Tidyverse : Visualisation des données

Eric Marcon

01 February 2024

Visualisation

ggplot2

Package destiné à la création de graphiques.

Respecte la grammaire graphique par couches :

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(
     mapping = aes(<MAPPINGS>),
     stat = <STAT>,
     position = <POSITION>
  ) +
  <COORDINATE_FUNCTION> +
  <FACET_FUNCTION>
```

Les données sont obligatoirement un dataframe (un tibble est un dataframe).

Esthétique

L'esthétique désigne ce qui est représenté :

- `x` et `y` (ou `fill` pour un histogramme...) ;
- transparence, couleur, type de courbe, taille, ... : voir l'aide de chaque `geom_`.

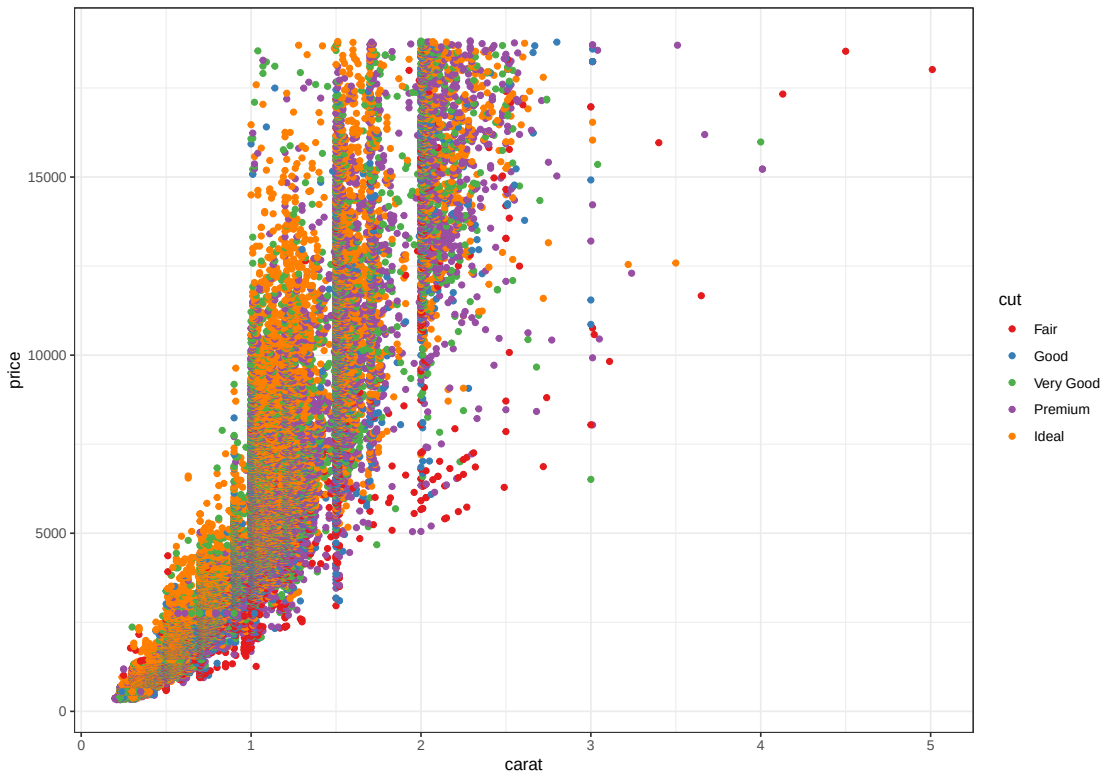
Fonction `aes()` à plusieurs niveaux :

- argument `mapping` de `ggplot()`, hérité par les couches (`geom_`) ;
- argument `mapping` de chaque couche.

Géométrie

La géométrie est définie par une fonction `geom_xxx` et une esthétique (ce qui est représenté).

```
ggplot(data = diamonds) +
  geom_point(mapping = aes(x = carat, y = price, color = cut)) +
  scale_colour_brewer(palette = "Set1")
```



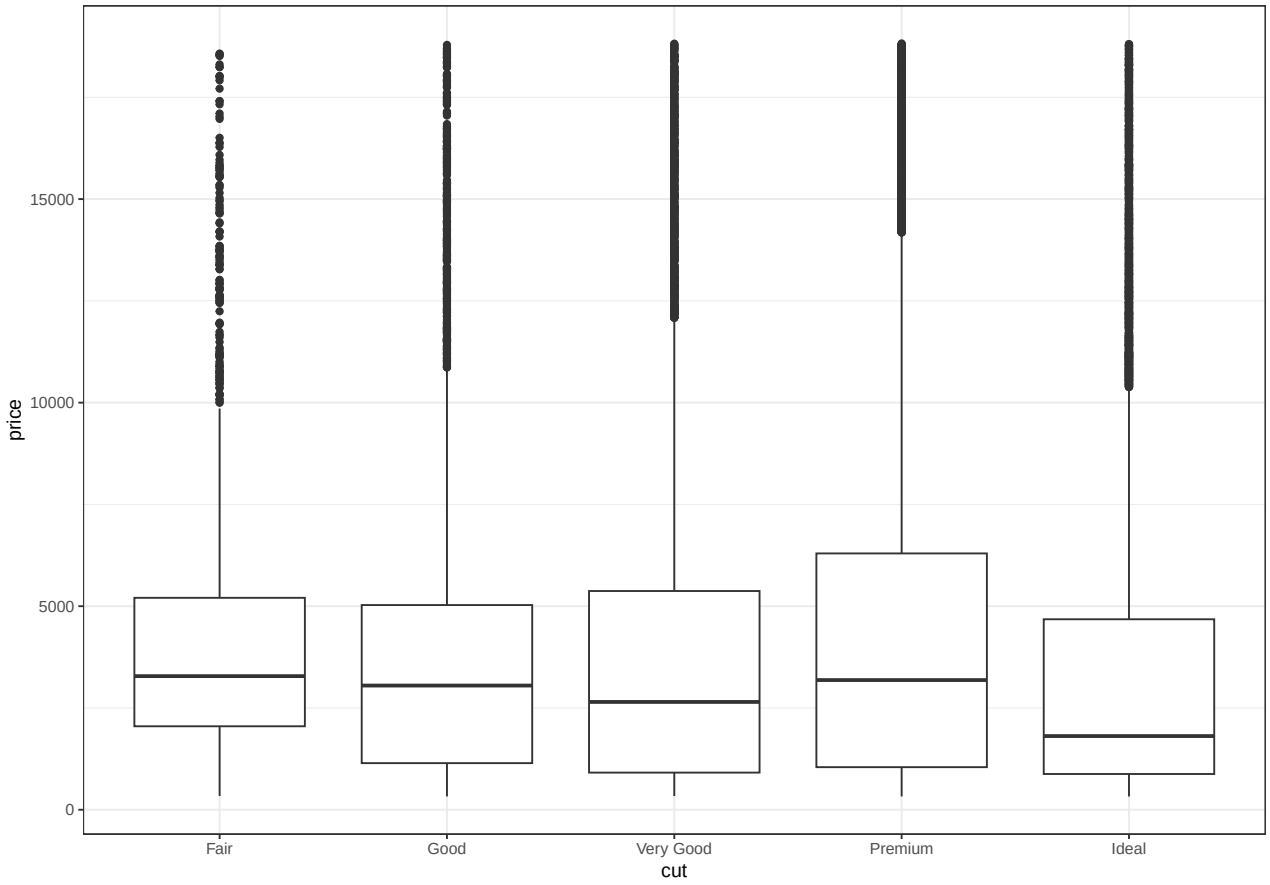
Statistiques

Chaque `geom_` va de pair avec une statistique de transformation des données :

- "identity" pour `geom_point` ;
- "boxplot" pour `geom_boxplot` ;
- 20 statistiques disponibles...

Statistiques


```
ggplot(data=diamonds) + geom_boxplot(mapping = aes(x = cut, y = price))
```



Statistiques

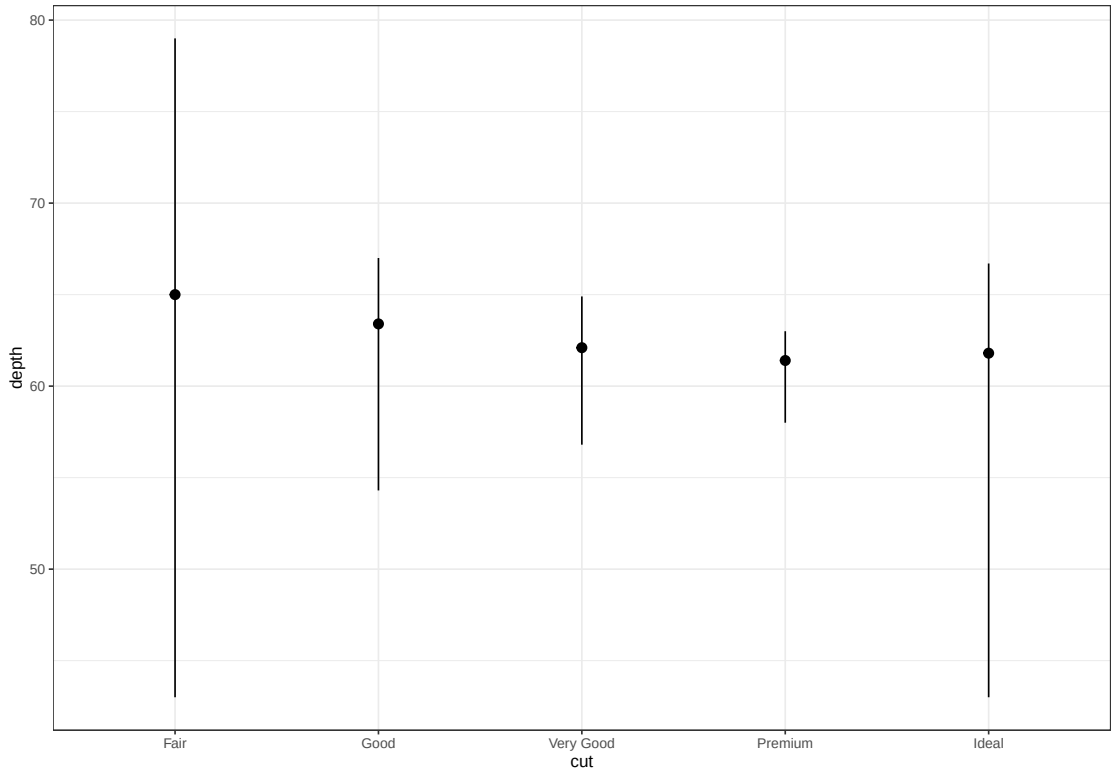
Différent de la transformation de variables (cf. *scale*) : le graphique utilise des données dérivées des données originales.

Chaque statistique a un `geom_` par défaut :

`stat_summary` est interchangeable avec `geom_pointrange`.

Statistiques

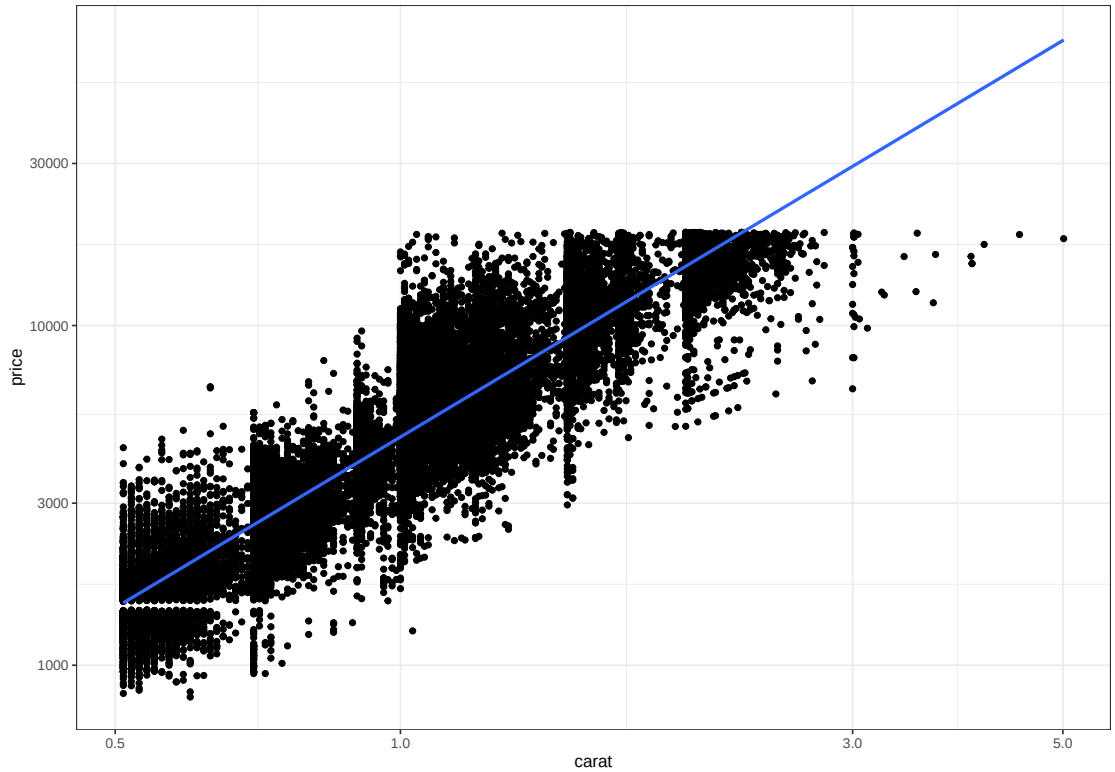

```
ggplot(data = diamonds) +
  stat_summary(
    mapping = aes(x = cut, y = depth),
    fun = median,
    fun.min = min,
    fun.max = max
  )
```



Echelle

Transformation de variable.

```
diamonds |> filter(carat>.5) |>
  ggplot(aes(x = carat, y=price)) + geom_point() +
  scale_x_log10() + scale_y_log10() + geom_smooth(method="lm")
```



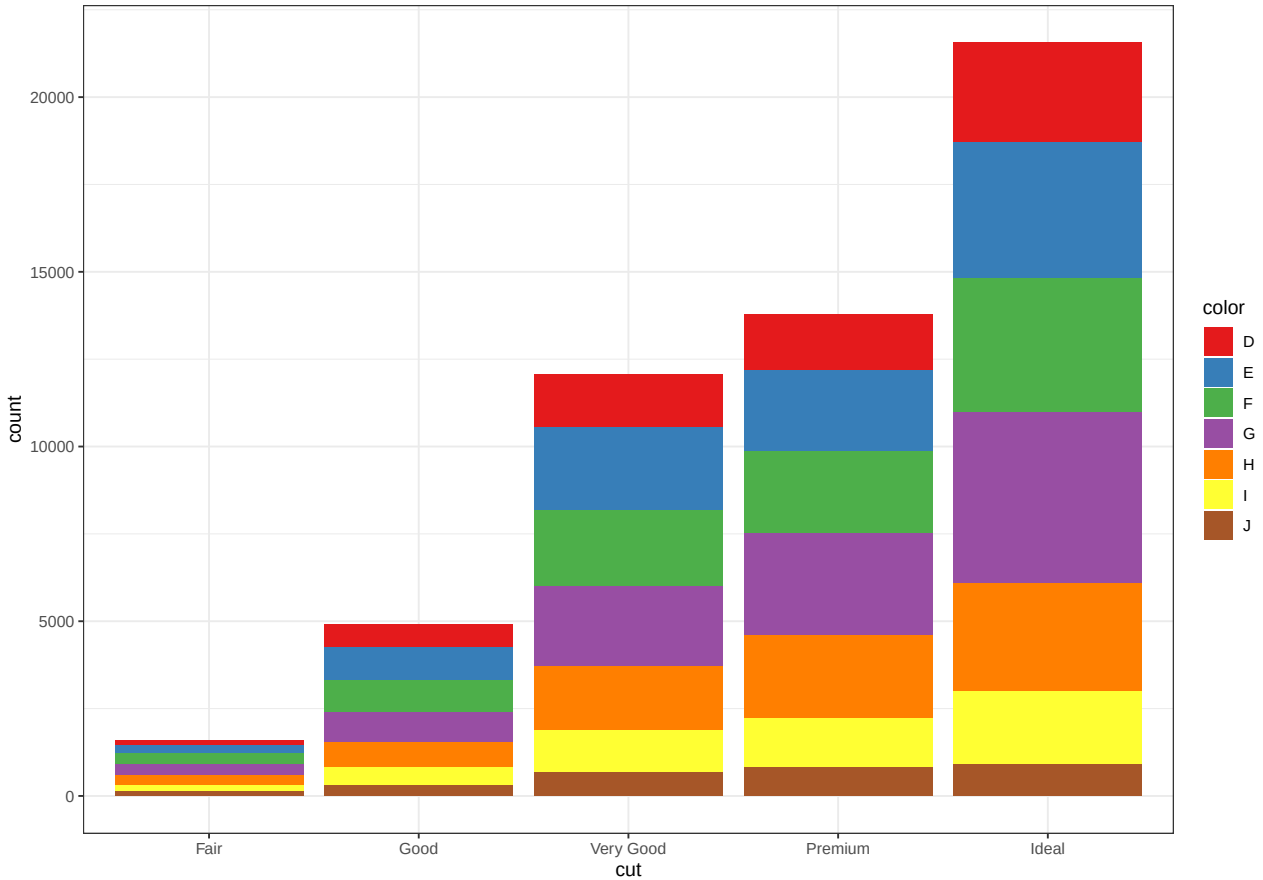
Position

La position définit l'emplacement des objets sur le graphique.

- "identity" en général ;
- "stack" empile les catégories dans un histogramme ;
- "jitter" déplace aléatoirement les points dans un `geom_point` pour éviter les superpositions.

Position


```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = color), position = "stack") +
  scale_fill_brewer(palette = "Set1")
```



Coordonnées

Système de coordonnées :

- `coord_flip()` intervertit `x` et `y` ;
- `coord_polar()` : coordonnées polaires ;
- `coord_trans()` transforme l'affichage des coordonnées (mais pas les données comme `scale_`) ;
- etc.

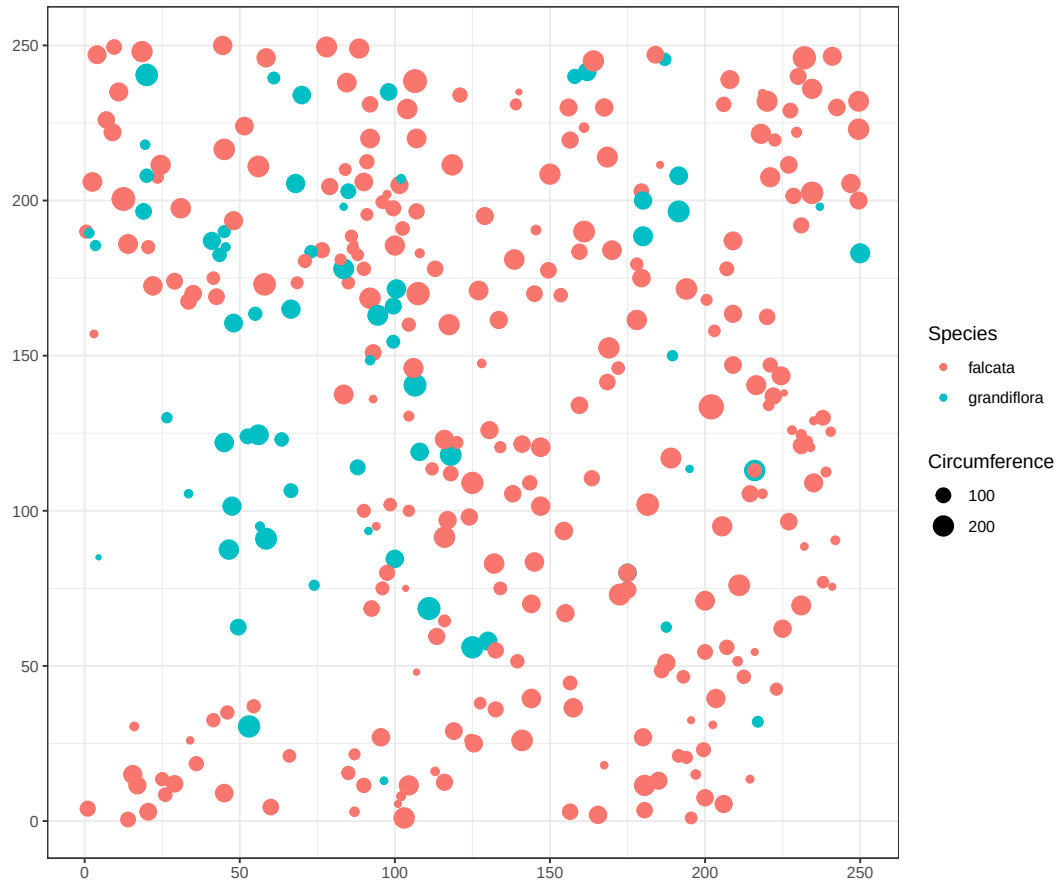
Exemple : tracer la carte des wapas de la parcelle 6 de Paracou.

Coordonnées


```
read_csv2("data/Paracou6.csv") |>
  filter(Genus == "Eperua") |>
  ggplot() +
  geom_point(
    aes(
      x = Xfield,
      y = Yfield,
      size = CircCorr,
      color = Species
    )
  ) +
  coord_fixed() +
  labs(x = NULL, y = NULL, size = "Circumference") ->
  paracou6_wapas_map
```

Coordonnées

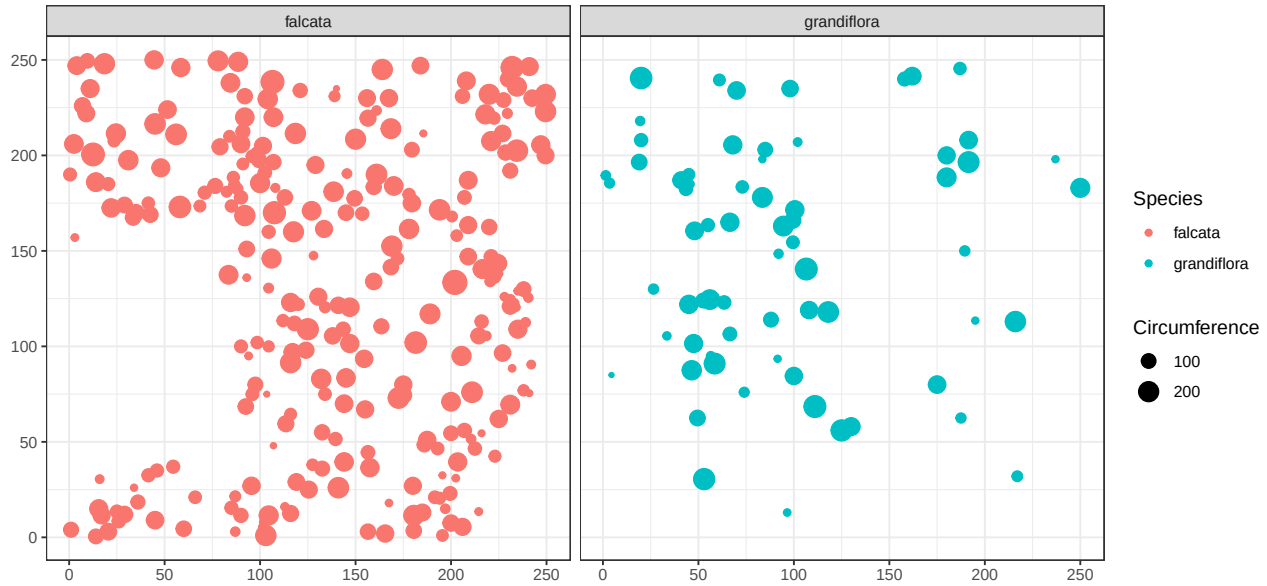

```
paracou6_wapas_map
```



Facettes

Présente plusieurs aspects du même graphique.

```
paracou6_wapas_map + facet_wrap(~ Species)
```



Remarquer : la possibilité d'affiner un graphique.

Thèmes

Les thèmes définissent l'aspect des graphiques (hors traitement des données).

Dans ce document : pas de fond grisé dans les graphiques (`theme_bw`), police 12, modifié pour que le fond soit transparent.

```
theme_set(theme_bw(base_size=12))
theme_update(
  panel.background = element_rect(fill = "transparent", colour = NA),
  plot.background = element_rect(fill = "transparent", colour = NA)
)
```

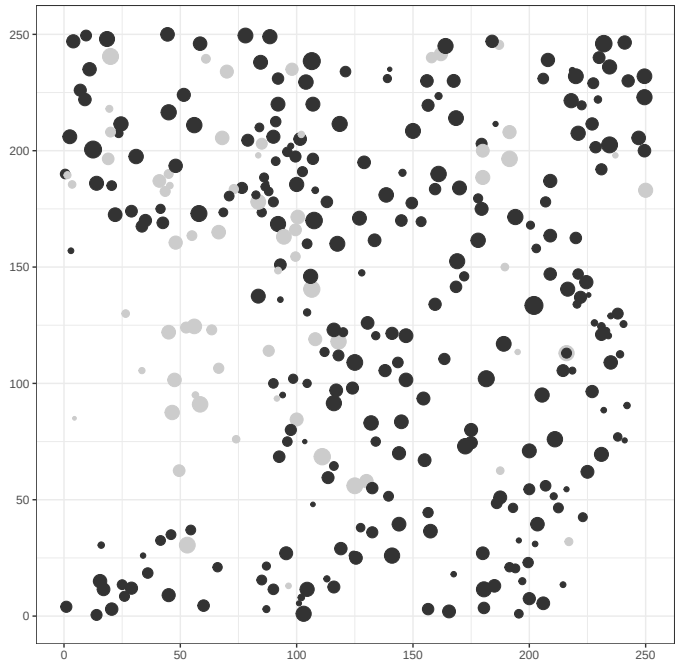
Ce sont des options globales, valides pour la session R en cours.

Styles

Possibilité d'enregistrer des paramètres de forme au-delà du thème dans une liste.

Préparation d'un style pour l'impression en noir et blanc, sans cartouches de légende.

```
style_nb <- list(scale_colour_grey(), theme(legend.position = "none"))
paracou6_wapas_map + style_nb
```

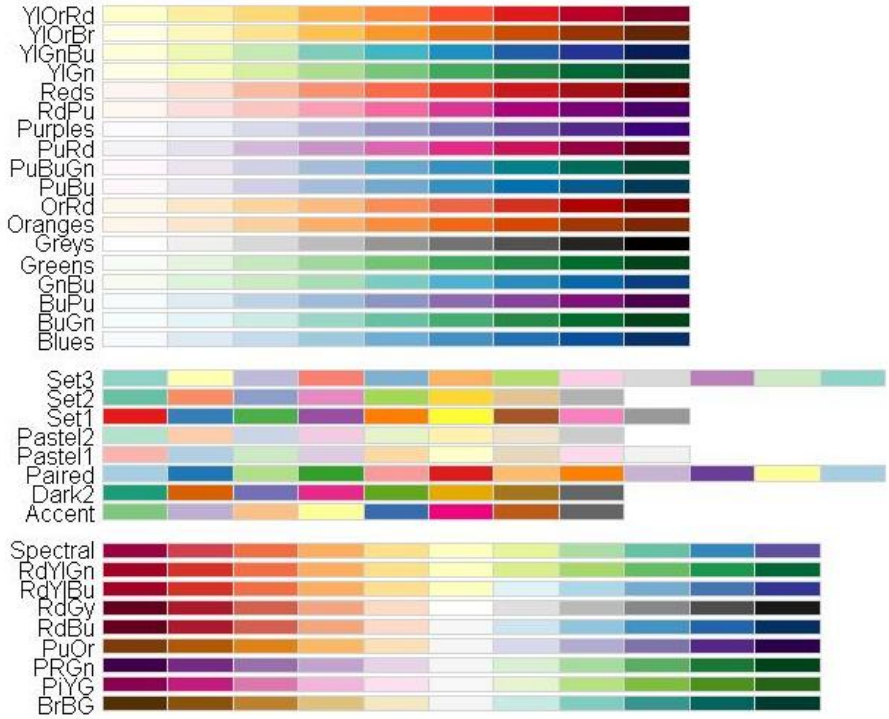


Gestion des couleurs

Les couleurs par défaut sont assez laides.

Utiliser `scale_color_xxx` et `scale_fill_xxx`

Le suffixe `_brewer` est pour utiliser des palettes de ColorBrewer



Gestion des couleurs

Le suffixe `_gradient` permet de produire un gradient de couleurs pour les valeurs continues.

Voir les autres fonctions dans l'aide du package.

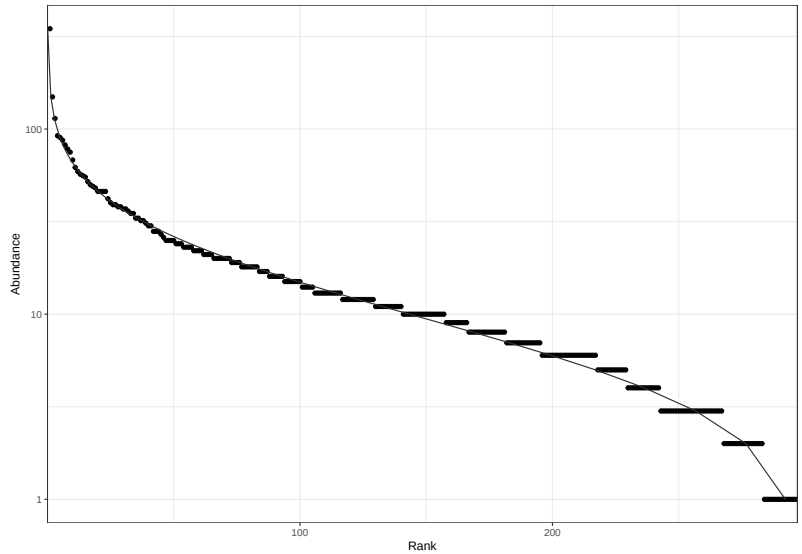
- Méthode : se créer progressivement des styles (par ex. : couleur et noir et blanc), les enregistrer et les utiliser systématiquement.

autoplot et qplot

`qplot()` mime la syntaxe de `plot()` avec *ggplot2*. Utiliser plutôt la syntaxe native.

`autoplot()` est un générique à étendre par des méthodes S3 pour faire des graphiques ggplot. Exemple:

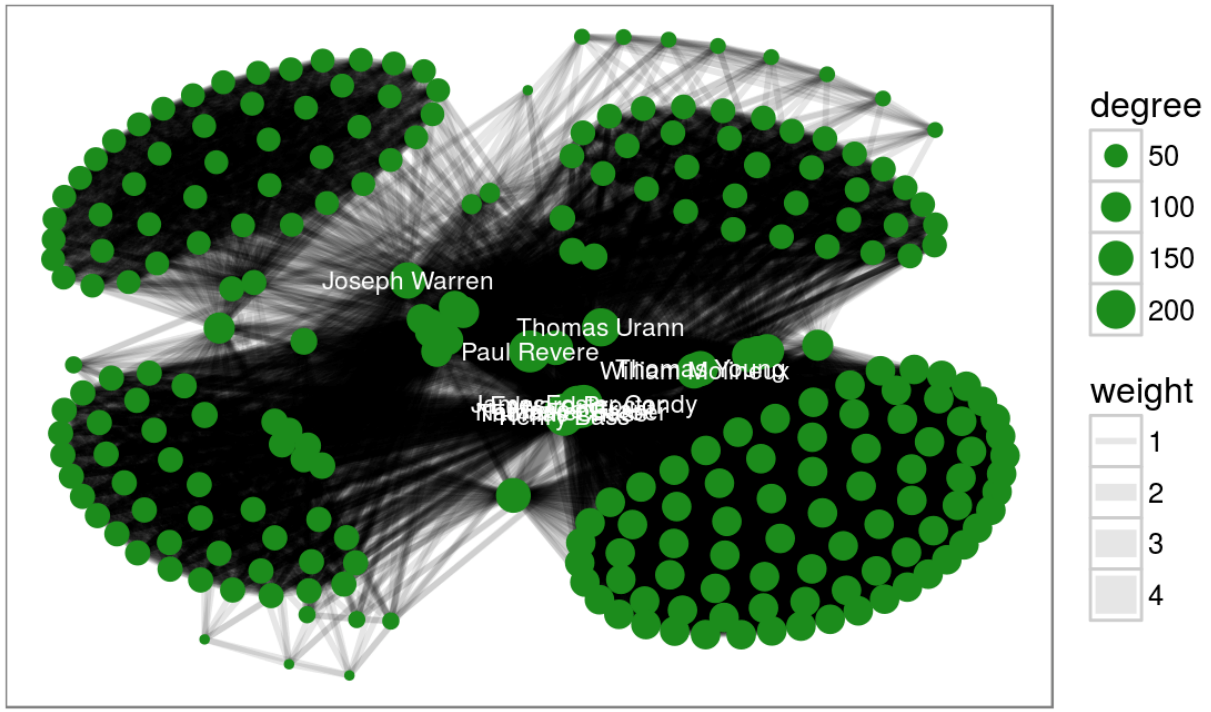
```
library("entropart")
rCommunity(1, size = 5000) |>
  autoplot(Distribution = "lnorm") + style_nb
```



Anti-sèche et extensions

Anti-sèche

De nombreux packages étendent *ggplot2* avec de nouveaux `geom_`. Exemple de *ggraph* :



AgroParisTech

Tidyverse : Visualisation des données

Eric Marcon

Visualisation