Notas de clase 2 - 26/05/2022

Tipos de datos

- **Diferencias entre un vector y un dataframe**

  Las principales diferencias que se pueden destacar es que un vector es unidimensional y un dataframe tiene más de una dimensión ya que está formado por filas y columnas. Además se puede decir que un vector es homogéneo ya que acepta solo valores del mismo tipo, en cambio un data frame es heterogéneo ya que acepta valores de diferentes tipos. También es importante destacar que se puede pensar un data frame como una estructura compuesta por vectores de la misma longitud.

  Para conocer más sobre los tipo de datos existentes en R:

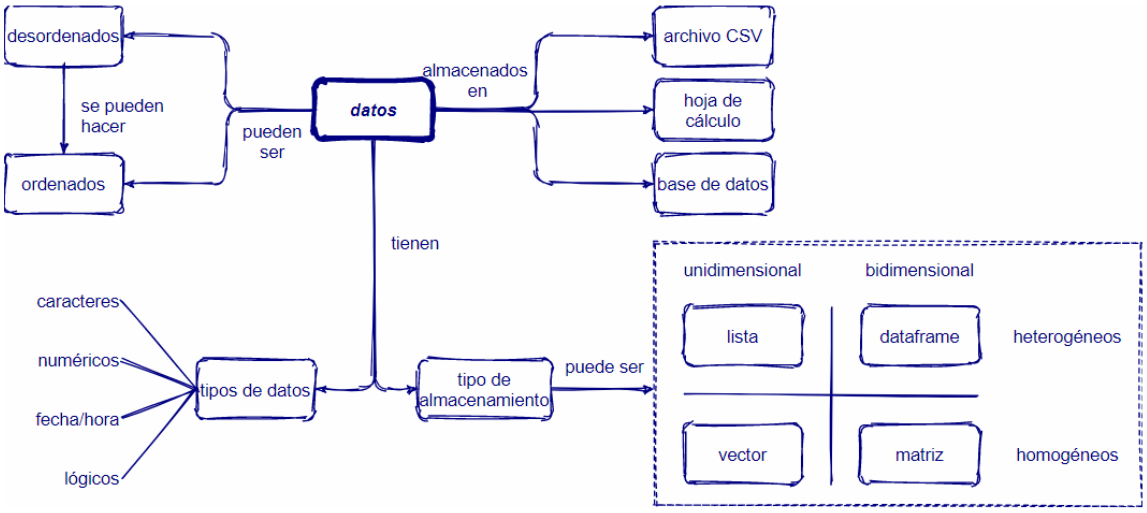


Imagen tomada del repositorio de Greg Wilson: https://github.com/gvwilson

- **Diferencias entre vector y objeto.**
  Podemos decir que en R todo es un objeto. Un vector es un objeto el cual nos permite almacenar datos del mismo tipo pero además del vector existen otros tipos de objetos como listas, variables, entre otros. Hay que tener en cuenta que en el curso generalmente utilizaremos el término objeto para describir cosas almacenadas en R.

Respecto a R base

- **¿Para qué sirve el símbolo # en un script?**
  Sirve para añadir comentarios sin afectar la ejecución de funciones de R dentro de un script.

- **En el comando:** `COSA <- 5*6` **¿Por qué no se usa la función** `c()`**?**
  En este caso R lo que hace es resolver primero la operación que tiene del lado derecho del operador de asignación (`5 * 6` = 30). Luego, esta operación da como resultado un vector numérico de longitud 1 (que tiene un único elemento: `30`). Entonces, una vez realizado el cálculo lo asigna al objeto "COSA". En este caso no necesitamos usar la función `c()` porque no es necesario combinar valores para generar nuestro vector.
  En este sentido, vale la pena aclarar que el orden en el que opera R es:
  - a) primero realiza todas las operaciones del lado izquierdo, si estas son operaciones matemáticas sigue el mismo orden de prioridad que utilizamos habitualmente para las operaciones algebraicas. en nuestro ejemplo, `5 * 6 = 30`
  - b) luego, realiza la asignación del resultado al objeto que le indicamos. Entonces, `COSA <- 30`.

- **¿Es posible borrar los "data" y "values" almacenados en un proyecto? ¿Cómo se borra un objeto?**

  Sí y para borrar un objeto usamos la función *rm(objeto1, objeto2),* donde *objeto* es el objeto que queremos borrar de nuestro ambiente.

- **¿Cómo se edita el nombre de un objeto?**

  El cambio de nombre de un objeto no es posible. Como alternativa lo que podemos hacer es asignar el objeto a un nuevo nombre (y luego borrar nuestro objeto "viejo"). Si el objeto que "queremos cambiar de nombre" es *x* entonces:

  - Asignamos el objeto a un nuevo nombre

    nuevo_nombre <- objeto
  - Borramos el que no vamos a usar

    rm(objeto)

- **Si queremos seleccionar solo algunos elementos de un vector** podemos **anteponer un signo negativo precediendo al vector donde indicaremos los valores que serán excluidos. Por ejemplo,**

  Creamos el vector de personajes:

```
personaje <- c('Marge', 'Homero', 'Lisa', 'Apu', 'Bart')
```

  Y si querríamos seleccionar todos los elementos menos el 2 y el 3 (y por lo tanto nos quedamos con el 1 y el 4), entonces lo haríamos así:

```
personaje <- personaje[-c(2, 3)]
```

  Obtenemos como resultado:

```
[1] "Marge" "Apu"   "Bart"
```

Importación de archivos externos en R

- **Las funciones read.csv y read_csv ¿son lo mismo?**

  La función `read.csv()` es del paquete `utils` (que pertenece a lo que podemos denominar "r base", visto en la primera clase), mientras que `read_csv()` pertenece al paquete `readr` (que es parte de `tidyverse`). Ambas funciones sirven para leer archivos que contienen valores separados por coma. Cada uno tiene características propias, pero la más importante es que read_csv , al igual que la mayoría de las funciones de importación de `tidyverse` en lugar de un objeto de la clase `data.frame`, genera un objeto de la clase `tibble`. Un `tibble` es un tipo de data.frame pero con algunas particularidades que quienes estén interesados pueden leer en este link o en https://es.r4ds.hadley.nz/tibbles.html.

  Las funciones read.csv y read_csv pueden tener diferencias menores en los argumentos que usan. Estas particularidades pueden leerse en los help o ayuda de cada función. Para llamar al help debemos utilizar `?read.csv` o, de manera más general `?nombre_función`.

- **¿Cómo hago para leer archivos separados por comas cuando estos tienen un encoding diferente? ¿Qué hago con los caracteres con acento en bases que no permiten una lectura clara en R?**

  Este no es en realidad un problema de R, sino de cómo hemos importado nuestros datos en R. Los programas tienen diferentes formas de guardar los datos para entender qué dicen estos (de "codificarlos"), cuando los abrimos en un nuevo programa la computadora necesita decodificar cómo fueron guardados para poder mostrarnos los datos que nosotros teníamos originalmente. La diferencia entre estos *encodings* puede hacer que nuestros datos se abran de manera equivocada si el método para leerlos por defecto es diferente al que usan los datos.

  En concreto, en R la mayoría de las funciones para leer datos permiten definir cuál es el encoding que tienen los datos y automáticamente los

transforma al encoding que estamos usando en nuestra sesión de R. Para ello tomemos un ejemplo con `read.csv()`.

Supongamos que queremos abrir un `archivo.csv` que está en nuestro actual directorio de trabajo y que responde a los parámetros definidos por defecto en `read.csv()`. Si nuestro archivo original está codificado en UTF-8 (el formato más recomendable para traspasar información), entonces nuestro código quedaría así:

```
base <- read.csv("archivo.csv", fileEncoding = "UTF-8")
```

En cambio, si usamos la función `readr::read_csv()` del paquete readr (perteneciente a tidyverse), nuestro código quedaría así:

```
library(readr)      # o directamente library(tidyverse)

base <- read_csv("archivo.csv",
                 locale = locale(encoding = "UTF-8"))
```

Más allá de estas adaptaciones una recomendación general es que definan su encoding por defecto en UTF-8. Para ello en RStudio deben ir a la pestaña:

Tools -> Global Options -> Code -> Saving

y allí definir la opción “Default text encoding” en el encoding que prefieran (UTF-8 sugerido).

También puede pasarnos que abramos nuestro script en una computadora que tiene definido otro encoding por defecto y nuestro script se vea con errores (fundamentalmente en acentos y ñ). Para ver correctamente nuestro script, podemos hacer lo siguiente:

- Colocamos el cursor en nuestro script a corregir y vamos a la pestaña:

  File -> Reopen with encoding…

Se nos abrirá un panel para elegir el encoding en el cual está escrito nuestro script. Este se reabrirá y debería leerse correctamente.

- **¿Cómo puedo modificar el "tipo" de mi columna cuando importo?**
  Imaginemos que nuestra base, guardada en `archivo.csv`, tiene una columna que está siendo importada de manera incorrecta. Un caso típico de esto es cuando tenemos un "id" que está compuesto sólo por números y puede suceder que R entienda que se trata de un número (en lugar de entenderlo como una cadena de caracteres numéricos). Esto puede ser un problema si algunos id son precedidos por un 0, porque R los "eliminará" (por ejemplo, si tenemos el número 04, R lo interpreta como 4).
  Supongamos que nuestro problema es como el descripto anteriormente y tenemos pensado utilizar la función `readr::read_csv()` (la función `read_csv()` dentro del paquete `readr` que pertenece a `tidyverse`). Si nuestra columna con problemas es la columna llamada `id` (la cual es de caracteres). El código quedaría de la siguiente manera:

```
base <- read_csv("archivo.csv",
                 col_types = cols(id = col_character()))
```

- **Paquete para abrir archivos cuando no sabemos los separadores**
  Como hemos comentado en clase existen diferentes paquetes para abrir archivos externos y cada uno de ellos tiene su particularidad. Si no sabemos cómo están hechos los separadores de nuestro archivo una opción es el uso del paquete `data.table`, como se muestra a continuación:

```
install.packages("data.table")    #Si no lo tienes instalado

library(data.table)      #Activamos la librería siempre que la vamos a usar
```

```
fread(file = "archivo.csv", header = TRUE)  # utilizamos la función fread para la lectura de documento
```

El argumento *header* se utiliza dentro de la función *fread()* para especificar que debe considerarse la primera línea de nuestro *archivo.csv* como el nombre de las columnas, más información en Introduction to data.table vignette. La librería *data.table* es una buena alternativa para cuando queremos trabajar con grandes volúmenes de datos.

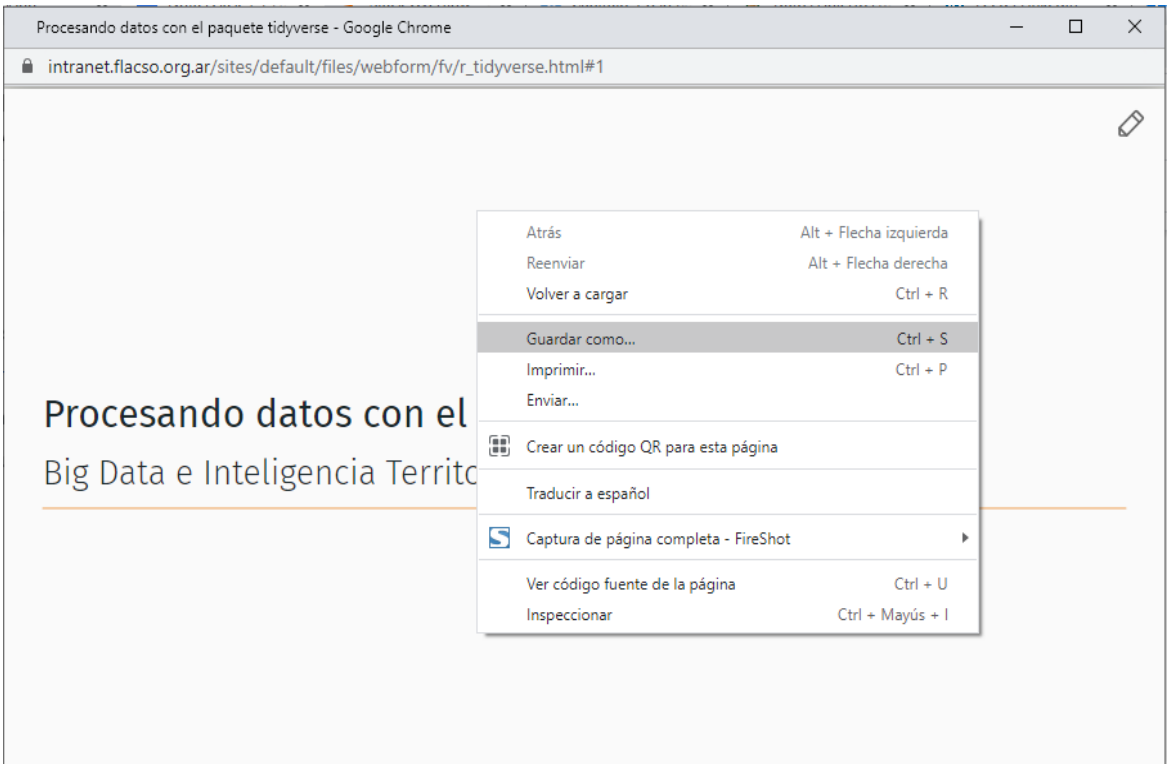
- **Librerías para trabajar con datos provenientes de Stata o SPSS**
  Otros dos paquetes muy interesantes para leer información que proviene de formatos de software "propietario" (como Stata y SPSS) son los paquetes **foreign y haven**. El primero fue diseñado por el equipo principal de desarrollo de R (RCore) y el segundo es parte del universo tidyverse pero cuidado porque no se carga con la meta-librería tidyverse, sino que es necesario cargarlo de forma separada. Para leer sobre ellos: Package foreign y Package haven.

- **Información extra sobre importación de archivos externos**
  Comparación de eficiencia en la carga de datos:
  https://tomaztsql.wordpress.com/2022/05/08/comparing-performances-of-csv-to-rds-parquet-and-feather-data-types/
  parquet:
  https://www.rstudio.com/blog/speed-up-data-analytics-with-parquet-files/
  Comparación data.table vroom:
  https://www.danielecook.com/speeding-up-reading-and-writing-in-r/

- **Cheatsheet o ayudamemoria con las principales funciones de dplyr para tenerlo siempre a mano:** https://github.com/rstudio/cheatsheets/blob/master/data-transformation.pdf

¿Cómo guardar los archivos HTML de la clase?

- Abrir el archivo
- Luego Click + guardar como
- Por último, elegir ubicación para guardar HTML en tu PC.



Atajos

ctrl + f = lleva al search, para buscar y reemplazar.

ctrl + shift + m = %>% (pipe) }.

ctrl + shift + C = comentar scripts selecionados.

Grupos y comunidades de R

- R en Baires: https://renbaires.github.io/ . Actualmente están llevando a cabo el club de lectura del libro Ciencia de Datos para Gente Sociable de Antonio Vázquez Brust.
- R-Ladies Buenos Aires: https://rladies.org/. Es posible consultar el material generado por R-ladies de habla hispana en: https://github.com/rladies/recursos_en_espanol
- Conferencia Latinoamericana sobre Uso de R en Investigación y desarrollo: https://latin-r.com/ .Hasta el 20 de junio es posible enviar propuestas y en la página pueden acceder al slack de la conferencia.
- https://www.facebook.com/groups/rprojectsp
- https://blog.revolutionanalytics.com/local-r-groups.html

Sitios de interés y/o consulta

- Stack Overflow en inglés: https://stackoverflow.com/
- Stack Overflow en español: https://es.stackoverflow.com/
- Comunidad de Rstudio: https://community.rstudio.com/
- Lo que olvidaron enseñarte acerca de R: https://rstats.wtf/