# Práctica 2: Limpieza y análisis de datos

Autores: Fernando García Dorador, Amanda Iglesias Moreno

24/12/2020



# 1 Descripción del dataset. ¿Por qué es importante y qué pregunta/problema pretende responder?

Google Play es la plataforma de distribución digital de Android para aplicaciones móviles y desde su fecha de lanzamiento hace poco más de 10 años crece de manera imparable ofreciendo al público aplicaciones en un amplio abanico de categorías desde apps financieras hasta aplicaciones dedicadas al bienestar emocional y la salud.

El conjunto de datos analizado se encuentra en Kaggle y contiene información sobre aproximadamente 10.000 aplicaciones diponibles en Googleplay. Entre las varibles presentes en del dataset encontramos:

- **App**: nombre de la aplicación
- **Category**: categoría temática a la que pertenece (existen 33 categorías)
- **Rating**: calificación de la app por parte de los usuarios (rango de 0 a 5)
- **Reviews**: número de reviews realizadas por parte de los usuarios de Googleplay
- **Size**: dimensiones de la app
- **Installs**: número de descargas
- **Type**: variable que indica si la aplicación es gratuíta o por el contrario de pago
- **Price**: precio de la app
- **Content Rating**: grupo de edad al que esta destinado la aplicación
- **Genre**: géneros (1 o varios) al que pertenece la aplicación
- **Last updated**: fecha de la última actualización de la aplicación
- **Curent Version**: versión de la aplicación
- **Android Version**: versión Android de la aplicación

En esta práctica, examinaremos que características son más representativas de las aplicaciones de la plataforma, haciendo hincapié en el análisis de las calificaciones. A partir de este conjunto de datos se pretende analizar la distribución de las puntaciones, determinando que características influyen en las mismas así como que variables carecen de relevancia. De esta manera, futuros desarrolladores informáticos podrán utilizar estos insights para crear apps de calidad que sean reconocidas por el amplio y creciente público que hace uso de Google Play.

# 2 Integración y selección de los datos de interés a analizar.

## 2.1 Integración

Los datos con los que vamos a trabajar en esta prácica provienen de un único fichero "*googleplaystore.csv*", obtenido en la base de datos de kaggle:

```
google_play <- read.csv("googleplaystore.csv")
```

## 2.2 Selección

En este proyecto nos interesa analizar toda la población de datos, por lo que no se realizará un filtrado de los datos.

Por otra parte, nos parece interesante analizar si el hecho de que una aplicación pertenezca a más de un género afecta a su rating. A continuación, creamos la variable *Multi_Genre*, que tomará valor 1 si tiene más de un género asociado o 0 en caso contrario:

```
Multi_Genre <- lapply(google_play$Genres, function(x) ifelse(grepl(";", x, fixed = TRUE), "yes",
"no"))

google_play$Multi_Genre <- as.character(Multi_Genre)
```

# 3 Limpieza de los datos

Una vez leídos los datos, analizamos la estructura y tipología de los datos mediante la función *str*. También importamos la librería *DataExplorer* que nos ayudará a realizar un análisis exploratorio visual posteriormente:

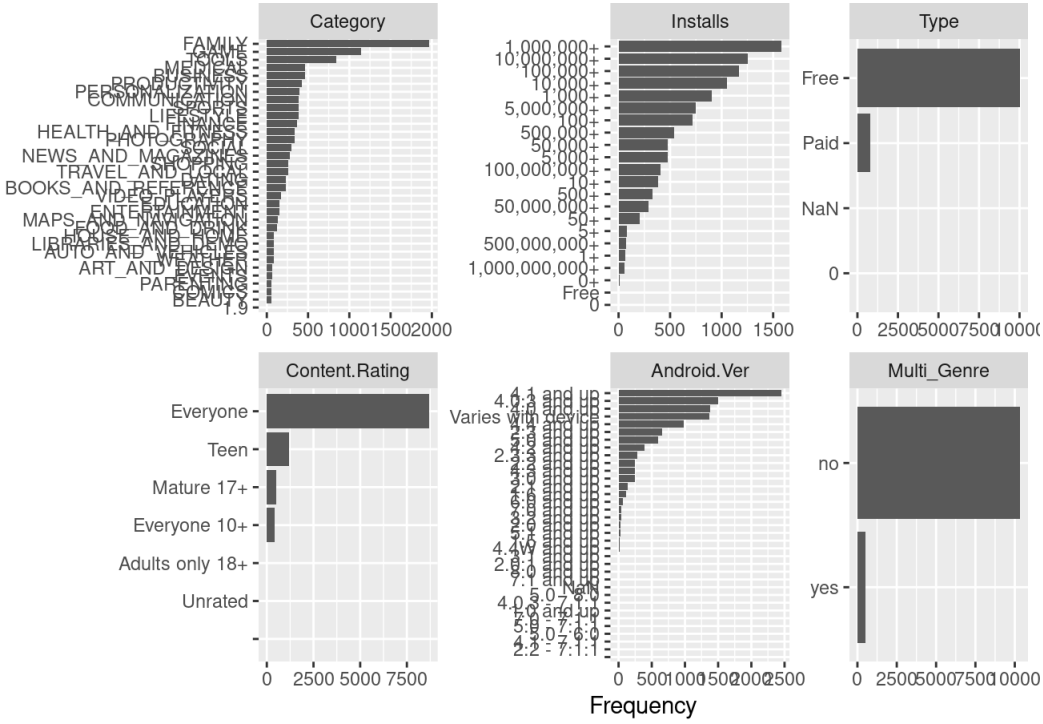
```
library(DataExplorer)
str(google_play)
```

```
## 'data.frame':    10841 obs. of  14 variables:
##  $ App           : chr  "Photo Editor & Candy Camera & Grid & ScrapBook" "Coloring book moana" "U
Launcher Lite – FREE Live Cool Themes, Hide Apps" "Sketch - Draw & Paint" ...
##  $ Category      : chr  "ART_AND_DESIGN" "ART_AND_DESIGN" "ART_AND_DESIGN" "ART_AND_DESIGN" ...
##  $ Rating        : num  4.1 3.9 4.7 4.5 4.3 4.4 3.8 4.1 4.4 4.7 ...
##  $ Reviews       : chr  "159" "967" "87510" "215644" ...
##  $ Size          : chr  "19M" "14M" "8.7M" "25M" ...
##  $ Installs      : chr  "10,000+" "500,000+" "5,000,000+" "50,000,000+" ...
##  $ Type          : chr  "Free" "Free" "Free" "Free" ...
##  $ Price         : chr  "0" "0" "0" "0" ...
##  $ Content.Rating: chr  "Everyone" "Everyone" "Everyone" "Teen" ...
##  $ Genres        : chr  "Art & Design" "Art & Design;Pretend Play" "Art & Design" "Art & Design"
...
##  $ Last.Updated  : chr  "January 7, 2018" "January 15, 2018" "August 1, 2018" "June 8, 2018" ...
##  $ Current.Ver   : chr  "1.0.0" "2.0.0" "1.2.4" "Varies with device" ...
##  $ Android.Ver   : chr  "4.0.3 and up" "4.0.3 and up" "4.0.3 and up" "4.2 and up" ...
##  $ Multi_Genre   : chr  "no" "yes" "no" "no" ...
```

Observamos que todas las variables menos *Rating* están en formato *chr*. De estas variables, algunas pueden convertirse a valor número como *Size*, *Reviews* y *Price*, y algunas otras se podrán discretizar si tienen pocos valores únicos.

A continuación realizamos un gráfico de barras de las variables con formato *chr* que tienen pocas categorías:

```
plot_bar(google_play)
```




vemos que hay 2 de las variables (Installs y Type) tienen categorías que no deberían. Analizamos en concreto las filas donde esto ocurre:

```
google_play[google_play$Installs == "Free",]
```

```
##                                             App Category Rating Reviews   Size
## 10473 Life Made WI-Fi Touchscreen Photo Frame      1.9     19    3.0M 1,000+
##       Installs Type    Price Content.Rating            Genres Last.Updated
## 10473     Free    0 Everyone               February 11, 2018       1.0.19
##       Current.Ver Android.Ver Multi_Genre
## 10473  4.0 and up                    no
```

```
#eliminamos elemento 10473, ya que las columnas no están en orden y falta un campo
googleplay_clean <- google_play[-c(10473),]
```

Eliminamos la fila donde ocurría, puesto vemos que se ha movido y faltan campos. A continuación, analizamos el caso donde Installs es 0:

```
str(googleplay_clean[googleplay_clean$Installs == "0",])
```

```
## 'data.frame':    1 obs. of  14 variables:
##  $ App           : chr "Command & Conquer: Rivals"
##  $ Category      : chr "FAMILY"
##  $ Rating        : num NaN
##  $ Reviews       : chr "0"
##  $ Size          : chr "Varies with device"
##  $ Installs      : chr "0"
##  $ Type          : chr "NaN"
##  $ Price         : chr "0"
##  $ Content.Rating: chr "Everyone 10+"
##  $ Genres        : chr "Strategy"
##  $ Last.Updated  : chr "June 28, 2018"
##  $ Current.Ver   : chr "Varies with device"
##  $ Android.Ver   : chr "Varies with device"
##  $ Multi_Genre   : chr "no"
```

```
#se trata de un error y le añadimos el + al final para que tenga la misma estructura que los demás
googleplay_clean$Installs[googleplay_clean$Installs == '0'] <- "0+"
googleplay_clean$Type[googleplay_clean$Type == "NaN"] <- "Free"

# Los datos que están en la categoría de Installs = "0+" quiere decir que no han sido instalados y p
or ello no tienen rating,
# eliminaremos estas observaciones del dataset puesto que nuestro objetivo es estudiar el rating de
aquellas aplicaciones instaladas.

googleplay_clean <- googleplay_clean[googleplay_clean$Installs != "0+",]
dim(googleplay_clean)
```

```
## [1] 10825    14
```

Hemos comprobado que se trataba de un error de formato, además se ha sustitudo el valor de Type a "Free", puesto que el precio es 0. Además, hemos decidido eliminar aquellas aplicaciones de nuestro dataset que no han sido instaladas (15 del total).

Por último, vamos a comprobar si existen valores duplicados en nuestro dataset, y en caso afirmativo, estudiaremos como tratarlos:

```
library(dplyr)
googleplay_clean <- googleplay_clean[!duplicated(googleplay_clean), ] #Con esta linea eliminamos las
filas exactamente iguales

#Añadimos un contador para ver cuantas lineas se repiten
googleplay_clean <- add_count(googleplay_clean, App)
googleplay_clean <- googleplay_clean[order(googleplay_clean$App),]
head(googleplay_clean[googleplay_clean$n > 1,])
```

```
##                         App Category Rating  Reviews Size     Installs Type
## 2698 365Scores - Live Scores   SPORTS    4.6   666521  25M  10,000,000+ Free
## 4958 365Scores - Live Scores   SPORTS    4.6   666246  25M  10,000,000+ Free
## 1469            8 Ball Pool     GAME    4.5 14198297  52M 100,000,000+ Free
## 1497            8 Ball Pool     GAME    4.5 14198602  52M 100,000,000+ Free
## 1545            8 Ball Pool     GAME    4.5 14200344  52M 100,000,000+ Free
## 1629            8 Ball Pool     GAME    4.5 14200550  52M 100,000,000+ Free
##      Price Content.Rating Genres  Last.Updated Current.Ver  Android.Ver
## 2698     0       Everyone Sports July 29, 2018       5.5.9   4.1 and up
## 4958     0       Everyone Sports July 29, 2018       5.5.9   4.1 and up
## 1469     0       Everyone Sports July 31, 2018       4.0.0 4.0.3 and up
## 1497     0       Everyone Sports July 31, 2018       4.0.0 4.0.3 and up
## 1545     0       Everyone Sports July 31, 2018       4.0.0 4.0.3 and up
## 1629     0       Everyone Sports July 31, 2018       4.0.0 4.0.3 and up
##      Multi_Genre n
## 2698          no 2
## 4958          no 2
## 1469          no 7
## 1497          no 7
## 1545          no 7
## 1629          no 7
```

Podemos observar que tras eliminar las filas exactamente iguales, aun quedan aplicaciones duplicadas con todas las columnas iguales menos *Reviews*, que varía ligeramente. Con el objetivo de obtener un estudio más preciso, escogeremos aquella fila que tiene el número máximo de reviews, que requerirá una conversión prévia de la variable a tipo numérico:

```
require(data.table)
googleplay_clean <- as.data.table(googleplay_clean)
googleplay_clean$Reviews <- as.numeric(googleplay_clean$Reviews)

googleplay_clean <- googleplay_clean[googleplay_clean[, .I[which.max(Reviews)], by=App]$V1]
#seleccionamos fila con más reviews
googleplay_clean <- googleplay_clean[, n := NULL] #eliminamos la columna contador
print(dim(googleplay_clean))
```

```
## [1] 9644   14
```

```
print(length(unique(googleplay_clean$App)))
```

```
## [1] 9644
```

Una vez eliminadas las filas duplicadas, observamos que la dimensión de nuestros datos coincide con el número de aplicaciones diferentes que tenemos.

# 3.1 Conversión de tipos de variables

## 3.1.1 Conversión de Size

Ahora realizaremos el preprocesado de la variable size, que consistirá en transformar el tipo de datos a numérico. Para ello, primero eliminaremos las "M" y "k" de los valores, que indican megabytes y kilobytes respectivamente, y sustituiremos el valor "Varies with device" por NaN, para posteriormente extrapolar este valor utilizando kNN:

```
mask1 <- grepl("[M]$", googleplay_clean$Size)
mask2 <- grepl("[k]$", googleplay_clean$Size)

googleplay_clean$Size <- lapply(googleplay_clean$Size, function(x) gsub("[M]$", "", x))
googleplay_clean$Size <- lapply(googleplay_clean$Size, function(x) gsub("[k]$", "", x))
googleplay_clean$Size <- lapply(googleplay_clean$Size, function(x) gsub("Varies with device", "NaN",
x))

googleplay_clean$Size <- as.numeric(googleplay_clean$Size)
googleplay_clean$Size <- (1000*mask2*googleplay_clean$Size) + (1000000*mask1*googleplay_clean$Size)

str(googleplay_clean$Size)
```

```
##  num [1:9644] 9100000 549000 36000000 1700000 203000 3600000 21000000 5600000 2500000 72000 ...
```

## 3.1.2 Conversión de Price

Hemos observado que la variable *Price* está en formato chr, nuestro objetivo es convertirla a valor numérico. Para conseguirlo, debemos eliminar el símbolo del $ que acompaña a los precios:

```
googleplay_clean$Price <- lapply(googleplay_clean$Price, function(x) gsub("\\$", "", x))
googleplay_clean$Price <- as.numeric(googleplay_clean$Price)

str(googleplay_clean$Price)
```

```
##  num [1:9644] 0 1.99 0 0 0 0 0 0 0 0 ...
```

## 3.1.3 Conversión de Reviews

Esta variable estaba en formato *chr* y fue convertida en formato númerico en el apartado anterior.

## 3.1.4 Conversión de Last.updated

```
library(lubridate)
Time <- lapply(googleplay_clean$Last.Updated, function(x) parse_date_time(x,orders = "mdy"))

googleplay_clean$Last.Updated <- as.POSIXct(unlist(Time), origin = '1970-01-01')
googleplay_clean$Last.Updated <- as.character(ceiling_date(googleplay_clean$Last.Updated, "day"))
head(googleplay_clean)
```

```
##                                     App       Category Rating Reviews     Size
## 1:           - Free Comics - Comic Apps         COMICS    3.5     115  9100000
## 2:                         ¡Ay Caramba!         FAMILY    NaN       0   549000
## 3:                           ¡Ay Metro!           GAME    3.8     489 36000000
## 4:                           ¿Es Vegan? FOOD_AND_DRINK    4.6     438  1700000
## 5:                                   .R          TOOLS    4.5     259   203000
## 6: "i DT" Fútbol. Todos Somos Técnicos.         SPORTS    NaN      27  3600000
##    Installs Type Price Content.Rating       Genres Last.Updated Current.Ver
## 1:  10,000+ Free  0.00     Mature 17+       Comics   2018-07-14      5.0.12
## 2:       1+ Paid  1.99       Everyone    Education   2014-06-14         1.2
## 3:  10,000+ Free  0.00   Everyone 10+       Arcade   2015-03-18     1.0.3.1
## 4:  10,000+ Free  0.00       Everyone Food & Drink   2017-08-02       2.2.3
## 5:  10,000+ Free  0.00       Everyone        Tools   2014-09-17      1.1.06
## 6:     500+ Free  0.00       Everyone       Sports   2017-10-08        0.22
##    Android.Ver Multi_Genre
## 1:  5.0 and up          no
## 2:  3.0 and up          no
## 3:  4.0 and up          no
## 4:  3.0 and up          no
## 5:  1.5 and up          no
## 6:  4.1 and up          no
```

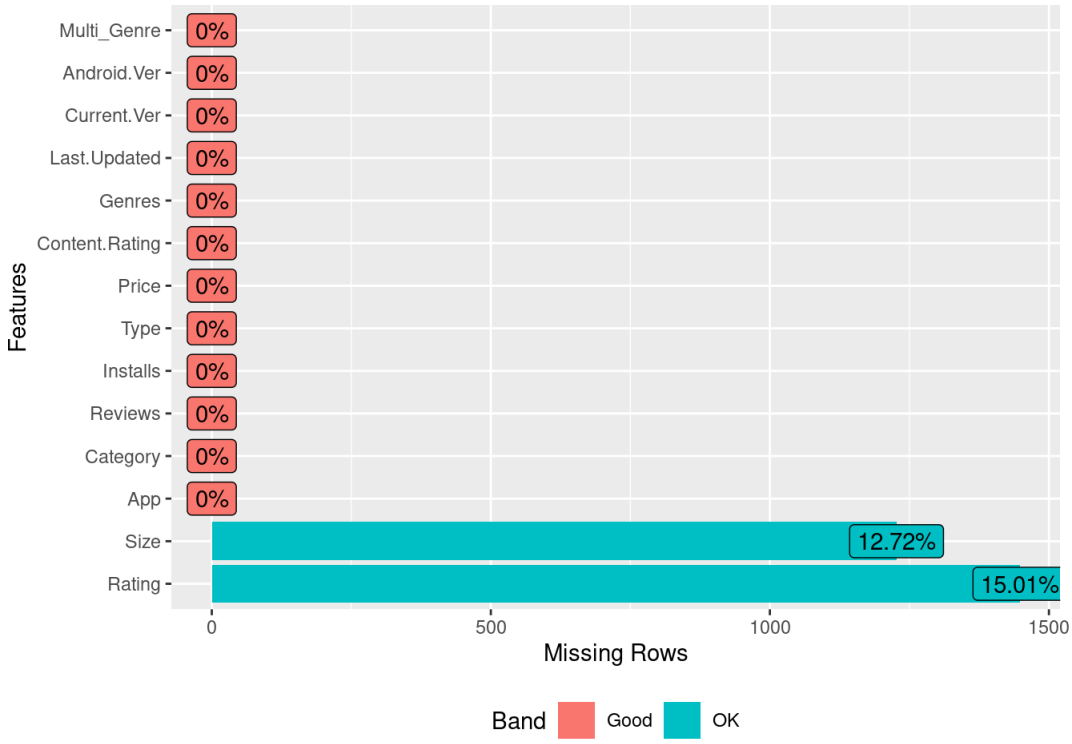
## 3.2 Análisis de valores perdidos

Ahora nos centraremos en el tratamiento de valores perdidos:

```
colSums(is.na(googleplay_clean))
```

```
##             App        Category          Rating         Reviews            Size
##               0               0            1448               0            1227
##        Installs            Type           Price Content.Rating          Genres
##               0               0               0               0               0
##    Last.Updated     Current.Ver     Android.Ver     Multi_Genre
##               0               0               0               0
```

```
plot_missing(googleplay_clean)
```




Las variables *Size* y *Rating* tienen valores perdidos. Puesto que la eliminicaión del las filas donde hay valores nulos supondría la pérdida

de un 15,64% de los datos que disponemos, no podemos considerarlo como una opción. En su lugar, imputaremos los datos perdidos utilizando la librería *mice*, concretamente el *predictive mean matching*:

```
summary(tmpData)
```

```
## Class: mids
## Number of multiple imputations:  5
## Imputation methods:
##            App       Category         Rating        Reviews           Size
##             ""             ""          "pmm"             ""          "pmm"
##       Installs           Type          Price Content.Rating         Genres
##             ""             ""             ""             ""             ""
##   Last.Updated    Current.Ver    Android.Ver    Multi_Genre
##             ""             ""             ""             ""
## PredictorMatrix:
##          App Category Rating Reviews Size Installs Type Price Content.Rating
## App        0        0      1       1    1        0    0     1              0
## Category   0        0      1       1    1        0    0     1              0
## Rating     0        0      0       1    1        0    0     1              0
## Reviews    0        0      1       0    1        0    0     1              0
## Size       0        0      1       1    0        0    0     1              0
## Installs   0        0      1       1    1        0    0     1              0
##          Genres Last.Updated Current.Ver Android.Ver Multi_Genre
## App           0            0           0           0           0
## Category      0            0           0           0           0
## Rating        0            0           0           0           0
## Reviews       0            0           0           0           0
## Size          0            0           0           0           0
## Installs      0            0           0           0           0
## Number of logged events:  10
##   it im dep     meth           out
## 1  0  0     constant           App
## 2  0  0     constant      Category
## 3  0  0     constant      Installs
## 4  0  0     constant          Type
## 5  0  0     constant Content.Rating
## 6  0  0     constant        Genres
```

```
data_clean <- complete(tmpData,1)
```

Verificamos que no hay ningún valor perdido:

```
colSums(is.na(data_clean))
```

```
##            App       Category         Rating        Reviews           Size
##              0              0              0              0              0
##       Installs           Type          Price Content.Rating         Genres
##              0              0              0              0              0
##   Last.Updated    Current.Ver    Android.Ver    Multi_Genre
##              0              0              0              0
```

# 3.3 Discretización de variable categóricas

Antes de discretizar en categorías las variables *chr*, primero analizamos el número de valores diferentes que tiene cada variable:

```
# ¿Con qué variables tendría sentido un proceso de discretización?
apply(data_clean,2, function(x) length(unique(x)))
```

```
##             App        Category          Rating         Reviews            Size 
##            9644              33              39            5334             459 
##        Installs            Type           Price Content.Rating          Genres 
##              19               2              88               6             118 
##    Last.Updated    Current.Ver    Android.Ver    Multi_Genre 
##            1376            2819              34               2
```

A partir de estos números, determinamos que será efectivo discretizar las variables *Category*, *Installs*, *Type*, *Content.rating*, *Android.Ver* y *Multi_Genre*:

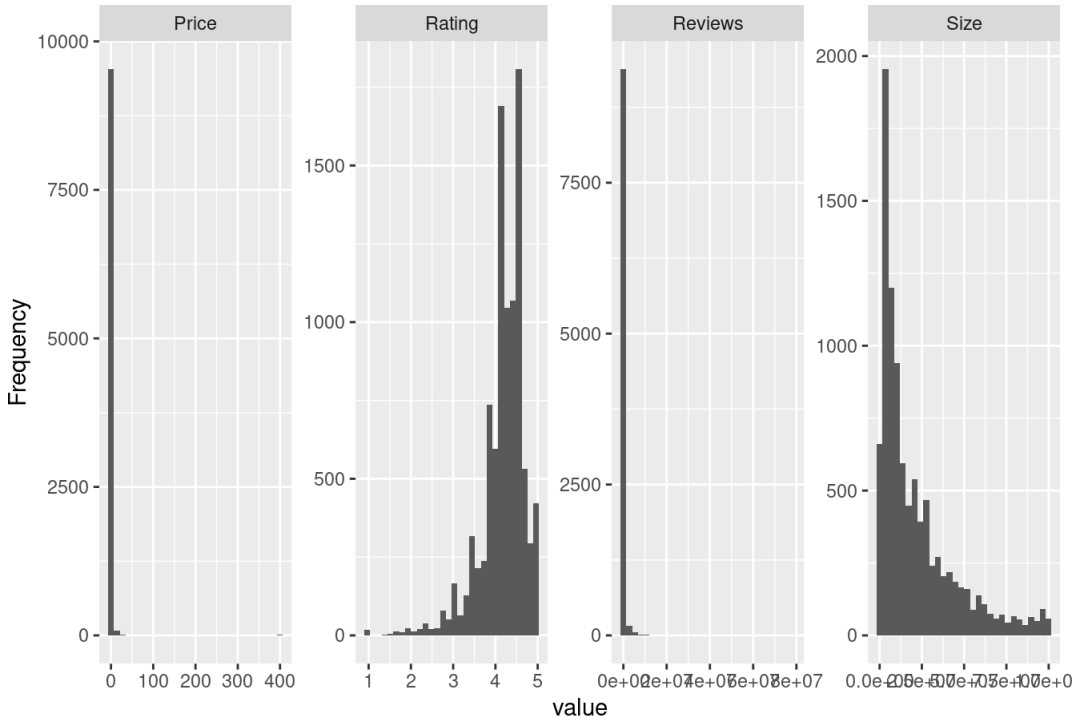
```
# Discretizamos las variables con pocas clases
cols<-c("Category","Installs","Type","Content.Rating", "Android.Ver", "Multi_Genre")
for (i in cols){
  data_clean[,i] <- as.factor(data_clean[,i])
}

# Después de los cambios, analizamos la nueva estructura del conjunto de datos
str(data_clean)
```

```
## 'data.frame':    9644 obs. of  14 variables:
##  $ App           : chr  "- Free Comics - Comic Apps" "¡Ay Caramba!" "¡Ay Metro!" "¿Es Vegan?" ...
##  $ Category      : Factor w/ 33 levels "ART_AND_DESIGN",..: 6 12 15 14 30 29 12 29 7 30 ...
##  $ Rating        : num  3.5 3.9 3.8 4.6 4.5 4.7 3.6 3.2 3.6 3.9 ...
##  $ Reviews       : num  115 0 489 438 259 ...
##  $ Size          : num  9100000 549000 36000000 1700000 203000 3600000 21000000 5600000 2500000
72000 ...
##  $ Installs      : Factor w/ 19 levels "1,000,000,000+",..: 6 4 6 6 6 19 2 10 10 18 ...
##  $ Type          : Factor w/ 2 levels "Free","Paid": 1 2 1 1 1 1 1 1 1 1 ...
##  $ Price         : num  0 1.99 0 0 0 0 0 0 0 0 ...
##  $ Content.Rating: Factor w/ 6 levels "Adults only 18+",..: 4 2 3 2 2 2 4 2 5 2 ...
##  $ Genres        : chr  "Comics" "Education" "Arcade" "Food & Drink" ...
##  $ Last.Updated  : chr  "2018-07-14" "2014-06-14" "2015-03-18" "2017-08-02" ...
##  $ Current.Ver   : chr  "5.0.12" "1.2" "1.0.3.1" "2.2.3" ...
##  $ Android.Ver   : Factor w/ 34 levels "1.0 and up","1.5 and up",..: 26 11 14 11 2 18 26 20 16 19
...
##  $ Multi_Genre   : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
```

## 3.4 Análisis de outliers

Las distribuciones de las variables numéricas de nuestro dataset no son gaussianas, tal y como podemos ver mediante los histogramas:

```
plot_histogram(data_clean)
```




Las distribuciones son *positive skewed* menos *Rating*, que es *negative skewed*.

Es por ello que para determinar los outliers de estas variables, utilizaremos la generalización de diagramas de cajas estandard (Vandervieren, E., Hubert, M. (2004) "An adjusted boxplot for skewed distributions"). Esta generalización tiene en cuenta la posible asimetría de los datos, que es lo que ocurre en nuestro caso.

Este diagrama de cajas generalizado se encuentra implementado en R en la librería *robustbase*. En primer lugar, contaremos el número de *outliers* que detecta en cada variable numérica:

```
library(robustbase)
str(adjboxStats(data_clean$Rating)$out)
```

```
##  num [1:611] 5 5 5 2.3 5 1.7 1.7 4.9 5 1.6 ...
```

```
str(adjboxStats(data_clean$Reviews)$out)
```

```
##  num [1:423] 1305050 16771865 1076243 1008012 14201891 ...
```

```
str(adjboxStats(data_clean$Price)$out)
```

```
##  num [1:744] 1.99 0.99 1.49 399.99 1.49 ...
```

```
str(adjboxStats(data_clean$Size)$out)
```

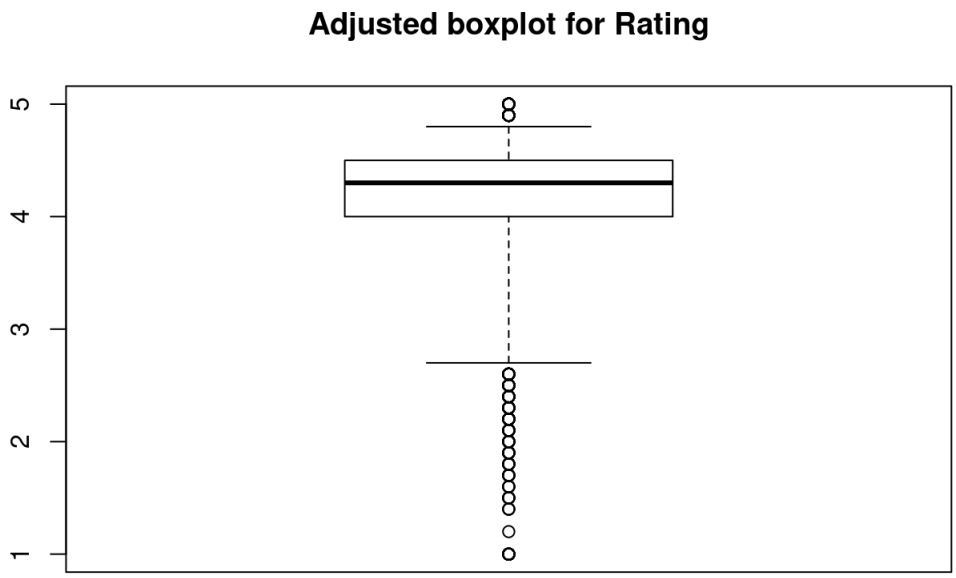
```
##  num(0)
```

Observamos que la variable *Size* no tiene ningún *outlier*, y en contraste, se detectan 611, 423 y 744 *outliers* en las variables *Rating*, *Reviews* y *Price* respectivamente.

A continuación, estudiaremos los diagramas de cajas de estas 3 variables.

## 3.4.1 Diagrama de cajas de Rating

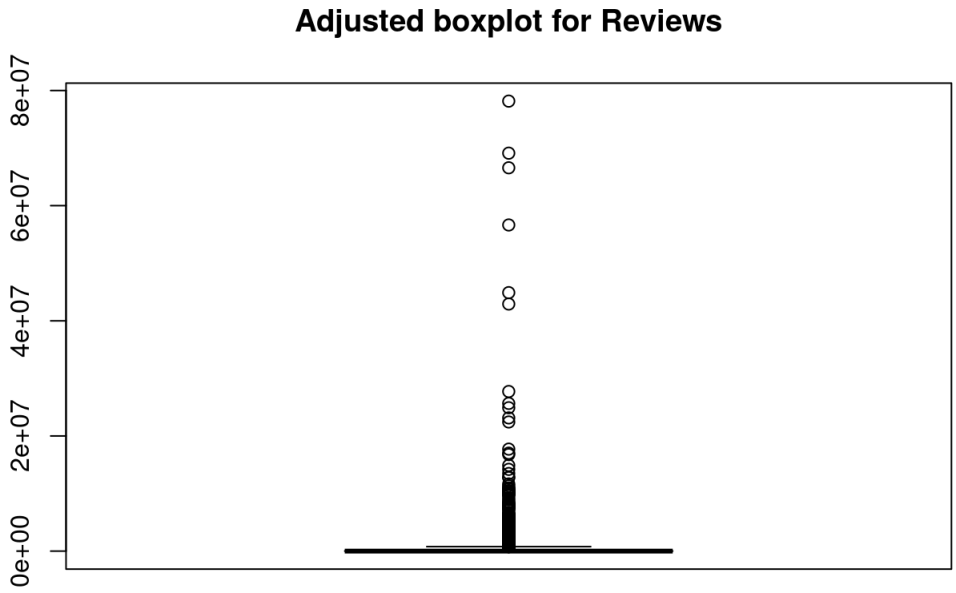
```
adjbox(data_clean$Rating, main = "Adjusted boxplot for Rating")
```




El diagrama de cajas muestra que todos los valores se encuentran entre 1 y 5, que es el rango posible de valores para *Rating*, por lo que no los consideraremos como *outliers*.

### 3.4.2 Diagrama de cajas de Reviews

```
adjbox(data_clean$Reviews, main = "Adjusted boxplot for Reviews")
```



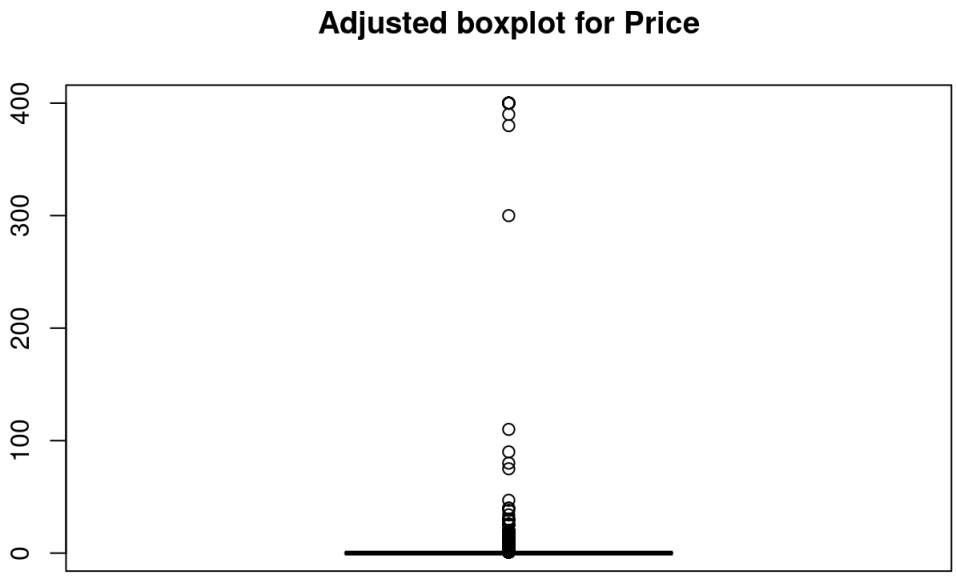

```
head(data_clean[data_clean$Reviews > 10000000,])
```

```
##                                                 App      Category Rating
## 76      360 Security - Free Antivirus, Booster, Cleaner         TOOLS    4.6
## 122                                       8 Ball Pool          GAME    4.5
## 1121                        BBM - Free Calls & Messages COMMUNICATION    4.3
## 1956 Cache Cleaner-DU Speed Booster (booster & cleaner)         TOOLS    4.5
## 2036                                    Candy Crush Saga          GAME    4.4
## 2447                                      Clash of Clans          GAME    4.6
##       Reviews    Size     Installs Type Price Content.Rating        Genres
## 76   16771865 5.2e+07 100,000,000+ Free     0       Everyone         Tools
## 122  14201891 5.2e+07 100,000,000+ Free     0       Everyone        Sports
## 1121 12843436 1.4e+07 100,000,000+ Free     0       Everyone Communication
## 1956 12759815 1.5e+07 100,000,000+ Free     0       Everyone         Tools
## 2036 22430188 7.4e+07 500,000,000+ Free     0       Everyone        Casual
## 2447 44893888 9.8e+07 100,000,000+ Free     0   Everyone 10+      Strategy
##      Last.Updated        Current.Ver        Android.Ver Multi_Genre
## 76     2018-08-05 Varies with device Varies with device          no
## 122    2018-08-01              4.0.0       4.0.3 and up          no
## 1121   2018-08-03 Varies with device       4.0.3 and up          no
## 1956   2018-07-26              3.1.2         4.0 and up          no
## 2036   2018-07-06          1.129.0.2         4.1 and up          no
## 2447   2018-07-16          10.322.16         4.1 and up          no
```

Tras analizar los valores de reviews que han sido detectados como *outliers*, podemos concluir que corresponden a aplicaciones que como mínimo tienen 10 veces más instalaciones que *reviews*. En definitiva, los valores son correctos y no los descartaremos de nuestro dataset.

### 3.4.3 Diagrama de cajas de Price

```
adjbox(data_clean$Price, main = "Adjusted boxplot for Price")
```




```
head(data_clean[data_clean$Price > 200,])
```

```
##                                App  Category Rating Reviews    Size Installs
## 22                          □ I'm rich LIFESTYLE    3.8     718 2.6e+07  10,000+
## 5465            I am extremely Rich LIFESTYLE    2.9      41 2.9e+06   1,000+
## 5470                     I am rich LIFESTYLE    3.8    3547 1.8e+06 100,000+
## 5471                     I am Rich   FINANCE    4.3     180 3.8e+06   5,000+
## 5472                     I Am Rich    FAMILY    3.6     217 4.9e+06  10,000+
## 5474 I am rich (Most expensive app)   FINANCE    4.1     129 2.7e+06   1,000+
##      Type  Price Content.Rating        Genres Last.Updated Current.Ver
## 22   Paid 399.99       Everyone     Lifestyle   2018-03-12       1.0.0
## 5465 Paid 379.99       Everyone     Lifestyle   2018-07-02         1.0
## 5470 Paid 399.99       Everyone     Lifestyle   2018-01-13         2.0
## 5471 Paid 399.99       Everyone       Finance   2018-03-23         1.0
## 5472 Paid 389.99       Everyone Entertainment   2018-06-23         1.5
## 5474 Paid 399.99           Teen       Finance   2017-12-07           2
##       Android.Ver Multi_Genre
## 22     4.4 and up          no
## 5465   4.0 and up          no
## 5470 4.0.3 and up          no
## 5471   4.2 and up          no
## 5472   4.2 and up          no
## 5474 4.0.3 and up          no
```

Tras analizar los valores de precios más extremos, podemos concluir que los precios son deliberadamente caros y no se trata de ningún error, por lo que incluiremos estas aplicaciones en el dataset.

# 4 Análisis

## 4.1 Selección de los grupos de datos que se quieren analizar/comparar

Con el objetivo de estudiar el comportamiento de la variable *Rating*, se han seleccionado los siguientes grupos:

- **Category**: se estudiará la variación de las distribuciones de *Rating* en las diferentes categorías con la intención de verificar si el rating varía significativamente entre ellas.
- **Type**: se estudiará si el hecho de que una aplicación sea de pago afecta significativamente al *rating*. ¿Se espera mayor calidad en una aplicación de pago? . **Multi_Genre**: se estudiará si el hecho de pertenecer a más de un género o no influye en el *rating* de la aplicación.

## 4.2 Comprobación de la normalidad y homogeneidad de la varianza

### 4.2.1 Comprobación de normalidad

A continuación, aplicaremos el test de Saphiro-Wilk para estudiar la normalidad de la variable *Rating*:

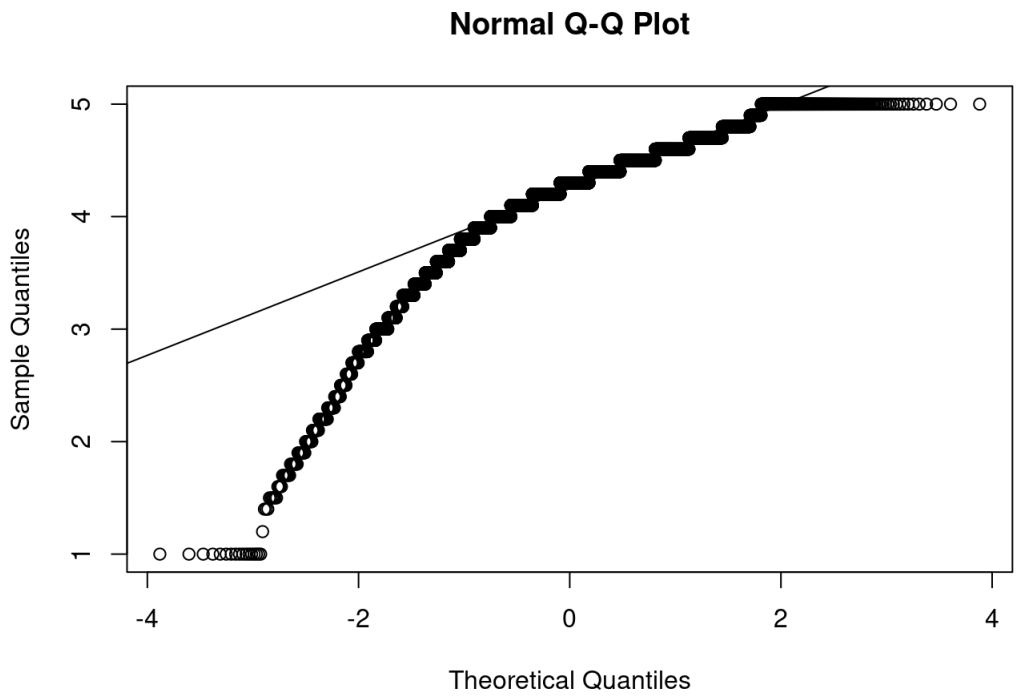
```
#Realizamos una muestra puesto que el algoritmo no acepta más de 5000 datos
shapiro.test(sample(data_clean$Rating, size = 4000))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  sample(data_clean$Rating, size = 4000)
## W = 0.86254, p-value < 2.2e-16
```

El valor de p que obtenemos es muy bajo, por lo que la hipótesis nula queda rechazada, es decir, la variable *Rating* no sigue una distribución normal. Como hemos podido ver anteriormente en los histogramas, la variable *Rating* es asimétrica centrada en 4.2 aproximadamente.

La normalidad también se puede comprobar visualmente empleando el gráfico de probabilidad normal. Este gráfico permite comparar una muestra de datos con la distribución normal.

```
qqnorm(data_clean$Rating)
qqline(data_clean$Rating)
```




Como se puede observar, los puntos no siguen una línea recta lo cual indica que las calificaciones, como se ha demostrado previamente con el test Shapiro-Wilk, no siguen una distribución normal,

## 4.2.2 Comprobación de la homocedasticidad

Puesto que nuestra variable objetivo no sigue una distribución normal, utilizaremos el test Fligner-Killeen para comprobar si las varianzas son homogenias entre las diferentes categorías de juegos, entre el tipo de aplicación (gratuita o de pago) y entre la variable *Multi_Genre*:

```
#Category
fligner.test(Rating ~ Category, data = data_clean)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  Rating by Category
## Fligner-Killeen:med chi-squared = 241.03, df = 32, p-value < 2.2e-16
```

```
#Type
fligner.test(Rating ~ Type, data = data_clean)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  Rating by Type
## Fligner-Killeen:med chi-squared = 0.2704, df = 1, p-value = 0.6031
```

```
#Multi-genre
fligner.test(Rating ~ Multi_Genre, data = data_clean)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  Rating by Multi_Genre
## Fligner-Killeen:med chi-squared = 44.255, df = 1, p-value = 2.882e-11
```

En el primer caso, se rechaza la hipótesis nula y se concluye que las varianzas no son homogénias en todas las categorías de juegos. Por otro lado, en el caso de *Type*, el valor de p es mayor que 0.05, por lo que podemos decir que las varianzas de *rating* son iguales tanto en

las aplicaciones de pago, como en la gratuitas.

Por último, también se rechaza la hipótesis nula en el caso de la variable *Multi_Genre*, es decir las varianzas son diferentes si el juego pertenece a más de una categoría o no.

# 4.3 Pruebas estadísticas

## 4.3.1 Comparación entre dos grupos

Puesto que la distribución de la variable *rating* no es normal, no podemos aplicar el test de t-student. En su lugar, se utilizará el test de Wilcoxon para determinar si las medias son iguales entre los dos grupos:

### 4.3.1.1 Comparación entre paid y free

```
wilcox.test(Rating ~ Type, data = data_clean )
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  Rating by Type
## W = 2861548, p-value = 6.484e-10
## alternative hypothesis: true location shift is not equal to 0
```

Tras aplicar el test, el valor de p obtenido es muy pequeño, por lo que se rechaza la hipótesis nula y se determina que hay diferencias significativas entre la media de la valoración en el caso de una aplicación de pago y la media de una gratuita.

### 4.3.1.2 Comparación entre multigénero y unigénero

```
wilcox.test(Rating ~ Multi_Genre, data = data_clean )
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  Rating by Multi_Genre
## W = 1683672, p-value = 0.01593
## alternative hypothesis: true location shift is not equal to 0
```

Tras aplicar el test, el valor de p obtenido es inferior a 0.05 (significancia del 5%), por lo que se rechaza la hipótesis nula y se determina que hay diferencias significativas entre la media de la valoración en el caso de una aplicación multigénero y la media de una aplicación unigénero.

## 4.3.2 Comparación entre categorías

El test de Kruskal-Wallis se trata de una alternativa no paramétrica del test ANOVA para trabajar con variables que no siguen una distribución normal. En el apartado 5 se visualizan las distribuciones de las calificaciones para cada una de las distintas categorías. Estas distribuciones son asimétricas (desplazadas hacia calificaciones altas) y por lo tanto no siguen una distribución normal de manera que el test ANOVA no podrá ser utilizado en este caso en particular.

Para realizar el test Kruskal se deben cumplir una serie de suposiciones. En primer lugar, las observaciones analizadas deben ser independientes y el número de grupos a a analizar debe ser mayor que 2 (en nuestro caso 33 grupos). En segundo lugar, la variable dependiente a evaluar debe ser continua (las calificaciones). Por último puntualizar que como se ha mencionado anteriormente la distribución de cada uno de los grupos no tiene porque ser normal.

Para aplicar el test en R se emplea la función kruskal.test() como se muestra a continuación.

```
kruskal.test(Rating ~ Category, data=data_clean)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  Rating by Category
## Kruskal-Wallis chi-squared = 206.8, df = 32, p-value < 2.2e-16
```

El p-valor obtenido es menor al nivel de significancia (0,05), de manera que podemos concluir que las calificaciones muestran diferencias

significativas para las distintas categorías.

### 4.3.3 Creación de un modelo de regresión: Random forest

En este apartado crearemos un modelo de regresión con el objetivo de intentar predecir las valoraciones de futuras aplicaciones. El modelo escogido es el Random forest. El random forest se basa en la construcción de varios árboles de decisión, escogiendo el promedio de los mismos para el modelo final. De esta forma se intenta evitar el sobreentrenamiento que suele ir asociado a los árboles de decisión, y así obtener mejores generalizaciones para futuros datos.

Los árboles de decisión, en nuestro caso de regresión, predicen la variable variable objetivo a partir de los valores de las demás variables introducidas en el modelo. Los árboles se van dividiendo en ramas y hojas, que van afinando más el rango del valor que queremos predecir. Contra más profundidad tenga un árbol, más precisa será la predicción, pero también se corre el riesgo de sobreentrenar el modelo.

Para la realización de nuestro modelo, se han excluído las variables: *App*, *Last.Updated*, *Genres* y *Current.Ver*, ya que aportaban un número excesivo de categorías.

```
library(randomForest)
library(caTools)
set.seed(71)
#Dividimos el dataset en 70% training y 30% test
data1= sample.split(data_clean,SplitRatio = 0.3)

#Obtenemos el train
train =subset(data_clean,data1==TRUE)

#obtenemos el test
test =subset(data_clean,data1==FALSE)

rf <- randomForest(Rating ~ Category + Reviews + Size + Installs + Type + Price + Content.Rating + A
ndroid.Ver + Multi_Genre, data = train)
print(rf)
```

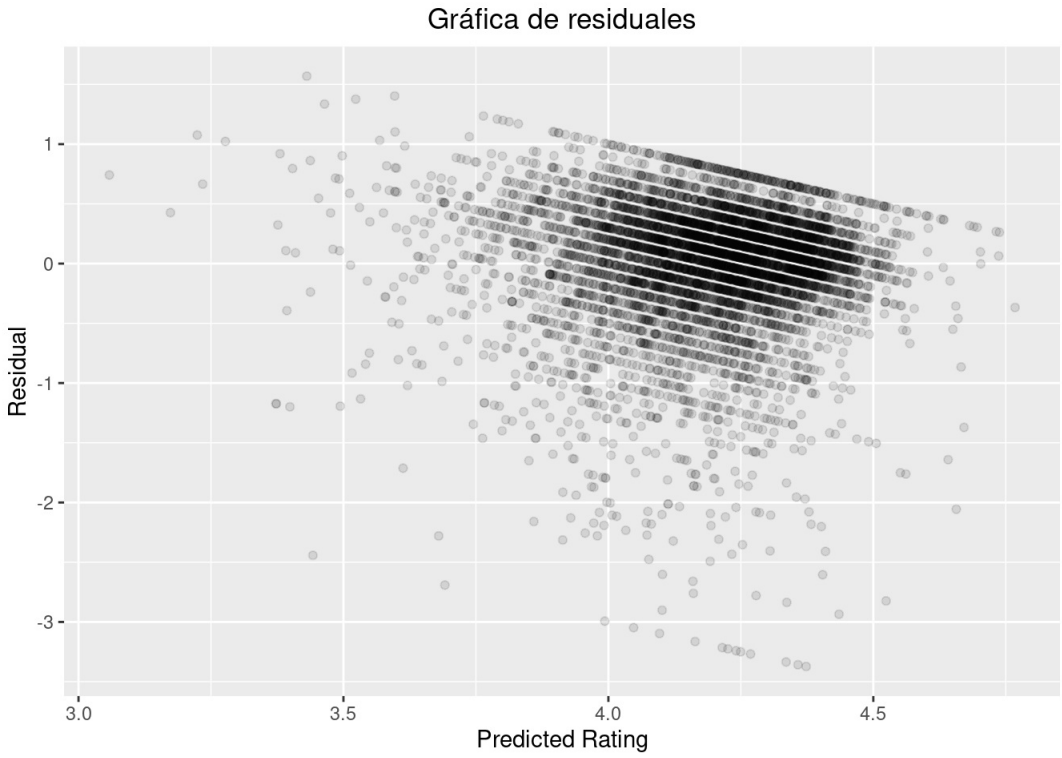
```
##
## Call:
##  randomForest(formula = Rating ~ Category + Reviews + Size + Installs +      Type + Price + Conte
nt.Rating + Android.Ver + Multi_Genre,      data = train)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 0.2475864
##                     % Var explained: 8.18
```

El modelo que hemos creado, tiene un error cuadrático medio de 0.25, lo cual no está diciendo que no se cometen muchos errores grandes, ya que estos tienen más peso cuando se calcula el error cuadrático medio.

Adicionalmente, el porcentaje de varianza nos indica qué tanto porciento de la varianza de *Rating* puede ser explicada por nuestro modelo. En nuestro caso, este valor es 8.18%, un valor bastante bajo. Esto quiere decir, que para poder hacer predicciones más acertadas, necesitaríamos la inclusión de nuevas variables de las que no disponemos.

```
pred <- predict(rf, newdata = test)
residuals <- test$Rating - pred
```

```
library(ggplot2)

fitting <- data.frame(pred, residuals)

ggplot(fitting, aes(x=pred, y=residuals))+
geom_point(alpha=0.1)+
  xlab("Predicted Rating") +
  ylab("Residual") +
  ggtitle("Gráfica de residuales")+
  theme(plot.title = element_text(hjust = 0.5),legend.position="right")
```




## 4.3.4 Regresión lineal: Installs vs Reviews

### 4.3.4.1 Conversión de installs

```
data_clean$Installs <- lapply(data_clean$Installs, function(x) gsub("\\+", "", x))
data_clean$Installs <- lapply(data_clean$Installs, function(x) gsub("\\,", "", x))
data_clean$Installs <- as.numeric(data_clean$Installs)

str(data_clean$Installs)
```

```
##  num [1:9644] 1e+04 1e+00 1e+04 1e+04 1e+04 5e+02 1e+06 1e+02 1e+02 5e+05 ...
```

Antes de calcular el modelo de regresión lineal realizamos una conversión logarítmica de ambas variables. Para evitar tener problemas al calcular el logaritmo en base 10 de 0 (que daría lugar a infinito), añadimos una pequeña constante en ambas variables, ya que tanto Reviews como Installs contienen valores igual a 0. Una vez realizada la conversión logarítmica de ambas variables podemos aplicar el modelo de regresión lineal, como se muestra a continuación.

```
C=1

modelo_lineal <-lm(log10(data_clean$Reviews+C)~log10(data_clean$Installs+C))

summary(modelo_lineal)
```

```
##
## Call:
## lm(formula = log10(data_clean$Reviews + C) ~ log10(data_clean$Installs +
##     C))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.9073 -0.3403  0.0019  0.3445  1.7233
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   -1.300697   0.013882   -93.7   <2e-16 ***
## log10(data_clean$Installs + C)  0.918176   0.002763   332.4   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5135 on 9642 degrees of freedom
## Multiple R-squared:  0.9197, Adjusted R-squared:  0.9197
## F-statistic: 1.105e+05 on 1 and 9642 DF,  p-value: < 2.2e-16
```

El coeficiente de determinacion (R-square) es una medida de calidad del modelo y toma valores entre 0 y 1, siendo su valor mayor cuanto mayor es el ajuste del modelo a los datos. En este caso se obtiene un coeficiente de determinación de 0.9197 lo que demuestra que existe una relación lineal marcada entre el logaritmo en base 10 del número de instalaciones y el logaritmo en base 10 del número de review (log10(Reviews) vs log10(Installs)), como observaremos posteriormente mediante un diagrama de dispersión en el apartado de visualizaciones. Además se puede considerar el modelo lineal estadísticamente significate, ya que el p-valor obtenido es menor al nivel de significancia (0,05).

Adicionalmente, se calcula el coeficiente de correlacion de Pearson, obteniendose un valor de 0.9590211. Lo que sugiere que existe una correlación lineal fuerte y positiva entre el logaritmo en base 10 de ambas variables (Installs y Reviews).

```
cor(log10(data_clean$Reviews+C),log10(data_clean$Installs+C))
```

```
## [1] 0.9590211
```

# 5 Representacion de los resultados a partir de graficas y tablas

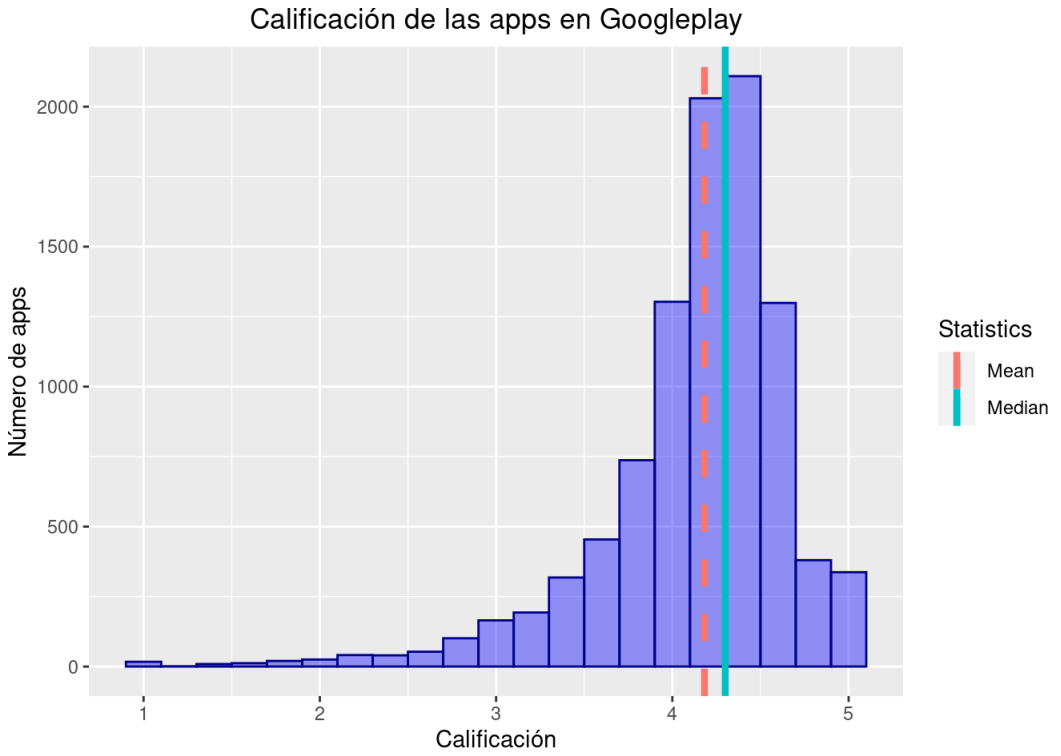
## 5.1 Instalar la libreria para la visualizacion

```
library(gridExtra)
library(GGally)
library(dplyr)
```

## 5.2 Distribucion de las puntuaciones

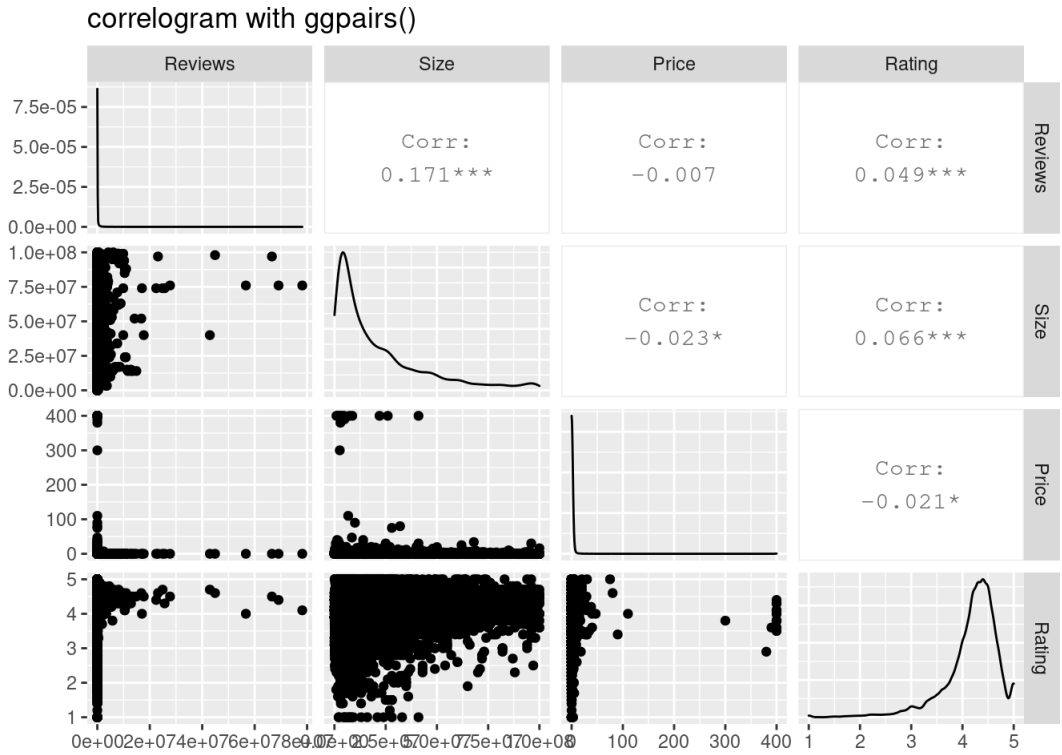
La calificaciones de las apps en google play siguen una distribución asímetrica desplazada hacia puntuaciones mayores como se observa en el siguente histograma, con una mediana(4.3) ligeramente superior a la media (4.183223).

```
ggplot(data_clean, aes(x=Rating))+
geom_histogram(binwidth= 0.2,color="darkblue", fill="blue", alpha=0.4)+
  geom_vline(aes(xintercept = median(data_clean$Rating),
                 color = "Median"),
             linetype = "solid",size=1.5) +
  geom_vline(aes(xintercept = mean(data_clean$Rating),
                 color = "Mean"),linetype=2,size=1.5)+
  xlab('Calificación')+
  ylab('Número de apps')+
  scale_color_hue(name = "Statistics")+
  ggtitle("Calificación de las apps en Googleplay")+
  theme(plot.title = element_text(hjust = 0.5))
```




## 5.3 Matriz de correlaciones

```
ggpairs(data_clean[, c(4, 5, 8, 3)], title="correlogram with ggpairs()")
```



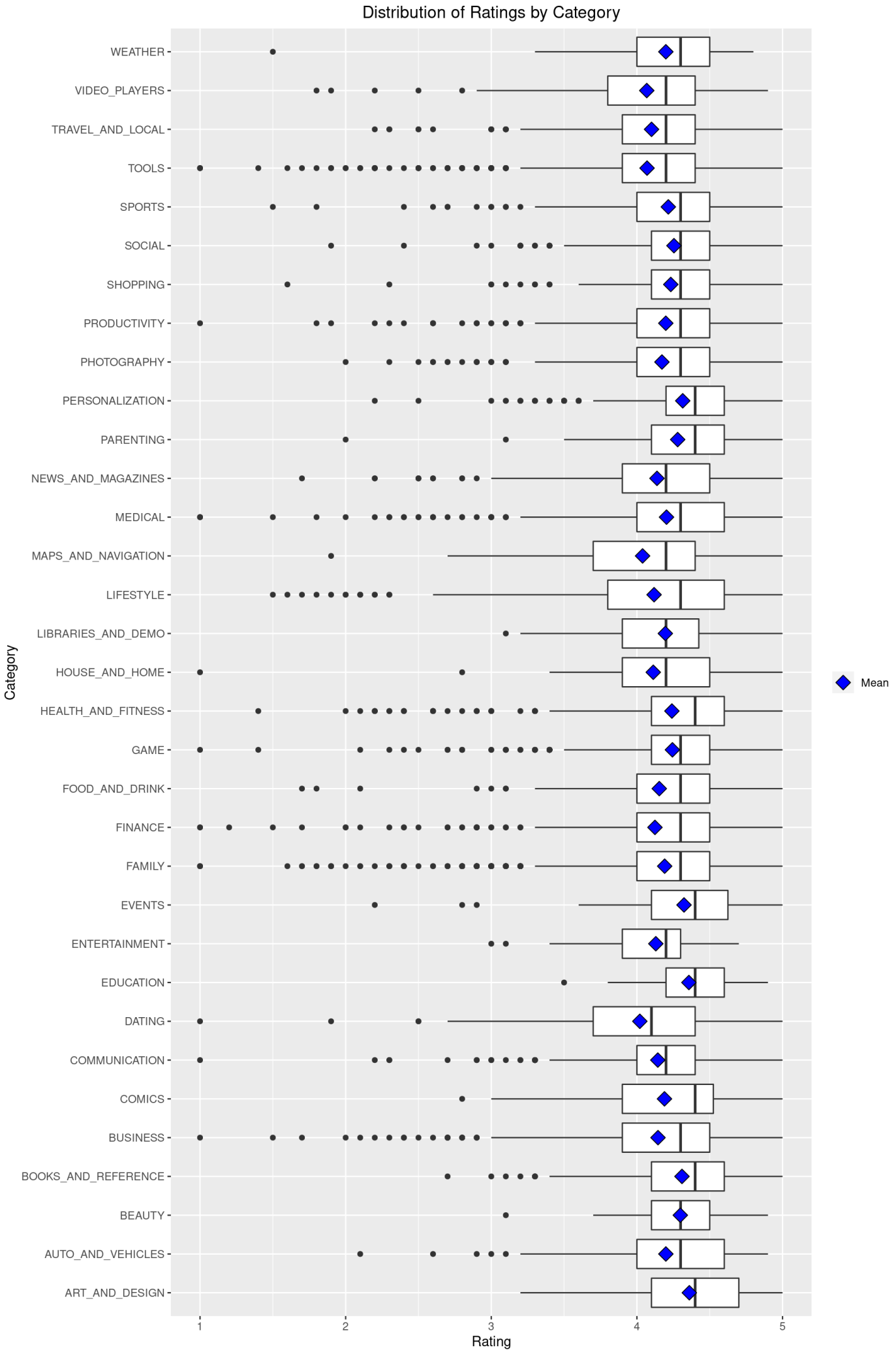

## 5.4 Matriz de correlaciones

```
ggcorr(data_clean, method = c("everything", "pearson"))
```


## 5.5 Diagramas de cajas de las categorias

Las calificaciones de las apps en google play siguen una distribución asimétrica (desplazada hacia calificaciones altas), de hecho la media de las calificaciones es superior a 4, siendo el rango posible de 0 a 5. En este apartado se muestra la distribución de las calificaciones para cada una de las categorías mediante diagramas de caja. Adicionalmente, se indica mediante un diamante la media para cada una de las categorías (33 en total). Como se muestra en la visualización todas las categorías presentan una distribución desplazada hacia puntuaciones elevadas. Además, todas ellas presentan una media en las calificaciones superior a 4 (como se demuestra posteriormente de manera analítica empleando la función aggregate).

```
ggplot(data_clean,aes(x=Rating,y=Category))+
  geom_boxplot() +
  geom_point(aes(fill='Mean'),stat='summary',
             fun.y=mean, shape = 23, size = 4) +
  scale_fill_manual('', values = c("Mean"='blue')) +
  xlab("Rating") +
  ylab("Category") +
  ggtitle("Distribution of Ratings by Category")+
  theme(plot.title = element_text(hjust = 0.5),legend.position="right")
```


Distribution of Ratings by Category
WEATHER
VIDEO_PLAYERS
TRAVEL_AND_LOCAL
TOOLS
SPORTS
SOCIAL
SHOPPING
PRODUCTIVITY
PHOTOGRAPHY
PERSONALIZATION
PARENTING
NEWS_AND_MAGAZINES
MEDICAL
MAPS_AND_NAVIGATION
LIFESTYLE
LIBRARIES_AND_DEMO
HOUSE_AND_HOME
HEALTH_AND_FITNESS
GAME
FOOD_AND_DRINK
FINANCE
FAMILY
EVENTS
ENTERTAINMENT
EDUCATION
DATING
COMMUNICATION
COMICS
BUSINESS
BOOKS_AND_REFERENCE
BEAUTY
AUTO_AND_VEHICLES
ART_AND_DESIGN
Category
1
2
3
4
5
Rating
Mean

El siguiente data frame contiene la media de las calificaciones para cada una de las categorías. Todas ellas presentan una media superior a 4, siendo dating (4.037) y arte y diseño (4.37) las categorias peor y mejor valoradas respectivamente.

```
means_category <- aggregate(Rating ~ Category, data_clean, mean)

means_category
```

```
##               Category   Rating
## 1       ART_AND_DESIGN 4.360000
## 2    AUTO_AND_VEHICLES 4.198824
## 3               BEAUTY 4.298113
## 4  BOOKS_AND_REFERENCE 4.309459
## 5             BUSINESS 4.144869
## 6               COMICS 4.189286
## 7        COMMUNICATION 4.143492
## 8               DATING 4.020000
## 9            EDUCATION 4.357009
## 10       ENTERTAINMENT 4.129885
## 11              EVENTS 4.323437
## 12              FAMILY 4.190593
## 13             FINANCE 4.123907
## 14      FOOD_AND_DRINK 4.153571
## 15                GAME 4.243869
## 16  HEALTH_AND_FITNESS 4.240278
## 17      HOUSE_AND_HOME 4.112329
## 18  LIBRARIES_AND_DEMO 4.195238
## 19           LIFESTYLE 4.117935
## 20 MAPS_AND_NAVIGATION 4.038931
## 21             MEDICAL 4.203299
## 22  NEWS_AND_MAGAZINES 4.137945
## 23           PARENTING 4.280000
## 24     PERSONALIZATION 4.314667
## 25         PHOTOGRAPHY 4.171886
## 26        PRODUCTIVITY 4.198391
## 27            SHOPPING 4.232673
## 28              SOCIAL 4.254430
## 29              SPORTS 4.215692
## 30               TOOLS 4.069723
## 31    TRAVEL_AND_LOCAL 4.100459
## 32       VIDEO_PLAYERS 4.067683
## 33             WEATHER 4.198734
```

```
print(min(means_category$Rating))
```

```
## [1] 4.02
```

```
means_category$Category[means_category$Rating==min(means_category$Rating)]
```

```
## [1] DATING
## 33 Levels: ART_AND_DESIGN AUTO_AND_VEHICLES BEAUTY ... WEATHER
```

```
print(max(means_category$Rating))
```

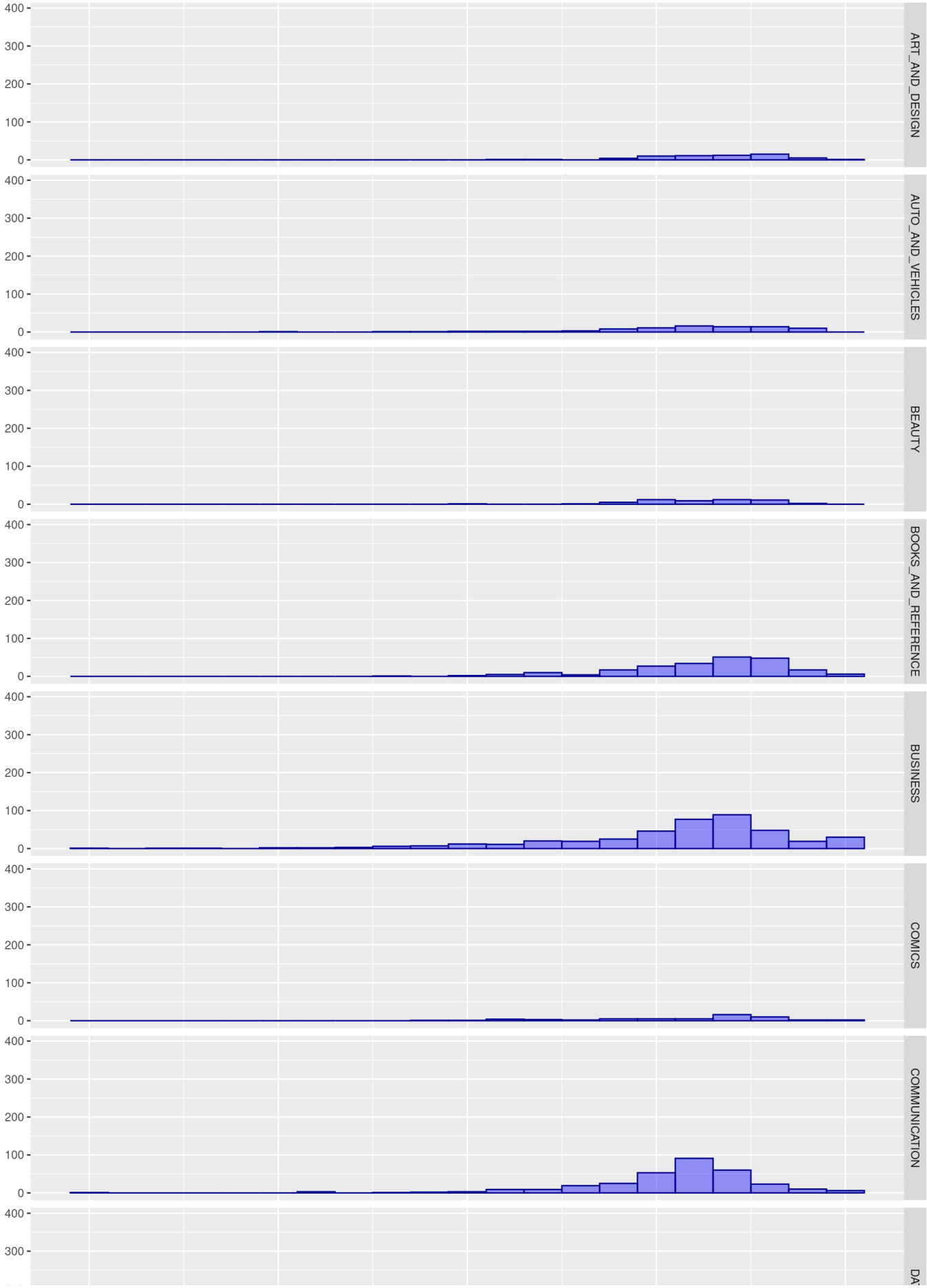
```
## [1] 4.36
```

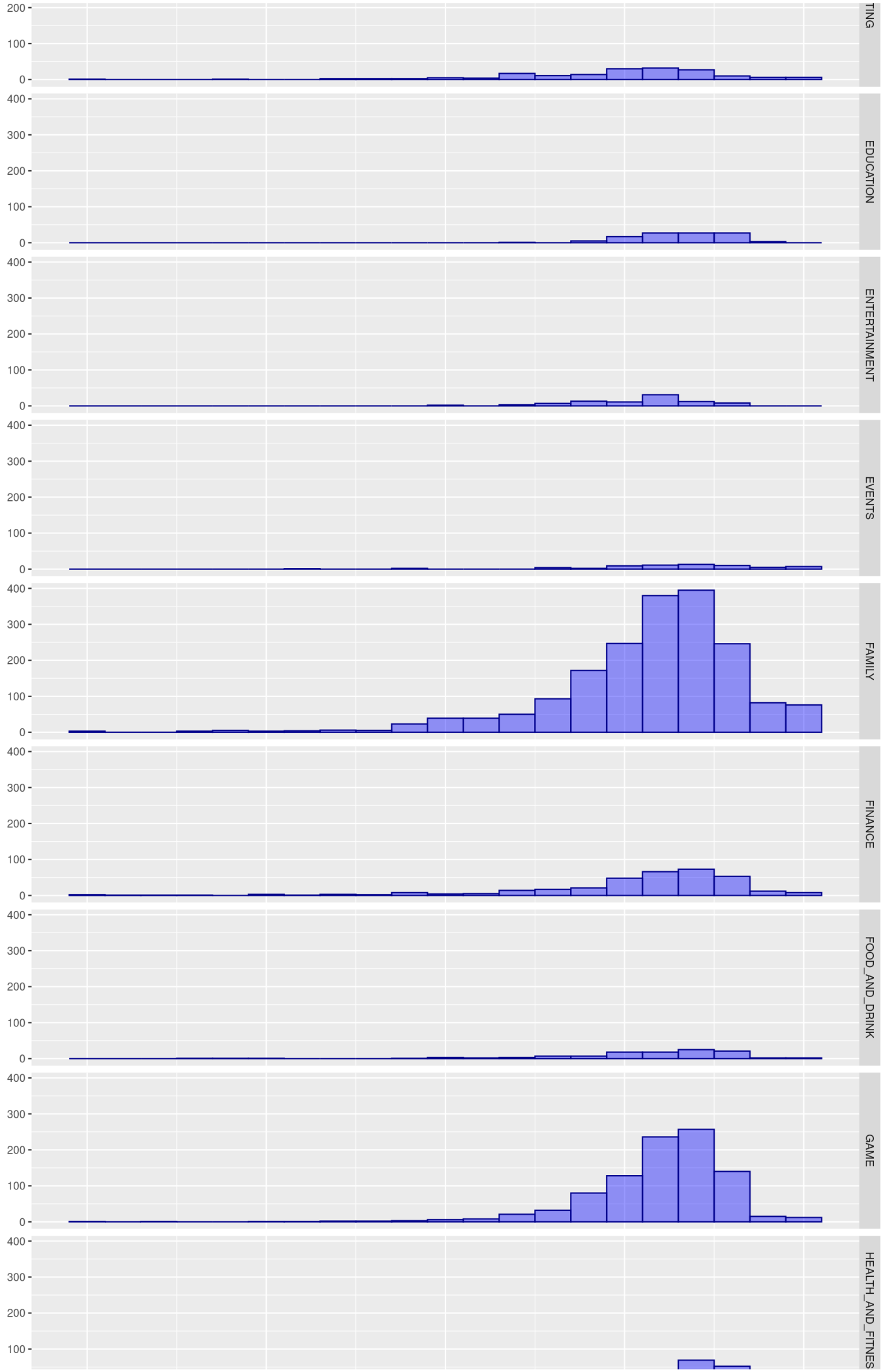
```
means_category$Category[means_category$Rating==max(means_category$Rating)]
```

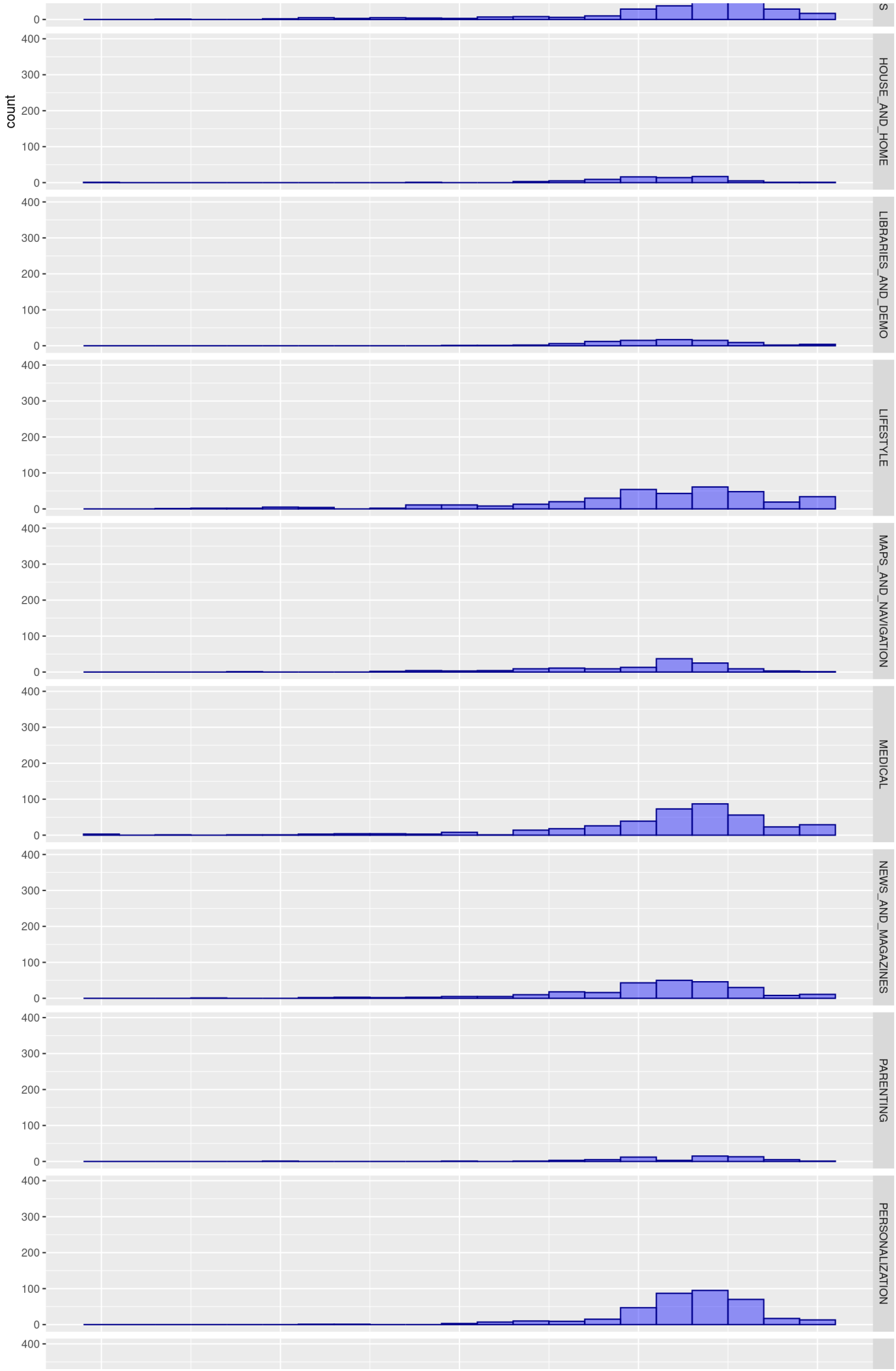
```
## [1] ART_AND_DESIGN
## 33 Levels: ART_AND_DESIGN AUTO_AND_VEHICLES BEAUTY ... WEATHER
```

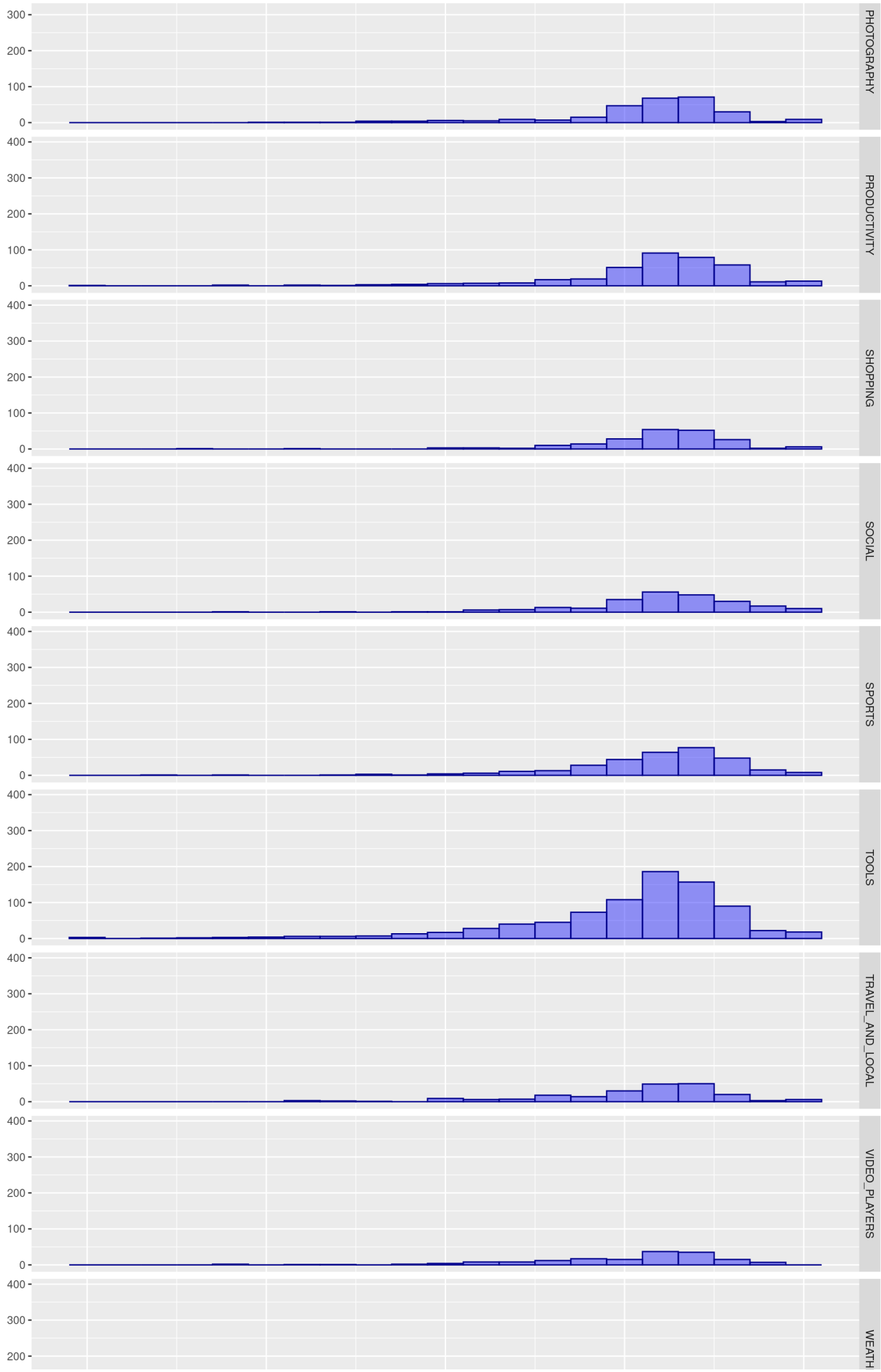
# 5.6 Distribucion de las calificaciones por categoria

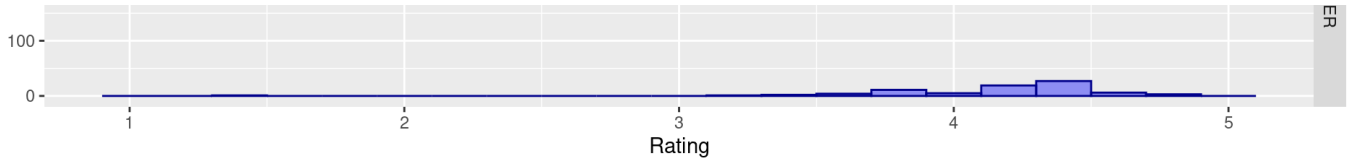
```
ggplot(data_clean, aes(x=Rating))+
geom_histogram(binwidth= 0.2,color="darkblue", fill="blue", alpha=0.4)+
  facet_grid(rows=vars(Category))
```




TING
EDUCATION
ENTERTAINMENT
EVENTS
FAMILY
FINANCE
FOOD_AND_DRINK
GAME
HEALTH_AND_FITNES
0
100
200
300
400


count
0
100
200
300
400
S
HOUSE_AND_HOME
LIBRARIES_AND_DEMO
LIFESTYLE
MAPS_AND_NAVIGATION
MEDICAL
NEWS_AND_MAGAZINES
PARENTING
PERSONALIZATION

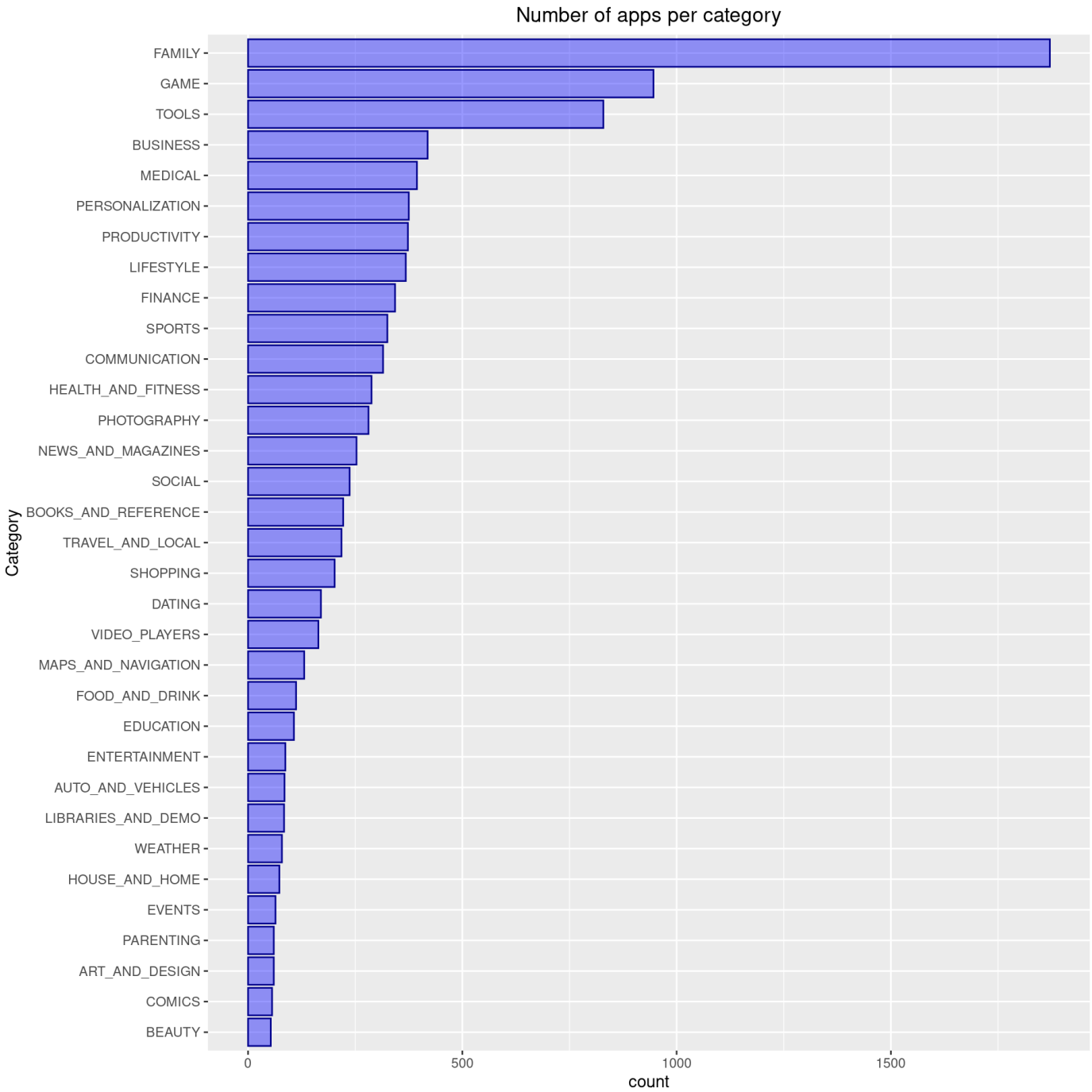

PHOTOGRAPHY
PRODUCTIVITY
SHOPPING
SOCIAL
SPORTS
TOOLS
TRAVEL_AND_LOCAL
VIDEO_PLAYERS
WEATH
0
100
200
300
400




## 5.7 Numero de apps por categoria

La siguiente gráfica muestra el número de apps disponibles en google play por categoría. Como se puede comprobar Family (19,4%), Game (9,8%) y Tools(8,6%) son las categorías predominantes en googleplay. Por el contrario, las apps dedicadas a belleza(0,05%), comics(0,06%) y art(0,06%) son las menos frecuentes.

```
data_clean <- within(data_clean,
Category <- factor(Category,
levels=names(sort(table(Category),
decreasing=FALSE))))

ggplot(data_clean, aes(y=Category))+
geom_bar(color="darkblue", fill="blue", alpha=0.4)+
ggtitle("Number of apps per category")+
theme(plot.title = element_text(hjust = 0.5))
```




La siguente tabla muestra las proporciones de cada una de las categorías

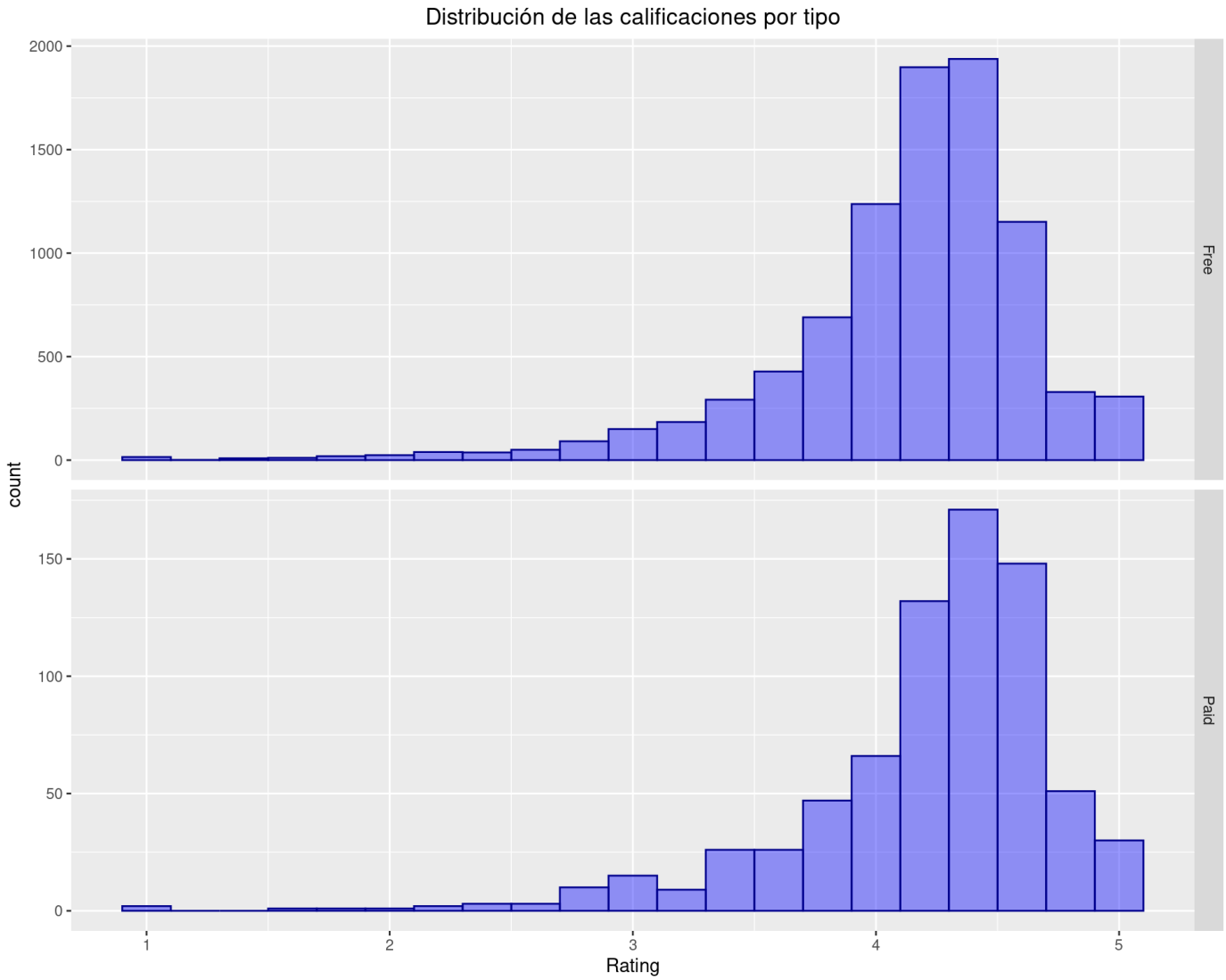
```
round(prop.table(table(data_clean$Category)),digits=3)
```

```
##
##              BEAUTY              COMICS      ART_AND_DESIGN           PARENTING
##               0.005               0.006               0.006               0.006
##              EVENTS      HOUSE_AND_HOME             WEATHER  LIBRARIES_AND_DEMO
##               0.007               0.008               0.008               0.009
##   AUTO_AND_VEHICLES       ENTERTAINMENT           EDUCATION      FOOD_AND_DRINK
##               0.009               0.009               0.011               0.012
## MAPS_AND_NAVIGATION       VIDEO_PLAYERS              DATING            SHOPPING
##               0.014               0.017               0.018               0.021
##    TRAVEL_AND_LOCAL BOOKS_AND_REFERENCE              SOCIAL  NEWS_AND_MAGAZINES
##               0.023               0.023               0.025               0.026
##         PHOTOGRAPHY  HEALTH_AND_FITNESS       COMMUNICATION              SPORTS
##               0.029               0.030               0.033               0.034
##             FINANCE           LIFESTYLE        PRODUCTIVITY     PERSONALIZATION
##               0.036               0.038               0.039               0.039
##             MEDICAL            BUSINESS               TOOLS                GAME
##               0.041               0.043               0.086               0.098
##              FAMILY
##               0.194
```

# 5.8 Distribucion de las calificaciones por tipo (free vs paid)

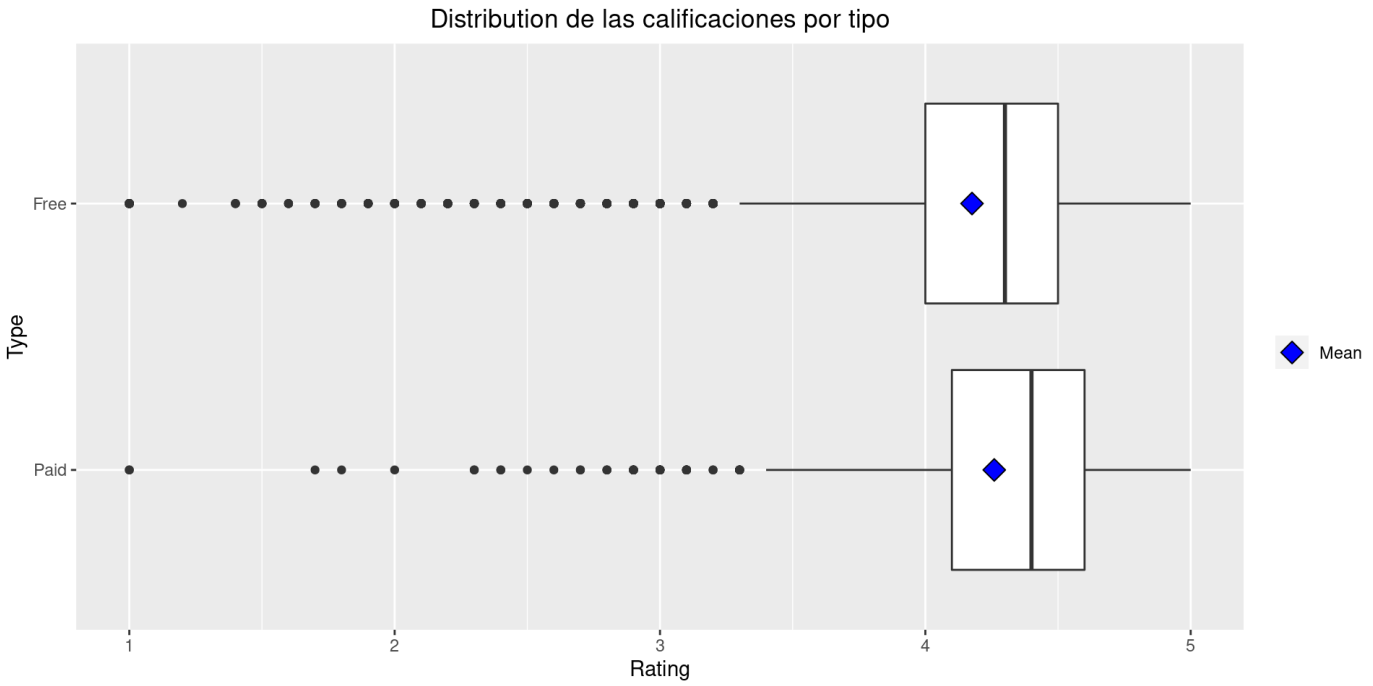
En este apartado se analizan gráficamente las calificaciones de los dos tipos principales de apps: (1) gratuitas y (2) de pago. En primer lugar, se visualiza la distribución de las calificaciones para cada uno de los grupos mediante un histograma. En la siguiente imagen, ambas gráficas comparten en mismo eje x para de este manera realizar más fácilmente la comparación entre las distribuciones, sin embargo, rango del eje y es distinto. Como ya hemos comentado anteriormente en la etapa de preprocesamiento, la mayoría de las aplicaciones disponibles en google apps son gratuitas, por este motivo, utilizar el mismo rango para el eje y en ambas gráficas dificultaría compararlas. Las dos distribuciones son asimétricas con la cola a la izquierda de la distribución de manera que en ambos casos la media es inferior a la mediana, sin embargo, se observa un ligero mayor sesgo a la izquierda en el caso de las aplicaciones de pago, como posteriormente se podrá corroborar con los diagramas de caja de ambas distribuciones.

```
ggplot(data_clean, aes(x=Rating))+
geom_histogram(binwidth= 0.2,color="darkblue", fill="blue", alpha=0.4)+
  facet_grid(rows=vars(Type),
  scales="free_y")+
  ggtitle("Distribución de las calificaciones por tipo")+
  theme(plot.title = element_text(hjust = 0.5))
```




La siguiente visualización muestra el diagrama de caja de ambos tipos de apps: (1) gratuitas y (2) de pago. El diagrama de caja de las puntuaciones muestra que las apps de pago presentan una distribución más desplazada hacia la derecha, presentando tanto una media como una mediana mayor que la distribución de la apps gratuitas. Por otra parte, esta gráfica también permite observar que la distribución de las calificaciones es asimétrica para los dos tipos de apps ya que la mediana es superior a la media en ambos casos.

```
data_clean$Type <- factor(data_clean$Type , levels=c("Paid", "Free"), ordered=TRUE)

ggplot(data_clean,aes(x=Rating,y=Type))+
  geom_boxplot() +
  geom_point(aes(fill='Mean'),stat='summary',
             fun.y=mean, shape = 23, size = 4) +
  scale_fill_manual('', values = c("Mean"='blue')) +
  xlab("Rating") +
  ylab("Type") +
  ggtitle("Distribution de las calificaciones por tipo")+
  theme(plot.title = element_text(hjust = 0.5),legend.position="right")
```




# 5.9 Distribucion de las calificaciones por presencia o no de Multigénero

En el apartado de integración y selección de los datos habíamos creado una variable categórica llamada Multi_Genre para indicar si la app en cuestión pertenece a un solo género (valor 0), o por el contrario, se puede asociar a múltiples géneros (valor 1).

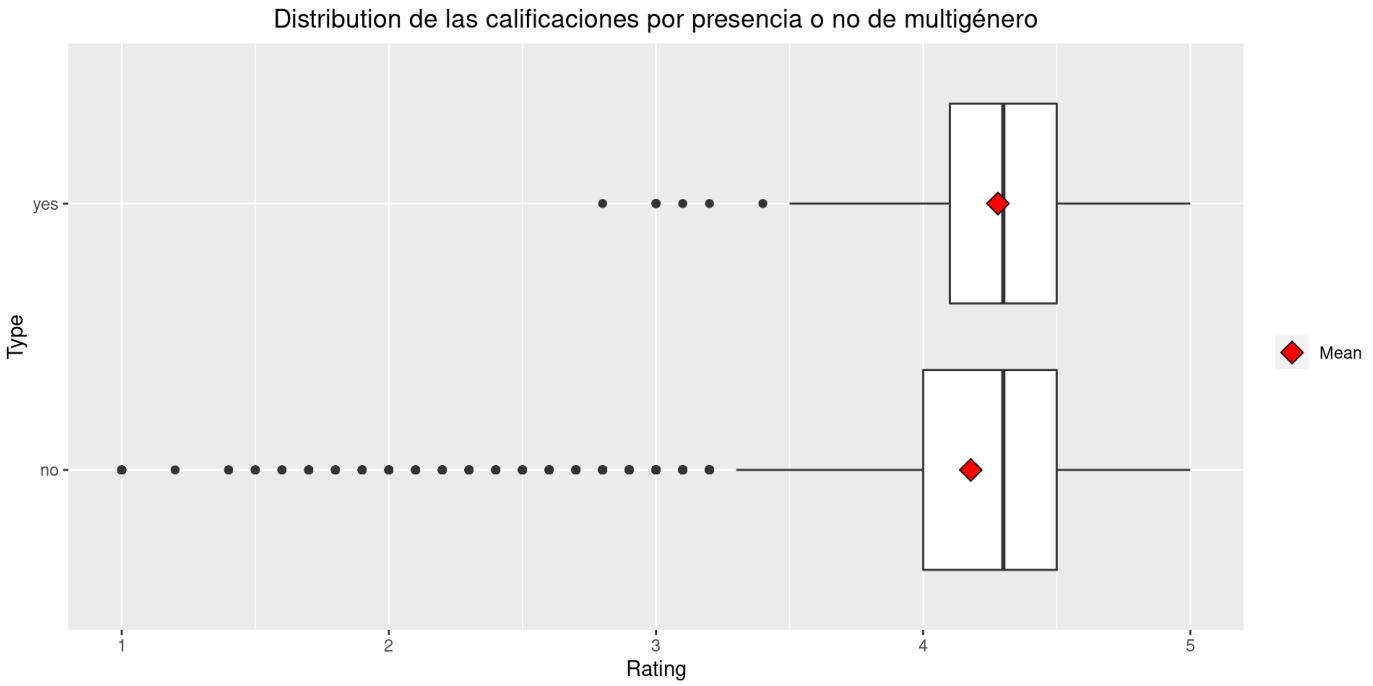
La función summary nos permite obtener un resumen estadístico de la variable de interés, en este caso Multi_Genre. Tras aplicar la función, se puede concluir que la gran mayoría de las apps pertenecen a un solo género (95.935%), como se muestra en la tabla inferior.

```
summary(data_clean$Multi_Genre)
```

```
##   no  yes
## 9252  392
```

En relación a la distribución de los datos, se observa una vez más que las calificaciones de ambos grupos se encuentran desplazadas hacia la derecha (puntuaciones elevadas), sin embargo, las aplicaciones multigénero presentan una media ligeramente superior que las aplicaciones de un solo genero. Además, se observa también que las aplicaciones multigénero presentan un menor número de outliers. Este hecho tiene cierta lógica ya que al estar disponible un menor número de apps multigénero en el mercado existe una probabilidad menor de que existan apps de muy baja calidad que generen el rechazo del público.

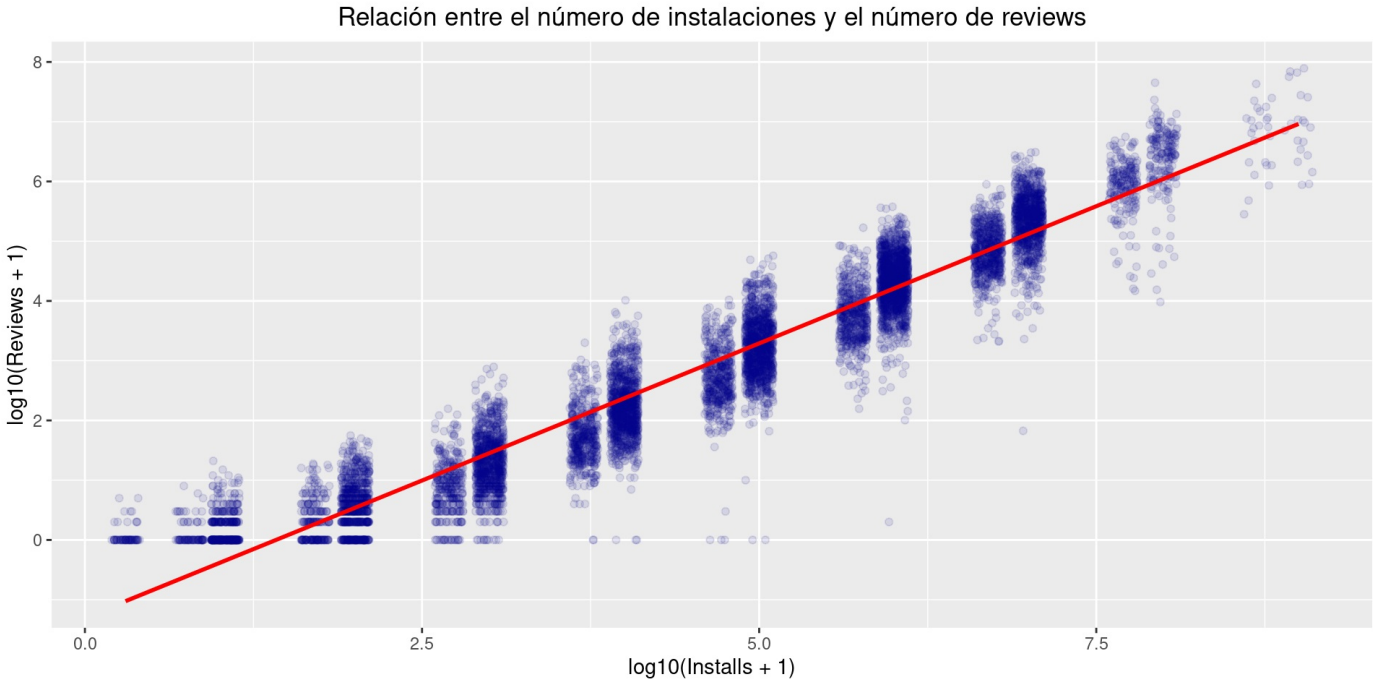
```
ggplot(data_clean,aes(x=Rating,y=Multi_Genre))+
  geom_boxplot() +
  geom_point(aes(fill='Mean'),stat='summary',
             fun.y=mean, shape = 23, size = 4) +
  scale_fill_manual('', values = c("Mean"='red')) +
  xlab("Rating") +
  ylab("Type") +
  ggtitle("Distribution de las calificaciones por presencia o no de multigénero")+
  theme(plot.title = element_text(hjust = 0.5),legend.position="right")
```




## 5.10 Diagrama de dispersión (Reviews vs Installs)

La siguente gráfica muestra el diagrama de dispersión del logaritmo en base 10 del número de reviews vs el logaritmo en base 10 del número de instalaciones. La variable instalaciones se proporciona en el conjunto de datos de manera discreta. Además, el número de visualizaciones observadas es relativamente alto (9644), por este motivo, se deben empleara las técnicas de jitter y transparencia para evitar el overplotting de los datos y visualizar de esta manera mejor los resultado.

```
ggplot(data_clean,aes(x=log10(Installs+1),y=log10(Reviews+1)))+
  geom_point(color="darkblue", fill="blue", alpha=0.1, position = "jitter")+
  geom_smooth(method = "lm", se = FALSE, col='red')+
  ggtitle("Relación entre el número de instalaciones y el número de reviews")+
  theme(plot.title = element_text(hjust = 0.5),legend.position="right")
```




Como hemos comentado anteriormente existe una relación lineal fuerte y positiva entre el logaritmo en base 10 de ambas variables, representada por la siguiente ecuación: log10(y)= -1.30069 + 0.918176 * log10(x). Esta línea de regresión se puede visualizar en rojo en el gráfico anterior.

# 6 Conclusiones

- La variable *Rating* no sigue una distribución normal, esto ha sido demostrado mediante el test de Saphiro-Wilk. La distribución que le corresponde a la variable *Rating* es una asimétrica negativa, es decir, la cola a la izquierda del pico es mayor que la cola del lado opuesto. Esta distribución presenta una media cercana a 4.2, lo que en general nos indica que las valoraciones de las aplicaciones son altas.
- Se ha determinado, mediante el test de Fligner-Killeen, que las variables *Category* y *Multi_genre* no tienen varianzas homogénias entre sus categorías. Por el contrario, la variable *Type*, muestra varianzas homogénias entre sus grupos ("Paid" y "Free").
- Se ha comprobado mediante el test de Wilcoxon que tanto la variable *Type* como la variable *Multi_Genre* presentan medias de rating diferentes en sus dos grupos ("Paid" y "Free"; "yes" y "no"). Se observan medias de *rating* más altas cuando la aplicación es de pago. Adicionalmente, se ha observado que las aplicaciones con multigénero tienen medias de *rating* más elevadas que cuando son unigénero.
- Las diferentes categorías de aplicaciones muestran, mediante el test Kruskal-Wallis, diferencias significativas en sus *mean ranks*. Las medias varían entre 4.37 (*ART_AND_DESIGN*) y 4.04 (*DATING*).
- Un modelo de random forest consigue explicar el 8% de la varianza de *Rating*, lo que se considera como mal modelo. Para una mayor precisión, se deberían incluir más variables en el conjunto de datos que se correlacionen de forma más directa con *Rating*.
- Se observa una relación lineal entre el número de *installs* y el número de *reviews* con un coeficiente de determinación de 0.83.