

Fundamentos de Lenguaje R

Autor: Carlos Carreño
ccarrenovi@icloud.com

Introducción al Lenguaje R

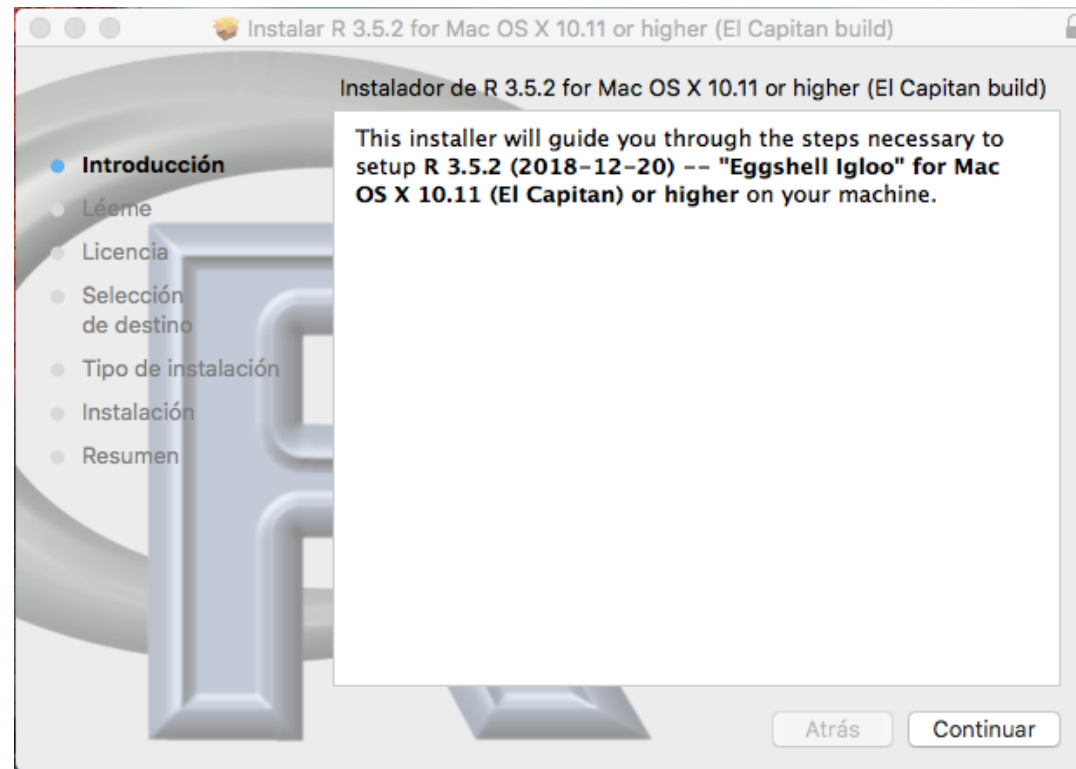
- Pasos Iniciales
- Tipos de Datos Disponibles
- Operaciones Básicas
- R Commander

Pasos Iniciales

Introducción al lenguaje R

Instalación

- R-project.org
- x-quartz



Configurando el Juego de Caracteres

- Ejecutar

1. Open Terminal

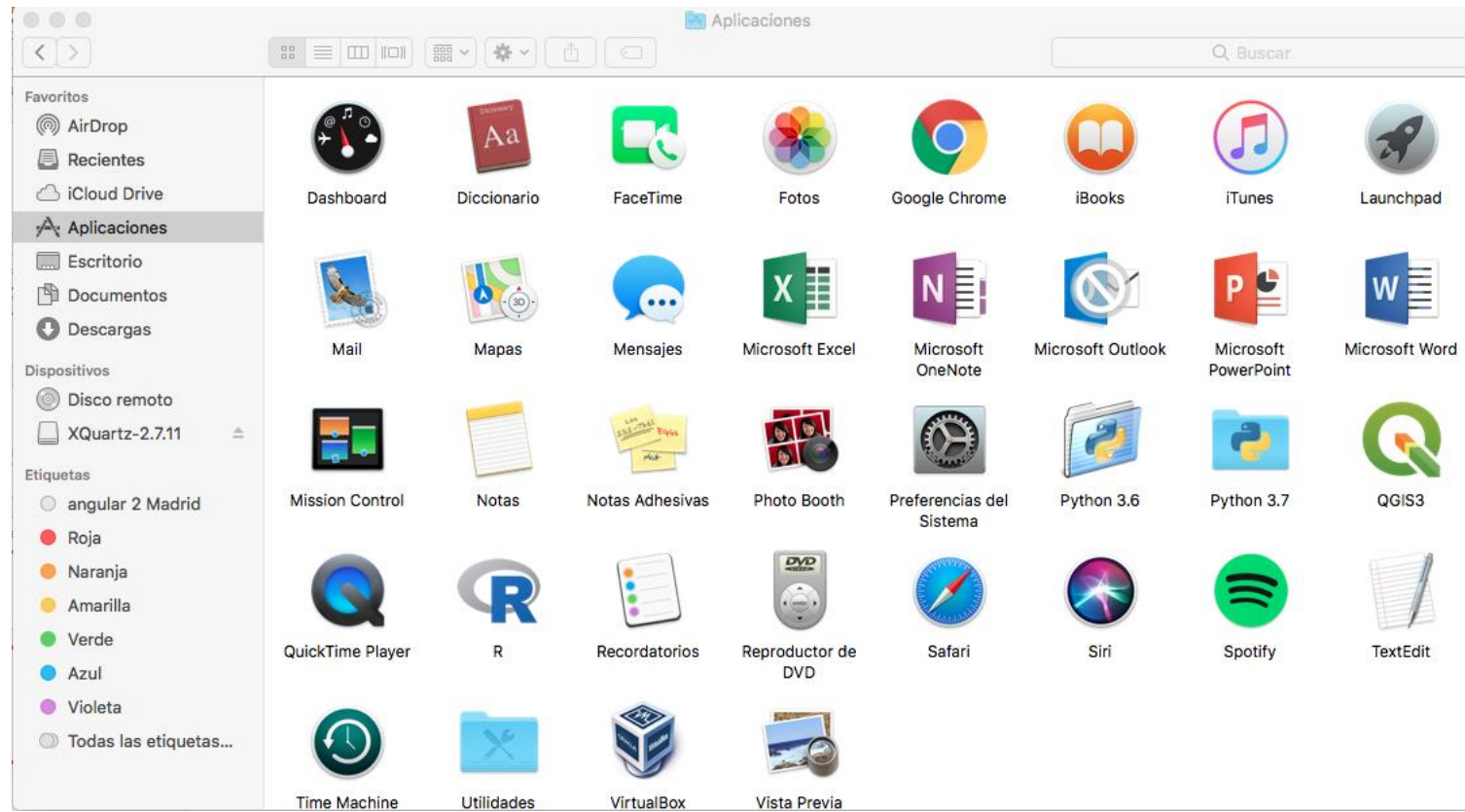
2. Write or paste in: `defaults write org.R-project.R force.LANG en_US.UTF-8`

3. Close Terminal

4. Start R

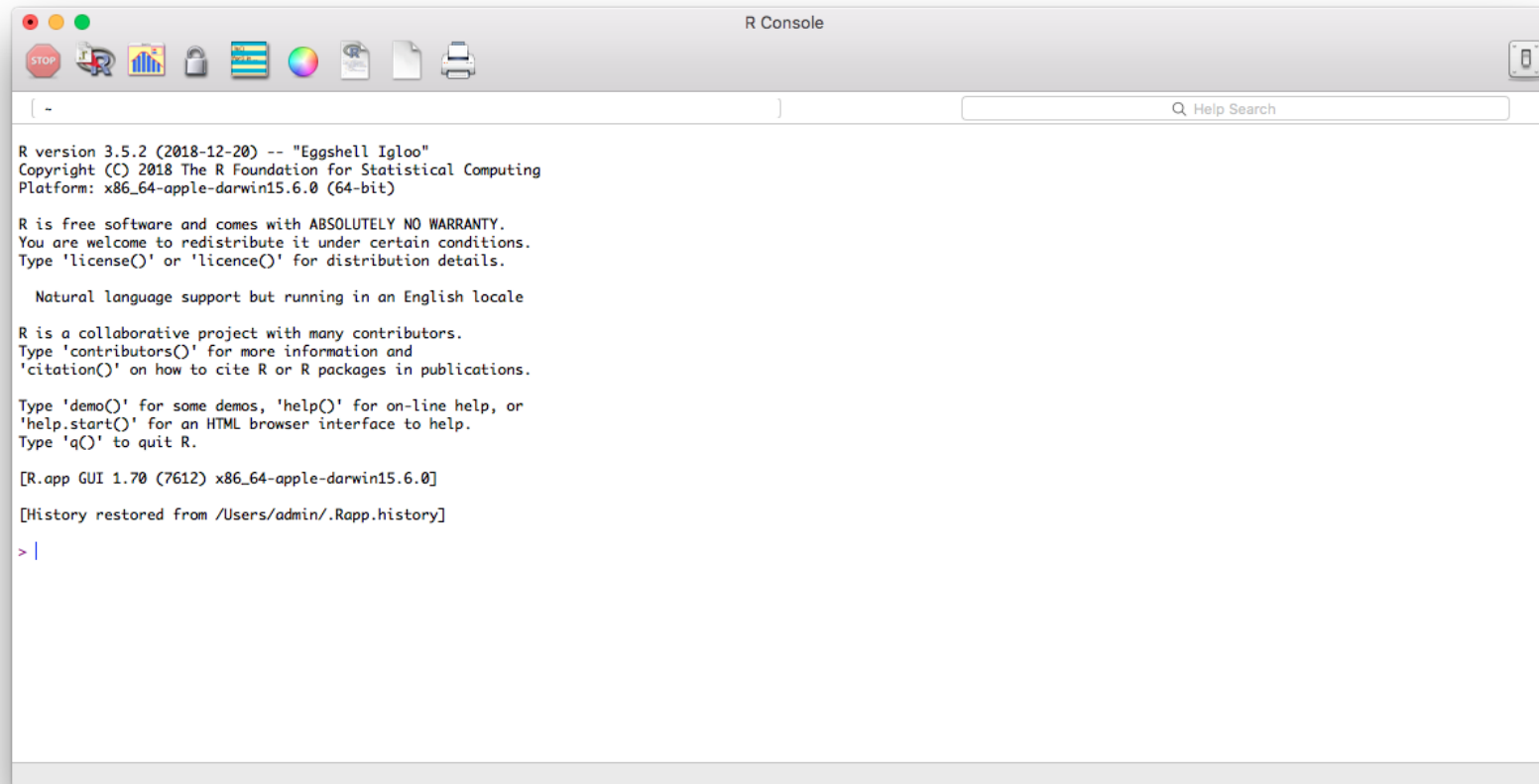
Iniciando R

- Abre la aplicación



Ventana de R

- Consola R: Desde aquí se puede ingresar cualquier comando.



```
R version 3.5.2 (2018-12-20) -- "Eggshell Igloo"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin15.6.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

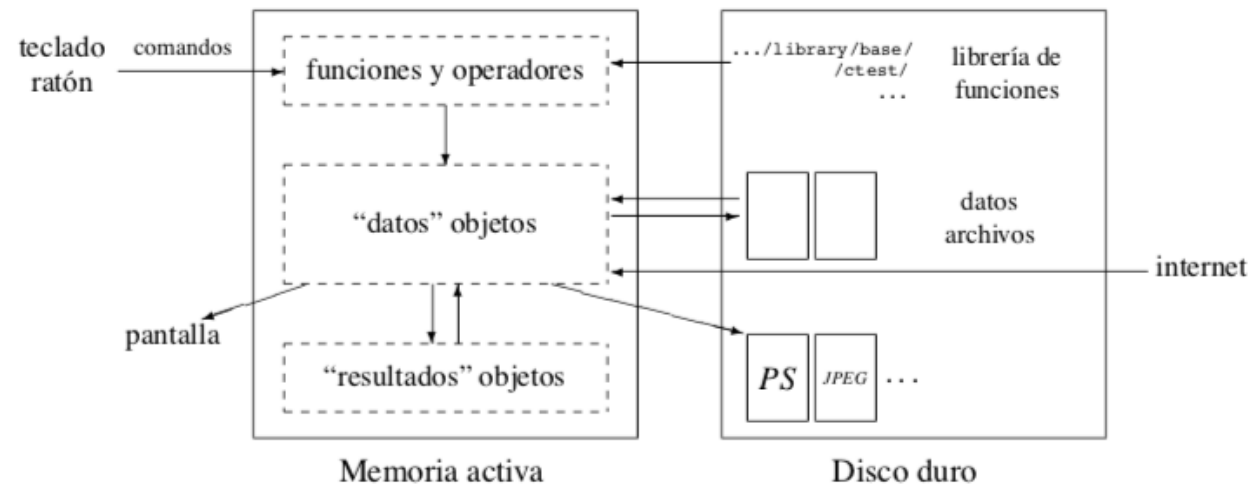
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[R.app GUI 1.70 (7612) x86_64-apple-darwin15.6.0]
[History restored from /Users/admin/.Rapp.history]
> |
```

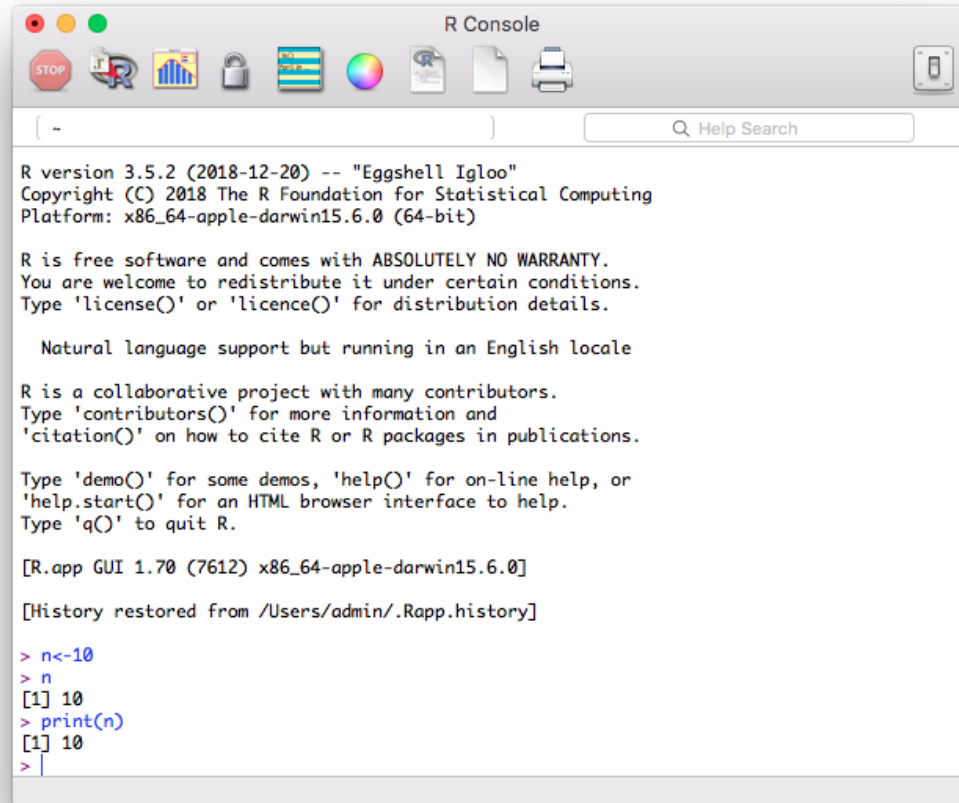
Como funciona R

- R es orientado a objeto, esto quiere decir que el resultado se guarda en memoria en un objeto.
- El nombre del objeto debe empezar con una letra, puede contener dígitos y punto(.).



Ejemplo

- Asignación de valores a objetos



```
R version 3.5.2 (2018-12-20) -- "Eggshell Igloo"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin15.6.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

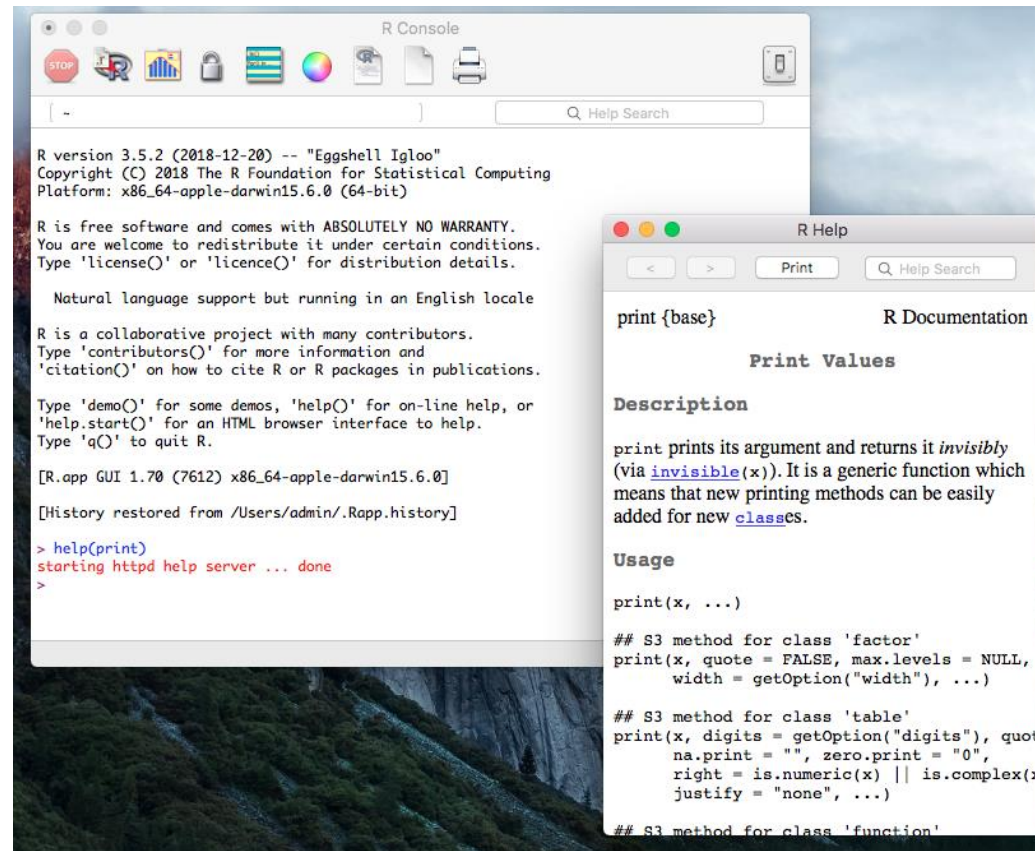
[R.app GUI 1.70 (7612) x86_64-apple-darwin15.6.0]

[History restored from /Users/admin/.Rapp.history]

> n<-10
> n
[1] 10
> print(n)
[1] 10
> |
```

Ayuda en línea

- `help()`

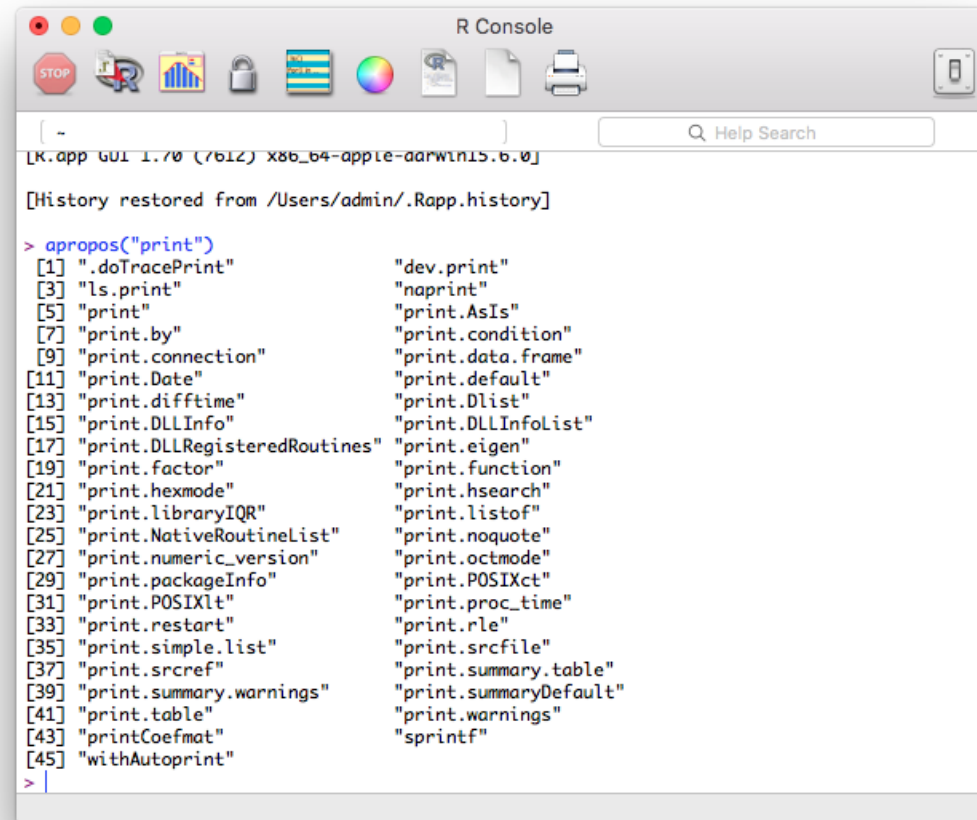


... continua

- `help.start()`
- `?print`
- `apropos("cadena")`

... continua

- `apropos("print")`



```
R Console
[~]
[Help Search]

[R.app GUI 1.70 (7612) x86_64-apple-darwin15.6.0]
[History restored from /Users/admin/.Rapp.history]

> apropos("print")
[1] ".doTracePrint"      "dev.print"
[3] "ls.print"           "naprint"
[5] "print"              "print.AsIs"
[7] "print.by"           "print.condition"
[9] "print.connection"   "print.data.frame"
[11] "print.Date"         "print.default"
[13] "print.difftime"     "print.Dlist"
[15] "print.DLLInfo"      "print.DLLInfoList"
[17] "print.DLLRegisteredRoutines" "print.eigen"
[19] "print.factor"       "print.function"
[21] "print.hexmode"      "print.hsearch"
[23] "print.libraryIQR"   "print.listof"
[25] "print.NativeRoutineList" "print.noquote"
[27] "print.numeric_version" "print.octmode"
[29] "print.packageInfo"  "print.POSIXct"
[31] "print.POSIXlt"      "print.proc_time"
[33] "print.restart"      "print.rle"
[35] "print.simple.list"  "print.srcfile"
[37] "print.srcfile"      "print.summary.table"
[39] "print.summary.warnings" "print.summaryDefault"
[41] "print.table"        "print.warnings"
[43] "printCoefmat"       "sprintf"
[45] "withAutoprint"

> |
```

Tipos de Datos Disponibles

Introducción al lenguaje R

Tipos básicos de datos

- En R (como en otros lenguajes de programación) hay dos clases fundamentales de datos: **numéricos** y **alfanuméricos** (tipo lógico, entero y complejo).

```
> x=1
> class(x)
[1] "numeric"
> y="1"
> class(y)
[1] "character"
> b=3>5
> class(b)
[1] "logical"
> b
[1] FALSE
> z=3+2i
> class(z)
[1] "complex"
> entero = integer(8)
> class(entero)
[1] "integer"
```

Tipos de Objetos

- Tabla resumen de tipo de objetos

objeto	tipos	varios tipos posibles en el mismo objeto?
vector	numérico, caracter, complejo o lógico	No
factor	numérico o caracter	No
arreglo	numérico, caracter, complejo o lógico	No
matriz	numérico, caracter, complejo o lógico	No
data.frame	numérico, caracter, complejo o lógico	Si
ts	numérico, caracter, complejo o lógico	Si
lista	numérico, caracter, complejo, lógico función, expresión, ...	Si

Valores Numéricos No Finitos

- Inf y -Inf

```
> x=5/0
> x
[1] Inf
> t=-6/0
> t
[1] -Inf
>
```


Operaciones Básicas

Introducción al lenguaje R

Operadores Aritméticos

- + adición
- - substracción
- * multiplicación
- / división
- ^ potencia
- % módulo o residuo
- %/% división de enteros

Operadores de Comparación

- > mayor que
- >= mayor igual que
- < menor que
- <= menor igual que
- == igual
- != diferente
- eval

Operadores Lógicos

- `! x` NO lógico. Negación
- `x & y` Y lógico Evalúa solo la primera
- `x && y` id.. Evalúa ambas expresiones
- `x | y` O lógico. Evalúa solo la primera
- `x || y` id. Evalúa ambas expresiones
- `xor(x, y)` O exclusivo, EXCLUSIVIDAD

Vectores en Lenguaje R

- Creación de Vectores
- Nombramiento de Vectores
- Selección de Vectores
- Operaciones Aritméticas con Vectores

Creación de Vectores

Vectores en lenguaje R

Vector

- Colección ordenada de datos o elementos del mismo tipo

```
> z=c('a','b','c')
> x=c(1,2,3,4)
> x
[1] 1 2 3 4
> z
[1] "a" "b" "c"
>
```

Creando Vectores

- Los vectores se pueden crear de varias maneras, asignando valores, usando secuencias o como resultado de alguna operación

```
> i<-c(1,2,3)
> i
[1] 1 2 3
> x=1:10
> x
[1] 1 2 3 4 5 6 7 8 9 10
> z=x^2
> z
[1] 1 4 9 16 25 36 49 64 81 100
>
```


Nombramiento de Vectores

Vectores en lenguaje R

Nombres de Vectores

- Los nombres validos para un objeto son combinaciones de letras, números, y el punto (".").
- Los nombres no pueden empezar con un número. R es "case-sensitive". **x** != **X**.
Hay nombres reservados ("function", "if", etc).

Consideraciones

- El uso del "." es diferente al de C++.
- Sobre el uso de ";". No es bueno abusar, porque hace el código *muy difícil* de leer. (Pero en estas transparencias, si no lo usara ocuparíamos muchísimo espacio).
- Mejor evitar nombres que R usa (ej., "c") (pero no es dramático si nos confundimos: podemos arreglarlo). **c <- 4; x <- c(3, 8); x; rm(c); c(7, 9)**
- Las asignaciones se hacen con "<-", y es buen estilo el rodear "<-" por un espacio a cada lado.

Estilos

x<-1:5 # Mal estilo

x <- 1:5 # Mucho mejor.

Selección en Vectores

Vectores en lenguaje R

Selección o Indexación

- Una de las gran virtudes de R es la flexibilidad en el acceso a los elementos de los vectores.
- **`x <- 1:5; x[1]; x[3]`**

Tipos de Indexación

Podemos indexar un vector de cuatro formas:

- Un vector lógico.
Un vector de enteros (integers) positivos.
- Un vector de enteros negativos.
- Un vector de cadenas de caracteres (character strings)

Otros Tipos de Indexación

- También, por supuesto, podemos indexar un vector usando cualquier expresión o función que resulte en una de las cuatro anteriores.
- No solo podemos acceder (devolver) el/los valores, sino también asignarlos:
`x[c(1, 3)] <- c(25, 79); x[x > 3] <- 97`

- "NA" es el código de "Not available".
- **`v <- c(1:3, NA)`**
- **`is.na(v);`**
- **`which(is.na(v))`**
- **`v == NA` # No funciona! Por qué?**
- Sustituir NA por, p.ej., 0: **`v[is.na(v)] <- 0.`**
- Infinito y NaN (not a number). Son distintos de NA.
- **`5/0; -5/0; 0/0`**
- **`is.infinite(-5/0); is.nan(0/0); is.na(5/0)`**
- **`xna <- c(1, 2, 3, NA, 4); mean(xna) mean(xna, na.rm = TRUE)`**

Tratamiento de NA

- Ejemplo: Eliminando NA en las operaciones

```
> xna=c(1,2,3,NA,4)
> xna
[1] 1 2 3 NA 4
> mean(xna)
[1] NA
> mean(xna,na.rm=TRUE)
[1] 2.5
>
```

Operaciones Aritméticas con Vectores

Vectores en lenguaje R

Operaciones Aritméticas

- $+$, $-$, $*$, $/$, \wedge , $\%\%$, $\%/ \%$
- Se pueden adicionar vectores con diferentes longitudes; en este caso el vector mas corto se recicla

```
> x<-1:4
> x
[1] 1 2 3 4
> y<-1:2
> y
[1] 1 2
> z<-x+y
> z
[1] 2 4 4 6
>
```

Multiplicando un Valor a los Elementos del Vector

- Si queremos agregar (o multiplicar) un mismo valor a todos los elementos de un vector:

```
> x <- 1:4
```

```
> a <- 10
```

```
> z <- a * x
```

```
> z
```

```
[1] 10 20 30 40
```

Operaciones de Conjunto

- `x <- 1:5; y <- c(1, 3, 7:10)`
- `union(x, y)`
- `intersect(x, y)`
- `setdiff(y, x)`
- `v <- c("bcA1", "bcA2", "bIX1")`
- `w <- c("bcA2", "xA3")`
- `union(v, w)`
- `intersect(v, w)`
- `setdiff(v, w)`

Generación de Secuencias

- `x <- c(1, 2, 3, 4, 5)`
- `x <- 1:10; y <- -5:3`
 - `x <- seq(from = 2, to = 18, by = 2)`
 - `x <- seq(from = 2, to = 18, length = 30)`
 - `x <- 1:10; y <- seq(along(x))`
 - `z2 <- c(1:5, 7:10, seq(from = -7, to = 5, by = 2))`

Generación de Secuencias Aleatorias

Generación de números aleatorios con parámetros

```
> rnorm(10)
```

```
> rnorm(10, mean = 13, sd = 18) > runif(15)
```

```
> runif(8, min = 3, max = 59)
```


Replicas

- Generando replicas

```
> x<-rep(1,5)
> x
[1] 1 1 1 1 1
> y<-rep(1:3, rep(5,3))
> y
[1] 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3
> |
```

Funciones con Vectores

<code>sum(x)</code>	suma de los elementos de <code>x</code>
<code>prod(x)</code>	producto de los elementos de <code>x</code>
<code>max(x)</code>	valor máximo en el objeto <code>x</code>
<code>min(x)</code>	valor mínimo en el objeto <code>x</code>
<code>which.max(x)</code>	devuelve el índice del elemento máximo de <code>x</code>
<code>which.min(x)</code>	devuelve el índice del elemento mínimo de <code>x</code>
<code>range(x)</code>	rango de <code>x</code> o <code>c(min(x), max(x))</code>
<code>length(x)</code>	número de elementos en <code>x</code>
<code>mean(x)</code>	promedio de los elementos de <code>x</code>
<code>median(x)</code>	mediana de los elementos de <code>x</code>
<code>var(x)</code> o <code>cov(x)</code>	varianza de los elementos de <code>x</code> (calculada en $n - 1$); si <code>x</code> es una matriz o un marco de datos, se calcula la matriz de varianza-covarianza
<code>cor(x)</code>	matriz de correlación de <code>x</code> si es una matriz o un marco de datos (1 si <code>x</code> es un vector)
<code>var(x, y)</code> o <code>cov(x, y)</code>	covarianza entre <code>x</code> y <code>y</code> , o entre las columnas de <code>x</code> y <code>y</code> si son matrices o marcos de datos
<code>cor(x, y)</code>	correlación lineal entre <code>x</code> y <code>y</code> , o la matriz de correlación si <code>x</code> y <code>y</code> son matrices o marcos de datos

Funciones con Vectores

<code>round(x, n)</code>	redondea los elementos de <code>x</code> a <code>n</code> cifras decimales
<code>rev(x)</code>	invierte el orden de los elementos en <code>x</code>
<code>sort(x)</code>	ordena los elementos de <code>x</code> en orden ascendente; para hacerlo en orden descendente: <code>rev(sort(x))</code>
<code>rank(x)</code>	alinea los elementos de <code>x</code>
<code>log(x, base)</code>	calcula el logaritmo de <code>x</code> en base " <code>base</code> "
<code>scale(x)</code>	si <code>x</code> es una matriz, centra y reduce los datos; si se desea centrar solamente utilizar <code>scale=FALSE</code> , para reducir solamente usar <code>center=FALSE</code> (por defecto <code>center=TRUE, scale=TRUE</code>)
<code>pmin(x,y,...)</code>	un vector en el que el <i>i</i> avo elemento es el mínimo de <code>x[i]</code> , <code>y[i]</code> ,...
<code>pmax(x,y,...)</code>	igual que el anterior pero para el máximo
<code>cumsum(x)</code>	un vector en el que el <i>i</i> avo elemento es la suma desde <code>x[1]</code> a <code>x[i]</code>
<code>cumprod(x)</code>	igual que el anterior pero para el producto
<code>cummin(x)</code>	igual que el anterior pero para el mínimo
<code>cummax(x)</code>	igual que el anterior pero para el máximo
<code>match(x, y)</code>	devuelve un vector de la misma longitud que <code>x</code> con los elementos de <code>x</code> que están en <code>y</code> (NA si no)
<code>which(x == a)</code>	devuelve un vector de los índices de <code>x</code> si la operación es (TRUE) (en este ejemplo, los valores de <code>i</code> para los cuales <code>x[i] == a</code>). El argumento de esta función debe ser una variable de tipo lógico
<code>choose(n, k)</code>	calcula el número de combinaciones de <code>k</code> eventos en <code>n</code> repeticiones $= n! / [(n - k)!k!]$
<code>na.omit(x)</code>	elimina las observaciones con datos ausentes (NA) (elimina la fila correspondiente si <code>x</code> es una matriz o un marco de datos)
<code>na.fail(x)</code>	devuelve un mensaje de error si <code>x</code> contiene por lo menos un NA
<code>unique(x)</code>	si <code>x</code> es un vector o un marco de datos, devuelve un objeto similar pero suprimiendo elementos duplicados
<code>table(x)</code>	devuelve una tabla con el número de diferentes valores de <code>x</code> (típicamente para enteros o factores)
<code>subset(x, ...)</code>	devuelve una selección de <code>x</code> con respecto al criterio (... , típicamente comparaciones: <code>x\$V1 < 10</code>); si <code>x</code> es un marco de datos, la opción <code>select</code> proporciona las variables que se mantienen (o se ignoran con <code>-</code>)
<code>sample(x, size)</code>	remuestra al azar y sin reemplazo <code>size</code> elementos en el vector <code>x</code> ; la opción <code>replace = TRUE</code> permite remuestrear con reemplazo

Matrices en Lenguaje R

- Creación de Matrices
- Operaciones Básicas con Matrices
- Rbind y Cbind

Creación de Matrices

Matrices en lenguaje R

Matriz

- Una matriz es realmente un vector con un atributo adicional (dim) el cual a su vez es un vector numérico de longitud 2, que define el número de filas y columnas de la matriz. Una matriz se puede crear con la función matrix:

```
matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)
```

- La opción byrow indica si los valores en data deben llenar las columnas sucesivamente (por defecto) o las filas (if TRUE). La opción dimnames permite asignar nombres a las filas y columnas.

Creando la Matriz

- Función matrix

```
> matrix(data=5, nr=2, nc=2)
      [,1] [,2]
[1,]    5    5
[2,]    5    5
> matrix(1:6, 2, 3)
      [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> matrix(1:6, 2, 3, byrow=TRUE)
      [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
```

...continua

Otra manera de crear una matriz es dando los valores apropiados al atributo dim (que inicialmente tiene valor NULL):

```
> x <- 1:15
> x
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
> dim(x)
NULL
> dim(x) <- c(5, 3)
> x
      [,1] [,2] [,3]
[1,]    1    6   11
[2,]    2    7   12
[3,]    3    8   13
[4,]    4    9   14
[5,]    5   10   15
```


Operaciones Básicas con Matrices

Matrices en lenguaje R

Operaciones Matemáticas

- Creación de una matriz con números aleatorios

```
> v <- sample(1:20, 20, replace = TRUE)
> m <- matrix(v, nrow = 4, ncol = 5, byrow = FALSE)
> m
```

	[,1]	[,2]	[,3]	[,4]	[,5]
[1,]	13	19	12	2	11
[2,]	2	12	8	2	3
[3,]	10	15	4	11	1
[4,]	17	9	17	14	14

```
>
```

... continua

- Multiplicando un valor a los elementos de la Matriz
- Exponenciando los elementos de la Matriz

```
> m * 2
```

	[,1]	[,2]	[,3]	[,4]	[,5]
[1,]	26	38	24	4	22
[2,]	4	24	16	4	6
[3,]	20	30	8	22	2
[4,]	34	18	34	28	28

```
> m ^ 2
```

	[,1]	[,2]	[,3]	[,4]	[,5]
[1,]	169	361	144	4	121
[2,]	4	144	64	4	9
[3,]	100	225	16	121	1
[4,]	289	81	289	196	196

```
> |
```

... continua

- Multiplicando Matrices

```
> m * m
      [,1] [,2] [,3] [,4] [,5]
[1,]  169  361  144    4  121
[2,]    4  144   64    4    9
[3,]  100  225   16  121    1
[4,]  289   81  289  196  196
> |
```

Operaciones con Matrices

- Suma

```
> m1 <- matrix(sample(1:9, 9, replace = TRUE), nrow = 3, ncol = 3, byrow = FALSE)
> m2 <- matrix(sample(1:9, 9, replace = TRUE), nrow = 3, ncol = 3, byrow = FALSE)
> m1
      [,1] [,2] [,3]
[1,]    5    8    8
[2,]    6    3    5
[3,]    3    7    5
> m2
      [,1] [,2] [,3]
[1,]    8    1    6
[2,]    3    2    5
[3,]    2    1    9
> m3 <- m1 + m2
> m3
      [,1] [,2] [,3]
[1,]   13    9   14
[2,]    9    5   10
[3,]    5    8   14
> |
```

... continua

- Multiplicación

```
> m4 <- m1 * m2
> m4
      [,1] [,2] [,3]
[1,]   40    8   48
[2,]   18    6   25
[3,]    6    7   45
>
```

Operaciones Estadísticas

- Promedios por columna y por fila

```
> m
      [,1] [,2] [,3] [,4] [,5]
[1,]   13   19   12    2   11
[2,]    2   12    8    2    3
[3,]   10   15    4   11    1
[4,]   17    9   17   14   14
> colMeans(m)
[1] 10.50 13.75 10.25  7.25  7.25
> rowMeans(m)
[1] 11.4  5.4  8.2 14.2
>
```

Rbind y Cbind

Matrices en lenguaje R

Usando Rbind y Cbind

- Agregando un columna con Cbind

```
> m1
      [,1] [,2] [,3]
[1,]    5    8    8
[2,]    6    3    5
[3,]    3    7    5
> m5 <- cbind(m1,c(1:3))
> m5
      [,1] [,2] [,3] [,4]
[1,]    5    8    8    1
[2,]    6    3    5    2
[3,]    3    7    5    3
>
```

Usando Rbind y Cbind

- Agregando una fila con Rbind

```
> m1
      [,1] [,2] [,3]
[1,]    5    8    8
[2,]    6    3    5
[3,]    3    7    5
> m6 <- rbind(m1,c(1:3))
> m6
      [,1] [,2] [,3]
[1,]    5    8    8
[2,]    6    3    5
[3,]    3    7    5
[4,]    1    2    3
>
```

Es una matrix?

- Preguntar si una variable es una matriz

```
> is.matrix(m6)
[1] TRUE
> |
```

Marcos de Datos en Lenguaje R

- Introducción a los Marcos de Datos
- Operaciones Básicas con Marcos de Datos
- Acceso a Variables en un Marco de Datos
- Comandos attach y detach
- Combinación y Ordenamiento de Marcos de Datos
 - rbind
 - merge

Dataframes o Marco de Datos

- Los *dataframes* son una clase de objetos especial en R. Normalmente, cuando se realiza un estudio estadístico sobre los sujetos u objetos de una muestra, la información se organiza precisamente en un *dataframe*: una hoja de datos, en los que cada fila corresponde a un sujeto y cada columna a una variable. La estructura de un `data.frame` es muy similar a la de una matriz. La diferencia es que una matriz sólo admite valores numéricos, mientras que en un *dataframe* podemos incluir también datos alfanuméricos.

Ejemplo de Dataframe

- Dataframe con tres variables

##		edad	tiempo	sexo
##	1	22	14.21	M
##	2	34	10.36	H
##	3	29	11.89	H
##	4	25	13.81	M
##	5	30	12.03	M
##	6	33	10.99	H
##	7	31	12.48	M
##	8	27	13.37	M
##	9	25	12.29	H
##	10	25	11.92	H

Operaciones Básicas con Marcos de Datos

Marcos de Datos en lenguaje R

Creación de un Dataframe

- El siguiente es un dataframe sobre una muestra de 10 personas, para cada una de las cuales se ha registrado su edad, sexo y tiempo en minutos que estuvo hablando por teléfono el día antes de la encuesta

```
> edad <- c(22, 34, 29, 25, 30, 33, 31, 27, 25, 25)
> tiempo <- c(14.21, 10.36, 11.89, 13.81, 12.03, 10.99, 12.48, 13.37, 12.29, 11.92)
> sexo <- c("M", "H", "H", "M", "M", "H", "M", "M", "H", "H")
> misDatos <- data.frame(edad, tiempo, sexo)
> misDatos
```

	edad	tiempo	sexo
1	22	14.21	M
2	34	10.36	H
3	29	11.89	H
4	25	13.81	M
5	30	12.03	M
6	33	10.99	H
7	31	12.48	M
8	27	13.37	M
9	25	12.29	H
10	25	11.92	H

```
> |
```


Estructura del Dataframe

- Revisando la estructura de los datos

```
> str(misDatos)
'data.frame':  10 obs. of  3 variables:
 $ edad  : num  22 34 29 25 30 33 31 27 25 25
 $ tiempo: num  14.2 10.4 11.9 13.8 12 ...
 $ sexo  : Factor w/ 2 levels "H","M": 2 1 1 2 2 1 2 2 1 1
>
```

Variables del Dataframe

- Visualizando las variables contenidas en el Dataframe

```
> misDatos
  edad tiempo sexo
1   22  14.21    M
2   34  10.36    H
3   29  11.89    H
4   25  13.81    M
5   30  12.03    M
6   33  10.99    H
7   31  12.48    M
8   27  13.37    M
9   25  12.29    H
10  25  11.92    H
> str(misDatos)
'data.frame':  10 obs. of  3 variables:
 $ edad  : num  22 34 29 25 30 33 31 27 25 25
 $ tiempo: num  14.2 10.4 11.9 13.8 12 ...
 $ sexo  : Factor w/ 2 levels "H","M": 2 1 1 2 2 1 2 2 1 1
> names(misDatos)
[1] "edad" "tiempo" "sexo"
>
```

Acceso a Variables en un Marco de Datos

Marcos de Datos en lenguaje R

Acceso a Variables en el Dataframe

- El acceso a los datos que se encuentran en un data.frame es muy similar al acceso a los datos de una matriz que ya vimos en la sección anterior. Sin embargo, para los dataframes R dispone de algunas funciones que facilitan la tarea de seleccionar o filtrar datos

Visualizando Filas

- Visualizando los datos entre las filas 3 y 6

```
> misDatos
  edad tiempo sexo
1   22  14.21    M
2   34  10.36    H
3   29  11.89    H
4   25  13.81    M
5   30  12.03    M
6   33  10.99    H
7   31  12.48    M
8   27  13.37    M
9   25  12.29    H
10  25  11.92    H
> misDatos[3:6,]
  edad tiempo sexo
3   29  11.89    H
4   25  13.81    M
5   30  12.03    M
6   33  10.99    H
>
```

Seleccionando los Datos de una Columna

- Visualizando los datos de la Edad

```
> misDatos[,1]  
[1] 22 34 29 25 30 33 31 27 25 25  
>
```

- Los siguientes comandos dan el mismo resultado:
 - `misDatos[,1]`
 - `misDatos$edad`
 - `misDatos["edad"]`
 - `misDatos[["edad"]]`

Comandos attach y detach

Marcos de Datos en lenguaje R

Attach y detach

- El acceso a las variables dentro de un dataframe puede hacerse engorroso cuando hemos de escribir constantemente el nombre del dataframe (en particular si éste es muy largo). Para evitar este problema podemos utilizar el comando `attach()`, cuyo objetivo consiste básicamente en “*enganchar*” el contenido del dataframe al entorno donde R busca los nombres de variable; de esta forma se puede acceder directamente a las variables del dataframe por su nombre, sin necesidad de que éste tenga que ser precedido con el nombre del dataframe y el símbolo `$`; una vez que hayamos acabado nuestra tarea “*desenganchamos*” el dataframe con `detach()`

... continua

```
> misDatos
  edad tiempo sexo
1   22  14.21    M
2   34  10.36    H
3   29  11.89    H
4   25  13.81    M
5   30  12.03    M
6   33  10.99    H
7   31  12.48    M
8   27  13.37    M
9   25  12.29    H
10  25  11.92    H
> attach(misDatos)
The following objects are masked _by_ .GlobalEnv:

    edad, sexo, tiempo

> table(edad)
edad
22 25 27 29 30 31 33 34
 1  3  1  1  1  1  1  1
> mean(edad[sexo=="M"])
[1] 27
> detach(misDatos)
>
```

Combinación y Ordenamiento de Marcos de Datos

Marcos de Datos en lenguaje R

Combinación de Dataframes

- Si tenemos dos dataframes con la misma estructura (idénticas variables), pero distintos datos, podemos combinarlos pegando uno a continuación del otro mediante `rbind`

```
animales1 = data.frame(animal=c("vaca", "perro", "rana", "lagarto", "mosca", "jilguero"),  
                        clase=c("mamífero", "mamífero", "anfibio", "reptil", "insecto", "ave"))
```

```
animales2 = data.frame(animal=c("atún", "cocodrilo", "gato", "rana"), clase=c("pez", "reptil", "mamífero", "anfibio"))
```

... continua

- Combinamos los dos dataframes

```
> animales3 <- rbind(animales1, animales2)
> animales3
  animal  clase
1   vaca mamífero
2   perro mamífero
3    rana  anfibio
4 lagarto  reptil
5   mosca  insecto
6 jilguero    ave
7    atún    pez
8 cocodrilo  reptil
9     gato mamífero
10    rana  anfibio
>
```

Merge

- El comando rbind no controla la posible aparición de casos repetidos en los dos dataframes (podemos comprobar que la rana está repetida en el dataframe 'animales3'). La función merge() evita este problema; utilizando la opción all=TRUE ó all=FALSE (valor por defecto) se consigue que se muestren **todos los datos** de ambos dataframes, o solo aquellos que son comunes a ambos

... continua

- Usando el comando merge, con la opción all=TRUE

```
> animales5=merge(animales1,animales2,all=TRUE)
> animales5
  animal  clase
1 jilguero   ave
2  lagarto  reptil
3   mosca  insecto
4   perro mamífero
5    rana  anfibio
6   vaca mamífero
7   atún    pez
8 cocodrilo  reptil
9    gato mamífero
> |
```

Ordenamiento

- Para ordenar un dataframe tenemos que aplicar la función `order()` al elemento o elementos por el que queremos ordenar, y utilizar el resultado de esta función como índice del dataframe.

```
> ordenacion=order(animales1$animal)
> ordenacion
[1] 6 4 5 2 3 1
> animales1=animales1[ordenacion,]
> animales1
  animal  clase
6 jilguero   ave
4 lagarto  reptil
5  mosca  insecto
2  perro mamífero
3   rana  anfibio
1   vaca mamífero
> |
```

... continua

- En un sola línea de comando

```
> animales1=animales1[order(animales1$animal),]  
> animales1  
  animal  clase  
6 jilguero   ave  
4 lagarto  reptil  
5   mosca insecto  
2   perro mamífero  
3    rana  anfibio  
1   vaca mamífero  
> |
```


... continua

- Si queremos ordenar nuestro primer dataframe (misDatos) primero por edad y luego por tiempo utilizando el celular

```
> misDatos=misDatos[order(misDatos$edad,misDatos$tiempo),]
```

```
> misDatos
```

	edad	tiempo	sexo
1	22	14.21	M
10	25	11.92	H
9	25	12.29	H
4	25	13.81	M
8	27	13.37	M
3	29	11.89	H
5	30	12.03	M
7	31	12.48	M
6	33	10.99	H
2	34	10.36	H

```
>
```

Laboratorio

- Creando Dataframes en el entorno R
- Usando Dataframes en R

Gráficos en Lenguaje R

- Introducción a los Gráficos con R
- Operaciones Básicas
- Funciones Graficas de Alto Nivel
- Funciones Graficas de Bajo Nivel
- Funciones de localización e identificación de puntos

Introducción a los Gráficos con R

Gráficos en lenguaje R

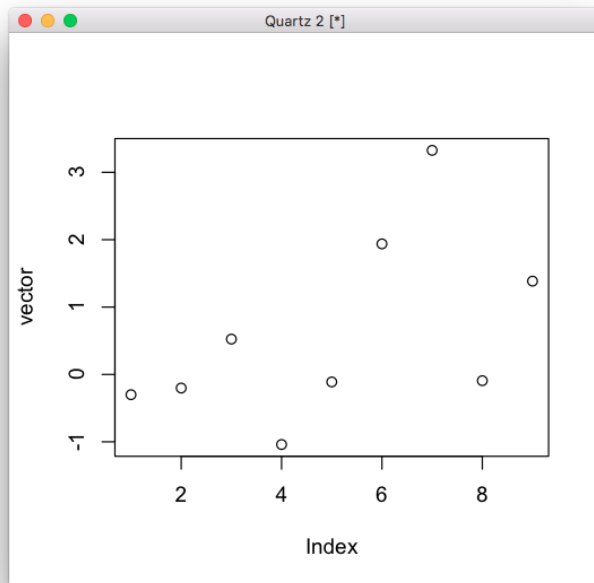
Introducción

- R dispone de múltiples funciones diseñadas para la representación gráfica de datos. Estas funciones se dividen en dos grandes grupos: funciones gráficas de **alto nivel** y de **bajo nivel**. La diferencia fundamental es que las funciones de alto nivel son las que generan gráficos completos, mientras que las de bajo nivel se limitan a añadir elementos a un gráfico existente (por tanto creado por una función de alto nivel).
- El paquete graphics (que se carga en memoria cada vez que arrancamos R) contiene un buen número de funciones de alto y bajo nivel para generar gráficos. Numerosos paquetes -plotrix, scatterplot3D, rgl, maps, shapes, ..., y sobre todo **ggplot2**- contienen muchísimas más funciones gráficas que mejoran y complementan las que vienen por defecto con R.

Creando el Grafico

- Datos y función

```
> vector <- c(rnorm(1:9))  
> vector  
[1] -0.30071273 -0.20204866  0.52456332 -1.04009713 -0.11121552  1.93779321  3.32597895 -0.09245835  
[9]  1.38482081  
> plot(vector)
```



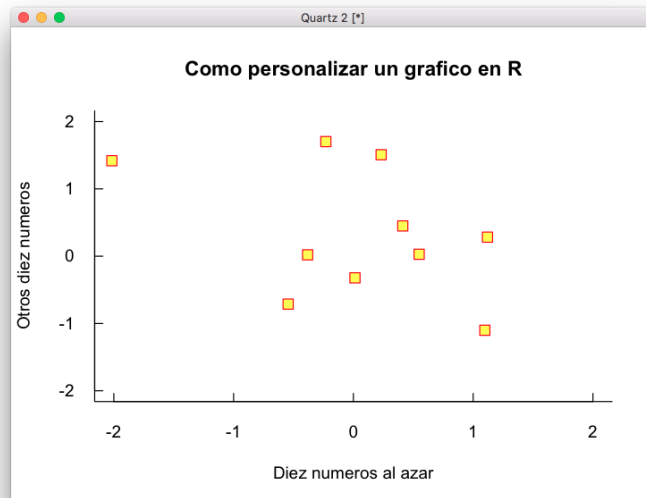
Operaciones Básicas

Gráficos en lenguaje R

Grafica Básica

- La función plot , permite realizar graficas básica, es parametrizable

```
> x <- rnorm(10)
> y <- rnorm(10)
> plot(x,y)
> plot(x, y, xlab="Diez numeros al azar", ylab="Otros diez numeros",
+       xlim=c(-2, 2), ylim=c(-2, 2), pch=22, col="red",
+       bg="yellow", bty="l", tcl=0.4,
+       main="Como personalizar un grafico en R", las=1, cex=1.5)
> |
```



Parámetros

- Parámetros de funciones graficas

adj	controla la justificación del texto (0 justificado a la izquierda, 0.5 centrado, 1 justificado a la derecha)
bg	especifica el color del fondo (ej.: <code>bg="red"</code> , <code>bg="blue"</code> ,...). La lista de los 657 colores disponibles se puede ver con <code>colors()</code>
bty	controla el tipo de caja que se dibuja alrededor del gráfico: <code>"o"</code> , <code>"l"</code> , <code>"7"</code> , <code>"c"</code> , <code>"ü"</code> o <code>"j"</code> (la caja se parece a su respectivo carácter); si <code>bty="n"</code> no se dibuja la caja
cex	un valor que controla el tamaño del texto y símbolos con respecto al valor por defecto; los siguientes parámetros tienen el mismo control para números en los ejes, <code>cex.axis</code> , títulos en los ejes, <code>cex.lab</code> , el título principal, <code>cex.main</code> y el subtítulo, <code>cex.sub</code>
col	controla el color de los símbolos; como en <code>cex</code> estos son: <code>col.axis</code> , <code>col.lab</code> , <code>col.main</code> y <code>col.sub</code>
font	un entero que controla el estilo del texto (1: normal, 2: cursiva, 3: negrilla, 4: negrilla cursiva); como en <code>cex</code> existen: <code>font.axis</code> , <code>font.lab</code> , <code>font.main</code> y <code>font.sub</code>
las	un entero que controla la orientación de los caracteres en los ejes (0: paralelo a los ejes, 1: horizontal, 2: perpendicular a los ejes, 3: vertical)
lty	un entero o carácter que controla el tipo de las líneas; (1: sólida, 2: quebrada, 3: punteada, 4: punto-línea, 5: línea larga-corta, 6: dos líneas cortas), o una secuencia de hasta 8 caracteres (entre <code>"0"</code> y <code>"9"</code>) que especifica alternativamente la longitud en puntos o píxeles, de los elementos dibujados y los blancos; por ejemplo <code>lty="44"</code> tendrá el mismo efecto que <code>lty=2</code>
lwd	un número que controla la anchura de las líneas
mar	un vector con 4 valores numéricos que controla el espacio entre los ejes y el borde de la gráfica en la forma <code>c(inferior, izquierda, superior, derecha)</code> ; los valores por defecto son <code>c(5.1, 4.1, 4.1, 2.1)</code>
mfc	un vector del tipo <code>c(nr,nc)</code> que divide la ventana gráfica como una matriz con <code>nr</code> filas y <code>nc</code> columnas; las gráficas se dibujan sucesivamente en las columnas (véase la sección 4.1.2)
mfrow	igual al anterior, pero las gráficas se dibujan en las filas (ver sección 4.1.2)
pch	controla el tipo de símbolo, ya sea un entero entre 1 y 25, o un carácter entre <code>" "</code> (Fig. 2)
ps	un entero que controla el tamaño (en puntos) de textos y símbolos
pty	un carácter que especifica el tipo de región a graficar, <code>"s"</code> : cuadrada, <code>"m"</code> : máxima
tck	un valor que especifica la longitud de los marcadores de eje como una fracción de la altura o anchura máxima del gráfico; si <code>tck=1</code> se dibuja una rejilla
tcl	un valor que especifica la longitud de los marcadores de eje como una fracción de la altura de una línea de texto (por defecto <code>tcl=-0.5</code>)
xaxt	si <code>xaxt="n"</code> el eje <code>x</code> se coloca pero no se muestra (util en conjunción con <code>axis(side=1, ...)</code>)
yaxt	si <code>yaxt="n"</code> el eje <code>y</code> se coloca pero no se muestra (util en conjunción con <code>axis(side=2, ...)</code>)

Funciones Graficas de Alto Nivel

Gráficos en lenguaje R

Función plot

- Esta función ofrece muchas variantes dependiendo del tipo de objeto al que se aplique. El caso más simple corresponde a la representación de dos variables x e y . En tal caso, `plot(x,y)` representa un diagrama de dispersión de puntos de y frente a x .

... continua

- Usando la función plot con dataframes

```
> edad <- c(22, 34, 29, 25, 30, 33, 31, 27, 25, 25)
> tiempo <- c(14.21, 10.36, 11.89, 13.81, 12.03, 10.99, 12.48, 13.37, 12.29, 11.92)
> sexo <- c("M", "H", "H", "M", "M", "H", "M", "M", "H", "H")
> misDatos <- data.frame(edad, tiempo, sexo)
> misDatos
```

	edad	tiempo	sexo
1	22	14.21	M
2	34	10.36	H
3	29	11.89	H
4	25	13.81	M
5	30	12.03	M
6	33	10.99	H
7	31	12.48	M
8	27	13.37	M
9	25	12.29	H
10	25	11.92	H

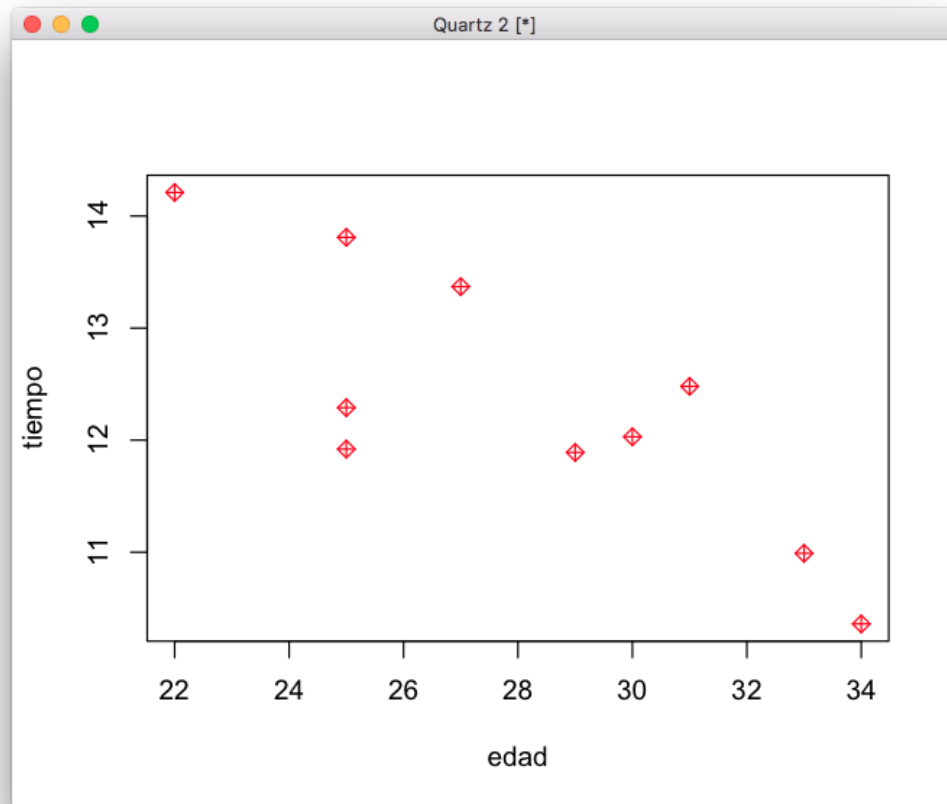
```
> attach(misDatos)
The following objects are masked _by_ .GlobalEnv:

    edad, sexo, tiempo

> plot(edad, tiempo, pch=9, col="red")
```

... continua

- Grafico de dispersión



Función hist()

- Esta función permite dibujar histogramas de frecuencias para variables continuas.

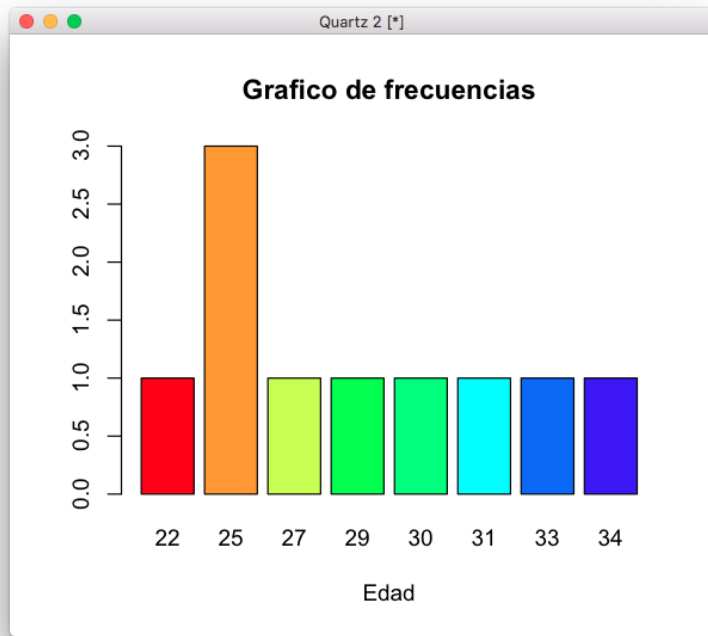
```
> hist(edad, col="green")
```



Función barplot

- Se utiliza para dibujar diagramas de barras

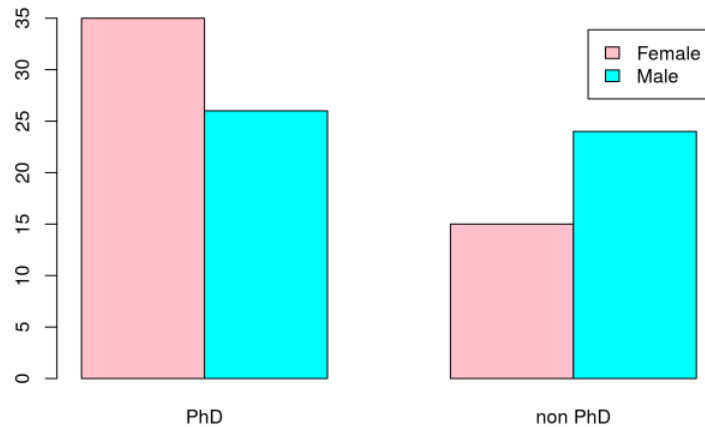
```
> barplot(table(edad),xlab="Edad",main="Grafico de frecuencias", col=rainbow(10))
```



Barplot y factores

- Es posible construir diagramas de barras por categorías; podemos, por ejemplo representar la frecuencia de doctores por sexo utilizando los datos del dataframe.

```
PhD=factor(PhD,levels=c(0,1),c("PhD","non PhD"))  
barplot(table(Gender,PhD),beside=TRUE,legend.text=TRUE,col=c("pink","cyan"))
```



Función pie

- Aporta la misma información que el diagrama de barras, pero en forma de diagrama de sectores

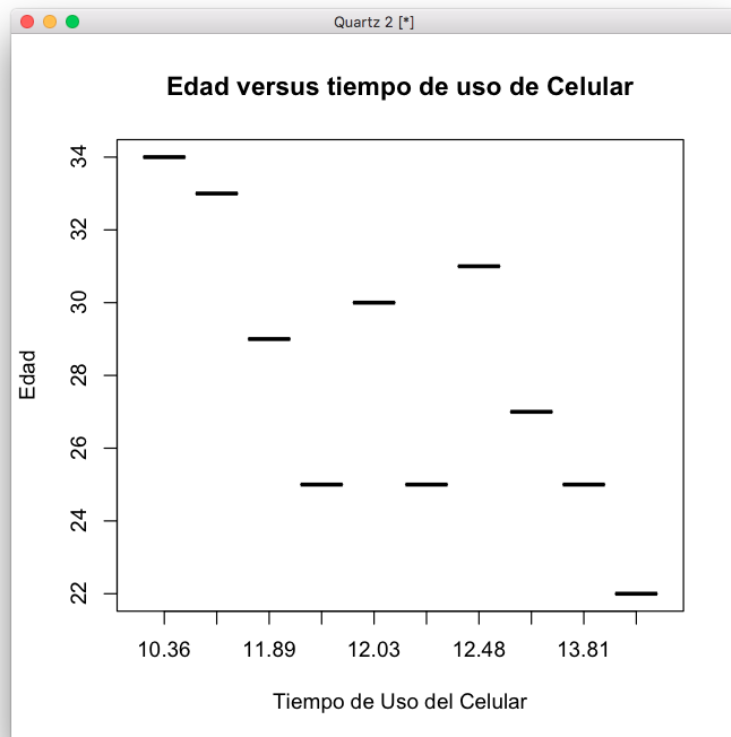
```
> pie(table(edad),main="Frecuencia de Edades de la Muestra")
```



Función boxplot

- Lleva a cabo la representación de gráficos de “*caja y bigote*”.

```
> boxplot(edad~tiempo,col="gold",xlab="Tiempo de Uso del Celular",ylab="Edad",  
+         main="Edad versus tiempo de uso de Celular")
```



Funciones Graficas de Bajo Nivel

Gráficos en lenguaje R

Funciones de Bajo Nivel

- Permiten añadir líneas, puntos, etiquetas... a un gráfico ya existente. Son de gran utilidad para completar un gráfico. Entre estas funciones cabe destacar:
 - `lines()`: Permite añadir líneas (uniendo puntos concretos) a una gráfica ya existente.
 - `abline()`: Añade líneas horizontales, verticales u oblicuas, indicando pendiente y ordenada.
 - `points()`: Permite añadir puntos.
 - `legend()`: Permite añadir una leyenda.
 - `text()`: Añade texto en las posiciones que se indiquen.
 - `grid()`: Añade una malla de fondo.
 - `title()`: permite añadir un título o subtítulo.

Funciones de localización e identificación de puntos

Gráficos en lenguaje R

Funciones de Localización

- Una vez que en la ventana de gráficos tenemos algo representado, existen dos funciones que permiten trabajar interactivamente con el gráfico:
- Función `locator()`
- Función `identity()`

Función locator()

- La función locator(): al situar el cursor sobre la ventana de gráficos, cada vez que pulsemos el botón izquierdo del ratón, se almacenan en memoria las coordenadas del punto que marquemos. Al pulsar la tecla < ESC >, R nos muestra dichas coordenadas en la consola (si hemos ejecutado simplemente locator()) o las guarda en el objeto al que hayamos asignado la salida dicha función. Por ejemplo, si hemos ejecutado posiciones=locator(), al pulsar < ESC > las coordenadas de los puntos que hayamos marcado en el gráfico se guardan en la variable posiciones.

Función `identity()`

- La función `identify()` permite identificar con el ratón a qué posiciones dentro del conjunto de datos corresponden los puntos que señalemos en un gráfico. Si, por ejemplo, graficamos con `plot()` y picamos en algunos puntos del gráfico con el ratón, al pulsar `< ESC >`, R nos devuelve en la consola (y representa en el gráfico) los números de orden dentro del dataframe a los que corresponden los puntos que hemos marcado.

Laboratorio

- Creando Gráficos en el entorno R
- Personalizando Gráficos en R

Funciones en Lenguaje R

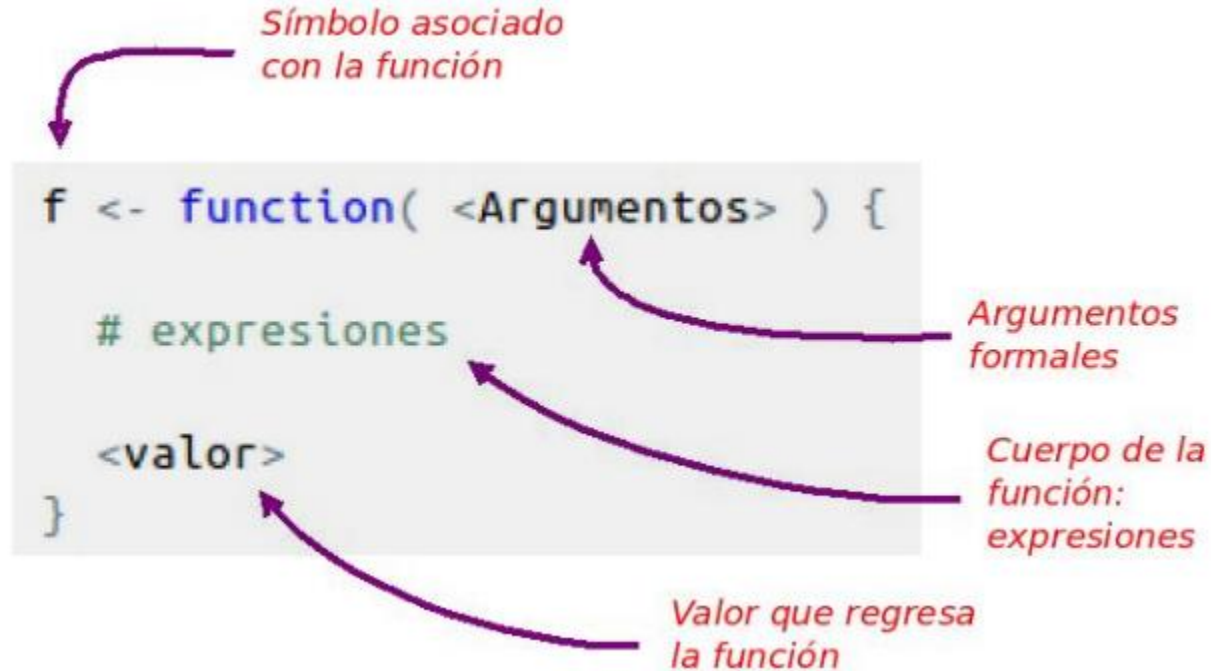
- Creacion de Funciones
- Recursividad
- Estructuras de Control

Creacion de funciones

Funciones en lenguaje R

Sintaxis de la Creacion de Funciones

- La sintaxis general para crear una funcion en R es:



Creacion de Funciones

- Ejemplo:

```
> MisFunciones.area <- function(radio,pi=3.1416){  
  result <- pi*radio^2  
  return (result)  
}  
> MisFunciones.area(0.4567)  
[1] 0.6552589  
>
```

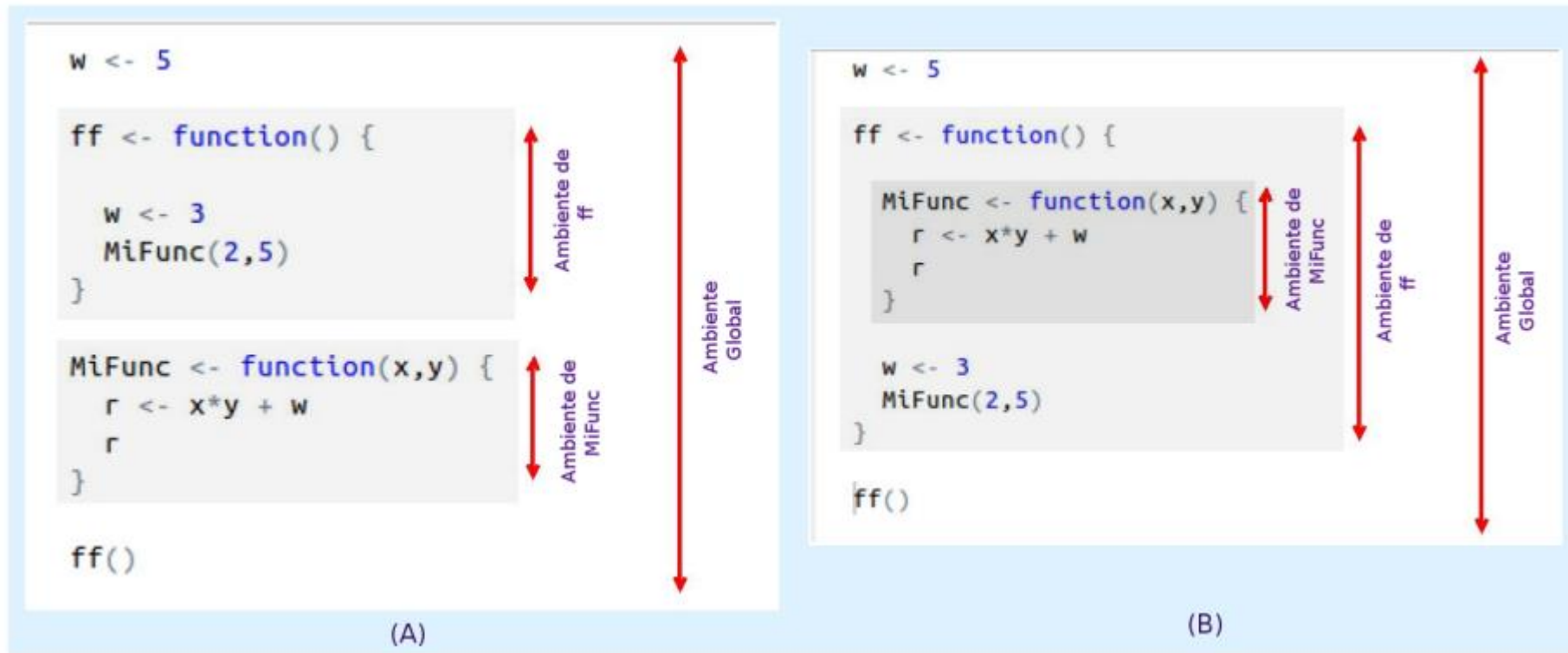
Argumento especial ...

Sirve para transferir un número variable de argumentos a otra función.

```
> s3 <- function(x, y, z=0){  
+   r <- x +y +z  
+   r  
+ }  
> s2 <- function(x,y,...){  
+   r <- s3(x,y,...)  
+   r  
+ }  
> s2(3,4)  
[1] 7  
> s2(3,4,9)  
[1] 16
```

Alcance de las Variables

- Variables libres



Recursividad

Funciones en lenguaje R

Recursividad

R soporta la recursividad, es decir la posibilidad de una función de llamarse o invocarse a sí misma

```
MiFact <- function(n) {  
  if (n==0) return (1) # salida inmediata  
  if (n > 0) return (n*MiFact(n-1))  
  return (NULL) # caso fallido  
}  
  
# Ahora se usa la función con 5 y 8  
MiFact(5)
```

Estructuras de Control

Funciones en lenguaje R

Estructuras de Control en R

Se entiende por estructuras de control aquellas construcciones sintácticas del lenguaje que dirigen el flujo de la ejecución de un programa en una dirección o en otra dentro de su código

IF-ELSE

La construcción IF admite una sola expresión, pero ésta puede ser la expresión compuesta, que se construye mediante los paréntesis de llave { }

```
if (10 > aa) { # 1er. bloque
    print("RANGO MENOR")
} else if ( 10 <= aa && aa <= 20) { # 2o. bloque
    print("primer renglon"); print("RANGO MEDIO")
} else { # 3er. bloque
    print("RANGO MAYOR")
}

## [1] "primer renglon"
## [1] "RANGO MEDIO"
```

Bucles o repeticiones

El lenguaje cuenta con varios tipos de ciclos o repeticiones, a saber: repeticiones por un número determinado de veces, repeticiones mientras se cumple una condición y repeticiones infinitas.

Bucle FOR

- Un numero determinado de veces

```
letras <- c("c", "l", "i", "M", "T", "A")  
for (i in 1:6) {  
  print(letras[i])  
}
```

```
## [1] "c"  
## [1] "l"  
## [1] "i"  
## [1] "M"  
## [1] "T"  
## [1] "A"
```

... continua

- Otra forma

```
for (i in seq_along(letras)) {  
    print(letras[i])  
}  
  
for (letra in letras) {  
    print(letra)  
}
```

Bucle WHILE

- Repite mientras se cumple una condicion

```
# Para la misma tarea de los ejemplos anteriores  
i <- 1  
while (i <= 6) {  
  print(letras[i])  
  i <- i + 1  
}
```


Bucle REPEAT

- Repeticiones indeterminadas

```
i <- 1
repeat {
  print(letras[i])
  i <- i + 1
  if (i > 6)
    break
}
```

Instruccion BREAK

el lenguaje provee facilidades para que desde el interior del bloque de expresiones que se repiten, se obligue la interrupción del ciclo, mediante la instrucción break,

```
i <- 1
repeat {
  print(letras[i])
  i <- i + 1
  if (i > 6)
    break
}
```

Instruccion NEXT

La instrucción next interrumpe el flujo normal de ejecución de un programa de una manera diferente: en vez de salir de un ciclo, solamente impide la ejecución de las instrucciones siguientes y regresa al principio del ciclo para ejecutar la siguiente iteración.

```
for (i in 1:7) {  
    if (3 <= i && i <= 5)  
        next  
    print(i)  
}
```

```
## [1] 1  
## [1] 2  
## [1] 6  
## [1] 7
```

Laboratorio

- Creando Funciones en el entorno R
- Usando Funciones Personalizadas en R