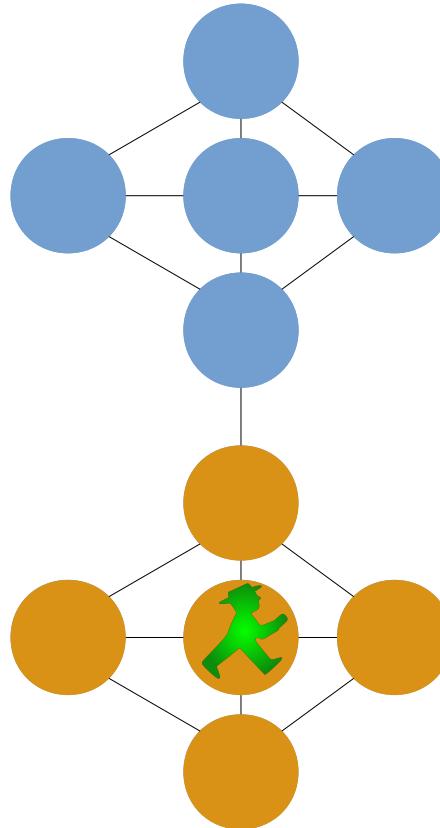
# Métodos con caminantes aleatorios

Ejemplos:

Infomap

Walktrap

Netwalk



# Otros

- Modelos de spin
- Sincronización
- Modelos de bloques estocásticos
- Etc.
- Etc.
- Etc.



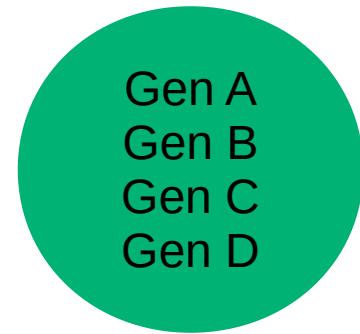
# ¿Cuál uso?

- Existen métricas para comparar las particiones obtenidas por diferentes algoritmos
- No existen “estándares de oro” para comunidades en redes biológicas
- Entonces.... ???
- Probar varios, analizar interpretación biológica, coincidencia... no hay una respuesta sencilla

# Enriquecimiento

# Enriquecimiento

Considere una función biológica F  
asociada a un conjunto de  
moléculas biológicos (genes,  
proteínas)

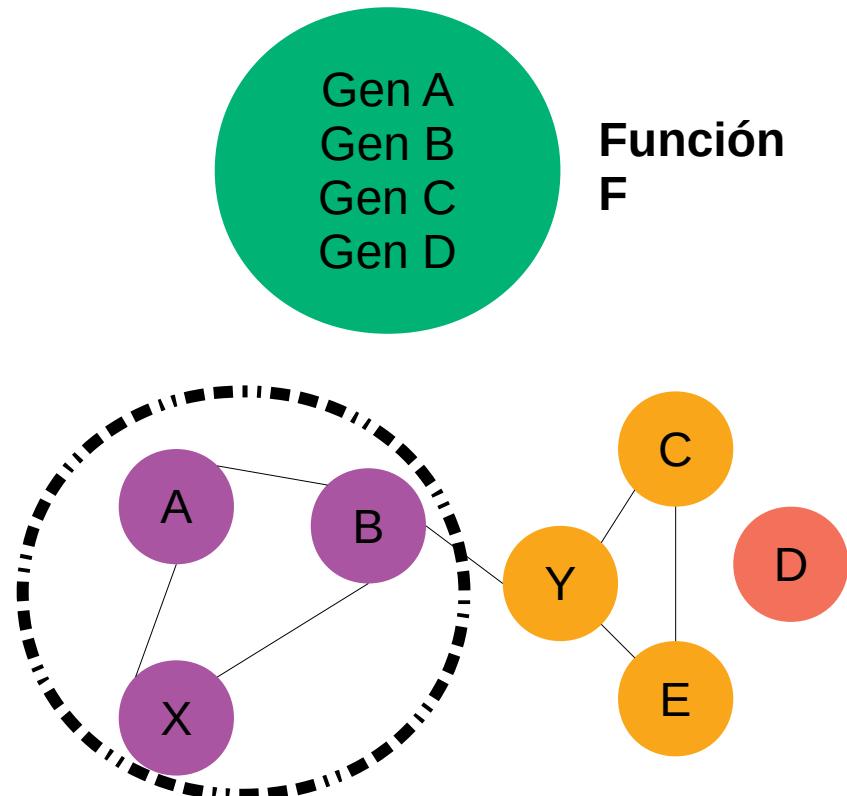


**Función  
F**

# Enriquecimiento

Considere una función biológica F  
asociada a un conjunto de  
moléculas biológicas (genes,  
proteínas)

Considere un conjunto de  
moléculas biológicas que forman  
una comunidad en una red

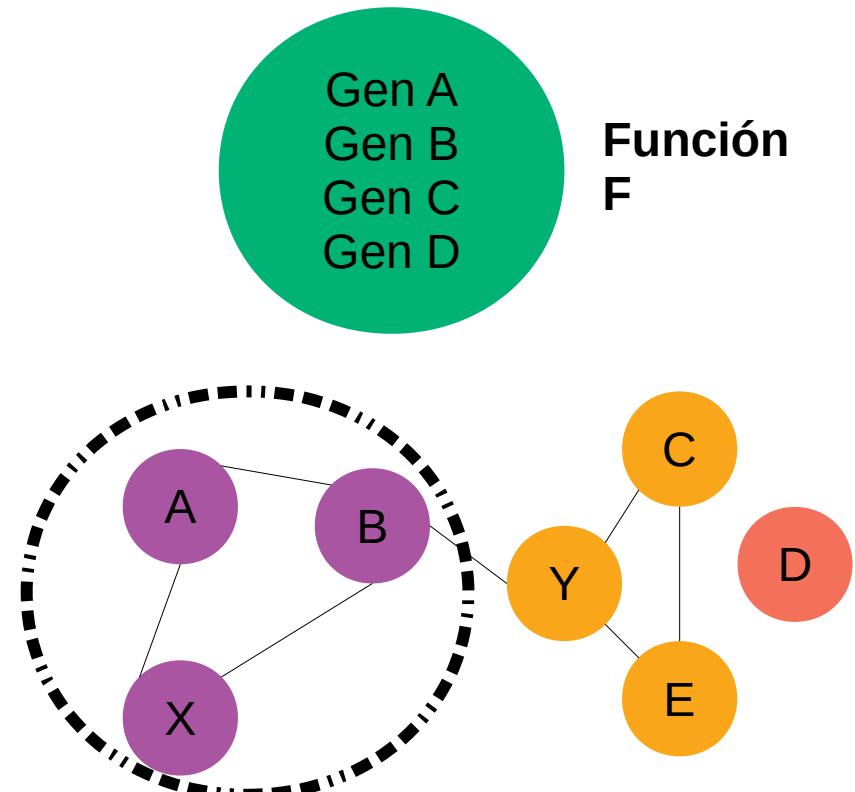


# Enriquecimiento

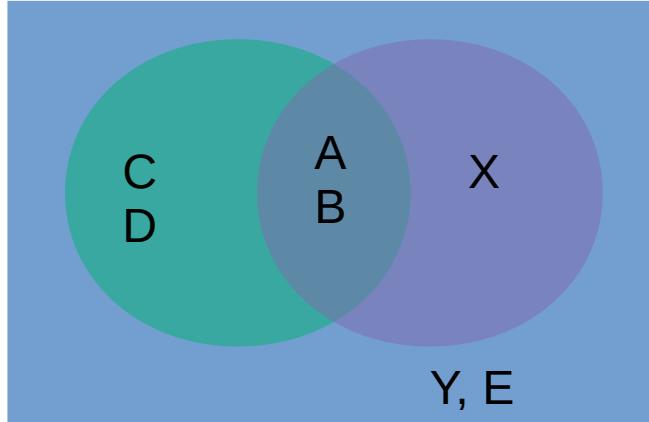
Considere una función biológica F  
asociada a un conjunto de  
moléculas biológicas (genes,  
proteínas)

Considere un conjunto de  
moléculas biológicas que forman  
una comunidad en una red

¿Está asociada la comunidad a la  
función F?



# Over-representation analysis

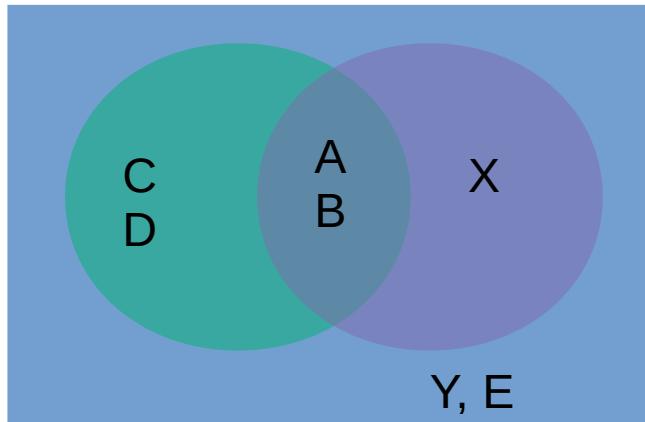


$N$  = genes en la red = 7  
 $n$  = genes en comm = 3  
 $K$  = genes en F = 4  
 $k$  = intersect(F, comm) = 2

$$\Pr(X = k) = \frac{\binom{K}{k} \binom{N-K}{n-k}}{\binom{N}{n}}$$

	En comunidad	NO en comunidad	TOTAL
en Función	2	2	4
NO en función	1	2	3
TOTAL	3	4	7

# Prueba exacta de Fisher / Hipergeométrica



$N$  = genes en la red = 7  
 $n$  = genes en comm = 3  
 $K$  = genes en F = 4  
 $k$  = intersect(F, comm) = 2

$$\Pr(X = k) = \frac{\binom{K}{k} \binom{N-K}{n-k}}{\binom{N}{n}}$$

	En comunidad	NO en comunidad	TOTAL
en Función	2	2	4
NO en función	1	2	3
TOTAL	3	4	7

# Herramientas para enrichment

<http://www.webgestalt.org/>

<https://david.ncifcrf.gov/>

HTSanalyzer (R)

Etc. etc. etc.



**DAVID Bioinformatics Resources 6.7**  
National Institute of Allergy and Infectious Diseases (NIAID), NIH

Welcome to DAVID 6.7  
2003 - 2010

Recommending: A paper published in *Nature Protocols* describes step-by-step procedure to use DAVID!

**Shortcut to DAVID Tools**

- Functional Annotation
- Gene-annotation enrichment analysis, functional annotation clustering , BioCarta & KEGG pathway mapping, gene-disease association, homologous match, 3D translation, literature match and 2D
- Gene Functional Classification
- Gene ID Conversion
- Gene Name Batch Viewer

The Database for Annotation, Visualization and Integrated Discovery (DAVID) v6.7 is an [update to the sixth version](#) of our original web-accessible programs. DAVID now provides a comprehensive set of functional annotation tools for investigators to understand the biological meaning behind large list of genes. For any given gene list, DAVID tools are able to:

- Identify enriched biological themes, particularly GO terms
- Discover enriched functional-related gene groups
- Cluster redundant annotation terms
- Visualize genes on BioCarta & KEGG pathway maps
- Display related many-genes-to-many-terms on 2-D view.
- Search for other functionally related genes not in the list

**What's Important in DAVID?**

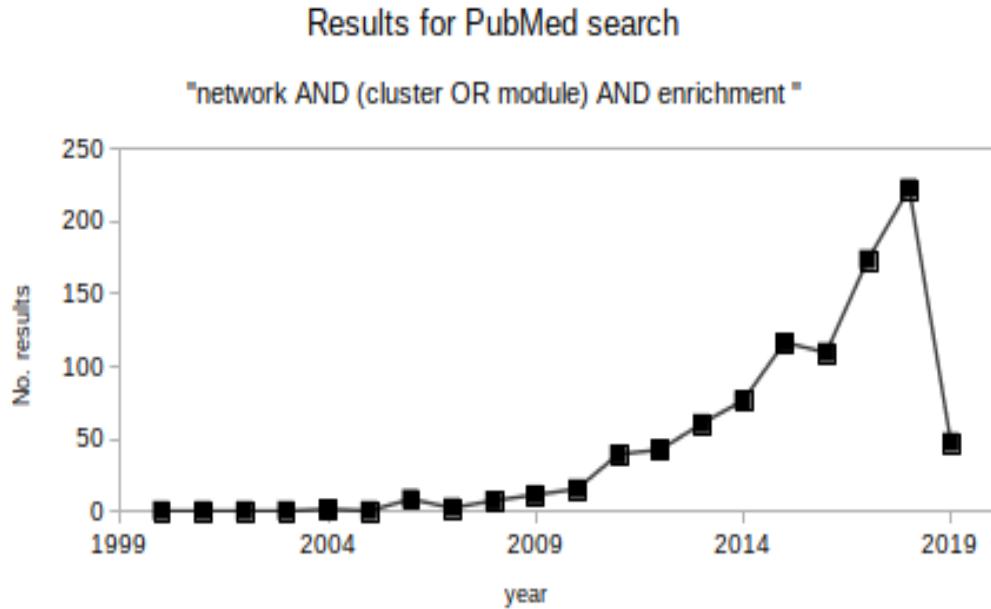
- Current (v 6.7) release note
- New requirement to cite DAVID IDs of Affy Exon and Gene arrays supported
- Novel Classification Algorithms
- Pre-built Affymetrix and Illumina backgrounds
- User's customized gene background
- Enhanced calculating speed

**Statistics of DAVID**

DAVID Citations per year (cumulative)  
Based on Google Scholar  
Updated in Jan, 2010

# Nota

Sacarle comunidades a una red biológica y enriquecerlas es relativamente **FÁCIL.**

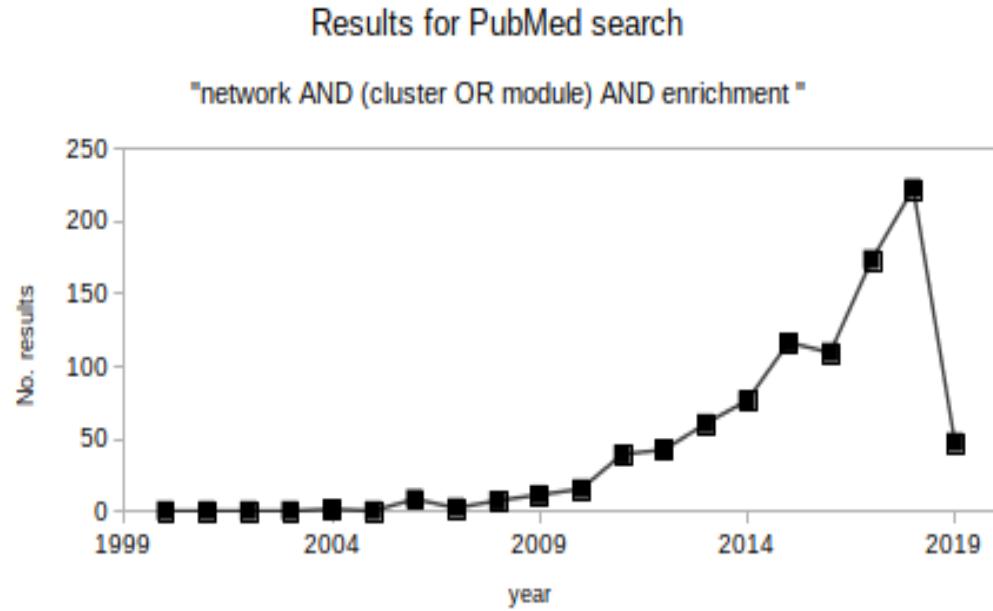


# Nota

Sacarle comunidades a una red biológica y enriquecerlas es relativamente **FÁCIL.**

Darle una interpretación **NO ES FÁCIL.**

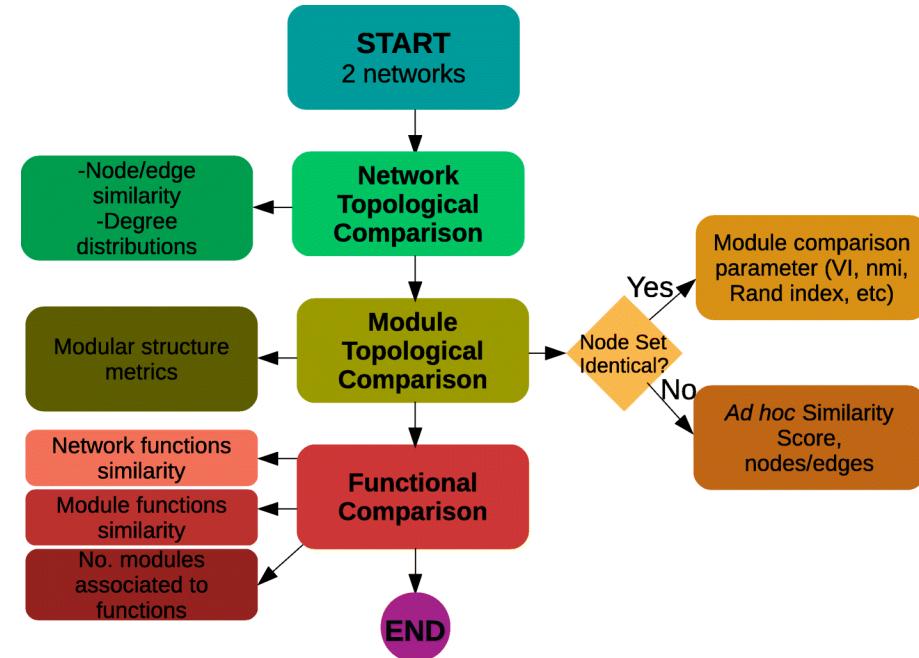
Hay que considerar todos los supuestos que llevaron a la construcción



# Nota

Comparar el **enriquecimiento** funcional de dos (o más) **redes** que representan diferentes **condiciones biológicas** ...

**NO ES TRIVIAL**



# Ejercicio

Usando una red de genes o proteínas

- 1) Detecte si su red tiene estructura modular usando un algoritmo de su elección
- 2) Realice un análisis de enriquecimiento para las comunidades de su red (máximo 5)
  - Considere valores de  $p < 0.05$
- 3) Repita con otro algoritmo y compare:  
¿tiene el mismo número de comunidades?  
¿Encontró las mismas funciones enriquecidas?

# Bibliografía

<https://arxiv.org/pdf/0906.0612.pdf>

<https://arxiv.org/pdf/cond-mat/0408187v2.pdf>

<https://appliednetsci.springeropen.com/articles/10.1007/s41109-018-0109-9>

<https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-4-2#Sec1>