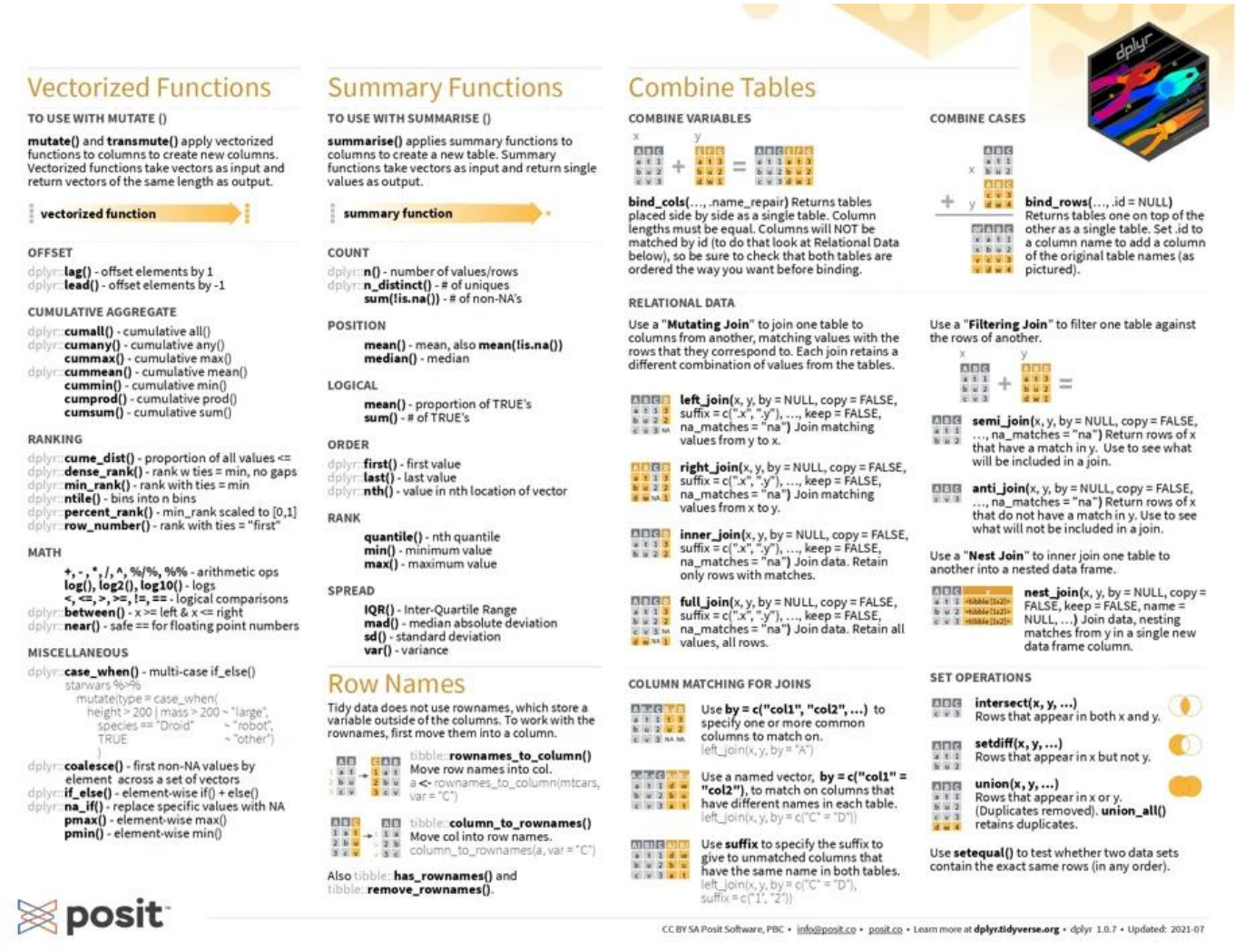
[Programación] R

- 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



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

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

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

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

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

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

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

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

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)
Return x sorted.
rev(x)
Return x reversed.
table(x)
See counts of values.
unique(x)
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)` 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.
`pairwise.t.test` Perform a t-test for paired data.

`prop.test` Test for a difference between proportions.
`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)` Values of x in order.
`plot(x, y)` Values of x against y.
`hist(x)` Histogram of x.

Dates

See the **lubridate** package.

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

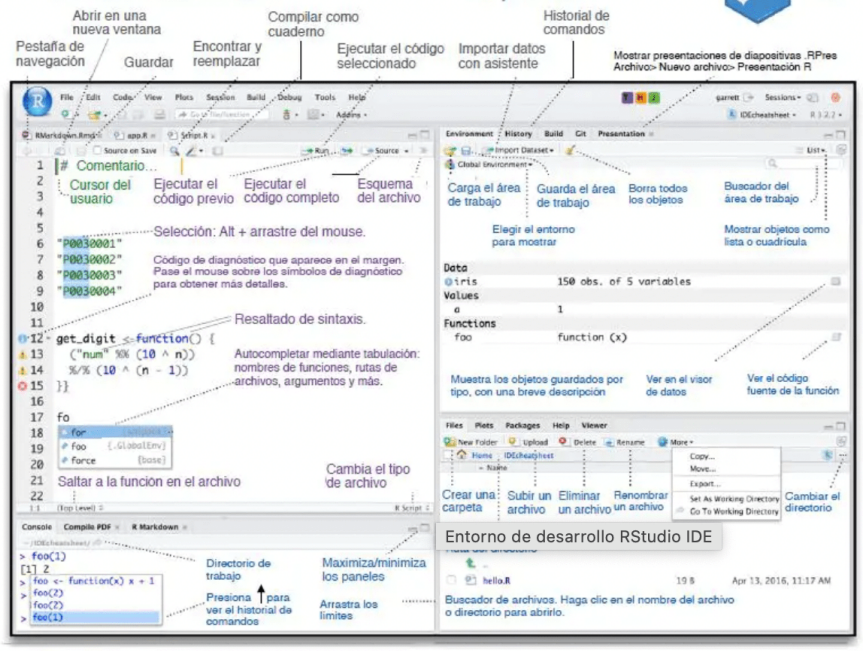
RStudio IDE : : GUÍA RÁPIDA

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

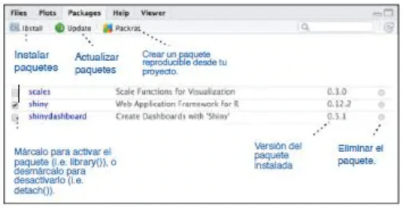


Entorno de desarrollo RStudio IDE

Soporte R

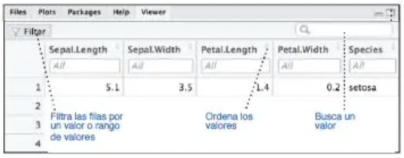
GRÁFICOS

PAQUETES



AYUDA

PANEL DE VISIÓN



Maxima Formación

Traducido por Rosana Ferrero • https://www.maximaformacion.es Para acceder a más guías visite https://www.rstudio.com/resources/cheatsheets/

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)
```

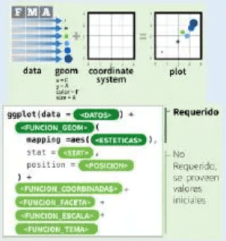
Forma

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

Gráficos

Conceptos básicos

PAQUETE ggplot2



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))
```

Maxima Formacion

Creado por Rosana Ferrero para Máxima Formación • https://www.maximaformacion.es • consultoria@maximainformacion.com • 958 32 70 46 (España) • Actualizado: 6/18