[Programación] R

- R es un lenguaje de programación y un entorno de software libre y de código abierto diseñado específicamente para el análisis estadístico y la visualización de datos.
- Es ampliamente utilizado en diversas disciplinas, incluyendo la ciencia de datos, la bioinformática, la economía, la epidemiología y muchas otras áreas donde se requiere análisis cuantitativo.

1. Introducción y primeros pasos con el lenguaje R

Vectorized Functions

TO USE WITH MUTATE ()

mutate() and **transmute()** apply vectorized functions to columns to create new columns. Vectorized functions take vectors as input and return vectors of the same length as output.

OFFSET
- dplyr::**lag()** - offset elements by 1
- dplyr::**lead()** - offset elements by -1

CUMULATIVE AGGREGATE
- dplyr::**cumall()** - cumulative all()
- dplyr::**cumany()** - cumulative any()
- **cummax()** - cumulative max()
- dplyr::**cummean()** - cumulative mean()
- **cummin()** - cumulative min()
- **cumprod()** - cumulative prod()
- **cumsum()** - cumulative sum()

RANKING
- dplyr::**cume_dist()** - proportion of all values <=
- dplyr::**dense_rank()** - rank w ties = min, no gaps
- dplyr::**min_rank()** - rank with ties = min
- dplyr::**ntile()** - bins into n bins
- dplyr::**percent_rank()** - min_rank scaled to [0,1]
- dplyr::**row_number()** - rank with ties = "first"

MATH
- +, -, *, /, ^, %/%, %% - arithmetic ops
- log(), log2(), log10() - logs
- <, <=, >, >=, !=, == - logical comparisons
- dplyr::**between()** - x >= left & x <= right
- dplyr::**near()** - safe == for floating point numbers

MISCELLANEOUS
- dplyr::**case_when()** - multi-case if_else()
```
starwars %>%
  mutate(type = case_when(
    height > 200 | mass > 200 ~ "large",
    species == "Droid"        ~ "robot",
    TRUE                      ~ "other"))
```
- dplyr::**coalesce()** - first non-NA values by element across a set of vectors
- dplyr::**if_else()** - element-wise if() + else()
- dplyr::**na_if()** - replace specific values with NA
- **pmax()** - element-wise max()
- **pmin()** - element-wise min()

Summary Functions

TO USE WITH SUMMARISE ()

summarise() applies summary functions to columns to create a new table. Summary functions take vectors as input and return single values as output.

COUNT
- dplyr::**n()** - number of values/rows
- dplyr::**n_distinct()** - # of uniques
- sum(!is.na()) - # of non-NA's

POSITION
- **mean()** - mean, also mean(!is.na())
- **median()** - median

LOGICAL
- **mean()** - proportion of TRUE's
- **sum()** - # of TRUE's

ORDER
- dplyr::**first()** - first value
- dplyr::**last()** - last value
- dplyr::**nth()** - value in nth location of vector

RANK
- **quantile()** - nth quantile
- **min()** - minimum value
- **max()** - maximum value

SPREAD
- **IQR()** - Inter-Quartile Range
- **mad()** - median absolute deviation
- **sd()** - standard deviation
- **var()** - variance

Row Names

Tidy data does not use rownames, which store a variable outside of the columns. To work with the rownames, first move them into a column.

tibble::**rownames_to_column()**
Move row names into col.
a <- rownames_to_column(mtcars, var = "C")

tibble::**column_to_rownames()**
Move col into row names.
column_to_rownames(a, var = "C")

Also tibble::**has_rownames()** and tibble::**remove_rownames()**.

Combine Tables

COMBINE VARIABLES

bind_cols(…, .name_repair) Returns tables placed side by side as a single table. Column lengths must be equal. Columns will NOT be matched by id (to do that look at Relational Data below), so be sure to check that both tables are ordered the way you want before binding.

RELATIONAL DATA

Use a **"Mutating Join"** to join one table to columns from another, matching values with the rows that they correspond to. Each join retains a different combination of values from the tables.

left_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), …, keep = FALSE, na_matches = "na") Join matching values from y to x.

right_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), …, keep = FALSE, na_matches = "na") Join matching values from x to y.

inner_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), …, keep = FALSE, na_matches = "na") Join data. Retain only rows with matches.

full_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), …, keep = FALSE, na_matches = "na") Join data. Retain all values, all rows.

COLUMN MATCHING FOR JOINS

Use **by = c("col1", "col2", …)** to specify one or more common columns to match on.
left_join(x, y, by = "A")

Use a named vector, **by = c("col1" = "col2")**, to match on columns that have different names in each table.
left_join(x, y, by = c("C" = "D"))

Use **suffix** to specify the suffix to give to unmatched columns that have the same name in both tables.
left_join(x, y, by = c("C" = "D"), suffix = c("1", "2"))

COMBINE CASES

bind_rows(…, .id = NULL) Returns tables one on top of the other as a single table. Set .id to a column name to add a column of the original table names (as pictured).

Use a **"Filtering Join"** to filter one table against the rows of another.

semi_join(x, y, by = NULL, copy = FALSE, …, na_matches = "na") Return rows of x that have a match in y. Use to see what will be included in a join.

anti_join(x, y, by = NULL, copy = FALSE, …, na_matches = "na") Return rows of x that do not have a match in y. Use to see what will not be included in a join.

Use a **"Nest Join"** to inner join one table into another into a nested data frame.

nest_join(x, y, by = NULL, copy = FALSE, keep = FALSE, name = NULL, …) Join data, nesting matches from y in a single new data frame column.

SET OPERATIONS

intersect(x, y, …) Rows that appear in both x and y.

setdiff(x, y, …) Rows that appear in x but not y.

union(x, y, …) Rows that appear in x or y. (Duplicates removed). **union_all()** retains duplicates.

Use **setequal()** to test whether two data sets contain the exact same rows (in any order).

posit

Data visualization with ggplot2 :: CHEAT SHEET

Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data** set, a **coordinate system**, and **geoms**—visual marks that represent data points.

data + **geom** $x = F$ $y = A$ + **coordinate system** = **plot**

To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.

data + **geom** $x = F$ $y = A$ color $= F$ size $= A$ + **coordinate system** = **plot**

Complete the template below to build a graph.

```
ggplot (data = <DATA>) +          required
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),
    stat = <STAT>, position = <POSITION>)) +    Not
  <COORDINATE_FUNCTION> +                       required,
  <FACET_FUNCTION> +                            sensible
  <SCALE_FUNCTION> +                            defaults
  <THEME_FUNCTION>                              supplied
```

ggplot(data = mpg, **aes**(x = cty, y = hwy)) Begins a plot that you finish by adding layers to. Add one geom function per layer.

last_plot() Returns the last plot.

ggsave("plot.png", width = 5, height = 5) Saves last plot as 5' x 5' file named "plot.png" in working directory. Matches file type to file extension.

Aes   Common aesthetic values.

color and **fill** - string ("red", "#RRGGBB")

linetype - integer or string (0 = "blank", 1 = "solid", 2 = "dashed", 3 = "dotted", 4 = "dotdash", 5 = "longdash", 6 = "twodash")

lineend - string ("round", "butt", or "square")

linejoin - string ("round", "mitre", or "bevel")

size - integer (line width in mm)

shape - integer/shape name or a single character ("a")

Geoms
Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

GRAPHICAL PRIMITIVES
a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))

a + geom_blank() and **a + expand_limits()** Ensure limits include values across all plots.

b + geom_curve(aes(yend = lat + 1, xend = long + 1), curvature = 1) - x, xend, y, yend, alpha, angle, color, curvature, linetype, size

a + geom_path(lineend = "butt", linejoin = "round", linemitre = 1) x, y, alpha, color, group, linetype, size

a + geom_polygon(aes(alpha = 50)) - x, y, alpha, color, fill, group, subgroup, linetype, size

b + geom_rect(aes(xmin = long, ymin = lat, xmax = long + 1, ymax = lat + 1)) - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

a + geom_ribbon(aes(ymin = unemploy - 900, ymax = unemploy + 900)) - x, ymax, ymin, alpha, color, fill, group, linetype, size

LINE SEGMENTS
common aesthetics: x, y, alpha, color, linetype, size

b + geom_abline(aes(intercept = 0, slope = 1))
b + geom_hline(aes(yintercept = lat))
b + geom_vline(aes(xintercept = long))
b + geom_segment(aes(yend = lat + 1, xend = long + 1))
b + geom_spoke(aes(angle = 1:1155, radius = 1))

ONE VARIABLE   continuous
c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)

c + geom_area(stat = "bin") x, y, alpha, color, fill, linetype, size

c + geom_density(kernel = "gaussian") x, y, alpha, color, fill, group, linetype, size, weight

c + geom_dotplot() x, y, alpha, color, fill

c + geom_freqpoly() x, y, alpha, color, group, linetype, size

c + geom_histogram(binwidth = 5) x, y, alpha, color, fill, linetype, size, weight

c2 + geom_qq(aes(sample = hwy)) x, y, alpha, color, fill, linetype, size, weight

discrete
d <- ggplot(mpg, aes(fl))

d + geom_bar() x, alpha, color, fill, linetype, size, weight

TWO VARIABLES
both continuous
e <- ggplot(mpg, aes(cty, hwy))

e + geom_label(aes(label = cty), nudge_x = 1, nudge_y = 1) - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

e + geom_point() x, y, alpha, color, fill, shape, size, stroke

e + geom_quantile() x, y, alpha, color, group, linetype, size, weight

e + geom_rug(sides = "bl") x, y, alpha, color, linetype, size

e + geom_smooth(method = lm) x, y, alpha, color, fill, group, linetype, size, weight

e + geom_text(aes(label = cty), nudge_x = 1, nudge_y = 1) - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

one discrete, one continuous
f <- ggplot(mpg, aes(class, hwy))

f + geom_col() x, y, alpha, color, fill, group, linetype, size

f + geom_boxplot() x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight

f + geom_dotplot(binaxis = "y", stackdir = "center") x, y, alpha, color, fill, group

f + geom_violin(scale = "area") x, y, alpha, color, fill, group, linetype, size, weight

both discrete
g <- ggplot(diamonds, aes(cut, color))

g + geom_count() x, y, alpha, color, fill, shape, size, stroke

e + geom_jitter(height = 2, width = 2) x, y, alpha, color, fill, shape, size

THREE VARIABLES
seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2)); l <- ggplot(seals, aes(long, lat))

l + geom_contour(aes(z = z)) x, y, z, alpha, color, group, linetype, size, weight

l + geom_contour_filled(aes(fill = z)) x, y, alpha, color, fill, group, linetype, size, subgroup

continuous bivariate distribution
h <- ggplot(diamonds, aes(carat, price))

h + geom_bin2d(binwidth = c(0.25, 500)) x, y, alpha, color, fill, linetype, size, weight

h + geom_density_2d() x, y, alpha, color, group, linetype, size

h + geom_hex() x, y, alpha, color, fill, size

continuous function
i <- ggplot(economics, aes(date, unemploy))

i + geom_area() x, y, alpha, color, fill, linetype, size

i + geom_line() x, y, alpha, color, group, linetype, size

i + geom_step(direction = "hv") x, y, alpha, color, group, linetype, size

visualizing error
df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
j <- ggplot(df, aes(grp, fit, ymin = fit - se, ymax = fit + se))

j + geom_crossbar(fatten = 2) - x, y, ymax, ymin, alpha, color, fill, group, linetype, size

j + geom_errorbar() - x, ymax, ymin, alpha, color, group, linetype, size, width Also **geom_errorbarh()**.

j + geom_linerange() x, ymin, ymax, alpha, color, group, linetype, size

j + geom_pointrange() x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

maps
data <- data.frame(murder = USArrests$Murder, state = tolower(rownames(USArrests)))
map <- map_data("state")
k <- ggplot(data, aes(fill = murder))

k + geom_map(aes(map_id = state), map = map) + **expand_limits**(x = map$long, y = map$lat) map_id, alpha, color, fill, linetype, size

l + geom_raster(aes(fill = z), hjust = 0.5, vjust = 0.5, interpolate = FALSE) x, y, alpha, fill

l + geom_tile(aes(fill = z)) x, y, alpha, color, fill, linetype, size, width

Base R
Cheat Sheet

Getting Help

Accessing the help files

?mean
Get help of a particular function.
help.search('weighted mean')
Search the help files for a word or phrase.
help(package = 'dplyr')
Find help for a package.

More about an object

str(iris)
Get a summary of an object's structure.
class(iris)
Find the class an object belongs to.

Using Packages

install.packages('dplyr')
Download and install a package from CRAN.

library(dplyr)
Load the package into the session, making all its functions available to use.

dplyr::select
Use a particular function from a package.

data(iris)
Load a built-in dataset into the environment.

Working Directory

getwd()
Find the current working directory (where inputs are found and outputs are sent).

setwd('C://file/path')
Change the current working directory.

Use projects in RStudio to set the working directory to the folder you are working in.

Vectors

Creating Vectors

c(2, 4, 6)	2 4 6	Join elements into a vector
2:6	2 3 4 5 6	An integer sequence
seq(2, 3, by=0.5)	2.0 2.5 3.0	A complex sequence
rep(1:2, times=3)	1 2 1 2 1 2	Repeat a vector
rep(1:2, each=3)	1 1 1 2 2 2	Repeat elements of a vector

Vector Functions

sort(x)	**rev(x)**
Return x sorted.	Return x reversed.
table(x)	**unique(x)**
See counts of values.	See unique values.

Selecting Vector Elements

By Position

x[4]	The fourth element.
x[-4]	All but the fourth.
x[2:4]	Elements two to four.
x[-(2:4)]	All elements except two to four.
x[c(1, 5)]	Elements one and five.

By Value

x[x == 10]	Elements which are equal to 10.
x[x < 0]	All elements less than zero.
x[x %in% c(1, 2, 5)]	Elements in the set 1, 2, 5.

Named Vectors

x['apple']	Element with name 'apple'.

Programming

For Loop

```
for (variable in sequence){
    Do something
}
```

Example

```
for (i in 1:4){
    j <- i + 10
    print(j)
}
```

While Loop

```
while (condition){
    Do something
}
```

Example

```
while (i < 5){
    print(i)
    i <- i + 1
}
```

If Statements

```
if (condition){
    Do something
} else {
    Do something different
}
```

Example

```
if (i > 3){
    print('Yes')
} else {
    print('No')
}
```

Functions

```
function_name <- function(var){
    Do something
    return(new_variable)
}
```

Example

```
square <- function(x){
    squared <- x*x
    return(squared)
}
```

Reading and Writing Data

Also see the **readr** package.

Input	Ouput	Description
df <- read.table('file.txt')	write.table(df, 'file.txt')	Read and write a delimited text file.
df <- read.csv('file.csv')	write.csv(df, 'file.csv')	Read and write a comma separated value file. This is a special case of read.table/write.table.
load('file.RData')	save(df, file = 'file.Rdata')	Read and write an R data file, a file type special for R.

Conditions								
	a == b	Are equal	a > b	Greater than	a >= b	Greater than or equal to	is.na(a)	Is missing
	a != b	Not equal	a < b	Less than	a <= b	Less than or equal to	is.null(a)	Is null

Learn more at **web page or vignette** • package version • Updated: 3/15

Types

Converting between common data types in R. Can always go from a higher value in the table to a lower value.

as.logical	TRUE, FALSE, TRUE	Boolean values (TRUE or FALSE).
as.numeric	1, 0, 1	Integers or floating point numbers.
as.character	'1', '0', '1'	Character strings. Generally preferred to factors.
as.factor	'1', '0', '1', levels: '1', '0'	Character strings with preset levels. Needed for some statistical models.

Maths Functions

log(x)	Natural log.	sum(x)		Sum.
exp(x)	Exponential.	mean(x)		Mean.
max(x)	Largest element.	median(x)		Median.
min(x)	Smallest element.	quantile(x)		Percentage quantiles.
round(x, n)	Round to n decimal places.	rank(x)		Rank of elements.
signif(x, n)	Round to n significant figures.	var(x)		The variance.
cor(x, y)	Correlation.	sd(x)		The standard deviation.

Variable Assignment

```
> a <- 'apple'
> a
[1] 'apple'
```

The Environment

ls()	List all variables in the environment.
rm(x)	Remove x from the environment.
rm(list = ls())	Remove all variables from the environment.

You can use the environment panel in RStudio to browse variables in your environment.

Matrices

```
m <- matrix(x, nrow = 3, ncol = 3)
```
Create a matrix from x.

m[2,] - Select a row

m[, 1] - Select a column

m[2, 3] - Select an element

t(m)
Transpose

m %*% n
Matrix Multiplication

solve(m, n)
Find x in: m * x = n

Lists

```
l <- list(x = 1:5, y = c('a', 'b'))
```
A list is a collection of elements which can be of different types.

l[[2]]	l[1]	l$x	l['y']
Second element of l.	New list with only the first element.	Element named x.	New list with only element named y.

Also see the **dplyr** *package.*

Data Frames

```
df <- data.frame(x = 1:3, y = c('a', 'b', 'c'))
```
A special case of a list where all elements are the same length.

x	y
1	a
2	b
3	c

List subsetting

df$x   df[[2]]

Understanding a data frame

View(df)	See the full data frame.
head(df)	See the first 6 rows.

Matrix subsetting

df[, 2]

df[2,]

df[2, 2]

nrow(df)	Number of rows.
ncol(df)	Number of columns.
dim(df)	Number of columns and rows.

cbind - Bind columns.

rbind - Bind rows.

Strings

Also see the **stringr** *package.*

paste(x, y, sep = ' ')	Join multiple vectors together.
paste(x, collapse = ' ')	Join elements of a vector together.
grep(pattern, x)	Find regular expression matches in x.
gsub(pattern, replace, x)	Replace matches in x with a string.
toupper(x)	Convert to uppercase.
tolower(x)	Convert to lowercase.
nchar(x)	Number of characters in a string.

Factors

factor(x)	cut(x, breaks = 4)
Turn a vector into a factor. Can set the levels of the factor and the order.	Turn a numeric vector into a factor by 'cutting' into sections.

Statistics

lm(y ~ x, data=df) Linear model.	t.test(x, y) Perform a t-test for difference between means.	prop.test Test for a difference between proportions.
glm(y ~ x, data=df) Generalised linear model.		
summary Get more detailed information out a model.	pairwise.t.test Perform a t-test for paired data.	aov Analysis of variance.

Distributions

	Random Variates	Density Function	Cumulative Distribution	Quantile
Normal	rnorm	dnorm	pnorm	qnorm
Poisson	rpois	dpois	ppois	qpois
Binomial	rbinom	dbinom	pbinom	qbinom
Uniform	runif	dunif	punif	qunif

Plotting

Also see the **ggplot2** *package.*

plot(x)	plot(x, y)	hist(x)
Values of x in order.	Values of x against y.	Histogram of x.

Dates

See the **lubridate** *package.*

Learn more at **web page or vignette** • package version • Updated: 3/15

RStudio IDE : : GUÍA RÁPIDA

Documentos y apps

RStudio desarrolla herramientas gratuitas y abiertas para R. Su entorno de desarrollo integrado (IDE) facilita el análisis de datos con R.

También ofrece: muchos paquetes R (e.g. tidyverse, sparklyr, ggplot2, dplyr), incluidos Shiny (crea aplicaciones web sencillas en R) y R Markdown (te permite convertir tus análisis en documentos, informes, presentaciones y paneles de alta calidad; compartirlos y reproducirlos).

Escribe tu código

- Abrir en una nueva ventana
- Pestaña de navegación
- Guardar
- Encontrar y reemplazar
- Compilar como cuaderno
- Ejecutar el código seleccionado
- Importar datos con asistente

Cursor del usuario
Ejecutar el código previo
Ejecutar el código completo
Esquema del archivo

Selección: Alt + arrastre del mouse.

Código de diagnóstico que aparece en el margen. Pase el mouse sobre los símbolos de diagnóstico para obtener más detalles.

Resaltado de sintaxis.

Autocompletar mediante tabulación: nombres de funciones, rutas de archivos, argumentos y más.

Salta a la función en el archivo

Cambia el tipo de archivo

Directorio de trabajo

Maximiza/minimiza los paneles

Presiona ↑ para ver el historial de comandos

Arrastra los límites

Soporte R

- Historial de comandos
- Mostrar presentaciones de diapositivas .RPres Archivo> Nuevo archivo> Presentación R

Carga el área de trabajo
Guarda el área de trabajo
Borra todos los objetos
Buscador del área de trabajo

Elegir el entorno para mostrar

Mostrar objetos como lista o cuadrícula

Data
@iris — 150 obs. of 5 variables
Values
a — 1
Functions
foo — function (x)

Muestra los objetos guardados por tipo, con una breve descripción
Ver en el visor de datos
Ver el código fuente de la función

Crear una carpeta
Subir un archivo
Eliminar un archivo
Renombrar un archivo
Cambiar el directorio

Copy…
Move…
Export…
Set As Working Directory
Go To Working Directory

Buscador de archivos. Haga clic en el nombre del archivo o directorio para abrirlo.

Entorno de desarrollo RStudio IDE

GRÁFICOS

Navegar en los últimos gráficos
Abrir en una ventana
Exportar el gráfico
Eliminar el gráfico
Eliminar todos los gráficos

PAQUETES

Instalar paquetes
Actualizar paquetes
Crear un paquete reproducible desde tu proyecto.

scales	Scale Functions for Visualization	0.3.0
shiny	Web Application Framework for R	0.12.2
shinydashboard	Create Dashboards with 'Shiny'	0.5.1

Márcalo para activar el paquete (i.e. library()), o desmárcalo para desactivarlo (i.e. detach()).
Versión del paquete instalada
Eliminar el paquete.

AYUDA

Página de ayuda con links útiles
Busca dentro de la página de ayuda
Busca una página de ayuda

PANEL DE VISIÓN

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
	All	All	All	All	All
1	5.1	3.5	1.4	0.2	setosa
2					
3					
4					

Filtra las filas por un valor o rango de valores
Ordena los valores
Busca un valor

```
1   # Comentario...
2
3
4
5
6   "P0030001"
7   "P0030002"
8   "P0030003"
9   "P0030004"
10
11
12  get_digit <-function() {
13    ("num" %% (10 ^ n))
14    %/% (10 ^ (n - 1))
15  }}
16
17  fo
18    for
19    foo      {.GlobalEnv}
20    force    {base}
21
22
```

```
> foo(1)
[1] 2
> foo <- function(x) x + 1
> foo(2)
foo(2)
> foo(1)
```

Maxima FORMACION

Traducido por Rosana Ferrero • https://www.maximaformacion.es Para acceder a más guías visite https://www.rstudio.com/resources/cheatsheets/
RStudio® es una marca registrada de RStudio, Inc. • CC BY RStudio • info@rstudio.com • 844-448-1212 • rstudio.com • Actualizado: 2/18

Estadística descriptiva : : GUÍA RÁPIDA

Resumen numérico

Conceptos básicos

VISUALIZA LOS DATOS　　F M A

```
head(mtcars)
View(mtcars)
str(mtcars)
```

Tipos de variables:

- Continua (números reales)
- Discretas (números enteros)
- Ordinales (categorías con orden)
- Nominales (categorías sin orden)

funciones de resumen

Una variable categórica:

```
table(mtcars$cyl)
prop.table(table(mtcars$cyl))
```

Más de una variable categórica:

```
table(mtcars$cyl, mtcars$vs)

gmodels::CrossTable(mtcars$cyl,
mtcars$vs)

ftable(mtcars$cyl, mtcars$vs,
mtcars$am)
```

Una variable numérica:

```
mean(mtcars$mpg)
```

Una numérica y una categórica

```
by(mtcars$mpg, mtcars$cyl, mean)

plyr::ddply(mtcars,"cyl",summarise,
            N=length(mpg),
            mean=mean(mpg),
            sd=sd(mpg))
```

Variables numéricas y categóricas:

```
summary(mtcars)
```

ESTADÍSTICOS DESCRIPTIVOS

Posición central　　*opcional*

```
mean(mtcars$mpg, tr=.2)
median(mtcars$mpg)
DescTools::Mode(mtcars$mpg)
```

Posición no central　　*opcional*

```
quantile(mtcars$mpg,c(.05,.95))
```

Dispersión　　*opcional*

```
var(mtcars$mpg)
sd(mtcars$mpg)
IQR(mtcars$mpg)
WRS2::trimse(mtcars$mpg, tr=.2)
msmedse(mtcars$mpg, sewarn=T)
```
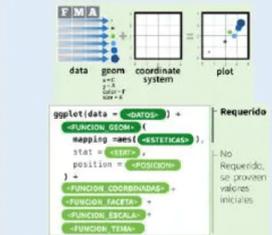
Forma

```
fBasics::skewness(mtcars$mpg)
fBasics::kurtosis(mtcars$mpg)
```

Gráficos

Conceptos básicos

PAQUETE ggplot2

F M A

```
data  geom  coordinate  plot
             system
```

```
ggplot(data = <DATOS> ) +
  <FUNCION GEOM>(
  mapping=aes(<ESTETICAS>),
  stat = <STAT> ,
  position = <POSICION>
) +
  <FUNCION COORDENADAS> +
  <FUNCION FACETA> +
  <FUNCION ESCALA> +
  <FUNCION TEMA>
```
Requerido

No Requerido, se proveen valores iniciales

Una variable numérica:

```
c<-ggplot(mtcars,aes(mpg))
```

```
c+geom_histogram(binwidth=5)
c+geom_density()
```

```
c<-ggplot(mtcars,aes(x="",y=mpg))
```

```
c+geom_boxplot()
```

Una variable categórica:　　*si no es un factor*

```
c<-ggplot(mtcars,aes(as.factor(am)))
```

```
c+geom_bar()
```

Una variable numérica y una categórica:

```
c<-ggplot(mtcars,aes(x=mpg,
fill=as.factor(cyl)))
```

```
c+geom_histogram(binwidth=5)
c+geom_density()
```

Una variable categórica:　　*si no son factores*

```
c<-ggplot(mtcars,
aes(x=as.factor(am)),
fill=as.factor(cyl))
```

```
c+geom_bar()
c+geom_bar(position="dodge")
```

FACETAS

Dividen el gráfico en subgráficos según una o más variables.

```
c<-ggplot(mtcars,aes(x=mpg,
fill=as.factor(cyl)))
```

```
c+geom_histogram(binwidth=5)+face
t_grid(.~as.factor(am))
```

```
c+geom_histogram(binwidth=5)+face
t_grid(as.factor(am)~.)
```

```
c+geom_histogram(binwidth=5)+face
t_grid(as.factor(am)~
as.factor(vs))
```

```
c+geom_histogram(binwidth=5)+face
t_wrap(~as.factor(am))
```

Introducción al lenguaje R : : GUÍA RÁPIDA

Pedir ayuda

PÁGINAS DE AYUDA

```
?mean
help.search('median')
help(package='dplyr')
```

para una función
para una palabra o frase
para un paquete

SOBRE UN OBJETO

Resumen de la estructura del objeto

```
str(iris)
```

Utilizar paquetes

Extiende las funciones de R mediante paquetes

```
install.packages('dplyr')
library(dplyr)
```

activa un paquete
instala un paquete

Accede a una base de datos de R

```
data(iris)
```

Utiliza una función de un paquete

```
dplyr::select
```

Directorio de trabajo

```
getwd()
setwd(dplyr)
```

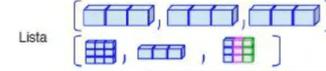muestra tu directorio
cambia tu directorio

Asignación

```
a<-'group' o a<-1
```

Tipos de datos

vector y factor

matriz
filas
columnas
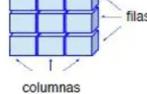
array
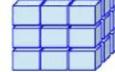
data frame o tabla

Lista

Vectores y factores

CREAR

```
x<-c(2,4,5)
```

opción	descripción
c(2,4,5)	une los elementos en un vector
2:6	crea una secuencia de enteros
seq(2,3,by=.5)	crea una secuencia más
rep(1:2, times, each)	repite elementos
factor(x)	convierte un vector en factor

SELECCIONAR

Por posición

opción	descripción
x[4]	el cuarto elemento
x[-4]	todos menos el cuarto
x[2:4]	elementos desde el 2º al 4º
x[c(1,3)]	1º y 3º elementos

Por valor o nombre

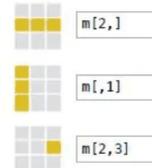| opción	descripción
x[x==10]	elementos con valor 10
x[x<0]	elementos menores a 0
x[x %in% c(1,3)]	elementos en el conjunto 1,3
x['group']	elementos con nombre 'group'

CARACTERÍSTICAS

```
unique(x)
sort(x)
length(x)
```

Matrices

CREAR

```
m<-matrix(x=1:9,nrow=3,ncol=3)
```

SELECCIONAR

```
m[2,]
```
```
m[,1]
```
```
m[2,3]
```

CARACTERÍSTICAS

```
t(x)
m %*% n
dim(x)
```

transpone
multiplica
dimensión

Listas

CREAR
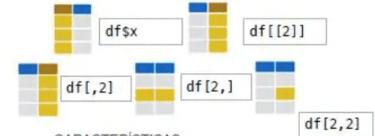
```
l<-list(x=1:5, y=c('a','b'))
```

SELECCIONAR

opción	descripción
l[[2]]	segundo elemento de l
l[1]	primer elemento de l
l$x	elemento llamado 'x'
l['y']	elemento llamado 'y'

Data frames

CREAR

```
df<-data.frame(x=1:3,y=c('a','b','cs'))
```

SELECCIONAR

```
df$x
```
```
df[[2]]
```
```
df[,2]
```
```
df[2,]
```
```
df[2,2]
```

CARACTERÍSTICAS

```
View(df)
head(df)
nrow(df) o ncol(df)
```

UNIR

```
cbind(df1,df2)
```
```
rbind(df1,df2)
```
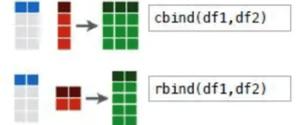
Creado por Rosana Ferrero para Máxima Formación • https://www.maximaformacion.es • consultoria@maximainformacion.com • 958 32 70 46 (España)
• Actualizado: 6/18 • CC BY Rosana Ferrero, con material de RStudio, Inc. Para acceder a más guías visite https://www.rstudio.com/resources/cheatsheets/

Máxima FORMACIÓN