

R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation

# R: Tidyverse

Eric Marcon

20 décembre 2021

R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation

# Manifeste

# Approche complète de l'analyse de données

R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation

Données bien rangées (*tidy*)

Enchaînement des opérations (%>% de *magrittr*, + de *ggplot2*)

Programmation fonctionnelle (pas orientée objet), optimisée pour les utilisateurs (lisibilité plutôt que performance)

```
library("tidyverse")
vignette("manifesto", package = "tidyverse")
```

Ensemble de packages, appelés par *tidyverse*

# Données rectangulaires

R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation

Modèle du data frame : une ligne par observation, une colonne par attribut.

Dataframe optimisé : tibble

Documentation : vignette("tibble", package="tibble")

```
ggplot2::diamonds
```

```
## # A tibble: 53,940 x 10
##   carat cut      color clarity depth table price
##   <dbl> <ord>    <ord> <ord>   <dbl> <dbl> <int>
## 1  0.23 Ideal     E      SI2     61.5    55    326
## 2  0.21 Premium   E      SI1     59.8    61    326
## 3  0.23 Good      E      VS1     56.9    65    327
## 4  0.29 Premium   I      VS2     62.4    58    334
## 5  0.31 Good      J      SI2     63.3    58    335
## # ... with 53,935 more rows, and 3 more
## #   variables: x <dbl>, y <dbl>, z <dbl>
```

# Pipe (tuyau)

R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation

Le package *magrittr* introduit le pipe `%>%` (Ctrl + Shift + m)



# Pipe (tuyau)

R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation

## Exemple :

```
1:10 %>%
    sum
```

```
## [1] 55
```

Principe : les données résultant d'un calcul sont passées à la fonction suivante.

## Enchaînement :

```
1:10 %>%
    sqrt %>%
    sum
```

```
## [1] 22.46828
```

Code plus lisible que `sum(sqrt(1:10))`

# Autres opérateurs

R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation

Tuyau avec retour :

```
library("magrittr")
x <- c(4, 9)
x %<>%
  sqrt
x
```

```
## [1] 2 3
```

Embranchement :

```
x %T>%
  plot %>%
  sum
```



```
## [1] 5
```

# Autres opérateurs

R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation

## Exposition :

```
diamonds %$%  
      mean(price)
```

```
## [1] 3932.8
```

```
# Équivalent à  
diamonds %>%  
  pull(price) %>%  
  mean
```

```
## [1] 3932.8
```

Le tuyau de base est accessible sans charger `magrittr`

Les autres sont moins utiles

# Méthode de travail

R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation

## Bagarre (*Wrangling*) :

- Importation des données
- Rangement (*Tidy*)
- Transformation

## Visualisation

Modélisation : non traitée ici. [A lire.](#)

Communication : RMarkdown et sorties graphiques. Lire :

- [Graphics for communication](#)
- [Top 50 ggplot2 Visualizations](#)

R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation

# Bagarre

# Package *readr*

R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation

Lecture de fichiers texte variés.

Importation dans un tibble.

Référence

# Fichier csv

R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation

Fonctions `read_csv()` et `read_csv2()`

Remplacent `read.csv()` et `read.csv2()` de base

Plus rapide que les fonctions originales.

# Rangement

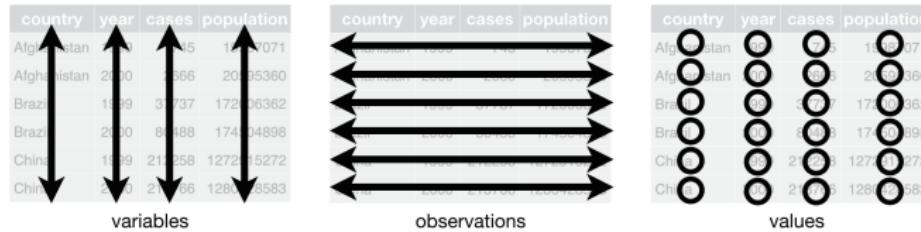
R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation



Approche habituelle en écologie (analyse multivariée par exemple)

Si les données sont mal rangées (“pas tidy”), quelques manipulations de base.

Référence

# Exemple

R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation

Données : inventaire d'une parcelle de Paracou, 4 carrés distincts.

Lire les données :

```
Paracou6 <- read_csv2("data/Paracou6.csv")
```

- Afficher Paracou6

# Rassemblement (*unite*)

R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation

Famille, genre et espèce des arbres sont dans 3 colonnes.

Créer une colonne avec le nom complet de l'espèce.

```
Paracou6 %<-%>  
  unite(col=spName, Family, Genus, Species, remove=FALSE)
```

- Afficher le résultat.

Le pipeline `%>%` (Ctrl + Shift + m) passe la donnée à la fonction suivante. Le pipeline avec retour modifie la variable de départ.

La commande classique est :

```
Paracou6 <- unite(data = Paracou6, col = spName, Family,  
  Genus, Species, remove = FALSE)
```

# Séparation (*separate*)

R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation

## Opération contraire

country	year	rate
Afghanistan	1999	<b>745</b> / 19987071
Afghanistan	2000	<b>2666</b> / 20595360
Brazil	1999	<b>37737</b> / 172006362
Brazil	2000	<b>80488</b> / 174504898
China	1999	<b>212258</b> / 1272915272
China	2000	<b>213766</b> / 1280428583

country	year	cases	population
Afghanistan	1999	<b>745</b>	19987071
Afghanistan	2000	<b>2666</b>	20595360
Brazil	1999	<b>37737</b>	172006362
Brazil	2000	<b>80488</b>	174504898
China	1999	<b>212258</b>	1272915272
China	2000	<b>213766</b>	1280428583

table3

# Rassembler des colonnes (*pivot\_longer*)

R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation

## Opération inverse de la création d'un tableau croisé

country	year	cases	country	1999	2000
Afghanistan	1999	745	Afghanistan	745	2666
Afghanistan	2000	2666	Brazil	37737	80488
Brazil	1999	37737	China	212258	213766
Brazil	2000	80488			
China	1999	212258			
China	2000	213766			

table4



# Séparer des colonnes (*pivot\_wider*)

R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation

Crée une colonne par modalité d'une variable

country	year	key	value	country	year	cases	population
Afghanistan	1999	cases	745	Afghanistan	1999	745	19987071
Afghanistan	1999	population	19987071	Afghanistan	2000	2666	20595360
Afghanistan	2000	cases	2666	Brazil	1999	37737	172006362
Afghanistan	2000	population	20595360	Brazil	2000	80488	174504898
Brazil	1999	cases	37737	China	1999	212258	1272915272
Brazil	1999	population	172006362	China	2000	213766	1280428583
Brazil	2000	cases	80488				
Brazil	2000	population	174504898				
China	1999	cases	212258				
China	1999	population	1272915272				
China	2000	cases	213766				
China	2000	population	1280428583				

table2

# Valeurs manquantes

R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation

Les valeurs manquantes explicites (valeur NA) peuvent être conservées dans les manipulations ou simplement supprimées avec l'option `na.rm=TRUE`.

`complete(var1, var2)` ajoute des enregistrements pour toutes les combinaisons de `var1` et `var2` manquantes.

[Référence](#)

# Transformation

R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation

## Outils du package *dplyr*

Idée :

- enchaîner les opérations de transformation avec les `%>%` ;
- les écrire et les tester une à une.

# Filtrer les lignes (*filter*)

R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation

## Filtrer par des conditions sur les différentes variables

```
# Nombre de lignes
```

```
Paracou6 %>%
  count %>%
  pull
```

```
## [1] 3541
```

```
# Après filtrage
```

```
Paracou6 %>%
  filter(SubPlot == 1) %>%
  count %>%
  pull
```

```
## [1] 942
```

Remarquer : `pull()` qui extrait la valeur finale du tibble de taille 1x1 produit par `count()`.

# Sélectionner les colonnes (*select*)

R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation

```
Paracou6 %>%
  select(SubPlot:Yfield, Family:Species, CircCorr) %>%
  ncol
```

```
## [1] 8
```

Remarquer : `ncol()` est une fonction de *base*, pas du tidyverse.

# Ajouter des variables calculées (*mutate*)

R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation

Des colonnes sont ajoutées au tibble

```
(Paracou6 %>%
  select(idTree, CircCorr) %>%
  mutate(Diametre = CircCorr/pi) -> Paracou6Taille)
```

```
## # A tibble: 3,541 x 3
##   idTree CircCorr Diametre
##   <dbl>    <dbl>    <dbl>
## 1 100655     44     14.0
## 2 100657     43.5    13.8
## 3 100658     53.5    17.0
## 4 100659     38.5    12.3
## 5 100660     77      24.5
## # ... with 3,536 more rows
```

Remarquer : les parenthèses pour `print()`

# Trier les lignes (*arrange*)

R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation

Afficher les plus gros arbres de la parcelle :

```
Paracou6Taille %>%
  arrange(desc(CircCorr))
```

```
## # A tibble: 3,541 x 3
##   idTree CircCorr Diametre
##   <dbl>    <dbl>     <dbl>
## 1 104455     318     101.
## 2 103939     317     101.
## 3 102249     300.     95.3
## 4 102086     290     92.3
## 5 100904     288.     91.8
## # ... with 3,536 more rows
```

# Regrouper et résumer

R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation

Quel est le diamètre moyen des arbres par famille ?

Paracou6 %>%

```
group_by(Family) %>%
  summarise(Dmean = mean(CircCorr)/pi, NbTrees = n()) %>%
  arrange(desc(Dmean))
```

```
## # A tibble: 51 x 3
##   Family           Dmean NbTrees
##   <chr>          <dbl>   <int>
## 1 Vochysiaceae    49.2     11
## 2 Combretaceae    48.4      2
## 3 Phyllanthaceae  41.9      3
## 4 Humiriaceae     38.6     20
## 5 Goupiaceae      37.4     19
## # ... with 46 more rows
```

# Lier deux tables

R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation

`bind_cols()` et `bindrows()`.

```
t1 <- tibble(col2 = c("A", "B"), col3 = 3:4)
tibble(col1 = 1:2) %>%
    bind_cols(t1)
```

```
## # A tibble: 2 x 3
##   col1 col2   col3
##   <int> <chr> <int>
## 1     1 A         3
## 2     2 B         4
```

Equivalent de `cbind()` et `rbind()`

# Joindre deux tables

R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation

inner\_join(), left\_join(), right\_join() et  
full\_join()

```
tibble(col2 = c("B", "C"), col5 = 5:6) %>%  
  inner_join(t1)
```

```
## # A tibble: 1 x 3  
##   col2    col5    col3  
##   <chr> <int> <int>  
## 1 B        5      4
```

R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation

# Visualisation

# ggplot2

R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation

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

R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation

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

R: Tidyverse

Eric Marcon

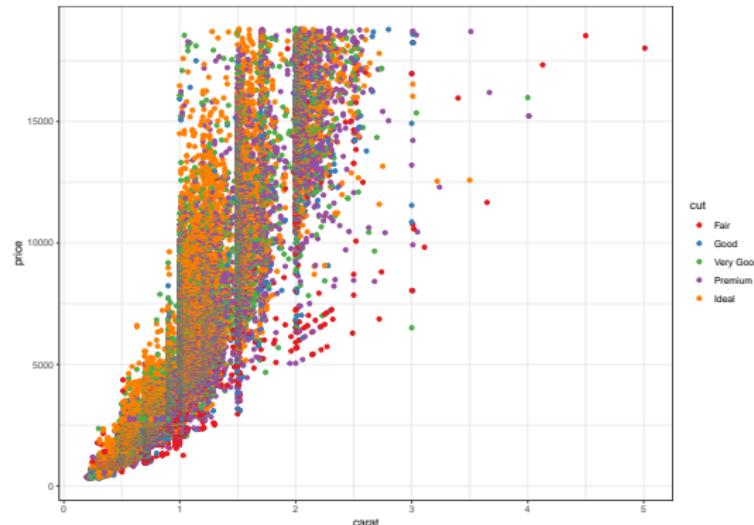
Manifeste

Bagarre

Visualisation

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

R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation

Chaque geom\_ va de pair avec une statistique de transformation des données :

- “identity” pour geom\_point
- “boxplot” pour geom\_boxplot
- 20 statistiques disponibles...

# Statistiques

R: Tidyverse

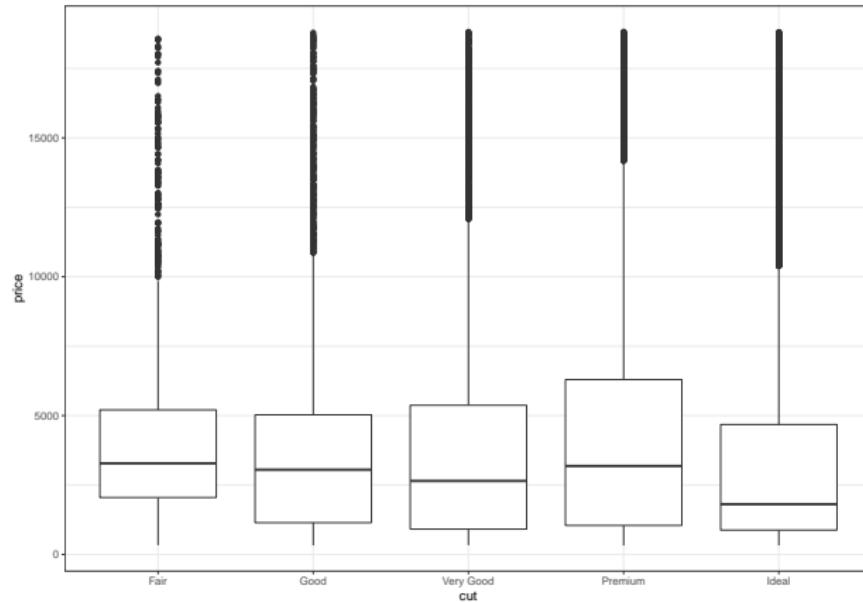
Eric Marcon

Manifeste

Bagarre

Visualisation

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



# Statistiques

R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation

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

R: Tidyverse

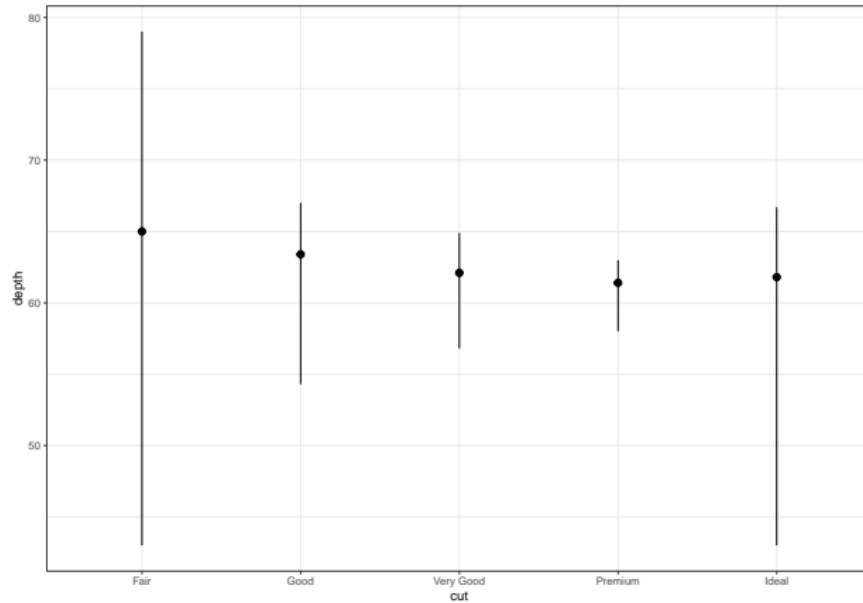
Eric Marcon

Manifeste

Bagarre

Visualisation

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



# Echelle

R: Tidyverse

Eric Marcon

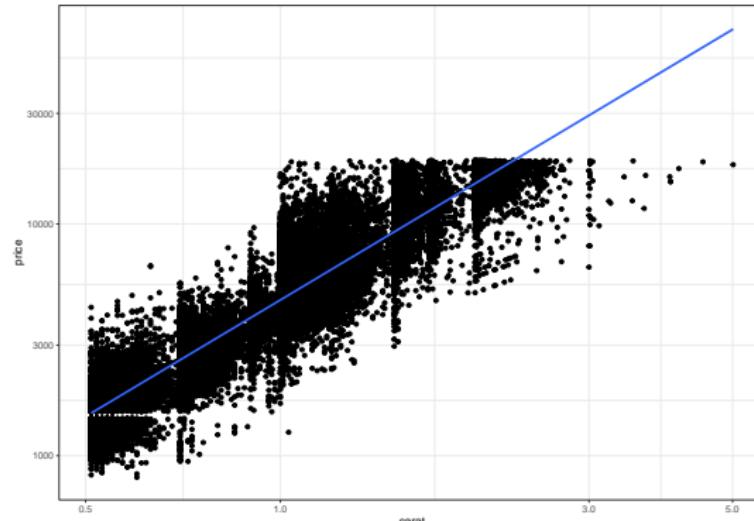
Manifeste

Bagarre

Visualisation

## Transformation de variable.

```
diamonds %>%  
  filter(carat > 0.5) %>%  
  ggplot(aes(x = carat, y = price)) + geom_point() +  
  scale_x_log10() + scale_y_log10() + geom_smooth(method = "lm")
```



# Position

R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation

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

## R: Tidyverse

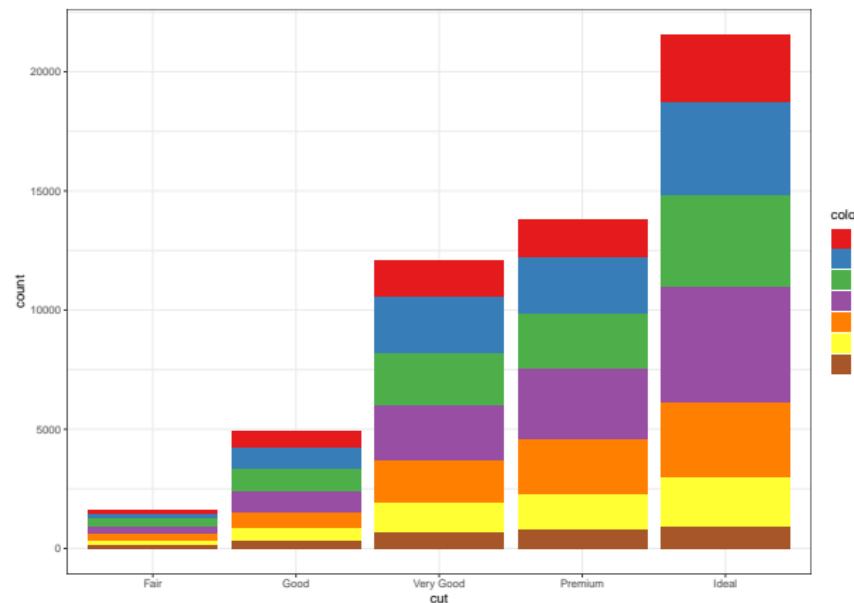
Eric Marcon

Manifeste

Bagarre

## Visualisation

```
(data = diamonds) +  
  _bar(mapping = aes(x = cut, fill = color), position="stack") +  
  _e_fill_brewer(palette = "Set1")
```



# Coordonnées

R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation

## 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.

# Coordonnées

R: Tidyverse

Eric Marcon

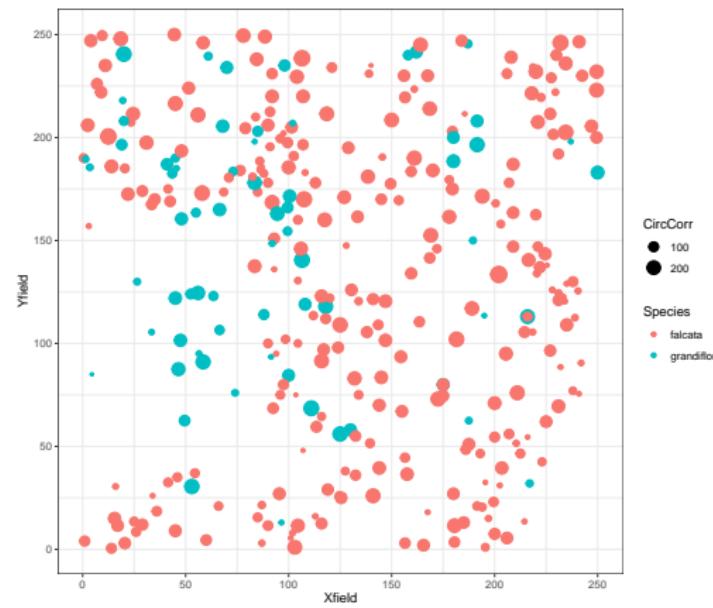
Manifeste

Bagarre

Visualisation

(Paracou6 %>%

```
filter(Genus == "Eperua") %>%
  ggplot() + geom_point(aes(x = Xfield, y = Yfield,
  size = CircCorr, color = Species)) + coord_fixed() ->
  P6Map)
```



# Facettes

R: Tidyverse

Eric Marcon

Manifeste

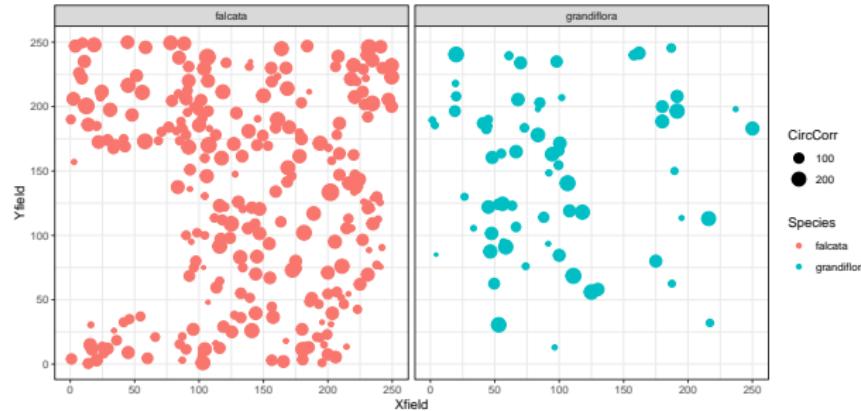
Bagarre

Visualisation

Présente plusieurs aspects du même graphique.

Remarquer : la possibilité d'affiner un graphique.

```
P6Map + facet_wrap(~Species)
```



# Thèmes

R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation

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

R: Tidyverse

Eric Marcon

Manifeste

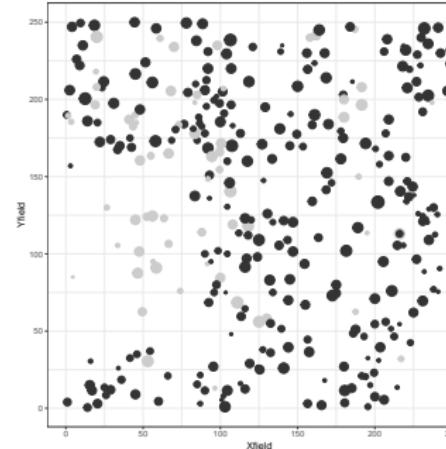
Bagarre

Visualisation

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.

```
MyStyle <- list(scale_colour_grey(), theme(legend.position = "none"))  
P6Map + MyStyle
```



# Gestion des couleurs

R: Tidyverse

Eric Marcon

Manifeste

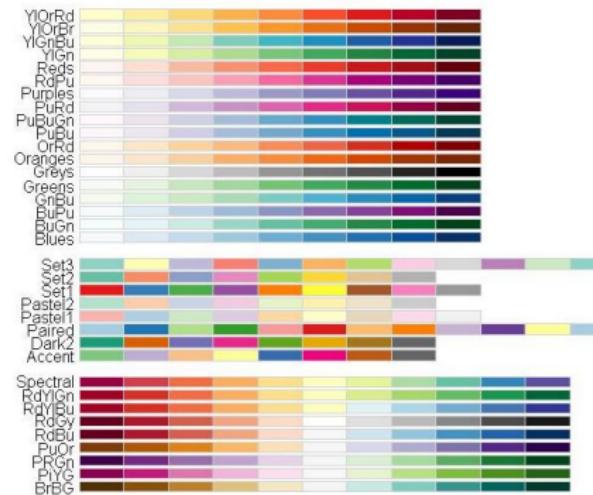
Bagarre

Visualisation

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

R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation

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

R: Tidyverse

Eric Marcon

Manifeste

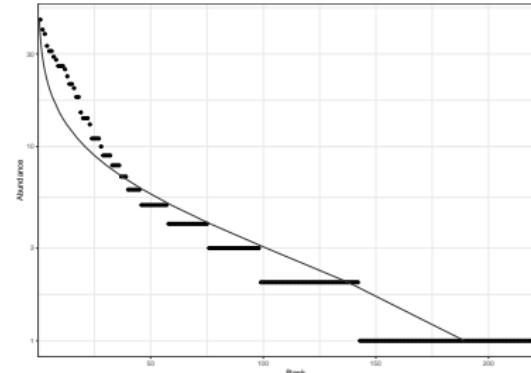
Bagarre

Visualisation

`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")
Paracou618.MC$Ns %>% as.AbdVector %>%
  autoplot(Distribution = "lnorm") + MyStyle
```



# Anti-sèche et extensions

R: Tidyverse

Eric Marcon

Manifeste

Bagarre

Visualisation

## Anti-sèche sur RStudio

De nombreux packages étendent *ggplot2* avec de nouveaux *geom\_*. Exemple de *ggraph* :

