

# [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.

**vectorized function**

**OFFSET**

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

**CUMULATIVE AGGREGATE**

dplyr: **cumall()** - cumulative all()  
dplyr: **cumany()** - cumulative any()  
dplyr: **cummax()** - cumulative max()  
dplyr: **cummean()** - cumulative mean()  
dplyr: **cummin()** - cumulative min()  
dplyr: **cumprod()** - cumulative prod()  
dplyr: **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  
dplyr: **pmax()** - element-wise max()  
dplyr: **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.

**summary function**

**COUNT**

dplyr: **n()** - number of values/rows  
dplyr: **n\_distinct()** - # of uniques  
dplyr: **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 to 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).





# 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.



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



Complete the template below to build a graph.

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),
  stat = <STAT>, position = <POSITION>) +
  <COORDINATE_FUNCTION> +
  <FACET_FUNCTION> +
  <SCALE_FUNCTION> +
  <THEME_FUNCTION>
```

required  
not required, sensible defaults supplied

ggplot(data = mpg, aes(x = cty, y = hwy)) Begins a plot that you finish by adding layers. 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, unemployment))
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, yend, y, yend,
alpha, angle, color, curvature, linetype, size)

a + geom_path(linetype = "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 = unemployment - 900,
ymax = unemployment + 900) - x, ymax, ymin,
alpha, color, fill, group, linetype, size)
```

### LINE SEGMENTS

```
common aesthetics: x, y, alpha, color, linetype, size
b + geom_sline(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 = 111.55, 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(fli))
d + geom_bar()
x, alpha, color, fill, linetype, size, weight
```

### TWO VARIABLES

```
both continuous
f <- 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, unemployment))

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
```

# 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

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

### Vector Functions

<b>sort(x)</b> Return x sorted.	<b>rev(x)</b> Return x reversed.
<b>table(x)</b> See counts of values.	<b>unique(x)</b> See unique values.

### Selecting Vector Elements

#### By Position

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

#### By Value

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

#### Named Vectors

<code>x['apple']</code>	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	Output	Description
<code>df &lt;- read.table('file.txt')</code>	<code>write.table(df, 'file.txt')</code>	Read and write a delimited text file.
<code>df &lt;- read.csv('file.csv')</code>	<code>write.csv(df, 'file.csv')</code>	Read and write a comma separated value file. This is a special case of read.table/write.table.
<code>load('file.Rdata')</code>	<code>save(df, file = 'file.Rdata')</code>	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

### Types

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

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

### Maths Functions

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

`l[1]` New list with only the first element.

`l$x` Element named x.

`l['y']` New list with only element named y.

### Data Frames

Also see the **dplyr** package.

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

### 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)` Turn a vector into a factor. Can set the levels of the factor and the order.

`cut(x, breaks = 4)` Turn a numeric vector into a factor by 'cutting' into sections.

### Statistics

`lm(y ~ x, data=df)` Linear model.

`glm(y ~ x, data=df)` Generalised linear model.

`summary` Get more detailed information out a model.

`t.test(x, y)` Perform a t-test for difference between means.

`prop.test` Test for a difference between proportions.

`pairwise.t.test` Perform a t-test for paired data.

`aov` Analysis of variance.

### Distributions

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

### Plotting

Also see the **ggplot2** package.

`plot(x)` Values of x in order.

`plot(x, y)` Values of x against y.

`hist(x)` Histogram of x.

### Dates

See the **lubridate** package.

# 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).



### Escribes tu código

### Soporte R

Annotations for 'Escribes tu código':

- Abrir en una nueva ventana
- Guardar
- Encontrar y reemplazar
- Compilar como cuaderno
- Ejecutar el código seleccionado
- 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.
- Cambia el tipo de archivo
- Salta a la función en el archivo
- Directorio de trabajo
- Maximiza/minimiza los paneles
- Presiona para ver el historial de comandos
- Arrastra los límites

Annotations for 'Soporte R':

- Historial de comandos
- Importar datos con asistente
- 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
- Muestra los objetos guardados por tipo, con una breve descripción
- Var en el visor de datos
- Ver el código fuente de la función
- Entorno de desarrollo RStudio IDE
- Buscador de archivos. Haga clic en el nombre del archivo o directorio para abrirlo.

## GRÁFICOS

Annotations for 'GRÁFICOS':

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

## PAQUETES

Annotations for 'PAQUETES':

- Instalar paquetes
- Actualizar paquetes
- Crear un paquete reproducible desde tu proyecto.
- Scale Functions for Visualization 0.1.0
- Web Application Framework for R 0.12.2
- Create Dashboards with 'Shiny' 0.5.1
- Márcalo para activar el paquete (i.e. library()) o desactivarlo para desactivarlo (i.e. detach()).
- Versión del paquete instalada
- Eliminar el paquete.

## AYUDA

Annotations for 'AYUDA':

- Pliega de ayuda con links útiles
- Busca dentro de la página de ayuda
- Busca una página de ayuda

## PANEL DE VISIÓN

Annotations for 'PANEL DE VISIÓN':

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

Filter	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
all	all	all	all	all	all
1	5.1	3.5	1.4	0.2	setosa



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

```
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)
```

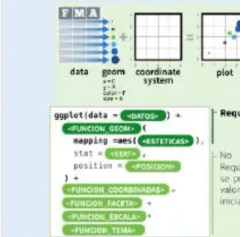
Forma

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

## Gráficos

### Conceptos básicos

PAQUETE ggplot2



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:

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

si no son factores

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

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()
```

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')
```

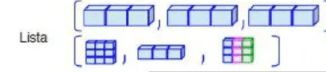
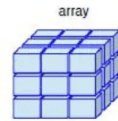
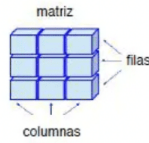
para una función  
para una palabra o frase

SOBRE UN OBJETO

Resumen de la estructura del objeto

```
str(iris)
```

## Tipos de datos



## Listas

CREAR

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

SELECCIONAR

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

## 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
```



## 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)
```

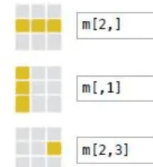
Creado por Rosana Ferrero para Máxima Formación • <https://www.maximainformacion.es> • [consultoria@maximainformacion.com](mailto:consultoria@maximainformacion.com) • 958 32 70 46 (España)  
• Actualizado: 6/18 • CC BY Rosana Ferrero, con material de RStudio, Inc. Para acceder a más guías visite <https://www.rstudio.com/resources/cheatsheets/>

## Matrices

CREAR

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

SELECCIONAR



CARACTERÍSTICAS

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

transpone  
multiplica  
dimensión

## Data frames

CREAR

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

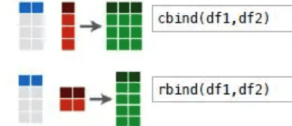
SELECCIONAR



CARACTERÍSTICAS

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

UNIR



From: <https://euloxio.myds.me/dokuwiki/> - Euloxio wiki

Permanent link: [https://euloxio.myds.me/dokuwiki/doku.php/doc:tec:prg:len\\_r:inicio](https://euloxio.myds.me/dokuwiki/doku.php/doc:tec:prg:len_r:inicio)

Last update: 2025/11/04 21:45

