

# Visualisation des données

*Joël K. Kazadi*

2023-08-23

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Grammaire graphique ggplot2</b>	<b>2</b>
2.1	Les couches de base . . . . .	2
2.2	La couche des échelles . . . . .	4
2.2.1	Gestion des axes du graphique . . . . .	5
2.2.2	Gestion des couleurs . . . . .	7
2.3	La couche de facettage . . . . .	10
2.3.1	Facette en grille . . . . .	10
2.3.2	Facettage en enveloppe . . . . .	13
<b>3</b>	<b>Personnalisation des graphiques</b>	<b>15</b>
3.1	Thème . . . . .	15
3.2	Légende . . . . .	18
3.3	Annotations . . . . .	20
<b>4</b>	<b>Astuces supplémentaires</b>	<b>24</b>
4.1	Graphiques interactifs . . . . .	24
4.2	Combinaison des graphiques . . . . .	25
4.3	Sauvegarde des graphiques . . . . .	27

# 1 Introduction

La visualisation des données est le processus de représentation graphique des données afin de les rendre plus faciles à comprendre. Elle permet de représenter une grande variété de données pour identifier des tendances dans les données, de communiquer des idées complexes ou de prendre des décisions éclairées. *La Data Viz, c'est l'art de communiquer des informations de façon claire et efficace à l'aide des graphiques.*

Il existe plusieurs manières de visualiser les données en R. Dans le cadre de cette leçon, nous n'allons nous concentrer que sur la grammaire graphique du package `ggplot2`. Développé par HADLEY WICKHAM, ce package a l'avantage d'avoir une syntaxe claire et concise, offre des possibilités de personnalisation des graphiques, et dispose d'une communauté très active de développeurs. Les notions suivantes seront abordées tout au long de la leçon : (i) les composants de la grammaire graphique `ggplot2`, (ii) la personnalisation des graphiques (annotation, thèmes et légendes), et (iii) quelques astuces supplémentaires (graphiques interactifs et combinaison des graphiques).

## 2 Grammaire graphique `ggplot2`

Par “grammaire graphique”, il faut entendre la méthodologie de conception d'un graphique statistique composants après composants. La puissance du package `ggplot2`, c'est que ses composants (layers) permettent un paramétrage poussé susceptible d'aboutir à des graphiques hautement sophistiqués. Les principaux layers sont :

- le dataset : c'est le jeu de données que l'on souhaite visualiser ;
- les aesthetiques : c'est essentiellement les variables à représenter ;
- les géométriques : c'est le type de graphique à tracer ;
- les échelles : c'est ce qui permet de contrôler le lien entre les données ;
- le facettement : c'est ce qui met en évidence des sous-ensembles dans les données.

### 2.1 Les couches de base

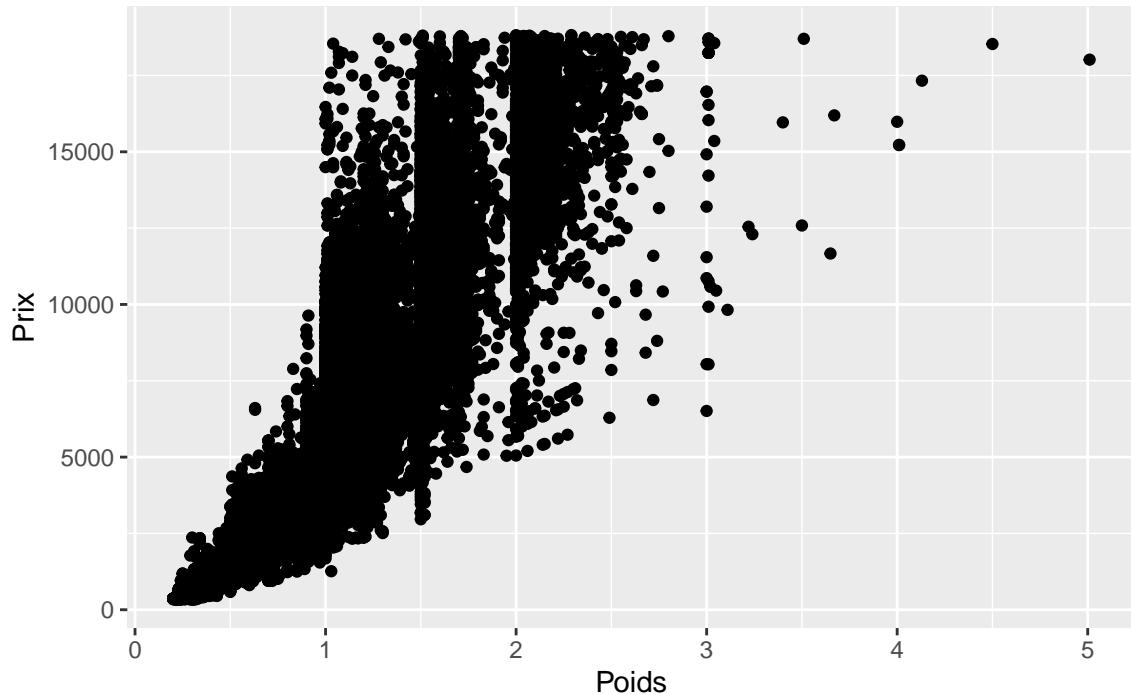
L'élaboration du graphique se réalise au moyen de la fonction `ggplot()` du package `ggplot2`. La structure de base pour l'élaboration consiste d'abord à passer en paramètre de la fonction `ggplot()` le dataset et les aesthetiques. Le dataset doit toujours être un objet de classe “dataframe”. Les aesthétiques définissent les propriétés visuelles du graphique, i.e. les variables en abscisse et en ordonnée, la couleur de remplissage, le type de lignes, etc. Ensuite, il faut définir le type de graphique à tracer (couche géométrique). Dans cette optique, on peut énumérer les histogrammes, les courbes, le diagramme à barres, le diagramme circulaire, le nuage de points, etc.

Considérons le cas du dataset `diamonds` logé dans le package `ggplot2`. Ce dataset contient les caractéristiques de 53 940 diamants (prix, poids, qualité, couleur, clarté, longueur, largeur, profondeur, ...). Représentons un nuage de points pour visualiser la relation existant entre le poids du diamant et son prix.

```
#install.packages("ggplot2", dependencies = TRUE)
library(ggplot2)

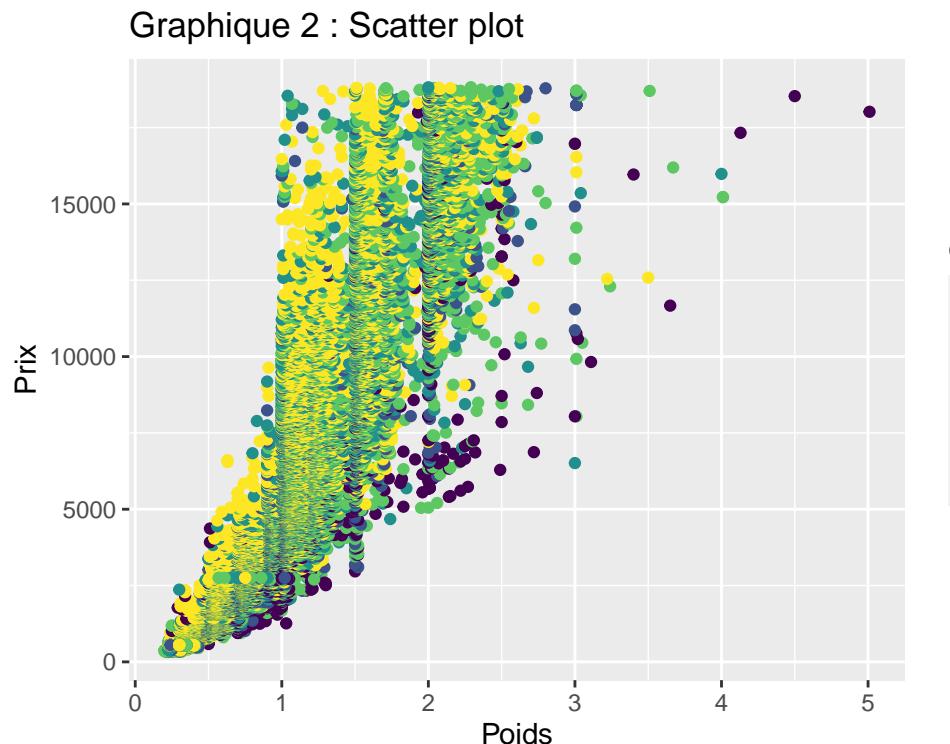
attach(diamonds)
ggplot(data = diamonds, mapping = aes(x = carat,
                                         y = price)) +
  labs(title = "Graphique 1 : Scatter plot",
       x = "Poids", y = "Prix") +
  geom_point()
```

Graphique 1 : Scatter plot



On comprend que l'ajout d'une couche supplémentaire dans l'élaboration du graphique se réalise au moyen du signe “+”. Le Graphique 1 indique que le prix du diamant est positivement corrélé avec son poids. Rajoutons une variable qualitative dans le graphique afin d'affiner l'analyse.

```
ggplot(data = diamonds, mapping = aes(x = carat,
                                       y = price,
                                       colour = cut)) +
  labs(title = "Graphique 2 : Scatter plot",
       x = "Poids", y = "Prix") +
  geom_point()
```



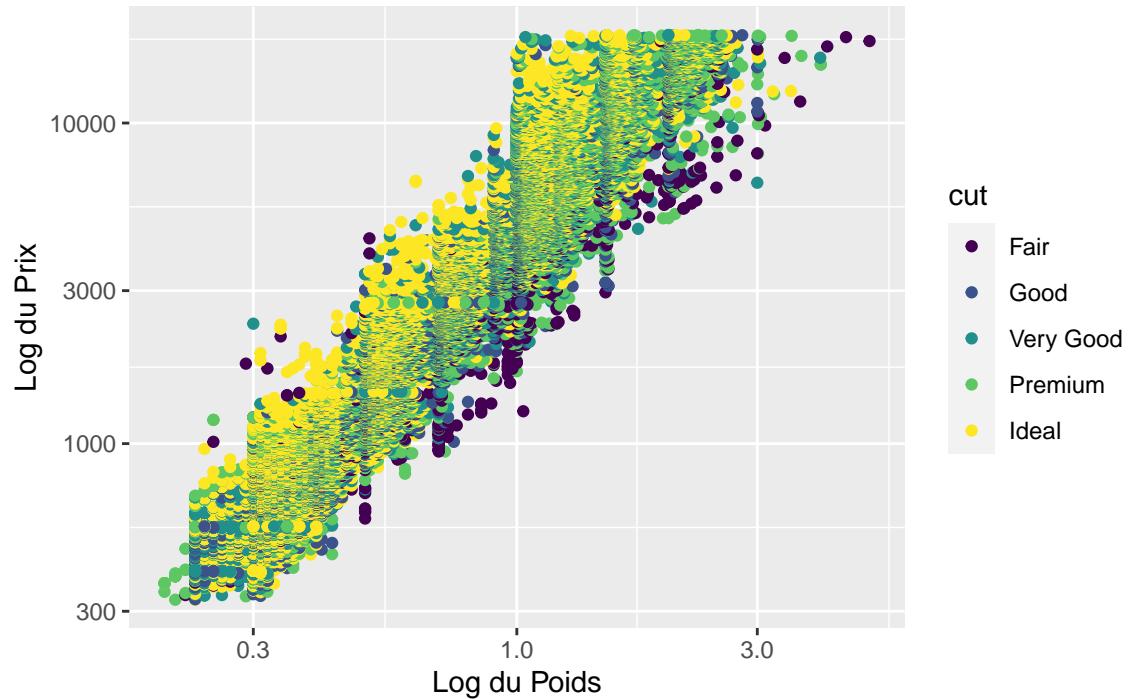
Dans le Graphique 2, les points sont coloriés selon la qualité du diamant. On constate que les prix des diamants de faible qualité augmentent très lentement en fonction de leurs poids. Par contre, les prix des diamants de haute qualité s'accroissent plus rapidement selon le poids.

## 2.2 La couche des échelles

Cette couche permet de gérer les axes du graphique, la palette des couleurs, la taille des tracés, mais aussi opérer des transformations statistiques sur les données brutes. Reprenons le graphique précédent, et effectuons une transformation logarithmique sur les variables à représenter.

```
ggplot(data = diamonds, mapping = aes(x = carat,
                                       y = price,
                                       colour = cut)) +
  labs(title = "Graphique 3 : Scatter plot",
       x = "Log du Poids", y = "Log du Prix") +
  geom_point() + scale_x_log10() + scale_y_log10()
```

Graphique 3 : Scatter plot



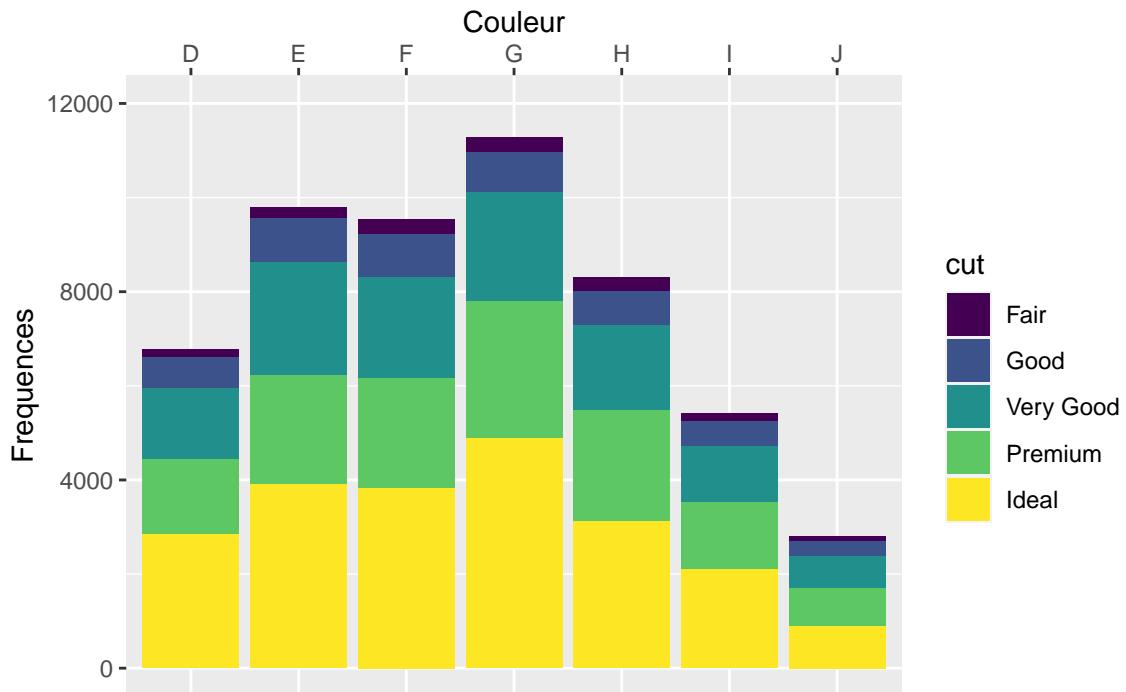
À la lumière du Graphique 3, l'on constate que la relation entre le poids et le prix du diamant révèle désormais une apparence linéaire, après transformation logarithmique. En effet, ce type de transformation rend les distributions des données plus linéaires et réduit l'influence des valeurs extrêmes (*outliers*).

### 2.2.1 Gestion des axes du graphique

Considérons à présent le cas de gestion des axes du graphique au moyen de la couche des échelles. Pour ce faire, représentons au moyen d'un diagramme à barres le nombre de diamants par couleur selon la qualité.

```
ggplot(data = diamonds, mapping = aes(x = color,
                                         fill = cut)) +
  labs(title = "Graphique 4 : Bar plot",
       x = "Couleur", y = "Frequencies") +
  geom_bar() +
  scale_y_continuous(limits = c(0,12000), n.breaks = 4) +
  scale_x_discrete(position = "top")
```

Graphique 4 : Bar plot



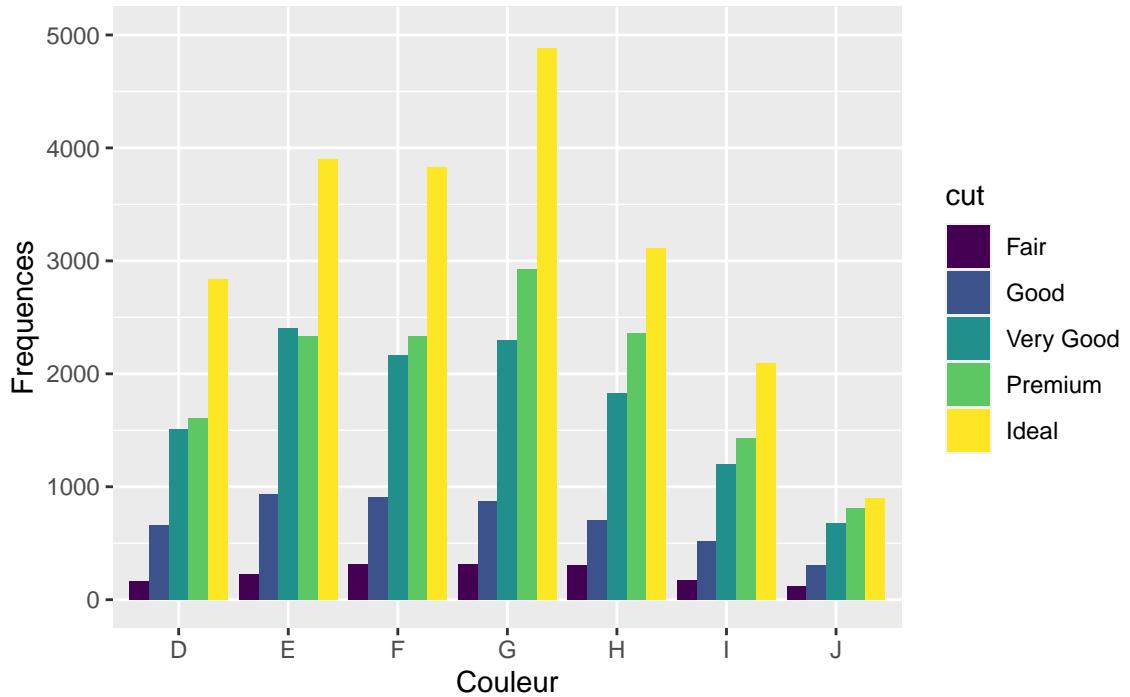
Dans le Graphique 4, nous avons défini les bornes limites de la plage de variation pour l'axe des ordonnées, tout en spécifiant le pas de cette plage (`n.breaks`). Pour l'axe des abscisses, nous avons défini la position des étiquettes (la position par défaut est “`bottom`”).

Nous pouvons modifier le Graphique 4 en passant d'une représentation empilée à une représentation multiple. Pour y parvenir, il suffira de spécifier l'argument `position` dans la fonction géométrique à “`dodge`”<sup>1</sup>.

```
ggplot(data = diamonds, mapping = aes(x = color,
                                         fill = cut)) +
  labs(title = "Graphique 5 : Bar plot",
       x = "Couleur", y = "Frequences") +
  geom_bar(position = "dodge") +
  scale_y_continuous(limits = c(0,5000), n.breaks = 5) +
  scale_x_discrete(position = "bottom")
```

<sup>1</sup>Cet argument est positionné à défaut à “`stack`” (empilement).

Graphique 5 : Bar plot



### 2.2.2 Gestion des couleurs

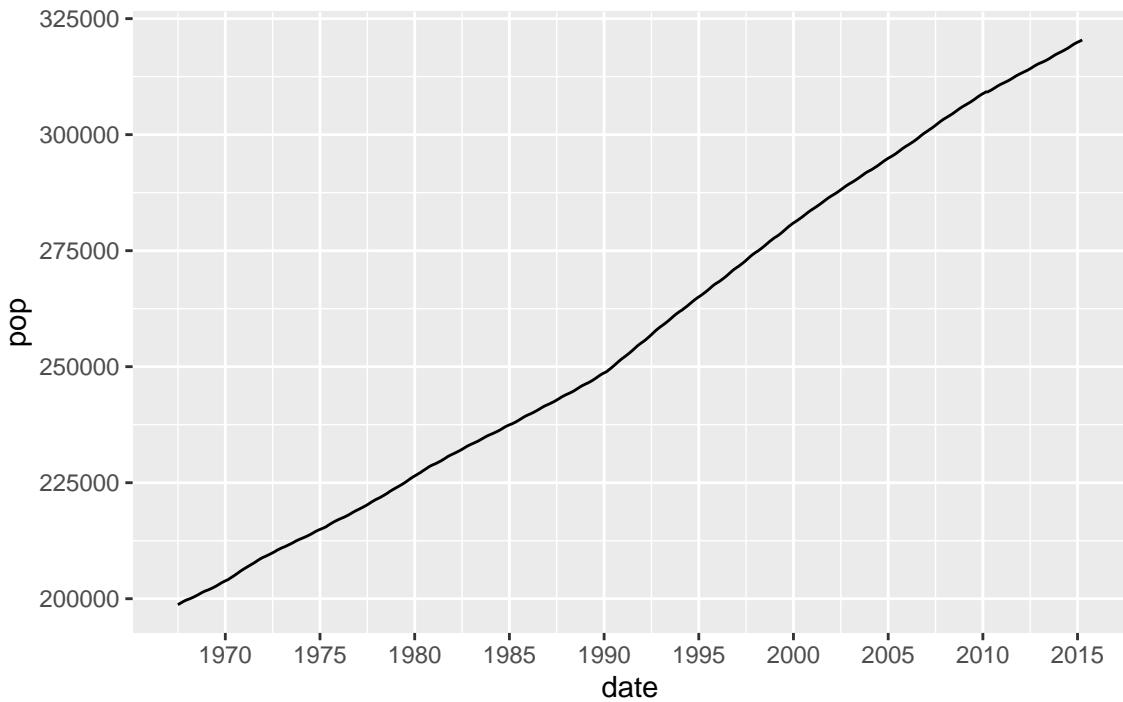
Prenons le cas de gestion des couleurs dans le graphique. Nous allons travailler, pour ce cas, avec le dataset `economics` logé dans le package `ggplot2`. Ce dataset se rapporte à cinq séries macroéconomiques mensuelles des USA de juillet 1967 à avril 2015 (dépenses de consommation, volume démographique, taux d'épargne, durée de chômage, population non-employée). Représentons au moyen d'une courbe l'évolution de la population totale dans le temps.

```
#install.packages("ggplot2", dependencies = TRUE)
library(ggplot2)

economics <- economics
attach(economics)

ggplot(data = economics, mapping = aes(x = date, y = pop)) +
  labs(title = "Graphique 6 : Line plot") +
  geom_line() +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y")
```

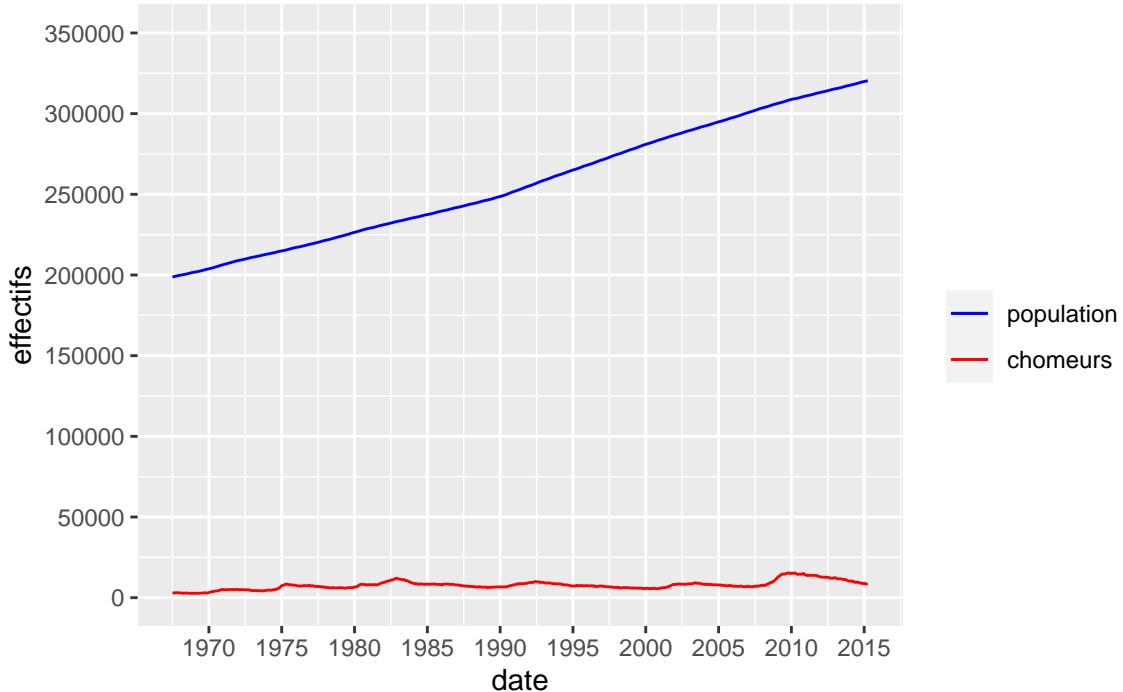
Graphique 6 : Line plot



Dans le Graphique 5, nous avons spécifié le pas temporel à 5 ans et n'avons fait apparaître que les années comme étiquettes de l'axe des abscisses (*cf.* gestion des axes). Comparons à présent l'évolution de la population totale à celle de la population en chômage. Nous allons utiliser des références pour définir les couleurs des deux courbes à tracer.

```
ggplot(data = economics, mapping = aes(x = date)) +
  labs(title = "Graphique 7 : Line plot", y = "effectifs") +
  geom_line(aes(y = pop, colour = "A")) +
  geom_line(aes(y = unemploy, colour = "B")) +
  scale_color_manual(name = "",
    labels = c("population", "chomeurs"),
    values = c("A" = "blue", "B" = "red")) +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y") +
  scale_y_continuous(n.breaks = 7, limits = c(0, 350000))
```

Graphique 7 : Line plot



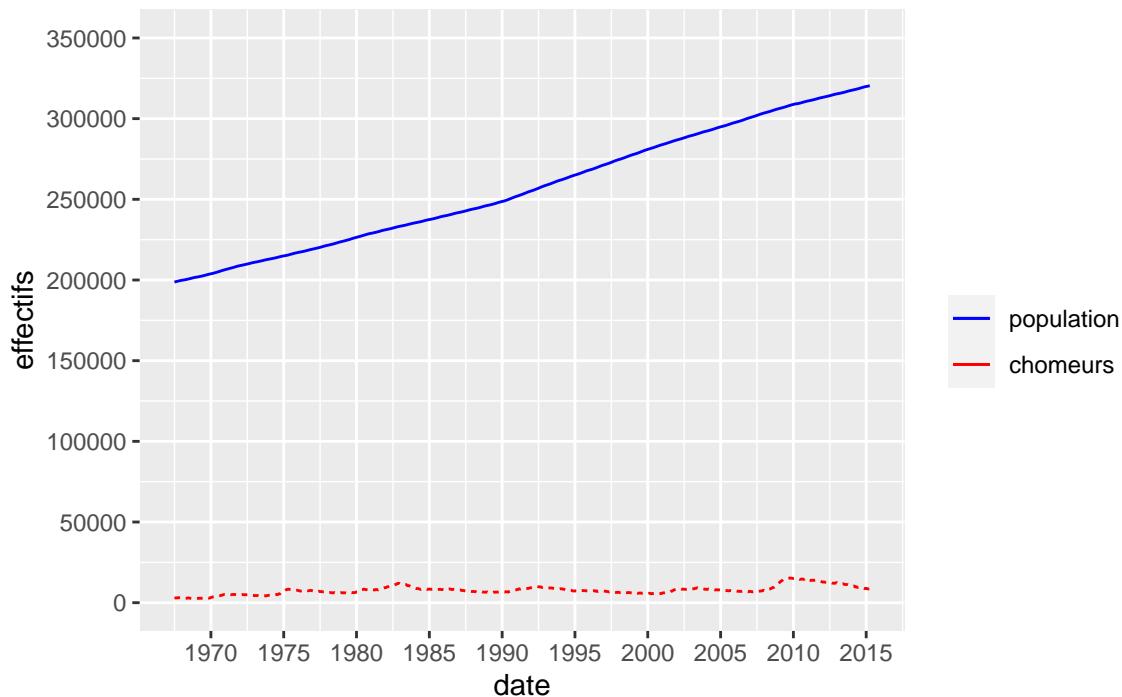
Nous pouvons aussi modifier le type de lignes à tracer sur le graphique. En combinant la modification des couleurs et celle des types de lignes, R produit automatiquement deux légendes pour les deux modifications. Ainsi, pour de raisons de commodité, nous avons désactivé la légende des types de lignes grâce à la fonction `guides`<sup>2</sup>.

```
ggplot(data = economics, mapping = aes(x = date)) +
  labs(title = "Graphique 8 : Line plot", y = "effectifs") +
  geom_line(aes(y = pop, colour = "A", linetype = "dashed")) +
  geom_line(aes(y = unemploy, colour = "B", linetype = "dotted")) +
  scale_color_manual(name = "",
    labels = c("population", "chomeurs"),
    values = c("A" = "blue", "B" = "red")) +
  guides(linetype = "none") +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y") +
  scale_y_continuous(n.breaks = 7, limits = c(0, 350000))
```

---

<sup>2</sup>Pour désactiver plutôt la légende des couleurs, placer l'argument `colour` à “`none`”.

Graphique 8 : Line plot



## 2.3 La couche de facettage

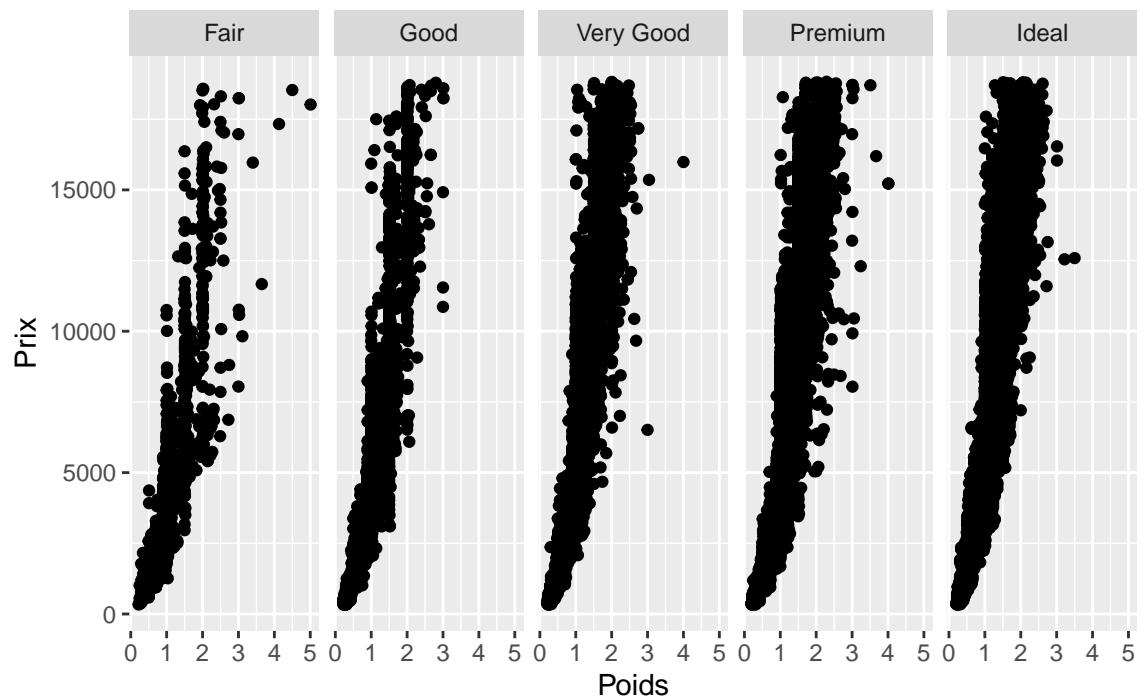
Le facettage permet de représenter de multiples graphiques avec des sous-ensembles du dataset créés à partir des variables discrètes. Avec le package `ggplot2`, il est possible de faire du facettage d'au moins deux manières : (i) le facette en grille, et (ii) le facettage en enveloppe.

### 2.3.1 Facette en grille

Ce type de facettage se réalise au moyen de la fonction `facet_grid()`. Cette fonction permet de créer des sous-groupes sur le graphique à partir de deux variables qualitatives à comparer sur une grille de panneaux. La fonction prend en paramètre le mode de subdivision sous forme d'une formule (arrangement vertical ou arrangement horizontal). Reprenons notre graphique en nuage de points.

```
ggplot(data = diamonds, mapping = aes(x = carat,
                                         y = price)) +
  labs(title = "Graphique 9 : Scatter plot",
       x = "Poids", y = "Prix") +
  geom_point() +
  facet_grid(. ~ cut)
```

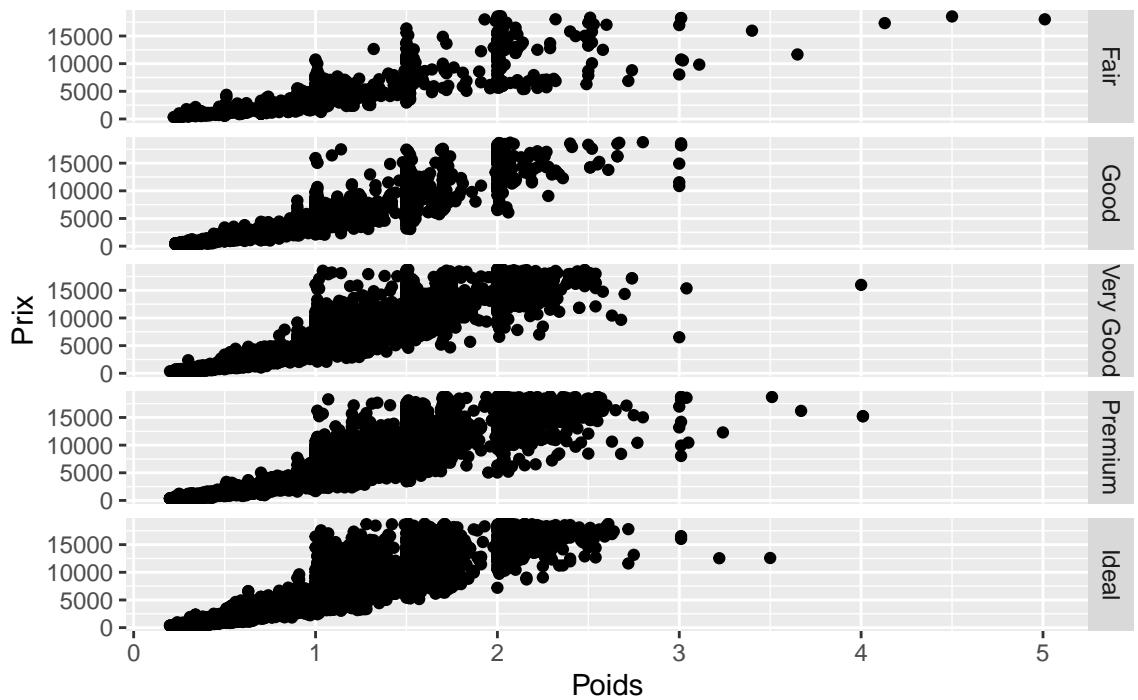
Graphique 9 : Scatter plot



Le facettage a permis de générer autant de nuages selon le nombre de modalités de la variable qualitative renseignée. Dans le Graphique 9, la grille est rangée verticalement. Il est possible d'adopter un arrangement horizontal en modifiant la formule renseignée dans la fonction de facettage.

```
ggplot(data = diamonds, mapping = aes(x = carat,
                                         y = price)) +
  labs(title = "Graphique 10 : Scatter plot",
       x = "Poids", y = "Prix") +
  geom_point() +
  facet_grid(cut ~ .)
```

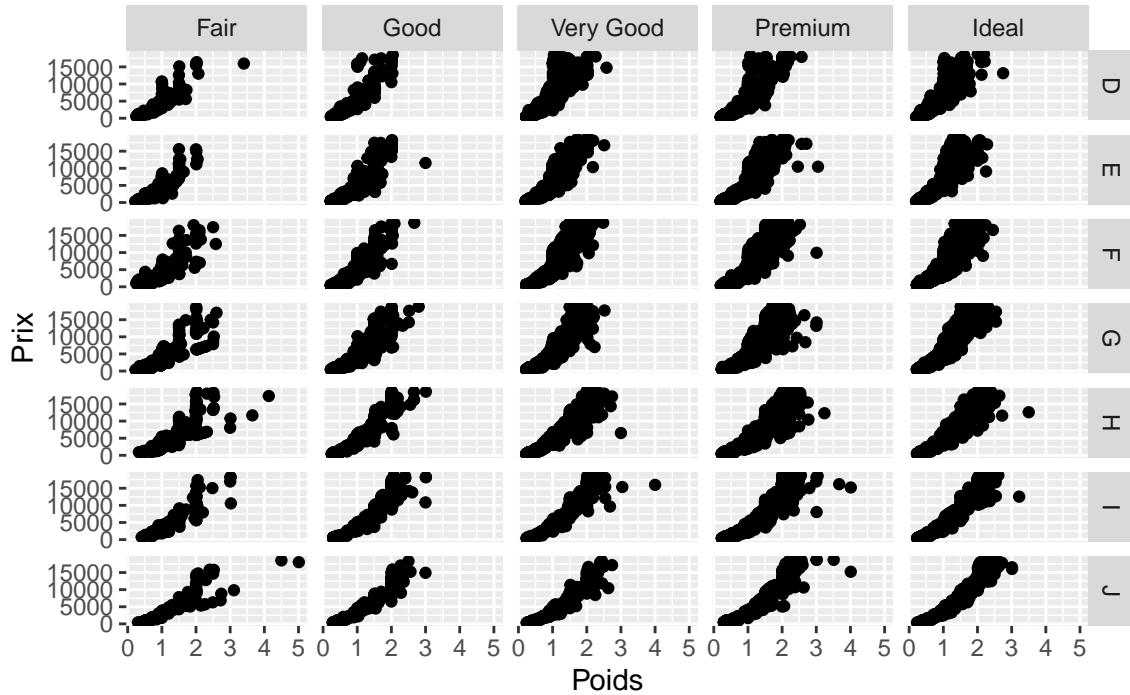
Graphique 10 : Scatter plot



Enfin, nous pouvons mixer les deux types d'arrangement de la grille en combinant deux variables qualitatives dans la fonction de facettement (qualités et couleurs des diamants).

```
ggplot(data = diamonds, mapping = aes(x = carat,
                                         y = price)) +
  labs(title = "Graphique 11 : Scatter plot",
       x = "Poids", y = "Prix") +
  geom_point() +
  facet_grid(color~cut)
```

Graphique 11 : Scatter plot

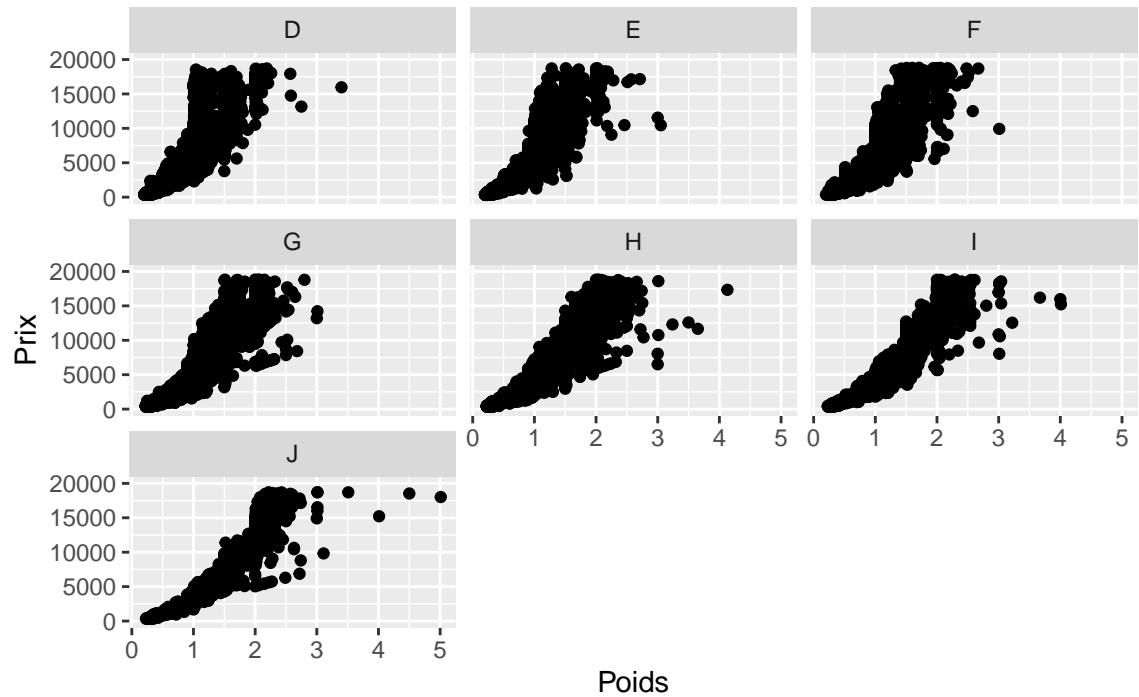


### 2.3.2 Facettage en enveloppe

Ce type de facettage se réalise au moyen de la fonction `facet_wrap()`. Cette fonction permet de créer des sous-groupes sur le graphique à partir d'une variable qualitative en enroulant les autres variables pour s'adapter aux panneaux. Tout comme le facettage en grille, la fonction prend en paramètre le mode de subdivision sous forme d'une formule. Cependant, elle fonctionne comme une matrice ; ce qui implique qu'il faut obligatoirement spécifier le nombre de lignes et de colonnes.

```
ggplot(data = diamonds, mapping = aes(x = carat,
                                         y = price)) +
  labs(title = "Graphique 12 : Scatter plot",
       x = "Poids", y = "Prix") +
  geom_point() +
  scale_y_continuous(limits = c(0,20000), n.breaks = 5) +
  facet_wrap(color~., nrow = 3, ncol = 3)
```

Graphique 12 : Scatter plot



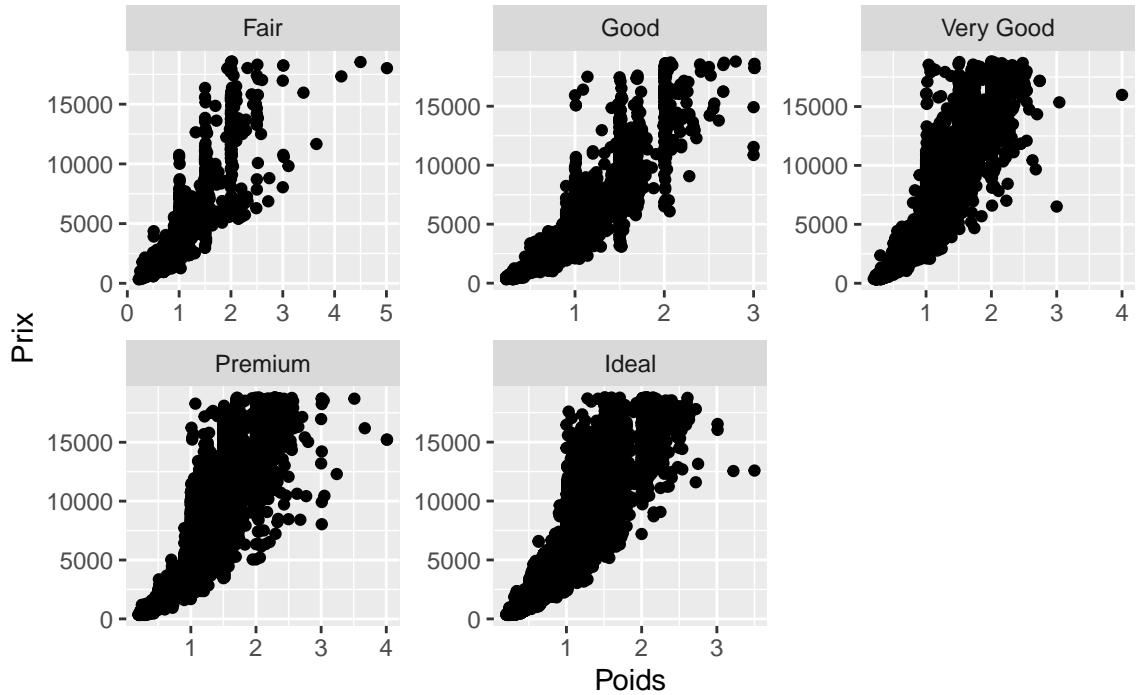
Il est également possible de contrôler les échelles des axes dans chaque panel sur le graphique. Par défaut, ces échelles sont les mêmes pour tous les panneaux. Pour les faire varier par sous-ensemble, il suffit de spécifier l'argument **scales** de la fonction de facettement à “*free*”<sup>3</sup>.

```
ggplot(data = diamonds, mapping = aes(x = carat,
                                         y = price)) +
  labs(title = "Graphique 13 : Scatter plot",
       x = "Poids", y = "Prix") +
  geom_point() +
  facet_wrap(cut ~ ., nrow = 2, ncol = 3, scales = "free")
```

---

<sup>3</sup>Cet argument est positionné par défaut à “*fixed*” (échelles identiques).

Graphique 13 : Scatter plot



### 3 Personnalisation des graphiques

Personnaliser un graphique, c'est ajuster son apparence globale pour le rendre plus informatif et attrayant. Nous verrons trois moyens de personnalisation : (i) la modification du thème, (ii) les manipulations de la légende, et (iii) l'ajout des annotations.

#### 3.1 Thème

Le thème d'un graphique est l'ensemble des éléments visuels qui le composent, tels que l'affichage de l'arrière-plan, les couleurs des tracés ou même la taille des polices. Commençons par l'affichage de l'arrière-plan. Reprenons le diagramme à barres, et modifions son thème par défaut.

```
ggplot(data = diamonds, mapping = aes(x = color,
                                         fill = cut)) +
  labs(title = "Graphique 14 : Bar plot",
       x = "Couleur", y = "Frequence") +
  geom_bar() +
  scale_y_continuous(limits = c(0,12000), n.breaks = 7) +
  theme_bw()
```

Graphique 14 : Bar plot

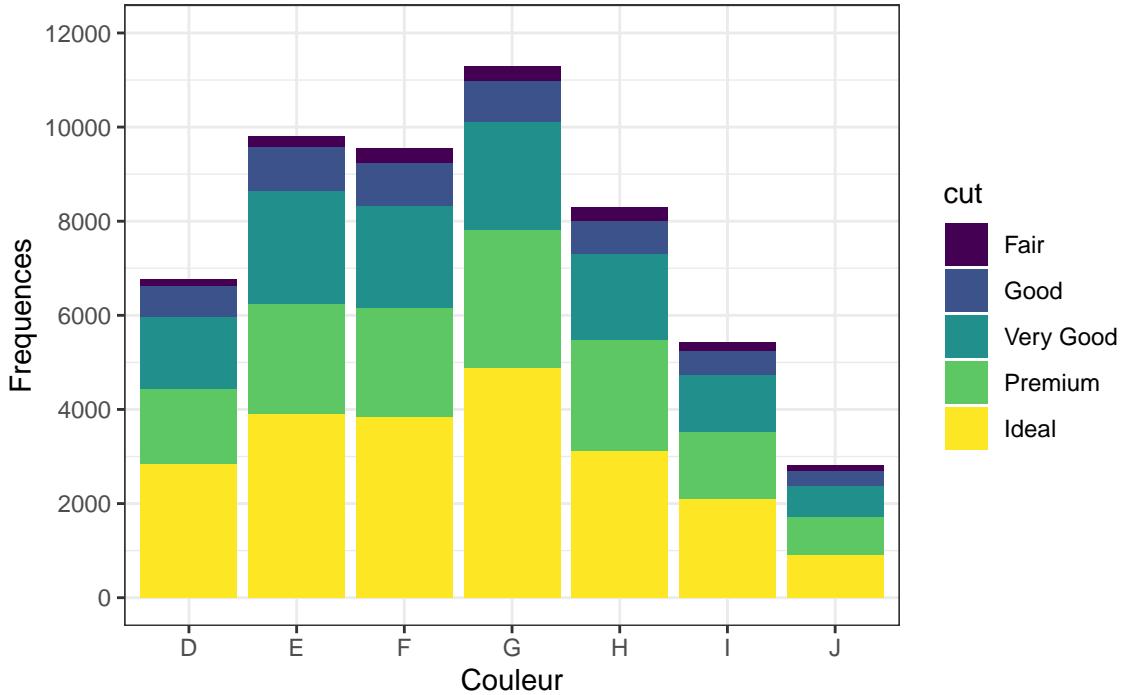


Table 1: Quelques thèmes fréquents

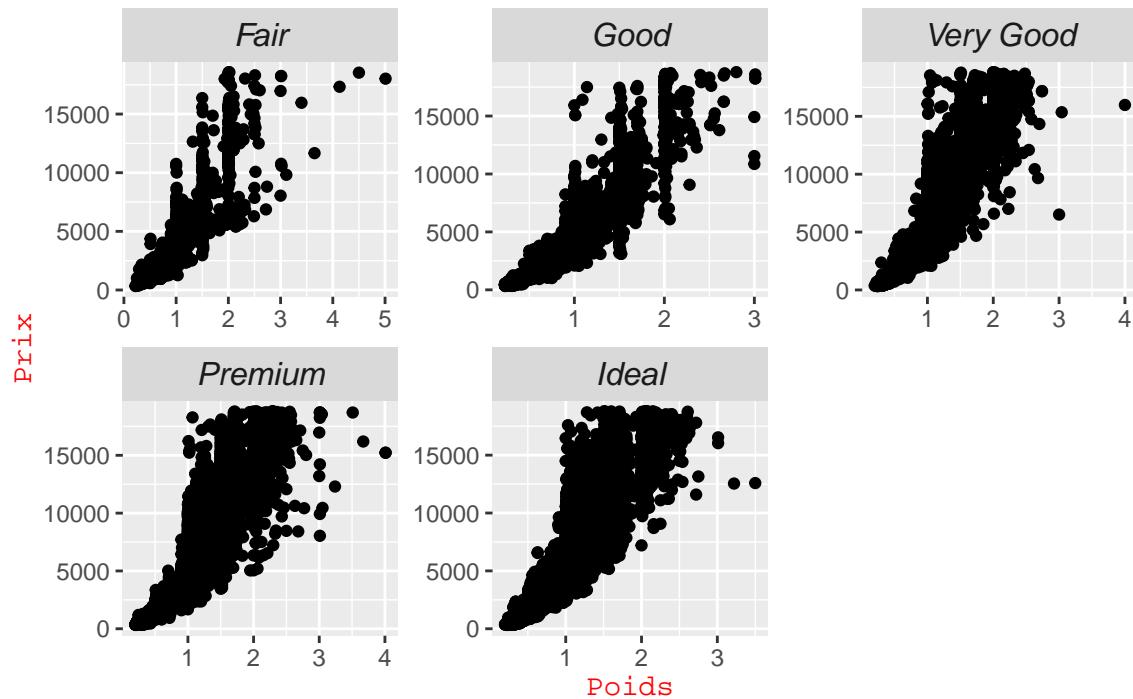
<code>theme_grey()</code>	fond gris (thème par défaut).
<code>theme_bw()</code>	fond blanc et encadré noir.
<code>theme_classic()</code>	thème sans grille d'arrière-plan.
<code>theme_light()</code>	fond clair avec grille grise.
<code>theme_minimal()</code>	thème avec grille, mais sans encadré.

Au-delà de l'affichage de l'arrière-plan, il existe plusieurs autres éléments constitutifs d'un thème permettant de personnaliser l'apparence du graphique. Considérons le cas des éléments textuelles. Reprenons le nuage de points en facettement d'enveloppe, et modifions l'apparence des titres. Le titre du graphique sera centré et spécifié en gras. Les titres des panneaux seront spécifiés en italique et de taille 12. Enfin, nous modifierons aussi la couleur des titres des axes ainsi que leurs polices d'écriture.

```
ggplot(data = diamonds, mapping = aes(x = carat,
                                         y = price)) +
  labs(title = "Graphique 15 : Scatter plot",
       x = "Poids", y = "Prix") +
  geom_point() +
  facet_wrap(cut ~ ., nrow = 2, ncol = 3, scales = "free") +
  theme(plot.title = element_text(face = "bold",
                                   hjust = 0.5),
        strip.text = element_text(face = "italic",
                                  size = 12),
```

```
axis.title = element_text(color = "red",
                           family = windowsFont("mono")))
```

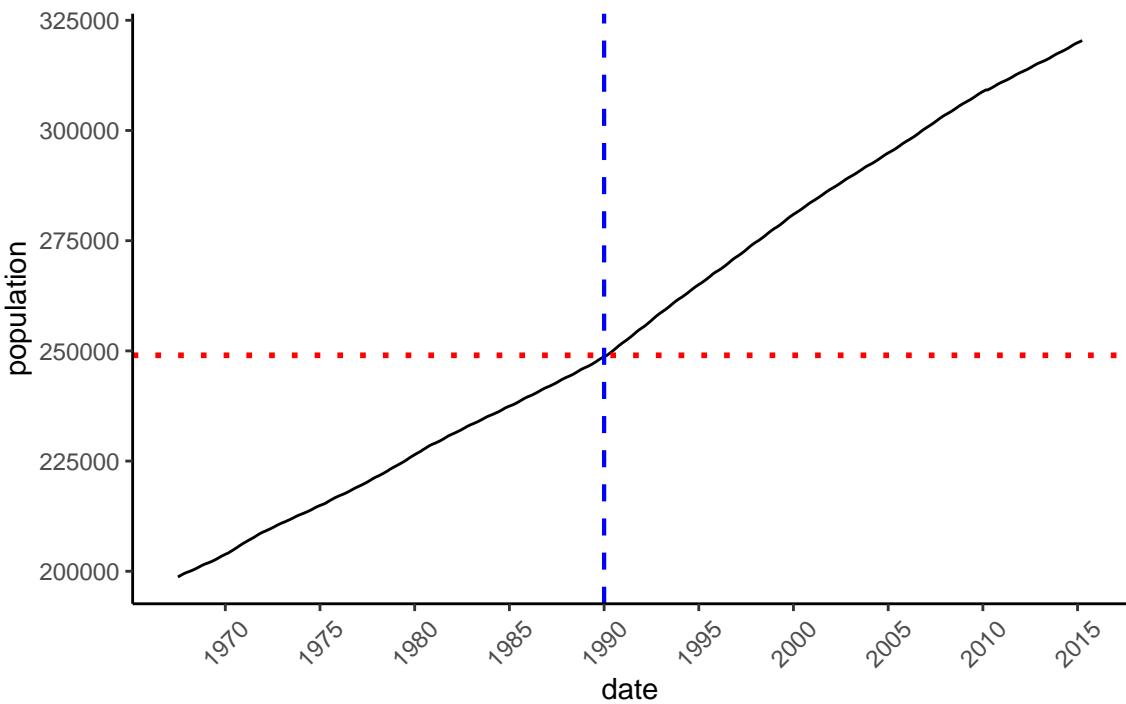
**Graphique 15 : Scatter plot**



Clôturons l'étude des éléments constitutifs d'un thème en prenant cette fois-ci le cas du graphique en courbes. Le titre du graphique sera centré et mis en gras. Les étiquettes de l'axe des abscisses subiront une rotation à 45° et une justification verticale. De surcroît, nous avons ajouté des lignes de référence horizontale et verticale sur le tracé afin de mettre en évidence des valeurs précises sur les deux axes.

```
ggplot(data = economics, mapping = aes(x = date, y = pop)) +
  labs(title = "Graphique 16 : Line plot",
       y = "population") +
  geom_line() +
  geom_hline(yintercept = 249000, linetype = "dotted",
             colour = "red", size = 1) +
  geom_vline(xintercept = as.Date("1990/01/01"),
             linetype = "dashed", colour = "blue", size = .75) +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y") +
  theme_classic() +
  theme(plot.title = element_text(face = "bold", hjust = 0.5),
        axis.text.x = element_text(angle = 45, vjust = 0.5))
```

**Graphique 16 : Line plot**

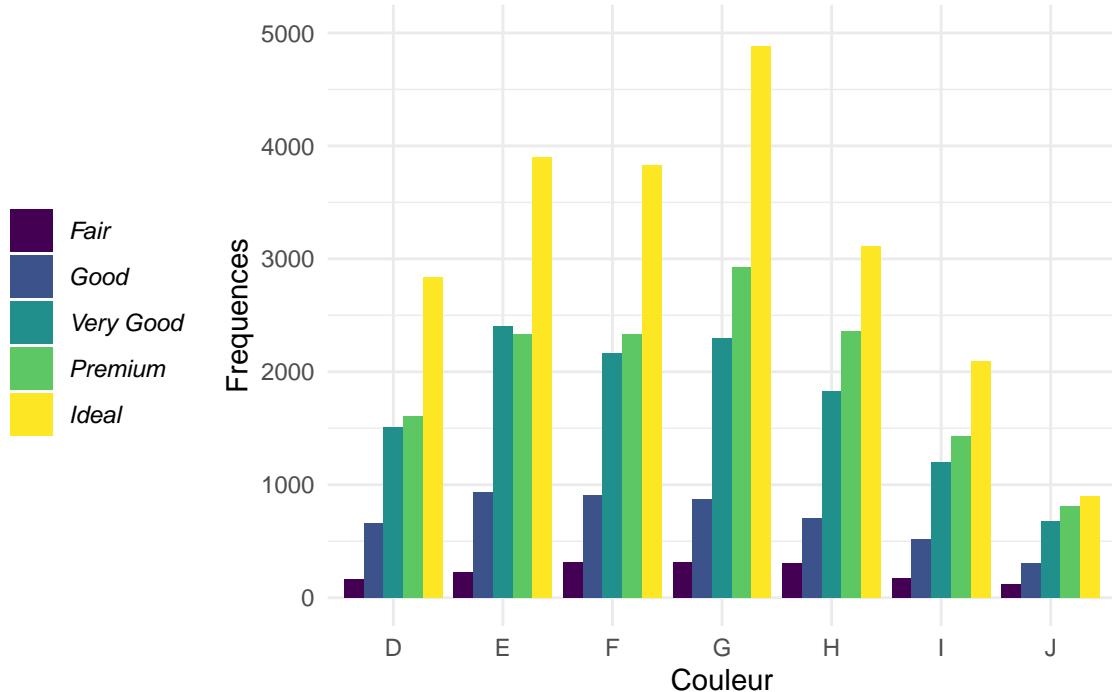


### 3.2 Légende

Le package ggplot2 génère automatiquement la légende des graphiques. Cependant, par souci de personnalisation, l'on peut être amené à manipuler le titre, les étiquettes ou l'arrière-plan d'une légende. Reprenons notre diagramme à barres multiples. Le titre de la légende sera masqué, les étiquettes spécifiées en italique, et la légende positionnée à gauche.

```
ggplot(data = diamonds, mapping = aes(x = color,
                                         fill = cut)) +
  labs(title = "Graphique 17 : Bar plot",
       x = "Couleur", y = "Fréquences") +
  geom_bar(position = "dodge") +
  scale_y_continuous(limits = c(0,5000), n.breaks = 5) +
  scale_x_discrete(position = "bottom") +
  theme_minimal() +
  theme(plot.title = element_text(face = "bold", hjust = 0.5),
        legend.title = element_blank(),
        legend.text = element_text(face = "italic"),
        legend.position = "left")
```

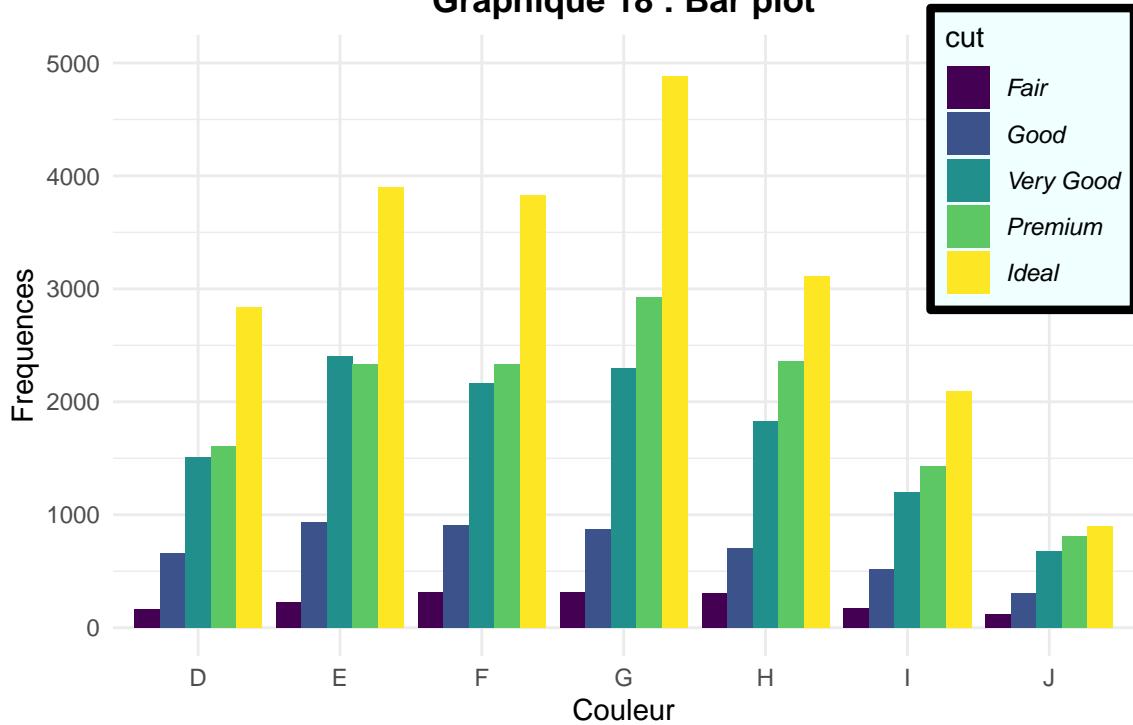
**Graphique 17 : Bar plot**



Réaménageons l'apparence de la légende en modifiant son arrière-plan par remplissage d'une couleur, et en définissant sa position à l'intérieur de la zone du tracé. En effet, la légende peut aussi être placée dans la zone du tracé, plutôt qu'à la marge. Pour ce faire, il faut renseigner à l'argument `legend.position` de la fonction `theme()` un vecteur de taille 2 contenant les coordonnées numériques du positionnement souhaité.

```
ggplot(data = diamonds, mapping = aes(x = color,
                                         fill = cut)) +
  labs(title = "Graphique 18 : Bar plot",
       x = "Couleur", y = "Frequences") +
  geom_bar(position = "dodge") +
  scale_y_continuous(limits = c(0,5000), n.breaks = 5) +
  scale_x_discrete(position = "bottom") +
  theme_minimal() +
  theme(plot.title = element_text(face = "bold", hjust = 0.5),
        legend.text = element_text(face = "italic"),
        legend.background = element_rect(fill = "azure",
                                         linetype = "solid",
                                         size = 1.5),
        legend.position = c(0.9,0.8))
```

**Graphique 18 : Bar plot**

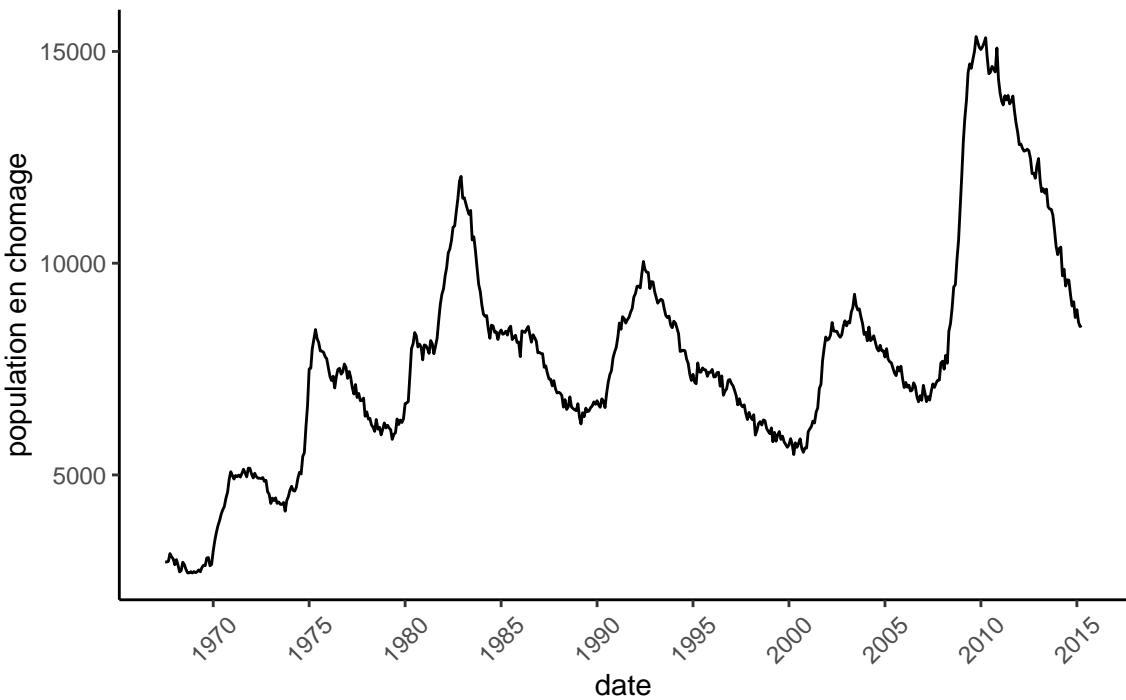


### 3.3 Annotations

Annoter un graphique permet de fournir des informations supplémentaires pour sa bonne lecture. Cette opération consiste à ajouter du texte sur la zone du tracé ou tout autre objet (flèches, bandes, etc.) à un endroit précis sur le graphique. Pour y parvenir, on utilise la fonction `annotate()`. Considérons l'évolution de la population en chômage aux USA (*cf.* dataset `economics`).

```
ggplot(data = economics, mapping = aes(x = date, y = unemploy)) +
  labs(title = "Graphique 19 : Line plot",
       y = "population en chomage") +
  geom_line() +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y") +
  scale_y_continuous(n.breaks = 4) +
  theme_classic() +
  theme(plot.title = element_text(face = "bold", hjust = 0.5),
        axis.text.x = element_text(angle = 45, vjust = 0.5))
```

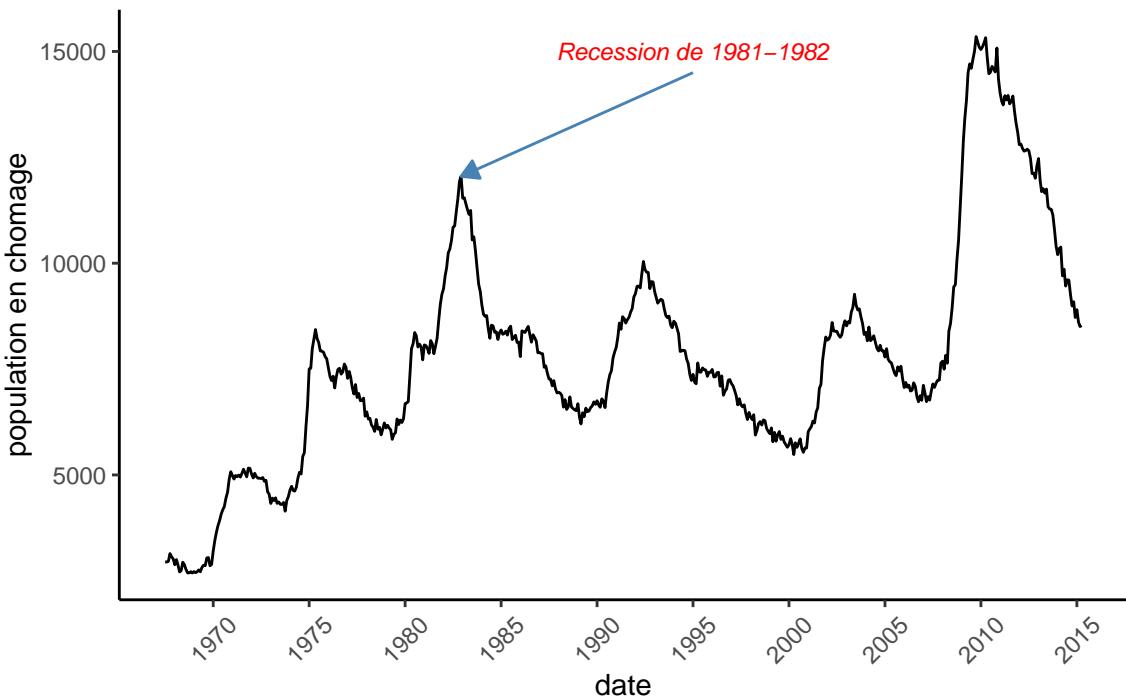
**Graphique 19 : Line plot**



Mettons en évidence le pic du chômage enregistrée en 1981-1982 à cause de la période de récession qu'a connue l'économie américaine. Nous allons ajouter du texte sur le graphique puis relier ce texte avec le pic sur la courbe au moyen d'une flèche. La taille de la flèche est gérée avec l'argument `arrow`.

```
ggplot(data = economics, mapping = aes(x = date, y = unemploy)) +
  labs(title = "Graphique 20 : Line plot",
       y = "population en chomage") +
  geom_line() +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y") +
  scale_y_continuous(n.breaks = 4) +
  theme_classic() +
  theme(plot.title = element_text(face = "bold", hjust = 0.5),
        axis.text.x = element_text(angle = 45, vjust = 0.5)) +
  annotate(geom = "text", x = as.Date("1995/01/01"), y = 15000,
           colour = "red", fontface = "italic", size = 3,
           label = "Recession de 1981-1982") +
  annotate(geom = "segment", x = as.Date("1995/01/01"),
           xend = as.Date("1982/12/01"), y = 14500, yend = 12051,
           colour = "steelblue",
           arrow=arrow(type="closed", length=unit(.1, "inches")))
```

**Graphique 20 : Line plot**

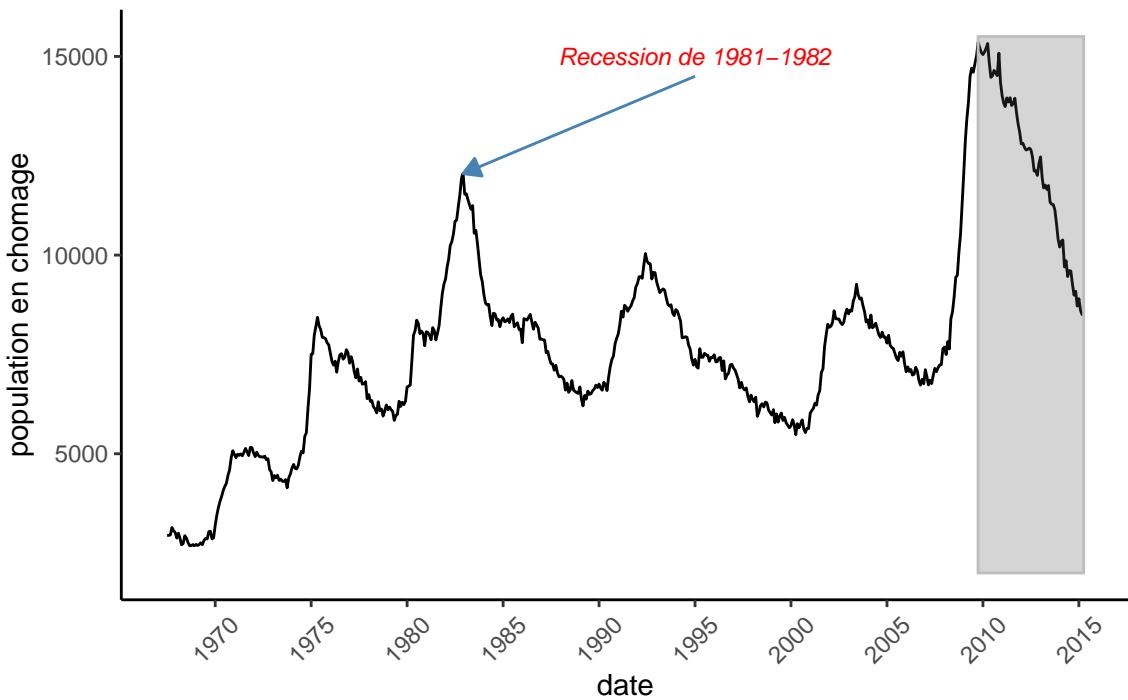


Exhibons à présent la période allant du dernier trimestre 2009 au premier trimestre 2015 afin de mettre en évidence la baisse du nombre de personnes en chômage due à la reprise économique durant cette période. La transparence de la zone en surbrillance est gérée avec l'argument `alpha`.

```
ggplot(data = economics, mapping = aes(x = date, y = unemploy)) +
  labs(title = "Graphique 21 : Line plot",
       y = "population en chomage") +
  geom_line() +
  scale_x_date(date_breaks = "5 years", date_labels = "%Y") +
  scale_y_continuous(n.breaks = 4) +
  theme_classic() +
  theme(plot.title = element_text(face = "bold", hjust = 0.5),
        axis.text.x = element_text(angle = 45, vjust = 0.5)) +
  annotate(geom = "text", x = as.Date("1995/01/01"), y = 15000,
           colour = "red", fontface = "italic", size = 3,
           label = "Recession de 1981-1982") +
  annotate(geom = "segment", x = as.Date("1995/01/01"),
           xend = as.Date("1982/12/01"), y = 14500, yend = 12051,
           colour = "steelblue",
           arrow=arrow(type="closed", length=unit(.1,"inches")))+
```

annotate(geom = "rect", xmin = as.Date("2009/10/01"),
 xmax = as.Date("2015/04/01"), alpha = .25,
 ymin = 2000, ymax = 15500, colour = "gray")

**Graphique 21 : Line plot**



Nous clôturons cette sous-section consacrée l'annotation des graphiques avec l'apprentissage d'un package permettant de gérer efficacement les étiquettes des données sur le graphique. Il s'agit du package `ggrepel`. Considérons pour cela le dataset `mtcars`. Ce dataset se rapporte à la consommation de carburant ainsi que dix autres aspects de la conception et de la performance des automobiles pour 32 véhicules. Étudions la relation entre la consommation en carburant et le poids des véhicules, tout en ajoutant les noms des véhicules comme étiquettes des données sur le nuage.

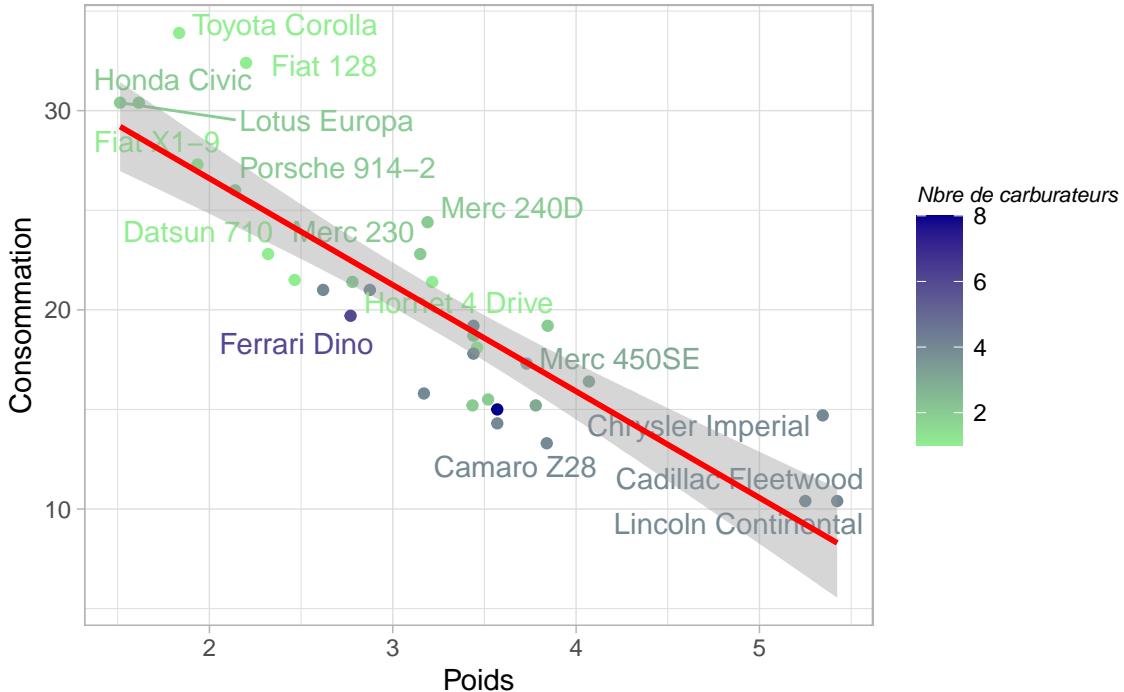
```
library(ggrepel)

mtcars <- mtcars

attach(mtcars)

ggplot(mtcars, aes(wt, mpg, colour = carb)) +
  geom_point() + geom_text_repel(label=row.names(mtcars)) +
  geom_smooth(method = "lm", se = TRUE, colour = "red") +
  labs(title = "Graphique 22 : Scatter plot",
       x = "Poids", y = "Consommation",
       colour = "Nbre de carburateurs") +
  scale_color_gradient(low="lightgreen", high="darkblue") +
  theme_light() +
  theme(plot.title = element_text(face = "bold", hjust = .5)) +
  theme(legend.title = element_text(face = "italic", size = 8))
```

**Graphique 22 : Scatter plot**



Pour optimiser la position des étiquettes des données, nous avons utilisé la fonction `geom_text_repel()` du package `ggrepel`, laquelle permet d'éviter le chevauchement des textes sur le graphique. Ensuite, nous avons colorié les points du nuage suivant le nombre de carburateurs que comportent les véhicules. À cet effet, la fonction `scale_color_gradient()` nous a permis de générer l'échelle des couleurs. Enfin, nous avons ajouté sur le graphique la droite de la régression linéaire décrivant la relation entre les deux variables représentées. La fonction `geom_smooth()` a été mobilisé à cette fin, et les intervalles de confiance autour de la droite sont aussi affichés.

## 4 Astuces supplémentaires

### 4.1 Graphiques interactifs

Un graphique interactif est celui qui permet à l'utilisateur d'interagir avec pour en savoir plus sur les données qui y sont représentées. Cette interaction peut être faite en sélectionnant des points, des lignes ou des zones sur le graphique, voire en zoomant ou en déplaçant le graphique. Les graphiques interactifs sont un outil précieux pour la visualisation des données car ils permettent aux utilisateurs de découvrir des informations qui ne seraient pas visibles à première vue.

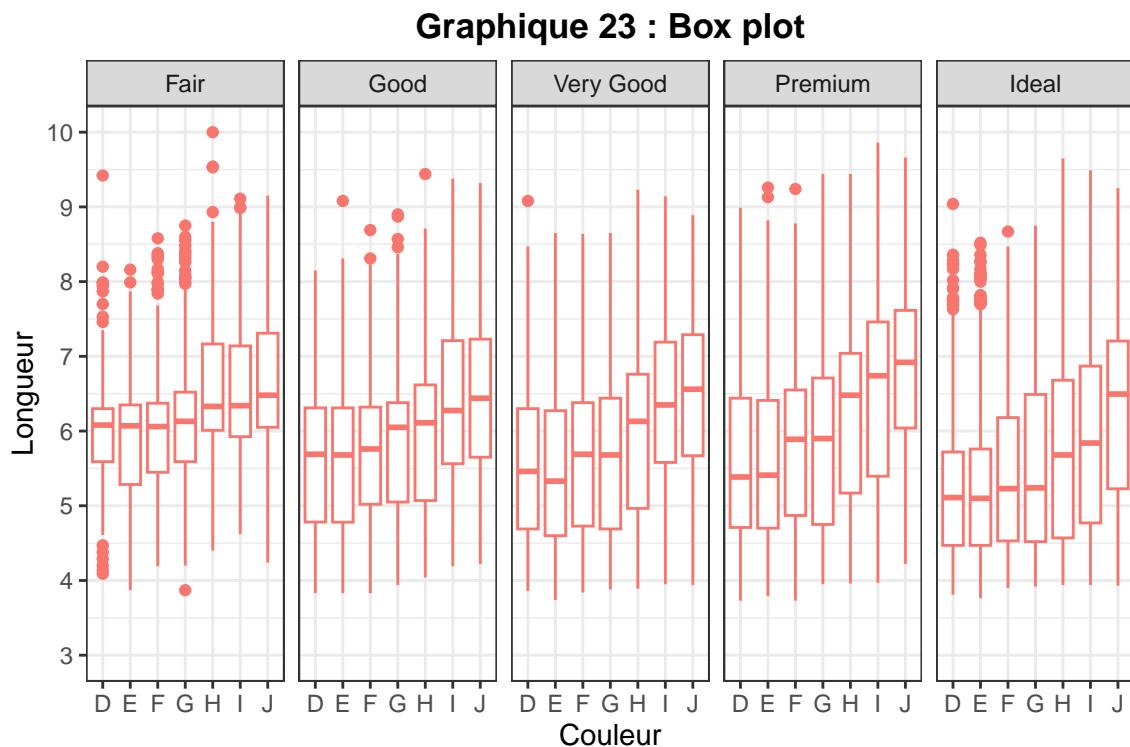
En R, les graphiques interactifs peuvent être créés à l'aide du package `plotly`. C'est un package très puissant et facile à utiliser permettant de créer des graphiques interactifs susceptibles d'être consultés via un navigateur web. Pour illustrer, reprennons le dataset `diamonds`. Représentions la distribution de la longueur des diamants selon leur couleur, et

par qualité, au moyen des boîtes à moustache.

```
library(plotly)

chart <- ggplot(diamonds, aes(x = color, y = length)) +
  geom_boxplot(aes(colour = "greenyellow",
                   outlier.colour = "deepskyblue4")) +
  labs(title = "Graphique 23 : Box plot",
       x = "Couleur", y = "Longueur") +
  scale_y_continuous(limits = c(3,10), n.breaks = 8) +
  guides(colour = "none") +
  facet_grid(.~cut) + theme_bw() +
  theme(plot.title = element_text(face = "bold", hjust = .5))

ggplotly(chart)
```



## 4.2 Combinaison des graphiques

L'opération de combinaison consiste à mixer sur une même interface graphique plusieurs figures de nature diverse. Elle permettrait donc d'aboutir à une espèce de tableau de bord statique plus complet et plus informatif sur le dataset sous examen. Cette opération est réalisée en R au moyen de la fonction `grid.arrange()` du package `gridExtra`. Reprenons le cas des données sur les diamants, et combinons sur une même interface le nuage de points, le diagramme à barres et la boîte à moustache.

```

library(gridExtra)

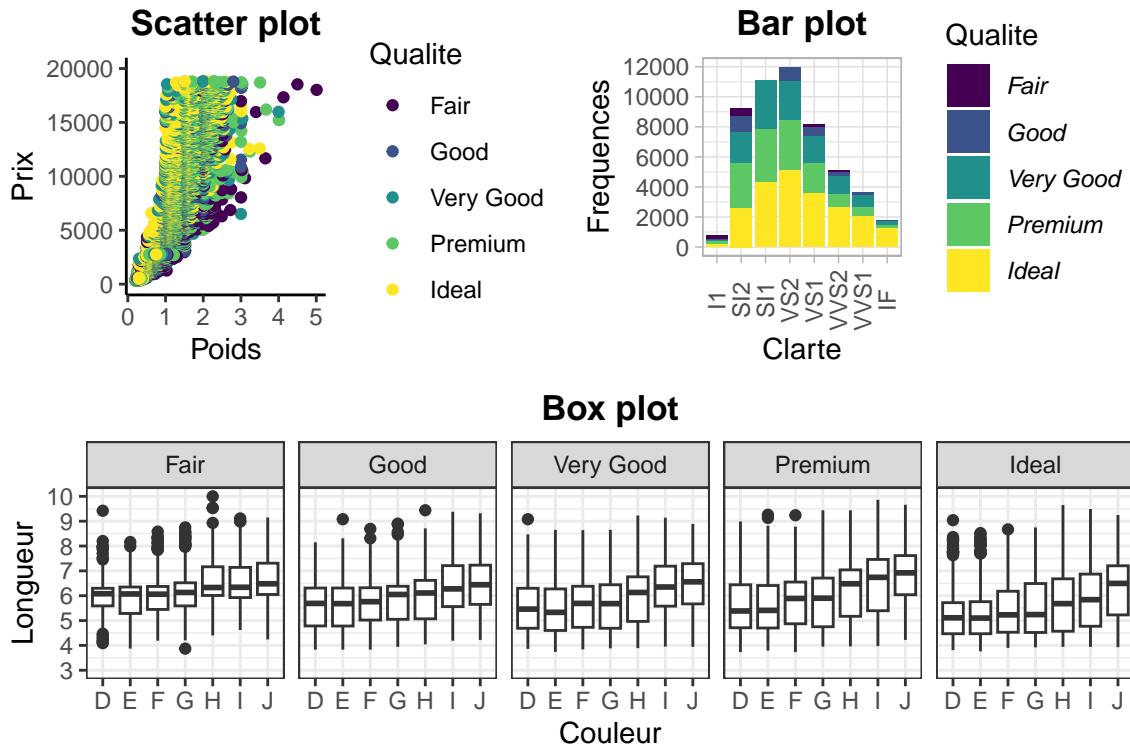
chart1 <- ggplot(data=diamonds, mapping=aes(x=carat,
                                              y = price,
                                              colour = cut)) +
  labs(title = "Scatter plot", x = "Poids", y = "Prix",
       colour = "Qualite") +
  geom_point() + theme_classic() +
  scale_y_continuous(n.breaks = 5, limits = c(0,20000)) +
  theme(plot.title = element_text(face = "bold", hjust = .5))

chart2 <- ggplot(data=diamonds, mapping=aes(x=clarity, fill=cut)) +
  labs(title = "Bar plot", x = "Clarte", y = "Frequences",
       fill = "Qualite") +
  geom_bar() + theme_light() +
  scale_y_continuous(limits = c(0,12000), n.breaks = 7) +
  theme(plot.title = element_text(face = "bold", hjust = .5),
        legend.text = element_text(face = "italic"),
        axis.text.x = element_text(angle = 90, vjust = .5))

chart3 <- ggplot(diamonds, aes(x = color, y = length)) +
  geom_boxplot() + theme_bw() +
  labs(title = "Box plot", x = "Couleur", y = "Longueur") +
  scale_y_continuous(limits = c(3,10), n.breaks = 8) +
  facet_grid(.~cut, scales = "free") +
  theme(plot.title = element_text(face = "bold", hjust = .5))

graphs <- list(chart1, chart2, chart3)
grids <- matrix(c(1,2,3,3), nrow = 2, ncol = 2, byrow = TRUE)
grid.arrange(grobs = graphs, layout_matrix = grids)

```



Dans un premier temps, il faut stocker les graphiques créés dans un objet de type “*liste*”. Ensuite, il faut découper les zones de l’interface graphique sous un format matriciel. Enfin, l’on peut combiner tous les graphiques sur la même interface selon le format préalablement défini grâce à l’argument `layout_matrix` de la fonction `grid.arrange()`.

### 4.3 Sauvegarde des graphiques

Pour enregistrer les graphiques dans le répertoire de travail, il faut utiliser la fonction `ggsave()`. Les graphiques peuvent être sauvegardés sous les formats .jpeg, .png ou .pdf.

```
# Format .jpeg
ggsave(path = "E:/Data_Science/RStudio/GitHub/",
       filename = "scatterplot.jpeg", plot = chart1)

# Format .png
ggsave(path = "E:/Data_Science/RStudio/GitHub/",
       filename = "barplot.png", plot = chart2)

# Format .pdf
ggsave(path = "E:/Data_Science/RStudio/GitHub/",
       filename = "boxplot.pdf", plot = chart3)
```