Démarrer en R

Anestis Antoniadis, Bernard Ycart

Le but de ce qui suit est d'aider le débutant en introduisant quelques unes des fonctionnalités de base sur une série d'exemples dont il pourra s'inspirer pour analyser ses propres données. Il est conseillé de lire ce document après avoir lancé R, en exécutant les commandes proposées une par une pour en observer l'effet. Les exemples ont été testés sur les versions 2.14.0 pour Windows et 2.15.0 pour Linux.

Table des matières

5 janvier 2015

1 Cours

1.1 Structures de données

À savoir pour commencer

R est un logiciel statistique, ou plus exactement un langage. C'est un logiciel libre, développé et maintenu par la communauté des participants au sein du projet R et proposé pour les trois systèmes d'exploitation : Unix/Linux, Windows et MacOS. C'est enfin un logiciel modulaire : de nombreux packages complémentaires offrent une grande variété de procédures mathématiques ou statistiques, incluant des méthodes de représentations graphiques complexes, le traitement des séries chronologiques, l'analyse des données, etc. On ne présente ici que quelques éléments de base permettant de comprendre le fonctionnement du langage.

À partir du site `http://www.r-project.org/`, du projet R, atteindre l'un des sites d'archives (CRAN pour Comprehensive R Archive Network), puis télécharger le fichier d'installation de la dernière version de R. Le logiciel s'installe par défaut dans la même langue que Windows, ce qui détermine les intitulés des menus et sous-menus, mais pas toutes les commandes et réponses du système qui resteront en anglais. Les modules complémentaires désirés seront directement installés à partir de R via une liaison internet.

L'application R se présente par une interface utilisateur simple. Elle est structurée autour d'une barre de menu et de diverses fenêtres. Les menus sont peu développés. Leur objectif est de fournir un raccourci vers certaines commandes parmi les plus utilisées, mais leur usage n'est pas exclusif, ces mêmes commandes pouvant être exécutées depuis la console.

Le menu **File** contient les outils nécessaires à la gestion de son espace de travail, tels que la sélection du répertoire par défaut, le chargement de fichiers sources externes, la sauvegarde et le chargement d'historiques de commandes, etc. Il est recommandé d'utiliser un répertoire différent pour chaque projet d'analyse, ce afin d'éviter la confusion dans les données et les programmes au fur et à mesure de l'accumulation des projets successifs.

Le menu **Edit** contient les habituelles commandes de copier-coller, ainsi que la boîte de dialogue autorisant la personnalisation de l'apparence de l'interface, tandis que le menu **Misc** traite de la gestion des objets en mémoire et permet d'arrêter une procédure en cours de traitement.

Le menu **Packages** automatise la gestion et le suivi des librairies de fonctions, permettant leur installation et leur mise à jour de manière transparente au départ du site du CRAN ou de toute autre source locale.

Enfin, les menus **Windows** et **Help** assument des fonctions similaires à celles qu'ils occupent dans les autres applications Windows, à savoir la définition spatiale des fenêtres et l'accès à l'aide en ligne et aux manuels de références de R.

Parmi les fenêtres, on distingue la console, fenêtre principale où sont réalisées par défaut les entrées de commandes et les sorties de résultats en mode texte. À celle-ci peuvent s'ajouter un certain nombre de fenêtres facultatives, telles que les fenêtres graphiques et les fenêtres d'information (historique des commandes, aide, visualisation de fichier, etc.), toutes appelées par des commandes spécifiques via la console.

Lorsque l'on lance R, la console s'ouvre et affiche quelques informations de démarrage puis l'invite de commande indiquée par `>` qui attend l'entrée d'une commande. On tapera les commandes nécessaires pour une tâche particulière après cette invite. Par exemple, tapez `1+1`, puis Entrée. Le résultat est calculé instantanément et affiché sous la forme : `[1] 2.`

Pour démarrer, et pour une utilisation « légère », vous rentrerez ainsi des commandes ligne par ligne. Un « retour-chariot » Entrée exécute la ligne. Dans une ligne de commande, tout ce qui suit `#` est ignoré, ce qui est utile pour les commentaires. Les commandes que nous proposons sur des lignes successives sont supposées être séparées par des retour-chariot. Ajouter un point virgule en fin d'une commande et poursuivre une autre commande sur la même ligne équivaut à exécuter les deux commandes ligne par ligne. Si votre commande est trop longue pour la taille de la fenêtre ou si elle est incomplète un signe `+` indique la continuité de l'invite.
Souvent des commandes doivent être répétées, ou légèrement modifiées. On peut naviguer dans les commandes précédentes avec les touches ↑ et ↓. Une fois rappelée une commande, elle peut être modifiée en se déplaçant dans la ligne : ← ou →.
L'aide en ligne est appelée par la fonction `help`, ou un simple point d'interrogation. La commande `help()` avec pour argument le nom de la fonction sur laquelle on désire une aide, génère une rubrique d'aide avec l'information désirée. Quand on ignore le nom exact d'une fonction mais que l'on pense le connaître en partie, on pourra chercher son nom en utilisant la fonction `help.search()` ou un double point d'interrogation.

```
help(plot)
help.search("table")
??table
?table
?c
```

On ouvre l'aide en ligne de toutes les fonctions installées par `help.start()`, ou bien par l'item **Aide** de la barre des menus de R. Pour quitter, taper `q()` ou utiliser l'item **Quitter** du menu **Fichier**.
Toutes les variables (scalaires, vecteurs, matrices, ...) créées dans R sont appelées des *objets*. Dans R, on affecte des valeurs aux variables en utilisant une flèche `<-` : taper les signes « inférieur » et « moins ». Par exemple on affectera le résultat du calcul 2×3 à la variable x en utilisant la commande `x <- 2*3`. Pour examiner le contenu d'un objet quelconque de R il suffit de taper son nom. Ainsi par exemple pour voir le contenu de x on tapera `x` et on observera après avoir validé la commande avec un retour chariot : `[1] 6`.

Vecteurs et listes

Il existe différents types d'objets dans R, comprenant entre autres des scalaires, des matrices, des tableaux, des « data frame » (tableaux, similaires à des matrices, mais dans lesquels les colonnes ne sont pas nécessairement de même type), des tables et des listes. Dans R, la structure élémentaire d'objet est le vecteur. Une variable scalaire telle que `x` ci-dessus peut être assimilée à un vecteur qui ne possède qu'une seule composante, un vecteur plus général est décrit par plusieurs composantes. Tous les éléments d'un vecteur doivent être du même type (par exemple numérique ou alphanumérique (caractère) ou logique), alors que les listes peuvent être composées d'éléments de types différents. Les nombres décimaux doivent être encodés avec un point décimal, les chaînes de caractères entourées de guillemets doubles `" "`, et les valeurs logiques codées par les valeurs `TRUE` et `FALSE` ou leurs abréviations `T` et `F` respectivement. Les données manquantes sont codées par défaut par la chaîne de caractères `NA`. Pour créer un vecteur on peut utiliser la commande `c()` de concaténation. Par exemple pour créer un vecteur appelé `monvecteur` de composantes 8, 6, 9, 10, et 5, on tape :

```
monvecteur <- c(8,6,9,10,5)
```

Pour examiner le contenu de la variable `monvecteur` il suffit de taper son nom

```
monvecteur
[1]  8  6 9 10 5
```

Le `[1]` est l'indice du premier élément du vecteur (les coordonnées sont numérotées à partir de 1 dans tous les objets : vecteurs, listes, matrices...). L'extraction d'éléments d'un vecteur est rendue possible par l'utilisation d'un vecteur d'indices, placé entre crochets à la suite du nom du vecteur. R possède une grande souplesse concernant cette indexation, ce qui rend les opérations d'extraction et de modification des éléments de vecteurs très aisées. On peut accéder à n'importe quelle composante du vecteur et tapant le nom du vecteur avec l'indice de la composante entre crochets. Par exemple pour obtenir la valeur de la troisième et quatrième composante du vecteur `monvecteur`, on tapera :

```
> monvecteur[c(3,4)]
[1]  9 10
```

Il existe plusieurs autres manières de définir un vecteur dans R. Une façon fréquente est d'utiliser des commandes d'itération : elle permettent de construire des vecteurs de nombres séparés par des pas positifs ou négatifs.

La syntaxe de base, `seq(deb,fin, by=pas)`, retourne un vecteur de valeurs allant de `deb` à `fin` par valeurs séparées par des multiples de `pas`. Les crochets sont facultatifs. Par défaut, `pas` vaut 1. Selon que `fin-deb` est ou non un multiple entier de `pas`, le vecteur se terminera ou non par `fin`.

Si l'on souhaite un vecteur commençant par `deb` et se terminant par `fin` avec `n` coordonnées régulièrement espacées, il est préférable d'utiliser `seq(deb,fin,length=n)`. La

commande `rep` permet de dupliquer les éléments d'un vecteur. Par exemple, `rep(5,6)` duplique le nombre 5 six fois.

```
v <- 0:10
v <- seq(0:10)
v <- seq(0,1,by=0.1)
v <- seq(0,.99,by=0.1)
v <- seq(0,1,by=0.15)
v <- seq(1,0,by=-0.15)
v <- rep(5,6)
```

Vecteurs	
`x:y`	nombres de `x` à `y` par pas de 1
`seq(x,y,by=p)`	nombres de `x` à `y` par pas de `p`
`seq(x,y,length=n)`	`n` nombres entre `x` et `y`
`v[i]`	`i`-ième coordonnée de `v`
`v[i1:i2]`	coordonnées `i1` à `i2` de `v`
`v[-(i1:i2)]`	supprimer les coordonnées `i1` à `i2` de `v`

À côté des vecteurs coexiste un second type d'objets très utilisé dans le langage R, la liste. À l'opposé d'un vecteur, une liste peut contenir des composantes de types différents, par exemple, aussi bien numériques qu'alphanumériques. Une liste peut également avoir pour composante un vecteur. On utilisera la fonction `list()` pour créer une liste dans R. On pourrait par exemple créer la liste `maliste` en tapant :

```
maliste <- list(nom="Fred", epouse="Marie", monvecteur)
```

On peut alors afficher le contenu de la liste `maliste` en tapant son nom :

```
maliste
$nom
[1] "Fred"
$epouse
[1]  "Marie"
[[3]]
[1] 8 6 9 10 5
```

Les composantes d'une liste sont numérotées et on peut y accéder en utilisant des indices. On peut donc extraire le contenu d'une composante d'un élément d'une liste en tapant le nom de la liste avec l'indice de la composante à extraire entre double crochets :

```
maliste[[2]]
[1] "Marie"
maliste [[3]]
[1] 8 6 9 10 5
```

Les composantes d'une liste peuvent être également nommées, et dans ce cas on peut accéder à leur contenu en tapant le nom de la liste suivi d'un `$`, suivi du nom de la composante. Par exemple `maliste$nom` est identique à `maliste[[1]]`. On peut retrouver les noms attribués aux champs d'une liste avec la fonction `attributes()`. Ainsi par exemple :

```
attributes(maliste)
$names
[1] "nom"  "epouse" ""
```

Il est possible de concaténer plusieurs listes en une seule au moyen de la fonction `c(liste1, liste2, ...)`, comme pour les vecteurs.

Les tableaux de données (data.frame) constituent une classe particulière de listes, consacrée spécifiquement au stockage des données destinées à l'analyse. Chaque composant (le plus souvent un vecteur) de la liste forme alors l'équivalent d'une colonne ou variable, tandis que les différents éléments de ces composants correspondent aux lignes ou individus. À cette classe d'objets sont associées une série de méthodes spécifiques dérivées des fonctions de base et destinées à faciliter la visualisation et l'analyse de leur contenu (`plot(), summary()`, ...). Nous verrons plus tard comment réaliser un tableau de données dans R.

Un autre type d'objet que vous rencontrerez avec R est une variable de type table. Ainsi par exemple, si l'on crée un vecteur `mesnoms` contenant les prénoms des élèves d'une classe, on pourra utiliser la fonction `table()` pour construire une table contenant le nombre d'élèves par prénom possible :

```
mesnoms<- c("Marie", "Jean", "Anne", "Jean", "Simon", "Jean", "Simon")
table(mesnoms)
mesnoms
  Marie Jean Anne  Simon
   1     3     1      2
```

Dans les *noms de variables*, les majuscules sont distinctes des minuscules. Toutes les variables d'une session sont globales et conservées en mémoire. Des erreurs proviennent souvent de confusions avec des noms de variables déjà affectés. Il faut penser à ne pas toujours utiliser les mêmes noms ou à libérer les variables par `rm()`. Vos variables sont listées par `ls()`.

```
a<-c(1,2); A<-c("b","c")  # affecte a et A
```

```
ls()                    # vos variables
rm(a)
ls()                    # a disparaît
```

Faites attention à ne pas utiliser des noms de fonctions prédéfinies. Les fonctions de R utilisent généralement des arguments, qui sont des variables d'entrée (des objets) sur lesquelles les calculs programmés dans la fonction sont effectués. Par exemple en utilisant `log10()` sur une entrée numérique `x`, le logarithme de base 10 de x est calculé :

```
x<-100     # affecte 100 à x
log10(x)   # appelle le log10 avec pour entrée le contenu de x
2          # le réesultat
```

Construire des matrices

En R, la structure de base est le vecteur. À partir de cette classe élémentaire sont définies un grand nombre d'autres classes plus spécifiques, chacune caractérisée par des attributs et des méthodes propres. Ainsi les matrices (arrays) sont constituées par un vecteur auquel on adjoint un attribut de dimension.

Il y a plusieurs solutions pour frabriquer des matrices à partir de vecteurs. L'une consiste à écrire ses vecteurs ligne ou colonne, puis à les relier horizontalement ou verticalement par `rbind` ou `cbind`. L'autre consiste à écrire tous les coefficients en un seul vecteur, redimensionné ensuite par la fonction `matrix`. Par défaut `matrix` structure le vecteur des coefficients verticalement colonne par colonne. Si on désire un arrangement ligne par ligne il faut appeler la fonction `matrix()` avec l'argument `byrow=TRUE`.

```
a <- c(1,2,3); b <- c(4,5,6)
rbind(a,b)
cbind(a,b)
c(a,b)
matrix(c(a,b),2,3)
matrix(c(a,b),2,3,byrow=T)
matrix(c(a,b),3,2)
matrix(c(a,b),3,2,byrow=T)
```

La transposée est réalisée par la commande `t`. Elle permet en particulier de changer un vecteur colonne de dimension n en un vecteur ligne (matrice $1 \times n$) et réciproquement.

```
x <- c(1,2,3)
x
t(x)
A <- matrix(c(1,2,3,4,5,6,7,8,9), nrow=3); A
t(A)
```

On a souvent besoin de connaître les dimensions d'une matrice (par exemple pour vérifier si un objet est un vecteur ligne). On utilise pour cela la fonction `dim`. Lorsque la fonction `dim` est appliquée sur un vecteur le résultat est `NULL` : un vecteur n'est pas une matrice, sa longueur est donnée par `length`. Par contre `dim` appliquée à toute matrice de taille $n \times p$ retourne un vecteur dont la première composante est n et la seconde est p. Les fonction `nrow` ou `ncol` désignent le nombre de lignes (rows) ou de colonnes (columns).

La fonction dim	
`dim(A)`	nombre de lignes et de colonnes
`nrow(A)` ou `dim(A)[1]`	nombre de lignes
`ncol(A)` ou `dim(A)[2]`	nombre de colonnes
`length(A)`	nombre total d'éléments

```
help(dim)
A <- matrix(c(1,2,3,4,5,6), nrow=2, byrow=T)
dim(A)
dim(t(A))
dim(A)[1]
dim(A)[2]
length(A)
```

L'élément de la `i`-ième ligne, `j`-ième colonne de la matrice `A` est `A[i,j]`. Si `v` et `w` sont deux vecteurs d'entiers, `A[v,w]` désigne la sous-matrice extraite de `A` en conservant les éléments dont l'indice de ligne est dans `v` et l'indice de colonne dans `w`. `A[v,]` (respectivement : `A[,w]`) désigne la sous-matrice formée des lignes (resp. : des colonnes) indicées par les coordonnées de `v` (resp. : `w`).

```
A <- matrix(c(1,2,3,4,5,6), nrow=2, byrow=T)
A[2,2]
A[,2]
A[,c(1,3)]
A[1,]
A[,2]
```

On peut modifier un élément, une ligne, une colonne, pourvu que le résultat reste une matrice. Pour supprimer des éléments, on donne un signe négatif à leurs indices.

```
A <- matrix(c(1,2,3,4,5,6), nrow=2, byrow=T); A
A[1,3] <- 30; A
A[,c(1,3)] <- c(10,20,30)    # erreur
A[,c(1,3)] <- rbind(c(10,30),c(40,60)); A
A[,-c(1,3)]; A               # supprime les colonnes 1 et 3
```

Faire appel à un élément dont un des indices dépasse le nombre de lignes ou de colonnes de la matrice provoque un message d'erreur (`subscript out of bounds`).

Matrices	
`A[i,j]`	coefficient d'ordre `i,j` de `A`
`A[i1:i2,]`	lignes `i1` à `i2` de `A`
`A[dim(A)[1],]`	dernière ligne de `A`
`A[-(i1:i2),]=`	supprimer les lignes `i1` à `i2` de `A`
`A[,j1:j2]`	colonnes `j1` à `j2` de `A`
`A[,dim(A)[2]]`	dernière colonne de `A`
`A[,-(j1:j2)]]`	supprimer les colonnes `j1` à `j2` de `A`
`diag(A)`	coefficients diagonaux de `A`

Si `A`, `B`, `C`, `D` sont 4 matrices, les commandes `cbind(A,B)`, `rbind(A,B)`, `rbind(cbind(A,B), cbind(C,D))` retourneront des matrices construites par blocs, pourvu que les dimensions coïncident. Si `v` et `w` sont deux vecteurs , `c(v,w)` les concaténera, `cbind(v,w)` les juxtaposera (en colonnes), `rbind(v,w)` les empilera (en lignes).

```
A <- matrix(c(1,2,3,4,5,6), nrow=2, byrow=T)
cbind(A,A)
rbind(A,A)
cbind(A,t(A))                # erreur
rbind(A,c(0,20,30))
```

Des fonctions prédéfinies permettent de construire certaines matrices particulières.

Matrices particulières	
`matrix(x,m,n)`	matrice constante de taille `m,n`
`diag(rep(1,n))`	matrice identité de taille `n`)
`diag(v)`	matrice diagonale dont la diagonale est le vecteur `v`
`diag(A)`	extrait la diagonale de la matrice `A`
`toeplitz`	matrices à diagonales constantes

```
matrix(3,2,3)
d <- 1:6
D <- diag(d)
A <- matrix(c(1,2,3,4,5,6), nrow=2, byrow=T)
d <- diag(A)
help(toeplitz)
M <- toeplitz(1:5)

v <- 1:6
help(matrix)
```

```
A <- matrix(v,2,2)
A <- matrix(v,3,3)
A <- t(matrix(v,2,3))
w <- as.vector(A)
A <- t(A); w <- t(as.vector(A))
```

Un peu d'algèbre permet souvent de construire des matrices particulières à peu de frais, en utilisant le produit matriciel, noté `%*%`, et la transposée `t()`.

```
A <- matrix(rep(1,4),4,1)%*%c(1,5)
A <- (1:4) %*% matrix(rep(1,5),1,5)
B <- matrix(c(1,2,3,4), byrow=T, nrow=2)
B*B
B*diag(rep(1,dim(B)[1]))
diag(rep(1,dim(B)[1]))*B
```

Opérations sur vecteurs et matrices

Les opérations numériques s'effectuent en suivant les ordres de priorité classiques (puissance avant multiplication, et multiplication avant addition). Pour éviter les doutes il est toujours prudent de mettre des parenthèses.

```
2+3*4
(2+3)*4
2^3*4
2^(3*4)
2^3^4
(2^3)^4
7%/%2     # division entière
7%%2      # modulo
```

Les vecteurs peuvent être utilisés tels quels dans des expressions arithmétiques classiques. Les opérations sont alors effectuées élément par élément. Les différents vecteurs intervenant dans l'expression ne doivent pas nécessairement présenter la même longueur. Si ce n'est pas le cas, le résultat est un vecteur possédant une longueur identique à celle du plus long vecteur de l'expression. Les valeurs des vecteurs plus courts sont alors recyclées, éventuellement partiellement, de manière à atteindre cette longueur.

```
x <- c(3,4,5); x
y <- matrix(c(1,2,3,4,5,6), ncol=2); y
z <- c(2,2); z
x*y
y*x
y*z
```

```
y^x
y/x
y-x
x*z # erreur
```

De la même manière, les vecteurs peuvent être utilisés dans des expressions logiques, renvoyant alors un vecteur logique correspondant au résultat de l'expression appliquée sur chaque élément du vecteur de départ.

```
x <- c(1, 2, 3, 4, 5)
x>3
```

Les opérations `+, -,%*%` sont matricielles. Tenter une opération entre matrices de tailles non compatibles retournera en général un message d'erreur.
Quelques fonctions simples s'appliquant sur des objets de type matrices sont résumées dans le tableau suivant :

Fonctions matricielles	
`dim`	dimension d'une matrice
`as.matrix`	transforme son argument en matrice
`%*%`	produit matriciel
`t`	transposition
`det`	déterminant d'une matrice carrée
`solve`	inverse d'une matrice carrée
`eigen`	valeurs propres et vecteurs propres

En voici quelques exemples :

```
a <- c(1,2,3,4,5,6,7,8,9,10)
A <- matrix(a,nrow=5,ncol=2)
B <- matrix(a, nrow = 5, ncol = 2, byrow = TRUE)
C <- matrix(a, nrow = 2, ncol = 5, byrow = TRUE)
t(C)
B%*%C
D <- C%*%B
det(D)
solve(D)
```

Fonctions

Les fonctions forment l'unité de base de la programmation sous R. Elles sont stockées en mémoire sous forme d'objets élémentaires de mode `function` et peuvent contenir une suite d'instructions R, y compris des appels à d'autres fonctions. C'est sur ce

principe pyramidal de fonctions construites sur la base d'autres fonctions que repose une grande part de la puissance et de la concision du langage. La grande majorité des fonctions internes sont construites sur ce mode directement en langage R, la base de cette pyramide étant formée d'un petit nombre de fonctions basiques externes. La syntaxe d'appel d'une fonction est simple et intuitive :
`nom_fonction(argument1 = valeur1, argument2 = valeur2, ...)`.
Les arguments peuvent être spécifiés sous forme d'une liste nommée, la reconnaissance s'effectuant sur base du nom de l'argument, d'une liste ordonnée, basée alors sur la position de l'argument dans la liste (l'emplacement des arguments manquants étant malgré tout réservé au moyen des virgules de séparation), ou d'un mélange des deux.
Exemple :

```
plot(x=hauteur, y=poids, type="p")
```

Cet appel de `plot()`, fonction graphique présentant ici trois arguments x, y et type, est équivalent à

```
plot(hauteur, poids, , ,"p")
```

les arguments `x, y` et `type` intervenant respectivement en première, deuxième et cinquième position dans la définition de la fonction, ou encore à :

```
plot(hauteur, poids, type="p")
```

les deux premiers arguments étant à leur position nominale et l'argument `type` nommé.
En outre la plupart des fonctions sont définies avec une liste d'arguments par défaut, consultable via l'aide associée. Les arguments possédant une valeur par défaut peuvent être omis lors de l'appel de la fonction et sont alors remplacés par cette valeur lors de l'interprétation du code. Cette propriété permet la définition de fonctions présentant un nombre important d'arguments et donc une grande souplesse d'exécution, tout en conservant la facilité d'emploi et la concision du code.
Étant donné le nombre important de fonctions disponibles sous l'environnement R et dans les librairies associées, un effort particulier a été consenti pour leur documentation. Outre une aide concise concernant la syntaxe, les arguments et les principes sous-jacents à l'utilisation de chaque fonction, accessible par la commande `help(nom_fonction)`, l'utilisateur peut bénéficier d'une démonstration par l'exemple de l'emploi des différentes fonctions, grâce à `example(nom_fonction)`.

```
x<-1:10
y<-sin(x)
y<-x*sin(x)
y<-sin(x)/x
A<-matrix(c(4,4,4,4),ncol=2)
sqrt(A)
A^(1/2)
```

Les fonctions vectorielles s'appliquent à l'ensemble d'un vecteur, d'une matrice ou d'un dataframe, et retournent un scalaire. Pour appliquer une telle fonction sur une matrice, colonne par colonne ou ligne par ligne, il faut utiliser la fonction `apply` avec l'argument `MARGIN=1` ou `MARGIN=2`. Il faut se souvenir qu'avec l'option `MARGIN=1` , la fonction retournera un vecteur avec autant de composantes que de lignes, et donc s'appliquera aux lignes. Par exemple, `apply(A,MARGIN=1,sum)` retourne un vecteur qui est formé des sommes des coefficients dans chaque ligne.

Fonctions vectorielles	
`max`	maximum
`min`	minimum
`sort`	tri par ordre croissant
`sum`	somme
`prod`	produit
`cumsum`	sommes cumulées
`cumprod`	produits cumulés
`mean`	moyenne
`median`	médiane
`sd`	écart-type

```
A <- matrix(c(1,2,3,4,5,6), ncol=3, byrow=T)
sum(A)
apply(A,MARGIN=1,sum)
apply(A,MARGIN=2,sum)
apply(A,MARGIN=1,cumsum)
cumsum(A)
mean(A[1,])
apply(A,MARGIN=1,mean)
```

Librairies externes

Le langage R est structuré en librairies de fonctions (packages ou libraries). Les fonctions élémentaires sont regroupées dans la librairie `base`, chargée en mémoire au démarrage. Une sélection d'autres librairies, recommandées par le groupe de coordination du projet R, est également installée par défaut en même temps que le programme principal. Afin de pouvoir utiliser une fonction externe appartenant à une librairie, il est nécessaire de charger cette librairie au préalable dans l'espace de travail courant. Cette opération peut être réalisée via la menu **Packages** ou par la commande `library(nom_librairie)`. Une communauté importante d'utilisateurs a pu contribuer à l'extension des fonctionnalités de R. Un large choix de librairies additionnelles est ainsi disponible sur le site du CRAN, couvrant des domaines très variés de l'analyse statistique au sens large (analyse multivariée, géostatistique, séries chronologiques, modèles non linéaires, ...). Leur installation est facilitée par les commandes du menu **Packages**, de même que leur mise à jour automatique.

Enfin, signalons que des interfaces vers des procédures externes d'autres langages courants, notamment vers le C et le Fortran sont également disponibles.

Types de données

R est un langage faiblement typé. Les variables ne sont pas déclarées, et la même lettre peut désigner un réel, un entier, un polynôme, ou une fonction. Les types existent cependant et l'attribut d'un objet devient important quand on manipule des objets. Tous les objets ont deux attributs : leur type et leur longueur. La fonction `mode()` peut être utilisée pour obtenir le type d'un objet alors que `length` retourne sa longueur. Il y a quatre types principaux de données dans R : *numérique, caractère, logique et complexe*. Par défaut, les nombres sont traités comme des réels en double précision. Les calculs ne sont donc pas « exacts ». La précision machine est de l'ordre de 10^{-16}. Cela ne signifie pas que les nombres inférieurs à 10^{-16} ou supérieurs à 10^{16} sont inaccessibles, mais R ne peut distinguer deux réels que à 10^{-16} près. Les objets `NULL` sont des objets vides sans type particulier et de longueur 0.

```
val <- 605           # numérique
mode(val)
length(val)
cha <- "Hello World" # caractère
mode(cha)
length(cha)
2<4 #logique
mode(2<4)
lcn<-2+3i            # complexe
mode(cn)
length(cn)
mode(sin)            # fonction
names(val)           # NULL
```

Souvent certains éléments dans un ensemble de données sont inconnus ou manquants et on leur attribue une valeur manquante. Le code pour une valeur manquante dans R est `NA`. Ce code indique que la valeur ou la composante de l'objet est inconnue. Toute opération avec un `NA` rend un `NA`. La fonction `is.na()` peut être utilisée pour contrôler l'existence de valeurs manquantes.

```
val <- c(3,6,23,NA)
is.na(val)           # indique les valeurs manquantes
any(is.na(val))      # y a-t-il une valeur manquante ?
na.omit(val)         # supprime les valeurs manquantes
```

Les valeurs non définies et infinies (`NaN, Inf, -Inf`) peuvent être testées à l'aide des fonctions `is.finite, is.infinite, is.nan` et `is.number` de façon similaire à `is.na()`. ces valeurs résultent souvent d'une division par zéro ou de l'application d'un logarithme sur une entrée négative ou nulle.

À part les réels, les entrées d'une matrice peuvent être des booléens, des complexes, ou des chaînes de caractères.

Les booléens sont `TRUE` (true) et `FALSE` (false) pour les saisies ou encore `T` et `F`. L'égalité logique est notée `==` pour la distinguer de l'affectation de paramètres.

```
a <- 2==3
b <- 2<3
c <- T
a|(b&c)
```

Opérateurs logiques

`==`	égalité	`!`	non
`<`	$<$	`>`	$>$
`<=`	$\leqslant$	`>=`	$\geqslant$
`&`	et	`|`	ou

Les matrices booléennes sont le résultat d'opérations logiques matricielles (toujours terme à terme). Par exemple, si `A` et `B` sont deux matrices (réelles) de même taille, `A<B` est la matrice des booléens `A[i,j]<B[i,j]`. Les opérateurs `and` et `or` peuvent s'appliquer à l'ensemble des éléments d'une matrice booléenne, à ses lignes ou à ses colonnes selon l'option choisie. Si `A` est une matrice et `B` est une matrice de booléens, la commande `A[B]` extrait de `A` les coordonnées correspondant aux indices pour lesquels `B` est à vrai.

Les complexes sont définis à l'aide de la constante `1i` ($\sqrt{-1}$).Les fonctions classiques appliquées à un complexe donnent toujours un résultat unique, même si mathématiquement elles ne sont pas définies de façon unique (racine carrée, logarithme, puissances).

```
help(complex)
A<-matrix(c(1,2,3,4),2,2)+1i*matrix(c(0,1,2,3),2,2)
Re(A)
Im(A)
Conj(A)
t(A)
Mod(A)
Arg(A)
A^3
```

Complexes	
`Re`	partie réelle
`Im`	partie imaginaire
`Conj`	conjugué
`Mod`	module
`Arg`	argument (en radians)

Les chaînes de caractères, encadrées par des cotes ou double cotes, (`"..."` ou `'...'`) servent d'intermédiaire pour des échanges de données avec des fichiers.

1.2 Graphiques

R offre une variété de graphiques remarquable. Chaque fonction graphique a un nombre considérable d'options qui rendent la production de graphiques extrêmement flexible. Il est impossible de décrire ici l'ensemble des fonctions graphiques et leurs multiples options. Certaines de ces options peuvent être attribuées par `par` (voir `help(par)`. Les démonstrations obtenues à l'aide de la commande `demo(graphics)` permettent de se faire une petite idée des possibilités graphiques offertes par R. Nous allons dans un premier temps donner quelques détails sur la gestion des fenêtres graphiques, puis énumérer les fonctions qui semblent les plus utiles.

Anatomie d'un tracé

Les commandes de tracé génèrent des figures. Lorsqu'une commande de tracé est tapée sur la console, une fenêtre graphique va s'ouvrir automatiquement avec le graphique demandé. Il est possible d'ouvrir une autre fenêtre en tapant :

```
windows()     # sous windows
quartz()      # sous Mac OS
x11()         # sous linux/unix
```

La fenêtre ainsi ouverte devient la fenêtre active sur laquelle seront affichés les graphes suivants. Pour savoir les fenêtres graphiques qui sont ouvertes :

```
> dev.list()
windows windows
2 3
```

Les chiffres qui s'affichent correspondent aux numéros des fenêtres qui doivent être utilisés si l'on veut changer la fenêtre active :

```
> dev.set(2)
windows
2
```

La commande `dev.off(i)` supprime la fenêtre numéro `i`.

Une figure est composée d'une région dans laquelle s'effectue le tracé, entourée de marges. La taille des marges est contrôlée avec l'argument `mai`. Il s'agit d'un vecteur `c(bottom,left,top,right)` de longueur 4 dont les valeurs des composantes sont les largeurs, en pouces, des marges correspondantes. Un appel typique de `par()` pour fixer les marges est

```
par(mai=c(5,5,8,5)/10)
```

qui permet d'avoir 0.8 pouces en haut et 0.5 pouces pour les autres côtés. Les axes, leurs étiquettes et les titres des figures apparaissent tous dans les marges de la figure.

Chaque marge est composée d'un nombre de *lignes de texte* :

– des lignes spécifiées en 0 correspondent aux bords de la région de tracé.

– des lignes avec des numéros plus élevés sont plus loin dans les marges du tracé.

Le paramètre graphique `mar` définit combien de lignes apparaîtront sur chacune des 4 marges.

Une autre fonction `layout()` partitionne la fenêtre graphique active en plusieurs parties sur lesquelles sont affichés les graphes successifs. Cette fonction est assez compliquée, par exemple, si l'on veut diviser la fenêtre en quatre parties égales :

```
layout(matrix(c(1,2,3,4), 2, 2))
```

où le vecteur indique les numéros des sous-fenêtres, et les deux chiffres 2 que la fenêtre sera divisée en 2 dans le sens de la hauteur, et en 2 dans le sens de la largeur. Pour visualiser la partition créée par `layout()` avant de faire les graphes, on peut utiliser la fonction `layout.show(2)`, si par exemple deux sous-fenêtres ont été définies. Une autre manière de tracer plusieurs figures sur la même fenêtre graphique en les disposant selon un tableau à $n \times m$ figures est donnée par `par(mfrow=c(n,m))`. Par exemple `par(mfrow=c(3,2))` produira une région de tracé sur laquelle seront disposées 6 figures en 3 lignes et 2 colonnes.

Il existe de nombreuses fonctions graphiques dans R. Les énumerer serait trop long, mais l'aide en ligne sur `help(plot)` et `help(par)` donne une description utile des commandes pour l'habillage des graphes (légendes, axes, texte, mise en forme, marges, couleurs, ...) et n'est donc pas répété ici.

Représenter des fonctions

Le plus simple pour commencer est de tracer le graphe d'une fonction de $\mathbb{R}$ dans $\mathbb{R}$ à l'aide de `plot`. On crée pour cela un vecteur `x` d'abscisses, et on prend l'image de ce vecteur par la fonction pour créer un vecteur `y` d'ordonnées. La commande `plot(x,y)` représente les points de coordonnées `(x[i],y[i])` en y traçant des symboles définis par la paramètre `pch` de valeurs possibles allant de 0 à 18 traits noirs (par défaut `pch=1`). Les points sont représentés par `pch=.` ou `pch=o`), ou selon un autre style. La qualité de la représentation dépend bien sûr du nombre de points.

```
x<-seq(0,3*pi, length=10); y<-x*sin(x)
plot(x,y)           # les points sont tracés avec le symbole o
plot(x,y, pch=16)   # les points sont tracés avec le symbole o noirci
plot(x,y, type="l") # les points sont reliés par des segments
x<-seq(0,3*pi, length=100); y=<-x*sin(x)
plot(x,y)
```

Quand on veut superposer plusieurs courbes avec les mêmes échelles de représentation, il est préférable d'utiliser `plot` d'abord avec `type=n` qui autorise le tracé des axes et la bonne échelle et ensuite des appels aux fonctions `lines` ou `points` qui permet le tracé avec des styles différents pour chaque courbe. La syntaxe générale est la suivante.

```
x<-seq(0,3*pi,length=30)
y<-x*sin(x)
y2<-2*y;
plot(x,y2, type="n")
y2<-2*y;
lines(x,y2)        # échelles différentes adaptées
lines(x,y)
points(x,y2,pch=16)
```

On peut également réaliser un graphique composé à l'aide de la libraire `ggplot2`. Ainsi par exemple, les commandes suivantes produisent la figure 1 :

```
library(ggplot2)> df<-as.data.frame(cbind(x,y))
x<-seq(0,3*pi,length=30)
y<-x*sin(x)
df<-as.data.frame(cbind(x,y))
names(df)<-c("x","y")
p<-ggplot(df,aes(x,y))
p+geom_line(aes(y=y),colour="red")+ geom_area(fill="grey")
      +geom_line(aes(y=2*y),colour="blue")
```

Il est fréquent qu'un graphique contienne non seulement une ou plusieurs représentations de fonctions, mais aussi des chaînes de caractères. Ainsi par exemple le code suivant produit les histogrammes de la figure 2 :

```
library(MASS)
data(crabe)      # on charge les données crabe
x <- crabs$FL
y <- crabs$CL
op <- par(mfrow=c(2,1))
 hist(x, col="light blue", xlim=c(0,50))
 hist(y, col="light blue", xlim=c(0,50))
par(op)
```

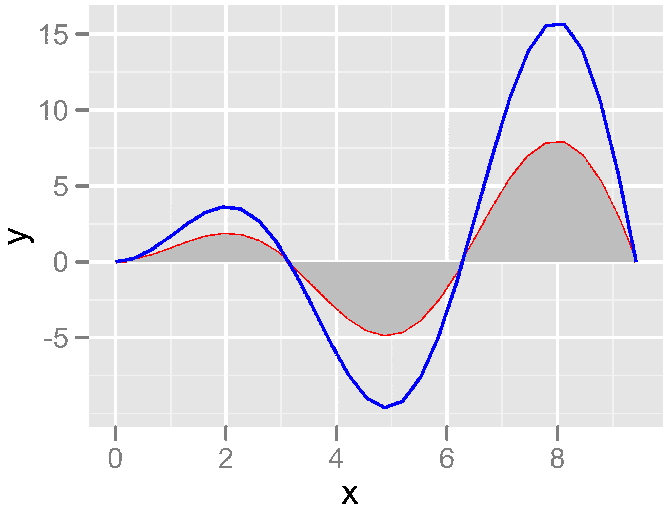

FIGURE 1 – Figure composée.

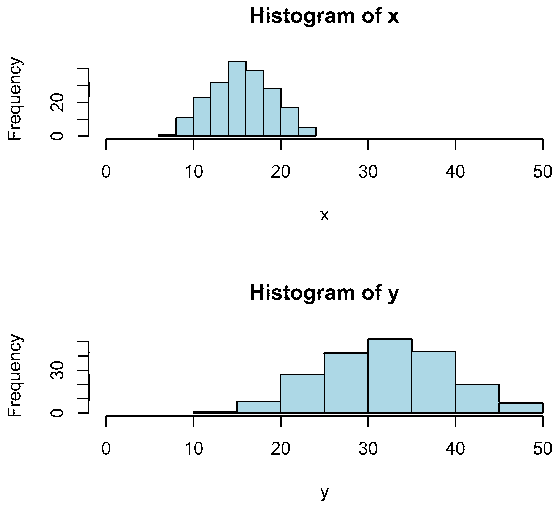

FIGURE 2 – Histogrammes

Des fonctions prédéfinies permettent d'effectuer des représentations planes particulières, comme des projections de surfaces par courbes de niveau ou niveaux de gris, ou des champs de vecteurs. Les exemples qui suivent en sont des illustrations.

```
library(chplot)
data(hdr)
x <- hdr$age
```

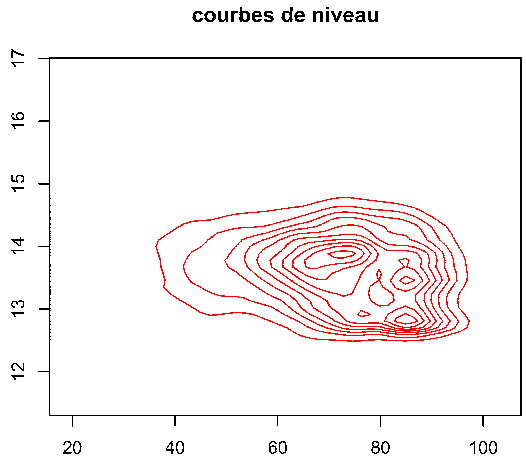

FIGURE 3 – Représentation par courbes de niveau.

```
y <- log(hdr$income)
library(MASS)
z <- kde2d(x,y, n=50)
image(z, main = "niveaux de gris")
{contour(z, col = "red", drawlabels = FALSE,
         main = "courbes de niveau")}
persp(z, main = "tracée de surface en perspective")
```

1.3 Programmation

Types de fichiers

R travaille à partir d'un répertoire de base, qui est donné par la commande `getwd()`. C'est là qu'il va chercher par défaut les fichiers à charger ou à exécuter. On peut le changer par la commande `setwd`. À défaut, il faut saisir le chemin d'accès complet du fichier que l'on souhaite charger ou sauvegarder. Le plus facile est d'utiliser le menu de l'interface Misc ⟶ Changer le répertoire courant

Bien que l'encodage des données directement dans l'environnement R soit possible, dans la majorité des cas les données à analyser proviennent de sources externes sous forme de fichiers. De plus, les objets créés doivent pouvoir être sauvegardés entre les différentes sessions afin de pouvoir reprendre le travail là où on l'avait laissé. C'est pourquoi divers outils d'accès aux fichiers ont été développés sous R. On distingue ainsi les accès aux fichiers propriétaires de R, aux fichiers ASCII ainsi qu'aux fichiers provenant d'autres logiciels d'analyse statistique.

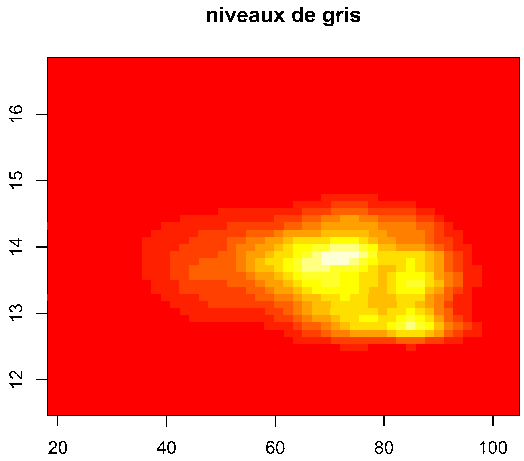

FIGURE 4 – Représentation par niveaux de gris.

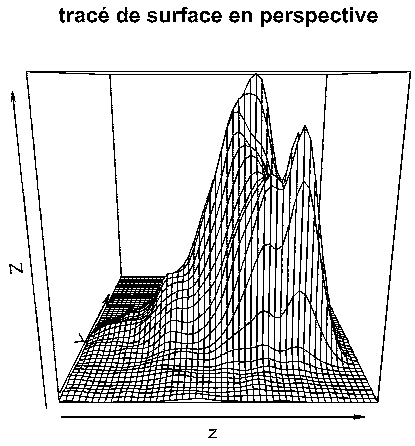

FIGURE 5 – Représentation en perspective

Il convient de distinguer plusieurs sortes de fichiers.

– *Les fichiers de sauvegarde.*
  Ce sont des fichiers binaires, créés par la commande `save` et rappelés par `load`. Ceci permet de reprendre un calcul en conservant les valeurs déjà affectées.
– *Les fichiers de commandes.*
  Ce sont des fichiers texte pour lesquels nous recommandons l'extension `.r`. Ils contiennent des suites d'instructions R, qui sont exécutées en les copiant succes-

sivement dans la console.

– *Les fichiers de fonctions.* Comme les fichiers de commandes, ce sont des fichiers texte, pour lesquels nous recommandons l'extension standard `.r`. Ils contiennent la définition d'une ou plusieurs fonctions. Ces fonctions sont disponibles pour R après les avoir chargées par `source()`.

– *Les fichiers de données (ASCII).*
Le format d'échange le plus courant en ce qui concerne les données brutes reste le fichier ASCII. La lecture de tels fichiers est prise en charge par la commande élémentaire `scan()`. Les arguments de cette fonction permettent de décrire précisément la structure interne du fichier texte afin d'interpréter correctement les caractères lus et de transférer le résultat de cette interprétation dans les objets adéquats. Afin de faciliter la lecture des fichiers de données structurés en colonnes, plusieurs commandes spécifiques ont été développées au départ de la fonction `scan()`. Ces fonctions (`read.table()` et ses dérivés) automatisent la lecture des fichiers de données ASCII standards (CSV, texte délimité, largeur fixe, ...) et stockent leurs résultats sous forme de tableaux de données prêts à l'analyse. Bien qu'elles soient plus spécifiques que `scan()`, ces fonctions conservent toutefois une grande adaptabilité grâce à l'utilisation de nombreux arguments permettant de préciser le format interne du fichier (présence de titres de colonnes, type de séparateur décimal utilisé, symbole(s) de valeur manquante,...). L'exportation de tableaux de données sous forme de fichiers ASCII standards peut être réalisée par la fonction `write.table()`, complémentaire des précédentes, et présentant les mêmes possibilités en matière de formatage des résultats que celles-ci.

– *Logiciels statistiques*
Lorsque les données ont été sauvegardées sous un format propriétaire d'un logiciel statistique tiers inaccessible à l'utilisateur, il est nécessaire de disposer d'outils permettant leur transfert vers le système R afin de pouvoir les analyser. La libraire `foreign` offre ces outils indispensables pour une sélection des logiciels statistiques les plus courants, à savoir Minitab, SAS, SPSS et Stata. Ainsi, la fonction `read.tmp()` importe les fichiers « Minitab Portable Worksheet » sous forme d'une liste, de même que `read.xport()` prend en charge les fichiers SAS Transport (XPORT) et `read.spss()` les données enregistrées au moyen des commandes « save » et « export » de SPSS. Enfin, les fonctions `read.dta()` et `write.dta()` permettent l'importation et l'exportation des fichiers binaires Stata versions 5.0, 6.0 et 7.0.

Définition de nouvelles fonctions

L'utilisation de R consiste en général à étendre le langage par de nouvelles fonctions, définies par des séquences d'instructions. Il est possible de définir des fonctions personnalisées soit directement au départ de la console, soit via un éditeur de texte externe grâce à la commande `fix(nom_fonction)`. La seconde possibilité (conseillée) permet la correction du code en cours d'édition, tandis que la première s'effectue ligne par

ligne, sans retour en arrière possible. Vous pouvez aussi copier-coller le code complet, préparé dans un éditeur de texte, dans la console R.

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

```
nom_fonction <- function(arg1[=expr1], arg2[=expr2], ...){
bloc d'instructions
}
```

Les accolades signalent à l'interpréteur de commande le début et la fin du code source de la fonction ainsi définie, tandis que les crochets ne font pas partie de l'expression mais indiquent le caractère facultatif des valeurs par défaut des arguments. Il est également possible de créer une fonction personnalisée au départ d'une fonction existante, tout en conservant l'original intact, grâce à

```
nom_fonction2 <- edit(nom_fonction1)
```

Lors de l'exécution, R renvoie par défaut le résultat de la dernière expression évaluée dans la fonction. Il est possible de préciser le résultat renvoyé grâce à la fonction `return(objet1[, objet2, ...])`, résultat qui prend la forme d'une liste nommée si plusieurs arguments sont spécifiés. Les arguments sont passés à la fonction par valeur et leur portée ainsi que celle de toute assignation classique à l'intérieur d'une fonction est locale. Lors de l'exécution, une copie des arguments est transmise à la fonction, laissant les originaux intacts. Les valeurs originelles des arguments ne sont donc pas modifiées par les expressions contenues dans la fonction.

```
x <- 2
carre <- function(z) {z<-z*z; return(z)}
carre(x) # retourne [1] 4
x  # retourne 2
```

Compte tenu de la puissance et de la concision du langage, il est fortement recommandé d'ajouter des commentaires au code des fonctions, en les faisant précéder du symbole dièse `#`. La suite de la ligne est alors ignorée lors de l'interprétation de la fonction et peut donc être complétée par un commentaire libre. Il est important de choisir des noms différents pour les nouvelles fonctions, sans quoi les définitions se superposeront, y compris à celles des fonctions prédéfinies. Quand on définit une nouvelle fonction numérique, on a toujours intérêt à faire en sorte qu'elle puisse s'appliquer correctement à une matrice, ce qui impose de veiller aux multiplications terme à terme.

Style de programmation

La philosophie de R est celle d'un langage fonctionnel. Au lieu de créer un logiciel avec programme principal et procédures, on étend le langage par les fonctions dont on a besoin. Le rôle du programme principal est joué par un fichier de commandes appelé *script* contenant essentiellement des appels de fonctions.

Certaines erreurs difficiles à trouver proviennent de confusions entre noms de variables ou de fonctions. R garde en mémoire tous les noms introduits tant qu'ils n'ont pas été libérés par `rm`. Il est donc prudent de donner des noms assez explicites aux variables. Les variables introduites dans la session ou dans les fichiers de commandes sont globales. Par défaut, toutes les variables introduites à l'intérieur d'une fonction sont locales. C'est une des raisons pour lesquelles on a intérêt à définir de nouvelles fonctions plutôt que d'utiliser des fichiers de commande exécutables.

R propose les commandes des langages de programmation classiques.

Commandes principales			
Pour	`for`	`(i in vecteur}){`	`instruction; }`
Tant que	`while`	`(booleen){`	`instruction }`
Si	`if`	`booleen then{`	`instruction }`
Sinon	`else`		`instruction }`

1.4 Exploration de données

Nous allons examiner dans cette section quelques unes des commandes de R, numériques ou graphiques, destinées à aider l'utilisateur à examiner ses données. Ces commandes sont très utiles pour mettre en évidence de manière descriptive certaines particularités des données. Ils permettent d'avoir une première impression d'ensemble sur les données. Les résultats numériques et graphiques obtenus aident beaucoup au choix de modèles et aux hypothèses que l'on sera amené à formuler pour poursuivre une étude statistique. Nous avons déjà remarqué que R est capable de distinguer des données de type différent, en particulier des données de type numérique et de type catégoriel. Les méthodes numériques et graphiques pour décrire un ensemble de données dépendent de leur type.

Résumés numériques

R contient une multitude de fonctions pour calculer des fonctions statistiques de base sur des échantillons de données, qu'elles soient numériques ou catégorielles. Pour des données numériques, le tableau suivant résume les plus importantes :

Résumés numériques	
`mean()`	moyenne aréthmétique
`median()`	médiane
`max`	plus grande valeur
`min`	plus petite valeur
`quantile()`	quantiles de l'échantillon
`var()`	variance
`sd()`	écart-type

Ces fonctions utilisent un ou plusieurs vecteurs numériques pour argument ; de plus, en général, elles peuvent s'appliquer sur toutes les composantes d'un `data.frame` lorsque

ce dernier est numérique et donné comme argument. À titre d'exemple d'utilisation de ces fonctions, considérons l'ensemble des données enregistré comme `data.frame` dans R sous le nom `mtcars`. Un appel d'aide `mtcars` permet d'obtenir un description détaillée des données. En résumé, cette base de données contient les mesures de 11 variables liées à l'aspect et aux performances d'un ensemble de 32 voitures (modèles 1973–1974) :

```
data(mtcars)
attach(mtcars)
mtcars # affiche le contenu suivant
Mazda RX4
Mazda RX4 Wag Datsun 710 Hornet 4 Drive Hornet Sportabout ...
 # load in dataset
 # add mtcars to search path
 mpg cyl  disp  hp drat    wt  qsec vs am gear carb
21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
. . .
```

Ce dataframe contient des variables numériques continues (e.g. `mpg`, `disp`, `wt`)) et discrètes, (e.g. `geai`, `carb`, `cil`)). Pour les numériques, on peut calculer par exemple

```
mean(hp)
var(mgp)
quantile(qsec, probe=c(.2,.8)) # quantiles d'ordre 0.2 et 0.8 de qsec
cor(wt,mpg) # correlation entre poids et consommation
```

Bien évidement, calculer une moyenne ou un écart-type sur un échantillon de valeurs d'une variable catégorielle n'a pas de sens (et même si elle est discrète, sa moyenne peut ne pas être sensée). De telles données sont en général décrites par des tables ou encore par des tracés graphiques que nous examinerons plus tard. Par exemple :

```
> table(cyl)
cyl
4  6  8
11 7 14
```

permet de voir que 11 des 32 véhicules ont 4 cylindres, 7 en ont 6 et 14 en ont 8. On peut transformer ces comptages en fréquences en divisant les entrées du tableau par le nombre total des observations :

```
> table(cyl)/length(cyl)
cyl
      4       6       8
0.34375 0.21875 0.43750
```

Résumés graphiques

Pour représenter une variable discrète ou catégorielle ne prenant qu'un nombre fini de valeurs distinctes on utilise un diagramme en bâtons. Lorsque la variable est catégorielle il est plus adéquat de tracer le diagramme en bâtons pour la table qui lui est associée. On pourra demander l'aide en ligne `?barplot` pour voir comment on peut changer la couleur ou ajouter des titres. Essayez par exemple :

```
layout(c(1,2),2,1)
{barplot(table(cyl), col="red",
         main="Diagramme en batons des comptages")}
{barplot(table(cyl)/length(cyl), col="red",
         main="Diagramme en batons des fréquences")}
```

Pour des données numériques continues on utilise l'histogramme. Les données sont réparties en classes disjointes et le nombre d'éléments (ou la fréquence relative) de chaque classe est graphiquement représentée, comme pour les diagrammes en bâtons pour les variables catégorielles, sauf que les bâtons se touchent. La hauteur de chaque rectangle est égale au nombre d'éléments présent dans la classe alors que lorsque l'on trace l'histogramme des fréquences, la hauteur est proportionnelle à la fréquence relative de sorte que la surface totale de la fonction constante par morceau ainsi tracée soit égale à 1, une propriété qui deviendra familière lorsque l'on étudiera les lois de probabilité usuelles plus tard.

Généralement on choisit un nombre x_0 (usuellement appelé origine de l'histogramme) et une longueur positive b appelée *pas* puis on partitionne l'axe réel en intervalles disjoints :

$$\cdots \cup [x_0 - 2b, x_0 - b[\cup [x_0 - b, x_0[\cup [x_0, x_0 + b) \cup [x_0 + b, x_0 + 2b) \cup \cdots$$

Pour l'histogramme des probabilités, sur chaque intervalle $I_j = [x_0 + jb, x_0 + (j+1)b[$ on trace un rectangle de hauteur $h = \alpha n_j / n$ où n est le nombre total de données dans l'échantillon, n_j le nombre de données dans l'intervalle I_j, le coefficient de proportionnalité $\alpha > 0$ étant choisi de telle sorte que la surface totale de la somme des rectangles soit 1. Il est à noter que le mot origine peut prêter à confusion car il ne signifie pas que l'histogramme démarre à ce point. Il donne plutôt un point de référence à partir duquel les intervalles de la subdivision seront calculés. La fonction permettant de tracer un histogramme dans R est `hist()`. Lisez l'aide par `help(hist)`. L'argument `breaks` spécifie le nombre de classes de l'histogramme. Une valeur trop petite ou trop grande donne une représentation peu informative de la distribution. Par défaut R prend comme

origine x_0 le minimum des données et utilise la règle de Sturges avec pour nombre de classes la partie entière de $\log_2(n) + 1$, n étant la longueur de l'échantillon. Il existe d'autres méthodes pour trouver le pas optimal que nous ne détaillerons pas ici. Afin de voir (figure 6) la sensibilité des histogrammes en fonction de leurs paramètres, regardons de près les données de la base de données `faithful` qui est un jeu de données bimodales célèbre contenant deux variables : les temps d'éruptions (en minutes) et les temps d'attente entre deux éruptions successives du Geyser « Old Faithful ».

```
data(faithful)
attach(faithful) # on charge les données
par(mfrow=c(1,2))
hist(eruptions,breaks=28,freq=F,col=0,xlab="",main="")
mtext("Eruption(min)",side=1,line=2)
mtext("(a) 18 classes",side=3,line=2,cex=.75)
hist(eruptions,breaks=12,freq=F,col=0,xlab="",main="")
mtext("Eruption(min)",side=1,line=2)
mtext("(b) 12 classes",side=3,line=2,cex=.75)
```

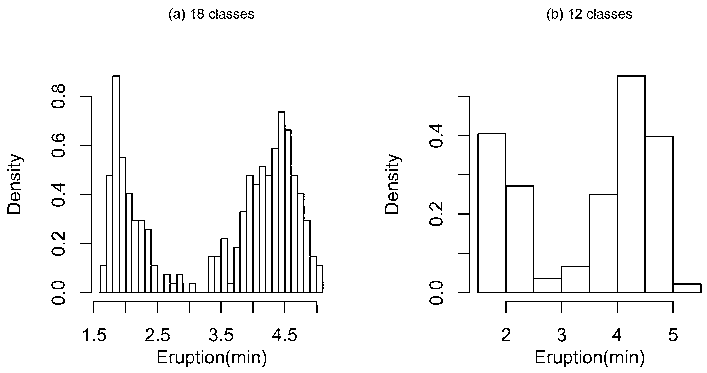

FIGURE 6 – Sensibilité de l'histogramme au nombre de classes

Pour se rendre compte de la sensibilité de l'histogramme au choix de l'origine, nous allons simuler un jeu de données bimodales et tracer les histogrammes pour des choix divers de l'origine x_0 (voir figure 7).

```
number<-rbinom(1,50,.5)
mixture<-rnorm(50)
mixture[1:number]<-mixture[1:number]+3 # on genère les donnees
par(mfrow=c(3,2))
hist(mixture,xlab=" ",breaks=-2.13:4.87,col=0,freq=F)
hist(mixture,xlab=" ",breaks=-2.33:5.67,col=0,freq=F)
```

```
hist(mixture,xlab=" ",breaks=-2.42:5.58,col=0,freq=F)
hist(mixture,xlab=" ",breaks=-2.53:5.47,col=0,freq=F)
hist(mixture,xlab=" ",breaks=-2.6:5.4,col=0,freq=F)
hist(mixture,xlab=" ",breaks=-3.1:4.9,col=0,freq=F)
```

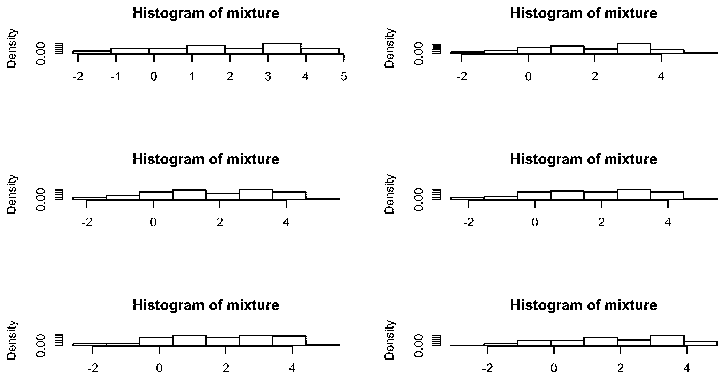

FIGURE 7 – Variabilité de l'histogramme en fonction de l'origine

L'estimation lisse de la densité permet de corriger les défauts de l'histogramme. L'une des raisons du manque d'efficacité de l'histogramme à représenter une densité continue est la rigidité de ses classes. La hauteur d'un rectangle est seulement fonction du nombre de données contenues dans sa base sans tenir compte de leur répartition sur cet intervalle. Ce problème disparaît si on laisse les intervalles varier avec les données. L'idée de départ de l'estimation par noyau est de placer un rectangle de taille fixe $1/n$ autour de chaque point pour être sûr que si plusieurs rectangles se chevauchent on les empile de manière à ce que l'aire totale soit toujours égale à 1. On aborde là le domaine de la statistique non paramétrique sur lequel nous ne nous attarderons pas dans ce document. Toutefois on utilisera souvent une estimation graphique de la densité avec la commande `density()`. Le choix de la taille de la fenêtre dans l'estimation par noyau est très difficile et certaine règle sont définies dans R (on pourra les consulter avec une aide en ligne sur la fonction `density()`) . Ainsi par exemple pour les données d'éruption dont nous avons tracé l'histogramme on obtient (voir figure 8) en utilisant le règle de Sether et Jones `sj` et un noyau gaussien :

```
d <- density(faithful$eruptions, bw = "sj")
plot(d, main="Estimation par noyau de la densité")
```

L'impression donnée par le tracé de la figure 8 est analogue à celle que nous avions obtenue avec le tracé de l'histogramme. La densité inconnue est probablement bimodale et assez lisse.

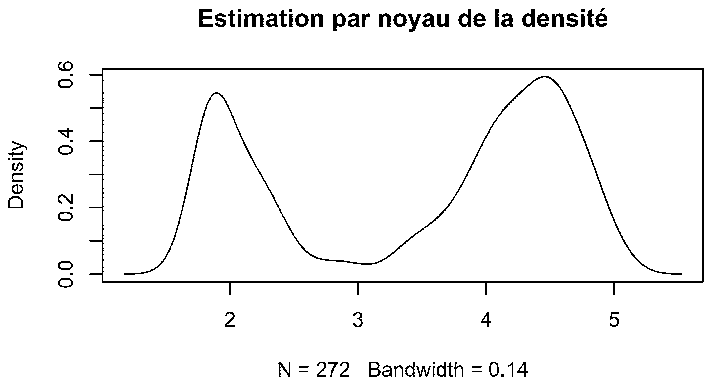

FIGURE 8 – Estimation par méthode du noyau

Le diagramme en boîte ou « à moustaches » (box plot en anglais) permet de résumer graphiquement les données de manière plus succinte. On peut de manière très rapide apprécier la symétrie de la distribution des données et la présence de données suspectes. Dans son utilisation la plus simple un diagramme à moustaches est une boîte rectangulaire délimitée par un côté (le quartile d'ordre 0.25), une barre découpant le rectangle en la médiane des données, l'autre côté étant situé sur le quartile d'ordre 0.75. Les moustaches s'étendent du min au max des données, avec la convention de les rétrécir à une longueur égale à 1,5 fois celle du rectangle. Tout point au-delà est indiqué par un tracé ponctuel. Par exemple, les commandes suivantes produisent le diagramme de la figure 9 :

```
boxplot(eruptions,main="éruptions",horizontal=TRUE)
```

Ce tracé montre que la distribution de la variable `éruptions` n'est pas symétrique. Elle est chargée à gauche.
Il peut être intéressant de comparer deux distributions en traçant conjointement les box-plot de chacune d'elle. Par exemple on pourra tracer les box-plot des variables `éruptions` et `wt` du dataframe `faithful` avec la commande :

```
boxplot(faithful,main="Données faithful",horizontal=TRUE)
```

Une autre manière d'examiner la distribution des valeurs d'une variable est de tracer sa fonction de répartition empirique. Cette dernière est calculée à l'aide de la commande `ecdf()`. Par exemple :

```
f<-ecdf(eruptions)
plot(f, main="Répartition empirique",cex=0.5)
```

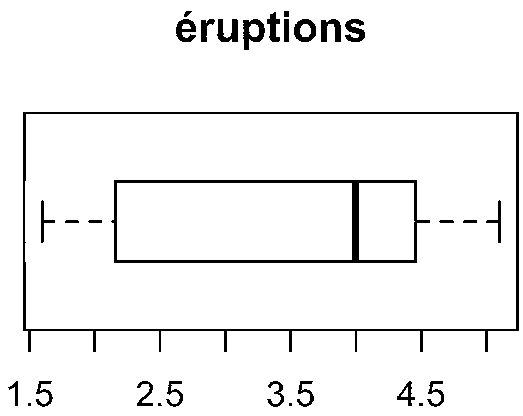

FIGURE 9 – Diagramme à moustaches

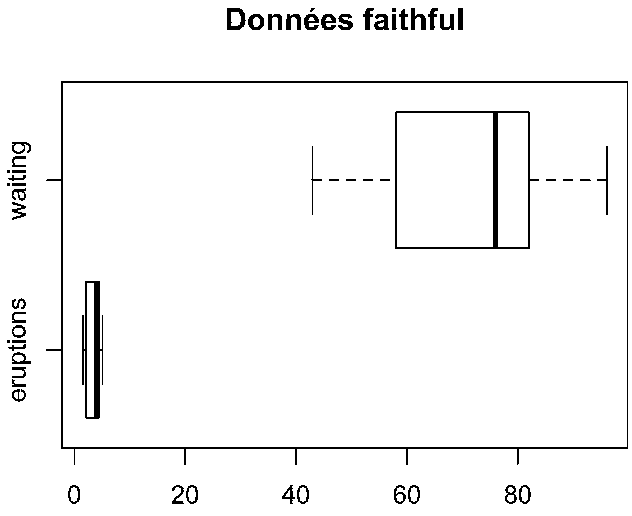

FIGURE 10 – Deux diagrammes en boîte

produit la figure 11.

Nous n'avons illustré que quelques unes des fonctions graphiques de base et surtout uniquement pour des cas unidimensionnels. Pour les cas multi-dimensionnels on pourra

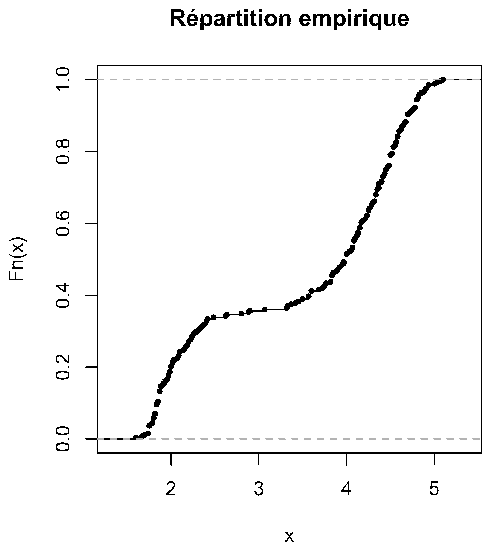

FIGURE 11 – Fonction de répartition empirique

aussi utiliser les fonctions suivantes.

Autres fonctions graphiques	
`pairs()`	trace tous les couples des variables
`persp()`	tracé 3D en perspective et en couleur
`pie()`	diagramme circulaire (cmembert)
`qqplot()`	quantile-quantile pour comparer deux distributions
`ts.plot()`	tracé de séries chronologiques

```
data(mtcars)
pairs(mtcars[,-c(2,7,8,9,10,11)])
```

produit la matrice de tracés de la figure 12.

1.5 Probabilités et Statistique avec R

Lois usuelles et génération de données aléatoires

Le logiciel R permet d'effectuer des calculs avec toutes les lois de probabilité usuelles, et aussi de simuler des échantillons issus de ces lois. Le tableau suivant résume les différentes lois implémentées dans R.

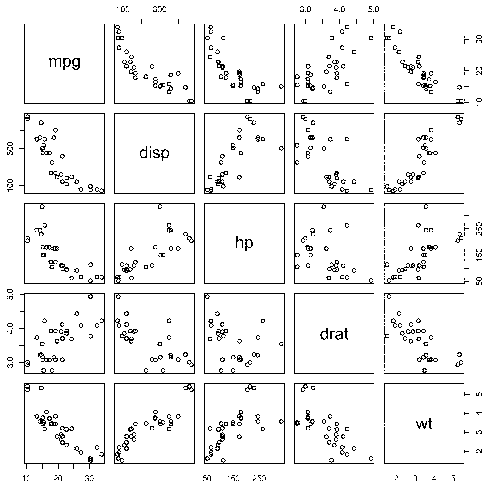

FIGURE 12 – Matrice de tracés

Loi	**appelation R**	**Arguments**
bêta	`beta`	`forme 1, forme 2`
binomiale	`binom`	`size, prob`
chi deux	`chisq`	`df` (degrés de liberté)
uniforme	`unif`	`min, max`
exponentielle	`exp`	`rate`
Fisher	`f`	`df1, df2`
gamma	`gamma`	`forme, echelle`
géometrique	`geom`	`prob`
hypergéometrique	`hyper`	`m, n, k` (taille échantillon)
binomiale négative	`nbinom`	`size, prob`
normale	`norm`	`mean, sd`
Poisson	`pois`	`lambda`
Student	`t`	`df`
Weibull	`weibull`	`forme, echelle`

Pour certaines lois, les paramètres ont des valeurs par défaut : parmi les plus utilisées, la loi uniforme `unif` porte par défaut sur l'intervalle $[0, 1]$, et la loi normale `norm` est centrée réduite par défaut.
Pour effectuer un calcul avec une de ces lois, il suffit d'utiliser comme fonction l'une des appellations R ci-dessus avec le préfixe `d` pour une densité, `p` pour une fonction de répartition, `q` pour une fonction quantile et `r` pour un tirage aléatoire. Voici quelques exemples :

```
x <- rnorm(100)          # 100 tirages, loi N(0,1)
w <- rexp(1000, rate=.1) # 1000 tirages, loi exponentielle
```

```
dpois(0:2, lambda=4)     # probablités de 0,1,2 pour la loi Poisson(4)
pnorm(12,mean=10,sd=2)   # P(X < 12) pour la loi N(10,4)
qnorm(.75,mean=10,sd=2)  # 3ème quartile de  la loi N(10,4)
```

Voici à titre d'exemple une utilisation de la génération de nombres aléatoires pour évaluer une intégrale numérique (Méthode de Monte Carlo). Soit g une fonction réelle intégrable sur un intervalle $[a, b]$. Nous souhaitons calculer une valeur approchée de $I = \int_a^b g(x)dx$. Générons un échantillon $(x_1, x_2, \ldots, x_n)$ de taille n de loi uniforme sur $[a, b]$ et calculons :

$$\hat{I}_n = (b-a)\frac{1}{n}\sum_{i=1}^{n} g(x_i) \ .$$

La *loi des grands nombres* assure la convergence presque sûre, quand n tend vers l'infini, de $\hat{I}_n$ vers I, et donc pour n grand, $\hat{I}_n$ est une approximation de I (moins bonne, certes, que celle obtenue par quadrature numérique). Considérons le calcul de l'intégrale $\int_0^{\pi/2} \sin(x)\exp(-x^2)dx$ par simulation de Monte Carlo, et son calcul numérique par la fonction `integrate` :

```
u <- runif(1000000, min=0, max=pi/2)
I <- pi/2* mean(sin(u)*exp(-u^2))
f<-function(x){sin(x)*exp(-x^2)}
integrate(f,lower=0,upper=pi/2)
```

Tracés de densités et de fonctions de répartitions

Des tracés de densités de probabilité ou de fonctions de répartition de lois diverses peuvent s'obtenir à l'aide de la function `plot()`. Ainsi par exemple pour tracer la densité (répartition des masses) d'une loi binomiale avec $n = 10$ and $p = .25$, reproduite dans la figure 13, on exécute dans R :

```
x <- 0:10
y <- dbinom(x, size=10, prob=.25)    # évalue les probas
{plot(x, y, type = "h", lwd = 30,
      main = "Densité Binomiale avec \n n
      = 10, p = .25", ylab = "p(x)", lend ="square")}
```

Pour cet exemple, nous avons d'abord créé le vecteur `x` contenant les entiers allant de 0 à 10. Nous avons ensuite calculé les probabilités qu'une variable de loi binomiale $B(10, 0.25)$ prenne chacune de ces valeurs, par `dbinom`. Le type de tracé est spécifié avec l'option `type=h` (lignes verticales d'un diagramme en bâtons), épaissies grâce à l'option `lwd=30`. L'option `lend ="square"` permet de tracer des barres rectangulaires. Nous avons ensuite ajouté des légendes.

Pour tracer des densités de lois absolument continues ou des fonctions de répartitions de celles-ci, on peut utiliser la fonction `curve()`. Par exemple, le tracé de la

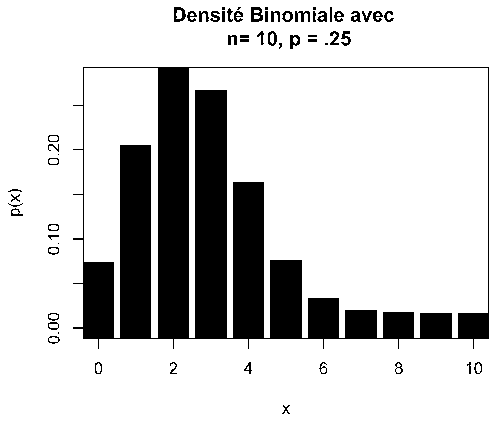

FIGURE 13 – Loi binomiale $\mathcal{B}(10, 0.25)$.

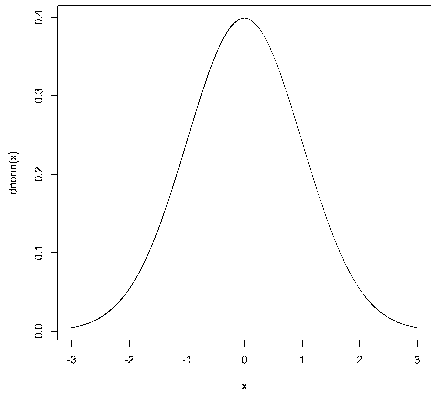

FIGURE 14 – Densité de la loi $\mathcal{N}(0, 1)$

densité d'une loi Gaussienne centrée réduite sur l'intervalle $[-3, 3]$ s'obtient avec la commande `curve(dnorm(x), from = -3, to = 3)` qui réalise la figure 14. alors que la commande `curve(pnorm(x, mean=10, sd=2), from = 4, to = 16)` produit le tracé de la fonction de répartition d'une loi $\mathcal{N}(10, 4)$ sur l'intervalle $[4, 16]$: Notons que la fonction `curve()` permet également de superposer une courbe sur un autre tracé (dans ce cas il est inutile de spécifier `from` et `to`). Essayons par exemple de comparer l'histogramme des fréquences des valeurs obtenues par un tirage de 1000 nombres selon la loi $\mathcal{N}(0, 1)$ avec la densité de la loi $\mathcal{N}(0, 1)$:

```
simu <- rnorm(1000)
{hist(simu, prob=T, breaks="FD",
      main="Histogramme de 1000 tirages N(0,1)")}
curve(dnorm(x), add=T)
```

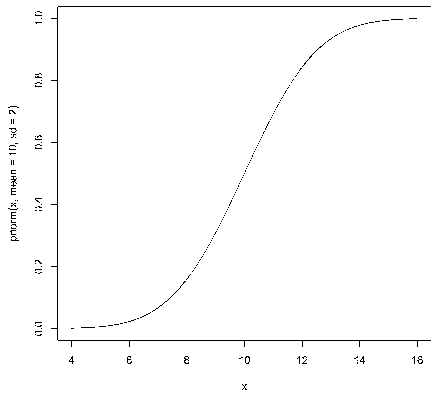

FIGURE 15 – Fonction de répartition de la loi $\mathcal{N}(10, 4)$

qui donne la figure 16.

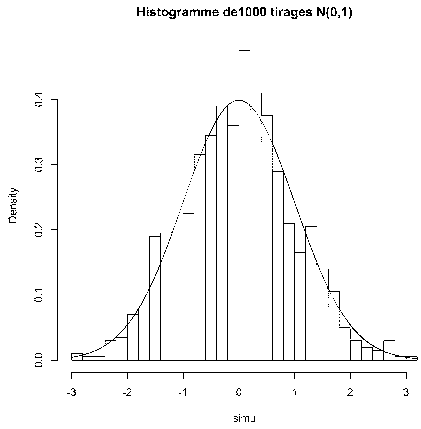

FIGURE 16 – Histogramme d'un échantillon de taille 1000 de la loi $\mathcal{N}(0, 1)$.

La fonction `sample` permet d'extraire des échantillons d'un ensemble fini quelconque, avec ou sans remise (option `replace`). En voici quelques exemples :

```
sample(1:100, 1)             # choisir un nombre entre 1 et 100
sample(1:6, 10, replace = T) # lancer un dé 10 fois
sample(1:6, 10, T, c(.6,.2,.1,.05,.03,.02))   #  un dé non équilibré
urn <- c(rep("red",8),rep("blue",4),rep("yellow",3))
sample(urn, 6, replace = F)  # tirer 6 boules dans cette urne
```

Notons enfin que bien d'autres lois de probabilités moins usuelles sont disponibles dans des librairies séparées (`evd`, `gld`, `stable`, `normalp`,...).

Notions de statistique inférentielle

Un des buts de la statistique est d'estimer les caractéristiques d'une loi de probabilité à partir d'un nombre fini d'observations indépendantes de cette dernière. Par exemple, on pourrait s'imaginer observer un échantillon empirique issu d'une loi normale de moyenne et de variance inconnue avec pour but l'identification de ces paramètres inconnus. Ainsi par exemple la loi des grands nombres justifierait l'estimation d'une espérance mathématique par la moyenne empirique, et on comprend qu'intuitivement que plus la taille de l'échantillon est importante, meilleure sera l'estimation. La théorie de la statistique inférentielle permet en particulier d'évaluer de manière probabiliste la qualité de ce type d'identifications. Cette section en donne quelques illustrations sommaires avec l'aide de R.

Estimation ponctuelle

Nous savons que le ou les paramètres d'une loi de probabilité résument de manière théorique certains aspects de cette dernière. Les paramètres les plus habituels sont l'espérance mathématique, notée μ, la variance σ^2, une proportion π, une médiane m, des quantiles, etc. Ce ou ces paramètres, génériquement notés θ sont en général inconnus et devront en général être approchés de manière optimale par des statistiques $\hat{\theta}$ calculées sur des échantillons aléatoires issus de la loi (ou population) étudiée. En pratique, la statistique calculée dépend de l'échantillon tiré et produira une estimation variable du paramètre inconnu. Ainsi par exemple si le paramètre inconnu est l'espérance μ, la moyenne empirique d'un échantillon $X_1, \ldots, X_n$ issu de la loi est la variable aléatoire *moyenne empirique* $\hat{\mu} = \bar{X}_n = \sum_{i=1}^{n} X_i/n$ elle même distribuée selon une certaine loi de probabilité. Il est important de se souvenir que l'on ne réalise un échantillon qu'une seule fois et donc que l'on ne dispose que d'une seule valeur de l'estimateur $\hat{\theta}$ (estimation ponctuelle).

La loi forte des grands nombres permet d'affirmer que pour une loi admettant une espérance μ et une variance σ^2, la moyenne empirique $\bar{X}_n$ converge vers μ alors que le théorème de la limite centrale dit que la loi de la moyenne empirique $\bar{X}_n$ peut être approchée de mieux en mieux quand la taille de l'échantillon tend vers l'infini par une loi normale d'espérance μ et de variance σ^2/n. On peut illustrer ces propriétés par des expériences aléatoires appropriés, programmées avec R de la manière suivante.

Pour un échantillon de la loi uniforme sur $[0, 1]$ de taille $n = 1000$, calculons les moyennes empiriques successives et traçons la moyenne empirique en fonction de la taille de l'échantillon. On observe que la moyenne empirique converge bien vers la valeur $\mu = 0.5$ représentée par une droite horizontale rouge sur le graphique 17.

```
n<-1000
X<-runif(n)
Y<-cumsum(X)
N<-seq(1,n, by=1)
plot(N, Y/N)
abline(h=0.5,col="red")
```

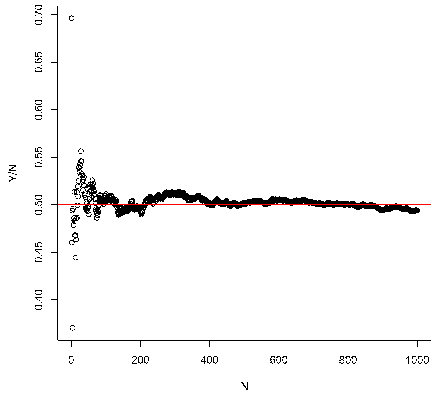

FIGURE 17 – Illustration de la loi des grands nombres

Si l'on simule maintenant 500 réalisations d'échantillons de tailles respectives $n = 10$ et $n = 100$, issus d'une loi uniforme sur $[0, 1]$ on peut tracer sur un même graphique l'histogramme des moyennes empiriques associées et illustrer le fait que cet histogramme est proche de la densité de la loi normale de moyenne $\mu = 0,5$ et de variance σ^2/n avec $\sigma^2 = 1/12$.

```
X<-matrix(runif(500*10), ncol=10, nrow=500)
X1<-apply(X,1,mean) # 500 tirages de moyenne empirique
Y<-matrix(runif(500*100), ncol=100, nrow=500)
Y1<-apply(Y,1,mean) # 500 tirages de moyenne empirique
par(mfrow=c(1,2))
hist(X1,main="Taille d'échantillon 10",prob=T)
curve(dnorm(x,0.5,sqrt(1/120)),add=T)
hist(Y1,main="Taille d'échantillon 100",prob=T)
curve(dnorm(x,0.5,sqrt(1/1200)),add=T)
```

Intervalles de confiance et test d'hypothèses

Un domaine très important de la statistique inférentielle est la construction d'intervalles de confiance (estimation ensembliste) et la construction de tests d'hypothèses sur les paramètres des lois qui régissent les observations faites. Le logiciel R couvre l'essentiel des besoins en probabilités et statistiques élémentaires avec en particulier ceux utiles pour l'estimation par intervalles de confiance et des tests de base, ainsi que plusieurs outils d'ajustements graphiques de distributions. Nous allons en donner un aperçu dans les sous-paragraphes suivants.

Estimation ensembliste et tests d'une proportion

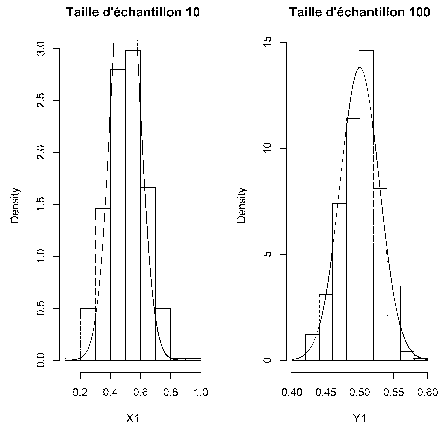

FIGURE 18 – Illustration de la convergence en loi

L'illustration la plus fréquente de la notion d'estimation ensembliste est l'estimation par intervalle de confiance d'une proportion inconnue π au vu d'une réalisation d'un échantillon de loi de Bernoulli. Par exemple, si lors d'un sondage d'une population vous observez la réponse de 100 personnes tirées au hasard dans la population et que vous observez que 42 parmi elles aiment la marque X, que pouvez vous dire de la probabilité π qu'une personne tirée au hasard dans la population aime la marque X ? Une première façon de répondre à cette question serait de proposer, comme dans les paragraphes précédents, une estimation ponctuelle de cette probabilité π, donc pour cet exemple, $\hat{\pi} = 0.42$. Cette estimation dépend trop de la réponse du sondage et pour une autre réalisation peut être différente. Il serait donc opportun de proposer plutôt une fourchette de valeurs pour π qui tienne compte de la variabilité de l'estimateur. Examinons cela de près : notons $X_1, \ldots, X_n$ les variables de Bernoulli $B(1, \pi)$ indépendantes modélisant la réponse éventuelle de chaque individu parmi les n tirés au hasard dans la population (dont l'effectif est très grand). Nous avons vu que l'estimateur ponctuel de π est la moyenne empirique

$$\hat{\pi}_n = \frac{X_1 + X_2 + \cdots + X_n}{n},$$

et que, si n est grand, d'après le théorème de la limite centrale, on peut approcher la loi de la variable

$$Z = \frac{\hat{\pi}_n - \pi}{\sqrt{\pi(1-\pi)}/\sqrt{n}},$$

par la loi $\mathcal{N}(0, 1)$ d'une variable aléatoire normale centrée réduite. Si l'on est dans ces conditions on peut donc évaluer la distance de Z à 0 avec une certaine probabilité. Par exemple, on peut affirmer, grâce aux propriétés de la loi normale, que $Z \in [-1, +1]$ avec une probabilité proche de 0.68, que $Z \in [-2, +2]$ avec une probabilité proche de 0.95, etc. En fait l'approximation normale de l'estimateur $\hat{\pi}_n$, permet de construire un

intervalle de confiance pour π de niveau $(1-\alpha)$ avec $0 < \alpha < 1$ en résolvant l'inégalité

$$|\pi - \hat{p}_n| \leqslant \Phi^{-1}(1-\alpha/2)\sqrt{\pi(1-\pi)}\sqrt{n}. \quad (1)$$

Cela se fait par résolution d'une inéquation du second degré, mais à titre de simplification, après avoir ré-écrit l'équation (1) sous la forme d'un intervalle $\hat{\pi}_n \pm \Phi^{-1}(1-\alpha/2)\sqrt{\pi(1-\pi)}\sqrt{n}$, nous allons approcher l'écart-type $\sqrt{\pi(1-\pi)}\sqrt{n}$ dans les bornes de cet intervalle par $\sqrt{\hat{\pi}(1-\hat{\pi})}\sqrt{n}$, en justifiant cette approximation par la loi forte des grands nombres. Pour notre exemple numérique du début de ce paragraphe, on peut donc proposer pour la proportion inconnue π d'individus préférant la marque X, la fourchette approchée $[0.3232643, 0.5167357]$ calculée avec

```
0.42 - qnorm(0.975)*sqrt(0.42*0.58)/sqrt{100}
0.42 + qnorm(0.975)*sqrt(0.42*0.58)/sqrt{100}
```

À titre d'exercice on pourra résoudre l'inégalité (1) et se passer ainsi de l'utilisation de la loi des grands nombres pour le calcul de l'intervalle de confiance.

Nous allons poursuivre cet exemple en illustrant par simulation le fait qu'un intervalle de confiance ne recouvre pas toujours la vraie valeur du paramètre π. Ceci ce réalise de manière simple avec la fonction `matplot` et les commandes suivantes, qui produisent la figure 19 :

```
m <- 50; n<-20; Pi <- .5;  #  20  pièces à lancer   50 fois
p <- rbinom(m,n,Pi)/n # simuler et calculer l'estimation
ET<- sqrt(p*(1-p)/n) # estimer l'écart-type
alpha = 0.10 # seuil de signification
zstar <-qnorm(1-alpha/2)
matplot(rbind(p-zstar*ET, p+zstar*ET),rbind(1:m,1:m),type="l",lty=1)
abline(v=Pi) # trace la  verticale en Pi
```

Bien évidemment la notion d'intervalle de confiance pour un paramètre est la notion duale d'un test bilatéral sur ce paramètre. Ainsi tester, au niveau de signification α, que la vraie proportion est π_0 contre son alternative $\pi \neq \pi_0$ au vu de la réalisation d'un échantillon, équivaut à regarder si l'intervalle de confiance pour π an niveau $(1-\alpha)$ recouvre ou non π_0. Ainsi pour notre exemple du sondage à propos de la marque X, ceci est réalisé avec la fonction `prop.test`.

```
prop.test(42,100,conf.level=0.95)
```

qui donne :

```
1-sample proportions test with continuity correction

data:  42 out of 100, null probability 0.5
```

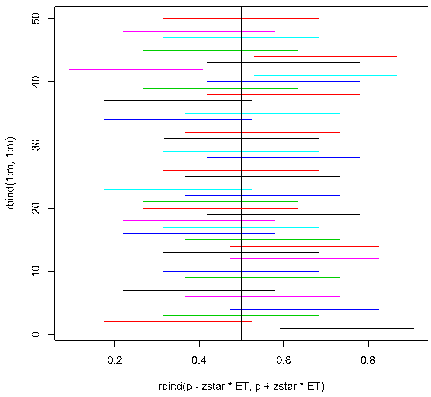

FIGURE 19 – Combien d'intervalles de confiance de niveau 80 % recouvrent-ils la vraie proportion $\pi = 0.5$?

```
X-squared = 2.25, df = 1, p-value = 0.1336
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.3233236 0.5228954
sample estimates:
   p
0.42
```

On notera les bornes de l'intervalle corrigées, qui différent légèrement de celles que nous avions déterminé avec la loi des grands nombres. Le résultat affiché est une liste, dont on peut extraire les composants. Par exemple pour l'intervalle de confiance :

```
prop.test(42,100,conf.level=0.95)$conf.int
```

Les tests et l'estimation par intervalle issus d'autres lois suivent la même configuration.

Estimation ensembliste et tests sur la moyenne

Considérons à titre d'exemple l'estimation ensembliste de la moyenne μ au niveau de confiance $(1 - \alpha)$ au vu d'une réalisation d'un échantillon gaussien $X_1, \ldots, X_n$ d'espérance μ et de variance connue σ^2. Nous savons que l'intervalle de confiance pour μ est de la forme $\bar{X}_n \pm \Phi^{-1}(1 - \alpha/2)\sigma/\sqrt{n}$. On peut construire une fonction simple pour son calcul comme suit :

```
ICsimple <-function(x,sigma,niv.conf=0.95){
n <- length(x)
xbar<-mean(x)
alpha <- 1-niv.conf
zstar <- qnorm(1-alpha/2)
```

```
ET <- sigma/sqrt(n)
xbar + c(-zstar*ET,zstar*ET)
}
```

qui donne, pour une réalisation `x` :

```
x<-c(175,176,173,175,174,173 ,173 ,176,173,179)
ICsimple(x,1.5)
[1] 173.7703 175.6297
```

De manière plus réaliste, la variance est plutôt inconnue et l'intervalle de confiance est calculé à l'aide de la loi de Student. En effet, si s est la racine carrée de l'estimation sans biais de la variance, on sait que la statistique $(\bar{X}_n - \mu)/\sigma/\sqrt{n}$ suit la loi de Student à $n-1$ degrés de liberté d'où l'intervalle de confiance $\bar{X}_n \pm qt(1-\alpha/2, n-1) * \sigma/\sqrt{n}$. Ainsi par exemple pour la réalisation x de l'exemple précédent on trouve comme intervalle de confiance pour μ au niveau de confiance de 0.95, l'intervalle $[173.3076, 176.0924]$, qui comme nous le constatons est plus long car la loi de Student a des ailes plus lourdes que celles d'une loi gaussienne.

L'implémentation du test de Student est réalisée dans R avec la fonction `t.test()` qui non seulement calcule le test de l'hypothèse désirée sur μ mais retourne aussi l'intervalle de confiance. Cette même fonction permet de comparer les moyennes de deux échantillons gaussiens appariés ou non. Ainsi par exemple pour tester si x est un échantillon d'espérance $\mu = 170$, on utilisera `t.test(x,alternative="two.sided", mu=170)` qui donne :

```
One Sample t-test

data:  x
t = 7.6356, df = 9, p-value = 3.206e-05
alternative hypothesis: true mean is not equal to 170
95 percent confidence interval:
 173.3076 176.0924
sample estimates:
mean of x
    174.7
```

et rejette l'hypothèse nulle puisque la p-valeur est $3.206\ 10^{-5}$. Pour un test unilatéral, `t.test(x,alternative="less", mu=170)` donne :

```
One Sample t-test

data:  x
t = 7.6356, df = 9, p-value = 1
alternative hypothesis: true mean is less than 170
```

```
95 percent confidence interval:
     -Inf 175.8284
sample estimates:
mean of x
    174.7
```

et accepte l'hypothèse $\mu \geqslant 170$.

Test d'adéquation et d'indépendance

Pour terminer ce paragraphe, voici les tests du chi-deux d'ajustement et de contingence. Un test d'ajustement permet de décider si la distribution des valeurs d'une échantillon est correctement ajustée par une loi de probabilité spécifiée à l'avance. Par exemple, si on lance un dé 150 fois de manière indépendante et que l'on observe les fréquences suivantes on peut se demander si le dé est bien équilibré.

face	1	2	3	4	5	6
fréq.	22	21	22	27	22	36

Pour cela il suffit d'évaluer la distance entre les fréquences observées et celles qui auraient dû être observées si le dé était bien équilibré. Cette distance est, dans le cas équilibré, distribuée selon une loi du chi deux à 5 degrés de liberté et est obtenue par :

```
freq <- c(22,21,22,27,22,36)
probs <- c(1,1,1,1,1,1)/6
chisq.test(freq,p=probs)
```

qui donne :

```
Chi-squared test for given probabilities
data:  freq
X-squared = 6.72, df = 5, p-value = 0.2423
```

confortant ainsi l'hypothèse d'un dé bien équilibré.

La même fonction peut être également utilisée pour tester l'indépendance des variables dans une table de contingence. On suppose observer deux variables catégorielles X et Y, respectivement à I et J modalités. On observe ainsi indépendamment le couple (X, Y) sur une population de n sujets. Soit alors N_{ij} le nombre de sujets pour lesquels on a $(X, Y) = (i, j)$, de sorte que $\sum_{i=1}^{I} \sum_{j=1}^{J} N_{ij} = n$. Les comptages observés peuvent être organisés en un tableau à double entrée :

$$\begin{array}{c|cccc} & Y=1 & Y=2 & \dots & Y=J \\ X=1 & n_{11} & n_{12} & \dots & n_{1J} \\ X=2 & n_{21} & n_{22} & \dots & n_{2J} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ X=I & n_{I1} & n_{I2} & \dots & n_{IJ} \end{array}$$

Le nombre total d'entrées de ce tableau de contingence est $N = IJ$. Les sommes marginales seront notées n_{i+}, n_{+j} et $n_{++} = n$.

À titre d'exemple considérons, dans une étude d'accidents de voiture, les variables X (port de ceinture, oui ou non) et Y (gravité de la blessure, aucune, minimale, légère, grave) avec pour tableau observé :

		aucune	minimale	légère	grave
Ceinture	Oui	128213	647	359	42
	Non	65963	4000	2642	303

On peut alors se poser la question de savoir si le port de ceinture modifie la répartition des blessures, autrement dit, est-ce que les lignes de ce tableau sont indépendantes. Ceci est assez simple à réaliser dans R.

```
avecceinture<-c(12813,647,359,42)
sansceinture<- c(65963,4000,2642,303)
chisq.test(data.frame(avecceinture,sansceinture))
```

qui donne

```
Pearson's Chi-squared test

data:  data.frame(avecceinture, sansceinture)
X-squared = 59.224, df = 3, p-value = 8.61e-13
```

et montre l'influence évidente du port de la ceinture.

Régression linéaire simple

Dans ce paragraphe nous allons examiner une modélisation particulière pour étudier (décrire, prédire, ...) une variable appelée variable *expliquée* (ou *réponse* ou *variable dépendante*) sur la base d'autres variables, appelées *explicatives* (ou *covariables* ou *variables indépendantes*). Nous nous resteindrons au cas d'une seule variable explicative d'où la notion de *régression simple*. Un modèle de régression linéaire simple décrit le cas particulier d'une seule variable explicative X de type quantitatif (aléatoire ou déterministe) pour un phénomène physique régi par l'équation

$$Y = aX + b + \text{bruit aléatoire}, \tag{2}$$

le bruit étant une variable aléatoire centrée. À titre d'exemple, il est généralement admis que la fréquence théorique maximale du pouls y d'une personne est relié en moyenne à l'âge noté x par une équation du type $y = ax + b$. Il y a deux manières d'interpréter le modèle (2) : soit on observe des réalisations indépendantes (y_i, x_i), $i = 1, \ldots, n$ du couple de variables aléatoires (Y, X) (comme pour l'exemple de l'étude de la fréquence théorique maximale du pouls), soit on observe des réalisations indépendantes de Y en des valeurs de la variable explicative $x_1, \ldots, x_n$ fixées à l'avance. Ceci ne modifie pas

les calculs ni l'interprétation des résultats à condition de supposer que dans le cas où X est aléatoire, il est indépendant du bruit. Les coefficients a et b sont inconnus et on peut vouloir les identifier, par exemple pour prédire la réponse y en un point x' quelconque. Dans la suite nous nous proposons d'illustrer les calculs et l'interprétation avec un exemple simulé, ainsi que sur des données expérimentales.

Considérons d'abord l'exemple d'une régression linéaire simple pour laquelle la droite de régression est donnée par l'équation $y = 1 + x/2$. Cette droite est représentée par la droite de couleur bleue dans le graphique de gauche de la figure 20. Les observations se font en des points x_i fixés, données par les abscisses $x_i = -8 + 2 * (i - 1) * 9$, $i = 1, \ldots, 9$ des points rouges, d'ordonnées $y_i = 1 + x_i/2$ situés sur la droite. En chaque point x_i on tire au hasard, 5 observations selon une loi gaussienne de moyenne x_i et de variance 1, représentées par les points bleus sur le graphique du centre. En réalité, l'ensemble des données observées pour cet exemple simulé est le nuage des points du dernier graphique. La fonction permettant de tracer des densités gaussiennes à la verticale est :

```
sideways.dnorm<-function(wx,wy,values=seq(-4,4,.1),magnify=4,...){
# ... est constitué des moyennes et écart-types,
# passés à la fonction dnorm
#
dens <- dnorm(x=values, ...)
x <- wx + dens * magnify
y <- wy + values
return(cbind(x,y))
}
```

et les graphiques de la figure 20 sont réalisés avec les commandes :

```
par(mfrow=c(1,3))
x<-seq(-8,8,2)
y <- 1 + .5*x
fit <- lm(y~x)
plot(x,y, xlim=c(-10,10), ylim=c(-3, 7), pch=19,
    cex=1.4,col="red",  main="équation de régression")
abline(fit, lwd=3, col="blue")
plot(x,y, xlim=c(-10,10), ylim=c(-8, 10), pch=19, cex=1.4
, main="Régression linéaire simple\navec erreurs gaussiennes"
, col="red")
abline(fit, lwd=3, col="blue")
where.normal.x<-sort(x)
xx<-NULL
zz<-NULL
# where.normal.x <- c(-4,0,4)
```

```
for(i in 1:length(where.normal.x)){
    where.x <- where.normal.x[i]
    where.y <- predict(fit, newdata=data.frame(x=where.x))
    xy <- sideways.dnorm(where.x=where.x, where.y=where.y, magnify=4)
    lines(xy)
    abline(h=where.y, lty=2,col="pink")
    abline(v=where.x, lty=2)
    xx<-c(xx,rep(where.x,5))
    z<-where.y+rnorm(5)
    zz<-c(zz,z)
    aux<-cbind(rep(where.x,5),z)
    points(aux,col="blue")
    }
plot(xx,zz,xlim=c(-10,10), ylim=c(-3, 7),main="les observations")
```

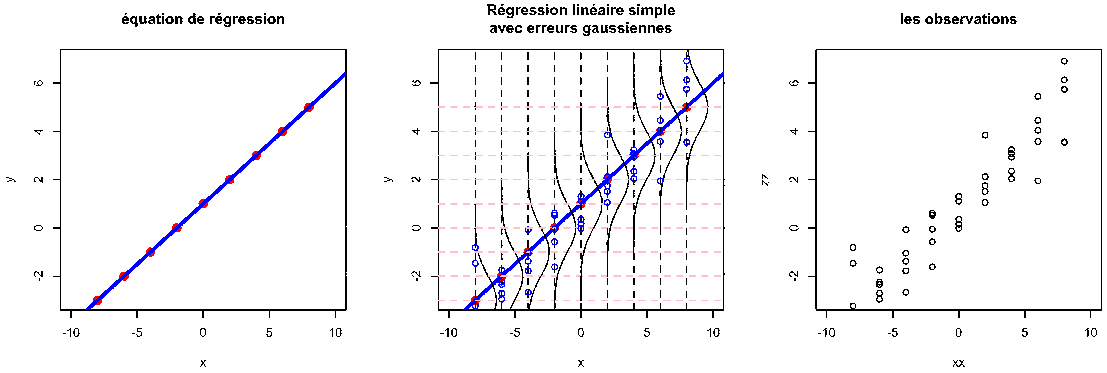

FIGURE 20 – Simulation selon un modèle de régression simple avec bruit gaussien

Pour continuer avec ce modèle simulé de régression linéaire simple, on voudrait à partir des $n = 45$ observations (x_i, y_i) estimer les coefficients a et b de la droite modélisant les y_i comme des réalisations de variables aléatoires Y_i, non corrélées, de moyennes $ax_i + b$ et de variance constante σ^2. Pour cela on utilise le principe des moindres carrés qui consiste à minimiser la fonction de perte :

$$\ell_2(a, b) = \sum_{i=1}^{n} [y_i - (ax_i + b)]^2,$$

et dont l'optimum est solution du système linéaire des équations normales :

$$\begin{cases} \sum_i^n x_i y_i - a \sum_i^n x_i^2 - b \sum_i^n x_i &= 0, \\ \sum_i^n y_i - a \sum_i^n x_i - bn &= 0. \end{cases}$$

En notant $\mathbf{y}$ le vecteur des observations, $\mathbf{x}$ le vecteur de composantes x_i et $\bar{\text{cov}}$, $\bar{v}(\mathbf{x})$, $\bar{y}$ et $\bar{x}$ les moments empiriques associés, on obtient les solutions :

$$\hat{a} = \frac{\bar{\text{cov}}(\mathbf{x}, \mathbf{y})}{\bar{v}(\mathbf{x})} \qquad \text{et} \qquad \hat{b} = \bar{y} - \hat{a}\bar{x}.$$

On remarquera ici que les estimateurs obtenus sont les versions empiriques des coefficients de régression $\beta_2 = \frac{\text{COV}(Y,X)}{\text{var}(X)}$ et $\beta_1 = E(Y) - \beta_2 E(X)$.

La dernière équation montre que la droite des moindres carrés passe par le centre de gravité du nuage $\{(x_i, y_i), \quad i = 1, \ldots, n\}$. On pouvait s'attendre à ce résultat car la meilleure prédiction de Y lorsque $x = \bar{x}$ doit bien être $\bar{y}$.

Pour ce cas particulier l'estimation sans biais de la variance σ^2 est donnée par

$$\hat{\sigma}^2 = \frac{1}{n-2} \sum_{i=1}^{n} [y_i - (\hat{a}x_i + \hat{b})]^2.$$

Pour l'exemple des données simulées, l'ajustement des moindres carrés s'obtient avec la commande `lm()` et on obtient la figure 21 avec les commandes

```
{plot(xx,zz,xlim=c(-10,10), ylim=c(-3, 7),
      main="Ajustement des moindres carrés")}
fitb<-lm(zz~xx)
abline(fitb,lty=2,col="blue",lwd=3)
abline(fit,lty=1,col="black")
```

Les sorties R associées à un ajustement d'un jeu de données par un modèle de régression linéaire à l'aide de la fonction `lm()` peut s'obtenir par un `summary()` de l'objet ajusté.

```
fitb<-lm(zz~xx)
summary(fitb)
```

qui donne

```
Call:
lm(formula = zz ~ xx)

Residuals:
     Min       1Q   Median       3Q      Max
-1.94795 -0.62929 -0.08101  0.43664  2.36308

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.86451    0.14747   5.862 5.79e-07 ***
xx           0.50503    0.02856  17.685  < 2e-16 ***
```

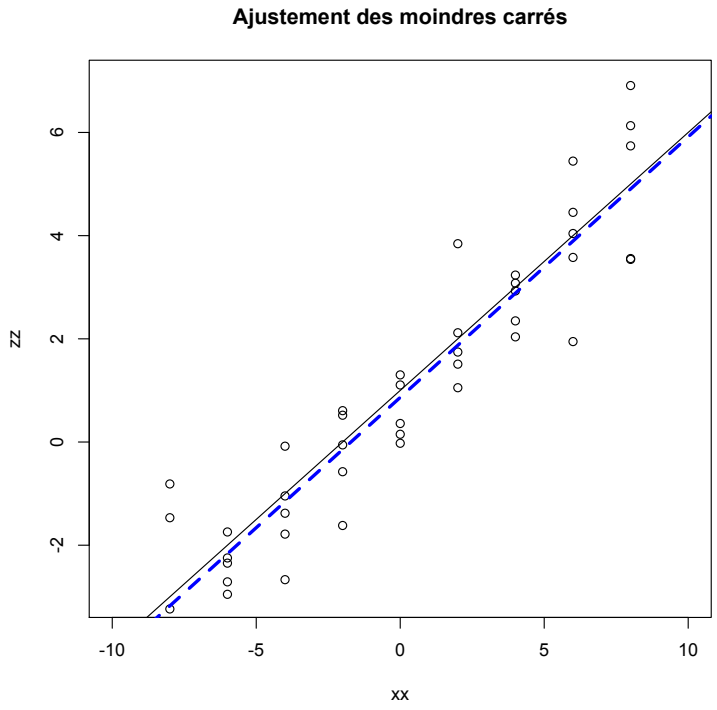

FIGURE 21 – Ajustement de la régression simple sur des données simulées. La droite en trait plein est le modèle exact, la droite en pointillés est estimée.

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9892 on 43 degrees of freedom
Multiple R-squared: 0.8791,Adjusted R-squared: 0.8763
F-statistic: 312.8 on 1 and 43 DF,  p-value: < 2.2e-16
```

On y retrouve une description par quantiles de la distribution des valeurs de la réponse (ici `zz`) puis les estimations ponctuelles de l'ordonnée à l'origine et de la pente, avec l'estimation de leurs écart-types et des p-valeurs des tests de nullité de chacun des paramètres (qui ne tiennent pas compte de leur corrélation). Enfin, l'estimation ponctuelle de l'écart-type est donnée sur la ligne suivant le tableau des coefficients. On y retrouve aussi les valeurs estimées du coefficient de détermination et la p-valeur du test d'absence d'influence du régresseur. Comme d'habitude, le résultat est une liste dont on peut extraire les composants. Par exemple pour l'ordonnée à l'origine, puis la pente :

```
fitb$coefficients[1]
fitb$coefficients[2]
```

2 Entraînement

2.1 Vrai ou faux

Vrai-Faux 1. Les lignes de commande suivantes affichent le vecteur ligne `[1] 1 2 3 4 5` : vrai ou faux et pourquoi ?

1. ☐ `(1,2,3,4,5)`
2. ☒ `c(1,2,3,4,5)`
3. ☒ `1:5`
4. ☐ `1,5`
5. ☐ `rep(1,5)`
6. ☒ `seq(1,5)`
7. ☒ `cumsum(rep(1,5))`
8. ☒ `which(rep(1,5)==1)`
9. ☐ `which(rbin(5,1,1)<1)`
10. ☒ `sort(sample(1:5,5))`
11. ☒ `sort(seq(5,1,-1))`
12. ☐ `for (i in 1:5){v[i] <- i}`
13. ☒ `v <- rep(1,5); for (i in 1:5){v[i] <- i}; v`
14. ☐ `v <- 1; for (i in 2:5){append(v,i)}; v`
15. ☒ `v <- 1; for (i in 2:5){v <- append(v,i)}; v`
16. ☒ `v <- 1; for (i in 2:5){v <- c(v,i)}; v`
17. ☐ `v <- 1; for (i in 2:5){v <- rbind(v,i)}; v`
18. ☐ `v <- rep(1,5); while (i<5){v[i] <- i}; v`
19. ☒ `v<- 1; i<-1; while (i<5){i<-i+1; v[i] <- i}; v`

Vrai-Faux 2. Les lignes de commande suivantes affichent le vecteur ligne `[1] 1 0.5 0.25 0.125 0.0625` : vrai ou faux et pourquoi ?

1. ☐ `2^-0:4`
2. ☐ `2^(-0:4)`
3. ☒ `2^-(0:4)`
4. ☐ `2^(-(0:4))`
5. ☐ `1/2^0:4`
6. ☒ `1/2^(0:4)`
7. ☐ `cumprod(rep(0.5,5))`
8. ☒ `cumprod(c(1,rep(0.5,4)))`

9. ☐ `v<-1; for (i in 1:4){v <- c(v,v/2)}; v`
10. ☐ `v<-rep(1,5); for (i in 0:4){v(i) <- 2^(-i)}; v`
11. ☐ `v<-1; for (i in 0:4){v[i] <- 1/2^i}; v`
12. ☒ `v<-1; for (i in 2:4){v[i] <- 1/2^i}; v`
13. ☐ `v<-1; while (v>0.1){v <- v/2}; v`
14. ☐ `x<-1; v<-x; while(x>0.1){x<-x/2; v<-c(v,x)}; v`

Vrai-Faux 3. Les lignes de commande suivantes affichent la matrice carrée dont la première ligne est le vecteur `0 1`, la seconde ligne est le vecteur `1 0` : vrai ou faux et pourquoi ?

1. ☐ `[[0,1];[1,0]]`
2. ☒ `rbind(c(0,1),c(1,0))`
3. ☒ `cbind(0:1,1:0)`
4. ☒ `matrix(c(0,1,1,0),2,2)`
5. ☐ `matrix(rep(c(0,1),2),2,2)`
6. ☐ `diag(rep(1,2))`
7. ☒ `1-diag(rep(1,2))`
8. ☐ `v <- c(1,0); cbind(v,v)`
9. ☒ `v <- c(1,0); rbind(sort(v),v)`
10. ☒ `c(1,0)%*%t(c(0,1))+c(0,1)%*%t(c(1,0))`
11. ☒ `1:0%*%t(0:1)+0:1%*%t(1:0)`
12. ☐ `1:0*t(0:1)+0:1*t(1:0)`
13. ☒ `toeplitz(c(0,1))`
14. ☒ `toeplitz(0:1)`

Vrai-Faux 4. Soit n un entier. Les lignes de commande suivantes affichent le vecteur de longueur $2n$ dont la coordonnée d'ordre $2i$ vaut i, les autres coordonnées étant nulles : vrai ou faux et pourquoi ?

1. ☒ `as.vector(rbind(rep(0,n),1:n))`
2. ☐ `matrix(cbind(rep(0,n),1:n),1,2*n)`
3. ☐ `matrix(rbind(rep(0,n),1:n),1,2*n)`
4. ☒ `as.vector(matrix(rbind(rep(0,n),1:n),1,2*n))`
5. ☐ `as.vector(matrix(rbind(1:n,rep(0,n)),2*n,1))`
6. ☒ `as.vector(0:1%*%t(1:n))`
7. ☐ `as.vector(1:n%*%t(1:0))`
8. ☐ `as.vector(1:n%*%t(0:1))`

9. ☐ `v<-0; for (i in 1:n){v[2*i]<-i}; v`
10. ☒ `v<-rep(0,2*n); for (i in 1:n){v[2*i]<-i}; v`
11. ☒ `v<-c(0,1); for (i in 2:n){v<-c(v,c(0,i))}; v`
12. ☐ `v<-c(0,1); for (i in 2:n){v<-c(v,0:i)}; v`

Vrai-Faux 5. Soit n un entier. Les lignes de commande suivantes affichent le vecteur dont la coordonnée d'ordre i vaut $\frac{i(i-1)}{2}$, pour $i = 1, \ldots, n$: vrai ou faux, et pourquoi ?

1. ☐ `1:n*0:n-1/2`
2. ☒ `1:n*0:(n-1)/2`
3. ☒ `1:n*seq(0,n-1)/2`
4. ☐ `v<-1:n; v^2-v/2`
5. ☒ `v<-1:n; (v^2-v)/2`
6. ☐ `v<-1:n; v%*%(v-1)/2`
7. ☐ `cumsum(0:n-1)`
8. ☒ `cumsum(0:(n-1))`
9. ☒ `cumsum(seq(0,n-1))`
10. ☐ `v<-rep(0,n); for (i in 1:n){v(i)<-i*(i-1)/2}; v`
11. ☒ `v<-rep(0,n); for (i in 1:n){v[i]<-i*(i-1)/2}; v`
12. ☒ `v<-0; for (i in 2:n){v<-c(v,i*(i-1)/2)}; v`

Vrai-Faux 6. Soit n un entier et $v = (v_1, \ldots, v_n)$ un vecteur d'entiers, tous compris entre 0 et 9. Les lignes de commande suivantes affichent le réel compris entre 0 et 1 dont les n décimales sont $v_1, \ldots, v_n$: vrai ou faux et pourquoi ?

1. ☐ `sum(v*10^-1:length(v))`
2. ☒ `sum(v*10^-(1:length(v)))`
3. ☒ `sum(v%*%10^-(1:length(v)))`
4. ☐ `paste("0",".",v)`
5. ☐ `paste(c("0",".",v))`
6. ☐ `paste(c("0",".",v),collapse='')`
7. ☒ `as.numeric(paste(c("0",".",v),collapse=''))`
8. ☒ `x<-0; d<-0.1; for (i in v){x<-x+d*i; d<-d/10}; x`
9. ☐ `x<-0; for (i in v){x<-x+i*10^(-i)}; x`

Vrai-Faux 7. Soit x un vecteur d'entiers tous compris entre 1 et 5. Les lignes de commande suivantes affectent à `v` un vecteur de taille 5, contenant les fréquences empiriques des entiers entre 1 et 5 : vrai ou faux et pourquoi ?

1. ☐ `v<-table(x)/length(x)`

2. ☐ `v<-table(factor(x,levels=1:5))/length(x)`
3. ☒ `v<-as.vector(table(factor(x,levels=1:5))); v/sum(v)`
4. ☐ `v<-rep(1,5); for (i in 1:5){v[i]<-which(x=i)}; v/sum(v)`
5. ☐ `v<-rep(1,5); for (i in 1:5){v[i]<-length(which(x==i))}; v/sum(v)`

Vrai-Faux 8. Soit x un vecteur numérique. Les lignes de commande suivantes retournent la moyenne de x : vrai ou faux et pourquoi ?

1. ☒ `mean(x)`
2. ☒ `sum(x)/length(x)`
3. ☐ `mean(table(x))`
4. ☐ `mean(as.vector(table(x))`
5. ☒

```
t <- table(x);
sum(as.numeric(row.names(t))*as.vector(t))/length(x)
```

Vrai-Faux 9. Soit x un vecteur numérique. Les lignes de commande suivantes retournent une médiane de x : vrai ou faux et pourquoi ?

1. ☒ `median(x)`
2. ☐ `quantile(x,0.5)`
3. ☒ `as.numeric(quantile(x,0.5))`
4. ☒ `as.vector(quantile(x,0.5))`
5. ☐ `x[length(x)/2]`
6. ☐ `sort(x)(round(length(x))/2)`
7. ☒ `sort(x)[floor(length(x)/2)]`

Vrai-Faux 10. Soit n un entier. Les lignes de commande suivantes retournent un échantillon de taille n de la loi de Bernoulli de paramètre $1/2$: vrai ou faux et pourquoi ?

1. ☐ `rbernoul(n,0.5)`
2. ☐ `rbinom(n,0.5)`
3. ☒ `rbinom(n,1,0.5)`
4. ☐ `runif(n,0,1)`
5. ☒ `floor(runif(n,0,2))`
6. ☒ `round(runif(n,0,1))`
7. ☐ `ceiling(runif(n,0,2))`
8. ☒ `ceiling(runif(n,0,2))-1`
9. ☒ `(1+sign(rnorm(10)))/2`

10. ☒ `(1+sign(rnorm(10,0,2)))/2`
11. ☐ `(1+sign(rnorm(10,2,0)))/2`
12. ☒ `ifelse(rnorm(n)<0,yes=0,no=1)`
13. ☐ `ifelse(runif(n,0,2)<0.5,yes=0,no=1)`
14. ☒ `rep(0,n); v[which(runif(n,0,1)<0.5)]=1`
15. ☐ `rep(n,0); v[which(runif(n,0,1)<0.5)]=1`
16. ☒ `ifelse(rgeom(n,0.5)>0,yes=1)`
17. ☒ `ifelse(rgeom(n,0.5)>0,yes=0,no=1)`
18. ☐ `sample(n,c(0,1))`
19. ☐ `sample(c(0,1),n)`
20. ☒ `sample(c(0,1),n,replace=TRUE)`

Vrai-Faux 11. Les lignes de commande suivantes retournent 0 : vrai ou faux et pourquoi ?

1. ☒ `qnorm(0.5)`
2. ☐ `qnorm(0.5,0.5)`
3. ☒ `dnorm(100)`
4. ☐ `pnorm(0.5)`
5. ☒ `pbinom(1,1,0,lower=F)`
6. ☐ `dbinom(1,1,1)`
7. ☒ `dbinom(2,1,0.5)`
8. ☒ `ppois(0,1000)`
9. ☐ `qgeom(0.6,0.5)`
10. ☒ `qgeom(0.4,0.5)`
11. ☒ `qt(0.5,3)`
12. ☐ `qchisq(0.5,3)`

Vrai-Faux 12. Soit x un échantillon de taille 1000 d'une loi d'espérance μ inconnue. Les lignes de commande suivantes retournent un vecteur à deux entrées dont les composantes sont les bornes d'un intervalle de confiance de niveau asymptotique 0.95 pour μ : vrai ou faux et pourquoi ?

1. ☐ `t.test(x)$conf.int`
2. ☒ `as.vector(t.test(x)$conf.int)`
3. ☐ `mean(x)+sd(x)*qnorm(0.975)*c(-1,1)`
4. ☒ `mean(x)+sd(x)*qnorm(0.975)*c(-1,1)/sqrt(length(x))`
5. ☐ `mean(x)+sd(x)*qnorm(0.025,0.975)/sqrt(length(x))`

6. ☐ `mean(x)+sd(x)*qnorm(c(0.025,0.975))/sqrt(length(x))`

Vrai-Faux 13. Soit x un échantillon de taille 1000 d'une loi d'espérance μ inconnue. Les lignes de commande suivantes retournent la p-valeur d'un test unilatéral de $\mathcal{H}_0$: $\mu = 0$ contre $\mathcal{H}_1$: $\mu > 0$: vrai ou faux et pourquoi ?

1. ☐ `t.test(x)$p.value`
2. ☒ `t.test(x,alternative="greater")$p.value`
3. ☐ `pnorm(mean(x)/sd(x))`
4. ☐ `pnorm(sqrt(length(x))*mean(x)/sd(x))`
5. ☒ `1-pnorm(sqrt(length(x))*mean(x)/sd(x))`
6. ☒ `pnorm(sqrt(length(x))*mean(x)/sd(x),lower.tail=F)`

2.2 Exercices

Exercice 1. Écrire (sans utiliser de boucle) les vecteurs suivants :

1. Nombres de 1 à 3 par pas de 0.1.
2. Nombres de 3 à 1 par pas de -0.1.
3. Carrés des 10 premiers entiers.
4. Nombres de la forme $(-1)^n n^2$ pour $n = 1, \ldots, 10$.
5. 10 "0" suivis de 10 "1".
6. 3 "0" suivis de 3 "1", suivis de 3 "2",..., suivis de 3 "9".
7. "1", suivi de 1 "0", suivi de "2", suivi de 2 "0",..., suivi de "8", suivi de 8 zéros, suivi de "9".
8. 1 "1" suivi de 2 "2", suivis de 3 "3",..., suivis de 9 "9".

Exercice 2. Écrire (sans utiliser de boucle) les matrices carrées d'ordre 6 suivantes :

1. Matrice diagonale, dont la diagonale contient les entiers de 1 à 6.
2. Matrice contenant les entiers de 1 à 36, rangés par lignes.
3. Matrice dont toutes les lignes sont égales au vecteur des entiers de 1 à 6.
4. Matrice diagonale par blocs, contenant un bloc d'ordre 2 et un d'ordre 4. Les 4 coefficients du premier bloc sont égaux à 2. Le deuxième bloc contient les entiers de 1 à 16 rangés sur 4 colonnes.
5. Matrice $A = \left((-1)^{i+j}\right)$, $i, j = 1, \ldots, 6$.
6. Matrice contenant des "1" sur la diagonale, des "2" au-dessus et au-dessous, puis des "3", jusqu'aux coefficients d'ordre $(1, 6)$ et $(6, 1)$ qui valent 6.

Exercice 3. Écrire la matrice $A = (a_{i,j})$ d'ordre 12 contenant les entiers de 1 à 144, rangés par lignes. Extraire de cette matrice les matrices B suivantes.

1. Coefficients $a_{i,j}$ pour $i = 1, \ldots, 6$ et $j = 7, \ldots, 12$.
2. Coefficients $a_{i,j}$ pour $i + j$ pair.
3. Coefficients $a_{i,j}$ pour $i, j = 1, 2, 5, 6, 9, 10$.

Exercice 4. Pour $(n = 100, p = 0.5)$, puis $(n = 1000, p = 0.5)$, $(n = 10000, p = 0.5)$, $(n = 1000, p = 0.3)$, $(n = 1000, p = 0.8)$:

1. Simuler un échantillon de n variables aléatoires de Bernoulli, de paramètre p,
   (a) en utilisant la fonction `rbinom`
   (b) en utilisant la fonction `runif`
   (c) en utilisant la fonction `sample`
2. Calculer les fréquences de 0 et de 1 dans l'échantillon,
   (a) en utilisant la fonction `sum`
   (b) en utilisant la fonction `which`
   (c) en utilisant la fonction `table`
3. Représenter les fréquences de 0 et de 1 par un diagramme en barres (fonction `barplot`). Représenter par un double diagramme en barre, les fréquences empiriques de 0 et de 1 en bleu et les probabilités théoriques $(1 - p)$ et p en rouge.
4. Utiliser votre échantillon pour simuler n parties d'un jeu de pile ou face où la probabilité de gagner 1 euro est p, la probabilité de perdre 1 euro est $1-p$. Calculer les valeurs successives de la fortune d'un joueur dont la fortune initiale est nulle (fonction `cumsum`). Représenter graphiquement ces valeurs (fonction `plot`).
5. Calculer les valeurs successives de la moyenne empirique des i premières valeurs de l'échantillon initial, pour i allant de 1 à n. Représenter graphiquement ces valeurs en bleu et superposer sur le même graphique la droite horizontale d'ordonnée p en rouge (fonction `abline`).

Exercice 5. On considère des algorithmes censés simuler des lancers de dé, bien que certains trichent... Pour chacun de ces algorithmes :

1. Vérifier que la valeur prise par D appartient à $\{1, \ldots, 6\}$.
2. Déterminer la loi de D.
3. Implémenter l'algorithme.
4. Tirer un échantillon de taille 10000 de D, calculer les fréquences empiriques.
5. Représenter graphiquement un diagramme en bâtons avec les fréquences empiriques en bleu et les fréquences théoriques en rouge.

Dans ces algorithmes, Random désigne un réel de loi uniforme sur $[0, 1]$, Floor désigne la partie entière, Round l'entier le plus proche, Mod le modulo.

- $X \longleftarrow$ Floor(Random $*$ 6)+1

- $X \longleftarrow$ Round(Random $* 5$)$+1$

- $X \longleftarrow$ Floor(Random $* 10$)
  $X \longleftarrow X$ Mod $6 + 1$

- $X \longleftarrow$ Floor(Random $* 12$)
  $X \longleftarrow X$ Mod $6 + 1$

- $U \longleftarrow$ Random
  $X \longleftarrow$ Floor($U * U * 6$)$+1$

- $U \longleftarrow$ Random + Random
  $X \longleftarrow$ Floor($U * 3$)$+1$

- Répéter
  $X \longleftarrow$ Floor(Random $* 10$) $+1$
  Jusqu'à $(X \leqslant 6)$

Exercice 6. Les tailles des échantillons à simuler sont fixées à 10000.

1. Simuler un echantillon de lancers d'un dé. Pour chaque entier i entre 1 et n, calculer la fréquence empirique des entiers supérieurs ou égaux à 3 parmi les i premiers. Représenter graphiquement ces valeurs en bleu et superposer sur le même graphique la droite horizontale d'ordonnée égale à la probabilité théorique en rouge. Calculer les valeurs successives de la moyenne empirique des i premières valeurs de l'échantillon initial, pour i allant de 1 à n. Représenter graphiquement ces valeurs en bleu et superposer sur le même graphique la droite horizontale d'ordonnée égale à l'espérance théorique en rouge.
2. Soit U une variable aléatoire de loi uniforme sur $[0, 1]$. On note N la partie entière de $1/U$. Montrer que pour tout entier n supérieur ou égal à 1, la probabilité que N prenne la valeur n est $1/n-1/(n+1)$. Simuler un échantillon de la loi uniforme sur $[0, 1]$, et en déduire un échantillon de la loi de N (fonction `floor`). Calculer les fréquences empiriques des entiers de 1 à 10. Représenter par un double diagramme en barre, les fréquences empiriques des entiers de 1 à 10 en bleu, et les probabilités théoriques en rouge.
3. Simuler un echantillon de la loi de Poisson de paramètre 3. Calculer les fréquences empiriques des entiers de 1 à 10. Représenter par un double diagramme en barre,

les fréquences empiriques des entiers de 1 à 10 en bleu, et les probabilités théoriques en rouge.

4. Pour chaque entier i entre 1 et n, calculer la fréquence empirique des entiers supérieurs ou égaux à 3 parmi les i premiers éléments de l'échantillon de la question précédente. Représenter graphiquement ces valeurs en bleu et superposer sur le même graphique la droite horizontale d'ordonnée égale à la probabilité théorique en rouge.
5. Simuler un échantillon x de la loi $\mathcal{N}(0,1)$. Pour $(a = 1, b = 2)$, $(a = -2, b = 2)$, $(a = -\infty, b = 2.576)$, $(a = -2.576, b = 2.576)$: calculer la fréquence empirique de l'intervalle $[a, b]$ parmi les i premiers éléments de l'échantillon. Représenter graphiquement ces valeurs en bleu et superposer sur le même graphique la droite horizontale d'ordonnée égale à la probabilité théorique en rouge.

Exercice 7.

1. Représenter sur le même graphique les densités des lois normales
$$\mathcal{N}(0,1)\ , \quad \mathcal{N}(0,10)\ , \quad \mathcal{N}(0,0.1)\ .$$
(fonctions `curve` et `dnorm`).
2. Simuler un échantillon x de taille 1000 de la loi $\mathcal{N}(0,1)$ (fonction `rnorm`).
3. Représenter un histogramme de cet échantillon en bleu. Superposer sur le même graphique une estimation de la densié (fonction `density`) en vert et la densité exacte de la loi $\mathcal{N}(0,1)$ en rouge.
4. Représenter la fonction de répartition empirique de l'échantillon en bleu
   (a) En appliquant la définition : fonction `sort` pour ranger par ordre croissant, `type="step"` dans `plot` pour une représentation en escalier).
   (b) En utilisant la fonction `ecdf`.
5. Superposer sur le même graphique la fonction de répartition de la loi $\mathcal{N}(0,1)$ (fonctions `pnorm` et `curve`) en rouge.
6. Calculer les valeurs successives de la moyenne empirique des i premières valeurs de l'échantillon initial, pour i allant de 1 à n. Représenter graphiquement ces valeurs en bleu et superposer sur le même graphique la droite horizontale d'ordonnée 0 en rouge (fonction `abline`).
7. Séparer l'échantillon initial x en deux échantillons de taille 500, et les ajouter pour en déduire un nouvel échantillon y (fonctions `matrix` et `colSums`). Représenter un histogramme de cet échantillon y en bleu. Superposer sur le même graphique la densité de la loi normale de moyenne nulle et de variance 2, en rouge. Ouvrir une nouvelle fenêtre graphique. Représenter la fonction de répartition empirique de l'échantillon y en bleu. Superposer sur le même graphique la fonction de répartition de la loi normale de moyenne nulle et de variance 2, en rouge.

8. Tirer 1000 échantillons de taille 12 de la loi uniforme sur $[-0.5, 0.5]$ et calculer les 1000 sommes de ces échantillons : soit s l'échantillon de taille 1000 ainsi obtenu (fonctions `matrix` et `rowSums`). Représenter un histogramme de cet échantillon s en bleu. Superposer sur le même graphique la densité de la loi normale de moyenne nulle et de variance 1, en rouge. Représenter sur une nouvelle fenêtre graphique les points dont les abscisses sont les valeurs de la fonction quantile de la loi $\mathcal{N}(0,1)$ (fonction `qnorm`) aux points $0.001, \ldots, 0.999$, et en ordonnée les valeurs de s triées par ordre croissant, sauf la dernière. Comparer votre graphique avec celui obtenu par `qqnorm(s)`.

Exercice 8. Pour les lois de probabilité P suivantes :

- Lois binomiales $\mathcal{B}(4, 0.5)$, $\mathcal{B}(4, 0.2)$, $\mathcal{B}(4, 0.8)$.
- Lois sur $\{0, \ldots, 4\}$ définies par les probabilités suivantes :

0	1	2	3	4
0.2	0.2	0.2	0.2	0.2
0.3	0.3	0.3	0.05	0.05
0.6	0.1	0.1	0.1	0.1
0.9	0.025	0.025	0.025	0.025

- Lois uniformes $\mathcal{U}(0,1)$, $\mathcal{U}(0,100)$.
- Lois exponentielles $\mathcal{E}(1)$, $\mathcal{E}(0.1)$.
- Lois normales $\mathcal{N}(0,1)$, $\mathcal{N}(10,100)$.
- Lois Gamma $\mathcal{G}(10,1)$, $\mathcal{G}(100,1)$.

1. Simuler 1000 échantillons de taille 100 de la loi P.
2. On note x^* l'échantillon des 1000 moyennes empiriques, centrées et réduites. Représenter un histogramme de l'échan-tillon x^* et superposer sur le même graphique la densité de la loi normale $\mathcal{N}(0,1)$.
3. Représenter la fonction de répartition empirique de l'échantillon x^* et superposer sur le même graphique la fonction de répartition de la loi normale $\mathcal{N}(0,1)$.
4. Idem pour l'échantillon des 1000 variances empiriques, centrées et réduites.
5. Idem pour l'échantillon des 1000 écarts-types empiriques, centrés et réduits.
6. Idem pour l'échantillon des 1000 médianes empiriques, centrées et réduites.

Exercice 9. Les tailles des échantillons à simuler sont fixées à 10000.

1. Soit U une variable aléatoire de loi uniforme sur $[0,1]$. La variable aléatoire $S = \sqrt{U}$ a pour fonction de répartition :

$$F_S(y) = \begin{cases} 0 & \text{si} \quad y \leqslant 0 \\ y^2 & \text{si} \quad 0 < y \leqslant 1 \\ 1 & \text{si} \quad y > 1 \ . \end{cases}$$

Elle a pour densité :

$$f_S(y) = \begin{cases} 2y & \text{si } 0 < y < 1 \\ 0 & \text{sinon .} \end{cases}$$

Simuler un échantillon de la loi uniforme sur $[0, 1]$, en déduire un échantillon de la loi de S. Représenter la fonction de répartition empirique de l'échantillon en bleu. Superposer sur le même graphique la fonction de répartition de S en rouge. Représenter un histogramme de l'échantillon en bleu. L'histogramme aura 100 classes, de même amplitude entre 0 et 1. Superposer sur le même graphique la densité de S en rouge.

2. Soit U une variable aléatoire de loi uniforme sur $[-1, 1]$. La variable aléatoire $Z = U^2$ a pour fonction de répartition

$$F_Z(y) = \begin{cases} 0 & \text{si } \ y \leqslant 0 \\ \sqrt{y} & \text{si } \ 0 < y \leqslant 1 \\ 1 & \text{si } \ y > 1 \ . \end{cases}$$

Elle a pour densité

$$f_Z(y) = \begin{cases} \frac{1}{2\sqrt{y}} & \text{si } 0 < y \leqslant 1 \\ 0 & \text{sinon .} \end{cases}$$

Simuler un échantillon de la loi uniforme sur $[-1, 1]$, et en déduire un échantillon de la loi de Z. Représenter la fonction de répartition empirique de l'échantillon en bleu. Superposer sur le même graphique la fonction de répartition de Z en rouge. Représenter un histogramme de l'échantillon en bleu. L'histogramme aura 100 classes, de même amplitude entre 0 et 1. Superposer sur le même graphique la densité de Z en rouge.

3. Soit U une variable aléatoire de loi uniforme sur [0,1]. La variable aléatoire $V = 1/U$ a pour fonction de répartition :

$$F_V(y) = \begin{cases} 1 - 1/y & \text{si } y > 1 \\ 0 & \text{sinon .} \end{cases}$$

Elle a pour densité :

$$f_V(y) = \begin{cases} 1/y^2 & \text{si } y > 11 \\ 0 & \text{sinon .} \end{cases}$$

Simuler un échantillon de la loi uniforme sur $[0, 1]$, et en déduire un échantillon de la loi de V. Choisir un réel L pour les représentations qui suivent (essayer plusieurs valeurs). Représenter la fonction de répartition empirique de l'échantillon en bleu, pour les abscisses comprises entre 0 et L. Superposer sur le même graphique la fonction de répartition de V en rouge. Représenter un histogramme de l'échantillon en bleu. L'histogramme aura 100 classes de même amplitude, entre 0 et L. Superposer sur le même graphique la densité de V en rouge.

4. Soit U une variable aléatoire de loi uniforme sur $[-\pi/2, \pi/2]$. La variable aléatoire $C = \tan(U)$ a pour fonction de répartition :

$$F_C(y) = \frac{1}{\pi} \arctan(y) + \frac{1}{2} .$$

Elle a pour densité :

$$f_C(y) = \frac{1}{\pi(1 + y2)} .$$

Simuler un échantillon de la loi uniforme sur $[-\pi/2, \pi/2]$, et en déduire un échantillon de la loi de C. Choisir un réel L pour les représentations qui suivent (essayer plusieurs valeurs). Représenter la fonction de répartition empirique de l'échantillon en bleu, pour les abscisses comprises entre $-L$ et L. Superposer sur le même graphique la fonction de répartition de C en rouge. Représenter un histogramme de l'échantillon en bleu. L'histogramme aura 100 classes, de même amplitude entre 0 et L. Superposer sur le même graphique la densité de V en rouge.

Exercice 10. Pour les lois de probabilité P suivantes :

- Lois uniformes $\mathcal{U}(0, 1)$, $\mathcal{U}(0, 100)$.
- Lois exponentielles $\mathcal{E}(1)$, $\mathcal{E}(0.1)$.
- Lois normales $\mathcal{N}(0, 1)$, $\mathcal{N}(10, 100)$.
- Lois Gamma $\mathcal{G}(10, 1)$, $\mathcal{G}(100, 1)$.
- Lois de Student $\mathcal{T}(1)$, $\mathcal{T}(100)$.
- Lois de Fisher $\mathcal{F}(2, 2)$, $\mathcal{F}(20, 20)$.

1. Simuler un échantillon x de taille 1000 de la loi P.
2. Pour $i = 0, \ldots, 20$, on note :

$$a_i = \min\{x\} + \frac{i}{20}(\max\{x\} - \min\{x\}) .$$

Calculer les fréquences empiriques des 20 classes
$[a_{i-1}, a_i]$ $(i = 1, \ldots, 20)$. Superposer sur un même graphique un histogramme de ces fréquences empiriques et la densité de la loi P.

3. Idem si les a_i sont les statistiques d'ordre d'un échantillon de taille 21 de la loi uniforme $\mathcal{U}(\min\{x\}, \max\{x\})$.
4. Idem si les a_i sont les statistiques d'ordre d'un échantillon de taille 21 de la loi P.

Exercice 11. Les tailles des échantillons à simuler sont fixées à 10000.

1. Simuler un échantillon x de la loi normale $\mathcal{N}(1, 4)$ et un échantillon y de la loi normale $\mathcal{N}(0, 1)$. Calculer $z = x + 2y$.
   (a) Représenter dans le plan les couples de points (xi,zi).

(b) Représenter un histogramme des valeurs de z. Superposer sur le même graphique la densité de la loi normale $\mathcal{N}(1, 8)$. Représenter la fonction de répartition empirique de z. Superposer sur le même graphique la fonction de répartition de la loi normale $\mathcal{N}(1, 8)$.

(c) Pour i allant de 1 à 10000, calculer la covariance empirique des i premières valeurs des échantillons x et z. Représenter graphiquement ces valeurs. Superposer sur le même graphique la droite horizontale dont l'ordonnée est égale à la covariance théorique (4).

2. Simuler 3 échantillons x_1, x_2, x_3, de lois respectives

$$\mathcal{N}(4, 0.01) \ , \quad \mathcal{N}(1, 0.01) \ , \quad \mathcal{N}(5, 0.01) \ .$$

Calculer les échantillons $z = x_1 - x_2$ et $t = x_1 + x_3$.

(a) Représenter dans le plan les couples de points dont les abscisses sont les valeur de z, les ordonnées les valeurs de t.

(b) Représenter un histogramme des valeurs de z. Superposer sur le même graphique la densité de la loi normale $\mathrm{N}(3, 0.02)$. Représenter la fonction de répartition empirique de z. Superposer sur le même graphique la fonction de répartition de la loi normale $\mathcal{N}(3, 0.02)$.

(c) Représenter un histogramme des valeurs de t. Superposer sur le même graphique la densité de la loi normale $\mathcal{N}(9, 0.02)$. Représenter la fonction de répartition empirique de t. Superposer sur le même graphique la fonction de répartition de la loi normale $\mathcal{N}(9, 0.02)$.

(d) Pour i allant de 1 à 10000, calculer la covariance empirique des i premières valeurs des échantillons z et t. Représenter graphiquement ces valeurs. Superposer sur le même graphique la droite horizontale dont l'ordonnée est égale à la covariance théorique (0.01).

3. Simuler 2 échantillons de la loi normale $\mathcal{N}(0, 1)$: x et y. Calculer l'échantillon $z = x^2 + y^2$. Représenter une estimation de la densité de z. Superposer sur le même graphique la densité de la loi du chi-deux à 2 degrés de liberté. Représenter la fonction de répartition empirique de z. Superposer sur le même graphique la fonction de répartition de la loi du chi-deux à 2 degrés de liberté.

4. Reprendre la question précédente avec 4 échantillons de la loi $\mathcal{N}(0, 1)$ et la loi du chi-deux à 4 degrés de liberté.

5. Calculer la moyenne empirique $m = (x_1 + \cdots + x_n)/n$. Calculer l'échantillon $z = y - nm^2$. Représenter un histogramme des valeurs de z. Superposer sur le même graphique la densité de la loi du chi-deux à $(n - 1)$ degrés de liberté. Représenter la fonction de répartition empirique de z. Superposer sur le même graphique la fonction de répartition de la loi du chi-deux à $(n - 1)$ degrés de liberté.

Exercice 12. Pour les lois de probabilité P suivantes :

- Lois binomiales $\mathcal{B}(4, 0.5)$, $\mathcal{B}(4, 0.2)$, $\mathcal{B}(4, 0.8)$.
- Lois sur $\{0, \ldots, 4\}$ définies par les probabilités suivantes :

0	1	2	3	4
0.2	0.2	0.2	0.2	0.2
0.3	0.3	0.3	0.05	0.05
0.6	0.1	0.1	0.1	0.1
0.9	0.025	0.025	0.025	0.025

1. Simuler 100 échantillons de taille 1000 de la loi P. Pour chacun des 100 échantillons, calculer la distance du chi-deux de sa distribution empirique par rapport à la distribution théorique P. Soit x l'échantillon de taille 100 des valeurs prises par la distance du chi-deux, multipliées par 1000.
2. Superposer sur un même graphique un histogramme de l'échantillon x, et la densité de la loi de chi-deux à 4 degrés de liberté.
3. Superposer sur un même graphique la fonction de répartition empirique de l'échantillon x et la fonction de répartition $F_{\mathcal{X}^2(4)}$ de la loi de chi-deux à 4 degrés de liberté.
4. Ajustement par quantiles : former le vecteur y, des centiles de la loi de chi-deux : $Q_{\mathcal{X}^2(4)}(i/100)$, $i = 1, \ldots, 99$. On note $(x_{(i)})_{i=1,\ldots,n}$ les statistiques d'ordre de x (valeurs rangées par ordre croissant). Représenter sur un même graphique le nuage des points $(x_{(i)}, y_i)$ et la première bissectrice.

Exercice 13. Pour les lois de probabilité P suivantes :

- Lois binomiales $\mathcal{B}(30, 0.5)$, $\mathcal{B}(30, 0.1)$, $\mathcal{B}(100, 0.1)$.
- Lois de Poisson $\mathcal{P}(30)$, $\mathcal{P}(100)$.
- Lois de Student $\mathcal{T}(10)$, $\mathcal{T}(30)$, $\mathcal{T}(100)$.
- Lois Gamma $\mathcal{G}(10, 1)$, $\mathcal{G}(30, 1)$, $\mathcal{G}(100, 1)$.

1. Simuler un échantillon de taille 100 de la loi P. Soit y l'échantillon formé des 99 premières statistiques d'ordre des valeurs simulées. Soit $x = (Q_{\mathcal{N}(0,1)}(i/100))$, $i = 1, \ldots, 99$, le vecteur des centiles de la loi $\mathcal{N}(0, 1)$.
2. Calculer $\overline{x}$, s_x^2, $\overline{y}$, s_y^2, c_{xy}, r_{xy}.
3. Calculer les coefficients $\widehat{a}$ et $\widehat{b}$ de la droite de régression linéaire de y sur x. Représenter le nuage des points (x_i, y_i) et la droite de régression linéaire sur le même graphique.
4. Comparer les valeurs de $\widehat{b}$ et $\widehat{a}$ à l'espérance et à l'écart-type de la loi P.
5. Représenter sur le même graphique un histogramme de l'échantillon y et la densité de la loi normale de même espérance et de même variance que la loi P.

Exercice 14. Choisir deux réels c et d tels que $c < d$. Simuler un échantillon de taille 100 de la loi uniforme $\mathcal{U}(c, d)$. Soit y l'échantillon des statistiques d'ordre des valeurs simulées et $x = (i/100)\,,\ i = 1, \ldots, 100$.

1. Calculer $\overline{x}$, s_x^2, $\overline{y}$, s_y^2, c_{xy}, r_{xy}.
2. Calculer les coefficients $\widehat{a}$ et $\widehat{b}$ de la droite de régression linéaire de y sur x. Représenter le nuage des points (x_i, y_i) et la droite de régression linéaire sur le même graphique.
3. Comparer les valeurs de $\widehat{b}$ et $\widehat{a}$ à c et $d-c$.

Exercice 15. Choisir deux réels μ et $\sigma > 0$. Simuler un échantillon de taille 100 de la loi normale $\mathcal{N}(\mu, \sigma^2)$. Soit y l'échantillon des 99 premières statistiques d'ordre des valeurs simulées. Soit $x = (Q_{\mathcal{N}(0,1)}(i/100))\,,\ i = 1, \ldots, 99$, le vecteur des centiles de la loi $\mathcal{N}(0, 1)$.

1. Calculer $\overline{x}$, s_x^2, $\overline{y}$, s_y^2, c_{xy}, r_{xy}.
2. Calculer les coefficients $\widehat{a}$ et $\widehat{b}$ de la droite de régression linéaire de y sur x. Représenter le nuage des points (x_i, y_i) et la droite de régression linéaire sur le même graphique.
3. Comparer les valeurs de $\widehat{b}$ et $\widehat{a}$ à μ et σ.

Exercice 16. Choisir deux réels $c > 0$ et $\lambda > 0$. Simuler un échantillon e de taille 100 de la loi de Weibull $\mathcal{W}(c, \lambda)$. Soit $y = (\log(e_{(i)})\,,\ i = 1, \ldots, 99$, où les $e_{(i)}$ sont les 99 premières statistiques d'ordre des valeurs simulées. Soit $x = (\log(-\log(1-i/100)))\,,\ i = 1, \ldots, 99$.

1. Calculer $\overline{x}$, s_x^2, $\overline{y}$, s_y^2, c_{xy}, r_{xy}.
2. Calculer les coefficients $\widehat{a}$ et $\widehat{b}$ de la droite de régression linéaire de y sur x. Représenter le nuage des points (x_i, y_i) et la droite de régression linéaire sur le même graphique.
3. Comparer les valeurs de $\widehat{a}$ et $\widehat{b}$ à $(1/c)$ et $(1/c)\log(1/\lambda)$.

Exercice 17. Soit $(X_1, \ldots, X_n)$ un échantillon de la loi uniforme sur $[0, \theta]$ où θ est un paramètre inconnu. On note $(X_{(1)}, \ldots, X_{(n)})$ les statistiques d'ordre de l'échantillon (valeurs de l'échantillon rangées par ordre croissant. On considère les estimateurs

convergents suivants du paramètre θ ($\lfloor\ \rfloor$ désigne la partie entière).

$$\begin{aligned}
T_1 &= \frac{2}{n}(X_1+\cdots+X_n)\\
T_2 &= \frac{3}{n}(X_1^2+\cdots+X_n^2)^{1/2}\\
T_3 &= \frac{4}{n}(X_1^3+\cdots+X_n^3)^{1/3}\\
T_4 &= \frac{3}{2n}(X_1^{1/2}+\cdots+X_n^{1/2})^2\\
T_5 &= \frac{1}{2n}(X_1^{-1/2}+\ldots+X_n^{-1/2})^{-2}\\
T_6 &= \mathrm{e}^1(X_1\ldots X_n)^{1/n}\\
T_7 &= 4X_{\lfloor n/4\rfloor}\\
T_8 &= 2X_{\lfloor n/2\rfloor}\\
T_9 &= \frac{3}{2}X_{\lfloor 2n/3\rfloor}\\
T_{10} &= \frac{4}{3}X_{\lfloor 3n/4\rfloor}\\
T_{11} &= \max\{X_1,\ldots,X_n\}\\
T_{12} &= \frac{n+1}{n}\max\{X_1,\ldots,X_n\}
\end{aligned}$$

1. Choisir une valeur de θ et simuler 1000 échantillons de taille 100 de la loi uniforme sur $[0,\theta]$. Calculer pour chacun de ces échantillons la valeur prise par les 12 estimateurs.
2. Calculer la moyenne empirique, et la variance empirique des 12 échantillons de taille 1000 ainsi obtenus. En déduire une estimation du biais et de l'erreur quadratique de chacun des 12 estimateurs.
3. À partir des échantillons de la question 1, représenter sur un même graphique les diagrammes en boîte (boxplot) des 12 estimateurs. Superposer sur le même graphique la vraie valeur du paramètre, en rouge.
4. À partir des échantillons de la question 1, représenter des histogrammes pour les 12 estimateurs, avec les mêmes échelles. Représenter sur chaque graphique la valeur exacte de θ par une ligne verticale rouge. Représenter par deux lignes verticales jaunes les bornes de l'intervalle de confiance de niveau 0.95 pour l'espérance de l'estimateur.

Exercice 18. Soit $(X_1,\ldots,X_n)$ un échantillon de la loi exponentielle $\mathcal{E}(\lambda)$, où λ est un paramètre inconnu. On considère les estimateurs suivants du paramètre λ. On admettra que ce sont tous des estimateurs convergents.

- $T_{1,n}=\left(\frac{1}{n}(X_1+\cdots+X_n)\right)^{-1}$

- $T_{2,n} = \left(\frac{1}{2n}(X_1^2 + \cdots + X_n^2)\right)^{-1/2}$
- $T_{3,n} = \frac{\mathrm{e}^{-X_1} + \cdots + \mathrm{e}^{-X_n}}{n - \mathrm{e}^{-X_1} - \cdots - \mathrm{e}^{-X_n}}$
- $T_{4,n} = \frac{\log(2)}{X_{(\lceil \frac{n}{2} \rceil)}}$
- $T_{5,n} = \frac{\log(4/3)}{X_{(\lceil \frac{n}{4} \rceil)}}$
- $T_{6,n} = \frac{\log(4)}{X_{(\lceil \frac{3n}{4} \rceil)}}$

(Pour $u \in]0,1[$, $\lceil \nu \rceil$ désigne l'entier i tel que $i-1 < \nu \leqslant i$, et $X_{(i)}$ est la i-ième statistique d'ordre de l'échantillon.)

1. Choisir une valeur de λ et simuler 1000 échantillons de taille 100 de la loi $\mathcal{E}(\lambda)$. Calculer pour chacun de ces échantillons la valeur prise par les 6 estimateurs. Calculer la moyenne empirique, et la variance empirique des 6 échantillons de taille 1000 ainsi obtenus. En déduire une estimation du biais et de l'erreur quadratique de chacun des 6 estimateurs.
2. À partir des échantillons de la question 1, représenter sur un même graphique les diagrammes en boîte (boxplot) des 6 estimateurs. Superposer sur le même graphique la vraie valeur du paramètre, en rouge.
3. À partir des échantillons de la question 1, représenter des histogrammes pour les 6 estimateurs, et proposer des intervalles de dispersion de niveau 0.9.
4. Proposer un classement des 6 estimateurs.

Exercice 19. Soit X une variable aléatoire de loi de Poisson $\mathcal{P}(\lambda)$ et $k \geqslant 1$ un entier. On admettra que pour tout $k \geqslant 1$:

$$\mathbb{E}[X(X-1)\cdots(X-k+1)] = \lambda^k \, .$$

Soit $(X_1, \ldots, X_n)$ un échantillon de la loi $\mathcal{P}(\lambda)$. On pose :

$$T_{k,n} = \frac{1}{n}\sum_{i=1}^{n} X_i(X_i-1)\cdots(X_i-k+1) \, .$$

La statistique $(T_{k,n}^{1/k})$ est donc un estimateur convergent de λ.

1. Choisir une valeur de λ. Simuler 1000 échantillons de taille 100 de la loi $\mathcal{P}(\lambda)$. Pour chacun des 1000 échantillons, calculer la valeur prise par les estimateurs $(T_{k,n}^{1/k})$, pour $k = 1, 2, 3, 4$. On obtient ainsi un échantillon de taille 1000 pour chacun des 4 estimateurs.
2. Pour chacun des 4 échantillons de la question précédente, représenter un histogramme, calculer la moyenne empirique et la variance empirique. En déduire une estimation du biais et de l'erreur quadratique des 4 estimateurs par rapport à λ.

3. Proposer un classement des 4 estimateurs.

Exercice 20. Pour chacune des lois P suivantes :

- Lois binomiales $\mathcal{B}(10, 0.5)$, $\mathcal{B}(10, 0.1)$.
- Lois géométriques $\mathcal{G}(0.1)$, $\mathcal{G}(0.9)$.
- Lois de Poisson $\mathcal{P}(0.1)$, $\mathcal{P}(10)$.
- Lois uniformes $\mathcal{U}(0, 0.1)$, $\mathcal{U}(0, 10)$.
- Lois exponentielles $\mathcal{E}(0.1)$, $\mathcal{E}(10)$.
- Lois normales $\mathcal{N}(0, 0.1)$, $\mathcal{N}(0, 100)$.

1. Donner la valeur de l'espérance μ, de la variance σ^2 et de l'écart-type σ de la loi P.
2. Simuler 1000 échantillons de taille 20 de la loi P, et calculer pour chacun la valeur prise par la moyenne empirique $\overline{X}$, la variance empirique S^2, la variance empirique non biaisée V, ainsi que par $\sqrt{S^2}$ et $\sqrt{V}$. On obtient ainsi 5 échantillons de taille 1000 de ces estimateurs. Utiliser ces 5 échantillons pour estimer le biais et l'erreur quadratique moyenne de $\overline{X}$ par rapport à μ, de S^2 et V par rapport à σ^2, et de $\sqrt{S^2}$ et $\sqrt{V}$ par rapport à σ.

Exercice 21. Le but de l'exercice est de comparer les estimateurs des paramètres μ et σ^2 de la loi normale $\mathcal{N}(\mu, \sigma^2)$, obtenus par la moyenne et la variance empirique, et par régression au sens des moindres carrés.

1. Choisir deux valeurs pour μ et σ^2. Simuler 1000 échantillons de taille 100 de la loi $\mathcal{N}(\mu, \sigma^2)$.
2. Pour chacun des 1000 échantillons, déterminer la moyenne empirique, et la variance empirique non biaisée. On obtient ainsi un échantillon de taille 1000 pour chacun des 2 estimateurs : représenter un histogramme.
3. Pour chacun des 1000 échantillons, calculer la série des statistiques d'ordre et déterminer les valeurs de μ et σ^2 déduites de la régression au sens des moindres carrés de ces statistiques d'ordre. Pour ces 1000 nouvelles estimations des deux paramètres, représenter des histogrammes, calculer les moyennes et les variances empiriques. En déduire une estimation du biais et de l'erreur quadratique des 2 estimateurs par rapport à μ et σ^2 respectivement.
4. Laquelle des deux méthodes conduit aux meilleurs estimateurs ?

Exercice 22. Pour chacune des lois P suivantes :

- loi binomiale $\mathcal{B}(15, 0.5)$, $\mathcal{B}(15, 0.9)$, $\mathcal{B}(150, 0.05)$.
- loi de Poisson $\mathcal{P}(1)$, $\mathcal{P}(5)$.
- loi géométrique $\mathcal{G}(0.5)$, $\mathcal{G}(0.2)$. $\mathcal{G}(0.1)$.
- loi exponentielle $\mathcal{E}(1)$, $\mathcal{E}(0.5)$, $\mathcal{E}(0.1)$.
- loi normale $\mathcal{N}(1, 9)$, $\mathcal{N}(1, 25)$, $\mathcal{N}(-10, 100)$.
- loi gamma $\mathcal{G}(10, 0.04)$, $\mathcal{G}(10, 0.1)$, $\mathcal{G}(5, 0.02)$.
- loi du chi-deux $\mathcal{X}^2(1)$, $\mathcal{X}^2(3)$.

- loi de Student $\mathcal{T}(1)$, $\mathcal{T}(10)$.
- loi de Fisher $\mathcal{F}(1,1)$, $\mathcal{F}(10,1)$, $\mathcal{F}(1,10)$.

1. Représenter graphiquement le diagramme en bâtons (lois discrètes) ou la densité (lois continues).
2. Une statistique de test T prend la valeur 10. Quelle décision prenez-vous au seuil $\alpha = 0.05$ pour un test unilatéral à droite, si la loi de T sous $\mathcal{H}_0$ est P.
3. Pour le même test, calculer la p-valeur correspondant à $T = 10$.
4. Reprendre les questions précédentes pour le test bilatéral de seuil $\alpha = 0.05$.

Exercice 23. On considère les algorithmes suivants :

a) $T \longleftarrow 0$
Répéter n fois
    Si $(0.4 \leqslant$ Random $\leqslant 0.9)$ alors $T \longleftarrow T + 1$
    finSi
finRépéter

b) $T \longleftarrow 0$
Répéter n fois
    $X \longleftarrow$ Random ; $Y \longleftarrow$ Random
    Si $(X < Y)$ alors $T \longleftarrow T + 1$
    finSi
finRépéter

c) $T \longleftarrow 0$
Répéter n fois
    $X \longleftarrow$ Random ; $Y \longleftarrow$ Random
    Si $(X^2 + Y^2 < 1)$ alors $T \longleftarrow T + 1$
    finSi
finRépéter

d) $T \longleftarrow 0$
Répéter n fois
    $X \longleftarrow$ Random ; $Y \longleftarrow$ Random ; $Z \longleftarrow$ Random
    Si $(X < Y$ et $X < Z)$ alors $T \longleftarrow T + 1$
    finSi
finRépéter

L'hypothèse nulle $\mathcal{H}_0$ est que les appels successifs de Random sont des variables aléatoires indépendantes, de loi $\mathcal{U}(0,1)$.

1. Montrer que sous l'hypothèse $\mathcal{H}_0$, T suit une loi binomiale $\mathcal{B}(n,p)$, et déterminer la valeur de p pour chacun des algorithmes.
2. Pour chacun des algorithmes, écrire une fonction qui prend en entrée une valeur de n, qui exécute n fois l'algorithme, et qui retourne en sortie la p-valeur du test bilatéral de proportion pour la valeur de p (fonction `prop.test`).
3. Pour $n = 100$, exécuter 1000 fois l'algorithme et représenter un histogramme des p-valeurs obtenues.

Exercice 24. On considère l'algorithme suivant.

$T \longleftarrow 0$
Répéter n fois
    TantQue Random $> p$
        $T \longleftarrow T + 1$
    finTantQue
finRépéter

L'hypothèse $\mathcal{H}_0$ est que les appels successifs de Random sont des variables aléatoires indépendantes de loi $\mathcal{U}(0,1)$. Sous cette hypothèse, on montrera, ou on admettra qu'en sortie de l'algorithme, T suit la loi binomiale négative $\mathcal{BN}(n,p)$. Pour $n = 5$, 10, 100 et $p = 0.1$, 0.5, 0.9 :

1. Calculer un intervalle de dispersion symétrique pour la loi $\mathcal{BN}(n,p)$, de niveau 0.95.
2. En déduire un test bilatéral de seuil 0.05 pour $\mathcal{H}_0$.
3. Exécuter l'algorithme 1000 fois. Représenter sur le même graphique le diagramme en bâtons des 1000 valeurs de T obtenues, et celui de la loi $\mathcal{BN}(n,p)$.
4. Appliquer le test aux 1000 valeurs et compter le nombre de rejets. En déduire une estimation de la probabilité de rejet.
5. Pour chacune des 1000 valeurs, calculer la p-valeur relative à la loi $\mathcal{BN}(n,p)$.

Exercice 25. Soit $(X_1, \ldots, X_n)$ un échantillon de la loi P. On considère l'hypothèse $\mathcal{H}_0$: $P = \mathcal{U}(0,1)$. Pour tester la valeur du quantile de P en u, on utilise la statistique :

$$T_u = \sum_{i=1}^{n} \mathbb{I}_{(-\infty,u]}(X_i) \ ,$$

qui suit la loi binomiale $\mathcal{B}(n,u)$ sous $\mathcal{H}_0$.

Pour $u = 0.25$, 0.5, 0.75 et les lois P suivantes :

- lois uniformes $\mathcal{U}(0,0.9)$, $\mathcal{U}(0,1)$, $\mathcal{U}(0,1.1)$.

- lois bêta $\mathcal{B}(0.9, 1.1)$, $\mathcal{B}(1.1, 1.1)$, $\mathcal{U}(1.1, 0.9)$.

1. Simuler 1000 échantillons de taille 100 de la loi P et calculer pour chacun la valeur prise par T_u.
2. Appliquer le test au seuil $\alpha = 0.05$ et compter le nombre de rejets.
3. Représenter sur un même graphique le diagramme en bâtons des 1000 valeurs de T_u et celui de la loi binomiale $\mathcal{B}(100, u)$.

Exercice 26. Soit $(X_1, \ldots, X_n)$ un échantillon de la loi P_0 et $(Y_1, \ldots, Y_n)$ un échantillon de la loi P_1, indépendant du précédent. On considère l'hypothèse $\mathcal{H}_0 \ : \ P_0 = P_1$. Le test des signes est basé sur la statistique :

$$T = \sum_{i=1}^{n} \mathbb{I}_{X_i < Y_i} \ ,$$

qui suit la loi binomiale $\mathcal{B}(n, 1/2)$ sous $\mathcal{H}_0$. On suppose que P_0 est la loi uniforme $\mathcal{U}(0, 1)$ et on considère les lois P_1 suivantes :

- lois uniformes $\mathcal{U}(0, 0.9)$, $\mathcal{U}(0, 1)$, $\mathcal{U}(0, 1.1)$.
- lois bêta $\mathcal{B}(0.9, 1.1)$, $\mathcal{B}(1.1, 1.1)$, $\mathcal{B}(1.1, 0.9)$.

1. Simuler 1000 échantillons de taille 100 de la loi P_0 et de la loi P_1. Calculer pour chacun la valeur prise par T.
2. Appliquer le test au seuil $\alpha = 0.05$ et compter le nombre de rejets.
3. Représenter sur un même graphique le diagramme en bâtons des 1000 valeurs de T et celui de la loi binomiale $\mathcal{B}(100, 0.5)$.
4. Reprendre les questions 2 et 3 en utilisant :
   (a) le test de Kolmogorov-Smirnov (fonciton `ks.test`)
   (b) le test de Wilcoxon (fonction `wilcox.test`)

Exercice 27. Soit $(X_1, \ldots, X_n)$ un échantillon de taille $n = 30$ de la loi P. On souhaite tester au seuil $\alpha = 0.05$:

$$\mathcal{H}_0 \ : \ P = \mathcal{U}(0, 1) \quad \text{contre} \quad \mathcal{H}_1 \ : \ P = \mathcal{B}(2, 1) \ .$$

(Loi uniforme sur $[0, 1]$ contre loi bêta de parmètres 2 et 1). On envisage pour cela les tests suivants :

a) Test du rapport de vraisemblance. La statistique de test est $T_a = -\sum \log(X_i)$. Elle suit la loi $\mathcal{G}(n, 1)$ sous $\mathcal{H}_0$, la loi $\mathcal{G}(n, 2)$ sous $\mathcal{H}_1$.

b) Test de la médiane. La statistique de test est $T_b = \sum \mathbb{I}_{[0,1/2]}(X_i)$. Elle suit la loi $\mathcal{B}(n, 1/2)$ sous $\mathcal{H}_0$, la loi $\mathcal{B}(n, 1/4)$ sous $\mathcal{H}_1$.

c) Test sur la valeur de la moyenne empirique. La statistique de test est $T_c = \overline{X}$. Sa loi est approximativement normale, de paramètres $(\frac{1}{2}, \frac{1}{12n})$ sous $\mathcal{H}_0$, $(\frac{2}{3}, \frac{1}{18n})$ sous $\mathcal{H}_1$.

d) Test de Kolmogorov-Smirnov.
e) Test du chi-deux. On considérera un découpage de l'intervalle $[0\,,1]$ en 10 classes d'amplitudes égales.

1. Pour les trois premiers tests, calculer la règle de décision, puis la puissance du test.
2. Simuler 1000 échantillons de taille 30 de la loi bêta $\mathcal{B}(2,1)$.
   (a) Appliquer les 5 tests à chacun des 1000 échantillons et compter le nombre de rejets. En déduire une estimation de la puissance de chaque test.
   (b) Proposer un classement des 5 tests, du plus au moins puissant.

Exercice 28. Si on joue à un jeu de hasard pour lequel la probabilité de gagner est strictement inférieure à $1/2$, on est sûr de se ruiner au bout d'un temps plus ou moins long. Si on gagne 1 euro avec probabilité $p < 1/2$, et on perd 1 euro avec probabilité $1-p$, alors partant d'une fortune initiale égale à a, on se ruine en moyenne au bout d'un temps égal à $a/(1-2p)$.

Écrire une fonction qui prend en entrée deux entiers a et n et un réel p strictement compris entre 0 et $1/2$. La fonction simule n trajectoires partant d'une fortune initiale a, d'un jeu pour lequel la probabilité de gagner $+1$ est p et la probabilité de perdre -1 est $1-p$. Pour chaque trajectoire elle calcule l'instant de ruine, à savoir le premier instant pour lequel la fortune atteint 0. Elle trace un diagramme en bâtons pour les valeurs des instants de ruine, et retourne un intervalle de confiance pour son espérance.

Exercice 29. Écrire une fonction qui prend en entrée une chaîne de caractères `chaine` donnant le nom d'une loi, parmi `unif`, `norm`, `gamma`, `weibull`, `beta`, `logis`, `lnorm`, deux paramètres réels a et b et un entier n.

La fonction simule un échantillon de taille n de la loi `chaine` ayant pour paramètres a et b. Elle ouvre une fenêtre graphique qu'elle divise en 2 sous-fenêtres. Elle affiche dans ces 2 fenêtres.

1. Un histogramme des valeurs de l'échantillon avec la densité de la loi.
2. La fonction de répartition empirique avec la fonction de répartition théorique.

Exercice 30. Écrire une fonction qui prend en entrée trois vecteurs $p = (p_i)_{i=1,\dots,d}$, $\mu = (\mu_i)_{i=1,\dots,d}$ et $\sigma = (\sigma_i)_{i=1,\dots,d}$. Le vecteur p est une loi de probabilité sur $\{1,\dots,d\}$. Par exemple :

$$p = (1/6, 1/2, 1/3)\ , \quad \mu = (-2, 0, 3)\ , \quad \sigma = (0.2, 1, 2)\ .$$

La fonction simule un échantillon de taille n de la loi mélange, telle que chaque composante est tirée selon la loi normale $\mathcal{N}(\mu_i, \sigma_i^2$ avec probabilité p_i. Elle ouvre une fenêtre graphique qu'elle divise en 2 sous-fenêtres. Elle affiche dans ces 2 fenêtres.

1. Un histogramme des valeurs de l'échantillon avec la densité de la loi théorique.
2. La fonction de répartition empirique avec la fonction de répartition théorique.

Exercice 31. Écrire une fonction qui prend en entrée 3 réels σ_1, σ_2 et ρ et un entier n. Le réel ρ doit être strictement compris entre -1 et 1. La fonction tire deux échantillons u_1 et u_2, de taille n, de la loi normale $\mathcal{N}(0,1)$. Elle calcule les échantillons

$$x = \sigma_1 u_1 \quad \text{et} y = \sigma_2 \rho u_1 + \sigma_2 \sqrt{1-\rho^2}\, u_2 \ .$$

Elle ouvre une fenêtre graphique qu'elle divise en deux. Elle affiche dans ces fenêtres.

1. Les points dont les abscisses sont les valeurs de l'échantillon x, les ordonnées sont les valeurs de l'échantillon y.
2. La surface d'équation $z = f(x,y)$, où

$$f(x,y) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \exp\left(-\frac{1}{2}(x,y)K^{-1}\begin{pmatrix}x\\y\end{pmatrix}\right) \ ,$$

où K^{-1} est la matrice inverse de la matrice de covariance K :

$$K = \begin{pmatrix} \sigma_1^2 & \sigma_1\sigma_2\rho \\ \sigma_1\sigma_2\rho & \sigma_2^2 \end{pmatrix}$$

2.3 QCM

Donnez-vous une heure pour répondre à ce questionnaire. Les 10 questions sont indépendantes. Pour chaque question 5 affirmations sont proposées, parmi lesquelles 2 sont vraies et 3 sont fausses. Pour chaque question, cochez les 2 affirmations que vous pensez vraies. Chaque question pour laquelle les 2 affirmations vraies sont cochées rapporte 2 points.

Question 1. La ligne de commande proposée affecte à `v` le vecteur des entiers consécutifs de 1 à 10.

A `v <- seq(1,1,10)`
B `v <- (0:1)`
C `v <- sort(sample(1:10,10))`
D `v <- 1; for (i in 2:10){append(v,i)}`
E `v <- rep(1,10); while (i<=10){v[i] <- i}`

Question 2. La ligne de commande proposée retourne la matrice identité à 2 lignes et 2 colonnes.

A `diag(rep(1,2))`
B `identity(2,2)`
C `rbind(c(1,0),c(0,1))`
D `1:0%*%t(0:1)+0:1%*%t(1:0)`
E `toeplitz(c(0,1))`

Question 3. Soit x un vecteur binaire. La ligne de commande proposée affiche un vecteur à deux entrées, dont la première est le nombre de coordonnées de x qui valent 0, la seconde le nombre de coordonnées qui valent 1.

A `as.vector(table(factor(x,levels=0:1)))`
B `which(x==c(0,1))`
C `ifelse(x==0,yes=1,no=0)`
D `tabular(x)`
E `c(length(x)-sum(x),sum(x))`

Question 4. Soit x un vecteur numérique d'entiers. La ligne de commande proposée affiche un diagramme en bâtons des valeurs de x.

A `barplot(table(x))`
B `plot(table(x))`
C `hist(x)`
D `barplot(x)`
E `plot(x)`

Question 5. Soit x un vecteur numérique. La ligne de commande proposée retourne un vecteur numérique dont les 3 coordonnées sont les quartiles de x.

A `quantile(x,c(0.25,0.5,0.75))`
B `as.vector(quantile(x,(1:3)/4))`
C `sort(x)[length(x)*(1:3)/4]`
D `x[floor(length(x)*(1:3)/4)]`
E `as.numeric(quantile(x,0.25*c(1,2,3)))`

Question 6. La ligne de commande proposée retourne un échantillon de taille 10 de la loi de Bernoulli de paramètre 1/3.

A `ifelse(runif(10)<1/3,yes=0,no=1)`
B `rbinom(10,1,1/3)`
C `sample(c(0.1),10,replace=T)`
D `ifelse(rgeom(10,0.3)==0,yes=1,no=0)`
E `floor(runif(10,0,3))`

Question 7. La ligne de commande proposée retourne 0.5.

A `pt(0,3)`
B `dbinom(0,1,1/2)`
C `qt(0)`
D `qnorm(0)`
E `pchisq(2,2)`

Question 8. La ligne de commande proposée définit la fonction `moins` qui à un couple de vecteurs associe sa différence.

A `function <- moins(x,y){return(u-v)}`
B `moins <- function{return(u-v)}`
C `moins <- function(u,v){return(u-v)}`
D `function moins(u,v){return(u-v)}`
E `moins <- function(u,v)u-v`

Question 9. Soit x un échantillon de taille 1000 d'une loi d'espérance μ inconnue. La ligne de commande proposée retourne un vecteur à deux entrées dont les composantes sont les bornes d'un l'intervalle de confiance de niveau asymptotique 0.95 pour μ.

A `mean(x)+sd(x)*qnorm(0.025,0.975)/sqrt(length(x))`
B `mean(x)+sd(x)*qnorm(0.975)*c(-1,1)`
C `as.vector(t.test(x)$conf.int)`
D `mean(x)+sd(x)*qnorm(0.975)*c(-1,1)/sqrt(length(x))`
E `t.test(x)$conf.int`

Question 10. Soit x un échantillon de taille 1000 d'une loi d'espérance μ inconnue. La ligne de commande proposée retourne la p-valeur d'un test unilatéral de $\mathcal{H}_0 : \mu = 1$ contre $\mathcal{H}_1 : \mu < 1$.

A `t.test(x,mu=1,alternative="less")$p.value`
B `pnorm(sqrt(length(x))*(mean(x)-1)/sd(x))`
C `1-pnorm(sqrt(length(x))*mean(x)/sd(x))`
D `t.test(x,alternative="less")$p.value`
E `pnorm((mean(x)-1)/(sd(x)*length(x)))`

Réponses : 1–BC 2–AC 3–AE 4–AB 5–BE 6–BD 7–AB 8–CE 9–CD 10–AB

2.4 Devoir

Essayez de bien rédiger vos réponses, sans vous reporter ni au cours, ni au corrigé. Si vous souhaitez vous évaluer, donnez-vous trois heures ; puis comparez vos réponses avec le corrigé et comptez un point pour chaque question à laquelle vous aurez correctement répondu.

Questions de cours :

1. Tirer un échantillon x de taille 1000 de la loi exponentielle $\mathcal{E}(1)$. Calculer la moyenne et l'écart-type empiriques, ainsi que les quantiles d'ordre 0.25, 0.5, 0.75.
2. Représenter un histogramme des valeurs obtenues en bleu. Superposer sur le même graphique une estimation de la densité en vert et la densité exacte de la loi $\mathcal{E}(1)$ en rouge.
3. Représenter la fonction de répartition empirique de x en bleu. Superposer sur le même graphique la fonction de répartition théorique en rouge.

4. On considère l'échantillon des parties entières des valeurs de x. Calculer la table de fréquence des valeurs observées, puis représenter un diagramme en bâtons en bleu. Juxtaposer sur le même graphique les probabilités de la loi géométrique de paramètre $1 - \exp(-1)$, en rouge.
5. Appliquer le test de Kolmogorov-Smirnov bilatéral pour tester l'ajustement de l'échantillon x à la loi $\mathcal{E}(1)$. Extraire du résultat la p-valeur.

Problème : Soit $(X_1, \ldots, X_n)$ un échantillon de la loi exponentielle $\mathcal{E}(\lambda)$, de densité $\lambda e^{-\lambda x}$ sur $\mathbb{R}^+$. L'estimateur du maximum de vraisemblance pour λ est noté T. C'est l'inverse de la moyenne empirique :

$$T = \frac{n}{X_1 + \cdots + X_n} .$$

On admettra que l'espérance de T vaut $\lambda n/(n-1)$, et que $n\lambda/T$ suit la loi gamma $\mathcal{G}(n, 1)$. On note $T' = T(n-1)/n$ l'estimateur sans biais basé sur T.

1. Tirer 1000 échantillons de taille 20 de la loi $\mathcal{E}(\lambda)$, pour $\lambda = 2$. Pour chacun des 1000 échantillons, calculer l'estimation ponctuelle de λ par l'inverse de la moyenne empirique, puis la valeur débiaisée. Représenter les estimations de densité de ces deux échantillons d'estimations. Superposer sur le même graphique une ligne verticale donnant la vraie valeur.
2. Écrire une fonction `ic.moy` qui prend en entrée un vecteur de données `X` et un niveau de confiance `nc` (valeur par défaut 0.95), et qui retourne en sortie un vecteur à deux entrées contenant les bornes de l'intervalle de confiance pour λ, déduit de l'intervalle de confiance générique sur la moyenne (fonction `t.test`).
3. Écrire une fonction `ic.sym` qui prend en entrée un vecteur de données `X` et un niveau de confiance `nc` (valeur par défaut 0.95), et qui retourne en sortie un vecteur à deux entrées contenant les bornes de l'intervalle de confiance symétrique de niveau `nv` pour l'échantillon `X`, basé sur la loi gamma.
4. Ecrire une fonction `test.gamma` qui prend en entrée un vecteur de données `X` et une valeur du paramètre `la0`, et qui retourne les p-valeurs de 2 tests unilatéraux de l'hypothèse $\mathcal{H}_0$: $\lambda =$`la0`, contre $\mathcal{H}_1$: $\lambda >$`la0`. Le premier test est le test de la moyenne (fonction `t.test`), le second utilise la loi de T (fonction `pgamma`).
5. On considère deux autres estimateurs de λ.

$$T_1 = \left(\frac{1}{2n}(X_1^2 + \cdots + X_n^2)\right)^{-1/2} \quad \text{et} \quad T_2 = \frac{\log(2)}{M} ,$$

où M désigne la médiane de l'échantillon. Sur les 1000 échantillons de taille 20 de la question 1, calculer les estimations obtenues par T, T', T_1 et T_2. Représenter sur un même graphique les 4 diagrammes en boîte verticaux (fonction `boxplot`). Représenter la vraie valeur par un trait horizontal.

6. Tirer un échantillon de taille $n = 1000$ de la loi $\mathcal{E}(\lambda)$, pour $\lambda = 2$: (X_i). On note $x_1, \ldots, x_{n-1}$ les valeurs de (X_i) rangées par ordre croissant (statistiques d'ordre, sauf le maximum). Pour tout $i = 1, \ldots, n-1$, on pose $y_i = -\log(1 - 1/i)$. Calculer la pente et l'ordonnée à l'origine de la droite de regression linéaire des y_i sur les x_i. Représenter sur un même graphique la droite de régression linéaire et les points de coordonnées (x_i, y_i).
7. On appelle T_r la pente de la droite de regression calculée comme dans la question précédente : c'est un nouvel estimateur de λ. Écrire une fonction `reg.est` qui prend en entrée un vecteur de données `X`, et retourne la valeur de T_r.
8. Écrire une fonction `compare.est`, qui prend en entrée une valeur `lambda`, deux entiers `E` et `n`. La fonction tire `E` échantillons de taille `n` de la loi exponentielle de paramètre `lambda`. Elle calcule pour chacun les estimations obtenues par T, T', T_1, T_2, et T_r. Elle calcule ensuite l'erreur quadratique moyenne des `E` estimations pour chacun des 5 estimateurs. Elle retourne dans une table ces 5 erreurs quadratiques moyennes.
9. Tirer deux échantillons de taille $n = 1000$ de la loi $\mathcal{E}(1)$: $(X_1, \ldots, X_n)$ et $(Y_1, \ldots, Y_n)$. Parmi les valeurs du premier échantillon, conserver les valeurs X_i qui vérifient $Y_i > (1 - X_i)^2/2$. Soit (U_i) l'échantillon des valeurs conservées. Représenter un histogramme de l'échantillon (U_i^2). Superposer sur le même graphique la densité de la loi de chi-deux à 1 degré de liberté.
10. Affecter chaque valeur de (U_i) du signe + ou −, choisi avec probabilité 1/2. Soit (Z_i) l'échantillon ainsi obtenu. Reésenter la fonction de répartition empirique de l'échantillon (Z_i). Superposer sur le même graphique la fonction de répartition théorique de la loi $\mathcal{N}(0, 1)$.
11. Vérifier l'ajustement de l'échantillon (Z_i) à la loi $\mathcal{N}(0, 1)$:
    (a) par les quantiles (fonction `qqnorm`),
    (b) par le test de Shapiro-Wilk (fonction `shapiro.test`),
    (c) par le test de Kolmogorov-Smirnov (fonction `ks.test`).
12. Simuler trois échantillons de taille $n = 1000$ de la loi $\mathcal{E}(1)$: (X_i), (Y_i), (Z_i). Pour tout $i = 1, \ldots, n$, on pose :

$$S_i = \frac{X_i}{X_i + Y_i + Z_i} \quad \text{et} \quad T_i = \frac{X_i + Y_i}{X_i + Y_i + Z_i} \, .$$

    Représenter par des points dans le plan les couples (S_i, T_i). Vérifier par le test de Kolmogorov-Smirnov, l'ajustement de l'échantillon S_i à la loi bêta $\mathcal{B}(1, 2)$, et l'ajustement de l'échantillon (T_i) à la loi bêta $\mathcal{B}(2, 1)$.
13. Simuler deux échantillons de taille $n = 1000$ de la loi uniforme sur $[0, 1]$: (U_i) et (V_i). Pour tout $i = 1, \ldots, n$, on pose :

$$S'_i = \min\{U_i, V_i\} \quad \text{et} \quad T'_i = \max\{U_i, V_i\} \, .$$

Vérifier l'ajustement des deux échantillons S et S' d'une part, T et T' d'autre part, par :

(a) le test de Kolmogorov-Smirnov

(b) le test de Wilcoxon (fonction `wilcox.test`)

14. Pour $\lambda = 1/2$, tirer un échantillon de taille $n = 1000$ de la loi $\mathcal{E}(1/2)$: (T_i). Pour tout $i = 1, \ldots, n$, calculer $p_i = \exp(-T_i)$, et simuler une variable aléatoire Y_i, de loi géométrique de paramètre p_i (le tout sans utiliser de boucle). Représenter les valeurs de l'échantillon par un diagramme en boîte. Calculer les déciles et le maximum.

15. On admettra que pour tout entier $k \in \mathbb{N}$, la probabilité que Y_i prenne la valeur k est $\lambda B((\lambda+1),(k+1))$, où $B(x,y)$ désigne la fonction `beta`. Dans l'échantillon Y, remplacer toutes les valeurs supérieures à 10 par 10. Calculer les probabilités théoriques, puis appliquer le test du chi-deux d'ajustement (fonction `chisq.test`).

2.5 Corrigé du devoir

Questions de cours :

1.

```
x <- rexp(1000)                # paramètre 1 par défaut
mean(x)
sd(x)
quantile(x,c(0.25,0.5,0.75))
```

2.

```
{hist(x,
      xlim=c(0,6),             # fixer pour plusieurs graphiques
      col="blue",              # la couleur
      freq=F,                  # fréquences plutôt que effectifs
      main="loi exponentielle",# titre du graphique
      ylab="")}                # pas d'étiquette en ordonnée
d <- density(x)                # estimation de densité
par(new=T)                     # pour superposer les graphiques
{plot(d,col="green",xlim=c(0,6),
       axes=F,main="",xlab="",ylab="")}
curve(exp(-x),col="red",add=T) # ajoute la vraie densité
```

3.

```
f <- ecdf(x)                   # fonction de répartition
plot(f,main="fonction de répartition",col="blue")
curve(1-exp(-x),col="red",add=T)
```

4.

```
i <- floor(x)                  # parties entières
t <- table(i)                  # valeurs et effectifs
val <- as.numeric(row.names(t))# valeurs
fr <- as.vector(t)             # effectifs
fr <- fr/sum(fr)               # fréquences
pr <- dgeom(val, 1-exp(-1))    # probabilités
fp <- rbind(fr,pr)             # matrice frequences-probabilites
                               # diagramme en barres double
barplot(fp,beside=T,col=c("blue","red"))
```

5.

```
ks.test(x,"pexp",1)            # appliquer le test
ks.test(x,"pexp",1)$p.value    # extraire la p-valeur
```

Problème :

1.

```
ech <- 1000                    # nombre d'échantillons
n <- 20                        # taille d'échantillon
lambda <- 2                    # valeur du paramètre
X <- rexp(ech*n,rate=lambda)   # tirages aléatoires
X <- matrix(X,ech,n)           # redimensionner
T <- 1/rowMeans(X)             # inverse des moyennes
Tp <- T*(n-1)/n                # estimations débiaisées
dT <- density(T)               # estimation de densité
dTp <- density(Tp)
rep <- lambda*c(0.5,2)         # intervalle de représentation
{plot(dT,col="blue",xlim=rep,
      main="estimateurs de lambda",xlab="")}
lines(dTp,col="green",xlim=rep)
abline(v=lambda,col="red")     # vraie valeur
```

2.

```
ic.moy <- function(X,nc=0.95){
#  retourne l'intervalle de confiance pour lambda
#  basé sur la fonction t.test
#
ic <- t.test(X,conf.level=nc)  # appel de la fonction
ic <- ic$conf.int              # extraire l'intervalle
ic <- as.vector(ic)            # conserver les deux bornes
ic <- 1/ic                     # inverser les deux bornes
ic <- sort(ic)                 # remettre dans l'ordre
```

```
return(ic)                      # retourner l'intervalle
}
```

3.

```
ic.sym <- function(X,nc=0.95){
#  retourne l'intervalle de confiance pour lambda
#  basé sur la loi gamma
#
a2 <- (1-nc)/2                # probabilité des ailes
n <- length(X)                # taille d'échantillon
T <- 1/mean(X)                # estimation
q1 <- qgamma(a2,shape=n)      # quantile inférieur
q2 <- qgamma(1-a2,shape=n)    # quantile supérieur
ic <- c(q1,q2)                # intervalle de quantiles
ic <- ic*T/n                  # intervalle de confiance
return(ic)                    # retourner l'intervalle
}
```

4.

```
test.gamma <- function(X,la0){
#  retourne deux p-valeurs pour le test
#  de lambda=la0 contre lambda>la0
#
p1 <- {t.test(X,              # test de la moyenne
       alternative="less",    # contre moyenne inférieure
       mu=1/la0)$p.value}     # extraire la p-valeur
T <- 1/mean(X)                # estimation
S <- n*la0/T                  # statistique de test
p2 <- pgamma(S,shape=n)       # loi sous l'hypothèse nulle
return(c(p1,p2))              # retourner les deux valeurs
}
```

5.

```
ech <- 1000                    # nombre d'échantillons
n <- 20                        # taille d'échantillon
lambda <- 2                    # valeur du paramètre
X <- rexp(ech*n,rate=lambda)   # tirages aléatoires
X <- matrix(X,ech,n)           # redimensionner
T <- 1/rowMeans(X)             # inverse des moyennes
Tp <- T*(n-1)/n                # estimations débiaisées
T1 <- (rowMeans(X^2)/2)^(-1/2) # estimation par les carrés
T2 <- log(2)/apply(X,1,median) # estimation par la médiane
A <- cbind(T,Tp,T1,T2)         # faire un tableau
```

```
{boxplot(data.frame(A),names=  # représenter les boxplots
         c("T","Tp","T1","T2"))}
abline(h=lambda,col="red")     # vraie valeur
```

6.

```
lambda <- 2                    # paramètre
n <- 1000                      # taille des échantillons
x <- rexp(n,rate=lambda)       # échantillon exponentiel
x <- sort(x)                   # valeurs triées
x <- x[-n]                     # supprimer la dernière
F <- (1:(n-1))/n               # ordonnées de la fonction
y <- -log(1-F)                 # changement de variable
plot(x,y, pch=".", col="blue") # couples de points
reg<-lm(y~x)                   # régression linéaire
abline(ref,col="red")          # tracer la droite
```

7.

```
reg.est <- function(X){
#  retourne l'estimation de lambda par
#  régression non linéaire sur la fonction
#  de répartition empirique de X
#
n <- length(X)                 # taille d'échantillon
x <- sort(X)                   # valeurs triées
x <- x[-n]                     # supprimer la dernière
F <- (1:(n-1))/n               # ordonnées de la fonction
y <- -log(1-F)                 # changement de variable
reg <- lm(y~x)                 # regression linéaire
return(reg$coefficients[2])    # retourner la pente
}
```

8.

```
compare.est <- function(lambda,E,n){
#  retourne l'erreur quadratique moyenne de
#  5 estimateurs de lambda, calculée sur
#  E échantillons de taille n
#
X <- rexp(E*n,rate=lambda)     # tirages aléatoires
X <- matrix(X,E,n)             # redimensionner
T <- 1/rowMeans(X)             # inverse des moyennes
Tp <- T*(n-1)/n                # estimations débiaisées
T1 <- (rowMeans(X^2)/2)^(-1/2) # estimation par les carrés
T2 <- log(2)/apply(X,1,median) # estimation par la médiane
```

```
Tr <- rep(0,E)                  # estimation par régression
for (i in 1:E){                 # pour chaque échantillon
   x <- X[i,]                   # extraire l'échantillon
   Tr[i]<-reg.est(x)            # estimation
}                               # fin de boucle
A <- rbind(T,Tp,T1,T2,Tr)       # faire un tableau
A <- (A-lambda)^2               # carrés des erreurs
q <- sqrt(rowMeans(A))          # erreurs quadratiques
return(q)                       # retourner les erreurs
}
> compare.reg(2,1000,100)
        T        Tp        T1        T2        Tr
0.2068059 0.2029628 0.2286295 0.3020214 0.2821136
```

9.

```
n <- 1000                       # taille des échantillons
X <- rexp(n); Y <- rexp(n)      # échantillons
ind <- which(Y>((1-X)^2)/2)     # indices à conserver
U <- X[ind]                     # valeurs conservées
hist(U^2,breaks=20,freq=F)      # histogramme
{curve(dchisq(x,1),col="red",   # densité de la loi du chi-deux
        add=T)}
```

10.

```
m <- length(U)                  # longueur du nouvel échantillon
S <- 2*rbinom(m,1,0.5)-1        # signes aléatoires
Z <- U*S                        # nouvelles valeurs
f <- ecdf(Z)                    # fonction de répartition
plot(f,col="blue",main="loi normale")
curve(pnorm(x),col="red",add=T)
```

11.

```
qqnorm(Z)
ks.test(Z,"pnorm")
shapiro.test(Z)
```

12.

```
n <- 1000                       # taille des échantillons
X<-rexp(n)
Y<-rexp(n)
Z<-rexp(n)
S <- X/(X+Y+Z)
T <- (X+Y)/(X+Y+Z)
```

```
plot(S,T, pch=".", col="blue") # couples de points
ks.test(S,"pbeta",1,2)
ks.test(T,"pbeta",2,1)
```

13.

```
n <- 1000;                     # taille des échantillons
X<-runif(2*n,0,1)              # tirages
X <- matrix(X,2,n)             # 2 lignes, n colonnes
X <- apply(X,2,sort)           # trier chaque colonne
Sp <- X[1,]                    # première ligne
Tp <- X[2,]                    # seconde ligne
ks.test(S,Sp)
ks.test(T,Tp)
wilcox.test(S,Sp)
wilcox.test(T,Tp)
```

14.

```
lambda <- 0.5                  # paramètre
n <- 1000                      # taille des échantillons
T <- rexp(n,rate=lambda)       # échantillon exponentiel
p <- exp(-T)                   # paramètres
Y <- rgeom(n,p)                # échantillon de géométriques
boxplot(Y)                     # des valeurs trop extrêmes
quantile(Y,seq(0.1,0.9,by=0.1))# déciles
max(Y)                         # maximum
```

15.

```
tr <- 10                       # valeur de troncature
Y[which(Y>tr)] <- tr           # troncature
pr <- lambda*beta((1+lambda)*rep(1,tr),1:tr)
pr <- append(pr,1-sum(pr))     # probabilités théoriques
t <- table(Y)                  # effectifs observés
chisq.test(t,p=pr)             # test du chi-deux
```

3 Compléments

3.1 À deux ou trois lieues près

> Par Décret Royal du Conseil des Indes de la Couronne de Castille, Van Langren a écrit un mémoire avec les règles théoriques et pratiques pour que les pilotes calculent les degrés de longitude à partir de la méthode du point fixe et par navigation à latitude constante. Et ayant corrigé la Géographie, ils pourront savoir combien de degrés il y a entre le navire et l'endroit où il désire aller, sans qu'il y ait nécessité de conjecture ni de quadrature, comme on le faisait jusqu'ici, à grand peine et perte de navires. Tout étant si exact que l'erreur sera moindre que deux ou trois lieues aux latitudes moyennes et de quatre ou cinq lieues à l'équateur.
>
> Le secret important de la longitude a été recherché par d'éminents scientifiques. Comme les rois catholiques ont offert beaucoup d'argent à celui qui le résoudrait, beaucoup ont tenté de le résoudre, d'où ont résulté beaucoup de dépenses et d'abus, d'où vient que les rois se défient des promesses de ceux qui disent détenir la solution.

En 1644, Michael Florent van Langren (1598–1675) signe sa lettre en tant que « cosmographe et mathématicien de Sa Majesté en Flandre » [1]. Sa majesté Philippe iv d'Espagne, règne alors sur une bonne partie de l'Europe de l'Ouest et de l'Amérique du Sud. Faire en sorte que Ses navires, chargés de l'or du Pérou ou des marchandises européennes parviennent à bon port, c'est s'assurer Sa reconnaissance, sonnante et trébuchante. Or si l'on sait depuis longtemps déterminer la latitude d'un lieu par observation des étoiles visibles, le calcul de la longitude, sans horloge d'une précision suffisante, est d'une toute autre difficulté. L'enjeu pour van Langren est de convaincre Sa Majesté qu'il peut faire mieux que les données disponibles du temps. « Deux ou trois lieues », soit une dizaine de kilomètres, cela peut vous paraître beaucoup quand il s'agit de ne pas manquer l'entrée d'un port. Mais van Langren sait que c'est bien plus précis que les données disponibles en son temps. Pour mieux en souligner la grande variabilité, il a une idée : il représente les estimations connues de la différence de longitude entre Tolède et Rome sur un axe horizontal (figure 22).

C'est la toute première représentation graphique de données statistiques. Van Langren faisait tout son possible pour emporter l'adhésion de Sa Majesté (et la gratification qui s'ensuivrait) ; et, si on peut oser cet anachronisme, il n'y avait pas photo ! Les cartes disponibles surestimaient toutes la longitude exacte de plusieurs degrés, jusqu'à plus de 10. Or à la latitude de Rome, 10 degrés de longitude, cela fait tout de même la bagatelle de 1000 kilomètres. Autant vous dire que les pilotes de Sa Majesté avaient plus intérêt à compter sur leur expérience de la navigation que sur leurs cartes !

1. M. Friendly, P. Valero-Mora, J. Ibáñez Ulargui : The first (known) statistical graph : Michael Florent van Langren and the "secret" of longitude, *http ://www.datavis.ca/gallery/langren/*

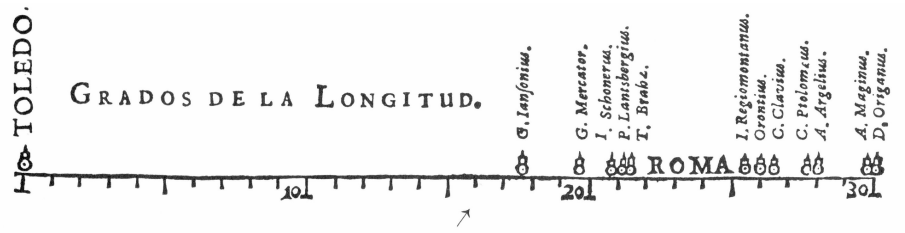

FIGURE 22 – M. F. van Langren : représentation graphique des différences de latitude estimées entre Tolède et Rome. La flèche indique la valeur exacte.

3.2 Un écossais a pris la Bastille

Parmi les quelques centaines d'habitants du faubourg Saint-Antoine qui, assemblés en milice le 14 juillet 1789, participèrent à la prise de la Bastille, il y avait un écossais : William Playfair (1759–1823) [2]. Après un emploi d'assistant dans la fabrique de machines à vapeur de James Watt (oui, ce Watt-là), il avait fondé à Londres une affaire d'orfèvrerie qui avait fait faillite. Il avait alors décidé ... en 1787 (!) que la France offrirait plus d'opportunité à un entrepreneur aussi inventif que lui. Des idées de nouvelles affaires à monter il n'en manquait jamais, quitte à franchir parfois les limites dictées par son nom de famille et les règles juridiques, au point même de se voir condamner à la prison pour détournement de fonds. La Révolution française, qu'il avait d'abord soutenue activement, sombra bientôt dans une violence qu'il n'approuvait plus. Au péril de sa vie, il alla jusqu'à affronter en 1791 une populace hostile qui avait pris à partie un de ses amis (et associé en affaires). Avec pas mal d'audace et de chance, il réussit à sauver son ami du lynchage, et se retrouve à l'abri dans une caserne. Sa manière de conclure l'aventure est typique de l'indéfectible optimisme avec lequel il traverse l'existence.

> When I found myself on the bed in the Corps de Garde I began to think I had run some risk, but when I was conducting my friend to a place of safety I thought nothing but of what I was about. I saw a boy in the place who seemed to be a sort of messenger and I asked him if he could get me some wine. He said the Swiss sold excellent wine at 15 sols. I desired him to fetch a dozen and I found it was the best common wine I ever tasted. I then sent for two dozen more and asked all the soldiers to drink. In return their onion soup, which had come along before but which they had been prevented from eating by the disturbance that had taken place, was offered to me and I ate of it with amazing appetite.

Après la chute de Napoléon, pour répondre à un récit trop favorable selon lui à la Révolution, Playfair expose sa vision de la France dans deux gros volumes, aussitôt traduits en français. Appreciez, Mesdemoiselles, l'admiration qu'il témoigne pour les françaises de son temps [3].

2. I. Spence, H. Wainer : Who was Playfair ? *Chance, 10, pp. 35–37, 1997*
3. W. Playfair : la France telle qu'elle est, *Cosson, Paris, 1820*

> Dans tous les exercices qui exigent de l'audace, ou qui sont dangereux, les femmes en France se piquent toujours de paroître au premier rang. Elles sont fières de déployer un mâle courage, ce qui n'est pas toujours très-convenable, ni même sans danger. Les hommes connoissent si bien toute la force de ce penchant, que, tout en les voyant avec peine oublier ainsi les plus beaux attributs de leur sexe, la pudeur et la modestie, ils n'osent les empêcher de suivre leur goût, persuadés que ce seroit les offenser inutilement, puisque toute remontrance seroit sans effet.
>
> [...]
>
> En traçant cette légère esquisse de la société parisienne, nous ne devons pas oublier ce sexe qui en fait le plus grand charme. Ce furent les Parisiennes qui, à l'époque de la révolution, prouvèrent que la sensibilité a aussi son héroïsme, et que l'affection peut donner une énergie qui fait braver tous les périls.
>
> Les Parisiennes ne craignirent pas de descendre dans de sombres cachots, de visiter les demeures ordinairement occupées par le crime, pour aller porter des paroles d'espoir et de consolation aux pauvres prisonniers. Elles ont montré qu'elles savoient compatir au malheur ; elles ont montré aussi qu'elles savoient supporter la douleur et la mort. La tendre fille, l'épouse fidèle, conduite sur l'échafaud avec un père ou un époux, sembloit oubliier qu'elle alloit perdre la vie, pour ne songer qu'à ranimer le courage abattu de celui pour lequel elle faisoit le sacrifice de son existence.
>
> [...]
>
> Ce qui fait le principal charme de la beauté chez les Françoises, c'est l'expression de la physionomie. Indépendamment de l'aisance de ses manières, la Françoise a ordinairement un air d'enjouement et de vivacité qui plaît et qui captive. Elle paroît toujours prête à se lier avec vous et vous fait un accueil gracieux ; mais, parce qu'elle est très-communicative, n'allez pas en conclure qu'elle ne soit pas réllement vertueuse. L'affabilité et l'enjouement sont des preuves aussi certaines de la vertu d'une femme que la réserve et la pruderie.

Fort bien : Playfair était un touche-à-tout bouillonnant d'idées, galant et amoureux de la vie, pas toujours très scrupuleux, mais intelligent et somme toute bien sympathique ; mais vous connaissez le principe de ces compléments : cela ne suffirait pas pour qu'il y figure s'il n'y avait une quelconque connexion avec le thème du chapitre, en l'occurrence le logiciel R. Cela ne vous a pas échappé, une grande partie de l'usage de R consiste en des *représentations graphiques* de données. Qui le premier a théorisé l'importance de la représentation graphique des données statistiques ? Qui a inventé le diagramme en barres et en camembert ? Oui, c'est bien notre touche-à-tout. Il avait parfaitement compris en quoi résidait la supériorité de ses représentations graphiques sur les tableaux de chiffres.

> The advantage proposed by this method, is not that of giving a more accurate statement than by figures, but it is to give a more simple and permanent idea of the gradual progress and comparative amounts, at different periods, by presenting to the eye a figure, the proportions of which correspond with the amount of sums intended to be expressed.

Il avait publié en 1786 à Londres un « Commercial and Political Atlas ». Un exemplaire, parvenu en France avait été offert au Roi Louis XVI, qui...

> ... at once understood the charts and was very pleased. He said they spoke all languages and were very clear and easily understood.

Playfair est resté très fier de la reconnaissance de son travail par Louis XVI, dont l'appui financier et politique lui avait été précieux. Il utilise ses techniques graphiques dans plusieurs ouvrages, dont son « Bréviaire Statistique », publié en 1801 [4]. Voici ce qu'il y dit de sa méthode graphique. Oubliez le style d'il y a deux siècles : les idées n'ont pas pris une ride.

> I have composed the following work upon the principle of which I speak ; this, however, I never should have thought of doing, had it not occured to me, that making an appeal to the eye when proportion and magnitude are concerned, is the best and readiest method of conveying a distinct idea.
>
> Statistical knowledge, though in some degree searched after in the most early ages of the world, has not, till within these last fifty years, become a regular object of study. Its utility to all persons connected in any way with public affairs, is evident : and indeed it is no less evident that everyone who aspires at the character of a well-informed man should acquire a certain degree of knowledge on a subject so universally important, and so generally canvassed.
>
> [...]
>
> The advantages proposed by this mode of representation, are to facilitate the attainment of information, and aid the memory in retaining it : which two points form the principal business in what we call learning, or the acquisition of knowledge.
>
> Of all the senses, the eye gives the liveliest and most accurate idea of whatever is suceptible of being represented to it ; and when proportion between different quantities is the object, then the eye has an incalculable superiority ; as from the constant, and even involuntary habit of comparing the sizes of the objects it has acquired the capacity of doing so, with an accuracy that is almost unequalled.
>
> [...]

4. W. Playfair : The statistical breviary ; shewing, on a principle entirely new, the resources of every state and kingdom in Europe, *Bensley, London, 1801*

> The author of this work applied the use of lines to matters of commerce and finance about sixteen years ago, with great success. His mode was generally approved of as not only facilitating, but rendering those studies more clear, and retained more easily by the memory.

3.3 Un amas indigeste de chiffres et de tableaux

Après Waterloo, il ne faisait pas bon en France avoir exprimé une quelconque sympathie bonapartiste. Alors pensez : ayant été nommé par Napoléon d'abord préfet de l'Isère, puis du Rhône après le retour de l'île d'Elbe, Joseph Fourier (1768–1830) n'en menait pas large [5].

> La seconde restauration trouva Fourier dans la capitale, sans emploi et justement inquiet pour son avenir. Celui qui, pendant quinze ans, administra un grand département ; qui dirigea des travaux si dispendieux ; qui, dans l'affaire des marais de Bourgoin, eut à stipuler pour tant de millions, avec les particuliers, les communes et les compagnies, ne possédait pas vingt mille francs de capital. Cette honorable pauvreté, le souvenir des plus importants, des plus glorieux services, devaient peu toucher des ministres voués alors aux colères de la politique et aux caprices de l'étranger. Une demande de pension fut donc repoussée avec brutalité. Qu'on se rassure ! La France n'aura pas à rougir d'avoir laissé dans le besoin une de ses principales illustrations. Le Préfet de Paris, je me trompe, Messieurs, un nom propre ne sera pas de trop ici, M. de Chabrol apprend que son ancien professeur à l'École polytechnique, que le secrétaire perpétuel de l'Institut d'Égypte, que l'auteur de la Théorie analytique de la chaleur, va être réduit, pour vivre, à courir le cachet. Cette idée le révolte. Aussi se montre-t-il sourd aux clameurs des partis, et Fourier reçoit de lui la direction supérieure du Bureau de la statistique de la Seine, avec 6000 francs d'appointements. J'ai cru, Messieurs, ne pas devoir taire ces détails. Les sciences peuvent se montrer reconnaissantes envers tous ceux qui leur donnent appui et protection quand il y a quelque danger à le faire, sans craindre que le fardeau devienne jamais trop lourd !
>
> Fourier répondit dignement à la confiance de M. de Chabrol. Les mémoires dont il enrichit les intéressants volumes publiés par la préfecture de la Seine, serviront désormais de guide à tous ceux qui ont le bon esprit de voir dans la statistique, autre chose qu'un amas indigeste de chiffres et de tableaux.

Rentré en grâce, et devenu secrétaire perpétuel de l'Académie des Sciences pour la section des Mathématiques, Fourier exprime sa reconnaissance dans l'Histoire de l'Académie pour l'année 1822.

5. F. Arago : Éloge historique de Joseph Fourier, *Académie des Sciences, Paris, 1833*

> L'Académie se souvient du beau travail dont M. le Comte de Chabrol a réuni les matériaux nombreux et authentiques, qu'il a publiés, en 1821, sous le titre de *Recherches statistiques sur la Ville de Paris et le département de la Seine*, et qui contient 62 tableaux. Elle apprend avec intérêt que ce magistrat continue ses précieuses recherches, les seules à présent dans leur genre, et que la suite en doit paraître incessament.
>
> Grâces soient rendues aux administrateurs qui font servir l'influence et l'autorité de leurs importantes fonctions, ainsi que les secours de tout genre dont ils peuvent disposer, à résoudre des questions d'un égal intérêt pour le Gouvernement et les particuliers, pour les Sciences exactes et pour les spéculations de l'Économie politique.

Fourier est peut être un peu excessif dans ses louanges. Il sait fort bien que M. le Comte de Chabrol n'a pas écrit une ligne du monument en 4 gros volumes que sont les « Recherches statistiques sur la Ville de Paris et le département de la Seine ». Ce « beau travail » est dû pour l'essentiel à son adjoint au Bureau des Statistiques F. Villot, et aussi à lui-même : Fourier a contribué au moins aux différentes préfaces et il a écrit deux mémoires sur le « calcul des erreurs » (nous dirions marges d'incertitude ou intervalles de confiance). Voici ce qui est dit dans la préface du tome 4, à propos de la présentation des résultats.

> On a conservé, dans tout le cours de cet ouvrage l'emploi des tableaux, ce qui permet de réunir un nombre immense de résultats, écarte les dissertations superflues, et facilite tous les rapprochements. Ce même mode de publication a été employé depuis dans d'autres ouvrages, pour un ordre de faits très importants, ceux qui ont été l'objet de décisions en matière criminelle.
>
> Dans les descriptions statistiques, on a distingué les éléments que l'on peut, en quelque sorte, regarder comme invariables, et ceux qui subissent des variations assez fréquentes. Les premiers sont déterminés par les conditions naturelles du climat ou par des institutions sensiblement fixes : on s'est attaché, dans les parties de cet ouvrage déjà publiées, à décrire avec soin cette première classe de faits, et l'on a présenté les résultats plus variables dans des tableaux annuels qu'il est devenu facile de rédiger ; en sorte que le plan général étant formé, il suffira de publier, après un certain intervalle de temps, des suppléments périodiques, pour constater les changements survenus ; on parvient ainsi à concilier l'unité de plan et de vues, avec la diversité des états successifs.

3.4 La Dame à la lampe

Ses parents lui avaient donné le nom de la ville où elle était née : Florence. Vous pouvez estimer qu'elle avait eu plus de chance que sa sœur, née à Naples, et donc

prénommée Parthenope. Pourtant en 1820, Florence n'était pas encore un prénom courant. Mais 60 ans plus tard...

> ... you are Myth in your own lifetime. Do you know that there are thousands of girls about the ages of 18 to 23 named after you ? Everyone has heard of you and has a sweet association with your name.

Quelle Florence est-elle donc devenue un mythe de son vivant ? Florence Nightingale (1820–1910) [6]. Se sentant appelée par Dieu et cherchant sa vocation, elle pense d'abord aux mathématiques, ce dont ses parents ne veulent pas entendre parler. La destinée de Florence, écrit sa mère, sera de se marier, et ... à quoi serviraient les mathématiques pour une femme mariée ? C'est vrai, le rang social de sa famille fait d'elle un beau parti ; elle est jolie, brillante, danse fort bien et fait chavirer bien des cœurs. Pourtant elle s'obstine. Son père, qui a été son précepteur et lui a enseigné la philosophie, le grec et le latin, s'étonne. « Why mathematics ? I cannot see that mathematics would do great service. History or philosophy, natural or moral, I should like best. » La réponse de Florence n'est peut-être pas celle à laquelle vous auriez pensé. « I don't think I shall succeed so well in anything that requires quickness as in what requires only work [mathematics] ». Comme toujours dans sa vie, elle finit par avoir le dernier mot et obtient ses leçons de maths. On a dit que Sylvester avait été son professeur, mais il n'en existe aucune preuve. Par contre, elle a bien croisé la route d'un autre nom célèbre de l'histoire des sciences, même s'il ne semble pas qu'elles aient parlé de mathématiques.

> Lady Lovelace, who was Byron's daughter, had a "passion" for Florence and handed round a set of verses she had written in praise of her "soft and silver voice," her "grave and lucid eye."

Entre-temps, Florence avait trouvé sa vraie vocation.

> My mind is absorbed with the idea of the sufferings of man, it besets me behind and before ... all that the poets sing of the glories of this world seems to me untrue. All that the people I see are eaten up with care or poverty or disease.

Elle allait dédier chacun des instants de sa vie, avec une volonté, une exigence et une puissance de travail exceptionnelles, à soulager les souffrances de ses contemporains. Contrastant avec la mentalité romantique de son temps, elle fuit l'agitation stérile des grandes idées, et exige des réalisations concrètes.

> I think one's feelings waste themselves in words, they ought all to be distilled into actions and into actions which bring results.

Des idées d'actions qui pourraient apporter des résultats, il n'en manque pas : quand la Reine Victoria accède au trône en 1837, l'espérance de vie en Grande-Bretagne est de 38 ans. La mortalité infantile est effrayante, de nombreuses jeunes femmes meurent en couche, le choléra et le typhus font périodiquement des ravages, les épidémies sont redoutées mais mal comprises. Il y a encore pire : les hôpitaux militaires. Qu'un soldat

6. C. Woodham-Smith : Florence Nightingale (1820–1910), *Mac Graw Hill, New York, 1951*

meure à la guerre est dans l'ordre des choses. L'homme de troupe a la réputation d'une brute avinée guère plus évoluée qu'un animal, et personne ne s'est jamais trop soucié de ses souffrances. Quand la guerre de Crimée est déclarée en 1854, la condition des soldats est terrible, comme dans toutes les guerres qui ont précédé. Mais quelque chose a changé : le « Times » envoie un correspondant de guerre, qui fait son métier. Ses articles rapportent les horreurs dont il est témoin, et au passage, touchent un point sensible. Les hôpitaux de campagne de l'allié français sont mieux organisés, et ceux-ci bénéficient du travail efficace d'infirmières religieuses : les « Sœurs de la Charité ». Quarante ans après Waterloo, les français ont beau être devenus alliés, il n'est pas question de leur reconnaître une quelconque supériorité. Le ministère de la Guerre envoie en Crimée Florence Nightingale, à la tête d'un bataillon de 40 infirmières. Imaginez une jeune femme de la haute, débarquant au milieu de militaires professionnels, bien décidés à ne pas laisser une Lady leur donner des leçons. En dépit des conditions matérielles catastrophiques et des oppositions farouches, son dévouement, sa puissance de travail, ses qualités d'organisatrice font merveille.

> Her calmness, her resource, her power to take action raised her to the position of a goddess. The men adored her. "If she were at our head", they said, "we should be in Sebastopol next week." The doctors came to be absolutely dependent on her, and Colonel Sterling wrote home : "Miss Nightingale now queens it with absolute power."

De retour en Angleterre, elle est devenue une icône. Pourtant, la perte de tant de jeunes hommes dans la force de l'âge, à cause de maladies infectieuses qui auraient pu être enrayées par quelques règles simples d'hygiène et d'asepsie, lui laisse un profond sentiment d'échec. Comment convaincre les autorités des réformes qui s'imposent ? Et pour commencer, comment leur faire comprendre que le taux de mortalité dans les hôpitaux militaires est anormalement élevé, bien supérieur à celui des hôpitaux civils ? Il faut exposer les faits, donner des chiffres bien sûr, mais cela risque de ne pas suffire [7].

> Diagrams are of great utility for illustrating certain questions of vital statistics by conveying ideas on the subject through the eye, which cannot be so readily grasped when contained in figures. This aid has therefore been called in to give greater clearness to the numerical results in the body of the Report and in the Appendix.

Elle imagine même une nouvelle forme de diagramme en « crête de coq » : une représentation circulaire mensuelle, où les nombres sont repésentés par des surfaces de secteurs circulaires (figure 23 [8]).

> The lesson is most instructively taught in the Diagram representing the *Causes of mortality* in the Army in the East. It is the same diagram as the former with this difference, that it exhibits, month by month, the mortality

7. Mortality of the British army, *Harrison and Sons, London 1858*

8. Contribution to the sanitary history of the British army during the late war with Russia, *Harrison and Sons, London 1859*

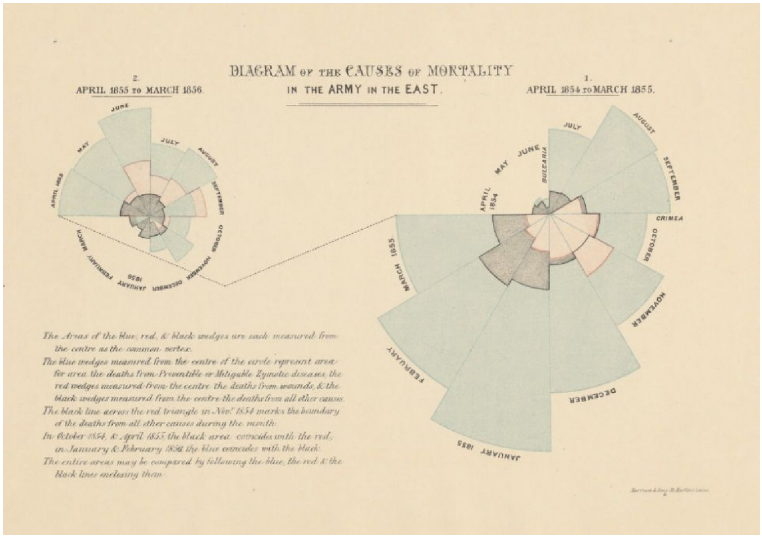

FIGURE 23 – Florence Nightingale : diagramme en crête de coq des causes de mortalité pendant la guerre de Crimée.

> from the diseases and accidents of war occurring during the memorable siege. Each diagram represents the mortality from zymotic disease, from wounds, and from all other causes.
>
> [...]
>
> The large diagram framed on Table III, represents a year's *Mortality in the Hospitals of the Bosphorus*, from October 1, 1854, to September 30, 1855. The long wedge on the right hand of the diagram represents the mortality for the first half of October, 1854, and each of the suceeding wedges denotes by its surface the mortality per 1000 of sick treated for the period stated on them.

Les « zymotic diseases » sont les maladies infectieuses (choléra, typhus, dysenterie). Le constat est accablant : il y a eu en une seule année 11999 décès dus aux maladies infectieuses, contre 1570 résultant de blessures. Florence Nightingale croira toujours au pouvoir de conviction des « hard facts, c'est-à-dire des statistiques. Elle y est aidée par un autre pionnier de l'épidémiologie statistique, le Docteur William Farr [9].

> She found statistics "more enlivening than a novel" and loved to "bite on a hard fact". Dr. Farr wrote in January, 1860 : "I have a New Year's Gift for you. It is in the shape of Tables." "I am exceedingly anxious to see your charming gift," she replied, "especially those returns showing the Deaths, Admissions, Diseases." Hilary Bonham Carter wrote that however exhausted Florence might be, the sight of long columns of figures was "perfectly reviving" to her.

9. Stephen Hollidays : William Farr : campaigning statistician, *London Historians, March 2011*

Les données de mortalité dans les hôpitaux de Crimée sont dans le dataframe `Nightingale` du package `HistData`, qui contient bien d'autres tableaux de données historiques. Les diagrammes en crête de coq peuvent être reproduits avec le package `ggplot2` ; on trouve facilement sur le web les scripts correspondants. Jouez avec, et souvenez-vous quand même que chacune de ces unités statistiques était un jeune homme de votre âge. Florence Nightingale, elle, ne les a jamais oubliés.

> Oh my poor men ; I am a bad mother to come home and leave you in your Crimean graves — 73 per cent in 8 regiments in 6 months from disease alone — who thinks of that now ?

Dans l'imaginaire collectif britannique, elle est restée cette ombre précédée d'une lampe, glissant sans bruit la nuit de lit en lit, réconfortant les mourants par sa seule présence : the « lady with a lamp » du poème de H. W. Longfellow.

> Lo ! in that house of misery
> A lady with a lamp I see
> Pass through the glimmering gloom,
> And flit from room to room.

3.5 A quite unbelievable success story

R est très loin d'être le premier logiciel de statistique, mais il est bien parti pour éclipser ses prédecesseurs. Très tôt, les statisticiens ont saisi l'opportunité de l'informatique, mais ils aussi souhaité s'affranchir des contraintes du codage et de la compilation. Pour répondre aux besoins de champs d'application particuliers se sont développés successivement, parmi les plus connus :

- BMDP (BioMedical Data Processing), à partir de 1957,
- SAS (Statistics Analysis Software, initialement pour les données agronomiques), à partir de 1966,
- SPSS (Statistical Package for the Social Sciences), à partir de 1968.

Même quand des librairies Fortran ont été disponibles, le besoin de les interfacer de manière conviviale a conduit à des langages de plus haut niveau, comme S. Voici comment l'un des créateurs de S en exprime la motivation [10].

> It was the realization that routine data analysis should not require Fortran programs that really got S going. Our initial goals were not lofty ; we simply wanted an interactive interface to the algorithms in the SCS library.

En 1991 à l'Université d'Auckland en Nouvelle Zélande, R. Gentleman et R. Ihaka commencent à développer, sur la même syntaxe que S, un logiciel open source basé sur Lisp. Le nom qu'ils lui donnent, rappelle à la fois S et leurs initiales : R. Le succès est quasi-immédiat [11].

10. R. A. Becker : A brief history of S, *Technical Report, AT&T Bell Laboratories, Murray Hill, N.J., 1994*
11. J. de Leeuw : Statistical software – Overview *UCLA preprint 1096, 2009*

R was written as an alternative implementation of the S language, using some ideas from the world of Lisp and Scheme. The short history of R is a quite unbelievable success story. It has rapidly taken over the academic world of statistical computation and computational statistics, to an ever increasing extend the world of statistics teaching, publishing and real-world application. SAS and SPSS, which initially tended to ignore and in some cases belittle R, have been forced to include interfaces to R, or even complete R interpreters, in their main products.