

Tratamiento Inteligente de datos (TID)

Prácticas de la asignatura
2018-2019

En colaboración con:



ugr

Universidad
de Granada

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

Descripción de los paquetes necesarios	1
1. Comprender el problema a resolver	3
2. Preprocesamiento de datos	3
2.1. Lectura del dataset	5
2.1.1. Lectura de datos train	5
2.1.2. Lectura de datos test	5
2.2. Preprocesamiento de los datos	6
2.2.1. Eliminar columnas	6
2.2.2. Eliminar filas	7
2.2.3. Eliminar elementos repetidos (medicamentos)	7
2.2.4. Cuantificación de variables	7
2.2.5. Cálculo del rating ponderado	9
2.2.6. Convertir el rating a variable binaria	10
2.2.7. Cambiar el orden para la columna sideEffectsNumber	10
2.2.8. Representación gráfica de los datos	11
2.2.9. Creación del corpus	13
2.2.10. Correlación	14
2.2.11. Representación gráfica de las frecuencias del Corpus	14
2.2.12. Eliminar signos de puntuación	15
2.2.13. Conversión de las mayúsculas en minúsculas	16
2.2.14. Eliminación de Stopwords	17
2.2.15. Agrupación de sinónimos	17
2.2.16. TF-IDF	19
2.2.17. Stemming	21
2.2.18. Valores perdidos	22
2.2.19. Borrar espacios en blanco innecesarios	23
2.2.20. Sparsity	23
2.2.21. Matriz de documentos de los términos	24
2.2.22. Nube de palabras	26
2.2.23. Convertir a dataframe nuestras modificaciones	26

Índice de figuras

1.	Medicamento "paxil" repetido 20 veces en el conjunto test	7
2.	Clasificación del medicamento por parte del paciente	12
3.	Clasificación de los efectos secundarios del medicamento según el paciente	12
4.	Clasificación de la efectividad del medicamento según el paciente	13
5.	Contenido para benefits_train_corpus I	14
6.	Contenido para benefits_train_corpus II	14
7.	Frecuencia de términos para la columna benefitsReview	15
8.	Frecuencia de términos para la columna sideEffectsReview	15
9.	Contenido de benefits sin signos de puntuación	16
10.	Contenido de effects sin signos de puntuación	16
11.	Contenido de benefits sin mayúsculas	17
12.	Contenido de effects sin mayúsculas	17
13.	Contenido de benefits sin stopwords	17
14.	Contenido de effects sin stopwords	18
15.	Contenido de benefits sin stopwords	27
16.	Contenido de benefits sin stopwords	28

Índice de cuadros

1.	Información del conjunto de datos	3
2.	Información contenida en una fila del conjunto de entrenamiento I	5
3.	Información contenida en una fila del conjunto de entrenamiento II	5
4.	Información contenida en una fila del conjunto de prueba I	6
5.	Información contenida en una fila del conjunto de prueba II	6

Descripción de los paquetes necesarios

A continuación, se describen los paquetes necesarios para el desarrollo 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")`.
- **caret** : Paquete para entrenamiento de clasificación y regresión. Se puede instalar usando : `install.packages("quantda")`.
- **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 conjuntamente 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")`.
- **ggplot2** : Paquete para realizar gráficas. Se puede instalar usando : `install.packages("ggplot2")`.
- **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")`.
- **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")`.
- **quantda** : 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("quantda")`.
- **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")`.

- **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 **biacin** 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	biacin	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, so-me 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 estructuración 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

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 medicamento es ineficaz y la etiqueta con valor 5 a que el medicamento 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.

```
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

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 comparación con las propias valoraciones de los usuarios.

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

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

  # Inicializamos la variable
  sideEffectRating <- 0

  # Convertimos el valor de sideEffect a la misma escala que effectiveness, ya que
  # sideEffect == 1 significa que no tiene efectos secundarios (lo cual es bueno),
  # y effectiveness == 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 realizamos el mismo procedimiento 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 *effectivenessNumber*, 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 *effectivenessNumber* 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 medicamento 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.

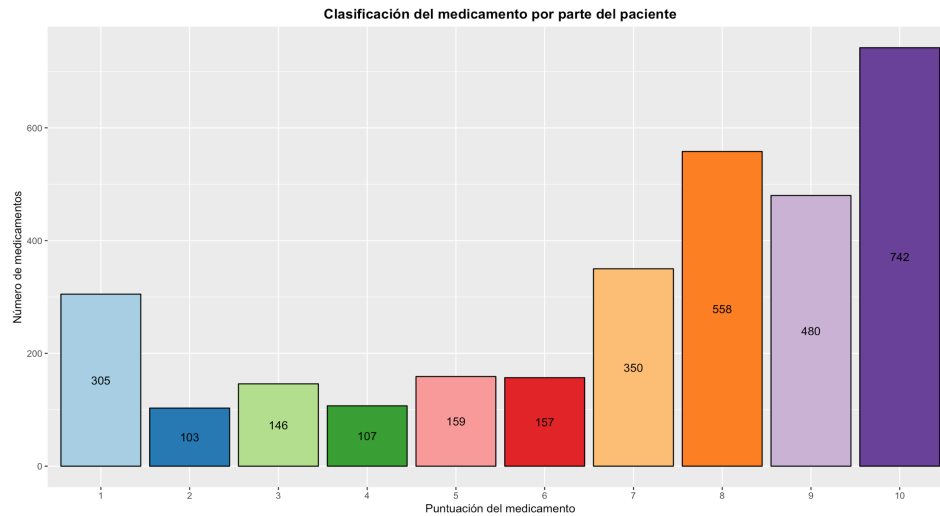


Figura 2: Clasificación del medicamento por parte del paciente

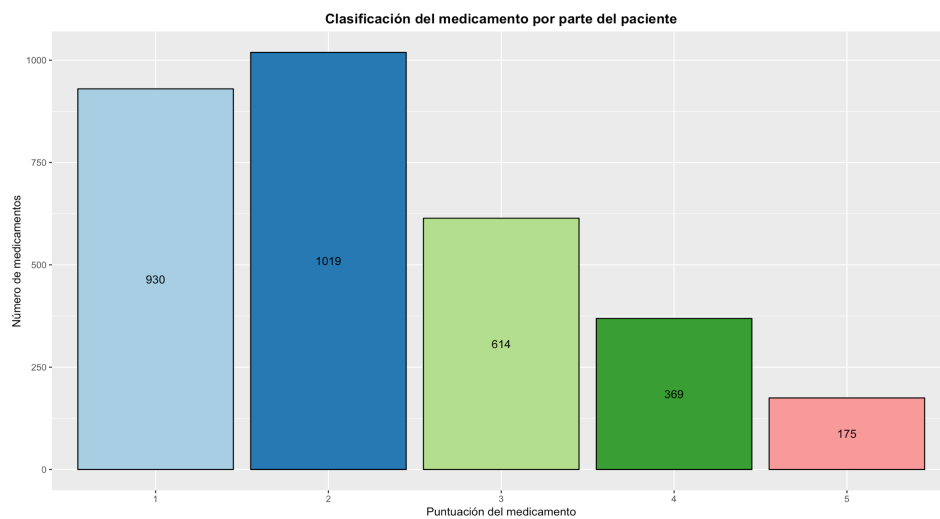


Figura 3: Clasificación de los efectos secundarios del medicamento según el paciente

Por último, vamos a visualizar gráficamente, la efectividad del medicamento 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*.

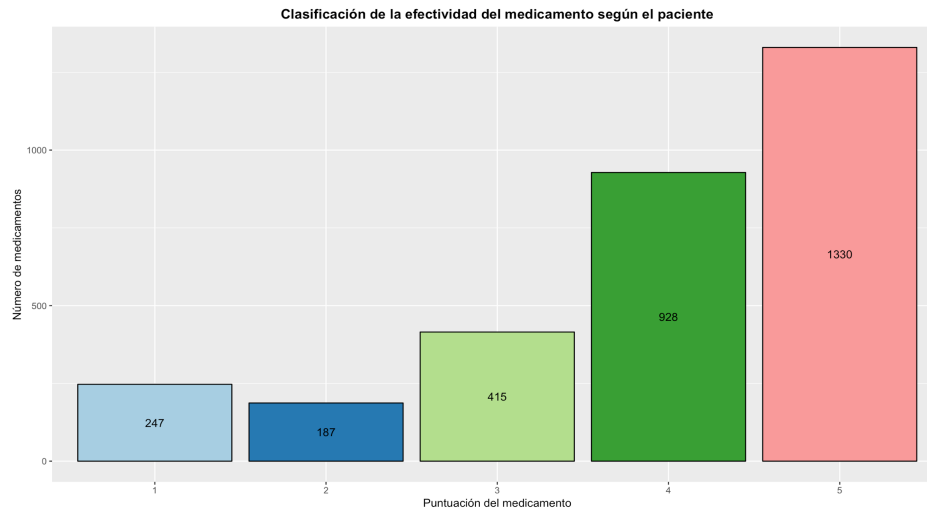


Figura 4: Clasificación de la efectividad del medicamento según el paciente

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 las 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...*

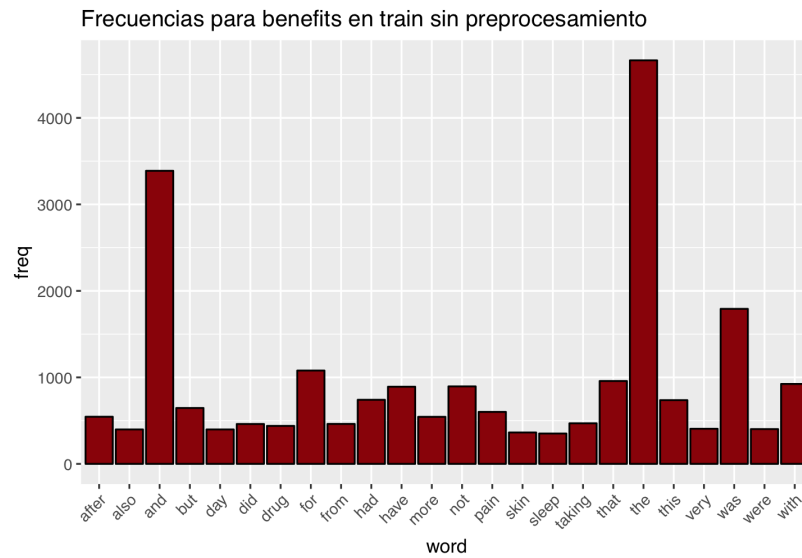


Figura 7: Frecuencia de términos para la columna benefitsReview

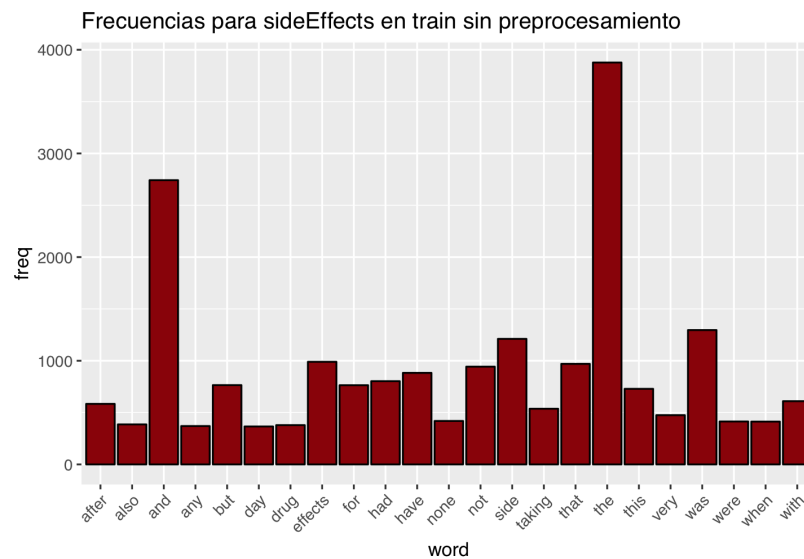


Figura 8: Frecuencia de términos para la columna sideEffectsReview

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 nausea 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 mayúsculas 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 nausea 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

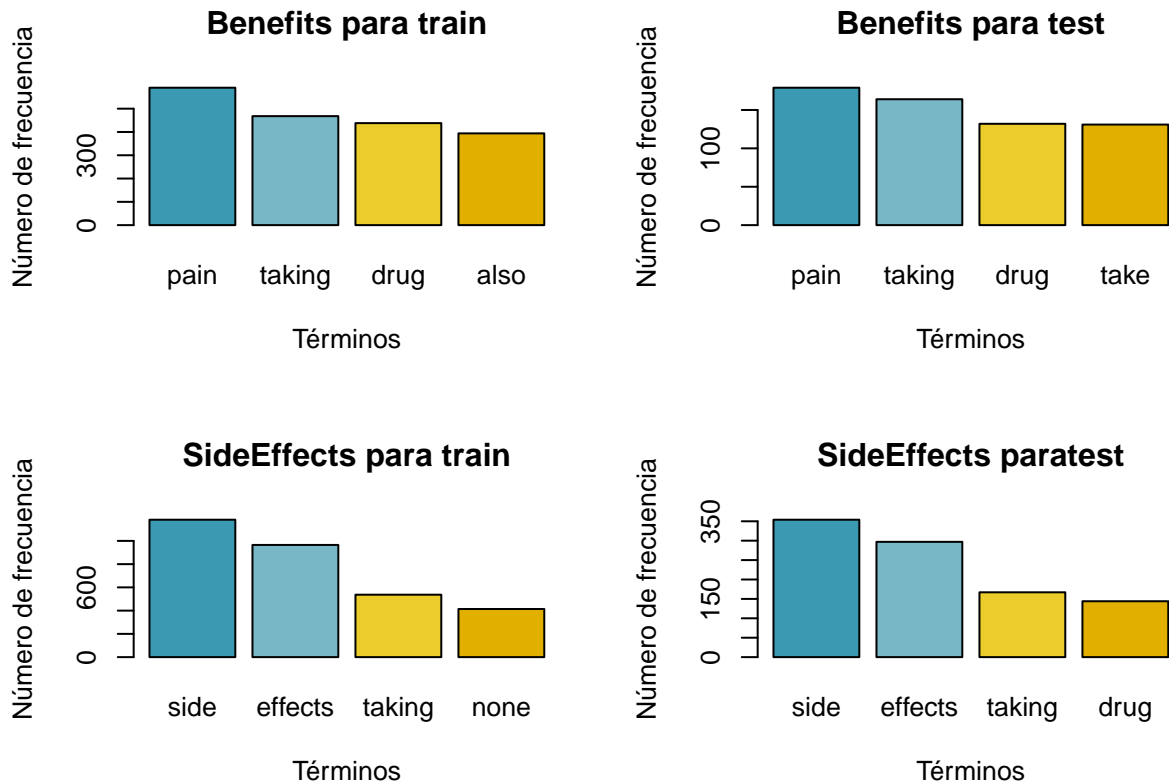
Con el fin de disminuir la dimensión del espacio a trabajar, se pueden identificar palabras distintas con el mismo significado y reemplazarlas por una sola palabra. Para ello se toman los sinónimos de dicha palabra.

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

[1] experiences nausea 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 *sideEffectsReview*) 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 librería `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. Entrenar los datos del fichero, con el fin de obtener los vectores de palabras que nos darán las palabras más similares.
3. Obtener para cada término del documento, la distancia con los términos del fichero, 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 Frequency*). 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:

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D)$$

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 convertimos 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
```

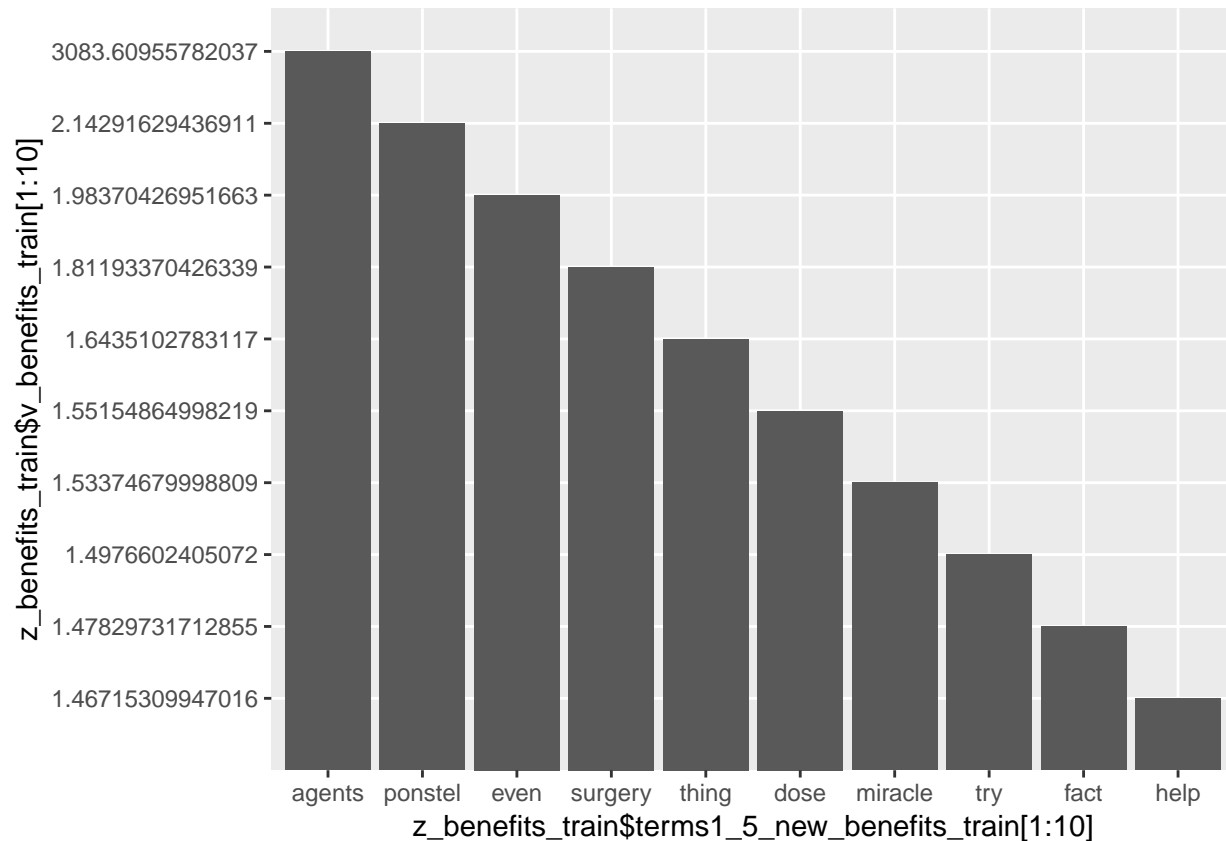
Ahora que tenemos nuestra matriz con el término frecuencia y el peso idf , estamos listos para calcular el peso total de $tf - idf$. Para hacer esta multiplicación de matrices, también tendremos que transformar el vector idf en una matriz diagonal. Ambos cálculos se muestran a continuación.

1. Creamos la matriz diagonal: ($idf_benefits_train <- diag(idf_benefits_train)$)
2. Hacemos la operación para calcular tf_idf como el producto de $tf \cdot idf$: $tf_idf_benefits_train <- crossprod(tf_benefits_train, idf_benefits_train)$
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 respir
```

```
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] overall 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$content!="")]
effects_train_corpus_new_load<-effects_train_corpus_new_load[which(effects_train_corpus_new_load$content!="")]
```

```
benefits_test_corpus_new_load<-benefits_test_corpus_new_load[which(benefits_test_corpus_new_load$content
effects_test_corpus_new_load<-effects_test_corpus_new_load[which(effects_test_corpus_new_load$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()`.

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

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    0    5    1    1    1
## 1189   4    0      0    0    0    0    0    5    1    1    1
## 1279   0    0      1    3    2    0    0    0    1    1    1
## 1824   1    0      0    3    1    0    0    0    1    2    0
## 2504   4    1      3    0    1    4    0    2    0    0    3
## 317    7    2      0    1    1    1    0    3    4    0    0
## 339    1    1      5    1    2    6    0    2    0    0    1
## 462    4    1      3    0    1    4    0    2    0    0    3
## 565    3    2      2    3    0    0    6    4    0    0    1
## 665    3    4      0    3    0    0    0    1    1    0    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 i -ésima 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

# Ordenamos 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)

# Ordenamos 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, freq), para posteriormente graficarlo.
ordena_mayor_matrix_corpus <- data.frame(palabra = names(ordena_mayor_matrix_corpus), freq = ordena_mayor_matrix_corpus)

```

Mostramos las más frecuentes:

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

```

##      palabra freq
## 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:

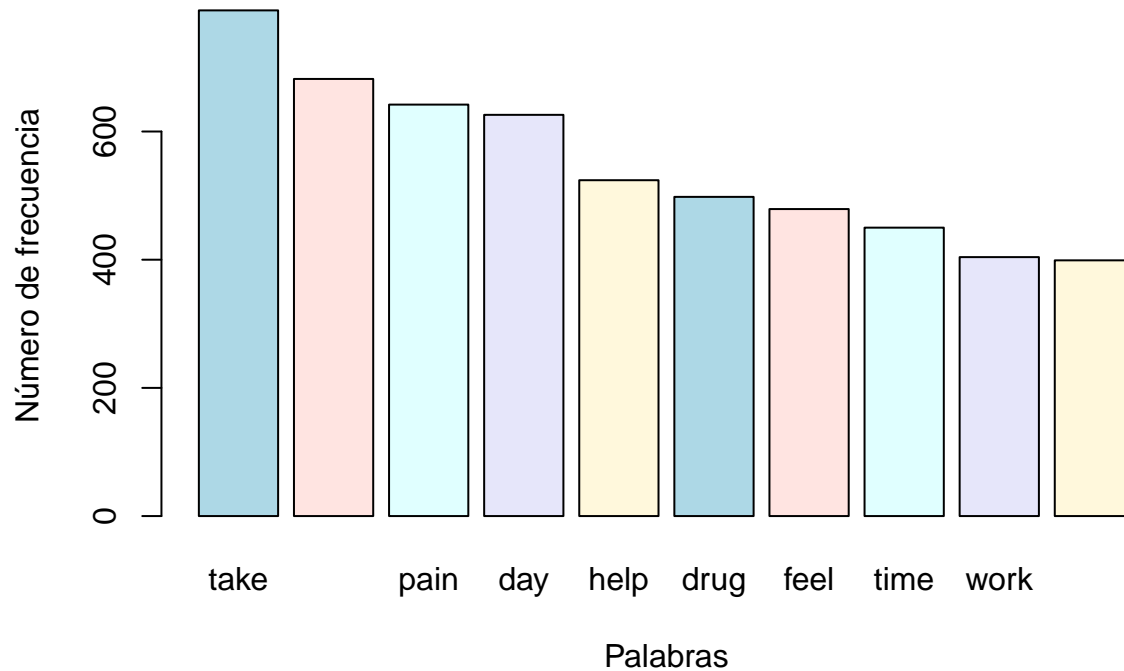
```

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))
```

Las diez palabras más frecuentes después del preprocesamiento



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.



