Domaines : Sciences et Technologies

Parcours : Licence de Mathématiques

Etablissement : Faculté des Sciences (FDS)

Code et Intitulé de l'UE : MTH 229 : TP de Statistique

Crédits : 2

Public cible : Etudiants en Licence de Mathématiques

Semestre : 4

Pré-requis : MTH 160

Enseignant responsable de l'UE :

- GERALDO Issa Cherif, Assistant, Mathématiques, Département de Maths/FDS

Disponibilité :

- Mardi 15h-17h
- Jeudi 10h-12h

Descriptif de l'UE MTH 229

Objectif général

Appliquer les méthodes usuelles de la statistique descriptive et du calcul de probabilités avec le logiciel statistique R.

Objectifs spécifiques : A la fin de cette UE, les étudiants seront capables de :

- Saisir des données dans le logiciel R
- Programmer en R
- Appliquer les méthodes de la statistique descriptive à l'aide du logiciel R
- Implémenter quelques scénarios probabilistes classiques (calcul de probabilités pour les lois usuelles, lancer de dés, simulations simples, etc…)

Bref descriptif de l'UE

Ce cours présente le logiciel statistique R et son utilisation pour mettre en œuvre les méthodes et techniques usuelles en statistique descriptive et en calcul de probabilités.

Contenu d'enseignement

1. Généralités dur le logiciel R
2. Classes d'objets et fonctions usuelles
3. Représentations graphiques
4. Programmation dans R
5. Statistique descriptive avec R
6. Calcul de probabilités et simulations

Plan du contenu d'enseignement

Séance n°	Rappel des objectifs spécifiques	Titres des parties/ chapitres / sous-chapitres
1	Saisir des données dans le logiciel R	Chapitre 1 : Généralités sur le logiciel R Exercices du chapitre 1
2		Chapitre 2 : Classes d'objets et fonctions usuelles
3		Chapitre 2 (suite et fin)
4		Exercices du chapitre 2
5		Exercices du chapitre 2 (suite et fin)
6	Programmer dans R	Chapitre 3 : Représentations graphiques
7		Exercices du chapitre 3
8		Chapitre 4 : Programmation dans R
9		Exercices du chapitre 4

Séance n°	Rappel des objectifs spécifiques	Titres des parties/ chapitres / sous-chapitres
10	Appliquer les méthodes de la statistique descriptive à l'aide du logiciel R	Chapitre 5 : Statistique descriptive
11		Exercices du chapitre 5
12	Implémenter quelques scénarios probabilistes classiques	Chapitre 6 : Calcul de probabilités et simulations Exercices du chapitre 6

Evaluation : 40% DS + 60% Examen

Bibliographie :

[1] F. Bertrand, et M. Maumy-Bertrand (2018). *Initiation à la Statistique avec R*. Dunod, Paris, 3ème édition.

[2] G. Broc (2016). *Stats faciles avec R*. De Boeck Superieur.

[3] P. Lafaye de Micheaux, R. Drouilhet et B. Liquet (2014). *Le logiciel R : Maîtriser le langage - Effectuer des analyses statistiques*. Springer, 2ème édition.

Licence de Mathématiques – Université de Lomé

MTH 229
Initiation au logiciel R

Issa C. GERALDO

Mai 2020

Table des matières

Chapitre 1

Généralités sur le logiciel R

1.1 Présentation

Le logiciel `R` est un logiciel d'analyse statistique et graphique créé par deux statisticiens Ross Ihaka (statisticien néo-zélandais) et Robert Gentleman (statisticien canadien) à l'université d'Auckland (Nouvelle-Zélande) dans le cadre d'un projet de recherche débuté en 1993.

Il est un clone gratuit du logiciel payant `S-plus`, dérivé du langage `S` (Logiciel payant) développé dans les années 1970 par une équipe de chercheurs menée par un statisticien du nom de John M. Chambers. Depuis 1997, son développement et sa distribution sont assurés par plusieurs statisticiens rassemblés dans le **R Development Core Team**.

Le logiciel `R` présente plusieurs avantages : c'est un logiciel **libre** et **gratuit**, très complet, en essor permanent, disponible pour divers environnements : Linux, Windows, Macintosh. C'est un logiciel utilisé en entreprise, dans le monde académique et de la recherche, au sein d'organismes publics ou d'ONG ou encore par les analystes (data miners, data scientists) spécialistes de la fouille de données.

Le site officiel du logiciel `R` est :

https://www.r-project.org/

1.2 Installation

1.2.1 Installation sous Windows

Pour télécharger le fichier d'installation, il faut :
- se rendre sur le site officiel de `R` (https://www.r-project.org/)
- cliquer sur le lien **CRAN** [1] dans le menu **Download** à gauche
- cliquer sur un site miroir de CRAN dans la liste proposée par pays (un site miroir de CRAN est une copie conforme de CRAN accessible à une adresse internet différente) [2]

1. Comprehensive R Archive Network
2. Le logiciel `R` est disponible sur plusieurs serveurs internet dans des pays différents

– une fois le site miroir chargé, cliquer sur le lien **Download R for Windows**
– cliquer sur **Install R for the first time**.

Après téléchargement du fichier d'installation, il faut l'exécuter par double-clic. **Durant l'installation, il ne faut pas oublier de cocher l'option de mettre un raccourci sur le bureau**. Une fois l'installation terminée, le logiciel `R` est accessible (comme tout autre logiciel) dans le menu *Démarrer* ou dans *Tous les programmes* ou encore sur le bureau. Sur le bureau, il est repérable par son icône :

1.2.2 Installation sous Linux

L'installation peut se faire via le **gestionnaire des paquets** en installant les paquets **`r-base-dev`** et **`r-base`**.

1.3 Lancement et fermeture de R

– Pour lancer `R` **sous Windows**, il suffit de double-cliquer sur l'icône

de RGUI (R Graphical User Interface). Il s'ouvre alors une interface graphique contenant une fenêtre de la forme de la figure 1.1.

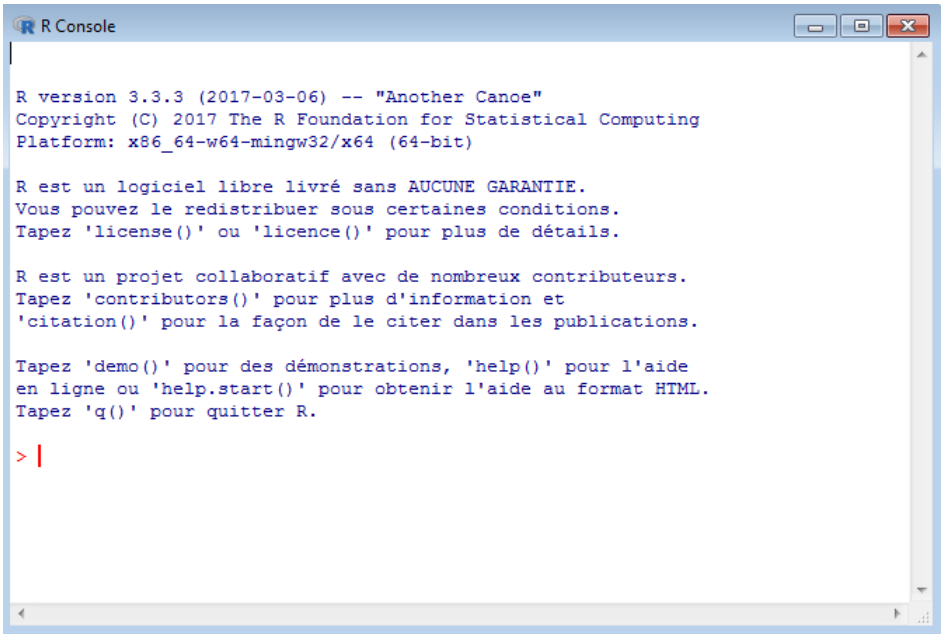


FIGURE 1.1 – Ecran de démarrage de `R`. Le symbole `>`, appelé prompt, signifie que `R` attend une commande.

– **Sous Linux**, l'on peut lancer `R` depuis un terminal Linux en tapant simplement la commande `R` dans le terminal (voir Figure).

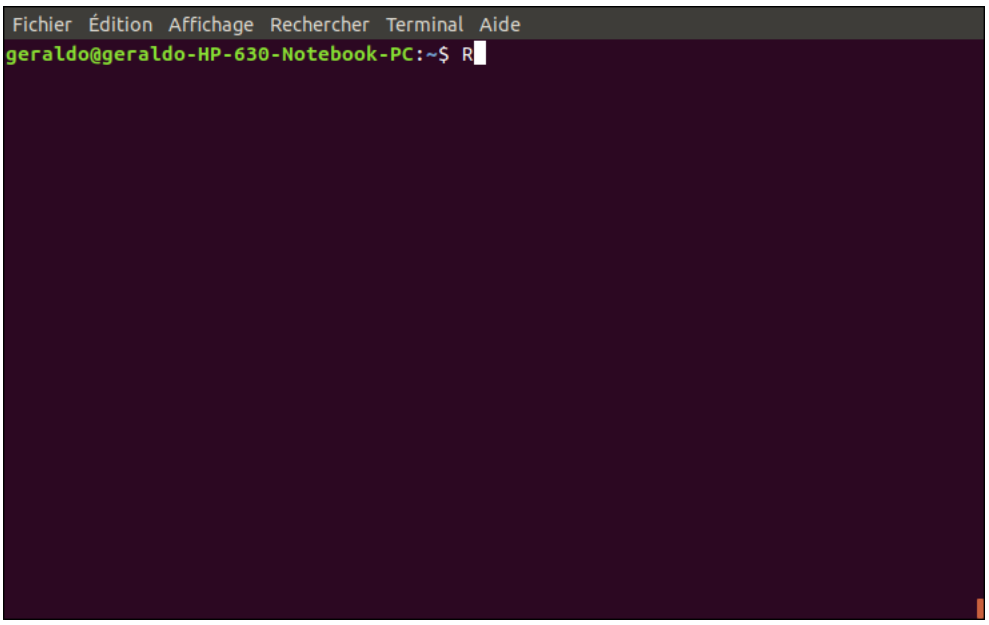


FIGURE 1.2 – Terminal Linux avant lancement de `R`

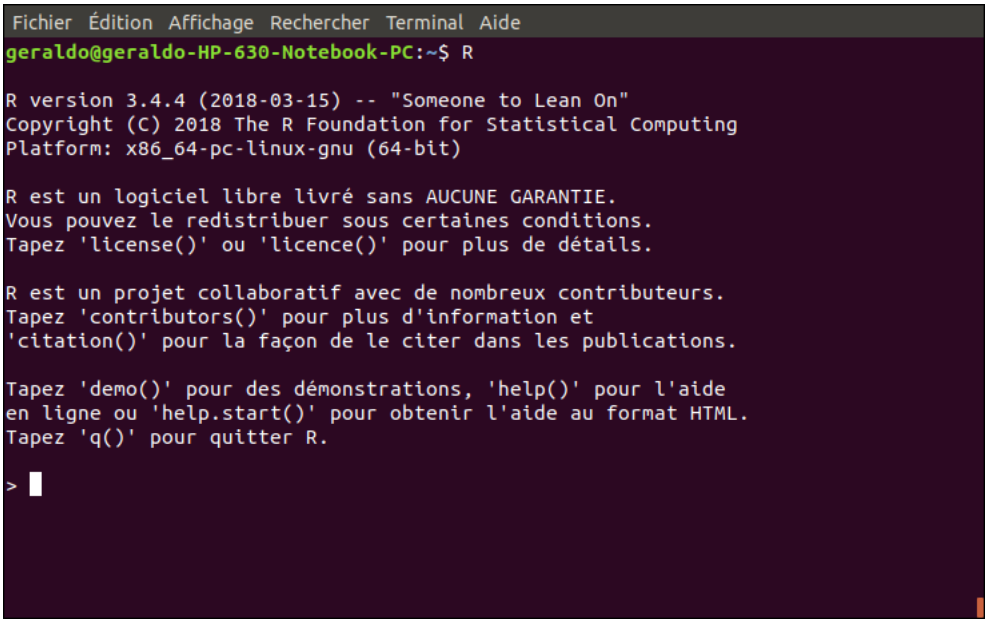


FIGURE 1.3 – Terminal Linux après lancement de `R`

`>`	Le symbole `>` appelé *prompt* indique que `R` est prêt à travailler et attend une commande.

Remarque 1.1. Lorsque le symbole `+` apparaît au lieu de `>`, cela signifie que la commande en cours de saisie est incomplète (par exemple, il manque une parenthèse ou une accolade) et que `R` attend qu'elle soit complétée. On peut alors soit compléter la commande soit appuyer sur la touche *Esc* pour annuler la commande et revenir à `>`.

`quit()` `q()`	Quitter R. Ne pas oublier les parenthèses.
`QUIT()` `Quit()`	Ne fonctionnent pas (`R` est sensible à la casse c'est-à-dire `R` distingue les majusules des minuscules).

1.4 Commandes élémentaires

1.4.1 Création de variables et affichage

`x=2`
`x <- 2`
`2 -> x`
`x = 1 + 1`
	Différentes façons d'affecter la valeur 2 à `x`.

`employe.salaire2 = 100`	Les noms de variables peuvent contenir des lettres, des chiffres, _ mais aussi des `"."`

Remarque 1.2. Même si les noms de variables sont très flexibles dans `R`, il y a quelques règles importantes à respecter :

1) Les noms de variables ne peuvent pas commencer par un chiffre ou un caractère spécial.
2) Comme dit plus haut, `R` est sensible à la casse donc les noms des variables sont aussi sensibles à la casse (les variables `x` et `X` sont deux variables différentes).
3) Quelques noms (comme `c`, `q`, `t`, `C`, `D`, `F`, `I`, `pi`, `T`, etc...) sont déjà utilisés par `R` pour désigner des objets internes et doivent être évités.

`x` `print(x)`	Affiche la valeur de l'objet `x`.
`print("Bonjour")`	Affichage de texte.

1.4.2 Séparation de commandes

`x=2; y=3; z=-1`	Commandes sur la même ligne séparées par un ";"

`x=2`
`y=3`
`z=-1`
	Commandes séparées par un retour à la ligne.

1.4.3 Espace de travail

`objects()` `ls()`	Deux commandes permettant d'obtenir la liste des objets (variables et fonctions) présents en mémoire.
`rm(a,b)`	Supprime les objets `a` et `b` de la mémoire.
`rm(list=ls())`	Supprime tous les objets en mémoire.
`history()`	Historique des commandes entrées dans le terminal dans la session courante.
`getwd()`	Afficher le repertoire courant (le repertoire dans lequel `R` écrit ou lit les fichiers par défaut).
`setwd("C:/Users/Toto/")`	Changer le repertoire courant Remarquer l'utilisation de "/" et non de "\".
`save.image("samedi.RData")`	Sauvegarde l'espace de travail (objets en mémoire) dans le fichier `samedi.RData` (dans le repertoire courant) Extension `RData` non obligatoire mais à utiliser de préférence (convention).
`load("samedi.RData")`	Charge un espace de travail sauvegardé. Comme aucun repertoire n'est précisé, `R` cherchera le fichier dans le repertoire courant.

1.4.4 Documentation de R

`help.start()`	Lance dans le navigateur Web par défaut la documentation HTML de R. Accessible hors ligne.
`help(solve)` `?solve`	Deux commandes équivalentes pour afficher l'aide HTML sur la fonction `solve`.
`help.search("bin")` `??bin`	Affiche la liste de toutes les pages de la documentation de R contenant le mot "bin".
`apropos("bin")`	Affiche la liste de tous les objets dont le nom contient "bin".

1.5 Deux façons d'exécuter des commandes

1) Taper les commandes dans l'invite de commandes de R.

2) Créer un fichier de commandes et l'exécuter :

`source("somme.R")`	Exécute les commandes contenues dans le fichier `somme.R`

Par défaut, R cherchera le fichier dans le repertoire courant (résultat de la commande `getwd()`). Si le fichier à exécuter se trouve dans un autre repertoire, il faudra donc au préalable utiliser la commande `setwd()` pour indiquer à R le repertoire dans lequel se trouve ledit fichier.

`note=10 # Valide de justesse`	Le symbole `#` pour les commentaires **sur une seule ligne** dans un fichier de commandes.

1.6 Bibliothèques ou packages

Une bibliothèque (ou package) est un ensemble de fonctions et de données permettant de faire des tâches bien précises. Par exemple la bibliothèque `stats` contient un ensemble de fonctionnalités permettant de faire des statistiques. Il existe deux types de bibliothèques :

- les bibliothèques de base qui sont déjà présentes dans l'installation de base de R : seules certaines d'entre elles sont chargées automatiquement au démarrage de R (exemple des bibliothèques `base` et `stats`).
- d'autres bibliothèques écrites par d'autres utilisateurs de R disponibles sur le site du CRAN (Comprehensive R Archive Network) : https://cran.r-project.org/web/packages/index.html.

A la date de la dernière mise à jour de ce cours (mai 2020), le site du CRAN compte un peu plus de 15 600 packages.

`library()`	Liste toutes les bibliothèques disponibles localement sur la machine où `R` est exécuté.
`library(stats)`	Chargement d'une bibliothèque. Cette commande est obligatoire pour utiliser les fonctionnalités (fonctions, données) d'une bibliothèque (ici stats).
`citation('stats')`	Fournit de l'aide sur la façon de citer la bibliothèque `stats` dans une publication.
`help(package=stats)`	Documentation du package `stats`.
`help("solve", try.all.packages = TRUE)`	Par défaut, la fonction `help` ne recherche que dans les packages chargés en mémoire. L'option `try.all.packages=TRUE` permet de chercher dans tous les packages disponibles sur la machine.
`search()`	Liste des bibliothèques chargées.
`install.package(nom)`	Télécharge la dernière version de la bibliothèque `nom` et l'installe.
`old.packages()`	Indique les packages dont une version plus récente (que celle installée sur la machine) est disponible sur le site du CRAN.
`update.packages()`	Propose de télécharger et installer de tels packages.

A l'exécution de la commande d'installation d'un package, `R` peut demander de spécifier un miroir. Les miroirs[3] sont des serveurs situés à plusieurs endroits dans le monde et disposant des mêmes fichiers que ceux téléchargeables sur le site du CRAN.

1.7 Exercices

Exercice 1.1. Ecrire et exécuter un code `R` qui :
- efface touts les objets en mémoire
- définit deux variables x et y de valeurs respectives 2 et -5
- affiche x et y

3. La liste des serveurs est disponible à l'adresse http://cran.r-project.org/mirrors.html

– permute les valeurs de x et y et affiche ensuite les nouvelles valeurs.
– affiche le message "Fin" pour informer l'utilisateur de la fin du travail.

Exercice 1.2. Utiliser l'aide pour comprendre l'utilisation des commandes suivantes :

1) `savehistory()`

2) `.Last.value`

3) `round`

4) `floor`

Chapitre 2

Classes d'objets et fonctions usuelles

2.1 Introduction

- Tous les objets `R` (variables, fonctions) ont une classe et les manipulations possibles sur un objet donné dépendent de sa classe.
- Les classes d'objets usuelles :
  `numeric`, `logical`, `character`, `list`, `matrix`, `factor`, `data.frame`, `function`.

`x=2; class(x)`	Affichage de la classe de l'objet `x`, ici `"numeric"`.

2.2 Vecteurs numériques (classe `numeric`)

2.2.1 Valeurs particulières

`NA`	Valeur manquante (Not Available).
`pi`	Variable interne contenant la valeur de π
`1/0 ; -1/0`	Respectivement `Inf` et `-Inf`.
`0/0 ; Inf-Inf`	`NaN` (*Not a Number*).

Nombres complexes

`sqrt(-1)`	Affiche `NaN`.
`sqrt(-1+0i)`	Affiche `0+1i`.

2.2.2 Création

`x = c(1,3,4,-5)`	Vecteur (1,3,4,-5) affecté à `x`. La lettre `c` pour *collect* ou *combine*.

`1:10`
`seq(from=1,to=10,by=1)`
 	
`seq(length=10,from=1,by=1)`
	Vecteur $(1, 2, \ldots, 9, 10)$.
	Idem (`by` représente l'incrément ou le pas).
	Idem.

`x=c(1,2);`
`c(x,x,x); rep(x,times=3)`
 	
`rep(x,each=3)`
 	
`rep(x,times=c(2,3))`
	Commandes équivalentes : `x` est mis bout à bout 3 fois. Le résultat sera $(1, 2, 1, 2)$.
	Répète 3 fois chaque composante de `x` puis passe à la composante suivante. Le résultat sera $(1, 1, 1, 2, 2, 2)$.
	Répète 2 fois la première composante et 3 fois la seconde. Le résultat sera $(1, 1, 2, 2, 2)$.

2.2.3 Opérations et fonctions

2.2.3.1 Opérations et fonctions usuelles

`x=c(1,3,4,-5); y=c(0,2,3,0)`
`z=x+y`
`z=x*y`
`z=x/y`
`z=2*x`
	Vecteurs de **même longueur**.
	Addition composante par composante.
	Idem pour la multiplication.
	Idem pour la division.
	Multiplication de toutes les composantes par 2.

`x+2`	Ajoute 2 à toutes les composantes de `x`.

`abs(x); x^2; x^(1/3); sqrt(x);`
`exp(x); log(x); sin(x); cos(x);`
`tan(x); sinh(x); cosh(x);`
`tanh(x); ...`
	Ces fonctions usuelles s'appliquent à toutes les composantes de `x` c'est-à-dire elles retournent un vecteur de même taille que `x` dont les composantes sont les images respectives de celles de `x`.

`factorial(x)`
 	
`choose(5,3)`
	Calcule les factorielles des éléments du vecteur `x`
	C_5^3

`length(x)` `max(x) ; min(x) ; range(x)`	Nombre de composantes. Maximum, minimum, vecteur des 2 valeurs

`sum(x) ; mean(x)`	Somme des composantes, moyenne.
`sort(x)` `sort(x, decreasing=TRUE)`	Tri par ordre croissant. Tri par ordre décroissant.
`rev(x)`	Renverse le vecteur `x`. Résultat : $(-5, 4, 3, 1)$.
`round(x); round(x,2)`	Arrondi à l'entier le plus proche ; arrondi à la seconde décimale.
`ceiling(x)`	Plus petit entier supérieur.
`floor(x)`	Fonction partie entière (plus grand entier inférieur).
`trunc(x)`	Troncature des décimales.

2.2.3.2 Nécessité de l'utilisation de parenthèses

`1:2-1 ; 1:(2-1)`	Respectivement (0, 1) et 1.
`2^2^3 ; (2^2)^3`	Respectivement $2^8 = 256$ et $4^3 = 64$.

2.2.4 Composantes

`x[1]`
`x[1:4]`
`x[length(x)]`
`x[c(2,3)]`
`x[-c(1,3)]`
	1ère composante.
	Quatre premières composantes.
	Dernière composante.
	2ème et 3ème composantes.
	Toutes les composantes sauf la première et la troisème.

`head(x,2)`	Extraction des 2 premiers éléments.
`tail(x,2)`	Extraction des 2 derniers éléments.
`x[x>2]` `subset(x,x>2)`	Commandes équivalentes permettant de récupérer les composantes de `x` strictement supérieures à 2.
`x[2]=-3`	Affectation.
`x[x>2]=1`	Remplace toutes les composantes de `x` strictement supérieures à 2 par 1.

2.2.5 Recherche d'éléments dans un vecteur

`x=c(3,-1,0,10,12); which(x>=10)`	Positions des éléments de `x` supérieurs ou égaux à 10. Retourne un vecteur de deux éléments 4 et 5.
`which.min(x)`	Position du minimum. Retourne 2.
`which.max(x)`	Position du maximum.
`match(c(20,30),x)`	Positions du premier 20 et du premier 30 dans `x`.

2.2.6 Opérations entières

`5%%2`	Reste de la division euclidienne de 5 par 2. Retourne 1.
`5%/%2`	Quotient de la division euclidienne de 5 par 2. Retourne 2.

2.3 Matrices (classe `matrix`)

2.3.1 Création et accès aux composantes

`A=matrix(0,nrow=2,ncol=3)` `A=matrix(0,2,3)`	Matrice à 2 lignes et 3 colonnes composée de 0.
`A=matrix(1:6,nrow=2,ncol=3)`	Matrice à 2 lignes et 3 colonnes dont les composantes sont les nombres de 1 à 6. Par défaut, la matrice est remplie colonne par colonne i.e. $$A = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix}.$$
`A=matrix(c(-1,3,4,-2,0,7),` `nrow=2, ncol=3, byrow=TRUE)`	L'argument `byrow=TRUE` permet de spécifier que le remplissage doit se faire suivant les lignes. $$A = \begin{pmatrix} -1 & 3 & 4 \\ -2 & 0 & 7 \end{pmatrix}.$$

`nrow(A) ; ncol(A)`	Nombres de lignes et de colonnes respectivement.
`dim(A)`	Un vecteur de deux composantes qui sont respectivement le nombre de lignes et le nombre de colonnes de `A`.

`A[1,3]`
 	
`A[,3]`
`A[1,]`
`A[1:2,2:3]`
	Elément de la 1ere ligne et de la 3e colonne.
	Toute la 3e colonne.
	Toute la 1ère ligne.
	Sous-matrice composée des deux 1ères lignes et des deux dernières colonnes.

`rownames(A); colnames(A)`
`rownames(A)=c("Jean","Afi")`
`colnames(A)=c("Alg1","Ana1","BSI")`
`A[1,3] ; A["Jean","BSI"]`
	Noms de lignes et de colonnes.
	Nomme les lignes.
	Nomme les colonnes.
	Commandes équivalentes.

2.3.2 Opérations et fonctions matricielles

`A+B ; A*B ; A^2`	Opérations termes à termes : `A` et `B` doivent être de même dimension.
`A %*% B`	Produit matriciel (nécessite que le nombre de colonnes de A soit égal au nombre de lignes de B).
`t(A)`	Transposée.

`diag(v)`
 	
`diag(A)`
 	
`diag(k)`
	`v` est un vecteur : donne une matrice diagonale avec `v` sur la diagonale.
	`A` est une matrice : extrait la diagonale de `A` sous forme d'un vecteur.
	`k` est un nombre : donne la matrice identité d'ordre `k`.

`det(A)`	Déterminant de A.
`solve(A)`	Inverse de A.
`solve(A,B)`	Résolution de l'équation $AX = B$ où A est une matrice carrée d'ordre n inversible et B est un vecteur de taille n ou une matrice à n lignes.

`eigen(A)`
 	
`eigen(A)$values`
`eigen(A)$vectors`
	Liste des vecteurs propres et valeurs propres de A.
	Valeurs propres de A.
	Vecteurs propres de A.

`apply(A,dim,f)`
 	
`apply(A,1,mean); rowMeans(A)`
`apply(A,2,sum); colSums(A)`
	Applique la fonction `f` à chaque ligne de A si `dim=1` et chaque colonne si `dim=2`.
	Moyennes par ligne.
	Somme par colonnes.

`addmargins(A)`	Rajoute à la matrice `A` des marges contenant respectivement les sommes par lignes et par colonnes.
`rbind(A,B,C)` `cbind(A,B,C)`	Met A, B et C bout à bout verticalement. Met A, B et C bout à bout horizontalement.
`outer(A,B,"*"); outer(A,B,"+")`	Produit cartésien des vecteurs numériques A et B et application des opérateurs * et +.

2.4 Vecteurs logiques (classe `logical`)

`y=(x>3)`
 	
`x>=3 ; x==3 ; x!=3`
	y est un vecteur de même longueur que `x` (vecteur numérique) avec des `TRUE` correspondant aux composantes de `x` strictement plus grandes que 3 et des `FALSE` ailleurs.
	Autres exemples : inégalité large, égalité, différent.

`y1 & y2`
`y1 | y2`
`!y1`
`xor(y1,y2)`
	Conjonction (« et » logique).
	Disjonction (« ou »).
	Négation.
	« ou » exclusif.

`x[x>3]`	Vecteur numérique composé uniquement des composantes de `x` strictement supérieures à 3.

`(x>3)+5`
 	
`as.numeric(TRUE)`
`as.numeric(FALSE)`
	Les `TRUE` et les `FALSE` deviennent respectivement 1 et 0 pour y ajouter 5.
	Retourne 1.
	Retourne 0.

Quelques fonctions logiques

`is.na(x)`	Retourne un vecteur ou une matrice de même taille que `x` indiquant pour chaque composante de `x` s'il s'agit d'une valeur manquante.
`is.vector(x), is.matrix(x), is.data.frame(x)`	L'objet `x` est-il un vecteur ? une matrice ? un tableau de données ? Ces fonctions retournent une seule valeur logique.
`60%in%x; is.element(60,x)`	Commandes équivalentes pour tester si 60 est une composante de `x`.
`any(x>2)`	Retourne `TRUE` si au moins un élément de `x` est strictement supérieur à 2 et `FALSE` sinon.
`all(x>2)`	Retourne `TRUE` si tous les éléments de `x` sont strictement supérieurs à 2 et `FALSE` sinon.
`exists("y")`	Retourne `TRUE` si la variable `y` existe et `FALSE` sinon.

2.5 Vecteurs de chaînes de caractères (classe `character`)

`y = c("Jean","Louise")`	`y` est un vecteur composé de deux chaînes de caractères. Ne pas oublier l'emploi des `""`.
`paste("bon","jour",sep="")` `paste("bon","jour",sep=" ")`	Donne la chaîne "bonjour". Donne la chaîne "bon jour".

`eleves = c("Jean","Louise")`
`notes = c(9,14)`
`names(notes)=eleves`
 	
`notes["Louise"]; notes[2]`
	Chaque composante de `notes` est associée à un nom d'élève.
	La 2e note, soit 14.

`as.character(10)`	Le nombre 10 devient la chaîne de caractères "10".
`nchar("Bonjour")`	Nombre de caractères de la chaîne : ici 7.

`toupper("bon"); tolower("BON")`	Conversion en majuscules / minuscules.

`z=c("A","B","C","A","A","B","B")` `table(z)`	Tableau de contingence donnant le nombre d'apparitions de chaque modalité. Le résultat est un objet de classe `table`.

Le résultat est :

```
z
A B C
3 3 1
```

`unique(z)`	Retourne un objet de même classe que `z` en enlevant les doublons.

Vecteurs constants de chaînes de caractères internes à R. R contient des vecteurs de chaînes de caractères sous forme de constantes internes. Parmi elles, on peut citer les suivantes.

`LETTERS`	Les 26 lettres de l'alphabet en majuscules.
`letters`	Les 26 lettres de l'alphabet en minuscules.
`month.name`	Les noms en anglais des 12 mois de l'année.
`month.abb`	Les abréviations de 3 lettres des noms en anglais des 12 mois de l'année.

2.6 Listes (classe `list`)

Une liste est une collection ordonnée d'objets qui peuvent être de différentes classes. C'est donc un outil très pratique pour disposer, sous la forme d'une seule variable de type liste, de plusieurs variables de différents types. Les listes présentent en particulier de l'intérêt lors de la création et l'utilisation des objets de type fonctions que nous verrons plus tard.

`etud=list(nom="Kodjo", age=21, genre="M")`	Création d'une liste à trois composantes de différentes classes : les composantes `nom` et `genre` sont de classe `character` alors que `age` est de classe `numeric`.
`names(etud)`	Noms des composantes de la liste `etud`.
`etud[[2]] ; etud$age; etud[["age"]]`	Commandes équivalentes pour obtenir la seconde composante `age`.

2.7 Tableaux de données (classe `data.frame`)

Les tableaux de données (ou structures de données hétérogènes) ou `data.frame` sont des tableaux dont les lignes sont des listes de mêmes composantes et dont les colonnes peuvent être de différents types (numérique, logique, chaînes de caractères). Ils se manipulent de façon très semblable aux matrices. Ce type de structure sert à stocker des informations (en colonnes) relatives aux individus (en lignes).

2.7.1 Création par lignes de commandes

`x=data.frame(nom=c("Afi","Koffi"), age=c(18,23), male=c(FALSE,TRUE))`	Création d'un tableau de données de dimension 2×3 avec des chaînes de caractères en 1ère colonne, du numérique en seconde et des booléens en dernière.

A l'affichage (en faisant `print(x)`), on obtient

```
    nom age  male
1   Afi  18 FALSE
2 Koffi  23  TRUE
```

`x=edit(data.frame())`	Ouvre le tableur interne de `R` permettant de saisir les données et de les enregistrer.

2.7.2 Jeux de données internes

Ce sont des jeux de données installées avec `R`, accessibles à chaque lancement de `R` et qui n'ont pas besoin d'être créés par l'utilisateur.

`data()`	Liste tous les jeux de données disponibles.
`data(iris)`	Chargement d'un jeu de données Charge le jeu de données `iris` : la variable `iris` est maintenant un objet de l'espace de travail.

2.7.3 Importation de données

On peut aussi importer des tableaux de données dans `R` depuis des fichiers texte ou d'autres types de fichiers tels que les fichiers `csv`, `xls` et `xlsx`.

2.7.3.1 Fichiers de texte

De préférence, un fichier de texte simple doit avoir une des trois extensions **.txt**, **.rda** ou **.data**. Le fichier de données doit être organisé de la façon suivante :

- La première ligne peut contenir ou non les noms des composantes ou variables.
- Chaque ligne suivante correspond à un nouvel individu décrit par les variables précédentes. Le début de la ligne peut contenir ou non l'identifiant de cet individu.
- Les colonnes sont séparées par un espace ou par un autre séparateur.

`maisons.data`

	Loyer	Cuisine	Nb.chambres	Quartier
01	25000	TRUE	2	Agoe
02	35000	TRUE	2	Agoe
03	40000	TRUE	2	Kegue
04	10000	FALSE	1	Be
05	15000	FALSE	1	Totsi
06	30000	TRUE	1	Agoe
07	12000	FALSE	1	Totsi

`x=read.table("maisons.data", header=TRUE)`	La variable `x` est de classe `data.frame` avec les noms de colonnes correspondants.
`xnew = edit(x)`	Modification de `x` dans un tableur.
`View(x)`	Affichage de `x` dans le tableur interne de `R`.

2.7.3.2 Autres types de fichiers

- Lecture de fichiers `csv` :
  - fonction `read.csv` pour les fichiers CSV avec « . » comme séparateur décimal et « , » comme séparateur de colonnes ;

  - fonction `read.csv2` pour les fichiers CSV avec « , » comme séparateur décimal et « ; » comme séparateur de colonnes.
- Lecture de fichiers `xls` ou `xlsx` : l'on peut utiliser les fonctions `read.xlsx` et `read.xlsx2` de la librairie `xlxs` (à télécharger et installer avant le premier chargement).

Le lecteur est invité à consulter l'aide pour plus de détails sur ces fonctions.

2.7.4 Manipulation

`x[["Loyer"]]; x[[1]] ; x$Loyer ; x[,1]`	Commandes équivalentes.

`names(x)`
`attach(x)`
 	
`detach(x)`
	Noms des composantes (colonnes) de `x`. Les composantes `Prix`, `Cuisine`, `Nb.chambres` et `Quartier` sont disponibles directement sous leur nom sans ajouter le préfixe `x`. Par exemple, `Loyer` est maintenant équivalent à `x$Loyer`. Opération inverse.

`tapply(x$Loyer, x$Quartier, mean)`	Calcul de la moyenne des loyers en découpant suivant les différents quartiers.
`transform(x,Loyer.K=Loyer/1000)`	Crée une copie de `x` en ajoutant une nouvelle colonne `Loyer.K` telle que définie.

2.8 Facteurs (classe `factor`)

Cette classe d'objets permet de saisir des données concernant une variable statistique qualitative ayant plusieurs modalités. Cette structure va au-delà du simple vecteur de caractères puisqu'elle est conçue pour gérer de façon spécifique les données qualitatives (ou catégorielles).

`x=factor(c("pos","neg","pos"))`	Création d'un vecteur de facteurs avec 3 composantes et 2 modalités "pos" et "neg".
`levels(x)`	Liste des niveaux (ou modalités) du facteur.
`summary(x); table(x)`	Deux commandes équivalentes permettant d'afficher les modalités et le nombre de fois que chacune apparaît.
`m=c("a","b","a","a")` `as.factor(m)`	Transformation d'un vecteur de chaînes de caractères en un vecteur de facteurs.

2.9 Exercices

Exercice 2.1.

1) Créer le vecteur `v1a` de composantes -1, 2.5, 3, -4, 6, 15, -4 et le vecteur `v1b` formé des entiers de 1 à 5. Créer, sans ressaisir les composantes, le vecteur `v1` composé des 5 premières composantes de `v1a` et de celles de `v1b`.
2) Créer le vecteur `v2` composé de la suite des nombres allant de -3π à 3π par pas de $\pi/2$. Calculer leurs cosinus respectifs dans un vecteur `v2.cos`.
3) Créer le vecteur `v3` contenant tous les multiples de 2 compris entre 2 et 50. Combien y en a-t-il ?
4) Créer le vecteur `v4` contenant tous les multiples de 3 compris entre 2 et 50. Combien y en a-t-il ?
5) En utilisant les fonctions `intersect` et `setdiff`, créer le vecteur `v5` des multiples communs à 2 et 3 et le vecteur `v6` des multiples de 3 qui ne sont pas multiples de 2.

Exercice 2.2.

1) Construire, en une seule commande, la matrice M suivante :

$$M = \begin{pmatrix} 2 & 4 & 1 & 2 \\ 3 & 4 & 5 & 6 \\ 7 & 8 & 9 & 10 \\ 11 & 12 & 35 & 7 \end{pmatrix}$$

2) Afficher la valeur de :
   (a) l'élément situé à l'intersection de la première ligne et de la troisième colonne ;
   (b) la première ligne de M ;
   (c) la troisième colonne de M ;
   (d) la sous-matrice obtenue après avoir enlevé la première ligne et la première colonne de M ;
   (e) la matrice M bordée par les sommes des lignes et les sommes des colonnes ;
   (f) la somme des éléments de M par lignes ;
   (g) la sous-matrice obtenue en ne gardant que les lignes dont la somme est supérieure ou égale à 25.

Exercice 2.3. Créer en un minimum de commandes :

1) La table de multiplication de 1 à 9 (penser à la commande `outer`).
2) Une matrice `A` d'ordre 10×10 avec des 1 partout sauf des 0 sur la diagonale (une ligne de commande suffit).
3) Une matrice `B` d'ordre 10×10 avec les nombres 1 à 199 de deux en deux et répartis de façon croissante colonne par colonne.
4) La matrice `B2` issue de `B` où les multiples de 3 sont remplacés par la valeur 0. Combien y avait-il de multiples de 3 ?

5) Une liste `L` à deux composantes : la 1ère, nommée `petit`, contient les valeurs strictement inférieures à la moyenne des valeurs du vecteur `x` , la 2nde, nommée `grand`, contient les autres valeurs.

Exercice 2.4.

1) Importer dans votre session `R` le jeu de données `precip`. Ce jeu de données est pré-enregistré dans le logiciel `R` et regroupe des données de précipitations dans différentes villes américaines.
2) De quelle classe de données s'agit-il ?
3) Donner la liste des villes pour lesquelles des mesures sont disponibles, combien y en a-t-il ?
4) Quel est le niveau de précipitation des villes suivantes : Philadelphia, Columbia, Baltimore, Sacramento ?

Exercice 2.5. On considère le tableau de données suivant :

Loyer	Cuisine	Nb.chambres	Quartier
25000	TRUE	2	Agoe
35000	TRUE	2	Agoe
40000	TRUE	2	Kegue
10000	FALSE	1	Be
15000	FALSE	1	Totsi
30000	TRUE	1	Agoe
12000	FALSE	1	Totsi

1) Saisir les données à l'aide du tableau de données `maisons`.
2) Calculer (à l'aide de commandes `R` appropriées) le loyer moyen, le loyer moyen par quartier, le nombre de quartiers concernés et le nombre d'appartements avec cuisine interne.

Exercice 2.6. Répondre aux questions ci-après en utilisant des commandes `R`.

1) Importer dans votre session `R` le jeu de données `WorldPhones`.
2) Afficher son contenu. Quelle est sa classe ?
3) Calculer le nombre total de téléphones pour chaque année (vecteur `nbrTelAn`).
4) Quel est le continent qui a le plus / moins de numéros attribués en 1961 ?
5) En 1961, dans combien de continents y a-t-il plus de : 20 millions de téléphones ?
6) En quelle année, le nombre total de téléphones a-t-il dépassé 120 millions pour la première fois ?
7) Pour chaque année, calculer la proportion qu'occupe le nombre de téléphones de l'Afrique.

Exercice 2.7.

1) Charger le jeu de données `airquality`. Que représente ce jeu de données ?
2) Quelle est la classe de l'objet obtenu ? Quelles sont ses composantes (colonnes) ?
3) Donner le nombre de valeurs manquantes (indiquées `NA`) pour chaque composante.

4) Créer, à partir de `airquality`, un tableau de données `airquality2` dont les températures seront converties en degrés Celsius. On rappelle la formule de conversion de Fahrenheit en Celsius : $^{\circ}\mathrm{C} = (^{\circ}\mathrm{F} - 32)/1.8$.

5) Calculer, en degrés Celsius, les températures moyennes, minimales et maximales pour chacun des mois de mai, juin, juillet, août et septembre.

6) Calculer la concentration moyenne d'ozone.

7) Calculer les concentrations d'ozone moyennes, minimales et maximales pour chacun des mois de mai, juin, juillet, août et septembre.

Chapitre 3

Fonctionnalités graphiques

3.1 Généralités

Les fonctionnalités graphiques sont une composante extrêment importante du système R. Il existe trois (03) types de commandes graphiques :

- Les fonctions **haut niveau** qui permettent de créer un nouveau graphique.
- Les fonctions **bas niveau** qui permettent d'ajouter des informations sur un graphique existant.
- Les fonctions **interactives** (non vues dans ce cours) qui permettent d'interagir sur le graphique afin d'ajouter ou d'en extraire des informations à l'aide de la souris.

3.2 Gestion de la fenêtre graphique

3.2.1 Création et fermeture

`x11() ; X11() ; windows();` `dev.new()`	Commandes équivalentes pour ouvrir une nouvelle fenêtre graphique vierge.

`dev.list()`
 	
`dev.set(1)`
	Vecteur contenant les numéros des fenêtres graphiques ouvertes.
	La fenêtre graphique numéro 1 est activée.

`dev.off(1)`
`dev.off()`
`graphics.off()`
	Ferme la fenêtre graphique numéro 1.
	Ferme la fenêtre graphique active.
	Ferme toutes les fenêtres graphiques.

3.2.2 Fractionnement d'une fenêtre grapique

La figure 3.1 donne un exemple de fenêtre graphique fractionnée.

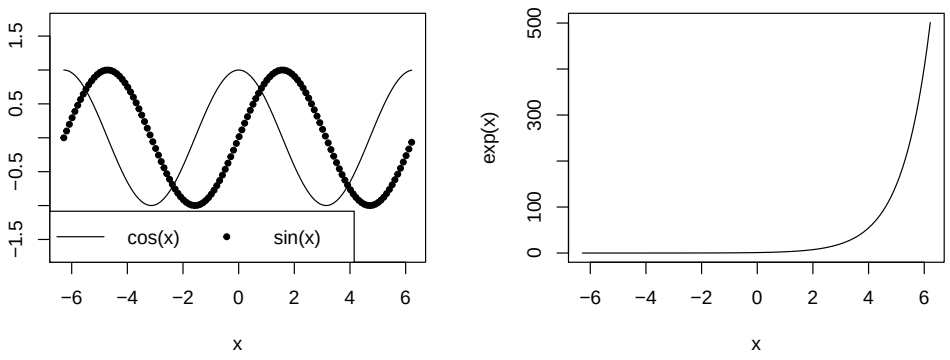


FIGURE 3.1 – Exemple de fenêtre graphique fractionnée

`split.screen(c(2,3))`	Fractionnement de la fenêtre graphique active en 6 graphiques (2 lignes et 3 colonnes) numérotées par lignes.
`screen(5)`	Active le 5ème graphique de la fenêtre active (donc celui en 2ème ligne et 2ème colonne).

3.2.3 Sauvegarde d'une fenêtre graphique dans un fichier

`dev.size("px")`	Donne les dimensions en pixels de la fenêtre graphique active.
`dev.print(pdf, "essai.pdf", width=6, height=4)`	Sauvegarde la fenêtre graphique active dans le fichier pdf "essai.pdf" avec les dimensions en pouces (1 pouce = 2.54cm). Le paramètre `pdf` peut être remplacé par `postscript` (fichier "essai.ps").
`dev.print(png, "essai.png", width=700, height=600)`	Sauvegarde la fenêtre graphique active dans le fichier png "essai.png" avec les paramètres spécifiés. Il faut préciser les dimensions (l'unité est le pixel). Le paramètre `png` peut être remplacé par `jpeg` (fichier "essai.jpg").

3.3 Fonctions haut niveau

Les fonctions haut niveau créent un nouveau graphique dans la fenêtre graphique active, éventuellement avec des caractéristiques complémentaires (axes, labels, titres, etc.). Si un graphique existait déjà dans la fenêtre, alors il est effacé.

3.3.1 Commande générique : la commande `plot`

`plot(x)`	`x` vecteur numérique : cette commande produit le nuage de points (i, x_i) avec les composantes de `x` en ordonnées et les index des composantes de `x` en abscisses.
`plot(x,y)`	`x` et `y` sont des vecteurs numériques de même longueur : cette commande produit le nuage de points (x_i, y_i) avec les composantes de `x` en abscisses et celles de `y` en ordonnées.
`plot(M)`	`M` matrice deux colonnes : cette commande produit un nuage de points avec les composantes de la 1ère colonne en abscisses et celles de la 2ème colonne en ordonnées.
`x=seq(-2,5,0.1)` `plot(x,exp(x))`	Représentation graphique de la fonction exponentielle à l'aide d'un nuage de points sur l'intervalle $[-2, 5]$.

Amélioration. Par défaut, la commande `plot` représente un nuage de points où chaque point est représenté par un cercle. Il est possible d'améliorer l'aspect du graphique en spécifiant la valeur de certains paramètres.

`plot(x,y,type="p",pch="+")`	Nuage de points avec le symbole "+".
`plot(x,y,type="l")`	Points reliés par des lignes.
`plot(x,y,type="o")`	Nuage de points représentés par des cercles et reliés par des lignes.
`plot(x,y,type="n")`	N'affiche rien mais prépare le graphique (création, axes) : utile pour les fonctions bas niveau.
`plot(x,y, axes=FALSE)`	Graphique sans axes.
`plot(x,y, main="Titre", xlab="X", ylab="Y")`	Titre du graphique, noms des axes.
`plot(x,y,col="blue")`	Gestion de la couleur.
`plot(x,y, xlim=c(-1,3), ylim=c(0,2))`	Les paramètres `xlim` et `ylim` permettent de délimiter l'axe des abscisses et l'axe des ordonnées.
`plot(x,y, lwd=2)`	Le paramètre `lwd` (line width) permet de gérer l'épaisseur des traits.
`help("par")`	Consulter l'aide pour plus de détails sur les paramètres graphiques.
`colors()`	Affiche la liste des couleurs disponibles.

3.3.2 Autres fonctions

Il existe d'autres fonctions haut niveau dont nous présenterons certaines dans les chapitres suivants concernant spécifiquement la statistique et la probabilité.

3.4 Fonctions bas niveau

Les fonctions bas niveau ajoutent des informations (points, lignes, légendes, titres) à un graphique déjà créé par une fonction haut niveau sans effacer ledit graphique. Il faut donc avoir déjà utilisé une fonction haut niveau auparavant.

3.4.1 Ajout de points, de lignes ou de symboles

`points(x,y,pch="+")`	Commande similaire à `plot(x,y, type="p", pch="+")` mais sans effacer le graphique.
`lines(x,y)`	Commande similaire à `plot(x,y, type="l")` mais sans effacer le graphique.
`lines(x,y,col="blue")`	Idem en couleur bleue.
`lines(x,y,lty="dotted")`	Idem en pointillés.
`abline(b=2,a=-3)`	Rajoute la droite d'équation $y = bx + a$ sur la fenêtre graphique active.
`segments(x0, y0, x1, y1)`	Trace une ligne du point `(x0,y0)` au point `(x1,y1)`.

3.4.2 Ajout de texte

`text(x,y,noms)`	En chaque position (x_i, y_i), affiche la composante correspondante du vecteur de chaînes de caractères `noms`.
`title("Titre du graphique")`	Titre du graphique.
`legend("topright", legend=c("courbe 1","courbe 2"), col=c("blue","red"), lty=c(1,2))`	Légende placée en haut à droite avec les deux libellés courbe 1 et courbe 2, le 1er associé à une ligne continue bleue, le 2nd à une ligne pointillée rouge.
`help(plotmath)`	Afficher l'aide sur l'ajout de symboles mathématiques (racine carrée, exposants, etc...) sur un graphique.

3.5 Exercices

Exercice 3.1. Représenter chacune des fonctions suivantes sur l'intervalle mentionné :

$$f : x \longmapsto e^x, \qquad I = [-3, 4]$$

$$g : x \longmapsto \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}, \qquad I = [-4, 4]$$

$$h : x \longmapsto \log(x), \qquad I =]0, 10]$$

Renseignez à chaque fois les noms des axes ainsi que le titre du graphique.

Exercice 3.2. Représenter les fonctions sinus et cosinus sur $[-2\pi, 2\pi]$ superposées dans le même graphe à la manière de la figure 3.2.

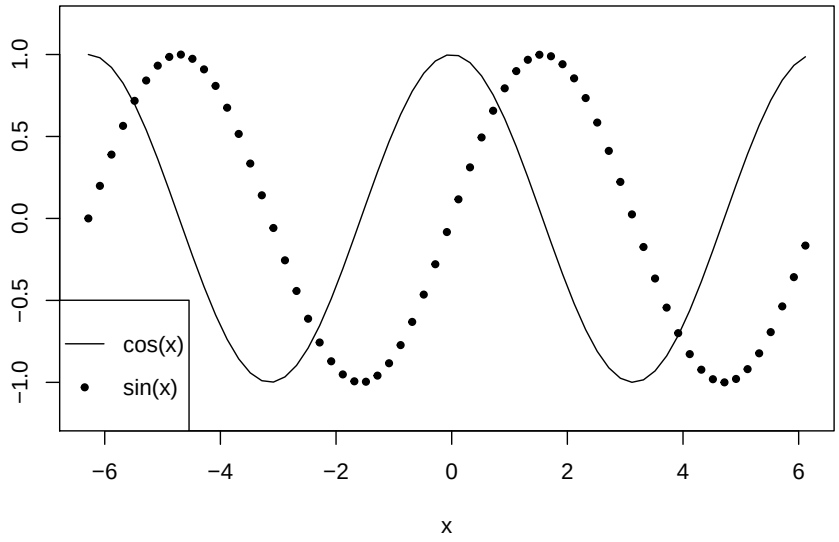


FIGURE 3.2 – Représentation des fonctions sinus et cosinus sur $[-2\pi, 2\pi]$.

Exercice 3.3. Tracer les courbes paramétrées suivantes pour $t \in [-10, 10]$:

1) $x(t) = \cos^2(t), \quad y(t) = \cos^3(t)\sin(t).$

2) $x(t) = \dfrac{t}{1+t^4}, \quad y(t) = \dfrac{t^3}{1+t^4}.$

3) $x(t) = \sin^3(t), \quad y(t) = \cos(t) - cos^4(t).$

4) $x(t) = [250 + a\sin(bt)]\cos(t), \quad y(t) = [250 + a\sin(bt)]\sin(t), \quad a = 220$ et $b = 6.$

Exercice 3.4. Représenter sur un même graphique les fonctions $x \mapsto x^2$ et $x \mapsto \sqrt{x}$ sur les intervalles $[0, 3]$ et $[0, 9]$ respectivement. Ajouter au graphique obtenu, la première bissectrice en pointillés pour apprécier la symétrie des deux courbes par rapport à cette dernière. Le résultat obtenu sera semblable à la figure 3.3.

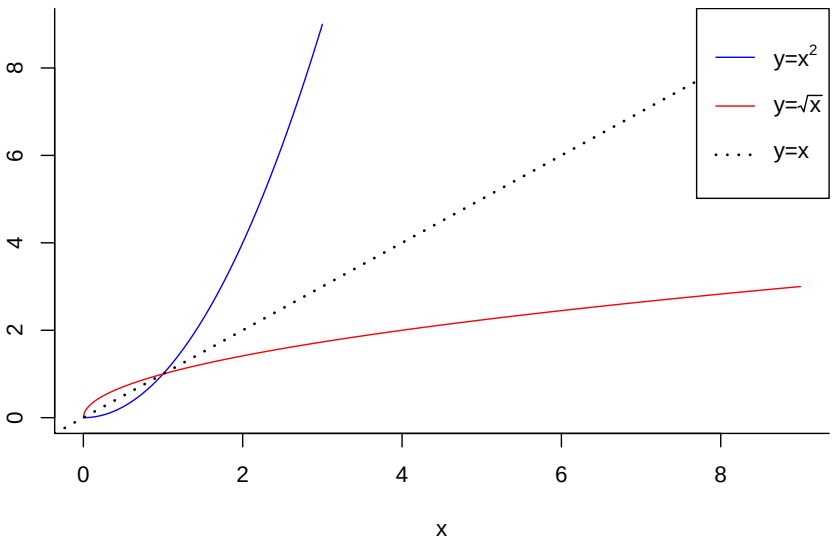


FIGURE 3.3

Chapitre 4

Programmation dans R

4.1 Structures de contrôle

4.1.1 Exécution conditionnelle

```
if(condition)
{
      # commandes a executer si condition est realisee
} else {
      # commandes a executer sinon
}
```

- `condition` est un variable logique ne contenant qu'une seule valeur. Si c'est un vecteur logique de longueur supérieure ou égal à 2, seul le premier élément est utilisé.
- Les accolades délimitent les deux groupes de commandes : celui où la condition est réalisée et celui où elle ne l'est pas.
- L'utilisation de `else` n'est pas obligatoire mais si nécessaire, la commande `else` doit être placée sur la même ligne que l'accolade fermante de `if`.

4.1.2 Boucles

`for(i in 1:10){print(i)}`	Boucle `for`.
`i=1` `while(i<=10){print(i); i=i+1}`	Boucle conditionnelle `while`. Nécessité de mettre à jour les compteurs afin d'éviter la boucle infinie.

`i=0`
`repeat{`
`i=i+1`
`if(i<=10){print(i)} else`
`{break}`
`}`
	Boucle conditionnelle `repeat`. La commande `break` est indispensable pour sortir de la boucle.

`break`	Sortie immédiate d'une boucle `for`, `while` ou `repeat`.

4.2 Créer ses propres fonctions et opérateurs

Une fonction est un objet particulier permettant de stocker un ensemble de commandes dans l'espace de travail. C'est un outil performant, pratique et généralement indispensable.

4.2.1 Création et exécution

4.2.1.1 Généralités

```
somme = function(a,b)
{
   return(a+b)
}
```

Fonction `somme` calculant la somme des deux nombres `a` et `b` donnés en arguments.

```
somme(2,5)
somme(a=2,b=5)
somme(b=5,a=2)
```

Appels de la fonction `somme` avec `a=2` et `b=5`.

```
args(somme)
```

Affiche les noms des arguments de la fonction `somme`.

4.2.1.2 Sorties multiples

```
som.prod = function(a,b)
{
   s=a+b
   p=a*b
   return(list(s=a+b, p=a*b))
}
```

Pour sortir plusieurs résultats en même temps d'une fonction, il faut créer une liste avec ces résultats. Ici, la fonction donne une liste de première composante `s` et de deuxième composante `p`.

4.2.1.3 Arguments par défaut

```
somme = function(a,b=0){
return(a+b)}
```

Par défaut `b` vaut 0. Si `b` n'est pas spécifié, il sera mis à 0.

```
somme(1,0) ; somme(1)
```

Commandes équivalentes.

4.2.1.4 Arguments indéfinis

L'objet `...` permet de spécifier des arguments non définis à une fonction.

`somme2 = function(...){`
`return(sum(...))`
`}`
	L'argument `...` est passé directement à la fonction `sum` appelée à l'intérieur de la fonction `somme2`.

`somme2(4,1)`	Résultat : 5.
`somme2(4,1,5)`	Résultat : 10.

4.2.2 Opérateurs

`"%+%"=function(a,b){max(a,b)}` `10 %+% 3`	Nouvel opérateur binaire. Appel de cet opérateur.

4.3 Exercices

Exercice 4.1. On considère le code suivant :

```
1 f = function(x)
2 {
3         if(x<=0){stop("L'argument ne doit pas etre negatif.")}
4         log(x)
5 }
```

1) Exécuter ce code pour des valeurs positives et négatives de l'argument.

2) Quel est le rôle la commande `stop` ?

Exercice 4.2. Programmer la fonction factorielle sous forme d'une fonction `R` récursive (c'est à dire qui s'appelle elle-même). Cette fonction sera appelée `fact` et retournera un message d'erreur lorsque l'argument n'est pas un entier naturel.

Exercice 4.3. Créer une matrice carrée A d'ordre 10 telle que

$$\begin{cases} a_{ii} = 2 & \forall i = 1, \dots, 10 \\ a_{i,i+1} = 1 & \forall i = 1, \dots, 9 \\ a_{i,i-1} = -1 & \forall i = 2, \dots, 10 \\ a_{ij} = 0 & \text{dans tous les autres cas.} \end{cases}$$

Exercice 4.4. Ecrire une fonction `matrix.power` qui prend en arguments une matrice carrée A et un entier naturel n et retourne la matrice A^n (on prendra $A^0 = I$). Un message d'erreur sera affiché si A n'est pas une matrice carrée ou si n n'est pas un entier naturel.

Exercice 4.5. Ecrire une fonction `solve.deg2` permettant de résoudre dans $\mathbb{R}$ l'équation $ax^2 + bx + c = 0$ où a, b et c sont des coefficients réels tels que $a \neq 0$. La fonction prendra trois arguments `a`, `b` et `c` et retournera une liste contenant la valeur du discriminant et un vecteur contenant la (ou les) solution(s). La fonction affichera un message concernant le nombre et la valeur des éventuelles solutions.

Exercice 4.6. On s'intéresse à la suite de terme général $S_n = 1 + 2^2 + 3^2 + \cdots + n^2$.

1) Ecrire deux fonctions `f1` et `f2` qui prennent en argument un entier naturel n et calculent S_n de deux façons différentes : l'une avec boucle et l'autre sans aucune boucle.
2) Que font les fonctions suivantes : `Sys.time` ? `difftime` ?
3) Proposer une méthode pour mesurer le temps d'exécution de la fonction `f1`.
4) Pour $n \in \{10,\ 10^2,\ 10^3,\ 10^4,\ 10^5,\ 10^6,\ 10^7\}$, mesurer les temps de calcul (en secondes) de S_n par les deux méthodes. Les résultats seront stockés sous la forme d'un tableau à trois colonnes correspondant respectivement aux valeurs de n, aux temps d'exécution de `f1` et `f2`. Que peut-on conclure ?

Chapitre 5

Statistique descriptive

5.1 Séries statistiques à une variable

5.1.1 Variable quantitative

5.1.1.1 Indicateurs numériques

Soit `x` un vecteur numérique à n composantes dont les composantes sont ici notées x_1, $\ldots$, x_n.

`min(x); max(x)`	Minimum et maximum des composazntes de `x`.
`sum(x)`	Somme des composantes de `x`
`mean(x)`	Moyenne des composantes de `x`
`var(x)`	Variance **corrigée** des composantes de `x`. Il est important de comprendre que, dans `R`, `var(x)` fournit plutôt la **variance corrigée** : $$s'^2_n = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x}_n)^2$$ au lieu de la variance usuelle $$s^2_n = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x}_n)^2.$$
`sd(x)`	Ecart-type corrigé $s'_n = \sqrt{s'^2_n}$.
`weighted.mean(c(0,1,2),c(3,2,5))`	Moyenne pondérée de 0, 1 et 2 avec poids respectifs 3, 2 et 5.

`median(x)`	Médiane des valeurs de `x`.
`quantile(x,probs=c(0.25,0.75))`	Les quantiles d'ordre 0.25 et 0.75 respectivement (encore appelés quartiles).
`summary(x)`	Résumé des indicateurs numériques (moyenne, quartiles, minimum, etc.) sur le vecteur `x` (fonction générique valable pour d'autres classes).
`table(x)`	Tableau de contingence de `x`.

`cumsum(table(x))`
`rev(cumsum(rev(table(x))))`
`prop.table(table(x))`
`table(x)/length(x)`
	Effectifs cumulés croissants.
	Effectifs cumulés décroissants.
	Fréquences.
	Permet aussi d'obtenir les fréquences.

`h=hist(x,plot=FALSE)`	La fonction `hist` de gérer les variables statistiques continues groupées (définies par intervalles). L'option `plot=FALSE` permet d'éviter de tracer l'histogramme.
`h=hist(x,breaks=c(0,1,3,4), plot=FALSE)`	Découpage en trois classes $[0,1]$, $]1,3]$ et $]3,4]$. Par défaut, les classes sont semi-fermées à droite et la première classe est fermée.
`h=hist(x,breaks=c(0,1,3,4), right=FALSE)`	Découpage en trois classes $[0,1[$, $[1,3[$ et $[3,4]$. Avec l'option `right=FALSE`, les classes sont semi-fermées à gauche et c'est la dernière classe qui est fermée.
`h=hist(x,breaks=3)`	Découpage des composantes de `x` en trois classes de même amplitude.
`h$counts`	Les effectifs des classes.
`h$mids`	Les centres des classes.

5.1.1.2 Représentation graphique

`hist(x)`
`hist(x,freq=FALSE)`
 	
`hist(x,breaks=c(1,2,7))`
 	
`hist(x,breaks=5)`
	Histogramme en effectif du vecteur `x`.
	Histogramme en fréquence (en densité et non en effectif).
	Histogramme en deux classes : [1,2] et]2,7].
	Histogramme en 5 classes de même amplitude.

`boxplot(x)`	Affiche la boîte à moustaches pour un vecteur numérique `x`. NB : Si `x` est une liste ou un objet de classe `data.frame`, cette commande produit sur le même graphique autant de boîtes à moustaches que de composantes.
`barplot(x)`	Affiche des barres de hauteurs respectivement égales à chaque composante du vecteur `x` (diagramme en barres).
`plot(table(x),lwd=5,lend=2)`	Diagramme en bâtons. L'option `lwd` représente la largeur de chaque trait. L'option `lend` (line end) permet de préciser la forme des extrémités (forme carrée ou circulaire).
`plot(x,y,type="h")`	L'option `type="h"` permet aussi de construire un diagramme en bâtons dont `x` contient les modalités et `y` contient les effectifs.

5.1.2 Variables qualitatives

`x=factor(c("y","n","y","n","y"))`
`table(x)`
`prop.table(table(x))`
	Tableau de contingence pour le facteur `x`.
	Fréquences.

`pie(c(0.2,0.5,0.3))`	Affiche un camembert (diagramme circulaire) dont les proportions sont respectivement de 20%, 50% et 30%.

5.2 Séries statistiques à deux variables

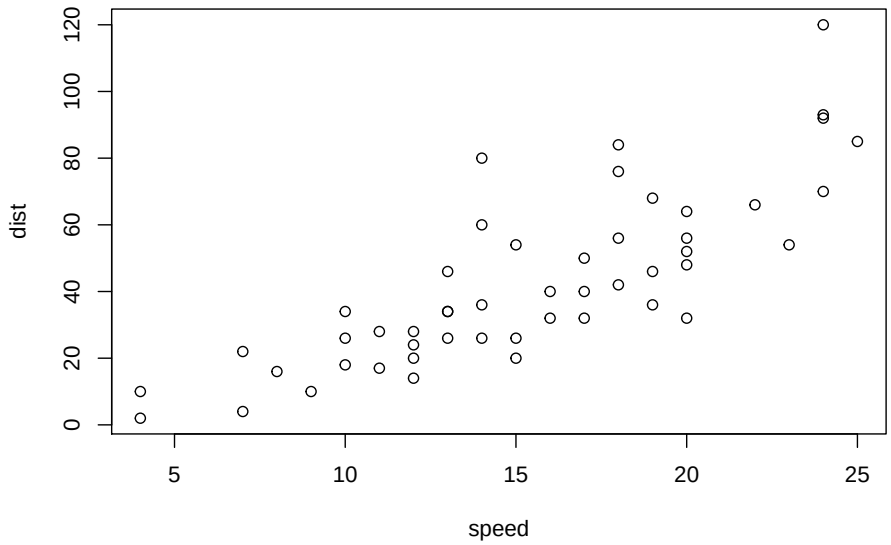
`table(x,y)`	Tableau de contingence entre les facteurs (ou les variables statistiques discrètes) `x` et `y`.
`cov(x,y)`	Covariance corrigée (divisée par $n-1$ et non n) des variables quantitatives `x` et `y` de n observations.
`plot(x,y,...)`	Représentation graphique des variables quantitatives `x` et `y` de même nombre d'observations. Voir chapitre 3 pour les détails.

5.3 Ajustement linéaire

Illustrons la notion d'ajustement linéaire à l'aide du jeu de données `cars` automatiquement chargé en mémoire.

`x=cars$speed; y=cars$dist`	`cars` est un objet de classe `data.frame` à 50 lignes et 2 colonnes donnant les vitesses de $n = 50$ voitures (en mph [1]) et les distances de freinage correspondantes (en ft [2]).

```
> plot(cars)
```



`reg=lm(y~x)`	Ajustement linéaire $y = ax + b$. L'estimation des coefficients a et b est faite par la méthode des moindres carrés.
`summary(reg)`	Affiche un résumé des informations sur le modèle `reg` : on peut y extraire notamment les coefficients (la constante est appelée *intercept*) et le coefficient de détermination R^2.

```
> x=cars$speed; y=cars$dist
> reg=lm(y~x)
> summary(reg)


Call:
lm(formula = y ~ x)

Residuals:
Min      1Q  Median      3Q     Max
-29.069  -9.525  -2.272   9.215  43.201

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -17.5791     6.7584  -2.601   0.0123 *
x             3.9324     0.4155   9.464 1.49e-12 ***
```

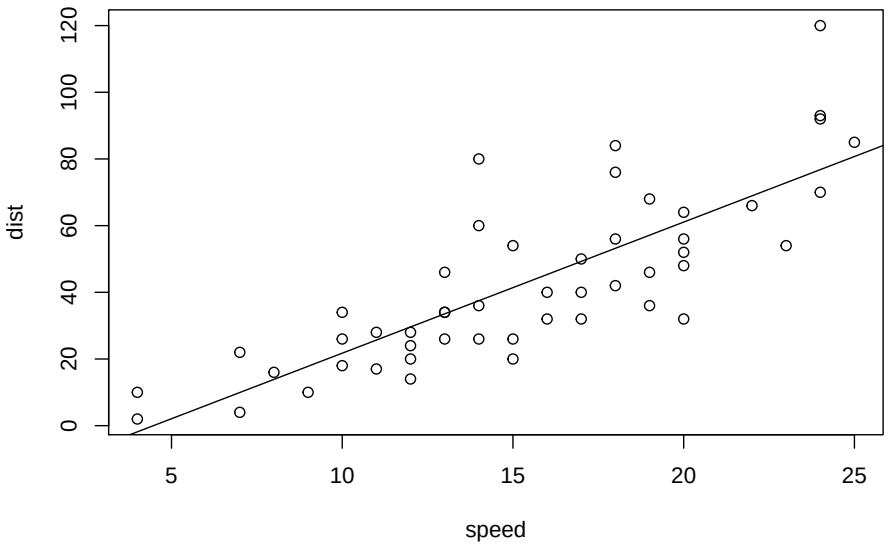
```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 15.38 on 48 degrees of freedom
Multiple R-squared:  0.6511,        Adjusted R-squared:  0.6438
F-statistic: 89.57 on 1 and 48 DF,  p-value: 1.49e-12
```

- La deuxième colonne du tableau `Coefficients` donne les estimations des coefficients par la méthode des moindres carrés. Ici $a = 3.9324$ et $b = -17.5791$
- La dernière colonne contenant les étoiles donne le seuil de significativité des coefficients, notion qui dépasse le cadre de ce cours.
- La qualité de l'ajustement sera mesurée grâce au `Multiple R-squared` (ici $R^2 = 0.6511$). Plus R^2 est proche de 1, meilleur est l'ajustement.

`names(reg)` `reg$coefficients`	Affiche la liste des composantes de `reg`. Accès aux coefficients du modèle.
`x.new=data.frame(x=c(5,21))` `predict(reg,x.new)`	Saisie de 2 nouvelles vitesses et prévision des distances de freinage correspondantes en utilisant le modèle `reg`.

On peut tracer la droite de régression :

```
> abline(reg$coefficients)
```



2. Le mille par heure (mph) est une unité anglo-saxonne de mesure de vitesse (1 mph = 1.609344 km/h).

2. Une unité de longueur anglo-saxonne appelée pied en français (1 ft = 0.3048 m).

5.4 Exercices

Exercice 5.1. On a relevé le nombre d'enfants de 40 couples. Les résultats obtenus sont donnés dans le tableau 5.1.

TABLE 5.1 – Nombre d'enfants pour 40 couples.

Nombre d'enfants x_i	0	1	2	3	4
Nombre de couples n_i	12	20	5	2	1

1) Saisir les données du tableau 5.1 sous la forme de deux vecteurs `nb.enfants` et `nb.couples`.
2) Représenter les données par un diagramme en bâtons.
3) Calculer la moyenne, la variance et l'écart-type du nombre d'enfants.
4) Calculer les fréquences et les fréquences cumulées croissantes. En donner des exemples d'interprétation.

Exercice 5.2. On a noté l'âge (arrondi à l'année près) de 48 salariés d'une entreprise. La série statistique brute (données obtenues pendant l'enquête) est donnée ci-dessous :

43	29	57	45	50	29	37	59	46	31	46	24
33	38	49	31	62	60	52	38	43	26	41	52
60	49	52	41	38	26	37	59	57	41	29	33
33	43	46	57	46	33	46	49	57	57	46	43

1) Saisir les données dans un vecteur `ages`.
2) Calculer la moyenne et la variance des âges.
3) Procéder à la distribution de l'âge des 48 salariés en considérant les classes d'âge $[20, 30[$, $[30, 40[$, $[40, 45[$, $[45, 50[$, $[50, 55[$, $[55, 60[$ et $[60, 65]$. Donner les effectifs et les fréquences par classe.
4) Tracer l'histogramme en fréquences des âges. Expliquer comment a été obtenue la hauteur du premier rectangle de l'histogramme.
5) Calculer à nouveau la moyenne et la variance en considérant uniquement la répartition en classes.
6) Comparer les réponses aux questions 2 et 5. Que peut-on conclure ?

Exercice 5.3. Le recensement général de la population et de l'habitat (RGPH) du Togo en 2010 a donné les résultats suivants (voir Tableau 5.2).

1) Saisir les données du tableau dans une matrice dont les lignes et les colonnes seront nommées.
2) Calculer l'effectif total de la population togolaise en 2010.
3) Calculer le nombre total d'hommes et de femmes ainsi que l'effectif total de la population par région.
4) Calculer les proportions d'hommes et de femmes pour la population totale puis pour chaque région.

TABLE 5.2 – RGPH 2010 (source : http://data.gouv.tg).

Région	Nb. d'hommes	Nb. de femmes
Lomé	402 172	437 394
Maritime*	846 182	914 207
Plateaux	678 191	696 974
Centrale	308 443	309 428
Kara	376 111	393 829
Savanes	397 996	430 228

* Lomé - Commune exclus.

5) Calculer les proportions de chaque région par rapport à la population totale puis les représenter à l'aide d'un diagramme circulaire sur lequel vous indiquerez à côté de chaque région son nom et la proportion de population correspondante.

6) Représenter les diagrammes en barres de la population par région et par sexe (voir figure 5.1). Les noms des couleurs sont `lightblue` et `tomato`.

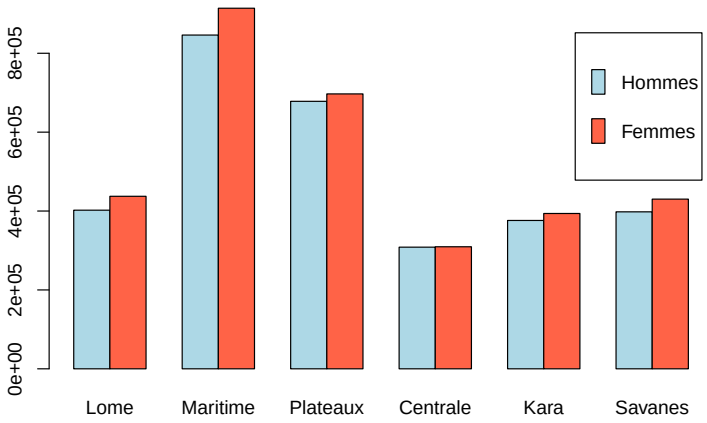


FIGURE 5.1

Exercice 5.4. L'observation du prix de la tomate en fonction des quantités vendues sur un marché a donné les résultats suivants :

Quantités x (kg)	10	20	35	50	70	90	110	130
Prix y du kg (kFCFA)	5	3.75	2.75	2.25	1.75	1.25	0.8	0.5

Ainsi, une quantité de 35 kg de tomates est vendue au prix de 2750 FCFA le kg.

1) Représenter graphiquement le nuage de points.

2) (a) Déterminer l'équation de la droite d'ajustement linéaire $y = ax + b$ qui permet d'expliquer le prix au kg par la quantité achetée.

(b) Prévoir le prix d'un kg de tomates pour un achat de 140 kg.

3) (a) Chercher maintenant un ajustement par une fonction logarithme de la forme $y = a\ln(x) + b$. On pourra poser $z = \ln(x)$ et se ramener à un ajustement linéaire $y = az + b$.

(b) Prévoir le prix du kg pour un achat de 140 kg.

4) Représenter à nouveau le nuage de points et y ajouter les deux ajustements. Lequel de ces deux modèles vous semble le plus judicieux ?

Chapitre 6

Probabilités et simulations

6.1 Lois de probabilités usuelles

Toutes les lois de probabilités usuelles sont disponibles dans `R`. Chaque loi est identifiée par une abréviation :

Table 6.1 – Liste des lois de probabilités usuelles

Loi	**Abréviation**
Beta	`beta`
Binomiale (y compris Bernoulli)	`binom`
Binomiale négative	`nbinom`
Cauchy	`cauchy`
Exponentielle	`exp`
Fisher-Snedecor	`f`
Gamma	`gamma`
Géométrique	`geom`
Hypergéométrique	`hyper`
Khi-deux	`chisq`
Log-normale	`lnorm`
Logistique	`logis`
Normale	`norm`
Poisson	`pois`
Student	`t`
Uniforme	`unif`
Weibull	`weibull`

Pour chaque loi d'abréviation `xxx`, quatre (04) fonctions sont disponibles, identifiées par un préfixe :

`dxxx`	Fonction densité de la loi `xxx`.
`pxxx`	Fonction de répartition de la loi `xxx`.
`qxxx`	Fonction quantile de la loi `xxx`.
`rxxx`	Génération de nombres aléatoires suivant la loi `xxx`.

Le lecteur est invité à consulter l'aide pour plus de détails sur les paramètres des différentes lois. Nous donnons ci-dessous un exemple de loi discrète et un exemple de loi continue.

`dbinom(3,20,1/2)`	Probabilité individuelle de la loi binomiale $\mathcal{B}(20, \frac{1}{2})$ évaluée en 3 c'est-à-dire $\mathbb{P}(X = 3)$ lorsque $X \rightsquigarrow \mathcal{B}(20, \frac{1}{2})$.
`pbinom(3,20,1/2)`	Fonction de répartition de la loi $\mathcal{B}(20, \frac{1}{2})$ évaluée en 3 c'est-à-dire $\mathbb{P}(X \leqslant 3)$.
`qbinom(0.9,20,1/2)`	Quantile d'ordre 0.9 de $\mathcal{B}(20, \frac{1}{2})$.
`rbinom(10,20,1/2)`	Simulation d'un vecteur de 10 réalisations indépendantes de $\mathcal{B}(20, \frac{1}{2})$.
`dnorm(1,0,2)`	Densité de la loi normale $\mathcal{N}(0, 2)$ évaluée en 1.
`pnorm(1,0,2)`	Fonction de répartition de la loi normale $\mathcal{N}(0, 2)$ évaluée en 1.
`qnorm(0.9,0,2)`	Quantile d'ordre 0.9 de $\mathcal{N}(0, 2)$.
`rnorm(10,0,2)`	Simulation d'un vecteur de 10 réalisations indépendantes de $\mathcal{N}(0, 2)$.
`replicate(n,exp)`	Exécute `n` fois le code `exp` et stocke les résultats sous forme de vecteur (si `exp` renvoie un nombre) et sous forme de matrice (si `exp` renvoie un vecteur).

La fonction `replicate` permet d'éviter l'usage de boucles. Par exemple, le code suivant simule 5 fois la génération de 1 000 valeurs suivant la loi normale centrée réduite et calcule la moyenne à chaque fois.

```
> replicate(5,mean(rnorm(1000)))
```

```
[1] -0.01514964 -0.04786793 -0.06151126 -0.07324169  0.01370721
```

6.2 Tirages avec ou sans remise

`sample(1:10,100,replace=TRUE)`	Tirage avec remise de 100 éléments parmi les nombres $1, \dots, 10$. L'option `replace=TRUE` précise que le tirage se fait avec remise. Par défaut, si rien n'est précisé, `replace=FALSE`.
`sample(1:10); sample(10)`	Commandes équivalentes désignant une permutation des entiers naturels 1, 2, ..., 10.
`sample(c("A","B","C"),size=5, replace=TRUE,prob=c(0.2,0.5,0.3))`	Tirage avec remise de 5 lettres parmi les lettres A, B et C. Si l'on ne veut pas un tirage équiprobable, on précise les probabilités de tirage dans le vecteur `prob`.

6.3 Exercices

Exercice 6.1. Soit X une variable aléatoire de loi binomiale de paramètres $n = 50$ et $p = 1/3$. Calculer les probabilités suivantes :

$$\mathbb{P}(X = 1) \quad ; \quad \mathbb{P}(X \leqslant 5) \quad ; \quad \mathbb{P}(X \geqslant 15) \quad ; \quad \mathbb{P}(X \notin \{15, 3, 4, 10\}) \quad ; \quad \mathbb{P}(X \in 2\mathbb{N}).$$

Exercice 6.2. Soit X une variable aléatoire de loi normale standard $N(0, 1)$.

1) Calculer les probabilités suivantes : $\mathbb{P}(X \leqslant 1)$, $\mathbb{P}(X \geqslant 2.6)$ et $\mathbb{P}(0.5 < X \leqslant 1)$.
2) Calculer les quantiles d'ordre 0.95 et 0.975.
3) Représenter graphiquement la densité et la fonction de répartition de la loi de X.

Exercice 6.3. Soit X une variable suivant la loi normale $\mathcal{N}(3, 2)$. Calculer les probabilités suivantes : $\mathbb{P}(X < 4)$, $\mathbb{P}(X < -1)$, $\mathbb{P}(X > 1)$, $\mathbb{P}(-3 < X < 4)$.

Exercice 6.4.

1) Créer un vecteur x de taille $n = 1000$ et dont les éléments suivent la loi normale centrée réduite.
2) Représenter l'histogramme des éléments de ce vecteur et superposer sur ce graphique la fonction de densité de la loi normale centrée réduite. Que remarquez-vous ?

Exercice 6.5.

1) En vous aidant de la fonction `sample` et du vecteur `piece=c("F","P")` (dont les composantes représentent FACE et PILE), écrire une fonction `lancer.piece(n)` qui simule n lancers d'une pièce de monnaie équilibrée.
2) Simuler $n = 5\,000$ lancers de ladite pièce. Calculer la proportion de F (vous devriez trouver une valeur proche de 0.5).

Exercice 6.6. Un ami vous propose le jeu suivant. Vous lancez chacun 2 fois un dé supposé parfaitement équilibré dont les faces sont numérotées de 1 à 6. A chaque lancer, si le résultat est 5 ou 6, vous gagnez 3 €; si le résultat est 4, vous gagnez 1 € et si c'est 3 ou moins, vous perdez 2.5 €.

1) Ecrire une fonction `jeu` permettant de simuler le jeu.
2) Simuler ce jeu 1 000 fois. Résumer les résultats à l'aide d'un tableau de contingence et représenter ce tableau à l'aide d'un diagramme en bâtons.
3) Ce jeu serait-il avantageux pour vous ? Justifier la réponse.

Références bibliographiques

[1] Bertrand, F. and Maumy-Bertrand, M. (2018). *Initiation à la Statistique avec R*. Dunod, Paris, 3ème edition.

[2] Biernacki, C. (2010). Logiciel R - Tutorial avec applications statistiques. Notes de cours, Université Lille 1.

[3] Broc, G. (2016). *Stats faciles avec R*. De Boeck Superieur.

[4] Lafaye de Micheaux, P., Drouilhet, R., and Liquet, B. (2014). *Le logiciel R : Maîtriser le langage - Effectuer des analyses statistiques*. Springer, 2ème edition.

[5] Paradis, E. (2005). R pour les débutants. URL http://cran.r-project.org/doc/contrib/Paradis-rdebuts_fr.pdf.

[6] Venables, W. N., Smith, D. M., and the R Core Team (2014). *An Introduction to R*. Notes on R : A Programming Environment for Data Analysis and Graphics Version 3.0.3 (2014-03-06).