Fake Accounts Instagram

Antonio Cañete Baena

Invalid Date

Table of contents

Introducción

Este book recoge el proyecto realizado en la asignatura de *Laboratorio de computación científica* de la Universidad de Málaga.

El objetivo de este proyecto es el análisis de un dataset de la plataforma Kagle para extraer el maximo conocimiento posible usando las técnicas vista durante la asignatura.

1. DataSet:

El dataSet que vamos a utilizar se llama '**Instagram fake spammer genuine accounts',** obtenido de la web de kaggle. Este dataSet se compone de diferentes cuentas de instgram tanto de spammers como de usuarios genuinos.

Dicho dataSet esta formado por dos archivos, por un lado *test.csv,* un set de 120 entradas, 60 de cuentas genuinas y 60 de cuentas de spammer. Y por otro lado, otro archivo *train.csv*, formado por 576 entradas, donde al igual que en el archivo anterior, la mitad son cuentas genuinas y la otra mitad son spammers.

Spammer El «***spam***» es cualquier comunicación no solicitada enviada en masa.El «***spamming***» (que en español podría traducirse como «espamear») es el acto de enviar estos mensajes. Y la persona que envía los mensajes es un «***spammer***».

Enlace al DataSet

1 Análisis exploratorio de datos.

El análisis exploratorio de datos consiste en analizar el conjunto o conjuntos de datos de entrada con el objetivo de resumir sus características principales ayudando a su comprensión para futuras técnicas. Es una fase crucial en la ciencia de datos, ya que ayuda a los analistas de datos a comprender mejor los datos antes de aplicar modelos estadísticos más complejos o herramientas mas sofisticadas.

Para realizar dicho análisis, vamos a utilizar el *meta-package* de `tidyverse`.

1.1 Carga de datos

Como hemos visto anteriormente, el dataSet contienen dos archivos con datos, sin embargo, puesto que el archivo de *train.csv* contiene mas entradas de datos, vamos a utilizar dicho conjunto de datos para aplicarle nuestras técnicas de análisis.

En primer lugar, vamos a cargar las librerias necesarias y los datos:

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.0     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to beco
```

```
library(readr)
datos <- read_csv("Data/train.csv")
```

```
Rows: 576 Columns: 12
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl (12): profile pic, nums/length username, fullname words, nums/length ful...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Una vez tenemos nuestros datos, podemos ponernos manos a la obra.

Vamos a comenzar ojeando cuantas filas tenemos:

```
nrow(datos)
```

```
[1] 576
```

Y también vemos cuantos atributos tiene cada fila y sus nombres:

```
ncol(datos)
```

```
[1] 12
```

```
colnames(datos)
```

```
 [1] "profile pic"          "nums/length username" "fullname words"
 [4] "nums/length fullname" "name==username"       "description length"
 [7] "external URL"         "private"              "#posts"
[10] "#followers"           "#follows"             "fake"
```

Veamos las primeras filas del dataset

```
head(datos)
```

```
# A tibble: 6 x 12
  `profile pic` `nums/length username` `fullname words` `nums/length fullname`
          <dbl>                  <dbl>            <dbl>                   <dbl>
1             1                   0.27                0                       0
2             1                   0                   2                       0
3             1                   0.1                 2                       0
4             1                   0                   1                       0
```

```
5             1                   0                   2                       0
6             1                   0                   4                       0
# i 8 more variables: `name==username` <dbl>, `description length` <dbl>,
#   `external URL` <dbl>, private <dbl>, `#posts` <dbl>, `#followers` <dbl>,
#   `#follows` <dbl>, fake <dbl>
```

Podemos ver que todas las columnas tienen valores numericos, pero vamos a comprobarlo mirando dentro su estructura:

```
str(datos)
```

```
spc_tbl_ [576 x 12] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ profile pic         : num [1:576] 1 1 1 1 1 1 1 1 1 1 ...
 $ nums/length username: num [1:576] 0.27 0 0.1 0 0 0 0 0 0 0 ...
 $ fullname words      : num [1:576] 0 2 2 1 2 4 2 2 0 2 ...
 $ nums/length fullname: num [1:576] 0 0 0 0 0 0 0 0 0 0 ...
 $ name==username      : num [1:576] 0 0 0 0 0 0 0 0 0 0 ...
 $ description length  : num [1:576] 53 44 0 82 0 81 50 0 71 40 ...
 $ external URL        : num [1:576] 0 0 0 0 0 1 0 0 0 1 ...
 $ private             : num [1:576] 0 0 1 0 1 0 0 0 0 0 ...
 $ #posts              : num [1:576] 32 286 13 679 6 344 16 33 72 213 ...
 $ #followers          : num [1:576] 1000 2740 159 414 151 ...
 $ #follows            : num [1:576] 955 533 98 651 126 ...
 $ fake                : num [1:576] 0 0 0 0 0 0 0 0 0 0 ...
 - attr(*, "spec")=
  .. cols(
  ..   `profile pic` = col_double(),
  ..   `nums/length username` = col_double(),
  ..   `fullname words` = col_double(),
  ..   `nums/length fullname` = col_double(),
  ..   `name==username` = col_double(),
  ..   `description length` = col_double(),
  ..   `external URL` = col_double(),
  ..   private = col_double(),
  ..   `#posts` = col_double(),
  ..   `#followers` = col_double(),
  ..   `#follows` = col_double(),
  ..   fake = col_double()
  .. )
 - attr(*, "problems")=<externalptr>
```

Hemos comprobado que todos los valores son del tipo numerico y double.

```
anyNA(datos)
```

```
[1] FALSE
```

Por ultimo comprobamos que no existen valores NA dentro del dataset, lo que nos ayudara en su próximo análisis exploratorio.

1.2 Analisis de atributos

Una vez visto un poco por encima la estructura del dataSet, vamos a explorar uno a uno los atributos de cada fila, explorando su significado, los valores limite, ...

1.2.1 profile pic

Este es un atributo binario que indica si un usuario tiene foto de perfil. Por lo tanto solo tenemos o 0 o 1.

```
str(datos$`profile pic`)
```

```
 num [1:576] 1 1 1 1 1 1 1 1 1 1 ...
```

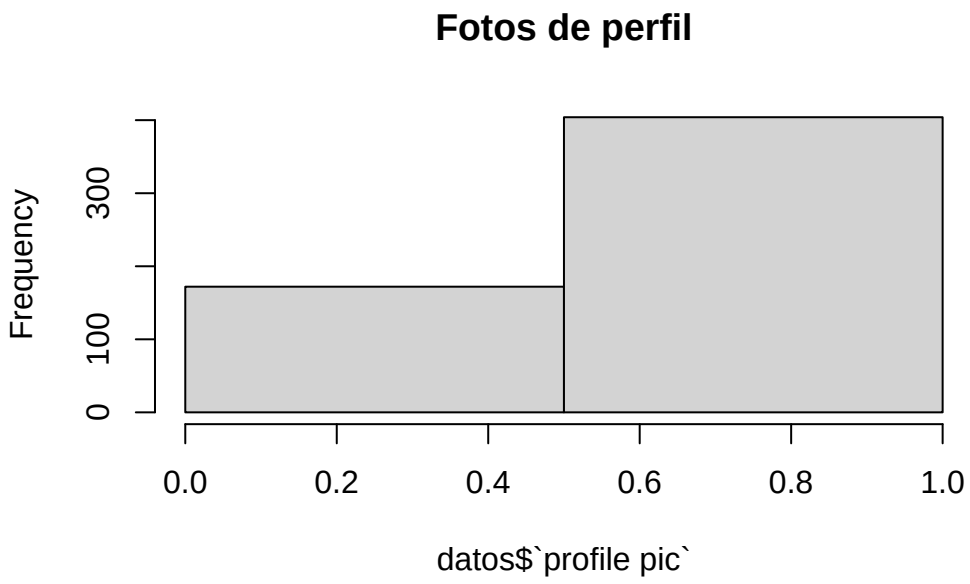
```
anyNA(datos$`profile pic`)
```

```
[1] FALSE
```

Vemos que esta columna no contiene ningun NA.

Vamos a visualizar la proporcion de usuarios con foto de perfil:

```
hist(datos$`profile pic`, breaks = 2, main="Fotos de perfil" )
```

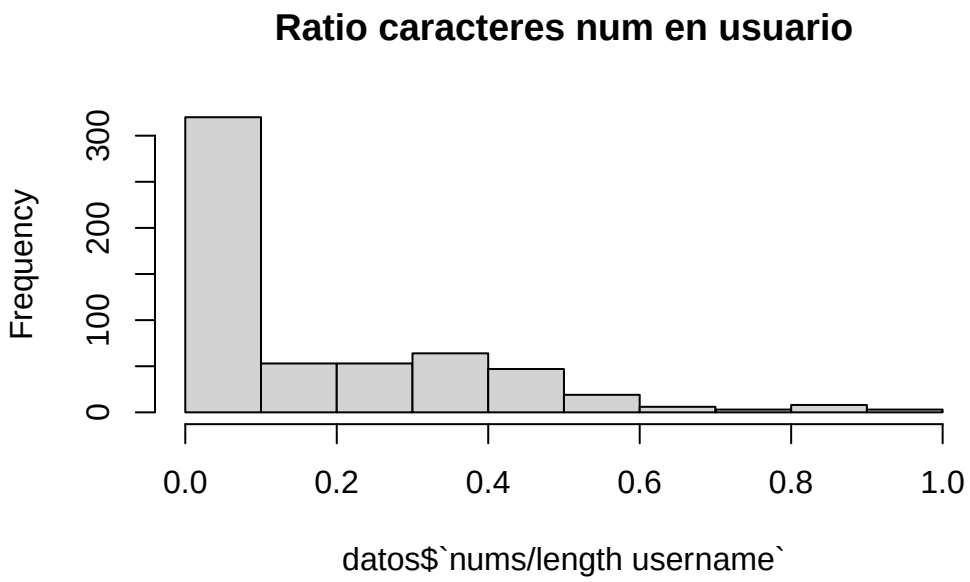


Observamos que más de la mitad de los usuarios tienen foto de perfil.

1.2.2 nums/length username

Este atributo representa el ratio de numero de caracteres numericos en el nombre de usuario respecto su longutud. Ej: Ant234 -> Ratio 1.

```
hist(datos$`nums/length username`, main="Ratio caracteres num en usuario" )
```



1.2.3 fullname words

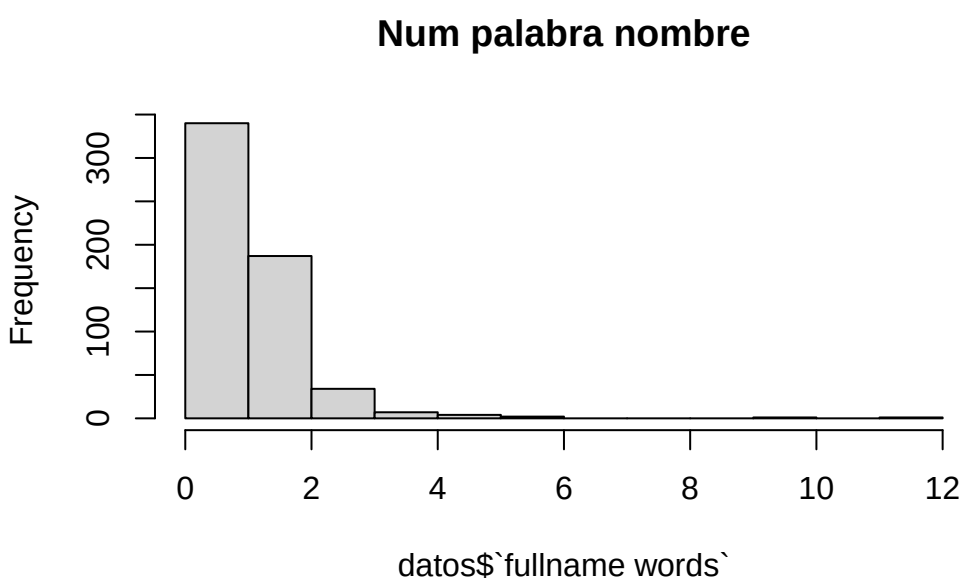
Este atributo representa la cantidad de palabras que componen el nombre del usuario.

```
max(datos$`fullname words`)
```

```
[1] 12
```

Observamos que hay uno o varios usuarios cuyo nombre tiene 12 palabras de longitud, algo que es poco común.

```
hist(datos$`fullname words`, main="Num palabra nombre" )
```



Analizando el histograma, vemos que la mayoría de usuarios tiene entre 0 y 1 palabras en su nombre.

```
count(filter(datos,`fullname words`==1 | `fullname words`==2))/count(datos)*100
```

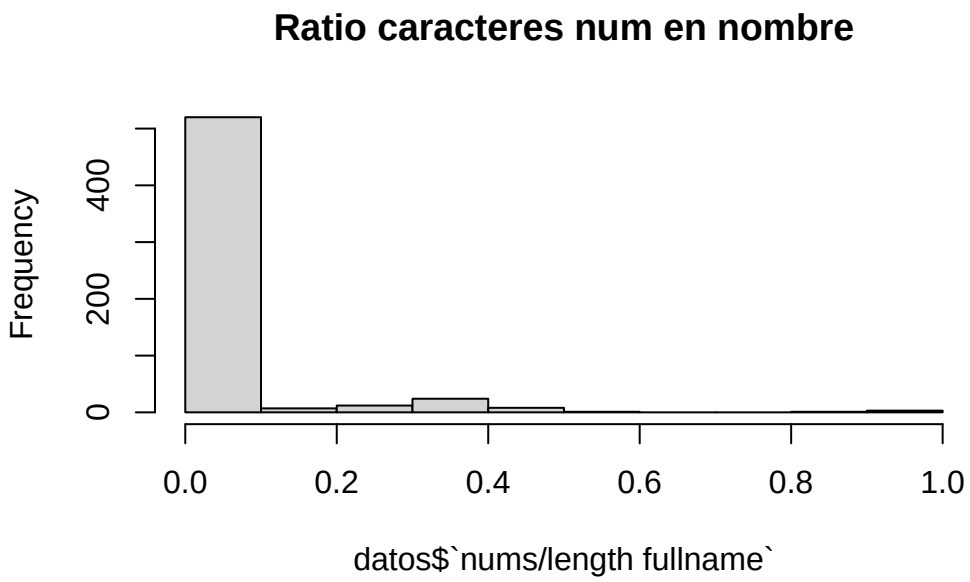
```
         n
1 81.59722
```

En concreto el 81,6% de los datos tienen entre 0 y 1 palabras en su nombre.

1.2.4 nums/length fullname

Este atributo representa el ratio de numero de caracteres numéricos en el nombre completo del usuario respecto su longitud.

```
hist(datos$`nums/length fullname`, main="Ratio caracteres num en nombre" )
```

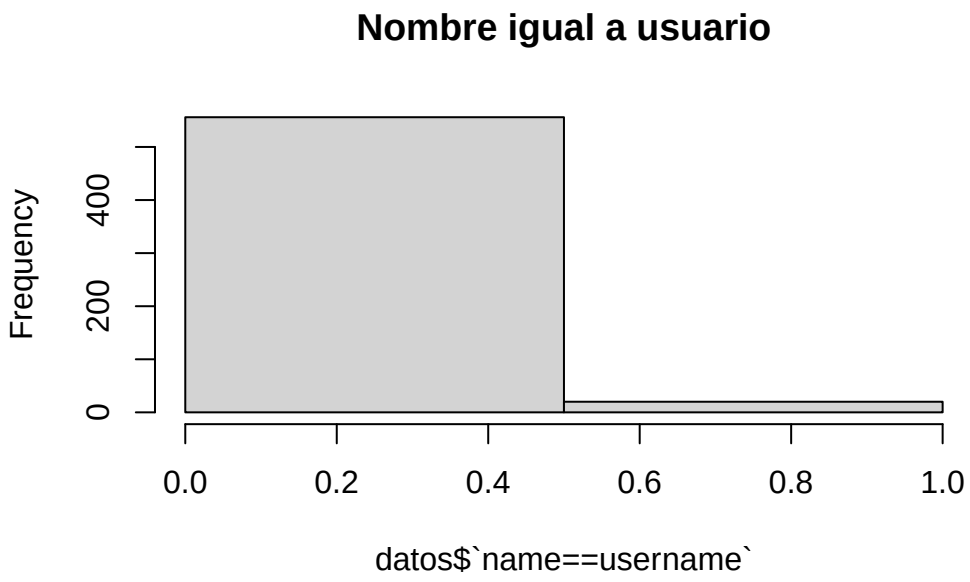


Observamos que es bastante inusual que un usuario tenga caracteres en su nombre completo, mientra que, como hemos visto antes, en el nombre de usuario, es más frecuente encontrar caracteres.

1.2.5 name==username

Este atributo es un atributo binario que representa si el usuario tiene el mismo nombre de usuario y nombre completo.

```
hist(datos$`name==username`, breaks = 2, main="Nombre igual a usuario" )
```

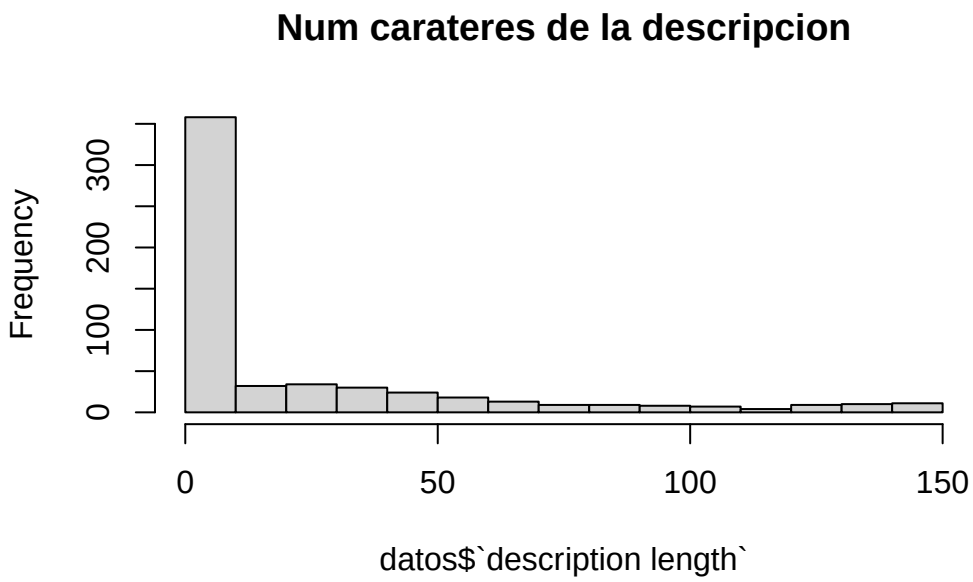


Concluimos que el bastante inusual que un usuarios tenga el mismo nombre de usuario y nombre completo.

1.2.6 description length

Este atributo representa la longitud de la descripción del perfil de usuario (en caracteres).

```
hist(datos$`description length`, main="Num carateres de la descripcion" )
```



Podemos intuir que el máximo de caracteres que ofrece Instagram en su descripción es 150, cuyo limite es alcanzado por pocos usuarios del DataSet:

```
filter(datos,datos$`description length` ==150) %>% count() %>% summarise(`Num de usuarios`=n)
```

```
# A tibble: 1 x 1
  `Num de usuarios`
              <int>
1                 2
```

Viendo el histograma, descubrimos que la mayoría de usuarios tienen una descripción con pocos caracteres, pero vamos a calcular la media para poder tener una idea:

```
mean(datos$`description length`)
```

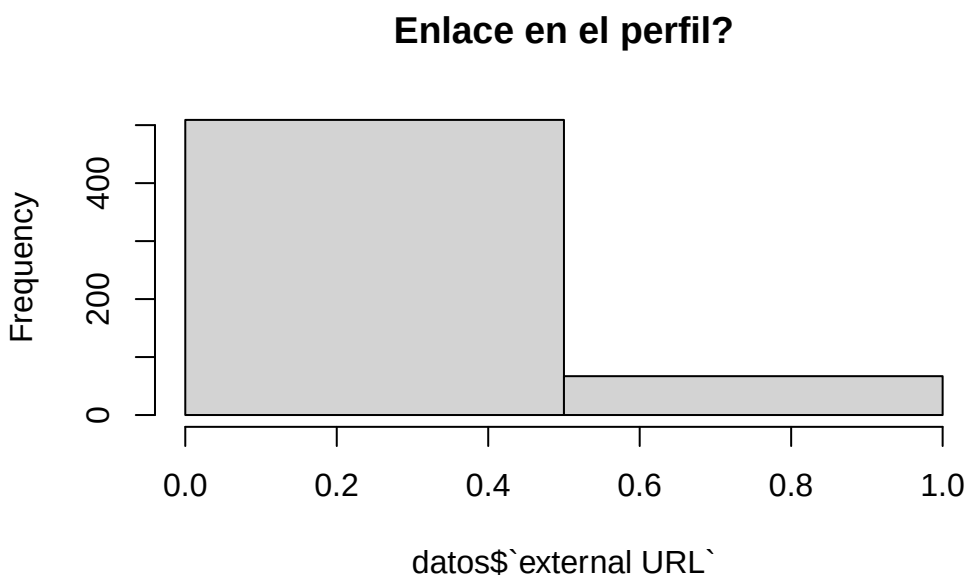
```
[1] 22.62326
```

Encontramos que la media de caracteres en la descripción, lo que, dependiendo del idioma, puede ser una pequeña frase o algunas palabras. Las descripciones largar son menos frecuentes.

1.2.7 external URL

Este atributo es un atributo binario que representa si el perfil tiene algún enlaza externo en el.

```
hist(datos$`external URL`, breaks = 2, main="Enlace en el perfil?" )
```

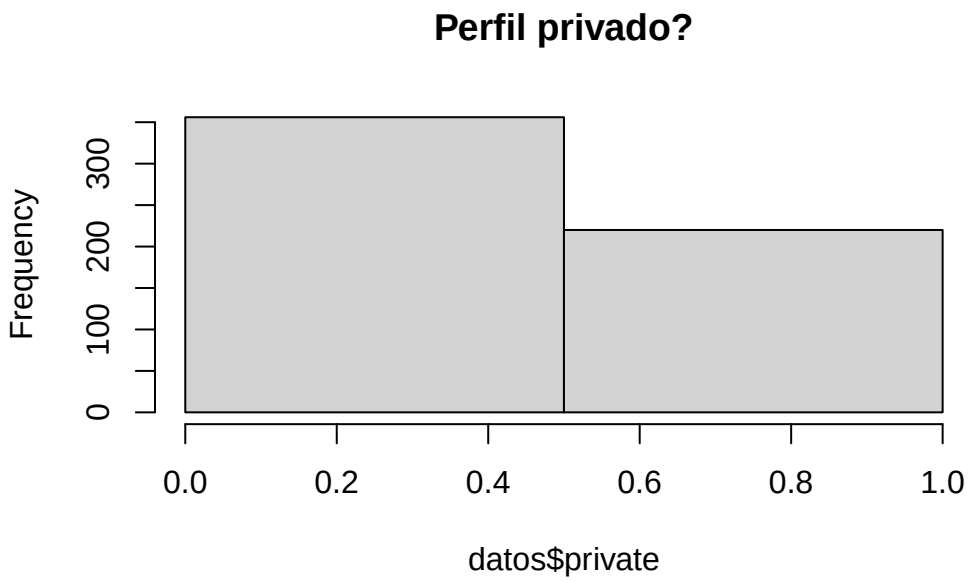


Lo mas común son los perfiles sin enlaces externos.

1.2.8 private

Este atributo es un atributo binario que representa si el perfil es privado o publico.

```
hist(datos$`private`, breaks = 2, main="Perfil privado?" )
```



En este atributo encontramos algo mas de igualdad, el numero de cuentas privadas son poco mas de la mitad del numero de publicas:

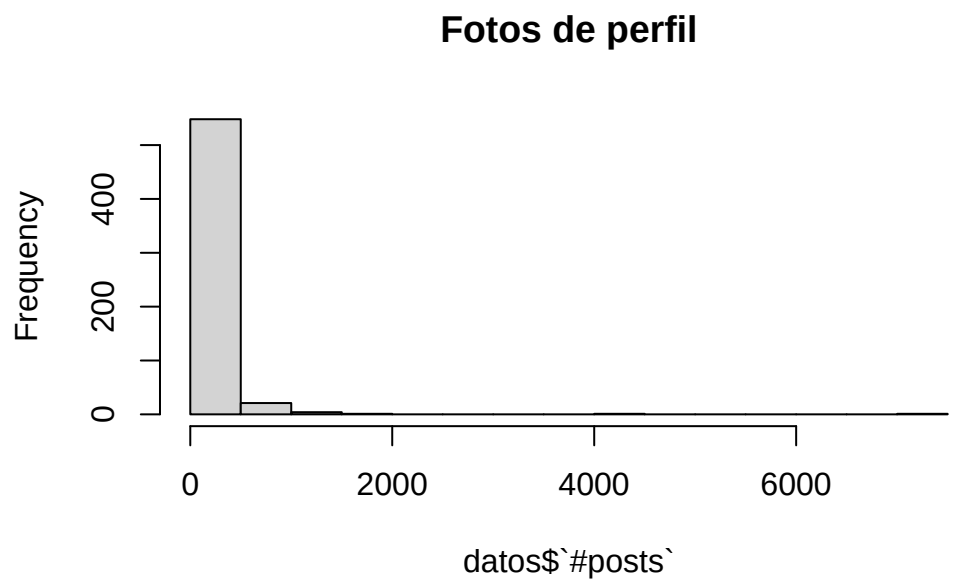
```
datos %>% mutate(private = ifelse(private==1,"Privada","Publica")) %>%  group_by(private) %>%
```

```
# A tibble: 2 x 2
  private Numero
  <chr>    <int>
1 Privada    220
2 Publica    356
```

1.2.9 post

Este atributo representa el numero de publicaciones de la cuenta.

```
hist(datos$`#posts`,  main="Fotos de perfil" )
```




```
max(datos$`#posts`)
```

```
[1] 7389
```

Obtenemos un histograma un poco extrano al haber algun valor muy alto de publicaciones, vamos a buscarlo:

```
max(datos$`#posts`)
```

```
[1] 7389
```

Vemos que es un valor bastante raro o que se podría tratar de alguna cuenta que publicase mucho contenido a diario. Vamos a verla:

```
datos %>% filter(`#posts`==7389)
```

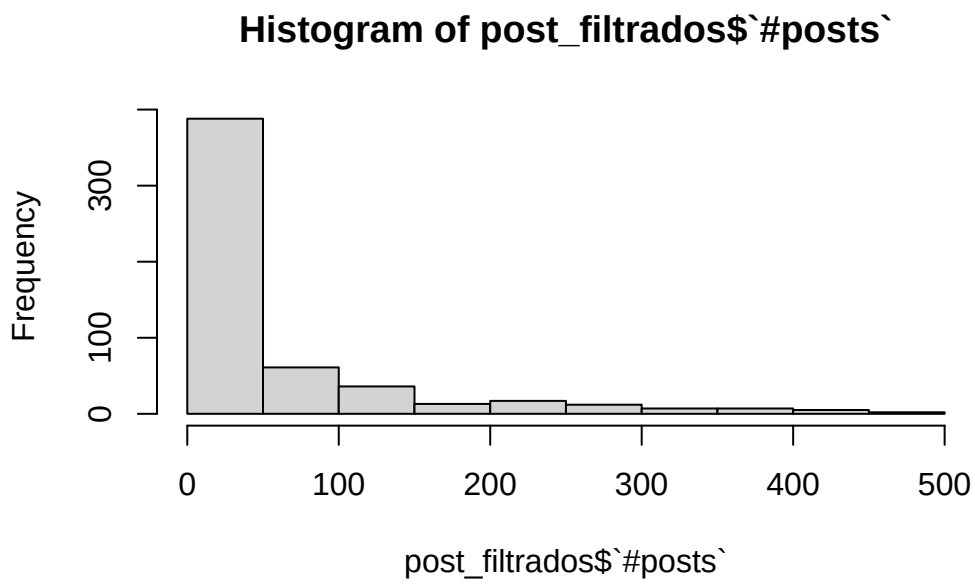
```
# A tibble: 1 x 12
  `profile pic` `nums/length username` `fullname words` `nums/length fullname`
          <dbl>                  <dbl>            <dbl>                   <dbl>
1             1                      0                0                       0
```

```
# i 8 more variables: `name==username` <dbl>, `description length` <dbl>,
#   `external URL` <dbl>, private <dbl>, `#posts` <dbl>, `#followers` <dbl>,
#   `#follows` <dbl>, fake <dbl>
```

Como dato, la cuenta de Dwayne Johnson, ex-luchador de la WWE y exitoso actor de Hollywood, tiene unos 7800 post, por lo que dicho valor puede ser debido a la cuenta de algún famoso.

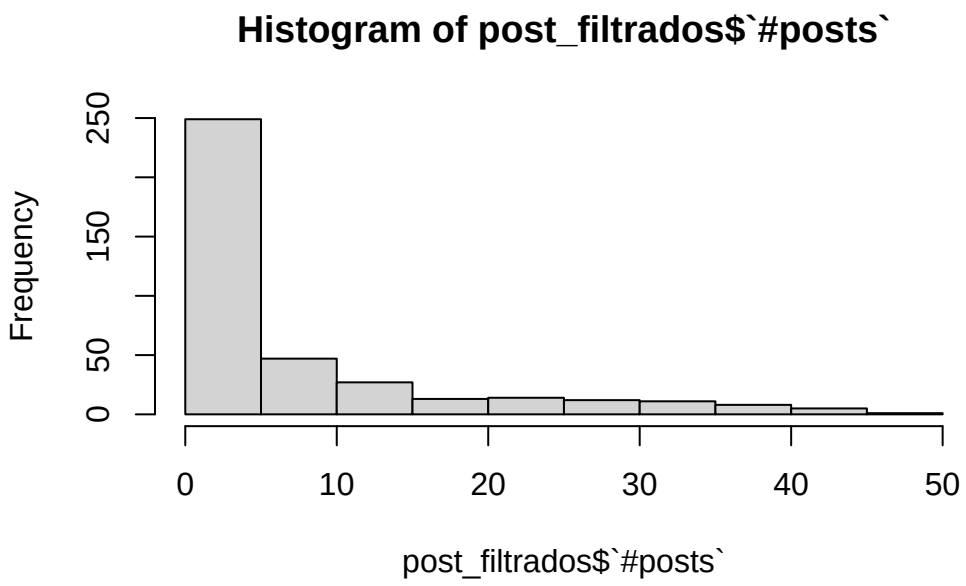
Vamos a volver a dibujar el histograma pero con un umbral un poco mas razonable:

```
post_filtrados <- datos %>% select(`#posts`)%>% filter(`#posts` <500)
hist(post_filtrados$`#posts`)
```



Ahora ya podemos extraer informacion mas facilmente, como que la mayoria de usuarios tiene menos de 50 publicaciones, vamos a verlo en mas detalle:

```
post_filtrados <- datos %>% select(`#posts`)%>% filter(`#posts` <50)
hist(post_filtrados$`#posts`)
```



Observamos que hay un gran numero de usuarios con menos de 5 posts. Vamos a ver cuantos de ellos tiene 0 posts y a ver la media total:

```
datos %>% filter(`#posts`==0) %>% count()
```

```
# A tibble: 1 x 1
      n
  <int>
1   157
```

```
mean(datos$`#posts`)
```

```
[1] 107.4896
```

Aunque como antes hemos visto que hay usuarios con un gran numero de posts, esta media puede ser no muy significativa.

Vamos a analizar entonces sus cuartiles y mediana:

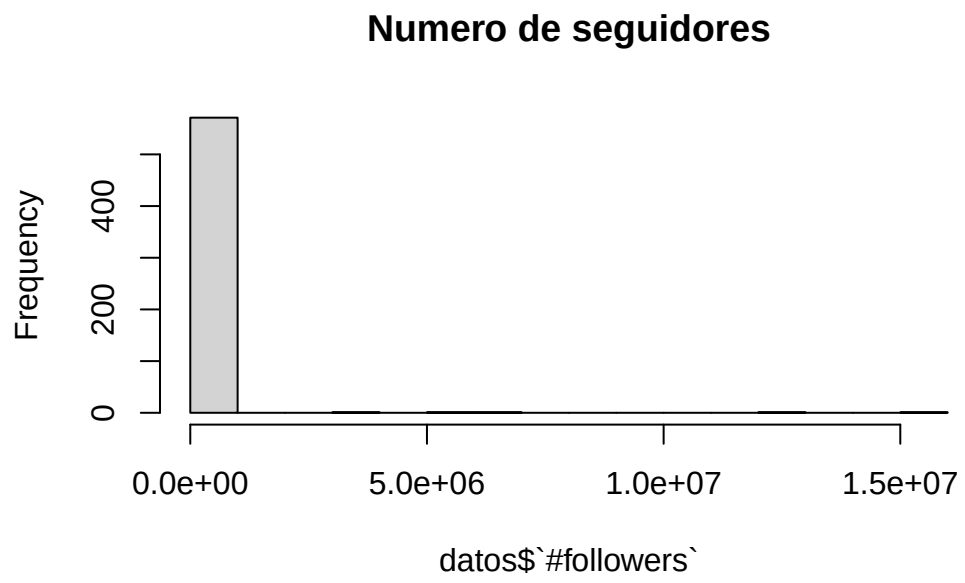
```
summary(datos$`#posts`)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    0.0     0.0     9.0   107.5    81.5  7389.0
```

1.2.10 followers

Este atributo representa el numero de seguidores de la cuenta.

```
hist(datos$`#followers`, main="Numero de seguidores" )
```




```
max(datos$`#followers`)
```

```
[1] 15338538
```

Como en el atributo anteriro, este histograma no tiene sentido porque hay algun valor muy alto

```
max(datos$`#followers`)
```

```
[1] 15338538
```

Dicho valor solo tiene sentido que sea debido a una cuenta de alguna celebridad, vamos a comprobar si es el mismo que tiene similitud con el valor anómalo de post encontrado anteriormente:

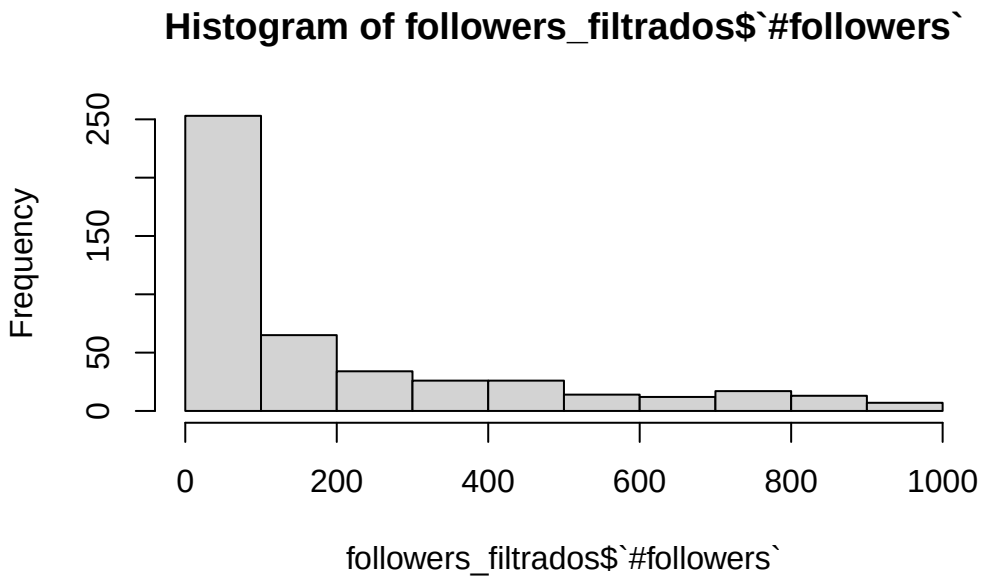
```
datos %>% filter (`#followers`==max(`#followers`)) %>% select(`#posts`)
```

```
# A tibble: 1 x 1
  `#posts`
     <dbl>
1      148
```

Aun pudiendo ser la cuenta de una celebridad, vemos que tiene un numero de post relativamente normal, comparado con el valor de 7389 post que obtuvimos anteriormente.

Vamos a volver a hacer el histograma con un nuevo umbral mas bajo:

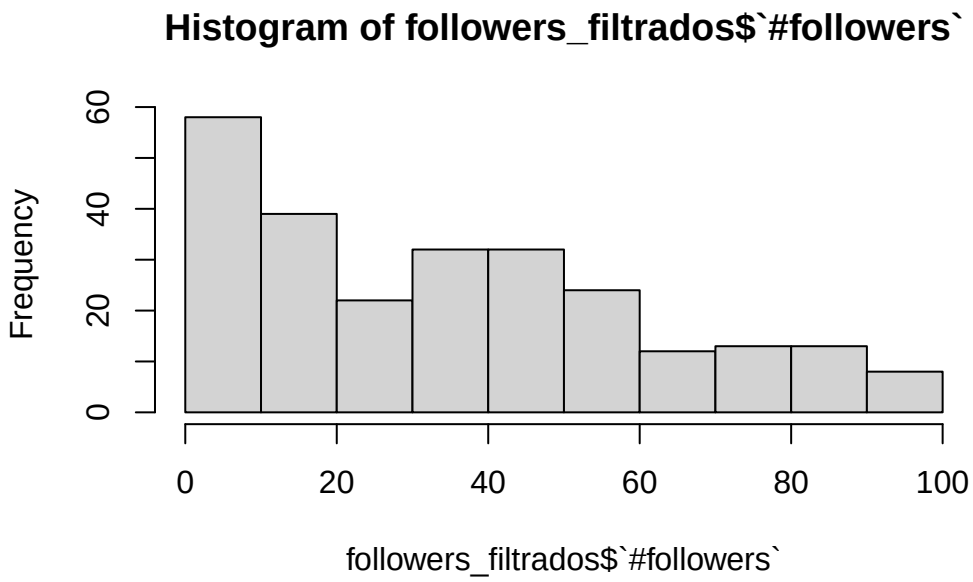
```
followers_filtrados <- datos %>% select(`#followers`)%>% filter(`#followers` <1000)
hist(followers_filtrados$`#followers`)
```



Observamos que la mayoría de usuarios no tiene gran numero de seguidores, en concreto, menos de 100.

Vamos a verlo

```
followers_filtrados <- datos %>% select(`#followers`)%>% filter(`#followers` <100)
hist(followers_filtrados$`#followers`)
```



Vemos que en este intervalo, si esta mas repartidas las frecuencias. Aunque resulta curioso que una gran cantidad de usuarios no llegue a los 50 seguidores.

Viendo que hay algunos usuarios con un gran numero de seguidores, no tiene sentido tomar el valor de la mediana como referencia ya que esta no es significativa en este caso, por lo que vamos a analizar los cuartiles y la mediana en su lugar.

```
mean(datos$`#followers`)
```

```
[1] 85307.24
```

```
summary(datos$`#followers`)
```

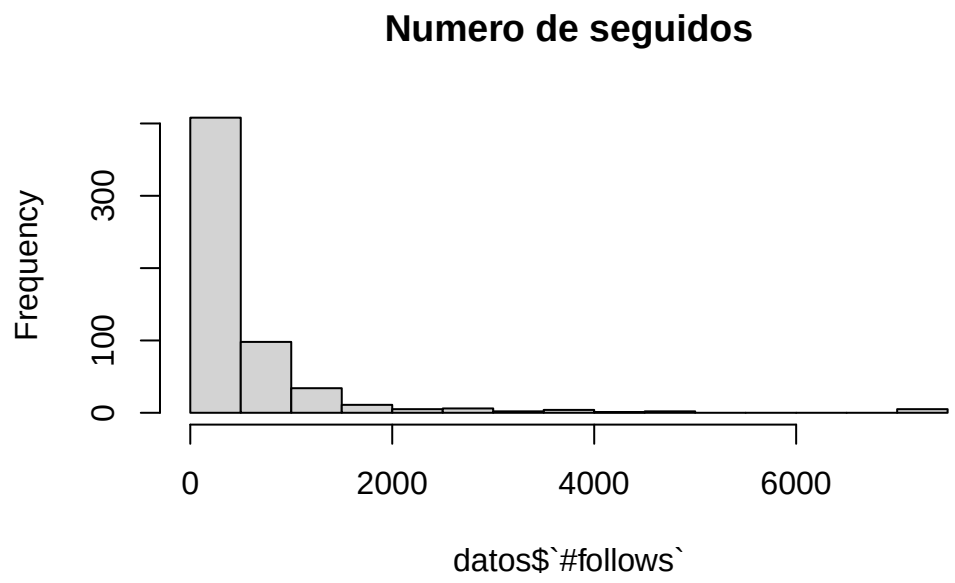
```
   Min.  1st Qu.   Median     Mean  3rd Qu.     Max. 
      0       39      150    85307      716 15338538 
```

Sabiendo que la mediana divide al 50% de los datos, dicho valor es mas significativo que la media.

1.2.11 follows

Este atributo representa el numero de usuarios seguidos por la cuenta.

```
hist(datos$`#follows`, main="Numero de seguidos" )
```




```
max(datos$`#follows`)
```

```
[1] 7500
```

Al igual que en los dos anteriores, los valores máximos hacen que nuestro histograma no sea muy entendible, vamos a estudiarlo:

```
max(datos$`#follows`)
```

```
[1] 7500
```

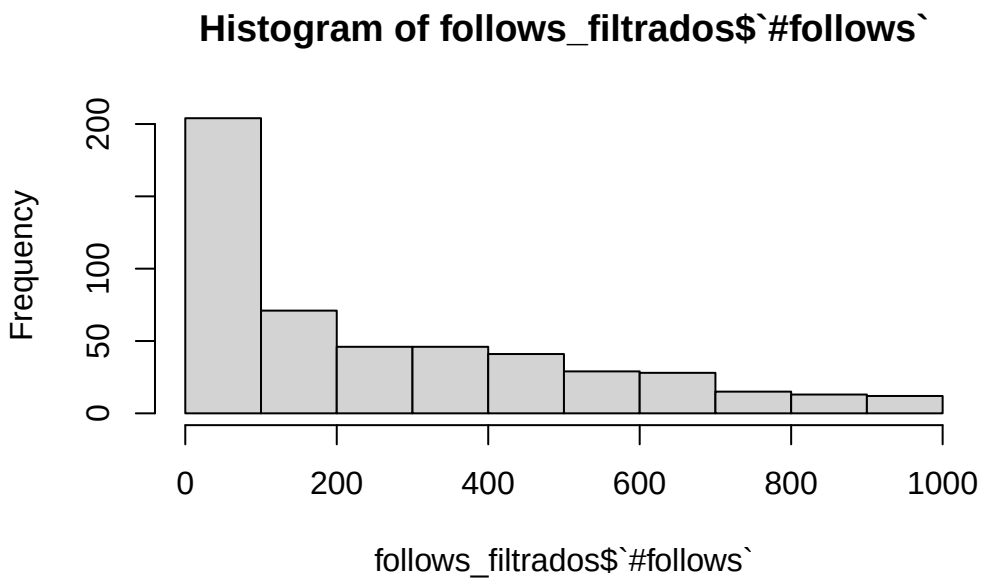
Dicho valor corresponde con el valor máximo de cuentas que Instagram permite a los usuarios seguir para reducir el spam. Por lo tanto, las cuentas que siguen a un gran numero de personas se pueden llegar a asociar a spammers. Vamos a ver cuantas cuentas están en este limite:

```
count(filter(datos,datos$`#follows`==7500))
```

```
# A tibble: 1 x 1
      n
  <int>
1     2
```

Ahora para poder ahcernos una mejor idea vamos a volver a dibujar el histograma con un nuevo umbral reducido.

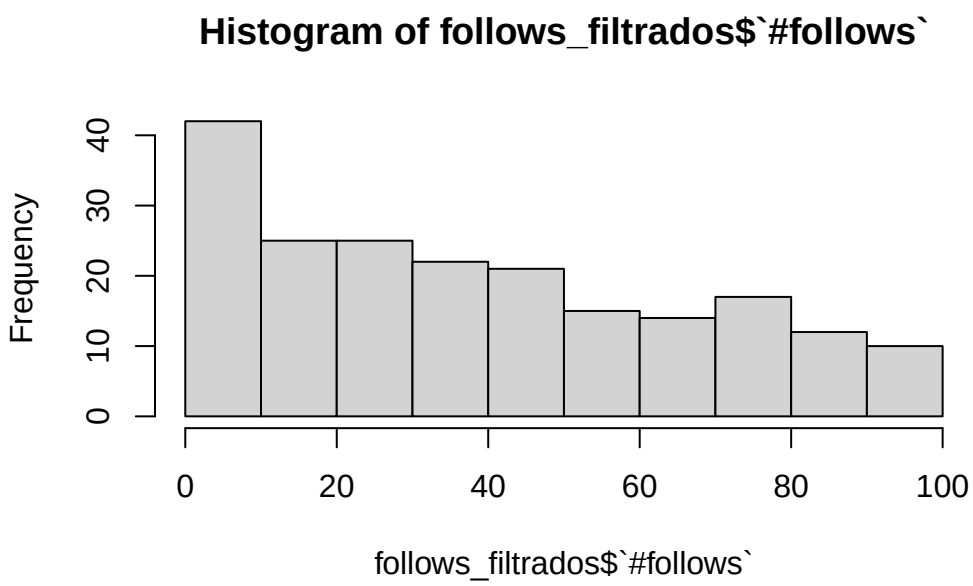
```
follows_filtrados <- datos %>% select(`#follows`)%>% filter(`#follows` <1000)
hist(follows_filtrados$`#follows`)
```



Observamos que mas de la mitad de usuarios no sigue a muchas otras cuentas, en concreto, menos de 100.

Vamos a verlo:

```
follows_filtrados <- datos %>% select(`#follows`)%>% filter(`#follows` <100)
hist(follows_filtrados$`#follows`)
```



Vemos que en este intervalo, si esta mas repartidas las frecuencias.

Viendo que hay algunos usuarios con un gran numero de cuentas seguidas, no tiene sentido tomar el valor de la mediana como referencia ya que esta no es significativa en este caso, por lo que vamos a analizar los cuartiles y la mediana en su lugar.

```
summary(datos$`#follows`)
```

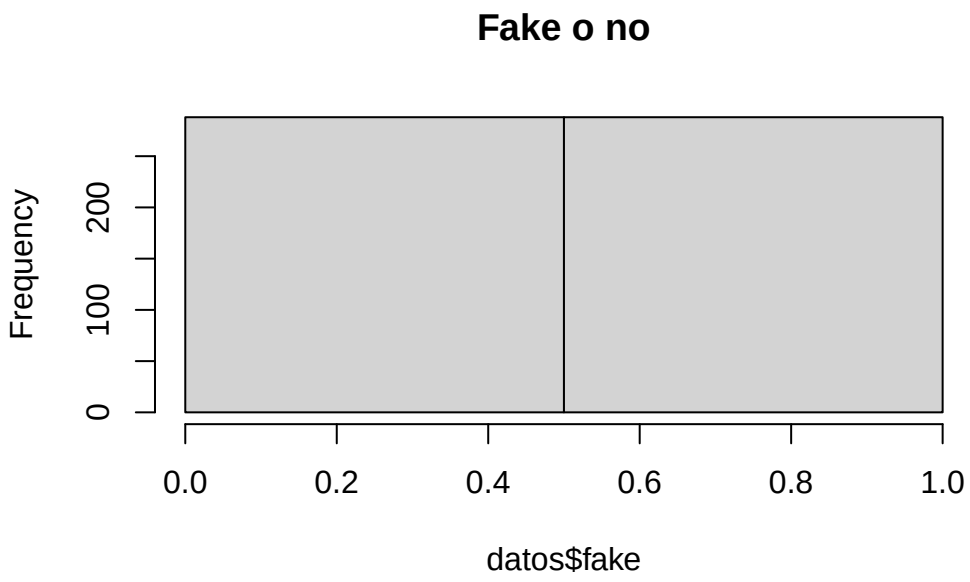
```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    0.0    57.5   229.5   508.4   589.5  7500.0
```

Ahora ya con estos valores ya podemos analizarlo un poco mejor y darnos cuenta que el 50% de los usuarios no sigue a mas de 229 cuentas.

1.2.12 fake

Por ultimo, este atributo es un atributo binario que representa si el perfil es verdadero o es un spammer.

```
hist(datos$fake, breaks = 2, main="Fake o no" )
```



Observamos que nuestro DataSet tiene un 50% de cuentas falsas y otro 50% de cuentas verdaderas.

1.3 Herramienta de DataExplorer

```
library(DataExplorer)
```

```
Warning: package 'DataExplorer' was built under R version 4.3.3
```

```
#create_report(datos)
```

DataExplorer: Automate Data Exploration and Treatment Automated data exploration process for analytic tasks and predictive modeling, so that users could focus on understanding data and extracting insights. The package scans and analyzes each variable, and visualizes them with typical graphical techniques. Common data processing methods are also available to treat and format data.

La librería DataExplorer es una herramienta diseñada para simplificar y acelerar el proceso de exploración y análisis de datos. Proporciona funciones que permiten generar rápidamente resúmenes estadísticos, visualizaciones y diagnósticos de los datos.

Algunas de las características clave incluyen la capacidad de generar perfiles de datos detallados, identificar valores atípicos, analizar la distribución de variables y explorar relaciones entre variables.

Podemos simplificar el proceso realizado anteriormente utilizando este paquete.

1.3.1 Funciones interesantes

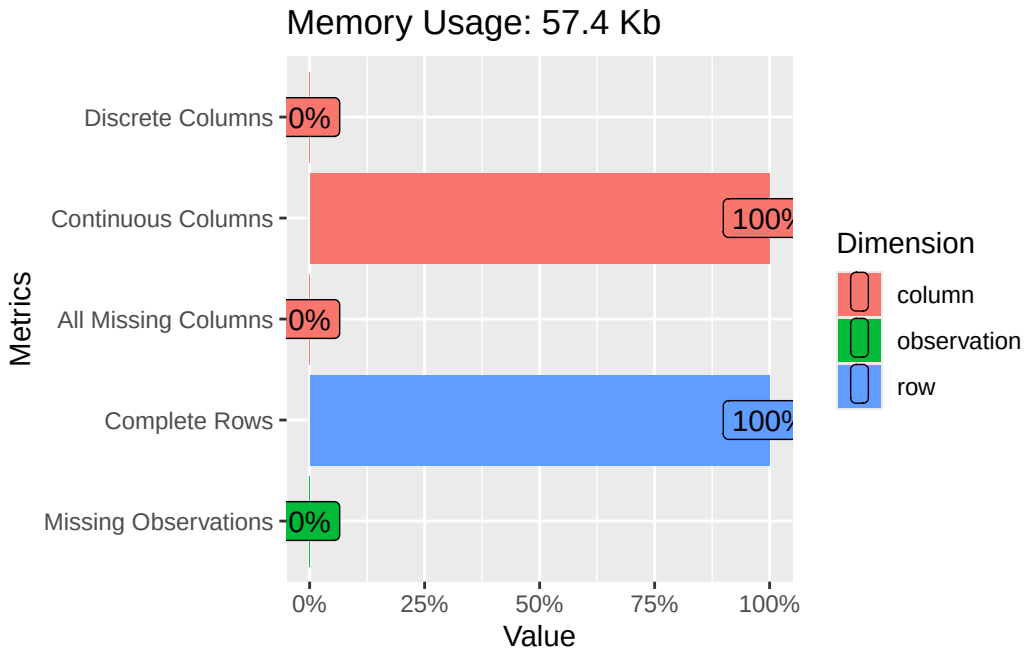
1.3.1.1 introduce

Genera un pequeño reporte con los datos mas relevantes como el numero de columnas, el tamano del datset, ...

```
introduce(datos)
```

```
# A tibble: 1 x 9
   rows columns discrete_columns continuous_columns all_missing_columns
  <int>   <int>            <int>              <int>                <int>
1   576      12                0                 12                    0
# i 4 more variables: total_missing_values <int>, complete_rows <int>,
#   total_observations <int>, memory_usage <dbl>
```

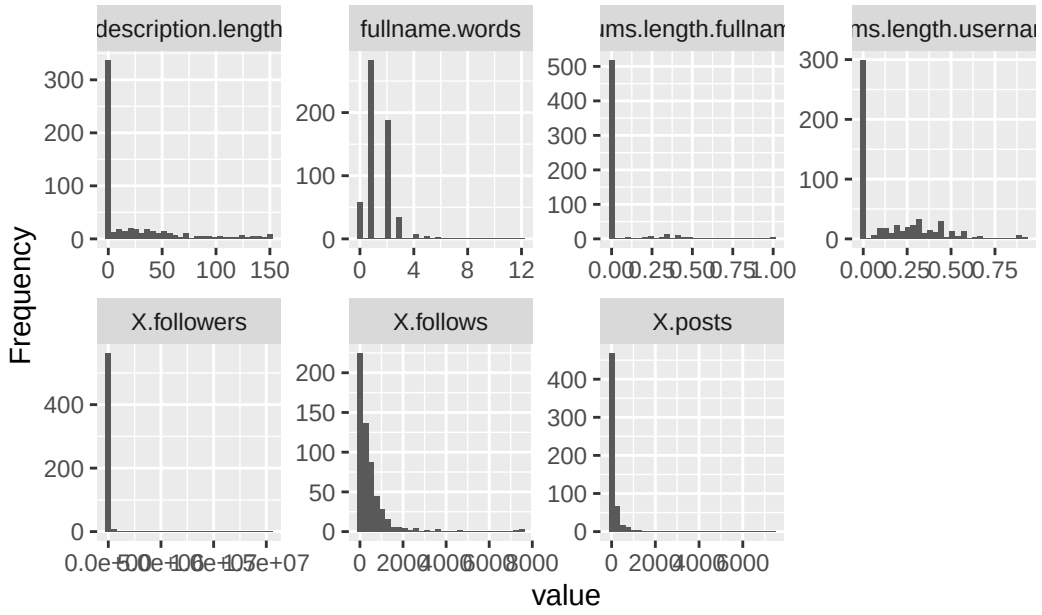
```
plot_intro(datos)
```



1.3.1.2 plot_histogram

Esta funcion nos muestra todos los histogramas de las variables/columnas.

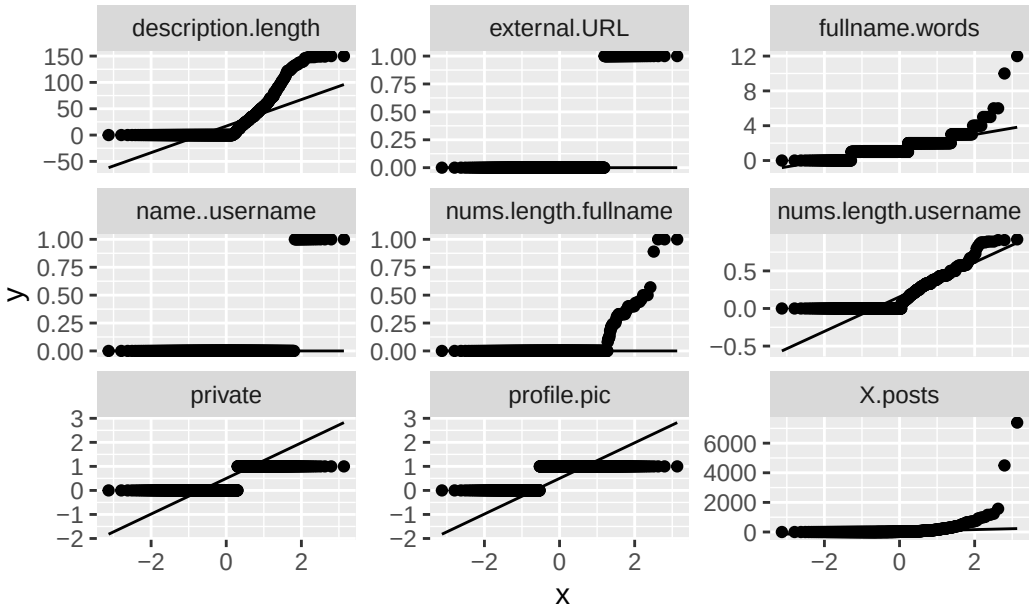
```
plot_histogram(datos)
```



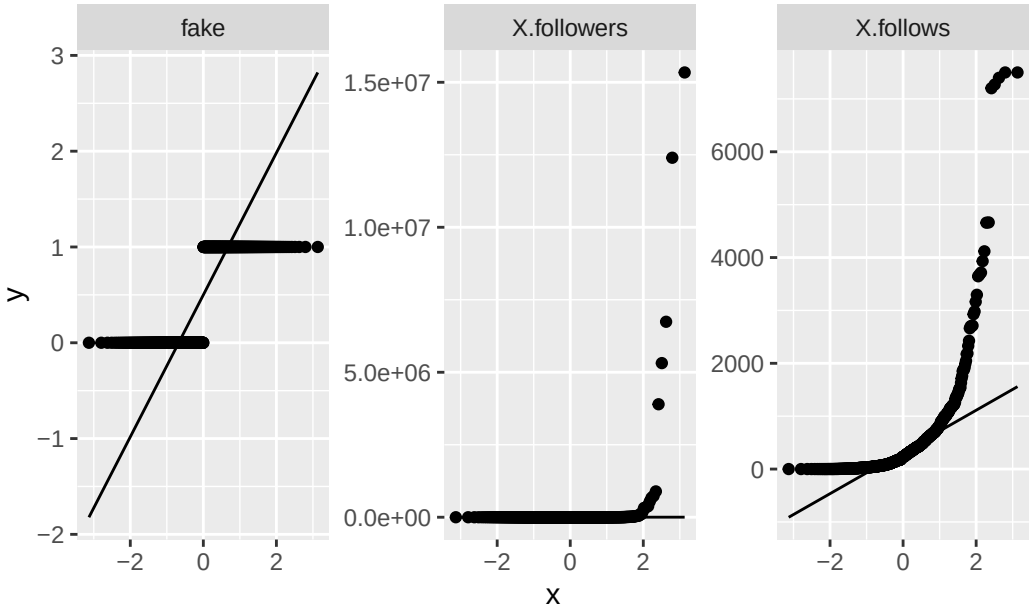
1.3.1.3 plot_qq

Este comando genera un gráfico de cuantiles-cuantiles, el cual es una forma de visualizar la desviación de una distribución de probabilidad específica.

```
plot_qq(datos)
```


description.length
external.URL
fullname.words
name..username
nums.length.fullname
nums.length.username
private
profile.pic
X.posts
y
x

Page 1


fake
X.followers
X.follows
y
x

Page 2

1.3.1.4 create_report

Este comando realiza las medidas mencionadas anteriormente y muchas otras que son útiles (como el análisis de componentes principales) para el análisis exploratorio y genera como salida un reporte completo de nuestros datos.

```
#create_report(datos)
```

2 Visualización de los Datos

Ahora que ya hemos analizado en profundidad cada atributo de nuestro DataSet, vamos a necesitar algunos gráficos que nos den ideas sobre como continuar nuestro análisis.

Para ello vamos a utilizar al herramienta de `ggplot2` , la cual nos va a permitir realizar los gráficos complejos de los que estamos hablando.

ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics A system for 'declaratively' creating graphics, based on "The Grammar of Graphics". You provide the data, tell 'ggplot2' how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details.

Enlace a la librería

Vamos a comenzar importando la librería y cargando nuestros datos.

```
library(ggplot2)
library(readr)
library(magrittr)
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
datos <- read_csv("Data/train.csv")
```

```
Rows: 576 Columns: 12
```

```
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl (12): profile pic, nums/length username, fullname words, nums/length ful...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

2.1 Pre-procesado

Para poder hacer este trabajo mas fácil, vamos a realizar un pre-procesado de los datos primero. Vamos a convertir todos los atributos que son discretos a factores:

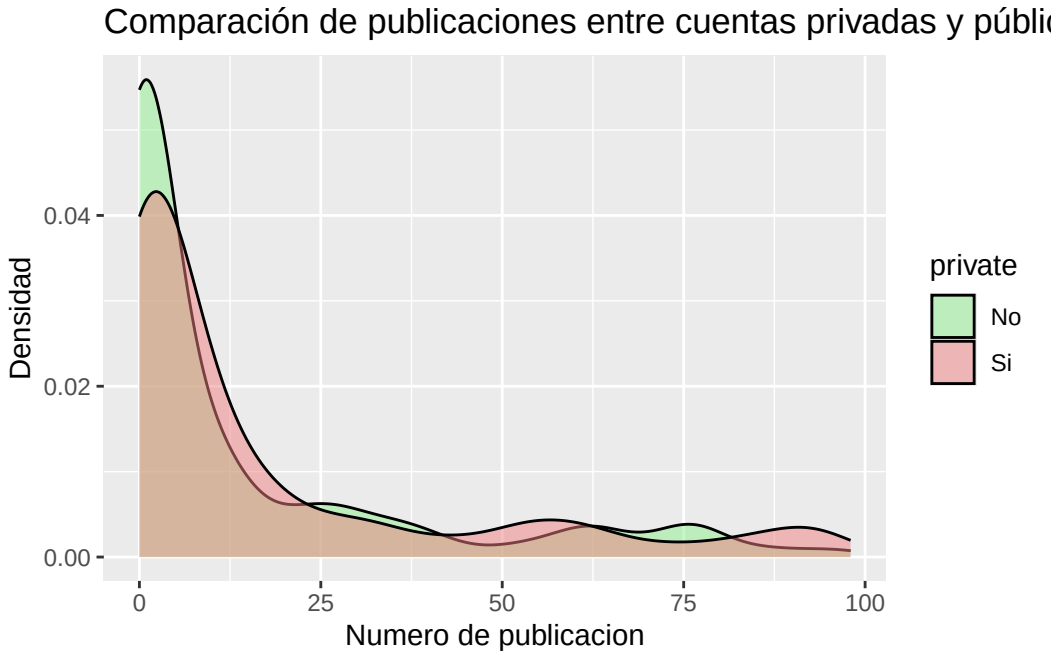
```
datos_refinados <- datos
columnas_binarias = c("profile pic","name==username","external URL","fake","private")
for (columna in columnas_binarias) {
  datos_refinados[[columna]] <-  factor(datos_refinados[[columna]], labels = c("No", "Si"))
}
```

Nuestro atributos discretos, son binarios, solo tienen o bien *Si* o *No*. Vamos a emplear ahora los gráficos para poder encontrar alguna relación entre las variables y sobre todo, lo que mas no interesa, si alguna tiene relación con las cuentas de spam.

2.2 Comparación de la cantidad de publicaciones entre cuentas privadas y públicas

Para ver como se comportan ambos tipos de usuarios, vamos a empezar analizando el numero de publicaciones entre los usuarios con cuentas publicas y con cuentas privadas, para ello vamos ver las densidades utilizando `geom_density` :

```
posts_filtrados <- datos_refinados %>%  filter(`#posts` <100)

ggplot(data = posts_filtrados, aes( x = `#posts`,fill = private)) +
  geom_density(alpha = 0.5) +
  labs(title = "Comparación de publicaciones entre cuentas privadas y públicas",
       x = "Numero de publicacion",
       y = "Densidad") +
  scale_fill_manual(values = c("lightgreen", "lightcoral"))
```



Vemos que ambos casos, nuestra gráfica se asemeja, por lo que el numero de publicaciones no depende de si es privada o publica.

Sin embargo, realmente a nosotros nos interese encontrar relaciones para intentar determinar si una cuenta es spammer o de un persona real. Por lo tanto vamos a centrarnos en comparar los atributos con el atributos spam.

2.3 Relación entre visibilidad del perfil y cuentas fake

Como nos interesa buscar las cuentas de spam, vamos a ver si la visibilidad del perfil (cuenta privada o publica), tiene algo que ver:

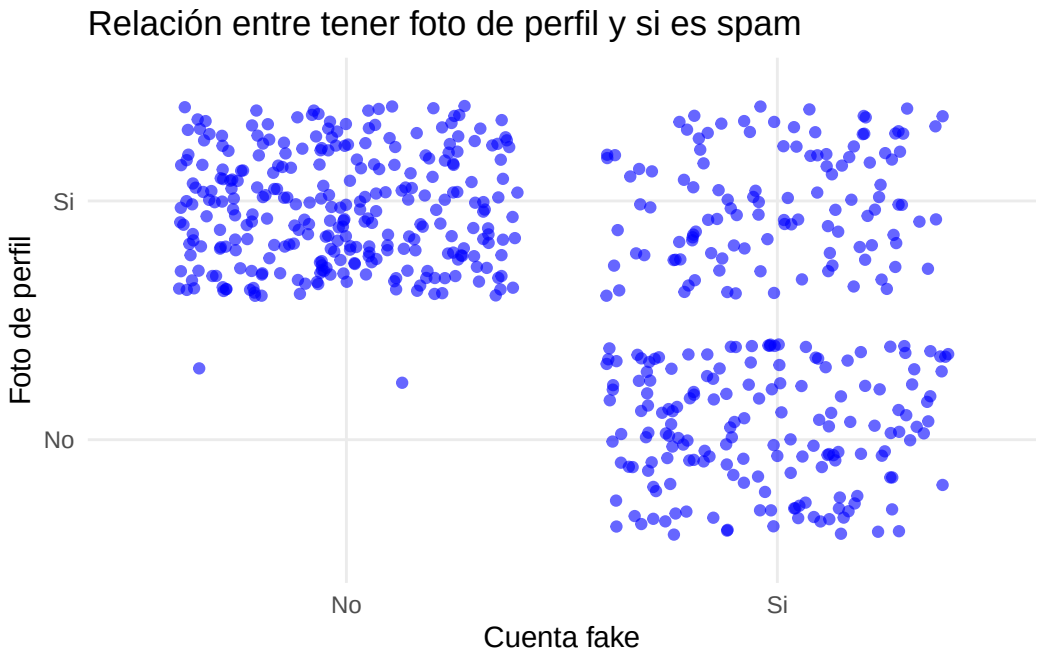
```
ggplot(datos_refinados, aes(x = `fake`, y = `private`)) +
  geom_count(color = "blue", alpha = 0.6) +
  scale_size_area()+
  labs(title = "Relación entre visibilidad del perfil y si es spam",
       x = "Cuenta fake",
       y = "Cuenta privada")+
  theme_minimal()
```

Utilizando `geom_count` con dos variables discretas, en este caso si un perfil es privado o no y si un perfil es fake o no, no podemos extraer mucha información relevante ya que vemos que hay aproximadamente un numero similar de cada combinación.

2.4 Relación entre tener foto de perfil y ser cuenta fake

Al igual que antes vamos a comprobar dos variables discretas, por lo que el aspecto del gráfico sera diferente. Vamos a comprobar si tener o no foto de perfil tiene algo de relación con ser un spammer.

```
ggplot(datos_refinados, aes(x = `fake`, y = `profile pic`)) +
  geom_jitter(color = "blue", alpha = 0.6) +
  scale_size_area()+
  labs(title = "Relación entre tener foto de perfil y si es spam",
       x = "Cuenta fake",
       y = "Foto de perfil")+
  theme_minimal()
```

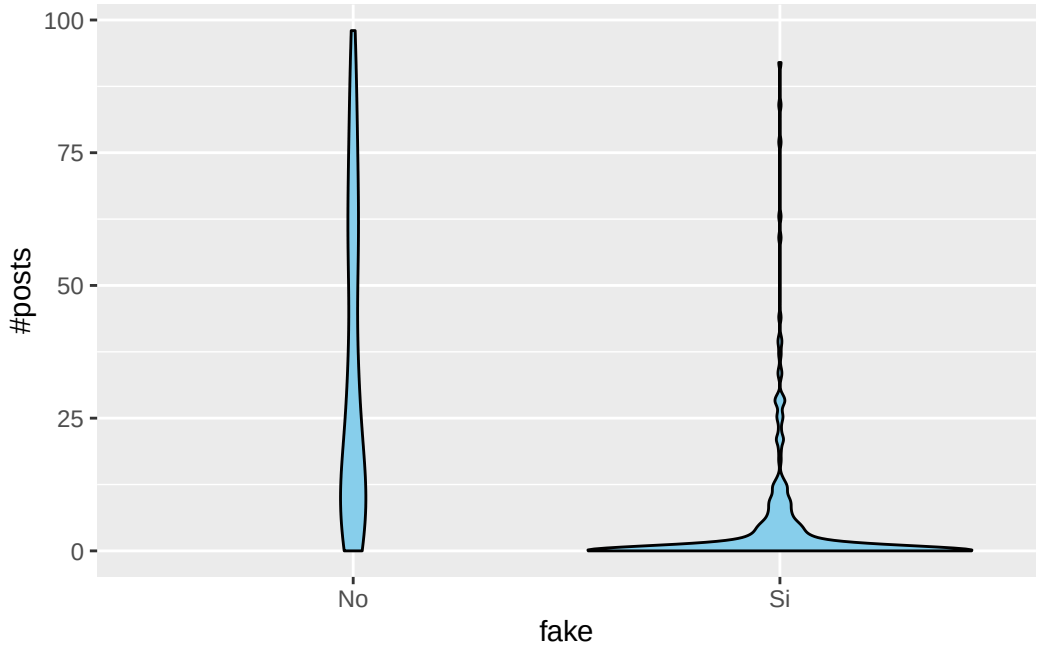


Hemos obtenido un resultado interesante, donde vemos que las cuentas reales, todas menos 2 tienen foto de perfil puesta, mientras que las cuentas fake hay mas o menos un mismo numero con foto de perfil y sin foto de perfil. Estos datos, combinados con otros que vamos a obtener mas adelante, nos pueden ayudar a diferenciar cuentas reales de falsas.

2.5 Relación entre numero de publicaciones y cuentas fake

Podemos suponer una posible hipótesis en la que los usuarios spammers, cuya tarea puede ser solo generar comentarios o likes, van a tener cuentas con menos numero de publicaciones que una cuenta de una persona verdadera. Vamos a visualizar esta idea:

```
posts_filtrados <- datos_refinados %>%  filter(`#posts` <100)
ggplot(posts_filtrados, aes(x = fake, y = `#posts`)) +
  geom_violin(fill = "skyblue", color = "black")
```



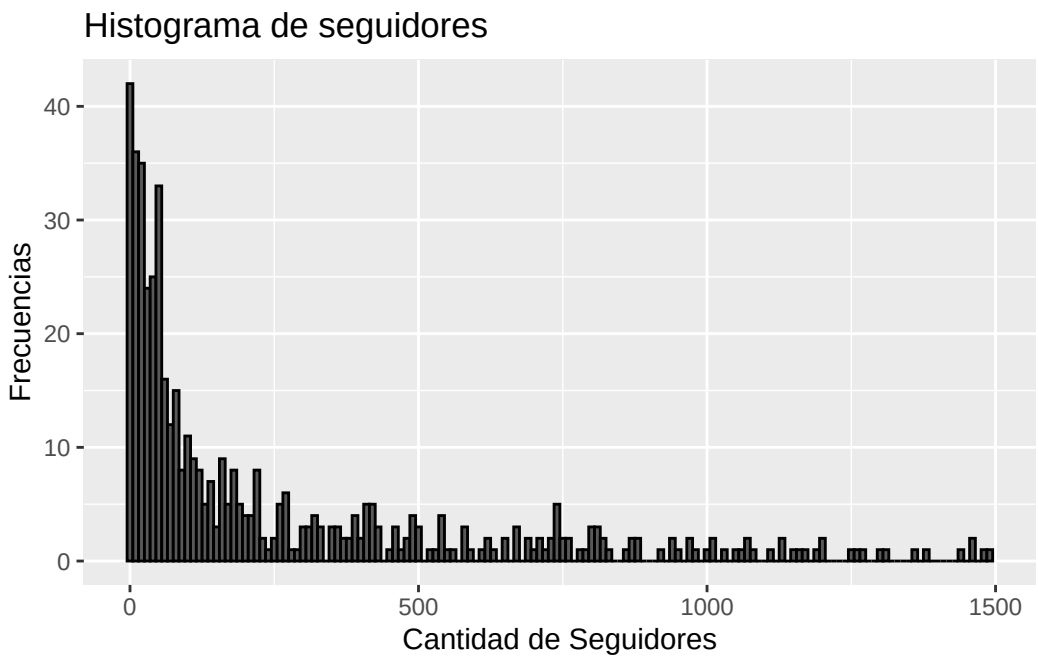
Observamos que teníamos razón, despues de eliminar aquellas cuentas con muchos post, vemos que las cuentas falsas suelen tener un numero reducido de publicaciones, mientras que las cuentas normales suelen tener una distribución mas uniforme.

2.6 Análisis de numero de seguidores

Unos de los atributos mas relevantes pueden ser el numero de seguidores el numero de seguidos, por lo quenecesitamos analizarlos en profundidad. Vamos a comenzar con el numero de seguidores.

Primero, como en el análisis exploratorio observamos que había algunas cuentas con muchos seguidores pero que no representaban un numero importante, vamos a eliminar esas escasas cuentas con un numero alto de seguidores con el fin de que los gráficos sean mas entendibles.

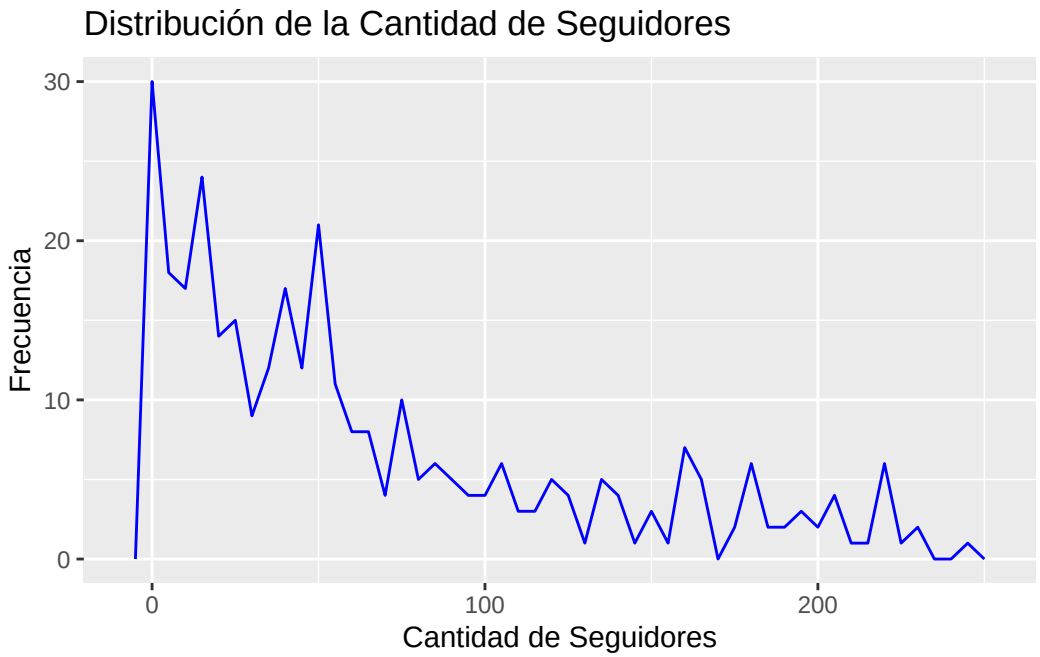
```
followers_filtrados <- datos_refinados %>% filter(`#followers` <1500)

ggplot(followers_filtrados, aes(x = `#followers`)) +
  geom_histogram(binwidth = 10, color = 'black') +
    labs(title = "Histograma de seguidores",
       x = "Cantidad de Seguidores",
       y = "Frecuencias")
```



Vemos que se concentra la mayoría en menos de 250 seguidores.

Vamos a utilizar una gráfica de frecuencia para ver como son nuestros datos con menos de 250 seguidores.

```
followers_filtrados <- datos_refinados %>%  filter(`#followers` <250)

ggplot(followers_filtrados, aes(x = `#followers`)) +
  geom_freqpoly(color = "blue", binwidth = 5) +
  labs(title = "Distribución de la Cantidad de Seguidores",
       x = "Cantidad de Seguidores",
       y = "Frecuencia")
```



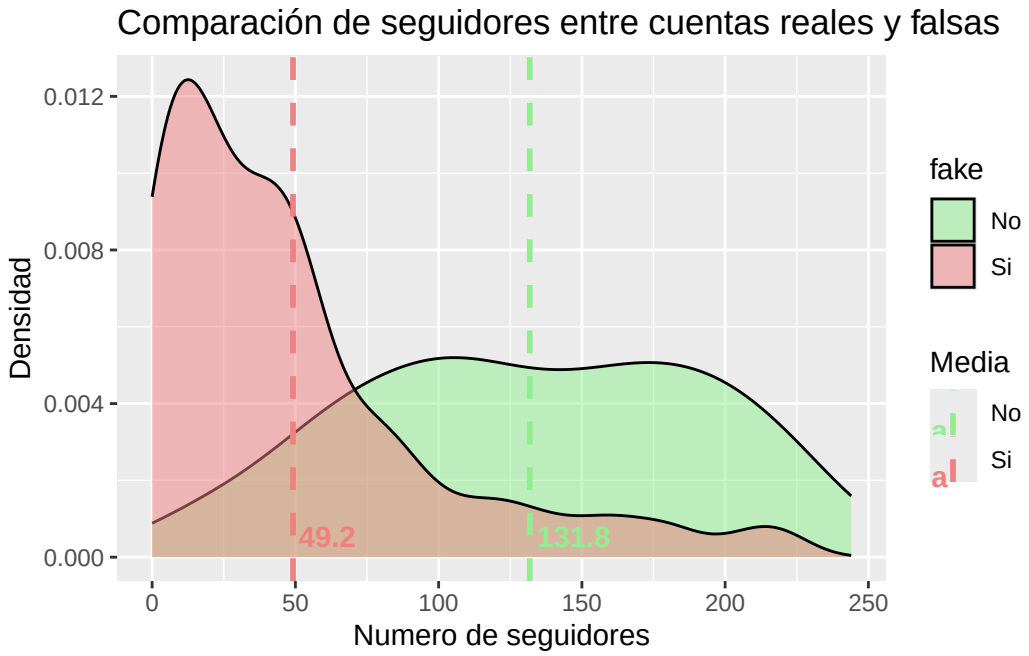
La mayor concentración se encuentra en menos de 100 seguidores y que es decreciente la frecuencia a medida que aumenta el numero de seguidores.

2.6.1 Comparación del numero de seguidores entre cuentas reales y falsas

Como nuestro principal objetivo es poder encontrar características similares que tengan las cuentas falsas para poder encontrarlas fácilmente, vamos a visualizar este atributo en relación con el numero de seguidores. Ademas añadimos las medias para obtener mas información.

```
mean_values <- followers_filtrados %>%
  group_by(fake) %>%
  summarize(mean_followerss = mean(`#followers`))

ggplot(data = followers_filtrados, aes( x = `#followers`,fill = `fake`)) +   geom_density(alp
  labs(title = "Comparación de seguidores entre cuentas reales y falsas",
       x = "Numero de seguidores",
       y = "Densidad") +
  scale_fill_manual(values = c("lightgreen", "lightcoral"))+
  geom_vline(data = mean_values, aes(xintercept = mean_followerss, color = fake), linetype =
   geom_text(data = mean_values, aes(x = mean_followerss, y = 0, label = round(mean_followers
             vjust = -0.5, hjust = -0.1, size = 4, fontface = "bold") +
  scale_color_manual(values = c("lightgreen", "lightcoral"), name = "Media")
```

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```



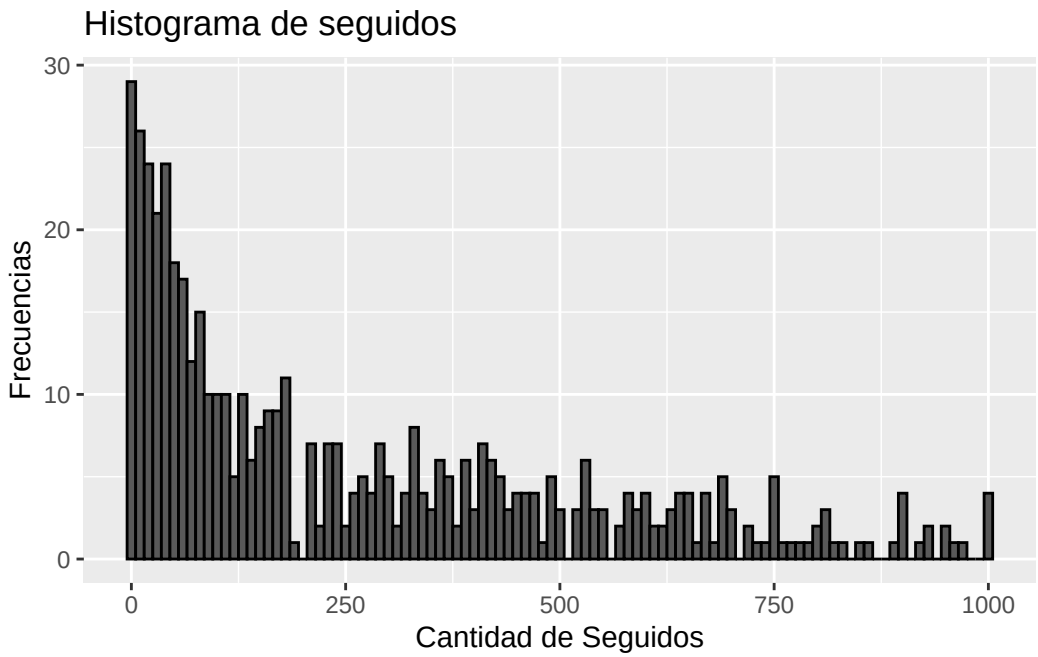
Aquí y obtenemos información mas interesante. Podemos observar que las cuentas falsas tienen a tener un menor numero de seguidores, mientras que las cuentas reales, aunque no tienen muchos seguidores, se suelen mantener en un intervalo entre 50 y 250. Esta información nos puede ser de importancia para los cálculos futuros.

2.7 Análisis de numero de seguidos

Ahora que hemos ya explorados gracias a varios gráficos como se comporta el numero de seguidores segun el tipo de cuentas, vamos a continuar ahora con el numero de seguidos.

Primero, como en el análisis exploratorio observamos que había algunas cuentas con muchos seguidos, pero que no representaban un numero importante, vamos a eliminar esas escasas cuentas con un numero alto de seguidos con el fin de que los gráficos sean mas entendibles.

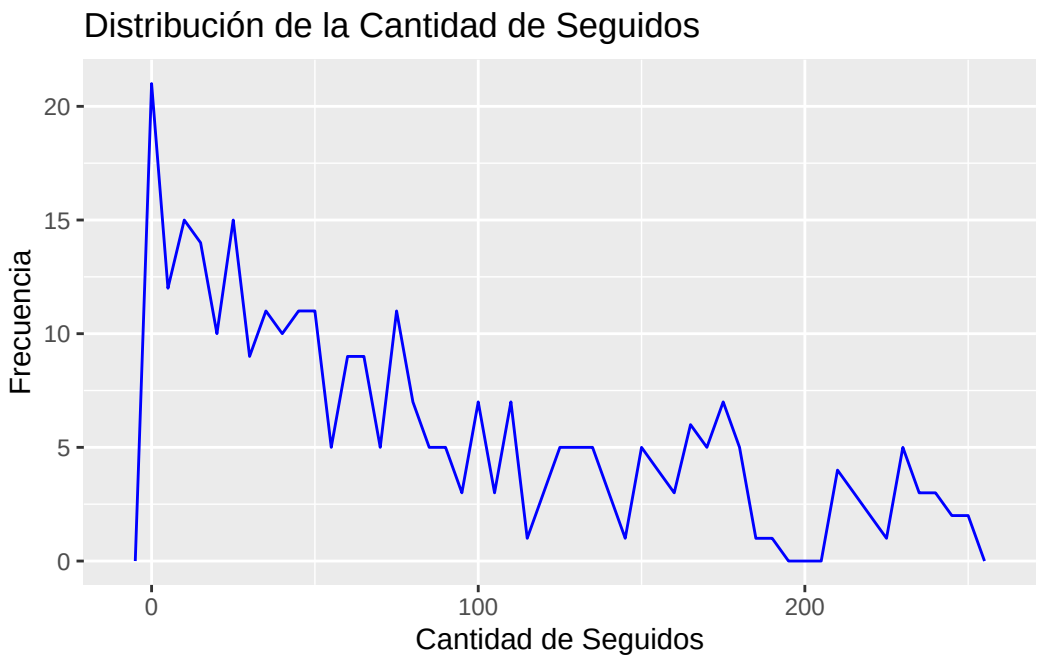
```
follows_filtrados <- datos_refinados %>%  filter(`#follows` <1000)
ggplot(follows_filtrados, aes(x = `#follows`)) +   geom_histogram(binwidth = 10, color = 'bla
  labs(title = "Histograma de seguidos",
       x = "Cantidad de Seguidos",
       y = "Frecuencias")
```



Vemos que se concentra la mayoría en menos de 250 seguidos.

Vamos a utilizar una gráfica de frecuencia para ver como son nuestros datos con menos de 250 seguidos.

```
follows_filtrados <- datos_refinados %>%  filter(`#follows` <250)
ggplot(follows_filtrados, aes(x = `#follows`)) +
  geom_freqpoly(color = "blue", binwidth = 5) +
  labs(title = "Distribución de la Cantidad de Seguidos",
       x = "Cantidad de Seguidos",
       y = "Frecuencia")
```



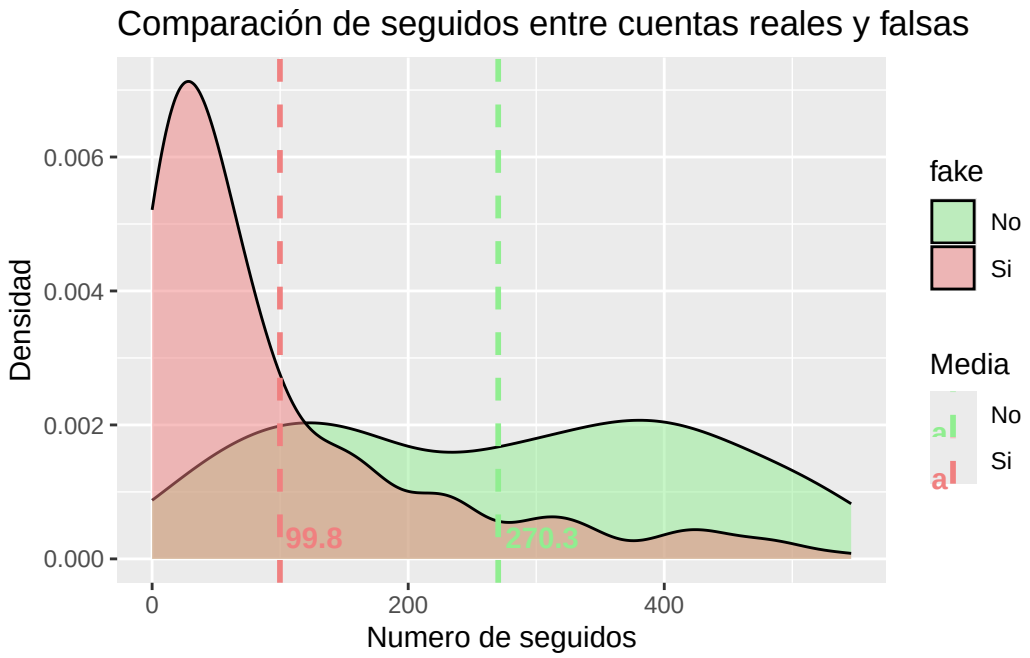
La mayor concentración se encuentra en menos de 100 seguidos y que es decreciente la frecuencia a medida que aumenta el numero de seguidores.

2.7.1 Comparación del numero de seguidos entre cuentas reales y falsas

Como nuestro principal objetivo es poder encontrar características similares que tengan las cuentas falsas para poder encontrarlas fácilmente, vamos a visualizar este atributo. Ademas añadimos las medias para obtener mas información.

```
follows_filtrados <- datos_refinados %>%  filter(`#follows` <550)

mean_values <- follows_filtrados %>%
  group_by(fake) %>%
  summarize(mean_follows = mean(`#follows`))

ggplot(data = follows_filtrados, aes( x = `#follows`,fill = `fake`)) +   geom_density(alpha =
  labs(title = "Comparación de seguidos entre cuentas reales y falsas",
       x = "Numero de seguidos",
       y = "Densidad") +
  scale_fill_manual(values = c("lightgreen", "lightcoral"))+
  geom_vline(data = mean_values, aes(xintercept = mean_follows, color = fake), linetype = "da
   geom_text(data = mean_values, aes(x = mean_follows, y = 0, label = round(mean_follows, 1)
```

```
            vjust = -0.5, hjust = -0.1, size = 4, fontface = "bold") +
  scale_color_manual(values = c("lightgreen", "lightcoral"), name = "Media")
```



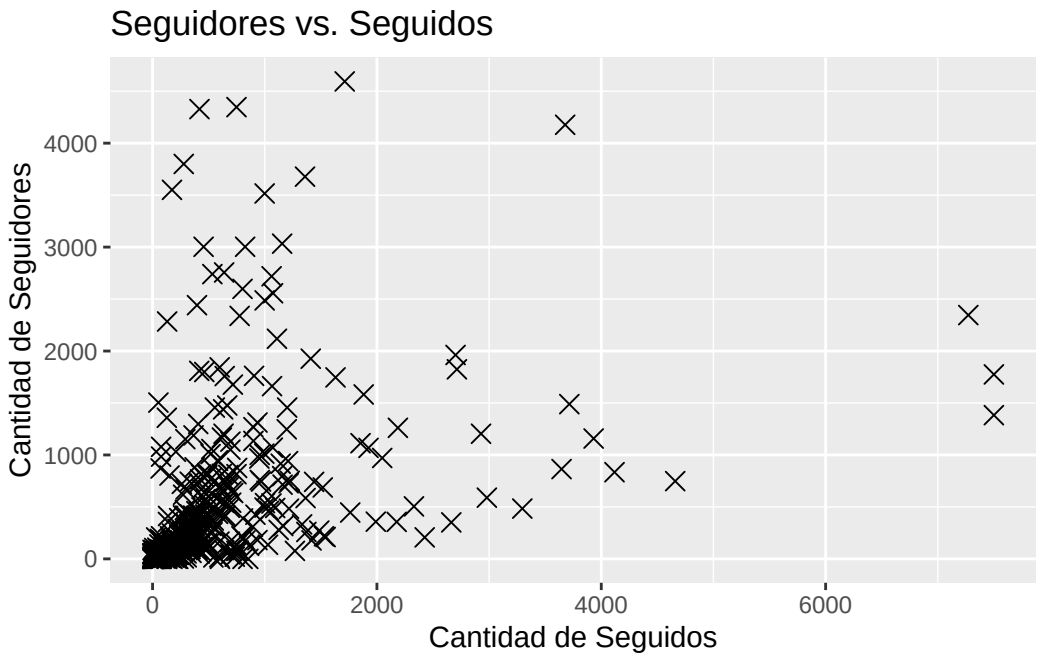
Aquí, al igual que con lo seguidores, obtenemos información mas interesante. Podemos observar que las cuentas falsas tienen a tener un menor numero de seguidores, pero no tan cercano al 0, mientras que las cuentas reales, suelen tener un numero mas repartido de seguidos. Esta información nos puede ser de importancia para los cálculos futuros.

2.8 Relación entre numero de seguidores y numero de seguidos

Ahora que la hemos visto ambas variables por separado, vamos a utilizar los gráficos de puntos o dispersión para ver varias variables juntas para intentar ver alguna relación o característica en estas.

```
followers_filtrados <- datos_refinados %>%  filter(`#followers` <5000)


ggplot(data = followers_filtrados, aes(x = `#follows`, y = `#followers`)) +
  geom_point(shape = 4, size = 3) +
  labs(title = "Seguidores vs. Seguidos",
       x = "Cantidad de Seguidos",
```

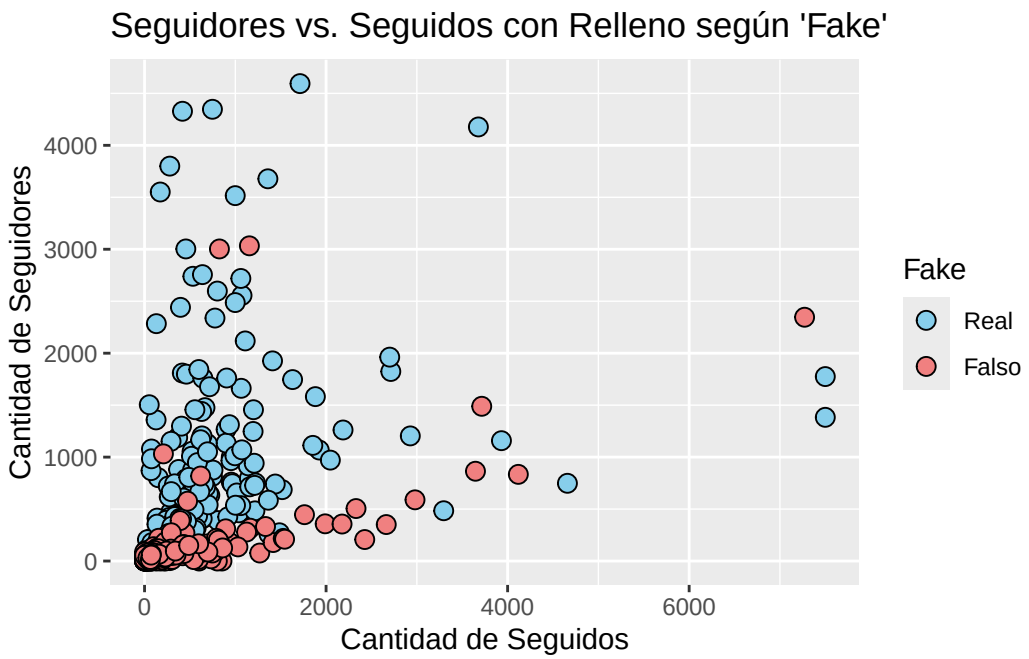
```
       y = "Cantidad de Seguidores")+
  scale_fill_manual(values = c("skyblue", "lightcoral"))
```



Viendo este gráfico solo podemos obtener que casi todo se concentra a un numero reducido tanto de seguidos y seguidores.

Aunque dicha información no nos sirve de mucho, vamos a añadir el parametro para diferenciar cuentas fake y reales. Podemos pensar que los seguidores y los seguidos tienen algo de relación con los usuarios que son fake, vamos a refinar un poco el DataSet eliminando los usuarios que tenian muchos seguidores, vamos a investigar:

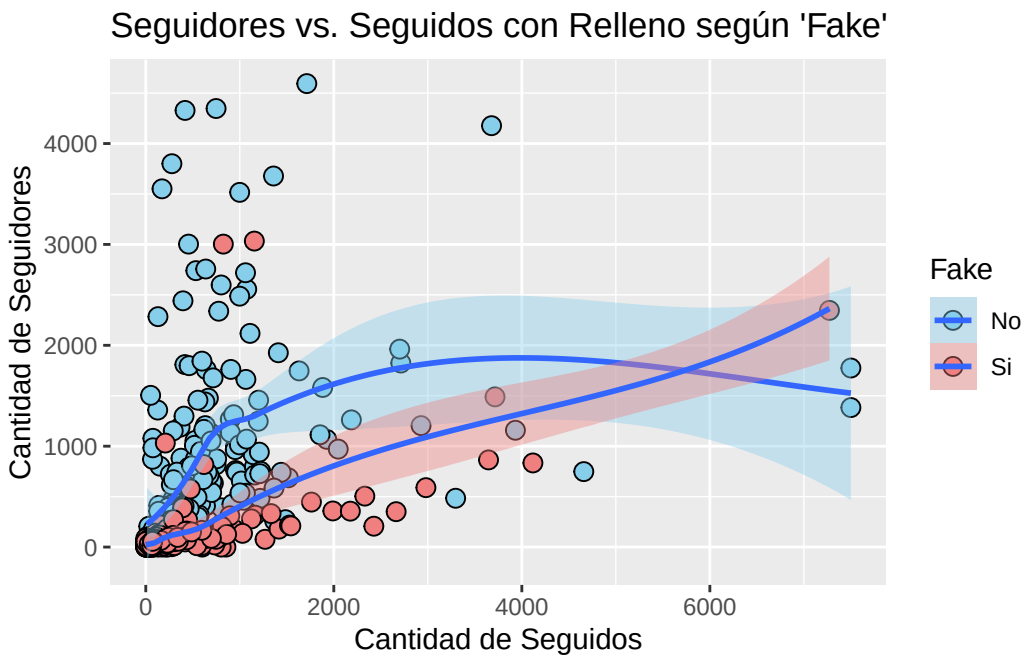
```
followers_filtrados <- datos_refinados %>%  filter(`#followers` <5000)


ggplot(data = followers_filtrados, aes(x = `#follows`, y = `#followers`, fill =fake)) +
  geom_point(shape = 21, size = 3) +
  labs(title = "Seguidores vs. Seguidos con Relleno según 'Fake'",
       x = "Cantidad de Seguidos",
       y = "Cantidad de Seguidores",
       fill = "Fake") +
  scale_fill_manual(values = c("skyblue", "lightcoral"), labels = c("Real", "Falso"))
```



Aquí podemos ver que hay una cierta tendencia. Las cuentas fake suelen tener mas cuentas seguidas que seguidores .Esto puede ser debido a que al ser cuentas generadas automaticamente, seguir a otra cuenta es una tarea que se puede automatizar, mientras que conseguir seguidores es algo mas complicado y requiere que de una segunda persona para que le siga. Vamos a utilizar el atributo de `geom_smooth` para poder visualizar una posible tendencia

```
ggplot(data = followers_filtrados, aes(x = `#follows`, y = `#followers`, fill = fake)) +
  geom_point(shape = 21, size = 3) +
   geom_smooth(method = "loess")+
  labs(title = "Seguidores vs. Seguidos con Relleno según 'Fake'",
       x = "Cantidad de Seguidos",
       y = "Cantidad de Seguidores",
       fill = "Fake") +
  scale_fill_manual(values = c("skyblue", "lightcoral") )
```

```
`geom_smooth()` using formula = 'y ~ x'
```



Ahora podemos reafirmar la idea de esa posible tendencia gracias a este gráfico. Vemos que los puntos rojos (fake) se ajustan en a la linea roja. Sin embargo las cuentas verdaderas tiene una tendencia mas dispersa.

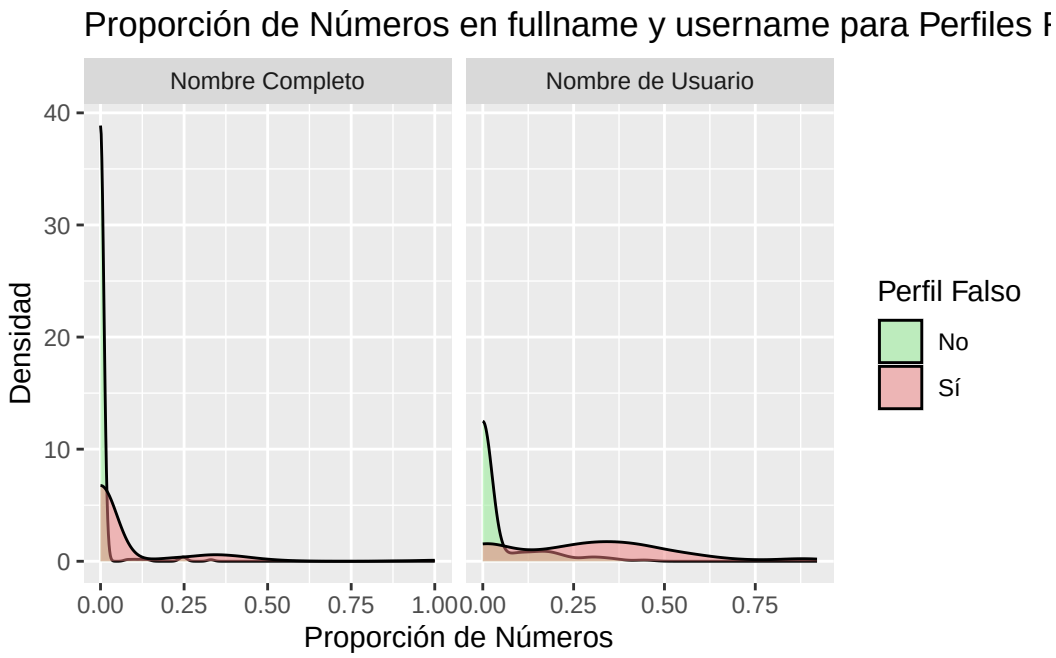
2.9 Importancia presencia de caracteres numéricos en usuario y nombre

Encontrar caracteres numéricos en nombre de usuario y nombres completos es algo que de primeras no podemos asociar a ningún tipo de cuenta, por lo tanto, nos vemos en la necesidad de analizarlo mas en profundidad

```
#Tenemos que duplicar los datos para poder poner una grafica al lado de otra
#Pivot_longer elimina las columnas conbinandola en dos columnas con el nombre y el valor
library(tidyr)
```

```
Attaching package: 'tidyr'

The following object is masked from 'package:magrittr':

    extract
```

```r
datos_comb  <-  datos_refinados %>%
  pivot_longer(cols = c(`nums/length fullname`, `nums/length username`),
               names_to = "variable",
               values_to = "value")

ggplot(data = datos_comb, aes(x = value, fill = fake)) +
  geom_density(alpha = 0.5, adjust = 1) +
  labs(title = "Proporción de Números en fullname y username para Perfiles Reales y Falsos",
       x = "Proporción de Números",
       y = "Densidad",
       fill = "Perfil Falso") +
  scale_fill_manual(values = c("lightgreen", "lightcoral"), labels = c("No", "Sí")) +  facet_
```



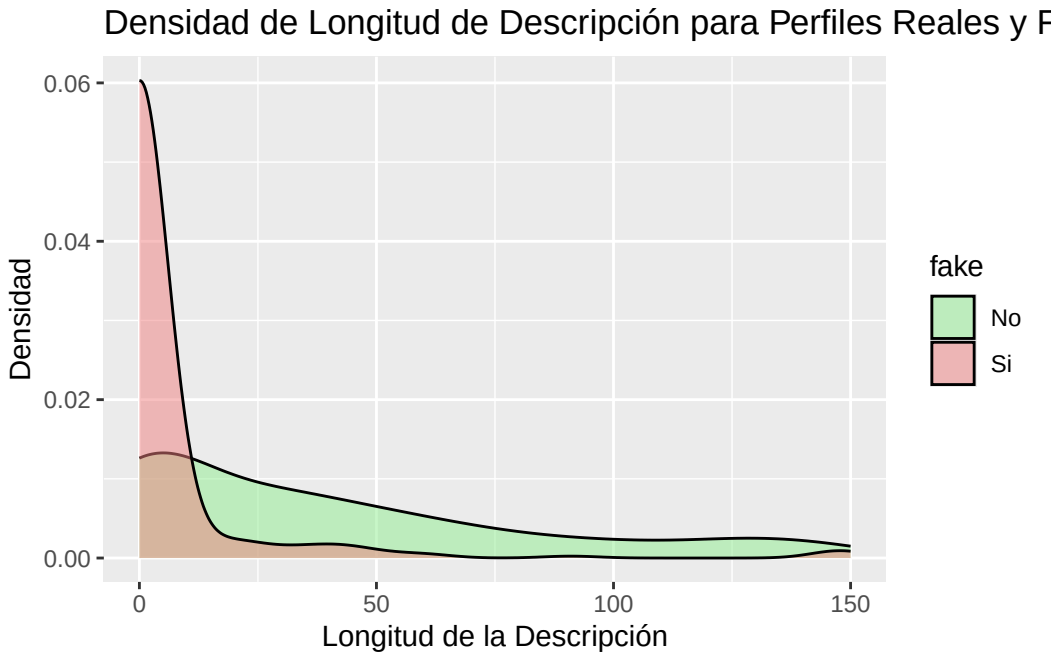
Podemos ver que realmente si hay una relación entre la presencia de caracteres numéricos en el nombre y el nombre de usuario respecto a si la cuenta es verdadera o spammer.

Podemos concluir que la cuentas falsas suelen contener mayor numero de caracteres numéricos en el nombre o nombre de usuario que las cuentas verdaderas.

2.10 Relación entre longitud de la descripción para perfiles reales y perfiles falsos

Por ultimo, otra posible hipótesis posible puede ser pensar que los usuarios falsos, tienen descripciones vacías o menos elaboradas que las de los perfiles reales.

```
ggplot(data = datos_refinados, aes(x = `description length`, fill = fake)) +
  geom_density(alpha = 0.5) +
  labs(title = "Densidad de Longitud de Descripción para Perfiles Reales y Falsos",
       x = "Longitud de la Descripción",
       y = "Densidad") +
  scale_fill_manual(values = c("lightgreen", "lightcoral"))
```

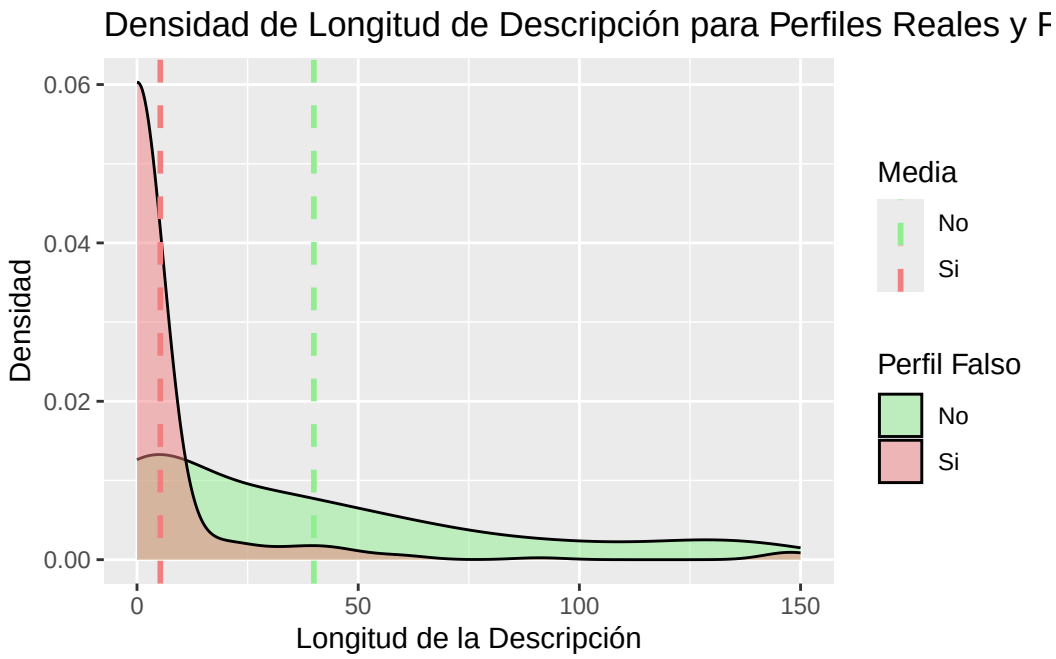


Y podemos comprobar que dicha idea era cierta, los perfiles falsos suelen tener un numero reducido de caracteres en su descripción, mientras que los reales esta mas repartidos.

Vamos a visualizar las medias:

```
mean_values <- datos_refinados %>%
  group_by(fake) %>%
  summarize(mean_desc_length = mean(`description length`))

ggplot(data = datos_refinados, aes(x = `description length`, fill =fake)) +
```

```r
  geom_density(alpha = 0.5) +
  labs(title = "Densidad de Longitud de Descripción para Perfiles Reales y Falsos",
       x = "Longitud de la Descripción",
       y = "Densidad",
       fill = "Perfil Falso") +
  scale_fill_manual(values = c("lightgreen", "lightcoral"))+
  geom_vline(data = mean_values, aes(xintercept = mean_desc_length, color = fake), linetype =
  scale_color_manual(values = c("lightgreen", "lightcoral"), name = "Media")
```



2.11 Conclusiones

Vistos todos los gráficos anteriores, podemos sacar algunas conclusiones interesantes:

1. Las cuentas falsas tienen menor numero de seguidores y mayor seguidos.
2. Las cuentas reales tienen descripciones con longitudes mas largas.
3. Las cuentas falsas tiene mas cantidad de caracteres numéricos en el nombre completo y nombre de usuario.
4. Las cuentas reales siempre suelen tener foto de perfil.

3 Reglas de asociación

Vamos a utilizar reglas de asociación para detectar cuentas falsas en Instagram. Las reglas de asociación son técnicas de minería de datos que permiten descubrir relaciones interesantes y útiles entre diferentes características o comportamientos observados en el dataSet.

Para realizar estas operaciones vamos a utilizar el paquete `arules`.

3.1 Características importantes

3.1.1 Medidas relevantes

Para evaluar la calidad y relevancia de las reglas de asociación, vamos a utilizar las medidas de:

1. **Soporte (Support)**:
    - El soporte de una regla mide la proporción de cuentas en el dataset que contienen ambos conjuntos de características A y B.
    - Un soporte alto indica que la regla se aplica a una gran proporción del dataset, lo que sugiere que la combinación de características es común y relevante.
2. **Confianza (Confidence)**:
    - La confianza de una regla mide cuán frecuentemente las características en B aparecen en las cuentas que contienen A.
    - A mayor confianza, mayor es la fiabilidad de que la presencia de las características en el antecedente de la regla (A) implicará la presencia de las características en el consecuente de la regla (B).
3. **Elevación (Lift)**:
    - La elevación mide la relación entre la aparición conjunta de A y B y la aparición esperada de A y B si fueran independientes.
    - Una elevación alta (mayor que 1) indica que la presencia de A incrementa significativamente la probabilidad de que B ocurra, lo que sugiere una fuerte asociación entre las características.

3.1.2 Algoritmo Apriori

Vamos a utilizar el algoritmo Apriori como nuestra forma de obtener reglas a partir de nuestros datos. Este es uno de los algoritmos más conocidos y se basa en la propiedad de que cualquier subconjunto de un conjunto frecuente también debe ser frecuente. El algoritmo itera a través de los conjuntos de características, incrementando su tamaño en cada iteración y manteniendo solo los conjuntos que cumplen con un umbral mínimo de soporte.

3.1.3 Reglas

Las reglas de asociación consisten en implicaciones del tipo "Si A entonces B", donde A y B son conjuntos de características o comportamientos de las cuentas. Por ejemplo, una regla podría ser "Si una cuenta tiene un número alto de cuentas seguidos y no tiene foto de perfil, entonces es probable que sea una cuenta falsa".

3.2 Cargar datos

Vamos a cargar las librería necesarias y nuestro dataSet:

```
library(arules)
```

```
Loading required package: Matrix
```

```

Attaching package: 'arules'
```

```
The following objects are masked from 'package:base':

    abbreviate, write
```

```
library(arulesViz)
```

```
Warning: package 'arulesViz' was built under R version 4.3.3
```

```
library(readr)
datos <- read_csv("Data/train.csv")
```

```
Rows: 576 Columns: 12
```

```
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl (12): profile pic, nums/length username, fullname words, nums/length ful...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

3.3 Discretizar datos

Puesto que el algoritmo de apriori necesita que el conjunto de datos sea binario o discreto.

Existen varias formas de discretizar datos, pero el objetivo principal es convertir las características continuas en valores discretos que representen de manera efectiva la información subyacente. Algunas técnicas comunes de discretización incluyen la binarización, la división en intervalos fijos o basados en cuantiles.

Tras haber realizado el previo análisis exploratorio podemos definir intervalos personalizados para las cada variable, para ello usaremos las funciones ordered y cut. Ademas las variables que son binarias como fake, vamos a ponerle "Si" o "No" para poder comprenderlo mejor.

```r
datos_refinados <- datos

columnas_binarias = c("profile pic","name==username","external URL","fake","private")

for (columna in columnas_binarias) {
  datos_refinados[[columna]] <-  factor(datos_refinados[[columna]], labels = c("No", "Si"))
}
```

```r
# Discretización de la columna #posts
datos_refinados$`#posts` <- ordered(cut(datos_refinados$`#posts`,
                                  breaks = c(0,1, 5, 10, 50, Inf),
                                  labels = c("muy bajo","medio", "alto", "muy alto", "extremad

# Discretización de la columna #followers
datos_refinados$`#followers` <- ordered(cut(datos_refinados$`#followers`,
                                      breaks = c(0, 10, 60, 200, Inf),
                                      labels = c("bajo", "medio", "alto", "muy alto"),include

# Discretización de la columna #follows
datos_refinados$`#follows` <- ordered(cut(datos_refinados$`#follows`,
                                    breaks = c(0, 10, 60, 200, Inf),
                                    labels = c("bajo", "medio", "alto", "muy alto"),include.lo
```

```
# Discretización de la columna nums/length username
datos_refinados$`nums/length username` <- ordered(cut(datos_refinados$`nums/length username`
                                                      breaks = c(0, 0.2, 0.4, 0.6, 0.8, 1),
                                                      labels = c("muy bajo", "bajo", "medio"
                                                      include.lowest = TRUE))

# Discretización de la columna nums/length fullname
datos_refinados$`nums/length fullname` <- ordered(cut(datos_refinados$`nums/length fullname`
                                                      breaks = c(0, 0.2, 0.4, 0.6, 0.8, 1),
                                                      labels = c("muy bajo", "bajo", "medio"
                                                      include.lowest = TRUE))
# Discretización de la columna description length
datos_refinados$`description length` <- ordered(cut(datos_refinados$`description length`,
                                                    breaks = c(0, 15, 25, 80, 150),
                                                    labels = c("muy corto" , "medio", "largo
                                                   include.lowest = TRUE))
# Discretización de la columna fullname words
datos_refinados$`fullname words` <- ordered(cut(datos_refinados$`fullname words`,
                                                  breaks = c(0, 1, 3, 5, Inf),
                                                  labels = c("muy corto", "medio", "largo", "
```

3.3.1 discretizeDF

Esta función del paquete de arules implementa varios métodos básicos no supervisados para convertir una variable continua en una variable categórica (factor) usando diferentes estrategias de agrupamiento.

Vamos a quitar primero las columnas binarias que le queremos poner un valor custom.

```
datos_refinados_clone <- datos

columnas_binarias = c("profile pic","name==username","external URL","fake","private")

for (columna in columnas_binarias) {
  datos_refinados_clone[[columna]] <-  factor(datos_refinados_clone[[columna]], labels = c("
}
```

Vamos a ver algunas estrategias:

3.3.1.1 K-means:

```
kmeansDisc <- discretizeDF(datos_refinados_clone, default = list(method = "cluster", breaks =
  labels = c("muy bajo", "bajo","medio","alto","muy alto")))
head(kmeansDisc)
```

```
# A tibble: 6 x 12
  `profile pic` `nums/length username` `fullname words` `nums/length fullname`
  <fct>         <fct>                  <fct>            <fct>
1 Si            bajo                   muy bajo         muy bajo
2 Si            muy bajo               bajo             muy bajo
3 Si            muy bajo               bajo             muy bajo
4 Si            muy bajo               bajo             muy bajo
5 Si            muy bajo               bajo             muy bajo
6 Si            muy bajo               medio            muy bajo
# i 8 more variables: `name==username` <fct>, `description length` <fct>,
#   `external URL` <fct>, private <fct>, `#posts` <fct>, `#followers` <fct>,
#   `#follows` <fct>, fake <fct>
```

3.3.1.2 interval

```
fixedDisc <- discretizeDF(datos_refinados_clone, default = list(method = "interval", breaks =
  labels = c("muy bajo", "bajo","medio","alto","muy alto")))
head(fixedDisc)
```

```
# A tibble: 6 x 12
  `profile pic` `nums/length username` `fullname words` `nums/length fullname`
  <fct>         <fct>                  <fct>            <fct>
1 Si            bajo                   muy bajo         muy bajo
2 Si            muy bajo               muy bajo         muy bajo
3 Si            muy bajo               muy bajo         muy bajo
4 Si            muy bajo               muy bajo         muy bajo
5 Si            muy bajo               muy bajo         muy bajo
6 Si            muy bajo               bajo             muy bajo
# i 8 more variables: `name==username` <fct>, `description length` <fct>,
#   `external URL` <fct>, private <fct>, `#posts` <fct>, `#followers` <fct>,
#   `#follows` <fct>, fake <fct>
```

3.4 Generar dataset de transacciones

Ahora una vez discretizado el dataframe, el siguiente paso es generar un dataset de transacciones. Este tipo de dataset es esencial para aplicar algoritmos de reglas de asociación como Apriori.

En un dataset de transacciones, cada fila representa una transacción, que es una colección de elementos o ítems.

```
datos_refinadosT <- as(datos_refinados, "transactions")
```

3.5 Generar reglas

Ahora que ya tenemos todo listo, podemos utilizar los algoritmos de generacion de reglas. En nuestro caso, vamos a utilizar apriori. Para generar reglas primero necesitamos establecer un valor para el suport y confianza minima, estos valores nos permitirán controlar la cantidad y calidad de las reglas que se generarán.

```
 rules <- apriori(datos_refinadosT,  parameter = list(supp = 0.3, conf = 0.01, target = "rule
```

```
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen
       0.01    0.1    1 none FALSE            TRUE       5     0.3      1
 maxlen target  ext
     10  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 172

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[40 item(s), 576 transaction(s)] done [0.00s].
sorting and recoding items ... [16 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 5 6 7 done [0.00s].
writing ... [1157 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

```
 rules
```

```
set of 1157 rules
```

Hemos obtenido una buena cantidad de reglas para continuar nuestro análisis.

3.6 Refinar reglas

Ahora que hemos obtenido las reglas necesitamos cribarlarlas y eliminar todas aquellas que no nos interesan, que sean redundantes o no significativas.

3.6.1 Eliminar reglas redundantes

```
rules <- rules[which(is.redundant(rules))]
```

3.6.2 Eliminar reglas no significativas

```
rules <- rules[which(is.significant(rules))]
```

Vamos a ver cuentas reglas han quedado después de filtrarlas:

```
length(rules)
```

```
[1] 569
```

3.7 Análisis de reglas obtenidas

Nuestro objetivo es detectar y diferenciar cuentas falses de las verdaderas, por lo tanto, vamos a centrar nuestro analisis en eso dos atributos “fake=Si” y “fake=No”. Como tenemos diferentes metricas, vamos a analizarlas por separado:

3.7.1 Support:

Vamos priemro a analizar las reglas ordenando primero por el soporte de la reglas. Recordamos que el soporte alto indica que la regla se aplica a una gran proporción del dataset, lo que sugiere que la combinación de características es común y relevante.

```
rules <- sort(rules,by="support")
inspect(head(rules))
```

```
    lhs                                   rhs                                support confidence
[1] {nums/length fullname=muy bajo,
     external URL=No}                  => {name==username=No}              0.7881944  0.9848156
[2] {nums/length username=muy bajo,
     name==username=No}                => {nums/length fullname=muy bajo} 0.6250000  0.9863014
[3] {nums/length fullname=muy bajo,
     name==username=No}                => {nums/length username=muy bajo} 0.6250000  0.6936416
[4] {name==username=No,
     description length=muy corto}     => {external URL=No}               0.6041667  0.9747899
[5] {name==username=No,
     external URL=No}                  => {description length=muy corto}  0.6041667  0.7102041
[6] {nums/length fullname=muy bajo,
     description length=muy corto}     => {external URL=No}               0.5590278  0.9728097
```

En este caso, el soporte es 0.7881944, lo que significa que el 78.82% de las transacciones en el dataset contienen tanto el antecedente {nums/length fullname=muy bajo, external URL=No} como el consecuente {name==username=No}.

```
r2 <- subset(rules, subset = rhs %in% c("fake=Si"))
inspect(head(r2))
```

```
    lhs                                   rhs         support confidence  coverage     lift cou
[1] {name==username=No,
     external URL=No}                  => {fake=Si} 0.4670139  0.5489796 0.8506944 1.097959   1
[2] {name==username=No,
     description length=muy corto}     => {fake=Si} 0.4253472  0.6862745 0.6197917 1.372549   1
[3] {name==username=No,
     description length=muy corto,
     external URL=No}                  => {fake=Si} 0.4253472  0.7040230 0.6041667 1.408046   1
[4] {nums/length fullname=muy bajo,
     external URL=No}                  => {fake=Si} 0.4218750  0.5271150 0.8003472 1.054230   1
[5] {nums/length fullname=muy bajo,
```

```
     description length=muy corto}   => {fake=Si} 0.3819444  0.6646526 0.5746528 1.329305   1
[6] {nums/length fullname=muy bajo,
     description length=muy corto,
     external URL=No}                => {fake=Si} 0.3819444  0.6832298 0.5590278 1.366460   1
```

3.7.2 Confianza:

Ahora va a analizar las reglas ordenando por la confianza en la reglas. Recordamos que a mayor confianza, mayor es la fiabilidad de que la presencia de las características en el antecedente de la regla A implicará la presencia de las características en el consecuente de la regla B.

```
rules <- sort(rules,by="confidence")
inspect(head(rules))
```

```
    lhs                                rhs                  support confidence  coverage
[1] {name==username=No,
     fake=Si}                       => {external URL=No}   0.4670139          1 0.4670139 1
[2] {description length=muy corto,
     fake=Si}                       => {external URL=No}   0.4531250          1 0.4531250 1
[3] {name==username=No,
     description length=muy corto,
     fake=Si}                       => {external URL=No}   0.4253472          1 0.4253472 1
[4] {nums/length fullname=muy bajo,
     fake=Si}                       => {external URL=No}   0.4218750          1 0.4218750 1
[5] {profile pic=Si,
     nums/length fullname=muy bajo,
     #followers=muy alto}           => {name==username=No} 0.4166667          1 0.4166667 1
[6] {nums/length fullname=muy bajo,
     name==username=No,
     fake=Si}                       => {external URL=No}   0.4097222          1 0.4097222 1
```

En este caso, la confianza es 1, lo que significa que el 100% de las transacciones que tienen el antecedente también tienen el consecuente.

```
r2 <- subset(rules, subset = rhs %in% c("fake=Si"))
inspect(head(r2))
```

```
    lhs                                rhs         support confidence  coverage     lift cou
[1] {external URL=No,
     #posts=muy bajo}               => {fake=Si} 0.3072917  0.9567568 0.3211806 1.913514   1
```

```
[2] {fullname words=muy corto,
     name==username=No,
     description length=muy corto,
     external URL=No}               => {fake=Si} 0.3454861  0.8122449 0.4253472 1.624490
[3] {fullname words=muy corto,
     name==username=No,
     description length=muy corto}  => {fake=Si} 0.3454861  0.8024194 0.4305556 1.604839
[4] {fullname words=muy corto,
     nums/length fullname=muy bajo,
     description length=muy corto,
     external URL=No}               => {fake=Si} 0.3072917  0.7972973 0.3854167 1.594595
[5] {fullname words=muy corto,
     nums/length fullname=muy bajo,
     description length=muy corto}  => {fake=Si} 0.3072917  0.7866667 0.3906250 1.573333
[6] {fullname words=muy corto,
     name==username=No,
     external URL=No}               => {fake=Si} 0.3750000  0.7105263 0.5277778 1.421053
```

3.7.3 Lift:

Por ultimo, vamos a analizar las reglas ordenando primero por el lift de la reglas. Recordamos que un lift alto indica que la presencia de A incrementa significativamente la probabilidad de que B ocurra, lo que sugiere una fuerte asociación entre las características.

```
rules <- sort(rules,by="lift")
inspect(head(rules))
```

```
    lhs                                rhs                     support confidence  coverage
[1] {name==username=No,
     #follows=muy alto,
     fake=No}                       => {#followers=muy alto} 0.3506944  0.9223744 0.3802083
[2] {profile pic=Si,
     #follows=muy alto,
     fake=No}                       => {#followers=muy alto} 0.3472222  0.9216590 0.3767361
[3] {profile pic=Si,
     name==username=No,
     #follows=muy alto,
     fake=No}                       => {#followers=muy alto} 0.3472222  0.9216590 0.3767361
[4] {nums/length fullname=muy bajo,
     #follows=muy alto,
     fake=No}                       => {#followers=muy alto} 0.3437500  0.9209302 0.3732639
```

```
[5] {nums/length fullname=muy bajo,
     name==username=No,
     #follows=muy alto,
     fake=No}                        => {#followers=muy alto} 0.3437500  0.9209302 0.3732639
[6] {nums/length username=muy bajo,
     #follows=muy alto,
     fake=No}                        => {#followers=muy alto} 0.3229167  0.9207921 0.3506944
```

En este caso, el lift es 5.731343, lo que sugiere que la aparicion de "external URL=Si" es aproximadamente 5.73 veces más probable cuando se dan las condiciones en el antecedente.

```
r2 <- subset(rules, subset = rhs %in% c("fake=Si"))
inspect(head(r2))
```

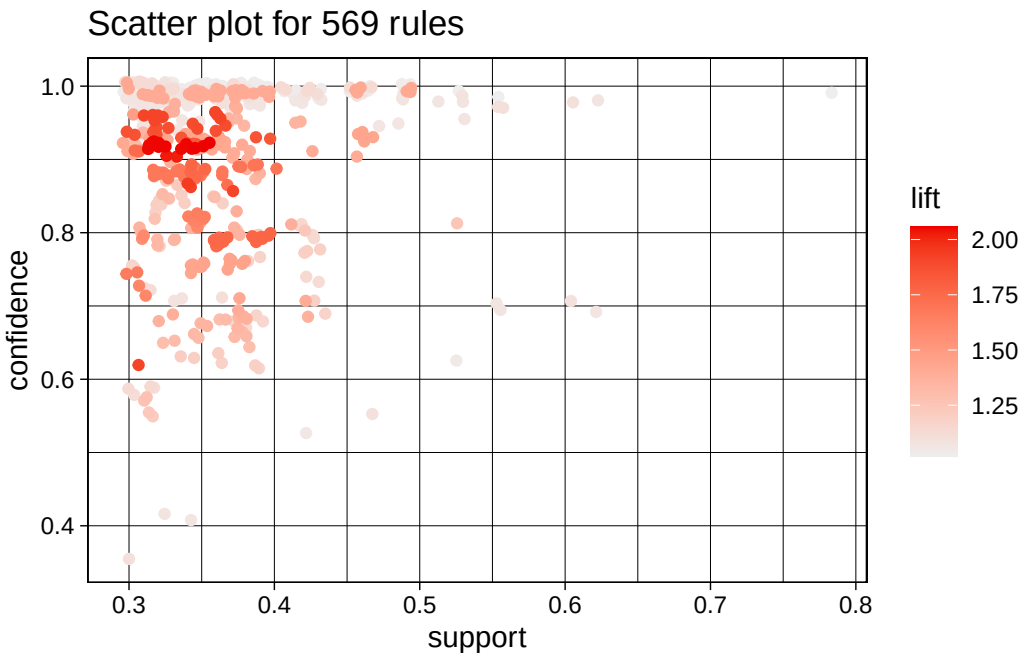
```
    lhs                                rhs         support confidence  coverage     lift cou
[1] {external URL=No,
     #posts=muy bajo}                => {fake=Si} 0.3072917  0.9567568 0.3211806 1.913514   1
[2] {fullname words=muy corto,
     name==username=No,
     description length=muy corto,
     external URL=No}                => {fake=Si} 0.3454861  0.8122449 0.4253472 1.624490   1
[3] {fullname words=muy corto,
     name==username=No,
     description length=muy corto}   => {fake=Si} 0.3454861  0.8024194 0.4305556 1.604839   1
[4] {fullname words=muy corto,
     nums/length fullname=muy bajo,
     description length=muy corto,
     external URL=No}                => {fake=Si} 0.3072917  0.7972973 0.3854167 1.594595   1
[5] {fullname words=muy corto,
     nums/length fullname=muy bajo,
     description length=muy corto}   => {fake=Si} 0.3072917  0.7866667 0.3906250 1.573333   1
[6] {fullname words=muy corto,
     name==username=No,
     external URL=No}                => {fake=Si} 0.3750000  0.7105263 0.5277778 1.421053   1
```

3.8

3.9 Visualizacion de reglas

```
plot(rules)
```

```
To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.
```



4 Formal Concept Analysis

El Formal Concept Analysis, o FCA, es una técnica de análisis de datos originada en la teoría de conjuntos formales, la lógica matemática y la teoría de retículos. Su objetivo principal es descubrir y representar estructuras conceptuales dentro de conjuntos de datos, especialmente conjuntos de datos que contienen información de tipo jerárquico o taxonómico.

Las pricipales aplicaiciones de FCA son la extracción de conocimiento, agrupamiento y clasificación, aprendizaje automático, conceptos, ontologías, reglas, reglas de asociación, implicaciones de atributos.

Para el FCA, nuestros datos se dividen en objetos y atributos. En nuestro dataSet, los objetos son las cuentas de usuario y los atributos son las culumnas como "Tiene foto de perfil, No tes fake, ….".

```
library(fcaR)
```

```
Warning: package 'fcaR' was built under R version 4.3.3
```

```
library(readr)
datos <- read_csv("Data/train.csv")
```

```
Rows: 576 Columns: 12
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl (12): profile pic, nums/length username, fullname words, nums/length ful...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
datos_refinados <- datos

columnas_binarias = c("profile pic","name==username","external URL","fake","private")

for (columna in columnas_binarias) {
  datos_refinados[[columna]] <-  factor(datos_refinados[[columna]], labels = c("No", "Si"))
```

```
}

fc_datos <- FormalContext$new(datos_refinados)
fc_datos
```

```
FormalContext with 576 objects and 12 attributes.
# A tibble: 576 x 12
   `profile pic` `nums/length username` `fullname words` `nums/length fullname`
   <fct>                          <dbl>            <dbl>                   <dbl>
 1 Si                              0.27                0                       0
 2 Si                              0                   2                       0
 3 Si                              0.1                 2                       0
 4 Si                              0                   1                       0
 5 Si                              0                   2                       0
 6 Si                              0                   4                       0
 7 Si                              0                   2                       0
 8 Si                              0                   2                       0
 9 Si                              0                   0                       0
10 Si                              0                   2                       0
# i 566 more rows
# i 8 more variables: `name==username` <fct>, `description length` <dbl>,
#   `external URL` <fct>, private <fct>, `#posts` <dbl>, `#followers` <dbl>,
#   `#follows` <dbl>, fake <fct>
```

4.1 Escalado

Como necesitamos que nuestro dataSet sea binario, necesitamos aplicarles tecniac como el escaldo para obtener el resultado deseado:

4.1.1 Escalado nominal

El escalado nominal se utiliza para atributos cuyos valores son excluyentes el uno del otro, como por ejemplo, los atributos que son Si y No.

```
fc_datos$scale("profile pic",type = "nominal",c("Si","No"))
fc_datos$scale("name==username",type = "nominal",c("Si","No"))
fc_datos$scale("fake",type = "nominal",c("Si","No"))
fc_datos$scale("private",type = "nominal",c("Si","No"))
fc_datos$scale("external URL",type = "nominal",c("Si","No"))
fc_datos
```

```
FormalContext with 576 objects and 17 attributes.
# A tibble: 576 x 17
   `profile pic = Si` `profile pic = No` `nums/length username` `fullname words`
                <dbl>              <dbl>                   <dbl>             <dbl>
 1                  1                  0                    0.27                 0
 2                  1                  0                    0                    2
 3                  1                  0                    0.1                  2
 4                  1                  0                    0                    1
 5                  1                  0                    0                    2
 6                  1                  0                    0                    4
 7                  1                  0                    0                    2
 8                  1                  0                    0                    2
 9                  1                  0                    0                    0
10                  1                  0                    0                    2
# i 566 more rows
# i 13 more variables: `nums/length fullname` <dbl>,
#   `name==username = Si` <dbl>, `name==username = No` <dbl>,
#   `description length` <dbl>, `external URL = Si` <dbl>,
#   `external URL = No` <dbl>, `private = Si` <dbl>, `private = No` <dbl>,
#   `#posts` <dbl>, `#followers` <dbl>, `#follows` <dbl>, `fake = Si` <dbl>,
#   `fake = No` <dbl>
```

4.1.2 Escalado intervalos

Como los demás datos son valores continuos, tenemos que utilizar un tipo de escalado distinto. Podemos utilizar modos como el ordinal, sin embargo, este nos generaría conceptos demasiados largos, por lo que el mejor modo a emplear para estos datos es el intervalo.

```
fc_datos$scale("nums/length username",
         type = "interval",
         values =c(0, 0.2, 0.4, 0.6, 0.8, 1)
         )

fc_datos$scale("nums/length fullname",
         type = "interval",
         values = c(0, 0.2, 0.4, 0.6, 0.8, 1)
        )

fc_datos$scale("fullname words",
         type = "interval",
         values = c(0, 1, 3, 5, Inf)
         )
```

```
fc_datos$scale("description length",
         type = "interval",
         values =c(0, 15, 25, 80, 150)
         )

fc_datos$scale("#posts",
         type = "interval",
         values =  c(0,1, 5, 10, 50, Inf)
         )

fc_datos$scale("#followers",
         type = "interval",
         values = c(0, 10, 60, 200, Inf)
         )

fc_datos$scale("#follows",
         type = "interval",
         values = c(0, 10, 60, 200, Inf)
         )
```

4.2 Conceptos

Una vez tenemos los datos en la forma que buscamos, podemos utilizar el paquete fcaR para generar conceptos.Los conceptos son componentes fundamentales que representan agrupaciones de objetos y atributos con una relación particular.

De manera formal, un concepto (,) se define como un par donde:

- es el conjunto de objetos (extensión) que tienen todos los atributos de .
- es el conjunto de atributos (intensión) que son poseídos por todos los objetos de .

4.2.1 Calculo de los conceptos del contexto

Para calcular los conceptos de nuestros datos utilizamos la función `find_concepts`

```
fc_datos$find_concepts()

fc_datos$concepts$size()
```

```
[1] 7008
```

Vemos que hemos obtenido un gran numero de conceptos, vamos a ver los primeros:

```
head(fc_datos$concepts)
```

```
A set of 6 concepts:
1: ({1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 2
2: ({1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 2
3: ({1, 2, 4, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 27, 31, 32, 34,
4: ({3, 5, 6, 7, 8, 25, 26, 28, 29, 30, 33, 39, 40, 47, 61, 63, 64, 73, 105, 110, 114, 121, 1
5: ({41, 44, 76, 102, 104, 205, 242, 245, 247, 259, 273, 281, 287, 290, 295, 297, 299, 300, 3
6: ({45, 166, 298, 301, 304, 308, 309, 316, 337, 341, 377, 409, 412, 418, 430, 456, 468, 469
```

Observamos un curioso resultado, vemos una gran cantidad de números, estos números representan los indices de las cuentas que tienen dichos atributos. Sin embargo, esta información no nos es útil. Vamos a calcular el extend al atributo "*fake = Si*", y vemos que nos devuelve los indices de todas las cuentas que son fake.

```
s1 <- Set$new(fc_datos$attributes)
s1$assign(fake = "Si")
fc_datos$extent(s1)
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
  23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41,
  42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60,
  61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
  80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98,
  99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114,
  115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129,
  130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144,
  145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159,
  160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174,
  175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189,
  190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204,
  205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219,
  220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234,
  235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249,
  250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264,
  265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279,
  280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294,
```

```
295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309,
310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324,
325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339,
340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354,
355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369,
370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384,
385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399,
400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414,
415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429,
430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444,
445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459,
460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474,
475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489,
490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504,
505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519,
520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534,
535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549,
550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564,
565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576}
```

4.3 Implicaciones

Las implicaciones son reglas derivadas de los datos que describen relaciones lógicas entre conjuntos de atributos. En FCA, las implicaciones se extraen a partir de los conceptos y se utilizan para describir las dependencias entre los atributos de manera formal.

Estas implicaciones las podemos ver como las reglas de asociación que obtuvimos anteriormente.

4.3.1 Calculo de los implicaciones del contexto

Para calcular las implicaciones de nuestros datos utilizamos la función `find_implications`

4.3.2 ¿Cuantas implicaciones se han extraido?

```
fc_datos$find_implications()

fc_datos$implications$cardinality()
```

```
[1] 1905
```

Vemos que hemos obtenido un gran numero de implicaciones, vamos a ver los primeros:

```
head(fc_datos$implications)
```

```
Implication set with 6 implications.
Rule 1: {fake = Si} -> {external URL = No}
Rule 2: {#follows is (200, Inf], fake = No} -> {name==username = No}
Rule 3: {#follows is (60, 200], fake = No} -> {profile pic = Si}
Rule 4: {#follows is (60, 200], #follows is (200, Inf]} -> {profile pic = Si,
  profile pic = No, nums/length username is (0, 0.2], nums/length username
  is (0.2, 0.4], nums/length username is (0.4, 0.6], nums/length username
  is (0.6, 0.8], nums/length username is (0.8, 1], fullname words is (0, 1],
  fullname words is (1, 3], fullname words is (3, 5], fullname words is (5,
  Inf], nums/length fullname is (0, 0.2], nums/length fullname is (0.2, 0.4],
  nums/length fullname is (0.4, 0.6], nums/length fullname is (0.6, 0.8],
  nums/length fullname is (0.8, 1], name==username = Si, name==username = No,
  description length is (0, 15], description length is (15, 25], description
  length is (25, 80], description length is (80, 150], external URL = Si,
  external URL = No, private = Si, private = No, #posts is (0, 1], #posts is (1,
  5], #posts is (5, 10], #posts is (10, 50], #posts is (50, Inf], #followers is
  (0, 10], #followers is (10, 60], #followers is (60, 200], #followers is (200,
  Inf], #follows is (0, 10], #follows is (10, 60], fake = Si, fake = No}
Rule 5: {#follows is (10, 60], fake = No} -> {profile pic = Si, name==username =
  No}
Rule 6: {#follows is (10, 60], #follows is (200, Inf]} -> {profile pic = Si,
  profile pic = No, nums/length username is (0, 0.2], nums/length username
  is (0.2, 0.4], nums/length username is (0.4, 0.6], nums/length username
  is (0.6, 0.8], nums/length username is (0.8, 1], fullname words is (0, 1],
  fullname words is (1, 3], fullname words is (3, 5], fullname words is (5,
  Inf], nums/length fullname is (0, 0.2], nums/length fullname is (0.2, 0.4],
  nums/length fullname is (0.4, 0.6], nums/length fullname is (0.6, 0.8],
  nums/length fullname is (0.8, 1], name==username = Si, name==username = No,
  description length is (0, 15], description length is (15, 25], description
  length is (25, 80], description length is (80, 150], external URL = Si,
  external URL = No, private = Si, private = No, #posts is (0, 1], #posts is (1,
  5], #posts is (5, 10], #posts is (10, 50], #posts is (50, Inf], #followers is
  (0, 10], #followers is (10, 60], #followers is (60, 200], #followers is (200,
  Inf], #follows is (0, 10], #follows is (60, 200], fake = Si, fake = No}
```

Como tenemos un gran numero de implicaciones, vamos a intentar reducirlas y quedarnos con las mas importantes aplicando técnicas de simplificación.

4.3.3 Calculo del tamaño de las implicaciones y la media de la parte y derecha de dichas implicaciones.

Este calculo nos proporciona una medida cuantitativa de las relaciones entre atributos. El tamaño de una implicación se refiere al número de atributos en sus conjuntos de premisa A y su consecuente B. La media de estos tamaños se obtiene haciendo la media del número de atributos en las partes izquierda y derecha de todas las implicaciones, ofreciendo una visión general.

```
colMeans(fc_datos$implications$size())
```

```
     LHS      RHS
5.809974 4.205249
```

Con esto valores obtenemos, en la parte derecha de la regla suele haber una media de 5,8 elementos mientras que en la parte derecha una media de 4,2 elementos.

4.3.4 Lógica de simplificación.

Vamos a intentar de simplificar nuestras implicaciones para poder quedarnos con las mas importantes y significativas.

```
fc_datos$implications$apply_rules(rules = c("simplification"))
```

```
Processing batch

--> Simplification: from 1905 to 1905.
```

```
head(fc_datos$implications)
```

```
Implication set with 6 implications.
Rule 1: {fake = Si} -> {external URL = No}
Rule 2: {#follows is (200, Inf], fake = No} -> {name==username = No}
Rule 3: {#follows is (60, 200], fake = No} -> {profile pic = Si}
Rule 4: {#follows is (60, 200], #follows is (200, Inf]} -> {profile pic = Si,
  profile pic = No, nums/length username is (0, 0.2], nums/length username
  is (0.2, 0.4], nums/length username is (0.4, 0.6], nums/length username
  is (0.6, 0.8], nums/length username is (0.8, 1], fullname words is (0, 1],
  fullname words is (1, 3], fullname words is (3, 5], fullname words is (5,
```

```
  Inf], nums/length fullname is (0, 0.2], nums/length fullname is (0.2, 0.4],
  nums/length fullname is (0.4, 0.6], nums/length fullname is (0.6, 0.8],
  nums/length fullname is (0.8, 1], name==username = Si, name==username = No,
  description length is (0, 15], description length is (15, 25], description
  length is (25, 80], description length is (80, 150], external URL = Si,
  external URL = No, private = Si, private = No, #posts is (0, 1], #posts is (1,
  5], #posts is (5, 10], #posts is (10, 50], #posts is (50, Inf], #followers is
  (0, 10], #followers is (10, 60], #followers is (60, 200], #followers is (200,
  Inf], #follows is (0, 10], #follows is (10, 60], fake = Si, fake = No}
Rule 5: {#follows is (10, 60], fake = No} -> {profile pic = Si, name==username =
  No}
Rule 6: {#follows is (10, 60], #follows is (200, Inf]} -> {profile pic = Si,
  profile pic = No, nums/length username is (0, 0.2], nums/length username
  is (0.2, 0.4], nums/length username is (0.4, 0.6], nums/length username
  is (0.6, 0.8], nums/length username is (0.8, 1], fullname words is (0, 1],
  fullname words is (1, 3], fullname words is (3, 5], fullname words is (5,
  Inf], nums/length fullname is (0, 0.2], nums/length fullname is (0.2, 0.4],
  nums/length fullname is (0.4, 0.6], nums/length fullname is (0.6, 0.8],
  nums/length fullname is (0.8, 1], name==username = Si, name==username = No,
  description length is (0, 15], description length is (15, 25], description
  length is (25, 80], description length is (80, 150], external URL = Si,
  external URL = No, private = Si, private = No, #posts is (0, 1], #posts is (1,
  5], #posts is (5, 10], #posts is (10, 50], #posts is (50, Inf], #followers is
  (0, 10], #followers is (10, 60], #followers is (60, 200], #followers is (200,
  Inf], #follows is (0, 10], #follows is (60, 200], fake = Si, fake = No}
```

```
fc_datos$implications$cardinality()
```

```
[1] 1905
```

Vemos que el numero de implicaciones no se ha reducido, como podíamos haber pensado. Esto se debe a que al simplificar realmente no reduce la cantidad e implicaciones, sino los atributos de estas, eliminando verdades absolutas o otras parámetros redundantes.

4.3.4.1 Eliminar la redundancia.

Vamos también a aplicar `composition, generalization, simplification y rsimplification` para eliminar la redundancia dentro de las implicaciones.

```
fc_datos$implications$apply_rules(rules = c("composition",
                                            "generalization",
                                           "simplification",
                                           "rsimplification"))
```

```
Processing batch

--> Composition: from 1905 to 1905.

--> Generalization: from 1905 to 1905.

--> Simplification: from 1905 to 1905.

--> Right Simplification: from 1905 to 1905.
```

```
head(fc_datos$implications)
```

```
Implication set with 6 implications.
Rule 1: {fake = Si} -> {external URL = No}
Rule 2: {#follows is (200, Inf], fake = No} -> {name==username = No}
Rule 3: {#follows is (60, 200], fake = No} -> {profile pic = Si}
Rule 4: {#follows is (60, 200], #follows is (200, Inf]} -> {#follows is (0, 10]}
Rule 5: {#follows is (10, 60], fake = No} -> {profile pic = Si, name==username =
  No}
Rule 6: {#follows is (10, 60], #follows is (200, Inf]} -> {#follows is (0, 10]}
```

```
fc_datos$implications$cardinality()
```

```
[1] 1905
```

Al igual que antes, el numero de implicaciones no se ha reducido, como podíamos haber pensado. Esto se debe a que al simplificar realmente no reduce la cantidad e implicaciones, sino los atributos de estas, eliminando verdades absolutas o otras parámetros redundantes.

```
colMeans(fc_datos$implications$size())
```

```
     LHS      RHS
3.317060 1.247769
```

Ahora, despues de simplificar nuestras implicaciones, la media de atributos de cada parte de la regla ha bajado considerablemente.

4.3.5 Análisis de implicaciones importantes

Al igual que con las reglas, nos interesa las implicaciones que tengan en su parte derecha los atributos sobre si la cuenta es falsa o no, puesto que nuestro objetivo es detectar estas cuentas falsas.

```
head(fc_datos$implications$filter(rhs="fake = Si"))
```

```
Implication set with 6 implications.
Rule 1: {#followers is (10, 60], #follows is (200, Inf]} -> {fake = Si}
Rule 2: {#followers is (10, 60], #follows is (0, 10]} -> {name==username = No,
  private = No, fake = Si}
Rule 3: {#followers is (0, 10], #follows is (60, 200]} -> {name==username = No,
  private = Si, fake = Si}
Rule 4: {#followers is (0, 10], #follows is (0, 10]} -> {fake = Si}
Rule 5: {private = No, #followers is (0, 10]} -> {fake = Si}
Rule 6: {description length is (25, 80], #posts is (1, 5], #follows is (10, 60]}
  -> {fake = Si}
```

Obervando esta serie de reglas, podemos obtener gran cantidad de informacion para poder detectar y diferencia rlas cuentas fake de las reales. Por ejemplo, una que puede parcer muy obvia, que si sigue a mucha gente, pero le siguen poca gente, es falsa.

Vamos a ver tambien las cuentas reales:

```
head(fc_datos$implications$filter(rhs="fake = No"))
```

```
Implication set with 6 implications.
Rule 1: {#followers is (200, Inf], #follows is (10, 60]} -> {private = No, fake
  = No}
Rule 2: {external URL = Si} -> {profile pic = Si, fake = No}
Rule 3: {#posts is (50, Inf], #follows is (0, 10]} -> {private = No, #followers
  is (200, Inf], fake = No}
Rule 4: {#posts is (0, 1], #followers is (200, Inf]} -> {#follows is (200, Inf],
  fake = No}
Rule 5: {description length is (25, 80], #posts is (50, Inf]} -> {profile pic =
  Si, fake = No}
Rule 6: {description length is (25, 80], #posts is (5, 10]} -> {fake = No}
```

Al contrario que lo anterior, si sigue a poca gente y le sigue mucha gente, significa que la cuenta es real.

Ambas suposiciones las podemos obtener gracias a que sabemos que para seguir a una persona, no es necesario que esa persona de su consentimiento, sino que puede ser algo automático. Sin embargo, obtener seguidores requiere a una segunda persona que desee seguir a esa cuenta, pudiendo verla previamente, lo que es mas difícil de conseguir para cuentas fake.

4.4 Funciones interesantes

Dentro del paquete de fcaR hay funciones interesantes para bien exportar a Latex, a arules, ...

```
reglas <- fc_datos$implications$to_arules()
#latex <- fc_datos$implications$to_latex()
```

También podemos hacer gráficos de nuestros conceptos:

```
#fc_datos$concepts$plot()
```

5 Regresión

La **regresión** es una técnica estadística y de machine learning utilizada para modelar y analizar relaciones entre variables. Su objetivo principal es entender cómo cambia una variable dependiente en función de una o más variables independientes. La regresión puede ser utilizada tanto para predecir valores futuros como para entender relaciones subyacentes en los datos.

En el contexto de la detección de cuentas falsas de Instagram, la regresión es una herramienta muy útil. Entrenamos y evaluamos el modelo en conjuntos de datos de entrenamiento y prueba, utilizando métricas para asegurar su efectividad. Finalmente, interpretamos los resultados para identificar las variables más influyentes y ajustamos el modelo para mejorar su precisión, creando así una herramienta eficaz para combatir el fraude en Instagram.

Ahora vamos a comenzar con

6

7 Summary

In summary, this book has no content whatsoever.

```
1 + 1
```

```
[1] 2
```