# Les packages ggrepel, ggthemes et paletteer

William Poisson et Mallorie Trottier-Lavoie

2020-03-21

## Table des matières

**1 Le package `ggrepel` 1**
- 1.1 Mise en contexte . . . 1
- 1.2 Le graphique de base pour `ggrepel` . . . 2
- 1.3 Quelques exemples de graphiques avec la fonction `geom_text_repel` . . . 3
- 1.4 Quelques exemples de graphiques avec la fonction `geom_label_repel` . . . 5
- 1.5 Nos impressions sur `ggrepel` . . . 7

**2 Le package `ggthemes` 7**
- 2.1 Mise en contexte . . . 7
- 2.2 Quelques exemples de thèmes avec `ggthemes` . . . 8
- 2.3 Quelques exemples d'échelles avec `ggthemes` . . . 12
- 2.4 La fonction `geom_tufteboxplot` de `ggthemes` . . . 14
- 2.5 La fonction `bank_slopes` de `ggthemes` . . . 18
- 2.6 Nos impressions sur `ggthemes` . . . 18

**3 Le package `paletteer` 18**
- 3.1 Mise en contexte . . . 18
- 3.2 Les palettes discrètes à largeur fixe . . . 19
- 3.3 Les palettes discrètes dynamiques . . . 21
- 3.4 Les palettes continues . . . 22
- 3.5 Nos impressions du package `paletteer` . . . 24

**4 Exemple combinant les packages `ggthemes` et `paletteer` 24**

**5 Exemple combinant les trois packages présentés 26**

**6 Références 27**

# 1 Le package `ggrepel`

## 1.1 Mise en contexte

La création d'un graphique est le moyen de prédilection afin de visualiser et faire parler des données. Le package `ggplot2` permet de créer des graphiques de façon simple dans R. Encore faut-il que ces graphiques aient une interface conviviale au premier regard. Pour se faire, plusieurs extensions de `ggplot2` ont été mises sur pied, tel que le package `ggrepel`. `ggrepel` a été créé en 2016 par Kamil Slowikowski. Comme son nom l'indique, ce package fait en sorte de repousser les étiquettes afin qu'elles ne soient pas superposées dans le graphique, ou encore qu'elles ne soient pas par-dessus les points ou trop près des limites du graphique. Ceci permet ainsi une visualisation beaucoup plus claire de ces dites étiquettes et par le fait même du graphique.

Plusieurs fonctions composent le package `ggrepel`, nous verrons ci-bas une description de `ggrepel` et de ses fonctions.

D'abord, il faut charger le package `ggplot2` et `ggrepel` :

```
library(ggplot2)
library(ggrepel)
```

Si vous n'avez jamais installé ces packages, il suffit de les télécharger à l'aide de la fonction `install.packages` avant de les charger avec la fonction `library`.

Les exemples suivants seront construits à partir du jeu de données `msleep` du package `ggplot2`. Il s'agit d'un data frame de 83 observations et 11 variables, contenant des temps de sommeil et d'éveil, ainsi que des poids de mammifères. Les variables utilisées dans ce jeu de données seront la longeur du cycle de sommeil (`sleep_cycle`) ainsi que la durée totale du sommeil (`sleep_total`). Voici une allure de ce jeu de données :

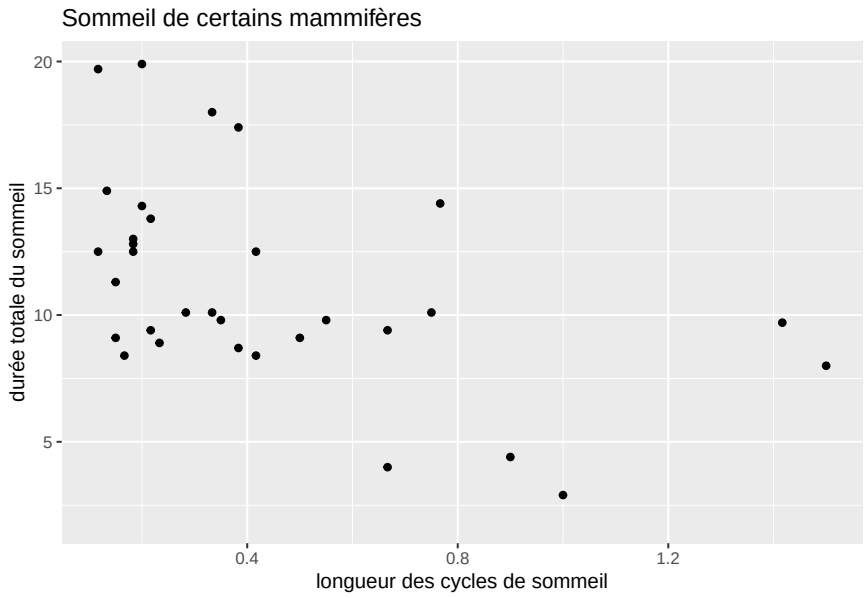
```
str(msleep)
```

```
Classes 'tbl_df', 'tbl' and 'data.frame':   83 obs. of  11 variables:
 $ name        : chr  "Cheetah" "Owl monkey" "Mountain beaver" "Greater short-tailed shrew" ...
 $ genus       : chr  "Acinonyx" "Aotus" "Aplodontia" "Blarina" ...
 $ vore        : chr  "carni" "omni" "herbi" "omni" ...
 $ order       : chr  "Carnivora" "Primates" "Rodentia" "Soricomorpha" ...
 $ conservation: chr  "lc" NA "nt" "lc" ...
 $ sleep_total : num  12.1 17 14.4 14.9 4 14.4 8.7 7 10.1 3 ...
 $ sleep_rem   : num  NA 1.8 2.4 2.3 0.7 2.2 1.4 NA 2.9 NA ...
 $ sleep_cycle : num  NA NA NA 0.133 0.667 ...
 $ awake       : num  11.9 7 9.6 9.1 20 9.6 15.3 17 13.9 21 ...
 $ brainwt     : num  NA 0.0155 NA 0.00029 0.423 NA NA NA 0.07 0.0982 ...
 $ bodywt      : num  50 0.48 1.35 0.019 600 ...
```

## 1.2 Le graphique de base pour ggrepel

Les exemples suivants seront appliqués sur le jeu de données `msleep` original. Le graphique de base avec lequel nous travaillerons est sous forme d'un diagramme de dispersion fait à partir de `ggplot2`, pour lequel l'axe des x représente la longueur des cycles de sommeil et l'axe des y la durée totale du sommeil.

```
graph_base <- ggplot(data = msleep) +
  geom_point(mapping = aes(x = sleep_cycle, y = sleep_total)) +
  labs(title = "Sommeil de certains mammifères") +
  labs(x = "longueur des cycles de sommeil") +
  labs(y = "durée totale du sommeil")
graph_base
```

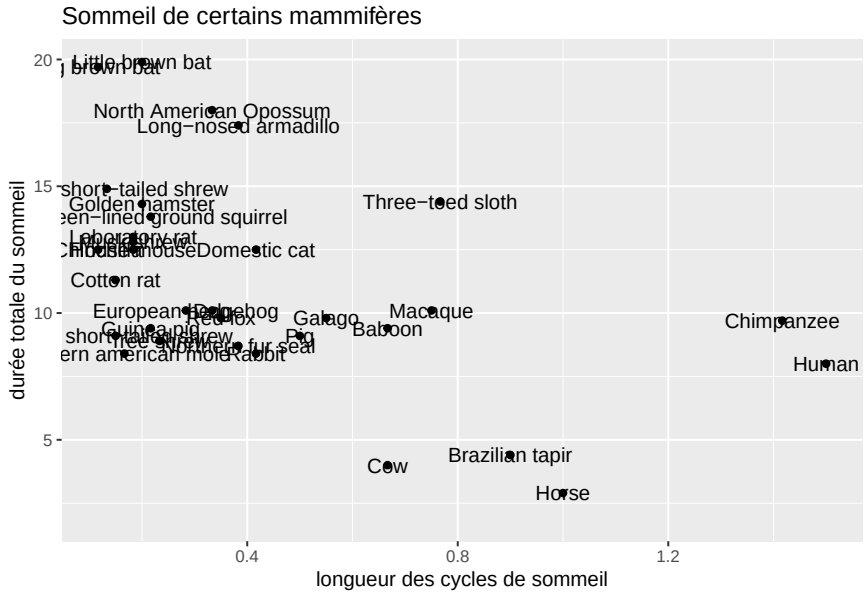



## 1.3 Quelques exemples de graphiques avec la fonction `geom_text_repel`

La fonction `geom_text_repel` est utilisée afin de faire l'ajout des étiquettes représentant le nom de chaque animal et ce directement dans le graphique. Cette fonction permet, comparativement à la fonction `geom_text` du package `ggplot2`, de placer les étiquettes de façon à ce que chacune d'elles soit lisible et associable à son point. Le premier graphique qui suit montre un exemple utilisant la fonction `geom_text` du package `ggplot2`. Comme il est possible de l'observer, les étiquettes sont superposées par endroit et il est très difficile d'associer chaque point à son étiquette respective. Le deuxième graphique utilise la fonction `geom_text_repel` et démontre la bonification en termes de lisibilité par rapport à `geom_text`.

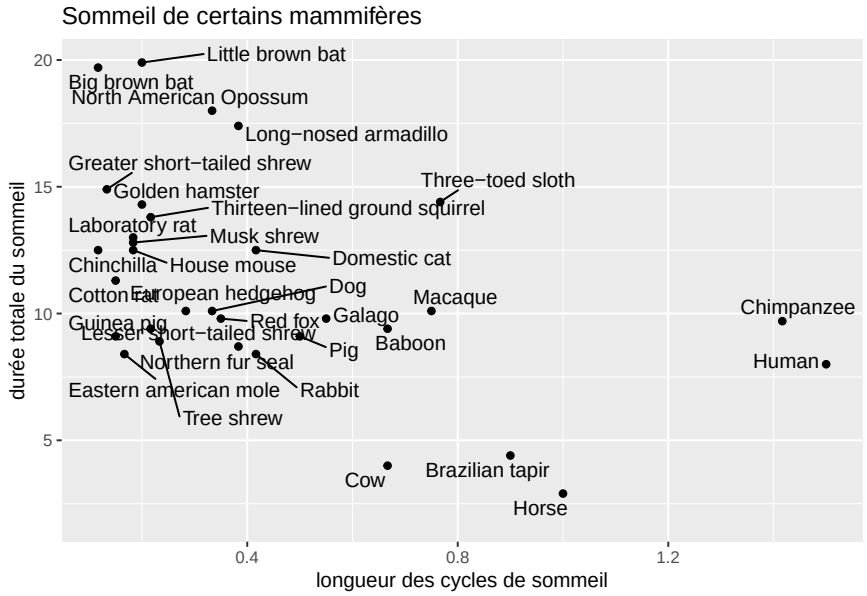
Avec `geom_text` :

```
graph_text <- ggplot(data = msleep) +
  geom_point(mapping = aes(x = sleep_cycle, y = sleep_total)) +
  labs(title = "Sommeil de certains mammifères") +
  labs(x = "longueur des cycles de sommeil") +
  labs(y = "durée totale du sommeil") +
  geom_text(aes(x = sleep_cycle, y = sleep_total, label = name))
graph_text
```



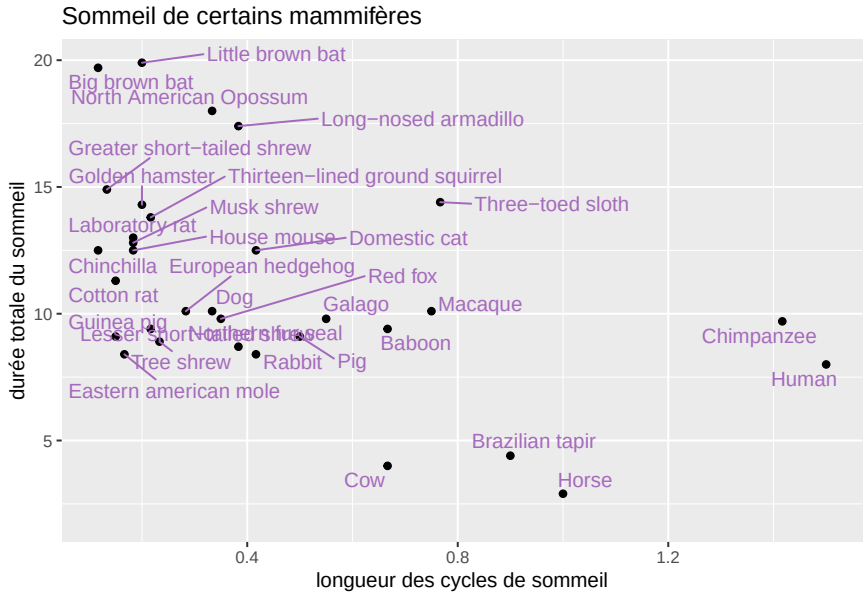

Avec `geom_text_repel` :

```
graph_text_repel <- graph_base +
  geom_text_repel(aes(x = sleep_cycle, y = sleep_total, label = name))
graph_text_repel
```




Si on revient à la fonction `geom_text_repel`, plusieurs arguments permettent de faire des modifications. Par exemple, il est possible d'augmenter la force de répulsion entre les étiquettes afin qu'elles soient davantage éloignées. Il s'agit de l'argument `force`, qui a comme valeur par défaut 1 (Slowikowski, 2019). De plus, il est possible de modifier la couleur du texte et du segment de l'étiquette avec l'argument `color`. Par défaut, la valeur de cet argument est "black" (Slowikowski, 2019). Voici un exemple avec une force de répulsion de 2 et la couleur #A569BD :

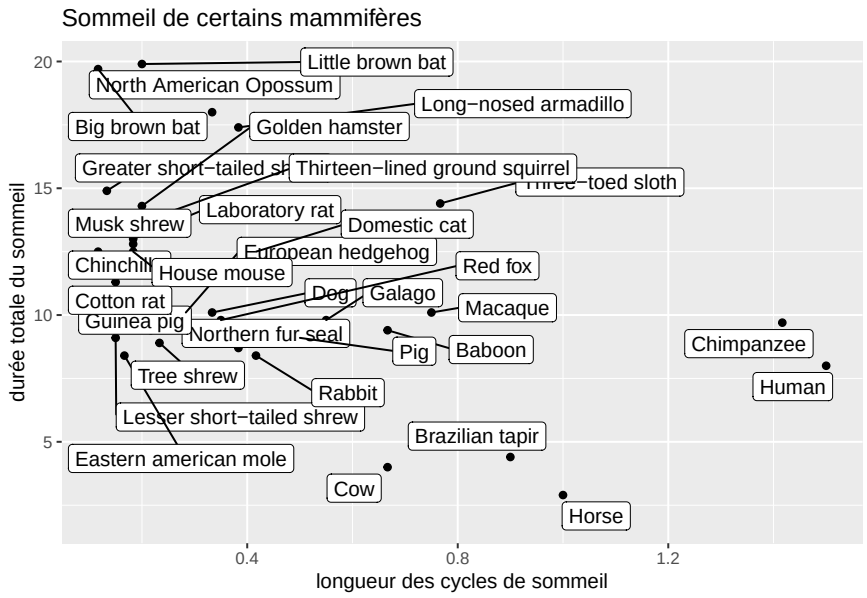
```
graph_base +
  geom_text_repel(
    aes(x = sleep_cycle, y = sleep_total, label = name),
    force = 2,
    color = "#A569BD"
  )
```




## 1.4 Quelques exemples de graphiques avec la fonction `geom_label_repel`

La fonction `geom_label_repel` fait l'ajout d'étiquettes sous forme de boîte. Cette mise en forme permet de mieux lire chacun des noms. En voici un exemple :

```
graph_label_repel <- graph_base +
  geom_label_repel(aes(x = sleep_cycle, y = sleep_total, label = name))
graph_label_repel
```

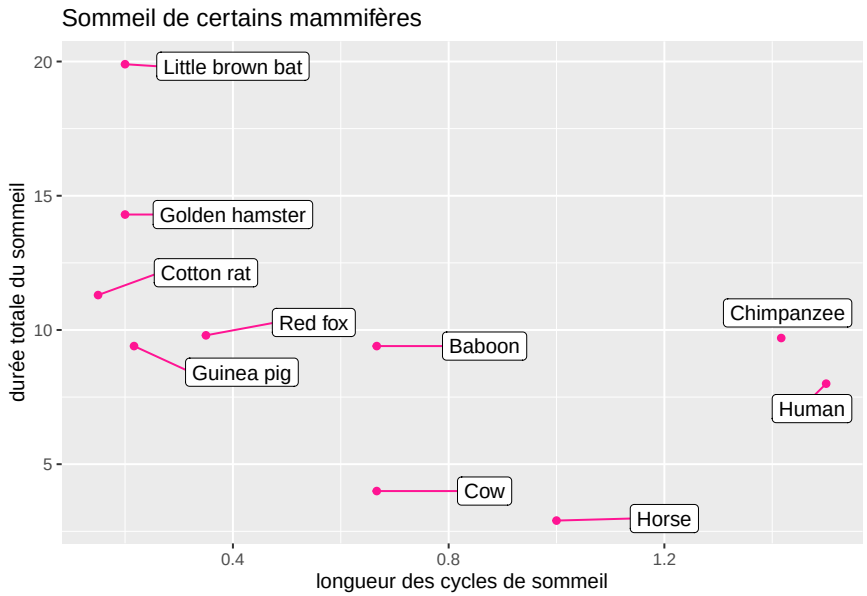



Comme il est possible d'observer dans l'exemple précédent, lorsque le graphique contient beaucoup de données rapprochées, les boîtes peuvent faire en sorte de cacher des points. Ainsi, lorsque l'on utilise cette fonction, il peut être nécessaire d'utiliser des arguments afin de distancer et modifier les boîtes. Par exemple, l'argument `box.padding` permet de changer le rembourrage autour de la boîte, qui par défaut est de 0.25, ce qui éloigne les boîtes l'une de l'autre (Slowikowski, 2019). Un autre argument intéressant est `direction` qui a par défaut la valeur "both", il est cependant possible de lui attribuer la valeur "y" (déplace les étiquettes verticalement) ou encore la valeur "x" (déplace les étiquettes horizontalement) (Slowikowski, 2019). Un autre argument intéressant afin de modifier la disposition des boîtes est `nudge_x` qui positionne les étiquettes à partir d'un certain point sur l'axe des x, ou `nudge_y`, qui fait la même chose, mais sur l'axe des y (Slowikowski, 2019). Leur valeur par défaut est 0 (Slowikowski, 2019).

Nous avons reproduit un exemple du graphique ci-haut en sélectionnant 10 animaux du jeu de données `msleep` afin de démontrer que la fonction `geom_label_repel` peut être très intéressante lorsque notre jeu de données est moins chargé. Nous avons d'ailleurs utilisé les arguments mentionnés dans les lignes précédentes, en plus d'ajouter l'argument `segment.colour` qui permet de changer la couleur du segment reliant le point à son étiquette (Slowikowski, 2019).

```
selection <- msleep[
  msleep$name %in% c("Horse", "Cow", "Baboon", "Human", "Little brown bat", "Guinea pig",
                     "Red fox", "Chimpanzee", "Golden hamster", "Cotton rat"),
]

ggplot(data = selection) +
  geom_point(
    mapping = aes(x = sleep_cycle, y = sleep_total), color = "deeppink") +
    labs(title = "Sommeil de certains mammifères") +
    labs(x = "longueur des cycles de sommeil") +
    labs(y = "durée totale du sommeil"
  ) +
  geom_label_repel(
    aes(x = sleep_cycle, y = sleep_total, label = name),
    box.padding = unit(0.40, "lines"),
    direction = "y",
    nudge_x = 0.2,
```

```
    segment.colour = "deeppink"
)
```




### 1.5 Nos impressions sur `ggrepel`

Suite à ces différents exemples effectués pour les deux fonctions du package `ggrepel`, il est intéressant de remarquer que, dans le cas de notre graphique de base, la fonction `geom_text_repel` permet d'obtenir une distribution des étiquettes plus conviviale que ce qui a été observé pour le graphique faisant appel à la fonction `geom_label_repel`. Ainsi, selon le type de données que l'on a, il est pertinent d'utiliser l'une ou l'autre des deux fonctions et les nombreux arguments offerts afin d'optimiser la convivialité de la disposition des étiquettes. Somme toute, `geom_text_repel` et `geom_label_repel` possèdent tous les arguments (et plusieurs autres arguments) retrouvés dans les fonctions `geom_text` ou `geom_label` du package `ggplot2`, excepté l'argument `check_overlap`.

## 2 Le package `ggthemes`

### 2.1 Mise en contexte

`ggthemes` est un package permettant de bonifier le répertoire de `ggplot2` en modifiant facilement l'allure des graphiques. Il a été créé par Jeffrey B. Arnold en 2012, puis optimisé depuis par plusieurs autres auteurs (Arnold, 2020a).

Ce package contient plusieurs palettes de couleurs, dispositions d'échelles (`scale`) et thèmes graphiques (`theme`). Il contient aussi des fonctions associées aux types de graphiques (`geom`) utilisés par le savant professeur Edward Tufte (Arnold, 2016 ; Yutani). Ainsi, l'utilisation de `ggthemes` permet, entre autres, d'emprunter les styles graphiques de journaux connus tels le journal The Economist, le Journal de Wall Street, de logiciels tels *Calc* et *Excel* , ou encore des styles intéressants pour présenter des données (Arnold, 2016). Ce package contient également une fonction permettant de bien illustrer les fluctuations représentées dans les graphiques à ligne brisée (Arnold, 2016).

Les exemples qui seront présentés se basent sur un jeu de données du package `datasets` nommé `iris` qui contient des données sur les longueurs et largeurs des pétales et des sépales de trois espèces d'iris. Voici un

aperçu de ce jeu de données :

```
head(iris)
```

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

Pour effectuer les commandes suivantes, certains packages doivent être chargés comme suit :

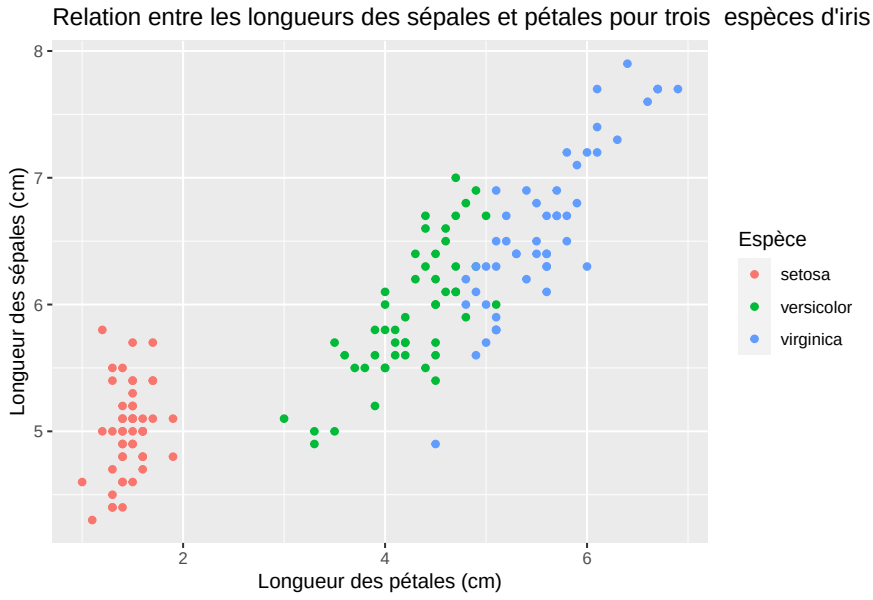
```
library(ggthemes)
library(gridExtra)
```

Si vous n'avez jamais installé ces packages, il suffit de les télécharger à l'aide de la fonction `install.packages` avant de les appeler avec la fonction `library`.

## 2.2 Quelques exemples de thèmes avec `ggthemes`

Les exemples de thèmes suivants s'appliqueront au jeu de données `iris` original. Voici un graphique de base des données de la longueur des sépales en fonction de la longueur des pétales :

```
graph_iris <- ggplot(data = iris) +
  geom_point(mapping = aes(x = Petal.Length, y = Sepal.Length, colour = Species)) +
  labs(
    title = "Relation entre les longueurs des sépales et pétales pour trois  espèces d'iris",
    x = "Longueur des pétales (cm)",
    y = "Longueur des sépales (cm)",
    colour = "Espèce"
  )
graph_iris
```



Il est possible d'observer sur ce graphique l'interaction entre la longueur des sépales et celle des pétales pour trois différentes espèces d'iris.

Une fois le graphique de base généré, il devient alors facile de changer les thèmes. Les sections suivantes montrent quelques exemples.

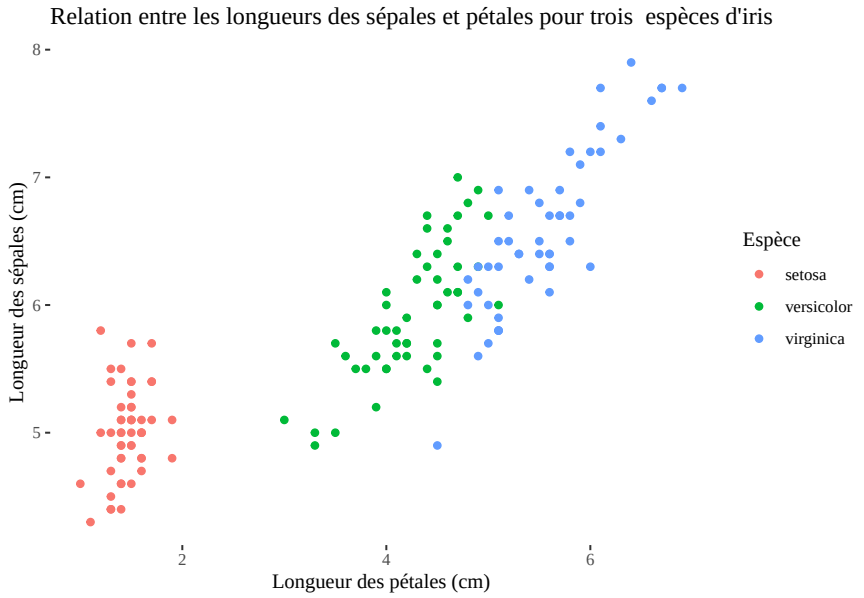
Une liste des différents thèmes est également détaillée sur https://mran.microsoft.com/snapshot/2017-02-04/web/packages/ggthemes/vignettes/ggthemes.html.

Il est à noter que divers arguments peuvent également être ajoutés aux fonctions `theme_` pour en modifier l'apparence. Ces arguments ne seront pas illustrés ici, mais ils sont détaillés dans la section `Help` de `RStudio`.

### 2.2.1 `theme_tufte`

Le thème *tufte* permet de créer un graphique de style simpliste basé sur le livre *The Visual Display of Quantitative Information* d'Edward Tufte (Arnold, 2016). En voici un exemple :

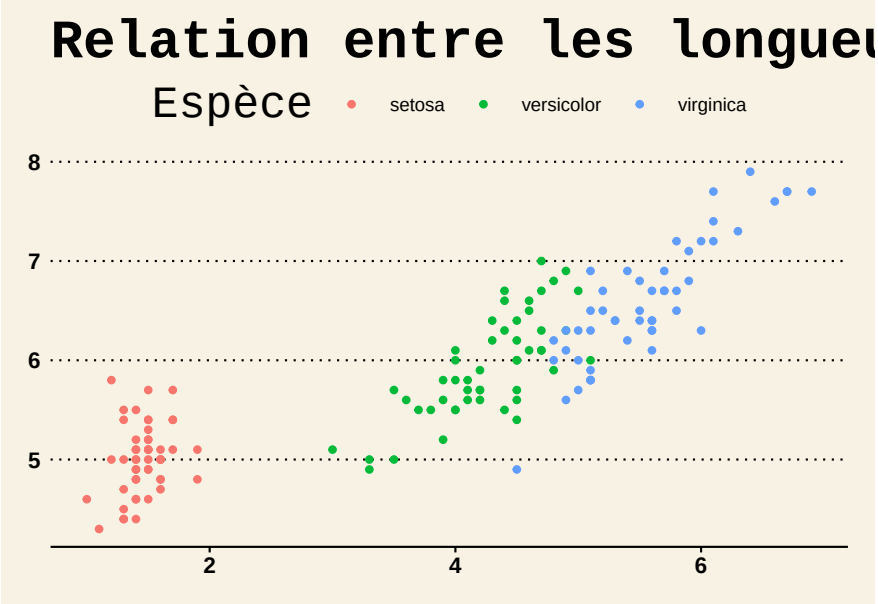
```
graph_iris +
  theme_tufte()
```




### 2.2.2 `theme_wsj`

Le thème *wsj* correspond au thème utilisé dans le *Wall Street Journal* (Arnold, 2016).

```
graph_iris +
  theme_wsj()
```

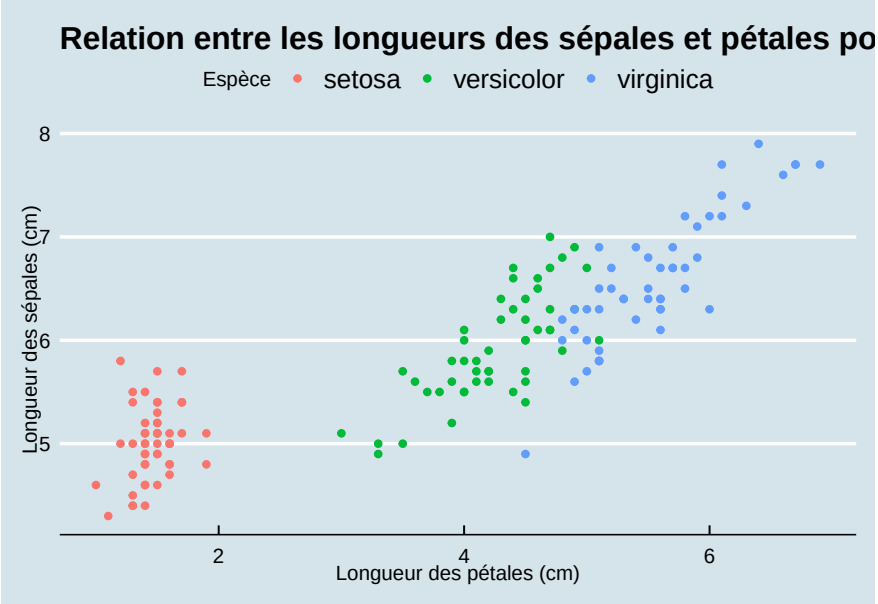



Notez que la taille de police du thème `wsj` entraine un dépassement important du titre. Cela peut représenter un problème associé à certains thèmes comme celui-ci. Il serait alors possible d'ajouter des `\n` pour mettre le titre sur plusieurs lignes. L'argument `base_size` de la fonction `theme_wsj` permet également de diminuer la taille de police, mais cette diminution s'applique à l'ensemble du texte du graphique, rendant parfois les textes d'axes illisibles.

### 2.2.3 `theme_economist`

Le thème *economist* permet pour sa part de recréer la mise en forme des graphiques du journal *The Economist* (Arnold, 2016).

```
graph_iris +
  theme_economist()
```

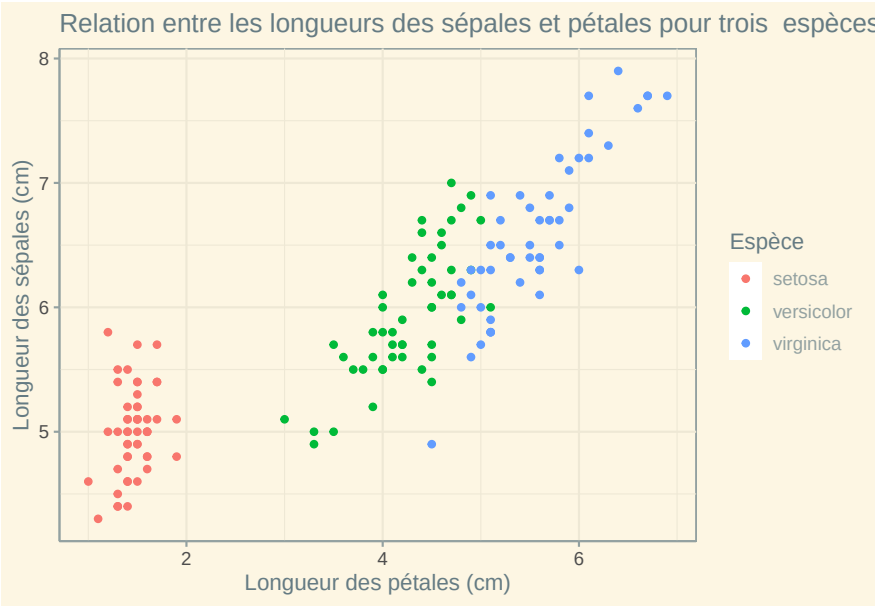



Notez que la taille de police du thème *economist* entraine aussi un dépassement important du titre. Les mêmes corrections que pour le thème *wsj* peuvent alors s'appliquer.

### 2.2.4 `theme_solarized`

Le thème *solarized* utilise les couleurs de la palette solarized (Arnold, 2016). En voici un exemple :

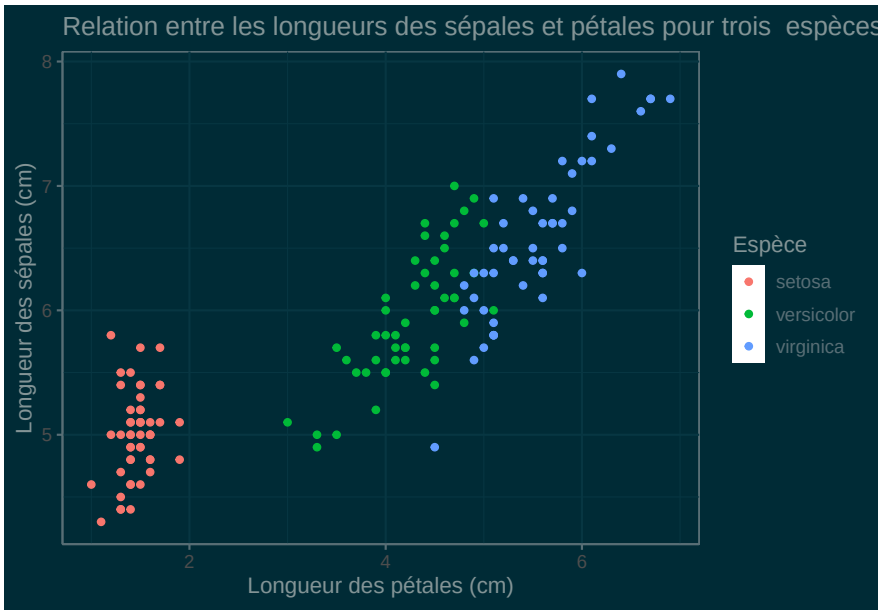
```
graph_iris +
  theme_solarized()
```




Bien que ce thème de graphique ne sort pas de l'ordinaire, il est particulièrement intéressant en version foncée

où il permet un bon contraste sur fond clair, rajoutant alors de l'emphase sur le graphique.

```
graph_iris +
  theme_solarized(light = FALSE)
```



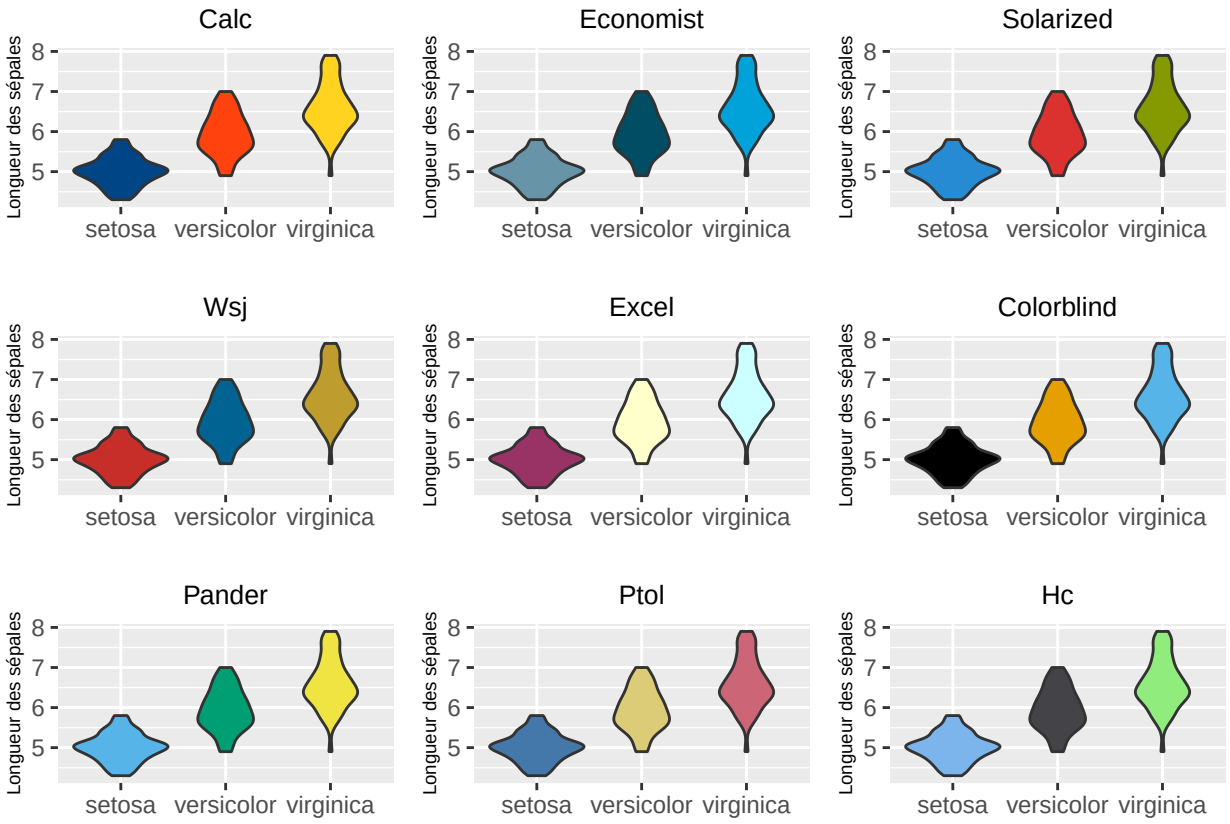

Notez que la taille de police du thème *solarized* entraine également un dépassement du titre. Encore une fois, les corrections soulevées pour les thèmes *wsj* et *economist* peuvent s'appliquer.

## 2.3 Quelques exemples d'échelles avec `ggthemes`

Le package `ggthemes` permet d'appliquer différentes palettes de couleurs associées, comme c'était le cas des thèmes, à des journaux, des logiciels ou autres. La mosaïque suivante contient un graphique de la longueur des sépales en fonction de l'espèce d'iris en utilisant les trois premières couleurs de différentes palettes de `ggthemes`.

```
base_violin <- ggplot(data = iris) +
  geom_violin(
    mapping = aes(x = Species, y = Sepal.Length, fill = Species),
    show.legend = FALSE
  ) +
  labs(x = "", y = "Longueur des sépales") +
  theme(plot.title = element_text(size=10)) +
  theme(axis.title = element_text(size=7)) +
  theme(plot.title = element_text(hjust = 0.5))

calc <- base_violin +
  labs(title = "Calc") +
  scale_fill_calc()

wsj <- base_violin +
  labs(title = "Wsj") +
  scale_fill_wsj()

excel <- base_violin +
```

```
  labs(title = "Excel") +
  scale_fill_excel()

economist <- base_violin +
  labs(title = "Economist") +
  scale_fill_economist()

colorblind <- base_violin +
  labs(title = "Colorblind") +
  scale_fill_colorblind()

solarized <- base_violin +
  labs(title = "Solarized") +
  scale_fill_solarized()

hc <- base_violin +
  labs(title = "Hc") +
  scale_fill_hc()

pander <- base_violin +
  labs(title = "Pander") +
  scale_fill_pander()

ptol <- base_violin +
  labs(title = "Ptol") +
  scale_fill_ptol()

grid.arrange(calc, economist, solarized, wsj, excel, colorblind, pander, ptol, hc, nrow=3, ncol=3)
```




Une fonction permettant de visualiser les échelles de couleurs complètes des palettes de `ggthemes` est documentée sur le site suivant : https://gt.rstudio.com/reference/info_paletteer.html.

## 2.4 La fonction `geom_tufteboxplot` de `ggthemes`

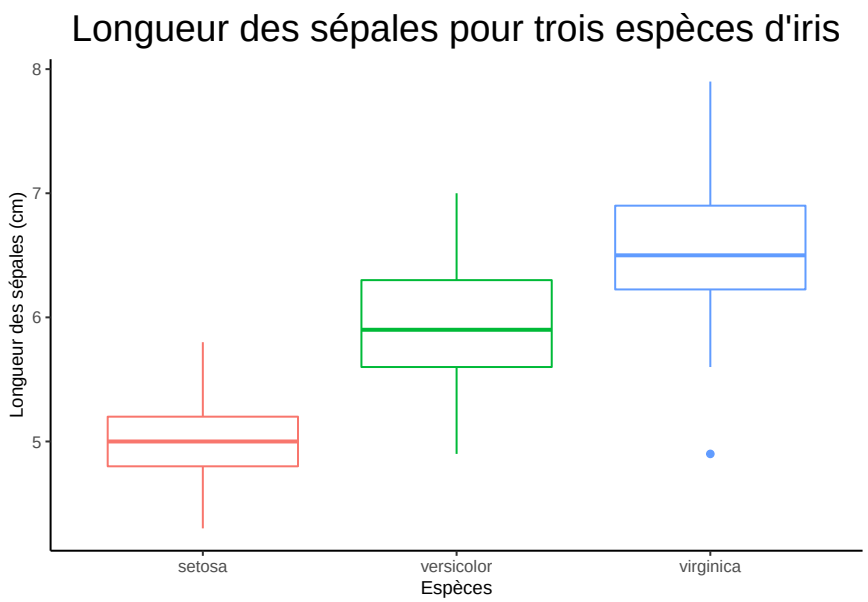
La fonction `geom_tufteboxplot` est intéressante puisqu'elle permet de créer des diagrammes en boîte simplistes constitués d'enchaînements de points, d'espaces et de traits continus pour illustrer les différentes parties des *boîtes*. Ces types de diagrammes en boîte se calquent au style simpliste d'Edward Tufte.

Voici un graphique de type *boxplot* standard fait avec la fonction `geom_boxplot` du package `ggplot2` auquel est appliqué le thème `theme_classic` pour mettre l'emphase sur le style des boîtes. Notez que pour ce graphique et les suivants, un commentaire a été inséré dans la commande R pour désigner ce qui change d'un exemple à l'autre.

```
boxplots <- ggplot(
  data = iris,
  aes(x = Species, y = Sepal.Length, colour = Species)
) +
  theme_classic() +
  labs(
    title = "Longueur des sépales pour trois espèces d'iris",
    x = "Espèces",
    y = "Longueur des sépales (cm)"
  ) +
```

```
  theme(plot.title = element_text(size = 20)) +
  theme(axis.title = element_text(size = 10)) +
  theme(plot.title = element_text(hjust = 0.5))
```

```
boxplots +
  # Fonction boxplot standard
  geom_boxplot(show.legend = FALSE)
```

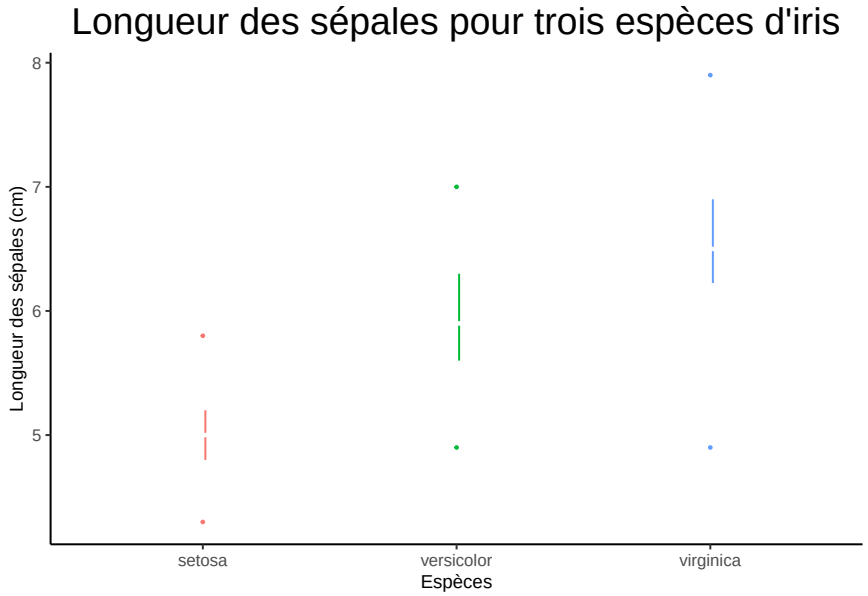



Il est possible de simplifier le graphique en utilisant `geom_tufteboxplot` de sorte que les différentes parties du diagramme en boîtes soient représentées par des enchaînements de lignes et d'espaces. La fonction géométrique de base de `geom_tufteboxplot` donne ceci :

```
boxplots +
  # Fonction geom_tufteboxplot par défaut
  # (espaces pour les espaces interquartiles et lignes pour les moustaches)
  geom_tufteboxplot(show.legend = FALSE)
```


En jouant sur les arguments `type` attribués à la médiane et aux moustaches, d'autres styles sont possibles :

```
boxplots +
  # Lignes pour les espaces interquartiles et espaces pour les moustaches
  geom_tufteboxplot(
    show.legend = FALSE,
    median.type = "line",
    whisker.type = "point"
  )
```

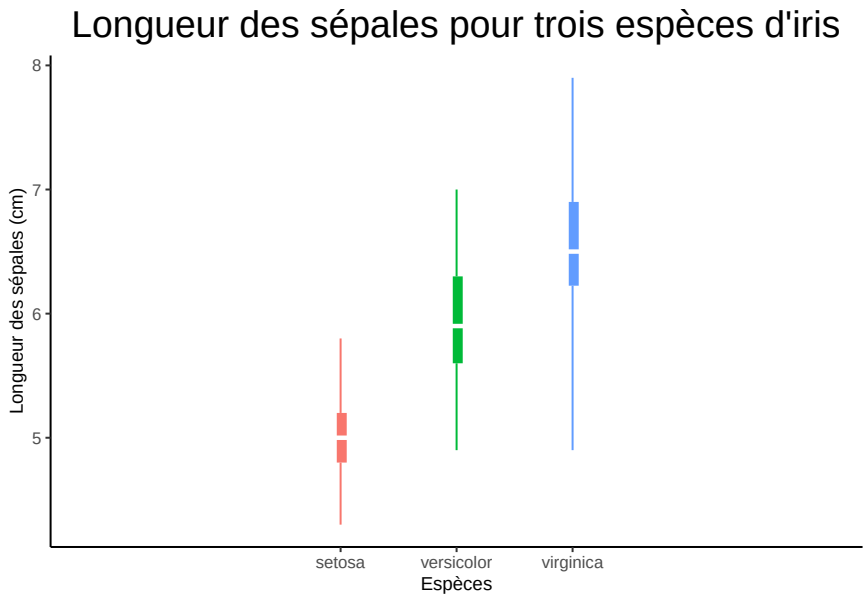



```
boxplots +
  # Lignes pour les espaces interquartiles et lignes pour les moustaches
  geom_tufteboxplot(
    show.legend = FALSE,
```

```
    median.type = "line",
    whisker.type = "line"
  )
```


Finalement, il est possible de se rapprocher du style standard de boxplot, mais où la boîte couvrant l'espace interquartile est remplacée par un trait épais et la médiane par un espace vide comme tel :

```
boxplots +
  # Fonction geom_tufteboxplot avec largeur sur les lignes interquartiles
  geom_tufteboxplot(
    show.legend = FALSE,
    median.type = "line",
    whisker.type = "line",
    width = 5
  )
```




### 2.5 La fonction `bank_slopes` de `ggthemes`

Finalement, une autre fonction appelée `bank_slopes` est disponible dans le package `ggthemes` pour optimiser l'apparence des graphiques à ligne brisée en jouant sur les dimensions du graphique et l'angle des pentes (Arnold, 2020b). Pour en savoir plus, cliquez ici.

### 2.6 Nos impressions sur `ggthemes`

La portion associée au thème de `ggthemes` est très facile d'utilisation et permet, en une fonction, de recréer des styles populaires sans toutefois devoir imposer d'innombrables paramètres. Plusieurs thèmes sont illustrés sur le web, permettant de visualiser les styles possibles avant l'exécution du graphique. Pour diverger légèrement des thèmes offerts, les diverses palettes et échelles permettent de produire des graphiques dans son propre style, mais d'y ajouter une touche connue avec la palette de couleur ou l'échelle de certains logiciels, livres ou journaux réputés.

Le principal hic soulevé est que l'utilisation de certains thèmes modifie la taille de caractère et entraine des débordements du graphique. L'argument `base_size` utile pour contrôler la taille des caractères ne résout pas le problème, puisqu'en rapetissant la taille, certains titres deviennent alors illisibles.

En ce qui a trait aux paramètres géométriques associés aux styles d'Edward Tufte, le style de nature simpliste de `geom_tufteboxplot` permet d'exprimer les données dans un graphique qui va directement au point et de façon facilement accessible. Les arguments de la fonction sont également faciles à comprendre.

## 3 Le package `paletteer`

### 3.1 Mise en contexte

Le package `paletteer` permet d'avoir accès à l'ensemble des palettes de couleurs de R provenant d'une cinquantaine de packages. De ce fait, il contient actuellement 1759 palettes (Hvitfeldt, 2020a). Ce package de regroupement a été conçu en 2018 par Emil Hvitfeldt (Hvitfeldt, 2020b).

Trois types de palettes sont présents en R et sont donc contenus dans le package. Les deux premières palettes sont regroupées dans la catégorie discrète : ce sont les palettes *discrètes* à largeur fixe et celles dites *dynamiques*. Le troisième type de palette regroupe les palettes *continues* (Hvitfeldt, 2020a). Il est possible d'obtenir les noms de palettes des trois types en lançant les commandes `palettes_d_names`, `palettes_dynamic_names` et `palettes_c_names` pour avoir les noms de palettes discrètes à largeur fixe, discrètes dynamiques et continues, respectivement.

Les palettes disponibles, disposées par package, sont visualisables au lien suivant : https://github.com/EmilHvitfeldt/r-color-palettes. D'autres palettes non disponibles avec le package `paletteer` du à leur grand nombre, sont disponibles sur ce site : http://soliton.vm.bytemark.co.uk/pub/cpt-city/index.html.

Afin de pouvoir visualiser les exemples suivants, il est nécessaire de charger le package `paletteer`.

```
library(paletteer)
```

Si vous n'avez jamais installé le package `paletteer`, il suffit de le télécharger à l'aide de la fonction `install.packages` avant de l'appeler avec la fonction `library`.

Les exemples utiliseront le jeu de données `quakes` du package `datasets`. Ces données contiennent, pour 1000 séismes survenus près des îles Fidji, les localisations géographiques (variables `lat` et `long`), la profondeur (variable `depth`), la magnitude (variable `mag`) et le nombre de stations sismiques ayant rapporté le tremblement de terre (variable `stations`).

```
str(quakes)
```

```
'data.frame':   1000 obs. of  5 variables:
 $ lat     : num  -20.4 -20.6 -26 -18 -20.4 ...
 $ long    : num  182 181 184 182 182 ...
 $ depth   : int  562 650 42 626 649 195 82 194 211 622 ...
 $ mag     : num  4.8 4.2 5.4 4.1 4 4 4.8 4.4 4.7 4.3 ...
 $ stations: int  41 15 43 19 11 12 43 15 35 19 ...
```

## 3.2 Les palettes discrètes à largeur fixe

Les palettes à largeur fixe contiennent un intervalle de couleurs distinctes défini. Ainsi, seul un certain nombre de couleurs différentes est présent dans la palette (Hvitfeldt, 2020a).

Pour visualiser une palette spécifique, la commande suivante peut être lancée en replaçant les éléments de la fonction par ceux de la palette voulue :

```
paletteer_d("NomDuPackage::NomDeLaPalette")
```

L'utilisation du « : : » permet de spécifier dans quel package aller chercher la palette de couleur. C'est une façon rapide de spécifier le chemin de recherche de la palette et d'éviter l'utilisation d'une autre palette portant le même nom dans un autre package placé avant celui désiré dans le répertoire de recherche, s'il y a lieu.

Par exemple, voici comment appeler la palette de couleur `default_ucscgb` du package `ggsci` :

```
paletteer_d("ggsci::default_ucscgb")
```

```
<colors>
#FF0000FF #FF9900FF #FFCC00FF #00FF00FF #6699FFFF #CC33FFFF #99991EFF #999999FF #FF00CCFF #CC0000FF #FF
```

La sortie ne contient pas les couleurs dans le document HTML, mais seulement les codes de couleurs.

Voici l'allure illustrée sous forme d'une bande de cette palette.

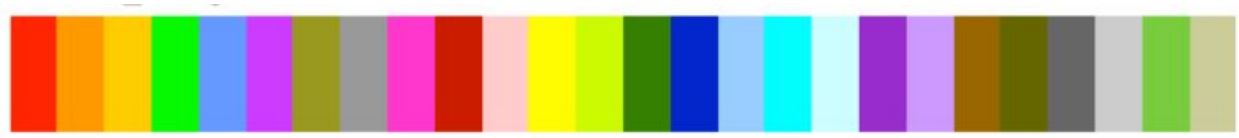


FIGURE 1 – Hvitfeldt, 2018

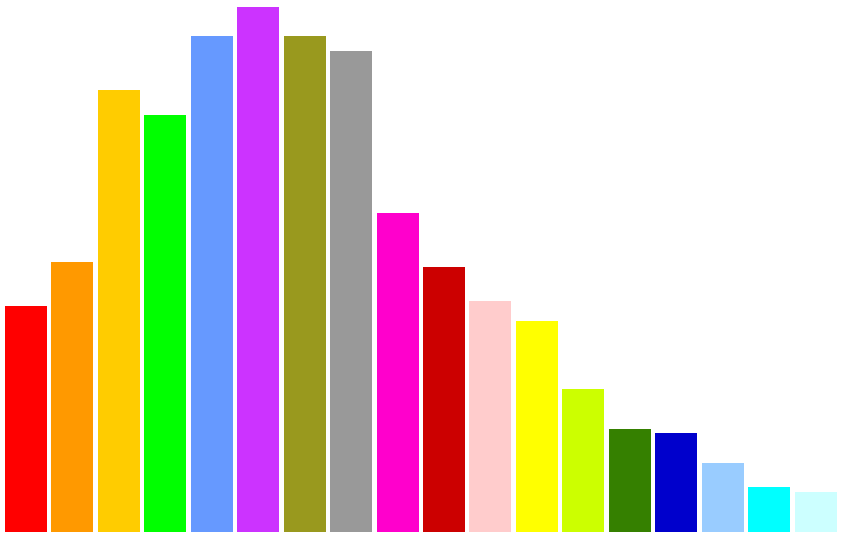
Pour une palette discrète à largeur fixe contenant beaucoup de couleurs, il est possible de savoir le nombre de couleurs qu'elle contient avec la fonction `length` comme suit avec comme exemple la palette `default_ucscgb` du package `ggsci` :

```
length(paletteer_d("ggsci::default_ucscgb"))
```

```
[1] 26
```

Pour illustrer l'application d'une palette discrète de largeur fixe sur un exemple, voici la palette `default_ucscgb` du package `ggsci` appliquée aux barres d'un diagramme à barres représentant les fréquences des 18 plus petites modalités de la variable `mag` du jeu de données `quakes` transformée en facteur :

```
ggplot(data = quakes) +
  geom_bar(
    mapping = aes(x = factor(mag), fill = factor(mag)),
    show.legend = FALSE
  ) +
  theme_solid() +
  scale_fill_paletteer_d("ggsci::default_ucscgb") +
  scale_x_discrete(limits = seq(from = 4,to = 5.7, by = 0.1))
```



Notez que dans la commande de ce graphique, l'appel à la palette `default_ucscgb` se fait par la fonction `scale_fill_paletteer_d()`. Le thème `solid` de `ggthemes` a également été utilisé pour mettre seulement

en valeur les bandes de couleur.

## 3.3 Les palettes discrètes dynamiques

Les palettes discrètes de type dynamique permettent de s'adapter au nombre de couleur voulu (Hvitfeldt, 2020a). Pour y arriver, un dégradé de teintes est effectué pour couvrir de façon distincte chaque élément du graphique.

Comme pour les palettes discrètes à largeur fixe, il est possible de visualiser une palette discrète dynamique avec la fonction suivante :

```
paletteer_dynamic("NomDuPackage::NomDeLaPalette", n = nombre_de_teintes_à_afficher)
```

Avec la palette `orange.pal` du package `cartography` et 10 dégradés de couleur, la commande va comme suit :

```
paletteer_dynamic("cartography::orange.pal", n = 10)
```

```
<colors>
#FDE687FF #FDD474FF #FDC362FF #FDB250FF #FEA03CFF #FE8924FF #FE720CFF #FC5200FF #F72900FF #F20000FF
```

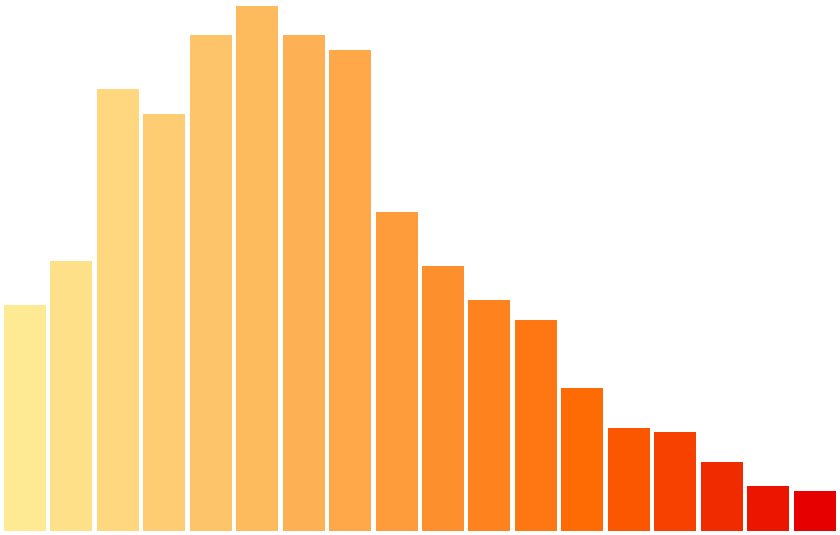
Encore une fois, seuls les codes de couleurs apparaissent au sein de la page HTML. Voici donc l'allure illustrée sous forme d'une bande de cette palette :

FIGURE 2 – Hvitfeldt, 2018

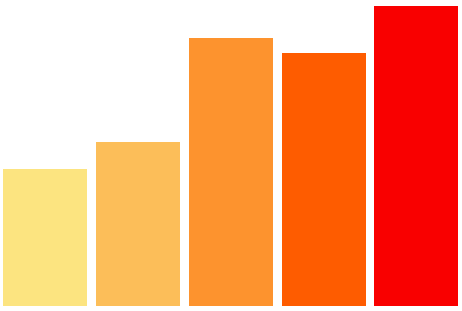
Voici maintenant un exemple de l'utilisation d'une palette discrète dynamique appliquée au graphique précédent. Un deuxième graphique est ensuite généré avec moins de bandes, soit 5 au lieu de 18. Il est alors possible d'observer que le dégradé se fait beaucoup plus rapidement.

```
palette_discrete <- ggplot(data = quakes) +
  geom_bar(
    mapping = aes(x = factor(mag), fill = factor(mag)),
    show.legend = FALSE
  ) +
  theme_solid() +
  scale_fill_paletteer_d("cartography::orange.pal", dynamic = TRUE)
```

```
palette_discrete +
  scale_x_discrete(limits = seq(from = 4,to = 5.7, by = 0.1))
```



```
palette_discrete +
  scale_x_discrete(limits = (seq(from = 4,to = 4.4, by = 0.1)))
```



Notez également que l'appel à la palette discrète dynamique `orange.pal` se fait par l'entremise de la fonction `scale_fill_paletteer_d()` comme c'était le cas pour les palettes dynamiques à largeur fixe, mais à laquelle est ajouté l'argument `dynamic = TRUE`.

## 3.4 Les palettes continues

Les palettes continues offrent un intervalle de couleur sans séparation franche entre chacune d'elles. Elles permettent ainsi un dégradé complet entre une ou plusieurs couleurs (Hvitfeldt, 2020a).

Ces palettes peuvent être appelées de façon similaire à celle des palettes discrètes, mais en remplaçant le *d* suivant `paletteer_` par un *c*. Le nombre de couleurs à afficher est également requis.

```
paletteer_c("NomDuPackage::NomDeLaPalette", n = nombre_de_couleurs_à_afficher)
```

Voici un appel à la palette `rainbow` du package `grDevices` avec 20 couleurs.

```r
paletteer_c("grDevices::rainbow", n = 20)
```

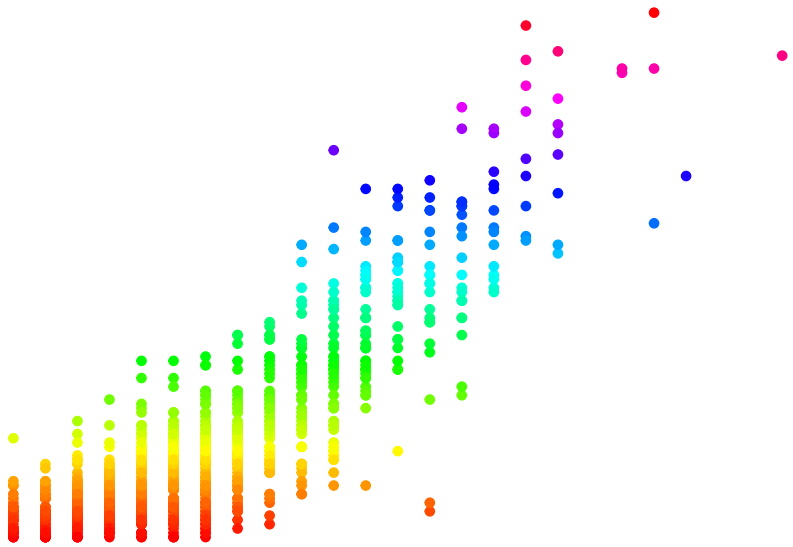
```
<colors>
#FF0000FF #FF4D00FF #FF9900FF #FFE500FF #CCFF00FF #80FF00FF #33FF00FF #00FF19FF #00FF66FF #00FFB2FF #00
```

Sous la forme d'une bande, cette palette ressemble à ceci :

FIGURE 3 – Hvitfeldt, 2018

À titre d'exemple appliqué, voici un graphique bâti à partir des données du package `quakes` en utilisant la fonction `geom_point` pour produire un diagramme de dispersion croisant deux variables continues. La palette continue `rainbow` du package `grDevices` est utilisée pour colorer les points selon le niveau de la variable en ordonnée.

```r
ggplot(data = quakes) +
  geom_point(
    mapping = aes(x = mag, y = stations, colour = stations),
    size = 2,
    show.legend = FALSE
  ) +
  theme_solid() +
  scale_colour_paletteer_c("grDevices::rainbow")
```



Notez que l'ajout de la palette continue `rainbow` a été effectué avec la fonction `scale_colour_paletteer_c()`.

### 3.5 Nos impressions du package `paletteer`

Le package `paletteer` est un outil fort intéressant d'autant plus qu'il permet de sauver du temps de recherche. En effet, au lieu de devoir charger plusieurs packages et chercher au travers de la plateforme R les palettes de couleurs qui conviendraient le mieux au graphique désiré, la fonction `paletteer` les regroupe toutes sous un même package. Les différentes palettes sont également illustrées sous forme de bandes sur le site Github. Le seul hic sur ce site est que les palettes ne sont pas identifiées comme étant discrètes à largeur fixe, dicrètes dynamiques ou continues, laissant au lecteur deviner de quel genre il s'agit.

Un élément problématique rencontré lors de l'utilisation du package `paletteer` est la visualisation de la palette dans R. En effet, lorsqu'une palette est appelée, elle ressort en petits carrés contenant le code de la couleur. Il est donc moins évident de voir l'effet visuel de la palette et les transitions de couleurs, comparé aux bandes disponibles sur le site *Github*. De plus, le HTML généré ne contient pas les couleurs des palettes appelées, mais seulement les codes de couleurs.

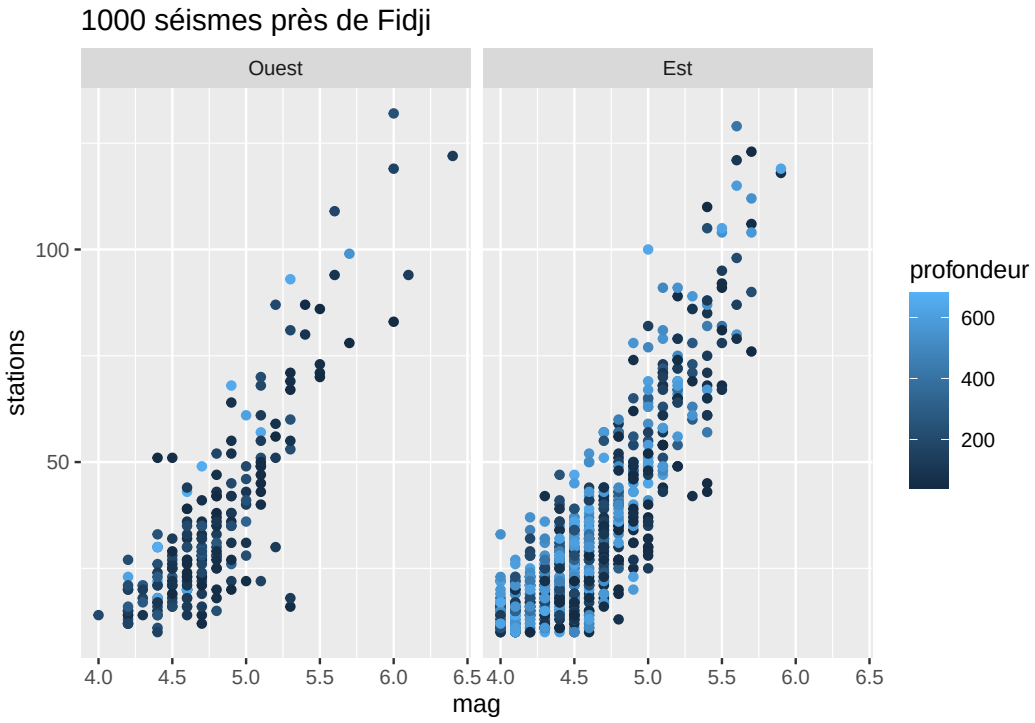
Dans une autre perspective, il serait intéressant que l'attribution d'une palette discrète à largeur fixe à un nombre de données supérieur à la largeur de la palette entraine le *recyclage* de ladite palette. De ce fait, les couleurs s'affichant sur les données supplémentaires pourraient être attribuées selon le recommencement de l'ordre de couleur de la palette, plutôt que d'afficher une absence de couleur sur ces données.

Néanmoins, le package `paletteer`, une fois la bonne palette déterminée, permet facilement d'appliquer le jeu de couleur voulu au sein du graphique généré.

## 4 Exemple combinant les packages `ggthemes` et `paletteer`

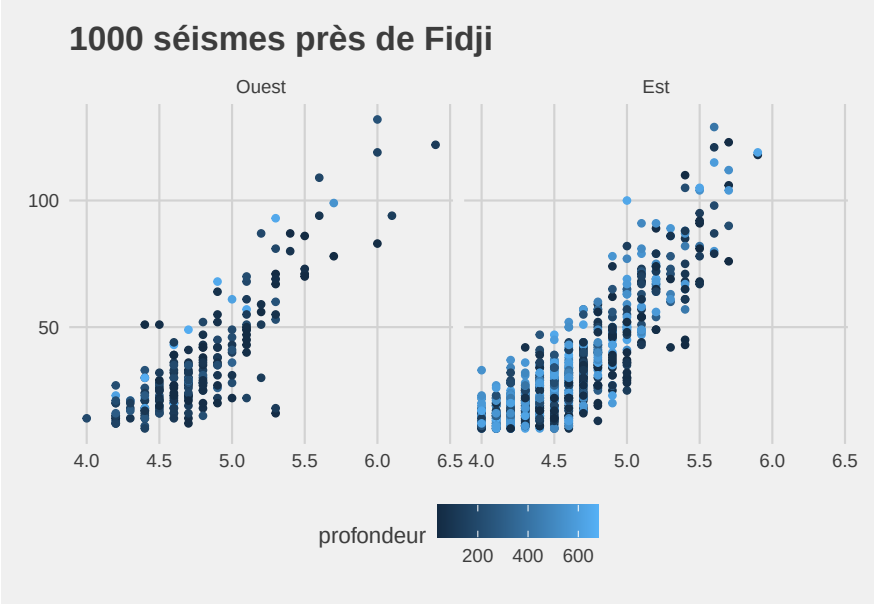
L'exemple suivant permet d'intégrer des notions propres aux packages `ggthemes` et `paletteer`. Tout d'abord, un graphique de base met en relation les nombres de stations les ayant détectés et la magnitude des séismes du jeu de données `quakes`. Un facteur nommé `region` est ajoutée aux données pour catégoriser les séismes selon leur longitude.

```
quakes$region <- factor(quakes$long >= 175, labels = c("Ouest", "Est"))

graph_ex <- ggplot(data = quakes) +
  geom_point(mapping = aes(x = mag, y = stations, colour = depth)) +
  labs(title = "1000 séismes près de Fidji") +
  labs(colour = "profondeur") +
  facet_wrap(~ region)

graph_ex
```



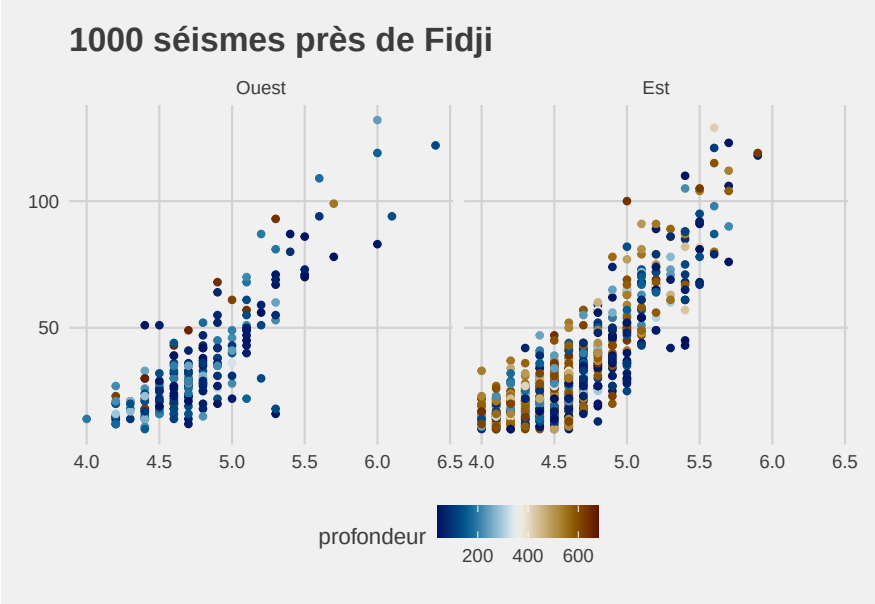

Le thème de graphique peut ensuite être modifié. Le thème `fivethirtyeight` sera appliqué ici.

```
graph_ex2 <- graph_ex +
  theme_fivethirtyeight()
graph_ex2
```




Pour le besoin de l'exemple, la palette de couleur `vik` du package `scico` sera utilisée à l'étape ci-bas.

```
graph_ex2 +
  scale_color_paletteer_c("scico::vik")
```

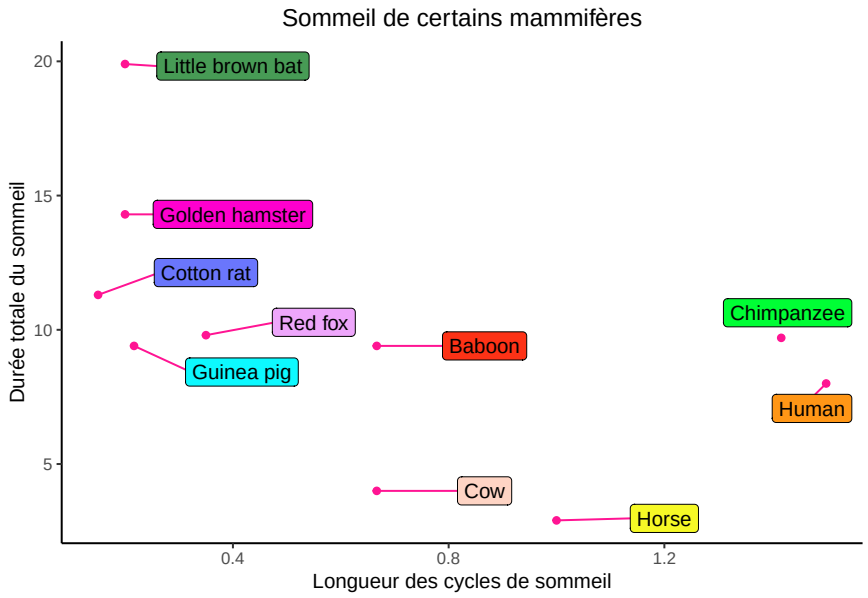



Comme il est possible de remarquer, la combinaison d'un thème et d'une palette est facilement exécutable avec `ggthemes` et `paletteer`.

# 5 Exemple combinant les trois packages présentés

Dans cette section, les packages `ggrepel`, `ggthemes` et `paletteer` seront utilisés afin de créer un graphique. Pour ce faire, le jeu de données `msleep` du package `ggplot2` sera utilisé et tronqué sélectivement afin de ne pas surcharger le graphique.

```
ggplot(data = selection) +
  geom_point(
    mapping = aes(x = sleep_cycle, y = sleep_total),
    color = "deeppink"
  ) +
  labs(title = "Sommeil de certains mammifères") +
  labs(x = "Longueur des cycles de sommeil") +
  labs(y = "Durée totale du sommeil") +
  geom_label_repel(
    aes(x = sleep_cycle, y = sleep_total, label = name, fill = factor(selection$name)),
    show.legend = FALSE,
    box.padding = unit(0.40, "lines"),
    direction = "y",
    nudge_x = 0.2,
    segment.colour = "deeppink"
  ) +
  theme_classic() +
  scale_fill_paletteer_d("Polychrome::light") +
  theme(plot.title = element_text(hjust = 0.5))
```




L'enchaînement du package `ggrepel` et de ses arguments avec les packages `ggthemes` et `paletteer` donne une allure intéressante au graphique, tout en se faisant avec peu de fonctions attributs à la fonction `ggplot`.

# 6 Références


Arnold, J. B. 2016. Introduction to ggthemes. https://mran.microsoft.com/snapshot/2017-02-04/web/packages/ggthemes/vignettes/ggthemes.html (page consultée le 3 mars 2020).

Arnold, J. B. 2020a. jrnold/ggthemes. https://github.com/jrnold/ggthemes (page consultée le 3 mars 2020).

Arnold, J. B. 2020b. bank_slopes function. https://www.rdocumentation.org/packages/ggthemes/versions/3.5.0/topics/bank_slopes (page consultée le 3 mars 2020).

Emaasit, D. 2016. ggplot2 extensions : ggrepel. https://www.ggplot2-exts.org/ggrepel.html (page consultée le 4 mars 2020).

Hvitfeldt, E. 2018. EmilHvitfeldt/r-color-palettes. GitHub. https://github.com/EmilHvitfeldt/r-color-palettes (page consultée le 5 mars 2020 ).

Hvitfeldt, E. 2020a. paletteer/paletteer.Rproj at master. https://github.com/EmilHvitfeldt/paletteer/blob/master/paletteer.Rproj (page consultée le 5 mars 2020).

Hvitfeldt, E. 2020b. Paletteer. https://cran.r-project.org/web/packages/paletteer/readme/README.html (page consultée le 5 mars 2020).

Slowikowski, K. 2016. ggrepel Usage Examples. https://mran.microsoft.com/snapshot/2017-08-20/web/packages/ggrepel/vignettes/ggrepel.html (page consultée le 4 mars 2020).

Slowikowski, K. 2019. ggrepel examples. https://cran.r-project.org/web/packages/ggrepel/vignettes/ggrepel.html (page consultée le 4 mars 2020).

Slowikowski, K. 2020. geom_label_repel function | R Documentation. https://www.rdocumentation.org/packages/ggrepel/versions/0.8.1/topics/geom_label_repel (page consultée le 4 mars 2020).

Slowikowski, K., A. Schep, S. Hughes, S. Lukauskas, J.-O. Irisson, Z. N. Kamvar, T. Ryan, D. Christophe, Y. Hiroaki, and P. Gramme. 2019. ggrepel : Automatically Position Non-Overlapping Text Labels with "ggplot2." https://CRAN.R-project.org/package=ggrepel (page consultée le 4 mars 2020).

Yutani, H. package :ggthemes. All Your Figure Are Belong To Us. https://yutannihilation.github.io/allYourFigureAreBelongToUs/ggthemes/ (page consultée le 5 mars 2020).