Procesamiento de datos con R

Clase: Introducción a R

Orlando Joaqui Barandica

Universidad del Valle

2023

Contenido

1 Introducción

2 Elementos en R

3 Objetos en R

4 Indexado

5 Valores perdidos

6 Importar datos en R

Contenido

1 Introducción

2 Elementos en R

3 Objetos en R

4 Indexado

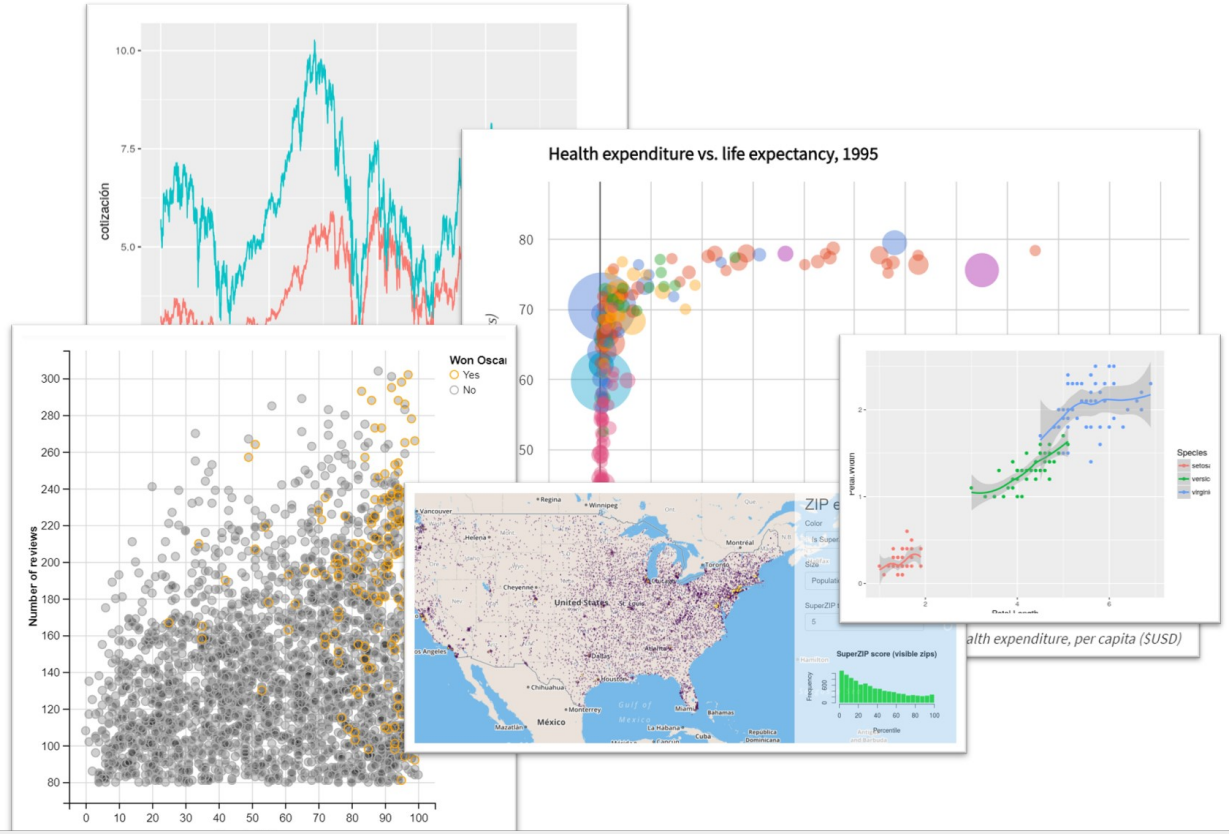
5 Valores perdidos

6 Importar datos en R

Introducción

Qué hay detrás de las estadísticas... De las predicciones... De las representaciones... en las que se basan los mejores científicos, analistas, estadísticos, emprendedores, etc.. a la hora de tomar decisiones.

Quizás no sea muy visible pero está ahí. Se llama R.



Qué es R?

- R es un conjunto integrado de programas para manipulación de datos, cálculo y gráficos.
- El cual fue creado por Ross Ihaka y Robert Gentleman (1996), del Departamento de Estadística de la Universidad de Auckland (Nueva Zelanda).
- Se considera un lenguaje estadístico-matemático que se encuentra en pleno crecimiento..
  https://jjallaire.shinyapps.io/shiny-crandash/
- Publicado como software libre con licencia GNU-GPL. Garantiza a los usuarios finales la libertad de usar, estudiar, compartir (copiar) y modificar el software.

Atractivos

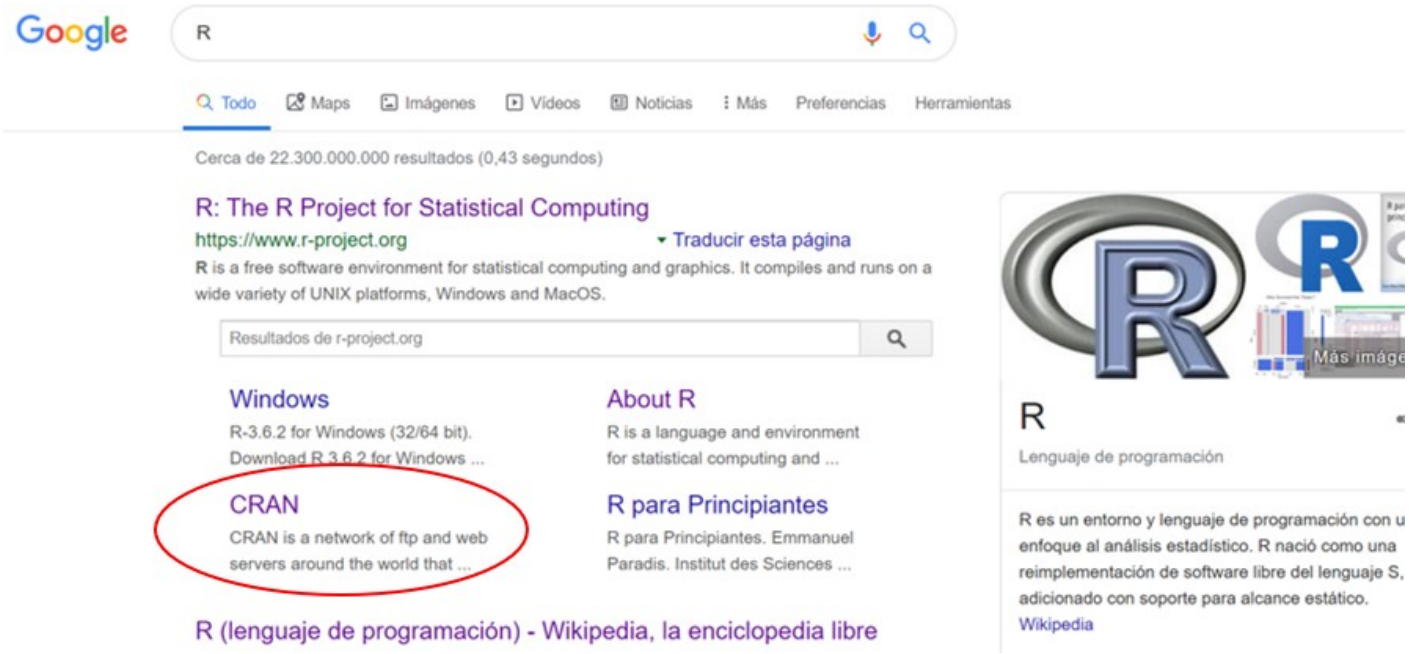
- Gran Flexibilidad para combinar diferentes análisis estadísticos, específicos para una situación. Capacidad de manipular y modificar datos, procedimientos y funciones.
- La Comunidad R es muy dinámica (Crecimiento constante de paquetes), integrada por estadísticos de gran renombre. Más de 15 mil paquetes. http://cran.es.r-project.org/web/packages/
- Gráficos de alta calidad (revelaciones de la visualización de datos y producción de gráficas, altamente compatibles con software de edición científicos).
- Rápida implementación de avances metodológicos en estadística.

Atractivos

- Libertad de estudiar el funcionamiento del programa y adaptarlo a sus necesidades (Acceso al código fuente).
- Libertad de distribuir copias para ayudar a los demás sin ánimo de lucro.
- Libertad de mejorar el programa y de publicar las mejoras, de modo que toda la comunidad se beneficie (Acceso al Código Fuente)
- Existe un respaldo en recurso de la información en la web.
  - https://www.r-bloggers.com/
  - https://stackoverflow.com/questions/tagged/r
  - Redes sociales: Facebook, Twitter, ..
  - ... la nube en general...

Iniciemos en R

Su búsqueda es sencilla..



O entrando directamente a la página principal de R.
https://cran.r-project.org/

Iniciemos en R

Elegir el sistema operativo y seguir los pasos de instalación..

CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed

The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

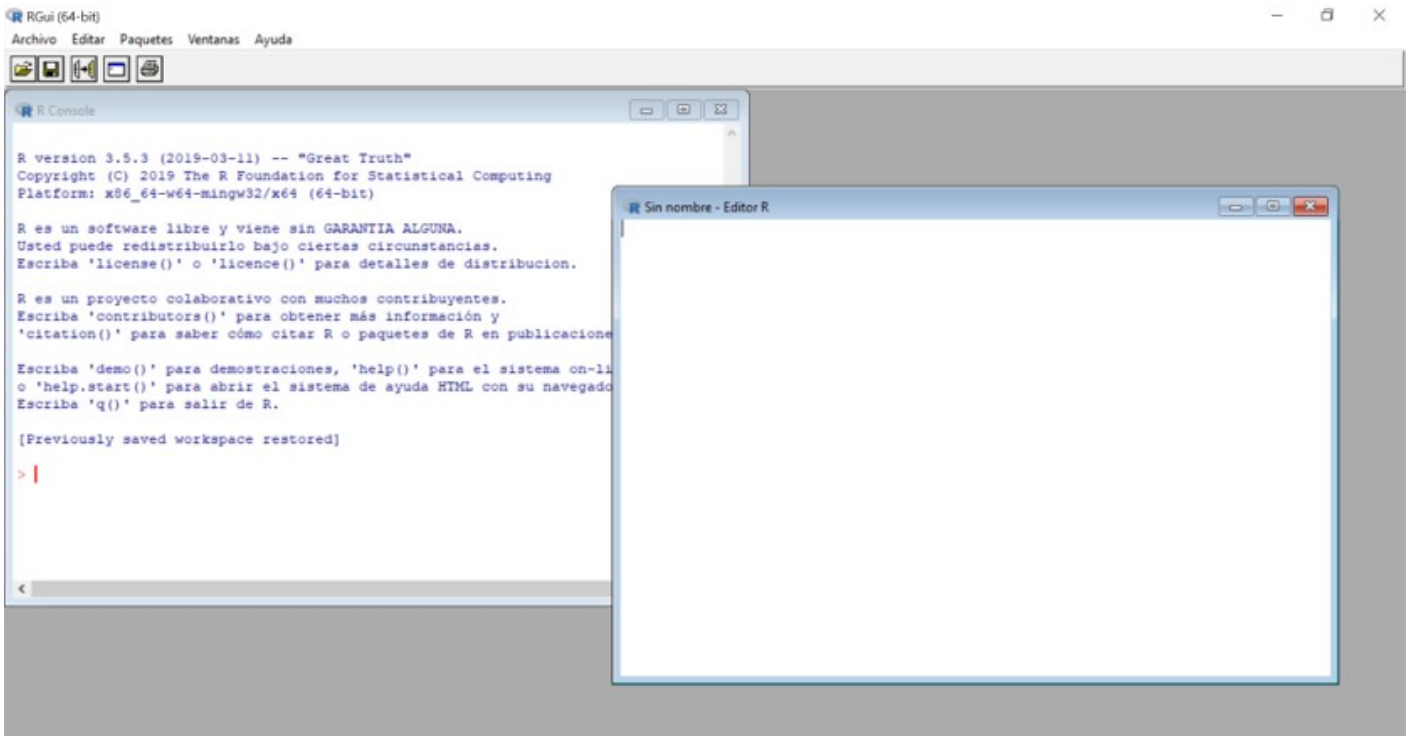
Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2019-12-12, Dark and Stormy Night) R-3.6.2.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.
- Contributed extension packages

Questions About R

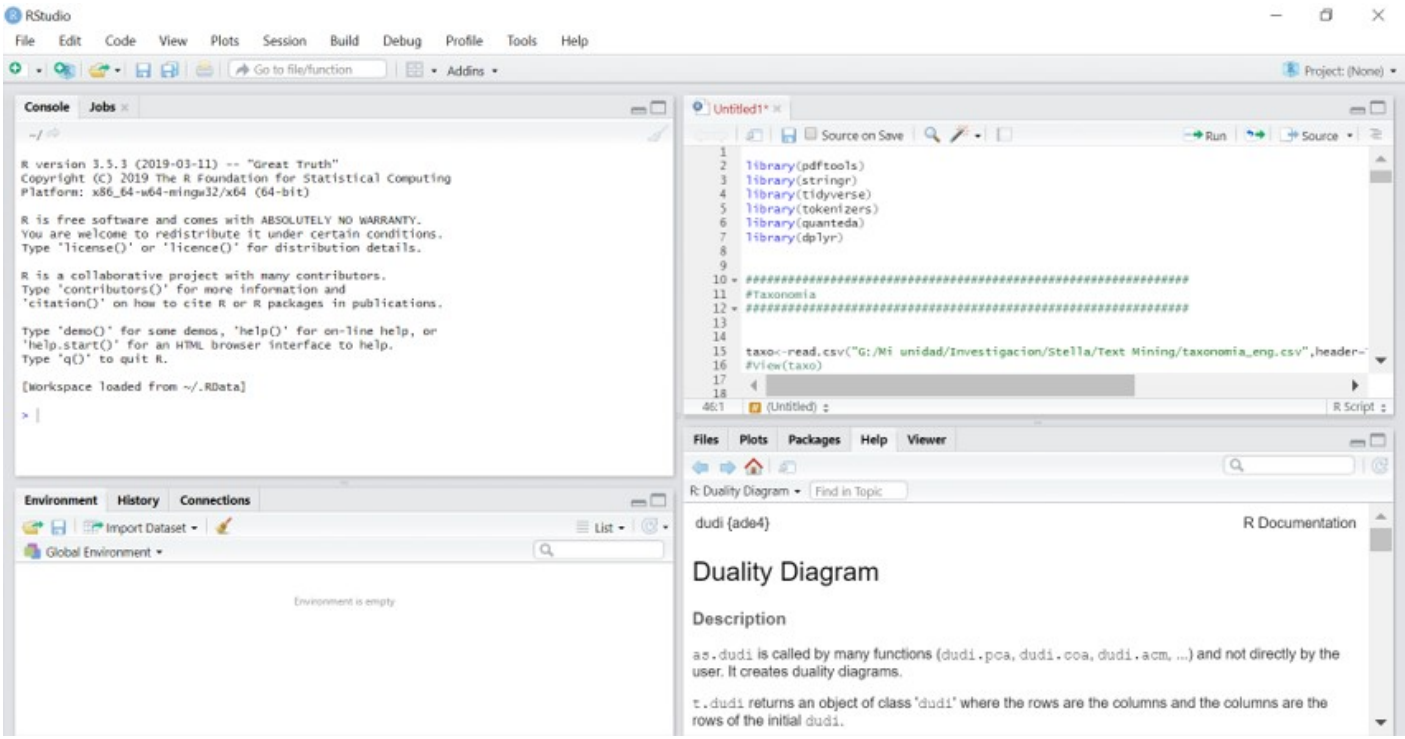
Iniciemos en R

R



Iniciemos en R

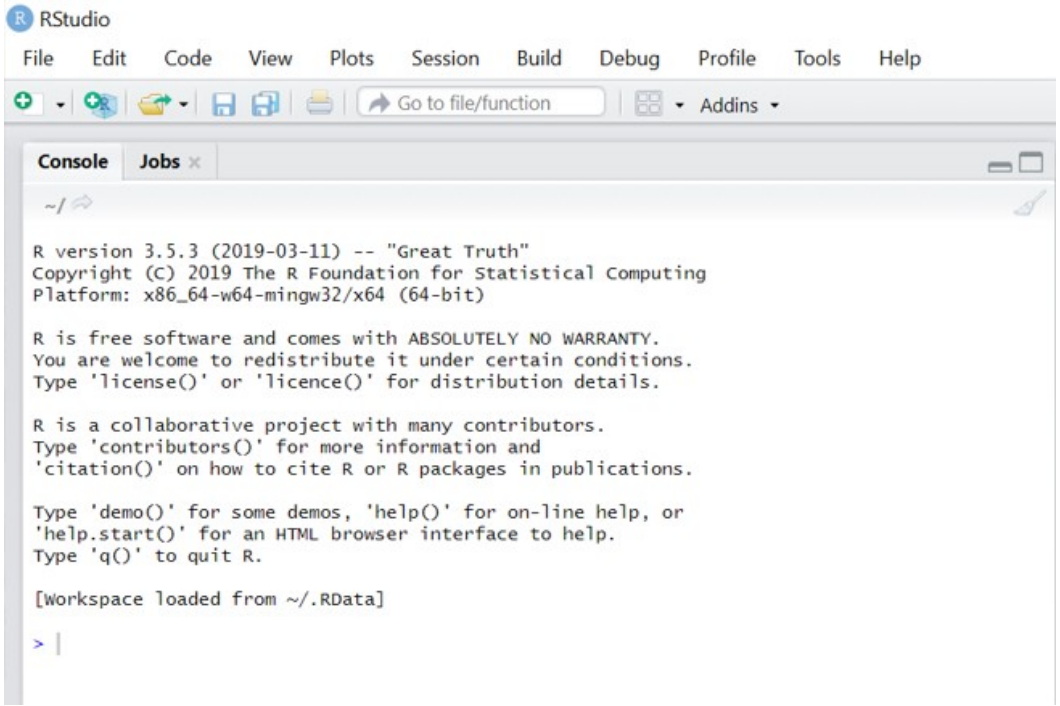
R-Studio



Descargar R-Studio. https://rstudio.com/

Iniciemos en R

Consola



El *prompt*:

- $>$ *"Listo para recibir la siguiente orden"*
- $+$ *"Hay un proceso en ejecución"*

Iniciemos en R

Reglas sintácticas

- R evalúa expresiones
- El lenguaje es sensible a mayúsculas
- Pueden utilizarse espacios entre elementos de sintaxis a discreción: *sin(x+b)* es igual que *sin (x + b)*
- Cada expresión se escribe en al menos una línea
- Dos o más expresiones puede utilizar una línea separándolas por el signo ';'
- En R, donde entra un valor puede entrar una expresión
- Los comentarios se realizan con '#'

Iniciemos en R

Matemáticas	Expresión en R
$x = 3$	`x <- 3`
$sin\,\alpha$	`sin( alpha )`
$log_{10}(x)$	`log( x, 10 )`
v_i	`v[ i ]`
$\sum_{i=1}^{n} v_i$	`sum( v )`

Trabajar con R

1. **Limpiar pantalla:** Ctrl + L
   La consola de salida de R trabaja de forma acumulativa, cada nuevo resultado se agrega al cumulo anterior.

2. **Verificar objetos existentes:** ls(), objects()
   Es posible que al dar inicio a una sesión sean cargados objetos previos...es necesario verificar.

3. **Remover objetos:** rm(objeto), rm(list=ls())
   El almacenamiento de objetos innecesarios genera un mayor consumo de memoria y un riesgo de cruce de objetos.

4. **Asignar directorio:** setwd(dir)
   setwd(Ç:/Users/Mi carpeta/Desktop/Curso R")

   El directorio base es el lugar donde R lee los archivos de entrada y ubica las salidas dirigidas por defecto.

Contenido

1 Introducción

2 Elementos en R

3 Objetos en R

4 Indexado

5 Valores perdidos

6 Importar datos en R

Elementos en R

Tipos de valores en R

- Enteros: 7
- Reales: 2.5e+11 ($2.5\ 10^{11}$)
  Importante! Los decimales se especifican con . y no con ,
- Caracter: "papá"
- Perdidos: NA
- No números: NaN (log("papá"))
- Indeterminaciones $(-\infty, \infty)$: - Inf, Inf

Operadores aritméticos

^	potencia
* /	producto, cociente
+ -	suma, resta
%/%	cociente entero
%%	módulo
:	generar una serie
%*%	producto matricial
()	paréntesis

Ejemplos

```
3 ^ 2
```

Ejemplos

```
3 ^ 2
```

```
[1] 9
```

Ejemplos

```
3 ^ 2
```

```
[1] 9
```

```
3 ^ 1 + 1
```

Ejemplos

```
3 ^ 2
```

```
[1] 9
```

```
3 ^ 1 + 1
```

```
[1] 4
```

Ejemplos

```
3 ^ 2
```

```
[1] 9
```

```
3 ^ 1 + 1
```

```
[1] 4
```

```
3 ^ (1 + 1)
```

Ejemplos

```
3 ^ 2
```

```
[1] 9
```

```
3 ^ 1 + 1
```

```
[1] 4
```

```
3 ^ (1 + 1)
```

```
[1] 9
```

Ejemplos

```
10 / 2 * 5
```

Ejemplos

```
10 / 2 * 5
```

```
[1] 25
```

Ejemplos

```
10 / 2 * 5
```

```
[1] 25
```

```
10 / 2 / 5
```

Ejemplos

```
10 / 2 * 5
```

```
[1] 25
```

```
10 / 2 / 5
```

```
[1] 1
```

Ejemplos

```
10 / 2 * 5
```

```
[1] 25
```

```
10 / 2 / 5
```

```
[1] 1
```

```
21 %% 5
```

Ejemplos

```
10 / 2 * 5
```

```
[1] 25
```

```
10 / 2 / 5
```

```
[1] 1
```

```
21 %% 5
```

```
[1] 1
```

Ejemplos

```
1:10
```

Ejemplos

```
1:10
```

```
[1]  1  2  3  4  5  6  7  8  9 10
```

Ejemplos

```
1:10
```

```
[1]  1  2  3  4  5  6  7  8  9 10
```

```
1:10 * 2
```

Ejemplos

```
1:10
```

```
[1]  1  2  3  4  5  6  7  8  9 10
```

```
1:10 * 2
```

```
[1]  2  4  6  8 10 12 14 16 18 20
```

Ejemplos

```
1:10
```

```
[1]  1  2  3  4  5  6  7  8  9 10
```

```
1:10 * 2
```

```
[1]  2  4  6  8 10 12 14 16 18 20
```

```
2^(0:8)
```

Ejemplos

```
1:10
```

```
[1]  1  2  3  4  5  6  7  8  9 10
```

```
1:10 * 2
```

```
[1]  2  4  6  8 10 12 14 16 18 20
```

```
2^(0:8)
```

```
[1]   1   2   4   8  16  32  64 128 256
```

Operadores lógicos

!	no
== !=	igual, distinto
> >=	mayor, mayor o igual
< <=	menor, menor o igual
| ||	o
& &&	y
#	comentario

Ejemplos

```
3 >= 2
```

Ejemplos

```
3 >= 2
```

```
[1] TRUE
```

Ejemplos

```
3 >= 2
```

```
[1] TRUE
```

```
0 != 0.000000000000000000000001
```

Ejemplos

```
3 >= 2
```

```
[1] TRUE
```

```
0 != 0.00000000000000000000001
```

```
[1] TRUE
```

Ejemplos

```
3 >= 2
```

```
[1] TRUE
```

```
0 != 0.00000000000000000000001
```

```
[1] TRUE
```

```
5*2 > 9 & 3/2 == 1.5
```

Ejemplos

```
3 >= 2
```

```
[1] TRUE
```

```
0 != 0.00000000000000000000001
```

```
[1] TRUE
```

```
5*2 > 9 & 3/2 == 1.5
```

```
[1] TRUE
```

Asignaciones

```
Variable <- expresión
```

Variable es un nombre que se utiliza como representación del resultado de una expresión.

<-	Asignar a la izquierda
->	Asignar a la derecha
=	Asignar a la izquierda

Ejemplos

```
a <- 3
a
```

Ejemplos

```
a <- 3
a
```

```
[1] 3
```

Ejemplos

```
a <- 3
a
```

```
[1] 3
```

```
a <- a + 1
a
```

Ejemplos

```
a <- 3
a
```

```
[1] 3
```

```
a <- a + 1
a
```

```
[1] 4
```

Ejemplos

```
a <- 3
a
```

```
[1] 3
```

```
a <- a + 1
a
```

```
[1] 4
```

```
(a <- a + 1)
```

Ejemplos

```
a <- 3
a
```

```
[1] 3
```

```
a <- a + 1
a
```

```
[1] 4
```

```
(a <- a + 1)
```

```
[1] 5
```

Ejemplos

```
r <- 1
area <- pi * (r ^ 2)
longitud <- 2 * pi * r
area
```

Ejemplos

```
r <- 1
area <- pi * (r ^ 2)
longitud <- 2 * pi * r
area
```

```
[1] 3.141593
```

Ejemplos

```
r <- 1
area <- pi * (r ^ 2)
longitud <- 2 * pi * r
area
```

```
[1] 3.141593
```

```
longitud
```

Ejemplos

```
r <- 1
area <- pi * (r ^ 2)
longitud <- 2 * pi * r
area
```

```
[1] 3.141593
```

```
longitud
```

```
[1] 6.283185
```

Ejemplos

```
r <- 1:10
area <- pi * (r ^ 2)
2 * pi * r -> longitud
area
```

Ejemplos

```
r <- 1:10
area <- pi * (r ^ 2)
2 * pi * r -> longitud
area
```

```
[1] 3.141593 12.566371 28.274334 50.265482
[5] 78.539816 113.097336 153.938040 201.061930
[9] 254.469005 314.159265
```

Funciones

- Una función es un procedimiento para realizar una determinada tarea o cálculo
- Función se asocia a un nombre, que sigue las mismas reglas que las variables
- **Nombre-de-la-Función**(*argumento 1, argumento 2, . . .)*
- Los argumentos son propios de cada función
- En algunos casos los argumentos tienen valores por defecto

Ejemplos

```
log( 2 )
```

Ejemplos

```
log( 2 )
```

```
[1] 0.6931472
```

```
log( 2, 10 )
```

Ejemplos

```
log( 2 )
```

```
[1] 0.6931472
```

```
log( 2, 10 )
```

```
[1] 0.30103
```

```
log( exp( 1 ) )
```

Ejemplos

```
log( 2 )
```

```
[1] 0.6931472
```

```
log( 2, 10 )
```

```
[1] 0.30103
```

```
log( exp( 1 ) )
```

```
[1] 1
```

Ejemplos

```
log( x = 2 , base = 10 )
```

Ejemplos

```
log( x = 2 , base = 10 )
```

```
[1]  0.30103
```

```
log( base = 10, x = 2 )
```

Ejemplos

```
log( x = 2 , base = 10 )
```

```
[1]  0.30103
```

```
log( base = 10, x = 2 )
```

```
[1] 0.30103
```

Funciones

Función	Acción
`length(obj)`	Número de componentes, elementos
`dim(obj)`	Dimensión de un objeto
`str(obj)`	Estructura de un objeto
`class(obj)`	Clase (class) o tipo de objeto
`names(obj)`	Nombres de los componentes de un objeto
`c(obj,obj,...)`	Combina objetos en un vector
`head(obj)`	Lista la primera parte de un objeto
`tail(obj)`	Lista la última parte (cola) de un objeto
`ls()`	Lista los objetos actuales
`rm(obj)`	Borra un objeto
`newobj \<- edit(obj)`	Edita un objeto y lo guarda
`fix(obj)`	Edita sobre un objeto ya creado

Funciones

`c()`	Concatenar los elementos que se indican, separados por comas
`seq()`	Generar una secuencia numérica
`rep()`	Generar un conjunto de valores repetidos
`sort()`	Ordena un vector

Funciones

`round()`	Redondeo de valores numéricos
`sqrt()`	Raíz cuadrada
`abs()`	Valor absoluto
`sin()`	Función trigonométricas seno
`cos()`	Función trigonométricas coseno
`log()`	Logaritmo natural
`exp()`	exponencial (e^x)

Funciones

`sum()`	Suma los elementos de un vector
`cumsum()`	Vector de sumas acumuladas
`max()`	Máximo de un vector
`min()`	Mínimo de un vector
`t()`	Transponer una matriz
`names()`	Nombres de filas o columnas
`nrow()`	Número de filas
`ncol()`	Número de columnas
`rownames()`	Nombre de las filas
`colnames()`	Nombres de las columnas

Funciones

`str()`	Proporciona información sobre la estructura de un objeto
`ls()`	Relación de objetos disponibles
`rm()`	Elimina uno o varios objetos
`read.table()`	Carga los datos de un fichero
`source()`	Carga el código de R escrito en un fichero

Funciones estadísticas

Función	lo que hace
`mean(x)`	Media
`median(x)`	Mediana
`sd(x)`	Desviación estándar
`var(x)`	Varianza
`sum(x)`	Suma
`range(x)`	Rango
`min(x)`	Mínimo
`max(x)`	Máximo
`scale(x,center=TRUE,scale=TRUE)`	Estandarizar

Contenido

1 Introducción

2 Elementos en R

3 Objetos en R

4 Indexado

5 Valores perdidos

6 Importar datos en R

Contenido

3 Objetos en R
- Vectores
- Matrices
- Data frames
- Listas
- Factores

Vectores

- Los vectores son un conjunto ordenado de valores
- Para calcular con todo el vector se emplea el nombre del objeto
- Para utilizar un subconjunto valores se emplea subíndices
- Los subíndices se incluyen entre corchetes (x [3])
- Los subíndices están en el rango: 1 — número de elementos del vector
- Los subíndices pueden ser expresiones

Ejemplos

```
x <- c( 8, 5, 2, 4, 1, 6, 3 )
length( x )
```

Ejemplos

```
x <- c( 8, 5, 2, 4, 1, 6, 3 )
length( x )
```

```
[1]  7
```

```
x
```

Ejemplos

```
x <- c( 8, 5, 2, 4, 1, 6, 3 )
length( x )
```

```
[1] 7
```

```
x
```

```
[1] 8 5 2 4 1 6 3
```

Ejemplos

```
x <- c( 8, 5, 2, 4, 1, 6, 3 )
length( x )
```

```
[1]  7
```

```
x
```

```
[1] 8 5 2 4 1 6 3
```

```
x[]
```

Ejemplos

```
x <- c( 8, 5, 2, 4, 1, 6, 3 )
length( x )
```

```
[1]  7
```

```
x
```

```
[1] 8 5 2 4 1 6 3
```

```
x[]
```

```
[1] 8 5 2 4 1 6 3
```

Ejemplos

```
x[1]
```

Ejemplos

```
x[1]
```

```
[1] 8
```

```
x[2:4]
```

Ejemplos

```
x[1]
```

```
[1] 8
```

```
x[2:4]
```

```
[1] 5 2 4
```

Ejemplos

```
x[1]
```

```
[1] 8
```

```
x[2:4]
```

```
[1] 5 2 4
```

```
x[ c( 3, 5 ) ]
```

Ejemplos

```
x[1]
```

```
[1] 8
```

```
x[2:4]
```

```
[1] 5 2 4
```

```
x[ c( 3, 5 ) ]
```

```
[1] 2 1
```

Ejemplos

```
x[1]
```

```
[1] 8
```

```
x[2:4]
```

```
[1] 5 2 4
```

```
x[ c( 3, 5 ) ]
```

```
[1] 2 1
```

```
x[-1]
```

Ejemplos

```
x[1]
```

```
[1] 8
```

```
x[2:4]
```

```
[1] 5 2 4
```

```
x[ c( 3, 5 ) ]
```

```
[1] 2 1
```

```
x[-1]
```

```
[1] 5 2 4 1 6 3
```

Ejemplos

```
x1 <- seq(1, 100, by=2)
x1
```

Ejemplos

```
x1 <- seq(1, 100, by=2)
x1
```

```
[1] 1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49
[26] 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99
```

Ejemplos

```
x1 <- seq(1, 100, by=2)
x1
```

```
[1] 1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49
[26] 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99
```

```
seq(1, 100, length=10)
```

Ejemplos

```
x1 <- seq(1, 100, by=2)
x1
```

```
[1] 1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49
[26] 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99
```

```
seq(1, 100, length=10)
```

```
[1] 1 12 23 34 45 56 67 78 89 100
```

Ejemplos

```
x <- c(1, 2, 3)
x
```

Ejemplos

```
x <- c(1, 2, 3)
x
```

```
[1] 1 2 3
```

Ejemplos

```
x <- c(1, 2, 3)
x
```

```
[1] 1 2 3
```

```
x <- seq(1, 100, length=10)
y <- seq(2, 100, length=50)
z <- c(x, y)
z
```

Ejemplos

```
x <- c(1, 2, 3)
x
```

```
[1] 1 2 3
```

```
x <- seq(1, 100, length=10)
y <- seq(2, 100, length=50)
z <- c(x, y)
z
```

```
[1] 1 12 23 34 45 56 67 78 89 100 2 4 6 8 10 12 14 16 18
[20] 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56
[39] 58 60 62 64 66 68 70 72 74 76 78 80 82 84 86 88 90 92 94
[58] 96 98 100
```

Ejemplos

```
x <- 1:5
x + 1
```

Ejemplos

```
x <- 1:5
x + 1
```

```
[1] 2 3 4 5 6
```

Ejemplos

```
x <- 1:5
x + 1
```

```
[1] 2 3 4 5 6
```

```
y <- 1:10
x + y
```

Ejemplos

```
x <- 1:5
x + 1
```

```
[1] 2 3 4 5 6
```

```
y <- 1:10
x + y
```

```
2 4 6 8 10 7 9 11 13 15
```

Ejemplos

```
x <- 1:5
x + 1
```

```
[1] 2 3 4 5 6
```

```
y <- 1:10
x + y
```

```
2 4 6 8 10 7 9 11 13 15
```

Función **which** devuelve la posición de los elementos que cumplen la condición.

```
w<-x + y
which(w>10)
```

Ejemplos

```
x <- 1:5
x + 1
```

```
[1] 2 3 4 5 6
```

```
y <- 1:10
x + y
```

```
2 4 6 8 10 7 9 11 13 15
```

Función **which** devuelve la posición de los elementos que cumplen la condición.

```
w<-x + y
which(w>10)
```

```
[1]  8  9 10
```

Ejemplos

Es posible crear vectores de caracteres, los cuales también son utilizadas en R, para etiquetar gráficos. Una cadena de caracteres se construye escribiendo entre comillas la sucesión de caracteres que la define, por ejemplo

```
y <- c("Bogota", "Cali", "Medellin", "Cartagena", "Barranquilla")
```

Ejemplos

Es posible crear vectores de caracteres, los cuales también son utilizadas en R, para etiquetar gráficos. Una cadena de caracteres se construye escribiendo entre comillas la sucesión de caracteres que la define, por ejemplo

```
y <- c("Bogota", "Cali", "Medellin", "Cartagena", "Barranquilla")
```

```
[1] "Bogota"  "Cali"  "Medellin"  "Cartagena"  "Barranquilla"
```

Ejemplos

La función **paste()** une todos los vectores que se suministran y construye una sola cadena de caracteres

```
y <- c("Bogota", "Cali", "Medellin", "Cartagena", "Barranquilla")
w <- c("Muy malo", "Malo", "Regular", "Bueno", "Muy bueno")
paste(y, w, sep=" ")
```

Ejemplos

La función **paste()** une todos los vectores que se suministran y construye una sola cadena de caracteres

```
y <- c("Bogota", "Cali", "Medellin", "Cartagena", "Barranquilla")
w <- c("Muy malo", "Malo", "Regular", "Bueno", "Muy bueno")
paste(y, w, sep=" ")
```

```
[1] "Bogota Muy malo" "Cali Malo" "Medellin Regular"
[4] "Cartagena Bueno" "Barranquilla Muy bueno"
```

Ejercicio 1.

1. Calcula la suma de 25 y 75 (+)
2. Calcula la raíz cuadrada de 25 (sqrt)
3. Crea una secuencia del 1 al 15 (:)
4. Crea una secuencia del 1 al 20 de 2 en 2 (seq)
5. Repite la secuencia anterior 3 veces (rep)
6. Crea un vector de longitud 5 y calcula su media (c, mean)
7. Súmale a todos los componentes de ese vector 3 y calcula su media (+, mean)
8. Crea un vector de longitud 5 con una posición vacía y calcula su media (c, NA, mean)

Ejercicio 2.

1. Genere el siguiente vector de edades
Edad<- c(25, 30, 22, 26, 32, 25, 21, 29, 34, 37, 30, 28, 41, 22)

2. Genere un vector de los nombres de los funcionarios. Asigne los nombres a las edades (names())
Pedro, James, Miguel, Daniel, Jose, Diana, Sandra, Adriana, Alexis, Claudia, Ana, Andrea, Jorge, Carlos

3. Identifique cuales son los nombres de los empleados con edades menores 25 ¿Como lo haría en R? (which())

4. Cree un objeto que junte el vector Edad con Nombres. De tal forma que se vizualice lo siguiente: (paste())
"Pedro tiene 25 años"

Contenido

3 Objetos en R
- Vectores
- Matrices
- Data frames
- Listas
- Factores

Matrices

- Una matriz es un conjunto ordenado de vectores
- Los elementos de la matriz están ordenados por filas y columnas
- Todos los vectores son del mismo tipo: enteros, caracteres, ...
- Los elementos de una matriz se identifican por dos subíndices
- El uso de los subíndices sigue las mismas reglas que en el caso de los vectores
- Se puede crear uniendo vectores o mediante la función matrix()

Ejemplos

```
m <- matrix( 1:12, 4, 3 )
m
```

Ejemplos

```
m <- matrix( 1:12, 4, 3 )
m
```

```
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
```

Contenido

3 Objetos en R
- Vectores
- Matrices
- Data frames
- Listas
- Factores

Data frames

- Son semejantes a las matrices
- Se organizan por filas y columnas
- Las columnas no tienen por que ser homogéneas
- Las columnas tienen nombre
- Habitualmente los data frames se obtienen de la lectura de un fichero de datos

Ejemplos

```
cars
```

Ejemplos

```
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
...
...
```

Ejemplos

```
dim(cars)
```

Ejemplos

```
dim(cars)
```

```
[1] 50  2
```

Ejemplos

```
dim(cars)
```

```
[1] 50  2
```

```
head(cars)
```

Ejemplos

```
dim(cars)
```

```
[1] 50  2
```

```
head(cars)
```

```
  speed dist
1     4    2
2     4   10
3     7    4
4     7   22
5     8   16
6     9   10
```

Ejemplos

Función **cbind**

```
z <- 1:10
y <- 1:10
x <- 1:10
M <- cbind(x, y, z)
M
```

Ejemplos

Función **cbind**

```
z <- 1:10
y <- 1:10
x <- 1:10
M <- cbind(x, y, z)
M
```

```
      x  y  z
 [1,]  1  1  1
 [2,]  2  2  2
 [3,]  3  3  3
 [4,]  4  4  4
 [5,]  5  5  5
 [6,]  6  6  6
 [7,]  7  7  7
 [8,]  8  8  8
 [9,]  9  9  9
[10,] 10 10 10
```

Ejemplos

Función **rbind**

```
M <- rbind(x, y, z)
M
```

Ejemplos

Función **rbind**

```
M <- rbind(x, y, z)
M
```

```
  [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
x    1    2    3    4    5    6    7    8    9    10
y    1    2    3    4    5    6    7    8    9    10
z    1    2    3    4    5    6    7    8    9    10
```

Data frames

Datasets

Así como la base utilizada *(cars)*, existen múltiples datasets preinstalados en R y se puede acceder a ellos directamente o mediante librerías.

En el siguiente link encontrará la descripción de los datasets.

https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/00Index.html

Contenido

3 Objetos en R
- Vectores
- Matrices
- Data frames
- Listas
- Factores

Listas

Son objetos que pueden contener conjuntos heterogéneos de objetos:

- valores
- vectores
- matrices
- data frames
- listas

Se suelen encontrar como resultado de funciones

Ejemplos

```
lista <- list(a=c(1,3,5),
              b=c('l', 'p', 'r', 's'),
              c=3)
lista
```

Ejemplos

```
lista <- list(a=c(1,3,5),
              b=c('l', 'p', 'r', 's'),
              c=3)
lista
```

```
$a
[1] 1 3 5
$b
[1] "l" "p" "r" "s"
$c
[1] 3
```

Ejemplos

```
lista <- list(a=c(1,3,5),
              b=c('l', 'p', 'r', 's'),
              c=3)
lista
```

```
$a
[1] 1 3 5
$b
[1] "l" "p" "r" "s"
$c
[1] 3
```

```
class(lista)
```

Ejemplos

```
lista <- list(a=c(1,3,5),
              b=c('l', 'p', 'r', 's'),
              c=3)
lista
```

```
$a
[1] 1 3 5
$b
[1] "l" "p" "r" "s"
$c
[1] 3
```

```
class(lista)
```

```
[1] "list"
```

Contenido

3 Objetos en R
- Vectores
- Matrices
- Data frames
- Listas
- Factores

Factores

Hay variables codificadas, ejemplo sexo, donde 1 es "Masculino" y 2 "Femenino", que en realidad tiene un carácter *cualitativo* o *categórico*.

```
sexo <- c(1, 1, 1, 1, 2, 2, 2)
peso <- c(60, 65, 70, 66, 80, 60, 76)
df <- data.frame(sexo, peso)
df$sexo <- factor(df$sexo)
df
```

Factores

Hay variables codificadas, ejemplo sexo, donde 1 es "Masculino" y 2 "Femenino", que en realidad tiene un carácter *cualitativo* o *categórico*.

```
sexo <- c(1, 1, 1, 1, 2, 2, 2)
peso <- c(60, 65, 70, 66, 80, 60, 76)
df <- data.frame(sexo, peso)
df$sexo <- factor(df$sexo)
df
```

```
  sexo peso
1    1   60
2    1   65
3    1   70
4    1   66
5    2   80
6    2   60
7    2   76
```

Factores

Creamos las etiquetas

```
df$sexo <- factor( df$sexo,
                   levels = c(1, 2),
                   labels = c("masculino", "femenino")
                  )
head(df)
```

Factores

Creamos las etiquetas

```
df$sexo <- factor( df$sexo,
                   levels = c(1, 2),
                   labels = c("masculino", "femenino")
                  )
head(df)
```

```
       sexo peso
1 masculino   60
2 masculino   65
3 masculino   70
4 masculino   66
5  femenino   80
6  femenino   60
```

Contenido

1 Introducción

2 Elementos en R

3 Objetos en R

4 Indexado

5 Valores perdidos

6 Importar datos en R

Contenido

4 Indexado
- Condiciones lógicas
- Vectores
- Matrices
- Listas
- Data frame

Condiciones lógicas simples

```
x <- seq(-1, 1, .1)
x
```

Condiciones lógicas simples

```
x <- seq(-1, 1, .1)
x
```

```
[1] -1.0 -0.9 -0.8 -0.7 -0.6 -0.5 -0.4 -0.3 -0.2 -0.1 0.0 0.1 0.2 0.3 0.4
[16] 0.5 0.6 0.7 0.8 0.9 1.0
```

Condiciones lógicas simples

```
x <- seq(-1, 1, .1)
x
```

```
[1] -1.0 -0.9 -0.8 -0.7 -0.6 -0.5 -0.4 -0.3 -0.2 -0.1 0.0 0.1 0.2 0.3 0.4
[16] 0.5 0.6 0.7 0.8 0.9 1.0
```

```
x < 0
```

Condiciones lógicas simples

```
x <- seq(-1, 1, .1)
x
```

```
[1] -1.0 -0.9 -0.8 -0.7 -0.6 -0.5 -0.4 -0.3 -0.2 -0.1 0.0 0.1 0.2 0.3 0.4
[16] 0.5 0.6 0.7 0.8 0.9 1.0
```

```
x < 0
```

```
[1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE FALSE
[13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

Condiciones lógicas simples

```
x <- seq(-1, 1, .1)
x
```

```
[1] -1.0 -0.9 -0.8 -0.7 -0.6 -0.5 -0.4 -0.3 -0.2 -0.1 0.0 0.1 0.2 0.3 0.4
[16] 0.5 0.6 0.7 0.8 0.9 1.0
```

```
x < 0
```

```
[1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE FALSE
[13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
x >= 0
```

Condiciones lógicas simples

```
x <- seq(-1, 1, .1)
x
```

```
[1] -1.0 -0.9 -0.8 -0.7 -0.6 -0.5 -0.4 -0.3 -0.2 -0.1 0.0 0.1 0.2 0.3 0.4
[16] 0.5 0.6 0.7 0.8 0.9 1.0
```

```
x < 0
```

```
[1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE FALSE
[13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
x >= 0
```

```
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE
[13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

Condiciones lógicas múltiples

```
cond <- (x > 0) & (x < .5)
cond
```

Condiciones lógicas múltiples

```
cond <- (x > 0) & (x < .5)
cond
```

```
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE
[13] TRUE TRUE TRUE FALSE FALSE FALSE FALSE FALSE FALSE
```

Condiciones lógicas múltiples

```
cond <- (x > 0) & (x < .5)
cond
```

```
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE
[13] TRUE TRUE TRUE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
cond <- (x >= .5) | (x <= -.5)
cond
```

Condiciones lógicas múltiples

```
cond <- (x > 0) & (x < .5)
cond
```

```
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE
[13] TRUE TRUE TRUE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
cond <- (x >= .5) | (x <= -.5)
cond
```

```
[1] TRUE TRUE TRUE TRUE TRUE TRUE FALSE FALSE FALSE FALSE FALSE FALSE
[13] FALSE FALSE FALSE TRUE TRUE TRUE TRUE TRUE TRUE
```

Operaciones con condiciones lógicas

```
sum(cond)
```

Operaciones con condiciones lógicas

```
sum(cond)
```

```
[1] 12
```

Operaciones con condiciones lógicas

```
sum(cond)
```

```
[1] 12
```

```
sum(!cond)
```

Operaciones con condiciones lógicas

```
sum(cond)
```

```
[1] 12
```

```
sum(!cond)
```

```
[1] 9
```

Operaciones con condiciones lógicas

```
sum(cond)
```

```
[1] 12
```

```
sum(!cond)
```

```
[1] 9
```

```
as.numeric(cond)
```

Operaciones con condiciones lógicas

```
sum(cond)
```

```
[1] 12
```

```
sum(!cond)
```

```
[1] 9
```

```
as.numeric(cond)
```

```
[1] 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1
```

Contenido

4 Indexado
- Condiciones lógicas
- Vectores
- Matrices
- Listas
- Data frame

Indexado en vectores

```
x <- seq(1, 100, 2)
x
```

Indexado en vectores

```
x <- seq(1, 100, 2)
x
```

```
 [1]  1  3  5  7  9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49
[26] 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99
```

Indexado en vectores

```
x <- seq(1, 100, 2)
x
```

```
[1] 1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49
[26] 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99
```

```
x[x > 20]
```

Indexado en vectores

```
x <- seq(1, 100, 2)
x
```

```
[1] 1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49
[26] 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99
```

```
x[x > 20]
```

```
[1] 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67 69
[26] 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99
```

Indexado en vectores

```
z <- seq(-10, 10, by = .5)
z
```

Indexado en vectores

```
z <- seq(-10, 10, by = .5)
z
```

```
 [1] -10.0 -9.5 -9.0 -8.5 -8.0 -7.5 -7.0 -6.5 -6.0 -5.5 -5.0 -4.5
[13] -4.0 -3.5 -3.0 -2.5 -2.0 -1.5 -1.0 -0.5  0.0  0.5  1.0  1.5
[25]  2.0  2.5  3.0  3.5  4.0  4.5  5.0  5.5  6.0  6.5  7.0  7.5
[37]  8.0  8.5  9.0  9.5 10.0
```

Indexado en vectores

```
z <- seq(-10, 10, by = .5)
z
```

```
 [1] -10.0 -9.5 -9.0 -8.5 -8.0 -7.5 -7.0 -6.5 -6.0 -5.5 -5.0 -4.5
[13] -4.0 -3.5 -3.0 -2.5 -2.0 -1.5 -1.0 -0.5  0.0  0.5  1.0  1.5
[25]  2.0  2.5  3.0  3.5  4.0  4.5  5.0  5.5  6.0  6.5  7.0  7.5
[37]  8.0  8.5  9.0  9.5 10.0
```

```
z[z < -5 | z > 5]
```

Indexado en vectores

```
z <- seq(-10, 10, by = .5)
z
```

```
[1] -10.0 -9.5 -9.0 -8.5 -8.0 -7.5 -7.0 -6.5 -6.0 -5.5 -5.0 -4.5
[13] -4.0 -3.5 -3.0 -2.5 -2.0 -1.5 -1.0 -0.5 0.0 0.5 1.0 1.5
[25] 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5
[37] 8.0 8.5 9.0 9.5 10.0
```

```
z[z < -5 | z > 5]
```

```
[1] -10.0 -9.5 -9.0 -8.5 -8.0 -7.5 -7.0 -6.5 -6.0 -5.5 5.5 6.0
[13] 6.5 7.0 7.5 8.0 8.5 9.0 9.5 10.0
```

Indexado en vectores

```
z <- seq(-10, 10, by = .5)
z
```

```
[1] -10.0 -9.5 -9.0 -8.5 -8.0 -7.5 -7.0 -6.5 -6.0 -5.5 -5.0 -4.5
[13] -4.0 -3.5 -3.0 -2.5 -2.0 -1.5 -1.0 -0.5 0.0 0.5 1.0 1.5
[25] 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5
[37] 8.0 8.5 9.0 9.5 10.0
```

```
z[z < -5 | z > 5]
```

```
[1] -10.0 -9.5 -9.0 -8.5 -8.0 -7.5 -7.0 -6.5 -6.0 -5.5 5.5 6.0
[13] 6.5 7.0 7.5 8.0 8.5 9.0 9.5 10.0
```

```
cond <- (z >= 0 & z <= 5)
z[cond]
```

Indexado en vectores

```
z <- seq(-10, 10, by = .5)
z
```

```
 [1] -10.0 -9.5 -9.0 -8.5 -8.0 -7.5 -7.0 -6.5 -6.0 -5.5 -5.0 -4.5
[13] -4.0 -3.5 -3.0 -2.5 -2.0 -1.5 -1.0 -0.5  0.0  0.5  1.0  1.5
[25]  2.0  2.5  3.0  3.5  4.0  4.5  5.0  5.5  6.0  6.5  7.0  7.5
[37]  8.0  8.5  9.0  9.5 10.0
```

```
z[z < -5 | z > 5]
```

```
 [1] -10.0 -9.5 -9.0 -8.5 -8.0 -7.5 -7.0 -6.5 -6.0 -5.5  5.5  6.0
[13]  6.5  7.0  7.5  8.0  8.5  9.0  9.5 10.0
```

```
cond <- (z >= 0 & z <= 5)
z[cond]
```

```
 [1] 0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```

Contenido

4 Indexado
- Condiciones lógicas
- Vectores
- Matrices
- Listas
- Data frame

Indexado en matrices

```
m <- matrix( 1:12, 4, 3 )
m
```

Indexado en matrices

```
m <- matrix( 1:12, 4, 3 )
m
```

```
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
```

Indexado en matrices

```
m <- matrix( 1:12, 4, 3 )
m
```

```
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
```

```
m[ 1, ]
```

Indexado en matrices

```
m <- matrix( 1:12, 4, 3 )
m
```

```
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
```

```
m[ 1, ]
```

```
[1] 1 5 9
```

Indexado en matrices

```
m <- matrix( 1:12, 4, 3 )
m
```

```
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
```

```
m[ 1, ]
```

```
[1] 1 5 9
```

```
m[3:4,2:3]
```

Indexado en matrices

```
m <- matrix( 1:12, 4, 3 )
m
```

```
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
```

```
m[ 1, ]
```

```
[1] 1 5 9
```

```
m[3:4,2:3]
```

```
     [,1] [,2]
[1,]    7   11
[2,]    8   12
```

Indexado en matrices

```
m[-1,]
```

Indexado en matrices

```
m[-1,]
```

```
     [,1] [,2] [,3]
[1,]    2    6   10
[2,]    3    7   11
[3,]    4    8   12
```

Indexado en matrices

```
m[-1,]
```

```
     [,1] [,2] [,3]
[1,]    2    6   10
[2,]    3    7   11
[3,]    4    8   12
```

```
m[2,3]<-"Cali"
```

Indexado en matrices

```
m[-1,]
```

```
     [,1] [,2] [,3]
[1,]    2    6   10
[2,]    3    7   11
[3,]    4    8   12
```

```
m[2,3]<-"Cali"
```

```
     [,1] [,2] [,3]
[1,] "1"  "5"  "9"
[2,] "2"  "6"  "Cali"
[3,] "3"  "7"  "11"
[4,] "4"  "8"  "12"
```

Indexado en matrices

```
m[-1,]
```

```
     [,1] [,2] [,3]
[1,]    2    6   10
[2,]    3    7   11
[3,]    4    8   12
```

```
m[2,3]<-"Cali"
```

```
     [,1] [,2] [,3]
[1,] "1"  "5"  "9"
[2,] "2"  "6"  "Cali"
[3,] "3"  "7"  "11"
[4,] "4"  "8"  "12"
```

```
class(m)
```

Indexado en matrices

```
m[-1,]
```

```
     [,1] [,2] [,3]
[1,]    2    6   10
[2,]    3    7   11
[3,]    4    8   12
```

```
m[2,3]<-"Cali"
```

```
     [,1] [,2] [,3]
[1,] "1"  "5"  "9"
[2,] "2"  "6"  "Cali"
[3,] "3"  "7"  "11"
[4,] "4"  "8"  "12"
```

```
class(m)
```

```
[1] "matrix"
```

Indexado en matrices

```
m[-1,]
```

```
     [,1] [,2] [,3]
[1,]    2    6   10
[2,]    3    7   11
[3,]    4    8   12
```

```
m[2,3]<-"Cali"
```

```
     [,1] [,2] [,3]
[1,] "1"  "5"  "9"
[2,] "2"  "6"  "Cali"
[3,] "3"  "7"  "11"
[4,] "4"  "8"  "12"
```

```
class(m)
```

```
[1] "matrix"
```

```
str(m)
```

Indexado en matrices

```
m[-1,]
```

```
     [,1] [,2] [,3]
[1,]    2    6   10
[2,]    3    7   11
[3,]    4    8   12
```

```
m[2,3]<-"Cali"
```

```
     [,1] [,2] [,3]
[1,] "1"  "5"  "9"
[2,] "2"  "6"  "Cali"
[3,] "3"  "7"  "11"
[4,] "4"  "8"  "12"
```

```
class(m)
```

```
[1] "matrix"
```

```
str(m)
```

```
chr [1:4, 1:3] "1" "2" "3" "4" "5" "6" "7" "8" "9" "Cali" "11" "12"
```

Ejercicio 3.

1. Los siguientes valores (50, 70, 130, 40, 100, 70, 140, 30, 110) corresponden al precio de licores (xbotella) en discotecas de Cali, clasificados por licores (columnas: Blanco, Caldas, Márquez) *(colnames())* y discotecas (filas: Living, Pérgola, Caderona) *(rownames())*
2. Se solicita anexar en la matriz los precios de las discotecas: Lolas(120,130,140) y Urbano(80, 100,120).
3. Además se solicita anexar el valor de la cerveza "corona" la cuál es muy vendida en dichas discotecas: (20,15,25,17,22)
4. Seleccione todos los precios de la discoteca Lolas
5. Identifique cual es la discoteca con precios más altos en licor Márquez
6. Identifique cuales son las discotecas con precios más altos en Blanco y Corona

Contenido

4 Indexado
- Condiciones lógicas
- Vectores
- Matrices
- Listas
- Data frame

Indexado en listas

```
length(lista)
```

Indexado en listas

```
length(lista)
```

```
[1] 3
```

Indexado en listas

```
length(lista)
```

```
[1] 3
```

```
lista$b
```

Indexado en listas

```
length(lista)
```

```
[1] 3
```

```
lista$b
```

```
[1] "l" "p" "r" "s"
```

Indexado en listas

```
length(lista)
```

```
[1] 3
```

```
lista$b
```

```
[1] "l" "p" "r" "s"
```

```
lista$b[3]
```

Indexado en listas

```
length(lista)
```

```
[1] 3
```

```
lista$b
```

```
[1] "l" "p" "r" "s"
```

```
lista$b[3]
```

```
[1] "r"
```

Indexado en listas

```
length(lista)
```

```
[1] 3
```

```
lista$b
```

```
[1] "l" "p" "r" "s"
```

```
lista$b[3]
```

```
[1] "r"
```

```
lista[[2]][3]
```

Indexado en listas

```
length(lista)
```

```
[1] 3
```

```
lista$b
```

```
[1] "l" "p" "r" "s"
```

```
lista$b[3]
```

```
[1] "r"
```

```
lista[[2]][3]
```

```
[1] "r"
```

Contenido

4 Indexado
- Condiciones lógicas
- Vectores
- Matrices
- Listas
- Data frame

Indexado en Data frame

```
cars[ ,2]
```

Indexado en Data frame

```
cars[ ,2]
```

```
 [1]   2  10   4  22  16  10  18  26  34  17  28  14  20  24  28  26  34  34  46  26
[21]  36  60  80  20  26  54  32  40  32  40  50  42  56  76  84  36  46  68  32  48
[41]  52  56  64  66  54  70  92  93 120  85
```

Indexado en Data frame

```
cars[ ,2]
```

```
 [1]   2  10   4  22  16  10  18  26  34  17  28  14  20  24  28  26  34  34  46  26
[21]  36  60  80  20  26  54  32  40  32  40  50  42  56  76  84  36  46  68  32  48
[41]  52  56  64  66  54  70  92  93 120  85
```

```
cars$dist
```

Indexado en Data frame

```
cars[ ,2]
```

```
 [1]   2  10   4  22  16  10  18  26  34  17  28  14  20  24  28  26  34  34  46  26
[21]  36  60  80  20  26  54  32  40  32  40  50  42  56  76  84  36  46  68  32  48
[41]  52  56  64  66  54  70  92  93 120  85
```

```
cars$dist
```

```
 [1]   2  10   4  22  16  10  18  26  34  17  28  14  20  24  28  26  34  34  46  26
[21]  36  60  80  20  26  54  32  40  32  40  50  42  56  76  84  36  46  68  32  48
[41]  52  56  64  66  54  70  92  93 120  85
```

```
cars[ ,"dist"]
```

Indexado en Data frame

```
cars[ ,2]
```

```
 [1]   2  10   4  22  16  10  18  26  34  17  28  14  20  24  28  26  34  34  46  26
[21]  36  60  80  20  26  54  32  40  32  40  50  42  56  76  84  36  46  68  32  48
[41]  52  56  64  66  54  70  92  93 120  85
```

```
cars$dist
```

```
 [1]   2  10   4  22  16  10  18  26  34  17  28  14  20  24  28  26  34  34  46  26
[21]  36  60  80  20  26  54  32  40  32  40  50  42  56  76  84  36  46  68  32  48
[41]  52  56  64  66  54  70  92  93 120  85
```

```
cars[ ,"dist"]
```

```
 [1]   2  10   4  22  16  10  18  26  34  17  28  14  20  24  28  26  34  34  46  26
[21]  36  60  80  20  26  54  32  40  32  40  50  42  56  76  84  36  46  68  32  48
[41]  52  56  64  66  54  70  92  93 120  85
```

Indexado en Data frame

```
cars[ 3, ]
```

Indexado en Data frame

```
cars[ 3, ]
```

```
  speed dist
3     7    4
```

Indexado en Data frame

```
cars[ 3, ]
```

```
  speed dist
3     7    4
```

```
cars[3,2]
```

Indexado en Data frame

```
cars[ 3, ]
```

```
  speed dist
3     7    4
```

```
cars[3,2]
```

```
 [1] 4
```

Indexado lógico en Data frame

```
df <- data.frame(x = 1:5,
                 y = rnorm(10),
                 z = 0)
```

Indexado lógico en Data frame

```
df <- data.frame(x = 1:5,
                 y = rnorm(10),
                 z = 0)
```

```
   x           y z
1  1  1.78139992 0
2  2  1.32193053 0
3  3  1.14856140 0
4  4 -1.26046862 0
5  5  0.52795348 0
6  1  0.86030751 0
7  2 -0.03876488 0
8  3 -1.31705080 0
9  4  0.46210892 0
10 5  0.74472228 0
```

Indexado lógico en Data frame

```
df[df$y > 0,]
```

Indexado lógico en Data frame

```
df[df$y > 0,]
```

```
  x         y z
1 1 1.8027998 0
4 4 1.2014048 0
6 1 0.3066946 0
7 2 0.3458662 0
8 3 1.9886410 0
9 4 0.2423531 0
```

Indexado lógico en Data frame

```
df[df$y > 0,]
```

```
  x         y z
1 1 1.8027998 0
4 4 1.2014048 0
6 1 0.3066946 0
7 2 0.3458662 0
8 3 1.9886410 0
9 4 0.2423531 0
```

La función **subset** simplifica el código.

```
subset(df, y > 0)
```

Indexado lógico en Data frame

```
df[df$y > 0,]
```

```
  x         y z
1 1 1.8027998 0
4 4 1.2014048 0
6 1 0.3066946 0
7 2 0.3458662 0
8 3 1.9886410 0
9 4 0.2423531 0
```

La función **subset** simplifica el código.

```
subset(df, y > 0)
```

```
  x         y z
1 1 1.8027998 0
4 4 1.2014048 0
6 1 0.3066946 0
7 2 0.3458662 0
8 3 1.9886410 0
9 4 0.2423531 0
```

Ejercicio 4.

Generemos el siguiente data.frame

```
id <- c( 1:5 )
date <- c( "10/07/08", "10/08/08", "10/09/08", "10/10/08", "10/11/08" )
country <- c( "US", "US", "UK", "UK", "UK" )
gender <- c( "M", "F", "F", NA, "F" )
age <- c( NA, 45, 25, 39, 99 )
q1 <- c( 5, 3, 3, 3, 2 )
q2 <- c( 5, 5, 5, NA, 2 )
q3 <- c( 5, 5, 2, NA, 1 )
df <- data.frame( id, date, country, gender, age, q1, q2, q3,
stringsAsFactors = FALSE )

df
```

Ejercicio 4.

Generemos el siguiente data.frame

```
id <- c( 1:5 )
date <- c( "10/07/08", "10/08/08", "10/09/08", "10/10/08", "10/11/08" )
country <- c( "US", "US", "UK", "UK", "UK" )
gender <- c( "M", "F", "F", NA, "F" )
age <- c( NA, 45, 25, 39, 99 )
q1 <- c( 5, 3, 3, 3, 2 )
q2 <- c( 5, 5, 5, NA, 2 )
q3 <- c( 5, 5, 2, NA, 1 )
df <- data.frame( id, date, country, gender, age, q1, q2, q3,
stringsAsFactors = FALSE )

df
```

```
  id     date country gender age q1 q2 q3
1  1 10/07/08      US      M  NA  5  5  5
2  2 10/08/08      US      F  45  3  5  5
3  3 10/09/08      UK      F  25  3  5  2
4  4 10/10/08      UK   <NA>  39  3 NA NA
5  5 10/11/08      UK      F  99  2  2  1
```

Ejercicio 4.

Con el objeto creado anteriormente df

1. Crea un nuevo data frame que incluya únicamente las variables q1, q2 y q3.
2. Crea de dos formas un nuevo data frame que incluya el país al que pertenecen los sujetos (con el nombre y con el índice)
3. Crea un nuevo data frame eliminando las variables q2 y q3
4. Selecciona las 3 primeras filas
5. ¿Cómo escribirías la condición de ser mujer y de UK?
6. Selecciona las observaciones que cumplan dicha condición

Crear una nueva variable en un data.frame

```
df$nueva <- df$q1 * 2
df
```

Crear una nueva variable en un data.frame

```
df$nueva <- df$q1 * 2
df
```

```
  id     date country gender age q1 q2 q3 nueva
1  1 10/07/08      US      M  NA  5  5  5    10
2  2 10/08/08      US      F  45  3  5  5     6
3  3 10/09/08      UK      F  25  3  5  2     6
4  4 10/10/08      UK   <NA>  39  3 NA NA     6
5  5 10/11/08      UK      F  99  2  2  1     4
```

Crear una nueva variable en un data.frame

```
df$agecat[ df$age > 75 ] <- "anciano" # crea la variable
df$agecat[ df$age <= 75 & df$age > 44 ] <- "maduro"
df$agecat[ df$age <= 44 ] <- "joven"
df
```

Crear una nueva variable en un data.frame

```
df$agecat[ df$age > 75 ] <- "anciano" # crea la variable
df$agecat[ df$age <= 75 & df$age > 44 ] <- "maduro"
df$agecat[ df$age <= 44 ] <- "joven"
df
```

```
  id     date country gender age q1 q2 q3 nueva  agecat
1  1 10/07/08      US      M  NA  5  5  5    10    <NA>
2  2 10/08/08      US      F  45  3  5  5     6  maduro
3  3 10/09/08      UK      F  25  3  5  2     6   joven
4  4 10/10/08      UK   <NA>  39  3 NA NA     6   joven
5  5 10/11/08      UK      F  99  2  2  1     4 anciano
```

Renombrar variables

```
names( df )
```

Renombrar variables

```
names( df )
```

```
[1] "id" "date" "country" "gender" "age" "q1" "q2"
[8] "q3" "nueva" "agecat"
```

Renombrar variables

```
names( df )
```

```
[1] "id" "date" "country" "gender" "age" "q1" "q2"
[8] "q3" "nueva" "agecat"
```

```
names( df )[ 1 ] <- "ID"
names( df )[ 3 ] <- "pais"
names( df )[ 4 ] <- "G"
names( df )[ 6:8 ] <- c( "it1", "it2", "it3" )
df
```

Renombrar variables

```
names( df )
```

```
[1] "id" "date" "country" "gender" "age" "q1" "q2"
[8] "q3" "nueva" "agecat"
```

```
names( df )[ 1 ] <- "ID"
names( df )[ 3 ] <- "pais"
names( df )[ 4 ] <- "G"
names( df )[ 6:8 ] <- c( "it1", "it2", "it3" )
df
```

```
  ID     date pais    G age it1 it2 it3 nueva  agecat
1  1 10/07/08   US    M  NA   5   5   5    10    <NA>
2  2 10/08/08   US    F  45   3   5   5     6  maduro
3  3 10/09/08   UK    F  25   3   5   2     6   joven
4  4 10/10/08   UK <NA>  39   3  NA  NA     6   joven
5  5 10/11/08   UK    F  99   2   2   1     4 anciano
```

Contenido

1 Introducción

2 Elementos en R

3 Objetos en R

4 Indexado

5 Valores perdidos

6 Importar datos en R

Valores perdidos

- Valores imposibles: NaN (Not a Number)
- Los valores perdidos: NA (Not available)

Hay muchas funciones para identificar estos valores: is.na()

Valores perdidos

```
y <- c( 1, 2, 3, NA )
is.na( y )
```

Valores perdidos

```
y <- c( 1, 2, 3, NA )
is.na( y )
```

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

Valores perdidos

```
y <- c( 1, 2, 3, NA )
is.na( y )
```

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

```
is.na( df[ , 4:10 ] )
```

Valores perdidos

```
y <- c( 1, 2, 3, NA )
is.na( y )
```

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

```
is.na( df[ , 4:10 ] )
```

```
         G   age   it1   it2   it3 nueva agecat
[1,] FALSE  TRUE FALSE FALSE FALSE FALSE   TRUE
[2,] FALSE FALSE FALSE FALSE FALSE FALSE  FALSE
[3,] FALSE FALSE FALSE FALSE FALSE FALSE  FALSE
[4,]  TRUE FALSE FALSE  TRUE  TRUE FALSE  FALSE
[5,] FALSE FALSE FALSE FALSE FALSE FALSE  FALSE
```

Recodificar valores perdidos

```
df$age[ is.na( df$age ) ] <- 9999
```

Recodificar valores perdidos

```
df$age[ is.na( df$age ) ] <- 9999
```

```
  ID     date pais    G  age it1 it2 it3 nueva  agecat
1  1 10/07/08   US    M 9999   5   5   5    10    <NA>
2  2 10/08/08   US    F   45   3   5   5     6  maduro
3  3 10/09/08   UK    F   25   3   5   2     6   joven
4  4 10/10/08   UK <NA>   39   3  NA  NA     6   joven
5  5 10/11/08   UK    F   99   2   2   1     4 anciano
```

Recodificar valores perdidos

```
df$age[ is.na( df$age ) ] <- 9999
```

```
  ID     date pais    G  age it1 it2 it3 nueva  agecat
1  1 10/07/08   US    M 9999   5   5   5    10    <NA>
2  2 10/08/08   US    F   45   3   5   5     6  maduro
3  3 10/09/08   UK    F   25   3   5   2     6   joven
4  4 10/10/08   UK <NA>   39   3  NA  NA     6   joven
5  5 10/11/08   UK    F   99   2   2   1     4 anciano
```

Excluir NA del análisis

```
x <- c( 1, 2, NA, 3 )
sum(x)
```

Recodificar valores perdidos

```
df$age[ is.na( df$age ) ] <- 9999
```

```
  ID     date pais    G  age it1 it2 it3 nueva  agecat
1  1 10/07/08   US    M 9999   5   5   5    10    <NA>
2  2 10/08/08   US    F   45   3   5   5     6  maduro
3  3 10/09/08   UK    F   25   3   5   2     6   joven
4  4 10/10/08   UK <NA>   39   3  NA  NA     6   joven
5  5 10/11/08   UK    F   99   2   2   1     4 anciano
```

Excluir NA del análisis

```
x <- c( 1, 2, NA, 3 )
sum(x)
```

```
[1] NA
```

Recodificar valores perdidos

```
df$age[ is.na( df$age ) ] <- 9999
```

```
  ID     date pais    G  age it1 it2 it3 nueva  agecat
1  1 10/07/08   US    M 9999   5   5   5    10    <NA>
2  2 10/08/08   US    F   45   3   5   5     6  maduro
3  3 10/09/08   UK    F   25   3   5   2     6   joven
4  4 10/10/08   UK <NA>   39   3  NA  NA     6   joven
5  5 10/11/08   UK    F   99   2   2   1     4 anciano
```

Excluir NA del análisis

```
x <- c( 1, 2, NA, 3 )
sum(x)
```

```
[1] NA
```

```
x <- c( 1, 2, NA, 3 )
sum(x, na.rm = TRUE)
```

Recodificar valores perdidos

```
df$age[ is.na( df$age ) ] <- 9999
```

```
  ID     date pais    G  age it1 it2 it3 nueva  agecat
1  1 10/07/08   US    M 9999   5   5   5    10    <NA>
2  2 10/08/08   US    F   45   3   5   5     6  maduro
3  3 10/09/08   UK    F   25   3   5   2     6   joven
4  4 10/10/08   UK <NA>   39   3  NA  NA     6   joven
5  5 10/11/08   UK    F   99   2   2   1     4 anciano
```

Excluir NA del análisis

```
x <- c( 1, 2, NA, 3 )
sum(x)
```

```
[1] NA
```

```
x <- c( 1, 2, NA, 3 )
sum(x, na.rm = TRUE)
```

```
[1] 6
```

na.omit()

na.omit() que elimina cualquier fila de un dataframe que tenga valores faltantes.

```
df <- na.omit( df )
df
```

na.omit()

na.omit() que elimina cualquier fila de un dataframe que tenga valores faltantes.

```
df <- na.omit( df )
df
```

```
  ID     date pais G age it1 it2 it3 nueva  agecat
2  2 10/08/08   US F  45   3   5   5     6  maduro
3  3 10/09/08   UK F  25   3   5   2     6   joven
5  5 10/11/08   UK F  99   2   2   1     4 anciano
```

Convertir tipos de variable

lógico	convierte
`is.numeric()`	`as.numeric()`
`is.character()`	`as.character()`
`is.vector()`	`as.vector()`
`is.matrix()`	`as.matrix()`
`is.data.frame()`	`as.data.frame()`

```
a <- c( 1, 2, 3 )
is.numeric( a )
```

Convertir tipos de variable

lógico	convierte
`is.numeric()`	`as.numeric()`
`is.character()`	`as.character()`
`is.vector()`	`as.vector()`
`is.matrix()`	`as.matrix()`
`is.data.frame()`	`as.data.frame()`

```
a <- c( 1, 2, 3 )
is.numeric( a )
```

```
[1] TRUE
```

Convertir tipos de variable

lógico	convierte
`is.numeric()`	`as.numeric()`
`is.character()`	`as.character()`
`is.vector()`	`as.vector()`
`is.matrix()`	`as.matrix()`
`is.data.frame()`	`as.data.frame()`

```
a <- c( 1, 2, 3 )
is.numeric( a )
```

```
[1] TRUE
```

```
b <- as.character( a )
b
```

Convertir tipos de variable

lógico	convierte
`is.numeric()`	`as.numeric()`
`is.character()`	`as.character()`
`is.vector()`	`as.vector()`
`is.matrix()`	`as.matrix()`
`is.data.frame()`	`as.data.frame()`

```
a <- c( 1, 2, 3 )
is.numeric( a )
```

```
[1] TRUE
```

```
b <- as.character( a )
b
```

```
[1] "1" "2" "3"
```

Contenido

1 Introducción

2 Elementos en R

3 Objetos en R

4 Indexado

5 Valores perdidos

6 Importar datos en R

Importar datos en R

Algunas consideraciones para importar datos

1. La función read.table()
2. Función read.csv()
3. Library(foreign) *## importa datos de spss, minitab, stata, etc*
4. Guardar datos en el directorio: función write.table() , write.csv()

Importar datos en R

Antes de importar un archivo se debe comprobar:

1. La ruta o directorio donde reposa el archivo.
2. Identificar el nombre del archivo.
3. Identificar la extensión o formato del archivo a importar: .xls, .xlsx, .txt, .csv, .sav, .mpj
4. Las columnas y las filas son consistentes, es decir, no tener celdas combinadas de Excel, filas o columnas en blanco.
5. Si las columnas o filas están etiquetadas, que no haya símbolos extraños, por preferencia omitir: $, Ñ, %, ' (tildes).
6. Si hay datos perdidos, que su codificación sea consistente: NA.

Importar datos en R

La función más importante para importar datos: read.table() automáticamente convierte los datos en un dataframe.

```
df <- read.table(file,
header = valor_logico,
sep = "delimitador",
dec = "signo del decimal")
```

- header TRUE/FALSE según la primera fila del fichero
- sep es el delimitador: el separador de campos + sep = ";", sep=".", sep=",",sep="\t" . .
- dec es el indicador del signo decimal + dec=".", dec=";". . .

Si el conjunto de datos se encuentra en el directorio de trabajo. Entonces Ej:

```
df<-read.table("Datos1.txt",header=T, sep="",dec=".")
```

Ejemplo

Lea el conjunto de datos en el archivo Base-1. Tenga en cuenta:

- El directorio
- La extensión del documento
- El separador
- El símbolo de decimales
- La cabecera (primera fila)

```
setwd("G:/Mi unidad/Clases/Univalle/2020-1/Clase 1/")
read.table("Base_1.csv",header=TRUE, sep=";",dec=",")
```

También puede utilizar funciones como read.csv()

Ejercicio 5

1. Guarda la base importada con el nombre "PibEsp"
2. Selecciona el valor total del PIB para Andalucía
3. Selecciona el valor del PIB en Andalucía correspondiente a la agricultura
4. Calcula el promedio de PIB por cada ciudad. ¿Quién tiene el mayor PIB?
5. Explore la función aggregate() con la ayuda de R, Nos puede ayudar en algo?
6. Utilice la función aggregate(), guarde el data.frame y renombre las columnas del data.frame
7. Explore la función boxplot() con la ayuda de R, Nos puede ayudar en algo?
8. Realice digramas de caja (boxplot) del PIB por cada ciudad en un solo gráfico
9. Guarde el data.frame del punto 6. con la función write.table()

Importar datos en R

Una forma sencilla de cargar un conjunto de datos es mediante la opción de copiar y pegar, sin embargo para que R asimile dicha información es necesario ejecutar un comando.

Por ejemplo: Genere un conjunto de datos en Excel, luego copie los datos con Ctrl + C y luego en R ejecute el siguiente código:

```
datos <- read.delim("clipboard")
datos
```

Importar datos en R

Existe un paquete clasificado como "paquete recomendado" que permite una comunicación fluida entre programas para el análisis de datos; foreign() .

El paquete foreign incorpora varias funciones para la lectura de datos provenientes de SPSS (.sav formato), SAS, Epi-Infor (.rec), Stata (.dta), Minitab ()

A veces los archivos provenientes de Stata tienen problemas por la versión del mismo. Recomiendo diferentes paquetes como

- read.dta13() de la librería readstata13
- read_dta() de la librería haven
- read.spss() de la librería foreign para archivos de SPSS

Preguntas?

Gracias!! ,

Jr.

orlando.joaqui@correounivalle.edu.co