Capsule 12. Visualiser des données - Partie 2

Simon LaRue

02/19/2023

Introduction

Dans cette deuxième capsule sur la visualisation des données, nous regarderons les options qu'offre la librairie `ggplot2` pour apporter des modifications à nos graphiques.

Base de données

Afin de réaliser les exemples de cette capsule, nous importerons le jeu de données apacheApsVar.csv qui contient les variables utilisées pour calculer l' « Acute Physiology Score », un indicateur de la sévérité d'une maladie à l'admission aux soins intensifs aux États-Unis durant la période de 2014-2015. Les fonctions de la librairie `ggplot2` dans `R` seront utilisées pour réaliser nos figures.

```
# Imporation de la librairie ggplot2 pour les graphique.
library(ggplot2)

# Importation de la librairie dplyr pour le traitement des données.
library(dplyr)

# Importation des données
apache_variables <- read.csv("apacheApsVar.csv", na.strings = "-1")

# Formatage des variables catégoriques
apache_variables <- mutate_at(apache_variables,
                              c("intubated","vent","dialysis","eyes","motor","verbal","meds"),
                              factor)
```

Modification d'un graphique

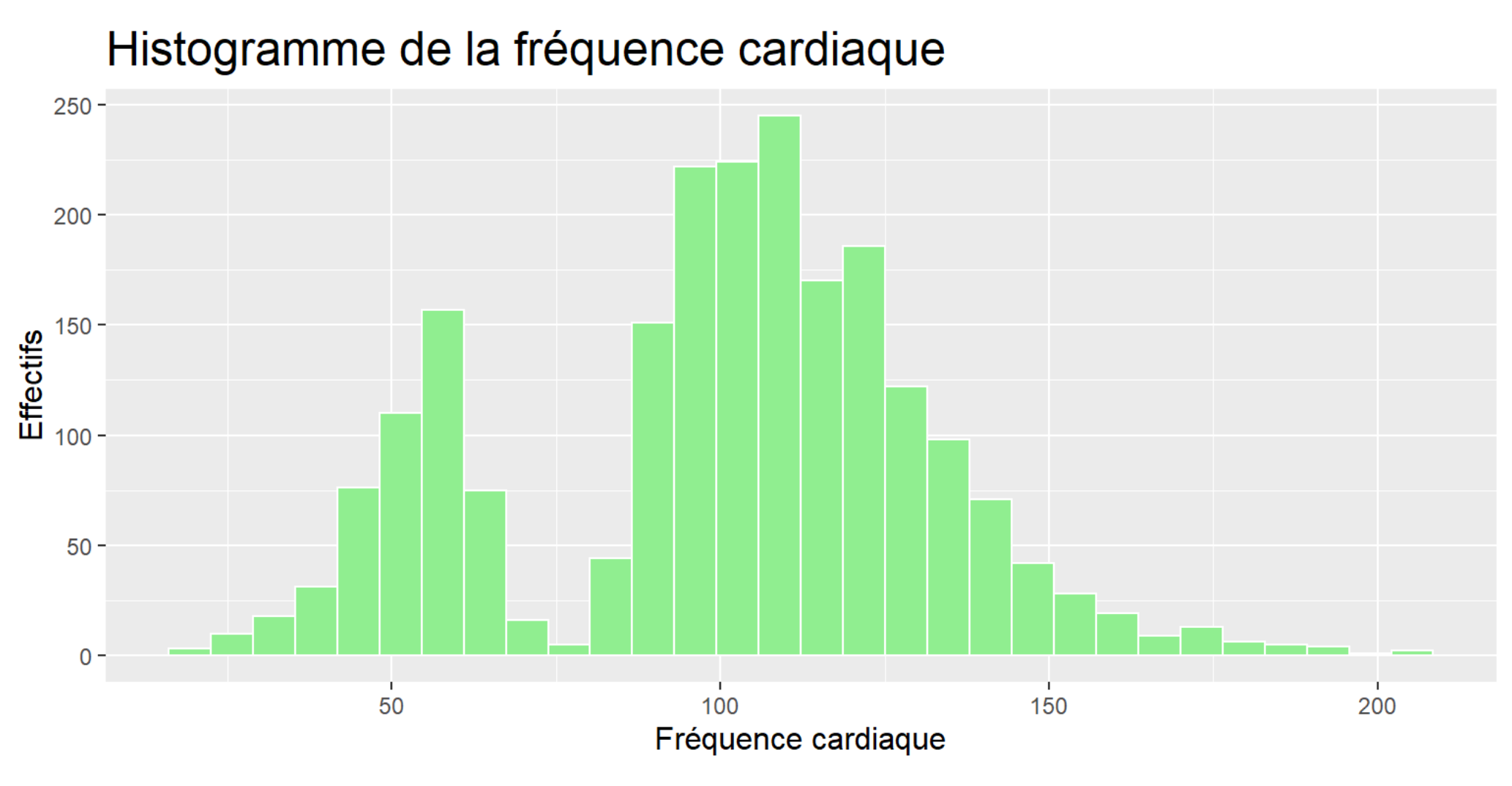
Les graphiques bruts ont une présentation de base qui n'est pas toujours suffisante pour une représentation adéquate de nos données. Nous allons généralement personnaliser nos graphiques afin de les adapter à nos questions de recherche et d'en simplifier la présentation. La librairie `ggplot2` comporte plusieurs fonctions qui permettent de modifier l'apparence de nos figures. Notamment, la fonction `theme()` qui peut être utilisée pour changer l'apparence des titres, de la légende, des lignes, des textes et bien plus.

Titres du graphique

Une première étape serait de donner des titres aux axes qui composent notre graphique. Les commandes `xlab()` et `ylab()` permettent de changer les noms de l'abscisse et de l'ordonnée respectivement. Aussi, nous pouvons donner un titre à notre graphique avec la fonction `ggtitle()`. La fonction `labs()` permet aussi de spécifier les titres des axes et du graphique dans la même commande tel qu'illustré ci-dessous.

```
ggplot(data = apache_variables, mapping = aes(x = heartrate)) +
  geom_histogram(fill = "lightgreen", color = "white") +
  xlab("Fréquence cardiaque") +
  ylab("Effectifs") +
  ggtitle("Histogramme de la fréquence cardiaque") +
  theme(plot.title = element_text(size = 18), # changer la taille du texte
        axis.title = element_text(size = 12))
```

```
ggplot(data = apache_variables, mapping = aes(x = heartrate)) +
  geom_histogram(fill = "lightgreen", color = "white") +
  labs(x = "Fréquence cardiaque",
       y = "Effectifs",
       title = "Histogramme de la fréquence cardiaque") +
  theme(plot.title = element_text(size = 18), # changer la taille du texte
        axis.title = element_text(size = 12))
```



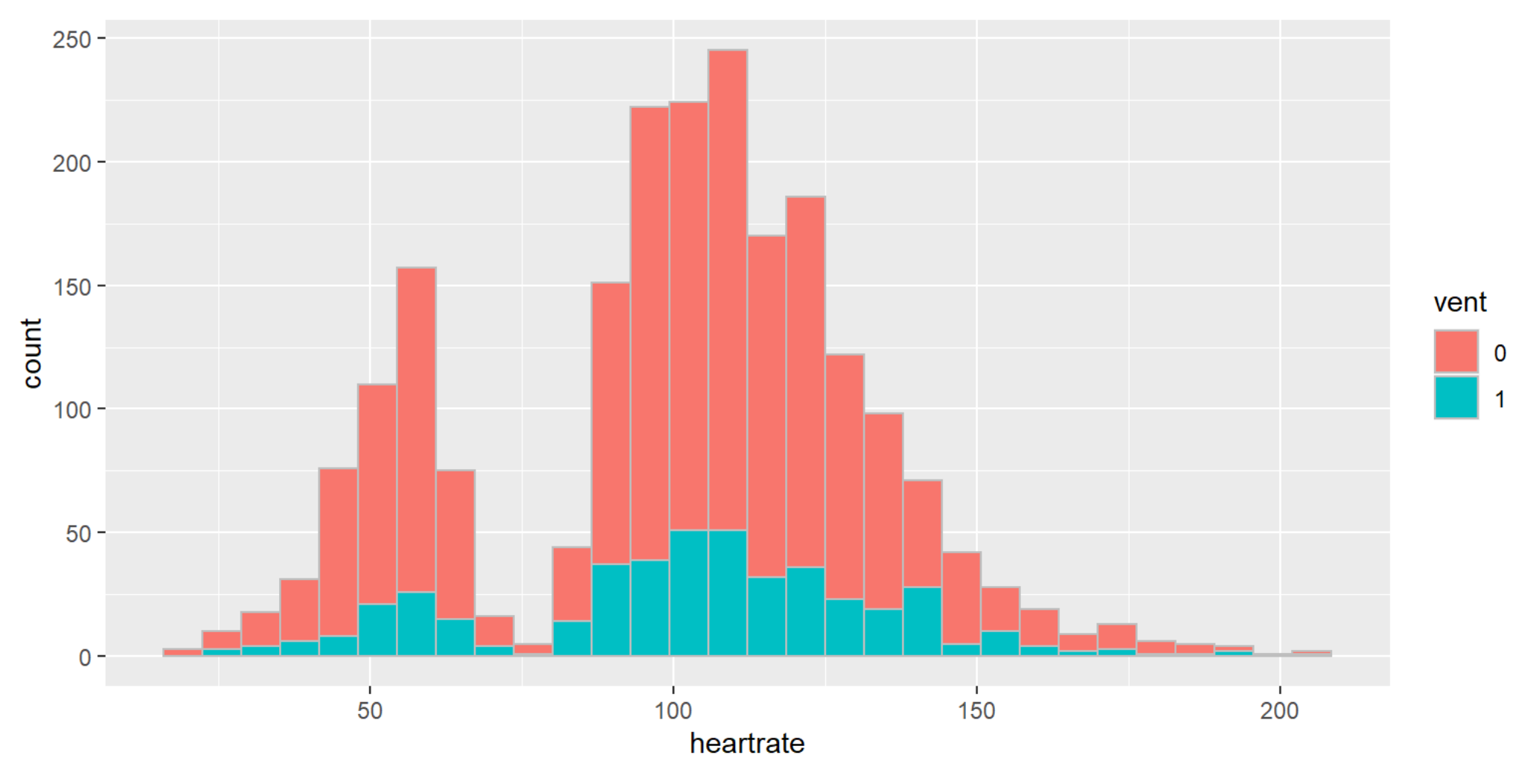
Couleur et forme

Par la suite, nous pouvons personnaliser notre graphique en utilisant de la couleur. La plupart du temps, la couleur sert à distinguer visuellement des groupes de sujet ou des catégories d'une variable dans les données. Il est aussi possible d'utiliser la couleur pour rendre notre figure plus vivante, par exemple en accordant les couleurs avec le thème d'une présentation. Cependant, il faut faire attention de ne pas utiliser la couleur de sorte à confondre le lecteur.

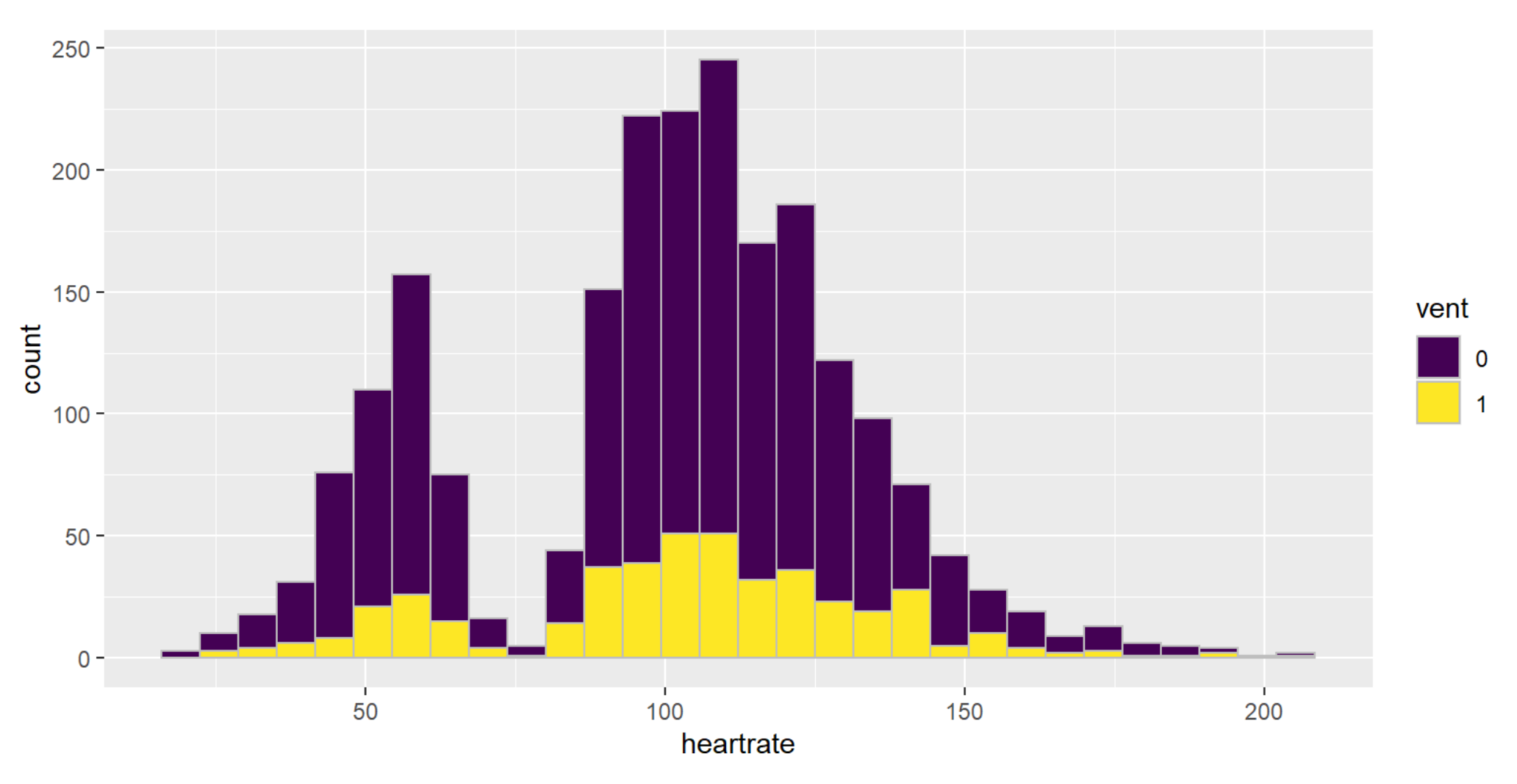
Deux arguments permettent de changer la couleur d'un graphique dans les composantes géométriques `geom_…()`. L'argument `fill=()` permet de fixer la couleur à l'intérieur des formes alors que l'argument `color=()` change les points et les contours des objets. Dans l'aesthétique `aes()`, ces arguments vont être plutôt utilisés afin de représenter une nouvelle variable catégorique. Il est aussi possible de modifier ces arguments en ajoutant une commande `scale_fill_…()` et `scale_color_…()`. Par exemple, `scale_fill_viridis_d()` permet de changer la couleur de l'argument `fill` avec une palette de couleur de `R`.

Le principe est le même pour les formes des points et des lignes avec les arguments `shape=` et `linetype=` respectivement. Pour modifier ces arguments, nous utiliserons les commandes `scale_shape_…()` et `scale_linetype_…()`.

```
# histogramme avec de la couleur
p <- ggplot(data = apache_variables, mapping = aes(x = heartrate, fill = vent)) +
  geom_histogram(color = "grey")
p
```



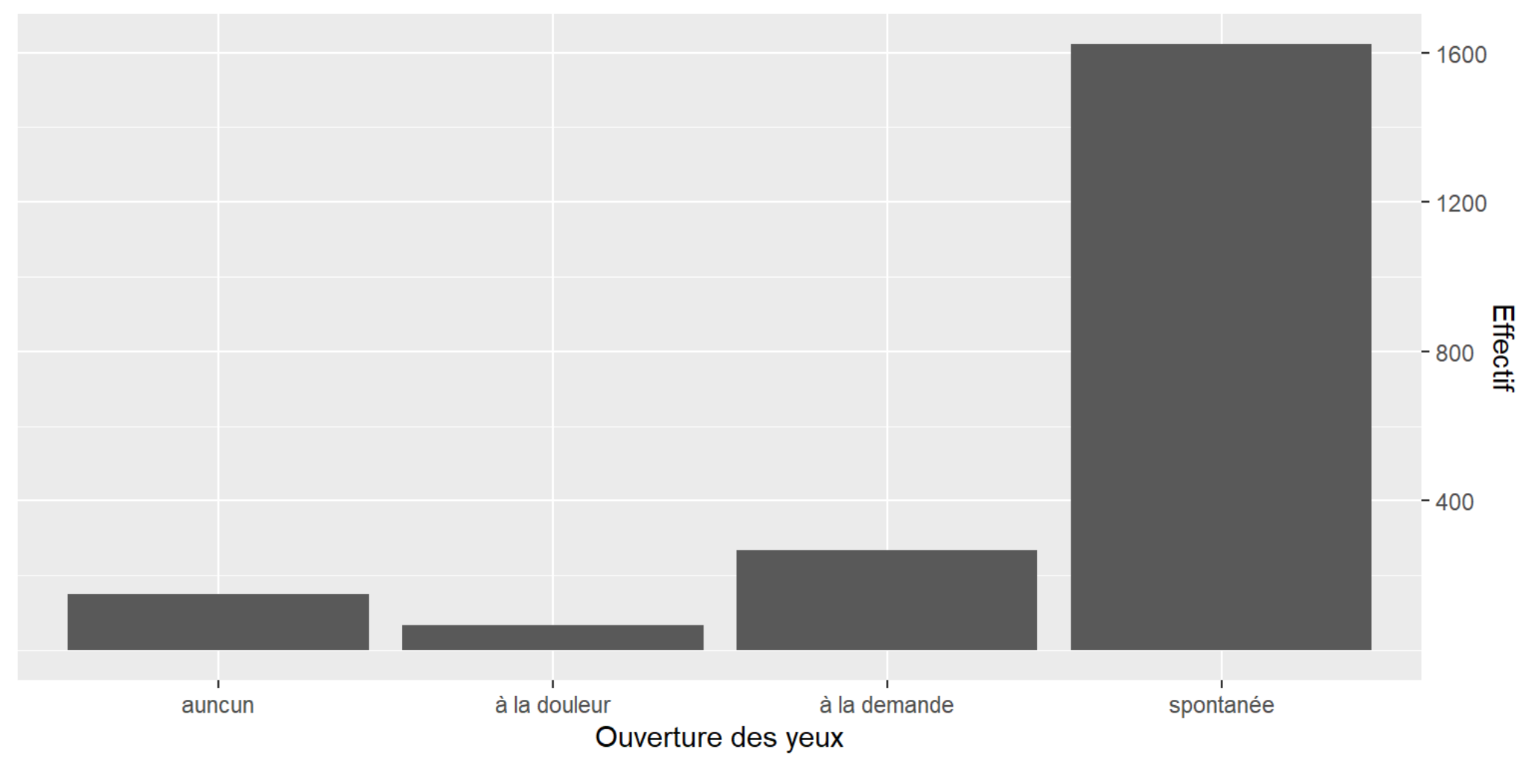
```
# changement de la palette de couleur
p + scale_fill_viridis_d()
```



Échelle des axes

Pour bien illustrer les quantités que mesurent nos variables, il nous est parfois nécessaire de personnaliser l'échelle des axes. Les fonctions `scale_x_discrete()` et `scale_x_continuous()` permettent de modifier l'échelle de l'abscisse de variables discrètes et de variables continues respectivement. Une version de ces fonctions existe aussi pour l'axe des ordonnées. Ces fonctions permettent de changer les limites des axes, les valeurs des axes, l'étiquette des valeurs des axes, la position de l'axe, etc. Dans l'exemple suivant, nous changerons la valeur des abscisses pour représenter les catégories de la variable "eyes" qui représente l'activité de l'ouverture des yeux du patient à l'urgence. Aussi, nous changerons la position ainsi que les valeurs présentées sur l'axe des ordonnées.

```
ggplot(data = subset(apache_variables, !is.na(eyes))) +
  geom_bar(mapping = aes(x = eyes)) +
  labs(x = "Ouverture des yeux", y = "Effectif") +
  scale_x_discrete(labels = c("auncun","à la douleur","à la demande","spontanée")) +
  scale_y_continuous(breaks = c(400,800,1200,1600), position = "right")
```



Légende

Si nous avons plusieurs courbes ou plusieurs groupes dans un même graphique, il est souhaitable d'avoir une légende qui accompagne pour bien illustrer quelles courbes, points ou couleurs représentent quels groupes. Dans les objets `ggplot`, une légende est ajoutée automatiquement lorsqu'on associe une variable à un élément de couleur ou de forme dans l'aesthétique, soit par l'élément `fill=`, `color=` ou `shape=`. Les noms affichés dans la légende vont alors être choisis automatiquement avec l'information présente dans les données. Cela dit, nous pouvons changer la légende en modifiant les attributs de l'élément choisi. Avec la commande `scale_colour_manual()`, `scale_fill_manual()` ou `scale_shape_manual()` nous renommerons la légende `name=`), les étiquettes (`labels=`) et même les couleurs (`values=`). Aussi, avec la fonction `theme()` nous pouvons changer la position et les formes de notre légende.

```
p <- ggplot(data = apache_variables) +
  geom_point(mapping = aes(x = respiratoryrate, y = heartrate, color = vent))

p + scale_colour_manual(
    name = "",
    values = c("0" = "navy", "1" = "orangered3"),
    labels = c("0" = "sans vent", "1" = "avec vent"),
  ) +
  theme(
    legend.position = "bottom"
    )
```

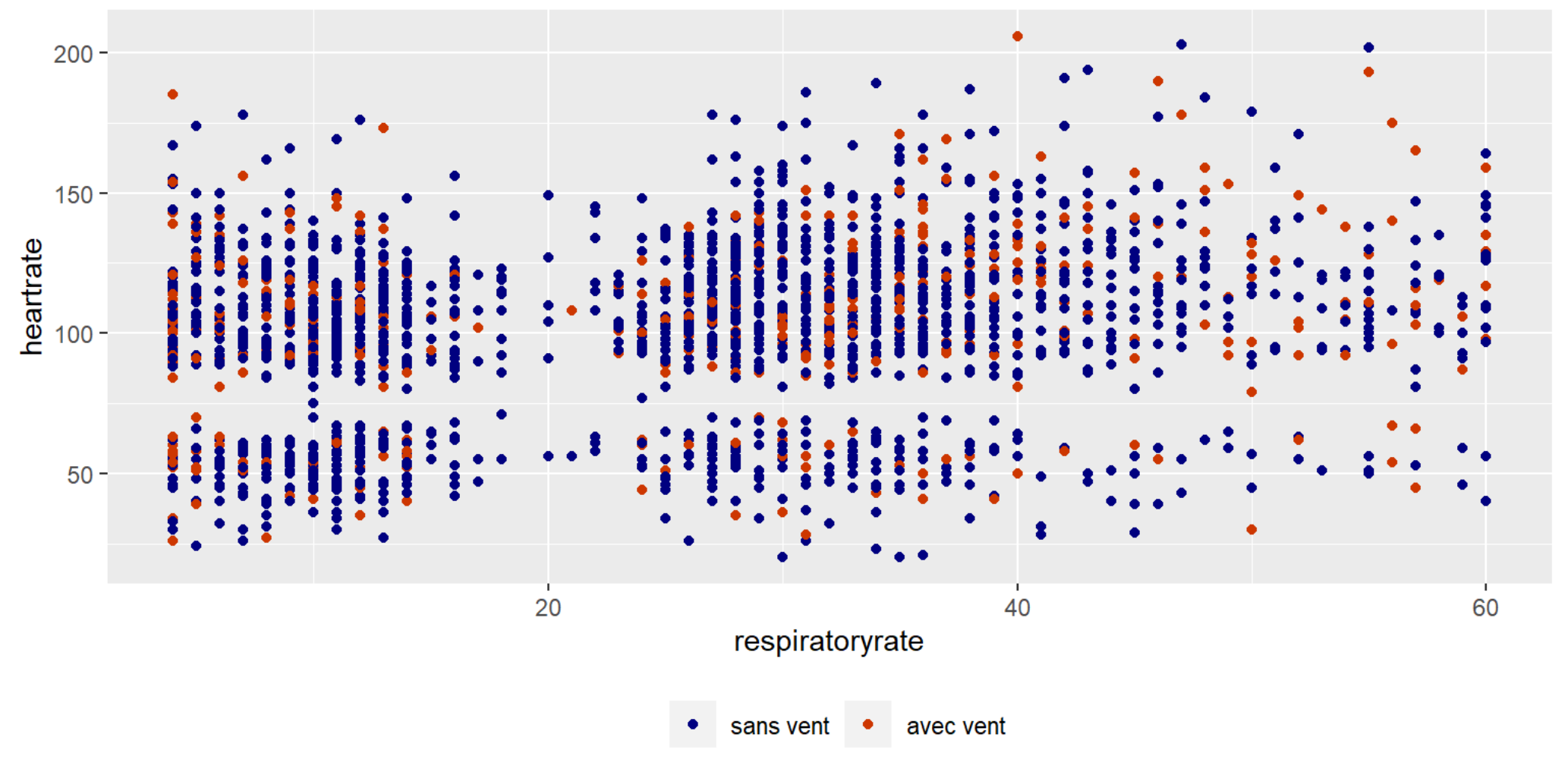
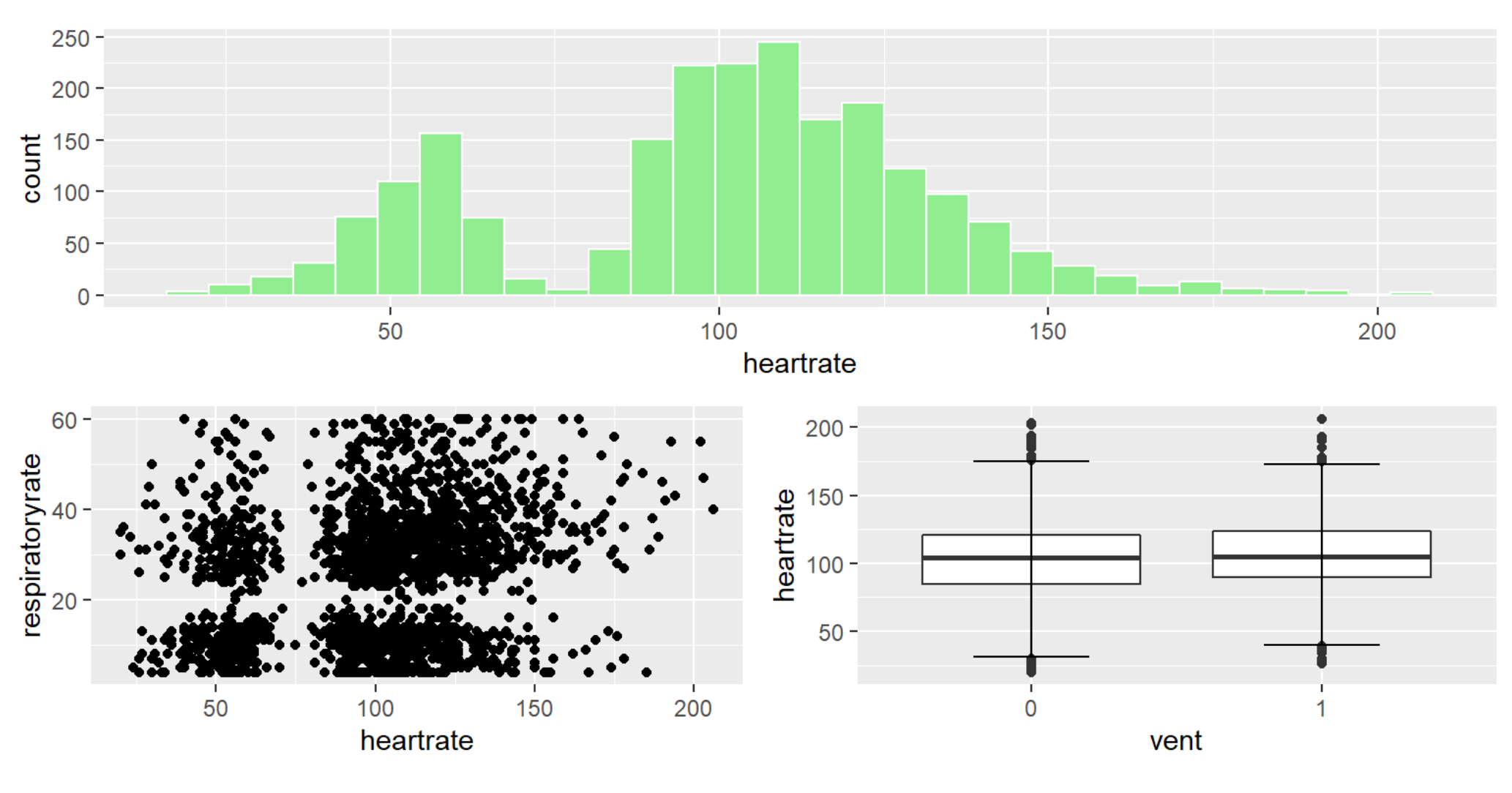


Figure multiple

Dans certains cas, nous voudrions grouper plusieurs de nos graphiques dans la même figure. La fonction `grid_arrange()` de la librairie `gridExtra` permet de positionner des graphiques dans une grille construite à partir d'une matrice. Cette matrice va indiquer quel graphique occupera quelle section de notre figure.

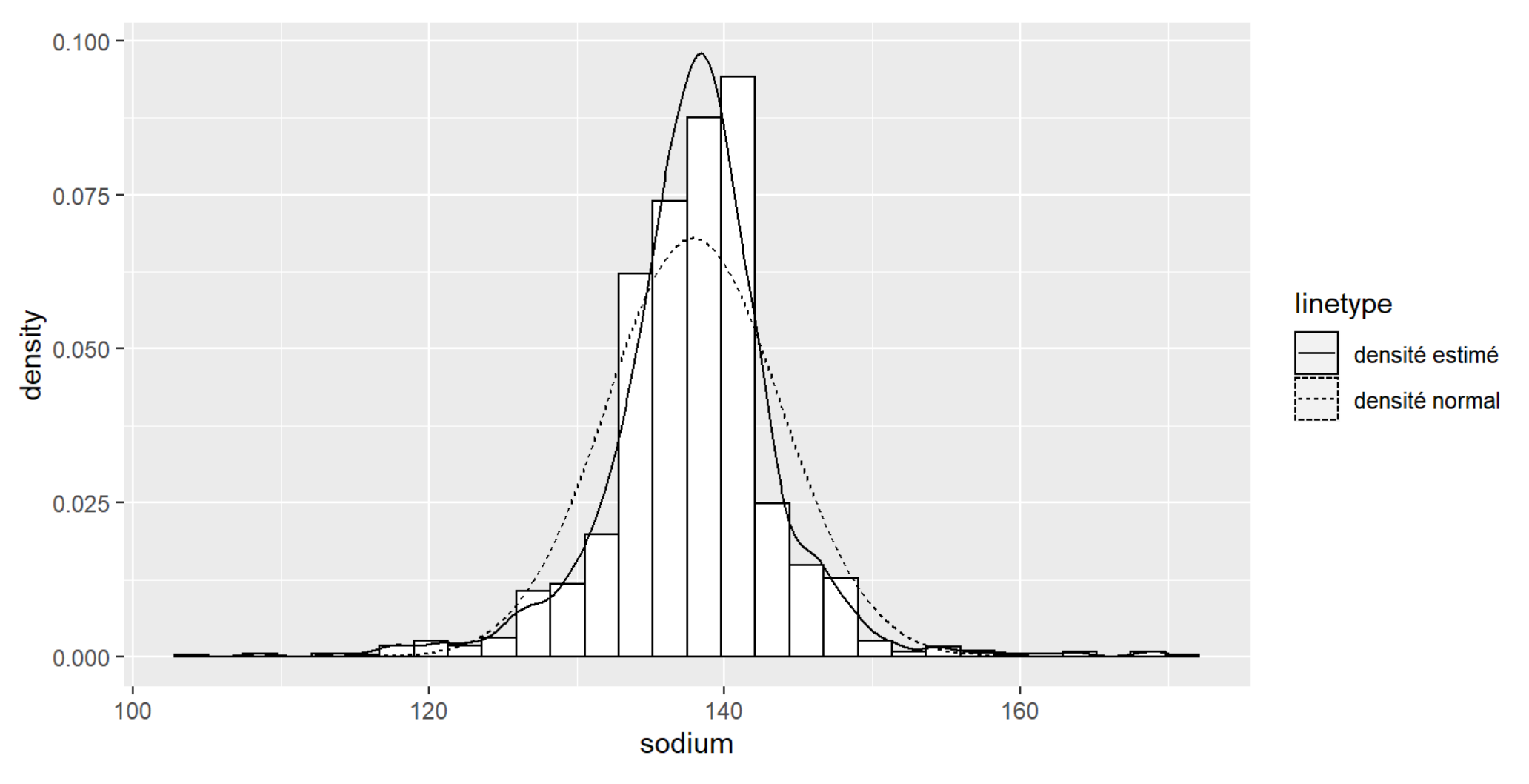
```
# chargement de la librairie gridExtra
library(gridExtra)

# histogramme avec de la couleur
p1 <- ggplot(data = apache_variables, mapping = aes(x = heartrate)) +
  geom_histogram(fill = "lightgreen", color = "white")

p2 <- ggplot(data = apache_variables) +
  geom_point(mapping = aes(x = heartrate, y = respiratoryrate))

p3 <- ggplot(data = apache_variables, mapping = aes(y = heartrate, x = vent)) +
  geom_boxplot() +
  stat_boxplot(geom = "errorbar", width = 0.4)

grid.arrange(p1, p2, p3, layout_matrix = matrix(c(1,2,1,3), nrow = 2))
```



Courbes de tendance et de densité

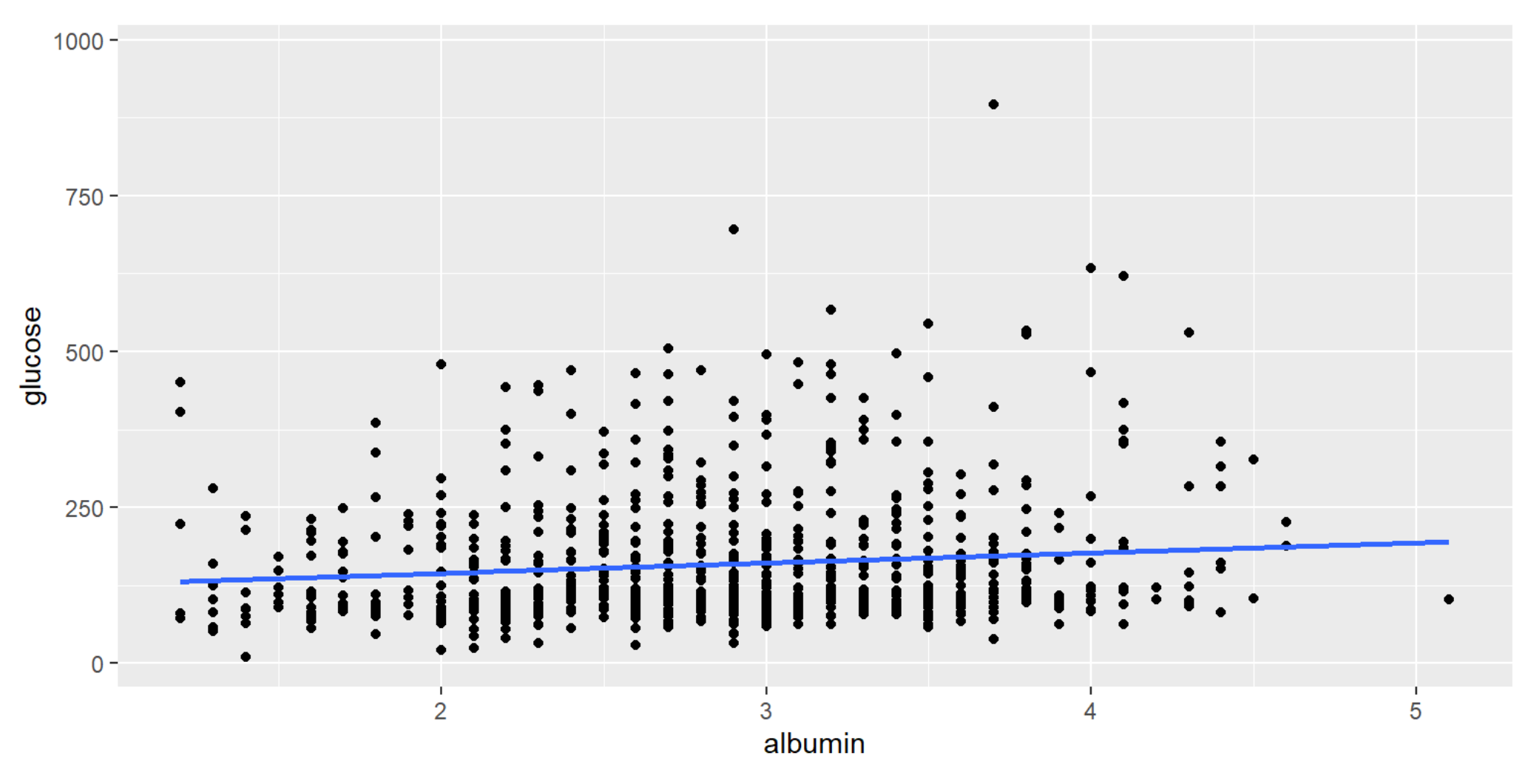
Il nous est possible d'ajouter des courbes de tendances et des courbes de densité sur nos graphiques. Par exemple, pour un histogramme, nous pouvons superposer la courbe de densité obtenue avec la fonction `geom_density()`. Il nous est aussi possible d'ajouter des fonctions avec la commande `stat_function()` qui, dans ce cas, va nous permettre d'ajouter la densité d'une loi normale à notre histogramme.

```
p <- ggplot(data = apache_variables, mapping = aes(x = sodium)) +
  geom_histogram(mapping = aes(y = after_stat(density)), color="black", fill="white")

p <- p + geom_density(aes(linetype = "densité estimé")) # ajout d'une densité de l'histogramme

# ajout d'une densité d'une loi normale de moyenne et ecart-type calculés avec les données
p + stat_function(
  aes(linetype = "densité normal"),
    fun = dnorm,
    args = list(mean = mean(apache_variables$sodium, na.rm = TRUE),
                sd = sd(apache_variables$sodium, na.rm = TRUE)))
```



Pour ajouter une courbe de tendance à un nuage de points, la fonction `geom_smooth()` permet d'ajuster une courbe à notre graphique.

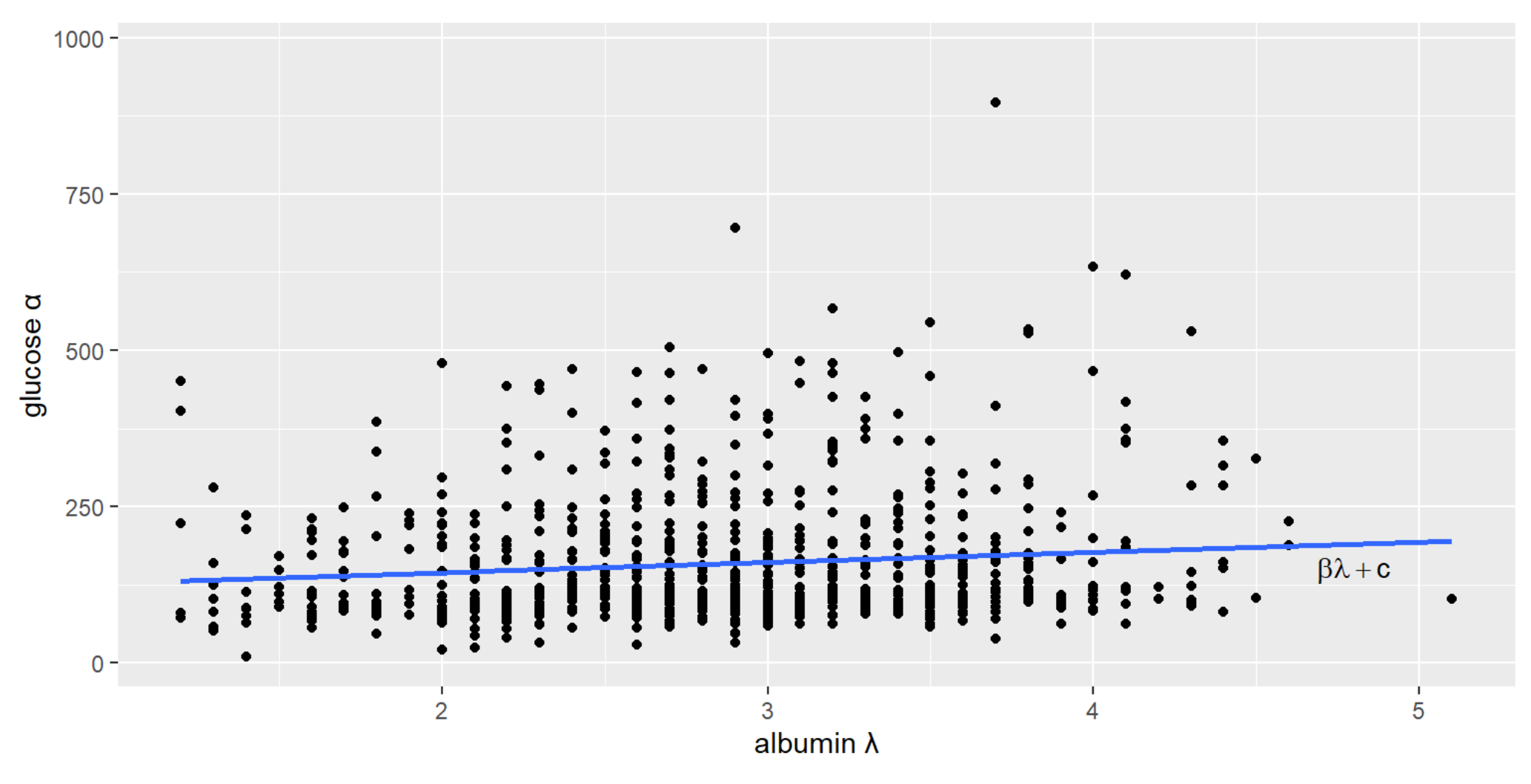
```
ggplot(data = apache_variables, mapping = aes(y = glucose, x = albumin)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE)
```



Annotation et symbole

Il arrive que nous souhaitons utiliser des caractères spéciaux ou des expressions mathématiques dans les textes de notre graphique. Avec la fonction `expression()`, il nous est possible d'insérer des formules mathématiques alors que les symboles peuvent être ajoutés en utilisant les unicodes standards pour les caractères spéciaux. Nous ajoutons ces symboles en utilisant "\u" suivi du code correspondant au caractère qui nous intéresse. Les unicodes peuvent être trouvés dans une liste sur le web comme à l'adresse https://en.wikibooks.org/wiki/Unicode/Character_reference/0000-0FFF.

```
ggplot(data = apache_variables, mapping = aes(y = glucose, x = albumin)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  labs(x = "albumin \u03BB",
       y = "glucose \u03B1") +
  annotate("text",
           x = 4.8,
           y = 150,
           label = (expression(paste(beta,lambda+c))))
```



Sauvegarde

Une fois satisfaits de notre représentation graphique, nous pouvons l'exporter afin de la joindre à un autre document ou travail. Il nous est possible d'enregistrer notre image sous plusieurs formats. La fonction `ggsave()` permet d'enregistrer un graphique fait à l'aide de la fonction `ggplot()`. Il suffit de spécifier le nom de fichier sous lequel nous voulons enregistrer notre objet ggplot avec le format de fichier de l'image (.png, .pdf, .eps, etc.). Il nous est aussi possible de fixer la taille de notre figure avec les options `width=` et `height=` dont les unités peuvent être spécifiées avec l'option `units=` en pouces "in", en centimètres "cm", en millimètres "mm" ou en pixels "px".

```
# graphique que je souhaite enregistrer.
p <- ggplot(data = apache_variables, mapping = aes(x = heartrate)) +
  geom_histogram(fill = "lightgreen", color = "white") +
  xlab("Fréquence cardiaque") +
  ylab("Effectifs") +
  ggtitle("Histogramme de la fréquence cardiaque") +
  theme(plot.title = element_text(size = 18),
        axis.title = element_text(size = 12))

# sauvegarde format pdf
ggsave("histogramme.pdf", p)

# sauvegarde format png
ggsave("histogramme.png", p, height = 900, width = 1800, units = "px")

# sauvegarde format eps
ggsave("histogramme.eps", p)
```