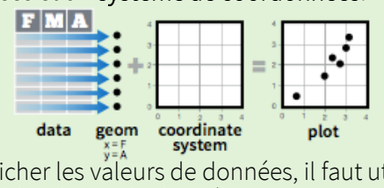
Les Graphiques
avec ggplot2
Aide mémoire

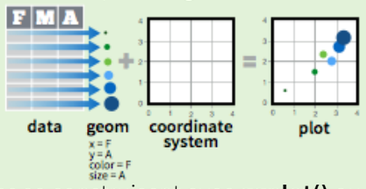
RStudio

Les Bases

ggplot2 est basé sur "**grammar of graphics**", le principe est que vous pouvez construire tous les graphiques à partir d'un même petit nombre d'éléments : un **jeu de données**, un ensemble de **geoms** (repères visuels) qui représentent les points de données et un **système de coordonnées**.



Pour afficher les valeurs de données, il faut utiliser les variables du jeu de données en tant que propriétés esthétiques du geom dans size, color, x et y.



Les graphs se construisent avec **ggplot()** ou **qplot()**

propriétés esthétiques · données · geom

qplot(x = cty, y = hwy, color = cyl, data = mpg, geom = "point"**)**
génère un graphique complet à partir des données, du geom et des propriétés esthétiques passés en paramètres et intègre de nombreux paramètres par défaut très utiles.

ggplot(data = mpg, **aes(**x = cty, y = hwy**))**
initialise un graphique à compléter en ajoutant des calques. Il n'y a pas de calques par défaut, mais cela permet plus de contrôle que qplot().

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

données · ajout de calques avec+ · calque = geom + stat par défaut+ calque spécifique · éléments complémentaires

On ajoute un calque à un graphique avec une fonction **geom_*()** ou **stat_*()**. Chacun génère un geom, un ensemble de propriétés esthétiques et un claque statistique.

last_plot()
Renvoie le dernier graphique

ggsave("plot.png", width = 5, height = 5)
Sauvegarde dans l'espace de travail le dernier graphique affiché dans un fichier "plot.png" de dimension 5' x 5'. Le type de fichier généré dépend directement de l'extension de nom de fichier indiquée.

Geoms - Utiliser un geom pour représenter les points de données, utiliser les propriétés esthétiques du geom pour représenter des variables. Chaque fonction renvoie un calque.

Une variable

Continue

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..))

Discrète

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

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

Éléments graphiques

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

Deux variables

X Continue,Y Continue

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

f + geom_blank()
(Utile pour étendre les limites)

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

X Discrète, Y Continue

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

X Discrète, Y Discrète

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

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

Distribution bivariée continue

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

Fonction continue

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

Marge d'erreur

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

Cartographie

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

l + geom_map(aes(map_id = etat), map = map**) + expand_limits(**x = map$long, y = map$lat**)**
map_id, alpha, color, fill, linetype, size

Trois 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 (rapide)

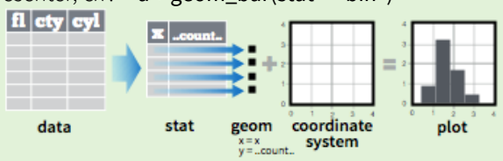
m + geom_tile(aes(fill = z)**)**
x, y, alpha, color, fill, linetype, size (lent)

Plus d'info sur **docs.ggplot2.org** • ggplot2 1.0.0 • Updated: 4/15

Traduit par Vincent Guyader • http://thinkr.fr

Stats - une autre façon de fabriquer un calque

Certains graphiques représentent une transformation du jeu de données originel. Il est possible d'utiliser un calque statistique pour choisir une transformation courante à représenter, ex : a + geom_bar(stat = "bin")



Chaque calque statistique crée des variables supplémentaires qui modifient l'affichage. Ces variables utilisent la syntaxe commune : **..nom..**

Les fonctions stat et geom combinent chacune un stat et un geom pour produire un calque, par exemple : **stat_bin(geom="bar")** revient à **geom_bar(stat="bin")**

fonction stat | calque spécifique | variable créée par transformation

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

geom du calque | Paramètre de stat

Distribution 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..

Distribution 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..

Comparaisons

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..

Fonctions

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..

Usage général

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()

Echelles (Scales)

Les « Scales » déterminent la façon dont un graphique affiche les valeurs de données en accord avec les paramètres esthétiques. Pour modifier le rendu, ajoutez une échelle personnalisée.

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

scale_ | paramètre esthétique à ajuster | scale préconfigurée | arguments spécifique du scale

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"))

plage de valeur à inclure | titre (légende et axes) | libellés (légende et axes) | Découpages (légende et axes)

scales couramment utilisées

A utiliser avec tous paramètres esthétiques: alpha, color, fill, linetype, shape, size

scale_*_continuous() échelle continue
scale_*_discrete() échelle discrète
scale_*_identity() échelle identité
scale_*_manual(values = c()**)** permet de choisir manuellement les valeurs de l'échelle

scales associées à X et Y

A utiliser avec le paramètre esthétique x ou y (exemple ici avec x)

scale_x_date(labels = date_format("%m/%d"), breaks = date_breaks("2 weeks"))** considère les valeurs de x en tant que date. cf ?strptime
scale_x_datetime() considère les valeurs de x en tant que datetime. Utilise les mêmes arguments que scale_x_date().
scale_x_log10() échelle logarithmique pour l'axe x
scale_x_reverse() inverse l'axe des x
scale_x_sqrt() échelle « racine carrée » pour l'axe x

scales de couleur et remplissage

Discrète

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

n + scale_fill_brewer(palette = "Blues")
Pour choisir une palette: library(RcolorBrewer) display.brewer.all()

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

Continue

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

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 = terrain.colors(6)) voir : rainbow(), heat.colors(), topo.colors(), cm.colors(), RColorBrewer::brewer.pal()

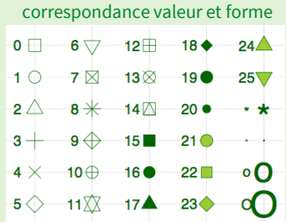
scales de forme

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

p + scale_shape(solid = FALSE)

p + scale_shape_manual(values = c(3:7))
Les formes possibles sont représentées sur le graphique à droite

correspondance valeur et forme



scales de taille

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

q + scale_size_area(max = 6)
Utilise une taille de point proportionnelle à sa valeur

Système de coordonnées

r <- b + geom_bar()

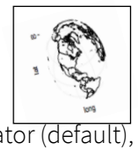
r + coord_cartesian(xlim = c(0, 5)**)**
xlim, ylim
Coordonnées cartésiennes par défaut

r + coord_fixed(ratio = 1/2**)**
ratio, xlim, ylim
Coordonnées cartésiennes à proportion fixe entre x et y

r + coord_flip()
xlim, ylim
Coordonnées cartésiennes inversées

r + coord_polar(theta ="x", direction=1 **)**
theta, start, direction
Coordonnées polaires

r + coord_trans(ytrans = "sqrt"**)**
xtrans, ytrans, limx, limy
Coordonnées cartésiennes transformées. Utilise des fonctions de fenêtrage dans xtrans et ytrans.

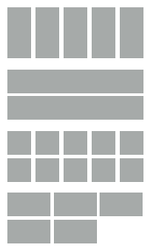


z + coord_map(projection = "ortho", orientation=c(41, 74, 0)**)**
projection, orientation, xlim, ylim
Cartographie du package mapproj (mercator (default), azequalarea, lagrange, etc.)

Vignettage

Permet de diviser un graphique en sous graphiques en fonction d'une ou plusieurs variables discrètes.

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



t + facet_grid(. ~ fl)
découpe en colonne en fonction de fl
t + facet_grid(année ~ .)
découpe en ligne en fonction de année
t + facet_grid(année ~ fl)
découpe en ligne et en colonne
t + facet_wrap(~ fl)
ajuste les vignettes dans un cadre rectangulaire

Utiliser le paramètre **scales** pour autoriser des limites d'échelles différentes entre graphiques

t + facet_grid(y ~ x, scales = "free")
ajuste les limites des axes x et y de chaque graph
- **"free_x"** ajuste les limites de l'axe x
- **"free_y"** ajuste les limites de l'axe y

Utiliser **labeller** pour ajuster les libellées

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}

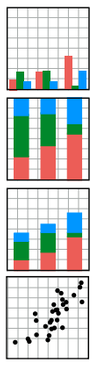
t + facet_grid(. ~ fl, labeller = label_parsed)

c	d	e	p	r

Paramètres de position

Les paramètres de position determinent comment organiser des geoms qui utiliseraient le même espace.

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



s + geom_bar(position = "dodge")
Positionne les éléments les uns à coté des autres
s + geom_bar(position = "fill")
Empile les éléments avec une hauteur normalisée
s + geom_bar(position = "stack")
Empile les éléments
f + geom_point(position = "jitter")
Rajoute de l'aléatoire en x et y sur chaque élément pour éviter les superpositions

Chaque paramètre de position peut être ajusté en tant que fonction avec des arguments **width** et **height**

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

Libellés

t + ggtitle("Nouveau titre du graphique"**)**
Ajoute un titre principal au graphique
t + xlab("titre des abscisses"**)**
Change le libellé des abscisses
t + ylab("titre des ordonnées"**)**
Change le libellé des ordonnées
t + labs(title="Titre", x= "abscisse", y ="ordonnée"**)**
Tout ce qui précède en une seule fonction

Utiliser les fonctions "scale" pour mettre a jour les libellés des légendes

Légende

t + theme(legend.position = "bottom"**)**
Déplace la légende : "bottom", "top", "left", ou "right"
t + guides(color = "none"**)**
Définit le format de la légende pour chaque propriété esthétique : colorbar,legend, ou none (aucune légende)
t + scale_fill_discrete(name = "Titre", labels = c("A", "B", "C"**)**
Précise le titre et le libellé de la légende avec une fonction scale.

Thèmes

r + theme_bw()
Fond blanc avec quadrillage gris

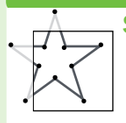
r + theme_classic()
Fond blanc sans quadrillage

r + theme_grey()
Fond Gris (thème par défaut)

r + theme_minimal()
Thème minimaliste

ggthemes - Package avec des thèmes ggplot2 complémentaires

Zoom



Sans coupure (à privilégier)
t + coord_cartesian(xlim = c(0, 100), ylim = c(10, 20)**)**

Avec coupure (supprime les points invisibles)

t + xlim(0, 100**) + ylim(**10, 20**)**
t + scale_x_continuous(limits = c(0, 100)**) + scale_y_continuous(**limits = c(0, 100)**)**

Traduit par Vincent Guyader • http://thinkr.fr