R en el análisis literario en español

Diego Giménez

Códigos en R utilizados para análisis literario a distancia en español (Giménez, 2024). Este documento actualiza y amplía una versión anterior desarrollada por Diego Giménez y Andressa Gomide en 2022, que se centraba en el análisis del “Libro del desasosiego”. En esta nueva versión, se analizan obras de Miguel de Cervantes, incluyendo “Don Quijote”, “Los trabajos de Persiles y Sigismunda”, “Novelas ejemplares”, “El Cerco de Numancia” y “Galatea”.

Herramientas y preparación de los datos

Instalación

Quanteda (Quantitative Analysis of Textual Data) es un paquete de R para la manipulación y análisis de datos textuales.

La instalación de R varía según el sistema operativo (ej.: Windows, Mac, Linux), así como sus diferentes versiones. Hay varias fuentes donde se pueden obtener instrucciones actualizadas sobre cómo instalar R (ej.: https://didatica.tech/como-instalar-a-linguagem-r-e-o-rstudio/). El Comprehensive R Archive Network (CRAN), la red oficial de distribución de R, ofrece instrucciones fiables para ello, aunque quizás no tan detalladas como en otras fuentes.

Otra sugerencia es instalar una interfaz gráfica de usuario, del inglés Graphical User Interface (GUI). Las GUIs facilitan considerablemente la interacción del usuario con el ordenador. (RStudio) es la GUI más utilizada para R y, al igual que R, es gratuita y de código abierto.

1.2 Configuración: preparando el entorno

Al reutilizar códigos, es una buena práctica estar atento a la versión instalada tanto de R como de las bibliotecas utilizadas. No es necesario que las versiones sean las mismas que las utilizadas durante la creación de los códigos; sin embargo, en algunos casos, puede no haber compatibilidad entre versiones diferentes y algunas funciones o paquetes pueden haber sido descontinuados. Este artículo fue escrito utilizando la versión 4.3.3 de R.

```
# Verificar la versión de R
R.version.string
```

```
## [1] "R version 4.3.3 (2024-02-29 ucrt)"
```

Para nuestro análisis, utilizaremos algunos paquetes ya existentes. Estos paquetes no son más que extensiones para R que normalmente contienen datos o códigos. Para utilizarlos, necesitamos instalarlos en el ordenador, si aún no se ha hecho, y cargarlos en R. Una ventaja de cargar solo los paquetes necesarios (en lugar de todos los paquetes instalados) es evitar procesamiento computacional innecesario. El código a continuación crea una lista de los paquetes utilizados en el presente análisis y los carga, instalando los que aún no estaban presentes.

```
# Listamos los paquetes que necesitamos

packages = c("quanteda", # aAnálisis cuantitativo de datos textuales.
             "quanteda.textmodels", # Complementa a Quanteda, proporcionando funcionalidades es
pecíficas para la modelización de texto.

             "quanteda.textstats", # Este paquete contiene funciones para calcular estadísticas
descriptivas y medidas de complejidad de texto, como la diversidad léxica y la densidad léxica.

             "quanteda.textplots", # Este paquete ofrece herramientas para la visualización de
datos textuales, incluyendo gráficos de dispersión de palabras, nubes de palabras y mapas de ca
```

```
lor.

             "newsmap", # Para clasificar documentos, basado en "seed words", es decir, palabra
s clave predefinidas que indican tópicos o categorías.
             "readtext", # Para leer diferentes formatos de texto.
             "spacyr", # Para anotación de clases gramaticales, reconocimiento de entidades y a
notación sintáctica (Python debe estar instalado).
             "ggplot2", # Para gráficos simples de las frecuencias.
             "seededlda", # Para modelización de tópicos.
             "stringr", # Para las expresiones regulares.
             "dplyr",  # Este paquete es parte del tidyverse y ofrece un conjunto de funciones
para la manipulación de datos tabulares en R, permitiendo realizar operaciones como filtrado, s
elección, agregación y unión de datos de forma sencilla y eficiente.
             "tidytext", # Este paquete complementa al tidyverse, proporcionando herramientas p
ara el análisis de texto en conjunto con los principios de organización de datos del tidyverse,
permitiendo integrar fácilmente análisis de texto en pipelines de análisis de datos.
             "knitr", # Este paquete se utiliza para la producción de documentos dinámicos en
R, permitiendo integrar código R y resultados de análisis en documentos Markdown, HTML, PDF y o
tros formatos.
             "stringr", # Este paquete proporciona funciones para la manipulación de cadenas de
texto en R, facilitando tareas como la coincidencia de patrones, la extracción de subcadenas y
la manipulación de texto.
             "igraph", # Este paquete se utiliza para el análisis y visualización de redes en
R, ofreciendo funciones para crear, manipular y representar grafos y redes complejas.
             "topicmodels" # Este paquete se utiliza para la modelización de tópicos en textos,
ofreciendo implementaciones de algoritmos como LDA (Latent Dirichlet Allocation) y LSA (Latent
Semantic Analysis) para la inferencia de tópicos en colecciones de documentos.
             )

# Instalamos (si es necesario) y cargamos los paquetes
package.check <- lapply(
  packages,
  FUN = function(x) {
    if (!require(x, character.only = TRUE)) {
      install.packages(x, dependencies = TRUE)
      require(x, character.only = TRUE)
    }
  }
)
```

Los códigos a continuación se implementaron en la versión 4.0.2 de Quanteda. Utilizar una versión diferente puede resultar en errores o resultados indeseados. Para verificar la versión de los paquetes, empleamos la función `packageVersion`. Para verificar la versión de R, utilizamos `R.version.string`.

```
# Verificar la versión de Quanteda
packageVersion("quanteda")
```

```
## [1] '4.0.2'
```

Por último, necesitamos establecer cuál será nuestro directorio de trabajo. Este será el lugar donde se guardarán los resultados. Para identificar cuál es el directorio de trabajo actual, utilizamos la función getwd(), que devuelve la ruta absoluta, es decir, la dirección completa del directorio. Para definir un nuevo lugar de trabajo, utilizamos la función setwd(). Archivos guardados en ese directorio pueden ser leídos solo con la indicación del nombre del archivo, pues podemos utilizar la ruta relativa, es decir, la dirección del archivo a partir del directorio en el que estamos trabajando.

1.3 Datos

Una vez instalados los paquetes necesarios, se puede proceder al análisis del corpus. Para ello, necesitamos cargar el corpus en R. Si estamos trabajando con datos almacenados localmente, es decir, disponibles en el ordenador donde se realizarán los análisis, basta con utilizar la función readtext(), indicando la ubicación (relativa o absoluta) del archivo deseado.

El libro “Don Quijote” puede ser leído como un archivo único,

```
# Para leer un archivo único con todo el contenido del libro
Don_Quijote <- readtext("~/corpora/Don Quijote.txt", encoding = "utf-8")

# Devuelve la estructura del objeto creado
str(Don_Quijote)
```

```
## Classes 'readtext' and 'data.frame': 1 obs. of  2 variables:
##  $ doc_id: chr "Don Quijote.txt"
##  $ text  : chr "El ingenioso hidalgo don Quijote de la Mancha\n\n\n\npor Miguel de Cervantes
Saavedra\n\n\n\n\n\nEl ingenioso h"| __truncated__
```

O considerando el libro como una unidad dentro de un corpus formado por varios documentos:

```
# Leer todos los archivos en la carpeta ldod del directorio corpora
cervantes_files <- readtext("~/corpora/cervantes", encoding = "utf-8")

# Devolver la estructura del objeto creado
str(cervantes_files)
```

```
## Classes 'readtext' and 'data.frame': 5 obs. of  2 variables:
##  $ doc_id: chr  "Don Quijote.txt" "El Cerco de Numancia.txt" "Galatea.txt" "Los trabajos de
Persiles y Sigismunda.txt" ...
##  $ text  : chr  "El ingenioso hidalgo don Quijote de la Mancha\n\n\n\npor Miguel de Cervante
s Saavedra\n\n\n\n\n\nEl ingenioso h"| __truncated__ "EL CERCO DE NUMANCIA\n\nFIGURAS SIGUIENTE
S:\n\nCIPIÓN, romano.\nIUGURTA, romano.\n_Gayo_ MARIO, romano.\nQUINTO"| __truncated__ "Yo, Mig
uel de Ondarza Zavala, escribano de cámara de Su Majestad, de los que residen en el su Consejo,
doy fe q"| __truncated__ "HISTORIA DE LOS TRABAJOS DE PERSILES Y SIGISMUNDA\n\n\nLIBRO I\n\nCAP
ITULO XXII\n\n_Donde el capitán da cuenta "| __truncated__ ...
```

Los textos anteriores derivan de la obra de Cervantes, disponible en el Proyecto Gutenberg Projeto Gutenberg.

Los archivos se guardaron con la codificación utf-8 y se eliminaron la información para-textual y editorial (como notas de los editores) que pudieran interferir en la investigación automática del software.

Los análisis a continuación se demostrarán utilizando los dos corpus, en diferentes momentos.

1.3.1 Limpieza

La limpieza a continuación se aplicó solo a los textos guardados por separado (cervantes_files). El archivo con el libro en un único texto (Don_Quijote) ya había sido limpiado anteriormente.

```
# Creamos una copia para recuperar el original en caso de que haya errores en la regex
cervantes_clean <- cervantes_files

## Eliminación de los elementos indeseados

# Eliminar números al inicio de líneas (índices)
cervantes_clean$text <- str_replace_all(cervantes_clean$text, "\\n\\d", "\n")
```

```
# Eliminar fechas
cervantes_clean$text <- str_replace_all(cervantes_clean$text, "\\d{1,2}-(\\d{1,2}|[IVX]{1,4})-19\\d{2}", "")
```

1.4 Investigaciones con Quanteda

Después de que los archivos se cargan en el sistema, necesitamos crear un objeto “corpus”, es decir, el formato necesario para que Quanteda pueda procesar y generar información sobre el(los) texto(s). Para ello, basta con aplicar la función `corpus` . Automáticamente, el texto se segmenta en tokens y frases. Los tokens corresponden a todas las ocurrencias (incluyendo repeticiones) de palabras, así como otros elementos como puntuación, números y símbolos. Al investigar el corpus con la función `summary` , obtenemos el recuento de frases, tokens y types (el número de tokens distintos en un corpus).

```
# Crear el corpus de varios archivos
corpus_clean <- corpus(cervantes_clean)
# ver un resumen del corpus
summary(corpus_clean)
```

	Text <chr>	Types <int>	Tokens <int>	Sentences <int>
1	Don Quijote.txt	24509	450642	7250
2	El Cerco de Numancia.txt	2513	9409	489
3	Galatea.txt	11618	139533	2611
4	Los trabajos de Persiles y Sigismunda.txt	3356	15863	266
5	Novelas Ejemplares.txt	16706	217842	3476

5 rows

```
# Crear corpus del archivo único
corpus_unico <- corpus(Don_Quijote)
# Presentar un resumen del corpus
summary(corpus_unico)
```

	Text <chr>	Types <int>	Tokens <int>	Sentences <int>
1	Don Quijote.txt	24509	450642	7250

1 row

Si es necesario, podemos alterar la estructura de nuestro corpus. En el `corpus_unico` , tenemos un corpus hecho con solo un texto. Con `corpus_reshape` podemos crear un nuevo corpus en el que cada frase se considere un texto, es decir, una unidad.

```
# Revelar el número de textos en el corpus
ndoc(corpus_unico)
```

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

```
# Reestructurar el corpus, convirtiendo cada frase en una unidad
corpus_sents <- corpus_reshape(corpus_unico, to = "sentences")

# Presentar un resumen del corpus
summary(corpus_sents)
```

	Text <chr>	Types <int>	Tokens <int>	Sentences <int>
1	Don Quijote.txt.1	601	2550	1
2	Don Quijote.txt.2	24	28	1
3	Don Quijote.txt.3	5	5	1
4	Don Quijote.txt.4	27	28	1
5	Don Quijote.txt.5	18	25	1
6	Don Quijote.txt.6	8	8	1
7	Don Quijote.txt.7	86	132	1
8	Don Quijote.txt.8	86	144	1
9	Don Quijote.txt.9	22	36	1
10	Don Quijote.txt.10	82	146	1

1-10 of 100 rows Previous **1** 2 3 4 5 6 … 10 Next

```
# Número total de unidades en la nueva estructura del corpus
ndoc(corpus_sents)
```

```
## [1] 7250
```

Los ejemplos anteriores nos muestran que un corpus es un conjunto de textos con información sobre cada texto (metadatos), de los cuales podemos extraer fácilmente el recuento de tokens, types y frases para cada texto. Sin embargo, para realizar análisis cuantitativos en el corpus, necesitamos dividir los textos en tokens (tokenización). También es posible filtrarlos, eliminando elementos como puntuación, símbolos, números, URLs y separadores.

```
# Tokenizar nuestros tres corpus

toks_unico <- tokens(corpus_unico)
toks_sents <- tokens(corpus_sents)
toks_files <- tokens(corpus_clean)

## A continuación, filtramos los tres corpus de diversas formas, para demostración

# Eliminar puntuación (corpus limpio con regex)
toks_nopunct_files <- tokens(corpus_clean, remove_punct = TRUE)
toks_nopunct_unico <- tokens(corpus_unico, remove_punct = TRUE)
```

```
# Eliminar números (corpus con solo un archivo)
toks_nonumbr <- tokens(corpus_unico, remove_numbers = TRUE)

# Eliminar separadores (categorías Unicode "Separator" [Z] y "Control" [C]) (corpus hecho por f
rases)
toks_nosept <- tokens(corpus_sents, remove_separators = TRUE)

# Eliminar varios elementos al mismo tiempo (corpus con solo un archivo)
toks_simples <- tokens(corpus_unico, remove_numbers = TRUE, remove_symbols = TRUE, remove_punct
= TRUE)
```

También es posible eliminar tokens no deseados. Quanteda ofrece una lista de 'stopwords' para diferentes lenguas. Las stopwords, o palabras vacías en español, son palabras que se deben eliminar durante el procesamiento de textos para análisis computacionales. No existe una lista estándar, pero generalmente las stopwords son las palabras más frecuentemente utilizadas en una lengua, como preposiciones y artículos. El bloque a continuación elimina las palabras incluidas en la lista de stopwords para el español y también incluye otras palabras que se repiten en el corpus en cuestión.

```
# Eliminar stopwords del corpus hecho con un solo archivo
toks_nostop <- tokens_select(toks_unico, pattern = stopwords("es"), selection = "remove")

# Eliminar tokens específicos del corpus hecho con varios archivos y limpio con regex, después
de eliminar las puntuaciones
toks_selected_files <- tokens_select(toks_nopunct_files, pattern = c("dijo", "tan", "así", "res
pondió", "sino", "pues", "dos", "aunque", "aquel", "alguna", "aquella", "cómo", "algún", "aun",
"oh", "á", "si","mas", stopwords("es")), selection = "remove")

# Eliminar tokens específicos del corpus hecho con un archivo, después de eliminar las puntuaci
ones
toks_selected_unico <- tokens_select(toks_nopunct_unico, pattern = c("dijo", "tan", "así", "res
pondió", "sino", "pues", "dos", "aunque", "aquel", "alguna", "aquella", "cómo", "algún", "aun",
"oh", "á", "si", "mas", stopwords("es")), selection = "remove")
```

Después de la tokenización, el siguiente paso es crear una tabla con la frecuencia de cada token por cada texto, o, en los términos de Quanteda, una document-feature-matrix (DFM). La DFM es un requisito previo para varias otras funciones en Quanteda, como es el caso de la topfeatures, que devuelve los tokens más frecuentes en un corpus.

```
# aAquí podemos ver las 20 palabras más frecuentes cuando eliminamos
# Números, símbolos y puntuación
dfm_simples <- dfm(toks_simples)
print("Con eliminación de números, símbolos y puntuación")
```

```
## [1] "Con eliminación de números, símbolos y puntuación"
```

```
topfeatures(dfm_simples, 20)
```

```
##   que    de     y    la     a    en    el    no   los    se   con   por    lo
## 20769 18410 18272 10492  9875  8285  8265  6346  4769  4752  4275  3945  3492
##   las    le    su   don   del    me  como
##  3486  3420  3388  2714  2536  2345  2270
```

```
dfm_nostop <- dfm(toks_nostop)
print("remoção de stopwords")
```

```
## [1] "remoção de stopwords"
```

```
topfeatures(dfm_nostop, 20)
```

```
##         ,         .         —         ;       don   quijote    sancho         :
##     40277      8208      6923      4802      2714      2241      2174      2047
##        si      dijo       tan     señor       así respondió       ser      bien
##      1968      1808      1245      1065      1065      1063      1059      1051
##         ?         ¿    merced      pues
##       960       959       900       865
```

```
dfm_selected_unico <- dfm(toks_selected_unico)
print("Eliminación de tokens seleccionados en el corpus previamente limpiado con regex y sin st
opwords")
```

```
## [1] "Eliminación de tokens seleccionados en el corpus previamente limpiado con regex y sin s
topwords"
```

```
topfeatures(dfm_selected_unico, 20)
```

```
##       don   quijote    sancho     señor       ser      bien    merced caballero
##      2714      2241      2174      1065      1059      1051       900       677
##     decir     hacer      dios    señora      aquí       mal      cosa      buen
##       578       535       531       518       516       463       447       446
##    verdad       tal      allí       ver
##       436       428       421       410
```

```
dfm_selected_files <- dfm(toks_selected_files)
print("Eliminación de tokens seleccionados en el corpus de archivo único y sin stopwords")
```

```
## [1] "Eliminación de tokens seleccionados en el corpus de archivo único y sin stopwords"
```

```
topfeatures(dfm_selected_files, 20)
```

```
##       don   quijote    sancho       ser      bien     señor    merced     decir
##      2754      2242      2185      1971      1935      1510      1063       972
##     hacer      vida caballero       mal     luego      aquí      allí      cosa
##       955       890       864       848       842       838       837       822
##    señora       tal       ver     habia
##       819       813       787       768
```

Después de generar la lista de tokens, podemos explorar el corpus. Una de las técnicas más simples y utilizadas para la investigación de corpus es a través de las líneas de concordancia, también conocidas como concordance lines o keywords in context (kwic). Las líneas de concordancia muestran fragmentos del corpus donde ocurren los términos buscados. El número de palabras en el contexto puede ser estipulado por el usuario, siendo 5 tokens a la izquierda y 5 a la derecha el estándar. La primera columna indica el nombre del archivo donde la

palabra buscada ocurre. Existen varias opciones para búsquedas. Pueden hacerse por palabras o por fragmentos, secuencias o combinaciones de las mismas.

```
# Ocurrencias de palabras que empiezan con "feli"
kwic(toks_unico, pattern =  "feli*")
```

docname <chr>	from <int>	to <int>	pre <chr>	keyword <chr>
Don Quijote.txt	168	168	con otros sucesos dignos de	felice
Don Quijote.txt	956	956	deceplinantes , a quien dio	felice
Don Quijote.txt	1087	1087	y otros sucesos dignos de	felice
Don Quijote.txt	1310	1310	don Quijote , con la	felicemente
Don Quijote.txt	1408	1408	Mancha , a quien dio	felice
Don Quijote.txt	5130	5130	su dístico : Donec eris	felix
Don Quijote.txt	8339	8339	los que compuso el famoso	Feliciano
Don Quijote.txt	17013	17013	, pareciéndole que había dado	felicísimo
Don Quijote.txt	22466	22466	que trata de la sabia	Felicia
Don Quijote.txt	23254	23254	de España , y fue	felicísimo

1-10 of 69 rows | 1-5 of 7 columns Previous **1** 2 3 4 5 6 7 Next

```
# Podemos también buscar más de una palabra al mismo tiempo
kwic(toks_unico, pattern = c("feli*", "alegr*"))
```

docname <chr>	from <int>	to <int>	pre <chr>	keyword <chr>
Don Quijote.txt	168	168	con otros sucesos dignos de	felice
Don Quijote.txt	956	956	deceplinantes , a quien dio	felice
Don Quijote.txt	1087	1087	y otros sucesos dignos de	felice
Don Quijote.txt	1310	1310	don Quijote , con la	felicemente
Don Quijote.txt	1408	1408	Mancha , a quien dio	felice
Don Quijote.txt	5130	5130	su dístico : Donec eris	felix
Don Quijote.txt	6849	6849	la Peña Pobre , de	alegre
Don Quijote.txt	7154	7154	no envidiara , y fuera	alegre
Don Quijote.txt	8339	8339	los que compuso el famoso	Feliciano
Don Quijote.txt	17013	17013	, pareciéndole que había dado	felicísimo

1-10 of 196 rows | 1-5 of 7 columns Previous 1 2 3 4 5 6 … 20 Next

```
# Por secuencia de más de un token
kwic(toks_unico, pattern = phrase("me fal*"))
```

docname <chr>	from <int>	to <int>	pre <chr>	keyword <chr>
Don Quijote.txt	15961	15962	descuidado , que cada día	me falta
Don Quijote.txt	37673	37674	— La del Señor no	me falte
Don Quijote.txt	59812	59813	mira cuántas muelas y dientes	me faltan
Don Quijote.txt	60306	60307	Y las alforjas que hoy	me faltan
Don Quijote.txt	60340	60341	Quijote . — Sí que	me faltan
Don Quijote.txt	60739	60740	bien cuántos dientes y muelas	me faltan
Don Quijote.txt	74464	74465	a la corte . También	me falta
Don Quijote.txt	92622	92623	don Quijote — . Ahora	me falta
Don Quijote.txt	104555	104556	y apriesa , temeroso no	me faltase
Don Quijote.txt	106795	106796	y así , aunque entonces	me falte

1-10 of 20 rows | 1-5 of 7 columns Previous 1 2 Next

1.4.1 N-gramas

Las listas de frecuencia de palabras pueden ser útiles para identificar elementos comunes en un texto. Sin embargo, en muchos casos, es igualmente importante saber en qué contexto se encuentran esas palabras. Identificar qué palabras coocurren frecuentemente en un corpus puede proporcionarnos aún más información sobre el texto. Por ejemplo, saber que la secuencia ‘estoy triste’ ocurre frecuentemente en el corpus nos proporciona información más rica que solo la frecuencia de la palabra ‘triste’ aislada. La secuencia ‘estoy triste’ es un ejemplo de lo que llamamos n-grams, o, en este caso específico, bigramas. Los n-grams son secuencias de dos o más palabras que ocurren en un texto. Para generar listas de n-grams, partimos de una lista de tokens y especificamos el número mínimo y máximo de tokens en cada n-grama.

```
# Crear una lista de bigramas, trigramas y tetragramas
toks_ngram <- tokens_ngrams(toks_simples, n = 2:4)
# Visualizar solo los 30 más frecuentes
head(toks_ngram[[1]], 30)
```

```
##  [1] "El_ingenioso"       "ingenioso_hidalgo"  "hidalgo_don"
##  [4] "don_Quijote"        "Quijote_de"         "de_la"
##  [7] "la_Mancha"          "Mancha_por"         "por_Miguel"
## [10] "Miguel_de"          "de_Cervantes"       "Cervantes_Saavedra"
## [13] "Saavedra_El"        "El_ingenioso"       "ingenioso_hidalgo"
## [16] "hidalgo_don"        "don_Quijote"        "Quijote_de"
## [19] "de_la"              "la_Mancha"          "Mancha_Tasa"
## [22] "Tasa_Testimonio"    "Testimonio_de"      "de_las"
```

```
## [25] "las_erratas"        "erratas_El"         "El_Rey"
## [28] "Rey_Al"             "Al_Duque"           "Duque_de"
```

1.4.2 Diccionario

Otra forma de extraer información de un texto es a través de la creación de "diccionarios". La función `dictionary` en Quanteda permite agrupar tokens por categorías. Esta categorización puede entonces ser utilizada para búsquedas en el corpus. Por ejemplo, podemos crear las categorías "alegría" y "tristeza" conteniendo palabras relacionadas con esos sentimientos, respectivamente. Con el diccionario creado, podemos identificar la distribución de esos términos en un corpus.

```
# Creación de diccionario a partir del corpus formado por un único documento
dict <- dictionary(list(alegria = c("alegr*", "allegr*", "feli*", "content*"),
                        tristeza = c("trist*", "infeli*")))

dict_toks <- tokens_lookup(toks_unico, dictionary = dict)
print(dict_toks)
```

```
## Tokens consisting of 1 document.
## Don Quijote.txt :
##  [1] "alegria"  "alegria"  "alegria"  "alegria"  "alegria"  "tristeza"
##  [7] "alegria"  "alegria"  "alegria"  "alegria"  "alegria"  "tristeza"
## [ ... and 540 more ]
```

```
dfm(dict_toks)
```

```
## Document-feature matrix of: 1 document, 2 features (0.00% sparse) and 0 docvars.
##                  features
## docs              alegria tristeza
##   Don Quijote.txt     403      149
```

```
# Creación de diccionario a partir del corpus formado por varios documentos
dict <- dictionary(list(alegria = c("alegr*", "allegr*", "feli*", "content*"),
                        tristeza = c("trist*", "infeli*")))

dict_toks <- tokens_lookup(toks_files, dictionary = dict)
print(dict_toks)
```

```
## Tokens consisting of 5 documents.
## Don Quijote.txt :
##  [1] "alegria"  "alegria"  "alegria"  "alegria"  "alegria"  "tristeza"
##  [7] "alegria"  "alegria"  "alegria"  "alegria"  "alegria"  "tristeza"
## [ ... and 540 more ]
##
## El Cerco de Numancia.txt :
##  [1] "tristeza" "tristeza" "tristeza" "tristeza" "alegria"  "tristeza"
##  [7] "tristeza" "tristeza" "tristeza" "tristeza" "tristeza" "tristeza"
## [ ... and 10 more ]
##
## Galatea.txt :
##  [1] "alegria"  "alegria"  "tristeza" "alegria"  "tristeza" "alegria"
##  [7] "alegria"  "alegria"  "alegria"  "alegria"  "tristeza" "alegria"
## [ ... and 431 more ]
```

```
##
## Los trabajos de Persiles y Sigismunda.txt :
##  [1] "alegria"  "alegria"  "alegria"  "alegria"  "alegria"  "alegria"
##  [7] "tristeza" "alegria"  "alegria"  "alegria"  "tristeza" "alegria"
## [ ... and 13 more ]
##
## Novelas Ejemplares.txt :
##  [1] "alegria" "alegria" "alegria" "alegria" "alegria" "alegria" "alegria"
##  [8] "alegria" "alegria" "alegria" "alegria" "alegria"
## [ ... and 302 more ]
```

```
dfm(dict_toks)
```

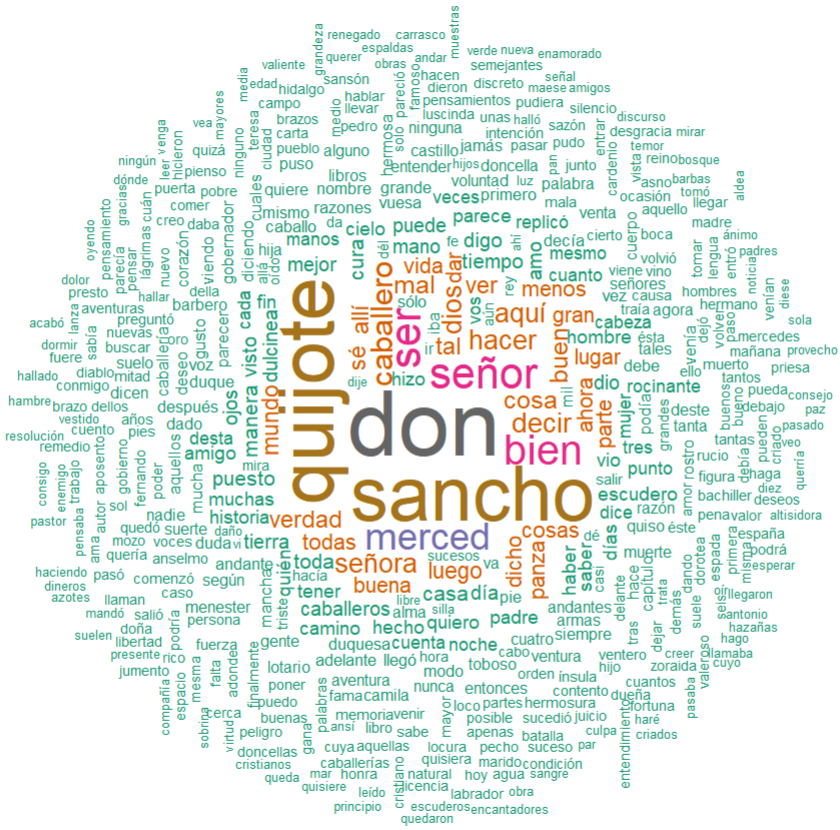
```
## Document-feature matrix of: 5 documents, 2 features (0.00% sparse) and 0 docvars.
##                                        features
## docs                                    alegria tristeza
##   Don Quijote.txt                           403      149
##   El Cerco de Numancia.txt                    4       18
##   Galatea.txt                               324      119
##   Los trabajos de Persiles y Sigismunda.txt  23        2
##   Novelas Ejemplares.txt                    272       42
```

2 VVisualización y análisis de los datos

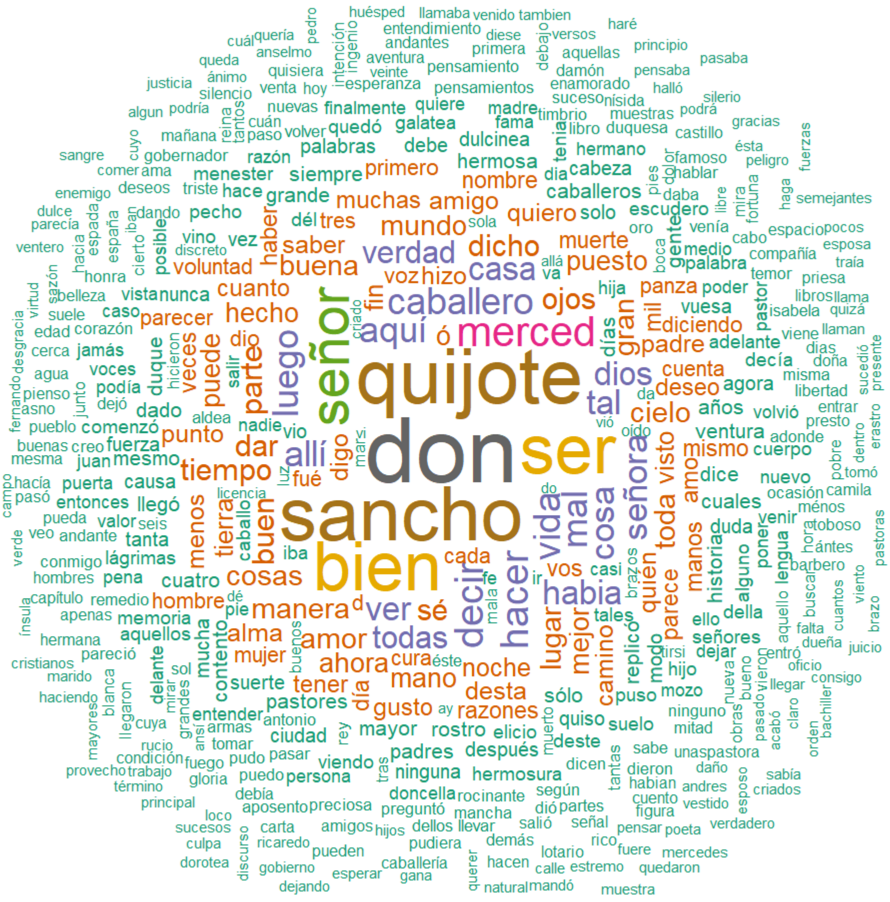
2.1 Nube de palabras y gráfico de frecuencia

En 1.4, creamos una DFM con la frecuencia de los tokens. Para absorber estas frecuencias de forma más rápida, podemos generar visualizaciones. Una opción es la nube de palabras, un gráfico que permite la rápida visualización de los términos más frecuentes.

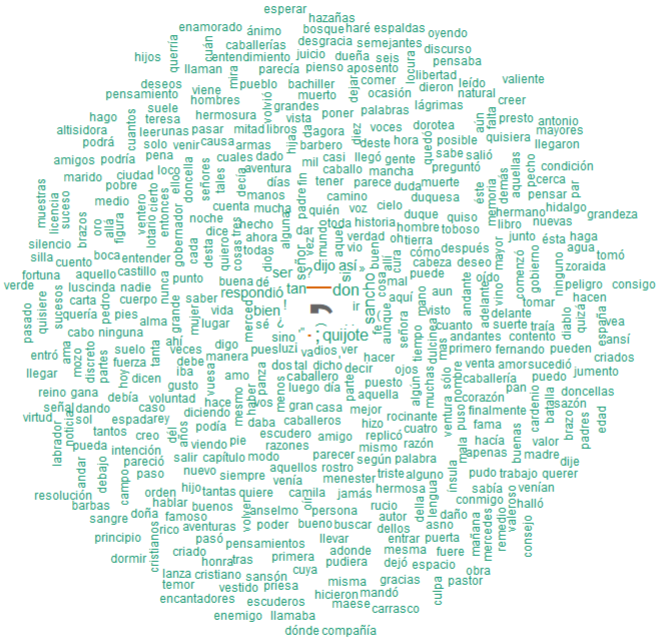
```
# Demostración de cómo cambian las frecuencias de palabras según la preparación del corpus corp
us
set.seed(100) #para reprodução dos resultados
textplot_wordcloud(dfm_selected_unico, min_count = 6, random_order = FALSE, rotation = .25, col
or = RColorBrewer::brewer.pal(8, "Dark2"))
```




```
set.seed(100)
textplot_wordcloud(dfm_selected_files, min_count = 6, random_order = FALSE, rotation = .25, col
or = RColorBrewer::brewer.pal(8, "Dark2"))
```



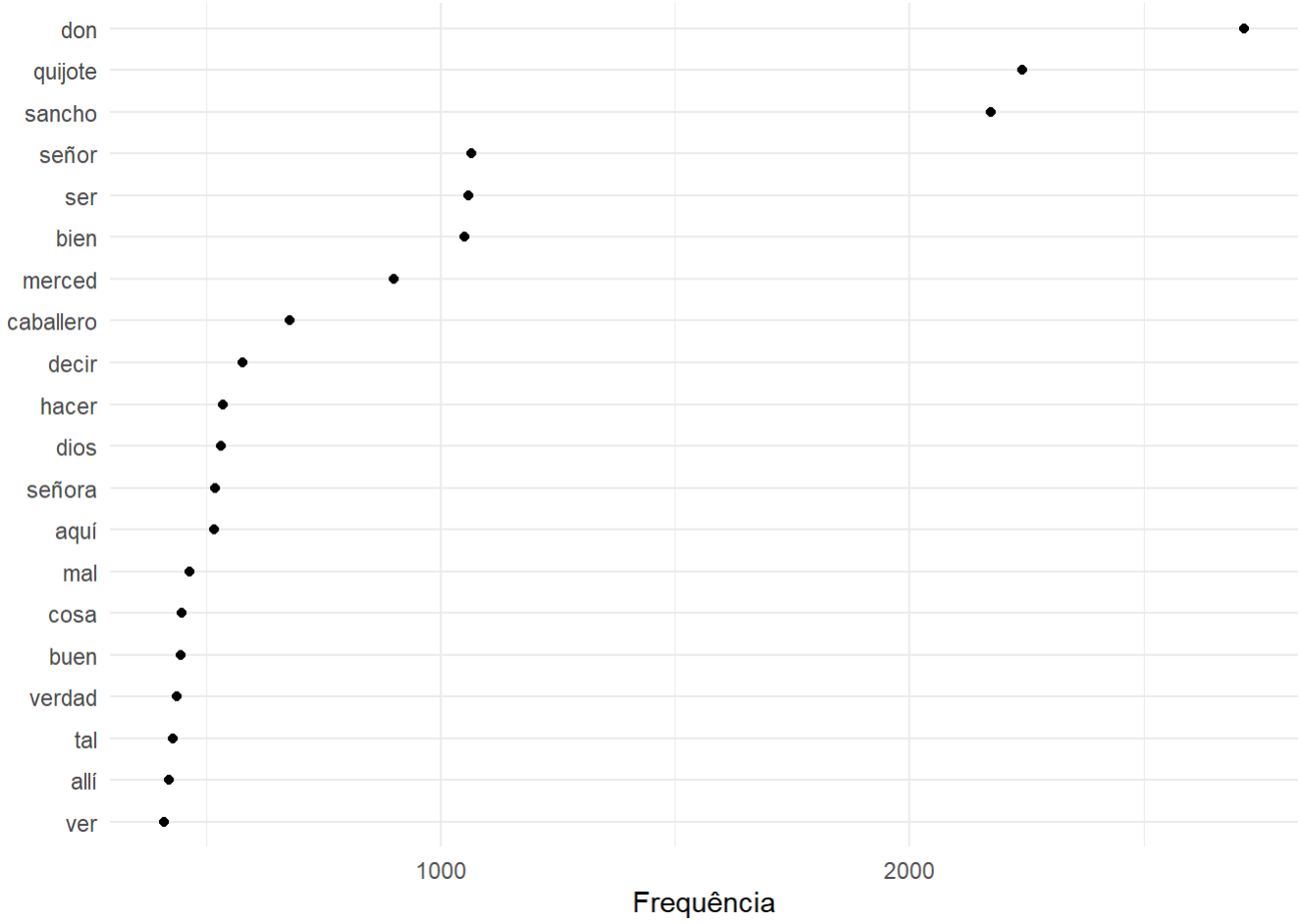
```
set.seed(100)
textplot_wordcloud(dfm_nostop, min_count = 6, random_order = FALSE, rotation = .25, color = RColorBrewer::brewer.pal(8, "Dark2"))
```



Otra solución es utilizar la biblioteca ggplot y representar en un gráfico el número de ocurrencias de las palabras más frecuentes.

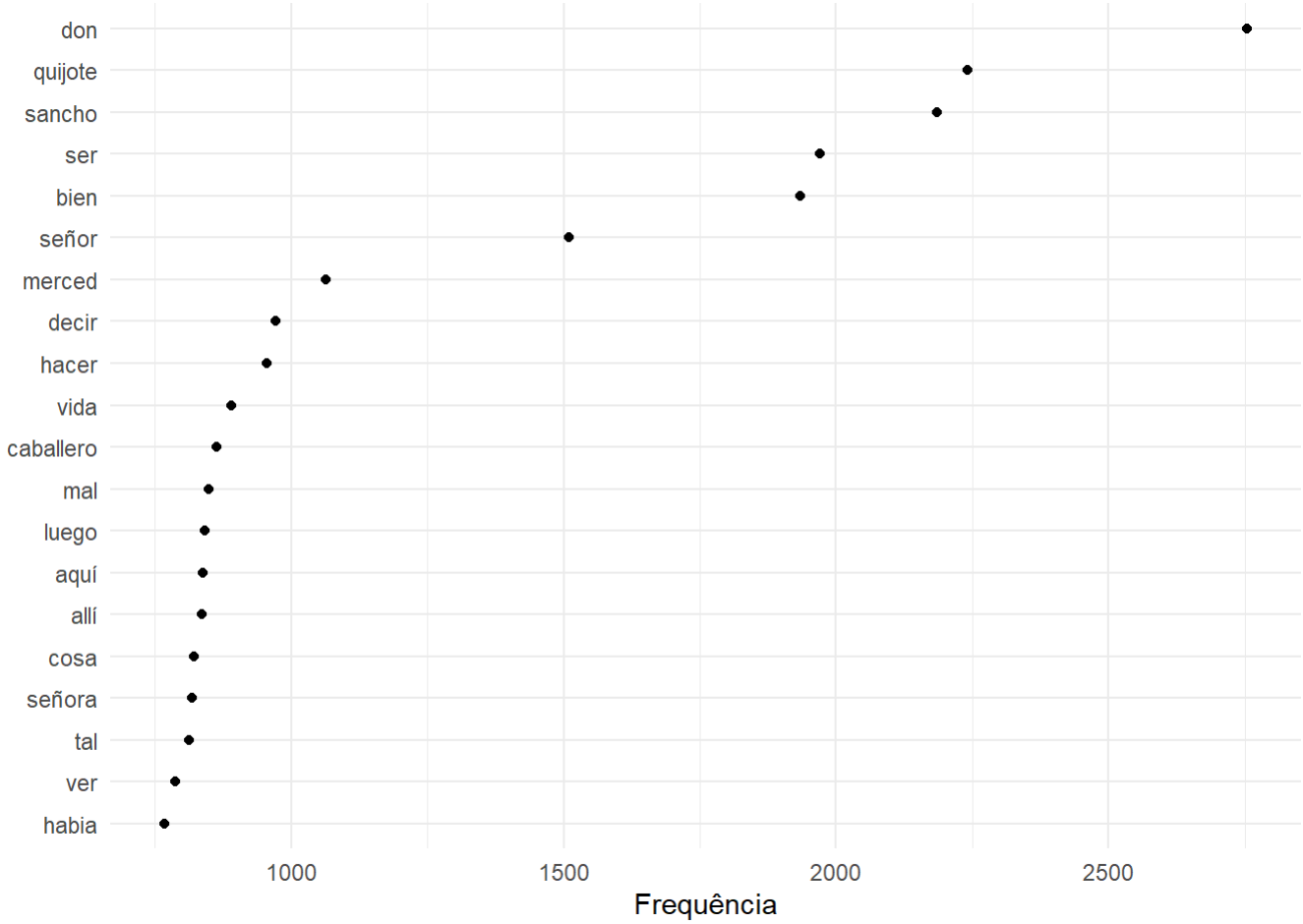
```
# A partir del corpus formado por un único documento

dfm_selected_unico %>%
  textstat_frequency(n = 20) %>%
  ggplot(aes(x = reorder(feature, frequency), y = frequency)) +
  geom_point() +
  coord_flip() +
  labs(x = NULL, y = "Frequência") +
  theme_minimal()
```




```
# A partir de un corpus formado por varios documentos

dfm_selected_files %>%
  textstat_frequency(n = 20) %>%
  ggplot(aes(x = reorder(feature, frequency), y = frequency)) +
  geom_point() +
  coord_flip() +
  labs(x = NULL, y = "Frequência") +
  theme_minimal()
```



2.2 Modelado de tópicos (LDA)

Otra función frecuentemente utilizada en el Procesamiento de Lenguaje Natural (PLN) es el modelado de tópicos, también conocido como topic modeling (TM). El modelado de tópicos aplica un modelo estadístico que busca comprender la estructura del corpus e identificar y agrupar palabras que se relacionan de alguna forma entre sí. El TM utiliza una técnica semi o no supervisada para la identificación de esos tópicos. En otras palabras, el programa aprende a reconocer patrones en los datos sin la necesidad de anotaciones previas. El código a continuación demuestra la aplicación del modelo Latent Dirichlet Allocation (LDA).

```
# Modelización de tópicos a partir del corpus formado por un único documento

lda <- LDA(dfm_selected_unico, k = 10)
terms(lda, 10)
```

```
##       Topic 1  Topic 2     Topic 3  Topic 4     Topic 5   Topic 6   Topic 7
##  [1,] "sancho" "quijote"   "merced" "sancho"    "don"     "don"     "don"
##  [2,] "don"    "señor"     "señor"  "don"       "quijote" "ser"     "quijote"
##  [3,] "ser"    "sancho"    "bien"   "bien"      "bien"    "quijote" "sancho"
##  [4,] "señora" "decir"     "ser"    "caballero" "merced"  "merced"  "señor"
##  [5,] "merced" "mal"       "don"    "quijote"   "aquí"    "señor"   "ser"
##  [6,] "dicho"  "caballero" "hacer"  "merced"    "ser"     "mundo"   "cosa"
##  [7,] "buen"   "dios"      "aquí"   "ser"       "decir"   "cosa"    "caballero"
##  [8,] "señor"  "hacer"     "sancho" "señora"    "señor"   "ahora"   "buen"
##  [9,] "sé"     "don"       "dios"   "decir"     "sancho"  "lugar"   "bien"
## [10,] "lugar"  "dio"       "menos"  "dios"      "mal"     "bien"    "vida"
##       Topic 8   Topic 9     Topic 10
##  [1,] "don"     "quijote"   "quijote"
##  [2,] "quijote" "sancho"    "don"
##  [3,] "señor"   "don"       "bien"
```

```
##  [4,] "decir"   "caballero" "sancho"
##  [5,] "merced"  "bien"      "ser"
##  [6,] "verdad"  "ser"       "señor"
##  [7,] "hacer"   "dios"      "señora"
##  [8,] "ser"     "dicho"     "dicho"
##  [9,] "bien"    "hacer"     "mundo"
## [10,] "mal"     "todas"     "merced"
```

```
# Modelización de tópicos a partir de un corpus formado por varios documentos

lda <- LDA(dfm_selected_files, k = 10)
terms(lda, 10)
```

```
##       Topic 1    Topic 2    Topic 3  Topic 4   Topic 5     Topic 6  Topic 7
##  [1,] "numancia" "amor"     "habia"  "don"     "don"       "ser"    "sancho"
##  [2,] "s"        "bien"     "hacer"  "quijote" "quijote"   "navío"  "quijote"
##  [3,] "°"        "ser"      "d"      "ser"     "señor"     "tierra" "don"
##  [4,] "1"        "pastores" "ojos"   "bien"    "merced"    "mar"    "bien"
##  [5,] "vida"     "elicio"   "ser"    "merced"  "caballero" "todas"  "merced"
##  [6,] "2"        "galatea"  "habian" "manera"  "dios"      "lugar"  "aquí"
##  [7,] "romanos"  "vida"     "ó"      "ver"     "sancho"    "rey"    "decir"
##  [8,] "muerte"   "mal"      "tal"    "mal"     "bien"      "fué"    "vida"
##  [9,] "aquí"     "cielo"    "bien"   "aquí"    "panza"     "luego"  "señora"
## [10,] "cip"      "ojos"     "cosa"   "señora"  "decir"     "cosa"   "toda"
##       Topic 8  Topic 9  Topic 10
##  [1,] "habia"  "fué"    "sancho"
##  [2,] "bien"   "casa"   "ser"
##  [3,] "ser"    "ó"      "don"
##  [4,] "ó"      "todas"  "señor"
##  [5,] "d"      "ser"    "hacer"
##  [6,] "luego"  "habia"  "bien"
##  [7,] "señora" "ahora"  "cosa"
##  [8,] "señor"  "merced" "verdad"
##  [9,] "decir"  "señor"  "caballero"
## [10,] "vida"   "cosa"   "dicho"
```

2.3 Red semántica

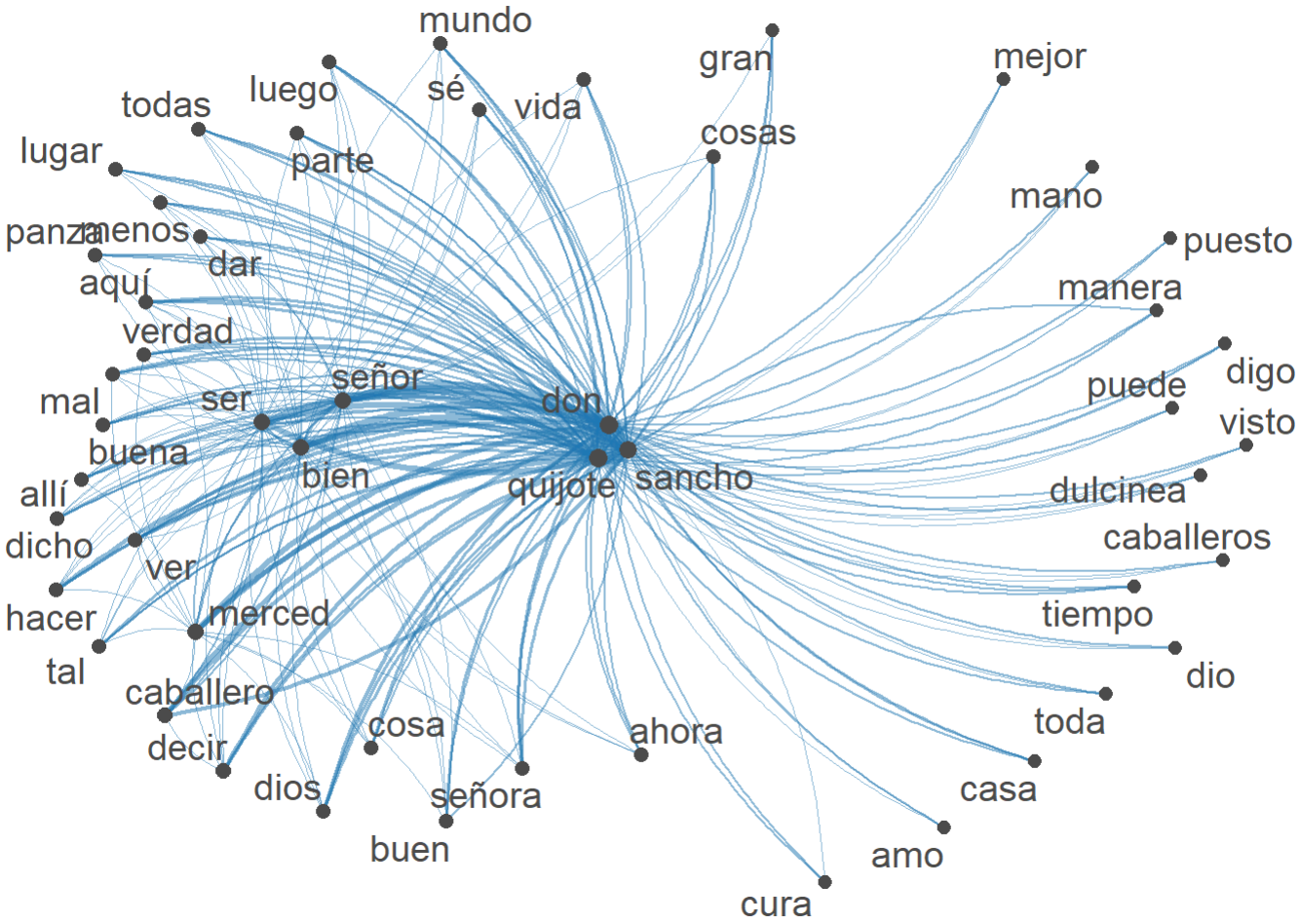
El Feature co-occurrence matrix (FCM) es similar al DFM, pero considera las coocurrencias, presentando un gráfico con las redes semánticas.

```
# Red a partir del corpus formado por un único documento

# Crear FCM a partir de DFM
fcm_nostop <- fcm(dfm_selected_unico)
# Listar las top features
feat <- names(topfeatures(dfm_selected_unico, 50))
# Seleccionar
fcm_select <- fcm_select(fcm_nostop, pattern = feat, selection = "keep")

size <- log(colSums(dfm_select(dfm_selected_unico, feat, selection = "keep")))
```

```
textplot_network(fcm_select, min_freq = 0.8, vertex_size = size / max(size) * 3)
```

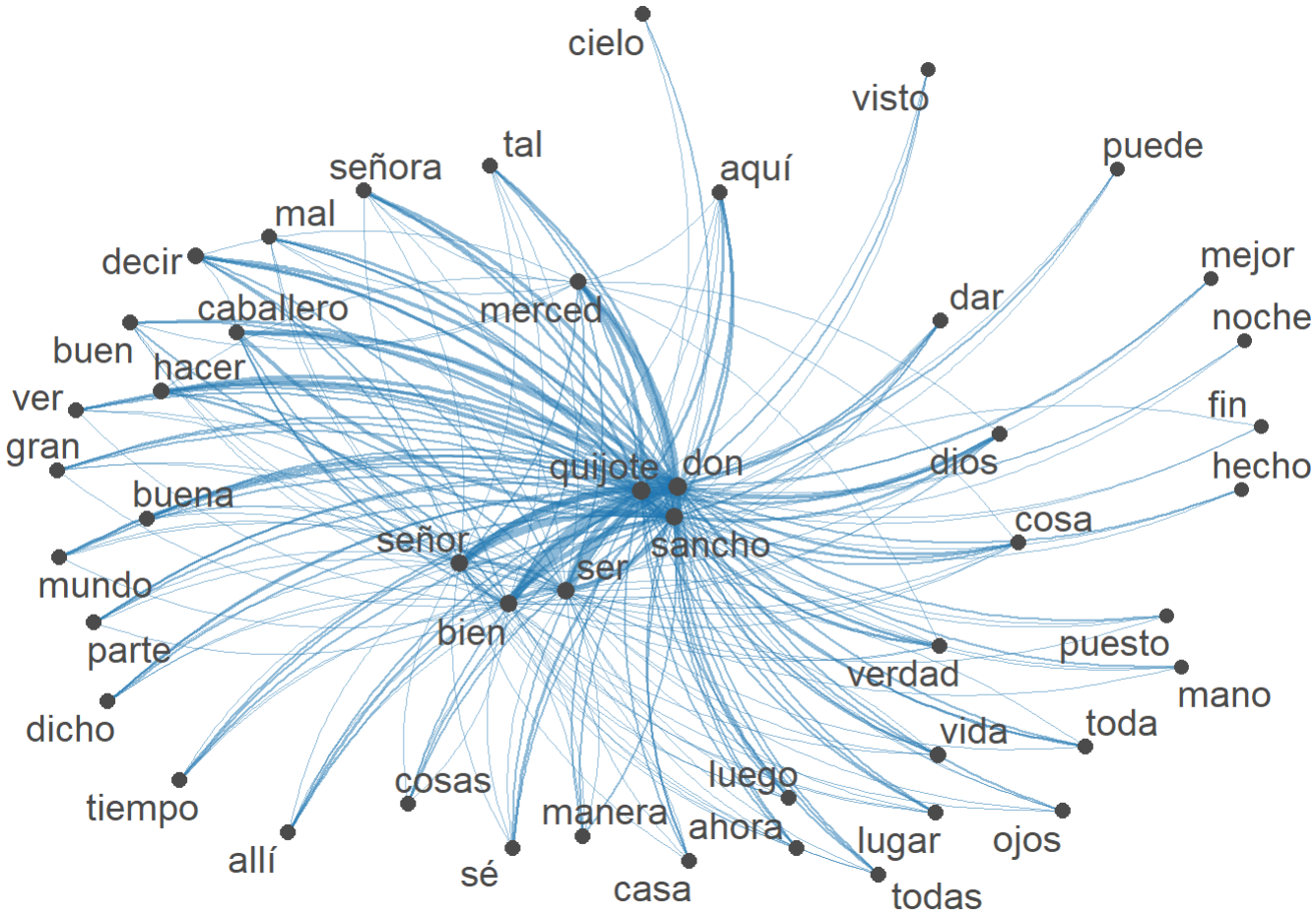



```
# Red a partir de un corpus formado por varios documentos

# Crear FCM a partir de DFM
fcm_nostop <- fcm(dfm_selected_files)
# Listar las top features
feat <- names(topfeatures(dfm_selected_files, 50))
# Seleccionar
fcm_select <- fcm_select(fcm_nostop, pattern = feat, selection = "keep")

size <- log(colSums(dfm_select(dfm_selected_files, feat, selection = "keep")))

textplot_network(fcm_select, min_freq = 0.8, vertex_size = size / max(size) * 3)
```



Datos y repositorio

Los datos y códigos están disponibles vía github https://github.com/DiegoEGimenez/R_literatura_Quanteda

El código puede ser visualizado en https://rpubs.com/DiegoEGimenez/r_literatura_quanteda_es

---

Agradecimientos

Este documento (2024) contiene una revisión y ampliación de códigos originalmente preparados por Diego Giménez y Andressa Gomide en 2022 para el análisis del “Libro del desasosiego”. Algunos de los códigos descritos en el documento de 2022 utilizaron los códigos gentilmente cedidos por Mark Alfano, usados en su trabajo “Nietzsche corpus analysis”.