Doble Grado en
Matemáticas y Estadística

TRABAJO FIN DE GRADO

Análisis estadístico de Redes Sociales

Juan Pedro Campos de los Reyes

Sevilla, Junio de 2018

Índice general

Prólogo . . . III
Resumen . . . V
Abstract . . . VI

1. Introducción **1**
1.1. Redes Sociales . . . 1
1.2. Twitter . . . 3
1.3. Lenguaje R . . . 3

2. Obtención de datos **5**
2.1. Modelización de datos . . . 10
2.1.1. Modelo Booleano . . . 10
2.1.2. Modelo de Grafos . . . 11
2.1.2.1. Grafos . . . 12

3. Tratamiento de datos **15**
3.1. Matriz término-documento . . . 15

4. Aplicaciones **19**
4.1. Nubes de palabras . . . 19
4.2. Clustering . . . 23
4.2.1. Distancias . . . 24
4.2.1.1. Distancia Euclídea . . . 24
4.2.1.2. Distancia del Máximo . . . 25
4.2.1.3. Distancia Manhattan . . . 25
4.2.1.4. Distancia de Mahalanobis . . . 25
4.2.1.5. Distancia de Minkowski . . . 25
4.2.1.6. Ditancia valores binarios . . . 25
4.2.2. Métodos particionales . . . 26
4.2.2.1. Algoritmo de k-medias . . . 26
4.2.2.2. Algoritmo de k-medioides . . . 27
4.2.3. Métodos Jerárquicos . . . 29
4.3. Geolocalización . . . 37
4.4. Hashtags . . . 40
4.4.1. Usuarios más activos . . . 40
4.4.2. Usuarios más retuiteados . . . 42
4.4.3. Usuarios más citados . . . 43
4.4.4. Resumen . . . 44

5. Análisis de Sentimientos **47**
5.1. Algoritmos de clasificación . . . 57
5.1.1. Clasificador Naives-Bayes . . . 57
5.1.2. Clasificador de Máxima Entropía . . . 57
5.1.3. Support Vector Machines . . . 57

6. Redes Multicapas **59**
6.1. Terminología y modelo . . . 59
6.2. Modelos relacionados . . . 61
6.2.1. Redes multiplex . . . 61
6.2.2. Redes multimodo y multinivel . . . 61
6.2.3. Redes Temporales . . . 62
6.3. Conjunto de datos . . . 63
6.4. Recopilación y preprocesamiento de datos . . . 64
6.5. Simplificación de red . . . 65
6.5.1. Aplanamiento y proyección . . . 65
6.6. Visualización de redes multicapas . . . 66
6.6.1. Cuatro enfoques principales . . . 66
6.6.2. Visualización de métricas . . . 67
6.6.2.1. Coordenadas paralelas . . . 67
6.6.2.2. Gráfico de cocurrencia de bordes . . . 68
6.6.3. Visualización de la estructura . . . 68
6.6.3.1. Corte de capas . . . 69
6.6.4. Visualización de red simplificada . . . 69
6.7. Patrones de borde . . . 70
6.7.1. Predicción del borde . . . 70
6.8. Ejemplos R . . . 72

Bibliografía **83**

Prólogo

El trabajo de fin de grado presentado a continuación lleva el título de "Análisis estadístico de Redes Sociales". Este trabajo ha sido escrito como parte de los requisitos de graduación para el programa del doble grado de Matemáticas y Estadística de la Universidad de Sevilla. El periodo de investigación y redacción de este trabajo de fin de grado ha durado desde septiembre de 2017 hasta junio de 2018.

El proyecto lo he llevado a cabo debido al gran interés que despierta en mí las aplicaciones de las matemáticas en las redes sociales, ya que estas son usadas por la gran mayoría de la población.El trabajo de investigación ha sido tedioso debido a los diferentes problemas que me han ido surgiendo relacionado con los tweets, Afortunadamente, mi tutor siempre ha estado disponible y dispuesto a ayudarme con todas mis cuestiones, que al final hemos logrado resolver.

Me gustaría, por tanto, dar las gracias a mi tutor por su excelente orientación y soporte durante todo el proceso de realización de mi trabajo.

A mis padres y mi hermano que se merecen un especial agradecimiento: gracias, gracias por apoyarme durante tantos años de estudios y por todos aquellos consejos que me daís, sin ustedes no hubiera tenido la fuerza para llegar hasta donde hoy estoy. Gracias también a mi tío Emilio, por ser un referente académico y por apoyarme desde siempre en mis estudios y trabajos.

Espero que disfruteis de la lectura.

Juan Pedro Campos

Resumen

Hoy en día las redes sociales poseen un gran impacto en la sociedad, llegando a un punto en el que un gran porcentaje de la población utiliza al menos una red social, a través de la cual interactúa de alguna manera con las demás personas, no solo las personas sino que muchas empresas toman partido en ello creando perfiles con los que relacionarse y promocionar sus productos o tratar con clientes. Es por ello que ha surgido un interes en intentar analizar las redes sociales apareciendo aplicaciones estadísticas que se encargan de ello facilitando una gran cantidad de información respecto a todo lo que ocurre dentro de estas. En este trabajo vamos a tratar algunas técnicas estadísticas con las que podemos analizar varios temas de la red social "Twitter".

En el primer capítulo realizaremos un breve resumen sobre la historia de las redes sociales y su evolución en el tiempo así como una descripción de la red social Twitter y el software que vamos a usar para el análisis.

En el segundo capítulo explicaremos como podemos obtener datos de la red social directamente usando el software R y la modelización de estos datos obtenidos asi como un ejemplo gráfico de su interpretación.

El tercer capítulo consta del tratamiento de datos, aquí veremos como podemos tratar los datos obtenidos en el capitulo anterior con el fin de realizar un posterior análisis. Además realizaremos un gráfico de frecuencias sobre unos datos ya tratados.

El cuarto capítulo, que es el mas denso, trata sobre las diferentes aplicaciones que podemos hacer con los datos tratados anteriormente, podemos ver desde gráficos de nubes de palabras y agrupamientos en clústeres hasta gráficos de geolocalización y estudios sobre hashtags.

En el quinto capítulo realizamos una de las aplicaciones más usadas en el campo del análisis de redes sociales como es el análisis de sentimientos de los tweets. Aquí podremos ver una clasificación tanto en sentimientos como en polaridad de los datos que obtendremos sobre una conocida empresa de ropa.

Por último veremos una introducción al mundo de las redes multicapas donde desarrollaremos una base teórica sobre ello para terminar viendo algunos ejemplos de lo que podemos hacer usando el software R.

Abstract

Today social networks have a great impact on society, reaching a point where a large percentage of the population uses at least one social network through which interacts in some way with other people, not just people but many companies take sides in creating profiles with which to relate and promote their products or deal with customers. That is why there has been an interest in trying to analyze social networks by appearing statistical applications that take care of it by providing a large amount of information about everything that happens within them. In this work we will discuss some statistical techniques with which we can analyze various topics of the social network "Twitter".

In the first chapter we will make a brief summary about the history of social networks and their evolution over time as well as a description of the social network Twitter and the software that we will use for the analysis.

In the second chapter we will explain how we can obtain data from the social network directly using the R software and the modeling of this data obtained as well as a graphic example of its interpretation.

The third chapter consists of the data processing, here we will see how we can treat the data obtained in the previous chapter in order to carry out a later analysis. In addition we will make a graph of frequencies on some data already treated.

The fourth chapter, which is the most dense, deals with the different applications that we can do with the data discussed above, we can see from graphics of word clouds and groupings in clusters to graphs of geolocation and studies on hashtags.

In the fifth chapter we made one of the most used applications in the field of social network analysis such as the analysis of feelings of tweets. Here we can see a classification of both feelings and polarity of the data we will obtain about a well-known clothing company.

Finally we will see an introduction to the world of multilayer networks where we will develop a theoretical basis about it to finish seeing some examples of what we can do using the software R.

Capítulo 1

Introducción

1.1. Redes Sociales

Si bien para muchos las redes sociales pueden llegar a ser un servicio moderno con escasa trayectoria en la web, debido a que la mayor explosión por el furor de las mismas surgió en los últimos años logrando una verdadera masificación en su uso, lo cierto es que su origen se remonta a más de una década.

Tras todos estos años de innovación continua de internet, las redes de interacción social han ido aumentando su peso llegando a ser uno de los elementos más difundidos, ofreciendo a sus usuarios un lugar virtual donde desarrollar comunicaciones de manera constante.

Con la llegada de los dispositivos móviles, ya sean smartphones u ordenadores portátiles, el uso de las redes sociales esta marcando una nueva tendencia en comunicación ya que ha posibilitado el uso de estas de manera remota no solo a traves de su computadora personal.

Un pequeño repaso a la historia de las redes sociales en el tiempo:

- **1971** cuando se envia el primer mail entre dos ordenadores estando juntos uno al lado del otro.
- **1994** se funda *GeoCities*, una de las primeras redes sociales de internet tal y ocmo hoy la conocemos. La idea era que los usuarios crearan paginas web y que las alojarán en determinados sitios según su contenido. (Hollywood, Wallstreet, etc.)
- **1995** *TheGlobe.com* da a sus usuarios la posibilidad de personalizar sus propias experiencias online publicando su propio contenido e interactuando con otras personas con intereses similares.
- **1997** se lanza *AOL Instant Messenger* un cliente de mensajería instantánea de America, que hoy en día comparte con Windows Live Messenger y Yahoo! Messenger los principales software usados para comunicarse.
- **1997** se crea *Sixdegrees.com* , que permite la creación de perfiles y listado de amigos.
- **2000** estalla la burbuja de internet.
- **2002** se lanza el portal *Friendster*, pionero en la conexión online de “amigos reales”, alcanzando 3 millones de usuarios en solo tres meses.

- **2003** se inaugura *MySpace*, se crea como una "copia" de Friendster.
- **2004** se lanza *Facebook* concebida originalmente como una plataforma para conectar estudiantes universitarios.
- **2006** se inaugura la red de microblogging *Twitter*.
- **2008** Facebook adelanta a MySpace como red social lider en cuanto a visitantes unicos mensuales.
- **2011** Facebook tiene 600 millones de usuarios repartidos por todo el mundo, MySpace 260 millones, Twitter 190 millones y Friendster apenas 90 millones.

Esta enorme tendencia de aumento no se frenó en 2011 sino que ha seguido aumentando con la incorporación de nuevas redes sociales con distintas características. Las redes sociales están actualmente extendidas por todo el mundo con Facebook como la mayor red mundial de este tipo. Es raro el estudiante de instituto o universitario que no utiliza habitualmente Facebook.

Estos sistemas se han hecho tan dominantes en las comunicaciones que los jóvenes apenas utilizan ya el correo electrónico. De sixdegrees a Friendster, de Friendster a Facebook, las redes sociales se han convertido en algo familiar y omnipresente de Internet.

Existen varios tipos de redes sociales categorizadas según público,objetivo y/o temática:

Redes sociales horizontales

Son aquellas dirigidas a todo tipo de usuario y sin una temática definida. Los ejemplos mas representativos del sector son Facebook, Twitter, etc.

Redes sociales verticales

Concebidas sobre un eje temático agregador donde su objetivo es el de congregar en torno a una temática definida a un colectivo concreto. Estas redes se pueden especializar como:

- *Redes sociales verticales profesionales* dirigidas a generar relaciones profesionales entrte los ususarios. Como por ejemplo: Viadeo, Linked In, etc.
- *Redes sociales verticales de ocio* congregar colectivos que desarrollan actividades de ocio, deporte, usuario de videojuegos,fans,etc. Como por ejemplo: Wipley, Minube Dogster, Last.FM y Moterus.
- *Redes sociales verticales mixtas* ofrecen a usuarios y empresas un entorno específico para desarrollar actividades tanto profesionales con personales en torno a sus perfiles; Yuglo, Unience, PideCita, 11870.

Viendo el impacto que tiene hoy en día las redes sociales en nuestra sociedad, surgío el análisis de estas mismas ya que conocer con detalle las distintas relaciones entre usuarios, las opiniones y sentimientos de estos mismos sobre procesos comerciales, sociales, empresariales es una información muy valiosa si queremos por ejemplos aumentar los beneficios de una empresa, centrar la campaña politica de un partido en una zona, encontrar personas que ejerzan gran influencia sobre las opiniones de los demas usuarios,etc.
Por ello, vamos a estudiar el análisis estadísstico que se esconde tras las redes sociales, más especificamente de Twitter.

1.2. Twitter

Twitter es un servicio de microblogging con sede en San Francisco, fue creado en 2006 como ya hemos dicho antes,desde entonces ha ganado tanta popularidad hasta llegar hoy en dia a tener 500 millones de usuarios, generando 65 millones de tuits al dia y más de 800.000 peticiones de búsqueda. La red permite enviar mensajes de texto plano de corta longitud, originalmente de 140 caracteres aunque se amplió a 280, los denominados tweets.

El elemento principal de la red Twitter son los tweets cuyo contenido puede ser de una gran infinidad de temas, entre los cuales destacan las conversaciones entre usuarios así como comentarios generales sobre diversos temas de la sociedad. Esto nos lleva a centrarnos en la información que recae en los tweets, aunque también existe información importante que no se enfoca en los tweets, como son los seguidores de los usuarios, la localización de los usuarios.
Dentro de los tweets los usuarios pueden "categorizarlos" a partir de una etiqueta (#) el hashtag, además los usuarios pueden nombrar a otros usuarios para que interacctuen con ellos de manera que añadimos (@) y el nombre del usuario al que queremos nombrar.

En la actualidad existen personas, denominadas twitstars, que son usuarios que poseen un gran número de seguidores, consiguiendo que sus tweets sean visto por muchas personas. Estos usuario son muy influyentes en nuestra sociedad llegando al punto que las empresas contratan a dichas personas para que publiquen tweets en referencias a sus productos, consiguiendo una nueva forma de hacer publicidad. Por ello también estudiaremos la importancia o el nivel al que pueden llegar a influenciar este tipo de usuarios.

Para todo este estudio vamos a utilizar un lenguaje de programación enfocado al análisis estadístico, R.

1.3. Lenguaje R

R es un entorno de programación para el análisis estadístico y gráfico que surge como dialecto libre del lenguaje S, desarrolladoen la Universidad de Auckland en 1993.

Entre una gran infinidad de usos, R destaca por estar orientado a las estadísticas ya que proporciona un amplio abanico de herramientas con las que poder trabajar, además gracias a su capacidad gráfica podemos quenerar gráficos con alta calidad por complejos que sean y en estos últimos años también es muy usado en la minería de datos debido a la potencia que tiene.

R proporciona una amplia variedad de técnicas estadísticas (modelado lineal y no lineal, pruebas estadísticas clásicas, análisis de series de tiempo, clasificación, agrupamiento, ...) y gráficos, y es altamente extensible ya que se puede extender a través de paquetes. Además de los paquetes suministrados con la distribución R, muchos más están disponibles a través de la familia de sitios de Internet de CRAN que cubren una amplia gama de estadísticas modernas.

Capítulo 2

Obtención de datos

Una vez visto la importancia que recae sobre el análisis estadístico de las redes sociales vamos a explicar como podemos obtener datos de la red social Twitter, no solo obtendremos tweets de diferentes usuarios sino que obtenedremos más información de estos, como son los seguidores, situación geográfica, etc.

Para la extracción de la información vamos a usar un proceso API de arquitectura REST, es decir, un interfaz entre sistemas que use HTTP para obtener datos o generar operaciones sobre esos datos en todos los formatos posibles, como XML y JSON.

Todas las grandes y conocidas redes sociales, al igual que cientos de empresas internacionales, generan negocio gracias al sistema REST (Transferencia de Estado Representacional) y Twitter, como Instagram, YouTube o Facebook, no iba a ser menos. Cuando hablamos de un sistema REST, nos referimos a cualquier interfaz entre sistemas que use HTTP para obtener datos o generar operaciones sobre esos datos en todos los formatos posibles, como XML y JSON.

La API REST de Twitter nos permite acceder, leer y escribir datos de Twitter, es decir, a través de ella se pueden crear nuevos tweets y leer el perfil de los usuarios y datos de sus seguidores (entre otros datos de cada perfil), pues identifica las distintas aplicaciones de Twitter y los usuarios que se registran usando la autenticación y autorización OAuth. Las respuestas de la API REST de Twitter están en formato JSON.

Veamos las características básicas de la API de Twitter:

- La API de Twitter cuenta con cuatro "objetos" principales: Tweets, Usuarios, Entidades y Lugares.
- Tiene limitaciones diarias para las llamadas y cambios en la API, para proteger a Twitter de abusos. En concreto, la limitación se realiza principalmente por usuario, o mejor dicho, por token de acceso de usuario. Los límites de frecuencia se dividen en intervalos de 15 minutos y todos los criterios de valoración requieren autenticación, por lo que no son posibles las llamadas a la API no autenticadas.
- La API está basada en HTTP (sobre SSL), por lo que los procesos que requieran un método HTTP determinado devolverán un error, si no realiza la solicitud correcta.

Pasemos a identificarnos mediante nuestras credenciales, para ello vamos a cargar las librerías de R específicas y a introducir los datos necesarios, como las credenciales son privadas no vamos a mostrar las usadas:

```
library(twitteR)
library(ROAuth)
library(RCurl)
library(httr)
library(tm)

requestURL <- "https://api.twitter.com/oauth/request_token"

accessURL <- "https://api.twitter.com/oauth/access_token"

authURL <- "https://api.twitter.com/oauth/authorize"

consumerKey<-"**********"

consumerSecret<- "**********"

as<-  "**********"

at<- "**********"
```

Una vez introducidos nuestros datos personales sobre la API, pasamos a autentificarnos:

```
# Establecemos conexión con autentificación directa.
# Esta función establece las credenciales para poder hacer la conexión.

# setup_twitter_oauth(consumer_key=consumerKey,
# consumer_secret=consumerSecret, access_token=at, access_secret=as)

source("credenciales.R")
```

```
## [1] "Using direct authentication"
```

Una vez cargadas las librerías y realizada la autentificación pasamos a realizar varias pruebas sobre lo que podemos hacer.
En primer lugar la **Extracción de datos**.

Podemos realizar tanto búsquedas de palabras como tweet de un usuario específico:

```
# Búsqueda de tweets con la palabra ("Matematicas")
# condicionando idioma (español) y número de tweets máximos (40).

tweets<-searchTwitter("Matematicas",n=40,lang = "es")
cat("El tipo de datos obtenido es: " ,typeof(tweets))
```

```
## El tipo de datos obtenido es:  list
```

Como vemos tenemos unos datos recogidos en forma de lista. Para tratarla vamos a pasarla a data.frame con la que podremos trabajar posteriormente.

```
tweets_df <- twListToDF(tweets)
l<-length(tweets)
```

Mostremos por ejemplo los diez primeros tweets que hemos obtenidos en la búsqueda:

```
for (i in 1:10) {
 cat(paste0("[", i, "] "))
 writeLines(strwrap(tweets_df$text[i], 40))
 }
```

```
## [1] Me ha gustado un vídeo de @YouTube
## (https://t.co/fkNWkA6Ycb - V. Completa.
## "Las matemáticas nos hacen más libres y
## menos
## [2] RT @vergueletor: Matemáticas <U+0001F480>
## https://t.co/lIlv5xX4pa
## [3] Jajajajaja ni cabida le paso al gordito
## de matematicas de chetoo ñoño todo bien
## [4] Tengo un 9 en Matemáticas y un 9,5 en
## Historia del Arte.
##
## Flipanding sigo https://t.co/7CoVAi1U6P
## [5] RT @Melina_Ntvgera: qué lindo cuando
## estás haciendo los ejercicios de
## matemáticas y te salen, esa felicidad
## que se siente no tiene explicac...
## [6] RT @MasterChef_es: Yo en los exámenes
## de matemáticas #MasterChef
## https://t.co/t15F4KDXd5
## [7] Estoy comiendo en la facultad de
## matemáticas y espero que no me este
## viendo la mutual porque entre que estoy
## comien... https://t.co/tuVRXl5MhQ
## [8] RT @FundacionBBVA: Fallados los Premios
## de Investigación Matemática Vicent
## Caselles de la @RealSocMatEsp y la
## Fundación BBVA #matematicas h...
## [9] RT @teainfo: La TEA publicó hoy los
## resultados de la primavera de 2018
## STAAR para matemáticas y lectura en los
## grados 3, 4, 5, 6, 7 y 8. ht...
## [10] Libro de Matemáticas 2018 Resuelto -
## Solucionario libro de matemática 8, 9 y
## 10 EGB Foros Ecuador 2018
## https://t.co/O9xHY4Q2lI
```

Busquemos ahora los tweets de un usuario especifico como por ejemplo la cuenta oficial de Zara en españa (zaraes), mostraremos unicamente los 10 primeros como en el ejemplo anterior:

```
zara_es<-userTimeline("zaraes" ,n = 50)
l_zara<- length(zara_es)
cat("El tipo de datos obtenido es: " ,typeof(zara_es))
```

```
## El tipo de datos obtenido es:  list
```

```
tweets_zara_df <- twListToDF(zara_es)

for (i in 1:10) {
 cat(paste0("[", i, "] "))
 writeLines(strwrap(tweets_zara_df$text[i], 40))
 }
```

```
## [1] @suretofall Hola María, hemos
## contestado a tu mensaje desde nuestra
## cuenta de atención al cliente en
## Twitter, ZARA Care. Muchas gracias.
## [2] Lemmie van den Berg entrena duro para
## el próximo partido. ¡Que empiece el
## juego! https://t.co/u8jCQ73PE9
## [3] Damaris Goddrie es nuestra máxima
## goleadora. ¡Que empiece el partido!
## https://t.co/Vf8EqAWXzN
## [4] Roberto Sipos es nuestro último fichaje
## en Zara. ¡Que empiece el juego!
## https://t.co/ek79JJmEgo
## [5] @Omar271990 Para futuras consultas, no
## dudes en escribirnos a nuestra cuenta
## de atención al cliente, @ZARA_Care.
## Muchas gracias.(2/2)
## [6] @Omar271990 Hola. Gracias por contactar
## con ZARA. De momento no contamos con la
## información que solicitas. Sin emba...
## https://t.co/HuhmMxPcA3
## [7] @Victoria_Vega_V Hola, sentimos las
## molestias ya que todos nuestros
## productos pasan por rigurosos controles
## de cali... https://t.co/MptX3kQwZL
## [8] @virmu75 Hola Virginia, por favor,
## envía un mensaje directo a nuestra
## cuenta de atención al cliente,
## @ZARA_Care, co... https://t.co/N51tyZv5WI
## [9] @saradolz Hola Sara, gracias por
## escribirnos. Por favor, envíanos tu
## número de pedido, nombre completo y
## dirección... https://t.co/xJFRENo2YM
## [10] Descubre la nueva colección de lino
## para niños en colores neutros, rayas o
## cuadros. https://t.co/GUe9uBT53F
## https://t.co/NyFoesF2ph
```

De los datos que hemos obtenido podemos tratar mucha información, por ejemplo:

Información sobre fecha y hora de creación

```
# Fecha y hora de creación del primer tweet extraido.
zara_es[[1]]$created
```

```
## [1] "2018-06-15 08:41:26 UTC"
```

Número de favoritos y retweets

```
# Fecha y hora de creación del primer tweet extraido.
cat("El número de favoritos del primer tweet es: " ,
    zara_es[[1]]$favoriteCount, " " , "y el número de RT :
    ",zara_es[[1]]$retweetCount)
```

```
## El número de favoritos del primer tweet es:  1   y el número de RT :
##      0
```

Supongamos que queremos ver cuales de todos los tweets obtenidos es el más favoriteado o el que tiene más retweets. Podríamos calcularlo de manera simple con una función de la forma:

```
# Tweet con mayor número de favoritos.

x_fav<-rep(0,l_zara)

for (i in 1:l_zara){
  x_fav[i]<- zara_es[[i]]$favoriteCount
}

maximo_fav <- which.max(x_fav)

cat("El tweet más favoriteado es: ","\n")
```

```
## El tweet más favoriteado es:
```

```
cat( writeLines(strwrap(tweets_zara_df$text[maximo_fav], 40)),
     " con", zara_es[[maximo_fav]]$favoriteCount, "favoritos.")
```

```
## Trae la ropa que no utilizas y dale una
## nueva vida. Toda la ropa recogida se
## entrega directamente a Cáritas. Con su...
## https://t.co/DK6DFxsDvQ
##   con 7 favoritos.
```

```
# Tweet con mayor número de favoritos.

x_RT<-rep(0,l_zara)

for (i in 1:l_zara){
  x_RT[i]<- zara_es[[i]]$retweetCount
}

maximo_RT <- which.max(x_RT)

cat("El tweet más retuiteado es: ","\n")
```

```
## El tweet más retuiteado es:
cat( writeLines(strwrap(tweets_zara_df$text[maximo_RT], 40)),
     " con", zara_es[[maximo_RT]]$retweetCount, "RTs")
```

```
## Trae la ropa que no utilizas y dale una
## nueva vida. Toda la ropa recogida se
## entrega directamente a Cáritas. Con su...
## https://t.co/DK6DFxsDvQ
##   con 8 RTs
```

Una vez visto que podemos hacer con los tweets vamos a sacar los datos de un usuario concreto, pero antes detallemos como modelizamos los datos que vamos a obtener, ya que existen muchos tipos de modelos.

2.1. Modelización de datos

Profundicemos ahora en la modelización de los datos que hemos tomado. Es importante el tipo de modelo que tomemos para la modelización ya que a partir de esta vamos a poder aprovechar mejor dichos datos, es por eso que a veces nos convenga más usar un tipo de modelización a otra.

2.1.1. Modelo Booleano

Los modelos de recuperación de información establecen el mecanismo empleado para realizar el procesamiento de las consultas de los usuarios, a partir de una necesidad de información que requerimos y de una colección de documentos, en nuestro caso palabras que conforman un tweet, podemos predecir que palabras son considerados relevantes o no y en que grado.

El modelo Booleano es el más utilizado no solo por ser uno de los primeros modelos que surgieron para la recuperación de información y su clasificación, sino por su formalismo y la facilidad de implementación a conjuntos de datos aleatorios. Este modelo se basa en la teoría de conjuntos y el álgebra de Boole, mediante la aplicación de operaciones lógicas AND (Intersección), OR (Unión), NOT (Resta) y XOR (Complemento).

El procesamiento automatizado de un documento textual comienza con la extracción de los términos de indización, es decir, los términos que van a ser utilizados para describir el contenido del documento, en nuestro caso estos términos corresponderían a los que conforman los tweets extraidos.

La posibilidad más simple consiste en considerar todas las palabras aisladas que aparecen en el texto como los términos de indización. Habitualmente se eliminan algunas (las denominadas palabras vacías, entre las que suelen figurar los números, las preposiciones, conjunciones, verbos ser, haber y estar), aunque la consideración o no de estos procesos añadidos (como la inclusión de una lista de palabras vacías) no influyen en absoluto sobre los principios teóricos del modelo.

Una vez extraídas las palabras del texto, se ordenan por orden alfabético y se guardan en un denominado fichero inverso, junto con la referencia del documento de donde proceden (normalmente un número de documento asignado previamente por el sistema). Si se repite este proceso con todos los documentos de la colección, obtendremos finalmente un fichero inverso que almacena los siguientes datos:

• En primer lugar, los términos de indización (las palabras) que aparecen en toda la colección (ya sean los propios textos, los resúmenes de los textos del fondo y/o los títulos).

• En segundo lugar, cada uno de dichos términos (palabras) incorpora una lista con los números de los documentos en los que aparece.

Conviene destacar en este proceso que no se ha guardado noticia alguna sobre la frecuencia de aparición de cada término en cada documento. De ahí que el modelo booleano clásico sea denominado modelo binario, pues de la consulta del fichero inverso únicamente puedo saber si un determinado término de indización está presente (en cuyo caso se simbolizará por el número 1) o está ausente (en cuyo caso se simbolizará por el número 0) en cada uno de los documentos de la colección.

De manera que el fichero inverso (en concreto el fichero diccionario) puede representarse por una tabla cuyos datos básicos son los siguientes:

	D1	D2	...	Dn
T1	1	0	...	1
T2	0	1	...	1
T3	0	0	...	1
...	...	...	...	...
Tt	1	1	...	0

Donde T1, T2, ..., Tn son los términos de indización empleados en la colección; D1, D2, ..., Dn son los documentos que componen la colección; y donde el "1" significa que el término correspondiente aparece en ese documento concreto, mientras que el "0" significa que el término no aparece en dicho documento.

2.1.2. Modelo de Grafos

Un grafo es un conjunto de nodos unidos por enlaces llamados aristas, esto nos permite representar relaciones binarias entre elementos de un conjunto. Por la formación, un grafo G=(V,E) lo representamos como una pareja ordenada en la que V es un conjunto no vacío de vértices y E es un conjunto de aristas. Donde V consta de pares no ordenados de vértices, tales como $\{x,y\} \in$E entonces decimos que x e y son adyacentes, en el grafo se represetna mediante una linea no orientada que une los dos vértices.

Podemos categorizar los grafos en dos grupos, dirigido y no dirigido:

- Un grafo dirigido es un grafo G=(V,E) tal que V es no vacio y el conjunto de pares de vértices distintos es un conjunto de pares ordenado de elementos de V tal que una arista (a,b) comienza en su nodo inicial a y termina en el nodo final b. Por definición este tipo de grafos no poseen bucles.

- Un grafo no dirigido es un grafo tal que las aristas que componen estos grafos no tienen orientación. A la hora de modelizar situaciones las relaciones entre nodos siempre serían bilaterales.

Dentro de la teoria de grafos debemos detacar el concepto de matriz de adyacencia. La matriz de adyacencia es una matriz cuadrada v x v con v = |V| que se utiliza como una forma de representar relaciones binarias. En el caso de los grafos no dirigidos la componente ij es igual a 1 si los nodos i y j están relacionados y 0 en caso contrario por lo que la matriz es simétrica. Para el caso de los grafos dirigidos la componente ij sería igual a 1 si existe una arista que vaya del nodo i al j y 0 en caso contrario.

El número de caminos $C_{i,j}(k)$, atravesando k aristas desde el nodo i al nodo j, viene dado por un elemento de la potencia k-ésima de la matriz de adyacencia: $C_{i,j}(k) = [A^k]_{i,j}$

2.1.2.1. Grafos

Una vez visto el concepto teórico de un grafo vamos a aplicarlo a la información obtenida. Tomemos primero un usuario cualquiera, el cual será nuestro nodo central sobre el que unirán las aristas con los demás nodos que corresponderán a los usuarios seguidores y/o seguidos de nuestro usuario principal.

Cargamos la libreria "igraph" que nos permite realizar y trabajar con grafos. Además cargamos el usuario principal para obtener sus seguidores y seguidos.

```
library(igraph)
# usuario2 <-getUser("*********")
load("usuario_grafo.RData")
friends.object <-lookupUsers(usuario2$getFriendIDs())
followers.object1 <-lookupUsers(usuario2$getFollowerIDs())

# Restringimos a 50 el número de seguidores
followers.object<-followers.object1[1:50]
```

Creamos un vector con el id de los usuarios junto con su respectivo nombre para que aparezcan en el grafo los nombres.

```
# Restringimos a 15 los usuarios tanto seguidores como
# seguidos para obtener un ejemplo de grafo que podamos
# ver claramente. Normalmente solemos tomar todos los
# usuarios quedando un grafo mucho mas complejo.

# friends <- sapply(friends.object[1:15],name)
# followers <- sapply(followers.object[1:15],name)

load("friends_grafo.RData")
load("followers_grafo.RData")
```

Creamos un dataframe con la relación de los seguidos y seguidores al usuario.

```
relations <- merge(data.frame(User='blog_imus', Follower=friends),
                   data.frame(User=followers, Follower='blog_imus'),
                   all=T)
```

Por último creamos un grafo con las relaciones. Y dibujamos el grafo.

```
g <- graph.data.frame(relations, directed = T)
V(g)$label <- V(g)$name
plot(g)
```

Capítulo 3

Tratamiento de datos

En este capitulo vamos a realizar un tratamiento de los datos, conocido más comúnmente como Text Mining, con el fin de analizar dichos datos. Una vez extraido los datos, como hemos hecho en el capitulo 1, pasamos a transformarlos creando una matriz término-documento. En esta matriz podremos encontrar la frecuencia y las asociaciones entre las palabras que encontramos en nuestros datos, los tweets.

3.1. Matriz término-documento

Una matriz de término-documento es una forma de representar las palabras en el texto como una tabla (o matriz) de números. Las filas de la matriz representan las respuestas de texto que se analizarán, es decir las filas corresponden a un término, y las columnas de la matriz representan las palabras del texto que se utilizarán en el análisis, es decir los distintos documentos.El conjunto de documentos que obtenemos de los datos es lo que denominamos *corpus*, podríamos referirnos a él como el diccionario a partir del cual nos basamos para realizar los distintos análisis, aunque también podemos usar el propio diccionario de una lengua específica. La versión más básica es una matriz binaria, en la que cada celda solo toma valores 0 ó 1, correspondería a un modelo Booleano que hemos visto en el capítulo anterior. Un ejemplo de una matriz término-documento binaria sería el visto anteriormente:

	D1	D2	...	Dn
T1	1	0	...	1
T2	0	1	...	1
T3	0	0	...	1
...	...	...	...	...
Tt	1	1	...	0

Cada valor que tomamos de la matriz corresponde al número de ocurrencias de un determinado término en un documento especifico, por ejemplo en nuestra matriz anterior, el término T2 aparece en el documento D2 al igual que en el Dn, sin embargo no aparece en el documento D1.

Si lo llevamos a los datos que estamos tratando, los tweets de un determinado usuario

nos encontramos que cada documento corresponderia a un tweet y cada término a una palabra. Por lo que nuestra matriz nos proporcionaría información sobre que palabra esta en un determinado tweet o no.

Pasemos a tratar los datos obtenidos con el fin de crear un corpus y una matriz término-documento.

```
# zara_es<-userTimeline("zaraes" ,n = 500)
load("zara_frec.RData")
cat("Obtenemos" ,length(zara_es), "tweets del usuario @zaraes")
```

```
## Obtenemos 500 tweets del usuario @zaraes
```

```
tweets_zara_df <- twListToDF(zara_es)
```

Una vez obtenido 500 tweets del usuario zaraes , el twitter oficial en españa de la empresa Zara, y convertirlo en un data.frame, cargamos la librería "tm" que nos facilitará el tratamiento y limpieza de datos para luego crear el corpus.

```
library(tm)
texto<- tweets_zara_df$text
  texto = gsub("ã¡", "a", texto)
  texto = gsub("ã©", "e", texto)
  texto = gsub("ã³", "o", texto)
  texto = gsub("ãº", "u", texto)
  texto = gsub("ã±", "ñ", texto)
  texto = gsub("ã¨", "e", texto)
  texto = gsub("ã²", "o", texto)
  texto = gsub("ã", "i", texto)
  texto = gsub("Ã", "io", texto)
myCorpus <- Corpus(VectorSource(texto))
```

Ahora que tenemos creado el corpus debemos realizar un tratamiento de limpieza de texto, que consiste en cambiar letras a minúsculas, para que todas las palabras sean iguales y no se diferencien por tener o no mayúsculas, eliminar todos los símbolos de puntuación, números y stop-words, palabras que no aportan significado y debemos de eliminar, por ejemplo "en", "a", "el". Ademas también eliminamos los hipervinculos que nos lleven a otras páginas.

```
# Convertimos a minúsculas
myCorpus <- tm_map(myCorpus, tolower)
# Quitamos los símbolos de puntuaciones
myCorpus <- tm_map(myCorpus, removePunctuation)
# Quitamos los números
myCorpus <- tm_map(myCorpus, removeNumbers)
# Quitamos URLs
removeURL <- function(x) gsub("http[[:alnum:]]*", "", x)
myCorpus <- tm_map(myCorpus, removeURL)
# Añadimos dos stop-words: "available" and "via"
myStopwords <- c(stopwords('spanish'), "available", "via")
# Quitamos zaraes de las stop-words ya que forman parte del
# nombre del usuario
```

```
myStopwords <- setdiff(myStopwords, c("zaraes"))
# Quitamos las stop-words del corpus
myCorpus <- tm_map(myCorpus, removeWords, myStopwords)
# Borramos los espacios en blanco extras
myCorpus <- tm_map(myCorpus, stripWhitespace)
```

La función tm_map nos permite aplicar transformaciones de manera rápida para la minería de texto. También podemos realizar una limpieza de los tweets a partir de la función gsub() con la que trabajamos con los textos de los tweets previamente extraídos,en este caso la hemos usado para poder limpiar algunos caracteres latinos que nos podrian producir problemas.

Después de tener el corpus hecho, pasamos a construir nuestra matriz término-documento. Para ello usaremos la función TermDocumentMatrix que crea la matriz de manera directa a partir del corpus.

```
myTdm <- TermDocumentMatrix(myCorpus)
myTdm
```

```
## <<TermDocumentMatrix (terms: 1597, documents: 500)>>
## Non-/sparse entries: 4703/793797
## Sparsity           : 99%
## Maximal term length: 16
## Weighting          : term frequency (tf)
```

A partir de la matriz podemos obtener por ejemplo las palabras más populares y la asociación entre ellas. Veamos las palabras con un mínimo de 10 apariciones en documentos (tweets).

```
findFreqTerms(myTdm, lowfreq=15)
```

```
##  [1] "colecciÃ"   "prendas"    "anos"       "comentas"   "envÃ"
##  [6] "favor"      "hola"       "mensaje"    "sentimos"   "zara"
## [11] "disponible" "descubre"   "colores"    "verano"     "mujer"
## [16] "compra"     "puedes"     "gracias"    "cuenta"     "mero"
## [21] "pedido"     "zaracare"   "artÃ"       "novedades"  "tonos"
## [26] "atenciÃ"    "cliente"    "nueva"      "directo"    "informaciÃ"
## [31] "tienda"     "denim"      "primavera"  "travÃ"       "consulta"
## [36] "cualquier"  "care"       "arte"       "campaÃ"     "fotografÃ"
```

Vemos que los acentos no los trata adecuadamente.

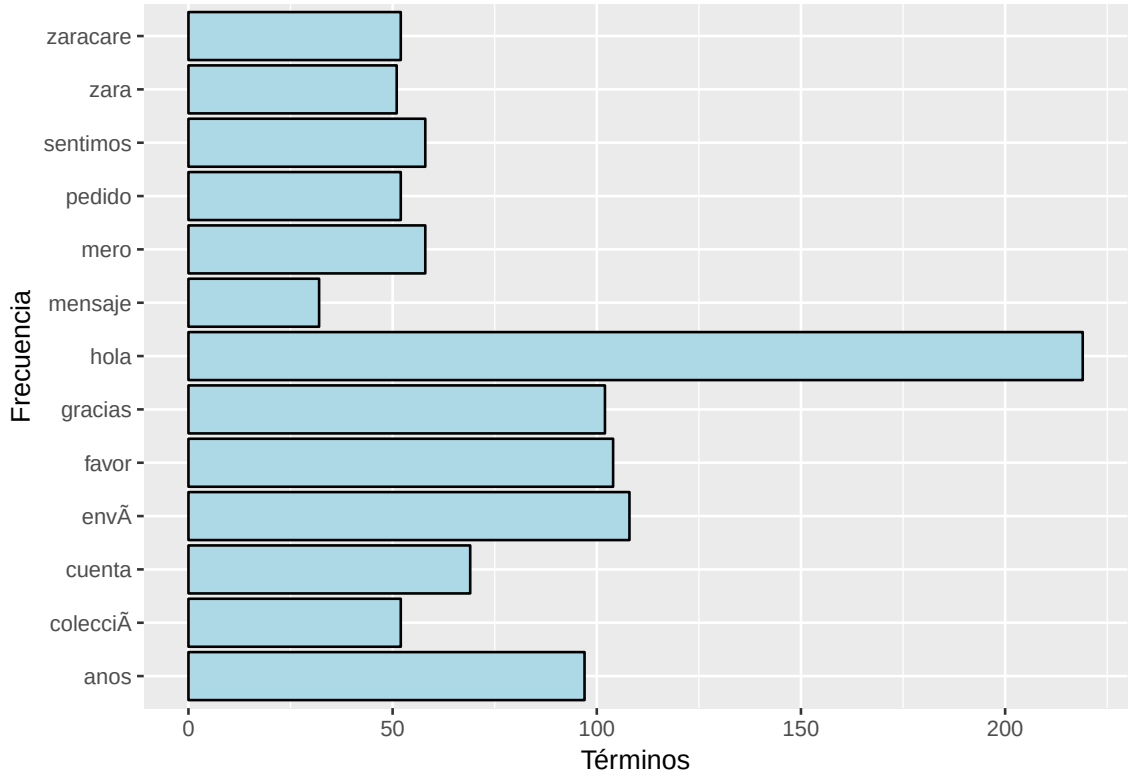
Para ver de manera más gráfica la frecuencia de cada palabra podemos realizar un gráfico de barras. Para realizar el grafico solo debemos de sumar las filas de nuestra matriz ya que asi obtendremos la frecuencia total en la que aparece un término (podemos hacerlo con la función rowSums) y seleccionando de estos aquellos que superen un mínimo.

La función que vamos a usar para realizar la gráfica es barplot que pertenece al paquete ggplot2.

```
library(ggplot2)
termFrequency <- rowSums(as.matrix(myTdm))
termFrequency <- subset(termFrequency, termFrequency>=30)
```

```
termFrequency_df<-as.data.frame(termFrequency)
termFrequency_df2<-data.frame(terminos = names(termFrequency),
                              frecuencia = termFrequency_df$termFrequency)
```

```
ggplot(termFrequency_df2,aes(terminos,frecuencia))+
  geom_bar(stat = "identity", color = "black",fill = "light blue")+
  coord_flip()+ xlab("Frecuencia")+ylab("Términos")
```



También podemos ver la asociación entre las palabras. Busquemos por ejemplo las palabras que tengan una correlación mínima de 0.25 con la palabra gracias, para ver asi que palabras van mas unida a la gratitud de los tweets que tenemos.

```
findAssocs(myTdm, 'gracias', 0.25)
```

```
## $gracias
##     muchas  fotografÃ respondido     interÃ      baron   direcciÃ
##       0.32       0.31       0.30       0.29       0.28       0.28
##     fabien     meisel     steven     campaÃ
##       0.26       0.26       0.26       0.25
```

Capítulo 4

Aplicaciones

Una vez creada la matriz término-documento y realizado el corpus vamos a ver distintas aplicaciones que podemos hacer con los datos que tenemos, entre ello veremos las nubes de palabras, clusteres, etc.

4.1. Nubes de palabras

Una nube de palabras, en inglés word cloud, es una representación visual de las palabras que conforman un texto, en donde el tamaño es mayor para las palabras que aparecen con más frecuencia. A partir de la matriz término-documento que hemos obtenido anteriormente podemos obtener la frecuencia de todas las palabras de nuestros tweets y realizar una nube de palabras a partir de esto, mostrando así la importancia que posee las palabras mas relevantes.

Gracias al paquete de r "wordcloud" podemos realizar una gráfica de nube de palabras facilmente. Para ello debemos de convertir primero nuestra matriz término-documento en una matriz normal con la que podamos trabajar y calcular las frecuencias de las palabras. Una vez que tenemos los términos y las frecuencias usando la función wordcloud() e imponiendo algunas condiciones gráficas.

Veamos una nube de palabras con los datos que tenemos sobre los tweets de Zara.

```
library(wordcloud)

m <- as.matrix(myTdm)

# Calculamos la frecuencia y ordenamos las palabras
 wordFreq <- sort(rowSums(m), decreasing=TRUE)

# Fijamos una semilla de generación apra hacerlo reproducible
set.seed(123)
wordcloud(words=names(wordFreq), freq=wordFreq, min.freq=4,
          random.order=F, colors = brewer.pal(8,"Dark2"))
```

Como vemos en la nube la mayor palabra es hola seguida de gracias,favor y envío esta claro que una cuenta twitter oficial de una empresa destaquen estas palabras.

Además de poder hacer una nube de palabras para mostrar las palabras mas relevantes sobre los tweets de un usuario en concreto, podemos realizar una nube de palabra comparativas en la que podemos observar en una misma nube las palabras mas destacadas de cada usuario. Tomemos por ejemplo algunas marcas de la empresa Inditex, por ejemplo, Zara, Stradivarius y Oysho todos ellos tomaremos los respectivos usuarios ("zaraes", "stradivarius", "Oysho").

Para ello debemos de tomar los datos de sus respectivos usuarios y realizar una nueva matriz y corpus con sus datos.

```
# zara tweets

zara_tweets = zara_es

# stradivarius tweets

# stra_tweets = userTimeline("stradivarius", n=500)
load("stra_tweets.RData")

# oysho tweest

# oysh_tweets = userTimeline("Oysho", n=500)
load("oysh_tweets.RData")

# Obtenemos el texto
zara_txt = sapply(zara_tweets, function(x) x$getText())
```

```
stra_txt = sapply(stra_tweets, function(x) x$getText())
oysh_txt = sapply(oysh_tweets, function(x) x$getText())


# Aplicamos una función de limpieza gsub()
clean.text <- function(some_txt)
{
  some_txt = gsub("(RT|via)((?:\\b\\W*@\\w+)+)", "", some_txt)
  some_txt = gsub("@\\w+", "", some_txt)
  some_txt = gsub("[[:punct:]]", "", some_txt)
  some_txt = gsub("[[:digit:]]", "", some_txt)
  some_txt = gsub("http\\w+", "", some_txt)
  some_txt = gsub("[ \t]{2,}", "", some_txt)
  some_txt = gsub("^\\s+|\\s+$", "", some_txt)

  # Definimos la función "tolower error handling"
  try.tolower = function(x)
  {
    y = NA
    try_error = tryCatch(tolower(x), error=function(e) e)
    if (!inherits(try_error, "error"))
      y = tolower(x)
    return(y)
  }

  some_txt = sapply(some_txt, try.tolower)
  some_txt = some_txt[some_txt != ""]
  names(some_txt) = NULL
  return(some_txt)
}

# Limpiamos los textos
zara_clean = clean.text(zara_txt)
stra_clean = clean.text(stra_txt)
oysh_clean = clean.text(oysh_txt)




zara = paste(zara_clean, collapse=" ")
stra = paste(stra_clean, collapse=" ")
oysh = paste(oysh_clean, collapse=" ")

# put everything in a single vector
todos = c(zara, stra, oysh)

# remove stop-words
```

```
todos = removeWords(todos,
                    c(stopwords("spanish"), "zaraes", "stradivarius",
                      "Oysho"))

todos=removeWords(todos,stopwords("english"))


# Creamos el  corpus
corpus = Corpus(VectorSource(todos))

# Creamos la matriz término-documento
tdm = TermDocumentMatrix(corpus)

# Convertimos a matriz normal
tdm = as.matrix(tdm)

# add column names
colnames(tdm) = c("zara", "stradivarius", "oysho")


# comparison cloud
comparison.cloud(tdm, random.order=FALSE,
                 colors = c("#00B2FF", "red", "#FF0099"),
                 title.size=1.5, max.words=500)
```

4.2. Clustering

Dentro de este apartado vamos profundizar sobre el concepto de clustering asi como desarrollar la teoría sobre sus principales modelos y algoritmos. De manera práctica, a medida que vayamos explicando distintos tipos de métodos iremos aplicándolos a los datos que poseemos con el fin de realizar particiones de las palabras que encontramos en los tweets.

El clustering, análisis de grupos o agrupamiento es la tarea de agrupar un conjunto de objetos de tal manera que los miembros del mismo grupo (llamado clúster) sean más similares, en algún sentido u otro. Es la tarea principal de la minería de datos exploratoria y es una técnica común en el análisis de datos estadísticos. Además es utilizada en múltiples campos como el aprendizaje automático, el reconocimiento de patrones, el análisis de imágenes, la búsqueda y recuperación de información.

El análisis de grupos no es en sí un algoritmo específico, sino una tarea pendiente de solución, ya que aún se siguen investigando diferentes métodos para mejorar la eficencia de los clústeres. Se puede hacer el clustering utilizando varios algoritmos que difieren significativamente en su idea de qué constituye un grupo y cómo encontrarlos eficientemente. Las ideas clásicas de clúster incluyen distancias pequeñas entre los miembros del mismo, áreas densas del espacio de datos, intervalos o distribuciones estadísticas particulares, por lo que, podríamos decir que el clustering puede ser formulado como un problema multi-objetivo de optimización. El algoritmo apropiado y los valores de los parámetros (incluyendo valores como la función de distancia para utilizar, un umbral de densidad o el número de grupos esperado) depende del conjunto de datos que se analiza y el uso que se le dará a los resultados.

Los algoritmos de clustering pueden ser categorizados atendiendo a su modelo de grupo, como se dijo antes. Sólo se listará en adelante los algoritmos más prominentes, ya que existen más de 100 publicados, de los cuales no todos proporcionan modelos para sus grupos y por esto no son fáciles de categorizar. No existe un algoritmo de agrupamiento "correcto" sino que para cada conjunto de datos podemos tener un algoritmo que se ajuste mejor o peor al resultado que buscamos. De esta manera, el algoritmo más apropiado para un problema particular a menudo necesita ser escogido experimentalmente, a no ser que haya una razón matemática para preferir un modelo de grupo sobre otro.

El proceso completo puede estructurarse de acuerdo con el siguiente esquema:

- Partimos de un conjunto de N individuos de los que se dispone de una información cifrada por un conjunto de n variables (una matriz de datos de N individuos ´ n variables) .
- Establecemos un criterio de similaridad para poder determinar: Una matriz de similaridades que nos permita relacionar la semejanza de los individuos entre sí (matriz de N individuos x N individuos).
- Escogemos un algoritmo de clasificación para determinar la estructura de agrupación de los individuos.
- Especificamos esa estructura mediante diagramas arbóreos o dendogramas u otros gráficos.

4.2.1. Distancias

Para medir lo similares (o disimilares) que son los individuos existe una enorme cantidad de índices de similaridad y de disimilaridad o divergencia. Todos ellos tienen propiedades y utilidades distintas y habrá que ser consciente de ellas para su correcta aplicación al caso que nos ocupe.

La mayor parte de estos índices serán o bien, indicadores basados en la distancia (considerando a los individuos como vectores en el espacio de las variables) (en este sentido un elevado valor de la distancia entre dos individuos nos indicará un alto grado de disimilaridad entre ellos); o bien, indicadores basados en coeficientes de correlación ; o bien basados en tablas de datos de posesión o no de una serie de atributos.

Veremos los criterios basados en distancias como indicadores de disimilaridad.

Por definición, distancia o disimilaridad entre dos individuos i y j a una medida, indicada por d(i,j) , que mide el grado de semejanza, o a mejor decir de desemejanza, entre ambos objetos o individuos, en relación a un cierto número de características cuantitativa y / o cualitativas. El valor de d(i,j) es siempre un valor no negativo, y cuanto mayor sea este valor mayor será la diferencia entre los individuos i y j.

Veamos las distancias Euclídea, del Máximo, Manhattan, Mahalanobis Minkowski y para valores binarios. En R se puede utilizar la función "dist" que nos devuelve una matriz de distancia, dentro de esta función podemos introducirle la distancia que vaya a utilizar.

4.2.1.1. Distancia Euclídea

La distancia euclídea es la disimilaridad más conocida y más sencilla de comprender, pues su definición coincide con el concepto más común de distancia.

$d_{euc}(X,Y) = \sqrt{\sum (X_i - Y_i)^2}$

La distancia euclídea, a pesar de su sencillez de cálculo y de que verifica algunas propiedades interesantes tiene dos graves inconvenientes:

- El primero de ellos es que la euclídea es una distancia sensible a las unidades de medida de las variables: las diferencias entre los valores de variables medidas con valores altos contribuirán en mucha mayor medida que las diferencias entre los valores de las variables con valores bajos. Como consecuencia de ello, los cambios de escala determinarán, también, cambios en la distancia entre los individuos. Una posible vía de solución de este problema es la tipificación previa de las variables, o la utilización de la distancia euclídea normalizada .
- El segundo inconveniente no se deriva directamente de la utilización de este tipo de distancia, sino de la naturaleza de las variables. Si las variables utilizadas están correlacionadas, estas variables nos darán una información, en gran medida redundante. Parte de las diferencias entre los valores individuales de algunas variables podrían explicarse por las diferencias en otras variables. Como consecuencia de ello la distancia euclídea inflará la disimilaridad o divergencia entre los individuos.

La solución a este problema pasa por analizar las componentes principales (que están incorrelacionadas) en vez de las variables originales. Otra posible solución es ponderar

la contribución de cada par de variables con pesos inversamente proporcionales a las correlaciones, lo que nos lleva, como veremos a la utilización de la distancia de Mahalanobis.

La distancia euclídea será, en consecuencia, recomendable cuando las variables sean homogéneas y estén medidas en unidades similares y/o cuando se desconozca la matriz de varianzas.

4.2.1.2. Distancia del Máximo

$d_{max}(X,Y) = máx|X_i - Y_i|$

4.2.1.3. Distancia Manhattan

La distancia de Manhattan, d_{manh} entre dos vectores X e Y en un espacio vectorial real n-dimensional con un sistema de Coordenadas cartesianas fijo es la suma de las longitudes de las proyecciones del segmento de línea entre los puntos sobre el sistema de ejes coordenados. Más formalmente:

$d_{manh}(X,Y) = \sum_i |X_i - Y_i|$

4.2.1.4. Distancia de Mahalanobis

La distancia de Mahalanobis es una medida de distancia cuya utilidad radica en que es una forma de determinar la similitud entre dos variables aleatorias multidimensionales. Se diferencia de la distancia euclídea en que tiene en cuenta la correlación entre las variables aleatorias.

$d_{mah}(X,Y) = \sqrt{(X-Y)'S^{-1}(X-Y)}$

Donde S es la matriz de covarianzas entre X e Y, en el caso de que esta matriz sea diagonal, es decir, X e Y son vectores aleatorios independientes, la distancia pasa a ser distancia euclídea estandarizada y se podria expresar como:

$d_{euc-est}(X,Y) = \sqrt{\frac{(x_i-y_i)^2}{s_i^2}}$

con s_i los elementos diagonales de la matriz de covarianzas S.

4.2.1.5. Distancia de Minkowski

La distancia de Minkowski es una métrica en un espacio vectorial normado que puede considerarse como una generalización tanto de la distancia euclídea como de la distancia de Manhattan.

$d_{mink,p}(X,Y) = (\sum |X-Y|^p)^{\frac{1}{p}}$

4.2.1.6. Ditancia valores binarios

Proporcion de variables donde un solo caso es 1 respecto del total de casos donde al menos una variable es 1.

$d_{bin}(X,Y) = \frac{\sum_i (X_i - Y_i)}{\sum_i X_i + \sum_i Y_i + \sum_i X_i Y_i}$

4.2.2. Métodos particionales

Una vez vistas las diferentes distancias que podemos usar para saber la similitud entre los distintos elementos pasamos a ver los métodos para tratar los datos. Podemos encontrar varios tipos de métodos, entre ellos, los particionales y los jerárquicos.

La principal diferencia entre ellos es que los métodos particionales solo dividen el conjunto de datos en k clusteres en base a un criterio de optimalidad de cierta función.

En los métodos de partición se fija a priori el número deseado de conglomerados, k. Destaca en primer lugar el método de los centros móviles, también llamado algoritmo de las k-medias.

4.2.2.1. Algoritmo de k-medias

El algoritmo de las K-medias (presentado por MacQueen en 1967) es uno de los algoritmos de aprendizaje no supervisado más simples para resolver el problema de la clusterización. El procedimiento aproxima por etapas sucesivas un cierto número (prefijado, k) de clusters haciendo uso de los centroides de los puntos que deben representar.

El algoritmo se compone de los siguientes pasos::

1. Sitúa K puntos en el espacio en el que "viven" los objetos que se quieren clasificar. Estos puntos representan los centroides iniciales de los grupos.

2. Asigna cada objeto al grupo que tiene el centroide más cercano.

3. Tras haber asignado todos los objetos, recalcula las posiciones de los K centroides.

4. Repite los pasos 2 y 3 hasta que los centroides se mantengan estables. Esto produce una clasificación de los objetos en grupos que permite dar una métrica entre ellos.

```
set.seed(123)
myTdm2 <- removeSparseTerms(myTdm, sparse=0.95)
m2 <- as.matrix(myTdm2)
m3 <- t(m2)
k<-7
kmeansResult <- kmeans(m3, k)
round(kmeansResult$centers, digits=3)
```

```
##   colecciÃ prendas  anos  envÃ favor  hola mensaje sentimos  zara descubre
## 1    0.444   0.000 0.000 0.000 0.000 0.000   0.000    0.000 0.111        1
## 2    0.000   0.000 1.000 1.000 1.000 0.898   0.017    0.339 0.441        0
## 3    0.167   0.105 0.004 0.004 0.000 0.000   0.000    0.031 0.031        0
## 4    0.000   0.000 0.000 0.000 0.000 0.600   0.100    0.100 1.000        0
## 5    0.000   0.022 0.822 0.978 0.956 0.933   0.533    0.111 0.000        0
## 6    0.025   0.025 0.000 0.025 0.025 1.000   0.000    0.300 0.062        0
## 7    0.000   0.000 0.000 0.039 0.000 0.745   0.118    0.020 0.000        0
##   verano puedes gracias cuenta  mero pedido zaracare nueva directo
## 1  0.000  0.000   0.000  0.000 0.000  0.000    0.000 0.185   0.000
## 2  0.000  0.000   0.254  0.593 0.525  0.458    0.000 0.000   0.000
## 3  0.127  0.031   0.070  0.013 0.000  0.000    0.013 0.105   0.000
## 4  0.000  0.300   0.600  0.800 0.100  0.100    0.000 0.000   0.100
```

```
## 5  0.000  0.000   0.289  0.422 0.467  0.444    0.911 0.000   0.489
## 6  0.000  0.175   0.000  0.012 0.038  0.025    0.050 0.000   0.000
## 7  0.000  0.059   1.020  0.059 0.039  0.039    0.078 0.000   0.078
##   primavera travÃ  care
## 1     0.000 0.222 0.000
## 2     0.000 0.169 0.322
## 3     0.123 0.022 0.000
## 4     0.000 0.200 1.000
## 5     0.000 0.089 0.000
## 6     0.000 0.012 0.000
## 7     0.000 0.020 0.000
```

Con esto podemos ver las tres palabras pertenecientes a los diferentes clústeres.

```
for (i in 1:k) {
cat(paste("cluster ", i, ": ", sep=""))
s <- sort(kmeansResult$centers[i,], decreasing=T)
cat(names(s)[1:3], "\n")
# Imprimir tweets de cada cluster
# print(rdmTweets[which(kmeansResult$cluster==i)])
}
```

```
## cluster 1: descubre colecciÃ travÃ
## cluster 2: anos envÃ favor
## cluster 3: colecciÃ verano primavera
## cluster 4: zara care cuenta
## cluster 5: envÃ favor hola
## cluster 6: hola sentimos puedes
## cluster 7: gracias hola mensaje
```

4.2.2.2. Algoritmo de k-medioides

Se buscan k puntos representativos de k conglomerados, llamados medioides, de modo que la suma de las distancias de los puntos a su medioide más cercano sea mínima. La función objetivo en este caso sería:

$\text{mín}_{m_i} \sum_i \text{mín}_{t=1..k}\, d(i, m_t)$

Cada caso es asignado al medioide más cercano. El algoritmo paso a paso sería:

Paso 1 : Construir k medioides iniciales:

$m_1 = arg\,\text{mín}_{j=1..n} \sum_i d(i, m_j)$

Los restantes m_j son los puntos que decrecen la función objetivo lo máximo posible.

Paso 2 : repetir hasta la convergencia.

Considerar todos los pares de casos (i,j) siendo i un medioide y j un punto no medioide, e intercambiar i con j para el par que se produce una mayor disminución de la función objetivo.

Este método no necesita la matriz de casos por variables, bastaría con la matriz de distancias. Es un método mas robusto (menos sensible a casos atípicos) que el método de k-medias.

Para este algoritmo en R usamos la libraría fpc:

```
##Algoritmo de k-medoids
library(fpc)
# Estimar el núumero de grupos
pamResult <- pamk(m3)
#pamk devuelve: un objeto pam, el nº óptimo
# de grupos y p-values del test Duda-Hart.
```

```
#El nº óptimo de grupos
k <- pamResult$nc
cat("El número óptimo de grupo es ", k)
```

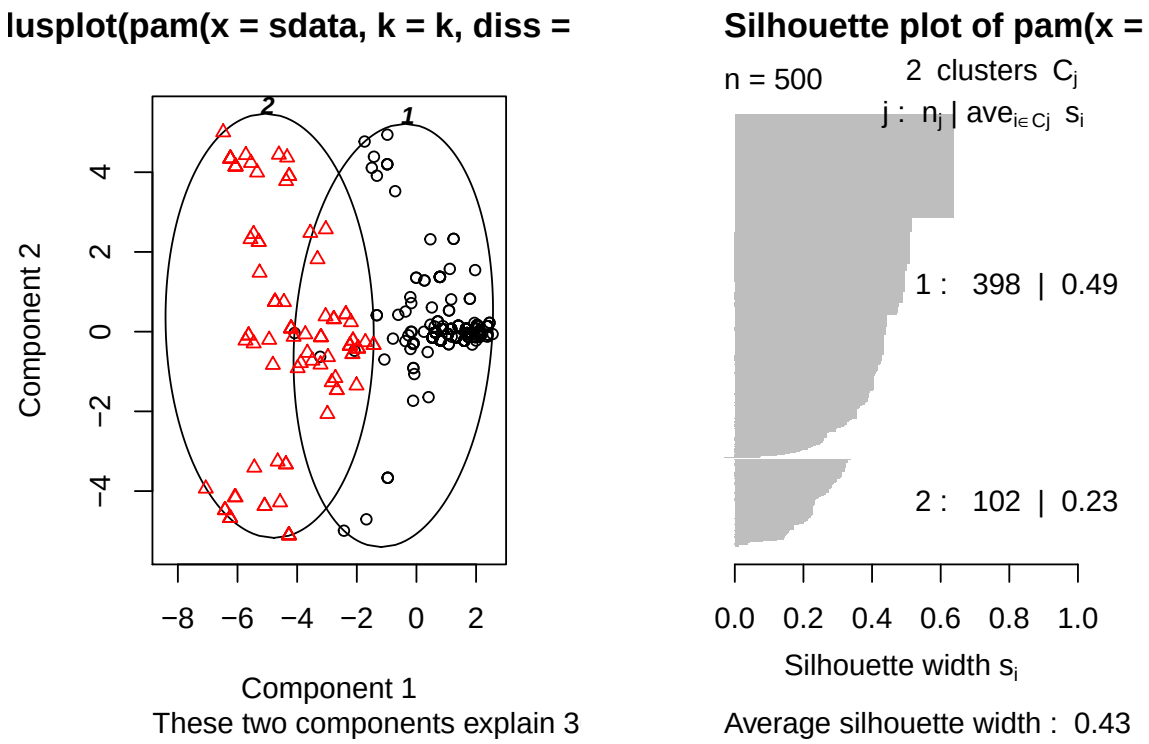
```
## El número óptimo de grupo es  2
```

```
pamResult <- pamResult$pamobject

for (i in 1:k) {
 cat(paste("cluster", i, ": "))
 cat(colnames(pamResult$medoids)
    [which(pamResult$medoids[i,]==1)], "\n")
 }
```

```
## cluster 1 :
## cluster 2 : anos envÃ favor hola cuenta
```

También podemos verlo de manera gráfica obteniendo una mejor visión sobre la distribución de los datos.

```
par(mfrow=c(1,2))
 plot(pamResult, color=F, labels=4,
 lines=0, cex=.8, col.clus=1,
 col.p=pamResult$clustering)
```




```
 library(cluster)
summary(silhouette(pamResult))
```

```
## Silhouette of 500 units in 2 clusters from pam(x = sdata, k = k, diss = diss) :
##  Cluster sizes and average silhouette widths:
##       398       102
## 0.4860005 0.2312061
## Individual silhouette widths:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.03068  0.31903  0.44068  0.43402  0.51420  0.63800
```

4.2.3. Métodos Jerárquicos

En los métodos jerárquicos los individuos no se particionan en clusters de una sola vez, sino que se van haciendo particiones sucesivas a " distintos niveles de agregación o agrupamiento ".

Fundamentalmente, los métodos jerárquicos suelen subdividirse en métodos aglomerativos (ascendentes), que van sucesivamente fusionando grupos en cada paso; y métodos divisivos (descendentes), que van desglosando en grupos cada vez más pequeños el conjunto total de datos.

Cabe concluir, por tanto, que la clusterización jerárquica produce taxones o clusters de diferentes niveles y estructurados de forma ordenada, para ser exactos, estableciendo una "jerarquía"; de ahí su nombre.

La representación de la jerarquía de clusters obtenida suele llevarse a cabo por medio de un diagrama en forma de árbol invertido llamado "dendograma", en el que las sucesivas

fusiones de las ramas a los distintos niveles nos informan de las sucesivas fusiones de los grupos en grupos de superior nivel (mayor tamaño, menor homogeneidad) sucesivamente.

El nivel de agrupamiento para cada fusión viene dado por un indicador llamado "valor cofenético" que debe ser proporcional a la distancia o disimilaridad considerada en la fusión (distancia de agrupamiento).Esta distancia o disimilaridad considerada en cada fusión estará definida, a veces, entre individuos y, otras, entre clusters; razón por la cual, ser necesario ampliar el concepto de distancia o disimilaridad de acuerdo con algún criterio que nos permita realizar el algoritmo de clasificación.

Una vez completamente definida la distancia para individuos, clusters y cluster-individuo, la clasificación jerárquica se puede llevar a cabo mediante un sencillo algoritmo general :

Paso 1 : Formamos la partición inicial:

P = $\{i_1\}$,$\{i_2\}$,...,$\{i_n\}$

Paso 2 : Determinamos los dos clusters más próximos i_i ,i_j y los agrupamos en uno solo.

Paso 3: Formamos la partición

P = $\{i_1\}$,$\{i_2\}$,...,$\{i_i u i_j\}$,...,$\{i_n\}$

Paso 4 : Repetimos los pasos 2 y 3 hasta obtener la partición Pr={W}.

Este algoritmo ser esencialmente el mismo para todos los métodos de clasificación jerárquica (ascendente); las diferencias residirán en el criterio de definición de la distancia entre clusters.

Veamos los diferentes métodos de unión de clusters según la definición de distancia.

1. Método de la distancia mínima (nearest neighbour o single linkage)

En este método se procede de acuerdo con el algoritmo general considerando la distancia ENTRE CLUSTERS como la distancia mínima entre los individuos más próximos

Este método es espacio-contractivo, esto es, tiende a aproximar los individuos más de lo que indicarían sus disimilaridades o distancias iniciales. El método del mínimo ha sido reivindicado "matemáticamente preferible" por sus propiedades por Jardine y Sibson . Sin embargo, ha sido muy criticado por ser muy sensible en aquellos casos en los que existen individuos perturbadores entre clusters bien diferenciados individuos intermedios) (casos con "ruido").

2. Método de la distancia máxima (furthest neighbour o complete linkage)

Este método, debido a Johnson ,utiliza el algoritmo general para la obtención de la clasificación jerárquica ascendente, pero considerando la distancia entre clusters con la distancia entre los individuos más alejados.

Por modificar la métrica en sentido inverso que el método anterior, este método es espacio-dilatante, en el sentido en que tiende a separar a los individuos en mayor medida que la indicada por sus disimilaridades iniciales.

El método de la distancia máxima se encuentra, como el anterior, en franca decadencia, ya que presenta los inconvenientes de alargar mucho el proceso y dar como resultado agrupaciones encadenadas.

Mientras el método de la distancia mínima asegura que la distancia entre los individuos m s próximos de un cluster ser siempre menor que la distancia entre elementos de distintos clusters, el de la distancia máxima va a asegurar que la distancia máxima dentro de un cluster será menor que la distancia entre cualquiera de sus elementos y los elementos más alejados de los demás clusters.

3. Método de la media (u.p.g.m.a.)

Los dos métodos anteriores, a pesar de poseer buenas propiedades teóricas tienen el inconveniente de distorsionar las medidas iniciales de disimilaridad, constringiendo o dilantando, respectivamente, la métrica. Una solución al problema fue el método ideado por Sokal y Michener, conocido como Group Average. Sokal y Michener propusieron utilizar como distancia entre un grupo I y un individuo j la media de las distancias entre los individuos del grupo I y el individuo j:

D (I,j) = $\frac{1}{N_I}$ S D(i,j)

Posteriormente, Lance y Williams extendieron la definición a la distancia entre dos grupos como la media de todas las distancias entre todos los pares de individuos de los dos grupos.

Este método es espacio-conservativo, ésto es, no hace variar considerablemente la métrica inicial, y resulta ser uno de los más utilizados, resolviendo de forma más aceptable la presencia de ruido.

4. Método del centroide

Fue propuesto originalmente, también, por Sokal y Michener, y utiliza como distancia entre grupos la distancia entre los centroides de cada grupo.Este método es, también, espacio-conservativo, pero presenta el inconveniente de dejarse influir excesivamente por los grupos de mayor tamaño. Esto hace que sea menos utilizado que el anterior.

5. Método de la mediana

La mayor desventaja del método del centroide es que si se fusionan dos grupos de diferente tamaño, el centroide del nuevo grupo queda más cerca del grupo de mayor tamaño y más alejado del de menor tamaño en proporción a sus diferencias de tamaño. Esto trae como consecuencia que durante el proceso aglomerativo de fusión se van perdiendo paulatinamente las propiedades de los grupos pequeños. Para evitar esto, puede suponerse, con independencia del tamaño que tengan los grupos en realidad, que los grupos son de igual tamaño. Llevando a cabo esta estrategia, la distancia entre un individuo o grupo K de centroide k y el grupo formado por la fusión de los grupos I y J de centroides i y j viene dada por la mediana del triángulo i,j, k. Razón por la cual Gower propuso el nombre de método (distancia) de la mediana.

Este método es, como el del centroide, espacio-conservativo, aunque también como él no resulta ser invariante ante transformaciones monótonas de la distancia empleada, cosa que sí ocurría con los tres primeros métodos.

6. Método de Ward

Ward propuso que la pérdida de información que se produce al integrar los distintos individuos en clusters puede medirse a través de la suma total de los cuadrados de las desviaciones entre cada punto (individuo) y la media del cluster en el que se integra.Para que el proceso de clusterización resulte óptimo, en el sentido de que los grupos formados

no distorsionen los datos originales, proponía la siguiente estrategia: En cada paso del análisis, considerar la posibilidad de la unión de cada par de grupos y optar por la fusión de aquellos dos grupos que menos incrementen la suma de los cuadrados de las desviaciones al unirse.

El método de Ward es uno de los más utilizados en la práctica; posee casi todas las ventajas del método de la media y suele ser más discriminativo en la determinación de los niveles de agrupación .Una investigación llevada a cabo por Kuiper y Fisher probó que este método era capaz de acertar mejor con la clasificación óptima que otros métodos (mínimo, máximo, media y centroide).

7. Método flexible de Lance y Williams

Las distintas distancias entre grupos definidas en los métodos anteriores se pueden expresar a través de una única formula recurrente de cuatro par metros; de forma que, para los distintos valores de éstos se generan las distintas distancias.En efecto, si consideramos el grupo formado por la fusión de los grupos I, J, (I,J) y el grupo exterior K, la distancia entre (I,J) y K puede expresarse como:

D((I,J),K) = a_I D(I,K)+ a_J D(J,K)+ b D(I,J)+ g |D(I,K)-D(J,K)|

En el caso del método del *mínimo*:

$a_I = a_J = 1/2$;; b = 0 ;; g = - 1/2

En el caso del método del *máximo*:

$a_I = a_J = 1/2$;; b = 0 ;; g = 1/2

En el caso del método de la *media*:

$\alpha_I = \frac{N_I}{N_I+N_J}$ $\alpha_J = \frac{N_J}{N_I+N_J}$ $\beta = \gamma = 0$

En el caso del método del *centroide*:

$\alpha_I = \frac{N_I}{N_I+N_J}$ $\alpha_J = \frac{N_J}{N_I+N_J}$ $\beta = -\alpha_I\alpha_J$ $\gamma = 0$

En el caso del método de la *mediana*:

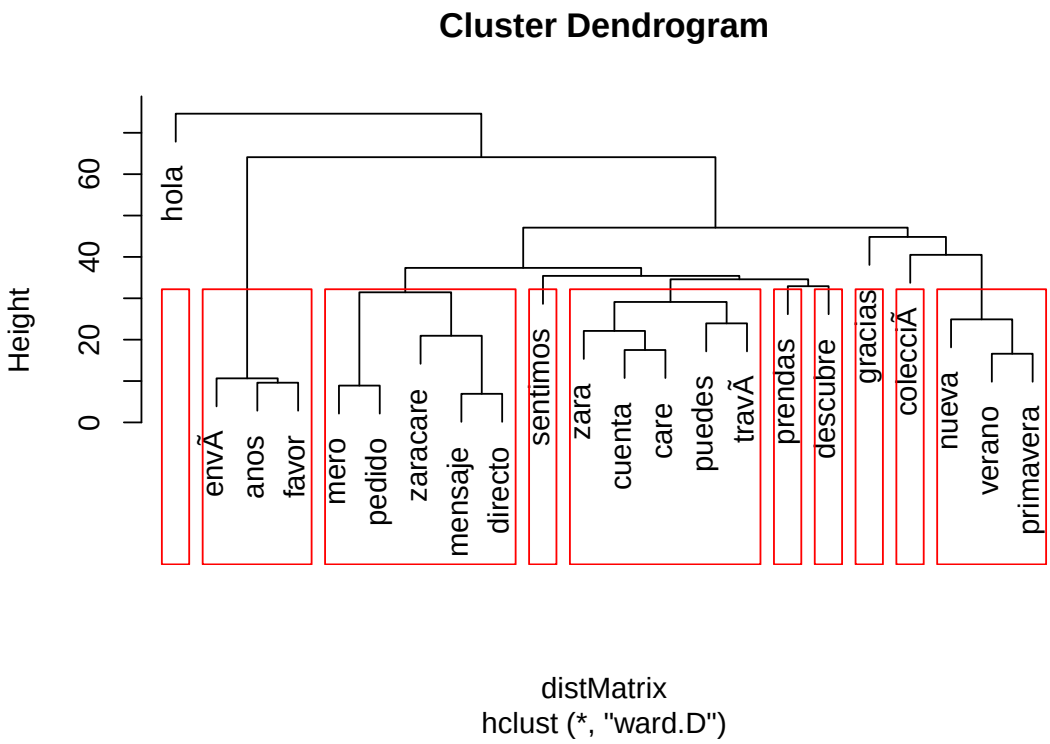
$a_I = a_J = 1/2$;; b = - 1/4 ;; g = 0

Y en el caso del método de *Ward*:

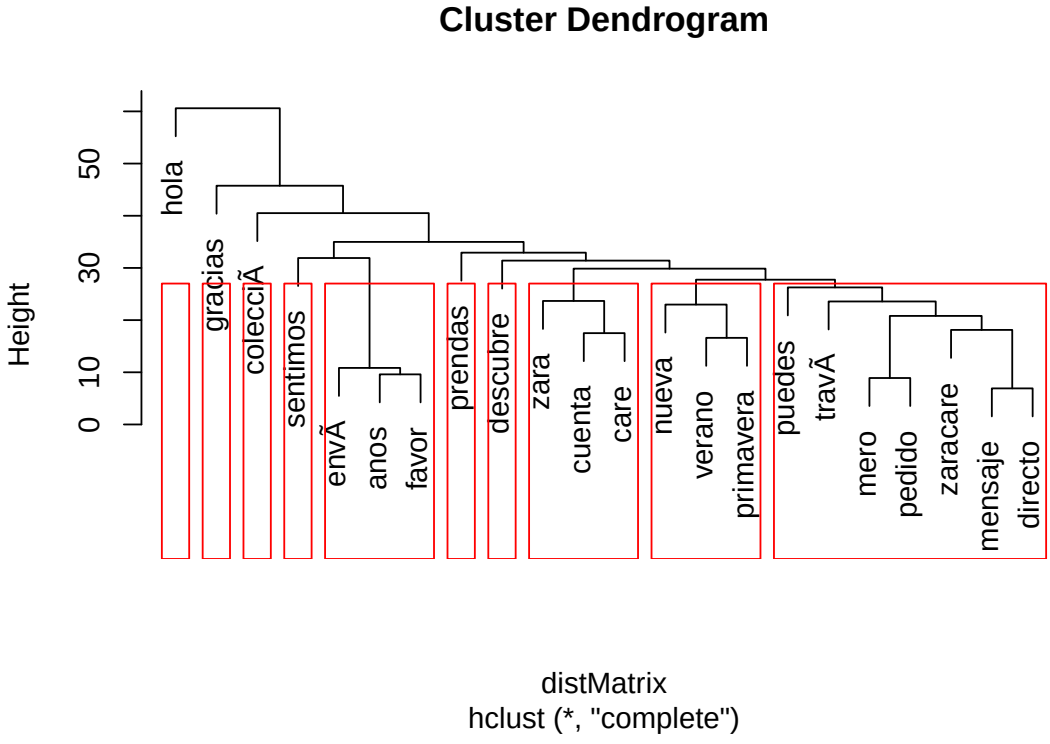
$\alpha_I = \frac{N_I+N_K}{N_I+N_J+N_K}$ $\alpha_J = \frac{N_J+N_K}{N_I+N_J+N_K}$ $\beta = \frac{-N_K}{N_I+N_J+N_K}$ $\gamma = 0$

Vamos a realizar una serie de gráficos con el fin de mostrar las diferentes salidas usando cada tipo de método que hemos visto anteriormente.

```
#Creación de grupos
distMatrix <- dist(scale(m2)) #Distancias entre los términos tipificados
fit <- hclust(distMatrix, method="ward.D") #Se hacen clusters.
plot(fit)#Dendrograma
rect.hclust(fit, k=10)
```

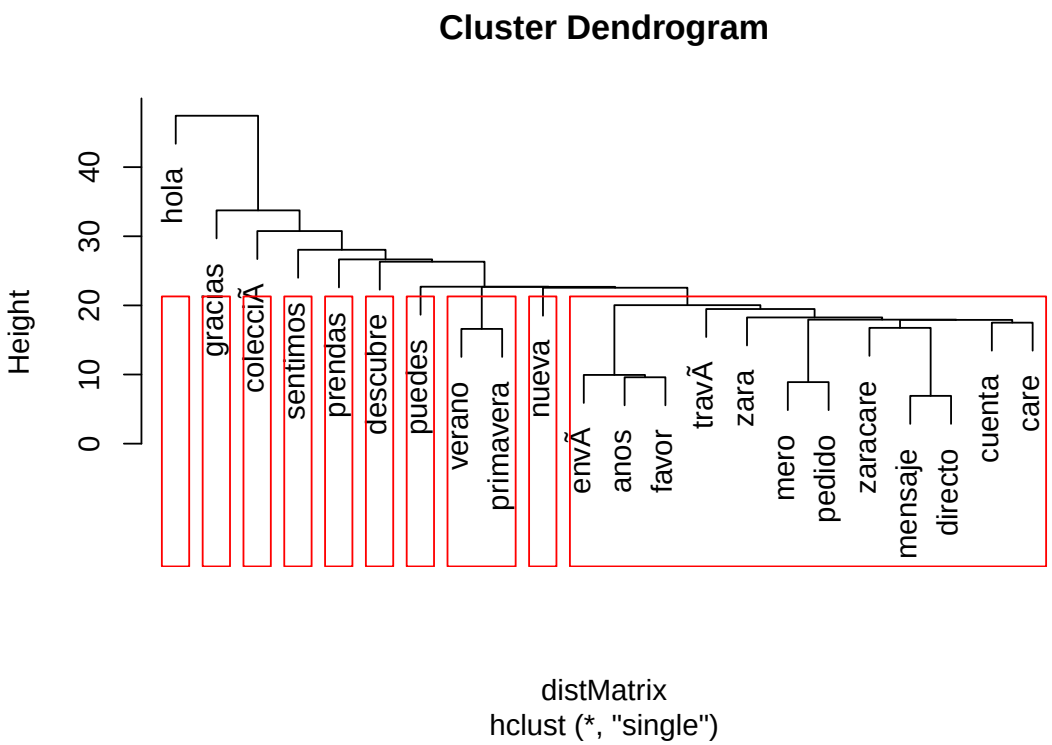



```
fitcomp <- hclust(distMatrix, method="complete")
plot(fitcomp)#Dendrograma
rect.hclust(fitcomp, k=10)
```

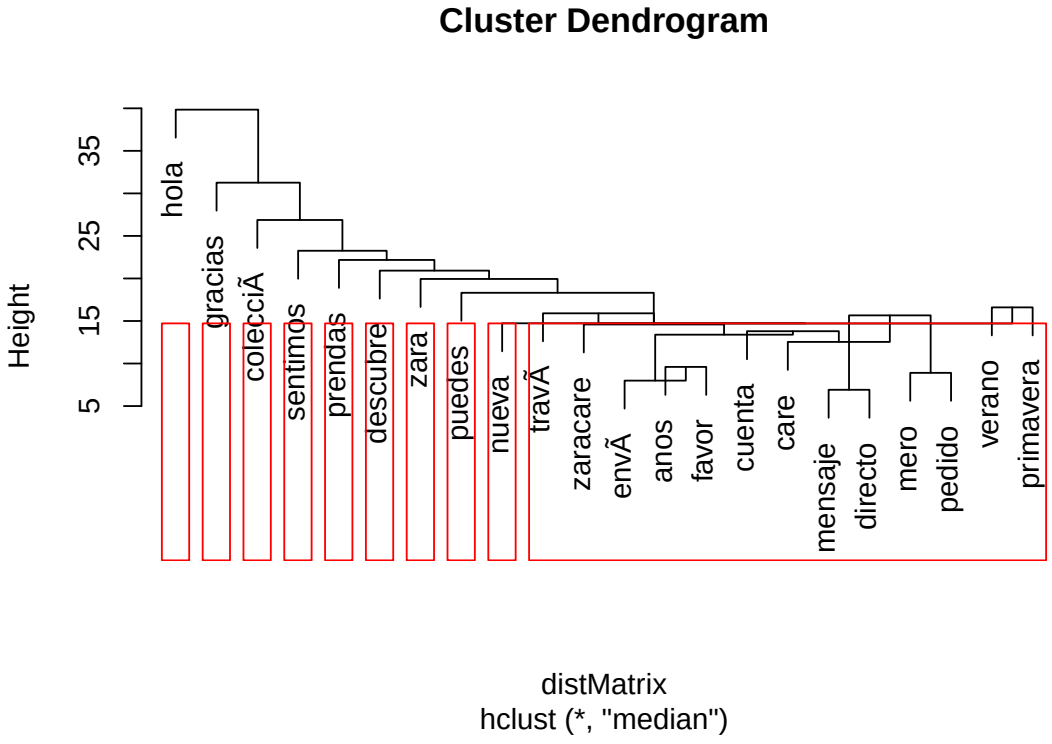



```
fitsingle <- hclust(distMatrix, method="single")
plot(fitsingle)#Dendrograma
```

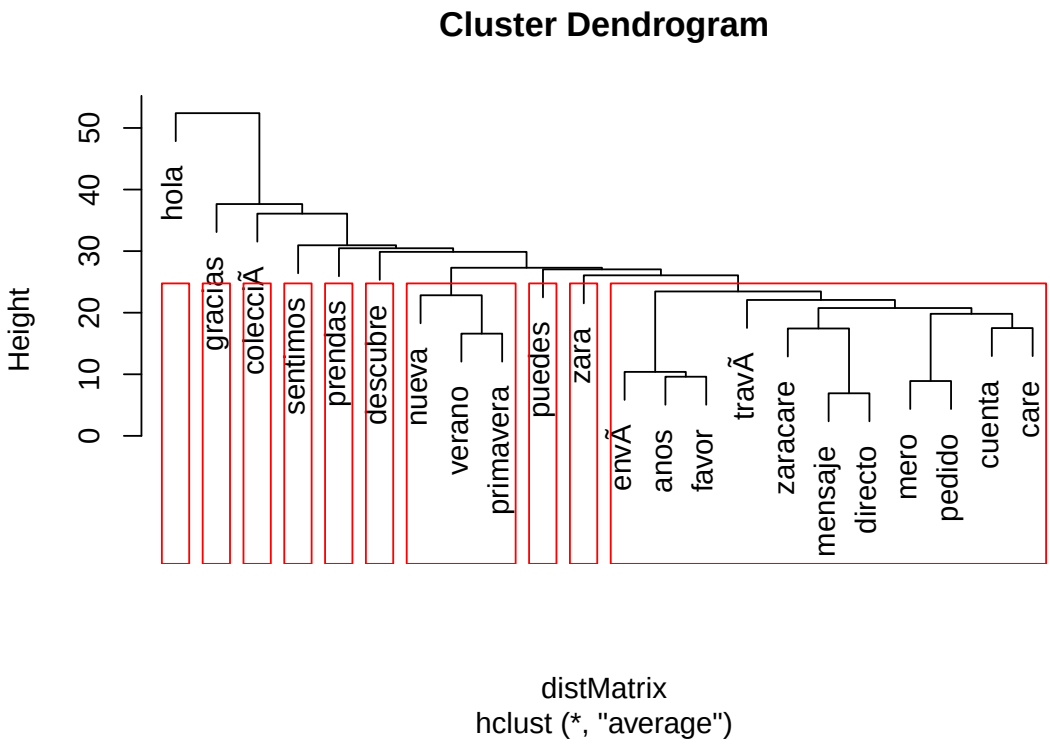
```
rect.hclust(fitsingle, k=10)
```



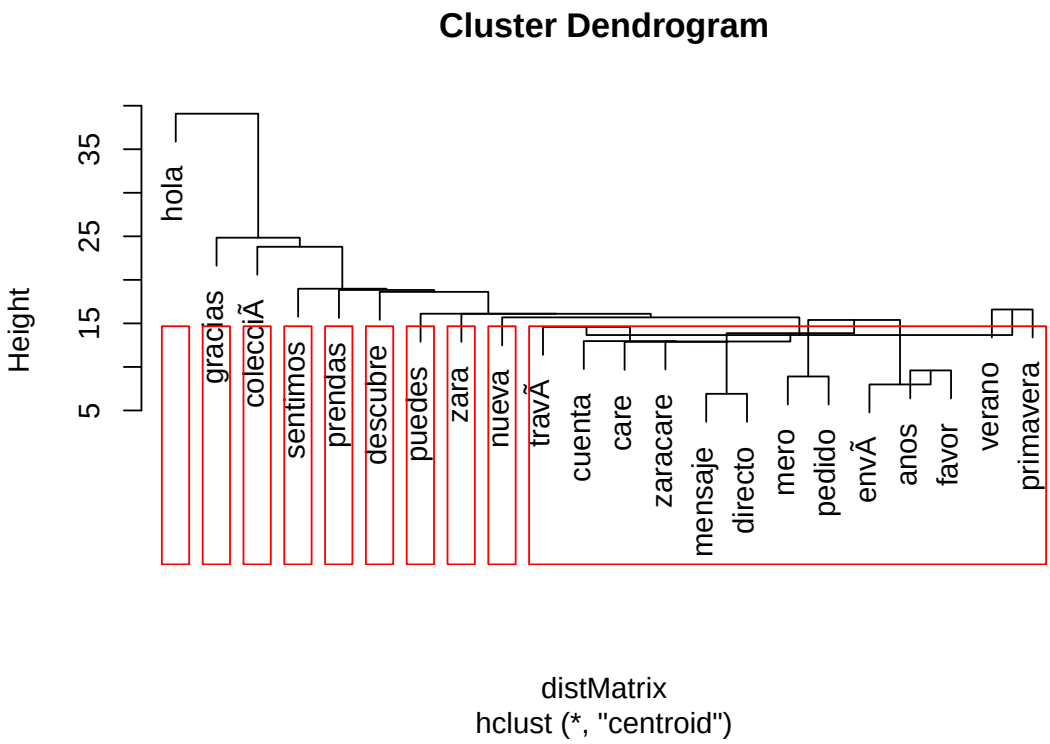

```
fitmedian <- hclust(distMatrix, method="median")
plot(fitmedian)#Dendrograma
rect.hclust(fitmedian, k=10)
```



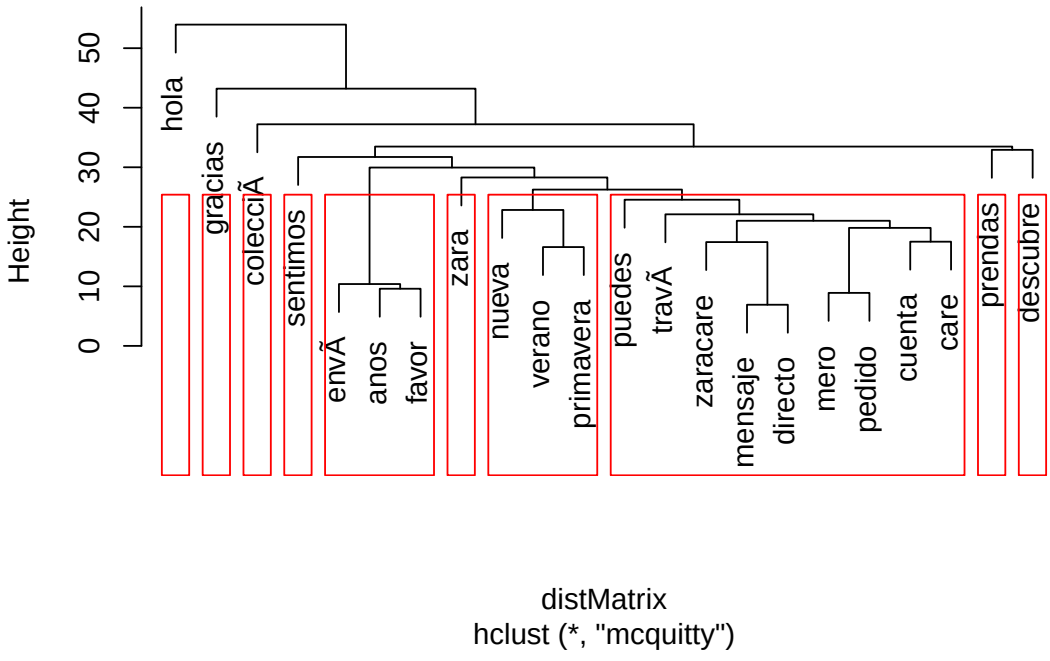
```
fitaver <- hclust(distMatrix, method="average")
plot(fitaver)#Dendrograma
rect.hclust(fitaver, k=10)
```




```
fitcentroid <- hclust(distMatrix, method="centroid")
plot(fitcentroid)#Dendrograma
rect.hclust(fitcentroid, k=10)
```




```
fitmc <- hclust(distMatrix, method="mcquitty")
plot(fitmc)#Dendrograma
rect.hclust(fitmc, k=10)
```



4.3. Geolocalización

En este apartado vamso a tomar un usuario determinado para poder ver la localización en un mapamundi de sus seguidores, hay que decir que solo tomaremos aquellos seguidores que hayan especificado correctamente una localización en sus perfiles, ya que mucho de estos o no lo completan al ser un campo no obligatorio o ponen situaciones que no son posibles de localizar.

En primer lugar obtenemos el usuario. En este caso tomaremos un usuario estandar ya que si tomaramos un usuario como zaraes necesitariamos compilar una gran cantidad de datos al tener un gran número de seguidores.

```
 # user <- getUser("********")
load("usuario_geo.RData")
```

Una vez tomado el usuario obtenemos de este los seguidores y seguidos de este usuario. Esto es posible gracias a la librería TwitteR que nos permite trabajar con el usuario de twitter directamente.

```
friends <- user$getFriends()
followers <- user$getFollowers()
```

Contruimos el vector que contendrá las localizaciones de nuestros seguidores.

```
localizaciones<-c()
```

Ejecutamos una recursion en la que obtenemos para cada seguidor su localización, en caso de tener completado este campo. Pasamos la tabla a un data.frame con el fin de tener una mejor visualización de las localizaciones así como de su frecuencia.

```
for (i in 1:length(followers))
{localizaciones<-c(localizaciones,followers[[i]]$location)
}
df_localizaciones<-as.data.frame(table(localizaciones))
head(df_localizaciones)
```

```
##                          localizaciones Freq
## 1                                        260
## 2                            · Sevilla ·   1
## 3  03/05/15 <U+2764><U+FE0F> tequieroo    1
## 4          c/ Reyes Católicos, Sevilla    1
## 5                               Rota Cai    1
## 6                                sevilla    1
```

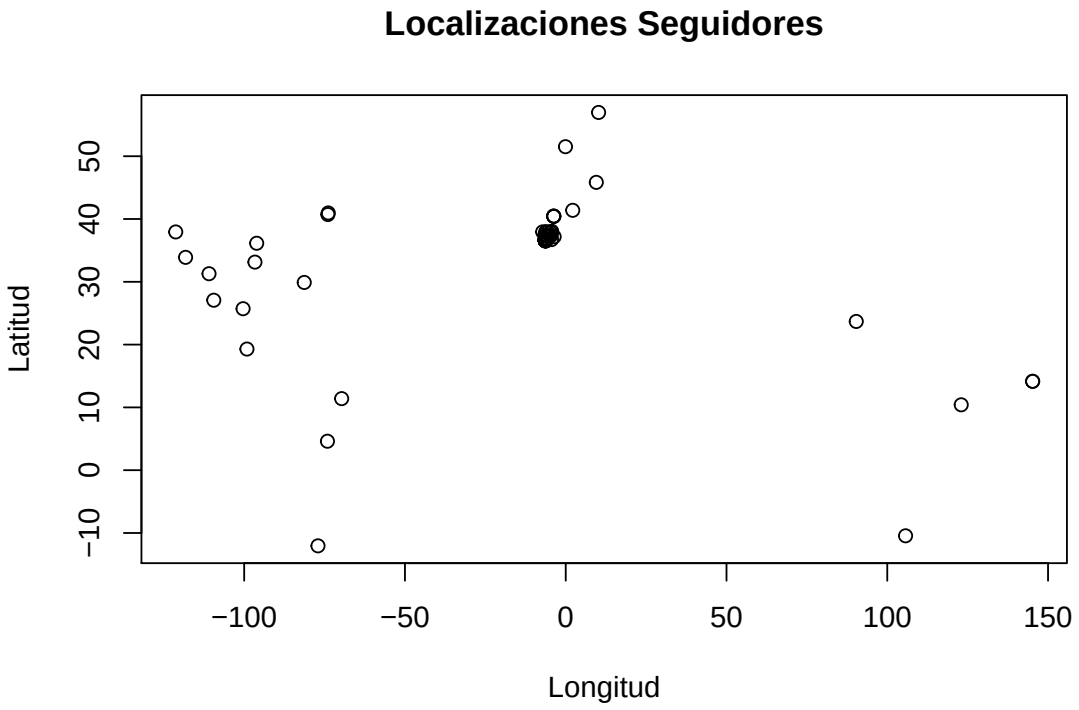
Para representar estas localizaciones vamos a usar un paquete llamado “ggmap” el cual nos permite obtener las latitudes y longitudes de una localización a partir de su búsqueda en Google Maps.
Hacer mapa de las localizaciones con Google map. Instalar y cargar paquete ggmap.

```
library(ggmap)
load(file= "df_seguidores.RData")
load(file = "coor_localizaciones.RData")
```

```
# coor_localizaciones<-geocode(as.character(df_localizaciones[,1]),
# source="google")
# df_coor_localizaciones_followers<-cbind(df_localizaciones,
# coor_localizaciones)
```

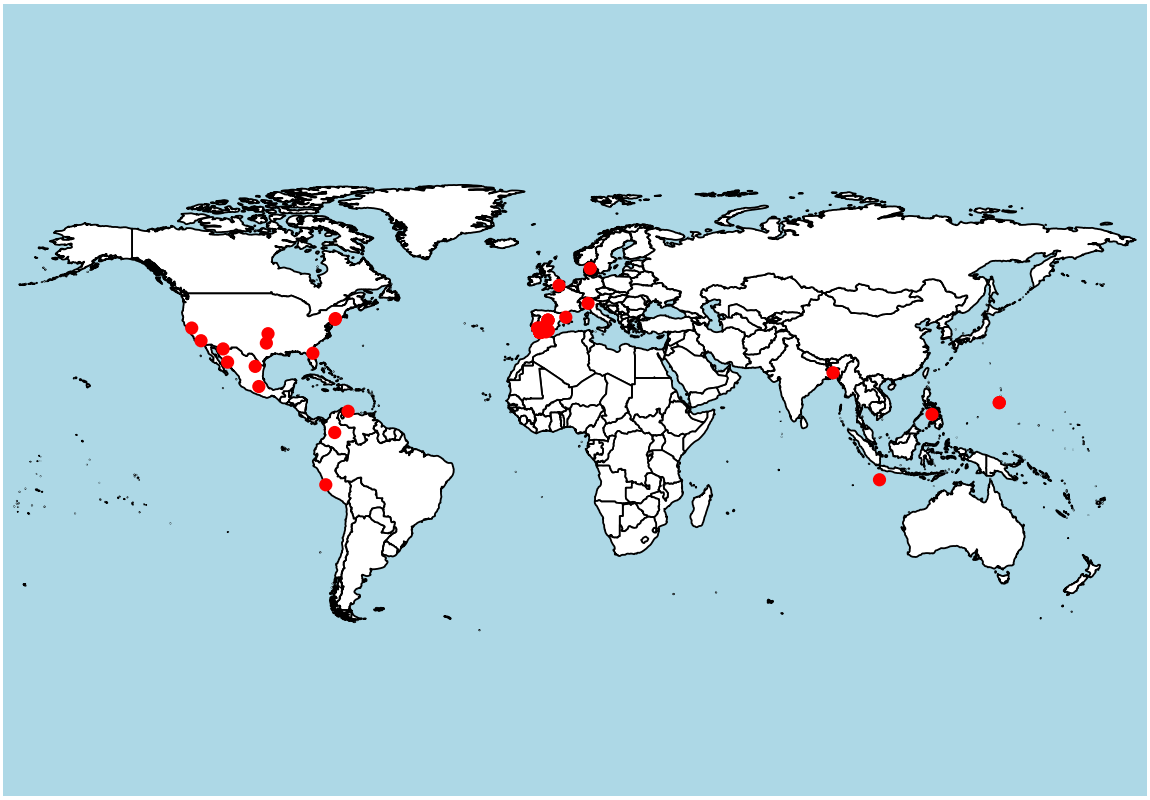
Una vez que tenemos las localizaciones en latitud y longitud de los seguidores podemos realizar un plot con el cual representariamos dichos números.

```
plot(coor_localizaciones,xlab="Longitud", ylab="Latitud",
     main="Localizaciones Seguidores")
```



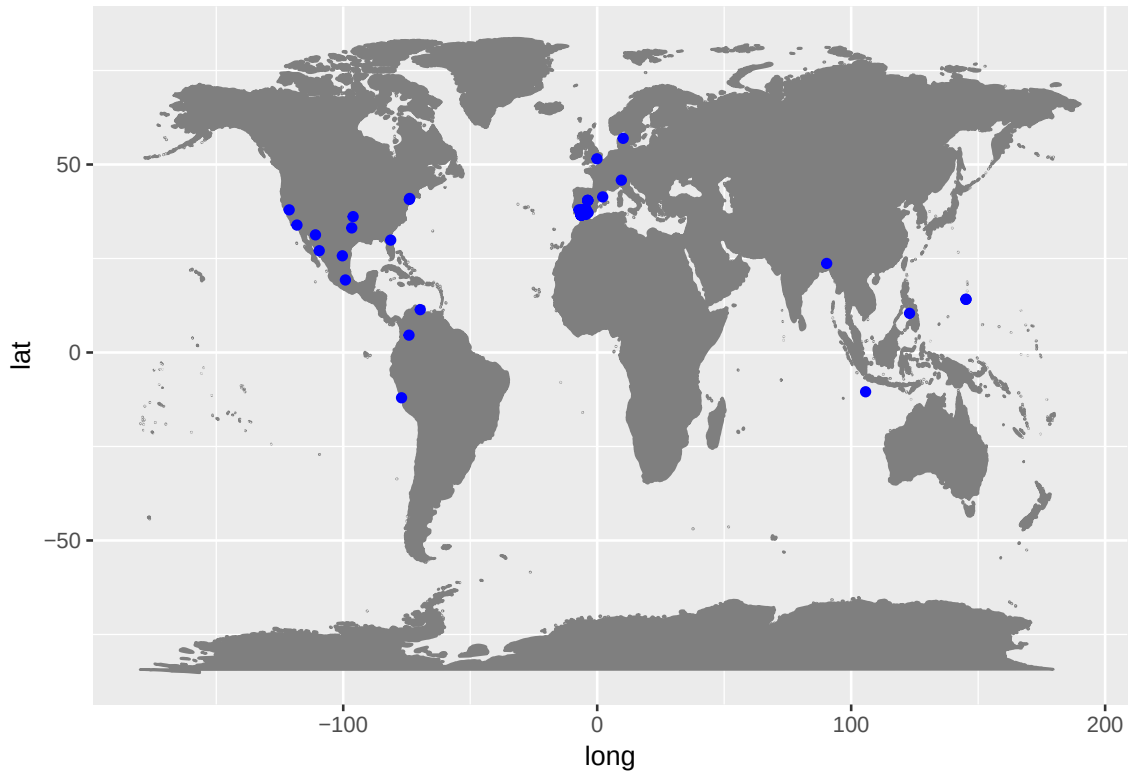
Para mejorar la visualización de nuestras localizaciones usaremos la libraría "maps" que nos permite diseñar un mapa y sobre el dibujaremos nuestras localizaciones, de esta manera obtenemos un mapamundi donde veremos la situación de nuestros usuarios.

```
library(tidyverse)
library(maps)
coordenadas<- drop_na(coor_localizaciones)
lon<-coordenadas$lon
lat<-coordenadas$lat
map("world", fill=TRUE, col="white", bg="lightblue",
    ylim=c(-60, 90), mar=c(0,0,0,0))
points(lon,lat, col="red", pch=16)
```



También podemos realizar nosotros directamente el mapa de manera que dibujaremos los bordes correspondientes a los continentes y luego localizaremos la situación de nuestros seguidores, todo ello usando la libraría "ggplot2"

```
mapWorld <- borders("world", colour="gray50", fill="gray50")
mp <- ggplot() +   mapWorld

mp <- mp+ geom_point(aes(x=lon, y=lat) ,color="blue", size=1.5)
mp
```



4.4. Hashtags

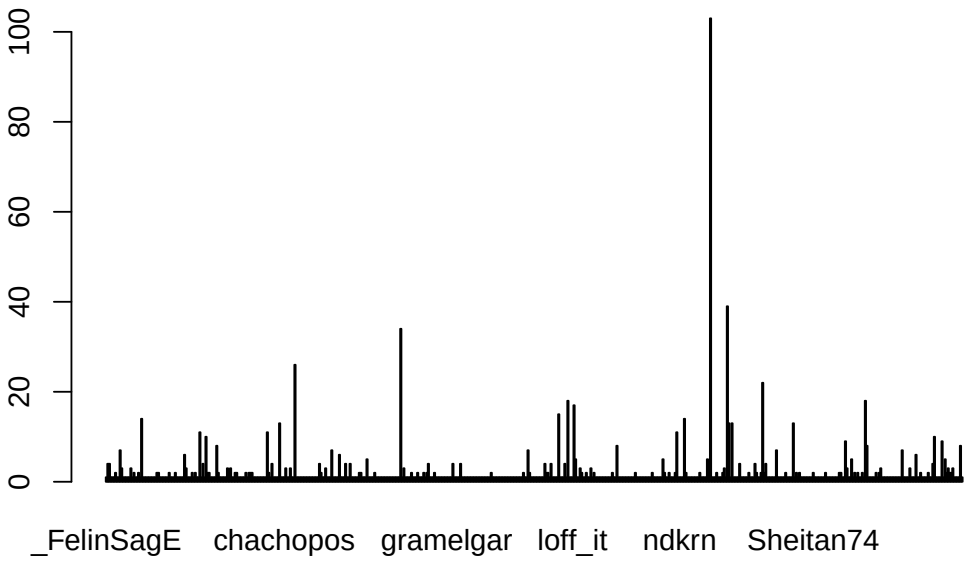
En este apartado vamos a realizar varios estudios sobre los hastags como cuales son los usuarios mas activos respecto a un hastag concreto, cuales son los mas retwiteados o los mas citados.

4.4.1. Usuarios más activos

Para empezar tomaremos una serie de aprox. 500 tweets sobre un hastag o palabra dado, como caso práctico usaremos el hastag #zapatos. Una vez tomado estos tweets procedemos a convertirlos en un data.frame y a tomar los nombres de usuario de aquellas personas que hayan tuiteado dicha palabra o hastag.

Realizamos un gráfico de barras con los principales nombres de usuarios de los que más han tuiteado dicho hastag.

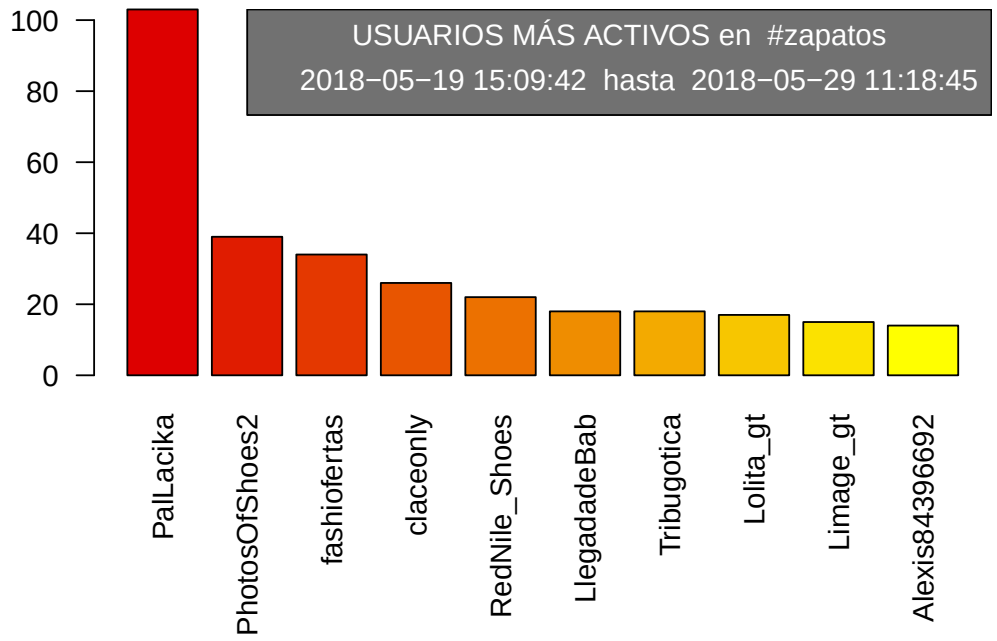
```
# rdmTweets = searchTwitter("#zapatos" , n=1500)
load("datos_hastag.RData")
df = do.call("rbind", lapply(rdmTweets, as.data.frame))
counts=table(df$screenName)
barplot(counts)
```




```
cc = sort(counts, decreasing=T)
```

Con el fin de realizar una presentación más simple solo tomaremos los 10 primeros usuarios con mayor actividad respecto de nuestro hastag, con los que realizamos un gráfico de barras más visible.

```
# Función para colorear
degradado = function (color1, color2, degradados)
{
  library(grDevices)
  palete = colorRampPalette(c(color1, color2))
  palete (degradados)
}
paleta = degradado ("#DD0000", "#FFFF00", 10)

last = function(x) { tail(x, n = 1) }

tweetsdf = twListToDF(rdmTweets)
par(mar=c(10,5,2,2), bg="white")
barplot(cc[1:10],las=2,cex.names =1, col=paleta)
legend("topright", title=("USUARIOS MÁS ACTIVOS en  #zapatos"),
       legend=paste(last(tweetsdf$created),
        " hasta ",tweetsdf$created[1]), text.col="#FFFFFF", bg="#333333B2",
       inset=0)
```



4.4.2. Usuarios más retuiteados

Veamos ahora los usuarios más retuiteados, pero primero debemos de eliminar posibles carácteres ya que vamos a trabajar con los textos de nuestros tweets con el fin de localizar la parte del tweets donde se hace referencia a que es un retuit.

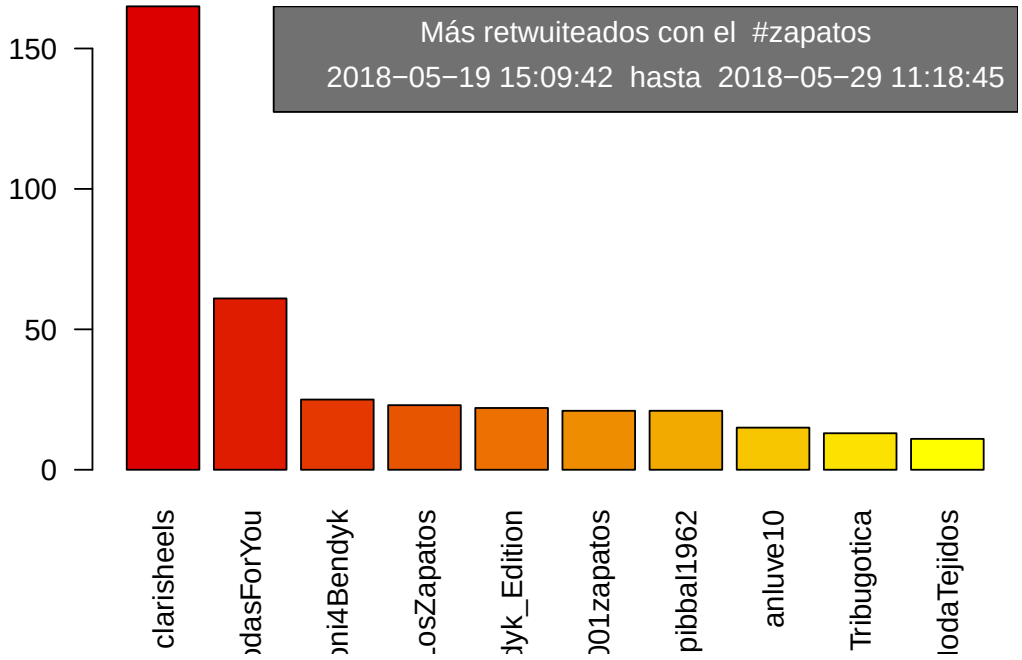
```
df$text=sapply(df$text,function(row) iconv(row,to='UTF-8'))
trim = function (x) sub('@','',x)
```

Ahora extraemos los usuarios que han sido retuiteados y generamos un data.frame que ordenaremos para obtener los 10 usuarios con mayor número de retuits.

```
df$rt=sapply(df$text,function(tweet)
  trim(str_match(tweet,"^RT (@[[:alnum:]_]*)")[2]))
df.ordered = as.data.frame(table(df$rt), row.names=TRUE)
order.freq = order(df.ordered$Freq, decreasing=T)
```

Como en el primer apartado realizamos un gráfico de barras con aquellos 10 usuarios.

```
barplot(df.ordered$Freq[order.freq][1:10],
        las=2,cex.names =1, col=paleta,
        names.arg=rownames(df.ordered)[order.freq][1:10])
legend("topright", title=paste("Más retwuiteados con el  #zapatos" ), legend=paste(la
                               " hasta ",tweetsdf$created[1]),
       text.col="#FFFFFF", bg="#333333B2", inset=0)
```



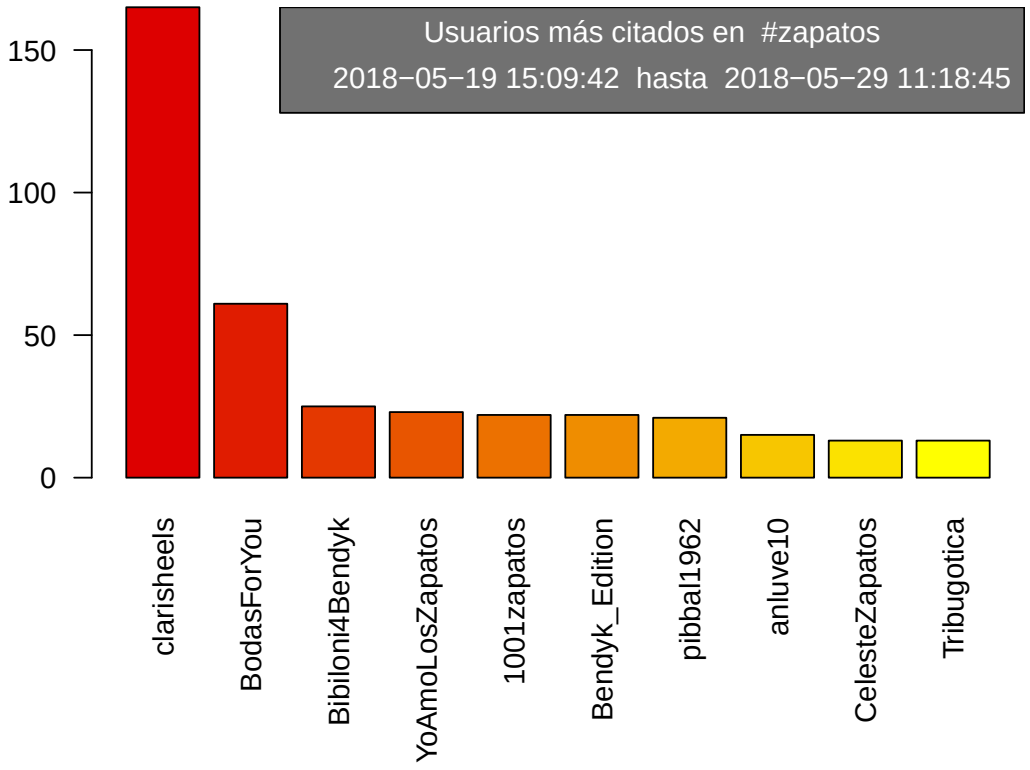
4.4.3. Usuarios más citados

Por último vamos a ver que usuarios son los más citados usando el hastag tomado. Procedemos de igual manera al anterior, extraiendo la información del tweet directamente y posteriormente crear el data.frame que ajustaremos borrando el arroba de los nombres del usuario y ordenandolo para nuestro interés.

```
df$to=sapply(df$text,function(tweet)
  str_extract(tweet,"(@[[:alnum:]_]*)"))
df$to=sapply(df$to,function(name) trim(name))
df.2 = as.data.frame(table(df$to), row.names=TRUE)
order2.freq = order(df.2$Freq, decreasing=T)
```

Realizamos un gráfico de barras para ver gráficamente el resultado de los 10 usuarios más citados respecto al hastag.

```
par(mar=c(8,3,1,3))
barplot(df.2$Freq[order2.freq][1:10], las=2,cex.names =1,
        col=paleta, names.arg=rownames(df.2)[order2.freq][1:10])
legend("topright", title=paste("Usuarios más citados en  #zapatos"), legend=paste(las
       text.col="#FFFFFF", bg="#333333B2", inset=0)
```

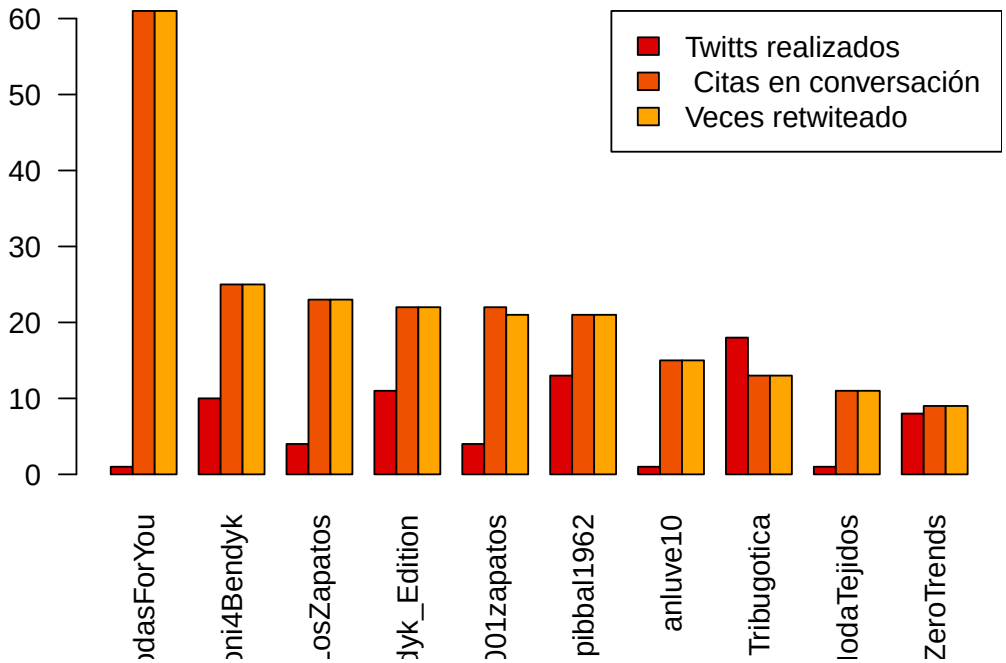


4.4.4. Resumen

Vamos a realizar un resumen sobre todo lo visto anteriormente sobre el hastag.

```
### Renombro columnas en todos mis data frames
cc = sort(counts, decreasing=T)
colnames(df.ordered) = c("Veces retwiteado")
colnames(df.2) = c( " Citas en conversación " )
cc = as.data.frame(cc)
row.names(cc)= cc$Var1
colnames(cc) = c("Usuario","Twitts realizados")


### Comienzo a juntarlos, renombrar columnas y
### borrar las sobrantes

total1 = merge(cc,df.2, by="row.names", all.x=T)
rownames(total1) = total1[,1]
drops = c("Row.names")
total1 = total1[,!(names(total1) %in% drops)]

total = merge(total1,df.ordered, by="row.names", all.x=T)
rownames(total) = total[,1]
total = total[,!(names(total) %in% drops)]

## Elimino los NA por 0 para quedarme una tabla más útil
```

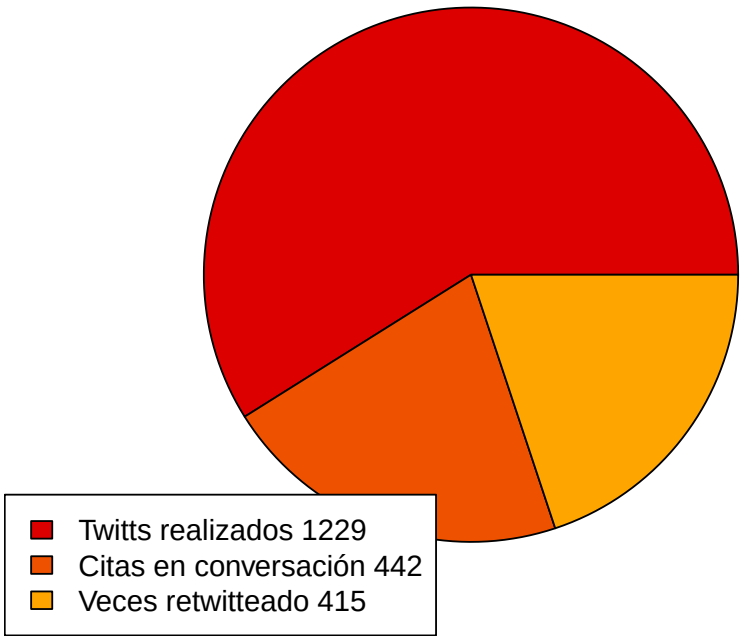
```
total[is.na(total)] = 0
total = total[,-1]

engagement = total[,2]+total[,3]
total ["Engagement"] = engagement
order.freq = order(total[,4], decreasing=T)
paleta2 = degradado ("#DD0000", "orange", 3)

# Gráfico de barras del resumen
barplot( rbind(total[,1][order.freq][1:10],
               total[,2][order.freq][1:10],
               total[,3][order.freq][1:10]), names=row.names(total)[order.freq][1:10]
         beside=TRUE, las=2,cex.names =1, col=paleta2)
legend("topright", legend=names(total)[c(1,2,3)],
       fill=paleta2)
```




```
a = colSums(total)
b= c(nrow(total), a)
b= b[-5]
names(b) = c("Usuarios", "Twitts realizados",
             "Citas en conversación", "Veces retwitteado")
print(b)
```

```
##              Usuarios     Twitts realizados Citas en conversación
##                   559                  1229                   442
##     Veces retwitteado
##                   415
```

```
lbls = paste(names(b),b, sep=" ")
par(mar=c(1,3,1,3), bg="white")
pie(b[2:4], main=paste("Actividad de los ",b[1],
                       " twitteros #zapatos "),
    col=paleta2, label="")
legend("bottomleft", legend=lbls[2:4], fill=paleta2)
```



Capítulo 5

Análisis de Sentimientos

El análisis de sentimiento, también conocido como minería de opinión (opinion mining), es un término muy discutido pero a menudo incomprendido.

En términos generales, el análisis de sentimiento intenta determinar la actitud de un interlocutor o usuario con respecto a algún tema o la polaridad contextual general de un documento. La actitud puede ser su juicio o evaluación, estado afectivo (o sea, el estado emocional del autor al momento de escribir), o la intención comunicativa emocional (o sea, el efecto emocional que el autor intenta causar en el lector).

En la última década, el análisis de sentimientos ha despertado un creciente interés. Resulta un gran reto para las tecnologías del lenguaje, ya que obtener buenos resultados es mucho más difícil de lo que muchos creen. La tarea de clasificar automáticamente un texto escrito en un lenguaje natural en un sentimiento positivo o negativo, opinión o subjetividad (Pang and Lee, 2008), es a veces tan complicada que incluso es difícil poner de acuerdo a diferentes anotadores humanos sobre la clasificación a asignar a un texto dado. La interpretación personal de un individuo es diferente de la de los demás, y además se ve afectada por factores culturales y experiencias propias de cada persona. Y la tarea es aún más difícil cuanto más corto sea el texto, y peor escrito esté, como es el caso de los mensajes en redes sociales como Twitter o Facebook.

El problema se ha abordado principalmente desde dos enfoques diferentes: técnicas de aprendizaje computacional (Pang, Lee, and Vaithyanathan, 2002) y aproximaciones semánticas (Turney, 2002).

Los **enfoques semánticos** se caracterizan por el uso de diccionarios de términos (lexicons) con orientación semántica de polaridad u opinión. Típicamente los sistemas preprocesan el texto y lo dividen en palabras, con la apropiada eliminación de las palabras de parada y una normalización lingüística por stemming o lematización, y luego comprueban la aparición de los términos del lexicon para asignar el valor de polaridad del texto mediante la suma del valores de polaridad de los términos. Típicamente los sistemas además incluyen un tratamiento más o menos avanzado de a) términos modificadores (como muy, poco, demasiado) que aumentan o reducen la polaridad del o los términos a los que acompañan, y b) términos inversores o negadores (como no, tampoco), que invierten la polaridad de los términos a los que afectan.

Por otra parte, los enfoques basados en **aprendizaje computacional** consisten en entrenar un clasificador usando un algoritmo de aprendizaje supervisado a partir de una

colección de textos anotados, donde cada texto habitualmente se representa con un vector de palabras (bag of words), n-gramas o skip-grams, en combinación con otro tipo de características semánticas que intentan modelar la estructura sintáctica de las frases, la intensificación, la negación, la subjetividad o la ironía. Los sistemas utilizan diversas técnicas, aunque las más populares son los clasificadores basados en *SVM (Support Vector Machines)*, *Naive Bayes* y *KNN (K-Nearest Neighbor)*. En las investigaciones más recientes se han empezado a utilizar otras técnicas más avanzadas, como *LSA (Latent Semantic Analisis)* e incluso *Deep Learning*.

La ventaja principal de los enfoques semánticos es que los errores son relativamente sencillos de corregir, añadiendo cuantos términos fuera necesario, y se podría obtener una precisión tan alta como se quisiera, simplemente invirtiendo más tiempo en la construcción del lexicon. En este sentido, los enfoques basados en aprendizaje automático suelen ser una caja negra en la que corregir errores o añadir nuevo conocimiento es más complicado, y muchas veces sólo es posible ampliando la colección de textos y volviendo a entrenar el modelo.

Por otra parte, la ventaja de los enfoques basados en aprendizaje automático es que cuesta muy poco construir un analizador de sentimientos a partir de la colección de textos etiquetados, ya que la tarea de modelado reside en el algoritmo. Por ello es relativamente fácil construir clasificadores adaptados a un dominio determinado. En contraposición, el esfuerzo para construir un lexicon para un cierto dominio, empezando de cero, es muy elevado, porque se basa en mucho trabajo manual, así que en general son menos adaptables.

Por ello vamos a aplicar diferentes paquetes para conocer la opinión sobre una marca en redes sociales utilizando R, gracias a los cuales podremos clasificar las emociones y polaridad de los tweets.

Hay que señalar que el análisis de sentimiento nunca es 100 % preciso y depende mucho del idioma, contexto (Diccionarios), el cuerpo con el que se ha entrenado, ironía/sarcasmo ("que bien que se me haya roto el coche ahora").

Ni siquiera nos ponemos nosotros de acuerdo: numerosos estudios muestran que el porcentaje de acuerdo entre personas al determinar el sentimiento de un texto es tan sólo del 70 %-80 %.

Para llevar a cabo nuestros análisis tanto de la polaridad de los comentarios como de los sentimientos necesitaremos instalar una serie de paquetes como "sentimentr" obtenido a partir del repositorio CRAN, este paquete es específico para ello. Además de dicho paquete debemos de buscar, como mencionamos antes, un diccionario de palabras (lexicon) a partir del cual evaluar los sentimientos y polaridad de las palabras, en nuestro caso, hemos tomado un diccionario que consta de aproximadamente 2000 palabras con su respectiva polaridad y su sentimiento.

A pesar de tener el paquete "sentimentr" hemos tenido que modificar las funciones que vienen en él puesto que estas utilizaban un diccionario en inglés. Una vez cambiado las funciones y tomando el diccionario correcto cogemos el ejemplo de los tweets correspondientes al usuario: zaraes

```
library(xlsx)
library(sentimentr)
```

```
library(syuzhet)
library(devtools)
```

```
source("classify_emotion.R")
source("classify_polarity.R")
source("create_matrix.R")
```

```
# zara_tweets = userTimeline("zaraes", n=5000)
load("zara_sentim.RData")
zara_txt = sapply(zara_tweets, function(x) x$getText())
clean.text <- function(some_txt)
{
  some_txt = gsub("(RT|via)((?:\\b\\W*@\\w+)+)", "", some_txt)
  some_txt = gsub("@\\w+", "", some_txt)
  some_txt = gsub("[[:punct:]]", "", some_txt)
  some_txt = gsub("[[:digit:]]", "", some_txt)
  some_txt = gsub("http\\w+", "", some_txt)
  some_txt = gsub("[ \t]{2,}", "", some_txt)
  some_txt = gsub("^\\s+|\\s+$", "", some_txt)

  # Definimos la función "tolower error handling"
  try.tolower = function(x)
  {
    y = NA
    try_error = tryCatch(tolower(x), error=function(e) e)
    if (!inherits(try_error, "error"))
      y = tolower(x)
    return(y)
  }

  some_txt = sapply(some_txt, try.tolower)
  some_txt = some_txt[some_txt != ""]
  names(some_txt) = NULL
  return(some_txt)
}

# Limpiamos los textos
zara_clean = clean.text(zara_txt)
zara_limpio = removeWords(zara_clean,
                  c(stopwords("spanish"), "zaraes"))
zara_limpio = removeWords(zara_limpio,stopwords("english"))
```

Una vez tomado y realizado de manera simplificada la limpieza de los tweets pasamos a usar la función classify_emotion para clasificar mediante el algoritmo de naives bayes los sentimientos de nuesstros tweets.

```
class_emo <- classify_emotion(zara_limpio,
                              algorithm="bayes", prior=1.0)
emotion = class_emo[,7]
```

```
emotion[is.na(emotion)] = "Desconocido"
head(emotion,20)
```

```
##  [1] "Tristeza" "Alegria"  "Sorpresa" "Sorpresa" "Alegria"  "Tristeza"
##  [7] "Sorpresa" "Alegria"  "Tristeza" "Alegria"  "Sorpresa" "Sorpresa"
## [13] "Sorpresa" "Enojo"    "Enojo"    "Enojo"    "Sorpresa" "Sorpresa"
## [19] "Enojo"    "Alegria"
```

Pasamos ahora a clasificar la polaridad mediante la otra función que tenemos.

```
class_pol = classify_polarity(zara_limpio, algorithm="bayes")
polarity = class_pol[,4]
head(polarity,20)
```

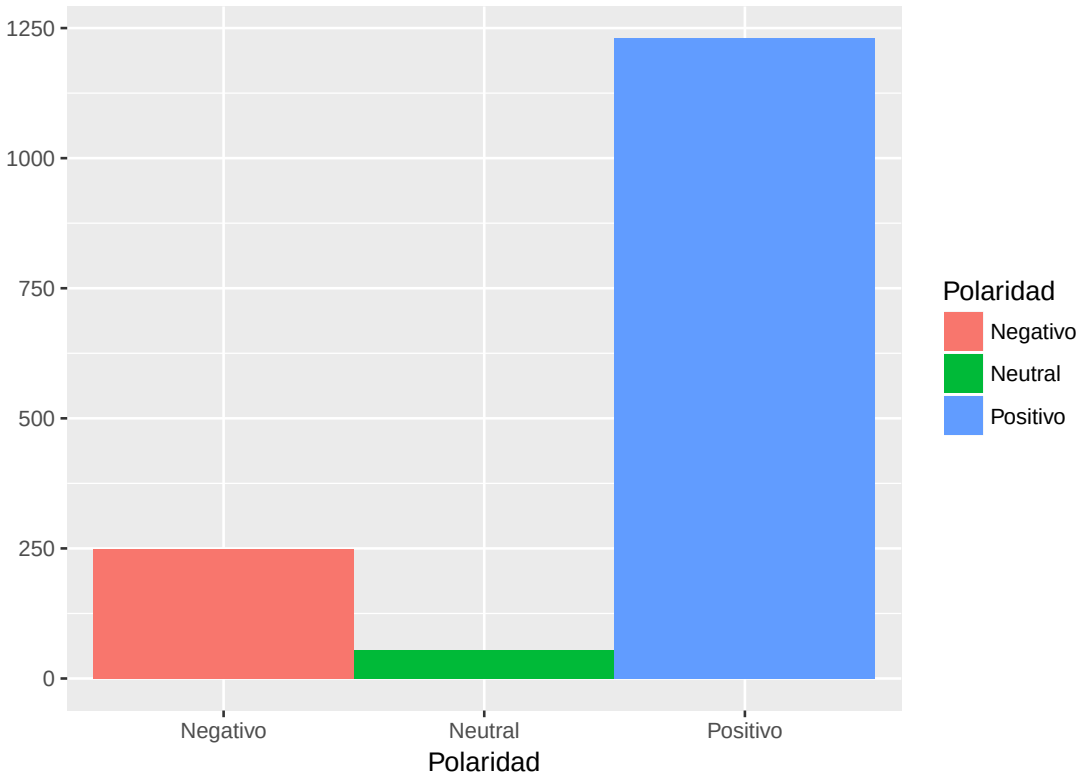
```
##  [1] "Negativo" "Positivo" "Positivo" "Positivo" "Positivo" "Negativo"
##  [7] "Positivo" "Positivo" "Negativo" "Positivo" "Negativo" "Positivo"
## [13] "Positivo" "Positivo" "Neutral"  "Positivo" "Positivo" "Positivo"
## [19] "Positivo" "Positivo"
```

Una vez realizada la clasificación creamos un data.frame que contenga ambas clasificaciones con el fin de realizar una representación que nos permita ver de manera simple y concisa nuestros resultados.

```
sent_df = data.frame(text=zara_limpio, emotion=emotion,
polarity=polarity, stringsAsFactors=FALSE)
```

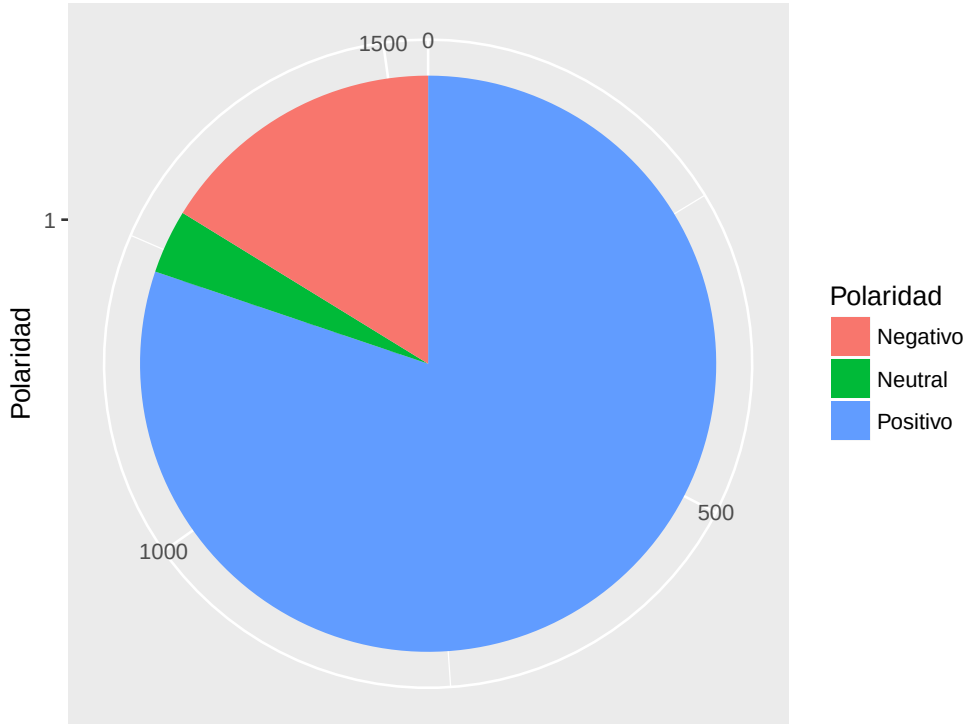
Mediante un gráfico de barras representamos la polaridad de los tweets así como la cantidad de tweets que pertenecen a ella.

```
Polaridad=sent_df$polarity
ggplot(sent_df,aes(x=polarity, fill = Polaridad))+
  geom_bar(width = 1)+xlab("Polaridad")+ylab("")
```



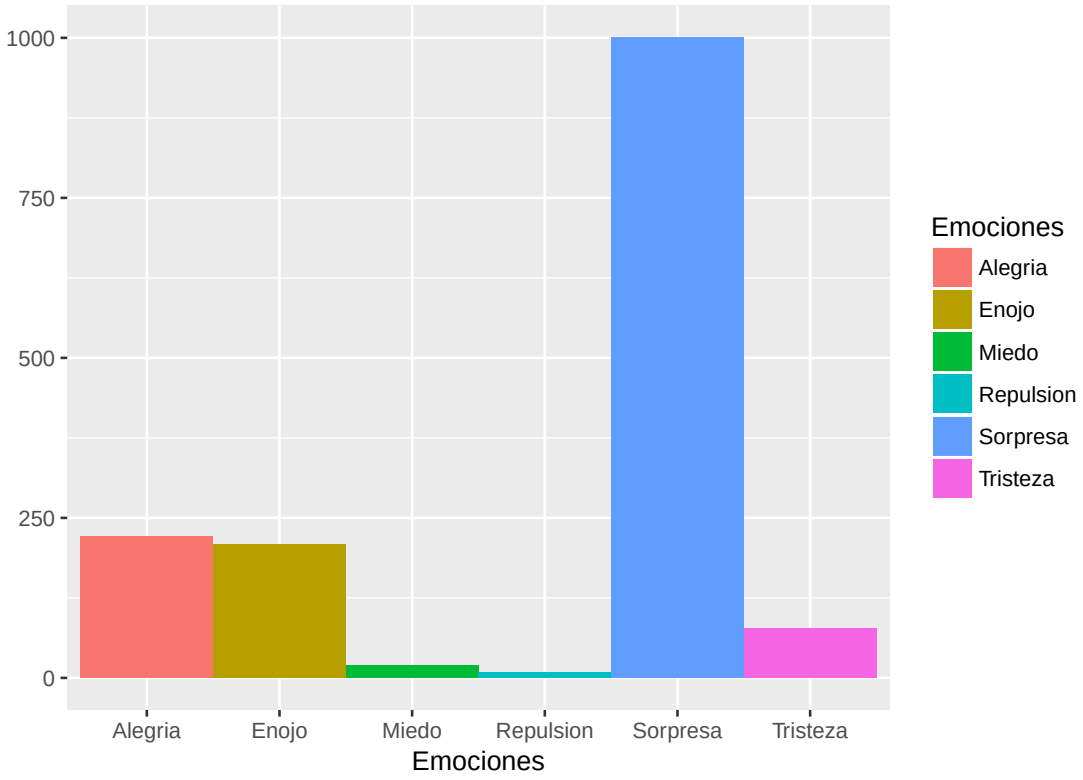
En el gráfico de barras podemos ver que la gran mayoria de las palabras de los tweets son categorizadas como positivo.

```
ggplot(sent_df,aes(x=factor(1),fill = Polaridad))+
  geom_bar(width = 1)+
  coord_polar(theta = 'y')+xlab("Polaridad")+ylab("")
```

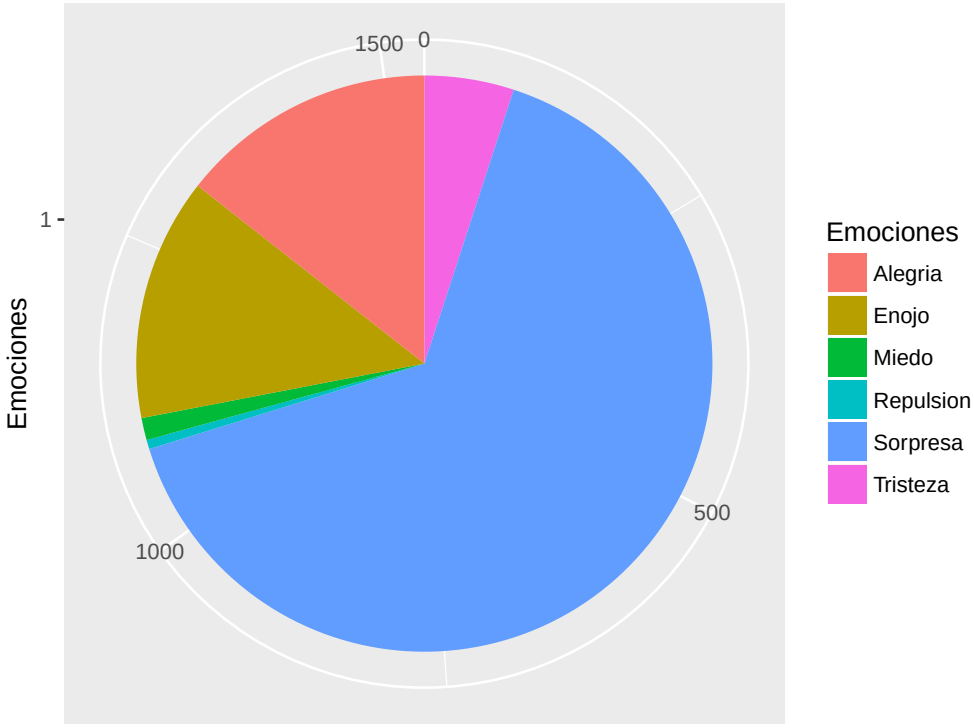


De igual manera, obtenemos el gráfico para los sentimientos de los tweets.

```
Emociones = sent_df$emotion
ggplot(sent_df,aes(x=emotion, fill = Emociones))+
  geom_bar(width = 1)+xlab("Emociones")+ylab("")
```




```
ggplot(sent_df,aes(x=factor(1),fill = Emociones))+
  geom_bar(width = 1)+
  coord_polar(theta = 'y')+xlab("Emociones")+ylab("")
```



Para la representación de los resultados obtenidos también podemos generar una nube de alabras a partir de la cual veamos para una polaridad especifica que palabras son las mas llamativas. Para esto debemos de separar y tomar aquellos tweets que pertenezcan a cada polaridad para posteriormente aplicar lo explicado en los capitulos anteriores, es decir, generar un corpus el cual representaremos.

```
tweets_neg <- sent_df$text[sent_df$polarity=="Negativo"]
tweets_neu <- sent_df$text[sent_df$polarity=="Neutral"]
tweets_pos <- sent_df$text[sent_df$polarity=="Positivo"]
```

```
require(tm)
tweets_neg = removeWords(tweets_neg, stopwords("spanish"))
tweets_pos = removeWords(tweets_pos, stopwords("spanish"))
tweets_neu = removeWords(tweets_neu, stopwords("spanish"))
```

```
corpus_neg <- Corpus(VectorSource(tweets_neg))
corpus_pos <- Corpus(VectorSource(tweets_pos))
```

```
library(wordcloud)
library(RColorBrewer)
wordcloud(corpus_neg, random.order= FALSE,
          colors=brewer.pal(8,"Dark2"),scale = c(3,.5))
```


```
wordcloud(corpus_pos, max.words=100, random.order= FALSE,
          colors=brewer.pal(8,"Dark2"),scale = c(3,.5))
```

Además de la nube de palabras que generamos con aquellas que pertenezcan a una polaridad dada también podemos realizar una nube de palabras comparativa en la que tomemos aquellas palabras que destacan dentro de cada sentimiento o polaridad y repre-

sentarlo todo en una misma nube. De estamanera, conseguimos ver gráficamente aquellas palabras más representativas de sus sentimientos.

En primer lugar vamos a verlo para los sentmientos.

```
emos = levels(factor(sent_df$emotion))
nemo = length(emos)
emo.docs = rep("", nemo)
for (i in 1:nemo)
{
tmp = tweets_neg[emotion == emos[i]]
emo.docs[i] = paste(tmp, collapse=" ")
}
# remove stopwords
emo.docs = removeWords(emo.docs, stopwords("spanish"))
# create corpus
corpus = Corpus(VectorSource(emo.docs))
tdm = TermDocumentMatrix(corpus)
tdm = as.matrix(tdm)
colnames(tdm) = emos
# comparison word cloud
comparison.cloud(tdm, colors = brewer.pal(nemo, "Dark2"),
                 scale = c(2.5,.5), random.order = FALSE,
                 title.size =1.5)
```

Veamoslo ahora para la polaridad.

```
emos2 = levels(factor(sent_df$polarity))
nemo2 = length(emos2)
```

```
emo.docs2 = rep("", nemo2)
for (i in 1:nemo2)
{
tmp2 = tweets_neg[polarity == emos2[i]]
emo.docs2[i] = paste(tmp2, collapse=" ")
}
# remove stopwords
emo.docs2 = removeWords(emo.docs2, stopwords("spanish"))
# create corpus
corpus2 = Corpus(VectorSource(emo.docs2))
tdm2 = TermDocumentMatrix(corpus2)
tdm2 = as.matrix(tdm2)
colnames(tdm2) = emos2
# comparison word cloud
comparison.cloud(tdm2, colors = brewer.pal(nemo2, "Dark2"),
                 scale = c(2.5,.5), random.order = FALSE,
                 title.size =1.5)
```

Detallemos ahora los distintos métodos de clasificación antes mencionados con los que podemos clasificar tanto la polaridad como los sentimientos de nuestros tweets.

5.1. Algoritmos de clasificación

5.1.1. Clasificador Naives-Bayes

Naive-Bayes es un método de clasificación supervisado y generativo que se basa en el teorema de Bayes y en la premisa de independencia de los atributos para obtener la probabilidad de que un documento pertenezca a una determinada clase como se indica en la ecuación que sigue :

$P(C|D) = P(C_i) \prod P(f_k|C_i)$

Donde f_k son los atributos del documento, C_i es la clase y $P(f_k|C_i)$ es la probabilidad de ocurrencia del atributo en la clase dada. La clase seleccionada por el clasificador sera la que maximice la probabilidad anterior.

Las implementaciones del algoritmo de Na¨ıve Bayes difieren principalmente en la aproximación de $P(f_k|C_i)$ y las técnicas de smoothing utilizadas para el tratamiento de probabilidades bajas o nulas.

En términos simples, un clasificador de Naives-Bayes asume que la presencia o ausencia de una característica particular no está relacionada con la presencia o ausencia de cualquier otra característica, dada la clase variable. Por ejemplo, una fruta puede ser considerada como una manzana si es roja, redonda y de alrededor de 7 cm de diámetro. Un clasificador de Naives-Bayes considera que cada una de estas características contribuye de manera independiente a la probabilidad de que esta fruta sea una manzana, independientemente de la presencia o ausencia de las otras características.

5.1.2. Clasificador de Máxima Entropía

Modelo de Máxima Entropía (MaxEnt) es un método de clasificación discriminativo donde los documentos del conjunto de datos son descriptos a partir de una lista de atributos, siendo cada uno una restricción del modelo. Este método se basa en seleccionar la distribución de probabilidad que satisfaga todas las restricciones del modelo y maximice la entropía. Esto apunta a preservar la incertidumbre tanto como sea posible.

En modelos de máxima entropía la probabilidad de que un documento pertenezca a una clase se define como sigue:

$$P(c|x) = \frac{e^{\sum w_{ci} f_i}}{\sum e^{\sum w_{c'i} f_i}}$$

Donde, c es la clase que se desea evaluar, x es el documento, f_i es cadaatributo, w_{ci} es el peso de ese atributo para la clase c que se está evaluando y $w_{c'i}$ es el peso del atributo en cada una de las posibles clases.

El cálcul de pesos es un problema complejo conocido como convex optimization que busca camximizar la verosimilitud del modelo.

5.1.3. Support Vector Machines

Support Vector Machines es un método supervisado de clasificación binaria en el cual el entrenamiento consiste en encontrar un hiperplano que separe los vectores de atributos que

representan los documentos del conjunto de datos en dos grupos, siendo esta separación la más grande posible. Aquellos vectores que definen los márgenes de la máxima separación entre las clases se conocen como support vectors

Para la predicción de la clase utilizando este modelo se define la ecuación que sigue:

$f(x) = sign(\sum \alpha_i x_i \mathring{u} x + b)$

Siendo, x el vector de atributos del documento a clasificar, α_i cada uno de los pesos que ponderan los vectores de atributos identificados como support features, x_i cada uno de los support features y b el término independiente. Un valor de -1 indicará que el documento pertenece a una clase y un valor de +1 a la otra, lo que representa de qué lado del hiperplano se encuentra x.

Capítulo 6

Redes Multicapas

En este capítulo vamos a profundizar sobre las redes multicapas y como estan relacionadas a las redes sociales, así como una aplicación.

6.1. Terminología y modelo

Las interacciónes sociales en cualquier red social generalmente se representa mediante un grafo G(V,E), donde los nodos que forman el conjunto V representan las personas y las aristas E representan las interacciones entre ellos, como ya vimos en el capítulo 2 donde desarrollamos brevemente el concepto de grafo.

Si bien esta representación ha sido inmensamente útil para el descubrimiento de muchos fenómenos sociales, se centra en una abstracción de capa única de las relaciones humanas. En este capítulo vamos a describir un modelo que representa la multiplexidad de enlaces (interacciones) mediante el apoyo de vínculos a través de diferentes capas.

Podemos dar una definición de red multicapa tal que sea **A** un conjunto de actores y **L** un conjunto de capas, una red multicapa se define como M = (A,L,V,E), donde (V,E) es un grafo y el conjunto de vértices esta contenido en los actores y sus capas.

Término	Sentido
Actor	Una persona, una organización o una entidad que puede tener relaciones con otros actores.
Capa	El mismo actor puede estar presente en diferentes capas, donde cada capa representa un tipo de actores o un tipo de ventaja entre los actores.
Nodo	Un actor específico en una capa específica. Como ejemplo, un nodo puede representar una cuenta de usuario en un OSN (Online Social Network). Se pueden usar varias capas para modelar diferentes OSN, y la misma persona (actor) puede tener cuentas (nodos) en varias OSN (capas).
Borde	Una relación entre dos nodos, por ejemplo, una relación siguiente entre dos nodos que representan dos cuentas de Twitter.

Término	Sentido
Red multicapa	Una red social representada como un conjunto de capas, donde los nodos en las diferentes capas se refieren a un conjunto global de actores y los bordes también pueden conectar nodos en la misma capa o en capas diferentes.
Red de capa única	Una red social representada como una red con solo una capa.

Veamos un ejemplo sobre una red multicapa.

Si nos fijamos en la figura 6.1, observamos cuatro capas que representan dos relaciones fuera de línea (colaboraciones de trabajo y amistades) y dos capas en línea basadas en redes sociales (LinkedIn y Facebook). Estas cuatro capas no son independientes, pero están conectadas a través de los actores comunes indicados en el recuadro del lado derecho. Cada nodo en cada una de las cuatro capas representa uno de estos actores o una cuenta de redes sociales propiedad de uno de ellos. Estos actores definen puentes entre las cuatro capas.

Aunque esta red multicapa es demasiado pequeña para permitirnos llegar a conclusiones significativas sobre las propiedades generales de la red, este ejemplo muestra cómo el análisis conjunto de múltiples capas puede proporcionar conocimiento que no está presente en cada capa cuando las capas se consideran independientemente una de la otra.

Además, las redes multicapa pueden afectar significativamente nuestra comprensión de un sistema social, incluso cuando solo están disponibles los datos de capa única. Un claro ejemplo son las conexiones de amistad de Facebook, que pueden indicar amigos, conocidos, colegas, familiares, etc. Como consecuencia, realizar tareas como identificar comunidades se vuelve muy complejo debido a la gran cantidad de contextos sociales superpuestos, y si algunos datos solo faltan de una capa oculta específica, entonces algunas medidas descriptivas de red pueden subestimarse o sobreestimarse, dependiendo de la capa donde faltan los datos.

En resumen, los avances en el análisis de redes multicapa también conducen a un replanteamiento de la forma en que analizamos las redes de una sola capa.

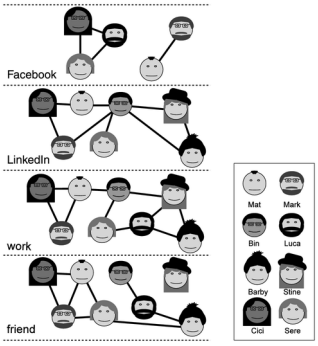


Figura 6.1: Multicapa

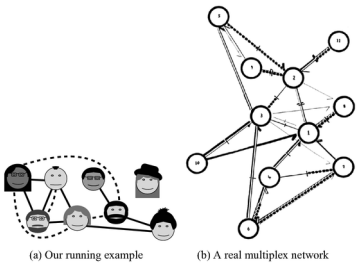


Figura 6.2: Red multiplex

$$
\begin{array}{c}
 \\
Cici \\
Mat \\
Mark \\
Sere \\
Bin \\
Luca \\
Stine \\
Baby
\end{array}
\begin{array}{c}
\begin{array}{cccccccc}
Cici & Mat & Mark & Sere & Bin & Luca & Stine & Baby
\end{array} \\
\left(\begin{array}{cccccccc}
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0
\end{array}\right)
\end{array}
$$

Figura 6.3: Matriz adyacencia

6.2. Modelos relacionados

Las redes sociales multicapa, al ser modelos extremadamente abstractos y flexibles, pueden representar una gran variedad de escenarios del mundo real. Muchos de ellos caen dentro de los tres dominios de problemas: la heterogeneidad de las relaciones, la heterogeneidad de los actores y la correlación de capas. Estos problemas específicos, o problemas con características similares, se han enfrentado muchas veces a lo largo de los años y se han abordado a través de diferentes enfoques. Por lo tanto, es importante reconocer cómo se relacionan estos enfoques y cómo han contribuido al concepto generalizado de la red social multicapa.

6.2.1. Redes multiplex

En una red multiplexada, un conjunto común de actores está conectado a través de múltiples tipos de bordes. En la figura 6.5 se muestra una representación visual donde se ve una red multiplexada, donde dos tipos diferentes de bordes (Facebook y amigo) se indican mediante líneas continuas y discontinuas, respectivamente. Esta forma de representar las redes sociales ha estado en uso durante mucho tiempo.

Una forma típica de representar matemáticamente una red multiplexada, ya utilizada por Bott (1928)), es un conjunto de matrices de adyacencia, uno para cada tipo de borde, con una fila / columna para cada actor y un elemento (i,j) que indique si los actores i y j están conectados por un borde del tipo correspondiente. Una representación de para nuestro ejemplo sería.

Representación matricial de la red multiplex (superior) Facebook y (abajo) de amigos:

6.2.2. Redes multimodo y multinivel

Mientras que las redes multiplex enfatizan diferentes tipos de relaciones dentro de un conjunto de nodos, una forma diferente de extender las redes de una sola capa es considerar

	Cici	*Mat*	*Mark*	*Sere*	*Bin*	*Luca*	*Stine*	*Baby*
Cici	0	0	0	1	0	1	0	0
Mat	0	0	1	0	0	0	0	0
Mark	0	1	0	0	0	0	0	0
Sere	1	0	0	0	0	1	0	0
Bin	0	0	0	0	0	0	0	0
Luca	1	0	0	1	0	0	0	0
Stine	0	0	0	0	0	0	0	0
Baby	0	0	0	0	0	0	0	0

Figura 6.4: Matriz adyacencia

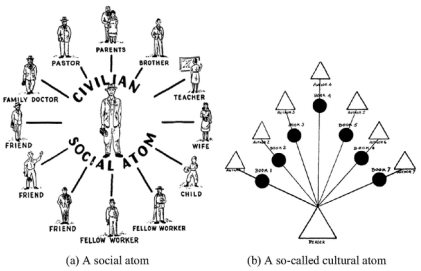


Figura 6.5: Red multimodo

múltiples tipos de actores dentro de la misma red. Esta extensión ha sido utilizada en SNA desde sus orígenes.

En esta figura podemos ver los componentes básicos de una red (llamados átomos)) se componen de diferentes tipos de individuos (átomo social) y personas y recursos, específicamente libros (átomo cultural).

La aplicación más común de este enfoque tiene como objetivo modelar casos con dos tipos de actores, por ejemplo, autores que escriben artículos o usuarios que publican mensajes. Una red donde se pueden identificar dos tipos de actores es una de dos modos (también llamada bipartita, o en algunos casos afiliación) red, donde hay dos tipos de nodos y bordes conectan pares de nodos de diferentes tipos.

Este tipo de red se ha estudiado durante mucho tiempo en antropología (Davis et al., 1941) y puede usarse para modelar diferentes tipos de escenarios, por ejemplo, autores y conferencias / revistas, así como miembros de varias juntas u oficinas.

6.2.3. Redes Temporales

Los estudios longitudinales, es decir, estudiar la evolución de algunas variables a través de observaciones repetidas en diferentes momentos, constituyen una importante herramienta de investigación en las ciencias sociales y la investigación médica. Aunque ejecutar un estudio longitudinal a menudo requiere muchos recursos, más de lo que se necesita para recopilar una instantánea de los datos correspondientes a una única hora específica, puede responder preguntas que no pueden abordarse mirando una sola instantánea. Este tipo de estudio ha sido una herramienta común en el dominio de SNA, y los datos recopilados de las encuestas de redes longitudinales están muy relacionados con los modelos multicapa: podemos pensar en diferentes capas como representando diferentes instantáneas de la misma red social en evolución en el tiempo.

Como diferentes puntos en el tiempo se pueden considerar como capas diferentes, en teoría, se puede analizar una red social observada en diferentes momentos utilizando métodos multicapa.

6.3. Conjunto de datos

Las redes sociales multicapa aparecen en diferentes contextos, donde los datos se caracterizan por diferentes tamaños (desde un puñado de nodos hasta varios millones), diferentes naturalezas (p. Ej., En línea, fuera de línea, híbridos) y semántica de capa diferente (contacto, comunicación, tiempo, contexto, etc.).

Veamos algunos ejemplos donde las redes de múltiples capas se puedan utilizar como una herramienta para estudiar los sistemas sociales e indicar una lista de conjuntos de datos disponibles que se pueden utilizar con fines de capacitación o investigación.

Nombre	Descripción
Familias florentinas	Estos datos, recopilados por John Padgett a partir de documentos históricos, describen las relaciones entre 16 familias políticamente prominentes en la ciudad de Florencia alrededor del año 1430: vínculos comerciales (específicamente, vínculos financieros registrados como préstamos, créditos y asociaciones conjuntas) y alianzas matrimoniales. Dos facciones aparecen en los datos, con familias fuertemente relacionadas con los Medici o con la familia Strozzi, lo que hace que los datos sean útiles para probar los métodos de detección de la comunidad.
El monasterio de Sampson	Se le pidió a un grupo de ocho monjes que especificaran sus tres opciones principales en cuatro pares de relaciones positivas / negativas: estima y desestimación, gusto y aversión, influencia positiva e influencia negativa, elogio y culpa. El límite de las tres preferencias principales impuestas en la encuesta puede sesgar los resultados de las medidas basadas en el grado, ya que el grado de cada actor es 3 por la construcción de los datos.
Cableado del Banco	Un actor específico en una capa específica. Como ejemplo, un nodo puede representar una cuenta de usuario en un OSN. Se pueden usar varias capas para modelar diferentes OSN, y la misma persona (actor) puede tener cuentas (nodos) en varias OSN (capas).
Terrorista	Un conjunto de datos puestos a disposición por Roberts y Everton (2011) y se extrajo principalmente de los datos recopilados por International Crisis Group, con relaciones entre 78 terroristas indonesios (confianza mutua, operaciones comunes, comunicaciones intercambiadas y relaciones comerciales).
Twitter-XF	Este conjunto de datos contiene respuestas y retweets de Twitter producidos por un pequeño grupo de usuarios (5137) que interactúan en línea durante el episodio final de un popular programa de televisión (XFactor Italia). Rossi y Magnani (2012) han utilizado estos datos para analizar el efecto de las conversaciones en línea sobre las conexiones en las redes de contacto, es decir, cómo los usuarios comenzaron a seguir a los demás después de haber interactuado sobre temas de interés común.

Más recientemente, con el advenimiento de las redes sociales en línea, los datos adicionales están disponibles. Una forma de recopilar redes multicapa desde OSN consiste en recuperar información sobre las cuentas del mismo conjunto de usuarios en diferentes plataformas. Esto se ha hecho a menudo extrayendo datos de los agregadores de redes sociales. Estos meta-servicios pueden ser sitios web ad hoc o aplicaciones independientes y permiten a las personas administrar de manera unificada sus múltiples identidades en línea.

Aunque los agregadores de redes sociales pueden representar una solución viable al problema de mapeo de usuarios a través de múltiples OSN, el acceso a estos datos agregados no es fácil, e incluso cuando son anónimos, pueden plantear serias preocupaciones éticas y de privacidad. Sin embargo, estos conjuntos de datos tienen algunas características peculiares que los distinguen de los casos anteriores y los hacen potencialmente valiosos para la investigación. En primer lugar, exponen explícitamente múltiples identidades en línea de los mismos individuos: diferentes cuentas en diferentes redes sociales pueden considerarse como entidades diferentes, incluso si son propiedad del mismo actor, porque se pueden usar para presentar diferentes aspectos de la persona y se pueden usar estratégicamente para llegar a diferentes audiencias. Los nodos correspondientes al mismo actor en diferentes capas no representan directamente al actor, sino solo cuentas de usuario, lo que hace que estos nodos sean claramente distintos entre sí. Podemos decir que estos datos se ajustan claramente a un modelo de varias capas,actor y el mismo actor operando en diferentes tipos de relaciones.

Aunque las redes sociales multicapa de múltiples medios sociales en línea son difíciles de recopilar debido al complejo problema de mapear usuarios a través de diferentes servicios, las redes multicapa se pueden utilizar para estudiar otros tipos de datos en línea. Las plataformas de redes sociales (de Facebook a Twitter y YouTube) a menudo ofrecen la oportunidad de comunicarse fuera de la red topológica definida por las conexiones de los usuarios. Las conversaciones basadas en hashtag de Twitter, así como grupos de Facebook o comentarios de YouTube en un video, representan ejemplos de cómo, dentro de estos servicios, los usuarios pueden comunicarse independientemente de la estructura definida por sus conexiones específicas, por ejemplo, amistad en Facebook y seguimiento en Twitter . En estos casos, la red topológica (por ejemplo, la amistad) se puede modelar como una capa de una red multicapa y cualquier otra acción comunicativa (por ejemplo, comentar en la misma página pública) puede ser representado por capas adicionales.

6.4. Recopilación y preprocesamiento de datos

El primer paso en el análisis de una red empírica multicapa es obtener y preparar los datos. Aunque esto puede parecer obvio, es fácil subestimar el impacto de la fase de recopilación y preproceso de datos en un proceso de minería de datos, o la proporción de tiempo dedicado a estas actividades y la posterior interpretación de los resultados si se compara con el tiempo necesario para ejecutar los algoritmos de minería de datos seleccionados. De hecho, esta es a menudo la parte más importante y requiere mucho tiempo del proceso y tiene un gran impacto en los resultados del análisis

Una vez que se han recopilado los datos, es posible que no estén listos para su análisis.Si consideramos una red social multicapa, podemos ver cada capa como una característica que describe a los actores en el modelo que se centra en las relaciones de los actores en lugar de sus características personales. De forma similar a como necesitamos eliminar o

transformar funciones cuando analizamos datos que no son de red, antes de analizar una red multicapa, es posible que deseemos mantener solo algunas capas relevantes entre las que se recopilaron inicialmente o combinar varias capas en una nueva que proporcione una visión diferente sobre los datos.

Pasamos por alto los problemas de recopilación de datos ??

6.5. Simplificación de red

Una forma de abordar el problema de especificación de límite vertical es recolectar tantas capas como sea posible y reducirlas a un número más pequeño y más enfocado en un momento posterior. Una forma básica de reducir el número de capas es (parcialmente) eliminar algunas de ellas o fusionar algunas en una sola capa. En última instancia, podemos reducir cualquier red multicapa a una red de una sola capa y analizarla utilizando métodos tradicionales pero corremos el riesgo de perder información relevante, esto todavía se puede considerar una forma para realizar una primera exploración de los datos. Este enfoque se llama **aplanamiento** o **proyección**, dependiendo de si las capas fusionadas contienen los mismos actores, por ejemplo, redes multiplex, o actores de diferentes tipos, por ejemplo, redes de afiliación.

6.5.1. Aplanamiento y proyección

Un enfoque básico para tratar con redes de múltiples capas es reconstruir una red social (ponderada) de capa única para que los métodos existentes, como la detección de comunidades, se puedan aplicar directamente. Se pueden usar dos enfoques generales de este tipo, dependiendo de la naturaleza de los nodos. Si los nodos en las diferentes capas corresponden a un conjunto común de actores, normalmente hablamos de aplanamiento, es decir, el proceso de fusionar todos los nodos correspondientes al mismo actor en un solo nodo. Cuando tenemos múltiples tipos de nodos, una operación común consiste en proyectar la red solo en un tipo de nodo, descartando los otros.

- **Aplastamiento**

Un proceso básico de aplanamiento consiste en crear una capa con un nodo para cada actor y un borde entre dos nodos si existe un borde entre dos nodos correspondiente a esos actores en algún lugar de la red multicapa.

Aplanamiento básico Un aplanamiento básico (no ponderado) de una red multicapa $G = (A, L, V, E)$ es un gráfico (V_f, E_f) donde $V_f = \{\ a|(a, l) \in V\ \}$ y $E_M = \{\ (a_i, a_j)|\ \{\ (a_i, l_q), (a_j, l_r)\ \} \in E\ \}$.

Una variación simple del aplanamiento básico consiste en agregar un peso a cada borde en la red aplanada proporcional al número o aristas entre los actores correspondientes a esos nodos. Un enfoque más general consiste en asignar un peso $\theta_{q,r}$ a cada par de capas (l_q, l_r) , de modo que la red de una capa resultante se puede expresar como una combinación lineal de la red multicapa original.

Aplanamiento ponderado

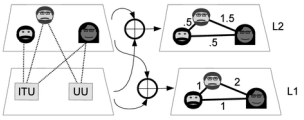


Figura 6.6: Ejemplo proyección

Sea $G = (A, L, V, E)$ una red multicapas, con una matriz θ de dimensiones $|L|x|L|$ donde $\theta_{q,r}$ indica el peso asignados a los nodos de aristas l_q y l_r. Un aplanamiento ponderado de G es un gráfico ponderado (V_f, E_f, w), donde (V_f, E_f) es un aplanamiento básico de G y $w(a_i, a_j) = \sum_{[(a_i,l_q),a_j,l_r]\in E} \theta_{q,r}$.

- **Proyección**

La proyección de red es un enfoque tradicional utilizado para simplificar las redes de dos nodos, que se pueden modelar como redes multicapa donde cada tipo de nodo (por ejemplo, A o B) está representado por una capa.

El enfoque más directo para la proyección de red consiste en eliminar los nodos de tipo B y agregar una ventaja entre cualquier par de nodos de tipo A conectados originalmente al mismo nodo de tipo B. En este enfoque, los nodos de tipo A se conectan por un borde ponderado con peso w definido como $w(i, j) = \sum_p 1$, donde P indica los nodos de tipo B conectados a ambos i y j.

Este es el caso para la capa L1 en la Figura 4.2: Luca, Cici y Mark formarían una camarilla no ponderada bajo una proyección simple, pero Mark y Cici están conectados a dos nodos de tipo B, ITU y UU. Por lo tanto, el peso del borde entre ellos es el doble del peso entre Cici y Luca, que solo comparten un nodo tipo B (ITU).

6.6. Visualización de redes multicapas

La visualización tiene dos objetivos principales: puede usarse como una herramienta exploratoria para obtener una comprensión básica de la estructura de red y de algunas propiedades de red, por ejemplo, la presencia de comunidades, o puede usarse para presentar los resultados de un análisis preexistente de una manera fácilmente accesible.

Vamos a presentar diferentes enfoques de visualización y los ejemplificamos en nuestro ejemplo de ejecución para resaltar el tipo de información que pueden extraer de una red multicapa real.

6.6.1. Cuatro enfoques principales

Hay dos aspectos principales en los que la representación visual de las redes sociales puede jugar un papel importante: la visualización de la estructura de la red, es decir, la disposición espacial de sus nodos y bordes, y una representación visual de sus propiedades numéricas, correspondientes a las diversas métricas.

Estos dos tipos de objetivos también se pueden fusionar en unavisualización única, ya sea mediante el aumento de un gráfico con información sobre las métricas o el uso de las métricas para eliminar la información irrelevante del gráfico.

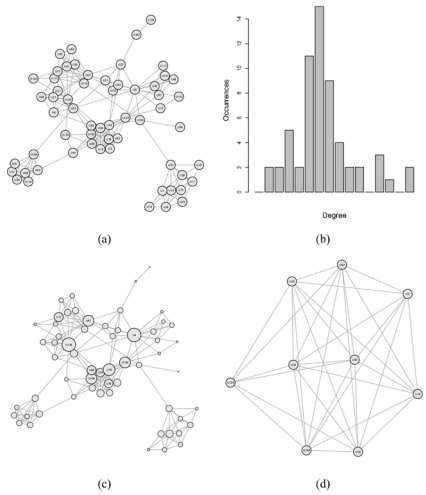


Figura 6.7: Ejemplo visualización

Cuatro visualizaciones típicas de una red única: (a) centrarse en la estructura, (b) centrarse en las métricas, (c) visualización aumentada, y (d) visualización simplificada (6 núcleos de la red).

Aunque probablemente sea la más atractiva, la visualización de la estructura de la red no es la única forma de dar sentido a una red: a veces puede ser más conveniente confiar en técnicas de visualización de datos más tradicionales. Las dos opciones mencionadas anteriormente (estructura y medidas) también se pueden combinar, y los sociogramas se pueden enriquecer con información sobre las métricas.

La forma opuesta de combinar las dos estrategias básicas de visualización es usar métricas de red para filtrar o simplificar la red. La idea subyacente es que los bordes de la red pueden llevar una gran cantidad de información, y visualizar todo puede llevar a que el ruido oculte el conocimiento relevante.

6.6.2. Visualización de métricas

Hay dos estrategias principales para lograr este objetivo: una forma de hacerlo es representar explícitamente a cada actor y mostrar cómo cambian sus valores para una medida dada sobre las diferentes capas utilizando una visualización de coordenadas paralelas, y una segunda manera es trazar directamente un medida de similitud de capa usando una visualización de matriz de bloques para la correlación de capas.

6.6.2.1. Coordenadas paralelas

Las coordenadas paralelas son herramientas típicas para visualizar datos multidimensionales.

Visualización de diferentes medidas enfatizando las relaciones entre las diferentes capas. (a) El valor de relevancia para cada actor en cada capa del conjunto de datos AUCS. (b) Número de vecinos para cada actor en cada capa. (c) coeficientes de correlación de Jaccard para cada par de capas.

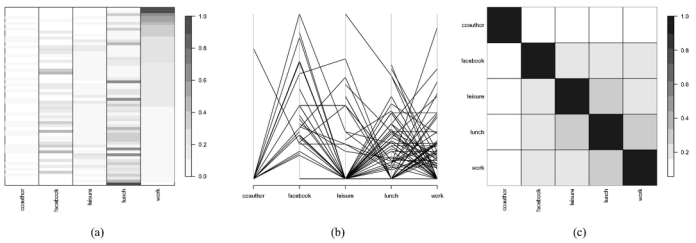


Figura 6.8: Coordenadas paralelas

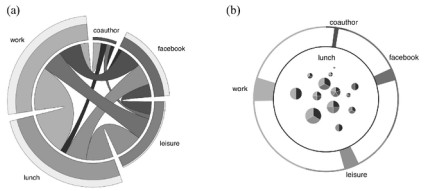


Figura 6.9: Gráfico de concurrencia

6.6.2.2. Gráfico de cocurrencia de bordes

Otra forma de visualizar las relaciones entre diferentes capas es usar las representaciones en la siguiente figura . La figura (a) muestra el porcentaje de bordes coocurrencia en cada par de capas, además también proporciona una intuición sobre el tamaño de las capas.

Un enfoque similar nos permite mover el foco de la visualización a una capa específica, por ejemplo la figura (b).En este caso, además de las coincidencias de borde entre la capa de almuerzo y las otras capas, podemos ver una distribución de patrones de múltiplex, que se indican dentro del círculo. La idea detrás de estos pasteles es que, además de visualizar relaciones por parejas, también podemos mostrar la frecuencia de patrones múltiplex más complejos que involucran más de dos capas.

Concurrencia de bordes en las capas: (a) vista global; (b) vista local.

6.6.3. Visualización de la estructura

Cuando pasamos de una perspectiva de red única a una de múltiples capas, la complejidad adicional en las relaciones agregadas hace que el cálculo e incluso la definición de un diseño apropiado sean más desafiantes. En la siguiente figura, podemos ver nuevamente la capa de almuerzo de la red AUCS a) y toda la red con los cuatro tipos adicionales de relaciones b).

Comparando las dos visualizaciones, podemos ver cómo la estructura clara de la red de almuerzos se vuelve más borrosa y confusa si tenemos en cuenta las conexiones de todas las capas. Como ejemplo, hemos resaltado un clúster claramente visible en la capa del lado izquierdo (nodos negros). Los mismos nodos también están marcados en negro en el gráfico aplanado de la derecha, y podemos ver que el grupo se ha visto parcialmente atraído hacia el centro de la figura y que algunos de sus nodos ahora están más conectados a otros nodos fuera del clúster.

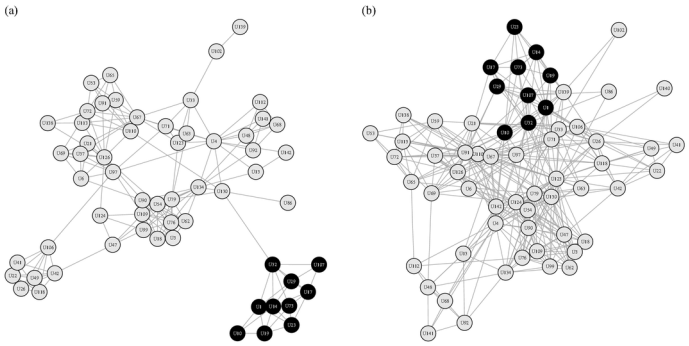


Figura 6.10: Gráfico de estructuras

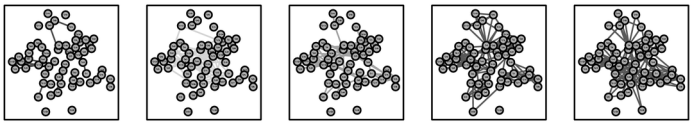


Figura 6.11: Gráfico en rodajas

6.6.3.1. Corte de capas

Una posible solución a este problema se obtiene mediante la idea de cortar capas. Dos visualizaciones alternativas se muestran seguidamente. Ambos métodos cortan la red en sus capas de composición. Para simplificar una comparación entre las capas, en la Figura, los nodos se han colocado utilizando el mismo diseño en cada sector.

6.6.4. Visualización de red simplificada

Una forma opuesta de visualizar una red es usar medidas analíticas para simplificarla en lugar de aumentar los sociogramas ya complicados con aún más información.

Esta figura nos muestra el sociograma de fusión local definido según la medida de relevancia con un umbral de .6, representado como un multigrafo. Definir un sociograma simplificado localmente basado en este valor significa mostrar un borde entre dos nodos de la red multicapa solo cuando una capa específica es más relevante que .6 para ambos actores. Este tipo de visualización considera el valor de relevancia para la díada y no para el actor individual, por lo tanto, los bordes se representan solo si pertenecen a una capa

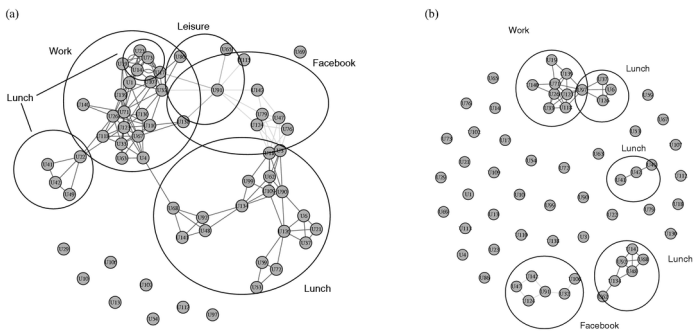


Figura 6.12: Visualización simplificado

que es relevante para ambos actores de la díada; esta es la razón por la que esto se ha llamado simplificación local.

Si nos fijamos en la parte a) podemos ver que los bordes considerados relevantes por sus puntos finales no están uniformemente distribuidos en la red, sino que tienden a agregarse en estructuras cohesivas correspondientes a grupos de actores que usan la red de diferentes maneras. A partir de esta figura, debe quedar claro cómo este método conserva solo porciones específicas (y locales) de cada capa.

En el caso b) nos muestra otra red simplificada basada en relevancia exclusiva. Mientras que en una red multicapa, los nodos y las conexiones a menudo se replican a través de varias capas, la relevancia exclusiva mide aquellas conexiones que están disponibles solo en una sola capa. Como en el ejemplo anterior, solo se visualizan los bordes que pertenecen a una capa con una relevancia exclusiva más alta que un umbral dado (.3 en nuestro ejemplo) para ambos actores de la díada.

6.7. Patrones de borde

Una de las tareas más populares de minería de datos para redes sociales es la prediccion de enlaces : dado el estado actual de una red social, ¿qué bordes es probable que aparezcan en un momento posterior? Este problema se definió originalmente para redes de una sola capa. Hay una suposición subyacente sobre las dinámicas sociales que conducen a la aparición de nuevas aristas, por ejemplo, el hecho de que los actores con muchas aristas tienen una mayor probabilidad de recibir incluso más aristas, o el hecho de que los actores con muchos puntos comunes con los vecinos tienen una mayor probabilidad de convertirse en vecinos.

Sin embargo, con los modelos de evolución de las redes sociales, el objetivo suele ser la generación de redes sociales sintéticas que tengan algunas propiedades globales deseadas, por ejemplo, un determinado coeficiente de agrupamiento o distribución de grados.

Aunque la predicción de borde para redes sociales multicapa se puede ver como una extensión directa de métodos existentes para redes de una sola capa, un tipo específico de conocimiento que solo tiene sentido cuando tenemos capas múltiples es la existencia de correlaciones de capa.

6.7.1. Predicción del borde

Predicción de borde, según lo definido por Liben-Nowell y Kleinberg, aborda la siguiente pregunta: qué bordes van a estar presentes en el momento t_1 dado los bordes existentes en un momento anterior t_0

En esta imagen vemos la capa de trabajo de nuestro ejemplo de ejecución en dos momentos diferentes en el tiempo: en la figura, aparecen tres nuevos bordes y uno desaparece a la vez . El objetivo es mirar el tiempo y producir una lista clasificada de bordes que se cree que aparecen o desaparecen pronto.

El problema de la predicción de bordes puede establecerse como un problema de clasificación. Primero elegimos un conjunto de características para describir cada par

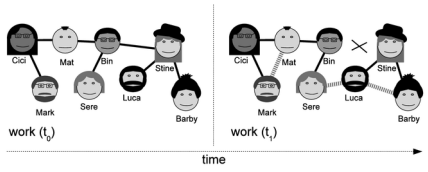


Figura 6.13: Predicción de bordes

de nodos en la red. Para que la predicción sea exitosa, estas características deben estar asociadas a la tendencia de los actores a conectarse o desconectarse. Luego, también agregamos un atributo de clase para cada par de nodos, lo que indica su estado de conectividad en el tiempo t_1.

La siguiente tabla nos muestra cómo una parte de la capa de trabajo en nuestro ejemplo de ejecución se puede formatear como un problema de predicción de bordes usando dos características simples: el producto de los dos grados de nodo y el número de vecinos comunes. En este punto, podemos aplicar cualquier algoritmo de clasificación para aprender automáticamente la relación entre los atributos de la característica y el atributo de la clase. En la práctica, podemos aprender una función que para cada par de nodostoma todas sus características como entrada y produce una respuesta sí / no, indicando si se cree que esos actores se conectarán / desconectarán o, más típicamente, un puntaje que indique la probabilidad de que esto suceda, lo cual puede usarse para clasificar a todos pares de nodos según nuestra creencia de que un nuevo borde aparecerá o desaparecerá entre ellos.

Par	Producto del grado	comm. relinchar.	Borde (t_0)	**Borde (t_1)**
Mark,Mat	6	1	no	**sí**
Mark,Bin	6	1	no	**no**
Mark,Serena	4	0	no	**no**
...	...	...	...	...

Esta tabla puede verse ligeramente diferente dependiendo de la formulación específica del problema de predicción de bordes.

Por ejemplo, podríamos tener un rastro de tiempos diferentes en el pasado cuando algunos bordes habían aparecido y desaparecido, por lo tanto, el borde de la columna (t_0) podría ser reemplazado por un conjunto de veces cuando el borde estaba presente. Se ha demostrado que esto mejora la precisión de las predicciones, explotando el hecho de que los bordes que a menudo están presentes o han estado presentes recientemente tienen una mayor probabilidad de estar presentes también en el futuro cercano. La mayoría de los trabajos existentes se han centrado en la predicción de nuevos bordes y no en la predicción de la desaparición del borde. En este caso, el borde de la columna (t_0) no se usaría, ya que solo mantendríamos en la tabla pares de actores que actualmente no están conectados; es decir, el borde ($t_0 =$)no.

6.8. Ejemplos R

En este apartado vamos a ver que podemos realizar en nuestro software R.

Instalamos la librería "multinet" con la que trabajaremos ya que posee tanto redes multinet ya predefinidas, usadas en las explicaciones anteriores, como funciones para su tratamiento. Además, vamos a cargar las redes multicapas que vamos a usar posteriormente.

```
library(multinet)
aucs <- ml.aucs()
florentine <- ml.florentine()
toy <- ml.toy()
```

Tomemos por ejemplo la red multicapa "Toy" que nos muestra el ejemplo usado con actores como Mark, Mat o Bin y con las capas "LinkedIn", "Work" o "Facebook". Veamos si es cierto que estamos tomando dicha red, para ello vamos a obtener los nombres tanto de las capas como de los actores. Además vamos a observar los nodos y las relaciones entre nuestros actores y capas, todo esto de manera descriptiva.

```
# Resumen de la red
file <- tempfile("aucs.mpx")
write.ml(toy,file)
read.ml(file,"Toy")
```

```
## Multilayer Network ("Toy": 4 layers, 8 actors, 27 nodes, 30 edges)
```

```
# Capas
layers.ml(toy)
```

```
## [1] "LinkedIn" "Work"     "Facebook" "Friend"
```

```
# Actores
actores <- actors.ml(toy, layers=character(0))
actores
```

```
## [1] "Cici"   "Mat"    "Serena" "Bin"    "Mark"   "Barby"  "Stine"  "Luca"
```

```
# Nodos
nodos <- nodes.ml(toy, layers=character(0))
colnames(nodos) <- c("Actor","Capa")
nodos
```

```
##     Actor     Capa
## 1    Luca     Work
## 2  Serena     Work
## 3   Stine     Work
## 4   Barby     Work
## 5     Mat   Friend
## 6    Mark Facebook
## 7    Cici   Friend
## 8  Serena Facebook
## 9    Luca   Friend
## 10   Mark   Friend
```

```
## 11 Serena   Friend
## 12  Barby   Friend
## 13    Bin   Friend
## 14    Mat LinkedIn
## 15   Mark LinkedIn
## 16   Cici LinkedIn
## 17    Bin LinkedIn
## 18   Cici     Work
## 19 Serena LinkedIn
## 20  Barby LinkedIn
## 21  Stine LinkedIn
## 22   Luca Facebook
## 23    Mat     Work
## 24    Mat Facebook
## 25    Bin     Work
## 26   Cici Facebook
## 27   Mark     Work
```

```
# Aristas
aristas<- edges.ml(toy, layers1=character(0),
                   layers2=character(0))
colnames(aristas) <- c("Actor salida","Capa salida",
                       "Actor llegada","Capa llegada",
                       "Dirección")
aristas
```

```
##    Actor salida Capa salida Actor llegada Capa llegada Dirección
## 1          Cici    LinkedIn           Mat     LinkedIn         0
## 2          Cici    LinkedIn          Mark     LinkedIn         0
## 3          Mark    LinkedIn           Bin     LinkedIn         0
## 4           Bin    LinkedIn         Stine     LinkedIn         0
## 5         Stine    LinkedIn         Barby     LinkedIn         0
## 6           Mat    LinkedIn           Bin     LinkedIn         0
## 7           Bin    LinkedIn        Serena     LinkedIn         0
## 8           Bin    LinkedIn         Barby     LinkedIn         0
## 9          Cici        Work          Mark         Work         0
## 10        Stine        Work         Barby         Work         0
## 11          Bin        Work         Stine         Work         0
## 12        Luca         Work         Stine         Work         0
## 13          Mat        Work           Bin         Work         0
## 14          Mat        Work          Mark         Work         0
## 15         Cici        Work           Mat         Work         0
## 16         Luca        Work         Barby         Work         0
## 17       Serena        Work           Bin         Work         0
## 18       Serena        Work          Luca         Work         0
## 19         Cici    Facebook          Luca     Facebook         0
## 20         Mark    Facebook           Mat     Facebook         0
## 21         Cici    Facebook        Serena     Facebook         0
## 22         Luca    Facebook        Serena     Facebook         0
```

```
## 23          Bin      Friend          Luca      Friend         0
## 24        Cici       Friend           Mat      Friend         0
## 25        Mark       Friend        Serena      Friend         0
## 26        Luca       Friend         Barby      Friend         0
## 27        Cici       Friend          Mark      Friend         0
## 28      Serena       Friend         Barby      Friend         0
## 29         Mat       Friend          Mark      Friend         0
## 30         Mat       Friend        Serena      Friend         0
```

```
# Número de elementos
numero <- c(num.layers.ml(toy),
            num.actors.ml(toy, layers=character(0)),
            num.nodes.ml(toy, layers=character(0))
,num.edges.ml(toy, layers1=character(0), layers2=character(0))
)
nombres<- c("Actores","Capas","Nodos","Aristas")
as.data.frame(numero,row.names = nombres)
```

```
##         numero
## Actores      4
## Capas        8
## Nodos       27
## Aristas     30
```

Realicemos ahora unos cortes con respecto a las capas, es decir, imaginemos que queremos restringir únicamente a la capa de "Facebook" y realizar el mismo estudio anterior.

```
# Actores
actores_facebook <- actors.ml(toy,"Facebook")
actores_facebook
```

```
## [1] "Mark"   "Serena" "Luca"   "Mat"    "Cici"
```

```
# Nodos
nodos_facebook <- nodes.ml(toy,"Facebook")
colnames(nodos_facebook) <- c("Actor","Capa")
nodos_facebook
```

```
##    Actor     Capa
## 1   Mark Facebook
## 2 Serena Facebook
## 3   Luca Facebook
## 4    Mat Facebook
## 5   Cici Facebook
```

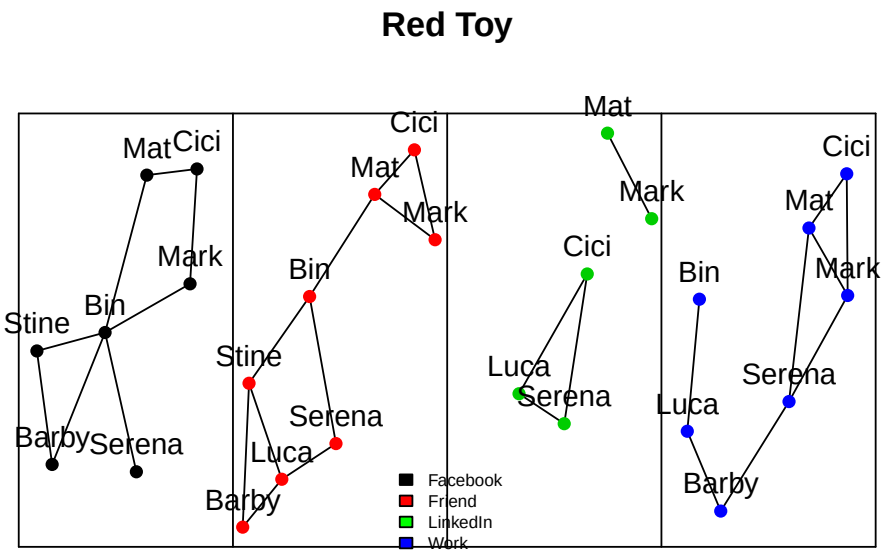
```
# Aristas
aristas_facebook<- edges.ml(toy,"Facebook" )
colnames(aristas_facebook) <- c("Actor salida",
                                "Capa salida",
                                "Actor llegada",
                                "Capa llegada",
```

```
                                  "Dirección")
aristas_facebook
```

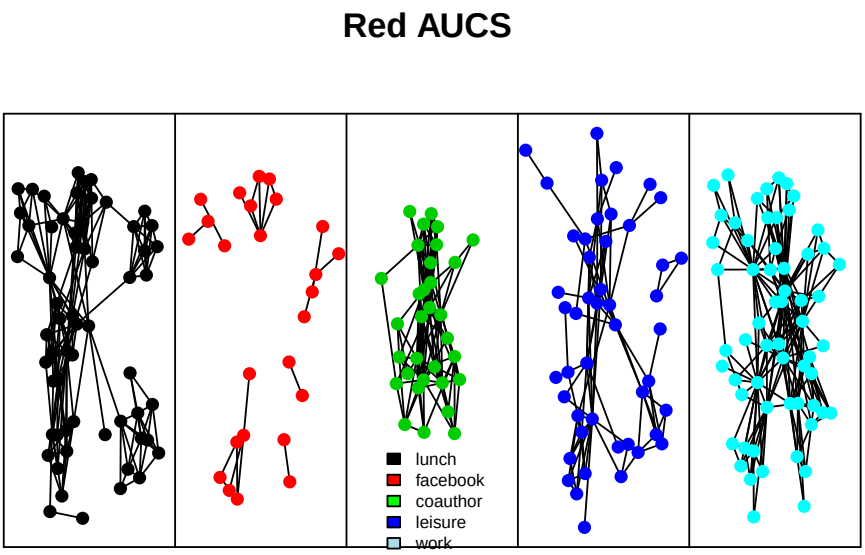
```
##   Actor salida Capa salida Actor llegada Capa llegada Dirección
## 1         Cici     Facebook          Luca     Facebook         0
## 2         Mark     Facebook           Mat     Facebook         0
## 3         Cici     Facebook        Serena     Facebook         0
## 4         Luca     Facebook        Serena     Facebook         0
```

Pasemos a la visualización gráfica de nuestros datos.

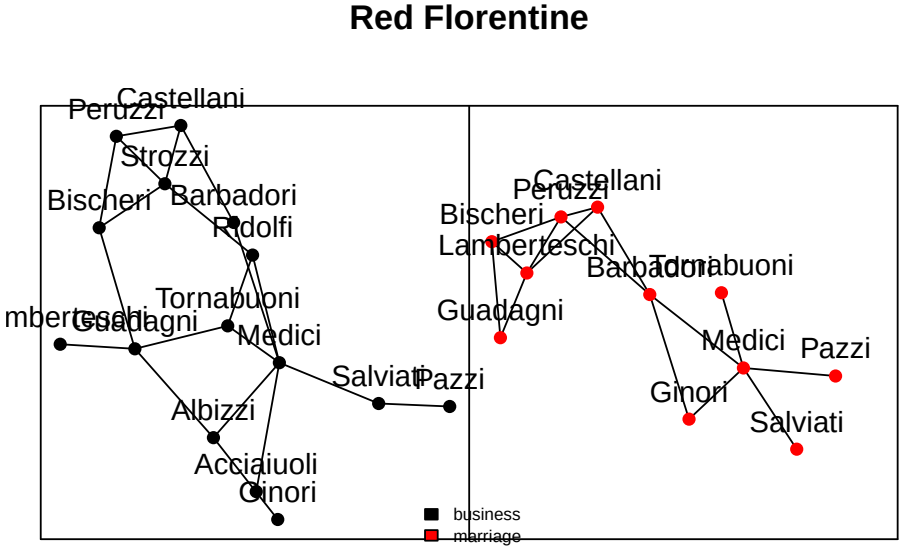
```
plot(toy)
title("Red Toy")
legend( 'bottom',c("Facebook", "Friend","LinkedIn",
                   "Work"), cex=0.6,
        bty="n", fill=c("Black", "Red","Green","Blue"))
```




```
plot(aucs,vertex.labels=NA)
title("Red AUCS ")
legend("bottom", c("lunch"   ,  "facebook" ,"coauthor",
                   "leisure" , "work"  ), cex=0.6,
bty="n", fill=c("Black", "Red","Green","Blue","Lightblue"))
```



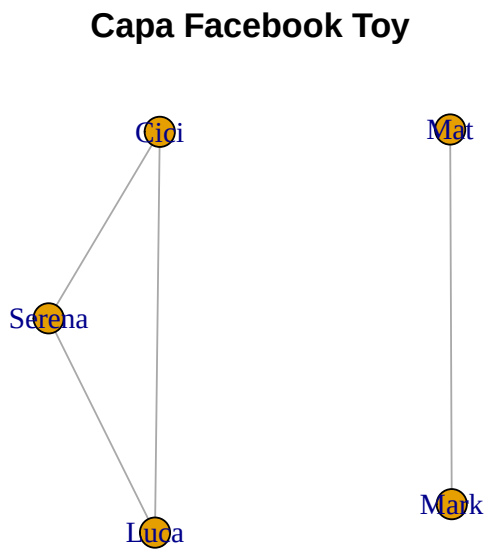

```
plot(florentine)
title("Red Florentine")
legend("bottom", c( "business" ,"marriage"), cex=0.6,
bty="n", fill=c("Black", "Red"))
```



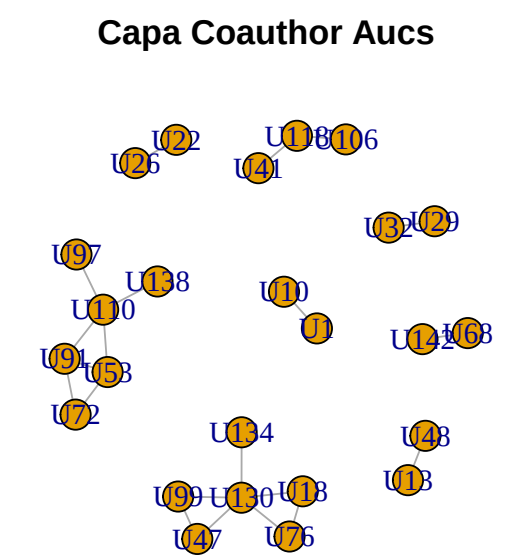
Procedamos ver de forma mas interna las distintas capas, por ejemplo tomemos un

unico grafo correspondiente a la capa “Facebook”

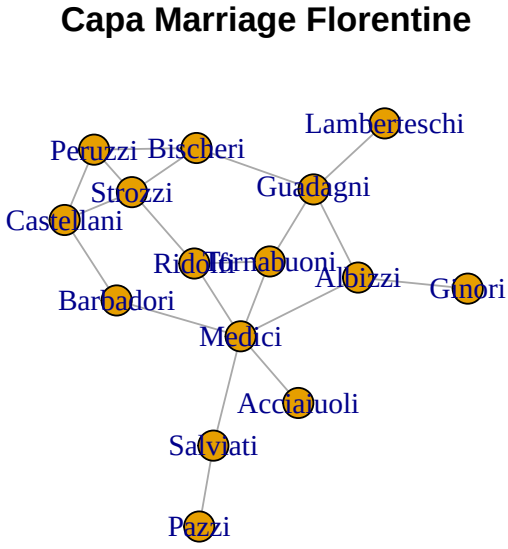
```
facebook_toy <- as.igraph(toy, c("Facebook"))
plot(facebook_toy)
title("Capa Facebook Toy")
```




```
facebook_aucs <- as.igraph(aucs, c("coauthor"))
plot(facebook_aucs)
title("Capa Coauthor Aucs")
```




```
facebook_florentine <- as.igraph(florentine, c("marriage"))
plot(facebook_florentine)
title("Capa Marriage Florentine")
```

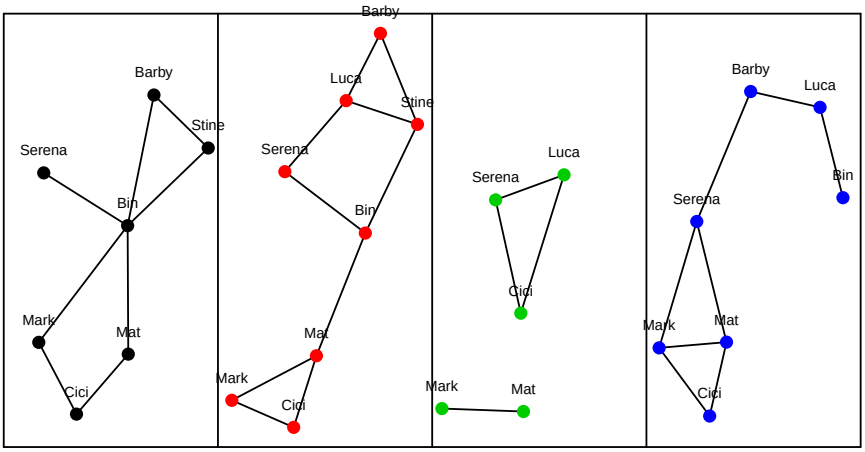


Representemos ahora por ejemplo las distintas comunidades que podemos encontrar, dependiendo de los métodos que tomemos.

```
c1<- abacus.ml(toy)
```

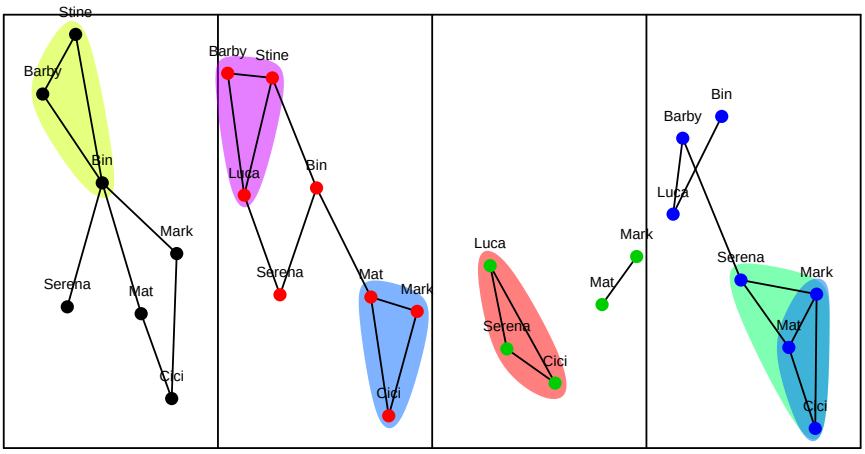
```
## Warning: could not run external library: File not found: Cannot open input tmp fil
## Returning empty community set.
```

```
c2<- clique.percolation.ml(toy)
c3<- glouvain.ml(toy)
c4<- lart.ml(toy)


plot(toy,vertex.labels.cex=.5,com=c1)
title("Abacus")
```



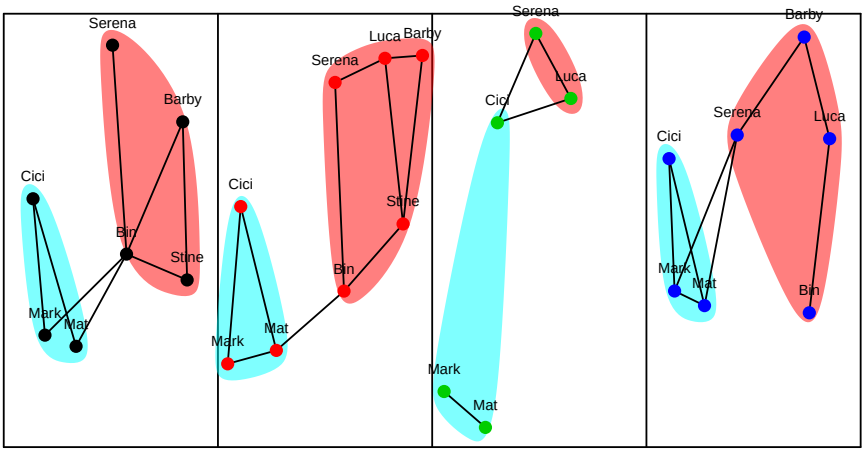

```
plot(toy,vertex.labels.cex=.5,com=c2)
title("Clique.percolation")
```

Clique.percolation

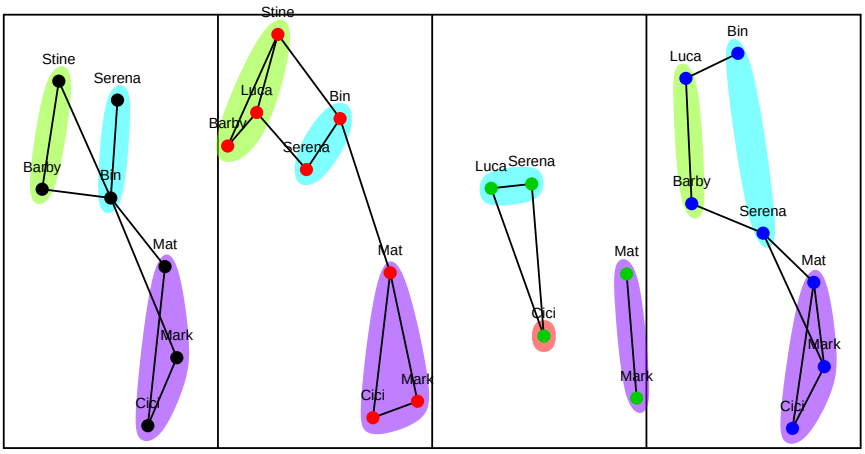



```
plot(toy,vertex.labels.cex=.5,com=c3)
title("glouvain")
```

glouvain




```
plot(toy,vertex.labels.cex=.5,com=c4)
title("lart")
```



Para finalizar podemos comentar como las matemáticas se encuentran tan sumergidas en el mundo del análisis de redes sociales, no solo por las técnicas que podemos aplicar sino por toda la información que podemos obtener de estas.

El mundo de las redes sociales cada vez está mas extendido y es por ello que, tanto las matemáticas como la estadística, también se estan desarrollando en torno a ellas, descubriendo cada dia nuevas técnicas y recursos a partir de los que obtener información, que hoy en día es algo que resulta muy valioso.

Bibliografía

[1] BBVAOPEN4U "Cómo usar la api de twitter en tu negocio". Disponible en https://bbvaopen4u.com/es/actualidad/como-usar-la-api-de-twitter-en-tu-negocio.

[2] Calvo, D. "Obtener datos de twitter con r, usando su api". Disponible en http://www.diegocalvo.es/obtener-datos-de-twitter-con-r-usando-su-api/.

[3] codeand.gradients "Twitter word cloud with r". Disponible en https://codeand.gradients.ml/2013/08/25/twitter-word-cloud-with-r/.

[4] Comeche, J.A.M. Los modelos clásicos de recuperación de información y su vigencia. *Memoria del Tercer Seminario Hispano-Mexicano de investigación en bibliotecología y documentación 29 al 31 de marzo de 2006*. 187.

[5] Facer, C. 2011. Text analysis: Hooking up your term document matrix to custom r code. (2011).

[6] Fernández, J.R. "Analizar con cierta profundidad la actividad de un #hashtag de twitter con r". Disponible en http://www.joserodriguez.info/bloc/analizar-actividad-hashtag-twitter-con-r/.

[7] Figueroa, J.C. "Sentiment-analysis-spanish". Disponible en https://github.com/JoseCardonaFigueroa/sentiment-analysis-spanish/tree/master/data.

[8] Gámez, J.C. "Programación en r: Paquete twitteR". Disponible en http://www.matematicasdigitales.com/programacion-en-r-paquete-twitter/.

[9] Kivelä, M., Arenas, A., Barthelemy, M., Gleeson, J.P., Moreno, Y. and Porter, M.A. 2014. Multilayer networks. *Journal of Complex Networks*. 2, 3 (2014), 203–271.

[10] Luque-Calvo, P.L. 2017. *Escribir un trabajo fin de estudios con r markdown*. Disponible en http://destio.us.es/calvo.

[11] Matteo Magnani, M.D. "Analysis and mining of multilayer social networks". Disponible en https://rdrr.io/cran/multinet/api/.

[12] normol 2011. Redes sociales. (2011).

[13] R Core Team 2018. *Package "multinet"*. R Foundation for Statistical Computing.

[14] R Core Team 2017. *Package "tm"*. R Foundation for Statistical Computing.

[15] R Core Team 2016. *Package "twitteR"*. R Foundation for Statistical Computing.

[16] Ríos Alcobendas, G. 2017. "Técnicas estadísticas en análisis de redes sociales". (2017).

[17] Rossi, M.E.D.M.M.L. 2016. *Multilayer social networks*. Cambridge University Press.

[18] Sanchez, G. "Mining twitter with r". Disponible en https://sites.google.com/site/

miningtwitter/intro/.

[19] Saracho, A. 2015. Definition - what does business intelligence (bi) mean? *AM de querétaro*. (2015).

[20] València, U. de "INTRODUCCIÓN al análisis cluster". Disponible en https://www.uv.es/ceaces/multivari/cluster/CLUSTER2.htm.

[21] Villena-Román, J. 2015. Introducción al análisis de sentimientos (minería de opiniones). (2015).