Fake Accounts Instagram

Antonio Cañete Baena

2024-05-24

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 Kaggle para extraer el máximo conocimiento posible usando las técnicas vistas 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 Instagram, tanto de spammers como de usuarios genuinos.

Dicho dataset está 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 más sofisticadas.

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

1.1 Carga de datos

Como hemos visto anteriormente, el dataset contiene dos archivos con datos. Sin embargo, puesto que el archivo *train.csv* contiene más entradas de datos, vamos a utilizar dicho conjunto de datos para aplicar nuestras técnicas de análisis.

En primer lugar, vamos a cargar las librerías 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 cuántas 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 numéricos, 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 numérico 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 Análisis 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 límite, ...

1.2.1 profile pic

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

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

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

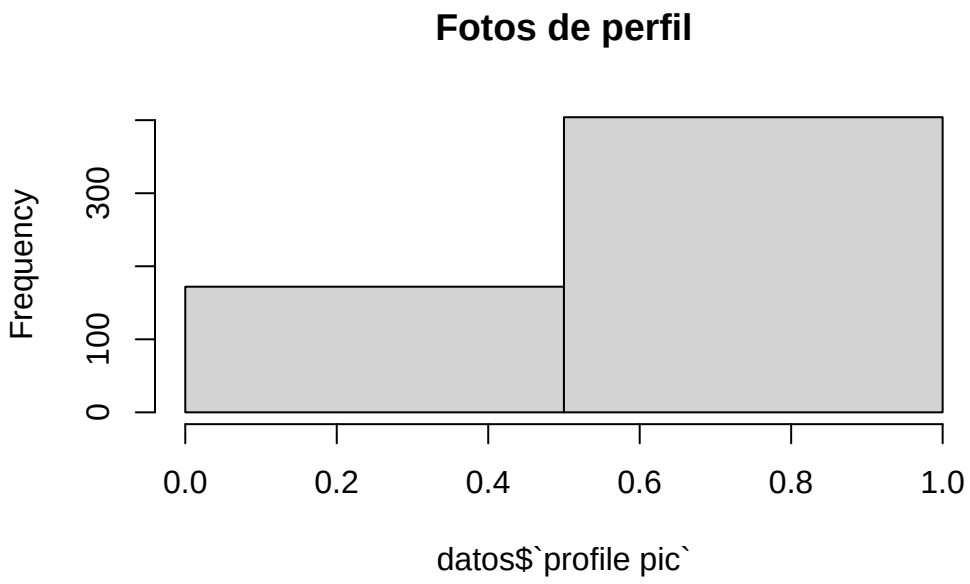
```
anyNA(datos$`profile pic`)
```

```
[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" )
```



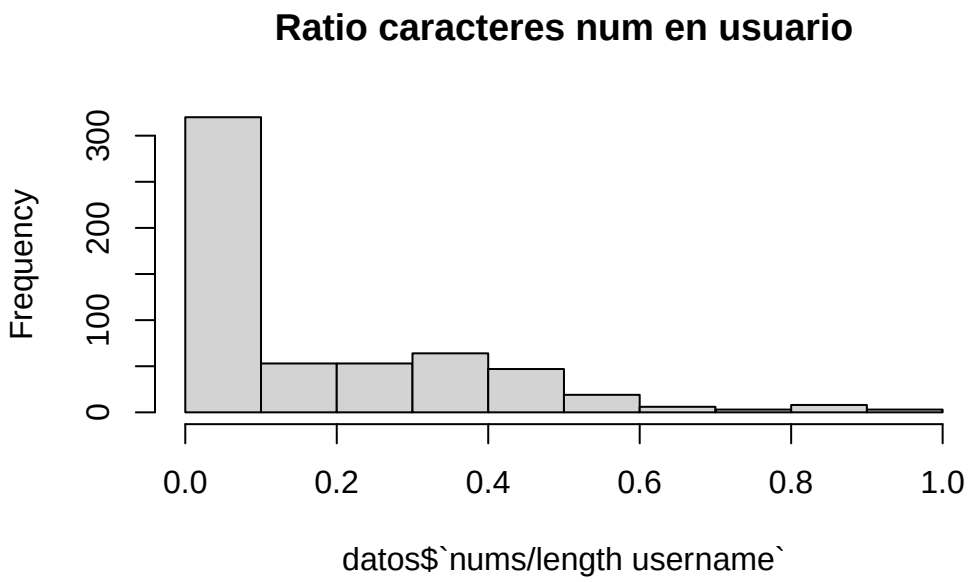
Observamos que más de la mitad de los usuarios tienen foto de perfil.

1.2.2 nums/length username

Este atributo representa el ratio de número de caracteres numéricos en el nombre de usuario respecto a su longitud.

Por ejemplo: Ant234 -> Ratio 1.

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



1.2.3 fullname words

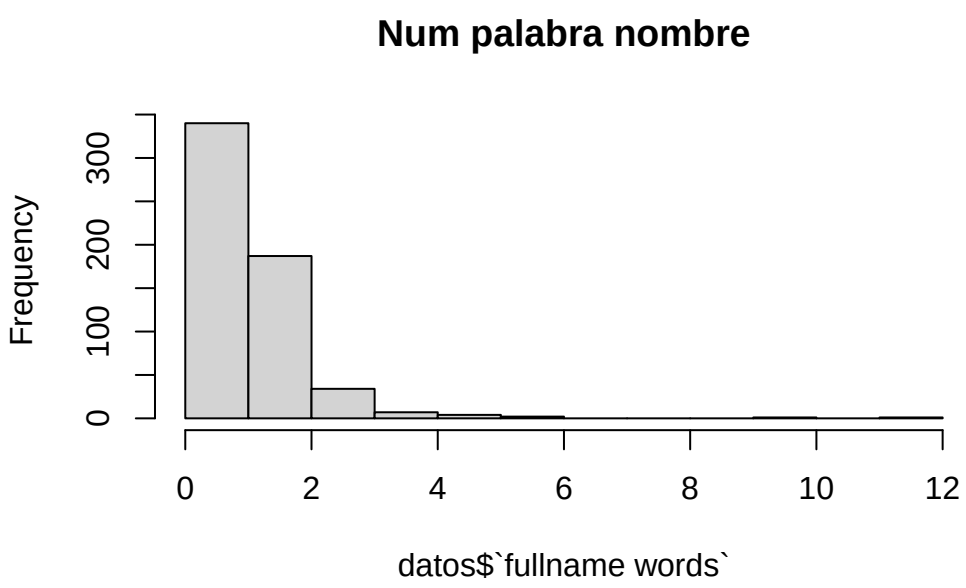
Este atributo representa la cantidad de palabras que componen el nombre del usuario.

```
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
```

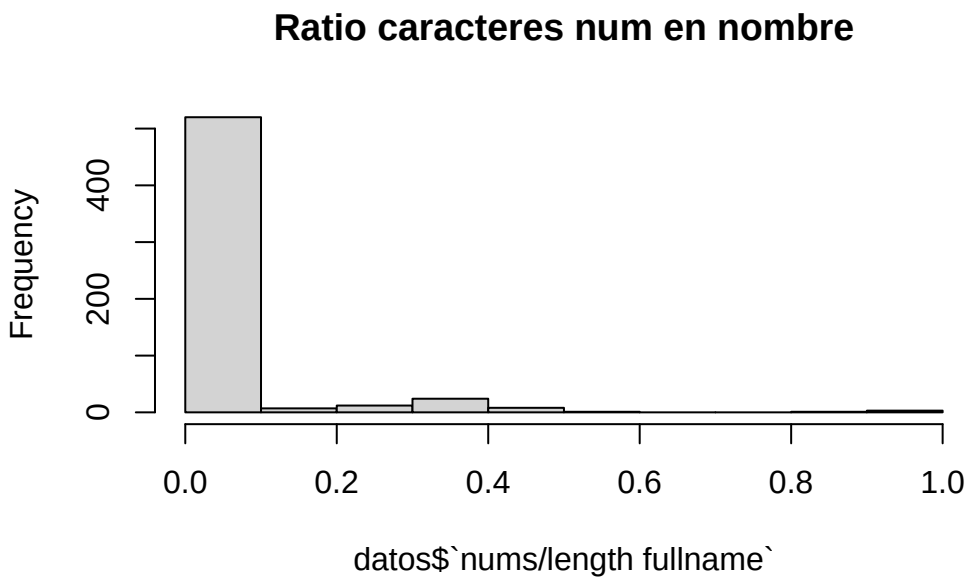
```
         n
1 81.59722
```

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 número de caracteres numéricos en el nombre completo del usuario respecto a su longitud.

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

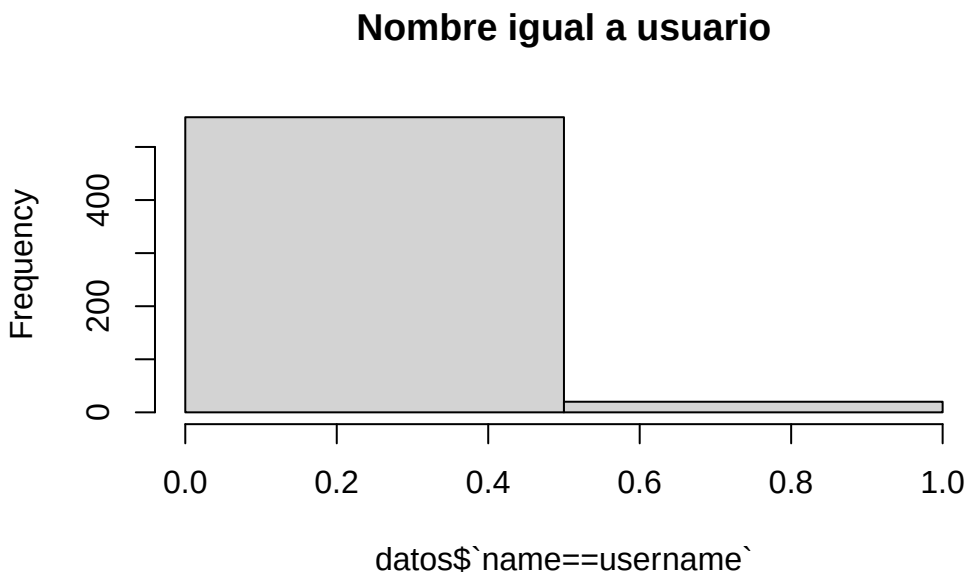


Observamos que es bastante inusual que un usuario tenga caracteres en su nombre completo, mientras que, como hemos visto antes, en el nombre de usuario, es más frecuente encontrar caracteres.

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" )
```

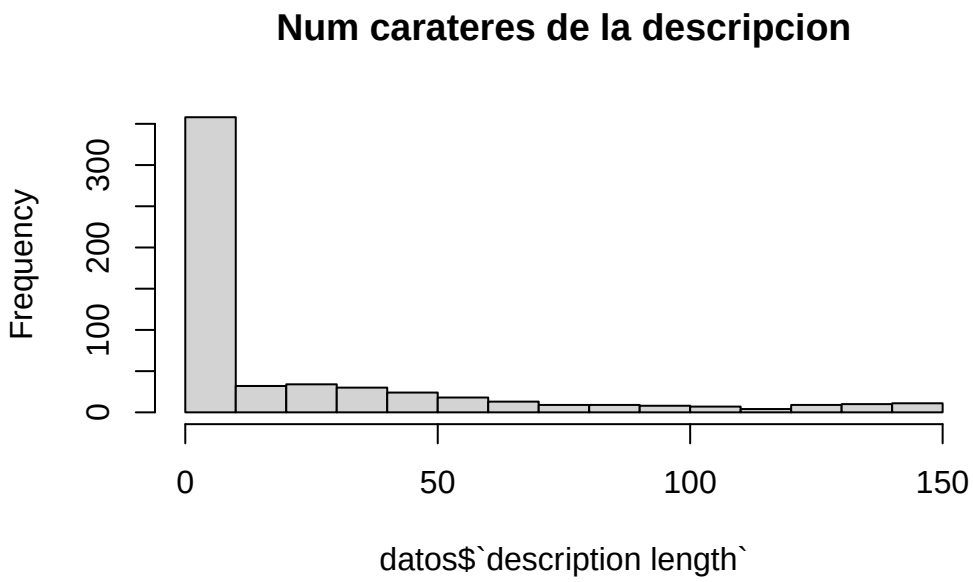


Concluimos que es bastante inusual que un usuario tenga el mismo nombre de usuario y nombre completo.

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 límite 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`)
```

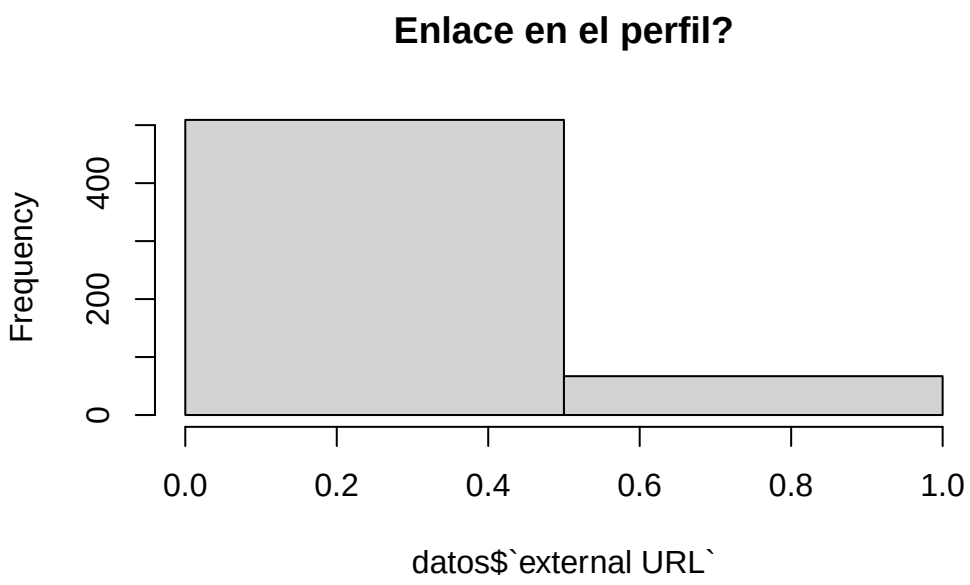
```
[1] 22.62326
```

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

1.2.7 external URL

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

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

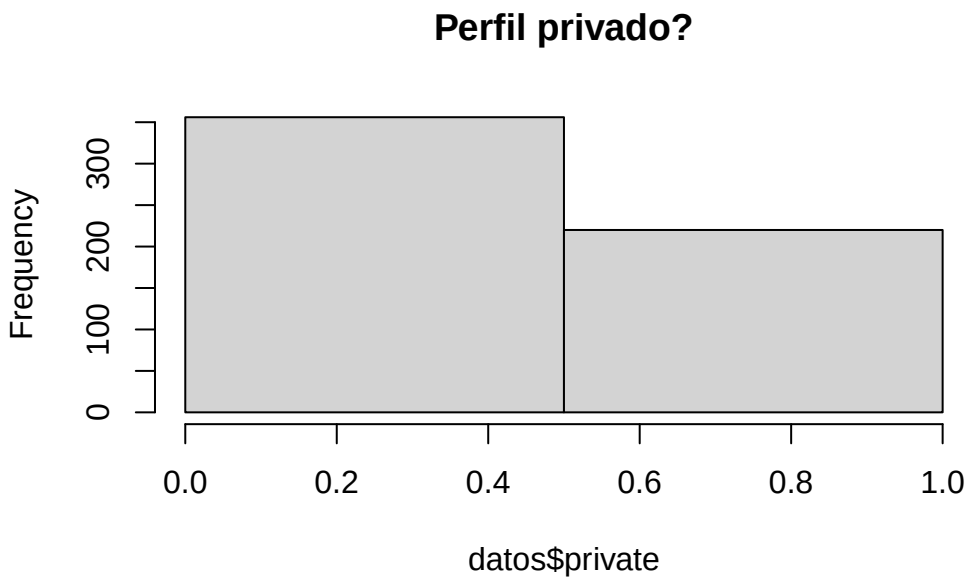


Lo mas común son los perfiles sin enlaces externos.

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 más de igualdad, el número de cuentas privadas es poco más de la mitad del número de cuentas públicas.

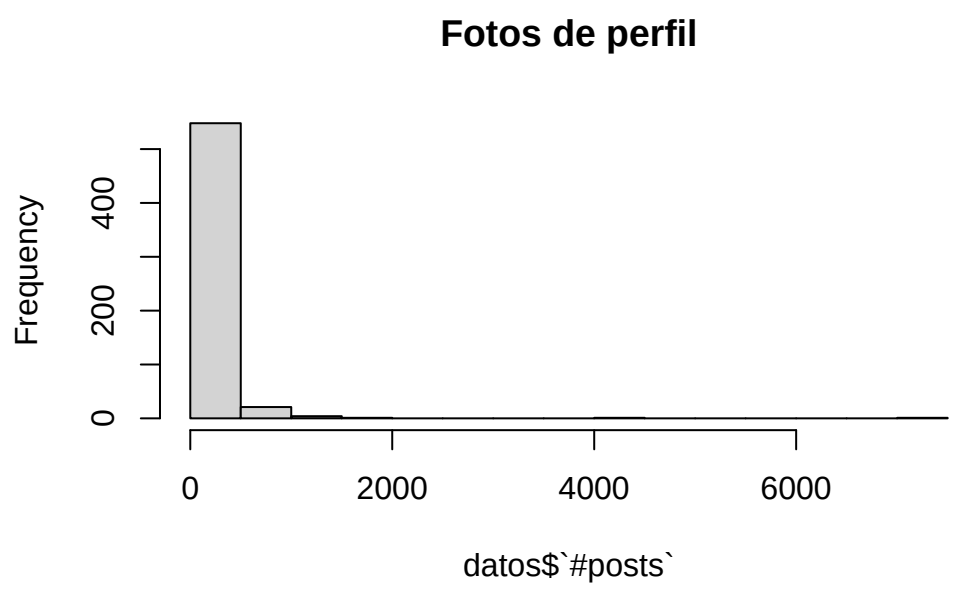
```
datos %>% mutate(private = ifelse(private==1,"Privada","Publica")) %>%  group_by(private) %>%
```

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

1.2.9 post

Este atributo representa el número de publicaciones de la cuenta.

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




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

```
[1] 7389
```

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

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

```
[1] 7389
```

Vemos que es un valor bastante inusual o que podría tratarse de alguna cuenta que publique mucho contenido a diario. Vamos a verla:

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

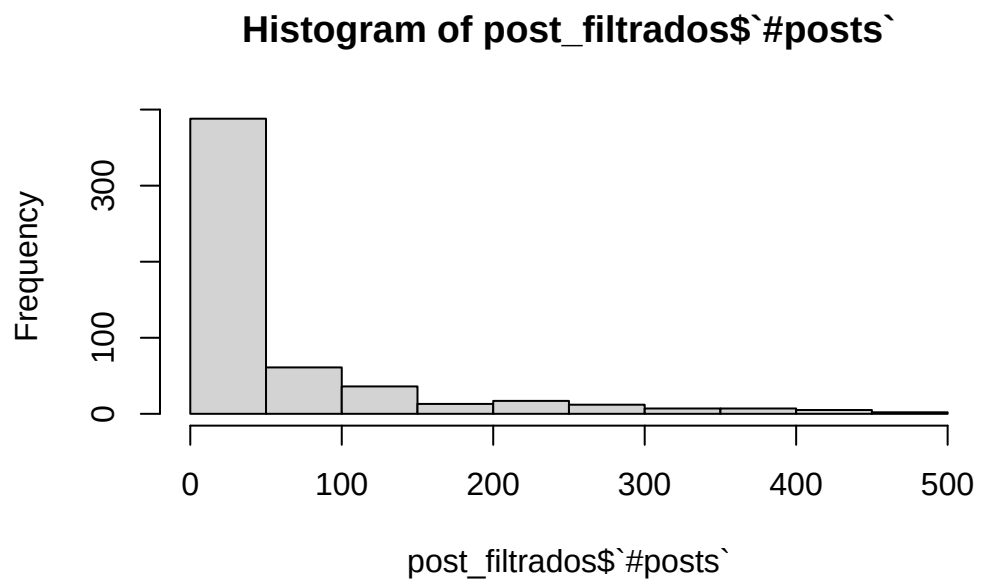
```
# A tibble: 1 x 12
  `profile pic` `nums/length username` `fullname words` `nums/length fullname`
          <dbl>                  <dbl>            <dbl>                   <dbl>
1             1                      0                0                       0
```

```
# 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 alrededor de 7800 publicaciones, por lo que dicho valor puede ser debido a la cuenta de algún famoso.

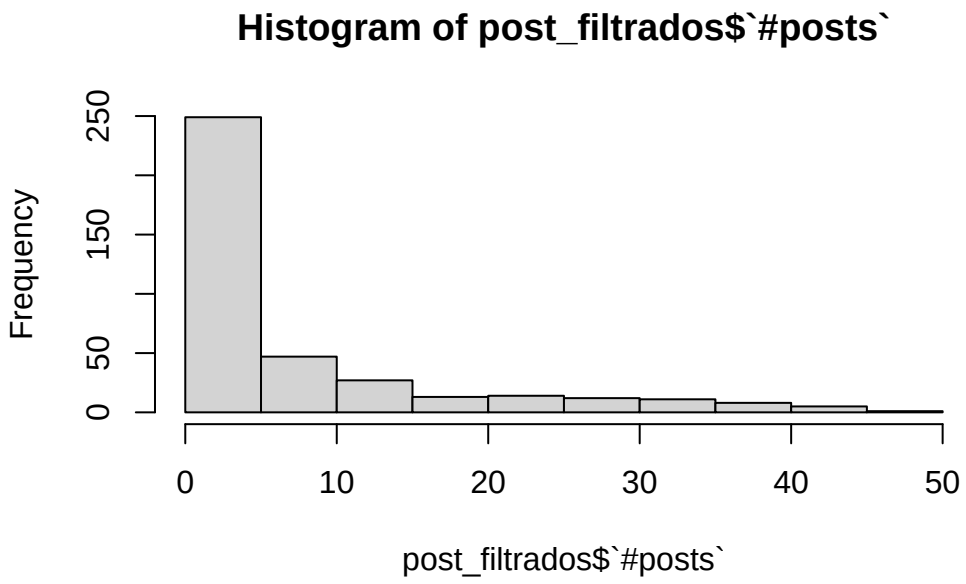
Vamos a volver a dibujar el histograma pero con un umbral un poco más razonable:

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



Ahora ya podemos extraer información más fácilmente, como que la mayoría de usuarios tiene menos de 50 publicaciones. Vamos a verlo en más detalle:

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



Observamos que hay un gran número de usuarios con menos de 5 publicaciones. Vamos a ver cuántos de ellos tienen 0 publicaciones y a calcular 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 número de publicaciones, esta media puede no ser muy significativa.

Vamos a analizar entonces sus cuartiles y mediana:

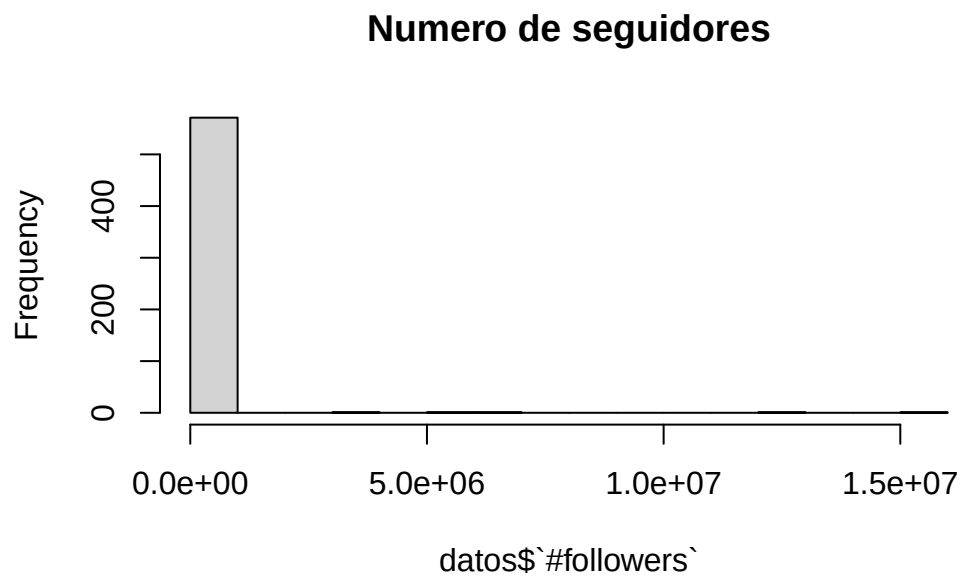
```
summary(datos$`#posts`)
```

```
   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 anterior, este histograma no tiene sentido porque hay algún 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 publicaciones encontrado anteriormente:

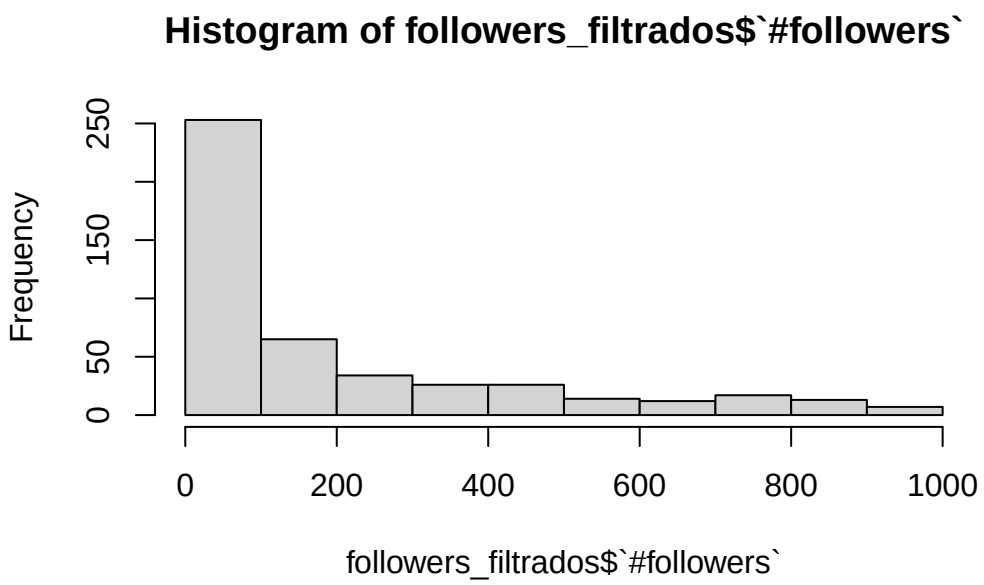
```
datos %>% filter (`#followers`==max(`#followers`)) %>% select(`#posts`)
```

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

Aunque podría ser la cuenta de una celebridad, vemos que tiene un número de publicaciones relativamente normal, comparado con el valor de 7389 publicaciones que obtuvimos anteriormente.

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

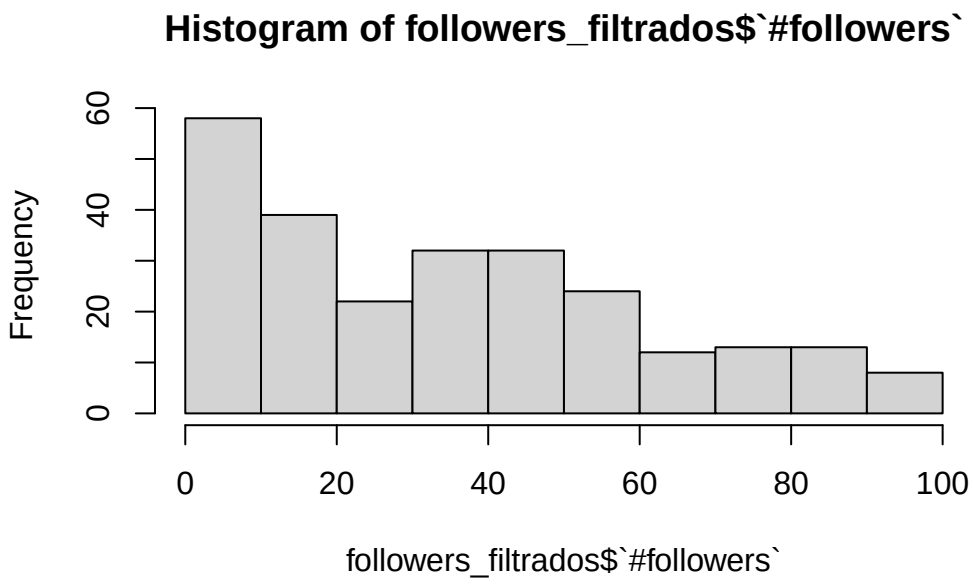
```
followers_filtrados <- datos %>% select(`#followers`)%>% filter(`#followers` <1000)
hist(followers_filtrados$`#followers`)
```



Observamos que la mayoría de usuarios no tienen un gran número de seguidores, en concreto, menos de 100.

Vamos a verlo:

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



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

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

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

```
[1] 85307.24
```

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

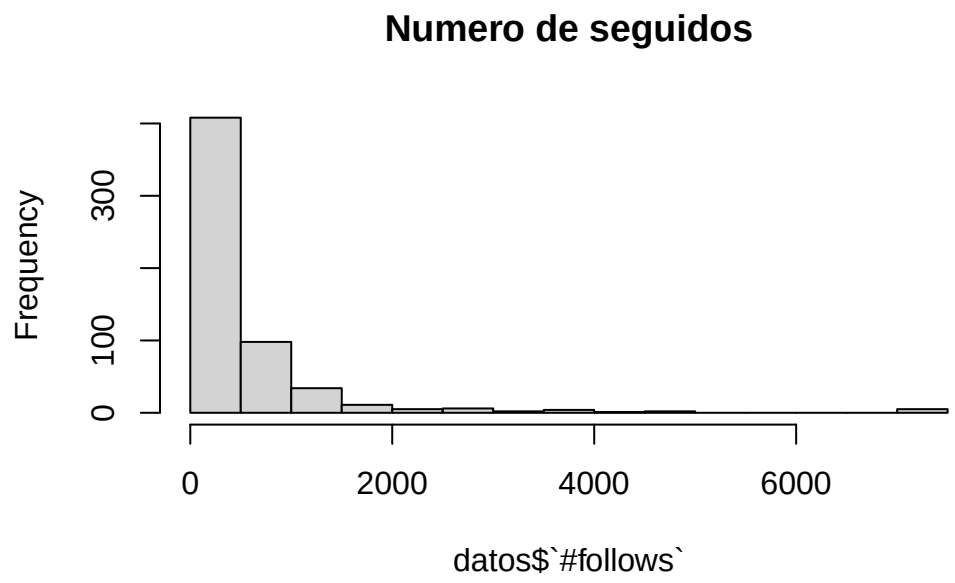
```
   Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
      0       39      150    85307      716 15338538
```

Sabiendo que la mediana divide al 50% de los datos, dicho valor es más 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
```

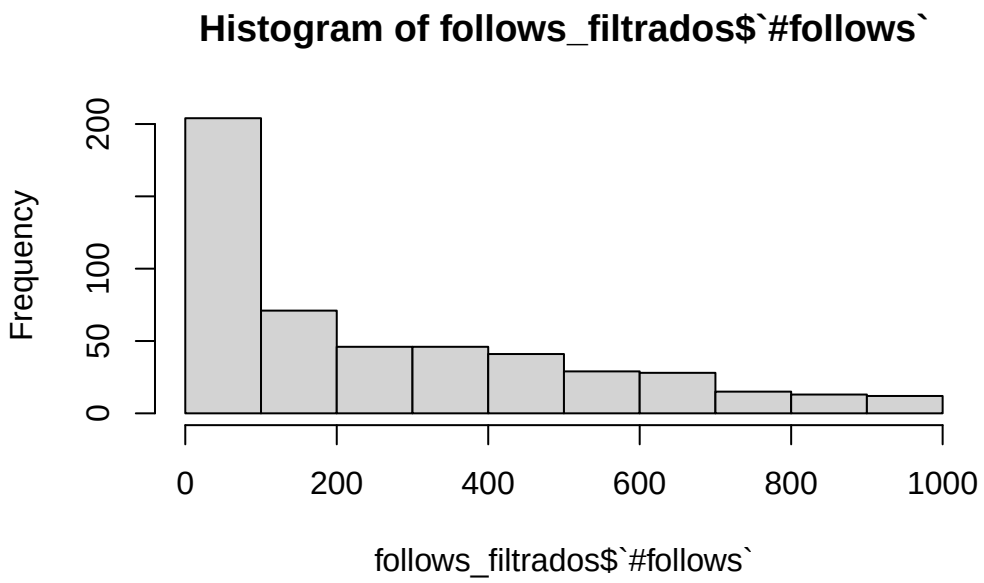
Dicho valor corresponde con el valor máximo de cuentas que Instagram permite a los usuarios seguir para reducir el spam. Por lo tanto, las cuentas que siguen a un gran número de personas se pueden llegar a asociar a spammers. Vamos a ver cuántas cuentas están en este límite:

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

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

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

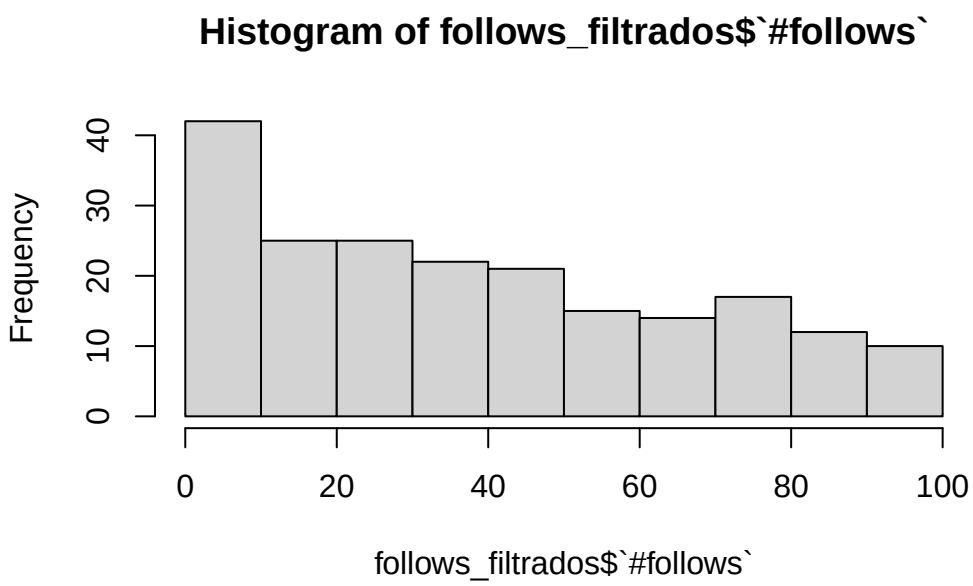
```
follows_filtrados <- datos %>% select(`#follows`)%>% filter(`#follows` <1000)
hist(follows_filtrados$`#follows`)
```



Observamos que más 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, las frecuencias están más repartidas.

Viendo que hay algunos usuarios con un gran número 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 tanto, vamos a analizar los cuartiles y la mediana en su lugar.

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

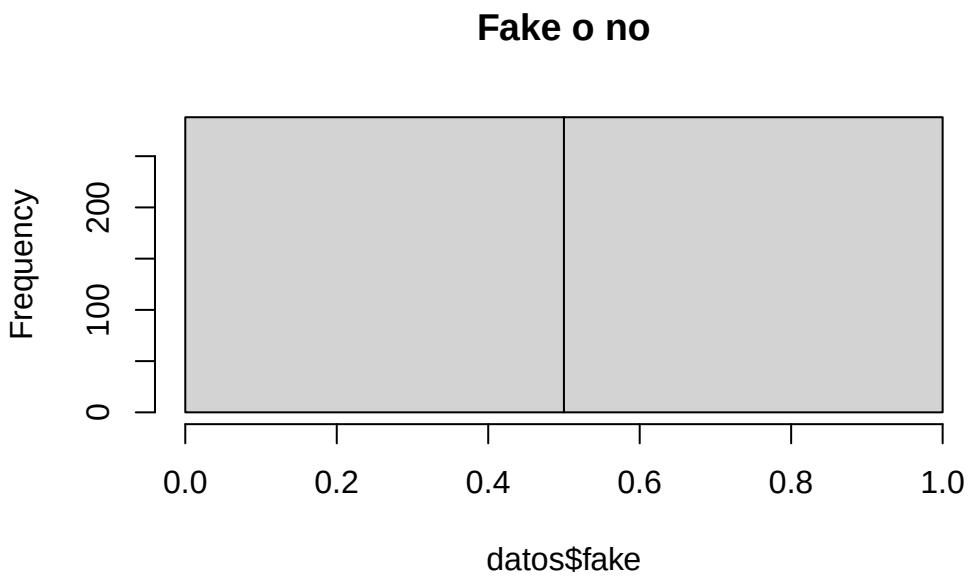
```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    0.0    57.5   229.5   508.4   589.5  7500.0
```

Ahora, con estos valores, ya podemos analizarlo un poco mejor y darnos cuenta de que el 50% de los usuarios no sigue a más 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 sus características cson 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

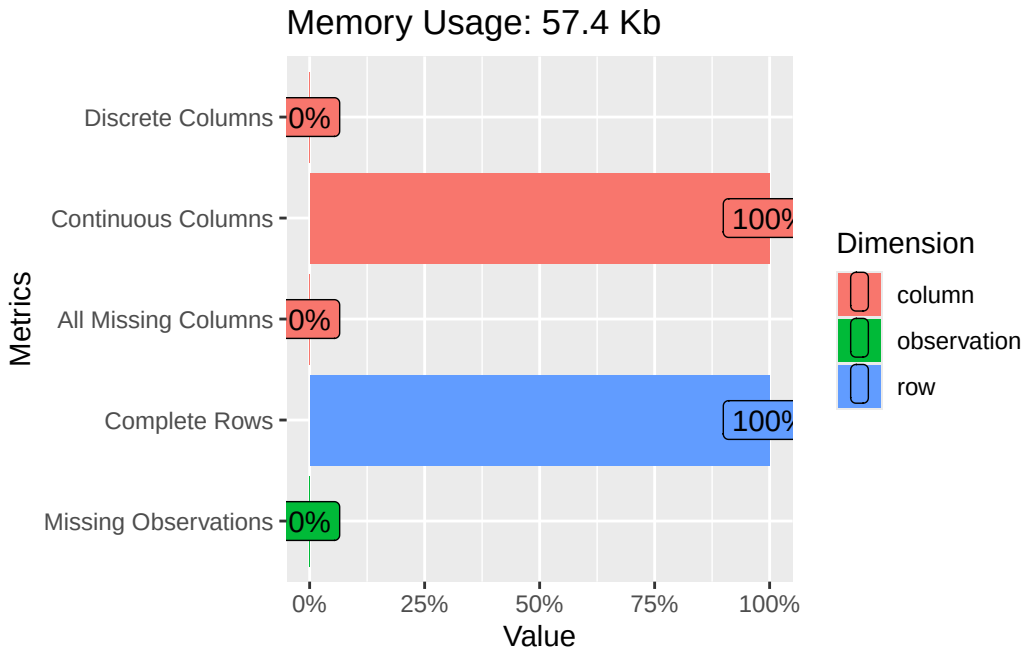
1.3.1.1 introduce

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>
```

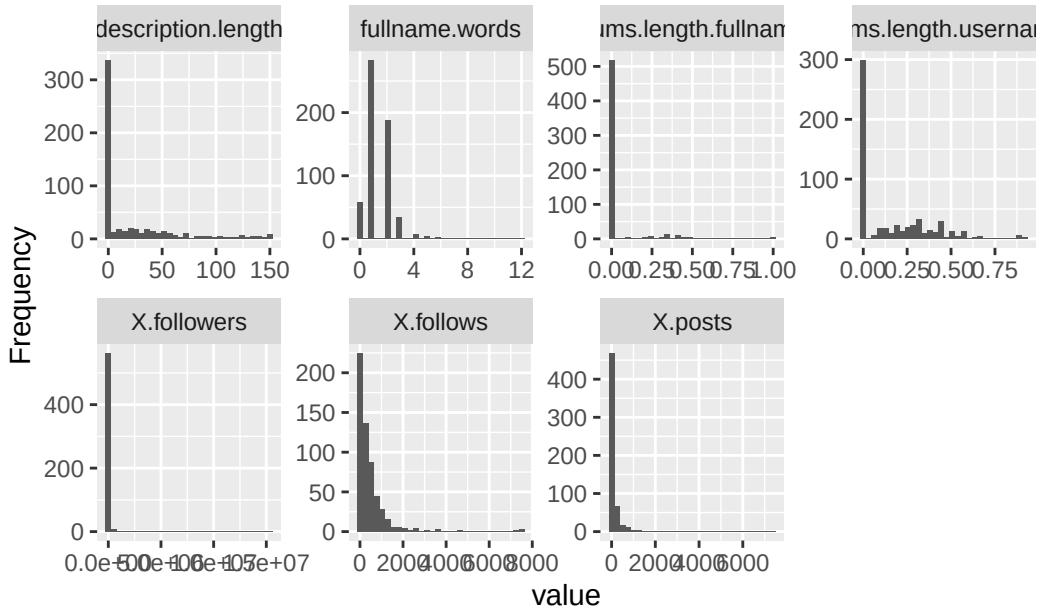
```
plot_intro(datos)
```



1.3.1.2 plot_histogram

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

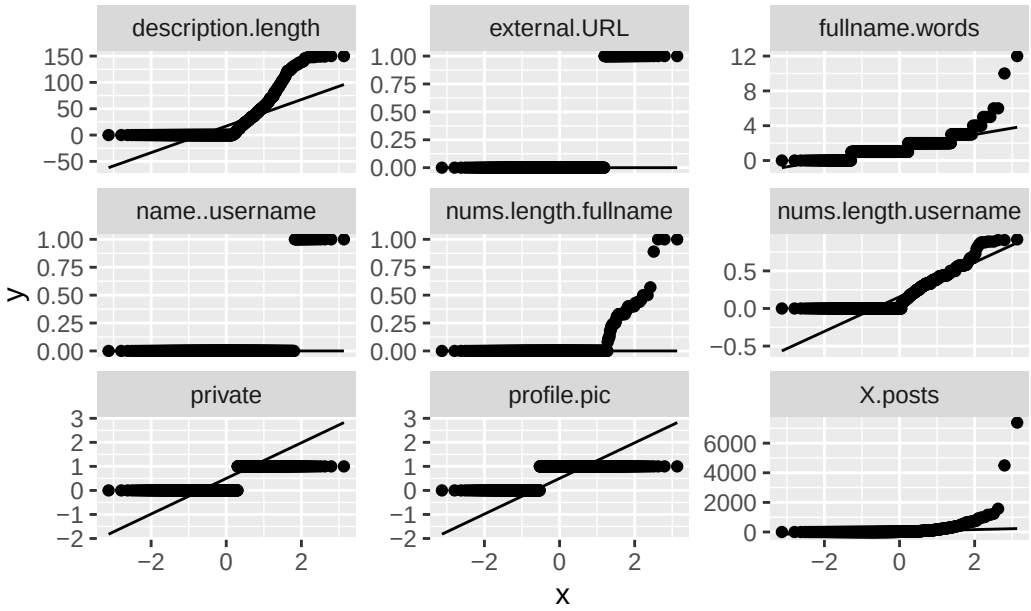
```
plot_histogram(datos)
```



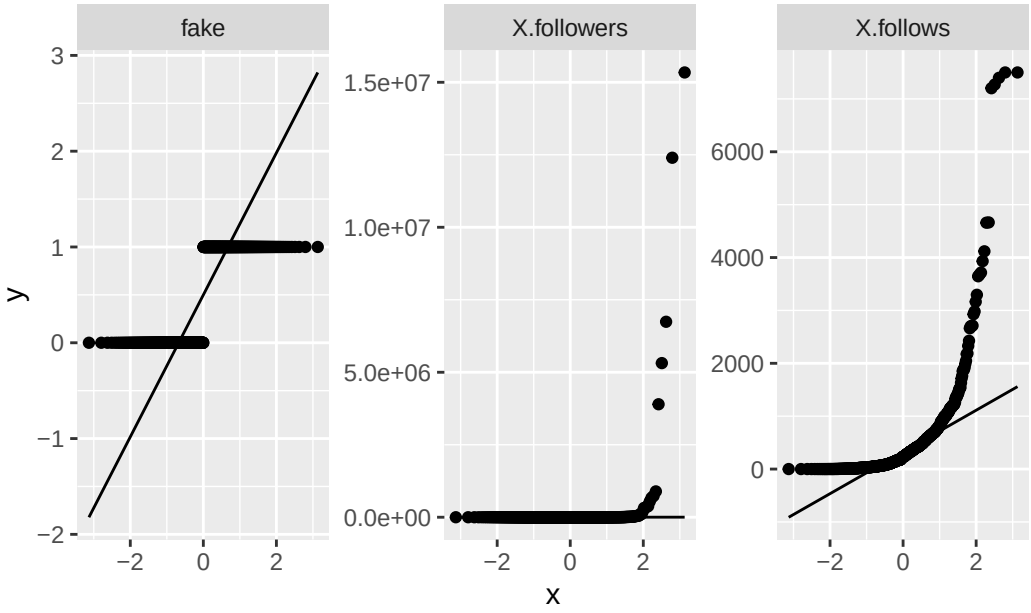
1.3.1.3 plot_qq

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 cómo continuar nuestro análisis.

Para ello, vamos a utilizar la herramienta `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 hacer este trabajo más fácil, vamos a realizar un preprocesado de los datos primero. Vamos a convertir todos los atributos que son discretos a factores.

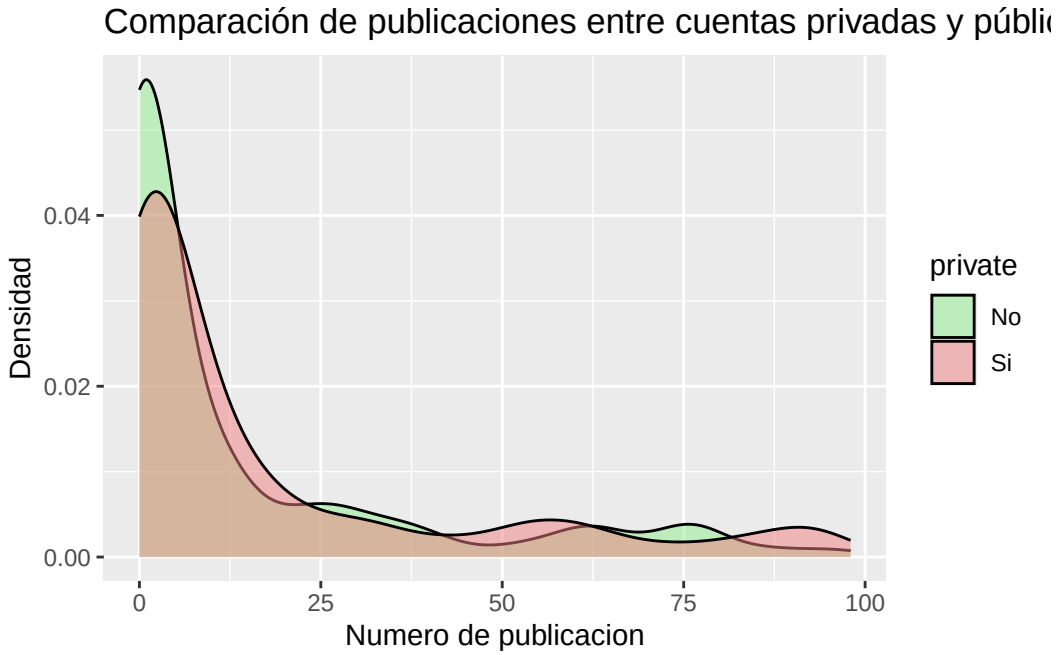
```
datos_refinados <- datos
columnas_binarias = c("profile pic","name==username","external URL","fake","private")
for (columna in columnas_binarias) {
  datos_refinados[[columna]] <-  factor(datos_refinados[[columna]], labels = c("No", "Si"))
}
```

Nuestros atributos discretos son binarios, solo tienen o bien Sí o No. Vamos a emplear ahora los gráficos para poder encontrar alguna relación entre las variables y, sobre todo, lo que más nos 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 cómo se comportan ambos tipos de usuarios, vamos a empezar analizando el número de publicaciones entre los usuarios con cuentas públicas y con cuentas privadas. Para ello, vamos a 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 en ambos casos, nuestra gráfica es similar, lo que sugiere que el número de publicaciones no depende de si la cuenta es privada o pública.

Sin embargo, lo que realmente nos interesa es encontrar relaciones para intentar determinar si una cuenta es de un spammer o de una persona real. Por lo tanto, vamos a centrarnos en comparar los atributos con el atributo "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 pública) tiene algo que ver:

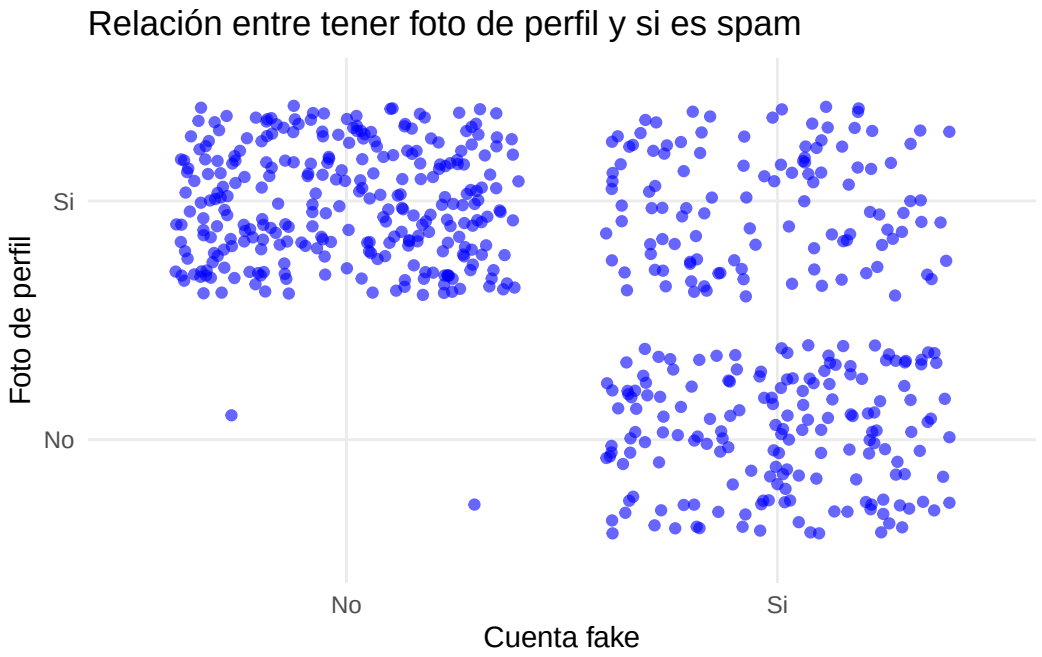
```
ggplot(datos_refinados, aes(x = `fake`, y = `private`)) +
  geom_count(color = "blue", alpha = 0.6) +
  scale_size_area()+
  labs(title = "Relación entre visibilidad del perfil y si es spam",
       x = "Cuenta fake",
       y = "Cuenta privada")+
  theme_minimal()
```

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

2.4 Relación entre tener foto de perfil y ser cuenta falsa.

Al igual que antes, vamos a comprobar dos variables discretas, por lo que el aspecto del gráfico será 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()
```

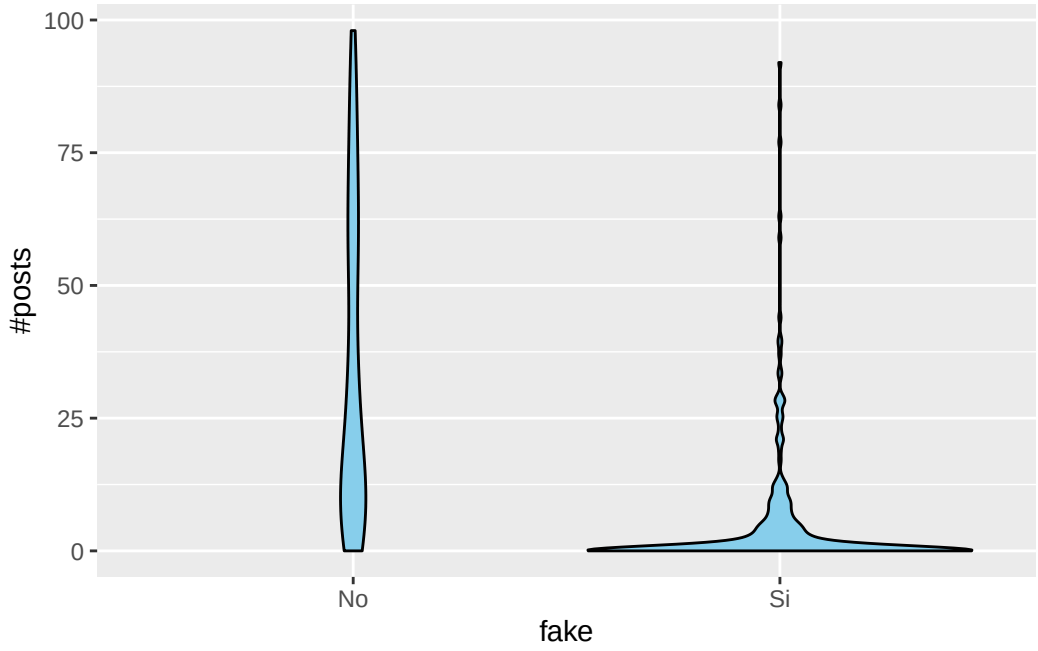


Hemos obtenido un resultado interesante, donde vemos que las cuentas reales, todas menos 2, tienen foto de perfil puesta, mientras que las cuentas falsas tienen más o menos un mismo número con foto de perfil y sin foto de perfil. Estos datos, combinados con otros que vamos a obtener más adelante, nos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

2.5 Relación entre número de publicaciones y cuentas falsas.

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 número 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")
```



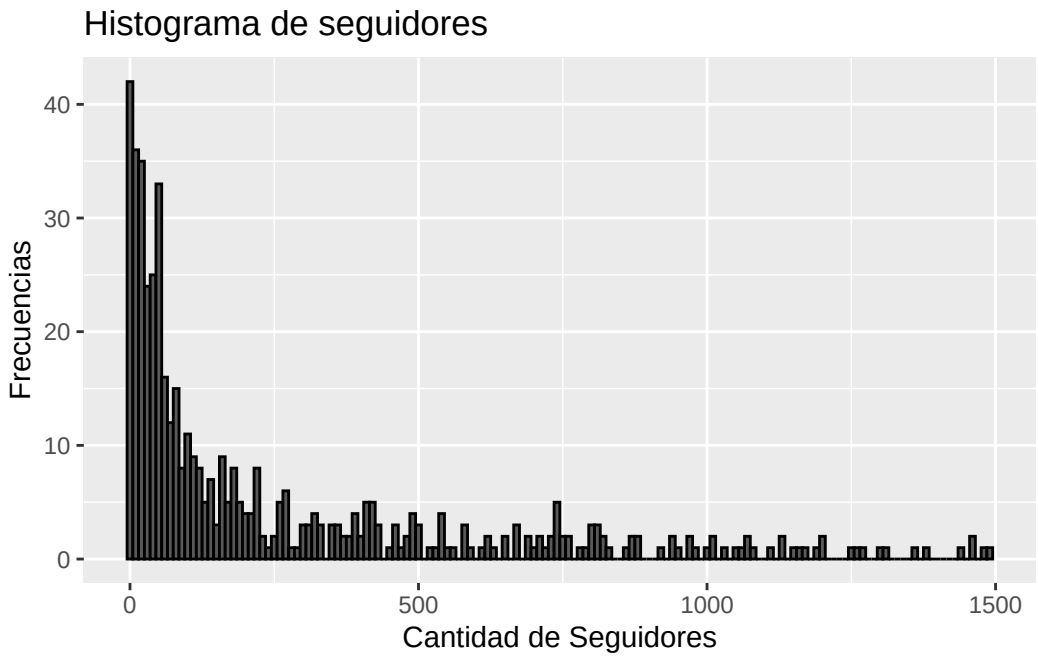
Observamos que teníamos razón. Después de eliminar aquellas cuentas con muchos posts, vemos que las cuentas falsas suelen tener un número reducido de publicaciones, mientras que las cuentas normales suelen tener una distribución más uniforme.

2.6 Análisis de número de seguidores.

Uno de los atributos más relevantes puede ser el número de seguidores. Por lo tanto, necesitamos analizarlo en profundidad. Vamos a comenzar con el número de seguidores.

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

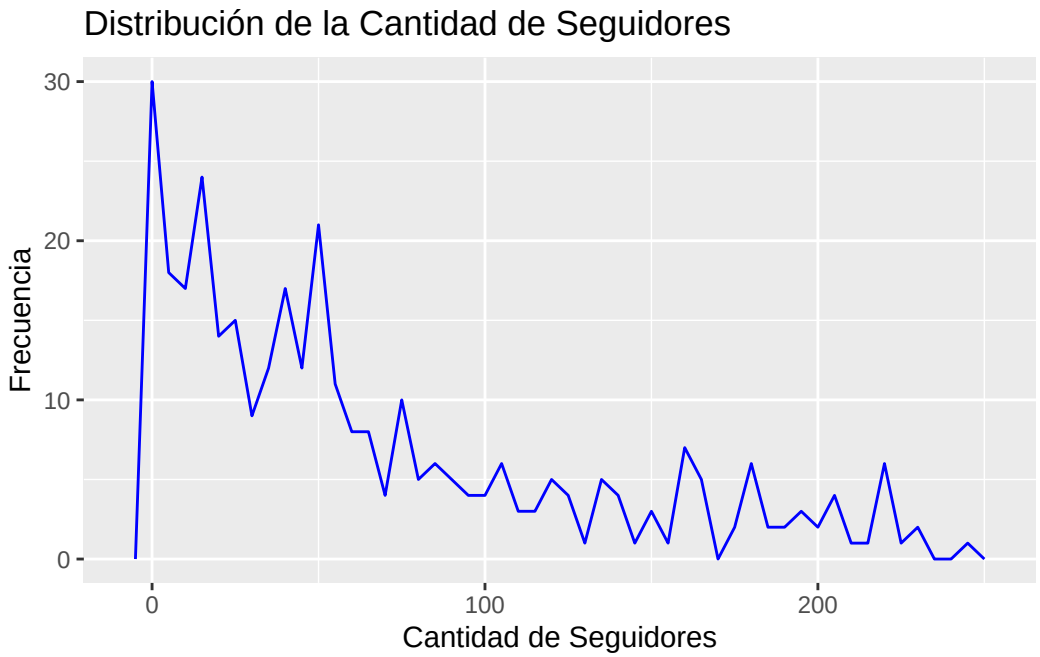
```
followers_filtrados <- datos_refinados %>% filter(`#followers` <1500)

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 la mayoría se concentra en menos de 250 seguidores.

Vamos a utilizar una gráfica de frecuencia para ver cómo 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 la frecuencia disminuye a medida que aumenta el número de seguidores.

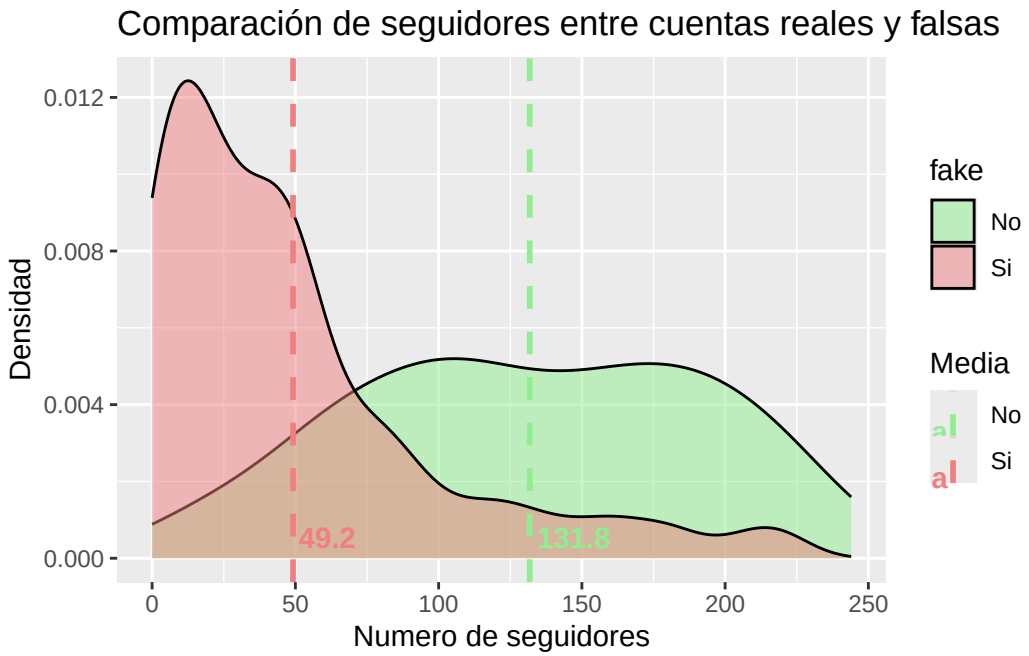
2.7 Comparación del número de seguidores entre cuentas reales y falsas.

Como nuestro principal objetivo es poder encontrar características similares que tengan las cuentas falsas para poder identificarlas fácilmente, vamos a visualizar este atributo en relación con el número de seguidores. Además, añadiremos las medias para obtener más información.

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

ggplot(data = followers_filtrados, aes( x = `#followers`,fill = `fake`)) +   geom_density(al
  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.
```



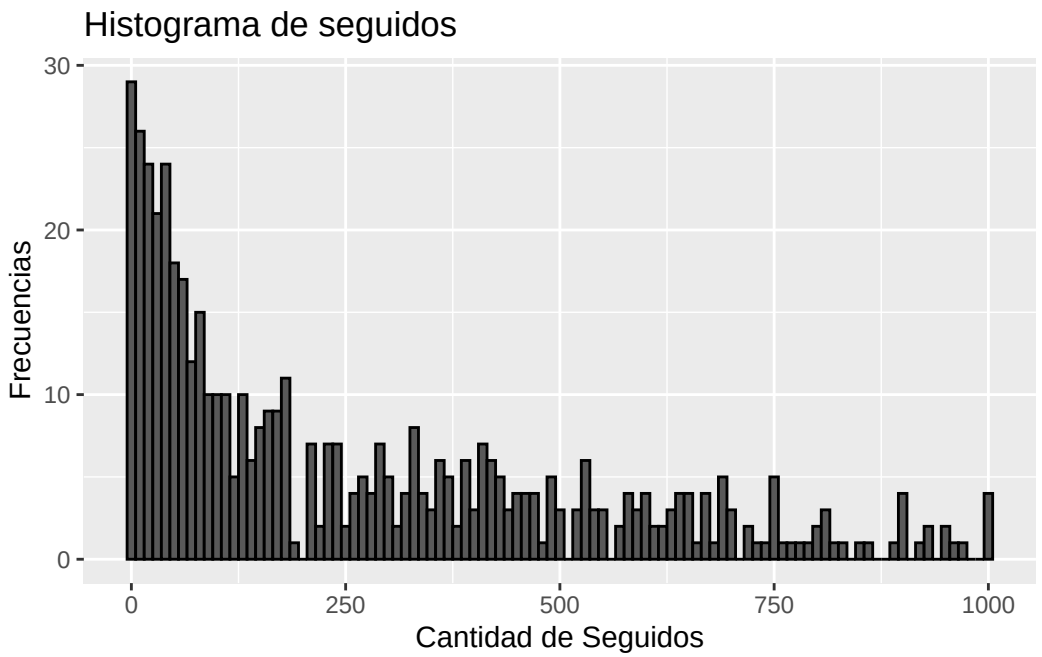
Aquí obtenemos información más interesante. Podemos observar que las cuentas falsas tienden a tener un menor número de seguidores, mientras que las cuentas reales, aunque no tienen muchos seguidores, suelen mantenerse en un intervalo entre 50 y 250. Esta información nos puede ser de importancia para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

2.8 Análisis de número de seguidos.

Ahora que hemos explorado cómo se comporta el número de seguidores según el tipo de cuentas a través de varios gráficos, vamos a continuar con el número de seguidos.

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

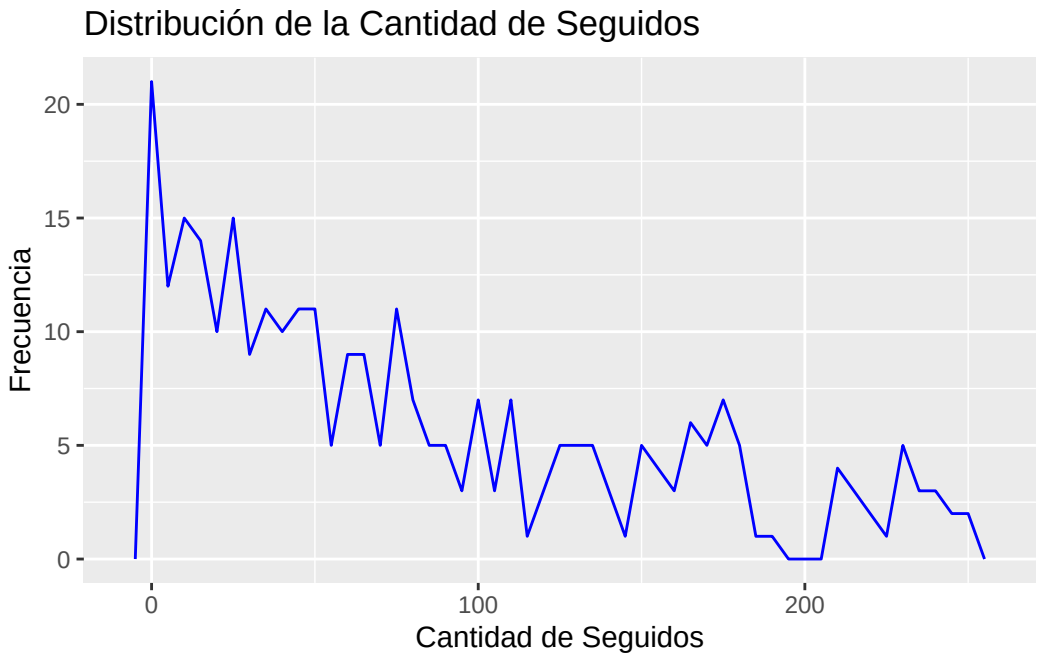
```r
follows_filtrados <- datos_refinados %>%  filter(`#follows` <1000)
ggplot(follows_filtrados, aes(x = `#follows`)) +   geom_histogram(binwidth = 10, color = 'bla
  labs(title = "Histograma de seguidos",
       x = "Cantidad de Seguidos",
       y = "Frecuencias")
```



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

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

```r
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")
```



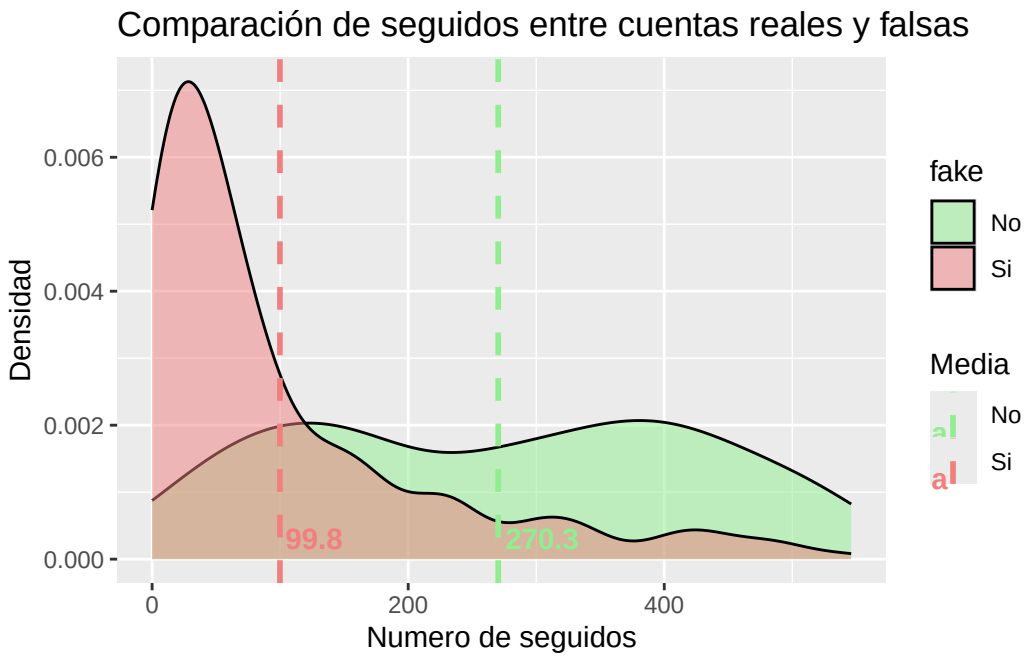
La mayor concentración se encuentra en menos de 100 seguidos y la frecuencia disminuye a medida que aumenta el número de seguidos.

2.9 Comparación del número de seguidos entre cuentas reales y falsas.

Como nuestro principal objetivo es poder encontrar características similares que tengan las cuentas falsas para poder identificarlas fácilmente, vamos a visualizar este atributo. Además, añadiremos las medias para obtener más 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")
```



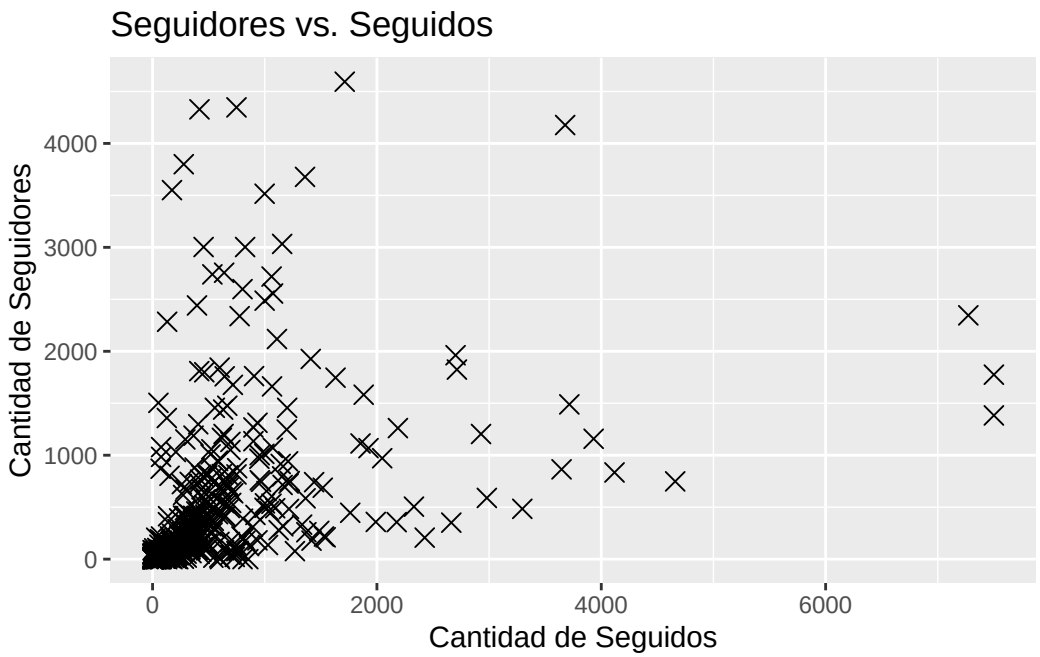
Aquí, al igual que con los seguidores, obtenemos información más interesante. Podemos observar que las cuentas falsas tienden a tener un menor número de seguidores, pero no tan cercano a 0, mientras que las cuentas reales suelen tener un número más repartido de seguidos. Esta información nos puede ser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

2.10 Relación entre número de seguidores y número de seguidos.

Ahora que hemos visto ambas variables por separado, vamos a utilizar gráficos de puntos o dispersión para ver varias variables juntas e intentar encontrar 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) +
```

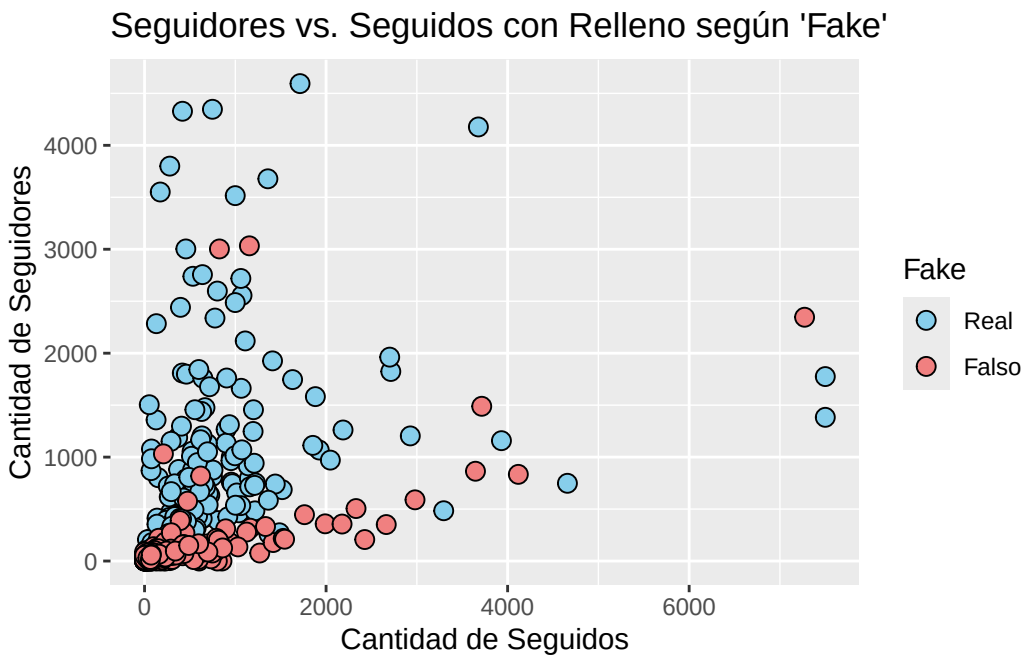
```
  labs(title = "Seguidores vs. Seguidos",
       x = "Cantidad de Seguidos",
       y = "Cantidad de Seguidores")+
  scale_fill_manual(values = c("skyblue", "lightcoral"))
```



Viendo este gráfico, solo podemos observar que casi todo se concentra en un número reducido tanto de seguidos como de seguidores.

Aunque dicha información no nos sea de mucha utilidad, vamos a añadir el parámetro para diferenciar cuentas falsas y reales. Podemos pensar que los seguidores y los seguidos tienen alguna relación con los usuarios que son falsos. Vamos a refinar un poco el DataSet eliminando los usuarios que tenían muchos seguidores. Vamos a investigar:

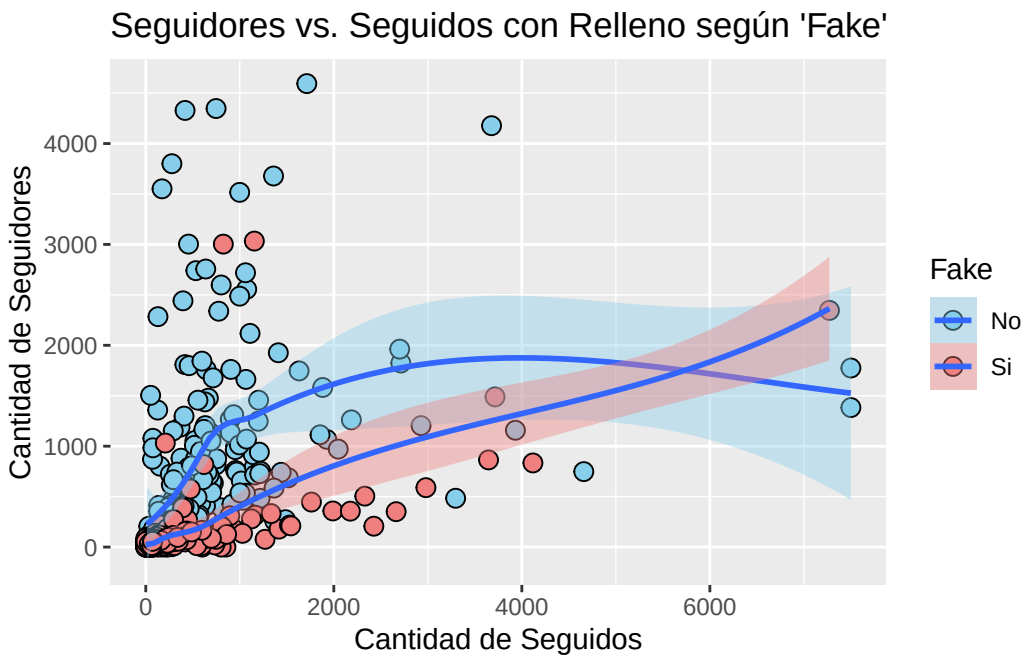
```
followers_filtrados <- datos_refinados %>%  filter(`#followers` <5000)


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 falsas suelen tener más cuentas seguidas que seguidores. Esto puede ser debido a que al ser cuentas generadas automáticamente, seguir a otras cuentas es una tarea que se puede automatizar, mientras que conseguir seguidores es algo más complicado y requiere de una acción activa por parte de otra persona para seguir la cuenta. 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 (falsos) se ajustan a la línea roja. Sin embargo, las cuentas verdaderas tienen una tendencia más dispersa.

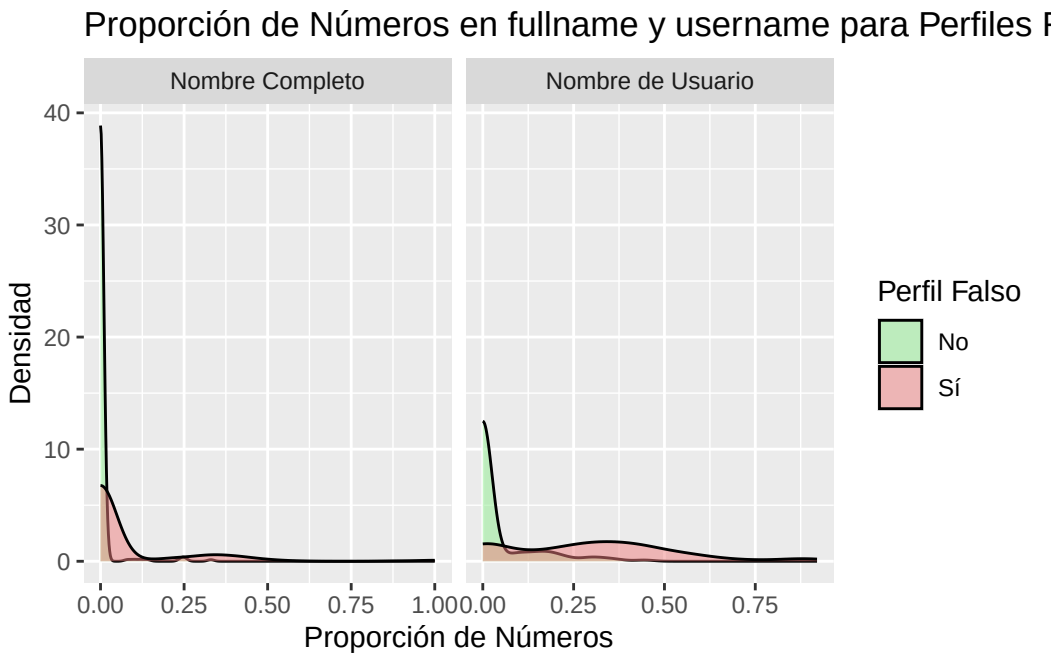
2.11 Importancia de la presencia de caracteres numéricos en el usuario y nombre.

Encontrar caracteres numéricos en el nombre de usuario y en los nombres completos es algo que, a primera vista, no podemos asociar con ningún tipo de cuenta. Por lo tanto, nos vemos en la necesidad de analizarlo más 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_
```



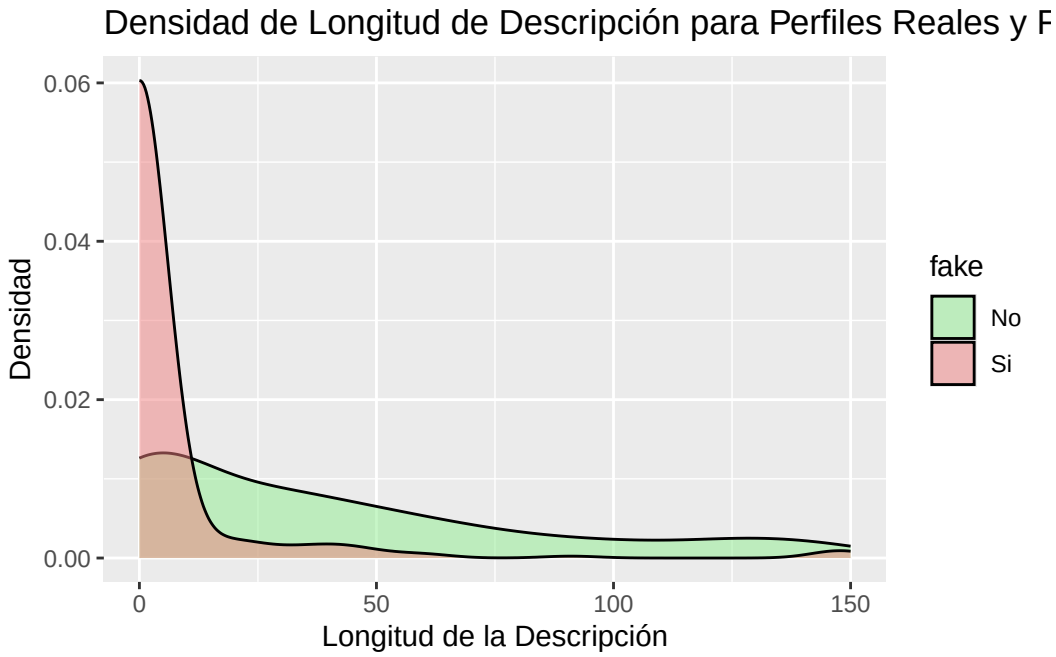
Podemos ver que realmente hay una relación entre la presencia de caracteres numéricos en el nombre y el nombre de usuario con respecto a si la cuenta es verdadera o spammer.

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

2.12 Relación entre longitud de la descripción para perfiles reales y perfiles falsos.

Por último, otra posible hipótesis podría ser 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"))
```

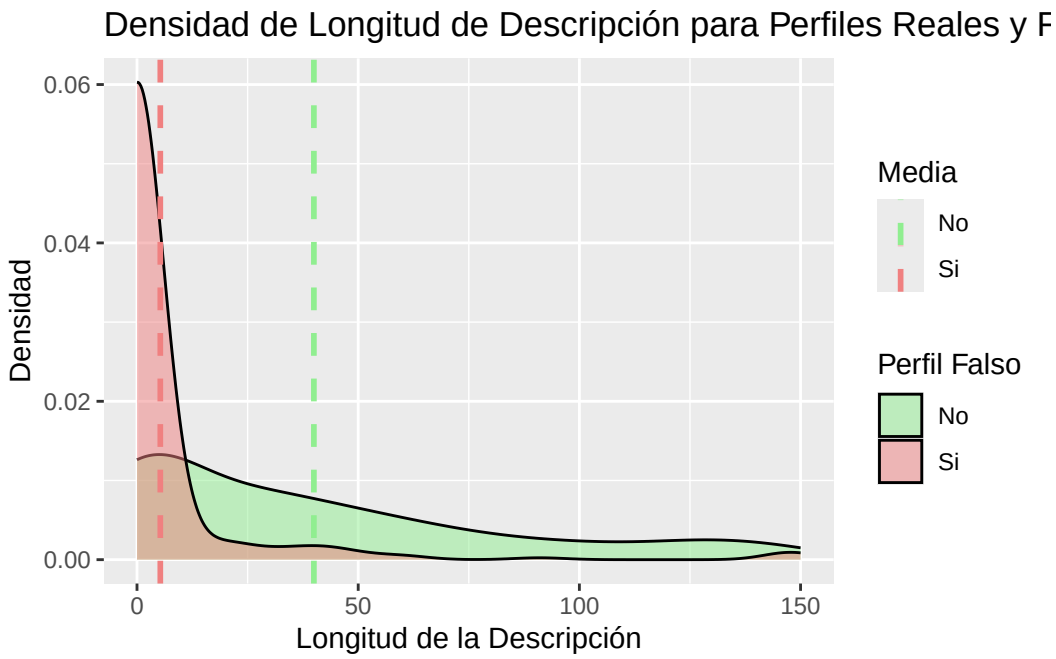


Y podemos comprobar que dicha idea era cierta. Los perfiles falsos suelen tener un número reducido de caracteres en su descripción, mientras que los perfiles reales están más repartidos.

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")
```



2.13 Conclusiones:

1. Las cuentas falsas tienden a tener un menor número de seguidores y un mayor número de seguidos.
2. Las cuentas reales tienen descripciones con longitudes más largas en comparación con las cuentas falsas.
3. Las cuentas falsas tienen una mayor cantidad de caracteres numéricos en el nombre completo y nombre de usuario en comparación con las cuentas reales.
4. Las cuentas reales suelen tener foto de perfil, mientras que las cuentas falsas pueden carecer de ella en muchos casos.

3 Reglas de asociación

Vamos a utilizar reglas de asociación para detectar cuentas falsas en Instagram. Este método nos permitirá descubrir relaciones interesantes entre diferentes características observadas en el dataset. Utilizaremos el paquete arules para llevar a cabo estas operaciones.

3.1 Características importantes:

3.1.1 Medidas relevantes

Para evaluar la calidad y relevancia de las reglas de asociación, utilizaremos las siguientes medidas:

1. Soporte (Support): Mide la proporción de cuentas en el dataset que contienen ambos conjuntos de características A y B. Un alto soporte 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): Mide cuán frecuentemente las características en B aparecen en las cuentas que contienen A. Una mayor confianza indica una mayor fiabilidad de que la presencia de las características en A implicará la presencia de las características en B.
3. Elevación (Lift): 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

Utilizaremos el algoritmo Apriori para obtener reglas a partir de nuestros datos. Este algoritmo se basa en la propiedad de que cualquier subconjunto de un conjunto frecuente también debe ser frecuente. 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 seguidas y no tiene foto de perfil, entonces es probable que sea una cuenta falsa".

3.2 Carga de datos:

Vamos a cargar las librerías 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 cada variable, para ello usaremos las funciones `ordered` y `cut`. Además, las variables que son binarias como "fake", vamos a ponerles "Si" o "No" para poder comprenderlas mejor.

```
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"))
}
```

```
# 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", "extrema

# 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.l

# 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 a las que queremos asignar un valor personalizado.

```
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.

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

3.5 Generar reglas

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

```r
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].
```

```r
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 cribarlas 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 falsas de las verdaderas, por lo tanto, vamos a centrar nuestro análisis en esos dos atributos: "fake=Si" y "fake=No". Como tenemos diferentes métricas, vamos a analizarlas por separado:

3.7.1 Soporte

Vamos primero a analizar las reglas ordenándolas por el soporte. Recordamos que 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.

```
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 vamos a analizar las reglas ordenándolas por la confianza. 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   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
[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 último, vamos a analizar las reglas ordenándolas primero por el lift de las 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 2.059, lo que sugiere que la aparición de "external URL=Si" es aproximadamente 2 veces más probable cuando se dan las condiciones en el antecedente.

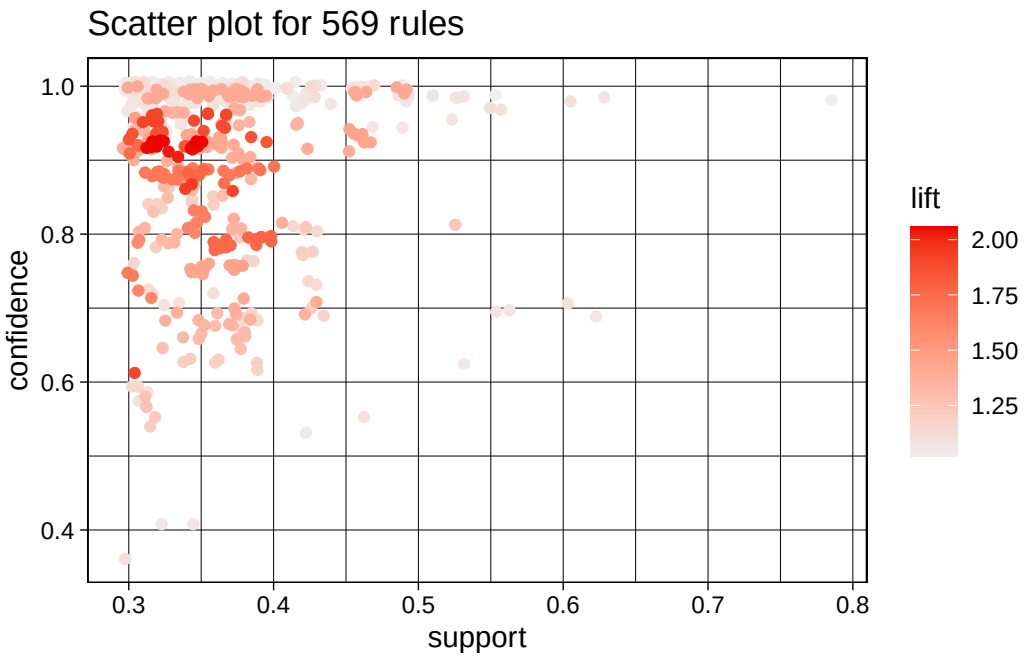
```
r2 <- subset(rules, subset = rhs %in% c("fake=Si"))
inspect(head(r2))
```

```
    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   2
```

3.8 Visualización de reglas

```
plot(rules)
```

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


Scatter plot for 569 rules
confidence
1.0
0.8
0.6
0.4
0.3
0.4
0.5
0.6
0.7
0.8
support
lift
2.00
1.75
1.50
1.25

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 principales aplicaciones de FCA son la extracción de conocimiento, agrupamiento y clasificación, aprendizaje automático, conceptos, ontologías, reglas, reglas de asociación e 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 columnas como "Tiene foto de perfil, No es 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 entre sí, como por ejemplo, los atributos que son “Sí” 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 intervalo

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 generarían conceptos demasiado largos. Por lo tanto, 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 .

4.2.0.1 Cálculo 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 índices de las cuentas que tienen dichos atributos. Sin embargo, esta información no nos es útil. Vamos a calcular el "extent" del atributo "fake = Si", y veremos que nos devuelve los índices de todas las cuentas que son falsas.

```
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`.

```
fc_datos$find_implications()
```

¿Cuántas implicaciones se han extraído?

```
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 número de implicaciones, vamos a intentar reducirlas y quedarnos con las más importantes aplicando técnicas de simplificación.

4.3.2 Cálculo de la media de la parte izquierda y derecha de las implicaciones

Este cálculo 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.

4.3.3 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 número de implicaciones no se ha reducido como podríamos haber pensado. Esto se debe a que al simplificar, realmente no se reduce la cantidad de implicaciones, sino los atributos de estas, eliminando verdades absolutas u otros parámetros redundantes.

4.3.4 Eliminar la redundancia

También vamos 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, después 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}
```

Entendido, aquí está la corrección:

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

Vamos a ver también 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 de lo anterior, si sigue a poca gente y mucha gente le sigue, 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 dé 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 más difícil de conseguir para cuentas falsas.

4.4 Funciones interesantes

Dentro del paquete fcaR hay funciones interesantes para 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 con conjuntos de datos de entrenamiento y prueba, utilizando métricas para asegurar su efectividad. Gracias a que tenemos dos DataSets, *train* y *test,* podemos probar nuestro modelo con datos nuevos. Finalmente, interpretamos los resultados para identificar las variables más influyentes y ajustamos el modelo para mejorar su precisión, intentando crear una herramienta fiable de detección de cuentas falsas.

Ahora vamos a comenzar con

```
library(readr)
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
```

```
library(ggplot2)
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.
```

```
datosTest <- read_csv("Data/test.csv")
```

```
Rows: 120 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.
```

Antes de intentar modelo de regresión, se debe explorar cuales son las correlaciones entre las variables numéricas.

```
cor(datos[c("nums/length username","fullname words","description length","#posts","#followers
```

```
                     nums/length username fullname words description length
nums/length username           1.00000000    -0.22547213       -0.321170271
fullname words                -0.22547213     1.00000000        0.272522165
description length            -0.32117027     0.27252216        1.000000000
#posts                        -0.15744211     0.07335018        0.144823702
#followers                    -0.06278509     0.03322460        0.005929455
#follows                      -0.17241327     0.09485496        0.226561422
                          #posts   #followers    #follows
nums/length username -0.15744211 -0.062785090 -0.17241327
fullname words        0.07335018  0.033224604  0.09485496
description length    0.14482370  0.005929455  0.22656142
#posts                1.00000000  0.321385480  0.09822504
#followers            0.32138548  1.000000000 -0.01106599
#follows              0.09822504 -0.011065994  1.00000000
```

Vamos a ordenar las correlaciones de mayor a menor y destacar las más significativas:

1. **description length y nums/length username**: -0.32117027

2. **#followers y #posts**: 0.32138548
3. **description length y fullname words**: 0.272522165
4. **description length y #follows**: 0.226561422
5. **nums/length username y fullname words**: -0.22547213
6. **nums/length username y #follows**: -0.17241327
7. **fullname words y description length**: 0.272522165
8. **fullname words y nums/length username**: -0.22547213

Estos valores nos indican las variables que tienen más relación entre sí. Es decir, las correlaciones altas señalan que cuando una variable cambia, la otra tiende a cambiar en la misma dirección o en dirección opuesta.

Vamos a emplear la librería `psych` para visualizar estas correlaciones de manera más intuitiva.

La parte superior de la visualización corresponde a la matriz de correlación. La diagonal muestra histogramas y además añade óvalos indicando la fuerza de correlación. Cuanto más se estire la elipse, más fuerte será la correlación. Cuanto más redondo el óvalo, más débil la correlación.

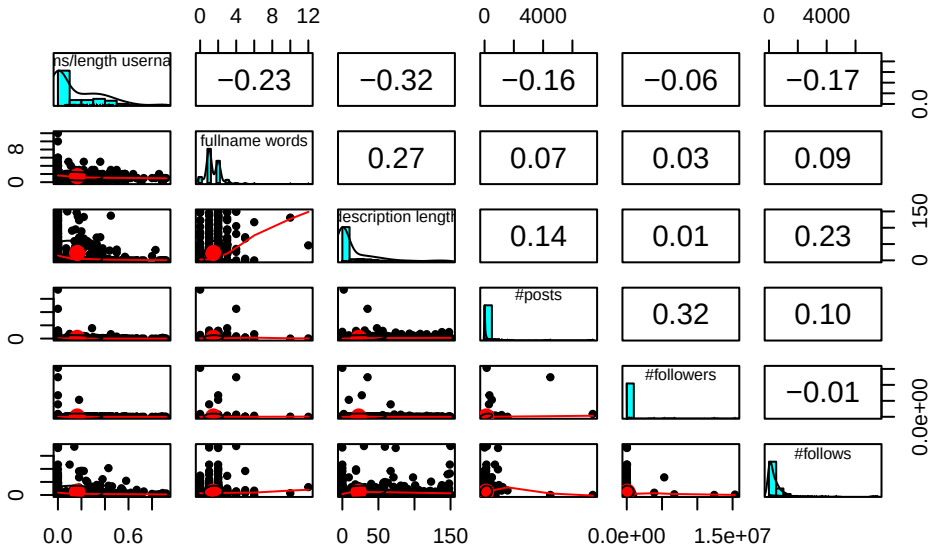
```
library(psych)
```

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

```
Attaching package: 'psych'
```

```
The following objects are masked from 'package:ggplot2':

    %+%, alpha
```

```
pairs.panels(datos[c("nums/length username","fullname words","description length","#posts","#
```



Ciertamente podemos ver que donde hay mayor elipse es en *description length* y *nums/length username* y en *followers y #posts*. Por lo tanto, las tendremos mas presentes para nuestro futuro modelo de regresión.

5.1 Construcción del modelo

Vamos a construir un primer modelo, donde vamos a enfrentar el atributo fake a todas las demás variables. Aunque seguramente no sea el mejor modelo, nos dará una primera idea de cómo podemos ir mejorándolo.

```
modelo1 <- lm(fake ~., data = datos)
summary(modelo1)
```

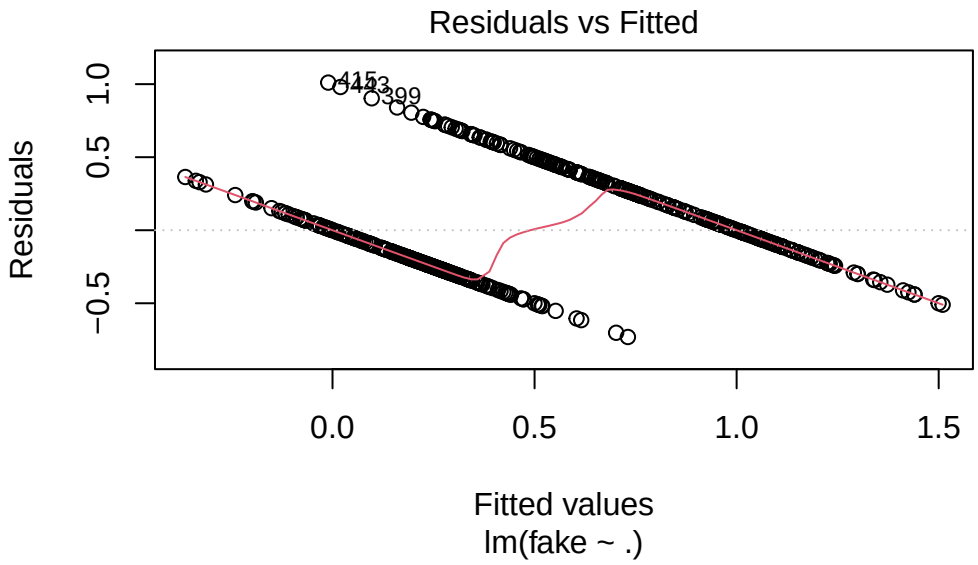
```
Call:
lm(formula = fake ~ ., data = datos)

Residuals:
     Min       1Q   Median       3Q      Max
-0.73096 -0.23729 -0.06653  0.24048  1.01052
```

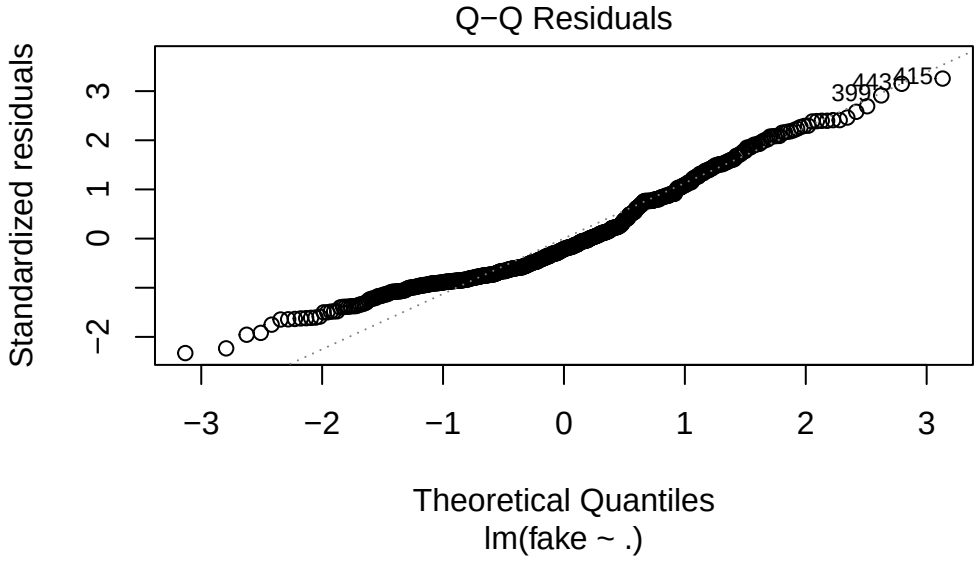
```
Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)              7.931e-01  3.798e-02  20.880  < 2e-16 ***
`profile pic`           -4.380e-01  3.345e-02 -13.094  < 2e-16 ***
`nums/length username`   8.062e-01  7.522e-02  10.718  < 2e-16 ***
`fullname words`        -3.354e-02  1.333e-02  -2.516 0.012142 *
`nums/length fullname` -2.775e-02  1.212e-01  -0.229 0.818988
`name==username`         2.241e-01  7.641e-02   2.933 0.003498 **
`description length`    -1.510e-03  4.342e-04  -3.478 0.000544 ***
`external URL`          -1.542e-01  4.800e-02  -3.213 0.001390 **
private                 -9.459e-03  2.843e-02  -0.333 0.739459
`#posts`                -9.094e-05  3.570e-05  -2.547 0.011120 *
`#followers`            -9.960e-09  1.539e-08  -0.647 0.517743
`#follows`              -1.850e-05  1.499e-05  -1.235 0.217530
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3166 on 564 degrees of freedom
Multiple R-squared:  0.6074,    Adjusted R-squared:  0.5998
F-statistic: 79.33 on 11 and 564 DF,  p-value: < 2.2e-16
```

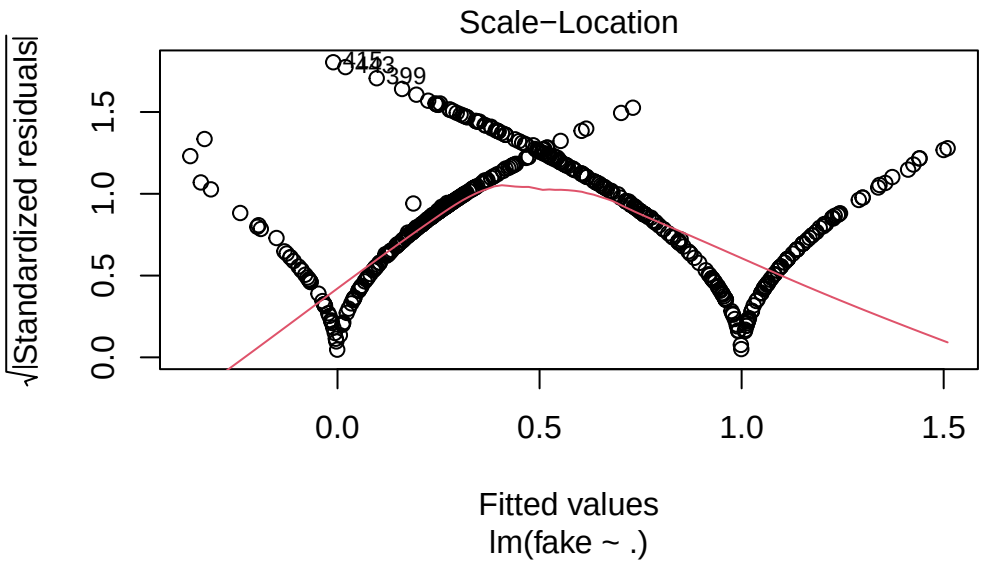
Vemos que obtenemos un modelo que no tiene un mal valor de R-squared, pero sigue siendo bajo. Además, ya podemos visualizar variables que se podrían eliminar. Esto se deduce de ver que su p-value es alto, como por ejemplo en el atributo private. Además de tener un residuo alto.

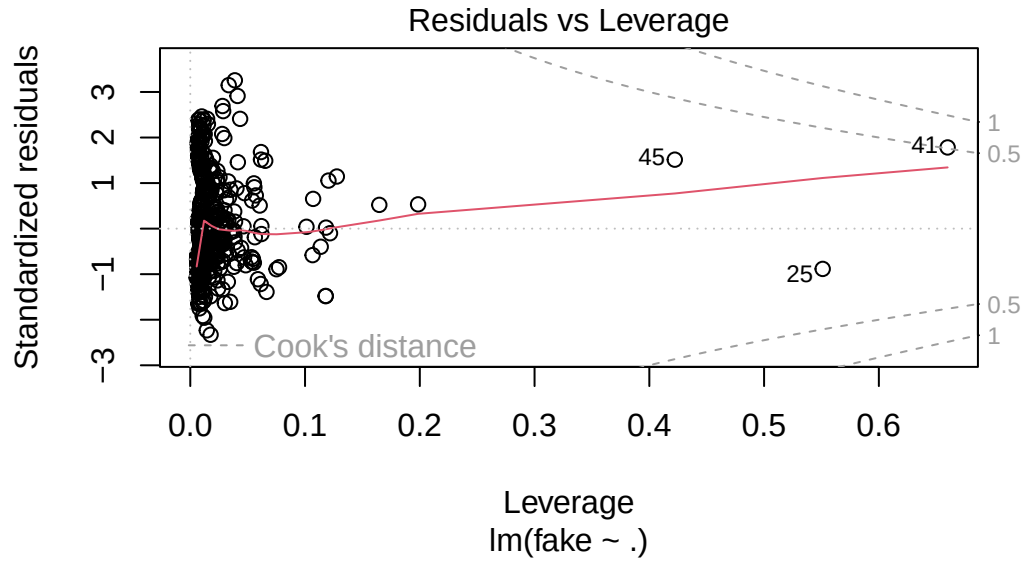
Vamos a ver gráficas sobre el modelo, donde podemos ver los residuos intuitivamente:

```
plot(modelo1)
```


Residuals vs Fitted
Residuals
1.0
0.5
−0.5
415
443
399
0.0
0.5
1.0
1.5
Fitted values
lm(fake ~ .)


Q−Q Residuals
Standardized residuals
3
2
1
0
−2
443
399
415
−3
−2
−1
0
1
2
3
Theoretical Quantiles
lm(fake ~ .)

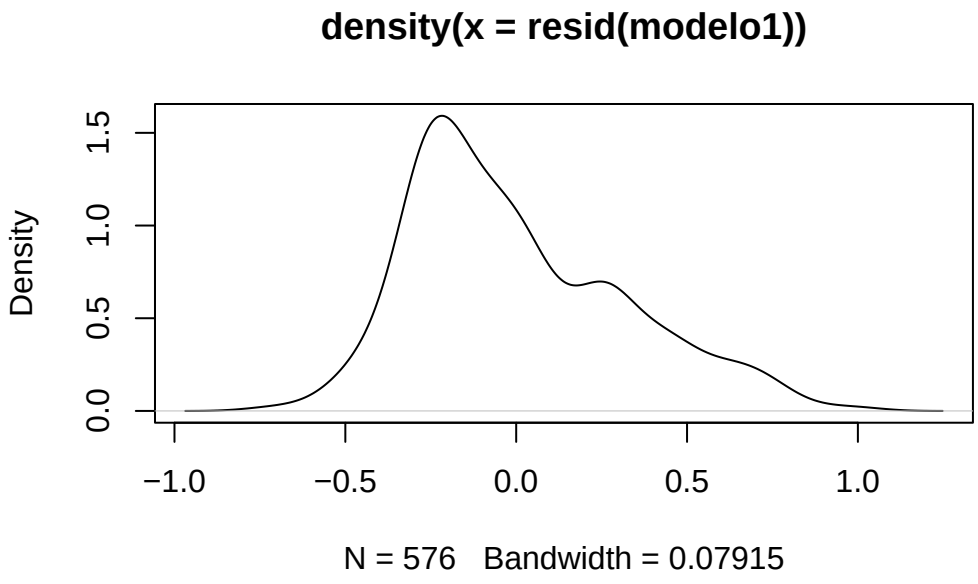




Podemos utilizar la gráfica de Residuals vs Leverage para ver la influencia de los puntos en nuestro modelo. Con esta información, observamos que en general no hay muchos puntos que afecten al modelo, los llamados "outliers"; solo podemos distinguir el 45, 25 y 41, los cuales pueden ser eliminados para mejorar el modelo.

Podemos visualizar la distribución de los residuos para evaluar si estos se comportan de manera aproximadamente normal, un supuesto común en muchos modelos estadísticos.

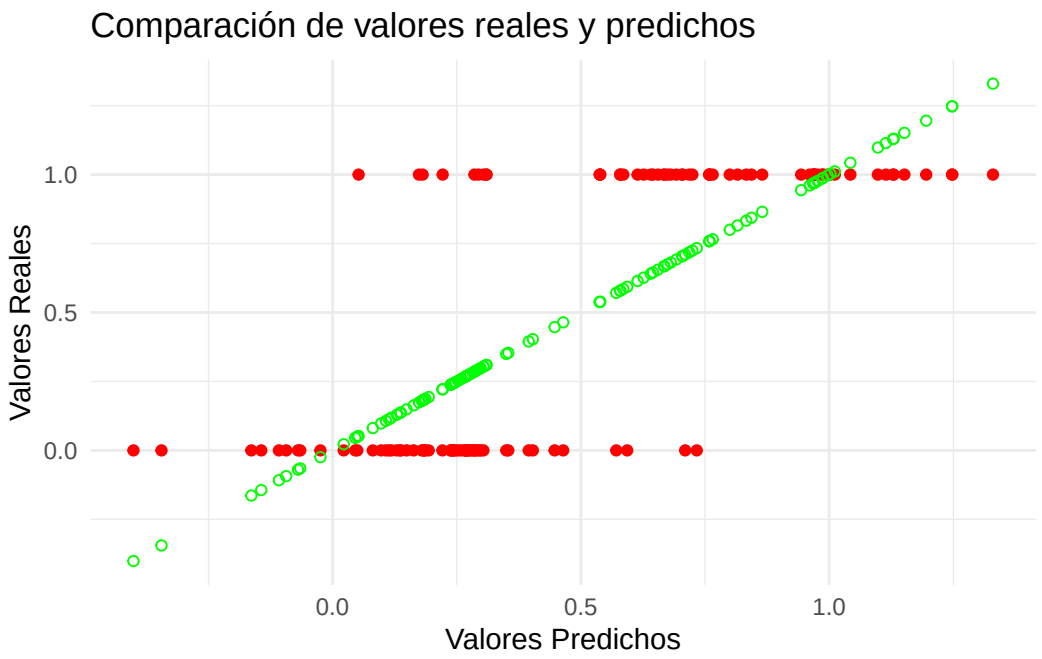
```
plot(density(resid(modelo1)))
```



La forma de la gráfica sugiere que los residuos del modelo no se distribuyen de forma normal y que podría haber problemas con el modelo.

Por último, vamos a ver el modelo prediciendo gráficamente. Podemos utilizar los datos de prueba que nos proporciona nuestro dataset.

```
modelo1_predic <- predict(modelo1, newdata = datosTest)

datosTest1 <- datosTest %>% mutate(pred = modelo1_predic)

# Rojos -> Reales, verdes -> Predichos
ggplot(datosTest1, aes(x = pred, y = fake)) +
  geom_point(color = "red") +
  geom_point(aes(x = pred, y = pred),
             color = "green", shape = 1) +
  labs(title = "Comparación de valores reales y predichos",
       x = "Valores Predichos",
       y = "Valores Reales") +
  theme_minimal()
```



Vemos que ciertamente, al usar una regresión "lineal", los valores se disponen en una linea recta, la que corresponde a la ecuación obtenida gracias a `lm`.

```
coef(modelo1)
```

```
          (Intercept)          `profile pic` `nums/length username`
         7.930850e-01          -4.380333e-01           8.062047e-01
    `fullname words` `nums/length fullname`       `name==username`
         -3.354355e-02          -2.775272e-02           2.240837e-01
`description length`         `external URL`                private
         -1.510047e-03          -1.542026e-01          -9.459373e-03
              `#posts`           `#followers`              `#follows`
         -9.094368e-05          -9.960487e-09          -1.850049e-05
```

Una vez visto que nuestro modelo inicial, con todas las variables, no es del todo bueno, vamos a eliminar variables con p-values altos, outliers, … e intentar mejorarlo.

5.2 Mejorando el modelo

5.2.1 Eliminando Outliers

Vamos a eliminar los valores que están muy separados y que pueden afectar al modelo.

```r
datos <- datos[-c(45,25,41),]
datos <- datos[-c(440,412,396),]
datos <- datos[-c(351,364,174),]
datos <- datos[-c(140,446,449),]
```

Hay que hacerlo con moderación ya que si eliminamos muchos valores que realmente no son "outliers" estamos obteniendo mejores modelos pero que realmente no son así.

5.3 Eliminar variables no significativas

```r
modelo2 <- lm(fake ~
                `profile pic`+
                `nums/length username`+
                `fullname words`+
                `name==username`+
                `description length`+
                `external URL`+
                `#posts`    , data = datos)
summary(modelo2)
```

```
Call:
lm(formula = fake ~ `profile pic` + `nums/length username` +
    `fullname words` + `name==username` + `description length` +
    `external URL` + `#posts`, data = datos)

Residuals:
     Min       1Q   Median       3Q      Max
-0.74205 -0.22951 -0.06475  0.22687  0.83615

Coefficients:
                         Estimate Std. Error t value Pr(>|t|)
(Intercept)             8.144e-01  3.594e-02  22.656  < 2e-16 ***
`profile pic`          -4.297e-01  3.140e-02 -13.685  < 2e-16 ***
`nums/length username`  7.921e-01  6.561e-02  12.072  < 2e-16 ***
`fullname words`       -5.062e-02  1.573e-02  -3.218  0.00136 **
`name==username`        1.471e-01  7.287e-02   2.019  0.04397 *
`description length`   -2.251e-03  4.363e-04  -5.160 3.45e-07 ***
`external URL`         -5.622e-02  4.740e-02  -1.186  0.23609
```

```
`#posts`               -3.114e-04  7.568e-05  -4.115 4.46e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3003 on 556 degrees of freedom
Multiple R-squared:  0.6443,    Adjusted R-squared:  0.6399
F-statistic: 143.9 on 7 and 556 DF,  p-value: < 2.2e-16
```

Hemos obtenido un modelo un poco mejor y con menos residuos, ahora vamos a intentar mejorar este modelo usando variables no lineales.

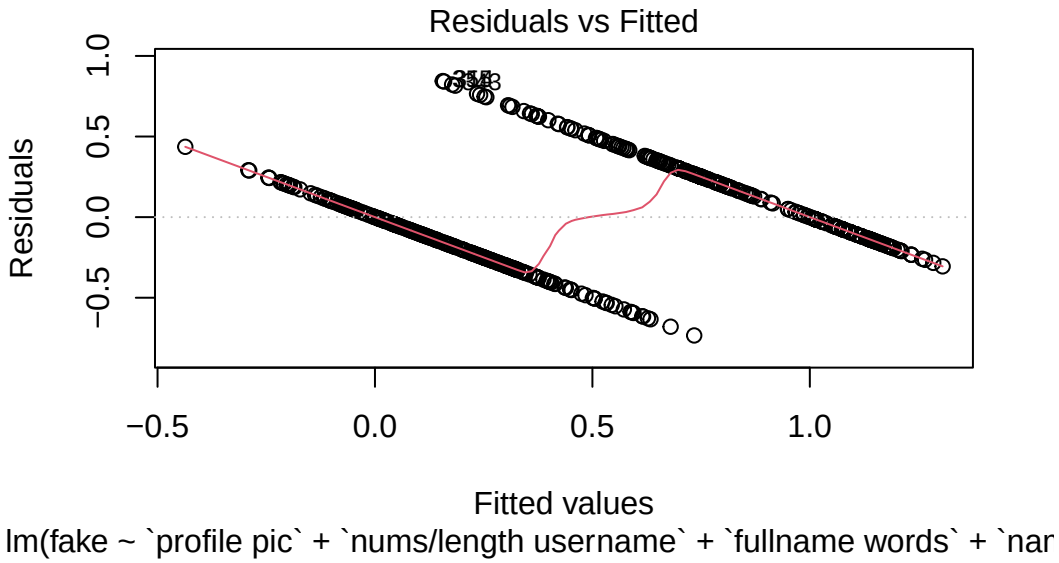
```
modelo3 <- lm(fake ~
                `profile pic` +
                `nums/length username` +
                `fullname words` +
                `name==username` +
                `description length` +
                `external URL` +
                `#posts` +
                I(`nums/length username`^2)+
                I(`description length`^2)+
                I(`#posts`^2),
              data = datos)

summary(modelo3)
```

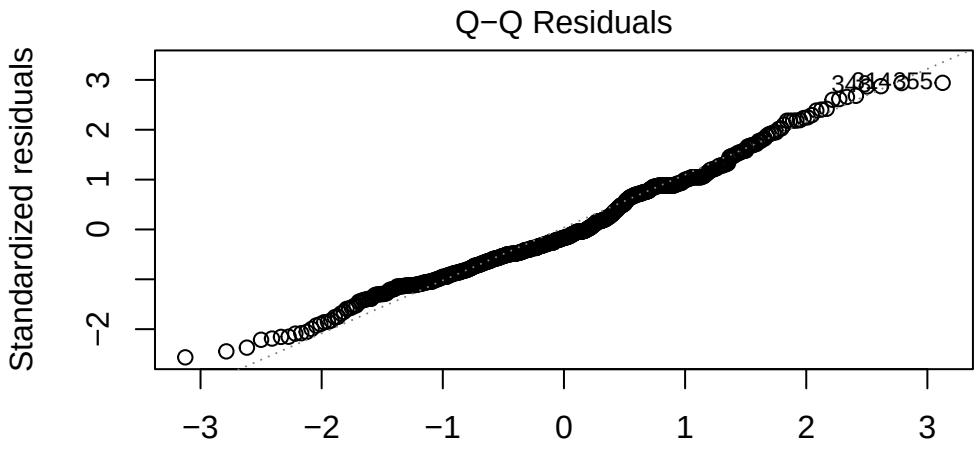
```
Call:
lm(formula = fake ~ `profile pic` + `nums/length username` +
    `fullname words` + `name==username` + `description length` +
    `external URL` + `#posts` + I(`nums/length username`^2) +
    I(`description length`^2) + I(`#posts`^2), data = datos)

Residuals:
    Min      1Q  Median      3Q     Max
-0.7340 -0.1915 -0.0474  0.2155  0.8442

Coefficients:
                              Estimate Std. Error t value Pr(>|t|)
(Intercept)                  7.954e-01  3.627e-02  21.933  < 2e-16 ***
`profile pic`               -3.725e-01  3.123e-02 -11.925  < 2e-16 ***
`nums/length username`       1.281e+00  1.547e-01   8.283 9.11e-16 ***
```

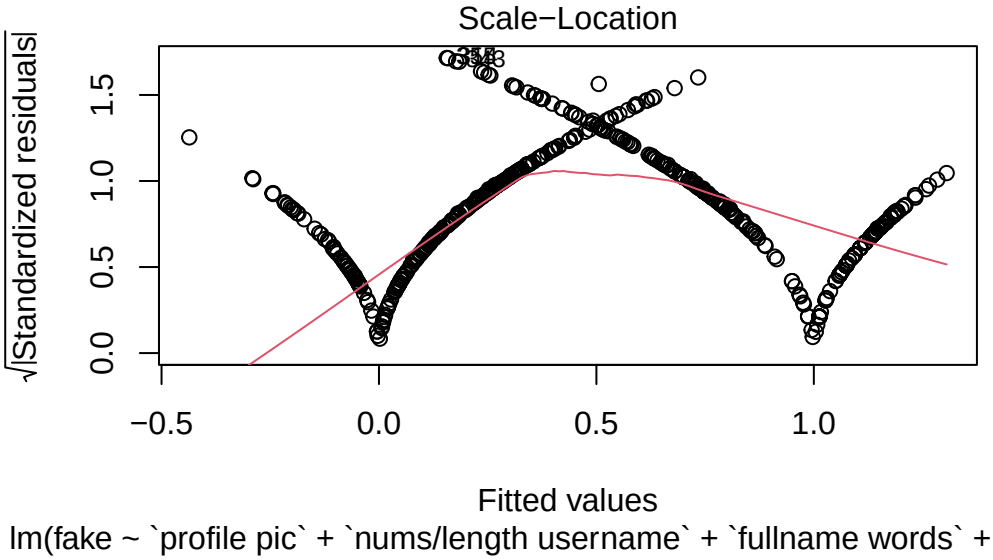
```
`fullname words`             -4.750e-02  1.510e-02  -3.145  0.00175 **
`name==username`              1.168e-01  7.019e-02   1.664  0.09663 .
`description length`         -6.116e-03  1.099e-03  -5.568 4.03e-08 ***
`external URL`               -3.135e-02  4.616e-02  -0.679  0.49739
`#posts`                     -8.571e-04  1.760e-04  -4.871 1.45e-06 ***
I(`nums/length username`^2) -8.867e-01  2.235e-01  -3.967 8.23e-05 ***
I(`description length`^2)    3.476e-05  8.677e-06   4.006 7.03e-05 ***
I(`#posts`^2)                6.033e-07  1.666e-07   3.621  0.00032 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.288 on 553 degrees of freedom
Multiple R-squared:  0.6747,    Adjusted R-squared:  0.6688
F-statistic: 114.7 on 10 and 553 DF,  p-value: < 2.2e-16
```

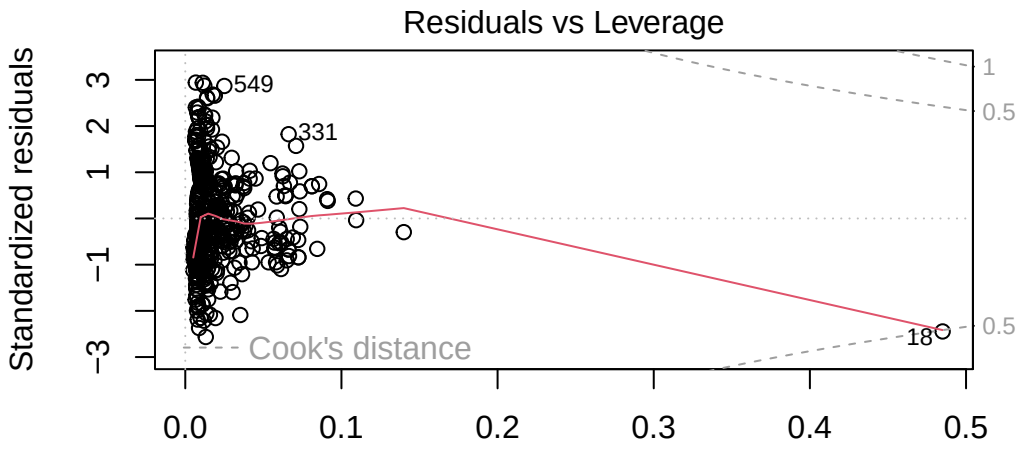
Vemos, que nuestro modelo ha mejorado un poco y tenemos menos residuos, que es lo que estamos buscando.

```
plot(modelo3)
```

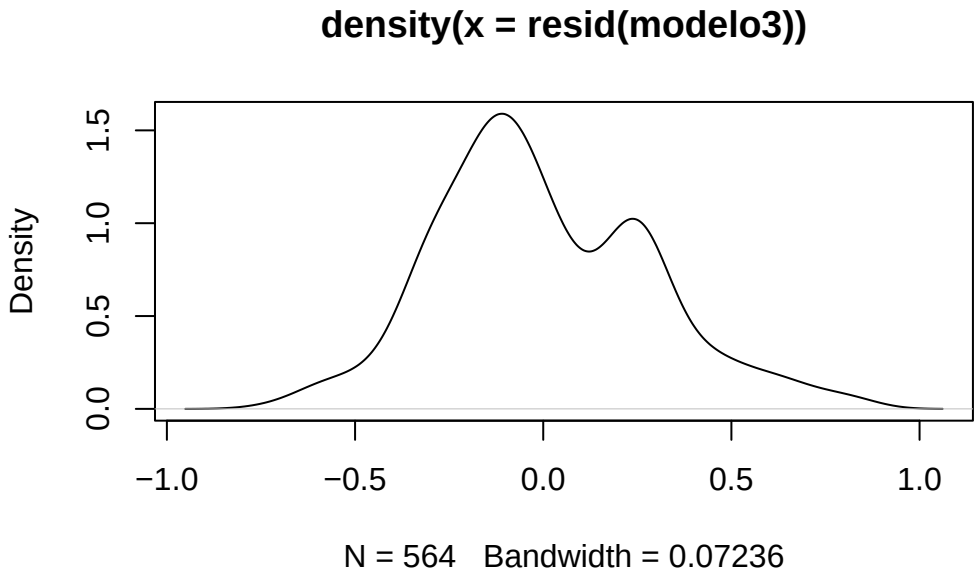



Q−Q Residuals
Standardized residuals
−2
0
1
2
3
−3
−2
−1
0
1
2
3
Theoretical Quantiles
lm(fake ~ `profile pic` + `nums/length username` + `fullname words` + `nar


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
−0.5
0.0
0.5
1.0
Fitted values
lm(fake ~ `profile pic` + `nums/length username` + `fullname words` + `nar



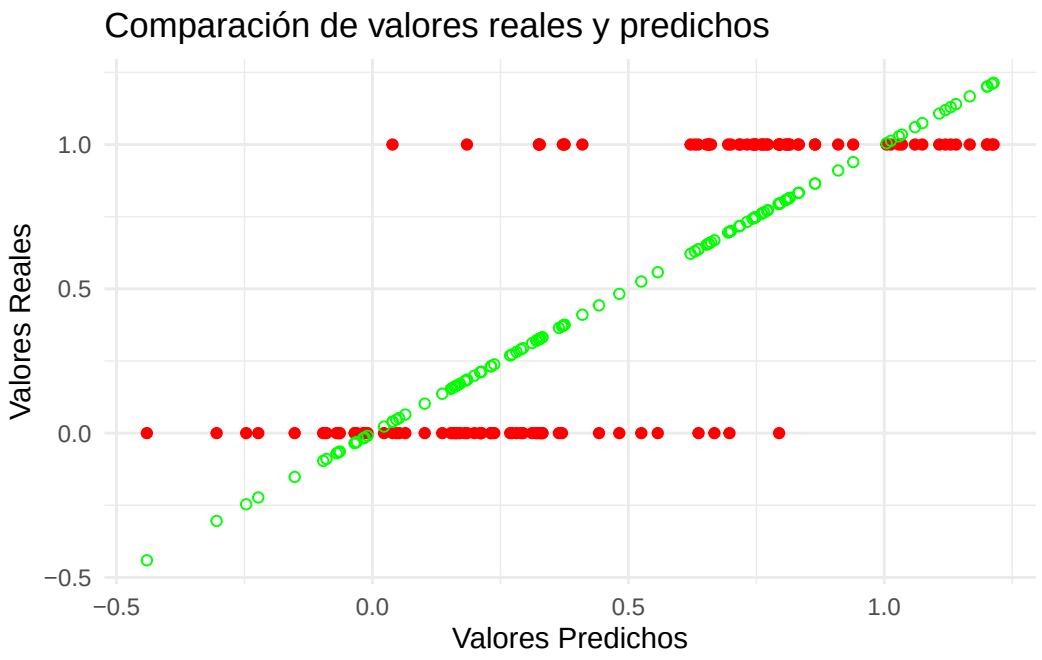

```
plot(density(resid(modelo3)))
```



Seguimos teniendo una distribución de residuos asimétrica y no normal.

Vamos a visualizar como predice este nuevo modelo:

```
modelo3_predic <- predict(modelo3, newdata = datosTest)

datosTest3 <- datosTest %>% mutate(pred = modelo3_predic)

# Rojos -> Reales, verdes -> Predichos
ggplot(datosTest3, aes(x = pred, y = fake)) +
  geom_point(color = "red") +
  geom_point(aes(x = pred, y = pred),
             color = "green", shape = 1) +
  labs(title = "Comparación de valores reales y predichos",
       x = "Valores Predichos",
       y = "Valores Reales") +
  theme_minimal()
```



Sin embargo, este gráfico, al ser los valores entre 0 y 1 es un poco confuso, vamos a ver el porcentaje de acierto mejor:

```
datosTest3 <- datosTest3 %>% mutate(pred = ifelse(modelo3_predic < 0.5, 0, 1))
# Calcular el porcentaje de aciertos
accuracy <- mean(datosTest3$pred == datosTest3$fake) * 100
accuracy
```

```
[1] 87.5
```

Vemos que ha acertado un 87.5% de las veces, un dato bastante bueno.

5.3.1 Exportar el modelo

Para poder utilizar el modelo en futuras aplicaciones, podemos guardarlo de la forma:

```
save(modelo3, file = "modelo3.rds")
```

5.4 Interacciones entre variables

Al incluir términos de interacción en el modelo de regresión, permitimos que el efecto de una variable sobre la otra varíe según los niveles de otras variables incluidas en la interacción.

Esto puede ser importante para capturar relaciones más complejas entre las variables.

```
modelo_interact <- lm(fake ~ `profile pic` * `nums/length username` +
                                      `fullname words` * `description length` +
                                      `name==username` * `external URL` +
                                      `#posts`, data = datos)
summary(modelo_interact)
```

```
Call:
lm(formula = fake ~ `profile pic` * `nums/length username` +
    `fullname words` * `description length` + `name==username` *
    `external URL` + `#posts`, data = datos)

Residuals:
     Min       1Q   Median       3Q      Max
-0.96028 -0.15321  0.00288  0.06898  0.93053

Coefficients:
                                     Estimate Std. Error t value Pr(>|t|)
(Intercept)                          1.032e+00  4.025e-02  25.644  < 2e-16
`profile pic`                       -7.191e-01  4.077e-02 -17.637  < 2e-16
`nums/length username`               4.660e-02  9.326e-02   0.500 0.617496
`fullname words`                    -5.708e-02  1.795e-02  -3.180 0.001555
`description length`                -2.705e-03  7.093e-04  -3.814 0.000152
```

```
`name==username`                          1.487e-01  6.866e-02   2.166 0.030756
`external URL`                           -3.287e-02  4.372e-02  -0.752 0.452397
`#posts`                                 -2.603e-04  6.964e-05  -3.738 0.000204
`profile pic`:`nums/length username`      1.250e+00  1.214e-01  10.297  < 2e-16
`fullname words`:`description length`     4.841e-04  3.308e-04   1.463 0.143985
`name==username`:`external URL`          -1.903e-01  2.866e-01  -0.664 0.507019

(Intercept)                              ***
`profile pic`                            ***
`nums/length username`
`fullname words`                         **
`description length`                     ***
`name==username`                         *
`external URL`
`#posts`                                 ***
`profile pic`:`nums/length username`     ***
`fullname words`:`description length`
`name==username`:`external URL`
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2749 on 553 degrees of freedom
Multiple R-squared:  0.7037,    Adjusted R-squared:  0.6984
F-statistic: 131.3 on 10 and 553 DF,  p-value: < 2.2e-16
```

```r
# Guardar el modelo en un archivo
saveRDS(modelo_interact, file = "modelo_interact.rds")
```

Este modelo vemos que ha mejorado frente a todos loa anteriores, por lo que tenemos que tenerlo en cuenta para nuestro modelo final.

5.5 Ingeniería de variables

La ingeniería de variables implica crear nuevas variables o transformar las existentes para mejorar el rendimiento de un modelo predictivo. Esto incluye crear características nuevas, transformar las existentes, entre otras técnicas.

Vamos a probarlo en nuestro modelo.

```r
modelo_nuevasVar <- lm(fake ~ `profile pic`+
                                    `nums/length username` +
                                    log(`description length` + 1) +
                                    `name==username` +
                                    log(`#posts`+1), data = datos)
summary(modelo_nuevasVar)
```

```
Call:
lm(formula = fake ~ `profile pic` + `nums/length username` +
    log(`description length` + 1) + `name==username` + log(`#posts` +
    1), data = datos)

Residuals:
     Min       1Q   Median       3Q      Max
-0.66063 -0.18128 -0.01962  0.16008  0.96220

Coefficients:
                              Estimate Std. Error t value Pr(>|t|)
(Intercept)                   0.839917   0.028606  29.362  < 2e-16 ***
`profile pic`                -0.248421   0.033391  -7.440 3.83e-13 ***
`nums/length username`        0.659306   0.061590  10.705  < 2e-16 ***
log(`description length` + 1) -0.050486   0.007843  -6.437 2.62e-10 ***
`name==username`              0.107825   0.066988   1.610    0.108
log(`#posts` + 1)            -0.080978   0.007900 -10.251  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2757 on 558 degrees of freedom
Multiple R-squared:  0.6992,    Adjusted R-squared:  0.6965
F-statistic: 259.4 on 5 and 558 DF,  p-value: < 2.2e-16
```

```r
# Guardar el modelo en un archivo
saveRDS(modelo_nuevasVar, file = "modelo_nuevasVar.rds")
```

De nuevo, este modelo ha sido mejor que todos los anteriores simplemente añadiendo el logaritmo de unas variables.

5.6 Modelo final

Vamos a combinar todos los métodos anteriores para encontrar el mejor modelo posible. Aplicaremos tanto variables no lineales como ingeniería de variables e interacción entre variables.

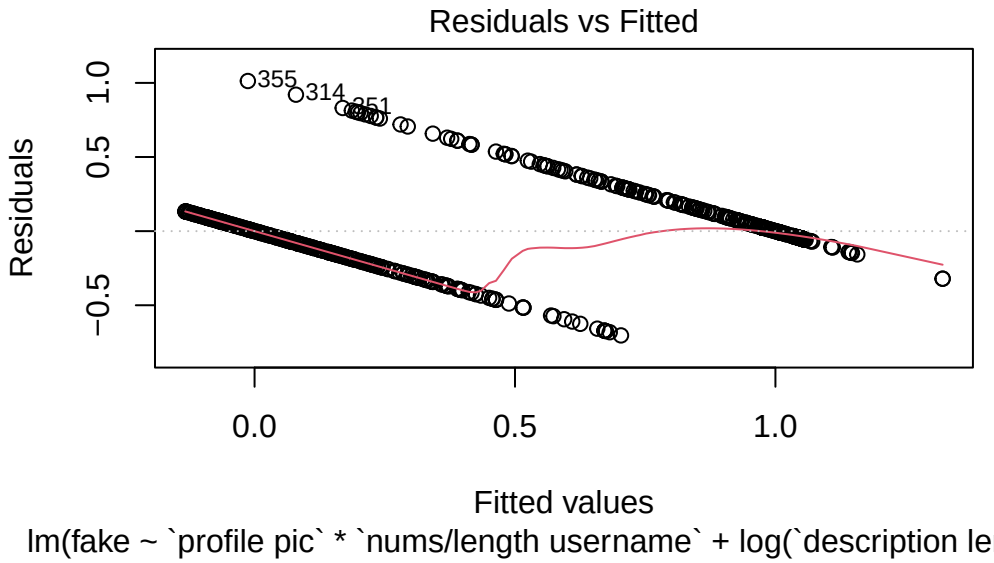
```
modelo_final <- lm(fake ~ `profile pic` * `nums/length username` +
                                log(`description length` + 1) +
                                `name==username` +
                                log(`#posts`+1)+
                                `#followers` +
                                I(`nums/length username`^2)+
                                I(`description length`^2)+
                                I(`#posts`^2),
                            data = datos)

summary(modelo_final)
```

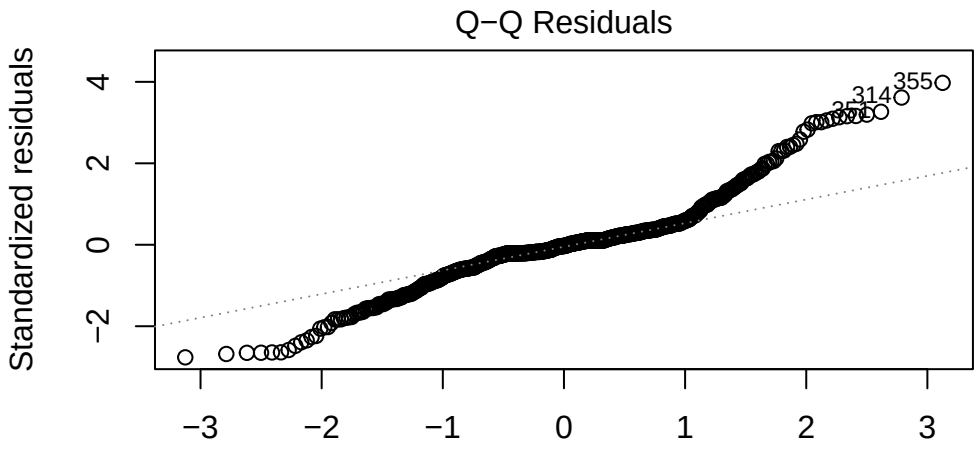
```
Call:
lm(formula = fake ~ `profile pic` * `nums/length username` +
    log(`description length` + 1) + `name==username` + log(`#posts` +
    1) + `#followers` + I(`nums/length username`^2) + I(`description length`^2) +
    I(`#posts`^2), data = datos)

Residuals:
     Min       1Q   Median       3Q      Max
-0.70351 -0.11215 -0.00771  0.08622  1.01284

Coefficients:
                                    Estimate Std. Error t value Pr(>|t|)
(Intercept)                         9.732e-01  3.395e-02  28.666  < 2e-16 ***
`profile pic`                      -5.097e-01  4.432e-02 -11.500  < 2e-16 ***
`nums/length username`              4.255e-01  1.583e-01   2.687  0.00742 **
log(`description length` + 1)      -4.738e-02  9.109e-03  -5.201 2.79e-07 ***
`name==username`                    9.404e-02  6.232e-02   1.509  0.13188
log(`#posts` + 1)                  -6.765e-02  8.414e-03  -8.041 5.44e-15 ***
`#followers`                        1.126e-09  2.733e-08   0.041  0.96716
I(`nums/length username`^2)        -5.574e-01  2.000e-01  -2.788  0.00549 **
I(`description length`^2)           2.373e-06  3.410e-06   0.696  0.48676
I(`#posts`^2)                       8.394e-08  6.657e-08   1.261  0.20789
`profile pic`:`nums/length username` 1.024e+00  1.158e-01   8.847  < 2e-16 ***
---
```

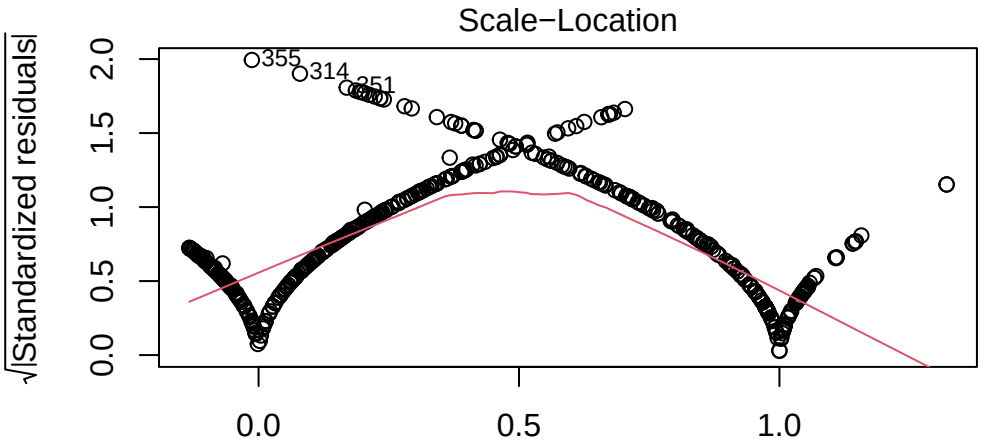
```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2557 on 553 degrees of freedom
Multiple R-squared:  0.7435,    Adjusted R-squared:  0.7389
F-statistic: 160.3 on 10 and 553 DF,  p-value: < 2.2e-16
```

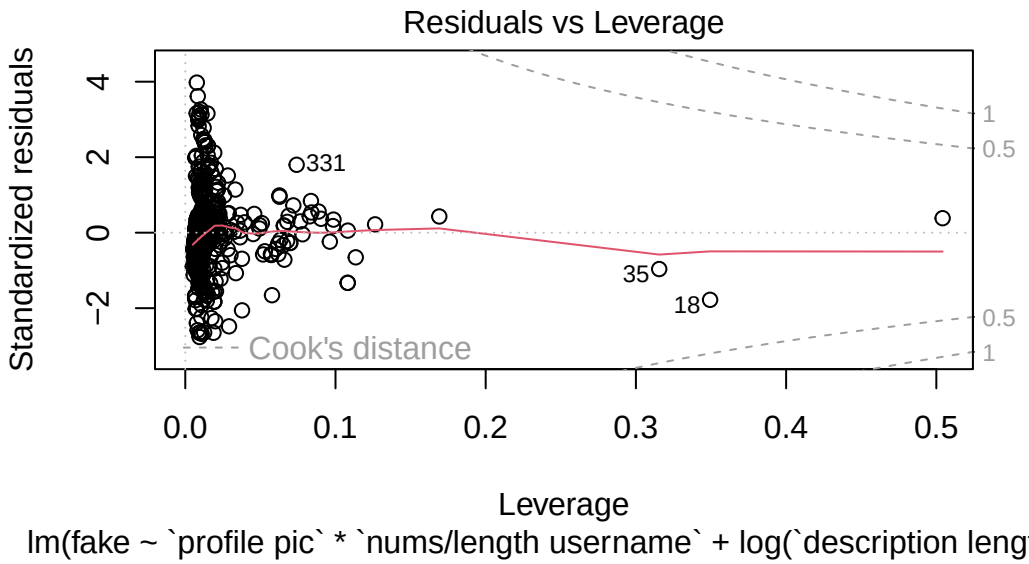
Vemos que el mejor modelo que hemos conseguido obtener ha mejorado bastante respecto al primer modelo obtenido, teniendo un mejor R cuadrado y menos residuos. Vamos a ver las demás métricas utilizadas anteriormente.

```
plot(modelo_final)
```




Q−Q Residuals
Standardized residuals
4
2
0
−2
355
314
−3
−2
−1
0
1
2
3
Theoretical Quantiles
lm(fake ~ `profile pic` * `nums/length username` + log(`description length


Scale−Location
√|Standardized residuals|
2.0
1.5
1.0
0.5
0.0
355
314
251
0.0
0.5
1.0
Fitted values
lm(fake ~ `profile pic` * `nums/length username` + log(`description length



Vamos a generar predicciones con el dataSet de test.

```
# Generar predicciones
modeloFinal_predic <- predict(modelo_final, newdata = datosTest)


datosTestFinal <- datosTest %>% mutate(pred = ifelse(modeloFinal_predic < 0.5, 0, 1))
# Calcular el porcentaje de aciertos
accuracy <- mean(datosTestFinal$pred == datosTestFinal$fake) * 100
accuracy
```

```
[1] 87.5
```

Por último, vemos que obtenemos un buen porcentaje de acierto con nuestro dataSet de prueba,

5.7 Otros modelos de regresión

Vamos a explorar otros modelos de regresión diferentes al clásico modelo de regeresion lineal que hemos estado trabajando hasta ahora. Puede ser que para nuestra investigación, un modelo diferente al lineal sea mas conveniente y nos pudiera ayudar mas.

5.7.1 Random Forest

Random Forest es un algoritmo de aprendizaje automático que se basa en la idea de crear múltiples árboles de decisión durante el proceso de entrenamiento y luego combinar sus predicciones para obtener una predicción más robusta y precisa.

```
library(randomForest)
```

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

```
randomForest 4.7-1.1
```

```
Type rfNews() to see new features/changes/bug fixes.
```

```

Attaching package: 'randomForest'
```

```
The following object is masked from 'package:psych':

    outlier
```

```
The following object is masked from 'package:ggplot2':

    margin
```

```
The following object is masked from 'package:dplyr':

    combine
```

```
datosdf <- data.frame(datos)
# Crea el modelo de Random Forest
modelo_rf <- randomForest(fake ~ .,ntree=4, data = datosdf)
```

```
Warning in randomForest.default(m, y, ...): The response has five or fewer
unique values.  Are you sure you want to do regression?
```

```
# Resumen del modelo
print(modelo_rf)
```

```
Call:
 randomForest(formula = fake ~ ., data = datosdf, ntree = 4)
               Type of random forest: regression
                     Number of trees: 4
No. of variables tried at each split: 3

          Mean of squared residuals: 0.07621379
                    % Var explained: 69.51
```

5.7.1.0.1 Importancia de las variables

```r
importance(modelo_rf)
```

```
                     IncNodePurity
profile.pic              23.7703260
nums.length.username     15.2267161
fullname.words            9.6991688
nums.length.fullname      3.2508379
name..username            0.6692343
description.length       18.2050266
external.URL              1.0490664
private                   0.8469135
X.posts                   9.8820519
X.followers              44.7077969
X.follows                 7.4875299
```

```r
modeloRF_predic <- predict(modelo_rf, newdata = data.frame(datosTest))


modeloRF_predic <- datosTest %>% mutate(pred = ifelse(modeloRF_predic < 0.5, 0, 1))
# Calcular el porcentaje de aciertos
accuracy <- mean(modeloRF_predic$pred == modeloRF_predic$fake) * 100
accuracy
```

```
[1] 92.5
```

5.7.2 Generalized Additive Model

Un GAM es un tipo de modelo estadístico que generaliza los modelos lineales al permitir relaciones no lineales entre las variables predictoras y la variable de respuesta.

En lugar de suponer una relación lineal entre las variables, los GAM permiten que cada variable explicativa tenga una relación suave con la variable de respuesta, modelada a través de funciones suaves.

```
library(mgcv)
```

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

```
Loading required package: nlme
```

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

```

Attaching package: 'nlme'
```

```
The following object is masked from 'package:dplyr':

    collapse
```

```
This is mgcv 1.9-1. For overview type 'help("mgcv-package")'.
```

```
modelo_gam = gam(fake ~ profile.pic +
                   nums.length.username +
                   fullname.words +
                   nums.length.fullname +
                   name..username +
                   description.length +
                   external.URL +
                   private +
                   X.posts +
                   X.followers +
                   X.follows,
                 data = datosdf)

summary(modelo_gam)
```

```

Family: gaussian
Link function: identity
```

```
Formula:
fake ~ profile.pic + nums.length.username + fullname.words +
    nums.length.fullname + name..username + description.length +
    external.URL + private + X.posts + X.followers + X.follows

Parametric coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)           8.190e-01  3.828e-02  21.397  < 2e-16 ***
profile.pic          -4.272e-01  3.221e-02 -13.264  < 2e-16 ***
nums.length.username  7.855e-01  7.193e-02  10.921  < 2e-16 ***
fullname.words       -5.088e-02  1.587e-02  -3.205 0.001427 **
nums.length.fullname  1.413e-02  1.160e-01   0.122 0.903121
name..username        1.461e-01  7.724e-02   1.891 0.059126 .
description.length   -2.262e-03  4.422e-04  -5.115 4.33e-07 ***
external.URL         -5.643e-02  4.800e-02  -1.176 0.240246
private              -8.206e-03  2.738e-02  -0.300 0.764497
X.posts              -3.042e-04  7.783e-05  -3.908 0.000105 ***
X.followers          -1.982e-08  3.227e-08  -0.614 0.539287
X.follows            -3.659e-06  1.456e-05  -0.251 0.801712
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1


R-sq.(adj) =  0.638   Deviance explained = 64.5%
GCV = 0.092734  Scale est. = 0.090761  n = 564
```

```r
modeloGam_predic <- predict(modelo_gam, newdata = data.frame(datosTest))


modeloGam_predic <- datosTest %>% mutate(pred = ifelse(modeloGam_predic < 0.5, 0, 1))
# Calcular el porcentaje de aciertos
accuracy <- mean(modeloGam_predic$pred == modeloGam_predic$fake) * 100
accuracy
```

```
[1] 89.16667
```

5.8 Conclusiones

Hemos explorado tanto los tradicionales modelos lineales como también nuevos enfoques de regresión. Durante este proceso, hemos descubierto modelos interesantes que muestran un

potencial considerable para generar predicciones precisas en contextos del mundo real. Al aplicar estos modelos a conjuntos de datos reales, estamos equipados para abordar problemas complejos y tenemos la herramientas para realizar predicciones certeras sobre datos reales, pudiendo servir de verdadera ayuda en el mundo real.

6 Series Temporales

Las series temporales y forecasting son dos herramientas fundamentales en el análisis de datos para identificar patrones y predecir valores futuros.

Para trabajar con series temporales vamos a necesitar los paquetes de paquetes `forecast`, y `tseries`.

```
library(tseries)
```

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

```
Registered S3 method overwritten by 'quantmod':
  method            from
  as.zoo.data.frame zoo
```

```
library(forecast)
```

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

Vamos a cargar datos

```
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.
```

Sin embargo, como su nombre indica, las series temporales necesitan un atributo que sea el tiempo, para poder ver la evolución de la variable y así analizar posibles patrones. El problema es que nuestro dataset no tiene esos tipos de datos, por lo que puede ser difícil aplicar series temporales.

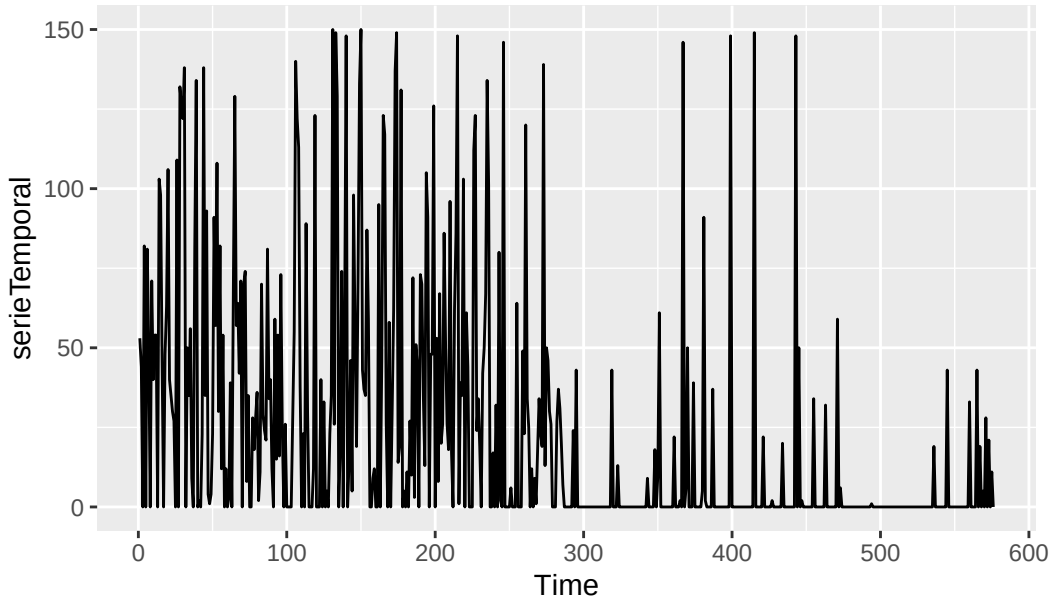
Vamos a intentar convertir alguna variable en un sustituto del tiempo para ver si podemos aplicar los conocimientos de series temporales.

```
serieTemporal <- ts(datos$`description length`)
time(serieTemporal)
```

```
Time Series:
Start = 1
End = 576
Frequency = 1
  [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
 [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
 [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
 [55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
 [73]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
 [91]  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108
[109] 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126
[127] 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144
[145] 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162
[163] 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
[181] 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198
[199] 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216
[217] 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234
[235] 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252
[253] 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270
[271] 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288
[289] 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306
[307] 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324
[325] 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342
[343] 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360
[361] 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378
[379] 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396
[397] 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414
[415] 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432
[433] 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450
[451] 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468
[469] 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486
[487] 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504
```

```
[505] 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522
[523] 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540
[541] 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558
[559] 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576
```

```
autoplot(serieTemporal)
```



Vemos que ciertamente no podemos hacer un análisis temporal de nuestros datos ya que no tenemos ese componente del tiempo.

7 Otras técnicas

7.1 Análisis de Redes Sociales

Esta técnica se basa en la teoría de grafos y matemáticas discretas. Sin embargo, para aplicarla, necesitaríamos relaciones entre los datos, como por ejemplo, relaciones entre seguidores y seguidos. Sin embargo, dichas relaciones no están presentes en nuestros datos y no tienen ninguna relación, por lo que no podemos aplicar esta técnica.

7.2 Análisis de Componentes Principales

El análisis de componentes principales es una técnica de reducción de dimensionalidad que permite pasar de una gran cantidad de variables interrelacionadas a unas pocas variables incorreladas entre sí, llamadas componentes principales.

Esta técnica sería muy útil si nuestro conjunto de datos tuviera una gran cantidad de variables, simplificando muchos procesos. Sin embargo, nuestros datos tienen pocas variables que no causan ningún problema a la hora de manejarlas.

8 Aplicación y demo

8.1 Descarga los diferentes modelo diseñado

Enlace Modelos

8.2 Demo:

Utilizando los modelo que hemos creado anteriormente, tanto los lineales, random forest, ... podemos predecir si una cuenta es fake o no:

```
#| standalone: true
#| viewerHeight: 600
library(shiny)
library(tibble)
library(readr)
library(randomForest)
library(mgcv)



# Define your Shiny UI here
ui <- fluidPage(
  titlePanel("Análisis de Perfil de Usuario"),
  sidebarLayout(
    sidebarPanel(
      checkboxInput("profile_pic", "Tiene foto de perfil?", value = FALSE),
      numericInput("username_ratio", "Cantidad de numeros en nombre del usuario:", value = 0)
      numericInput("fullname_ratio", "Cantidad de numeros en el nombre completo:", value = 0)
      numericInput("fullname_length", "Longitud del nombre completo:", value = 0),
      checkboxInput("name_equals_username", "¿El nombre es igual al nombre de usuario?", valu
      numericInput("description_length", "Longitud de la descripción:", value = 0),
      checkboxInput("external_url", "Tiene URL externa?", value = FALSE),
      checkboxInput("private", "¿Es una cuenta privada?", value = FALSE),
      numericInput("num_posts", "Número de publicaciones:", value = 0),
```

```
      numericInput("num_followers", "Número de seguidores:", value = 0),
      numericInput("num_follows", "Número de seguidos:", value = 0),
      fileInput("file1", "Elige tu modelo de regresion",
            accept = c("text/rds",
                       "text/comma-separated-values,
                       .rds")),
      actionButton("submit_button", "Analizar")
    ),
    mainPanel(
      verbatimTextOutput("results")
    )
  )
)

procesar_datos <- function(profile_pic, username_ratio, fullname_ratio, fullname_length, name
                           description_length, external_url, private, num_posts, num_followe

  datos <- tibble(`profile pic` = ifelse(profile_pic, 1, 0),
                        `nums/length username` = as.numeric(username_ratio),
                        `fullname words` = as.numeric(fullname_length),
                        `nums/length fullname` = as.numeric(fullname_ratio),
                        `name==username` = ifelse(name_equals_username, 1, 0),
                        `description length` = as.numeric(description_length) ,
                        `external URL` = ifelse(external_url, 1, 0) ,
                        `private` = ifelse(private, 1, 0) ,
                        `#posts` = as.numeric(num_posts) ,
                        `#followers` = as.numeric(num_followers) ,
                        `#follows` = as.numeric(num_follows),
                       )

  prediction <- tryCatch({
  predict(modelo_guardado, newdata = datos)
}, error = function(e) {
  datos2 <- data.frame(datos)
  return( predict(modelo_guardado, newdata = datos2))
})
  return(ifelse(prediction[[1]] <0.5 , "Cuenta real", "Cuenta falsa"))
}

# Define your Shiny server logic here
server <- function(input, output, session) {
  modelo_guardado <- reactive({
```

```
    infile <- input$file1
    if (is.null(infile)) {
      return(NULL)
    }
   readRDS(infile$datapath)
  })

  # Manejar el evento del botón de enviar
  observeEvent(input$submit_button, {
    # Llama a la función para procesar los datos y muestra los resultados
    res <- procesar_datos(input$profile_pic, input$username_ratio, input$fullname_ratio, inpu
                     input$name_equals_username, input$description_length, input$external_url
                     input$num_posts, input$num_followers, input$num_follows, modelo_guardado
    output$results <- renderPrint({
      res
    })
  })
}

# Create and launch the Shiny app
shinyApp(ui, server)
```

9 Resultados

Se han realizado diversas actividades de análisis y modelado de datos para comprender y predecir la presencia de cuentas falsas en Instagram, obteniendo insights valiosos y construyendo modelos predictivos para este propósito.

Resultados obtenidos :

1. Identificación de características clave para distinguir cuentas reales de falsas, como número de seguidores y seguidos, presencia de foto de perfil y caracteres numéricos en el nombre de usuario.
2. Descubrimiento de reglas de asociación e implicaciones que proporcionan insights para detectar cuentas falsas.
3. Mejora de los modelos de regresión mediante la eliminación de variables no significativas y la inclusión de términos de interacción.
4. Evaluación de modelos no lineales como Random Forest como alternativa al modelo lineal tradicional.

Hemos obtenido resultados que nos permiten diferenciar entre cuentas falsas y reales de forma sencilla, lo cual era nuestro objetivo principal, por lo que nuestro trabajo ha sido c

Referencias

https://amorabonilla.quarto.pub/-conocimiento-desde-los-datos/