

Redacción de artículos académicos con R

Episodio 1: Programación con R, elementos y fundamentos

Bajaña Alex

Chanatasig Evelyn

Heredia Aracely

2022-06-04



¡Bienvenidos!

Temas generales

Este curso está creado como una guía para llevar a cabo estudios y análisis que califiquen como *investigación reproducible*, esto significa que debemos aprender 4 habilidades:

- **Dominio de R como lenguaje de programación:** Aprenderás los conceptos esenciales de la programación en [R-Base](#). La creación de objetos y las reglas que rigen la interacción entre estos serán conocimientos clave para el desarrollo de nuestra investigación.
- **Limpieza y manejo de tablas estadísticas:** Exploraremos las funciones del paquete [tidyverse](#), el cual colecciona herramientas de lectura, limpieza, transformación, visualización y escritura de tablas para facilitar el análisis reproducible.
- **Edición de documentos académicos:** Ya sea en forma de documento científico o en forma de reporte sobre hallazgos, los procedimientos y resultados de tu análisis puedes documentarlos y compartirlos con [RMarkdown](#).
- **Control de versiones y trabajo colaborativo:** Cada avance diario suma a la consecución del análisis, con [GIT](#) y [Github](#) podemos mantener una bitácora de cada etapa del desarrollo de una investigación. Así mismo, estas herramientas nos permiten colaborar y recibir colaboraciones de una forma fácil y segura.

Todas estas herramientas se manejan de forma integrada en RStudio .

Límites del curso

- Durante el desarrollo del curso emplearemos algunos conceptos de los campos de la estadística y la matemática. Sin embargo no se realizará una discusión ampliada sobre métodos, pruebas o modelos que salgan por fuera del marco de la investigación que se propone como caso de estudio.
- Así mismo cae por fuera de las habilidades que aprenderás en este curso, la optimización de recursos computacionales, se emplearán librerías integradas en el entorno de programación de R y que ya hayan sido publicadas en su repositorio CRAN, las funciones se utilizarán de forma íntegra.

Nota:  Pese a estas limitaciones, se dejan estos dos temas para la discusión e investigación futuras. Si tienes dudas y quieres asistencia con código relacionado a estos u otros temas puedes enviarnos un correo a [ERGOSTATS](#).

Nota:  Por favor, si encuentras errores u observaciones en el material que tienes en mano, te pedimos por favor nos hagas llegar un **issue** a nuestro repositorio de Github [Repositorio del curso](#).

Inspiración de este curso

Hadley Wickham



Libro: R Para data Science

Yihui Xie



Libro: Guía definitiva para Rmarkdown

Principio de reproducibilidad

- ¿Las tablas y figuras son reproducibles a partir del código y los datos?
- ¿El código realmente hace lo que crees que hace?
- Además de lo que se hizo, ¿está claro por qué se hizo? Por ejemplo, ¿cómo se eligieron las configuraciones de los parámetros?

Aún más lejos

- ¿Es posible extender el código a otro conjunto de datos?
- ¿Si me cambio de computador puedo reejectuar el código sin mayores configuraciones salvo por la instalación?

Principio de tidy data

Hay tres reglas que ordenan un conjunto de datos:

- Cada variable debe tener su propia columna.
- Cada observación debe tener su propia fila.
- Cada valor debe tener su propia celda.

country	year	cases	population
Afghanistan	1999	745	1998071
Afghanistan	2000	6666	2059360
Brazil	1999	31737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	21666	128042583

variables

country	year	cases	population
Afghanistan	1999	745	1998071
Afghanistan	2000	6666	2059360
Brazil	1999	31737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	216766	128042583

observations

country	year	cases	population
Afghanistan	1999	745	1998071
Afghanistan	2000	6666	2059360
Brazil	1999	31737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	216766	128042583

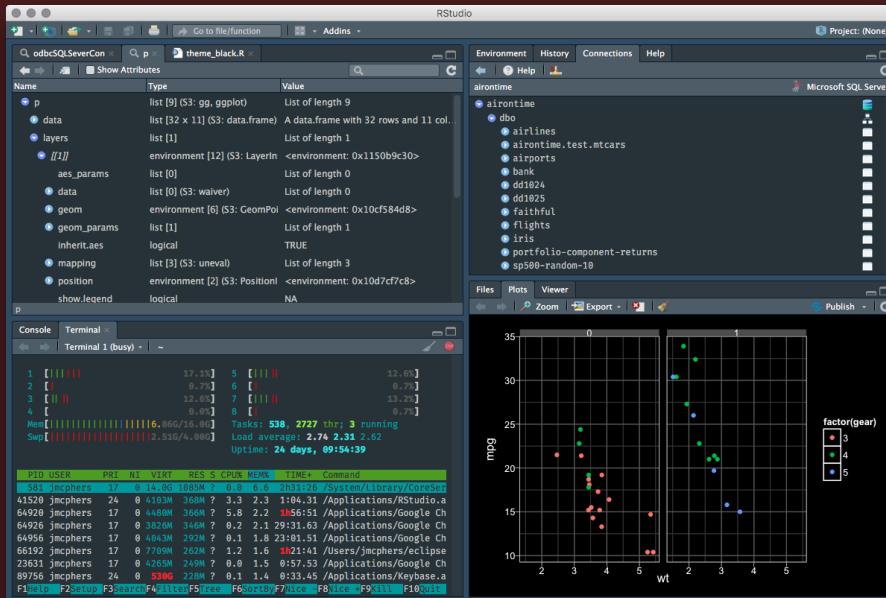
values

A golden retriever is sitting at a desk, looking at a laptop screen. The dog's front paws are resting on the keyboard. The laptop screen shows a blue gradient. In the background, there is a window with blinds, a lamp, and some framed pictures on the wall.

Primeros pasos

Atajos de teclado de RStudio

Si se trabaja en un sistema MacOs se debe presionar la tecla **command** en lugar de la tecla **ctrl**



Comentarios

Cuando escribimos código recordemos que alguien más lo va a leer.

- Para comentar una línea se pone el símbolo numeral `#` antes de lo que se quiera comentar.
- Para comentar un párrafo o una sección completa primero se debe seleccionar el contenido a comentar y a continuación presionar las teclas: `ctrl + shift + C`

```
# Este es un comentario
print(x) # La función print muestra el contenido de 'x'
# print(y) Esta linea no se va a ejecutar
```

- Añadir una nueva sección: `ctrl + shift + R`

```
# Este es el título de mi sección -----
```

Es importante mantener nuestro código ordenado y comentado para cumplir con el principio de reproducibilidad.

Ejecución de código

- Para ejecutar el código línea a línea te debes colocar en la línea y presionar: **ctrl + enter**, otra opción es seleccionar el código que deseas ejecutar y usar la misma combinación de teclas. Ejemplos:

Caso 1:

Más adelante conoceremos el operador **%>%** tambien conocido como **pipe**. Aquí no importa donde hagamos la ejecución.

```
mtcars %>%
  dplyr::filter(cyl>4) %>%
  dplyr::group_by(vs) %>%
  dplyr::tally()
```

RStudio 1.4 tiene la opción de colorear los paréntesis

Caso 2:

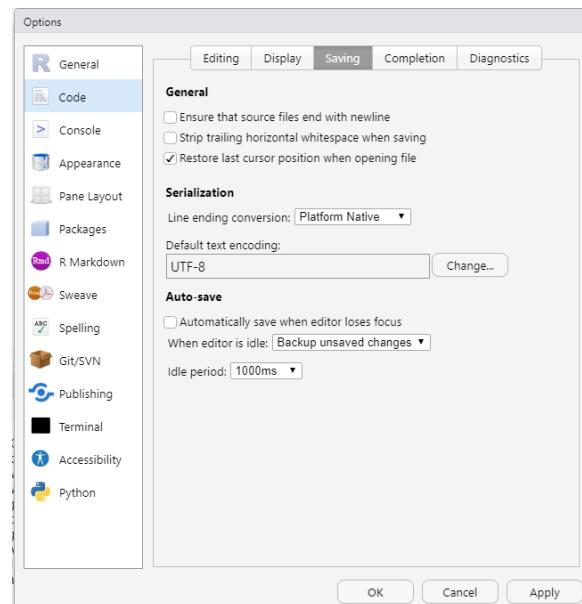
En caso de sentencias largas procura ubicarte al inicio o al fin del código que deseas ejecutar.

```
mi_funcion <- function(argumento){
  x <- operacion(argumento)
  otra_funcion <-
    function(otro_argumento){
      otra_operacion(otro_argumento)
    }
  return(otra_funcion(x))
}
```

Guardar mis archivos

Recuerda verificar el [encoding](#) a la hora de guardar tus archivos. Para verificar que estás empleando [UTF-8](#) (es el encoding que abarca los símbolos del lenguaje español latino) accede al menú [Tools > Global Options](#)

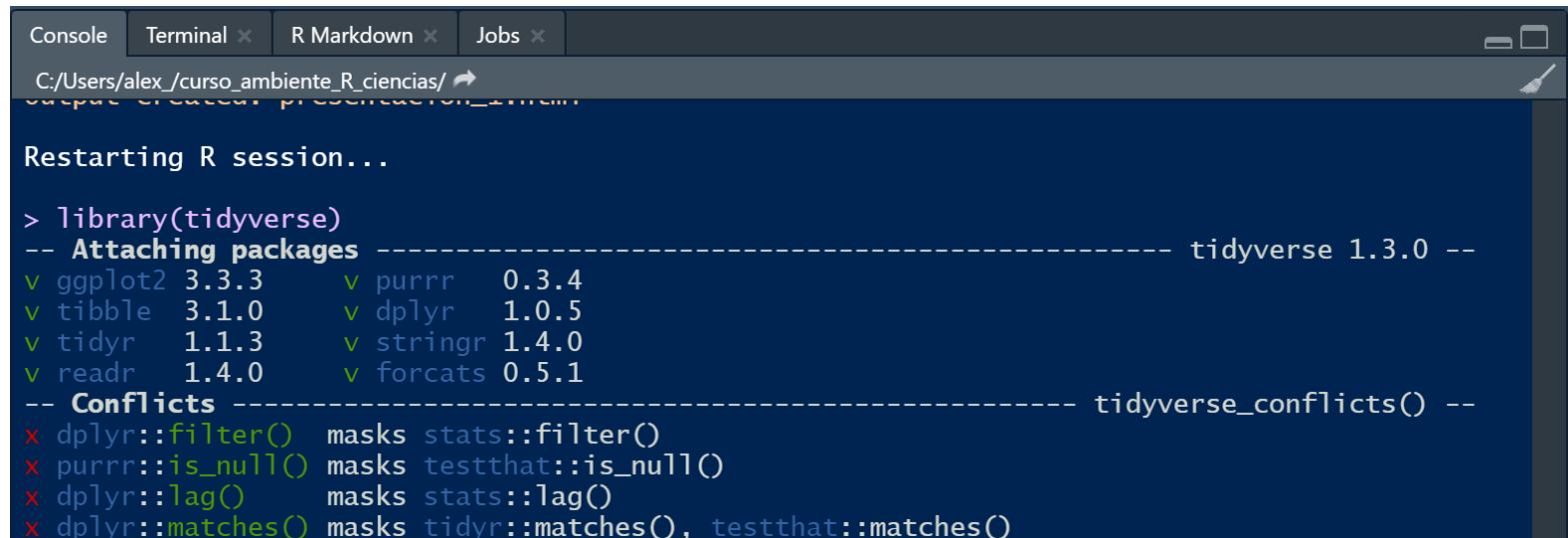
Atajo de teclado para guardar: [ctrl + S](#)



Consideraciones cuando trabajes con R

Mensajes

Cuando un mensaje aparece, R está indicando el correcto funcionamiento de esa función o sentencia. **Ejemplo:** al cargar la librería `tidyverse` el mensaje indica todos los conflictos y detalles que implica usar esta librería en el environment.



The screenshot shows the RStudio interface with the 'Console' tab selected. The console window displays the following output:

```
Console Terminal × R Markdown × Jobs ×
C:/Users/alex/_curso_ambiente_R_ciencias/ ↵
Restarting R session...

> library(tidyverse)
-- Attaching packages -----
  ✓ ggplot2 3.3.3    ✓ purrr   0.3.4
  ✓ tibble   3.1.0    ✓ dplyr    1.0.5
  ✓ tidyverse 1.1.3    ✓ stringr  1.4.0
  ✓ readr    1.4.0    ✓ forcats  0.5.1
-- Conflicts -----
  ✗ dplyr::filter() masks stats::filter()
  ✗ purrr::is_null() masks testthat::is_null()
  ✗ dplyr::lag()    masks stats::lag()
  ✗ dplyr::matches() masks tidyverse::matches(), testthat::matches()
```

En el ejemplo el mensaje está indicando un correcto funcionamiento de la función `library`, empleando como argumento el texto "`tidyverse`".

Errores

```
> "a" + "b"  
Error in "a" + "b" : argumento no-numérico para operador binario
```

En presencia de errores, R tratará de ser lo más informativo en cuanto al error y su posible ubicación en nuestro código. **Un error, no devuelve resultados a menos que se escriba una excepción.**

Warnings

```
> as.numeric("a")  
[1] NA  
Warning message:  
NAs introducidos por coerción
```

R devolverá un resultado pero te dirá qué consideraciones debes tomar acerca de tus resultados.

Fuentes de ayuda

Buscar ayuda en R:

- Abrir una ventana de ayuda:

```
help.start()
```

- Buscar información acerca de un tema específico:

```
help.search("...")
```

- Buscar ayuda sobre una función:

```
help(...)
```

```
?...
```

Recuerda que R es en primera instancia un software de carácter estadístico, los temas de ayuda incrementan a medida que cargues librerías.

Viñetas:

- Buscar viñetas disponibles: `vignete()`
- Buscar una viñeta específica: `vignette("...")`

Web:

- Stackoverflow
- Rpubs
- Mailing List
- Github

¿Cómo buscar ayuda efectivamente?

Si te aparecen mensajes no debes preocuparte pero es diferente si te aparece un error o warning. Para buscar ayuda, corregir el **error** o alterar el comportamiento que dio lugar al **warning** te presentamos los siguientes consejos:

1. Siempre ten presente la versión de R que estás manejando con `R.version`
2. Comprender el error que se presenta en consola.
3. Agotar la lista de recursos *locales* (`help`, `vignette`). La documentación es la fuente primaria de ayuda cuando trabajar con R.
4. Si el problema persiste.
 - Resumir el error de forma concisa y clara. Usualmente la consola ya nos da un texto de esta manera.
 - En el buscador de `google` escribimos la versión de R que estamos usando seguido del problema (resumido). Ej: "R 4.0.4 argumento no-numérico para operador binario"

Consejos:

- Definir bien el objetivo del análisis para ahondar en las herramientas que R nos ofrece.
- Interpretar adecuadamente el comportamiento de R.

Pregunta de investigación:

¿Cuales han sido los avances y retrocesos en el cierre de la brecha salarial por género en Ecuador?

**Caso de estudio empleando la Encuesta Nacional de Empleo, Desempleo y Subempleo (ENEMDU)
2018-2019**

Ejemplos de busquedas de información y ayuda

```
help.search("linear regression")
```

The screenshot shows the R help search interface. At the top, there's a menu bar with 'Files', 'Plots', 'Packages', 'Help', 'Viewer', and a search bar. Below the menu, it says 'R: Search Results' and 'Find in Topic'. The main area is titled 'Code demonstrations:' and lists a single item: 'SparseM::LeastSquares' which is described as 'Least Squares Linear Regression' with a link '(Run demo in console)'. Below this, under 'Help pages:', there are several entries: 'coda::line' (Simple linear regression example), 'datasets::anscombe' (Anscombe's Quartet of 'Identical' Simple Linear Regressions), 'ellipse::ellipse.nls' (Outline an approximate pairwise confidence region), 'ellipse::pairs.profile' (Profile pairs), 'FactoMineR::RegBest' (Select variables in multiple linear regression), 'Hmisc::transace' (Additive Regression and Transformations using ace or avas), 'KernSmooth::dpill' (Select a Bandwidth for Local Linear Regression), 'MASS::area' (Adaptive Numerical Integration), and 'MASS::rms.curv' (Relative Curvature Measures for Non-Linear Regression).

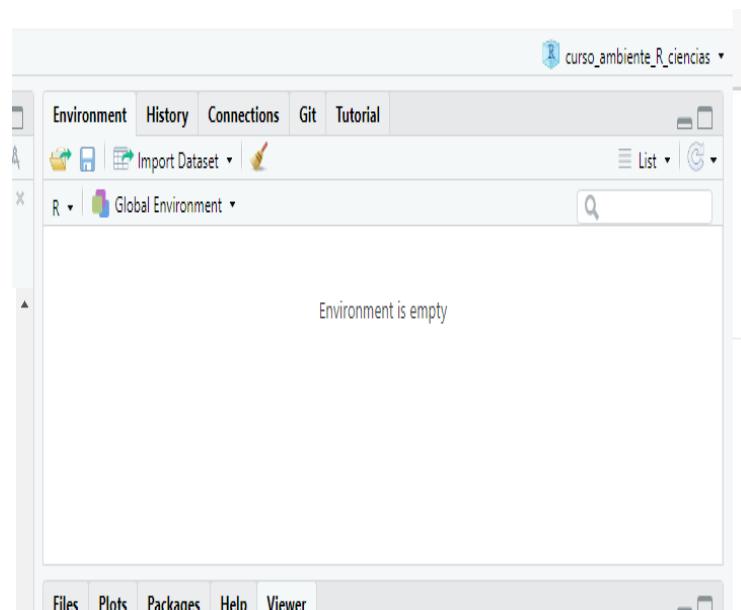
Consideremos el caso en que ya hemos definido el tema de investigación pero nos hallamos con un error. Para mejorar la búsqueda de soluciones:

The screenshot shows a Google search results page for the query 'R 4.04 error 'family' not recognized'. The first result is a link to a Stack Overflow post titled 'GLM function in R with log link not working - Stack Overflow'. The post has 2 responses and was posted on 26 nov. 2012. The second result is a link to another Stack Overflow post titled 'Error in glm() in R' with a timestamp of 7 de jun. de 2017. There are also links to 'Can not install packages on R Version 4.0.0' (5 de may. de 2020) and 'Más resultados de stackoverflow.com'. At the bottom, there's a snippet of text: 'Falta(n): 4:04 | Debe incluir lo siguiente: 4:04'.

Los elementos esenciales de R



Global Environment



Definición:

El ambiente de trabajo o environment almacena todos los vectores, tablas, listas, y demás objetos de R que vayamos creando en nuestros análisis. En ocasiones estos elementos nos serán de utilidad de forma temporal, en otros casos los utilizaremos más de una vez.

Gestión de recursos

Recuerda que R, en tanto lenguaje de programación, es un **lenguaje de programación orientada a objetos** esto quiere decir que a medida que vayamos creando objetos, deberemos considerar las limitaciones de nuestros recursos.

- Para conocer los elementos que están dentro del environment usamos la función: `ls()`.

Limpiar el environment:

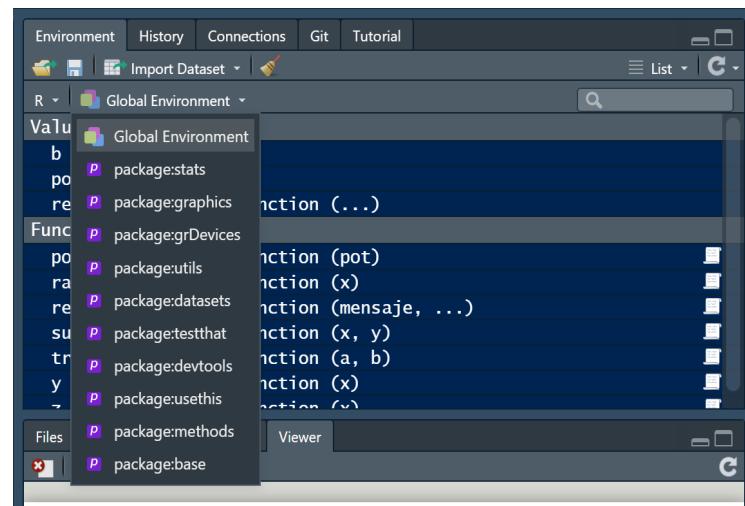
Para borrar elementos del environment:

- 1 solo elemento: `rm("...")`
- Varios elementos `rm(..., "...", "...")`
- Todos los elementos del environment: `rm(list = ls())`

Para borrar los elementos temporales almacenados en *cache* tambien tenemos el comando `gc()` o *garbage collector*.

Librerías o paquetes:

Son colecciones de funciones, documentación (incluidas las [vignette](#)), tablas de datos, y en ocasiones utilidades como [Addins](#) para Rstudio, que permiten llegar a un objetivo establecido. Existen librerías que están preestablecidas desde el momento que abres R, todas ellas bajo un enfoque estadístico.



Nuevas librerías ampliarán el catálogo de funciones y datos que podamos tener a disposición.

Las [vignettes\(\)](#) son de gran utilidad para entender lo que una librería o paquete hace a través de ejemplos y guías documentadas.

Instalar paquetes:

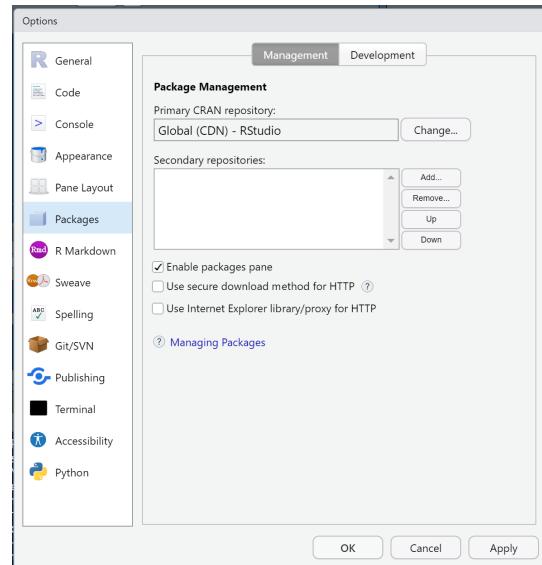
CRAN

Es el catálogo de librerías donde están almacenadas las versiones revisadas por un comité especializado que garantiza la calidad de cada una de las librerías.

The screenshot shows the CRAN homepage with the title "The Comprehensive R Archive Network". It features a large "R" logo and links for "About R", "R Homepage", and "Task Views". The main content area is titled "Download and Install R" and provides links for "Download R for Linux", "Download R for Mac OS X", and "Download R for Windows". Below this, there's information about Linux distributions and source code availability. A sidebar on the right lists various R-related topics like "Sweave", "Spelling", and "Git/SVN".

Algunos ajustes:

Cambiar las opciones en **Tools/Global Options:**



Instalar paquetes:

En el transcurso de este curso haremos uso del paquete `tidyverse`. Esta es una librería especial. Para ser preciso es una *pseudo librería* que al ser invocada añade una serie de librerías para manipulación de datos. Suponiendo que no lo he instalado ya:

- **Instalar un paquete:**

```
install.packages("tidyverse", dependencies = T)
```

- **Ver la versión en desarrollo:**

```
devtools::install_github("https://github.com/tidyverse/dplyr")
```

- **Cargar la librería para poder usarlas:**

```
library(tidyverse)
```

- **Ver librerías instaladas en el equipo:**

```
View(installed.packages())
```

Conflictos entre librerías

En presencia de conflictos se va a utilizar la función del namespace más reciente añadido (attach, en inglés) al **global environment**. Consideremos el caso de una serie temporal:

```
y <- ts(rnorm(100,mean = 20,sd = 12))
```

```
> library("dplyr")
```

```
Attaching package: 'dplyr'
```

```
The following objects are masked from 'package:stats':
```

```
filter, lag
```

```
The following objects are masked from 'package:base':
```

```
intersect, setdiff, setequal, union
```

```
> filter(y)
```

```
Error: Problem with `filter()` input `data`.
```

```
x `data` is a <ts> object, not a data source.
```

```
i Did you want to use `stats::filter()`?
```

```
Run `rlang::last_error()` to see where the error occurred.
```

En este caso el **namespace** más reciente que se agregó fue **dplyr**. En este, la función **filter** se aplica sobre tablas. Para solicitar a R que utilice la función de un determinado namespace utilizamos **::**:

Ejemplo: `stats::filter(y)`



Manos a la obra

Valores atómicos

Son los insumos para construir elementos más complejos. Inicialmente existen 4 elementos básicos:

- **Enteros o integer:** contiene los números naturales. Ej: contar personas

```
class(1L)
```

- **Numéricos o numeric:** contiene números decimales. Ej: estatura o peso

```
class(2.4)
```

- **Carácter o character:** cualquier palabra. Ej: "Hola mundo"

```
class("a")
```

- **Lógico o logical:** Toma valores `TRUE` y `FALSE` que se traducen en `1` y `0` respectivamente. Ej: para saber si está empleado o no

```
class(TRUE)
```

Vectores

Es la colección de varios atómicos de la misma clase. Los vectores almacenan elementos únicamente del mismo tipo.

- R emplea el comando `c(...)` para juntar elementos de un mismo tipo en un vector:

```
c(35, 42, 21, 18)
```

- En R se utiliza el símbolo `<-` para asignar un vector (de lado derecho del símbolo) a un nombre (del lado izquierdo del símbolo). A este símbolo se lo conoce como **asignación**:

```
edades <- c(35, 42, 21, 18)
```

- Los vectores que se crean tras la asignación van a guardarse en el **Global Environment**.

Los vectores tienen propiedades:

- **Clase de un vector:** se refiere al tipo de valor atómico que tiene, como ya se vio existen cuatro clases: entero, numérico, carácter y lógico.

```
class(...)
```

- **Dimensión o tamaño:** muestra el número de elementos que tiene un vector.

```
length(...)
```

Para conocer todos los atributos de un vector o elemento de R podemos usar la función `attributes(...)` sin embargo, **los atómicos no tienen atributos**. Por ello la función devuelve el valor `NULL`

Reglas de coherción:

A pesar de que los vectores almacenan elementos del mismo tipo, se puede unir o concatenar vectores de diferente tipo. Cuando esto ocurre, el vector final será de determinada clase de acuerdo a las siguientes reglas de coherción:

1. Si se juntan vectores de tipo `character` con cualquier otro, el vector resultante será `character`
2. Si se juntan un vector con enteros `integer` y uno de tipo numérico `numeric` el vector resultante es `numeric`
3. Si se concatenan un vector de `tipo logical` con un `integer` o `numeric` el tipo del vector resultante lo definirá este segundo vector

R aplicará estas reglas de forma automática cuando concatenamos vectores de distintos tipos.

Nombres sintácticos:

- El **nombre que le demos al vector** no debe contener espacios, en lugar de ello se pueden usar los símbolos: _ o . Por ejemplo: `nombre_del_vector` o `nombre.del.vector`. Así también se recomienda **NO** usar tildes.
- En caso de que se quiera crear un nombre de vector no sintáctico (con espacios o símbolos), este debe seguir la siguiente sintaxis: `....`
- Se recomienda no usar como nombres de vectores, aquellas palabras que usa **R Studio** para sus comandos como: `TRUE`, `FALSE`, `name`, entre otros, a estos se los conoce como nombres o palabras reservadas.

Ejemplos:

Nombres no sintácticos:

- `vector?edades`
- `tabla parroquia Quito`

Nombres sintácticos:

- `vector_edades`
- `tabla_parroquia_Quito`



Recursos

- Advanced R Programming
- Rmarkdown definitive guide
- R for data Science
- <https://spcanelon.github.io/xaringan-basics-and-beyond/index.html>