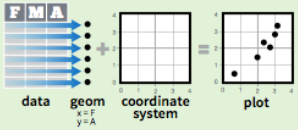
Visualización de Datos con ggplot2

Hoja de Referencia

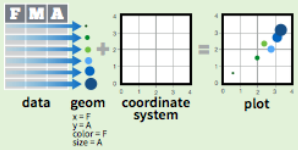
RStudio

Básico

ggplot2 está basado en **grammar of graphics**, la ideas es que pueda construir cada gráfico a partir de unos pocos componentes iguales: unos **datos**, unas **geoms**—marcas visuales que representan los puntos de datos, y un **sistema de coordenadas.**



Para visualizar los datos, hay que mapear las variables de los datos a propiedades estéticas de la geom como **tamaño**, **color**, y las posiciones **x** e **y.**



Construir un gráfico con **qplot()** o **ggplot()**

mapeos estéticos · datos · geom

qplot(x = cty, y = hwy, color = cyl, data = mpg, geom = "point"**)**

Crea un gráfico completo con los datos, geom y mapeos. Proporciona muchos valores por defecto.

ggplot(data = mpg, **aes(**x = cty, y = hwy**))**

Crea un grafico que terminará añadiendo capas. Sin valores por defecto , pero que proporciona más control que qplot().

datos

```
ggplot(mpg, aes(hwy, cty)) +
 geom_point(aes(color = cyl)) +
 geom_smooth(method ="lm") +
 coord_cartesian() +
 scale_color_gradient() +
 theme_bw()
```

añade capas, elementos con +

capa = geom + estadística por defecto + capa de mapeos específicos

elementos adicionales

Añade una nueva capa a un gráfico con las funciones **geom_*()** or **stat_*()**. Cada una proporciona una geom, un conjunto de mapeos estéticos, una estadística por defecto y un ajuste de la posición.

last_plot()

Devuelve el último gráfico

ggsave("plot.png", width = 5, height = 5)

Guarda el último gráfico de 5' x 5' en un fichero con nombre "plot.png" en el directorio de trabajo. Ajusta el tipo de fichero a la extensión.

Geoms - Usa una geom para representar los datos, usa las propiedades estéticas de la geom para representar variables. Cada función devuelve una capa.

Una Variable

Contínua

a <- ggplot(mpg, aes(hwy))

a + geom_area(stat = "bin")
x, y, alpha, color, fill, linetype, size
b + geom_area(aes(y = ..density..), stat = "bin")

a + geom_density(kernel = "gaussian"**)**
x, y, alpha, color, fill, linetype, size, weight
b + geom_density(aes(y = ..county..))

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

a + geom_freqpoly()
x, y, alpha, color, linetype, size
b + geom_freqpoly(aes(y = ..density..))

a + geom_histogram(binwidth = 5**)**
x, y, alpha, color, fill, linetype, size, weight
b + geom_histogram(aes(y = ..density..))

Discreta

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

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

Primitivas Gráficas

c <- ggplot(map, aes(long, lat))

c + geom_polygon(aes(group = group)**)**
x, y, alpha, color, fill, linetype, size

d <- ggplot(economics, aes(date, unemploy))

d + geom_path(lineend="butt", linejoin="round', linemitre=1**)**
x, y, alpha, color, linetype, size

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

e <- ggplot(seals, aes(x = long, y = lat))

e + geom_segment(aes(
xend = long + delta_long,
yend = lat + delta_lat)**)**
x, xend, y, yend, alpha, color, linetype, size

e + geom_rect(aes(xmin = long, ymin = lat,
xmax= long + delta_long,
ymax = lat + delta_lat)**)**
xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

Dos Variables

Continua X, Continua Y

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

f + geom_blank()

f + geom_jitter()
x, y, alpha, color, fill, shape, size

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

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

f + geom_rug(sides = "bl"**)**
alpha, color, linetype, size

f + geom_smooth(model = lm**)**
x, y, alpha, color, fill, linetype, size, weight

f + geom_text(aes(label = cty)**)**
x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

Discreta X, Continua Y

g <- ggplot(mpg, aes(class, hwy))

g + geom_bar(stat = "identity")
x, y, alpha, color, fill, linetype, size, weight

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

g + geom_dotplot(binaxis = "y", stackdir = "center"**)**
x, y, alpha, color, fill

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

Discreta X, Discreta Y

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

h + geom_jitter()
x, y, alpha, color, fill, shape, size

Distribución Bivariada Continua

i <- ggplot(movies, aes(year, rating))

i + geom_bin2d(binwidth = c(5, 0.5)**)**
xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size, weight

i + geom_density2d()
x, y, alpha, colour, linetype, size

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

Función Continua

j <- ggplot(economics, aes(date, unemploy))

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

j + geom_line()
x, y, alpha, color, linetype, size

j + geom_step(direction = "hv"**)**
x, y, alpha, color, linetype, size

Visualizando el error

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

k + geom_crossbar(fatten = 2**)**
x, y, ymax, ymin, alpha, color, fill, linetype, size

k + geom_errorbar()
x, ymax, ymin, alpha, color, linetype, size, width (also **geom_errorbarh()**)

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

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

Mapas

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

l + 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))
m <- ggplot(seals, aes(long, lat))

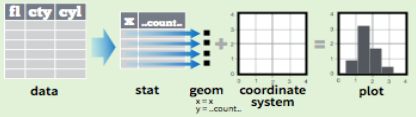
m + geom_contour(aes(z = z)**)**
x, y, z, alpha, colour, linetype, size, weight

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

m + geom_tile(aes(fill = z)**)**
x, y, alpha, color, fill, linetype, size

Stats – Una forma alternativa de crear una capa

Algunos gráficos visualizan una **transformación** de los datos originales. Usar una **stat** (función estadística) para elegir una transformación común a representar, e.g. **a + geom_bar(stat = "bin")**



Cada stat crea variables adicionales para modificar la estética. Esas variables utilizan la sintaxis habitual de **..nombre..**

Las funciones stat y geom combinan una stat con un geom para crear una capa, i.e. **stat_bin(geom="bar")** es equivalente a **geom_bar(stat="bin")**

función estadística | Valores específicos capa | variable generada por transformación

i + stat_density2d(aes(fill = ..level..), geom = "polygon", n = 100)

geom para capa | parámetros para estadística

Distribuciones 1D

a + stat_bin(binwidth = 1, origin = 10)
x, y | ..count.., ..ncount.., ..density.., ..ndensity..
a + stat_bindot(binwidth = 1, binaxis = "x")
x , y, | ..count.., ..ncount..
a + stat_density(adjust = 1, kernel = "gaussian")
x, y, | ..count.., ..density.., ..scaled..

Distribuciones 2D

f + stat_bin2d(bins = 30, drop = TRUE)
x, y, fill | ..count.., ..density..
f + stat_binhex(bins = 30)
x, y, fill | ..count.., ..density..
f + stat_density2d(contour = TRUE, n = 100)
x, y, color, size | ..level..

3 Variables

m + stat_contour(aes(z = z))
x, y, z, order | ..level..
m+ stat_spoke(aes(radius= z, angle = z))
angle, radius, x, xend, y, yend | ..x.., ..xend.., ..y.., ..yend..
m + stat_summary_hex(aes(z = z), bins = 30, fun = mean)
x, y, z, fill | ..value..
m + stat_summary2d(aes(z = z), bins = 30, fun = mean)
x , y, z, fill | ..value..

Comparaciones

g + stat_boxplot(coef = 1.5)
x, y | ..lower.., ..middle.., ..upper.., ..outliers..
g + stat_ydensity(adjust = 1, kernel = "gaussian", scale = "area")
x, y | ..density.., ..scaled.., ..count.., ..n.., ..violinwidth.., ..width..

Funciones

f + stat_ecdf(n = 40)
x, y | ..x.., ..y..
f + stat_quantile(quantiles = c(0.25, 0.5, 0.75), formula = y ~ log(x), method = "rq")
x, y | ..quantile.., ..x.., ..y..
f + stat_smooth(method = "auto", formula = y ~ x, se = TRUE, n = 80, fullrange = FALSE, level = 0.95)
x, y | ..se.., ..x.., ..y.., ..ymin.., ..ymax..

Propósito General

ggplot() + stat_function(aes(x = -3:3), fun = dnorm, n = 101, args = list(sd=0.5))
x | ..y..
f + stat_identity()
ggplot() + stat_qq(aes(sample=1:100), distribution = qt, dparams = list(df=5))
sample, x, y | ..x.., ..y..
f + stat_sum()
x, y, size | ..size..
f + stat_summary(fun.data = "mean_cl_boot")
f + stat_unique()

Escalas (Scales)

Scales controla cómo elabora un gráfico los datos dentro de los valores visuales de una estética (aesthetic). Para cambiar la elaboración, añadir una escala personalizada.

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

scale_ | aesthetic a modificar | Scale predefinida | argumentos específicos

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 mapear | título de leyenda/ejes | Etiquetas de leyenda/ejes | Intervalos en leyenda/ejes

Scales de uso general

Usar con cualquier aesthetic:
alpha, color, fill, linetype, shape, size

scale_*_continuous() – valores continuos a gradación
scale_*_discrete() - valores discretos a gradación
scale_*_identity() - datos **como** valores visuales
scale_*_manual(values = c()) – valores discretos convertidos a una escala elegida a mano

Scales de posición X e Y

Usar con estética para x o y (aquí mostramos x)

scale_x_date(labels = date_format("%m/%d"), breaks = date_breaks("2 weeks")) – Trata los datos de x cómo fecha. Ver ?strptime para formatos
scale_x_datetime() - Trata los datos de x tiempo. Usar los mismos argumentos que scale_x_date()
scale_x_log10() - Representa x en escala log10
scale_x_reverse() – Invierte la dirección del eje x
scale_x_sqrt() – Escala x a raíz cuadrada de x

Escalas de color y relleno

Discretas

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

n + scale_fill_brewer(palette = "Blues")
Para opciones de paleta: library(RcolorBrewer) display.brewer.all()

n + scale_fill_grey(start = 0.2, end = 0.8, na.value = "red")

Continuas

o <- a + geom_dotplot(aes(fill = ..x..))

o + scale_fill_gradient(low = "red", high = "yellow")

o + scale_fill_gradient2(low = "red", hight = "blue", mid = "white", midpoint = 25)

o + scale_fill_gradientn(colours = terrain.colors(6))
También: rainbow(), heat.colors(), topo.colors(), cm.colors(), RColorBrewer::brewer.pal()

Formas para las escalas

p <- f + geom_point(aes(shape = fl))

p + scale_shape(solid = FALSE)

p + scale_shape_manual(values = c(3:7))
Se muestran valores de la forma a la derecha del gráfico

Valores manuales para formas

Tamaño escalas

q <- f + geom_point(aes(size = cyl))

q + scale_size_area(max = 6)
Valor mapeado al área del círculo (no el radio)

Sistemas de Coordenadas

r <- b + geom_bar()

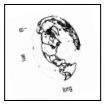
r + coord_cartesian(xlim = c(0, 5))
xlim, ylim
Por defecto sistema coordenadas cartesiano

r + coord_fixed(ratio = 1/2)
ratio, xlim, ylim
Coordenadas cartesianas con proporción fija entre unidades x e y

r + coord_flip()
xlim, ylim
Coordenadas cartesianas invertidas

r + coord_polar(theta = "x", direction=1)
theta, inicio, dirección
Coordenadas polares

r + coord_trans(ytrans = "sqrt")
xtrans, ytrans, limx, limy
Coordenadas cartesianas transformadas.
Ajusta xtrans e ytrans al nombre de la función



z + coord_map(projection = "ortho", orientation=c(41, -74, 0))
projection, orientation, xlim, ylim
Dibuja proyecciones del paquete mapproj (por defecto), azequalarea, lagrange, etc.)

(mercator

Facetas

Las facetas dividen los gráficos en subgráficos a partir de los valores de una o más variables discretas.

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

t + facet_grid(. ~ fl)
divide en columnas a partir de fl

t + facet_grid(year ~ .)
divide en filas a partir year

t + facet_grid(year ~ fl)
divide tanto en filas como columnas

t + facet_wrap(~ fl)
ajusta las facetas de forma rectangular

Ajusta **scales** para variar los límites de los ejes en las facetas

t + facet_grid(y ~ x, scales = "free")
límites de ejes x e y se ajustan a cada faceta
- **"free_x"** - ajusta los límites del eje x
- **"free_y"** - ajusta los límites del eje y

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

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 ^ .(x)))**

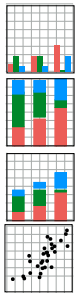
α^{c} α^{d} α^{e} α^{p} α^{r}

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

c	d	e	p	r

Ajustes de Posición

Los ajustes de posición determinan como se ajustan los geoms que de otra forma ocuparían el mismo espacio. s <- ggplot(mpg, aes(fl, fill = drv))



s + geom_bar(position = "dodge")
Ordena una al lado del otro
s + geom_bar(position = "fill")
Coloca los elementos uno encima del otro. Altura normalizada
s + geom_bar(position = "stack")
Coloca elementos uno encima de otro
f + geom_point(position = "jitter")
Añade un ruido aleatorio a la posición de X e Y para evitar imprimir varias veces en el mismo punto

Cada ajuste de posición puede redefinirse como función ajustando manualmente los argumentos **ancho** y **alto**

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

Etiquetas

t + ggtitle("Nuevo Título del Gráfico")
Añade el título principal sobre el gráfico
t + xlab("Nueva etiqueta X")
Cambia la etiqueta del eje X
t + ylab("Nueva etiqueta Y")
Cambia la etiqueta del eje Y
t + labs(title =" New title", x = "New x", y = "New y")
Todo lo de arriba

Usa funciones de escala actualizar las etiquetas de la leyenda

Leyendas

t + theme(legend.position = "bottom")
Emplaza leyenda en "bottom", "top", "left", or "right"
t + guides(color = "none")
Define tipo de leyenda para cada estética: colorbar,legend, o none (sin leyenda)
t + scale_fill_discrete(name = "Title", labels = c("A", "B", "C"))
Define título de leyenda y etiquetas con una función de escala.

Temas

r + theme_bw()
Fondo blanco con líneas de cuadrícula

r + theme_classic()
Fondo blanco sin líneas de cuadrícula

r + theme_grey()
Fondo gris
(tema por defecto)

r + theme_minimal()
Tema mínimo

ggthemes – Paquete con temas adicionales para ggplot2

Ampliación

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

Con recorte (elimina 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))