

# 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](mailto:alejandroac79@correo.ugr.es)

Gema Correa Fernández: [gecorrea@correo.ugr.es](mailto:gecorrea@correo.ugr.es)

Luis Gallego Quero: [lgag94@correo.ugr.es](mailto:lgag94@correo.ugr.es)

Jonathan Martín Valera: [jmv742@correo.ugr.es](mailto:jmv742@correo.ugr.es)

Andrea Morales Garzón: [andreamgmg@correo.ugr.es](mailto:andreamgmg@correo.ugr.es)

# Índice

<b>Descripción de los paquetes necesarios</b>	<b>1</b>
<b>1. Comprender el problema a resolver</b>	<b>2</b>
<b>2. Preprocesamiento de datos</b>	<b>2</b>
2.1. Lectura de datos . . . . .	3
2.2. Procesar datos . . . . .	6
2.2.1. Eliminar columnas . . . . .	6
2.2.2. Creación del corpus . . . . .	7
2.2.3. Eliminar signos de puntuación . . . . .	8
2.2.4. Conversión de las mayúsculas en minúsculas . . . . .	8
2.2.5. Agrupación de sinónimos: . . . . .	8
2.2.6. Eliminación de Stopwords . . . . .	9
2.2.6. Stemming . . . . .	9
2.2.8. Borrar espacios en blanco innecesarios . . . . .	10
<b>Term Document Matrix</b>	<b>10</b>
Gráficas . . . . .	12

## Índice de figuras

## Índice de cuadros

1.	Relevancia de requisitos para preselección de candidatos. . . . .	2
----	---	---

## Descripción de los paquetes necesarios

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

- **tm** : Paquete específico para minería de datos, permite procesar datos de tipo texto. Se puede instalar usando : *install.packages("tm")*.
- **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")*.
- **wordcloud** : Paquete para crear gráficas de nubes de palabras, permitiendo visualizar las diferencias y similitudes entre documentos. Se puede instalar usando : *install.packages("wordcloud")*.
- **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")*.

## 1. Comprender el problema a resolver

Para la realización y aplicación de las técnicas explicadas a lo largo del curso, se ha seleccionado el *dataset* **Drug Review Dataset**, proporcionado por *UCI Machine Learning Repository*. Dicho *dataset* contiene una exhaustiva base de datos de medicamentos específicos, en donde, el conjunto de datos proporciona revisiones de pacientes sobre medicamentos específicos para unas condiciones particulares. Las 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, una calificación en base a los efectos secundarios y otra a la efectividad del medicamento.

En este proyecto nos centraremos en el **análisis y experiencia de los usuarios**, obtenido mediante la ingesta de ciertos medicamentos. Para ello, se proponen los siguientes objetivos principales del estudio:

- Realizar un análisis de sentimientos en relación con la experiencia en el uso de dichos medicamentos, como por ejemplo ver la efectividad del medicamento con los efectos secundarios o beneficios del mismo.
- 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:

<b>Características del Data Set</b>	Multivariable, texto
<b>Características de los atributos</b>	Discreto, texto
<b>Tareas asociadas</b>	Clasificación, regresión, Clustering
<b>Número de instancias</b>	4143
<b>Número de atributos</b>	8
<b>Valores vacíos</b>	N/A
<b>Área</b>	N/A
<b>Fecha de donación</b>	10/02/2018
<b>Veces visualizado</b>	9130

Cuadro 1: Relevancia de requisitos para preselección de candidatos.

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
2. **rating** (numerical): clasificación de 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 puntos para la aplicación de técnicas. Para poder analizar el *dataset* y realizar el preprocesamiento al mismo, lo primero que se va hacer es leer el conjunto de datos *train* y *test*.

## 2.1. Lectura de datos

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

```
# Visualizar las 5 primeras filas para los datos train
head(datos_train, 5)
```

```
##      X      urlDrugName rating      effectiveness      sideEffects
## 1 2202      enalapril      4      Highly Effective      Mild Side Effects
## 2 3117 ortho-tri-cyclen      1      Highly Effective      Severe Side Effects
## 3 1146      ponstel      10      Highly Effective      No Side Effects
## 4 3947      prilosec      3      Marginally Effective      Mild Side Effects
## 5 1951      lyrica      2      Marginally Effective      Severe Side Effects
##
##      condition
## 1 management of congestive heart failure
## 2      birth prevention
## 3      menstrual cramps
## 4      acid reflux
## 5      fibromyalgia
##
## 1
## 2
## 3
## 4 The acid reflux went away for a few months after just a few days of being on the drug. The heartbu
## 5
##
## 1      cough, hypotension , proteinuria, imp
## 2 Heavy Cycle, Cramps, Hot Flashes, Fatigue, Long Lasting Cycles. It's only been 5 1/2 months, but i
## 3
## 4
## 5
##
## 1
## 2
## 3 I took 2 pills at the onset of my menstrual cramps and then every 8-12 hours took 1 pill as needed
## 4
## 5
```

```
# Resumen sobre los datos train
summary(datos_train)
```

```
##      X      urlDrugName      rating
## Min.   : 0      lexapro : 63      Min.   : 1.000
## 1st Qu.:1062      prozac  : 46      1st Qu.: 5.000
## Median :2092      retin-a : 45      Median : 8.000
## Mean   :2081      zolofit : 45      Mean   : 7.006
## 3rd Qu.:3092      paxil   : 38      3rd Qu.: 9.000
## Max.   :4161      propecia: 38      Max.   :10.000
##
##      (Other) :2832
##
##      effectiveness      sideEffects
## Considerably Effective: 928      Extremely Severe Side Effects: 175
## Highly Effective      :1330      Mild Side Effects      :1019
## Ineffective           : 247      Moderate Side Effects   : 614
```

```

## Marginally Effective : 187   No Side Effects           : 930
## Moderately Effective : 415   Severe Side Effects      : 369
##
##
##           condition
## depression      : 236
## acne            : 165
## anxiety         : 63
## insomnia        : 54
## birth control   : 49
## high blood pressure: 42
## (Other)         :2498
##
## none
## None
## NONE
## None.
## The treatment benefits were marginal at best. Mood neither improved nor deteriorated, and anxiety v
## Before the use of vagifem tablets, I had to endure a series of urinary infections after sometimes p
## (Other)
##           sideEffectsReview      commentsReview
## none                : 112      n/a                : 7
## None                : 73       none               : 6
## None.               : 19       None             : 4
## No side effects.    : 9        .                : 3
## There were no side effects.: 6    One tablet once a day: 3
## no side effects     : 5        (Other)           :3083
## (Other)             :2883      NA's              : 1

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

# Visualizar las 5 primeras filas para los datos test
head(datos_test, 5)

##      X urlDrugName rating      effectiveness      sideEffects
## 1 1366      biaxin      9 Considerably Effective Mild Side Effects
## 2 3724    lamictal      9      Highly Effective Mild Side Effects
## 3 3824    depakene      4 Moderately Effective Severe Side Effects
## 4 969      sarafem     10      Highly Effective No Side Effects
## 5 696      accutane     10      Highly Effective Mild Side Effects
##           condition
## 1      sinus infection
## 2      bipolar disorder
## 3      bipolar disorder
## 4 bi-polar / anxiety
## 5      nodular acne
##
## 1
## 2 Lamictal stabilized my serious mood swings. One minute I was clawing up the walls in pure mania, t
## 3
## 4
## 5
##

```



```
## 1
## 2 Drowsiness, a bit of mental numbness. If you take too much, you will feel sedated. Since you have
## 3
## 4
## 5
##
## 1
## 2
## 3 Depakote was prescribed to me by a Kaiser psychiatrist in Pleasant Hill, CA in 2006. The medication
## 4
## 5
```

```
# información sobre los datos test
summary(datos_test)
```

```
##           X           urlDrugName           rating
## Min.      : 1.0    paxil          : 20    Min.      : 1.000
## 1st Qu.: 968.2    effexor-xr: 17    1st Qu.: 5.000
## Median :2048.0    accutane   : 16    Median : 8.000
## Mean    :2085.4    synthroid  : 15    Mean    : 6.767
## 3rd Qu.:3199.8    differin   : 13    3rd Qu.: 9.000
## Max.     :4157.0    effexor    : 13    Max.     :10.000
##              (Other) :942
##              effectiveness              sideEffects
## Considerably Effective:310  Extremely Severe Side Effects: 80
## Highly Effective          :411  Mild Side Effects              :330
## Ineffective               : 82  Moderate Side Effects          :236
## Marginally Effective      : 76  No Side Effects                :268
## Moderately Effective      :157  Severe Side Effects            :122
##
##
##              condition
## depression      : 66
## acne            : 46
## anxiety          : 27
## insomnia        : 21
## high blood pressure: 20
## birth control    : 19
## (Other)         :837
##
## none
## None
## elevation of mood and clarity of thought. Progress stalled out at 300 mg, but with increase to 450
## I've only been on it for a week but I've noticed a change already. I am more awake and it seems as
## The benefits of using Tretinoin were great. First of all I noticed that my skin started glowing and
## !0 years after spinal stenosis, scar-tissue and additional narrowing of nerve canals causes sever i
## (Other)
##
## none
## None
## None.
## none at 300 mg. Possible tinnitus from increase to 450 mg, not evaluated by an audiologist yet tho
## dryness
## luckily I did not notice any negative side effects. The positive effects that I noticed out weighed
## (Other)
```

```
##
## Initial treatment included therapy and Lexapro in addition to Wellbutrin XL. Now only on the Wellbutrin XL.
## My doctor added Abilify to my 60 mg of Cymbalta because I was feeling really fatigued and unable to sleep.
## My treatment details are as follows: I Used Avita (Tretinoin) every night after cleansing my face. I also used
## none
## see above
## ?
## (Other)
```

```
View(datos_train)    # vista de la tabla
View(datos_test)     # vista de la tabla
```

## 2.2. Procesar datos

### 2.2.1. Eliminar columnas

#### 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, 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 el de train, para que los resultados sean consistentes.

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

#### Eliminar columna de commentsReview

Consideramos que este tipo de información en las instancias de nuestros datos no es de nuestro interés. En este atributo se almacena texto, los consumidores de las drogas suelen poner en la mayoría de casos la frecuencia con la que consumen la misma. En otros casos menos frecuentes, se establecen comentarios más arbitrarios en el que muestran sus sensaciones o información sin relevancia. Incluso en algunos casos este campo aparece vacío.

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

Una vez eliminadas las columnas anteriores, 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. La más conocida se llama **tm**, aunque también haremos uso del paquete **SnowballC** para realizar el *Stemming*. Si no tenemos instaladas las librerías:

```
# Paquete para minería de datos, permite procesar datos de tipo texto
library("tm")
```

```
## Loading required package: NLP
```

```
# Paquete para minería de datos, agrupa aquellos términos que contienen la misma raíz
library("SnowballC")
```

### 2.2.2. 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. Sólo nos faltaría convertir cada elemento del vector al formato de documento. Podemos usar la función `VectorSource` para hacer esta conversión.

```
# 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_review_data = as.vector(datos_train$benefitsReview)
effects_review_data = as.vector(datos_train$sideEffectsReview)

# Lo convertimos en la estructura de documento, y lo guardamos ya en el corpus
# que lo vamos a utilizar
benefits_corpus = (VectorSource(benefits_review_data))
effects_corpus = (VectorSource(effects_review_data))

# Creamos el propio corpus
benefits_corpus <- Corpus(benefits_corpus)
effects_corpus <- Corpus(effects_corpus)
```

Podemos ver que funciona accediendo a uno cualquiera, por ejemplo, vamos a acceder al cuarto comentario almacenado de la columna comentarios.

```
# Si nos fijamos en el contenido, vemos que tiene signos de puntuación y exclamación
inspect(benefits_corpus[4])
```

```
## <<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 heart
benefits_corpus[[4]]$content
```

```
## [1] "The acid reflux went away for a few months after just a few days of being on the drug. The heart"
```

Tenemos los documentos relacionados con los efectos aislados en un contenedor distinto. Por ejemplo, podemos acceder al séptimo comentario de este tipo y comprobar su contenido.

```
# Si nos fijamos en el contenido, vemos que tiene signos de puntuación y exclamación
inspect(effects_corpus[7])
```

```
## <<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,
effects_corpus[[7]]$content
```

```
## [1] "a few experiences of nausea, heavy moodswings on the days I do not take it, decreased appetite"
```

### 2.2.3. 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.

```
# Una vez que tenemos el corpus creado, continuamos con el procesamiento
benefits_corpus <- tm_map(benefits_corpus, content_transformer(removePunctuation))
effects_corpus <- tm_map(effects_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.

```
inspect(benefits_corpus[4])
```

```
## <<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 heartb
```

Ocurre lo mismo con el comentario de efectos número siete.

```
inspect(effects_corpus[7])
```

```
## <<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
```

### 2.2.4. Conversión de las mayúsculas en minúsculas

Para poder hacer uso de los términos por igual, debemos convertir las mayúsculas en minúsculas.

```
benefits_corpus <- tm_map(benefits_corpus, content_transformer(tolower))
inspect(benefits_corpus[4])
```

```
## <<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 heartb
```

### 2.2.5. Agrupación de sinónimos:

Si queremos hacer uso del paquete wordnet, se debe descargar

<https://stackoverflow.com/questions/25129073/wordnet-getdict-could-not-find-wordnet-dictionary>

```
# No funciona, parece que hay que instalar algo en Java
#install.packages('wordnet')
#Sys.setenv(WNHOME = file.path("/", "WordNet-3.0/dict"))
#library(wordnet)
#setDict("/usr/local/Cellar/wordnet/3.1")
#initDict()
```

```
#getDict()
# synonyms('write', "VERB")

# Para que funcione el archivo tiene que estar en .bin
#library(devtools)
#install_github("mukul13/rword2vec")
#library(rword2vec)
#ls("package:rword2vec")
#dist=distance(file_name = "vec.bin",search_word = "king",num = 10)
# convert Stata to SPSS
#library("magrittr")
#library("rio")
#convert("datos/drugLibTrain_raw.tsv", "datos/drugLibTrain.bin")
#dist
```

### 2.2.6. Eliminación de Stopwords

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

```
benefits_corpus <- tm_map(benefits_corpus, content_transformer(removeWords),
                          stopwords("english"))
inspect(benefits_corpus[4])
```

```
## <<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
```

Ahora ya hemos eliminado las stopwords de forma correcta.

### 2.2.6. 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_corpus[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_corpus[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_corpus <- tm_map(benefits_corpus, 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_corpus[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_corpus[213])
```

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

### 2.2.8. 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 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_corpus <- tm_map(benefits_corpus, stripWhitespace)
```

## Term Document Matrix

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_corpus)
```

Podemos observar que tenemos 5838 términos, esto quiere decir que tenemos 5838 palabras diferentes en nuestro Corpus, ## Frecuencia de 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] 5838 3107
```

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*.

```
# suma las filas
```

```
suma_matrix_corpus <- rowSums(matrix_corpus)
head(suma_matrix_corpus,5)
```

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

```
# los ordena 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    643    626    524    498    479    450    404    399
```

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

```
# los ordena 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[,2])
```

Creamos nube de palabras:

```
# instalar paquete wordcloud
```

```
#wordcloud(
#  words = ordena_mayor_matrix_corpus$palabra,
#  freq = ordena_mayor_matrix_corpus$freq,
#  max.words = 80,
#  random.order = F,
#  colors=brewer.pal(name = "Dark2", n = 8)
# )
```

Mostramos las más frecuentes:

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

```
##           palabra freq
## take           take  789
## effect          effect 682
## pain            pain  643
## 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
```

## Gráficas

```
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", font = 4))
```



***Las diez palabras más frecuentes***