

Recuperación y análisis de texto con R

Clase 1 - Educación Permanente (FCS-UdelaR)

Mag. Elina Gómez (UMAD)

elina.gomez@cienciassociales.edu.uy

www.elinagomez.com

Mag. Gustavo Méndez Barbato

gustavo.mendez@cienciassociales.edu.uy



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

Aspectos generales

- Curso presencial (Sala C3)
- Del 28 de junio al 13 de julio. Lunes, miércoles y jueves de 10 a 12.30
- Espacio virtual EVA y [repositorio GitHub](#)
- Requisito un 80% de asistencia, para certificado de asistencia.
- Para aprobación se prevé un Trabajo final (fecha de entrega a definir).

Objetivos del curso

Bases teóricas:

- Contextualizar las **Ciencias sociales computacionales**
- Emergencia de nuevos recursos y técnicas para la investigación social en la era digital.

Objetivos del curso

Generalidades del lenguaje R:

- R como software libre y gratuito
- Comunidades y foros
- Tidyverse
- Manipulación básica de strings

Objetivos del curso

Exploración de fuentes de datos textuales:

- Exploración y obtención de datos de diversa índole, contemplando las diferentes fuentes posibles: OCR , web sacraping, prensa digital, redes sociales, Youtube, APIs.

Objetivos del curso

Análisis textual:

- Codificación manual de textos y creación de redes multinivel (categorías, códigos y citas) mediante la plataforma RQDA().
- Abordaje de los requerimientos previos (limpieza y homogenización) para el análisis de textos.
- Trabajo con minería de textos, el cual se centrará en la noción de *corpus* y sus posibilidades analíticas, desde lo más descriptivo a la aplicación de técnicas más complejas.

Objetivos del curso

Análisis textual:

- Trabajo con diccionarios: Introducción al uso de diccionarios (manuales y automáticos), para la clasificación de documentos masivos según intereses particulares.
- Clasificación de textos: clasificación de textos según temas o emociones asociadas a partir de la aplicación de diferentes técnicas existentes.

Objetivos del curso

Visualización: Exploración de las diferentes posibilidades gráficas de visualización de los resultados del análisis textual (nubes de palabras, frecuencias, dendrogramas, etc.) y algunos ejemplos de visualización interactiva.

Metodología

- El enfoque del curso es práctico (hands-on)
- Trabajaremos con estrategia de live-coding y ejercicios prácticos para cada tema.
- Posibilidad de clonar repositorio GitHub y trabajar con proyecto y control de versiones.
- <https://github.com/elinagomez/analisistextoEPUdelar2023>

Recursos bibliográficos básicos

- Bit by bit (Matthew J. Salganik)
- Data Feminism o Feminismo de Datos (Catherine D'Ignazio y Lauren F. Klein)
- R para Ciencia de Datos (Hadley Wickham y Garrett Grolemund)
- Text Mining with R!(Julia Silge y David Robinson)
- Hojas de ruta en español
- Intro web scraping con R (Riva Quiroga)

Repositorio con recursos varios

Objetivos de hoy

- Introducción conceptual y teórica
- Generalidades de la exploración y manipulación de datos con *tidyverse*.
- Manipulación de strings

Introducción conceptual y teórica

Contexto

- El abordaje metodológico y técnico que veremos en el curso se enmarca en las llamadas **ciencias sociales computacionales**

Introducción conceptual y teórica

“Una nueva disciplina como las Ciencias Sociales Computacionales (Cioffi-Revilla, 2017; Lazer et al., 2009; Mann, 2016) que aglutina la aplicación tanto de las ciencias de la complejidad como del análisis computacional en las Ciencias Sociales, aporta un enfoque innovador para la comprensión del comportamiento humano y social (...) Se promueven así no solo otro tipo de preguntas, sino también la cooperación y la colaboración entre disciplinas. Asimismo, se pueden explorar también nuevas hipótesis a partir de la disponibilidad de nuevos conjuntos de datos y nuevas capacidades de exploración de estos. El desarrollo de nuevos paradigmas y la colaboración entre diferentes campos de conocimiento tienen el potencial de promover nuevos escenarios de investigación, evitando la reiteración en construcciones interpretativas preexistentes (Cioffi-Revilla, 2017; Conte et al., 2012).” (Caro, Jorge et al., 2020)

Introducción conceptual y teórica

Características:

- Perspectiva transdisciplinar
- Dialogo con nuevos desarrollos vinculados a la ciencia de datos, el aprendizaje automático o estadístico, modelado, análisis espacial.
- Explotación del *big data* como fuente de datos y nuevos campos de estudio.

Sobre Big data

“Researchers are in the process of making a change akin to the transition from photography to cinematography.” (Salganik, 2018)

- Bit by Bit: Social Research in the Digital Age (Matthew J. Salganik, 2018). [Reseña en español](#)
- Big Data: datos en línea (búsquedas y rrss); registros administrativos.

Sobre Big data

- Diez características según Salganik (2018): *grande; siempre encendido; no reactivo; incompleto; inaccesible; no representativo; a la deriva; algorithmically confounded; sucio; y sensible.*
- Procesamiento: *“contar cosas; pronosticar cosas; y aproximar experimentos”.*
- Complementariedad con técnicas tradicionales (encuestas): preguntas enriquecidas y pregunta amplificada.
- Nuevas formas de colaboración (crowd-sourcing) y comunidades de desarrollo.
- Desafíos éticos, cajas negras

Panorama regional

- En América Latina existen diversos grupos académicos que trabajan esta línea ya sea desde las llamadas *humanidades digitales* o de las *ciencias sociales computacionales*

Panorama regional

Algunas iniciativas regionales interesantes:

- Programming Historian
- Laboratorio de Humanidades Digitales - UFBA
- Observatorio de conflictividad - UNMdP
- Nucleo de Innovación Social - Colegio de Sociólogos de la Provincia de Buenos Aires
- Instituto Milenio Fundamento de los Datos
- Observatorio de Redes
- Maestría en Humanidades Digitales - Uniandes

R

- ¿Qué es R?
- Consejos
- Generalidades de tidyverse
- Manipulación de strings

¿Qué es R?

- Un lenguaje de programación y un programa estadístico.
- Es **software libre**: se distribuye con licencia [GNU General Public License](#) que implica libertad de uso, modificación y distribución.
- Es **gratuito**, se descarga desde el [CRAN](#).
- Hay una **comunidad** mundial que usa R y lo mejora constantemente, hoy hay más de 10.000 [paquetes](#) disponibles para descargar
- Usuarios se ayudan entre sí: [stackoverflow](#), [talkstats](#), [\(rusers\)](#) y localmente [meetup R-Ladies Montevideo](#).

Consejos

La curva de aprendizaje de R al comienzo suele resultar muy empinada. ¿Cómo podemos evitar o superar la frustración?

- **Usá** R a diario.
- **Traducí** a R una sintaxis sencilla de otro programa que conozcas.
- Recurrí a los **foros** y a la ayuda de R para encontrar las soluciones a los problemas que te surjan: stackoverflow
- Recurrí a otrxs **usuarixs** de R que conozcas.
- Prestá atención a los **mensajes** de error y advertencia.
- **Escribí** tus sintaxis en un script y **comentalas** detalladamente.
- **Reutilizá** sintaxis existentes.

[Hoja de ayuda de R](#)

Carga de datos R

- Carga archivo en formato .RData
- `save(obj1, obj2, obj3, file = "ruta/archivo.Rdata")` *Respalda algunos objetos en formato .RData*
- `save.image("ruta/respaldo.RData")` *Respalda todo el entorno de trabajo*

A diferencia de otras funciones para cargar archivos, la función `load()` es la única en que no se debe asignar a un objeto, ya que para que un archivo sea guardado como Rdata debió haber sido un objeto de R y por ende, al cargar el archivo se carga en la memoria el objeto asociado.

Carga de datos csv o txt

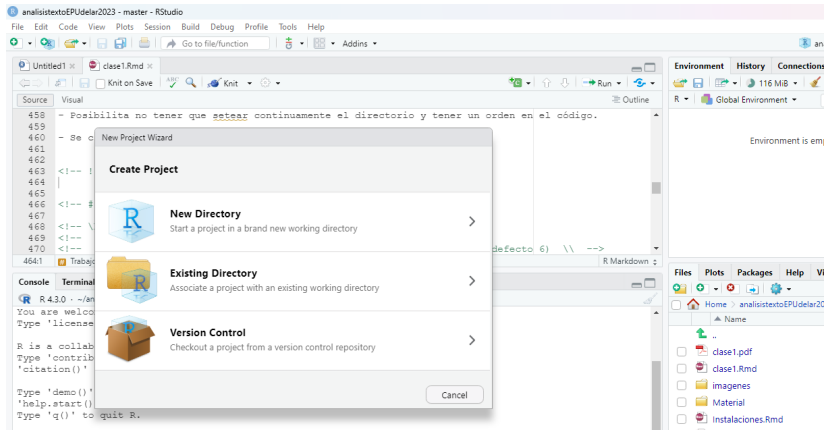
```
datos <- read.table(file="ruta/archivo.txt", header=FALSE,  
  sep=" ")# otros posibles valores de sep= son ",", ";" o "/"  
datos <- read.csv(file="ruta/archivo.csv", header=TRUE,  
  sep=",")  
datos <- read.delim(file="ruta/archivo.txt", header = FALSE,  
  sep = "/")
```

Las rutas deben escribirse con la barra / y no la contrabarra \. Si el archivo está en el directorio de trabajo, alcanza solo con llamar al archivo sin especificar la ruta.

Trabajo con proyectos

- Se recomienda el trabajo por **Proyectos** ya sea a nivel local o conectando con repositorio Git.
- Posibilita no tener que setear continuamente el directorio y tener un orden en el código.
- Se crea desde la barra de herramientas: File > New Project
- Se aloja en un archivo *.Rproj*
- [Tutorial R + GitHub](#)

Trabajo con proyectos



Generalidades de tidyverse

Recomiendo la consulta:

- [Hoja de ruta de dplyr en español](#)
- [Hoja de ruta de ggplot2 en español](#)

Selección por atributos

- *Base R* incluye las funciones de selección `[`, `subset()` y `$`. y **dplyr** aporta `select()`, `slice()`, `filter()`, and `pull()`.

Selección por atributos

- `select()` selecciona columnas por nombre o por posición.
- `pull()` extrae una columna como un vector
- `slice()` es el equivalente de `select()` pero para las filas.
- `filter()` es el equivalente en **dplyr** al `subset()` del *base* de R.
- Otras funciones: `contains()`, `starts_with()` y `num_range()`

Conectando funciones

- Un beneficio de **dplyr** es su compatibilidad con el operador `%>%`.
- Este 'pipe' de R, toma su nombre del pipe de Unix `|`, y es parte del paquete **magrittr**
- Su función es "conectar" la salida de un comando anterior al primer argumento de la siguiente función.
- Esto permite encadenar comandos de análisis de datos, pasando el marco de datos de una función a la siguiente.
- Una ventajas adicional es que fomenta la adición de comentarios a funciones autónomas y permiten líneas simples comentadas sin romper el código.

Agregación de atributos: ejemplo

- Las operaciones de agregación resumen los conjuntos de datos por una 'variable de agrupación' (típicamente una columna de atributo) o un objeto espacial.
- Para calcular el número de personas por continente en base al objeto `world`
- usaremos la columna `pop` que contiene la población por país y la variable de agrupación `continent`.
- En la base R esto se hace con `aggregate()` y `sum()` de la siguiente manera:

Agregación de atributos

```
aggregate(pop ~ continent,  
           FUN = sum,  
           data = world,  
           na.rm = TRUE)
```

Agregación de atributos

`summarize()` es el equivalente en **dplyr** de `aggregate()`, y usa la función `group_by()` para agrupar la variable. Y se implementaría así:

```
group_by(world, continent) %>%  
  summarize(pop = sum(pop, na.rm = TRUE))
```

Agregación de atributos

- Este enfoque es flexible, lo que permite nombrar las columnas resultantes.
- El omitir la variable de agrupación pone todo en un grupo.
- Esto significa que `summarize()` se puede usar para calcular la población total de la Tierra (~ 7 mil millones) y el número de países.
- Utilicemos `sum()` and `n()` para generar las columnas `pop` y `n_countries`

Agregación de atributos

```
world %>%  
  summarize(pop = sum(pop, na.rm = TRUE),  
            n_countries = n())
```

Agregación de atributos

Combinemos todo lo anterior para identificar los 3 continentes más poblados (usando `dplyr::n()`) y el número de países que contienen:

```
world %>%  
  dplyr::select(pop, continente = continent) %>%  
  group_by(continente) %>%  
  summarize(población = sum(pop, na.rm = TRUE),  
            n_paises = n()) %>%  
  slice_max(n = 3, order_by = población)
```

Agregación de atributos

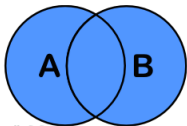
Table 1: Los 3 continentes más poblados, y su número de países.

continente	población	n_paises
Africa	1147005839	51
Asia	4306025131	47
Europe	739178065	39

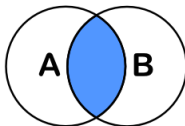
Combinando objetos

- Join combina tablas basadas en una variable compartida ("key")
- **dplyr** presenta varias funciones para ello:
 - `left_join()` - Une las filas coincidentes de b en a
 - `right_join()` - Une las filas coincidentes de a en b
 - `inner_join()` - Une reteniendo solo las filas de ambos conjuntos,
 - `full_join` - Une los datos conservando todos los valores, todas las filas
 - `semi_join()` - Todas las filas en a que tienen una coincidencia en b
 - `anti_join()` - Todas las filas en a que no tienen una coincidencia en b
- Estos nombres de funciones siguen las convenciones utilizadas en el lenguaje de bases de datos [SQL](#).

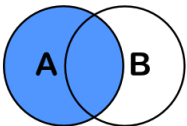
Combinando objetos



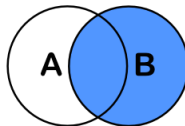
full_join(A,B)



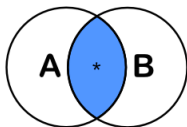
inner_join(A,B)



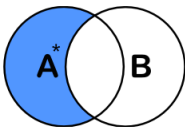
left_join(A,B)



right_join(A,B)



semi_join(A,B)



anti_join(A,B)

Combinando objetos

- Las funciones “..._join” de **dplyr** trabaja tanto con `data.frames`
- El orden de los factores altera el resultado... la clase del primer objeto es la que conserva el resultado.
- Nos centraremos en los `left` e `inner` “..._join” que son los más utilizados, que utilizan la misma sintaxis que los otros tipos de unión.

Combinando objetos

dplyr	base
<code>left_join(x, y)</code>	<code>merge(x, y, all.x=TRUE)</code>
<code>right_join(x, y)</code>	<code>merge(x, y, all.y=TRUE)</code>

Manipulación de *strings*

Antes de realizar un análisis o de construir un modelo de aprendizaje, la discusión de datos es un paso crítico para preparar los datos de texto sin procesar en un formato apropiado.

El texto puede ser considerado como una colección de documentos y un documento puede ser analizado en cadenas. En la limpieza de texto, los patrones de búsqueda se definen en expresiones regulares (abreviadas como `regex()` o `regexp()`) para “encontrar y eliminar” o “buscar y reemplazar” cadenas.

Stringr

- Para realizar estas tareas hay funciones básicas de *R base*. En el curso veremos las funciones que se incluyen en la librería *stringr* de Tidyverse.
- Procesamiento de cadenas
- Hoja de ruta stringr

Stringr

Funciones para separar un texto:

```
library(stringr)

str_split(x, "\r")

str_split(x, boundary("word"))

##type = c("character", "line_break", "sentence", "word")
```

Stringr

Funciones para combinar un texto:

```
library(stringr)

str_c("x", "y", sep = ", ")
> [1] "x, y"
##sep para controlar
```

Stringr

Funciones para reemplazar un texto:

```
library(stringr)
#la primer coincidencia
str_replace(string, pattern, replacement)

#todas las coincidencias
str_replace_all(string, pattern, replacement)
```

Stringr

Funciones para pasar mayúscula/minúscula:

```
library(stringr)

str_to_upper(c("i", "1"))
#> [1] "I" "I"

str_to_lower(c("I", "I"))
#> [1] "i" "i"
```

Stringr

Funciones para pasar eliminar espacios en blanco:

```
library(stringr)

str_trim(string, side = c("both", "left", "right"))

str_trim(" String with trailing and leading white space\t")
#> [1] "String with trailing and leading white space"
str_trim("\n\nString with trailing and leading white space\n\n")
#> [1] "String with trailing and leading white space"
```


Stringr

Funciones para pasar eliminar espacios en blanco:

```
library(stringr)

str_squish(" String with trailing, middle,  and leading white space\t")
#> [1] "String with trailing, middle, and leading white space"
str_squish("\n\nString with excess, trailing and leading white space\n\n")
#> [1] "String with excess, trailing and leading white space"
```

Caracteres especiales

Para construir consultas que incluyan metacaracteres, i.e.

`\\$ * \\+ \\. \\? \\[\\] \\^ \\{ \\} \\\\ \\\(\\\)`

Se debe agregar una retrobarra `\\`

Metacaracteres especiales

`\\t` : Tabulador

`\\n` : Nueva línea

`\\v` : Tabulación vertical

`\\f` : Salto de formulario

`\\r` : Salto de línea

Cuantificadores

Los Cuantificadores especifican cuantas veces el patrón consultado pueda ocurrir.

$*$: coincide al menos 0 veces.

$+$: coincide al menos 1 vez.

$?$: coincide a lo sumo 1 vez.

$\{n\}$: coincide exactamente n veces.

$\{n, \}$: coincide al menos n veces.

$\{, m\}$: coincide a lo sumo m vez.

$\{n, m\}$: coincides entre n y m veces.

Posición

`^` : Al inicio de la cadena.

`$` : Al final de la cadena.

`\b` : En los extremos de la palabra.

`\B` : No en los extremos de la palabra.

`\<` : Al inicio de la palabra.

`\>` : Al final de la palabra.

Caracteres por clases

`[[:digit:]]` or `\\d` or `[0-9]` : dígitos 0 1 2 3 4 5 6 7 8 9 `\\D`
or `[^0-9]` : no-dígitos

`[[:lower:]]` or `[a-z]` : letras minúsculas `[[:upper:]]` or
`[A-Z]` : letras mayúsculas `[[:alpha:]]` or
`[[:lower:]][[:upper:]]` or `[A-z]` : caracteres alfabéticos

Caracteres por clases

`[[:alnum:]]` or `[[:alpha:]][[:digit:]]` or `[A-z0-9]`:
caracteres alfanuméricos

`\\w` or `[[:alnum:]]_` or `[A-z0-9_]` : caracteres de palabra

`\\W` or `[^A-z0-9_]` : No caracteres de palabra

`[[:xdigit:]]` or `[0-9A-Fa-f]` : dígitos hexadecimales (base 16)

0 1 2 3 4 5 6 7 8 9 A B C D E F a b c d e f

Caracteres y espacios

`[[:blank:]]` : espacios y tabulaciones

`[[:space:]]` or `\s` : todo tipo de caracteres de espaciado:
tabulación, nueva línea, tabulación vertical, salto de formulario,
Salto de linea, espacios

`\\S` : Caracteres que no sean espacios

`[[:punct:]]` : caracteres de puntuación ! " # \$ % & () * +
, - . / : ; < = > ? @ [] ^ _ { | } ~

Caracteres y espacios

`[[:graph:]]` or `[[:alnum:]][[:punct:]]` : caracteres gráficos (legibles)

`[[:print:]]` or `[[:alnum:]][[:punct:]]\\s` : caracteres sin espacios

`[[:cntrl:]]` or `\\c` : caracteres de control, como `\\n` or `\\r`, etc.