Introducción al scraping y a la minería de texto

Olivier Nuñez

2018-06-01

Contents

Preámbulo

Internet es una fuente inagotable de información y datos. Desafortunadamente, la mayoría de las veces, los datos están integrados a una pagina web y no pueden ser extraídos directamente para su análisis estadístico.

Cuando el volumen de datos es pequeño, es posible copiarlos "a mano". Pero si la cantidad de datos es demasiado grande o si están dispersos en un gran número de páginas, este método no es factible.

Esta introducción tiene como objetivo explicar cómo extraer esta información utilizando herramientas de R y mostrar algunas operaciones de procesamiento y análisis de datos textuales.

Lo que sigue requiere los paquetes `rvest` y `tidytext`:

```
install.packages(c("rvest","tidytext"),dep=TRUE)
```

1 Scraping con R

1.1 Qué se puede "rascar" de la web?

> "*Si puedes verlo, puedes rascarlo*"

Cualquier cosa en una página web:

- Tablas
- Texto
- Vínculos

- Metadatos (tiempo de publicación, actualización, . . .)
- Atributos de página web (colores, fuentes, tamaño de texto utilizado, ..)
- Imágenes

1.2 Formato HTML

El código HTML está compuesto de etiquetas predefinidas que le dicen al navegador cómo mostrar los datos. A menudo, estas etiquetas están anidadas:

```
<ul>
    <li>Articulo 1</li>
    <li>Articulo 2</li>
</ul>
```

Dará:

- Artículo 1
- Artículo 2

La etiqueta "<ul>" indica una lista desordenada y "<li>" indica un elemento de la lista. Una etiqueta que comienza con una barra inclinada finaliza la etiqueta anterior de ese tipo.

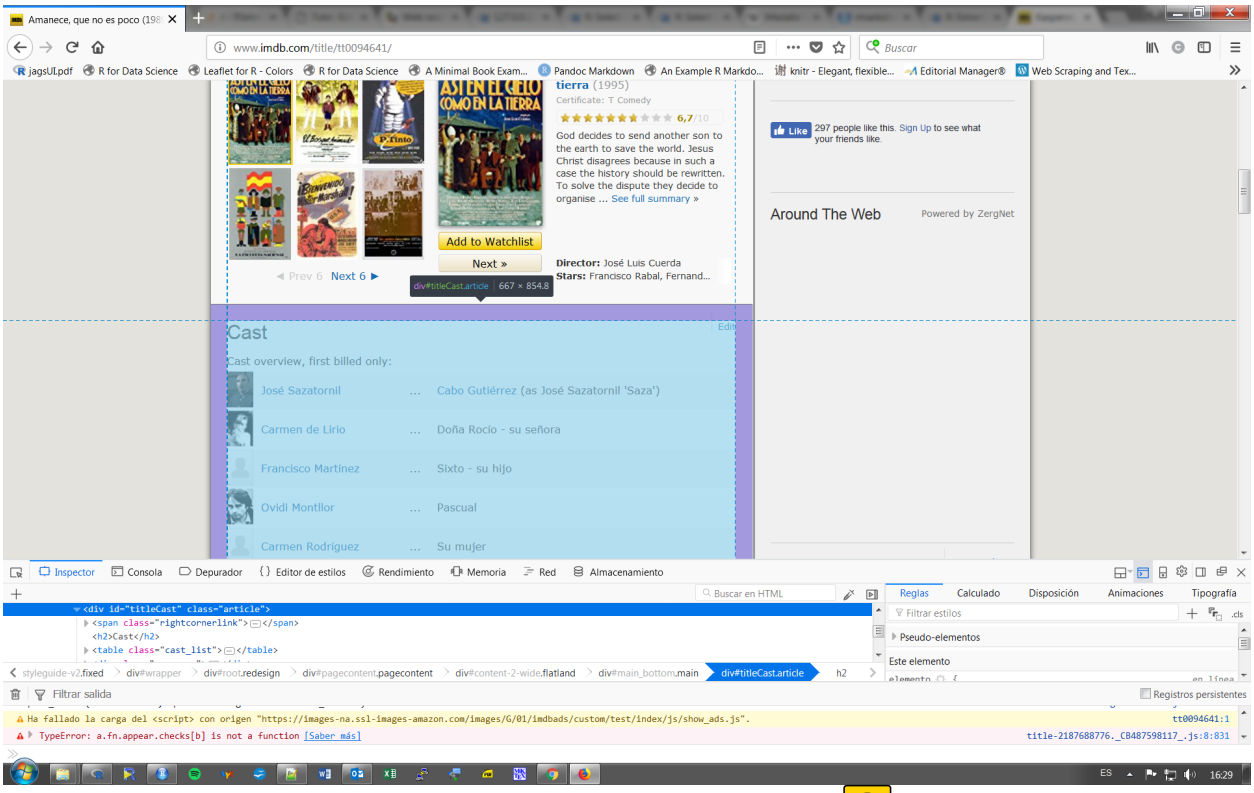
HTML se vuelve más complicado, pero entender las etiquetas es suficiente para comenzar a rascar.

1.3 Identificar elementos de interés en una página

Puede ser útil ilustrar brevemente algunas prácticas para identificar elementos en una página de la cual uno desea extraer los datos. En este sentido, la mayoría de los navegadores web ofrecen no solo la posibilidad de ver el código fuente de una página, sino también analizar los elementos.

Esta función de *inspección* suele estar disponible directamente en el menú contextual del navegador. Se señala el elemento que nos interesa, se hace clic con el botón derecho y se elige el elemento del menú contextual que permite analizarlo.

Captura de pantalla de la pagina de IMDB dedicada a la película "Amanece que no es poco"



Aquí hay una lista de criterios para la identificación de los elementos:

- **ID** de atributo : si el elemento tiene un atributo id (caracterizado en el identificador con el carácter #), se supone que este atributo es exclusivo del elemento. Por lo tanto, es muy útil para extraerlo de la página, pero no es adecuado para extraer elementos recursivos (por ejemplo, varios párrafos), ya que cada uno debe tener una identificación única.
- **Clases** : Las clases son una buena forma de identificar elementos recursivos en el mismo nivel jerárquico. Por ejemplo, los comentarios de blog a menudo comparten la misma clase (por ejemplo, class = "comment"). Para maximizar la probabilidad de identificar el elemento correcto, en el caso de que múltiples elementos en diferentes ubicaciones en la página tengan la misma clase, este criterio puede estar asociado con el nombre de la etiqueta (por ejemplo, div.comment).
- **Posición jerárquica** : el análisis de elementos a menudo propone la posición relativa a la raíz del DOM ocupado por el elemento seleccionado. Se usa si las etiquetas nunca tienen atributos de identificación o clase.

1.4 El paquete `rvest`

Este paquete de "cosecha" extrae contenidos de páginas web en HTML/XML usando la sintaxis de los selectores de CSS. Por ejemplo, un párrafo que está directamente dentro de una etiqueta de tipo *div* puede identificarse usando la notación $div > p$.

Este paquete es bastante simple, porque no tiene muchas funciones, pero proporciona las principales características necesarias para la identificación y extracción de datos en una página, así como algunas funciones que permiten explorar las páginas emulando un navegador web.

1.4.1 Cargar la pagina

Para ilustrar el uso de `rvest`, vamos a extraer información sobre la película "Amanece, que no es poco" de su pagina en el IMDb.

```
require(rvest)
amanece <- read_html("http://www.imdb.com/title/tt0094641/")
```

La función `read_html`permite importar en R el contenido html de una página web. El argumento principal de esta función es la dirección de la pagina web (o el path de fichero html local).

1.4.2 Extracción

La función `html_nodes` acepta dos argumentos, ambos necesarios:

`html_nodes(x, css)`

- El argumento x representa el código HTML importado mediante la función `read_html`
- El segundo argumento es un criterio de selección que utiliza la gramática de los selectores de CSS.

Esta función devuelve una lista (matriz) de las ocurrencias encontradas en la pagina de acuerdo al criterio de selección. Así, el comando siguiente da la lista de las tablas incluidas en la pagina:

```
html_nodes(amanece, "table")
```

```
## {xml_nodeset (2)}
## [1] <table class="cast_list">\n<tr><td colspan="4" class="castlist_label ...
## [2] <table class="footer" id="amazon-affiliates">\n<tr>\n<td colspan="8" ...
```

1.4.2.1 Extracción de texto

Se puede también extraer su titulo

```
titulo <- html_node(amanece, "title")
html_text(titulo)
```

```
## [1] "Amanece, que no es poco (1989) - IMDb"
```

Sólo hay una etiqueta "title" en una página, por lo que utilizaremos `html_node()` (sin la *s* final) en lugar de `html_nodes()`, porque sólo devuelve un elemento en vez de una lista.

La función `html_text` elimina todas las etiquetas del código y muestra solo el contenido textual. Una opción muy útil de esta función es `trim = TRUE` que elimina los espacios antes y después del texto.

Cabe mencionar, que mediante la gramática de "tuberias", la secuencia anterior de comandos, puede ser expresada en una solo linea:

```
amanece %>% html_node("title") %>% html_text()
```

Descargar el ultimo discurso del rey de España desde la siguiente dirección:

http://www.casareal.es/ES/Actividades/Paginas/actividades_discursos_detalle.aspx?data=5738

1.4.2.2 Extracción de tablas

La función `html_table`, como su nombre indica, está especialmente diseñada para extraer los contenidos de una tabla HTML, manteniendo la estructura en filas y columnas-

Así, el comando siguiente permite extraer la lista de actores de la película que viene en la segunda columna de la tabla con clase "cast_list":

```
html_node(amanece, "table.cast_list") %>% html_table(header=TRUE) %>% .[[2]] # .[[2]] para segunda colu
```

```
##  [1] "José Sazatornil"     "Carmen de Lirio"     "Francisco Martínez"
##  [4] "Ovidi Montllor"      "Carmen Rodríguez"    "Rafael Díaz"
##  [7] "Amada Tercero"       "Cassen"              "Manuel Alexandre"
## [10] "María Ángeles Ariza" "Rafael Alonso"       "Fedra Lorente"
## [13] "Cris Huerta"         "Elisa Belmonte"      "María I. González"
```

Descargar las cotizaciones del IBEX 35 en tiempo real desde la siguiente pagina

ibex35 <- "http://www.bolsamadrid.es/esp/aspx/Mercados/Precios.aspx?indice=ESI100000000"

1.4.2.3 Extracción de vinculos

Para ilustrar este tipo de extracción, vamos a importar las direcciones de las paginas de los actores de la película. Empezamos importando todos los enlaces (etiqueta '<a>´de "ancla") contenidos en una tabla:

```
enlaces <- html_nodes(amanece, "table a")
enlaces
```

```
## {xml_nodeset (41)}
##  [1] <a href="/name/nm0768574/?ref_=tt_cl_i1"><img height="44" width="32 ...
##  [2] <a href="/name/nm0768574/?ref_=tt_cl_t1" itemprop="url"> <span clas ...
##  [3] <a href="/title/tt0094641/characters/nm0768574?ref_=tt_cl_t1">Cabo  ...
##  [4] <a href="/name/nm0513922/?ref_=tt_cl_i2"><img height="44" width="32 ...
##  [5] <a href="/name/nm0513922/?ref_=tt_cl_t2" itemprop="url"> <span clas ...
##  [6] <a href="/name/nm1771790/?ref_=tt_cl_i3"><img height="44" width="32 ...
##  [7] <a href="/name/nm1771790/?ref_=tt_cl_t3" itemprop="url"> <span clas ...
##  [8] <a href="/name/nm0600120/?ref_=tt_cl_i4"><img height="44" width="32 ...
##  [9] <a href="/name/nm0600120/?ref_=tt_cl_t4" itemprop="url"> <span clas ...
## [10] <a href="/name/nm1771909/?ref_=tt_cl_i5"><img height="44" width="32 ...
## [11] <a href="/name/nm1771909/?ref_=tt_cl_t5" itemprop="url"> <span clas ...
## [12] <a href="/name/nm0246717/?ref_=tt_cl_i6"><img height="44" width="32 ...
## [13] <a href="/name/nm0246717/?ref_=tt_cl_t6" itemprop="url"> <span clas ...
## [14] <a href="/name/nm1773519/?ref_=tt_cl_i7"><img height="44" width="32 ...
## [15] <a href="/name/nm1773519/?ref_=tt_cl_t7" itemprop="url"> <span clas ...
## [16] <a href="/name/nm0144107/?ref_=tt_cl_i8"><img height="44" width="32 ...
## [17] <a href="/name/nm0144107/?ref_=tt_cl_t8" itemprop="url"> <span clas ...
## [18] <a href="/title/tt0094641/characters/nm0144107?ref_=tt_cl_t8">Cura  ...
## [19] <a href="/name/nm0018872/?ref_=tt_cl_i9"><img height="44" width="32 ...
## [20] <a href="/name/nm0018872/?ref_=tt_cl_t9" itemprop="url"> <span clas ...
## ...
```

Luego extraemos las direcciones (atributo `href`) a las que apuntan dichos enlaces utilizando la función `html_attr`:

```
actores <- enlaces %>% html_attr("href")
raiz="http://www.imdb.com"
browseURL(paste(raiz,actores[1],sep="/")) #pagina de José Sazatornil
```

1.5 Navegación

El paquete `rvest` proporciona también funciones para emular un navegador web.

1.5.1 Inición de la navegación

Con la función `html_session`se crea un punto de entrada para la navegación.

```
sesion <- html_session("http://www.imdb.com")
```

La variable "sesion" contiene ahora información de la página "visitada".

```
sesion
```

```
## <session> https://www.imdb.com/
##   Status: 200
##   Type:   text/html;charset=UTF-8
##   Size:   192658
```

1.5.2 Funciones `jump_to` y `follow_link`

Estas dos funciones tienen el mismo propósito, es decir, seguir un enlace para ir de una página a otra. La diferencia entre las dos funciones se refiere a la forma de referirse al enlace.

La función `jump_to` toma una url (ya sea relativa o absoluta)

```
sesion %>%  jump_to("boxoffice") %>% session_history()
```

```
##   https://www.imdb.com/
## - https://www.imdb.com/chart/boxoffice
```

Mientras que la función `follow_link` acepta la referencia a un enlace en una página en tres formas distintas:

- Un número cardinal (ej., "5" para la quinta etiqueta `a` presente en el HTML de la página).
- Un selector CSS (ej., "p a" para enlaces en un párrafo)
- Una palabra o frase que representa la etiqueta textual del enlace (ej.)

toma una expresión que hace referencia a un enlace (una etiqueta $<a>$) de la página:

```
sesion %>% follow_link(5)
```

```
## Navigating to /chart/toptv/?ref_=nv_tp_tv250_2
```

```
## <session> https://www.imdb.com/chart/toptv/?ref_=nv_tp_tv250_2
##   Status: 200
##   Type:   text/html;charset=UTF-8
##   Size:   572853
```

```
sesion %>% follow_link(css = "p a") #enlace en un parráfo
```

```
## Navigating to /movies-in-theaters/?ref_=nv_tp_inth_1
```

```
## <session> https://www.imdb.com/movies-in-theaters/?ref_=nv_tp_inth_1
##   Status: 200
##   Type:   text/html;charset=UTF-8
##   Size:   198241
```

```
sesion %>% follow_link("Indian")
```

```
## Navigating to /india/top-rated-indian-movies?ref_=nv_mv_250_in_7
```

```
## <session> https://www.imdb.com/india/top-rated-indian-movies/?ref_=nv_mv_250_in_7
##   Status: 200
##   Type:   text/html;charset=UTF-8
##   Size:   592337
```

Cualquier movimiento con estas funciones se grabará en la sesión de navegación, de modo que se pueda realizar un seguimiento de la navegación. La función `session_history` muestra la cronología completa de las páginas visitadas en la misma sesión:

```
sesion %>%  jump_to("boxoffice") %>% session_history()
```

```
##   https://www.imdb.com/
## - https://www.imdb.com/chart/boxoffice
```

1.5.3 html_form(sesion)

```
html_form(sesion)
```

```
## [[1]]
## <form> 'navbar-form' (GET /find)
##   <button submit> '<unnamed>
##   <input hidden> 'ref_': nv_sr_fn
##   <input text> 'q':
##   <select> 's' [0/6]
##
## [[2]]
## <form> 'ue_backdetect' (GET get)
##   <input hidden> 'ue_back': 1
```

Buscamos las películas con "amanece" en su título:

```
busqueda <-html_form(sesion)[[1]] %>% set_values(`q` = "amanece", s="Titles")
busqueda %>% submit_form(session=sesion) %>% html_nodes("td.result_text") %>% html_text()
```

```
## Submitting with '<unnamed>'
```

```
##  [1] " Le jour se lève (1939) aka \"Amanece\" "
##  [2] " Amanece (2017) (Short) "
##  [3] " Amanece (2010) (Short) "
##  [4] " La saga Crepúsculo: Amanecer - Parte 2 (2012) "
##  [5] " Amanda Jane Cooper (Actress, Selfie (2014))"
##  [6] " Dan Eckman (Director, Checkout (2006))"
##  [7] " breaking-a-man's-neck (4 titles) "
##  [8] " andaman-sea (2 titles) "
##  [9] " Amanecer Latino [es] (Production) "
## [10] " Amanecer Films [es] (Production) "
```

Extraer cotizaciones del ibex 35 utilizando navegación virtual. Inicio de sesión:

cotiz <- html_session("http://www.bolsamadrid.es/esp/aspx/Mercados/Precios.aspx")

2 Introducción a la minería de texto

2.1 Aplicaciones

La minería de texto nace de una combinación de minería de datos, análisis de texto cuantitativo y procesamiento automático del lenguaje.

Las principales aplicaciones son:

- Motores de búsqueda
- Detección de plagio
- Clasificación de correo electrónico (detección de SPAM)
- Búsqueda de opiniones (evaluaciones positivas o negativas de un servicio, etc.)
- Organización de información (tipologías, ontologías)
- Traducción automática
- ...

2.2 Conceptos básicos

2.2.1 Formato de los datos textuales

Podemos distinguir tres tipos principales de textos:

1. Tablas que consisten en filas y columnas
2. Textos sin formato (extracción de pdf, ...)
3. Documentos semiestructurados (paginas web, correos electrónicos, RSS)

2.2.2 Unidad de análisis: el token

Un **token** es una unidad significativa de texto, a menudo una palabra (o una secuencia de palabras, oración, ..), en la cual estamos interesados en utilizar para un análisis posterior. La **tokenización** es el proceso de dividir el texto en tokens y es uno de los primeros pasos del análisis de textos.

2.2.3 Preparación de los datos textuales

Antes de poder utilizar métodos puramente cuantitativos de minería de textos, se debe preparar los documentos. Como regla general, pasamos por los siguientes pasos (no necesariamente en este orden preciso):

1. Creación del corpus
2. Limpieza y filtraje de la información relevante
3. Análisis del texto e interpretación de los resultados

2.3 Manipulación y análisis básicos de texto

Las tablas bajadas de Internet (y datos procedentes de otras fuentes) exigen frecuentemente un proceso de limpieza de datos o de extracción de la información que contienen.

La función `gsub` se usa muy a menudo para dicha limpieza de datos. Una llamada a `gsub` tiene la forma

```
gsub("h", "H", c("hola", "búho"))
```

```
## [1] "Hola" "búHo"
```

donde el primer argumento, `"h"` es una expresión regular; la función `gsub` modifica las ocurrencias de esta expresión regular por el segundo argumento, `"H"` en este caso. El tercer argumento es un vector que contiene cadenas de texto en las que se realiza la sustitución.

Las expresiones regulares son muy útiles para manipular texto. Conviene aprender algunas de las más frecuentes, como por ejemplo, las que identifican caracteres que aparecen al principio de un texto,

```
gsub("^h", "H", c("hola", "búho"))
```

```
## [1] "Hola" "búho"
```

o al final del mismo,

```
gsub("o$", "os", c("hola", "búho"))
```

```
## [1] "hola"  "búhos"
```

Una función emparentada con `gsub` es `grep`, que busca cadenas en las que aparece una determinada expresión regular:

```
grep("^h", c("hola", "búho"))
```

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

La salida de la expresión anterior nos indica que el patrón *cadena de texto que comienza con la letra h* aparece solo en la posición número 1 del vector.

`colors()` es una función que devuelve el nombre de más de 600 colores en R. Usándolo,

Encontrar * Aquellos cuyo nombre contenga un número (posiblemente tengas que investigar cómo se expresa *cualquier número* como expresión regular) * Aquellos que comiencen con `yellow` * Aquellos que contengan `blue`

Los números que aparecen en la tabla descargada en la sección anterior (y contenidos en `ibex`) no tienen formato numérico. Para convertirlos en números *de verdad*, transfórmalos adecuadamente:

- Usar `gsub` para cambiar "." por "" (i.e., nada) en las columnas de interés. Ten en cuenta que `.` es el comodín de las expresiones regulares; el punto es `\\.`.
- Usar `gsub` para cambiar `,` por `.` en las columnas de interés.
- Finalmente, usar `as.numeric` para cambiar texto resultante por valores numéricos.

Otra función muy útil para procesar texto es `paste`, que tiene un comportamiento distinto según se use con el argumento `sep` o `collapse`.

```
paste("A", 1:6, sep = ",")
```

```
## [1] "A,1" "A,2" "A,3" "A,4" "A,5" "A,6"
```

```
paste("Hoy es ", date(), " y tengo clase de R", sep = "")
```

```
## [1] "Hoy es Fri Jun 01 10:11:42 2018 y tengo clase de R"
```

```
paste("A", 1:6, collapse = ",")
```

```
## [1] "A 1,A 2,A 3,A 4,A 5,A 6"
```

`sep` y `collapse` pueden combinarse:

```
paste("A", 1:6, sep = "_", collapse = ",")
```

```
## [1] "A_1,A_2,A_3,A_4,A_5,A_6"
```

Para la operación inversa, la de partir cadenas de texto, se usa la función `strsplit`:

```
strsplit("Hoy es martes", split = " ")
```

```
## [[1]]
## [1] "Hoy"    "es"     "martes"
```

```
strsplit(c("hoy es martes", "mañana es miércoles"), split = " ")
```

```
## [[1]]
## [1] "hoy"    "es"     "martes"
##
## [[2]]
```

```
## [1] "mañana"    "es"        "miércoles"
```

Advierte que esta función devuelve una lista de cadenas de texto.

Crea una función que tome los nombres de ficheros

```
ficheros <- c("ventas_20160522_zaragoza.csv",                    "pedidos_firmes_20160422_soria.csv")
```

y genere una tabla con una fila por fichero y tres columnas: el nombre del fichero, la fecha y y la provincia. Nota: puedes crear una función que procese solo un nombre de fichero y aplicársela *convenientemente* al vector de nombres.

Esas son las funciones fundamentales para la manipulación básica de texto en R. Existen funciones más ágiles en otros paquetes como `stringro` `tidyr`.

A continuación una aplicación de uso del paquete `stringr` donde se describe como se distribuyen las Medallas Fields (el "nobel" en matemáticas) entre los países, utilizando la información proporcionada por la wikipedia.

Empezamos extrayendo la tabla de interés desde la Wikipedia:

```
require(rvest)
mfield<-read_html("https://es.wikipedia.org/w/index.php?title=Medalla_Fields&oldid=103644843")
mfield %>% html_nodes("table")
```

```
## {xml_nodeset (2)}
## [1] <table class="infobox" style="width:22.7em; line-height: 1.4em; text ...
## [2] <table class="wikitable" border="1">\n<tr>\n<th>Año</th>\n<th>Medall ...
```

```
tabla <- mfield %>% html_nodes("table") %>% .[[2]] %>% html_table(header=TRUE)
knitr:::kable(tabla %>% head(10))
```

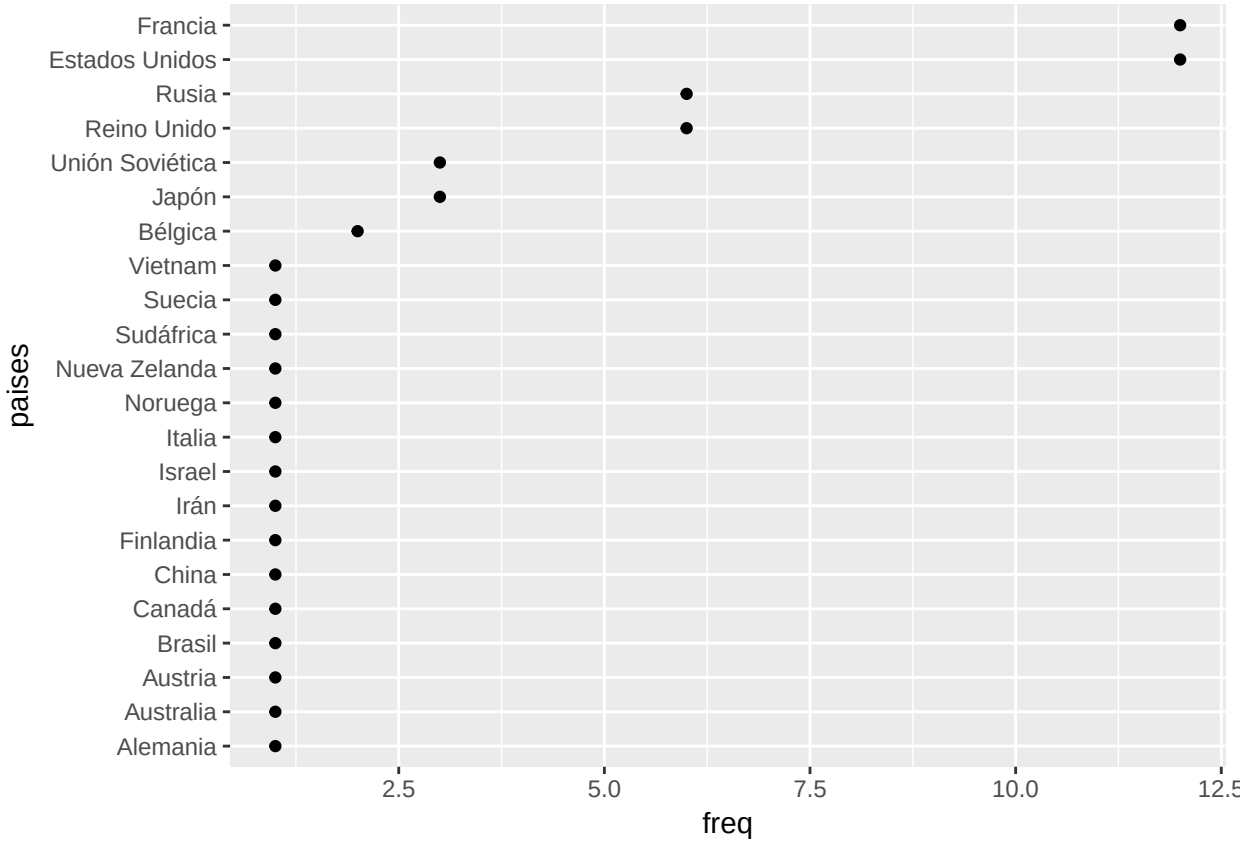
Año	Medallistas
1936	Lars Ahlfors (Finlandia), Universidad Harvard
1936	Jesse Douglas (Estados Unidos), Instituto Tecnológico de Massachusetts
1950	Laurent Schwartz (Francia), Universidad de Nancy
1950	Atle Selberg (Noruega), Instituto de Estudios Avanzados de Princeton
1954	Kunihiko Kodaira (Japón), Universidad de Princeton
1954	Jean-Pierre Serre (Francia), Universidad de París
1958	Klaus Friedrich Roth (Reino Unido), Universidad de Londres
1958	René Thom (Francia), Universidad de Estrasburgo
1962	Lars V. Hörmander (Suecia), Universidad de Estocolmo
1962	John Willard Milnor (Estados Unidos), Universidad de Princeton

Ahora, se extraen los países que vienen entre paréntesis usando expresiones regulares (ver ayuda de R sobre estas expresiones:

```
require(tidyverse)
tmp <- tabla$Medallistas %>% str_extract("\\([^()]+\\)") #extrae contenido entre parentesis
tmp <- substring(tmp,2,nchar(tmp)-1)
paises<- tmp %>% str_split_fixed(" y ", 2) %>% str_trim() %>% c()
```

Representación de distribución de medallas entre los países:

```
freq=c(table(paises))[-1] #el -1 es para quitar la frecuencia de ""
qplot(freq,reorder(names(freq),freq),ylab="paises")
```



2.4 Creación de un Corpus con `tidytext`

El formato de texto `tidy` es básicamente una tabla con un **token** por fila. Este formato se presta muy bien a la minería de datos textuales.

2.4.1 Tokenización con la función `unnest_tokens`

Aquí unas frases extraídas del libro "Niebla" de Unamuno:

```
texto<-c("Eso es insultar al lector, es llamarle torpe","Es decirle: ¡fíjate, hombre, fíjate, que aquí
texto
```

```
## [1] "Eso es insultar al lector, es llamarle torpe"
## [2] "Es decirle: ¡fíjate, hombre, fíjate, que aquí hay intención!"
## [3] "Y por eso le recomendaba yo a un señor que escribiese sus artículos todo en bastardilla"
## [4] "Para que el público se diese cuenta de que eran intencionadísimos desde la primera palabra a la
```

Para analizar este tipo de información textual con `tidytext`, se le da un formato de tabla:

```
require(tidyverse)
texto_df <- data_frame(fila = 1:4, texto = texto)
texto_df
```

```
## # A tibble: 4 x 2
##    fila texto
##   <int> <chr>
```

```
## 1     1 Eso es insultar al lector, es llamarle torpe
## 2     2 Es decirle: ¡fíjate, hombre, fíjate, que aquí hay intención!
## 3     3 Y por eso le recomendaba yo a un señor que escribiese sus artícul~
## 4     4 Para que el público se diese cuenta de que eran intencionadísimos~
```

Todavía esta tabla no permite un análisis del texto. No podemos filtrar las palabras o calcular sus frecuencias, puesto que cada fila se compone de varias palabras combinadas. Necesitamos transformarla de manera que un **token** por fila .

A menudo, el token es una secuencia de caracteres entre dos separadores. Un separador puede ser un "blanco", una puntuación, un paréntesis, etc. Para segmentar el texto en tokens individuales y transformarlo en una estructura de datos utilizamos aquí la función 'unnest_tokens`del paquete`tidytext`:

```
require(tidytext)
```

```
## Loading required package: tidytext
```

```
## Warning: package 'tidytext' was built under R version 3.4.4
```

```
texto_df %>% unnest_tokens(palabra, texto)
```

```
## # A tibble: 51 x 2
##     fila palabra
##    <int> <chr>
##  1     1 eso
##  2     1 es
##  3     1 insultar
##  4     1 al
##  5     1 lector
##  6     1 es
##  7     1 llamarle
##  8     1 torpe
##  9     2 es
## 10     2 decirle
## # ... with 41 more rows
```

Los dos argumentos básicos de esta función son nombres de columnas. Primero tenemos el nombre de la columna de salida que se creará cuando el texto se procese ('palabra' en este caso) y luego la columna de entrada de la que proviene el texto ('texto' en este caso).

Esta función usa el paquete `tokenizers` para separar cada línea de texto en tokens. La tokenización predeterminada es para palabras, pero otras opciones incluyen caracteres, n-grams, oraciones, líneas, párrafos o expresiones regulares. A continuación extraemos bigramms que pueden ser muy útiles para identificar el idioma:

```
require(tidytext)
texto_df %>% unnest_tokens(palabra, texto, token="ngrams", n=2) # bigramm
```

```
## # A tibble: 47 x 2
##     fila palabra
##    <int> <chr>
##  1     1 eso es
##  2     1 es insultar
##  3     1 insultar al
##  4     1 al lector
##  5     1 lector es
##  6     1 es llamarle
##  7     1 llamarle torpe
##  8     2 es decirle
```

```
##  9     2 decirle fíjate
## 10     2 fíjate hombre
## # ... with 37 more rows
```

Después de usar `unnest_tokens`, hemos dividido cada fila para que haya un token (palabra) en cada fila de la nueva base de datos; la tokenización predeterminada en unnest_tokens() es para palabras sueltas. Cabe mencionar que en el resultado:

- Se conservan otras columnas, como el número de la fila de cada palabra.
- La puntuación ha sido eliminada.
- Por defecto, los tokens están en minúsculas, lo que hace más fácil la comparación con otros textos (usar el `to_lower = FALSE` para desactivarlo).

2.4.2 Tokenización de la obra de Jane Austen

Con el formato anterior se puede manejar varios documentos incluyéndolos en una única base. Para ilustrarlo, se considera el texto de las seis novelas publicadas por Jane Austen incluidas en el paquete `janeaustenr`.

En este paquete, los textos vienen en filas que son parecidas a las líneas impresas en un libro físico. A continuación, se anota cada fila por su numero, el capitulo y libro al que pertenece.

```
require(janeaustenr)
libros <- austen_books() %>%
  group_by(book) %>%
  mutate(linenumber = row_number(),
         chapter = cumsum(str_detect(text, regex("^chapter [[:digit:]ivxlc]", ignore_case=TRUE)))) %>%
  ungroup()
```

A partir de esta base de datos textuales ordenados, se puede generar corpus de tokens tal y como se hizo anteriormente mediante la función `unnest_tokens`:

```
tokens <- libros %>% unnest_tokens(word, text)
tokens
```

```
## # A tibble: 725,055 x 4
##    book                linenumber chapter word
##    <fct>                    <int>   <int> <chr>
##  1 Sense & Sensibility          1       0 sense
##  2 Sense & Sensibility          1       0 and
##  3 Sense & Sensibility          1       0 sensibility
##  4 Sense & Sensibility          3       0 by
##  5 Sense & Sensibility          3       0 jane
##  6 Sense & Sensibility          3       0 austen
##  7 Sense & Sensibility          5       0 1811
##  8 Sense & Sensibility         10       1 chapter
##  9 Sense & Sensibility         10       1 1
## 10 Sense & Sensibility         13       1 the
## # ... with 725,045 more rows
```

2.5 Análisis de frecuencias de tokens

Una pregunta recurrente en la minería de textos y el procesamiento del lenguaje natural es tener una idea global de su contenido. Para este propósito se puede proporcionar las palabras más frecuente incluidas en el texto.

Sin embargo, hay palabras que ocurren muchas veces pero que no caracterizan el texto; en castellano, palabras como “del”, “es”, “para”, ... Es por lo tanto, importante disponer de una lista de dichas palabras para eliminarlas antes del análisis.

Estas palabras “inútiles” están incluidas en el paquete `stopwords` y se pueden quitar del corpus mediante la función `anti_join`. El propio paquete `tidytext`contiene una base llamada `stop_words`de estas palabras, pero sólo en ingles.

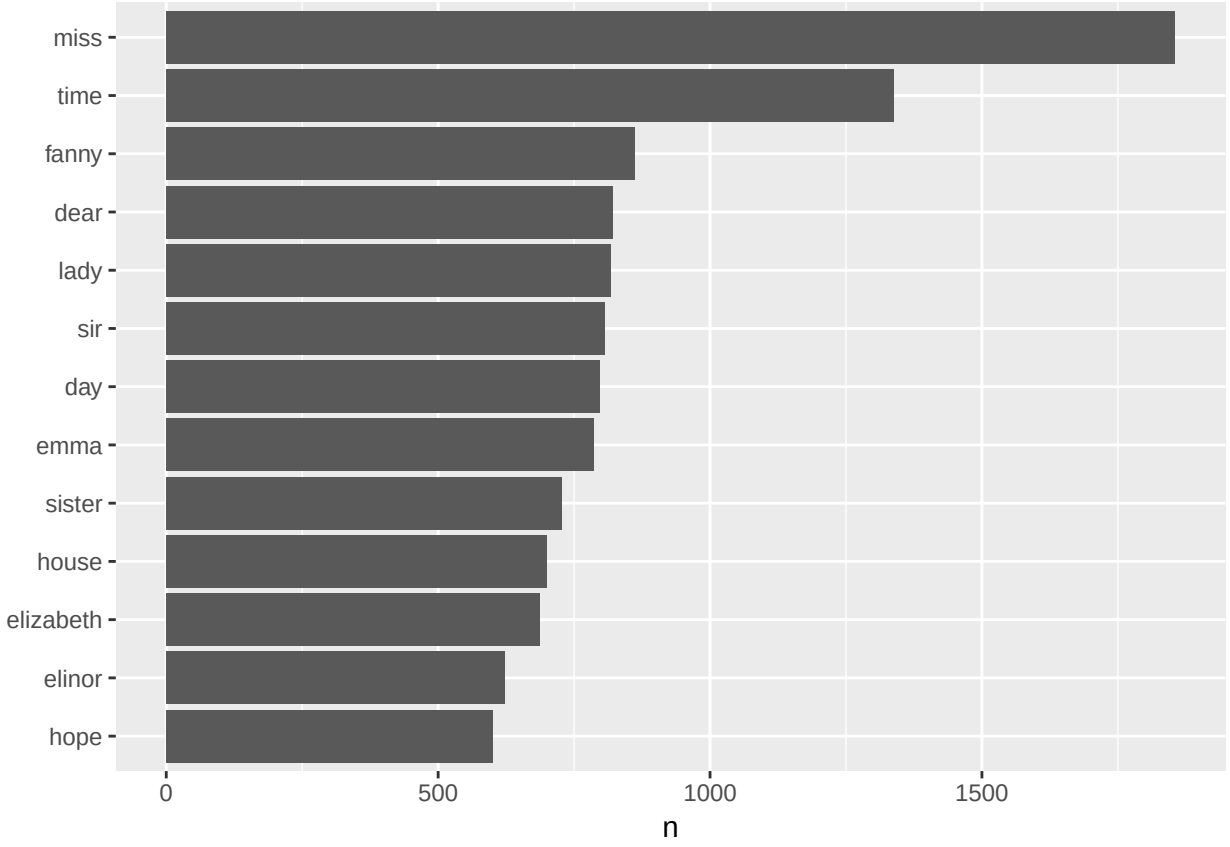
```
tokens <- tokens %>% anti_join(stop_words)
```

Podemos ahora procurar caracterizar la obra de Jane Austeen calculando las frecuencias de las palabras incluidas en sus novelas:

```
freq <- tokens %>% count(word, sort = TRUE)
```

Y representar la distribución de las palabras más frecuentes:

```
require(ggplot2)
freq %>%
  filter(n > 600) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n)) +
  geom_col() +
  xlab(NULL) +
  coord_flip()
```



O, mediante una nube de palabras utilizando el paquete `wordcloud`:

```
require(wordcloud)
```

```
## Warning: package 'wordcloud' was built under R version 3.4.4
```

```
wordcloud(words = freq$word, freq = freq$n, min.freq = 300,
          max.words=100, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Dark2"))
```

Otro enfoque es observar la frecuencia inversa de documentos (`idf`) para una palabra dada, que aumenta su peso si se usa en pocos documentos de la colección:

Si N denota el número total de documentos y N_p el número de documentos que contienen la palabra p, entonces el `idf` de dicha palabra es:

$$\text{idf} = -\log\left(\frac{N_p}{N}\right)$$

Esto se puede combinar con la frecuencia de la palabra `tf` para calcular el `tf-idf` de un término (`producto de idf`y `tf`), es decir, la frecuencia de una palabra multiplicado por su especificidad al documento.

Por lo tanto, la medida `tf-idf` mide hasta que punto una palabra caracteriza un documento dado dentro de una colección (o corpus) al cual pertenece dicho documento.

Si lo aplicamos a las novelas de Jane Austeen, obtenemos:

```
book_words <- austen_books() %>% unnest_tokens(word, text) %>%
  count(book, word, sort = TRUE) %>%
  ungroup()
```

```
freq_rel <- book_words %>% bind_tf_idf(word, book, n)
freq_rel
```

```
## # A tibble: 40,379 x 6
##    book              word      n     tf   idf tf_idf
##    <fct>             <chr> <int>  <dbl> <dbl>  <dbl>
##  1 Mansfield Park    the    6206 0.0387    0.     0.
##  2 Mansfield Park    to     5475 0.0341    0.     0.
##  3 Mansfield Park    and    5438 0.0339    0.     0.
##  4 Emma              to     5239 0.0325    0.     0.
##  5 Emma              the    5201 0.0323    0.     0.
##  6 Emma              and    4896 0.0304    0.     0.
##  7 Mansfield Park    of     4778 0.0298    0.     0.
##  8 Pride & Prejudice the    4331 0.0354    0.     0.
##  9 Emma              of     4291 0.0267    0.     0.
## 10 Pride & Prejudice to     4162 0.0341    0.     0.
## # ... with 40,369 more rows
```

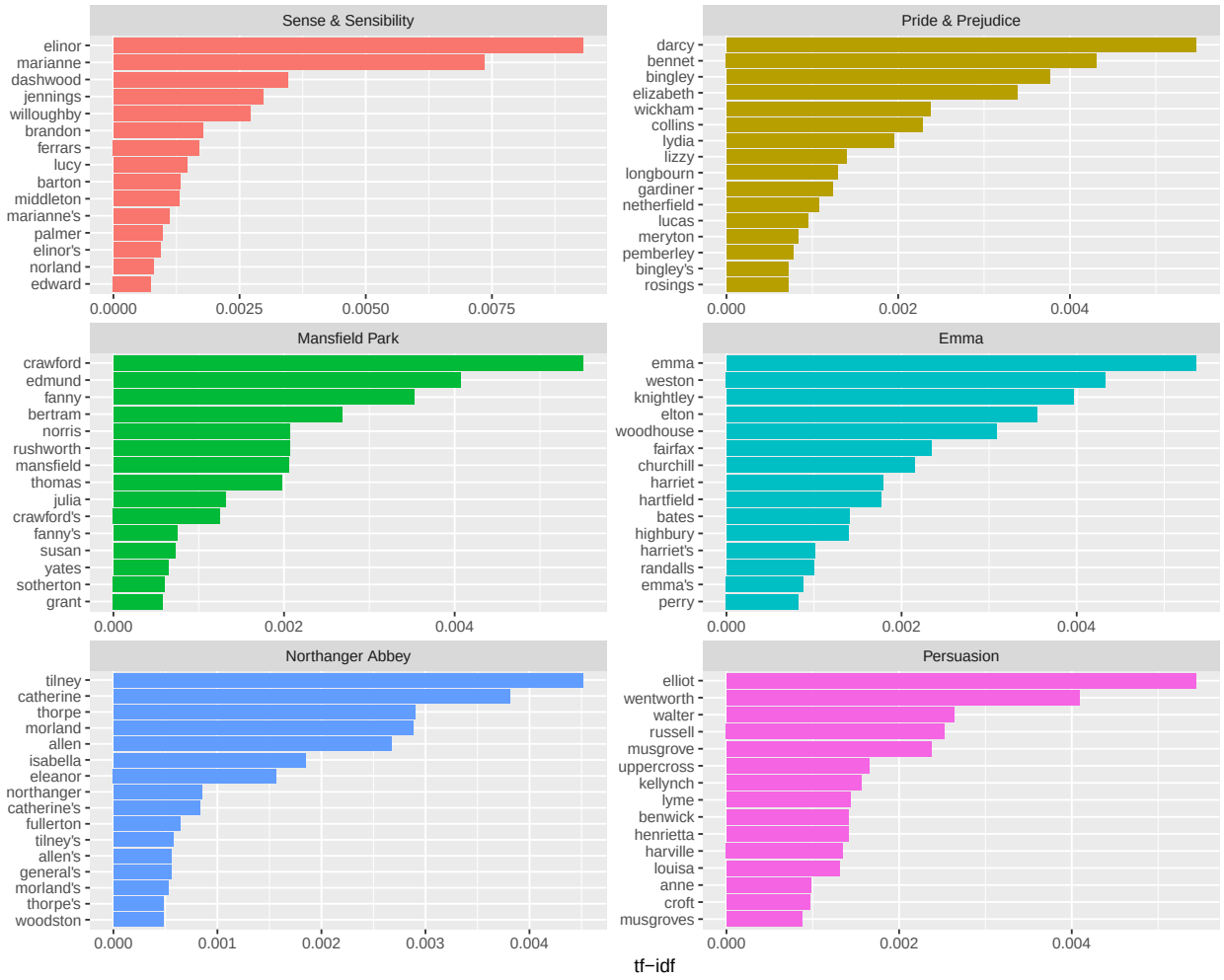
Podemos observar que para estas palabras de uso muy corriente `idf` es igual a cero. Para ver las palabras con una elevada importancia escribimos:

```
freq_rel %>% arrange(desc(tf_idf))
```

```
## # A tibble: 40,379 x 6
##    book                word          n      tf   idf  tf_idf
##    <fct>               <chr>     <int>   <dbl> <dbl>   <dbl>
##  1 Sense & Sensibility elinor      623 0.00519  1.79 0.00931
##  2 Sense & Sensibility marianne    492 0.00410  1.79 0.00735
##  3 Mansfield Park      crawford    493 0.00307  1.79 0.00551
##  4 Pride & Prejudice   darcy       373 0.00305  1.79 0.00547
##  5 Persuasion          elliot      254 0.00304  1.79 0.00544
##  6 Emma                emma        786 0.00488  1.10 0.00536
##  7 Northanger Abbey    tilney      196 0.00252  1.79 0.00452
##  8 Emma                weston      389 0.00242  1.79 0.00433
##  9 Pride & Prejudice   bennet      294 0.00241  1.79 0.00431
## 10 Persuasion          wentworth   191 0.00228  1.79 0.00409
## # ... with 40,369 more rows
```

Y así podemos representar una caracterización de cada novela mediante dichas palabras:

```
freq_rel %>% arrange(desc(tf_idf)) %>%
  mutate(word = factor(word, levels = rev(unique(word)))) %>%
  group_by(book) %>%
  top_n(15) %>%
  ungroup() %>%
  ggplot(aes(word, tf_idf, fill = book)) +
  geom_col(show.legend = FALSE) +
  labs(x = NULL, y = "tf-idf") +
  facet_wrap(~book, ncol = 2, scales = "free") +
  coord_flip()
```



Caracterizar algunos capítulos de "Pride and Prejudice" mediante el indicador `tf-idf`.

Descargar 20 discursos del rey de España y caracterizarlos. Utilizar la dirección siguiente donde *data* corresponde al número del discurso.

http://www.casareal.es/ES/Actividades/Paginas/actividades_discursos_detalle.aspx?data=5738

De manera general, se puede importar libros mediante el proyecto Gutenberg y el paquete `gutenbergr` (Robinson, 2016). Así, se puede importar la novela "Niebla" de Unamuno, de la siguiente manera:

```
require(gutenbergr)
unamuno <- gutenberg_works(title=="Niebla\n(Nivola)",languages="es")$gutenberg_id %>%
gutenberg_download() %>%
gutenberg_strip() #quita encabezado y pie de pagina
```

3 Referencias

Este curso está basado en los siguientes textos:

- *R para profesionales de los datos*, Carlos Bellosta, 2017, `https://datanalytics.com/libro_r`
- *Text Mining with R, A Tidy Approach*, Julia Silge and David Robinson, 2018, `https://www.tidytextmining.com/tidyte`

- *Scraping HTML Text*, Bradley Boehmke, 2015, `http://bradleyboehmke.github.io/2015/12/scraping-html-text.ht`