

Langage R pour la statistique

Chapitre 1 - Présentation du langage R et de RStudio

M. Bessec, S. Dufour & M. Menéndez

Septembre 2024

- Manuel en ligne : Rcompanion : <https://rcompanion.org>
- Decourt O. (2018). Le langage R au quotidien : Traitement et analyse de données volumineuses. Dunod.
- Maumy-Bertrand M. et Bertrand F. (2018). Initiation à la statistique avec R. Dunod.

Plan du chapitre

- 1 Installation de R et RStudio
- 2 Les fenêtres de l'interface RStudio
- 3 La structure d'un programme ou script R
- 4 Les différents objets dans R

Plan du chapitre

- 1 Installation de R et RStudio
- 2 Les fenêtres de l'interface RStudio
- 3 La structure d'un programme ou script R
- 4 Les différents objets dans R

- R a été créé en 1993 par Ross Ihaka et Robert Gentleman à l'Université d'Auckland (NZ).
- R est un langage de programmation et un environnement mathématique utilisé pour le traitement de données.
- R est une version libre et gratuite du langage S-Plus.
- R est écrit en C, C++, Fortran et Java. Il est orienté programmation objet.

Le langage R (suite)

- Le langage R est en accès libre sur l'internet sur le site <https://www.r-project.org/>.
- Un groupe de développeurs (la *R core development Team*) en assure la maintenance et l'évolution des fonctionnalités basiques.
- Le langage évolue en permanence. Une nouvelle version est disponible à peu près tous les 6 mois.
- Le logiciel et ses extensions sont diffusées via un réseau de serveurs nommé **CRAN** (*Comprehensive R Archive Network*); ainsi, ils sont disponibles simultanément à plusieurs endroits du monde pour faire face aux nombreux téléchargements.

Installation de R

Le logiciel se télécharge gratuitement depuis l'internet sur le site <https://www.r-project.org/> :

- ❶ Cliquer dans le menu sur CRAN
- ❷ Sélectionner Download R for Windows puis Download R 4.4.1 for Windows ou Download R for (Mac) OS X sur un Mac (versions disponibles en juillet 2024).
- ❸ Télécharger le fichier sélectionné et exécuter le fichier en choisissant une installation par défaut

Sous Windows, l'interface par défaut par laquelle on accèdera à R est l'interface graphique RGUI.

Installation de RStudio

Dans ce cours, on utilisera l'interface **RStudio** créée pour travailler avec R et qui offre un environnement convivial pour accéder aux fichiers script, la console R, les rubriques d'aide, les graphiques, etc.

On installera la version libre (Open Source) de RStudio depuis le site <https://www.rstudio.com>.

- Cliquer dans le menu sur Download RStudio.
- Sélectionner RStudio Desktop FREE.
- Télécharger RStudio-2024.04.2-764.exe pour une installation sous Windows et RStudio-2024.04.2-764.dmg pour une installation sur un Mac et exécuter le fichier.

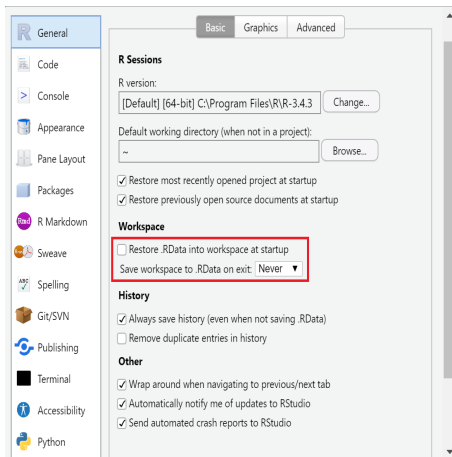
Lors de la première utilisation de RStudio

- Lors de la première utilisation de RStudio, pour sauvegarder dans nos scripts les commandes avec tous les caractères utilisés en français (les accents notamment), il faudra aller dans le menu principal Tools/Global options/Code/Saving et dans default text encoding, choisir **UTF-8**.
- Lors de la première utilisation de RStudio, il est également fortement recommandé de désactiver la sauvegarde automatique de l'espace de travail, i.e. la sauvegarde de l'ensemble des objets existant dans l'environnement lors de la séance de travail. Il s'agit d'éviter que la prochaine fois que R est lancé dans le même dossier, RStudio restaure l'ensemble des objets dans l'état où ils étaient.

Gestion de la mémoire (suite)

Pour cela, aller dans le menu **Tools**, puis **Global Options**, et s'assurer que :

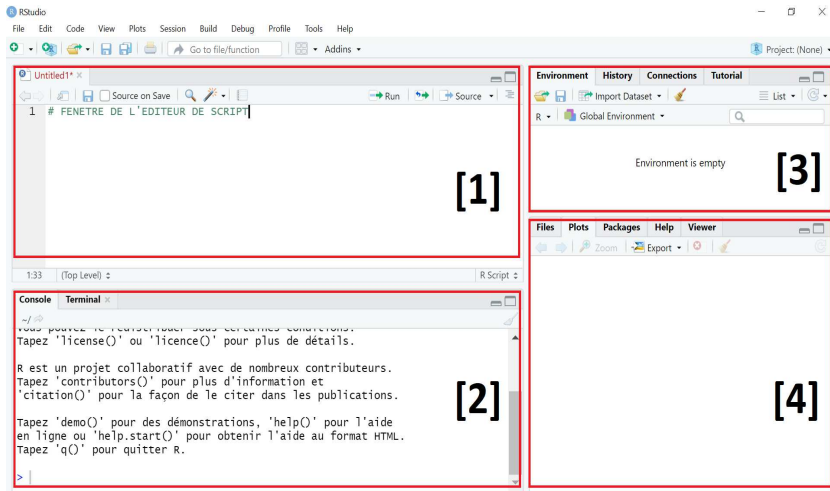
- 1) la case “Restore .RData into workspace at startup” est décochée.
- 2) le champ “Save workspace to .RData on exit” vaut Never :



Plan du chapitre

- 1 Installation de R et RStudio
- 2 Les fenêtres de l'interface RStudio**
- 3 La structure d'un programme ou script R
- 4 Les différents objets dans R

Les fenêtres de RStudio




Les fenêtres de RStudio (suite)

- ❶ (en haut à gauche) l'[éditeur de script](#) (voir les transparents suivants)
- ❷ (en bas à gauche) la [console](#) (voir les transparents suivants)
- ❸ (en haut à droite) [Environment/History/Connections](#). On trouvera dans ces fenêtres, les objets, les fonctions créés, le nom et les caractéristiques des fichiers/répertoires ouverts et les packages installés (Environment), le code de toutes les commandes exécutées durant la session via le script ou l'utilisation des menus (History).
- ❹ (en bas à droite) [Files/Plots/Packages/Help/Viewer](#). On retrouve l'explorateur (Files), les graphiques résultant des commandes (Plots); les packages déjà installés et une plateforme d'installation d'autres packages (Packages), l'onglet Help (manuels, aide en ligne, description des commandes).

Les fenêtres de RStudio : la console

- Apparaîtront dans cette fenêtre les commandes exécutées (**en bleu**), les résultats des commandes (**en noir**) et les messages R (d'erreur par exemple, **en rouge**).
- La console contient le signe `>` après lequel on peut saisir de façon interactive les commandes qui seront exécutées une à une en appuyant sur Entrée (mais elles seront perdues une fois la session fermée).
- Dans la console, on pourra utiliser les flèches vers le haut et vers le bas pour naviguer dans l'historique des commandes que l'on a exécutées précédemment. Le raccourci CTRL+L permet d'effacer la console.


Les fenêtres de RStudio : l'éditeur de script

- Plutôt que de saisir et d'exécuter une à une nos commandes dans la console, on peut les écrire et les regrouper dans un fichier ou programme appelé **Script**.
- Un Script R est un fichier texte brut (ouvrable avec le Bloc-notes ou un éditeur de texte), contenant des commandes R à faire exécuter et nous permettant de garder une trace de nos traitements.
- Pour ouvrir l'éditeur de script, il faut aller dans le menu *File/New File/R Script* (ou *File/Open file* s'il s'agit d'ouvrir un script qui existe déjà).
- Une fois le Script ouvert, on y écrit les commandes que l'on exécute en les sélectionnant et en cliquant sur  **Run** ou en tapant CTRL+Entrée.

Plan du chapitre

- 1 Installation de R et RStudio
- 2 Les fenêtres de l'interface RStudio
- 3 La structure d'un programme ou script R
- 4 Les différents objets dans R

Création d'un script

- Pour ouvrir l'éditeur de script, il faut aller dans le menu *File/New File/R Script* (ou *File/Open file* pour ouvrir un script déjà créé); on pourra aussi utiliser l'icône à gauche dans la barre d'outils 
- On utilisera le menu *File/Save as* pour enregistrer le script. Les Scripts sauvegardés depuis RStudio auront pour extension automatique ".R" (fichiers R).
- Pour sauvegarder dans le script les commandes avec tous les caractères utilisés en français (les accents notamment), il faudra aller lors de la première utilisation de RStudio dans le menu principal *Tools/Global options/Code/Saving* et dans *default text encoding*, choisir **UTF-8**.

Définition du répertoire de travail

- Il est pratique de créer un répertoire de travail et d'y enregistrer les scripts, les bases de données, etc.
- Cela nous permet d'éviter de spécifier les chemins de nos bases de données ou autres fonctions définies par l'utilisateur et utiles dans l'exécution du programme.
- Avant d'exécuter le script, on définira ce répertoire comme notre espace de travail en utilisant le menu *Session/Set working directory/Choose directory*.
- On utilisera de manière alternative la commande `setwd()`, par exemple `setwd("C:/TP_R")` pour travailler dans le répertoire TP_R (la commande `getwd()` permet de connaître le répertoire actif). Attention à ne pas utiliser \ mais / ou \\ lorsque l'on précise l'emplacement d'un répertoire.

- On peut mettre des éléments en commentaire en les précédant du symbole `#` (tout ce qui trouve derrière jusqu'à la fin de la ligne sera ignoré). Si l'on souhaite mettre plusieurs lignes du script en commentaire, on les entoure de guillemets "lignes non exécutées".

```
> note <- 11  # la note de statistique
```

- Lorsqu'un script est long, RStudio permet de créer des "sections" facilitant la navigation. Il suffit de faire suivre une ligne de commentaire par plusieurs tirets, quatre au minimum, soit par exemple :

```
> # Partie 1 du script ----
```

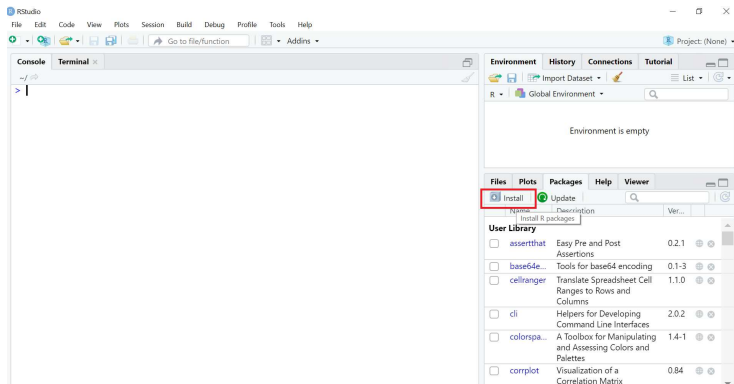
RStudio affiche alors dans la marge de gauche du script un petit triangle noir qui permet de replier ou déplier le contenu de la section en question.

Installation et chargement d'un package

- Les “packages” sont des bibliothèques de fonctions additionnelles créées par la communauté des utilisateurs de R, et diffusées via le réseau de serveurs CRAN.
- Pour télécharger ces extensions, on peut utiliser l'onglet **Packages** de RStudio (fenêtre en bas à droite) et installer l'extension désirée sur le disque dur.
- Il faut le faire une seule fois par package, mais une fois les extensions nécessaires installées, il faudra les “charger” avant de pouvoir les utiliser dans une session de travail. Ceci peut se faire au début du script avec la fonction **library()** et le nom du package concerné entre parenthèses. Ainsi, bien souvent, on regroupe en début de script toute une série d'appels avec **library()** pour charger tous les packages utilisés dans le script.

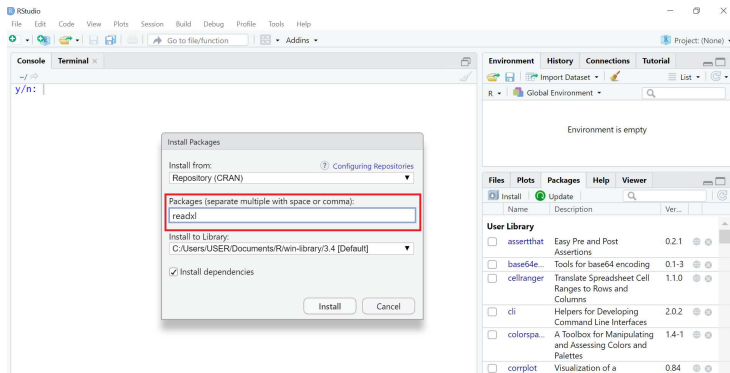
Exemple : chargement du package readxl

On clique sur **Install** dans l'onglet **Packages** dans la fenêtre en bas à droite de RStudio.



Exemple : chargement du package readxl (suite)

Dans la fenêtre **Install Packages** qui apparaît, rentrer le nom du ou des package(s) que l'on souhaite charger dans RStudio et cliquer sur **Install**.



Pour pouvoir ensuite utiliser ce package, on écrira `library(readxl)` au début du script.

- On utilisera `rm(list = ls())` pour vider tous les objets en mémoire (visible dans la partie Environnement ou en exécutant la commande `ls()`). Cette opération est recommandée en début du programme pour éviter d'utiliser par erreur des variables qui auraient été définies préalablement dans un autre programme.
- R distingue les minuscules et les majuscules.
- On pourra réunir plusieurs lignes de commande sur une même ligne en les séparant par un point virgule ;
- `options(digits=3)` pour limiter l'affichage à une précision de trois chiffres après la virgule (ici).

Exemple de début de script

```
> ### Exemple de Script d'introduction à R
>
> ## POUR COMMENCER AVEC UN ENVIRONNEMENT VIDE -----
> rm(list = ls())
> ## CHARGEMENT DES EXTENSIONS NECESSAIRES -----
> library(readxl)      # pour importer fichiers Excel
> library(ggplot2)     # pour faire des graphiques
```


Plan du chapitre

- 1 Installation de R et RStudio
- 2 Les fenêtres de l'interface RStudio
- 3 La structure d'un programme ou script R
- 4 Les différents objets dans R

La notion d'objet

- Le langage R est dit orienté **objet**, i.e. on y manipule des objets. Un objet est un espace de stockage d'éléments qui nous intéressent.
- Un objet a différentes caractéristiques :
 - ▷ un **nom** qui doit commencer par une lettre et ne peut pas comporter d'espace ou de caractères spéciaux autres que le point et l'espace souligné `_` ;
 - ▷ une **classe**, i.e. la nature de l'objet comme un vecteur, matrice, array etc (voir slide suivant) ;
 - ▷ un **mode**, i.e. la nature des éléments qu'il contient (*numeric* s'il contient des nombres, *character* s'il contient des chaînes de caractères, etc. voir slide suivant) ;
 - ▷ une **taille**, i.e. le nombre d'éléments qu'il contient.

Remarque : la classe d'un objet sera obtenue avec la commande `class()`, son mode avec la commande `mode()` et sa taille avec la commande `dim()` par exemple.

Les différentes classes d'objet

On sera amené à manipuler différentes classes d'objets :

- le **vecteur** : un vecteur d'éléments de même mode
- la **matrice** : tableau bidimensionnel ; par rapport au vecteur, peut contenir plusieurs colonnes
- l'**array** : tableau multidimensionnel (par rapport à la matrice, la dimension peut-être supérieure à 2)
- le **data.frame** : tableau bidimensionnel comme la matrice mais les colonnes peuvent être de mode différents ; utile en statistique quand on manipule des bases avec des individus en ligne et des variables de différents types en colonne
- la **list** : par rapport au data.frame, les colonnes peuvent avoir des tailles différentes
- la **ts** : une série temporelle (un vecteur numérique avec des dates)

Dans les 3 premiers, les éléments contenus sont nécessairement de même type ou mode (i.e. numérique, chaîne de caractères, etc)

Les modes des éléments d'un objet

Dans ce cours, on manipulera des éléments de différents types ou modes

- **numeric** (par exemple, 4.3, 100, -333)

```
> x <- 4.3
```

- **character** : (par exemple, "h", "Bonjour", "11.5")

```
> x <- "Hello"
```

- **logical** : (TRUE ou FALSE, ou plus simplement T ou F)

```
> x <- TRUE
```

On pourra utiliser la fonction `mode()` pour vérifier le type de la variable.

Opérations élémentaires sur un objet : affectation

- L'opérateur d'assignation `<-` permet d'affecter le résultat d'une fonction ou directement un élément à un objet. Cette flèche stocke ce qu'il y a à sa droite dans un objet dont le nom est indiqué à sa gauche.
- Par exemple, on stocke ici dans `x` le nombre 3 avec la commande suivante écrite dans un script ou la console

```
> x <- 3
```

- L'assignation peut généralement aussi se faire avec le signe `"="`, mais l'opérateur `"<-"` reste le plus répandu et sera privilégié dans ce cours.
- On pourra utiliser le raccourci clavier **ALT + -** pour insérer plus rapidement l'opérateur dans une ligne de commande.

Opérations élémentaires : affichage

Pour **afficher** un objet, on utilise la commande `print(nom_objet)` ou on tape directement le nom de l'objet dans la console (on pourra préciser le nombre de décimales avec l'argument `digits`).

```
> x <- 3  
> print(x) # ou print(x,digits=0)
```

```
[1] 3
```

Si l'on souhaite afficher plusieurs objets (par exemple `x` et `y`), on utilise l'opérateur de concaténation (voir aussi le chapitre 3) :

```
> x <- 3  
> y <- 5  
> print(c(x,y))
```

```
[1] 3 5
```

Opérations élémentaires : suppression

Pour **détruire** un objet, on utilise la commande `rm(nom_objet)` ; par exemple, pour détruire la variable `x` précédente :

```
> rm(x)
```

Si l'on souhaite détruire plusieurs objets, on les écrit séparés par des virgules, par exemple `rm(x,y)` pour détruire à la fois `x` et `y` (et `rm(list = ls())` pour vider l'environnement global).

Annexe : Synthèse des raccourcis clavier utiles

- **ALT + -** pour insérer l'opérateur d'affectation `<-` (moins du pavé numérique)
- **CTRL + entrée** dans l'éditeur de script pour exécuter un bloc de lignes de commande préalablement sélectionnées
- **CTRL + ALT + E** dans l'éditeur de script pour exécuter le code depuis la ligne active jusqu'à la fin du script
- **CTRL + SHIFT + C** dans l'éditeur de script pour mettre en commentaire (ou l'inverse) un bloc de lignes préalablement sélectionnées
- **F1** dans l'éditeur de script pour afficher de l'aide sur une fonction préalablement sélectionnée
- **Echap** pour interrompre l'exécution d'un script
- **CTRL + L** dans la console pour effacer la console

Langage R pour la statistique

Chapitre 2 - Traitements statistiques de base sur les données

M. Bessec, S. Dufour & M. Menéndez

Septembre 2024

- Manuel en ligne : Rcompanion : <https://rcompanion.org>
- Decourt O. (2018). Le langage R au quotidien : Traitement et analyse de données volumineuses. Dunod.
- Maumy-Bertrand M. et Bertrand F. (2018). Initiation à la statistique avec R. Dunod.

1 Importation de données

- Importation d'une base
- Visualisation de la base et de ses caractéristiques
- Création de variables/sous-blocs de la base

2 Statistique descriptive univariée

- Résumés numériques
- Tableaux de fréquences
- Graphiques

3 Statistique descriptive bivariable

- Résumés numériques
- Graphiques
- Tableaux de fréquences

Plan du chapitre

1 Importation de données

- Importation d'une base
- Visualisation de la base et de ses caractéristiques
- Création de variables/sous-blocs de la base

2 Statistique descriptive univariée

- Résumés numériques
- Tableaux de fréquences
- Graphiques

3 Statistique descriptive bivariable

- Résumés numériques
- Graphiques
- Tableaux de fréquences

Importation de données

Il existe de nombreux packages permettant l'importation de données depuis un grand nombre de formats de fichiers.

On utilisera par exemple :

- le package `readxl` : pour importer des données Excel
- le package `readr` : pour importer des données en format texte
- le package `haven` : pour des données en SAS, Stata, SPSS, etc

Il faut installer préalablement ces packages dans le disque dur de l'ordinateur depuis la fenêtre **Packages** en bas à droite ou avec la commande `install.packages(nom_package)`, puis les “charger” pour la séance de travail quand on en a besoin, en incluant la commande `library(nom_package)` au sein du script (voir chapitre 1).

On détaille dans la suite l'importation de données Excel.

Exemple de la base Excel BaseR

Les exemples qui suivent portent sur la base BaseR.xlsx (en ligne sur l'espace du cours).

	A	B	C	D	E
1	nom	sexe	argent_poch	diplome	fratrie
2	Dupuis	homme	45	L1	2
3	Dupont	homme	28	L2	0
4	Dupond	homme	100	L2	1
5	Meyer	femme	25	L1	1
6	Berthelot	homme	13	Master	1
7	Romagny	femme	28	L2	1
8	Lefèvre	femme	46	L2	0
9	Klap	femme	84	L1	2
10	Jamet	homme	75	L2	3
11	Terran	femme	62	L2	5
12	Bouabdal	femme	31	Master	2
13	Petit	femme	45	L2	2
14	Nguyen	homme	95	L3	2
15	Varga	homme	70	L3	1
16	Py	femme	52	L3	3
17	Ladam	homme	45	L2	2
18	Fontaine	femme	16	L3	4
19	Monier	homme	25	L1	1
20	Moniot	homme	30	L3	2
21	Vedie	homme	75	L2	3
22	Chapuis	homme	37	L1	2
23	Gerami	femme	32	L3	2
24	Gerard	femme	66	L1	1
25	Bourguignon	femme	75	L2	1
26	Bernabé	femme	39	L2	0
27	Dubois	homme	30	L1	1
28	Fleury	homme	22	Master	0
29	Gourdier	femme	12	Master	2
30	Bagard	femme	55	L2	4
31	Boucher	femme	38	L1	2
32					

data +

La base contient les caractéristiques de 30 étudiants à Dauphine.

Importation d'une base excel

On décrit ici comment importer la base Excel BaseR.xlsx et la stocker dans un `data.frame` (qui est appelé ici `Base`).

On utilise la fonction `read_excel` du package `readxl` :

```
> library(readxl) # Chargement de readxl  
> Base <- read_excel("BaseR.xlsx")  
> Base <- as.data.frame(Base)
```

L'objet `Base` qui est créé est un `data.frame` (voir chapitre 1). Il s'agit d'un tableau rectangulaire de données contenant en colonne des variables observées ici sur 30 individus (si on omet la dernière ligne, il s'agit d'un *tibble*, un objet peu différent d'un `data.frame` et ce qui suit reste valide).

Importation d'une base excel (suite)

Il est aussi possible de spécifier la feuille et la plage de cellules du fichier Excel que l'on souhaite importer avec les arguments `sheet` et `range` de la fonction `readxl` :

```
> library(readxl) # Chargement de readxl
> Base2 <- read_excel("BaseR.xlsx", sheet="data",
+                      range="A1:B16")
```

On copie ici dans le data.frame `Base2` le nom et le sexe des 15 premiers individus de la base.

Importation d'une base excel (suite)

Par défaut, les noms des variables sont les intitulés de chaque colonne. Dans le cas où la base excel ne comporte pas d'intitulés de colonnes, on utilisera l'argument `col_names = FALSE` de `read_excel`. Pour attribuer des labels aux colonnes et aux lignes, on utilisera les fonctions `colnames()` et `rownames()` sur le `data.frame`.

Par exemple, on utilise ici la première colonne de la base (les noms des individus) comme étiquettes de lignes et on supprime ensuite la colonne nom du `data.frame`.

```
> rownames(Base) <- Base[,1]
> Base <- Base[,-1]
```

On reviendra sur la syntaxe de la dernière ligne dans le chapitre 3.

Visualisation de la base et de ses caractéristiques

Pour afficher les données importées sous forme de feuille de calcul, on peut cliquer sur le nouvel objet qui apparait dans la fenêtre **Environnement** (en haut à droite) ou écrire la commande `View(Base)`.

	sexe	argent_poche	diplome	fratrie
Dupuis	homme	45	L1	2
Dupont	homme	28	L2	0
Dupond	homme	100	L2	1
Meyer	femme	25	L1	1
Berthelot	homme	13	Master	1
Romagny	femme	28	L2	1
Lefèvre	femme	46	L2	0
Klap	femme	84	L1	2
Jamet	homme	75	L2	3
Terran	femme	62	L2	5
Bouabdal	femme	31	Master	2
Petit	femme	45	L2	2
Nguyen	homme	95	L3	2
Varga	homme	70	L3	1
Py	femme	52	L3	3
Ladam	homme	45	L2	2

Attention à la majuscule de la commande View !

Visualisation de la base et de ses caractéristiques (suite)

Les fonctions `head()` et `tail()` affichent les premières et dernières lignes de la base.

```
> head(Base)
```

	sexe	argent_poche	diplome	fratrie
Dupuis	homme	45	L1	2
Dupont	homme	28	L2	0
Dupond	homme	100	L2	1
Meyer	femme	25	L1	1
Berthelot	homme	13	Master	1
Romagny	femme	28	L2	1

Visualisation de la base et de ses caractéristiques (suite)

La fonction `str()` renvoie un descriptif de la structure de la base de données (liste des variables, leur mode et les premières valeurs). Par exemple, `str(Base)` renvoie les éléments suivants dans la console :

```
'data.frame': 30 obs. of 4 variables:
 $ sexe      : chr  "homme" "homme" "homme" "femme" ...
 $ argent_poche: num  45 28 100 25 13 28 46 84 75 62 ...
 $ diplome   : chr  "L1" "L2" "L2" "L1" ...
 $ fratrie    : num  2 0 1 1 1 1 0 2 3 5 ...
```

Sous RStudio, on peut aussi afficher la structure d'un objet en cliquant sur le triangle sur fond bleu à gauche du nom de l'objet dans l'onglet **Environment** (fenêtre en haut à droite).

Visualisation de la base et de ses caractéristiques (suite)

La fonction `dim()` renvoie les dimensions de la base de données (nombre de lignes et de colonnes hors intitulés des variables).

```
> dim(Base)
```

```
[1] 30  4
```

La fonction `anyNA()` renvoie FALSE (TRUE) si la base ne comporte pas (comporte) des valeurs manquantes. La commande `sum(anyNA())` renverra le nombre de valeurs manquantes.

```
> anyNA(Base)
```

```
[1] FALSE
```

Création de variables

Pour faire appel à une variable de la base, `nom_base$nom_variable` avec `nom_variable` le nom de la variable du data.frame `nom_base`;

Par exemple, on crée ici une variable `x1` qui contient la variable `sexe` de la base :

```
> x1 <- Base$sexe
```

Pour alléger les notations, on pourra utiliser la commande `attach(nom_base)`. On n'a plus besoin de préciser dans quelle base chercher la variable et il suffira de donner simplement son nom (dans l'exemple qui suit, `sexe` au lieu de `Base$sexe`).

```
> attach(Base)
> x1 <- sexe
```

On annulera cette commande avec `detach(nom_base)`.

Extraction d'une partie de la base

La fonction `subset()` permet de sélectionner simplement des variables et des observations d'un fichier de données. Elle prend trois arguments principaux :

- le nom de l'objet de départ
- une condition sur les observations.
- éventuellement une condition sur les colonnes (`select`).

Dans l'exemple qui suit, on extrait et on stocke dans `BaseA` les observations de la base sur les étudiants en L1, puis dans `BaseB` les seules variables `sexe` et `fratrie` pour les étudiants de L1.

```
> BaseA<-subset(Base,diplome=="L1")  
> BaseB<-subset(Base,diplome=="L1",select=c(sexe,fratrie))
```

On pourra alternativement utiliser les opérations présentées dans la partie programmation (voir chapitre 3).

Plan du chapitre

1 Importation de données

- Importation d'une base
- Visualisation de la base et de ses caractéristiques
- Création de variables/sous-blocs de la base

2 Statistique descriptive univariée

- Résumés numériques
- Tableaux de fréquences
- Graphiques

3 Statistique descriptive bivariable

- Résumés numériques
- Graphiques
- Tableaux de fréquences

Notre base Excel BaseR contient un tableau de données individuelles.

	A	B	C	D	E
1	nom	sexe	argent_poche	diplome	fratrie
2	Dupuis	homme	45	L1	2
3	Dupont	homme	28	L2	0
4	Dupond	homme	100	L2	1
5	Meyer	femme	25	L1	1
6	Berthelot	homme	13	Master	1
7	Romagny	femme	28	L2	1
8	Lefèvre	femme	46	L2	0
9	Klap	femme	84	L1	2
10	Jamet	homme	75	L2	3
11	Terran	femme	62	L2	5
12	Bouabdal	femme	31	Master	2
13	Petit	femme	45	L2	2
14	Nguyen	homme	95	L3	2
15	Varga	homme	70	L3	1
16	Py	femme	52	L3	3
17	Ladam	homme	45	L2	2
18	Fontaine	femme	16	L3	4
19	Monier	homme	25	L1	1
20	Moniot	homme	30	L3	2
21	Vedie	homme	75	L2	3
22	Chapuis	homme	37	L1	2
23	Gerami	femme	32	L3	2
24	Gerard	femme	66	L1	1
25	Bourguignon	femme	75	L2	1
26	Bernabé	femme	39	L2	0
27	Dubois	homme	30	L1	1
28	Fleury	homme	22	Master	0
29	Gourdier	femme	12	Master	2
30	Bagard	femme	55	L2	4
31	Boucher	femme	38	L1	2
32					

< > data (+)

Ce tableau donne les caractéristiques de 30 individus : leur sexe (variable *qualitative nominale*), leur niveau d'étude (variable *qualitative ordinale*), leur nombre de frères et soeurs (variable *quantitative discrète*) et le montant de leur argent de poche (variable *quantitative continue*).

Cette présentation a l'avantage d'être très détaillée (on dispose de l'information individu par individu) mais lorsque la population ou l'échantillon est de grande taille, elle devient peu lisible. On a alors recours à des mesures numériques et à des tableaux et graphiques pour synthétiser l'information.

Pour alléger les commandes dans les exemples qui suivent, on définit les 4 variables suivantes :

```
> sexe <- Base$sexe  
> diplome <- Base$diplome  
> fratrie <- Base$fratrie  
> argent_poche <- Base$argent_poche
```

On utilisera alternativement la commande `attach()`.

Plan du chapitre

1 Importation de données

- Importation d'une base
- Visualisation de la base et de ses caractéristiques
- Création de variables/sous-blocs de la base

2 Statistique descriptive univariée

- Résumés numériques
- Tableaux de fréquences
- Graphiques

3 Statistique descriptive bivariable

- Résumés numériques
- Graphiques
- Tableaux de fréquences

Caractéristiques de position

Les caractéristiques de position renseignent sur l'ordre de grandeur des variables, leur valeur centrale etc. Plusieurs fonctions R peuvent être utilisées avec comme argument le nom de la variable (dans les exemples qui suivent `x`) observée sur `n` individus.

- `mean(x)` : moyenne de la variable `x`.
- `median(x)` : médiane de la variable `x`.
- `quantile(x, probs=p)` : quantile d'ordre `p` de `x` (par exemple, `quantile(x, probs=0.25)` donne le premier quartile).
- `min(x)` renvoie le minimum de `x` et `max(x)` son maximum.
- `summary(x)` renvoie min, Q1, Q2, moy, Q3, max pour `x`.

NB : Pour le calcul de plusieurs quantiles à la fois, on utilise une séquence de probabilités. Par exemple, `quantile(x, probs=seq(0.1,0.9,0.1))` renverra les déciles de `x` (`seq(min,max,a)` génère une suite arithmétique de `min` à `max` par pas de `a`, voir chapitre 3).

Caractéristiques de position - exemple

Exemple pour la variable argent de poche de la base :

```
> mean(argent_poche) # moyenne
```

```
[1] 46.53333
```

```
> median(argent_poche) # médiane
```

```
[1] 42
```

```
> quantile(argent_poche, probs=c(0.25,0.5,0.75)) # quartiles
```

```
 25%  50%  75%  
28.5 42.0 65.0
```

```
> summary(argent_poche) # résumé
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
12.00	28.50	42.00	46.53	65.00	100.00

Caractéristiques de dispersion

Les caractéristiques de dispersion renseignent sur la dispersion ou l'étalement des valeurs d'une variable. Plusieurs fonction R peuvent être utilisées avec comme argument le nom de la variable (dans les exemples qui suivent `x`) observée sur n individus.

- `var(x)` renvoie la variance corrigée $s'^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$ avec \bar{x} la moyenne de `x`
- `sd(x)` renvoie l'écart-type corrigé s'
- `diff(range(x))` renvoie l'étendue de `x` (ou `max(x)-min(x)`)
- `IQR(x)` renvoie l'écart interquartile ($Q3-Q1$), i.e. la longueur de l'intervalle situé au centre de la série de valeurs de `x` et contenant 50% des observations.
- `sd(x)/mean(x)` renvoie le coefficient de variation `x`.

Plus ces indicateurs sont élevés, plus la dispersion de la série est forte.

Caractéristiques de dispersion - exemple

Exemple pour la variable argent de poche de la base :

```
> sd(argent_poche) # écart-type corrigé
```

```
[1] 24.2156
```

```
> diff(range(argent_poche)) # étendue
```

```
[1] 88
```

```
> IQR(argent_poche) # écart interquartile
```

```
[1] 36.5
```

Fonctions de synthèse

Certaines fonctions permettent d'obtenir une grande partie de ces indicateurs. C'est par exemple le cas de `stat.desc(x)` qui nécessite d'avoir installé et appelé le package `pastecs` :

```
> library(pastecs)
> options(digits=2) # limite l'affichage à 2 décimales
> stat.desc(argent_poche)
```

nbr.val	nbr.null	nbr.na	min
30.00	0.00	0.00	12.00
max	range	sum	median
100.00	88.00	1396.00	42.00
mean	SE.mean	CI.mean.0.95	var
46.53	4.42	9.04	586.40
std.dev	coef.var		
24.22	0.52		

Plan du chapitre

1 Importation de données

- Importation d'une base
- Visualisation de la base et de ses caractéristiques
- Création de variables/sous-blocs de la base

2 Statistique descriptive univariée

- Résumés numériques
- **Tableaux de fréquences**
- Graphiques

3 Statistique descriptive bivariable

- Résumés numériques
- Graphiques
- Tableaux de fréquences

Tableau de fréquences

Les tableaux de fréquences fournissent un autre résumé utile des valeurs prises par une variable. Un tableau de fréquences contient les différentes modalités d'une variable et leur fréquence d'apparition dans la population ou l'échantillon.

Tableau de fréquences

X	x_1	\dots	x_j	\dots	x_l	Total
effectif	n_1	\dots	n_j	\dots	n_l	n
fréquence	$f_1 = \frac{n_1}{n}$	\dots	$f_j = \frac{n_j}{n}$	\dots	$f_l = \frac{n_l}{n}$	1

La première ligne contient les différentes modalités x_j , $j = 1, \dots, l$ (éventuellement regroupées en classes) et les lignes suivantes les effectifs (ou fréquences absolues) n_j (nombre d'individus de la base présentant la modalité x_j) et les fréquences relatives f_j (proportion d'individus présentant la modalité x_j).

Tableau de fréquences sous R

- Les effectifs (ou fréquences absolues) des modalités n_j sont obtenus avec la commande `table()` de R appliquée à une variable.
- L'option `useNA="always"` permet de faire apparaître la modalité Valeur manquante.
- Pour ajouter l'effectif total n de l'échantillon ou de la population sur laquelle est observée la variable, on utilisera la fonction `addmargins()` appliquée au tableau des effectifs précédent.
- On obtiendra les fréquences relatives f_j avec la fonction `prop.table()` appliquée au tableau des effectifs précédent (ou de façon alternative en divisant les effectifs par le nombre d'individus de la base).

Exemple de tableau de fréquences

On construit ici le tableau des effectifs de la variable diplôme.

```
> table(diplome)
```

```
diplome
  L1    L2    L3 Master
   8    12    6      4
```

On ajoute une colonne contenant l'effectif total :

```
> addmargins(table(diplome))
```

```
diplome
  L1    L2    L3 Master    Sum
   8    12    6      4    30
```

Exemple de tableau de fréquences (suite)

On calcule les fréquences relatives des diplômes :

```
> prop.table(table(diplome))
```

```
diplome
      L1      L2      L3 Master
0.27    0.40    0.20    0.13
```

On ajoute une colonne total :

```
> addmargins(prop.table(table(diplome)))
```

```
diplome
      L1      L2      L3 Master      Sum
0.27    0.40    0.20    0.13    1.00
```

Mise en forme du tableau : changement d'étiquettes

L'étiquette de la dernière colonne créée par `addmargins` est le nom de la fonction utilisée (`sum` par défaut). Pour changer l'intitulé tout en continuant de faire une somme, on pourra créer une nouvelle fonction (par exemple `Total`) et utiliser l'argument `FUN` de la fonction :

```
> Total <- sum # on crée une fonction Total  
> addmargins(table(diplome), FUN = Total)
```

diplome				
L1	L2	L3	Master	Total
8	12	6	4	30

```
> addmargins(prop.table(table(diplome)), FUN = Total)
```

diplome				
L1	L2	L3	Master	Total
0.27	0.40	0.20	0.13	1.00

L'intitulé de la dernière colonne est maintenant `Total`.

Mise en forme du tableau : changement d'étiquettes

De manière alternative, on pourra remplacer les étiquettes en ligne avec la fonction `names()` (on ne peut pas utiliser la fonction `colnames()` évoquée précédemment quand le tableau comporte une seule ligne).

```
> tab <- addmargins(table(diplome))  
> names(tab) <- c('Licence 1', 'Licence 2',  
+                 'Licence 3', 'Master', 'Total')  
> print(tab)
```

Licence 1	Licence 2	Licence 3	Master	Total
8	12	6	4	30

Cette option permet de changer les intitulés de toutes les colonnes.

Tableau de fréquences : regroupement des modalités

Dans le cas des variables quantitatives continues (et discrètes si elles présentent un grand nombre de valeurs possibles), il est souhaitable de faire des regroupements des valeurs de la variable en classes.

Sous R, on utilise la fonction `cut(variable, bornes)` avec bornes un vecteur contenant les bornes des classes. On pourra aussi utiliser la fonction `hist` (voir aussi la sous-section suivante).

Exemple de regroupement des modalités avec cut

Par exemple, on utilise ici la fonction `cut` pour regrouper les valeurs de la variable `argent` de poche par classes de 20€ avant de calculer les fréquences de chaque classe.

```
> breaks_argent <- seq(0,100,20) # classes  
> argent_classe <- cut(argent_poche,breaks_argent)  
> table(argent_classe)
```

```
argent_classe  
 (0,20]  (20,40]  (40,60]  (60,80]  (80,100]  
      3      12      6      6      3
```

Exemple de regroupement des modalités avec hist

Même application avec la fonction `hist`¹ :


```
> histo <- hist(argent_poche)
> print(histo$breaks) # les bornes des classes
```

```
[1] 0 20 40 60 80 100
```

```
> print(histo$counts) # les effectifs
```

```
[1] 3 12 6 6 3
```

Pour les deux fonctions `cut` et `hist`, les classes sont définies en excluant la borne inférieure et en incluant la borne supérieure $]a, b]$. Si l'on souhaite le contraire, on ajoutera l'option `right = FALSE`.

1. Par défaut, nombre de classes = $\text{ceiling}(\log(n)+1)$ selon la règle de Sturges. 

Plan du chapitre

1 Importation de données

- Importation d'une base
- Visualisation de la base et de ses caractéristiques
- Création de variables/sous-blocs de la base

2 Statistique descriptive univariée

- Résumés numériques
- Tableaux de fréquences
- Graphiques

3 Statistique descriptive bivariable

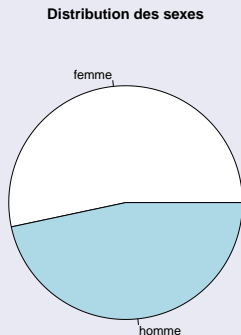
- Résumés numériques
- Graphiques
- Tableaux de fréquences

R comporte de nombreuses fonctions graphiques. Associées à la fonction `table()` décrite plus haut, elles permettront de représenter la distribution empirique des variables.

Variable qualitative nominale

La fonction `pie` sera utilisée pour les variables qualitatives nominales (diagramme en secteurs circulaires) :

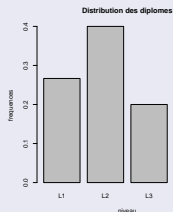
```
> pie(table(sexe),main="Distribution des sexes")
```



Variable qualitative ordinale

On utilise `barplot` pour représenter la distribution des variables qualitatives ordinales sous la forme d'un diagramme en tuyaux d'orgue :

```
> n <- 30 # nombre d'individus de la base
> barplot(prop.table(table(diplome)),
+         main="Distribution des diplomes",
+         xlab="niveau", ylab="frequences")
```

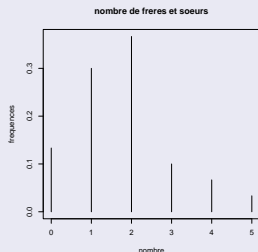


Remarque : argument `main` pour le titre, `xlab` pour l'étiquette de l'axe des abscisses, `ylab` pour l'axe des ordonnées.

Variable quantitative discrète

On utilisera un diagramme en bâtons pour la distribution d'une variable quantitative discrète. Sous R, on utilise la commande `plot` :

```
> plot(prop.table(table(fratrie)),  
+       main = "nombre de freres et soeurs",  
+       xlab="nombre",ylab="frequences")
```

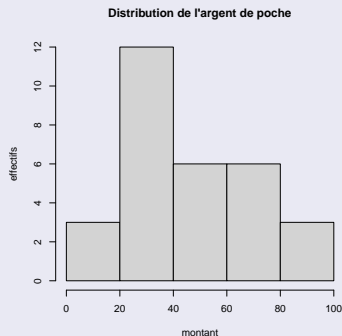


Remarque : de même, argument `main` pour le titre, `xlab` pour l'étiquette de l'axe des abscisses, `ylab` pour l'axe des ordonnées.

Variable quantitative continue

Pour une variable quantitative continue (ou discrète dont les modalités sont regroupées en classes), on utilise la commande `hist` :

```
> hist(argent_poche,  
+      main="Distribution de l'argent de poche",  
+      xlab="montant", ylab="effectifs")
```



Variable quantitative continue (suite)

On rajoutera l'option `freq = FALSE` pour afficher les fréquences relatives (les effectifs, qui se disent *frequency* en anglais, sont donnés par défaut) :

```
> hist(argent_poche, freq=FALSE,  
+      main="Distribution de l'argent de poche",  
+      xlab="montant", ylab="frequences")
```



Variable quantitative continue (suite)

Attention, lorsque l'on utilise `hist`, il n'est pas nécessaire d'utiliser la commande `table()` pour le calcul des fréquences.

Par défaut, les classes sont définies en excluant la borne inférieure et en incluant la borne supérieure $]a,b]$. Dans le cas contraire, on utilisera l'option `right = FALSE`.

Histogramme - gestion du nb et de l'amplitude des classes

On pourra utiliser l'argument `breaks` de la fonction `hist` pour définir les classes de l'histogramme :

- Pour répartir les observations en $k+1$ classes d'amplitude égale (i.e. $k+1$ rectangles sur le graphique), ajouter l'argument `breaks=k` à la fonction `hist`.
- Pour définir les classes, définir un vecteur `vecteur_bornes` contenant les bornes des classes et utiliser à nouveau l'argument `breaks=vecteur_bornes`.

Remarque : lorsque les classes sont d'amplitudes inégales, les fréquences sont corrigées automatiquement en fonction de l'amplitude des classes (de façon à ce que l'aire des rectangles soit bien proportionnelle à la fréquence de la classe représentée); il n'est pas nécessaire de faire cette correction comme sous Excel.

Gestion du nombre et de l'amplitude des classes : exemple

On utilise l'argument `breaks` de la fonction `hist`.

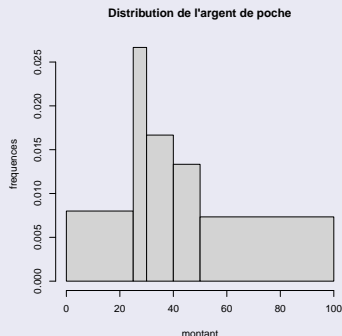
```
> hist(argent_poche, freq=FALSE,  
+      main="Distribution de l'argent de poche",  
+      xlab="montant",ylab="frequences",  
+      breaks = 4) # 5 classes de même amplitude
```



Gestion du nombre et de l'amplitude des classes : exemple

On utilise l'argument `breaks` de la fonction `hist`.

```
> hist(argent_poche,  
+      main="Distribution de l'argent de poche",  
+      xlab="montant",ylab="frequences",  
+      breaks = c(0,25,30,40,50,100)) # 5 classes inégales
```



Plan du chapitre

1 Importation de données

- Importation d'une base
- Visualisation de la base et de ses caractéristiques
- Création de variables/sous-blocs de la base

2 Statistique descriptive univariée

- Résumés numériques
- Tableaux de fréquences
- Graphiques

3 Statistique descriptive bivariable

- Résumés numériques
- Graphiques
- Tableaux de fréquences

Plan du chapitre

1 Importation de données

- Importation d'une base
- Visualisation de la base et de ses caractéristiques
- Création de variables/sous-blocs de la base

2 Statistique descriptive univariée

- Résumés numériques
- Tableaux de fréquences
- Graphiques

3 Statistique descriptive bivariable

- Résumés numériques
- Graphiques
- Tableaux de fréquences

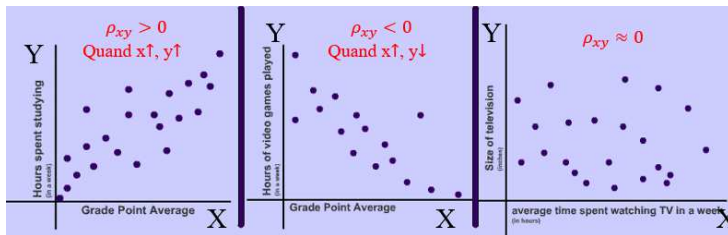
Supposons que l'on dispose de 2 variables quantitatives x et y observées sur n individus, x_i et y_i , $i = 1, \dots, n$.

Pour analyser la liaison entre ces deux variables, on pourra calculer la covariance et la corrélation des 2 variables :

- `cov(x,y)` renvoie la covariance entre x et y
- `cor(x,y)` renvoie la corrélation entre x et y

Rappel : le coefficient de corrélation

- Avantages de la corrélation ρ_{XY} par rapport à la covariance : 1) nombre sans unité donc permet de comparer des phénomènes mesurés dans des unités différentes ; 2) nombre borné, compris entre -1 et 1.
- Interprétation : le signe renseigne sur le sens de la relation, sa valeur absolue sur l'intensité de la relation (linéaire du type $Y = aX + b$) entre X et Y. Plus sa valeur absolue est proche de 1, plus la liaison entre X et Y est forte.



Plan du chapitre

1 Importation de données

- Importation d'une base
- Visualisation de la base et de ses caractéristiques
- Création de variables/sous-blocs de la base

2 Statistique descriptive univariée

- Résumés numériques
- Tableaux de fréquences
- Graphiques

3 Statistique descriptive bivariable

- Résumés numériques
- Graphiques
- Tableaux de fréquences

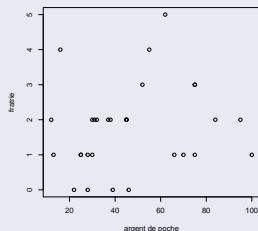
On pourra également analyser la liaison entre 2 variables (quantitatives) (x,y) en utilisant un nuage de points. Chaque point de cette représentation correspond à un individu i de la base et a pour abscisse x_i (la valeur de x pour l'individu i) et pour ordonnée y_i (la valeur de y pour l'individu i)

Nuage de points - exemple

Exemple pour le couple fratrie et argent de poche de la base :

Exemple de nuage de points

```
> plot(argent_poche,fratrie,  
+       xlab="argent de poche",ylab="fratrie")
```



Il n'y a pas de liaison évidente entre les 2 variables.

Plan du chapitre

1 Importation de données

- Importation d'une base
- Visualisation de la base et de ses caractéristiques
- Création de variables/sous-blocs de la base

2 Statistique descriptive univariée

- Résumés numériques
- Tableaux de fréquences
- Graphiques

3 Statistique descriptive bivariable

- Résumés numériques
- Graphiques
- Tableaux de fréquences

Tableau de contingence

Le **tableau de contingence** décrit la distribution jointe d'un couple de variables $\{X, Y\}$:

Tableau de contingence de X et Y

$X \backslash Y$	y_1	\dots	y_j	\dots	y_c	Total
x_1	n_{11}	\dots	n_{1j}	\dots	n_{1c}	$n_{1\bullet}$
\vdots	\vdots		\vdots		\vdots	\vdots
x_i	n_{i1}	\dots	n_{ij}	\dots	n_{ic}	$n_{i\bullet}$
\vdots	\vdots		\vdots		\vdots	\vdots
x_I	n_{I1}	\dots	n_{Ij}	\dots	n_{Ic}	$n_{I\bullet}$
Total	$n_{\bullet 1}$	\dots	$n_{\bullet j}$	\dots	$n_{\bullet c}$	n

avec n_{ij} le nombre d'individus présentant à la fois la modalité x_i et la modalité y_j (effectifs partiels) et $n_{i\bullet}$ et $n_{\bullet j}$ les effectifs de la modalité x_i et y_j (effectifs marginaux).

Tableau de contingence sous R

On utilise la commande `table(x,y)` pour un couple de variables x et y.

```
> table(sexe,diplome)
```

	diplome			
sexe	L1	L2	L3	Master
femme	4	7	3	2
homme	4	5	3	2

```
> addmargins(table(sexe,diplome))
```

	diplome				
sexe	L1	L2	L3	Master	Sum
femme	4	7	3	2	16
homme	4	5	3	2	14
Sum	8	12	6	4	30

De même, on utilisera la commande `addmargins()` pour ajouter les effectifs marginaux (dernière ligne et dernière colonne du tableau).

Tableau de contingence sous R

De même, on pourra renommer les dernières ligne et colonne créées par `addmargins` :

```
> Total <- sum  
> addmargins(table(Base$sexe,Base$diplome),FUN = Total)
```

Margins computed over dimensions
in the following order:

1:

2:

	L1	L2	L3	Master	Total
femme	4	7	3	2	16
homme	4	5	3	2	14
Total	8	12	6	4	30

Tableau des profils-lignes

Le **tableau des profils-lignes** présente la distribution de la variable Y conditionnellement à la variable X . Il est obtenu en rapportant les effectifs n_{ij} aux effectifs de la dernière colonne $n_{i\bullet}$ du tableau de contingence.

Tableau des profils-lignes

$X \backslash Y$...	y_j	...	Total
x_i	...	$f_{j i}$...	1
Tout i	...	$f_{\bullet j}$...	1

$f_{j|i} = n_{ij}/n_{i\bullet}$ représente la proportion d'individus présentant la modalité y_j sachant qu'ils présentent la modalité x_i . La dernière ligne du tableau reporte les fréquences marginales $f_{\bullet j} = n_{\bullet j}/n$ de la variable Y .

Tableau des profils-lignes sous R

On pourra construire le tableau des profils-lignes en spécifiant `margin = 1` (ou simplement 1) à la fonction `prop.table`.

```
> tab_conting <- table(sexe,diplome)
> prop.table(tab_conting,1)
```

	diplome			
sexe	L1	L2	L3	Master
femme	0.25	0.44	0.19	0.12
homme	0.29	0.36	0.21	0.14

```
> addmargins(prop.table(addmargins(tab_conting,1),1),2)
```

	diplome				
sexe	L1	L2	L3	Master	Sum
femme	0.25	0.44	0.19	0.12	1.00
homme	0.29	0.36	0.21	0.14	1.00
Sum	0.27	0.40	0.20	0.13	1.00

Tableau des profils-lignes sous R

De même, on pourra renommer les dernières ligne et colonne créées par `addmargins` :

```
> Total <- sum
> Tout <- sum
> TC <- table(sexe,diplome)
> addmargins(prop.table(addmargins(TC,1,FUN=Tout),1),2,
+             FUN=Total)
```

	diplome				
sexe	L1	L2	L3	Master	Total
femme	0.25	0.44	0.19	0.12	1.00
homme	0.29	0.36	0.21	0.14	1.00
Tout	0.27	0.40	0.20	0.13	1.00

Attention à l'ordre des étiquettes !

Mise en forme du tableau : changement d'étiquettes

De manière alternative, on pourra remplacer les étiquettes en ligne et en colonne avec les fonctions `colnames` et `rownames`.

```
tab <- addmargins(prop.table(addmargins(TC,1),1),2)
colnames(tab) <- c('L1','L2','L3','master','Total')
rownames(tab) <- c('femmes','hommes','Tout')
```

Tableau des profils-colonnes

Le **tableau des profils-colonnes** est obtenu en rapportant les effectifs n_{ij} aux effectifs de la dernière ligne $n_{\bullet j}$ du tableau de contingence. Il décrit la distribution de la variable X conditionnellement à la variable Y .

Tableau des profils-colonnes

$X \backslash Y$...	y_j	...	Tout j
x_i	...	$f_{i j}$...	$f_{i\bullet}$
Total		1		1

$f_{i|j} = n_{ij}/n_{\bullet j}$ représente la proportion d'individus présentant la modalité x_i parmi les individus présentant la modalité y_j . La dernière colonne du tableau contient les fréquences marginales $f_{i\bullet} = n_{i\bullet}/n$ de la variable X .

Tableau des profils-colonnes sous R

On pourra construire le tableau des profils-colonnes en spécifiant `margin = 2` (ou simplement 2) à la fonction `prop.table`.

```
> TC <- table(sexe,diplome)
> addmargins(prop.table(addmargins(TC,2),2),1)
```

	diplome				
sexe	L1	L2	L3	Master	Sum
femme	0.50	0.58	0.50	0.50	0.53
homme	0.50	0.42	0.50	0.50	0.47
Sum	1.00	1.00	1.00	1.00	1.00

Tableau des profils-colonnes sous R

De même, on pourra renommer les dernières ligne et colonne créées par `addmargins` en modifiant la valeur de `FUN` ou en utilisant les fonctions `colnames` et `rownames`. Par exemple, en suivant la première stratégie :

```
> TC <- table(sexe,diplome)
> addmargins(prop.table(addmargins(TC,2,FUN=Tout),2),
+           1,FUN=Total)
```

	diplome				
sexe	L1	L2	L3	Master	Tout
femme	0.50	0.58	0.50	0.50	0.53
homme	0.50	0.42	0.50	0.50	0.47
Total	1.00	1.00	1.00	1.00	1.00

Langage R pour la statistique

Chapitre 3 - Éléments de programmation

M. Bessec, S. Dufour & M. Menéndez

Septembre 2024

- Manuel en ligne : Rcompanion : <https://rcompanion.org>
- Decourt O. (2018). Le langage R au quotidien : Traitement et analyse de données volumineuses. Dunod.
- Maumy-Bertrand M. et Bertrand F. (2018). Initiation à la statistique avec R. Dunod.

Plan du chapitre

1 Manipulation des vecteurs et matrices

- Vecteur
- Matrice

2 La condition if

3 Boucles for et while

- Boucle while
- Boucle for

4 Les fonctions

5 Fonctions prédéfinies sous R

- Fonctions mathématiques
- Fonctions de statistique descriptive
- Fonctions de probabilité

Plan du chapitre

1 Manipulation des vecteurs et matrices

- Vecteur
- Matrice

2 La condition if

3 Boucles for et while

- Boucle while
- Boucle for

4 Les fonctions

5 Fonctions prédéfinies sous R

- Fonctions mathématiques
- Fonctions de statistique descriptive
- Fonctions de probabilité

- Comme indiqué précédemment, on manipule différentes classes d'objets sous R (vector, matrix, array, factor, time series, data.frame, list etc).
- Nous nous intéressons plus spécifiquement ici aux vecteurs et aux matrices; il s'agit de lignes ou de tableaux d'éléments de même mode (par opposition aux objets *data.frame* et *list* qui peuvent contenir des éléments de nature différente).

Plan du chapitre

1 Manipulation des vecteurs et matrices

- Vecteur
- Matrice

2 La condition if

3 Boucles for et while

- Boucle while
- Boucle for

4 Les fonctions

5 Fonctions prédéfinies sous R

- Fonctions mathématiques
- Fonctions de statistique descriptive
- Fonctions de probabilité

Déclaration d'un vecteur

Un **vecteur** est une ligne d'éléments de même mode (il ne pourra par exemple pas contenir à la fois des éléments numériques et des chaînes de caractères).

Pour **déclarer un vecteur**, on indiquera son type et sa longueur, i.e. le nombre d'éléments qu'il contiendra (en pratique, si cette étape est omise, R devinera le type du vecteur à partir de la nature des objets attribués).

Exemples

```
> x1 <- vector(mode="numeric",length=4)
> # déclare un vecteur numérique de taille 4
```

```
> x2 <- vector(mode="character",length=3)
> # déclare un vecteur de caractères de taille 3
```

```
> x3 <- vector(mode="logical",length=5)
> # déclare un vecteur booléen de taille 5
```

Attribution de valeurs à un vecteur

On peut utiliser la fonction `c()` avec entre les parenthèses les différents éléments (de même type) séparés par des virgules

```
> x1 <- c(3,6,9,0)
```

```
> x2 <- c("janvier","février","mars")
```

```
> x3 <- c(TRUE,FALSE,TRUE,TRUE,TRUE)
```

Les trois vecteurs contiennent des éléments de même type.

- `x1[2]` \rightsquigarrow 2e élément de `x1` (dans l'exemple, 6)
- `x1[2:4]` \rightsquigarrow éléments 2 à 4 de `x1` (dans l'exemple, 6,9,0)
- `x1[c(4,1)]` \rightsquigarrow éléments 4 et 1 de `x1` (dans l'exemple, 0,3)
- `x1[x1>5]` \rightsquigarrow éléments de `x1` > 5 (dans l'exemple : 6,9)
- `x1[c(TRUE,FALSE,FALSE,FALSE)]` \rightsquigarrow élément 1 de `x1` (dans l'exemple : 3)

Fonctions utiles pour manipuler des vecteurs

- `rep(x,n)` \rightsquigarrow vecteur de `x` de dimension `n`
- `length(x)` \rightsquigarrow longueur du vecteur `x`
- `rev(x)` \rightsquigarrow inverse l'ordre des éléments du vecteur `x`
- `sort(x)` \rightsquigarrow vecteur `x` trié en ordre croissant (ou alphabétique);
option `decreasing = TRUE` ou `T` pour un ordre décroissant
- `sample(1:100,5,replace = T)` \rightsquigarrow vecteur contenant 5 entiers tirés
au hasard et avec remise entre 1 et 100 (le tirage est sans remise si
`replace = F`).
- `seq(min,max,pas)` \rightsquigarrow suite arithmétique de raison `pas`, de premier
terme `min` et de dernier terme `max`; si le `pas` est omis, il vaut 1 par
défaut (dans ce cas, on peut aussi utiliser simplement `min:max`).

Exemple

```
> print(rep(0,4))
```

```
[1] 0 0 0 0
```

```
> print(sample(1:10,10,replace=FALSE))
```

```
[1] 5 3 7 9 1 4 8 6 10 2
```

```
> print(seq(2,11,3))
```

```
[1] 2 5 8 11
```

```
> print(2:10)
```

```
[1] 2 3 4 5 6 7 8 9 10
```

On pourra faire des **opérations** usuelles sur les vecteurs :

- $a+b \rightsquigarrow$ addition des vecteurs a et b
- $a-b \rightsquigarrow$ différence des vecteurs a et b
- $a*b \rightsquigarrow$ produit *élément par élément*
- $a/b \rightsquigarrow$ division *élément par élément*
- $c(a,b) \rightsquigarrow$ concaténation des vecteurs a et b

Dans les 4 premiers cas, les vecteurs doivent être de même dimension.

Exemple

```
> a = c(1,2,3,4)
> b = c(2,3,4,5)
> print(a+b) # addition
```

```
[1] 3 5 7 9
```

```
> print(a*b) # produit élément par élément
```

```
[1] 2 6 12 20
```

```
> print(c(a,b)) # concaténation
```

```
[1] 1 2 3 4 2 3 4 5
```

Plan du chapitre

1 Manipulation des vecteurs et matrices

- Vecteur
- Matrice

2 La condition if

3 Boucles for et while

- Boucle while
- Boucle for

4 Les fonctions

5 Fonctions prédéfinies sous R

- Fonctions mathématiques
- Fonctions de statistique descriptive
- Fonctions de probabilité

Déclaration et remplissage d'une matrice

Une **matrice** est un tableau d'éléments de même type (le tableau est bidimensionnel ; on utilisera un `array` pour des ensembles de dimension plus élevée).

Pour **déclarer et affecter des valeurs** à une matrice, on utilisera la fonction `matrix` avec comme arguments les éléments de la matrice, le nombre de lignes et le nombre de colonnes.

```
> mat <- matrix(c(1,2,4,3,9,6), nrow = 2, ncol = 3)
> print(mat)
```

	[,1]	[,2]	[,3]
[1,]	1	4	9
[2,]	2	3	6

Par défaut, le remplissage de la matrice se fait colonne par colonne, i.e. 1 et 2 dans la 1ère colonne 1, 4 et 3 dans la 2ème colonne et 9 et 6 dans la 3ème colonne (remplissage en ligne avec l'argument `byrow=TRUE`).

Déclaration et remplissage d'une matrice

Quelques raccourcis utiles pour générer des matrices particulières :

- la matrice identité

```
> diag(2)
```

	[,1]	[,2]
[1,]	1	0
[2,]	0	1

- une matrice de 0 (remplacer 0 par l'élément que l'on veut répéter)

```
> matrix(0,2,3)
```

	[,1]	[,2]	[,3]
[1,]	0	0	0
[2,]	0	0	0

Accès aux éléments d'une matrice

Pour accéder à un élément, une ligne ou une colonne d'une matrice, on utilise une syntaxe similaire à celle pour les vecteurs :

- `mat[2,1]` \rightsquigarrow élément de la 2e ligne et 1ère colonne de `mat`
- `mat[2,]` \rightsquigarrow 2e ligne de `mat`
- `mat[,1]` \rightsquigarrow 1ère colonne de `mat`
- `mat[,1:3]` \rightsquigarrow colonnes 1 à 3 de `mat`
- `mat[,c(1,3)]` \rightsquigarrow 1ère et 3e colonnes de `mat`

Ajout/suppression des éléments d'une matrice

- `new_mat1 <- cbind(mat, c(3,10))` pour ajouter une colonne qui contiendra 3 et 10
- `new_mat2 <- rbind(mat, c(1,5,10))` pour ajouter une ligne qui contiendra 1,5 et 10
- `new_mat3 <- mat[-1,]` pour supprimer la 1ère ligne
- `new_mat4 <- mat[, -c(1,3)]` pour supprimer les colonnes 1 et 3
- `new_mat5 <- mat[-1, -2]` pour supprimer la 1ère ligne et la 2e colonne

Fonctions utiles sur une matrice

- `dim(mat)` \rightsquigarrow renvoie le nombre de lignes et de colonnes de `mat` (la fonction `length` renverra le produit de ces deux nombres)
- `nrow(mat)` \rightsquigarrow renvoie le nombre de lignes de `mat`
- `ncol(mat)` \rightsquigarrow renvoie le nombre de colonnes de `mat`
- `det(mat)` \rightsquigarrow calcule le déterminant de `mat`
- `t(mat)` \rightsquigarrow transpose `mat`
- `solve(mat)` \rightsquigarrow calcule l'inverse de `mat`

Enfin, on peut faire les opérations usuelles sur les matrices en s'assurant du respect des conditions sur leurs dimensions :

- $A+B \rightsquigarrow$ addition des matrices A et B
- $A-B \rightsquigarrow$ différence des matrices A et B
- $A*B \rightsquigarrow$ produit élément par élément de A et B
- $A/B \rightsquigarrow$ division élément par élément de A et B
- $A\%*B \rightsquigarrow$ produit matriciel de A et B

Dans les 4 premiers cas, A et B doivent avoir la même dimension ; dans le dernier cas, le nombre de colonnes de A doit être égal au nombre de lignes de B.

Plan du chapitre

1 Manipulation des vecteurs et matrices

- Vecteur
- Matrice

2 La condition if

3 Boucles for et while

- Boucle while
- Boucle for

4 Les fonctions

5 Fonctions prédéfinies sous R

- Fonctions mathématiques
- Fonctions de statistique descriptive
- Fonctions de probabilité

- On utilise la commande `if` pour n'exécuter un bloc de commandes que si une condition est satisfaite.
- Par exemple, on tire au hasard un nombre et on affiche ce nombre s'il est pair.
- Il est possible de définir un bloc de commandes alternatif si la condition n'est pas satisfaite en utilisant les commandes `else` ou `else if`.

Plusieurs syntaxes sont possibles :

```
if (condition) {instructions}
```

```
if (condition) {instructions} else if (condition) {instructions}
```

```
if (condition) {instructions} else if (condition) {instructions} else {instructions}
```

```
if (condition) {instructions} else {instructions}
```

Exemple

Dans l'exemple qui suit, on affiche l'humeur du jour suivant que l'on est ou non en début de semaine :

```
> date <- Sys.Date() # date du jour
> jour <- weekdays(date) # jour de la semaine
> if (jour=="lundi"){print("Je suis de mauvaise humeur")}
+ } else {print("La vie est belle")}
```

```
[1] "La vie est belle"
```

Attention à la **position des accolades** lorsque l'on écrit les instructions conditionnelles sur plusieurs lignes et que l'on utilise `else` ou `else if`.

Les accolades doivent apparaître devant `else` et `else if` (sinon, R considère l'instruction conditionnelle terminée et ne comprend pas le `sinon`).

On aura par **exemple** dans la syntaxe 3 :

```
if (condition) {instructions  
} else if (condition) {instructions  
} else {instructions}
```


Opérateur de comparaison

Des **opérateurs de comparaison** peuvent être utiles pour définir des conditions portant par exemple sur deux variables a et b :

- égalité : if $a == b$
- inégalité : if $a != b$
- supérieur (ou égal) : if $a > b$ (if $a >= b$)
- inférieur (ou égal) : if $a < b$ (if $a <= b$)

Pour **combinaison** plusieurs conditions, opérateurs $\&$ (pour et) | (pour ou) :

- if $(a == b) \& (a == c)$ les deux conditions doivent être satisfaites
- if $(a == b) | (a == c)$ au moins une des conditions doit être satisfaite

Dans le premier cas, l'instruction sera exécutée si la variable a est égale à la variable b et à la variable c . Dans le second cas, une seule égalité suffit.

Plan du chapitre

1 Manipulation des vecteurs et matrices

- Vecteur
- Matrice

2 La condition if

3 Boucles for et while

- Boucle while
- Boucle for

4 Les fonctions

5 Fonctions prédéfinies sous R

- Fonctions mathématiques
- Fonctions de statistique descriptive
- Fonctions de probabilité

Pour répéter certains blocs de commandes, on pourra utiliser une boucle `while` ou un boucle `for` (on peut aussi utiliser `repeat` sous R qui n'est pas présenté ici).

Boucle while

- La **boucle while** exécute des instructions tant qu'une condition est satisfaite. On sort de la boucle (sans exécuter le bloc d'instructions) lorsque la condition n'est plus valide.
- **Syntaxe** : `while (condition) {instructions}`
- **Exemple** :

```
> x <- 0  
> while (x<30){  
+   x <- x+10  
+   print(x)  
+ }
```

```
[1] 10  
[1] 20  
[1] 30
```

- La **boucle** `for` permet de répéter des instructions un nombre prédéfini de fois. On définit un vecteur dont on parcourt les éléments; lorsque le dernier élément est atteint, on parcourt une dernière fois les instructions dans la boucle et l'on en sort.
- **Syntaxe** : `for` (var in vecteur) {instructions}
- **Exemple** :

```
> for (i in 1:3) {print("bonjour")}
```

```
[1] "bonjour"  
[1] "bonjour"  
[1] "bonjour"
```

Les fonctions `break` ou `skip` permettent de forcer la sortie de la boucle ou de sauter une itération.

Plan du chapitre

1 Manipulation des vecteurs et matrices

- Vecteur
- Matrice

2 La condition if

3 Boucles for et while

- Boucle while
- Boucle for

4 Les fonctions

5 Fonctions prédéfinies sous R

- Fonctions mathématiques
- Fonctions de statistique descriptive
- Fonctions de probabilité

- Il existe un grand nombre de fonctions prédéfinies sous R (par exemple, `mean()` qui renvoie la moyenne d'un vecteur)...
- ... mais l'utilisateur peut aussi souhaiter définir ses propres fonctions.
- Ces fonctions peuvent (ou non) avoir des arguments et peuvent (ou non) renvoyer des valeurs.

- **Définition d'une fonction**

```
nom_fonction <- function(x1,x2,...){ instructions return(y)}
```

- **Remarques**

- ▶ Si la fonction ne comporte pas d'argument, écrire simplement `function()`.
- ▶ Si la fonction ne retourne pas d'output, omettre la ligne `return`.
- ▶ Dans le cas contraire, la fonction ne retourne qu'un seul output mais il est possible, si l'on souhaite retourner plusieurs éléments, de créer un objet qui est une liste d'éléments et de retourner cette liste.

- **Appel d'une fonction** : écrire le nom de la fonction suivie entre parenthèses de ses arguments ou de parenthèses vides à défaut (si on omet les parenthèses, on affiche le code de la fonction).

- La fonction doit être compilée une première fois avant d'être utilisée dans un script ou être définie dans le script.
- Pour alléger nos programmes, on peut enregistrer une fonction dans un script séparé 'nom_fichier.R' et appeler cette fonction avec `source('nom_fichier.R')` de façon similaire à l'appel des packages au début des programmes.
- Pour détruire la fonction, on utilisera `rm(nom_fonction)`.

Exemple 1

Dans l'exemple suivant, la fonction `absolu()` calcule la valeur absolue d'une variable `x` (en pratique, on utilisera directement `abs(x)` qui fait le même travail!) :

```
> absolu <- function(x){  
+   if (x<0){y=-x} else {y=x}  
+   return(y)  
+ }  
> absolu(-5)
```

```
[1] 5
```

Exemple 2

Dans notre deuxième exemple, la fonction `puissance()` élève le nombre x à la puissance y et affiche le nombre obtenu :

```
> puissance <- function(x,y){  
+   z   <- x^y  
+   print(z)  
+ }  
> puissance(4,2)
```

```
[1] 16
```

Ici, la fonction n'a pas d'output.

Exemple 3

Dans le dernier exemple, la fonction `message_accueil()` affiche simplement un message. Elle ne comporte ni input ni output :

```
> message_accueil <- function(){  
+   print("Coucou!")  
+ }  
> message_accueil()
```

```
[1] "Coucou!"
```

Elle sera appelée en écrivant `message_accueil()`.

Remarque : statut des variables dans une fonction

- Toutes les variables définies en dehors de la fonction sont **globales** et peuvent être utilisées dans la fonction (même si elles ne sont pas données dans les arguments).
- En revanche, les variables définies dans les fonctions sont dites **locales**, i.e. n'existent pas en dehors de la fonction (à moins de leur donner un statut global en utilisant l'opérateur d'affectation `<< -` au lieu de `< -`).

Plan du chapitre

1 Manipulation des vecteurs et matrices

- Vecteur
- Matrice

2 La condition if

3 Boucles for et while

- Boucle while
- Boucle for

4 Les fonctions

5 Fonctions prédéfinies sous R

- Fonctions mathématiques
- Fonctions de statistique descriptive
- Fonctions de probabilité

Commande R	Résultat	Exemple
<code>sqrt(x)</code>	\sqrt{x}	<code>sqrt(4) => 2</code>
<code>abs(x)</code>	$ x $	<code>abs(-4) => 4</code>
<code>log(x, base=y)</code>	$\log(x)$	<code>log(1, base=0) => 0</code>
<code>log(x)</code>	$\ln(x)$	<code>log(1) => 0</code>
<code>factorial(x)</code>	$x!$	<code>factorial(3) => 6</code>
<code>floor(x)</code>	entier inférieur	<code>floor(1.1) => 1</code>
<code>ceiling(x)</code>	entier supérieur	<code>ceiling(1.1) => 2</code>
<code>round(x)</code>	entier le + proche	<code>round(1.1) => 1</code>
<code>min(x)</code>	élément minimum	<code>min(c(8,1,5)) => 1</code>
<code>max(x)</code>	élément maximum	<code>max(c(8,1,5)) => 8</code>
<code>a%%n</code>	modulo a mod n	<code>7%%3 => 1</code>

Commande R	Résultat
<code>mean(x)</code>	$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$
<code>sum(x)</code>	$\sum_{i=1}^n x_i$
<code>prod(x)</code>	$\prod_{i=1}^n x_i$
<code>quantile(x, probs=p)</code>	y tel que $P(X < y) = p$
<code>median(x)</code>	médiane
<code>var(x)</code>	$s'^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$
<code>sd(x)</code>	écart-type corrigé s'
<code>IQR(x)</code>	écart interquartile Q3-Q1
<code>summary(x)</code>	min, Q1, Q2, \bar{x} , Q3, max

De façon générale, on utilise les préfixes `d`, `p`, `q` et `r` que l'on accole au nom de la distribution `nom_dist` :

- `d` : pour la fonction de masse (loi discrète) ou de densité (loi continue)
- `p` : pour la fonction de répartition
- `q` : pour les quantiles
- `r` : pour générer aléatoirement des réalisations indépendantes de `nom_dist`.

avec `nom_dist = binom` pour la loi binomiale, `pois` pour la loi de Poisson, `norm` pour la loi normale, `t` pour la loi de Student, `chisq` pour la loi du Chi-deux et `f` pour la loi de Fisher.

Fonctions de probabilité (suite)

Loi binomiale $X \sim B(n, q)$	<code>dbinom(x,n,q)</code> <code>pbinom(x,n,q)</code> <code>qbinom(p,n,q)</code>	$P(X = x)$ $P(X \leq x)$ Plus petit x tq $P(X \leq x) \geq p$
Loi de Poisson $X \sim P(\lambda)$	<code>dpois(x,λ)</code> <code>ppois(x,λ)</code> <code>qpois(p,λ)</code>	$P(X=x)$ $P(X \leq x)$ Plus petit x tel que $P(X \leq x) \geq p$
Loi normale $X \sim N(\mu, \sigma)$	<code>dnorm(x,μ,σ)</code> <code>pnorm(x,μ,σ)</code> <code>qnorm(p,μ,σ)</code>	$f(x)$ $F(x) = P(X \leq x)$ x tel que $F(x) = p$
Loi normale centrée réduite $X \sim N(0, 1)$	<code>dnorm(x)</code> <code>pnorm(x)</code> <code>qnorm(p)</code>	$f_0(x)$ $F_0(x) = P(X \leq x)$ x tel que $F_0(x) = p$
Loi de Student $X \sim St(\nu)$	<code>dt(x,ν)</code> <code>pt(x,ν)</code> <code>qt(p,ν)</code>	$f(x)$ $F(x) = P(X \leq x)$ x tel que $F(x) = p$
Loi du chi-deux $X \sim \chi^2(\nu)$	<code>dchisq(x,ν)</code> <code>pchisq(x,ν)</code> <code>qchisq(p,ν)</code>	$f(x)$ $F(x) = P(X \leq x)$ x tel que $F(x) = p$
Loi de Fisher $X \sim F(\nu_1, \nu_2)$	<code>df(x,ν₁,ν₂)</code> <code>pf(x,ν₁,ν₂)</code> <code>qf(p, ν₁, ν₂)</code>	$f(x)$ $F(x) = P(X \leq x)$ x tel que $F(x) = p$

Langage R pour la statistique

Annexe - Rapport dynamique avec R Markdown

M. Bessec, S. Dufour & M. Menéndez

Septembre 2024

On pourra écrire des rapports sur R Markdown pour présenter dans un document unique :

- du texte
- des codes R
- les résultats des codes (graphique, tableau, régression, etc)

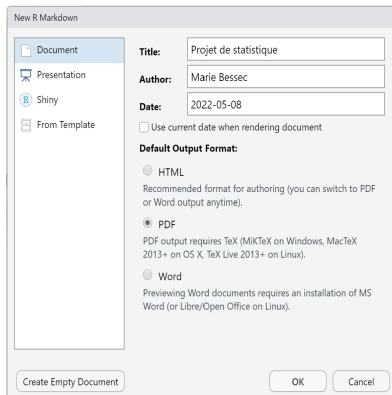
On aura un document dont les traitements sont transparents et reproductibles et qui pourra être mis à jour facilement en cas de changements de données.

Lors de la première utilisation, il faudra avoir installé les packages `knitr`, `markdown` et `rmarkdown`.

Source : <https://delladata.fr/guide-de-demarrage-en-r-markdown/>

Mode d'emploi : création du document R markdown

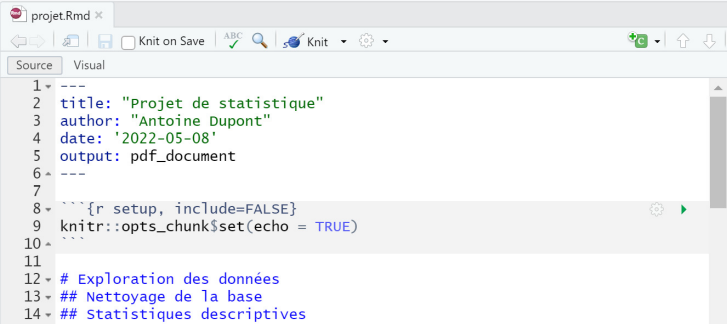
File/New File/R Markdown puis préciser le titre du document, son auteur et le format du fichier de sortie (ici un document en format pdf).



NB : pour le format pdf, il faut disposer de LATEX sur son ordinateur ; sinon exécuter à la première utilisation la commande `tinytex::install_tinytex()` dans la console.

Mode d'emploi : création du document R markdown

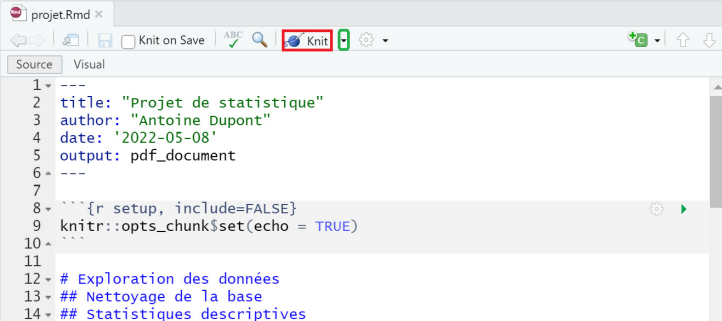
Un fichier est créé qui pourra être enregistré avec File/Save as; le fichier aura une extension .Rmd (ici projet.Rmd).



```
1 ---
2 title: "Projet de statistique"
3 author: "Antoine Dupont"
4 date: '2022-05-08'
5 output: pdf_document
6 ---
7
8 ```{r setup, include=FALSE}
9 knitr::opts_chunk$set(echo = TRUE)
10 ```
11
12 # Exploration des données
13 ## Nettoyage de la base
14 ## Statistiques descriptives
```

Mode d'emploi : générer le document final

Pour générer le rapport (un document pdf dans notre exemple), cliquer sur Knit en haut de l'écran (la petite pelote de laine) :



```
1 ---
2 title: "Projet de statistique"
3 author: "Antoine Dupont"
4 date: '2022-05-08'
5 output: pdf_document
6 ---
7
8 ```{r setup, include=FALSE}
9 knitr::opts_chunk$set(echo = TRUE)
10 ```
11
12 # Exploration des données
13 ## Nettoyage de la base
14 ## Statistiques descriptives
```

si l'on clique sur la petite flèche (encadrée en vert) juste à côté, il est possible de modifier le format de la sortie, qui sera par défaut celui choisi précédemment, pdf dans notre exemple)

Mode d'emploi : entête du document

L'entête du document se situe au début entouré de trois tirets :

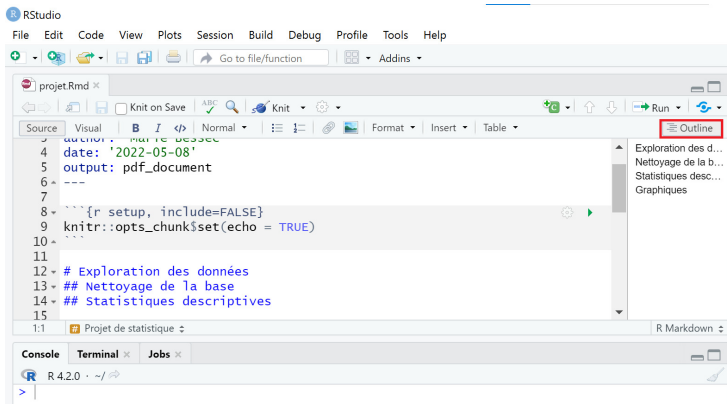
```
1 ---  
2 title: "Projet de statistique"  
3 author: "Antoine Dupont"  
4 date: '2022-05-08'  
5 output: pdf_document  
6 ---
```

On pourra y préciser le titre, l'auteur, la date, le format du fichier de sortie, etc

Mode d'emploi : structure du document

pour une section, ## pour une sous-section, ### pour une sous-sous-section

On pourra visualiser le plan du document en cliquant sur le bouton sur le côté :



Du texte pourra être inséré partout en dehors des chunks (bloc de codes voir ci-dessous).

Mode d'emploi : insertion de formule dans le texte

Il est possible d'écrire des formules mathématiques dans le texte. Si vous êtes familiers de Latex, vous retrouverez la syntaxe qui y est utilisée. On mettra les formules entre `$$`. Quelques exemples de syntaxes suivent avec le rendu à droite dans le rapport :

`H_0`

$$H_0$$

`x_t^2`

$$x_t^2$$

`$\alpha, \beta, \sigma, \chi^2_n$`

$$\alpha, \beta, \sigma, \chi_n^2$$

`$\sum_{i=1}^n X_i$`

$$\sum_{i=1}^n X_i$$

`$\frac{1}{n}$`

$$\frac{1}{n}$$

`\sqrt{n}`

$$\sqrt{n}$$

`$\overline{X} \sim N(\mu, \frac{\sigma}{\sqrt{n}})$`

$$\overline{X} \sim N\left(\mu, \frac{\sigma}{\sqrt{n}}\right)$$

`$\overline{X} \overset{p}{\rightarrow} m$`

$$\overline{X} \overset{p}{\rightarrow} m$$

`$\left\{ \begin{array}{c} H_0: p=p_0 \\ H_1: p=p_1 \end{array} \right\}$`

$$\left\{ \begin{array}{l} H_0 : p = p_0 \\ H_1 : p = p_1 \end{array} \right.$$

`$\right.$`

Mode d'emploi : insertion de formule dans le texte

Pour définir une équation qui sera centrée dans le texte, on encadrera la formule par `\[formule \]` au lieu des dollars. On obtiendra par exemple l'affichage suivant dans le rapport pour le jeu d'hypothèses précédent :

$$\begin{cases} H_0 : p = p_0 \\ H_1 : p = p_1 \end{cases}$$

Mode d'emploi : mise en forme du texte

Liste numérotée

1. Premier élément
2. Deuxième élément

Listes de puces

- * Premier élément
 - + sous élément a
 - + sous élément b
- * Deuxième élément

****élément en gras****

élément en italique

Liste numérotée

1. Premier élément
2. Deuxième élément


Listes de puces

- Premier élément
 - sous élément a
 - sous élément b
- Deuxième élément

élément en gras

élément en italique

Mode d'emploi : code - affichage

Les blocs de code R (ou chunk) sont entourés de “`{r} CODE`”. Pour insérer un chunk, cliquer sur  ou utiliser le raccourci CTRL + ALT + i

```
```{r}  
plot(x)
```
```

On peut éventuellement donner un nom à chaque bloc (dans l'exemple ci-dessous, `code1` :

```
```{r code1}  
x <- rnorm(10)
summary(x)
```
```

ce qui, en cas d'erreur, peut faciliter la recherche d'erreur.

Il existe différentes possibilités d'affichage du code et des résultats :

- `include (TRUE/FALSE)` \rightsquigarrow pour afficher ou non le code et ses résultats
- `echo (TRUE/FALSE)` \rightsquigarrow pour afficher ou non le code (TRUE par défaut)
- `results ("markup"/"hide")` \rightsquigarrow pour afficher ou non les résultats du code (par défaut, "markup")
- `warning (TRUE/FALSE)` \rightsquigarrow pour afficher ou pas les avertissements générés par le code
- `message (TRUE/FALSE)` \rightsquigarrow pour afficher ou pas les messages générés par le code

Mode d'emploi : code

On génère ici un vecteur gaussien de taille 10 et on calcule des statistiques descriptives. Les différentes possibilités d'affichage dans le rapport sont illustrées ici :

On affiche le code et les résultats:

```
```{r}  
x <- rnorm(10)
summary(x)
```
```

On affiche les résultats mais pas le code:

```
```{r,echo=FALSE}  
summary(x)
```
```

On affiche le code mais pas les résultats:

```
```{r,results="hide"}  
summary(x)
```
```

On n'affiche ni le code ni les résultats:

```
```{r,echo=FALSE,results="hide"}  
summary(x)
```
```

Dans le dernier cas, on pourra utiliser aussi `include=F`.

Mode d'emploi : options globales du code

Il est possible de définir ces options d'affichage dans un chunk au début du document en utilisant la commande `knitr::opts_chunk$set()`. Ces choix s'appliqueront à tous les blocs de code du document.

```
1 |---  
2 title: "Projet de statistique"  
3 author: "Antoine Dupont"  
4 date: '2022-05-08'  
5 output: pdf_document  
6 |---  
7  
8 ```{r setup, include=FALSE}  
9 knitr::opts_chunk$set(echo = TRUE)  
10 ```
```

Mode d'emploi : insertion de figure

On pourra insérer des figures

```
```{r}  
plot(x)
```

On choisit la dimension de la figure avec les arguments `fig.width` et `fig.height`.

```
```{r, fig.width=3,fig.height=3}  
plot(x)  
```
```