

R : Introduction à R — Analyse R | Cours 7

Ményssa Cherifa-Luron

2023-09-15

Contents

<i>Introduction</i>	1
Pourquoi la Visualisation de Données?	2
Qu'est-ce que <code>ggplot2</code> ?	2
Objectifs du Cours	2
Organisation	2
Ressources	2
Création de Graphiques de Base	2
barplot	4
histogrammes	5
scatterplot et jitter	6
boxplot	8
lines	9
Personnalisation des graphiques : couleurs, légendes, et titres.	10
Techniques Avancées de Visualisation :	18
Utilisation de facettes pour des graphiques multi-panneaux avec les données <code>babynames</code>	18
Graphique de l'évolution du prix du bitcoin	20
Créer une carte avec <code>leaflet</code>	21
Exercices	23

Introduction

Ce cours vise à vous fournir les compétences nécessaires pour créer des visualisations de données percutantes et informatives en utilisant la bibliothèque `ggplot2` de R. Que vous soyez un analyste de données, un scientifique de données, ou simplement quelqu'un d'intéressé par la data science, ce cours vous aidera à transformer vos données en insights visuels compréhensibles.

Pourquoi la Visualisation de Données?

La visualisation de données est essentielle dans le monde d'aujourd'hui.

- **Communication Efficace:** La visualisation aide à communiquer les résultats et les insights de manière claire et efficace.
- **Exploration de Données:** Elle joue un rôle clé dans l'exploration de données, permettant de découvrir des tendances, des motifs et des anomalies.
- **Prise de Décision Basée sur les Données:** Les visualisations rendent les données compréhensibles, facilitant ainsi la prise de décisions éclairées.

Qu'est-ce que ggplot2?

ggplot2: Un Outil Puissant pour la Visualisation de Données

- **Partie de l'écosystème R:** ggplot2 est une bibliothèque R largement utilisée pour la création de visualisations de données de haute qualité.
- **Basé sur la Grammaire de la Graphique:** ggplot2 utilise les concepts de la "grammaire de la graphique", une approche systématique pour construire des graphiques en couches.
- **Personnalisable et Flexible:** Il offre une grande flexibilité pour personnaliser les graphiques, ce qui le rend adapté pour une large gamme de scénarios.

Objectifs du Cours

À la fin de ce cours, vous serez capable de :

- Comprendre les principes fondamentaux de la visualisation de données.
- Utiliser ggplot2 pour créer une variété de graphiques.
- Personnaliser vos graphiques pour les rendre plus informatifs et visuellement attrayants.
- Appliquer les meilleures pratiques en matière de design de graphiques.

Organisation

Contenu du Cours : - Démonstrations - Exercices

Ressources

- Rbloggers : Blog Populaire sur le langage R
- Datacamp : Plateforme d'apprentissage de la data science interactive
- Big Book of R : Edition qui rassemble des livres open-source sur le langage R
- Introduction accélérée au langage R pour la data science : L'essentiel et plus de tout ce que nous verrons

Création de Graphiques de Base

Chaque graphique utilise `ggplot()` pour initialiser l'objet ggplot et spécifie la source de données (`data`) et l'aesthetic mapping (`aes`) qui définit quelles variables sont utilisées sur les axes x et y, ainsi que d'autres paramètres esthétiques tels que la couleur, la taille, etc.



Figure 1: Illustration du langage R par DALL-E

barplot

```
library(tidyverse)

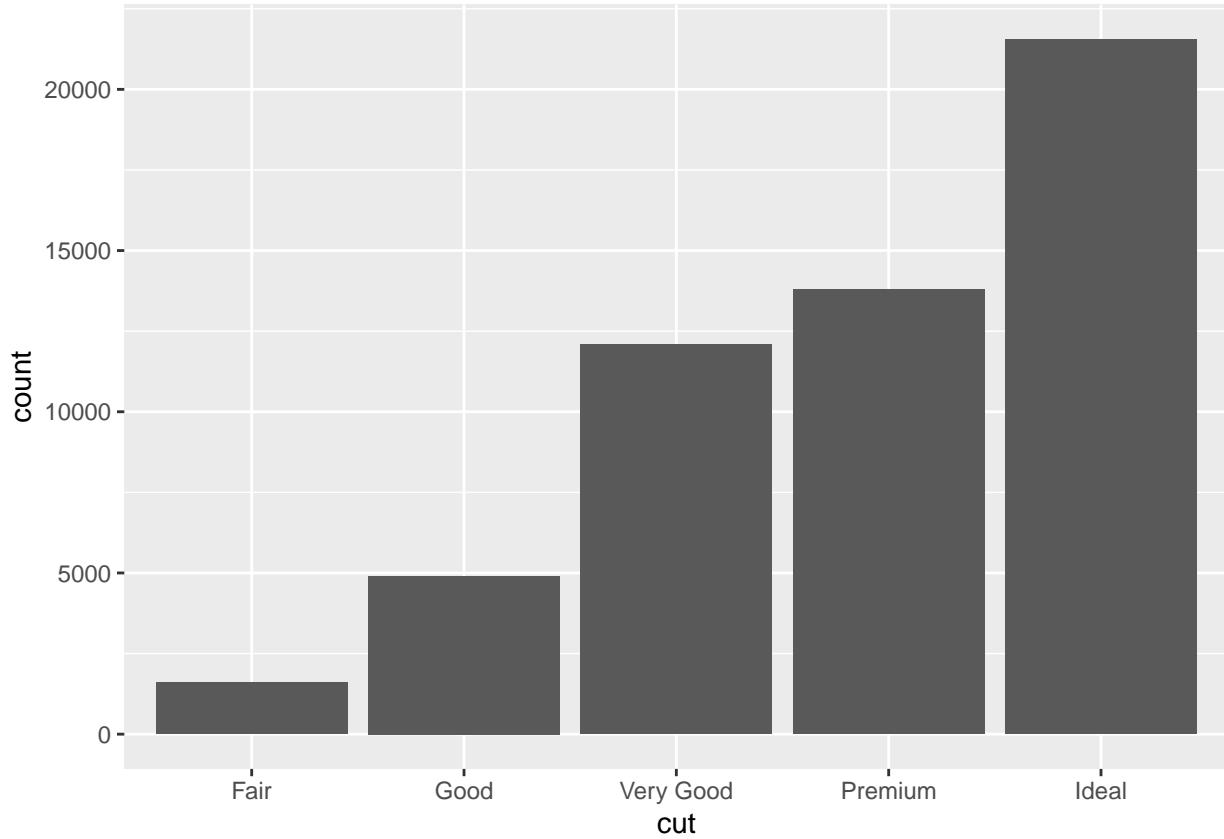
## -- Attaching core tidyverse packages ----- tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr    1.3.0
## v purrr    1.0.1

## -- Conflicts ----- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()   masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

head(diamonds)

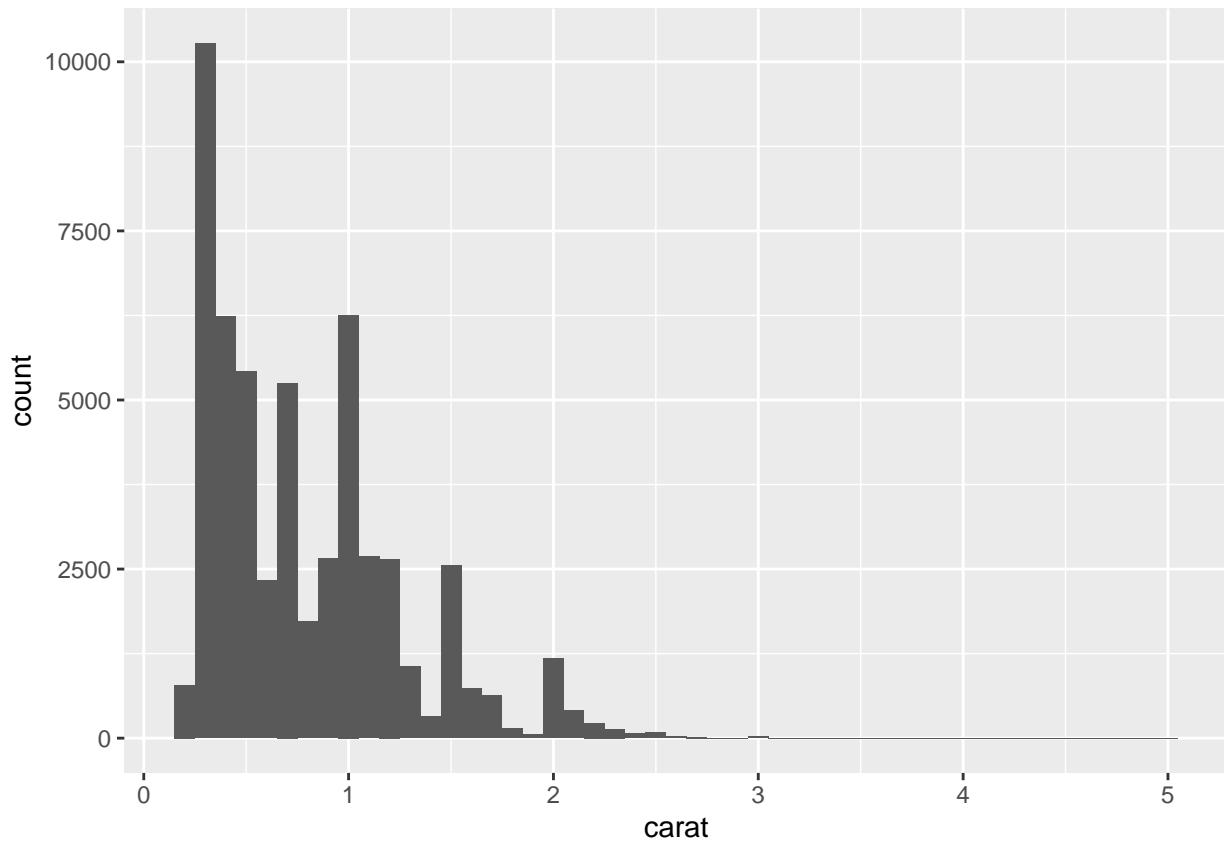
## # A tibble: 6 x 10
##   carat cut      color clarity depth table price     x     y     z
##   <dbl> <ord>    <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1 0.23 Ideal    E     SI2     61.5    55    326  3.95  3.98  2.43
## 2 0.21 Premium  E     SI1     59.8    61    326  3.89  3.84  2.31
## 3 0.23 Good     E     VS1     56.9    65    327  4.05  4.07  2.31
## 4 0.29 Premium  I     VS2     62.4    58    334  4.2   4.23  2.63
## 5 0.31 Good     J     SI2     63.3    58    335  4.34  4.35  2.75
## 6 0.24 Very Good J     VVS2    62.8    57    336  3.94  3.96  2.48

ggplot(data = diamonds, aes(x = cut)) +
  geom_bar()
```



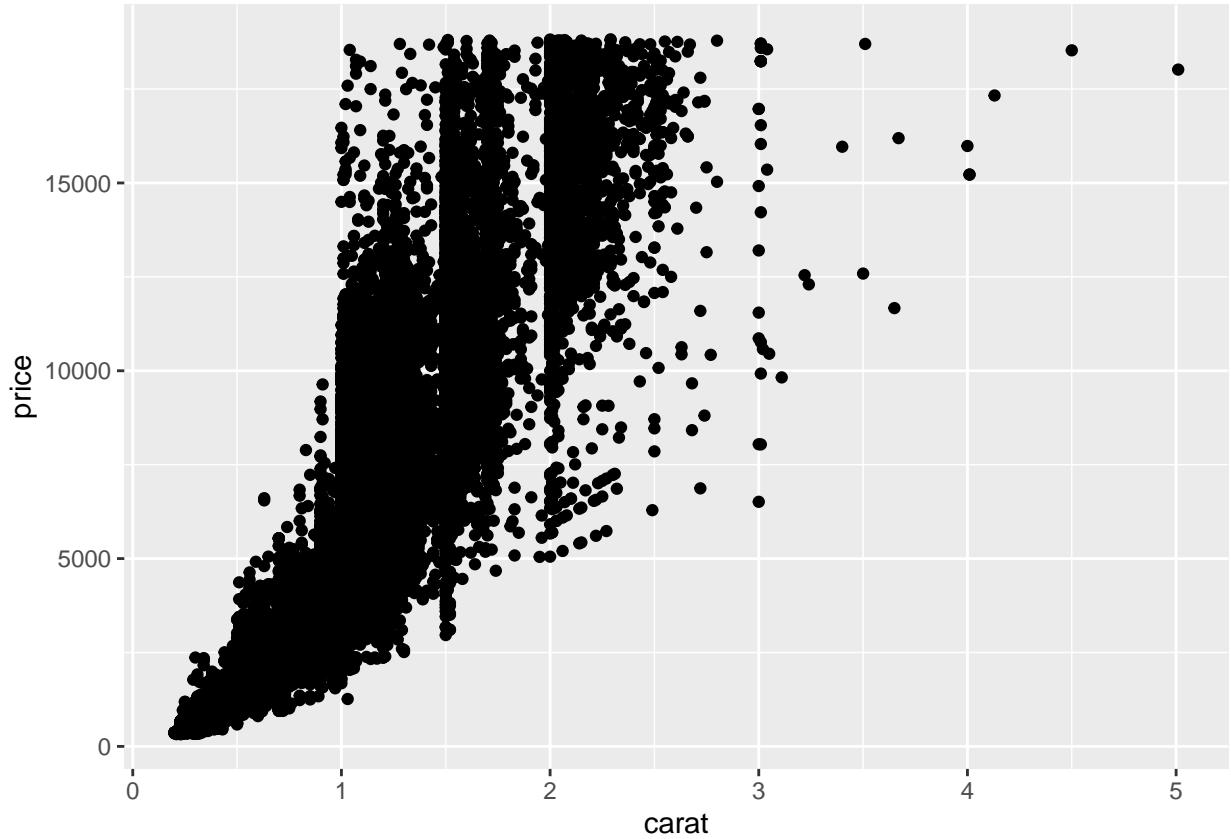
histogrammes

```
ggplot(data = diamonds, aes(x = carat)) +  
  geom_histogram(binwidth = 0.1)
```

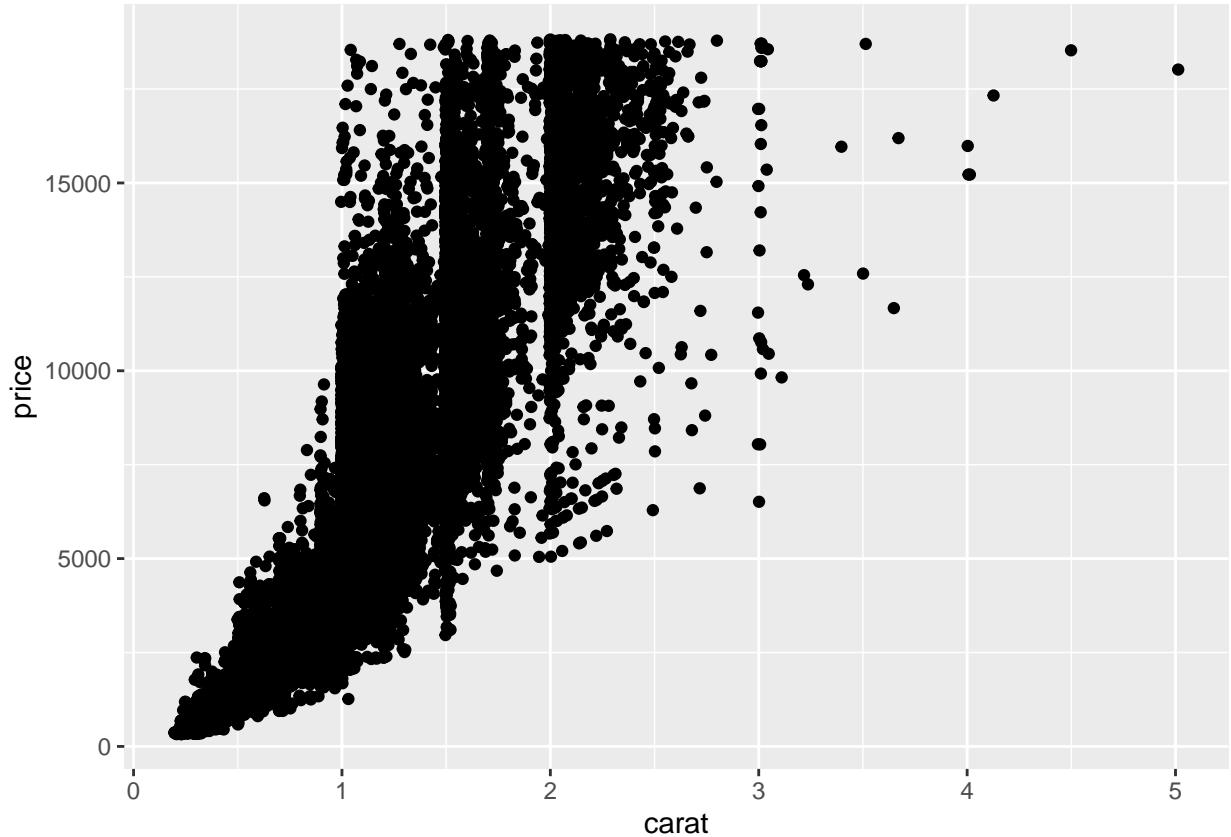


scatterplot et jitter

```
ggplot(data = diamonds, aes(x = carat, y = price)) +  
  geom_point()
```



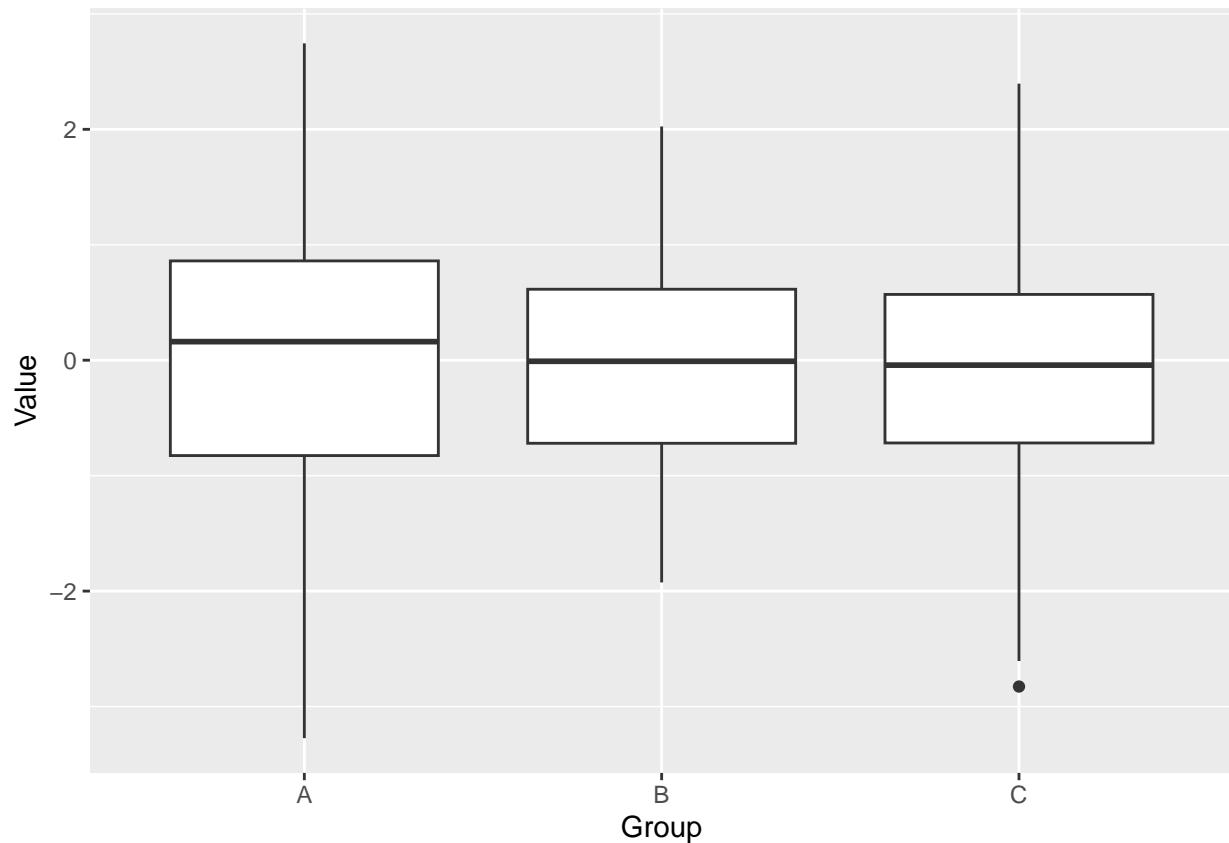
```
ggplot(data = diamonds, aes(x = carat, y = price)) +  
  geom_jitter()
```



boxplot

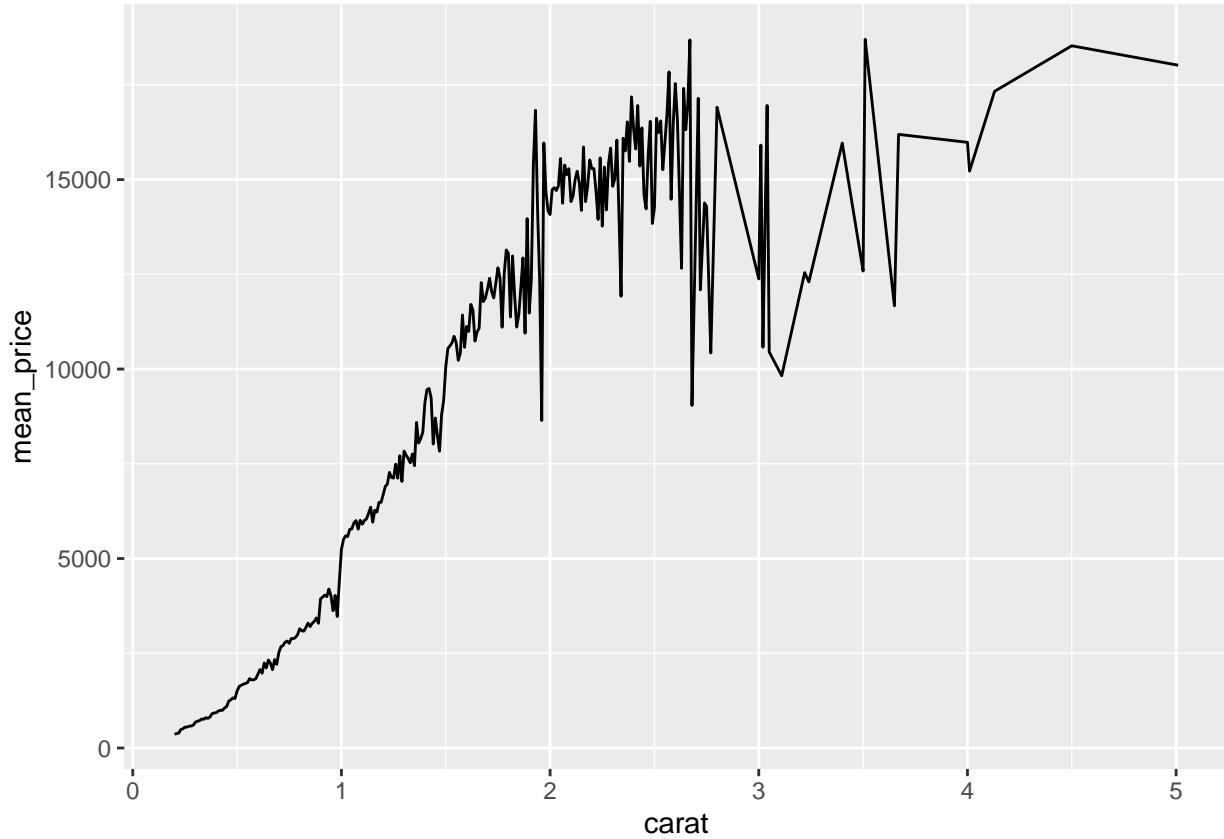
```
data <- data.frame(
  Group = rep(c("A", "B", "C"), each = 100),
  Value = rnorm(300)
)

# Créez un boxplot en utilisant ggplot2
ggplot(data, aes(x = Group, y = Value)) +
  geom_boxplot()
```



lines

```
diamonds %>%
  group_by(carat) %>%
  summarise(mean_price = mean(price)) %>%
  ggplot(aes(x = carat, y = mean_price)) +
  geom_line()
```



Personnalisation des graphiques : couleurs, légendes, et titres.

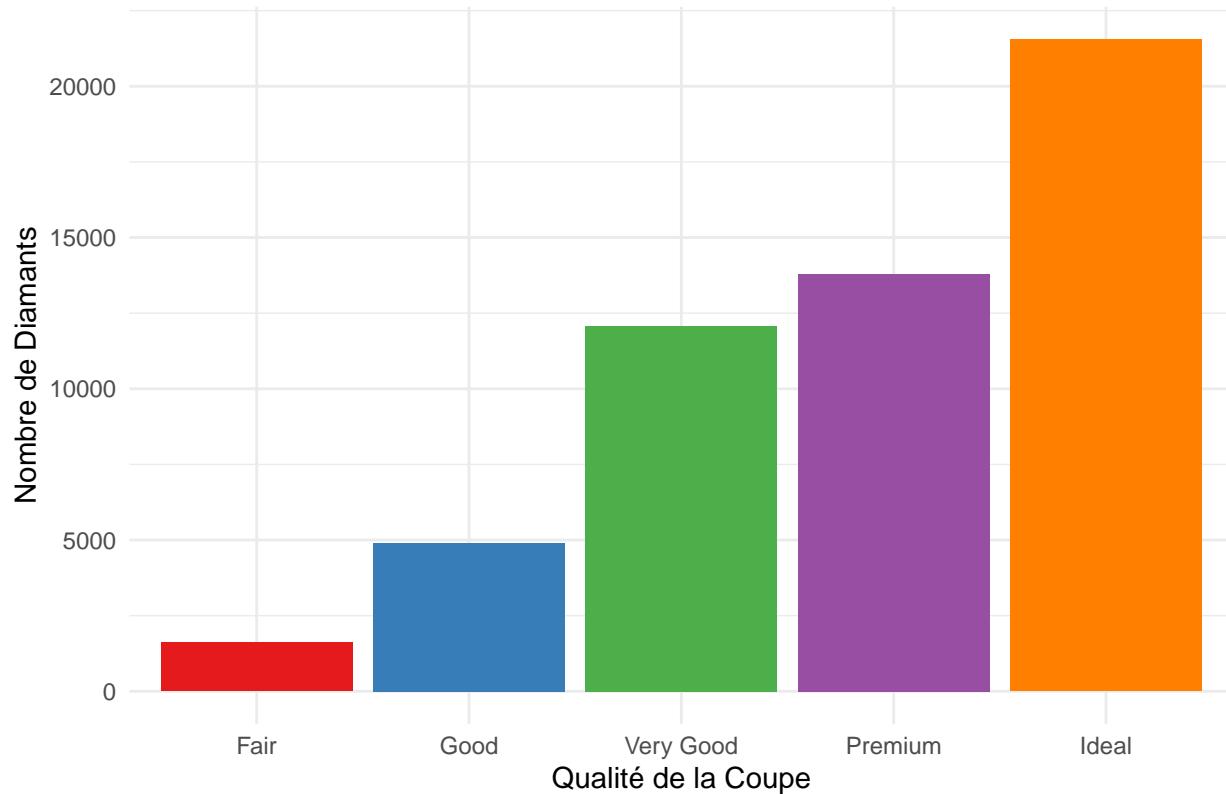
ggplot2 est un système de création de graphiques pour R, basé sur la grammaire des graphiques. Il permet de créer des graphiques complexes en assemblant des éléments de base de manière intuitive.

```
# Barplot des Diamants par Qualité de Coupe
?scale_fill_brewer()

## starting httpd help server ... done

ggplot(data = diamonds, aes(x = cut, fill = cut)) +
  geom_bar() +
  labs(title = "Distribution des Diamants par Qualité de Coupe",
       x = "Qualité de la Coupe",
       y = "Nombre de Diamants") +
  scale_fill_brewer(palette = "Set1", guide = 'none') +
  theme_minimal()
```

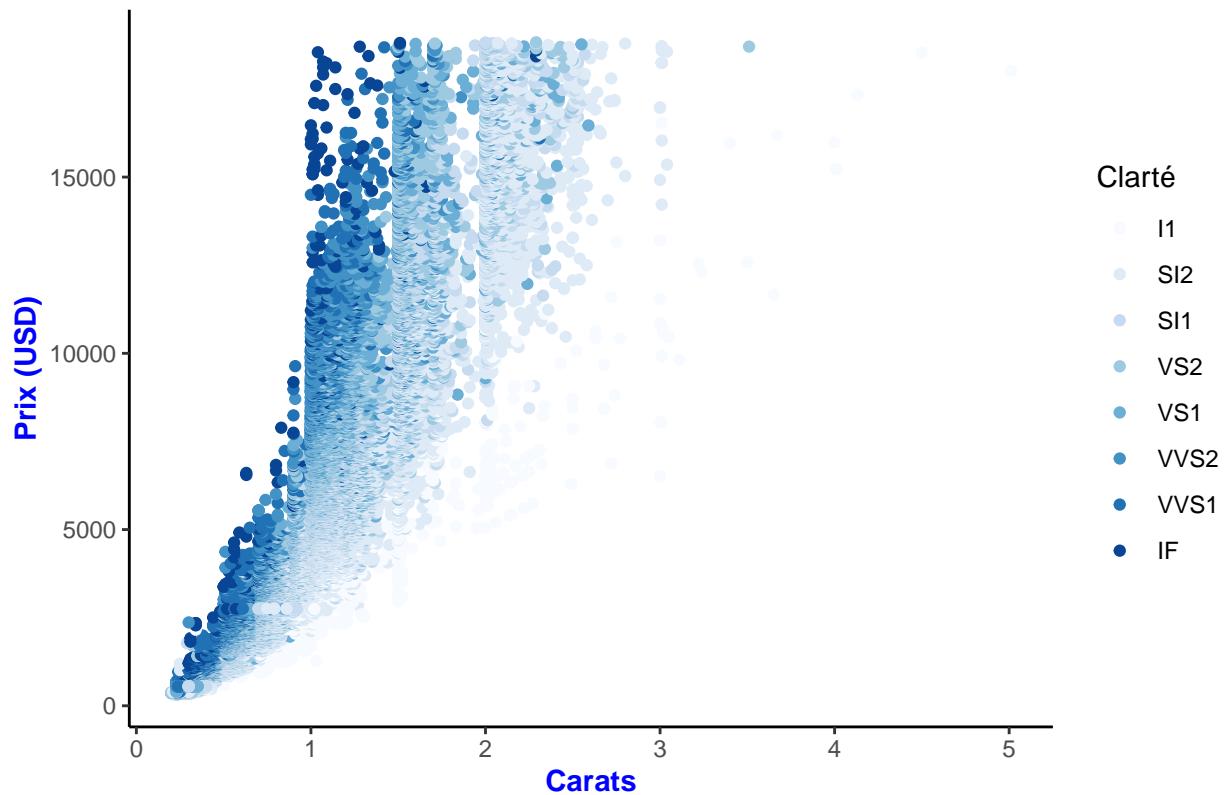
Distribution des Diamants par Qualité de Coupe



- `ggplot(data = diamonds, aes(x = cut, fill = cut))`: Initialise un graphique avec les données `diamonds`, avec `cut` comme axe x et couleur.
- `geom_bar()`: Ajoute des barres au graphique.
- `labs()`: Définit le titre et les étiquettes des axes.
- `scale_fill_brewer()`: Personnalise les couleurs des barres.
- `theme_minimal()`: Utilise un thème minimaliste pour le graphique.

```
ggplot(data = diamonds, aes(x = carat, y = price, color = clarity)) +
  geom_point() +
  labs(title = "Relation entre le Poids et le Prix des Diamants",
       x = "Carats",
       y = "Prix (USD)",
       color = "Clarté") +
  scale_color_brewer(type = "seq", palette = "Blues") +
  theme_classic() +
  theme(
    axis.title = element_text(colour = "blue", face = "bold"),
    plot.title = element_text(colour = "blue", face = "bold"),
    plot.subtitle = element_text(colour = "blue", face = "bold"))
```

Relation entre le Poids et le Prix des Diamants

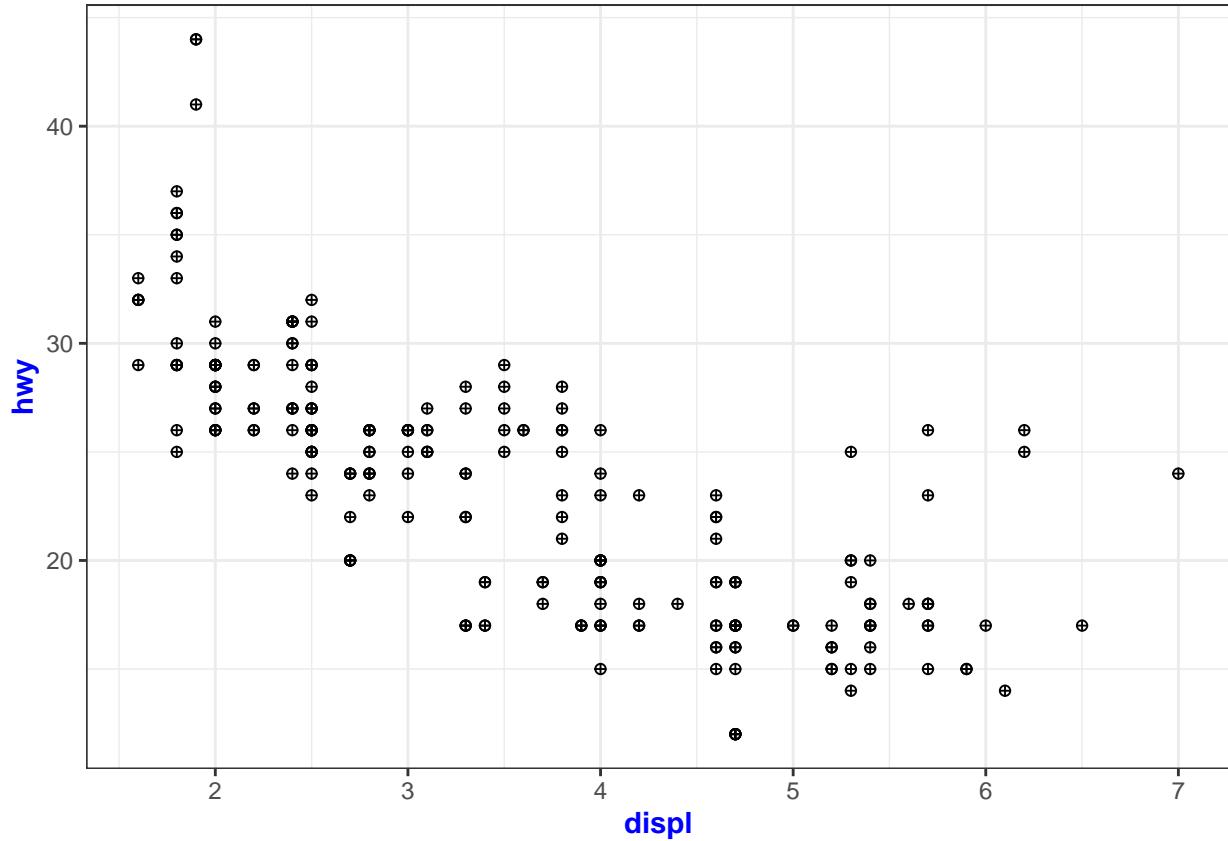


- Utilise la clarté (clarity) pour colorer les points.
- `scale_color_brewer()`: Applique une palette de couleurs.
- `theme_classic()`: Utilise un thème classique.
- `theme()`: Personnalise davantage les éléments du graphique.

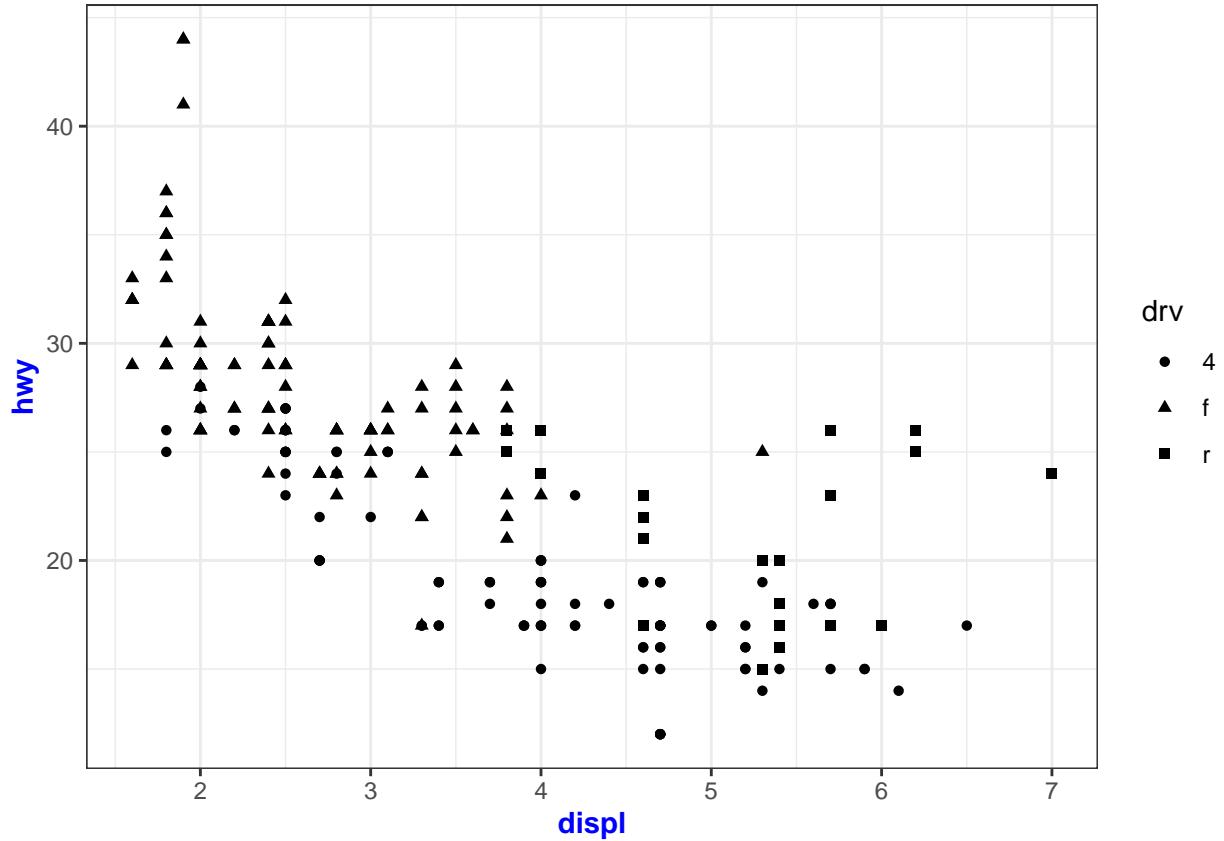
```
# Utilisation de `theme_set`  
theme_set(theme_bw() + theme(  
  axis.title = element_text(colour = "blue", face = "bold"),  
  plot.title = element_text(colour = "blue", face = "bold"),  
  plot.subtitle = element_text(colour = "blue", face = "bold")))
```

- Définit un thème par défaut pour tous les graphiques ultérieurs.

```
# Différence entre Setting et Mapping  
  
# gauche : setting  
ggplot(mpg, aes(x = displ, y = hwy)) +  
  geom_point(shape = 10)
```



```
# droite : mapping
ggplot(mpg, aes(x = displ, y = hwy, shape = drv)) +
  geom_point()
```

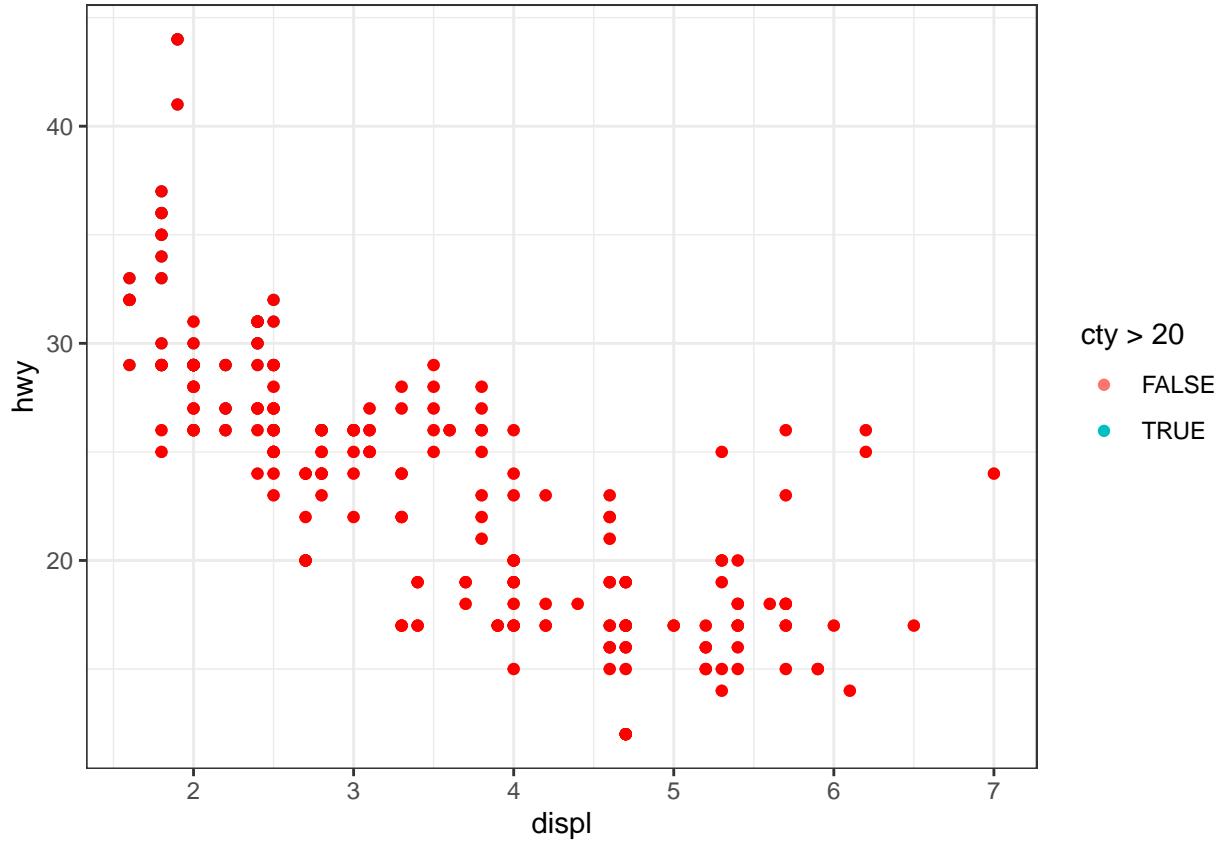


- Montre la différence entre définir une propriété pour tous les points (`setting`) et mapper une propriété à une variable (`mapping`).

```
# Plusieurs façons de modifier les couleurs et les thèmes
p <- ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_point()

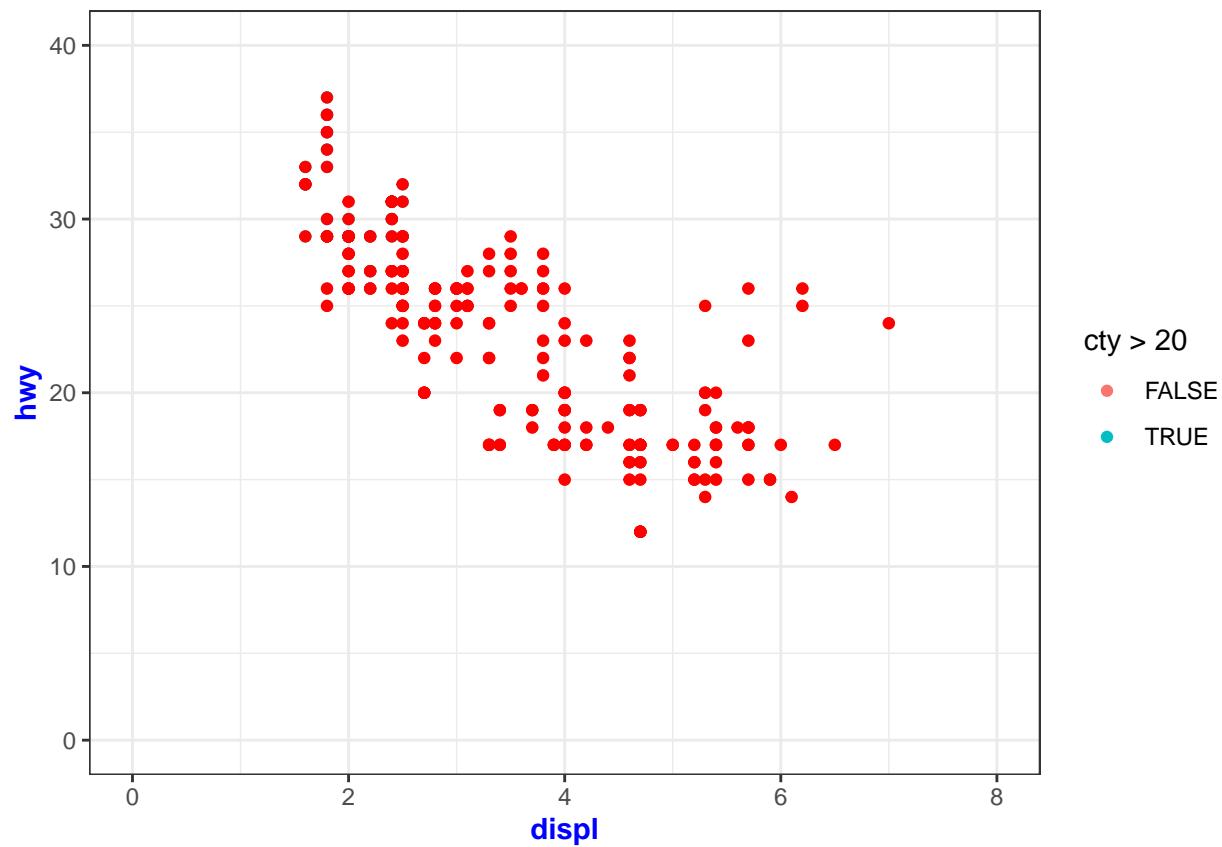
p <- p + geom_point(color = "red")
p <- p + aes(color = drv)
p <- p + aes(color = cty > 20)

p + theme_bw()
```

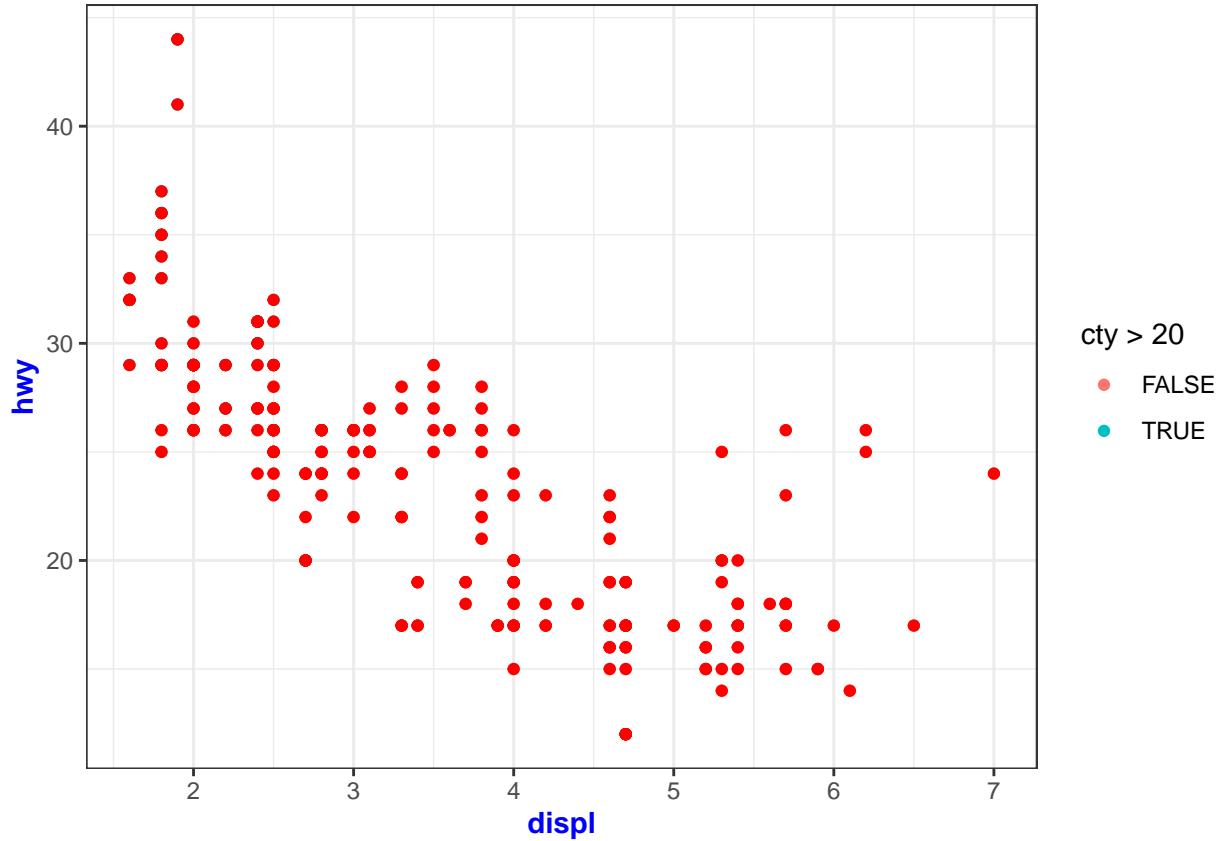


```
p + xlim(0, 8) + ylim(0, 40)
```

```
## Warning: Removed 3 rows containing missing values (`geom_point()`).
## Removed 3 rows containing missing values (`geom_point()`).
```

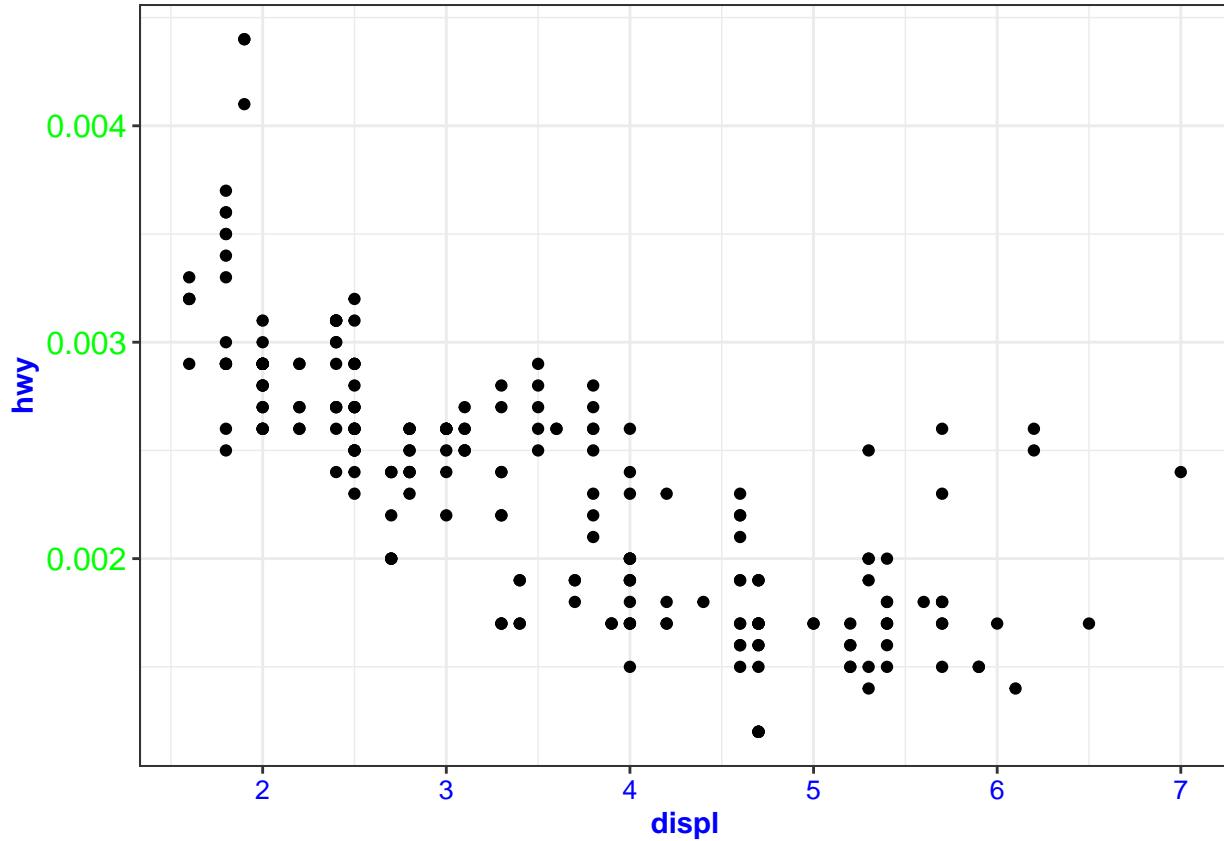


p



- Explore différentes manières de modifier les couleurs et les aspects du graphique.
- Utilise des fonctions comme `theme_bw()`, `xlim()`, `ylim()`, `scale_x_continuous()`, et `scale_y_continuous()` pour personnaliser le graphique.

```
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_point() +
  scale_x_continuous(breaks = seq(0, 10, 1)) +
  scale_y_continuous(labels = scales::comma_format(scale = 1e-4)) +
  theme(axis.text.x = element_text(size = 10, color = "blue"),
        axis.text.y = element_text(size = 12, color = "green"))
```



Techniques Avancées de Visualisation :

Utilisation de facettes pour des graphiques multi-panneaux avec les données `babynames`

- Chargement des bibliothèques `:babynames`, et `viridis` utilisées pour les données, et les palettes de couleurs, respectivement.
- Préparation des données : Sélection des prénoms spécifiques dans le dataset `babynames` et filtrage par sexe.
- Création du graphique :
 - `ggplot()` initialise le graphique.
 - `geom_area()` crée des graphiques à aire.
 - `scale_fill_viridis()` applique une palette de couleurs.
 - `theme()` et `theme_ipsum()` sont utilisés pour personnaliser l'apparence du graphique.
 - `facet_wrap(~name, scale="free_y")` divise le graphique en plusieurs panneaux, un pour chaque nom, avec des échelles y libres.

Ce script crée un graphique qui montre la popularité de certains prénoms américains au fil du temps.

```
library(babynames)
library(viridis)
```

```
## Loading required package: viridisLite
```

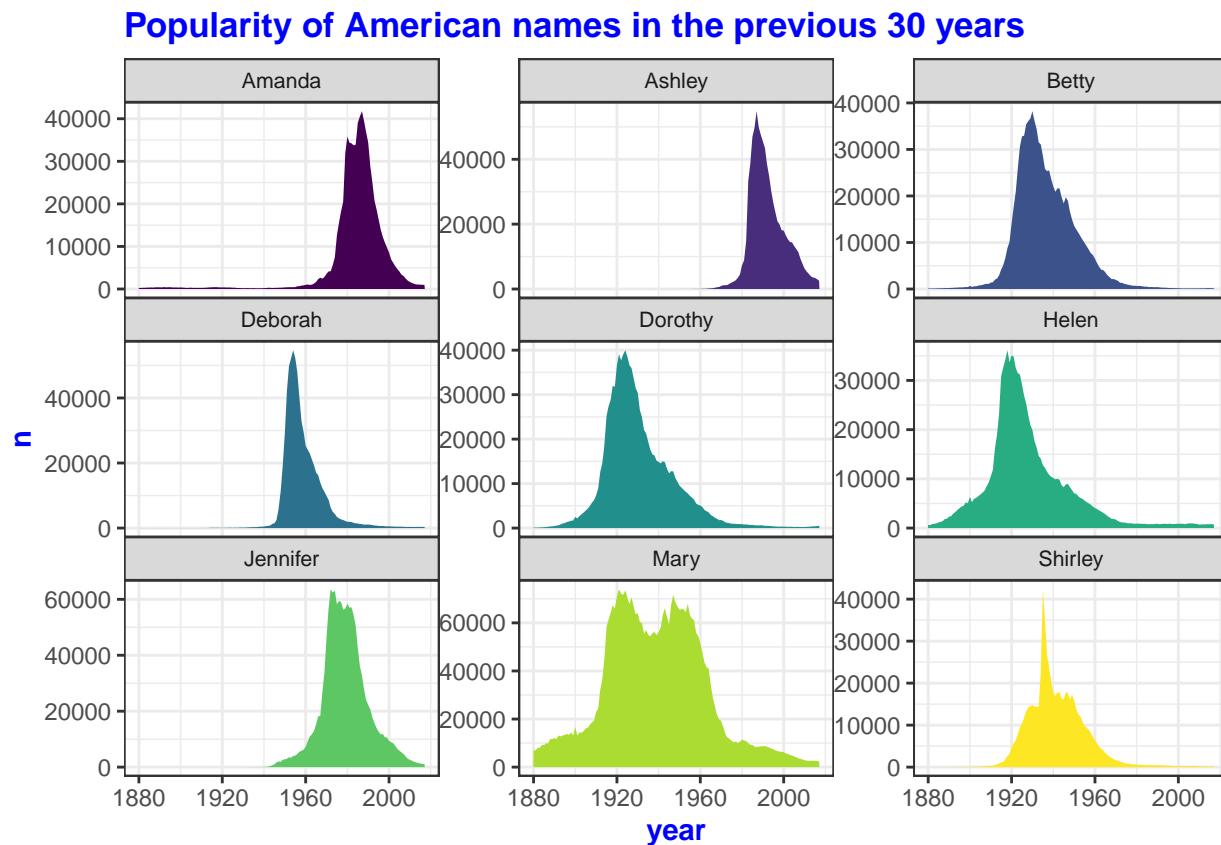
```

# Load dataset from github
data <- read.table("https://raw.githubusercontent.com/holtzy/data_to_viz/master/Example_dataset/3_TwoNu")
data$date <- as.Date(data$date)

# Load dataset from github
don <- babynames %>%
  filter(name %in% c("Ashley", "Amanda", "Mary", "Deborah",
                     "Dorothy", "Betty", "Helen", "Jennifer", "Shirley")) %>%
  filter(sex=="F")

# Plot
don %>%
  ggplot( aes(x=year, y=n, group=name, fill=name)) +
  geom_area() +
  scale_fill_viridis(discrete = TRUE) +
  theme(legend.position="none") +
  ggtitle("Popularity of American names in the previous 30 years") +
  theme(
    legend.position="none",
    panel.spacing = unit(0, "lines"),
    strip.text.x = element_text(size = 8),
    plot.title = element_text(size=13)
  ) +
  facet_wrap(~name, scale="free_y")

```



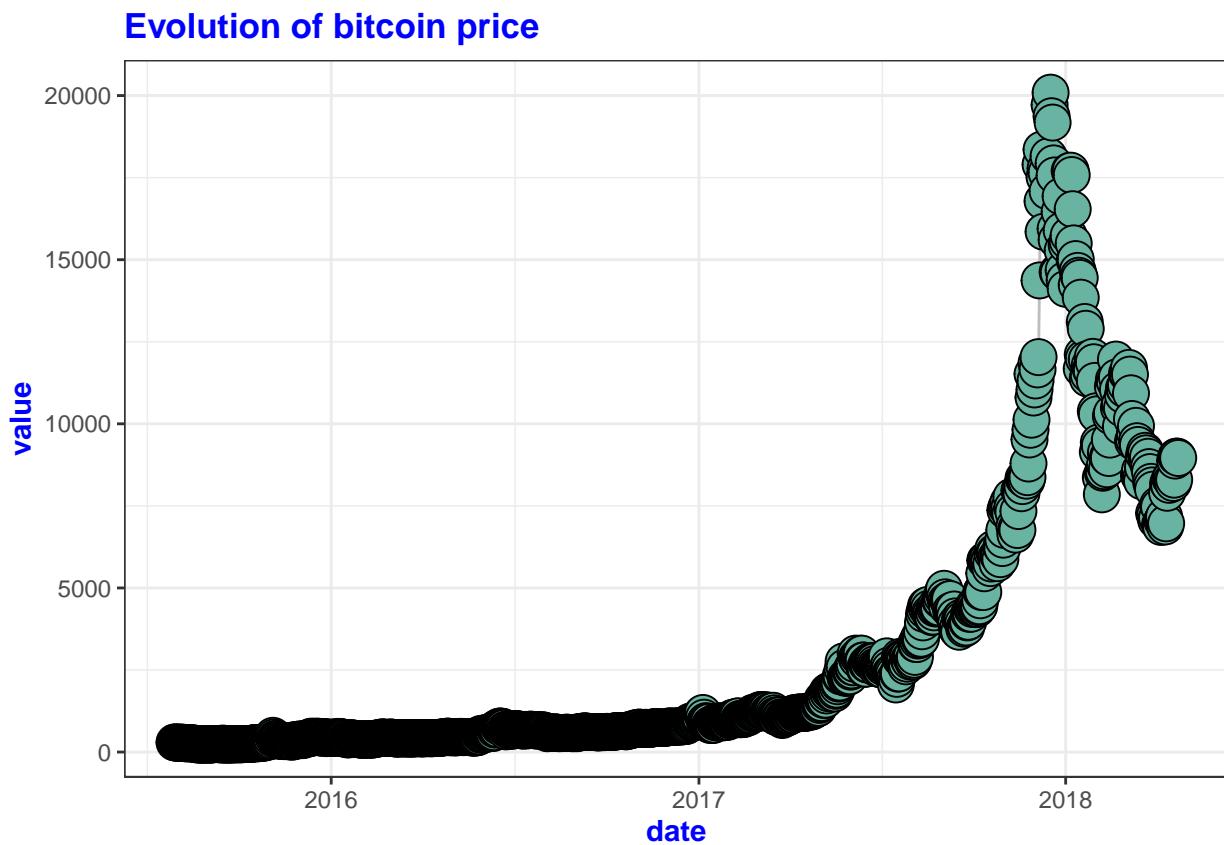
Graphique de l'évolution du prix du bitcoin

- **Chargement et préparation des données :** Les données sont chargées d'une source en ligne et la colonne date est convertie au format Date.
- **Création du graphique :**
 - `ggplot()` initialise le graphique.
 - `geom_line()` et `geom_point()` sont utilisés pour créer un graphique linéaire avec des points mis en évidence.
 - `theme_ipsum()` avec `base_family = "Arial"` applique un thème avec une police standard.
 - `ggtitle()` ajoute un titre au graphique.

Ce script montre l'évolution récente du prix du bitcoin.

```
data <- read.table("https://raw.githubusercontent.com/holtzy/data_to_viz/master/Example_dataset/3_TwoNum")
data$date <- as.Date(data$date)

# Plot
data %>%
  tail(1000) %>%
  ggplot( aes(x=date, y=value)) +
  geom_line( color="grey") +
  geom_point(shape=21, color="black", fill="#69b3a2", size=6) +
  ggtitle("Evolution of bitcoin price")
```



Créer une carte avec leaflet

```
# Installer et charger le package leaflet
if (!require(leaflet)) {
  install.packages("leaflet")
}

## Loading required package: leaflet

library(leaflet)

# Créer une carte de base
map <- leaflet() %>%
  addTiles() # Ajoute les tuiles de base d'OpenStreetMap

# Ajouter un marqueur
map <- map %>%
  addMarkers(lng = -0.09, lat = 51.50, popup = "The marker is placed at London")

# Changer le fond de carte
map <- map %>%
  addProviderTiles(providers$Stamen.TonerLite)

# Données pour les marqueurs supplémentaires
data <- data.frame(
  lng = c(-0.09, -0.11, -0.12),
  lat = c(51.50, 51.52, 51.48),
  label = c("Point 1", "Point 2", "Point 3")
)

# Ajouter des marqueurs supplémentaires
map <- map %>%
  addMarkers(data = data, ~lng, ~lat, popup = ~label)

# Afficher la carte
map

## PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is installed, please

Paris <- c(2.351462,48.8567)
m2 <- leaflet() %>% setView(lng = Paris[1], lat = Paris[2], zoom = 12) %>%
  addTiles()
m2 %>% addProviderTiles("Stamen.Toner")

content <- paste(sep = "<br/>",
  "<b><a href='https://www.intelligence-artificielle-school.com/ecoole/nos-campus/toulouse/'>IA School -",
  "2 Bd de Strasbourg",
  "31000 Toulouse"
)

leaflet() %>%
```

```

addTiles() %>%
  addPopups(1.4485130539619018, 43.60676750775245,
             content, options = popupOptions(closeButton = FALSE)
  )

# Chargement des bibliothèques nécessaires
library(plotly)

## 
## Attaching package: 'plotly'

## The following object is masked from 'package:ggplot2':
## 
##     last_plot

## The following object is masked from 'package:stats':
## 
##     filter

## The following object is masked from 'package:graphics':
## 
##     layout

library(tidyverse)
library(lubridate)

# Configurer la graine aléatoire pour la reproductibilité
set.seed(123)

# Créer une séquence de dates (chaque mois de 2020)
dates <- seq(ymd("2020-01-01"), ymd("2020-12-31"), by="month")

# Codes d'état des États-Unis (exemple avec 5 états)
state_codes <- c("AL", "AK", "AZ", "AR", "CA")

# Générer des données aléatoires
num_rows <- length(dates) * length(state_codes)
df <- data.frame(
  time = rep(dates, each=length(state_codes)),
  state_code = rep(state_codes, times=length(dates)),
  num_colonies = sample(100:10000, num_rows, replace = TRUE),
  percent_lost = runif(num_rows, 0, 100),
  varroa_mites = runif(num_rows, 0, 100),
  other_pests = runif(num_rows, 0, 100),
  diseases = runif(num_rows, 0, 100),
  pesticides = runif(num_rows, 0, 100),
  other = runif(num_rows, 0, 100),
  unknown = runif(num_rows, 0, 100)
)
# Afficher les premières lignes du dataframe
head(df)

```

```

##          time state_code num_colonies percent_lost varroa_mites other_pests
## 1 2020-01-01        AL      2562    33.28235    35.39046   96.110479
## 2 2020-01-01        AK      2610    48.86130    64.99852   72.839443
## 3 2020-01-01        AZ      8817    95.44738    37.47140   68.637508
## 4 2020-01-01        AR      3085    48.29024    35.54454    5.284394
## 5 2020-01-01        CA      1941    89.03502    53.36879   39.522013
## 6 2020-02-01        AL      9433    91.44382    74.03344   47.784538
##      diseases pesticides     other unknown
## 1 13.710608    11.94048 13.70675 43.61300
## 2 39.658459    52.60297 90.53096 46.40166
## 3 22.498533    22.50734 57.63018 16.52981
## 4  5.795856    48.64118 39.54489 58.49366
## 5 39.589269    37.02148 44.98025 27.07780
## 6  6.492830    98.33502 70.65019 23.00969

fig <- df %>%
  plot_ly() %>%
  add_trace(
    type = "choropleth",
    locations = ~state_code,
    locationmode = "USA-states",
    z = ~varroa_mites,
    colorscale = "Viridis_r",
    color = ~varroa_mites
  ) %>%
  layout(
    geo = list(scope = "usa"),
    frame = ~time
  )
fig

```

Warning: 'layout' objects don't have these attributes: 'frame'

Valid attributes include:

'_deprecated', 'activeshape', 'annotations', 'autosize', 'autotypenumbers', 'calendar', 'clickmode',

Pour en savoir plus : (data to viz)[<https://www.data-to-viz.com/>]

Exercices

Pour commencer à pratiquer, suivez ces étapes :

- Accédez au Dépôt GitHub :** Visitez l'URL fournie : <https://github.com/universdesdonnees/Introduction-a-R> pour accéder au dépôt GitHub contenant les matériaux du cours.
- Trouvez le Fichier des Exercices :** Dans le dépôt, localisez le fichier nommé **exercices7.txt**. Ce fichier contient les premiers exercices que vous devez pratiquer.
- Lisez et Essayez de Résoudre les Exercices :** Ouvrez le fichier **exercices7.txt** et lisez attentivement les exercices. Essayez de les résoudre par vous-même dans votre environnement R (comme RStudio). Il est important de pratiquer par vous-même avant de regarder les solutions pour mieux apprendre.

4. **Consultez la Correction :** Une fois que vous avez tenté de résoudre les exercices, ou si vous rencontrez des difficultés, consultez le fichier **correction_exercices7.R** pour voir les solutions. Analysez les solutions pour comprendre les méthodes et logiques utilisées.