**2020/01/24**

**INTRODUCCIÓN A R**

**Fátima Sánchez Cabo**

Bioinformática Data Science. Matemática. Análisis de datos genómicos.

Dr Fuster, dirige el CNIC. Fátima dirige la unidad de bioinformática.

R nació en el año 1998. Era un lenguaje matricial y desde el 2017 se está intentando mover a un leguaje más orientado a objetos.

Una de las cosas buenas de R es que tiene muchísimas funciones o paquetes desarrollados.

GapMinder. Recoger indicadores del mundo para desmitificar cosas.

.rmd = R markdown -> a partir de él puedo genera pdf, html, y más. Este formato puede intercalar texto y código, como los note books.

Los scripts planos (sin texto libre) son formato .r

*Hands-on programming with R*

*Data Science with R*

Son libros buenos de consulta, gratis en versión electrónica. Tienen un github asociado donde podemos descargar el cógido y ejemplos usados.

vignette() es la documentación de los paquetes…

Es un lenguaje interpretado: no necesita compilación

No está hecho por informáticos.

El código fuente está en R, en Fortran y C. Al estar en R podemos descargar, meternos dentro de cualquier función, entender cómo está hecha y modificarlo si queremos.

Puntos fuertes:

- Hace buenos gráficos. Aunque es verdad que ahora python con matplotlib hace gráficos muy buenos.

- Gran comunidad detrás desarrollando los paquetes.

- Es intereactivo. No hace falta compilarlo para ejecutarlo, se puede ejecutar según se escribe.

Características:

- La funciones de R son *objetos* de R. Tienen su propio *tipo.*

- La distribución básica de R trae 8 paquetes: stats, graphics, grDevices, utils, datasets, methods, base.

CRAN: Comprehensive R Archive Network. Aquí es donde se suben todos los paquetes. Todos podemos desarrollar un paquete y subirlo ahí aunque tienen unos mínimos requerimientos (documentación, normas de escritura del código).

Dentro de CRAN hay un subconjunto de paquetes llamado Bioconductor.

También hay paquetes, no supervisados, fuera de CRAN que cualquiera puede publicar en su web o su github.

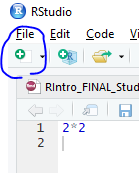
Abrimos Rstudio desde Anaconda de windows. También se podría instalar Rstudio para windows independiente de anaconda.

Lo primero es decirle a R dónde estamos trabajando, osea nuestro directoio base donde tenemos el archivo de trabajo. En Session > Set Working Directory > choose directory

Se escribe automáticamente una línea de código así:

setwd("C:/Users/usuario/AppData/Local/Packages/CanonicalGroupLimited.UbuntuonWindows\_79rhkp1fndgsc/LocalState/rootfs/home/angelrps/git/MasterDataScience/200124\_R\_Import\_Collection\_Of\_Data")

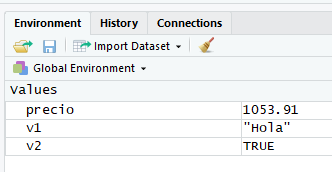
Podemos abrir un nuevo R Script aquí. Un Script es un simple texto.



para ejecutar una linea del R Script en la consola de abajo pulsamos ctrl + intro

En un rmd (R markdown) donde se mezcla texto y código se ejecutan los chunks de código ahí mismo.

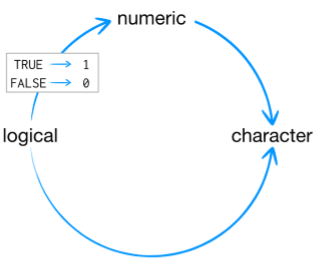
En Environment irán apareciendo todos los objetos (variables) que iremos creando en la sesión.

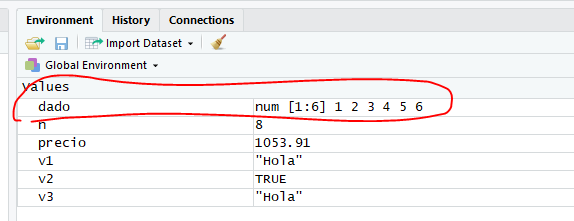


Los objetos se van guardando en memoria RAM.

Se pueden asignar variable con <- o con =. Se suele usar <-

tipos de objetos:

* Vector (1\*n) atómico de tamaño n= 1: es el tipo más básico que tenemos en R. Puede ser de tipo numérico (double or int), de tipo carácter (string), de tipo lógico (true or false).
* Vector de tipo numérico y tamaño 6.
* Matriz (m\*n): vector de 2 dimensiones sólo. Todos los elementos **DEBEN SER DEL MISMO TIPO**. Y si no lo son R hace transformaciones para que lo sean. Los numéricos los transforma en texto y los lógicos los transforma en numéricos. Según este ciclo.
* 
* Array (m\*n\*n\*n\*n\*….): igual que la matriz pero con n dimensiones.
* data.frame: como una matriz que permite tener **ELEMENTOS DE DIFERENTE TIPO.**
* list.
* tibble: una mejora del data frame.



> dado <- c(1,2,3,4,5,6)

> dado <- 1:6

Los dos de arriba crean la misma variable. A veces te pone num, otras, int porque no maneja bien los tipos. Si queremos que entienda forzamente los números como int tenemos que crear así la variable

Para crear un vector siempre se usa c(). La C viene de *concatenar.*

> dado <- c(1L,2L,3L,4L,5L,6L)

> length(dado)

para interrogar el tipo de objeto

> is.numeric(dado)

[1] TRUE

> class(dado)

[1] "integer"

> typeof(Dado)

[1] "integer"

operaciones elemento a elemento

> dado\*2

[1] 2 4 6 8 10 12

> dado/2

[1] 0.5 1.0 1.5 2.0 2.5 3.0

> dado+2

[1] 3 4 5 6 7 8

> dado/dado

[1] 1 1 1 1 1 1

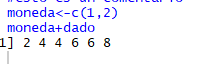
operación matricial

> dado%\*%dado

[,1]

[1,] 91

Cuidado! Si sumas dos vectores de diferente rango R te los suma añadiendo el vector menor tantas veces como sea necesario para llegar al final

Es algo muy raro, CUIDADO!!

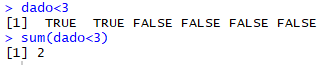
**La precisión numérica de R son 16 dígitos.** Ojo por si a veces hay problemas de precisión de cálculos.

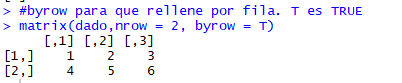
NOTA: en la vida real cuando trabajemos en R es mejor trabajar en Rmd en lugar de un R script.

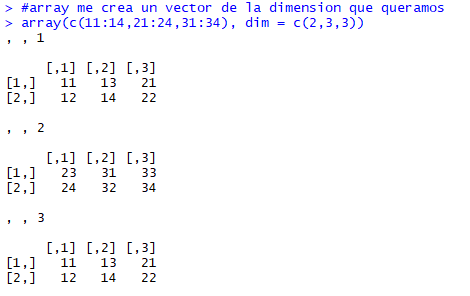


mejor no poner espacios porque a veces R da problemas añadiendo un punto. Así que ponemos barra baja.

Se pueden sumar vectores lógicos. TRUE sería 1 y FALSE 0





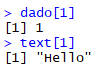


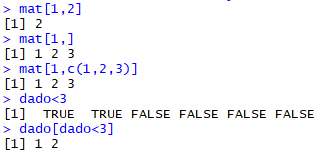
**cbind**(vector1, vector2, vector3,…) para pegarlos columna a columna

**rbind**(vector1, vector2, vector3,…) para pegarlos fila a fila

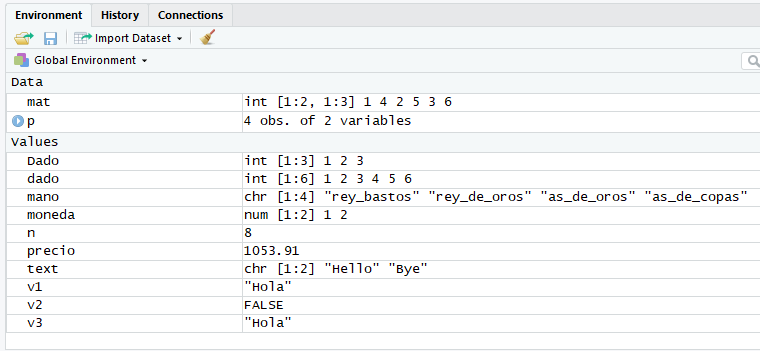
**cómo acceder a los elementos de los diferentes tipos de objetos?**

**indexando con corchetes y OJO, los indices comienzan por 1, no por 0**





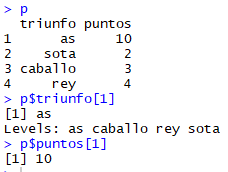
NOTA SOBRE TIPOS

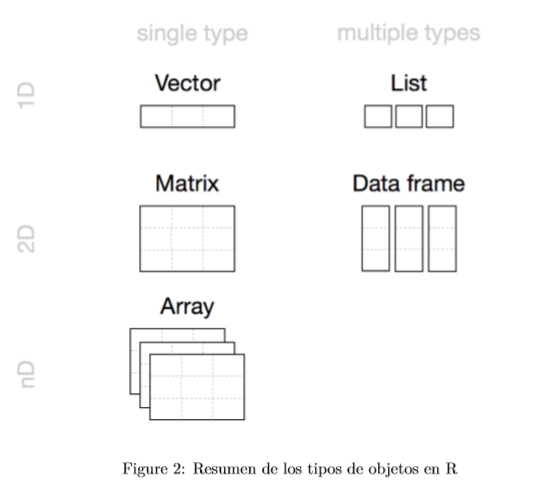


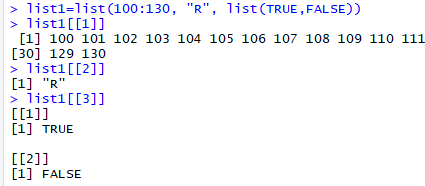
los vectores de una dimensión, ya seaen num, int, char, o lógicos, se agrupan en *values.* Las matrices y los data frames se agrupan en *Data.*

factor es otro tipo de dato. Sirven para definir categorías. Ojo, R por detrás los almacena numéricamente osea que si los quiero ordenar creo que no lo haría alfabéticamente.

Cómo indexar como data frames. Nombrando direntamente el nombre de las variables con $ delante.







funciones. La sintaxis es

functionname <- function(){

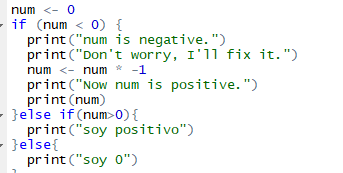
return

}

.Rdata es un *objeto* que guarda el workspace que incluye las variables que he usado. Normalmente no lo guardamos.

**2020/01/25**

Condiciones



con source(‘nombredelscript.r) se ejecuta un archivo que hayamos guardado con extensión .r

para instalar paquetes nuevos:

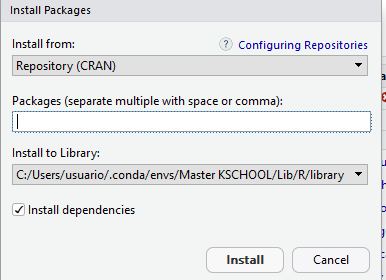
si están en CRAN uso: install.packages(nombre del paquete)

o varios a la vez metiendo un vector conlos nombres de los paquetes:

install.packages(c(nombre1, nombre2, nombre3, …))

si están en GitHub uso: install\_github()

también se puede hacer desde los menús: Tools > Install Packages



Install simplemente lo descarga al ordenador per

Una vez se han instalado los paquetes hay que hacerlos disponibles con la función library()

sessioninfo() nos da información sobre la sesion incluido sobre los paquetes que hay disponibles