

# **Recuperación y análisis de texto con R**

## **Clase 2 - Educación Permanente FCS**

**Mag. Elina Gómez (UMAD)**

[elina.gomez@cienciassociales.edu.uy](mailto:elina.gomez@cienciassociales.edu.uy)

[www.elinagomez.com](http://www.elinagomez.com)

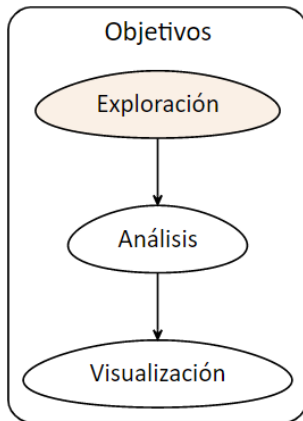
**Mag. Gustavo Méndez Barbato**

[gustavo.mendez@cienciassociales.edu.uy](mailto:gustavo.mendez@cienciassociales.edu.uy)



Este trabajo se distribuye con una licencia Creative Commons Attribution-ShareAlike 4.0 International License

# 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: OCR, Scraping web y parlamentario, prensa digital

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

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

- [readtext](#)
- [pdftools](#)

## tesseract 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 cuanto a mejora de la calidad de las imágenes que sirven de input. Cuenta con varias funciones para mejorar la resolución, el color, contraste, espacios en blanco. Puede ser utilizado como paso previo.

## 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_name()` obtiene los atributos html
- `html_text()` extrae el texto html
- `html_attr()` regresa los atributos específicos html (ej. href)
- `html_attrs()` obtiene los atributos html
- `html_table()` convierte una tabla html en una estructura de datos en R

## Ejemplo rvest: texto

Descargo la extensión del [SelectorGadget](#) de Chrome e [instalo](#) y busco el nombre del nodo o elementos en una pagina que me interese scrapear

## 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://en.wikipedia.org/wiki/R_(programming_language)'  
  
url %>% read_html() %>%  
  html_elements(css = '.wikitable') %>%  
  html_table()
```

# 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 <- "http://bit.ly/35AUVF4"  
session <- speech_build(file = url)
```

# speech

```
#Función completa

sesion <- speech_build(file = url,
#url a pdf
compiler = FALSE,
#compila discursos de una misma 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", "PRfSIENTE"))
##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

## *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/](https://bancodedatos-fcs.shinyapps.io/shiny_speech/)

## 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.

## Proyecto Gdelt

El proyecto **GDEL**T cuenta con *una base de datos global de la sociedad que monitorea las noticias de impresas y web del mundo desde casi todos los rincones de cada país en más de 100 idiomas e identifica las personas, ubicaciones, organizaciones, temas, fuentes, emociones, recuentos, citas, imágenes y eventos que impulsan nuestra sociedad global cada segundo de cada día, creando una plataforma abierta y gratuita para la informática en todo el mundo.*



## *Proyecto Gdelt*

Existe un paquete de R llamado [gdeltr2](#) que no se encuentra bien documentado pero que cuenta con mucho potencial. Las consultas a la base pueden hacerse también desde [Big Query de Google](#) y procesamiento posterior en R.

## *Proyecto Gdelt*

- GDELT Events Database [EVENTS]: Global Events, 1979 to present.
- GDELT Global Knowledge Graph [GKG] : GDELT's Knowledge Graph, April 2013 to present.
- GDELT Full Text API [Full Text API]: Full text search for all monitored sources within a 24 hour window. Output includes raw data, sentiment, and word counts.
- GDELT Visual Knowledge Graph VGKG: Google Cloud Vision API output for every indexed piece of GKG media.

## *Proyecto Gdelt*

Proyecto reciente en Argentina usando Gdelt para obtener noticias sobre antivacunismo.

# *gdelt2*

## Tutorial

### Instalación:

```
devtools::install_github("hadley/devtools")  
devtools::install_github("hafen/trelliscopejs")  
devtools::install_github("abresler/gdeltr2")
```

## *gdelt2*

El mode *ArtList* recupera todo los artículos que tienen esa mención en un determinado tiempo. Está restringido a 250 resultados y 52 semanas. Para hacer búsquedas combinadas: ‘“Lacalle Pou” covid’

```
articulos = gdelt2::ft_v2_api(  
  terms = c("Lacalle Pou"),  
  modes = c("ArtList"),  
  visualize_results = F,  
  timespans = "55 days",  
  source_countries = "UY")
```

## gdelt2

El mode *TimelineVol* recupera una métrica diaria de la intensidad del volumen de los artículos que coinciden con una búsqueda específica.

```
intensidad = gdeltr2::ft_v2_api(  
  terms = c("Lacalle Pou"),  
  modes = c("TimelineVol"),  
  visualize_results = F,  
  timespans = "55 days",  
  source_countries = "UY"  
)
```

## *gdelt2*

El mode *TimelineVol* recupera una métrica diaria de la intensidad del volumen de los artículos que coinciden con una búsqueda específica. El mode *TimelineVolInfo* es igual pero con información anexa y desagregada para cada artículo.

```
intensidad = gdeltr2::ft_v2_api(  
  terms = c("Lacalle Pou"),  
  modes = c("TimelineVol"),  
  visualize_results = F,  
  timespans = "55 days",  
  source_countries = "UY"  
)
```

## gdelt2

El mode *TimelineTone* recupera el tono (positivo y negativo) de los artículos que conciden con la búsqueda, por día.

```
tono_diario = gdelt2::ft_v2_api(  
  terms = c("Lacalle Pou"),  
  modes = c("TimelineTone"),  
  visualize_results = F,  
  timespans = "30 days",  
  source_countries = "UY"  
)
```



## *gdelt2*

El mode *ToneChart* recupera el tono (positivo y negativo) de los artículos que conciden con la búsqueda, por artículo.

```
prueba4 = gdelt2::ft_v2_api(  
  terms = c("Lacalle Pou"),  
  modes = c("ToneChart"),  
  visualize_results = F,  
  timespans = "30 days",  
  source_countries = "UY"  
)
```

## *gdelt2*

Últimos términos, lugares, personas, cosas, de los últimos 15 minutos a nivel mundial.

```
ultimo = gdeltr2::ft_trending_terms()
```

## gdelt2

Tablas de inestabilidad con variables *'instability'*, *'tone'*, *'protest'*, *'conflict'*, *'artvolnorm'* . Es posible visualizar gráficos.

```
inestabilidad_zona <-  
  gdelt2::instability_api_locations(  
    location_ids = c("UY"),  
    use_multi_locations = c(T, F),  
    variable_names = c('instability', 'tone', 'protest', 'conflict','artvolnorm'),  
    time_periods = c('daily'),  
    nest_data = F,  
    days_moving_average = NA,  
    return_wide = T,  
    return_message = T,  
    visualize = T  
  )
```

## gdelt2

Por último, recuperar temas pre-calasificados con AA (IA), hay 59840.

```
##carga códigos de temas
df_gkg <-
  gdelt2::dictionary_ft_codebook(code_book = "gkg")

tema = ft_v2_api(gkg_themes = "WB_2901_GENDER_BASED_VIOLENCE", modes = c("Artlist"),
  visualize_results = F,
  timespans = "55 days")
```