WEB SCRAPING

Sebastian Niklitschek

INTRODUCCIÓN

- La “world wide web” es sin duda el principal medio de transmisión de información a nivel mundial;
- Vastas cantidades de datos son almacenados en línea, tanto estructurados como no estructurados;
- Recoger estos datos desde la web no es una tarea trivial, existen distintos tipos de tecnologías desarrolladas para compartir esta información:
  - HTML; XML; JSON.

INTRODUCCIÓN

- Dada esta tremenda variedad en herramientas y formatos necesitamos aprender a procesar estos datos con R;
- Aprenderemos a hacer lo siguiente:
  - Cargar archivos almacenados en la red;
  - Rasgar texto html;
  - Rasgar datos tabulados en HTML;
  - Utilizar API para descargar los datos.

IMPORTAR ARCHIVOS ALMACENADOS ONLINE

- Una de las maneras más simples de obtener datos almacenados en línea es importar datos tabulados almacenados en algún servidor;
- Esto usualmente no es considerado “web scraping” pero es un buen punto de partida para comenzar a interactuar con la red para obtener información;
- Hay muchas fuentes de datos tabulados en línea, un gran repositorio es el portal data.gov. Comencemos por descargar el archivo data.gov.csv.

IMPORTAR ARCHIVOS ALMACENADOS ONLINE

```r
# La url del conjunto de datos
url <- "https://s3.amazonaws.com/bsp-ocsit-prod-east-appdata/datagov/wordpress/agency-participation.csv"

# usaremos read.csv para importar los datos
data_gov <- read.csv(url, stringsAsFactors = FALSE)

# Despleguemos las primeras seis filas
head(data_gov)
```

ARCHIVOS ALMACENADOS ONLINE

- Recordemos que no existen funciones en el paquete de base que nos permitan procesar archivos excel, sin embargo existen muchas librerías auxiliares que nos permiten hacerlo;
- Una librería que nos permitirá extraer planillas excel almacenadas en línea es “gdata”, mediante su función read.xls():

```
library(gdata)
# La url del conjunto de datos
url <- "http://www.huduser.org/portal/datasets/fmr/fmr2015f/FY2015F_4050_Final.xls"

# # usaremos read.xls para importar los datos
rents <- read.xls(url)

rents[1:6, 1:10]
```

ARCHIVOS ALMACENADOS ONLINE

- Otro formato frecuente para los ficheros almacenados en línea es .zip;
- En este caso utilizaremos la función download.file() para descargar los datos y almacenarlos en el directorio que deseemos:

```
url <- "http://www.bls.gov/cex/pumd/data/comma/diary14.zip"

# Descargar el .zip y descomprimir el contenido
download.file(url, dest = "dataset.zip", mode = "wb")
unzip("dataset.zip", exdir = "./")

# Ejemplo:
list.files("diary14")
##  [1] "dtbd141.csv" "dtbd142.csv" "dtbd143.csv" "dtbd144.csv" "dtid141.csv"
##  [6] "dtid142.csv" "dtid143.csv" "dtid144.csv" "expd141.csv" "expd142.csv"
## [11] "expd143.csv" "expd144.csv" "fmld141.csv" "fmld142.csv" "fmld143.csv"
## [16] "fmld144.csv" "memd141.csv" "memd142.csv" "memd143.csv" "memd144.csv"

# Podmeos utilizar unz() para descomprimir archivos particulares:
zip_data <- read.csv(unz("dataset.zip", "diary14/expd141.csv"))
zip_data[1:5, 1:10]
```

ARCHIVOS ALMACENADOS ONLINE

- La extensión .zip es utilizada para reducir el tamaño de los archivos almacenados en línea;
- Es posible que queramos descargar sólo parte del archivo y no almacenar el archivo completo en la memoria física de nuestro computador;
- Esto puede hacerse con el siguiente procedimiento que sólo almacena el .zip de manera temporal en el computador:

```r
# Creamos un .temp
temp <- tempfile()

# Usamos download.file() para cargar el archivo en el archivo temporal
download.file("http://www.bls.gov/cex/pumd/data/comma/diary14.zip",temp)

# Usamos unz() para extraer el contenido que nos interesa
zip_data2 <- read.csv(unz(temp, "diary14/expd141.csv"))

# Removemos el archivo temporal utilizando unlink()
unlink(temp)

zip_data2[1:5, 1:10]
```

EJERCICIOS

- Descargue el siguiente archivo csv:

https://bradleyboehmke.github.io/public/data/reddit.csv

- Descargue el siguiente archivo xlsx:

http://www.huduser.gov/portal/datasets/fmr/fmr2017/FY2017_4050_FMR.xlsx

- Descargue los datos de clima de una ciudad a su elección desde el enlace:

http://academic.udayton.edu/kissock/http/Weather/citylistUS.htm

RASGANDO TEXTO HTML

- Mucha de la información que existe en la red aparece en la forma de una secuencia interminable de páginas web;
- Mucha de esta información es texto no estructurado que podría ser útil en nuestro análisis;
- Aprenderemos a extraer texto de páginas web, mediante un ejemplo desarrollado sobre la página web de wikipedia.

HTML

- Los elementos HTML son siempre escritos utilizando alguna etiqueta, la forma de hacerlo es encerrando el contenido a desplegar utilizando la siguiente sintaxis

<tag> Contenido </tag>

- Usualmente el contenido textual que se despliega en las páginas está ingresado utilizando estas etiquetas. Algunas etiquetas frecuentes son:
  - <h1>, <h2>, … : Encabezados;
  - <p> : Párrafos;
  - : Listas no ordenadas;
  - : listas ordenadas;
  - : División o sección;
  - <table> : Tabla.

RASGAR TEXTO DE INTERNET

- Para poder descargar contenido textual de páginas de internet, utilizaremos la librería rvest;
- Esta librería provee diversas funcionalidades, que van más allá de las que revisaremos aquí;
- Para extraer texto de una página, primero debemos especificar qué elementos html queremos seleccionar, utilizando la función html_nodes();
- Supongamos que queremos rasgar la página:

https://en.wikipedia.org/wiki/Web_scraping

WEB SCRAPING HTML

- La función html_nodes() identificará los encabezados de la página:

```
library(rvest)

scraping_wiki <- read_html("https://en.wikipedia.org/wiki/Web_scraping")

scraping_wiki %>%
  html_nodes("h1")
```

- Para extraer sólo el texto correspondiente al encabezado <h1>, utilizamos la función html_text():

```
scraping_wiki %>%
  html_nodes("h1") %>%
  html_text()
```

ENCABEZADOS

- Si queremos detectar todos los encabezados en el segundo nivel, debemos elegir los nodos etiquetados como “h2”:

```
scraping_wiki %>%
  html_nodes("h2") %>%
  html_text()
```

- Ahora podemos extraer el texto de esta página que está principalmente contenido en párrafos;
- Para ello, seguimos el mismo procedimiento seguido hasta ahora pero rescatando la información contenida con la etiqueta “p”:

```
p_nodes <- scraping_wiki %>%
  html_nodes("p")

length(p_nodes)
p_nodes[1:6]
```

```
p_text <- scraping_wiki %>%
  html_nodes("p") %>%
  html_text()

p_text[1]
```

LISTAS

- Notemos que si bien pudimos obtener parte de la información, no logramos capturarla toda;
- Esto se debe a que parte de la información en la página se encuentra en formato lista;
- Para rasgar esta información, necesitaremos rescatar todo lo que se encuentre desplegado bajo la etiqueta “ul”:

```
ul_text <- scraping_wiki %>%
  html_nodes("ul") %>%
  html_text()

length(ul_text)
ul_text[1]
substr(ul_text[2], start = 1, stop = 200)
```

ALTERNATIVA

- Otra posibilidad es rasgar todo el contenido que se encuentra bajo la etiqueta “li”;
- Esto descargará todo el texto contenido en todas las listas, de cualquier tipo;

```
li_text <- scraping_wiki %>%
  html_nodes("li") %>%
  html_text()

length(li_text)
li_text[1:8]
```

OBSERVACIÓN

- Podemos pensar que en este punto ya tenemos toda la información que queremos;
- Podríamos entonces proceder a unir la información de cada párrafo con la información de las listas de modo de poder realizar un análisis de minería de texto;
- Sin embargo, notemos que hemos descargado más contenido del necesario, por ejemplo, los enlaces contenidos en el margen izquierdo de la página también fueron capturados:

```
li_text[104:136]
```

OBSERVACIÓN

- Si lo anterior no representa un problema, entonces existe un enfoque más directo;
- Podemos rasgar todo el contenido textual contenido en la página, sin importar su naturaleza, si utilizamos la etiqueta "div":

```
all_text <- scraping_wiki %>%
  html_nodes("div") %>%
  html_text()
```

RASGAR NODOS ESPECÍFICOS

- Sin embargo, lo más frecuente es que estemos interesados en rescatar contenido específico en la página;
- Para hacer esto, tendremos que utilizar herramientas en nuestro navegador para ser capaces de determinar qué nodo contiene la información que nos interesa (ver chrome);
- Ahora, podemos utilizar esta información para descargar sólo el texto que nos interesa:

```
body_text <- scraping_wiki %>%
  html_nodes("#mw-content-text") %>%
  html_text()

substr(body_text, start = 1, stop = 207)
substr(body_text, start = nchar(body_text)-73, stop = nchar(body_text))
```

EJERCICIOS

- Intente rasgar el texto del primer capítulo de la Piedra Filosofal del siguiente enlace:

  http://www.readbooksvampire.com/J.K._Rowling/Harry_Potter_and_the_Philosophers_Stone/01.html

- Para ello:
  - Importe el contenido HTML/XML;
  - El contenido del libro está etiquetado bajo la etiqueta <td>;
  - Extraiga el contenido de los nodos de la etiqueta <td>.