

# Visualisation de données avec **ggplot2**

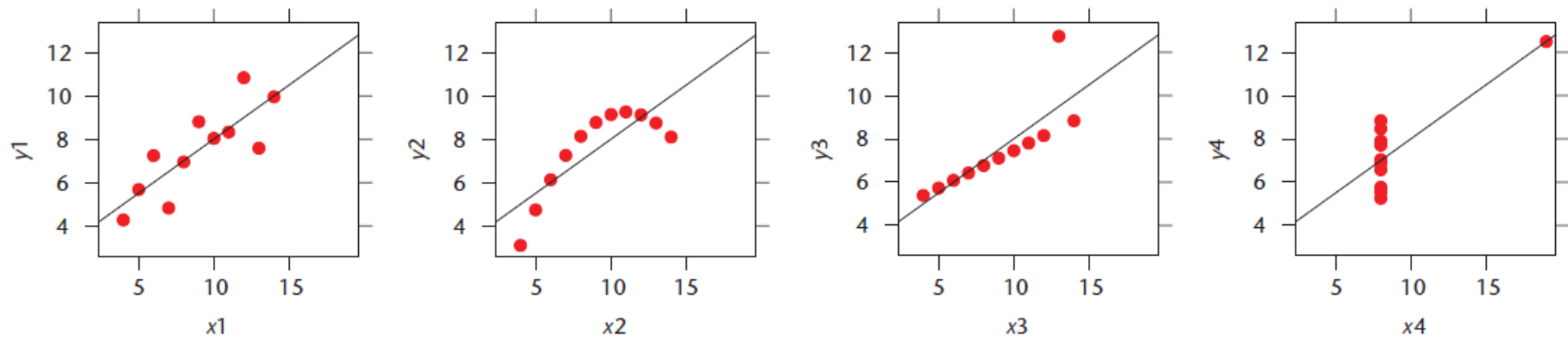
Colloque R à Québec 2019 - Atelier de formation

Sophie Baillargeon

13 mai 2019

# La visualisation, une étape importante

Quatre jeux de données, mêmes paramètres estimés de régression linéaire simple ( $y = 3 + 0.5x$ ), mais relations bien différentes!



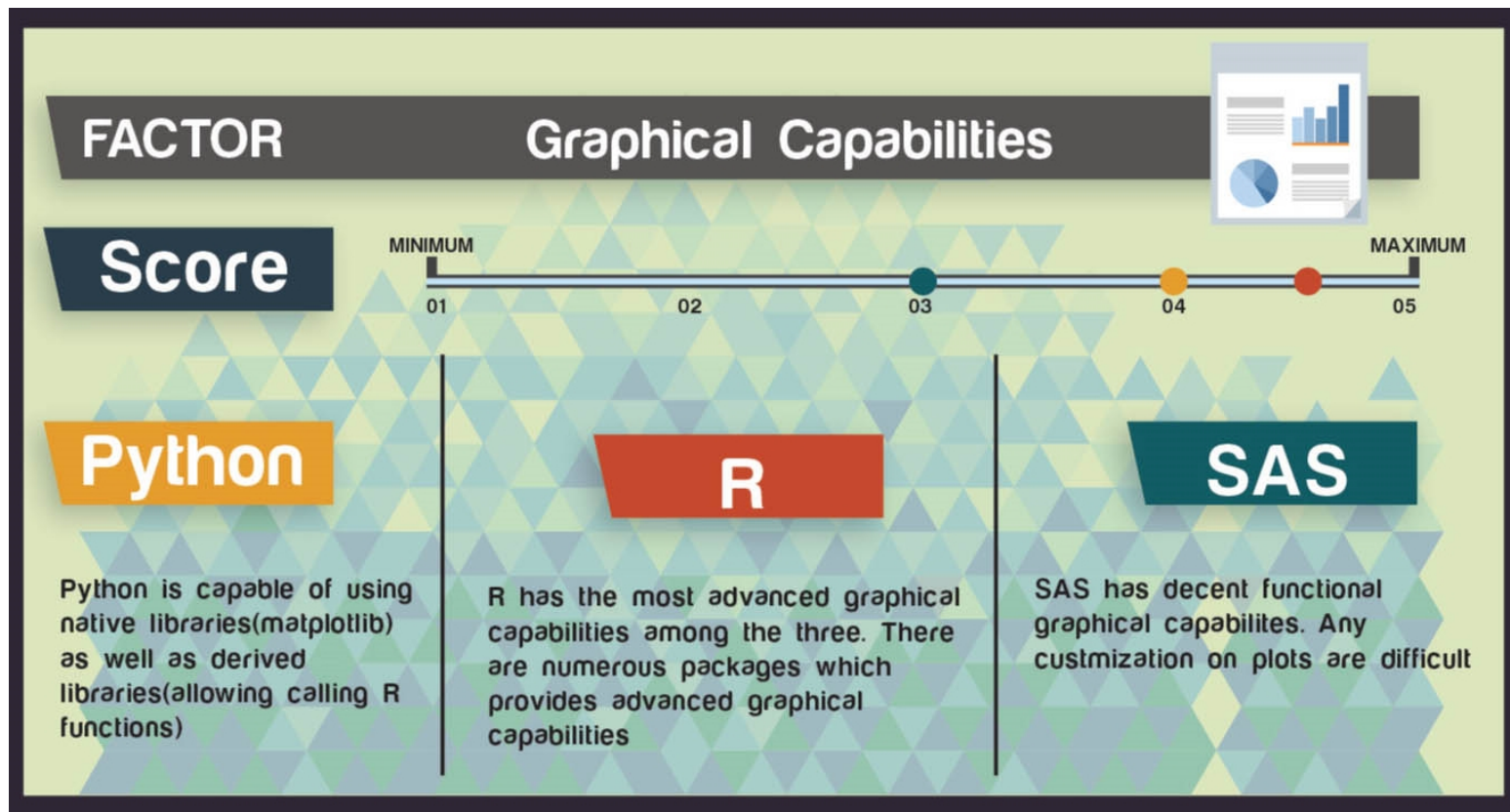
Anscombe, F.J. (1973). Graphs in Statistical Analysis. *The American Statistician*, 27(1). pp. 17-21.

Autres exemples : <https://www.autodeskresearch.com/publications/samestats>

# Les graphiques, une force de R

« R is a free software environment for statistical computing and **graphics**. »

<https://www.r-project.org/>



<https://www.analyticsvidhya.com/blog/2015/05/infographic-quick-guide-sas-python/>

# Systèmes graphiques en R

## 1. Système graphique de base (packages **graphics** et **grDevices**)

- fait partie de l'installation de base de R depuis le début;
- développé par le « **R core team** ».

## 2. Package **lattice**

- publié pour la première fois en 2001;
- développé par **Deepayan Sarkar** (<https://CRAN.R-project.org/package=lattice>);
- basé sur le « **trellis** » système de S-PLUS, qui implémente le système graphique présenté dans : Cleveland, W. S. (1993). *Visualizing data*. Hobart Press.

## 3. Package **ggplot2**

- publié pour la première fois en 2006 sous le nom de **ggplot**;
- amélioré de façon importante et renommé **ggplot2** en 2007;
- créé par **Hadley Wickham** (créateur) et maintenant développé par toute une équipe (<https://CRAN.R-project.org/package=ggplot2>);
- implémente la **grammaire graphique** présentée dans : Wilkinson, L. (2005). *The grammar of graphics*, Second Edition. Springer.

# Système graphique de base

Procédure de création d'un graphique :

1. La configuration de paramètres graphiques (au besoin) :

- énoncé par ou layout.

2. L'initialisation d'un graphique :

- fonction générique de base : plot (choisit un graphique pertinent à produire selon ce qu'elle reçoit en entrée),
- ou fonction pour un type spécifique de graphiques : pairs, matplot, pie, barplot, mosaicplot, hist, boxplot, qqnorm, qqplot, curve, etc.

3. L'ajout séquentiel d'éléments au graphique (au besoin) :

- fonctions d'ajouts à un graphique déjà initialisé :
  - points, matpoints, lines, matlines, abline, segments, arrows, rect, polygon, legend, text, mtext, title, axis, box, qqline, etc.;
  - matplot, barplot, hist, boxplot, curve avec l'argument `add = TRUE`.

Arguments et paramètres graphiques ⇒ mise en forme et annotations

Pour en savoir plus : [https://stt4230.rbind.io/communication\\_resultats/graphiques\\_r/](https://stt4230.rbind.io/communication_resultats/graphiques_r/)

# Exemple graphique - système de base

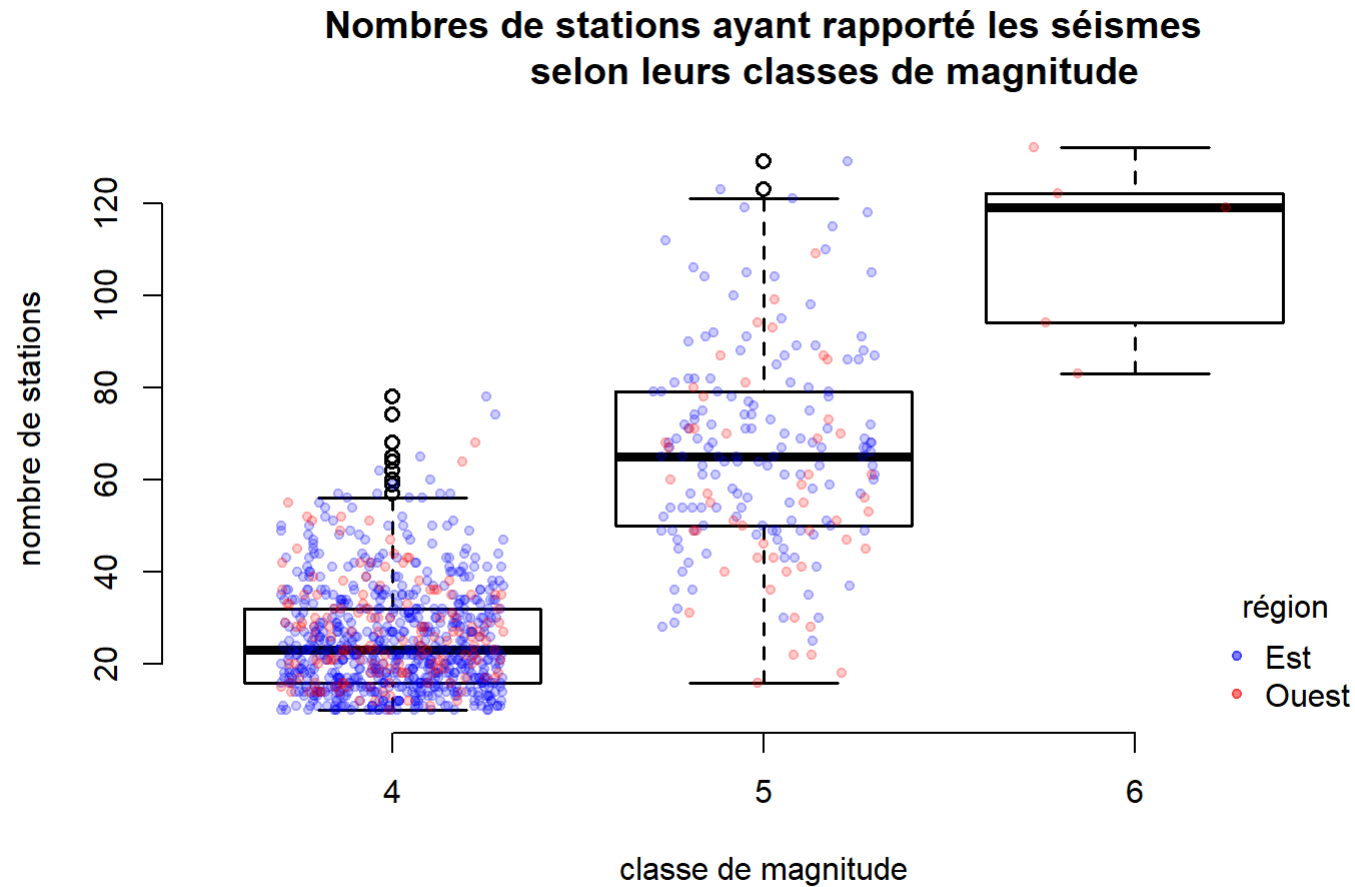
```
# 1. Configuration de paramètres graphiques
par.default <- par(bty = 'n')

# 2. Initialisation d'un graphique (contenant ici des boxplots)
plot(stations ~ magFactor, data = quakes, lwd = 1.5,
     main = "Nombres de stations ayant rapporté les séismes
           selon leurs classes de magnitude",
     xlab = "classe de magnitude", ylab = "nombre de stations")

# 3. Ajout séquentiel d'éléments au graphique
# Ajout de diagrammes de dispersion 1D par région
stripchart(stations ~ magFactor, data = quakes, subset = region == "Est",
           method = "jitter", jitter = 0.3, pch = 20,
           col = rgb(0,0,1,0.2), add = TRUE, vertical = TRUE)
stripchart(stations ~ magFactor, data = quakes, subset = region == "Ouest",
           method = "jitter", jitter = 0.3, pch = 20,
           col = rgb(1,0,0,0.2), add = TRUE, vertical = TRUE)
# Ajout d'une légende
legend(x = "bottomright", pch = 20, bty = "n",
      col = c(rgb(0,0,1,0.5), rgb(1,0,0,0.5)),
      legend = c("Est", "Ouest"), title = "région")

# Réattribution des valeurs par défaut aux paramètres graphiques modifiés
par(par.default)
```

# Exemple graphique - système de base



# Données utilisées dans les exemples

Data frame `quakes` du package `datasets` : observations pour 1000 évènements sismiques survenus près des îles Fidji depuis 1964

- `lat` et `long` : localisation géographique du séisme en latitude-longitude,
- `depth` et `mag` : [profondeur](#) et [magnitude](#) du séisme,
- `stations` : nombre de stations sismiques ayant rapporté l'évènement.

*# Variables catégoriques ajoutées :*

```
quakes$magFactor <- factor(floor(quakes$mag))           # classe de magnitude
quakes$region <- factor(quakes$long >= 175, labels = c("Ouest", "Est")) # région
```

```
str(quakes)
```

```
## 'data.frame':    1000 obs. of  7 variables:
## $ lat      : num  -20.4 -20.6 -26 -18 -20.4 ...
## $ long     : num   182 181 184 182 182 ...
## $ depth    : int   562 650 42 626 649 195 82 194 211 622 ...
## $ mag      : num    4.8 4.2 5.4 4.1 4 4 4.8 4.4 4.7 4.3 ...
## $ stations : int    41 15 43 19 11 12 43 15 35 19 ...
## $ magFactor: Factor w/ 3 levels "4","5","6": 1 1 2 1 1 1 1 1 1 1 ...
## $ region   : Factor w/ 2 levels "Ouest","Est": 2 2 2 2 2 2 1 2 2 2 ...
```

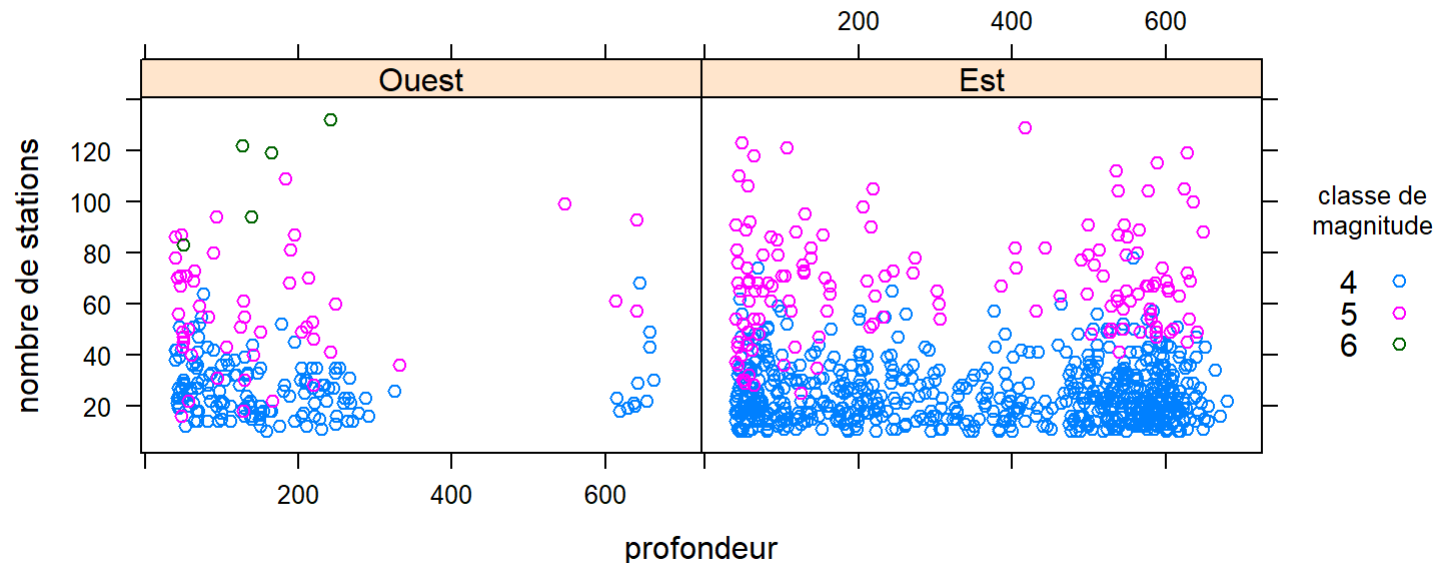


# Package lattice

Force = représentations multivariées

```
library(lattice)
xyplot(stations ~ depth | region, data = quakes, groups = magFactor,
       main = "Nombres de stations ayant rapporté les séismes selon différentes variables",
       xlab = "profondeur", ylab = "nombre de stations",
       auto.key = list(title = "classe de\nmagnitude", space = "right", cex.title = 0.8))
```

**Nombres de stations ayant rapporté les séismes selon différentes variables**



# Package ggplot2

Conçu en ayant comme objectif la **simplicité d'utilisation** et la **qualité des graphiques produits**

- Forces reprises des systèmes graphiques R précédents :
  - système de base : création de graphiques par **couches** (ajouts séquentiels d'éléments);
  - package **lattice** : **représentations multivariées** simples.
- Améliorations apportées aux systèmes graphiques R précédents :
  - fournir une **esthétique** par défaut pensée de façon à transmettre plus efficacement les informations contenues dans le graphique (ex. choix de couleur),
  - **automatiser certaines configurations** graphiques (ex. les légendes),
  - **faciliter l'ajout de transformations statistiques** communes (ex. courbes de lissage, barres d'erreur).

# Plan de l'atelier

1. Premiers pas avec `ggplot2`
2. Apprendre par des exemples
3. Packages souvent utilisés avec `ggplot2` et extensions de `ggplot2`
4. Conclusion

## Fichiers

- Présentation : `Visualisation_ggplot2_RaQ2019.pdf`
- Exemples : `Visualisation_ggplot2_RaQ2019_Exemples.Rmd`  
(ou version HTML)
- Exercices : `Visualisation_ggplot2_RaQ2019_Exercices.html`

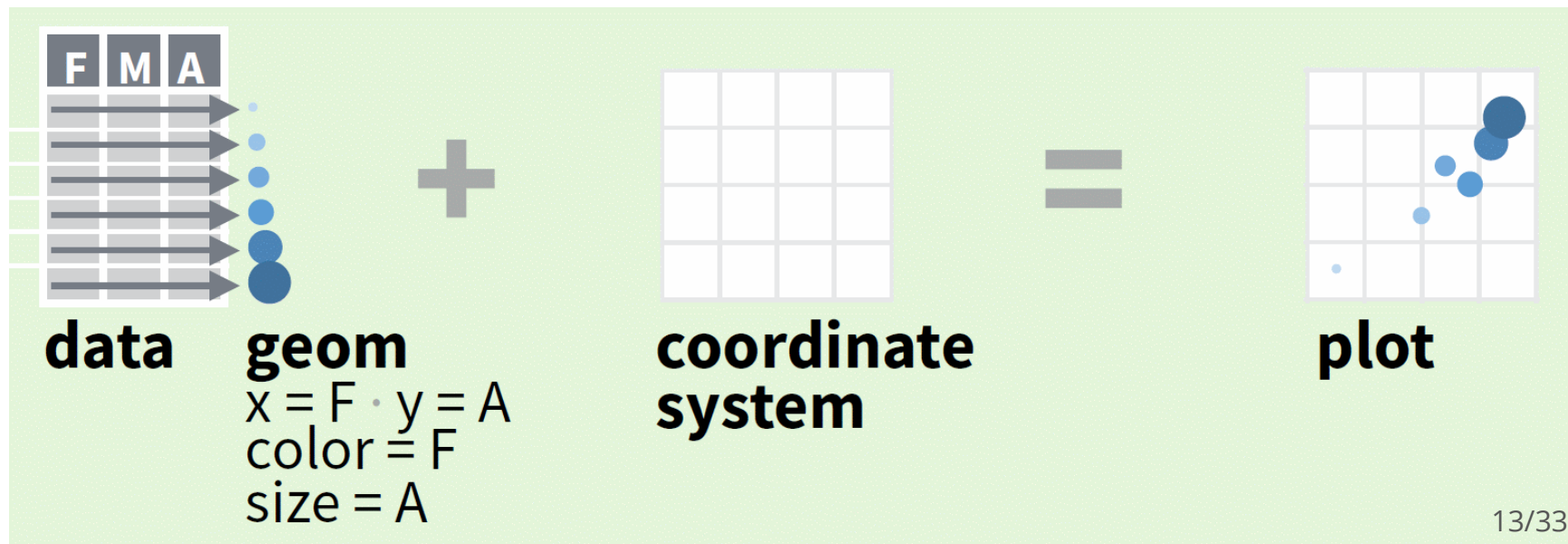
Premiers pas avec **ggplot2**

# La grammaire graphique

Graphique statistique = représentation de **données**, dans un **système de coordonnées** spécifique, divisée en éléments de base :

- **éléments géométriques** (*geoms*) : points, lignes, barres, etc.;
- **propriétés visuelles** (*aesthetics*) des éléments géométriques : axes, couleurs, formes, tailles, etc.
- **transformations statistiques**, si désirées : courbe de lissage, région d'erreur, etc.

Un graphique est spécifié en **associant des variables**, provenant des données, à des **propriétés visuelles** des éléments géométriques du graphique.



# Survol des principales fonctions

- `ggplot` : initialisation d'un objet de classe `ggplot`
- `qplot` : initialisation rapide (q pour *quick*) d'un objet `ggplot`
  - Utilisation plus intuitive que `ggplot` pour des gens familiers avec `plot`.
  - N'offre cependant pas toutes les possibilités de `ggplot`.
  - *Ne sera pas couvert ici.*
- `+` : opérateur pour l'ajout de couches ou la modification de configurations dans un objet `ggplot`
- fonctions de type `geom_` (p.ex. `geom_point`, `geom_boxplot`, `geom_bar`, etc.) : spécification de couches à ajouter à un graphique
- `aes` : création d'un `mapping`, soit une association entre des propriétés visuelles et des variables
- `ggsave` : enregistrement d'un graphique

Produire un graphique = afficher un objet `ggplot`

# Code gabarit minimaliste

```
ggplot(data = <DATA>) +  
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

- éléments entre < et > à remplacer :
  - <DATA> = jeu de données, obligatoirement stocké dans un **data frame**, dans lequel les **variables catégoriques** doivent être des vecteurs de chaînes de caractères ou des **facteurs** (dont **les libellés des niveaux ont avantage à être informatifs**, car ils apparaîtront dans le graphique);
  - <GEOM\_FUNCTION> = le nom d'une fonction de type **geom\_** pour ajouter une couche d'éléments géométriques au graphique;
  - <MAPPINGS> = arguments à fournir à la fonction **aes**, (les arguments acceptés varient un peu selon la <GEOM\_FUNCTION>);

Source : <http://r4ds.had.co.nz/data-visualisation.html#a-graphing-template>

Les arguments `data` et `mapping` peuvent être fournis :

- à la fonction `ggplot` : leur valeur est alors commune à toutes les couches (il est tout de même possible de forcer l'utilisation d'autres données ou d'autres associations pour des couches spécifiques);
- ou à une `<GEOM_FUNCTION>` : leur valeur est alors spécifique à la couche produite.

## Syntaxes équivalentes au gabarit minimaliste

*# Version « tout sur la même ligne » du gabarit :*

```
ggplot(data = <DATA>) + <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

Étant donné que le graphique produit par ce gabarit ne comprend qu'une seule couche, les syntaxes suivantes sont équivalentes à celle du gabarit.

*# argument data spécifique (fourni dans l'appel à la <GEOM\_FUNCTION>)*

```
ggplot() + <GEOM_FUNCTION>(data = <DATA>, mapping = aes(<MAPPINGS>))
```

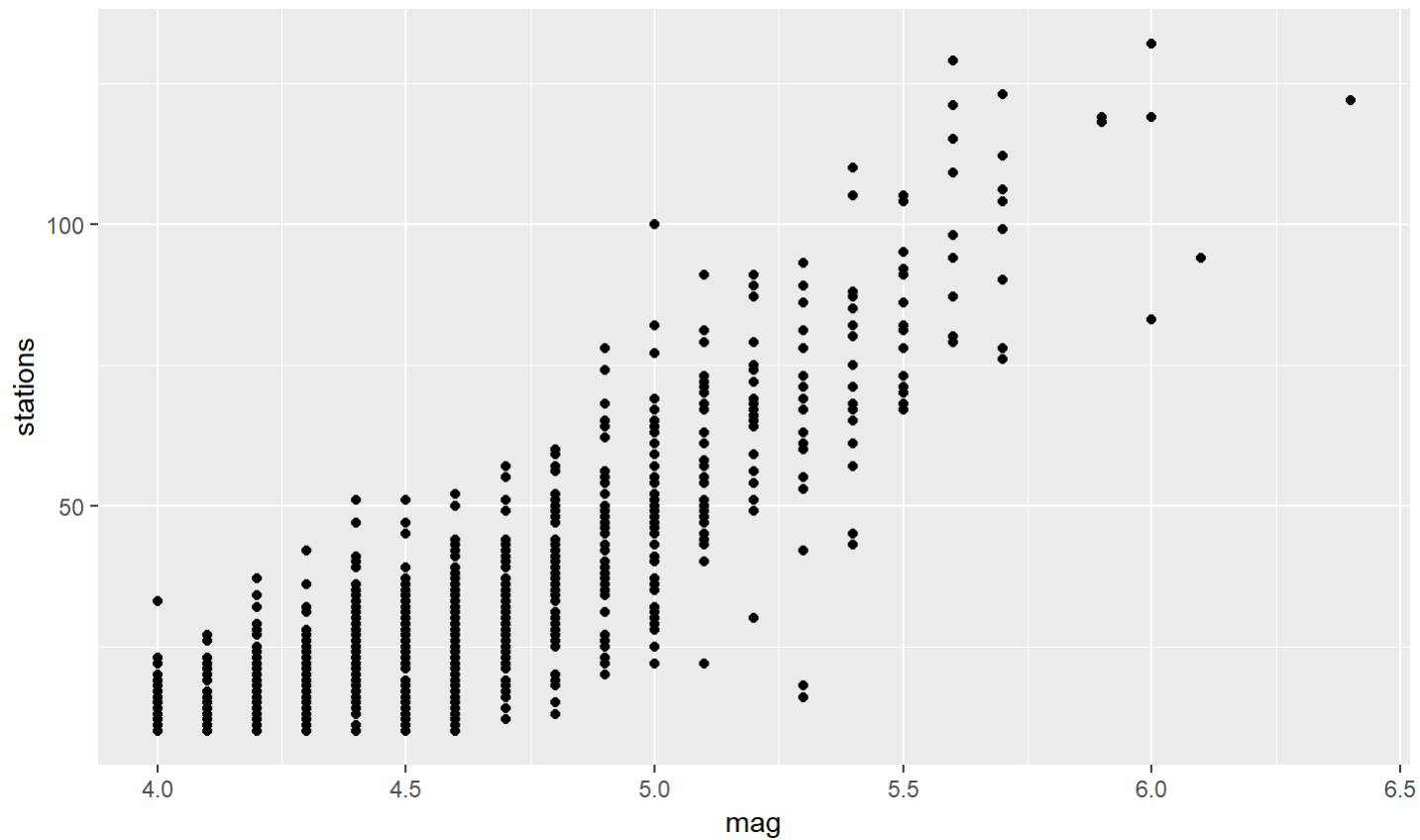
*# argument mapping global (fourni dans l'appel à ggplot)*

```
ggplot(data = <DATA>, mapping = aes(<MAPPINGS>)) + <GEOM_FUNCTION>()
```



# Premier exemple

```
library(ggplot2)
ggplot(data = quakes) + geom_point(mapping = aes(x = mag, y = stations))
```



# Objet de classe `ggplot`

Un graphique produit en `ggplot2` est en fait un objet de classe `ggplot`.

On commence par l'initialiser :

```
monGraph <- ggplot(data = <DATA>)
```

puis y ajouter des couches :

```
monGraph <- monGraph + <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

on peut observer le contenu de l'objet :

```
str(monGraph)  
summary(monGraph)
```

et enfin produire un graphique à partir de l'objet, simplement en l'affichant :

```
monGraph # ou print(monGraph)
```

# Quelques fonctions de type `geom_`

Fonction	Type de graphique	Élément(s) ajouté(s)
<code>geom_point</code>	diagramme de dispersion	points selon des coordonnées
<code>geom_line</code>	diagramme en lignes	segments de droites reliant des points
<code>geom_bar</code>	diagramme à barres	barres disjointes, de hauteurs spécifiées ou calculées = fréquences des niveaux d'un facteur
<code>geom_histogram</code>	histogramme	barres collées, de hauteurs calculées = fréquences d'observations d'une variable numérique tombant dans des intervalles joints ( <i>bin</i> )
<code>geom_boxplot</code>	diagramme en boîte	<i>boxplots</i>
<code>geom_density</code>	courbe de densité à noyau	courbe de la densité estimée par noyau ( <i>kernel density</i> )
:	il en existe plusieurs autres	voir <a href="http://ggplot2.tidyverse.org/reference/">http://ggplot2.tidyverse.org/reference/</a>

---

La plupart de ces fonctions cachent des transformations statistiques (ex. `geom_bar` et `geom_histogram` calculent des fréquences, `geom_boxplot` calcule des quantiles, `geom_density` estime une densité, etc.)

# Quelques autres fonctions

Pour ajouter des couches ou modifier des configurations dans un objet **ggplot** initialisé :

- fonctions **labs**, **ggtitle**, **xlab**, **ylab** : ajouter un titre, modifier les noms d'axes;
- fonctions de type **coord\_** : modifier des configurations liées au système de coordonnées;
- fonctions de type **facet\_** : créer des panneaux (*panel*) de graphiques côte-à-côte
  - chacun des sous-graphiques est conditionnel à la valeur de facteurs(s), il représente donc seulement le sous-ensemble des observations ayant une modalité particulière pour ce(s) facteur(s).
- fonctions de type **theme\_** : modifier des configurations liées à l'apparence du graphique;
- fonctions de type **scale\_** : modifier les échelles de certaines propriétés visuelles (ex. couleurs, formes, tailles, etc.)

Il y a beaucoup d'autres fonctions! <http://ggplot2.tidyverse.org/reference/>

# Code gabarit plus complet

Complete the template below to build a graph.

**ggplot (data = <DATA>) +**

**<GEOM\_FUNCTION> (mapping = aes(<MAPPINGS>),**

**stat = <STAT>, position = <POSITION>) +**

**<COORDINATE\_FUNCTION> +**

**<FACET\_FUNCTION> +**

**<SCALE\_FUNCTION> +**

**<THEME\_FUNCTION>**

**required**

Not  
required,  
sensible  
defaults  
supplied

<https://github.com/rstudio/cheatsheets/blob/master/data-visualization-2.1.pdf>

# Démarche de création d'un graphique

## 1. Planifier le travail

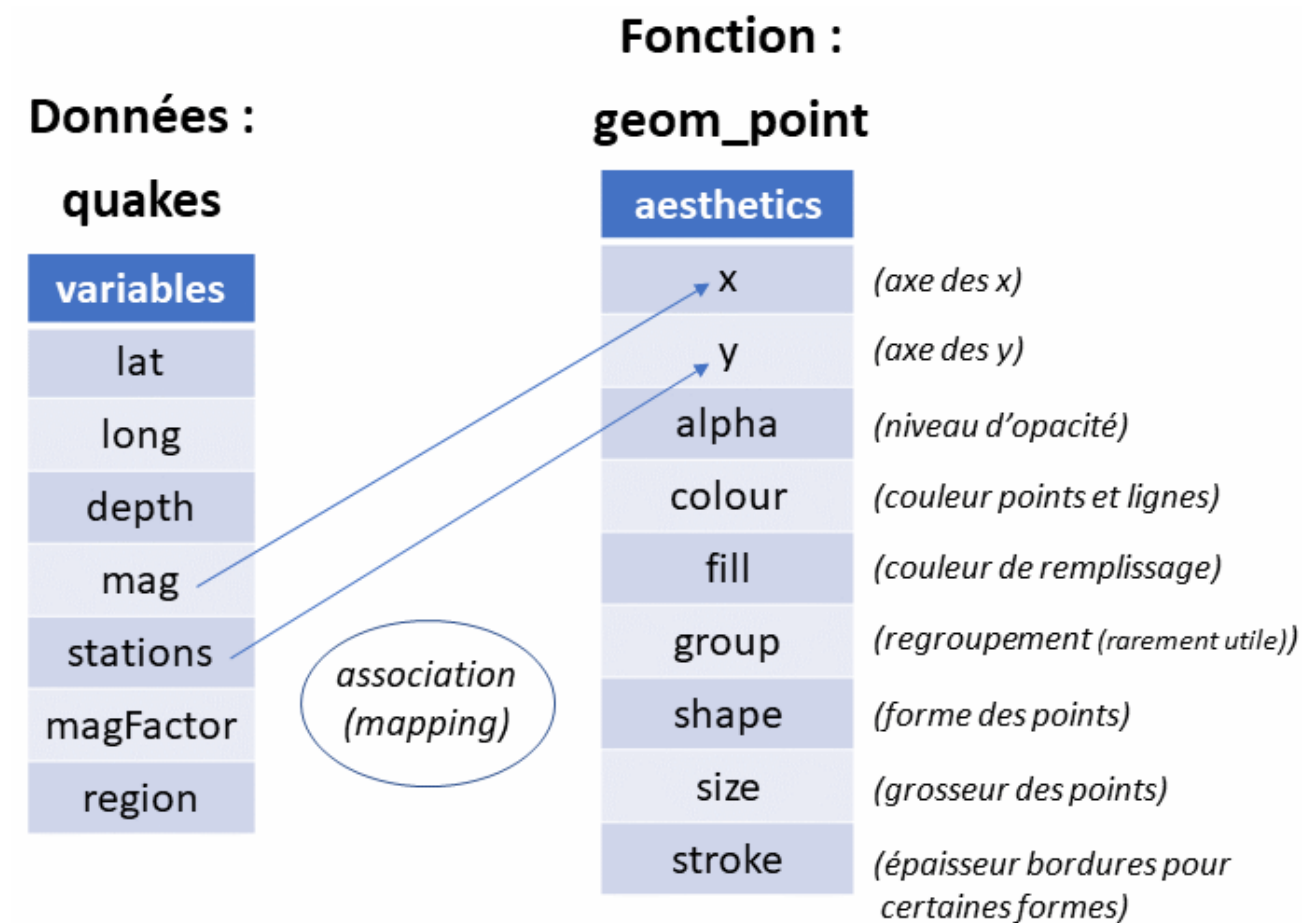
1. Je veux représenter **quelles variables**, de quel jeu de données ?
2. Je veux créer **quel type de graphique** ?
3. **Quelle fonction de type `geom_`** me permettra de produire les éléments géométriques de ce graphique ? (voir <http://ggplot2.tidyverse.org/reference/>)
4. Cette fonction accepte **quelles propriétés visuelles** ?  
(voir <http://ggplot2.tidyverse.org/reference/>)
5. Je veux **associer** les variables concernées à quelles propriétés visuelles ?

## 2. Écrire et soumettre la **première version** du code de création du graphique

## 3. Modifier le code de création du graphique pour **ajuster la mise en forme** (titre, nom d'axes, autres annotations, palette de couleur, etc.) selon mes besoins

- travail itératif (cycle « modification code » → « jugement du graphique créé » répété) jusqu'à l'obtention d'un résultat satisfaisant

# Exemple de planification



[https://ggplot2.tidyverse.org/reference/geom\\_point.html](https://ggplot2.tidyverse.org/reference/geom_point.html)

Apprendre par des exemples



# Exemples de graphiques avec ggplot2

## Flipbook

[https://evamaerey.github.io/ggplot\\_flipbook/ggplot\\_flipbook\\_xaringan.html#1](https://evamaerey.github.io/ggplot_flipbook/ggplot_flipbook_xaringan.html#1)

L'impact de chaque ajout au code de création du graphique peut être visualisé.

## Fichier d'exemples

`Visualisation_ggplot2_RaQ2019_Exemples.Rmd`

Exemples que j'ai préparés, faisant intervenir divers types de graphiques, pour illustrer des possibilités de ggplot2.

Packages souvent utilisés avec  
**ggplot2** et extensions de  
**ggplot2**

# Le Tidyverse

ggplot2 fait partie du Tidyverse : <https://www.tidyverse.org/>

Il est souvent utilisé conjointement à d'autres packages de ce regroupement de packages, notamment :

- `dplyr` pour la manipulation / le prétraitement de données
  - <https://dplyr.tidyverse.org/>
- `magrittr` qui offre l'opérateur « pipe » `%>%` :
  - <https://magrittr.tidyverse.org/>
  - introduction à l'opérateur `%>%` :  
[https://stt4230.rbind.io/amelioration\\_code/bonnes\\_pratiques\\_r/#operateur-pipe](https://stt4230.rbind.io/amelioration_code/bonnes_pratiques_r/#operateur-pipe)
- `forcats` pour la manipulation de facteurs :
  - <https://forcats.tidyverse.org/>
  - (la fonction `fct_rev` de ce package a été utilisée dans un exemple)

# Extensions

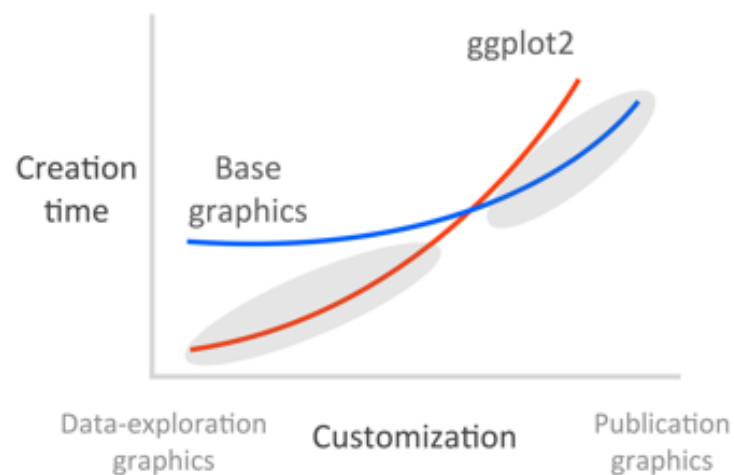
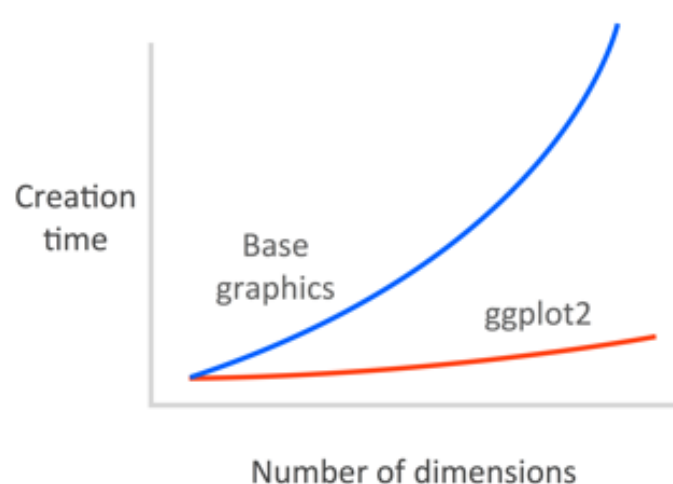
Un nombre important d'extensions de `ggplot2` sont offertes. Plusieurs sont énumérées sur le site web suivant : <http://www.ggplot2-exts.org/gallery/>

Quelques extensions intéressantes :

- `ggthemes` : thèmes complets supplémentaires;
- `ggrepel` : fonction `geom_text_repel()` pour éviter les textes superposés;
- `cowplot` : thème adapté aux publications scientifiques et possibilité de combiner plusieurs graphiques en un;
- `ggExtra` : ajout facile d'histogrammes marginaux;
- `GGally` : matrice de diagrammes de dispersion (équivalent de `pairs`) et plus;
- `ggiraph` : graphiques interactifs (format HTML);
- `gganimate` : gif animés (ou autres formats d'animation);
- ajout de cartes en arrière-plan :
  - `maps` et `mapdata` (s'utilisent aussi avec le système graphique de base),
  - `ggmap` : intégration de données Google Maps (mais requière un compte Google avec facturation activée);
- etc.

# Conclusion

# Comparaison ggplot2 versus système graphique R de base



<http://seananderson.ca/ggplot2-FISH554/>

# Références sur ggplot2

Quelques sites web (il y en a bien plus!) :

- Page web sur le CRAN : <https://CRAN.R-project.org/package=ggplot2>
- Documentation en ligne : <http://ggplot2.tidyverse.org/>
- Feuille de triche (cheat sheet) : <https://github.com/rstudio/cheatsheets/blob/master/data-visualization-2.1.pdf>
- Flipbook : [https://evamaerey.github.io/ggplot\\_flipbook/ggplot\\_flipbook\\_xaringan.html#1](https://evamaerey.github.io/ggplot_flipbook/ggplot_flipbook_xaringan.html#1)
- Exemples de graphiques (tous systèmes, mais beaucoup de `ggplot2`) avec leur code source permettant de les reproduire: <http://www.r-graph-gallery.com/>
- Extensions de `ggplot2` : <https://www.ggplot2-exts.org>
- Extension et tutoriel de la BBC : <https://bbc.github.io/rcookbook/>
- Vidéos de Roger Peng concernant `ggplot2` :
  - <https://www.youtube.com/watch?v=HeqHMM4ziXA>
  - <https://www.youtube.com/watch?v=n8kYa9vu1l8>

- Livres :

- Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis*, Second Edition. Springer. URL pour le code R : <https://github.com/hadley/ggplot2-book>
- Wickham, H. et Grolemund, G. (2016). *R for Data Science*. O'Reilly Media, Inc. Chapitre 3. URL <http://r4ds.had.co.nz/>
- Wilkinson, L. (2005). *The grammar of graphics*, Second Edition. Springer.
- Chang, W. (2012). *R Graphics Cookbook: Practical Recipes for Visualizing Data*. O'Reilly Media, Inc. URL pour le code R : <http://www.cookbook-r.com/Graphs/>

Pour en apprendre davantage sur le système de base :

- Notes de cours (traitent principalement du système graphique R de base) :
  - Baillargeon, S. (2019). *Graphiques en R. STT-4230/STT-6230 : R pour scientifique*. Université Laval. URL [https://stt4230.rbind.io/communication\\_resultats/graphiques\\_r/](https://stt4230.rbind.io/communication_resultats/graphiques_r/)



**Merci !**