Proyecto Final: Amazon Reviews

Andrea Moreno, J. Alejandro Flores, Alejandro Grimaldi y Luis Alfredo Trejo

28 de mayo de 2021

```
library(ROSE)
library(glmnet)
library(gamlr)
library(knitr)
library(tidytext)
library(tm)
library(wordcloud)
library(dplyr)
library(ggplot2)
library(kableExtra)
library(readxl)
library(stringr)
library(tidyverse)
library(sentimentr)
library(topicmodels)
library(broom)
library(tictoc)
library(spacyr)
library(dplyr)
library(knitr)
library(textdata)
library(igraph)
library(ggraph)
library(RCT)
library(stargazer)
library(glmnet)
library(distrom)
library(UBL)
library(caret)
library(pROC)
library(taRifx)
```

```
datos <- read.table("C:/Users/agrimaldi/Dropbox/ITAM MEA/Cuarto Semestre/Economía Computacional/Proyect

#datos <- read.table("C:/Users/Luis.Trejo/Documents/Luis/Economía_Computacional/Proyecto/Review_subset.

#datos <- read.table("C:/Users/andrea.moreno/Desktop/ITAM 4S/Eco Computacional/FinalCompu/Review_subset

#datos <- read.table("C:/Users/jesus.flores/OneDrive - CFEnergia, S.A. de C.V/ITAM/Eco Computacional/pr

datos <- as.data.frame(datos)
```

Introducción

Utilizando los datos entregados de los comentarios y calificaciones a los productos vendidos por Amazon, presentamos el análisis de cuales son los productos que pueden predecir los resultados de las calificaciones a la plataforma, así como una análisis de sensibilidad de los temas que más se destacan en los comentarios a los productos.

A partir de un análisis de sentimientos de los comentarios, se les asignó un valor entre -5 y 5 (de negativo a positivo) a cada palabra para crear un score de sentimientos el cual se utilizó para predecir el score numérico original, en conjunto con dummies por palabra de los reviews. Para realizar dicha predicción, se dividió a la base en dos: una de entrenamiento (70%) donde se construyó el modelo y una de validación (30%) donde se probaron las predicciones.

Se concluyó que un modelo Distribute Multinomial Regression (DMR) es la mejor aproximación para estimar el Score numérico a partir del número de comentarios, el mes de la compra, la categoría y grupo del producto, así como dummies por palabra en el comentario y un score numérico de sentimientos utilizando el diccionadio *afinn*. El modelo presentó un área bajo la curva ROC del 71.3% con mayor sensibilidad que especificidad para las cinco clases del Score.

Al final, se concluye con una serie de sugerencias para el cliente a partir del análisis y las predicciones anteriores.

1. Limpieza de la base de datos

Los datos con los que se cuentan son los productos vendidos a través de la plataforma durante el 2012. Cada producto se identifica por un código, un código de usuario, calificación otorgada por el cliente al producto, tipo de producto, departamento, las variables (Nrev y Length) y comentario que le dio el cliente.

La base de datos contiene 9 columnas o variables: un identificador del producto, otro identificador por usuario, la calificación que le puso al producto, la fecha, el número de reviews (Nrev), una variable desconocida (Lenght), la categoría del producto, el grupo del producto y los comentarios del usuario al producto.

El primer paso para limpiar la base de datos, consiste en pasar a minúsculas el texto de los comentarios, quitar signos de puntuación y eliminar "stopwords":

```
### Pasar a minúsculas:
datos <- datos %>%
    unnest_tokens(output = Summary, input = Summary, token = "sentences")

### Quitar signos de puntuación:
datos$Summary <- removePunctuation(datos$Summary)

### Quitar Stopwords:
datos$Summary <- removeWords(datos$Summary, stopwords("english"))
```

Se identifica la existencia de NA en dos variables: categoría del producto ("Prod_Category") y grupo del producto ("Prod_Group"). Podríamos remover las filas con NA, pero perderíamos el 11.6% de las observaciones y, además, podría ser relevante para predecir el puntaje que sean productos sin categoría. Por ello, decidimos crear una categoría de grupos y productos que sea "NAs", lo cual es similar a tener una categoría tipo "otros":

```
### Contamos los NA por columna y sacamos el porcentaje por variable:

na_count <- sapply(datos, function(datos) (sum(length(which(is.na(datos))))/length(datos))*100)
```

Table 1: Porcentaje de variables con NA

	Porcentaje
ProductId	0.00000
UserId	0.00000
Score	0.00000
Time	0.00000
Summary	0.00000
Nrev	0.00000
Length	0.00000
Prod_Category	11.58764
Prod_Group	11.58764

```
### Transforma en data frame:

na_count <- data.frame(na_count)
#row.names(na_count) <- col_2
colnames(na_count)[1]<-c("Porcentaje")

### La tabla:

na_count <- na_count %>%
  kbl(caption = "Porcentaje de variables con NA") %>%
  kable_classic(full_width = F, html_font = "Cambria")
na_count
```

```
### Asignamos la categoría de NA:

datos$Prod_Category[is.na(datos$Prod_Category)] <- "NAs"
datos$Prod_Group[is.na(datos$Prod_Group)] <- "NAs"
```

Finalmente, tokenizamos las palabras de manera individual, lo cual nos servirá después para asignarles sentimientos.

```
### Tokenizamos las palabras:

tokens <- datos %>%
  unnest_tokens(output = word, input =Summary)

cleaned_tokens <- tokens %>%
  anti_join(get_stopwords())

nums <- cleaned_tokens %>%
  filter(str_detect(word, "^[0-9]")) %>%
  select(word) %>% unique()
nums
```

```
         word
1     12ounce
2        4abc
```

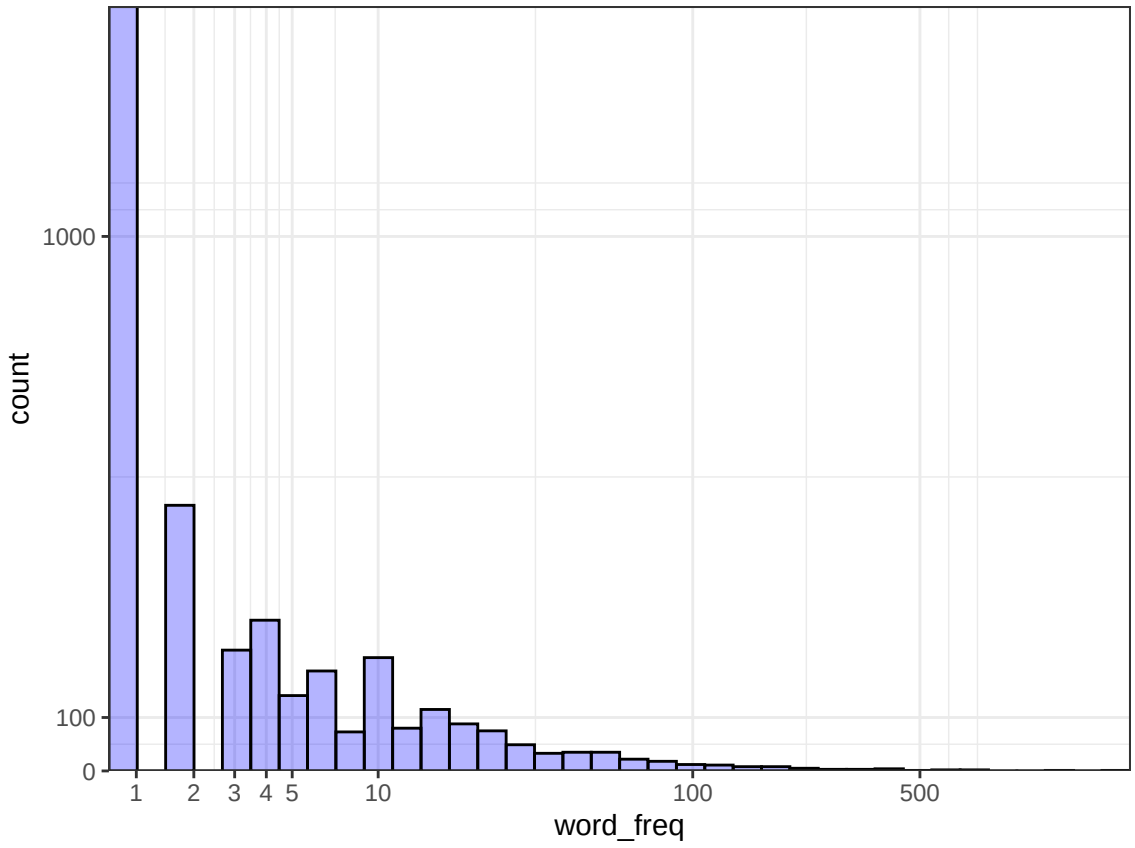
3	3ab
4	1st
5	2012
6	4
8	100
9	9
10	2
11	36
15	1
17	3
24	35
25	24
26	23
38	8th
39	7th
42	54
46	5
49	20
53	10
55	06202
56	12pack
57	125
58	0z
91	12
93	6
94	25
99	10th
126	155
127	30
129	19
135	95
136	2009
140	65
142	33ounce
145	16oz
146	17
178	1700
179	50
183	32
189	110
205	80
206	069cup
218	4ab
220	52
224	524
239	18
240	11
256	0calorie
271	21
272	34
292	60
294	17oz
298	70
300	3s

```
301        10x
302   162ounce
305        50s
310          0
314        810
315         26
316         13
319        338
339        2nd
348        737
351    16ounce
360         15
390 60servings
391    17ounce
400  95percent
402   132ounce
416       32oz
```

```
cleaned_tokens <- cleaned_tokens %>%
  anti_join(nums, by = "word")

length(unique(cleaned_tokens$word))
```

```
[1] 3659
```

```
cleaned_tokens$Prod_Category <- str_replace_all(cleaned_tokens$Prod_Category[], "[^[:alnum:]]", "")

cleaned_tokens$Prod_Group <- str_replace_all(cleaned_tokens$Prod_Group[], "[^[:alnum:]]", "")

cleaned_tokens %>%
  count(word, sort = T) %>%
  rename(word_freq = n) %>%
  ggplot(aes(x=word_freq)) +
  geom_histogram(aes(y=..count..), color="black", fill="blue", alpha=0.3, binwidth = 0.2) +
  scale_x_continuous(breaks=c(0:5,10,100,500,10e3), trans="log1p", expand=c(0,0)) +
  scale_y_continuous(breaks=c(0,100,1000,5e3,10e3,5e4,10e4,4e4), expand=c(0,0)) +
  theme_bw()
```



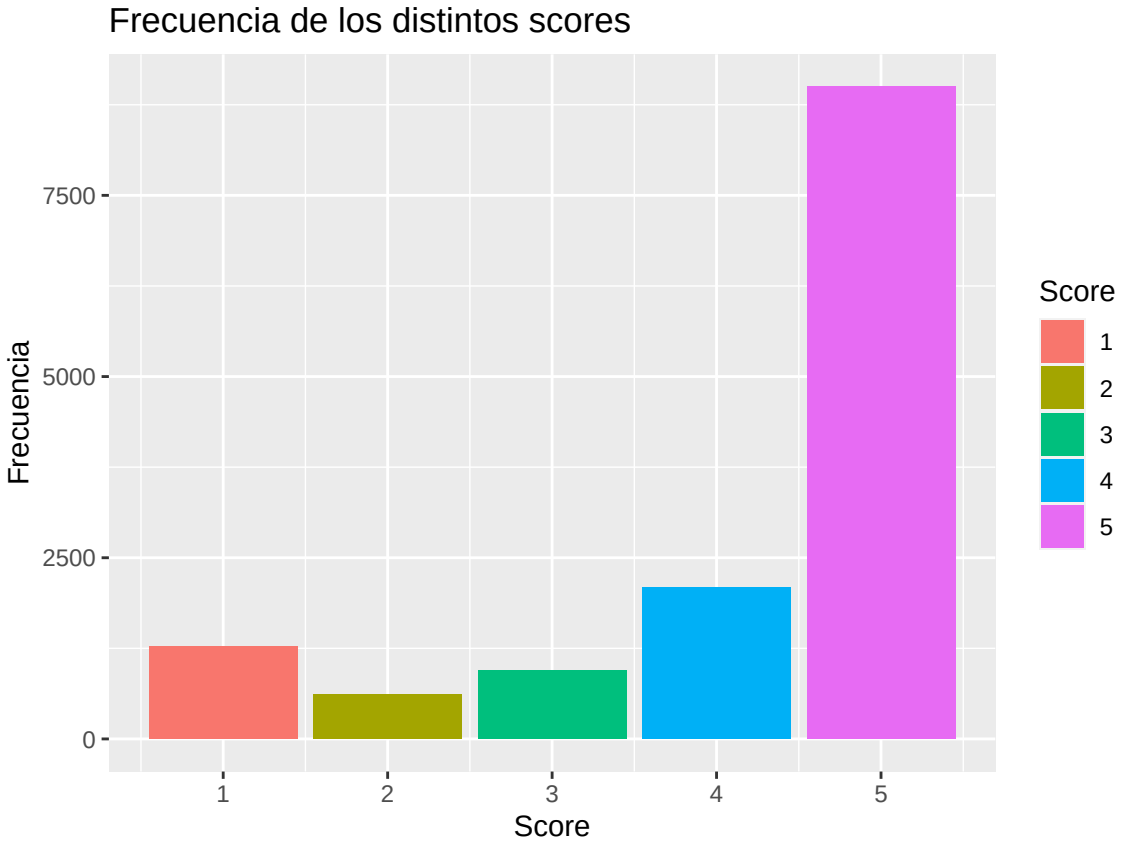
2. Descripción de los datos

Ahora presentamos un análisis gráfico de una de la variable de interés como es el score. Observamos que la gran mayoría de las calificaciones son positivas, casi 9 mil calificaciones de 5 estrellas, que es la calificación más alta que un cliente puede otorgar a un producto, lo que equivale al 64% del universo de los artículos vendidos. La calificación que menos se da a los productos es el 2, seguido por el 3 y la calificación promedio obtenida es de 4.2, de acuerdo con la tabla de estadísticas descriptivas.

```
### Observamos la frequencia de los scores:

ggplot(data = datos, mapping = aes(x = Score, fill = as.factor(Score))) +
  geom_bar() +
  labs(title="Frecuencia de los distintos scores", x = "Score", y = "Frecuencia", fill = "Score")
```

Table 2: Estadísticas descriptivas de variables numéricas

variable	mean	n	0	0.05	0.1	0.25	0.5	0.75	0.9	0.95	1
Score	4.214368	13920	1	1	2	4	5	5	5	5	5
Nrev	5.291164	13920	1	1	1	1	4	8	11	14	28
Length	3.846336	13920	0	1	1	2	3	5	7	9	20




```
### Estadísticas descriptivas:

stat_datos <- summary_statistics(datos)

tabla <- stat_datos %>%
  kbl(caption = "Estadísticas descriptivas de variables numéricas") %>%
  kable_classic(full_width = F, html_font = "Cambria")
tabla
```

Al desagregar los productos por grupos, observamos que el mayor número se refiere a Grocery con 9,136 productos; el resto de los grupos, están por debajo de 2,000 productos. Respecto de sus scores, podemos notar que los productos de Health and Beauty y Sports reciben el mayor porcentaje de 5's en el score (79% y 72%, respectivamente). Sin embargo, el grupo de Sports es, también, el que recibe mayor porcentaje de 1's solo por debajo de Lawn & Patio (19% y 24%, respectivamente).

Sobre la desagregación de categorías, es importante destacar que la base de datos cuenta con 143 categorías, lo que dificulta su visualización en tablas y gráficas. Por lo tanto, solo se muestran las cifras para las 10 categorías con el mayor número de productos.

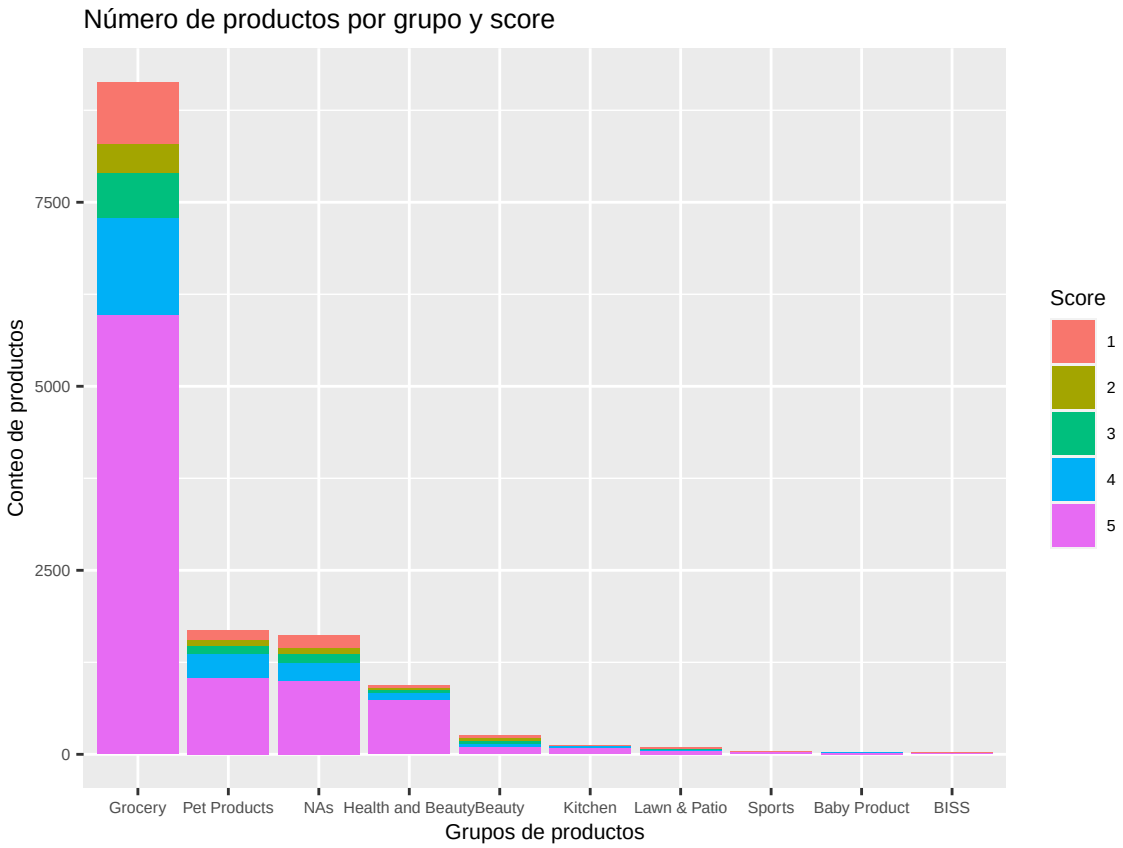
La categoría con mayor número de productos son Single-Serve Capsules & Pods (1,521 productos), por debajo de la categoría de NAs u otros que creamos anteriormente (1,613 productos). De estas 10 categorías más grandes, los productos Herbal y Canned reciben el mayor porcentaje de 5s en el score (78% y 76%, respectivamente). Por otro lado, las categorías de Peanut Butter y Products muestran el mayor porcentaje de 1s (19% y 15%).

A pesar de que no se muestran aquí, llaman la atención los productos de café instantáneo por sus altas calificaciones: recibieron un score de 4 o 5 en el 98% de compras. Asimismo, productos como el té helado muestra como calificación máxima 4; es decir, en ningún caso un producto de esta categoría fue calificado con la máxima puntuación, al igual que la categoría de Sticks & Twists. Por otro lado, los productos vendidos como orquídeas, Treats y espagueti obtuvieron la calificación máxima de 5 en todas las compras.

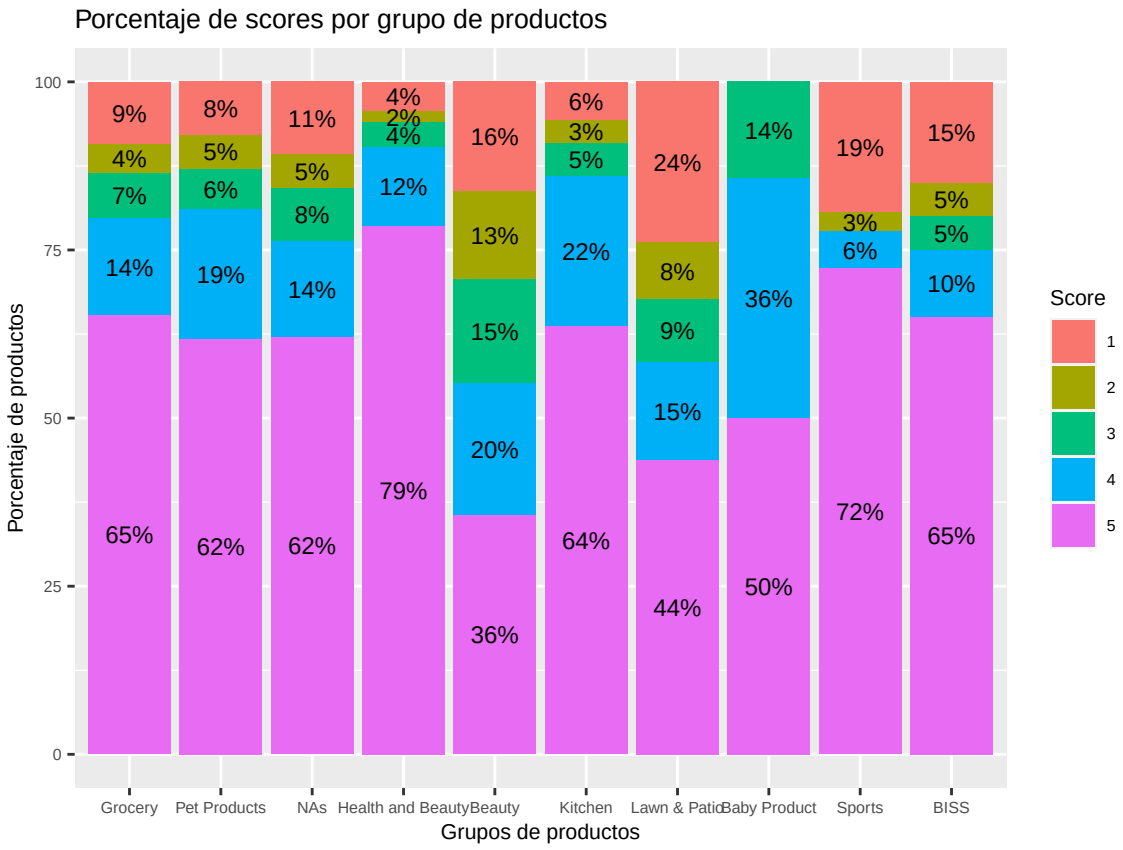
```
### Scores por grupos de productos:

tab_datos <- table(datos$Score, datos$Prod_Group)

tab_datos
```

```
  Baby Product Beauty BISS Grocery Health and Beauty Kitchen Lawn & Patio
1            0     42    3     850                41       7           23
2            0     34    1     387                15       4            8
3            4     40    1     619                35       6            9
4           10     51    2    1317               108      27           14
5           14     92   13    5963               731      77           42

   NAs Pet Products Sports
1  173          133      7
2   83           84      1
3  125          102      0
4  232          324      2
5 1000         1038     26
```

```
ggplot(data = as.data.frame(tab_datos), mapping = aes(x = reorder(Var2, -Freq), fill = Var1, y = Freq))
  geom_col() +
  labs(title="Número de productos por grupo y score", x = "Grupos de productos", y = "Conteo de producto
  theme(text = element_text(size=8))
```




```
datos %>%
  count(Score, Prod_Group) %>%
  group_by(Prod_Group) %>%
  mutate(porc = n/sum(n) * 100) %>%
  ggplot() + aes(x=reorder(Prod_Group, -n), porc, fill=factor(Score), label=paste0(round(porc, 0), "%")
  geom_bar(stat = "identity") +
  geom_text(position=position_stack(vjust=0.5), size=3) +
  labs(title= "Porcentaje de scores por grupo de productos", x="Grupos de productos", y="Porcentaje de
  theme(text = element_text(size=8))
```

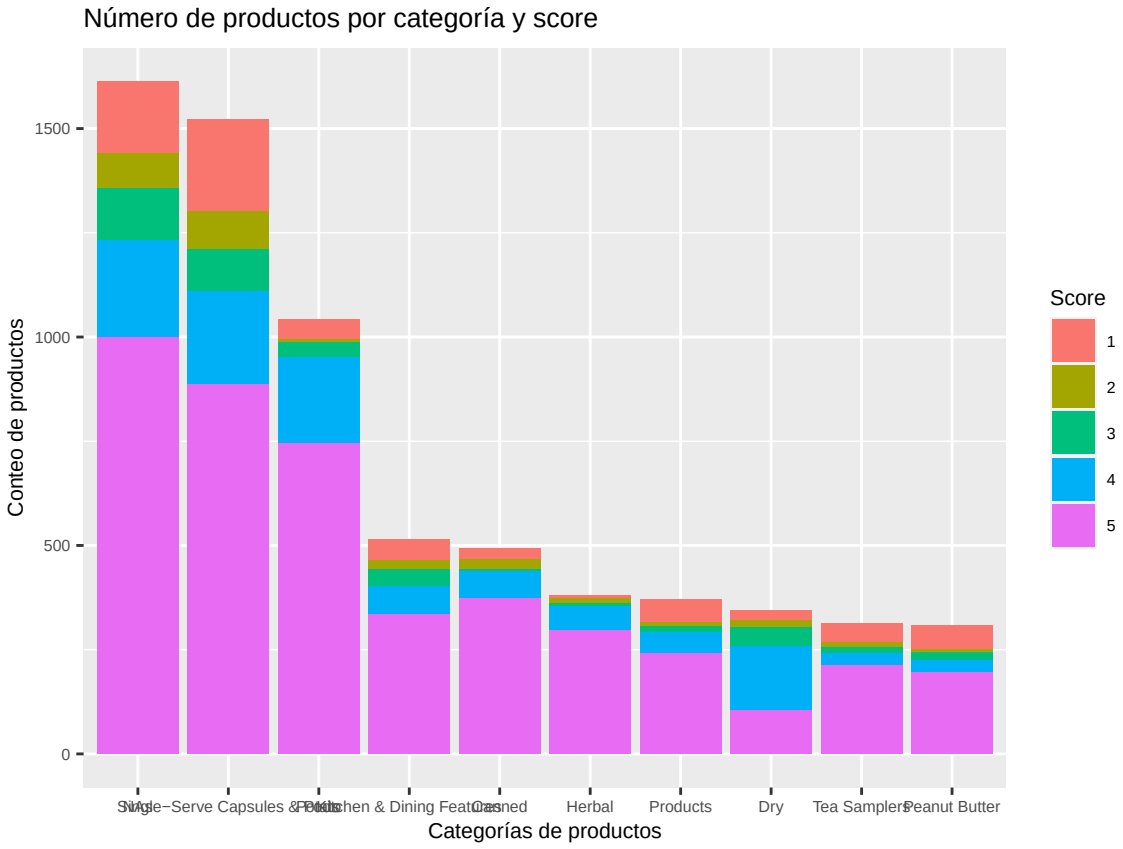



```
### Scores por categorías de productos:

datos <- datos %>%
  group_by(Prod_Category) %>%
  add_tally() %>%
  ungroup()

temp <- datos[which(datos$n >= 260),]

tab_datos <- table(temp$Score, temp$Prod_Category)

tab_datos
```

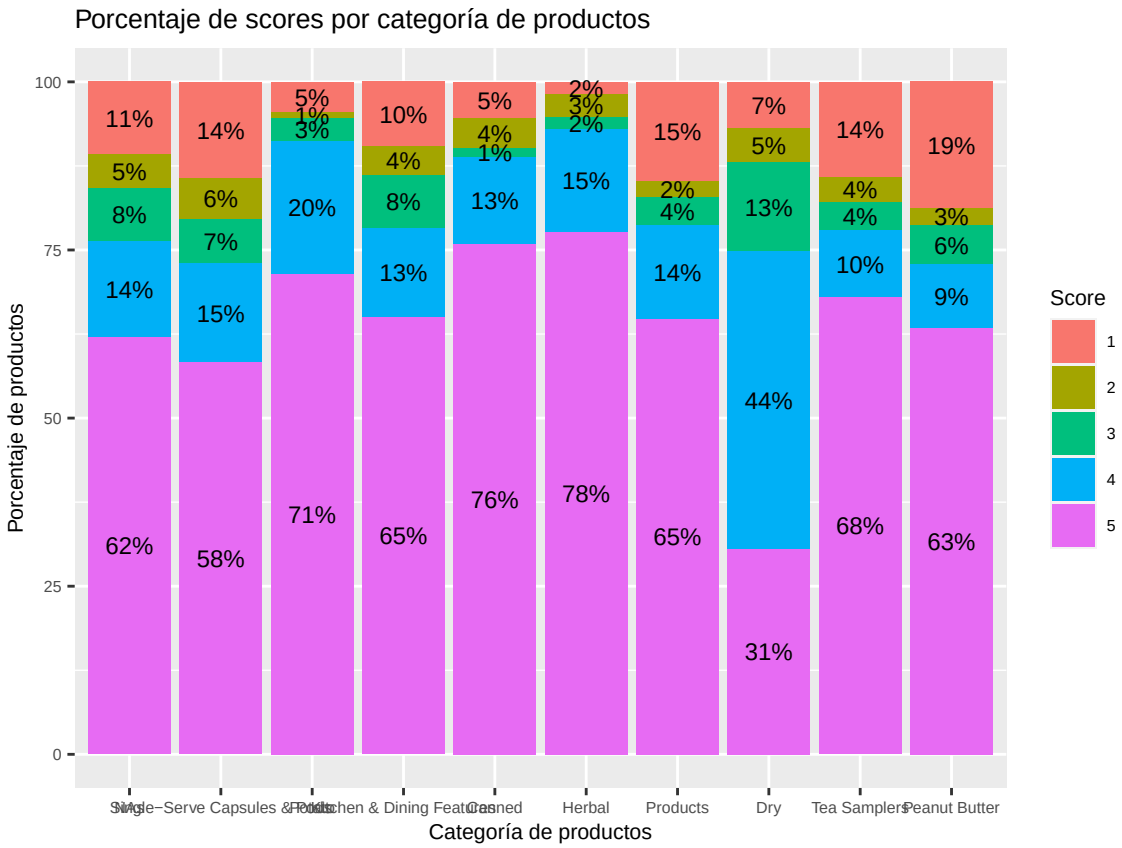
```
  Canned  Dry Herbal Kitchen & Dining Features  NAs Peanut Butter Potato
1     27   24      7                       49  173             58     47
2     22   17     13                       22   83              8      9
3      6   46      7                       41  125             18     36
4     64  152     58                       68  232             29    206
5    374  105    296                      334 1000            196    745

  Products Single-Serve Capsules & Pods Tea Samplers
1       55                         219           44
2        9                          91           12
3       15                         101           13
```

```
  4        52                          224            31
  5       240                          886           212
```

```
ggplot(data = as.data.frame(tab_datos), mapping = aes(x = reorder(Var2, -Freq), fill = Var1, y = Freq))
  geom_col() +
  labs(title="Número de productos por categoría y score", x = "Categorías de productos", y = "Conteo de
  theme(text = element_text(size=8))
```




```
temp %>%
  count(Score, Prod_Category) %>%
  group_by(Prod_Category) %>%
  mutate(porc = n/sum(n) * 100) %>%
  ggplot() + aes(x=reorder(Prod_Category, -n), porc, fill=factor(Score), label=paste0(round(porc, 0), "
  geom_bar(stat = "identity") +
  geom_text(position=position_stack(vjust=0.5), size=3) +
  labs(title= "Porcentaje de scores por categoría de productos", x="Categoría de productos", y="Porcenta
  theme(text = element_text(size=8))
```



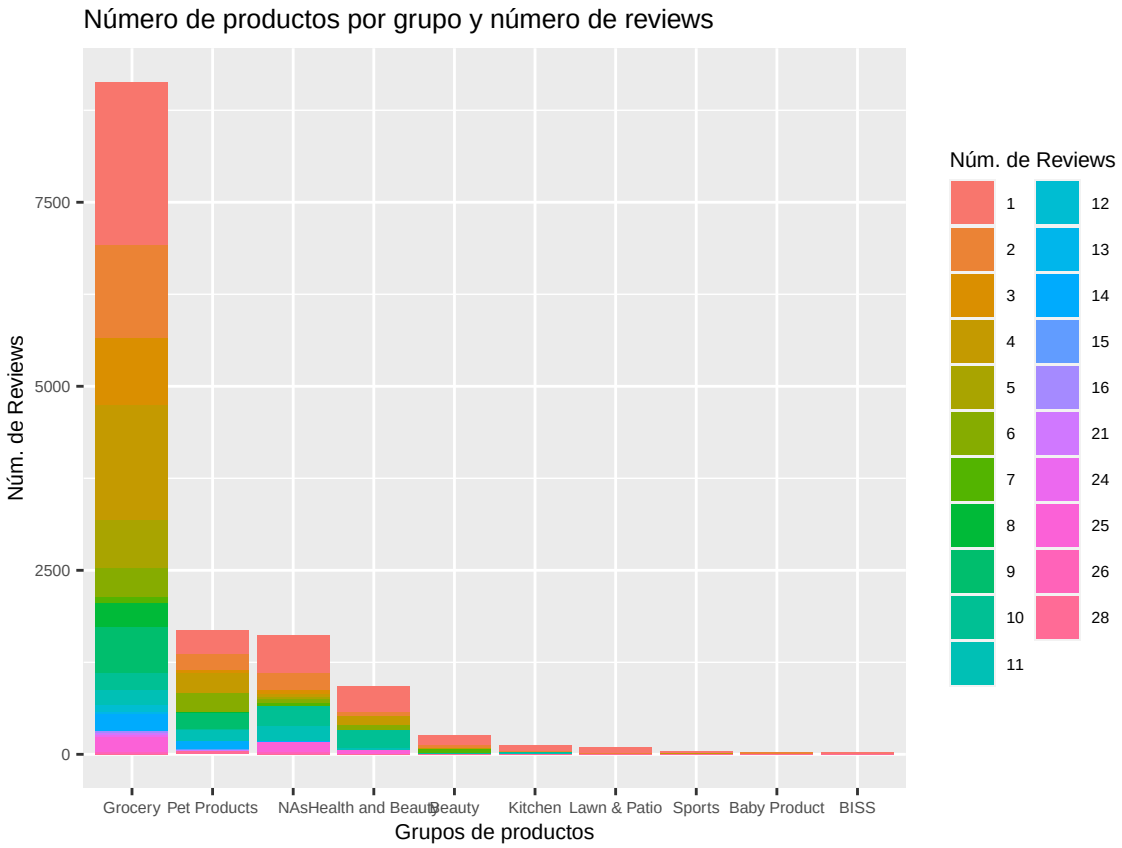

```
datos$n <- NULL
```

Además, es importante considerar el número de reviews que los productos tienen, ya sea por grupo o categoría, ya que este puede ser un indicador relevante sobre cuales son los productos con más liquidez o movimiento de la plataforma. Se puede observar que tanto por grupo como por categoría de producto, la mayoría de los productos sólo tiene un review; es decir, que la mayoría de los clientes tienden a no dejar ningún comentario, ya sea negativo o positivo. Esta indiferencia de los clientes a dar su opinión debería ser considerara por la empresa, ya que un mayor número de opiniones podría traer más liquidez a productos que no tienen tanta circulación.

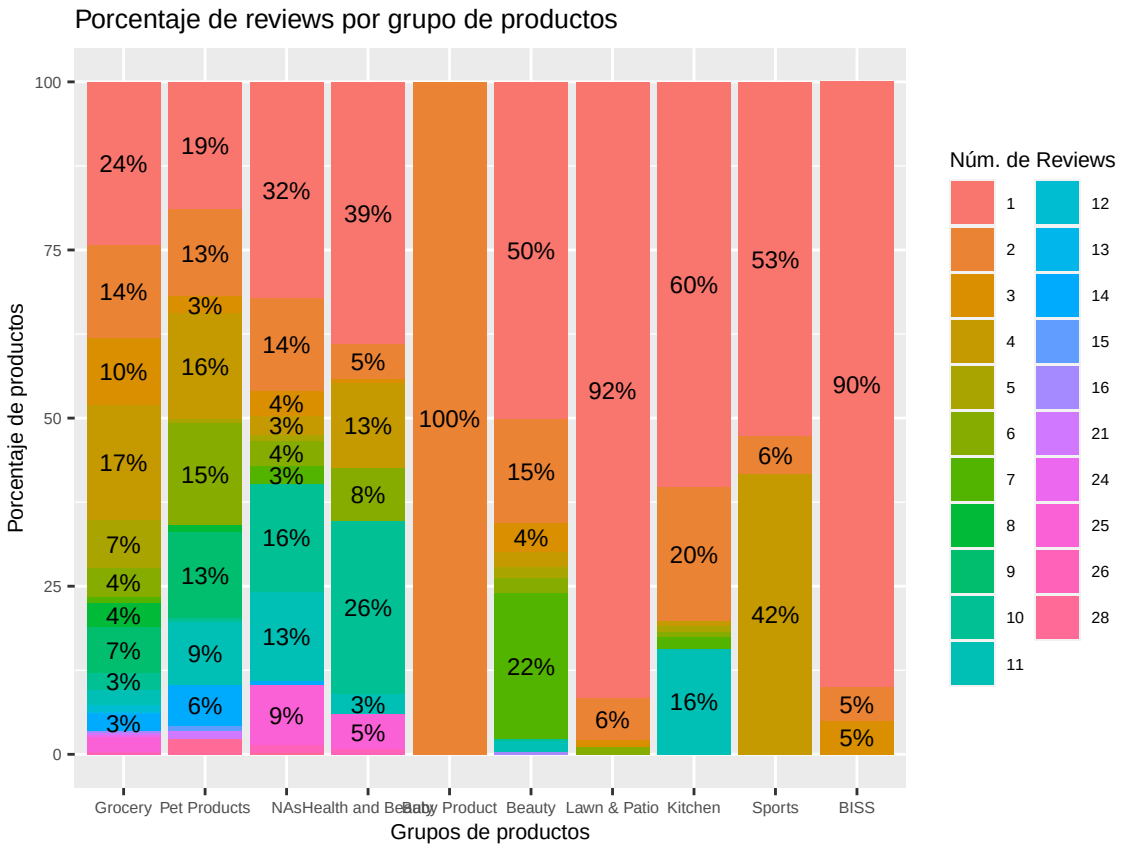
En este caso, se puede observar que para el grupo de productos Grocery, en el cual si bien un 24% de los clientes sólo deja un review, el siguiente número de reviews más frecuente es 4 con 17%. Por otro lado, el 92% de los productos del grupo de jardinería sólo tienen un review.

Respecto a categorías de productos, se observa un escenario más diversificado, ya que, en este caso, para la categoría Tea Samplers el 56% de los productos tienen 4 revisiones y en el caso de Peanut Butter es de 52%.

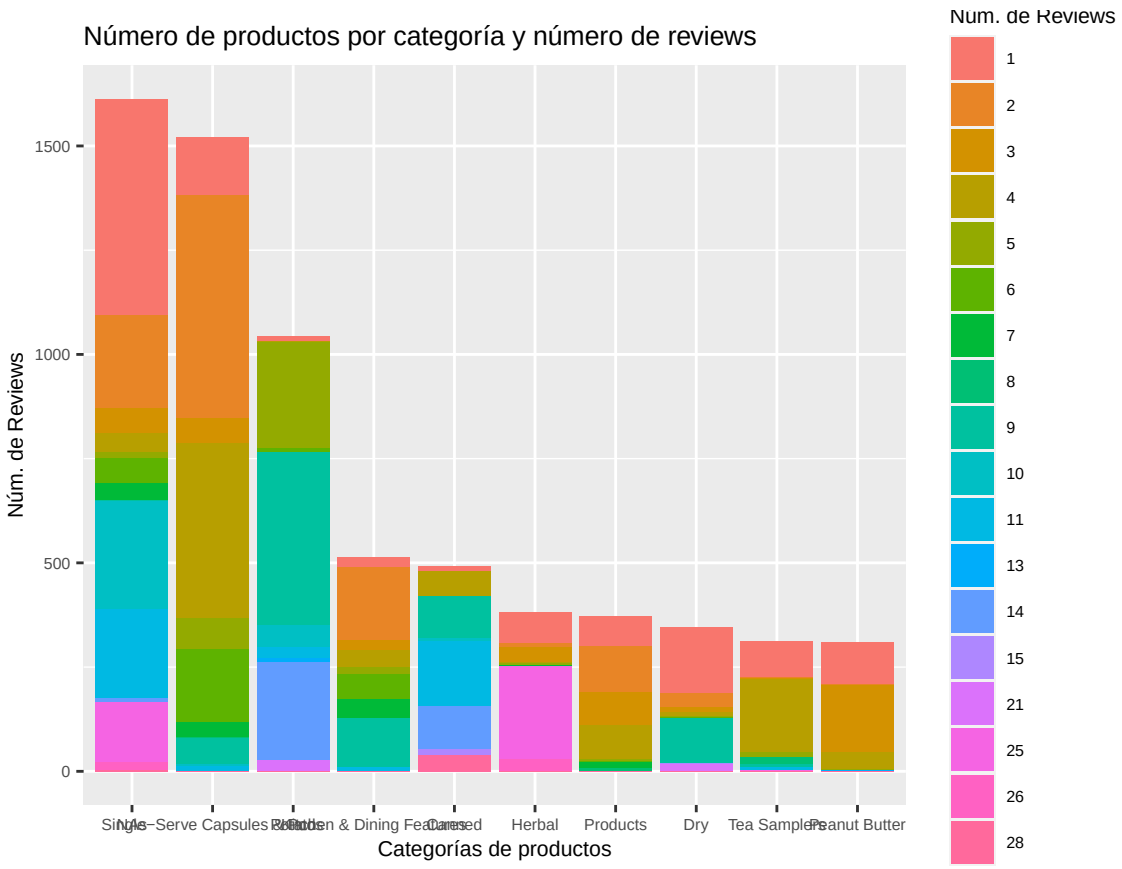
```
### Comentarios por grupos de productos:

tab_datos <- table(datos$Nrev, datos$Prod_Group)

ggplot(data = as.data.frame(tab_datos), mapping = aes(x = reorder(Var2, -Freq), fill = Var1, y = Freq))
  geom_col() +
  labs(title="Número de productos por grupo y número de reviews", x = "Grupos de productos", y = "Núm.
  theme(text = element_text(size=8))
```



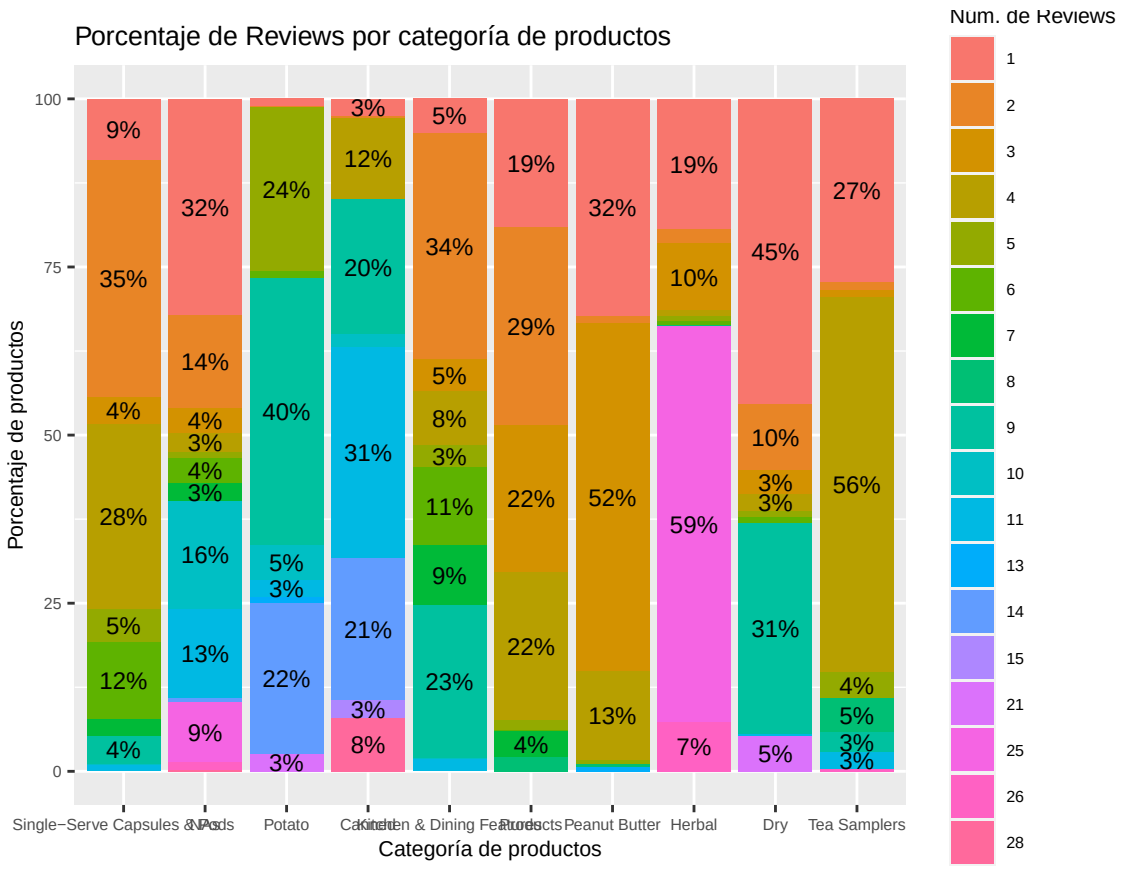

```
datos %>%
  count(Nrev, Prod_Group) %>%
  group_by(Prod_Group) %>%
  mutate(porc = n/sum(n) * 100) %>%
  ggplot() + aes(x=reorder(Prod_Group, -n), porc, fill=factor(Nrev), label=ifelse(porc>2.5, paste0(roun
  geom_bar(stat = "identity") +
  geom_text(position=position_stack(vjust=0.5), size=3) +
  labs(title= "Porcentaje de reviews por grupo de productos", x="Grupos de productos", y="Porcentaje de
  theme(text = element_text(size=8))
```




```
### Comentarios por categorías de productos:

datos <- datos %>%
  group_by(Prod_Category) %>%
  add_tally() %>%
  ungroup()

temp <- datos[which(datos$n >= 260),]

tab_datos <- table(temp$Nrev, temp$Prod_Category)

ggplot(data = as.data.frame(tab_datos), mapping = aes(x = reorder(Var2, -Freq), fill = Var1, y = Freq))
  geom_col() +
  labs(title="Número de productos por categoría y número de reviews", x = "Categorías de productos", y =
  theme(text = element_text(size=8))
```




```
temp %>%
  count(Nrev, Prod_Category) %>%
  group_by(Prod_Category) %>%
  mutate(porc = n/sum(n) * 100) %>%
  ggplot() + aes(x=reorder(Prod_Category, -n), porc, fill=factor(Nrev), label=ifelse(porc>2.5, paste0(r
  geom_bar(stat = "identity") +
  geom_text(position=position_stack(vjust=0.5), size=3) +
  labs(title= "Porcentaje de Reviews por categoría de productos", x="Categoría de productos", y="Porcen
  theme(text = element_text(size=8))
```




```
datos$n <- NULL
```

A continuación, se muestra un wordcloud con los conceptos más mencionados en los comentarios. Las 5 palabras que más destacan en términos del número de menciones son: great, good, love, coffee y best. Por lo tanto, es posible identificar que, en general, una gran parte de los comentarios tienen un sentido positivo. Además, algunos de los productos o grupos más mencionados son el café, té, comida y perros.

```
### Creamos un vector que solo contenga los comentarios:

Comentarios <- datos$Summary

V_Coment <- Corpus(VectorSource(Comentarios))

X <- TermDocumentMatrix(V_Coment)

matriz <- as.matrix(X)

words <- sort(rowSums(matriz),decreasing=TRUE)

df <- data.frame(word = names(words),freq=words)

wordcloud(words = df$word, freq = df$freq, min.freq = 1, max.words=200, random.order=FALSE, rot.per=0.3
```

3. Predicción y Análisis de Sentimiento

A continuación, usaremos diccionarios para identificar sentimientos vinculados con cada palabra y, a su vez, se le asigna un sentido (positivo o negativo) y un score que va de -5 a 5 entre emociones negativas y positivas. En la siguiente tabla, se muestran las 15 emociones más frecuentes. De estas, 12 son positivas y 3, negativas.

```
### Diccionarios y sentimientos:

sentiment_reviews = cleaned_tokens %>%
  left_join(get_sentiments("nrc")) %>%
  rename(nrc = sentiment) %>%
  left_join(get_sentiments("bing")) %>%
  rename(bing = sentiment) %>%
  left_join(get_sentiments("afinn")) %>%
  rename(afinn = value)

### Sentimientos más frecuentes:

temp <- sentiment_reviews %>%
  filter(!is.na(bing)) %>%
  count(word, bing, sort = TRUE)

tabla <- temp[which(temp$n >= 255),] %>%
  kbl(caption = "Sentimientos más frecuentes") %>%
```

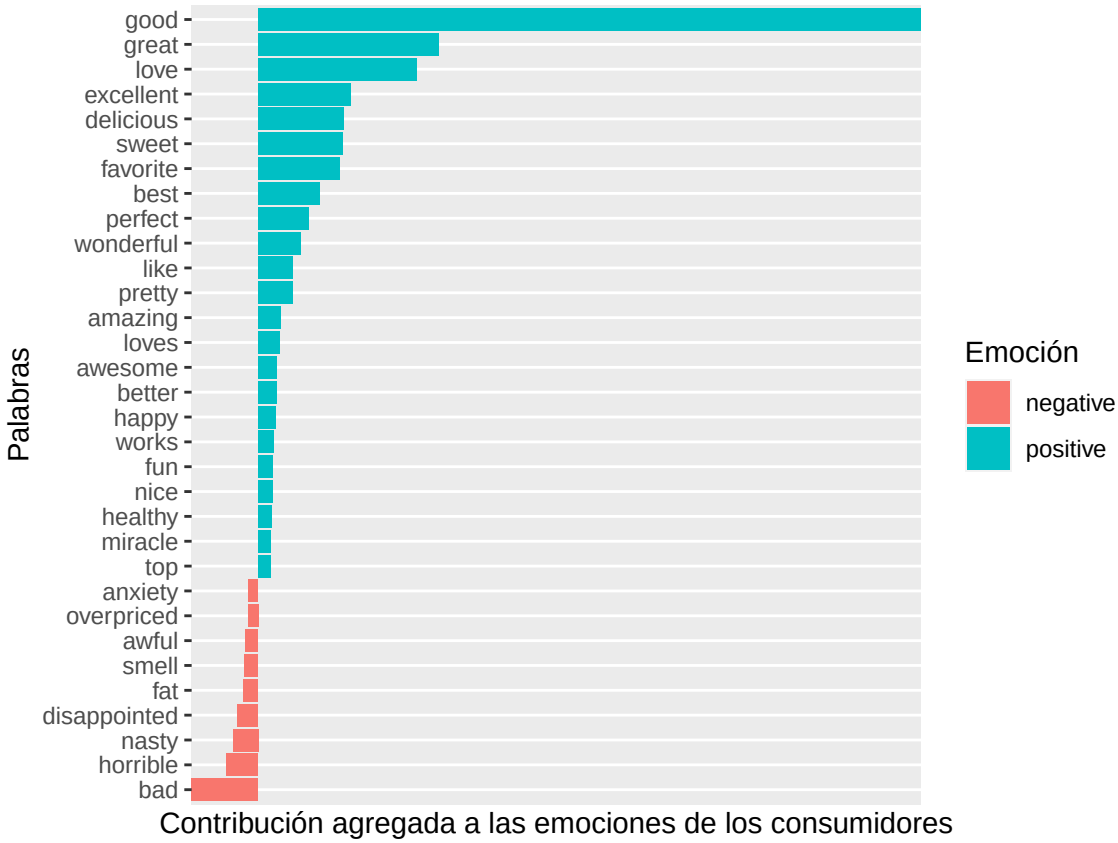
```
  kable_classic(full_width = F, html_font = "Cambria") %>%
  kable_styling(latex_options = "HOLD_position")
tabla
```

Table 3: Sentimientos más frecuentes

word	bing	n
good	positive	6680
great	positive	1821
love	positive	1598
excellent	positive	930
delicious	positive	860
sweet	positive	855
favorite	positive	819
bad	negative	670
best	positive	620
perfect	positive	508
wonderful	positive	424
like	positive	346
pretty	positive	344
horrible	negative	316
nasty	negative	255

Ahora, se muestra una gráfica con las palabras que tienen más de 100 menciones y su contribución agregada a los sentimientos de los consumidores: es decir, la suma de su score según el diccionario *Afinn*. Los conceptos que más reflejan emociones positivas entre los consumidores de Amazon son good, great y love. Por el contrario, los conceptos que más reflejan emociones negativas son bad, horrible y nasty.

```
temp %>%
  filter(n >= 100) %>%
  mutate(n = ifelse(bing == "negative", -n, n)) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n, fill = bing)) +
  geom_col() +
  coord_flip() +
  scale_y_discrete() +
  labs(y = "Contribución agregada a las emociones de los consumidores", fill = "Emoción", x = "Palabras
```



Además, se propone transformar las palabras de los reviews en variables categóricas. Posteriormente, esto nos servirá para identificar si determinadas palabras tienen una capacidad predictiva importante sobre el score. También, se crearán dummies por grupo, categoría y emociones de acuerdo con *nrc* y *bing* (se creó una categoría de NAs), así como dummies para cada mes, tomando como referencia la fecha. Ahora, la base de datos cuenta con 3,845 variables o columnas.

```
detectCores()
```

```
[1] 12
```

```
cl <- makeCluster(8)

### Dummies por grupo, categoría, palabra y mes:

sentiment_reviews <- fastDummies::dummy_cols(sentiment_reviews, select_columns = "Prod_Group")

sentiment_reviews <- fastDummies::dummy_cols(sentiment_reviews, select_columns = "Prod_Category")

sentiment_reviews <- fastDummies::dummy_cols(sentiment_reviews, select_columns = "word")

sentiment_reviews$nrc[is.na(sentiment_reviews$nrc)] <- "NAs"

sentiment_reviews$bing[is.na(sentiment_reviews$bing)] <- "NAs"

sentiment_reviews <- fastDummies::dummy_cols(sentiment_reviews, select_columns = "nrc")
```

```
sentiment_reviews <- fastDummies::dummy_cols(sentiment_reviews, select_columns = "bing")

sentiment_reviews$month <- format(as.Date(sentiment_reviews$Time), "%m")

sentiment_reviews <- fastDummies::dummy_cols(sentiment_reviews, select_columns = "month")

sentiment_reviews$month <- NULL

stopCluster(cl)
```

En este punto, la información del score de sentimientos se encuentra a nivel de palabra para los comentarios de cada producto. Por lo tanto, el paso siguiente es agrupar esta información por producto promediando el valor numérico asignado a las emociones de cada concepto. Cabe destacar que le asignamos un valor de 0 a todos los conceptos que tenían *NA* en el score del diccionario de *afinn*.

```
### Promediar score de sentimientos por producto:

detectCores()
```

```
[1] 12
```

```
cl <- makeCluster(8)

sentiment_reviews$afinn[is.na(sentiment_reviews$afinn)] <- 0
sentiment_reviews_dmr <- sentiment_reviews
sentiment_reviews <- sentiment_reviews %>%
  group_by(ProductId, UserId, Score, Time, Nrev, Length, Prod_Category, Prod_Group) %>%
  summarise_at(vars(afinn:month_10), mean) %>%
  ungroup()

stopCluster(cl)
```

Antes de aplicar los modelos para predecir el score de los productos, es importante revisar si se generaron NAs en el proceso. Como se observa en la siguiente tabla, no tenemos ningún *NA* en la base de datos.

```
### Contamos los NA por columna y sacamos el porcentaje por variable:

na_count <- sapply(sentiment_reviews, function(sentiment_reviews) (sum(length(which(is.na(sentiment_rev

### Transforma en data frame:

na_count <- data.frame(na_count)
#row.names(na_count) <- col_2
colnames(na_count)[1]<-c("Porcentaje")

### La tabla:

na_count <- na_count %>%
  kbl(caption = "Porcentaje de variables con NA") %>%
  kable_classic(full_width = F, html_font = "Cambria") %>%
  kable_styling(latex_options = "HOLD_position")
na_count
```

Table 4: Porcentaje de variables con NA

	Porcentaje
ProductId	0
UserId	0
Score	0
Time	0
Nrev	0
Length	0
Prod_Category	0
Prod_Group	0
afinn	0
Prod_Group_BabyProduct	0
Prod_Group_Beauty	0
Prod_Group_BISS	0
Prod_Group_Grocery	0
Prod_Group_HealthandBeauty	0
Prod_Group_Kitchen	0
Prod_Group_LawnPatio	0
Prod_Group_NAs	0
Prod_Group_PetProducts	0
Prod_Group_Sports	0
Prod_Category_	0
Prod_Category_Almond	0
Prod_Category_AlmondButter	0
Prod_Category_Almonds	0
Prod_Category_AsianDishes	0
Prod_Category_BakingMixes	0
Prod_Category_Balsamic	0
Prod_Category_Bars	0
Prod_Category_BarsSnacks	0
Prod_Category_Black	0
Prod_Category_Bones	0
Prod_Category_Breads	0
Prod_Category_CandyChocolate	0
Prod_Category_Canned	0
Prod_Category_CatDoors	0
Prod_Category_Categories	0
Prod_Category_Catnip	0
Prod_Category_Cereals	0
Prod_Category_ChiaSeeds	0
Prod_Category_ChiliPowder	0
Prod_Category_Ciders	0
Prod_Category_Cinnamon	0
Prod_Category_Cleansers	0
Prod_Category_CoarseSalt	0
Prod_Category_Cocoa	0
Prod_Category_Coconut	0
Prod_Category_CoconutFlakes	0
Prod_Category_CoconutWater	0
Prod_Category_CoffeePodHolders	0
Prod_Category_ColdCereals	0
Prod_Category_ComputersFeatures	0
Prod_Category_Conditioner	0
Prod_Category_Cookies	0
Prod_Category_CookingBaking	0
Prod_Category_CookingOilsVinegarsSprays	0
Prod_Category_Creams	0

Otra cosa que haremos antes de estimar el score numérico, será dividir la base en dos: el 70% como base de entrenamiento y el 30% como base de validación.

```
set.seed(1990)

sentiment_reviews <-  sentiment_reviews %>%
  mutate(training_sample=rbinom(n = nrow(sentiment_reviews), 1, 0.7))

training <- sentiment_reviews[which(sentiment_reviews$training_sample == 1),]

validation <- sentiment_reviews[which(sentiment_reviews$training_sample == 0),]
```

Posteriormente, podemos estimar el score numérico que los clientes le asignaron a cada producto utilizando la información que ya se encontraba en la base de datos (número de reviews y length), pero también el nuevo score de sentimientos que generamos a partir de los comentarios y del diccionario *afinn* y las dummies por emociones, palabras, grupos, categorías y meses.

En primer lugar, podemos correr una regresión simple de las 3,839 variables sobre el score. A continuación, se muestra la tabla de regresión omitiendo al conjunto de dummies por cuestión de espacio.

```
reg_data <- select(training, -c("ProductId", "UserId", "Time", "Prod_Category", "Prod_Group", "training

detectCores()
```

```
[1] 12
```

```
cl<-makeCluster(8)

reg_scores <- lm(Score ~ ., data = reg_data)

stopCluster(cl)

noshow <- colnames(reg_data[5:3840])

stargazer(reg_scores, title = "Regresión sobre el score numérico", font.size="scriptsize", digits=3, om
```

```
\begin{table}[!htbp] \centering
  \caption{Regresión sobre el score numérico}
  \label{}
\scriptsize
\begin{tabular}{@{\extracolsep{5pt}}lc}
\\[-1.8ex]\hline
\hline \\[-1.8ex]
 & \multicolumn{1}{c}{\textit{Dependent variable:}} \\
\cline{2-2}
\\[-1.8ex] & Score \\
\hline \\[-1.8ex]
 Nrev & 0.017$^{***}$ \\
  & (0.003) \\
  & \\
 Length & $-$0.018$^{***}$ \\
  & (0.006) \\
  & \\
```

```
 afinn & $-$1.115 \\
  & (2.127) \\
  & \\
 Constant & 8.285 \\
  & (6.388) \\
  & \\
\hline \\[-1.8ex]
Observations & 9,297 \\
R$^{2}$ & 0.866 \\
Adjusted R$^{2}$ & 0.808 \\
Residual Std. Error & 0.563 (df = 6506) \\
F Statistic & 15.034$^{***}$ (df = 2790; 6506) \\
\hline
\hline \\[-1.8ex]
\textit{Note:}  & \multicolumn{1}{r}{$^{*}$p$<$0.1; $^{**}$p$<$0.05; $^{***}$p$<$0.01} \\
\end{tabular}
\end{table}
```

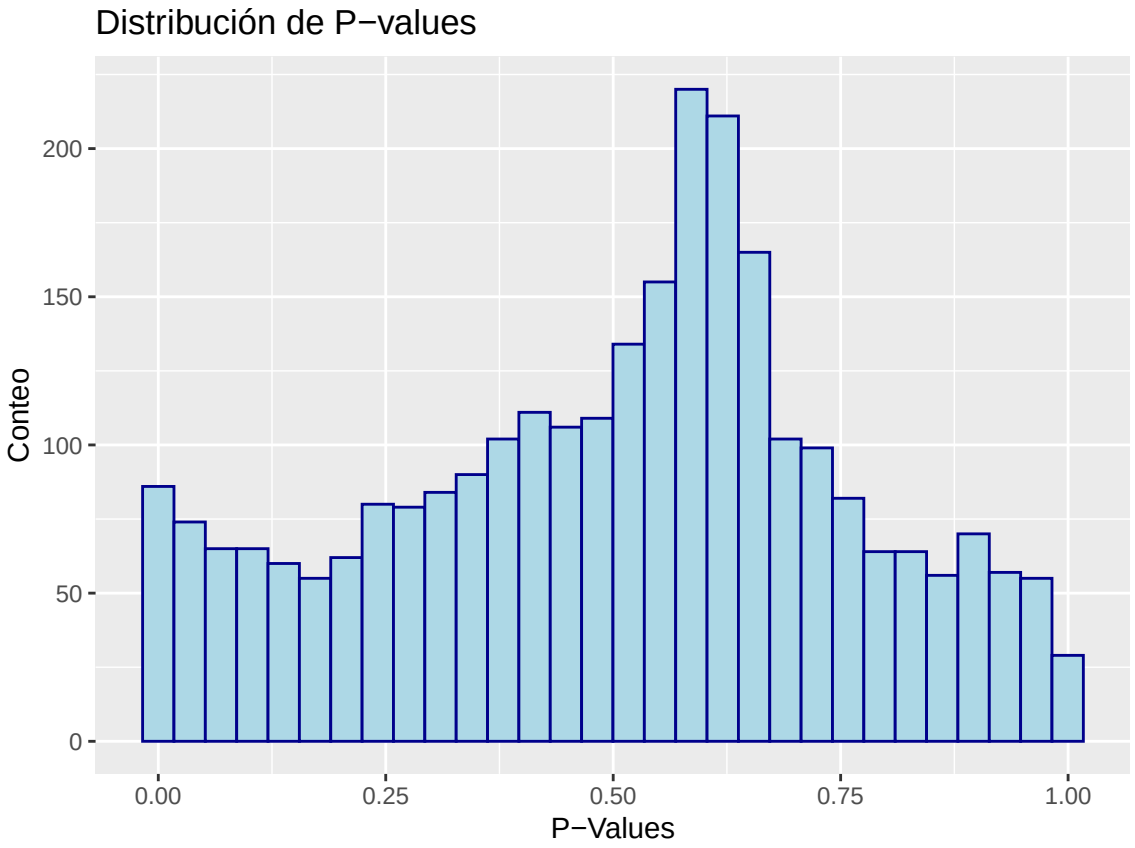
Se tienen 2,791 p-values (es probable que múltiples dummies tuvieran multicolinealidad y fueran eliminadas de la especificación), cuyo histograma se muestra a continuación:

```
pval <- summary(reg_scores)$coefficients
pval <- as.data.frame(pval)
pval$Estimate <- NULL
pval$`Std. Error` <- NULL
pval$`t value` <- NULL

pval <- pval %>%
  kbl(caption = "P-Values de la regresión de scores") %>%
  kable_classic(full_width = F, html_font = "Cambria") %>%
  kable_styling(latex_options = "HOLD_position")
pval
```

Table 5: P-Values de la regresión de scores

	Pr(>|t|)
(Intercept)	0.1946706
Nrev	0.0000000
Length	0.0045533
afinn	0.6002029
Prod_Group_BabyProduct	0.9015172
Prod_Group_Beauty	0.9946183
Prod_Group_BISS	0.7638015
Prod_Group_Grocery	0.7178295
Prod_Group_HealthandBeauty	0.6516703
Prod_Group_Kitchen	0.9138723
Prod_Group_LawnPatio	0.0558246
Prod_Group_NAs	0.8052986
Prod_Group_PetProducts	0.3791116
Prod_Category_	0.2159890
Prod_Category_Almond	0.3990791
Prod_Category_AlmondButter	0.5604188
Prod_Category_Almonds	0.9907855
Prod_Category_AsianDishes	0.9516525
Prod_Category_BakingMixes	0.9756222
Prod_Category_Balsamic	0.5330506
Prod_Category_Bars	0.7209949
Prod_Category_BarsSnacks	0.9815678
Prod_Category_Black	0.5899745
Prod_Category_Bones	0.2210342
Prod_Category_Breads	0.0042687
Prod_Category_CandyChocolate	0.6281183
Prod_Category_Canned	0.5459438
Prod_Category_CatDoors	0.0200511
Prod_Category_Categories	0.6065405
Prod_Category_Catnip	0.1623214
Prod_Category_Cereals	0.0003333
Prod_Category_ChiaSeeds	0.0669953
Prod_Category_ChiliPowder	0.5074170
Prod_Category_Ciders	0.0000001
Prod_Category_Cinnamon	0.5420552
Prod_Category_Cleansers	0.9016921
Prod_Category_CoarseSalt	0.0515213
Prod_Category_Cocoa	0.1356764
Prod_Category_Coconut	0.7160889
Prod_Category_CoconutFlakes	0.7108353
Prod_Category_CoconutWater	0.0692114
Prod_Category_CoffeePodHolders	0.5354749

```
pval <- summary(reg_scores)$coefficients
pval <- as.data.frame(pval)
pval$Estimate <- NULL
pval$`Std. Error` <- NULL
pval$`t value` <- NULL

ggplot(pval, aes(x=`Pr(>|t|)`)) +
  geom_histogram(color="darkblue", fill="lightblue") +
  ggtitle("Distribución de P-values") +
  labs(x="P-Values", y="Conteo")
```



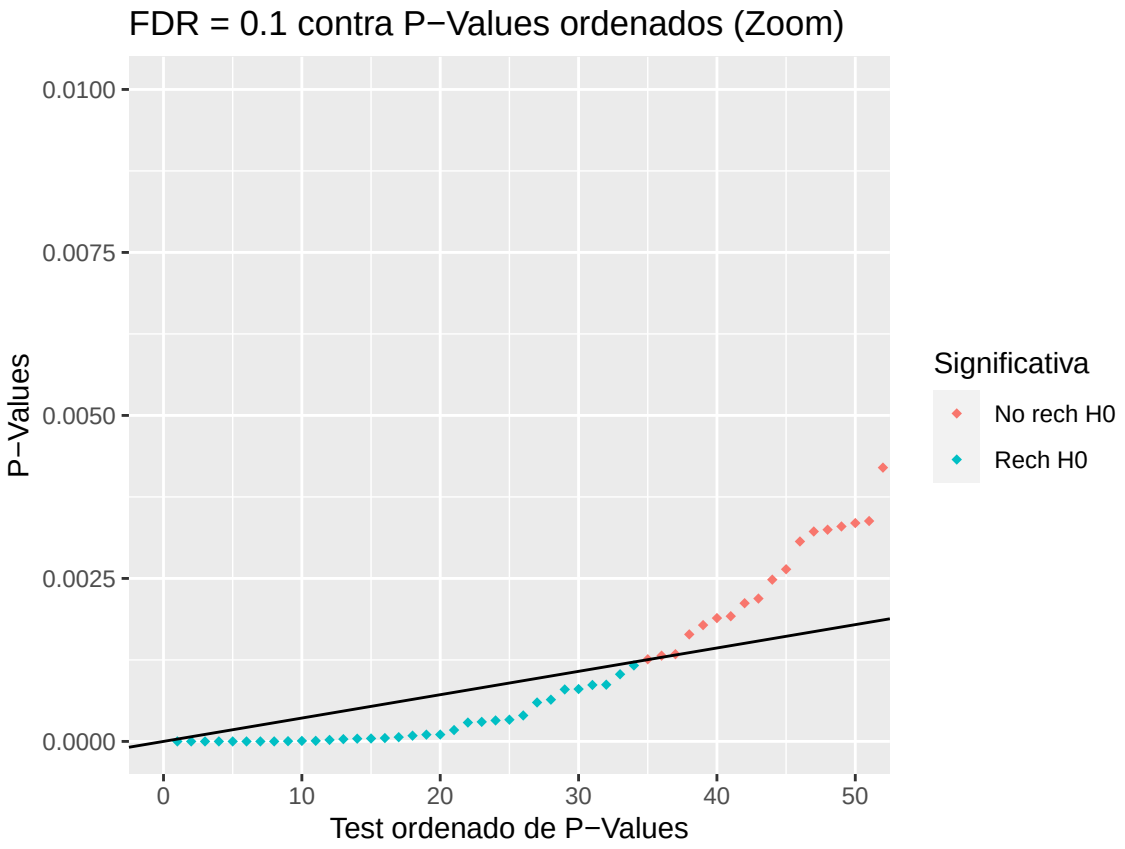
Como se observa en el histograma anterior, hay múltiples coeficientes no significativos que podrían causar ruido en la estimación. Dos métodos que sirven para seleccionar qué variables tomar en cuenta al hacer predicciones son usando False Discovery Rate (FDR) o el método de Holm-Bonferroni (HB).

A continuación, se lleva a cabo un ajuste FDR al 10% para reducir las variables:

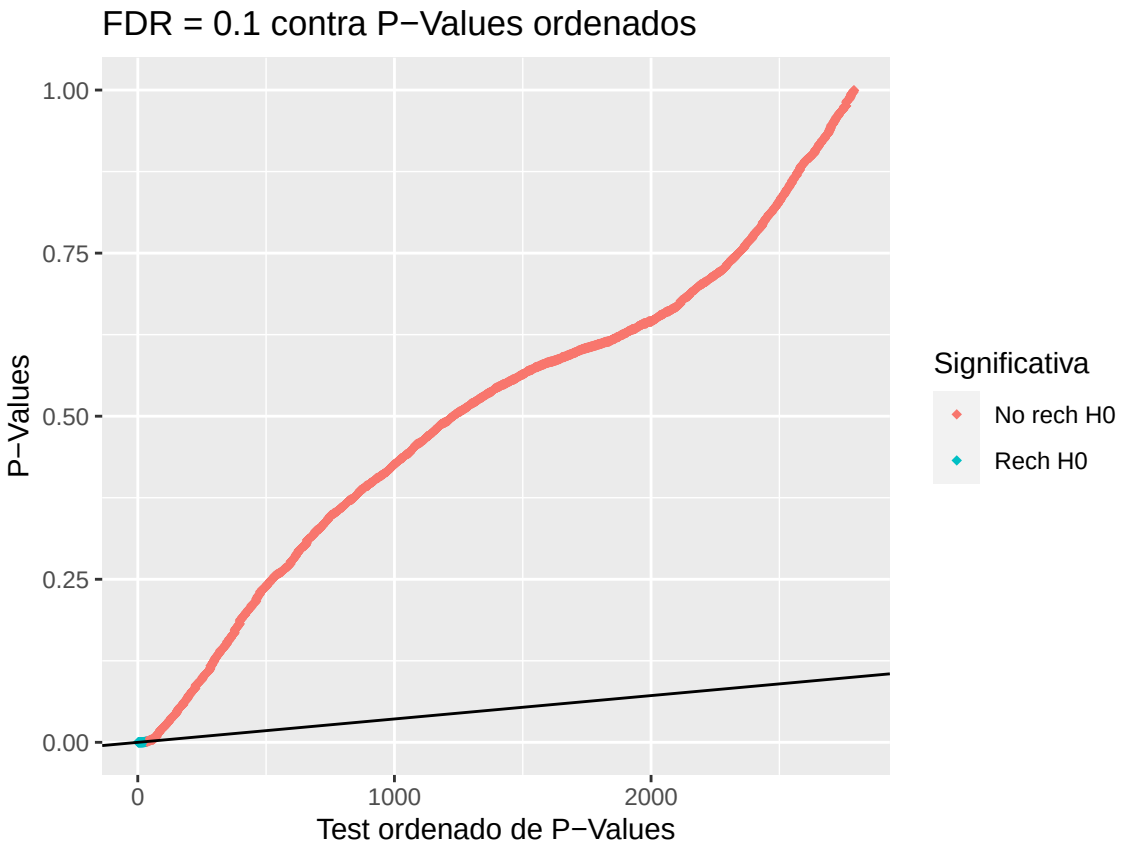
```
pval_1 <- arrange(pval)%>%
mutate(ranking = row_number(pval))

pval_1 <- mutate(pval_1, `d*i/n` = 0.10*(ranking/2791))

pval_1 <- mutate(pval_1, si = pval <= `d*i/n`)

pval_1$significativa[pval_1$si != "TRUE" ] <- "No rech H0"
```

```
pval_1$significativa[pval_1$si != "FALSE" ] <- "Rech H0"
```

```
ggplot(pval_1, aes(x=ranking, y=`Pr(>|t|)`, color=significativa)) + geom_point(shape=18)+
  ggtitle("FDR = 0.1 contra P-Values ordenados (Zoom)") +
  labs(y="P-Values", x="Test ordenado de P-Values") +
  geom_abline(slope=(.1/2791)) +labs(color = "Significativa") +
  coord_cartesian(xlim=c(0,50), ylim=c(0,.01))
```




```
ggplot(pval_1, aes(x=ranking, y=`Pr(>|t|)`, color=significativa)) + geom_point(shape=18)+
  ggtitle("FDR = 0.1 contra P-Values ordenados") +
  labs(y="P-Values", x="Test ordenado de P-Values") +
  geom_abline(slope=(.1/2791)) +labs(color = "Significativa")
```

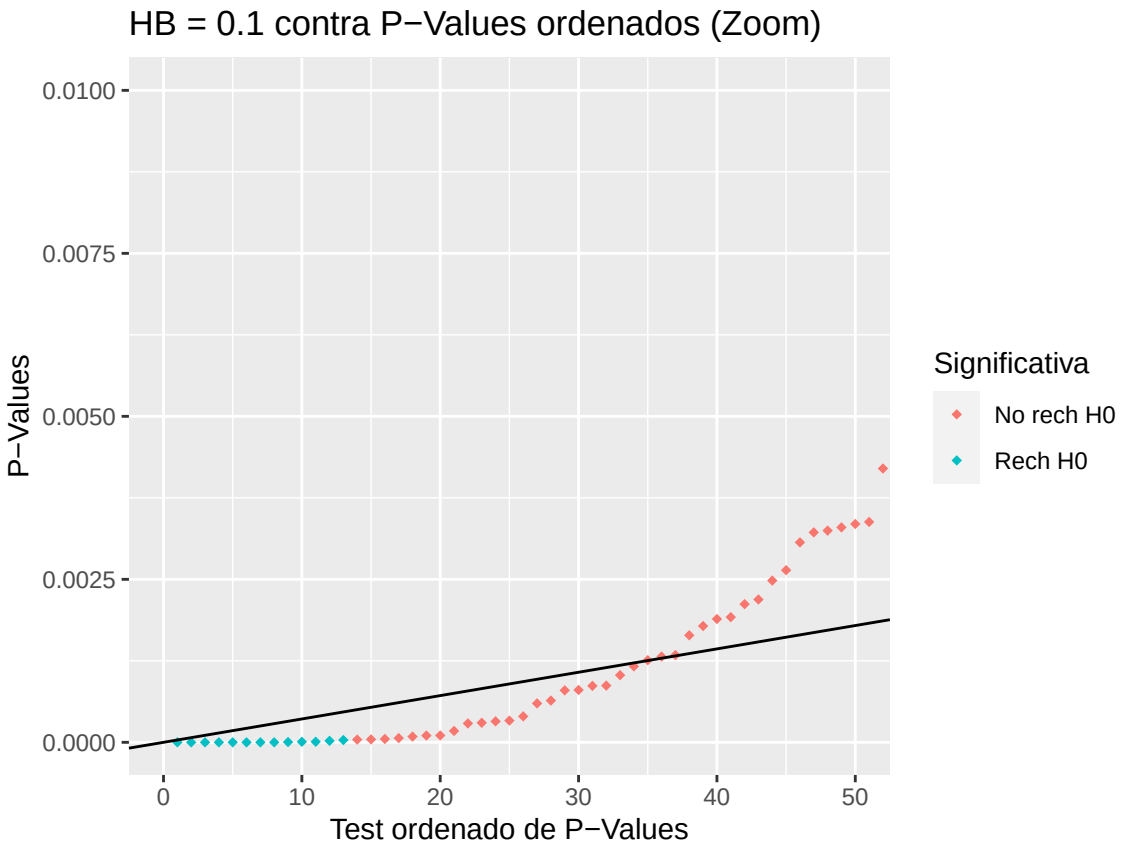



```
table(pval_1$significativa)
```

```
No rech H0    Rech H0
      2757         34
```

Como se puede observar, se tienen 34 p-values que cumplen el criterio FDR. Ahora se procede a hacer el análisis de Holm-Bonferroni:

```
pval_1 <- mutate(pval_1, HB = 0.10/(2791+1-ranking))
pval_1 <- mutate(pval_1, sighb = pval < HB)

pval_1$significativa_HB[pval_1$sighb != "TRUE" ] <- "No rech H0"
pval_1$significativa_HB[pval_1$sighb != "FALSE" ] <- "Rech H0"

ggplot(pval_1, aes(x=ranking, y=`Pr(>|t|)`, color=significativa_HB)) + geom_point(shape=18) +
  ggtitle("HB = 0.1 contra P-Values ordenados (Zoom)") +
  labs(y="P-Values", x="Test ordenado de P-Values") +
  labs(color = "Significativa") +
  geom_abline(slope=(.1/2791)) +
  coord_cartesian(xlim=c(0,50), ylim=c(0,.01))
```




```
table(pval_1$significativa_HB)
```

```
No rech H0    Rech H0
      2778         13
```

Se observa que, bajo el criterio HB, solo 13 variables cumplen el criterio, por lo que la probabilidad de cometer el error tipo 1 disminuye. Esto quiere decir que, para evaluar que variables tienen mayor peso estadístico en el score de un producto, solo es necesario considerar estas variables, de acuerdo con el criterio HB.

```
### Base de datos para meter las predicciones:

predicciones <- validation %>% select(ProductId, UserId, Time, Score)
```

Por lo tanto, se procedió a realizar las predicciones sobre la base de validación utilizando las variables del criterio de FDR y las del criterio de HB:

```
### Seleccionamos las variables que cumplen con el criterio FDR:

test_dataFDR <- select(validation, c("Nrev", "Prod_Category_Cereals", "Prod_Category_Ciders", "Prod_Cat

### Estimamos la regresión con variables resultantes del FDR:

reg_scoresFDR <- lm(Score ~ ., data = test_dataFDR)
```

```
predict_FDR <- predict(reg_scoresFDR, validation, type = "response")
```

```
Warning in predict.lm(reg_scoresFDR, validation, type = "response"): prediction
from a rank-deficient fit may be misleading
```

```
predicciones <- bind_cols(predicciones, predict_FDR)

predicciones <- rename(predicciones, FDR = 5)
```

```
### Seleccionamos las variables que cumplen con el criterio Holm - Bonferroni:

test_dataHB <- select(validation, c("Nrev", "Prod_Category_Ciders", "word_bear", "word_coat", "word_cor

# Estimamos la regresión con variables resultantes del Holm - Bonferroni:

reg_scoresHB <- lm(Score ~ ., data = test_dataHB)

predict_HB <- predict(reg_scoresHB, validation, type = "response")
```

```
Warning in predict.lm(reg_scoresHB, validation, type = "response"): prediction
from a rank-deficient fit may be misleading
```

```
predicciones <- bind_cols(predicciones, predict_HB)

predicciones <- rename(predicciones, HB = 6)
```

El siguiente paso es contruir un LASSO: el cual estima una λ que representa el costo de incluir cada variable. A partir de ese hiperparámetro, LASSO fija las β de variables que aportan poco o nada a la estimación en cero para evitar que causen ruido y resten grados de libertad.

Dado que la variable Scores no es una variable dicotómica, es necesario realizar un Distribute Multinomial Regression (DMR). Cada clase se puede correr de manera independiente, lo que hace que no tengamos que estimar tantos parámetros y podamos agilizar el proceso. Lo más interesante es que también hace selección de variables tipo LASSO y puedes usar parallel para tener resultados más rápidos.

```
detectCores()
```

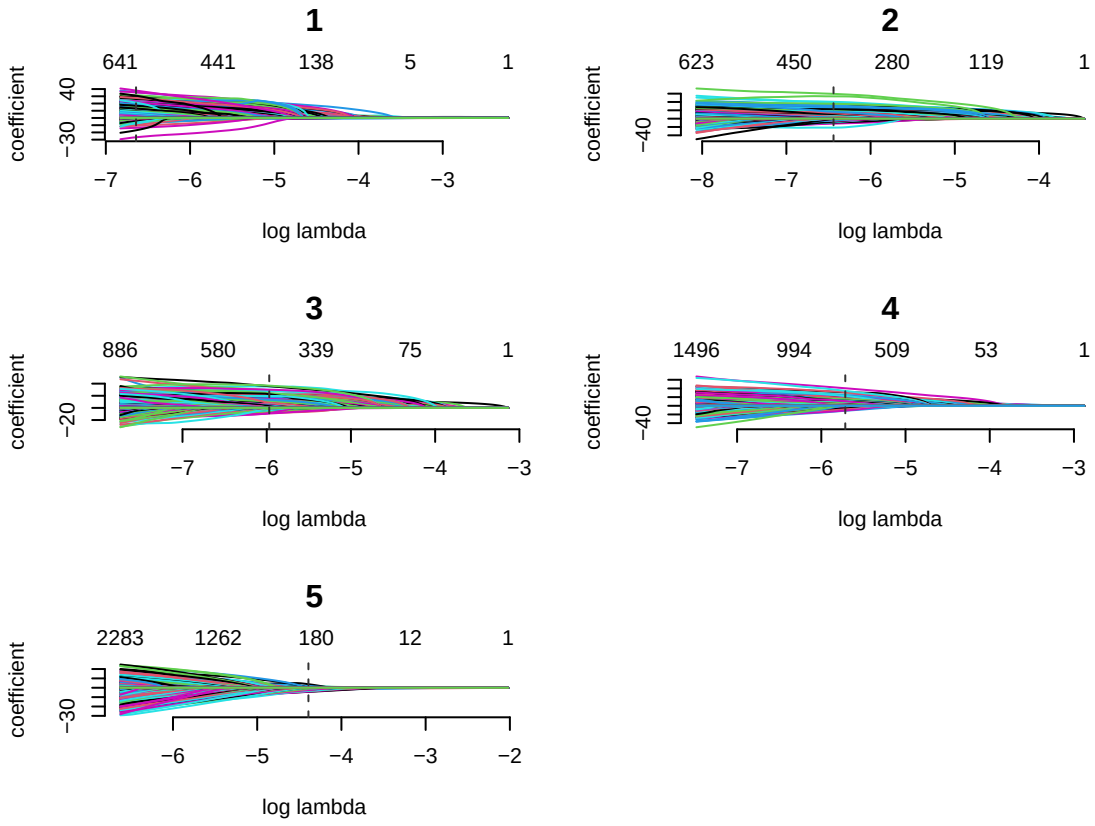
```
[1] 12
```

```
cl <- makeCluster(8)
cl
```

```
socket cluster with 8 nodes on host 'localhost'
```

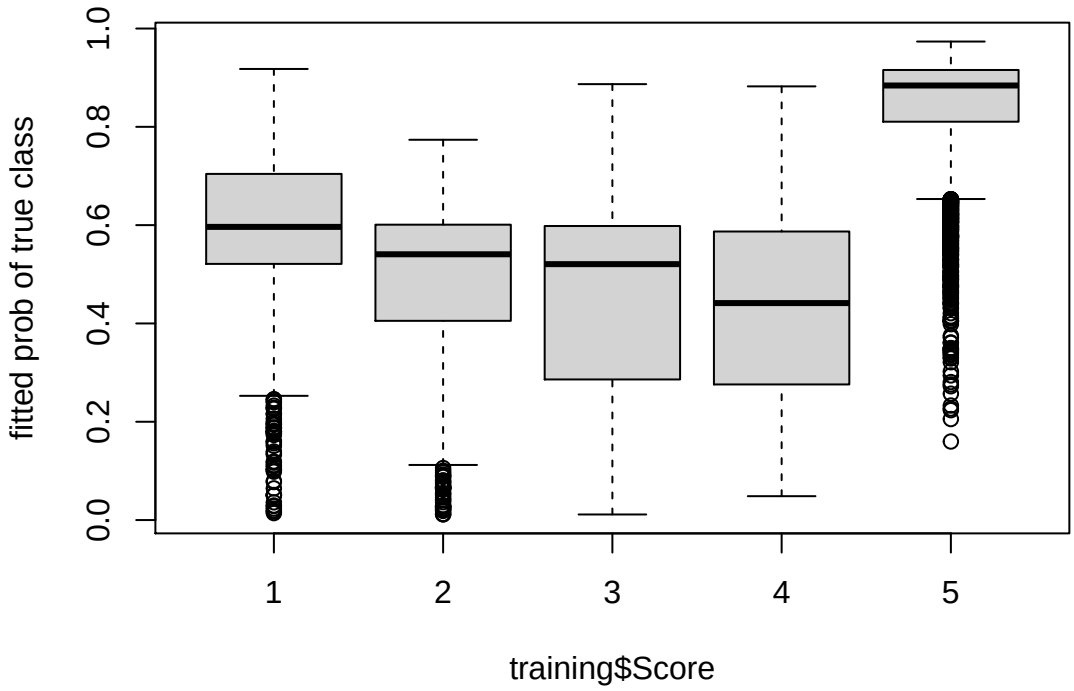
```
training_score<- as.factor(training$Score)

### Matriz con los regresores:

temp <- training %>% select(-ProductId,-UserId, -Time, -Prod_Category, -Prod_Group, -training_sample, -
```

```
training_mtx <- sparse.model.matrix(~.+0, data = temp)

fits <- dmr(cl, training_mtx, training_score)
#plot(fits$`1`)
#plot(fits$`2`)
#plot(fits$`3`)
#plot(fits$`4`)
#plot(fits$`5`)

par(mfrow=c(3,2))
for(j in 1:5){
    plot(fits[[j]])
    mtext(names(fits)[j],font=2,line=2) }

stopCluster(cl)
```




```
### AICc model selection.
### Aquí lo que nos da son los coeficientes por clase los que son cero y los que no:

B <- coef(fits)

## Fitted probability by true response
par(mfrow=c(1,1))
P <- predict(B, training_mtx, type="response")
boxplot(P[cbind(1:9297,training$Score)]~training$Score,
```

```
    ylab="fitted prob of true class")
```

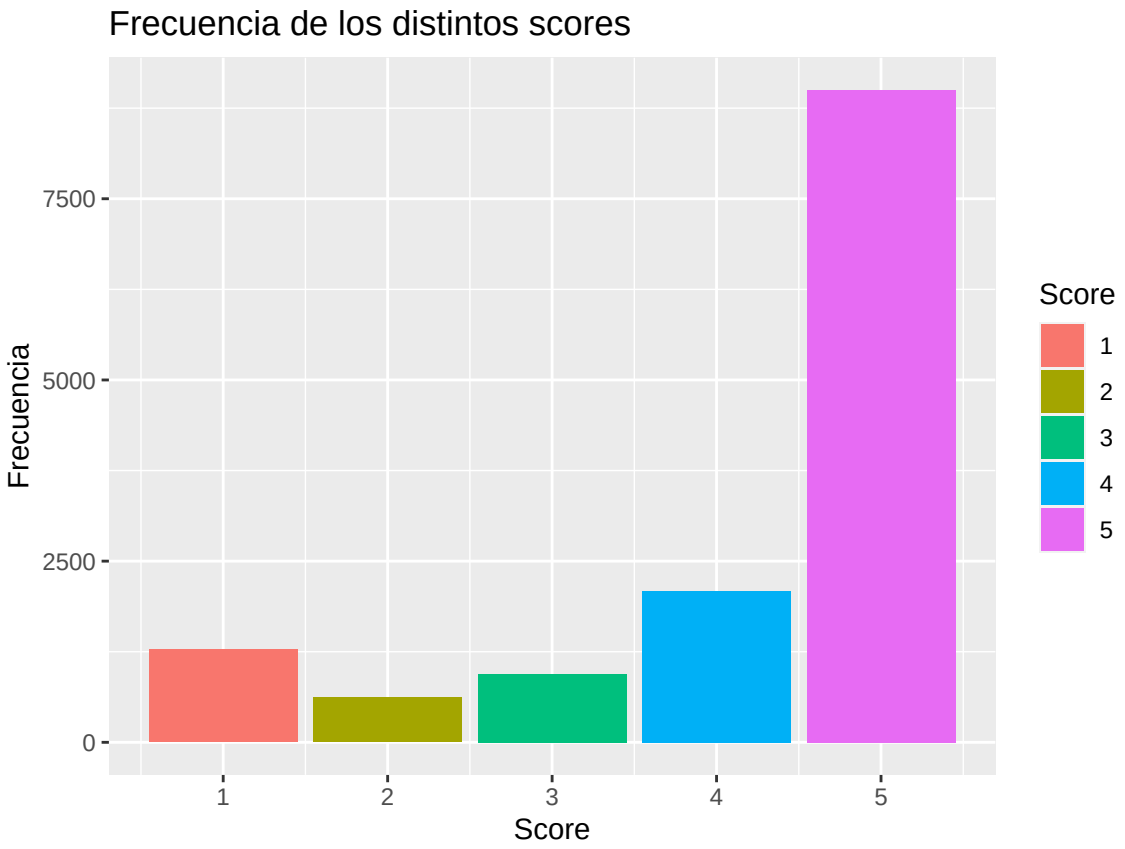



```
##############

### Matriz de covariantes de la base de validación:

temp <- validation %>% select(-ProductId,-UserId, -Time, -Prod_Category, -Prod_Group, -training_sample,

validation_mtx <- sparse.model.matrix(~.+0, data = temp)

### Predecir sobre la base de validación:

pred <- predict(B, validation_mtx, type = "response", select = "1se")

pred <- drop(pred)

pred <- as.data.frame(pred)

### Pegamos las predicciones en validación:

predicciones <- bind_cols(predicciones, pred)

### Le asignamos la clase con mayor probabilidad:

predicciones <- predicciones %>%
```

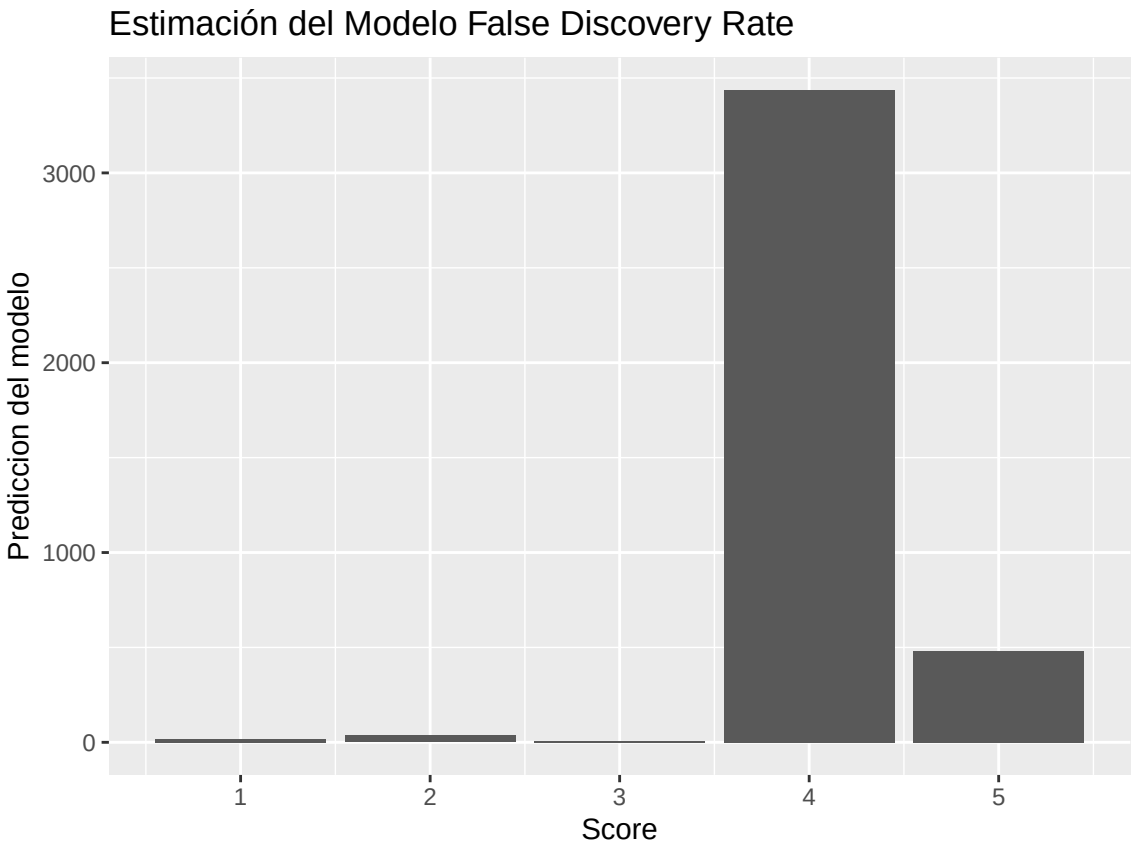
```
  rownames_to_column('id') %>%
  gather(dept, cnt, '1':'5') %>%
  group_by(id) %>%
  slice(which.max(cnt))

predicciones <- predicciones %>% select(-cnt)

predicciones <- rename(predicciones, DMR = dept)
```

A continuación, se muestran las gráficas de dispersión de los tres modelos, LASSO, FDR y HB (redondeados) contra el score numérico original en la base de validación.

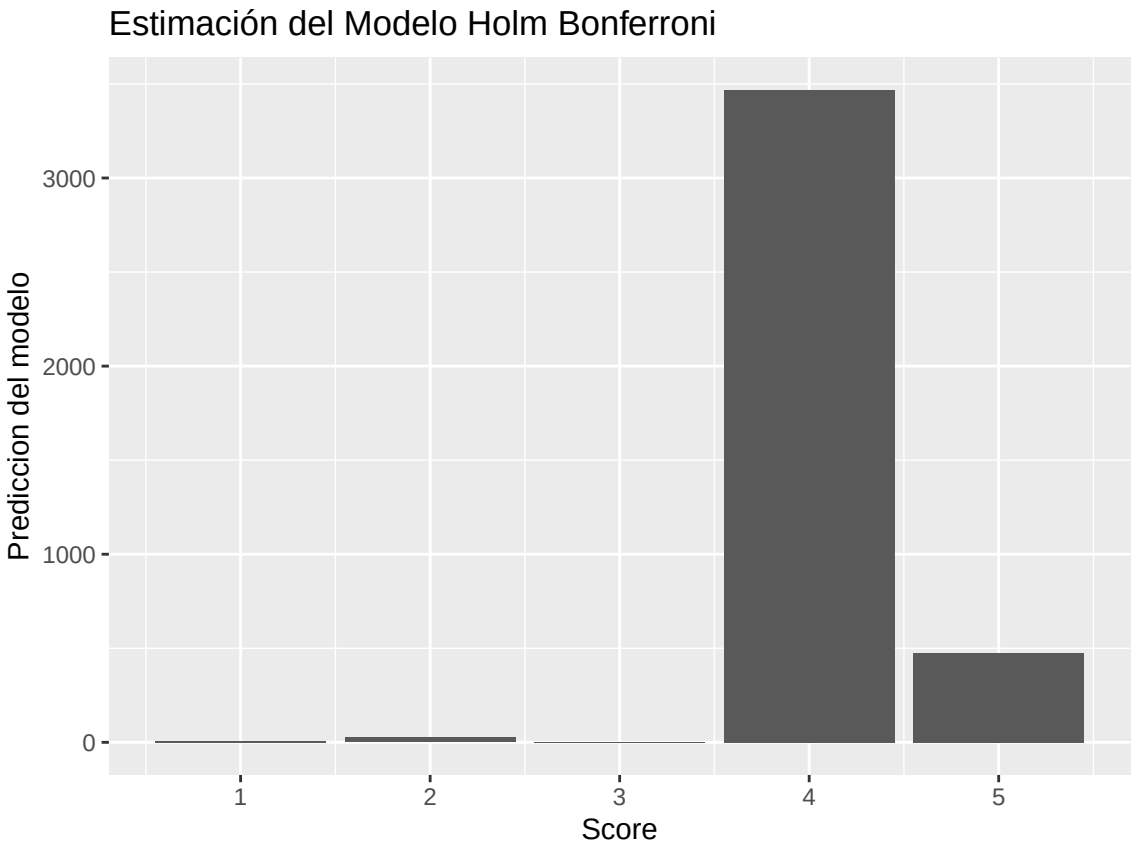
```
### Primero, hay que redondear las predicciones de FDR y HB:

predicciones$FDR[predicciones$FDR>5] <- 5
predicciones$HB[predicciones$HB>5] <- 5

round_FDR <- data.frame(round(predicciones$FDR, 0))
round_HB <- data.frame(round(predicciones$HB, 0))

predicciones <- bind_cols(predicciones, round_FDR)
predicciones <- bind_cols(predicciones, round_HB)

predicciones <- rename(predicciones, c("round_FDR" = "round.predicciones.FDR..0."))
predicciones <- rename(predicciones, c("round_HB" = "round.predicciones.HB..0."))

### Gráficas:

ggplot(data = datos, mapping = aes(x = Score, fill = as.factor(Score))) +
  geom_bar() +
  labs(title="Frecuencia de los distintos scores", x = "Score", y = "Frecuencia", fill = "Score")
```



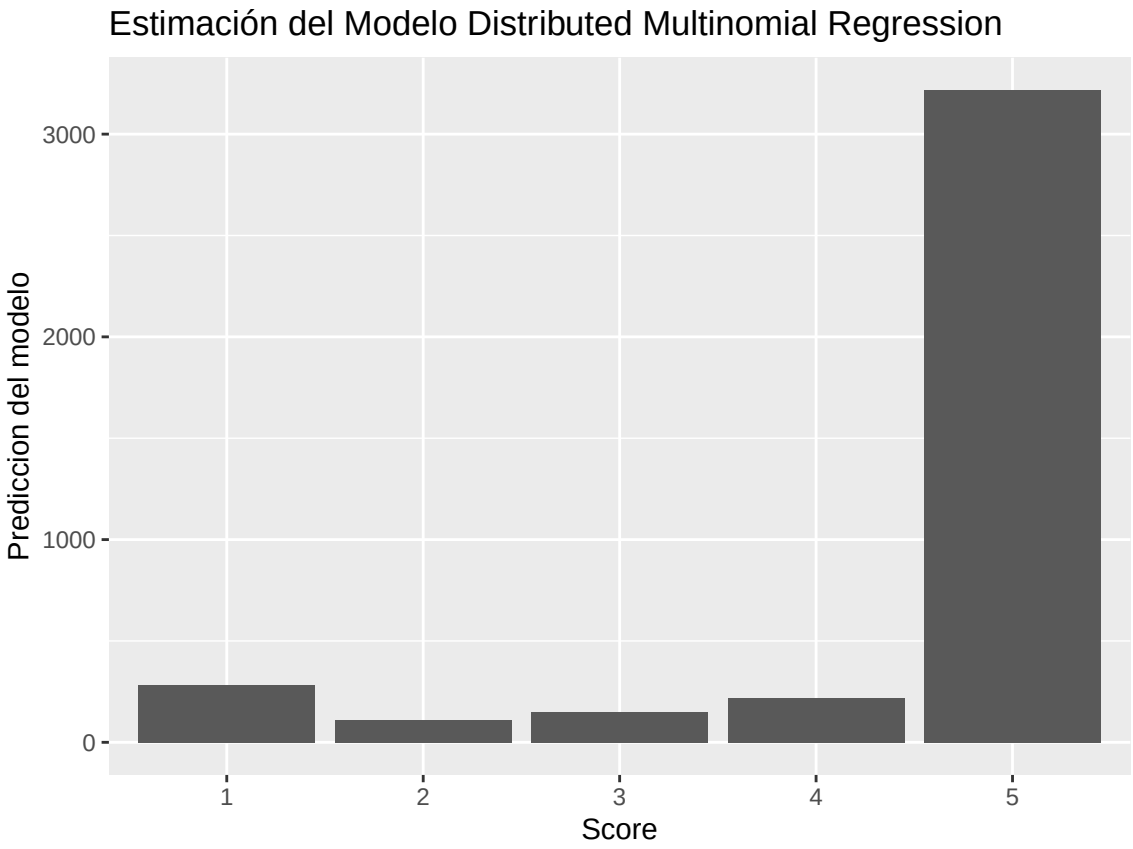

```
charts_FDR <- ggplot(predicciones, aes(round_FDR)) +
  geom_bar() +
  ggtitle("Estimación del Modelo False Discovery Rate") +
  labs(x="Score", y="Prediccion del modelo")
charts_FDR
```




```
charts_HB <- ggplot(predicciones, aes(round_HB)) +
  geom_bar() +
  ggtitle("Estimación del Modelo Holm Bonferroni") +
  labs(x="Score", y="Prediccion del modelo")
charts_HB
```




```
charts_DMR <- ggplot(predicciones, aes(DMR)) +
  geom_bar() +
  ggtitle("Estimación del Modelo Distributed Multinomial Regression") +
  labs(x="Score", y="Prediccion del modelo")
charts_DMR
```



Podemos observar que en el 78% de los casos el Score estimado con DMR obtuvo los mismos valores que el Score observado:

```
numero <- c("1","2","3","4","5")
numero <- as.data.frame(numero)

dmr <- table(predicciones$Score, predicciones$DMR)
dmr <- as.data.frame.matrix(dmr)

dmr <- cbind(numero, dmr)

row.names(dmr) <- c("1","2","3","4","5")
colnames(dmr)[1] <- c("Observado/Pred","1","2","3","4","5")
```

```
Warning in colnames(dmr)[1] <- c("Observado/Pred", "1", "2", "3", "4", "5"):
número de items para para sustituir no es un múltiplo de la longitud del
reemplazo
```

```
dmr <- as.data.frame(dmr)

### Matriz de confusión:

dmr <- dmr %>%
  kbl(caption = "Matriz de Confusión para DMR") %>%
```

Table 6: Matriz de Confusión para DMR

Observado/Pred	1	2	3	4	5
1	240	10	9	4	103
2	17	79	5	5	70
3	9	8	117	11	130
4	7	6	8	170	386
5	9	6	9	30	2527

Table 7: Matrix de Confusión para FDR

Observado/Pred	1	2	3	4	5
1	16	24	1	325	0
2	1	4	1	166	4
3	0	5	2	260	8
4	0	2	1	532	42
5	0	1	2	2152	426

```
  kable_classic(full_width = F, html_font = "Cambria")
dmr
```

Observamos que solo en 24% de los valores estimados bajo FDR fueron igual al valor del Score observado, siendo la categoría 4 donde peor desempeño tuvo, ya que los clasificó como Score igual a 5 cuando en realidad eran 4.

```
FDR <- table(predicciones$Score, predicciones$round_FDR)
FDR <- as.data.frame.matrix(FDR)

FDR <- cbind(numero,FDR)

colnames(FDR)[1] <- c("Observado/Pred","1","2","3","4","5")
```

```
Warning in colnames(FDR)[1] <- c("Observado/Pred", "1", "2", "3", "4", "5"):
número de items para para sustituir no es un múltiplo de la longitud del
reemplazo
```

```
FDR <- FDR %>%
  kbl(caption = "Matrix de Confusión para FDR") %>%
  kable_classic(full_width = F, html_font = "Cambria")
FDR
```

La predicción del Score utilizando HB solo muestra un 24% de coincidencia con el Score observado, siendo al igual que FDR en el Score 4 su peor desempeño, de igual forma se podría decir que falló en clasificar el Score igual a 2.

```
HB <- table(predicciones$Score, predicciones$round_HB)
HB <- as.data.frame.matrix(HB)

HB <- cbind(numero, HB)

colnames(HB)[1] <- c("Observado/Pred","1","2","3","4","5")
```

Table 8: Matrix de Confusión para HB

Observado/Pred	1	2	3	4	5
1	6	21	0	339	0
2	1	0	0	171	4
3	0	2	1	264	8
4	0	2	0	533	42
5	0	1	0	2159	421

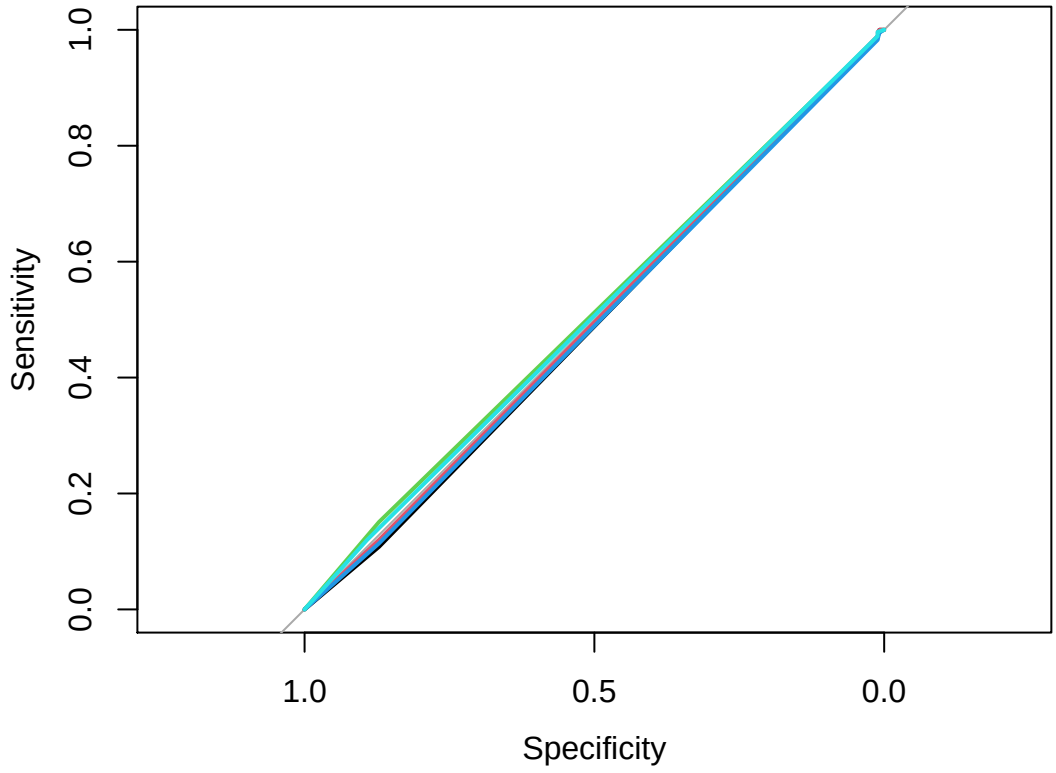
```
Warning in colnames(HB)[1] <- c("Observado/Pred", "1", "2", "3", "4", "5"):
número de items para para sustituir no es un múltiplo de la longitud del
reemplazo
```

```
HB <- HB %>%
  kbl(caption = "Matrix de Confusión para HB") %>%
  kable_classic(full_width = F, html_font = "Cambria")
HB
```

Como se puede observar tanto los modelos False Discovery Rate como Holm Bonferroni predicen un score de 4 dado las variables significativas obtenidas después de la aplicación de ambos modelos. Sin embargo, el modelo Distributed Multinomial Regression (DMR) predice un score de 5.

Con el propósito de entender la sensibilidad y especificad de los modelos obtenemos las curvas ROC:

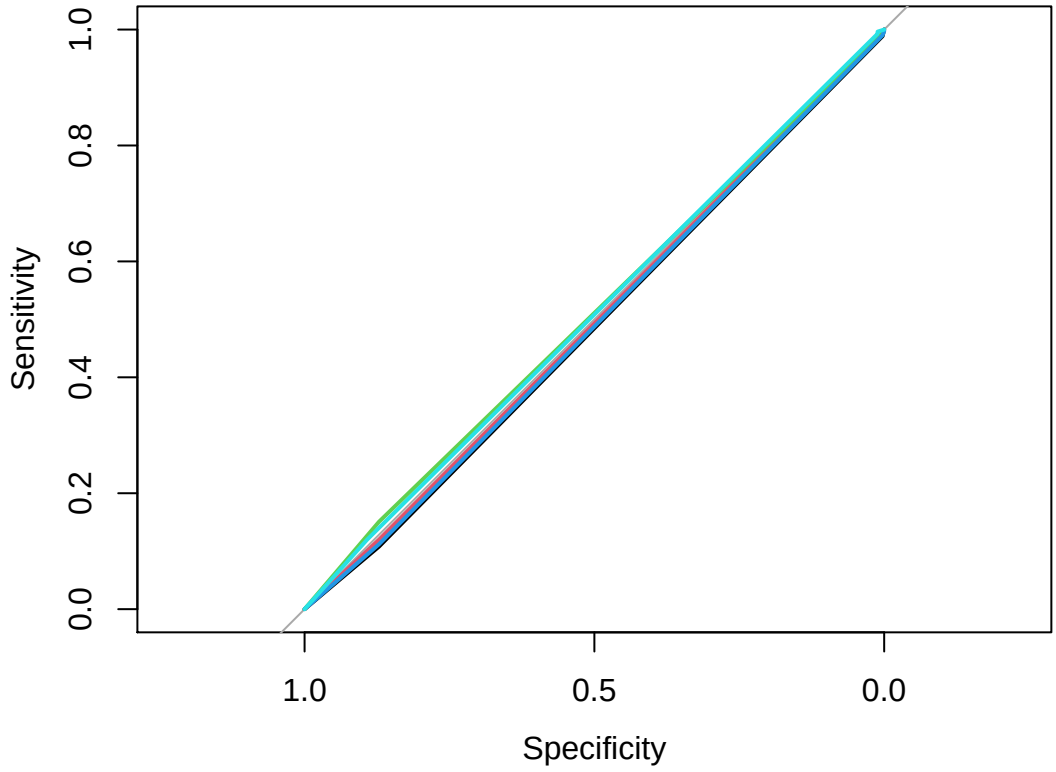
```
### 1) curva ROC del Modelo FDR:

predicciones$round_FDR <- destring(predicciones$round_FDR)

roc <- multiclass.roc(validation$Score, predicciones$round_FDR)

rs <- roc[['rocs']]

plot.roc(rs[[1]])

sapply(2:5,function(i) lines.roc(rs[[i]],col=i))
```



	[,1]	[,2]	[,3]	[,4]
percent	FALSE	FALSE	FALSE	FALSE
sensitivities	Numeric,6	Numeric,6	Numeric,6	Numeric,5
specificities	Numeric,6	Numeric,6	Numeric,6	Numeric,5
thresholds	Numeric,6	Numeric,6	Numeric,6	Numeric,5
direction	"<"	"<"	"<"	"<"
cases	Numeric,275	Numeric,577	Numeric,2581	Numeric,275
controls	Numeric,366	Numeric,366	Numeric,366	Numeric,176
fun.sesp	?	?	?	?
call	Expression	Expression	Expression	Expression
original.predictor	Numeric,3975	Numeric,3975	Numeric,3975	Numeric,3975
original.response	Integer,3975	Integer,3975	Integer,3975	Integer,3975
predictor	Numeric,641	Numeric,943	Numeric,2947	Numeric,451
response	Integer,641	Integer,943	Integer,2947	Integer,451
levels	Character,2	Character,2	Character,2	Character,2

```
### 2) curva ROC del Modelo HB:

predicciones$round_HB <- destring(predicciones$round_HB)

roc <- multiclass.roc(validation$Score, predicciones$round_HB)

rs <- roc[['rocs']]

plot.roc(rs[[1]])
```

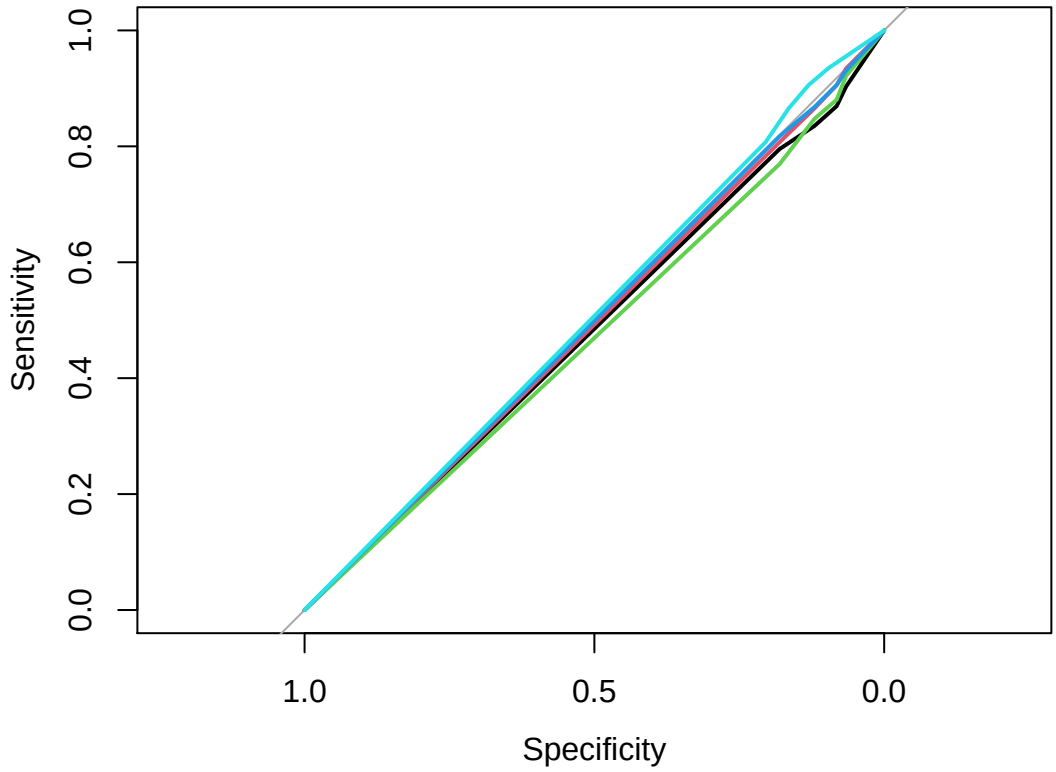
```
sapply(2:5,function(i) lines.roc(rs[[i]],col=i))
```



	[,1]	[,2]	[,3]	[,4]
percent	FALSE	FALSE	FALSE	FALSE
sensitivities	Numeric,5	Numeric,4	Numeric,5	Numeric,5
specificities	Numeric,5	Numeric,4	Numeric,5	Numeric,5
thresholds	Numeric,5	Numeric,4	Numeric,5	Numeric,5
direction	"<"	"<"	"<"	"<"
cases	Numeric,275	Numeric,577	Numeric,2581	Numeric,275
controls	Numeric,366	Numeric,366	Numeric,366	Numeric,176
fun.sesp	?	?	?	?
call	Expression	Expression	Expression	Expression
original.predictor	Numeric,3975	Numeric,3975	Numeric,3975	Numeric,3975
original.response	Integer,3975	Integer,3975	Integer,3975	Integer,3975
predictor	Numeric,641	Numeric,943	Numeric,2947	Numeric,451
response	Integer,641	Integer,943	Integer,2947	Integer,451
levels	Character,2	Character,2	Character,2	Character,2

```
### 3) curva ROC del Modelo DMR:

predicciones$DMR <- destring(predicciones$DMR)

roc <- multiclass.roc(validation$Score, predicciones$DMR)
```

```r
rs <- roc[['rocs']]

plot.roc(rs[[1]])

sapply(2:5,function(i) lines.roc(rs[[i]],col=i))
```



	[,1]	[,2]	[,3]	[,4]
percent	FALSE	FALSE	FALSE	FALSE
sensitivities	Numeric,6	Numeric,6	Numeric,6	Numeric,6
specificities	Numeric,6	Numeric,6	Numeric,6	Numeric,6
thresholds	Numeric,6	Numeric,6	Numeric,6	Numeric,6
direction	"<"	"<"	"<"	"<"
cases	Numeric,275	Numeric,577	Numeric,2581	Numeric,275
controls	Numeric,366	Numeric,366	Numeric,366	Numeric,176
fun.sesp	?	?	?	?
call	Expression	Expression	Expression	Expression
original.predictor	Numeric,3975	Numeric,3975	Numeric,3975	Numeric,3975
original.response	Integer,3975	Integer,3975	Integer,3975	Integer,3975
predictor	Numeric,641	Numeric,943	Numeric,2947	Numeric,451
response	Integer,641	Integer,943	Integer,2947	Integer,451
levels	Character,2	Character,2	Character,2	Character,2

Para obtener una conclusión más precisa se elabora la tabla AUC ROC:

Table 9: AUC ROC por modelo

Modelo	AUC
FDR	0.5496383
GB	0.5444067
DMR	0.7139142

```
auc <- matrix(c(auc(predicciones$Score, predicciones$round_FDR), auc(predicciones$Score, predicciones$r
```

```
Warning in roc.default(response, predictor, auc = TRUE, ...): 'response'
has more than two levels. Consider setting 'levels' explicitly or using
'multiclass.roc' instead

Warning in roc.default(response, predictor, auc = TRUE, ...): 'response'
has more than two levels. Consider setting 'levels' explicitly or using
'multiclass.roc' instead

Warning in roc.default(response, predictor, auc = TRUE, ...): 'response'
has more than two levels. Consider setting 'levels' explicitly or using
'multiclass.roc' instead
```

```
auc <- as.data.frame(auc)

auc <- rename(auc, AUC = V1)

auc$Modelo <- c("FDR", "GB", "DMR")

auc <- subset(auc, select = c(Modelo, AUC))

auc <- auc %>%
  kbl(caption = "AUC ROC por modelo") %>%
  kable_classic(full_width = F, html_font = "Cambria")

auc
```

Por lo tanto, en las gráficas ROC se observa que, si bien los tres modelos son mejores que el azar, i) esto no es así para todas las clases del Score y ii) DMR es el mejor modelo en términos generales, lo cual se confirma en la estimación del área bajo la curva ROC (AUC ROC) con una precisión del 71.4%. Si observamos las gráficas ROC, el modelo DMR resulta más sensible que específico para prácticamente todas las clases del Score: es decir, es mejor prediciendo valores positivos de cada clase, que encontrando valores negativos. En el caso de los modelos FDR y HB, son modelos ligeramente más sensibles que específicos.

4. Propuestas y conclusiones

A manera de resumen, se utilizó una base de datos de Amazon Reviews que tenía comentarios y calificaciones para los productos vendidos por Amazon. Llevamos a cabo un análisis sobre las calificaciones de estos productos y se identificó que el mayor número de ventas se refiere a productos de Grocery con 9,136 ventas; el resto de los grupos, están por debajo de 2,000 productos. sobre sus calificaciones, los productos de Health and Beauty y Sports reciben el mayor porcentaje de 5's en el score (79% y 72%, respectivamente). Sin

embargo, el grupo de Sports es, también, el que recibe mayor porcentaje de 1's solo por debajo de Lawn & Patio (19% y 24%, respectivamente).

Sobre los conceptos más mencionados en los reviews de estos productos, destacan: great, good, love, coffee y best. Por lo tanto, es posible identificar que, en general, una gran parte de los comentarios tienen un sentido positivo. Además, algunos de los productos o grupos más mencionados son el café, té, comida y perros. Los conceptos que más reflejan emociones positivas en el agregado de los consumidores de Amazon son good, great y love. Por el contrario, los conceptos que más reflejan emociones negativas son bad, horrible y nasty.

A partir de las emociones vinculadas con cada palabra, se les asignó un valor entre -5 y 5 (de negativo a positivo) para crear un score de sentimientos el cual se utilizó para predecir el score numérico original, en conjunto con dummies por palabra de los reviews. Para realizar dicha predicción, se dividió a la base en dos: una de entrenamiento (70%) donde se construyó el modelo y una de validación (30%) donde se probaron las predicciones.

Se probaron regresiones con métodos de FDR y HB para reducir el número de variables, pero se concluyó que un modelo Distribute Multinomial Regression (DMR) es la mejor aproximación para estimar el Score numérico a partir del número de comentarios, el mes de la compra, la categoría y grupo del producto, así como dummies por palabra en el comentario y un score numérico de sentimientos utilizando el diccionadio *afinn*. El modelo presentó un área bajo la curva ROC del 71.3% con mayor sensibilidad que especificidad para las cinco clases del Score.

A continuación se detallan algunas sugerencias para Amazon:

- El score de sentimientos no sustituye al score numérico; principalmente, porque no todos los comentarios contienen palabras que se pueden vincular con emociones y, por lo tanto, en este score de sentimientos, se crearon más del 50
- Sin embargo, este score de sentimientos, en conjunto con las dummies por palabras, son una buena aproximación para estimar el score numérico. Si bien no es tan preciso, es factible que arroje información "diferente" que podría no estar capturada en el score numérico y que podría ser relevante para focalizar campañas o acciones.
- Antes de esperar a que los clientes pongan el Score de los productos, se podría predecir la probabilidad de cada cliente de poner un Score negativo y, por lo tanto, minimizar esta probabilidad. Por ejemplo, realizarles llamadas o enviar mensajes de seguimiento a estos clientes u ofrecerles alguna garantía especial. Para ello, se podrían diseñar experimentos interesantes de bajo costo que pudieran reducir el número de Scores bajos para determinados grupos de productos.
- Se propone al cliente trabajar en el reposicionamiento de aquellos con productos con menos reviews, ya que estos tienen una menor liquidez (rotación), por lo que podría llevar a que la empresa tenga costos hundidos de inventario. Además, un mayor número de reviews podría ayudar a predecir mejor el comportamiento de los clientes sobre su satisfacción con los productos (y, por lo tanto, focalizar mejor acciones frente a ellos).
- El cliente debería continuar con el uso del sentiment analysis, ya que parece ser un buen predictor del score que puede recibir un producto lo cual les permitiría implementar métricas respecto a la calidad de un bien y en posterior análisis ligar la calidad de un bien con base en el sentiment analysis y la probabilidad de vender más un producto.
- De igual forma, esta metodología permite determinar el nivel de satisfacción y tomar decisiones para mejorar los servicios y productos. Con la codificación del texto libre se determinaron aquellos temas en los cuales actuar de manera prioritaria utilizando un indicador para determinar el potencial impacto de cada producto. Gracias a este cálculo, se podrían introducir mejoras en los procesos que permitan mejorar la tasa de satisfacción de los clientes.