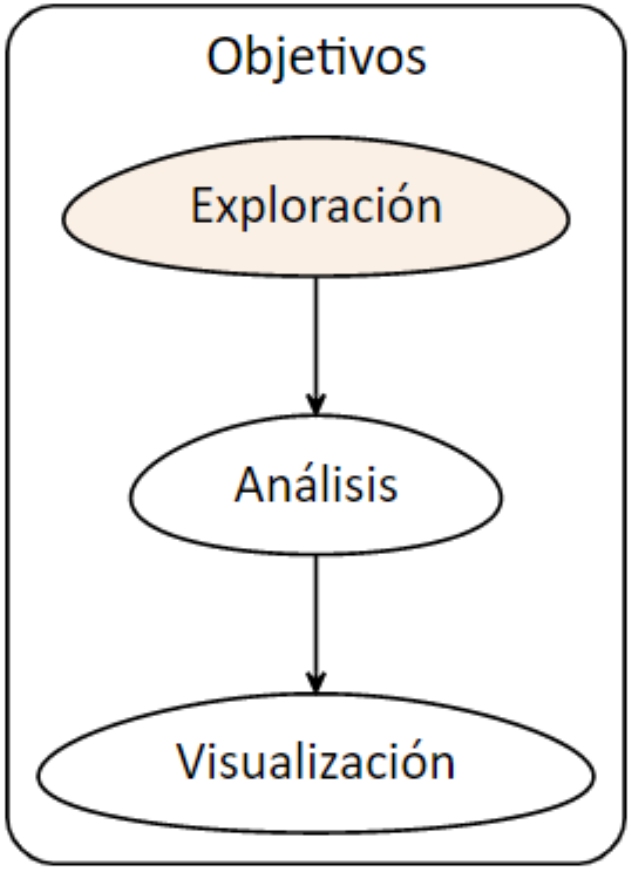
Introducción a la programación y análisis de texto con R

Clase 4 - Licenciatura en Ingeniería de Medios (UdelaR)

Objetivos de hoy



Fuentes de datos

Las fuentes de datos que vamos a ver son:

1. Recuperación de documentos en imagen o pdf (OCR)
2. Scraping web y parlamentario
3. Prensa digital
4. Google Trends
5. Audio
6. YouTube

Objetivos de hoy

- Fuentes de datos:

1. Recuperación de documentos en imagen o pdf (OCR)
2. Scraping web y parlamentario
3. Prensa digital

Previo: carga de archivos de texto

Existen diferentes librerías de R que nos permiten recuperar documentos en diferentes formatos:

- readtext
- pdftools

readtext

- El paquete readtext tiene una función con le mismo nombre *readtext()* que permite cargar archivos en cualquier formato de texto (txt, pdf, doc, docx, odt o incluso alojado en uno de estos formatos en la web).
- `readtext::readtext()`

```
library(readtext)
##Abro los textos en formato .txt y visualizo cómo los carga
txt <- readtext::readtext("Clase4/Material/Mujeres_Adultos_1.txt")
# Determinamos el pdf con el que trabajar
pdf <- readtext("Clase4/Material/text.pdf")
url <- readtext("https://www.ingenieria.unam.mx/dcsyhfi/material_didactico/Literatura_Hispanoamericana_Cor
```

pdftools

Para recuperar textos en pdf existe la librería pdftools que se basa en el paquete *Rpoppler* (Kurt Hornik).

- `pdftools::pdf_text()`

```
library(pdftools)
# Extraemos el texto
pdf_texto <- pdf_text("Clase4/Material/marcha_1973.pdf")
```

1. Recuperación de documentos en imagen o pdf (OCR)

Tesseract es un motor de OCR *(reconocimiento óptico de caracteres)* para varios sistemas operativos. Es software libre, liberado bajo la licencia Apache, Versión 2.0 y su desarrollo es financiado por Google desde el 2006.

Acá se encuentra la documentación, cuenta con *más de 100 idiomas.*

tesseract OCR

Existe un paquete de R bien documentado que se llama *tesseract* y que cuenta con funciones que permiten el reconocimiento de caracteres incluso en español, descargando una base de entrenamiento del motor.

tesseract OCR

Descargo un documento histórico del repositorio Internet Archive

```
##Chequear los idiomas disponibles
tesseract_info()
# Bajar por unicamente español para entrenar
tesseract_download("spa")
# asignar
(espanol <- tesseract("spa"))
#Probamos:
transcribopdf <- ocr("analesUruguay.pdf", engine = espanol)
```

tesseract OCR

La función *ocr_data()* devuelve una tabla dónde cada fila es una palabra con la confianza asociada a la misma y la ubicación exacta.

magick

El paquete *magick* complementa a *tesseract* en cuento a mejora de la calidad de las imagenes que sirven de input. Cuenta con varias funciones para mejorar la resolución, el color, contraste, espacios en blanco. Puede ser utilizado como paso previo.

Ejercicio 1

Reconocimiento óptico de caracteres

1. Replicar el OCR para los archivos *analesUruguay3* y *marcha_1973*
2. Hacer la tabla de ambas

2. Web scraping

¿Qué es web scraping?

Web scraping es una **técnica** para obtener datos no estructurado (etiquetas HTML) de una página web, a un formato estructurado de columnas y renglones, en los cuales se puede acceder y usar más fácilmente.

2. Web scraping

¿Para qué sirve Web scraping?

- Obtener datos de texto.
- Consolidar datos de redes sociales o extraer comentarios de usuarios/as.
- Precios de tiendas online, a través del análisis histórico de la competencia.
- Búsqueda en Google de diversas palabras clave.
- Etiquetas de imágenes, para clasificación de imágenes.

2. Scraping web y parlamentario

En el curso vamos a ver tres formas de Web scraping:

- Paquete *rvest*
- Paquete *speech* (Uruguay)
- Gdelt project

rvest

rvest es un paquete para scraping (raspado) y análisis web de Hadley Wickham.

Documentación

- Tutorial recomendado de Riva Quiroga (Chile)

https://programminghistorian.org/es/lecciones/introduccion-al-web-scraping-usando-r

¿Cómo usar rvest?

Para usar rvest, se requiere conocer las instrucciones en código, a las que llamaremos funciones, para para hacer las tareas más comunes en la extracción y manipulación de datos web.

- `read_html(«url»)` con esta función se crea un objeto que contiene todo el código o etiquetas HTML.
- `html_elements(«objeto html», «etiqueta css»)` se usa para seleccionar partes del objeto que contiene todo el código html. El segundo parámetros es la clase CSS que está relacionada con la sección que deseamos extraer.

¿Cómo usar rvest?

- `html_elements()` devuelve los elementos html seleccionados
- `html_name()` devuelve el nombre de un elemento html
- `html_attr()` regresa los atributos específicos html (ej. href)
- `html_text()` extrae el texto html
- `html_table()` convierte una tabla html en una estructura de datos en R

Ejemplo rvest: texto

- Opción 1: Descargo la extensión del SelectorGadget de Chrome e instalo y busco el nombre del nodo o elementos en una pagina que me interese scrapear
- Opción 2: Usar las herramientas de desarrollo de los navegadores a través de la opción *inspect* o *inspeccionar* que muestra el código html de la página y las reglas de estilo (CSS)
- Opción 3: Usar plataformas web con herramientas de scrapeo como Apify

Ejemplo rvest: texto

```
library(rvest)
library(dplyr)

#Defino mi sitio html: Montevideo portal
mvdportal = read_html("https://www.montevideo.com.uy/index.html")

resumenes = mvdportal %>%
  html_elements(".text")%>% #defino los elementos que identifiqué con el SelectorGadget
  html_text()

titulares = mvdportal %>%
  html_elements("a")%>%
  html_text()
```

Ejemplo rvest: texto

Un ejemplo concreto para el caso uruguayo !

Ejemplo rvest: tabla

```
url <- 'https://es.wikipedia.org/wiki/Anexo:Ríos_de_Uruguay'

url %>% read_html() %>%
  html_elements(css = '.wikitable') %>%
  html_table()
```

Ejercicio 2

Scrapeo web con rvest

1. Descargar noticias o información de otra web
2. Scapear dos elementos html diferentes

speech

The `speech` package

Nicolás Schmidt, Diego Luján, Juan Andrés Moraes

CRAN 0.1.0 – a year ago | devel version 0.1.1 | R-CMD-check passing | repo status Active

downloads 289/month | DOI 10.5281/zenodo.3766618

Description

Converts the floor speeches of Uruguayan legislators, extracted from the parliamentary minutes, to tidy data.frame where each observation is the intervention of a single legislator.

Installation

```
# Install speech from CRAN
install.packages("speech")

# The development version from GitHub:
if (!require("remotes")) install.packages("remotes")
remotes::install_github("Nicolas-Schmidt/speech")
```

speech

El paquete speech convierte los diarios de sesiones legisladorxs uruguayxs, en un marco de datos ordenado donde cada observación es la intervención de unx solx legisladorx.

Acá se encuentra la documentación del paquete con descripción de las funciones y argumentos.

speech

```
##Recomiendo instalar versión en desarrollo:

if (!require("remotes")) install.packages("remotes")
remotes::install_github("Nicolas-Schmidt/speech")

library(speech)
```

speech

```
url <- "https://parlamento.gub.uy/documentosyleyes/documentos/diarios-de-sesion/5515/IMG"

sesion <- speech::speech_build(file = url)
```

speech

```
#Función completa

sesion <- speech::speech_build(file = url,
#url a pdf
compiler = FALSE,
#compila discursos de unx mismx legisladorx
quality = TRUE,
#aporta dos índices de calidad
add.error.sir = c("SEf'IOR"),
##forma errónea que lo que identifica a el/la legisladorx
rm.error.leg = c("PRtSIDENTE", "SUB", "PRfSlENTE"))
##identifica a el/la legisladorx que debe eliminarse
```

speech

Variables que incluye la tabla ordenada:

- legislator: nombre
- speech: discurso/s
- date: fecha de sesión
- id: identificador
- legislature: número de legislatura
- chamber: cámara del documento (representantes, senadores, asamblea general, comisión permanente)

Si quality es TRUE:

- index_1: index_1. Proporción del documento recuperado con respecto al original.
- index_2: index_2. Proporción del documento final en función del recuperado. Proporción del documento donde hay intervenciones de lxs legisladorxs.

puy

- Es posible combinar con el paquete *puy* para recuperar el dato del partido político al que pertenece
- `puy::add_party()`

```
#agrego partido político
sesion <- puy::add_party(sesion)
```

speech App

- Existe una Shiny de speech que permite descargar de forma tabulada las sesiones sin escribir código: https://bancodedatos-fcs.shinyapps.io/shiny_speech/

Ejercicio 3

Scrapeo parlamentario con speech

1. Elegir una sesión parlamentaria
2. Aplicar la funcion speech_build
3. Agregar etiqueta partidaria
4. Guardar en formato tabulado

3. Prensa digital

Monitor de prensa

- Existe un monitor de prensa (en Twitter) que permite descargar http://137.184.138.178
- Desarrollada por Leandro Domínguez, Guillermo Eijo y Sebastian Felix en el marco del proyecto de grado "Análisis de publicaciones sobre seguridad ciudadana en redes sociales" (FING-Udelar) - Agosto 2022
- Acumula desde enero 2009. Tiene tres módulos: Indicadores, Entidades y Cluster.

Consulta combinada de palabras claves: ["Lacalle Pou", "Cabildo Abierto"] -> Lacalle Pou y Cabildo Abierto