Análisis estadístico de datos de redes con R

Workshop on Big Data and Statistics

Pedro Galeano
pedro.galeano@uc3m.es

Departamento de Estadística
Instituto en Big Data Financiero UC3M - Banco Santander

Universidad
Carlos III de Madrid
www.uc3m.es

Introducción

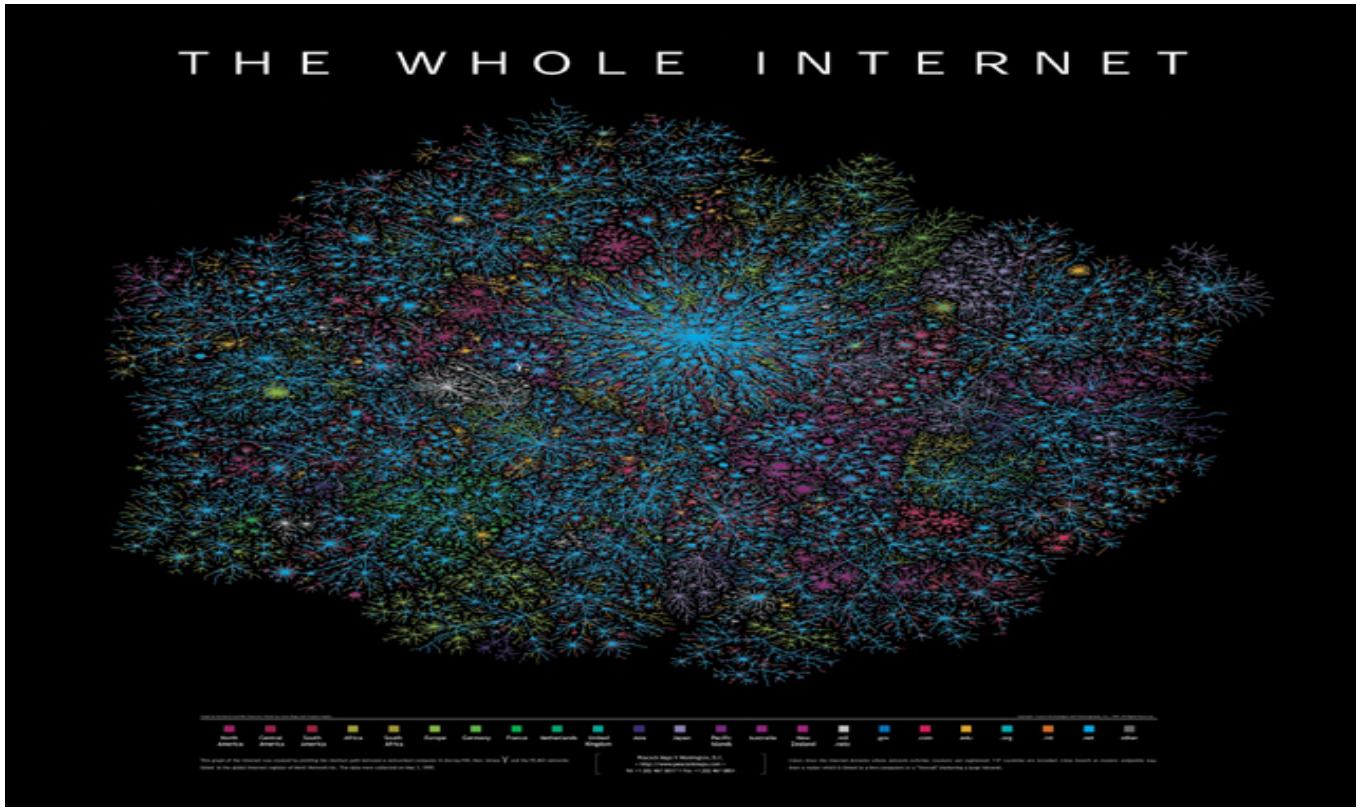
- Una **red** en el ámplio sentido de la palabra se puede definir como un conjunto de objetos interrelacionados.
- En los últimos años ha nacido un gran interés en el análisis de **datos de redes**. Con este nombre nos referimos a medidas que son tomadas dentro un sistema conceptualizado como una red.
- El interés se debe a dos causas fundamentales:
  1. el propio interés suscitado por muchas estructuras que son redes; y
  2. las nuevas capacidades que permiten la colección de grandes masas de datos.
- Este taller está pensado para ofrecer un cierto conocimiento del **análisis estadístico de datos de redes** mediante el uso del software estadístico **R**.

Ejemplos de redes

- Ejemplos de análisis de datos basados en redes se pueden encontrar en un gran número de áreas.
- Veamos algunos ejemplos en cuatro familias de redes (Kolaczyk, 2009):
  1. **Redes tecnológicas**.
  2. **Redes sociales**.
  3. **Redes biológicas**.
  4. **Redes de información**.

Redes tecnológicas

- Ejemplos de redes tecnológicas son redes de comunicación (telefónica, internet,...), redes de transporte (líneas de ferrocarril, líneas aéreas,...) o redes de energia (circuitos eléctricos, gaseoductos,...).



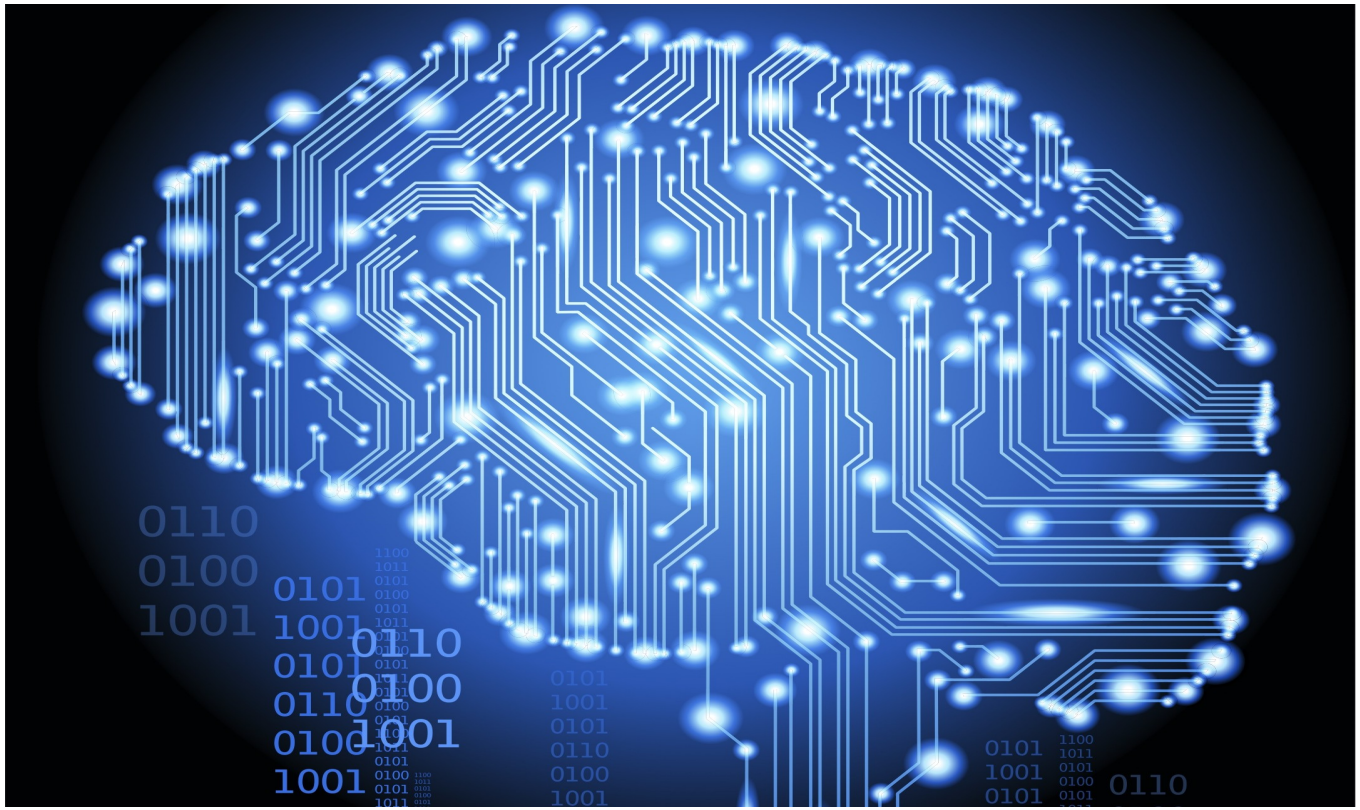
Redes sociales

- Ejemplos de redes sociales son redes de amistad entre un grupo de amigos, redes de pertenecia a un club, redes de contactos o redes sociales en internet (Facebook, Twitter,...).

- **Experimento del mundo pequeño** y **seis grados de separación**.

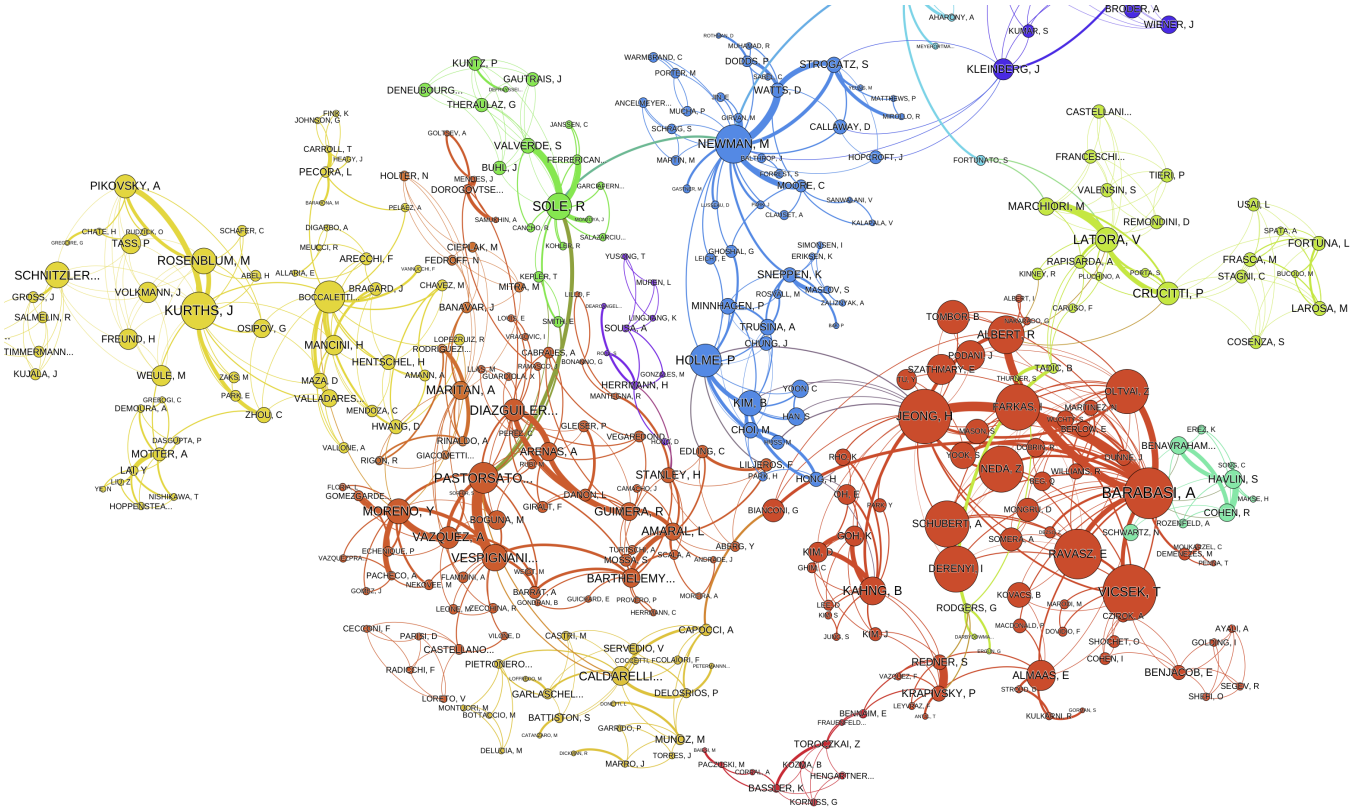
Redes biológicas

- Ejemplos de redes biólogicas son redes de neuronas, redes ecológicas (relaciones depredador-presa) o redes epidemiológicas.



Redes de información

- Ejemplos de redes de información son redes de citas entre artículos académicos (Researchgate,...), redes de autoría de artículos (Erdös number,...), redes de relaciones semánticas (sinónimos, antónimos,...) o redes de páginas webs (links,...).



Datos de redes

- En todos estos ejemplos, además de la propia estructura de la red, podemos encontrar medidas que son tomadas dentro de dicha red.
- Estas medidas (**datos**) se pueden utilizar para profundizar en las relaciones que hay entre los miembros de la red y para utilizar dicha información con diferentes fines.
- Por ejemplo, en el caso de internet nos podemos preguntar:
  1. ¿Cómo de grande es?
  2. ¿Que aspecto tiene?
  3. ¿Cuánto tráfico está fluyendo en internet?
  4. ¿El tráfico anterior es el esperado o es anómalo?

Teoría de grafos

- La representación matemática de las redes son los **grafos**.
- La teoría de grafos estudia las características matemáticas de los grafos.
- Un grafo consiste en un conjunto de **vértices** o **nodos**, con ciertos pares de estos vértices conectados por **ejes**, que pueden ser **dirigidos** o **no dirigidos**.
- No vamos a entrar en los detalles de la teoría de grafos si no a lo largo del taller iremos viendo algunos conceptos necesarios para algunos de los análisis que vamos a hacer.

Computación y big data

- Vamos a utilizar **R** para el análisis de datos de redes. Sin embargo, **R** por si sólo no es capaz de manejar **grandes** masas de datos.
- Fan, Han and Liu (2014): **En general**, la estrategia fundamental para poder análizar big data computacionalmente es la de **dividir y conquistar**, es decir, dividir el problema en subproblemas más tratables e independientes.
- Cada subproblema se aborda en paralelo por diferentes procesadores y los resultados se combinan para obtener la solución final.
- A pequeña escala, tal estrategia se puede implementar, o bien con múltiples núcleos en un ordenador, o bien con una red de múltiples servidores.
- Sin embargo a gran escala, esta estrategia es muy complicada, dado que podemos necesitar un número elevadísimo de servidores, lo que complica enormemente la escritura del software necesario y además existe una alta probabilidad de que algunos de los servidores fallen durante el proceso.

Computación y big data

- Estas y otras razones han motivado la llegada de nueva infraestructura computacional para el almacenamiento y el procesamiento de datos masivos.
- Posiblemente, **Hadoop** es el ejemplo más conocido de este tipo de infraestructura.
- Hadoop es un entorno de software basado en Java para el manejo y procesamiento de datos.
- Contiene un conjunto de librerías para el cálculo distribuido utilizando el modelo de programación **MapReduce** y que contiene su propio sistema de archivos llamado **HDFS**.
- Por lo tanto, Hadoop contiene dos ingredientes básicos:
  1. HDFS (self-healing, high-bandwidth, clustered storage file system), and;
  2. MapReduce (modelo de programación distribuida desarrollado por Google).

Computación y big data

- HDFS es un sistema de archivos diseñado para hospedar y proporcionar acceso a grandes bases de datos que se almacenan en múltiples máquinas.
- Esencialmente, la idea es que si disponemos de un archivo de tamaño enorme al que queremos acceder con frecuencia, HDFS lo divide en bloques pequeños que se almacenan en máquinas diferentes.
- Toda la información sobre la partición se almacena en un servidor único que es al que se accede para manejar al fichero dividido.

Computación y big data

- MapReduce es un modelo de programación para el procesamiento de grandes bases de datos en paralelo.
- Para entender como funciona MapReduce pongamos un ejemplo: tenemos un simbolo, digamos "ATGCCATTGCCTA" y buscamos escribir un programa que cuente el número de veces que aparece cada letra.
- La idea más sencilla es empezar con la primera letra y añadir uno cada vez que aparece. Si el simbolo que hay a continuación no está en la lista de simbolos conocidos, se añade y se continua del mismo modo.
- Si el problema es leer estos símbolos en los genomas de muchos sujetos biológicos, el problema se complica.
- La idea tras MapReduce es dividir la secuencia original en varios archivos (formato HDFS) que a su vez pueden ser divididos en más subsecuencias y resolver el problema en cada una de las subsecuencias en procesadores diferentes.

Computación y big data

- Notar que para poder implementar todo lo anterior necesitamos disponer de un **gran centro de procesamiento**.

Computación y big data

- Un caso extremo: la granja de servidores de Facebook en Lulea (Suecia).

Servicios de cloud computing

- La alternativa es el uso de **cloud computing**.
- La idea es alquilar memoria y capacidad de cálculo por un cierto tiempo y realizar computación intensiva en maquinas virtuales con mayor memoria RAM.
- Algunas opciones:
  1. Windows Azure Virtual Machines: https://azure.microsoft.com/en-us/
  2. Amazon Web Services EC2: http://aws.amazon.com/ec2/
  3. Google Compute Engine: https://cloud.google.com/products/compute-engine/
  4. Google App Engine: https://developers.google.com/appengine
  5. Salesforce Platform: http://www.salesforce.com/platform/

R y Hadoop

- Existen varias maneras de utilizar **R** en un entorno Hadoop siendo la más popular **RHadoop**.
- RHadoop es una colección de cinco librerías de **R** que permite manejar y analizar datos con Hadoop, utilizar bases de datos de Hadoop, y archivos en varios formatos:
  1. rmr,
  2. rhdfs,
  3. rhbase,
  4. plyrmr, y
  5. ravro.

R y cálculo de altas prestaciones

- Más localmente, la web de **R** contiene una página que recoge una serie de librerías útiles para el cálculo de altas prestaciones con **R**

  http://cran.es.r-project.org/web/views/HighPerformanceComputing.html

- Por ejemplo, para el cálculo en paralelo son muy populares las librerías **parallel** o **snow**.

- Muchas otras librerías contienen funciones que permiten paralelizar (en procesadores o núcleos) determinados procesos (**caret** o **pmclust**).

- Otras librerías permiten el uso de grandes cantidades de memoria (como matrices de dimensiones altísimas) para usos determinados (**biglm**, **ff** o **bigmemory**).

Librerías para el análisis de datos de redes en **R**

- **graph**, contiene procedimientos para representar redes.
  1. **RBGL**, contiene varios algoritmos para el análisis de redes.
  2. **Rgraphviz**, contiene procedimientos para visualizar redes en una gran variedad de diseños.
- **igraph**, contiene procedimientos para representar redes, algoritmos para su análisis y procedimientos para visualizar redes en una gran variedad de diseños.
- **gRbase**, contiene algoritmos para la representación gráfica que se pueden utilizar con las dos librerías anteriores.
- **network**, **sna**, **eigenmodel**, **ergm**, **mixer**, **glasso**, **huge**,...
- Ver `http://cran.es.r-project.org/web/views/gR.html`, para más información.

Librería **igraph**

- En este taller, vamos a utilizar fundamentalmente la librería **igraph**.
- El creador de esta librería es Gabor Csardi, investigador post-doctoral en el Departamento de Estadística de la Universidad de Harvard:

  `http://gaborcsardi.org/`

- La librería tiene una página web con documentación:

  `http://igraph.org/`
  `http://igraph.org/r/`

- Esta librería es adecuada para el análisis de datos de redes ya que:
  1. tiene implementados un gran número de algoritmos para el análisis de redes;
  2. permite manejar redes de gran tamaño, de millones de vértices y ejes; y
  3. tiene asociado un libro recientemente publicado (Kolaczyk and Csárdi, 2014).

El plan del taller es...

- Manipular datos de redes.
- Visualizar datos de redes.
- Realizar un análisis descriptivo de los datos de redes.
- Proponer extensiones del análisis.

Componentes fundamentales de una red

- Los componentes fundamentales de una red son:
  1. **vértices** (nodos): colección de objetos que se interrelacionan. El número total de vértices es el **orden** de la red y se denota por N_v.
  2. **ejes** (aristas): colección de pares de vértices que se relacionan. El número total de ejes es el **tamaño** de la red y se denota por N_e.
  3. **atributos** de vértices o ejes: características de dichos vértices o ejes.
- En particular, decimos que:
  1. la red es **dirigida** si los ejes son dirigidos, es decir, si existe un orden entre los vértices que unen.
  2. un vértice es **aislado** si no tiene relación con el resto.
  3. un eje es un **lazo** si conecta a un vértice consigo mismo.

Red de aeropuertos de Estados Unidos

- Como ilustración, analizamos un conjunto de datos con información sobre todos los vuelos de pasajeros entre aeropuertos de Estados Unidos durante Diciembre de 2010.
- Los **vértices de la red** son los aeropuertos de pasajeros que hay en Estados Unidos. En total, tenemos $N_v = 755$ vértices (el **orden** de la red).
- Los **ejes de la red** son las conexiones entre dos de los anteriores aeropuertos que hubo dicho mes. En total, tenemos $N_e = 23473$ ejes (el **tamaño** de la red).
- Este es un ejemplo de **red dirigida** donde las direcciones de los ejes vienen dadas por las direcciones de los vuelos.
- Cada eje es específico de una compañía. Es decir, vuelos que enlazan los mismos aeropuertos en la misma dirección forman lo que llamamos **ejes múltiples**.

Red de aeropuertos de Estados Unidos

- Los aeropuertos (vértices) tienen los siguientes **atributos**:
  1. Código IATA del aeropuerto:

     `http://www.iata.org/publications/Pages/code-search.aspx`
  2. Ciudad y estado en los que está localizado el aeropuerto.
  3. Posición del aeropuerto en coordenadas WGS.

Red de aeropuertos de Estados Unidos

- Las conexiones (ejes) tienen los siguientes **atributos**:
  1. Nombre de la aerolínea.
  2. Número de salidas para una aerolínea y tipo de aeronave.
  3. Número total de asientos disponibles en los vuelos para una aerolínea y tipo de aeronave.
  4. Número total de pasajeros en los vuelos para una aerolínea y tipo de aeronave.
  5. Tipo de aeronave.
  6. Distancia entre los dos aeropuertos, en millas.

Crear una red en **R** con la librería **igraph**

- La librería **igraph** dispone de la clase `igraph` para redes.
- Hay varias maneras de crear un objeto de clase `igraph` que contenga toda la información que disponemos sobre la red.
- Para **redes (muy) pequeñas**:
  1. `graph.formula`: introducir a mano los vértices y los ejes mediante ciertas expresiones que dependen de las características de vértices y ejes.
  2. Una vez definidos vértices y ejes se pueden incluir sus atributos de manera sencilla (ver la ayuda de estas funciones).
- Para **redes especiales** como grafos de estrella, anillos, árboles,... se utilizan las funciones `graph`, `graph.atlas` o `graph.famous`, entre otras.

Crear una red en **R** con la librería **igraph**

- Para **redes grandes** (red de aeropuertos), la red se define mediante dos `data.frame`.
- En el primer `data.frame` tenemos la información relativa a los vértices: en la primera columna el nombre de los vértices, y el resto de columnas el valor de los atributos de cada vértice.
- En el segundo `data.frame` tenemos la información relativa a los ejes: en la primera columna los vértices relacionados, y el resto de columnas el valor de los atributos de cada eje.
- Los `data.frame` se usan para crear la red mediante `graph.data.frame`.
- Realizamos estas operaciones en **R** para los datos de los aeropuertos obteniendo un objeto de tipo `igraph` con la red definida con el nombre `Aeropuertos USA`.
- Veamos en la siguiente transparencia los contenidos de dicha red que coinciden con los descritos previamente.

Red de aeropuertos de Estados Unidos

1. `DN` indica que la red es dirigida, por lo que hay un orden en los ejes.
2. 755 23473 indica que hay 755 aeropuertos y 23473 vuelos.
3. `name (g/c)` indica que el nombre de la red es de tipo carácter.
4. `name (v/c)` indica un atributo de los vértices de tipo carácter.
5. `City (v/c)` indica un atributo de los vértices de tipo carácter.
6. `Position (v/c)` indica un atributo de los vértices de tipo carácter.
7. `Carrier (e/c)` indica un atributo de los ejes de tipo carácter.
8. `Departures (e/n)` indica un atributo de los ejes de tipo numérico.
9. `Seats (e/n)` indica un atributo de los ejes de tipo numérico.
10. `Passengers (e/n)` indica un atributo de los ejes de tipo numérico.
11. `Aircraft (e/n)` indica un atributo de los ejes de tipo numérico.
12. `Distance (e/n)` indica un atributo de los ejes de tipo numérico.

La matriz de adyacencia

- La información de los vértices y los ejes de las redes se puede almacenar en la **matriz de adyacencia**.
- Esta matriz, digamos A, tiene dimensión $N_v \times N_v$ y se define como:

$$A_{ij} = \begin{cases} n_{ij} & \text{si } \{i,j\} \text{ es un eje múltiple de tamaño } n_{ij} \\ 0 & \text{en cualquier otro caso} \end{cases}$$

- Esta matriz de muy importante porque permite realizar varios cálculos sobre la red de manera eficiente necesarios para algunas técnicas que veremos más adelante.
- Para obtener la matriz de adyancencia utilizamos la función `get.adjacency`.
- Dado que la red puede tener un tamaño muy grande, en la librería **igraph** la matriz de adyacencia es un objeto de la clase `dgCMatrix`.
- Este tipo de objetos permite definir matrices "huecas" en un formato que hace uso de columnas comprimidas y que evita crear objetos de mayor tamaño.

Algunas preguntas

- Una vez que tenemos la red podemos responder algunas preguntas descriptivas:
  1. ¿Qué ciudad tiene un mayor número de aeropuertos?
  2. ¿Qué compañia tiene un mayor número de conexiones?
  3. ¿Cúal es el mayor número de salidas para una conexión y compañia determinada?
  4. ¿Cúal es el mayor número de plazas ofertadas para una conexión y compañia determinada?
  5. ¿Cúal es el mayor número de pasajeros para una conexión y compañia determinada?
  6. ¿Cúal es la mayor distancia en una conexión?

Red ponderada

- Los atributos de vértices y ejes son útiles para obtener un mayor entendimiento de los componentes de la red.
- A veces, estos atributos pueden formar parte de la red en forma de **pesos de los ejes**.
- Los pesos de los ejes suelen ser valores no negativos y, por convención, se suelen escalar para que estén entre 0 y 1.
- Por ejemplo, si queremos que las conexiones entre aeropuertos estén ponderadas por el número de plazas ofertadas o por el número de pasajeros sólo debemos incluir el atributo correspondiente como peso mediante la función `E(red)$weight`.
- En lo que sigue no vamos a realizar ponderación alguna pero los resultados se pueden modificar de esta manera sencilla.

Visualizar redes

- Visualizar redes es más complicado que visualizar otro tipos de estructuras.
- Para visualizar una red no sólo hay que considerar una combinación de matemáticas, estadística, algoritmos,. . . también hay que considerar aspectos estéticos.
- Por lo general, se suelen representar los vértices mediante figuras geométricas (puntos, círculos, cuadrados,. . .) y los ejes mediante curvas suaves.
- Para la percepción humana es más sencillo dibujar una red en dos dimensiones.

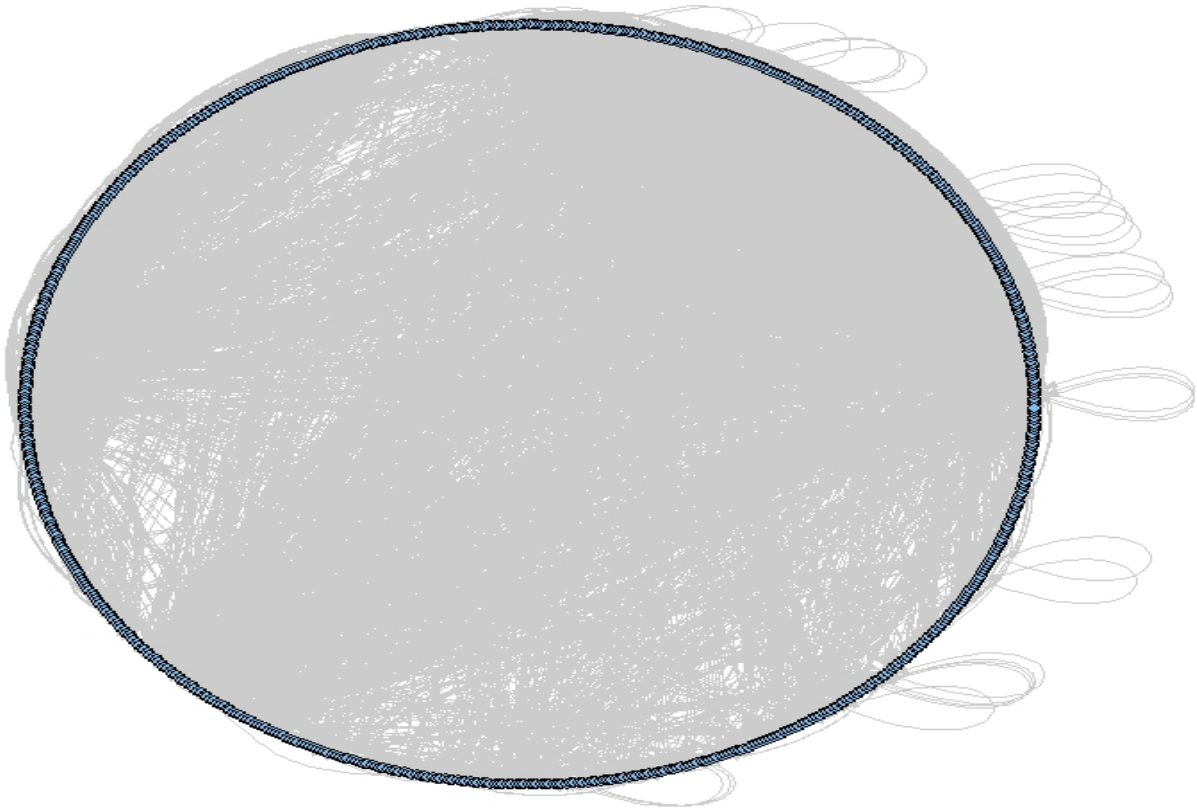
Visualizar redes

- Evidentemente, hay un número infinito de maneras de dibujar una red. La pregunta fundamental es como dibujar la red de manera que se muestren la información más relevante de la red.
- Vamos a ver diferentes visualizaciones de la red de aeropuertos obtenidas mediantes métodos diferentes.
- Normalmente estos métodos están basados en resolver problemas de optimización sobre determinados parámetros.
- Sobre redes de gran tamaño, estos problemas son complicados por lo que existen algoritmos que proporcionan soluciones aproximadas.
- De hecho, no siempre obtenemos el mismo gráfico aplicando el mismo método con los mismos datos.
- Además, la obtención del gráfico con determinados métodos es costosa computacionalmente.

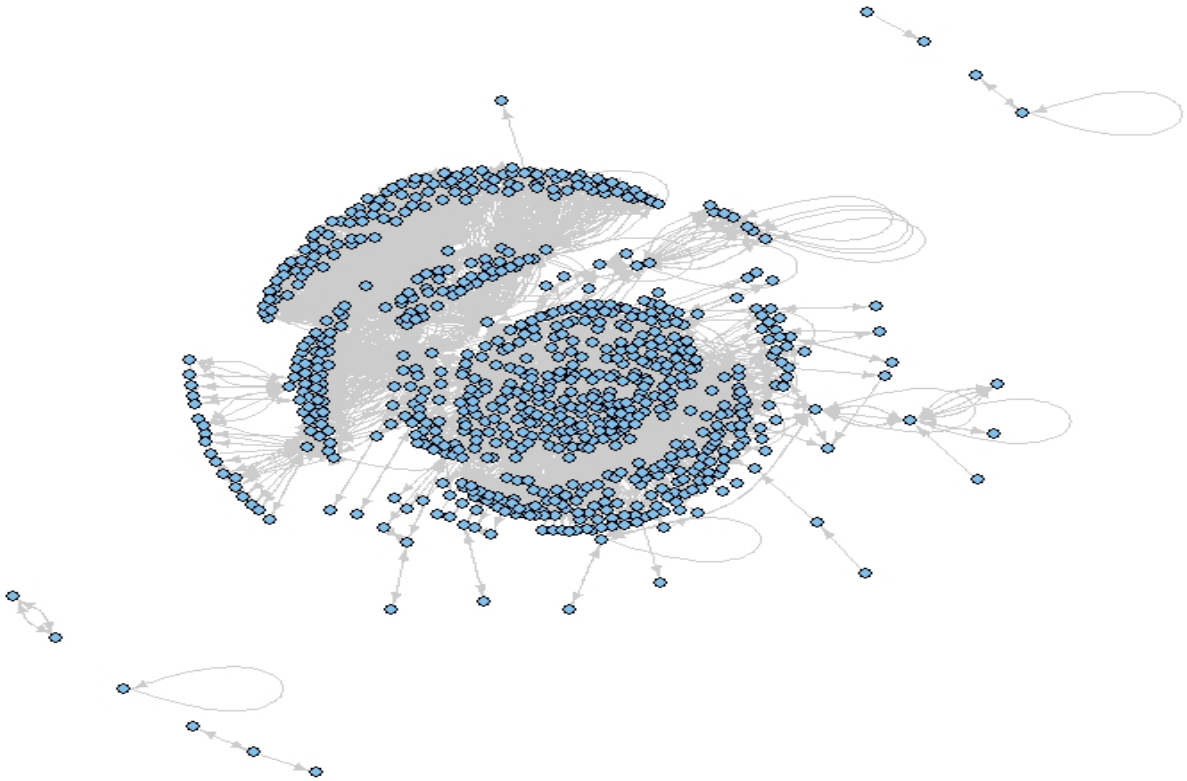
Visualizar redes

- Vemos las redes con los métodos:
  1. **Método del círculo**: consiste en disponer los vértices en un círculo y unir los vértices con los ejes.
  2. **Método de Kamada y Kawai**: consiste en disponer los vértices con la misma longitud aproximadamente y con pocos cruces entre ellos.
  3. **Método de Fruchterman y Reingold**: similar al método de Kamada y Kawai si bien la función a optimizar es diferente.
  4. **Método DrL**: consiste en agrupar vértices y está especialmente diseñado para visualizar redes de gran tamaño.

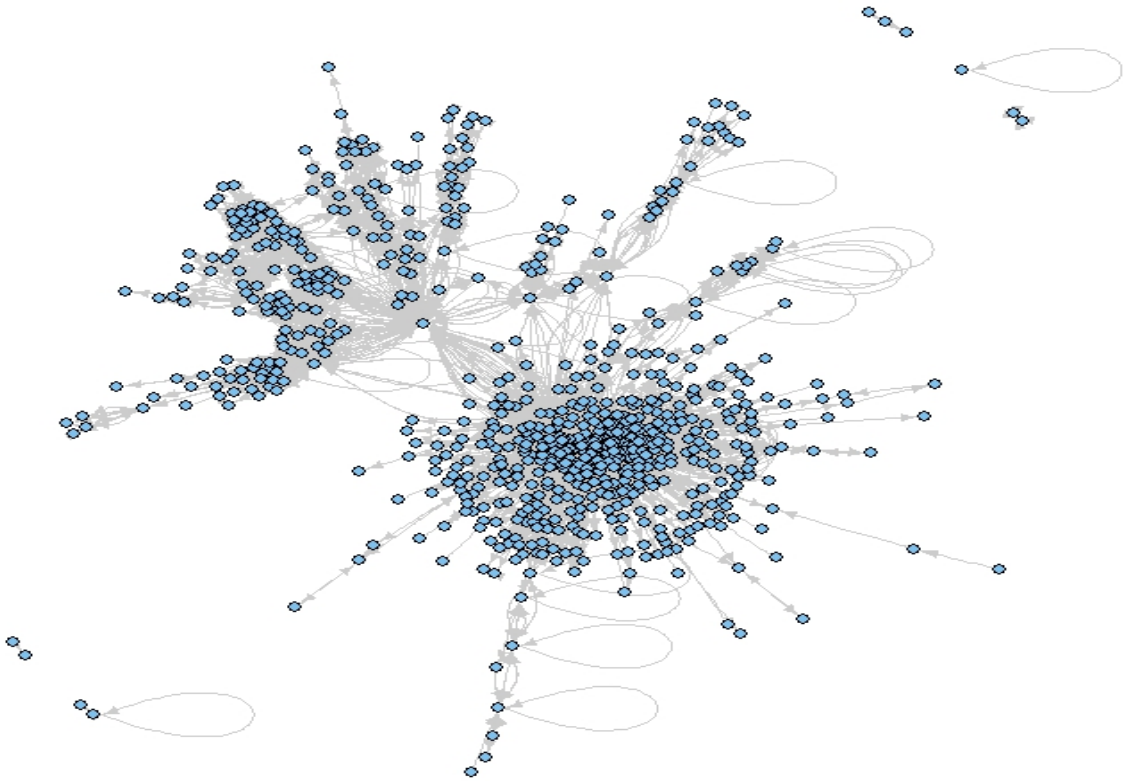
Método del círculo



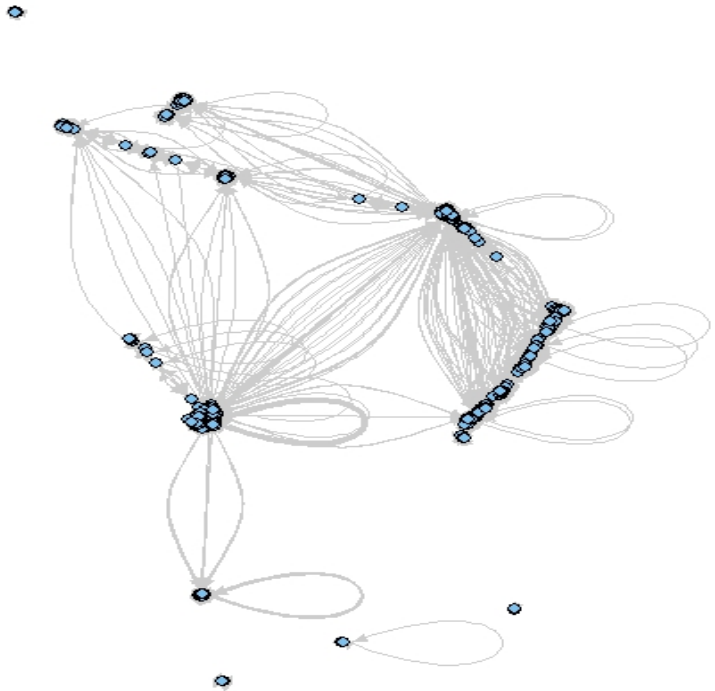
Método de Kamada y Kawai



Método de Fruchterman y Reingold



Método DrL



Visualizar redes

- Algunas conclusiones:
  1. Parece haber **grupos** de aeropuertos.
  2. También aeropuertos aislados, ya bien tengan relación sólo con otro aeropuerto que también está aislado, o con otro aeropuerto que si tiene muchas relaciones con otros.
- Podemos modificar muchas de las opciones que aparecen en estos gráficos. Por ejemplo, podemos hacer uso de los atributos de vértices y ejes para potenciar algún aspecto de dichos atributos en el gráfico.
- Volveremos a la visualización de redes más adelante una vez que veamos algunas técnicas del análisis descriptivo.

Análisis descriptivo de las características de las redes

- El objetivo a continuación es explorar características y propiedades estructurales de las redes.
- Empezaremos por el concepto de **conectividad** de la red. La pregunta es cuando una red se puede separar en diferentes sub-redes no conectadas.
- A continuación calcularemos métricas sencillas que resuman la estructura topológica de la red y acabaremos estrayendo grupos (**comunidades**) a partir de las relaciones en la red.
- Para ello, vamos a dividir el análisis en tres partes:
  1. Conectividad de la red.
  2. Caracterización de vértices y ejes: **grado de los vértices**, **medidas de centralidad de los vértices** y extensiones para los ejes.
  3. Caracterización de la cohesión de la red: **densidad** y **detección de comunidades**.

Conectividad de la red

- Como se ha mencionado, dada una red, buscamos si existen sub-redes formadas por vértices que no estén conectados entre ellos.
- Diremos que una red está **conectada** si todo vértice se puede alcanzar desde cualquier otro siguiendo un **camino**, es decir, la unión de ejes consecutivos.
- Notar que en el caso de redes dirigidas, como es el caso de la red de aeropuertos, obviaremos la dirección de los ejes para aplicar la definición anterior (**conectividad débil**), si bien se puede extender al caso dirigido (**conectividad fuerte**).
- En la red de aeropuertos hay una sub-red formada por 745 de los 755 aeropuertos. Es lo que se denomina la **componente gigante** de la red.
- En lo que resta trabajaremos con esta componente gigante, que será una red conectada. El resto de definiciones siguientes supondra que la red está conectada.

Caracterización de vértices y ejes: grado de los vértices

- El **grado** de un vértice es el número de ejes que inciden (entran o salen) en dicho vértice.
- Por lo tanto, el grado proporciona una medida de como el vértice está conectado con el resto de la red.
- Como en el caso de la red de los aeropuertos, los ejes son dirigidos. En estos caso, además del grado, hablaremos del **grado de entrada** y del **grado de salida**.
- La **distribución del grado de la red** se define como las frecuencias relativas de los grados de los vértices que forman la red. Definiciones análogas se dan para los grados de entrada y de salida.
- Las distribuciones de los grados suelen presentar una alta asimetría positiva.

Caracterización de vértices y ejes: centralidad

- Una problema fundamental en el análisis de una red es entender cual es la importancia de un vértice dentro de una red.
- Varias medidas de **centralidad** han sido propuestas para cuantificar la noción de "importancia" de un vértice.
- Sin embargo, hay una gran falta de consenso sobre que significa que un vértice sea importante y que medida debe ser utilizada para medir dicha centralidad.
- No obstante, hay una medida de centralidad que es muy utilizada y que acabamos de definir: el grado de los vértices.
- Nos centramos entonces en ver otros tipos de medidas de centralidad de los vértices: **centralidad por cercanía**, **centralidad por intermediación** y **centralidad por status**.
- También veremos el concepto de **centralidad de ejes**.

Caracterización de vértices y ejes: centralidad

- Las definiciones de las medidas de centralidad que vamos a ver dependen del concepto de **distancia entre vértices**.
- Posiblemente, la noción de distancia entre vértices más importante es la llamada **distancia geodésica**.
- Para una red no dirigida, la distancia geodésica entre dos vértices u y v, denotada por $dist\,(u, v)$, se define como la longitud del **camino más corto** entre los vértices, es decir, el número de ejes para ir de un vértice al otro.
- Para una red dirigida, como la red de aeropuertos, debemos distinguir entre la distance **desde** el vértice u al v (modo salida) y la distancia **al** vértice u desde el v (modo entrada).
- En particular, $dist\,(u, v) = \infty$ si no puedo llegar de u al v o viceversa.
- En particular, el valor de la mayor distancia geodésica (excluyendo ∞) en una red se denomina **diámetro** de la red.

Caracterización de vértices y ejes: centralidad

- Decimos que un vértice es el más **central por cercanía** si es el vértice que está más "cerca" del resto de vértices.
- El procedimiento estándar es que la centralidad de un vértice varíe inversamente con la distancia total del vértice al resto de vértices.
- Por lo tanto, definimos la centralidad por cercanía de un vértice u como:

$$c_{cer}(u) = \frac{1}{\sum_v dist(u, v)},$$

donde de nuevo hay que tener en cuenta que las distancias se calculan en el modo de entrada o de salida.

Caracterización de vértices y ejes: centralidad

- Decimos que un vértice es el más **central por intermediación** si sus ejes adyacentes son los que más sirven de ruta más corta del resto de vértices.
- Por lo tanto, definimos la centralidad por intermediación de un vértice u como:

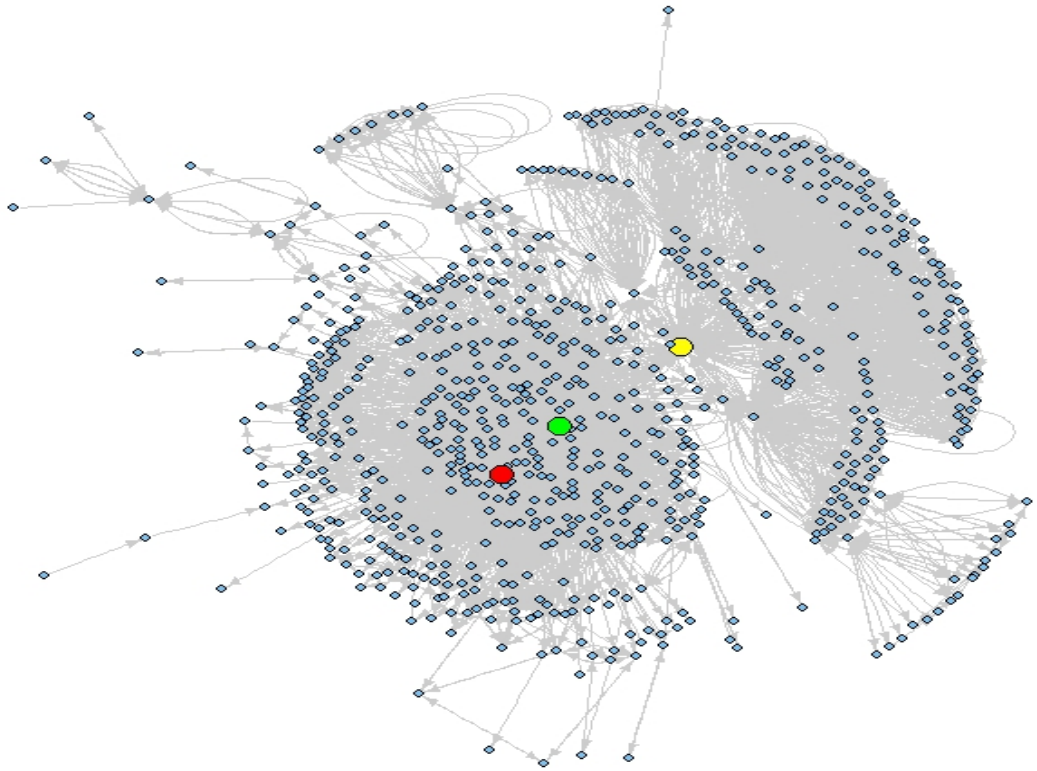
$$c_{int}(u) = \sum_{s \neq t \neq u} \frac{\sigma(s, t|u)}{\sigma(s, t)},$$

donde $\sigma(s, t|u)$ es el número total de caminos más cortos entre los vértices s y t que pasan por u, y $\sigma(s, t)$ es el número total de caminos más cortos entre los vértices s y t.

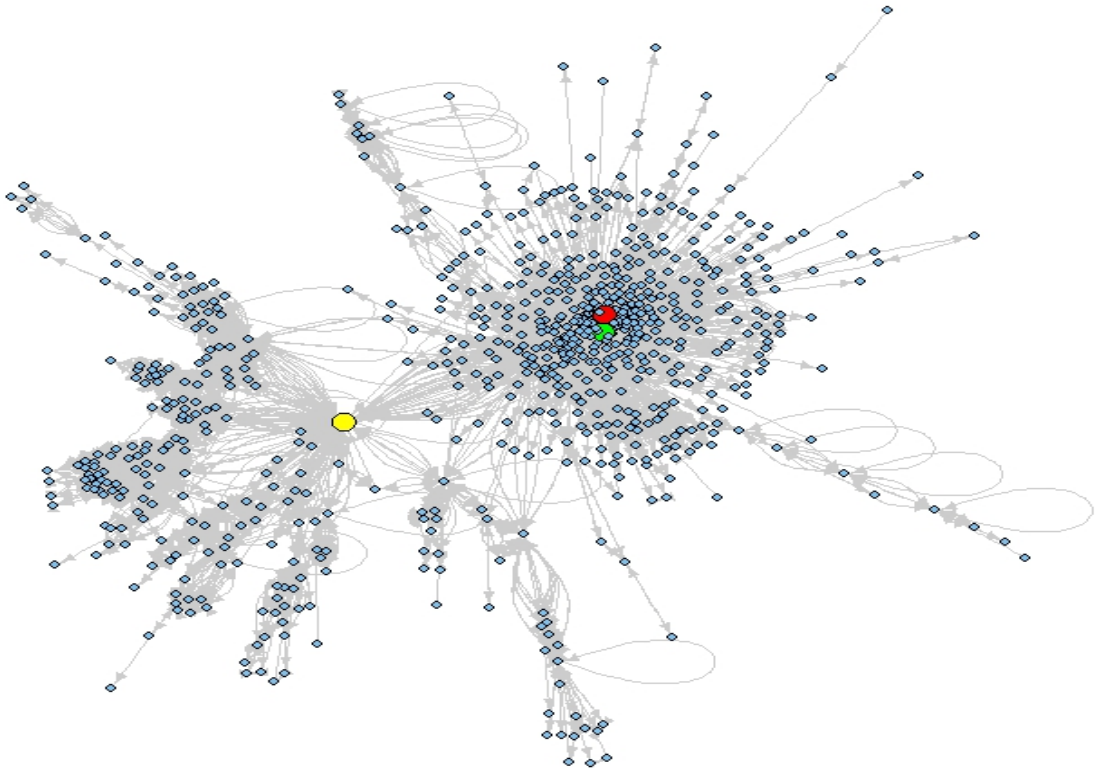
Caracterización de vértices y ejes: centralidad

- Decimos que un vértice es el más **central por status** si es a su vez el vértice con vecinos más centrales.
- Como vemos, la definición anterior es inherentemente implícita. Su plasmación numérica se suele expresar en términos de autovectores de ciertos sistemas lineales de equaciones. Damos la definición más extendida.
- Definimos la centralidad por status de los vértices que forman una red como el vector $a_1 c_{est}$ de dimensión $N_v \times 1$, siendo a_1 es mayor autovalor de la matriz de adyacencia A ligado al autovector c_{est}.
- Por lo tanto, denotamos cada componente de dicho vector mediante $c_{est}(u)$, siendo u un vértice de la red.

Método de Kamada y Kawai



Método de Fruchterman y Reingold



Caracterización de vértices y ejes: centralidad

- Decimos que un eje es el más **central por intermediación** si dicho eje es el que más veces aparece en los caminos más cortos entre los vértices.
- Por lo tanto, definimos la centralidad por intermediación de un eje e como:

$$c_{int,eje}(e) = \sum_{s \neq t} \frac{\sigma(s,t|e)}{\sigma(s,t)},$$

donde $\sigma(s,t|e)$ es el número total de caminos más cortos entre los vértices s y t que pasan por e, y $\sigma(s,t)$ es el número total de caminos más cortos entre los vértices s y t.

Caracterización de la cohesión de la red

- Al principio de este capítulo hemos visto el concepto de conectividad. Gracias a él, hemos podido definir varios sub-redes, una de ellas, la componente gigante de la red.
- La conectividad es una posible caracterízación de la **cohesión** de la red. Por ejemplo, nos podemos preguntar si dos aeropuertos que tienen conexión con un tercero tiene a su vez conexión entre ellos. También nos podemos preguntar si hay grupos de aeropuertos "separados" respecto de otros.
- Vamos a ver dos maneras de medir la cohesión de una red:
  1. Densidad; y
  2. Detección de comunidades.

Caracterización de la cohesión de la red: densidad

- La **densidad** de una red se define como el ratio entre ejes existentes y ejes posibles.
- Por ejemplo, en una red dirigida sin lazos y sin ejes múltiples la densidad de una red (R) se define por:

$$den(R) = \frac{N_e}{N_v\left(N_v - 1\right)}$$

- La densidad es entonces un número entre 0 y 1 que mide como de cerca está la red de tener tantos ejes como sea posible.
- Sin embargo, la definición de densidad local no tiene una definición clara en el caso de redes con lazos y ejes múltiples.

Caracterización de la cohesión de la red: detección de comunidades

- Por **detección de comunidades**, entendemos la segmentación de los vértices de la red en grupos disjuntos.
- En el análisis de redes, la detección de comunidades en una red es útil para encontrar grupos disjuntos cuyos vértices tengan una cierta cohesión con respecto a las relaciones entre ellos.
- Más concretamente, buscamos grupos de vértices que:
  1. están bien conectados entre ellos; y
  2. están relativamente bien separados del resto de vértices en otros grupos.

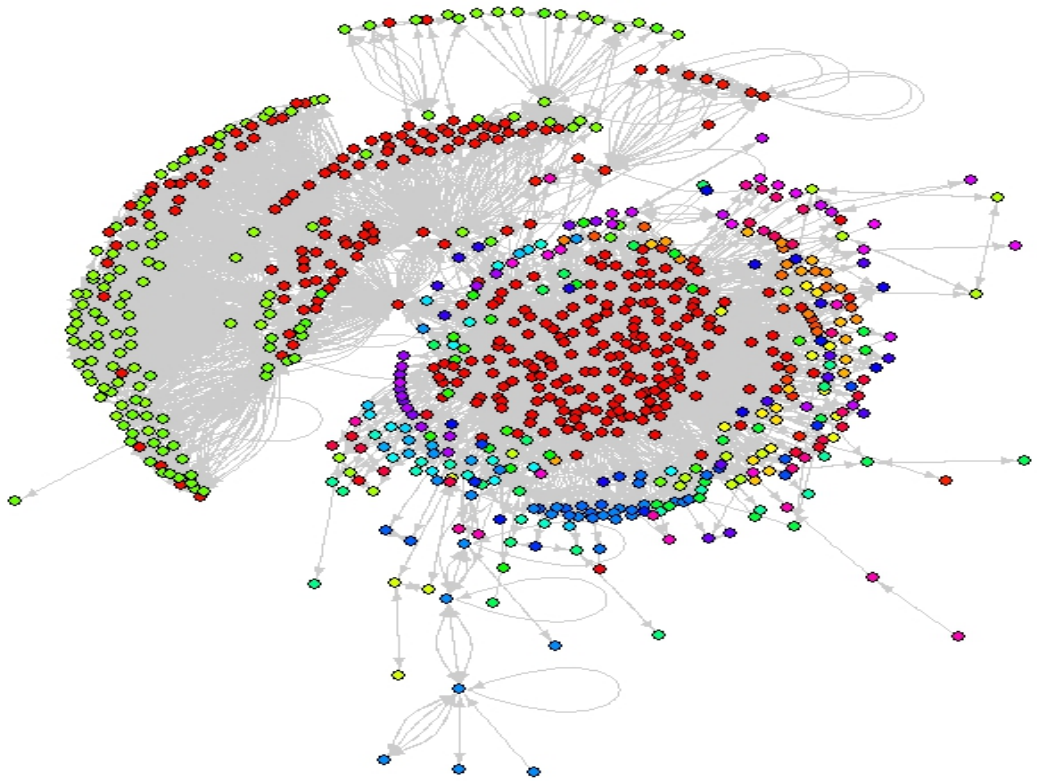
Caracterización de la cohesión de la red: detección de comunidades

- Hay un número importante de algoritmos de detección de comunidades que a su vez pueden ser divididos en varios tipos de métodos: métodos jerárquicos, métodos de partición espectral,...
- Sin embargo, la mayoría de ellos presentan dos problemas importantes:
  1. están diseñados para redes no dirigidas; y
  2. tienen un coste computacional muy alto para redes muy grandes.
- En la librería `igraph` podemos encontrar hasta ocho algoritmos diferentes. Vamos a ver tres de ellos:
  1. El algoritmo de Newman-Girvan (Newman and Girvan, 2004);
  2. el algoritmo de Pons-Latapy (Pons and Latapy, 2005); y
  3. el algoritmo de Reichardt-Bornholdt (Reichardt and Bornholdt, 2006).

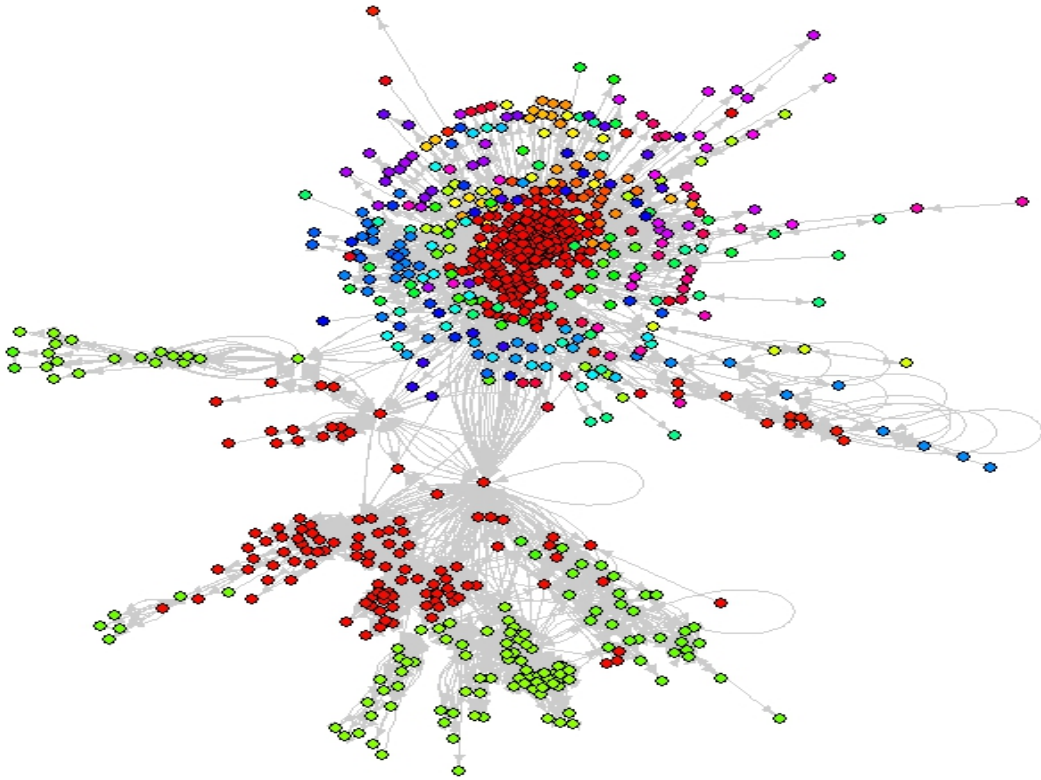
Caracterización de la cohesión de la red: detección de comunidades

- Como hemos visto, la cercanía por intermediación de un eje mide el número de caminos más cortos que pasan por dicho eje.
- La idea principal del algoritmo de **Newman-Girvan** está basada en esperar que ejes que conecten comunidades diferentes tengan un alto valor de la intermediación ya que todos los caminos más cortos deben pasar por dicho eje.
- Por lo tanto, si eliminamos el eje con mayor cercanía por intermediación dividiremos la red en dos sub-redes, que a su vez se pueden dividir de manera similar.
- Por lo tanto, este algoritmo es un algoritmo de tipo jerárquico y podemos obtener un **dendograma** como en el caso de datos multivariantes.
- El problema fundamental de este algoritmo es que es computacionalmente bastante costoso.

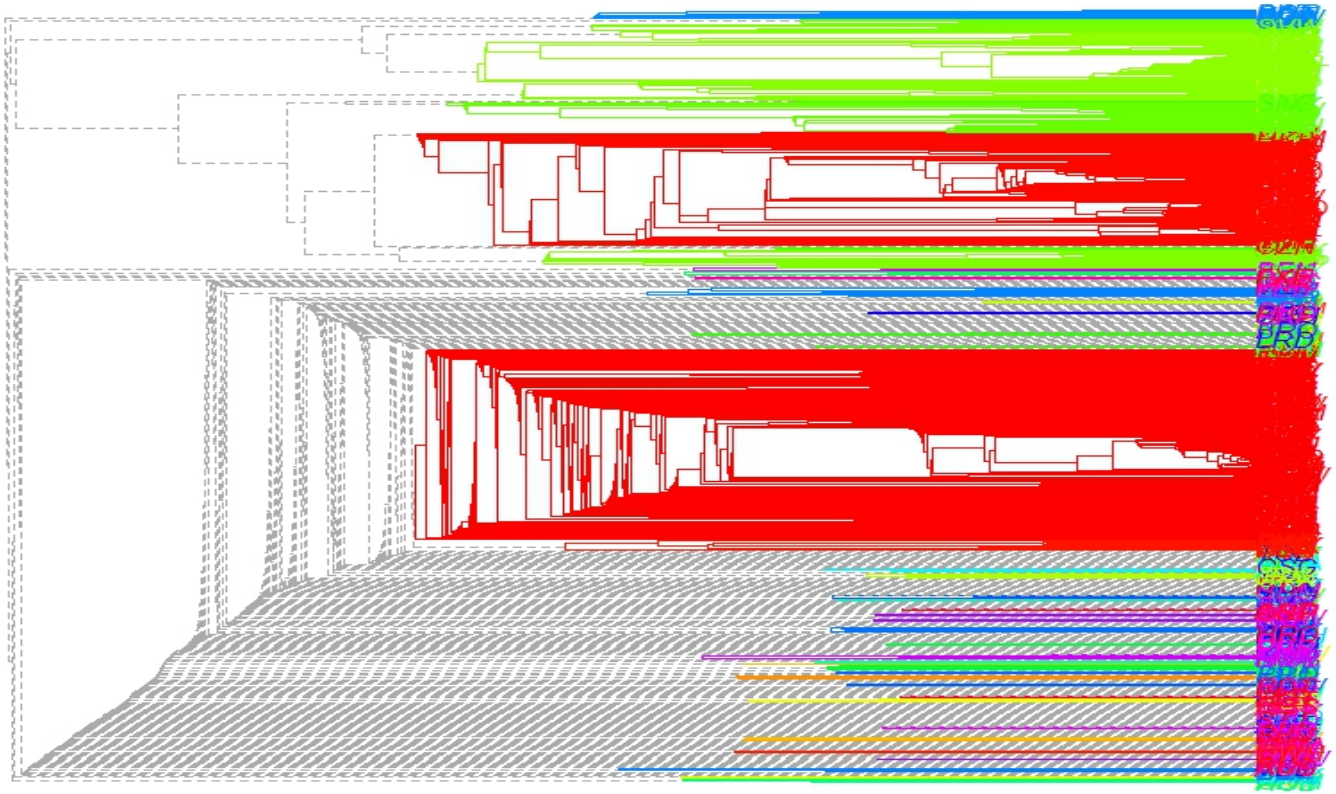
Algoritmo de Girvan-Newman con método de Kamada y Kawai



Algoritmo de Girvan-Newman con método de Fruchterman y Reingold



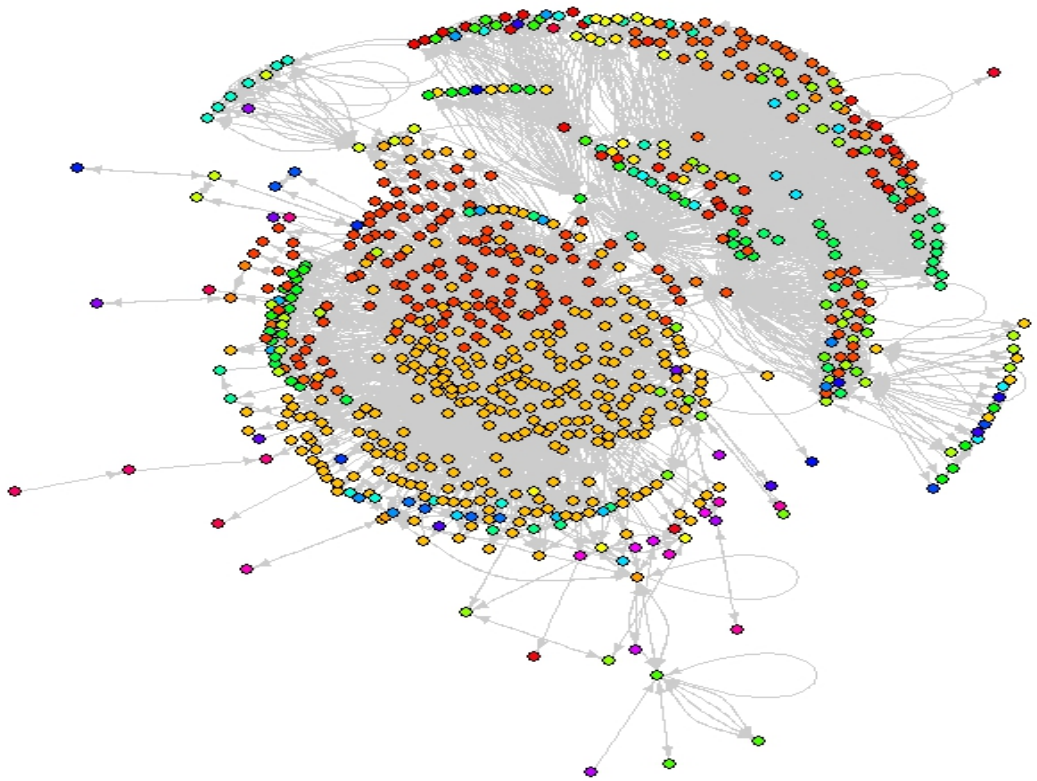
Algoritmo de Girvan-Newman - Dendograma



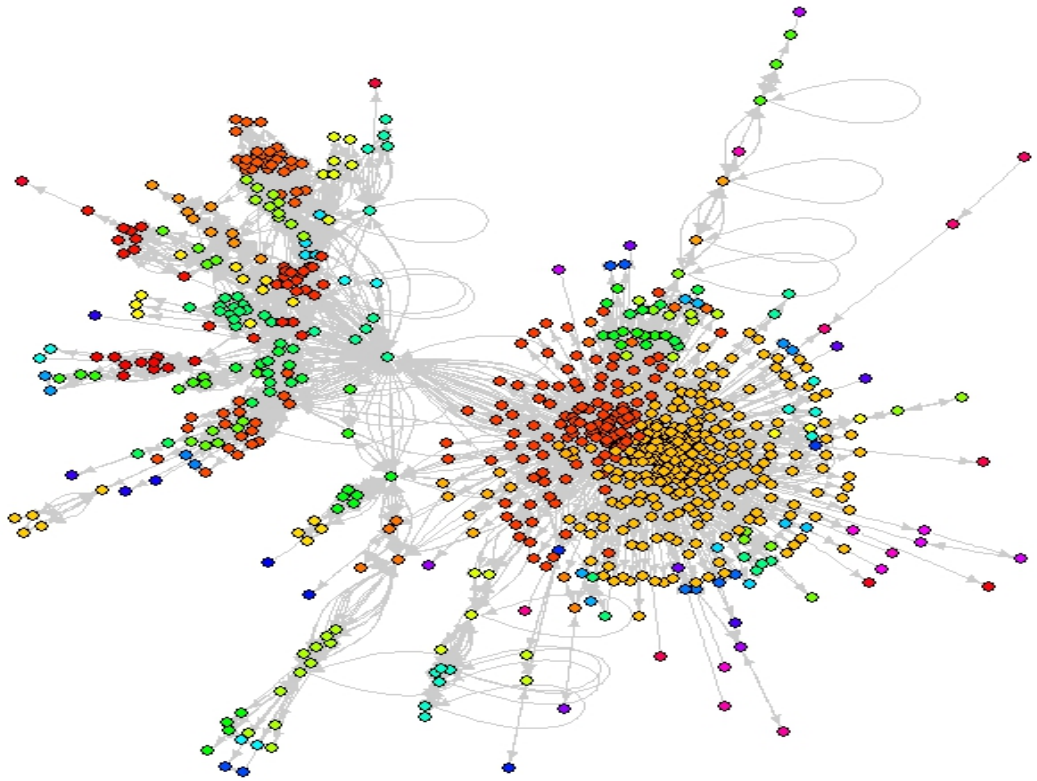
Caracterización de la cohesión de la red: detección de comunidades

- La idea del principal del algoritmo de **Pons-Latapy** está basada en encontrar grupos que estén densamente conectados mediante el uso de paseos aleatorios.
- Se espera que los vértices que recorren paseos aleatorios estén dentro de la misma comunidad.
- Este algoritmo ignora la dirección de los ejes al buscar los grupos. Sin embargo, es extremadamente rápido incluso en redes de un tamaño muy grande.
- El problema está en la elección de la longitud de los paseos aleatorios utilizados. La elección habitual es 4, pero si aumentamos este valor, el número de comunidades detectadas se puede reducir bastante.

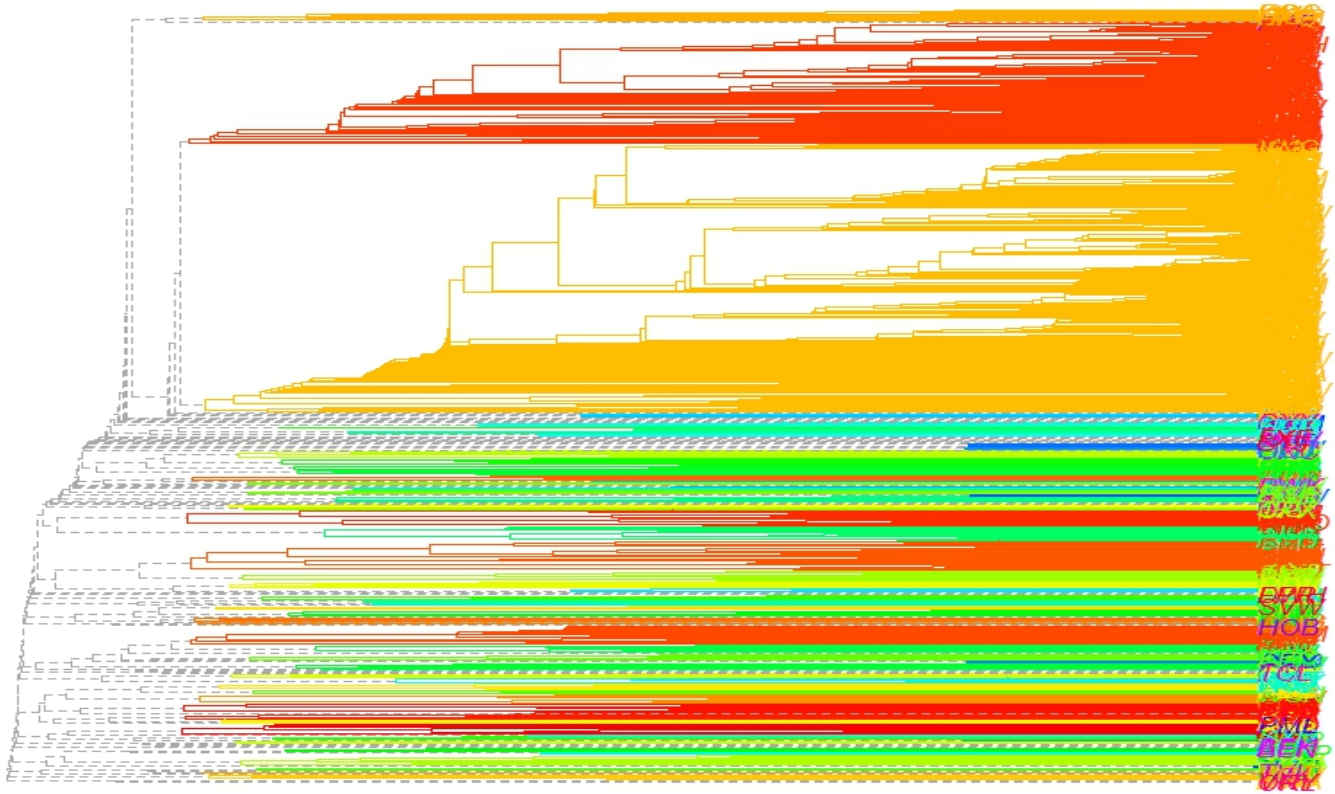
Algoritmo de Pons-Latapy (steps=4) con método de Kamada y Kawai



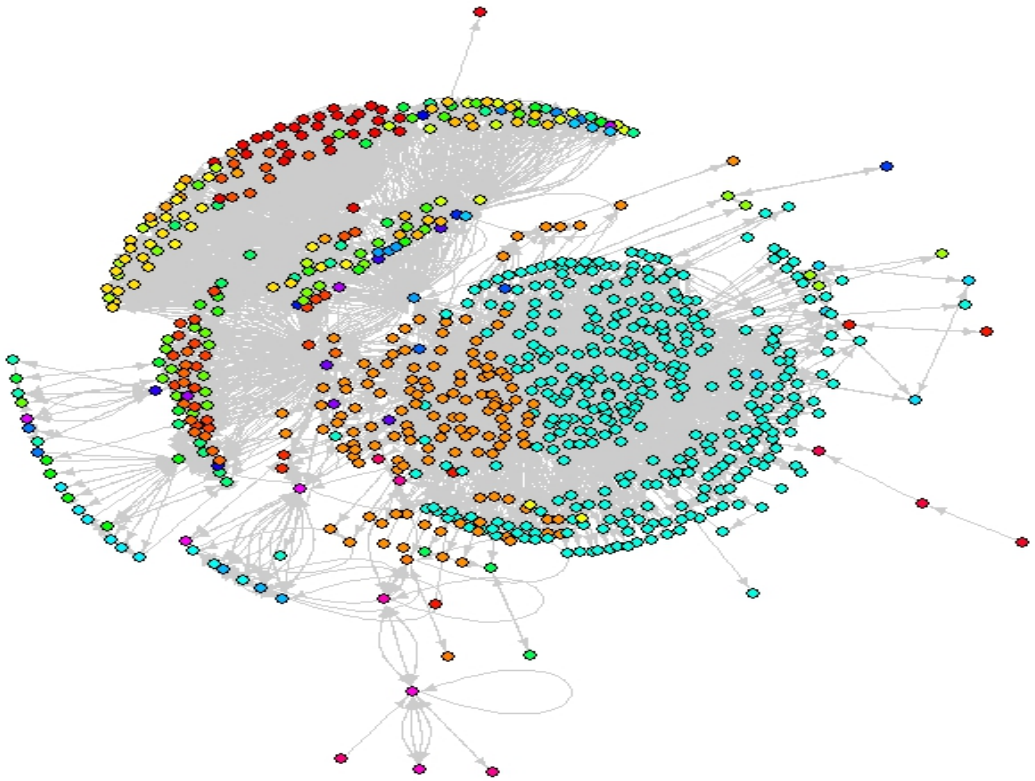
Algoritmo de Pons-Latapy (steps=4) con método de Fruchterman y Reingold



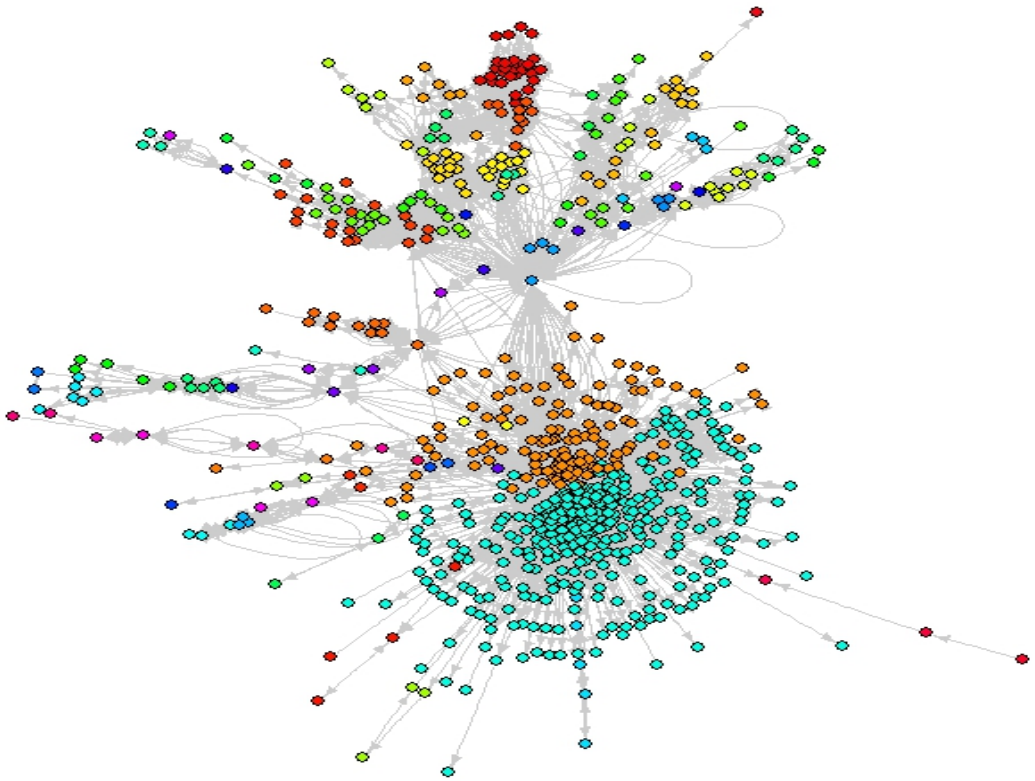
Algoritmo de Pons-Latapy (steps=4) - Dendograma



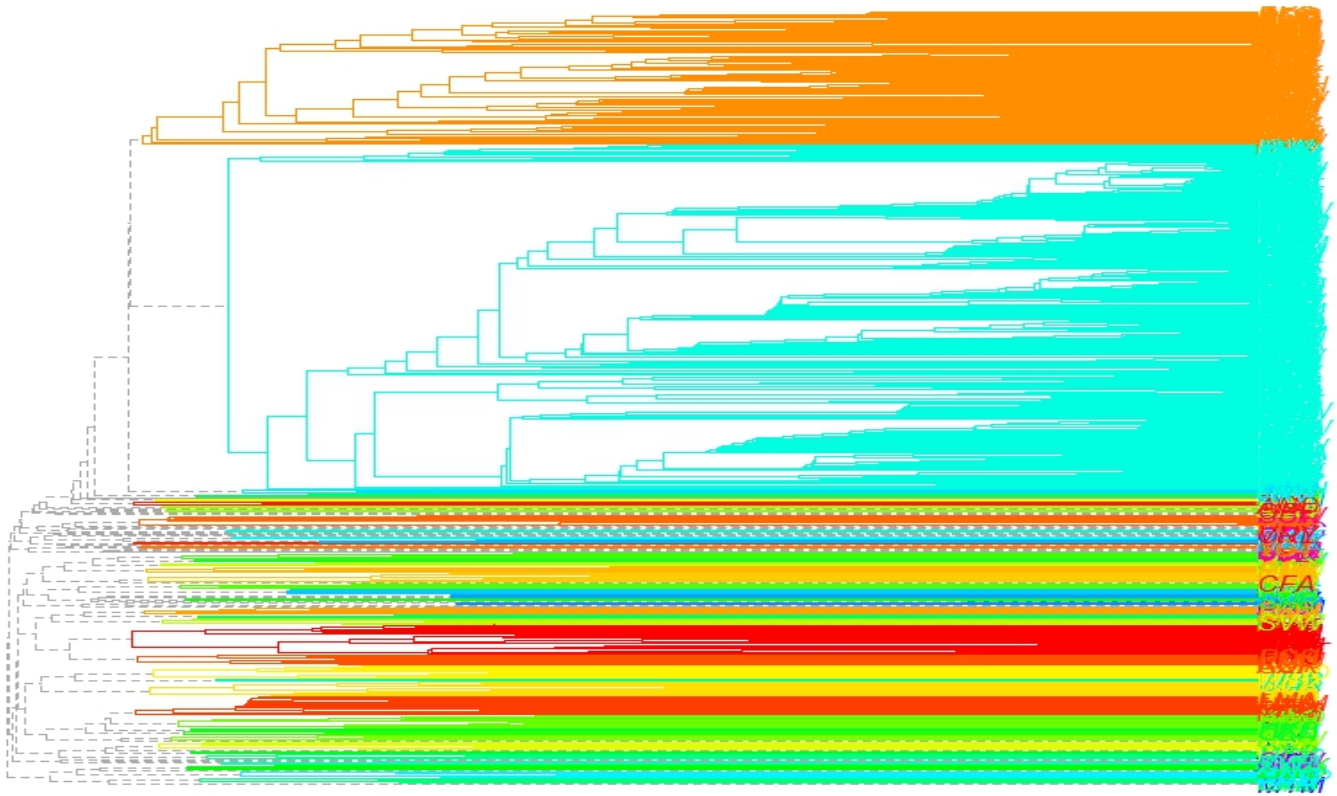
Algoritmo de Pons-Latapy (steps=20) con método de Kamada y Kawai



Algoritmo de Pons-Latapy (steps=20) con método de Fruchterman y Reingold



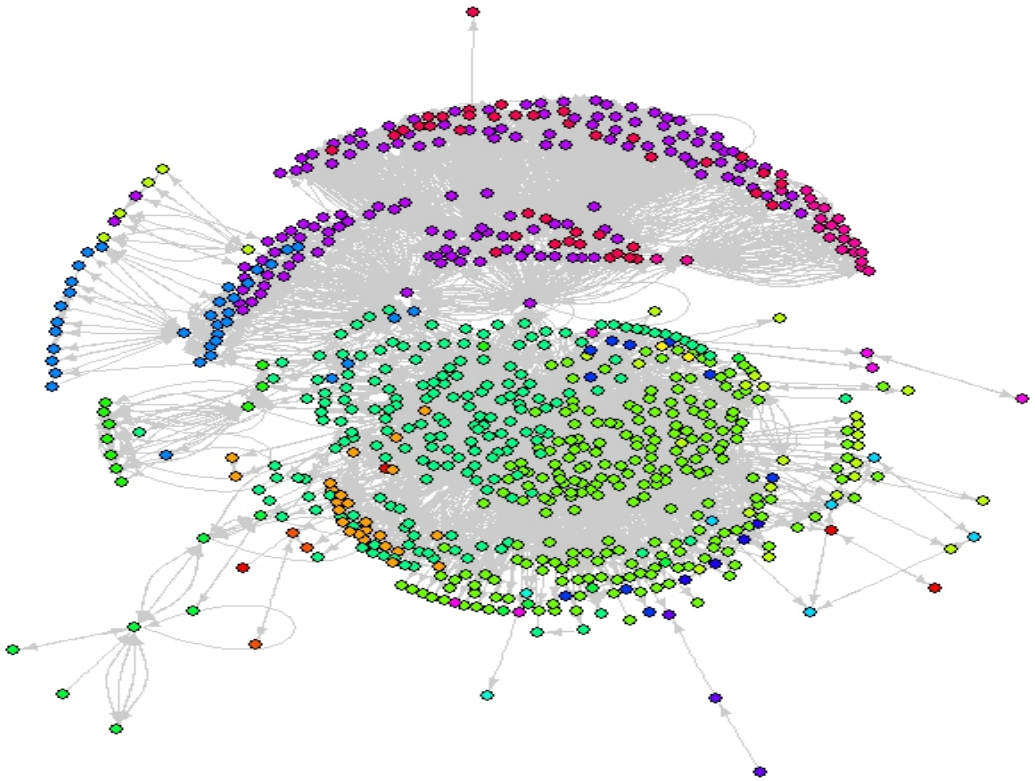
Algoritmo de Pons-Latapy (steps=20) - Dendograma



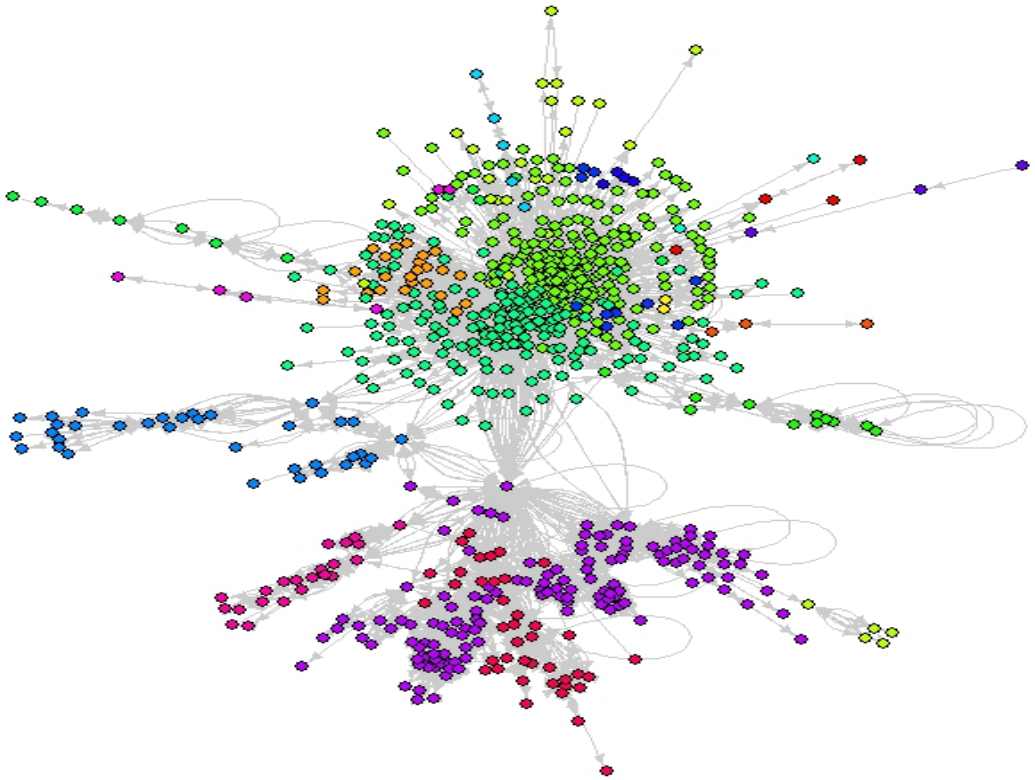
Caracterización de la cohesión de la red: detección de comunidades

- El método de **Reichardt-Bornholdt** es el más complejo de todos los anteriores ya que su idea está basada en la Estadística Física.
- En pocas palabras, la idea es encontrar la partición de los vértices que optimiza una función de dichos vértices denominada **energía**.
- De alguna manera, este tipo de algoritmos es similar a los métodos de partición de datos multivariantes, como k-medias.

Algoritmo de Reichardt-Bornholdt con método de Kamada y Kawai



Algoritmo de Reichardt-Bornholdt con método de Fruchterman y Reingold



¿Dónde seguir?

- **Modelos estadísticos para datos de redes**: modelos para generar ejes e inferencia sobre parámetros de dichos modelos dada una red de datos reales. Se pueden tener en cuenta los atributos o no.
- **Modelización y predicción para procesos en redes**: inferencia sobre determinadas características de la red en función de vértices y ejes.
- **Redes dinámicas**: ¿como evolucionan las redes?

Breve bibliografía

- Kolaczyk, E. D. (2009) Statistical analysis of network data: methods and models. Springer.
- Kolaczyk, E. D. and Csárdi, G. (2014) Statistical analysis of network data with R. Springer.
- Newman, M. E. J. and Girvan, M. (2004) Finding and evaluating community structure in networks. Physical Review E, 69, 026113.
- Pons, P. and Latapy, M. (2005) Computing communities in large networks using random walks. Lecture Notes in Computer Science, 3733, 284-293.
- Reichardt, J. and Bornholdt, S. (2006) Statistical mechanics of community detection. Physical Review E, 74, 016110.