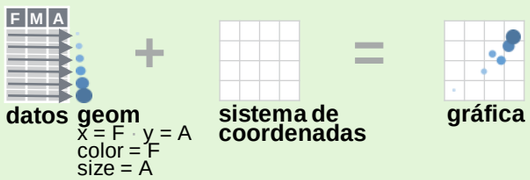
Visualización de datos con ggplot2 : : GUÍA RÁPIDA

Básico

ggplot2 se basa en la gramática de los gráficos, la idea de que se pueden construir todos los gráficos a partir de los mismos componentes: un conjunto de datos, un sistema de coordenadas y geoms, marcas visuales que representan puntos de datos.

Para mostrar valores, asigne variables de los datos a propiedades visuales del geom (estética) como el tamaño, el color y las ubicaciones x e y.



Complete la siguiente plantilla para crear un gráfico.

```
ggplot (data = <DATOS>) +                          ┐ requerido
  <FUNCIÓN_GEOM> (mapping = aes( <MAPEADO> ),      ┘
  stat = <STAT>, position = <POSICIÓN> ) +         ┐
  <FUNCIÓN_COORDENADAS> +                          │
  <FUNCIÓN_FACETADO> +                             │ No requerido, sensible predeterminados suministrados
  <FUNCIÓN_ESCALA> +                               │
  <FUNCIÓN_TEMA>                                   ┘
```

ggplot(data = mpg, **aes(**x = cty, y = hwy**))** Comienza un gráfico al que se termina añadiendo capas. Agregar una función geom por capa.

last_plot() Devuelve la última gráfica.

ggsave("plot.png", width = 5, height = 5**)** Guarda el último gráfico como un archivo de 5' x 5' llamado "plot.png" en el directorio de trabajo. Coincide el tipo de archivo con la extensión del archivo.

Aes Valores estéticos comunes.

color y **fill** - texto ("red", "#RRGGBB")

linetype – entero o texto (0 = "blank", 1 = "solid", 2 = "dashed", 3 = "dotted", 4 = "dotdash", 5 = "longdash", 6 = "twodash")

size - enterp (en mm para el tamaño de los puntos y el texto)

linewidth - entero (en mm para el ancho de líneas)

shape - entero/nombre de la forma o un cáracter ("a")

Geoms

Utilice una función geom para representar puntos de datos, utilice las propiedades estéticas del geom para representar variables. Cada función devuelve una capa.

GRÁFICAS PRIMITIVAS

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

a + geom_blank() y **a + expand_limits()**
Asegúrese de que los límites incluyan valores en todas las gráficas.

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

SEGMENTOS DE LINEA

Estéticas común: 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

ambas continuas

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

una discreta, una 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

ambas discretas

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

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_density_2d()
x, y, alpha, color, group, linetype, size

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

visualización de 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

mapas

Dibuje el objeto geométrico apropiado en función de las características simples presentes en los argumentos de data. aes():
map_id, alpha, color, fill, linetype, linewidth.

nc <- **sf::st_read(**system.file("shape/nc.shp", package = "sf"**))**

ggplot(nc) +
  geom_sf(aes(fill = AREA)**)**

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, 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

Estadísticas

Una forma alternativa de crear una capa.

Una estadística crea nuevas variables para trazar (por ejemplo, recuento, prop).

Visualice una estadística cambiando la estadística predeterminada de una función geom, geom_bar(stat="count") o utilizando una función stat, stat_count(geom="bar"), que llama a una función geom predeterminada para crear una capa (equivalente a una función geom).
Utilice la sintaxis after_stat(nombre) para asignar el nombre de la variable stat a una estética.

geom a usar · función stat · geommapeado

i + stat_density_2d(aes(fill = after_stat(level)), geom = "polygon"**)**

variable creada por stat

c + stat_bin(binwidth = 1, boundary = 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() + xlim(-5, 5**) + stat_function(**fun = dnorm, n = 20, geom = "point"**) x** | x, y

ggplot() + stat_qq(aes(sample = 1:100)**)**
x, y, sample | sample, theoretical

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

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

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

e + stat_identity()

e + stat_unique()

Escalas

Anular los valores predeterminados con el paquete scales.

scales asigna valores de datos a los valores visuales de una estética. Para cambiar una asignación, agregue una nueva escala.

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

scale_ · estética a ajustar · escala preempaquetada usar · argumentos específicos 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 que se incluirán en la asignación · título para usar en la leyenda/eje · etiquetas para usar en la leyenda/eje · saltos para usar en leyenda/eje

ESCALAS DE USO GENERAL

Úselo con la mayoría de las estéticas

scale_*_continuous() - Asignar valores continuos a valores visuales.

scale_*_discrete() - Asignar valores discretos a valores visuales.

scale_*_binned() - Asignación de valores continuos a bins discretos.

scale_*_identity() - Usar valores de datos como visuales.

scale_*_manual(values = c()**)** - Asigne valores discretos a valores visuales elegidos manualmente.

scale_*_date(date_labels = "%m/%d"), date_breaks = "2 weeks"**)** - Tratar los valores de datos como fechas.

scale_*_datetime() - Tratar los valores de datos como fechas y horas.
Igual que scale_*_date(). Vea ?strptime para formatos de etiquetas.

ESCALAS DE UBICACIÓN X E Y

Úselo con la estética x o y (x se muestra aquí)

scale_x_log10() - Trazar x en escala log10.
scale_x_reverse() - Invertir la dirección del eje x.
scale_x_sqrt() - Gráfica x en escala de raíz cuadrada.

ESCALAS DE COLOR Y RELLENO (DISCRETAS)

n + scale_fill_brewer(palette = "Blues"**)**
Para opciones de paleta:
RColorBrewer::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(colors = topo.colors(6)**)**
Además: 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**)**

Sistemas de Coordenadas

r <- d + geom_bar()

r + coord_cartesian(xlim = c(0, 5)**)** - xlim, ylim
El sistema de coordenadas cartesianas predeterminado.

r + coord_fixed(ratio = 1/2**)**
ratio, xlim, ylim - Coordenadas cartesianas con eelación de aspecto fija entre las unidades X e Y.

r + coord_flip()
Invertir las coordenadas cartesianas cambiando asignaciones estéticas X e Y.

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

r + coord_trans(y = "sqrt"**)** - x, y, xlim, ylim
Coordenadas cartesianas transformadas. Asigne a xtrans y ytrans al nombre de una función de ventana.

π + coord_sf() - xlim, ylim, crs. Asegura todas las capas utilizan un sistema de referencia de coordenadas común.

Ajustes de Posición

Los ajustes de posición determinan cómo organizar los geomos que, de otro modo, ocuparían el mismo espacio.

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

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

s + geom_bar(position = "fill")
Apila elementos uno encima de otro, normaliza la altura.

e + geom_point(position = "jitter")
Agregue ruido aleatorio a las posiciones X e Y de cada elemento para evitar el trazado excesivo.

e + geom_label(position = "nudge")
Alejar las etiquetas de los puntos.

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

Cada ajuste de posición se puede reformular como una función con argumentos manuales de anchura y altura:
s + geom_bar(position = position_dodge(width = 1))

Temas

r + theme_bw()
Fondo blanco con líneas de cuadrícula.

r + theme_gray()
Fondo gris (tema predeterminado).

r + theme_dark()
Oscuro para contrastar.

r + theme_classic()

r + theme_light()

r + theme_linedraw()

r + theme_minimal()
Minimal theme.

r + theme_void()
Empty theme.

r + theme() Personalice aspectos del tema, como las propiedades del eje, la leyenda, el panel y las facetas.
r + ggtitle("Title") + theme(plot.title.postion = "plot")
r + theme(panel.background = element_rect(fill = "blue"))

Facetas

Las facetas dividen una gráfica en subgráficas en función de los valores de una o más variables discretas.

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

t + facet_grid(. ~ fl)
Faceta en columnas basada en fl.

t + facet_grid(year ~ .)
Faceta en filas según el año.

t + facet_grid(year ~ fl)
Faceta en filas y columnas.

t + facet_wrap(~ fl)
Envolver facetas en un diseño rectangular.

Establezca escalas para permitir que los límites de los ejes varíen según las facetas.

t + facet_grid(drv ~ fl, scales = "free")
Los límites de los ejes X e Y se ajustan a facetas individuales:
"free_x" - ajuste de los límites del eje X
"free_y" - ajuste de los límites del eje Y

Configurar la etiquetadora para ajustar la etiqueta de faceta:

t + facet_grid(. ~ fl, labeller = label_both)

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

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

α^c α^d α^e α^p α^r

Etiquetas y Leyendas

Usa labs() para etiquetar los elementos de tu gráfica.
t + labs(x = "Nueva etiqueta del eje x",
y = "Nueva etiqueta del eje Y",
title ="Añade un título encima de la gráfica",
subtitle = "Añade un subtítulo debajo del título",
caption = "Agregue una leyenda debajo de la gráfica",
alt = "Añadir texto alternativo a la gráfica",
<AES> = "Nuevo <AES> título de la leyenda"**)**

t + annotate(geom = "text", x = 8, y = 9, label = "A"**)**
Coloca un geom con estéticas seleccionada manualmente.

p + guides(x = guide_axis(n.dodge = 2)**)** Evite las aglomeraciones o etiquetas superpuestas con guide_axis(n.dodge o angle).

n + guides(fill = "none"**)** Establecer el tipo de leyenda para cada estética: colorbar, legend, o none (no leyenda).

n + theme(legend.position = "bottom"**)**
Coloque la leyenda en "bottom", "top", "left", o "right".

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

Zoom

Sin recortes (preferible):
t + coord_cartesian(xlim = c(0, 100), ylim = c(10, 20)**)**

Con recorte (elimina los puntos de datos no vistos):
t + xlim(0, 100**) + ylim(**10, 20**)**
t + scale_x_continuous(limits = c(0, 100)**) + scale_y_continuous(**limits = c(0, 100)**)**