Réaliser ses graphiques avec ggplot2

Lise Vaudor

2016-08-29

Contents

1 Introduction

ggplot2 est un des packages qui depuis quelques années font fureur parmi les utilisateurs de R.

ggplot2 est utilisé pour réaliser des graphiques. Il est bâti selon une philosophie qui lui est propre (en l'occurrence, inspirée du livre "The **G**rammar of **G**raphics" de Leland Wilkinson, d'où son nom ggplot), et qui le distingue des autres outils de production graphique sous R, notamment les fonctions graphiques "de base" comme "hist", "boxplot", etc.

L'idée générale est de **décrire** et donc **produire** un **graphique** comme un **assemblage de couches** (cela devrait parler aux SIGistes parmi vous...).

Sans partir dans des considérations théoriques (puisque rien n'est plus parlant que des exemples, et ceux-ci viendront par la suite), disons que cela permet de produire des graphiques relativement complexes et riches en information, et ce de manière assez simple et conviviale.

2 Comment ça fonctionne?

Commençons par charger le package `ggplot2`-après l'avoir, si nécessaire, installé-:

```
require(ggplot2)
```

2.1 Jeu de données illustratif

Pour vous illustrer les principes et résultats graphiques du package `ggplot2`, je vais utiliser un jeu de données issu de mon imagination fertile quoique mono-maniaque, et portant donc sur une population de chats, décrits par les variables suivantes:

- **haircolor**: la couleur du poil (catégoriel)
- **hairpattern**: le "pattern" coloré du poil (catégoriel) (si vous voulez en savoir plus sur les couleurs et patterns de poils de chat vous pouvez aller voir ici, c'est fascinant)
- **sex**: le sexe du chat (catégoriel)
- **weight**: son poids (quantitatif)
- **age**: son âge (quantitatif)
- **foodtype**: le type d'alimentation (catégoriel)

```
catdata <- read.csv("../../datasets/catdata.csv", sep=";")
```

Ce jeu de données est téléchargeable ici:

2.2 Les bases: jeu de données, aethetics, et geoms

On **crée** un graphique à l'aide de la fonction **ggplot**. On spécifie sur quel **jeu de données** le graphique va être construit, ainsi que les **aesthetics** x et y:

```
p1a=ggplot(catdata,aes(x=haircolor,y=weight))
#plot(p1a)
```

A ce stade, si l'on trace p1a, le graphique qui s'affiche n'est pas très informatif (figure 1.a)... En effet nous n'avons pas précisé quel type d'objet géométrique ("**geom**") allait être utilisé pour représenter l'information. Choisissons par exemple de représenter l'information à l'aide d'un **geom "point"**:

```
p1b=p1a+geom_point()
#plot(p1b)
```

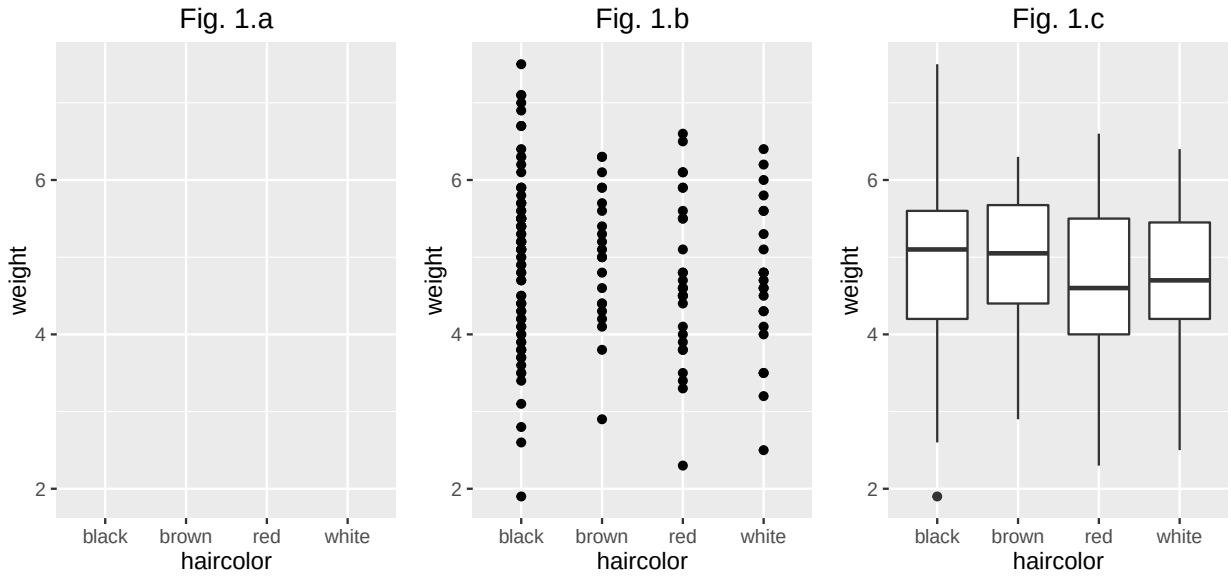
Cette fois le graphique affiche l'information qui nous intéresse (figure 1.b).

Remarquez comme on construit itérativement le graphique... Pour construire p1b, on a repris p1a, et on lui a ajouté un geom.

Si je souhaite réaliser un graphique de type boxplot plutôt que point alors il faut que je modifie le type de geom que j'utilise (figure 1.c).

```
p1c=p1a+geom_boxplot()
#plot(p2a)
```

Fig.1



Evidemment tous **les geoms peuvent être paramétrés**. Je peux par exemple modifier la couleur de mes boîtes à moustache de la façon suivante (figure 2.b):

```
p2a=p1c #on repart du dernier graphique

p2b=p1a+geom_boxplot(fill="red")
#plot(p2b)
```

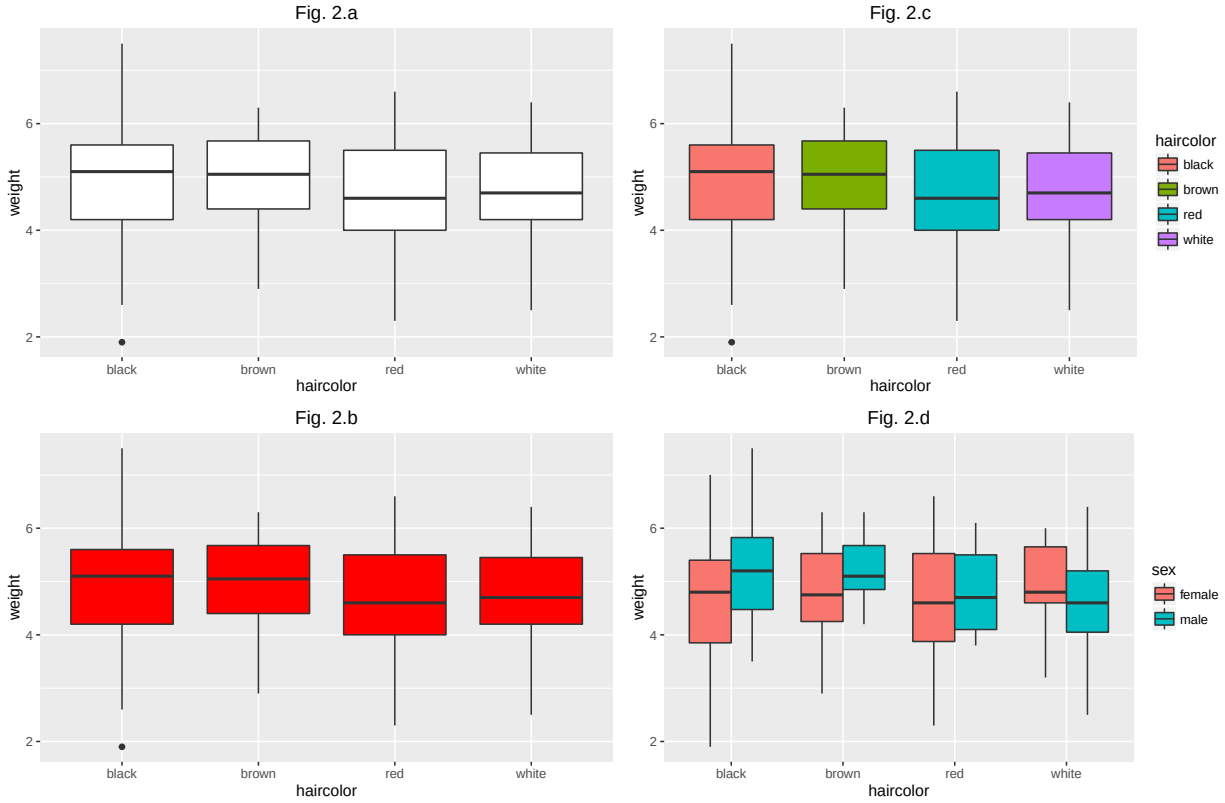
Je peux aussi non pas définir la couleur de remplissage de manière "absolue" mais en fonction des valeurs prises par une variable (en faisant du "**mapping**" -je n'ai pas trouvé de traduction adéquate...-). Cette variable est alors une autre **aesthetic** (figure 2.c). Le graphique compte alors trois aesthetics. Les deux variables qui donnent les positions x et y des objets, et la variable haircolor qui donne la couleur de remplissage des objets.

```
p2c=p1a+geom_boxplot(aes(fill=haircolor))
#plot(p2c)
```

Ici, l'information affichée par la couleur de remplissage est certes redondante avec celle donnée par l'axe x, mais on peut très bien (et même, c'est fait pour cela) utiliser en aesthetic des variables autres que celles qui définissent x et y... Cela veut dire que l'on peut très simplement **rajouter des couches d'information** (et les légendes correspondantes!) au graphique (figure 3.a).

```
p2d=p1a+geom_boxplot(aes(fill=sex))
#plot(p3a)
```

Fig.2



A ce stade, `ggplot2`devrait déjà vous sembler d'une intelligence et d'une efficacité diaboliques...

Remarque

Une des rares réserves que j'ai à l'encontre de `ggplot2` (et plus généralement, j'imagine, contre le cadre conceptuel proposé par Leland Wilkinson avec sa Grammar of Graphics) est que le terme d'**aesthetic** me semble assez contre-intuitif... J'ai en effet tendance à assimiler à "esthétique" quelque chose qui relèverait de la forme (par exemple la couleur des points ou la taille des étiquettes) plutôt que du fond... Or ici, il n'en est rien, car par exemple les variables x et , et les différents groupes considérés, qui sont des aesthetics, correspondent à des informations pour le moins capitales pour construire le graphique! De ce fait, ce terme m'a pas mal perturbée quand je découvrais le package (mais ça doit être mon côté psychorigide...). Le but de cette remarque est donc de soulever le problème une bonne fois pour toute, pour l'évacuer le plus vite possible...

3 Ajout d'information statistique

3.1 statistiques descriptives

Le package `ggplot2` permet en outre de rajouter des couches d'information "statistique" via la fonction `stat_summary`. Par exemple, on peut rajouter un geom de type point qui nous informe sur les moyennes par groupes -définis pour la variable x, c'est à dire haircolor- (figure 3.a).

```
p3a=p2d #on repart du dernier graphique

p3b=p3a+stat_summary(fun.y=mean,
                     geom="point")
```

On peut calculer une statistique sur les autres aesthetics. Ici par exemple on combien les groupes données par la variable haircolor et par la variable sex (figure 3.c):

```
p3c=p3a+stat_summary(fun.y=mean,
                     geom="point",
                     aes(fill=sex))
```

Pour ajuster la position (en x) des moyennes en fonction de sex on peut procéder comme suit (figure 3.d):

```
p3d=p3a+stat_summary(fun.y=mean,
                     geom="point",
                     aes(fill=sex),
                     position=position_dodge(width=0.75))
```

Fig. 3

Fig. 3.a

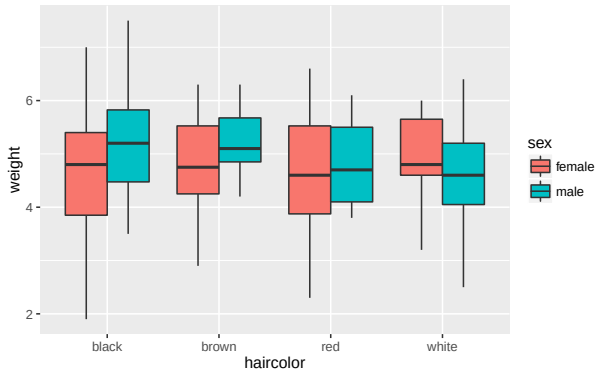


Fig. 3.c

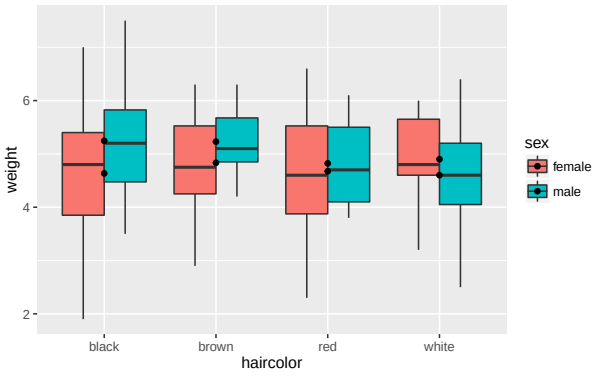


Fig. 3.b

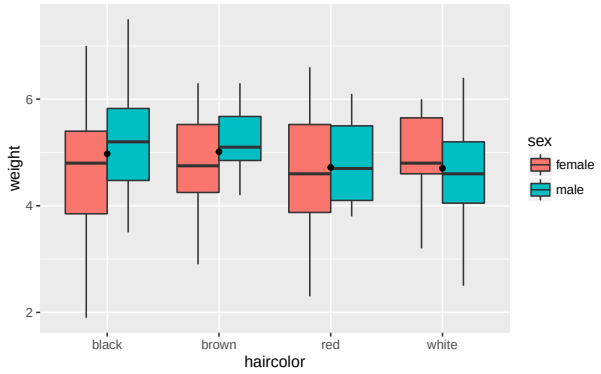
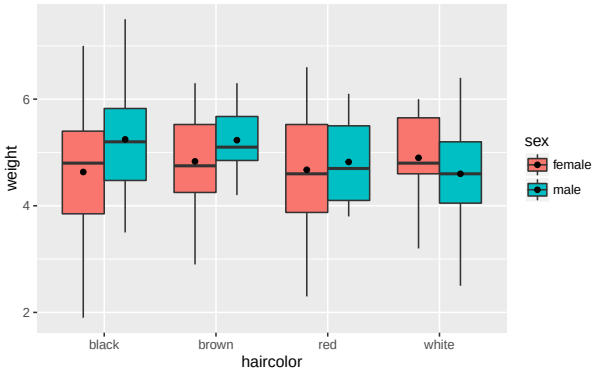


Fig. 3.d



3.2 modèles

Le package `ggplot2` offre aussi des possibilités intéressantes pour travailler sur le lien entre deux variables quantitatives (modèles de régression):

```
p4=ggplot(catdata, aes(x=age,y=weight))
p4a=p4+geom_point(aes(col=sex))
#plot(p4a)
```

On ajoute une droite de régression avec la fonction `geom_smooth` (ici par défaut, régression de type loess). Allez voir le fichier d'aide associé à la fonction `geom_smooth` pour voir l'ensemble des paramètres de cette fonction...

```
p4b=p4a+geom_smooth()
#plot(p4b)
```

On peut préciser le modèle en précisant des aesthetics supplémentaires:

```
p4c=p4a+geom_smooth(mapping=aes(col=sex))
#plot(p4c)
```

Le type de régression réalisé est également paramétrable. Ici, par exemple, on réalise une régression linéaire:

```
p4d=p4a+geom_smooth(mapping=aes(col=sex),
                    method="lm")
#plot(p4d)
```

Fig. 4

Fig. 4.a

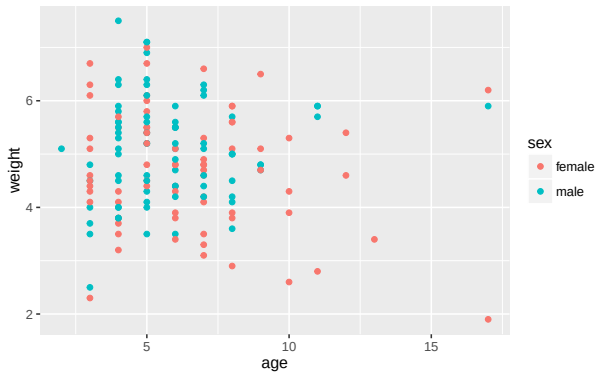


Fig. 4.b

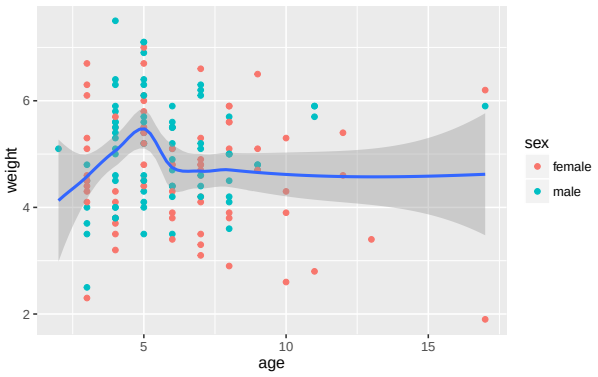


Fig. 4.c

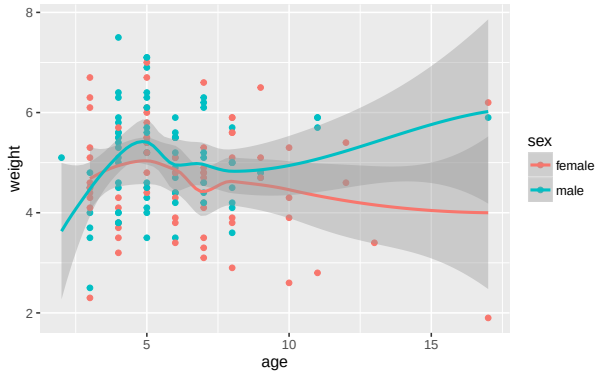
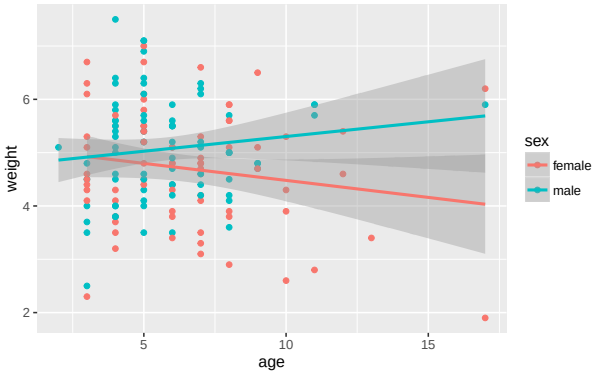


Fig. 4.d



4 Paramétrisation des titres, échelles, axes

`ggplot2` offre également des outils pour travailler facilement sur les problèmes de titres, d'échelles et d'axes. On peut ainsi facilement modifier les titres d'axe (figure 4.a). Cette figure vous montre également un nouveau type de geom (un "violon" qui vous permet d'apprécier la distribution, un peu comme un histogramme à la verticale):

```
p5=ggplot(catdata, aes(x=foodtype, y=age))
p5a=p5+geom_violin(fill="powderblue")+xlab("type de nourriture")+ylab("âge")
#plot(p5a)
```

Un des aspects les plus intéressants de `ggplot2`est la facilité avec laquelle on peut transformer les variables de position x et y. Ici par exemple en travaillant sur une échelle y log (figure 5b; dans ce cas particulier ça n'a aucun intérêt, mais c'est pour l'exemple!)

```
p5b=p5a+scale_y_log10()
#plot(p5b)
```

On peut également définir des limites d'axes (figure 5.c)

```
p5c=p5a+scale_y_continuous(limits=c(0,20))
#plot(p5c)
```

Ou définir les endroits où les barres s'affichent sur l'axe (argument "breaks"), ainsi que les étiquettes associées (argument "labels").

```
p5d=p5a+
    scale_y_continuous(limits=c(0,20),
                       breaks=c(0,5,10,15),
                       labels=c("bébé","jeune","moyen","âgé"))+
    scale_x_discrete(labels=c("dry"="croquettes",
                              "wet"="pâtée",
                              "other"="autre"))

#plot(p5d)
```

Fig. 5

Fig. 5.a

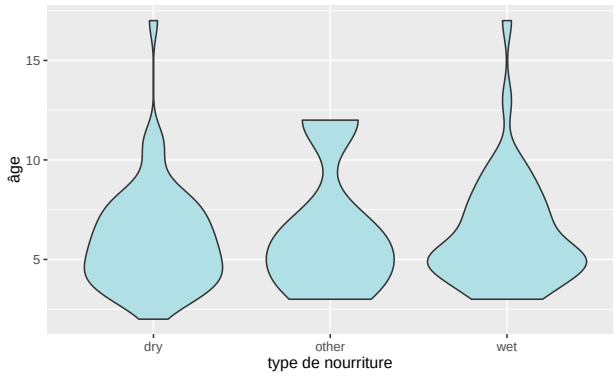


Fig. 5.b

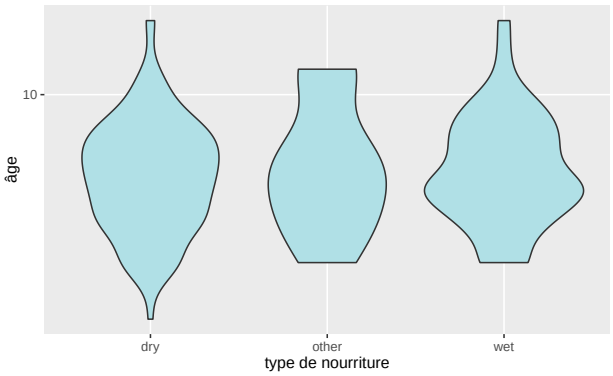


Fig. 5.c

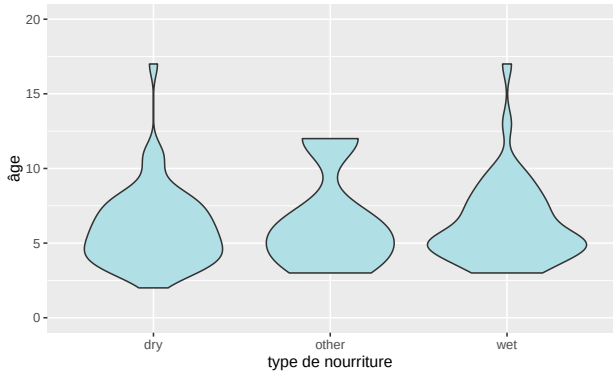
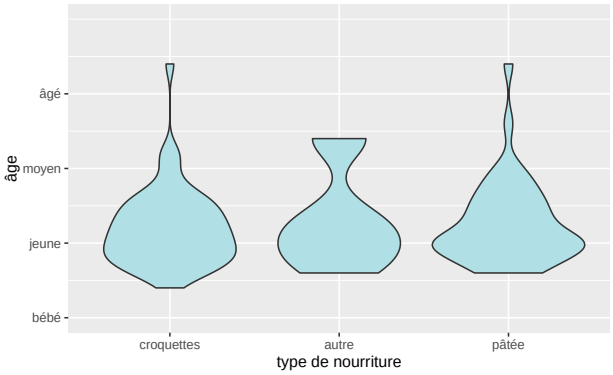


Fig. 5.d



5 Production de graphiques à facettes

Si pour le moment le concept de graphique à facettes vous semble a priori moins séduisant que celui de boule à facettes, cela ne devrait pas durer...

En effet, voyez plutôt:

```
p6a=p5d+facet_grid(hairpattern~.)
#plot(p6a)
```

En un coup de cuillère à pot, on a "multiplié" les facettes de notre graphique, ici définies verticalement par la variable hairpattern (figure 6.a).

On peut ainsi définir des facettes verticalement, horizontalement, ou les deux à la fois, par exemple ici avec des facettes verticales correspondant à la variable hairpattern et des facettes horizontales correspondant à la variable sex (figure 6.b)

```
p6b=p5d+facet_grid(hairpattern~sex)
#plot(p6b)
```

Fig. 6

Fig. 6.a

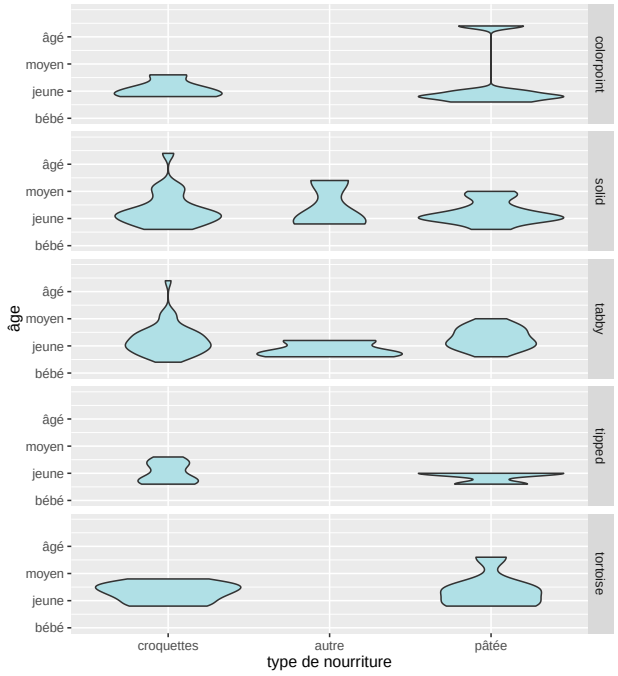
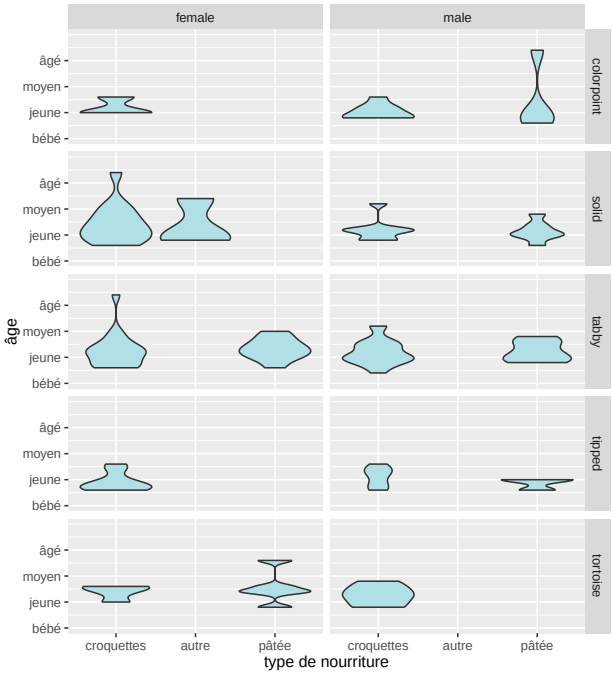


Fig. 6.b



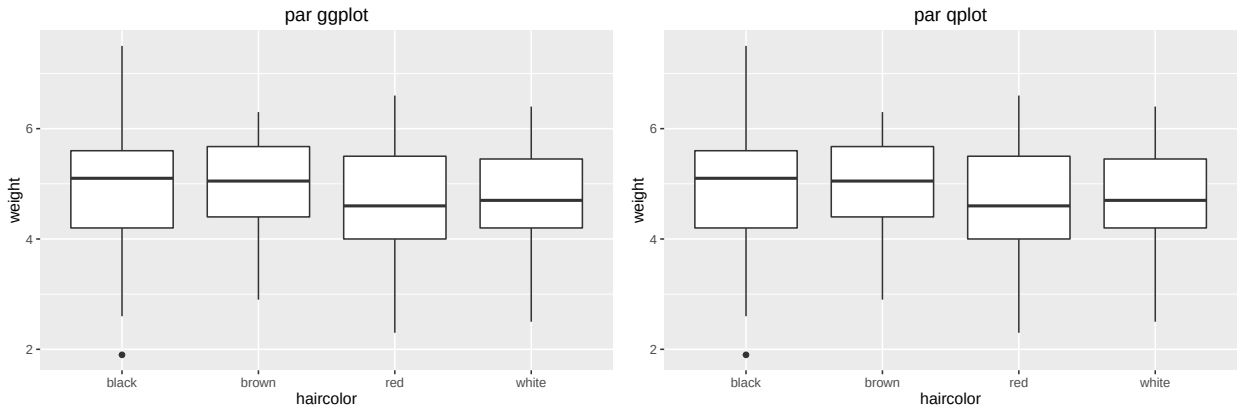
Puissant, non?

6 Raccourci de ggplot en qplot

Maintenant que la logique de `ggplot2` n'a plus de secret pour vous, je vais vous montrer comment la fonction `qplot` (pour **q**uick **plot**) fait office de raccourci pour les cas où l'on veut produire rapidement un graphique simple.

```r
# par ggplot:
pg=ggplot(catdata,aes(x=haircolor,y=weight))
pg=pg+geom_boxplot()
pg=pg+ggtitle("par ggplot")

# par qplot:
pq=qplot(x=haircolor,y=weight,
         geom="boxplot",
         main="par qplot",
         data=catdata)
```



Les deux graphiques sont rigoureusement identiques, mais le deuxième, produit via la fonction `qplot`, correspond à un code (légèrement) plus court. Au-delà de sa **concision**, la fonction `qplot` a pour avantage d'avoir une syntaxe qui ressemble un peu plus à celle de la fonction `plot` du package base de R (pour un nouvel utilisateur de ggplot2, elle demande donc un peu moins de gymnastique mentale... tout en produisant des graphiques qui ont le "look ggplot2"). Comme pour `plot` en effet, on précise une variable x et une variable y, un titre principal, etc.

En revanche, `qplot` ne remplace avantageusement `ggplot` que pour les **graphiques les plus simples** puisqu'avec `qplot` il n'est plus question de rajouter couche après couche d'information au graphique initial.

7 Un peu de lecture

J'ai ici abordé rapidement les principes et le fonctionnement du package `ggplot2` en essayant, à travers quelques exemples, de mettre en évidence les fonctionnalités que je trouve les plus utiles. Ce billet est évidemment très loin d'être exhaustif sur le sujet.

Pas de quoi désespérer car, bien évidemment, les tutoriels, livres et billets de blog à propos de `ggplot2` pullulent sur la toile.

Parmi les supports les plus utiles, il y a une "antisèche ggplot2" qui donne à voir les fonctionnalités de ce package en deux pages. Personnellement, je la garde toujours sous le coude (en fait, punaisée au mur à côté de mon ordi) pour me rafraîchir la mémoire en cas de besoin.

Pour ceux qui seraient encore dubitatifs quant à l'utilité de ggplot2, il existe des comparatifs qui discutent des situations où les fonctionnalités graphiques de base de R sont suffisantes, ou au contraire des situations où l'utilisation de `ggplot2` est préférable.

Si vous cherchez d'autres exemples de graphiques réalisés avec `ggplot2` (et bien entendu les lignes de commande qui les ont générés!), vous pouvez aller voir d'autres billets de blog comme celui-ci (en français) ou celui-là (plus détaillé, en anglais).

Enfin pour les gens qui ne jurent que par les livres, il y a celui-ci (dont on peut dire qu'il s'agit de la bible en la matière puisqu'il a été écrit par Hadley Wickham lui-même, à savoir le développeur du package).

8 Exercices

Considérons maintenant un autre jeu de données. Il s'agit du jeu de données "diamonds", qui est un des jeux de données d'exemple chargés avec `ggplot2`:

```
data(diamonds)
colnames(diamonds)
```

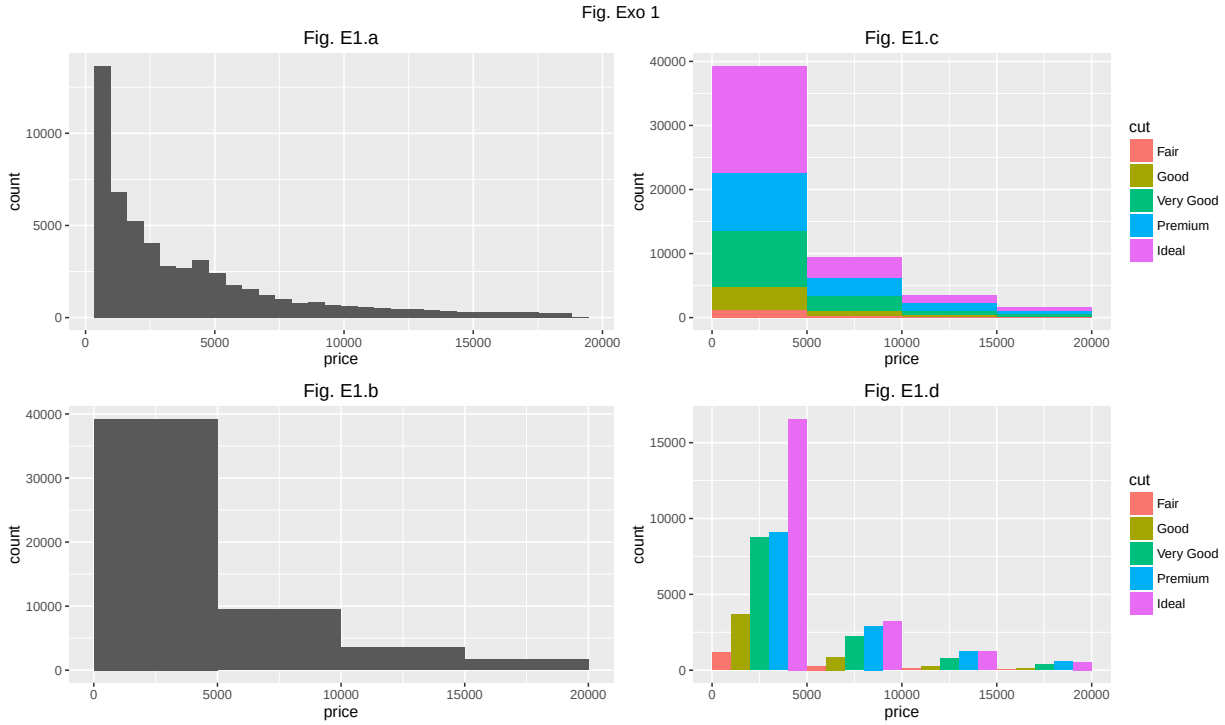
```
##  [1] "carat"   "cut"     "color"   "clarity" "depth"   "table"   "price"
##  [8] "x"       "y"       "z"
```

Pour plus d'info sur les variables de ce jeu de données, taper `help(diamonds)`

Pour faire les exercices ci-dessous, aidez-vous de l'antisèche ggplot2 et des fichiers d'aide associés aux fonctions!

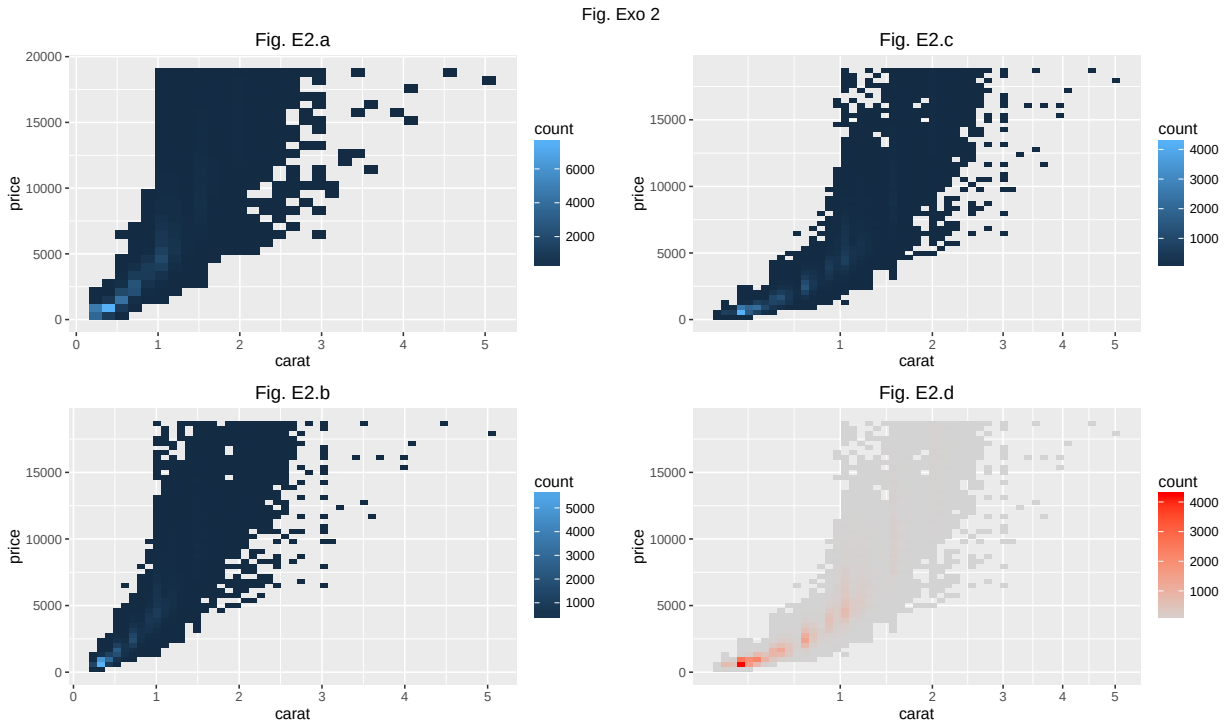
8.1 Exercice 1

Reproduisez les quatre graphiques ci-dessous (E1.a devrait vous aider à construire E1.b, qui devrait vous aider à construire E1.c, etc.):



8.2 Exercice 2

Reproduisez les quatre graphiques ci-dessous (E2.a devrait vous aider à construire E2.b, qui devrait vous aider à construire E2.c, etc.):



8.3 Exercice 3

Reproduisez les quatre graphiques ci-dessous (E3.a devrait vous aider à construire E3.b, qui devrait vous aider à construire E3.c, etc.):