

session_4_pdf

Julian Maitra

Session 4

La session 4 a pour but d'introduire trois éléments fondamentaux en analyse de données avec R :

1. Installer et charger des **paquets R** sur son ordinateur
2. Utiliser des **jeux de données intégrés en R** pour s'entraîner en analyse de données
3. Apprendre à utiliser le paquet **ggplot2** pour créer des visualisations de données

Les paquets R

À noter

Dans la programmation R, un **paquet** (angl. *package*) est un ensemble de fonctions, de données et de code compilé dans un format bien défini.

Les paquets étendent les fonctionnalités de R de base en fournissant des outils supplémentaires pour l'analyse des données, la visualisation, l'apprentissage automatique, la modélisation statistique et beaucoup d'autres applications spécialisées.

Les paquets R peuvent être développés par n'importe qui et partagés gratuitement avec d'autres (open source). Par conséquent, une grande multitude de paquets R sont disponibles.

Notez que la plupart des paquets R doivent être installés séparément, normalement depuis le site [Comprehensive R Archive Network \(CRAN\)](#). Dans ce cours, nous nous concentrerons sur deux paquets célèbres :

- **ggplot2** pour créer des **visualisations de données**
- **dplyr** pour **manipuler les jeux de données**

Tous deux font partie du `tidyverse`, une collection de paquets conçus pour la science des données. Nous utiliserons également le paquet suivant :

- `ggthemes` pour **augmenter le design des graphiques**

Pour utiliser un paquet R sur votre ordinateur, il faut :

1. D'abord **installer** le paquet en question avec la fonction `install.packages()`. Cela peut durer quelques minutes.
2. Ensuite **activer** le paquet avec la fonction `library()`.

Astuce

Dans RStudio, on peut aussi installer les paquets via l'onglet **Tools** et le menu déroulant : Cliquez sur *Install Packages...* et cherchez les paquets que vous voulez installer.

Utilisez le code suivant pour installer les paquets R requis pour notre cours. Enlevez les croisillons, exécutez le code avec `install.packages()` puis réajouter les croisillons pour éviter de réinstaller les paquets chaque fois que vous exécutez le code dans votre script ! Activer ensuite les paquets avec `library()`.

```
# install.packages("tidyverse")
# install.packages("ggthemes")

library(tidyverse)
library(ggthemes)
```

À noter : `ggplot2` et `dplyr` sont installés avec le `tidyverse`, il ne faut donc plus les installer séparément.

Attention

Notez qu'une fois installé, vous n'avez pas besoin de le réinstaller les paquets à chaque fois avec `install.packages()`. Cependant, vous devez **activer** tous les paquets R que vous voulez utiliser dans une session avec la commande `library()`. Sans activation, vous allez recevoir des messages d'erreur.

Certains paquets ont des documentations auxquelles vous pouvez accéder grâce au point d'interrogation. Ces infos sont affichés dans le volet *Output* sous l'onglet *Help*.

- `?ggplot2`
- `?dplyr`

En ligne, vous pouvez également trouver de nombreuses ressources qui expliquent en détail les fonctionnalités des différents paquets R et vous renvoient vers d'autres livres, tutoriels, cours, etc., p. ex. :

- Documentation officielle de [ggplot2](#)
- Documentation officielle de [dplyr](#)
- Exemples de graphiques créés avec les différents designs de [ggthemes](#)

Vous pouvez vérifier quels paquets sont **actuellement installés sur votre ordinateur** avec la fonction `installed.packages()` (vous pouvez laisser vide l'espace entre parenthèses) :

```
installed.packages()
```

Les jeux de données intégrés

Une excellente façon, et probablement la plus efficace, d'apprendre à utiliser R pour l'analyse de données consiste à travailler avec des **jeux de données intégrés dans R**.

Ces jeux de données exemplaires présentent deux avantages principaux.

1. Il est facile d'y accéder et de les charger dans votre session RStudio, car ils sont intégrés à R et à certains de ses paquets.

Cela les distingue fortement des données du “monde réel”, comme les données de questionnaires ou de médias sociaux, qu'il faut souvent d'abord nettoyer et formater, puis importer dans RStudio sous forme de fichiers externes. Avec les jeux de données intégrés, vous pouvez immédiatement commencer l'analyse et la visualisation des données.

2. Ces jeux de données ont prouvé leur utilité dans d'innombrables supports d'apprentissage.

Par conséquent, vous pouvez facilement trouver des ressources supplémentaires et des exemples de code à leur sujet.

Dans ce qui suit, nous allons explorer quatre jeux de données classiques de ce type :

- `iris`
- `mtcars`
- `diamonds`
- `palmerpenguins`

Astuce

Vous pouvez afficher les jeux de données qui sont actuellement accessible sur votre ordinateur avec la commande `data()`.

le jeu de données `iris`

Le jeu de données `iris` est un ensemble de données célèbre qui contient les mesures en centimètres des variables longueur et largeur du sépale, ainsi que longueur et largeur du pétale, pour trois espèces de fleurs d'iris (Setosa, Virginica et Versicolor). Il y a 150 observations en total (50 pour chaque espèce).

Le jeu de données a été rendu célèbre par le statisticien et biologiste britannique **Ronald Fisher** dans un article scientifique de 1936 qui avait beaucoup d'influence dans le domaine des statistiques. Les données d'iris eux-mêmes ont été récolté par le botaniste Edgar Anderson en 1935 sur la péninsule de Gaspésie au Québec, Canada.

Plus sur Ronald Fisher

Ronald Fisher (1890-1962) était un statisticien, généticien et biologiste britannique qui a joué un rôle crucial dans le développement de la science statistique moderne. Il est connu pour son travail dans le développement des fondements des méthodes statistiques, incluant l'analyse de variance (ANOVA), le test exact de Fisher et l'estimation du maximum likelihood. Fisher a également apporté des contributions significatives en génétique, où son travail sur la théorie de la sélection naturelle et la génétique des populations a aidé à combler le fossé entre la génétique mendélienne et l'évolution darwinienne. Il est considéré comme l'un des architectes principaux de la synthèse néo-darwinienne. Son influence s'étend au-delà de la statistique et de la génétique aux domaines plus larges de la biologie et de la recherche agricole, faisant de lui une figure pivotale de la science du 20e siècle.

Dans l'apprentissage de R, ce jeu de données est souvent utilisé pour illustrer diverses techniques d'analyse de données, y compris le clustering, la classification et la visualisation.

Explorons un peu le jeu de données `iris` avec du code !

```
dim(iris) # Ceci affiche le nombre de lignes et de colonnes

head(iris) # Affiche les six premières lignes

str(iris) # Informations sur la structure du jeu de données

names(iris) # Affiche les noms des colonnes
```

Regardons ce que ça donne après l'exécution de chaque ligne de code :

```
dim(iris)
```

```
[1] 150 5
```

```
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

```
str(iris)
```

```
'data.frame': 150 obs. of 5 variables:  
 $ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...  
 $ Sepal.Width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...  
 $ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...  
 $ Petal.Width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...  
 $ Species      : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 1 1 1 ...
```

```
names(iris)
```

```
[1] "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width" "Species"
```

Astuce

Vous pouvez consulter la documentation pour beaucoup de jeux de données avec un point d'interrogation : `?iris()`.

le jeu de données mtcars

Utilisez le script suivant pour analyser `mtcars`(lisez également la documentation avec `?mtcars`) !

```

# Chargez le jeu de données mtcars. Il devrait apparaître dans l'environnement
data(mtcars)

# Affichez mtcars
mtcars

# vous pouvez explorer mtcars manuellement en double cliquant dessus dans l'environnement

# Affichez seulement les six premières observations
head(mtcars)

# Affichez seulement les six dernières observations
tail(mtcars)

# Appliquez quelques autres commandes pour explorer ce jeu de données
class(mtcars)
dim(mtcars)

str(mtcars)
summary(mtcars)

glimpse(mtcars) # cette commande marche seulement si vous avez activé le tidyverse

# Extraire la colonne mpg ("miles per gallon") avec le signe $
# et calculer la moyenne, la médiane, le minimum et le maximum

mtcars$mpg

mean(mtcars$mpg)

median(mtcars$mpg)

min(mtcars$mpg)

max(mtcars$mpg)

```

Un des modèles de mtcars : Le Toyota Corolla de 1974. [Source : wikimedia.org](#)

le jeu de données diamonds

C'est maintenant à vous d'explorer ce jeu de données avec du code !

Utilisez également `?diamonds()` !

Qu'est-ce qui définit le prix d'un diamant ? Source : [wikimedia.org](https://commons.wikimedia.org)

Introduction aux visualisations avec ggplot2

i À noter

`ggplot2`, qui fait partie du `tidyverse`, est un paquet de visualisation de données très populaire dans R. Il a été conçu pour faciliter la création de graphiques complexes et multicouches avec une syntaxe cohérente et consistante nommée *grammar of graphics*. `ggplot2` est donc un acronyme formé à partir des mots *grammar of graphics plots* (français : *grammaire des graphiques*). Le “2” dans `ggplot2` indique simplement la deuxième génération de ce paquet R.

Avec `ggplot2`, les utilisateurs peuvent construire des graphiques de manière incrémentale en ajoutant des couches, ce qui le rend flexible pour une large gamme de besoins graphiques. Cela permet de créer des visualisations hautement personnalisables et esthétiquement attrayantes, allant de simples diagrammes de dispersion à des figures multi-panneaux complexes.

Le tableau suivant décrit les couches d'un graphique ggplot :

Table 1: Les couches d'un graphique ggplot

Couche	Déscription/Variables	Obligatoire ?
<code>data</code>	le jeu de données que nous voulons visualiser	oui
<code>aesthetics</code>	le <i>mapping</i> de nos variables : axes <i>x</i> et <i>y</i> , <i>color</i> , <i>fill</i> , <i>alpha</i> , <i>line width</i> , etc.	oui
<code>geometries</code>	le type visuel du graphique : <i>point</i> , <i>line</i> , <i>histogram</i> , <i>bar</i> , <i>boxplot</i>	oui
<code>themes</code>	Le design du graphique	non
Autres couches optionnelles : <i>facets</i> , <i>statistics</i> , <i>coordinates</i>	Pour des visualisations plus complexes	non

Pour comprendre comment cela fonctionne, voici un exemple de code dans lequel les différentes couches d'un graphique `ggplot` sont ajoutées étape par étape dans la fonction `ggplot()`. (Normalement, nous ne procéderions pas ainsi, mais utiliserions directement le code en entier pour créer le graphique).

Le but dans cet exemple est de visualiser le jeu de données `diamonds` pour comprendre la relation entre les variables `price` (prix) et `carat` (poids) des diamants dans le jeu de données !

i À noter

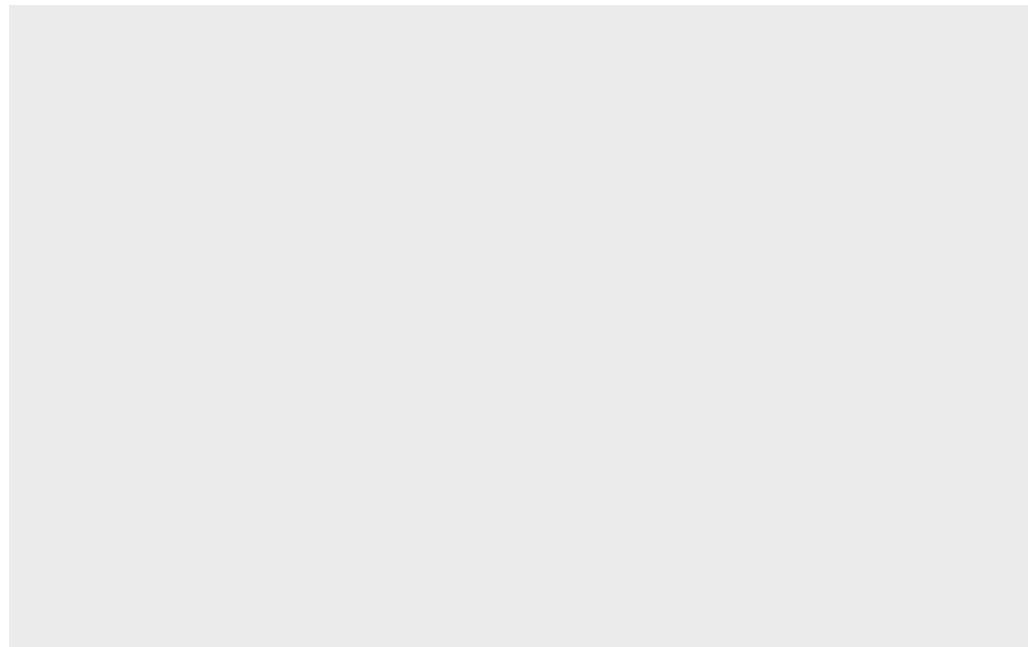
Important : avant d'analyser un ensemble de données, il faut toujours acquérir des **connaissances contextuelles** sur ce que les données représentent.

Dans notre exemple, il s'agit de diamants. Dans le monde des diamants, il est important de comprendre qu'il existe **quatre critères de qualité** qui déterminent **la valeur** (donc le prix) d'un diamant. Ce sont les suivants :

- La taille (angl. *cut*)
- La pureté (*clarity*)
- La couleur (*color*)
- Le poids en carats (*carat*)

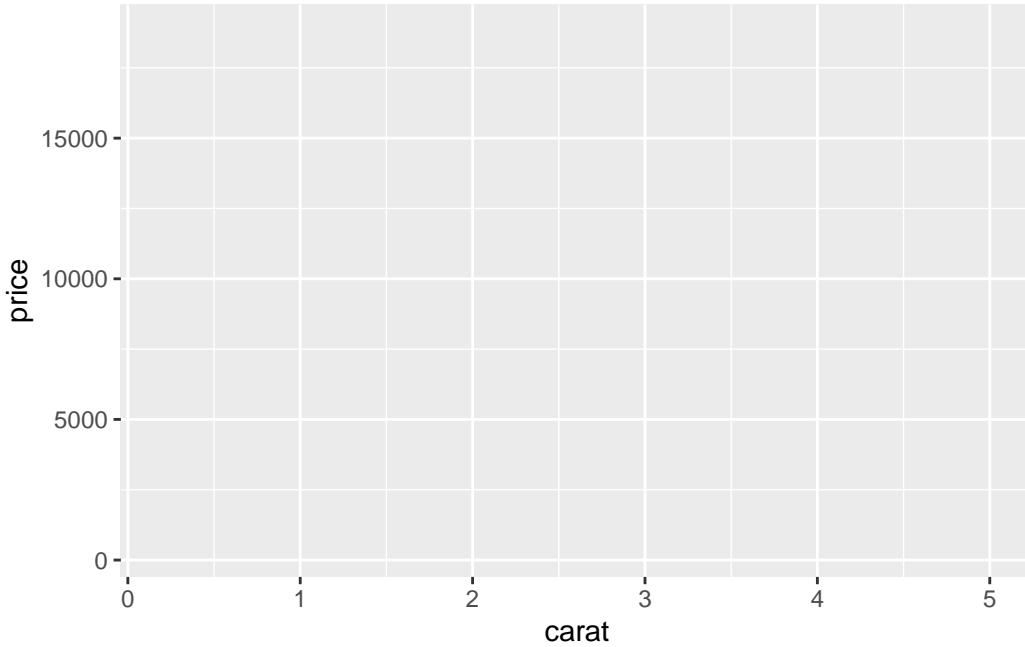
En anglais, on parle donc aussi des 4C. Le jeu de données `diamonds` contient précisément ces variables.

```
# La première couche, data, définit le jeu de données
ggplot(data = diamonds)
```



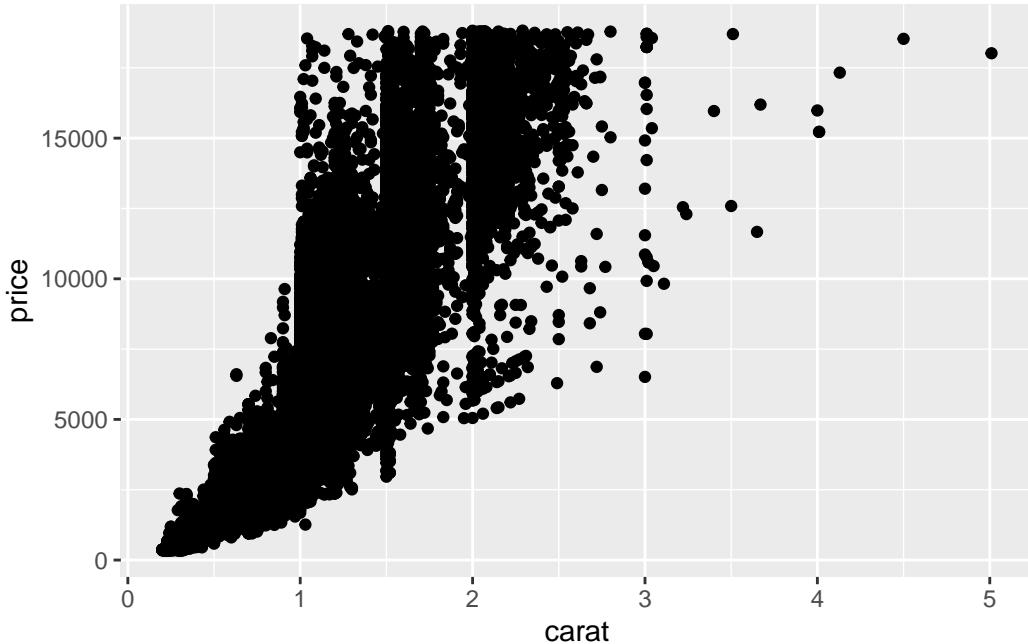
Notre graphique n'est pour l'instant qu'une zone grise ! Catastrophe ? Bien sûr que non .
Nous devons maintenant définir les autres paramètres de la fonction `ggplot()` !

```
# La deuxième couche, aesthetics, définit le mapping de nos variables sur les axes y et x
ggplot(data = diamonds,
       mapping = aes(x = carat, y = price))
```



La zone grise a maintenant reçu des unités pour ses axes x (`carat`) et y (`price`). Mais il n'y a toujours pas de données affichées !

```
# La troisième couche, geometries, définit le type visuel de notre graphique
ggplot(data = diamonds,
       mapping = aes(x = carat, y = price)) +
       geom_point()
```

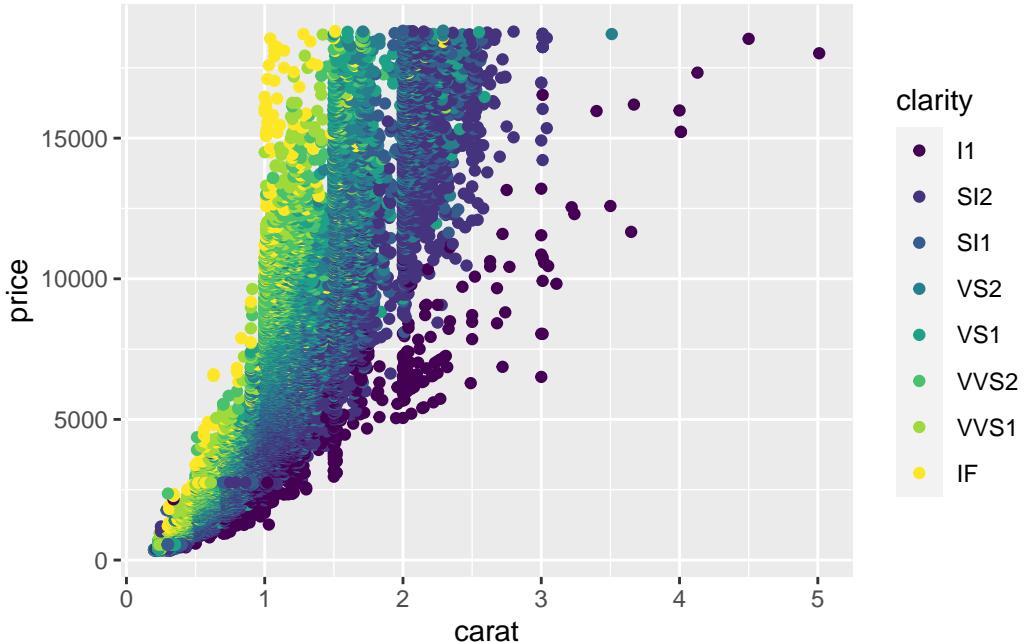


Ouf, si vous avez exécuté le dernier code, vous avez maintenant 53 940 points de données affichées dans votre graphique ! Car il y a tant d'observations dans le jeu de données **diamonds**.

Bon courage si vous avez un vieil ordinateur, cela peut prendre un certain temps avant que R ne produise le graphique...

Mais jetons un coup d'œil au graphique. Qu'est-ce qui saute aux yeux, pouvez-vous déjà identifier des tendances ?

```
# On peut maintenant encore optimiser notre graphique
ggplot(data = diamonds,
        mapping = aes(x = carat, y = price, color = clarity)) +
        geom_point()
```

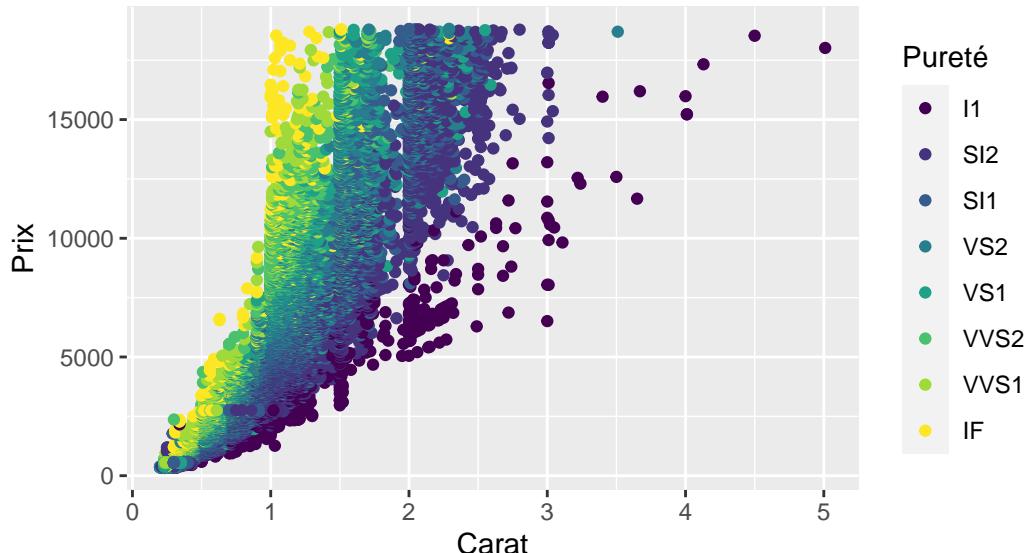


Vous avez remarqué ce qui a changé dans le code ?

```
# Ajoutons encore une couche optionnelle mais utile au graphique : labs() nous permet d'ajouter des informations supplémentaires
ggplot(data = diamonds,
       mapping = aes(x = carat, y = price, color = clarity)) +
  geom_point() +
  ggtitle("") +
  labs(title = "La relation entre prix et carat des diamants",
       subtitle = "n = 53 940 observations",
       y = "Prix",
       x = "Carat",
       colour = "Pureté")
```

La relation entre prix et carat des diamants

n = 53 940 observations



Maintenant il s'agit d'interpréter le graphique. Que nous dit-il sur la relation entre prix et carats des diamants ? Quel rôle joue la variable *pureté* (*clarity*)?

Devoir pour Session 5

- Exécutez toutes les sections de code de la session 4 sur votre ordinateur.
- Entraînez-vous à ouvrir un nouveau script R, à définir un répertoire de travail et à enregistrer le script dans un dossier sur votre ordinateur.
- Assurez-vous que le paquet R `tidyverse` est installé correctement sur votre ordinateur.
- Utilisez le code R de manière autonome pour explorer le jeu de données `diamonds`. Vous devez être en mesure de comprendre et d'expliquer comment il est structuré (nombre d'observations n, unité statistique, variables, origine des données, etc).