Etudes Statistiques et introduction à R

L2 MIASHS - UBO

Juliette Legrand

Janvier-Avril 2020

Partie 1. Introduction à R

1 Introduction

R est un langage de programmation interactif interprété et orienté objet qui contient une très large collection de méthodes statistiques ainsi que des facilités graphiques importantes. Initié dans les années 90 par Robert Gentleman et Ross Ihaka (Département de Statistique, Université d'Auckland, Nouvelle-Zélande), auxquels sont venus depuis s'ajouter de nombreux chercheurs, le logiciel R constitue aujourd'hui un langage de programmation intégré d'analyse statistique.

Le site Internet du "R coredevelopment Team" *http ://www.r-project.org* est la meilleure source d'informations sur le logiciel libre R. Vous pourrez y trouver les différentes distributions du logiciel, de nombreuses bibliothèques de fonctions et des documents d'aide. Des bibliothèques supplémentaires sont aussi disponibles sur le "comprehensive R archive network"(CRAN) *http ://lib.stat.cmu.edu/R/CRAN/*

2 RStudio

RStudio est environnement de développement intégré pour R. L'interface RStudio est composée de différents panneaux, dont l'arrangement peut être reconfiguré, incluant une console, un navigateur de fichiers et graphiques, l'espace de travail et l'historique des commandes.

Parmi ces panneaux, on distingue en particulier :

- la **console** qui est la fenêtre principale où sont exécutées les commandes et où apparaissent les résultats
- l'**espace de travail** (ou éditeur de texte) dans lequel on tapera les commandes. On peut ensuite exécuter simplement les commandes dans la console en sélectionnant une ligne ou un bloc de commandes, puis en cliquant sur "run" ou en tapant CTRL+Entrée. Les commandes sont alors copiées dans la console et exécutées. On sauvera régulièrement le fichier texte afin de pouvoir retrouver les commandes lors des TD suivants (on pourra créer un fichier par TD ou chapitre)

3 Les objets

Les éléments de base du langage R sont des objets qui peuvent être des vecteurs, des matrices, des tableaux de données, des séries chronologiques, des fonctions, des graphiques... Les objets R se

différencient par leur **mode**, qui décrit ce qu'un objet peut contenir, et leur **classe**, qui décrit leur structure.

Les différents modes sont :

- Entier : `integer`
- Logique (ou booléen) TRUE/FALSE : `logical`
- Numérique : `numeric`
- Complexe : `complex`
- Caractère : `character`

Les principales classes d'objets sont :

- Les vecteurs : `vector`
- Les matrices : `matrix`
- Les tableaux : `array`
- Les listes : `list`
- Les tableaux de données : `data.frame`
- Les séries temporelles : `time.series`

On peut avoir des vecteurs, matrices, tableaux, séries temporelles de mode `integer`, `logical`, `numeric`, `complex`, `character`, mais les modes doivent être tous identiques, alors que dans une liste ou un tableau de données les modes peuvent être hétérogènes (par exemple, on peut avoir une colonne de nombres donnant l'âge des individus et une colonne de caractères donnant le sexe des mêmes individus).

3.1 Les vecteurs

Il s'agit de l'objet de base dans R. Un vecteur est une entité unique formée d'une collection ordonnée d'éléments de même mode (c'est à dire `integer` ou `logical`,...). Dans R, les vecteurs peuvent être constitués d'éléments numériques, logiques ou alphanumériques. La constitution manuelle d'un vecteur est réalisée grâce à la fonction `c()`.

Les nombres décimaux doivent être encodés avec un point décimal (et pas une virgule), les chaînes de caractères entourées par des guillemets doubles " ", et les valeurs logiques sont codées par les chaînes de caractères `TRUE` et `FALSE` ou leurs abréviations respectives `T` et `F`. Enfin, les données manquantes sont codées par la chaîne de caractères `NA`.

3.1.1 Création d'un vecteur

Exécuter les séquences suivantes ligne par ligne dans votre console et essayer de comprendre tous les résulats obtenus. Le caractère > est une invite de R signalant que la console est en attente d'une instruction, le caractère + signale que la console attend la suite d'une instruction, le caractère # sert à ajouter des commentaires.

```
c(5,6,1)      # La fonction c permet de concaténer des valeurs
a1<-c(5,6,1)  # "Dans l'objet a1 mettre le vecteur (5,6,1)"
a1            # Affiche l'objet a1 dans la console
a2=c(4.3,-5)  # On peut remplacer <- par =
```

```
a=c(a1,a2)     # c permet aussi de concaténer des vecteurs
a              # affichage dans la console
?c             # Aide sur la fonction c
texte=c("grand","petit") # Création du vecteur caractère "texte"
```

3.1.2 Affichage des résulats

Lorsqu'on veut savoir ce que contient un objet, on peut taper le nom de l'objet dans la console comme dans les exemples ci-dessus. Le contenu de l'objet est alors affiché dans la console. Sous RStudio , on peut aussi regarder dans la fenêtre "Environment" (en haut à droite par défaut) qui décrit toutes les variables présentes dans l'environnement de travail. Pour le moment, votre environnement de travail contient quatre objets, nommés `a`, `a1`, `a2` et `texte` qui sont des vecteurs numériques. Pour éviter les erreurs de programmation lorsqu'on travaille avec R, il faut regarder régulièrement ce que contiennent les objets qui sont créés. Dans la suite, vous devez vous demander à chaque ligne de commande si des nouveaux objets ont été créés ou modifiés (c'est le cas lorsque le signe = apparaît), et, lorsque c'est le cas, ce que ces objets contiennent.

La commande `ls()` permet à tout moment d'obtenir la liste des variables de votre espace de travail.

3.1.3 Suppression d'un objet

On peut effacer un objet créé avec les commandes `remove` ou `rm`.

Executer la commande suivante et vérifier que la variable `a2` a bien disparu de l'environnement de travail.

```
rm(a2)
```

3.1.4 Sélection et modification d'une partie d'un vecteur

La séléction d'une partie d'un vecteur s'opère avec l'opérateur de séléction [ ] et un vecteur de sélection. Le vecteur de séléction peut être un vecteur d'entiers positifs, d'entiers négatifs ou de logiques.

```
a[1]            # Permet d'accéder au premier élément de a
ind = c(2,4)    # Que contient le vecteur ind ?
a[ind]          # Extrait les coordonnées associées au vecteur ind
a[c(2,4)]       # Appel plus compact
d=a[c(1,3,5)]   # Le résultat de l'extraction est stocké dans d
```

On peut également enlever des éléments d'un vecteur à l'aide de la sélection par des entiers négatifs.

```
a[-1]       # Donne a sans son premier élément
a[-(1:3)]   # Donne a sans les trois premiers éléments
a[-c(1,3)] # Donne a sans le premier et le troisième élément
```

La sélection des éléments d'un vecteur permet également de substituer à un sous-ensemble sélectionné des nouvelles valeurs

```
a[1]=0     # Vérifier que la première composante du vecteur a est modifiée
x=3:14     # La commande ":" permet aussi de créer certains vecteurs
x[1:5]=1   # Modification de plusieurs composantes d'un vecteur
```

3.1.5 Opérations sur les vecteurs

Une fois les vecteurs créés, on peut effectuer différentes opérations

```
2*a                  # Multiplication par 2 de chaque coordonnée du vecteur a
e=3/d                # Création du vecteur numérique e de dimension 3 et
                     # de coordonnées 3/5,3,-3/5
f=a-4                # Création du vecteur f de dimension 5 dont les coor-
                     # données sont égales à celles de a moins 4
d=d+e                # Remplacement du vecteur d par le vecteur résultant
                     # de la somme des vecteurs d et e
d=d-e                # Remplacement du vecteur d par le vecteur résultant
                     # de la différence entre les vecteurs d et e
d*e                  # Multiplication terme à terme des vecteurs d et e
e/d                  # Division terme à terme entre les vecteurs e et d
sum(d)               # Calcul de la somme de d
length(d)            # Affichage de la dimension du vecteur d
t(d)                 # Transposition du vecteur d
t(d)%*%e             # Produit matriciel entre le vecteur ligne t(b) et le vec-
                     # teur colonne e
```

3.1.6 Tests

Les tests sont effectués grâce à des opérateurs logiques : `>`, `>=`, `<`, `=<`, `==`, `!=`,... Ils peuvent aussi être effectués par des fonctions de R : `is.numeric()`, `is.character()`,...

```
1==2                 # == réalise un test d'égalité et renvoie un booléen
1==1
1>2
1!= 2                # != signifie "différent de"
1!= 2
(1==1) & (1>2)       # & signifie "ET"
(1==1) | (1>2)       # | signifie "OR"
d==5                 # Test sur les composantes d'un vecteur
a>1 & a<=5
ind=(a>1 & a<=5)     # Le résultat du test est stocké dans ind
is.vector(a)         # La fonction is.vector() renvoie l'expression logique TRUE
                     # si son argument est un vecteur et FALSE sinon
is.numeric(a)
is.character(a)
mode(a)
a*a
```

```
a[1]="test"
mode(a)
a*a                 # On ne peut pas multiplier deux vecteurs de charactères
```

3.1.7 Extraction d'individus

Lorsqu'on manipule des jeux de données en statistique, on est souvent amené à extraire des individus ayant certaines caractéristiques (par exemple tous les cadres de sexe masculin avec un âge compris entre 30 et 35 ans dans une base de données actuarielle). L'utilisation d'une boucle peut sembler naturelle mais ce n'est pas optimal en temps de calcul (ce sera très long si la base de donnée est importante). Il est préférable d'utiliser des tests comme dans les exemples ci-dessous. D'une manière générale, il faut éviter au maximum d'utiliser les boucles quand on travaille avec R.

En utilisant `which`

```
x=-5:5
x<0
ind=which(x<0)  # which permet d'extraire les indices TRUE dans un vecteur de
                # booléen
ind
x[ind]
x[-ind]
```

Sans utiliser `which`

```
x=-5:5
ind=x>0      # ind est un vecteur de booléen
x[ind]       # renvoie les valeurs de x pour lesquelles ind égal TRUE
x[x>=0]
x[x<0]=10    # Remplace les valeurs négatives par 10
```

3.1.8 Valeurs manquantes

Pour différentes raisons, il se peut que certaines données ne soient pas collectées. On parle alors de données manquantes. Elles sont notées "NA" dans R ("Not Avaible"). Ce n'est pas un véritable mode et il possède ses propres règles de calcul.

```
b=NA
b+1
x=1:10
mean(x)
x[10]=NA
x
mean(x)
mean(x,na.rm = TRUE)  # Moyenne sans les valeurs manquantes
is.na(x)              # Permet de savoir où se trouvent les valeurs manquantes
which(is.na(x))
```

3.1.9 Quelques fonctions utiles

On a déjà vu deux méthodes pour construire un vecteur : à l'aide de la fonction `c()` ou bien à l'aide de la fonction ":" Il existe d'autres fonctions comme les fonctions `rep` et `seq` :

```
a=rep(1:4,2)
rep(c(1.4,3,5),each=2)
seq(0,1, length=11)
seq(1.575, 5.125, by=0.05)
unique(a)                    # Permet d'éliminer les doublons
```

D'autres fonctions qui vous seront utiles :

```
b=c(3,4,2,1)
sort(b)          # Permet de trier dans l'ordre croissant (ou décroissant)
order(b)         # Indices des valeurs triées dans l'ordre croissant
order(b)[1]      # Indice du plus petit élément de b
max(b)
letters[1:10]
abs(d)
log(abs(d))
sqrt(d)
```

Exercice 3.1
Soit $x = (7, 9, 13, 8, 4, 2, 16, 1, 6, 19, 15, 12, 19, 14, 8, 2, 19, 11, 18, 7)$. Dans chaque cas, créer un objet y qui contient les éléments suivants :

1. Le deuxième élément du vecteur x.
2. Les cinq premiers éléments du vecteur x.
3. Les éléments strictement supérieurs à 14 du vecteur x.
4. Tous les éléments du vecteur x sauf les éléments en positions 6, 10 et 12.

Exercice 3.2

Le vecteur `letters` de R contient les 26 lettres de l'alphabet. À l'aide de la commande "==", trouver la position de la lettre q dans l'alphabet et créer le vecteur de coordonnées $a1, b2$, et ainsi de suite jusqu'à q??, où ?? désigne la position de q (indice : utiliser la fonction `paste` qui permet de concaténer des objets différents).

Exercice 3.3

À l'aide des fonctions `rep` et `seq` seulement, générer les séquences suivantes (on pourra utiliser la fonction `c` mais pas directement, *ie* on n'écrira pas c(0,6,0,6,0,6) pour la première question par exemple)

1. 0 6 0 6 0 6
2. 1 4 7 10

3. 1 2 3 1 2 3 1 2 3 1 2 3
4. 1 2 2 3 3 3
5. 1 1 1 2 2 3
6. 1 5.5 10
7. 1 1 1 1 2 2 2 2 3 3 3 3

Exercice 3.4

Générer les suites de nombres suivantes à l'aide des fonctions ":" et `rep` seulement, donc sans utiliser la fonction `seq`

1. 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2
2. 1 3 5 7 9 11 13 15 17 19
3. -2 -1 0 1 2 -2 -1 0 1 2
4. -2 -2 -1 -1 0 0 1 1 2 2
5. 10 20 30 40 50 60 70 80 90 100

Exercice 3.5

Étant donné un vecteur d'observations $x = (x_1, x_2, \ldots, x_n)$ et un vecteur de poids correspondants $w = (w_1, w_2, \ldots, w_n)$, on calcule la moyenne pondérée des observations par

$$\sum_{i=1}^{n} \frac{w_i}{w_\Sigma} x_i$$

où $w_\Sigma = \sum_{i=1}^{n} w_i$

Tester l'expression avec les vecteurs de données $x = (7, 13, 3, 8, 12, 12, 20, 11)$ et

$w = (0.15, 0.04, 0.05, 0.06, 0.17, 0.16, 0.11, 0.09)$.

Exercice 3.6

Étant donné un vecteur d'observations $x = (x_1, x_2, \ldots, x_n)$, on calcule sa moyenne harmonique selon la formule suivante

$$\frac{n}{\frac{1}{x_1} + \cdots + \frac{1}{x_n}}$$

Calculer la moyenne harmonique du vecteur $x = (7, 13, 3, 8, 12, 12, 20, 11)$.

3.2 Les matrices

Les matrices, comme les vecteurs, sont de mode quelconque, mais elles ne peuvent pas contenir des éléments de nature différente (c'est à dire qu'une matrice ne peut pas contenir à la fois des valeurs numériques et du texte par exemple).

La syntaxe de base pour créer une matrice est `matrix(vec,nrow=n,ncol=p,byrow=T)` où `vec` est le vecteur contenant les éléments de la matrice, qui seront rangés en colonne (sauf si l'option `byrow=T` est choisie).

3.2.1 Création d'une matrice

```
a=1:20
x1=matrix(a,nrow=5)          # Création de la matrice numérique x1 de dimension
                             # 5*4 ayant pour première colonne 1,2,3,4,5
x1
x2=matrix(a,nrow=5,byrow=T) # Création de la matrice numérique x2 de dimension
                             # 5*4 ayant pour première ligne 1,2,3,4
```

Lorsque la longueur du vecteur est différente du nombre d'éléments de la matrice, R remplit toute la matrice. Si le vecteur est trop grand, il prend les premiers éléments, si le vecteur est trop petit, R le repète.

```
m1=matrix(1:4,nrow=3,ncol=3)
m1
m2=matrix(1,nrow=2,ncol=4)
m2
```

Un vecteur n'est pas considéré par R comme une matrice. On peut transformer un vecteur en une matrice d'une colonne avec la fonction `as.matrix`

```
x=seq(1,10,by=2)
x
as.matrix(x)
```

3.2.2 Quelques opérations sur les matrices

```
x3=t(x2)             # Transposition de la matrice x2
b=x1*x2              # Produit élément par élément
b=x3%*%x2            # Produit matriciel
dim(b)               # Affichage de dimension de la matrice b
b[3,2]               # Sélection de l'élément [3,2] de la matrice b
b[3,2] =5            # Change le coefficient de la matrice
b[,2]                # Sélection de la deuxième colonne de b
b[c(3,4),]           # Sélection des troisième et quatrième lignes de b
b[-2,]               # Suppression de la deuxième ligne de b
b[,-c(2,4)]          # Suppression des deuxième et quatrième colonnes de b
b[,2]                # Sélection de la deuxième colonne de b
b[1,]>600            # Test sur la première ligne de la matrice
ind=which(b[1,]>600)
b[,ind]              # Sélection des colonnes de b telles que la première
                     # ligne est supérieure à 600
```

```r
rbind(x1,x2)        # Concaténation verticale des matrices x1 et x2
cbind(x1,x2)        # Concaténation horizontale des matrices x1 et x4
```

La fonction `apply` permet d'appliquer une fonction f aux lignes (`MARGIN`= 1) ou aux colonnes (`MARGIN`= 2) de la matrice

```r
apply(x1,2,sum)  # Sommes par colonne
apply(x1,1,mean) # Moyennes par ligne
apply(x1,1,max)  # Maximum par ligne
```

Exercice 3.7

1. Créer la matrice `mat` suivante

$$\begin{pmatrix} 1 & 5 & 5 & 0 \\ 0 & 5 & 6 & 1 \\ 3 & 0 & 3 & 3 \\ 4 & 4 & 4 & 2 \end{pmatrix}$$

2. Créer une matrice contenant uniquement les deux premières lignes de `mat`.
3. Créer une matrice contenant uniquement les deux dernières colonnes de `mat`.
4. Créer une matrice contenant toutes les colonnes de `mat` sauf la troisième.
5. Que permettent de calculer les commandes `t(mat)`, `det(mat)` et `diag(mat)` ? En déduire la trace de `mat`.
6. Extraire les lignes de la matrice `mat` pour lesquelles la somme des coefficients est supérieure à 10 (en une seule ligne de commande).

Exercice 3.8

1. Que permettent de calculer les commandes 1/`mat`, `mat^(-1)` et `solve(mat)` ?
2. Résoudre le système ci-dessous :

$$\begin{aligned} 23x + 31y &= 1 \\ 34x + 46y &= 2 \end{aligned}$$

3.3 Les tableaux de données

Un tableau de données R (`data.frame`) est un tableau dont les colonnes sont de même longueur mais peuvent être hétérogènes (c'est-à-dire de modes différents). Il peut par exemple contenir des colonnes de caractères et des colonnes de numériques, mais chacune des colonnes contiendra des éléments de même nature.

Les tableaux de données sont très souvent utilisés en statistique puisqu'ils sont consacrés spécifiquement au stockage des données. Chaque colonne décrit une variable quantitative ou qualitative mesurée sur des mêmes individus (chaque ligne du tableau correspondant à un individu).

On peut alors mettre dans un tableau de données des variables qualitatives décrites par des caractères (par exemple une colonne qui décrit le sexe M/F d'un individu ou sa catégorie socio-professionnelle) et des variables quantitatives décrites par des nombres (par exemple le montant des sinistres ou l'âge).

Les principales méthodes pour créer un tableau de données consistent à utiliser les fonctions suivantes

- `data.frame` (permet de regrouper des vecteurs de même taille et de modes différents)
- `read.table` (permet d'importer un tableau de données à partir d'une source externe)
- `as.data.frame` (permet de transformer une matrice en tableau de données)

3.3.1 Création et manipulation d'un tableau de données

```
vec1=c("Sophie", "Paul", "Camille")
vec2=c(42, 34, 19)
vec3=c(TRUE, TRUE, FALSE)
vec4=c("F","H","F")
df=data.frame(Noms = vec1,Age = vec2,Fumeur = vec3,Sexe = vec4)
            # Création d'un tableau de données avec 4 variables et 3 individus
class(df)   # Donne la classe de l'objet df

df$Age      # On accède aux variables avec $ ou []
df[,2]
df$Age[3]   # Permet d'accéder au troisième élément du vecteur df$Noms
df[3,2]     # Donne le même résultat

summary(df) # Remarquer que le résultat de summary est différent pour
            # les variables qualitatives et quantitatives
```

La fonction `summary` permet d'effectuer un résumé d'une variable. Quand il s'agit d'une variable quantitative, le minimum, le maximum, les quartiles et la moyenne sont affichés. Pour une variable qualitative, le nombre d'observations de chaque élément est donné.

```
ma=matrix(1:15,nrow=3)
ma=as.data.frame(ma)        # Transformation de ma en un objet de type data.frame
is.data.frame(ma)           # Vérifie si l'objet est bien de type data.frame
```

3.3.2 Jeu de données natifs de R

De nombreux jeux de données sont disponibles nativement dans R. Pour en avoir une liste complète taper `data()` dans la console. Il faut ensuite taper leur nom dans la console pour le charger.

```
data()              # Ouvre une fenêtre texte listant l'ensemble des tableaux
                    # de données disponibles dans R
women               # Affiche le tableau de données women
```

```
? women              # Descriptif du tableau de données women
names(women)         # Affiche le nom des variables de women
attributes(women)    # Donne les caractéristiques du tableau de données women
women$height
apply(women,1,sum) # La fonction apply peut être utilisée avec des data.frame
apply(women,2,max)
```

Exercice 3.9

On considère dans cet exercice le jeu de données "iris" disponible sous R.

1. Vérifier que c'est un objet de type `data.frame`.
2. Que contient ce jeu de données ?
3. Taper la commande `summary(iris)`. Combien d'individus de l'espèce `versicolor` sont dans la base de données ? Quelle est la moyenne de la variable `Sepal.Width` ?
4. Créer un objet de type `data.frame`, nommé `iris2`, qui contient seulement les individus de l'espèce `setosa`.
5. Trier par ordre décroissant les données de `iris2` en fonction de la longueur du sépale (variable `Sepal.Length`).
6. Retrouver les moyennes des variables `Sepal.Length`, `Sepal.Width`, `Petal.Length` et `Petal.Width` à l'aide des fonctions `apply` et `mean`.

Exercice 3.10

On considère les notes (comprises entre 0 et 20) obtenues par 6 élèves d'une même classe dans 4 matières : mathématiques, physique, français et anglais.

	Maths	Physique	Français	Anglais
Louise	6.0	6.0	5.0	5.5
Jean	11.0	9.5	12.5	10.0
Camille	14.5	14.5	15.5	15.0
Sophie	13.0	12.5	8.5	9.5
Antoine	5.5	7.0	14.0	11.5
Raphaël	14.0	14.0	12.0	12.5

1. Créer un objet z de type `data.frame` qui contient ce jeu de données. On renseignera en particulier le nom des colonnes (avec la commande `names`) et des lignes (avec la commande `row.names`).
2. Calculer la moyenne de chacun des élèves puis ordonner le tableau de données par ordre croissant de cette variable (indice : utiliser la fonction `order`).
3. Calculer la moyenne générale de chaque matière puis ordonner le tableau de données par ordre croissant de cette variable.
4. Créer un `data.frame` qui contient seulement les élèves pour lesquels la note moyenne en sciences est supérieure à la note moyenne dans les matières littéraires (en une seule ligne de commande).

3.4 Les listes

Une liste est un objet hétérogène. C'est une collection ordonnée d'objets qui n'ont pas toujours même mode ou même longueur. Les éléments des listes peuvent donc être n'importe quel objet défini dans R. On peut par exemple créer une liste qui contient un vecteur numérique et une matrice de caractères ! Les listes sont des objets importants car toutes les fonctions de R qui retournent plusieurs objets le font sous forme d'une liste.

3.4.1 Création d'une liste

```
vec=seq(2,10,by=3)
mat=matrix(1:8,ncol=2)
caract=c("M","M","F","M","F","M","M","M")
liste=list(vec,mat,caract)                    # Crée une liste contenant 3 objets
liste
length(liste)
mode(liste)
str(liste)                                    # Donne la structure d'un objet

names(liste)=c("vecteur","matrice","caractère")# Nomme les composantes de la liste
liste

lapply(liste,mean)  # Permet d'appliquer une fonction à chaque élément d'une liste
```

3.4.2 Extraction des éléments d'une liste

On peut accéder à chaque élément de la liste à l'aide de son index entre double crochets [[...]], ou par son nom précédé du signe $.

```
liste[[3]]
liste[[1]]
liste$matrice
liste[["vecteur"]]
```

Exercice 3.11

Créer une liste qui contient votre nom de famille, le prénom de vos parents et votre département de naissance.

4 Éléments de programmation

La programmation sur R est basée sur les mêmes principes que pour d'autres logiciels/langages de calculs scientifiques (*e.g.* Python, Scilab,...). On retrouve des instructions de condition (`if`,`else`) et de répétition (`for`,`while`), mais également des fonctions prédéfinies propres à R

(`apply`,`lapply`,`tapply`,...). On peut également, comme avec d'autres langages de calculs, définir ses propres fonctions lorsqu'elles n'existent pas déjà dans R.

4.1 Les conditions

Il s'agit d'exécuter une instruction si, et seulement si, une condition donnée est vérifiée.

```
x=1
if (x!=0) x+1  # la condition est entre parenthèses, ensuite on écrit l'instruction
if (x) x+1     # formulation équivalente
```

4.2 Les boucles

⚠ **Les boucles sont à utiliser avec parcimonie avec R**, car elles sont généralement inefficaces (coûteuses en temps de calcul). Dans la majeure partie des cas, il est possible de vectoriser les calculs pour éviter les boucles explicites, ou encore de s'enremettre aux fonctions propres à R (`apply`,`lapply`,`tapply`,...) pour réaliser les boucles de manière plus efficace.

```
for (i in 1:99) print(i)             # i varie de 1 à 99
for (i in seq(1,99,by=2))  print(i) # i varie de 1 à 99 avec un pas de 2
```

S'il y a plusieurs instructions à effectuer, le bloc d'instructions est délimité par des accolades {...}

```
for (i in 1:50){
  if (i %% 5) print(paste(i,"n'est pas multiple de 5"))
   # %% est l'opérateur "modulo", il retourne le reste de la division euclienne
  else print(i)
}
```

La boucle `while` permet d'effectuer un bloc d'instructions tant qu'une condition est vraie

```
i=1
while (i<20){
  print(i)
  i=i+1
}
```

Une dernière possibilité est la boucle `repeat` qui permet de répéter indéfiniment des instructions. Pour sortir de cette boucle on utilise la commande `break`

```
x=NULL
i=1
repeat{
  x[i]=i
  if (i>=10) break # on sort de la boucle lorsque i dépasse 10
  i=i+1
}
x
```

Exercice 3.12

- A l'aide d'une boucle `for`, calculer 10!.
- En utilisant les fonctions `prod` ou `factorial` calculer 10! sans utiliser de boucle.

Exercice 3.13

On considère la suite $u = (u_n)_{n\geq 1}$ définie par $u_1 = 2$ et $u_{n+1} = \sqrt{u_n}$ pour $n \geq 1$.

- Écrire une boucle `for` qui calcule les n premiers termes de la suite u (on créera un vecteur `u` contenant les différentes valeurs de la suite à l'aide de la fonction `append`). En déduire la limite de la suite u.
- Écrire une boucle `while` qui calcule les m premiers termes de la suite u où m est le plus petit entier tel que $|u_{m+1} - u_m| < \epsilon$ et ϵ est une valeur fixée. Donner la valeur de m correspondant à $\epsilon = 10^{-4}$.

4.3 Les fonctions

Une fonction permet d'effectuer un certain nombre d'ordres R, ces ordres peuvent dépendre d'arguments fournis en entrée mais cela n'est pas obligatoire. En plus des fonctions présentes de base dans R (`c`, `mean`, `sum`, `log`, `sqrt`,...), il est possible de définir ses propres fonctions.

De manière générale, la définition d'une nouvelle fonction passe par l'expression suivante :

$$\text{fun} = \text{function(arguments)} \ \{\text{expression}\}$$

où "`fun`" est le nom de la fonction, "`arguments`" est la liste des arguments de la fonction (séparés par une virgule) et "`expression`" constitue le bloc d'instructions.

Un premier exemple est le suivant, où l'on définit une fonction calculant la somme des n premiers entiers :

```
som = function(n){  # il y a un seul argument "n" et la fonction s'appelle "som"
  resultat=sum(1:n)
  return(resultat)  # indique l'objet à retourner
}                   # "}" termine le bloc de définition de la fonction

som(3)              # calcul de la somme des 3 premiers entiers
```

Lorsque la fonction doit retourner plusieurs objets, on utilise généralement une liste. On peut aussi donner des valeurs par défaut à certains paramètres d'entrée.

Dans l'exemple suivant on définit une fonction qui simule un tirage aléatoire au loto, l'argument d'entrée est le nombre de tirage que l'on effectue. L'exemple comporte également deux nouvelles fonctions de R : `date` et `sample`. En vous appuyant sur l'aide de R, essayer de comprendre ce qu'elles font.

```
loto=function(a=5) {                          # Par défaut on effectue 5 tirages
  d=date()                                    # Heure et date actuelle
  n=sample(x=1:50,size=a)                     # Tirage aléatoire sans remise de
                                              # a nombres entre 1 et 50
  print(paste('Tirage du',d,':'),quote=FALSE)
  print(n)
  li=list(dat=d,alea=n)
  return(li)
}

res=loto()        # Sans argument R exécute la fonction
                  # avec la valeur par défaut a=5
res$alea          # "$" permet d'extraire un élément de
                  # la liste retournée par la fonction
res$date
loto(8)           # On peut modifier la valeur de l'argument en entrée
```

Enfin, lorsque vous créez une nouvelle fonction n'oubliez pas de la commenter à l'aide de la touche #. Cela permettra à toute personne extérieure de comprendre ce que fait votre fonction.

Exercice 3.14

Sans utiliser l'opérateur $\%*\%$, écrire une fonction `prod.mat` qui effecte le produit matriciel de deux matrices seulement si les dimensions de celles-ci le permettent. Cette fonction aura deux arguments (`mat1` et `mat2`) et devra tout d'abord vérifier si le produit matriciel est possible. Si le produit matriciel est impossible, la fonction devra retourner un message d'erreur.

Exercice 3.15

Ecrire une fonction `puiss.mat` qui calcule la puissance n-ième d'une matrice A en utilisant une boucle `for`. Cette fonction aura comme arguments A et n.

Exercice 3.16

On rappelle qu'un entier naturel n est un nombre premier si, et seulement si, il admet deux diviseurs distincts entiers et positifs (qui sont alors 1 et lui-même).

Par exemple 4 n'est pas un nombre premier (puisqu'il est divisible par 2) mais 5 est un nombre premier (puisqu'il est divisible seulement par 1 et par 5).

Un algorithme simple pour vérifier si un entier n est un nombre premier est donc de tester s'il est divisible par les nombres entiers compris entre 2 et $\sqrt{n}$.

a. Créer une fonction `est.premier` qui prend en argument un entier n et qui renvoie :

- Un message d'erreur si n n'est pas un entier naturel
- Un booléen de valeur `TRUE` si n est un nombre premier
- Un booléen de valeur `FALSE` si n n'est pas un nombre premier

b. En 1640, Fermat a conjecturé que les nombres de la forme $F_n = 2^{2^n} + 1$ pour $n \in \mathbb{N}$ sont des nombres premiers. Vérifier si F_1, F_2, F_3, F_4 et F_5 sont bien des nombres premiers.

Exercice 3.17

Écrire une fonction `phi` servant à calculer la fonction de densité de probabilité d'une loi normale centrée réduite, soit

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$

avec $x \in \mathbb{R}$. La fonction prendra en argument un réel (ou un vecteur) x.

Ensuite regarder l'aide de R concernant la fonction `dnorm` (`?dnorm`) et comparer les résultats obtenus entre cette fonction et la fonction que vous venez de créer.

Exercice 3.18

Un palindrome est un mot pouvant se lire aussi bien de gauche à droite que de droite à gauche. Par exemple le mot `RADAR` est un palindrome.

Créer une fonction `palindrome` qui prend comme argument une chaîne de caractères et qui en sortie indique si le mot est un palindrome. On pourra utiliser les fonctions suivantes :

- `substr` permet d'extraire une partie d'une chaîne de caractères
- `nchar` donne la taille d'une chaîne de caractères

Par exemple l'appel `palindrome("RADAR")` devra retourner `"RADAR est un palindrome"` (on utilisera la fonction `paste`).

Tester votre fonction avec différents mots, par exemple avec “kayak”, “rouge”, “tarte”, “été”,...

Partie 2. Statistique descriptive

1 Introduction et vocabulaire

Lors de toute étude statistique, il est nécessaire de décrire et explorer les données avant d'en tirer de quelconques lois ou modèles prédictifs. La statistique descriptive permet de synthétiser l'information contenue dans une série de valeurs (quantitatives ou qualitatives), observées sur une population.

On appelle **population** l'ensemble des individus sur lesquels porte l'étude statistique.

On appelle variable **quantitative** une variable quantifiable (représentée par un nombre). Par exemple un âge, un poids, une note, une durée,...

Une variable qui n'est pas quantitative est appelée variable **qualitative**. Par exemple le sexe (féminin/masculin), une couleur, un prénom,...

Exercice 1.1

Pour chacun des exemples ci-dessous, dire si les variables sont quantitatives ou qualitatives. On donnera également la population sur laquelle porte l'étude.

- les notes sur 20 d'une classe en mathématiques;
- le PIB (produit intérieur brut) de différents pays dans le monde;
- les marques des voitures vendues en France sur une année;
- la température maximale sur un mois des métropoles françaises;
- le relevé des différentes espèces d'arbres dans une forêt.

---

2 Statistique descriptive univariée

2.1 Variables quantitatives

On distingue deux types de variables quantitatives, les variables quantitatives discrètes et les variables quantitatives continues.

On dit qu'une variable quantitative est **discrète** lorsqu'elle ne peut prendre qu'un nombre limité de valeurs. Par exemple le nombre d'enfants dans une famille, une note entre 0 et 20,...

On dit qu'une variable quantitative est **continue** lorsqu'elle peut prendre, en théorie, une infinité des valeurs. Par exemple les revenus d'un groupe d'individus, une durée entre deux événements,...

2.1.1 Indicateurs statistiques

Les résumés numériques suivants permettent de synthétiser une série de données au moyen d'un nombre restreint de valeurs numériques. On distingue deux types d'indicateurs statistiques : les

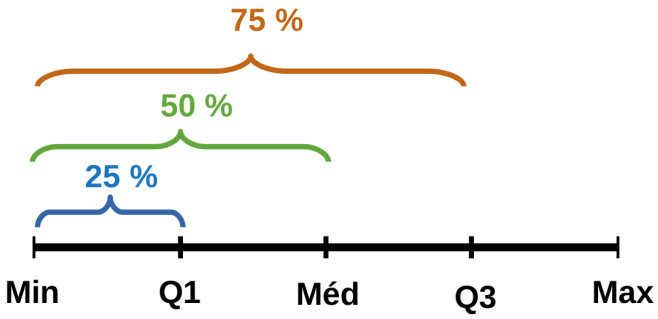
indicateurs de position et les indicateurs de dispersion.

Attention : tous les indicateurs numériques ci-dessous sont *empiriques*, c'est-à-dire calculés à partir d'un échantillon de valeurs, et non pas *théoriques* comme vu dans le cours de probabilités.

a) Indicateurs de position

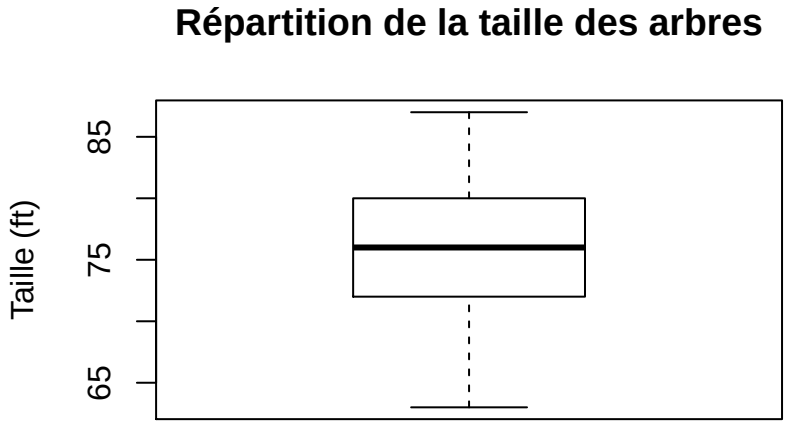
Les indicateurs de position permettent de fournir un ordre de grandeur de la série étudiée. Ces indicateurs sont :

- la **moyenne** : somme de toutes les valeurs de la série divisée par l'effectif total
- la **médiane** : valeur qui partage la série des observations en deux ensembles d'effectifs égaux (50% des observations lui sont inférieur et 50% lui sont supérieur)
- les **quartiles** :
  - 25% des observations sont inférieures au premier quartile
  - 25% des observations sont supérieures au troisième quartile



Le **diagramme en boîte** (ou boîte à moustaches) permet de résumer une série à partir de ses valeurs extrêmes, de ses quartiles et de sa médiane.

```r
help(trees)              # Informations sur le jeu de données "trees" de R

mean(trees$Volume)       # Moyenne de la variable "Volume"
median(trees$Volume)     # Médiane de la variable "Volume"
quantile(trees$Volume)   # Donne les différents quartiles

summary(trees)           # Indicateurs statistiques de la série de données

boxplot(trees$Height)    # Diagramme en boîte de la variable "Height"
```



On vient de voir un premier graphique que l'on peut faire avec R. Plusieurs paramètres graphiques peuvent être ajoutés. De manière générale, on peut spécifier :

- le titre des axes : `xlab`, `ylab`
- la couleur du graphique : `col`
- le titre principal du graphique : `main`
- différents paramètres de taille du texte : `cex.axis` (taille des annotations des axes), `cex.lab` (taille de la légende des axes), `cex.main` (taille du titre),...

Une liste exhaustive des paramètres graphiques s'obtient avec la commande `?par`

```
boxplot(trees$Girth,col="blue")  # Couleur

boxplot(trees$Girth,ylab="Circonférence", main= "Diagramme en boîte")
          # Ajout d'un nom pour l'axe des ordonnées et d'un titre
```

b) Indicateurs de dispersion

Les indicateurs de dispersion permettent de préciser la variabilité de la série étudiée. Ces indicateurs seront d'autant plus grands que les valeurs de la série sont étalées.

Les indicateurs que l'on rencontre le plus souvent sont :

- la **variance** : moyenne des carrés des écarts à la moyenne
- l'**écart-type** : racine carré de la variance

```
sd(trees$Height)  # Ecart-type de la variable "Height"
var(trees$Girth)  # Variance de la variable "Girth"
```

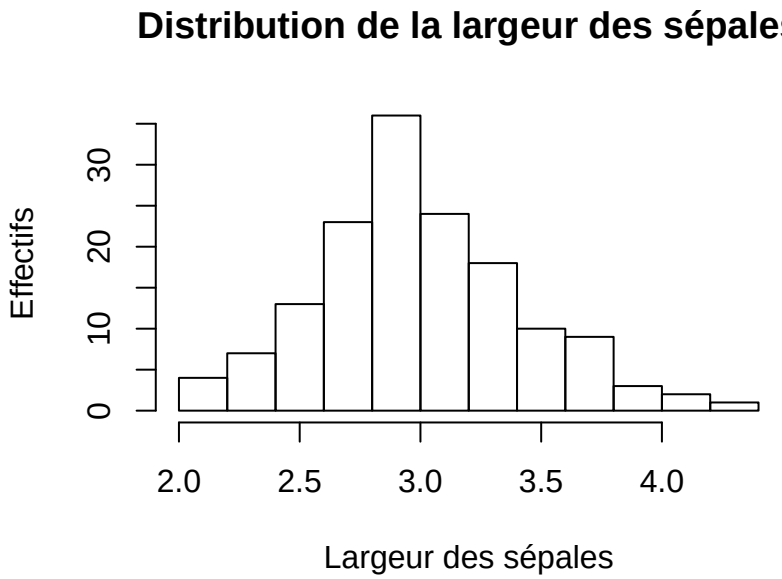
2.1.2 Représentation graphique

Pour visualiser la distribution des valeurs d'une variable quantitative on utilise un **histogramme**.

Tout d'abord, on découpe l'ensemble du domaine des valeurs possibles de la variable étudiée en intervalles adjacents dont on choisit le nombre et les bornes. Cela forme différentes classes. L'histogramme est alors constitué de rectangles accolés ayant pour base chacune des classes et une aire proportionnelle à l'effectif de la classe correspondante.

Sur R, on trace un histogramme avec la fonction `hist` :

```r
hist(iris$Sepal.Width) # Histogramme représentant la distribution de la variable
                       # "Sepal.Width" du jeu de données iris
```



Il y a plusieurs paramètres possibles avec la fonction `hist` :

- `freq` : indique si l'on regarde la proportion des classes ou les effectifs
- `breaks` : permet de contrôler la longueur des classes, soit en indiquant le nombre de classes ou bien soit en indiquant les limites des classes

```r
hist(iris$Sepal.Width,freq=F)    # Proportion des classes affichée sur
                                 # l'axe des ordonnées
hist(iris$Sepal.Width,breaks=8) # Découpage en 8 classes des données
hist(iris$Sepal.Width,breaks=c(2,2.7,3.4,4.1,4.5))
   # Découpage en spécifiant les limites de chaque classe
```

2.2 Variables qualitatives

Parmi les variables qualitatives on distingue deux types différents :

- les variables qualitatives **nominales** : couleurs des yeux, profession dans une population de personnes actives,...
- les variables qualitatives **ordinales** : mention au bac, classement à une course,...

Les valeurs que peut prendre une variable qualitative sont appelées **modalités**. Par exemple pour la couleur des yeux, les différentes modalités sont bleus, verts, marrons,...

2.2.1 Indicateurs statistiques

Contrairement aux variables quantitatives, on ne peut pas calculer de caractéristiques numériques avec une variable qualitative. Pour étudier ce type de variable on utilise plutôt un **tableau statistique** (comme pour le cas d'une variable quantitative discrète).

Pour chaque modalité, on appelle **effectif** le nombre d'observations présentant la modalité en question. L'effectif total de l'échantillon est alors égal à la somme des effectifs de chaque modalité.

On appelle **fréquence** de la modalité le rapport entre l'effectif de la modalité et l'effectif total.

Ces informations sont ensuite résumées dans un tableau statistique de la forme suivante (recensement de différents types d'arbres) :

Type d'arbre	Effectif	Fréquence
Eucalyptus	31	0.33
Sapin	24	0.26
Hêtre	28	0.30
Cyprès	10	0.11

Sur R on obtient le tableau des effectifs avec la fonction `table`. Il suffit ensuite de diviser par l'effectif total de la population pour obtenir le tableau des fréquences.

```
table(iris$Species)              # Tableau des effectifs
table(iris$Species)/nrow(iris)   # Tableau des fréquences
```

Pour les variables qualitatives ordinales on peut ajouter deux autres indicateurs : l'**effectif cumulé** et la **fréquence cumulée**. ⚠ Ces indicateurs n'ont de sens que pour les variables ordinales car on peut ordonner les modalités.

Par exemple, on peut recenser les mentions obtenues au Bac :

Mention	Effectif	Fréquence	Effectif cumulé	Fréquence cumulée
Sans mention	230000	0.46	230000	0.46
Assez Bien	140000	0.28	370000	0.74
Bien	85000	0.17	455000	0.91
Très Bien	45000	0.09	500000	1

Sur R les effectifs cumulés et fréquences cumulées s'obtiennent avec la fonction `cumsum` :

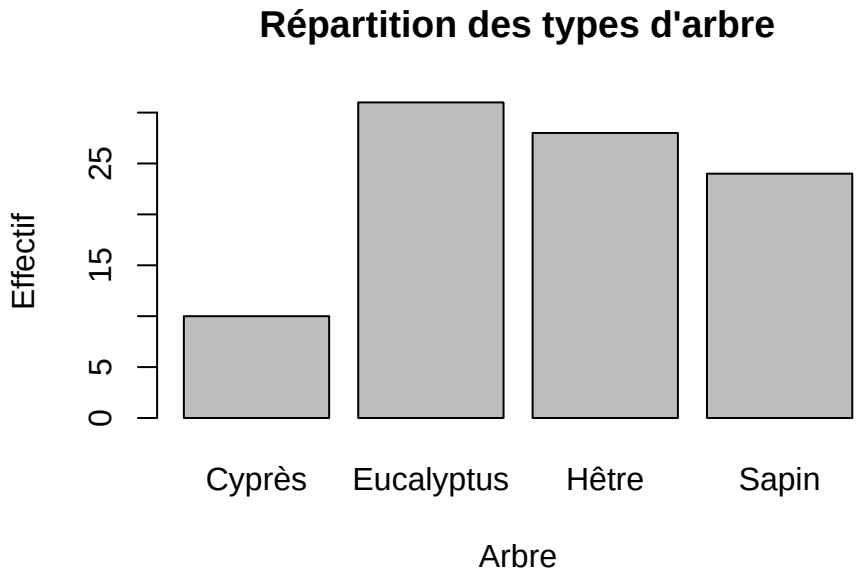
```
cumsum(table(iris$Species))              # Tableau des effectifs cumulés
cumsum(table(iris$Species)/nrow(iris))   # Tableau des fréquences cumulées
```

2.2.2 Représentations graphiques

La représentation graphique usuelle d'une variable qualitative est le **diagramme en barre**. On peut également rencontrer des **diagrammes circulaires**.

Le diagramme en barre permet de donner une vision d'ensemble des observations réalisées. La longueur de chaque barre est proportionnelle aux effectifs ou à la fréquence de chaque modalité.

Ci-dessous est représenté le diagramme en barre des données de rencensement des arbres :



Sur R les deux graphiques précédents s'obtiennent avec les fonctions `barplot` et `pie` :

```
barplot(table(iris$Species))  # Diagramme en barre
pie(table(iris$Species))      # Diagramme circulaire
```

2.3 Cas particulier des variables quantitatives discrètes

Dans le cas des variables quantitatives discrètes, il est assez fréquent de présenter les données à l'aide d'un tableau statistique comme pour les variables qualitatives.

```
?mtcars

table(mtcars$cyl)                       # Tableau des effectifs
cumsum(table(mtcars$cyl))               # Tableau des effectifs cumulés
table(mtcars$cyl)/nrow(mtcars)          # Tableau des fréquences
cumsum(table(mtcars$cyl)/nrow(mtcars)) # Tableau des fréquences cumulées
```

La représentation graphique usuelle dans le cas des variables quantitatives discrètes est alors le diagramme en barre :

```
barplot(table(mtcars$cyl))  # Diagramme en barre du nombre de cylindres
```

Exercice 2.1

On considère le jeu de données `airquality` disponible dans R.

1. Charger les données et comprendre d'où elles proviennent.
2. Afficher les noms des variables considérées.
3. Afficher le nombre de lignes et de colonnes du jeu de données.
4. Calculer les paramètres statistiques de base à l'aide de la commande `summary`. Puis retrouver graphiquement ces résultats en traçant des diagrammes en boîte.
5. Calculer séparément la moyenne, la médiane et le premier et troisième quartile des valeurs de la variable `Temp`.

6. Calculer la variance et l'écart-type pour les valeurs de la variable `Temp`.
7. Tracer l'histogramme en fréquence de la variable `Temp` avec 20 classes.
8. Tracer l'histogramme de la variable `Ozone` avec les proportions des classes affichées sur l'axe des ordonnées et avec 18 classes. Ajouter un titre, modifier le noms des axes, et colorer les barres de l'histogramme en gris et les traits de l'histogramme en bleu.
9. Regarder l'aide de la fonction `density`. Appliquer cette fonction à la variable `Ozone`. Cela retourne-t-il une erreur ? Pourquoi ? Corriger la en spécifiant une option.
10. Ajouter la courbe de densité précédemment générée à l'histogramme en utilisant la fonction `lines`.
11. Créer un nouveau jeu de donées correspondant au jeu de données `airquality` privé des lignes de `Ozone` contenant des valeurs manquantes.

Exercice 2.2

Certains jeux de données sont présents dans des **packages** de R. Un package est un ensemble de programmes qui complète les fonctionnalités de R. Il en existe un très grand nombre, chacun amenant des outils statistiques spécifiques.

Dans cet exercice on va utiliser un jeu de données présent dans le package `MASS`.

1. L'installation d'un package se fait en deux étapes : il faut d'abord le télécharger puis ensuite le charger. Installer le package `MASS` à l'aide des commandes suivantes

```
install.packages("MASS") # téléchargement du package
library(MASS)            # chargement du package
```

2. Afficher le jeu de données `Cars93`. Que contient-il ?
3. Combien y a-t-il de variables ? Afficher les noms des variables.
4. De quel type est la variable `Type` ? Quels sont les indicateurs statistiques adaptés à l'étude de ce type de variable ?
5. Calculer les indicateurs statistiques adaptés à la variable `Type`.
6. Proposer et réaliser une représentation graphique adaptée à la variable `Type`.
7. Reprendre les questions 4 à 6 avec la variable `Price` puis avec la variable `Cylinders`. On précisera à chaque fois le type de variable qu'on considère.

---

3 Compléments sur les graphiques avec R

Dans les sections précédentes nous avons vu différentes fonctions permettant de tracer des graphiques avec R (`boxplot`, `hist`, `barplot`,...).

Dans RStudio, les graphiques s'affichent dans la fenêtre **plot** en bas à droite de l'environnement de travail. On peut effctuer différentes actions à l'aide de la barre d'outils de la fenêtre **plot** :

- naviguer entre les différents graphiques tracés à l'aide des flèches
- afficher dans une autre fenêtre et zoomer en cliquant sur **Zoom**
- enregistrer son graphique en format *pdf* ou image (*png*, *jpeg*,...) à l'aide du bouton **Export**

La méthode basique pour créer un graphique est d'utiliser la commande **plot** avec laquelle on spécifie le vecteur x des abscisses et le vecteur y des ordonnées des points à tracer.

Différents paramètres peuvent ensuite entre donnés (comme on l'a déjà vu avec la fonction `boxplot`). Pour rappel la commande `?par` permet d'obtenir une liste des paramètres graphiques.

```
x=1:10              # Vecteur des abscisses
y=x*x               # Vecteur des ordonnées
plot(x,y)           # Graphe de la fonction f(x)=x^2 évaluée en {0,1,...,10}
plot(x,y,pch=8)     # Changement du type de points tracés

plot(x,y,col="green",pch=19)  # Type de points et couleur
colors()                      # Liste des couleurs disponibles

plot(x,y,type='l')            # Trace une courbe au lieu de points
plot(x,y,type='l',lty=3)      # Courbe en pointillés
plot(x,y,type='l',lty=3,lwd=4)# Épaisseur de la courbe

plot(x,y,lty=3,col="red",main='Premier graphique avec R',lwd=2, type='b')
```

Les fonctions `points`, `lines` et `abline` permettent d'ajouter des éléments sur un graphique existant :

```
y2=exp(x)
points(x,y2,col='blue',pch=19)  # Ajout de points sur le graphique
lines(x,y2,col='gray')          # Ajout de lignes sur le graphique
abline(v=6,lty=2,col='green')   # Ajout d'une droite verticale sur le graphique
abline(h=80,lty=3)              # Ajout d'une droite horizontale sur le graphique
```

Exercice 3.1

1. Tracer la fonction sinus sur l'intervalle $[0, 10]$.
2. Reprendre le graphique précédent en imposant les limites $[-1.5, 1.5]$ pour l'axe des ordonnées (option `ylim`). On donnera le titre 'Graphique de la fonction sinus' au graphique, le nom 'Angle' à l'axe des abscisses et le nom 'Sinus' à l'axe des ordonnées.
3. Rajouter la droite d'équation $y = \frac{x}{2\pi}$ sur le graphique précédent à l'aide de la fonction `lines`. On tracera cette droite en rouge.
4. Rajouter les points de coordonnées $(0, 1.3)$ et $(2, -1)$ sur le graphique précédent à l'aide de la fonction `points`. On représentera ces points à l'aide du symbole '$*$' colorié en bleu.
5. Taper la commande `plot(sin,0,10)`.

Exercice 3.2 (Extrait d'examen)

1. Taper la commande `?EuStockMarkets`. Que contient le jeu de données EuStockMarkets ?
2. A l'aide de la fonction `boxplot`, visualiser la répartition des différentes variables.
3. Taper les commandes suivantes :

```
t=as.vector(time(EuStockMarkets))
z=as.data.frame(EuStockMarkets)
```

Que contiennent les variables z et t ainsi créées ? A quelles classes appartiennent ces objets ?

4. Tracer l'évolution temporelle des quatre indices (DAX, SMI, CAC, et FTSE) sur un même graphique avec des couleurs différentes.
5. On peut ajouter une légende à un graphique à l'aide de la fonction `legend`. Essayer de légender votre graphique.
6. Calculer le rapport entre le cours du `DAX` et le cours du `CAC` pour chaque jour et réaliser un histogramme de ce rapport.

---

4 Statistique descriptive bivariée

L'analyse statistique univariée se concentrait sur l'étude d'une seule variable. Dans certains cas il peut être intéressant d'étudier simultanément deux variables mesurées sur une même population. Ce croisement d'informations a essentiellement pour objectif de mettre en évidence d'éventuelles *liaisons* entre deux variables.

Dans la suite on note X et Y deux variables étudiées sur une même population. Il existe différents types de liaisons. On parle de :

- liaison causale : si la variable X explique la variable Y
- liaison symétrique : si les deux variables X et Y jouent des rôles symétriques

Comme dans le cas univarié, selon le type de variables que l'on croise, l'étude statistique à mener n'est pas la même. On différencie le croisement de :

- deux variables quantitatives
- une variable quantitative et une variable qualitative
- deux variables qualitatives

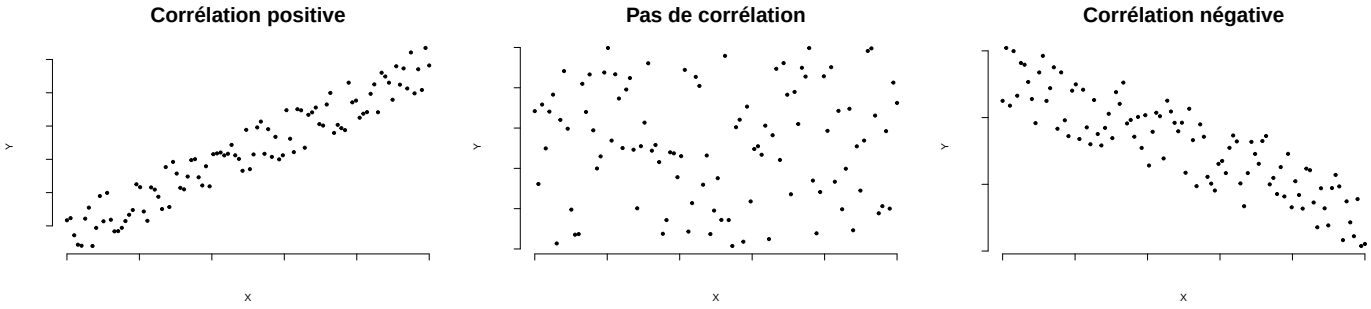
4.1 Deux variables quantitatives

4.1.1 Représentation graphique

Pour représenter graphiquement le croisement de deux variables quantitatives, on réalise un **nuage de points**. Ce type de graphique permet de représenter les observations simultanées de deux variables quantitatives.

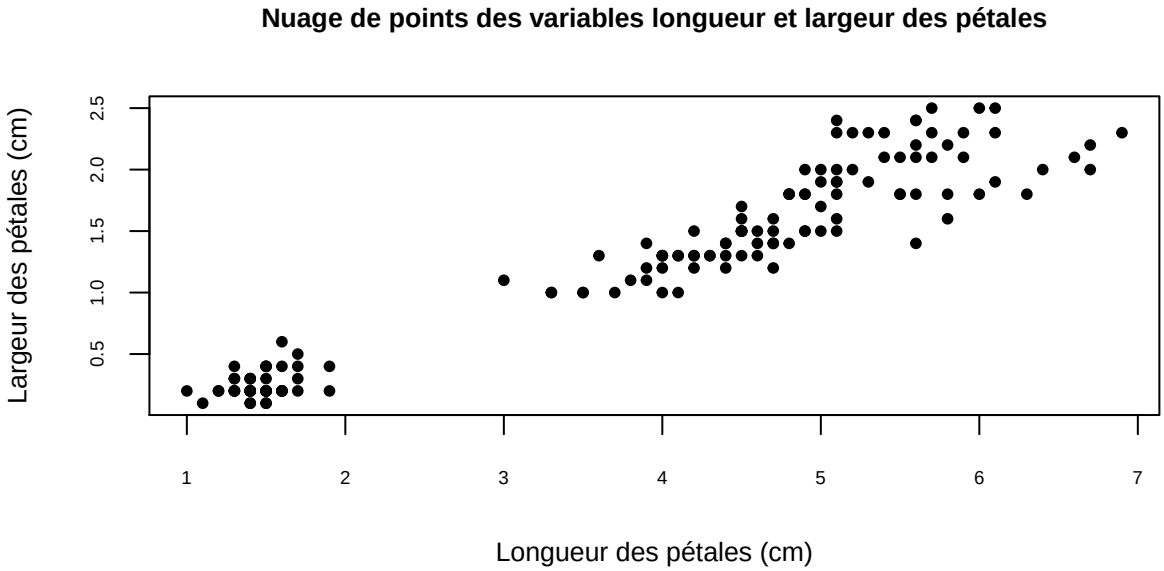
L'axe des abscisses représente la variable X et l'axe des ordonnées la variable Y. On trace ensuite chaque point de coordonées (X_i, Y_i), où X_i (respectivement Y_i) est la valeur de la variable X (respectivement Y) pour le $i^{\text{ième}}$ individu.

Ce nuage de points ainsi tracé donne une assez bonne idée du type de liaison entre les deux variables.



Sur R, un nuage de point entre deux variables quantitatives est obtenu avec la fonction `plot` :

```
plot(iris$Petal.Length, iris$Petal.Width)
```



Sur ce nuage de point on observe une forte liaison linéaire entre la longueur et la largeur des pétales :

- quand la longueur des pétales augmente, la largeur des pétales augmente
- quand la longueur des pétales diminue, la largeur des pétales diminue

On observe également que la relation est *symétrique*, on peut permuter les deux axes :

```
plot(iris$Petal.Width, iris$Petal.Length)
```

4.1.2 Indicateur statistique

Pour rendre compte numériquement la manière dont deux variables varient simultanément, on calcule le **coefficient de corrélation linéaire de Pearson**. Cet indicateur statistique permet de mesurer l'intensité du lien entre deux variables.

Ce coefficient est défini comme le rapport entre la covariance et le produit des écarts-types de chaque variable :

$$r = \frac{Cov(X,Y)}{\sigma_X \sigma_Y}$$

Quelques propriétés du coefficient de corrélation :

- Le coefficient de corrélation est symétrique et prend ses valeurs entre -1 et $+1$.
- Les valeurs -1 et $+1$ correspondent à une liaison linéaire parfaite entre X et Y.
- Lorsque $r \geq 0$ (resp. $r \leq 0$), les variables X et Y varient dans le même sens (resp. sens opposés). On dit que les variables sont positivement (resp. négativement) corrélées.
- Lorsque $r = 0$, il n'y a pas de lien linéaire entre X et Y (on parle alors de variables non corrélées). ⚠ $r = 0$ ne signifie pas nécessairement qu'il n'y a pas de lien entre les deux variables, cela signifie seulement que la relation n'est pas linéaire (elle peut être d'un autre type).

Sur R, le calcul de ce coefficient s'effectue à l'aide de la fonction `cor` :

```
x=seq(1,10,by=0.2)
cor(x,x)          # Corrélation entre x et lui-même
plot(x,x)         # Graphiquement on retrouve la corrélation parfaite

x=-20:20
y=x^2
plot(x,y)
cor(x,y)          # Corrélation entre x et y nulle alors que x et y sont liés
                  # corrélation nulle n'implique pas nécessairement indépendance !

cor(iris$Petal.Width,iris$Petal.Length) # Corrélation largeur/longueur des pétales
```

On retrouve numériquement que la longueur et la largeur des pétales des iris sont très corrélées, ce qu'on pouvait supposer au vu du nuage de points tracé précédement !

Exercice 4.1

On étudie dans cet exercice le jeu de données `USArrests` de R.

1. Que contient ce jeu de données ? Quel est le nombre de variables et d'individus ? Les variables sont-elles qualitatives ou quantitatives ?
2. Donner les statistiques de base de la variable `Murder`.
3. Mener une étude statistique adaptée afin de déterminer s'il existe une éventuelle liaison entre le nombre d'arrestations pour meurtre et le nombre d'arrestations pour agression.

4.2 Une variable quantitative et une variable qualitative

Supposons dans cette partie que X soit une variable qualitative ayant m modalités $(x_1, x_2, ..., x_m)$ et Y une variable quantitative.

Pour chaque modalité x_j de X, on définit une sous-population de la population totale. Cette sous-population est l'ensemble des individus sur lesquels on observe la modalité x_j. On forme ainsi une partition de la population de départ en m sous-populations.

L'étude statistique entre les deux variables X et Y consiste à étudier les différences entre les différentes sous-populations. Si l'on n'observe pas de différence notoire entre les indicateurs

statistiques des différentes populations, il n'y a pas de lien entre les variables X et Y.

4.2.1 Représentation graphique

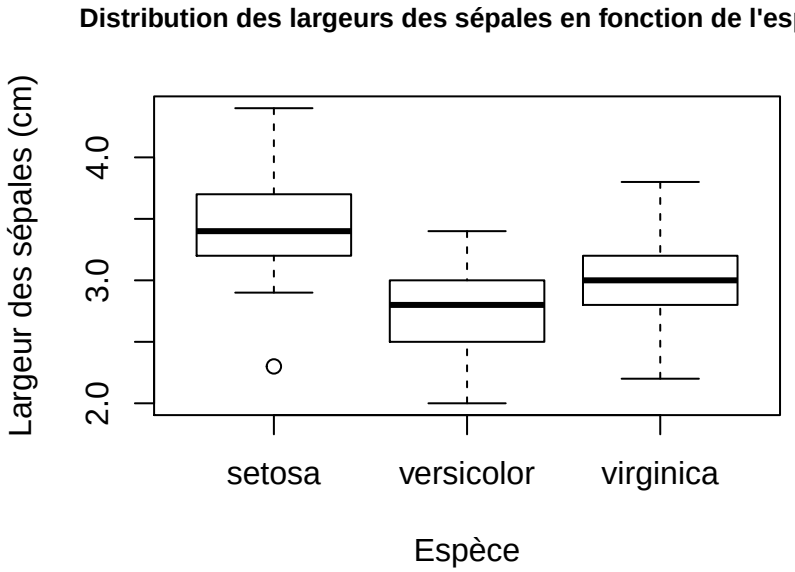
Une représentation graphique usuelle dans l'étude simultannée d'une variable quantitative et d'une variable qualitative est la réalisation de **diagrammes en boîtes parallèles**. Il s'agit de représenter sur un même graphique (ayant une échelle unique) un diagramme en boîte pour Y sur chacune des sous-population définies par X.

La comparaison de ces boîtes donne une idée assez claire de l'influence de X sur les valeurs de Y, c'est-à-dire de la liaison entre X et Y.

Sur R, si on souhaite par exemple visualiser la répartition de la largeur des sépales selon l'espèce d'iris, on utilise la syntaxe suivante :

```
boxplot(iris$Sepal.Width~iris$Species)
```

Cette syntaxe de diagramme en boîte utilise une nouvelle notation "$\sim$". Ce symbole peut se lire comme "en fonction de" : on représente les diagrammes en boîtes de la largeur des sépales *en fonction de* l'espèce.



4.2.2 Indicateurs statistiques

Comme Y est une variable quantitative, on peut calculer ses indicateurs de position et de dispersion en considérant la restriction de la variable Y à chacune des sous-populations.

Si on note n_j l'effectif de la sous-population sur laquelle on observe la modalité x_j, on définit la **moyenne** et la **variance partielle** de Y sur la sous-population j par :

$$\overline{y_j} = \frac{1}{n_j}\sum_{i=1}^{n_j} y_{ij} \ , \ \sigma_j^2 = \frac{1}{n_j}\sum_{i=1}^{n_j}(y_{ij} - \overline{y_j})^2$$

Sur R, on obtient les indicateurs statistiques de chaque sous-population à l'aide de la fonction `tapply`. Cette fonction prend comme arguments une variable quantitative, une variable qualitative puis une fonction de R qu'on applique sur chaque sous-population définie par les modalités de la variable qualitative :

```
tapply(iris$Sepal.Width,iris$Species,FUN=mean)
      # Moyennes partielles de la largeur des sépales sur
      # chaque sous-population
tapply(iris$Sepal.Width,iris$Species,FUN=var)
      # Variances partielles de la largeur des sépales sur
      # chaque sous-population
```

Pour définir une mesure de lien entre une variable quantitative et une variable qualitative, on définit les indicateurs suivants :

- la **variance résiduelle** : moyenne pondérée des variances des sous-populations

$$\sigma_R^2 = \frac{1}{n}\sum_{j=1}^{m} n_j \sigma_j^2$$

- la **variance expliquée** : moyenne pondérée des carrés des variations des sous-populations

$$\sigma_E^2 = \frac{1}{n}\sum_{j=1}^{m} n_j (\overline{y_j} - \overline{y})^2$$

On peut décomposer la variance de Y selon ces deux indicateurs statistiques : $\sigma_Y^2 = \sigma_E^2 + \sigma_R^2$

L'indice de liaison servant à mesurer le lien entre une variable quantitative et une variable qualitative est le **rapport de corrélation**. Il est défini par la racine carré du rapport entre la variance expliquée et la variance totale :

$$s_{Y/X} = \sqrt{\frac{\sigma_E^2}{\sigma_Y^2}}$$

Quelques propriétés du rapport de corrélation :

- X et Y n'étant pas de même type, cet indice de liaison n'est pas symétrique
- le rapport de corrélation est compris entre 0 et 1
- il est d'autant plus grand que le lien entre les variables X et Y est fort :
  - si le rapport est proche de 0, X et Y ne sont pas liées
  - s'il est proche de 1, les deux variables sont liées

Exercice 4.2

L'objectif de cet exercice est de définir une fonction permettant de calculer le rapport de corrélation entre une variable quantitative Y et une variable qualitative X.

1. A l'aide de la fonction `tapply`, quelle commande permettrait de calculer les effectifs de la variable Y sur chaque sous-population formée par la variable X ?
2. Ecrire une fonction `var.exp` qui prend en entrée une variable X qualitative et une variable Y quantitative et qui calcule la variance expliquée de Y par la variable X.

3. Tester votre fonction en calculant la variance expliquée de la variable "`Sepal.Width`" par la variable "`Species`" du jeu de données `iris`.
4. Ecrire une seconde fonction `rapport.cor` qui prend en entrée une variable X qualitative et une variable Y quantitative et qui calcule le rapport de corrélation entre ces deux variables. Cette fonction fera appel à la fonction `var.exp` définie précédemment.
5. Calculer le rapport de corrélation entre les variables "`Sepal.Width`" et "`Species`" du jeu de données `iris`.

Exercice 4.3

On étudie dans cet exercice le jeu de données `InsectSprays` de R.

1. Prendre connaissance de ce jeu de données (Que contient-il ? Combien y a-t-il de variables ? d'individus ?)
2. De quels types sont les différentes variables ?
3. Mener une étude statistique univariée adaptée à chacune des variables du jeu de données. On donnera une représentation graphique ainsi que les valeurs des indicateurs statistiques adéquats.
4. Proposer et réaliser une représentation graphique adaptée à l'étude simultannée des variables `count` et `spray`.
5. Calculer le rapport de corrélation entre les deux variables et interpréter le résultat.

4.3 Deux variables qualitatives

On considère à présent deux variables qualitatives X et Y ayant respectivement l et m modalités :

$$X = (x_1, ..., x_l) \quad \text{et} \quad Y = (y_1, ...y_m)$$

On peut par exemple vouloir croiser l'information entre une catégorie socio-professionnelle et le sexe de différents individus afin d'observer la part de femmes et d'hommes selon le type d'emploi.

Pour présenter ce type de données, on dresse une **table de contingence**. C'est un tableau à double entrée dans lequel on dispose les l modalités de X en lignes et le m modalités de Y en colonnes. L'élément de la $i^{\text{ième}}$ ligne $(1 \leq i \leq l)$ et de la $j^{\text{ième}}$ colonne $(1 \leq j \leq m)$ représente le nombre d'observations ayant les modalités x_i et y_j.

Ci-dessous est donné la table de contingence du croisement des variables `AirBags` et `Type` du jeu de données `Cars93` (*cf* ex.2.2)

Table 1: Exemple de table de contingence

	Compact	Large	Midsize	Small	Sporty	Van
Driver & Passenger	2	4	7	0	3	0
Driver only	9	7	11	5	8	3
None	5	0	4	16	3	6

Parfois, il peut être intéressant de connaître les effectifs de chaque modalités de X et de Y, appelés **effectifs marginaux**. On ajoute ces informations sur la table de contingence en sommant chaque ligne et chaque colonne

Table 2: Exemple de table de contingence avec effectifs

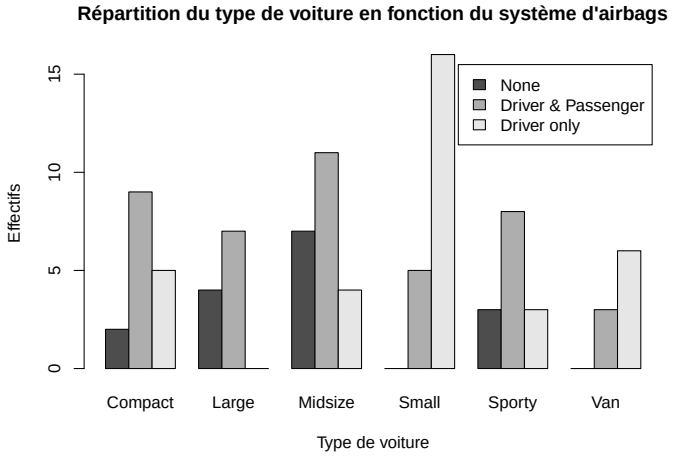
	Compact	Large	Midsize	Small	Sporty	Van	Effectifs
Driver & Passenger	2	4	7	0	3	0	16
Driver only	9	7	11	5	8	3	43
None	5	0	4	16	3	6	34
Effectifs	16	11	22	21	14	9	93

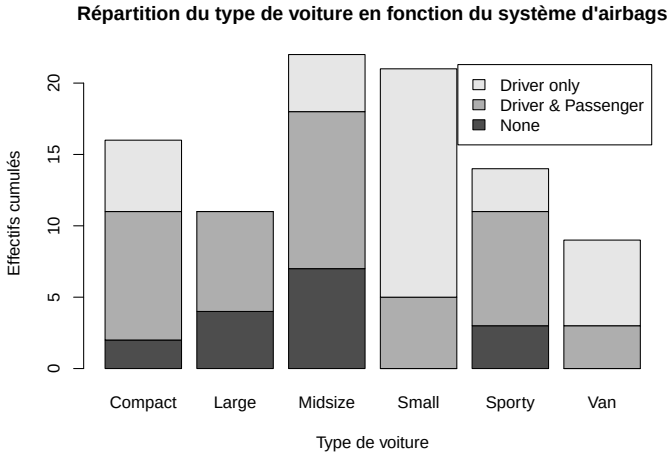
Avec R, on obtient une table de contingence à l'aide de la fonction `table` et les effectifs marginaux avec la fonction `addmargins`

```
Sexe=sample(c("H","F"),30,replace=TRUE)
Lettre=sample(LETTERS[1:5],30,replace=TRUE)
df=data.frame(Sexe,Lettre)
      # On crée un faux tableau de données en effectuant des tirages
      # aléatoires avec remises dans un ensemble de sexe H/F et de lettres

table(df$Sexe,df$Lettre)
      # Table de contingence des variables Sexe et Lettre
addmargins(table(df$Sexe,df$Lettre))
      # Table de contingence avec effectifs marginaux

library(MASS)                          # Chargement du package "MASS"
table(Cars93$AirBags,Cars93$Type) # Table de contingence des variables
                                       # AirBags et Type du jeu de données Cars93
addmargins(table(Cars93$AirBags,Cars93$Type))
```

4.3.1 Représentations graphiques

Dans le cadre de l'étude de deux variables qualitatives, on peut envisager d'utiliser les mêmes représentations graphiques que dans le cas univarié. Par exemple, dans le cas d'un diagramme en barres, on peut soit juxtaposer les colonnes pour chacune des modalités des deux variables ou bien représenter un diagramme en barre des modalités d'une des deux variables et découper chaque colonne selon les effectifs des modalités de la seconde variable.

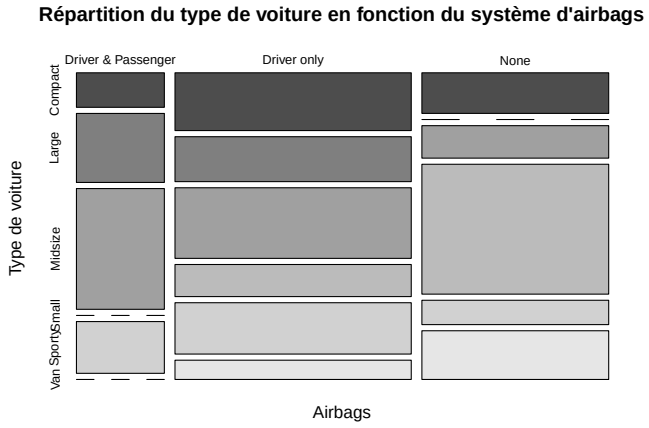


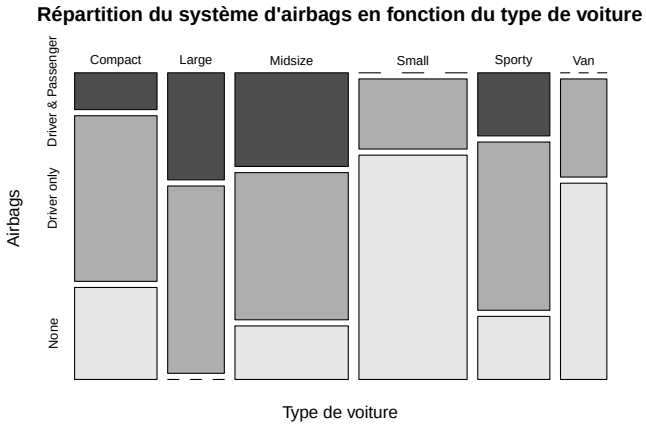


Sur R, on trace ce type de graphique à l'aide de la fonction `barplot` appliquée au tableau de contingence

```
tab=table(df$Sexe,df$Lettre)  # Tableau de contingence
barplot(tab)                  # Diagramme barre des variable Sexe et
                              # Lettre du tableau de données df
barplot(tab,legend.text=unique(df$Sexe))
                                  # Ajout d'une légende
barplot(tab,legend.text=unique(df$Sexe),beside=TRUE)
                                  # Barres juxtaposées

barplot(table(Cars93$AirBags,Cars93$Type),legend.text = unique(Cars93$AirBags))
```

Il est cependant plus fréquent de réaliser un **diagramme en mosaïques**. Ce type de graphique permet de visualiser graphiquement une table de contingence : il est divisé en plusieurs rectangles (ou mosaïques) dont la surface est proportionnelle aux effectifs reportés dans chaque cellule de la table de contingence. Ce type de graphique donne une idée assez précise du lien entre les deux variables qualitatives.





La fonction graphique `mosaicplot` permet de produire ce type de graphique avec R

```
mosaicplot(tab)    # Diagramme en mosaïques des variables Sexe et Lettre
mosaicplot(tab,main="Diagramme en mosaïques") # Ajout d'un titre
mosaicplot(tab,color=TRUE)  # Couleur selon les différentes modalités
mosaicplot(tab,col=1:5)     # On peut choisir les couleurs
```

```r
mosaicplot(t(tab),color=T)  # Représentation de la variable Sexe en
mosaicplot(t(tab),col=c("navy","coral")) # Choix des couleurs

mosaicplot(table(Cars93$Type,Cars93$AirBags),color=TRUE)
```

4.3.2 Indicateur statistique

Pour mesurer le lien entre deux variables qualitatives on calcule le **coefficient du khi-deux**. Cette quantité permet de tester l'indépendance entre deux variables observées sur une population en calculant la distance entre la table de contingence des observations que l'on a et celle attendue si les deux variables étaient indépendantes.

Pour $1 \leq j \leq l$ et $1 \leq j \leq m$, on note :

- $n_{i,j}$ le nombre d'observations simultanées des modalités x_i de X et y_j de Y. $n_{i,j}$ est donc le coefficient de la $i^{\text{ième}}$ ligne et de la $j^{\text{ième}}$ colonne de la table de contingence des variables X et Y
- $n_{i,+}$ l'effectif marginal de la modalité x_i de X, c'est-à-dire la somme des éléments de la $i^{\text{ième}}$ ligne de la table de contingence
- $n_{+,j}$ l'effectif marginal de la modalité y_j de Y, c'est-à-dire la somme des éléments de la $j^{\text{ième}}$ colonne de la table de contingence

Si l'on reprend la table de contingence des données `Cars93` (*cf* Table 2 p.31), on a par exemple :

- $n_{1,2} = 4$
- $n_{3,+} = 34$
- $n_{+,5} = 14$

Le coefficient du khi-deux est défini par $$\chi^2 = \sum_{i,j} \frac{\left(n_{i,j} - \frac{n_{i,+}n_{+,j}}{n}\right)^2}{\frac{n_{i,+}n_{+,j}}{n}}$$

Propriétés du coefficient du chi-deux :

- χ^2 est toujours positif ou nul
- si $\chi^2 = 0$, les deux variables sont indépendantes
- il est d'autant plus grand que la liaison est forte

Avec R le coefficient du khi-deux s'obtient à l'aide de la fonction `chisq.test` qui réalise un test statistique du khi-deux d'indépendance. L'interpétation d'un test statistique fera l'objet d'une autre partie du cours.

```r
tab=table(df$Sexe,df$Lettre) # Table de contingence des variables Sexe et Lettre
res=chisq.test(tab)          # Test du khi-deux d'indépendance
                             # Le message d'erreur est dû aux faibles valeurs
res$statistic                # Valeur du coefficient du khi-deux

res2=chisq.test(table(Cars93$Type,Cars93$AirBags))
res2$statistic
```

Partie 3. Initiation aux probabilités

1 Introduction

En plus d'outils statistiques, R contient de nombreuses fonctions permettant de simuler des variables aléatoires et ainsi de retrouver de manière empirique les notions de probabilités.

Un premier élément de simulation est la génération de nombres aléatoires. Sur R, la fonction `sample` permet de tirer (*ie* générer) des nombres au hasard dans un ensemble donné.

```
x=sample(1:30,size=1)  # nombre tiré au hasard dans {1,...,30}
x

sample(1:30,size=5)    # 5 nombres tirés au hasard dans {1,...,30}

sample(seq(2,5,by=0.1),size=2)
      # on peut spécifier l'ensemble dans lequel le tirage s'effectue

sample(2:5,size=4,replace=TRUE) # tirage avec remise

sample(c(0,1),size=10, replace=TRUE)
      # répétition de 10 épreuves de Bernoulli

x=sample(1:10,size=100,replace=T)
barplot(table(x))       # répartition des 100 tirages avec remise
```

Exercice 1.1

L'argument `prob` de la fonction `sample` permet de spécifier avec quelle probabilité on tire chaque élément. Une urne contient des boules numérotées 4, 5 ou 6. On a une chance sur deux de tirer une boule numérotée 4 et une chance sur cinq de tirer une boule numérotée 5. On tire 100 boules avec remise dans cette urne, on note le résultat à chaque tirage. Simuler le résultat de ce tirage. Quelle est la proportion de boules numérotées 5 ?

Exercice 1.2

On lance n fois succissevement un dé équilibré à 6 faces. Écrire une fonction `lancer.dé` qui prend en argument le nombre n de lancer de dé et qui renvoie les résultats successifs du lancer.

Exercice 1.3

On jette une pièce de monnaie équilibrée. Simuler 10 lancers de cette pièce. On effectuera pour cela un tirage avec remise dans l'ensemble {"pile","face"}.

---

2 Variables aléatoires

Les lois usuelles de probabilités sont présentes dans R. Pour chaque loi il existe quatre fonctions différentes qui s'obtiennent en ajoutant différents préfixes devant le nom de la loi.

Voici un tableau donnant quelques lois usuelles présentes dans R :

Loi	Commande R	Arguments
Binomiale	binom	size, prob
Poisson	pois	lambda
Géométrique	geom	prob
Uniforme	unif	min, max
Exponentielle	exp	rate
Normale	norm	mean, sd

2.1 Fonction de densité

Pour calculer la fonction de densité de probabilité $f(x)$ (resp. la fonction de masse de probabilité $\mathbb{P}(X = x)$) d'une variable aléatoire continue (resp. discrète), on rajoute `d` devant le nom de la loi

```
dbinom(2,10,0.4)     # P(X=2) où X~B(10,0.4)
dnorm(3,5,2)         # f(3) où X~N(5,4) (écart-type et non variance)
dexp(2, c(1, 2, 3)) # f(2) pour X~E(1), X~E(2) et X~E(3)
dpois(c(0,4,9), 1)  # P(X=0), P(X=4) et P(X=9) où X~P(1)

x=dnorm(seq(-5,5,by=0.1),0,1)
plot(seq(-5,5,by=0.1),x,type='l') # densité de la loi N(0,1)
```

2.2 Fonction de répartition

Pour calculer la fonction de répartion $F(x) = \mathbb{P}(X \leq x)$ d'une variable aléatoire, on rajoute `p` devant le nom de la loi

```
pbinom(2,10,0.4)     # P(X<=2) où X~B(10,0.4)
pnorm(3,5,2)         # P(X<=3) où X~N(5,4)
```

On peut obtenir la fonction de survie $\mathbb{P}(X > x) = 1 - F(x)$ avec l'argument `lower.tail`

```
ppois(0, 1,lower.tail=FALSE) # P(X>0) pour X~P(1)
```

Comme pour la fonction de densité, on peut obtenir le graphe de la fonction de répartition en l'évaluant en un nombre convenable de points

```
x=seq(-10,10,by=0.1)
plot(x,pnorm(x,0,2),type="l")    # fonction de répartition de la loi N(0,4)
```

```
y=0:15
plot(y,pbinom(y,15,0.6),type='s') # fonction de répartition de la loi B(15,0.6)
```

2.3 Fonction quantile

Pour calculer la fonction quantile $Q(q) = \inf\{x : F(x) \geq q\}$, on ajoute `q` devant le nom de la loi. Pour rappel, $Q(0.5)$ est la médiane, $Q(0.25)$ le premier quartile, $Q(0.75)$ le troisième quartile,...

```
qbinom(0.25,20,0.2) # premier quartile de la loi B(20,0.2)
qnorm(0.5,5,3)      # médiane de la loi N(5,9)
qexp(p=c(0.05,0.1,0.25,0.5,0.75,0.9,0.95),rate=2)
                    # calcul de plusieurs quantiles de la loi E(2)
```

2.4 Simulation d'une variable aléatoire

Pour simuler des réalisations $x_1, ..., x_n$ d'une loi, on ajoute `r` devant le nom de la loi

```
rnorm(10,0,1)  # Réalisations de 10 variables aléatoires de loi N(0,1)
runif(5,0,1)   # Réalisations de 5 variables aléatoires de loi U([0,1])
rpois(3, 2)    # Réalisations de 3 variables aléatoires de loi P(2)
```

Après avoir simulé un échantillon de taille suffisante, on peut représenter la distribution empirique de l'échantillon à l'aide d'un histogramme. On peut ensuite superposer la densité théorique à partir de laquelle les observations ont été réalisées. Plus la taille de l'échantillon simulé est grande, plus la densité empirique coïncidera avec la densité théorique.

```
X=rnorm(100,0,2) # Réalisations de 100 variables aléatoires de loi N(0,2)
hist(X,freq=F)   # Histogramme de l'échantillon simulé
abs=seq(-7,7,by=0.1)
lines(abs,dnorm(abs,0,2),col='red')

X=runif(1000,0,1) # Réalisations de 1000 variables aléatoires de loi U([0,1])
hist(X,freq=F)
curve(dunif(x,0,1),add=T,col='red') # Trace la courbe d'une fonction donnée
```

Exercice 2.1

Représenter sur un même graphique, la fonction de répartition associée à la loi binomiale de paramètres $n = 50$ et $p = 0.4$ et celle associée à la loi normale de moyenne 20 et d'écart-type 3.

Exercice 2.2

Tracer le graphe de la fonction de répartition d'une variable aléatoire X suivant

- une loi de Bernoulli de paramètre $p = 0.8$
- une loi binomiale de paramètres $n = 15$ et $p = 0.5$

- une loi de Poisson de paramètre $\lambda = 2$.

Exercice 2.3

1. Constuire un échantillon de taille 100 d'une loi binomiale de paramètres $n = 2$ et $p = 0.4$. Quelle est la proportion de 1 dans l'échantillon ?
2. Construire un échantillon de taille 100 d'une loi de Bernoulli de paramètre $p = 0.4$. Quelle est la proportion de 1 dans l'échantillon ?

Exercice 2.4

Soit X une variable aléatoire suivant une loi normale de moyenne 15 et d'écart-type 3.

1. Calculer les probabilités : $\mathbb{P}(16 \leq X \leq 20)$, $\mathbb{P}(X > 18)$, $\mathbb{P}(X < 6)$, $\mathbb{P}(|X - 15| > 5.88)$
2. Représenter le graphe de la fonction de répartition de X sur $[6, 24]$.

Exercice 2.5

Soient $k \in \mathbb{N}$ et $p \in]0, 1[$. La commande `dgeom(k,p)` donne la probabilité qu'une variable aléatoire X suivant une loi géométrique de paramètre p soit égale à k, c'est-à-dire

$$\mathbb{P}(X = k) = (1 - p)^k p$$

1. Ecrire une fonction sur R équivalente à la fonction `dgeom`.
2. On fixe $p = 0.35$. Calculer $\mathbb{P}(X \leq 3)$, $\mathbb{P}(X \geq 4)$ et $\mathbb{P}(X \neq 5)$ à l'aide de la fonction définie précédemment. Comparer vos résultats avec la fonction `dgeom`.

Exercice 2.6

On considère un groupe de n personnes.

1. Estimer la probabilité qu'au moins deux personnes du groupe ont leur anniversaire le même jour
2. Tracer cette probabilité en fonction de $n \in [10, 30]$
3. Estimer la limite lorsque $n \to \infty$
4. Estimer la probabilité qu'exactement deux personnes ont leur anniversaire le même jour
5. Tracer cette probabilité en fonction de $n \in [10, 30]$

Exercice 2.7

Alice et Bob lancent une même pièce de monnaie à tour de rôle. Alice commence. Le premier qui obtient "face" gagne. La probabilité d'obtenir "face" est $p \in]0, 1[$.

1. On limite le nombre de lancers à $N = 3$. Quelle est la probabilité d'avoir un gagnant si $p = \dfrac{1}{2}$?
2. Tracer la probabilité d'avoir un gagnant en fonction de $p \in]0, 1[$. Reprendre pour $N = 5$. Comparer les graphes.

3. Pour $N = 3$ et $p = \frac{1}{2}$, quelle est la probabilité que Alice gagne ? Que Bob gagne ? Que ni l'un ni l'autre ne gagne ?
4. Sachant qu'il y a un gagnant, quelle est la probabilité que Alice gagne ? Tracer cette probabilité en fonction de $p \in]0, 1[$.

Exercice 2.8

Les coefficients de l'équation $ax^2 + bx + c = 0$ sont obtenus en lançant un dé trois fois de suite.

1. Estimer la probabilité que l'équation n'ait pas de racines réelles
2. Estimer la probabilité que les racines soient doubles
3. Estimer la probabilité que $x = 0$ soit une racine
4. Estimer la probabilité que $x = 1$ soit une racine
5. Estimer la probabilité que $x = 1$ soit une racine sachant que $x = 0$ est une racine

---

Exercice 2.9

Soit X une variable aléatoire dont la loi est donnée par

$$\mathbb{P}(X = 0) = 0.2, \quad \mathbb{P}(X = 2) = 0.5, \quad \mathbb{P}(X = 5) = 0.3$$

Simuler 1000 réalisations de X et préciser les effectifs associés aux valeurs de X.

Exercice 2.10

Une urne contient 15 boules dont 5 blanches et 10 noires. On considère les deux expériences suivantes

- E1 : On tire successivement 10 boules de l'urne avec remise
- E2 : On tire successivement 10 boules de l'urne sans remise

On s'intéresse à la variable aléatoire X (resp. Y) égale au nombre de boules blanches tirées l'or de l'expérience E1 (resp. E2).

1. Quelle est la loi de X ? Celle de Y ?
2. Simuler 500 réalisations de chacune de ces deux variables aléatoires.
3. Comparer, suivant le type d'épreuve, le diagramme en barre des fréquences observées avec la distribution des lois correspondantes de X et de Y.

Exercice 2.11

La durée de vie d'un homard peut être modélisée par une loi exponentielle de paramètre $\lambda = 1/20$. Soit X la variable aléatoire égale à la durée de vie en années d'un homard. Sa densité est donnée par

$$f(x) = \frac{1}{20} e^{-x/20} 1_{]0,\infty[}(x), \ x \in \mathbb{R}$$

1. Ecrire une nouvelle fonction R équivalente à la fonction `dexp`. On utilisera la structure `if ... else` pour l'indicatrice.
2. Vérifier numériquement que $\int_0^\infty f(x)dx \approx 1$

3. Séparer l'écran graphique en deux : dans la fenêtre 1, représenter le graphe de la densité f et, dans la fenêtre 2, celui de la fonction de répartition de X.
4. Calculer la probabilité que la durée de vie d'un homard dépasse 3 ans.
5. On suppose qu'un homard vit plus de 2 ans, quelle est la probabilité qu'il vive encore 3 ans ?

---

3 La loi faible des grands nombres

Exercice 3.1

On reprend dans cet exercice le lancer de pièce de l'exercice 1.3 dans lequel on a simulé 10 lancers d'une pièce équilibrée.

1. Reproduire cette simulation en supposant cette fois-ci que la pièce est truquée : la probabilité d'obtenir "face" est de 0.3.
2. Si l'on effectue 10 lancers, combien de "face" obtient-on ? (indice : si l'on note `x` le vecteur contenant les 10 simulations, alors à l'aide d'un test d'égalité et de la fonction `sum`, on peut obtenir le nombre de "face" dans le vecteur `x`)
3. On note E l'expérience suivante "on effectue 10 lancers d'une pièce truquée dont la probabilité d'obtenir"face" est de 0.3. La question 2 est une réalisation de cette expérience. On souhaite réaliser 1000 fois cette expérience :
   - créer un vecteur vide `x.barre`
   - écrire une boucle `for` contenant 1000 itérations où à chaque itération on réalise l'expérience E et on stocke le nombre de "face" obtenus dans le vecteur `x.barre`
   - à la fin des 1000 itérations, le vecteur `x.barre` doit être de longueur 1000
4. Tracer un diagramme en barres qui représente les résultats obtenus (*ie* le nombre de "face") lors de la répétition de cette expérience. On doit observer une plus grande proportion du chiffre 3 : lorsqu'on répète un nombre suffisamment grand de fois l'expérience, le nombre de "face" obtenus lorsqu'on lance 10 fois une pièce truquée est de 3 (nb : $3/10 = 0.3$ qui est la probabilité d'obtenir "face").
5. Au lieu de répéter un grand nombre de fois l'expérience E, on peut augmenter le nombre de lancers que l'on effectue. Reprendre la question 2 en effectuant 1000 lancers au lieu de 10. Remarquer que le nombre de "face" obtenus est proche de 300.

L'exercice ci-dessus est en fait une illustration de la loi des grands nombres (avec les $X_1, ..., X_n$ suivant une loi de Bernoulli de paramètre 0.3).

Enoncé : Soient $X_1, ..., X_n$ une suite de variables aléatoires indépendantes et identiquement distribuées d'espérance μ. On note $\overline{X_n} = \frac{1}{n}\sum_{i=1}^{n} X_i$. Alors $\overline{X_n}$ converge en probabilité vers μ.

On peut illustrer graphiquement cette convergence avec R, ci-dessous on prend l'exemple d'une suite de variables aléatoires *iid* suivant une loi de Poisson de paramètre $\lambda = 5$ puis suivant une loi uniforme

sur $[0, 1]$

```
n=5000
x=rpois(n,5)              # Réalisations de n variables aléatoires de loi P(5)
x_barre=cumsum(x)/(1:n)   # Termes successifs de la moyenne empirique des X
plot(x_barre,type="l")
mu=5
abline(h=mu,col="red")

x=runif(n)                # Réalisations de n variables aléatoires de loi U([0,1])
x_barre=cumsum(x)/(1:n)
plot(x_barre,type="l")
mu=0.5
abline(h=mu,col="red")
```

Exercice 3.2

On veut étudier le comportement asymptotique de $\overline{X_n}$ lorsque $(X_n)_{n\geq 1}$ est une suite de variables aléatoires *iid* suivant une loi de Bernoulli de paramètre $p \in]0, 1[$.

1. Ecrire une fonction `LGN` qui prend comme paramètres n et p et calcule les valeurs successives de $\overline{X_n}$ sous la forme d'un vecteur. Pour cela, on calculera la réalisation de n variables aléatoires de loi de Bernoulli de paramètre p et on utilisera la fonction `cumsum` pour calculer les valeurs successives de $\overline{X_n}$, sans oublier de diviser par n comme dans l'exemple du cours.
2. Appliquer la fonction `LGN` avec $n = 5000$ et $p = 0.5$, mettre le résultat dans un vecteur qu'on notera `x.barre`.
3. Tracer l'évolution du vecteur `x.barre` en fonction de n. On doit observer sur le graphique la convergence de la moyenne empirique vers la moyenne théorique.
4. Pour illustrer la convergence en probabilité, il faut réaliser plusieurs fois l'expérience ci-dessus. Tracer sur un même graphique 10 courbes (de couleurs différentes) représentant l'évolution de $\overline{X_n}$ en fonction de n pour $n = 5000$ et $p = 0.5$. La convergence vers une valeur commune des différentes courbes permet d'illustrer la convergence en probabilité de la loi des grands nombres.

---

4 Relations entre les lois usuelles de probabilité

Ci-dessous quelques exemples où l'on approche une loi par une autre loi de probabilité.

Exercice 4.1

La loi binomiale de paramètres (n, p) peut être approchée par la loi de Poisson de paramètre $\lambda = np$. En particulier, cette approximation est bonne pour $n > 20$ et $np \leq 10$.

L'objectif de cet exercice est d'étudier cette approximation avec R.

1. Représenter les graphes des lois suivantes à l'aide de diagrammes en barre, on observera les similitudes et différences entre les différents graphiques (utiliser la commande `par(mfrow = c(3, 2))` afin d'obtenir les graphiques côte à côte) :

- Loi Binomiale de paramètres $(10, 0.5)$
- Loi Binomiale de paramètres $(20, 0.25)$
- Loi Binomiale de paramètres $(50, 0.1)$
- Loi Binomiale de paramètres $(100, 0.05)$
- Loi de Poisson de paramètre 5

2. On souhaite calculer le maximum de l'erreur commise lorsque l'on approche la loi binomiale par la loi de Poisson. Pour cela, on calcule la quantité suivante :

$$E_n = \max_{0 \leq k \leq n} |\mathbb{P}(X = k) - \mathbb{P}(Y = k)|$$

où X est une variable aléatoire de loi binomiale de paramètres $(n, p = \frac{5}{n})$ et Y est une variable aléatoire de loi de Poisson de paramètre $\lambda = 5$. Que vaut E_n pour $n = 5$? Puis pour $n = 10$, $n = 20$, $n = 50$, et enfin $n = 100$?

3. Qu'en déduit-on sur l'approximation d'une loi binomiale par une loi de Poisson ? On remarquera que l'erreur diminue lorsque n augmente.
4. Application : Suite à une campagne de vaccination contre le paludisme, on estime à 2% la proportion de personnes qui seraient atteintes de la maladie. Soit X la variable aléatoire égale au nombre de personnes malades dans un petit village de 100 habitants.

- Déterminer la loi de X
- A l'aide de R, calculer la probabilité de constater au moins une personne malade
- A l'aide de R, calculer la probabilité de constater au plus 10 personnes malades
- Reprendre ces deux calculs en utilisant une approximation de la loi de X par une loi de Poisson
- Quelle est la qualité de l'approximation ?

Exercice 4.2

La loi hypergéométrique de paramètres (l, m, n) avec $n \leq 0.1(l + m)$ peut être approchée par la loi binomiale avec $p = \frac{l}{l+m}$.

1. Calculer les probabilités $\mathbb{P}(X = k)$ pour $k \in \{0, ..., 20\}$ lorsque X est une variable aléatoire suivant :
   - la loi hypergéométrique de paramètres $(1000, 500, 20)$ (fonction `dhyper` de R)
   - la loi binomiale de paramètres $(20, \frac{1000}{1500})$
2. Séparer l'écran en deux avec la commande `par(mfrow = c(2,1))`. Dans la fenêtre 1, représenter le graphe de la densité de la loi hypergéométrique et dans la fenêtre 2, celui de la densité de la loi binomiale sur $\{0, ..., 20\}$. On observera les similitudes entre les différents graphiques.
3. La loi hypergéométrique (resp. binomiale) permet de modéliser un tirage de boules dans une urne sans remise (resp. avec remise). L'approximation ci-dessus permet donc d'assimiler des tirages sans remise à des tirages avec remise à condition que le nombre de boules dans l'urne soit suffisamment grand et que l'on tire peut de boules. Illustrer cette approximation en vous

appuyant sur l'exercice 2.10 en modifiant le nombre de tirages et le nombre de boules que l'on tire.

---

5 Algorithmes de simulation de lois discrètes

La génération de nombres aléatoires uniformément distribués est présente dans la plupart des langages de programmation. Sur `R`, on a vu que l'on pouvait générer des nombres aléatoires avec la fonction `sample` (ou bien avec `runif`).

Cette brique de base peut nous permettre de générer d'autres lois à partir d'algorithmes implémentables sur tous les langages de programmation. Les exercices ci-dessous montrent certains de ces algorithmes.

Exercice 5.1

On montre ici comment on peut simuler une variable aléatoire suivant une loi de Bernoulli à partir d'une variable uniformément distribuée.

1. On considère une variable aléatoire U suivant une loi uniforme sur $[0, 1]$. Séparer la fenêtre graphique en deux. Puis tracer dans la première fenêtre la fonction de répartition et dans la seconde la fonction de densité de U.
2. Montrer (sur une feuille) que la variable aléatoire X suit une loi de Bernoulli de paramètre p où X est définie par :
$$X = \begin{cases} 1 \text{ si } U < p \\ 0 \text{ si } U \geq p \end{cases}$$
3. En déduire un algorithme permettant de simuler une variable aléatoire suivant une loi de Bernoulli. On simulera pour cela une variable uniforme sur $[0, 1]$ à l'aide de la commande `runif`.
4. On rappelle que la somme de n variables indépendantes suivant une même loi de Bernoulli de paramètre p est une variable aléatoire qui suit une loi biomiale de paramètres (n, p). A partir de cette propriété, trouver un algorithme permettant de simuler une variable aléatoire Y suivant une loi binomiale de paramètres $(50, 0.2)$. Indications : simuler n réalisations de la loi uniforme sur $[0, 1]$ et copier la méthode vue en question 3.
5. La question précédente a permis de simuler 1 réalisation d'une loi $\mathcal{B}(50, 0.2)$. Ecrire une boucle permettant de simuler $k = 1000$ réalisations de cette loi. On notera X le vecteur de longueur k contenant chacune des simulations.
6. Tracer l'histogramme de votre simulation X et superposer la fonction de masse théorique de la loi binomiale de paramètres $(50, 0.2)$. (indication : utiliser la fonction `points` pour superposer la fonction de masse de la loi $\mathcal{B}(50, 0.2)$)

Exercice 5.2

On montre ici comment on peut simuler une variable aléatoire suivant une loi Exponentielle à partir d'une variable uniformément distribuée.

1. Rappeler la définition de la fonction de répartition d'une loi exponentielle de paramètre λ.
2. Montrer (sur une feuille) que la variable aléatoire définie par $Z = -\ln(U)$ suit une loi exponentielle, préciser son paramètre. (indication : calculer $\mathbb{P}(Z \leq x)$ pour $x \geq 0$)
3. En déduire un algorithme permettant de simuler une variable aléatoire suivant une loi exponentielle à partir de la simulation d'une variable uniforme sur $[0, 1]$ (`runif`).
4. La question précédente a permis de simuler 1 réalisation d'une loi exponentielle. Ecrire une boucle permettant de simuler $k = 1000$ réalisations de cette loi. On notera X le vecteur de longueur k contenant chacune des simulations.
5. Tracer l'histogramme de votre simulation X et superposer la densité théorique de la loi exponentielle de paramètre correspondant à votre résultat à la question 2. (indication : utiliser la fonction `lines` pour superposer la densité théorique)
