

Curso Primeros Pasos en R

Clase 2: Tipos y estructura de datos en R

Profesora: Ana María Alvarado

Pontificia Universidad Católica de Chile

Noviembre 2021

Clase 2: Tipos y estructura de datos en R

- Funciones y objetos
- Tipos de datos
- Estructuras de datos
- Vectores
- Matrices
- Data frame
- Tibble
- Operadores lógicos
- Taller práctico

Funciones y objetos en R

Lógica de Funcionamiento en R

En lo relacionado con el análisis de datos, la lógica de R es bastante amigable



Los ... corresponden a los argumentos que modifican la acción a realizar, dependiendo de la función esta puede o no tener argumentos y estos pueden ser tanto obligatorios como opcionales.

Funciones en R

Las funciones son operaciones que están guardadas con un nombre específico en R.

Hay funciones que requieren **argumentos** (inputs) y otras no. Los argumentos de una función van siempre dentro del paréntesis:

```
sqrt(521)  
getwd()  
Sys.Date()
```

Al escribir y ejecutar solo el nombre de la función podemos consultar el código almacenado en ella, por ejemplo:

```
sqrt
```

Para ayuda y detalles de la función, anteponemos **?** o usamos la función **help()**:

```
?sqrt  
help(sqrt)
```

Actividad práctica

1.- ¿Qué argumentos recibe la función `log()`?

Respuesta: Podemos revisar la ayuda de la función:

```
help(log)
```

O bien utilizar el comando `args(log)`:

```
args(log)
```

```
## function (x, base = exp(1))
## NULL
```

La función `log` tiene como argumentos `x` (el valor a calcular el logaritmo) y `base` (la base del logaritmo, por defecto la base es `e`).

Actividad práctica

2.- ¿Cuál es la diferencia entre `log(100, 10)` y `log(10, 100)`

Respuesta En el primero calculamos el logaritmo de 100 en base 10, en el segundo calculamos el logaritmo de 10 en base 100.

```
log(100, 10)
```

```
## [1] 2
```

```
log(10, 100)
```

```
## [1] 0.5
```

Actividad práctica

3.- ¿Cuál es el error en el siguiente código?:

```
log(base = 10)
```

Respuesta:

El error es que no estamos agregando el argumento `x`, en esta función sólo el argumento `x` es obligatorio.

Objetos en R

Un **objeto** es una colección de información indexada, bajo un nombre previamente definido diferenciando mayúsculas de minúsculas.

Para asignar o crear un objeto, utilizaremos el operador de asignación: `<-`.

```
objeto <- valor
```

También podemos usar `=` ó `->`

```
objeto = valor  
valor -> objeto
```

Se recomienda usar `<-`, es fácil de reconocer visualmente y genera menos confusión que el signo `=`. El operador `->` es poco usual y nos genera desorden en nuestro código.

Para generar el operador de asignación puedes usar:

alt + -

Objetos en R

- La función `ls()` muestra todos los objetos creados.
- La función `rm(objeto)` elimina cierto objeto, mientras que `rm(list=ls())` elimina toda la memoria.
- R recordará todos los objetos creados, hasta el momento que cierras la consola, o bien, hasta que elimines la información.

Actividad práctica

1.- Defina un objeto llamado 'numero' asignele un valor numerico:

```
numero <- 256
```

Los nombres nos permiten representar resultados de una manera más sencilla y usarlos así en nuestro código, incluso podemos aplicarles funciones:

2.- Defina a la raíz cuadrada de número como 'resultado':

```
resultado <- sqrt(numero)
```

3.- Imprima en la consola el valor guardado en 'resultado':

```
resultado
```

```
## [1] 16
```

Objetos en R

Si el nombre de un objeto ya existe en nuestra sesión, R lo va a sobreescribir. Por ejemplo:

```
a <- 5  
b <- a  
a <- 3
```

¿Cuál es el valor de **a**?

```
a
```

```
## [1] 3
```

¿Cuál es el valor de **b**?

```
b
```

```
## [1] 5
```

Recomendaciones

1. Utilizar un nombre que tenga alguna relación con los datos que contiene el objeto
2. Evitar caracteres especiales, como ñ, tildes o espacios
3. Para separar palabras se puede utilizar un guión bajo (proyeccion_enero) o mayúscula inicial (ProyeccionEnero). Lo importante es ser consistente en la opción elegida.

¡Recuerda que R es sensible a mayúsculas y minúsculas!

¿Qué pasa al ejecutar el siguiente código?

```
polera <- c(254, 203, 182, 50)  
mean(Polera)  
sum(poleras)
```

Errores

La mayoría de los errores que cometemos son por problemas de tipeo:

- Escribimos mal el nombre de una función u objeto.
- Nos falta cerrar un paréntesis.
- Nos falta una coma.

En caso de que falte un paréntesis o una coma, el editor de RStudio nos lo advertirá

A veces se generan problemas porque olvidamos correr una línea de código o porque sobreescribimos un objeto. En esos casos, lo mejor es reiniciar R y volver a ejecutar el código desde el principio.

¡Recuerda que equivocarse es normal!

Buscadores como Google ayudan mucho a resolver errores, saber como buscar soluciones a tus dificultades de manera eficiente es una herramienta importante para cualquier lenguaje de programación.

Tipos de datos

Tipos de datos

En R existen distintas clases y tipos de datos. Estos dependen del tipo de información que contengan en su interior. Con las funciones `class()` y `typeof()` se puede preguntar qué tipo de dato es cada elemento.

Los principales tipos de datos en R son los siguientes:

Tipo de dato	Descripción	Ejemplo
<code>integer</code>	Números enteros	-1, 0, 1
<code>numeric</code>	Números reales	-0.5, 1/2, 1
<code>character</code>	Texto	"palabra", "y"
<code>logical</code>	Verdadero o Falso	'TRUE' 'FALSE'

Hay otros tipos de datos en R, como `complex` (números complejos) y `raw` (datos sin procesar).

Estructuras de datos

Estructuras de datos en R

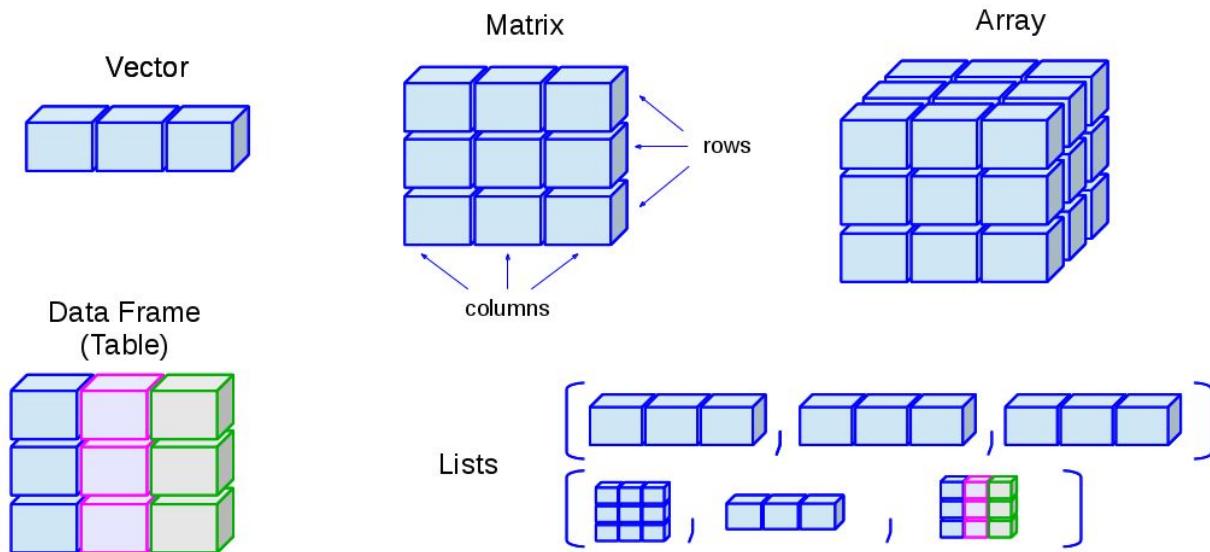
En R los distintos tipos de datos definidos anteriormente se organizan en estructuras.

Las estructuras de datos son **objetos** que contienen datos.

Estructura	Instrucción en R
Vector	<code>c()</code>
Matriz	<code>matrix()</code>
Array	<code>array()</code>
Data Frame	<code>data.frame()</code>
Lista	<code>list()</code>
Factor	<code>factor()</code>
date	<code>as.Date()</code>
Time Series	<code>ts()</code>

Cuando trabajamos con R, lo que estamos haciendo es manipular estas estructuras.

Estructuras de datos en R



Vectores

Vectores

Un vector es un ordenamiento unidimensional de datos. Dentro de un vector solo puede existir un tipo de dato. En R, los vectores se definen con el comando `c()` y cada elemento se separa con una coma:

Vector numérico

```
vector_1 <- c(-1/2, 1, 0.5)
class(vector_1)

## [1] "numeric"
```

Vector con letras

```
vector_2 <- c("A", "B", "C")
class(vector_2)

## [1] "character"
```

¿Qué tipo de dato será el vector `c(1, "dos", 3)`?

Funciones para generar vectores

En la siguiente tabla, se muestran algunas de las principales funciones usadas para facilitar la creación de vectores en R:

Función	Descripción	Ejemplo
a:b	Genera una secuencia de números naturales entre a y b	5:10
seq()	Genera una secuencia de números a un intervalo regular	seq(from= -5, to=5, by= 0.5)
rep()	Genera una secuencia de repetición de elementos de un vector base	rep(c("a", "b"), times=3)
letters	Vector constante con las letras desde la "a" a la "z" en minúscula	letters
LETTERS	Vector constante con las letras desde la "A" a la "Z" en mayúscula	LETTERS

Funcionalidades adicionales

Con el comando `length()` se puede extraer el largo de un vector.

Se puede consultar en R si un elemento es o no un vector con el comando `is.vector()`

```
is.vector( c(1, 2, 3) )
```

```
## [1] TRUE
```

De la misma forma, se puede transformar un objeto a un vector con el comando `as.vector()`.

Seleccionar un elemento dentro de un vector

Para seleccionar el elemento de un vector en R se utilizan los corchetes `[]`, de la siguiente forma:

`objeto[i]` extrae el i-ésimo elemento del vector llamado *objeto*

Matrices

Matrices

Una matriz (`matrix`) es un arreglo de datos en una estructura bi-dimensional, entendidas como filas y columnas. En R, las matrices funcionan como una extensión de los vectores, dado que dentro de una matriz solo puede existir un tipo de dato. Hay muchas formas de generar matrices, la manera más directa es usando el comando `matrix()`:

```
matrix(vector, ncol = n, n_row = m, byrow = FALSE )
```

```
matrix(1:9, ncol = 3)
```

```
##      [,1] [,2] [,3]
## [1,]     1     4     7
## [2,]     2     5     8
## [3,]     3     6     9
```

```
matrix(letters[1:9], ncol = 3)
```

```
##      [,1] [,2] [,3]
## [1,] "a"  "d"  "g"
## [2,] "b"  "e"  "h"
## [3,] "c"  "f"  "i"
```

Concatenación de vectores

Se puede transformar un conjunto de vectores en una matriz con ayuda de los comandos `cbind()` y `rbind()`. El primero, junta vectores de manera vertical (columna), y el segundo de manera horizontal (fila):

```
v_1 <- 1:5  
v_2 <- LETTERS[1:5]
```

Concatenación por columnas (c)

```
cbind(v_1, v_2)
```

```
##      v_1 v_2  
## [1,] "1" "A"  
## [2,] "2" "B"  
## [3,] "3" "C"  
## [4,] "4" "D"  
## [5,] "5" "E"
```

Concatenación por filas (r)

```
rbind(v_1, v_2)
```

```
## [,1] [,2] [,3] [,4] [,5]  
## v_1 "1" "2" "3" "4" "5"  
## v_2 "A" "B" "C" "D" "E"
```

Funcionalidades adicionales

Seleccionar elementos de una matriz

En R, los arreglos bi-dimensionales funcionan siempre con el orden (filas, columnas). Utilizando esta lógica, es posible rescatar elementos desde una matriz:

`nombre_matriz[i, j]` ~ i=filas, j=columnas

Ejemplos:

- `nombre_matriz[i,]`: Selecciona la i-ésima **fila** de la matriz.
- `nombre_matriz[, j]`: Selecciona la j-ésima **columna** de la matriz.
- `nombre_matriz[i, j]`: Selecciona el j-ésimo elemento de la i-ésima fila.

El comando `dim()` permite obtener las dimensiones de la matriz. El primer número corresponde al número de filas, el segundo al número de columnas.

Se puede consultar si un objeto es una matriz con el comando `is.matrix()` y se puede transformar un objeto a una matriz con el comando `as.matrix()`

Data frame

Data frame

Un data frame es un arreglo de datos en una estructura bi-dimensional. A diferencia de una matriz, un data frame puede tener columnas de distintos tipos de datos. En un data frame, las columnas representan variables y las filas representan a individuos u observaciones.

	Variable 1	Variable 2	...	Variable n
Observación 1	dato	dato	dato	dato
Observación 2	dato	dato	dato	dato
...
Observación m	dato	dato	dato	dato

Los data frame en R se pueden generar directamente con el comando `data.frame()` o se pueden transformar otras estructuras definidas. Por ejemplo, una matriz, con el comando `as.data.frame()`

Creación de un data frame

El comando `data.frame()` se puede usar de distintas formas:

Forma 1: Usar vectores ya creados. En este, todos los vectores deben ser del mismo largo. El nombre de cada vector se guardará como el nombre de cada columna del data frame.

```
var_1 <- c(a1, a1, ..., an)
...
var_m <- c(z1, z2, ..., zm)

base_ejemplo <- data.frame(var_1, var_2, ... , var_n)
```

Forma 2: Definir directamente las columnas dentro del dataframe

```
base_ejemplo <- data.frame(var_1 = c(a1, a1, ..., an),
                           ...,
                           var_m = c(z1, z2, ..., zm))
```

Actividad práctica

Considere la siguiente tabla:

Nombre	Grupo Sanguíneo	Altura (cm)
Andrea	AB	165
Eduardo	O	180
Camilo	A	158
Daniela	B	162

Defina esta tabla como un data frame de usando las dos formas anteriores.

Actividad práctica

Forma 1

1.- Se definen las columnas de las matrices en vectores:

```
nombre <- c("Andrea", "Bastian", "Camilo", "Daniela")
grupo_s <- c("AB", "0", "A", "B")
altura_cm <- c(165, 180, 158, 170)
```

2.- Se guarda en el objeto 'tabla' el data frame:

```
tabla <- data.frame(nombre, grupo_s, altura_cm)
tabla
```

```
##      nombre grupo_s altura_cm
## 1    Andrea      AB       165
## 2   Bastian        0       180
## 3   Camilo        A       158
## 4 Daniela        B       170
```

```
names(tabla)
```

```
## [1] "nombre"     "grupo_s"     "altura_cm"
```

Actividad práctica

Forma 2

Se definen directamente los vectores dentro del comando `data.frame()`:

```
tabla <- data.frame(  
  nombre = c("Andrea", "Bastian", "Camilo", "Daniela"),  
  grupo_s = c("AB", "0", "A", "B"),  
  altura_cm = c(165, 180, 158, 170)  
)
```

```
tabla
```

```
##     nombre grupo_s altura_cm  
## 1  Andrea      AB      165  
## 2 Bastian        0      180  
## 3  Camilo       A      158  
## 4 Daniela       B      170
```

```
names(tabla)
```

```
## [1] "nombre"      "grupo_s"      "altura_cm"
```

Selección de observaciones de un data frame

Un data frame funciona igual que una matriz, ya que se puede seleccionar directamente filas y columnas específicas con el uso de corchetes `[]`.

En los data frame, es normal que las variables (columnas) tengan nombre. En R, se pueden usar tales nombres para seleccionar elementos de la base de datos:

Seleccionar variables dado nombre de variables

```
nombre_base$nombre_variable  
nombre_base[["nombre_variable"]]
```

Seleccionar observaciones de un data frame

- `base[i,]`: Todas las columnas, i-ésima observación (fila).
- `base[i, j]`: i-ésima fila, j-ésima variable (columna).
- `base[v_filas, v_columnas]`: todas las filas que están en `v_filas` y columnas en `v_columnas`.

Volvemos en ...

10:00

Tibble

Tibble

Los tibbles son data frames, pero tienen características que hacen su manejo más fácil y más acorde con la actualidad. La librería `tibble` pertenece a la librería `tidyverse`, que se verá en profundidad en las próximas clases. Se instalará inmediatamente:

```
install.packages("tidyverse")
library(tidyverse)
```

Para transformar un data frame o una matriz a un objeto `tibble`, basta con utilizar el comando `as_tibble()` de la siguiente forma:

```
base_tibble <- as_tibble(base_dataframe)
```

El comando `tbl_df()` de la librería `dplyr`, librería presente en `tidyverse()`, permite transformar directamente un data frame en un tibble.

Dependiendo de cómo se importan las bases de datos externas, es probable que las bases de datos ya estén con formato `tibble`.

Beneficios de usar tibble

Impresión en la consola

En comparación a un data frame, los tibble se imprimen de una manera mucho más limpia en la consola, representando un resumen claro de los datos:

```
cars
```

```
##   speed dist
## 1     4    2
## 2     4   10
## 3     7    4
## 4     7   22
## 5     8   16
## 6     9   10
## 7    10   18
## 8    10   26
## 9    10   34
## 10   11   17
## 11   11   28
## 12   12   14
## 13   12   20
## 14   12   24
## 15   12   28
## 16   13   26
## 17   13   34
## 18   13   34
## 19   13   46
```

```
cars_tibble <- as_tibble(cars)
cars_tibble
```

```
## # A tibble: 50 x 2
##       speed   dist
##       <dbl> <dbl>
## 1       4      2
## 2       4     10
## 3       7      4
## 4       7     22
## 5       8     16
## 6       9     10
## 7      10     18
## 8      10     26
## 9      10     34
## 10     11     17
## # ... with 40 more rows
```

Operadores lógicos

Operadores lógicos

En R hay operadores lógicos. Estos permiten realizar preguntas a R, las que serán respondidas como verdadero (`TRUE`) o falso (`FALSE`). Algunos operadores básicos son:

Operador	Código
Menor que	<
Mayor que	>
Menor o igual que	<=
Mayor o igual que	>=
Igual que	==
Distinto que	!=
Intersección	&
Unión	
Dentro de	%in%

```
10 > 5
```

```
## [1] TRUE
```

```
"Gato" == "gato"
```

```
## [1] FALSE
```

```
c("a", "B", "c") %in% letters
```

```
## [1] TRUE FALSE TRUE
```

Usos de operadores lógicos

Los operadores tienen múltiples usos. Principalmente, se usan para generar condiciones en la creación de funciones y filtrar bases de datos y/o vectores. Utilizando el ejemplo anterior:

```
tabla
```

```
##     nombre grupo_s altura_cm
## 1  Andrea      AB      165
## 2 Bastian       0      180
## 3 Camilo        A      158
## 4 Daniela       B      170
```

Suponga que se quiere seleccionar aquellas filas en donde la altura sea menor a 170 cm. Esto, se puede realizar de la siguiente forma:

```
tabla[tabla$altura_cm < 170, ]
```

```
##     nombre grupo_s altura_cm
## 1  Andrea      AB      165
## 3 Camilo        A      158
```

Taller práctico

Taller práctico

El paquete **NHSRdatasets** contiene datos gratuitos del Servicio Nacional de Salud del Reino Unido (NHS) y otros datos relacionados con la salud de la población con fines educativos y de formación.

El conjunto de datos **ae_attendances** contiene asistencias reportadas, infracciones de 4 horas y admisiones para todos los departamentos de A&E en Inglaterra durante los años 2016/17 hasta 2018/19 (abril-marzo).

A continuación realice los siguientes ejercicios.

1. Instale y active la librería NHSRdatasets.
2. Revise la descripción del conjunto de datos **stranded_data** usando la función de ayuda.
3. Asigne a un objeto llamado **datos** el conjunto de datos **stranded_data**.

Taller práctico

Mediante selección de elementos de dataframe y operadores logicos, indique:

- a. Toda la información del cuarto registro.
- b. La edad del paciente del registro 20.
- c. La edad de los pacientes del registro 20 al 30.
- d. La edad e indice de fragilidad de los pacientes del registro 20 al 30.
- e. Toda la información de los pacientes con 80 años o más.
- f. Construya un nuevo data frame que contenga todos los pacientes que requieren apoyo y atención de salud mental.

¡Gracias!

Ana María Alvarado Celis
amalvara@uc.cl

Esteban Rucán
errucan@uc.cl