Mineria de textos con R

Juan Bosco Mendoza Vega

18 de junio de 2016

Table of Contents

[Paquetes necesarios 1](#_Toc508053565)

[Nuestro texto: Niebla 2](#_Toc508053566)

[Preparación del texto 2](#_Toc508053567)

[Creación de "párrafos" 3](#_Toc508053568)

[Limpieza del texto 4](#_Toc508053569)

[Análisis del Corpus 5](#_Toc508053570)

[Nube de palabras 5](#_Toc508053571)

[Más depuración 6](#_Toc508053572)

[Term Document Matrix 7](#_Toc508053573)

[Frecuencia de palabras 8](#_Toc508053574)

[Gráficas de frecuencia 10](#_Toc508053575)

[Asociaciones entre palabras 11](#_Toc508053576)

[Agrupamiento jerárquico (Hierarchical clustering) 13](#_Toc508053577)

[Eliminar términos dispersos 13](#_Toc508053578)

[Matriz de distancia 14](#_Toc508053579)

[hclust 15](#_Toc508053580)

[Agnes (Agglomerative Nesting) 16](#_Toc508053581)

[Conclusión 18](#_Toc508053582)

[Referencias 18](#_Toc508053583)

En este documento se presentará cómo realizar operaciones elementales de mineria de textos usando **R**, las cuales deberían proporcionar suficientes herramientas para análisis más completos y complejos. No pretende ser una guía exahustiva, sólo un documento introductorio de referencia.

## Paquetes necesarios

Empezamos por cargar a nuestro espacio de trabajo los paquetes que usaremos:

* tm, específico para minería de textos.
* wordcloud, para graficar nubes de palabras.
* ggplot2, una gramática de gráficas que expande las funciones base de **R**.
* dplyr, con funciones auxiliares para manipular y transformar datos.
* readr, facilitará leer y escribir documentos.
* cluster, con funciones para realizar análisis de grupos.

library(tm)  
library(SnowballC)  
library(wordcloud)  
library(ggplot2)  
library(dplyr)  
library(readr)  
library(cluster)

## Nuestro texto: Niebla

El texto con el que trabajeremos es el texto del libro **Niebla** de **Miguel de Unamuno** que se puede descargar en Gutenberg dando [click a este enlace](https://www.gutenberg.org/files/49836/49836-0.txt).

Una vez hemos descargado este documento a nuestro directorio de trabajo, procedemos a leerlo usando la función read\_lines de readr.

Nuestro interés está en el contenido de este libro y no en los avisos legales de Gutenberg, prólogo y anotaciones, así que los omitiremos de la lectura. Empezaremos a leer el documento desde la línea 420, que es donde termina el prólogo, introducción e índice de **Niebla**, para ello nos saltaremos (skip) 419 líneas previas. De manera complementaria, detendremos la lectura en la línea 8313, que es donde inician los avisos legales de *Gutenberg*, por lo tanto leeremos un máximo (nmax) de 8313-419 líneas.

Realizamos estas operaciones y asignamos el resultado al objeto **nov\_raw**.

nov\_raw <- read\_lines("49836-0.txt", skip = 419, n\_max = 8313-419)

## Preparación del texto

El objeto **nov\_raw** que obtuvimos es uno de tipo character, con 7894 elementos.

str(nov\_raw)

## chr [1:7894] "NIEBLA" "" "" "" "" "I" "" "" ...

Cada uno de estos elementos corresponde a un renglón de **Niebla** y tiene ancho máximo 70 caracteres, pues este es el ancho usado en los textos electrónicos de **Gutenberg**. Esta es una cantidad muy pequeña de texto para encontrar asociaciones entre palabras, por lo que necesitamos crear elementos con una mayor cantidad de caracteres en cada uno.

Por lo tanto, crearemos elementos del tamaño aproximado a un párrafo.

### Creación de "párrafos"

Creamos un vector llamado **diez** con 10 repeticiones (rep) de los números desde 1 hasta el número de renglones en el documento, dividido entre 10 (length(nov\_raw)/10.

Con esto, tendremos un vector con diez 1, luego diez 2, etc, hasta llegar al número máxico de grupos de diez posibles en función del número de renglones de nuestro documento.

Usaremos estos números para hacer grupos de diez renglones consecutivos.

diez <- rep(1:ceiling(length(nov\_raw)/10), each = 10)

De este vector, nos quedamos con un número de elementos igual al número de renglones del objeto **nov\_raw** (length(nov\_raw)), para facilitar combinarlos.

diez <- diez[1:length(nov\_raw)]

Combinamos **diez** con **now\_raw** y los asignamos al objeto **nov\_text**. Así tenemos una columna con los renglones de texto y otra con un número que identifica a qué grupo de diez renglones pertenece.

Además, convertimos a data.frame para que las columnas estén identificadas con un nombre, lo cual será útil en los siguientes pasos.

nov\_text <- cbind(diez, nov\_raw) %>% data.frame()

Usamos aggregate para concatenar los renglones (FUN = paste, con collapse = " " para preservar el espacio entre palabras), agrupados por diez (formula = nov\_raw ~ diez).

nov\_text <- aggregate(formula = nov\_raw ~ diez,  
 data = nov\_text,  
 FUN = paste,  
 collapse = " ")

Como sólo necesitamos la columna con los ahora párrafos de texto, con eso nos quedamos. Aprovechamos para transformar **nov\_text** en una matrix, pues esto nos facilitará los pasos siguientes.

nov\_text <- nov\_text %>% select(nov\_raw) %>% as.matrix  
  
dim(nov\_text)

## [1] 790 1

Por supuesto, podemos realizar todo lo anterior en un único paso.

nov\_text <-  
 cbind(  
 rep(1:ceiling(length(nov\_raw)/10), each = 10) %>%  
 .[1:length(nov\_raw)],  
 nov\_raw  
 ) %>%  
 data.frame %>%  
 aggregate(  
 nov\_raw ~ V1,  
 data = .,  
 FUN = paste,  
 collapse=" ") %>%  
 select(nov\_raw) %>%  
 as.matrix  
  
dim(nov\_text)

## [1] 790 1

### Limpieza del texto

Necesitamos limpiar el texto de caracteres que son de poca utilidad en la mineria de textos.

Empezamos por aseguramos de que no queden caracteres especiales de la codificación, como saltos de línea y tabulaciones, con un poco de ayuda de *Regular Expressions*.

nov\_text <- gsub("[[:cntrl:]]", " ", nov\_text)

Convertimos todo a minúsculas.

nov\_text <- tolower(nov\_text)

Usamos removeWords con stopwords("spanish") para eliminar *palabras vacias*, es decir, aquellas con poco valor para el análisis, tales como algunas preposiciones y muletillas.

nov\_text <- removeWords(nov\_text, words = stopwords("spanish"))

Nos deshacemos de la puntuación, puesto que *fin* y *fin.* son identificadas como palabras diferentes, lo cual no deseamos.

nov\_text <- removePunctuation(nov\_text)

En este caso, removemos los números, pues en **Niebla** no hay fechas y otras cantidades que deseemos conservar.

nov\_text <- removeNumbers(nov\_text)

Por último eliminamos los espacios vacios excesivos, muchos de ellos introducidos por las transformaciones anteriores.

nov\_text <- stripWhitespace(nov\_text)

## Análisis del Corpus

Con nuestro documento preparado, procedemos a crear nuestro Corpus, es decir, esto es nuestro acervo de documentos a analizar.

En nuestro caso, nuestro Corpus se compone de todos los parrafos del libro **Niebla** y los asignaremos al objeto **nov\_corpus** usando las funciones VectorSource y Corpus.

nov\_corpus <- Corpus(VectorSource(nov\_text))  
  
nov\_corpus

## <<VCorpus>>  
## Metadata: corpus specific: 0, document level (indexed): 0  
## Content: documents: 790

Como podemos ver, nustro Copus está compuesto por 790 documentos. Los siguientes análisis se harán a partir de este Corpus.

### Nube de palabras

Mapearemos nuestro Corpus como un documento de texto plano usando las funciones tm\_map y PlainTextDocument).

nov\_ptd <- tm\_map(nov\_corpus, PlainTextDocument)

Con nuestro Corpus mapeado de esta manera, podemos crear fácilmente una nube de palabras (wordcloud de la librería del mismo nombre) que nos muestro los términos más frecuentes en **Niebla**.

wordcloud(nov\_ptd, max.words = 80, random.order = F, colors = brewer.pal(name = "Dark2", n = 8))



Podemos observar que en nuestro Corpus aun se encuentran palabras de poco interés para el análisis, tales como "usted", "tal" y "sino", lo cual nos indica que debemos realizar una segunda limpieza de nuestros textos.

### Más depuración

Como observamos en las nubes de palabras que generamos, aún tenemos palabras que aparecen con mucha frecuencia en nuestro texto que en realidad no son de mucha utilidad para el análisis.

Usaremos la función removeWords indicando en el argumento words que palabras deseamos eliminar de nuestro Corpus.

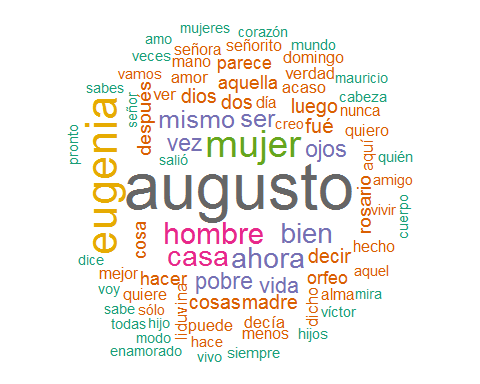
Esta función requiere de un vector de caracteres como argumento principal, así que para esta operación usaremos nuestro objeto *nov\_text*.

Una vez hecho esto, usaremos nuestra nueva versión de *nov\_text* para generar nuestro Corpus y para mapearlo como documento de texto plano.

nov\_text <- removeWords(nov\_text, words = c("usted", "pues", "tal", "tan", "así", "dijo", "cómo", "sino", "entonces", "aunque", "don", "doña"))  
  
nov\_corpus <- nov\_text %>% VectorSource() %>% Corpus()  
nov\_ptd <- nov\_corpus %>% tm\_map(PlainTextDocument)

Generamos un nube de palabras nueva, en la cual es posible ver una diferencia significativa en su composición.

wordcloud(  
 nov\_ptd, max.words = 80,   
 random.order = F,   
 colors=brewer.pal(name = "Dark2", n = 8)  
 )



***Augusto*** y ***Eugenia***, como podrás imaginar, son los protagonistas de **Niebla** y gran parte de la acción en este libro ocurre en la "casa" de uno u otro protagonista, discutiendo las relaciones entre "hombre" y "mujer".

Esta nube nos da una idea más precisa que la anterior del contenido de **Niebla** y aunque nuestro Corpus puede ser depurado aún más, lo dejaremos así.

### Term Document Matrix

Mapearemos nuestro Corpus indicando que es una matriz de términos, de esta manera podremos hacer realizar operaciones como identificar asociaciones entre palabras.

Usaremos la función TermDocumentMatrix en nuestro Corpus y asignaremos el resultado al objeto **nov\_tdm**.

nov\_tdm <- TermDocumentMatrix(nov\_corpus)  
nov\_tdm

## <<TermDocumentMatrix (terms: 7236, documents: 790)>>  
## Non-/sparse entries: 20527/5695913  
## Sparsity : 100%  
## Maximal term length: 22  
## Weighting : term frequency (tf)

Podemos observar que tenemos 7236 *terms*, esto quiere decir que tenemos 7236 palabras diferentes en nuestro Corpus. Lo cual es una cantidad considerable de vocabulario pero no esperaríamos otra cosa de una obra de *Miguel de Unamuno*.

### Frecuencia de palabras

Aunque una nube de palabras nos muestra de manera visual la frecuencia de las palabras en nuestro Corpus, no nos devuelve cantidades.

Para obtenerlas, primero transformaremos nuestro objeto **nov\_tdm** en un objeto de clase *matrix*, que de nuevo tendrá un número de renglones igual al número de palabras distintas de nuestro Corpus y número de columnas igual a su número de documentos.

nov\_mat <- as.matrix(nov\_tdm)  
dim(nov\_mat)

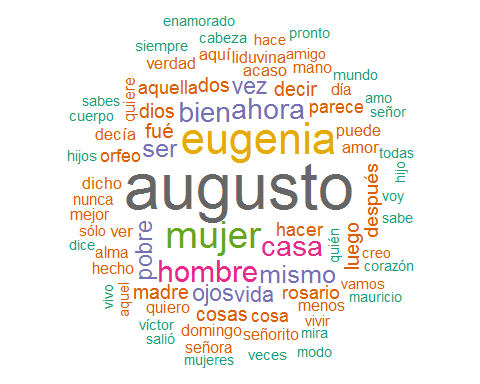
## [1] 7236 790

Obtenemos las sumas de renglones (rowSums) odenadas de mayor a menor (sort con decreasing = TRUE)para conocer la frecuencia de cada palabra y después transformamos los resultados a objeto de clase data.frame de dos columnas, *palabra* y *frec*, que nos permitirá graficar fácilmente su contenido.

nov\_mat <- nov\_mat %>% rowSums() %>% sort(decreasing = TRUE)  
nov\_mat <- data.frame(palabra = names(nov\_mat), frec = nov\_mat)

Con este objeto también podemos crear una nube de palabras, aunque con argumentos un poco diferentes.

wordcloud(  
 words = nov\_mat$palabra,   
 freq = nov\_mat$frec,   
 max.words = 80,   
 random.order = F,   
 colors=brewer.pal(name = "Dark2", n = 8)  
 )



Además, podemos obtener fácilmente las palabras más frecuentes. Por ejemplo, las veinte más frecuentes, pidiendo los primeros veinte renglones de **nov\_mat**.

nov\_mat[1:20, ]

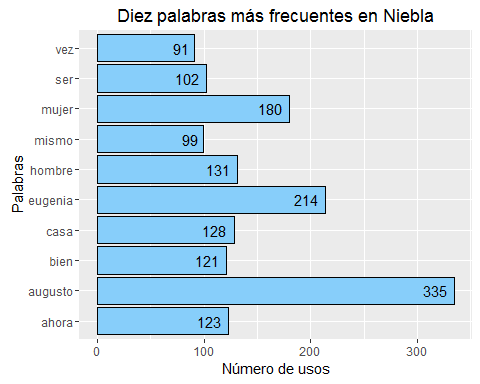
## palabra frec  
## augusto augusto 335  
## eugenia eugenia 214  
## mujer mujer 180  
## hombre hombre 131  
## casa casa 128  
## ahora ahora 123  
## bien bien 121  
## ser ser 102  
## mismo mismo 99  
## vez vez 91  
## pobre pobre 90  
## ojos ojos 87  
## vida vida 84  
## fué fué 79  
## dos dos 74  
## luego luego 74  
## dios dios 71  
## cosas cosas 70  
## decir decir 70  
## madre madre 70

### Gráficas de frecuencia

Crearemos un par de gráficas. Primero, la frecuencia de uso de las palabras más frecuentes en **Niebla**.

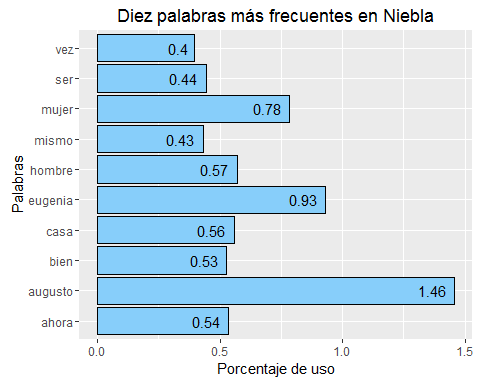
Para esto usaremos ggplot2, que tiene su propia gramática para construir gráficas. Para fines de este documento, no nos detendremos a explicar a detalle su uso. Lo relevante aquí es notar que estamos obteniendo la información para construir las gráficas solicitando renglones del objeto **nov\_mat**

nov\_mat[1:10, ] %>%  
 ggplot(aes(palabra, frec)) +  
 geom\_bar(stat = "identity", color = "black", fill = "#87CEFA") +  
 geom\_text(aes(hjust = 1.3, label = frec)) +   
 coord\_flip() +   
 labs(title = "Diez palabras más frecuentes en Niebla", x = "Palabras", y = "Número de usos")



La misma información expresada como proporción de uso cada palabra. Para esta gráfica usamos la función mutate de dplyr para obtener el porcentaje de uso de cada palabra antes de graficar.

nov\_mat %>%  
 mutate(perc = (frec/sum(frec))\*100) %>%  
 .[1:10, ] %>%  
 ggplot(aes(palabra, perc)) +  
 geom\_bar(stat = "identity", color = "black", fill = "#87CEFA") +  
 geom\_text(aes(hjust = 1.3, label = round(perc, 2))) +   
 coord\_flip() +  
 labs(title = "Diez palabras más frecuentes en Niebla", x = "Palabras", y = "Porcentaje de uso")



A partir de las dos gráficas anteriores podemos observar, entre otras cosas, que aunque "augusto" es la palabra más usada, representa menos de 1.5% del total de usos de las palabras en nuestro Corpus.

### Asociaciones entre palabras

Veamos ahora cómo se asocian algunas palabras (terms) en **Niebla** con la función findAssocs. Como podemos introducir un vector, podemos obtener las asociaciones de varias palabras a la vez. He elegido "augusto", "eugenia", "hombre" y "mujer".

Es importante recordar que con esto no estamos pidiendo la asociacion de estas cuatro palabras entre si, sino las asociaciones para cada una de las cuatro, que no necesariamente deben coincidir.

Esta también nos pide el límite inferior de correlación (corlimit) para mostrarnos. Valores cercanos a 1 indican que las palabras aparecen casi siempre asociadas una con otra, valores cercanos a 0 nos indican que nunca o casi nunca lo hacen.

El valor que decidamos depende del tipo de documento y el tipo de asociaciones que nos interesen. para nuestros fines, lo he fijado en .25.

findAssocs(nov\_tdm, terms = c("augusto", "eugenia", "hombre", "mujer"), corlimit = .25)

## $augusto  
## numeric(0)  
##   
## $eugenia  
## abajo fugitiva hominis lucharemosiba militia   
## 0.33 0.33 0.33 0.33 0.33   
## super terram venceré victoria vita   
## 0.33 0.33 0.33 0.33 0.33   
## yunta finalidad lucharemos américa buscarme   
## 0.33 0.31 0.31 0.27 0.27   
## colón compadecida descubre encontrar pronunciando   
## 0.27 0.27 0.27 0.27 0.27   
## est   
## 0.26   
##   
## $hombre  
## salida cínicos confundían engendrarlo mortal perrunos   
## 0.29 0.28 0.28 0.28 0.28 0.28   
## respire resucitarlo trajes hipocresía cinismo   
## 0.28 0.28 0.28 0.27 0.26   
##   
## $mujer  
## colectividad concretó dejarla enamoras   
## 0.32 0.32 0.32 0.32   
## enamoraste explicártelo valiesen abstracto   
## 0.32 0.32 0.32 0.30   
## fogueteiro hermosura busto convertida   
## 0.28 0.27 0.26 0.26   
## desfigurada desfiguramiento desvanecidos explosión   
## 0.26 0.26 0.26 0.26   
## gravísimas inspirarle lazarilla musa   
## 0.26 0.26 0.26 0.26   
## orgulloso pirotécnicas pólvora ponderándola   
## 0.26 0.26 0.26 0.26   
## prende quemaduras quemó genérico   
## 0.26 0.26 0.26 0.25

Aunque "augusto" es la palabra más frecuente en **Niebla** no tiene relaciones tan fuertes como las demás palabras que elegimos. De hecho, no tiene ninguna superior a .25, por eso el resultado mostrado.

A partir de estos resultados podemos observar algunas cosas, por ejemplo, que "hombre" está asociada con adjetivos negativos, que "mujer" se asocia con palabras que hacen alusión a fuego y explosiones, y que "lucharemosiba" aparece como una palabra, que nos indica que sería buena idea realizar aún más limpieza en nuestro corpus.

Esto es de esperar, pues difícilmente una primera ronda de limpieza capturará todas las anomalías en un conjunto de documentos. Nosotros continuaremos con las agrupaciones.

## Agrupamiento jerárquico (Hierarchical clustering)

Realizaremos análisis de agrupaciones jerárquicas para identificar grupos de palabras relacionados entre sí, a partir de la distancia que existe entre ellos.

Empezaremos por eliminar los términos dispersos en nuestra matriz de términos, para así conservar únicamente las palabras más frecuentes y obtener resultados más interpretables del agrupamiento.

### Eliminar términos dispersos

Usaremos las función removeSparseItems para depurar nuestra matriz de términos de aquellas palabras que aparecen con muy poca frecuencia, es decir, son dispersos ("sparse").

Esta función requiere que especifiquemos el argumento *sparse*, que puede asumir valores de 0 a 1. Entre valor representa la dispersión de las palabras que queremos conservar. Si lo fijamos muy alto (cerca de 1, pero no 1), conservaremos muchas palabras, casi todas, pues estamos indicando que queremos conservar términos aunque sean muy dispersos. Naturalmente, lo opuesto ocurre si fijamos este valor muy bajo (cerca de 0, pero no 0), pudiendo incluso quedarnos con ningún término, si las palabras en nuestros documentos son dispersas en general.

Qué valor fijemos depende del tipo de documento que tengamos, por lo que es aconsejable realizar ensayos hasta encontrar un equilibrio entre dispersión y número de términos. En este caso, he decidido fijarlo en .95 y guardaremos la nueva matriz de términos en el objeto *nov\_new*

nov\_new <- removeSparseTerms(nov\_tdm, sparse = .95)

Comparamos cuántos términos teníamos originalmente y con cuántos nos hemos quedado, observando a cuánto equivale *terms*.

nov\_tdm

## <<TermDocumentMatrix (terms: 7236, documents: 790)>>  
## Non-/sparse entries: 20527/5695913  
## Sparsity : 100%  
## Maximal term length: 22  
## Weighting : term frequency (tf)

nov\_new

## <<TermDocumentMatrix (terms: 42, documents: 790)>>  
## Non-/sparse entries: 2861/30319  
## Sparsity : 91%  
## Maximal term length: 8  
## Weighting : term frequency (tf)

De 7236 términos que teníamos, nos hemos quedado con 42, lo cual reduce en gran medida la dificultad y complejidad de los agrupamientos, lo cual es deseable. Es poco útil tener agrupaciones que son únicamente visualizaciones del texto original.

También podemos ver el número de términos pidiéndo el número de renglones de nuestra matriz de términos, que es igual al número de palabras que contiene.

nov\_tdm$nrow

## [1] 7236

nov\_new$nrow

## [1] 42

Transformamos esta matriz de términos a un objeto de tipo matrix para así poder realizar las operaciones posteriores.

nov\_new <- nov\_new %>% as.matrix()

En este objeto de clase matrix cada renglon es una palabra y cada columna es un documento, de modo tal que cada celda nos indica el número de veces que aparece una palabra en un documento.

### Matriz de distancia

Necesitamos crear una matriz de distancias para empezar agrupar, lo cual requiere que los valores en las celdas sean estandarizados de alguna manera.

Podríamos usar la función scale, pero realiza la estandarización usando la media de cada columna como referencia, mientras que nosotros necesitamos como referencia la media de cada renglón.

Así que obtenemos una estandarización por renglones de manera manual.

nov\_new <- nov\_new / rowSums(nov\_new)

Hecho esto, nuestra matriz ha sido estandarizada.

Procedemos a obtener una matriz de distancia a partir de ella, con el método de distancias euclidianas y la asignamos al objeto **nov\_dist**.

nov\_dist <- dist(nov\_new, method = "euclidian")

### hclust

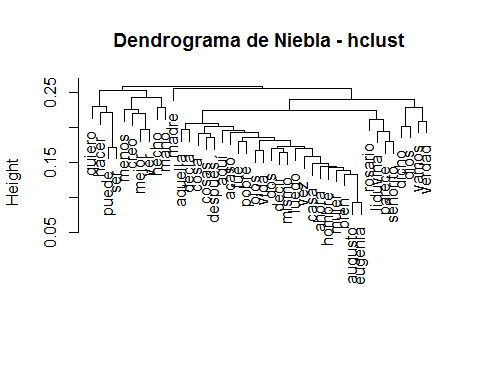
Realizaremos nuestro agrupamiento jerárquico usando la función hclust, de la base de **R**. Este es en realidad un procedimiento muy sencillo una vez que hemos realizado la preparación.

Usaremos el método de Ward (ward.D), que es el método por defecto de la función hclust y asignaremos sus resultados al objeto **nov\_hclust**.

nov\_hclust <- hclust(nov\_dist, method = "ward.D")

Graficamos los resultados usando plot para generar un dendrograma.

plot(nov\_hclust, main = "Dendrograma de Niebla - hclust", sub = "", xlab = "")



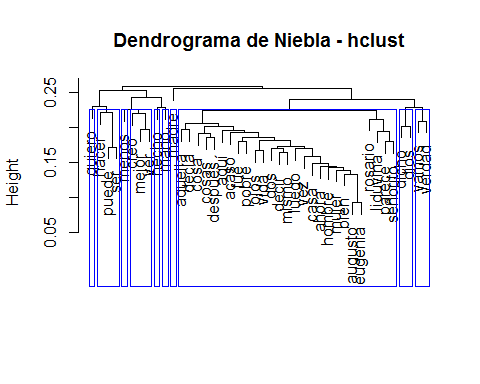
De este modo podemos observar los grupos de palabras que existen en **Niebla**. Por ejemplo, "augusto" y "eugenia" forman un grupo, "puede" y "ser", forman otro grupo ("puede ser" es una frase común en este libro).

Además, podemos ver qué palabras pertenecen a grupos lejanos entre sí, por ejemplo, "quiero" y "verdad".

Podemos enfatizar los grupos de palabras trazando un rectángulo usando rect.hclust y con especificando cuántos grupos (k) deseamos resaltar.

Crearemos el mismo gráfico pidiendo diez grupos.

plot(nov\_hclust, main = "Dendrograma de Niebla - hclust", sub = "", xlab = "")  
rect.hclust(nov\_hclust, k = 10, border="blue")



### Agnes (Agglomerative Nesting)

El paquete cluster nos proporciona más métodos para realizar agrupamientos. Uno de ellos es agnes, que inicia asumiendo que cada elemento a agrupar por si mismo es un grupo y después crea grupos de grupos a partir de las distancias entre ellos, hasta que no es posible crear más grupos.

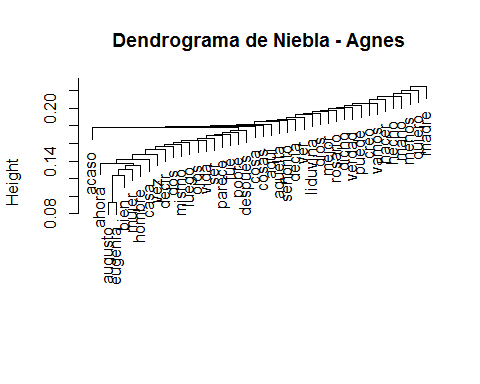
Realizamos prácticamente el mismo procedimiento que con hclust, sólo cambiamos el método a *average*. Asignaremos nuestros resultados al objeto **nov\_agnes**.

nov\_agnes <- agnes(nov\_dist, method = "average")

Ahora graficamos nuestros resultados. Un agrupamiento creado con agnes genera dos gráficos, el primero muestra cómo se obtuvieron los grupos finales y el segundo es un dendrograma.

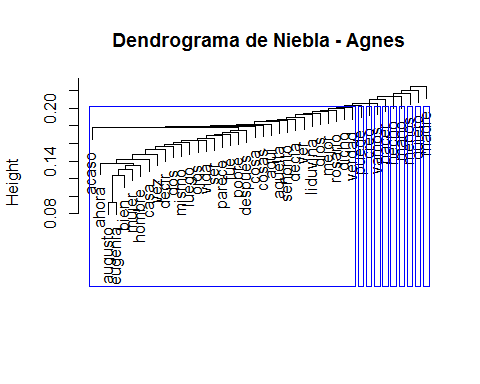
Pediremos el segundo gráfico (which.plots = 2).

plot(nov\_agnes, which.plots = 2, main = "Dendrograma de Niebla - Agnes", sub = "", xlab = "")



Enfatizamos diez grupos.

plot(nov\_agnes, which.plots = 2, main = "Dendrograma de Niebla - Agnes", sub = "", xlab = "")  
rect.hclust(nov\_agnes, k = 10, border = "blue")



Las agrupaciones que hemos obtenido usando hclust y agnes son diferentes entre sí. La decisión de qué método usemos depende de nuestros propósitos y de nuestra familiaridad con ellos.

## Conclusión

En este documento revisamos cómo importar y preparar documentos de texto para realizar distintas operaciones de mineria de textos con ellos, tales como obtener palabras frecuentes, asocioaciones de palabras y análisis de agrupamiento.

Por supuesto la minería de textos es un área de análisis extensa. Las siguientes referencias han sido de gran ayuda para realizar este documento y amplían lo aquí presentado.

## Referencias

* [Building Wordclouds in R](http://www.r-bloggers.com/building-wordclouds-in-r/)
* [You Must Allow Me To Tell You How Ardently I Admire and Love Natural Language Processing](http://juliasilge.com/blog/You-Must-Allow-Me/)
* [Basic Text Mining in R](https://rstudio-pubs-static.s3.amazonaws.com/31867_8236987cf0a8444e962ccd2aec46d9c3.html)
* [Clustering the Words of William Shakespeare](http://www.exegetic.biz/blog/2013/09/clustering-the-words-of-william-shakespeare/)
* [Handling and Processing Strings in R](http://gastonsanchez.com/Handling_and_Processing_Strings_in_R.pdf)