Proyecto final - Amazon reviews

Montserrat Aldave Alejandro Fajardo Carlos Tabares Jose Pablo García
Rafael Gurgel

Mayo 2021

1 - Introducción

En este proyecto, estudiaremos los reviews de Amazon y buscaremos pronosticar la calificación de los productos con base en los reviews que dejan los clientes. Pondremos en práctica técnicas de Text mining para la realización de word clouds y la elaboración de un índice numérico a partir del análisis de sentimiento de los reviews. Consecuentemente, utilizaremos un modelo Multinomial Lasso y un Multinomial Distributed Regression para predecir con base en el texto de los reviews la calificación de cada producto.

2- Exploración de la base de datos

Exploratory data analysis

1) Queremos encontrar la presencia de NA's en nuestra base de datos y posteriormente decidiremos que hacer con los valores omitidos.

Encontramos 3,198 observaciones faltantes en nuestra base de datos. No obstante, estas representan un porcentaje muy pequeño del total de observaciones en la base (119,871). Los missing values sólo se encuentran en las columnas de Grupo y Categoría del producto ('Prod_group' y 'Prod_category'), y representan el 11.6% y 12.4% de estas variables, respectivamente. Por ello, asignaremos estos valores a una nueva categoría que llamaremos "others".

2) ¿A qué nivel está la base de datos?

La base está a nivel cliente y producto. Tenemos 13,319 ventas, 5893 clientes y 613 productos diferentes.

3) Vamos a convertir la base de datos a one-token-per-row para poder trabajarla

Descripción de la Base de datos

La base de datos esta a nivel cliente (UserId). Contamos con el número de reviews hecho por cada cliente, el grupo y categoría de producto que ha sido calificado, el Score atribuído y el texto del producto. La base de datos cuenta con 613 productos únicos, que son agrupados en 142 categorías y 10 grupos (Productos de bebé; cocina; belleza; jardín; artículos de escritorio e industria (BISS); comestibles; productos para mascotas; belleza y salud; deportes; y otros). La mayoría de los productos que compran los clientes pertenecen al grupo de 'Grocery' (artículos del súper), seguido del grupo 'Pet' (artículos para mascotas).

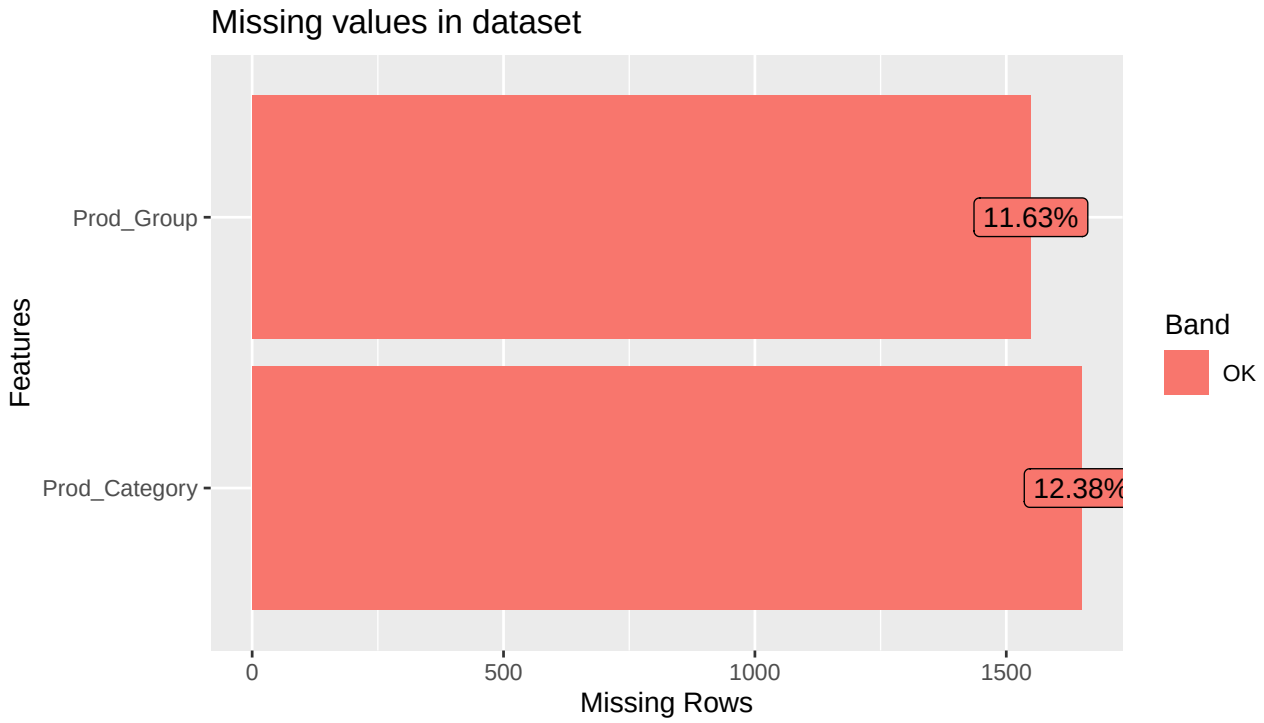
Hay un total de 13.319 reviews hechas por 5893 clientes. En promedio, cada cliente hizo 2,26 reviews y el máximo de reviews por un único cliente observado fue 28. Observando la gráfica de frequencia de cantidad de reviews, observamos que el 27% de los clientes hizo sólo un reviews, esa es la frecuencia más comun. También hay un número razonable de clientes que hicieron apenas 2 reviews (13,2%) o 4 reviews (14,3%). En general, el 66,4% de los clientes hicieron hasta 5 reviews, y apenas el 17,5% de los clientes hicieron 10 o más reviews.

Llama la atención la distribución de los scores, que se parece al símbolo de Nike. El 9% de los reviews dan calificación 1 (la peor) y el 4,4%, 2; las calificaciones neutras son sólo el 6,7% ; y las calificaciones positivas son el del 14,8%, para el nivel 4, y de sorpreendentes 64,8% para el nivel 5 (la calificación máxima). Podemos soponer que las personas suelen comentar más los productos que les gustan mucho, menos los productos que no les gustan y no suelen comentar los productos que les despiertan una reacción neutral.

Observando la distribución de las calificaciones por grupo de productos observamos un patrón semenjante, con una cantidad mucho mayor de calificaciones 4 y 5, seguidas de 1, que calificaciones de nivel 2 y 3. Es curioso que los productos para bebé casi no tienen reviews negativos. Eso puede indicar un mayor estándar de calidad o nada más reflejar el hecho de que los bebés no escriben reviews (el patrón semejante observado en los productos para mascotas da fuerza a esa segunda hipotesis).

En ese contexto, nos interesa evaluar cuál es la reacción (sentimiento) que despertó cada producto con base en los textos de los reviews y cuáles términos son los mejor predictores de la calificación atribuída a los productos.

```
db <- db %>% mutate_all(na_if,"")    # ponemos NA donde hay espacios vac????os

plot_missing(db, missing_only = TRUE, title = "Missing values in dataset",
             theme_config = list(legend.position = c("right")))
```




```
missings <- map_dbl(db%>%select_all(),~100*sum(is.na(.))/nrow(db))
missings[missings>0]
```

```
Prod_Category    Prod_Group
     12.38081      11.63000
```

```
# Como tenemos un 11% de prod_group y 12% prod_category sin categor????a vamos a meterlos en una catego

#convert all NA's to "Others"
db$Prod_Group[is.na(db$Prod_Group)] = "Others"
db$Prod_Category[is.na(db$Prod_Category)] = "Others"
```

```r
# Con esto corremos el código de arriba y verificamos que ya no hay valores vacios ni NAs

db <- db%>%
  mutate(Prod_Category = factor(Prod_Category),
         Prod_Group = factor(Prod_Group))


# Veamos como se distribuyen las categor????as y cuantas son
users <- unique(db$UserId)
categories <- unique(db$Prod_Category)
products <- unique(db$Prod_Group)

#Revisemos cuantos clientes duplicados hay
sum(duplicated(db$UserId))
```

```
[1] 7426
```

```r
# Veamos un histograma de las calificaciones por tipo de productos, que aparentemente sólo son 10 nivel
paste("No. grupos de productos: ", length(products))
```

```
[1] "No. grupos de productos:  10"
```

```r
paste("No de categorias: ", length(categories))
```

```
[1] "No de categorias:  142"
```

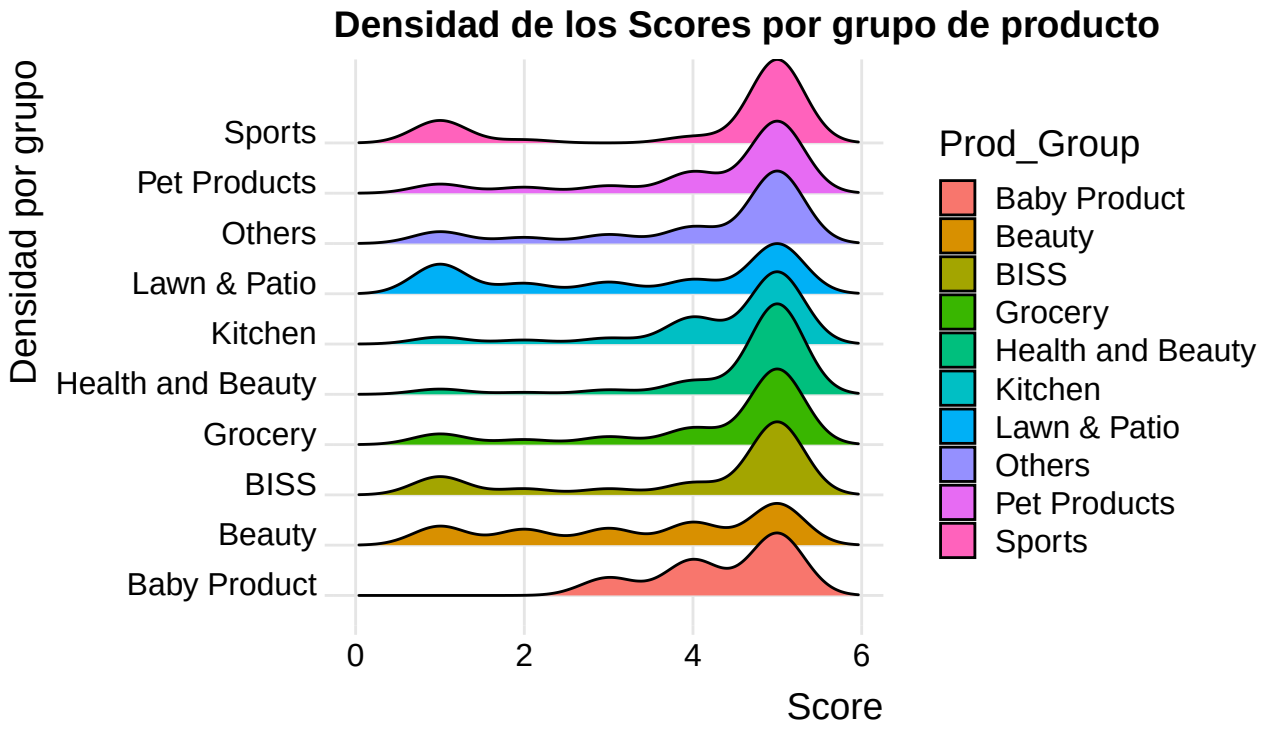
```r
paste("No de clientes: ", length(users))
```

```
[1] "No de clientes:  5893"
```

```r
paste("No. productos: ", length(unique(db$ProductId)))
```

```
[1] "No. productos:  613"
```

```r
# Histograma por producto

histo_prod <- db %>%
  ggplot(aes(x = Score, fill = Prod_Group, y = Prod_Group)) +
  geom_density_ridges()+
  labs(y = "Densidad por grupo", x="Score", title = "Densidad de los Scores por grupo de producto")

histo_prod+ theme_joy()
```




```
# Vemos que la base no se encuentra a nivel usuario, sino a nivel compra x usuario
str(db$UserId)
```

```
 chr [1:13319] "#oc-R163CP16SRRI50" "#oc-R163CP16SRRI50" ...
```

```
str(unique(db$UserId))
```

```
 chr [1:5893] "#oc-R163CP16SRRI50" "#oc-R19QDOY2PXS15" "#oc-R1B9W981WGB5D0" ...
```

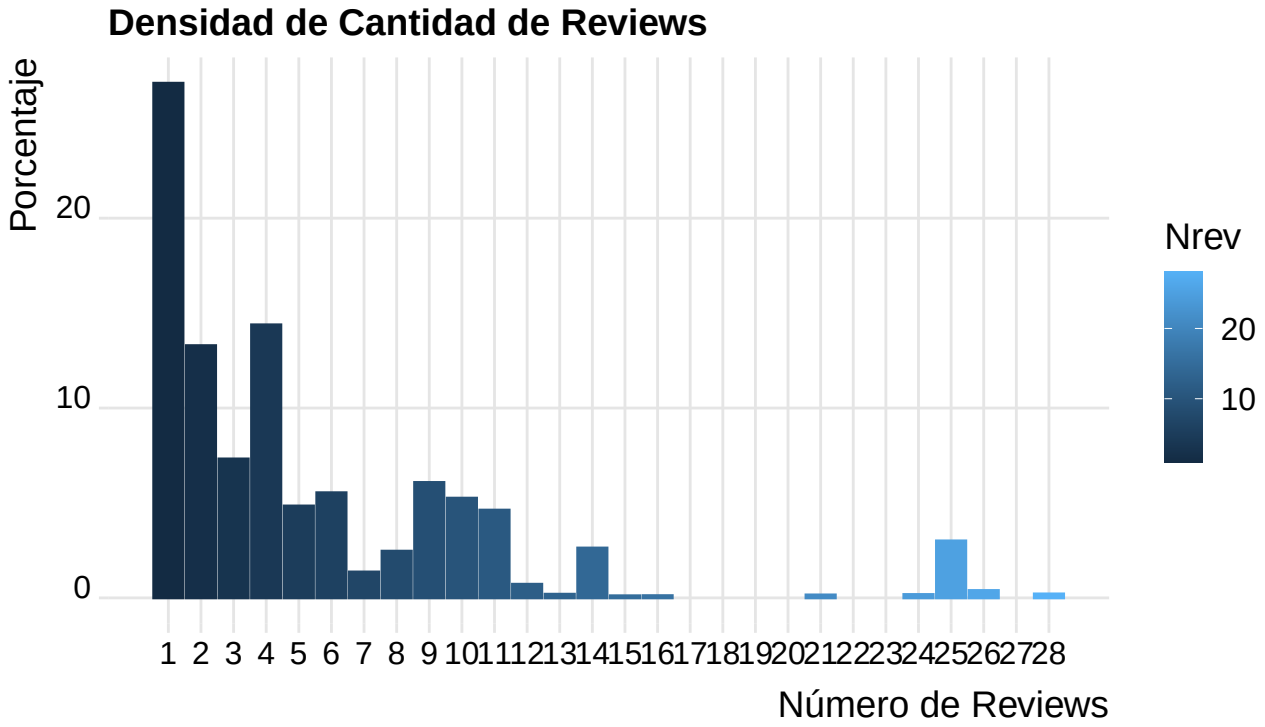
```
str(db$ProductId)
```

```
 chr [1:13319] "B005HG9ESG" "B005HG9ERW" "B005HG9ET0" "B006Q820X0" ...
```

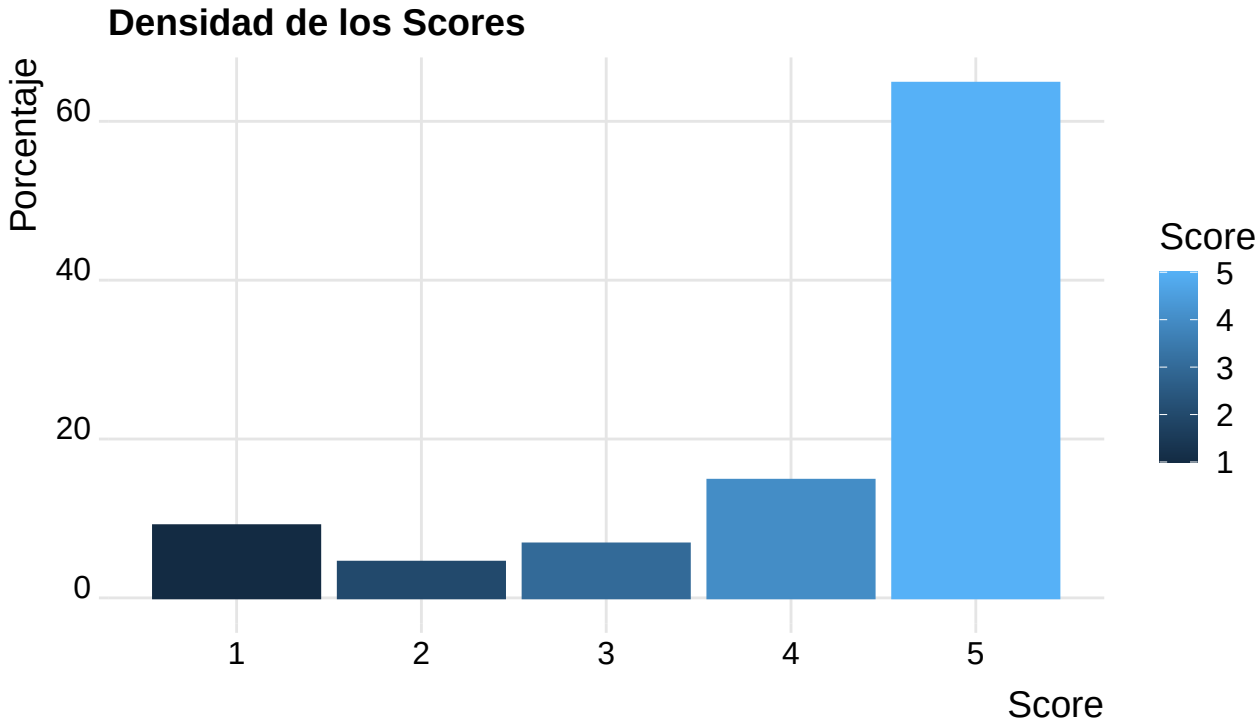
```
str(unique(db$ProductId))
```

```
 chr [1:613] "B005HG9ESG" "B005HG9ERW" "B005HG9ET0" "B006Q820X0" ...
```

```
# Frequencia de numeros de reviews
db_cut1 = db %>%
  group_by(Nrev) %>%
  summarise (n = n()) %>%
  mutate(freq = n / sum(n))

# Densidad de número de reviews
histo_Nrev <-
  ggplot(data = db_cut1, aes(x = Nrev, y = freq*100, color = Nrev, fill = Nrev)) +
  geom_bar(stat="identity") +
  labs(y = "Porcentaje", x="Número de Reviews", title = "Densidad de Cantidad de Reviews") +
  scale_x_continuous(breaks = 1:28)

histo_Nrev+theme_joy()
```




```
# Densidad de Scores
db_cut = db %>%
  group_by(Score) %>%
  summarise (n = n()) %>%
  mutate(freq = n / sum(n))

histo_prod2 <-
  ggplot(data = db_cut, aes(x = Score, y = freq*100, color = Score, fill = Score)) +
  geom_bar(stat="identity") +
  labs(y = "Porcentaje", x="Score", title = "Densidad de los Scores")

histo_prod2+theme_joy()
```




```
# Frecuencia de grupos
table(db$Prod_Group)
```

```
     Baby Product            Beauty              BISS           Grocery
               26               250                19              8725
Health and Beauty           Kitchen      Lawn & Patio            Others
              905               117                90              1549
     Pet Products            Sports
             1602                36
```

```
# Frecuencia de categorias productos
table(db$Prod_Category)
```

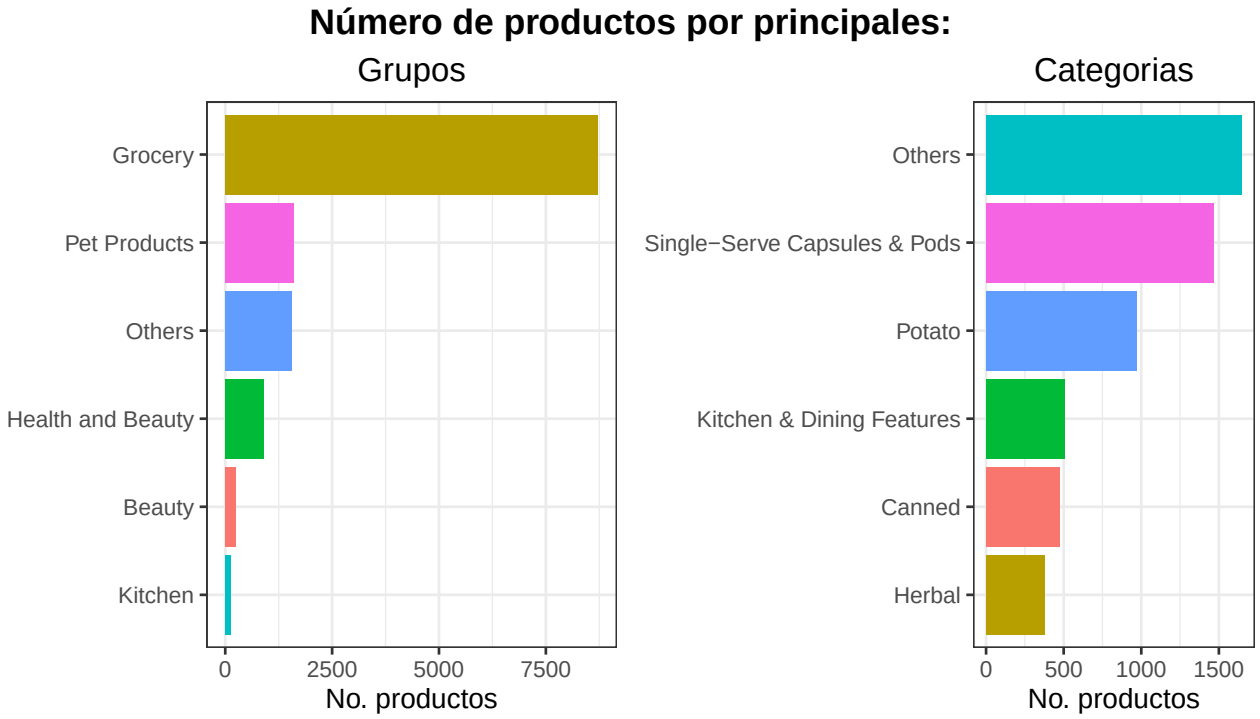
```
                 Almond           Almond Butter
                     17                      12
               Almonds              Asian Dishes
                    132                      31
          Baking Mixes                 Balsamic
                     48                      12
                  Bars              Bars & Snacks
                     12                      17
                 Black                     Bones
                     43                      81
               Breads          Candy & Chocolate
                     22                      11
               Canned                 Cat Doors
                    475                      11
            Categories                    Catnip
                     44                      23
               Cereals                 Chia Seeds
```

41	23
Chili Powder	Ciders
11	48
Cinnamon	Cleansers
15	13
Coarse Salt	Cocoa
22	115
Coconut	Coconut Flakes
113	26
Coconut Water	Coffee Pod Holders
154	35
Cold Cereals	Computers Features
48	11
Conditioner	Cookies
18	15
Cooking & Baking	Cooking Oils, Vinegars & Sprays
24	152
Creams	Curry Sauce
48	76
Daily Shampoo	Decorative Boxes
23	15
Dietary Fibers	Dried Fruits
30	13
Dried Seaweed & Nori	Drinks
17	71
Dry	Electronics Features
318	67
Energy & Nutritional	Energy Drinks
195	45
Extracts & Flavoring	Fettuccine
12	28
Flavor Syrups	Flavored Salt
143	20
Flours & Meals	Food Bars
37	17
Food Strainers	Fruit Juice
19	25
Fruity Candy	Garbanzo
15	38
Ghee	Ginger Candy
46	15
Granola	Green
100	76
Green Food Combinations	Ground Coffee
71	129
Gummy Candy	Hair & Scalp Treatments
23	95
Healthcare	Hemp
22	31
Hemp Seeds	Herbal
30	374
Hip & Joint Care	Home & Kitchen Features
17	83
Honey	House Plants

39	11
Iced Tea	Instant Coffee
92	200
Jelly Beans	Jerky & Dried Meats
82	45
Kitchen & Dining Features	Leaveners & Yeasts
503	12
Magnesium	Maple Syrup
240	17
Meals	Meat, Poultry & Seafood
18	24
Mice & Animal Toys	Multiple Vitamin-Mineral Supplements
18	240
Nut & Seed Butters	Nutrition Bars
39	110
Oatmeal	Oolong
55	30
Orchids	Others
13	1649
Outdoor Recreation	Palm Sugar
21	16
Pancakes & Waffles	Pasta & Noodles
20	24
Peanut Butter	Pie & Pastry Fillings
295	12
Pill Cases	Potato
22	969
Powdered	Powdered Cheese
14	14
Products	Protein Bars
356	195
Quinoa	Reusable Filters
15	16
Roasted Coffee Beans	Rock Salt
11	153
Ropes	Salmon
78	11
Seaweed Snacks	Shampoo
16	12
Shave & Hair Removal	Single-Serve Brewers
13	39
Single-Serve Capsules & Pods	Snacks
1467	142
Soda Maker Parts & Accessories	Soft Drinks
18	60
Spaghetti	Sparkling Juice
12	14
Sports & Fitness Features	Sports Nutrition
12	220
Squeak Toys	Stevia
24	26
Sticks & Twists	Sugar Substitutes
19	58
Sugars	Super-Automatic Espresso Machines

```
                              16                                12
                       Superfoods                      Supplements
                              121                               15
                     Tea Samplers                      Thickeners
                              307                               15
                         Tortilla                            Traps
                              13                                35
                           Treats                      Treats & Toys
                              34                               223
                        Tuna Fish                         Unpopped
                              17                                90
                     Vanilla Beans                     Vegetables
                              19                                12
      Vitamins & Dietary Supplements                         Water
                             158                               114
                      Weight Loss                           Wild
                              16                                12
```

```
# Grupos de producto
pxg <-   db %>%   group_by(Prod_Group) %>%
     tally() %>%
     arrange(desc(n)) %>%
  head() %>%
     ggplot( aes(reorder(Prod_Group,n),n, fill= Prod_Group))+
      geom_bar(stat = "identity") + coord_flip() +
      theme_bw()  +
      theme(legend.position="none") +
      labs( title = "Grupos",
       y = "No. productos",
        x = NULL ) +
      theme(plot.title = element_text(hjust = 0.5))

# Categor????as más populares
 pxc <-  db %>%   group_by(Prod_Category) %>%
     tally() %>%
     arrange(desc(n)) %>%
    head() %>%
     ggplot( aes(reorder(Prod_Category,n),n, fill= Prod_Category))+
      geom_bar(stat = "identity") + coord_flip() +
      theme_bw()  +
      theme(legend.position="none") +
      labs( title = "Categorias",
       y = "No. productos",
        x = NULL ) +
      theme(plot.title = element_text(hjust = 0.5))

 grafs <- ggarrange(pxg, pxc, nrow = 1)

annotate_figure(grafs,
                top = text_grob("Número de productos por principales:", color = "black",
                                face = "bold", size = 14) )
```




```
rm(pxg, pxc)
```

3 - Análisis de Texto

Text mining

Hicimos la transformación del texto de los reviews a columnas de palabras y quitamos las palabras conectoras ("and", "by", "of" etc) para evaluar cuáles son los términos más hablados en los reviews. Generamos el wordcloud general y por categoría de productos.

En general, los términos más frecuentes están relacionados a comidas y bebidas ("coffee", "tea", "food", "snack", "chips"), a adjetivos y verbos sensoriales ("taste", "yummy", "tasty", "delicious", "flavor"), y sentimientos o adjetivos positivos ("love", "favorite", "amazing", "awesome", "nice"). Los términos de connotación negativa ("nasty", "horrible", "disappointed") con claramente minoritarios. Eso va en línea a lo observado en el análisis de la base de datos, donde la mayor parte de los reviews es de 4 o 5 estrellas.

El wordcloud por categoría presenta un patrón semejante: palabras describiendo el producto ("popcorn", "product", "plants") y adjetivos y verbos de connotación positiva ("brilliant", "nice", "easier"). Sin embargo, los clientes parecen comentar más en productos comestibles y para mascotas y menos en productos de otras categorías.

Comparando el resultado para los 100 primeros reviews de los tres diccionarios, los resultados son semejantes. Así, optamos por utilizar el diccionario AFFIN, que tiene mayor variación de scores, en las estimaciones de las siguientes secciones.

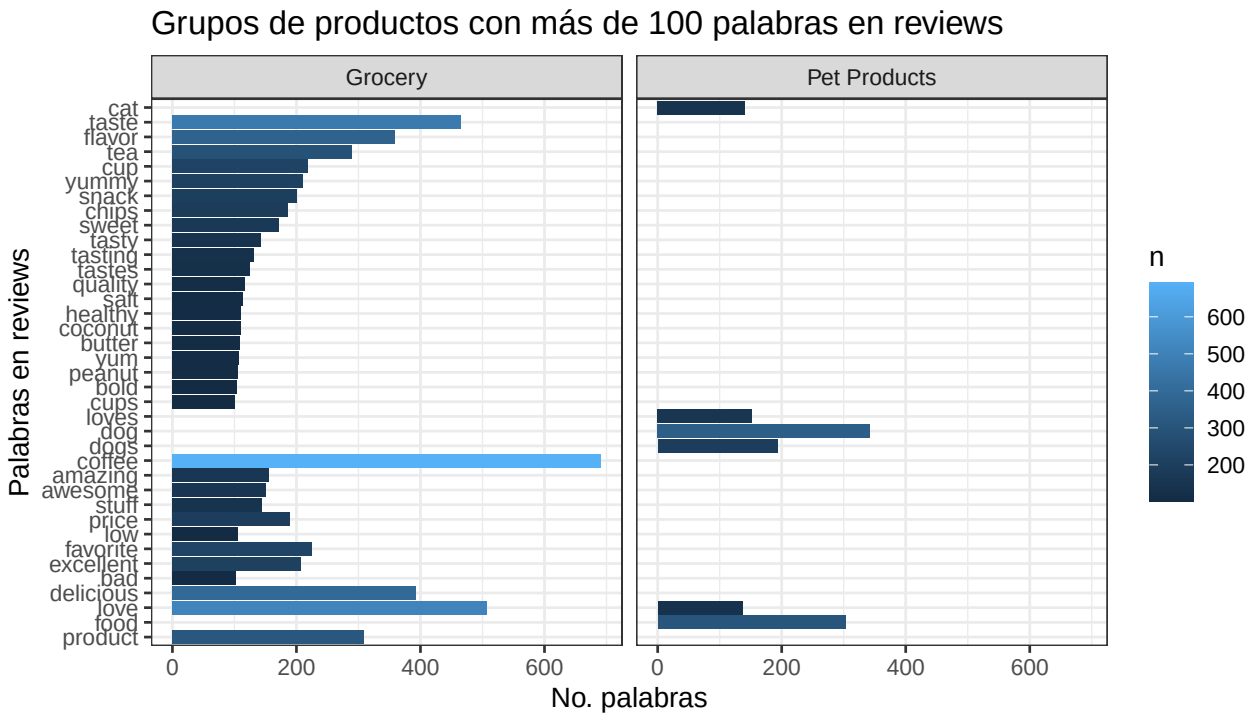
Análisis de sentimiento

Con base en el text mining, procedemos al análisis de sentimiento de los reviews con base en tres distintos diccionarios de sentimientos: AFFIN, BING y NCR. El diccionario AFFIN atribuye un score de sentimiento que va de -5 (muy negativo) a +5 (muy positivo) a cada palabra no conectora. Ya los diccionarios BING y NCR funcionan de manera binaria, una palabra es considerada positiva o negativa. El NCR, además, atribuye una calificación de sentimiento a las palabras ("anger", "joy", "trust", "surprise", "sadness" etc).

1) ¿Qué temas son los que más se hablan en cada review?

Observamos que los grupos de productos que tienen mayor número de palabras en reviews son 'Grocery' y 'Pet products'. De estos, los artículos de súper ('Grocery') son los que más reviews tienen en nuestra base. Además, elaboramos wordclouds para la base completa y cada uno de los grupos de productos.

```
# Ahora transformemos nuestra base de datos para poder utilizar los algoritmos de tidytext
# Como no nos interesa la venta a nivel producto, decidimos mantener la base de datos a nivel usuario
# Es muy importante darnos cuenta que estamos trabajando con un panel ahora, pero de palabras y calific

db2 <- db %>%
  mutate(Summary= as.character(Summary)) %>%
  unnest_tokens(word, Summary)

# Cargamos las palabras que sirven para conectar
data("stop_words")

# las eliminamos con un antijoin y por la palabra word
db2 <- db2 %>%
  anti_join(stop_words)

# Ahora si podemos hacer cosas interesantes para ver que palabras se repiten más
# Como los grupos de productos son 10 nos centraremos en esa var y no por categorias que llegan hasta 1
# Decidimos ordenar las palabras por producto y que nos las cuente
plot1 <- db2 %>%
  group_by(Prod_Group) %>%
    count(word, sort = TRUE) %>%
    mutate(word = reorder(word, n)) %>%
    filter(n>100) %>%
    ggplot(aes(x=n, y=word, fill=n))+
    geom_col()+
    facet_wrap(~Prod_Group)+
   labs( title = "Grupos de productos con más de 100 palabras en reviews",
       y = "Palabras en reviews",
        x = "No. palabras") +
      theme(plot.title = element_text(hjust = 0.5)) +
      theme_bw()

# Visualicemos con diferentes arranques y por grupos
plot1
```




```
# Hagamos una nube de datos de toda la base
db2 %>%
  count(word) %>%
  with(wordcloud(word, n, max.words = 110,
                 random.order=FALSE, rot.per=0.35, min.freq = 1,
                                  colors=brewer.pal(20, "Paired")))
```

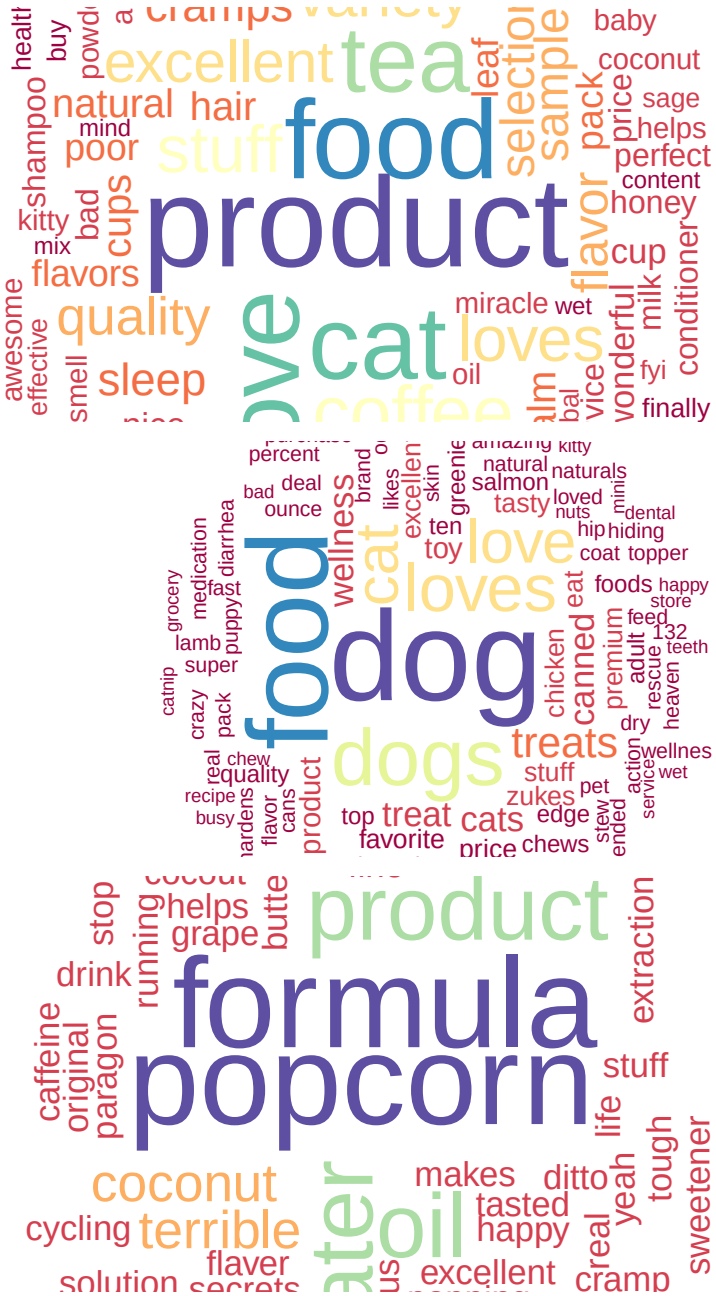
```r
# Un wordcloud por tipos de productos
db3<- db2 %>% split(.$Prod_Group)      #se hace una lista(vector) con elementos, ya se puede iterar.

map(db3,  ~wordcloud(words = .$word,
                                  max.words=100, random.order=FALSE, rot.per=0.35, min.freq = 1,
                                  colors=brewer.pal(11, "Spectral")))
```


coffee
love
taste
delicious
product
flavor
yummy
tastes
bold
senseo
service
kettle
blend
sugar
water
snack
salt
favorite
tea
healthy
tasting
strong
chips
horrible
cup
excellent
chocolate
potato
tasty
cups
easy
flavors
time
bars
super
energy
baby
popchips
pretty
morning
texture
roast
instant
stuff
hot
oil
buy
bar
dark
free
spicy
green
beans
milk
artificial
light
review
fresh

product
coconut
oil
love
coffee
stuff
formula
excellent
calm
helps
anxiety
delicious
hair
sleep
cramps
quality
taste
highly
symptoms
healthy
tasting
smells
health
softener
caffeine
calcium
recommendation
deal
wow
leaf
low
tastes
stevia
viper
pressure
awesome
wonders
virgin
chia
eye
terrible
recommended
restless
drink
liquid
yummy

coffee
love
keurig
sturdy
supposed
cups
drink
smart
automatic
nice
maker
flavor
cup
officepro
nasty
vienna
cola
single
money
bigger
sodastream
brings
home
star
tight
saeco
red
basic
super
slide
unit
item
hold
ease
fantastic
worth
stuff
excellent
redbull
pods
set

plants
service
buy
squirrels
purchase
carnivorous
arrived
healthy
rabbits
spreader
created
sprayer
meat
difficult
design
care
dollar
love
persistence
store
summer
bottles
expected
bad
time
worst
stars
pet
fun
cool
chemical
horsesh
scooter
days
ago
day
dangerous
bug
dud
alive
dogs
godsend
delivery
moles
spray
mice
neat
critter
hungry
seller
fast
cellent
ridder
cats
barn
kill
shipping
piece
havahart
joels
living
west
easy
highly
keeping
limited
call
happy
recommend



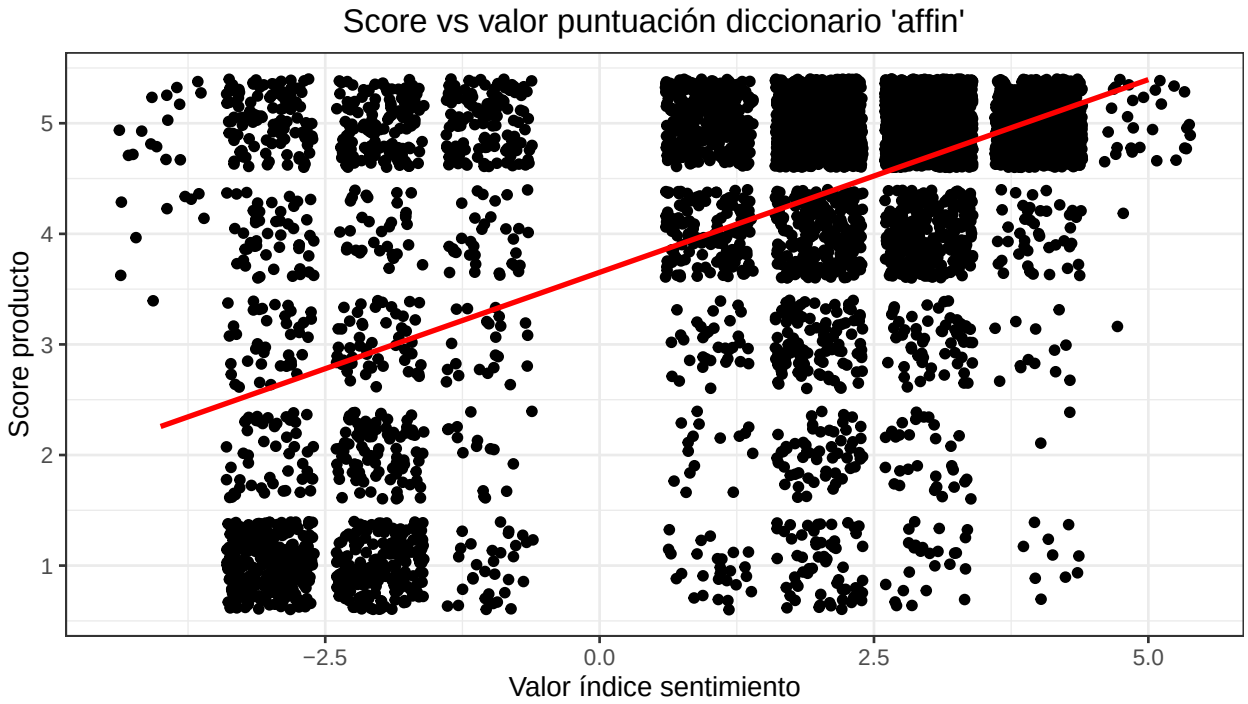

```
$`Baby Product`
NULL

$Beauty
NULL

$BISS
NULL

$Grocery
NULL

$`Health and Beauty`
```

```
NULL

$Kitchen
NULL

$`Lawn & Patio`
NULL

$Others
NULL

$`Pet Products`
NULL

$Sports
NULL
```

2) Índices de sentimientos

Utilizamos los diferentes diccionarios que nos brinda la libreria (textdata) para incoporarlos a nuestra base y obtener un score para cada palabra que aparece en los reviews.

```
library(textdata)

db2 <- db %>%
  mutate(Summary= as.character(Summary)) %>%
  unnest_tokens(word, Summary)

# Cargamos las palabras que sirven para conectar
data("stop_words")

# las eliminamos con un antijoin y por la palabra word
db2 <- db2 %>%
  anti_join(stop_words)

# con el innerjoin permitimos que se junten los dataframes
dbaffin <- db2 %>%
  inner_join(get_sentiments("afinn"))

scatt_affin <- dbaffin %>%
  ggplot(aes(x= value, y = Score))+
  geom_jitter()+
  geom_smooth(method=lm , color="red", se=FALSE) +
  theme_bw()  +
      labs( title = "Score vs valor puntuación diccionario 'affin'",
       y = "Score producto",
        x = "Valor índice sentimiento" ) +
      theme(plot.title = element_text(hjust = 0.5))

scatt_affin
```




```
###
dbbing <- db2 %>%
  inner_join(get_sentiments("bing"))

kable ( a <- as.data.frame((summary(as.factor(dbbing$sentiment)))) %>%
          rename(wrds = `(summary(as.factor(dbbing$sentiment)))`) %>%
         arrange(desc(wrds)) %>%
          mutate(prop = wrds / sum(wrds) * 100) ,
        caption = "Sentimiento con libreria 'bing'",
        col.names =  c("No. palabras", "Prop. aparición"),
                       digits = 2)
```

Table 1: Sentimiento con libreria 'bing'

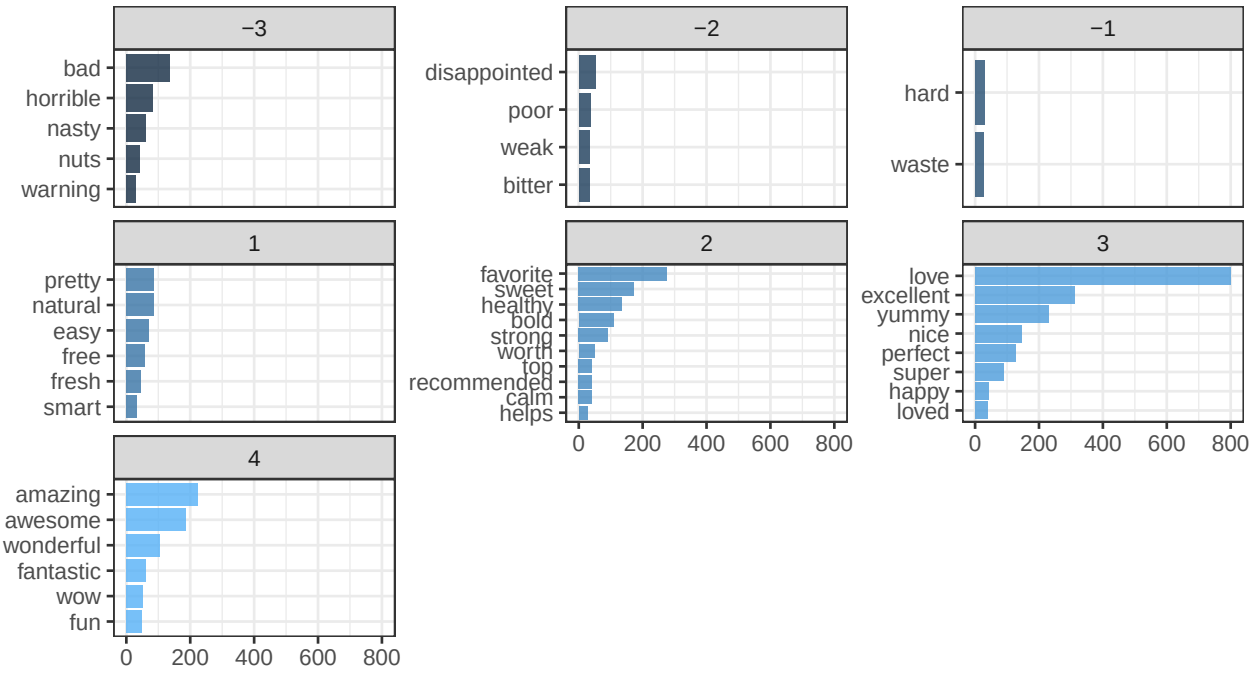
	No. palabras	Prop. aparición
positive	5596	73.3
negative	2038	26.7

```
###

dbnrc <- db2 %>%
  inner_join(get_sentiments("nrc"))

kable ( a <- as.data.frame((summary(as.factor(dbnrc$sentiment)))) %>%
          rename(wrds = `(summary(as.factor(dbnrc$sentiment)))`) %>%
         arrange(desc(wrds)) %>%
          mutate(prop = wrds / sum(wrds) * 100) ,
        caption = "Sentimiento con libreria 'nrc'",
        col.names =  c("No. palabras", "Prop. aparición"),
                       digits = 2)
```

Table 2: Sentimiento con libreria 'nrc'

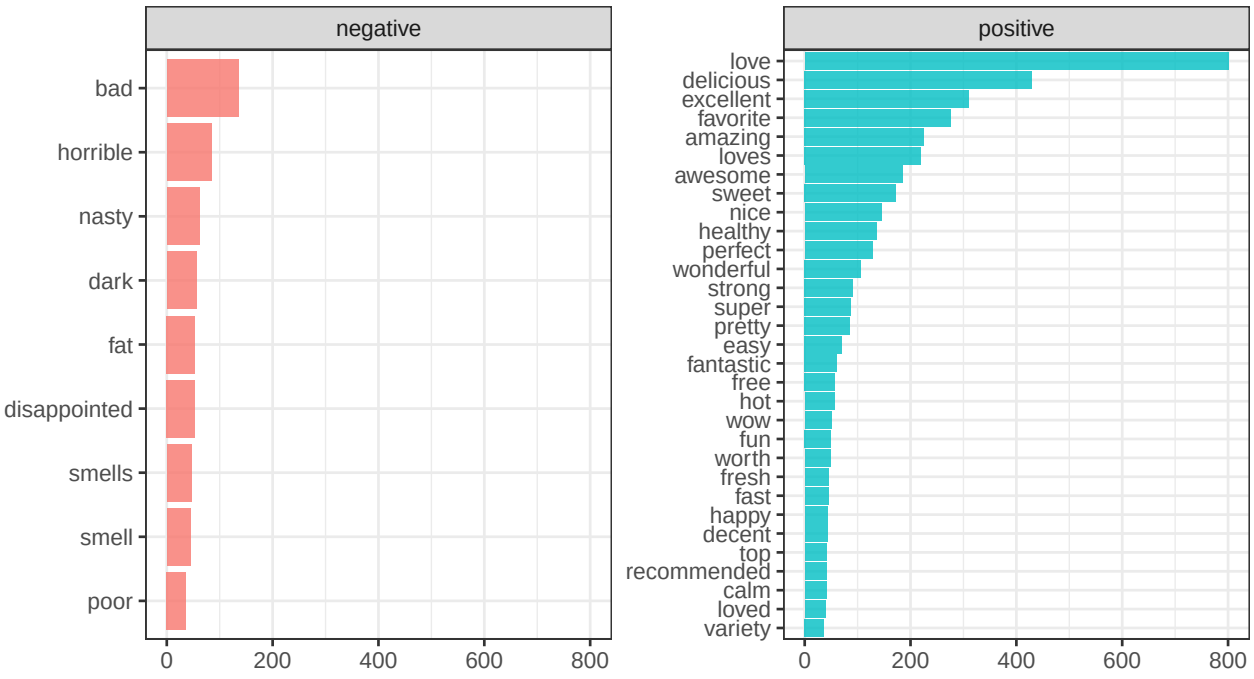
	No. palabras	Prop. aparición
positive	5908	28.25
joy	3978	19.02
trust	2903	13.88
negative	1875	8.96
anticipation	1552	7.42
disgust	1017	4.86
surprise	945	4.52
anger	940	4.49
sadness	912	4.36
fear	886	4.24

```
###

dbaffin2 <- dbaffin %>%
  count(word, value, sort = TRUE) %>%
  ungroup() %>%
  top_n(40) %>%
  ggplot(aes(reorder(word, n), n, fill = value))+
  geom_bar(alpha = 0.8, stat = "identity", show.legend = FALSE) +
  facet_wrap(~value, scales = "free_y") +
  coord_flip() +
  #labs(title = "Sentiment AFINN", y = "Contribution", x = NULL) +
  labs(y = "Contribution to sentiment AFFIN", x = NULL) +
  theme(plot.title = element_text(hjust = 0.5)) +
   theme_bw()

dbaffin2
```

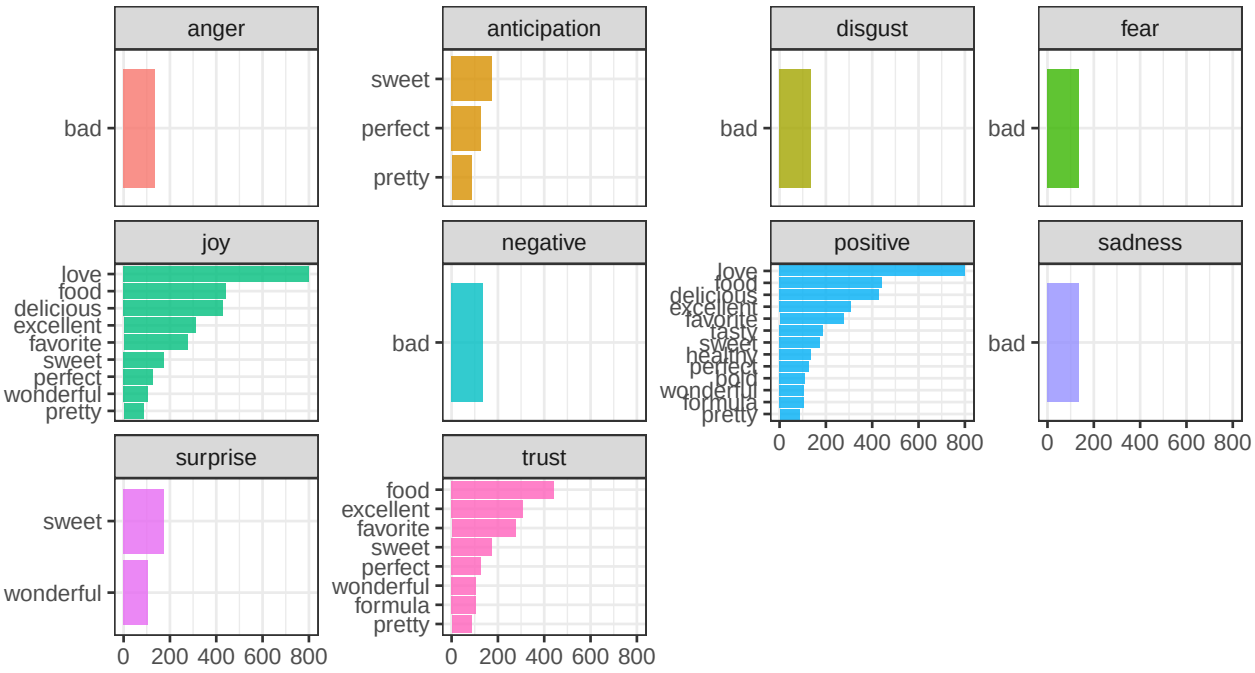


Contribution to sentiment AFFIN

```
dbbing2 <- dbbing %>%
  count(word, sentiment, sort = TRUE) %>%
  ungroup() %>%
  top_n(40) %>%
  ggplot(aes(reorder(word, n), n, fill = sentiment))+
  geom_bar(alpha = 0.8, stat = "identity", show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y") +
  labs(y = "Contribution to sentiment BING", x = NULL) +
  coord_flip() + theme_bw()

dbbing2
```



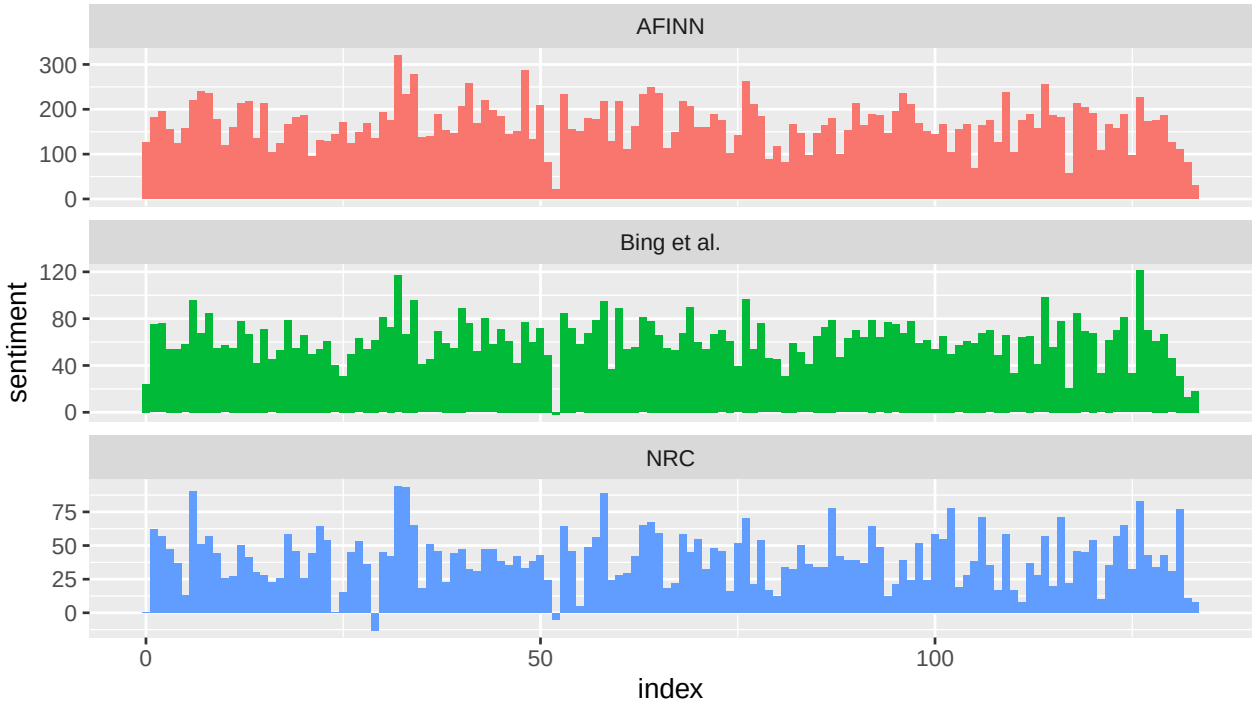

```
dbnrc2 <- dbnrc %>%
  count(word, sentiment, sort = TRUE) %>%
  ungroup() %>%
  top_n(40) %>%
  ggplot(aes(reorder(word, n), n, fill = sentiment))+
  geom_bar(alpha = 0.8, stat = "identity", show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y") +
  labs(y = "Contribution to sentiment NRC", x = NULL) +
  coord_flip() + theme_bw()

dbnrc2
```



Contribution to sentiment NRC

```
#Creamos índices de todos los diccionarios y los comparo, solo tomaré 100 reviews para ejemplificar.

db2 <- db %>%
  mutate(linenumber = row_number()) %>%
  mutate(Summary= as.character(Summary)) %>%
  unnest_tokens(word, Summary)

afinn <- db2 %>%
  inner_join(get_sentiments("afinn")) %>%
  group_by(index = linenumber %/% 100) %>%
  summarise(sentiment = sum(value)) %>%
  mutate(method = "AFINN")

bing_and_nrc <- bind_rows(
  db2 %>%
    inner_join(get_sentiments("bing")) %>%
    mutate(method = "Bing et al."),
  db2 %>%
    inner_join(get_sentiments("nrc") %>%
                 filter(sentiment %in% c("positive",
                                         "negative"))
    ) %>%
    mutate(method = "NRC")) %>%
  count(method, index = linenumber %/% 100, sentiment) %>%
  pivot_wider(names_from = sentiment,
              values_from = n,
              values_fill = 0) %>%
  mutate(sentiment = positive - negative)

bind_rows(afinn,
          bing_and_nrc) %>%
```

```
  ggplot(aes(index, sentiment, fill = method)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~method, ncol = 1, scales = "free_y")
```




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

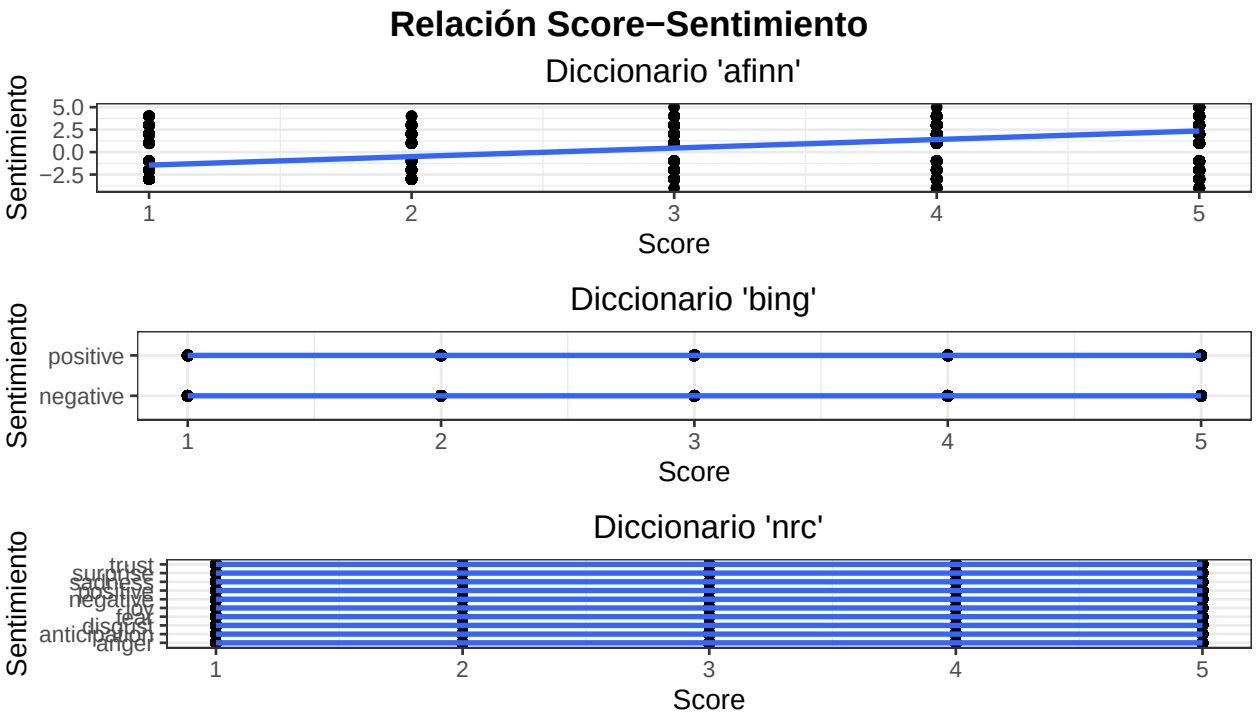
Elaboramos un índice numérico para cada uno de los diccionarios para los primeros 80 reviews, encontramos que las trayectorias son similares en los tres diccionarios, pero con diferencias en magnitudes. Además, encontramos que el diccionario 'affin' nos provee de mayor granularidad en el análisis, al abarcar más categorías de palabras.

3) Score númerico vs Score de NLP (i.e. un análisis de sentimiento)

En esta sección queremos probar si se puede reemplazar un score número con un score de análisis de sentimiento. Encontramos que el índice de sentimiento que genera el diccionario 'afinn' es el único que parece tener una relación positiva con su valor y el score de los productos, por lo que sólo creemos que es recomendable reemplazar el score número con un score de NLP siempre y cuando el diccionario que se utilice sea el 'afinn', ya que con los diccionarios 'bing' y 'nrc' no se observa una relación clara.

```
g1 <- ggplot(dbaffin, aes(x=Score, y=value)) +
  geom_point()+
  geom_smooth(method=lm)+
  theme_bw()  +
      labs( title = "Diccionario 'afinn'",
       y = "Sentimiento",
        x = "Score" ) +
      theme(plot.title = element_text(hjust = 0.5))

g2 <- ggplot(dbbing, aes(x=Score, y=sentiment)) +
  geom_point()+
  geom_smooth(method=lm)+
```

```
  theme_bw()  +
      labs( title = "Diccionario 'bing'",
        y = "Sentimiento",
        x = "Score"  ) +
      theme(plot.title = element_text(hjust = 0.5))


g3 <-ggplot(dbnrc, aes(x=Score, y=sentiment)) +
  geom_point()+
  geom_smooth(method=lm)+
  theme_bw()  +
      labs( title = "Diccionario 'nrc'",
        y = "Sentimiento",
        x = "Score"  ) +
      theme(plot.title = element_text(hjust = 0.5))


 grafs1 <- ggarrange(g1, g2, g3, ncol=1)

annotate_figure(grafs1,
                top = text_grob("Relación Score-Sentimiento", color = "black",
                                face = "bold", size = 14) )
```



4 - Modelo

El objetivo de esta sección es elaborar un modelo de predicción de la calificación con base en el tipo de producto y el texto de los reviews. En este sentido, decidimos primero implementar un CV Multinomial Lasso y en seguida modelo de regresión distributiva multinomial (DMR).

```
# Creamos una base en donde palabras queden en columnas  (spread(words)) y creamos un índice agregando

db_index <- dbbing %>%
  group_by(UserId,word, Prod_Category)%>%
  mutate(indice = row_number())%>%
  filter(indice==1)%>%
  add_count(Prod_Category, word, name = "veces_por_anuncio") %>%
  add_count(word, name = "veces_totales") %>%
  mutate(sent_index = ifelse(sentiment=="positive", 1,0)) %>%
  pivot_wider(id_cols = c("ProductId","UserId","Score","Prod_Category") ,
              names_from = "word",
              values_from = "sent_index",
              values_fn = list("sent_index" = median))

Sent_index = rowSums(db_index[,5:558], na.rm = TRUE)
db_index$Sent_index <- Sent_index

for (i in 5:ncol(db_index)){
  db_index[is.na(db_index[,i]),i] <- 0
}
```

1) Multinomial Logit Lasso

A continuación se elabora un modelo del tipo multinomial logit con la finalidad de predecir la calificación de los reviews de los clientes con base en el tipo de producto adquirido y la información sobre los los textos de los comentarios. El proceso de modelaje considera un algortimo multinomial de tipo LASSO que emplea información numérica en conjunto con una metodología de procesamiento de lenguaje natural (NLP) para clasificar la calificación del review dentro de una categoría k. En este sentido, las etiquetas de calificación varían en un rango entre 1 y 5 conforme a la satisfacción de la compra del cliente.

El procesamiento del texto consideró la tokenización de los comentarios para implementar un análisis de sentimiento. Se utilizó el diccionario de léxico "Afinn" para asignar un valor númerico a cada palabra (token) con base en su connotación, que puede ser tanto positiva como negativa.

```
library('stringi')
library('stopwords')
library(textdata)

base_tidy<-  db %>%
  group_by(UserId,Summary,Prod_Category)%>%
  mutate(indice = row_number())%>%
  filter(indice==1)

base_tidy<- as.data.frame(base_tidy) %>%
  mutate(Summary = as.character(Summary)) %>%
  unnest_tokens(output= palabra, input =Summary)%>%
  ungroup()

base_tidy<-
  base_tidy %>%
  filter(!(palabra %in% stopwords("en") ))%>%
  select(-c(Time,indice))%>%
  mutate(palabra = gsub(' ', '', palabra),
         palabra = gsub("'" ,'', palabra))%>%
```

```r
  inner_join(get_sentiments("afinn"), by=c("palabra"="word"))


# Conteos por categoria y palabra
base_tidy<-
  base_tidy %>%
  group_by(ProductId,UserId,Prod_Category,Prod_Group)%>%
  mutate(sentimiento = sum(value, na.rm = T))%>%
  ungroup()%>%
  add_count(Prod_Category, palabra, name = "veces_por_anuncio") %>%
  add_count(palabra, name = "veces_totales")

base_tidy<-
  base_tidy %>%
  group_by(Prod_Category) %>%
  mutate(proporcion = veces_por_anuncio / sum(veces_por_anuncio) * 100)


base_wide<-   # Aqui ya tengo prop palabras mencionadas por categoria.
  base_tidy %>%
  pivot_wider(id_cols = c("ProductId","UserId","Score","Prod_Category","Prod_Group","sentimiento","Nrev
              names_from = "palabra",
              values_from = "proporcion",
              values_fn = list("proporcion" = mean))

for (i in 8:ncol(base_wide)){
  base_wide[is.na(base_wide[,i]),i] <- 0
}

base_wide <- base_wide%>%
  mutate(review_id = paste(ProductId,UserId,sep ="_"))%>%
  select(review_id,everything())

rm(base_tidy)
```

```r
#Vamos a hacer la prueba con dos modelos: CV Multinomial LASSO

universo <- base_wide %>%
  filter(!duplicated(review_id))

# 1) Dividimos la base training (70%) y validation set (30%)
asignacion <- treatment_assign(data = universo,
                                       share_control = 0.7,
                                       n_t = 1,
                                       strata_varlist ='Prod_Category',
                                       missfits = "global",
                                       seed = 1908,
                                       key ='review_id')

data <- asignacion$data


# Juntamos las bases
universo <- bind_cols(universo,data%>%ungroup()%>%select(treat,strata))
```

```r
# Divido entre training y validation

universo_training <- universo %>%
  filter(treat==0)%>%select(-treat,-strata)


universo_validation <- universo %>%
  filter(treat==1)%>%select(-treat,-strata)


# Undersampling
prop.table(table(universo_training$Score))
```

```
        1         2         3         4         5
0.0740093 0.0419580 0.0600233 0.1561772 0.6678322
```

```r
table(universo_training$Score)
```

```
   1    2    3    4    5
 254  144  206  536 2292
```

```r
universo_training_2 <- universo_training%>%filter(Score==2)

undersampling_1 <- treatment_assign(data = universo_training%>%filter(Score==1)%>%mutate(Score=factor(S
                                    share_control = 0.57,
                                    n_t = 1, strata_varlist ="Prod_Category",missfits = "global",
                                    seed = 1908, key ='review_id')$data

undersampling_1 <- undersampling_1%>%filter(treat==0)

undersampling_3 <- treatment_assign(data = universo_training%>%filter(Score==3)%>%mutate(Score=factor(S
                                    share_control = 0.7,
                                    n_t = 1, strata_varlist ="Prod_Category",missfits = "global",
                                    seed = 1908, key ='review_id')$data

undersampling_3 <- undersampling_3%>%filter(treat==0)

undersampling_4 <- treatment_assign(data = universo_training%>%filter(Score==4)%>%mutate(Score=factor(S
                                    share_control = 0.27,
                                    n_t = 1, strata_varlist ="Prod_Category",missfits = "global",
                                    seed = 1908, key ='review_id')$data

undersampling_4 <- undersampling_4%>%filter(treat==0)

undersampling_5 <- treatment_assign(data = universo_training%>%filter(Score==5)%>%mutate(Score=factor(S
                                    share_control = 0.065,
                                    n_t = 1, strata_varlist ="Prod_Category",missfits = "global",
                                    seed = 1908, key ='review_id')$data

undersampling_5 <- undersampling_5%>%filter(treat==0)
```

```
universo_training_1<- universo_training%>%
  filter(review_id %in% intersect(undersampling_1$review_id,universo_training$review_id))

universo_training_3<- universo_training%>%
  filter(review_id %in% intersect(undersampling_3$review_id,universo_training$review_id))

universo_training_4<- universo_training%>%
  filter(review_id %in% intersect(undersampling_4$review_id,universo_training$review_id))

universo_training_5<- universo_training%>%
  filter(review_id %in% intersect(undersampling_5$review_id,universo_training$review_id))


universo_training_undersampling <- bind_rows(universo_training_1,universo_training_2)

universo_training_undersampling <- bind_rows(universo_training_undersampling,universo_training_3)

universo_training_undersampling <- bind_rows(universo_training_undersampling,universo_training_4)

universo_training_undersampling <- bind_rows(universo_training_undersampling,universo_training_5)

table(universo_training_undersampling$Score)
```

```
  1   2   3   4   5
144 144 144 144 148
```

```
rm(undersampling_1,undersampling_3,undersampling_4,undersampling_5,
   universo_training_1,universo_training_2,universo_training_3,universo_training_4,universo_training_5)
```

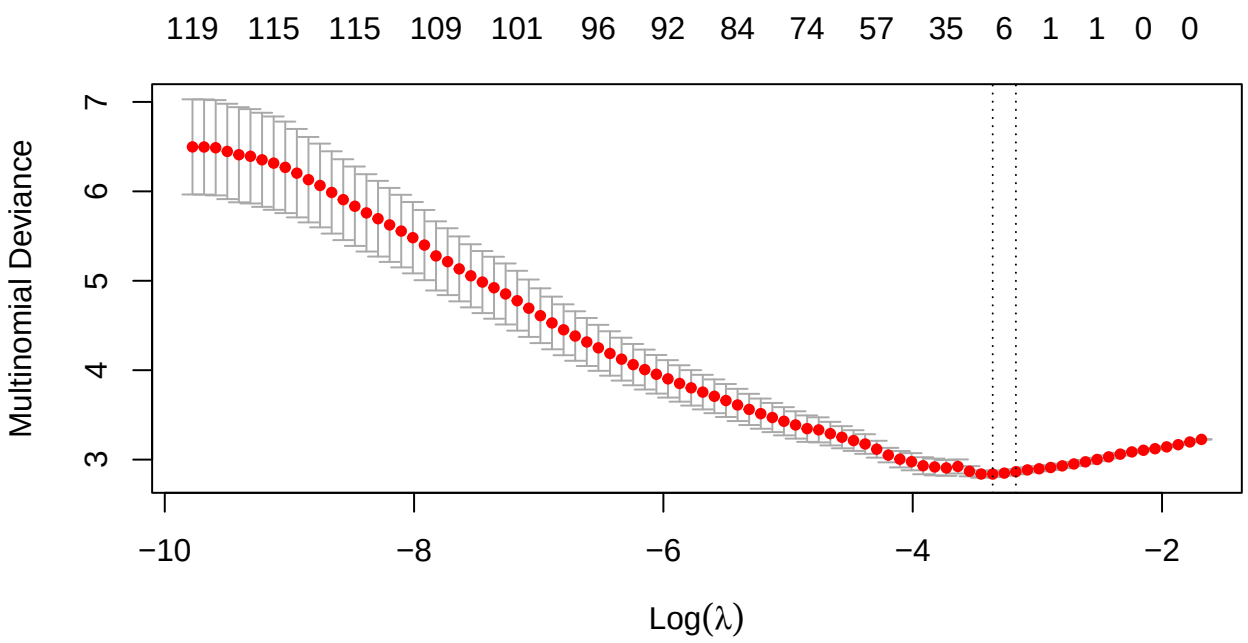
```
#-------------------------------------------------------------------------------
#1 CV MULTINOMIAL LASSO
#-------------------------------------------------------------------------------
# Creamos los vectores y y la matriz de X's
score <- universo_training_undersampling$Score

X <- universo_training_undersampling%>%ungroup()%>%
  select(-c(review_id,ProductId,UserId,Score))

# Creamos la matriz sparse
X <- sparse.model.matrix(~.+0,data = X)
dim(X)
```

```
[1] 724 514
```

```
library(glmnet)
cl<-makeCluster(8)
inicio<-Sys.time()

multinomial_lasso<-cv.glmnet(x = X, y = score, verb = T, cl = cl, family = "multinomial")

(tiempo<-Sys.time() - inicio)
```

```
Time difference of 25.9288 secs
```

```
#save(multinomial_lasso,file="Modelos/multinomial_lasso.RData")

# Graficamos el CV Binomial Deviance vs Complexity

plot(multinomial_lasso)
```




```
multinomial_lasso$lambda.min
```

```
[1] 0.03479203
```

```
#BUSCAR COMO GRAFICAR LOS COEFICIENTES CON GLMNET

# Generamos las predicciones

X <- universo_validation%>%ungroup()%>%
  select(-c(review_id,ProductId,UserId,Score))

X <- sparse.model.matrix(~.+0,data = X)
dim(X)
```

```
[1] 1471  514
```

```
predicciones<- universo_validation%>%
  select(review_id,Score)

prediccion <- data.frame(predict(multinomial_lasso, newx = X, type ='response'))

predicciones_lasso <- bind_cols(predicciones,prediccion)

predicciones_lasso <- predicciones_lasso%>%
  rename(score_1 = "X1.1",
         score_2 = "X2.1",
         score_3 = "X3.1",
         score_4 = "X4.1",
```

```
         score_5 = "X5.1")

rm(predicciones,prediccion)
```

Se estiman las curvas ROC para validar el desempeño del modelo en términos de su poder de clasificación. Se observa un bajo desempeño predictivo que sugiere que la información empleada no logra predecir la calificación del review del cliente con exactitud. Adicionalmente, es probable que esto se explique por el reducido grupo de observaciones que resulta del proceso de undersampling, y que afecta la capacidad de aprendizaje del modelo.

```
library(fastDummies)
library(tidymodels)

#Generamos las graficas de la curva ROC y la matriz de confusión

predicciones_lasso <- predicciones_lasso%>%
  mutate(Score = factor(Score, levels = c("5","4","3","2","1")))%>%
  ungroup()

curva_roc <- predicciones_lasso%>%
  dummy_cols(select_columns = "Score")%>%
  mutate(pred_1 = if_else(score_1>score_2 & score_1>score_3 & score_1>score_4 & score_1>score_5,1,0),
         pred_2 = if_else(score_2>score_1 & score_2>score_3 & score_2>score_4 & score_2>score_5,1,0),
         pred_3 = if_else(score_3>score_1 & score_3>score_2 & score_3>score_4 & score_3>score_5,1,0),
         pred_4 = if_else(score_4>score_1 & score_4>score_2 & score_4>score_3 & score_4>score_5,1,0),
         pred_5 = if_else(score_5>score_1 & score_5>score_2 & score_5>score_3 & score_5>score_4,1,0))

curva_roc <- curva_roc%>%
  mutate_at(vars(Score_1,Score_2,Score_3,Score_4,Score_5), function(x) x = factor(x))


# Estimamos las curvas ROC

roc_1 <- roc_curve(data=curva_roc, truth=Score_1, score_1)
roc_auc(curva_roc, truth=Score_1, score_1)
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 roc_auc binary         0.117
```

```
roc_2 <- roc_curve(data=curva_roc, truth=Score_2, score_2)
roc_auc(curva_roc, truth=Score_2, score_2)
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 roc_auc binary         0.267
```

```
roc_3 <- roc_curve(data=curva_roc, truth=Score_3, score_3)
roc_auc(curva_roc, truth=Score_3, score_3)
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 roc_auc binary         0.459
```

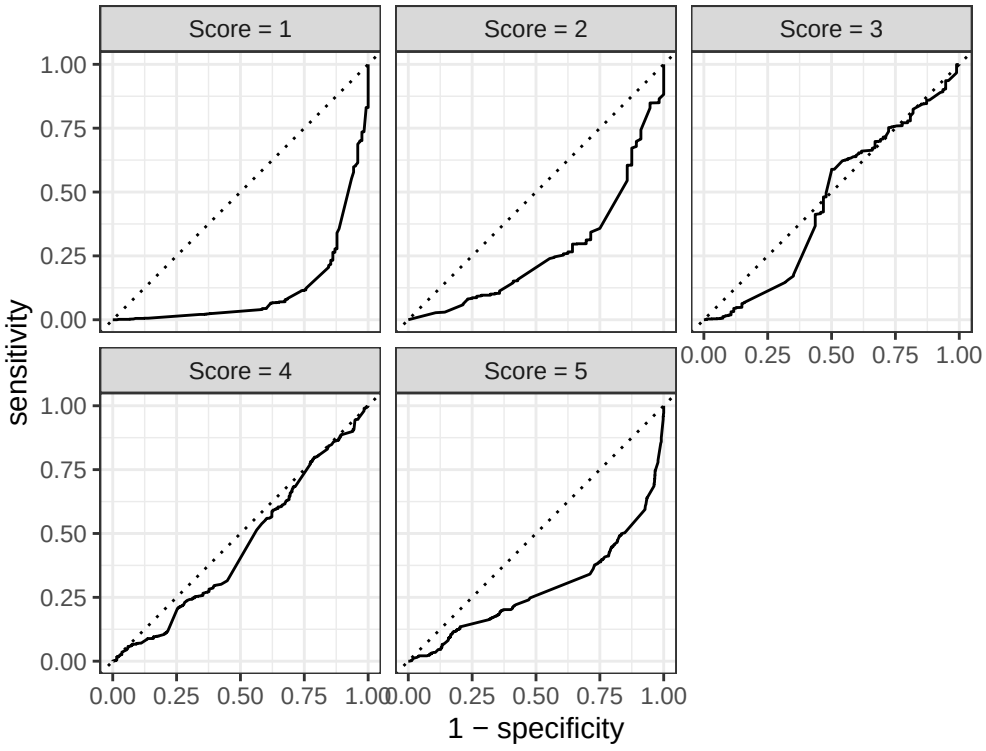
```
roc_4 <- roc_curve(data=curva_roc, truth=Score_4, score_4)
roc_auc(curva_roc, truth=Score_4, score_4)
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 roc_auc binary         0.451
```

```
roc_5 <- roc_curve(data=curva_roc, truth=Score_5, score_5)
roc_auc(curva_roc, truth=Score_5, score_5)
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 roc_auc binary         0.284
```

```
roc <- bind_rows(roc_1%>%mutate(Pred = "Score = 1"),roc_2%>%mutate(Pred = "Score = 2"))
roc <- bind_rows(roc,roc_3%>%mutate(Pred = "Score = 3"))
roc <- bind_rows(roc,roc_4%>%mutate(Pred = "Score = 4"))
roc <- bind_rows(roc,roc_5%>%mutate(Pred = "Score = 5"))

rm(roc_1,roc_2,roc_3,roc_4,roc_5)

ggplot(roc,aes(x=1-specificity,y=sensitivity))+
  geom_path()+
  geom_abline(lty =3)+coord_equal()+theme_bw()+
  facet_wrap(~Pred,2)
```



2) Modelo de Regresión Distributiva Multinomial (DMR)

Adicionalmente decidimos correr un modelo de regresión distributiva multinomial propuesto por Taddy (2018). La idea de utilizar esta metodología surge de identificar que los scores son variables mutuamente excluyentes y que pueden pertenecer a más de dos categorias, además la ventaja de utilizar esta metodología nos permite correr regresiones simultaneas e individuales en un modelo logístico asumiento que cada categoría fue generada mediante una distribución Poisson.

Una ventaja sobre los modelos de Taddy es que están optimizados mediante la librería parallel y por lo tanto corre simultaneamente los modelos. Esto hace que el tiempo de estimación sea mínimo

Para correr el modelo tomaremos las bases de datos construidas en la secciones anteriores utilizando, nuevamente, la técnica de undersampling.

```
#Creemos la base de datos
score <- universo_training_undersampling$Score

X <- universo_training_undersampling%>%ungroup()%>%
  select(-c(review_id,ProductId,UserId,Score))

X <- sparse.model.matrix(~.+0,data = X)
dim(X)
```

```
[1] 724 514
```

```
score <- as.data.frame(score)
dim(score)
```

```
[1] 724   1
```

```
# VAMOS A CORRER EL MODELO
library(parallel)
library(distrom)
cl <- makeCluster(4)
cl
```

```
socket cluster with 4 nodes on host 'localhost'
```

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

#Esta parte nos da las betas y lambdas de los coeficientes del multinomial distributed regression
par(mfrow=c(3,2))
for(j in 1:5){
    plot(fits[[j]])
    mtext(names(fits)[j],font=2,line=2) }

stopCluster(cl)

# Tomamos los coeficientes
B <- coef(fits)
# Creamos las base de validación
X_val <- universo_validation%>%ungroup()%>%
  select(-c(review_id,ProductId,UserId,Score))
```

```r
X_val <- sparse.model.matrix(~.+0,data = X_val)
dim(X_val)
```

```
[1] 1471  514
```

```r
## Fitted probability by true response
predicciones<- universo_validation%>%
  select(review_id,Score)

P <- data.frame(predict(B, X_val, type="response"))

predicciones_dmr <- bind_cols(predicciones, P)

predicciones_dmr <- predicciones_dmr%>%
  rename(score_1 = "X1",
         score_2 = "X2",
         score_3 = "X3",
         score_4 = "X4",
         score_5 = "X5")

#rm(predicciones_dmr,P)


# Curvas ROC

library(fastDummies)
library(tidymodels)

predicciones_dmr <- predicciones_dmr%>%
  mutate(Score = factor(Score, levels = c("5","4","3","2","1")))%>%
  ungroup()%>%
  select(-Prod_Category)

curva_roc <- predicciones_dmr%>%
  dummy_cols(select_columns = "Score")%>%
  mutate(pred_1 = if_else(score_1>score_2 & score_1>score_3 & score_1>score_4 & score_1>score_5,1,0),
         pred_2 = if_else(score_2>score_1 & score_2>score_3 & score_2>score_4 & score_2>score_5,1,0),
         pred_3 = if_else(score_3>score_1 & score_3>score_2 & score_3>score_4 & score_3>score_5,1,0),
         pred_4 = if_else(score_4>score_1 & score_4>score_2 & score_4>score_3 & score_4>score_5,1,0),
         pred_5 = if_else(score_5>score_1 & score_5>score_2 & score_5>score_3 & score_5>score_4,1,0))

curva_roc <- curva_roc%>%
  mutate_at(vars(Score_1,Score_2,Score_3,Score_4,Score_5), function(x) x = factor(x))


# Estimamos las curvas ROC

roc_1 <- roc_curve(data=curva_roc, truth=Score_1, score_1)
roc_auc(curva_roc, truth=Score_1, score_1)
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 roc_auc binary         0.117
```

```r
roc_2 <- roc_curve(data=curva_roc, truth=Score_2, score_2)
roc_auc(curva_roc, truth=Score_2, score_2)
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 roc_auc binary         0.285
```

```r
roc_3 <- roc_curve(data=curva_roc, truth=Score_3, score_3)
roc_auc(curva_roc, truth=Score_3, score_3)
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 roc_auc binary         0.412
```

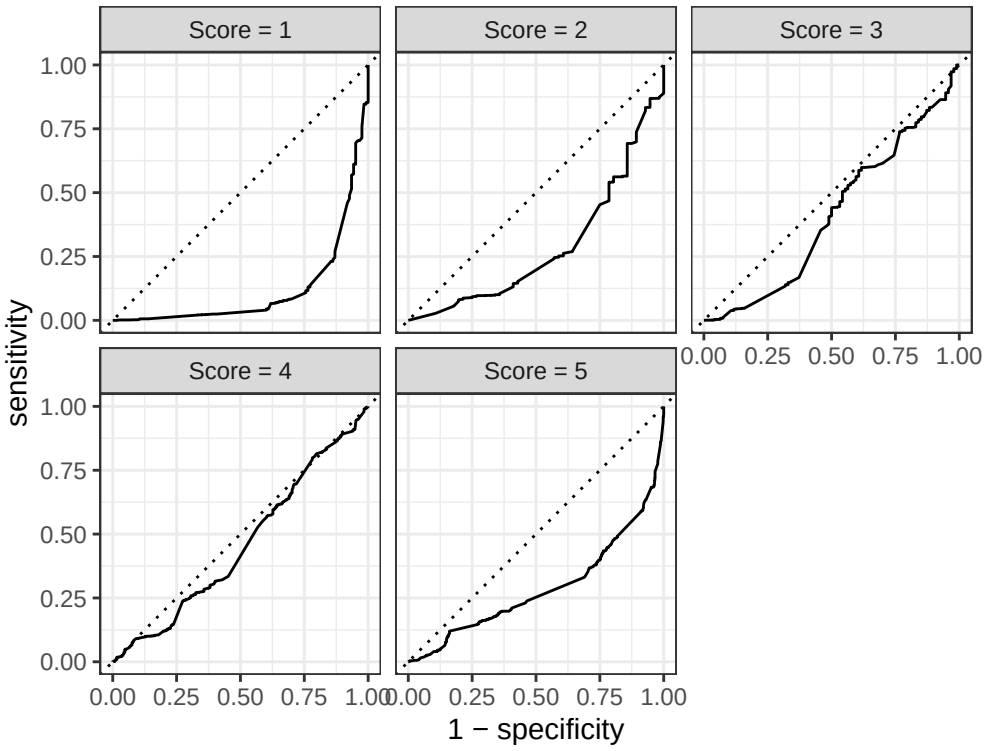
```r
roc_4 <- roc_curve(data=curva_roc, truth=Score_4, score_4)
roc_auc(curva_roc, truth=Score_4, score_4)
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 roc_auc binary         0.459
```

```r
roc_5 <- roc_curve(data=curva_roc, truth=Score_5, score_5)
roc_auc(curva_roc, truth=Score_5, score_5)
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 roc_auc binary         0.288
```

```r
roc <- bind_rows(roc_1%>%mutate(Pred = "Score = 1"),roc_2%>%mutate(Pred = "Score = 2"))
roc <- bind_rows(roc,roc_3%>%mutate(Pred = "Score = 3"))
roc <- bind_rows(roc,roc_4%>%mutate(Pred = "Score = 4"))
roc <- bind_rows(roc,roc_5%>%mutate(Pred = "Score = 5"))

ggplot(roc,aes(x=1-specificity,y=sensitivity))+
  geom_path()+
  geom_abline(lty =3)+coord_equal()+theme_bw()+
  facet_wrap(~Pred,2)
```



5 - Conclusión

El diccionario de sentimientos affin transforma las palabras en calificaciones que van del -5 al 5, donde el 5 corresponde a los sentimientos más positivos y el -5 a los más negativos. Resultaba previsible que al convertir las palabras a variables numéricas pudiéramos encontrar una relación entre dichas y el score que se asigna a cada venta; sin embargo, esto no resultó como se esperaba. Algunas de las razones que encontramos son las siguientes.

Una primera, establece que las variables numéricas que arrojan los diccionarios corresponden unicamente a una palabra y no toman en consideración palabras conjuntas conocidas como bigramas, trigramas o incluso n-gramas. Esto elimina cierta flexibilidad a las combinaciones posibles que pudieron haberse obtenido para potencializar los resultados.

Una segunda se desenvuelve en el contexto de una muestra relativamente pequeña y en la cual no todas las calificaciones contenían comentarios, además de que no todos los 10 tipos de productos contienen la misma cantidad de palabras. Encontramos que los alimentos tienen el mayor número de palabras repetidas, seguidos de compras para mascotas. Esto adiciona cierta restricción al modelo.

Un tercer elemento que sobresalió durante el análisis es que las calificaciones no se distribuyen homogeneamente, pues aparentemente hay una mayor propensión a calificaciones más elevadas que a las mas bajas. No descartamos que la mayoría de la gente que califica muy bien gusta de hacerlo, mientras que las menores calificaciones al final pudieran estar relacionadas con un sesgo por evitar ser quien le resultó defectuosa la compra o simplemente se decide no contestar.

En este contexto, se recomienda utilizar los scores de sentimiento no como predictores de la calificación de los productos, sino como una medida complementaria de estas. La utilidad de un score de sentimiento no es tanto predecir el score de un produto -dado que la mayoría de las personas califican con 4 o 5 estrellas sus productos y además pareciera ser el caso de que solo dejan reseñas cuando un producto les pareció excelente y son poco propensos a escribir reseñas malas cuando el producto fue satisfactorio o malo- sino diferenciar entre productos con la misma calificación, es decir, aún dentro de un producto con califación 5, el score de sentimiento distingue de manera rápida y clara, los productos que son buenos de aquellos que son excelentes.

De este modo, la incorporación de análisis de sentimiento a la estrategia de ventas podría ser ventajoso para una campaña focalizada en los productos no solo de alta califación, sino que contienen la mayor cantidad de reviews positivos.