

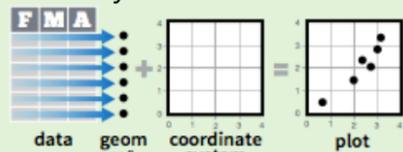
# Les Graphiques avec ggplot2

## Aide mémoire

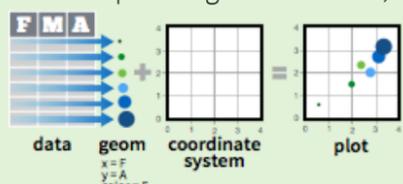


### Les Bases

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().

données

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

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 claqué 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

### 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

**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))

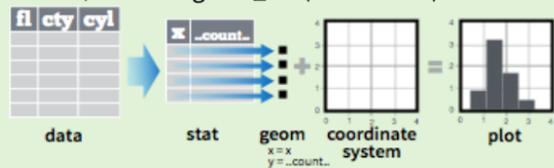
**l + geom\_map**(aes(map\_id = etat), map = map) +  
**expand\_limits**(x = map\$long, y = map\$lat)  
map\_id, alpha, color, fill, linetype, size

**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**)

## Stats - une autre façon de fabriquer un calque

Certains graphiques représentent une transformation du jeu de données original. 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**

**a + stat\_bin(binwidth = 1, origin = 10)**    Distribution 1D  
x, y | ..count.., ..ncount.., ..density.., ..ndensity..  
**a + stat\_bin2d(binwidth = 1, binaxis = "x")**  
x, y, | ..count.., ..ncount..  
**a + stat\_density(adjust = 1, kernel = "gaussian")**  
x, y, | ..count.., ..density.., ..scaled..

**f + stat\_bin2d(bins = 30, drop = TRUE)**    Distribution 2D  
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..

**m + stat\_contour(aes(z = z))**    3 Variables  
x, y, z, order | ..level..  
**m + stat\_spoke(aes(radius = z, angle = z))**  
angle, radius, x, yend, 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..

**g + stat\_boxplot(coef = 1.5)**    Comparaisons  
x, y | ..lower.., ..middle.., ..upper.., ..outliers..  
**g + stat\_ydensity(adjust = 1, kernel = "gaussian", scale = "area")**  
x, y | ..density.., ..scaled.., ..count.., ..n.., ..violinwidth.., ..width..

**f + stat\_ecdf(n = 40)**    Fonctions  
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..

**ggplot() + stat\_function(aes(x = 3:3), fun = dnorm, n = 101, args = list(sd=0.5))**    Usage général  
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))**

**scale\_**    paramètre esthétique à ajuster    **scale** préconfigurée    arguments spécifiques 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    Continue

**n <- b + geom\_bar(aes(fill = fl))**    **o <- a + geom\_dotplot(aes(fill = ..x..))**

**n + scale\_fill\_brewer(palette = "Blues")**    **o + scale\_fill\_gradient(low = "red", high = "yellow")**  
(Pour choisir une palette: library(RcolorBrewer) display.brewer.all())

**n + scale\_fill\_grey(start = 0.2, end = 0.8, na.value = "red")**    **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**    correspondance valeur et forme

**p <- f + geom\_point(aes(shape = fl))**    0 □ 6 ▽ 12 ⊕ 18 ◆ 24 ▲

**p + scale\_shape(solid = FALSE)**    1 ○ 7 ⊗ 13 ⊗ 19 ● 25 ▼

**p + scale\_shape\_manual(values = c(3:7))**    2 △ 8 \* 14 ⊕ 20 ● \* 3 + 9 ◆ 15 ■ 21 ●

Les formes possibles sont représentées sur le graphique à droite

4 × 10 ⊕ 16 ● 22 ■ ○

5 ◇ 11 ⊗ 17 ▲ 23 ◆ ○

**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.)

## Paramètres de position

Les paramètres de position déterminent 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 côté 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))**

## 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

## 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 libellés

**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 <sup>c</sup>	α <sup>d</sup>	α <sup>e</sup>	α <sup>p</sup>	α <sup>r</sup>
----------------	----------------	----------------	----------------	----------------

**t + facet\_grid(. ~ fl, labeller = label\_parsed)**

c	d	e	p	r
---	---	---	---	---

## Libellés

**t + ggtitle("Nouveau titre du graphique")**  
Ajoute un titre principal au graphique

**t + xlab("titre des abscisses")**    **t + ylab("titre des ordonnées")**    Utiliser les fonctions "scale" pour mettre à jour les libellés des légendes

Change le libellé des abscisses

Change le libellé des ordonnées

**t + labs(title = "Titre", x = "abscisse", y = "ordonnée")**  
Tout ce qui précède en une seule fonction

## 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.

## 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))**