



# MDMS - PEC1- Memoria

Adrián José Zapater Reig

Se pide:

1. Extraer datos de Twitter y crear una red a partir de ellos.
2. Realizar un análisis de la red.
3. Medir la asortabilidad de la red para 3 escenarios:
  - a. Todos los nodos de la red.
  - b. Cada clase identificada con un algoritmo de modularidad.
  - c. Cada clase identificada con un algoritmo de aprendizaje no supervisado ej. Clústering.

## Objetivo:

### Resumen

Analizar las relaciones de amistad de 9 corredores de F1 en Twitter para encontrar usuarios y comunidades homofílicas.

Realizar un estudio comparativo de 3 experimentos con niveles de agrupación distintos: sin agrupación, modularidad y clústering, para ver cual organiza los nodos en comunidades con mayor asortatividad y porqué.

### Expectativas

Puesto que la información proviene de 9 nodos que probablemente estén inter-relacionados, es de esperar que los nodos de nuestro grafo tengan una cohesión y densidad alta, y por lo tanto el experimento sin agrupación muestre un alto grado de asortatividad. A pesar de esto, es de esperar que las comunidades agrupadas por modularidad tengan una asortatividad aun mayor ya que los miembros de una misma comunidad tienden a tener conexiones mas densas entre si que con miembros de otras comunidades y esto podría ser por la homofilia.

Con respecto al 3er escenario, espero poder obtener metadatos suficientes y relevantes a través de la API de Twitter como para aplicar un algoritmo de clústering que maximice la homofilia entre los clústers.

## 1. Extracción de información

La homofilia es la tendencia de las personas a entablar amistad con personas con atributos similares. Para estudiarla he decidido utilizar las relaciones de amistad en Twitter entre 9 pilos de F1:

- '@Charles\_Leclerc'
- '@PierreGASLY'
- '@alo\_oficial'
- '@Carlossainz55'
- '@GeorgeRussell63'
- '@LewisHamilton'
- '@danielricciardo'
- '@LandoNorris'
- '@SchumacherMick'

### Diseño de la información a extraer

Para analizar esta información, he modelado las personas como vértices, uno por cada cuenta de Twitter, y las relaciones de amistad ("follow") como aristas dirigidas con origen en la persona que sigue ("follower") y destino en la persona que es seguida ("followee"). Por ejemplo, si @alo\_oficial sigue a @Carlossainz55, la arista irá desde @alo\_oficial a @Carlossainz55, pero no implica que la relación exista en sentido contrario.

Para poder estudiar la homofilia de este modelo necesitamos ver qué atributos tenemos para cada nodo y cuales pueden ser útiles para modelar la similitud entre nodos. Los atributos que nos ofrece la API de Twitter que considero mas interesantes son:

- Location – Ubicación de la cuenta.
- Num\_followers - Número de followers (seguidores).
- Num\_followings - Número de followings (personas a las que sigue).
- Num\_tweets - Número de Tweets.
- Is\_verified – Si tiene el flag de verificación de cuenta oficial o no.

id	Label	Interval	location	num_followers	num_followings	num_tweets	is_verified
1000609382	LoveForNicolex		#TeamNicoleScherzyUK	499	614	297	<input type="checkbox"/>
1001841554183606272	TorchM7		Queretaro, MX	51	277	103	<input type="checkbox"/>
100220864	BrundMars			43129385	93	4831	<input checked="" type="checkbox"/>
100312878	brandnewrock		Long Island, NY	137364	29	842	<input checked="" type="checkbox"/>
1003157563649560577	pgfanbase		Tokyo/東京	2546	239	39627	<input type="checkbox"/>

Ilustración 1 - Muestra de los datos extraídos

Location es probablemente el atributo mas interesante de los 5, aunque, como se puede ver en las ilustraciones 2 y 3, contiene un texto heterogéneo y no una lista cerrada de valores que nos permita analizar por país o región.

MSportMonday		At a race track near you...	6
henrywinter		At a game.	1
SophiaBush		Asking questions	1
isco_alarcon		Arroyo de la miel	5
BenBarnicoat		□round the □	5
CanalplusF1		Around the World !	1
daniclos		Around the world!	4
ValtteriBottas		Around the World	5
mitchevans_		Around the world	2
tiffanycromwell		Around the world	1
f1_hamilton		Around the world	2
teamhamilton01		around the UK :)	1
TeamCarlosSainz		Around the globe	5
virgilabloh		Around	5
HusKerrs		Arizona, USA	1
TGR_Arg		Argentina	1
vroomkart		Aprilia-Italy	3
natashabdnfield		anywhere	3

Ilustración 2 - Nombre de usuario en la columna izquierda y location en la columna derecha

Algunos usuarios utilizan este campo para almacenar URLs o lugares ficticios.

Label	Interval	location ▼	n
Codemasters		□□●□	1
AmberLoungeLtd		□□	9
rudygobert27		□□□	5
CDNThe3rd		□	4
UberFacts		□	1
TobyMoody		□	7
M_Bortolotti		□	4
Martial_Anthony		□□	4
YacineB15		□□□□	2
VancityReynolds		□□ & □□□□□□	1
pamfoundation		□□	1
_markgallagher		□□□□	2
unionbidxb		□□□□□ □□□□□□ □□□□□□ ...	5

Ilustración 3 - Muestra de Location con caracteres extraños.

Los atributos num\_followers, num\_followings y num\_tweets son útiles por si solos o combinados para crear métricas ordinales de la centralidad de un nodo.

Por ejemplo:

- Ratio followers/followings
- Ratio followers/followings ponderado por número de tweets

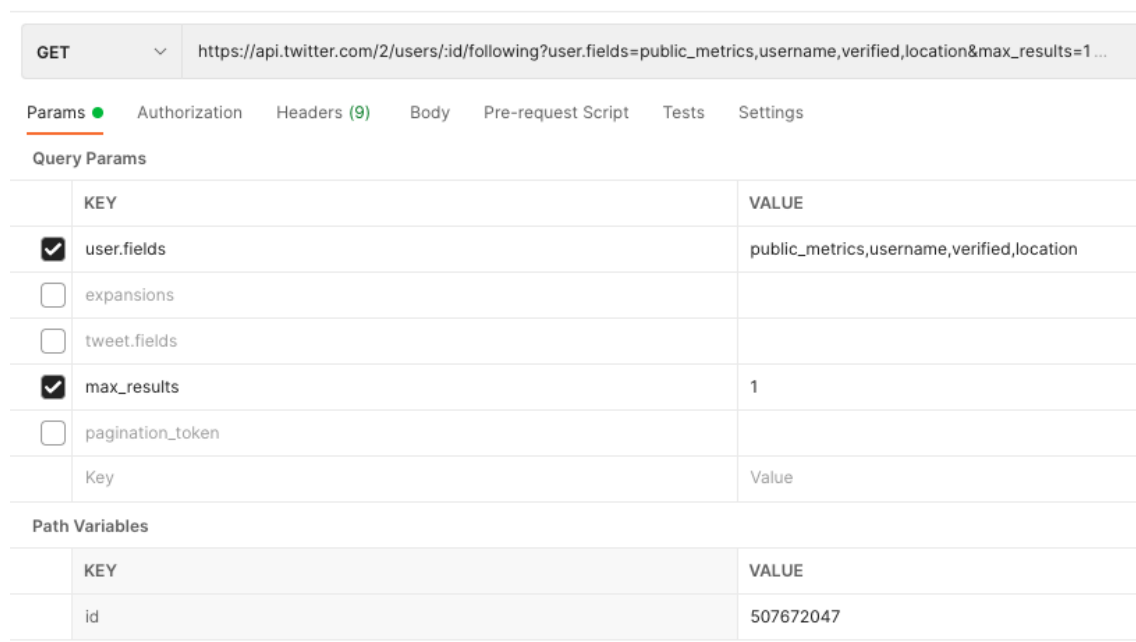
Por último, el flag is\_verified nos indica con True o False si la cuenta está verificada, lo que se puede interpretar como una señal de que es un nodo con mucha importancia y puede utilizarse para modelar la similitud entre nodos. 2 nodos con el flag a true serán similares (ratio de followers/followings alto).

Me hubiera gustado poder obtener información acerca de la fecha en la que un nodo empezó a seguir a otro para poder hacer un análisis dinámico del grafo, pero desgraciadamente ninguna de las APIs (ni siquiera por web-scraping) ofrece esta información. Como ya sabemos, los datos de los medios sociales a veces son incompletos y el analista está a merced de lo que el propietario quiera exponer en la API.

Tener esta fecha me hubiera permitido analizar la asortabilidad en 2 intervalos de tiempo distintos y analizar si aumenta o decrece en el tiempo.

### Extracción

Para extraer la información he hecho uso de la nueva [API REST v2 de Twitter](#) por su facilidad de uso y porque ya estoy familiarizado con la versión v1.1. Me he apoyado en la herramienta POSTMAN para preparar las llamadas y probarlas antes de codificarlo en python. El código empleado se entregará adjunto a la memoria.



GET ▼ https://api.twitter.com/2/users/:id/following?user.fields=public\_metrics,username,verified,location&max\_results=1 ...

Params ● Authorization Headers (9) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE
<input checked="" type="checkbox"/> user.fields	public_metrics,username,verified,location
<input type="checkbox"/> expansions	
<input type="checkbox"/> tweet.fields	
<input checked="" type="checkbox"/> max_results	1
<input type="checkbox"/> pagination_token	
Key	Value

Path Variables

KEY	VALUE
id	507672047

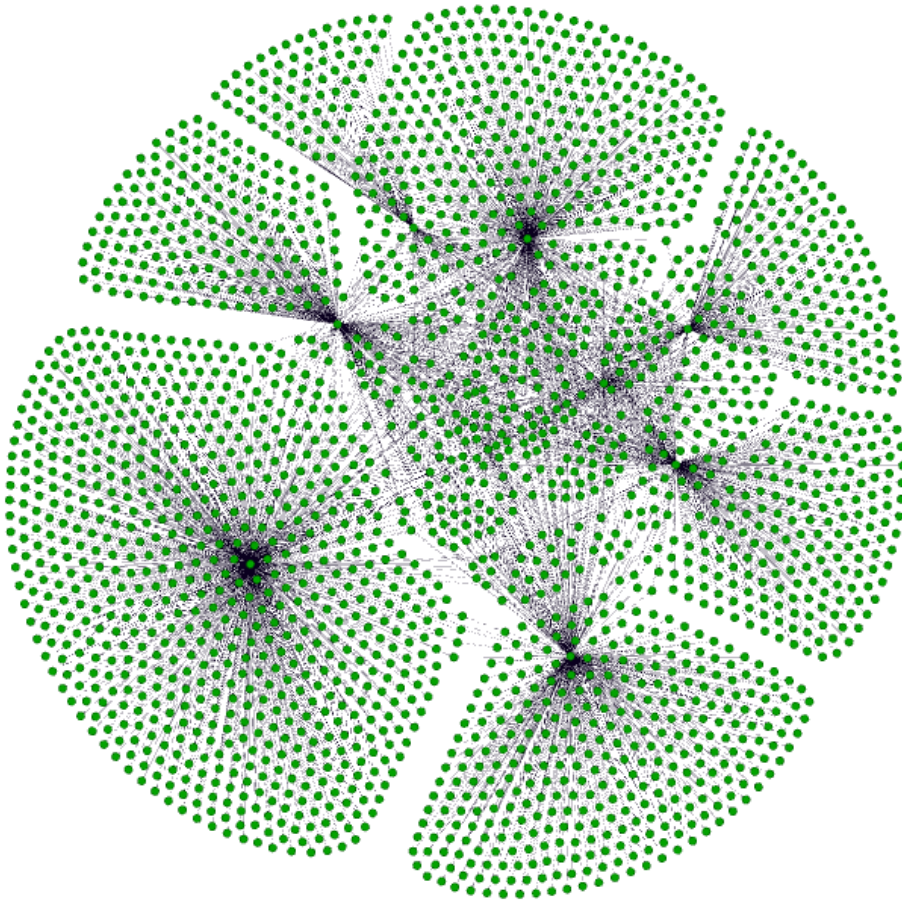
Ilustración 4 - Herramienta POSTMAN

Resultado de la extracción:

- Número de Vértices: 2336
- Número de Aristas: 3034

## 2. Análisis

En primer lugar, vamos a visualizar el grafo completo con el “layout” Fruchterman Reingold con los valores por defecto. Este “layout” junta los nodos que comparten una arista con una gravedad parametrizada, lo que nos ayudará a ver cuantos componentes tenemos y si hay comunidades claras.



*Ilustración 5 - grafo con layout Fruchterman Reingold*

Como podemos ver en la ilustración 5, tenemos un único componente (no hay comunidades aisladas) y se pueden identificar con facilidad por lo menos 7 comunidades claramente definidas. En el centro del grafo es más difícil identificar las comunidades claramente ya que los nodos están más interconectados.

Desde Gephi podemos calcular métricas generales a nivel de la red que nos ayudarán a entender la estructura de la misma. Veamos las más interesantes:

### Densidad

Al comienzo de la práctica dije que esperaba tener una red muy interconectada con una alta cohesión y densidad, pero como se puede ver en la imagen, me equivocaba. Gephi calcula una densidad de 0.001, que probablemente es el valor mínimo de la métrica, lo que quiere decir que estamos ante una red “sparse” con nodos muy poco conectados entre sí. Esto se debe a que sólo hemos capturado las relaciones salientes de los 9 pilotos de F1 y por lo tanto la mayoría de los nodos son de grado 1.

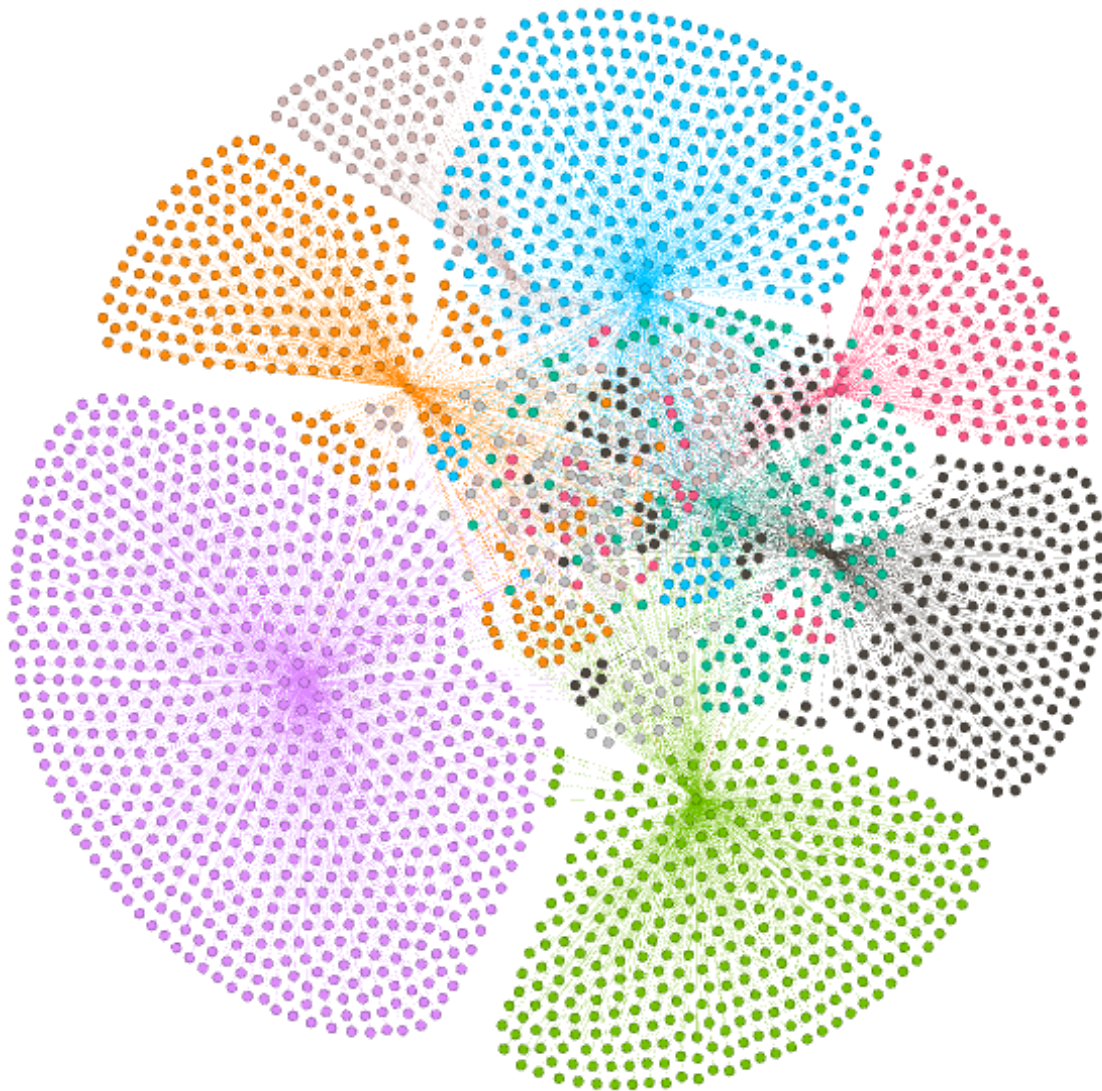


### Centralidad de Grado

La media de grado por por vértice es de 1.299, que sigue en la misma línea que la densidad: tenemos muchos nodos con una única arista y esto se debe a que nuestra información está incompleta: sólo hemos capturado las relaciones salientes de 9 pilotos de F1.

### Modularidad

La red tiene una modularidad de 0.63, lo cual es un valor bastante alto, y un total de 9 comunidades. Un valor alto de modularidad significa que, a nivel estructural, existe una división clara de comunidades.



*Ilustración 6 - Nodos de cada clase de modularidad coloreados con distintos colores*

En la ilustración 6 se puede apreciar como las comunidades están claramente separadas. Como es de esperar, el centro de cada comunidad es el piloto de F1.

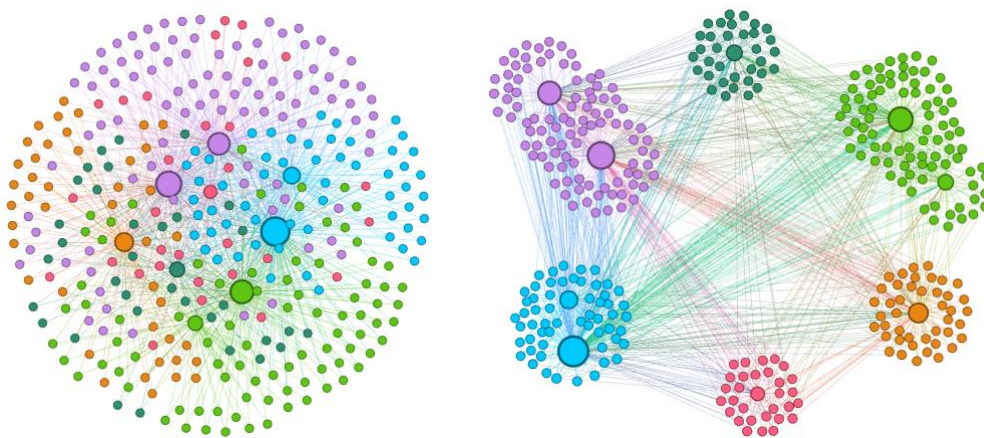
## Preprocesado

Ahora que conocemos la topología del grafo en crudo vamos a llevar a cabo unas tareas de limpieza y preprocesado para aumentar la relación señal-ruido.

Uno de problemas destacados que hemos encontrado es la gran cantidad de nodos de grado 1 que tenemos que no aportan nada de información y que se aglomeran alrededor de los nodos de los pilotos. Vamos a considerar que estos nodos son ruido y vamos a descartar los nodos con grado menor a 2.

Otro problema que tenemos es que nuestra información es limitada y sólo fluye en un sentido (del piloto a otro nodo), pero no porque las relaciones de amistad sólo existan en ese sentido, sino porque sólo hemos recuperado esa información (Twitter solo permite lanzar 15 consultas de seguidores cada 15 minutos, lo que sería 1.6 días para los 2300 usuarios), por lo que no tiene mucho sentido guardar información de la dirección de la amistad. Vamos a simplificar el problema y tratar la red como un grafo no dirigido.

La siguiente imagen muestra el resultado visualizado con “layout” Fruchterman Reingold, coloreado por las comunidades de la nueva modularidad (6) con tamaño por nodo proporcional al grado del nodo:



*Ilustración 7 - Nodos con grado mayor o igual a 2 coloreados por modularidad*

En la imagen de la derecha he separado las comunidades.

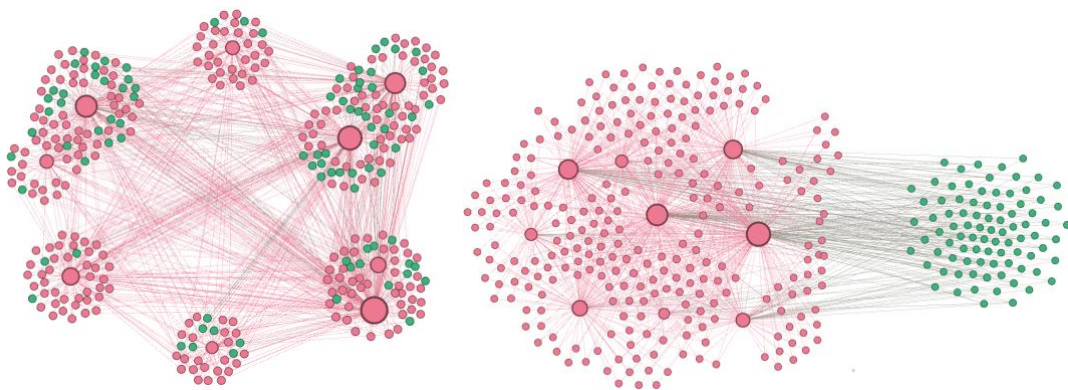
El resultado del preprocesado nos deja un grafo con una densidad de 0.016 y una media de grado por nodo de 5.819, ambos muy superiores al grafo en crudo, y un grafo mas manejable para el análisis de asortabilidad.

### Modelado de la similitud

Las métricas que hemos visto hasta ahora nos han ayudado a tener una visión general del grafo con el que estamos trabajando, pero no nos sirven para estudiar la asortividad, o por lo menos no directamente.

Para estudiar la asortividad tenemos que encontrar una manera de modelar la similitud entre nodos con los atributos que tenemos.

En primer lugar, vamos a estudiar el atributo `is_verified`. Este atributo booleano identifica los nodos que tienen el “check” de verificado en Twitter. Este check lo concede Twitter a las cuentas mas influyentes (famosos, empresas, etc...). En Gephi podemos dibujar el grafo coloreando los nodos con True en rosa y los nodos con False en Verde para ver la distribución:



*Ilustración 8 - Atributo `is_verified` (True = Rosa)*

La ilustración 9 muestra la organización por comunidad con el flag `is_verified` coloreado en la izquierda y la misma información, pero organizada por flag, en la derecha.

En la imagen de la izquierda vemos que el atributo `is_verified` está repartido entre todas las comunidades, no parece que sea algo propio de una comunidad en concreto. La imagen de la derecha nos muestra que los nodos con `is_verified` a true tienden a relacionarse con otros nodos con ese atributo, mientras que los nodos con `is_verified` a false tienden a relacionarse también con los que tienen el flag a True.

Este comportamiento parece mas propio de la influencia que de la homofilia, y aunque nos permite modelar la asortividad, no es lo que buscamos. Descartamos el atributo.

Después de probar varias combinaciones de atributos extraídos de twitter y las métricas proporcionadas por Gephi, está claro que tenemos una red con relaciones debidas a la influencia muy marcadas. Casi cualquier métrica de centralidad nos identifica 2 grupos muy marcados: Los pilotos de F1 y el resto de nodos.



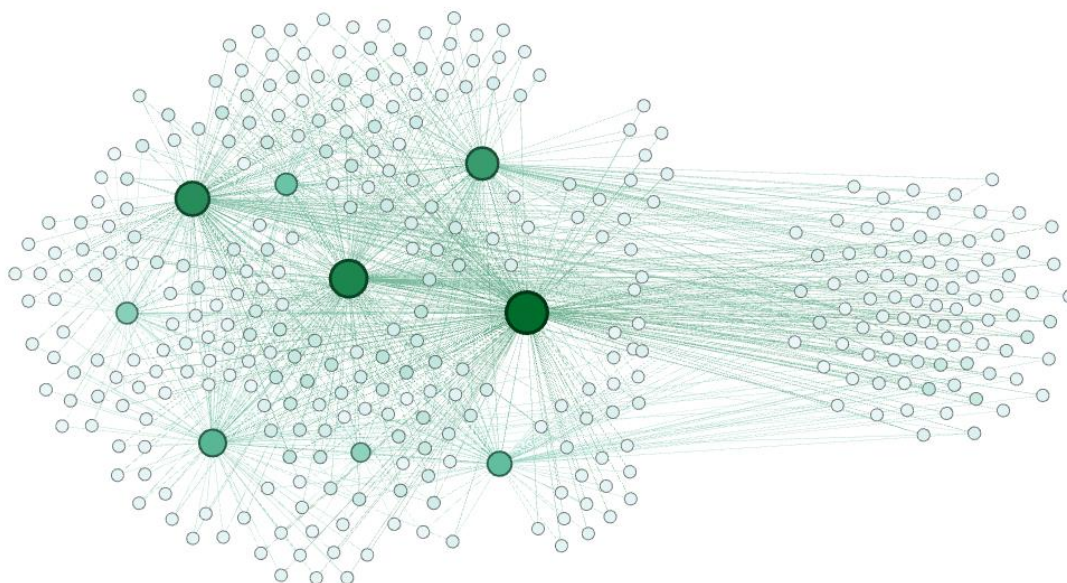


Ilustración 9 - Métrica de autoridad del algoritmo de HITS. En verde los pilotos de F1

Pero todavía queda un atributo mas: Location. En la sección 1 hemos comentado este atributo y las dificultades que trae consigo dado lo heterogéneos que son sus valores. Para paliar estas dificultades he creado 4 columnas a partir de esta: "is\_from\_esp", "is\_from\_uk", "is\_from\_monaco", "is\_from\_france". Cada una de estas columnas de tipo booleano identifica si un nodo es de una de las 4 nacionalidades mas frecuentes en los datos mediante expresiones regulares.

.*(UK uk (L l)ondon (E e)ngland (U u)nited (K k)ingdom).*
---

Tabla 1 - Expresión regular para identificar a los nodos de inglaterra.

Partiendo de la premisa de que las personas tienden a relacionarse mas con otras personas de la misma nacionalidad, y asumiendo que la ubicación que nos ha proporcionado Twitter (que es la que tiene publicada cada usuario) es la nacionalidad real de cada usuario, no es descabellado asumir que las relaciones entre nodos que compartan la misma nacionalidad son debidas a la similitud que comparten ambos nodos y en concreto a la homofilia.

Si además encontramos que tenemos nodos que no son centrales o influyentes que han desarrollado una relación con otros que tampoco lo son, podemos asumir con cierto grado de confianza que la influencia no ha jugado un papel principal en la creación de estas relaciones y atribuir el mérito a la homofilia.

Volvemos a Gephi a comprobar la hipótesis. Eliminamos los nodos que no tengan location informado y filtramos las aristas para mostrar sólo las que comiencen y terminen en un nodo con la misma nacionalidad.

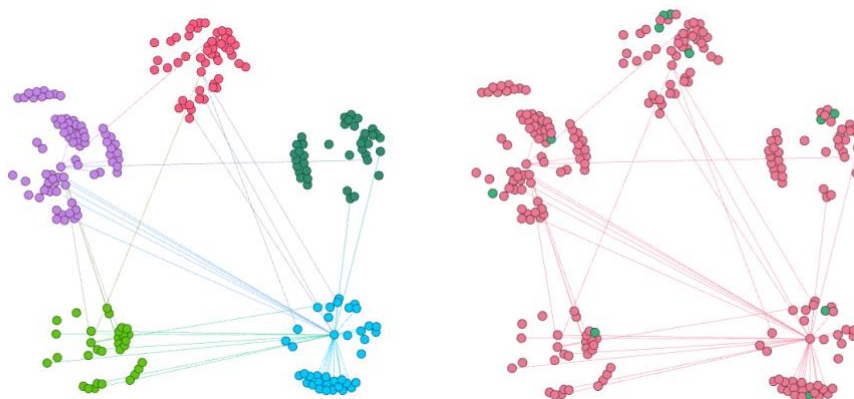


Ilustración 10 -(Izquierda) coloreado por modularidad (Derecha) coloreado por `is_from_esp`. Verde = True

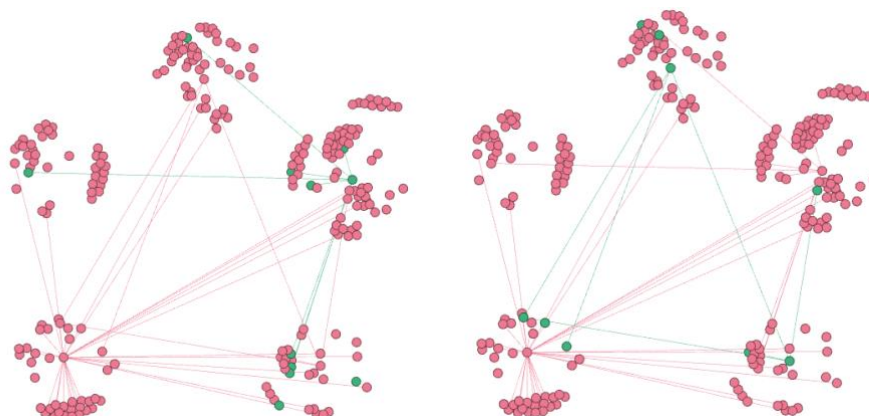


Ilustración 11 - (Izquierda) Coloreado por `is_from_france` (Derecha) Coloreado por `is_from_monaco`

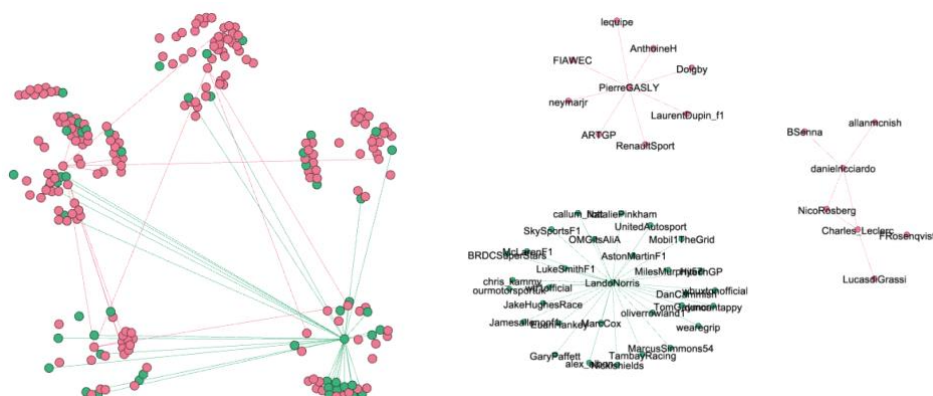


Ilustración 12 – (Izquierda) Coloreado por `is_from_uk`. (Derecha) Grafo reorganizado coloreado por `is_from_uk`

Si analizamos las ilustraciones 10, 11 y 12 podemos ver que efectivamente, los nodos marcados en verde (con los flags de nacionalidad) solo se relacionan con los nodos de la misma nacionalidad. Hay unos pocos “outliers” que podrían ser nacionalidades no publicadas o erróneas.

Un caso muy interesante es el de la nacionalidad española, como ni Carlos Sainz ni Fernando Alonso tienen su nacionalidad correctamente informada, y como solo tenemos las relaciones que parten desde un piloto de F1, los nodos con nacionalidad española quedan huérfanos de relación.

Id	Label	Interval	location
507672047	alo_oficial		instagram: fernandoa...
353786894	Carlossainz55		planet earth

Ilustración 13 - Nodos de Carlos Sainz y Fernando Alonso

La segunda premisa que planteamos al comienzo del análisis no se ha cumplido, todos los nodos que no son centrales o con baja influencia (nodo que no es piloto de F1) tienen una relación con un piloto de F1, pero esto se debe a la información que tenemos y no a la influencia, dado que sólo tenemos las relaciones salientes de los 9 pilotos de F1.

### 3. Medición de la asortavilidad

#### Escenario 1 – Asortavilidad global de la red.

Para medir la asortavilidad global de la red, vamos a necesitar generar la matriz de adyacencia de la red y calcular el coeficiente de correlación de Pearson sobre ella. Partiremos de la red que teníamos en la ilustración 9 (Nodos con mas de 1 grado y que tengan location informado). Junto a la memoria se adjunta un notebook con el código para generar la matriz de adyacencia y el cálculo del coeficiente de correlación de Pearson.

La media de coeficiente de correlación de Pearson para este escenario es de: 0.34938 Utilizaremos este escenario como baseline para la comparativa con los otros 2.

#### Escenario2 – Clases de modularidad

Para este experimento vamos a aplicar el algoritmo de modularidad de Gephi y calcular la matriz de adyacencia y coeficiente de correlación de Pearson para cada una de las clases.

## Results:

Modularity: 0.283

Modularity with resolution: 0.283

Number of Communities: 6

Ilustración 14 - Resultado de algoritmo de modularidad

El algoritmo de modularidad identifica 6 comunidades. Si seguimos los mismos pasos que en el apartado anterior para cada clase, obtenemos los siguientes resultados:

Clase 0 = 0.5425332465145296

Clase 1 = 0.935501209352324

Clase 2 = 0.9574517285235289

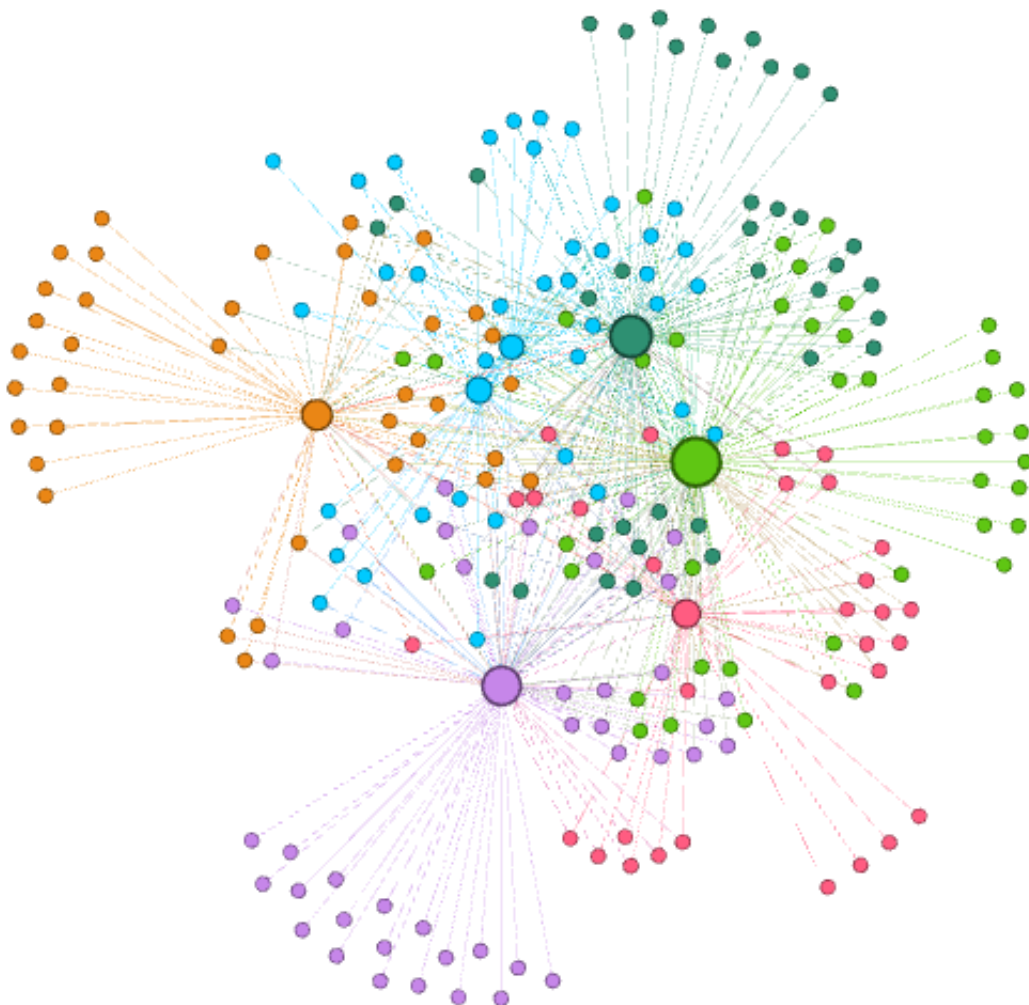
Clase 3 = 0.9705898491083674

Clase 4 = 0.9245562130177513

Clase 5 = 0.9626201495194019

Tenemos una media de coeficiente de correlación de Pearson muy alta porque dentro de cada clase, todos los nodos están contactados a un único nodo central (el corredor de F1), por lo que todos los vectores que compara el algoritmo de Pearson son 0 excepto para la posición del nodo central.

0.54 es el coeficiente mas bajo porque esta clase tiene 2 nodos centrales como se puede ver en los nodos gordos azules de la ilustración 15. Esta clase, al tener 2 nodos centrales, tiene un coeficiente que es aproximadamente la mitad del resto.



*Ilustración 15 - Grafo coloreado por módulos con tamaño definido por grado del nodo*

### Escenario3 – Clustering

Para el tercer escenario, vamos a aplicar el algoritmo de clustering KMEANS para que separe los nodos en varios clústers de nodos similares. Alimentaremos el algoritmo con los atributos que hemos estado explorando anteriormente: is\_from\_uk, is\_from\_spain, is\_from\_monaco y is\_from\_france.

	Id	is_from_uk	is_from_esp	is_from_france	is_from_monaco
0	716238475	0	0	0	0
1	188039706	0	0	0	0
2	437610627	0	0	0	0
3	248807431	0	0	0	0
4	138456571	0	0	0	0

Ilustración 15 - utilizado para alimentar el modelo

El algoritmo de KMeans necesita que le digamos el número de clústers que tiene que utilizar. Utilizaremos el método de “elbo” para seleccionar el número de clústers idóneo.

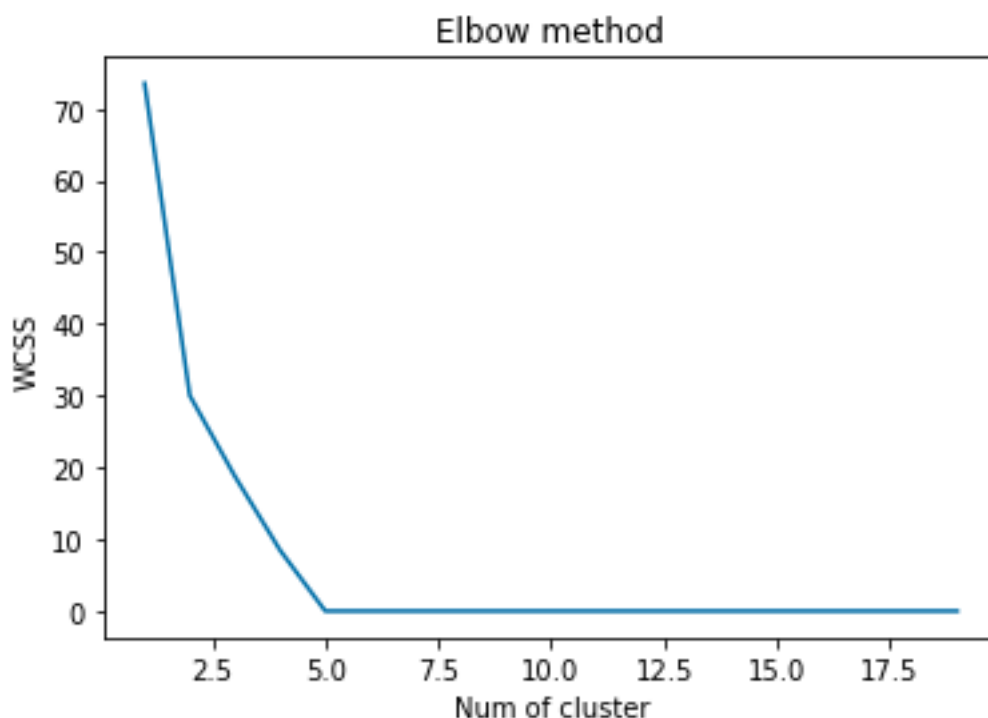


Ilustración 16 - Método de Elbow.

Podemos ver como el codo (o elbow) está en 5 clusters. Parametrizamos el algoritmo y le pasamos los datos.



Es muy importante que nos aseguremos de no alimentar el algoritmo con una columna con el id de los nodos, ya que lo tendrán en cuenta a la hora de hacer los clústers. Separamos el id y lo volvemos a incorporar a la salida del algoritmo.

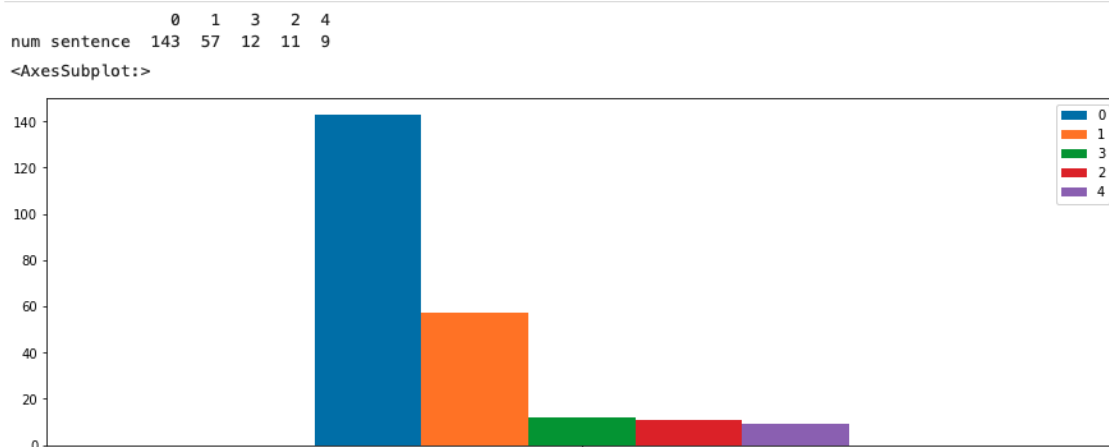


Ilustración 17 - Distribución del algoritmo de clústering

Vemos como hay un clúster principal con 140 nodos, otro mediano con 57 nodos y 3 mas pequeños con menos de 15 nodos.

Ahora deberemos seguir los mismos pasos que en el apartado anterior y calcular la correlación de Pearson: Exportaremos las aristas de Gephi, cruzaremos con los nodos de cada clúster y calcularemos la matriz de adyacencia para cada uno de los clústers por separado teniendo en cuenta únicamente a los nodos de ese clúster.

Resultados:

Clúster 0: 0.3253232053692988

Clúster 1: 0.3134698464103467

Clúster 2: 0.08846759755393964

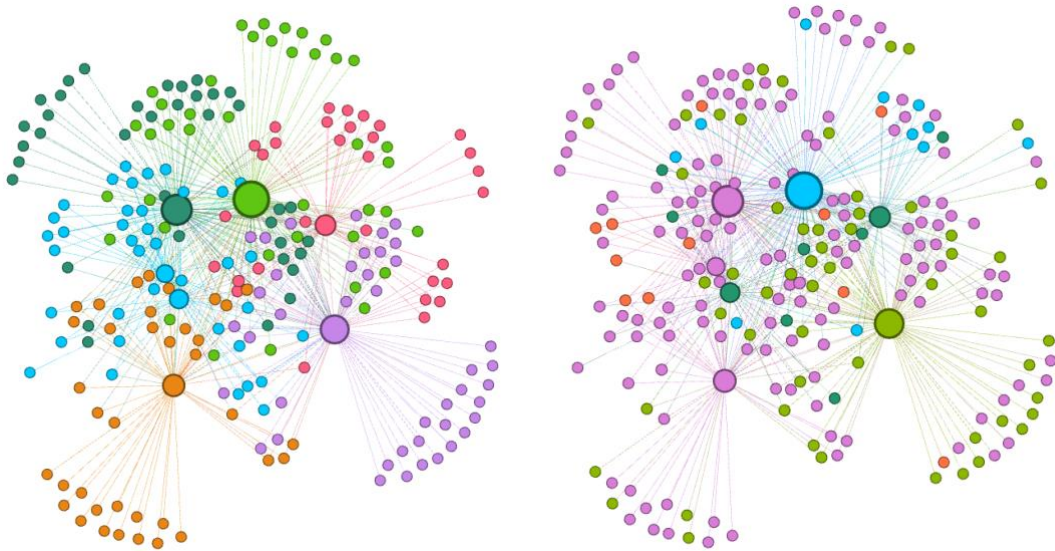
Clúster 3: 0.19373641666946914

Clúster 4: 0.12523828785894103

La correlación de Pearson en general no es muy alta, ninguno de los clústers supera al primer escenario, lo que quiere decir que o los atributos que hemos elegido no son los mas indicados o nos falta incluir mas registros. Como podemos ver en los resultados, los 2 clústers que tienen mas nodos (clúster 0 tiene 143 y el clúster 1 tiene 57) si que tienen un coeficiente mas alto, lo que apoyaría la teoría de que con mas registros obtendríamos mejores resultados.

## 4. Conclusiones

El escenario que mejor ha agrupado los nodos ha sido el escenario 2. Si utilizamos el escenario 1 como baseline del experimento, podríamos decir que el algoritmo de modularidad si que ofrece una mejora a la hora de agrupar nodos similares frente a agruparlos de manera aleatoria, mientras que el algoritmo de clústering aplicado no. Esto podría ser por la falta de información, tener sólo 232 nodos ha impactado muy negativamente en el algoritmo de aprendizaje automático.



*Ilustración 18 - Coloreado por modularidad (Izquierda) y por Clúster (Derecha)*