Tratamiento Inteligente de datos

(TID)

Prácticas de la asignatura

2018-2019

En colaboración con:

Participantes

Alejandro Campoy Nieves: alejandroac79@correo.ugr.es

Gema Correa Fernández: gecorrea@correo.ugr.es

Luis Gallego Quero: lgaq94@correo.ugr.es

Jonathan Martín Valera: jmv742@correo.ugr.es

Andrea Morales Garzón: andreamgmg@correo.ugr.es

Índice

Índice de figuras

Índice de cuadros

Descripción de los paquetes necesarios

A continuación, se describen los paquetes necesarios para el desarollo del proyecto:

- `arules` : Paquete que proporciona la infraestructura para representar, manipular y analizar datos y patrones de transacción (conjuntos de elementos frecuentes y reglas de asociación). Se puede instalar usando : *install.packages("arules")*.
- `arulesViz` : Paquete que extiende el paquete 'arules' con varias técnicas de visualización para reglas de asociación y conjuntos de elementos. El paquete también incluye varias visualizaciones interactivas para la exploración de reglas. Se puede instalar usando : *install.packages("arulesViz")*.
- `car` : Paquete que nos propociona distintas funciones. : *install.packages("car")*.
- `cluster` : Paquete que nos proporciona métodos para el análisis de clusters. : *install.packages("cluster")*.
- `caret` : Paquete para entrenamiento de clasificación y regresión. Se puede instalar usando : *install.packages("quanteda")*.
- `dbscan` : Paquete que proporciona implementaciones de varios algoritmos basados en densidad de la familia DBSCAN para datos espaciales. Se puede instalar usando : *install.packages("dbscan")*.
- `devtools` : Paquete que contiene una colección de herramientas de desarrollo de paquetes, usando conjuntamento con `rword2vec`, para obtener la agrupación de sinónimos. Se puede instalar usando : *install.packages("devtools")*.
- `dplyr` : Paquete que agiliza el trabajo con los datos. Se puede instalar usando : *install.packages("dplyr")*.
- `e1071` : Paquete para realizar *fuzzy clustering*, clasificador de *Naive Bayes*... Se puede instalar usando : *install.packages("e1071")*.
- `fclust` : Paquete que nos proporciona algoritmos para la agrupación difusa, índices de validez de clústeres y gráficos para la validez de estos. Además de la visualización de los resultados de la agrupación difusa. : *install.packages("fclust")*.
- `factoextra` : Proporciona algunas funciones sencillas para extraer y visualizar la producción de análisis de datos multivariados. : *install.packages("factoextra")*.
- `ggpubr` : Paquete que proporciona algunas funciones de fácil manejo para crear y personalizar gráficos listos para ser usados en *ggplot2*. : *install.packages("ggpubr")*.
- `ggplot2` : Paquete para realizar gráficas. Se puede instalar usando : *install.packages("ggplot2")*.
- `glmnet` : Paquete que nos proporciona rocedimientos eficaces para la adaptación de la red para la regresión lineal, los modelos de regresión logística y multinomial, la regresión de Poisson y el modelo de Cox. : *install.packages("glmnet")*.
- `igraph` : Paquete que nos proporciona procesos para gráficos simples y análisis de redes. Puede manejar gráficos grandes muy bien, gráficos aleatorios y gráficos regulares, además de visualización de estos. : *install.packages("igraph")*.
- `MASS` : Paquete que nos propociona funciones y conjuntos de datos de soporte. : *install.packages("MASS")*.
- `magrittr` : Paquete que proporciona un mecanismo para encadenar comandos con %>%. Se puede instalar usando : *install.packages("magrittr")*.
- `NLP` : Paquete con clases básicas y métodos para el procesamiento del lenguaje natural. Se puede instalar usando : *install.packages("NLP")*.
- `ppclust` : Paquete que nos permite la agrupación de particiones de un conjunto de datos en subconjuntos o clústeres no superpuestos mediante el uso de los algoritmos de agrupación probabilística basados en prototipos. : *install.packages("ppclust")*.

- `plyr` : Paquete que nos ofrece herramientas para dividir, aplicar y combinar datos. Se puede instalar usando : *install.packages("plyr").*
- `proxy` : Paquete que proporciona un *framework* para cálculo eficiente. Se puede instalar usando : *install.packages("proxy").*
- `quanteda` : Paquete para análisis cuantitativo de texto en R, usado para gestionar corpus, crear y manipular tokens y n-gramas, analizar palabras clave..., representando visualmente el texto y los análisis de texto. Se puede instalar usando : *install.packages("quanteda").*
- `rlm` : Paquete que nos propociona una adaptación robusta de un modelo lineal que puede responder en forma de matriz. : *install.packages("rlm").*
- `rattle` : Paquete que permite al usuario cargar rápidamente datos desde un archivo CSV, transformarlos y explorarlos, construir y evaluar modelos, y además, exportarlos. : *install.packages("rattle").*
- `randomForest` : Paquete que nos proporciona clasificación y regresión basada en árboles usando entradas aleatorias. : *install.packages("randomForest").*
- `RColorBrewer` : Paquete que proporciona paletas de colores. Se puede instalar usando : *install.packages("wordcloud").*
- `rlist` : Paquete que proporciona un conjunto de funciones para la manipulación de datos en forma de lista. Se puede instalar usando : *install.packages("rlist").*
- `ROCR` : Paquete que nos permite hacer uso de gráficos ROC (curva ROC), entre otros. Se puede instalar usando : *install.packages("ROCR").*
- `RTextTools` : Paquete para realizar clasificación automática de textos mediante aprendizaje supervisado. Se puede instalar usando : *install.packages("RTextTools").*
- `rword2vec` : Paquete que toma un corpus de texto como entrada y produce los vectores de palabra como salida, usado especialmente para obtener las distancias que existen entre un término y los términos semejantes en el texto de formación (aprende la representación vectorial de las palabras). Se puede instalar usando : *install_github("mukul13/rword2vec").*
- `stargazer` : Paquete que produce código LaTeX, código HTML/CSS y texto ASCII para tablas bien formateadas que contienen resultados del análisis de regresión de varios modelos en paralelo, así como un resumen de estadísticas. : *install.packages("stargazer").*
- `SnowballC` : Paquete adicional para minería de datos, implementa un algoritmo que permite reducir el número de términos con lo que trabajar, es decir, agrupa aquellos términos que contienen la misma raíz. El paquete soporta los siguientes idiomas: alemán, danés, español, finlandés, francés, húngaro, inglés, italiano, noruego, portugués, rumano, ruso, sueco y turco. Se puede instalar usando : *install.packages("SnowballC").*
- `tidytext` : Paquete para minería de textos para procesamiento de textos y análisis de sentimientos usando `dplyr`, `ggplot2`, y otras herramientas ordenadas. Se puede instalar usando : *install.packages("tidytext").*
- `tm` : Paquete específico para minería de datos, permite procesar datos de tipo texto. Se puede instalar usando : *install.packages("tm").*
- `tseries` : Paquete para análisis de series temporales y computación financiera. Se puede instalar usando : *install.packages("tseries").*
- `wesanderson` : Paquete que proporciona paletas de colores. Se puede instalar usando : *install.packages("wesanderson").*
- `wordcloud` : Paquete para crear gráficas de nubes de palabras, permitiendo visualizar las diferencias y similitudes entre documentos. Se puede instalar usando : *install.packages("wordcloud").*

1. Comprender el problema a resolver

El *dataset* **Drug Review Dataset**, proporcionado por *UCI Machine Learning Repository*, contiene una exhaustiva base de datos de medicamentos específicos, en la cual, el conjunto de datos muestra revisiones de pacientes sobre medicamentos específicos para unas condiciones particulares. Dichas revisiones se encuentran desglosadas en función del tema que se esté tratando: beneficios, efectos secundarios y comentarios generales. De igual modo, se dispone de una calificación de satisfacción general, es decir, de una calificación en base a los efectos secundarios del medicamento y de otra, en base a la efectividad del mismo.

En este proyecto nos centraremos en el **análisis y experiencia qué tienen los usuarios con ciertos tipos de medicamentos**, para la realización y aplicación de las técnicas explicadas a lo largo del curso. Para ello, se proponen los siguientes objetivos principales:

- Realizar un análisis de sentimientos a partir de la experiencia de dichos usuarios en el uso de ciertos medicamentos, como por ejemplo ver la efectividad del medicamento cuánto está relacionado con los efectos secundarios o beneficios del mismo.
- Comparar la efectividad o efectos secundarios del medicamento, de acuerdo a la puntuación del medicamento según el paciente.
- Compatibilizar dicho modelo de datos con otros conjuntos de datos aportados en **Drugs.com**.

Las características de este conjunto de datos vienen descritas en la siguiente tabla 1:

Características del Data Set	Multivariable, texto
Características de los atributos	Entero
Tareas asociadas	Clasificación, regresión, clustering
Número de instancias	4143
Número de atributos	8
Valores vacíos	N/A
Área	N/A
Fecha de donación	10/02/2018
Veces visualizado	16759

Cuadro 1: Información del conjunto de datos

Los datos se dividen en un conjunto train (75 %) y otro conjunto test (25 %) y se almacenan en dos archivos *.tsv* (tab-separated-values), respectivamente. Los atributos que tenemos en este dataset son:

1. **urlDrugName** (categorical): nombre del medicamento/fármaco
2. **rating** (numerical): clasificación o puntuación del 1 a 10 del medicamento según el paciente
3. **effectiveness** (categorical): clasificación de la efectividad del medicamento según el paciente (5 posibles valores)
4. **sideEffects** (categorical): clasificación de los efectos secundarios del medicamento según el paciente (5 posibles valores)
5. **condition** (categorical): nombre de la condición (diagnóstico)
6. **benefitsReview** (text): opinión del paciente sobre los beneficios
7. **sideEffectsReview** (text): opinión del paciente sobre los efectos secundarios
8. **commentsReview** (text): comentario general del paciente

2. Preprocesamiento de datos

En este apartado, pondremos los datos a punto para la aplicación de diversas técnicas. Por tanto, para poder analizar dicho *dataset* y realizar el preprocesamiento al mismo, lo primero que se va hacer es leer el conjunto

de datos *train* y *test*. Una vez leídos, se procederán a aplicar las siguientes técnicas con el fin de limpiar los datos:

1. Lectura del dataset
2. Eliminación de columnas que información irrelevante para el análisis
3. Eliminación de filas que no proporcionan información alguna
4. Cuantificación de variables
5. Representación gráfica de los datos
6. Creación del corpus
7. Representación gráfica de las frecuencias
8. Correlación
9. Eliminar signos de puntuación
10. Conversión de las mayúsculas en minúsculas
11. Eliminación de Stopwords
12. Agrupación de sinónimos
13. TF-IDF
14. Stemming
15. Borrar espacios en blanco innecesarios
16. Valores perdidos
17. Term Document Matrix
18. Sparsity
19.
20.
21.
22.
23.

2.1. Lectura del dataset

A continuación, mediante la función `read.table(...)` procedemos a la lectura de los datos explicados previamente:

2.1.1. Lectura de datos train

Se va a proceder a la lectura del conjunto de datos de entrenamiento.

```
# Lectura de datos train
datos_train <- read.table("datos/drugLibTrain_raw.tsv", sep="\t", comment.char="",
                          quote = "\"", header=TRUE)
```

Disponemos de una matriz de 3107 filas x 9 columnas, asimismo vamos a ver un ejemplo de cómo está distribuida la información. Por ejemplo, para la tercera fila encontramos la siguiente información:

X	urlDrugName	rating	effectiveness	sideEffects	condition
1146	ponstel	10	Highly Effective	No Side Effects	menstrual cramps

Cuadro 2: Información contenida en una fila del conjunto de entrenamiento I

benefitsReview	sideEffectsReview	commentsReview
I was used to having cramps so badly that they would leave me balled up in bed for at least 2 days. The Ponstel doesn't take the pain away completely, but takes the edge off so much that normal activities were possible. Definitely a miracle medication!!	Heavier bleeding and clotting than normal.	I took 2 pills at the onset of my menstrual cramps and then every 8-12 hours took 1 pill as needed for about 3-4 days until cramps were over. If cramps are bad, make sure to take every 8 hours on the dot because the medication stops working suddenly and unfortunately takes about an hour to an hour and a half to kick back in.. if cramps are only moderate, taking every 12 hours is okay.

Cuadro 3: Información contenida en una fila del conjunto de entrenamiento II

De las tablas 2 y 3 podemos extraer que el medicamento **ponstel** con identificador **1146**, tiene la máxima puntuación por parte del paciente (**rating = 10**), el cual tiene un alto nivel de efectividad (**Highly Effective**) sin efectos secundarios (**No Side Effects**), usado para dolores menstruales (**menstrual cramps**), en donde el paciente dice que de estar tumbado en la cama con dolores ha pasado a poder realizar actividades cotidianas sin ningún impedimento. Además, asegura que tomar este medicamento le ha supuesto un sangrado más abundante y coagulación de lo normal. La dosis del medicamento oscila entre una píldora cada 8-12 horas durante 3-4 días.

2.1.2. Lectura de datos test

Se va a proceder a la lectura del conjunto de datos de prueba.

```
# Lectura de datos test
datos_test <- read.table("./datos/drugLibTest_raw.tsv", sep="\t", comment.char="",
                         quote = "\"", header=TRUE)
```

Disponemos una matriz de 1036 filas x 9 columnas, asimismo vamos a ver un ejemplo de cómo está distribuida la información. Por ejemplo, para la primera fila encontramos la siguiente información:

De las tablas 4 y 5 podemos extraer que el medicamento **biaxin** con identificador **1366**, tiene una puntuación de 9 por parte del paciente (**rating = 9**), el cual tiene un nivel considerable de efectividad (**Considerably**

Effective) con efectos secundarios leves (**Mild Side Effects**), usado para la infección sinusal (**sinus infection**), en donde el paciente dice que no está muy seguro de si el antibiótico ha destruido las bacterias que causan su infección sinusal. Además, asegura que tomar este medicamento le da algo de dolor de espalda y algunas náuseas. El paciente tomó los antibióticos durante 14 días y la infección sinusal desapareció al sexto día.

X	urlDrugName	rating	effectiveness	sideEffects	condition
1366	biaxin	9	Considerably Effective	Mild Side Effects	sinus infection

Cuadro 4: Información contenida en una fila del conjunto de prueba I

benefitsReview	sideEffectsReview	commentsReview
The antibiotic may have destroyed bacteria causing my sinus infection. But it may also have been caused by a virus, so its hard to say.	Some back pain, some nauseau.	Took the antibiotics for 14 days. Sinus infection was gone after the 6th day.

Cuadro 5: Información contenida en una fila del conjunto de prueba II

Una vez leídos nuestros datos, procedemos a la transformación y preprocesamiento de los mismos. En donde la representación del documento se llevará a cabo utilizando palabras, después de un debido filtrado para minimizar la dimensión del espacio de trabajo.

2.2. Preprocesamiento de los datos

Dado que la representación total del documento puede tener una alta dimensión, se va a proceder a construir un corpus, necesario para la aplicación de métodos de limpieza y esructuración del texto de entrada e identificación de un subconjunto simplificado de las características del documento, con el fin de poder ser representado en un análisis posterior.

2.2.1. Eliminar columnas

El primer paso que vamos a realizar es la **eliminación de columnas**, las cuales contienen información irrelevante para nuestro análisis.

Eliminar columna ID

Al conjunto de datos utilizado se le ha añadido de forma automática una novena columna, que representa un ID para cada uno de los datos con los que estamos trabajando. Como este ID no nos aporta información alguna, hemos decidido quitarla directamente del *dataframe*. Esta columna se corresponde con la primera columna, por lo cuál, debemos eliminar la columna que se corresponde con la posición 1. Los cambios que hacemos en el *dataset* deben modificarse tanto en el conjunto de test como train para que los resultados sean consistentes.

```
datos_train = datos_train[-1] # Eliminar columna para el ID en el train
datos_test = datos_test[-1] # Eliminar columna para el ID en el test
```

Eliminar columna de commentsReview

Consideramos que la información contenida en *commentsReview* no es de nuestro interés. En este atributo se almacena texto, en el cual los consumidores de los medicamentos suelen poner en la mayoría de casos la frecuencia o la dosis con la que consumen la misma. En otros casos menos frecuentes, se establecen comentarios más arbitrarios en el que se muestran sus sensaciones o información sin relevancia. Incluso en algunos casos

este campo aparece vacío. Es por eso, que hemos decidido eliminar la columna, tanto para el conjunto test como train.

```
datos_train = datos_train[-8] # Eliminar columna para el commentsReview en el train
datos_test = datos_test[-8] # Eliminar columna para el commentsReview en el test
```

2.2.2. Eliminar filas

Además de eliminar las columnas innecesarias, se han localizado tres filas que no aportan información a nuestro análisis. Asimismo, dichas filas perjudicaban la aplicación de técnicas.

- Se elimina la fila 387, porque no dispone de información alguna ni en *benefitsReview* y *sideEffectsReview*.
- Se elimina la fila 928, porque en la columna *condition* dispone de un carácter raro.
- Se elimina la fila 3105, porque no tienen información en *benefitsReview* ("—").

```
datos_train = datos_train[-c(387, 928, 3105),] # Eliminar filas en el train
datos_test = datos_test[-c(387, 928, 3105),] # Eliminar filas en el test
```

2.2.3. Eliminar elementos repetidos (medicamentos)

Como se puede ver a continuación, existen medicamentos repetidos. Sin embargo, no se ha contabilizado realizar una eliminación de dichos medicamentos a priori, debido a que le damos más importancia a las opiniones de los pacientes, con el fin de obtener la efectividad o efectos secundarios del medicamento.

	urlDrugName <fctr>	rating <int>	effectiveness <fctr>	sideEffects <fctr>	condition <fctr>
43	paxil	8	Considerably Effective	Mild Side Effects	depression
70	paxil	1	Ineffective	Severe Side Effects	depression
127	paxil	7	Considerably Effective	Moderate Side Effects	depression
155	paxil	1	Ineffective	Extremely Severe Side Effects	severe depression
180	paxil	1	Ineffective	Extremely Severe Side Effects	ocd
207	paxil	8	Highly Effective	Moderate Side Effects	stress and depression
384	paxil	8	Highly Effective	No Side Effects	anxiety and depression
399	paxil	10	Highly Effective	Mild Side Effects	anxiety
451	paxil	8	Considerably Effective	Mild Side Effects	mild depression
453	paxil	1	Ineffective	Severe Side Effects	depression

1–10 of 20 rows | 1–6 of 7 columns Previous 1 2 Next

Figura 1: Medicamento "paxil" repetido 20 veces en el conjunto test

2.2.4. Cuantificación de variables

Para poder analizar y trabajar más fácilmente con la información de *sideEffects* y *effectiveness*, se va a realizar una conversión de dichas columnas a forma cuantitativa, es decir, vamos asignar una etiqueta numérica a cada valor pertinente, tanto para para *train* como *test*.

A continuación, vamos a cuantificar la columna de *sideEffects*, para ello se añade una nueva columna a nuestro conjunto de datos denominada *sideEffectsNumber* que nos clasifica los posibles valores de la columna *sideEffects* en un rango numérico, comprendido entre 1 y 5. Dicha columna hace referencia a la clasificación de los efectos secundarios del medicamento según el paciente, en donde la etiqueta con valor 1 hará referencia a que no haya ningún efecto secundario y la etiqueta con valor 5 a que tiene efectos secundarios extremadamente graves:

- Extremely Severe Side Effects (efectos secundarios extremadamente graves) : 5
- Severe Side Effects (efectos secundarios graves): 4
- Moderate Side Effects (efectos secundarios moderados) : 3

- Mild Side Effects (efectos secundarios leves) : 2
- No Side Effects (sin efectos secundarios) : 1

```
# Datos Train
datos_train$sideEffectsNumber[datos_train$sideEffects=="Extremely Severe Side Effects"]<-5
datos_train$sideEffectsNumber[datos_train$sideEffects=="Severe Side Effects"]<-4
datos_train$sideEffectsNumber[datos_train$sideEffects=="Moderate Side Effects"] <- 3
datos_train$sideEffectsNumber[datos_train$sideEffects=="Mild Side Effects"]<- 2
datos_train$sideEffectsNumber[datos_train$sideEffects=="No Side Effects"]<- 1

# Datos Test
datos_test$sideEffectsNumber[datos_test$sideEffects=="Extremely Severe Side Effects"]<-5
datos_test$sideEffectsNumber[datos_test$sideEffects=="Severe Side Effects"]<-4
datos_test$sideEffectsNumber[datos_test$sideEffects=="Moderate Side Effects"]<-3
datos_test$sideEffectsNumber[datos_test$sideEffects=="Mild Side Effects"]<-2
datos_test$sideEffectsNumber[datos_test$sideEffects=="No Side Effects"]<-1
```

Podemos comprobar que se ha creado la nueva columna *sideEffectsNumber*, y que se han añadido los cambios comentados anteriormente.

```
head(datos_train$sideEffects, 5)
```

```
## [1] Mild Side Effects   Severe Side Effects No Side Effects
## [4] Mild Side Effects   Severe Side Effects
## 5 Levels: Extremely Severe Side Effects ... Severe Side Effects
```

```
head(datos_train$sideEffectsNumber, 5)
```

```
## [1] 2 4 1 2 4
```

Volvemos a aplicar el mismo procedimiento para la columna de *effectiveness*, creándonos para ello una columna denominada *effectivenessNumber*. Dicha columna, hace referencia a la clasificación de la efectividad del medicamento según el paciente, en donde la etiqueta con valor 1 hace referencia a que el medicamente es ineficaz y la etiqueta con valor 5 a que el medicamente es altamente eficaz:

- Highly Effective (altamente efectivo): 5
- Considerably Effective (considerablemente efectivo) : 4
- Moderately Effective (moderadamente efectivo) : 3
- Marginally Effective (marginalmente efectivo) : 2
- Ineffective (ineficaz) : 1

```
# Datos de entrenamiento
datos_train$effectivenessNumber[datos_train$effectiveness=="Highly Effective"]<-5
datos_train$effectivenessNumber[datos_train$effectiveness=="Considerably Effective"]<-4
datos_train$effectivenessNumber[datos_train$effectiveness=="Moderately Effective"]<-3
datos_train$effectivenessNumber[datos_train$effectiveness=="Marginally Effective"]<-2
datos_train$effectivenessNumber[datos_train$effectiveness=="Ineffective"]<- 1

# Datos de test
datos_test$effectivenessNumber[datos_test$effectiveness=="Highly Effective"]<-5
datos_test$effectivenessNumber[datos_test$effectiveness=="Considerably Effective"]<-4
datos_test$effectivenessNumber[datos_test$effectiveness=="Moderately Effective"]<-3
datos_test$effectivenessNumber[datos_test$effectiveness=="Marginally Effective"]<-2
datos_test$effectivenessNumber[datos_test$effectiveness=="Ineffective"]<-1
```

Comprobamos que se ha creado la nueva columna *effectivenessNumber*, y que se han añadido los nuevos cambios.

```r
head(datos_train$effectiveness, 5)
```

```
## [1] Highly Effective     Highly Effective     Highly Effective
## [4] Marginally Effective Marginally Effective
## 5 Levels: Considerably Effective Highly Effective ... Moderately Effective
```

```r
head(datos_train$effectivenessNumber, 5)
```

```
## [1] 5 5 5 2 2
```

2.2.5. Cálculo del rating ponderado

En este subapartado, se va a realizar una agregación de varias columnas, en donde queremos realizar una valoración general del medicamento. Para ello, se va a realizar una ponderación entre la columna que contiene los efectos secundarios y la efectividad del medicamento. Asimismo, se ha considerado a los efectos secundarios del medicamento con mayor importancia, es por eso que se le ha otorgado una ponderación del 70 % frente al 30 % de la efectividad del medicamento.

$$(sideEffects \cdot 07) + (effectiveness \cdot 03)$$

El motivo de esta ponderación es que se considera que es peor tener efectos secundarios severos en un medicamento, que ser efectivo. Dicha agregación se ha añadido a una nueva columna, denominada **weightedRating**, cuyo resultado contiene una valoración general del medicamento. En donde, dicha transformación, puede ser usada para realizar una compararación con las propias valoraciones de los usuarios.

```r
# Recorremos el dataframe para el conjunto train
for (i in 1:length(datos_train[[1]])) {

  # Obtenemos el valor de sideEffect
  sideEffectNumber <- datos_train$sideEffectsNumber[i]
  # Obtenemos el valor de efectiveness
  effectivenessRating <- datos_train$effectivenessNumber[i]

  # Inicializamos la variable
  sideEffectRating <- 0

  # Convertimos el valor de sideEffect a la misma escala que efectiveness, ya que
  # sideEffect == 1 significa que no tiene efectos secundarios (lo cual es bueno),
  # y efecctiveness == 1 significa que no es efectivo (lo cual no es bueno). Para
  # ello realizamos la siguiente conversión:
  if(sideEffectNumber == 1)
     sideEffectRating <- 5
  else if(sideEffectNumber == 2)
    sideEffectRating <- 4
  else if(sideEffectNumber == 3)
    sideEffectRating <- 3
  else if(sideEffectNumber == 4)
    sideEffectRating <- 2
  else if(sideEffectNumber == 5)
    sideEffectRating <- 1

  # Obtenemos el resultado ponderado en tipo float
  floatResult <- effectivenessRating * 0.7 + sideEffectRating * 0.3
```

```
  # Convertimos el resultado a valor entero.
  integerResult <- as.integer(floatResult)

  # Calculamos la parte decimal y redondeamos
  if(floatResult - integerResult < 0.5) result <- integerResult
  else result <- integerResult+1

  # Añadimos el resultado obtenido y lo multiplicamos por dos para pasarlo a
  # escala (1-10)
  datos_train$weightedRating[i] <- result * 2
}
```

Comprobamos que se ha creado la nueva columna *weightedRating*, y que se han añadido los nuevos cambios.

```
head(datos_train$weightedRating, 10)
```

```
##  [1] 10  8 10  6  4  2 10  8 10  2
```

Ahora ralizamos el mismo proceimiento para el conjunto test, y comprobamos que se ha creado la nueva columna *weightedRating*, y que se han añadido los cambios comentados.

```
head(datos_test$weightedRating, 10)
```

```
##  [1]  8  8  4 10  8  8  6  8 10  2
```

2.2.6. Convertir el rating a variable binaria

Para la realización de algunas técnicas que se explicarán a continuación, se necesita tener una etiqueta o variable binaria comprendida entre 0 y 1, es por eso que se ha optado por escoger la columna que contiene la puntuación del medicamento por parte del paciente, y establecerla entre 0 y 1. En donde el 0, tendrá los valores comprendidos entre 1 y 4 y será que el medicamento no es favorable; y en donde el 1, tendrá los valores comprendidos entre 5 y 10 y será que el medicamento es favorable. Dichos cambios, serán añadidos a una nueva columna llamada **ratingLabel**. Realizamos dicho cambio, tanto para test como train.

```
# Recorremos el dataframe para el conjunto train
for (i in 1:length(datos_train[[1]])){

  # Obtenemos el valor de rating
  rating <- datos_train$rating[i]

  # Cuando los valores comprendidos entre 1 y 4 - no favorable - 0
  if (datos_train$rating[i] < 5) result <- 0
  # Cuando los valores comprendidos entre 5 y 10 - favorable - 1
  else if (datos_train$rating[i] >= 5)
    result <- 1

  # Asignamos el resultado a la nueva variable
  datos_train$ratingLabel[i]  <- result
}
```

2.2.7. Cambiar el orden para la columna sideEffectsNumber

Al comparar las columna *sideEffectsNumber* y *effectivenesNumber*, llegamos a la conclusión, de que ambas siguen órdenes distintos, es decir, el valor 5 en *sideEffectsNumber* dice que el medicamento tiene efectos secundarios extremadamente graves, y el valor 5 en *effectivenesNumber* dice que el medicamento es altamente

efectivo. Por tanto, se ha considerado, que el valor correspondiente al 5, sea positivo y el valor correspondiente a 1 negativo. Para ello, se ha modificado el orden de la columna *sideEffectsNumber*, en donde, la nueva columna **sideEffectsInverse** tiene:

- Extremely Severe Side Effects (efectos secundarios extremadamente graves) : 1
- Severe Side Effects (efectos secundarios graves): 2
- Moderate Side Effects (efectos secundarios moderados) : 3
- Mild Side Effects (efectos secundarios leves) : 4
- No Side Effects (sin efectos secundarios) : 5

Realizamos dicha modificación, tanto para train como test.

```
# Recorremos el dataframe para el conjunto train
for (i in 1:length(datos_train[[1]])){

  # Obtenemos el valor de sideEffect
  sideEffectNumber <- datos_train$sideEffectsNumber[i]

  # Inicializamos la variable
  sideEffectRating <- 0

  # Cambiamos el orden para el el valor de sideEffect
  if(sideEffectNumber == 1)
     sideEffectRating <- 5
  else if(sideEffectNumber == 2)
    sideEffectRating <- 4
  else if(sideEffectNumber == 3)
    sideEffectRating <- 3
  else if(sideEffectNumber == 4)
    sideEffectRating <- 2
  else if(sideEffectNumber == 5)
    sideEffectRating <- 1

  # Añadimos el resultado obtenido
  datos_train$sideEffectsInverse[i] <- sideEffectRating
}
```

```
head(datos_train$sideEffectsNumber, 10)
```

```
##  [1] 2 4 1 2 4 4 2 1 1 5
```

```
head(datos_train$sideEffectsInverse, 10)
```

```
##  [1] 4 2 5 4 2 2 4 5 5 1
```

2.2.8. Representación gráfica de los datos

Antes de comenzar con el análisis exploratorio, vamos realizar distintas gráfica de barras con el fin de comprender mejor los datos, antes del procesamiento. Primero realizaremos un gráfico de barras de las clasificaciones del medicamente por parte del paciente.

En la figura 2, se observa como más del 50 % de los medicamentos obtienen una nota superior 6 por parte del paciente. Esto nos sugiere que el paciente, tiene una buena opinión sobre un alto porcentaje de los medicamentos, lo que puede dar lugar a opiniones más positivas. Por otro lado, vamos a mostrar gráficamente los efectos secundarios del medicamento según el paciente. En donde el 1 significa que el medicamento tiene pocos efectos secundarios y el 5 que tiene muchos efectos secundarios. Como se aprecia en la figura 4, un alto porcentaje de los medicamentos no tiene efectos secundarios, de acuerdo a las valoraciones de los pacientes.

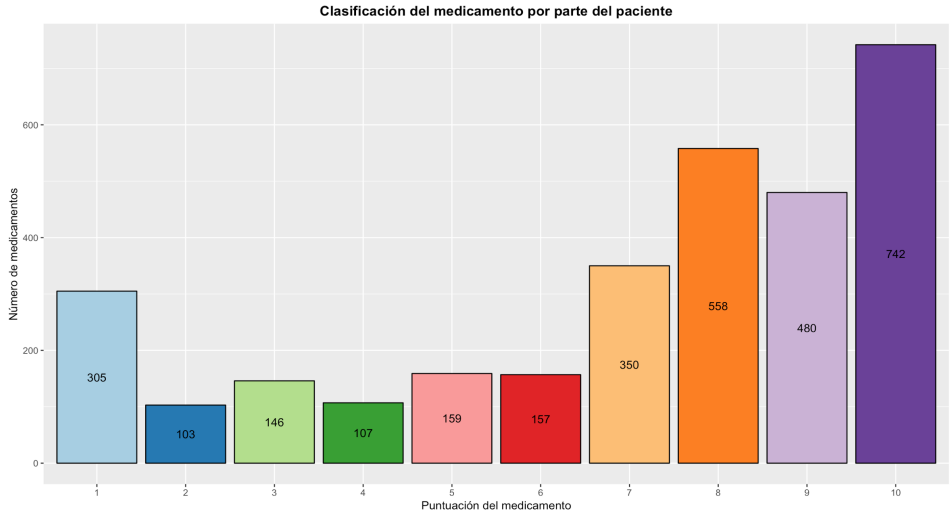


Figura 2: Clasificación del medicamento por parte del paciente

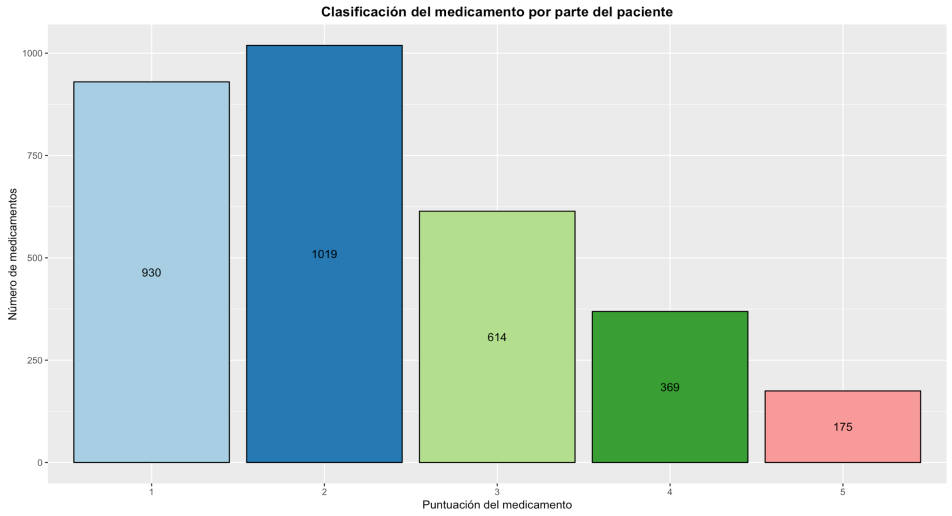


Figura 3: Clasificación de los efectos secundarios del medicamento según el paciente

Por último, vamos a visualizar gráficamente, la efectividad del medicamentoe según del paciente. En donde el 1 significa que el medicamento tiene poca efectividad y el 5 que tiene mucha efectividad. De acuerdo a la figura 4 y sabiendo que los pacientes aseguran que los medicamentos tienen pocos efectos secundarios, no es de extrañar que también tengan una alta efectividad.

Acabamos de comprobar, como las valoraciones de los medicamentos por parte de los pacientes son mayormente positivas. Por tanto, una vez comprendidos los datos y eliminadas las columnas anteriores y modificadas las necesarias, ya podemos continuar con el procesamiento de los datos. Para ello, lo primero tenemos que hacer es cargar la librería que procesa los datos de tipo texto en R, para la construcción y manipulación del corpus. La librería más conocida se llama **tm**, aunque también haremos uso del paquete **SnowballC** para realizar el *Stemming*.

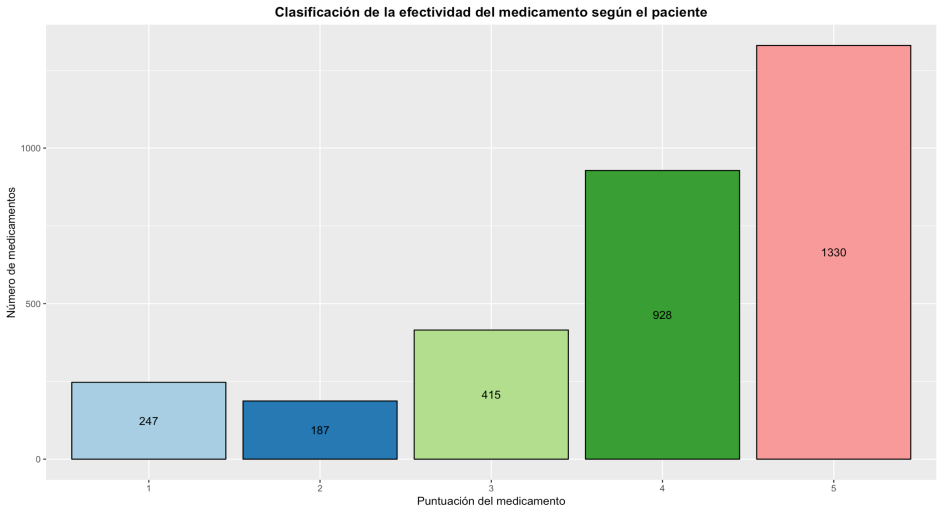


Figura 4: Clasificación de la efectividad del medicamento según el paciente

2.2.9. Creación del corpus

Para poder obtener la estructura con la que vamos a procesar nuestra información, debemos obtener un vector con documentos. En nuestro caso, cada uno de los documentos se corresponde con una opinión sobre un fármaco (*benefitsReview*) y los efectos que tiene (*sideEffectsReview*). Para ello, primero debemos de construir un vector con todas los opiniones del *dataframe* y convertir cada elemento del vector al formato de documento. Podemos usar la función *VectorSource* para hacer esta conversión. Se deberán realizar todas las modificaciones tanto para el conjunto train como test.

```
# Datos train

# Nos quedamos con la única columna del dataset que nos interesa.
# Necesitamos obtenerla en forma de vector, y no como un dataframe de una columna,
# por lo que usamos as.vector para hacer la conversión
benefits_train_review_data = as.vector(datos_train$benefitsReview)
effects_train_review_data = as.vector(datos_train$sideEffectsReview)

# Lo convertimos en la estructura de documento, y lo guardamos ya en el corpus
# que lo vamos a utilizar
benefits_train_corpus = (VectorSource(benefits_train_review_data))
effects_train_corpus = (VectorSource(effects_train_review_data))

# Creamos el propio corpus
benefits_train_corpus <- Corpus(benefits_train_corpus)
effects_train_corpus <- Corpus(effects_train_corpus)
```

```
# Datos test

# Nos quedamos con la única columna del dataset que nos interesa.
# Necesitamos obtenerla en forma de vector, y no como un dataframe de una columna,
# por lo que usamos as.vector para hacer la conversión
benefits_test_review_data = as.vector(datos_test$benefitsReview)
effects_test_review_data = as.vector(datos_test$sideEffectsReview)

# Lo convertimos en la estructura de documento, y lo guardamos ya en el corpus
```

```
# que lo vamos a utilizar
benefits_test_corpus = (VectorSource(benefits_test_review_data))
effects_test_corpus = (VectorSource(effects_test_review_data))

# Creamos el propio corpus
benefits_test_corpus <- Corpus(benefits_test_corpus)
effects_test_corpus <- Corpus(effects_test_corpus)
```

Podemos ver que funciona accediendo a uno cualquiera, de la forma `inspect(benefits_train_corpus[4])`:

```
<<SimpleCorpus>>
Metadata:  corpus specific: 1, document level (indexed): 0
Content:  documents: 1

[1] The acid reflux went away    months  just   days     drug. The heartburn started   soon  I stopped taking . So I began treatment . 6
months passed  I stopped taking . The heartburn came back,  seemed worse even. The doctor said I  try another 6 month treatment. I ,
exact thing happened. This went    three years. I asked    curing  reflux. The doctor quite frankly told      cure,   "treatment
symptoms". I  told  I  probably      rest   life.
```

Figura 5: Contenido para benefits_train_corpus I

O de la forma `benefits_train_corpus[[4]]$content`:

```
[1] "The acid reflux went away    months  just   days     drug. The heartburn started   soon  I stopped taking . So I began treatment . 6
months passed  I stopped taking . The heartburn came back,  seemed worse even. The doctor said I  try another 6 month treatment. I ,
exact thing happened. This went    three years. I asked    curing  reflux. The doctor quite frankly told      cure,   \"treatment
symptoms\". I  told  I  probably      rest   life."
```

Figura 6: Contenido para benefits_train_corpus II

Y si nos fijamos en el contenido, vemos que tiene signos de puntuación y exclamación.

2.2.10. Correlación

Una forma de medir la distancia es calcular la correlación entre un término y todos los demás de la matriz. Como en este caso, estamos usando variables textuales, no se aconseja hacer la correlación como tal. Debido a que la correlación indica la fuerza y la dirección de una relación lineal y proporcionalidad entre dos variables estadísticas, ya que está pensada para variables cuantitativas. Por otro lado, si que disponemos de variables categóricas, pero también se ha despreciado hacer la correlación entre ellas, debido a que no podemos prescindir de las etiquetas, ni agrupar distintos medicamentos en uno solo.

Además, la correlación está relacionada con el análisis de componentes principales (PCA), el cual, es una técnica utilizada para describir un conjunto de datos en términos de nuevas variables no correlacionadas. Por lo que se ha despreciado, de primeras realizar tal método.

2.2.11. Representación gráfica de las frecuencias del Corpus

Una vez creado el corpus y antes de aplicar las técnicas de preprocesamiento, vamos a visualizar las frecuencias para las dos columnas (*benefitsReview* y *sideEffectsReview*) de textos a las que vamos aplicar el preprocesamiento. Para ello, debemos calcular la matriz de términos y obtener los términos con mayor frecuencia.

Como se puede apreciar en la gráfica 7, los términos que más se repiten son *the* y *and*, además de otras preposiciones y conjunciones. Si se observa, las palabras que aparecen en la gráfica, no nos aportan información alguna, es por eso que vamos hacer una transformación a los términos, como la eliminación de los *stopwords*.

A continuación, mostramos los términos para la columna *sideEffects*, realizando el mismo procedimiento que antes. Y como se puede ver en la gráfica 8, obtenemos la misma conclusión que antes. En donde, tenemos palabras que no nos aportan nada de información: *very*, *the*, *that*, *and*...

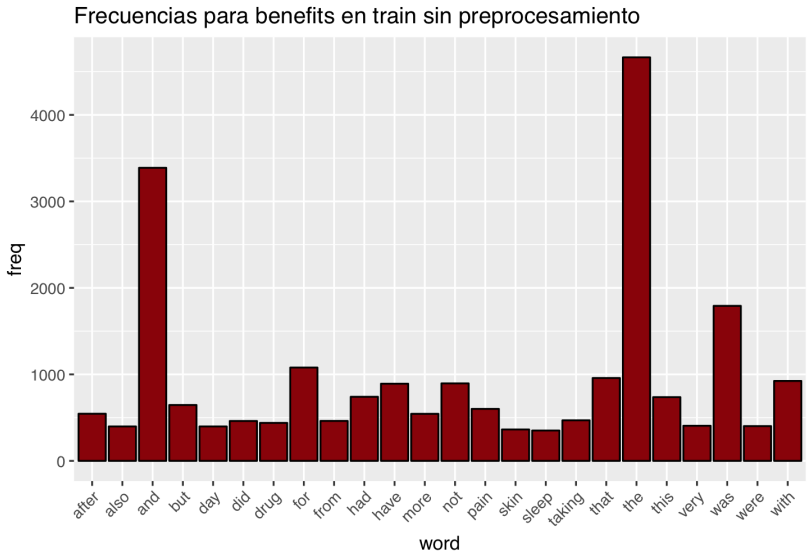


Figura 7: Frecuencia de términos para la columna benefitsReview

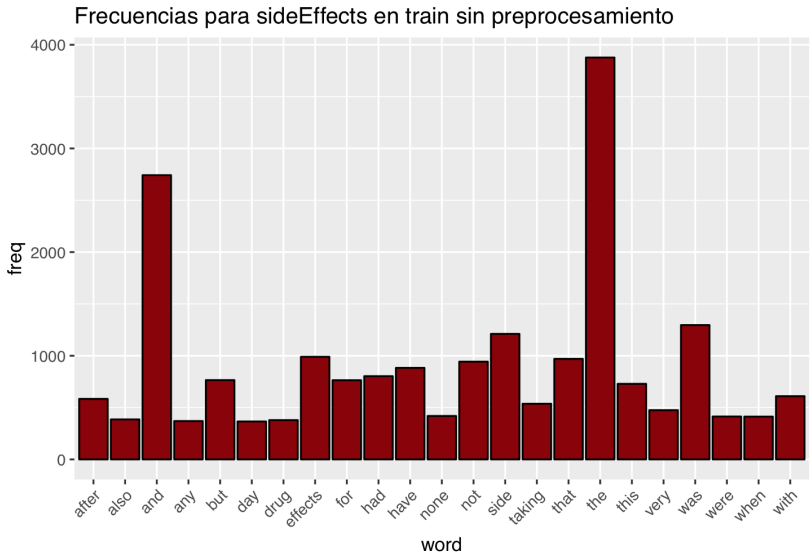


Figura 8: Frecuencia de términos para la columna sideEffectsReview

2.2.12. Eliminar signos de puntuación

Como hemos podido ver en el documento que se ha mostrado por pantalla, en él se aprecia el uso de signos de puntuación y exclamación. En un principio, no tiene sentido en *Data Mining* contemplar los signos de puntuación, ya que no nos van a aportar información. Por ello, los quitamos, como se puede ver a continuación. Con `tm_map(corpus, removePunctuation)`, se eliminan los símbolos: ! " $ % & ‘() * + , - . / : ; < = > ? @ [ ] ^ _’ { | } ~, tanto para train como test.

```
# Una vez que tenemos el corpus creado, continuamos con el procesamiento para datos train
benefits_train_corpus <- tm_map(benefits_train_corpus,
                                content_transformer(removePunctuation))
effects_train_corpus <- tm_map(effects_train_corpus,
                               content_transformer(removePunctuation))

# Una vez que tenemos el corpus creado, continuamos con el procesamiento para datos test
benefits_test_corpus <- tm_map(benefits_test_corpus,
                               content_transformer(removePunctuation))
effects_test_corpus <- tm_map(effects_test_corpus,
                              content_transformer(removePunctuation))
```

Si volvemos a mostrar la opinión número cuatro, vemos como todos los signos han desaparecido. De hecho, podemos inspeccionar el corpus, y se ve como todos los signos de puntuación, exclamación y derivados ya no están.

```
<<SimpleCorpus>>
Metadata:  corpus specific: 1, document level (indexed): 0
Content:  documents: 1

[1] The acid reflux went away for a few months after just a few days of being on the drug The heartburn started again as soon as I stopped
taking it So I began treatment again 6 months passed and I stopped taking it The heartburn came back and seemed worse even The doctor said
I should try another 6 month treatment I did and the same exact thing happened This went on for about three years I asked why this wasnt
curing my reflux The doctor quite frankly told me that it wasnt a cure but a treatment for the symptoms I was told that I would probably
be on it for the rest of my life
```

Figura 9: Contenido de benefits sin signos de puntuación

Ocurre lo mismo con el comentario de efectos número siete.

```
<<SimpleCorpus>>
Metadata:  corpus specific: 1, document level (indexed): 0
Content:  documents: 1

[1] a few experiences of nausiea heavy moodswings on the days I do not take it decreased appetite and some negative affect on my shortterm
memory
```

Figura 10: Contenido de effects sin signos de puntuación

2.2.13. Conversión de las mayúsculas en minúsculas

Para poder hacer uso de los términos por igual, debemos convertir las mayúsculas en minúsculas. Ya que normalmente se convierte en minúsculas todas las letras para que los comienzos de oración no sean tratados de manera diferente por los algoritmos, tanto para train como test.

```
# Datos train
benefits_train_corpus <- tm_map(benefits_train_corpus, content_transformer(tolower))
effects_train_corpus <- tm_map(effects_train_corpus, content_transformer(tolower))

# Datos test
benefits_test_corpus <- tm_map(benefits_test_corpus, content_transformer(tolower))
effects_test_corpus <- tm_map(effects_test_corpus, content_transformer(tolower))
```

Si volvemos a mostrar las opiniones, vemos como todas las mayúsuculas han desaparecido.

```
<<SimpleCorpus>>
Metadata:  corpus specific: 1, document level (indexed): 0
Content:  documents: 1

[1] the acid reflux went away for a few months after just a few days of being on the drug the heartburn started again as soon as i stopped
taking it so i began treatment again 6 months passed and i stopped taking it the heartburn came back and seemed worse even the doctor said
i should try another 6 month treatment i did and the same exact thing happened this went on for about three years i asked why this wasnt
curing my reflux the doctor quite frankly told me that it wasnt a cure but a treatment for the symptoms i was told that i would probably
be on it for the rest of my life
```

Figura 11: Contenido de benefits sin mayúsculas

```
<<SimpleCorpus>>
Metadata:  corpus specific: 1, document level (indexed): 0
Content:  documents: 1

[1] a few experiences of nausiea heavy moodswings on the days i do not take it decreased appetite and some negative affect on my shortterm
memory
```

Figura 12: Contenido de effects sin mayúsculas

2.2.14. Eliminación de Stopwords

En cualquier idioma, hay palabras que son tan comunes o muy utilizadas que no aportan información relevante, a dichas palabras se las conoce como *stopwords* o palabras *stop*. Por ejemplo, en español, las palabras "la", "a", "en", "de" son ejemplos de *stopwords*. Este tipo de palabras debemos de suprimirlas de nuestro corpus. Como, en nuestro caso, el contenido del corpus está en inglés, debemos especificar el idioma correcto para que nos elimine del corpus las palabras adecuadas en dicho idioma, tanto en train como en corpus.

```
# Datos train
benefits_train_corpus <- tm_map(benefits_train_corpus, content_transformer(removeWords),
                                stopwords("english"))
effects_train_corpus <- tm_map(effects_train_corpus, content_transformer(removeWords),
                               stopwords("english"))

# Datos test
benefits_test_corpus <- tm_map(benefits_test_corpus, content_transformer(removeWords),
                               stopwords("english"))
effects_test_corpus <- tm_map(effects_test_corpus, content_transformer(removeWords),
                              stopwords("english"))
```

Si volvemos a mostrar las opiniones, vemos como por ejemplo las palabras como *the* o *and*, han desaparecido de nuestro corpus.

```
<<SimpleCorpus>>
Metadata:  corpus specific: 1, document level (indexed): 0
Content:  documents: 1

[1]  acid reflux went away    months  just   days     drug  heartburn started   soon   stopped taking    began treatment  6 months passed
stopped taking   heartburn came back  seemed worse even  doctor said   try another 6 month treatment      exact thing happened  went
three years  asked   wasnt curing  reflux  doctor quite frankly told    wasnt  cure   treatment   symptoms   told    probably      rest
life
```

Figura 13: Contenido de benefits sin stopwords

Ahora ya hemos eliminado las *stopwords* de forma correcta, y pasamos a realizar la agrupación de sinónimos.

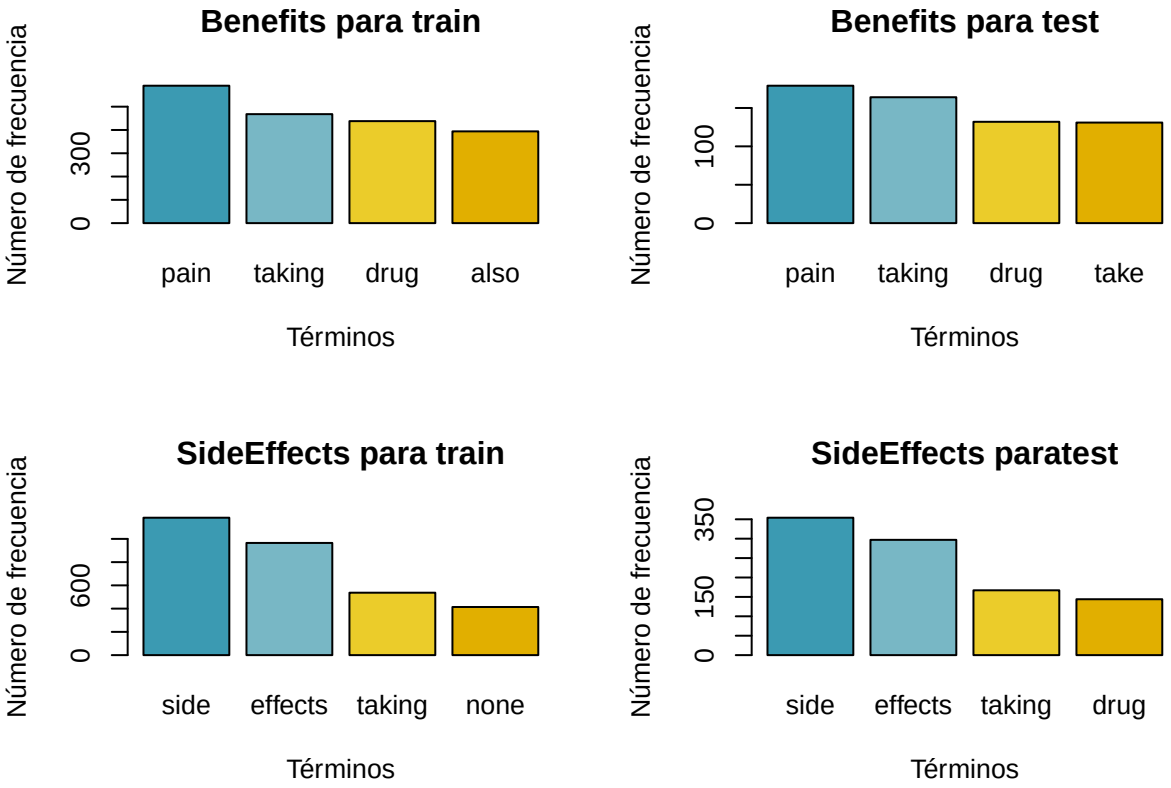
2.2.15. Agrupación de sinónimos

Con el fin de disminuir la dimensión del espacio a trabajar, se pueden identificar palabras distintas con el mismo significado y reemplazarlas por una sola palabra. Para ello se toman los sinónimos de dicha palabra.

```
<<SimpleCorpus>>
Metadata:  corpus specific: 1, document level (indexed): 0
Content:  documents: 1

[1]   experiences  nausiea heavy moodswings   days    take  decreased appetite   negative affect   shortterm memory
```

Figura 14: Contenido de effects sin stopwords

Dentro de las librerías que podemos usar para agrupar sinónimos, destacamos dos: `wordnet` y `rword2vec`. Sin embargo, por su sencillez se va hacer uso de `rword2vec`. Previamente, se obtendrán que palabras son las que mayor frecuencia presentan en nuestro texto, tanto para *benefitsReview* como *sideEffectsReview* del conjunto train y test. Y visualizamos los 4 primeros términos gráficamente para cada columna y conjunto train y test:



Una vez que tenemos los términos con mayor frecuencia en nuestras columnas (*benefitsReview* y *sideEffects-Review*) y su frecuencia asociada, pasamos a matriz dichos datos, con el fin de obtener solo las palabras y descartar su frecuencia.

Como ya sabemos las palabras a usar, es decir, los términos que más se repite, procedemos a la agrupación por sinónimos. En donde, mediante la función *distance(. . .)* de la libería *rword2vec*, obtendremos todas palabras más similares de nuestro conjunto, en nuestro caso nos vamos a quedar con las 2 primeras, tanto para *benefitsTrainReview* como *sideEffectsReview* del conjunto train y test. A continuación, se muestran los pasos seguidos.

1. Escribir un fichero los datos asociados a dicha columna.
2. Entrenat los datos del fichero, con el fin de obtener los vectores de palabras que nos darán los palabras más similares.
3. Obtener para cada término del documento, la distancia con los térmios del ficheros, quedándonos con las de mayor frecuencia.
4. Guardar en un fichero dichas distancias.

Una vez, que tenemos todas las palabras con los 2 términos más similares, procedemos a sustituir todos esos términos por el término general. Veamos un ejemplo sencillo, en donde las palabras “medicine” o “medication”,

van a ser sustituidas por "drug".

```
# Obtenemos el tercer término -> "drug"
terms_benefits_train_corpus[3]
```

```
## [1] "drug"
```

```
# Vamos a sustituir "pain" por sus dos palabras más similares
dist_terms_benefits_train_corpus_new[[3]]
```

```
## [1] medication medicine
## Levels: medication medicine
```

Se debe tener en cuenta, que los términos más similares han sido creados solo para nuestros documentos. Por último, ya solo nos queda hacer el reemplazamiento, para ello se usará la función *gsub(...)* sobre el corpus (*benefits_corpus* y *sideEffectsReview*). Para sustituir las palabras en el texto, se ha hecho uso de la función `gsub(pattern, replacement, x, ignore.case = FALSE, perl = FALSE, fixed = FALSE, useBytes = FALSE)`. A continuación, se describe el proceso seguido:

1. Iteramos sobre los términos del documento
2. Iteramos sobre las palabras más frecuentes de cada término
3. Realizamos el reemplazamiento de las palabras más similares por los términos generales, previa conversión a minúsculas.
4. Guardamos los resultados en ficheros.

2.2.16. TF-IDF

Para estudiar la importancia de los términos de un documento en particular, en lugar de utilizar la frecuencia de cada uno de los términos directamente, se pueden utilizar diferentes ponderaciones denominadas TF-IDF (*Term Frequency-Inverse Document Frecuency*). Estas ponderaciones se calculan como el producto de dos medidas, la frecuencia de aparición del término (tf) y la frecuencia inversa del documento (idf). La fórmula matemática para esta métrica es la siguiente:

$$tfidf(t, d, D) = tf(t, d)Öidf(t, D)$$

donde t es el término, d denota cada documento, D el espacio total de documentos y $tfidf$ es el peso asignado a ese término en el documento correspondiente.

La combinación de los valores de tf e idf da una métrica que permite saber cómo de únicas son las palabras de un documento. La ponderación asigna un alto peso a un término si se produce con frecuencia en ese documento, pero rara vez en la colección completa. Sin embargo, si el término ocurre pocas veces en el documento, o aparece prácticamente en todos ellos, disminuye el peso asignado por la ponderación $tfidf$.

El peso aumenta proporcionalmente al número de veces que una palabra aparece en el documento, pero es compensada por la frecuencia de la palabra en la colección de documentos, lo que permite filtrar las palabras más comunes. Para ello, necesitamos que convertir nuestro corpus a un *dataframe*, en donde cada columna será una palabra del comentario y cada fila un comentario.

Frecuencia del término

La primera parte de la fórmula $tf(t, d)$ es simplemente calcular el número de veces que aparece cada palabra en cada documento:

1. Creamos el corpus utilizando el vector de string que hemos creado previamente.
2. Generamos la matriz de términos
3. Convertimos a matriz.

Frecuencia inversa del documento

Durante el cálculo de la frecuencia del término se considera que todos los términos tienen igual importancia, no obstante, se conocen casos en los que ciertos términos pueden aparecer muchas veces pero tienen poca importancia. Esta segunda parte de la fórmula completa el análisis de evaluación de los términos y actúa como corrector de tf. Usando la matriz de frecuencia de términos, el peso idf se puede calcular de la siguiente forma:

```
# Calculamos los pesos asociados a cada término en train
terms_benefits_train <- ( idf_benefits_train <- log( ncol(tf_benefits_train)
                                                     / ( 1+rowSums(tf_benefits_train != 0))))
terms_effects_train <- ( idf_effects_train <- log( ncol(tf_effects_train)
                                                   / ( 1+rowSums(tf_effects_train != 0))))

# Calculamos los pesos asociados a cada término en test
terms_benefits_test <- ( idf_benefits_test <- log( ncol(tf_benefits_test)
                                                   / ( 1+rowSums(tf_benefits_test != 0))))
terms_effects_test <- ( idf_effects_test <- log( ncol(tf_effects_test)
                                                 / ( 1+rowSums(tf_effects_test != 0))))

# Muestra los pesos asociados a cada término (los 5 primeros)
terms_benefits_train[1:5]
```

```
##      agents       alone  congestive dysfunction      failur
##    6.941835    5.044715    6.654153    6.941835    7.347300
```

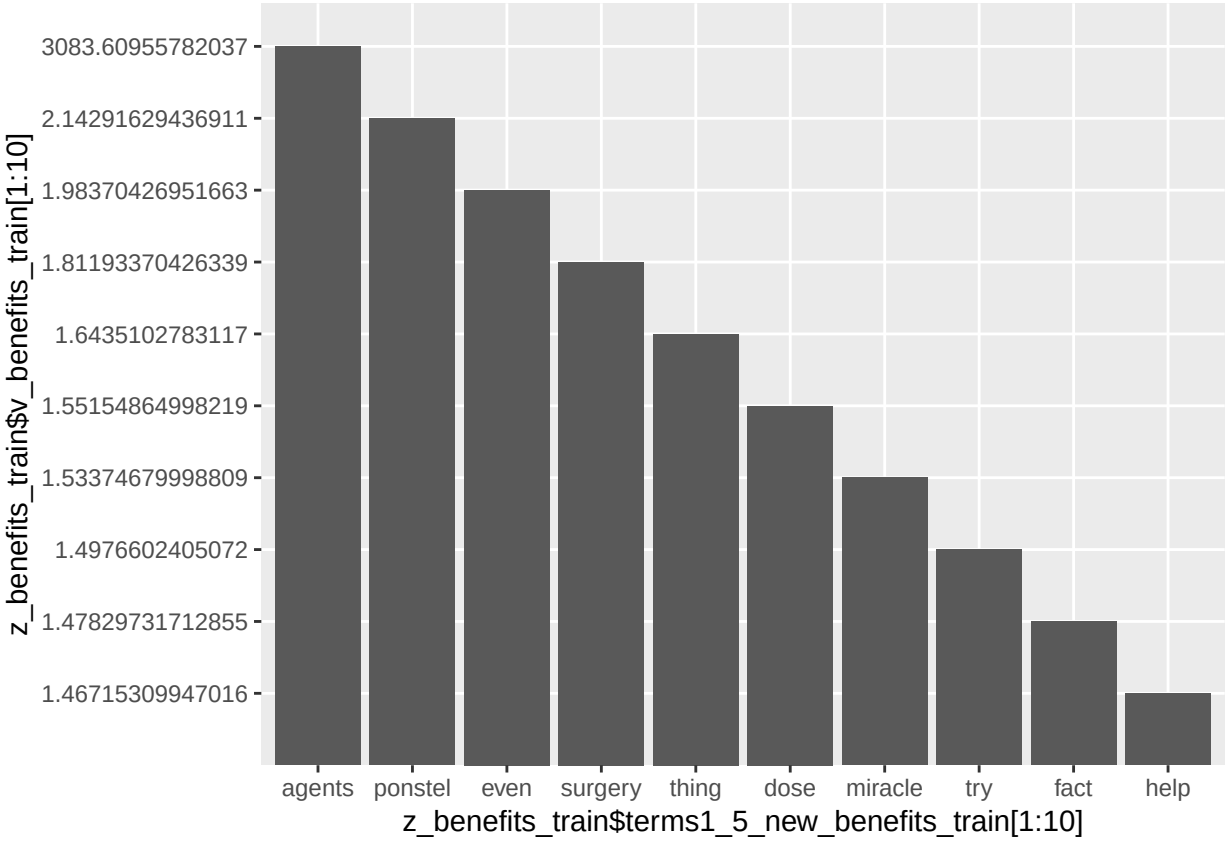
Ahora que tenemos nuestra matriz con el término frecuencia y el peso idf, estamos listos para calcular el peso total de $tf - idf$. Para hacer esta multiplicación de matrices, también tendremos que transformar el vector idf en una matriz diagonal. Ambos cálculos se muestran a continuación.

1. Creamos la matriz diagonal: `( idf_benefits_train <- diag(idf_benefits_train) )`
2. Hacemos la operación para calcular tf_idf como el producto de $td \cdot idf$: `tf_idf_benefits_train<-crossprod(tf_benefits_train, idf_benefits_train)`
3. Guardamos los datos en ficheros, con el fin de reducir el tiempo de procesamiento.

Hay que recordar que en la sección tf (frecuencia del término), estamos representando cada término como el número de veces que aparecieron en el documento. El principal problema para esta representación es que creará un sesgo hacia documentos largos, ya que un término dado tiene más posibilidades de aparecer en documentos más largos, lo que los hace parecer más importantes de lo que realmente son. Por lo tanto, el enfoque para resolver este problema es la buena normalización:

$$t\_benefits\_train = \frac{tf\_idf\_benefits\_train\_new}{\sqrt{(rowSums(tf\_idf\_benefits\_train\_new^2))}}$$

```
# Graficamos los resultados para benefits train
ggplot() +
 geom_bar(data=z_benefits_train[1:10,],aes(x=z_benefits_train$terms1_5_new_benefits_train[1:10],
                                           y=z_benefits_train$v_benefits_train[1:10]), stat='identity',
          position='dodge')
```



2.2.17. Stemming

El siguiente paso consiste en reducir el número de palabras totales con las que estamos trabajando. En este caso, se trata de reducir aquellas que no nos aportan nada relevante a lo que ya tenemos. En la columna con la que estamos trabajando en este dataframe, se repite una gran cantidad de veces la palabra "benefit", al igual que "benefits".

Sin embargo, realizar el análisis de nuestros datos con ambas palabras no tiene gran relevancia, ya que una no aporta nada respecto a la otra. Este es un ejemplo del tipo de casos que se nos dan en nuestro dataset. Igual ocurre con "reduce" y "reduced", por ejemplo. Este tipo de situaciones son las que intentamos corregir con este paso. Vamos a ver un ejemplo de este suceso, que se da por ejemplo en los siguientes valores del corpus (y en muchos más).

```
inspect(benefits_train_corpus_new_load[183])
```

```
## <<SimpleCorpus>>
## Metadata:  corpus specific: 1, document level (indexed): 0
## Content:  documents: 1
##
## [1]  treatment benefits   temporary    made sneezing  watery eyes diminish    address  issue  respira
```

```
inspect(benefits_train_corpus_new_load[213])
```

```
## <<SimpleCorpus>>
## Metadata:  corpus specific: 1, document level (indexed): 0
## Content:  documents: 1
##
## [1] overall   ease mantally    true benefit  felt
```

A continuación, aplicamos el proceso de stemming mediante la siguiente orden:

```
benefits_train_corpus_new_load <- tm_map(benefits_train_corpus_new_load, stemDocument)
effects_train_corpus_new_load <- tm_map(effects_train_corpus_new_load, stemDocument)

benefits_test_corpus_new_load <- tm_map(benefits_test_corpus_new_load, stemDocument)
effects_test_corpus_new_load <- tm_map(effects_test_corpus_new_load, stemDocument)
```

Si ahora volvemos a mostrar el contenido de dichas opiniones, podemos ver que el stemming se ha hecho efectivo: donde ponía *benefits*, ahora pone *benefit*, como se puede comprobar si volvemos a mostrar dichos elementos del corpus. De hecho, si nos fijamos, no solo esta palabra ha resultado modificada, sino que se han resumido muchas más palabras en comparación a como teníamos los documentos en el momento previo a la aplicación del método *Stem*. Desde este momento, ya tenemos nuestro conjunto reducido a nivel de concepto.

```
inspect(benefits_train_corpus_new_load[183])
```

```
## <<SimpleCorpus>>
## Metadata:  corpus specific: 1, document level (indexed): 0
## Content:  documents: 1
##
## [1] treatment benefit temporari made sneez wateri eye diminish address issu respiratori difficulti f
```

```
inspect(benefits_train_corpus_new_load[213])
```

```
## <<SimpleCorpus>>
## Metadata:  corpus specific: 1, document level (indexed): 0
## Content:  documents: 1
##
## [1] overal eas mantal true benefit felt
```

2.2.18. Valores perdidos

Tras el proceso de limpieza anterior en un volumen tan grande de datos cabe esperar que algún documento estuviera formado por tan solo palabras vacías, enlaces o combinaciones de estos, es por ello, que por medio de filtrado básico de R se obtienen aquellos que no contienen ninguna palabra y se elimina del conjunto del dataset para evitar problemas en los procesos posteriores.

```
# https://github.com/joseangeldiazg/twitter-text-mining/blob/master/ner.R
# Localizamos posibles valores perdidos que se hayan generado tras el proceso de limpieza
which(benefits_train_corpus_new_load$content=="")
```

```
## integer(0)
```

```
which(effects_train_corpus_new_load$content==" ")
```

```
## integer(0)
```

```
which(benefits_test_corpus_new_load$content=="  ")
```

```
## integer(0)
```

```
which(effects_test_corpus_new_load$content=="   ")
```

```
## integer(0)
```

```
# Vemos que no hay muchos vacios
benefits_train_corpus_new_load<-benefits_train_corpus_new_load[which(benefits_train_corpus_new_load$con
effects_train_corpus_new_load<-effects_train_corpus_new_load[which(effects_train_corpus_new_load$conten
```

```
benefits_test_corpus_new_load<-benefits_test_corpus_new_load[which(benefits_test_corpus_new_load$conten
effects_test_corpus_new_load<-effects_test_corpus_new_load[which(effects_test_corpus_new_load$content!=
```

2.2.19. Borrar espacios en blanco innecesarios

Hasta el momento hemos hecho distintos cambios en el texto de nuestro dataset. No solo hemos modificado algunas palabras, sino que también hemos borrado otras muchas. Por ello, es adecuado asegurarnos de que no hay más espacios en blanco que los que separan las palabras del texto. Para asegurarnos de ello, podemos ejecutar la siguiente orden, que se encarga de suprimir los espacios en blanco sobrantes.

```
benefits_train_corpus_new_load <- tm_map(benefits_train_corpus_new_load, stripWhitespace)
effects_train_corpus_new_load <- tm_map(effects_train_corpus_new_load, stripWhitespace)

benefits_test_corpus_new_load <- tm_map(benefits_test_corpus_new_load, stripWhitespace)
effects_test_corpus_new_load <- tm_map(effects_test_corpus_new_load, stripWhitespace)
```

2.2.20. Sparsity

También puede resultar muy útil eliminar los términos que aparecen en muy pocos documentos antes de proceder a la clasificación. El motivo principal es la factibilidad computacional, ya que este proceso reduce drásticamente el tamaño de la matriz sin perder información significativa. Además puede eliminar errores en los datos, como podrían ser palabras mal escritas. Para suprimir estos términos, denominados escasos, se utiliza el comando `removeSparseTerms()`.

$$df(t) > N\mathring{u}(1! = sparse)$$

siendo df la frecuencia de documentos del término t y N el número de vectores. El parámetro sparse toma valores entre 0 y 1. En este caso, el umbral de escasez es 0.999, se toman los términos que aparecen en más del 1 % de documentos.

```
dtm <- DocumentTermMatrix(benefits_train_corpus_new_load)
inspect(dtm)
```

```
## <<DocumentTermMatrix (documents: 3104, terms: 5835)>>
## Non-/sparse entries: 52316/18059524
## Sparsity           : 100%
## Maximal term length: 33
## Weighting          : term frequency (tf)
## Sample             :
##       Terms
## Docs   day drug effect feel help medic pain take time work
##   1049   4    0      0    0    0     0    5    1    1    1
##   1189   4    0      0    0    0     0    5    1    1    1
##   1279   0    0      1    3    2     0    0    1    1    1
##   1824   1    0      0    3    1     0    0    1    2    0
##   2504   4    1      3    0    1     4    0    2    0    3
##   317    7    2      0    1    1     1    0    3    4    0
##   339    1    1      5    1    2     6    0    2    0    1
##   462    4    1      3    0    1     4    0    2    0    3
##   565    3    2      2    3    0     0    6    4    0    1
##   665    3    4      0    3    0     0    0    1    1    0
```

```
dtm <- removeSparseTerms(dtm, sparse=0.999)
inspect(dtm)
```

```
## <<DocumentTermMatrix (documents: 3104, terms: 1702)>>
## Non-/sparse entries: 46811/5236197
## Sparsity           : 99%
## Maximal term length: 17
## Weighting          : term frequency (tf)
## Sample             :
##       Terms
## Docs   day drug effect feel help medic pain take time work
##   1049   4    0      0    0    0     0    5    1    1    1
##   1189   4    0      0    0    0     0    5    1    1    1
##   1279   0    0      1    3    2     0    0    1    1    1
##   1824   1    0      0    3    1     0    0    1    2    0
##   2504   4    1      3    0    1     4    0    2    0    3
##   317    7    2      0    1    1     1    0    3    4    0
##   339    1    1      5    1    2     6    0    2    0    1
##   462    4    1      3    0    1     4    0    2    0    3
##   565    3    2      2    3    0     0    6    4    0    1
##   665    3    4      0    3    0     0    0    1    1    0
```

Se observa como de x términos pasamos a x términos, ni un 5 % del total, por tanto se ha considerado despreciar aplicar esta técnica. Por tanto, ya tenemos nuestros datos listos.

2.2.21. Matriz de documentos de los términos

Ahora vamos a mapear nuestro corpus creando una matriz de términos, donde las filas corresponden a los documentos y las columnas a los términos. Para ello usaremos la función TermDocumentMatrix:

```
matrix_corpus <- TermDocumentMatrix(benefits_train_corpus_new_load)
```

Podemos observar que tenemos 5838 términos, esto quiere decir que tenemos 5838 palabras diferentes en nuestro Corpus. Obtengamos la *frecuencia de las palabras*:

```
class(matrix_corpus)
```

```
## [1] "TermDocumentMatrix"    "simple_triplet_matrix"
```

Como podemos ver, actualmente aún no tenemos nuestros datos en la matriz que buscamos, sino en un vector, por tanto:

```
matrix_corpus <- as.matrix(matrix_corpus)
class(matrix_corpus)
```

```
## [1] "matrix"
```

```
dim(matrix_corpus)
```

```
## [1] 5835 3104
```

Con este método, hemos obtenido la ocurrencia de las palabras que tenemos en nuestro dataset para cada uno de los documentos/comentarios. Esta matriz tiene 5838 columnas, que representa la totalidad de palabras diferentes que hay en los comentarios de la columna *benefitsReview*, y 3107 filas, donde cada una representa un comentario. Por tanto, en la fila iésima la matriz, tendremos la ocurrencia de las palabras en *benefitsReview* que existen en el comentario *i*.

```
# Sumamos las filas
suma_matrix_corpus <- rowSums(matrix_corpus)
head(suma_matrix_corpus,5)
```

```
##    agent     alon  congest dysfunct   failur
##        2       19       19        2        7
```

```
# Ordenanamos de mayor a menor y muestra los 10 primeros
ordena_mayor_matrix_corpus <- sort(suma_matrix_corpus, decreasing = TRUE)
head(ordena_mayor_matrix_corpus,10)
```

```
##   take effect   pain    day   help   drug   feel   time   work  medic
##    789    682    642    626    524    498    479    450    404    399
```

```
copia_ordena_mayor = ordena_mayor_matrix_corpus # Para graficos (evitando data.frame)

# Ordenanamos de menor a mayor y muestra los 10 primeros
ordena_menor_matrix_corpus <- sort(suma_matrix_corpus, decreasing = FALSE)
head(ordena_menor_matrix_corpus,10)
```

```
##         mangag          overt    ventricular            con           pros
##              1              1              1              1              1
##        ponstel          frank       valerian allergiesirrit          dryer
##              1              1              1              1              1
```

```
# Transformamos a objeto data.frame, con dos columnas (palabra, frec), para posteriormente graficarlo.
ordena_mayor_matrix_corpus <- data.frame(palabra = names(ordena_mayor_matrix_corpus), frec = ordena_may
```

Mostramos las más frecuentes:

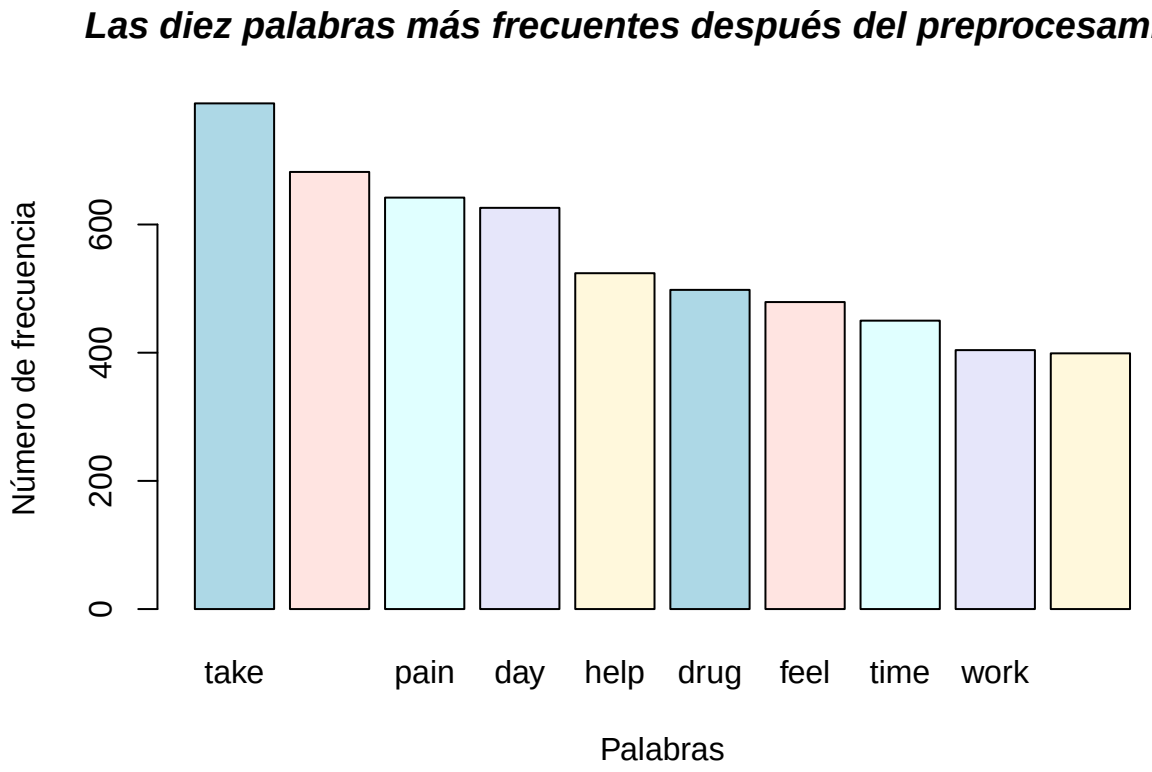
```
ordena_mayor_matrix_corpus[1:20,]
```

```
##             palabra frec
## take           take  789
## effect       effect  682
## pain           pain  642
## day             day  626
## help           help  524
## drug           drug  498
## feel           feel  479
## time           time  450
## work           work  404
## medic         medic  399
## also           also  394
## use             use  381
## sleep         sleep  381
## reduc         reduc  381
## year           year  369
## get             get  368
## skin           skin  358
## treatment treatment  357
## benefit     benefit  353
## depress     depress  349
```

Y obtenemos la *gráfica*:

```
copia_ordena_mayor <- as.matrix(copia_ordena_mayor)
barplot(copia_ordena_mayor[1:10,],  xlab="Palabras", ylab="Número de frecuencia",
```

```
        col = c("lightblue", "mistyrose", "lightcyan",
                "lavender", "cornsilk"))
title(main = list("Las diez palabras más frecuentes después del preprocesamiento", font = 4))
```



2.2.22. Nube de palabras

Por último, vamos a visualizar la nube de palabras para benefits preprocesado, tanto al principio del procesamiento como al final.

2.2.23. Convertir a dataframe nuestras modificaciones

Por último, con el fin de mejorar los tiempos computacionales y poder hacer uso todos los integrantes del grupo de las columnas preprocesadas, se ha optado por añadir dichos cambios en el dataset y guardarlos en un fichero.

Nota: Cuando se necesite en las sucesivas técnicas, se realizarán las transformaciones necesarias de acuerdo a cada técnica en cuestión.

Lectura de datos

```
# Cargamos los tados
datos_train <- read.table("datos/datos_train_preprocesado.csv", sep=",", comment.char="",quote = "\"", h

datos_test <- read.table("datos/datos_test_preprocesado.csv", sep=",", comment.char="",quote = "\"", hea
# Establecemos la semilla
```

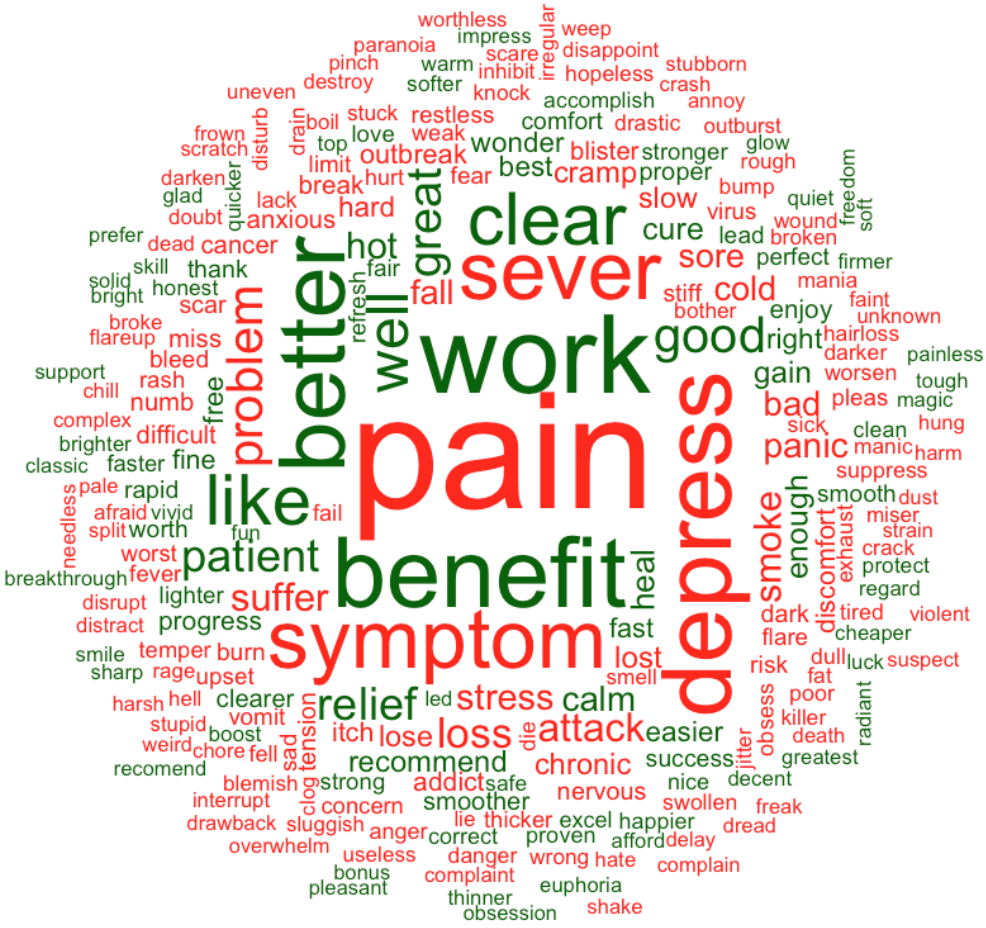


Figura 15: Contenido de benefits sin stopwords

Análisis exploratorio de los datos

El análisis exploratorio de datos o (EDA) engloba un conjunto de técnicas para poder comprender de manera rápida la naturaleza de una colección de datos o dataset.

Se basa principalmente en dos criterios: las **estadísticas de resumen** y la **visualización de datos**.

En primer lugar, vamos a realizar un resumen de nuestros datos utilizando la función `summary`. Dicha función nos mostrará información relevante para cada una de las columnas del datataset, mostrando información general como valores mínimos, máximos, media, mediana..

El resultado que obtenemos al evaluar nuestro dataset es el siguiente:

```
summary(datos_train)
```

```
##    urlDrugName       rating                    effectiveness
##  lexapro :  63   Min.   : 1.000   Considerably Effective: 926
##  prozac  :  46   1st Qu.: 5.000   Highly Effective      :1330
##  retin-a :  45   Median : 8.000   Ineffective           : 247
```

```
##  zoloft  :  45   Mean   : 7.008   Marginally Effective  : 186
##  paxil   :  38   3rd Qu.: 9.000   Moderately Effective  : 415
##  propecia:  38   Max.   :10.000
##  (Other) :2829
##                          sideEffects                 condition
##  Extremely Severe Side Effects: 175   depression         : 236
##  Mild Side Effects            :1019   acne               : 165
##  Moderate Side Effects        : 612   anxiety            :  63
##  No Side Effects              : 930   insomnia           :  54
##  Severe Side Effects          : 368   birth control      :  49
##                                       high blood pressure:  42
##                                       (Other)            :2495
##
##  none
##  None
##  NONE
##  None.
##  The treatment benefits were marginal at best.  Mood neither improved nor deteriorated, and anxiety u
##  Before the use of vagifem tablets, I had to endure a series of urinary infections after sometimes pa
##  (Other)
##                    sideEffectsReview sideEffectsNumber effectivenessNumber
##  none                         : 112    Min.   :1.000     Min.   :1.000
##  None                         :  73    1st Qu.:1.000     1st Qu.:3.000
##  None.                        :  19    Median :2.000     Median :4.000
##  No side effects.             :   9    Mean   :2.304     Mean   :3.936
##  There were no side effects.:   6    3rd Qu.:3.000     3rd Qu.:5.000
##  no side effects              :   5    Max.   :5.000     Max.   :5.000
##  (Other)                      :2880
##  weightedRating    ratingLabel     sideEffectsInverse
##  Min.   : 2.000   Min.   :0.0000   Min.   :1.000
##  1st Qu.: 6.000   1st Qu.:1.0000   1st Qu.:3.000
##  Median : 8.000   Median :1.0000   Median :4.000
##  Mean   : 7.737   Mean   :0.7874   Mean   :3.696
##  3rd Qu.:10.000   3rd Qu.:1.0000   3rd Qu.:5.000
##  Max.   :10.000   Max.   :1.0000   Max.   :5.000
##
##
##  none
##  lower blood pressur
##  prevent pregnanc
##  treatment benefit margin best mood neither improv deterior anxieti never signific allevi unsurpris i
##  abl work without hassl im free pain right now there need cri anymor whenev urin
##  believ multipl treatment benefit take tylenol headach tylenol safe inexpens requir physician prescri
##  (Other)
##         effects_preprocesado
##  none              : 214
##  side effect       :  51
##  none notic        :  17
##  none awar         :   8
##  notic side effect:   7
##  none can tell     :   6
##  (Other)           :2801
```

A continuación se va a realizar un análisis de la información más relevante no textual, como el valor de

rating de los usuarios, la **efectividad** y los **efectos secundarios** de dicho medicamento y por último, la **valoración ponderada del rating** teniendo en cuenta la proporción entre efectividad y efectos secundarios del medicamento.

Valoraciones de los medicamentos por parte de los usuarios.

En primer lugar vamos a analizar si el *rating* aportado por los usuarios sobre los medicamentos son buenos o no.

Empezamos obteniendo las frecuencias y porcentaje total de las valoraciones aportadas por los usuarios. Para ello se va a calcular la frecuencia de dicho atributo y su porcentaje respecto del total.

```
# Obtener frecuencias del rating
table(datos_train$rating)
```

```
##
##   1   2   3   4   5   6   7   8   9  10
## 305 102 146 107 158 157 349 558 480 742
```

```
# Calculamos el número de documentos
numDocuments <- dim(datos_train)[1]

# Calculamos el porcentaje de cada puntuación respecto del total.
table(datos_train$rating)/numDocuments
```

```
##
##          1          2          3          4          5          6
## 0.09826031 0.03286082 0.04703608 0.03447165 0.05090206 0.05057990
##          7          8          9         10
## 0.11243557 0.17976804 0.15463918 0.23904639
```

Como podemos observar, hay una mayoría de valoraciones positivas respecto a las negativas. De hecho el mayor porcentaje (casi el 24 %) tienen la máxima valoración.

Podemos comprobar ésto mediante el uso de la moda.

```
# Función para calcular la moda. Se le pasa como parámetro un atributo
calcularModa<-function(var){
  frec.var<-table(var)
  valor<-which(frec.var==max(frec.var))  # Elementos con el valor m
  names(valor)
}

# Obtenemos la moda para el rating
calcularModa(datos_train$rating)
```

```
## [1] "10"
```

Como resumen en general del rating, se va a calcular la media y la mediana para calcular la tendencia central para dicha variable.

La media es la siguiente:

```
# Media
mean(datos_train$rating)
```

```
## [1] 7.008376
```

La mediana es la siguiente:

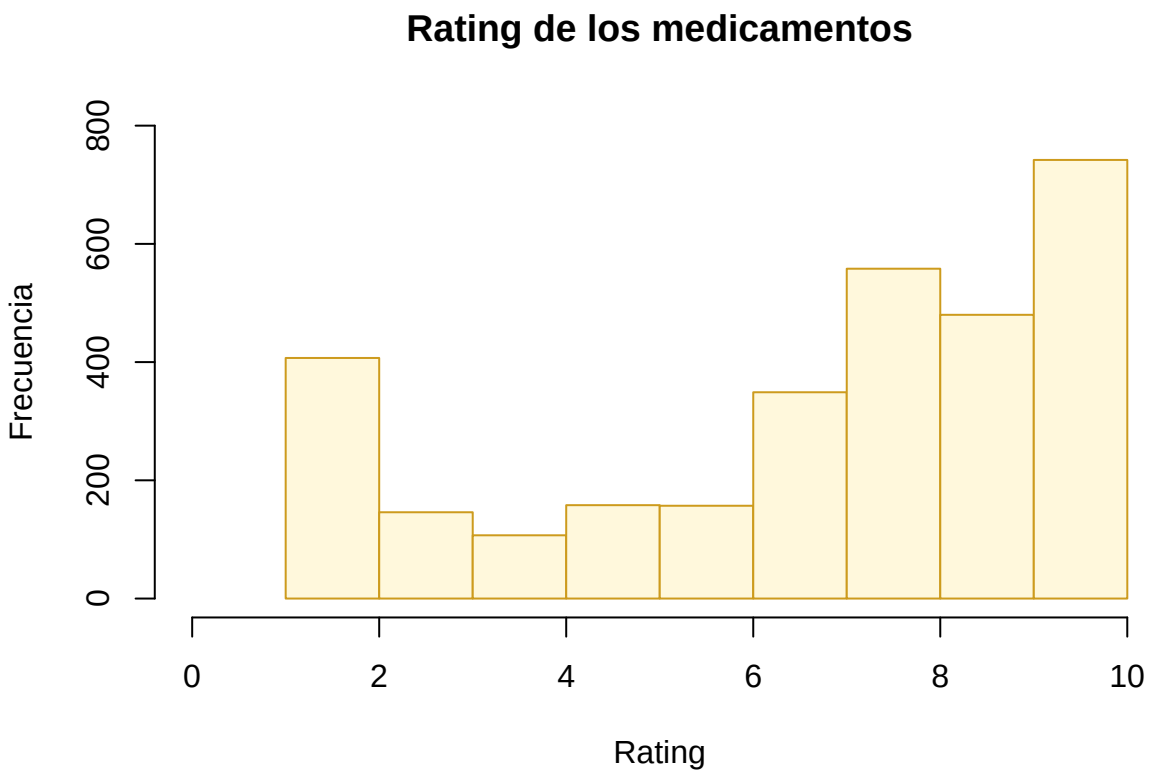
```r
# Mediana
median(datos_train$rating)
```

```
## [1] 8
```

El valor obtenido es medio obtenido es 7 y la mediana es 8. Podemos concluir con dicha información, que en general las valoraciones sobre los medicamentos son bastante positivas, situándose el 50 % de dichas valoraciones en el valor 8.

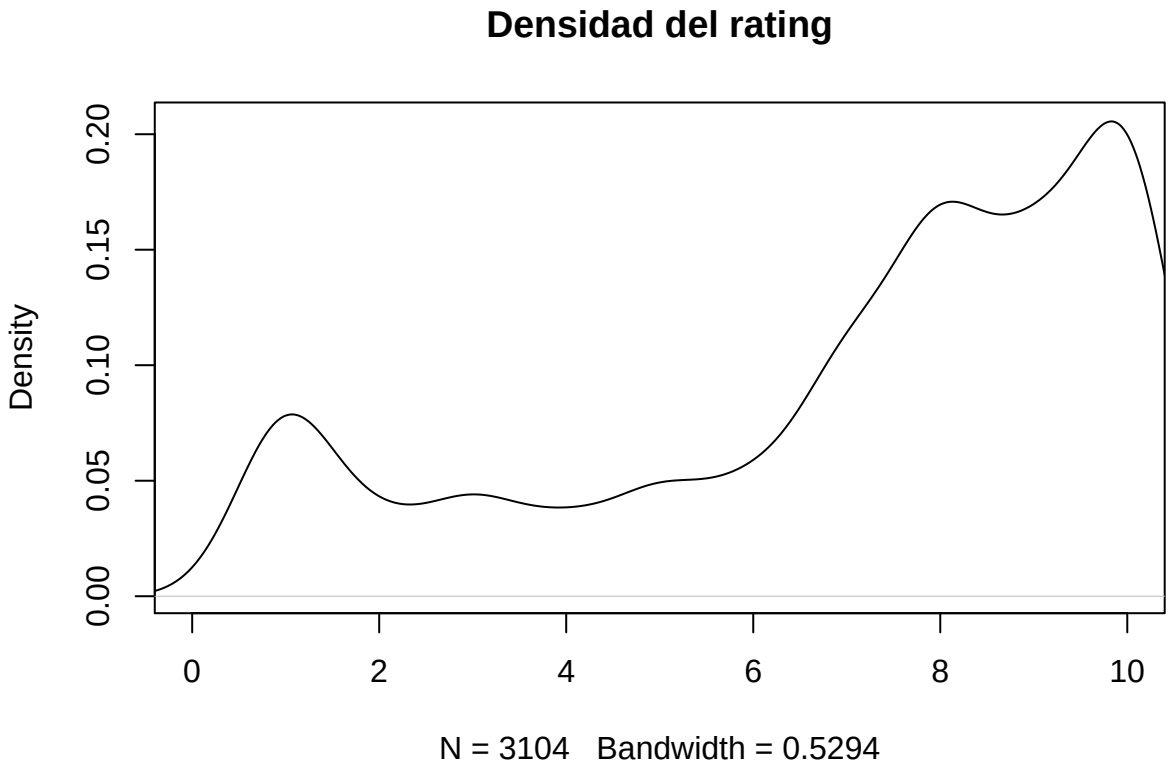
A continuación, se va a visualizar dicha información gráficamente:

```r
# Histograma de la valoración dada por los usuarios sobre los medicamentos

ratingExploration <- datos_train$rating

hist(ratingExploration,
     main="Rating de los medicamentos",
     xlab="Rating",
     ylab="Frecuencia",
     border="goldenrod3",
     xlim=c(0,10),
     ylim=c(0,800),
     col= "cornsilk",
     breaks=10,


    )
```

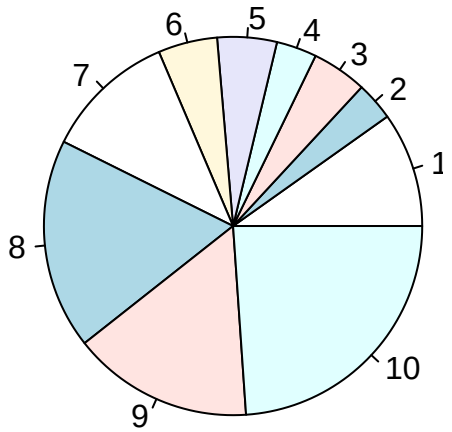



```r
# Diagrama de densidad de la valoración dada por los usuarios sobre los medicamentos

plot(density(ratingExploration),
```

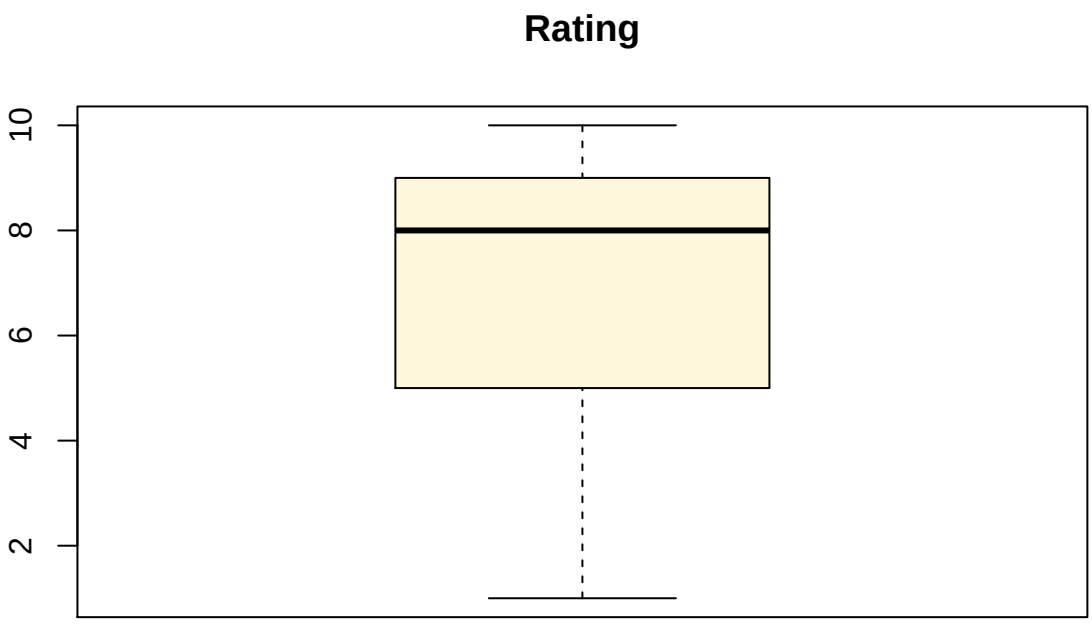
```
     main="Densidad del rating",
     xlim=c(0,10),
     )
```




```
# Diagrama de sectores de las valoraciones dadas por los usuarios
pie(table(datos_train$rating))
```




```
# Diagrama de cajas sobre las valoraciones dadas por los usuarios
boxplot(datos_train$rating,main="Rating", col= "cornsilk" )
```



Como medidas de dispersión, se va a calcular la **desviación típica**:

```
# Desviación típica
sd(datos_train$rating)
```

```
## [1] 2.937406
```

Como se puede observar, la desviación típica nos da un valor de 2.93. Esto quiere decir que los valores no están concentrados en un único valor, sino que la mayoría se sitúan en el un intervalo con distancia 3 despecto de la media.

Este valor concuerda, puesto que si observamos el histograma anterior, vemos que la mayoría de las puntuaciones se sitúan entre 5 y 10.

Esto también nos da como **conclusión** que en general las **opiniones** sobre los medicamentos **son buenas**, puesto que la mayor cantidad se sitúan en el intervalo [5.10].

Efectividad del medicamento

En esta sección, se va a analizar si se consideran que los medicamentos son efectivos o no. Para ello se va a analizar el atributo **effectivenessNumber** (que mide la efectividad del medicamento, siendo 1 menos efectivo y 5 más afectivo)

Empezamos obteniendo las frecuencias y porcentaje total de las anotaciones de efectividad. Para ello se va a calcular la frecuencia de dicho atributo y su porcentaje respecto del total.

```
# Obtener frecuencias del efectivenessNumber
table(datos_train$effectivenessNumber)
```

```
##
##    1    2    3    4    5
##  247  186  415  926 1330
```

```
# Calculamos el número de documentos
numDocuments <- dim(datos_train)[1]

# Calculamos el porcentaje de cada valor de efectividad respecto del total.
table(datos_train$effectivenessNumber)/numDocuments
```

```
##
##          1          2          3          4          5
## 0.07957474 0.05992268 0.13369845 0.29832474 0.42847938
```

Como podemos observar, la mayoría de los medicamentos se consideran que son efectivos. De hecho, la mayoría de los medicamentos se consideran altamente efectivos (con un 42 %)

Podemos comprobar ésto mediante el uso de la moda.

```
# Obtenemos la moda para el efectivenessNumber
calcularModa(datos_train$effectivenessNumber)
```

```
## [1] "5"
```

Como resumen en general de la efectividad, se va a calcular la media y la mediana para calcular la tendencia central para dicha variable.

La media es la siguiente:

```
# Media
mean(datos_train$effectivenessNumber)
```

```
## [1] 3.936211
```

La mediana es la siguiente:

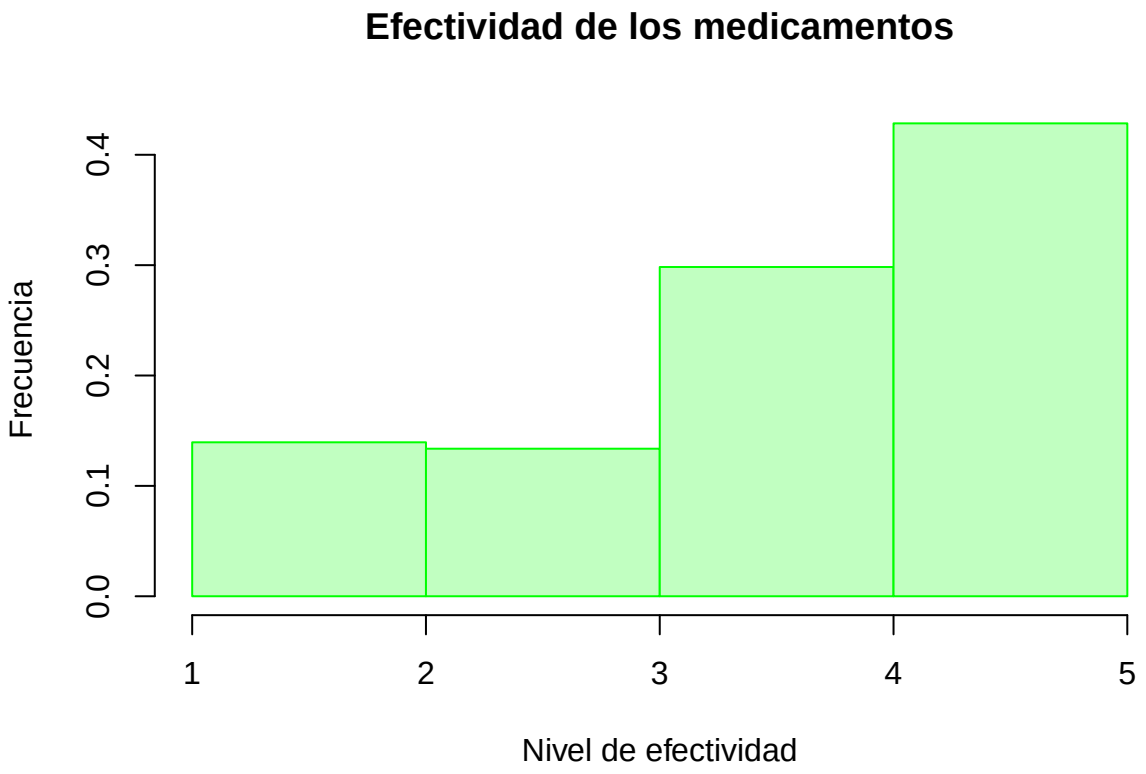
```
# Mediana
median(datos_train$effectivenessNumber)
```

```
## [1] 4
```

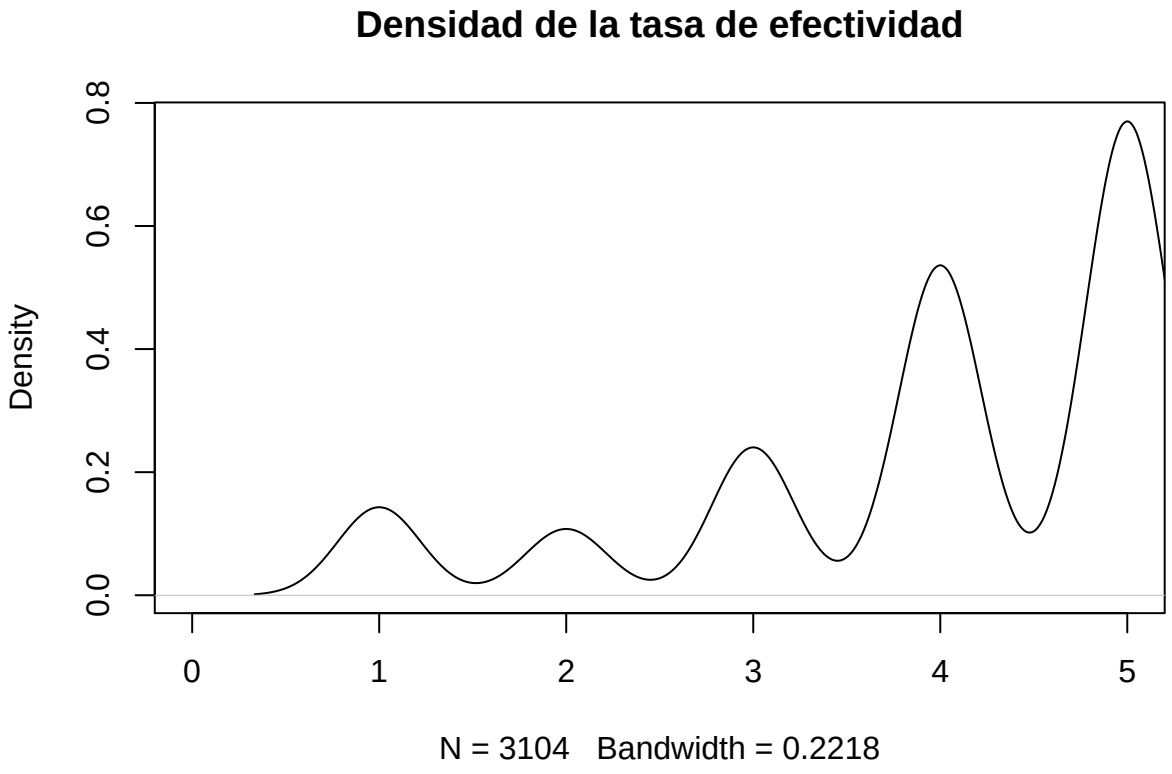
El valor medio obtenido es 3.93 sobre 5 y la mediana es 4. Podemos concluir con dicha información, que en general los medicamentos son bastantes efectivos, situándose el 50 % de dichas mediciones sobre el valor 4.

A continuación, se va a visualizar dicha información gráficamente:

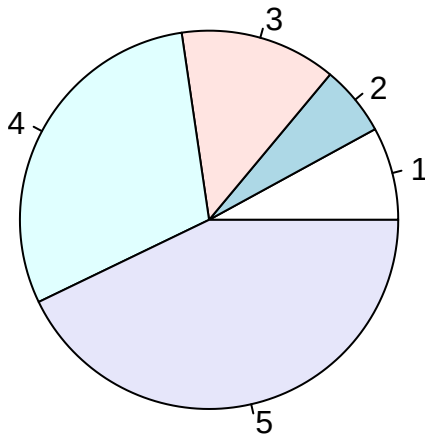
```
# Histograma sobre la efectividad de los medicamentos

efecctivenessNumberExploration <- datos_train$effectivenessNumber

hist(efecctivenessNumberExploration,
     main="Efectividad de los medicamentos",
     xlab="Nivel de efectividad",
     ylab="Frecuencia",
     border="green",
     xlim=c(1,5),
     col= "darkseagreen1",
     breaks=5,
     prob=TRUE
    )
```



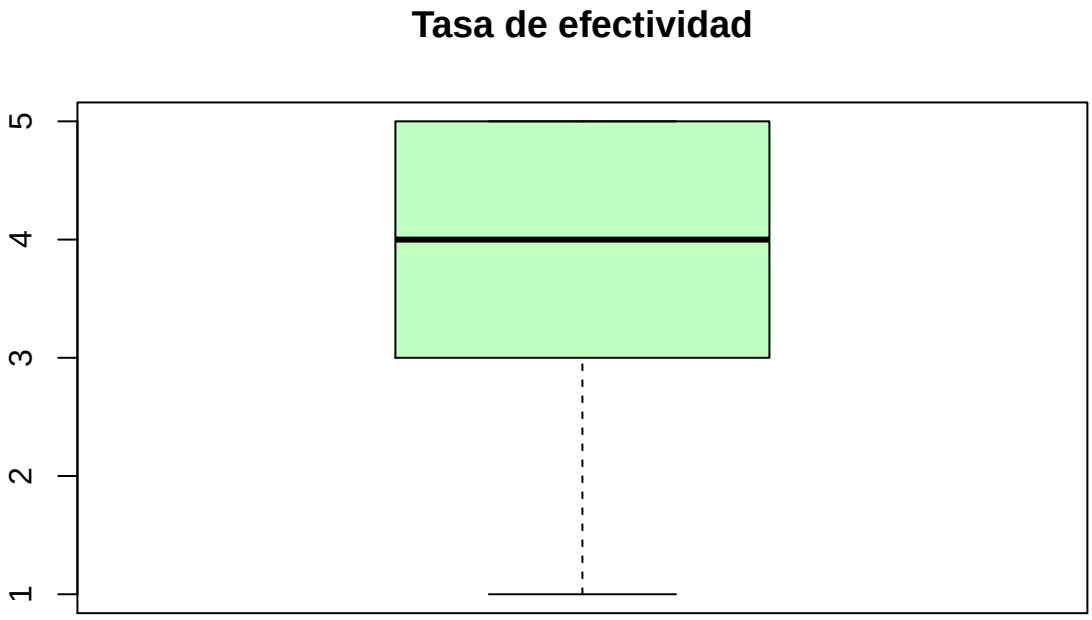

```
# Diagrama de densidad de la efectividad de los medicamentos

plot(density(datos_train$effectivenessNumber),
     main="Densidad de la tasa de efectividad",
     xlim=c(0,5),
     )
```




```r
# Diagrama de sectores de la efectividad de los medicamentos
pie(table(datos_train$effectivenessNumber))
```




```r
# Diagrama de cajas sobre la efectividad de los medicamentos
boxplot(datos_train$effectivenessNumber,main="Tasa de efectividad", col= "darkseagreen1" )
```



Como medidas de dispersión, se va a calcular la **desviación típica**:

```
# Desviación típica
sd(datos_train$effectivenessNumber)
```

```
## [1] 1.230634
```

Como se puede observar, la desviación típica nos da un valor de 1.23. Esto quiere decir que la mayor parte de los valores se sitúan en un intervalo con una distancia de uno de la media.

Este valor concuerda, puesto que si observamos el histograma anterior, vemos que la mayoría de las puntuaciones se sitúan entre 3 y 5.

Esto también nos da como **conclusión** que en general los medicamentos tienen una tasa bastante **buena de efectividad** puesto que su tasa se sitúa entre [3.5].

Efectos secundarios del medicamento

En esta sección, se va a analizar si se consideran que los medicamentos tienen efectos secundarios o no. Para ello se va a analizar el atributo **sideEffectsNumber** (que mide la tasa de efectos secundarios del medicamento, siendo 1 el mínimo de efectos secundarios y 5 el máximo de efectos secundarios)

Empezamos obteniendo las frecuencias y porcentaje total de las anotaciones de efectos secundarios. Para ello se va a calcular la frecuencia de dicho atributo y su porcentaje respecto del total.

```
# Obtener frecuencias del sideEffectsNumber
table(datos_train$sideEffectsNumber)
```

```
##
##    1    2    3    4    5
##  930 1019  612  368  175
```

```
# Calculamos el número de documentos
numDocuments <- dim(datos_train)[1]

# Calculamos el porcentaje de tasa de efectos secundarios respecto del total.
table(datos_train$sideEffectsNumber)/numDocuments
```

```
##
##          1          2          3          4          5
## 0.29961340 0.32828608 0.19716495 0.11855670 0.05637887
```

Como podemos observar, la mayoría de los medicamentos se consideran que no tienen efectos secundarios severos. De hecho, la mayoría de los medicamentos se sitúan entre sin efectos secundarios (29 %) o que tienen efectos secundarios leves (32 %).

Podemos comprobar ésto mediante el uso de la moda.

```
# Obtenemos la moda para el sideEffectsNumber
calcularModa(datos_train$sideEffectsNumber)
```

```
## [1] "2"
```

Como resumen en general de sobre la tasa de efectos secundarios, se va a calcular la media y la mediana para calcular la tendencia central para dicha variable.

La media es la siguiente:

```
# Media
mean(datos_train$sideEffectsNumber)
```

```
## [1] 2.303802
```

La mediana es la siguiente:

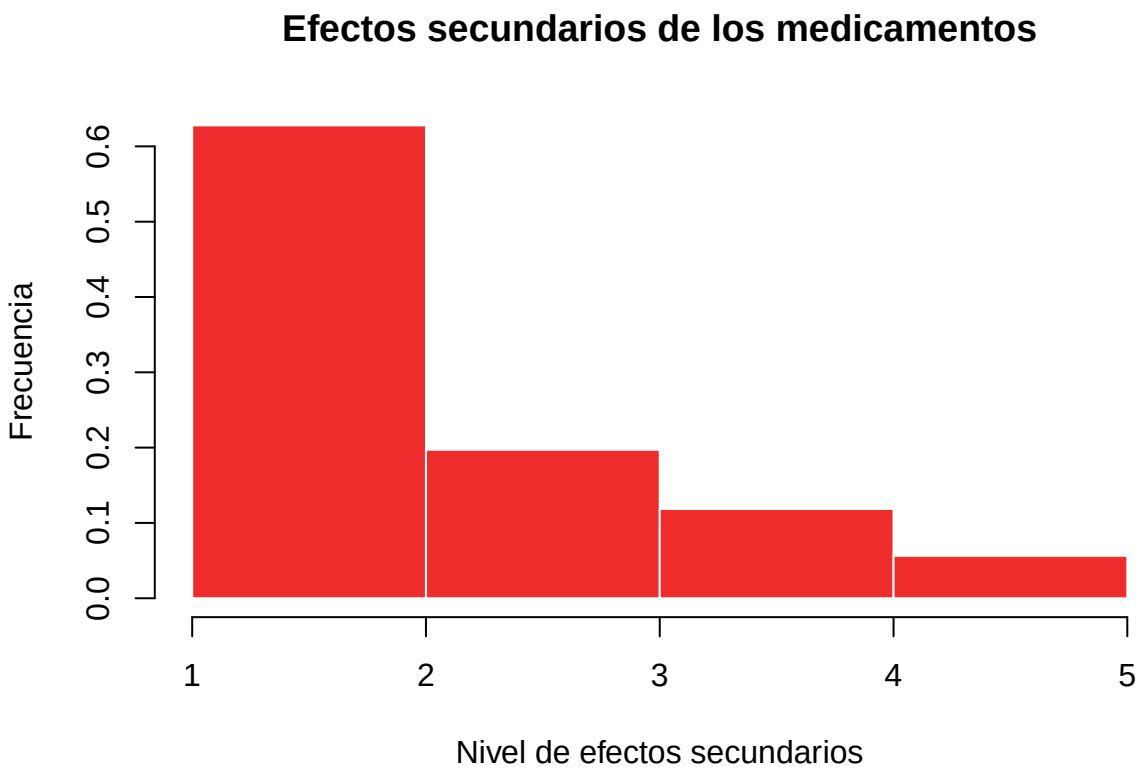
```
# Mediana
median(datos_train$sideEffectsNumber)
```

```
## [1] 2
```

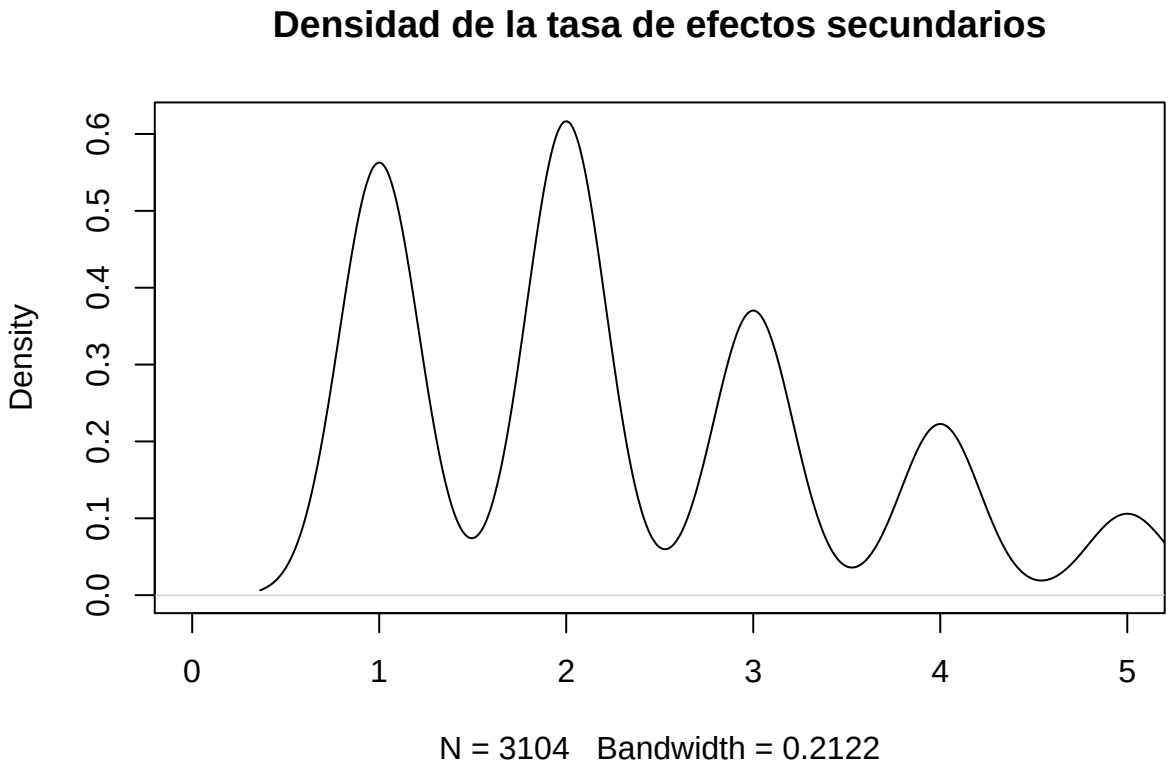
El valor medio obtenido es 2.30 sobre 5 y la mediana es 2. Podemos concluir con dicha información, que en general los medicamentos no tienen efectos secundarios o que dichos efectos son leves.

A continuación, se va a visualizar dicha información gráficamente:

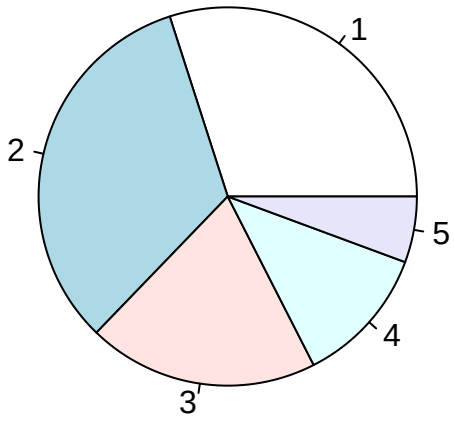
```
# Histograma de la tasa de efectos secundarios

sideEffectsNumberExploration <- datos_train$sideEffectsNumber

hist(sideEffectsNumberExploration,
     main="Efectos secundarios de los medicamentos",
     xlab="Nivel de efectos secundarios",
     ylab="Frecuencia",
     border="white",
     xlim=c(1,5),
     col= "firebrick2",
     breaks=5,
     prob=TRUE
    )
```



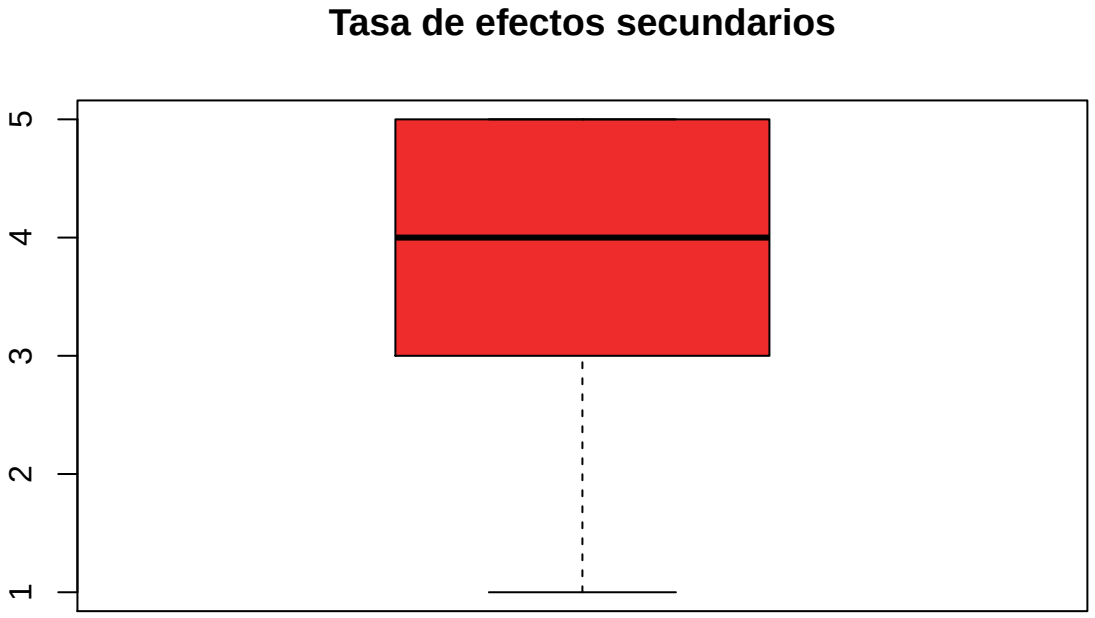

```r
# Diagrama de densidad sobre la tasa de efectos secundarios

plot(density(datos_train$sideEffectsNumber),
     main="Densidad de la tasa de efectos secundarios",
     xlim=c(0,5),
     )
```




```r
# Diagrama de sectores de los efectos secundarios de los medicamentos
pie(table(datos_train$sideEffectsNumber))
```




```r
# Diagrama de cajas de los efectos secundarios de los medicamentos
boxplot(datos_train$effectivenessNumber,main="Tasa de efectos secundarios", col= "firebrick2" )
```



Como medidas de dispersión, se va a calcular la **desviación típica**:

```
# Desviación típica
sd(datos_train$sideEffectsNumber)
```

```
## [1] 1.177525
```

Como se puede observar, la desviación típica nos da un valor de 1.17. Esto quiere decir que la mayor parte de los valores se sitúan en un intervalo con una distancia de uno de la media.

Este valor concuerda, puesto que si observamos el histograma anterior, vemos que la mayoría de las puntuaciones se sitúan entre 1 y 2.

Esto también nos da como **conclusión** que en general los medicamentos **no tienen efectos secundarios o son muy leves**.

Valoración ponderada sobre el medicamento

En esta sección, se va a analizar si se consideran que los medicamentos son buenos o no teniendo en cuenta la relación entre los beneficios que aporta (efectividad) y las inconvenientes que tiene (efectos secundarios). Para ello se va a analizar el atributo **weightedRating** (que mide dicha relación teniendo en cuenta una tasa de efectividad del 30 % y una tasa de efectos secundarios del 70 %), y siendo 1 peor valorado y 10 mejor valorado.

Empezamos obteniendo las frecuencias y porcentaje total de las anotaciones sobre la puntuación ponderada. Para ello se va a calcular la frecuencia de dicho atributo y su porcentaje respecto del total.

```
# Obtener frecuencias del weightedRating
table(datos_train$weightedRating)
```

```
##
##    2    4    6    8   10
##  151  236  494 1212 1011
```

```
# Calculamos el número de documentos
numDocuments <- dim(datos_train)[1]

# Calculamos el porcentaje de puntuación ponderada respecto del total.
table(datos_train$weightedRating)/numDocuments
```

```
##
##          2          4          6          8         10
## 0.04864691 0.07603093 0.15914948 0.39046392 0.32570876
```

Como podemos observar, la mayoría de los medicamentos se consideran que son generalmente beneficiosos. De hecho, la mayoría de los medicamentos se sitúan con una valoración de 8 sobre 10 teniendo en cuenta la relación beneficio/perjuicio.

Podemos comprobar ésto mediante el uso de la moda.

```
# Obtenemos la moda para el weightedRating
calcularModa(datos_train$weightedRating)
```

```
## [1] "8"
```

Como resumen en general de sobre la tasa de efectos secundarios, se va a calcular la media y la mediana para calcular la tendencia central para dicha variable.

La media es la siguiente:

```
# Media
mean(datos_train$weightedRating)
```

```
## [1] 7.737113
```

La mediana es la siguiente:

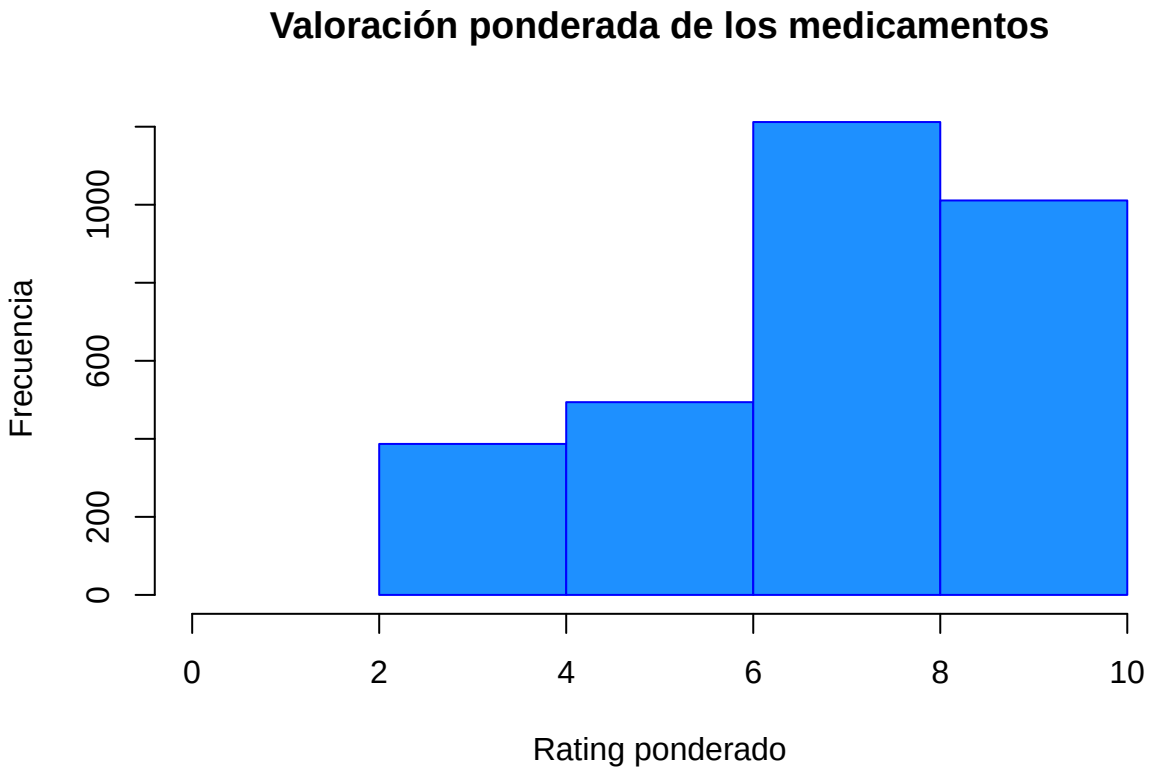
```
# Mediana
median(datos_train$weightedRating)
```

```
## [1] 8
```

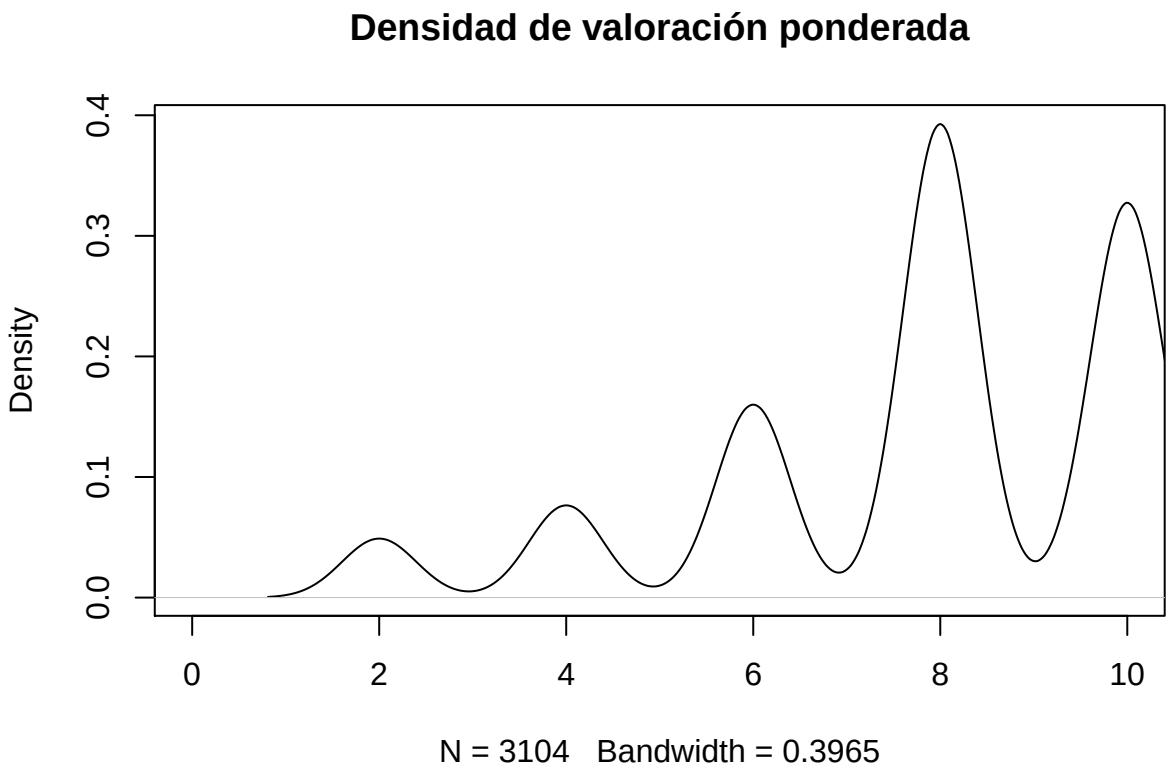
El valor medio obtenido es 7.52 sobre 10 y la mediana es 2. Podemos concluir con dicha información que la puntuación general sobre los medicamentos es de **notable**.

A continuación, se va a visualizar dicha información gráficamente:

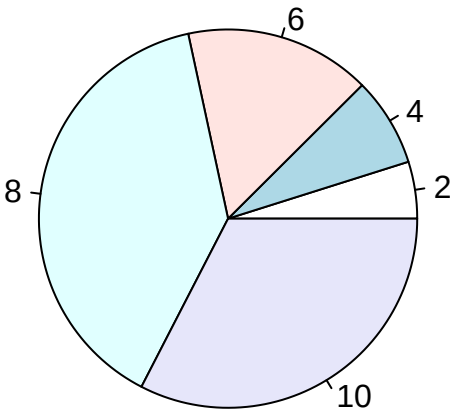
```
# Histograma de valoración ponderada

weightedRatingExploration <- datos_train$weightedRating

hist(weightedRatingExploration ,
     main="Valoración ponderada de los medicamentos",
     xlab="Rating ponderado",
     ylab="Frecuencia",
     border="blue",
     xlim=c(0,10),
     col= "dodgerblue1",
     breaks=5

    )
```



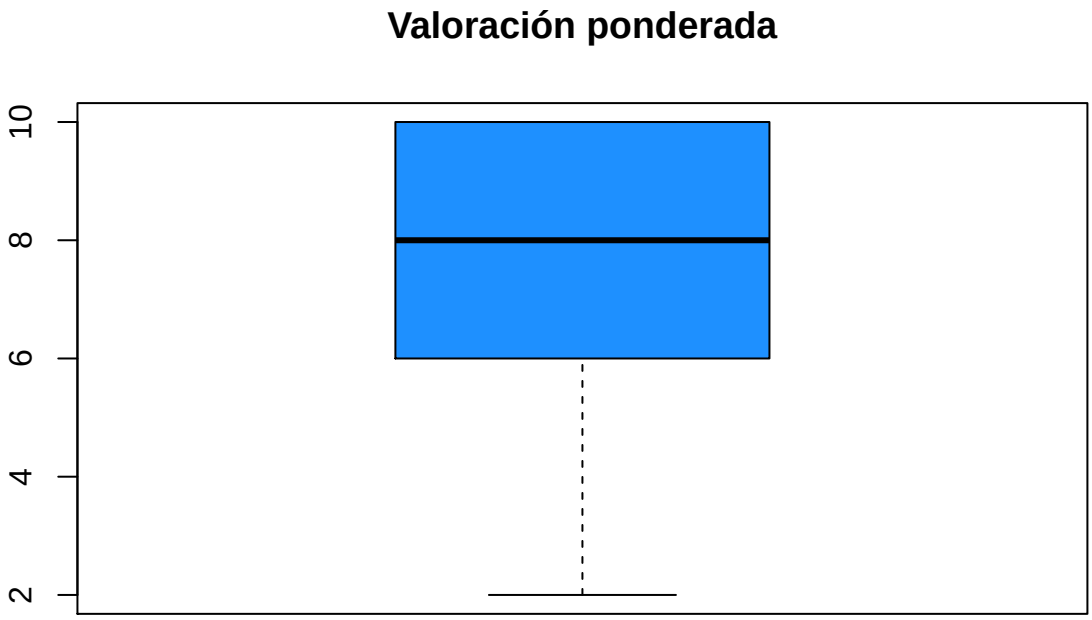

```r
# Diagrama de densidad sobre la valoración ponderada

plot(density(datos_train$weightedRating),
     main="Densidad de valoración ponderada",
     xlim=c(0,10),
     )
```




```r
# Diagrama de sectores sobre la valoración ponderada
pie(table(datos_train$weightedRating))
```




```r
# Diagrama de cajas sobre la valoración ponderada
boxplot(datos_train$weightedRating,main="Valoración ponderada", col= "dodgerblue1" )
```



Como medidas de dispersión, se va a calcular la **desviación típica**:

```
# Desviación típica
sd(datos_train$weightedRating)
```

```
## [1] 2.199924
```

Como se puede observar, la desviación típica nos da un valor de 2.07. Esto quiere decir que la mayor parte de los valores se sitúan en un intervalo con una distancia de dos de la media.

Este valor concuerda, puesto que si observamos el histograma anterior, vemos que la mayoría de las puntuaciones se sitúan entre 6 y 10.

Esto también nos da como **conclusión** que en general los medicamentos **son convenientes tomarlos**.

Correlación sobre las variables

En esta sección se va a comprobar la correlación que existe entre las variables que miden la efectividad(effectivenessNumber), los efectos secundarios(sideEffectsNumber), la valoración aportada por los usuarios(rating) y la valoración ponderada que se ha realizado sobre el medicamento(weightedRating).

Empezamos calculando la correlación entre la variable que mide la efectividad y los efectos secundarios.

```
# Correlación lineal entre efectivenessNumber y sideEffectsNumber
cor(datos_train[,c(8,9)])
```

```
##                     sideEffectsNumber effectivenessNumber
## sideEffectsNumber           1.0000000          -0.3953789
## effectivenessNumber        -0.3953789           1.0000000
```

Podemos observar que cuantos más efectos secundarios tiene, menor es la efectividad del medicamento. Esto puede estar influido por las valoraciones subjetivas del usuario, ya que si ha tenido una mala experiencia (debido a los efectos secundarios) por la ingesta del medicamento, no va a hacer énfasis en los beneficios del medicamento, sino que hará un mayor énfasis en los aspectos negativos.

A continuación vamos a calcular la correlación entre la efectividad y la valoración ponderada del medicamento.

```
# Correlación lineal entre efectivenessNumber y weightedRating
cor(datos_train[,9:10])
```

```
##                      effectivenessNumber weightedRating
## effectivenessNumber           1.0000000      0.9237202
## weightedRating                0.9237202      1.0000000
```

Como se puede observar, cuando el medicamento es más efectivo, la valoración ponderada del medicamento acumenta (como es obvio), y si ahora calculamos la valoración ponderada del medicamento teniendo en cuenta los efectos secundarios

```
# Correlación lineal entre sideEffectsNumber y weightedRating
cor(datos_train[,c(8,10)])
```

```
##                    sideEffectsNumber weightedRating
## sideEffectsNumber          1.000000      -0.649161
## weightedRating            -0.649161       1.000000
```

Observamos como si el medicamento tiene una mayor tasa de efectos secundarios, la valoración ponderada disminuye considerablemente (obvio porque el 70 % de la valoración ponderada tiene en cuenta los efectos secundarios del medicamento).

Ahora vamos a comprobar la relación que existe entre la valoración dada por el usuario y los efectos secundarios.

```
# Correlación lineal entre rating y sideEffectsNumber
cor(datos_train[,c(2,8)])
```

```
##                       rating sideEffectsNumber
## rating             1.000000         -0.682939
## sideEffectsNumber -0.682939          1.000000
```

Podemos comprobar como si el medicamento tiene una mayor tasa de efectos secundarios, la valoración dada por el usuario disminuye

```
# Correlación lineal entre rating y effectivenessNumber
cor(datos_train[,c(2,9)])
```

```
##                         rating effectivenessNumber
## rating              1.0000000           0.7498171
## effectivenessNumber 0.7498171           1.0000000
```

Y si la efectividad del medicamento es alta, la valoración del usuario se incrementa.

Como observación general, se puede destacar que **la valoración del usuario está condicionada más por la efectividad del medicamento** (relación 1/0.74) que por los efectos secundarios (relación 1/-0.68).

Por último, vamos a observar en el siguiente gráfico como se relacionan las variables entre sí en función se sus valores.

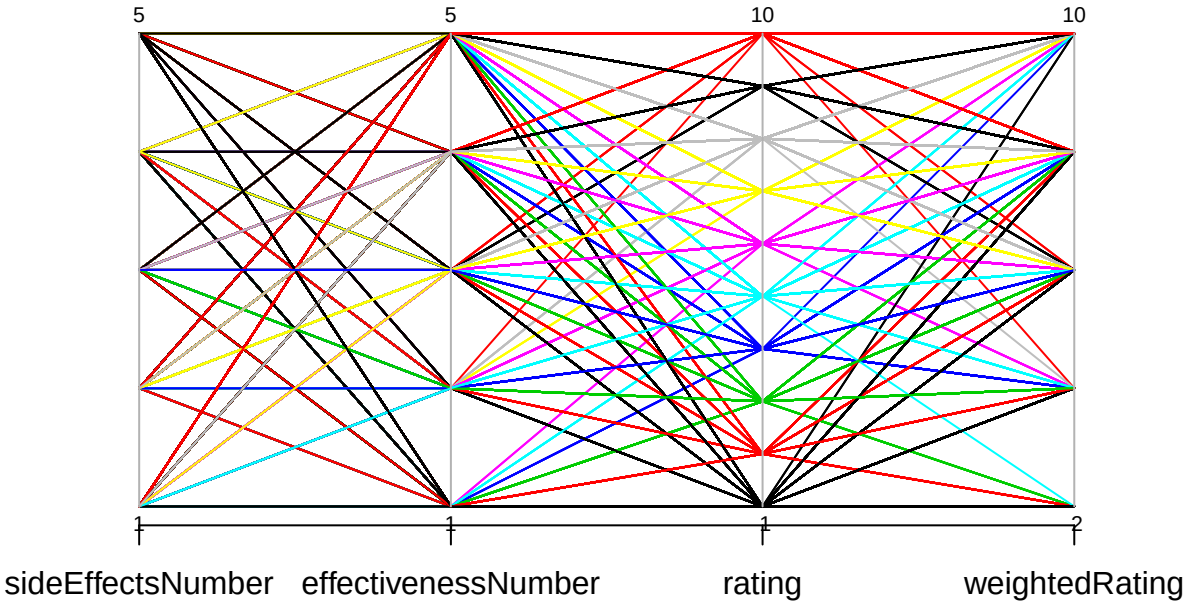
```
# Gráfico de coordenadas paralelas
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:rlm':
##
##     rlm
```

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

```
parcoord(datos_train[,c(8,9,2,10)], col=datos_train$rating,var.label=T)
```



Por ejemplo, podemos destacar como si el número de efectos secundarios es 1 (no tiene efectos secundarios) y la efectividad del medicamento es 5 (muy efectivo), entonces la valoración del usuario será 10 y la valoración ponderada será también 10.

5. Regresión

En este apartado se va hacer uso de la técnica de **regresión**, con el objetivo de describir dependencias significativas entre las variables incluidas en la base de datos. La regresión es un modelo de predicción de variables continuas, en donde las variables independientes tambien son continuas o al menos numéricas. Dicho proceso viene caracterizado por aprender una función que aplica un conjunto de atributos $X_1 X_n$ en otro atributo Y. En nuestro caso, vamos a hacer uso de las columnas:

- **ratingLabel**: etiqueta de 0 ó 1, que contempla la opinión del paciente sobre el medicamento si es favorable (1) o no es favorable (0).
- **effectivenessNumber**: clasificación de la efectividad del medicamento según el paciente (5 posibles valores), en donde el 1 significa que es ineficaz y el 5 que es altamente eficaz.
- **sideEffectsInverse**: clasificación de los efectos secundarios del medicamento según el paciente (5 posibles valores), en donde el 1 significa que teine efectos secundarios extremadamente graves y el 5 que es sin efectos secundarios.

Como se observa en la gráfica 17, se va hacer uso de variables discretas. El objetivo de aplicar regresión, es obtener la curva ROC, la matriz de confusión y el error en el conjunto test, para obtener así la precisión de nuestro modelo y saber según el paciente qué medicamentos favorables o no, tienen efectos secundarios o son efectivos.

Como se ha comentado, se va hacer uso de la matriz de confusión, la cual es una herramienta que permite la visualización del desempeño de un algoritmo, en donde cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real. Una de sus ventajas es que permite ver si la predicción falla o no.

De esta forma, se observa como en la figura 18, la diagonal principal contiene la suma de todas las predicciones correctas (el modelo dice “S” y acierta, tiene efectos secundarios, o dice “N” y acierta también, no tiene efectos secundarios). La otra diagonal refleja los errores del clasificador: los falsos positivos o “true positives”

(dice que es tiene efectos secundarios "S", pero en realidad no tiene "n"), o los falsos negativos o "false negatives" (dice que es no tiene efectos secundarios "N", pero en realidad los tiene "p"). Además, otra de las medidas a usar, será la curva ROC, para caracterizar el comportamiento de la predicción.

Pero previo a la realización de las técnicas, se ha optado por eliminar los fármacos repetidos, para así obtener una predicción más acertada. Para dicha eliminación, se han cogido todos los medicamentos duplicados y eliminados los necesarios. Así que si tenemos 5 filas de un mismo medicamentos, se eliminan las 4 consecutivas al primero.

5.1. Regresión Lineal

La regresión lineal simple consiste en generar un modelo de regresión (ecuación de una recta) que permita explicar la relación lineal que existe entre dos variables. A la variable dependiente o respuesta se le identifica como Y y a la variable predictora o independiente como X. Con el comando `lm()` podemos ajustar el modelo. Algunos parámetros importantes de esta función son:

`lm(formula, data, subset, weights)`

- *formula*: definimos el modelo como: $Y \sim X$.
  - Regresion Múltiple: $y \sim X_1 + X_2 + X_n$.
  - Regresión Polinómica: $y \sim poly(x = X, degree = k)$
  - Interacción de variables: $y \sim X_1 \cdot X_2$. Si sólo queremos el término de interacción: $X1 : X2$
- *data* : especificamos el dataset a utilizar
- *subset* : si dividimos la muestra en training y testing, podemos indicar el subconjunto de entrenamiento con un vector que indique sus números de fila.

5.1.1. Regresión Lineal Simple

Se pretende predecir el valor del rating (etiqueta 0-1) en función de la puntuación de los medicamentos y, por otro lado para los efectos secundarios del mismo. Para ello se va a emplear la función `lm()`, la cual genera un modelo de regresión lineal por mínimos cuadrados en el que la variable respuesta es ratingLabel y el predictor sideEffectsInverse, y por otro lado, la variable respuesta será ratingLabel y el predictor effectivenessNumber.

El modelo de regresión lineal simple se describe de acuerdo a la siguiente ecuación:

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$$

Siendo β_0 la ordenada en el origen, β_1 la pendiente y ϵ el error aleatorio. Este último representa la diferencia entre el valor ajustado por la recta y el valor real y recoge el efecto de todas aquellas variables que influyen en Y pero que no se incluyen en el modelo como predictores. Al error aleatorio también se le conoce como residuo.

A continuación, se realiza una función con la cual obtener los errores dentro y fuera de la muestra, y la matriz de confusión.

```
# Función que calcula los errores y E_test para regresión logística
errorres_regresion_lineal <- function(m){
  probTr = predict(m, type="response")
  probTst = predict(m, data.frame(data_test_procesado), type="response")

  predTst = rep(0, length(probTst)) # predicciones por defecto 0
  predTst[probTst >= 0.5] = 1 # >= 0.5 clase 1

  predTr = rep(0, length(probTr)) # predicciones por defecto 0
  predTr[probTr >= 0.5] = 1 # >= 0.5 clase 1 # Para el calculo del Etest
```

```
  # Calculamos Etest y mostramos la matriz de confusión
  print(table(pred=predTst, real=data_test_procesado$ratingLabel))

  Etrain = mean(predTr != data_train_procesado$ratingLabel)
  Etest = mean(predTst != data_test_procesado$ratingLabel)

  # Devolvemos el eror para el conjunto train y test
  list(Etrain=Etrain*100, Etest=Etest*100)
}
```

Primero vamos a obtener un modelo en donde la variable respuesta es ratingLabel y el predictor sideEffectsInverse. Con el fin de obtener, que medicamentos favorables tienen efectos secundarios. Por lo que tendremos que hacer caso a la otra dioganal de la matriz de confusión.

```
# Creamos el modelo para ratingLabel ~ sideEffectsInverse
lm_effects = lm(data = data_train_procesado, formula = ratingLabel ~ sideEffectsInverse)
summary(lm_effects)
```

```
##
## Call:
## lm(formula = ratingLabel ~ sideEffectsInverse, data = data_train_procesado)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.0164 -0.0164 -0.0164  0.1615  0.6953
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)         0.12683    0.05284    2.40   0.0167 *
## sideEffectsInverse  0.17792    0.01311   13.57   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3328 on 500 degrees of freedom
## Multiple R-squared:  0.2691, Adjusted R-squared:  0.2676
## F-statistic: 184.1 on 1 and 500 DF,  p-value: < 2.2e-16
```

```
# Evaluamos el modelo
errorres_regresion_lineal(lm_effects)
```

```
##     real
## pred   0   1
##    0  47  11
##    1  27 228
```

```
## $Etrain
## [1] 12.3506
##
## $Etest
## [1] 12.14058
```

Para el modelo generado, tanto la ordenada en el origen como la pendiente son significativas (p-values $<$ 0.05). El valor de R^2 indica que el modelo calculado explica el 26.91 % de la variabilidad presente en la variable respuesta (ratingLabel) mediante la variable independiente (sideEffectsInverse). Como se observa se obtiene que hay 11 falsos positivos, es decir, medicamentos que se dicen que tienen efectos secundarios pero no los tienen. Y existen 27 falsos negativos, es decir, medicamentos que se dicen que no tienen efectos

secundarios pero que los tienen. Dichos resultados son los obtenidos según los comentarios de los pacientes. Como podemos apreciar, el error del test nos devuelve un valor de 12.14058, un error aceptable teniendo en cuenta los resultados obtenidos en la matriz de confusión. Pero posiblemente dicho error se pueda reducir aplicando sobre los datos otros modelos que veremos más adelante.

A continuación, vamos a crear el modelo para la variable respuesta ratingLabel y el predictor effectiveness-Number. Con el fin de obtener, que medicamentos favorables son efectivos. Por lo que tendremos que hacer caso a la otra dioganal de la matriz de confusión.

```
# Creamos el modelo para ratingLabel ~ effectivenessNumber
lm_effectiveness = lm(data = data_train_procesado, formula = ratingLabel ~ effectivenessNumber)
summary(lm_effectiveness)
```

```
##
## Call:
## lm(formula = ratingLabel ~ effectivenessNumber, data = data_train_procesado)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.01938 -0.01938 -0.01938  0.17667  0.76480
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          0.03915    0.04733   0.827    0.409
## effectivenessNumber  0.19605    0.01145  17.129   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.309 on 500 degrees of freedom
## Multiple R-squared:  0.3698, Adjusted R-squared:  0.3685
## F-statistic: 293.4 on 1 and 500 DF,  p-value: < 2.2e-16
```

```
# Evaluamos el modelo
errorres_regresion_lineal(lm_effectiveness)
```

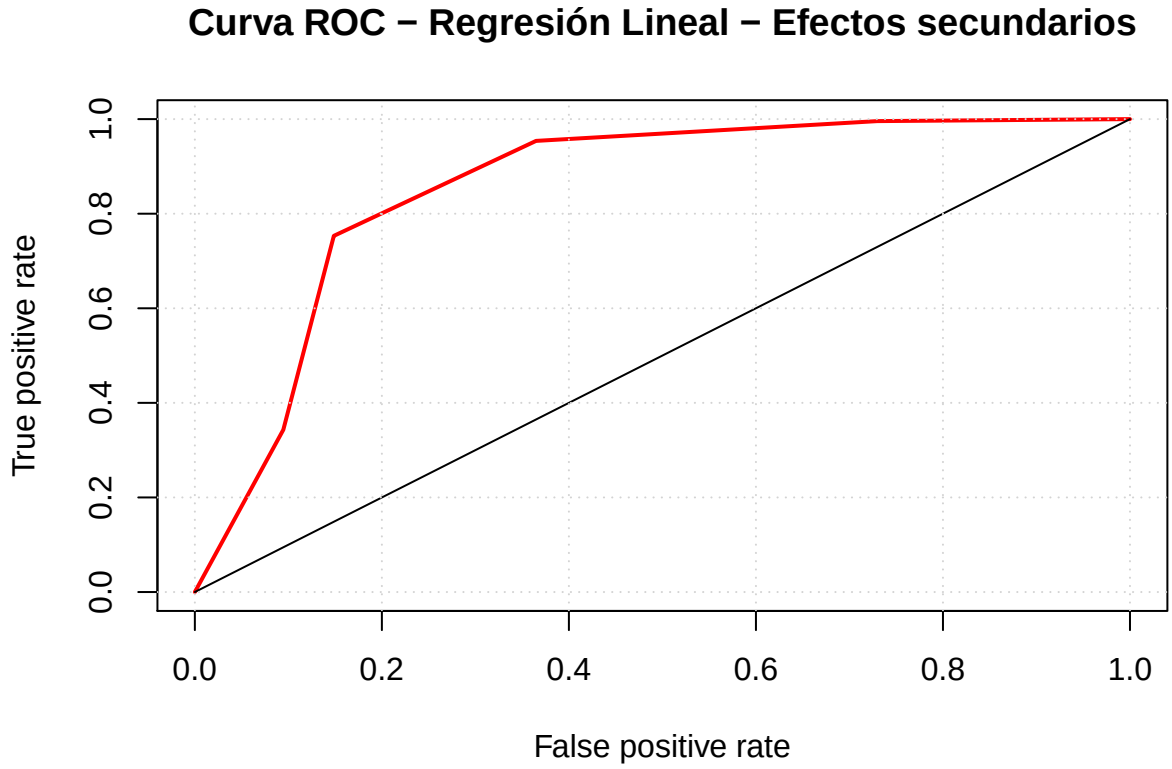
```
##     real
## pred   0   1
##    0  39   3
##    1  35 236
```

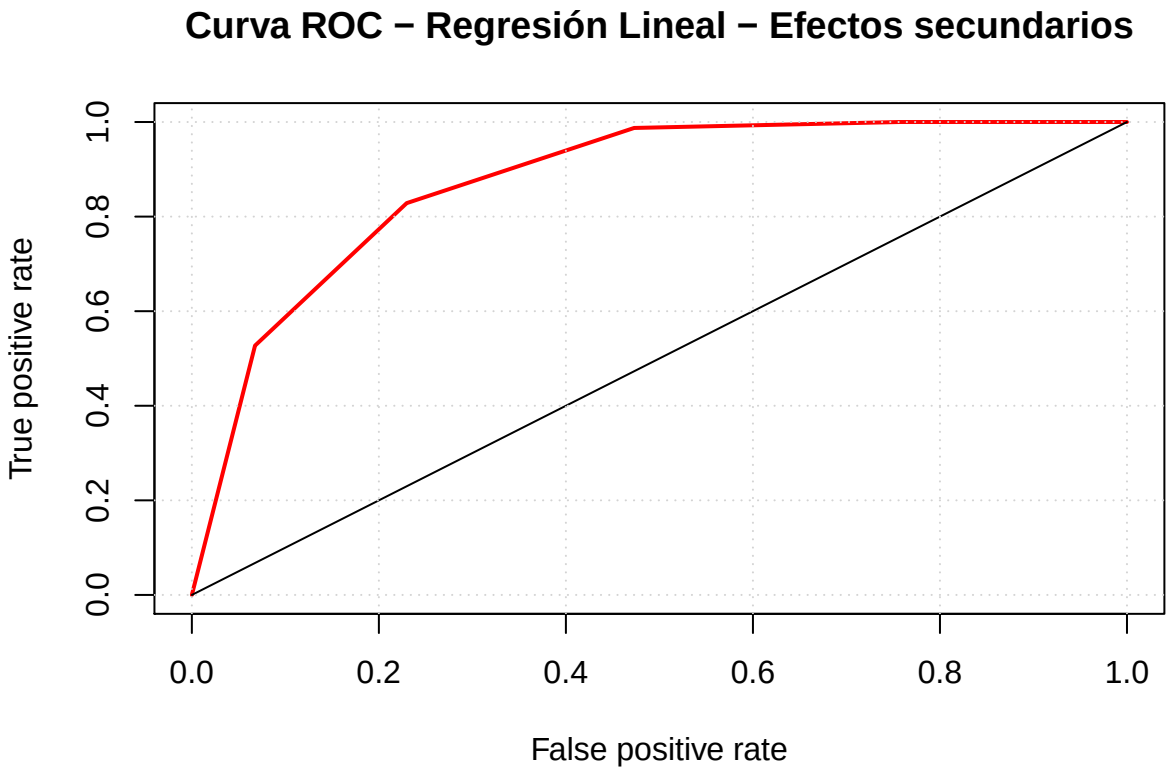
```
## $Etrain
## [1] 10.15936
##
## $Etest
## [1] 12.14058
```

Para el modelo generado, tanto la ordenada en el origen como la pendiente son significativas (p-values $<$ 0.05). El valor de R^2 indica que el modelo calculado explica el 36.98 % de la variabilidad presente en la variable respuesta (ratingLabel) mediante la variable independiente (effectivenessNumber). Como se observa se obtiene que hay 3 falsos positivos, es decir, medicamentos que se dicen que son efectivos pero que no lo son. Y existen 35 falsos negativos, es decir, medicamentos que se dicen que no son efectivos pero los son Dichos resultados, según los comentarios de los pacientes. Como podemos apreciar, el error del test nos devuelve un valor de 12.14058, un error aceptable teniendo en cuenta los resultados obtenidos en la matriz de confusión. Pero posiblemente dicho error se pueda reducir aplicando sobre los datos otros modelos que veremos más adelante.

Una vez que hemos obtenidos las prediciones para nuestro modelo, vamos a obtener las probabilidades para el mismo con el fin de obtener la curva ROC.

```r
# Función que dibuja una curva ROC
plotROC <- function(modelo, etiq_real, adicionar=FALSE,color="red") {

  # Realizamos la predicción con la función de RORCR
  pred <- prediction(modelo, etiq_real)
  perf <- performance(pred,"tpr","fpr")
  plot(perf, col=color, add=adicionar,
       main="Curva ROC - Regresión Lineal - Efectos secundarios", lwd = 2)
  segments(0, 0, 1, 1, col='black')
  grid()
}
```





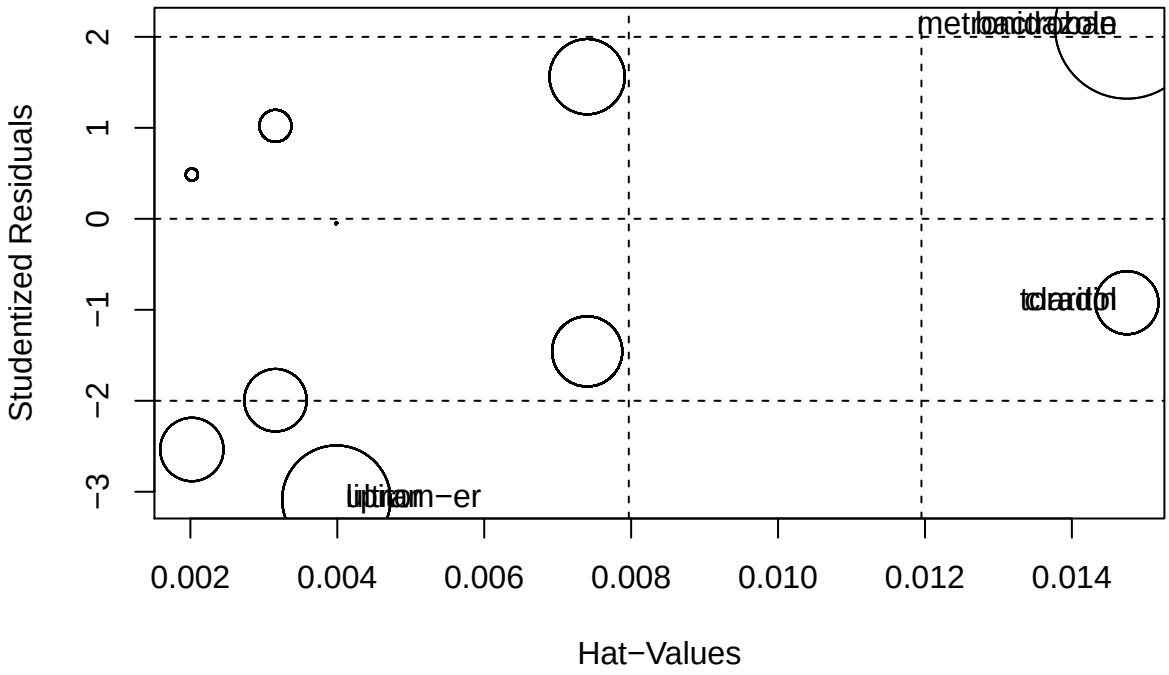
5.1.1.1. Casos atípicos

A continuación para el modelo lineal simple, vamos a obtener los casos atípicos, una forma rápida de identificarlos es usando la función `influencePlot()` del paquete `car`. Esta función produce un gráfico que señala a los casos atípicos influyentes.

Una de las fuentes de violaciones de los supuestos en el modelado lineal es la presencia de casos atípicos. Un caso atípico es aquel que, dados ciertos valores de $x \sim 1$, $x \sim 2$, $x \sim n$ tiene valores de y muy diferentes a los demás y por lo tanto producen un residuo muy alto. Estos casos atípicos pueden tener gran influencia sobre el modelo, ya que el criterio de mínimos cuadrados buscará minimizar el error y cambiará la pendiente para dar cuenta de estos casos.

Primero vamos a obtener los casos atípicos para el modelo de los efectos secundarios del medicamento.

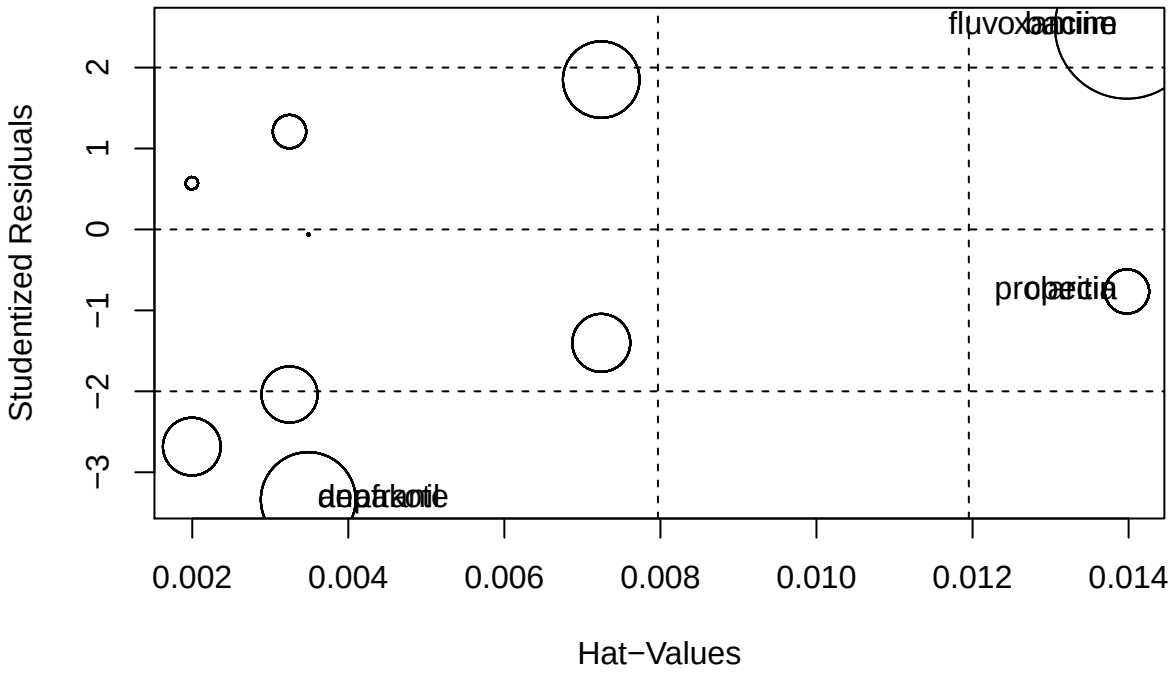
```
# Detección y visualización de observaciones influyentes para effects
influencePlot(lm_effects, xlab="Hat-Values", ylab="Studentized Residuals")
```




```
##                   StudRes         Hat      CookD
## claritin      -0.9223289 0.014750230 0.00636977
## lipitor       -3.0860011 0.003986799 0.01874047
## ultram-er     -3.0860011 0.003986799 0.01874047
## toradol       -0.9223289 0.014750230 0.00636977
## metronidazole  2.1118521 0.014750230 0.03315542
## bactroban      2.1118521 0.014750230 0.03315542
```

Para dicho modelo, los medicamentos *claritin*, *lipitor*, *ultram-er*, *metronidazole*, *toradol* y *bactroban* aparecen como casos atípicos. A continuación, vamos a obtener los casos atípicos para el modelo de la efectividad del medicamento.

```
# Detección y visualización de observaciones influyentes para effectivenessNumber
influencePlot(lm_effectiveness, xlab="Hat-Values", ylab="Studentized Residuals")
```




```
##                StudRes         Hat       CookD
## propecia    -0.7661251 0.013979059 0.004164089
## claritin    -0.7661251 0.013979059 0.004164089
## depakote    -3.3376924 0.003486567 0.019101083
## anafranil   -3.3376924 0.003486567 0.019101083
## fluvoxamine  2.5054155 0.013979059 0.044031318
## baciim       2.5054155 0.013979059 0.044031318
```

Para dicho modelo, los medicamentos *propecia*, *claritin*, *depakote*, *anafranil* y *fluvoxamine* aparecen como casos atípicos.

5.1.1.2. Regresión Robusta

Una alternativa para controlar los casos atípicos es ajustar un modelo lineal robusto. El modelo lineal robusto utiliza criterios diferentes al de los mínimos cuadrados y pondera la influencia de los casos atípicos, por lo que producen coeficientes y sobre todo errores estándar más confiables. El paquete `MASS` incluye la función `rlm()`, de sintaxis similar a `lm()` que implementa el método para el ajuste de los coeficientes y cálculo de los errores estándar.

Primero, calcularemos el modelo en donde la variable respuesta es ratingLabel y el predictor sideEffectsInverse. Con el fin de obtener, que medicamentos favorables tienen efectos secundarios. Por lo que tendremos que hacer caso a la otra dioganal de la matriz de confusión.

```
# Creamos el modelo para ratingLabel ~ sideEffectsInverse
rlm_effects = MASS::rlm(ratingLabel ~ sideEffectsInverse, data = data_train_procesado)
```

```
## Warning in rlm.default(x, y, weights, method = method, wt.method =
## wt.method, : 'rlm' failed to converge in 20 steps
```

```
stargazer(lm_effects, rlm_effects, type = "text", model.numbers = FALSE, title="Comparación de modelo O
```

```
##
## Comparación de modelo OLS y Robusto
## ===============================================================
##                                        Dependent variable:
```

```
##                                 ---------------------------------
##                                           ratingLabel
##                                          OLS             robust
##                                                          linear
## ------------------------------------------------------------------
## sideEffectsInverse                     0.178***         0.006***
##                                        (0.013)          (0.0004)
##
## Constant                               0.127**          0.971***
##                                        (0.053)          (0.002)
##
## ------------------------------------------------------------------
## Observations                             502              502
## R2                                      0.269
## Adjusted R2                             0.268
## Residual Std. Error (df = 500)          0.333            0.007
## F Statistic                      184.050*** (df = 1; 500)
## ==================================================================
## Note:                                  *p<0.1; **p<0.05; ***p<0.01
```

```r
# Creamos el modelo para ratingLabel ~ effectivenessNumber
rlm_effectiveness = MASS::rlm(ratingLabel ~ effectivenessNumber, data = data_train_procesado)
stargazer(lm_effectiveness, rlm_effectiveness, type = "text", model.numbers = FALSE, title="Comparación
```

```
##
## Comparación de modelo OLS y Robusto
## ==================================================================
##                                        Dependent variable:
##                                 ---------------------------------
##                                           ratingLabel
##                                          OLS             robust
##                                                          linear
## ------------------------------------------------------------------
## effectivenessNumber                    0.196***         0.213***
##                                        (0.011)          (0.008)
##
## Constant                                0.039            0.006
##                                        (0.047)          (0.034)
##
## ------------------------------------------------------------------
## Observations                             502              502
## R2                                      0.370
## Adjusted R2                             0.369
## Residual Std. Error (df = 500)          0.309            0.210
## F Statistic                      293.392*** (df = 1; 500)
## ==================================================================
## Note:                                  *p<0.1; **p<0.05; ***p<0.01
```

El modelo robusto aumenta los coeficientes y reduce los errores estándar, aunque no en gran cuantía. En este caso interpretamos que los casos atípicos no son un problema grave.

5.1.2. Regresión Lineal Múltiple

Una vez que hemos la regresión lineal simple, vamos a intentar predecir el valor del ratingLabel en función de la puntuación de los medicamentos y de los efectos secundarios del mismo. Para ello se va a emplear la misma

(`lm()`), la cual genera un modelo de regresión lineal por mínimos cuadrados en el que la variable respuesta es ratingLabel y los predictores son sideEffectsInverse y effectivenessNumber. Previamenete a la realización del modelo, vamos a comprobar que que las variables a usar pueden ser aplicadas juntas con la función `step()`, para así determinar la calidad del modelo.

```
# Creamos el modelo para ratingLabel ~ effectivenessNumber + sideEffectsInverse
lm_multiple <- lm(ratingLabel ~ sideEffectsInverse + effectivenessNumber, data = data_train_procesado)

# https://rpubs.com/Cristina_Gil/Regresion_Lineal_Multiple
# Nos dice si existe alguna variable que estamos usando que no nos hace falta
step(lm_multiple, direction = "both", trace = 1)
```

```
## Start:  AIC=-1296.57
## ratingLabel ~ sideEffectsInverse + effectivenessNumber
##
##                        Df Sum of Sq    RSS     AIC
## <none>                              37.481 -1296.6
## - sideEffectsInverse    1    10.270 47.751 -1177.0
## - effectivenessNumber   1    17.903 55.384 -1102.6
```

```
##
## Call:
## lm(formula = ratingLabel ~ sideEffectsInverse + effectivenessNumber,
##     data = data_train_procesado)
##
## Coefficients:
##         (Intercept)   sideEffectsInverse  effectivenessNumber
##             -0.3362               0.1311                0.1628
```

```
# Obtenemos el error d
errorres_regresion_lineal(lm_multiple)
```

```
##     real
## pred   0   1
##    0  50   5
##    1  24 234
```

```
## $Etrain
## [1] 8.366534
##
## $Etest
## [1] 9.265176
```

Como se aprecia en salida de la función `step(lm_multiple, direction = "both", trace = 1)`, las dos variables deben ser incluidas en el proceso de selección.

Por tanto, se observa se obtiene que hay 5 falsos positivos, es decir, medicamentos que se dicen que son favorables pero que no lo son. Y existen 24 falsos negativos, es decir, medicamentos que se dicen que no favorables pero los son. Se debe tener en cuenta que dichos resultados obtenidos han sido según los comentarios de los pacientes. Como podemos apreciar, el error del test nos devuelve un valor de 9.265176, un buen error si tenemos en cuenta, los obtenidos con la regresión simple.

5.2. Regresión Logística

A continuación, vamos a ajustar un modelo con Regresión Logística binario sin regularización. De este modo vamos a comprobar, si se puede intentar encontrar un buen modelo lineal para nuestro problema. Además,

discutiremos las necesidades de regularización, para ver si de esta forma conseguimos mejorarlo en cuanto a resultados.

Para la obtención de la regresión, vamos hacer uso del comando `glm()`.

5.2.1. Regresión Logística Simple

La regresión logística simple, es un método de regresión que permite estimar la probabilidad de una variable cualitativa binaria en función de una variable cuantitativa. Una de las principales aplicaciones de la regresión logística es la de clasificación binaria, en el que las observaciones se clasifican en un grupo u otro dependiendo del valor que tome la variable empleada como predictor. Es decir para predecir la probabilidad de pertenencia o no a una determinada clase (efectividad - no efectividad).

Para ello, vamos a utilizar la función `glm(..)` que en base al conjunto de muestras de entrenamiento suministrado, es capaz de calcular una probabilidad para cada dato. Si esta probabilidad sobrepasa el valor 0.5, entonces consideramos que dicha muestra pertence a la clase 1, sin embargo, si su probabilidad es menor que 0.5, entonces consideramos que se encuentra en la clase con etiqueta 0. Una vez tengamos las etiquetas predichas a partir de las probabilidades que calcula el modelo, vamos a calcular el error dentro de la muestra y el error fuera de la muestra. Para ello, vamos a especificar como primer argumento, la variable respuesta denominada ratingLabel y a continuación, la variable en la que nos vamos a fijar para ajustar el modelo.

```
# Función que calcula los errores y E_test para regresión logística
errorres_regresion_logistica <- function(m){
  probTr = predict(m, type="response")
  probTst = predict(m, data.frame(data_test_procesado), type="response")

  predTst = rep(0, length(probTst)) # predicciones por defecto 0
  predTst[probTst >= 0.5] = 1 # >= 0.5 clase 1

  predTr = rep(0, length(probTr)) # predicciones por defecto 0
  predTr[probTr >= 0.5] = 1 # >= 0.5 clase 1 # Para el calculo del Etest

  print(table(pred=predTst, real=data_test_procesado$ratingLabel)) # Calculamos Etest

  Etrain = mean(predTr != data_train_procesado$ratingLabel)
  Etest = mean(predTst != data_test_procesado$ratingLabel)

  list(Etrain=Etrain*100, Etest=Etest*100)
}
```

Primero vamos a obtener el modelo, que predice la variable sideEffectsInverse a partir del ratingLabel.

```
# Creamos el modelo para ratingLabel ~ sideEffectsInverse
gml_effects = glm(ratingLabel ~ sideEffectsInverse, family = binomial(logit), data = data_train_procesa
# Evaluamos el modelo
errorres_regresion_logistica(gml_effects)
```

```
##     real
## pred   0   1
##    0  47  11
##    1  27 228
```

```
## $Etrain
## [1] 12.3506
##
## $Etest
```

```
## [1] 12.14058
```

Como se observa se obtiene que hay 11 falsos positivos, es decir, medicamentos que se dicen que tienen efectos secundarios pero no los tienen. Y existen 27 falsos negativos, es decir, medicamentos que se dicen que no tienen efectos secundarios pero que los tienen. Dichos resultados son los obtenidos según los comentarios de los pacientes. Como podemos apreciar, el error del test nos devuelve un valor de 12.14058, un error aceptable teniendo en cuenta los resultados obtenidos en la matriz de confusión. Pero se ha comprobado, que se obtiene un mejor error con la regresión lineal múltiple.

A continuación, vamos a crear el modelo para la variable respuesta ratingLabel y el predictor effectiveness-Number. Con el fin de obtener, que medicamentos favorables son efectivos. Por lo que tendremos que hacer caso a la otra diagonal de la matriz de confusión.

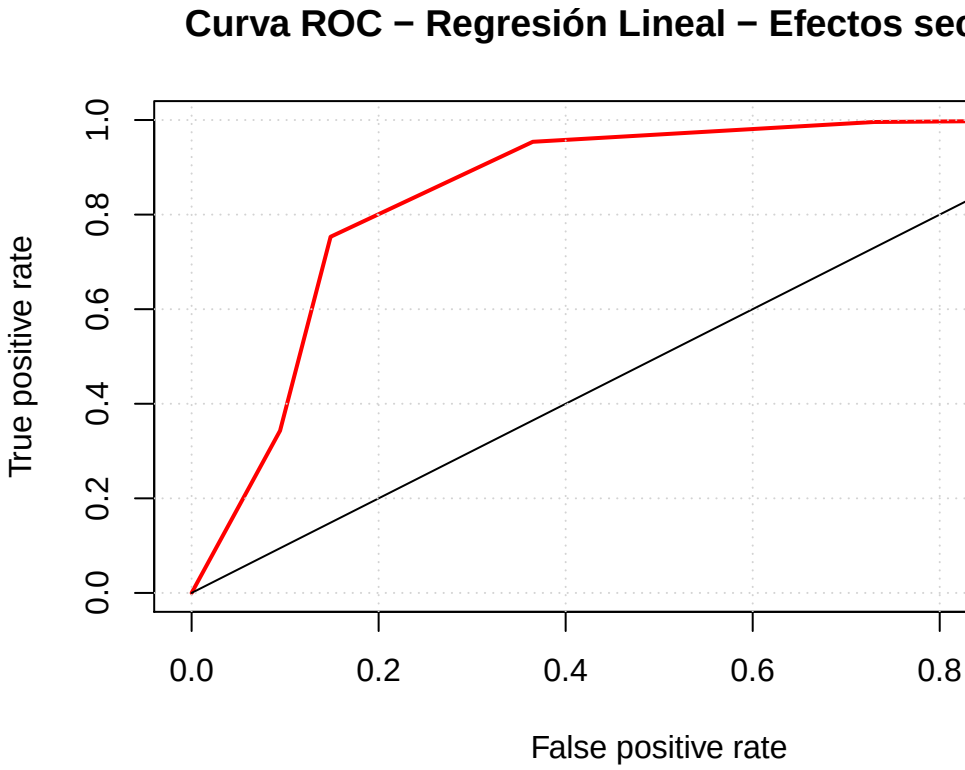
```
# Creamos el modelo para ratingLabel ~ effectivenessNumber
gml_effectiveness = glm(ratingLabel ~ effectivenessNumber, family = binomial(logit), data = data_train_
# Evaluamos el modelo
errorres_regresion_logistica(gml_effectiveness)
```

```
##     real
## pred   0   1
##    0  39   3
##    1  35 236
```

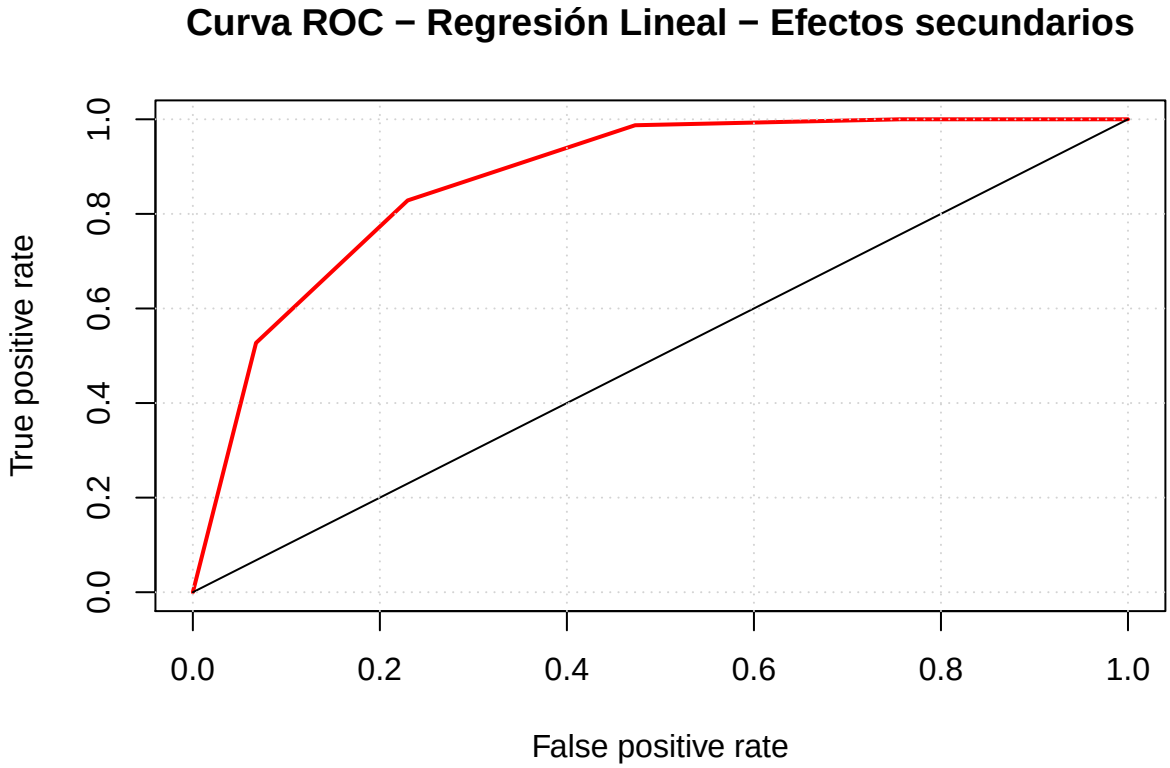
```
## $Etrain
## [1] 10.15936
##
## $Etest
## [1] 12.14058
```

Como se observa se obtiene que hay 3 falsos positivos, es decir, medicamentos que se dicen que son efectivos pero que no lo son. Y existen 35 falsos negativos, es decir, medicamentos que se dicen que no son efectivos pero los son Dichos resultados, según los comentarios de los pacientes. Como podemos apreciar, el error del test nos devuelve un valor de 12.14058, un error aceptable teniendo en cuenta los resultados obtenidos en la matriz de confusión. Pero posiblemente dicho error se pueda reducir aplicando sobre los datos otros modelos que veremos más adelante.

Una vez que hemos obtenidos las prediciones para nuestro modelo, vamos a obtener las probabilidades para el



mismo con el fin de obtener la curva ROC.



5.2.2. Regresión Logística Multiple

Una vez que hemos la regresión logística simple, vamos a intentar predecir el valor del ratingLabel en función de la puntuación de los medicamentos y de los efectos secundarios del mismo. Para ello se va a emplear la

misma (`glm()`).

```
# Creamos el modelo para ratingLabel ~ sideEffectsInverse + effectivenessNumber
modelo_glm_multiple <- glm(ratingLabel ~ sideEffectsInverse + effectivenessNumber, data = data_train_pr
# Evaluamos el modelo
errorres_regresion_logistica(modelo_glm_multiple)
```

```
##     real
## pred   0   1
##    0  50   5
##    1  24 234
```

```
## $Etrain
## [1] 8.366534
##
## $Etest
## [1] 9.265176
```

5.2.3. Regularización en Regresión Logística

En este apartado vamos a ajustar un modelo a través de la regresión logística con regularización meidante la técnica Lasso Regression.

5.2.3.1. Rigde Regression

Esta ténica, en esencia, es la regresión lineal con Weight Decay que utilizamos en la Práctica 2, y cuya fórmula es: $(XTX + lambdaI) - 1 * X\hat{}T$. Es el parámetro lambda, el que en función de su valor, marca el equilibrio entre los componentes de sesgo y varianza. Cuanto mayor sea su valor, mayor sesgo pero menor varianza. Es decir, cuanto mayor valor tenga lambda, mayor penalización y mayor reducción se les aplica a los valores de los coeficientes.

5.2.3.2. Lasso Regression

Esta técnica, además de realizar la misma tarea que la anterior, pero con una fórmula distinta para aplicar la penalización. Aún así, también utiliza el parámetro lambda para concretar el equilibrio entre sesgo y varianza. Además, también es capaz de realizar una selección de variables, anulando algunos coeficientes. Por lo que además de la reducción de los valores de estos coeficientes también reduce el número de estos necesarios para ajustar el modelo.

Por tanto, para aplicar la regularización, primero debemos averiguar el valor de alpha, que representa la técnica a utilizar. Si alpha=0 entonces la regularización se aplica con la técnica Ridge Regression. Si por el contrario alpha=1, entonces la técnica a utilizar será Lasso Regression. Una vez sabemos qué técnica aplicar a través del valor de alpha, tendremos que concretar el valor de lambda, para que dicha técnica pueda ajustar el modelo con su respectiva penalización.

Para ello, vamos a usar la función `train(..)` que es capaz de probar distintos valores para alpha y lambda a través del ajuste de varios modelos, de modo que devuelve el mejor valor de lambda y el mejor valor de alpha. Para ello, como primer argumento debemos indicarle los valores de las etiquetas, que se corresponden con los valores de la variable explicativa. A continuación, vamos a darle la oportunidad de explorar con las columnas sideEffectsInverse + effectivenessNumber. Como segundo argumento le proporcionamos el conjunto de entrenamiento. El tercer argumento especifica el tipo de técnica a utilizar para explorar todos los modelos posibles. Con `glmnet()`, le concretamos que queremos que ajuste dichos modelos a través de la Lasso Regression y de Ridge Regression. Y por último, como cuarto argumento le especificamos la clase de funciones que debe utilizar para ajustar un modelo a través de regresión logística.

```r
library(glmnet)
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-16
```

```r
# Ajustamos un modelo a través de Regresión Logística multiclase
modelbest = train(form=as.factor(ratingLabel) ~ sideEffectsInverse + effectivenessNumber,
                  data=data_train_procesado, method="glmnet", family="binomial")

modelbest$bestTune$alpha
```

```
## [1] 0.1
```

```r
modelbest$bestTune$lambda
```

```
## [1] 0.0004725087
```

Como se puede observar, la función nos dice que el mejor modelo que podemos ajustar para nuestro conjunto de entrenamiento, es el que tiene el valor de alpha 0.1. Con este valor tan cercano a 0 podemos intuir que la técnica que va a emplear es Ridge Regression, ya que esta se representa a través de alpha = 0.

En cuanto al valor de lambda, podemos comprobar que es bastante próximo a 0, y que por tanto, la penalización que va a aplicar Ridge Regression no es demasiado agresiva. Por lo que podemos intuir que el modelo ajustado tendrá un cierto grado de flexibilidad en referencia a las muestras mal clasificadas. También indica, que el modelo que va a ajustar, por el valor pequeño de lambda, se puede caracterizar por un bajo sesgo pero una alta varianza. Lo que puede desembocar en un modelo sobreajustado a los datos de entrenamiento, con una capacidad de generalización bastante escasa.

Para ajustar este modelo hemos utilizado la función `glmnet(..)`. Como primer argumento especificamos una matriz con las muestras de entrenamiento pero sin sus etiquetas, las cuales se le proporcionan a la función en el segundo argumento. Como tercer argumento le volvemos a indicar la familia de funciones que debe utilizar para que sea un modelo ajustado a partir de regresión logística. Los dos siguientes parámetros se corresponden con el mejor alpha y el mejor lambda que hemos obtenido en el proceso anterior.

```r
# Conjunto Train

# Primero generamos una matriz con el conjunto de train separando las etiquetas de los datos
x = model.matrix(ratingLabel~sideEffectsInverse + effectivenessNumber, data_train_procesado)[,-ncol(data
y = data_train_procesado$ratingLabel

# Conjunto Test
x.test = model.matrix(ratingLabel~sideEffectsInverse + effectivenessNumber, data_test_procesado)[,-ncol
y.test = data_test_procesado$ratingLabel

# Reproducimos el modelo ajustado con el mejor lambda y el mejor alpha
ridge.mod = glmnet(x=x, y=y, family="binomial", alpha=modelbest$bestTune$alpha,lambda=modelbest$bestTune

# Calculamos la predicción
predicciones.ridge = predict(ridge.mod, s=ridge.mod$lambda, newx=x.test, type="response")

# Error de regresión con penalización Ridge
error.ridge = mean (( predicciones.ridge - y.test)^2)
cat("Error con la técnica Ridge Regression: ",error.ridge*100,"%\n")
```

```
## Error con la técnica Ridge Regression:  10.37528 %
```

Como se observa, se obtiene un error similar a los anteriores.

5.3. Regresión Polinomial

En algunos casos, la verdadera relación entre la variable respuesta y los predictores puede no ser lineal, por lo que podemos aplicar por ejemplo una regresión polinomial. Una forma simple de incorporar asociaciones no lineales en un modelo lineal es incluir versiones transformadas de los predictores, elevándolos a distintas potencias, evitando un exceso de grados para evitar el sobreajuste o overfitting.

La forma más sencilla de incorporar flexibilidad a un modelo lineal es introduciendo nuevos predictores obtenidos al elevar a distintas potencias el predictor original.

Partiendo del modelo lineal: $Y_i = \beta_0 + \beta_1 X_i + \epsilon$

Se obtiene un modelo polinómico de grado d a partir de la ecuación:

$$Y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + + \beta_d x_i^d + \epsilon_i$$

```
# Hacemos uso del poly para 2
lm_poly = lm(data = data_train_procesado, formula = ratingLabel ~ poly(sideEffectsInverse, 2) + effecti
summary(lm_poly)
```

```
##
## Call:
## lm(formula = ratingLabel ~ poly(sideEffectsInverse, 2) + effectivenessNumber,
##     data = data_train_procesado)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.06537 -0.06628  0.05724  0.09101  1.13669
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    0.19607    0.04170   4.702 3.34e-06 ***
## poly(sideEffectsInverse, 2)1  3.37481    0.27214  12.401  < 2e-16 ***
## poly(sideEffectsInverse, 2)2 -1.82668    0.26304  -6.945 1.19e-11 ***
## effectivenessNumber            0.15638    0.01012  15.456  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.262 on 498 degrees of freedom
## Multiple R-squared:  0.549,  Adjusted R-squared:  0.5463
## F-statistic: 202.1 on 3 and 498 DF,  p-value: < 2.2e-16
```

```
errorres_regresion_lineal(lm_poly)
```

```
##     real
## pred   0   1
##    0  52   6
##    1  22 233
```

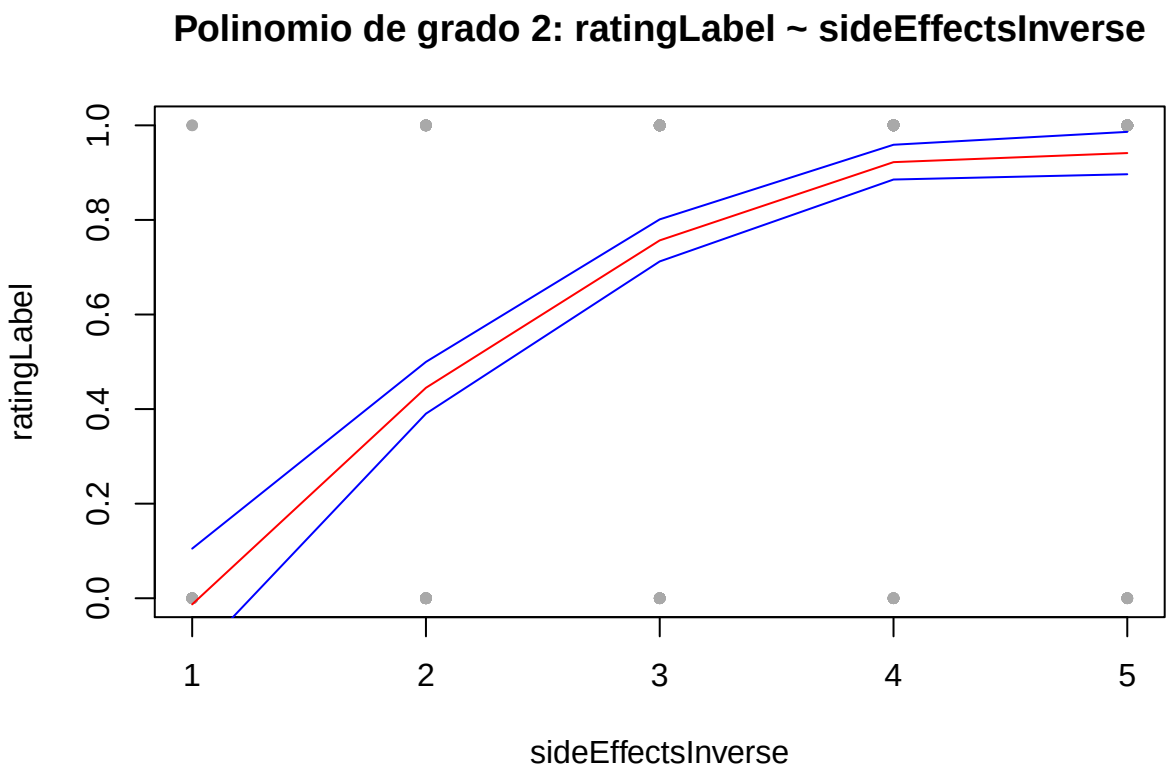
```
## $Etrain
## [1] 7.768924
##
## $Etest
## [1] 8.945687
```

Los p-values individuales de sideEffectsInverse, apuntan a que un polinomio de grado 2 es suficiente para modelar el ratingLabel en función de sideEffectsInverse.

```r
# Calculo del polinomio de grado 2
modelo_poli2 <- lm(data = data_train_procesado, formula = ratingLabel ~ poly(sideEffectsInverse, 2))
summary(modelo_poli2)
```

```
##
## Call:
## lm(formula = ratingLabel ~ poly(sideEffectsInverse, 2), data = data_train_procesado)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.94138  0.05862  0.05862  0.07781  1.01293
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    0.81474    0.01421  57.345  < 2e-16 ***
## poly(sideEffectsInverse, 2)1  4.51518    0.31833  14.184  < 2e-16 ***
## poly(sideEffectsInverse, 2)2 -2.19534    0.31833  -6.897 1.62e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3183 on 499 degrees of freedom
## Multiple R-squared:  0.3327, Adjusted R-squared:   0.33
## F-statistic: 124.4 on 2 and 499 DF,  p-value: < 2.2e-16
```

```r
# Interpolacion de puntos dentro del rango predictos
limites <- range(data_train_procesado$sideEffectsInverse)
nuevos_puntos <- seq(from = limites[1], to = limites[2], by = 1)
nuevos_puntos <- data.frame(sideEffectsInverse = nuevos_puntos)

# Predicción de la variable respuesta y del error estándar
predicciones <- predict(modelo_poli2, newdata = nuevos_puntos, se.fit = TRUE,
                        level = 0.95)

# Cálculo del intervalo de confianza superior e inferior 95%
intervalo_conf <- data.frame(inferior = predicciones$fit -
                                        1.96*predicciones$se.fit,
                             superior = predicciones$fit +
                                        1.96*predicciones$se.fit)

attach(data_train_procesado)
plot(x = sideEffectsInverse, y = ratingLabel, pch = 20, col = "darkgrey")
title("Polinomio de grado 2: ratingLabel ~ sideEffectsInverse")
lines(x = nuevos_puntos$sideEffectsInverse, predicciones$fit, col = "red", pch = 20)
lines(x = nuevos_puntos$sideEffectsInverse, intervalo_conf$inferior, col = "blue", pch = 4)
lines(x = nuevos_puntos$sideEffectsInverse, intervalo_conf$superior, col = "blue", pch = 4)
```



5.4. Conclusiones

Figura 16: Contenido de benefits sin stopwords

effectivenessNumber	weightedRating	ratingLabel	sideEffectsInverse
5	10	0	4
5	8	0	2
5	10	1	5
2	6	0	4
2	4	0	2
1	2	0	2
5	10	1	4
4	8	1	5
5	10	1	5
1	2	0	1
4	8	1	3

Figura 17: Visualización de las columnas a las que aplica regresión

	N (modelo)	S (modelo)
n (real)	Negativos Reales	Falsos Positivos
p (real)	Falsos Negativos	Positivos reales

Figura 18: Matriz de confusión para clasificador binario.