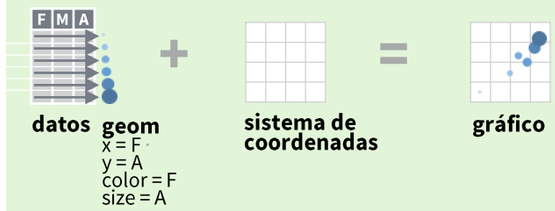
Visualización de datos con ggplot2 : : **GUÍA RÁPIDA**

Conceptos básicos

ggplot2 está basado en la **gramática de gráficos**, la idea de que se puede construir cualquier gráfico a partir de los mismos componentes: **datos**, un **sistema de coordenadas** y **objetos geométricos** (***geom***) —marcas visuales que representan puntos de datos.

Para mostrar los valores, asigna las variables de los datos a las propiedades visuales del geom (***aesthetics***), como tamaño (***size***), **color** y posiciones en **x** e **y**.



Completa esta plantilla para construir un gráfico.

```
ggplot (data = <DATOS>) +                          Requerido
<FUNCIÓN_GEOM> (mapping = aes( <ESTÉTICAS> ),
  stat = <ESTADÍSTICA> ,                           No requerido,
position = <POSICIÓN> ) +                          se proveen
<FUNCIÓN_COORDENADAS> +                            valores
<FUNCIÓN_FACETA> +                                 iniciales
< FUNCIÓN_ESCALA> +                                razonables
< FUNCIÓN_TEMA>
```

ggplot(data = mpg, **aes(**x = cty, y = hwy**))** inicia un gráfico, el cual se finaliza añadiendo capas. Agrega una función geom por capa.

estéticas · datos · geom

qplot(x = cty, y = hwy, data = mpg, geom = "point"**)** crea un gráfico completo con los datos, el geom y las estéticas asignadas. Provee valores iniciales útiles.

last_plot() Devuelve el último gráfico

ggsave("plot.png", width = 5, height = 5) graba el último gráfico como un archivo de imagen de 5' x 5' llamado "plot.png" en el directorio de trabajo. Hace coincidir el tipo de archivo con la extensión indicada.

Geom

Utiliza una función geom para representar puntos de datos y usa las propiedades estéticas para representar variables. Cada función genera una capa.

PRIMITIVAS GRÁFICAS

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

a + geom_blank()
(Util para expandir límites)

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(group = group)**)**
x, y, alpha, color, fill, group, 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

LÍNEAS

Estéticas comunes: 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)**)**

UNA VARIABLE

continua

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

discreta

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

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

DOS VARIABLES

x continua , y continua

e <- ggplot(mpg, aes(cty, hwy))

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

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

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, check_overlap = TRUE**)**, x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

x discreta , y continua

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

x discreta, y discreta

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

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

distribución bivariada continua

h <- ggplot(diamonds, aes(carat, price))

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

h + geom_density2d()
x, y, alpha, colour, group, linetype, size

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

función continua

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

visualizando el 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 (también **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

mapas

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

TRES 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, colour, group, linetype, size, weight

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

Estadísticas (*stat*)

Una forma alternativa de construir una capa. Crea nuevas variables para realizar el gráfico (por ejemplo, ***count*** y ***prop***).

Visualiza un stat modificando el stat predeterminado de un geom, **geom_bar(stat="count")**, o bien utilizando la función stat, **stat_count(geom="bar")**, la cual llama a un geom básico para generar una nueva capa (equivalente a una función geom).
Utiliza la sintaxis **..name..** para mapear variables stat a estéticas (*aes*).

función geom | función stat | estéticas

i + stat_density2d(aes(fill = ..level..), geom = "polygon")

variable creada por stat

c + stat_bin(binwidth = 1, origin = 10**)**
x, y | ..count.., ..ncount.., ..density.., ..ndensity..

c + stat_count(width = 1**) x, y,** | ..count.., ..prop..

c + stat_density(adjust = 1, kernel = "gaussian"**)**
x, y, | ..count.., ..density.., ..scaled..

e + stat_bin_2d(bins = 30, drop = T**)**
x, y, fill | ..count.., ..density..

e + stat_bin_hex(bins=30**) x, y, fill** | ..count.., ..density..

e + stat_density_2d(contour = TRUE, n = 100**)**
x, y, color, size | ..level..

e + stat_ellipse(level = 0.95, segments = 51, type = "t"**)**

l + stat_contour(aes(z = z)**) x, y, z, order** | ..level..

l + stat_summary_hex(aes(z = z), bins = 30, fun = max**)**
x, y, z, fill | ..value..

l + stat_summary_2d(aes(z = z), bins = 30, fun = mean**)**
x, y, z, fill | ..value..

f + stat_boxplot(coef = 1.5**) x, y** | ..lower.., ..middle.., ..upper.., ..width.. , ..ymin.., ..ymax..

f + stat_ydensity(kernel = "gaussian", scale = "area"**) x, y** | ..density.., ..scaled.., ..count.., ..n.., ..violinwidth.., ..width..

e + stat_ecdf(n = 40**) x, y** | ..x.., ..y..

e + stat_quantile(quantiles = c(0.1, 0.9), formula = y ~ log(x), method = "rq"**) x, y** | ..quantile..

e + stat_smooth(method = "lm", formula = y ~ x, se=T, level=0.95**) x, y** | ..se.., ..x.., ..y.., ..ymin.., ..ymax..

ggplot() + stat_function(aes(x = -3:3), n = 99, fun = dnorm, args = list(sd=0.5)**) x** | ..x.., ..y..

e + stat_identity(na.rm = TRUE**)**

ggplot() + stat_qq(aes(sample=1:100), dist = qt, dparam=list(df=5)**) sample, x, y** | ..sample.., ..theoretical..

e + stat_sum() x, y, size | ..n.., ..prop..

e + stat_summary(fun.data = "mean_cl_boot"**)**

h + stat_summary_bin(fun.y = "mean", geom = "bar"**)**

e + stat_unique()

Escalas (*scale*)

Mapea los valores de los datos a los valores visuales de una estética. Para modificar un mapeo, agrega una nueva escala.

(n <- d + geom_bar(aes(fill = fl)))

escala | estética a ajustar | escala del paquete | argumentos de la escala

n + scale_fill_manual(
values = c("skyblue", "royalblue", "blue", "navy"),
limits = c("d", "e", "p", "r"), breaks =c("d", "e", "p", "r"),
name = "fuel", labels = c("D", "E", "P", "R"))

rango de valores a incluir en la estética | título de leyenda/eje | etiquetas de leyenda/eje | intervalos de leyenda/eje

ESCALAS DE USO GENERAL

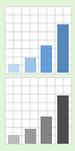
Utilízalas con la mayoría de las estéticas.

scale_*_continuous() – asigna valores continuos a los visuales.
scale_*_discrete() – asigna valores discretos a los visuales.
scale_*_identity() – usa valores de datos como valores visuales
scale_*_manual(values = c()**)** – asigna valores discretos a visuales elegidos manualmente.
scale_*_date(date_labels = "%m/%d"), date_breaks = "2 weeks"**)** – trata a los valores de los datos como fechas.
scale_*_datetime() - trata a los valores de los datos como fecha-hora. Usa los mismos argumentos que scale_x_date(). Ve ?strptime para formatos de etiquetas.

ESCALAS DE LOCALIZACIÓN X E Y

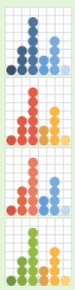
Utilícelas con estéticas x o y (en este caso, x)
scale_x_log10() – usa una escala logarítmica de base 10
scale_x_reverse() – invierte la dirección del eje x
scale_x_sqrt() – usa la escala de raíz cuadrada

ESCALAS DE COLOR Y RELLENO (DISCRETAS)



n <- d + geom_bar(aes(fill = fl)**)**
n + scale_fill_brewer(palette = "Blues"**)**
Opciones de paleta:
ColorBrewer::display.brewer.all()
n + scale_fill_grey(start = 0.2, end = 0.8, na.value = "red"**)**

ESCALAS DE COLOR Y RELLENO(CONTINUAS)



o <- c + geom_dotplot(aes(fill = ..x..))
o + scale_fill_distiller(palette = "Blues"**)**
o + scale_fill_gradient(low="red", high="yellow"**)**
o + scale_fill_gradient2(low="red", high="blue", mid = "white", midpoint = 25)
o + scale_fill_gradientn(colours=topo.colors(6)**)**
También: rainbow(), heat.colors(), terrain.colors(), cm.colors(), RColorBrewer::brewer.pal()

ESCALAS DE FORMA Y TAMAÑO

p <- e + geom_point(aes(shape = fl, size = cyl))
p + scale_shape() + scale_size()
p + scale_shape_manual(values = c(3:7)**)**
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

p + scale_radius(range = c(1,6)**)**
p + scale_size_area(max_size = 6**)**

Sistema de coordenadas

r <- d + geom_bar()

r + coord_cartesian(xlim = c(0, 5)**)**
xlim, ylim
El sistema de coordenadas cartesiano por defecto

r + coord_fixed(ratio = 1/2**)**
ratio, xlim, ylim
Coordenadas cartesianas con una relación de aspecto fija entre unidades de x e y

r + coord_flip()
xlim, ylim
Coordenadas cartesianas volteadas

r + coord_polar(theta = "x", direction=1 **)**
theta, start, direction
Coordenadas polares

r + coord_trans(ytrans = "sqrt"**)**
xtrans, ytrans, limx, limy
Coordenadas cartesianas transformadas.Asigne xtrans y ytrans al nombre de una función ventana.

π + coord_quickmap()
π + coord_map(projection = "ortho", orientation=c(41, -74, 0)**)**projection, xlim, ylim
Asigne proyecciones utilizando el paquete mapproj (mercator (default), azequalarea, lagrange, etc.)

Ajuste de posiciones

Determinan cómo ordenar geoms que, de otra manera, se superpondrían.

s <- ggplot(mpg, aes(fl, fill = drv))

s + geom_bar(position = "dodge")
Ubica los elementos uno al lado del otro.

s + geom_bar(position = "fill")
Apila los elementos uno encima del otro y normaliza la altura.

e + geom_point(position = "jitter")
Agrega ruido aleatorio a las posiciones X e Y de cada elemento para evitar superposición.

e + geom_label(position = "nudge")
Empuja las etiquetas para evitar superposición con los puntos.

s + geom_bar(position = "stack")
Apila los elementos uno encima del otro.

Cada ajuste se puede explicitar como una función con argumentos manuales de ancho (*width*) y alto (*height*).

s + geom_bar(position = position_dodge(width = 1))

Temas

r + theme_bw()
Fondo blanco con cuadrícula.

r + theme_gray()
Fondo gris (tema inicial)

r + theme_dark()
Oscuro, para contrastar.

r + theme_classic()

r + theme_light()

r + theme_linedraw()

r + theme_minimal()
Temas minimalistas.

r + theme_void()
Tema vacío.

Facetas

Dividen el gráfico en subgráficos basados en el valor de una o más variables discretas.

t <- ggplot(mpg, aes(cty, hwy)) + geom_point()

t + facet_grid(cols = vars(fl))
Divide en columnas según *fl*

t + facet_grid(rows = vars(year))
Divide en filas según *year*

t + facet_grid(rows = vars(year), cols = vars(fl)
Divide en filas y columnas

t + facet_wrap(vars(fl))
Divide en una disposición rectangular

Usa **scales** para permitir que los límites de los ejes varíen entre facetas.
t + facet_grid(rows = vars(drv), cols = vars(fl), scales = "free")
Los límites de los ejes x e y se ajustan a cada faceta.
"free_x" – se ajustan los límites del eje x
"free_y" - se ajustan los límites del eje y

Usa **labeller** para ajustar las etiquetas de las facetas

t + facet_grid(cols = vars(fl), labeller = label_both)

fl: c	fl: d	fl: e	fl: p	fl: r

t + facet_grid(rows = vars(fl), labeller = label_bquote(alpha ^ .(fl)**))**

α^c α^d α^e α^p α^r

Etiquetas

t + labs(**x** = "Nueva etiqueta x", **y** = "Nueva etiqueta y"
title ="Título encima del gráfico",
subtitle = "Subtítulo debajo del gráfico",
caption = "Epígrafe debajo del gráfico",
<AES> = "Nuevo título de leyenda <aes>")

Usa escalas para actualizar etiquetas de la leyenda

t + annotate(geom = "text", x = 8, y = 9, label = "A")

geom a agregar | valores manuales para las estéticas del geom

Leyendas

n + theme(legend.position = "bottom"**)**
Coloca la leyenda debajo (*bottom*), encima (*top*), a la izquierda (*left*) o a la derecha (*right*).

n + guides(fill = "none"**)**
Ajusta el tipo de leyenda para cada estética: colorbar, legend, o "none" (sin leyenda).

n + scale_fill_discrete(name = "Título", labels = c("A", "B", "C", "D", "E")**)**
Indica el título y las etiquetas de la leyenda con una función de escala.

Acercamiento

Sin recorte (preferida)
t + coord_cartesian(
xlim = c(0, 100), ylim = c(10, 20)**)**

Con recorte (remueve los datos que no se ven)
t + xlim(0, 100**) + ylim(**10, 20**)**
t + scale_x_continuous(limits = c(0, 100)**) + scale_y_continuous(**limits = c(0, 100)**)**