Visualisation de données avec R

Antoine Neuraz

2019-06-05

Contents

Cours introductif à la visualisation de données avec R. Ce cours a pour but d introduire les concepts théoriques de base en visualisation ainsi que des exemples concrets.

Prerequis

Pour les parties théoriques, aucun prérequis n'est nécessaire. Les exemples pratiques sont conçus avec le logiciel R et le package `ggplot2`

Installer R

Vous devez avoir installé le logiciel R pour pouvoir suivre les exemples pratiques. Vous trouverez les liens de téléchargement ici : https://cran.r-project.org

Installer RStudio (facultatif)

Nous conseillons également d'installer l'interface de développement (IDE) RStudio qui vous facilitera les choses pour la prévisualisation des contenus et des rendus. Bien entendu, si vous avez déjà une IDE préférée, vous pouvez continuer à l'utiliser. Vous trouverez la RStudio Desktop en version open source (gratuite) ici : https://www.rstudio.com/products/rstudio/#Desktop

Installer les packages nécessaires

Dans ce cours, un certain nombre de packages sont utilisés très fréquement et doivent être installés :

```
pkg_list_req = c("tidyverse",
                 "ggplot2",
                 "see")
```

```
install.packages(pkg_list_req)
```

Un certain nombre d'autres packages, utilisés plus ponctuellement vous seront indiqués dans les différents chapitres. Voici une liste exhaustive des packages utilisés dans ce cours :

tidyverse, ggplot2, see

Chapter 1

Intro

1.1 Mappings in ggplot

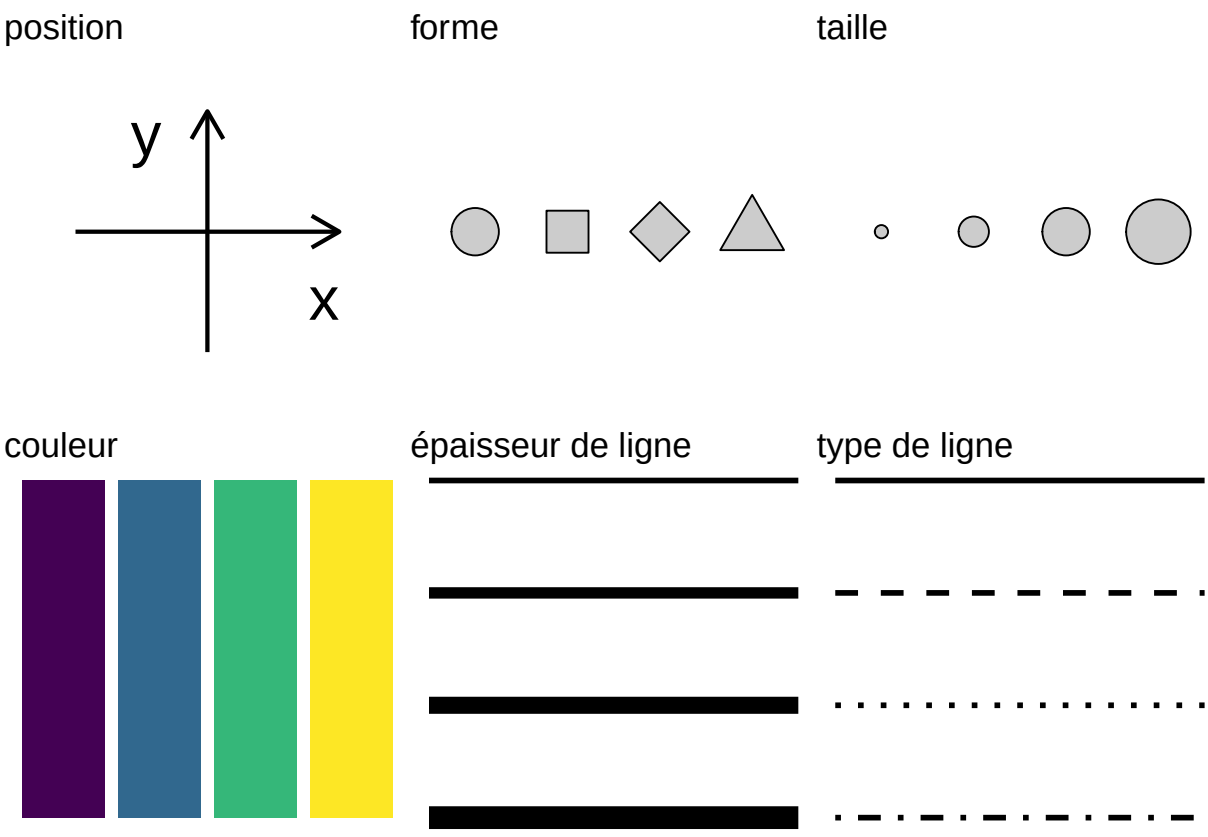


figure from https://serialmentor.com/dataviz/aesthetic-mapping.html

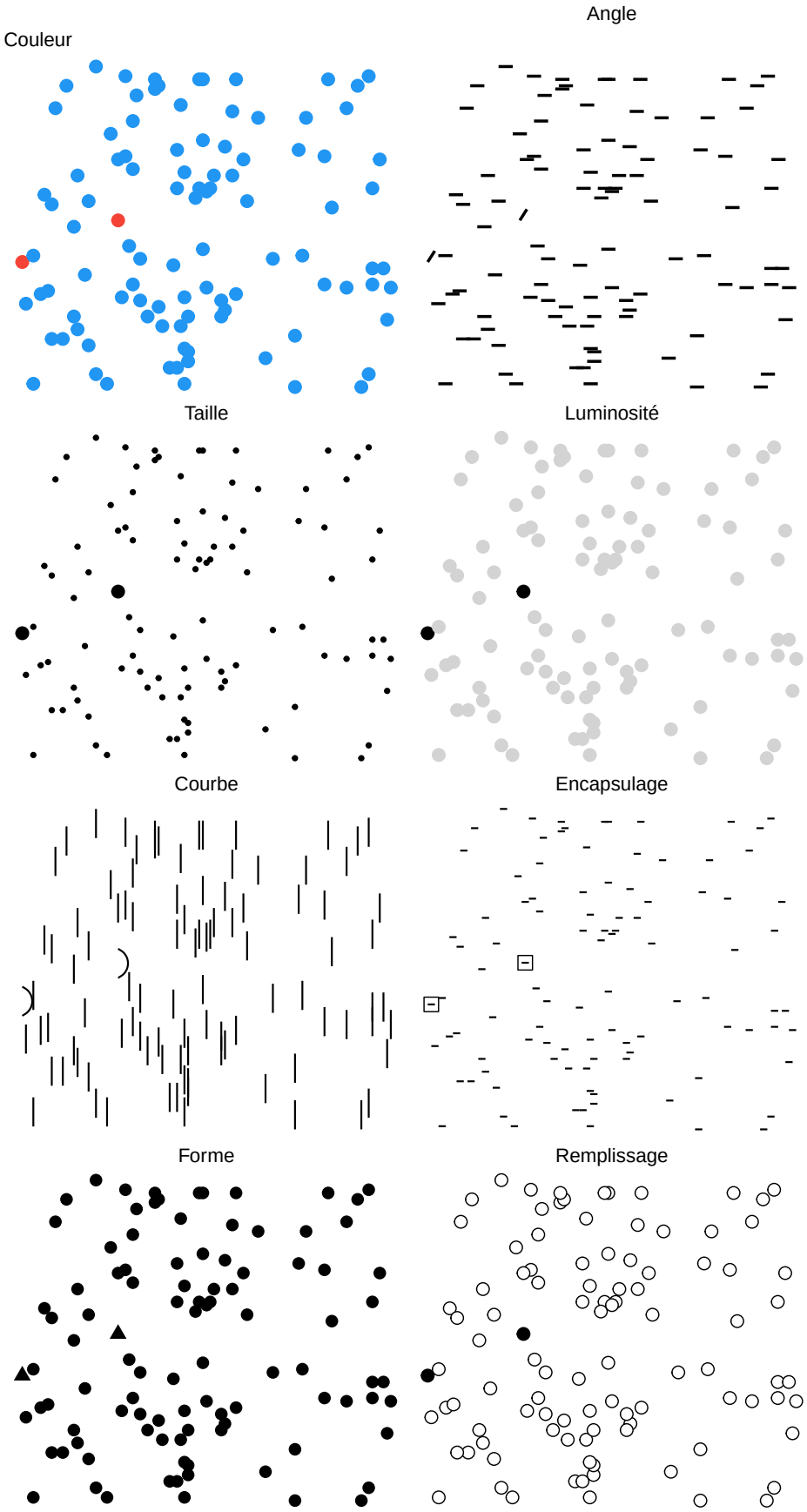


Figure 1.1: Exemples d'encodage

1.2 scales in ggplot

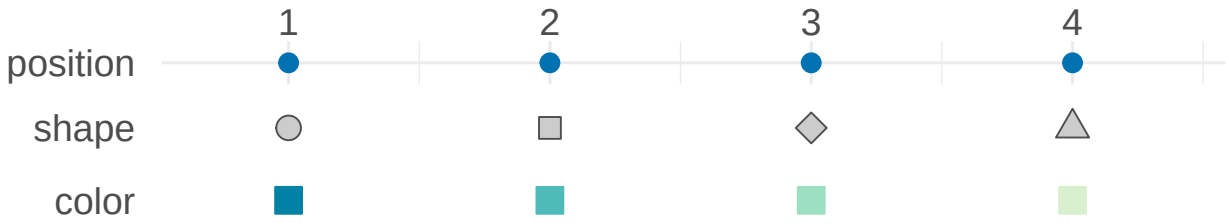
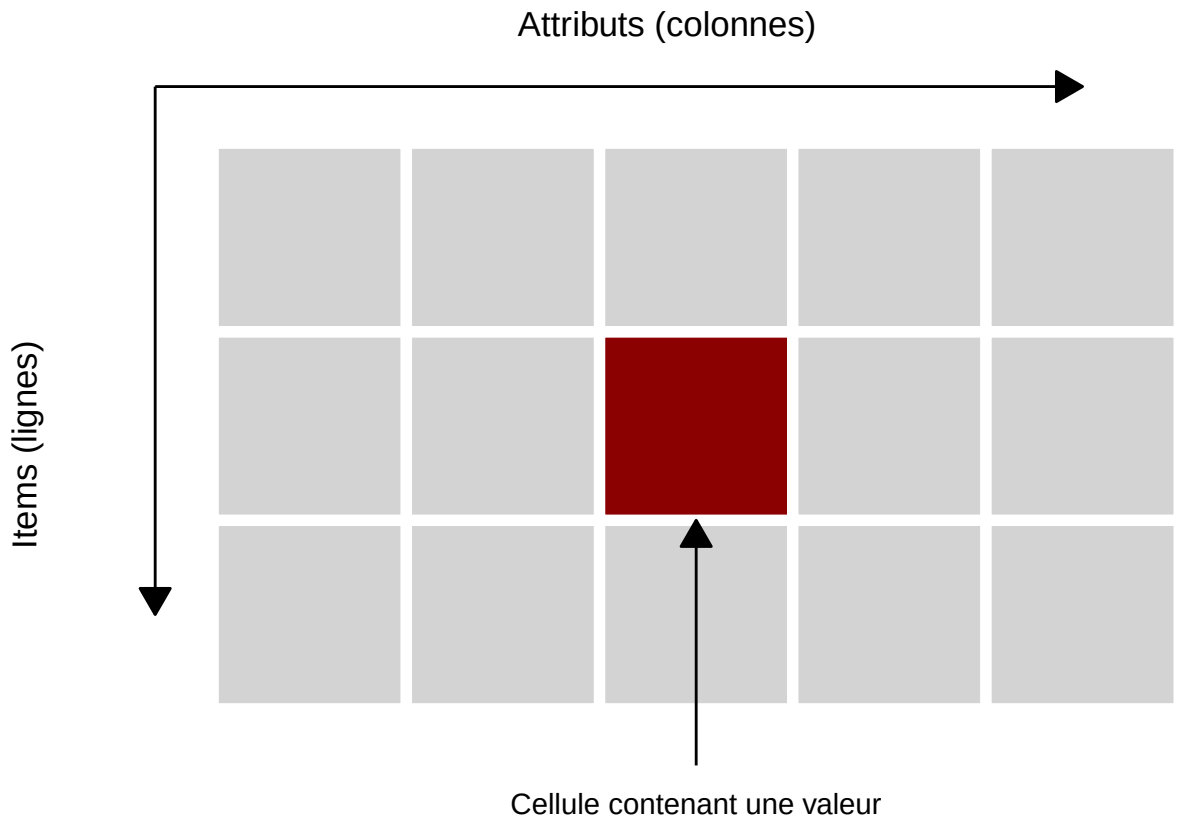


figure from https://serialmentor.com/dataviz/aesthetic-mapping.html



Chapter 2

ggplot2 : techniques avancées

2.1 Créer son propre thème ggplot2

Les thèmes de ggplot2 permettent de contrôler l'apparence des plots. Il est possible de modifier un thème standard en utilisant la fonction `theme()`. Mais nous allons voir ici comment créer un thème personalisé.

Il est bien entendu possible de créer un thème de toutes pièces. Pour cela, il faut définir un à un tous les éléments possibles du thème mais c'est très long et rébarbatif. Dans ggplot2, le seul thème défini de cette façon est le thème de base `theme_grey()` (voir le repo officiel). Les autres thèmes héritent les attribut de ce premier thème et modifient uniquement éléments nécéssaires. Par exemple, `theme_bw()` est construit à partir de `theme_grey()` et `theme_minimal()` se base sur `theme_bw()`. C'est beaucoup plus pratique de définir les thèmes de cette façon.

2.1.1 un thème est une fonction

Un thème est une fonction R classique qui prend comme arguments 4 variables : - base_size : taille de base du texte (défaut = 11) - base_family : famille de polices de base (défaut = "") - base_line_size : taille de base des éléments `line` (défaut = base_size / 22) - base_rect_size : taille de base des éléments `rect` (default = base_size / 22)

```
my_theme <- function(base_size = 11,
                     base_family = "",
                     base_line_size = base_size / 22,
                     base_rect_size = base_size / 22) {}
```

2.1.2 modifier un thème de base avec %+replace%

Ensuite, nous allons choisir un thème de base duquel notre thème personalisé va hériter les éléments par défaut. En effet, tous les éléments que nous ne spécifieront pas seront basés sur le thème de base. Par exemple, nous pouvons choisir `theme_minimal()`.

Pour modifier les éléments du thème de base, il faut utiliser l'opérateur `%+replace%` suivi de la fonction `theme()`. C'est dans cette dernière que nous pourrons spécifier les différents éléments à modifier par rapport au thème de base.

```
my_theme <- function(base_size = 11,
                     base_family = "",
                     base_line_size = base_size / 22,
```

```
                    base_rect_size = base_size / 22) {

  theme_minimal(base_size = base_size,
                base_family = base_family,
                base_line_size = base_line_size,
                base_rect_size = base_rect_size) %+replace%
    theme(
      # éléments à modifier
    )
}
```

2.1.3 définir de nouveaux attributs

Nous pouvons a présent inserer dans la fonction thème les éléments à modifier. Notez qu'il ne faut pas utiliser de tailles absolues mais définir des tailles relatives avec la fonction `rel()`.

`my_theme()`

Voici un exemple de thème personnalisé (très fortement inspiré du thème `theme_modern()` du package `see`) basé sur `theme_minimal()` :

```
my_theme <- function(base_size = 11,
                     base_family = "",
                     base_line_size = base_size / 22,
                     base_rect_size = base_size / 22) {

  half_line <- base_size/2

  theme_minimal(base_size = base_size,
                base_family = base_family,
                base_line_size = base_line_size,
                base_rect_size = base_rect_size) %+replace%
    theme(
      ## Panel grid ##
      panel.border = element_blank(),
      panel.grid.major = element_blank(),
      panel.grid.minor = element_blank(),
      ## Title ##
      plot.title = element_text(size = rel(1.3),
            face = "plain", margin = margin(0, 0, 20, 0)),
      ## Axis ##
      axis.line = element_line(colour = "black", size = rel(0.5)),
      axis.title.y = element_text(margin = margin(t = 0, r = rel(20), b = 0, l = 0),
                                  angle = 90),
      axis.title.x = element_text(margin = margin(t = rel(20), r = 0, b = 0, l = 0)),
      axis.title = element_text(size = rel(1.2),
            face = "plain"),
      axis.text = element_text(size = rel(.8)),
      axis.ticks = element_blank(),
      ## Legend ##
      legend.key = element_blank(),
      legend.position = "bottom",
```

```
      legend.text = element_text(size = rel(1.1)),
      legend.title = element_text(size = rel(1.1)),
      legend.spacing.x = unit(2, "pt"),
      ## Background ##
      strip.background = element_blank(),
      plot.tag = element_text(size = rel(1.3), face = "bold"),
      strip.text = element_text(face = "bold")
    )
}
```

`my_theme_dark()`

Et un thème sombre basé sur `my_theme()` et très fortement inspiré de `theme_blackboard()` du package `see` :

```
my_theme_dark <-function(base_size = 11,
                     base_family = "",
                     base_line_size = base_size / 22,
                     base_rect_size = base_size / 22) {

  dark_color = "#0d0d0d"
  light_color = "#E0E0E0"

  my_theme(base_size = base_size,
              base_family = base_family,
              base_line_size = base_line_size,
              base_rect_size = base_rect_size) %+replace%

    theme(
          ## Backgrounds ##
          plot.background = element_rect(fill = dark_color),
          panel.background = element_rect(fill = dark_color, color=dark_color),
          legend.background = element_rect(fill = dark_color, color=dark_color),
          ## Lines ##
          axis.line = element_line(color = light_color),
          ## Text ##
          text = element_text(family = base_family, face = "plain",
                              color = light_color, size = base_size,
                              hjust = 0.5, vjust = 0.5, angle = 0,
                              lineheight =0.9, margin = margin(),
                              debug = FALSE),
          axis.text = element_text(color = light_color)
        )
}
```

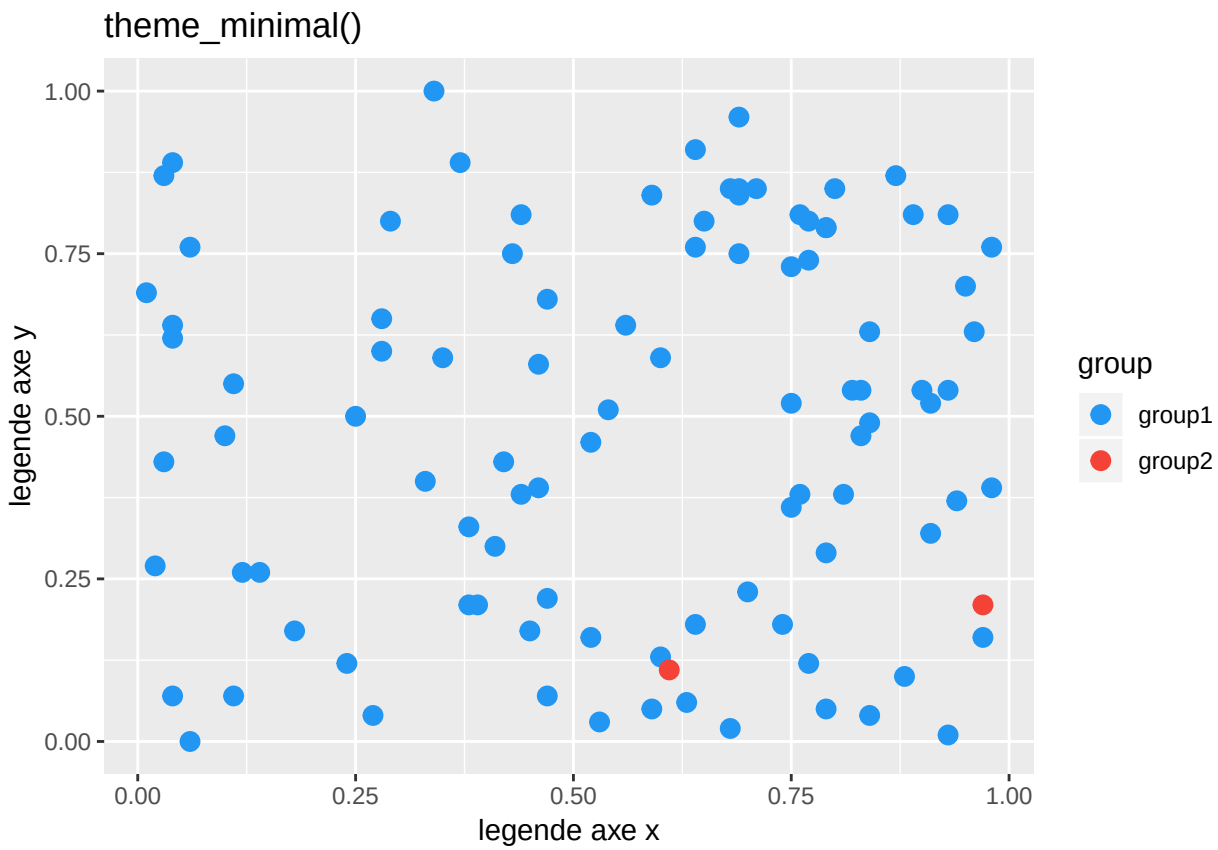
2.1.4 Exemple

Créons un plot d'exemple à partir d'un jeu de données synthétique

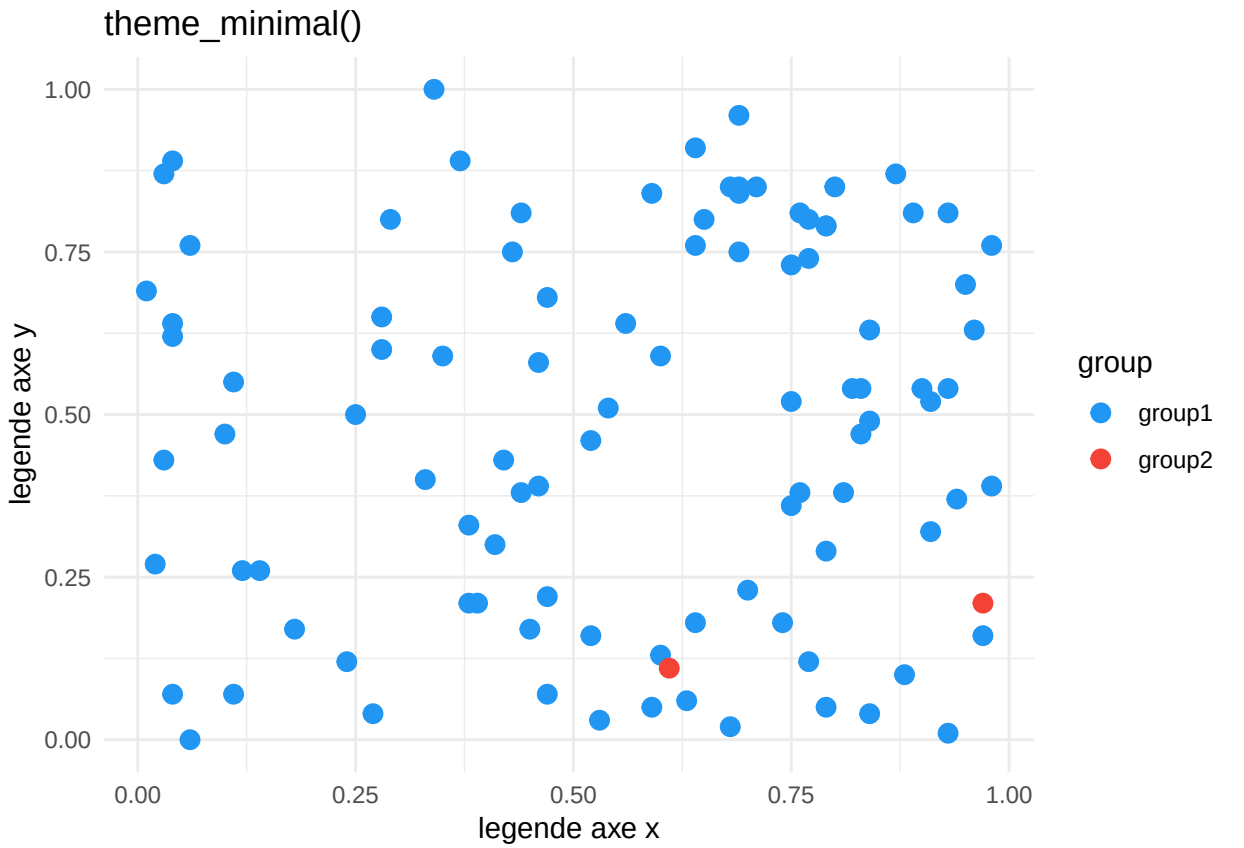
```
p <- generate_uniform_dataset(
  dataset_size = 100,
  min_x = 0, max_x = 1,
```

```
  outliers = 2, seed = 506
) %>%
  ggplot(data = ., aes( x = x, y = y, color = group)) +
  geom_point(size = 3) +
  scale_color_material_d() +
  labs(title= "theme_minimal()",
       x="legende axe x",
       y="legende axe y")
```

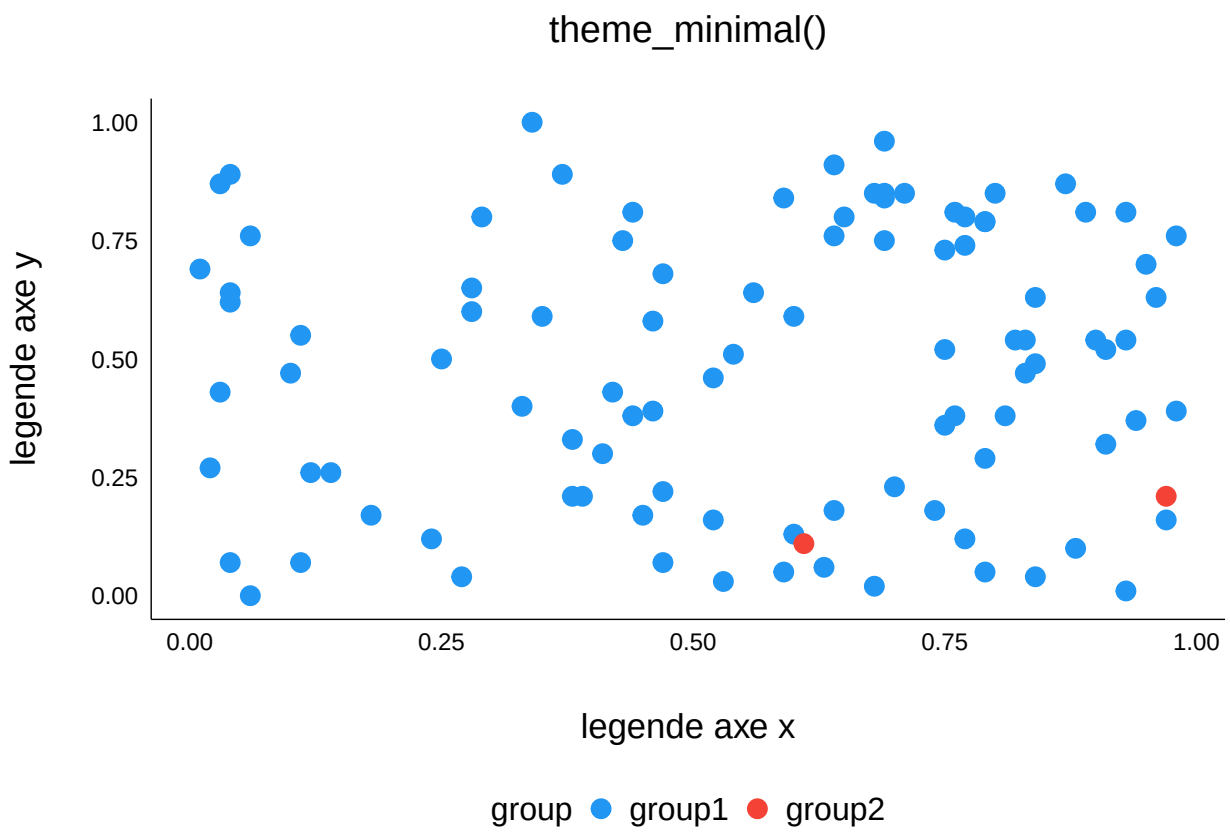
```
p
```



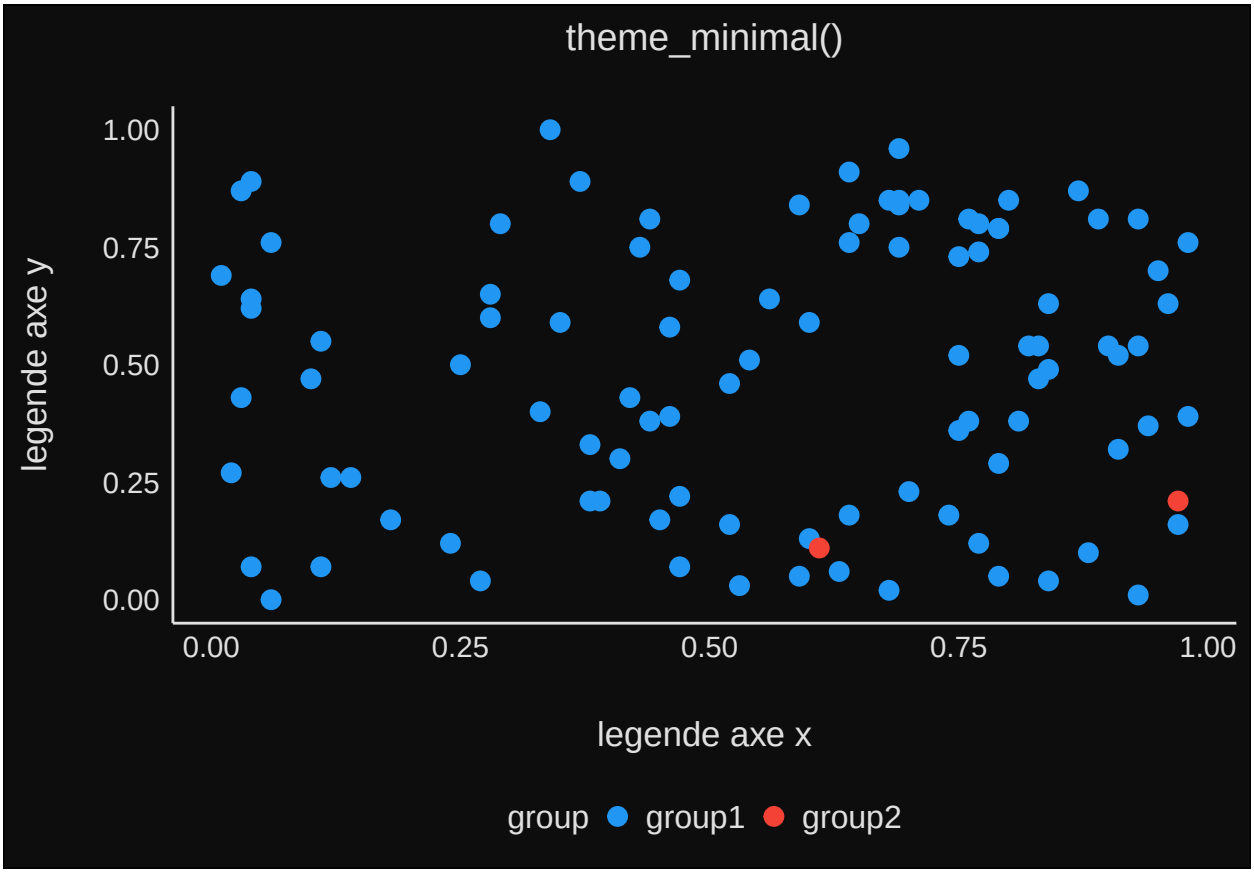

```
p + theme_minimal()
```


theme_minimal()
1.00
0.75
0.50
0.25
0.00
legende axe y
0.00
0.25
0.50
0.75
1.00
legende axe x
group
group1
group2

```
p + my_theme()
```




```
p + my_theme_dark()
```


theme_minimal()
1.00
0.75
0.50
0.25
0.00
legende axe y
0.00
0.25
0.50
0.75
1.00
legende axe x
group
group1
group2