Titre

Prénom Nom

10 avril 2017

Contents

Introduction **2**

Comment ça marche ? . . . 4

Utilisations de RMarkdown . . . 4

Avantages . . . 4

Étapes pour Générer le Document **4**

Packages Nécessaires . . . 5

Pour le Format PDF . . . 5

Après l'Installation . . . 5

Éléments d'un document R Markdown **6**

En-tête (préambule) . . . 6

Texte du document . . . 6

Titres . . . 6

Emphase . . . 6

Listes . . . 6

Liens . . . 6

Images . . . 6

Citations . . . 6

Code . . . 6

Tableaux . . . 7

Lignes horizontales . . . 7

Sauts de ligne . . . 7

Texte préformaté . . . 7

Exemple d'analyse . . . 7

Préparation de l'Environnement . . . 7

Aperçu des Données . . . 8

Analyse de la Distribution des Prix . . . 8
Relation entre le Prix et la Qualité du Carat . . . 9
Analyse par Catégorie de Taille . . . 10
Corrélation entre les Caractéristiques . . . 11
Conclusion de l'Analyse . . . 11

Nom et Options **11**

Nom . . . 12
Options . . . 12

Rendu des tableaux **12**

Tableaux croisés kable() . . . 12
Tableaux croisés tbl_summary() . . . 13
Exemple 1: Résumé Basique . . . 14
Exemple 2: Résumé avec Statistiques Spécifiques . . . 15
Exemple 3: Résumé par Groupes . . . 15
Exemple 4: Ajout de Labels et Modification de l'Apparence . . . 16

Ecrire du langage R en ligne **17**

Exercices **18**

Introduction

L'extension `rmarkdown` est un outil puissant pour la création de documents dynamiques dans l'environnement de programmation R. Il permet d'intégrer du code R, des résultats, et des narrations dans un seul document. Voici une explication détaillée de ses caractéristiques et de son fonctionnement :

Les documents générés peuvent être au format HTML, PDF, Word, et bien d'autres[1]. C'est donc un outil très pratique pour l'exportation, la communication et la diffusion de résultats d'analyse.

```
knitr::opts_chunk$set(echo = TRUE, message = FALSE, warning = FALSE)
```

Le chunk (zone grise) juste au dessus est le chunk qui fixe les options pour l'ensemble du document. Les paramètres fixés dans ce chunk agiront pour tous les autres.

- Pour retrouver la fiche récapitulative de RMarkdown : > Help/Cheat Sheets/ R Markdown Cheat Sheet

[1] On peut citer les formats `odt`, `rtf`, Markdown, etc.

Figure 1: Illustration du langage R par DALL-E

Comment ça marche ?

- **Fusion de R et Markdown :** RMarkdown combine le langage de programmation R avec la syntaxe de formatage de texte Markdown. Cela permet d'écrire du texte formaté (comme en HTML ou LaTeX) tout en exécutant du code R.
- **Documents Dynamiques :** Les documents RMarkdown sont dynamiques, ce qui signifie que les résultats du code R (comme les graphiques ou les tableaux) sont intégrés directement dans le document.
- **Flexibilité de Format :** Les documents RMarkdown peuvent être convertis en plusieurs formats, y compris HTML, PDF, et Word.

1. **Écriture du Document :** Vous commencez par écrire un document en utilisant la syntaxe Markdown pour le texte et des blocs de code spéciaux (délimités par `{r} et`) pour le code R.
2. **Exécution et Conversion :** Lorsque le document est "tricoté" (le processus de conversion), le code R est exécuté et ses résultats sont incorporés dans le document final.
3. **Résultats Intégrés :** Les résultats, qu'il s'agisse de sorties textuelles, de tableaux, ou de graphiques, sont affichés à l'endroit exact où le bloc de code correspondant se trouve dans le texte.

Utilisations de RMarkdown

- **Rapports de Données :** Pour générer des rapports qui incluent à la fois l'analyse (code R) et les interprétations (texte).
- **Documents Académiques :** Pour des articles de recherche, des thèses, où les analyses et leurs descriptions vont de pair.
- **Présentations :** Peut être utilisé pour créer des diapositives de présentation intégrant des analyses en temps réel.
- **Notebooks :** Pour créer des notebooks interactifs, similaires à Jupyter Notebooks, mais dans R.

Avantages

- **Reproductibilité :** Les documents RMarkdown facilitent la reproductibilité des analyses, un élément clé de la science des données.
- **Efficacité :** Ils permettent une mise à jour automatique des résultats et des graphiques en cas de modification des données ou du code.
- **Polyvalence :** Adapté pour une large gamme d'usages, de l'éducation à la recherche, en passant par le reporting d'entreprise.

En résumé, RMarkdown est un outil précieux pour ceux qui travaillent dans le domaine de la science des données, de la recherche, de l'enseignement, ou toute autre discipline où l'intégration de l'analyse de données et du rapport est essentielle.

Pour générer un document à partir d'un fichier RMarkdown une fois que votre analyse est terminée, vous utiliserez principalement la fonction "Knit" dans RStudio. Voici les étapes à suivre et les packages nécessaires :

Étapes pour Générer le Document

1. **Ouvrez votre fichier RMarkdown dans RStudio.**
2. **Cliquez sur le bouton "Knit" dans la barre d'outils.** Ce bouton se trouve généralement en haut du script RMarkdown. Il est représenté par une petite icône avec une pelote de laine et une aiguille.
3. **Choisissez le format de sortie.** RMarkdown supporte plusieurs formats de sortie tels que HTML, PDF, ou Word. Le format de sortie peut être spécifié dans l'en-tête YAML de votre document.

4. **Nommez et enregistrez votre fichier.** RStudio générera le document dans le format choisi et vous demandera où l'enregistrer.

Packages Nécessaires

Pour utiliser RMarkdown et générer des documents, vous aurez besoin de certains packages. Les plus importants sont :

1. **rmarkdown :** Le package principal pour travailler avec RMarkdown.

```
#install.packages("rmarkdown")
```

2. **knitr :** Utilisé pour "tricoter" le document, c'est-à-dire pour exécuter le code R et le combiner avec le texte.

```
#install.packages("knitr")
```

3. **ggplot2 (optionnel) :** Si votre analyse implique des visualisations de données.

```
#install.packages("ggplot2")
```

4. **dplyr (optionnel) :** Pour la manipulation de données.

```
#install.packages("dplyr")
```

Pour le Format PDF

Si vous souhaitez générer des documents au format PDF, vous aurez besoin d'une installation LaTeX. Une option facile est d'installer TinyTeX, une distribution LaTeX légère :

- Installer TinyTeX :

```
#install.packages("tinytex")
#tinytex::install_tinytex()
```

Après l'Installation

Après avoir installé les packages nécessaires, vous pouvez simplement ouvrir votre fichier RMarkdown dans RStudio et cliquer sur "Knit" pour générer le document. Assurez-vous que toutes les dépendances de votre analyse (comme les packages R spécifiques) sont également installées et disponibles.

Éléments d'un document R Markdown

En-tête (préambule)

La première partie du document est son *en-tête*. Il se situe en tout début de document, et est délimité par trois tirets (`---`) avant et après:

Cet en-tête contient les métadonnées du document, comme son titre, son auteur, sa date, plus tout un tas d'options possibles qui vont permettre de configurer ou personnaliser l'ensemble du document et son rendu. Ici, par exemple, la ligne `output: html_document` indique que le document généré doit être au format HTML.

Texte du document

Titres

- `#` pour un titre de niveau 1 (le plus grand)
- `##` pour un titre de niveau 2
- `###` pour un titre de niveau 3, et ainsi de suite jusqu'à `######` pour un titre de niveau 6.

Emphase

- `*texte*` ou `_texte_` pour de l'italique
- `**texte**` ou `__texte__` pour du gras
- `***texte***` ou `___texte___` pour gras et italique
- `~~texte~~` pour barrer le texte

Listes

- Utilisez `*`, `+`, ou `-` pour les listes non ordonnées
- Utilisez des nombres suivis d'un point pour les listes ordonnées

Liens

- `[Texte du lien](URL)` pour un lien standard
- `[Texte du lien](URL "Titre facultatif")` pour un lien avec un titre

Images

- `![Texte alternatif](URL de l'image)` pour insérer une image

Citations

- `>` pour une citation

Code

- Utilisez des guillemets inversés `` ` `` autour d'un petit segment de code
- Utilisez trois guillemets inversés ```` ``` ```` pour un bloc de code

Tableaux

- Utilisez `|` pour séparer les colonnes et `-` pour les en-têtes de colonnes

Lignes horizontales

- Utilisez trois ou plus de `-`, `*`, ou `_` pour créer une ligne horizontale

Sauts de ligne

- Utilisez deux espaces ou plus à la fin d'une ligne pour un saut de ligne

Texte préformaté

- Utilisez ` (trois espaces) devant et derrière le texte pour préserver la mise en forme exacte

Cette syntaxe de base permet de créer des documents clairs et structurés en Markdown, que ce soit pour des documents RMarkdown ou autres projets utilisant Markdown.

> Pour retrouver les commandes usuelles RMarkdown : **lien**

Exemple d'analyse

L'ensemble de données `diamonds` disponible dans le package `ggplot2` en R est un excellent choix pour effectuer une analyse exploratoire.

Cet ensemble de données contient les prix et les attributs de près de 54 000 diamants. Voici un exemple d'analyse que vous pourriez réaliser avec ces données :

Préparation de l'Environnement

Comme pour tout projet vous devez d'abord commencer par chargez vos libraries et vos données.

```
# Chargement des packages nécessaires
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
# Chargement des données
data("diamonds")
```

Aperçu des Données

```r
# Affichage des premières lignes
head(diamonds)
```

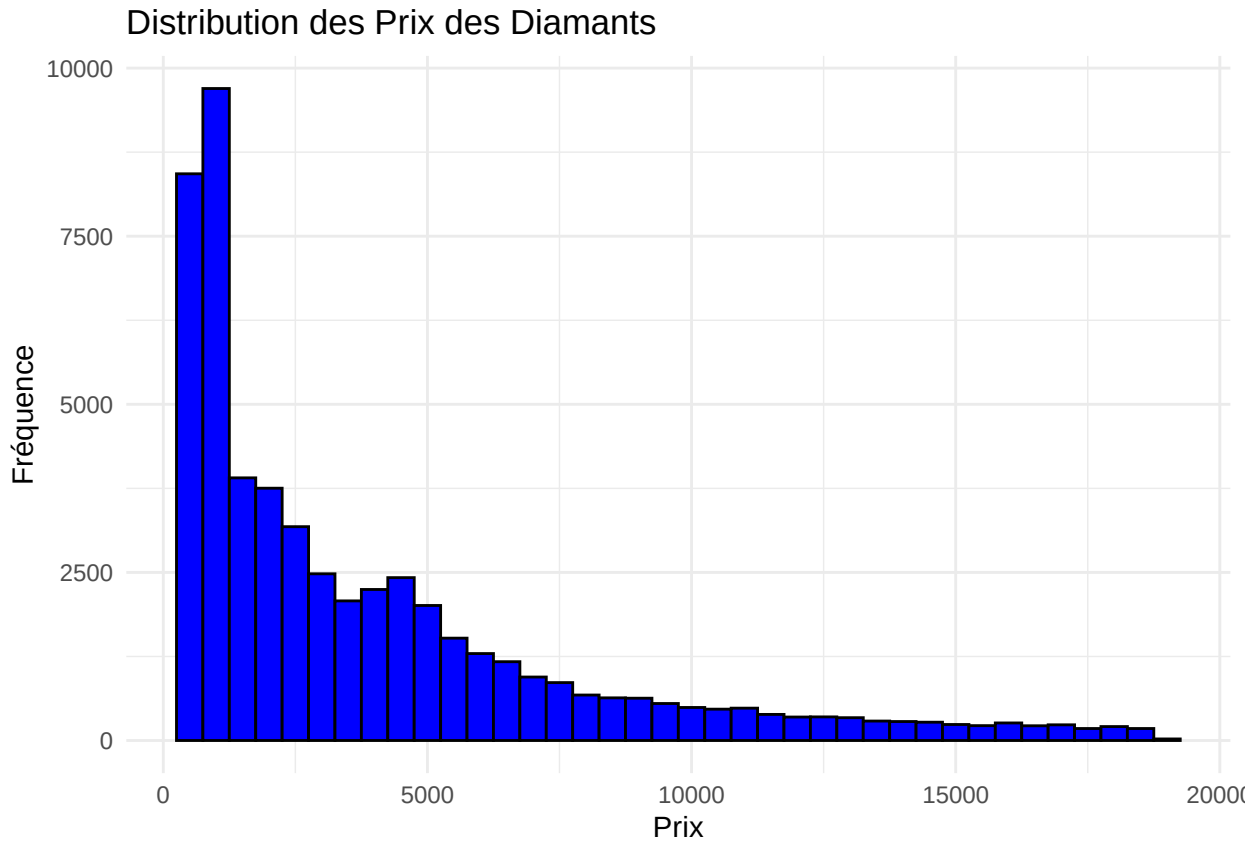
```
## # A tibble: 6 x 10
##   carat cut       color clarity depth table price     x     y     z
##   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1  0.23 Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
## 2  0.21 Premium   E     SI1      59.8    61   326  3.89  3.84  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
```

```r
# Résumé des données
summary(diamonds)
```

```
##      carat               cut        color        clarity          depth
##  Min.   :0.2000   Fair      : 1610   D: 6775   SI1    :13065   Min.   :43.00
##  1st Qu.:0.4000   Good      : 4906   E: 9797   VS2    :12258   1st Qu.:61.00
##  Median :0.7000   Very Good:12082   F: 9542   SI2    : 9194   Median :61.80
##  Mean   :0.7979   Premium   :13791   G:11292   VS1    : 8171   Mean   :61.75
##  3rd Qu.:1.0400   Ideal     :21551   H: 8304   VVS2   : 5066   3rd Qu.:62.50
##  Max.   :5.0100                     I: 5422   VVS1   : 3655   Max.   :79.00
##                                     J: 2808   (Other): 2531
##      table           price             x                y
##  Min.   :43.00   Min.   :  326   Min.   : 0.000   Min.   : 0.000
##  1st Qu.:56.00   1st Qu.:  950   1st Qu.: 4.710   1st Qu.: 4.720
##  Median :57.00   Median : 2401   Median : 5.700   Median : 5.710
##  Mean   :57.46   Mean   : 3933   Mean   : 5.731   Mean   : 5.735
##  3rd Qu.:59.00   3rd Qu.: 5324   3rd Qu.: 6.540   3rd Qu.: 6.540
##  Max.   :95.00   Max.   :18823   Max.   :10.740   Max.   :58.900
##
##        z
##  Min.   : 0.000
##  1st Qu.: 2.910
##  Median : 3.530
##  Mean   : 3.539
##  3rd Qu.: 4.040
##  Max.   :31.800
##
```

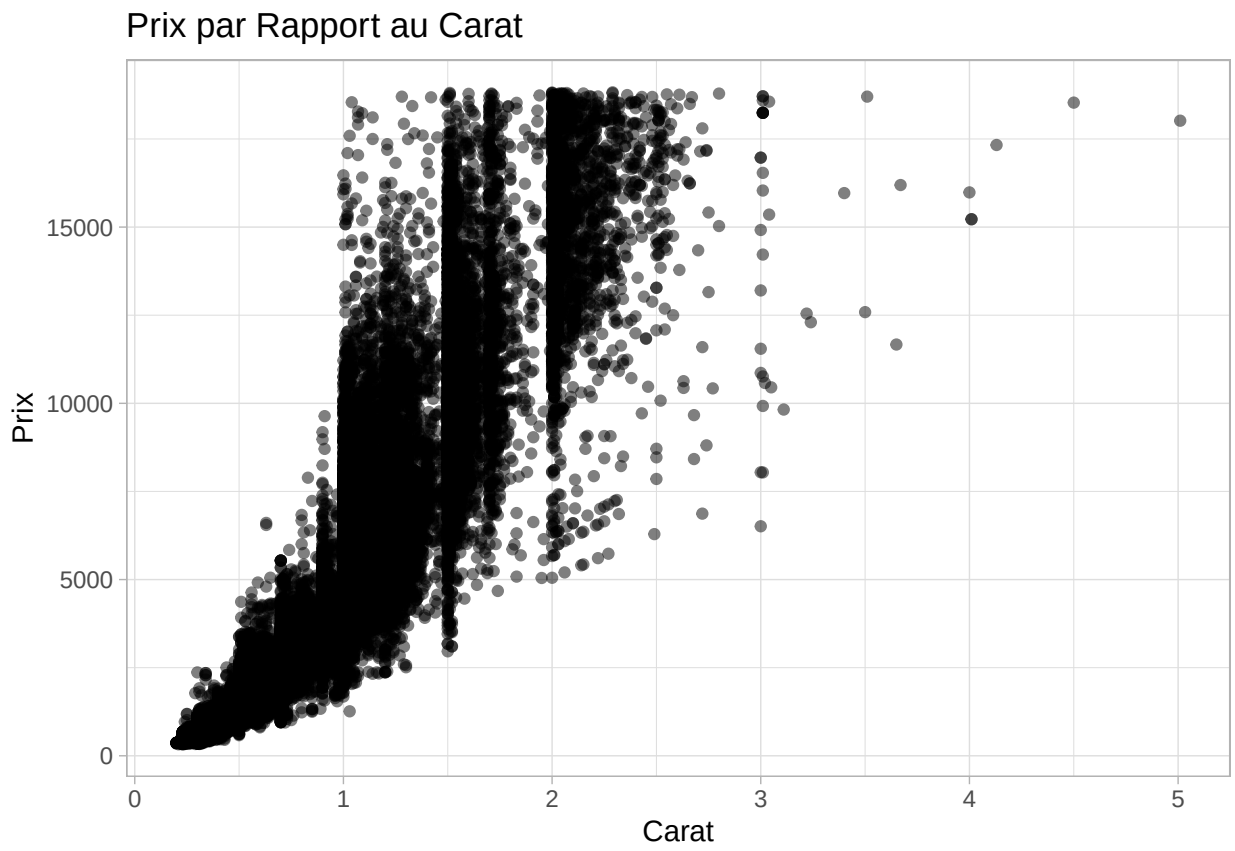
Analyse de la Distribution des Prix

```r
# Histogramme de la distribution des prix
ggplot(diamonds, aes(x = price)) +
  geom_histogram(binwidth = 500, fill = "blue", color = "black") +
  theme_minimal() +
  labs(title = "Distribution des Prix des Diamants",
       x = "Prix", y = "Fréquence")
```



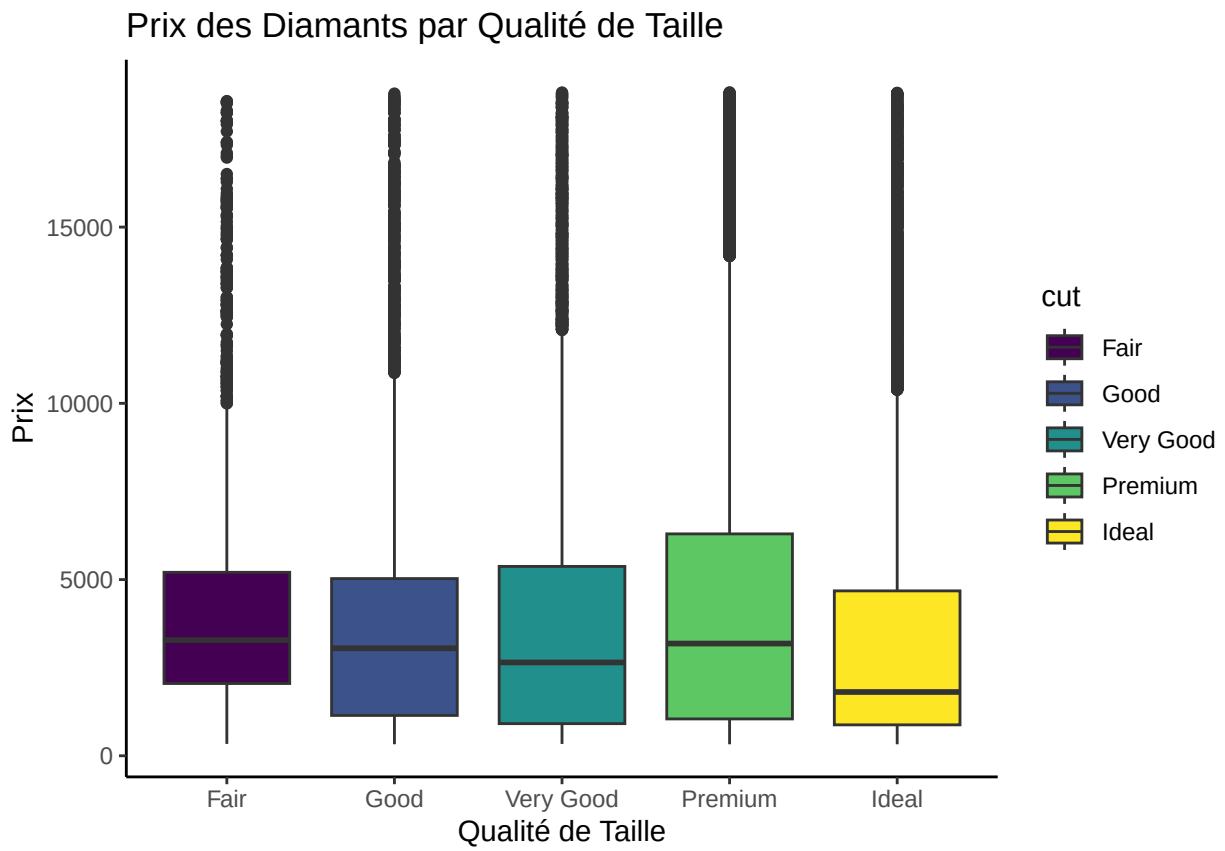
Relation entre le Prix et la Qualité du Carat

```r
# Graphique de dispersion montrant la relation entre le prix et le carat
ggplot(diamonds, aes(x = carat, y = price)) +
  geom_point(alpha = 0.5) +
  theme_light() +
  labs(title = "Prix par Rapport au Carat",
       x = "Carat", y = "Prix")
```



Analyse par Catégorie de Taille

```
# Boxplot du prix par catégorie de taille
ggplot(diamonds, aes(x = cut, y = price, fill = cut)) +
  geom_boxplot() +
  theme_classic() +
  labs(title = "Prix des Diamants par Qualité de Taille",
       x = "Qualité de Taille", y = "Prix")
```



Corrélation entre les Caractéristiques

```
# Utilisation de cor() pour calculer les corrélations
cor(diamonds[,c("price", "carat", "depth", "table")])
```

```
##             price      carat       depth      table
## price   1.0000000 0.92159130 -0.01064740  0.1271339
## carat   0.9215913 1.00000000  0.02822431  0.1816175
## depth  -0.0106474 0.02822431  1.00000000 -0.2957785
## table   0.1271339 0.18161755 -0.29577852  1.0000000
```

Conclusion de l'Analyse

- Ces graphiques et analyses fournissent un aperçu de la distribution des prix des diamants, ainsi que de la façon dont les attributs comme le carat et la qualité de la taille influencent ces prix.
- La corrélation peut révéler des relations intéressantes entre les différentes variables numériques de l'ensemble de données.

Nom et Options

Les options d'un bloc de code sont à placer à l'intérieur des accolades `{r}`.

Nom

La première possibilité est de donner un *nom* au bloc. Celui-ci est indiqué directement après le `r`:

```
{r nom_du_bloc}
```

Il n'est pas obligatoire de nommer un bloc, mais cela peut être utile en cas d'erreur à la compilation, pour identifier le bloc ayant causé le problème. Attention, on ne peut pas avoir deux blocs avec le même nom.

Options

En plus d'un nom, on peut passer à un bloc une série d'options sous la forme `option = valeur`. Voici un exemple de bloc avec un nom et des options:

Voici une liste de quelques unes des options disponibles :

Option	Valeurs	Description
echo	TRUE/FALSE	Afficher ou non le code R dans le document
eval	TRUE/FALSE	Exécuter ou non le code R à la compilation
include	TRUE/FALSE	Inclure ou non le code R et ses résultats dans le document
results	"hide"/"asis"/"markup"/"hold"	Type de résultats renvoyés par le bloc de code
warning	TRUE/FALSE	Afficher ou non les avertissements générés par le bloc
message	TRUE/FALSE	Afficher ou non les messages générés par le bloc

Il existe de nombreuses autres options décrites notamment dans guide de référence R Markdown (PDF en anglais).

Rendu des tableaux

Tableaux croisés kable()

Par défaut, les tableaux issus de la fonction `table` sont affichés comme ils apparaissent dans la console de R, en texte brut :

```
library(questionr)
data(hdv2003)
tab <- lprop(table(hdv2003$qualif, hdv2003$sexe))
tab
```

```
##
##                           Homme Femme Total
##   Ouvrier specialise       47.3  52.7 100.0
##   Ouvrier qualifie         78.4  21.6 100.0
##   Technicien               76.7  23.3 100.0
##   Profession intermediaire 55.0  45.0 100.0
##   Cadre                    55.8  44.2 100.0
##   Employe                  16.2  83.8 100.0
##   Autre                    36.2  63.8 100.0
##   Ensemble                 44.8  55.2 100.0
```

On peut améliorer leur présentation en utilisant la fonction `kable` de l'extension `knitr`. Celle-ci fournit un formatage adapté en fonction du format de sortie. On aura donc des tableaux "propres" que ce soit en HTML, PDF ou aux formats traitements de texte :

```
library(knitr)
kable(tab)
```

	Homme	Femme	Total
Ouvrier specialise	47.29064	52.70936	100
Ouvrier qualifie	78.42466	21.57534	100
Technicien	76.74419	23.25581	100
Profession intermediaire	55.00000	45.00000	100
Cadre	55.76923	44.23077	100
Employe	16.16162	83.83838	100
Autre	36.20690	63.79310	100
Ensemble	44.82759	55.17241	100

Différents arguments permettent de modifier la sortie de `kable`. `digits`, par exemple, permet de spécifier le nombre de chiffres significatifs à afficher dans les colonnes de nombres:

```
kable(tab, digits = 1)
```

	Homme	Femme	Total
Ouvrier specialise	47.3	52.7	100
Ouvrier qualifie	78.4	21.6	100
Technicien	76.7	23.3	100
Profession intermediaire	55.0	45.0	100
Cadre	55.8	44.2	100
Employe	16.2	83.8	100
Autre	36.2	63.8	100
Ensemble	44.8	55.2	100

Tableaux croisés tbl_summary()

Le `tbl_summary()` est une fonction de la bibliothèque `gtsummary` en R, qui est utilisée pour créer des résumés statistiques de données. Voici une explication brève de son utilisation :

1. **Installation de la bibliothèque**: Avant de pouvoir utiliser `tbl_summary()`, vous devez installer et charger la bibliothèque `gtsummary` dans votre environnement R. Cela se fait généralement avec la commande `install.packages("gtsummary")`, suivie de `library(gtsummary)`.
2. **Préparation des données**: Assurez-vous que vos données sont dans un dataframe R. `tbl_summary()` fonctionne avec des dataframes.
3. **Utilisation de la fonction**:
   - **Syntaxe de base**: La fonction s'utilise en passant votre dataframe comme premier argument. Par exemple, `tbl_summary(your_data)`.
   - **Spécification des statistiques**: Vous pouvez spécifier les statistiques à calculer pour chaque variable (moyenne, médiane, etc.) en utilisant l'argument `statistic`.

- **Groupement des données**: Si vous voulez comparer des groupes dans vos données, utilisez l'argument `by`. Par exemple, `tbl_summary(your_data, by = "group_variable")` pour résumer les données par groupe.

4. **Personnalisation**: `tbl_summary()` offre diverses options de personnalisation, comme le changement des labels des variables, l'ajout de notes au bas du tableau, etc.

5. **Résultat**: Le résultat est un tableau formaté qui résume les statistiques clés de chaque variable dans le dataframe, avec des options pour la comparaison de groupes et la personnalisation de l'affichage.

C'est un outil très utile pour l'exploration de données et la communication des résultats statistiques de manière concise et claire.

Voici quelques exemples d'utilisation de la fonction `tbl_summary()` de la bibliothèque `gtsummary` en R, illustrant différentes manières de résumer et de présenter des données statistiques :

Exemple 1: Résumé Basique

```r
# Chargement de la bibliothèque
library(gtsummary)

# Exemple de données
data(mtcars)

# Création d'un résumé basique
table1 <- tbl_summary(mtcars)
table1
```

Characteristic	N = 32
mpg	19.2 (15.4, 22.8)
cyl	
4	11 (34%)
6	7 (22%)
8	14 (44%)
disp	196 (121, 326)
hp	123 (97, 180)
drat	3.70 (3.08, 3.92)
wt	3.33 (2.58, 3.61)
qsec	17.71 (16.89, 18.90)
vs	14 (44%)
am	13 (41%)
gear	
3	15 (47%)
4	12 (38%)
5	5 (16%)
carb	
1	7 (22%)
2	10 (31%)
3	3 (9.4%)
4	10 (31%)
6	1 (3.1%)
8	1 (3.1%)

Characteristic	N = 32

Dans cet exemple, `tbl_summary()` crée un tableau résumant chaque colonne du dataframe `mtcars` (comme la moyenne pour les variables continues, et les fréquences pour les variables catégorielles).

Exemple 2: Résumé avec Statistiques Spécifiques

```
table2 <- tbl_summary(
  mtcars,
  statistic = list(all_continuous() ~ "{mean} ({sd})", # Pour les variables continues
                   all_categorical() ~ "{n} / {N} ({p}%)") # Pour les variables catégorielles
)
table2
```

Characteristic	N = 32
mpg	20.1 (6.0)
cyl	
4	11 / 32 (34%)
6	7 / 32 (22%)
8	14 / 32 (44%)
disp	231 (124)
hp	147 (69)
drat	3.60 (0.53)
wt	3.22 (0.98)
qsec	17.85 (1.79)
vs	14 / 32 (44%)
am	13 / 32 (41%)
gear	
3	15 / 32 (47%)
4	12 / 32 (38%)
5	5 / 32 (16%)
carb	
1	7 / 32 (22%)
2	10 / 32 (31%)
3	3 / 32 (9.4%)
4	10 / 32 (31%)
6	1 / 32 (3.1%)
8	1 / 32 (3.1%)

Ici, le tableau inclut la moyenne et l'écart-type pour les variables continues, et le compte, la taille totale de l'échantillon et le pourcentage pour les variables catégorielles.

Exemple 3: Résumé par Groupes

```
table3 <- tbl_summary(
  mtcars,
  by = "cyl", # Groupe selon la variable 'cyl'
```

```
  statistic = all_continuous() ~ "{mean} ({sd})"
)
table3
```

Characteristic	**4**, N = 11	**6**, N = 7	**8**, N = 14
mpg	26.7 (4.5)	19.7 (1.5)	15.1 (2.6)
disp	105 (27)	183 (42)	353 (68)
hp	83 (21)	122 (24)	209 (51)
drat	4.07 (0.37)	3.59 (0.48)	3.23 (0.37)
wt	2.29 (0.57)	3.12 (0.36)	4.00 (0.76)
qsec	19.14 (1.68)	17.98 (1.71)	16.77 (1.20)
vs	10 (91%)	4 (57%)	0 (0%)
am	8 (73%)	3 (43%)	2 (14%)
gear			
3	1 (9.1%)	2 (29%)	12 (86%)
4	8 (73%)	4 (57%)	0 (0%)
5	2 (18%)	1 (14%)	2 (14%)
carb			
1	5 (45%)	2 (29%)	0 (0%)
2	6 (55%)	0 (0%)	4 (29%)
3	0 (0%)	0 (0%)	3 (21%)
4	0 (0%)	4 (57%)	6 (43%)
6	0 (0%)	1 (14%)	0 (0%)
8	0 (0%)	0 (0%)	1 (7.1%)

Ce tableau résume les données en affichant des statistiques pour les variables continues, séparées par les groupes de la variable 'cyl' (nombre de cylindres).

Exemple 4: Ajout de Labels et Modification de l'Apparence

```
table4 <- tbl_summary(
  mtcars,
  by = "am", # Groupe selon la transmission (automatique ou manuelle)
  label = list(mpg = "Miles per Gallon",
               hp = "Horsepower"),
  statistic = all_continuous() ~ "{mean} ({sd})",
  missing = "no" # Exclure les valeurs manquantes
) %>%
  modify_header(label ~ "**Variable**")
table4
```

Variable	**0**, N = 19	**1**, N = 13
Miles per Gallon	17.1 (3.8)	24.4 (6.2)
cyl		
4	3 (16%)	8 (62%)
6	4 (21%)	3 (23%)
8	12 (63%)	2 (15%)
disp	290 (110)	144 (87)

Variable	**0**, N = 19	**1**, N = 13
Horsepower	160 (54)	127 (84)
drat	3.29 (0.39)	4.05 (0.36)
wt	3.77 (0.78)	2.41 (0.62)
qsec	18.18 (1.75)	17.36 (1.79)
vs	7 (37%)	7 (54%)
gear		
3	15 (79%)	0 (0%)
4	4 (21%)	8 (62%)
5	0 (0%)	5 (38%)
carb		
1	3 (16%)	4 (31%)
2	6 (32%)	4 (31%)
3	3 (16%)	0 (0%)
4	7 (37%)	3 (23%)
6	0 (0%)	1 (7.7%)
8	0 (0%)	1 (7.7%)

Dans cet exemple, des labels personnalisés sont ajoutés pour certaines variables, et l'en-tête de la colonne 'label' est modifiée.

Pour en savoir plus sur le représentation des tableaux :

ThinkR

danieldsjoberg

larmarange.github.io

Ecrire du langage R en ligne

R Markdown permet d'intégrer du code R dans un document texte.

Voici une explication :

Lorsque que vous déclararez une variable dans un chunk vous pouvez ensuite directement afficher le contenu de la variable en utilisant la syntaxe suivante :

```r
# Calcul de la moyenne de l'age des femmes
moyenne_femme_age <-  hdv2003 %>%
  filter(sexe == "Femme") %>%
  select(age) %>%
  summarise(moyenne_age = mean(age)) %>%
  pull()

# Calcul du revenu moyen par catégorie professionnelle
moyenne_femme_heures_tv <- hdv2003 %>%
  filter(sexe == "Femme") %>%
  select(heures.tv) %>%
  summarise(moyenne_tv = mean( heures.tv)) %>%
  pull()
```

Nous observons que l'age moyen dans les données hdv20003 est de 48.1534968.

Exercices

Pour commencer à pratiquer, suivez ces étapes :

1. **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.
2. **Trouvez le Fichier des Exercices :** Dans le dépôt, localisez le fichier nommé **exercices6.txt**. Ce fichier contient les premiers exercices que vous devez pratiquer.
3. **Lisez et Essayez de Résoudre les Exercices :** Ouvrez le fichier **exercices6.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_exercices6.R** pour voir les solutions. Analysez les solutions pour comprendre les méthodes et logiques utilisées.