

R: Tidyverse

Eric Marcon

02 mai 2018

Manifeste

Approche complète de l'analyse de données

Eric Marcon

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

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 0.230 Ideal  E      SI2      61.5  55.0   326
## 2 0.210 Premium E      SI1      59.8  61.0   326
## 3 0.230 Good   E      VS1      56.9  65.0   327
## 4 0.290 Premium I      VS2      62.4  58.0   334
## 5 0.310 Good   J      SI2      63.3  58.0   335
## # ... with 5.394e+04 more rows, and 3 more
## #   variables: x <dbl>, y <dbl>, z <dbl>
```

Pipe (tuyau)

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



Modèle du pipeline de la programmation système repris par la bioinformatique

Pipe (tuyau)

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

Tuyau avec retour :

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

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

Embranchement :

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



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

Autres opérateurs

Eric Marcon

Manifeste

Bagarre

Visualisation

Exposition :

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

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

Le tuyau de base est accessible sans charger magrittr

Les autres sont moins utiles

Méthode de travail

Eric Marcon

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

Bagarre

Package *readr*

1.1.1 Références

Eric Marcon

Manifeste

Bagarre

Visualisation

Lecture de fichiers texte variés.

Importation dans un tibble.

Référence

Fichier csv

Présentation

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.

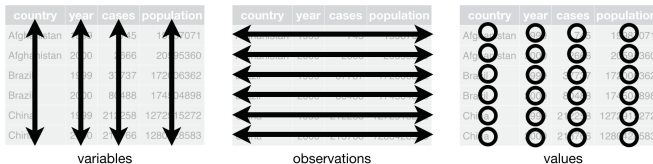
Rangement

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

Eric Marcon

Manifeste

Bagarre

Visualisation

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

Installer le package EcoFoG à partir de GitHub

```
devtools::install_github("EcoFoG/EcoFoG")
```

Extraire les données

```
library("EcoFoG")  
Paracou15 <- as.tibble(Paracou2df("Plot='15' AND CensusYear=2016"))
```

- Afficher Paracou15

Rassemblement (*unite*)

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

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

```
Paracou15 %>%
```

```
  unite(col=spName, Family, Genus, Species, remove=FALSE) -> Paracou15
```

- Afficher le résultat.

Le pipeline %>% (Ctrl + Shift + m) passe la donnée à la fonction suivante.

La commande classique est :

```
Paracou15 <- unite(data = Paracou15, col = spName,
  Family, Genus, Species, remove = FALSE)
```

Séparation (*separate*)

Eric Marcon

Manifeste

Bagarre

Visualisation

Opération contraire

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

table3

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

Rassembler des colonnes (*gather*)

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 (*spread*)

Eric Marcon

ManIFESTE

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

Tableaux

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

1. Prérequis

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*)

Filtrer par des conditions sur les différentes variables

```
# Nombre de lignes
Paracou15 %>% count %>% pull
```

```
## [1] 4151
```

```
# Après filtrage
Paracou15 %>% filter(SubPlot == 1 & CodeAlive == TRUE) %>%
  count %>% pull
```

```
## [1] 827
```

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

Sélectionner les colonnes (*select*)

Eric Marcon

Manifeste

Bagarre

Visualisation

Ne retenir que les colonnes intéressantes

```
Paracou15 %>% select(SubPlot:Yfield, Family:Species,  
  CircCorr) %>% ncol
```

```
## [1] 10
```

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

Ajouter des variables calculées (*mutate*)

Des colonnes sont ajoutées au tibble

```
(Paracou15Taille <- Paracou15 %>% select(idTree, CircCorr) %>%  
  mutate(Diametre = CircCorr/pi))
```

```
## # A tibble: 4,151 x 3  
##   idTree CircCorr Diametre  
##   <int>    <dbl>    <dbl>  
## 1 145312     65.0     20.7  
## 2 145313     78.0     24.8  
## 3 145314    172     54.6  
## 4 145315     49.0     15.6  
## 5 145316     57.5     18.3  
## # ... with 4,146 more rows
```

Remarquer : le pipe bidirectionnel, les parenthèses pour `print()`

Trier les lignes (*arrange*)

Manifeste

Eric Marcon

Manifeste

Bagarre

Visualisation

Afficher les plus gros arbres de la parcelle :

```
Paracou15Taille %>% arrange(desc(CircCorr))
```

```
## # A tibble: 4,151 x 3
##   idTree CircCorr Diametre
##   <int>    <dbl>    <dbl>
## 1 145508      332      106
## 2 145326      275      87.5
## 3 145658      273      86.9
## 4 146314      270      85.9
## 5 147958      258      82.1
## # ... with 4,146 more rows
```


Regrouper et résumer

Eric Marcon

Manifeste

Bagarre

Visualisation

Quel est le diamètre moyen des arbres par famille ?

```
Paracou15 %>% group_by(Family) %>% summarise(Dmean = mean(CircCorr)/pi,
  NbTrees = length(idTree)) %>% arrange(desc(Dmean))
```

```
## # A tibble: 56 x 3
##   Family      Dmean NbTrees
##   <chr>      <dbl>   <int>
## 1 Loganiaceae   54.3         4
## 2 Nyctaginaceae 50.1         2
## 3 Araliaceae    31.9         4
## 4 Goupiaceae    31.7        11
## 5 Vochysiaceae  30.4         6
## # ... with 51 more rows
```

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).

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_`.

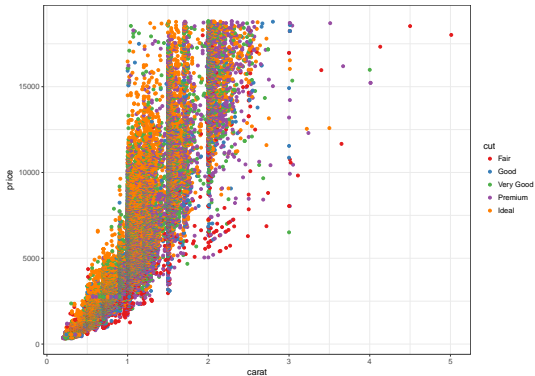
Fonction `aes()` à plusieurs niveaux :

- 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

Présentation

Eric Marcon

Manifeste

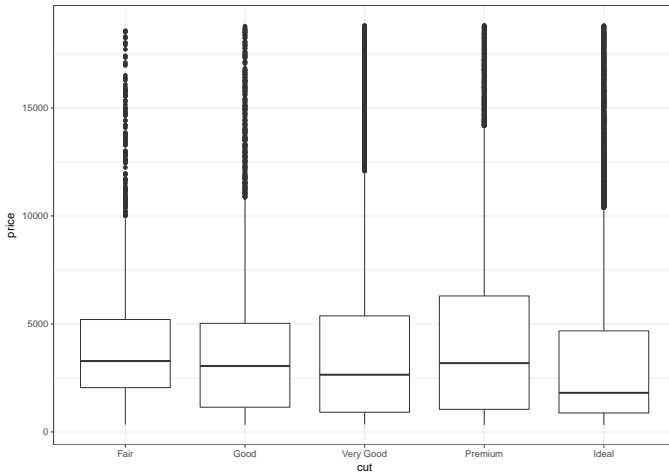
Bagarre

Visualisation

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

- “identity” pour `geom_point`
- “boxplot” pour `geom_boxplot`
- 20 statistiques disponibles...

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



Statistiques

Le Programme

Eric Marcon

Manifeste

Bagarre

Visualisation

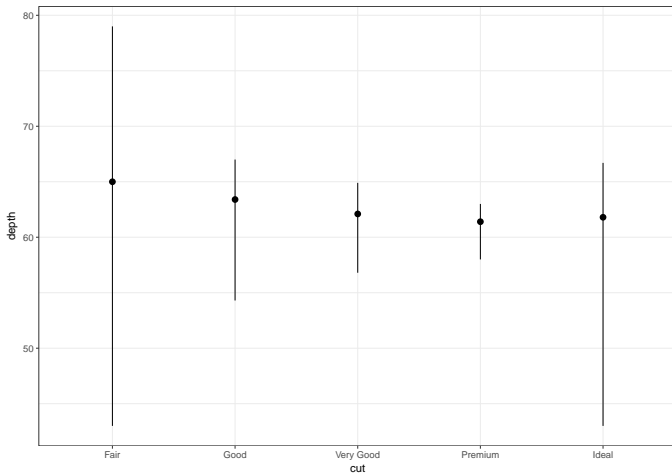
Différent de la transformation de variables (cf. *scale*) : le graphique utilise des données dérivées des données originales.

Chaque statistique a un `geom_` par défaut :

`stat_summary` est interchangeable avec `geom_pointrange`

Statistiques

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



Eric Marcon

Manifeste

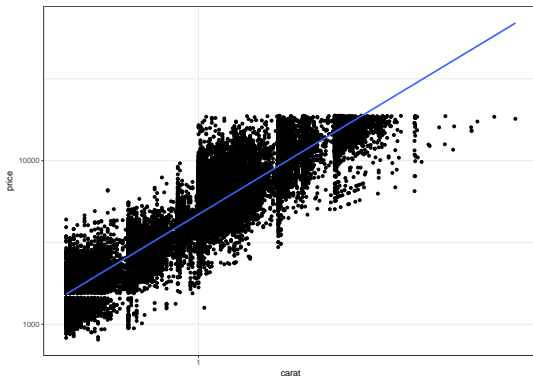
Bagarre

Visualisation

Echelle

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

Écologie

Eric Marcon

Manifeste

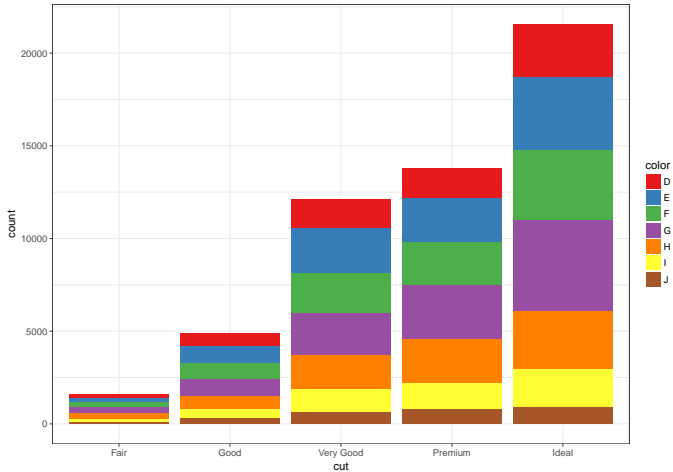
Bagarre

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.

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



Coordonnées

Eric Marcon

Manifeste

Bagarre

Visualisation

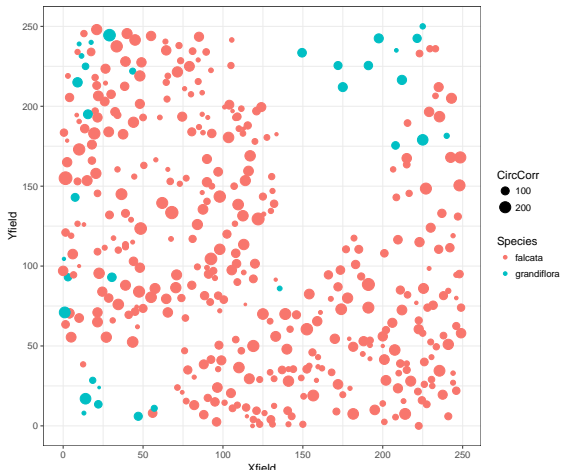
Système de coordonnées :

- `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 wacapous de la parcelle 15.

Coordonnées

```
(P15Map <- Paracou15 %>% filter(Genus == "Eperua") %>%
  ggplot() + geom_point(aes(x = Xfield, y = Yfield,
    size = CircCorr, color = Species)) + coord_fixed())
```



Eric Marcon

Manifeste

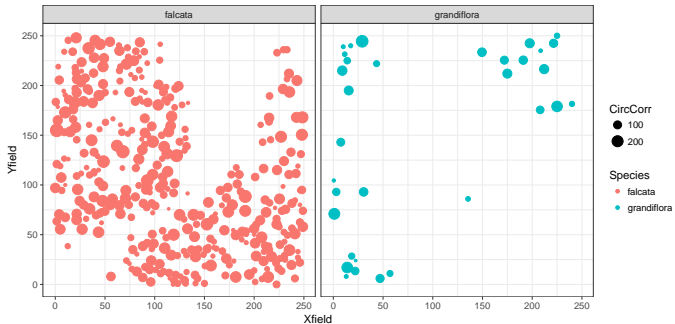
Bagarre

Visualisation

Facettes

Présente plusieurs aspects du même graphique

P15Map + `facet_wrap(~Species)`



Thèmes

Thèmes

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))
```

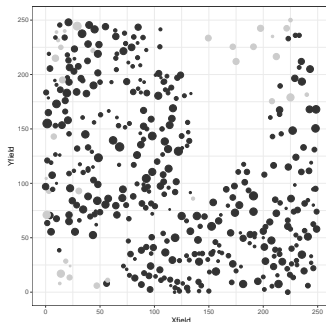
Ce sont des options globales, valides pour la session R en cours.

Styles

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"))
P15Map + MyStyle
```

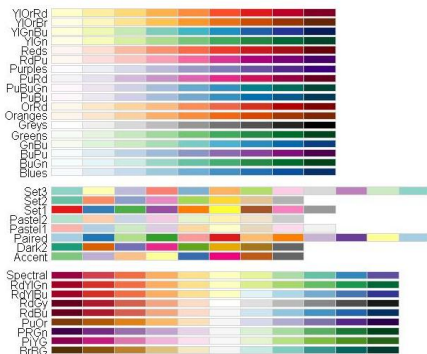


Gestion des couleurs

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

Présentation

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.

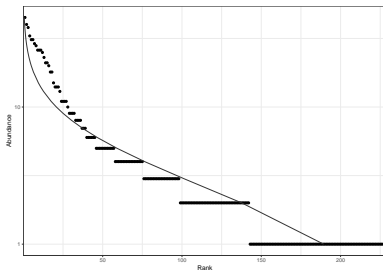
- Méthode : se créer progressivement des styles (par ex. : couleur et noir et blanc), les enregistrer et les utiliser systématiquement.

autoplot et qplot

`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

Anti-sèche sur RStudio

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

